Manual-Machbase 5.0

Introduction

- Introduction of Machbase
- Machbase Features
- Introduction of Machbase Products

Table Types

- Tag Table
- Log Table
- Lookup Table
- Volatile Table
- STREAM
- Backup & Mount

Installation

- Package Overview
- Linux Installation
- Windows Installation
- License Installation
- Cluster Edition Installation

Upgrade

- Linux Upgrade
- Windows 업그레이드 Windows Upgrade
- Cluster Edition Upgrade

Config/Monitoring

- Meta Table
- Virtual Table
- Property
- Property (Cluster)

Tools

- Console
- Collector
- Visualization Tool
- Cluster management tools

SQL Reference

- Datatypes
- DDL
- DML
- SELECT
- SELECT Hint
- User Management
- Functions
- System/Session Management

SDK

- CLI/ODBC
- CLI/ODBC Examples
- JDBC
- Python
- RESTful API
- .NET Connector
- Tag Table RESTful API

Overview

Introduction of Machbase

Machbase is a new generation columnar time series database that ingests and stores large amounts of log or time stamped data. Machbase is designed for collecting, in real time, "machine or sensor data" from various IoT environments while simultaneously allowing the data to be queried and analyzed using standard SQL and APIs at very high speeds. Machbase supports any IoT data architecture, and works extremely well on limited edge compute devices, or gateway/fog platforms, as well as cloud/cluster implementations.

Machbase solves the problem of storing and processing massive amounts of sensor data that could not be solved by existing current big data solutions, while safeguarding the future emergence of new sensor data requirements through various capabilities and functions.

Index

The Emergence of New Data

Through the recent development of applying big data and real-time analytics to the internet of Things (IoT) domain and with the promise of Machine Learning (ML), various and vast amounts of data are being captured and accumulated. With the future of 5G systems entering the market, machine to machine data traffic will continue to explode.

Therefore, by analyzing these data in real-time, new applications, better efficiencies, greater reliability and fewer failures will be achieved.
In particular, the number of source devices, such as edge devices or sensors, have dramatically increased in the last few years, and as a result the generated data from these devices have also increased exponentially.
However, traditional, and even big data processing software are not providing adequate solutions to meet the volume and performance needs.
The reasons why the current, conventional solutions are not suitable for the current data processing are summarized as follows:

First, the rate of data generation is increasing exponentially.

As the number of data sources grows and the types of information that need to be processed increases, the rate at which the server can receive and store the data is entirely different from before.
There is no software solution optimized for storing tens of thousands to hundreds of thousands of data per second aside from saving it as a plain text file in the file system.

Second, the demand for real - time data analysis is increasing in proportion to the rate of data generation.

Using big data technics to help decision making is very important; however, existing solutions are not technically advanced enough to index hundreds of thousands of data per second and deliver results to the user, through search and analysis, in real-time.

Finally, as previously mentioned, the characteristics of "sensor data" is completely different.

Conventional databases have architectures that are inadequate for processing sensor-type machine data, and to address this, a new technological approach is needed.
For this reason, Machbase has been newly developed with the best architecture to handle new "sensor-type machine data" and is the only database solution that can store, process, and analyze real-time data.

Types of Sensor Data

ID	Time	Data
Temperature01	2020-01-01 00:00:00	20
Pressure01	2020-01-01 00:00:00	0.98

The figure above shows typical sensor data divided into three types. The properties and characteristics of each types are as follows.

ID

This value represents a symbol and a number indicating the uniqueness of the device (source code) where the corresponding machine data is generated.

It is the serial number of the machine or sensor and is represented as a 32-bit or 64-bit integer.

TIME

This value represents the time when the corresponding machine data occurred. This time stamp has the tendency to increase continuously and is usually in units of a second but can also be as fine as in nanoseconds.

DATA

This value is binary data, mainly in the form of integer type, real number type, or IP address value. Typically, this domain includes numeric values such as temperature, acceleration, and brightness from a specific sensor, or fixed data of 4-byte or 16-byte similar to IPv4 or IPv6. In certain applications the value can be in binary format as from camera or audio devices.

By providing a Tag Table, Machbase offers optimized architecture that can receive hundreds of thousands of sensor data per second.

Sensor Data Structure in PLC

In actuality, sensor data used in Programmable Logic Control (PLC) is not always represented by the above three types.

In most cases, a large number of sensor data are stored in a collected form as shown below.
Therefore, the following types of data can be accepted together and should be easily converted.

data from PLC

Time	SN01M	SN02M	SN03M	SN04M	SN05M	SN06M	SN07M	SN08M	SN09M	SN10M
04:01:56.005	11.1	1	0	0	0	1	0	0	0	0
04:01:56.057	11.3	1	0	0	0	1	0	0	0	1
04:01:56.109	11.1	1	0	1	0	1	0	1	1	0
04:01:56.161	12.3	1	0	0	0	1	0	0	0	1
04:01:56.213	9.1	1	1	0	0	1	0	0	0	0
04:01:56.266	0.9	1	0	0	1	1	0	0	0	0
04:01:56.319	8.9	1	0	1	0	1	0	1	0	1
04:01:56.370	1.3	1	0	0	0	1	0	0	0	0
04:01:56.422	33.1	1	0	0	0	1	0	0	0	0
04:01:56.474	3.3	1	0	0	0	1	0	0	0	0
04:01:56.526	5.6	1	0	0	1	1	0	0	0	1
04:01:56.578	5.5	1	0	0	0	1	0	0	1	0
04:01:56.630	4.5	1	0	0	0	1	0	0	0	0
04:01:56.682	5.3	1	0	0	0	1	0	0	0	0
04:01:56.733	1.2	1	0	0	0	1	0	0	0	1
04:01:56.785	3.4	1	0	0	1	1	0	0	0	0
04:01:56.838	11.3	1	0	1	0	1	0	1	0	0
04:01:56.890	11.2	1	0	0	0	1	0	0	0	0
04:01:56.942	9.9	1	0	0	0	1	0	0	0	1
04:01:56.994	8.4	1	0	0	0	1	0	0	0	0
04:01:57.046	8.4	1	0	1	0	1	0	0	0	0
04:01:57.097	1.2	1	0	0	1	1	0	1	1	1
04:01:57.149	1.3	1	0	0	0	1	0	0	0	0
04:01:57.200	11.2	1	0	0	0	1	0	0	0	0
04:01:57.252	3.1	1	0	0	0	1	0	0	0	0

Machbase provides a Log Table, which also provides a structure to store the above PLC data.

This table also provides the ability to enter and analyze tens of thousands of data per second.

Machbase Features

Support for Various Table Structures

Machbase provides four table types for users according to one's usage. (Tag, Log, Volatile, Lookup)

This is because client requirements for storing sensor data are very diverse and any one business does not have just one specific data pattern.
Therefore, it's important to understand these business requirements and select the appropriate tables for them.
The table below shows the characteristics of each table.

Index

Table Type	Tag Table	Log Table	Volatile Table	Lookup Table
PURPOSE	Optimized for processing sensor time series data in the form of *<sensor name, time, sensor value>*	Optimized for processing PLC log time series data (text included)	Real-time processing of volatile memory data	Manages master data that can be stored permanently
DESCRIPTION	Used when storing sensor data at high speed, extracting corresponding data at high speed, or creating statistical tables in real-time Mainly stores real-time sensor data	Used when storing log data including text and analyzing it in the form of general DBMS Mainly stores historical user data	Used when Insert, Delete, Update, Select is required for memory-based performance (tens of thousands per second) **All data is lost when the system is shut down.** Mainly used for key-value based monitoring.	Used to permanently store user-editable master data. SELECT has high-speed performance, but INSERT, UPDATE, and DELETE provide disk-based performance.
TABLE STRUCTURE	*<Sensor name, time, sensor value>* is the basic type, with the ability for assigning additional columns.	Any schema possible	Any schema possible (Primary Key can be assigned)	
INSERT (INPUT) PERFORMANCE	Millions per second	Millions per second	Tens of Thousands per second	Hundreds per second
SELECT	Sensor Name + Limited Time Range	All inquiries possible		
DELETE	Real time deletion of data before an arbitrary point	Real time deletion of arbitrary point / interval data	Primary Key Record Delete Support (※ Primary Key Designation Required)	
UPDATE	Not supported (※ Only Metadata column is editable)	Not supported	Primary Key Update Support (※ Primary Key Designation Required)	
STORAGE SIZE LIMITS	Disk limit		Memory limit	
INDEX STRUCTURE	Three-step partitioning real-time index (※ default creation)	LSM index	Red / Black memory index	
STREAM SUPPORT	arget only (save target)	Both source / target (read and save target)	Not possible	
CONSIDERATIONS	Consider enough storage to erase historical data	Consider as temporary storage for Tag input	Consider memory limit	

Hardware Support for Various Sizes

Machbase provides various product editions according to user environments as listed below.

Edge Edition

This product runs on small Edge devices running on ARM or Intel ATOM CPUS.

However, even in such small devices, Machbase can be useful for storing and filtering tens of thousands of sensor data per second.
It is mainly required for storing various sensor data at the terminal stage of robots and factory production lines, buildings, etc. at high speed and high capacity.

Fog Edition

This product is used to achieve high-speed data processing on a single server.

It runs on Windows or Linux operating systems based on Intel x86 CPU and provides very fast sensor data storage and analysis that other DBMSs can not provide.
In most cases, it is used to store real-time data input from hundreds or more of edge devices and to perform secondary analysis.

Cluster Edition

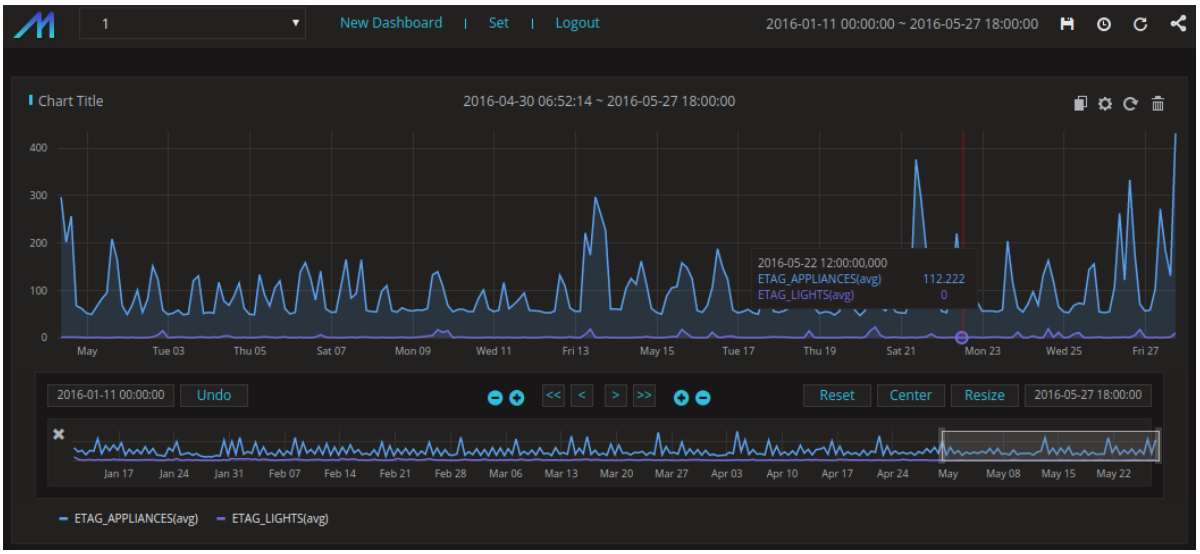
This product was developed for the purpose of storing large-scale sensor data for large manufacturing plants.

A number of physical servers operate in clusters to store more than 10 million data per second in semiconductor or display, power generation, and steel production processes.
It is used in an increasingly data-rich environment where data capacity needs to be continuously maintained.

Tag Analyzer: Data Visualization Solution Support

Machbase provides real-time visualization of hundreds of millions of sensor data stored in Machbase (since Version 5).

In other words, an arbitrary tag ID is designated, and the trend chart for the period in which the ID is input can be instantaneously checked on the web-based basis.
In addition, it provides not only simple tag data but also a statistical chart during that period, so statistical analysis is possible beyond simple visualization.



Write Once, Read Many

Sensor data is rarely edited or deleted once it is entered into the database.

Therefore, Machbase is designed so that once the key time series data is inputted to maximize the characteristics of the machine data, an UPDATE can not occur.
Once the log data has been entered, it cannot be altered or deleted by malicious users, so there should be no concerns.

Lock-free Architecture Support

The most important aspect in sensor data processing is that data input, update, delete operation and read operation should be processed as independent as possible without conflicts.

Because of this, Machbase is designed not to allocate any locks for the SELECT operation, and it is designed with a high performance structure that never conflicts with the operation of input or deletion changes.
Therefore, even when hundreds of thousands of data are entered and some of them are deleted in real time, the SELECT operation can speed up statistical operations on millions of records.

High Speed Data Storage

Machbase provides data storage performance that is exponentially faster than conventional databases. Even if there are many indexes in a specific table, data can be received from at least 300,000 to at most 10 million per second.

This is possible because Machbase is designed to optimize time series data.

STREAM Function Support

Starting from Machbase Version 5, Edge and Fog Editions provide STREAM functionality to support real-time data filtering.

This STREAM performs a condition evaluation on real-time data input in DBMS at high speed and transmits the result to an arbitrary table.
This function is very useful for generating a warning when the value of a certain sensor exceeds a specific range or real time evaluation of internally input data is needed.

Configuring Real-Time Index

Machbase innovatively improves on conventional database structure (where the more indexes you have the slower your data insert performance is) and can build indexes in near real-time, even with hundreds of thousands of data inserts per second.

This feature is a key technology for analyzing time series data, such as machine data, because it provides a powerful functional foundation for instant retrieval of actual data as it occurs.

Real-Time Data Compression

The characteristic of time series data such as machine data is that data is generated constantly. This inevitably means that not only will the storage space of the database becomes eventually inadequate, but it will not have enough data to process.

In particular, although conventional databases input data at a high speed, as the number of indexes increases, the occupied data space also greatly increases. Therefore, conventional databases are quite unsuitable for storing and analyzing machine data.
Machbase uses two innovative real-time compression techniques to compress and store up to a hundred times more data without any setbacks in performance.

Logical Real-Time Data Compression Technology Support

First, Machbase supports logical real-time data compression technology.

This is based on the data redundancy of the machine data derived from a column-type database. It is an innovative technique to reduce the data storage space by coding redundant data as the number of data having the same value increases, which allows high redundancy data to be compressed hundreds of times the original amount.

Physical Data Compression Technology (Patented Technology)

The second is Machbase's patented physical data compression technology.

This is a technology that reduces the amount of physical data to be stored by dividing a physical data block to be stored in a disk into a predetermined size partition, compressing it into a disk separately, and further reducing the I/O cost caused by the system. This helps to increase the efficiency of the storage space by compressing the actual logically compressed data once more.

Outstanding Query Performance

The innovative and technological superiority of Machbase is that the search and statistical analysis of millions or tens of millions of previously stored historical data is very fast, even with the simultaneous input of hundreds of thousands of data per second.

This is possible because of Machbase's own indexing technology that provides superior performance for both insertion and analysis, and will play a key role in real-time business decision making.
Unlike conventional databases, Machbase can process two or more indexes in a single query, which can be expected to perform several times faster when processing data in parallel.
The following is an example of using two or more indexes in a single query.

```
SELECT * FROM table1 WHERE c1 = 1 and c2 = 2;
```

Time Series Data Characteristics SQL Syntax Support

In the case of sensor data, the newest data is several times more valuable than the older data, and also the "access frequency" of the latest data is characterized as being several times more compared to old data.

For this reason, Machbase supports time series data features through two types of tables: Tag and Log.

Log Table

The log table supported by Machbase has the following features.

First, it automatically saves input time

Whenever a record is stored in the database, a timestamp in nanoseconds is stored as a field called _arrival_time.
This means that all records stored by Machbase can be searched for or given condition on a time basis.

Second, it prioritizes lookup of recent data

When retrieving data, the latest time is output before the old time. That is, when SELECT is performed, the latest data is output first.
The result is the descending sort based on the _arrival_time column mentioned earlier.

Third, the DURATION keyword

The DURATION keyword is provided to enable quick lookup of specific time range data based on input time.
In the case of machine data analysis, these characteristics are provided at the SQL level because they often specify a specific time range.
This makes it easy to analyze data without stating "where" clause to complex time operators.

```
-- Example 1) View data statistics from 10 minutes ago
SELECT SUM(traffic) FROM t1 DURATION 10 MINUTE;

-- Example 2) View data statistics for 30 minutes from 1 hour ago
SELECT SUM(traffic) FROM t1 DURATION 30 MINUTE BEFORE 1 HOUR;
```

Tag Table

The tag table that is supported from Machbase 5.0 has the following features.

First, high-speed TAGID / time condition search performance

The tag table is excellent at any time and any ID based search performance.
It boasts ultra-fast data extraction performance that can not be achieved with existing RDBMSs, ensuring the same speed even when billions of sensor data are stored.

Second, the high-speed tag data input

The tag table supports high-speed data input.
As in the previous log table, data can be input without difficulty even with the input of hundreds of thousands of sensor data per second.

Third, real-time statistics function

The tag table supports real-time statistics function.
Machbase automatically generates five types of statistics in real time for the data stored in this tag table and provides a function to access them in real time.

Supports Text Search Function

One of the most important practical uses for users to store and use logarithmic time series data is to determine if a specific event occurred at a particular point in time.

Time-series data processing is possible at a specific point in time, but in most cases the occurrence of a specific event requires searching for a specific "word" in a text field stored in a particular column.
However, in a traditional database, in order to search for a word in a specific field, the exact match or LIKE clause is used to check the condition of some initial character through B + Tree. In most cases, this results in a very slow response.
That's why searching for a particular word in a conventional database is very weak and frustrating.
On the other hand, with Machbase, the SEARCH keyword based on the log table is provided to enable real-time word search.
This makes it possible to quickly search for any error text generated from the equipment.

```
-- Example 1) Output record containing Error or 102 in msg field
SELECT id, ipv4 FROM devices WHERE msg SEARCH 'Error' or msg SEARCH '102';

-- Example 2) Output record containing Error and 102 in msg field
SELECT id, ipv4 FROM devices WHERE msg SEARCH 'Error 102';
```

Optional Deletion Support

In the case of sensor data, it is true that deletion operations are rarely generated after insertion.

However, with embedded devices, there is a limited storage space that is not carefully managed by users.
In this case, if a 'disk full' occurs or a failure occurs due to machine data, the company could suffer a lot of damage.
Machbase provides the ability to delete records for a given condition in this environment.
Therefore, embedded developers can use CRON or periodic programs to easily manage Machbase to not keep data over a certain size.

For Log Tables

The following commands are supported:

```
-- Example 1) Delete oldest last 100.
DELETE FROM devices OLDEST 100 ROWS;

-- Example 2) Delete all but 1000 most recent.
DELETE FROM devices EXCEPT 1000 ROWS;

-- Example 3) Delete all of them from now on except one day.
DELETE FROM devices EXCEPT 1 DAY;

-- Example 4) Delete all data from before June 1, 2014.
DELETE FROM devices BEFORE TO_DATE('2014-06-01', 'YYYY-MM-DD');
```

For Tag Tables

The following command is supported:

```
-- Delete all data from before June 15, 2016.
DELETE FROM tag BEFORE TO_DATE('2016-06-15', 'YYYY-MM-DD');
```

Automated Data Collection

Machbase provides a "Collector" function that reads data from scattered machine data log files and automatically transfers them.

It not only collects pre-formatted data such as syslog and web server logs, but also provides a function that can be easily converted and automatically collected even if the log format is arbitrarily defined by the user.

Introduction of Machbase Products

Machbase has the following three products that can be applied to each type of business environment.

- Edge Edition
- Fog Edition
- Cluster Edition

Edge Edition

Necessity

The Edge Edition refers to the Machbase line of products that operate at high speeds on small devices with limited resources.
Since late 2010, Edge equipment with a certain amount of computing power has been released, and various businesses are being developed through the Edge equipment in the field of Industrial IoT.
It is common to monitor the state of the production equipment in a specific environment, store data generated in a robot or place where a large amount of sensing is required, and transmit the data to the parent server when necessary.
However, in order to operate smoothly on these small-scale Edge devices, the key is to store data at high speed and extract data in real time.
Machbase enables this requirement through its innovative technology and supports the following hardware.

Supported Hardware Examples

	Rasberry PI 3	Samsung ARTIK 7	LattePanda	nvidia Jetson TX2
형태 Type				
CPU	ARM Cortex-A53 (64bit)	ARM Cortex-A53 (64bit)	Intel ATOM (x5-Z8350)	ARM Coretx-A57 (64bit)
MEMORY	1 GB	1 GB	4 GB	8 GB

Fog Edition

Necessity

Fog Edition is a product used to store and analyze large amounts of data transmitted by the Edge compute device.
This Fog Edition is designed for maximum performance on a single hardware device, so it can be installed, managed and operated very quickly and easily, without the need for complex installation and configuration.
In particular, it is built in a single appliance and installed and used as factory and building equipment, and plays a primary role of collecting client's data.

Supported Hardware and Operating Systems

Fog Edition supports 64-bit Linux and Windows 8 or later operating system based on Intel CPU.

Cluster Edition

The Cluster Edition is a product with high input speed and standard SQL that can process large data input / inquiries in a distributed environment that not even the Machbase Fog Edition can process, such as on multiple servers either on premise or in the cloud.

Index

Why Should I Use the Cluster Edition?

Machbase offers Fog Edition which inputs time series data at extremely high speed. However, the following disadvantages exist:

- Because it consists of a single process, it lacks in high availability (HA).
- Because one process is dedicated to analyzing data, there is a limit to increasing the performance of large data analysis.

To overcome these shortcomings, there is a need for a higher end distributed product that can ensure availability and scalability when storing and analyzing large amounts of data. The Machbase Cluster Edition meets these requirements.

Terms

Host

Represents one physical server, or one OS instance in the cloud/VM.

Node

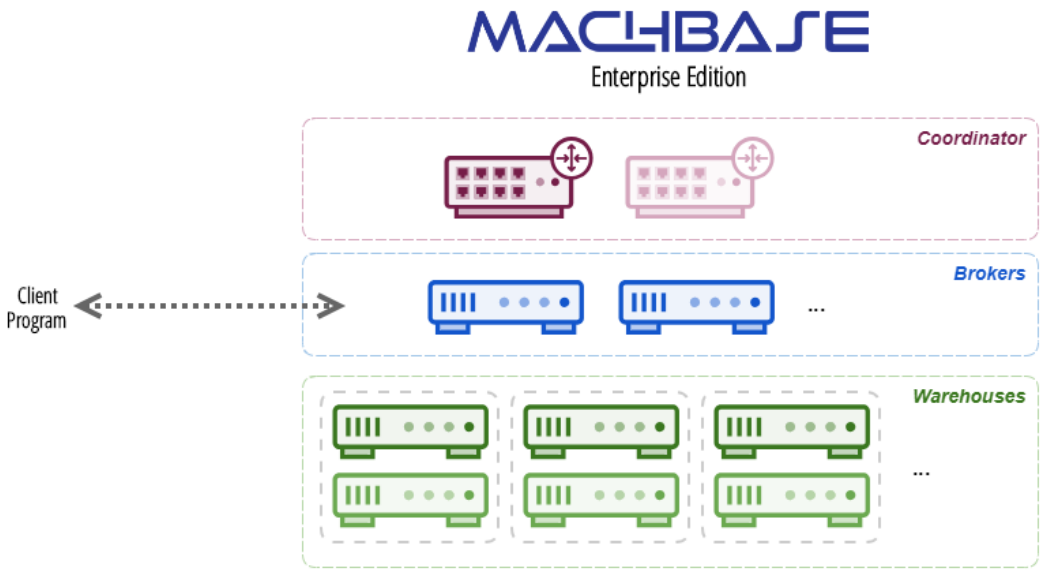
Represents the Machbase Process residing on the server.

The Process type is the same as the Node types below.

- Coordinator
- Deployer
- Broker
- Warehouse

Structure

In the Machbase Cluster Edition, several nodes that reside in the Host constitute one cluster.



High Availability

The service can be continued even if one of all the internal nodes is interrupted.

High Scalability

Data storage can be distributed, and parallel analysis is possible from the distributed data, so performance increases as the cluster grows.

Classifications of Node

Each Node can be classified as follows:

Classification	Description	Process Name
Coordinator	The process of managing all general purpose servers and nodes	machcoordinatord
Deployer	The process that resides on each host Responsible for installing, upgrading, and monitoring the Broker / Warehouse.	machdeployerd
Broker	The process of welcoming an actual client program. Serves to distribute client data insert / data lookup queries to the warehouse.	machbased
	Leader: Brokers that can perform DDL	
Warehouse	The process that stores the actual data Stores some of the entire cluster data and executes the commands received from the Broker.	machbased

Coordinator

Coordinator is a process that manages the state of all nodes, and can be a maximum of two.

First, the generated Coordinator is called the Primary Coordinator, and the other is called the Secondary Coordinator, and only the Primary Coordinator manages the state of all the Nodes.
When the Primary Coordinator is down, the Secondary Coordinator is upgraded to Primary Coordinator.

Special Node : Deployer

It is managed by the Coordinator, but it is simply **a process that installs / removes Broker / Warehouse Nodes.**

Normally, only one Deployer can be added to the Host at a time when installing Nodes, but multiple Deployers can be added for installation performance.

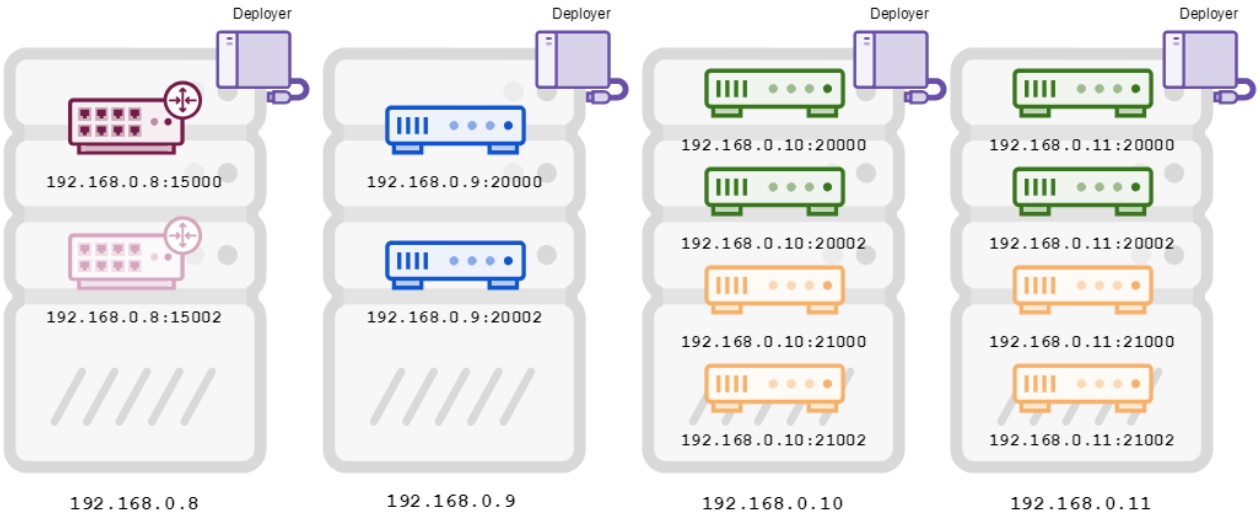


Figure (a) Deploying MachBase Nodes in Commodity Servers

Example of Installation From Server

Figure (a) below shows the installation of two Coordinators, two Brokers, four Warehouse Active, and four Warehouse Standby Nodes on four generic servers.

As shown in the figure, you can distinguish each node with **'hostname: port'** followed by the host name of the general-purpose server and the assigned port number.

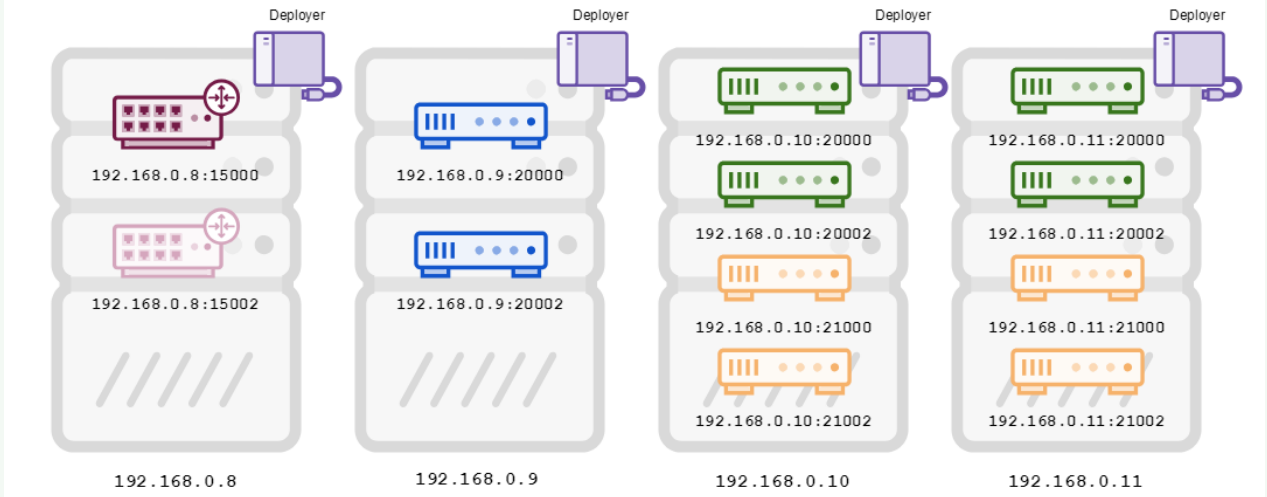


Figure (a) Deploying MachBase Nodes in Commodity Servers

Broker

The Broker delivers the Client's commands to the Warehouse, and then collects the results of the Warehouse and transmits them back into the Client.

- When entering data, the Broker makes sure the data enters evenly into the Warehouse.
- When retrieving data, the Broker fetches the data to the Warehouse and collects and delivers all the results.

The Broker does not have the data in the Log Table, but it has the data in the Volatile Table.

Leader Broker

If all Brokers run DDL at the same time, there will be a problem with the overall consistency of the cluster.

If the DDL command starts to be controlled at the cluster level to solve this problem, the performance of the DDL itself deteriorates.

Therefore, there needs to be a specified Broker that can perform DDL, which is called the **Leader Broker** .
The Leader Broker can perform DDL, but other Brokers can not perform DDL.

Warehouse

The Warehouse will store the Log Table data directly, and will act as the actual execution of the command delivered by the Broker.
Like the Broker, there is direct client access to the Warehouse, but the data can not be input / updated / deleted; the Warehouse data can only be retrieved.

Warehouse Group

The Warehouse can specify the Group to which it belongs.

- When the Broker inputs data, all Warehouses in the same Group receive the same records.
- Even if a specific Warehouse of the group is dropped, there are no issues with data retrieval.
- When a new Warehouse is added to a group, the same records are maintained through Replication.

Warehouse Group State

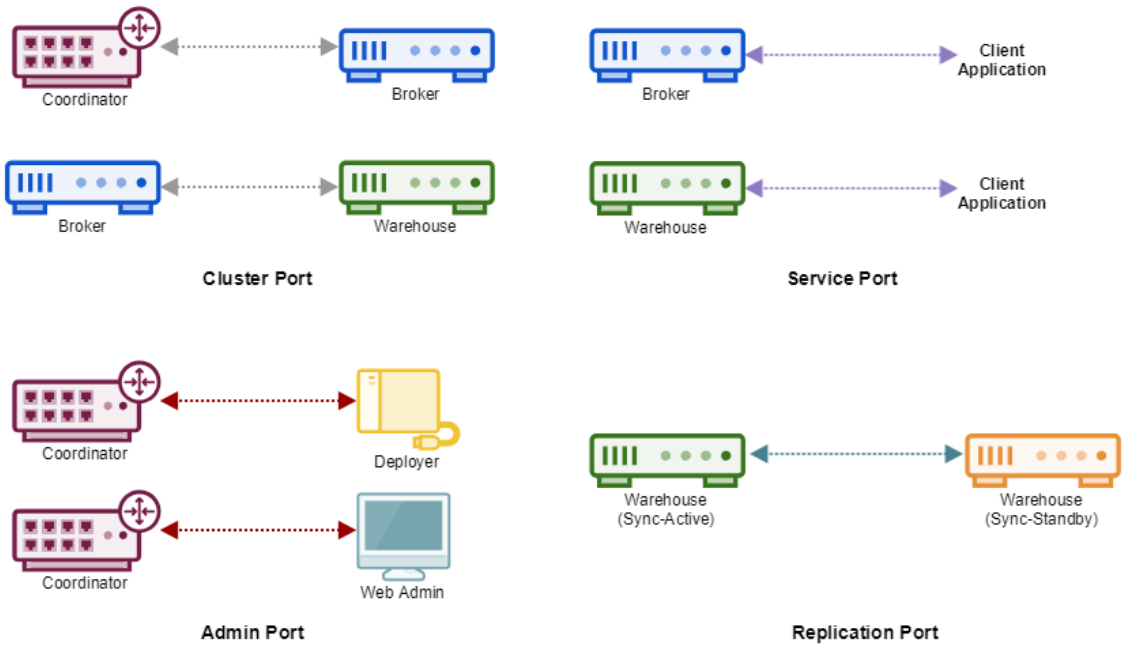
State	INSERT / APPEND	SELECT
Normal	O	O
Readonly	X	O

The conditions that change to Readonly state are as follows.

- During INSERT / APPEND, if some Warehouses in the Group fail to be input
  - Because there is a data inconsistency between the failed Warehouses and the successful Warehouses, the failed Warehouse is placed in the Scrapped state and the group is moved to the Readonly state to avoid receiving further input.
- When a new Warehouse is added
  - If the input is received even while the replication process is in progress, the state is changed to the Readonly state because the end of the replication is unknown.

Node Port Management

Each Node must have several ports open, which are distinguished as follows:



Port Classification	Description	Required Node
Cluster Port	Port for communication between Nodes	All Nodes
Service Port	Port directly connected by client	Broker / Warehouse

Port Classification	Description	Required Node
Admin Port	Port for communication for management purpose	Coordinator / Deployer
Replication Port	Port for communication between warehouses for replication	Warehouse

Commands That Can Be Executed After Direct Connection

The following table lists the possible and not possible commands to connect directly to each Node.
All nodes are accessible via the client, but there are queries that are not possible depending on the type of Node.

	Broker (Leader)	Broker (non-leader)	Warehouse Standby
Client Connection	O	O	O
DDL	O	X	X
DELETE	O	O	X
INSERT	O	O	X [1)]
APPEND	O	O	X [1)]
SELECT	O	O	O

Save / Lookup Data

Machbase Cluster Edition can distribute the data and collect the results computed by distributed query execution. This section explains how to store and lookup the table type.

Log Table

When data is entered into the Log Table through the Broker, it is distributed to all warehouses. (The data is not stored in the Broker that performs the input.)
The Coordinator determines the database size of each warehouse, and the Broker distributes the data based on that.
If data is entered directly into the Log Table via the Warehouse, it is stored only in the corresponding Warehouse. Can be selected to avoid performance degradation due to distributed algorithm and network bottleneck.

Volatile Table

When a Broker enters data into a Volatile Table, it is stored in the corresponding Broker. In other words, no data is entered or synchronized with other Brokers. The reason for not supporting replication for Volatile Tables is that if it matches the characteristics of the Volatile Table able to DELETE, it affects the replication performance.
Volatile tables are created only in the Broker, so they can not be entered in the Warehouse.

Data Retrieval

Log Table

When you view the data in the Log Table through the Broker, queries are distributed to all Warehouses. Each Warehouse actually performs the query, exchanging intermediate results between the Warehouses if necessary. The Broker collects the partial results generated in this way and returns the final result.
When viewing the data in the Log Table through the Warehouse, the query is executed only in the corresponding warehouse. This process is identical to query execution in the Fog Edition.

Volatile Table

When viewing data in a Volatile Table through a Broker, the query is executed only by the Broker. This process is identical to query execution in the Fog Edition. JOIN can not be done through the Warehouse, because Volatile Tables are not created.

JOIN Between Two Tables

When joining a Log Table and a Volatile Table through a Broker, the connected Broker and the rest of the Warehouse execute the query at the same time. The Broker distributes the results of the Volatile table to the Warehouse.
The Warehouse JOINs the data delivered by the broker and returns the result. The Broker collects the partial results generated in this way and returns the final result.
JOIN can not be done through the warehouse, because volatile tables are not created.

Replication

Replication refers to the process of replicating the same node in preparation for failure of an existing node.

Coordinator Replication

Up to two Coordinators can be created in Cluster Edition.

Both Coordinators continuously maintain Cluster Node information.
Even if either end abnormally, the remaining Coordinator can continue managing the Cluster Node.

> When the Primary Coordinator is restarted, the existing secondary coordinator is upgraded to primary and the restarting coordinator becomes secondary.

The Broker is not a Replication target.
Broker Replication

Therefore, the data record of the Volatile Table in Broker A is not kept the same in Broker B. (not synchronized)
However, because the table / index scheme of the entire Cluster are all the same, if the Volatile Table **`VOL_TBL1`** exists with Broker A, Volatile Table **`VOL_TBL1`** also exists with Broker B.

Warehouse Replication

If a new Warehouse is added to the Group, the Warehouse is replicated through the following process.

1. The Coordinator starts DDL replication to the new Warehouse.
2. Group switches to Readonly state.3
3. One of the Warehouses in the group starts data replication to the new warehouse.
4. When the data replication is completed, the group is switched to the Normal state.

In the case of data insert, the Broker guarantees redundancy by sending the same data to the same Group.

How To Recover

Even if the Node terminates abnormally, the service can be continued in the following manner.

For more information, refer to the Operations Guide.

Type	Fail-over Method
Coordinator	Even if the Primary Coordinator is abnormally terminated, the Secondary Coordinator becomes the Primary Coordinator and the cluster management can continue. Even if the Coordinator is terminated in the worst case scenario, the entire service (data insert / inquiry) can be continued without the cluster management. (Of course, when the Broker or Warehouse is shut down at this time, you can not manage the cluster.)
Deployer	Node operation (ADD, REMOVE ..) can not be performed on the corresponding Host, and statistical information of the host can not be collected.
Broker	Even if the Broker is terminated, the service can continue if another Broker exists. However, because the connection to the client that has been terminated is disconnected, it must be reconnected to another Broker. If the Leader Broker is stopped, the Coordinator re-selects the Leader Broker in the Broker to enable DDL execution. 📌 Lookup Table data stored in the Broker is not replicated, so be careful.
Warehouse	If there is another / other Warehouse (s) in the Group, the Warehouse (s) will participate in SELECT and APPEND.

Not Supported Features

Query Statement

TABLESPACE

Currently, the Cluster Edition does not distinguish between table spaces.

BACKUP / MOUNT

Currently, the Cluster Edition does not distinguish between databases.

LOAD IN FILE

The ability to read and distribute CSV files is currently not implemented.

ALTER TABLE FORGERY CHECK

Result File can not be collected in one place as client's user data is checked for any changes.

Clause / Function

UNION ALL

Execution units are complex and are currently not supported.

GROUP_CONCAT() function

The entire contents of the CONCAT for the subgroups collected by each Warehouse can not be processed as a simple accumulation.
(ORDER BY in GROUP CONCAT)

TS_CHANGE_COUNT() function

The TS_CHANGE_COUNT results for the subgroups collected in each Warehouse can not be processed as simple accumulations.
In addition, TS_CHANGE_COUNT () is significant if the entire result is sorted, but if the results are distributed in the Warehouse, it is meaningless.

Supported Hardware and Operating Systems

CPU	Intel Core i Series (Nehalem ~) or higher recommended
Memory	2 GB or more recommended for each Node to be installed
Operating System	Linux (Any distribution)

Installation

Package Overview

Package Type

MACHBASE provides manual installation and package installation files.

Installation type	Description	Note
manual installation	Has a compressed file format and the extension tgz for Unix. The user decompresses using tar and GNU gzip to proceed with the installation.	Can be installed only in console environment
package installation	Provides an installation package for each operating system environment • Windows : msi • RHEL/CentOS Linux : rpm • Debian/Ubuntu : deb • Docker : (Dockerhub image)	Can be installed only in console environment

Index

Package File Name Structure

The package file name is configured as follows.

> "*machbase*-EDITION-VERSION-OS-CPU-BIT-MODE-OPTIONAL.EXT"

item	Description
EDITION	Indicates the edition of the package. edge : Edge Edition fog : Fog Edition cluster : Cluster Edition
VERSION	Indicates the version of the package. In detail, it is classified as MajorVersion.MinorVersion.FixVersion.AUX by numbers and characters. Major Version : Product main version (number) Minor Version : A version with relatively large features added in the same main version. DB file / protocol compatibility is not guaranteed. (number) Fix Version : A bug / minor feature added in the same main version. DB file / protocol compatibility is guaranteed. (number) AUX : Indicates the package classification (character) • official : general package • community : community edition package
OS	Indicates the operating system name. (Example) LINUX, WINDOWS
CPU	Indicates the type of CPU installed in the operating system. (Example) X86, IA64
BIT	Indicates whether the compiled binary is 32-bit or 64-bit. (Example) 32, 64
MODE	Indicates the release mode of the binary once compiled. (Example) release, debug, prerelease

<table>
<tr><th>item</th><th>Description</th></tr>
<tr><td></td><td></td></tr>
<tr><td>OPTIONAL</td><td>Only displayed in Enterprise Edition.

<table><tr><td>lightweight</td><td>Indicates a lightweight package to be added to the Coordinator.</td></tr></table></td></tr>
<tr><td>EXT</td><td>The package file extension. Depending on the package, it is available as tgz, rpm, deb, and msi.</td></tr>
</table>

Linux Installation

Preparing Linux Environment for Installation

Check and Set Maximum Number of Files

1. Check the maximum number of Linux files with the following command.

```
[machbase@localhost ~] ulimit -Sn
1024
```

2. If the result is less than 65535, modify the file below and reboot the server.

```
[machbase@localhost ~] sudo vi /etc/security/limits.conf


#<domain>      <type>  <item>         <value>
#

*               hard    nofile          65535
*               soft    nofile          65535
```

3. Reboot the server and check the value again.

```
[machbase@localhost ~] ulimit -Sn
65535
```

Check and Set Server Time

Because Machbase is a database that deals with time series data, you need to set the time value correctly on the server where Machbase will be installed.

Setting Time Zone

Since Machbase uses all the data in the local time where the server is located, you need to make sure that the timezone matches the time of the current server. Make sure it matches the timezone where you are located with the following command: If different, select the correct region from /usr/share/zoneinfo and link.

```
[machbase@localhost ~] ls -l /etc/localtime
lrwxrwxrwx 1 root root 32 Sep 27 14:08 /etc/localtime -> ../usr/share/zoneinfo/Asia/Seoul


# You can check the timezone set through the date command.
[machbase@localhost ~] date
Wed Jan  2 11:12:44 KST 2019
```

Setting Time

If the current local time is not correct, reset the time using the following command.

```
[machbase@localhost ~] sudo date -s '2018/12/25 12:34:56'
```

Tarball Installation

Index

Create User

Create a Linux user 'machbase' for installing and using Machbase.

```
sudo useradd machbase
```

After setting the password, log in as 'machbase' account.

Package Installation

Create a directory called 'macbase_home' and download and install the package from the Machbase download site.

```
[machbase@localhost ~]$ wget http://www.machbase.com/dist/machbase-fog-x.x.x.of
[machbase@localhost ~]$ mkdir machbase_home
[machbase@localhost ~]$ mv machbase-fog-x.x.x.official-LINUX-X86-64-release.tgz
[machbase@localhost ~]$ cd machbase_home/
[machbase@localhost machbase_home]$ tar zxf machbase-fog-x.x.x.official-LINUX-X

[machbase@loclahost machbase_home]$ ls -l
drwxrwxr-x  5 machbase machbase        64 Oct 30 16:10 3rd-party
drwxrwxr-x  2 machbase machbase      4096 Oct 30 16:10 bin
drwxrwxr-x  6 machbase machbase       189 Dec 21 14:04 collector
drwxrwxr-x  2 machbase machbase       306 Jan  2 11:36 conf
drwxrwxr-x  2 machbase machbase       136 Jan  2 11:37 dbs
drwxrwxr-x  3 machbase machbase        22 Oct 30 16:10 doc
drwxrwxr-x  2 machbase machbase        96 Oct 30 16:10 include
drwxrwxr-x  2 machbase machbase        29 Oct 30 16:10 install
drwxrwxr-x  2 machbase machbase       283 Oct 30 16:10 lib
-rw-rw-r--  1 machbase machbase 139888377 Dec 20 11:33 machbase-fog-x.x.x.offic
drwxrwxr-x  2 machbase machbase        22 Dec 21 15:43 msg

drwxrwxr-x  2 machbase machbase         6 Oct 30 16:10 package
drwxrwxr-x 12 machbase machbase       140 Oct 30 16:10 sample
drwxrwxr-x  2 machbase machbase      4096 Jan  2 09:37 trc
drwxrwxr-x 10 machbase machbase       160 Oct 30 16:10 tutorials
drwxrwxr-x  3 machbase machbase        19 Oct 30 16:10 webadmin

[machbase@loclahost machbase_home]$
```

The directory descriptions installed are as follows.

Directory	Description
bin	Executable files
collector	Log collector files
conf	Configuration files
dbs	Data storage space
doc	License files
include	Various header files for the CLI program
install	mk files for Makefile
lib	Various libraries
msg	Machbase server error messages

Directory	Description
package	(Cluster edition) Path to save the added package
sample	Various example files
trc	Machbase server logs and trace contents
webadmin	MWA Web Server Files
3rd-party	Grafana plug-in files

Set Environment Variable

Add Machbase-related environment variables to your .bashrc file.

```
export MACHBASE_HOME=/home/machbase/machbase_home
export PATH=$MACHBASE_HOME/bin:$PATH
export LD_LIBRARY_PATH=$MACHBASE_HOME/lib:$LD_LIBRARY_PATH

# Apply the changes with the following command.
source .bashrc
```

Set Machbase Property

The $MACHBASE_HOME/conf directory contains the file macbase.conf.sample.

```
[machbase@localhost ~]$ cd $MACHBASE_HOME/conf
[machbase@localhost conf]$ ls -l
-rw-rw-r-- 1 machbase machbase   106 Oct 30 16:10 machtag.sql.sample
-rw-rw-r-- 1 machbase machbase 17556 Oct 30 16:10 machbase.conf.sample
-rw-rw-r-- 1 machbase machbase  1706 Oct 30 16:10 machcollector.conf.sample

[machbase@localhost conf]$
```

You can also change the Machbase connection port number using the Linux environment variable. Below is an example of switching to a different port number (7878) other than the default value (5656).

```
export MACHBASE_PORT_NO=7878
```

Machbase Simple Usage

Create Database

To create the database, use the machadmin utility. You can see the command with the --help option.

```
[machbase@localhost machbase_home]$ machadmin --help
------------------------------------------------------------------
     Machbase Administration Tool
     Release Version - x.x.x.official
     Copyright 2014, MACHBASE Corp. or its subsidiaries
     All Rights Reserved
------------------------------------------------------------------
<< available option lists >>
  -u, --startup                        Startup Machbase server.
      --recovery[=simple,complex,reset] Recovery mode. (default: simple)
  -s, --shutdown                       Shutdown Machbase server.
  -c, --createdb                       Create Machbase database.
```

```
  -d, --destroydb                     Destroy Machbase database.
  -k, --kill                          Terminate Machbase server.
  -i, --silence                       Produce less output.
  -r, --restore                       Restore Machbase database.
  -x, --extract                       Extract BackupFile to BackupDirectory.
  -w, --viewimage                     Display information of BackupImageFile.
  -e, --check                         Check whether Machbase Server is running.
  -t, --licinstall                    Install the license file.
  -f, --licinfo                       Display information of installed license file.

[machbase@localhost machbase_home]$
```

Create a database with the -c option.

```
[machbase@localhost machbase_home]$ machadmin -c
------------------------------------------------------------------
     Machbase Administration Tool
     Release Version - x.x.x.official
     Copyright 2014, MACHBASE Corp. or its subsidiaries
     All Rights Reserved
------------------------------------------------------------------
Database created successfully.
[machbase@localhost machbase_home]$
```

Launch Machbase Server

Run the Machbase server with the -u option.

```
[machbase@localhost machbase_home]$ machadmin -u
------------------------------------------------------------------
     Machbase Administration Tool
     Release Version - x.x.x.official
     Copyright 2014, MACHBASE Corp. or its subsidiaries
     All Rights Reserved
------------------------------------------------------------------
Waiting for Machbase server start.
Machbase server started successfully.
[machbase@localhost machbase_home]$
```

You can see that the server daemon, machbased, is running through the ps command as shown below.

```
[machbase@localhost machbase_home]$  ps -ef |grep machbased
machbase 11178     1  2 11:25 ?        00:00:01 /home/machbase/machbase_home/bin/machbased -s --recovery=simple
machbase 11276  9867  0 11:26 pts/1    00:00:00 grep --color=auto machbased
[machbase@localhost machbase_home]$
```

Machbase Server Connection

Connect to the Machbase server using an access utility called machsql.

The administrator account SYS is ready and the password is set to MANAGER.

```
[machbase@localhost machbase_home]$  machsql
==================================================================
     Machbase Client Query Utility
     Release Version x.x.x.official
     Copyright 2014 MACHBASE Corporation or its subsidiaries.
     All Rights Reserved.
==================================================================
Machbase server address (Default:127.0.0.1) :
Machbase user ID  (Default:SYS)
Machbase User Password :
MACHBASE_CONNECT_MODE=INET, PORT=5656
```

```
Type 'help' to display a list of available commands.
Mach>
```

Let's create a simple table and input / output data.

```
create table hello( id integer );
insert into hello values( 1 );
insert into hello values( 2 );
select * from hello;
select _arrival_time, * from hello;
```

```
Mach> create table hello( id integer );
Created successfully.
Elapsed time: 0.054
Mach> insert into hello values( 1 );
1 row(s) inserted.
Elapsed time: 0.000
Mach> insert into hello values( 2 );
1 row(s) inserted.
Elapsed time: 0.000
Mach> select * from hello;
ID
--------------
2
1
[2] row(s) selected.
Elapsed time: 0.000
Mach> select _arrival_time, * from hello;
_arrival_time                   ID
-----------------------------------------------
2019-01-02 11:33:00 122:806:804 2
2019-01-02 11:32:57 383:848:361 1
[2] row(s) selected.
Elapsed time: 0.000
Mach>
```

The above SELECT results show that the most recently input data is displayed first.

Also, it can be seen through the _arrival_time column that the input time of the record is set to the nanosecond.

Stop Machbase Server

Shut down the Machbase server with the -s option.

Delete Database

Delete the database with the -d option.

> Be very careful because all data will be deleted.

```
[machbase@localhost machbase_home]$ machadmin -d
------------------------------------------------------------------
     Machbase Administration Tool
     Release Version - x.x.x.official
     Copyright 2014, MACHBASE Corp. or its subsidiaries
     All Rights Reserved
------------------------------------------------------------------
Destroy Machbase database. Are you sure?(y/N) y
Database destoryed successfully.
[machbase@localhost machbase_home]$
```

```
[machbase@localhost machbase_home]$ machadmin -s
------------------------------------------------------------------
     Machbase Administration Tool
     Release Version - x.x.x.official
```

```
     Copyright 2014, MACHBASE Corp. or its subsidiaries
     All Rights Reserved
--------------------------------------------------------------------
Waiting for Machbase server shut down...
Machbase server shut down successfully.
[machbase@localhost machbase_home]$
```

Installing Machbase

Linux on the Redhat / CentOS family can install Machbase via the rpm file.

Download the latest RPM package first. You can download it with the command below.

```
$ wget http://www.machbase.com/dist/machbase-fog-x.x.x.official-LINUX-X86-64-re
```

The command to install after downloading the file is as follows.

```
$ sudo yum install machbase-fog-x.x.x.official-LINUX-X86-64-release.rpm

Loaded plugins: fastestmirror, refresh-packagekit, security
Setting up Install Process
Examining machbase-fog-x.x.x.official-LINUX-X86-64-release.rpm: machbase-x.x.x-
Marking machbase-fog-x.x.x.official-LINUX-X86-64-release.rpm to be installed
Loading mirror speeds from cached hostfile
 * base: data.aonenetworks.kr
 * extras: data.aonenetworks.kr
 * updates: data.aonenetworks.kr
Resolving Dependencies
--> Running transaction check
---> Package machbase.x86_64 0:x.x.x-official will be installed
--> Finished Dependency Resolution
```

```
Dependencies Resolved
================================================================================================================================
 Package         Arch          Version                 Repository                                                      S
================================================================================================================================
Installing:
 machbase        x86_64        x.x.x-official         /machbase-fog-x.x.x.official-LINUX-X86-64-release        632

Transaction Summary
================================================================================================================================
Install       1 Package(s)

Total size: 632 M
Installed size: 632 M
Is this ok [y/N]: y
Downloading Packages:
Running rpm_check_debug
Running Transaction Test
Transaction Test Succeeded
Running Transaction
Warning: RPMDB altered outside of yum.
  Installing : machbase-x.x.x-official.x86_64                                                                    1

Create database
-----------------------------------------------------------------
     Machbase Administration Tool
     Release Version - x.x.x.official
     Copyright 2014, MACHBASE Corp. or its subsidiaries
     All Rights Reserved
-----------------------------------------------------------------
Database created successfully.

Ulimit check
65535 PASS

Machbase startup
-----------------------------------------------------------------
```

```
     Machbase Administration Tool
     Release Version - x.x.x.official
     Copyright 2014, MACHBASE Corp. or its subsidiaries
     All Rights Reserved
-----------------------------------------------------------------
-----------------------------------------------------------------
     Machbase Administration Tool
     Release Version - x.x.x.official
     Copyright 2014, MACHBASE Corp. or its subsidiaries
     All Rights Reserved
-----------------------------------------------------------------
Waiting for Machbase server start.
Machbase server started successfully.

MWA startup
SERVER HAS BEEN RESET
SERVER STARTED, PID : 7757
     Connection URL : http://192.168.0.55:5001
Machbase has been installed in : /opt/machbase/
To start Machbase, run the command : service machbased start
To change server port, run the command : service machbased port
To use interactive SQL, execute : machsql
Documentation is available at http://www.machbase.com/document
  Verifying  : machbase-x.x.x-official.x86_64                                                     1

Installed:
  machbase.x86_64 0:x.x.x-official

Complete!
```

When the installation is complete, the / opt / machbase folder is created and the default port is set to 5656. **After that, the database is created and the Machbase server and the MWA Web server are automatically executed.**
In the machbase directory, there is a directory named 'current' which is a symbolic link to the latest version. In the versions directory, files are located according to the Machbase version.

```
[root@localhost ~]$ cd /opt/machbase
[root@localhost machbase]# ls -l
total 4
lrwxrwxrwx. 1 root     root       28 Jan  2 14:12 current -> /opt/machbase/versions/x.x.x
drwxr-xr-x. 3 machbase machbase 4096 Jan  2 14:12 versions
[root@localhost machbase]$
```

A shell script, machbased, is installed in the /etc/init.d directory, and you can manage Machbase with this.

```
[root@localhost ~]$ cd /etc/init.d
[root@localhost init.d]# ls -l machbased
-rwxr-xr-x. 1 root root 8446 Oct 30 16:11 machbased
[root@localhost machbase]$
```

Deleting Machbase

To delete Machbase, use the following command.

```
[root@localhost ~]$ sudo yum remove machbase
```

Managing Machbase

If you install the Machbase server using rpm, the /etc/init.d/machbased script file is installed and you can manage it conveniently.

The basic functions to be supported are as follows.

```
[root@localhost init.d]$ service machbased
Usage: /etc/init.d/machbased {start|stop|kill|restart|createdb|destorydb|check|MWA|console|port|exe|collector|help]
```

Start Server

Start the Machbase server. It is the same as machadmin -u.

```
[root@localhost ~]$ sudo service machbased start
------------------------------------------------------------------
     Machbase Administration Tool
     Release Version - x.x.x.official
     Copyright 2014, MACHBASE Corp. or its subsidiaries
     All Rights Reserved
------------------------------------------------------------------
Waiting for Machbase server start.
Machbase server started successfully.
[root@localhost ~]$
```

Shut Down Server

Shut down the Machbase server normally. It is the same as machadmin -s.

```
[root@localhost ~]$ sudo service machbased stop
------------------------------------------------------------------
     Machbase Administration Tool
     Release Version - x.x.x.official
     Copyright 2014, MACHBASE Corp. or its subsidiaries
     All Rights Reserved
------------------------------------------------------------------
Waiting for Machbase server shut down...
Machbase server shut down successfully.
[root@localhost ~]$
```

Stop Server

Force the Machbase server to abort. It is the same as machadmin -k.

```
[root@localhost ~]$ sudo service machbased kill
------------------------------------------------------------------
     Machbase Administration Tool
     Release Version - x.x.x.official
     Copyright 2014, MACHBASE Corp. or its subsidiaries
     All Rights Reserved
------------------------------------------------------------------
Waiting for Machbase server terminated.
Machbase server terminated successfully.
[root@localhost ~]$
```

Restart Server

Shut down normally and re-start the Machbase server.

```
[root@localhost ~]$ sudo service machbased restart
------------------------------------------------------------------
     Machbase Administration Tool
     Release Version - x.x.x.official
     Copyright 2014, MACHBASE Corp. or its subsidiaries
     All Rights Reserved
------------------------------------------------------------------
Waiting for Machbase server shut down...
Machbase server shut down successfully.
```

```
-----------------------------------------------------------------
     Machbase Administration Tool
     Release Version - x.x.x.official
     Copyright 2014, MACHBASE Corp. or its subsidiaries
     All Rights Reserved
-----------------------------------------------------------------
Waiting for Machbase server start.
Machbase server started successfully.
[root@localhost ~]$
```

Create Database

Create a Machbase database. It is the same as machadmin -c.

```
[root@localhost ~]$ sudo service machbased createdb
-----------------------------------------------------------------
     Machbase Administration Tool
     Release Version - x.x.x.official
     Copyright 2014, MACHBASE Corp. or its subsidiaries
     All Rights Reserved
-----------------------------------------------------------------
Database created successfully.
[root@localhost ~]$
```

Delete Database

Delete the Machbase database. It is the same as machadmin -d.

```
[root@localhost ~]$ sudo service machbased destroydb
-----------------------------------------------------------------
     Machbase Administration Tool
     Release Version - x.x.x.official
     Copyright 2014, MACHBASE Corp. or its subsidiaries
     All Rights Reserved
-----------------------------------------------------------------
Destroy Machbase database. Are you sure?(y/N) y
Database destoryed successfully.
[root@localhost ~]$
```

Check Server Status

Check the status of the Machbase server operation. It is the same as machadmin -e.

```
[root@localhost ~]$ sudo service machbased check
-----------------------------------------------------------------
     Machbase Administration Tool
     Release Version - x.x.x.official
     Copyright 2014, MACHBASE Corp. or its subsidiaries
     All Rights Reserved
-----------------------------------------------------------------
Machbase server is running with PID(23542).
[root@localhost ~]$
```

MWA Management

This is the command related to the MWA (Machbase Web Analytics) web server.

```
[root@localhost ~]$ sudo service machbased  MWA
start | restart | stop | reset | port

# Start the MWA server.
```

```
[root@localhost ~]$ sudo service machbased MWA start
SERVER ALREADY STARTED
Connection URL : http://192.168.0.10:5001

# Shut down the MWA server.
[root@localhost ~]$ sudo service machbased MWA stop
SERVER STOPPED

# Change the MWA server port.
[root@localhost ~]$ sudo service machbased MWA port 5050
WEBSERVER PORT CHANGED : 5050

# Shut down and restart the MWA server.
[root@localhost ~]$ sudo service machbased MWA restart
SERVER IS RESTARTING
SERVER STOPPED
SERVER STARTED, PID : 23810
     Connection URL : http://192.168.0.10:5001
[root@localhost ~]$
```

Change Server Port

Change the port of the Machbase server. If you execute the command and enter the port to be changed, it changes to that port. After the port change, you must restart the Machbase server to apply

```
[root@localhost ~]$ sudo service machbased port
The default port for the Machbase server is 5656. If you want to use 5656 as Machbase server port, press return key
Use current port.
[root@localhost ~]$
```

Collector Management

These are the commands to manage Machcollector.

```
[root@localhost ~]$ sudo service machbased collector
List of commands:
  * machbased collector start
    Machcollectormanager starts.
  * machbased collector stop
    Machcollectormanager shutdown.
  * machbased collector kill
    Terminate Machcollectormanager.
  * machbased collector destroy
    Destroy Machcollectormanager meta data.
  * machbased collector add_server
    Add an Machbase server to Machcollectormanager.
  * machbased collector port
    Change the default port. Now: 9999
[root@localhost ~]$
```

DEB Installation

Installing Machbase

Download and install the Machbase package from the Machbase download site.

```
root@ubuntu:/usr/local/src# wget http://www.machbase.com/dist/machbase-fog-x.x.
root@ubuntu:/usr/local/src# sudo dpkg -i machbase-fog-x.x.x.community-LINUX-X86
Selecting previously unselected package machbase.
(Reading database ... 464623 files and directories currently installed.)
Preparing to unpack machbase-fog-x.x.x.community-LINUX-X86-64-release.deb ...
Group machbase exist
Unpacking machbase (5.1.9) ...
Setting up machbase (5.1.9) ...

Create database
-----------------------------------------------------------------
     Machbase Administration Tool
     Release Version - 5.1.9.community
     Copyright 2014, MACHBASE Corp. or its subsidiaries
     All Rights Reserved
-----------------------------------------------------------------
Database created successfully.

Ulimit check
65535 PASS

Machbase startup
-----------------------------------------------------------------
     Machbase Administration Tool
     Release Version - 5.1.9.community
     Copyright 2014, MACHBASE Corp. or its subsidiaries
     All Rights Reserved
-----------------------------------------------------------------
Waiting for Machbase server start.
Machbase server started successfully.

MWA startup
SERVER HAS BEEN RESET
SERVER STARTED, PID : 1957
     Connection URL : http://192.168.0.183:5001
Machbase has been installed in : /opt/machbase/
To start Machbase, run the command : service machbased start
To change server port, run the command : service machbased port
To use interactive SQL, execute : machsql
Documentation is available at http://www.machbase.com/document
Processing triggers for systemd (229-4ubuntu21.1) ...
Processing triggers for ureadahead (0.100.0-19) ...
root@ubuntu:/usr/local/src#
```

Index

If you have dependency problems during installation, you can install them with the following command:

```
root@ubuntu:/usr/local/src# sudo apt-get install -f
root@ubuntu:/usr/local/src# sudo dpkg -i machbase-fog-x.x.x.community-LINUX-X86-64-release.deb
```

When the installation is complete, the / opt / machbase folder is created and the default port is set to 5656. **After that, the database is created and the Machbase server and the MWA Web server are automatically executed.**
In the Machbase directory, there is a directory named 'current' which is a symbolic link to the latest version. In the versions directory, files are located according to the Machbase version.

```
root@ubuntu:/usr/local/src# cd /opt/machbase
root@ubuntu:/opt/machbase# ls -l
total 4
lrwxrwxrwx 1 root     root       28 Jan  3 00:25 current -> /opt/machbase/versions/5.1.9
```

```
drwxrwxr-x 3 machbase machbase 4096 Jan  3 00:25 versions
root@ubuntu:/opt/machbase#
```

Deleting Machbase

To delete Machbase, use the following command.

```
root@ubuntu:/opt/machbase# sudo dpkg -r machbase
```

Managing Machbase

If you install the Machbase server using deb, the /etc/init.d/machbased script file is installed and you can manage it conveniently.

The basic functions to be supported are as follows.

```
root@ubuntu:/opt/machbase# cd /etc/init.d
root@ubuntu:/etc/init.d# sudo service machbased
Usage: /etc/init.d/machbased {start|stop|kill|restart|createdb|destroydb|check|MWA|console|port|exe|collector|help}
root@ubuntu:/etc/init.d#
```

Start Server

Start the Machbase server. It is the same as machadmin -u.

```
[root@localhost ~]$ sudo service machbased start
------------------------------------------------------------------
     Machbase Administration Tool
     Release Version - x.x.x.official
     Copyright 2014, MACHBASE Corp. or its subsidiaries
     All Rights Reserved
------------------------------------------------------------------
Waiting for Machbase server start.
Machbase server started successfully.
[root@localhost ~]$
```

Shut Down Server

Shut down the Machbase server normally. It is the same as machadmin -s.

```
[root@localhost ~]$ sudo service machbased stop
------------------------------------------------------------------
     Machbase Administration Tool
     Release Version - x.x.x.official
     Copyright 2014, MACHBASE Corp. or its subsidiaries
     All Rights Reserved
------------------------------------------------------------------
Waiting for Machbase server shut down...
Machbase server shut down successfully.
[root@localhost ~]$
```

Stop Server

Force the Machbase server to abort. It is the same as machadmin -k.

```
[root@localhost ~]$ sudo service machbased kill
------------------------------------------------------------------
     Machbase Administration Tool
     Release Version - x.x.x.official
     Copyright 2014, MACHBASE Corp. or its subsidiaries
     All Rights Reserved
------------------------------------------------------------------
Waiting for Machbase server terminated.
Machbase server terminated successfully.
```

```
[root@localhost ~]$
```

Restart Server

Shut down normally and re-start the Machbase server.

```
[root@localhost ~]$ sudo service machbased restart
----------------------------------------------------------------
     Machbase Administration Tool
     Release Version - x.x.x.official
     Copyright 2014, MACHBASE Corp. or its subsidiaries
     All Rights Reserved
----------------------------------------------------------------
Waiting for Machbase server shut down...
Machbase server shut down successfully.
----------------------------------------------------------------
     Machbase Administration Tool
     Release Version - x.x.x.official
     Copyright 2014, MACHBASE Corp. or its subsidiaries
     All Rights Reserved
----------------------------------------------------------------
Waiting for Machbase server start.
Machbase server started successfully.
[root@localhost ~]$
```

Create Database

Create a Machbase database. It is the same as machadmin -c.

```
[root@localhost ~]$ sudo service machbased createdb
----------------------------------------------------------------
     Machbase Administration Tool
     Release Version - x.x.x.official
     Copyright 2014, MACHBASE Corp. or its subsidiaries
     All Rights Reserved
----------------------------------------------------------------
Database created successfully.
[root@localhost ~]$
```

Delete Database

Delete the Machbase database. It is the same as machadmin -d.

```
[root@localhost ~]$ sudo service machbased destroydb
----------------------------------------------------------------
     Machbase Administration Tool
     Release Version - x.x.x.official
     Copyright 2014, MACHBASE Corp. or its subsidiaries
     All Rights Reserved
----------------------------------------------------------------
Destroy Machbase database. Are you sure?(y/N) y
Database destoryed successfully.
[root@localhost ~]$
```

Check Server Status

Check the status of the Machbase server operation. It is the same as machadmin -e.

```
[root@localhost ~]$ sudo service machbased check
----------------------------------------------------------------
     Machbase Administration Tool
```

```
     Release Version - x.x.x.official
     Copyright 2014, MACHBASE Corp. or its subsidiaries
     All Rights Reserved
--------------------------------------------------------------------
Machbase server is running with PID(23542).
[root@localhost ~]$
```

MWA Management

These are the commands related to the MWA (Machbase Web Analytics) web server.

```
[root@localhost ~]$ sudo service machbased  MWA
start | restart | stop | reset | port

# Start the MWA server.
[root@localhost ~]$ sudo service machbased MWA start
SERVER ALREADY STARTED
Connection URL : http://192.168.0.10:5001

# Shut down the MWA server.
[root@localhost ~]$ sudo service machbased MWA stop
SERVER STOPPED

# Change the MWA server port.
[root@localhost ~]$ sudo service machbased MWA port 5050
WEBSERVER PORT CHANGED : 5050

# Shut down and restart the MWA server.
[root@localhost ~]$ sudo service machbased MWA restart
SERVER IS RESTARTING
SERVER STOPPED
SERVER STARTED, PID : 23810
     Connection URL : http://192.168.0.10:5001
[root@localhost ~]$
```

Change Server Port

Change the port of the Machbase server. If you execute the command and enter the port to be changed, it changes to that port. After the port change, you must restart the Machbase server to apply

```
[root@localhost ~]$ sudo service machbased port
The default port for the Machbase server is 5656. If you want to use 5656 as Machbase server port, press return key
Use current port.
[root@localhost ~]$
```

Collector Management

These are the commands to manage Machcollector.

```
[root@localhost ~]$ sudo service machbased collector
List of commands:
  * machbased collector start
    Machcollectormanager starts.
  * machbased collector stop
    Machcollectormanager shutdown.
  * machbased collector kill
    Terminate Machcollectormanager.
  * machbased collector destroy
    Destroy Machcollectormanager meta data.
  * machbased collector add_server
    Add an Machbase server to Machcollectormanager.
  * machbased collector port
    Change the default port. Now: 9999
```

```
[root@localhost ~]$
```

Installing Docker

Installing Docker

Machbase provides a Docker image. Assuming that Docker is already installed, use the following process of installing Machbase on Docker.

> To install Docker , refer to the Docker Installation Page . Docker Hub for Machbase is available on this page .

```
$ docker pull machbase/machbase
Using default tag: latest
latest: Pulling from machbase/machbase
3a291d7fe8d1: Pull complete
f1e7bd0ef2d1: Pull complete
78632f9cbb53: Pull complete
f4f6c5358244: Pull complete
a3e04b27f9cd: Pull complete
a3ed95caeb02: Pull complete
e03e135c0eda: Pull complete
26612cd7ebc1: Pull complete
b61e71cf4bc2: Pull complete
09c9c411b936: Pull complete
2b1cdec8c664: Pull complete
fd9a9a288691: Pull complete
d8852dedc8a1: Pull complete
cba7e30dbb6f: Pull complete
c7ead0fa7c49: Pull complete
6af02fe4c01f: Pull complete
d18db958464f: Pull complete
1fb93627ec0f: Pull complete
265b8b73294a: Pull complete
f122e6396b46: Pull complete
3b2f248fb414: Pull complete
07ed5a8f0935: Pull complete
44ec57c5ed31: Pull complete
59383e5f4c61: Pull complete
542101ec7002: Pull complete
Digest: sha256:aa6a982d35946b3fb33930de91cad61bfe7d3e9a559080526ed8e9a511c82c2b
Status: Downloaded newer image for machbase/machbase:latest
```

```
# Check the installed Machbase image.
$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
machbase/machbase   latest              dfb90844e7da        2 months ago        1.09 GB
```

```
# Execute the Machbase image.
$ docker run -it machbase/machbase
------------------------------------------------------------------
     Machbase Administration Tool
     Release Version - x.x.x.community
     Copyright 2014, MACHBASE Corp. or its subsidiaries
     All Rights Reserved
------------------------------------------------------------------
Database created successfully.
------------------------------------------------------------------
     Machbase Administration Tool
     Release Version - x.x.x.community
     Copyright 2014, MACHBASE Corp. or its subsidiaries
     All Rights Reserved
------------------------------------------------------------------
Waiting for Machbase server start.
Machbase server started successfully.
SERVER HAS BEEN RESET
```

```
SERVER STARTED, PID : 56
     Connection URL : http://172.17.0.2:5001
machbase@5ba45a22d140:~$
```

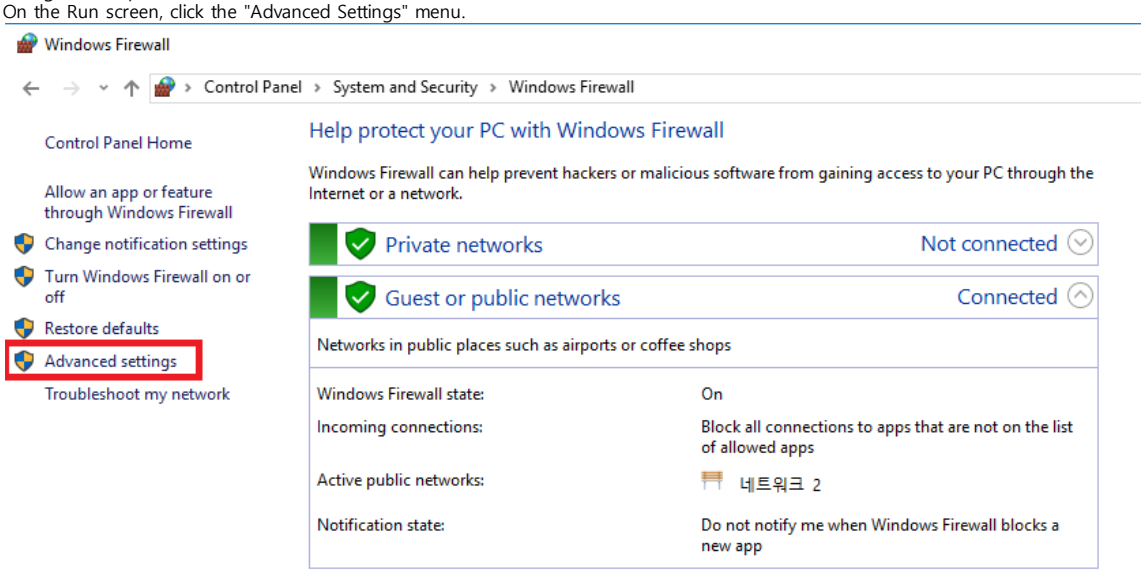
Windows Installation

Preparing Windows Environment for Installation

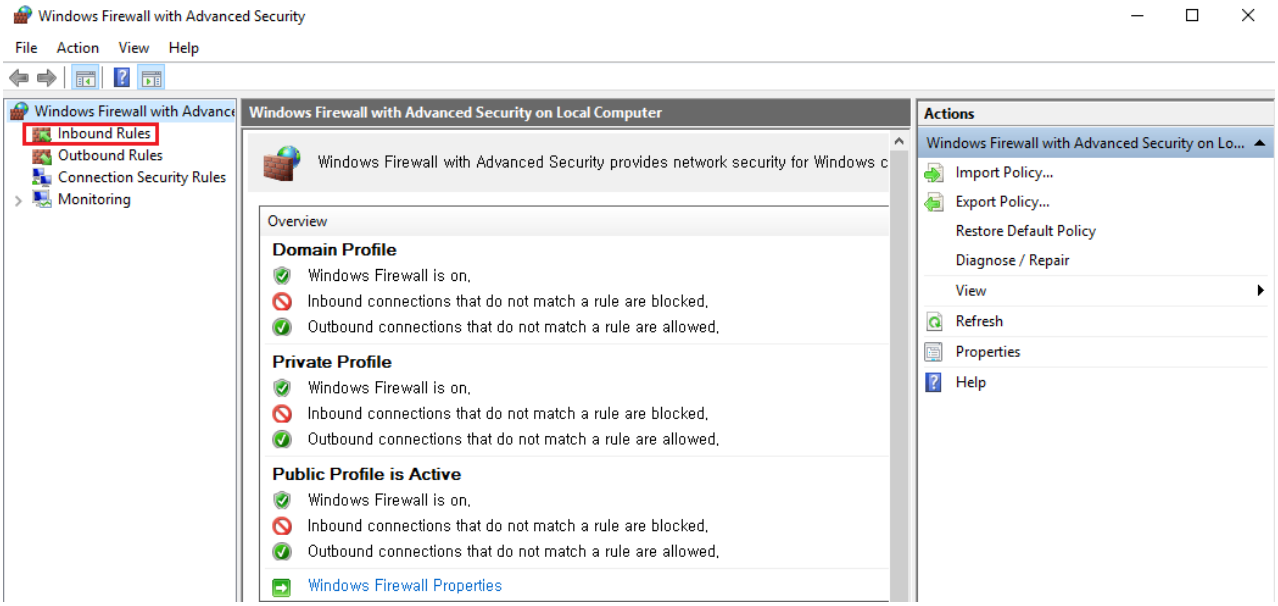
Open Firewall Port

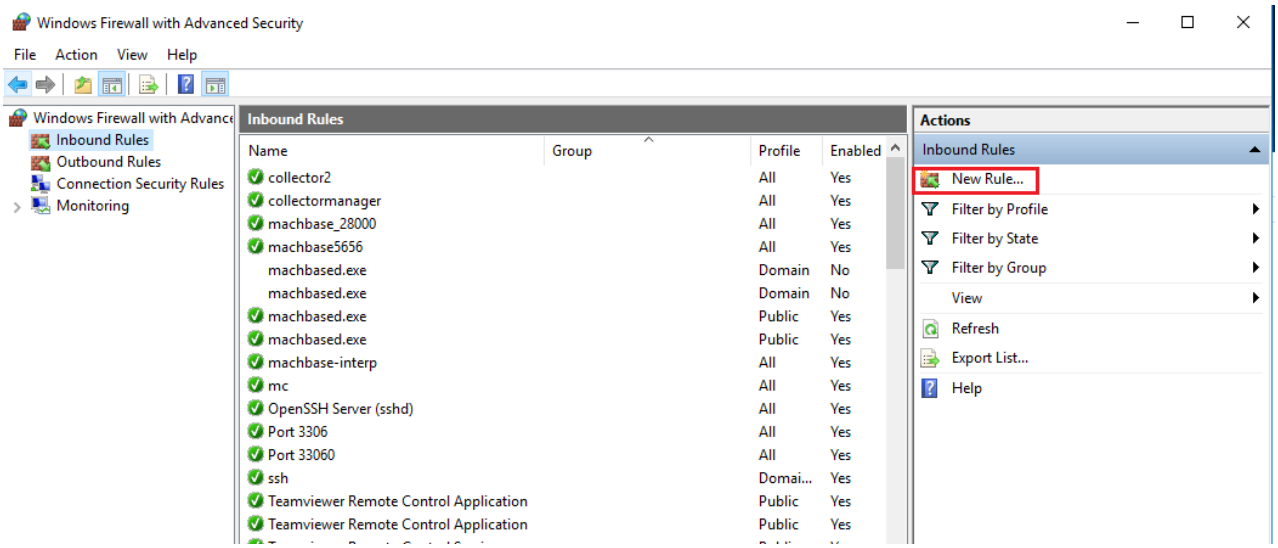
If you install Machbase in Windows, you must open the port that Machbase uses in the Windows Firewall.
In general, Machbase uses two ports: **5656 and 5001** .

1. To register the port on the firewall , select **Control Panel - Windows Firewall** or **Windows Defender Firewall**.
   On the Run screen, click the "Advanced Settings" menu.

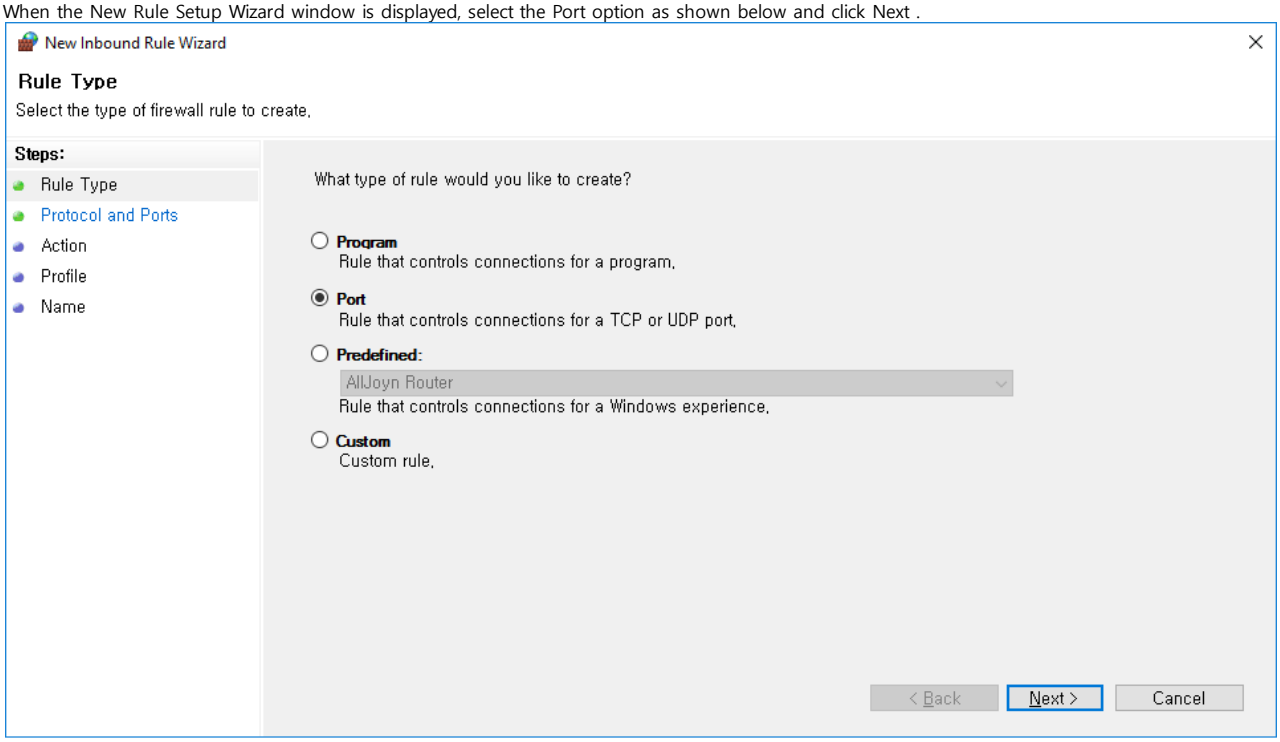


2. Under Advanced Settings, select **Inbound Rules - New Rule** and click.

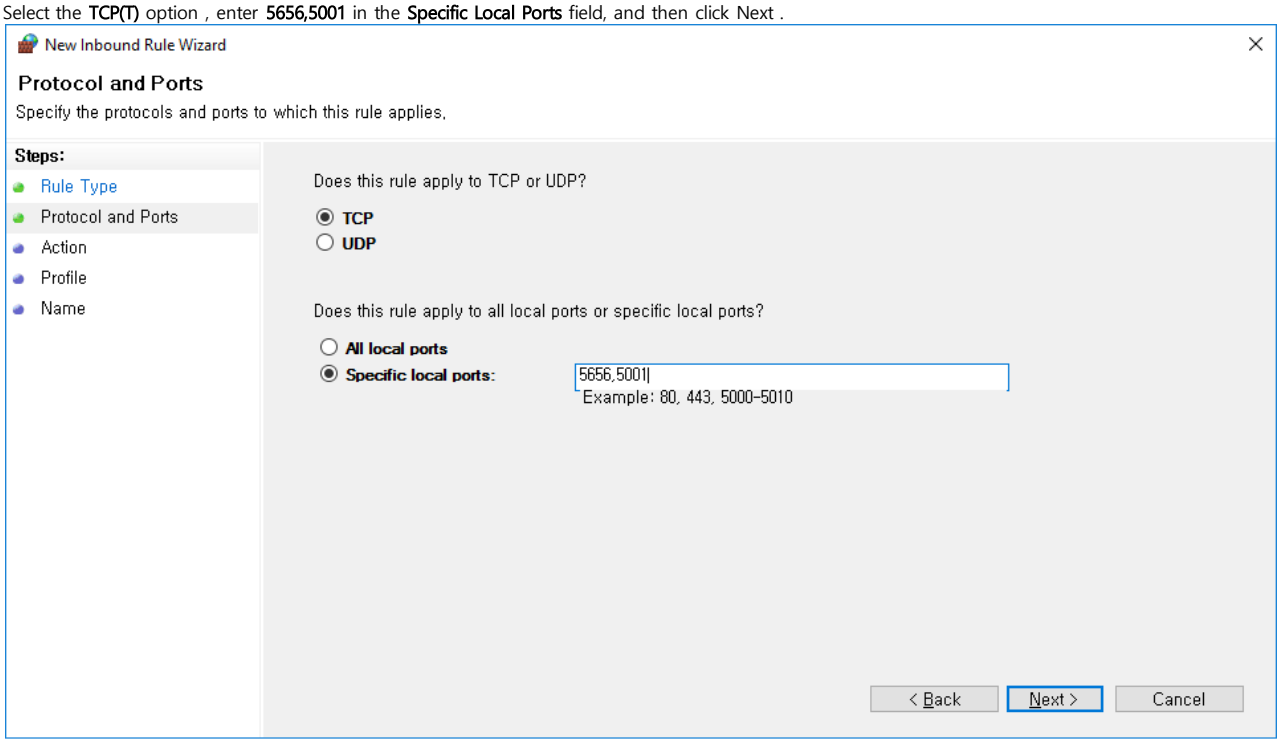




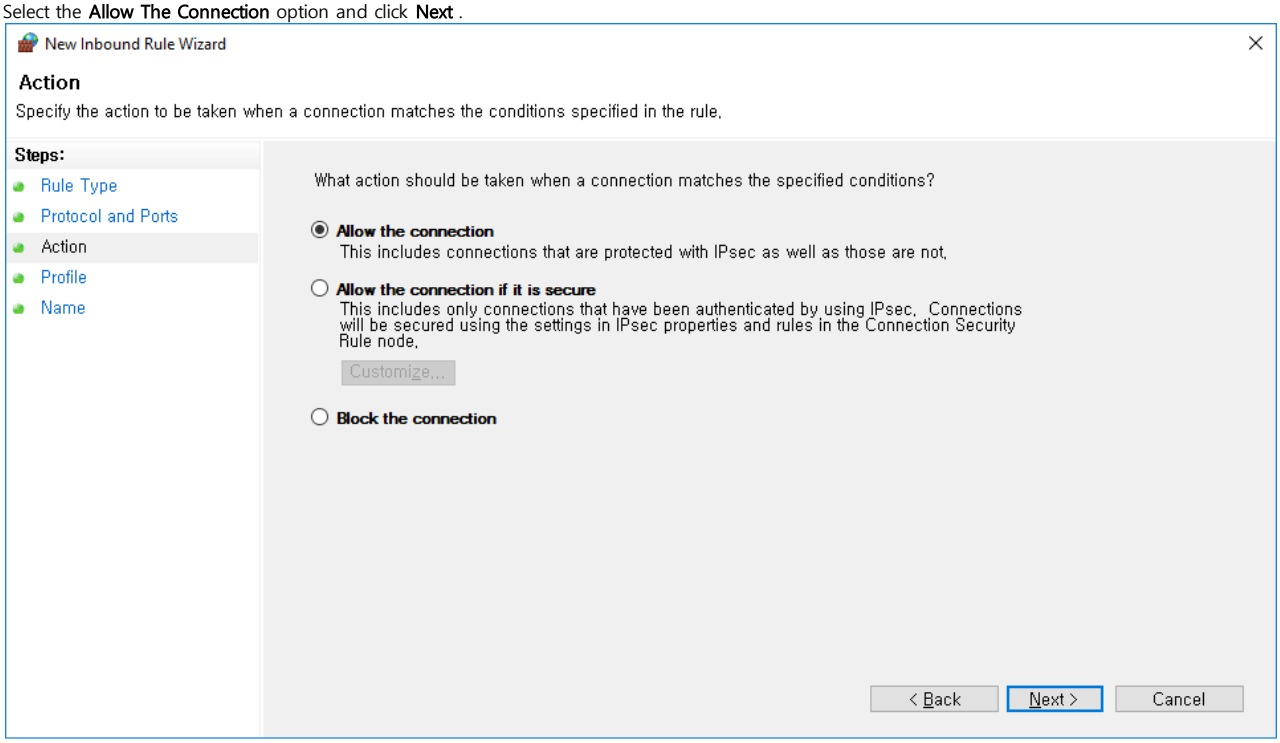
3. When the New Rule Setup Wizard window is displayed, select the Port option as shown below and click Next .



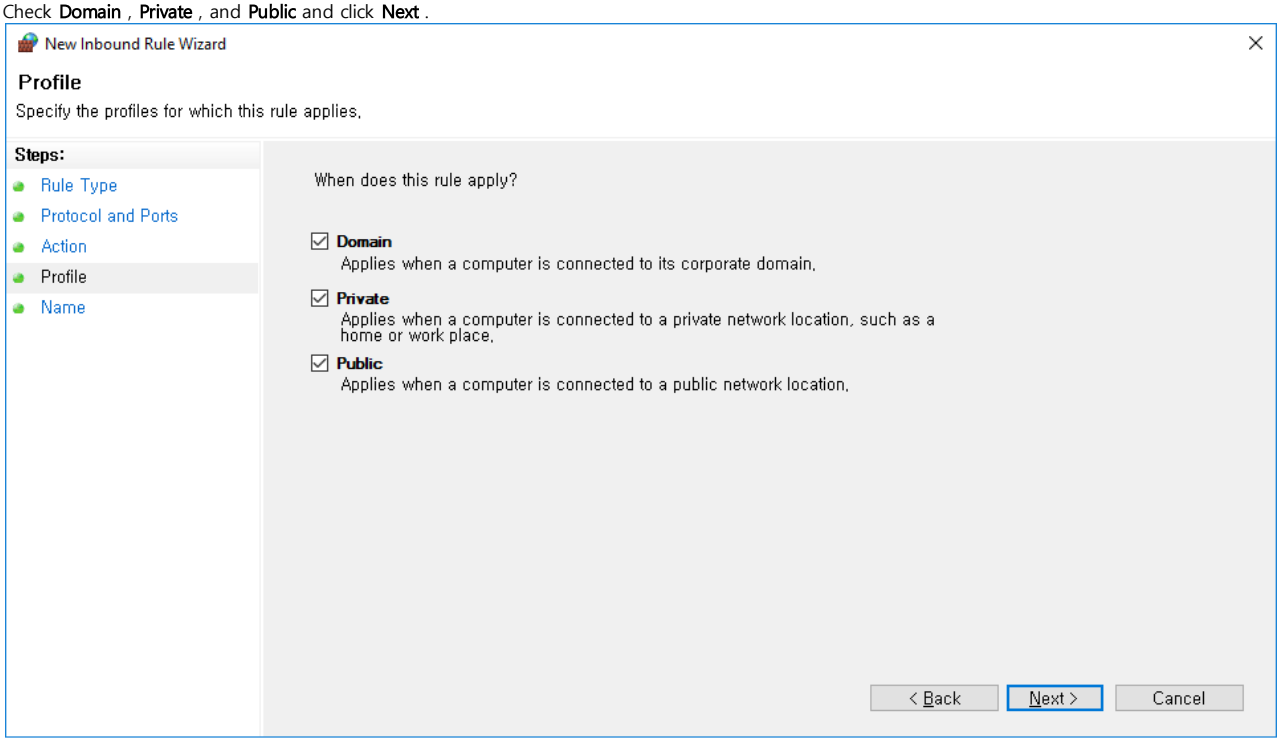
4. Select the **TCP(T)** option , enter **5656,5001** in the **Specific Local Ports** field, and then click Next .



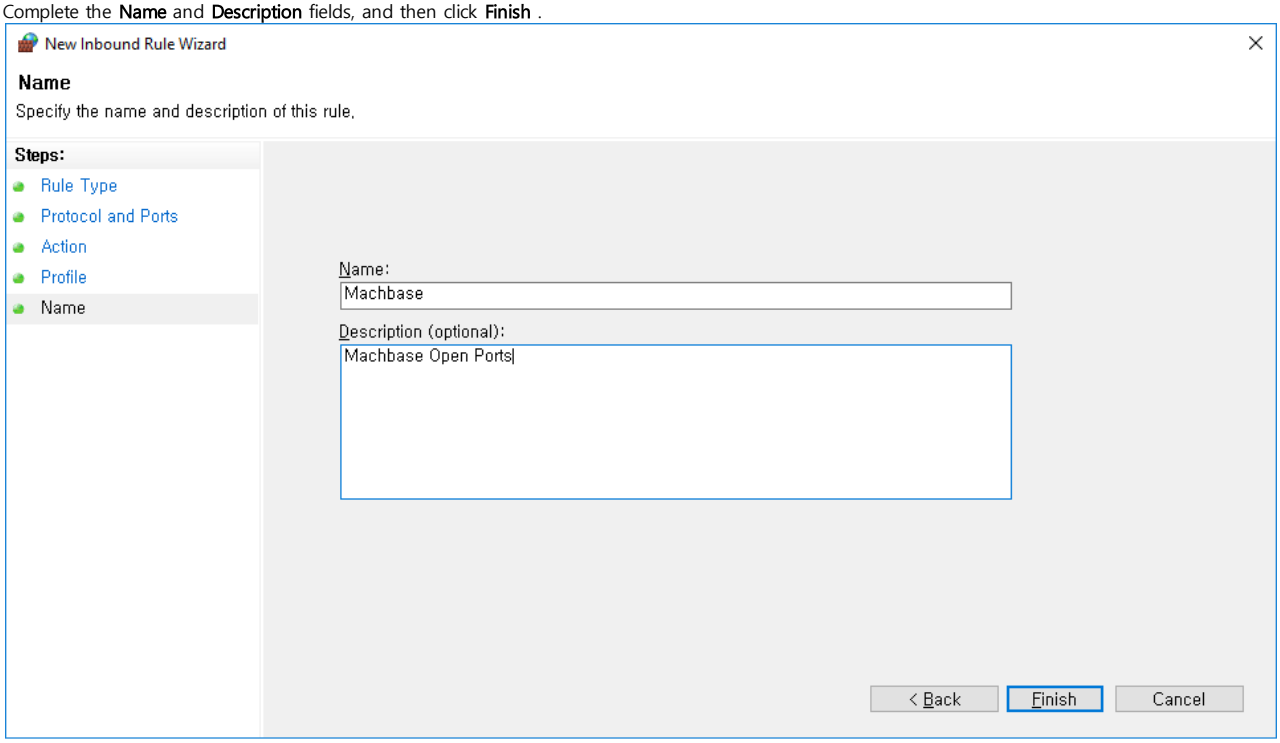
5. Select the **Allow The Connection** option and click **Next** .



6. Check **Domain** , **Private** , and **Public** and click **Next** .

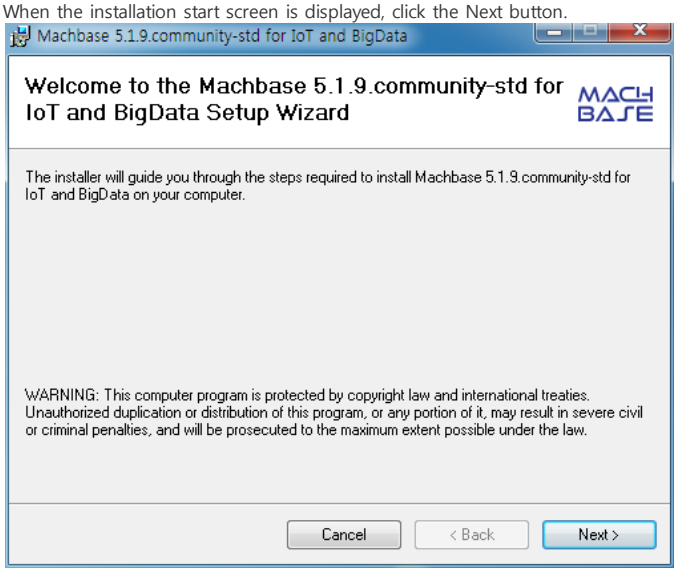


7. Complete the **Name** and **Description** fields, and then click **Finish** .



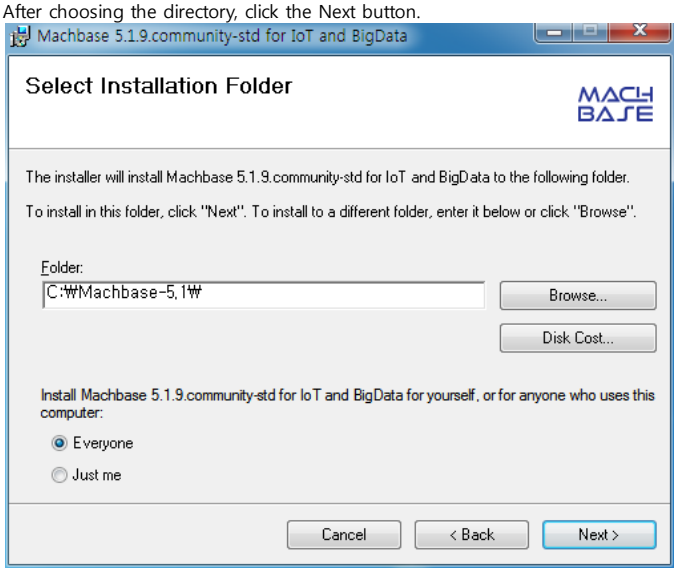
MSI Installation

1. When the installation start screen is displayed, click the Next button.

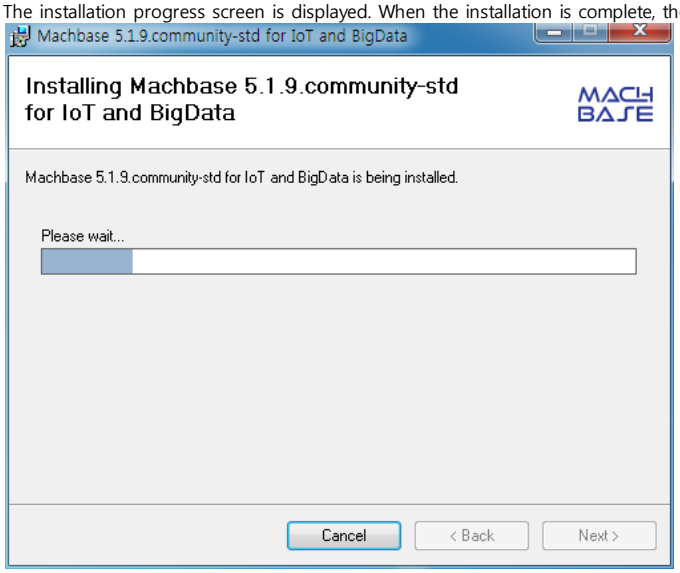


2. On the screen to select the directory folder, the default is usually `"C:\Machbase-5.1\"`.
   If you want to install to a different directory, you can change the path.

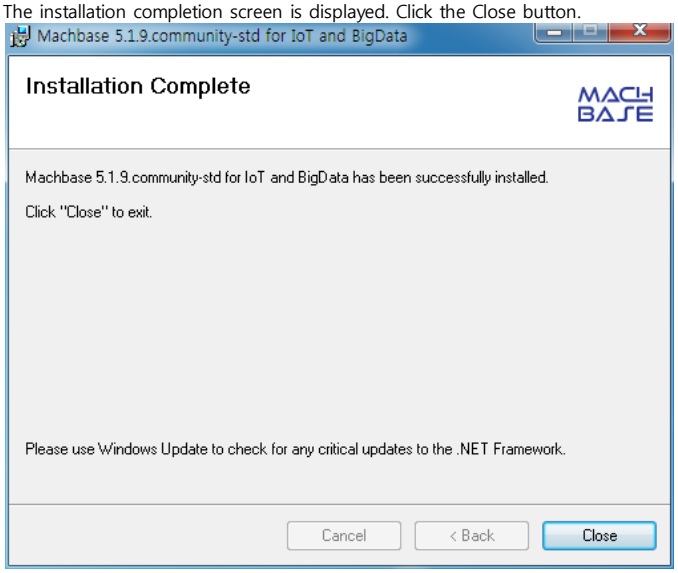
   After choosing the directory, click the Next button.



3. The installation progress screen is displayed. When the installation is complete, the Next button is activated. Click this button.



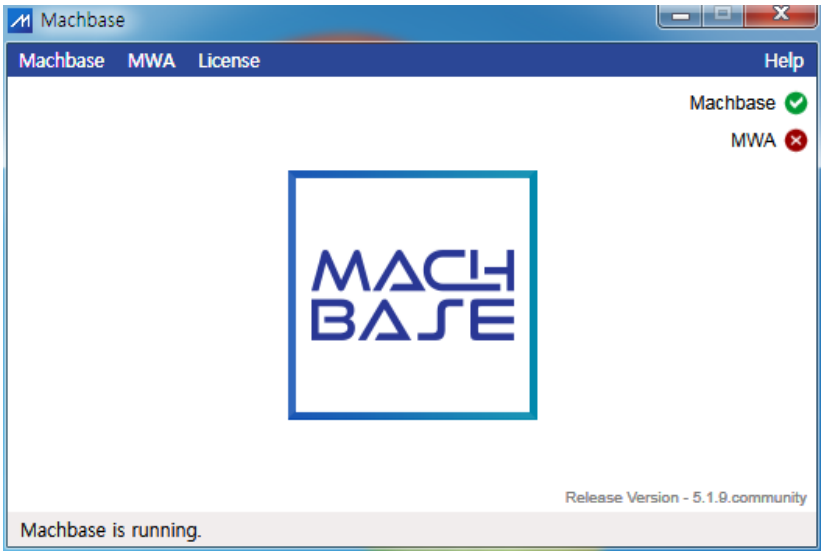
4. The installation completion screen is displayed. Click the Close button.



Launch Machbase

1. When the Machbase installation is completed, the Machbase shortcut icon is displayed on the desktop.
Double-click to run the Machbase server.

2. This is the window interface screen for managing the Machbase server.
You can control the Machbase server and the MWA Web server by clicking the menu.


Machbase
Machbase MWA License Help
Machbase
MWA
MACH
BASE
Release Version - 5.1.9.community
Machbase is running.

License Installation

License key installation is usually performed after MACHBASE installation is finished. If you do not install a specific license after installation, you can use MACHBASE with some restrictions. This section describes MACHBASE's license policy, structure, and installation method.

Index

License File Structure

The MACHBASE license is managed in the license.dat file. Licenses purchased for the product or for testing are displayed in a text file.

```
mach@localhost:~$ cat license.dat

\#Company\#ID-ProjectName: test\#0-Machbase
 \#License Policy: SIZE4DAY
 \#License Type \(Version 2\): OFFICIAL
 \#Issue DATE: 20160216
 \#Expiry DATE: 20160319
 VBz5h4TC-d3+Bf3Efkpdp/Tx873PpZA-78LRSdrxbPY-xhGf4355/iXaY5/jfnn+Jdpjn+N+ef4l2m
```

No License File

The server will run even if there is no license, but there are some limitations. The server can only be used for evaluation purposes, so if you intend to use it formally, you must obtain the license in a legitimate procedure.

If there is no license file, the following functional limitations will exist.

1. If you enter more than 100 million records through the Append protocol in one session, a warning message is displayed. Append input is then stopped. The input limit state is released only when the server is restarted.
2. When creating a tablespace, you can not create more than two disk directories. If you use more than one disk, the following warning indicating that the parallel I / O function for high performance data input can not be used will be displayed.

```
CREATE TABLESPACE tbs1 DATADISK disk1 (disk_path="tbs1_disk1"), disk2 (disk_path="tbs1_disk2"), disk3 (disk_path="
[ERR-00867 : Error in adding disk to tablespace. You cannot use multiple disks for tablespace without valid license
```

License Installation

The MACHBASE license must be installed in *$MACHBASE_HOME/conf*, and the default name is *license.dat* .

Copy the License File to $ MACHBASE_HOME / conf

At this time, the name of the license file issued must be changed to **license.dat** and copied. Then, when the server is started, it will determine if the license is appropriate and begin the installation.

Launch `machadmin -t 'licensefile_path'`

The advantage of this method is that it can be easily installed with commands without having to adjust the license file name or location.
Installing as a query: This is a way to install a license using a query statement while the server is running.

Verifying License Installation

License Installed

If the license file is installed, the following is displayed in machbase.trc after the server is started.

```
[2016-02-17 14:51:00 P-20913 T-140709874054912][INFO] LICENSE [License Type (Version 2)][OFFICIAL]
[2016-02-17 14:51:00 P-20913 T-140709874054912][INFO] LICENSE [License Policy] [CORE]
[2016-02-17 14:51:00 P-20913 T-140709874054912][INFO] LICENSE [Host ID] [FFFFFFFFFFFFFF]
[2016-02-17 14:51:00 P-20913 T-140709874054912][INFO] LICENSE [Expiry DATE] [25300318]
```

```
[2016-02-17 14:51:00 P-20913 T-140709874054912][INFO] Machbase Logs Signature! : OFFICIAL:CORE:FFFFFFFFFFFFFF:253(
```

You can also use the machadmin -f command.

License Not Installed

If the license file is not installed or if an abnormal file is used, the following output is displayed.

```
[2016-02-17 14:49:54 P-6620 T-140539052701440][INFO] LICENSE [License Type(Version 2)][Only for evaluation (No lice
[2016-02-17 14:49:54 P-6620 T-140539052701440][INFO] LICENSE [License Policy] [None]
[2016-02-17 14:49:54 P-6620 T-140539052701440][INFO] LICENSE [Host ID] [Unknown]
[2016-02-17 14:49:54 P-6620 T-140539052701440][INFO] LICENSE [Expiry DATE] [N/A]
```

Cluster Edition Installation

Preparing for Cluster Edition Installation

Confirm and Change File LIMIT

To increase the maximum number of files that can be opened, do the following.

1. Modify the file `/etc/security/limits.conf`

```
sudo vi /etc/security/limits.conf
*       hard    nofile      65535
*       soft    nofile      65535
```

2. `Reboot.`

```
sudo reboot
# or
sudo shutdown -r now
```

3. Check the results. If the output is 65535, it has been successfully changed.

```
ulimit -Sn
```

Index

Server Time Synchronization

You must synchronize the server time between each host. If they are already synchronized, check for confirmation.

1. Synchronize the time on all servers with the time server.

```
# Synchronize with the following command.
/usr/bin/rdate -s time.bora.net && /sbin/clock -w
```

2. If the time server can not be used, modify it with a direct command.

```
# Modify with the following command.
date -s "2017-10-31 11:15:30"
```

3. Check the modified time

```
# Check with the following command.
date
```

Change Network Kernel Parameters

1. Check the current set value.

```
Check with the following command.
sysctl -a | egrep 'mem_(max|default)|tcp_.*mem'
```

2. Change the setting value with the following command (for 64GB Memory).

```
sysctl -w net.core.rmem_default = "33554432"     # 32MB
sysctl -w net.core.wmem_default = "33554432"
sysctl -w net.core.rmem_max     = "268435456"    # 256MB
sysctl -w net.core.wmem_max     = "268435456"
sysctl -w net.ipv4.tcp_rmem     = "262144 33554432 268435456"
sysctl -w net.ipv4.tcp_wmem     = "262144 33554432 268435456"
```

```
# 8388608 Page * 4KB = 32GB
sysctl -w net.ipv4.tcp_mem      = "8388608 8388608 8388608"
```

3. To keep the changes, add them to the /etc/sysctl.conf file and restart the host OS.

```
# Modify the file /etc/sysctl.conf.
net.core.rmem_default = "33554432"
net.core.wmem_default = "33554432"
net.core.rmem_max     = "268435456"
net.core.wmem_max     = "268435456"
net.ipv4.tcp_rmem     = "262144 33554432 268435456"
net.ipv4.tcp_wmem     = "262144 33554432 268435456"
net.ipv4.tcp_mem      = "8388608 8388608 8388608"
```

Create User

1. Create a Linux OS user 'machbase' for Machbase installation. The user account directory is created as: /home/machbase.

```
$ sudo useradd machbase --home-dir "/home/machbase"
```

2. Set password (machbase)

```
sudo passwd machbase
```

Cluster Edition Installation(Command-line)

(1) Coordinator / Deployer Installation, Package Add

Installing Coordinator

Configuration

Log in with machbase account and execute with machbase privilege.
Configure installation directory and path information.

```
# Edit .bashrc.
export MACHBASE_COORDINATOR_HOME=~/coordinator
export MACHBASE_DEPLOYER_HOME=~/deployer
export MACHBASE_HOME=~/coordinator
export PATH=$MACHBASE_HOME/bin:$PATH
export LD_LIBRARY_PATH=$MACHBASE_HOME/lib:$LD_LIBRARY_PATH

# Reflect changes.
source .bashrc
```

Index

Create and Unzip Directory

Create a dedicated directory and unzip the package archive into that directory.

```
# Create directory.
mkdir $MACHBASE_COORDINATOR_HOME

# Unzip.
tar zxvf machbase-ent-x.y.z.official-LINUX-X86-64-release.tgz -C $MACHBASE_COORDINATOR_HOME
```

Port Configuration and Service Activation

Modify the machbase.conf file to set the port and start the service.

```
# Set port from machbase.conf file.
cd $MACHBASE_COORDINATOR_HOME/conf
vi machbase.conf
CLUSTER_LINK_HOST       = 192.168.0.83 (Node ip to be added)
CLUSTER_LINK_PORT_NO    = 5101
HTTP_ADMIN_PORT         = 5102

# Create meta information and run service.
$MACHBASE_COORDINATOR_HOME/bin/machcoordinatoradmin -c
$MACHBASE_COORDINATOR_HOME/bin/machcoordinatoradmin -u
```

Node Registration and Verification

Add and confirm the Coordinator node.

```
# Register node.
$MACHBASE_COORDINATOR_HOME/bin/machcoordinatoradmin --add-node="192.168.0.83:5101" --node-type=coordinator

# Check node.
$MACHBASE_COORDINATOR_HOME/bin/machcoordinatoradmin --cluster-status
```

Optional	Description	Example
--add-node	Specifies the node name to be added as "IP: PORT". The PORT value is the CLUSTER_LINK_PORT_NO value.	192.168.0.83:5101
--node-type	Specifies the node type. There are four types: coordinator / deployer / broker / warehouse.	coordinator

Delete Coordinator

Connect to the server where the Coordinator is installed, terminate the Coordinator process properly, and delete the Coordinator directory.

```
# Terminate coordinator and delete directory.
process$MACHBASE_COORDINATOR_HOME/bin/machcoordinatoradmin -s
rm -rf $MACHBASE_COORDINATOR_HOME
```

Secondary Coordinator Installation

If you install an additional Coordinator in addition to the Primary Coordinator, note the following:

- Before the Startup of the Secondary Coordinator, you must go to the Primary Coordinator and Add-Node the Secondary Coordinator.
- When you start the Secondary Coordinator, you must specify the Primary Coordinator as the **`--primary`** option.
- Do not add-node the Primary Coordinator to the Secondary Coordinator.

If this is not followed, the Secondary Coordinator will behave like a Primary Coordinator.

Create and Unzip Directory

Create a dedicated directory and unzip the package archive into that directory.

```
# Create directory.
mkdir $MACHBASE_COORDINATOR_HOME

# Unzip.
tar zxvf machbase-ent-x.y.z.official-LINUX-X86-64-release.tgz -C $MACHBASE_COORDINATOR_HOME
```

Port Settings

Modify the machbase.conf file to set the port only. **When the service starts, it will work like the Primary Coordinator.**

```
# Set port from machbase.conf file.
cd $MACHBASE_COORDINATOR_HOME/conf
vi machbase.conf
CLUSTER_LINK_HOST       = 192.168.0.83 (ip address to be added)
CLUSTER_LINK_PORT_NO    = 5111
HTTP_ADMIN_PORT         = 5112
```

Node Registration and Verification

In the Primary Coordinator, add and confirm the Secondary Coordinator node.

```
# Register node.
$MACHBASE_COORDINATOR_HOME/bin/machcoordinatoradmin --add-node="192.168.0.83:5111" --node-type=coordinator

# Check node.
$MACHBASE_COORDINATOR_HOME/bin/machcoordinatoradmin --cluster-status
```

Run Service

Now run the Secondary Coordinator. During startup, the Primary Coordinator must be specified as a **`--primary`** option.

```
# Create meta information and run service.
$MACHBASE_COORDINATOR_HOME/bin/machcoordinatoradmin -c
$MACHBASE_COORDINATOR_HOME/bin/machcoordinatoradmin -u --primary="192.168.0.83:5101"
```

Delete Secondary Coordinator

After removing the Secondary Coordinator registered in the Primary Coordinator, the Secondary Coordinator must be terminated properly.

```
# Delete node.
$MACHBASE_COORDINATOR_HOME/bin/machcoordinatoradmin --remove-node="192.168.0.83:5101"

# Terminate secondary coordinator and delete directory.
$MACHBASE_COORDINATOR_HOME/bin/machcoordinatoradmin -s
rm -rf $MACHBASE_COORDINATOR_HOME

# Check node.
$MACHBASE_COORDINATOR_HOME/bin/machcoordinatoradmin --cluster-status
```

Optional	Description	Example
--remove-node	Specifies the name of the node to be deleted as "IP: PORT". The PORT value is the CLUSTER_LINK_PORT_NO value.	192.168.0.84:5201

Deployer Installation

> **Information**
>
> The Deployer must be installed in advance on all hosts (= servers) where the broker and warehouse are installed

Configuration

Configure installation directory and path information.

```
# Edit .bashrc.
export MACHBASE_DEPLOYER_HOME=~/deployer
export MACHBASE_HOME=~/deployer
export PATH=$MACHBASE_HOME/bin:$PATH
export LD_LIBRARY_PATH=$MACHBASE_HOME/lib:$LD_LIBRARY_PATH

# Reflect changes.
source .bashrc
```

Create and Unzip Directory

Create a dedicated directory and unzip the package archive into that directory.

```
# Create directory.
mkdir $MACHBASE_DEPLOYER_HOME

# Unzip.
tar zxvf machbase-ent-x.y.z.official-LINUX-X86-64-release.tgz -C $MACHBASE_DEPLOYER_HOME
```

Port Configuration and Service Activation

Modify the machbase.conf file to set the port and start the service.

```
# Set port from machbase.conf file.
```

```
cd $MACHBASE_DEPLOYER_HOME/conf
vi machbase.conf
CLUSTER_LINK_HOST       = 192.168.0.84
CLUSTER_LINK_PORT_NO    = 5201
HTTP_ADMIN_PORT         = 5202

# Create meta information and run service.
$MACHBASE_DEPLOYER_HOME/bin/machdeployeradmin -c
$MACHBASE_DEPLOYER_HOME/bin/machdeployeradmin -u
```

Node Registration and Verification

> **Caution**
>
> This should be done at the coordinator node.

Add and verify the Deployer node.

```
# Register node.
$MACHBASE_COORDINATOR_HOME/bin/machcoordinatoradmin --add-node="192.168.0.84:5201" --node-type=deployer

# Check node.
$MACHBASE_COORDINATOR_HOME/bin/machcoordinatoradmin --cluster-status
```

Optional	Description	Example
--add-node	Specifies the node name to be added as "IP: PORT". The PORT value is the CLUSTER_LINK_PORT_NO value.	192.168.0.84:5201
--node-type	Specifies the node type. There are four types: coordinator / deployer / broker / warehouse.	deployer

Delete Deployer

You must delete the Deployer node from the Coordinator node and properly terminate the Deployer process on the server where the Deployer is located.

```
# Delete node.
$MACHBASE_COORDINATOR_HOME/bin/machcoordinatoradmin --remove-node="192.168.0.84:5201"

# Terminate deployer and delete directory.
$MACHBASE_DEPLOYER_HOME/bin/machdeployeradmin -d
rm -rf $MACHBASE_DEPLOYER_HOME

# Check node.
$MACHBASE_COORDINATOR_HOME/bin/machcoordinatoradmin --cluster-status
```

Add Package

Add a package to be installed as broker and warehouse to Coordinator. At this time, the registered package registers the lightweight version excluding MWA.

```
# Register installation package.
$MACHBASE_COORDINATOR_HOME/bin/machcoordinatoradmin --add-package=machbase \
    --file-name="/home/machbase/machbase-ent-x.y.z.official-LINUX-X86-64-release-lightweight.tgz"
```

Optional	Description	Example
--add-package	Specifies the package name to be added.	machbase

Optional	Description	Example
--file-name	Specifies the full path and file name of the package file. Specifies a lightweight package that excludes MWA files, since this package is for broker and warehouse installations only.	/home/machbase/machbase-ent-5.0.0.official-LINUX-X86-64-release-lightweight.tgz

Delete Package

Delete the package registered in Coordinator.

```
# Delete registered package.
$MACHBASE_COORDINATOR_HOME/bin/machcoordinatoradmin --remove-package=machbase
```

(2) Broker / Warehouse Installation

Broker Installation

Add a broker node to the coordinator node. Multi-broker node registration is possible.

The deployer node must be installed on the server in advance.
Once the deployer node is installed, there is no setting by connecting to the server as all work can be done on the coordinator node.
The first registered node becomes the leader broker, and the additional registered node becomes the follower broker.

```
# Add broker node.
$MACHBASE_COORDINATOR_HOME/bin/machcoordinatoradmin --add-node="192.168.0.84:53
        --node-type=broker --deployer="192.168.0.84:5201" --port-no="5656"
        --home-path="/home/machbase/broker" --package-name=machbase

# Run broker node.
$MACHBASE_COORDINATOR_HOME/bin/machcoordinatoradmin --startup-node="192.168.0.8
```

Option Items	Description	Example
--add-node	Specifies the node name to be added as "IP: PORT". The PORT value is set to the CLUSTER_LINK_PORT_NO value.	192.168.0.84:5301
--node-type	Specifies the node type. There are four types: coordinator, deployer, broker, and warehouse.	broker
--deployer	Register the deployer node information of the server to be installed.	192.168.0.84:5201
--port-no	Specifies 'machbased' port. The Broker specifies a default value of 5656. This port is used when connecting to client and machsql.	5656
--home-path	Specifies the path to install. Specifies /home/machabse/broker in machbase account	/home/machabse/broker
--package-name	Sets the package name specified when package was added.	machbase

Index

Delete Broker

Remove the broker node from the Coordinator node.

```
# Delete broker node.
$MACHBASE_COORDINATOR_HOME/bin/machcoordinatoradmin --remove-node="192.168.0.84:5301"
```

Shut Down / Stop Broker

There is a way to shut down / kill the broker node on the Coordinator node.

```
# Terminate broker node.
$MACHBASE_COORDINATOR_HOME/bin/machcoordinatoradmin --shutdown-node="192.168.0.84:5301"

# Stop broker node.
$MACHBASE_COORDINATOR_HOME/bin/machcoordinatoradmin --kill-node="192.168.0.84:5301"
```

Alternatively, you can shut down / kill the process directly from the server where the broker is installed.

```
# Terminate broker node.
$MACHBASE_HOME/bin/machadmin -s

# Stop broker node.
$MACHBASE_HOME/bin/machadmin -k
```

Warehouse Installation

Install the active node and the standby node from the Coordinator node.

They will be installed through a pre-installed deployer.

Group 1 Installation

Install the first Warehouse Group1 node.

```
# Install group1 warehouse.
$MACHBASE_COORDINATOR_HOME/bin/machcoordinatoradmin --add-node="192.168.0.83:5401"  \
        --node-type=warehouse --deployer="192.168.0.83:5201" --port-no="5400"       \
        --home-path="/home/machbase/warehouse_g1" --package-name=machbase           \
        --replication="192.168.0.83:5402"  --group="group1" --no-replicate

# Run installed node.
$MACHBASE_COORDINATOR_HOME/bin/machcoordinatoradmin --startup-node="192.168.0.84:5401"
```

Option Items	Description	Example
--add-node	Specifies the node name to be added as "IP: PORT". The PORT value is set to the CLUSTER_LINK_PORT_NO value.	192.168.0.84:5401
--node-type	Specifies the node type. There are four types: coordinator, deployer, broker, and warehouse.	warehouse
--deployer	Registers the deployer node information of the server to be installed.	192.168.0.84:5201
--port-no	Specifies the working port of 'machbased'. Since the value was set to 5656 on the Broker, a different port must be specified if it is installed on the same server. The warehouse port number is set as 5400. This port is used when connecting to client and machsql.	5400
--home-path	Specifies the path to install. To distinguish the groups, set them in order of warehouse_g1, g2, g3.	/home/machbase/warehouse_g1
--package-name	Sets the package name specified when adding the package.	machbase
--replication	Specifies the node in charge of replication as "IP: PORT". The port value is set to the warehouse port number 5402.	192.168.0.84:5402
--group	Specifies the Group name.	group1
--no-replicate	Specifies whether to replicate data when adding a node if there is warehouse data in the group.	
--set-group-state	Specifies the state of the group as normal and readonly. Normal is read, write / readonly is read only	

Add Node to Group 1

Add another node to Warehouse Group1.

```
# Add warehouse node to group1.
$MACHBASE_COORDINATOR_HOME/bin/machcoordinatoradmin --add-node="192.168.0.84:5401"  \
        --node-type=warehouse --deployer="192.168.0.84:5201" --port-no="5400"       \
        --home-path="/home/machbase/warehouse_g1" --package-name=machbase           \
        --replication="192.168.0.84:5402" --group="group1" --no-replicate

# Run installed node.
```

```
$MACHBASE_COORDINATOR_HOME/bin/machcoordinatoradmin --startup-node="192.168.0.84:5401"
```

Option Items	Description	Example
--add-node	Specifies the node name to be added as "IP: PORT". The PORT value is set to the CLUSTER_LINK_PORT_NO value.	192.168.0.84:5401
--node-type	Specifies the node type. There are four types: coordinator, deployer, broker, and warehouse.	warehouse
--deployer	Registers the deployer node information of the server to be installed.	192.168.0.84:5201
--port-no	Specifies the working port of 'machbased'. Since the value was set to 5656 on the Broker, a different port must be specified if it is installed on the same server. The warehouse port number is set as 5400. This port is used when connecting to client and machsql.	5400
--home-path	Specifies the path to install. To distinguish the groups, set them in order of warehouse_g1, g2, g3.	/home/machabse/warehouse_g1
--package-name	Sets the package name specified when adding the package.	machbase
--replication	Specify the node in charge of replication as "IP: PORT". The port value is set to the warehouse port number 5402.	192.168.0.84:5402
--group	Specifies the Group name.	group1
--no-replicate	Specifies whether to replicate data when adding a node if there is warehouse data in the group.	
--set-group-state	Specifies the state of the group as normal and readonly. Normal is read, write / readonly is read only	

Group 2 Installation

Install the second Warehouse Group2 node.

```
# Install group1 warehouse.
$MACHBASE_COORDINATOR_HOME/bin/machcoordinatoradmin --add-node="192.168.0.84:5411"  \
        --node-type=warehouse --deployer="192.168.0.84:5201" --port-no="5410"       \
        --home-path="/home/machbase/warehouse_g2" --package-name=machbase           \
        --replication="192.168.0.84:5412"  --group="group2" --no-replicate

# Run installed node.
$MACHBASE_COORDINATOR_HOME/bin/machcoordinatoradmin --startup-node="192.168.0.84:5411"
```

Option Items	Description	Example
--add-node	Specifies the node name to be added as "IP: PORT". The PORT value is set to the CLUSTER_LINK_PORT_NO value.	192.168.0.84:5411
--node-type	Specifies the node type. There are four types: coordinator, deployer, broker, and warehouse.	warehouse
--deployer	Registers the deployer node information of the server to be installed.	192.168.0.84:5201
--port-no	Specifies the working port of 'machbased'. Since the value was set to 5656 on the Broker, a different port must be specified if it is installed on the same server. The warehouse port number is set as 5410. This port is used when connecting to client and machsql.	5410
--home-path	Specifies the path to install. To distinguish the groups, set them in order of warehouse_g1, g2, g3.	/home/machbase/warehouse_g2
--package-name	Sets the package name specified when adding the package.	machbase
--replication	Specifies the node in charge of replication as "IP: PORT". The port value is set to the warehouse port number 5412.	192.168.0.84:5412
--group	Specifies the Group name.	group2

Option Items	Description	Example
--no-replicate	Specifies whether to replicate data when adding a node if there is warehouse data in the group.	
--set-group-state	Specifies the state of the group as normal and readonly. Normal is read, write / readonly is read only	

Add Node to Group 2

Add another node to Warehouse Group2.

```
# Add warehouse node to group1.
$MACHBASE_COORDINATOR_HOME/bin/machcoordinatoradmin --add-node="192.168.0.83:5411"  \
        --node-type=warehouse --deployer="192.168.0.83:5201" --port-no="5410"       \
        --home-path="/home/machbase/warehouse_g2" --package-name=machbase           \
        --replication="192.168.0.83:5412" --group="group2" --no-replicate

# Run installed node.
$MACHBASE_COORDINATOR_HOME/bin/machcoordinatoradmin --startup-node="192.168.0.83:5411"
```

Option Items	Description	Example
--add-node	Specifies the node name to be added as "IP: PORT". The PORT value is set to the CLUSTER_LINK_PORT_NO value.	192.168.0.83:5411
--node-type	Specifies the node type. There are four types: coordinator, deployer, broker, and warehouse.	warehouse
--deployer	Registers the deployer node information of the server to be installed.	192.168.0.83:5201
--port-no	Specifies the working port of 'machbased'. Since the value was set to 5656 on the Broker, a different port must be specified if it is installed on the same server. The warehouse port number is set as 5400. This port is used when connecting to client and machsql.	5410
--home-path	Specifies the path to install. To distinguish the groups, set them in order of warehouse_g1, g2, g3.	/home/machbase/warehouse_g2
--package-name	Sets the package name specified when adding the package.	machbase
--replication	Specifies the node in charge of replication as "IP: PORT". The port value is set to the warehouse port number 5412.	192.168.0.83:5412
--group	Specifies the Group name.	group2
--no-replicate	Specifies whether to replicate data when adding a node if there is warehouse data in the group.	
--set-group-state	Specifies the state of the group as normal and readonly. Normal is read, write / readonly is read only	

Delete Warehouse

Delete the warehouse node from the Coordinator node.

```
# Delete warehouse node.
$MACHBASE_COORDINATOR_HOME/bin/machcoordinatoradmin --remove-node="192.168.0.83:5401"
```

Shut Down / Stop Warehouse

There is a way to shut down / kill the warehouse node at the Coordinator node.

```
# Terminate warehouse node.
$MACHBASE_COORDINATOR_HOME/bin/machcoordinatoradmin --shutdown-node="192.168.0.83:5401"

# Stop warehouse node.
```

```
$MACHBASE_COORDINATOR_HOME/bin/machcoordinatoradmin --kill-node="192.168.0.83:5401"
```

Otherwise, the process can be shut down / killed directly from the server where the warehouse is installed.

```
# Terminate warehouse node.
$MACHBASE_HOME/bin/machadmin -s

# Stop warehouse node.
$MACHBASE_HOME/bin/machadmin -k
```

Cluster Edition Installation by MWA

(1) MWA Installation

Cluster Edition Installation (MWA)

- (1) MWA Installation
- (2) Coordinator / Deployer Installation
- (3) Broker / Warehouse Installation

To install MWA, you need to unzip the package in the command-line environment.

Configuration

First, connect to the server, assuming you have the machbase / machbase account.

After that, configure environment for installation directory and path information.

```
export MACHBASE_HOME=~/coordinator
export PATH=$MACHBASE_HOME/bin:$PATH
export LD_LIBRARY_PATH=$MACHBASE_HOME/lib:$LD_LIBRARY_PATH
```

Index

Create and Unzip Coordinator Directory

Create a directory set as `$MACHBASE_HOME` and unzip the package file into that directory.

```
# Create directory.
mkdir $MACHBASE_HOME

# Download installation package and copy to MACHBASE_HOME.
cp -f machbase-ent-x.x.x.official-LINUX-X86-64-release.tgz $MACHBASE_HOME

# Go to MACHBASE_HOME and unzip.
cd $MACHBASE_HOME
tar zxvf machbase-ent-x.x.x.official-LINUX-X86-64-release.tgz
```

MWA Settings for Cluster Edition Installation

The Cluster Edition installation requires many entries. MWA provides default values for these items at installation time.

This default value can be changed to the value you want.
Modify the necessary parts of the file `$MACHBASE_HOME/webadmin/flask/MWA.conf` and restart the MWA.

The default values set in MWA.conf are as follows.

```
31
32  ################################################################################
33  # Default value in the node installation of cluster admin.
34  # - You can use $, # for the DEFAULT_[BROKER|WAREHOUSE]_HOME_PATH.
35  #   $ : Group Name
36  #   # : Warehouse index in same group
37  # - PORT_INCREMENT_VALUE           : When installing to a new group,
38  #                                    increase the port number by this value.
39  # - PORT_INCREMENT_VALUE_IN_GROUP  : When installing to an existing group,
40  #                                    increase the port number by this value.
41  # - CLUSTER_ADMIN_REFRESH_INTERVAL : Information update interval
42  #                                    for Cluster Admin (second)
43  ################################################################################
44  DEFAULT_COORDINATOR_SERVICE_PORT = 5100
45  DEFAULT_COORDINATOR_LINK_PORT    = 5101
```

```
DEFAULT_COORDINATOR_ADMIN_PORT   = 5102
DEFAULT_COORDINATOR_HOME_PATH    = /home/machbase/coordinator

DEFAULT_DEPLOYER_SERVICE_PORT    = 5200
DEFAULT_DEPLOYER_LINK_PORT       = 5201
DEFAULT_DEPLOYER_ADMIN_PORT      = 5202
DEFAULT_DEPLOYER_HOME_PATH       = /home/machbase/deployer

DEFAULT_BROKER_SERVICE_PORT      = 5656
DEFAULT_BROKER_LINK_PORT         = 5301
DEFAULT_BROKER_HOME_PATH         = /home/machbase/broker

DEFAULT_WAREHOUSE_SERVICE_PORT   = 5400
DEFAULT_WAREHOUSE_LINK_PORT      = 5401
DEFAULT_WAREHOUSE_REPL_PORT      = 5402

# Use if there is no same group in the server
DEFAULT_WAREHOUSE_HOME_PATH      = /home/machbase/warehouse_$
PORT_INCREMENT_VALUE             = 10
PORT_INCREMENT_VALUE_IN_GROUP    = 0

# Use if there is a same group in the server
#DEFAULT_WAREHOUSE_HOME_PATH      = /home/machbase/warehouse_$_w#
#PORT_INCREMENT_VALUE             = 100
#PORT_INCREMENT_VALUE_IN_GROUP    = 10

DEFAULT_SSH_USER_ID              = machbase
CLUSTER_ADMIN_REFRESH_INTERVAL   = 5
```

- Installation folders for BROKER and WAREHOUSE can use $ and # as default values.
- $ Is replaced by the Group name, and # is replaced by the number of Warehouses in the Group.
- PORT_INCREMENT_VALUE is the increment value of the port number when a Group is added.
  If PORT_INCREMENT_VALUE is 10, each port number is increased by 10 each time a Group is added.
- PORT_INCREMENT_VALUE_IN_GROUP is increment value of the port number when a Warehouse in the group is added.
  If PORT_INCREMENT_VALUE_IN_GROUP is 10, each port number is incremented by 10 whenever a warehouse is added in the same group.
  In practice, the Warehouse of the same Group are rarely installed on one server, so it is mainly used for testing purposes.
- DEFAULT_SSH_USER_ID is the default value of the ID used for SSH connection.
  Since Coordinator and Deployer installation requires SSH connection, the default SSH ID and Password are used during installation.
- CLUSTER_ADMIN_REFRESH_INTERVAL is the number of seconds to check the status of the cluster in Cluster Admin.
  MWA connects to the coordinator on this value basis to retrieve the status of the cluster.

⚠ If it starts with #, it is recognized as a comment.

Run WMA Service

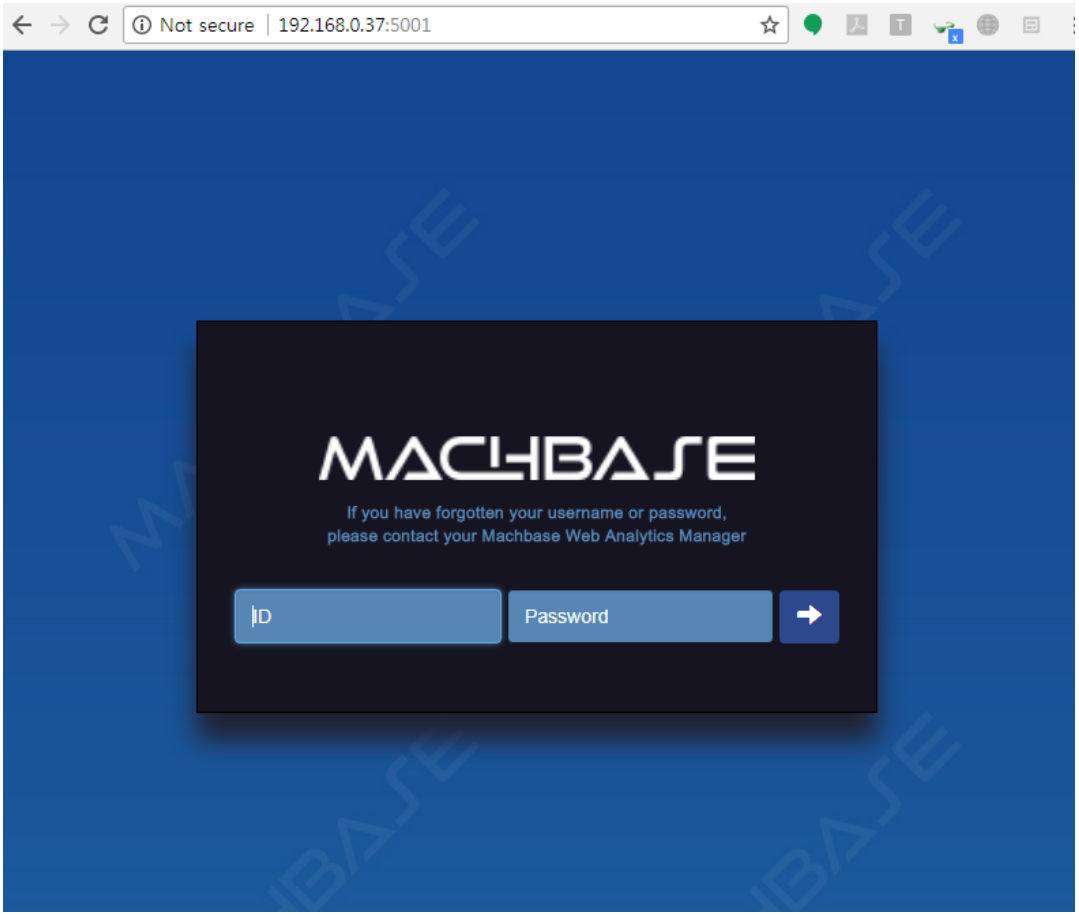
If the following command is used to run the MWA service, the service connection address will be output.

```
# Move to directory
cd $MACHBASE_HOME/bin

# Run MWA service
MWAserver start
```

MWA Service Connection

Open the browser and connect to the output address and port.

The initial web login account information is admin / machbase.


Not secure
192.168.0.37:5001
MACHBASE
If you have forgotten your username or password,
please contact your Machbase Web Analytics Manager
ID
Password

(2) Coordinator / Deployer Installation

Cluster Edition Installation (MWA)

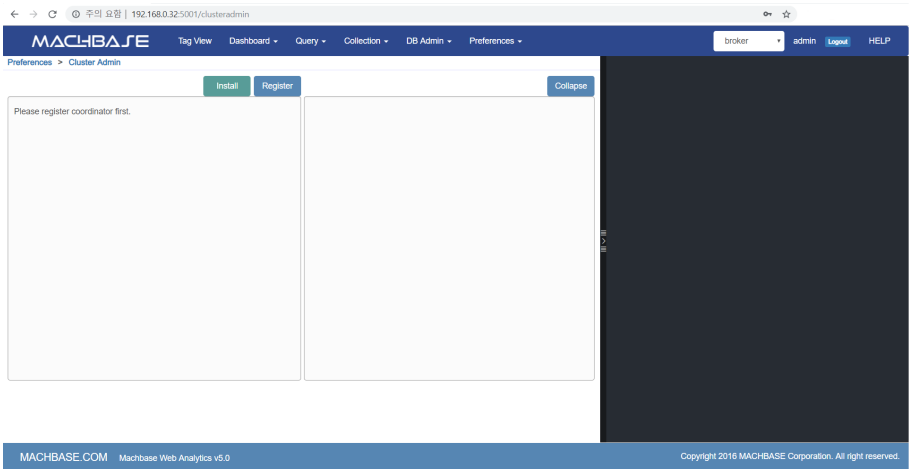
- (1) MWA Installation
- (2) Coordinator / Deployer Installation
- (3) Broker / Warehouse Installation

Index

Move to Cluster Admin Page

When accessing Web Admin, the Cluster Admin screen appears. It also appears when you click on the menu Preferences at the top.

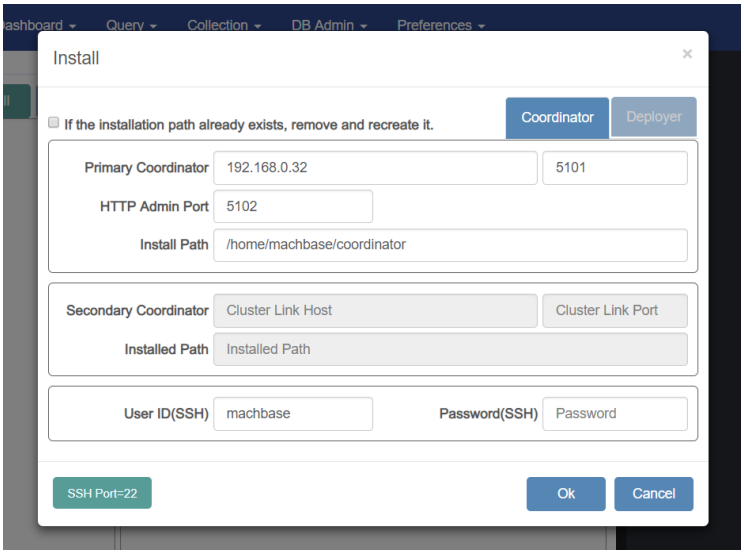
- If you are installing Coordinator / Deployer for the first time , click the **[Install]** button at the top to proceed.
- To register Coordinator / Deployer already installed with Command Line, click the **[Register]** button at the top .



Coordinator Installation

On the Cluster Admin page, click **Install**.

Enter in each item and click **[Ok]**.



Item	Description	Example
Primary Coordinator / Secondary Coordinator	The name of the node to be added is configured in the format "IP: PORT". The PORT value is set to the CLUSTER_LINK_PORT_NO value.	192.168.0.32:5101

Item	Description	Example
HTTP Admin Port	The PORT value is set to the HTTP_ADMIN_PORT value.	5102
Install Path	Specifies the path to install.	/home/machbase/coordinator
User ID(SSH) / Password(SSH)	If the server you want to install is different from the server running MWA, enter SSH's User ID and Password.	

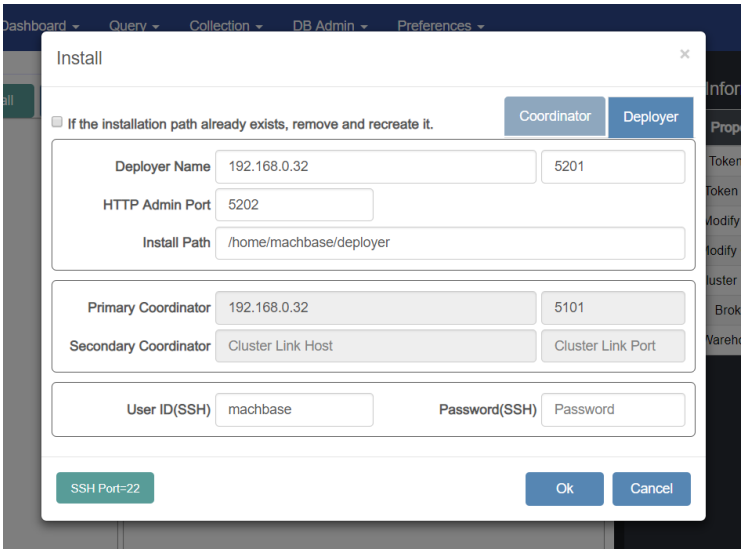
Delete Coordinator

⚠ In MWA, there is no ability to delete coordinator. You must delete it directly with command-line.

Deployer Installation

On the Cluster Admin page, click **Install**.

Enter in each item and click **[Ok]**.



Item	Description	Example
Deployer Name	The name of the node to be added is configured in the format "IP: PORT". The PORT value is set to the CLUSTER_LINK_PORT_NO value.	192.168.0.32:5201
HTTP Admin Port	The PORT value is set to the HTTP_ADMIN_PORT value.	5202
Install Path	Specifies the path to install.	/home/machbase/deployer
User ID(SSH) / Password(SSH)	If the server you want to install is different from the server running MWA, enter SSH's User ID and Password.	

Delete Deployer

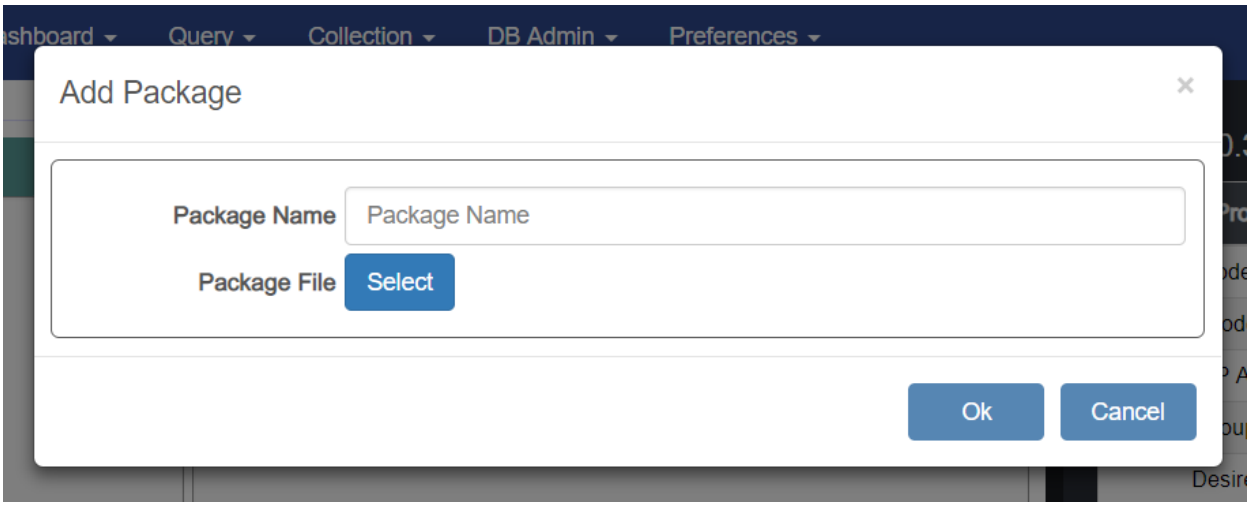
On the Cluster Admin page, click the Deployer icon(D), and then click **[Remove Deployer]**.

Then, exit the deployer with command-line and delete the directory.

Add Package

Adding the package to be installed as Broker and Warehouse in Coordinator. At this time, the registered package registers the lightweight version excluding MWA.

On the Cluster Admin page, click the Coordinator icon(C), and then click **[Add Package]**.



Item	Description	Example
Package Name	Specifies the package name to be added.	machbase
Package File	Selects the package file. Specifies a lightweight package that excludes MWA files, since this package is for broker and warehouse installations only.	/home/machbase/machbase-ent-x.x.x.official-LINUX-X86-64-release-lightweight.tgz

Delete Package

Removing the package registered in Coordinator.

On the Cluster Admin page, click the Coordinator icon( ), and then click **[Remove Package]**.

image2021-11-5_18-27-27.png

Item	Description	Example
Package Name	Specifies the name of the package to be deleted.	machbase

(3) Broker / Warehouse Installation

Cluster Edition Installation (MWA)

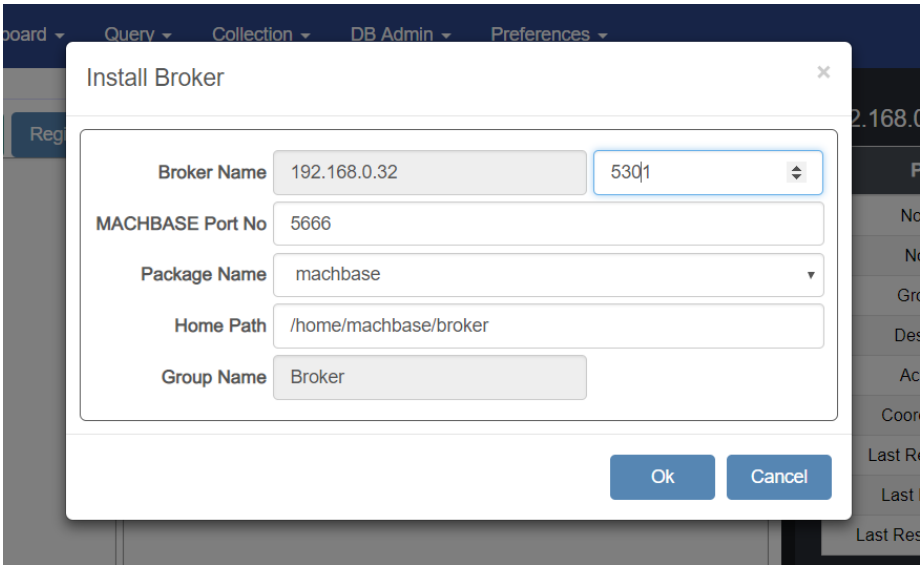
- (1) MWA Installation
- (2) Coordinator / Deployer Installation
- (3) Broker / Warehouse Installation

Index

Broker Installation

On the Cluster Admin page, click the Deployer icon ( ), and then click **[Install Broker]**.

Enter in each item and click **[Ok]**.

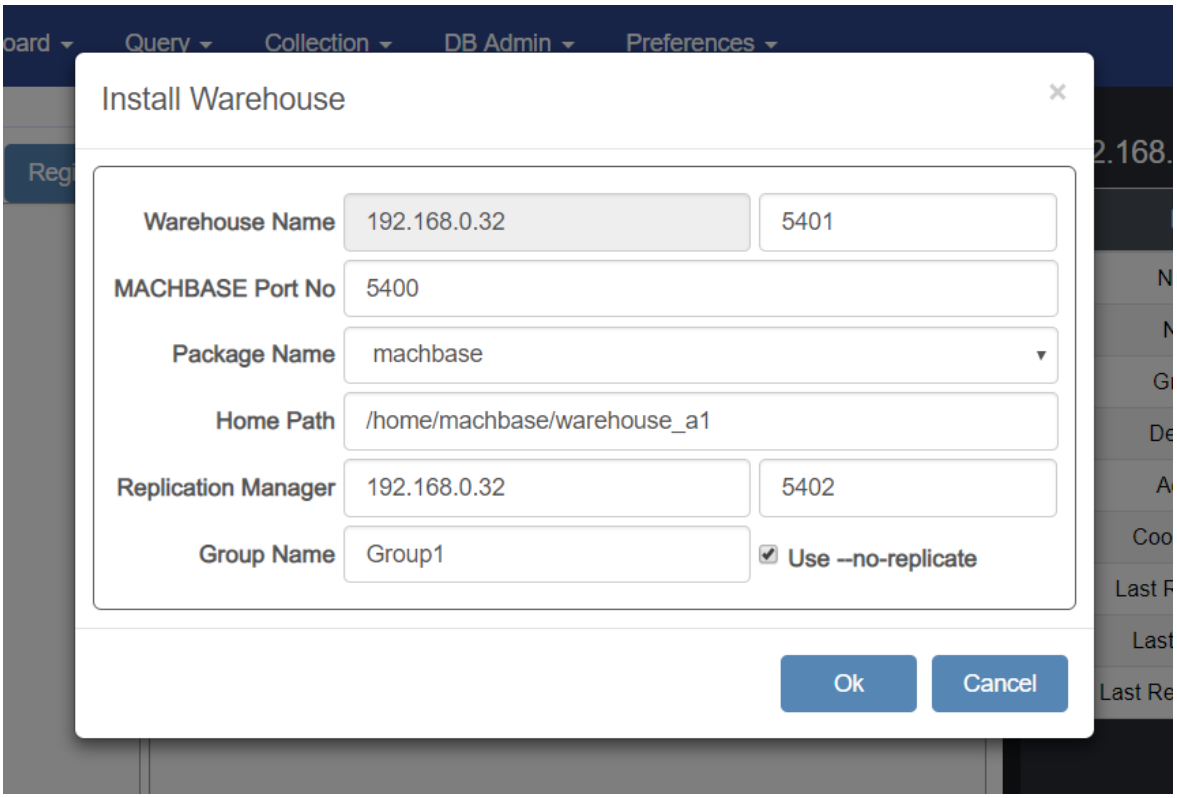


Item	Description	Example
Broker Name	The name of the node to be added is configured in the format "IP: PORT". The IP value is automatically set to the Deployer's IP value. The PORT value is set to the CLUSTER_LINK_PORT_NO value.	192.168.0.32:5301
MACHBASE Port No	Specifies the machbased working port.	5666
Package Name	Sets the package name specified when adding the package.	machbase
Home Path	Specifies the path to install.	/home/machbase/broker

Warehouse Installation

On the Cluster Admin page, click the Deployer icon, and then click **[Install Warehouse]**.

Enter in each item and click **[Ok]**.



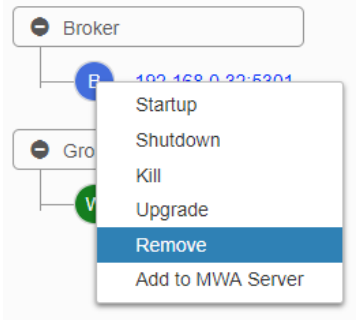
Item	Description	Example
Warehouse Name	The name of the node to be added is configured in the format "IP: PORT". The IP value is automatically set to the Deployer's IP value. The PORT value is set to the CLUSTER_LINK_PORT_NO value.	192.168.0.32:5401
MACHBASE Port No	Specifies the machbased working port.	5400
Package Name	Sets the package name specified when adding the package.	machbase
Home Path	Specifies the path to install.	/home/machbase/warehouse_a1
Replication Manager	Specifies the node that will be responsible for replication in "IP: PORT" format.	192.168.0.32:5402
Group Name	Specifies the Group name.	group1

If you want the Warehouse that you are installing to replicate, **uncheck the [Use - no-replicate] check box** .

When adding a Warehouse to extend the cluster with data already entered, you can uncheck the check box.
Adding a Warehouse to an empty cluster with the checkbox unchecked is allowed, but the replication function is meaningless and can lead to performance degradation during installation.

Delete Broker / Warehouse

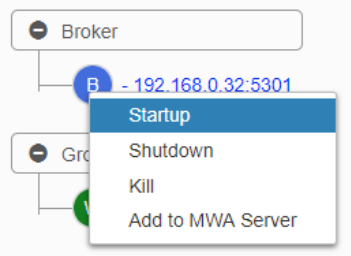
On the Cluster Admin page, click the Broker / Warehouse icon (/), and then click **[Remove]**.



If Cluster Status is Service status, you must change it to Deactivate status. Click the icon to change the Cluster Status to Service / Deactivate status.

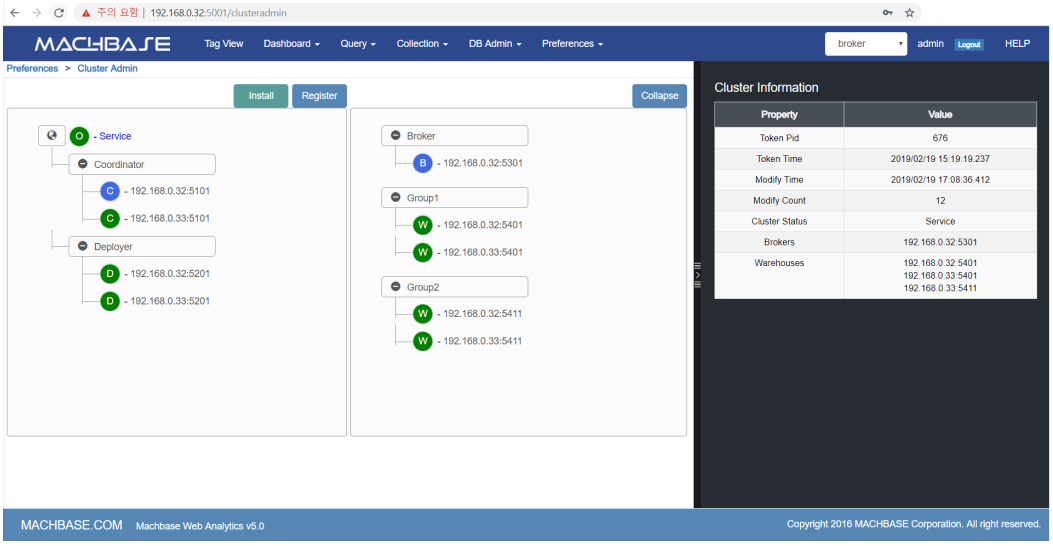
Start Up / Shut Down / Stop Broker / Warehouse

On the Cluster Admin page, click the Broker / Warehouse icon(B / W), then click **[Startup / Shutdown / Kill]**.



Complete Cluster Configuration

If you installed four warehouses consisting of two Coordinators, two Deployers, one Broker, and two groups, your screen should match the image below.



Upgrade

To Prepare: Check Compatible Version

To upgrade, you need to check which version of Machbase is currently installed and which version of Machbase you want to install.

- If there is a difference between the major or minor versions, you must request the migration tool through Machbase technical support.
- If major and minor versions are the same and only the patch version is different, upgrades are possible.

How to Upgrade

- Linux Upgrade
- Windows 업그레이드 Windows Upgrade
- Cluster Edition Upgrade

Linux Upgrade

페이지 잠금 상태입니다. 외부에서 공개되지 않고 로그인 사용자만 접근 가능합니다.
이유 : 업그레이드 실험을 해 보고 적어야 할 것 같은데 당장 할 수 없어서 잠급니다 (interp)

Tarball

RPM/Debian Image

Docker

Docker 이미지 태그가 동일한 경우에만 사용이 가능하며, 다른 태그의 MACHBASE 이미지를 업그레이드 하게 되면 메타 업그레이드 이슈가 발생할 수 있다.

Windows 업그레이드 Windows Upgrade

페이지 잠금 상태입니다. 외부에서 공개되지 않고 로그인 사용자만 접근 가능합니다.
이유 : 업그레이드 실험을 해 보고 적어야 할 것 같은데 당장 할 수 없어서 잠급니다 (interp)

추가바람 ⚠️

Cluster Edition Upgrade

Index

Coordinator Upgrade

Coordinator / Deployer must be upgraded manually.

Precautions

- DDL or DELETE must not be in use. (INSERT, APPEND, SELECT do not matter.)
- You can not issue commands such as adding / starting / terminating / deleting nodes during the upgrade.

Coordinator Shutdown

Coordinator / Deployer does not affect INSERT, APPEND, SELECT in Broker / Warehouse even if it is shut down.

However, it does not detect that the Broker / Warehouse also shuts down while it is shutting down. (Normally detected after restart)

```
machcoordinatoradmin --shutdown
```

Coordinator Backup (Optional)

Back up the `dbs/` and `conf/` directories located in `$MACH_COORDINATOR_HOME`.

Coordinator Upgrade

Proceed with full package instead of lightweight package.

```
tar zxvf machbase-ent-new.official-LINUX-X86-64-release.tgz -C $MACHBASE_COORDINATOR_HOME
```

Unzip and overwrite the package to `$MACH_COORDINATOR_HOME`.

Coordinator Startup

```
machcoordinatoradmin --startup
```

Deployer Upgrade

This has the same process as the Coordinator.

Precautions

- You can not issue commands such as adding / starting / terminating / deleting nodes during the upgrade.

Deployer Shutdown

```
machdeployeradmin --shutdown
```

Deployer Backup (Optional)

Back up the `dbs/` and `conf/` directories located in `$MACH_DEPLOYER_HOME`.

Deployer Upgrade

If you are running MWA or not running Collector on the Host the Deployer is installed, you can proceed with the lightweight package.

```
tar zxvf machbase-ent-new.official-LINUX-X86-64-release.tgz -C $MACH_DEPLOYER_HOME
```

Unzip and overwrite the package to `$MACH_DEPLOYER_HOME`.

Deployer Startup

```
machdeployeradmin --startup
```

Package Registration

To upgrade Broker / Warehouse, register the Package in Coordinator and proceed with the upgrade.

> It is recommended to register with a lightweight package.

First, move the package to the Host where `$MACH_COORDINATOR_HOME` is located.

Next, add the package using the following command.

```
machcoordinatoradmin --add-package=new_package --file-name=./machbase-ent-new.official-LINUX-X86-64-release-lightwe
```

Option	Description
--add-package	Specifies the name of the package to add.
--file-name	Specifies the path to the package file to add. ⚠ If a package with the same filename is added, you will receive an error, so check the file name.

Broker/Warehouse Upgrade

In the Coordinator, run the following command.

Node Shutdown

```
machcoordinatoradmin --shutdown-node=localhost:5656
```

Node Upgrade

```
machcoordinatoradmin --upgrade-node=localhost:5656 --package-name=new_package
```

Option	Description
--upgrade-node	Enters the name of the upgrade target Node.
--package-name	Enters the name of the Package to be upgraded.

> If you upgrade the Node without shutting down the Node, it will automatically shut down the Node and perform the Node upgrade.

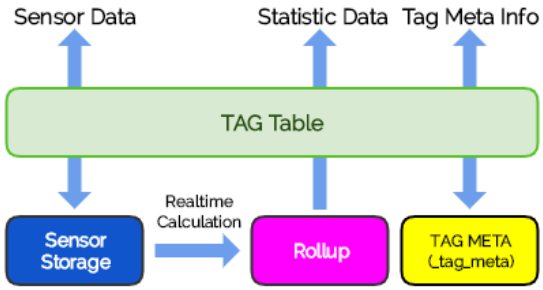
However, for stability, you should explicitly shut down the Node before upgrading.

Node Startup

```
machcoordinatoradmin --startup-node=localhost:5656
```

Tag Table

Concept



The TAG table is a virtual table that can be created by only one user, and is responsible for data storage and related additional information management for sensor data processing.

The TAG table provides three conceptual data processing spaces as described below.

Sensor Partition

This is an internal sensor data table based on the schema defined by the user when TAG table is created.

This data can be extracted through a SELECT query on the TAG table.

This table has very strong data management capabilities as listed below.

1. Hundreds of thousands of sensor data can be loaded at high speed.
2. Tens of thousands of sensor data can be retrieved at high speed given the time range condition.
3. Real-time compression allows long-term storage of sensor data.
4. Chronological deletion of sensor data beginning with the oldest is possible.

The user sensor data to be stored is, in basic, a time series data and is a specific data type with the corresponding tag of the name, time, and 64-bit real value.

TagName(User defined length string)	Time(64bit)	Real value(64bit)	(User defined Extended Columns...)

ROLLUP Partition

This is an internal table that automatically generates statistical data based on the sensor data stored in the sensor storage.

This was developed in order to obtain statistical data over a long period of time within a few seconds in real time analysis.

ROLLUP table supports three types of tables: hour, minute, and second; and five types of statistics functions: MIN, MAX, AVG, SUM, COUNT.

To get this ROLLUP result value, you must execute a SELECT query with HINT in the TAG table.

META Table

This is another table that stores the name and additional meta information for TAG data.

Users can generate this meta information with the INSERT clause and manipulate it in several ways through SELECT, UPDATE, and DELETE.

How to use Tag table

Creation and Dropping of Tag table

The user must explicitly create TAG table.

Note that there is no TAG table when the database is first installed.

Since the TAG table is, in basic, intended to store sensor data, the following three essential items must be included.

- Tag name
- Input time
- Sensor value

However, the Machbase TAG table is accompanied by keywords for the above required columns, as it allows input of the above three and additional columns.

- Tag name : PRIMARY KEY
- Input time : BASETIME
- Sensor value : SUMMARIZED

This tag name is used as tag meta information described in the next section.

Index

Creation of Tag table

The simplest tag table is generated as follows.

```
Mach> create tagdata table TAG (name varchar(20) primary key, time datetime, value double);
[ERR-02253: Mandatory column definition (PRIMARY KEY / BASETIME / SUMMARIZED) is missing.]
==> If you omit some keywords on creating tag table, error occurs.

Mach> create tagdata table TAG (name varchar(20) primary key, time datetime basetime, value double summarized);
Executed successfully.

Mach> desc tag;
[ COLUMN ]
--------------------------------------------------------------------
NAME      TYPE        LENGTH
--------------------------------------------------------------------
NAME      varchar         20
TIME      datetime        31
VALUE     double          17
```

A table named TAG was created.

Additional sensor columns

In reality, it is sometimes difficult to solve a given problem with just three columns when using the TAG table..

In particular, since the information of the sensor data to be input may be a specific group or an Internet address as well as a name, a time, and a value, the following can be added.

```
Mach> create tagdata table TAG (name varchar(20) primary key, time datetime basetime, value double summarized, grp
Executed successfully.

Mach> desc tag;
[ COLUMN ]
--------------------------------------------------------------------
NAME             TYPE        LENGTH
--------------------------------------------------------------------
NAME             varchar         20
TIME             datetime        31
VALUE            double          17
GRPID            short            6       <=== added column
MYIP             ipv4            15       <=== added column
```

Note, however, that in **older versions, including 5.5, values of type VARCHAR can not fit into the supplementary column.**

```
Mach> create tagdata table TAG (name varchar(20) primary key, time datetime basetime, value double summarized, myna
[ERR-01851: Variable length columns are not allowed in tag table.]
```

In the case of string type, the above error occurs. In versions 5.6 and later, VARCHAR is also supported for additional columns in the TAG table.

Additional metadata columns

It is not only possible to add sensor columns to the TAG table, but also to input information dependent on each tag name.

Since this information does not need to be redundantly stored in the sensor data, it is necessary to add a separate column definition syntax METADATA (...) for efficient management.

```
Mach> create tagdata table TAG (name varchar(20) primary key, time datetime basetime, value double summarized)
   2  metadata (room_no integer, tag_description varchar(100));
```

Here, room_no and tag_description are information dependent on name. For example, you can input this information.

name	room_no	tag_description
temp_001	1	Reads current temperature as Celsius
humid_001	1	Reads current humidity as percentage

After input, you can query with TAG table through SELECT.

```
Mach> SELECT name, time, value, tag_description FROM tag LIMIT 1;
name                 time                           value
--------------------------------------------------------------------------------
tag_description
--------------------------------------------------------------------------------
temp_001             2019-03-01 09:52:17 000:000:000 25.3
It reads current temperature as Celsius
```

Specify the numberof partitions

Although the number of partitions is set to four by default, you can specify this number to control memory and CPU usage as follows.

Create the table by specifying its value in the table property **tag_partition_count**.

If you increase this value, the higher the parallelism and the better the performance, but the memory usage increases and the CPU usage also increases.

The following example shows setting tag_partition_count to 1 for the purpose of minimizing memory and CPU usage.

```
Mach> create tagdata table TAG (name varchar(20) primary key, time datetime basetime, value double summarized) tag_
Executed successfully.
```

Dropping tag table

If you need to recreate the generated tag table, or if you need to free up disk space, you can use the following DROP command to drop it.

Note that all data related to the TAG table, ie tag data, metadata, and ROLLUP tables are deleted.

```
Mach> drop table tag;
Dropped successfully.

Mach> desc tag;
tag does not exist.
```

Managing tag meta (tag name)

Concept of tag meta

The tag meta represents the name and additional information of an arbitrary tag stored in the Machbase.

That is, if there are three tags existing in a specific apparatus, an arbitrary name representing the tag and related additional information are required, which are all referred to as meta information of the tag.

This tag meta should specify at least a name and, if necessary, specify various data types for the device.

Tag meta with tag name only

Creation of tag meta

Below is the command to create the TAG table where the most basic tag meta is created.

```
create tagdata table TAG (name varchar(20) primary key, time datetime basetime,
Mach> desc tag;
[ COLUMN ]
---------------------------------------------------------------
NAME                          TYPE               LENGTH
---------------------------------------------------------------
NAME                          varchar            20
TIME                          datetime           31
VALUE                         double             17
```

The above is the basic TAG table creation, and there is no separate information about the tag meta.

In this case, the tag meta has only basic information of VARCHAR (20).

Input of tag meta

Now, let's insert one piece of tag information named TAG1.

```
Mach> insert into tag metadata values ('TAG_0001');
1 row(s) inserted.
```

Through the above query, we created one tag named TAG_0001.

Output of tag meta

Machbase provides a special table **_tag_meta** for identifying the information of the input tag meta.

Therefore, the user can confirm the information of all the tags input in the Machbase through the following query.

```
Mach> select * from _tag_meta;
ID                  NAME
-----------------------------------------------
1                   TAG_0001
[1] row(s) selected.
```

The ID is automatically assigned as an internally managed value.

Modifying tag meta

Machbase allows you to modify the input tag meta information. The name can be modified as follows.

```
Mach> update tag metadata set name = 'NEW_0001' where NAME = 'TAG_0001';
1 row(s) updated.

Mach> select * from _tag_meta;
ID                  NAME
-----------------------------------------------
1                   NEW_0001
```

Index

```
[1] row(s) selected.
```

You can see that the name has been changed from TAG_0001 to NEW_0001 as above.

Deleting tag meta

You can delete the actual tag meta information as shown below.

```
Mach> delete from tag metadata where name = 'NEW_0001';
1 row(s) deleted.

Mach> select * from _tag_meta;
ID                  NAME
-----------------------------------------------
[0] row(s) selected.
```

However, it should be noted that even if the tag meta is deleted, the actual data of the tag inputted in the past will not be deleted.

Tag meta with additional information

Creation of tag meta

Below, we will add 16 bit integer, time, and IPv4 information to the tag meta information.

Note that once you have created a tag meta, you can modify the value except the structure.

```
create tagdata table TAG (name varchar(20) primary key, time datetime basetime, value double summarized)
metadata (type short, create_date datetime, srcip ipv4) ;

Mach> desc tag;
[ COLUMN ]
-----------------------------------------------------------------
NAME                        TYPE                LENGTH
-----------------------------------------------------------------
NAME                        varchar             20
TIME                        datetime            31
VALUE                       double              17
[ META-COLUMN ]
-----------------------------------------------------------------
NAME                        TYPE                LENGTH
-----------------------------------------------------------------
TYPE                        short               6
CREATE_DATE                 datetime            31
SRCIP                       ipv4                15
```

Input of tag meta

You can check the information by typing the following with the additional information other than the name.

```
Mach> insert into tag metadata(name) values ('TAG_0001');
1 row(s) inserted.

Mach> select * from _tag_meta;
ID                  NAME                TYPE        CREATE_DATE                 SRCIP
-------------------------------------------------------------------------------------------------------------
1                   TAG_0001            NULL        NULL                        NULL
[1] row(s) selected.
```

As you can see, NULL is inserted in columns other than NAME as above.

Now let's add more information as shown below.

```
Mach> insert into tag metadata values ('TAG_0002', 99, '2010-01-01', '1.1.1.1');
1 row(s) inserted.
```

```
Mach> select * from _tag_meta;
ID                   NAME                 TYPE        CREATE_DATE                    SRCIP
-------------------------------------------------------------------------------------------------------------
1                    TAG_0001             NULL        NULL                           NULL
2                    TAG_0002             99          2010-01-01 00:00:00 000:000:000 1.1.1.1
[2] row(s) selected.
```

Modifying tag meta

Now let's change the type of TAG_0001 from NULL to 11.

```
Mach> update tag metadata set type = 11 where name = 'TAG_0001';
1 row(s) updated.

Mach> select * from _tag_meta;
ID                   NAME                 TYPE        CREATE_DATE                    SRCIP
-------------------------------------------------------------------------------------------------------------
2                    TAG_0002             99          2010-01-01 00:00:00 000:000:000 1.1.1.1
1                    TAG_0001             11          NULL                           NULL
[2] row(s) selected.
```

You can modify the values of all fields through the UPDATE statement.

However, it is a common constraint that NAME must be specified in the WHERE clause.

Tag meta lookup via RESTful API

Get all tag lists

Below is an example of getting a list of all the tags in Machbase.

```
Host:~$ curl  -G  "http://192.168.0.148:5001/machiot-rest-api/tags/list"
{"ErrorCode": 0,
 "ErrorMessage": "",
 "Data": [{"NAME": "TAG_0001"},
          {"NAME": "TAG_0002"}]}
Host:~$
```

Get the time range of a specific tag

Below is an example of obtaining the minimum and maximum time range of the data that the desired tag has.

This feature is very useful when charting specific tags.

Syntax

```
{MWA URL}/machiot-rest-api/tags/range/  # Time Range of whole DB
{MWA URL}/machiot-rest-api/tags/range/{TagName}  # Time Range of a specific Tag
```

Getting time range of all tags

```
Host:~$ curl  -G  "http://192.168.0.148:5001/machiot-rest-api/tags/range/"
{"ErrorCode": 0,
 "ErrorMessage": "",
 "Data": [{"MAX": "2018-02-10 10:00:00 000:000:000", "MIN": "2018-01-01 01:00:00 000:000:000"}]}
```

Getting Time range of a specific tag

```
Host:~$ curl  -G  "http://192.168.0.148:5001/machiot-rest-api/tags/range/TAG_0001"
{"ErrorCode": 0, "ErrorMessage": "", "Data": [{"MAX": "2018-01-10 10:00:00 000:000:000", "MIN": "2018-01-01 01:00:(
```

```
Host:~$
Host:~$ curl  -G  "http://192.168.0.148:5001/machiot-rest-api/tags/range/TAG_0002"
{"ErrorCode": 0, "ErrorMessage": "", "Data": [{"MAX": "2018-02-10 10:00:00 000:000:000", "MIN": "2018-02-01 01:00:0
```

Manipulating tag data

Input tag data

Index

Input using the INSERT statement

The easiest way to insert data is through the INSERT statement.

This is used for the small data sets, but if you want to load large amounts of data quickly, use another method.

```
Mach> create tagdata table TAG (name varchar(20) primary key, time datetime bas
Executed successfully.

Mach>  insert into tag metadata values ('TAG_0001');
1 row(s) inserted.

Mach> insert into tag values('TAG_0001', now, 0);
1 row(s) inserted.

Mach> insert into tag values('TAG_0001', now, 1);
1 row(s) inserted.

Mach> insert into tag values('TAG_0001', now, 2);
1 row(s) inserted.

Mach> select * from tag where name = 'TAG_0001';
NAME                 TIME                           VALUE
------------------------------------------------------------------------------
TAG_0001             2018-12-19 17:41:37 806:901:728 0
TAG_0001             2018-12-19 17:41:42 327:839:368 1
TAG_0001             2018-12-19 17:41:43 812:782:202 2
[3] row(s) selected.
```

We put the three TAG values as the current time.

Load all at once through a CSV file

Machbase allows you to load large amounts of CSV files through a csvimport tool.

More details can be found through practical examples, and are described in brief below.

CSV Format (data.csv)

```
TAG_0001, 2009-01-28 07:03:34 0:000:000, -41.98 TAG_0001, 2009-01-28 07:03:34 1:000:000, -46.50 TAG_0001, 2009-01-
```

Prepare a csv file consisting of <tag name, time, value> as above.

Of course, the tag name TAG_0001 must be present in the tag meta.

Using the loading program csvimport

```
csvimport -t TAG -d data.csv -F "time YYYY-MM-DD HH24:MI:SS mmm:uuu:nnn" -l error.log
```

This loads data.csv into a table called TAG.

The -F option specifies the time format stored in the data.csv file.

In addition, -l error.log records the error that occurred when inputting as a separate file.

Input via the RESTful API

For more detailed usage of the RESTful API, please refer to the following examples.

Syntax of Input API

Machbase provides the RESTful API as follows:

```
{ "values":[ [TAG_NAME, TAG_TIME, VALUE], # Specify Tagname,Time,Value. if you define additional columns, the colur
```

It is requested the number of columns in the defined TAG schema to match the above structure.

Input data via the SDK

Machbase provides standard development tools for a variety of languages, including:

- C/C++ library
- JAVA library
- Python library
- C# library

Through these libraries, users can create various application programs according to their environment and input data to Machbase.

Extract tag data

Machbase provides high-speed tag data extraction, especially for the time range of a specific tag.

Index

Sample Schema

In the following example, we created a TAG table and created two tags as shown below.

For each tag, data from January 1, 2018 to February 10, 2018 were inserted.

```
create tagdata table TAG (name varchar(20) primary key, time datetime basetime,

insert into tag metadata values ('TAG_0001');
insert into tag metadata values ('TAG_0002');

insert into tag values('TAG_0001', '2018-01-01 01:00:00 000:000:000', 1);
insert into tag values('TAG_0001', '2018-01-02 02:00:00 000:000:000', 2);
insert into tag values('TAG_0001', '2018-01-03 03:00:00 000:000:000', 3);
insert into tag values('TAG_0001', '2018-01-04 04:00:00 000:000:000', 4);
insert into tag values('TAG_0001', '2018-01-05 05:00:00 000:000:000', 5);
insert into tag values('TAG_0001', '2018-01-06 06:00:00 000:000:000', 6);
insert into tag values('TAG_0001', '2018-01-07 07:00:00 000:000:000', 7);
insert into tag values('TAG_0001', '2018-01-08 08:00:00 000:000:000', 8);
insert into tag values('TAG_0001', '2018-01-09 09:00:00 000:000:000', 9);
insert into tag values('TAG_0001', '2018-01-10 10:00:00 000:000:000', 10);

insert into tag values('TAG_0002', '2018-02-01 01:00:00 000:000:000', 11);
insert into tag values('TAG_0002', '2018-02-02 02:00:00 000:000:000', 12);
insert into tag values('TAG_0002', '2018-02-03 03:00:00 000:000:000', 13);
insert into tag values('TAG_0002', '2018-02-04 04:00:00 000:000:000', 14);
insert into tag values('TAG_0002', '2018-02-05 05:00:00 000:000:000', 15);
insert into tag values('TAG_0002', '2018-02-06 06:00:00 000:000:000', 16);
insert into tag values('TAG_0002', '2018-02-07 07:00:00 000:000:000', 17);
insert into tag values('TAG_0002', '2018-02-08 08:00:00 000:000:000', 18);
insert into tag values('TAG_0002', '2018-02-09 09:00:00 000:000:000', 19);
insert into tag values('TAG_0002', '2018-02-10 10:00:00 000:000:000', 20);
```

Extract all TAG data

```
Mach> select * from tag;
NAME TIME VALUE
--------------------------------------------------------------------------------
TAG_0001 2018-01-01 01:00:00 000:000:000 1
TAG_0001 2018-01-02 02:00:00 000:000:000 2
TAG_0001 2018-01-03 03:00:00 000:000:000 3
TAG_0001 2018-01-04 04:00:00 000:000:000 4
TAG_0001 2018-01-05 05:00:00 000:000:000 5
TAG_0001 2018-01-06 06:00:00 000:000:000 6
TAG_0001 2018-01-07 07:00:00 000:000:000 7
TAG_0001 2018-01-08 08:00:00 000:000:000 8
TAG_0001 2018-01-09 09:00:00 000:000:000 9
TAG_0001 2018-01-10 10:00:00 000:000:000 10
TAG_0002 2018-02-01 01:00:00 000:000:000 11
TAG_0002 2018-02-02 02:00:00 000:000:000 12
TAG_0002 2018-02-03 03:00:00 000:000:000 13
TAG_0002 2018-02-04 04:00:00 000:000:000 14
TAG_0002 2018-02-05 05:00:00 000:000:000 15
TAG_0002 2018-02-06 06:00:00 000:000:000 16
TAG_0002 2018-02-07 07:00:00 000:000:000 17
TAG_0002 2018-02-08 08:00:00 000:000:000 18
TAG_0002 2018-02-09 09:00:00 000:000:000 19
TAG_0002 2018-02-10 10:00:00 000:000:000 20
[20] row(s) selected.
```

If there is no special condition as described above, data can be extracted for each tag arranged in each time order.

Extract data for a specific tag name

Below is an example of data with TAG name TAG_0002.

```
Mach> select * from tag where name='TAG_0002';
NAME                 TIME                            VALUE
-------------------------------------------------------------------------------------
TAG_0002             2018-02-01 01:00:00 000:000:000 11
TAG_0002             2018-02-02 02:00:00 000:000:000 12
TAG_0002             2018-02-03 03:00:00 000:000:000 13
TAG_0002             2018-02-04 04:00:00 000:000:000 14
TAG_0002             2018-02-05 05:00:00 000:000:000 15
TAG_0002             2018-02-06 06:00:00 000:000:000 16
TAG_0002             2018-02-07 07:00:00 000:000:000 17
TAG_0002             2018-02-08 08:00:00 000:000:000 18
TAG_0002             2018-02-09 09:00:00 000:000:000 19
TAG_0002             2018-02-10 10:00:00 000:000:000 20
[10] row(s) selected.
```

Query for time range

The following is a query of a time range for TAG_0002 and receives data.

> It is common practice to use the between clause to give a time range. Of course, you can get the same result by typing the time range with time < or >.

```
Mach> select * from tag where name = 'TAG_0002' and time between to_date('2018-02-01') and to_date('2018-02-05');
NAME                 TIME                            VALUE
-------------------------------------------------------------------------------------
TAG_0002             2018-02-01 01:00:00 000:000:000 11
TAG_0002             2018-02-02 02:00:00 000:000:000 12
TAG_0002             2018-02-03 03:00:00 000:000:000 13
TAG_0002             2018-02-04 04:00:00 000:000:000 14
[4] row(s) selected.

Mach> select * from tag where name = 'TAG_0002' and time > to_date('2018-02-01') and time < to_date('2018-02-05');
NAME                 TIME                            VALUE
-------------------------------------------------------------------------------------
TAG_0002             2018-02-01 01:00:00 000:000:000 11
TAG_0002             2018-02-02 02:00:00 000:000:000 12
TAG_0002             2018-02-03 03:00:00 000:000:000 13
TAG_0002             2018-02-04 04:00:00 000:000:000 14
[4] row(s) selected.
```

Time range search for multiple tags

Below is an example of retrieving the same time range data for two or more tags.

If you want to get fast results for a large number of tags at the same time, it is preferable to perform the following type of query.

```
Mach> select * from tag where name in ('TAG_0002', 'TAG_0001') and time between to_date('2018-01-05') and to_date(
NAME                 TIME                            VALUE
-------------------------------------------------------------------------------------
TAG_0001             2018-01-05 05:00:00 000:000:000 5
TAG_0001             2018-01-06 06:00:00 000:000:000 6
TAG_0001             2018-01-07 07:00:00 000:000:000 7
TAG_0001             2018-01-08 08:00:00 000:000:000 8
TAG_0001             2018-01-09 09:00:00 000:000:000 9
TAG_0001             2018-01-10 10:00:00 000:000:000 10
TAG_0002             2018-02-01 01:00:00 000:000:000 11
TAG_0002             2018-02-02 02:00:00 000:000:000 12
```

```
TAG_0002              2018-02-03 03:00:00 000:000:000 13
TAG_0002              2018-02-04 04:00:00 000:000:000 14
[10] row(s) selected.
```

Search data over a certain value

The conditions for the tag value can also be given as follows.

Filtering was performed for those values greater than 12 and less than 15 among the values of TAG_0002.

```
Mach> select * from tag where name = 'TAG_0002' and value > 12 and value < 15 and time between to_date('2018-02-01
NAME                  TIME                          VALUE
--------------------------------------------------------------------------------
TAG_0002              2018-02-03 03:00:00 000:000:000 13
TAG_0002              2018-02-04 04:00:00 000:000:000 14
[2] row(s) selected.
```

Extracting Data via RESTful API

Preparations for the RESTful API

You must run Machbase Web Analyzer (MWA) to make the Web service available, and then do the following:

MWA의 수행

```
$ MWAserver start
SERVER STARTED, PID : 27307
     Connection URL : http://192.168.0.148:5001
```

RESTful API calling conventions

SELECT FORM

```
{MWA URL}/machiot-rest-api/datapoints/raw/{TagName}/{Start}/{End}/{Direction}/{Count}/{Offset}/
```

Parameter	Description
TagName	Name of tag, If you want to specify multiple TagName, They should separated by comma(',').
Start, End	Duration. Formats supported are YYYY-MM-DD HH24:MI:SS or YYYY-MM-DD or YYYY-MM-DD HH24:MI:SS,mmm (mmm: millisecond, if it is omitted start is 000, End is 999, micro second and nano seconds are also 999).If you specify as a string, add 'T' between date and time to remove space.
Direction	Currently, only 0 (ascending) is supported.
Count	Limit number of rows. if you specify 0, all of the rows are retrieved.
Offset	Starting offset (default : 0)

A sample of importing single tag data via CURL

If you make a call to Machbase installed on 192.168.0.148 as shown below, you can import the data from the web.

Single Tag

```
$ curl -G "http://192.168.0.148:5001/machiot-rest-api/v1/datapoints/raw/TAG_0001/2018-01-01T00:00:00/2018-01-06T00

{"ErrorCode": 0,
 "ErrorMessage": "",
 "Data": [{"DataType": "DOUBLE",
 "ErrorCode": 0,
 "TagName": "TAG_0001",
 "CalculationMode": "raw",
 "Samples": [{"TimeStamp": "2018-01-01 01:00:00 000:000:000", "Value": 1.0, "Quality": 1},
             {"TimeStamp": "2018-01-02 02:00:00 000:000:000", "Value": 2.0, "Quality": 1},
```

```
            {"TimeStamp": "2018-01-03 03:00:00 000:000:000", "Value": 3.0, "Quality": 1},
            {"TimeStamp": "2018-01-04 04:00:00 000:000:000", "Value": 4.0, "Quality": 1},
            {"TimeStamp": "2018-01-05 05:00:00 000:000:000", "Value": 5.0, "Quality": 1}]}]
}
```

Importing multiple tag data via CURL

Below is a sample example that gets the values for the two tags.

```
$ curl -G "http://192.168.0.148:5001/machiot-rest-api/datapoints/raw/TAG_0001,TAG_0002/2018-01-05T00:00:00/2018-02-
{"ErrorCode": 0,
 "ErrorMessage": "",
 "Data": [{"DataType": "DOUBLE",
           "ErrorCode": 0,
           "TagName": "TAG_0001,TAG_0002",
           "CalculationMode": "raw",
           "Samples": [{"TimeStamp": "2018-01-05 05:00:00 000:000:000", "Value": 5.0, "Quality": 1},
                       {"TimeStamp": "2018-01-06 06:00:00 000:000:000", "Value": 6.0, "Quality": 1},
                       {"TimeStamp": "2018-01-07 07:00:00 000:000:000", "Value": 7.0, "Quality": 1},
                       {"TimeStamp": "2018-01-08 08:00:00 000:000:000", "Value": 8.0, "Quality": 1},
                       {"TimeStamp": "2018-01-09 09:00:00 000:000:000", "Value": 9.0, "Quality": 1},
                       {"TimeStamp": "2018-01-10 10:00:00 000:000:000", "Value": 10.0, "Quality": 1},
                       {"TimeStamp": "2018-02-01 01:00:00 000:000:000", "Value": 11.0, "Quality": 1},
                       {"TimeStamp": "2018-02-02 02:00:00 000:000:000", "Value": 12.0, "Quality": 1},
                       {"TimeStamp": "2018-02-03 03:00:00 000:000:000", "Value": 13.0, "Quality": 1},
                       {"TimeStamp": "2018-02-04 04:00:00 000:000:000", "Value": 14.0, "Quality": 1}
]}]}
```

Specifying Scan Direction with Hint

In general, TAG table query retrieves the oldest record first. When the newest record is to be retrieved, the scan direction can be specified with SELECT HINT.

Forward scanning.

Forward scanning is possible with /*+ SCAN_BACKWARD(table_name) */ hint.

```
Mach> SELECT * FROM tag WHERE t_name='TAG_99' LIMIT 10;
T_NAME                T_TIME                          T_VALUE
-------------------------------------------------------------------------------
TAG_99                2017-01-01 00:00:49 500:000:000 0
TAG_99                2017-01-01 00:01:39 500:000:000 1
TAG_99                2017-01-01 00:02:29 500:000:000 2
TAG_99                2017-01-01 00:03:19 500:000:000 3
TAG_99                2017-01-01 00:04:09 500:000:000 4
TAG_99                2017-01-01 00:04:59 500:000:000 5
TAG_99                2017-01-01 00:05:49 500:000:000 6
TAG_99                2017-01-01 00:06:39 500:000:000 7
TAG_99                2017-01-01 00:07:29 500:000:000 8
TAG_99                2017-01-01 00:08:19 500:000:000 9
[10] row(s) selected.
Elapsed time: 0.001

Mach> SELECT /*+ SCAN_FORWARD(tag) */  * FROM tag WHERE t_name='TAG_99' LIMIT 10;
T_NAME                T_TIME                          T_VALUE
-------------------------------------------------------------------------------
TAG_99                2017-01-01 00:00:49 500:000:000 0
TAG_99                2017-01-01 00:01:39 500:000:000 1
TAG_99                2017-01-01 00:02:29 500:000:000 2
TAG_99                2017-01-01 00:03:19 500:000:000 3
TAG_99                2017-01-01 00:04:09 500:000:000 4
TAG_99                2017-01-01 00:04:59 500:000:000 5
TAG_99                2017-01-01 00:05:49 500:000:000 6
TAG_99                2017-01-01 00:06:39 500:000:000 7
TAG_99                2017-01-01 00:07:29 500:000:000 8
TAG_99                2017-01-01 00:08:19 500:000:000 9
[10] row(s) selected.
Elapsed time: 0.001
```

```
Mach>
```

Backward scanning.

Backward scanning is possible with /*+ SCAN_BACKWARD(table_name) */ hint.

```
Mach> SELECT /*+ SCAN_BACKWARD(tag) */ * FROM tag WHERE t_name='TAG_99' LIMIT 10;
T_NAME                T_TIME                          T_VALUE
--------------------------------------------------------------------------------
TAG_99                2017-02-27 20:53:19 500:000:000 9
TAG_99                2017-02-27 20:52:29 500:000:000 8
TAG_99                2017-02-27 20:51:39 500:000:000 7
TAG_99                2017-02-27 20:50:49 500:000:000 6
TAG_99                2017-02-27 20:49:59 500:000:000 5
TAG_99                2017-02-27 20:49:09 500:000:000 4
TAG_99                2017-02-27 20:48:19 500:000:000 3
TAG_99                2017-02-27 20:47:29 500:000:000 2
TAG_99                2017-02-27 20:46:39 500:000:000 1
TAG_99                2017-02-27 20:45:49 500:000:000 0
[10] row(s) selected.
Elapsed time: 0.001
Mach>
```

Property to Set Default Direction

The defualt direction of scan can be set with **DISK_SCAN_DIRECTION** property. The value can be 0 or 1. Default value is 1.

The scan direction is forward with 1, and backward with 0, when no hint was given in SELECT phrase.

Delete tag data

Tag data deletion constraints

Machbase only supports deletion of data before a specific time.

Unsupported tag data deletion condition

- Delete specific tag data
- Delete data for a specific time range
- Delete specific time range data for a specific tag

Supported tag data deletion condition

- Delete all tags before a specific time

Index

DELETE

Syntax

```
DELETE FROM TAG BEFORE TO_DATE('Time-string');
```

In the above example, if you specify the time of the BEFORE statement, all tags before that time are deleted.

If the BEFORE clause is not specified, all data in the tag table is deleted.

Example of some data deletion

```
# Original Data
Mach> select * from tag;
NAME TIME VALUE
--------------------------------------------------------------------------------
TAG_0001 2018-01-01 01:00:00 000:000:000 1
TAG_0001 2018-01-02 02:00:00 000:000:000 2
TAG_0001 2018-01-03 03:00:00 000:000:000 3
TAG_0001 2018-01-04 04:00:00 000:000:000 4
TAG_0001 2018-01-05 05:00:00 000:000:000 5
TAG_0001 2018-01-06 06:00:00 000:000:000 6
TAG_0001 2018-01-07 07:00:00 000:000:000 7
TAG_0001 2018-01-08 08:00:00 000:000:000 8
TAG_0001 2018-01-09 09:00:00 000:000:000 9
TAG_0001 2018-01-10 10:00:00 000:000:000 10
TAG_0002 2018-02-01 01:00:00 000:000:000 11
TAG_0002 2018-02-02 02:00:00 000:000:000 12
TAG_0002 2018-02-03 03:00:00 000:000:000 13
TAG_0002 2018-02-04 04:00:00 000:000:000 14
TAG_0002 2018-02-05 05:00:00 000:000:000 15
TAG_0002 2018-02-06 06:00:00 000:000:000 16
TAG_0002 2018-02-07 07:00:00 000:000:000 17
TAG_0002 2018-02-08 08:00:00 000:000:000 18
TAG_0002 2018-02-09 09:00:00 000:000:000 19
TAG_0002 2018-02-10 10:00:00 000:000:000 20
[20] row(s) selected.

Mach> delete from tag before to_date('2018-02-01');
10 row(s) deleted.

Mach> select * from tag;
NAME TIME VALUE
--------------------------------------------------------------------------------
TAG_0002 2018-02-01 01:00:00 000:000:000 11
TAG_0002 2018-02-02 02:00:00 000:000:000 12
TAG_0002 2018-02-03 03:00:00 000:000:000 13
TAG_0002 2018-02-04 04:00:00 000:000:000 14
TAG_0002 2018-02-05 05:00:00 000:000:000 15
TAG_0002 2018-02-06 06:00:00 000:000:000 16
```

```
TAG_0002 2018-02-07 07:00:00 000:000:000 17
TAG_0002 2018-02-08 08:00:00 000:000:000 18
TAG_0002 2018-02-09 09:00:00 000:000:000 19
TAG_0002 2018-02-10 10:00:00 000:000:000 20
[10] row(s) selected.
```

Example of deleting all data

```
Mach> delete from tag;
10 row(s) deleted.

Mach> select * from tag;
NAME TIME VALUE
--------------------------------------------------------------------------------
[0] row(s) selected.
```

All of tag data were deleted.

Manipulating Rollup Tables (1)

Manipulating Rollup Tables

As mentioned previously, ROLLUP tables are read-only and designed to quickly get statistics for a particular tag within a desired time range.

ROLLUP table queries can be done by performing a SELECT on the TAG table with HINT to indicate that the query is for ROLLUP.

Index

Syntax

```
SELECT /*+ ROLLUP(TAG, TIME_UNIT, FUNCTION_NAME) */ COLUMN_LIST FROM TAG WHERE
```

- TIME_UNIT : You can specify one of these item (HOUR, MIN, SEC). (Note: if you specify MIN as MINUTE, it will not work correctly.)
- FUNCTION_NAME : You can specify one of these function names (AVG, MIN, MAX, COUNT, SUM).
- COLUMN_LIST : You can specify column name of TAG table. but statistic values are from columns which is specified as SUMMERIZED. (normally value).

Data Sample

Below is sample data for ROLLUP test.

```
create tagdata table TAG (name varchar(20) primary key, time datetime basetime, value double summarized);

insert into tag metadata values ('TAG_0001');

insert into tag values('TAG_0001', '2018-01-01 01:00:01 000:000:000', 1);
insert into tag values('TAG_0001', '2018-01-01 01:00:02 000:000:000', 2);
insert into tag values('TAG_0001', '2018-01-01 01:01:01 000:000:000', 3);
insert into tag values('TAG_0001', '2018-01-01 01:01:02 000:000:000', 4);
insert into tag values('TAG_0001', '2018-01-01 01:02:01 000:000:000', 5);
insert into tag values('TAG_0001', '2018-01-01 01:02:02 000:000:000', 6);

insert into tag values('TAG_0001', '2018-01-01 02:00:01 000:000:000', 1);
insert into tag values('TAG_0001', '2018-01-01 02:00:02 000:000:000', 2);
insert into tag values('TAG_0001', '2018-01-01 02:01:01 000:000:000', 3);
insert into tag values('TAG_0001', '2018-01-01 02:01:02 000:000:000', 4);
insert into tag values('TAG_0001', '2018-01-01 02:02:01 000:000:000', 5);
insert into tag values('TAG_0001', '2018-01-01 02:02:02 000:000:000', 6);

insert into tag values('TAG_0001', '2018-01-01 03:00:01 000:000:000', 1);
insert into tag values('TAG_0001', '2018-01-01 03:00:02 000:000:000', 2);
insert into tag values('TAG_0001', '2018-01-01 03:01:01 000:000:000', 3);
insert into tag values('TAG_0001', '2018-01-01 03:01:02 000:000:000', 4);
insert into tag values('TAG_0001', '2018-01-01 03:02:01 000:000:000', 5);
insert into tag values('TAG_0001', '2018-01-01 03:02:02 000:000:000', 6);
```

ROLLUP force update

Machbase creates ROLLUP data in real time, but in case of HOUR or MIN, it does not update if the time has not been exceeded.

However, since it is necessary to update ROLLUP data even before the time is up, Machbase provides the ability to force update through the **EXEC ROLLUP_FORCE** statement.

This procedure can take a very long time depending on the amount of data, so it should be executed only when absolutely necessary. (In most cases, you do not need to perform this function.)

```
Mach> EXEC ROLLUP_FORCE;
```

```
Executed successfully.
```

Getting average value using ROLLUP data.

Below is an example of getting the average value in seconds, minutes, and hours for a specific tag.

```
Mach> SELECT /*+ ROLLUP(TAG, sec, AVG) */ time, value FROM TAG WHERE name = 'TAG_0001' order by time;
time                          value
---------------------------------------------------------------
2018-01-01 01:00:01 000:000:000 1
2018-01-01 01:00:02 000:000:000 2
2018-01-01 01:01:01 000:000:000 3
2018-01-01 01:01:02 000:000:000 4
2018-01-01 01:02:01 000:000:000 5
2018-01-01 01:02:02 000:000:000 6
2018-01-01 02:00:01 000:000:000 1
2018-01-01 02:00:02 000:000:000 2
2018-01-01 02:01:01 000:000:000 3
2018-01-01 02:01:02 000:000:000 4
2018-01-01 02:02:01 000:000:000 5
2018-01-01 02:02:02 000:000:000 6
2018-01-01 03:00:01 000:000:000 1
2018-01-01 03:00:02 000:000:000 2
2018-01-01 03:01:01 000:000:000 3
2018-01-01 03:01:02 000:000:000 4
2018-01-01 03:02:01 000:000:000 5
2018-01-01 03:02:02 000:000:000 6
[18] row(s) selected.

Mach> SELECT /*+ ROLLUP(TAG, min, AVG) */ time, value FROM TAG WHERE name = 'TAG_0001' order by time;
time                          value
---------------------------------------------------------------
2018-01-01 01:00:00 000:000:000 1.5
2018-01-01 01:01:00 000:000:000 3.5
2018-01-01 01:02:00 000:000:000 5.5
2018-01-01 02:00:00 000:000:000 1.5
2018-01-01 02:01:00 000:000:000 3.5
2018-01-01 02:02:00 000:000:000 5.5
2018-01-01 03:00:00 000:000:000 1.5
2018-01-01 03:01:00 000:000:000 3.5
2018-01-01 03:02:00 000:000:000 5.5
[9] row(s) selected.

Mach> SELECT /*+ ROLLUP(TAG, hour, AVG) */ time, value FROM TAG WHERE name = 'TAG_0001' order by time;
time                          value
---------------------------------------------------------------
2018-01-01 01:00:00 000:000:000 3.5
2018-01-01 02:00:00 000:000:000 3.5
2018-01-01 03:00:00 000:000:000 3.5
[3] row(s) selected.
```

Getting min/max values using ROLLUP data.

Below is an example of getting the minimum / maximum value according to the time range of the tag.

```
Mach> SELECT /*+ ROLLUP(TAG, hour, min) */ time, value FROM TAG WHERE name = 'TAG_0001' order by time;
time                          value
---------------------------------------------------------------
2018-01-01 01:00:00 000:000:000 1
2018-01-01 02:00:00 000:000:000 1
2018-01-01 03:00:00 000:000:000 1
[3] row(s) selected.

Mach> SELECT /*+ ROLLUP(TAG, hour, max) */ time, value FROM TAG WHERE name = 'TAG_0001' order by time;
time                          value
---------------------------------------------------------------
```

```
2018-01-01 01:00:00 000:000:000 6
2018-01-01 02:00:00 000:000:000 6
2018-01-01 03:00:00 000:000:000 6
[3] row(s) selected.

Mach> SELECT /*+ ROLLUP(TAG, min, min) */ time, value FROM TAG WHERE name = 'TAG_0001' order by time;
time                          value
------------------------------------------------------------
2018-01-01 01:00:00 000:000:000 1
2018-01-01 01:01:00 000:000:000 3
2018-01-01 01:02:00 000:000:000 5
2018-01-01 02:00:00 000:000:000 1
2018-01-01 02:01:00 000:000:000 3
2018-01-01 02:02:00 000:000:000 5
2018-01-01 03:00:00 000:000:000 1
2018-01-01 03:01:00 000:000:000 3
2018-01-01 03:02:00 000:000:000 5
[9] row(s) selected.

Mach> SELECT /*+ ROLLUP(TAG, min, max) */ time, value FROM TAG WHERE name = 'TAG_0001' order by time;
time                          value
------------------------------------------------------------
2018-01-01 01:00:00 000:000:000 2
2018-01-01 01:01:00 000:000:000 4
2018-01-01 01:02:00 000:000:000 6
2018-01-01 02:00:00 000:000:000 2
2018-01-01 02:01:00 000:000:000 4
2018-01-01 02:02:00 000:000:000 6
2018-01-01 03:00:00 000:000:000 2
2018-01-01 03:01:00 000:000:000 4
2018-01-01 03:02:00 000:000:000 6
[9] row(s) selected.
```

Getting Sum/Count values using ROLLUP data.

Below is an example of getting the sum / count value according to the time range of the tag.

```
Mach> SELECT /*+ ROLLUP(TAG, min, sum) */ time, value FROM TAG WHERE name = 'TAG_0001' order by time;
time                          value
------------------------------------------------------------
2018-01-01 01:00:00 000:000:000 3
2018-01-01 01:01:00 000:000:000 7
2018-01-01 01:02:00 000:000:000 11
2018-01-01 02:00:00 000:000:000 3
2018-01-01 02:01:00 000:000:000 7
2018-01-01 02:02:00 000:000:000 11
2018-01-01 03:00:00 000:000:000 3
2018-01-01 03:01:00 000:000:000 7
2018-01-01 03:02:00 000:000:000 11
[9] row(s) selected.

Mach> SELECT /*+ ROLLUP(TAG, min, count) */ time, value FROM TAG WHERE name = 'TAG_0001' order by time;
time                          value
------------------------------------------------------------
2018-01-01 01:00:00 000:000:000 2
2018-01-01 01:01:00 000:000:000 2
2018-01-01 01:02:00 000:000:000 2
2018-01-01 02:00:00 000:000:000 2
2018-01-01 02:01:00 000:000:000 2
2018-01-01 02:02:00 000:000:000 2
2018-01-01 03:00:00 000:000:000 2
2018-01-01 03:01:00 000:000:000 2
2018-01-01 03:02:00 000:000:000 2
[9] row(s) selected.
```

Getting ROLLUP data using RESTful API

Syntax

You can get ROLLUP data from the RESTful API using the following syntax:

```
{MWA URL}/machiot-rest-api/datapoints/calculated/{TagName}/{Start}/{End}/{CalculationMode}/{Count}/{IntervalType}/
```

Parameter	Description
TagName	• Specify name of tag • In case of multiple tags, they should separated by comma(',').
Start, End	• Specify time range of data • Supported date format is YYYY-MM-DD HH24:MI:SS or YYYY-MM-DD
CalcurationMode	• Specify type of statistics data • Supported type are (min/max/sum/cnt/avg)
Count	• Specify maximal number of rows • In case of 0, all data will be printed.
IntervalType	• Specify the groping duration • You can specify (sec, min, hour)
IntervalValue	• Specify value for IntervalType • If the IntervalType is min and the IntervalValue is 5, the statistical value of 5 minutes

Average statistics for a minute

Below is an example of outputting the statistical value per minute from the value of TAG_0001.

```
Host:~/work/nfx$ curl -G "http://192.168.0.148:5001/machiot-rest-api/v1/datapoints/calculated/TAG_0001/2018-01-01T0
{"ErrorCode": 0,
 "ErrorMessage": "",
 "Data": [{"DataType": "DOUBLE",
           "ErrorCode": 0,
           "TagName": "TAG_0001",
           "CalculationMode": "avg",
           "Samples": [{"TimeStamp": "2018-01-01 01:00:00 000:000:000", "Value": 1.5, "Quality": 0},
                       {"TimeStamp": "2018-01-01 01:01:00 000:000:000", "Value": 3.5, "Quality": 0},
                       {"TimeStamp": "2018-01-01 01:02:00 000:000:000", "Value": 5.5, "Quality": 0},
                       {"TimeStamp": "2018-01-01 02:00:00 000:000:000", "Value": 1.5, "Quality": 0},
                       {"TimeStamp": "2018-01-01 02:01:00 000:000:000", "Value": 3.5, "Quality": 0},
                       {"TimeStamp": "2018-01-01 02:02:00 000:000:000", "Value": 5.5, "Quality": 0},
                       {"TimeStamp": "2018-01-01 03:00:00 000:000:000", "Value": 1.5, "Quality": 0},
                       {"TimeStamp": "2018-01-01 03:01:00 000:000:000", "Value": 3.5, "Quality": 0},
                       {"TimeStamp": "2018-01-01 03:02:00 000:000:000", "Value": 5.5, "Quality": 0}]}]}
```

Average statistics per 2 minutes

Below is a similar case as above, but with 2 minutes instead of 1 minute.

```
Host:~/work/nfx$ curl -G "http://192.168.0.148:5001/machiot-rest-api/v1/datapoints/calculated/TAG_0001/2018-01-01T0
{"ErrorCode": 0,
 "ErrorMessage": "",
 "Data": [{"DataType": "DOUBLE",
           "ErrorCode": 0,
           "TagName": "TAG_0001",
           "CalculationMode": "avg",
           "Samples": [{"TimeStamp": "2018-01-01 01:00:00 000:000:000", "Value": 2.5, "Quality": 0},
                       {"TimeStamp": "2018-01-01 01:02:00 000:000:000", "Value": 5.5, "Quality": 0},
                       {"TimeStamp": "2018-01-01 02:00:00 000:000:000", "Value": 2.5, "Quality": 0},
                       {"TimeStamp": "2018-01-01 02:02:00 000:000:000", "Value": 5.5, "Quality": 0},
                       {"TimeStamp": "2018-01-01 03:00:00 000:000:000", "Value": 2.5, "Quality": 0},
                       {"TimeStamp": "2018-01-01 03:02:00 000:000:000", "Value": 5.5, "Quality": 0}]}]}
```

As described above, the user can designate arbitrary statistical functions for arbitrary tags in arbitrary time units and output them in real time.

An example of tag table

Index

Introduction

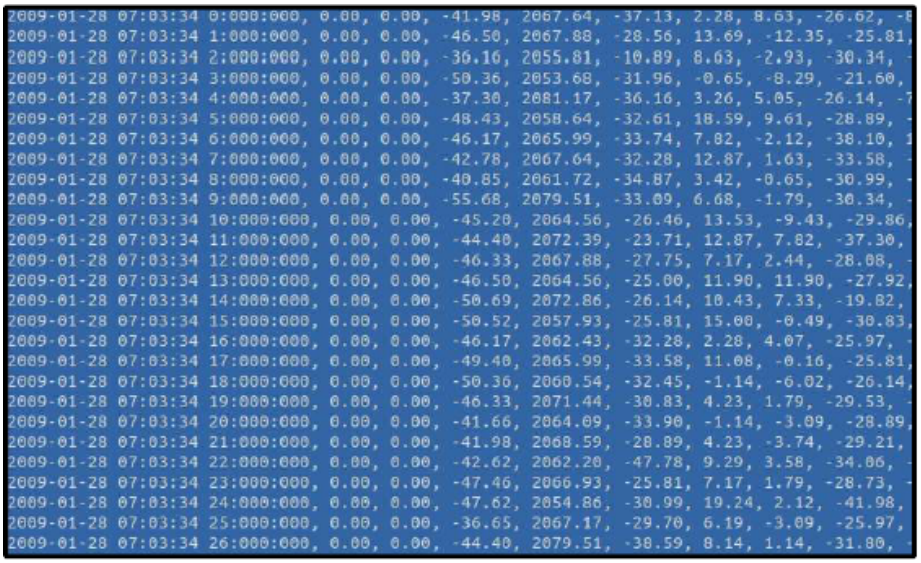
The TAG table can load the structure type of the file in which general sensor data is stored.

The most common types of text storage files are <value, value, value> <value, value, value> <repeat> which is a random file content that just lists a number of numeric values .

In the case of a file containing time, there are <time, value, value, value> <time, value, value, value> <repeat ..>.

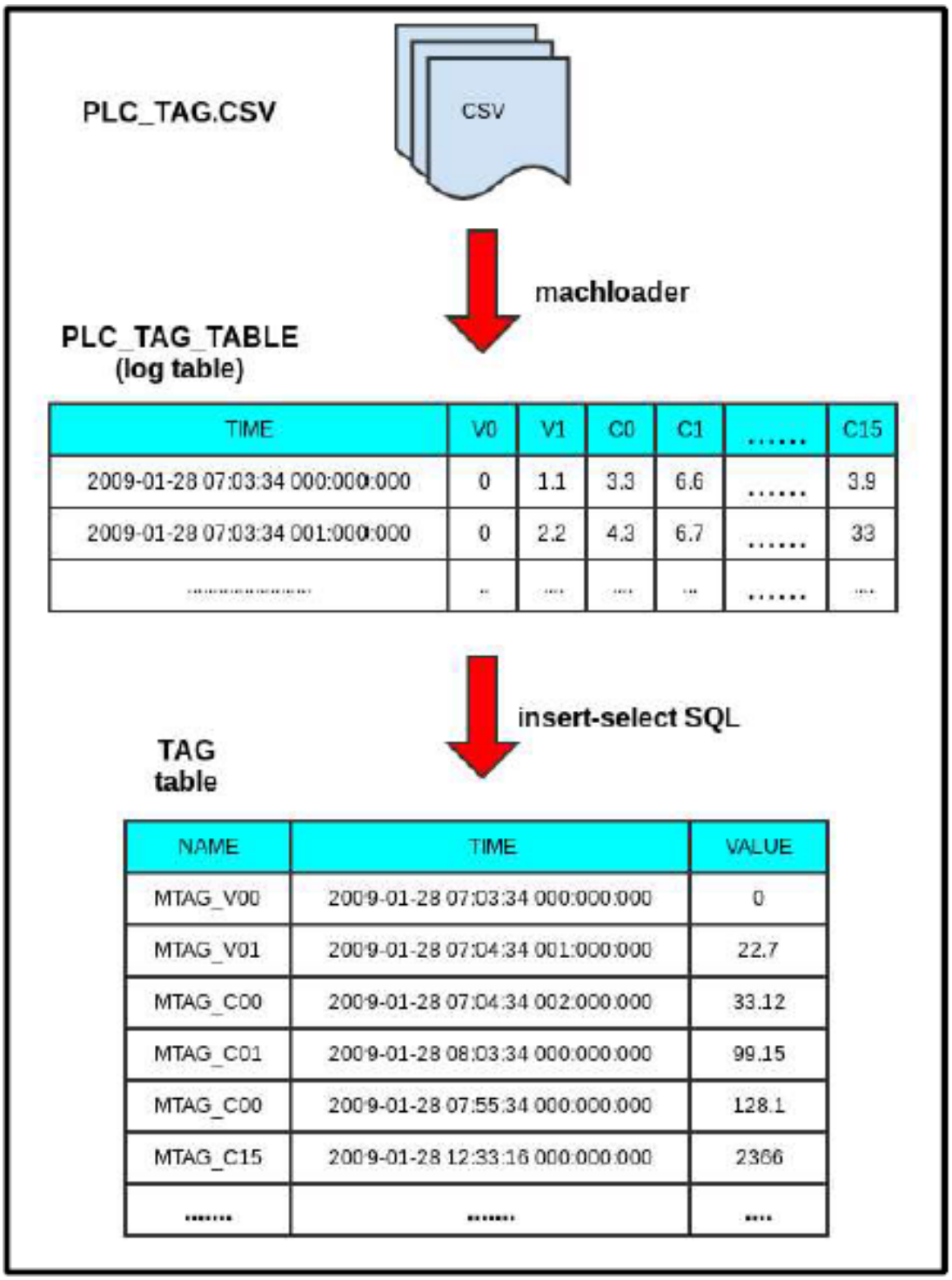
The data in these files is created when a device called PLC (programmable logic controller) collects data from one or more sensors continuously over a long period of time.

The following is the picture of PLC example file.



Now we will load this file into Machbase's TAG table.

Data conversion flowchart



As you can see in the figure above, we will load the raw CSV file into Machbase's log table and convert it into a tag table.

Tag table creation and tag meta loading

Create TAG table as shown below and load the tag names (tag meta) stored in the CSV file at once using a tool called tagmetaimport.

```
Mach> create tagdata table tag (name varchar(32) primary key, time datetime basetime, value double
summarized);
Executed successfully.
Elapsed time: 3.032

$ cat tag_meta.csv
MTAG_V00
MTAG_V01
MTAG_C00
```

```
MTAG_C01
MTAG_C02
MTAG_C03
MTAG_C04
MTAG_C05
MTAG_C06
MTAG_C07
MTAG_C08
MTAG_C09
MTAG_C10
MTAG_C11
MTAG_C12
MTAG_C13
MTAG_C14
MTAG_C15

$ tagmetaimport -d tag_meta.csv
Import time : 0 hour 0 min 0.340 sec
Load success count : 18
```

As shown above, 18 tag meta information were loaded successfully.

Create table for PLC data loading

Execute the following query to create the log table.

```
create table plc_tag_table(
    tm datetime,
    V0 DOUBLE ,
    V1 DOUBLE ,
    C0 DOUBLE ,
    C1 DOUBLE ,
    C2 DOUBLE ,
    C3 DOUBLE ,
    C4 DOUBLE ,
    C5 DOUBLE,
    C6 DOUBLE ,
    C7 DOUBLE ,
    C8 DOUBLE ,
    C9 DOUBLE ,
    C10 DOUBLE ,
    C11 DOUBLE ,
    C12 DOUBLE ,
    C13 DOUBLE ,
    C14 DOUBLE ,
    C15 DOUBLE
);
```

Note that this table is a log type of table (do not get confused by file names). In Machbase, if you do not specify a separate table type, the default type of table is log.

Loading PLC data

Input the plc_tag.csv file, which contains 2 million original PLC data, using the machloader as PLC input in the log table plc_tag_table created above.

In the plc_tag.csv file, the first column is time, then V0, V1, ... Columns are divided up to C15.

```
$ machloader -t plc_tag_table -i -d plc_tag.csv -F "tm YYYY-MM-DD HH24:MI:SS mmm:uuu:nnn"
----------------------------------------------------------------
Machbase Data Import/Export Utility.
Release Version 5.5.0.official
Copyright 2014, MACHBASE Corporation or its subsidiaries.
All Rights Reserved.
----------------------------------------------------------------
NLS : US7ASCII EXECUTE MODE : IMPORT
TARGET TABLE : plc_tag_table DATA FILE : 4_plc_tag.csv
IMPORT MODE : APPEND FIELD TERM : ,
ROW TERM : \n ENCLOSURE : "
```

```
ESCAPE : \ ARRIVAL_TIME : FALSE
ENCODING : NONE HEADER : FALSE
CREATE TABLE : FALSE
Progress bar Imported records Error records
============================== 2000000 0
Import time : 0 hour 0 min 26.544 sec
Load success count : 2000000
Load fail count : 0
```

Tag meta name generation rules

Now you must re-insert data in log table into the tag table in order to see the data through the Tag Analyzer.

For this, the insert-select statement will duplicate each record in the log table into the TAG table.

The name of each tag is determined as follows.

Column name of log table	Tag name values of tag table
V0	MTAG_V00
V1	MTAG_V01
C0	MTAG_C00
C1	MTAG_C01
...	
C15	MTAG_C15

Loading Tag table data

It's time to load the actual data into the tag table.

The following query will generate TAG data based on TAG names.

```
Mach> insert into tag select 'MTAG_V00', tm, v0 from plc_tag_table;
2000000 row(s) inserted.
Elapsed time: 4.898
Mach> insert into tag select 'MTAG_V01', tm, v1 from plc_tag_table;
2000000 row(s) inserted.
Elapsed time: 5.577
Mach> insert into tag select 'MTAG_C00', tm, c0 from plc_tag_table;
2000000 row(s) inserted.
Elapsed time: 6.327
Mach> insert into tag select 'MTAG_C01', tm, c1 from plc_tag_table;
2000000 row(s) inserted.
Elapsed time: 7.445
Mach> insert into tag select 'MTAG_C02', tm, c2 from plc_tag_table;
2000000 row(s) inserted.
Elapsed time: 6.898
Mach> insert into tag select 'MTAG_C03', tm, c3 from plc_tag_table;
2000000 row(s) inserted.
Elapsed time: 7.078
Mach> insert into tag select 'MTAG_C04', tm, c4 from plc_tag_table;
2000000 row(s) inserted.
Elapsed time: 6.799
Mach> insert into tag select 'MTAG_C05', tm, c5 from plc_tag_table;
2000000 row(s) inserted.
Elapsed time: 7.210
Mach> insert into tag select 'MTAG_C06', tm, c6 from plc_tag_table;
2000000 row(s) inserted.
Elapsed time: 9.232
Mach> insert into tag select 'MTAG_C07', tm, c7 from plc_tag_table;
2000000 row(s) inserted.
Elapsed time: 6.398
Mach> insert into tag select 'MTAG_C08', tm, c8 from plc_tag_table;
2000000 row(s) inserted.
Elapsed time: 6.432
Mach> insert into tag select 'MTAG_C09', tm, c9 from plc_tag_table;
2000000 row(s) inserted.
```

```
Elapsed time: 6.734
Mach> insert into tag select 'MTAG_C10', tm, c10 from plc_tag_table;
2000000 row(s) inserted.
Elapsed time: 7.692
Mach> insert into tag select 'MTAG_C11', tm, c11 from plc_tag_table;
2000000 row(s) inserted.
Elapsed time: 8.628
Mach> insert into tag select 'MTAG_C12', tm, c12 from plc_tag_table;
2000000 row(s) inserted.
Elapsed time: 8.229
Mach> insert into tag select 'MTAG_C13', tm, c13 from plc_tag_table;
2000000 row(s) inserted.
Elapsed time: 9.517
Mach> insert into tag select 'MTAG_C14', tm, c14 from plc_tag_table;
2000000 row(s) inserted.
Elapsed time: 7.231
Mach> insert into tag select 'MTAG_C15', tm, c15 from plc_tag_table;
2000000 row(s) inserted.
Elapsed time: 7.830
```

A total of 36 millions of records are created.

Manipulating Rollup Tables (2)

Inquiry of the Manipulating Rollup Tables (1) showed the lookup of the rollup table using a hint.

However, this method has the following disadvantages.

- The average, maximum, and count in a rollup table cannot be obtained from a single query.
- Only 1 second, 1 minute, 1 hour can be used. If it is a specific time unit (for example, 5 seconds), it is difficult to process in a single query.
- You cannot use aggregate functions on other columns.

To solve this inconvenience, a rollup table lookup function using the ROLLUP clause has been added.

> This functionality is supported since 5.7.

Index

Syntax

```
SELECT TIME ROLLUP 3 SECOND, AVG(VALUE) FROM TAG WHERE ...;
```

If you specify a ROLLUP clause after the Datetime column specified in the BASETIME attribute as above, the rollup table lookup is performed.

```
[BASETIME_COLUMN] ROLLUP [PERIOD] [TIME_UNIT]
```

- BASETIME_COLUMN : Datetime column of the TAG table specified by the BASETIME attribute.
- PERIOD : You can specify the range of time units that can be used in the DATE_TRUNC () function. (See below)
- TIME_UNIT : All the time units available in the DATE_TRUNC () function can be used. (See below)

Depending on the TIME_UNIT selection, the rollup table to be queried is different.

nanosecond (nsec)	1000000000 (1 sec)	SECOND
microsecond (usec)	60000000 (60 sec)	SECOND
milisecond (msec)	60000 (60 sec)	SECOND
second (sec)	86400 (1 day)	SECOND
minute (min)	1440 (1 day)	MINUTE
hour	24 (1 day)	HOUR
day	1	HOUR
month	1	HOUR
year	1	HOUR

Because using the ROLLUP clause is a direct lookup of the rollup table, using the aggregate function has the following characteristics:

- You must call an aggregate function on the column specified by the SUMMARIZED attribute. However, only five aggregate functions (SUM, COUNT, MIN, MAX, AVG) are supported in rollup tables.
  - (For non-SUMMARIZED attributes, all aggregate functions can be called only for PRIMARY KEY and METADATA columns.)
- GROUP BY must be done directly with the BASETIME column to ROLLUP.
  - You can use the same ROLLUP clause as it is.

Alternatively, you can add an alias to the ROLLUP clause and create the alias in GROUP BY.

```
SELECT time rollup 3 sec mtime, avg(value)FROM TAGGROUP BY time rollup 3 sec mt
-- OR
SELECT time rollup 3 sec mtime, avg(value)FROM TAGGROUP BY mtime;
```

Data Sample

Below is sample data for ROLLUP test.

```
create tagdata table TAG (name varchar(20) primary key, time datetime basetime, value double summarized);

insert into tag metadata values ('TAG_0001');
```

```
insert into tag values('TAG_0001', '2018-01-01 01:00:01 000:000:000', 1);
insert into tag values('TAG_0001', '2018-01-01 01:00:02 000:000:000', 2);
insert into tag values('TAG_0001', '2018-01-01 01:01:01 000:000:000', 3);
insert into tag values('TAG_0001', '2018-01-01 01:01:02 000:000:000', 4);
insert into tag values('TAG_0001', '2018-01-01 01:02:01 000:000:000', 5);
insert into tag values('TAG_0001', '2018-01-01 01:02:02 000:000:000', 6);

insert into tag values('TAG_0001', '2018-01-01 02:00:01 000:000:000', 1);
insert into tag values('TAG_0001', '2018-01-01 02:00:02 000:000:000', 2);
insert into tag values('TAG_0001', '2018-01-01 02:01:01 000:000:000', 3);
insert into tag values('TAG_0001', '2018-01-01 02:01:02 000:000:000', 4);
insert into tag values('TAG_0001', '2018-01-01 02:02:01 000:000:000', 5);
insert into tag values('TAG_0001', '2018-01-01 02:02:02 000:000:000', 6);

insert into tag values('TAG_0001', '2018-01-01 03:00:01 000:000:000', 1);
insert into tag values('TAG_0001', '2018-01-01 03:00:02 000:000:000', 2);
insert into tag values('TAG_0001', '2018-01-01 03:01:01 000:000:000', 3);
insert into tag values('TAG_0001', '2018-01-01 03:01:02 000:000:000', 4);
insert into tag values('TAG_0001', '2018-01-01 03:02:01 000:000:000', 5);
insert into tag values('TAG_0001', '2018-01-01 03:02:02 000:000:000', 6);
```

Getting average value using ROLLUP data.

Below is an example of getting the average value in seconds, minutes, and hours for a specific tag.

```
Mach> SELECT time rollup 1 sec mtime, avg(value) FROM TAG WHERE name = 'TAG_0001' group by mtime order by mtime;
mtime                          avg(value)
---------------------------------------------------------------
2018-01-01 01:00:01 000:000:000 1
2018-01-01 01:00:02 000:000:000 2
2018-01-01 01:01:01 000:000:000 3
2018-01-01 01:01:02 000:000:000 4
2018-01-01 01:02:01 000:000:000 5
2018-01-01 01:02:02 000:000:000 6
2018-01-01 02:00:01 000:000:000 1
2018-01-01 02:00:02 000:000:000 2
2018-01-01 02:01:01 000:000:000 3
2018-01-01 02:01:02 000:000:000 4
2018-01-01 02:02:01 000:000:000 5
2018-01-01 02:02:02 000:000:000 6
2018-01-01 03:00:01 000:000:000 1
2018-01-01 03:00:02 000:000:000 2
2018-01-01 03:01:01 000:000:000 3
2018-01-01 03:01:02 000:000:000 4
2018-01-01 03:02:01 000:000:000 5
2018-01-01 03:02:02 000:000:000 6
[18] row(s) selected.

Mach> SELECT time rollup 1 min mtime, avg(value) FROM TAG WHERE name = 'TAG_0001' group by mtime order by mtime;
mtime                          avg(value)
---------------------------------------------------------------
2018-01-01 01:00:00 000:000:000 1.5
2018-01-01 01:01:00 000:000:000 3.5
2018-01-01 01:02:00 000:000:000 5.5
2018-01-01 02:00:00 000:000:000 1.5
2018-01-01 02:01:00 000:000:000 3.5
2018-01-01 02:02:00 000:000:000 5.5
2018-01-01 03:00:00 000:000:000 1.5
2018-01-01 03:01:00 000:000:000 3.5
2018-01-01 03:02:00 000:000:000 5.5
[9] row(s) selected.

Mach> SELECT time rollup 1 hour mtime, avg(value) FROM TAG WHERE name = 'TAG_0001' group by mtime order by mtime;
mtime                          avg(value)
---------------------------------------------------------------
2018-01-01 01:00:00 000:000:000 3.5
2018-01-01 02:00:00 000:000:000 3.5
```

```
2018-01-01 03:00:00 000:000:000 3.5
[3] row(s) selected.
Mach> SELECT /*+ ROLLUP(TAG, hour, AVG) */ time, value FROM TAG WHERE name = 'TAG_0001' order by time;
time                          value
-------------------------------------------------------------
2018-01-01 01:00:00 000:000:000 3.5
2018-01-01 02:00:00 000:000:000 3.5
2018-01-01 03:00:00 000:000:000 3.5
[3] row(s) selected.
```

Getting min/max values using ROLLUP data.

Below is an example of getting the minimum / maximum value according to the time range of the tag.

```
Mach> SELECT time rollup 1 hour mtime, min(value), max(value) FROM TAG WHERE name = 'TAG_0001' group by mtime order
mtime                         min(value)                max(value)
-------------------------------------------------------------------------------------------
2018-01-01 01:00:00 000:000:000 1                         6
2018-01-01 02:00:00 000:000:000 1                         6
2018-01-01 03:00:00 000:000:000 1                         6
[3] row(s) selected.

Mach> SELECT time rollup 1 min mtime, min(value), max(value) FROM TAG WHERE name = 'TAG_0001' group by mtime order
mtime                         min(value)                max(value)
-------------------------------------------------------------------------------------------
2018-01-01 01:00:00 000:000:000 1                         2
2018-01-01 01:01:00 000:000:000 3                         4
2018-01-01 01:02:00 000:000:000 5                         6
2018-01-01 02:00:00 000:000:000 1                         2
2018-01-01 02:01:00 000:000:000 3                         4
2018-01-01 02:02:00 000:000:000 5                         6
2018-01-01 03:00:00 000:000:000 1                         2
2018-01-01 03:01:00 000:000:000 3                         4
2018-01-01 03:02:00 000:000:000 5                         6
[9] row(s) selected.
```

Getting Sum/Count values using ROLLUP data.

Below is an example of getting the sum / count value according to the time range of the tag.

```
Mach> SELECT time rollup 1 min  mtime, sum(value), count(value) FROM TAG WHERE name = 'TAG_0001' group by mtime ord
mtime                         sum(value)                count(value)
-------------------------------------------------------------------------------------
2018-01-01 01:00:00 000:000:000 3                         2
2018-01-01 01:01:00 000:000:000 7                         2
2018-01-01 01:02:00 000:000:000 11                        2
2018-01-01 02:00:00 000:000:000 3                         2
2018-01-01 02:01:00 000:000:000 7                         2
2018-01-01 02:02:00 000:000:000 11                        2
2018-01-01 03:00:00 000:000:000 3                         2
2018-01-01 03:01:00 000:000:000 7                         2
2018-01-01 03:02:00 000:000:000 11                        2
[9] row(s) selected.
```

Group at various time intervals

The advantage of the ROLLUP clause is that you don't need to intentionally use DATE_TRUNC () to vary the time interval.

To get the sum of the 3 second intervals and the number of data:
Example The time range is only 0 seconds, 1 second, and 2 seconds, so you can see that all converged to 0 seconds.

As a result, it matches the results of the 'minute rollup' query.

```
Mach> SELECT time rollup 3 sec  mtime, sum(value), count(value) FROM TAG WHERE name = 'TAG_0001' GROUP BY mtime ORD
mtime                         sum(value)                count(value)
```

```
--------------------------------------------------------------------------------
2018-01-01 01:00:00 000:000:000 3                          2
2018-01-01 01:01:00 000:000:000 7                          2
2018-01-01 01:02:00 000:000:000 11                         2
2018-01-01 02:00:00 000:000:000 3                          2
2018-01-01 02:01:00 000:000:000 7                          2
2018-01-01 02:02:00 000:000:000 11                         2
2018-01-01 03:00:00 000:000:000 3                          2
2018-01-01 03:01:00 000:000:000 7                          2
2018-01-01 03:02:00 000:000:000 11                         2
```

Log Table

Concept

The **log table** is a table capable of storing machine log data in which input data is time series data.

Data is infinitely entered into this table, and the value of each field has a unique meaning.
You can also retrieve data through text fields (varchar or text) and perform fast statistical calculations.

Generally, a Machbase table refers to the Log Table.

The characteristics of the log table are as follows.

Existing Hidden Time Columns

Every log table has a hidden column called _arrival_time.
This column stores the time at which the record was generated, and supports nanosecond precision.

Reverse Time Search

The general database is output regardless of the input order when retrieving data.
However, Machbase's log tables always have the most up-to-date data as long as they do not have a sort option through a separate ORDER BY.
This can also be seen in the _arrival_time column.

This is because the importance of recent data in machine log data is much higher than that of previous data.

View-Only Post Input

Machbase's log tables do not allow Update operations. In other words, once user log data is stored in Machbase,

By disallowing changes to the data, it is possible to support the stability of the data and the integrity of the log data itself at the engine level.

Limited Deletion Allowed

Machbase permits the deletion of necessary data in special situations, even if the data can not be changed.
However, you can not delete arbitrary data as in a conventional database, and it is only possible to delete the oldest data sequentially.
With this function, it is possible to conveniently manage data by deleting data periodically from embedded devices that are limited in storage space or devices that are not easy to manage.

Supports Text Search Function

Machbase goes one step further in the way it handles strings in generic databases, and provides word-based searching.
This function is best suited for the purpose of machine log data, and searching for log data stored at a specific time is a major function frequently used in real business situations.

To do this, Machbase provides a real-time reverse index, which enables real-time text search at the same time as insertion of data.

Special Data Type Support

Machbase supports IPv4 and IPv6. This is a special type of Internet address, and many machine log data are represented by this type of address system.
This data type makes it easy to search and extract specific addresses.

It also provides the ability to search or extract some address ranges of a particular address scheme, using extended syntax such as `select * from t1 where ipaddr = '192.168.0.*'`.
You can also use the netmask operator to easily determine whether a particular Internet address is included in a particular address range.

Large Object (LOB) Data Support

The log table provides text and binary types that can store up to 64MB of bytes.

- If the data is required to be retrieved as a text document type, it can be stored as a text type and retrieved.
- If the data is a binary data type such as picture or music, it can be saved as a binary type.

Time-based Partitioning

A log table is a sequence of partition files that maintains a certain number of records and indexes relative to the time axis.
In other words, as the data continues to be entered, a new partition file is created, and if a certain number of records in that partition are full, the next partition will be created.

Partition management mainly reflects the characteristics of log data in which search is performed based on time, and has a great advantage in data input performance.
It is also an easy structure for high-speed data access for statistical analysis.

Operations

- Creating and Managing Log Table
- Log Data Input
- Log Data Extraction
- Deletion of Log Data
- Index for log table
- Example of Log Table

Creating and Managing Log Table

The log table can be simply generated as follows. Let's create a table called sensor_data and delete it.

Data types compatible with Machbase can be found in the SQL Reference Types.

Index

Creating Log Table

Create a log table with the 'CREATE TABLE' syntax.

```
Mach> CREATE TABLE sensor_data
      (
          id VARCHAR(32),
          val DOUBLE
       );
Created successfully.

Mach> DROP TABLE sensor_data;
Dropped successfully.
```

Deleting Log Table

Delete log table with 'DROP TABLE' statement.

```
Mach> DROP TABLE sensor_data;
Dropped successfully.

-- TRUNCATE deletes only data and keeps table.
Mach> TRUNCATE TABLE sensor_data;
Truncated successfully.
```

Log Data Input

There are many ways to input log data into Machbase.

- Insert
- Append
- Import
- Load by SQL

Insert

Similar to other commercial RDBMSs, you can first create the table and enter the data using the INSERT INTO statement.

Machbase provides the 'machsql' tool as an interactive query processor.

Index

Create Table

```
CREATE TABLE table_name ( column1 datatype, column2 datatype, column3 datatype, .... );
```

```
CREATE TABLE sensor_data ( id VARCHAR(32), val DOUBLE );
```

Data Insertion

```
INSERT INTO table_name VALUES (value1, value2, value3, ...);
```

```
INSERT INTO sensor_data VALUES('sensor1', 10.1); INSERT INTO sensor_data VALUES('sensor2', 20.2); INSERT INTO senso
```

Confirm Data Insertion

```
SELECT column1, column2, ... FROM table_name;
```

```
SELECT * FROM sensor_data;
```

Entire Process

Below is an example using machsql.

```
Mach> CREATE TABLE sensor_data (id VARCHAR(32), val DOUBLE);
 Created successfully.
Mach> INSERT INTO sensor_data VALUES('sensor1', 10.1);
 1 row(s) inserted.
Mach> INSERT INTO sensor_data VALUES('sensor2', 20.2);
 1 row(s) inserted.
Mach> INSERT INTO sensor_data VALUES('sensor3', 30.3);
 1 row(s) inserted.
Mach> SELECT * FROM sensor_data;
ID VAL
----------------------------------------------------------------
sensor3 30.3
sensor2 20.2
sensor1 10.1
[3] row(s) selected.
```

Append

This is a fast real-time data input API provided by Machbase.
It can be entered using C, C ++, C #, Java, Python, PHP, or Javascript.
Refer to the SDK Guide for details.

Import

With the machloader tool, you can enter a text file that is separated by a CSV or other delimiter.

See the machloader documentation for a detailed description of the machloader tool.

Index

Create Table

```
CREATE TABLE import_sample
(
    srcip     IPV4,
    srcport   INTEGER,
    dstip     IPV4,
    dstport   INTEGER,
    protocol  SHORT,
    eventlog  VARCHAR(1024),
    eventcode SHORT,
    eventsize LONG
);
```

Importing Data

Use the machloader tool to enter the csv file.

```
machloader  -i  -t  import_sample   -d  sample_data.csv
```

Confirm Data Insert

Check the input data.

```
SELECT  COUNT(*)    FROM    import_sample;
```

Sample Example

Below is a sample process using the actual machloader and machsql.

```
Mach> CREATE TABLE import_sample
     (
         srcip     IPV4,
         srcport   INTEGER,
         dstip     IPV4,
         dstport   INTEGER,
         protocol  SHORT,
         eventlog  VARCHAR(1024),
         eventcode SHORT,
         eventsize LONG
     );
Created successfully.
Mach> quit
```

```
[mach@localhost ~]$ cd $MACHBASE_HOME/sample/quickstart
[mach@localhost ~]$ ls -l sample_data.csv
-rw-r--r--- 1 mach mach 110477124 2017-02-23 15:18 sample_data.csv
```

```
[mach@localhost ~]$ machloader -i -t import_sample -d sample_data.csv
-----------------------------------------------------------------
     Machbase Data Import/Export Utility.
     Release Version x.x.x.official
     Copyright 2014, Machbase Inc. or its subsidiaries.
     All Rights Reserved.
-----------------------------------------------------------------
NLS            : US7ASCII            EXECUTE MODE   : IMPORT
TARGET TABLE   : import_sample
DATA FILE      : sample_data.csv
IMPORT_MODE    : APPEND
FILED TERM     : ,                   ROW TERM       : \n
ENCLOSURE      : "                   ARRIVAL_TIME   : FALSE
ENCODING       : NONE                HEADER         : FALSE
CREATE TABLE   : FALSE
 Progress bar                       Imported records        Error records
                                           1000000                    0
Import time         :  0 hour  0 min  2.39 sec
Load success count  : 1000000
Load fail count     : 0
[mach@localhost ~]$
```

```
Mach> SELECT COUNT(*) FROM import_sample;
COUNT(*)
-----------------------
1000000
[1] row(s) selected.
Mach>
```

Load by SQL

The 'Load Data' statement puts the data in the csv file into Machbase.

First, create a table to store the data, using the first line of the csv file to create the columns.

- The data type of the generated columns is VARCHAR (32768).
- The data file path is a relative path based on `$MACHBASE_HOME`. It can also be set to an absolute path.

To save the table data as csv file, use the SAVE DATA statement.

If you already know the data type for each field in the csv file, you can create the table in advance and enter the data.

If you enter the file 'load_sample.csv' into the LOAD DATA statement, the table 'load_sample' is automatically created.

Contents

Index

Loading Data

```
LOAD DATA INFILE 'sample/quickstart/load_sample.csv' INTO TABLE load_sample AUTO HEADUSE;
```

Confirm Data Loading

```
SELECT * FROM load_sample;
```

Sample Example

Using the sample file, you can do the following.

```
[mach@localhost ~]$ cd $MACHBASE_HOME/sample/quickstart
[mach@localhost ~]$ ls -l load_sample.csv
-rw-r
--r--- 1 root root 2827 2017-02-23 15:01 load_sample.csv

[mach@localhost ~]$ machsql
================================================================
     Machbase Client Query Utility
     Release Version x.x.x.official
     Copyright 2014, Machbase Inc. or its subsidiaries.
     All Rights Reserved
================================================================
Machbase server address (Default:127.0.0.1) :
Machbase user ID  (Default:SYS)
Machbase User Password :
MACH_CONNECT_MODE=INET, PORT=5656

Mach> LOAD DATA INFILE 'sample/quickstart/load_sample.csv' INTO TABLE load_sample AUTO HEADUSE;
50 row(s) loaded. Failed to load 0 row(s).
Mach> DESC load_sample;
----------------------------------------------------------------
NAME                          TYPE               LENGTH
----------------------------------------------------------------
SENSOR_ID                     varchar            32767
EPOCH_TIME                    varchar            32767
E_YEAR                        varchar            32767
```

```
E_MONTH                        varchar             32767
E_DAY                          varchar             32767
E_HOUR                         varchar             32767
E_MINUTE                       varchar             32767
E_SECOND                       varchar             32767
VALUE                          varchar             32767
Mach> SELECT COUNT(*) FROM load_sample;
COUNT(*)
----------------------
50
[1] row(s) selected.
Mach>
```

Log Data Extraction

Machbase can extract data using standard ANSI SQL syntax and also provides an extended syntax to conveniently manipulate time series data.

- Data Retrieval
- Time Series Data Retrieval
- Text Search
- Simple Join
- Network Data Type / Operator

Data Retrieval

You can retrieve data in ANSI standard SQL.

The following example shows a search without creating an index.

In other words, the last input data is outputed first.

For more information, see the SELECT section of the SQL Reference.

Index

Basic access

```
SELECT * FROM table_name;
```

```
Mach> SELECT * FROM mach_log;
DEVICE          TM                              TEMP
-----------------------------------------------------------------
MSG
-------------------------------------------------------------------------------------
192.168.0.1     NULL                            NULL
NULL
192.168.0.2     2014-06-15 19:50:03 484:382:010 82
error code = 20, critical warning
192.168.0.2     2014-06-15 19:50:03 484:382:008 57
error code = 20
192.168.0.1     2014-06-15 19:50:03 484:382:006 99
error code = 10, critical bug
192.168.0.1     2014-06-15 19:50:03 484:382:004 55
error code = 10
192.168.0.2     2014-06-15 19:50:03 484:382:002 31
normal state
192.168.0.1     2014-06-15 19:50:03 484:382:000 32
normal state
[7] row(s) selected.
Mach>
```

View Conditional Clause

```
SELECT column_name,column_name
FROM table_name
WHERE column_name operator value;
```

```
Mach> SELECT * FROM mach_log WHERE device = '192.168.0.1';
DEVICE          TM                              TEMP
-----------------------------------------------------------------
MSG
-------------------------------------------------------------------------------------
192.168.0.1     NULL                            NULL
NULL
192.168.0.1     2014-06-15 19:50:36 488:663:006 99
error code = 10, critical bug
192.168.0.1     2014-06-15 19:50:36 488:663:004 55
error code = 10
192.168.0.1     2014-06-15 19:50:36 488:663:000 32
normal state
[4] row(s) selected.

Mach> SELECT * FROM mach_log WHERE device = '192.168.0.1' AND temp > 30 AND temp < 50;
DEVICE          TM                              TEMP
-----------------------------------------------------------------
MSG
```

```
--------------------------------------------------------------------------------
192.168.0.1     2014-06-15 19:50:36 488:663:000 32
normal state
[1] row(s) selected.

Mach> SELECT * FROM mach_log where device > '192.168.0.1';
DEVICE          TM                              TEMP
--------------------------------------------------------------------
MSG
--------------------------------------------------------------------------------
192.168.0.2     2014-06-15 19:50:36 488:663:010 82
error code = 20, critical warning
192.168.0.2     2014-06-15 19:50:36 488:663:008 57
error code = 20
192.168.0.2     2014-06-15 19:50:36 488:663:002 31
normal state
[3] row(s) selected.

Mach> SELECT * FROM mach_log WHERE msg LIKE '%error%';
DEVICE          TM                              TEMP
--------------------------------------------------------------------
MSG
--------------------------------------------------------------------------------
192.168.0.2     2014-06-15 19:50:36 488:663:010 82
error code = 20, critical warning
192.168.0.2     2014-06-15 19:50:36 488:663:008 57
error code = 20
192.168.0.1     2014-06-15 19:50:36 488:663:006 99
error code = 10, critical bug
192.168.0.1     2014-06-15 19:50:36 488:663:004 55
error code = 10
[4] row(s) selected.
```

Time Series Data Retrieval

The DURATION clause of the SELECT statement defines the time condition to be searched. The main reason for using the DURATION clause is to improve the performance even when retrieving a large amount of data by reducing the search target.

Since Machbase divides and stores the data based on the input time, the data is easily searched based on the time condition. The input time is stored in the auto generated column named '_ARRIVAL_TIME', not the user defined column. Therefore, in order to use Machbase most efficiently, it is better to use the built-in '_ARRIVAL_TIME' column without specifying an additional time column.

Machbase outputs data in the reverse order of the input order. In other words, the newest data is outputted first, and the oldest data is outputted later. Generally, when retrieving time series data, the most recent data is more important and often needs to be obtained first. Also, the data output by all DURATION conditionals is outputted from the latest to the last. If you want to output the reverse, from the past to the latest, you should use the AFTER clause. The syntax is as follows.

Index

Syntax

```
DURATION    time_expression [BEFORE time_expression | TO_DATE(time) ];
DURATION    time_expression [AFTER TO_DATE(time)];
time_expression
 -  ALL
 -  n   year
 -  n   month
 -  n   week
 -  n   day
 -  n   hour
 -  n   minute
 -  n   second
```

DURATION...BEFORE

As mentioned earlier, explicit or undefined use of BEFORE (automatically applying BEFORE) outputs data in the order of the most recent to the oldest.

You can query data by absolute time value or relative time value.

Search Based On Absolute Time Value

```
Mach> CREATE TABLE time_table (id INTEGER);
Created successfully.

Mach> INSERT INTO time_table(_arrival_time, id) VALUES(TO_DATE('2014-6-12 10:00:00', 'YYYY-MM-DD HH24:MI:SS'), 1);
1 row(s) inserted.

Mach> INSERT INTO time_table(_arrival_time, id) VALUES(TO_DATE('2014-6-12 11:00:00', 'YYYY-MM-DD HH24:MI:SS'), 2);
1 row(s) inserted.

Mach> INSERT INTO time_table(_arrival_time, id) VALUES(TO_DATE('2014-6-12 12:00:00', 'YYYY-MM-DD HH24:MI:SS'), 3);
1 row(s) inserted.

Mach> INSERT INTO time_table(_arrival_time, id) VALUES(TO_DATE('2014-6-12 13:00:00', 'YYYY-MM-DD HH24:MI:SS'), 4);
1 row(s) inserted.

Mach> INSERT INTO time_table VALUES(5);
1 row(s) inserted.

Mach> SELECT _arrival_time, * FROM time_table DURATION 1 MINUTE;
_arrival_time                   ID
-----------------------------------------------
2017-02-16 12:17:01 880:937:028 5
[1] row(s) selected.

Mach> SELECT _arrival_time, * FROM time_table DURATION 1 DAY BEFORE TO_DATE('2014-6-12 12:00:00', 'YYYY-MM-DD HH24
_arrival_time                   ID
-----------------------------------------------
2014-06-12 12:00:00 000:000:000 3
```

```
2014-06-12 11:00:00 000:000:000 2
2014-06-12 10:00:00 000:000:000 1
[3] row(s) selected.
```

Search Based On Relative Time Value

A search based on relative time values can be viewed as a search based on the current time.

```
Mach> CREATE TABLE relative_table(id INTEGER);
Created successfully.

Mach> INSERT INTO relative_table values(1);
1 row(s) inserted.

------ WAIT for 30 SECONDS before the second value ------

Mach> INSERT INTO relative_table values(2);
1 row(s) inserted.

Mach> SELECT _arrival_time, * FROM relative_table;
_arrival_time                   ID
-----------------------------------------------
2017-02-16 12:35:34 476:055:014 2
2017-02-16 12:35:04 430:802:356 1
[2] row(s) selected.

Mach> SELECT id FROM relative_table DURATION 30 second ;
id
--------------
2
[1] row(s) selected.

Mach> SELECT id FROM relative_table DURATION 60 second ;
id
--------------
2
1
[2] row(s) selected.

Mach> SELECT id FROM relative_table DURATION 30 second BEFORE 30 second;
id
--------------
1
[1] row(s) selected.
```

DURATION...AFTER

When AFTER is applied, the data is outputted from the past to the latest.

The BEFORE command automatically outputs the data in reverse order based on the input time as compared to the past output.

```
Mach> CREATE TABLE after_table (id INTEGER);
Created successfully.

Mach> INSERT INTO after_table(_arrival_time, id) VALUES(TO_DATE('2016-6-12 10:00:00', 'YYYY-MM-DD HH24:MI:SS'), 1);
1 row(s) inserted.

Mach> INSERT INTO after_table(_arrival_time, id) VALUES(TO_DATE('2016-6-12 11:00:00', 'YYYY-MM-DD HH24:MI:SS'), 2);

Mach> INSERT INTO after_table(_arrival_time, id) VALUES(TO_DATE('2016-6-12 12:00:00', 'YYYY-MM-DD HH24:MI:SS'), 3);
1 row(s) inserted.

Mach> INSERT INTO after_table(_arrival_time, id) VALUES(TO_DATE('2016-6-12 13:00:00', 'YYYY-MM-DD HH24:MI:SS'), 4);
1 row(s) inserted.

Mach> INSERT INTO after_table(_arrival_time, id) VALUES(TO_DATE('2016-6-12 14:00:00', 'YYYY-MM-DD HH24:MI:SS'), 5);
1 row(s) inserted.
```

```
Mach> select _arrival_time, * from after_table duration ALL after TO_DATE('2016-6-12 11:00:00', 'YYYY-MM-DD HH24:MI

_arrival_time                   ID
-----------------------------------------------
2016-06-12 11:00:00 000:000:000 2
2016-06-12 12:00:00 000:000:000 3
2016-06-12 13:00:00 000:000:000 4
2016-06-12 14:00:00 000:000:000 5
[4] row(s) selected.

Mach> select _arrival_time, * from after_table duration ALL before TO_DATE('2016-6-12 13:00:00', 'YYYY-MM-DD HH24:M
_arrival_time                   ID
-----------------------------------------------
2016-06-12 13:00:00 000:000:000 4
2016-06-12 12:00:00 000:000:000 3
2016-06-12 11:00:00 000:000:000 2
2016-06-12 10:00:00 000:000:000 1
[4] row(s) selected.
```

DURATION...FROM/TO

When the user tries to retrieve data based on two absolute times, a conditional form of the form "DURATION FROM A TO B" is used.

A and B are absolute times and are expressed using the TO_DATE function. A and B can be set differently according to the user's intention. E.g,

- When A comes after B, the search direction outputs the data in order of latest to oldest just as it is used for BEFORE.
- When B comes before A, the search direction outputs the data in order of oldest to latest just as it is used for AFTER.

The following example shows how the data is output.

```
Mach> CREATE TABLE from_table (id INTEGER);
Created successfully.

Mach> INSERT INTO from_table(_arrival_time, id) VALUES(TO_DATE('2016-6-12 10:00:00', 'YYYY-MM-DD HH24:MI:SS'), 1);
1 row(s) inserted.

Mach> INSERT INTO from_table(_arrival_time, id) VALUES(TO_DATE('2016-6-12 11:00:00', 'YYYY-MM-DD HH24:MI:SS'), 2);
1 row(s) inserted.

Mach> INSERT INTO from_table(_arrival_time, id) VALUES(TO_DATE('2016-6-12 12:00:00', 'YYYY-MM-DD HH24:MI:SS'), 3);
1 row(s) inserted.

Mach> INSERT INTO from_table(_arrival_time, id) VALUES(TO_DATE('2016-6-12 13:00:00', 'YYYY-MM-DD HH24:MI:SS'), 4);
1 row(s) inserted.

Mach> INSERT INTO from_table(_arrival_time, id) VALUES(TO_DATE('2016-6-12 14:00:00', 'YYYY-MM-DD HH24:MI:SS'), 5);
1 row(s) inserted.

Mach> INSERT INTO from_table(_arrival_time, id) VALUES(TO_DATE('2016-6-12 15:00:00', 'YYYY-MM-DD HH24:MI:SS'), 6);
1 row(s) inserted.

Mach> SELECT _arrival_time, * FROM from_table DURATION FROM TO_DATE('2016-6-12 12:00:00', 'YYYY-MM-DD HH24:MI:SS')
_arrival_time                   ID
-----------------------------------------------
2016-06-12 12:00:00 000:000:000 3
2016-06-12 13:00:00 000:000:000 4
2016-06-12 14:00:00 000:000:000 5
[3] row(s) selected.

Mach> SELECT _arrival_time, * FROM from_table limit 2 DURATION FROM TO_DATE('2016-6-12 12:00:00', 'YYYY-MM-DD HH24
'YYYY-MM-DD HH24:MI:SS');
_arrival_time                   ID
-----------------------------------------------
2016-06-12 12:00:00 000:000:000 3
2016-06-12 13:00:00 000:000:000 4
[2] row(s) selected.

Mach> SELECT _arrival_time, * FROM from_table DURATION FROM TO_DATE('2016-6-12 15:00:00', 'YYYY-MM-DD HH24:MI:SS')
```

```
_arrival_time                   ID
-----------------------------------------------
2016-06-12 15:00:00 000:000:000 6
2016-06-12 14:00:00 000:000:000 5
2016-06-12 13:00:00 000:000:000 4
2016-06-12 12:00:00 000:000:000 3
[4] row(s) selected.

Mach> SELECT _arrival_time, * FROM from_table LIMIT 2 duration FROM TO_DATE('2016-6-12 15:00:00', 'YYYY-MM-DD HH24
'YYYY-MM-DD HH24:MI:SS');
_arrival_time                   ID
-----------------------------------------------
2016-06-12 15:00:00 000:000:000 6
2016-06-12 14:00:00 000:000:000 5
[2] row(s) selected.

Mach> SELECT _arrival_time, * from from_table duration FROM TO_DATE('2016-6-12 13:00:00', 'YYYY-MM-DD HH24:MI:SS')
_arrival_time                   ID
-----------------------------------------------
2016-06-12 13:00:00 000:000:000 4
[1] row(s) selected.

Mach> SELECT _arrival_time, * from from_table duration FROM TO_DATE('2016-6-12 13:00:00', 'YYYY-MM-DD HH24:MI:SS')
_arrival_time                   ID
-----------------------------------------------
2016-06-12 13:00:00 000:000:000 4
2016-06-12 14:00:00 000:000:000 5
2016-06-12 15:00:00 000:000:000 6
[3] row(s) selected.

Mach> SELECT _arrival_time, * from from_table duration FROM TO_DATE('2016-6-12 20:00:00', 'YYYY-MM-DD HH24:MI:SS')
_arrival_time                   ID
-----------------------------------------------
2016-06-12 15:00:00 000:000:000 6
2016-06-12 14:00:00 000:000:000 5
2016-06-12 13:00:00 000:000:000 4
[3] row(s) selected.
```

Text Search

This document deals with text search using keyword indexes.

Text search is faster than comparable DBMS LIKE search because it searches a special kind of index called "reverse index" to search the desired string pattern. Keyword indexes can only be created for varchar and text type columns, which are variable-length character columns. However, the search target string must match exactly. Machbase does not perform keywords based on special characters or morphological analysis.

Index

SEARCH

```
SELECT  column_name(s)
FROM    table_name
WHERE   column_name
SEARCH  pattern;
```

```
Mach> CREATE TABLE search_table (id INTEGER, name VARCHAR(20));
Created successfully.

Mach> CREATE INDEX idx_SEARCH ON SEARCH_table (name) INDEX_TYPE KEYWORD;
Created successfully.

Mach> INSERT INTO search_table VALUES(1, 'time flys');
1 row(s) inserted.

Mach> INSERT INTO search_table VALUES(1, 'time runs');
1 row(s) inserted.

Mach> SELECT * FROM search_table WHERE name SEARCH 'time' OR name SEARCH 'runs2' ;
ID          NAME
------------------------------------
1           time runs
1           time flys
[2] row(s) selected.

Mach> SELECT * FROM search_table WHERE name SEARCH 'time' AND name SEARCH 'runs2' ;
ID          NAME
------------------------------------
[0] row(s) selected.

Mach> SELECT * FROM search_table WHERE name SEARCH 'flys' OR name SEARCH 'runs2' ;
ID          NAME
------------------------------------
1           time flys
[1] row(s) selected.
```

Multilingual Search

Machbase can search variable-length strings of various kinds of languages stored in ASCII and UTF-8. In order to search only part of a sentence in a language such as Korean or Japanese, a 2-gram technique is used.

```
SELECT  column_name(s)
FROM    table_name
WHERE   column_name
SEARCH  pattern;
```

```
Mach> CREATE TABLE multi_table (message varchar(100));
Created successfully.

Mach> CREATE INDEX idx_multi ON multi_table(message)INDEX_TYPE KEYWORD;
Created successfully.

Mach> INSERT INTO multi_table VALUES("Machbase is the combination of ideal solutions");
1 row(s) inserted.

Mach> INSERT INTO multi_table VALUES("Machbase is a columnar DBMS");
1 row(s) inserted.

Mach> INSERT INTO multi_table VALUES("Machbaseは理想的なソリューションの組み合わせです");
1 row(s) inserted.

Mach> INSERT INTO multi_table VALUES("Machbaseは円柱状のDBMSです");
1 row(s) inserted.

Mach>  SELECT * from multi_table WHERE message SEARCH 'Machbase DBMS';
MESSAGE
----------------------------------------------------------------------------------
Machbaseは円柱状のDBMSです
Machbase is a columnar DBMS
[2] row(s) selected.

Mach> SELECT * from multi_table WHERE message SEARCH 'DBMS is';
MESSAGE
----------------------------------------------------------------------------------
Machbase is a columnar DBMS
[1] row(s) selected.

Mach> SELECT * from multi_table WHERE message SEARCH 'DBMS' OR message SEARCH 'ideal';
MESSAGE
----------------------------------------------------------------------------------
Machbaseは円柱状のDBMSです
Machbase is a columnar DBMS
Machbase is the combination of ideal solutions
[3] row(s) selected.

Mach> SELECT * from multi_table WHERE message SEARCH '組み合わせ';
MESSAGE
----------------------------------------------------------------------------------
Machbaseは理想的なソリューションの組み合わせです
[1] row(s) selected.
Elapsed time: 0.001
Mach> SELECT * from multi_table WHERE message SEARCH '円柱';
MESSAGE
----------------------------------------------------------------------------------
Machbaseは円柱状のDBMSです
[1] row(s) selected.
```

When the input data is "Republic of Korea", three words of "대한," "한민," and "민국" are recorded in the index. Therefore, you can search for "Korea" with the keywords "대한" or "민국".

Basically, the keywords entered in the search statement are searched by the AND condition, so even if you enter only three words, the result is displayed very accurately. For example, if the search target keyword is a "computer utilization guide", the three words "computer", "utilization", and "guide" are set as AND conditions.

ESEARCH

The ESEARCH operator is used to expand the search target keyword. The search target keyword must be ASCII. Search keywords can be set using the % character. Using a keyword that begins with the % character, such as the LIKE conditional, searches all records, but searches for this condition on words in the keyword index, which makes searching faster than LIKE. This feature is useful for quickly searching for alphabet strings (such as error statements or code).

```
SELECT  column_name(s)
FROM    table_name
WHERE   column_name
ESEARCH pattern;
```

```
Mach> CREATE TABLE esearch_table(id INTEGER, name VARCHAR(20), data VARCHAR(40));
```

```
Created successfully.

Mach> CREATE INDEX idx1 ON esearch_table(name) INDEX_TYPE KEYWORD;
Created successfully.

Mach> CREATE INDEX idx2 ON esearch_table(data) INDEX_TYPE KEYWORD;
Created successfully.

Mach> INSERT INTO esearch_table VALUES(1, 'machbase', 'Real-time search technology');
1 row(s) inserted.

Mach> INSERT INTO esearch_table VALUES(2, 'mach2flux', 'Real-time data compression');
1 row(s) inserted.

Mach> INSERT INTO esearch_table VALUES(3, 'DB MS', 'Memory cache technology');
1 row(s) inserted.

Mach> INSERT INTO esearch_table VALUES(4, 'ファ ッションアドバイザー、', 'errors');
1 row(s) inserted.

Mach> INSERT INTO esearch_table VALUES(5, '인피 니 플럭스', 'socket232');
1 row(s) inserted.

Mach> SELECT * FROM esearch_table where name ESEARCH '%mach';
ID          NAME                 DATA
--------------------------------------------------------------------------------
1           machbase           Real-time search technology
[1] row(s) selected.
Elapsed time: 0.001
Mach> SELECT * FROM esearch_table where data ESEARCH '%echn%';
ID          NAME                 DATA
--------------------------------------------------------------------------------
3           DB MS                Memory cache technology
1           machbase           Real-time search technology
[2] row(s) selected.

Mach> SELECT * FROM esearch_table where name ESEARCH '%피니%럭스';
ID          NAME                 DATA
--------------------------------------------------------------------------------
[0] row(s) selected.

Mach> SELECT * FROM esearch_table where data ESEARCH '%232';
ID          NAME                 DATA
--------------------------------------------------------------------------------
5           인피 니 플럭스  socket232
[1] row(s) selected.
```

REGEXP

The REGEXP operator is used to perform a text search on data through a regular expression. The REGEXP operator is executed by performing a regular expression on the target column, and because the index is not available, the search performance may be degraded. Therefore, it is a good idea to add another search condition that can use the index as an AND operator to improve the search speed.

Applying a SEARCH or ESEARCH operator that can use an index before searching for a particular regular expression pattern is a good way to improve search performance by first reducing the result set and then using REGEXP.

```
Mach> CREATE TABLE regexp_table(id INTEGER, name VARCHAR(20), data VARCHAR(40));
Created successfully.

Mach> INSERT INTO regexp_table VALUES(1, 'machbase', 'Real-time search technology');
1 row(s) inserted.

Mach> INSERT INTO regexp_table VALUES(2, 'mach2base', 'Real-time data compression');
1 row(s) inserted.

Mach> INSERT INTO regexp_table VALUES(3, 'DBMS', 'Memory cache technology');
1 row(s) inserted.
```

```
Mach> INSERT INTO regexp_table VALUES(4, 'ファ ッショ', 'errors');
1 row(s) inserted.

Mach> INSERT INTO regexp_table VALUES(5, '인피니플럭스', 'socket232');
1 row(s) inserted.

Mach> SELECT * FROM regexp_table WHERE name REGEXP 'mach';
ID          NAME                 DATA
------------------------------------------------------------------------------
2           mach2base           Real-time data compression
1           machbase            Real-time search technology
[2] row(s) selected.

Mach> SELECT * FROM regexp_table WHERE data REGEXP 'mach[1]';
ID          NAME                 DATA
------------------------------------------------------------------------------
[0] row(s) selected.

Mach> SELECT * FROM regexp_table WHERE data REGEXP '[A-Za-z]';
ID          NAME                 DATA
------------------------------------------------------------------------------
5           인피니플럭스  socket232
4           ファ ッショ      errors
3           DBMS                 Memory cache technology
2           mach2base           Real-time data compression
1           machbase            Real-time search technology
[5] row(s) selected.
```

LIKE

Machbase also supports the SQL standard LIKE operator. The LIKE operator is available in Korean, Japanese, and Chinese.

```
SELECT  column_name(s)
FROM    table_name
WHERE   column_name
LIKE    pattern;
```

Example:

```
Mach> CREATE TABLE like_table (id INTEGER, name VARCHAR(20), data VARCHAR(40));
Created successfully.

Mach> INSERT INTO like_table VALUES(1, 'machbase', 'Real-time search technology');
1 row(s) inserted.

Mach> INSERT INTO like_table VALUES(2, 'mach2base', 'Real-time data compression');
1 row(s) inserted.

Mach> INSERT INTO like_table VALUES(3, 'DBMS', 'Memory cache technology');
1 row(s) inserted.

Mach> INSERT INTO like_table VALUES(4, 'ファ ッションアドバイザー、', 'errors');
1 row(s) inserted.

Mach> INSERT INTO like_table VALUES(5, '인피 니 플럭스', 'socket232');
1 row(s) inserted.

Mach> SELECT * FROM like_table WHERE name LIKE 'mach%';
ID          NAME                 DATA
------------------------------------------------------------------------------
2           mach2base           Real-time data compression
1           machbase            Real-time search technology
[2] row(s) selected.

Mach> SELECT * FROM like_table WHERE name LIKE '%니%';
ID          NAME                 DATA
------------------------------------------------------------------------------
```

```
5           인피 니 플럭스  socket232
[1] row(s) selected.

Mach> SELECT * FROM like_table WHERE data LIKE '%technology';
ID          NAME                  DATA
------------------------------------------------------------------------------
3           DBMS                  Memory cache technology
1           machbase            Real-time search technology
[2] row(s) selected.
```

Simple Join

Log tables, volatile tables, lookup tables and meta tables can be searched by Join.

Index

Simple Join

```
Mach> CREATE TABLE logtable (code INT,value INT);
Created successfully.

Mach> INSERT INTO logtable VALUES(1,20 );
1 row(s) inserted.

Mach> INSERT INTO logtable VALUES(2,10 );
1 row(s) inserted.

Mach> INSERT INTO logtable VALUES(3,15 );
1 row(s) inserted.

Mach> INSERT INTO logtable VALUES(4,20 );
1 row(s) inserted.

Mach> INSERT INTO logtable VALUES(5,10 );
1 row(s) inserted.

Mach> CREATE VOLATILE table VTABLE (code INT,name VARCHAR(32));
Created successfully.

Mach> INSERT INTO vtable VALUES(1, 'Sam');
1 row(s) inserted.

Mach> INSERT INTO vtable VALUES(3, 'Thomas');
1 row(s) inserted.

Mach> INSERT INTO vtable VALUES(5, 'Micheal');
1 row(s) inserted.

Mach> INSERT INTO vtable VALUES(7, 'Jessica');
1 row(s) inserted.

Mach> SELECT name,value FROM logtable, vtable WHERE logtable.code=vtable.code;
name                            value
------------------------------------------------
Micheal                         10
Thomas                          15
Sam                             20
[3] row(s) selected.
```

Join Using Alias

When using Join, an alias can be used for the join target table.

```
SELECT c.name FROM m$sys_tables t, m$sys_columns c WHERE t.id = c.table_id AND t.name = 'T1'
AND c.id NOT IN(0, 65534) ORDER BY c.name;

c.name
-------------------------------------------
ADDR
ISTYPE
SRCIP
[3] row(s) selected.
```

GROUP BY/ORDER BY

GROUP BY, ORDER BY, and aggregate functions are also available.

```
Mach> SELECT t.name, COUNT(c.name) FROM m$sys_columns c, m$sys_tables t WHERE t.id = c.table_id GROUP BY t.name ORD
t.name                                   count(c.name)
-------------------------------------------------------------------
COMMON_TABLE                             5
DURATIONT                                3
[2] row(s) selected.
```

Join without JOIN clause

A join query without a JOIN clause causes an error. Because there is so much data in the log table, the speed of queries without join conditionality is unpredictably slow.

Also, two log table joins can be very slow. So, when designing a database, it is better to design so that join does not occur considering denormalization.

```
Mach> CREATE TABLE log_table1(i1 INTEGER);
Created successfully.
Mach> INSERT INTO log_table1 VALUES(1);
1 row(s) inserted.
Mach> INSERT INTO log_table1 VALUES(20);
1 row(s) inserted.
Mach> INSERT INTO log_table1 VALUES(30);
1 row(s) inserted.


Mach>CREATE TABLE log_table2(i1 INTEGER);
Created successfully.
Mach> INSERT INTO log_table2 VALUES(1);
1 row(s) inserted.
Mach> INSERT INTO log_table2 VALUES(30);
1 row(s) inserted.
Mach> INSERT INTO log_table2 VALUES(50);
1 row(s) inserted.

Mach> SELECT log_table1.i1 FROM log_table1, log_table2;
[ERR-02101 : Error in joining tables. Cannot join without join predicate.]

Mach> SELECT log_table1.i1 FROM log_table1, log_table2 where log_table1.i1 = 1;
[ERR-02101 : Error in joining tables. Cannot join without join predicate.]

Mach> SELECT log_table1.i1 from log_table1, log_table2 WHERE log_table1.i1 = log_table2.i1;
i1
--------------
30
1
[2] row(s) selected.
```

Inner Join / Outer Join

ANSI type INNER, LEFT OUTER, or RIGHT OUTER join can be used, but FULL OUTER JOIN can not be used.

```
FROM    TABLE_1 [INNER|LEFT OUTER|RIGHT OUTER]  JOIN    TABLE_2 ON  expression
```

```
SELECT t1.i1 t2.i1 FROM t1 LEFT OUTER JOIN t2 ON (t1.i1 = t2.i1) WHERE t2.i2 = 1;
```

The above query is changed to Inner Join by t2.i2 = 1 condition in the where clause.

Network Data Type / Operator

Machbase supports network data types and supports functions available in SELECT statements.

- IPv4 format: 4 byte address type
- IPv6 format: 16 byte address type
- Network mask: Network mask specification format (/ number of bits) for IPv4 or IPv6

Index

IPv4

INSERT

```
INSERT INTO table_name VALUES (value1,value2,value3,...);
```

```
CREATE TABLE addrtable (addr IPV4);
INSERT  INTO addrtable VALUES ('127.0.0.1');
INSERT  INTO addrtable VALUES ('127.0' || '.0.2');
INSERT  INTO addrtable VALUES ('127.0.0.3');
INSERT  INTO addrtable VALUES ('127.0.0.4');
INSERT  INTO addrtable VALUES ('127.0.0.5');
INSERT  INTO addrtable VALUES ('255.255.255.255');
```

SELECT

```
SELECT  column_name,column_name FROM    table_name;
```

```
Mach> SELECT addr FROM addrtable WHERE addr = '127.0.0.3' or addr = '127.0.0.5';
addr
------------------
127.0.0.5
127.0.0.3
[2] row(s) selected.

Mach> SELECT addr FROM addrtable WHERE addr > '127.0.0.3' AND addr < '127.0.0.5';
addr
------------------
127.0.0.4
[1] row(s) selected.

Mach> SELECT addr FROM addrtable WHERE addr <> '127.0.0.3';
addr
------------------
255.255.255.255
127.0.0.5
127.0.0.4
127.0.0.2
127.0.0.1
[5] row(s) selected.

Mach> SELECT addr FROM addrtable WHERE addr = '127.0.0.*';
addr
------------------
127.0.0.5
127.0.0.4
127.0.0.3
127.0.0.2
127.0.0.1
[5] row(s) selected.

Mach> SELECT addr FROM addrtable WHERE addr = '*.0.0.*';
addr
```

```
------------------
127.0.0.5
127.0.0.4
127.0.0.3
127.0.0.2
127.0.0.1
[5] row(s) selected.
```

IPv6

INSERT

```
INSERT  INTO    table_name  VALUES  (value1,value2,value3,...);
```

```
CREATE TABLE addrtable6 (addr ipv6);
INSERT INTO addrtable6 VALUES ('::0.0.0.0');
INSERT INTO addrtable6 VALUES ('::127.0' || '.0.1');
INSERT INTO addrtable6 VALUES ('::127.0.0.3');
INSERT INTO addrtable6 VALUES ('::127.0.0.4');
INSERT INTO addrtable6 VALUES ('21DA:D3:0:2F3B:2AA:FF:FE28:9C5A');
INSERT INTO addrtable6 VALUES ('::FFFF:255.255.255.255');
```

SELECT

```
SELECT  column_name,column_name FROM    table_name;
```

```
Mach> SELECT addr FROM addrtable6 WHERE addr = '::127.0.0.3' or addr = '::127.0.0.5';
addr
----------------------------------------------------------------
::127.0.0.3
[1] row(s) selected.

Mach> SELECT addr FROM addrtable6 WHERE addr > '::127.0.0.3' and addr < '::127.0.0.5';
addr
----------------------------------------------------------------
::127.0.0.4
[1] row(s) selected.

Mach> SELECT addr FROM addrtable6 WHERE addr <> '::127.0.0.3';
addr
----------------------------------------------------------------
::ffff:255-255.255.255
21da:d3::2f3b:2aa:ff:fe28:9c5a
::127.0.0.4
::127.0.0.1
::
[5] row(s) selected.

Mach> SELECT addr FROM addrtable6 WHERE addr >= '21DA::';
addr
----------------------------------------------------------------
21da:d3::2f3b:2aa:ff:fe28:9c5a
[1] row(s) selected.

Mach> SELECT addr FROM addrtable6 order by addr desc;
addr
----------------------------------------------------------------
21da:d3::2f3b:2aa:ff:fe28:9c5a
::ffff:255.255.255.255
::127.0.0.4
```

```
::127.0.0.3
::127.0.0.1
::
[6] row(s) selected.
```

Network Mask

The network mask is an expression format that specifies whether a particular address is included in a particular network. Machbase supports network mask types and related operators.

Mask Representation Type

Like the normal network representation, the network address is represented by the / symbol and the number of bits at the end.

```
'192.128.0.0/16'
'FFFF::192.128.99.0/32'
```

Mask Operator

CONTAINS

This operator should have a network mask on the left and a network address data type on the right. In other words, it checks whether the input address is included in a given network mask. The NOT operator can be used together.

```
SELECT addr FROM addrtable  WHERE '192.0.0.0/16'    CONTAINS    addr;
SELECT addr FROM addrtable  WHERE '192.128.99.0/32' NOT CONTAINS    addr;
```

CONTAINED

Oppositely to CONTAINS, the network address is left and the network mask is right. It checks whether the left address is part of the right mask.

```
SELECT addr FROM addrtable  WHERE addr CONTAINED '192.0.0.0/16';
SELECT addr FROM addrtable  WHERE addr NOT CONTAINED '192.128.99.0/32';
```

Example of Mask Usage

An example of a search using the network mask type is as follows.

```
CREATE TABLE ip_table (addr4 IPV4, addr6 IPV6);

INSERT INTO ip_table VALUES ('192.0.0.1','FFFF::192.0.0.1');
INSERT INTO ip_table VALUES ('192.0.10.1','FFFF::192.0.10.1');
INSERT INTO ip_table VALUES ('192.128.0.1','FFFF::192.128.0.1');
INSERT INTO ip_table VALUES ('192.128.99.128','FFFF::192.128.99.128');
INSERT INTO ip_table VALUES ('192.128.99.64','FFFF::192.128.99.64');
INSERT INTO ip_table VALUES ('192.128.99.32','FFFF::192.128.99.32');
INSERT INTO ip_table VALUES ('192.128.99.16','FFFF::192.128.99.16');
INSERT INTO ip_table VALUES ('192.128.99.8','FFFF::192.128.99.8');
INSERT INTO ip_table VALUES ('192.128.99.4','FFFF::192.128.99.4');
INSERT INTO ip_table VALUES ('192.128.99.2','FFFF::192.128.99.2');
INSERT INTO ip_table VALUES ('192.128.99.1','FFFF::192.128.99.1');

Mach> SELECT addr4 FROM ip_table WHERE '192.0.0.0/16' CONTAINS addr4;
addr4
-----------
192.0.10.1
192.0.0.1
[2] row(s) selected.

Mach> SELECT addr4 FROM ip_table WHERE '192.128.0.0/16' CONTAINS addr4;
addr4
```

```
-----------
192.128.99.1
192.128.99.2
192.128.99.4
192.128.99.8
192.128.99.16
192.128.99.32
192.128.99.64
192.128.99.128
192.128.0.1
[9] row(s) selected.

Mach> SELECT addr4 FROM ip_table WHERE '192.0.10.0/24' CONTAINS addr4;
addr4
------------------------------------------------------------------
192.0.10.1
[1] row(s) selected.

Mach> SELECT addr4 FROM ip_table WHERE '192.128.99.0/31' CONTAINS addr4;
addr4
-------------------------------------------------------
192.128.99.1
[1] row(s) selected.

Mach> SELECT addr4 FROM ip_table WHERE '192.128.99.0/32' NOT CONTAINS addr4;
addr4
-----------
192.128.99.1
192.128.99.2
192.128.99.4
192.128.99.8
192.128.99.16
192.128.99.32
192.128.99.64
192.128.99.128
192.128.0.1
192.0.10.1
192.0.0.1
[11] row(s) selected.

Mach> SELECT addr4 FROM ip_table WHERE addr4 CONTAINED '192.0.0.0/16';
addr4
-------------------------------------
192.0.10.1
192.0.0.1
[2] row(s) selected.

Mach> SELECT addr4 FROM ip_table WHERE addr4 CONTAINED '192.128.0.0/16';
addr4
-------------------------------------
192.128.99.1
192.128.99.2
192.128.99.4
192.128.99.8
192.128.99.16
192.128.99.32
192.128.99.64
192.128.99.128
192.128.0.1
[9] row(s) selected.

Mach> SELECT addr4 FROM ip_table WHERE addr4 CONTAINED '192.0.10.0/24';
addr4
----------------------------
192.0.10.1
[1] row(s) selected.

Mach> SELECT addr4 FROM ip_table WHERE addr4 not CONTAINED '192.128.99.0/32';
addr4
-------------------------------------------------
```

```
192.128.99.1
192.128.99.2
192.128.99.4
192.128.99.8
192.128.99.16
192.128.99.32
192.128.99.64
192.128.99.128
192.128.0.1
192.0.10.1
192.0.0.1
[11] row(s) selected.

Mach> SELECT addr6 FROM ip_table WHERE 'FFFF::192.0.0.0/104' CONTAINS addr6;
addr6
------------------------------------
ffff::c080:6301
ffff::c080:6302
ffff::c080:6304
ffff::c080:6308
ffff::c080:6310
ffff::c080:6320
ffff::c080:6340
ffff::c080:6380
ffff::c080:1
ffff::c000:a01
ffff::c000:1
[11] row(s) selected.

Mach> SELECT addr6 FROM ip_table WHERE 'FFFF::192.128.0.0/112' CONTAINS addr6;
addr6
-----------------------------------
ffff::c080:6301
ffff::c080:6302
ffff::c080:6304
ffff::c080:6308
ffff::c080:6310
ffff::c080:6320
ffff::c080:6340
ffff::c080:6380
ffff::c080:1
[9] row(s) selected.

Mach> SELECT addr6 FROM ip_table WHERE 'FFFF::192.0.10.0/120' CONTAINS addr6;
addr6
-----------------------------------------------
ffff::c000:a01
[1] row(s) selected.

Mach> SELECT addr6 FROM ip_table WHERE 'FFFF::192.128.99.0/31' CONTAINS addr6;
addr6
-------------------------------------------
ffff::c080:6301
ffff::c080:6302
ffff::c080:6304
ffff::c080:6308
ffff::c080:6310
ffff::c080:6320
ffff::c080:6340
ffff::c080:6380
ffff::c080:1
ffff::c000:a01
ffff::c000:1
[11] row(s) selected.

Mach> SELECT addr6 FROM ip_table WHERE 'FFFF::192.128.99.0/32' not CONTAINS addr6;
addr6
------------------------------------
[0] row(s) selected.
```

```
Mach> SELECT addr6 FROM ip_table WHERE addr6 CONTAINED 'FFFF::192.0.0.0/104';
addr6
-------------------------------------
ffff::c080:6301
ffff::c080:6302
ffff::c080:6304
ffff::c080:6308
ffff::c080:6310
ffff::c080:6320
ffff::c080:6340
ffff::c080:6380
ffff::c080:1
ffff::c000:a01
ffff::c000:1
[11] row(s) selected.

Mach> SELECT addr6 FROM ip_table WHERE addr6 CONTAINED 'FFFF::192.128.0.0/112';
addr6
-------------------------------------
ffff::c080:6301
ffff::c080:6302
ffff::c080:6304
ffff::c080:6308
ffff::c080:6310
ffff::c080:6320
ffff::c080:6340
ffff::c080:6380
ffff::c080:1
[9] row(s) selected.

Mach> SELECT addr6 FROM ip_table WHERE addr6 CONTAINED 'FFFF::192.0.10.0/120';
addr6
-------------------------------------
ffff::c000:a01
[1] row(s) selected.

Mach> SELECT addr6 FROM ip_table WHERE addr6 not CONTAINED 'FFFF::192.128.99.0/32';
addr6
-------------------------------------
[0] row(s) selected.
```

Deletion of Log Data

The DELETE statement in Machbase can be performed on the log table.

In addition, it is not possible to delete data in an arbitrary position in the middle, and it is possible to erase consecutively from the arbitrary position to the last (oldest log) record. This is a policy that takes advantage of the characteristics of log data. It is a DB format representation of the act of deleting a file in order to secure space when it is entered once.

Below is the type of expression you can use.

Index

Syntax

```
DELETE FROM table_name;
DELETE FROM table_name OLDEST number ROWS;
DELETE FROM table_name EXCEPT number ROWS;
DELETE FROM table_name EXCEPT number [YEAR | MONTH | WEEK | DAY | HOUR | MINUTE | SECOND];
DELETE FROM table_name BEFORE datetime_expr;
```

Example

```
-- Delete all data.
mach>DELETE FROM devices;
10 row(s) deleted.

-- Delete oldest 5.
mach>DELETE FROM devices OLDEST 5 ROWS;
10 row(s) deleted.

-- Delete all except last 5.
mach>DELETE FROM devices EXCEPT 5 ROWS;
15 row(s) deleted.

-- Delete all data from before June 1, 2018.
mach>DELETE FROM devices BEFORE TO_DATE('2018-06-01', 'YYYY-MM-DD');
50 row(s) deleted.
```

Index for log table

Two index types can be created in the Machbase log table.

For more information, refer to the DDL page CREATE INDEX section of the SQL Reference .

- LSM Index: The LSM index can be created in all columns except Text and Binary types.
- KEYWORD Index: Used to search strings as it can be generated only for Varchar and Text column.

Index

Create Index

Create an index on a specific column using the CREATE INDEX statement.

```
CREATE INDEX index_name ON table_name (column_name) [index_type] [tablespace] [index_prop_list]
    index_type ::= INDEX_TYPE { LSM | KEYWORD }
    tablespace ::= TABLESPACE tablesapce_name
    index_prop_list ::= value_pair, value_pair, ...
    value_pair ::= property_name = property_value
```

```
Mach> CREATE INDEX id_index ON log_data(id) INDEX_TYPE LSM TABLESPACE tbs_data MAX_LEVEL=3;
Created successfully.
```

Change Index

Change the index attribute using the ALTER INDEX statement.

```
ALTER INDEX index_name SET KEY_COMPRESS = { 0 | 1 }
```

```
Mach> ALTER INDEX id_index SET KEY_COMPRESS = 1;
```

Delete Index

Delete the specified index using the DROP INDEX statement. However, if there is another session in which the table is being searched, it will fail with an error.

```
DROP INDEX index_name;
```

```
Mach> DROP INDEX id_index;
Dropped successfully.
```

Example of Log Table

Installing the Machbase package provides a tutorial that creates a log table, populates the generated table with the log data, and displays the log data.

You can find it in the path below.

```
[machbase@localhost tutorials]$ cd $MACHBASE_HOME/tutorials
[machbase@localhost tutorials]$ ls -l
total 0
drwxrwxr-x 2 machbase machbase 103 Oct 30 16:10 backup_mount
drwxrwxr-x 2 machbase machbase  44 Oct 30 16:10 connect_r
drwxrwxr-x 2 machbase machbase 177 Oct 30 16:10 csvload
drwxrwxr-x 2 machbase machbase  49 Oct 30 16:10 export_data
drwxrwxr-x 2 machbase machbase  32 Oct 30 16:10 install_docker_image
drwxrwxr-x 2 machbase machbase  49 Oct 30 16:10 ip_address
drwxrwxr-x 2 machbase machbase  75 Oct 30 16:10 searchtext
drwxrwxr-x 2 machbase machbase  93 Oct 30 16:10 time_series
[machbase@localhost tutorials]$
```

Index

Create Log Table

The log data to be input is a file in the following csv format.

```
[machbase@localhost csvload]$ cd $MACHBASE_HOME/tutorials/csvload
[machbase@localhost csvload]$ more sample_data.csv
2015-05-20 06:00:00,63.214.191.124,2296,122.195.164.32,5416,12,GET /twiki/bin/view/Main/TWikiGroups?rev=1.2 HTTP/1
2015-05-20 06:00:07,212.237.153.79,6203,71.129.68.118,8859,67,GET /twiki/bin/view/Main/WebChanges HTTP/1.1,200,405
2015-05-20 06:00:07,243.9.49.80,344,122.195.164.32,6203,46,GET /twiki/bin/view/Main/TWikiGroups?rev=1.2 HTTP/1.1,2
2015-05-20 06:00:07,232.191.241.129,5377,174.47.129.59,1247,17,GET /mailman/listinfo/hsdivision HTTP/1.1,200,6291
2015-05-20 06:00:07,121.67.24.216,2296,212.237.153.79,6889,68,GET /twiki/bin/view/TWiki/WebTopicEditTemplate HTTP/1
2015-05-20 06:00:07,31.224.72.52,450,100.46.183.122,10541,20,GET /twiki/bin/view/Main/WebChanges HTTP/1.1,200,4052
2015-05-20 06:00:07,210.174.159.227,6180,173.149.119.202,6927,2,GET /twiki/bin/rdiff/TWiki/AlWilliams?rev1=1.2&rev
2015-05-20 06:00:07,210.174.159.227,10124,16.194.51.72,10512,69,GET /twiki/bin/rdiff/TWiki/AlWilliams?rev1=1.2&rev
2015-05-20 06:00:07,60.48.99.15,12333,85.183.139.166,12020,64,GET /robots.txt HTTP/1.1,200,68
```

Check each field value of log data and create a table. You can create it in machsql using 'CREATE TABLE' syntax.

```
CREATE TABLE SAMPLE_TABLE
(
    srcip        IPV4,
    srcport      INTEGER,
    dstip        IPV4,
    dstport      INTEGER,
    protocol     SHORT,
    eventlog     VARCHAR(1204),
    eventcode    SHORT,
    eventsize    LONG
);
```

Alternatively, you can create a table creation script file and run machsql on the OS command line.

```
[machbase@localhost csvload]$ machsql -s localhost -u sys -p manager -f create_sample_table.sql
=================================================================
     Machbase Client Query Utility
     Release Version x.x.x.official
     Copyright 2014 MACHBASE Corporation or its subsidiaries.
     All Rights Reserved.
=================================================================
MACHBASE_CONNECT_MODE=INET, PORT=5656
Type 'help' to display a list of available commands.
Mach> CREATE TABLE SAMPLE_TABLE
```

```
(
    srcip        IPV4,
    srcport      INTEGER,
    dstip        IPV4,
    dstport      INTEGER,
    protocol     SHORT,
    eventlog     VARCHAR(1204),
    eventcode    SHORT,
    eventsize    LONG
);
Created successfully.
```

Log Data Insert

Since the log data is a csv format file, you can load it using csvimport.

The first field in the log file is the date, which specifies the option to enter this value into the _arrival_time column.

```
[machbase@localhost csvload]$ csvimport -t sample_table -d sample_data.csv -a -F "_arrival_time YYYY-MM-DD HH24:MI
----------------------------------------------------------------
     Machbase Data Import/Export Utility.
     Release Version x.x.x.official
     Copyright 2014, MACHBASE Corporation or its subsidiaries.
     All Rights Reserved.
----------------------------------------------------------------
NLS            : US7ASCII            EXECUTE MODE   : IMPORT
TARGET TABLE   : sample_table        DATA FILE      : sample_data.csv
IMPORT_MODE    : APPEND              FILED TERM     : ,
ROW TERM       :
                   ENCLOSURE      : "
ESCAPE         : "                   ARRIVAL_TIME   : TRUE
ENCODING       : NONE                HEADER         : FALSE
CREATE TABLE   : FALSE

 Progress bar                       Imported records        Error records
                                            1000000                    0

Import time         :  0 hour  0 min  5.728 sec
Load success count  : 1000000
Load fail count     : 0

[machbase@localhost csvload]$
```

Log Data Retrieval

Check the data in machsql.

```
[machbase@localhost csvload]$ machsql
================================================================
     Machbase Client Query Utility
     Release Version x.x.x.official
     Copyright 2014 MACHBASE Corporation or its subsidiaries.
     All Rights Reserved.
================================================================
Machbase server address (Default:127.0.0.1) :
Machbase user ID  (Default:SYS)
Machbase User Password :
MACHBASE_CONNECT_MODE=INET, PORT=5656
Type 'help' to display a list of available commands.
Mach> show tables;
NAME                                                                                    TYPE
-----------------------------------------------------------------------------------------------------
SAMPLE_TABLE                                                                            LOG
[1] row(s) selected.
```

```
Mach> desc sample_table;
[ COLUMN ]
---------------------------------------------------------------
NAME                          TYPE               LENGTH
---------------------------------------------------------------
SRCIP                         ipv4               15
SRCPORT                       integer            11
DSTIP                         ipv4               15
DSTPORT                       integer            11
PROTOCOL                      short              6
EVENTLOG                      varchar            1204
EVENTCODE                     short              6
EVENTSIZE                     long               20

Mach> SELECT COUNT(*) FROM SAMPLE_TABLE;
COUNT(*)
-----------------------
1000000
[1] row(s) selected.

Mach> SELECT SRCIP, COUNT(*) FROM SAMPLE_TABLE GROUP BY SRCIP ORDER BY 2 DESC LIMIT 10;
SRCIP           COUNT(*)
---------------------------------------
96.128.212.177  13594
173.149.119.202 13546
219.229.142.218 13537
69.99.246.62    13511
239.81.105.222  13501
86.45.186.17    13487
231.146.69.51   13483
248.168.229.34  13472
105.9.103.49    13472
115.18.128.171  13468
[10] row(s) selected.
Mach>
```

Create and View Index

Create a keyword index for the eventlog column of varchar type in the generated sample_table column and search for text.

```
-- Create eventlog_index index.
Mach> CREATE INDEX eventlog_index ON SAMPLE_TABLE( eventlog) INDEX_TYPE KEYWORD;
Created successfully.
Elapsed time: 0.442

-- Check created index.
Mach> desc sample_table;
[ COLUMN ]
---------------------------------------------------------------
NAME                          TYPE               LENGTH
---------------------------------------------------------------
SRCIP                         ipv4               15
SRCPORT                       integer            11
DSTIP                         ipv4               15
DSTPORT                       integer            11
PROTOCOL                      short              6
EVENTLOG                      varchar            1204
EVENTCODE                     short              6
EVENTSIZE                     long               20

[ INDEX ]
---------------------------------------------------------------
NAME                          TYPE               COLUMN
---------------------------------------------------------------
EVENTLOG_INDEX                KEYWORD_LSM        EVENTLOG
```

```
-- Retrieve data containing 'view' using SEARCH syntax.
Mach> SELECT EVENTLOG FROM SAMPLE_TABLE WHERE EVENTLOG SEARCH 'view' LIMIT 10;
EVENTLOG
--------------------------------------------------------------------------------
GET /twiki/bin/view/TWiki/ManagingWebs?skin=print HTTP/1.1
GET /twiki/bin/view/Main/TokyoOffice HTTP/1.1
GET /twiki/bin/view/TWiki/ManagingWebs?rev=1.22 HTTP/1.1
GET /twiki/bin/view/Main/DCCAndPostFix HTTP/1.1
GET /twiki/bin/view/TWiki/WebTopicEditTemplate HTTP/1.1
GET /twiki/bin/view/Main/TokyoOffice HTTP/1.1
GET /twiki/bin/view/TWiki/WikiCulture HTTP/1.1
GET /twiki/bin/view/Main/MikeMannix HTTP/1.1
GET /twiki/bin/view/TWiki/WikiCulture HTTP/1.1
GET /twiki/bin/view/TWiki/WikiCulture HTTP/1.1
[10] row(s) selected.

-- Obtain number of data containing 'robots.txt'.
Mach> SELECT COUNT(*) FROM SAMPLE_TABLE WHERE EVENTLOG SEARCH 'robots.txt';
COUNT(*)
----------------------
40283
[1] row(s) selected.

-- Aggregate data containing 'robots.txt' by SRCIP and output only top 10.
Mach> SELECT SRCIP, COUNT(*) FROM SAMPLE_TABLE WHERE EVENTLOG SEARCH 'robots.txt' GROUP BY SRCIP ORDER BY 2 DESC LI
SRCIP           COUNT(*)
---------------------------------------
81.227.25.139   616
162.80.44.96    596
7.234.88.67     595
227.106.13.91   578
220.192.100.45  570
46.201.48.18    570
231.146.69.51   564
185.22.195.164  564
64.58.31.79     561
50.5.206.126    561
[10] row(s) selected.
```

Time Series Data Retrieval

Machbase provides a convenient syntax for querying time series data. Learn how to query fast data using DURATION.

```
-- Check maximum and minimum values entered in _arrival_time column.
Mach> SELECT MIN(_ARRIVAL_TIME), MAX(_ARRIVAL_TIME) FROM SAMPLE_TABLE;
MIN(_ARRIVAL_TIME)              MAX(_ARRIVAL_TIME)
-------------------------------------------------------------------
2015-05-20 06:00:00 000:000:000 2015-05-20 06:40:10 000:000:000
[1] row(s) selected.

-- Use DATE_TRUNC() to obtain count per minute.
Mach> SELECT DATE_TRUNC('minute', _ARRIVAL_TIME) as TIME, COUNT(*) as COUNT FROM SAMPLE_TABLE GROUP BY TIME ORDER B
TIME                            COUNT
-----------------------------------------------------
2015-05-20 06:00:00 000:000:000 32001
2015-05-20 06:01:00 000:000:000 28000
2015-05-20 06:02:00 000:000:000 24000
2015-05-20 06:03:00 000:000:000 32000
2015-05-20 06:04:00 000:000:000 16000
2015-05-20 06:05:00 000:000:000 16000
2015-05-20 06:06:00 000:000:000 32000
2015-05-20 06:07:00 000:000:000 32000
2015-05-20 06:08:00 000:000:000 20000
2015-05-20 06:09:00 000:000:000 24000
2015-05-20 06:10:00 000:000:000 20000
2015-05-20 06:11:00 000:000:000 20000
```

```
2015-05-20 06:12:00 000:000:000 24000
2015-05-20 06:13:00 000:000:000 20000
2015-05-20 06:14:00 000:000:000 32000
2015-05-20 06:15:00 000:000:000 24000
2015-05-20 06:16:00 000:000:000 32000
2015-05-20 06:17:00 000:000:000 28000
2015-05-20 06:18:00 000:000:000 32000
2015-05-20 06:19:00 000:000:000 12000
2015-05-20 06:20:00 000:000:000 24000
2015-05-20 06:21:00 000:000:000 28000
2015-05-20 06:22:00 000:000:000 28000
2015-05-20 06:23:00 000:000:000 24000
2015-05-20 06:24:00 000:000:000 28000
2015-05-20 06:25:00 000:000:000 28000
2015-05-20 06:26:00 000:000:000 32000
2015-05-20 06:27:00 000:000:000 20000
2015-05-20 06:28:00 000:000:000 20000
2015-05-20 06:29:00 000:000:000 20000
2015-05-20 06:30:00 000:000:000 28000
2015-05-20 06:31:00 000:000:000 32000
2015-05-20 06:32:00 000:000:000 32000
2015-05-20 06:33:00 000:000:000 28000
2015-05-20 06:34:00 000:000:000 20000
2015-05-20 06:35:00 000:000:000 24000
2015-05-20 06:36:00 000:000:000 24000
2015-05-20 06:37:00 000:000:000 16000
2015-05-20 06:38:00 000:000:000 24000
2015-05-20 06:39:00 000:000:000 16000
2015-05-20 06:40:00 000:000:000 3999
[41] row(s) selected.

-- Use DURATION statement to specify time range one minute before specified time reference.
Mach> SELECT MIN(_ARRIVAL_TIME), MAX(_ARRIVAL_TIME), COUNT(*) as COUNT FROM SAMPLE_TABLE DURATION 1 MINUTE BEFORE T
MIN(_ARRIVAL_TIME)              MAX(_ARRIVAL_TIME)              COUNT
-------------------------------------------------------------------------------------
2015-05-20 06:29:05 000:000:000 2015-05-20 06:29:45 000:000:000 20000
[1] row(s) selected.

-- Use DURATION syntax to specify time range after one minute from specific time reference.
Mach> SELECT MIN(_ARRIVAL_TIME), MAX(_ARRIVAL_TIME), COUNT(*) as COUNT FROM SAMPLE_TABLE DURATION 1 MINUTE AFTER TO
MIN(_ARRIVAL_TIME)              MAX(_ARRIVAL_TIME)              COUNT
-------------------------------------------------------------------------------------
2015-05-20 06:30:04 000:000:000 2015-05-20 06:30:57 000:000:000 28000
[1] row(s) selected.


-- Use DURATION statement to specify FROM to TO time range.
Mach> SELECT MIN(_ARRIVAL_TIME), MAX(_ARRIVAL_TIME), COUNT(*) as COUNT FROM SAMPLE_TABLE DURATION FROM TO_DATE('201
MIN(_ARRIVAL_TIME)              MAX(_ARRIVAL_TIME)              COUNT
-------------------------------------------------------------------------------------
2015-05-20 06:20:03 000:000:000 2015-05-20 06:29:45 000:000:000 252000
[1] row(s) selected.
```

Internet Address Type Data Retrieval

Machbase provides data types for Internet addresses and can be conveniently searched.

```
-- Set IP band in Netmask format and inquire.
Mach> SELECT COUNT(*) FROM SAMPLE_TABLE WHERE SRCIP CONTAINED '100.195.159.0/24';
COUNT(*)
-----------------------
13097
[1] row(s) selected.


-- Equal (=) search is also possible using '*'.
Mach> SELECT COUNT(*) FROM SAMPLE_TABLE WHERE SRCIP = '100.195.159.*';
```

```
COUNT(*)
------------------------
13097
[1] row(s) selected.
```

Volatile Table

Concept

The volatile table is a temporary table in which all the data resides in the temporary memory space and enriches the data through joining with the log table.

The volatile table is a supplementary information table that stores various information of a specific device or equipment represented by a simple symbol or a number in a log table. It can be input and updated at high speed, and is used in a case where the present state of the data (which does not match the time series data) needs to be maintained in real time.

The characteristics of this table are as follows

Index

Preserving Schema

The structure (schema) information of the volatile table is maintained even if the Machbase server is shut down and then restarted. To drop the table, you need to explicitly execute the DROP table.

Data Volatility

The data in the volatile table disappears as soon as the server is shut down. Therefore, when the server is started, the contents of the volatile table must be INSERTed again.

Index Providing and Join Function

It provides a RED-BLACK index, which is a real-time optimized index for fast data access of volatile tables. Therefore, it can be efficiently used in Join and searching process with log tables.

- You can specify a primary key in the table column.
- When inserting data having a duplicate primary key value, it is possible to UPDATE the value of existing data.
- The condition clause (`WHERE` clause) can be used to delete data that matches the primary key value condition.
- The `_ARRIVAL_TIME` column does not exist.

Primary Key

The primary key can be created to form a unique constraint on a table column value and to specify a key column to distinguish table data.

When inserting data into a volatile table with a primary key specified, the primary key column value of the insert data must be different from the other primary key column values in the table. This constraint is called a unique constraint.

The creation constraints of the primary key are as follows.

- Primary keys can only be created in volatile tables.
- You can specify only one primary key column, and you can not specify more than one column as primary keys.

Update Function

Unlike other table types, volatile tables provide a limited update function.

If the primary key value of the data to be inserted overlaps with one of the primary key values of the other data, the mode is changed to the 'update' mode instead of 'insertion', and another column value of the existing data having the duplicated key value is inserted, it is changed to the column value of the data to be processed. The update function can be used only in the volatile table with the primary key specified. If the primary key value is not specified during the insertion, the update function can not be used.

Delete Function

Volatile tables provide the ability to delete specific data using primary key values.

If you add a condition clause (`WHERE` clause) in the DELETE clause to specify a primary key value, you can delete it only if there is data corresponding to that primary key value. The delete function can be used only on the volatile table with the primary key specified, and the condition `(Primary Key Column) = (Value)` that can be entered in the condition clause is limited.

- Creating and Managing Volatile Table
- Volatile Data Extraction
- Inserting and Updating Volatile Data
- Deleting Volatile Data
- Creating and Managing Volatile Index
- Volatile Table Utilization Example

Creating and Managing Volatile Table

The creation and deletion methods of volatile tables are as follows.

Create

```
create volatile table vtable (id1 integer, name varchar(20));
```

Delete

```
drop table vtable;
```

Volatile Data Extraction

Data Retrieval

As with other table types, data retrieval can be performed as follows.

```
Mach> create volatile table vtable (id integer primary key, name varchar(20));
Created successfully.
Mach> insert into vtable values(1, 'west device');
1 row(s) inserted.
Mach> insert into vtable values(2, 'east device');
1 row(s) inserted.
Mach> insert into vtable values(3, 'north device');
1 row(s) inserted.
Mach> insert into vtable values(4, 'south device');
1 row(s) inserted.
Mach> select * from vtable;
ID          NAME
-------------------------------------
1           west device
2           east device
3           north device
4           south device
[4] row(s) selected.
Mach> select * from vtable where id = 1;
ID          NAME
-------------------------------------
1           west device
[1] row(s) selected.
Mach> select * from vtable where name like 'west%';
ID          NAME
-------------------------------------
1           west device
[1] row(s) selected.
```

Inserting and Updating Volatile Data

Data Insert

The data insert of the volatile table is as follows.

```
Mach> create volatile table vtable (id integer, name varchar(20));
Created successfully.
Mach> insert into vtable values(1, 'west device');
1 row(s) inserted.
Mach> insert into vtable values(2, 'east device');
1 row(s) inserted.
Mach> insert into vtable values(3, 'north device');
1 row(s) inserted.
Mach> insert into vtable values(4, 'south device');
1 row(s) inserted.
```

Data Update

When inputting data in a volatile table, data with duplicate primary key values can be updated using the ON DUPLICATE KEY UPDATE clause.

Update Data Value to be Inserted

If the INSERT statement specifies data to be inserted, but there is other data that matches the primary key value of the insert data, the INSERT statement fails and the corresponding data is not inserted. If there is another data that matches the primary key value of the insertion data, and if you wish to update the corresponding data instead of insertion, a `ON DUPLICATE KEY UPDATE` clause can be added.

- If there is no duplicate primary key data, the contents of the data to be inserted are inserted as is.
- If there is duplicate primary key data, the existing data is updated with the contents of the data to be inserted.

The constraints for using this function are as follows.

- The primary key must be specified in the volatile table.
- The value to be inserted must include the primary key value.

```
Mach> create volatile table vtable (id integer primary key, direction varchar(10), refcnt integer);
Created successfully.
Mach> insert into vtable values(1, 'west', 0);
1 row(s) inserted.
Mach> insert into vtable values(2, 'east', 0);
1 row(s) inserted.
Mach> select * from vtable;
ID          DIRECTION   REFCNT
----------------------------------------
1           west       0
2           east        0
[2] row(s) selected.

Mach> insert into vtable values(1, 'south', 0);
[ERR-01418 : The key already exists in the unique index.]
Mach> insert into vtable values(1, 'south', 0) on duplicate key update;
1 row(s) inserted.

Mach> select * from vtable;
ID          DIRECTION   REFCNT
----------------------------------------
1           south        0
2           east        0
[2] row(s) selected.
```

Specify Data Value to be Updated

Similar to above, but if you need to update to a different column value than the data value to be inserted, it can be specified through the `ON DUPLICATE KEY UPDATE SET` clause. The data value to be updated can be specified under the `SET` clause.

- If the primary key duplication data does not exist, the contents of the *embedded data* are inserted as it is.
- If there is duplicate primary key data, the existing data is updated only with the *update data* specified in the `SET` clause.

- The primary key value can not be specified as the data value to be updated.
- The values of the columns not specified in the SET clause are not updated.

```
Mach> create volatile table vtable (id integer primary key, direction varchar(10), refcnt integer);
Created successfully.
Mach> insert into vtable values(1, 'west', 0);
1 row(s) inserted.
Mach> insert into vtable values(2, 'east', 0);
1 row(s) inserted.
Mach> select * from vtable;
ID          DIRECTION   REFCNT
----------------------------------------
1           west        0
2           east        0
[2] row(s) selected.

Mach> insert into vtable values(1, 'west', 0) on duplicate key update set refcnt = 1;
1 row(s) inserted.

Mach> select * from vtable;
ID          DIRECTION   REFCNT
----------------------------------------
1           west        1
2           east        0
[2] row(s) selected.
```

Deleting Volatile Data

Delete Data

Volatile tables can delete data using the primary key value condition in the condition clause (WHERE clause).

- **The primary key column must be specified in the volatile table.**
- Only the (Primary key column) = (value) condition is allowed, and can not be used with other conditions.
- You can not use a column other than the primary key column.

```
Mach> create volatile table vtable (id integer primary key, name varchar(20));
Created successfully.
Mach> insert into vtable values(1, 'west device');
1 row(s) inserted.
Mach> insert into vtable values(2, 'east device');
1 row(s) inserted.
Mach> insert into vtable values(3, 'north device');
1 row(s) inserted.
Mach> insert into vtable values(4, 'south device');
1 row(s) inserted.
Mach> select * from vtable;
ID          NAME
-------------------------------------
1           west device
2           east device
3           north device
4           south device
[4] row(s) inserted.
Mach> delete from vtable where id = 2;
[1] row(s) deleted.
Mach> select * from vtable;
ID          NAME
-------------------------------------
1           west device
3           north device
4           south device
[3] row(s) selected.
```

Creating and Managing Volatile Index

Create and Use Index

The volatile table provides a RED-BLACK Tree optimized for real-time search. Indexes can be set for all data types. However, one index can be created for one column, and no composite index is provided.

```
Mach> create volatile table vtable (id integer, name varchar(10));
Created successfully.
Mach> create index idx_vrb on vtable (name) index_type redblack;
Created successfuly.
Mach> desc vtable;
-----------------------------------------------------------------
NAME                          TYPE               LENGTH
-----------------------------------------------------------------
ID                            integer            11
NAME                          varchar            10

[ INDEX ]
-----------------------------------------------------------------
NAME                          TYPE               COLUMN
-----------------------------------------------------------------
IDX_VRB                       REDBLACK           NAME
iFlux>
```

Primary Key Index

When a primary key is assigned to a specific column of a volatile table, a RED-BLACK Tree index is automatically generated. In this case, a special index with a Uniqueness attribute is created and does not allow duplicate values.

```
Mach> create volatile table vtable (id integer primary key, name varchar(20));
Created successfully.
Mach> desc vtable;
-----------------------------------------------------------------
NAME                          TYPE               LENGTH
-----------------------------------------------------------------
ID                            integer            11
NAME                          varchar            20

[ INDEX ]
-----------------------------------------------------------------
NAME                          TYPE               COLUMN
-----------------------------------------------------------------
__PK_IDX_VTABLE               REDBLACK           ID

iFlux>
```

Other Index Types

The bitmap or keyword index used in the log table can not be used in a volatile table.

```
Mach> create bitmap   index idx_1237 on vtable(id1);
[ERR-02069 : Error in index for invalid table. BITMAP Index can only be created for LOG Table.]
Mach> create keyword  index idx_1238 on vtable(name);
[ERR-02069 : Error in index for invalid table. KEYWORD Index can only be created for LOG Table.]
```

Volatile Table Utilization Example

Save Sensor Data Current Value

The data of the volatile table exist only in the memory, and the update operation by the primary key is very fast. Using this feature, it creates a table that stores the current values of the sensors that change very quickly. An example of a table creation script is shown below.

```
create volatile table sensor_current (sensor_id varchar(40) primary key, value double);
```

Input and Update Volatile Data

Since the table has been created, the current value of the sensor can be reflected through data input and update operations. The input sensor value is determined based on the primary key sensor_id column as to whether to perform input or update. Input or update can be performed with the following query.

```
insert into sensor_current values('SENSOR_001',100.0) on duplicate key update set value=100.0;
```

The data input in the above query statement updates the value column value of the record with the sensor_id value 'SENSOR_001', which is the column corresponding to the primary key, to 100.0. If there is no data, insert a new record according to the syntax of the insert statement.

Volatile Data Search

To find the current value of specific sensor data, search using the following query. You can perform searches using the same syntax as a regular SQL query.

```
SELECT value FROM sensor_current WHERE sensor_id = 'SENSOR_001'
```

Lookup Table

Concept

Like the volatile table, the lookup table resides in the memory, so that it can perform fast query processing.

In addition, data input and change are reflected on disk to ensure data permanence. Compared with the volatile table, the query processing performance is the same, but the data input and change performance is somewhat lower.

The characteristics of this table are as follows.

Index

Low Input / Update Performance

Unlike volatile tables, the performance of input and update due to the maintenance of disk data images is low, making them unsuitable for tables for real-time data representation such as dashboards.

Schema Preservation

Again, unlike volatile tables, the structure (schema) information in the lookup table is retained even after the server is restarted. You must explicitly use `DROP TABLE` to drop the table .

Data Preservation

Unlike the volatile table, the lookup table is restored to its original state when the server is restarted.Providing Index

Like the volatile table, it provides a RED-BLACK index. Therefore, it can be efficiently used in the search process or the join process with the log table.

Creating and Managing Lookup Table

The method of creating the reference table is as follows.

Create

```
CREATE LOOKUP TABLE lktable (id INTEGER, name VARCHAR(20));
```

Delete

```
DROP TABLE lktable;
```

Lookup Data Insert

It is the same as the input and update method of the volatile table.

Lookup Data Extraction

The data is extracted using SQL statement and the usage method is the same as the volatile table.

Lookup Data Deletion

It is the same as the volatile table.

Creating and Managing Lookup Index

As with volatile tables, RED-BLACK indexes are provided as standard, and their usage is the same as for volatile tables. It is possible to create RED-BLACK index like volatile table, and it is impossible to generate keyword and LSM index.

Lookup Table Utilization Example

Create Lookup Table

The lookup table can be updated and is used to add data that is not in the original log data through a join. The following example shows an example of creating a log table and a lookup table.

```
-- Create log table.
create table weblog (addr ipv4, msg varchar(100));
-- Input sample data.
insert into weblog values ('127.0.0.1', 'a test msessage');
-- Create lookup table.
create lookup table dnslookup (addr ipv4 primary key, hostname varchar (100));
```

Let's insert or update the data in the lookup table.

```
insert into dnslookup values ('127.0.0.1', 'localhost') on duplicate key update set hostname = '127.0.0.1'
```

You can retrieve data from lookup tables and log tables through join.

```
select msg, hostname from weblog, dnslookup where weblog.addr = dnslookup.addr;
```

STREAM

Concept

Since Machbase 5, STREAM is a newly supported real-time data processing function based on Continuous Query Language (CQL).
In other words, it is possible to extract the data satisfying a condition of incremental data inputted to the log table and to load it in real-time into another table. If a conditional search is performed on all the data without using the stream function, the retrieval of the accumulated data is not only slow but also may be a heavy burden imposed on the system.
Stream can be used to retrieve conditions for specific log data entered in real time and respond quickly to events.

Restrictions

- Currently, Machbase (version 5.5) supports STREAM only in Edge Edition and Fog Edition.
    - The Cluster Edition will be support STREAM in a future versions.
- The data input source is only available with the log table.
    - The ability to use the tag table as a source will be supported in future versions.
- The data output destination is available for log and tag tables.

Creating and Deleting Stream

Create Stream

The stream query can only be generated in the form of Insert... Select. When generating the stream, the query is checked to see if it is a query that can be executed normally.
Use the following stored procedure to create the stream.

Even if the stream is successfully created, execution will not start immediately. For more information, refer to Stream Startup and Shutdown.

```
EXEC STREAM_CREATE(stream_name, stream_query_string);
```

The basic stream query is executed on every input data. In this case, statistical queries such as SUM and AVG cannot be used.

```
EXEC STREAM_CREATE(normal_query, 'INSERT INTO CEP_LOG_TABLE SELECT * FROM EVENT WHERE C1 = 0');
```

However, if the period at which the stream query is to be executed is set at the end of the insert select statement, the statistical query for the input data can be used at regular intervals.

```
EXEC STREAM_CREATE(aggr_1_sec, 'insert into aggr select sum(i1), i2 from base group by i2 BY 1 SECOND');
```

The above stream query executes a group by query on the latest data entered after the last execution every second and inputs the result to the "aggr" log table.

If the user wants to define the execution time of the stream query, specify the following in the execution cycle setting section.

The stream query is not executed before the explicit call of the user.

```
EXEC STREAM_CREATE(base_trig, 'insert into aggr select sum(i1), i2 from base group by i2 BY USER');
```

If the condition for executing a stream query is BY USER, it will not be executed until the stream query is explicitly called using the STREAM_EXECUTE procedure. This called with STREAM_EXECUTE, executes a stream query only for incremental data added during execution, except for those previously read.

Delete Stream

The list of generated streams can be retrieved using the V$STREAMS meta table. To delete a stream, use the following stored procedure with the name of the stream that you determined when you created the stream as a parameter.

```
EXEC STREAM_DROP(stream_name);
```

A running stream can not be deleted. The stream must first be shut down before deleting the stream. For more information, refer to Stream Startup and Shutdown.

Stream Startup and Shutdown

Stream Startup

Executes the registered stream using the stored procedure. Once executed, the stream is continuously executed. Even if the server is restarted, the continuous stream query is executed for the data inputted after the last execution.

```
EXEC STREAM_START(stream_name);
```

Direct execution of the stream

If the stream execution condition is set to BY USER, the query will not be executed without explicit call by the user. The following stored procedure is used to execute this stream query.

```
EXEC STREAM_EXECUTE(stream_name);
```

When creating a stream query to be called, an error occurs if the stream is not created as a BY USER condition or if the stream has not been executed with STREAM_START.

Stream Shutdown

Use the following stored procedure to shut down a running stream.

```
EXEC STREAM_STOP(stream_name);
```

Backup and Mount

It is very important to backup your data on a regular basis. This chapter describes how to back up Machbase data and restore backed up data. Machbase also provides a mount function that allows you to retrieve backed up data without restoring it. The mount function can be executed very quickly when the backed up data needs to be read.

Backup Overview

BACKUP/MOUNT

To ensure the permanence of the database, the data stored in memory is stored on the disk as soon as possible. In case of a general failure such as Process Failure, Restart Recovery makes the database consistent. However, in case of power failure or hardware damage caused by fire, database recovery is impossible. In order to solve this problem, the database backup and recovery function saves data to another disk or hardware periodically in another area and recovers the data using the corresponding data in case of an emergency.

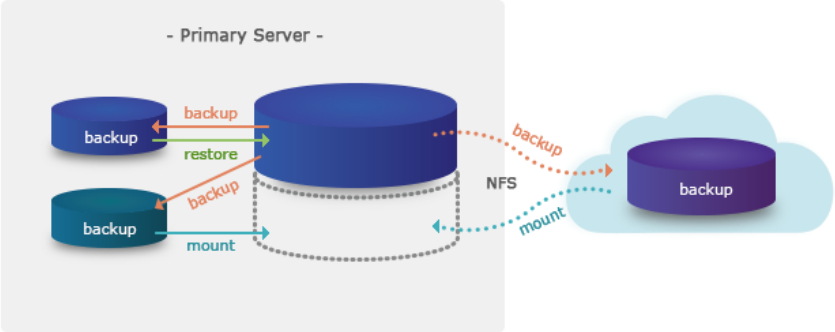
Database backups are divided into two types depending on when they are performed.

- Offline Backup
- Online Backup

First, the Offline Backup function is called Cold Backup as it shuts down the DBMS and copies the database. It is very simple, but it has the disadvantage of the user's service being interrupted. Therefore, it is rarely used during operation and tends to be used only for initial testing or data construction.

Second, Online Backup is called Hot Backup as a function to backup the database when DBMS is running. This function can be performed without interrupting the service, increasing the user's service availability. Most DBMS Backup refers to Online Backup. Unlike other database backups, Machbase, a time-series database, provides Duration Backup. This allows you to specify the time of the database to be backed up at the time of backup so that only the data at the desired time will be backed up.

Index




```
backup database into disk = 'backup';
backup database from to_date('2015-07-14 00:00:00','YYYY-MM-DD HH24:MI:SS') to to_date('2015-07-14 23:59:59 999:999
               into disk = 'backup_20150714';
```

The backed-up database can be used as an existing database through a recovery process. This recovery method is called Restore. This Restore function deletes the damaged database and restores the backed up database image to the Primary Database. Therefore, when recovering, delete the existing database and restore it using machadmin -r.

```
machadmin -r 'backup'
```

The mount / unmount function is an online function that attaches a backed up database to the currently running database.

```
mount database 'backup' to mountName;
umount database mountName;
```

Database Backup

Machbase offers two options for backing up your data. It provides DATABASE backup function which backs up information of running DB and TABLE backup function which can select only necessary tables to back up.
The backup command provided by DB is as follows.

```
BACKUP [ DATABASE | TABLE table_name ]  [ time_duration ] INTO [ DISK | IBFILE ] = 'path/backup_name';
time_duration = FROM start_time TO end_time
path = 'absolute_path' or  'relative_path'
```

```
# Directory backup
       BACKUP DATABASE INTO DISK = 'backup_dir_name';
# a single file backup
       BACKUP DATABASE INTO IBFILE = backup_ibfile_name;
# Set backup duration
      - Directory backup
       BACKUP DATABASE FROM TO_DATE('2015-07-14 00:00:00','YYYY-MM-DD HH24:MI:SS')
                         TO TO_DATE('2015-07-14 23:59:59','YYYY-MM-DD HH24:MI:SS')
                         INTO DISK = '/home/machbase/backup_20150714'
     - File backup
       BACKUP DATABASE FROM TO_DATE('2015-07-14 00:00:00','YYYY-MM-DD HH24:MI:SS')
                         TO TO_DATE('2015-07-14 23:59:59','YYYY-MM-DD HH24:MI:SS')
                         INTO IBFILE = '/home/machbase/backup_20150714.ibf'
# a single table backup
      BACKUP TABLE SYSLOG_TABLE INTO IBFILE = '/home/machbase/backup/syslog_table';
```

When performing a DB backup, the backup type, time duration, and path must be entered as options. When backing up DATABASE entirely, type DATABASE for the backup type. To backup only a specific table, enter TABLE, and then enter the name of the table to be backed up. The TIME_DURATION statement can be set to back up only the data for the required period. In the FROM field, enter the start time of the date you want to back up, and enter the time of the last date in the TO field. In Example 3, the TIME_DURATION is set to FROM, "July 14, 2014, 0, 0, 0", and TO "July 14, 2015, 23:59:59" meaning only 14 days of data are set to be backed up. If information is not entered for the DURATION item, the FROM item is set to 'January 1, 1970, 9:00:00,' and the TO item is automatically set to the time to execute the command.

Finally, a storage medium to store needs to be configured to store the results of the backup. If you want to create a backup in a single file, set the creation type to IBFILE, or enter DISK to create it in directory units. Note that you can specify the PATH information to store the product. If you enter a relative path, it will be created in the path specified in the DB_PATH item of the current DB configuration. If you want to store it somewhere other than DB_PATH, you must enter an absolute path starting with '/'.

Database Restore

The Database Restore feature is not provided as a syntax, and can be recovered offline using machadmin -r. You must check the following before restoration.

- Has Machbase been shutdown?
- Has the previously created DB been deleted?

```
machadmin -r backup_database_path;
```

```
backup database into disk = '/home/machbase/backup';
```

```
machadmin -k
machadmin -d
machadmin -r /home/machbase/backup;
```

Extract Single File

A single backup file created in the form of INTO IBFILE can not be restored immediately, and must be converted to a directory using machadmin.
After the conversion, files in the form of directory backup are created in the path of target_path/backup_file_name.

```
machadmin -x single_file_backup_file_name extract_target_path
```

```
machadmin -x backup_20150101.ibf /db/data/
machadmin -r /db/data/backup_20150101.ibf/
```

Single Backup File Information Retrieval

Information about a single backup file created by the backup command can be retrieved with the machadmin -w command.

```
machadmin -w single_backup_file_name
```

```
machadmin -w ib_backup01
...
Display information of Backup Image successfully.
-----------------------------------------------------------------
```

```
     File name        - mach_backup_19700101090000_20150725142017_3
     Create time      - 2015-07-25 14:19:22
     Data duration    - 1970-01-01 09:00:00 ~ 2015-07-25 14:20:17
     Backup duration  - 2015-07-25 14:20:18 ~ 2015-07-25 14:20:18
     Version          - DB(4.0) Meta(1.6) CM(1.5)
------------------------------------------------------------
```

Database Mount

The following problems arise when periodically backing up a large number of databases and adding data continuously in preparation for a system failure.

- Increased disk cost to store data
- Limitations of the physical disk space of the running machine

In order to solve this problem, periodical deletion is performed by leaving only data necessary for the current service. However, if you need to refer to the past data, you need to restore the backed up database. In case of a very large backup image, recovery time is long and additional equipment is needed. This is because the Restore function can only be performed by deleting the currently running database. To solve this problem, Machbase provides the Database Mount function.

The Database Mount function is an online function that attaches a backed up database to the currently running database. By attaching multiple backup databases to the primary database, the user can refer to multiple backup databases as if they were one database. The mounted database is read-only.

The Mount DATABASE command is a function that prepares the database or table DATA created by Backup in a state that it can be viewed from the currently running database. So, Mounted DATABASE can query the data using the same DB command.

The current Database Mount function restrictions are as follows.

- The backup information must be compatible with the database to be mounted, the DB major number, and the Meta major number.
- When mounting backup data, it is read-only and does not support index creation, data insertion or deletion.
- Information about the currently mounted DATABASE can be found by querying V$STORAGE_MOUNT_DATABASES.

Mount

To execute the mount command, Backup_database_path information and DatabaseName are required. Backup_database_path is the location information of the DB created by Backup command. DatabaseName is the name that can be distinguished when mounting to Database. Backup_database_path is searched based on the directory specified in the DB_PATH set in the environment variable of the DB when the relative path is entered in the same way as when performing the backup.

```
MOUNT DATABASE 'backup_database_path' TO mount_name;
```

```
MOUNT DATABASE '/home/machbase/backup' TO mountdb;
```

Unmount

If the mounted database will no longer be used, it can be removed using the unmount command.

```
UNMOUNT DATABASE mount_name;
```

```
UNMOUNT DATABASE mountdb;
```

MOUNT DB Data Retrieval

When querying DATA of Backup DB, it can be retrieved by using the same SQL statement when querying the DATA of the DB in operation.

The mounted DB can retrieve data only by the SYS admin user of the DB in operation. To retrieve the data, you must put MountDBName and UserName in front of the TableName to be queried, and use '.' for each delimiter. MountDBName is used to refer to a specific DB among currently mounted DBs, and UserName refers to the information of the user that owns the mounted DB table.

```
SELECT column_name FROM mount_name.user_name.table_name;
```

```
SELECT * FROM mountdb.sys.backuptable;
```

Database Mount

- MOUNT
- UNMOUNT
- Read data from a mounted database

Continuously storing a large amount of data in order to analyze the data greatly increases its volume, resulting in the following problems:

- Increased disk cost by storing a large volume of data
- Running into equipment disk limits for data analysis

To solve the problem, it is necessary to back up old data and periodically delete it. If you need to read old data later and you perform data recovery to read the backed up database, it will not only take a long time to recover, but it will also take a long time to convert the database to offline, There is a problem that requires separate equipment to continue the service. Machase supports the MOUNT command to solve this problem.

The MOUNT command reads the backed up data while the database is in service and creates a new database independent of the currently running database. One server can add multiple backed-up databases to retrieve data at the same time, but the mounted database is read-only and can not add or delete data.

The Database MOUNT command allows you to read both the backup data and the main database contents simultaneously. Therefore, the mounted database can retrieve data in the same way as the existing data retrieval method.

To execute the MOUNT instruction, the following conditions must be met.

- The backup database version and the metadata version must be compatible.
- You can not create tables, create/delete indexes, or add/delete data to a mounted backup database.

Information about mounted databases can be obtained from the V$STORAGE_MOUNT_DATABASES meta table.

Index

MOUNT

To run the mount command, the backup database pathname and the name of the database to be mounted must be entered.
The backup database path sets the location of the directory executed by the backup command. The name of the database to be mounted must be given a separate name to distinguish it from the active database.
The backup database pathname can be an absolute pathname (a pathname beginning with the "/" character), or a relative pathname based on $MACHBASE_HOME/dbs with the same rules as the backup command.

Syntax:

```
MOUNT DATABASE 'backup_database_path' TO mount_name;
```

Example:

```
MOUNT DATABASE '/home/machbase/backup' TO mountdb;
```

UNMOUNT

If the mounded database data no longer needs to be read, use the UNMOUNT command to release the mounted state.

Syntax:

```
UNMOUNT DATABASE mount_name;
```

Example:

```
UNMOUNT DATABASE mountdb;
```

Reading Data from Mounted Database

When retrieving data from a mounted database, use the same SQL statement as before.
Only the SYS user can read the mounted data. To specify the mounted database table in an SQL statement, the mount_name and user_name must be set connected to a "." character.

Syntax:

```
SELECT column_name FROM mount_name.user_name.table_name;
```

Example:

```
SELECT * FROM mountdb.sys.backuptable;
```

Backup and Recovery

Index

Database Backup

There are two methods of Machbase backup:
1. Full backup of current DB
2. Select and backup only specific tables

Syntax:

```
BACKUP [ DATABASE | TABLE table_name ]  [ time_duration ] INTO [ DISK | IBFILE
time_duration = FROM start_time TO end_time
path = 'absolute_path' or  'relative_path'
```

Example:

```
# Directory backup
       BACKUP DATABASE INTO DISK = 'backup_dir_name';
# a single file backup
       BACKUP DATABASE INTO IBFILE = backup_ibfile_name;
# Set backup duration
      - Directory backup
       BACKUP DATABASE FROM TO_DATE('2015-07-14 00:00:00','YYYY-MM-DD HH24:MI:SS')
                        TO TO_DATE('2015-07-14 23:59:59','YYYY-MM-DD HH24:MI:SS')
                        INTO DISK = '/home/machbase/backup_20150714'
     - File backup
       BACKUP DATABASE FROM TO_DATE('2015-07-14 00:00:00','YYYY-MM-DD HH24:MI:SS')
                        TO TO_DATE('2015-07-14 23:59:59','YYYY-MM-DD HH24:MI:SS')
                        INTO IBFILE = '/home/machbase/backup_20150714.ibf'
# a single table backup
      BACKUP TABLE SYSLOG_TABLE INTO IBFILE = '/home/machbase/backup/syslog_table';
```

When executing the backup command, the backup type, duration condition, and backup destination path must be defined. To back up the entire database, you must specify "DATABASE". To back up specific tables, specify "TABLE" as the backup type. When backing up specific tables, you must specify the table name.

You can specify the backup destination using the DURATION conditional statement. Specify the start time and end time of the backup target data in the FROM and TO clauses. In the example above, "2015-07-14 00:00:00" is defined as FROM and "2015-07-14 23:59:59" is defined as TO, so the user will be able to see all data for July 14, 2015 Quot; Duration If you do not specify a time condition, "1970-01-01 00:00:00" is set to FROM and the current time at which it is executed is set in the TO clause.

Specify "IBFILE" to create a backup file as a single file, or "DISK" as a backup option to create multiple files and directories. Note that when specifying a backup path, backup files are created under "$ MACHBASE_HOME / dbs" when a relative path is specified. To specify an absolute path, you must always set a path that starts with "/".

Database Recovery

When restoring data from a backup file, it can not be executed by a query command, and the command "machadmin -r" should be executed in a situation where the database does not operate. Before running the backup, you should review for the following conditions:

- Is the Machbase database down?
- Is the current database deleted and replaced with the data to be recovered, and is the deletion of the current database allowed?

Syntax:

```
machadmin -r backup_database_path
```

Example:

```
backup database into disk = '/home/machbase/backup';

machadmin -k
machadmin -d
machadmin -r /home/machbase/backup;
```

Single Backup File Extraction

Single backup file (IBFILE) can not be used immediately for recovery. After extracting the data with the "machadmin" tool, the backup files are created in the "target_path/backup_file_name/" directory.

Syntax:

```
machadmin   -x  single_file_backup_file_name    extract_target_path
```

Example:

Single Backup File Information Confirmation

The single backup file information created by the backup command can be confirmed by machadmin -w command.

Syntax:

```
machadmin   -w  single_backup_file_name
```

Example:

```
machadmin -w ib_backup01
...
Display information of Backup Image successfully.
------------------------------------------------------------------
     File name        - mach_backup_19700101090000_20150725142017_3
     Create time      - 2015-07-25 14:19:22
     Data duration    - 1970-01-01 09:00:00 ~ 2015-07-25 14:20:17
     Backup duration  - 2015-07-25 14:20:18 ~ 2015-07-25 14:20:18
     Version          - DB(4.0) Meta(1.6) CM(1.5)
------------------------------------------------------------
```

The displayed information includes the name of the backup file, the date of creation, the start time of the backed up data, the end time, and the version of the backed-up database.

Tools

Machbase provides a variety of command-line tools, web-based management tools, and data collection tools.

- Console
- Collector
- Visualization Tool
- Cluster management tools

Console

These are the utilities used for starting, shutting down, and entering/outputting data in the Machbase server.

- MACHADMIN
- MACHLOADER
- MACHSQL
- CSVIMPORT/CSVEXPORT

MACHADMIN

machadmin is used to start up or shut down the Machbase server and to check the creation, deletion, and execution status.

Index

Option and Features

The options for machadmin are as follows. The functions described in the previous installation section are omitted.

```
mach@localhost:~$ machadmin -h
```

Option	Description
-u, --startup/ --recovery[=simple,complex,reset]	Machbase server startup/recovery mode (default: simple)
-s, --shutdown	Machbase server shuts down normally
-c, --createdb	Creates Machbase database
-d, --destroydb	Deletes Machbase database
-k, --kill	Force quits Machbase server
-i, --silence	Runs without output
-r, --restore	Recovers database from backup
-x, --extract	Converts backup files to backup directory
-w, --viewimage	Outputs backup image file information
-e, --check	Checks Machbase server run status
-t, --licinstall	Installs license file
-f, --licinfo	Outputs installed license information

Recovery Mode

Syntax:

```
machadmin -u --recovery=[simple | complex | reset]
```

The recovery mode is as follows:

- simple: If there is no power loss when the server is running, simple recovery mode is run by default.
- complex: The complex recovery mode takes longer to execute than the simple mode. It is executed by default when restarting after the power is turned off.
- reset: When recovery is not performed in simple or complex mode, all data in all tables are checked to recover the database. In this case, some loss of data may occur.

Server Normal Shutdown

Example:

```
mach@localhost:~$ machadmin -s

------------------------------------------------------------------
     Machbase Administration Tool
     Release Version - 5.1.9.community
     Copyright 2014, MACHBASE Corp. or its subsidiaries
     All Rights Reserved
------------------------------------------------------------------
Waiting for the server shut down...
Server shut down successfully.
```

Create Database

Example:

```
mach@localhost:~$ machadmin -c
------------------------------------------------------------------
     Machbase Administration Tool
     Release Version - 5.1.9.community
     Copyright 2014, MACHBASE Corp. or its subsidiaries
     All Rights Reserved
------------------------------------------------------------------
Database created successfully.
```

Delete Database

Example:

```
mach@localhost:~$ machadmin -d
------------------------------------------------------------------
     Machbase Administration Tool
     Release Version - 5.1.9.community
     Copyright 2014, MACHBASE Corp. or its subsidiaries
     All Rights Reserved
------------------------------------------------------------------
Destroy Machbase database- Are you sure?(y/N) y
Database destroyed successfully.
```

Force to abort Server

Syntax:

```
machadmin -k
```

Example:

```
mach@localhost:~$ machadmin -k
------------------------------------------------------------------
     Machbase Administration Tool
     Release Version - 5.1.9.community
     Copyright 2014, MACHBASE Corp. or its subsidiaries
     All Rights Reserved
------------------------------------------------------------------
Waiting for Machbase terminated...
Server terminated successfully.
```

Run Silent Mode

Removes the message that is output when 'machadmin' runs.

Syntax:

```
machadmin -i
```

Database Recovery

Syntax:

```
machadmin -r backup_database_path
```

Example:

```
mach@localhost:~$ machadmin -r 'backup'
----------------------------------------------------------------
     Machbase Administration Tool
     Release Version - 5.1.9.community
     Copyright 2014, MACHBASE Corp. or its subsidiaries
     All Rights Reserved
----------------------------------------------------------------
Backed up database restored successfully.
```

Convert to Backup Folder

Syntax:

```
machadmin -x backup_file extract_path
```

Outputs a single backup file (* .ibf) to the backup directory: -x backup_file extract_path.
If you back up to a single backup file, you must convert it to a directory-type backup to be able to restore the backup.

Example:

```
mach@localhost:~$ machadmin -x 'backup.ibf' 'backup'
----------------------------------------------------------------
     Machbase Administration Tool
     Release Version - 5.1.9.community
     Copyright 2014, MACHBASE Corp. or its subsidiaries
     All Rights Reserved
----------------------------------------------------------------
Extract backup image successfully.
```

Check Backup File Information

Syntax:

```
machadmin -w backup_file
```

Example:

```
mach@localhost:~$ machadmin -w 'backup'
----------------------------------------------------------------
     Machbase Administration Tool
     Release Version - 5.1.9.community
     Copyright 2014, MACHBASE Corp. or its subsidiaries
     All Rights Reserved
----------------------------------------------------------------
Display information of backup image file.
------------------------------------------------------------
     File name        - mach_backup_19700101090000_20150805092853_1
     Create time      - 2015-08-04 15:35:56
     Data duration    - 1970-01-01 09:00:00 ~ 2015-08-05 09:28:53
     Backup duration  - 2015-08-05 09:28:53 ~ 2015-08-05 09:28:53
     Version          - DB(4.0) Meta(2.0) CM(1.5)
----------------------------------------------------------------
```

Check Server Run Status

Syntax:

```
machadmin -e
```

Example output when the server is not running

```
mach@localhost:~$ machadmin -e
------------------------------------------------------------------
     Machbase Administration Tool
     Release Version - 5.1.9.community
     Copyright 2014, MACHBASE Corp. or its subsidiaries
     All Rights Reserved
------------------------------------------------------------------
[ERR] Server is not running.
```

Example output when the server is running

```
mach@localhost:~$ machadmin -e
------------------------------------------------------------------
     Machbase Administration Tool
     Release Version - 5.1.9.community
     Copyright 2014, MACHBASE Corp. or its subsidiaries
     All Rights Reserved
------------------------------------------------------------------
Machbase server is already running with PID (14098).
```

Install License File

Syntax:

```
machadmin -t license_file
```

Example:

```
mach@localhost:~$ machadmin -t license.dat
------------------------------------------------------------------
     Machbase Administration Tool
     Release Version - 5.1.9.community
     Copyright 2014, MACHBASE Corp. or its subsidiaries
     All Rights Reserved
------------------------------------------------------------------
License installed successfully.
```

License Check

Example:

```
mach@localhost:~$ machadmin -f
------------------------------------------------------------------
     Machbase Administration Tool
     Release Version - 5.1.9.community
     Copyright 2014, MACHBASE Corp. or its subsidiaries
     All Rights Reserved
------------------------------------------------------------------
                  INFORMATION
Install Date                     : 2018-12-20 11:34:43
Company#ID-ProjectName           : machbase
License Policy                   : CORE
License Type(Version 2)          : OFFICIAL
Host ID                          : FFFFFFFFFFFFFF
```

```
Issue Date                        : 2013-03-25
Expiry Date                       : 2037-03-18
Max Data Size For a Day(GB)       : 0
Percentage Of Data Addendum(%)    : 0
Overflow Action                   : 0
Overflow Count to Stop Per Month  : 0
Stop Action                       : 0
Reset Flag                        : 0
-----------------------------------------------------------------
                  STATUS
Usage Of Data(GB)                 : 0.000000
Previous Checked Date             : 2018-12-22
Violation Count                   : 0
Stop Enabled                      : 0
-----------------------------------------------------------------
License information displayed successfully.
```

MACHLOADER

Machloader

machloader is used to import/export text file data to the Machbase server. It works with CSV files by default, but it also supports other formats.

The features of machloader are as follows.

- machloader can specify a datetime type in the schema file. The datetime type specified must be of the type supported by the Machbase server. One datetime type can be applied to all fields, and each field can have a different format.
- To delete and input the input target table data, use the "-m replace" option.
- machloader does not verify the schema and data file consistency. The user must check that the schema, tables, and data files meet the consistency.
- machloader supports APPEND mode by default.
- machloader does not use the "_ARRIVAL_TIME" column by default. You must use the "-a" option to import/export the corresponding column data.

The options for machloader can be seen with the following command:

```
[mach@localhost]$ machloader -h
```

Index

Option	Description
-s, --server=SERVER	Enters Machbase server IP address (default: 127.0.0.1)
-u, --user=USER	Enters connecting user name (default: SYS)
-p, --password=PASSWORD	Connecting user password (default: MANAGER)
-P, --port=PORT	Machbase server port number (default: 5656)
-i, --import	Data import command option
-o, --export	Data export command option
-c, --schema	Command option to create schema file using the database table information
-t, --table=TABLE_NAME	Sets table name that is creating a schema file
-f, --form=SCHEMA_FORM_FILE	Specifies schema filename
-d, --data=DATA_FILE	Specifies a data file name
-l, --log=LOG_FILE	Specifies a machloader execution log file
-b, --bad=BAD_FILE	Records the data in which the input error occurred and specifies the file name that records the error description when executing -i option.
-m, --mode=MODE	Indicates import method when executing -i option. The append or replace option is available. Append enters the data after the existing data and replace deletes the existing data and enters the data.
-D, –delimiter=DELIMITER	Sets each field delimiter. The default value is ','.
-n, --newline=NEWLINE	Sets each record separator. The default is '₩n'.
-e, --enclosure=ENCLOSURE	Sets the enclosing delimiter for each field.
-r, --format=FORMAT	Specifies the format for file input/output. (default: csv)
-a, --atime	Determines whether to use the built-in column "_ARRIVAL_TIME". The default value is to not use the column.
-I, --silent	Does not display copyright-related output and import/export status information.
-h, --help	Displays a list of options.
-F, --dateformat=DATEFORMAT	Sets the column dateformat. ("_arrival_time YYYY-MM-DD HH24:MI:SS") • If you set 'unixtimestamp' instead of dateformat, the input value is regarded as the unix timestamp value. ("time_column unixtimestamp") • If you set 'nanotimestamp' instead of dateformat, the input value is regarded as a timestamp value in nanoseconds. ("time_column nanotimestamp") ⓘ The unixtimestamp and nanotimestamp formats are supported since 5.7.

Option	Description
-E, --encoding=CHARACTER_SET	Sets the encoding of input/output files. Supported encodings are UTF8 (default), ASCII, MS949, KSC5601, EUCJP, SHIFTJIS, BIG5, GB231280, and UTF16.
-C, --create	Creates a table if one does not exist upon import.
-H, --header	Sets whether header information is present upon import/export. The default value is unset.
-S, --slash	Specifies the backslash delimiter.

Basic Usage

The table must be created first before running the usage below.

CSV File Import

Imports CSV file to Machbase server.

Option:

```
-i: import specification options
-d: data file naming options
-t: table name specification option
```

Example:

```
machloader -i -d data.csv -t table_name
```

CSV File Export

Writes data to a CSV file.

Option:

```
-o: export specification options
-d: data file naming options
-t: table name specification option
```

Example:

```
machloader -o -d data.csv -t table_name
```

Use CSV File Header

The header-related setting of the CSV file.

Option:

```
-i -H: Upon import, the first line of the csv file is recognized as a header. Therefore, the first line is excluded
-o -H: Upon export, generates the csv header as the column name of the table.
```

Example:

```
machloader -i -d data.csv -t table_name -H
machloader -o -d data.csv -t table_name -H
```

Automatic Table Creation

Regards automatic table creation.

Option:

```
-C: Automatically generates the table when importing. The column names are automatically generated as c0, c1, ....
-H: Generates column names with csv header name when importing.
```

Example:

```
machloader -i -d data.csv -t table_name -C
machloader -i -d data.csv -t table_name -C -H
```

Files Not CSV Format

Sets delimiter for files that are not in CSV format.

Option:

```
-D: Delimiter option for each field
-n: Specifies each record delimiter option
-e: Specifies the enclosing character for each field.
```

Example:

```
machloader -i -d data.txt -t table_name -D '^' -n '\n' -e '"'
machloader -o -d data.txt -t table_name -D '^' -n '\n' -e '"'
```

Specify Input Mode

When importing (with -i option), there are two modes, REPLACE and APPEND. APPEND is the default. Use REPLACE mode with caution because it deletes existing data.

Option:

```
-m: Specifies import mode
```

Example:

```
machloader -i -d data.csv -t table_name -m replace
```

Specify Connection Information

Specifies server IP, user, and password separately.

Option:

```
-s: Specifies server IP address (default: 127.0.0.1)
-P: Specifies server port number (default: 5656)
-u: Specifies the connecting user name (default: SYS)
-p: Specifies the password of the connecting user (default: MANAGER)
```

Example:

```
machloader -i -s 192.168.0.10 -P 5656 -u mach -p machbase -d data.csv -t table_name
```

Create Log File

Creates the execution log file for machloader.

Option:

```
-b: Sets the name of the log file to generate the data that is not input when importing.
-l: Sets the name of the log file to generate the data and error message that were not input when importing.
```

Example:

```
machloader -i -d data.csv -t table_name -b table_name.bad -l table_name.log
```

Create Schema File

The machloader schema file can be created. Import/export is possible even if the data type format is changed using a schema file or the number of columns in the table and data file is different.

Option:

```
-c: schema file creation options
-t: table name specification option
-f: created schema file name specification option
```

Example:

```
machloader -c -t table_name -f table_name.fmt
machloader -c -t table_name -f table_name.fmt -a
```

Set datetime Format in Schema File

The date format can be set to preference with the DATEFORMAT option.

Syntax:

```
# Set for all datetime columns.
DATEFORMAT <dateformat>

# Set for individual datetime column.
DATEFORMAT <column_name> <format>
```

Example:

```
-- Set dateformat for each field in datetest.csv file in the schema file (datetest.fmt).
datetest.fmt
table datetest
{
INS_DT datetime;
UPT_DT datetime;
}
DATEFORMAT ins_dt "YYYY/MM/DD HH12:MI:SS"
DATEFORMAT upt_dt "YYYY DD MM HH12:MI:SS"

datetest.csv
2017/02/20 11:05:23,2017 20 02 11:05:23
2017/02/20 11:06:34,2017 20 02 11:06:34

-- Import datetest.csv file and check input data.
machloader -i -f datetest.fmt -d datetest.csv
---------------------------------------------------------------
Machbase Data Import/Export Utility.
Release Version 5.1.9.community
Copyright 2014, MACHBASE Corporation or its subsidiaries.
All Rights Reserved.
---------------------------------------------------------------
Import time : 0 hour 0 min 0.39 sec
Load success count : 2
Load fail count : 0

mach> SELECT * FROM datetest;
INS_DT UPT_DT
---------------------------------------------------------------
2017-02-20 11:06:34 000:000:000 2017-02-20 11:06:34 000:000:000
```

```
2017-02-20 11:05:23 000:000:000 2017-02-20 11:05:23 000:000:000
[2] row(s) selected.
Elapsed time: 0.000
```

IGNORE

When you do not want to enter a specific field in the CSV file, you can set the IGNORE option in the fmt file.
The ignoretest.csv file has three fields, but if the last field is not needed, specify IGNORE in the column that is not needed in the fmt file.

Example:

```
-- Set ignore option for last field in ignoretest.fmt file.
ignoretest.fmt
table ignoretest
{
ID integer;
MSG varchar(40);
SUB_ID integer IGNORE;
}

ignoretest.csv
1, "msg1", 3
2, "msg2", 4


-- Import ignoretest.csv file and check input data.
machloader -i -f ignoretest.fmt -d ignoretest.csv
------------------------------------------------------------------
Machbase Data Import/Export Utility.
Release Version 5.1.9.community
Copyright 2014, MACHBASE Corporation or its subsidiaries.
All Rights Reserved.
------------------------------------------------------------------
NLS : US7ASCII EXECUTE MODE : IMPORT
SCHMEA FILE : ignoretest.fmt DATA FILE : ignoretest.csv
IMPORT_MODE : APPEND FILED TERM : ,
ROW TERM : \n ENCLOSURE : "
ARRIVAL_TIME : FALSE ENCODING : NONE
HEADER : FALSE CREATE TABLE : FALSE

Progress bar Imported records Error records
2 0

Import time : 0 hour 0 min 0.39 sec
Load success count : 2
Load fail count : 0


mach> SELECT * FROM ignoretest;
ID MSG
------------------------------------------------------------
2 msg2
1 msg1
[2] row(s) selected.
Elapsed time: 0.000
```

If Number of Columns Is More Than Number of Fields

If the number of columns in the table is greater than the number of fields in the data file, only the columns specified in the schema file are entered, and the other columns are entered as NULL.

If Number of Columns Is Less Than Number of Fields

If the number of columns in the table is less than the number of fields in the data file, fields not in the table must be excluded with the IGNORE option.

Example:

```
-- Import ignoretest.csv file and exclude input data by setting ignore option for last field.
loader_test.fmt
table loader_test
{
ID integer;
MSG varchar (40);
SUB_ID integer IGNORE;
}
```

MACHSQL

MACHSQL is an interactive tool that performs SQL queries through the terminal screen.

Index

Run Option Description

```
[mach@localhost]$ machsql -h
```

Short Option	Long Option	Description
-s	--server	Connecting server IP address (default: 127.0.0.1)
-u	--user	User name (default: SYS)
-p	--password	User password (default: MANAGER)
-P	--port	Server port number (default: 5656)
-n	--nls	NLS configuration
-f	--script	SQL script file to run
-o	--output	Filename to save query results
-i	--silent	Runs without the copyright notice
-v	--verbose	Detailed output
-r	--format	Specifies output file format (default: csv)
-h	--help	Displays options

Example:

```
machsql -s localhost -u sys -p manager
machsql --server=localhost --user=sys --password=manager
machsql -s localhost -u sys -p manager -f script.sql
```

SHOW Command

Displays information such as tables, tablespaces, and indexes.
SHOW command list:

- SHOW INDEX
- SHOW INDEXES
- SHOW INDEXGAP
- SHOW LSM
- SHOW LICENSE
- SHOW STATEMENTS
- SHOW TABLE
- SHOW TABLES
- SHOW TABLESPACE
- SHOW TABLESPACES
- SHOW USERS

SHOW INDEX

Displays index information.

Syntax:

```
SHOW INDEX index_name
```

Example:

```
Mach> CREATE TABLE t1 (c1 INTEGER, c2 VARCHAR(10));
Created successfully.
```

```
Mach> CREATE VOLATILE TABLE t2 (c1 INTEGER, c2 VARCHAR(10));
Created successfully.
Mach> CREATE INDEX t1_idx1 ON t1(c1) INDEX_TYPE LSM;
Created successfully.
Mach> CREATE INDEX t1_idx2 ON t1(c1) INDEX_TYPE BITMAP;
Created successfully.
Mach> CREATE INDEX t2_idx1 ON t2(c1) INDEX_TYPE REDBLACK;
Created successfully.
Mach> CREATE INDEX t2_idx2 ON t2(c2) INDEX_TYPE REDBLACK;
Created successfully.

Mach> SHOW INDEX t1_idx2;
TABLE_NAME                              COLUMN_NAME                             INDEX_NAME
-----------------------------------------------------------------------------------------------------------------------------------
INDEX_TYPE   BLOOM_FILTER  KEY_COMPRESS  MAX_LEVEL   PART_VALUE_COUNT BITMAP_ENCODE
-------------------------------------------------------------------------------------
T1                                      C1                                      T1_IDX2
LSM          ENABLE   COMPRESSED    2           100000      EQUAL
[1] row(s) selected.
```

SHOW INDEXES

Displays entire list of indexes.

Syntax:

```
SHOW INDEXES
```

Example:

```
Mach> CREATE TABLE t1 (c1 INTEGER, c2 VARCHAR(10));
Created successfully.
Mach> CREATE VOLATILE TABLE t2 (c1 INTEGER, c2 VARCHAR(10));
Created successfully.
Mach> CREATE INDEX t1_idx1 ON t1(c1) INDEX_TYPE LSM;
Created successfully.
Mach> CREATE INDEX t1_idx2 ON t1(c1) INDEX_TYPE BITMAP;
Created successfully.
Mach> CREATE INDEX t2_idx1 ON t2(c1) INDEX_TYPE REDBLACK;
Created successfully.
Mach> CREATE INDEX t2_idx2 ON t2(c2) INDEX_TYPE REDBLACK;
Created successfully.

Mach> SHOW INDEXES;
TABLE_NAME                              COLUMN_NAME                             INDEX_NAME
-----------------------------------------------------------------------------------------------------------------------------------
INDEX_TYPE
---------------
T1                                      C1                                      T1_IDX1
LSM
T1                                      C1                                      T1_IDX2
LSM
T2                                      C2                                      T2_IDX2
REDBLACK
T2                                      C1                                      T2_IDX1
REDBLACK
[4] row(s) selected.
```

SHOW INDEXGAP

Displays index building GAP information.

Example:

```
Mach> SHOW INDEXGAP
TABLE_NAME                              INDEX_NAME                              GAP
```

```
-----------------------------------------------------------------------------------------------------------
INDEX_TABLE                              T1_IDX1                                  0
INDEX_TABLE                              T1_IDX2                                  0
```

SHOW LSM

Displays LSM index building information.

Example:

```
Mach> SHOW LSM;
TABLE_NAME                               INDEX_NAME                               LEVEL       COUNT
-----------------------------------------------------------------------------------------------------------
T1                                       IDX1                                     0           0
T1                                       IDX1                                     1           100000
T1                                       IDX1                                     2           0
T1                                       IDX1                                     3           0
T1                                       IDX2                                     0           100000
T1                                       IDX2                                     1           0
[6] row(s) selected.
```

SHOW LICENSE

Displays license information.

Example:

```
Mach> SHOW LICENSE
INSTALL_DATE          ISSUE_DATE            EXPIRY_DATE  TYPE        POLICY
-------------------------------------------------------------------------------
2016-07-01 10:24:37   20160325              20170325    2           0
[1] row(s) selected.
```

SHOW STATEMENTS

Displays all query statements (Prepare, Execute, Fetch) registered in the server.

Example:

```
Mach> SHOW STATEMENTS
USER_ID     SESSION_ID  QUERY
-----------------------------------------------------------------------------------------------------------
0           2           SELECT ID USER_ID, SESS_ID SESSION_ID, QUERY FROM V$STMT
[1] row(s) selected.
```

SHOW TABLE

Displays information about the table created by the user.

Syntax:

```
SHOW TABLE table_name
```

Example:

```
Mach> CREATE TABLE t1 (c1 INTEGER, c2 VARCHAR(10));
Created successfully.
Mach> CREATE INDEX t1_idx1 ON t1(c1) INDEX_TYPE LSM;
Created successfully.
Mach> CREATE INDEX t1_idx2 ON t1(c1) INDEX_TYPE BITMAP;
Created successfully.
```

```
Mach> SHOW TABLE T1
[ COLUMN ]
-------------------------------------------------------------
NAME                          TYPE               LENGTH
-------------------------------------------------------------
C1                            integer            11
C2                            varchar            10

[ INDEX ]
-------------------------------------------------------------
NAME                          TYPE               COLUMN
-------------------------------------------------------------
T1_IDX1                       LSM                C1
T1_IDX2                       LSM                C1
```

SHOW TABLES

Displays a list of all tables created by the user.

Example:

```
Mach> SHOW TABLES
NAME
-------------------------------------------
BONUS
DEPT
EMP
SALGRADE
[4] row(s) selected.
```

SHOW TABLESPACE

Displays tablespace information.

Example:

```
Mach> CREATE TABLE t1 (id integer);
Created successfully.
Mach> CREATE INDEX t1_idx_id ON t1(id);
Created successfully.

Mach> SHOW TABLESPACE SYSTEM_TABLESPACE;
[TABLE]
NAME                                  TYPE
-------------------------------------------------------
T1                                    LOG
[1] row(s) selected.

[INDEX]
TABLE_NAME                            COLUMN_NAME                           INDEX_NAME
-------------------------------------------------------------------------------------------------------------------------
T1                                    ID                                    T1_IDX_ID
[1] row(s) selected.
```

SHOW TABLESPACES

Displays a complete list of tablespaces.

Example:

```
Mach> CREATE TABLESPACE tbs1 DATADISK disk1 (DISK_PATH="tbs1_disk1"), disk2 (DISK_PATH="tbs1_disk2"), disk3 (DISK_P
Created successfully.

-- Insert data here
```

```
...
...


Mach> SHOW TABLESPACES;
NAME                                                                                       DISK_COUNT  USAGE
-------------------------------------------------------------------------------------------------------------------------------
SYSTEM_TABLESPACE                                                                          1           0
TBS1                                                                                       3           25824256
[2] row(s) selected.
```

SHOW USERS

Displays a list of users.

Example:

```
Mach> CREATE USER testuser IDENTIFIED BY 'test1234';
Created successfully.

Mach> SHOW USERS;
USER_NAME
--------------------------------------------
SYS
TESTUSER
[2] row(s) selected.
```

CSVIMPORT/CSVEXPORT

'csvimport' and 'csvexport' are tools used to import/export CSV files to the Machbase server.

The options have been simplified for simpler use of the CSV file using the machloader.

Index

csvimport

CSV files can be easily entered into the server using csvimport.

Basic Usage

Enter the table name and data file name according to the following options.

Option:

```
-t: table name specification option
-d: data file naming options
* You can do this with just the table name and data file name without specifying the option.
```

Example:

```
csvimport -t table_name -d table_name.csv
csvimport table_name file_path
csvimport file_path table_name
```

CSV Header Exception

Use the following option to enter the CSV file except for the header at the time of input.

Option:

```
-H: Will not recognize the first line of the csv file as a header.
```

Example:

```
csvimport -t table_name -d table_name.csv -H
```

Automatic Table Creation

If a table is not created to be entered at the time of input, the table can be created at the same time through the following options.

Option:

```
-C: Automatically creates the table during import. Column names are automatically created as c0, c1, .... The creat
-H: Creates column name with csv header name during import.
```

Example:

```
csvimport -t table_name -d table_name.csv -C
csvimport -t table_name -d table_name.csv -C -H
```

csvexport

The database table data can be easily exported to the CSV file with 'csvexport'.

Basic Usage

Option:

```
-t: table name specification option
-d: data file naming options
* You can do this with just the table name and data file name without specifying the option.
```

Example:

```
csvexport -t table_name -d table_name.csv
csvexport table_name file_path
csvexport file_path table_name
```

Using CSV Header

With the following option, you can add a header to the CSV file to be exported with a column name.

Option:

```
-H: Creates the header of the csv file with the table column name.
```

Example:

```
csvexport -t table_name -d table_name.csv -H
```

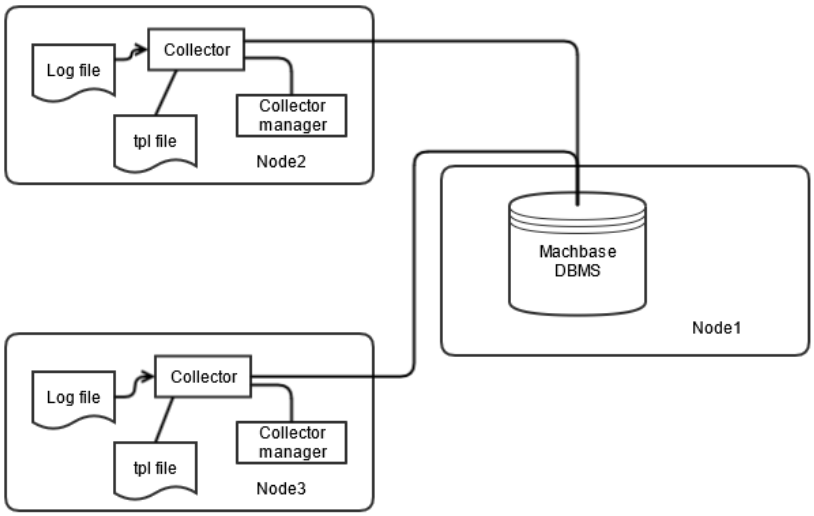
Collector

Machbase Collector is a tool that extracts log data and inputs the data to the Machbase database in real time after converting it.

Machbase Collector can collect log data in real time and input it through the network by being installed in a separate device from the Machbase server. It operates as a separate process from the Machbase server and can run multiple Collectors at the same time. Each Collector process processes one data source.

Index

Concept



The above figure shows Node-2 and Node-3 Collectors collect data and input to Node-1 where the database server is installed.

- In Node-2 and Node-3, the Collectors can be seen **as running a separate process on a specific log file** to send data.
- You can see that each collector process gets **detailed information about the log data using a given tpl file** .
- **The Collector manager** is installed in each Node, manages and monitors the Collector process running on that node.

Characteristic

The main features of the Machbase Collector are described below.

Consistent Interface

Machbase does not require any additional programs in addition to SQL-based commands to execute the Collector. Simply use the following command to manage and monitor the Collector.

```
CREATE Collector MANAGER LOCALHOST AT '127.0.0.1:9999';
CREATE Collector LOCALHOST.MYADP FROM 'syslog.tpl';
ALTER Collector LOCALHOST.MYADP START;
```

Improved Data Collection Performance

Machbase Collector is designed to collect data with a separate Collector for each log data type so each process can process each log file at high speed.

Since separate processes each process log data, they are not affected by other log file processing.
Collector is executed with optimized code for each log type, and data is input with dedicated protocol that minimizes resource usage, so that the best performance can be obtained.

Collection Method

Collectors can be used to collect log data in a variety of ways. The data collection method can be set by modifying the tpl file. The following collection methods are supported.

Method Name	Description
FILE	Collects files from local host.

Method Name	Description
SFTP	Collects files from remote host.
SOCKET	Collects data coming into port.
ODBC	Collects files from other databases.

Log Data Types

Machbase Collector supports regular expressions for various types of log data.

The user can simply modify the existing regular expression to analyze various log files. Currently, the following log types are supported.

Regular Expression File Name	Supported Type	Data Default Location (can be modified)
machbase.rgx	Machbase trace log	$MACHBASE_HOME/trc/machbase.trc
apache_access.rgx	Apache web server access file	/var/log/apache2/access.log
apache_error.rgx	Apache web server error file	/var/log/apache2/access.log
syslog.rgx	sysglog file	/var/log/syslog
custom.rgx	Custome type	Custom file

Easily Supports Custom Logs

Machbase Collector can process various kinds of log files that can be represented as regular expressions.

Even if you do not have a log file, you can test sample log messages and regular expressions using machregex.

Prevents Data Loss in the Event of a Failure

Machbase Collector provides the ability to correctly retransmit data that failed to be transmitted in the event of a failure, after the failure has been resolved.

When a failure occurs, the Collector records the last location it sent to the server, resets the fault, and then resends the data from that location.
So, even if you do not write any additional operations or code to overcome the obstacle, you can transfer it to the server without losing any data.

Ensures High Availability

To ensure high availability of services, multiple Collectors can operate simultaneously on the same data source, and these Collectors transfer data to different Machbase servers.

In this way, even if an error occurs in the Machbase server, the same data is continuously stored in another server, so that the service can be continued.
After resolving the error and restarting the server, the Collector can retransmit the untransmitted log data correctly, thus automatically replicating the data to provide high availability.

Integrated Monitoring Through MWA

Machbase Collector manager synchronizes the Collector's execution information to the Machbase server.

Using this, it is possible to perform integrated monitoring through MWA (Machbase Web Admin).
By using MWA, it is possible to monitor various status information of the running Collector and the status information of the server running the Collector in real time.

Log Pre-Processing Using Python Script

You can write a Python script to manipulate the Collector before it processes the data.

The input data can be processed so as not to input unnecessary data, or the parsed data can be changed.

Collector Installation

Manual Installation

Create `$MACHBASE_COLLECTOR_HOME` directory, copy the downloaded collector package, and unzip it.

```
[mach@localhost ~]$ mkdir mach_collector_home
[mach@localhost ~]$ cp machcollector-x.x.x.official-LINUX-X86-64-release.tgz ma
[mach@localhost ~]$ cd mach_collector_home/
[mach@localhost ~/mach_collector_home]$ tar -zxvf machcollector-x.x.x.official-
bin/
bin/machregex
bin/machcollectord
bin/machcollector
bin/machcollectoradmin
collector/
collector/sign/
collector/apache_access.tpl
collector/samples/
collector/samples/test.rgx
collector/samples/test.tpl
collector/samples/test.log
collector/preprocess/
collector/preprocess/custom.py
collector/preprocess/skip.py
collector/preprocess/replace.py
collector/preprocess/pypyodbc_sample.py
collector/preprocess/trace.py
collector/custom.tpl
collector/syslog.tpl
collector/machbase.tpl
collector/regex/
collector/regex/meta.rgx
collector/regex/syslog.rgx
collector/regex/nxlog.rgx
collector/regex/machbase.rgx
collector/regex/unparse.rgx
collector/regex/apache_error.rgx
collector/regex/apache_access.rgx
conf/
doc/
meta/
trc/
webadmin/
webadmin/flask/
webadmin/flask/Python/
...
```

After decompressing the file, the following directory will be created.

```
[mach@localhost ~/mach_collector_home]$ ls -al
total 60372
drwxrwxr-x  9 mach mach     4096 Jun 27 23:58 .
drwx------ 20 mach mach     4096 Jun 27 23:40 ..
drwxrwxr-x  2 mach mach     4096 Jun 27 23:31 bin
drwxrwxr-x  6 mach mach     4096 Jun 27 23:31 collector
drwxrwxr-x  2 mach mach     4096 Jun 27 23:31 conf
drwxrwxr-x  2 mach mach     4096 Jun 27 23:31 doc
-rw-r--r--  1 mach mach 61783606 Jun 27 23:41 machcollector-x.x.x.official-LINUX-X86-64-release.tgz
drwxrwxr-x  2 mach mach     4096 Jun 27 23:31 meta
drwxrwxr-x  2 mach mach     4096 Jun 27 23:31 trc
drwxrwxr-x  3 mach mach     4096 Jun 27 23:31 webadmin
.. skip
```

Index

Name of directory	Description
bin	Executable file
collector	Configuration file example
conf	Configuration file
doc	License file
lib	Library file
msg	Message file
webadmin	Python package

Environment Variables Configuration

The executable, the shared object file, and the path environment variable need to be set.

```
export MACHBASE_COLLECTOR_HOME=/home/machbase/machbase_home
export PATH=$MACHBASE_COLLECTOR_HOME/bin:$PATH
export LD_LIBRARY_PATH=$MACHBASE_COLLECTOR_HOME/lib:$LD_LIBRARY_PATH
export MACH_COLLECTOR_PORT=9999
```

To apply the above environment variable, execute the following command.

```
source .bashrc
```

If `$MACHBASE_COLLECTOR_PORT` is not specified, the **PORT_NO** value of `$MACH_COLLECTOR_HOME/conf/machcollector.conf` is used.

Creating Collector

Let's look at an example of collecting data in real time from syslog on a Linux system.

Normally, the collector and the Machbase server run on different machines, but in this example we assume that they run on the same machine for convenience.

Index

Run Collector Manager

The collector manager controls the collectors. To control the collector, the collector manager must be executed first. Run the collector manager with the following command:

```
[mach@localhost ~]$ machcollectoradmin --startup
Waiting for collector manager starts.
Collector Manager started successfully.
```

Run the following netstat command to see if the collector manager is running:

```
[mach@localhost ~]$ netstat -anp | grep "LISTEN "
(Not all processes could be identified, non-owned process info
will not be shown, you would have to be root to see it all.)
...
tcp 0 0 0.0.0.0:9999 0.0.0.0:* LISTEN 21163/machcollecto
...
[mach@localhost ~]$
```

Register Collector Manager

To connect the collector manager with the Machbase server, register the collector manager with the Machbase server. Execute the following command using machsql.

```
CREATE COLLECTORMANAGER manager_name AT "host_addr:host_port";
```

- manager_name : The name of the collector manager. Duplicate values are not allowed.
- host_addr: The IP address of the server where the collector manager is running.
- host_port: Port number of the server on which collector manager is running.

```
[mach@localhost ~/mach]$ machsql
================================================================
    Machbase Client Query Utility
    Release Version x.x.x.official
    Copyright 2015, Machbase Inc. or its subsidiaries.
    All Rights Reserved.
================================================================
Machbase server address (Default:127.0.0.1):
Machbase user ID (Default:SYS)
Machbase user password:
MACHBASE_CONNECT_MODE=INET, PORT=5656
mach>CREATE COLLECTORMANAGER LOCALHOST AT "127.0.0.1:9999";
Created successfully.
```

After registering a collector manager on the Machbase server, you can query the status in the m$sys_collectormanagers table.

```
mach> SELECT * FROM m$sys_collectormanagers;
MANAGER_ID MANAGER_NAME MANAGER_HOST MANAGER_PORT MANAGER_STATE
-----------------------------------------------------------------------------------------------------------------------
1 LOCALHOST 127.0.0.1 9999 1
[1] row(s) selected.
```

In the table, the identifier, name, port number, address, and execution status of the collector manager can be inquired.

Create Collector

After registering the collector manager, create the collector object through the collector manager.

Information about the Collector is stored in the Machbase server and can be retrieved. Execute the following command through machsql to create a collector.

```
CREATE COLLECTOR manager_name.collector_name FROM "path_for_template.tpl";
```

- manager_name : The name of the collector manager that runs the collector.
- collector_name: The name of the collector object.
- path_for_template.tpl: The path to the configuration file for collector. The various sample configuration files are located in the "$MACHBASE_COLLECTOR_HOME/collector" directory. It is recommended to select the desired sample file, modify it, and save it as another file.

Prepare Template File

The template file is a text file that describes the Collector's data source, processing method, and storage method. Sample files are provided in the $MACHBASE_COLLECTOR_HOME/collector directory.

Template File Structure

The template file has a structure of "variable name = value" similar to the Machbase property file. Detailed information of each setting variable is shown in the following table.

⚠ Configuration files after Machbase version 3.5 are not backward compatible.

Variable Name	Description	Remarks
COLLECT_TYPE	Data collection method	Sets the data collection method. The data collection method is as follows. FILE defaults to a specific file on the device where the collector is installed. SFTP: Remote SFTP file path, SOCKET: Enters socket input data. ODBC : Enters data from database set to ODBC.
LOG_SOURCE	Location of log file to be read	The location of the data file to be read. In SFTP mode, you must specify the absolute path of the remote host. Not used in SOKET and ODBC modes. It is also possible to set multiple source files or set them to regular expressions.
SFTP_HOST	SFTP_HOST	Host Ip Address
SFTP_PORT	SFTP_PORT	Is set to 22 by default if not set.
SFTP_USER	SFTP username	Is set to anonymous by default.
SFTP_PASS	SFTP password	Is set to anonymous by default.
SOCKET_PORT	Socket port number on which the Collector enters data	
SOCKET_PROTOCOL	Collector socket protocol type	Possible values are TCP and UDP. The default value is TCP.
ODBC_DSN	ODBC mode DSN	".odbc.ini" value
ODBC_QUERY	ODBC mode query	Query string executed to obtain input data from an ODBC data source
ODBC_SEQ_COLUMN	Increased column names in ODBC mode	Only numeric columns are allowed.
LIB_NAME	External link library pass	Not used yet.
REGEX_PATH	Regular expression file for analyzing input data	Not used in ODBC mode.
PREPROCESS_PATH	Location of Python script files for data preprocessing	
SLEEP_TIME	Wait time after inputting data	In milliseconds, with a default of 1000.
DB_TABLE_NAME	Table name to be entered	
DB_ADDR	Database IP address to be entered	
DB_PORT	Database port number	
DB_USER	Database username	
DB_PASS	Database password	
APPEND_MODE	Data input method configuration	Not used as a value for compatibility with past versions.
AUTO_ADD_COLUMN	Whether to automatically generate a table column if it does not exist	Default value is 1.

Variable Name	Description	Remarks
	If 0, it is not generated. If1, it is generated automatically.	
CREATE_TABLE_MODE	Set an operation on the input table. (0: do nothing. 1: truncate the existing table 2: create the table. If an error occurs, write the error to trc and continue 3: drop the table and recreate the table)	Generally recommended to set to 2.
LANG	Specifies the encoding of the input data file.	Available values are UTF-8 (default), CP949 (MS949), KSC5601, EUCJP, SHIFTJIS, BIG5 and BG231280.
REGEX_SORT	Determines the order of the input files.	Default value is ASC and DESC is also possible.
ROTATE_FILE_PATH	Rotation file path configuration	
ROTATE_FILE_COUNT	Rotation file number configuration	
ROTATE_REGEX_SORT	Rotation file order configuration	Default value is ASC. DESC is also possible.

REGEX_PATH, and PREPROCESS_PATH are the files that the collector refers to at run time. Below is a description of the rgx file set in REGEX_PATH.

Variable Name	Description	Remarks
LOG_TYPE	Regular expression name	Value that can be modified, but it is better to keep the value because it is stored together in the database.
COL_LIST	List of columns in the table	Information on the columns belonging to the table
REGEX	Regular expressions for data analysis	
END_REGEX	Regular expression that signifies the end of a record	Regular expression to separate each record. If not set, "\n" line break is used as default.

COL_LIST describes the information linking the log file to the database column. You must set the result of the regular expression and various information to set the column. Complex log data can be entered into structured table columns using COL_LIST.

Variable Name	Description	Remarks
NAME	Column name	String that does not contain spaces.
TYPE	Column data type	Name of the type.
SIZE	Column size	Refers to the actual specified size of the column. The string specifies a different value depending on the size to be created or created. ((short (6), int (11), long (20), float (17), double (17), datetime -defined), ipv4 (15), ipv6 (45), text (64MB), binary (64MB))
DATE_FORMAT	Datetime data format when type is datetime	Internally parses the value using the "strptime" function. e.g.) 'Aug 19 07:56:16' has the format 'month day hour: minute: second'. Therefore, the format values used are as follows. "% b% d% H:% M:% S" add) Additional factors are % 0,% 1,% 2, and are entered with three digits representing milliseconds, microseconds, and nanoseconds, respectively.
USE_INDEX	Whether to create index	Creates LSM or KEYWORD LSM index based on type. 0: Do not create. / 1: Create.
REGEX_NO	Token number within regular expression	Among the REGEX syntax specified in the regular expression file, the "()" parenthesized area is a token. 0 means the entire record data. After that, it becomes the first token from the first parenthesis.

syslog.tpl Example

Below is an example of a syslog.tpl file. The file is provided as a sample in $MACHBASE_COLLECTOR_HOME/collector/syslog.tpl.

```
##############################################################################
# Copyright of this product 2013-2023,
# Machbase Inc. or its subsidiaries.
```

```
# All Rights reserved
##############################################################################

#
#  This file is for Machbase collector template file.
#

######################################################################
# Input setting
######################################################################

COLLECT_TYPE=FILE <== It specifies a method to collect local data.
LOG_SOURCE=/var/log/syslog <== It specifies a location of source file.

######################################################################
# Process setting
######################################################################

REGEX_PATH=syslog.rgx        <== Regular expression file location.
                                 Set $MACHBASE_HOME/collector/regex/ to root

######################################################################
# Output setting
######################################################################

DB_TABLE_NAME = "syslogtable" <== Table name: Data entered here
DB_ADDR       = "127.0.0.1"   <== Running Machbase server IP/PORT
DB_PORT       = 5656
DB_USER       = "SYS"
DB_PASS       = "MANAGER"

# 0: Direct insert
# 1: Prepared insert
# 2: Append
APPEND_MODE=2 <== Data insertion in APPEND mode.

# 0: None, just append.
# 1: Truncate.
# 2: Try to create table. If table already exists, warn it and proceed.
# 3: Drop and create.
CREATE_TABLE_MODE=2 <== Create a table if there is none.
```

The syslog.rgx file is a regular expression file set in the syslog.tpl file. When setting up an rgx file, you can either set it to an absolute path or relative path based on $MACHBASE_COLLECTOR_HOME/collector/regex.

```
##############################################################################
# Copyright of this product 2013-2023,
# Machbase Corporation (Incorporation) or its subsidiaries.
# All Rights reserved
##############################################################################

#
#  This file is for Machbase collector regex file.
#

LOG_TYPE=syslog

COL_LIST= (
     (
        REGEX_NO = 0 <== Regular expression token number
        NAME = tm
        TYPE = datetime
        SIZE = 8
        DATE_FORMAT="%b %d %H:%M:%S" <== datetime format used by strptime function
         ),
     (
        REGEX_NO = 4
        NAME = host
```

```
        TYPE = varchar
        SIZE = 128
        USE_INDEX = 1 <== Whether index is in use
         ),
     (
        REGEX_NO = 5
        NAME = msg
        TYPE = varchar
        SIZE = 512
        USE_INDEX = 1
         )
)

# Below is the regular expression to pares syslog data. It may not work properly if it is modified.
REGEX="(([a-zA-Z]+)\s+([0-9]+)\s+([0-9:]*))\s(\S+)\s+([^\n]+)"

END_REGEX="\n"
```

Create Collector

Create the collector "syslog_test" as shown below.

```
mach> CREATE COLLECTOR localhost.syslog_test FROM "/home/mach/mach_collector_home/collector/syslog.tpl";
Created successfully.
```

Check Collector

The M$SYS_COLLECTORS table contains information about the registered collectors. The collector with the "RUN_FLAG" column value of 1 is running and if it is 0, the execution is stopped.

```
mach> SELECT collector_name, run_flag FROM m$sys_collectors;
collector_name                          run_flag
------------------------------------------------------
SYSLOG_TEST                             0
[1] row(s) selected.
mach> SELECT * FROM m$sys_collectors;
COLLECTOR_ID MANAGER_NAME                                COLLECTOR_NAME
-------------------------------------------------------------------------------------------------------
LOG_TYPE                                TABLE_NAME
-------------------------------------------------------------------------------------
TEMPLATE_NAME                                                                  COLLECT_TYPE
-------------------------------------------------------------------------------------------------------------------------
COLLECTOR_SOURCE
-------------------------------------------------------------------------------
COLLECTOR_LIB                                                                  COL_COUNT
-------------------------------------------------------------------------------------------------
PREPROCESS_PATH
-------------------------------------------------------------------------------
REGEX_PATH
-------------------------------------------------------------------------------
REGEX
-------------------------------------------------------------------------------
END_REGEX
-------------------------------------------------------------------------------
DEFAULT_ADDR                                                                   LANGUAGE
-------------------------------------------------------------------------------------------------------------------------
SLEEP_TIME  DB_ADDR                                DB_PORT     DB_USER
-------------------------------------------------------------------------------------------------------------------
DB_PASS                                 RUN_FLAG
------------------------------------------------------
1           LOCALHOST                              SYSLOG_TEST
syslog                                  syslogtable
/home/mach/mach_collector_home/collector/syslog.tpl                            FILE
/var/log/syslog
```

```
NULL                                                                    7
NULL
syslog.rgx
(([a-zA-Z]+)\s+([0-9]+)\s+([0-9:]*))\s(\S+)\s+([^\n]+)
\n
192.168.122.1                                                           UTF-8
1000        127.0.0.1                               5656        SYS
MANAGER                                 0
[1] row(s) selected.
```

Run Collector

```
ALTER COLLECTOR manager_name.collector_name START [TRACE];
```

To start the registered collector, use the ALTER COLLECTOR statement.

- manager_name : Name of the registered collector manager
- collector_name: The name of the collector to execute.

If an error occurs when executing Collector, you can refer to $MACHBASE_COLLECTOR_HOME/trc/machcollector.trc file for troubleshooting.

```
mach> ALTER COLLECTOR localhost.syslog_test START;
Altered successfully.
mach> SELECT collector_name, run_flag FROM m$sys_collectors;
collector_name                          run_flag
-----------------------------------------------------------
SYSLOG_TEST                             1
[1] row(s) selected.
```

When you start collector with the ALTER COLLECTOR statement, you can see that the value of the RUN_FLAG column has changed by one.

When you start the Collector, a log table is created on the database server where the collected data is stored. The values of collector_type, collector_addr, collector_origin, and collector_offset are set to default values. The tmp, host, and msg columns set in the syslog.tpl file are also created.

```
mach> ALTER COLLECTOR localhost.syslog_test START;
Altered successfully.
mach> SELECT collector_name, run_flag FROM m$sys_collectors;
collector_name                          run_flag
-----------------------------------------------------------
SYSLOG_TEST                             1
[1] row(s) selected.
```

When you execute a query using machsql, you need to make sure that it is connected to the Machbase server and is running. If the Machbase server and collector are installed on different machines, it may not execute normally if the server to which machsql is connected is collector.
When the Collector is executed, the Collector reads the position of the last data entered and re-executes the data.

Data Check

Below is a comparison of the last 10 syslog logs with data and input data.

```
[mach@localhost ~/mach]$ tail -n 10 /var/log/syslog
Jun 28 21:05:01 localhost CROND[12285]: (root) CMD (LANG=C LC_ALL=C /usr/bin/mrtg /etc/mrtg/mrtg.cfg --lock-file /v
Jun 28 21:10:01 localhost CROND[12442]: (root) CMD (LANG=C LC_ALL=C /usr/bin/mrtg /etc/mrtg/mrtg.cfg --lock-file /v
Jun 28 21:10:01 localhost CROND[12443]: (root) CMD (/usr/lib64/sa/sa1 1 1)
Jun 28 21:15:01 localhost CROND[12527]: (root) CMD (LANG=C LC_ALL=C /usr/bin/mrtg /etc/mrtg/mrtg.cfg --lock-file /v
Jun 28 21:20:01 localhost CROND[12609]: (root) CMD (/usr/lib64/sa/sa1 1 1)
Jun 28 21:20:01 localhost CROND[12608]: (root) CMD (LANG=C LC_ALL=C /usr/bin/mrtg /etc/mrtg/mrtg.cfg --lock-file /v
Jun 28 21:25:01 localhost CROND[12707]: (root) CMD (LANG=C LC_ALL=C /usr/bin/mrtg /etc/mrtg/mrtg.cfg --lock-file /v
Jun 28 21:25:01 localhost CROND[12708]: (pcp) CMD ( /usr/libexec/pcp/bin/pmlogger_check -C)
Jun 28 21:25:43 localhost su: pam_unix(su:session): session opened for user root by mach(uid=506)
Jun 28 21:26:02 localhost su: pam_unix(su:session): session closed for user root
```

The following is the last 10 data entered into the Machbase server.

```
mach> SELECT tm, msg FROM syslogtable LIMIT 10;
```

```
tm                              msg
-------------------------------------------------------------------------------------------------------------------
2016-06-28 21:26:02 000:000:000 su: pam_unix(su:session): session closed for user root
2016-06-28 21:25:43 000:000:000 su: pam_unix(su:session): session opened for user root by mach(uid=506)
2016-06-28 21:25:01 000:000:000 CROND[12708]: (pcp) CMD ( /usr/libexec/pcp/bin/pmlogger_check -C)
2016-06-28 21:25:01 000:000:000 CROND[12707]: (root) CMD (LANG=C LC_ALL=C /usr/bin/mrtg /etc/mrtg/mrtg.cfg --loc
                                k-file /var/lock/mrtg/mrtg_l --confcache-file /var/lib/mrtg/mrtg.ok)
2016-06-28 21:20:01 000:000:000 CROND[12608]: (root) CMD (LANG=C LC_ALL=C /usr/bin/mrtg /etc/mrtg/mrtg.cfg --loc
                                k-file /var/lock/mrtg/mrtg_l --confcache-file /var/lib/mrtg/mrtg.ok)
2016-06-28 21:20:01 000:000:000 CROND[12609]: (root) CMD (/usr/lib64/sa/sa1 1 1)
2016-06-28 21:15:01 000:000:000 CROND[12527]: (root) CMD (LANG=C LC_ALL=C /usr/bin/mrtg /etc/mrtg/mrtg.cfg --loc
                                k-file /var/lock/mrtg/mrtg_l --confcache-file /var/lib/mrtg/mrtg.ok)
2016-06-28 21:10:01 000:000:000 CROND[12443]: (root) CMD (/usr/lib64/sa/sa1 1 1)
2016-06-28 21:10:01 000:000:000 CROND[12442]: (root) CMD (LANG=C LC_ALL=C /usr/bin/mrtg /etc/mrtg/mrtg.cfg --loc
                                k-file /var/lock/mrtg/mrtg_l --confcache-file /var/lib/mrtg/mrtg.ok)
2016-06-28 21:05:01 000:000:000 CROND[12285]: (root) CMD (LANG=C LC_ALL=C /usr/bin/mrtg /etc/mrtg/mrtg.cfg --loc
                                k-file /var/lock/mrtg/mrtg_l --confcache-file /var/lib/mrtg/mrtg.ok)
[10] row(s) selected.
```

You can check whether the collector is executed by the following query.

```
mach> SELECT collector_name, run_flag FROM m$sys_collectors;
collector_name                          run_flag
-------------------------------------------------------
SYSLOG_TEST                             1
[1] row(s) selected.
```

Stop Collector

```
ALTER COLLECTOR manager_name.collector_name STOP;
```

```
mach> ALTER COLLECTOR localhost.syslog_test STOP;
Altered successfully.
```

You can stop the collector with the following command:

```
mach> ALTER COLLECTOR localhost.syslog_test STOP;
Altered successfully.
```

Drop Collector

```
DROP COLLECTOR manager_name.collector_name;
```

```
mach> DROP COLLECTOR localhost.syslog_test;
Dropped successfully.
```

Whether the collector is dropped can be confirmed by the following query.

```
mach> SELECT collector_name, run_flag FROM m$sys_collectors;
collector_name                          run_flag
-------------------------------------------------------
[0] row(s) selected.
```

Update Collector

```
ALTER COLLECTOR manager_name.collector_name RELOAD;
```

This is used to change the template file after creating the collector and to apply the new contents. The contents of the template file updated at the time of execution are applied. The following example changes the table into "anothertable" instead of the original value.

```
mach> ALTER COLLECTOR localhost.custom RELOAD;
Altered successfully.

mach> SELECT * FROM m$sys_collectors;
COLLECTOR_ID MANAGER_NAME                                COLLECTOR_NAME
-------------------------------------------------------------------------------------------------------
LOG_TYPE                                 TABLE_NAME
-------------------------------------------------------------------------------------
TEMPLATE_NAME                                                                    COLLECT_TYPE
-------------------------------------------------------------------------------------------------------------------------------------------------
COLLECTOR_SOURCE
-------------------------------------------------------------------------------
COLLECTOR_LIB                                                                    COL_COUNT
-------------------------------------------------------------------------------------------------
PREPROCESS_PATH
-------------------------------------------------------------------------------
REGEX_PATH
-------------------------------------------------------------------------------
REGEX
-------------------------------------------------------------------------------
END_REGEX                                                                        LANGUAGE
-------------------------------------------------------------------------------------------------------------------------------------------------
SLEEP_TIME  DB_ADDR                                  DB_PORT     DB_USER
-------------------------------------------------------------------------------------------------------------
DB_PASS                                  PROCESS_BYTE         PROCESS_RECORD       RUN_FLAG
-------------------------------------------------------------------------------------------------
4           LOCALHOST                                CUSTOM
syslog                                   anothertable
syslog.tpl                                                                       FILE
/var/log/syslog
NULL                                                                             7
NULL
syslog.rgx
(([a-zA-Z]+)\s+([0-9]+)\s+([0-9:]*))\s(\S+)\s+([^\n]+)
\n                                                                               UTF-8
1000        127.0.0.1                                5656        SYS
MANAGER                                  0                    0                    0
[1] row(s) selected.
```

When you look up the meta table, you can see that the input table has been changed to anothertable.

Data Collection Method (1) FILE/SFTP

The various data acquisition modes supported by the Machbase collector and how to apply them are described here. Collector supports FILE, SFTP, SOCKET, and ODBC data collection modes.

The description below assumes that the collector and the Machbase server are installed on the same machine, the hostname of the server is "localhost", and the collector is running at 127.0.0.1:9999.

> **Data Input Mode Configuration**
>
> The data insertion mode can be changed by adjusting the COLLECT_TYPE variable.
>
> You can now configure FILE, SFTP, SOCKET, ODBC, and so on. Additional variables must be set for each mode.

Index

FILE Method

When the file mode input is set, the collector reads and processes the file of the running server.

Additional Option Configuration

In file mode, data is read from localhost, so you only need to check the pathname and file read permissions of that file.

Option Name	Description	Remarks
LOG_SOURCE	Path where log file is located	Must be written in absolute path.

Example (1) Checking Log File

The following example is the file mode input method that collects data from the "/var/log/syslog" file and inputs it to the Machbase server.
First, you need to make sure that the input file can be read by the collector process.

```
[mach@localhost ~]$ head /var/log/syslog
head: cannot open '/var/log/syslog' for reading: Permission denied
```

The above results show that there is no read access to the input file.
You need to grant read permission to the input file (/var/log/syslog) to the user running collector so that the collector process can read the input file.
The following example shows the process.

```
[mach@localhost ~]$ su
# chmod 744 /var/log/syslog
# exit
[mach@localhost ~]$
```

Again, if you check whether you can read the input file using the head command, you get the following result.

```
[mach@localhost ~]$ head /var/log/syslog
Jun 20 04:31:43 localhost kernel: imklog 5.8.10, log source = /proc/kmsg started.
Jun 20 04:31:43 localhost rsyslogd: [origin software="rsyslogd" swVersion="5.8.10" x-pid="15062" x-info="http://www
Jun 20 04:31:46 localhost kernel: imklog 5.8.10, log source = /proc/kmsg started.
Jun 20 04:35:01 localhost CROND[15111]: (root) CMD (LANG=C LC_ALL=C /usr/bin/mrtg /etc/mrtg/mrtg.cfg --lock-file /v
Jun 20 04:40:01 localhost CROND[15188]: (root) CMD (LANG=C LC_ALL=C /usr/bin/mrtg /etc/mrtg/mrtg.cfg --lock-file /v
Jun 20 04:40:01 localhost CROND[15187]: (root) CMD (/usr/lib64/sa/sa1 1 1)
Jun 20 04:45:01 localhost CROND[15265]: (root) CMD (LANG=C LC_ALL=C /usr/bin/mrtg /etc/mrtg/mrtg.cfg --lock-file /v
Jun 20 04:50:01 localhost CROND[15341]: (root) CMD (LANG=C LC_ALL=C /usr/bin/mrtg /etc/mrtg/mrtg.cfg --lock-file /v
Jun 20 04:50:01 localhost CROND[15342]: (root) CMD (/usr/lib64/sa/sa1 1 1)
Jun 20 04:55:01 localhost CROND[15419]: (root) CMD (LANG=C LC_ALL=C /usr/bin/mrtg /etc/mrtg/mrtg.cfg --lock-file /v
[mach@localhost ~]$
```

Example (2) Creating Regular Expression File

After verifying the permissions on the input file, you need to create a regular expression file to parse the data.

You can use the machregex tool provided by Machbase to check that your regular expression matches the data format.

The machregex tool sets two regular expressions as parameters and enters data.

- The first regular expression (REGEX) is used to parse the input data.
- The second regular expression (END_REGEX) is used to separate input data one by one.

The following example shows how to run machregex using regular expressions and data files.

```
[mach@localhost ~]$ head /var/log/syslog > syslog
[mach@localhost ~]$ machregex "^(([a-zA-Z]+)\\s+([0-9]+)\\s+([0-9:]*))\\s(\\S*)\\s+((?:[^\\0])*)$" ".*" < syslog
Pattern => (^(([a-zA-Z]+)\s+([0-9]+)\s+([0-9:]*))\s(\S*)\s+((?:[^\0])*)$)
================================================================================
SUCCESS[2] (rc=7)(Jun 20 04:31:43 localhost kernel: imklog 5.8.10, log source = /proc/kmsg started.
)
  ALL (0:82) => [Jun 20 04:31:43 localhost kernel: imklog 5.8.10, log source = /proc/kmsg started.
]
  0 (0:15) => [Jun 20 04:31:43]
  1 (0:3) => [Jun]
  2 (4:6) => [20]
  3 (7:15) => [04:31:43]
  4 (16:25) => [localhost]
  5 (26:82) => [kernel: imklog 5.8.10, log source = /proc/kmsg started.
]
```

The output value of the parsing result is represented by six tokens. According to this file, Machbase uses 0, 4, and 5 tokens and discards the remaining tokens.

- 0: Time representation string. To convert this string to datetime, you must specify format.
- 4: Host name.
- 5: Log data message.

> If the data you want to analyze is processed without any problems according to the above parsing rules, you can use the same syslog.rgx file provided by Machbase.
>
> (This file is in the `$MACHBASE_HOME/collector/regex/` folder.)
> If you use the regular expression file in this folder, you can set only the file name without using the file path in the **REGEX_PATH** variable of the template file .

Example (3) Creating Template File

Below is an example of a syslog.tpl template file.

```
##############################################################################
# Copyright of this product 2013-2023,
# Machbase Inc. or its subsidiaries.
# All Rights reserved
##############################################################################

#
#  This file is for Machbase collector template file.
#

#########################################################################
# Collect setting
#########################################################################

COLLECT_TYPE=FILE

LOG_SOURCE=/var/log/syslog

#########################################################################
# Process setting
#########################################################################

REGEX_PATH=syslog.rgx

#########################################################################
```

```
# Output setting
######################################################################

DB_TABLE_NAME = "file_syslogtable"
DB_ADDR       = "127.0.0.1"
DB_PORT       = 5656
DB_USER       = "SYS"
DB_PASS       = "MANAGER"

# 0: Direct insert
# 1: Prepared insert
# 2: Append
APPEND_MODE=2

# 0: None, just append.
# 1: Truncate.
# 2: Try to create table. If table already exists, warn it and proceed.
# 3: Drop and create.
CREATE_TABLE_MODE=2
```

In collector settings, files that are not the default file path must specify an absolute path (a path beginning with '/') and a file name.

The file name to be read is specified in the **LOG_SOURCE** variable, and the regular expression file for parsing the data must also be set.

Example (4) Running Collector

Set the information for connecting to the Machbase server and the table generation method, and when the template file is set, execute the collector as follows.

```
Mach> create collector localhost.file_syslog from "/home/machbase_home/collector/syslog.tpl";
Created successfully.

Mach> ALTER COLLECTOR localhost.file_syslog START;
Altered successfully.
```

If the collector is successfully created and executed (and only if there is no table), the table will be created and data will begin to be input.

To check if the data is being input normally, you can check it by executing a SELECT query on the generated table.

```
Mach> SELECT * FROM file_syslogtable ORDER BY _arrival_time asc LIMIT 10;
COLLECTOR_TYPE                   COLLECTOR_ADDR
-----------------------------------------------------
COLLECTOR_ORIGIN                                                                                                 COLLECTOR_OFFSET
-------------------------------------------------------------------------------------------------------------------------------
TM                              HOST
-------------------------------------------------------------------------------------------------------------------------------
MSG
-------------------------------------------------------------------------------
FILE                             127.0.0.1
/var/log/syslog                                                                  81
2016-06-20 04:31:43 000:000:000 localhost
kernel: imklog 5.8.10, log source = /proc/kmsg started.
FILE                             127.0.0.1
/var/log/syslog                                                                  217
2016-06-20 04:31:43 000:000:000 localhost
rsyslogd: [origin software="rsyslogd" swVersion="5.8.10" x-pid="15062" x-info="h
ttp://www.rsyslog.com"] start
FILE                             127.0.0.1
/var/log/syslog                                                                  256
2016-06-20 04:31:46 000:000:000 localhost
kernel: imklog 5.8.10, log source = /proc/kmsg started.
FILE                             127.0.0.1
/var/log/syslog                                                                  431
2016-06-20 04:35:01 000:000:000 localhost
CROND[15111]: (root) CMD (LANG=C LC_ALL=C /usr/bin/mrtg /etc/mrtg/mrtg.cfg --loc
k-file /var/lock/mrtg/mrtg_l --confcache-file /var/lib/mrtg/mrtg.ok)
FILE                             127.0.0.1
/var/log/syslog                                                                  606
2016-06-20 04:40:01 000:000:000 localhost
CROND[15188]: (root) CMD (LANG=C LC_ALL=C /usr/bin/mrtg /etc/mrtg/mrtg.cfg --loc
k-file /var/lock/mrtg/mrtg_l --confcache-file /var/lib/mrtg/mrtg.ok)
FILE                             127.0.0.1
```

```
/var/log/syslog                                                              681
2016-06-20 04:40:01 000:000:000 localhost
CROND[15187]: (root) CMD (/usr/lib64/sa/sa1 1 1)
FILE                            127.0.0.1
/var/log/syslog                                                              856
2016-06-20 04:45:01 000:000:000 localhost
CROND[15265]: (root) CMD (LANG=C LC_ALL=C /usr/bin/mrtg /etc/mrtg/mrtg.cfg --loc
k-file /var/lock/mrtg/mrtg_l --confcache-file /var/lib/mrtg/mrtg.ok)
FILE                            127.0.0.1
/var/log/syslog                                                             1031
2016-06-20 04:50:01 000:000:000 localhost
CROND[15341]: (root) CMD (LANG=C LC_ALL=C /usr/bin/mrtg /etc/mrtg/mrtg.cfg --loc
k-file /var/lock/mrtg/mrtg_l --confcache-file /var/lib/mrtg/mrtg.ok)
FILE                            127.0.0.1
/var/log/syslog                                                             1106
2016-06-20 04:50:01 000:000:000 localhost
CROND[15342]: (root) CMD (/usr/lib64/sa/sa1 1 1)
FILE                            127.0.0.1
/var/log/syslog                                                             1281
2016-06-20 04:55:01 000:000:000 localhost
CROND[15419]: (root) CMD (LANG=C LC_ALL=C /usr/bin/mrtg /etc/mrtg/mrtg.cfg --loc
k-file /var/lock/mrtg/mrtg_l --confcache-file /var/lib/mrtg/mrtg.ok)
[10] row(s) selected.
```

SFTP Method

You can use SFTP mode to collect data from remote files.

Remote files must be accessible via SFTP. It is similar to FILE mode, but SFTP related variables should be set because the file is accessed through SFTP.

Additional Option Configuration

To collect data in SFTP mode, the following variables must be set.

Option Name	Description	Remarks
LOG_SOURCE	Data file path and file name of remote location	Must be written in absolute path.
SFTP_HOST	SFTP server IP address	
SFTP_PORT	SFTP server port number	Is set to 22 by default if not port number is set.
SFTP_USER	SFTP username	
SFTP_PASS	SFTP password	

Example (1) Checking SFTP Access

If you can not read the login and collection files with SFTP, you should try the following.

1. Troubleshoot SFTP sign-in issues
2. Troubleshoot file permissions (see FILE method)

Example (2-3) Creating Regular Expression/Template File

Once this is done, create a regular expression file and a template file.
For the description of this operation, refer to FTP method above.

```
################################################################################
# Copyright of this product 2013-2023,
# Machbase Corporation (Incorporation) or its subsidiaries.
# All Rights reserved
################################################################################

#
#  This file is for Machbase collector template file.
#

#####################################################################
```

```
# Collect setting
######################################################################

COLLECT_TYPE=SFTP

SFTP_HOST=127.0.0.1
SFTP_PORT=22
SFTP_USER=mach
SFTP_PASS=mach

LOG_SOURCE=/var/log/syslog

######################################################################
# Process setting
######################################################################

REGEX_PATH=syslog.rgx

######################################################################
# Output setting
######################################################################

DB_TABLE_NAME = "sftp_syslogtable"
DB_ADDR       = "127.0.0.1"
DB_PORT       = 5656
DB_USER       = "SYS"
DB_PASS       = "MANAGER"

# 0: Direct insert
# 1: Prepared insert
# 2: Append
APPEND_MODE=2

# 0: None, just append
# 1: Truncate
# 2: Try to create table. If table already exists, warn it and proceed.
# 3: Drop and create
CREATE_TABLE_MODE=2
```

Example (4) Running Collector

The following example shows how to create a collector using SFTP with the above template file.

```
Mach> create collector localhost.sftp_syslog from "/home/mach/mach_collector_home/collector/sftp_syslog.tpl";
Created successfully.

Mach> alter collector localhost.sftp_syslog start;
Altered successfully.
```

If you have successfully created and started the Collector, you can see the data collected by the Collector as follows:

```
Mach> select * from sftp_syslogtable order by _arrival_time asc limit 10;
COLLECTOR_TYPE                   COLLECTOR_ADDR
-------------------------------------------------
COLLECTOR_ORIGIN                                                                                    COLLECTOR_OFFSET
-----------------------------------------------------------------------------------------------------------------------
TM                              HOST
-----------------------------------------------------------------------------------------------------------------------------------------------
MSG
--------------------------------------------------------------------------------
SFTP                             127.0.0.1
/var/log/syslog                                                                                     81
2016-06-20 04:31:43 000:000:000 localhost
kernel: imklog 5.8.10, log source = /proc/kmsg started.
SFTP                             127.0.0.1
/var/log/syslog                                                                                     217
2016-06-20 04:31:43 000:000:000 localhost
```

```
rsyslogd: [origin software="rsyslogd" swVersion="5.8.10" x-pid="15062" x-info="h
ttp://www.rsyslog.com"] start
SFTP                            127.0.0.1
/var/log/syslog                                                          256
2016-06-20 04:31:46 000:000:000 localhost
kernel: imklog 5.8.10, log source = /proc/kmsg started.
SFTP                            127.0.0.1
/var/log/syslog                                                          431
2016-06-20 04:35:01 000:000:000 localhost
CROND[15111]: (root) CMD (LANG=C LC_ALL=C /usr/bin/mrtg /etc/mrtg/mrtg.cfg --loc
k-file /var/lock/mrtg/mrtg_l --confcache-file /var/lib/mrtg/mrtg.ok)
SFTP                            127.0.0.1
/var/log/syslog                                                          606
2016-06-20 04:40:01 000:000:000 localhost
CROND[15188]: (root) CMD (LANG=C LC_ALL=C /usr/bin/mrtg /etc/mrtg/mrtg.cfg --loc
k-file /var/lock/mrtg/mrtg_l --confcache-file /var/lib/mrtg/mrtg.ok)
SFTP                            127.0.0.1
/var/log/syslog                                                          681
2016-06-20 04:40:01 000:000:000 localhost
CROND[15187]: (root) CMD (/usr/lib64/sa/sa1 1 1)
SFTP                            127.0.0.1
/var/log/syslog                                                          856
2016-06-20 04:45:01 000:000:000 localhost
CROND[15265]: (root) CMD (LANG=C LC_ALL=C /usr/bin/mrtg /etc/mrtg/mrtg.cfg --loc
k-file /var/lock/mrtg/mrtg_l --confcache-file /var/lib/mrtg/mrtg.ok)
SFTP                            127.0.0.1
/var/log/syslog                                                         1031
2016-06-20 04:50:01 000:000:000 localhost
CROND[15341]: (root) CMD (LANG=C LC_ALL=C /usr/bin/mrtg /etc/mrtg/mrtg.cfg --loc
k-file /var/lock/mrtg/mrtg_l --confcache-file /var/lib/mrtg/mrtg.ok)
SFTP                            127.0.0.1
/var/log/syslog                                                         1106
2016-06-20 04:50:01 000:000:000 localhost
CROND[15342]: (root) CMD (/usr/lib64/sa/sa1 1 1)
SFTP                            127.0.0.1
/var/log/syslog                                                         1281
2016-06-20 04:55:01 000:000:000 localhost
CROND[15419]: (root) CMD (LANG=C LC_ALL=C /usr/bin/mrtg /etc/mrtg/mrtg.cfg --loc
k-file /var/lock/mrtg/mrtg_l --confcache-file /var/lib/mrtg/mrtg.ok)
[10] row(s) selected.
```

Collection Method (2) Socket/ODBC

Socket Method

The collector reads data from the socket, parses it, and enters it into the database server. To use socket mode, you must set the port number. With this mode, log data processed by programs such as rsyslog, logstash, and nxlog can be received and processed.

Additional Value Configuration

Setting the SOCKET_PORT variable is necessary to receive data in SOCKET mode.

Option Name	Description	Remarks
SOCKET_PORT	Port number on which to receive data	You must specify a port that other programs are not using.

Index

Example (1) Creating Regular Expression/Template File

This example is to collect data in SOCKET mode. SOCKET mode opens a socket and waits for an external program to enter data.

It is recommended to use rsyslog to collect syslogs. This example uses rsyslog to collect and analyze data.
Data transmission is performed by another program. In the following configuration file, only the configuration of the rgx file and the tpl file is handled.

> The syslog.rgx used in the FILE mode data processing example is still available.

In the example below, the socket port number is 33333.This port must allow access from the firewall.

```
##############################################################################
# Copyright of this product 2013-2023,
# Machbase Corporation (Incorporation) or its subsidiaries.
# All Rights reserved
##############################################################################

# This file is for Machbase collector template file.

#########################################################################
# Collect setting
#########################################################################

COLLECT_TYPE=SOCKET

SOCKET_PORT=33333

#########################################################################
# Process setting
#########################################################################

REGEX_PATH=syslog.rgx

#########################################################################
# Output setting
#########################################################################

DB_TABLE_NAME = "socket_syslogtable"
DB_ADDR = "127.0.0.1"
DB_PORT = 5656
DB_USER = "SYS"
DB_PASS = "MANAGER"
```

```
# 0: Direct insert
# 1: Prepared insert
# 2: Append
APPEND_MODE=2

# 0: None, just append.
# 1: Truncate.
# 2: Try to create table. If table already exists, warn it and proceed.
# 3: Drop and create.
CREATE_TABLE_MODE=2
```

Example (2) Running Collector

Use the socket_syslog.tpl file in the example above to create a collector and run it. (Creation and execution are the same as FILE mode.)

```
Mach> create collector localhost.file_syslog from "/home/machbase_home/collector/socket_syslog.tpl";
Created successfully.

Mach> ALTER COLLECTOR localhost.file_syslog START;
Altered successfully.
```

Run the following command to verify that the collector is running normally and is connected.

```
[mach@localhost ~]$ netstat -anp | grep "LISTEN " | grep 33333
(Not all processes could be identified, non-owned process info will not be shown, you would have to be root to see
tcp 0 0 0.0.0.0:33333 0.0.0.0:* LISTEN 20818/machcollecto
[mach@localhost ~]$
```

From the above results, you can see that the collector is waiting for data input on port 33333.

Example (3) Data Insert

Enter the data using rsyslog.

Log in the rsyslog program as a user with root privileges and create the rsyslog configuration file (/etc/rsyslog.d/.conf) as follows.

```
*.* @@127.0.0.1:33333
```

Or you can make the following detailed settings.

Create the following file as "/etc/rsyslog.d/127.0.0.1_syslog.conf" file. When rsyslog is restarted, it outputs data to socket every time syslog data is generated.

```
$ModLoad imfile

$InputFileName /var/log/syslog
$InputFileTag syslog_file:
$InputFileStateFile stat-syslog
$InputFilePollInterval 1
$InputRunFileMonitor

if $programname == 'syslog_file' then @@127.0.0.1:33333
```

Then restart the rsyslog daemon.

```
# service rsyslog restart
rsyslog stop/waiting
rsyslog start/running, process 22936
```

When rsyslog is restarted, /var/log/syslog data is written to the port 127.0.0.1:33333.

The collector gathers data through a socket connection, analyzes it, and enters it into the Machbase server.
Since the **$InputFilePollInterval** variable is set to 1 in the rsyslog setting, the data input rate may be slow at the beginning.
If the data is entered normally, the database server can check the data in the corresponding table using the following SELECT query.

```
Mach> select * from socket_syslogtable order by _arrival_time asc limit 10;
COLLECTOR_TYPE COLLECTOR_ADDR
-----------------------------------------------------
COLLECTOR_ORIGIN COLLECTOR_OFFSET
-------------------------------------------------------------------------------------------------------------
TM HOST
-------------------------------------------------------------------------------------------------------------------------------
MSG
-------------------------------------------------------------------------------
SOCKET 127.0.0.1
NULL 1
2016-06-28 23:50:17 000:000:000 localhost
syslog_file: Jun 20 04:31:43 localhost kernel: imklog 5.8.10, log source = /proc
/kmsg started.
SOCKET 127.0.0.1
NULL 2
2016-06-28 23:50:17 000:000:000 localhost
syslog_file: Jun 20 04:31:43 localhost rsyslogd: [origin software="rsyslogd" swV
ersion="5.8.10" x-pid="15062" x-info="http://www.rsyslog.com"] start
SOCKET 127.0.0.1
NULL 3
2016-06-28 23:50:17 000:000:000 localhost
syslog_file: Jun 20 04:31:46 localhost kernel: imklog 5.8.10, log source = /proc/kmsg started.
SOCKET 127.0.0.1
NULL 4
2016-06-28 23:50:17 000:000:000 localhost
syslog_file: Jun 20 04:35:01 localhost CROND[15111]: (root) CMD (LANG=C LC_ALL=C
/usr/bin/mrtg /etc/mrtg/mrtg.cfg --lock-file /var/lock/mrtg/mrtg_l --confcache-
file /var/lib/mrtg/mrtg.ok)
SOCKET 127.0.0.1
NULL 5
2016-06-28 23:50:17 000:000:000 localhost
syslog_file: Jun 20 04:40:01 localhost CROND[15188]: (root) CMD (LANG=C LC_ALL=C
/usr/bin/mrtg /etc/mrtg/mrtg.cfg --lock-file /var/lock/mrtg/mrtg_l --confcache-
file /var/lib/mrtg/mrtg.ok)
SOCKET 127.0.0.1
NULL 6
2016-06-28 23:50:17 000:000:000 localhost
syslog_file: Jun 20 04:40:01 localhost CROND[15187]: (root) CMD (/usr/lib64/sa/s
a1 1 1)
SOCKET 127.0.0.1
NULL 7
2016-06-28 23:50:17 000:000:000 localhost
syslog_file: Jun 20 04:45:01 localhost CROND[15265]: (root) CMD (LANG=C LC_ALL=C
/usr/bin/mrtg /etc/mrtg/mrtg.cfg --lock-file /var/lock/mrtg/mrtg_l --confcache-
file /var/lib/mrtg/mrtg.ok)
SOCKET 127.0.0.1
NULL 8
2016-06-28 23:50:17 000:000:000 localhost
syslog_file: Jun 20 04:50:01 localhost CROND[15341]: (root) CMD (LANG=C LC_ALL=C
/usr/bin/mrtg /etc/mrtg/mrtg.cfg --lock-file /var/lock/mrtg/mrtg_l --confcache-
file /var/lib/mrtg/mrtg.ok)
SOCKET 127.0.0.1
NULL 9
2016-06-28 23:50:17 000:000:000 localhost
syslog_file: Jun 20 04:50:01 localhost CROND[15342]: (root) CMD (/usr/lib64/sa/s
a1 1 1)
SOCKET 127.0.0.1
NULL 10
2016-06-28 23:50:17 000:000:000 localhost
syslog_file: Jun 20 04:55:01 localhost CROND[15419]: (root) CMD (LANG=C LC_ALL=C
/usr/bin/mrtg /etc/mrtg/mrtg.cfg --lock-file /var/lock/mrtg/mrtg_l --confcache-
file /var/lib/mrtg/mrtg.ok)
[10] row(s) selected.
```

Log Collector Configuration

The socket input mode example is executed using *rsyslog, logstash, and nxlog* .

When these programs input log data through a socket, the collector collects them and inputs them to the database server.

rsyslog

rsyslog is often included by default in recent Linux distributions.

So there is no need to install it any more. Just add the configuration file to the /etc/rsyslog.d/ directory and restart the rsyslog daemon.
rsyslog does not only transmit log data that is already recorded, but also transmits data whenever a new log data is recorded.
Below is a list of settings.

Simple Setting: Set Log Forwarding Address

This is the only way to specify the address to be forwarded when the log is created. It is simpler than other methods.

Restart rsyslog after writing the following in /etc/rsyslog.d/.conf file.

```
*.* @@<Collector host>:<Collector port>
```

The syslog data is then sent to **collector host: port** .

Complex Setting: Set Input Log File / Transmission Frequency

A more complex method is to set the input log file and transmission frequency.

```
$ModLoad imfile

$InputFileName /var/log/syslog
$InputFileTag syslog_file:
$InputFileStateFile stat-syslog
$InputFilePollInterval 1
$InputRunFileMonitor

if $programname == 'syslog_file' then @@<Collector host>:<Collector port>
```

Create a file that ends in .conf in the /etc/rsyslog.d/ folder, and then restart rsyslog.

See the rsyslog description for more details .

logstash

To install logstash, refer to Getting Started with Logstash .

You can modify the logstash conf file to send the desired data to the socket.
See the example below.

```
input {
   file {
      path => "<Absolute path of log file>"
      }
   }
output {
   tcp {
      host => "<Collector host>"
      port => "<Collector port>"
        }
      }
```

- Set the location of the input data file in the "input" section. If you want to enter syslog, set /var/log/syslog.
- In the "output" section, you need to enter the collector's tcp socket, so set tcp and set ip and port number.

Nxlog

Nxlog is a log collector for Windows.

The configuration of the rgx, tpl file for the collector for the socket input mode is the same, and an example of the configuration file for nxlog is as follows.
Usually nxlog is installed in "C:₩Program Files₩nxlog" or "C:₩Program Files (x86)₩nxlog".
Create a configuration file located in the above path as follows.

```
## This is a sample configuration file. See the nxlog reference manual about the
## configuration options. It should be installed locally and is also available
## online at http://nxlog-ce.sourceforge.net/nxlog-docs/en/nxlog-reference-manual.html
```

```
## Set the ROOT to the folder your nxlog was installed,
## otherwise it will not start.


#define ROOT C:\Program Files\nxlog
define ROOT C:\Program Files\nxlog


Moduledir %ROOT%\modules
CacheDir %ROOT%\data
Pidfile %ROOT%\data\nxlog.pid
SpoolDir %ROOT%\data
LogFile %ROOT%\data\nxlog.log


<Input in>
Module im_msvistalog
# For windows 2003 and earlier use the following:
# Module im_mseventlog
</Input>


<Output out>
Module om_tcp
Host <Collector host>
Port <Collector port>
</Output>


<Route 1>
Path in => out
</Route>
```

In the example above, when data is written to the im_msvistalog file, data transmission through the socket is set to <collector ip>: <collector port>.

Changing the configuration file and restarting the service will send the data to the collector via the socket.
Refer to the nxlog manual for details.

ODBC Method

ODBC mode is a method of collecting data from a database that can be accessed by an ODBC connection to a collector.

In Linux environments, you need to install the unixODBC package.
The following example shows how to collect data from a MySQL database through unixODBC.
Please refer to the respective websites for how to install unixODBC and MyODBC .

Additional Value Configuration

The following variables must be set.

Option Name	Description	Remarks
ODBC_DSN	DSN for accessing the database	You must use the DSN described in the ODBC configuration.
ODBC_QUERY	Query for data retrieval	
ODBC_SEQ_COLUMN	Column name of incremental value	Must be one of the columns to be queried. Only numeric types are supported.

Example (1) Generating Data

You must first enter the MySQL data. Enter data as follows.

```
0,2015-05-20 06:00:00,16.194.51.72,6790,183.103.50.46,5281,20,GET /twiki/bin/view/TWiki/KlausWriessnegger HTTP/1.1,
1,2015-05-20 06:00:02,96.40.75.42,11011,31.224.72.52,12069,55,GET /twiki/bin/search/Main/SearchResult?scope=text®ex
2,2015-05-20 06:00:02,174.47.129.59,6032,96.40.75.42,6442,72,GET /twiki/bin/edit/Main/WebSearch?t=1078669682 HTTP/1
3,2015-05-20 06:00:02,153.199.166.54,4220,86.45.186.17,2245,1,GET /twiki/bin/oops/TWiki/RichardDonkin?template=oops
4,2015-05-20 06:00:02,226.7.237.25,10805,50.230.44.173,179,70,GET /twiki/bin/oops/TWiki/AppendixFileSystem?template
5,2015-05-20 06:00:02,183.103.50.46,7175,96.128.212.177,7175,73,GET /twiki/bin/view/TWiki/DontNotify HTTP/1.1,200,4
```

```
6,2015-05-20 06:00:02,123.198.82.192,6784,63.214.191.124,10825,21,GET /twiki/bin/view/TWiki/DontNotify HTTP/1.1,200
7,2015-05-20 06:00:02,214.153.107.182,5562,85.183.139.166,1367,8,GET /twiki/bin/oops/TWiki/RichardDonkin?template=o
8,2015-05-20 06:00:02,245.13.24.17,7451,69.99.246.62,4497,20,GET /twiki/bin/view/Main/SpamAssassin HTTP/1.1,200,408
9,2015-05-20 06:00:02,239.81.105.222,2245,71.129.68.118,1641,59,GET /twiki/bin/view/Main/TokyoOffice HTTP/1.1,200,3
```

The table columns to be collected are seq, at, srcip, srcport, dstip, dstport, protocol, eventlog, eventcode, and eventsize. Set seq as a sequential increment column. To create a table with this structure, perform the following query in mysql:

```
mysql> create table odbc_seq_int_10 (seq int(9), at timestamp, srcip varchar(20), srcport int(6), dstip varchar(20)
```

After successfully creating the table, enter the data into the MySQL database with the following command:

```
mysql> load data infile '<File Path>' into table <Table Name> fields terminated by ',';
Query OK, 10 rows affected (0.00 sec)
Records: 10 Deleted: 0 Skipped: 0 Warnings: 0
```

For more information about running MySQL databases, see the MySQL manual.

Example (2) ODBC Configuration

Create an ODBC configuration file to access the MySQL database.

```
[MYSQL]
Driver=MySQL
Server=<Database host address>
Port=<Database host port>
Database=<Name of default database to access>
UID=<User ID>
PWD=<User password>
```

In unixODBC, USER DATA SOURCES is used first, so make the above contents in the .odbc.ini file in the user home directory that runs collector.

Example (3) Checking ODBC Settings

Use the unixODBC isql program to verify that the ODBC configuration is working properly. Execute the following as a parameter of MYSQL which is the set DSN.

```
$ isql -v MYSQL
```

If it is installed and configured normally, you will get the following results.

```
+---------------------------------------+
| Connected!                            |
|                                       |
| sql-statement                         |
| help [tablename]                      |
| quit                                  |
|                                       |
+---------------------------------------+
SQL>
```

You can query the input data through isql.

```
SQL> select * from odbc_seq_int_10;
+-----+---------------------+-----------------+---------+----------------+---------+----------+--------------------
| seq | at                  | srcip           | srcport | dstip          | dstport | protocol | eventlog
+-----+---------------------+-----------------+---------+----------------+---------+----------+--------------------
| 0   | 2015-05-20 06:00:00 | 16.194.51.72    | 6790    | 183.103.50.46  | 5281    | 20       | GET /twiki/bin/viev
| 1   | 2015-05-20 06:00:02 | 96.40.75.42     | 11011   | 31.224.72.52   | 12069   | 55       | GET /twiki/bin/sear
| 2   | 2015-05-20 06:00:02 | 174.47.129.59   | 6032    | 96.40.75.42    | 6442    | 72       | GET /twiki/bin/edit
| 3   | 2015-05-20 06:00:02 | 153.199.166.54  | 4220    | 86.45.186.17   | 2245    | 1        | GET /twiki/bin/oops
| 4   | 2015-05-20 06:00:02 | 226.7.237.25    | 10805   | 50.230.44.173  | 179     | 70       | GET /twiki/bin/oops
| 5   | 2015-05-20 06:00:02 | 183.103.50.46   | 7175    | 96.128.212.177 | 7175    | 73       | GET /twiki/bin/viev
| 6   | 2015-05-20 06:00:02 | 123.198.82.192  | 6784    | 63.214.191.124 | 10825   | 21       | GET /twiki/bin/viev
| 7   | 2015-05-20 06:00:02 | 214.153.107.182 | 5562    | 85.183.139.166 | 1367    | 8        | GET /twiki/bin/oops
| 8   | 2015-05-20 06:00:02 | 245.13.24.17    | 7451    | 69.99.246.62   | 4497    | 20       | GET /twiki/bin/viev
```

```
| 9   | 2015-05-20 06:00:02 | 239.81.105.222  | 2245    | 71.129.68.118  | 1641    | 59       | GET /twiki/bin/viev
+-----+---------------------+-----------------+---------+----------------+---------+----------+--------------------
SQLRowCount returns 10
10 row(s) fetched.
SQL>
```

If you can not get these results, check the names of unixODBC and DSN. Refer to the unixODBC documentation for related details.

Example (4) Collector Configuration

Create a tpl file using the query, DSN, username and password used above.

In the ODBC mode input method, REGEX_PATH is not needed because data is provided separately for each column.

```
##############################################################################
# Copyright of this product 2013-2023,
# Machbase Corporation (Incorporation) or its subsidiaries.
# All Rights reserved
##############################################################################

#
# This file is for Machbase collector template file.
#

######################################################################
# Collect setting
######################################################################

COLLECT_TYPE=ODBC
ODBC_DSN=MYSQL <= Name of driver specified in "odbc.ini"
ODBC_QUERY="select * from sample_seq_int_10" <= Not required to input other queries except select <Columns> from <T
ODBC_SEQ_COLUMN=seq <= The base column name that determines the order in the query results

######################################################################
# Process setting
######################################################################

#PREPROCESS_PATH=Python script file path

######################################################################
# Output setting
######################################################################

DB_TABLE_NAME = "sample_seq_int"
DB_ADDR = <Machbase Server Host>
DB_PORT = <Machbase Server Port>
DB_USER = "SYS"
DB_PASS = "MANAGER"

# 0: Direct insert
# 1: Prepared insert
# 2: Append
APPEND_MODE=2

# 0: None, just append.
# 1: Truncate.
# 2: Try to create table. If table already exists, warn it and proceed.
# 3: Drop and create.
CREATE_TABLE_MODE=2
```

Use the above tpl file to create and start the collector, and then check the results using machsql.

```
Mach> select * from sample_seq_int limit 50;
COLLECTOR_TYPE COLLECTOR_ADDR
--------------------------------------------------
COLLECTOR_ORIGIN COLLECTOR_OFFSET SEQ
-------------------------------------------------------------------------------------------------------------------
AT SRCIP SRCPORT DSTIP DSTPORT PROTOCOL
-------------------------------------------------------------------------------------------------------------------
```

```
EVENTLOG EVENTCODE EVENTSIZE
-----------------------------------------------------------------------------------------------------------------
ODBC 192.168.122.1
MYSQL 9 9
1900-01-03 19:54:55 000:000:000 239.81.105.222 2245 71.129.68.118 1641 59
GET /twiki/bin/view/Main/TokyoOffice HTTP/1.1 200 3853
ODBC 192.168.122.1
MYSQL 8 8
1900-01-03 19:54:55 000:000:000 245.13.24.17 7451 69.99.246.62 4497 20
GET /twiki/bin/view/Main/SpamAssassin HTTP/1.1 200 4081
ODBC 192.168.122.1
MYSQL 7 7
1900-01-03 19:54:55 000:000:000 214.153.107.182 5562 85.183.139.166 1367 8
GET /twiki/bin/oops/TWiki/RichardDonkin?template=oopsmore¶m1=1.2¶m2=1.2 HTTP/1
.1 200 11281
ODBC 192.168.122.1
MYSQL 6 6
1900-01-03 19:54:55 000:000:000 123.198.82.192 6784 63.214.191.124 10825 21
GET /twiki/bin/view/TWiki/DontNotify HTTP/1.1 200 4140
ODBC 192.168.122.1
MYSQL 5 5
1900-01-03 19:54:55 000:000:000 183.103.50.46 7175 96.128.212.177 7175 73
GET /twiki/bin/view/TWiki/DontNotify HTTP/1.1 200 4140
ODBC 192.168.122.1
MYSQL 4 4
1900-01-03 19:54:55 000:000:000 226.7.237.25 10805 50.230.44.173 179 70
GET /twiki/bin/oops/TWiki/AppendixFileSystem?template=oopsmore¶m1=1.12¶m2=1.12
HTTP/1.1 200 11382
ODBC 192.168.122.1
MYSQL 3 3
1900-01-03 19:54:55 000:000:000 153.199.166.54 4220 86.45.186.17 2245 1
GET /twiki/bin/oops/TWiki/RichardDonkin?template=oopsmore¶m1=1.2¶m2=1.2 HTTP/1
.1 200 11281
ODBC 192.168.122.1
MYSQL 2 2
1900-01-03 19:54:55 000:000:000 174.47.129.59 6032 96.40.75.42 6442 72
GET /twiki/bin/edit/Main/WebSearch?t=1078669682 HTTP/1.1 401 12846
ODBC 192.168.122.1
MYSQL 1 1
1900-01-03 19:54:55 000:000:000 96.40.75.42 11011 31.224.72.52 12069 55
GET /twiki/bin/search/Main/SearchResult?scope=text®ex=on&search=Office%20*Locat
ions[notA-Za-z] HTTP/1.1 200 7771
ODBC 192.168.122.1
MYSQL 0 0
1900-01-03 19:54:55 000:000:000 16.194.51.72 6790 183.103.50.46 5281 20
GET /twiki/bin/view/TWiki/KlausWriessnegger HTTP/1.1 200 3848
[10] row(s) selected.
```

You can see that the data is output in reverse order to the MySQL output.

Collecting Custom Logs

This chapter describes the definition of user-defined log data and how it is collected through the collector.

Index

Data Conversion Concept

The following figure shows the order in which the collector operates. First, the original data is read and parsed using a regular expression file. (The location of this regular expression file is described in the template file.) The parsed data is then assigned to the column using the COL_LIST value defined in the rgx file.

To collect custom log data, create a regular expression to parse the original data, describe the COL_LIST and regular expression file in the template file, and create and execute the collector using the template.

machregex

To collect data with the Collector, you must create a regular expression to parse the input data. Machbase provides machregex, a tool for checking whether a regular expression you create can correctly parse the desired input data.

Machbase provides examples of regular expressions that can parse SYSLOG, the ACCESS log of the Apache web server, and the TRACE LOG of the Machbase server. Machbase uses Perl Compatible Regular Expressions (PCRE) libraries to support regular expressions.

Run machregex

```
[mach@localhost ~/mach_collector_home/bin]$ ./machregex
================================================================
     Machbase Collector Regex Utility
     Release Version 3.0.0.8634.official
     Copyright 2015, Machbase Inc. or its subsidiaries.
     All Rights Reserved.
================================================================

Usage> ./machregex Pattern NewlinePattern

Result file : machregex.ok machregex.err

<< APACHE access log >>
  => machregex "^([0-9.:]+)\\s([\\w.-]+)\\s([\\w.-]+)\\s(\\[[^\\[\\]]+\\])\\s\"((?:[^\"]|\")+)\"\\s(\\d{3})\\s(\\d-
((?:[^\"]|\")*)\"$" "^([0-9.:]+)\s" < DATA.LOG

<< MACH trace log >>
  => machregex "^\\[(\\d+[-]\\d+[-]\\d+\\s\\d+[:]\\d+[:]\\d+)+\\s([P][-]\\d+)+\\s([T][-]\\d+)+\\]\\s((?:[^\\0])*)$"

<< syslog >>
  => machregex "^(([a-zA-Z]+)\\s+([0-9]+)\\s+([0-9:]*))\\s(\\S*)\\s+((?:[^\\0])*)$" ".*" < DATA.LOG
```

This is an example of the machregex run screen.

machregex Test

This is a test that parses Syslog data into machregex using regular expressions.

```
[mach@localhost bin]$ machregex "^(([a-zA-Z]+)\\s+([0-9]+)\\s+([0-9:]*))\\s(\\S*)\\s+((?:[^\\0])*)$" ".*" </var/log
machregex "^(([a-zA-Z]+)\\s+([0-9]+)\\s+([0-9:]*))\\s(\\S*)\\s+((?:[^\\0])*)$" ".*" </var/log/syslog
Pattern => (^(([a-zA-Z]+)\s+([0-9]+)\s+([0-9:]*))\s(\S*)\s+((?:[^\0])*)$)
================================================================================
.............
================================================================================
SUCCESS[107] (rc=7)(Aug 19 18:17:01 localhost CRON[6553]: (root) CMD (   cd / && run-parts --report /etc/cron.hourl
)
  ALL (0:110) => [Aug 19 18:17:01 localhost CRON[6553]: (root) CMD (   cd / && run-parts --report /etc/cron.hourly)
]
  0 (0:15) => [Aug 19 18:17:01]
  1 (0:3) => [Aug]
  2 (4:6) => [19]
  3 (7:15) => [18:17:01]
  4 (16:37) => [localhost]
```

```
  5 (38:110) => [CRON[6553]: (root) CMD (   cd / && run-parts --report /etc/cron.hourly)
]
================================================================================
SUCCESS[107] (rc=7)(Aug 19 18:39:01 localhost CRON[6616]: (root) CMD (  [ -x /usr/lib/php5/maxlifetime ] && [ -x /u
[ -d /var/lib/php5 ] && /usr/lib/php5/sessionclean /var/lib/php5 $(/usr/lib/php5/maxlifetime))
)
  ALL (0:232) => [Aug 19 18:39:01 localhost CRON[6616]: (root) CMD (  [ -x /usr/lib/php5/maxlifetime ] && [ -x /usr
[ -d /var/lib/php5 ] && /usr/lib/php5/sephp5/maxlifetime))
]
  0 (0:15) => [Aug 19 18:39:01]
  1 (0:3) => [Aug]
  2 (4:6) => [19]
  3 (7:15) => [18:39:01]
  4 (16:37) => [localhost]
  5 (38:232) => [CRON[6616]: (root) CMD (  [ -x /usr/lib/php5/maxlifetime ] && [ -x /usr/lib/php5/sessionclean ] &&
]
Summary : Success(107), Failure(0) <== It shows that all of them were successfully completed.
```

In the above example, machregex parses the syslog text file into the given regular expression and splits it into six tokens. To use 0, 4, or 5 of these tokens as database input, use the COL_LIST variable in the template file to associate the token with the database column.

Example of Creating Custom Template

In this chapter, we will use a sample text log file to create a collector template that collects data from this file.

test.log

The input sample text file looks like this:

```
[2014-08-18 13:51:19] spiderman message-1 : This is the best machine data DBMS ever.
[2014-08-18 13:51:19] superman  message-2 : This is the best machine data DBMS ever.
[2014-08-18 13:51:33] spiderman message-3 : This is the best machine data DBMS ever.
[2014-08-18 13:51:33] superman  message-4 : This is the best machine data DBMS ever.
[2014-08-18 13:51:34] batman    message-5 : This is the best machine data DBMS ever.
[2014-08-18 13:52:34] superman  message-6 : This is the best machine data DBMS ever.
[2014-08-18 13:53:34] batman    message-7 : This is the best machine data DBMS ever.
[2014-08-18 13:54:31] superman  message-8 : This is the best machine data DBMS ever.
[2014-08-18 13:55:30] batman    message-9 : This is the best machine data DBMS ever.
[2014-08-18 13:56:44] spiderman message-10 : This is the best machine data DBMS ever.
[2014-08-18 13:57:59] superman  message-11 : This is the best machine data DBMS ever.
```

The above sample file can be converted into three columns: tm, user, and msg. The data type of each column can be specified as datetime, varchar (16), varchar (512).

Example of Creating Regular Expression

Creating Regular Expression

`\[([0-9-: ]+)\]`: First, date data enclosed in square brackets comes in. The following expressions are used to retrieve only the numeric values inside the tokens except for the square brackets.
`(\S+)`: Second, user name data comes in, and strings excluding blanks are input.
`([^\0]*)`: Third, string is entered to the end.
`\[([0-9-: ]+)\]\s(\S+)\s+([^\0]*)`: Combines the space between the three tokens.
`"\\[([0-9-: ]+)\\]\\s(\\S+)\\s+([^\\0]*)"`: Processes double slashing to use strings in the shell.
`"^\\["`: New line regular expression is a square bracket at the beginning of time.

Checking Regular Expression

```
[mach@localhost ~/mach_collector_home/bin]$ machregex "\\[([0-9-: ]+)\\]\\s(\\S+)\\s+([^\\0]+)" "\\[" <test.log
Pattern => (\[([0-9-: ]+)\]\s(\S+)\s+([^\0]+))
================================================================================
SUCCESS[2] (rc=4)([2014-08-18 13:51:19] spiderman message-1 : This is the best machine data DBMS ever.
)
  ALL (0:85) => [[2014-08-18 13:51:19] spiderman message-1 : This is the best machine data DBMS ever.
```

```
]
  0 (1:20) => [2014-08-18 13:51:19]
  1 (22:31) => [spiderman]
  2 (32:85) => [message-1 : This is the best machine data DBMS ever.
]
================================================================================
SUCCESS[3] (rc=4)([2014-08-18 13:51:19] superman  message-2 : This is the best machine data DBMS ever.
)
  ALL (0:85) => [[2014-08-18 13:51:19] superman  message-2 : This is the best machine data DBMS ever.
]
  0 (1:20) => [2014-08-18 13:51:19]
  1 (22:30) => [superman]
  2 (32:85) => [message-2 : This is the best machine data DBMS ever.
]
================================================================================
SUCCESS[4] (rc=4)([2014-08-18 13:51:33] spiderman message-3 : This is the best machine data DBMS ever.
)
  ALL (0:85) => [[2014-08-18 13:51:33] spiderman message-3 : This is the best machine data DBMS ever.
]
  0 (1:20) => [2014-08-18 13:51:33]
  1 (22:31) => [spiderman]
  2 (32:85) => [message-3 : This is the best machine data DBMS ever.
]
================================================================================
SUCCESS[5] (rc=4)([2014-08-18 13:51:33] superman  message-4 : This is the best machine data DBMS ever.
)
  ALL (0:85) => [[2014-08-18 13:51:33] superman  message-4 : This is the best machine data DBMS ever.
]
  0 (1:20) => [2014-08-18 13:51:33]
  1 (22:30) => [superman]
  2 (32:85) => [message-4 : This is the best machine data DBMS ever.
]
================================================================================
SUCCESS[6] (rc=4)([2014-08-18 13:51:34] batman    message-5 : This is the best machine data DBMS ever.
)
  ALL (0:85) => [[2014-08-18 13:51:34] batman    message-5 : This is the best machine data DBMS ever.
]
  0 (1:20) => [2014-08-18 13:51:34]
  1 (22:28) => [batman]
  2 (32:85) => [message-5 : This is the best machine data DBMS ever.
]
================================================================================
SUCCESS[7] (rc=4)([2014-08-18 13:52:34] superman  message-6 : This is the best machine data DBMS ever.
)
  ALL (0:85) => [[2014-08-18 13:52:34] superman  message-6 : This is the best machine data DBMS ever.
]
  0 (1:20) => [2014-08-18 13:52:34]
  1 (22:30) => [superman]
  2 (32:85) => [message-6 : This is the best machine data DBMS ever.
]
================================================================================
SUCCESS[8] (rc=4)([2014-08-18 13:53:34] batman    message-7 : This is the best machine data DBMS ever.
)
  ALL (0:85) => [[2014-08-18 13:53:34] batman    message-7 : This is the best machine data DBMS ever.
]
  0 (1:20) => [2014-08-18 13:53:34]
  1 (22:28) => [batman]
  2 (32:85) => [message-7 : This is the best machine data DBMS ever.
]
================================================================================
SUCCESS[9] (rc=4)([2014-08-18 13:54:31] superman  message-8 : This is the best machine data DBMS ever.
)
  ALL (0:85) => [[2014-08-18 13:54:31] superman  message-8 : This is the best machine data DBMS ever.
]
  0 (1:20) => [2014-08-18 13:54:31]
  1 (22:30) => [superman]
  2 (32:85) => [message-8 : This is the best machine data DBMS ever.
]
================================================================================
SUCCESS[10] (rc=4)([2014-08-18 13:55:30] batman    message-9 : This is the best machine data DBMS ever.
```

```
)
  ALL (0:85) => [[2014-08-18 13:55:30] batman    message-9 : This is the best machine data DBMS ever.
]
  0 (1:20) => [2014-08-18 13:55:30]
  1 (22:28) => [batman]
  2 (32:85) => [message-9 : This is the best machine data DBMS ever.
]
================================================================================
SUCCESS[11] (rc=4)([2014-08-18 13:56:44] spiderman message-10 : This is the best machine data DBMS ever.
)
  ALL (0:86) => [[2014-08-18 13:56:44] spiderman message-10 : This is the best machine data DBMS ever.
]
  0 (1:20) => [2014-08-18 13:56:44]
  1 (22:31) => [spiderman]
  2 (32:86) => [message-10 : This is the best machine data DBMS ever.
]
================================================================================
SUCCESS[11] (rc=4)([2014-08-18 13:57:59] superman  message-11 : This is the best machine data DBMS ever.)
  ALL (0:85) => [[2014-08-18 13:57:59] superman  message-11 : This is the best machine data DBMS ever.]
  0 (1:20) => [2014-08-18 13:57:59]
  1 (22:30) => [superman]
  2 (32:85) => [message-11 : This is the best machine data DBMS ever.]
Summary : Success(11), Failure(0)
```

Creating test.rgx

After checking that the generated regular expression is parsed normally through the above process, if there is no problem in parsing, write rgx file for regular expression and column binding as follows. This file is written in `$MACHBASE_HOME/collector/samples/test.rgx`.

```
################################################################################
# Copyright of this product 2013-2023,
# Machbase Corporation (Incorporation) or its subsidiaries.
# All Rights reserved
################################################################################

#
# This file is for Machbase trace collector regex file.
#

LOG_TYPE=custom

COL_LIST= (
     (
        REGEX_NO = 0
        NAME = tm
        TYPE = datetime
        SIZE = 8
        DATE_FORMAT="%Y-%m-%d %H:%M:%S"
         ),
     (
        REGEX_NO = 1
        NAME = user
        TYPE = varchar
        SIZE = 16
        USE_INDEX = 1
         ),
     (
        REGEX_NO = 2
        NAME = msg
        TYPE = varchar
        SIZE = 512
        USE_INDEX = 1
         )
)

REGEX="\[([0-9-: ]+)\]\s(\S+)\s+([^\0]+)"
```

```
END_REGEX="\["
```

Creating test.tpl

`$MACHBASE_HOME/collector/custom.tpl` is copied to the `$MACHBASE_HOME/collector/test.tpl` name and modifies the file as follows:

```
##############################################################################
# Copyright of this product 2013-2023,
# Machbase Corporation(Incorporation) or its subsidiaries.
# All Rights reserved
##############################################################################

#
# This file is for Machbase collector template file.
#

########################################################################
# Collect setting
########################################################################

COLLECT_TYPE=FILE
LOG_SOURCE=/home/mach/machbase_home/collector/samples/test.log

########################################################################
# Process setting
########################################################################

REGEX_PATH=/home/mach/machbase_home/collector/samples/test.tpl

########################################################################
# Output setting
########################################################################

DB_TABLE_NAME = "custom_table"
DB_ADDR       = "127.0.0.1"
DB_PORT       = 5656
DB_USER       = "SYS"
DB_PASS       = "MANAGER"

# 0: Direct insert
# 1: Prepared insert
# 2: Append
APPEND_MODE=2

# 0: None, just append.
# 1: Truncate.
# 2: Try to create table. If table already exists, warn it and proceed.
# 3: Drop and create.
CREATE_TABLE_MODE=2

Create and Execute a Collector
```

Create/Run Collector

Create a "myclt" collector and run it.

```
Mach> create collector localhost.myclt from "/home/mach/mach_collector_home/collector/samples/test.tpl";
Created successfully.
Elapsed Time : 0.106
Mach>
Mach> alter collector localhost.myclt start;
Altered successfully.
```

Debugging Collector

TESTTABLE was not created to record the input data.

```
Mach> select * from custom_table;
[ERR-02025 : Table CUSTOM_TABLE does not exist.]
```

Writes the error of the collector to the trace file and generates trace file to solve the error. Execute the following command to create a trace file.

```
Mach> alter collector localhost.myclt stop;
Altered successfully.
Mach> alter collector localhost.myclt start trace;
Altered successfully.
```

Problem Detection/Resolution Through Trace Log

If there is an error when running the Collector, you can look for the `$MACHBASE_HOME/trc/machbase.trc` file and look for database execution errors. If an error occurs in the collector, you must run collector in TRACE mode.

```
[2016-03-13 23:44:35 P-29741 T-139982693979904][INFO] PREPARE Error [create table custom_table ( collector_type var
collector_offset long, tm datetime, user varchar(16), msg varchar(512))] (100007DA:Error in parse (syntax): near to
```

Looking at the above message, the table creation query failed because the user set to the column name is not a built-in keyword and can not be used as a column name. Therefore, in the COL_LIST section of the rgx file, change the user column to myuser and run the collector again.

```
A partial contents from "test.rgx"
...........

COL_LIST= (
     (
        REGEX_NO = 0
        NAME = tm
        TYPE = datetime
        SIZE = 8
        DATE_FORMAT="%Y-%m-%d %H:%M:%S"
         ),
     (
        REGEX_NO = 1
        NAME = myuser   <== Modified part
        TYPE = varchar
        SIZE = 16
        USE_INDEX = 1
         ),
     (
        REGEX_NO = 2
        NAME = msg
        TYPE = varchar
        SIZE = 512
        USE_INDEX = 1
         )
)
.................
```

Check Run/Results

Rerun it with the modified rgx file.

```
Mach> alter collector localhost.myclt stop; <== Stop the TRACE mode.
Altered successfully.
Mach> alter collector localhost.myclt start; <== Execute it again in a normal mode after the modification
Altered successfully.
```

If executed normally, the collector can query the contents of the table in which the data is stored.

```
Mach> select tm, myuser, msg from custom_table;
tm                              myuser
------------------------------------------------------
msg
```

```
--------------------------------------------------------------------------------
2014-08-18 13:57:59 000:000:000 superman
message-11 : This is the best machine data DBMS ever.

2014-08-18 13:56:44 000:000:000 spiderman
message-10 : This is the best machine data DBMS ever.

2014-08-18 13:55:30 000:000:000 batman
message-9 : This is the best machine data DBMS ever.

2014-08-18 13:54:31 000:000:000 superman
message-8 : This is the best machine data DBMS ever.

2014-08-18 13:53:34 000:000:000 batman
message-7 : This is the best machine data DBMS ever.

2014-08-18 13:52:34 000:000:000 superman
message-6 : This is the best machine data DBMS ever.

2014-08-18 13:51:34 000:000:000 batman
message-5 : This is the best machine data DBMS ever.

2014-08-18 13:51:33 000:000:000 superman
message-4 : This is the best machine data DBMS ever.

2014-08-18 13:51:33 000:000:000 spiderman
message-3 : This is the best machine data DBMS ever.

2014-08-18 13:51:19 000:000:000 superman
message-2 : This is the best machine data DBMS ever.

2014-08-18 13:51:19 000:000:000 spiderman
message-1 : This is the best machine data DBMS ever.

[11] row(s) selected.
```

Collector Preprocessing Framework

The Machbase collector collects log data, analyzes it, and sends it to the Machbase server.

For additional data processing in addition to data collection and analysis, the Machbase collector provides a data preprocessing framework using python.

Index

Environment Variables for Preprocessing Configuration

We use python version 2.6 as a preprocessing framework. It is recommended that this version of python be installed with the Machbase server.

Installed python is in the `$MACH_COLLECTOR_HOME/webadmin/flask/Python/bin` path.
Running python for additional installation of the python library should also be done in the above directory to avoid conflicts with other existing versions of python.

> To use python, which is provided with the Machbase collector by default, you must set the PATH environment variable correctly and set USER_PREPROCESS_LIB_PATH. When you add an additional path to USER_PREPROCESS_LIB_PATH, you must put a ":" character between paths to separate the path values.

Preprocessing Order

Describes a preprocessing execution order for converting and manipulating log data.

1. Message Preprocessing

When data is input to the original log data file, each log data is separated into log units.

Let's say the log data is origin_msg. Each origin_msg is processed one step at a time.

- For example, if the first message entered is **"Aug 19 15:37:12 localhost NetworkManager [1340]: (eth1): bringing up device."**, the input origin_msg is separated into tokens by a regular expression. This is called message parsing.
- You can preprocess origin_msg before message parsing.
- If you change the origin_msg using the preprocessing script, the modified result message must be a message able to be parsed.

2. Column Preprocessing

After parsing the log message, the resulting token values are generated. If there is no processing, this value is stored in the database.

The preprocessing of the second stage can be executed before passing the parsed tokens to the database.
At this time, the field name described in the rgx file can be used to change or use it. Changing the token to a different type than the data type described in the rgx file may cause an error.
For more information, refer to the example script below.

Preprocessing Script

The preprocessing script must be written in Python. For ease of use, we recommend that you change the "custom.py" file to the desired format.

```
PRS_SUCCESS       = ( 0, None )
PRS_SUCCESS_INFO  = ( 1, "Info Msg")
PRS_SUCCESS_SKIP  = ( 2, None )
PRS_FAILURE       = (-1, "Error Msg" )

class mach_preprocess:
    def __init__(self):
        return
    def mach_msg_preprocess(self, dict):
        return PRS_SUCCESS;
    def mach_column_preprocess(self, dict):
        return PRS_SUCCESS;
    def __del__(self):
        return
```

Defining Result Value

The return value is used to pass the execution result of the preprocessing script to the collector. The result values (code, message) that the collector refers to after executing the preprocessing script are of the tuple type.

```
PRS_SUCCESS       = ( 0, None )
PRS_SUCCESS_INFO  = ( 1, "Info Msg")
PRS_SUCCESS_SKIP  = ( 2, None )
PRS_FAILURE       = (-1, "Error Msg" )
```

Here PRS_SUCCESS, PRS_SUCCESS_INFO, PRS_SUCCESS_SKIP, PRS_FAILURE are the results that the collector refers to.

- If the result is PRS_SUCCESS, the collector normally enters data.
- If the result is PRS_SUCCSS_INFO, the data is processed normally and the transferred message is written to the trc file.
- If the result is PRS_SUCCESS_SKIP, the data is discarded and new data processing is started.
- If the result is PRS_FAILURE, an error message is written to the trc file and the next message is processed.

PRS_SUCCESS_SUCCESS and PRS_SUCCESS_SKIP result value are used for data processing control, and PRS_SUCCESS_INFO or PRS_FAILURE result value is used for leaving message in trc.

Class Definition

The Machbase collector performs preprocessing by calling functions of predefined classes written in the python language.

The following example shows each function in the class and the "dict" parameter and return value.

> Will not run if the class name or function name is changed. Therefore, it must be created with caution.

```
class mach_preprocess:
    def __init__(self):
        return
    def mach_msg_preprocess(self, dict):
        return PRS_SUCCESS;
    def mach_column_preprocess(self, dict):
        return PRS_SUCCESS;
    def __del__(self):
        return
```

The predefined class name is "mach_preprocess".

- Parameters are passed as "self" instances when the method is called.
- __init__ and __del__ are default object constructors/removers for the python language.
  So __init__ is called when the collector's process is created, and __del__ is called when the collector is finished.
- You can initialize variables in __init__ and release the resources allocated by __del__.
  These two methods have no return value.

The methods called for data preprocessing are "mach_msg_preprocess" and "mach_column_preprocess".
Each method is described below.

mach_msg_preprocess

This method is called before the input message is separated into tokens.

Since the message is executed before parsing, the value passed is the collector related metadata and the original message "origin_msg".
The collector metadata is the table name, the collector type, and the name and offset of the currently running collector. This information is provided as reference information and is not reflected in the collector even if it is changed.
When "origin_msg" is changed, the changes are reflected in the collector.
If you change the message so that it does not pass the regular expression set in the rgx file, errors may occur during parsing.

Key	Description	Can Changes Be Reflected
table_name	Table name	X
collect_type	Collector type	X
collector_name	Name of currently running collector	X
data_source	Source file path to be data source	X
origin_msg	Raw data message of source file	O

An unnecessary message can return PRS_SUCCESS_SKIP, which can be processed faster by omitting the parsing process later.

If you can identify an unwanted message at this stage, you should first treat it as PRS_SUCCESS_SKIP.

mach_column_preprocess

This method is called before parsing the input message and entering the token decomposed value into the database.

Like "mach_msg_preprocess", the transferred metadata is not reflected in the collector.

Key	Description	Can Changes Be Reflected
table_name	Table name	X
collect_type	Collector type	X
collect_name	Name of currently running collector	X
data_source	Source file path to be data source	X
origin_msg	Raw data message of source file	X
column_name	nth column token	O

Example Script

The default example is for syslog files and is in the `$MACH_COLLECTOR_HOME/collector` directory.

Let's look at how to perform preprocessing in the example template syslog.tpl.

```
##############################################################################
Copyright of this product 2013-2023,
Machbase Inc. or its subsidiaries.
All Rights reserved
##############################################################################
#
This file is for Machbase collector template file.
#
#########################################################################
Collect setting
#########################################################################

COLLECT_TYPE=FILE
LOG_SOURCE=/var/log/syslog

#########################################################################
Process setting
#########################################################################

REGEX_PATH=syslog.rgx
PREPROCESS_PATH=script_path

#########################################################################
Output setting
#########################################################################
DB_TABLE_NAME = "syslogtable"
DB_ADDR = "127.0.0.1"
DB_PORT = 5656
DB_USER = "SYS"
DB_PASS = "MANAGER"
#
0: Direct insert
1: Prepared insert
2: Append
APPEND_MODE=2
#
0: None, just append.
1: Truncate.
2: Try to create table. If table already exists, warn it and proceed.
3: Drop and create.
CREATE_TABLE_MODE=2
```

To specify the location of the preprocessing script file, set PREPROCESS_PATH in the tpl file. The pathname specifies an absolute path (a path starting with /) or a default path (if only a file name is specified) of $MACH_COLLECTOR_HOME/collector/preprocess.

SKIP

This is a script that examines an input message and does not enter a specific word if it exists.

You can set PREPROCESS_PATH = skip.py in the collector template file.
Since a pathname was not specified, the file needs to be created in the `$MACH_COLLECTOR_HOME/collector/preprocess/` directory.

```
PRS_SUCCESS       = ( 0, None )
PRS_SUCCESS_INFO  = ( 1, "Info Msg" )
PRS_SUCCESS_SKIP  = ( 2, None )
PRS_FAILURE       = (-1, "Error Msg" )

class mach_preprocess:
    def __init__(self):
        return
    def mach_msg_preprocess(self, dict):
        if dict['origin_msg'].find("CMD") is not -1: <== String search "CMD"
            return PRS_SUCCESS_SKIP <== Omit if string does not contain "CMD"
        else:
            return PRS_SUCCESS;
    def mach_column_preprocess(self, dict):
        return PRS_SUCCESS;
    def __del__(self):
        return

#Test code
if __name__ == "__main__":
    pre_obj = mach_preprocess()
    dict = {"origin_msg":"Jul 16 07:09:01 mach-Precision-T1700 CRON[1220]: (root) CMD (  [ -x /usr/lib/php5/maxlife
[ -d /var/lib/php5 ] && /usr/lib/php5/sessionclean /var/lib/php5 $(/usr/lib/php5/maxlifetime))"}
    print pre_obj.mach_msg_preprocess(dict)

    dict = {"origin_msg":"Jul 16 07:39:31 mach-Precision-T1700 cracklib: no dictionary update necessary."}
    print pre_obj.mach_msg_preprocess(dict)
```

This is an example of setting the "origin_msg" parameter in the "mach_msg_preprocess" method, which has not been parsed, to skip the message if it checks for the "CMD" string.

The source line after "if __name__ ==" __main__ " is the code written to test whether the script works properly.
For more information, see the 'Preprocessor Script Test' below.

REPLACE

This is an example of converting the string "CRON" to the string "cron-exectue" if the msg column is parsed after parsing.

This is also done by specifying the `$MACH_COLLECTOR_HOME/collector/preprocess/` directory's replace.py file in the tpl file as:

sample.tpl

```
PREPROCESS_PATH=replace.py
```

```
PRS_SUCCESS       = ( 0, None )
PRS_SUCCESS_INFO  = ( 1, "Info Msg" )
PRS_SUCCESS_SKIP  = ( 2, None )
PRS_FAILURE       = (-1, "Error Msg" )

class mach_preprocess:
    def __init__(self):
        return
    def mach_msg_preprocess(self, dict):
        return PRS_SUCCESS;
    def mach_column_preprocess(self, dict):
        dict['msg'] = dict['msg'].replace("CRON", "cron-execute") <== Replace "CRON" with "cron-execute"
        return PRS_SUCCESS;
    def __del__(self):
        return

#Test code
```

```
if __name__ == "__main__":
    pre_obj = mach_preprocess()
    dict = {"tm":"Jul 16 07:39:01", "host":"mach-Precision-T1700", "msg":"CRON[1377]: (root) CMD (  [ -x /usr/lib/p
[ -x /usr/lib/php5/sessionclean ] && [ -d /var/lib/php5 ] && /usr/lib/php5/sessionclean /var/lib/php5 $(/usr/lib/ph
    (code, msg) = pre_obj.mach_column_preprocess(dict);
    if code >= 0:
        print dict
    else:
        print msg
```

The original message is parsed and separated into tokens, which can be processed by the mach_column_ preprocess method.

The above example is an example of converting "CRON" string to "cron-execute". The code following "if name ==" __main__ " is for debugging script execution.

TRACE

The TRACE script writes the input data to a file in the "mach_msg_preprocess" and "mach_column_preprocess" methods.

It works by adding PREPROCESS_PATH = trace.py to the tpl file and writing the script file to the `$MACH_COLLECTOR_HOME/collector/preprocess` directory.

```
PRS_SUCCESS       = ( 0, None )
PRS_SUCCESS_INFO  = ( 1, "Info Msg" )
PRS_SUCCESS_SKIP  = ( 2, None )
PRS_FAILURE       = (-1, "Error Msg" )

class mach_preprocess:
    def __init__(self):
        self.msg_file = open("/tmp/msg.log", 'a') <== Specify file
        self.column_file = open("/tmp/column.log", 'a')
        return
    def mach_msg_preprocess(self, dict);
        self.msg_file.write(str(dict)+"\n"); <== Write factor value to file
        self.msg_file.write("\n");
        return PRS_SUCCESS;
    def mach_column_preprocess(self, dict):
        self.column_file.write(str(dict)+"\n");
        self.column_file.write("\n");
        return PRS_SUCCESS;
    def __del__(self):
        self.msg_file.close() Close file
        self.column_file.close()
        return

#Test code
if __name__ == "__main__":
    pre_obj = mach_preprocess()
    dict = {"origin_msg":"Jul 16 06:39:01 mach-Precision-T1700 CRON[1149]: (root) CMD (  [ -x /usr/lib/php5/maxlife
[ -x /usr/lib/php5/sessionclean ] && [ -d /var/lib/php5 ] && /usr/lib/php5/sessionclean /var/lib/php5 $(/usr/lib/ph
    pre_obj.mach_msg_preprocess(dict)
    dict = {"origin_msg":"Jul 16 06:39:01 mach-Precision-T1700 CRON[1149]: (root) CMD (  [ -x /usr/lib/php5/maxlife
[ -d /var/lib/php5 ] && /usr/lib/php5/sessionclean /var/lib/php5 $(/usr/lib/php5/maxlifetime))", "tm":"Jul 16 06:39
"msg":"CRON[1149]: (root) CMD (  [ -x /usr/lib/php5/maxlifetime ] && [ -x /usr/lib/php5/sessionclean ] && [ -d /var
/usr/lib/php5/sessionclean /var/lib/php5 $(/usr/lib/php5/maxlifetime))"}
    pre_obj.mach_column_preprocess(dict)
```

You can create and open the msg_file, column_file object when the collector is executed, and close each file at the end using the point where init and del are executed at the start and end.

These two variables can be accessed by other methods of the object. The code following "if __name__ == "__main__" tests the script to see if it works properly.

ODBC

The following ODBC script retrieves the search key from the database and enters the value into the specified table if the search key is present in the input message.

The way to run this example is the same as the previous one by assigning the **PREPROCESS_PATH** value to the template file.
In this example, pypyodbc is used. To use ODBC in the collector script, you must use pypyodbc.

> The imported pypyodbc is not a basic module and needs to be installed in advance.
>
> You should install pypyodbc in the `$MACH_COLLECTOR_HOME/webadmin/flask/Python/bin/python` path.
> The installation path is `$MACH_COLLECTOR_HOME/webadmin/flask/Python/lib/python2.7/site-packages/pypyodbc-1.3.3-py2.7.egg`.

Then you must set the path of the import module. There are two ways to set the path.

- The first way is to modify the path of the sys module or provide the module path to the collector.
- The second way is to set the environment variable **USER_PREPROCESS_LIB_PATH** .

```
export USER_PREPROCESS_LIB_PATH=$MACH_COLLECTOR_HOME/webadmin/flask/Python/lib/python2.7/site-packages/p
```

```
import pypyodbc

PRS_SUCCESS       = ( 0, None )
PRS_SUCCESS_INFO  = ( 1, "Info Msg" )
PRS_SUCCESS_SKIP  = ( 2, None )
PRS_FAILURE       = (-1, "Error Msg" )

class mach_preprocess:
    def __init__(self):
        self.con = pypyodbc.connect("DSN=MYSQL") <== Input pre-declared MySQL DNS value
        self.cursor = self.con.cursor()
        self.table_name = "error_msg"
        self.test_data_make(); <== Generate random data
        return
    def mach_msg_preprocess(self, dict):
        return PRS_SUCCESS;
    def mach_column_preprocess(self, dict):
        result = self.cursor.execute("select code, msg from %s where code = %d"%(self.table_name, int(dict['code_ty
        if result is not None: <== When result value exists
            dict['code_type'] = result.fetchall()[0][1] <== Replace related data
        else:
            print "failure "+str(dict)
        return PRS_SUCCESS;
    def __del__(self):
        self.cursor.close()
        self.con.close()
        return

    #for test
    def test_data_make(self):
        self.table_check("create table %s (code integer, msg varchar(255))");
        return
    def table_check(self, query):
        self.tables = self.cursor.tables().fetchall()
        self.table_list = []
        for (db, user, table, info, none) in self.tables:
            self.table_list.append(table.upper())
        if self.table_name.upper() in self.table_list: <== Delete table and create new one if table already exists
            self.cursor.execute("drop table %s"%self.table_name)
        self.cursor.execute(query%self.table_name);
        self.insert_error_msg()
        self.cursor.commit()
        return
    def insert_error_msg(self): <== Replace code and message
        error = ((0, "SUCCESS"), (1, "SUCCESS_WITH_INFO"), (-1, "FAILURE"))
        for (code, msg) in error:
            self.cursor.execute("insert into %s values ( %d, '%s')"%(self.table_name, code, msg))
        return

# Test code
if __name__ == "__main__":
    pre_obj = mach_preprocess()
    pre_obj.test_data_make()
    dict = {"tm":"Jul 16 07:39:01","host":"mach-Precision-T1700","msg":"CRON[1377]: (root) CMD (  [ -x /usr/lib/php
[ -x /usr/lib/php5/sessionclean ] && [ -d /var/lib/php5 ] && /usr/lib/php5/sessionclean /var/lib/php5 $(/usr/lib/pl
    pre_obj.mach_column_preprocess(dict)
    print dict
    dict = {"tm":"Jul 15 11:31:54","host":"mach-Precision-T1700","msg":"NetworkManager[1340]: <error> [1405391514.2
nm_system_iface_get_flags(): (unknown): failed to get interface link object","code_type":"0"}
    pre_obj.mach_column_preprocess(dict)
```

```
    print dict
    dict = {"tm":"Jul 15 11:31:54","host":"mach-Precision-T1700","msg":"NetworkManager[1340]: <warn> sysctl: faile
open '/proc/sys/net/ipv6/conf/eth1/use_tempaddr': (2) No file in directory","code_type":"1"}
```

Preprocessing Script Test

The pypyodbc module to be used in the above ODBC example can be loaded from the specified environment variable. After you create a new script, you need to make sure that the script works correctly.

There are two methods for script testing: direct execution and indirect execution.
The direct method is that the script executes the test directly, and the indirect method is to import and test the script.

Direct Execution

The test code added at the bottom of the preprocessing script example checks the result by calling the preprocessing class object directly. The example below is the test code for the skip.py script.

```
if __name__ == "__main__":
    pre_obj = mach_preprocess()
    dict = {"origin_msg":"Jul 16 07:09:01 mach-Precision-T1700 CRON[1220]: (root) CMD (  [ -x /usr/lib/php5/maxlife
[ -x /usr/lib/php5/sessionclean ] && [ -d /var/lib/php5 ] && /usr/lib/php5/sessionclean /var/lib/php5 $(/usr/lib/ph
    print pre_obj.mach_msg_preprocess(dict)

    dict = {"origin_msg":"Jul 16 07:39:31 mach-Precision-T1700 cracklib: no dictionary update necessary."}
    print pre_obj.mach_msg_preprocess(dict)
```

The test script starts with "__name__ ==" __main__ ":".

If you run this script directly with the Python interpreter, set the __name__ variable to __main__ and run the test code.
When called from the collector, the above test code is not executed.
If you look at the execution process of the test code, you first create pre-process object pre_obj by calling mach_process () function.
At this point, the __init__ method is called. After setting dict, which is a parameter passed to mach_msg_preprocess, call the method to perform the test.
If you need to set the dict value but you do not know what value to pass to the actual script, you can use the "trace.py" script to get the value passed to the method and test it.
The data generated by the trace.py script is written to /tmp/msg.log and /tmp/column.log.

Indirect Execution

This is a method of importing and executing preprocessing script that has already been created by import.

```
import skip <== Import already created script

if __name__ == "__main__":
    pre_obj = skip.mach_preprocess() <== Call mach_preprocess function when creating class
    dict = {"origin_msg":"Jul 16 07:09:01 mach-Precision-T1700 CRON[1220]: (root) CMD (  [ -x /usr/lib/php5/maxlife
[ -d /var/lib/php5 ] && /usr/lib/php5/sessionclean /var/lib/php5 $(/usr/lib/php5/maxlifetime))"}
    print pre_obj.mach_msg_preprocess(dict)

    dict = {"origin_msg":"Jul 16 07:39:31 mach-Precision-T1700 cracklib: no dictionary update necessary."}
    print pre_obj.mach_msg_preprocess(dict)
```

MACHCOLLECTORADMIN

machcollectoradmin is a tool for managing Collector Manger.

Index

Option List

```
[mach@localhost ~]$ machcollectoradmin --help
------------------------------------------------------------------
     machcollector administration tool
     Release Version - x.x.x.official
     Copyright 2014, Machbase Inc. or its subsidiaries
     All Rights Reserved.
------------------------------------------------------------------
<< Available option lists >>
  -u, --startup                         Startup collectormanager.
  -s, --shutdown                        Shutdown collectormanager.
  -d, --destroy                         Destroy meta data.
  -k, --kill                            Terminate collectormanager.
      --showcollectors                  Show collector list.
      --showservers                     Show server list.
  -h, --help                            Print collector argument guide.
      --createcollector=collector_name  Create collector.
      --dropcollector=collector_name    Drop collector.
      --startcollector=collector_name   Start collector.
      --stopcollector=collector_name    Stop collector.
  -m, --template=template_path          Set template path. Template path is rec
  -t, --trace=[0,1]                     Set trace option. Start collector mode
      --createserver=host:port          Manager registers itself to the server.
      --managername=managername         Specify a name of the manager. If not s
```

Running Collector Manager

The Collector Manager can not be run directly, but must be run through machcollectoradmin.

```
[mach@localhost ~]$ machcollectoradmin --startup
------------------------------------------------------------------
     machcollector administration tool
     Release Version - x.x.x.official
     Copyright 2014, Machbase Inc. or its subsidiaries
     All Rights Reserved
------------------------------------------------------------------
Waiting for collectormanager start.
Collectormanager started successfully.
```

Shutting Down Collector Manager

When you stop the Collector Manager, the collector process managed by the collector manager is stopped at the same time.

```
[mach@localhost ~]$ machcollectoradmin --shutdown
------------------------------------------------------------------
     machcollector administration tool
     Release Version - x.x.x.official
     Copyright 2014, Machbase Inc. or its subsidiaries
     All Rights Reserved
------------------------------------------------------------------
CollectorManager server shutdown successfully.
```

Deleting Collector Manager Meta Information

Deletes information about collector, database server, etc. managed by Collector Manager. If the Collector Manager is running, this command is treated as an error.

```
[mach@localhost ~]$ machcollectoradmin --destroy
----------------------------------------------------------------
machcollector administration tool
Release Version - x.x.x.official
Copyright 2014, Machbase Inc. or its subsidiaries
All Rights Reserved.
----------------------------------------------------------------
Destroyed meta data successfully.
```

Force Quit Collector Manager

The Collector Manager stops immediately without waiting for the shutdown process.

```
[mach@localhost ~]$ machcollectoradmin --kill
----------------------------------------------------------------
     machcollector administration tool
     Release Version - x.x.x.official
     Copyright 2014, Machbase Inc. or its subsidiaries
     All Rights Reserved.
----------------------------------------------------------------
Waiting for collectormanager terminated.
Collectormanager terminated successfully.
```

Register Machbase Server

Sets up the Machbase database server connected to the Collector Manager. After that, management through machsql is possible.

```
[mach@localhost ~]$ machcollectoradmin --createserver=127.0.0.1:5757
----------------------------------------------------------------
     machcollector administration tool
     Release Version - x.x.x.official
     Copyright 2014, Machbase Inc. or its subsidiaries
     All Rights Reserved.
----------------------------------------------------------------
Created the server successfully.
```

Naming Collector Manager

When registering the Collector Manager on the Machbase server, you can set the name of the Collector Manager to register.

```
[mach@localhost ~]$ machcollectoradmin --createserver=127.0.0.1:5757 --managername=mach_manager
----------------------------------------------------------------
     machcollector administration tool
     Release Version - x.x.x.official
     Copyright 2014, Machbase Inc. or its subsidiaries
     All Rights Reserved.
----------------------------------------------------------------
Created the server successfully.
```

Displaying List of Registered Machbase Servers

Displays the list of database servers registered in Collector Manager.

```
[mach@localhost ~]$ machcollectoradmin --showservers
------------------------------------------------------------------
     machcollector administration tool
     Release Version - x.x.x.official
     Copyright 2014, Machbase Inc. or its subsidiaries
     All Rights Reserved.
------------------------------------------------------------------
ID   1
NAME 192.168.0.34:5757
ADDR 192.168.0.34
PORT 5757
```

Creating Collector

Creates a collector managed by the Collector Manager.

When you create a collector, you are required to be prompted for a name and the path to the template file.
The name of the collector is entered as the first factor, and the template path must be specified with the -m or --template option.

```
[mach@localhost ~]$ machcollectoradmin --createcollector=syslog --template=/home/hanchi/work/nfx/machbase_home/coll
------------------------------------------------------------------
machcollector administration tool
Release Version - x.x.x.official
Copyright 2014, Machbase Inc. or its subsidiaries
All Rights Reserved.
------------------------------------------------------------------
Created the collector successfully.
```

Specifying Template File

Specifies the path to the template file that is the collector configuration file when creating a collector.

Relative path is relative to `$MACHBASE_HOME/collector/` directory.

```
[mach@localhost ~]$ machcollectoradmin --createcollector=syslog --template=syslog.tpl
------------------------------------------------------------------
     machcollector administration tool
     Release Version - x.x.x.official
     Copyright 2014, Machbase Inc. or its subsidiaries
     All Rights Reserved.
------------------------------------------------------------------
Created the collector successfully.
```

Running Collector

Runs the registered collector process. You must specify the collector name at runtime.

```
[mach@localhost ~]$ machcollectoradmin --startcollector=syslog
------------------------------------------------------------------
     machcollector administration tool
     Release Version - x.x.x.official
     Copyright 2014, Machbase Inc. or its subsidiaries
     All Rights Reserved.
------------------------------------------------------------------
Started the collector successfully.
```

Collector Log Message Configuration

Lets the user generate a log message when executing the collector.

If the value is set to 1, it is generated. If it is set to 0, it is not generated.

```
[mach@localhost ~]$ machcollectoradmin --startcollector=syslog --trace=1
------------------------------------------------------------------
     machcollector administration tool
     Release Version - x.x.x.official
     Copyright 2014, Machbase Inc. or its subsidiaries
     All Rights Reserved.
------------------------------------------------------------------
Started the collector successfully.
```

Stopping Collector

Stops the running collector.

```
[mach@localhost ~]$ machcollectoradmin --stopcollector=syslog
------------------------------------------------------------------
     machcollector administration tool
     Release Version - x.x.x.official
     Copyright 2014, Machbase Inc. or its subsidiaries
     All Rights Reserved.
------------------------------------------------------------------
Stopped the collector successfully.
```

Dropping Collector

Deletes the meta information associated with the collector. You must specify the name of the collector at runtime.

```
[mach@localhost ~]$ machcollectoradmin --dropcollector=syslog
------------------------------------------------------------------
     machcollector administration tool
     Release Version - x.x.x.official
     Copyright 2014, Machbasae Inc. or its subsidiaries
     All Rights Reserved.
------------------------------------------------------------------
Dropped the collector successfully.
```

Displaying Collector List

Displays a list of registered collectors.

```
[mach@localhost ~]$ machcollectoradmin --showcollectors
------------------------------------------------------------------
     machcollector administration tool
     Release Version - x.x.x.official
     Copyright 2014, Machbase Inc. or its subsidiaries
     All Rights Reserved.
------------------------------------------------------------------
ID                  1
NAME                  SYSLOG
TEMPLATE_PATH       syslog.tpl
COLLECT_TYPE        FILE
SOURCE_FILE         /var/log/syslog
LOG_TYPE            syslog
PREPROCESS_PATH
REGEX_PATH          syslog.rgx
REGEX               (([a-zA-Z]+)\s+([0-9]+)\s+([0-9:]*))\s(\S+)\s+([^\n]+)
END_REGEX           \n
LANGUAGE            UTF-8
SLEEP_TIME          1000
PROCESS_BYTE        0
```

```
PROCESS_RECORD      0
PREV_PROCESS_BYTE   0
PREV_PROCESS_RECORD 0
RUN_FLAG            0
```

Remote Collector Node Management

This chapter describes how to manage the Machbase collector through the Machbase server.

Index

Fixed Table

You can check the status of the collector by querying the Fixed table on the Machbase server.

m$sys_collector_sources

This is a table that keeps information of the configuration files managed by the collector manager registered in the Machbase server. The table column information is shown below.

Name	Description
MANAGER_ID	Collection manager identifier
MANAGER_NAME	Collector manager name
SOURCE_TYPE	File type (Template, Regular Expression, and Python script)
SOURCE_PATH	File absolute path
CONTEXT	File contents

Procedure

It is possible to control the collector by a procedure performed by the Machbase server.

INSERT_COLLECTOR_SOURCE

You can use the Insert_collector_source procedure to send configuration files to the collector running on the remote server.

```
EXECUTE INSERT_COLLECTOR_SOURCE ("manager_name", "path", "context");
```

The meaning of each parameter is as follows.

- manager_name: the name of the Collector manager to transfer the file to
- path: the pathname of the file to transfer
- context: the contents of the file

If the file already exists on the remote server, the original file is changed to `Filaname.bak`

RENAME_COLLECTORMANAGER

This is the procedure to change the name of the already registered collector manager.

```
EXECUTE RENAME_COLLECTORMANAGER ("old_name", "new_name");
```

Visualization Tool

- MWA (Machbase Web Analytics)
- Tag Analyzer

MWA (Machbase Web Analytics)

Machbase Web Analytics (MWA) is a Web application developed with Python 2.7 and Flask-based Werkzeug and Jinja2.

Index

Configuration

The MWA uses the 5001 port to communicate with the client. To verify that the port is available, the linux operating system should use the iptables command and windows should consult the firewall settings to resolve the communication problem.
In addition, you must add the $MACHBASE_HOME/lib folder to $LD_LIBRARY_PATH and make sure that the libmachbasecli_dll.so file is in that folder.

How to Run the Server

The MWA server is executed using the $ MACHBASE_HOME/bin/MWAserver script file. This script uses $ MACHBASE_HOME environment variable, so you must set the environment variable. This script supports the START, STOP, RESTART, RESET, and PORT command options. You can run the server with the 'MWAserver start' command. The port command option allows you to configure the MWA to work using a port other than the default port number 5001 used by the MWAserver.

Example:

```
[mach@localhost ]$./MWAserver port 1234
WEBSERVER PORT CHANGED : 1234
```

List of commands

```
[mach@localhost flask]$./MWAserver help
List of commands:
  * MWAserver start; Startup WebServer
  * MWAserver restart; Restart WebServer
  * MWAserver stop; Shutdown WebServer
  * MWAserver reset; Reset WebServer database
  * MWAserver port NUMBER; Change WebServer port.
```

Connect to MWA with Web Browser

You can connect to the MWA by entering the MWA server IP and port number (ex: http: 127.0.0.1: 5001) through your web browser. We recommend the latest Chrome browser as a web browser. If the connection is successful, the login screen will be displayed. The default login and password are "admin"/"machbase".

.

How to Use MWA

The main menus of MWA are Dashboard, Query, Collection, DB Admin, and Preferences.

Overview

User Authentication

There are two types of users in MWA: ADMIN and USER. The ADMIN user account can use all menus and functions, and the USER account can only use functions authorized by the ADMIN user.

Group

To facilitate user authorization and configuration, multiple users can be grouped. You can select the desired group category in the upper left of the screen. Group categories can be set in the "Available Groups" menu of the "Users" menu. The ADMIN account user has full control. Only ADMIN account users can register groups.

Permission

Set permissions for resources such as saved bookmark queries, grids, and dashboards. There are three permissions: ALL, USER, and OWNER.

ALL: All users can access the resource without logging in.

USER: Only the logged in user can access the resource.

OWNER: Only the user who created the resource can access it.

Only a user with ADMIN authority can change without permission unless it is a resource creator.

Changing Server

You can see the list of registered servers in the right corner of the screen, and you can connect to another server according to your selection. The server list shows only the servers that are running normally. You can check the connection server by using the IP address and port number of the server.

Dashboard

MWA has the ability to display query results as grid and charts and to display the results in a dashboard form. To create a dashboard, you first create and register a grid, a chart, create a dashboard, place the charts and grids you want, and register your dashboard. Once you register a dashboard, you can change the original grid or chart, but the charts and grids registered on that dashboard will not change.

The user can create, change, delete or select information associated with the dashboard on the right. You can enter an external page in the dashboard by typing the external url in the "Link URL" field at the bottom right of the dashboard. Each user may have different dashboard entries that can be displayed depending on their permissions, so you need to set the permissions to ALL to display the data without logging in.

Creating and Editing Dashboard

To create a dashboard and add a chart or grid, you can create a row using the "Row +" button on the dashboard editing screen, and delete the row from the bottom using the "Row-" button. If you have created a panel in a row, you can delete the row after deleting the panel. You can add panels to the row using Chart +, Grid +, HTML +, and URL + buttons. You can select the chart or grid you have saved as the contents of the panel. However, charts and grids with input parameters that receive input from the user can not be registered in the panel.

HTML + buttons can be used to display HTML or JavaScript, jquery data using chart.js. If you set the ID attribute, it can be set to the same value as another dashboard component, so you can specify the ID as "_ID".

Caution should be used when accessing URLs because they may not be accessible according to the value of 'X-Frame-Options'. Each panel in the dashboard can be set to refresh.

Chart

SQL queries can be displayed as graphs. You can check the query results in the Result tab of the chart, and set the parameters related to the chart display in the Setting tab, and the resulting chart can be found in the Chart tab.

MWA provides a "builder" to easily generate time series charts. The user can set the time interval to be aggregated, and can set data corresponding to the X and Y axes. If the Y axis value is a number, you can use a function such as SUM or AVERAGE. If it is not a number, it can be displayed using the record count. When you create a time series graph, you create a line chart by default. The default settings can be changed on the Setting tab. To display the grid at the bottom of the chart screen, click the Config button and select the desired column from the Columns tab. When column data is displayed on the grid, the left and right widths are set according to the ratio. For example, if Width is set to 1, 2,1, the screen display ratio is set to 25%, 50%, and 25%.

When executing a query, you can set the input parameters. Input parameters must be registered as Config button in the Variables tab. The type of the parameter can be text, number, date, time, datetime, and select query.

In the query statement, the name of the parameter is enclosed in the {} character. When the query is executed, the parameter is replaced with the value entered by the user. Quotation marks should be used when using strings because they are simply replaced. For SELECT statements, you must separate them with the ',' character in the option.

```
e.g.: SELECT * FROM TEST_TABLE WHERE C3 = '{V3}'
```

The Preview button allows you to display chart data without saving the chart settings. You can save the results as Excel, JSON, csv, or Tableau data extract (TDE) files with the Output button on the View tab.

Grid

Use the Grid to display the query results in a table format. You can also display columns that you do not need in the query results. You can use the Builder button to help you create a WHERE clause. However, this function supports only simple conditional statements. To select only specific columns in the result data, click on the Config button, click the "+" button in the Columns tab, and set the title and output width.

The column width is treated as a percentage rate as described in the chart's grid settings. Query parameter setting also performs the same function as Chart. The same function as the chart is also available to save query results in various files in Output.

Query

SQL

Performs the query to display the results. LIMIT or DURATION related settings are displayed in the right panel.

The query can be saved and replayed. You can also invoke the executed queries from the history window. In the query execution window, you can switch the results to a grid or chart screen. The bookmarked query can be found in the "Bookmark Queries" menu.

You can see the list of tables created in the "Tables" tab and the schema of each table. Clicking on a table name displays some data from that table. Rerun the query every 5 seconds and refresh the results. You can see the query results in the "Results" tab. If the "Input selected text at cursor position" check box is selected, the character string at the clicked position is automatically entered in the query input window. You can save query results to files in various formats in the same way as Grid or Chart.

Table Explorer

The table navigator displays the input status of the table data. The sum of the number of data inputs is obtained by the input time and displayed in a graph form. You can see the number of input records during that period by selecting the area of the graph as drag.

There is ON / OFF button of Zoom mode (used for time range adjustment) in the upper right part of the screen.

Bookmark Queries

This is the management screen of query inquiry bookmarked after execution on SQL screen. If you click the query list, you can see the details of the query. You can also switch to the SQL screen using the "to SQL" button.

Collection

Data Collection

The collector manager and collector list registered in the mark base server are displayed in a tree form.

Template files (.tpl), regular expression files (.rgx), and preprocessing script files (.py) must be located in the following path relative to the MACHBASE_HOME path.

- Template file (.tpl): $MACHBASE_HOME/collector
- Regular expression file (.tpl): $MACHBASE_HOME/collector/regex
- Preprocessing script file (.py): $MACHBASE_HOME/collector/preprocess

On the right side of the table displayed on the screen, the number of records to the right of the eye-shaped icon is displayed. Move the mouse cursor over the eye icon to display the "View Table" window. Click on the eye icon to switch to the "Table Explorer" screen and check the table contents. At the bottom of the table, you can see the execution status of the collector and collector manager, and the data collection rate. To the right of each collector name is an icon that executes a run, stop, or delete command on the collector.

Pressing the "add Manager" button opens a window that runs the collector manager in a separate window. In this window, you can register new collector manager with "Create Manager" button, and there is a button to execute RENAME, DROP, LIST function on the right side. The LIST button displays a list of collectors managed by the collector manager.

A collector managed by a specific collector manager can be created using the "add Manager" button. Select the collector manager, and then enter the name of the collector to be created. The name of the collector must be unique to the collector manager.

Click the "Template" button to display a new window. You can create a new template file using the "New" button. To the right of each template file name is a button for modifying the template file. Using this function, a new template file can be created using the existing template file. Before creating, modifying, and saving template files, make sure that the DB_ADDR and DB_PORT fields are appropriate values. Hover the mouse cursor over each component displayed on the screen to see a description of the component.

Preprocess

This function is used to manage the preprocessing script file. Select the collector manager in the right corner of the screen to display a list of preprocessing script files. Press the "Reload" button to re-read the file list.

You can edit the file by clicking on the script file name, and you can create a new script file using the "New" button. You can use the "Save" button to record the changed or created file.

Regular Expression

This is a function that manages the regular expression file used in the template file. When you select Collector Manager on the right, it displays a list of regular expression files. You can use the "Reload" button to reload the list.

When you click on a file in the list of regular expression files, the file contents are displayed and you can change the file. You can use the "New" button to create a new regular expression file.

The regular expression file must contain the following components:

- REGEX: Regular expression for analyzing data.
- START_REGEX: A regular expression specifying the starting point of the data to be analyzed. This data is included in the REGEX analysis data.
- END_REGEX: A regular expression that indicates the end of the analysis data. The values after this are not included in the analysis. If this value is not specified, the collector will not work.

On the lower screen, a list of columns of the table into which the analysis data will be input is displayed. Regex No is the serial number generated as a result of machregex, followed by the column name, type, and size. You can run the sample test by clicking the Test tab. If the test is successful, select the desired column in the Columns tab. When you run the test, the window shows whether the test run was successful. After success, set the desired column name, type and size, and hit the "Apply" button to display the column list again.

DB Admin

Tables

This is the screen to manage table and table space. You can see the table schema and the table index.

Running Queries

Displays the currently executing queries. You can interrupt a running query.

System Monitoring

Displays information about the system on which the MWA server is running. If information can not be obtained, it is displayed as blank. To monitor a device that is not currently running an MWA, you must run the MWA on the desired device and register the "Web URL" for that device.

Preferences

Servers

This is a menu for registering a Machbase server. Only users with ADMIN authority can access MWA.

Server Name	Description
Host and Port	Sets the IP address and port number of the Machbase DB server. You can use 127.0.0.1 or localhost if you are running a Machbase DB on the same machine as the MWA server.
Web URL	You can connect using this URL.
UserID and Password	User name and password of the Machbase DB server

You must register at least one server. If the machine running MWA server is running Machbase DB, you can execute commands such as server start and stop, DB creation, etc. by using "Command" button. For the Machbase DB server running on another machine, you can use the functions except the command to execute with the "Command" button. If the MWA server is running in the same way as the DB server, commands of the "Command" button can be executed on other apparatuses other than the currently running MWA server.

Groups

Manages MWA user groups. Only MWA users with ADMIN authority are allowed access. Chart, Grid, Dashboard, and Bookmark queries can be registered and managed for each group.

Users

Manages user accounts. This user account refers to the MWA user, not the Machbase DB user. In the case of a dashboard, a screen for selecting a previously created dashboard is displayed.

ADMIN users can register and manage each user and server.

Available Groups: set ADMIN / USER to Authority.
Available Servers: set default server.

Users with the USER privilege can only update the PASSWORD.

Tag Analyzer

The Tag Analyzer provides the ability to query and analyze data by using the ROLLUP function of Tag Table.

Index

Configuration

MWA must be available because it is called in MWA, and the table used is specified in the TAG table.
Among the settings defined in $MACHBASE_HOME/webadmin/flask/MWA.conf file, the items used by Tag Analyzer are as follows.

Item	Description	Default Value
USE_TAG_ANALYZER_AUTO_DRILLDOWN	If no rollup data is found, the data is searched for in the rollup a step below.	Y
MAX_TAG_COUNT	The maximum number of tags available in a chart	12

Start

When you run MWA, click "Tag View" on the top left of the visible menu to run Tag Analyzer on the new tab (window).
If you are logged in to MWA, you can directly connect to the Tag Analyzer using the **MWA URL**/tagview (ex: 127.0.0.1:5001/tagview) or **MWA URL**/tagview?id=*dashboard_id* (ex: 127.0.0 .0.1: 5001/tagview?id=board1) to display the corresponding Dashboard.

The Tag Analyzer consists of a Dashboard consisting of several charts. Each column of the Dashboard consists of one Chart.

Overview

In the top left corner, a menu for managing the Tag Analyzer is displayed.

Select Dashboard

Select the Dashboard that is displayed on the current screen among the saved Dashboards.

New Dashboard

Creates a new Dashboard.

Preference

Sets the environment of the Tag Analyzer.

Item	Description	Default Value
UI Thema	Sets the background color of the Dashboard. (machIoTchartBlack/machIoTchartWhite)	machIoTchartBlack
Home Dashboard	Selects the Dashboard to be selected when Tag Analyzer is called without specifying Dashboard.	없음.(New Dashboard) None (New Dashboard)
Query Timeout	If there is no response within the time set in the Query call, a Timeout error is generated.	20

Manage Dashboard

Manages the saved Dashboard.

Button	Description	Remarks
	Retrieves Dashboard in a new tab (window)	
	Modifies Dashboard Title	
	Deletes Dashboard	

Request Rollup

Executes the command " **EXEC ROLLUP_FORCE** ", which is a ROLLUP forced update command . This takes about 6 seconds.
Logout

Logs out of MWA.

Dashboard

Creating Chart

Each row in the Dashboard consists of a single chart . You can create a chart by pressing the **+** button on the bottom panel.
Select the tags and aggregation method you want to use and select Chart type to create a chart. Click on the Tag in the Selected Tag list to deselect it.

Configuration

The buttons related to Dashboard can be found in the upper right corner.

Button	Description	Remarks
Save Dashboard	Saves the current Dashboard. The Board ID is used in URLs when viewed. If you change the Board ID, a new Dashboard will be created if "Save as Copy" is checked, otherwise the Board ID will be changed.	Board ID can only contain alphanumeric characters and underscore (_)
Time Range	Sets the time range to be inquired and the refresh interval. • Time range: It is possible to specify relative time using time selection (input) or now. (ex: now, now-5d, now-3M) The time specification based on now y (year), M (month), d (day), h (hour), m (minute), and s (second) is possible. • Refresh cycle: It is possible to input in h (hour), m (minute), s (second) unit. If you set the time range or the refresh interval separately in Chart setting, it operates based on its own setting value regardless of Dashboard setting.	YYYY-MM-DD HH24:MI:SS
Refresh all	Refreshes all charts.	
Share Dashboard	Retrieves the current Dashboard in a new tab (window) (open as a URL for viewing)	Only available if saved

Chart

Charts are drawn with selected tags and aggregation method. You can see the time and series value by moving the mouse in the chart.
You can see the graph of the selected tag by clicking on the legend shown in the bottom part, and you can see all series again by clicking the same tag again.

Zoom

If you select the part you want to enlarge in the chart by dragging it with the mouse, you can see the corresponding part in detail. In the lower part of the chart, the whole graph according to Time Range is displayed, and the portion drawn in the current chart is displayed. When the part displayed on the chart is enlarged, the ROLLUP data in seconds is displayed, and if the selected part becomes very small, a chart showing raw data is drawn.

In the case of Raw data chart, a button to move the time to the left and right of the Time Range displayed in the Chart Header is displayed and can be viewed while moving the Chart. Raw data charts are so small that they can not be moved using the viewport.

Auto Refresh does not work when in Zoom mode.

- viewport: The part of the chart where the entire graph is displayed. Displayed only when zoomed in on a part of the chart (starting with Zoom mode is the default). Press [**x**] on the left side of the viewport to exit Zoom mode and close the viewport.
- window: Displays the portion of the viewport currently drawn in the chart. You can move it by dragging with the mouse. You can enlarge or reduce the chart by changing the size by dragging the left and right ends.
- viewport buttons: These buttons are used to move or manipulate the viewport.

Button	Description	Remarks
Time Range	Sets the Time Range using the time and duration.	From & 6M
Undo	Cancels the Zoom operation. The right mouse button also performs the same function.	Currently, **only available with previous Zoom.**
Reset	Sets the Time Range to the minimum to maximum of all data.	
Center	Adjusts the Time Range so that the current window is centered in the viewport.	
Resize	Adjust the Time Range so that the current window is centered at 20%.	
- , +	Zooms in / out the window to the left (right).	Zoom out / Zoom in
<< , < , > , >>	Moves the window 100% (50%) left and right.	

Chart buttons

The buttons on the upper right of the chart are as follows.

Button	Description	Remarks
Preview	Only the corresponding chart is displayed in a new tab (window).	
Edit options	Modifies the properties of the Chart. You can change the shape of the chart or add/modify tags.	Apply ✓ after modifying
Refresh	Draws the chart again.	
Delete	Removes the chart from the Dashboard.	

Chart properties

Property is modified in the bottom panel, press ✓ to confirm that it has been modified, and then click on the top ✓ to apply. The property of each tab is as follows.

General

Property	Description	Default Value	Remarks
Chart Title	Chart Title	Chart Title	
Width	Chart Width (0: Full Size)	0	Currently, there is no meaning except 0
Height	Chart Height (0: Full Size)	300	Currently 0 is calculated as 300
Action On Click	Actions when clicking on the chart node - No Action - Shows Raw data chart - Shows Raw data table	No Action	Show Raw data chart means the chart corresponding to the time range of the clicked node.
Zoom	Whether to zoom when dragging Chart	Y	
Drill down	Whether to drill down when zooming	Y	
Start with Zoom	When you drawing chart, start in Zoom mode.	Y	Since Auto Refresh does not work in Zoom state, this function should be turned off to use Auto Refresh.

Data

Changes the tag to use and the aggregation method.
Tag Names can be tagged as a series by typing multiple tags with ",". If there are several tags corresponding to the time zone, the value of the Tag is selected in alphabetical order.

Axes

Property	Description	Default Value	Remarks
Interval	Sets the time interval on the X axis. If not specified, it is automatically calculated according to the time range and screen size.		Supports h, m, and s
Pixels between tick marks	Sets the pixel between the X axis scale.	3	
The scale of the y-axis start at zero.	Y scale starts from zero.	N	
Custom scale	Sets the range of scale on the Y axis. If not specified, it is specified automatically using minimum value and maximum value.		
Custom scale for drill down chart	Sets the scale of the Y scale of the drill down chart . If not specified, it is specified automatically using minimum value and maximum value.		

Display

Property	Description	Default Value	Remarks
Show data points	Displays the point on the chart's node.	N	

Property	Description	Default Value	Remarks
Point radius	The radius of the displayed point (pixel) in the above case	3	
Legend	Displays legend.	Y	
Opacity of fill area	Sets the transparency of the fill area of the graph.	0.15	0: transparent, 1: opaque
Line thickness	Sets the leading edge of the line chart.	1.5	0: no line
Border color	Sets the border color of the chart. If not input, it is set to Background Color.		Enter "#" when changing and input "none" when changing to input space

Time Range

Enters the Time Range and Refresh period that apply only to this chart. When these values are set, the chart will only work with this value, regardless of the Dashboard settings.

Cluster management tools

- machdeployeradmin
- machcoordinatoradmin

machdeployeradmin

machdeployeradmin

Index

You can check the status of the Deployer, or directly issue the Deployer's startup/shutdown/stop commands.

Normally the fastest way to issue the commands is through machcoordinatoradmin, but if not possible, you must do the following.

Options and Features

The options for machdeployeradmin are as follows. The functions described in the previous section are omitted.

```
mach@localhost:~$ machdeployeradmin -h
```

Options	Description
-u, --startup	Runs Deployer process
-s, --shutdown	Terminates Deployer process
-k, --kill	Stops Deployer process
-c, --createdb	Creates Deployer meta
-d, --destroydb	Deletes Deployer meta
-i, --silence	Runs without output
-e, --check	Checks to see if Deployer process is running

Checking Running Status

Example:

```
mach@localhost:~$ machdeployeradmin -e
--------------------------------------------------------------------------
     Machbase Deployer Administration Tool
     Release Version - e3c0717.develop
     Copyright 2014, MACHBASE Corp. or its subsidiaries
     All Rights Reserved
--------------------------------------------------------------------------
Machbase Deployer is running with pid(29373)!
```

machcoordinatoradmin

Coordinator is a cluster-wide management tool.

Options and Features

The options for machcoordinatoradmin are as follows. The functions described in the previous section are omitted.

```
mach@localhost:~$ machcoordinatoradmin -h
```

Options	Description
-u, --startup	Runs the Coordinator process
-s, --shutdown	Terminates Coordinator process
-k, --kill	Stops Coordinator process
-c, --createdb	Creates Coordinator meta
-d, --destroydb	Removes Coordinator meta, Deletes the package files in $MACHBASE_COORDINATOR_HOME/package
-e, --check	Checks that the Coordinator process is running
-i, --silence	Runs without output
--configuration[=name]	Outputs keys and values in configuration settings (only certain keys can be output)
--activate	Switches Cluster status to Service
--deactivate	Switches Cluster status to Deactivate
--list-package[=package]	Lists information of registered packages (only specific packages can be output)
--add-package=package	Adds package
--remove-package=package	Deletes package
--list-node[=node]	Lists information of nodes (only specific nodes can be output)
--add-node=node	Adds node
--remove-node=node	Deletes node
--upgrade-node=node	Upgrades node
--startup-node=node	Runs node
--shutdown-node=node	Terminates node
--kill-node=node	Stops node
--cluster-status	Outputs each node status of the cluster
--cluster-status-full	Outputs of each node status of cluster in detail
--cluster-node	Outputs information of cluster
--set-group-state=[normal | readonly]	Changes the status of a specific warehouse group
--get-host-resource	Outputs host resource information where each node is located
--host-resource-enable	Starts collecting Host resource information of each node

Options	Description
--host-resource-disable	Stops collecting Host resource information for each node

Additional Options	Description	Required Options
--file-name=filename	File name	--add-package
--port-no=portno	Port number	--add-node
--deployer=node	Deployer node name	--add-node
--package-name=packagename	Package name to be the installation source	--add-package
--home-path=path	Based on Deployer server, installation path of current Node	--add-node
--node-type=[broker | warehouse]	Node type to install (choose between broker/warehouse)	--add-node
--group=groupname	Group name of the node to install	--add-node
--replication=host:port	host: port to exchange replication	--add-node
--no-replicate	Does not use Replication on the node to install	--add-node
--primary=host:port	Specifies the node name of the Primary Coordinator when installing the Secondary Coordinator	-u, --startup
--host=host	Specifies specific host to output Host resource information	--get-host-resource
--metric=[cpu|memory|disk|network]	Specifies specific metric to output Host resource information	--get-host-resource

Check Running Status

Example:

```
mach@localhost:~$ machcoordinatoradmin -e
------------------------------------------------------------------------
     Machbase Coordinator Administration Tool
     Release Version - e3c0717.develop
     Copyright 2014, MACHBASE Corp. or its subsidiaries
     All Rights Reserved
------------------------------------------------------------------------
Machbase Coordinator is running with pid(29245)!
```

Create / Delete Meta

Example:

```
mach@localhost:~$ machcoordinatoradmin -c
------------------------------------------------------------------------
     Machbase Coordinator Administration Tool
     Release Version - e3c0717.develop
     Copyright 2014, MACHBASE Corp. or its subsidiaries
     All Rights Reserved
------------------------------------------------------------------------
Coordinator metadata created successfully.

mach@localhost:~$ machcoordinatoradmin -d
------------------------------------------------------------------------
     Machbase Coordinator Administration Tool
     Release Version - e3c0717.develop
     Copyright 2014, MACHBASE Corp. or its subsidiaries
     All Rights Reserved
------------------------------------------------------------------------
Coordinator metadata destroyed successfully.
```

Output Configuration

Syntax:

```
machcoordinatoradmin --configuration[=name]
```

Example:

```
mach@localhost:~$ machcoordinatoradmin --configuration
--------------------------------------------------------------------------
     Machbase Coordinator Administration Tool
     Release Version - e3c0717.develop
     Copyright 2014, MACHBASE Corp. or its subsidiaries
     All Rights Reserved
--------------------------------------------------------------------------
Name  : CLUSTER
Value : 3

Name  : DECISION
Value : ON

Name  : HOST-RESOURCE
Value : OFF

mach@localhost:~$ machcoordinatoradmin --configuration=decision
--------------------------------------------------------------------------
     Machbase Coordinator Administration Tool
     Release Version - e3c0717.develop
     Copyright 2014, MACHBASE Corp. or its subsidiaries
     All Rights Reserved
--------------------------------------------------------------------------
              Name : DECISION
             Value : ON
            Format : text/plain
```

Change Cluster Status

Example:

```
mach@localhost:~$ machcoordinatoradmin --activate
--------------------------------------------------------------------------
     Machbase Coordinator Administration Tool
     Release Version - e3c0717.develop
     Copyright 2014, MACHBASE Corp. or its subsidiaries
     All Rights Reserved
--------------------------------------------------------------------------
              Name : CLUSTER
             Value : 3
            Format : text/plain


mach@localhost:~$ machcoordinatoradmin --deactivate
--------------------------------------------------------------------------
     Machbase Coordinator Administration Tool
     Release Version - e3c0717.develop
     Copyright 2014, MACHBASE Corp. or its subsidiaries
     All Rights Reserved
--------------------------------------------------------------------------
              Name : CLUSTER
             Value : 0
            Format : text/plain
```

List Package Information

Syntax:

```
machcoordinatoradmin --list-package[=package]
```

Example:

```
mach@localhost:~$ machcoordinatoradmin --list-package
------------------------------------------------------------------------
     Machbase Coordinator Administration Tool
     Release Version - e3c0717.develop
     Copyright 2014, MACHBASE Corp. or its subsidiaries
     All Rights Reserved
------------------------------------------------------------------------
Package Name : machbase
File Name    : machbase-cluster-6bab497c9.develop-LINUX-X86-64-release-lightweight.tgz
File Size    : 64630670 bytes

Package Name : machbase2
File Name    : machbase-cluster-e3c0717.develop-LINUX-X86-64-release-lightweight.tgz
File Size    : 64677030 bytes


mach@localhost:~$ machcoordinatoradmin --list-package=machbase
------------------------------------------------------------------------
     Machbase Coordinator Administration Tool
     Release Version - e3c0717.develop
     Copyright 2014, MACHBASE Corp. or its subsidiaries
     All Rights Reserved
------------------------------------------------------------------------
Package Name : machbase
File Name    : machbase-cluster-6bab497c9.develop-LINUX-X86-64-release-lightweight.tgz
File Size    : 64630670 bytes
```

List Node Information

Syntax:

```
machcoordinatoradmin --list-node[=node]
```

Example:

```
mach@localhost:~$  machcoordinatoradmin --list-node
------------------------------------------------------------------------
     Machbase Coordinator Administration Tool
     Release Version - e3c0717.develop
     Copyright 2014, MACHBASE Corp. or its subsidiaries
     All Rights Reserved
------------------------------------------------------------------------
Node Name             : 192.168.0.32:5101
Node Type             : coordinator
HTTP Admin Port       : 5102
Group Name            : Coordinator
Desired State         : primary
Actual State          : primary
Coordinator Host      : 192.168.0.32:5101
Last Response Time    : 497590
Last Modify Time      : 421020408
Last Response Elapsed : 1006148

Node Name             : 192.168.0.32:5201
Node Type             : deployer
Group Name            : Deployer
```

```
Desired State         : normal
Actual State          : normal
Coordinator Host      : 192.168.0.32:5101
Last Response Time    : 497594
Last Modify Time      : 404915419
Last Response Elapsed : 1006128

Node Name             : 192.168.0.32:5301
Node Type             : broker
Port Number           : 5757
Deployer              : 192.168.0.32:5201
Package Name          : machbase
Home Path             : /home/machbase/broker1
Group Name            : Broker
Desired State         : leader
Actual State          : leader
Coordinator Host      : 192.168.0.32:5101
Last Response Time    : 497544
Last Modify Time      : 353606480
Last Response Elapsed : 1006157

Node Name             : 192.168.0.32:5401
Node Type             : warehouse
Port Number           : 5400
Deployer              : 192.168.0.32:5201
Package Name          : machbase
Home Path             : /home/machbase/warehouse_a1
Group Name            : Group1
Desired State         : normal
Actual State          : normal
Coordinator Host      : 192.168.0.32:5101
Last Response Time    : 497556
Last Modify Time      : 332480933
Last Response Elapsed : 1006160

mach@localhost:~$  machcoordinatoradmin --list-node=192.168.0.32:5401
------------------------------------------------------------------------
     Machbase Coordinator Administration Tool
     Release Version - e3c0717.develop
     Copyright 2014, MACHBASE Corp. or its subsidiaries
     All Rights Reserved
------------------------------------------------------------------------
Node Name             : 192.168.0.32:5401
Node Type             : warehouse
Port Number           : 5400
Deployer              : 192.168.0.32:5201
Package Name          : machbase
Home Path             : /home/cumulus/warehouse_a1
Group Name            : Group1
Desired State         : normal
Actual State          : normal
Coordinator Host      : 192.168.0.32:5101
Last Response Time    : 648879
Last Modify Time      : 419153148
Last Response Elapsed : 1005962
```

Output Cluster Node Status

Example:

```
mach@localhost:~$ machcoordinatoradmin --cluster-status
------------------------------------------------------------------------
     Machbase Coordinator Administration Tool
     Release Version - e3c0717.develop
     Copyright 2014, MACHBASE Corp. or its subsidiaries
     All Rights Reserved
------------------------------------------------------------------------
+-------------+------------------+------------------+------------------+--------------+
```

```
|  Node Type  |     Node Name     |    Group Name     |    Group State    |     State    |
+-------------+-------------------+-------------------+-------------------+--------------+
| coordinator | 192.168.0.32:5101 | Coordinator       | normal            | primary      |
| deployer    | 192.168.0.32:5201 | Deployer          | normal            | normal       |
| broker      | 192.168.0.32:5301 | Broker            | normal            | leader       |
| warehouse   | 192.168.0.32:5401 | Group1            | normal            | normal       |
+-------------+-------------------+-------------------+-------------------+--------------+

mach@localhost:~$ machcoordinatoradmin --cluster-status-full
-------------------------------------------------------------------------
     Machbase Coordinator Administration Tool
     Release Version - e3c0717.develop
     Copyright 2014, MACHBASE Corp. or its subsidiaries
     All Rights Reserved
-------------------------------------------------------------------------
+-------------+-------------------+-------------------+-------------------+-------------------------------+--------
|  Node Type  |     Node Name     |    Group Name     |    Group State    |    Desired & Actual State     |  RP Sta
+-------------+-------------------+-------------------+-------------------+-------------------------------+--------
| coordinator | 192.168.0.32:5101 | Coordinator       | normal            | primary       | primary       | -------
| deployer    | 192.168.0.32:5201 | Deployer          | normal            | normal        | normal        | -------
| broker      | 192.168.0.32:5301 | Broker            | normal            | leader        | leader        | -------
| warehouse   | 192.168.0.32:5401 | Group1            | normal            | normal        | normal        | -------
+-------------+-------------------+-------------------+-------------------+-------------------------------+--------
```

Output Cluster Information

Example:

```
mach@localhost:~$ machcoordinatoradmin --cluster-node
-------------------------------------------------------------------------
     Machbase Coordinator Administration Tool
     Release Version - e3c0717.develop
     Copyright 2014, MACHBASE Corp. or its subsidiaries
     All Rights Reserved
-------------------------------------------------------------------------
Token Pid      : 29245
Token Time     : 1553153902646178
Modify Time    : 1553154010296715
Modify Count   : 8
Cluster Status : Service
Broker         : 192.168.0.32:5301
Warehouse      : 192.168.0.32:5401
```

Change Group State

Syntax:

```
machcoordinatoradmin --set-group-state=[ normal | readonly ] --group=group
```

Example:

```
mach@localhost:~$ machcoordinatoradmin --set-group-state=readonly --group=Group1
-------------------------------------------------------------------------
     Machbase Coordinator Administration Tool
     Release Version - e3c0717.develop
     Copyright 2014, MACHBASE Corp. or its subsidiaries
     All Rights Reserved
-------------------------------------------------------------------------
Group Name: Group1
Flag      : 1

mach@localhost:~$ machcoordinatoradmin --cluster-status
-------------------------------------------------------------------------
     Machbase Coordinator Administration Tool
```

```
     Release Version - e3c0717.develop
     Copyright 2014, MACHBASE Corp. or its subsidiaries
     All Rights Reserved
--------------------------------------------------------------------------
+-------------+------------------+------------------+-------------------+--------------+
|  Node Type  |     Node Name    |    Group Name    |    Group State    |     State    |
+-------------+------------------+------------------+-------------------+--------------+
| coordinator | 192.168.0.32:5101 | Coordinator      | normal            | primary      |
| deployer    | 192.168.0.32:5201 | Deployer         | normal            | normal       |
| broker      | 192.168.0.32:5301 | Broker           | normal            | leader       |
| warehouse   | 192.168.0.32:5401 | Group1           | readonly          | normal       |
+-------------+------------------+------------------+-------------------+--------------+
```

Output Host Resource

Syntax:

```
machcoordinatoradmin --host-resource-enable [--metric=metric] [host=host]
```

Example:

```
mach@localhost:~$ machcoordinatoradmin --host-resource-enable
--------------------------------------------------------------------------
     Machbase Coordinator Administration Tool
     Release Version - e3c0717.develop
     Copyright 2014, MACHBASE Corp. or its subsidiaries
     All Rights Reserved
--------------------------------------------------------------------------
              Name : HOST-RESOURCE
             Value : ON
            Format : text/plain

mach@localhost:~$ machcoordinatoradmin --get-host-resource
--------------------------------------------------------------------------
     Machbase Coordinator Administration Tool
     Release Version - e3c0717.develop
     Copyright 2014, MACHBASE Corp. or its subsidiaries
     All Rights Reserved
--------------------------------------------------------------------------
Host Name : 192.168.0.32
   CPU Info :
      Model Name          : Intel(R) Xeon(R) CPU E3-1231 v3 @ 3.40GHz
      Number of CPUs      : 8
      Number of CPU Cores : 4
      CPU Utilization     : 14.0%
      CPU IOWait Ratio    : 0.0%
   Memory Info :
      Physical Memory Utilization : 99.1%
      Virtual Memory Utilization  : 98.6%
   Network Info :
      Receive Bytes(per second)    : 42809
      Receive Packets(per second)  : 337
      Transmit Bytes(per second)   : 42885
      Transmit Packets(per second) : 332
   Disk Info :
      /dev/sda1 : 87.4%
         |-> 192.168.0.32:5101   /home/cumulus/coordinator1
         |-> 192.168.0.32:5301   /home/cumulus/broker1
         |-> 192.168.0.32:5401   /home/cumulus/warehouse_a1
Host Name : 192.168.0.33
   CPU Info :
      Model Name          : Intel(R) Xeon(R) CPU E3-1231 v3 @ 3.40GHz
      Number of CPUs      : 8
      Number of CPU Cores : 4
      CPU Utilization     : 2.0%
      CPU IOWait Ratio    : 0.0%
```

```
   Memory Info :
      Physical Memory Utilization : 46.9%
      Virtual Memory Utilization  : 22.8%
   Network Info :
      Receive Bytes(per second)    : 12336
      Receive Packets(per second)  : 103
      Transmit Bytes(per second)   : 13500
      Transmit Packets(per second) : 103
   Disk Info :
      /dev/sda1 : 64.2%
         |-> 192.168.0.33:5101   /home/cumulus/coordinator2
         |-> 192.168.0.33:5401   /home/cumulus/warehouse_a2

mach@localhost:~$ machcoordinatoradmin --get-host-resource --metric=cpu
------------------------------------------------------------------------
     Machbase Coordinator Administration Tool
     Release Version - e3c0717.develop
     Copyright 2014, MACHBASE Corp. or its subsidiaries
     All Rights Reserved
------------------------------------------------------------------------
Host Name : 192.168.0.32
   CPU Info :
      Model Name          : Intel(R) Xeon(R) CPU E3-1231 v3 @ 3.40GHz
      Number of CPUs      : 8
      Number of CPU Cores : 4
      CPU Utilization     : 13.9%
      CPU IOWait Ratio    : 0.0%
Host Name : 192.168.0.33
   CPU Info :
      Model Name          : Intel(R) Xeon(R) CPU E3-1231 v3 @ 3.40GHz
      Number of CPUs      : 8
      Number of CPU Cores : 4
      CPU Utilization     : 1.9%
      CPU IOWait Ratio    : 0.0%

mach@localhost:~$ machcoordinatoradmin --get-host-resource --host=192.168.0.33
------------------------------------------------------------------------
     Machbase Coordinator Administration Tool
     Release Version - e3c0717.develop
     Copyright 2014, MACHBASE Corp. or its subsidiaries
     All Rights Reserved
------------------------------------------------------------------------
Host Name : 192.168.0.33
   CPU Info :
      Model Name          : Intel(R) Xeon(R) CPU E3-1231 v3 @ 3.40GHz
      Number of CPUs      : 8
      Number of CPU Cores : 4
      CPU Utilization     : 2.0%
      CPU IOWait Ratio    : 0.0%
   Memory Info :
      Physical Memory Utilization : 46.9%
      Virtual Memory Utilization  : 22.8%
   Network Info :
      Receive Bytes(per second)    : 12588
      Receive Packets(per second)  : 106
      Transmit Bytes(per second)   : 13330
      Transmit Packets(per second) : 100
   Disk Info :
      /dev/sda1 : 64.2%
         |-> 192.168.0.33:5101   /home/cumulus/coordinator2
         |-> 192.168.0.33:5401   /home/cumulus/warehouse_a2

mach@localhost:~$ machcoordinatoradmin --host-resource-disable
------------------------------------------------------------------------
     Machbase Coordinator Administration Tool
     Release Version - e3c0717.develop
     Copyright 2014, MACHBASE Corp. or its subsidiaries
     All Rights Reserved
------------------------------------------------------------------------
              Name : HOST-RESOURCE
```

```
 Value : OFF
Format : text/plain
```

Configuration/Monitoring

This chapter describes the meaning of Machbase properties, how to set them, meta tables, and virtual tables.

Meta Table

The Meta Tables are tables that present the schema information of Machbase. The table names begin with "M$".

These tables hold the table name, column information, and index information, and reflect the creation, modification and deletion information resulting from the DDL statement.
The Meta Tables can not be added, deleted, or changed by the user.

Index

User Objects

M$SYS_TABLES

Displays the table created by the user.

Column Name	Description
NAME	Table name
TYPE	Table type
DATABASE_ID	Database identifier
ID	Table identifier
USER ID	User of created table
COLCOUNT	Number of columns

M$SYS_COLUMNS

Displays the column information of the user table displayed in M$SYS_TABLES.

Column Name	Description
NAME	Column name
TYPE	Column type
DATABASE_ID	Database identifier
ID	Column identifier
LENGTH	Column length
TABLE_ID	Table identifier of column
FLAG	(Information for internal use of the server)
PART_PAGE_COUNT	Pages per partition
PAGE_VALUE_COUNT	Number of data per page
MINMAX_CACHE_SIZE	Size of MIN-MAX cache
MAX_CACHE_PART_COUNT	Maximum number of partition caches

M$SYS_INDEXES

Displays the index information generated by the user.

Column Name	Description
NAME	Index name
TYPE	Index type
DATABASE_ID	Database identifier
ID	Index identifier
TABLE_ID	Table of index identifier

Column Name	Description
COLCOUNT	Number of columns of created index
PART_VALUE_COUNT	Number of data per partition of index table
BLOOM_FILTER	Availability of Bloom Filter
KEY_COMPRESS	Compression status of key values
MAX_LEVEL	Maximum level of index (LSM only)
PAGE_SIZE	Page size
MAX_KEYWORD_SIZE	Maximum keyword length (keyword only)
BITMAP_ENCODE	Bitmap encoding type (RANGE / EQUAL)

M$SYS_INDEX_COLUMNS

Displays the column information of the user index shown in M$SYS_INDEXES.

Column Name	Description
INDEX_ID	Index identifier
INDEX_TYPE	Index type
NAME	Column name
COL_ID	Column identifier
DATABASE_ID	Database identifier
TABLE_ID	Table identifier
TYPE	Data type of column

M$SYS_TABLESPACES

Displays the table space information created by the user.

Column Name	Description
NAME	Tablespace name
ID	Tablespace identifier
DISK_COUNT	Number of disks in tablespace

M$SYS_TABLESPACE_DISKS

Maintains the disk information used by the tablespace.

Column Name	Description
NAME	Disk name
ID	Disk identifier
TABLESPACE_ID	Disk tablespace identifier
PATH	Disk path
IO_THREAD_COUNT	Number of IO threads allocated to this disk
VIRTUAL_DISK_COUNT	Number of Virtual Disk units assigned to this disk

M$SYS_USERS

Maintain user information registered in Machbase.

Column Name	Description
USER_ID	User identifier
NAME	User name

Collectors

M$SYS_COLLECTORS

Displays the collector information registered in the Machbase server.

Column Name	Description
COLLECTOR_ID	Collector identifier
COLLECTOR_NAME	Collector name
TABLE_NAME	Name of table where Collector will enter data
TEMPLATE_NAME	Template file name
COLLECTOR_TYPE	Collector type
COLLECTOR_SOURCE	Input log file location
COLLECTOR_LIB	Input library name
COL_COUNT	Number of columns
PREPROCESS_PATH	Preprocessor file path
REGEX_PATH	Regular Expression file path
REGEX	Regular Expression
END_REGEX	Signifies end of regular expression
LANGUAGE	Log file language setting (UTF-8)
SLEEP_TIME	Collector input cycle
DB_ADDR	Machbase server IP address
DB_PORT	Machbase server port
DB_USER	Database user name
DB_PASS	Database user password
PROCESS_BYTE	Data size input at once (reset upon re-input)
PROCESS_RECORD	Number of data records input at once (reset upon re-input)
TOTAL_PROCESS_BYTE	Total size of input data after startup
TOTAL_PROCESS_RECORD	Total number of data records input after startup
LAST_PROCESS_TIME	Last input time
RUN_FLAG	Collector run status (0:STOP, 1:START)

M$SYS_COLLECTOR_COLUMNS

Displays the column information of the table entered by the collector.

Column Name	Description
COLLECTOR_ID	Collector identifier
COL_ID	Column identifier
NAME	Column name
TYPE_NAME	Column data type
DATE_FORMAT	Converted string format when datatype is Datetime
TYPE_CODE	Column data type code
SIZE	Column length
USE_INDEX	Index in use status
REGEX_NO	Number of registered regular expressions

M$SYS_COLLECTOR_OFFSETS

Displays the last offset information of the table entered by the collector and the source checksum at that time.

> Available from version 5.5.
>
> In the previous version, the offset information stored in the table was directly inquired. However, since 5.5, the information is continually managed in the memory.

Column Name	Description
USER_NAME	User name
TABLE_NAME	Table name
ADDRESS	Collector location
CHECKSUM1	Collector Source Checksum 1
CHECKSUM2	Collector Source Checksum 2
OFFSET	Collector Source Offset

M$SYS_COLLECTORMANAGERS

Displays the Collector manager information managing the Collector.

Column Name	Description
MANAGER_ID	Collector manager identifier
MANAGER_NAME	Collector manager name
MANAGER_HOST	Collector manager host
MANAGER_PORT	Collector manager port number
MANAGER_LAST_PROCESS_TIME	Last process time of collector manager

M$SYS_COLLECTOR_SOURCES

Displays the format of the source that the Collector collects. This source is managed by the Collector manager.

Column Name	Description
MANAGER_ID	Collector manager identifier
MANAGER_NAME	Collector manager name
SOURCE_TYPE	Log file type
SOURCE_PATH	Log file location path
CONTEXT	

Others

M$TABLES

Display all meta tables beginning with M$.

Column Name	Description
NAME	Meta table name
TYPE	Table type
DATABASE_ID	Database identifier
ID	Meta table identifier
USER ID	Table user (in this case, SYS)
COLCOUNT	Number of columns

M$COLUMNS

Displays the column information of the meta table displayed in M$TABLES.

Column Name	Description
NAME	Column name
TYPE	Column type
DATABASE_ID	Database identifier
ID	Column identifier
LENGTH	Column length
TABLE_ID	Column table identifier
FLAG	(Information for internal use of the server)
PART_PAGE_COUNT	Pages per partition
PAGE_VALUE_COUNT	Number of data per page
MINMAX_CACHE_SIZE	Size of MIN-MAX cache
MAX_CACHE_PART_COUNT	Maximum number of partition caches

Virtual Table

The Virtual Tables are virtual tables that represent various operational information of the Machbase server in the form of a table. The names of these tables begin with "V$".

This data is used to know what state the Machbase server is operating in.
In addition, various information can be obtained through JOIN operation with other tables in this virtual table.

Virtual Tables are read-only and can not be added / deleted / updated by the user.

Session/System

V$PROPERTY

Displays the property information set in the server.

Column Name	Description
NAME	Property name
VALUE	Property value
TYPE	Data type
DEFLT	Default value
MIN	Minimum set value
MAX	Maximum set value

V$SESSION

Displays session information connected to the Machbase server.

Column Name	Description
ID	Session identifier
CLOSED	Whether connection is closed
USER_ID	User identifier
LOGIN_TIME	Connection time
SQL_LOGGING	Leave message in session trace log status 1: Leaves errors occurring in the Parsing, Validation, Optimization steps 2: Leaves result performance of DDL 3: (Leaves both cases above)
SHOW_HIDDEN_COLS	Whether hidden columns are shown upon SELECT
FEEDBACK_APPEND_ERROR	Whether there is feedback to client on APPEND error
DEFAULT_DATE_FORMAT	Default input format upon Datetime input
HASH_BUCKET_SIZE	Number of Buckets in Temp Hashtable created when performing query
MAX_QPX_MEM	Maximum memory size available when performing query
RS_CACHE_ENABLE	Whether Result Cache in in use
RS_CACHE_TIME_BOUND_MSEC	Maximum elapsed time to store results when using Result Cache
RS_CACHE_MAX_MEMORY_PER_QUERY	Maximum size of memory used per query when using Result Cache
RS_CACHE_MAX_RECORD_PER_QUERY	Maximum number of results used per query when using Result Cache

Index

Column Name	Description
RS_CACHE_APPROXIMATE_RESULT_ENABLE	Whether to cache approximate query results when using Result Cache

- V$TABLES
- V$COLUMNS

V$SESMEM

Displays session memory information.

Column Name	Description
SID	Session identifier
ID	Memory manager identifier
USAGE	Usage size

V$SESSTAT

Displays statistical information of the session.

Column Name	Description
SID	Session identifier
ID	Statistical information identifier
VALUE	Statistical information value

V$SESTIME

Displays the time information of the session.

Column Name	Description
SID	Session identifier
ID	Performance unit identifier
ACCUM_TICK	Cumulative time
MAX_TICK	Maximum time (per each performance unit)

V$SYSMEM

Displays memory information of the system.

Column Name	Description
ID	Memory manager identifier
NAME	Memory manager name
USAGE	Current usage
MAX_USAGE	(Recorded) Maximum usage

V$SYSSTAT

Displays statistical information of the system.

Column Name	Description
ID	Statistical information identifier
NAME	Statistical information name
VALUE	Statistical information value

V$SYSTIME

Displays the time information of the system.

Column Name	Description
ID	Performance unit identifier
NAME	Performance unit name
ACCUM_TICK	Cumulative time
AVG_TICK	Average time (per each performance unit)
MIN_TICK	Minimum Time (per each performance unit)
MAX_TICK	Maximum Time (per each performance unit)
COUNT	Performance frequency

V$STMT

Displays information about the query statement that the user is currently executing.

Column Name	Description
ID	Query identifier
SESS_ID	Performed query session identifier
STATE	Query status
RECORD_SIZE	Resulting record size of select statements
QUERY	Query statement

V$VERSION

Displays information about Machbase version.

Column Name	Description
BINARY_DB_MAJOR_VERSION	Database major version
BINARY_DB_MINOR_VERSION	Database minor version
BINARY_META_MAJOR_VERSION	META major version
BINARY_META_MINOR_VERSION	META minor version
BINARY_CM_MAJOR_VERSION	Client (Communication Level) major version
BINARY_CM_MINOR_VERSION	Client (Communication Level) minor version
BINARY_SIGNATURE	Version name of DB data files.
FILE_DB_MAJOR_VERSION	File DB major version
FILE_DB_MINOR_VERSION	File DB minor version
FILE_META_MAJOR_VERSION	File META major version
FILE_META_MINOR_VERSION	File META minor version
FILE_CM_MAJOR_VERSION	File Client (Communication Level) major version
FILE_CM_MINOR_VERSION	File Client (Communication Level) minor version
FILE_CREATE_TIME	File creation time
EDITION	Machbase type

Result Cache

V$RS_CACHE_LIST

Display the result cache list.

Column Name	Description
TOUCH_TIME	Time cache was used or created
USER_ID	Cache user identifier
QUERY	Cache query statement
TIME_SPENT	Time spent producing result
TABLE_COUNT	Number of tables associated with query statement
RECORD_COUNT	Number of result records
REFERENCE_COUNT	Number of sessions currently being referenced
HIT_COUNT	Cache hit count
AGGR_TOUCH_TIME	Time the cache was used or created for aggregate results
AGGR_HIT_COUNT	Cache hit count for aggregate results

V$RS_CACHE_STAT

Display statistical information of result cache in one session.

Column Name	Description
CACHE_COUNT	Number of result caches
CACHE_HIT	Total cache hit count
AGGR_HIT	Total cache hit count for aggregate results
CACHE_REPLACED	Cache replacement count
CACHE_MEMORY_USAGE	Size of cache memory used

Storage

V$STORAGE

Displays internal information of the storage system.

Column Name	Description
DC_TABLE_FILE_SIZE	Total capacity of disk column data
DC_INDEX_FILE_SIZE	Total capacity of index file data
DC_TABLESPACE_DWFILE_SIZE	Total capacity of DWFILE for all column data
DC_KV_TABLE_FILE_SIZE	Total number of data files of TAGDATA table partition tables

V$STORAGE_MOUNT_DATABASES

Displays the information of the mounted backup database using the mount function.

Column Name	Description
NAME	Mounted database name
PATH	Backup file location
BACKUP_TBSID	Backup database tablespace identifier
BACKUP_SCN	Backup database identifier
MOUNTDB	Backup time
DB_BEGIN_TIME	Backup database first entry time
DB_END_TIME	Backup database last entry time

Column Name	Description
BACKUP_BEGIN_TIME	Backup begin time
BACKUP_END_TIME	Backup end time
FLAG	Property flag

V$CACHE

Displays the comprehensive information on the cache objects containing the results read from the storage system.

Column Name	Description
OBJ_COUNT	Current number of result set cache objects

V$CACHE_OBJECTS

Displays information about each cache object that contains the results read from the storage system.

Column Name	Description
OID	Object identifier
REF_COUNT	Reference count
FLAG	(Internal server use flag)

V$STORAGE_DC_TABLESPACES

Displays the table space information of the storage system.

Column Name	Description
NAME	Tablespace name
ID	Tablespace identifier
FLAG	Flag indicating tablespace property
REF_COUNT	Tablespace reference count
DISK_COUNT	Tablespace disk count

V$STORAGE_DC_TABLESPACE_DISKS

Displays the table space information of the storage system.

Column Name	Description
NAME	Disk name
ID	Disk identifier
TABLESPACE_ID	Disk tablespace identifier
PATH	Disk path
IO_THREAD_COUNT	I/O Thread count
IO_JOB_COUNT	I/O Job count
VIRTUAL_DISK_COUNT	Virtual disk count

V$STORAGE_DC_DWFILES

Displays the information of the double-write file (DW File) operated by the storage system.

Column Name	Description
TBS_ID	Tablespace identifier
DISK_ID	Disk identifier

Column Name	Description
FILE	File path
TABLE_ID	Table identifier
COLUMN_ID	Column identifier
PARTITION_ID	Partition identifier
PAGE_ID	Page identifier
DISK_OFFSET	Disk offset
DISK_IMAGE_SIZE	Disk image size
HEAD_CRC32CODE_IMAGE	Head CRC32 Code Image
TAIL_CRC32CODE_IMAGE	Tail CRC32 Code Image
CRC32CODE_PAGE	CRC32 Code Page
HEAD_TIMESTAMP_PAGE	Head Timestamp Page
TAIL_TIMESTAMP_PAGE	Tail Timestamp Page

V$STORAGE_DC_PAGECACHE

Displays information about the Page Cache operating on the storage system

Column Name	Description
MAX_MEM_SIZE	Maximum memory size of Page Cache
CUR_MEM_SIZE	Current memory size of Page Cache
PAGE_CNT	Number of cached pages
CHECK_TIME	Check time

V$STORAGE_DC_PAGECACHE_LRU_LST

Displays information about the LRU List of Page Cache operated by the storage system.

Column Name	Description
OBJECT_ID	Object identifier
LEVEL	Partition level
PARTITION_ID	Partition identifier
OFFSET	Page Cache Offset
SIZE	Page size
REF_CNT	Reference count

Log Table

V$STORAGE_DC_TABLES

Displays internal information about Log Table.

Column Name	Description
ID	Table identifier
DATABASE_ID	Database identifier
CREATE_SCN	System Change Number at time of creation
UPDATE_SCN	System Change Number at time of most recent update
DDL_REF_COUNT	Number of sessions referencing table in DDL syntax execution

Column Name	Description
BEGIN_RID	Minimum table RID
END_RID	Last row ID of table + 1
BEGIN_META_RID	ID at start of recording meta information
END_META_RID	ID at end of recording meta information
END_SYNC_RID	Last row ID recorded on disk + 1
FLAG	Flag indicating table property
COLUMN_COUNT	Table column count
INDEX_COUNT	Table index count
INDEX_MIN_END_RID	Last RID recorded in index + 1
LAST_ARRIVAL_TIME	Last recorded _arrival_time value
LAST_CHECKPOINT_TIME	Last checkpoint time
TYPE	Table type
REMAINING_ROW_COUNT	Number of records not deleted when using auto delete
KEPT_DURATION	How long to keep data when using auto delete function

V$STORAGE_DC_TABLES_STAT

Displays internal information about Log Table.

Column Name	Description
TABLESPACE_ID	Tablespace identifier
TABLE_ID	Table identifier
COLUMN_ID	Column identifier
COUNT	Record count

V$STORAGE_DC_TABLE_COLUMNS

Displays information about the columns in the Log Table.

Column Name	Description
TABLE_ID	Table identifier
DATABASE_ID	Database identifier
ID	Column identifier
FLAG	Property flag
SIZE	Column data size
PARTITION_VALUE_COUNT	Maximum number of data stored in partition
PAGE_VALUE_COUNT	Maximum number of data stored in page
CACHE_VALUE_COUNT	Maximum number of cache values
MINMAX_CACHE_SIZE	Maximum size of MIN / MAX cache for column partitions
CUR_APPEND_PARTITION_ID	Current partition in progress of input identifier
CUR_CACHE_PARTITION_COUNT	Number of partitions that have read data in current cache
CUR_MINMAX_CACHE_SIZE	Current Min / MAX cache size
END_RID_FOR_DEFAULT_VALUE	Location value of end rid maintaining default value
DISK_FILE_SIZE	Total size of column partition data file for that column
MEMORY_TOTAL_SIZE	Memory size used by table
MEMORY_ALLOC_SIZE	Memory size allocated by table

V$STORAGE_DC_TABLE_COLUMN_PARTS

Displays column partition information of log table.

Column Name	Description
TABLE_ID	Table identifier
DATABASE_ID	Database identifier
COLUMN_ID	Column identifier
ID	Partition identifier
FLAG	Flag indicating column property
BEGIN_RID	First RID stored in partition
END_RID	Last RID stored in partition
END_SYNC_RID	Last RID SYNC ended. Data with a RID greater than the starting RID and less than the last SYNC RID is recorded in the partition file.
MIN_TIME	First time data was entered into column partition
MAX_TIME	Last time data was entered into column partition
MAX_VALUE_COUNT_PER_PARTITION	Maximum partition data count
MAX_VALUE_COUNT_PER_PAGE	Maximum page data count
MAX_PAGE_COUNT	Maximum partition page count
PAGE_SIZE	Page size stored in column partition
PAGE_COUNT	Page count created in current column partition
COMPRESS_RATIO	Column partition compression ratio. If it is 0, data compression has not been performed yet.
DISK_FILENAME	Partition file name
EXTERNAL_PART_SIZE	A large amount of data is written to the external partition file, indicating the size of the file
MIN_VALUE	Minimum column partition value
MAX_VALUE	Maximum column partition value

V$STORAGE_DC_TABLE_INDEXES

Displays index information generated in Log Table.

Column Name	Description
TABLE_ID	Table identifier
DATABASE_ID	Database identifier
ID	Index identifier
FLAG	Flag indicating index property
TABLE_BEGIN_RID	First RID entered into table
TABLE_END_RID	Last table RID
BEGIN_RID	First index RID
END_RID	Last index RID
END_SYNC_RID	Last recorded RID in file + 1
COLUMN_COUNT	Index column count
BEGIN_PART_ID	Index first partition identifier
END_PART_ID	Index last partition identifier
FLUSH_REQUEST_COUNT	Number of index partitions requested to reflect on disk
MAX_KEY_SIZE	Maximum key size
INDEX_TYPE	Index type
DISK_FILE_SIZE	Total size of index partition file for that index

Column Name	Description
LAST_CHECKPOINT_TIME	Last checkpoint time

LSM(Log Structured Merge) Index

V$STORAGE_DC_LSMINDEX_LEVEL_PARTS

Displays information about LSM Index partitions.

Column Name	Description
TABLE ID	Index table identifier
TABLESPACE_ID	Tablespace identifier
INDEX_ID	Index identifier
LEVEL	Index partition LSM level
PARTITION_ID	Partition identifier
BEGIN_RID	First RID entered into partition
END_RID	Last RID entered into partition + 1
KEY_VALUE_COUNT	Key value count entered into partition
KEY_VALUE_TABLE_SIZE	Size of page storing key value
KEY_VALUE_TABLE_PAGE_COUNT	Number of pages storing key value
MIN_KEY_VALUE	Minimum key value
MAX_KEY_VALUE	Maximum key value
BITMAP_TABLE_SIZE	Total size of page storing bitmap value
BITMAP_TABLE_PAGE_COUNT	Number of pages storing bitmap value
META_SIZE	Total size of page storing meta information
META_PAGE_COUNT	Number of pages storing meta information
TOTAL_BUILD_MSEC	Total time to complete partition
KEYVAL_BUILD_MSEC	Total time to complete partition for KeyValue Mode
BITMAP_BUILD_MSEC	Total time to complete partition for Bitmap Mode

V$STORAGE_DC_LSMINDEX_LEVEL_PARTS_CACHE

Displays information about the LSM Index partition cache.

Column Name	Description
TABLESPACE_ID	Tablespace identifier
TABLE_ID	Index Table identifier
INDEX_ID	Index identifier
LEVEL	Index partition LSM level
PARTITION_ID	Partition identifier
BEGIN_RID	First RID entered into partition
END_RID	Last RID entered into partition + 1
KEY_VALUE_COUNT	Number of key values entered into partition
KEY_VALUE_TABLE_SIZE	Size of page storing key value
KEY_VALUE_TABLE_PAGE_COUNT	Number of pages storing key value
BITMAP_TABLE_SIZE	Total size of page storing bitmap value
BITMAP_TABLE_PAGE_COUNT	Number of pages storing bitmap value

Column Name	Description
META_SIZE	Total size of page storing meta information
META_PAGE_COUNT	Number of pages storing meta information
MEMORY_SIZE	Memory usage
MEMORY_SIZE_RBTREE	Redblack Tree memory usage

V$STORAGE_DC_LSMINDEX_LEVELS

Displays information about the level of the LSM index.

Column Name	Description
TABLE ID	Table identifier
DATABASE_ID	Database identifier
INDEX_ID	Index identifier
LEVEL	Level
BEGIN_RID	First partition RID
END_RID	Last partition RID + 1
META_BEGIN_RID	RID at start time of recording meta information
META_END_RID	RID at end time of recording meta information
DELETE_END_RID	Maximum deleted RID + 1

V$STORAGE_DC_LSMINDEX_FILES

Displays information about the files that make up the LSM Index.

Column Name	Description
TABLE_ID	Table identifier
DATABASE_ID	Database identifier
INDEX_ID	Index identifier
LEVEL	Index partition LSM level
PARTITION_ID	Partition identifier
BEGIN_RID	Partition first RID
END_RID	Partition last RID + 1
PATH	Index file location

V$STORAGE_DC_LSMINDEX_AGER_JOBS

Displays working status of Ager responsible for LSM Index deletion.

Column Name	Description
TABLE_ID	Table identifier
INDEX_ID	Index identifier
LEVEL	Index partition LSM level
BEGIN_RID	First partition RID
END_RID	Last partition RID + 1
STATE	Index Ager working status

Volatile Table

V$STORAGE_DC_VOLATILE_TABLE

Displays information about Volatile Table.

Column Name	Description
MAX_MEM_SIZE	Maximum Volatile Tablespace size
CUR_MEM_SIZE	Current Volatile Tablespace size

Tag Table

V$STORAGE_TAG_TABLES

Displays information about the partition table in the Tagdata Table.

Column Name	Description
ID	Table identifier
TABLE_BEGIN_RID	Table start RID
TABLE_END_RID	Table end RID
WRITE_END_RID	Last RID which is written to data file.
DISK_INDEX_END_RID	Index end RID stored in storage
MEMORY_INDEX_END_RID	Table end RID in memory index
INDEX_STATE	Current Index Build Stats • IDLE: Build Complete, waiting • PROGRESS: Build in progress • IOWAIT: Waiting for I/O operation in storage • PENDING: Waiting for table read lock • SHUTDOWN: Stopped. DELETE operation or DROP operation in progress. • ABNORMAL: Abnormal end
DELETE_STATE	Current DELETE operation state. There is no IDLE because it is performed only when a DELETE command is entered. • PROGRESS: Deletion in progress • IOWAIT: Waiting for I/O operation in storage • PENDING: Waiting for table read/write lock • SHUTDOWN: Stopped. DELETE operation or DROP operation in progress. • ABNORMAL: Abnormal end

V$STORAGE_TAG_CACHE

Displays the cache information used in the partition table of the Tagdata Table.

Column Name	Description
CATEGORY	Type of object in cache
USED_MEMORY	Size of memory in use
TAGMAP_CNT	Index (cache) count
TAGMAP_HIT	Index cache hit count
TAGMAP_MISS	Index cache miss count
TAGMAP_FLUSHOUT	Number of page flushouts due to index cache crash
TAGMAP_COLDREAD	Number of index pages read directly from storage
TAGMAP_MEMORY_WAIT	Number of times index memory waited for cache crash
TAGMAP_IO_WAIT	Index read operation wait count
BLOCK_COUNT	Data cache count
CACHE_HIT	Data cache hit count
CACHE_MISS	Data cache miss count

Column Name	Description
FLUSHOUT	Number of page flushouts due to data cache crash
COLDREAD	Number of data pages read directly from storage
MEMORY_WAIT	Number of times data memory waited for cache crash
IO_WAIT	Data read operation wait count

V$STORAGE_TAG_TABLE_FILES

Displays the file information of the partition table of the Tag Table.

Column Name	Description
TABLE_ID	Table identifier
FILE_ID	File identifier
STATE	Index status • COMPLETE: Data stored, index build complete • INDEXING: Index build in progress • FILLED: Data is full, waiting for Index build • PARTIAL: Data not yet full, waiting for Index build
MIN_DATE	Minimum datatime value of this data file.=
MAX_DATE	Maximum datatime value of this data file.=

Tag Rollup

V$ROLLUP

Displays the Rollup information that stores statistical information of the Tagdata table.

Column Name	Description
ID	Rollup job ID
ROLLUP_TABLE_NAME	Table name to store Rollup information
SOURCE_TABLE_NAME	Name of tag table that Rollup will query
END_RID	Source Table end RID
ENABLED	Indicates Rollup progress status
LAST_INSERTED_ROW_COUNT	Number of recently entered records

V$ROLLUP_STATUS

Displays the rollup status of the Tagdata table.

Column Name	Description
SRC_TABLE	Name of table that Rollup will query
ROLLUP_TABLE	Name of table that Rollup will store
ENABLED	Indicates Rollup progress status
ROLLUP_END_RID	End RID of Rollup save table
LAST_ELAPSED_MSEC	Elapsed recent Rollup time

Stream

V$STREAMS

Column Name	Description
NAME	The name of stream query.
LAST_EX_TIME	Last execution time of this query.
TABLE_NAME	The name of table which searched from the query
END_RID	The last RID read by stream query
STATE	Current state of stream query
QUERY_TXT	query text
ERROR_MSG	Error message of the last stream execution
FREQUENCY	Minimum wait time for query execution. If it is 0, it is executed every record. If it is not 0, it is executed each time. The unit is nanoseconds.

License

V$LICENSE_INFO

Displays license information.

Column Name	Description
INSTALL_DATE	Installation date
TYPE	License type
POLICY	License policy type
CUSTOMER	Customer name
ISSUE_DATE	Issue date
ID	Host ID
EXPIRY_DATE	Expiration date
SIZE_LIMIT	Work input limit
ADDENDUM	Additional data rate
VIOLATION_ACTION	Indicates license violation
VIOLATION_LIMIT	Number of violations to suspend service (monthly update)
STOP_ACTION	Indicates database is terminated in the event of license violation
RESET_FLAG	(Internal server use)

V$LICENSE_STATUS

Displays the license status.

Column Name	Description
USER_DATA_PER_DAY	Amount of data that can be entered per day
PREVIOUS_CHECK_DATE	Previous license check date
VIOLATION_COUNT	License violation count

Cluster

V$NODE_STATUS

Displays the Node status for each Cluster. Only one is displayed.

Column Name	Description
NODETYPE	Node type. There are two types that can be viewed by queries. • Broker • Warehouse
STATE	Node status

V$DDL_INFO

Displays DDL information performed by Cluster.

Column Name	Description
SEQUENCENUMBER	DDL sequence number
TIME	DDL execution time
VALUE	DDL query result value (Internal server use)
CLIENT	Client name
BROKER	Lead Broker Node name
USER	User name
SQL	DDL query value

V$REPLICATION

Displays information about the replication operation.

Column Name	Description
HOSTNAME	Replication Node Hostname
MODE	(Internal server use)
STATE	Node status
ADDR	Replication Manager address
PORT_NO	Replication Manager port number
MAX_SENDER_COUNT	Maximum number of Senders that can be created
RUN_SENDER_COUNT	Maximum number of active Senders

V$REPL_SENDER

Displays Sender replication when running Replication.

Column Name	Description
HOSTNAME	Replication Node Hostname
ID	Sender identifier
STATUS	Sender operational status
PAYLOAD_RECV_COUNT	Number of payloads received from sender
PAYLOAD_RECV_BYTES	Total payload size received from Sender
QUEUE_REMAIN_COUNT	Number of buffers remaining in the Receive Queue
NET_SEND_COUNT	Net send count
NET_SEND_SIZE	Net send size
NET_RECV_COUNT	Net receive count
NET_RECV_SIZE	Net receive size

V$REPL_SENDER_META

Displays Sender metadata when running Replication.

Column Name	Description
HOSTNAME	Replication Node Hostname
SENDER_ID	Sender identifier
TABLE_ID	Target table identifier
TABLE_TYPE	Target table type
BEGIN_RID	Target record start RID
END_RID	Target record end RID

V$REPL_RECEIVER

Displays Receiver information when running Replication.

Column Name	Description
HOSTNAME	Replication Node Hostname
STATUS	Receiver operational status
PAYLOAD_RECV_COUNT	Number of payloads received from sender
PAYLOAD_RECV_BYTES	Total payload size received from Sender
QUEUE_REMAIN_COUNT	Number of buffers remaining in the Receive Queue
NET_SEND_COUNT	Net send count
NET_SEND_SIZE	Net send size
NET_RECV_COUNT	Net receive count
NET_RECV_SIZE	Net receive size

V$REPL_RECEIVER_META

Displays Receiver metadata when running Replication.

Column Name	Description
HOSTNAME	Replication Node Hostname
TABLE_ID	Target table identifier
TABLE_TYPE	Target table type
BEGIN_RID	Target record start RID
END_RID	Target record end RID

V$REPL_READER

Displays Reader information when running Replication.

Column Name	Description
HOSTNAME	Replication Node Hostname
SENDER_ID	Sender identifier
ID	Reader identifier
STATUS	Reader operation status
FETCH_COUNT	FETCH count

V$REPL_READER_META

Displays Reader metadata when running Replication.

Column Name	Description
HOSTNAME	Replication Node Hostname

Column Name	Description
SENDER_ID	Sender identifier
ID	Reader identifier
TABLE_ID	Target table identifier
TABLE_TYPE	Target table type
BEGIN_RID	Target record start RID
END_RID	Target record end RID

V$REPL_WRITER

Displays Writer information when running Replication.

Column Name	Description
HOSTNAME	Replication Node Hostname
ID	Writer identifier
STATUS	Writer operational status
APPEND_COUNT	APPEND count

V$REPL_WRITER_META

Displays Writer metadata when running Replication.

Column Name	Description
HOSTNAME	Replication Node Hostname
ID	Writer identifier
TABLE_ID	Target table identifier
TABLE_TYPE	Target table type
BEGIN_RID	Target record start RID
END_RID	Target record end RID

Others

V$TABLES

Displays all Virtual Tables that start with "V$".

Column Name	Description
NAME	Table name
TYPE	Table type
DATABASE_ID	Database identifier
ID	Table identifier
USER ID	User who created table
COLCOUNT	Column count

V$COLUMNS

Displays column information of Virtual Tables.

Column Name	Description
NAME	Column name

Column Name	Description
TYPE	Column data type
DATABASE_ID	Database identifier
ID	Column identifier
LENGTH	Column size
TABLE_ID	Table identifier
FLAG	Private data

Property

The properties are the settings used by the Machbase server and stored as key-value pairs in the $MACHBASE_HOME/conf/machbase.conf file.
These values are set when the Machbase server starts and are used continuously during runtime. To change this value for performance tuning, you must understand the meaning of these values and set them carefully.

CPU_AFFINITY_BEGIN_ID

This is the start number of the CPU used by the Machbase server. It is used to control the CPU usage of the Machbase server.

	Value
Minimum	0
Maximum	2 ^ 32 - 1
Default	0

CPU_AFFINITY_COUNT

This is the number of CPUs that the Machbase server will use. If set to 0, the Machbase server uses all CPUs.

	Value
Minimum	0
Maximum	2 ^ 32 - 1
Default	0

CPU_COUNT

Specifies the number of CPUs set in the system. Based on this value, the Machbase Thread determines the number. If set to 0, all CPUs in the system are used.

	Value
Minimum	0 (auto detect the physically installed count of CPU on the system)
Maximum	2^32 -1
Default	0

CPU_PARALLEL

Specifies the number of threads to spawn per CPU. If this value is 2 and the number of CPUs is 2, then two parallel threads are created per CPU, so the number of parallel processing threads is four. If this value is too large, memory can be consumed quickly.

	Value
Minimum	1
Maximum	2^32 -1
Default	1

DBS_PATH

Specifies the path where the basic data of the Machbase server will be stored. The default is "? Dbs", which means `$MACHBASE_HOME/dbs`.

Index

	Value
Default	?/dbs

DEFAULT_LSM_MAX_LEVEL

Sets the base level of the LSM index. If you do not enter a MAX_LEVEL value when creating an index, this value applies.

	Value
Minimum	0
Maximum	3
Default	2

DISK_BUFFER_COUNT

Specifies the number of buffers for disk I/O.

	Value
Minimum	1
Maximum	4G (4 * 1024 * 1024 * 1024)
Default	16

DISK_COLUMNAR_INDEX_CHECKPOINT_INTERVAL_SEC

Sets the checkpoint interval for the index. If set too long, errors may occur during index creation.

	Value
Minimum	1 (sec)
Maximum	2^32 -1 (sec)
Default	120 (sec)

DISK_COLUMNAR_INDEX_FDCACHE_COUNT

Specifies the number of opened index partition file descriptors.

	Value
Minimum	0
Maximum	2^32 -1
Default	0

DISK_COLUMNAR_INDEX_SHUTDOWN_BUILD_FINISH

Sets whether or not to reflect index information on the disk when the Machbase server is shutdown. If this value is set to '1', all index information is reflected on the disk and ends, so waiting times may be long.

	Value
Minimum	0 (False)
Maximum	1 (True)
Default	0 (False)

DISK_COLUMNAR_PAGE_CACHE_MAX_SIZE

Sets the maximum size of the page cache.

	Value
Minimum	0
Maximum	2^64 - 1
Default	2 * 1024 * 1024 * 1024

DISK_COLUMNAR_TABLE_CHECKPOINT_INTERVAL_SEC

Sets checkpoint period of table data. If this value is too large, the recovery time will be longer at restart. If this value is too small, I/O will frequently occur and the overall performance may be degraded.

	Value
Minimum	1 (sec)
Maximum	2^32 -1 (sec)
Default	120 (sec)

DISK_COLUMNAR_TABLE_COLUMN_FDCACHE_COUNT

Specifies the maximum number of open file descriptors for column data in the table.

	Value
Minimum	0
Maximum	2^32 - 1
Default	0

DISK_COLUMNAR_TABLE_COLUMN_MINMAX_CACHE_SIZE

Sets the size of the default MINMAX cache set in the _ARRIVAL_TIME column.

	Value
Minimum	0
Maximum	2^64 - 1
Default	100 *1024 * 1024

DISK_COLUMNAR_TABLE_COLUMN_PART_FLUSH_MODE

Sets the automatic flush interval for column partition files.

	Value
Minimum	0 (sec)
Maximum	2^32-1 (sec)
Default	60 (sec)

DISK_COLUMNAR_TABLE_COLUMN_PART_IO_INTERVAL_MIN_SEC

Sets the frequency with which the partition file is reflected on the disk. When more data is input than the number of partitions set, it is reflected on the disk regardless of this period.

	Value
Minimum	0 (sec)
Maximum	2^32-1 (sec)
Default	3 (sec)

DISK_COLUMNAR_TABLE_COLUMN_PARTITION_PRECREATE_COUNT

Defines the number of pre-generated column partition objects to be used for the table.

	Value
Minimum	1
Maximum	2^32-1
Default	3

DISK_COLUMNAR_TABLE_TIME_INVERSION_MOD

If set to 1, the input is allowed even if the value of the _ARRIVAL_TIME column is reduced. If it is 0, a value smaller than the Maximum of the _ARRIVAL_TIME column value is entered as an error.

	Value
Minimum	0 (False)
Maximum	1 (True)
Default	1 (True)

DISK_COLUMNAR_TABLESPACE_DWFILE_EXT_SIZE

Specifies the size at which the double write file used for recovery at startup increases at one time.

	Value
Minimum	1024 * 1024
Maximum	2^32 - 1
Default	1024 * 1024

DISK_COLUMNAR_TABLESPACE_DWFILE_INT_SIZE

Specifies the amount of space secured by the double write file when the file is created.

	Value
Minimum	1024 * 1024
Maximum	2^32 - 1
Default	2* 1024 * 1024

DISK_COLUMNAR_TABLESPACE_MEMORY_EXT_SIZE

Specifies the block size of the memory to reserve for the column partition.

	Value
Minimum	1024 * 1024
Maximum	2^64 - 1
Default	2* 1024 * 1024

DISK_COLUMNAR_TABLESPACE_MEMORY_MAX_SIZE

Specifies the maximum amount of memory allocated by the log table. If the server allocates more than this amount of memory, the memory allocation will wait until the memory usage drops below this value. It is recommended to set this value to 50 ~ 80% of physical memory.

	Value
Minimum	256 * 1024 * 1024
Maximum	2^64 - 1
Default	8 * 1024 * 1024 * 1024

DISK_COLUMNAR_TABLESPACE_MEMORY_MIN_SIZE

When the Machbase server starts, it pre-allocates memory by this value to prevent performance degradation due to memory allocation. Since this memory is used only as a data input buffer, it is recommended to use it only when memory is sufficient.

Table 24. Range of values

	Value
Minimum	1024 * 1024
Maximum	2^64 - 1
Default	100 * 1024 * 1024

DISK_COLUMNAR_TABLESPACE_MEMORY_SLOWDOWN_HIGH_LIMIT_PCT

Limits the performance when the memory usage exceeds the set value when data is input to the log table.

```
DISK_COLUMNAR_TABLESPACE_MEMORY_MAX_SIZE * (DISK_COLUMNAR_TABLESPACE_MEMORY_SLOWDOWN_HIGH_LIMIT_PCT / 100)
```

	Value
Minimum	0
Maximum	100
Default	80

DISK_COLUMNAR_TABLESPACE_MEMORY_SLOWDOWN_MSEC

Sets the next wait time for each record entry if the memory usage for the column data file exceeds the criterion.

	Value
Minimum	0 (msec)
Maximum	2^32 - 1 (msec)
Default	1 (msec)

DISK_IO_THREAD_COUNT

Sets the number of I/O threads that write data to disk.

	Value
Minimum	1
Maximum	2^32 -1
Default	3

DISK_TABLESPACE_DIRECT_IO_FSYNC

When running Direct I/O, fsync is unnecessary for data files. Disable fsync when using Direct I/O to improve data I/O performance (Set to 0).
Although fsync is unncessary, fsync must be set to perform in case of failure situations such as a power outage because in a normal situation there is no data loss,

	Value
Minimum	0
Maximum	1
Default	0

DISK_TABLESPACE_DIRECT_IO_READ

Sets whether to use DIRECT I/O for data read operation.

	Value
Minimum	0
Maximum	1
Default	0

DISK_TABLESPACE_DIRECT_IO_WRITE

Sets whether to use DIRECT I/O for data write operation.

	Value
Minimum	0
Maximum	1
Default	1

DUMP_APPEND_ERROR

If this value is set to 1, the `$MACHBASE_HOME/trc/machbase.trc` file will record the error if the Append API fails.
In this situation, the append performance is very low, so it is recommended to use for testing purposes only.

If you want to check for errors in the user application, it is helpful to use the SQLAppendSetErrorCallback API.

	Value
Minimum	0
Maximum	1
Default	0

DUMP_TRACE_INFO

The server periodically records the DBMS system status information in the machbase.trc file at regular intervals, and sets this period.
If it is set to 0, it is not recorded.

	Value
Minimum	0 (sec)
Maximum	2^32 - 1 (sec)
Default	60 (sec)

DURATION_BEGIN

Sets the start time of the duration value that sets the default for the SELECT statements that do not specify the DURATION clause.
If set to 60, data will be retrieved 60 seconds before the current time.

The default is 0 to retrieve all data.

	Value
Minimum	0
Maximum	2^32 - 1
Default	0

DURATION_GAP

Sets the start time of the duration value that sets the default for the SELECT statements that do not specify the DURATION clause.

- If set to 60, data will be retrieved for 60 seconds from the current time.
- If the DURATION_BEGIN value is 60, the data is retrieved from 60 seconds before to 60 seconds from the current time.

The default is 0 to retrieve all data.

	Value
Minimum	0

	Value
Maximum	Non-zero
Default	0

FEEDBACK_APPEND_ERROR

Sets whether to send error data to the client when an Append API error occurs. If 0, no error data is sent to the client. If it is 1, error information is sent to the client.

	Value
Minimum	0
Maximum	1
Default	1

GRANT_REMOTE_ACCESS

Determines whether the database can be accessed remotely. If 0, the remote connection is blocked.

	Value
Minimum	0 (False)
Maximum	1 (True)
Default	1 (True)

INDEX_BUILD_MAX_ROW_COUNT_PER_THREAD

If the number of records not indexed is greater than this value, the index build thread begins to add indexes.

	Value
Minimum	1
Maximum	2^32 - 1
Default	100000

INDEX_BUILD_THREAD_COUNT

Specifies the number of index creation threads. If set to 0, no index is created.

	Value
Minimum	0
Maximum	2 ^ 32 - 1
Default	3

INDEX_FLUSH_MAX_REQUEST_COUNT_PER_INDEX

Specifies the maximum number of flush requests per index.

	Value
Minimum	0
Maximum	2 ^ 32 - 1
Default	3

INDEX_LEVEL_PARTITION_AGER_THREAD_COUNT

Specifies the number of threads to delete index files that are not needed when creating LSM indexes.

	Value
Minimum	0
Maximum	1024
Default	1

INDEX_LEVEL_PARTITION_BUILD_MEMORY_HIGH_LIMIT_PCT

Sets the maximum memory usage for LSM index creation as a percent. This percent is set based on the maximum memory usage used by Machbase. If the memory usage exceeds the limit, the LSM partition merge is stopped.

	Value
Minimum	0
Maximum	100
Default	70

INDEX_LEVEL_PARTITION_BUILD_THREAD_COUNT

Determines the number of threads performing the merge operation for the creation of the LSM index.

	Value
Minimum	1
Maximum	1024
Default	3

MAX_QPX_MEM

Sets the maximum amount of memory used by the query processor to perform the GROUP BY, DISTINCT, and ORDER BY clauses.
If one query uses memory with a larger value, the query is canceled. At this time, an error message is sent to the client, and the relevant content is recorded in the machbase.trc file.

	Value
Minimum	1024 * 1024
Maximum	2^64 - 1
Default	500 * 1024 * 1024

MEMORY_ROW_TEMP_TABLE_PAGESIZE

Sets the page size of the temporary tablespace for volatile tables and lookup tables. Because this page stores volatile tables and lookup table records, it should be larger than the maximum record size for volatile tables.
If you want to enter N records into the page, you should set this value to the maximum record size * N.

	Value
Minimum	8 * 1024
Maximum	2^32 - 1
Default	32 * 1024

PID_PATH

Specifies the location where the PID file of the Machbase server process is to be written. The default is "?/Conf", which means `$MACHBASE_HOME/conf`.

	Value
Default	?/conf

PID_PATH Value	PID File Location Path
Not Specified	$MACHBASE_HOME/conf/machbase.pid

PID_PATH Value	PID File Location Path
?/test	$MACHBASE_HOME/test/machbase.pid
/tmp	/tmp/machbase.pid

PORT_NO

Specifies the TCP/IP port for the Machbase server process to communicate with the client. The Default is 5656.

	Value
Minimum	1024
Maximum	65535
Default	5656

PROCESS_MAX_SIZE

Specifies the maximum memory size used by machbased programs that are Machbase server processes. If you try to use more memory than the set limit, the server operates as follows to reduce the memory usage.

- Stops data insert or treats it as an error
- Decreased index creation speed

In this case, the performance is greatly degraded, so the cause of overuse of the memory must be found and solved.

	Value
Minimum	1024 * 1024 * 1024
Maximum	2^64 - 1
Default	8 * 1024 * 1024 * 1024

QUERY_PARALLEL_FACTOR

Specifies the number of execution threads of the parallel query executor.

	Value
Minimum	1
Maximum	100
Default	8

RS_CACHE_APPROXIMATE_RESULT_ENABLE

Determines whether to use the approximate result mode of the result cache. If this value is 1, the speculative value is obtained (very fast but the data may be inaccurate) when using the result cache, and if it is 0, the correct value is obtained.

	Value
Minimum	0 (False)
Maximum	1 (True)
Default	0 (False)

RS_CACHE_ENABLE

Determines whether to use the result cache.

	Value
Minimum	0 (False)
Maximum	1 (True)
Default	1 (True)

RS_CACHE_MAX_MEMORY_PER_QUERY

Sets the amount of memory the result cache will use. If the memory usage of a particular query result exceeds this value, the result of the query is not stored in the result cache.

	Value
Minimum	1024
Maximum	2^64 - 1
Default	16 * 1024 * 1024

RS_CACHE_MAX_MEMORY_SIZE

Specifies the maximum memory usage of the result cache.

	Value
Minimum	32 * 1024
Maximum	2^64 - 1
Default	512 * 1024 * 1024

RS_CACHE_MAX_RECORD_PER_QUERY

The maximum number of records to be stored in the result cache. If the number of records resulting from the query is greater than this value, the query result is not stored in the cache.

	Value
Minimum	1
Maximum	2^64 - 1
Default	10000

RS_CACHE_TIME_BOUND_MSEC

If a particular query is executed very quickly, it is better not to store it in the result cache because it can reduce memory usage.
This value determines how fast the query executed should not be stored in the cache. When set to 0, all query results are stored in the result cache.

	Value
Minimum	0 (msec)
Maximum	2^64 - 1 (msec)
Default	1000 (msec)

SHOW_HIDDEN_COLS

If set to the Default of 0, the _ARRIVAL_TIME column is not displayed by the SELECT * FROM query. If this value is set to 1, the corresponding column is displayed.

	Value
Minimum	0
Maximum	1
Default	0

TAGDATA_AUTO_META_INSERT

> In 5.5 version, this property name was TAGDATA_AUTO_NAME_INSERT and supports only 0 or 1.

When entering data through APPEND / INSERT into the TAGDATA table, specify how to handle it if there is no matching TAG_NAME.

- 0: Input fails.
- 1: Input TAG_NAME value to input. If there are additional metadata columns, the values of all columns are entered as NULL.
- 2: Enter the additional metadata column value along with the TAG_NAME value you want to enter.
    - This setting is valid only in APPEND. INSERT works like 1 because you cannot enter additional metadata column values.

- After this setting, the APPEND parameter must include the metadata column value in APPEND.

	Value
Minimum	0
Maximum	2
Default	1

TRACE_LOGFILE_COUNT

Specifies the maximum number of log trace files generated in TRACE_LOGFILE_PATH. To save disk space, delete the oldest log file if more than the maximum number of log files are created.

	Value
Minimum	1
Maximum	2^32 - 1
Default	5

TRACE_LOGFILE_PATH

Set the path of the log trace files (machbase.trc, machadmin.trc, machcollector.trc, machsql.trc).
These files continuously record internal information at the start, end, and run of Machbase. The default ?/trc means `$MACHBASE_HOME/trc`.

	Value
Default	?/conf

TRACE_LOGFILE_PATH	trc direction location
Not Specified	$MACHBASE_HOME/trc/
?/test	$MACHBASE_HOME/test/
/tmp	/tmp/

TRACE_LOGFILE_SIZE

Sets the maximum size of the log trace file. If it is necessary to record more data than the size, a new log file is created.

	Value
Minimum	10 * 1024 * 1024
Maximum	2^32 - 1
Default	10 * 1024 * 1024

UNIX_PATH

Sets the path to the Unix domain socket file. The Default when not set by user is ?/conf/machbase-unix.

	Value
Default	?/conf/machbase-unix

VOLATILE_TABLESPACE_MEMORY_MAX_SIZE

Sets the total amount of memory usage for all volatile and lookup tables in the system.

	Value
Minimum	0

	Value
Maximum	2^64 - 1
Default	2 * 1024 * 1024 * 1024

DISK_BUFFER_COUNT

Specifies the number of buffers to perform disk I/O.

	Value
Minimum	1
Maximum	2^32 - 1
Default	16

Property (Cluster)

Separate from Property, Property (Cluster) organizes the Property only available in Cluster Edition.

CLUSTER_LINK_ACCEPT_TIMEOUT

Timeout until receiving Handshake message after Accept when connecting to a specific Node
Failure to receive within the timeout will cause the connection to fail.
The default value is 5 seconds.

(usec)	Value
Minimum	0
Maximum	2^64 - 1
Default	5000000

CLUSTER_LINK_CHECK_INTERVAL

Check interval of the Timeout Thread that checks the Sockets connected to a specific Node
There is a Timeout Thread that checks RECEIVE_TIMEOUT and SESSION_TIMEOUT.
The shorter the cycle is, the more frequently it is checked, but the Timeout determination is made according to the following values.
The default value is 1 second.

(usec)	Value
Minimum	0
Maximum	2^64 - 1
Default	1000000

CLUSTER_LINK_CONNECT_RETRY_TIMEOUT

Timeout to repeat reconnection attempt after connection failure with a specific Node
If it is not connected within the timeout, it is determined to be completely disconnected.
The default value is 1 minute.

(usec)	Value
Minimum	0
Maximum	2^64 - 1
Default	60000000

CLUSTER_LINK_CONNECT_TIMEOUT

Time to wait when trying to connect to a specific Node
If it does not connect within the Timeout, it will try to reconnect until CLUSTER_LINK_CONNECT_RETRY_TIMEOUT has passed.
The default value is 5 seconds.

(usec)	Value
Minimum	0
Maximum	2^64 - 1
Default	5000000

CLUSTER_LINK_ERROR_ADD_ORIGIN_HOST

You can choose whether to add an errored host name to error messages that occur during communication between the Cluster.
If you want to display a detailed error message, set the property to 1.
The default value is 0, which means the host name is not displayed.

(boolean)	Value

Minimum	0
Maximum	1
Default	0

CLUSTER_LINK_HANDSHAKE_TIMEOUT

Timeout until receiving a Handshake message while connected to a specific Node and Cluster Socket
Two Nodes that have just finished connecting exchange small size Handshake messages to check the connection status.
The Accept Node sends the Handshake message first, and the time to wait for the response is set here.
The default value is 5 seconds.

(usec)	Value
Minimum	0
Maximum	2^64 - 1
Default	5000000

CLUSTER_LINK_BUFFER_SIZE

Available in 5.7 and after

The size of the send / receive buffer.

If this size is insufficient, the transmission will retry until the buffer is empty.

The default value is 32M.

(byte)	Value
Minimum	1024768
Maximum	2^32 - 1
Default	33554432 (32M)

CLUSTER_LINK_SEND_RETRY_COUNT

Available in 5.7 and after

Number of times to retry sending until the send buffer is empty.

Every retry will take 1ms off. If you retry beyond this number, you will be disconnected.

The default value is 5000 (msec).

(count)	Value
Minimum	0
Maximum	2^32 - 1
Default	5000

CLUSTER_LINK_HOST

Host name of the current Node to connect to a specific Node and Cluster Socket

(string)	Value
Default	localhost

CLUSTER_LINK_LONG_TERM_CALLBACK_INTERVAL

If the execution time of Receive Callback to process a message received on Cluster Socket exceeds the set value, it is recognized as Long-Term Callback.
Since the number of receive Threads is limited, Receive Callback should not process messages for a long time.

If Receive Callback processes the message after this time, it recognizes it as Long-Term Callback and records it in Trace Log.
The default value is 1 second.

(usec)	Value
Minimum	0
Maximum	2^64 - 1
Default	1000000

CLUSTER_LINK_LONG_WAIT_INTERVAL

If the time until the arrival of a message received on Cluster Socket exceeds the set value, it is recognized as Long-Wait Message.
If the time from receiving start to receiving end is long, it can be regarded as a problem of the network environment.
If the received message does not arrive after this time, it is recognized as a Long-Wait Message and recorded in the Trace Log.
The default value is 1 second.

(usec)	Value
Minimum	0
Maximum	2^64 - 1
Default	1000000

CLUSTER_LINK_MAX_LISTEN

The maximum number of Socket's Accept Queue when connecting to a specific Node

(count)	Value
Minimum	1
Maximum	2^32 - 1
Default	512

CLUSTER_LINK_MAX_POLL

The maximum number of Events that can be retrieved at a time by Poll when communicating with a specific node

(count)	Value
Minimum	1
Maximum	2^32 - 1
Default	4096

CLUSTER_LINK_PORT_NO

The port number of the current Node for connecting the specific Node to the Cluster Socket

(port)	Value
Minimum	1024
Maximum	65535
Default	3868

CLUSTER_LINK_RECEIVE_TIMEOUT

Timeout until the Timeout Thread determines that the connection has been disconnected since the last reception
Connections that exist in the 'Linked List' should be continuously receiving because the connection between Cluster Nodes is terminated when the reception is complete.
If the last received time is not updated after the set time has elapsed, the Timeout Thread records its contents in the Trace Log and closes the Socket.

(usec)	Value
Minimum	0

Maximum	2^64 - 1
Default	30000000

CLUSTER_LINK_REQUEST_TIMEOUT

Timeout from when a request message is sent from the Cluster Socket to when a response to the request is received
For specific messages, specify the time to wait for a response after the request.
If the response message does not arrive at this time, write to the Trace Log and close the Socket.
The default value is 60 seconds, and the Timeout is long because it is not known what kind of message and receive processing will happen.

(usec)	Value
Minimum	0
Maximum	2^64 - 1
Default	60000000

CLUSTER_LINK_SEND_TIMEOUT

Timeout to set when sending messages through Cluster Socket
Set the corresponding timeout when transmitting
If transmission is not completed until Timeout, it is recorded in the Trace Log.

(usec)	Value
Minimum	0
Maximum	2^64 - 1
Default	30000000

CLUSTER_LINK_SESSION_TIMEOUT

Timeout until the Timeout thread determines that the connection has been disconnected since the last receive in a specific session.
Cluster connection manages the session of all messages internally, which is a necessary property in case the session can suddenly not be fixed.
If the last receive time for the session is not updated after this time, the Timeout Thread writes to the Trace Log and closes the session.
The default value is 1 hour.

(usec)	Value
Minimum	0
Maximum	2^64 - 1
Default	3600000000

CLUSTER_LINK_THREAD_COUNT

The number of Threads to process the received messages when communicating with a specific Node
If the size of the Cluster grows or the number of operations to be processed increases, the number of Threads that can be received increases.

(count)	Value
Minimum	1
Maximum	4096
Default	8

CLUSTER_QUERY_STAT_LOG_ENABLE

Outputs statistical information about the executed query to the trace log.

(boolean)	Value
Minimum	0
Maximum	1

Default	0

CLUSTER_REPLICATION_BLOCK_SIZE

The size of the data to be sent at once when the Replication for adding Node is performed in the Cluster Edition
The Property must be applied directly to the warehouse (=Transmitting Warehouse) that becomes the Replication Active.
The default value is 640 KB.

(size)	Value
Minimum	64 * 1024
Maximum	100 * 1024 * 1024
Default	640 * 1024 (655360)

CLUSTER_WAREHOUSE_DIRECT_DML_ENABLE

It is made possible to connect directly to the Warehouse to perform DML in Cluster Edition.

- 1: Executable
- 2: Not executable. An error is returned.

When directly performing the DML in Warehouse, there are performance advantages over Brokers but there is an issue where the DML is not propagated to the same Group.
Therefore, it is used only for emergency recovery due to data discrepancies, or if the data discrepancies of the Group can be taken into account.
You must apply Properties directly to the specific Warehouse you want.
The default value is 0.

> ⚠ The Coordinator does not check for data discrepancies, even if there is a data difference between the Warehouses in the Group with the corresponding Property turned on.

(boolean)	Value
Minimum	0
Maximum	1
Default	0

COORDINATOR_DBS_PATH

Specifies the directory where the Coordinator data file will be created
The default value is set to ?/dbs, and ? is replaced with the $ MACHBASE_COORDINATOR_HOME environment variable.
This is an environment variable `$MACHBASE_COORDINATOR_HOME/dbs` directory.
It must be applied to the Coordinator, and it has no effect on other Nodes.

(path)	Value
Default	?/dbs

COORDINATOR_DDL_REQUEST_TIMEOUT

Timeout until the Coordinator waits after requesting the Node to execute DDL
This value refers to the time the Coordinator waits after requesting each Node to perform DDL.

(usec)	Value
Minimum	0
Maximum	2^64 - 1
Default	300000000

COORDINATOR_DDL_TIMEOUT

Timeout until the Broker waits after requesting Coordinator to perform DDL through CC
This value refers to the time that the Broker waits after requesting the Coordinator to perform DDL for the entire Cluster.

(usec)	Value
Minimum	0
Maximum	2^64 - 1
Default	3600000000

COORDINATOR_DECISION_DELAY

Timeout until the Coordinator requests the status change and effectively reflects it.
If the status does not actually change over this time, disable the cluster status.
If the status of the Warehouse Active is not changed but the connected Standby exists, the Fail-Over operation starts.

(usec)	Value
Minimum	0
Maximum	2^64 - 1
Default	1000000

COORDINATOR_DECISION_INTERVAL

Time to determine how often the Coordinator changes status

(usec)	Value
Minimum	0
Maximum	2^64 - 1
Default	1000000

COORDINATOR_HOST_RESOURCE_COLLECT_INTERVAL

Interval at which Cluster Nodes collect Host Resources

(usec)	Value
Minimum	0
Maximum	2^64 - 1
Default	1000000

COORDINATOR_HOST_RESOURCE_INTERVAL

Interval at which the Coordinator exchanges Host Resources with Nodes

(usec)	Value
Minimum	0
Maximum	2^64 - 1
Default	1000000

COORDINATOR_HOST_RESOURCE_REQUEST_TIMEOUT

Time that the Coordinator waits after requesting the Host Resource information from the Nodes

(usec)	Value
Minimum	0
Maximum	2^64 - 1
Default	10000000

COORDINATOR_NODE_REQUEST_TIMEOUT

Timeout until the Coordinator waits after requesting the Node to execute the command
Because the Add/Remove-node and Add/Remove-Package includes the Node command execution, if it is caught in a short time, the command processing may not be completed.

(usec)	Value
Minimum	0
Maximum	2^64 - 1
Default	600000000

COORDINATOR_NODE_TIMEOUT

Time the Coordinator waits before determining that the Node has failed

(usec)	Value
Minimum	0
Maximum	2^64 - 1
Default	30000000

COORDINATOR_STARTUP_DELAY

Grace time until activating the Decision Thread immediately after Coordinator startup
If it takes a long time to run the entire Cluster, you can start the Node control of Coordinator later by setting a larger value.
If the Decision Thread runs before the entire drive, there is a high likelihood that the Coordinator will be misplaced.

(usec)	Value
Minimum	0
Maximum	2^64 - 1
Default	3000000

COORDINATOR_STATUS_NODE_INTERVAL

Interval in which the Coordinator exchanges status inquiry messages with the Node

(usec)	Value
Minimum	0
Maximum	2^64 - 1
Default	1000000

COORDINATOR_STATUS_NODE_REQUEST_TIMEOUT

Time the Coordinator waits after requesting status inquiries from Nodes.
If there is no status inquiry response during that time, the Coordinator proceeds without updating the status of the corresponding Node.
If the network situation is not good and you need to update the state, you could consider increasing the value.
Instead, if there is no status query response, the Coordinator will wait for as much as the value was increased.

(usec)	Value
Minimum	0
Maximum	2^64 - 1
Default	3000000

DEPLOYER_DBS_PATH

Specifies the directory where the Deployer's data files will be created.
The default value is set to?/dbs, and ? is replaced with the $ MACHBASE_DEPLOYER_HOME environment variable.

This is an environment variable `$MACHBASE_DEPLOYER_HOME /dbs` directory.
It must be applied to Deployer, and it has no effect on other Nodes.

(path)	Value
Default	?/dbs

EXECUTION_STAGE_MEMORY_MAX

The maximum amount of Memory used by the Stage Thread performing the SELECT query in Cluster Edition
Because it is the maximum size of each Stage, the complexity of the SELECT query with an increase in the number of Stages can lead to a larger memory requirement.
If there is a Stage that exceeds the maximum size, the Stage is canceled and the Query is canceled with an error.
You must apply Properties directly to the specific Warehouse you want.
The default value is 1GB.

(size)	Value
Minimum	1024
Maximum	2^64 - 1
Default	1024 *1024 * 1024

HTTP_ADMIN_PORT

Port number to receive requests from MWA or machcoordinatoradmin

(port)	Value
Minimum	1024
Maximum	65535
Default	5779

HTTP_CONNECT_TIMEOUT

Timeout used when connecting to machcoordinatoradmin

(usec)	Value
Minimum	0
Maximum	2^64 - 1
Default	30000000

HTTP_RECEIVE_TIMEOUT

Timeout used when communicating with machcoordinatoradmin

(usec)	Value
Minimum	0
Maximum	2^64 - 1
Default	3600000000

HTTP_SEND_TIMEOUT

Timeout used when communicating with machcoordinatoradmin

(usec)	Value
Minimum	0
Maximum	2^64 - 1
Default	60000000

INSERT_BALANCE_MODE

When append or INSERT-SELECT is executed, it is instructed to input uniformly to each group.
If it is 0, it turns off the feature. If it is 1, it turns on the feature.

> ⚠ When the corresponding function is turned on, the Group receiving data is limited. This is because the input to the disk scarce Warehouse will not be completed.
> Therefore, the overall property of the input may seem to be degraded, so the default value of that Property is zero.

(mode)	Value
Minimum	0
Maximum	1
Default	0

INSERT_BULK_DATA_MAX_SIZE

Maximum size of input data block when executing Append or INSERT-SELECT

(size)	Value
Minimum	1024
Maximum	10 * 1024 * 1024
Default	1024 * 1024

INSERT_RECORD_COUNT_PER_NODE

Number of data inputs that lead to the warehouse group conversion when performing the input

(count)	Value
Minimum	1
Maximum	2^64 - 1
Default	10000

STAGE_RESULT_BLOCK_SIZE

Maximum block size created in one stage

(size)	Value
Minimum	1024
Maximum	2^64 - 1
Default	1024 * 1024

SQL Reference

Datatypes

Data Type Table

Index

Type Name	Description	Value Range	NULL Value
short	16-bit signed integer data type	-32767 ~ 32767	-32768
ushort	16-bit unsigned integer type data type	0 ~ 65534	65535
integer	32-bit signed integer data type	-2147483647 ~ 2147483647	-2147483648
uinteger	32-bit unsigned integer data type	0 ~ 4294967294	4294967295
long	64-bit signed integer data type	-9223372036854775807 ~ 9223372036854775807	-9223372036854775808
ulong	64-bit unsigned integer data type	0~18446744073709551614	18446744073709551615
float	32-bit floating point data type	-	-
double	64-bit floating point data type	-	-
datetime	Time and date	1970-01-01 00:00:00 000:000:000 ~	-
varchar	Variable-length character strings (UTF-8)	Length : 1 ~ 32768 (32K)	-
ipv4	Version 4 Internet address type (4 bytes)	"0.0.0.0" ~ "255.255.255.255"	-
ipv6	Version 6 Internet address type (16 bytes)	"0000:0000:0000:0000:0000:0000:0000:0000" ~ "FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF"	-
text	Text data type (keyword index can be generated)	Length : 0 ~ 64M	-
binary	Binary data type (index creation not possible)	Length: 0 ~ 64M	-

short

This is the same as the 16-bit signed integer data of the C language. For the minimum negative value, it is recognized as NULL. May be displayed as "int16".

integer

This is the same as 32-bit signed integer data in C language. For the minimum negative value, it is recognized as NULL. May be displayed as "int32" or "int".

long

This is the same as 64-bit signed integer data in C language. For the minimum negative value, it is recognized as NULL. May be displayed as "int64".

float

This is equivalent to the C 32-bit floating-point data type float. For a positive maximum value, it is recognized as NULL.

double

This is equivalent to the 64-bit floating-point data type double of C language. For a positive maximum value, it is recognized as NULL.

datetime

In Machbase, this type maintains the nano value of the time elapsed since midnight January 1, 1970.

Thus, Machbase provides the ability to process values up to nano units for all datetime type related functions.

varchar

This is a variable string data type and can be generated up to 32K bytes in length.

This length criterion is based on one character in English, so it is different from the actual number of characters to be output in UTF-8 and should be set to an appropriate length.

IPv4

This type is a type that can store addresses used in Internet Protocol version 4.

It is internally represented using 4 bytes, and can be expressed from "0.0.0.0" to "255.255.255.255".

IPv6

This type is a type that can store addresses used in Internet Protocol version 6.

16 bytes are internally represented and can be expressed from "0000: 0000: 0000: 0000: 0000: 0000: 0000: 0000" to "FFFF: FFFF: FFFF: FFFF: FFFF: FFFF: FFFF: FFFF" . Since the abbreviation type is also supported when inputting data, it can be expressed as follows using the symbol `::`.

- ":: FFFF: 1232": all leading with zeros
- ":: FFFF: 192.168.0.3": Support for IPv4 type compatibility
- ":: 192.168.3.1": Support for deprecated IPv4 type compatibility

text

This type is a data type for storing text or documents beyond the size of a VARCHAR.

This data type can be searched through keyword indexes and can store up to 64 megabytes of text.
This type is mainly used to store and retrieve large text files as separate columns.

binary

This type is a supported type for storing unstructured data in columns.

It is used to store binary data such as image, video, or audio. Indexes can not be created for this type.
The maximum data size for storing is up to 64 megabytes, the same as the TEXT type.

SQL Datatype Table

The following table shows the SQL data types and C data types corresponding to the mark base data types.

Machbase Datatype	Machbase CLI Datatype	SQL Datatype	C Datatype	Basic types for C	Description
short	SQL_SMALLINT	SQL_SMALLINT	SQL_C_SSHORT	int16_t (short)	16-bit signed integer data type
ushort	SQL_USMALLINT	SQL_SMALLINT	SQL_C_USHORT	uint16_t (unsigned short)	16-bit unsigned integer type data type
integer	SQL_INTEGER	SQL_INTEGER	SQL_C_SLONG	int32_t (int)	32-bit signed integer data type
uinteger	SQL_UINTEGER	SQL_INTEGER	SQL_C_ULONG	uint32_t (unsigned int)	32-bit unsigned integer data type
long	SQL_BIGINT	SQL_BIGINT	SQL_C_SBIGINT	int64_t (long long)	64-bit signed integer data type
ulong	SQL_UBIGINT	SQL_BIGINT	SQL_C_UBIGINT	uint64_t (unsigned long long)	64-bit unsigned integer data type
float	SQL_FLOAT	SQL_REAL	SQL_C_FLOAT	float	32-bit floating point data type
double	SQL_DOUBLE	SQL_FLOAT, SQL_DOUBLE	SQL_C_DOUBLE	double	64-bit floating point data type
datetime	SQL_TIMESTAMP SQL_TIME	SQL_TYPE_TIMESTAMP SQL_BIGINT SQL_TYPE_TIME	SQL_C_TYPE_TIMESTAMP SQL_C_UBIGINT SQL_C_TIME	char * (YYYY-MM-DD HH24:MI:SS) int64_t (timestamp: nano seconds)	Time and date

Machbase Datatype	Machbase CLI Datatype	SQL Datatype	C Datatype	Basic types for C	Description
				struct tm	
varchar	SQL_VARCHAR	SQL_VARCHAR	SQL_C_CHAR	char *	String
ipv4	SQL_IPV4	SQL_VARCHAR	SQL_C_CHAR	char * (enter ip string) unsigned char[4]	Version 4 Internet address type
ipv6	SQL_IPV6	SQL_VARCHAR	SQL_C_CHAR	char * (enter ip string) unsigned char[16]	Version 6 Internet address type
text	SQL_TEXT	SQL_LONGVARCHAR	SQL_C_CHAR	char *	Text
binary	SQL_BINARY	SQL_BINARY	SQL_C_BINARY	char *	Binary data

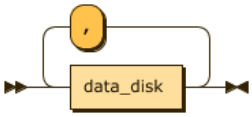
DDL

CREATE TABLESPACE

Index

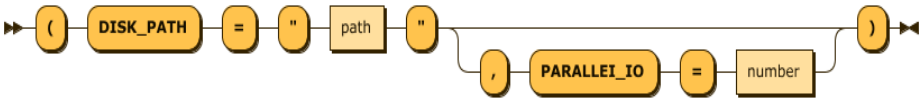
create_tablespace_stmt:

datadisk_list:



data_disk:

data_disk_property:




```
create_tablespace_stmt ::= 'CREATE TABLESPACE' tablespace_name 'DATADISK' datac
datadisk_list ::= data_disk ( ',' data_disk )*
data_disk ::= disk_name data_disk_property
data_disk_property ::= '(' 'DISK_PATH' '=' '"' path '"' ( ',' 'PARALLEL_IO' '='
```

```
-- Example
create tablespace tbs1 datadisk disk1 (disk_path=""); -- $MACHBASE_HOME/dbs/  (
create tablespace tbs1 datadisk disk1 (disk_path="tbs1_disk1"); -- $MACHBASE_HO
create tablespace tbs2 datadisk disk1 (disk_path="tbs2_disk1", parallel_io = 5)
create tablespace tbs1 datadisk disk1 (disk_path="tbs1_disk1", parallel_io = 10
```

The CREATE TABLESPACE statement creates a tablespace in $MACHBASE_HOME/dbs/ where the indexes of the log table or log table will be stored.

Tablespace can have multiple disks. When each Partition File that stores data of Table and Index is stored, it is distributed and stored in Data Disks belonging to Tablespace.
If two or more disks are used, the index and table files are distributed and stored on each disk, and I/O is performed in parallel on each device. As the number of disks increases, disk I / O throughput increases, and a large amount of data can be stored on the disk quickly
Also, if tables and index tablespace are separately created and different disks are defined, I/O of table and index can be logically separated without reconfiguration of physical disk.

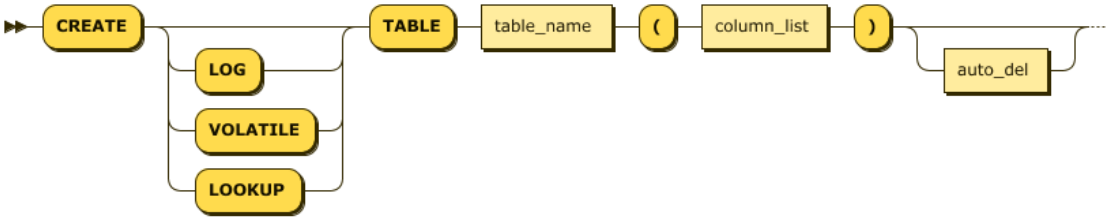
DATA DISK

Defines disk belonging to a tablespace. Each Disk has the following properties.

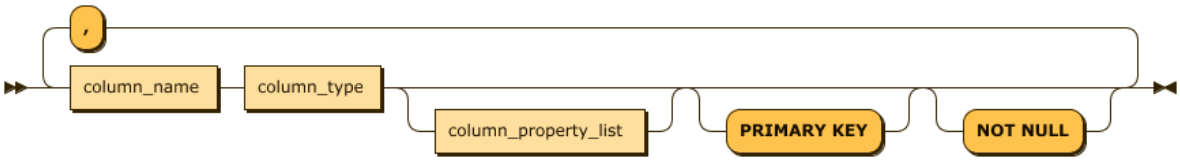
Property	Description
disk_name	Specifies the name of the Disk object. It is used to change the attributes of the Disk object through Alter Tablespace syntax later.
data_disk_property	Specifies the attributes of the disk.
disk_path	Specifies the Directory Path of the disk. This Directory must be created. When a path is specified as a relative path, PATH is searched based on $MACHBASE_HOME/dbs. For example, if PATH = 'disk1', Disk Path is recognized as $MACHBASE_HOME/dbs/disk1.
parallel_io	Determines how many disk IO requests are allowed to be paralleled. (DEF: 3, MIN: 1, MAX: 128)

CREATE TABLE

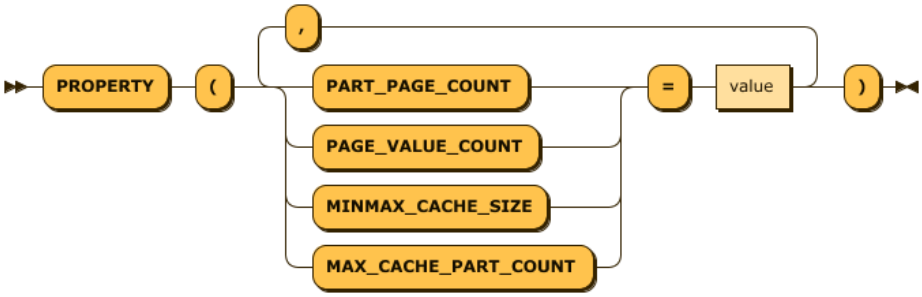
create_table_stmt:



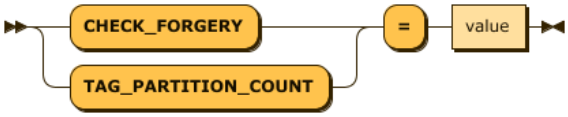
column_list:



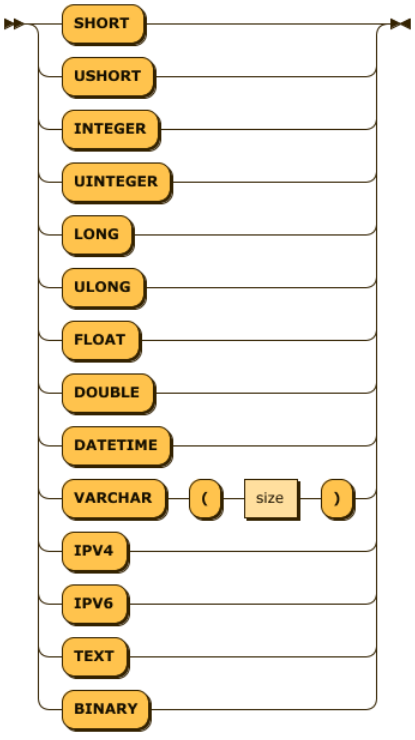
column_property_list:



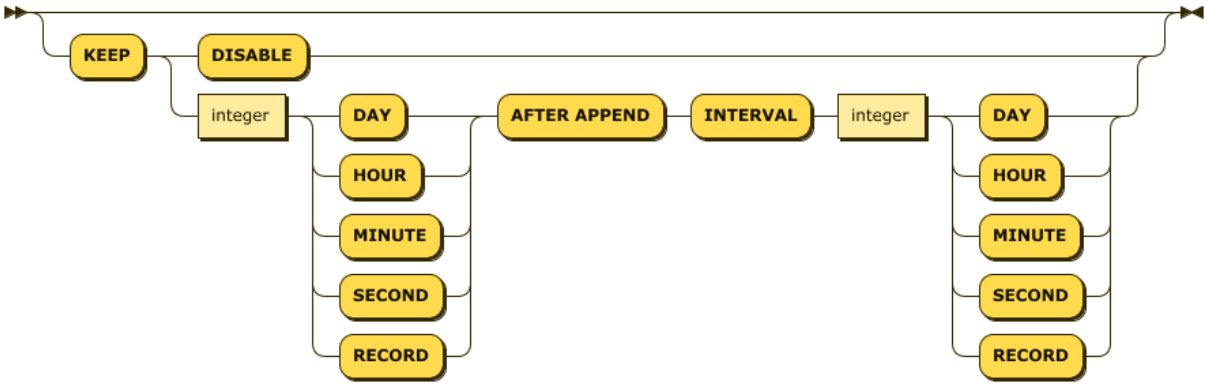
table_property_list:



column_type:



auto_del:




```
create_table_stmt ::= 'CREATE' ( 'LOG' | 'VOLATILE' | 'LOOKUP' )? 'TABLE' table_name '(' column_list ')' table_prop
column_list ::= column_name column_type column_property_list? 'PRIMARY KEY'? 'NOT NULL'? ( ',' column_name column_t
column_property_list ::= 'PROPERTY' '(' ( 'PART_PAGE_COUNT' | 'PAGE_VALUE_COUNT' | 'MINMAX_CACHE_SIZE' | 'MAX_CACHE
table_property_list ::=  ( 'CHECK_FORGERY' | 'TAG_PARTITION_COUNT' ) '=' value
column_type ::= 'SHORT' | 'USHORT' | 'INTEGER' | 'UINTEGER' | 'LONG' | 'ULONG' | 'FLOAT' | 'DOUBLE' | 'DATETIME' |
```

```
-- Create ctest table with 5 columns
Mach> create table ctest (id integer, name varchar(20), sipv4 ipv4,dipv6 ipv6,comment text);
Created successfully.
```

Table Type

Table Type	Description
LOG_TABLE	If there is no keyword between CREATE TABLE, a log table is created.
VOLATILE_TABLE	VOLATILE_TABLE is a temporary table in which all data resides in temporary memory and joins the log table to improve the results, The Machbase server disappears as soon as it is shut down.
LOOKUP_TABLE	Like VOLATILE_TABLE, LOOKUP_TABLE can perform fast query processing by storing all the data in memory.

Data Column

A table column is usually a string, but it can be enclosed in single quotes (') or double quotes (") to contain special characters.

```
Mach> CREATE TABLE special_tbl ( with.dot INTEGER );
[ERR-02010: Syntax error: near token (.dot INTEGER )).]
CREATE TABLE special_tbl ( "with.dot" INTEGER ); -- (Possible)
CREATE TABLE special_tbl ( 'with.dot' INTEGER ); -- (Possible)
```

When querying the column through a SELECT query, it must be enclosed in either single or double quotes.

- When enclosed in single quotation marks, the name of the SELECT query result column is output with single quotation marks.
- When enclosed in double quotes, the name of the SELECT query result column is output as it is.

```
Mach> SELECT 'with.dot' FROM special_tbl;
'with.dot'
--------------
[0] row(s) selected.

Mach> SELECT "with.dot" FROM special_tbl;
with.dot
--------------
[0] row(s) selected.
```

Table Property

Specifies the attributes for the table.

Property Name	Available Table Types
CHECK_FORGERY	LOG TABLE

CHECK_FORGERY(Default:0)

This is a property supported only for Disk Column Table. If appended data is maliciously changed later, it can check whether data is changed or not. Check the ALTER TABLE CHECK FORGERY RESULTFILE syntax.

Column Property

Specifies the attribute for the column.

Property Name	Available Table Types
PART_PAGE_COUNT	LOG TABLE
PAGE_VALUE_COUNT	LOG TABLE
MAX_CACHE_PART_COUNT	LOG TABLE
MINMAX_CACHE_SIZE	LOG TABLE

PART_PAGE_COUNT

This property represents the number of pages a partition has. The number of values that a partition has is PART_PAGE_COUNT * PAGE_VALUE_COUNT.

PAGE_VALUE_COUNT

This property represents the number of values that a page has.

MAX_CACHE_PART_COUNT (Default : 0)

This property sets the cache area for performance.

When Machbase accesses a partition, it first looks for a structure that contains the meta information of that partition in memory. It determines how many partition information it contains in memory. Larger size will help performance, but memory usage will increase. The minimum value is 1 and the maximum value is 65535.

MINMAX_CACHE_SIZE (Default : 10240)

This property specifies how much cache memory to use for the MINMAX of the corresponding column. The default is 100MB for _ARRIVAL_TIME, the 0th hidden column. However, other columns are specified as 10KB by default. This size can be changed after the creation of the table through the "ALTER TABLE MODIFY" statement.

NOT NULL Constraint

Specifies NOT NULL if the column value does not allow NULL, and omit it if it is allowed (Default).

You can change the constraint with the ALTER TABLE MODIFY COLUMN command to drop or add this constraint defined after the creation of the table.

```
# Column c1 is not null and c2 is created without not null constraint.
CREATE TABLE t1(c1 INTEGER NOT NULL, c2 VARCHAR(200));
```

Pre-defined System Columns

When you create a table using the Create Table statement, the system creates two additional predefined system columns. _ARRIVAL_TIME and _RID columns.

The _ARRIVAL_TIME column is inserted into the DATETIME column based on the system time at which data is inserted into the INSERT statement or AppendData, and the value can be used as the unique key of the generated record. The value of this column can be inserted by specifying the value in the machloader or INSERT statement if the order is guaranteed (in the order of past-present). When data is retrieved using the DURATION conditional expression, data is retrieved based on the value of this column.
The _RID column is created by the system as a unique value for a particular record. The data type of this column is a 64-bit integer. For this column, the user can not specify a value and can not create an index. It is automatically generated at the time of data INSERT. You can retrieve records by the value of the _RID column.

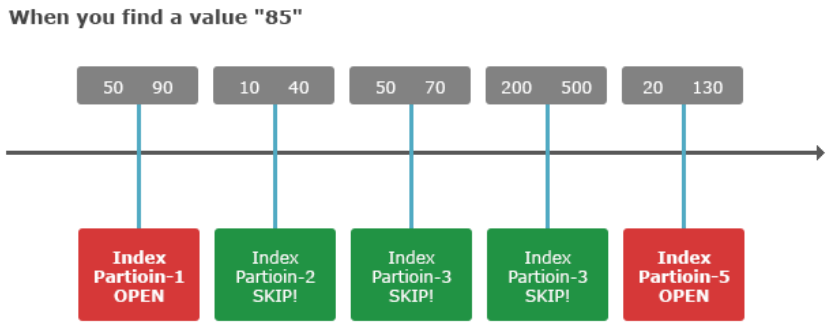
```
create volatile table t1111 (i1 integer);
Created successfully.
Mach> desc t1111;

---------------------------------------------------------------
NAME                          TYPE              LENGTH
```

```
----------------------------------------------------------------
_ARRIVAL_TIME                datetime           8
I1                           integer            4

Mach>insert into t1111 values (1);
1 row(s) inserted.
Mach>select _rid from t1111;
_rid
----------------------
0
[1] row(s) inserted.

Mach>select i1 from t1111 where _rid = 0;
i1
--------------
1
[1] row(s) selected.
```

MINMAX Cache Concept

In general, in the Disk DBMS, when a specific value is searched using the index, the disk is accessed to access the disk area including the index, and the final disk page including the corresponding value is searched.

On the other hand, Machbase is a chronologically partitioned structure in order to maintain time series information, which means that a particular piece of index information is divided into chunks of files in chronological order. Therefore, when a Machbase index is used, an index file fragmented by such a partition is sequentially searched.
If the range of data to be searched is divided into 1000 partitions, it means that 1000 files should be opened and retrieved every time. Although it is designed as an efficient columnar database structure, the MINMAX_CACHE structure is a way to improve the performance because the I/O cost is proportional to the number of index partitions.
MINMAX_CACHE is a structure that holds the index file information of the partition in memory, and is a contiguous memory space that keeps the minimum and maximum values of the column in memory. By maintaining such structure, when a partition containing a specific value is searched, if the value is smaller than the minimum value of the index or is larger than the maximum value, the corresponding partition can be skipped altogether, thereby enabling high-performance data analysis.



As shown in the figure above, to find the value 85, only the partitions 1 and 5 included in MIN/MAX among the 5 partitions are actually searched, and the partitions 2, 3 and 4 are skipped altogether.

MINMAX Cache Column

You can decide whether to use MINMAX Cache for a particular column when creating the table.

If the minmax_cache_size is set to a value other than 0, the MINMAX Cache will be active when the index is searched for that column and will not be active if `MINMAX_CACHE_SIZE` = 0.
Please note the following when using this MINMAX Cache.

1. MINMAX Cache does not need to explicitly create an index on the column.
2. As default for all columns, `MINMAX_CACHE_SIZE` is set to 10KB and the Alter Table syntax can be used to reset the memory size to a reasonable size.
3. The hidden column _arrival_time is 100MB by default and automatically uses MINMAX Cache memory.
4. In the case of VARCHAR type, MINMAX Cache is not covered. Therefore, if you explicitly specify whether the VARCHAR type is cached, an error will occur.
5. When the corresponding table is created, the `MINMAX_CACHE_SIZE` maximum memory can be used as much as the property is set. As the number of partitions grows, the memory grows gradually and increases by the maximum memory above.
6. If there are no records in the table, MINMAX Cache memory is not allocated at all.

Below is an example of table creation using actual MINMAX.

```
-- MINMAX_CACHE_SIZE = 0 for VARCHAR is allowed semantically.
CREATE TABLE ctest (id INTEGER, name VARCHAR(100) PROPERTY(MINMAX_CACHE_SIZE = 0));
```

```
Created successfully.
Mach>

-- Cache applied to id column.
CREATE TABLE ctest2 (id INTEGER PROPERTY(MINMAX_CACHE_SIZE = 10240), name VARCHAR(100) PROPERTY(MINMAX_CACHE_SIZE =
Created successfully.
Mach>

-- Applied to id1, id2, and id3.
CREATE TABLE ctest3 (id1 INTEGER PROPERTY(MINMAX_CACHE_SIZE = 10240), name VARCHAR(100) PROPERTY(MINMAX_CACHE_SIZE
Created successfully.
Mach>

-- MINMAX_CACHE_SIZE is specified in column units or set to 0.
CREATE TABLE ctest4 (id1 INTEGER PROPERTY(MINMAX_CACHE_SIZE=10240), name VARCHAR(100) PROPERTY(MINMAX_CACHE_SIZE=0)
Created successfully.
Mach>
```

Primary Key

This is a constraint that can be assigned to a Volatile/Lookup table column. The Volatile / Lookup table does not always need to have a primary key, but you can not use the INSERT ON DUPLICATE KEY UPDATE statement without a primary key.

When a primary key is assigned, a red-black tree index corresponding to the primary key is generated.

AUTO_DEL Clauses

> Supported version 5.5 or later.

This feature limits the amount of data to maintain disk storage space. It is only supported for log table. It is set by using AUTO_DEL clause before specifying table property in CREATE TABLE statement. The AUTO_DEL clause can be set based on the storage time or the number of records.

```
CREATE TABLE t1 (c1 INT) KEEP 30 DAY AFTER APPEND INTERVAL 5 SECOND;
```

The above example deletes data that is more than 30 days old if there is more input five seconds after the automatic deletion is performed. If the interval specified is too long, auto delete is performed for a long time, which may affect the input. If it is too short, it may affect overall system performance.

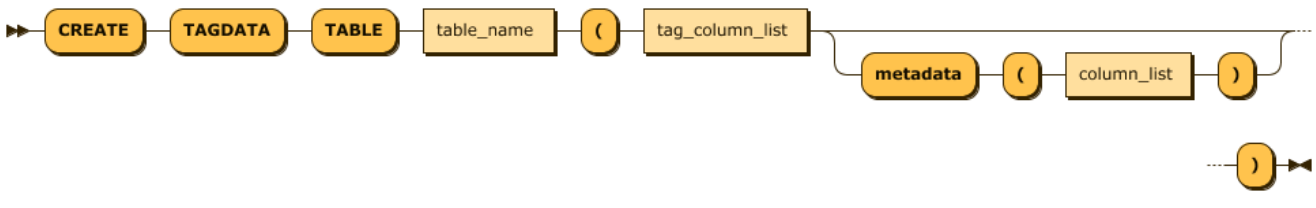
The following example shows how to specify the automatic deletion function as the number of data to save.

```
CREATE TABLE t1 (c1 INT) KEEP 3 RECORD AFTER APPEND INTERVAL 5 RECORD;
```

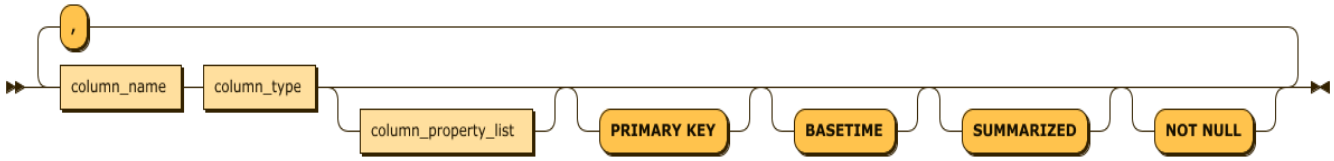
It checks the number of records in the table for every 5 input data and automatically deletes it and keep only 3 records if there are more than three.

CREATE TAG TABLE

create_tag_table_stmt:



tag_column_list:



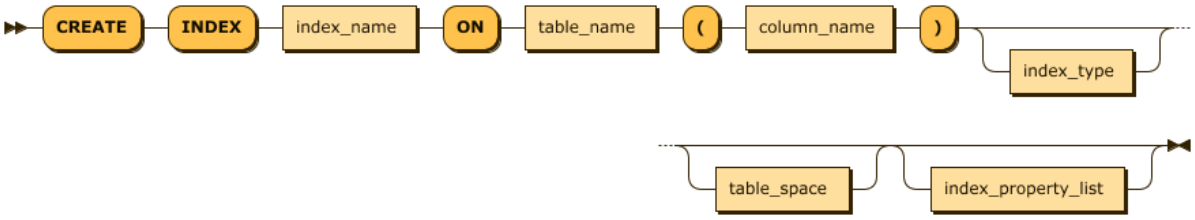

```
create_tag_table_stmt ::= 'CREATE' 'TAGDATA' 'TABLE' table_name '(' tag_column_list ( 'metadata' '(' column_list ')
tag_column_list ::= column_name column_type column_property_list? 'PRIMARY KEY'? 'BASETIME'? 'SUMMARIZED'? 'NOT NUL
```

The primary key, basetime, and summarized must be included in the tag table creation.

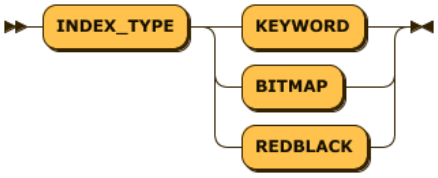
```
-- Example
CREATE TAGDATA TABLE tag (name varchar(20) primary key, time datetime basetime, value double summarized);
CREATE TAGDATA TABLE tag (name varchar(20) primary key, int_column int, time datetime basetime, value double summa
CREATE TAGDATA TABLE tag (name varchar(20) primary key, time datetime basetime, value double summarized, value2 flo
```

CREATE INDEX

create_index_stmt:

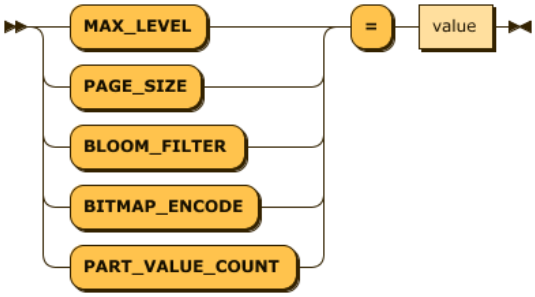


index_type:



table_space:

index_property_list:




```
create_index_stmt ::= 'CREATE' 'INDEX' index_name 'ON' table_name '(' column_name ')' index_type? table_space? inde
index_type ::= 'INDEX_TYPE' ( 'KEYWORD' | 'BITMAP' | 'REDBLACK' )
table_space ::= 'TABLESPACE' table_space_name
index_property_list ::= ( 'MAX_LEVEL' | 'PAGE_SIZE' | 'BLOOM_FILTER' | 'BITMAP_ENCODE' | 'PART_VALUE_COUNT' ) '=' v
```

Index Type

Specifies the Index Type to be created. If it is not Keyword Index, Index Type is created as Default Index Type according to Table Type if Index Type is not specified.

Table Type	Default Index Type
Volatile Table	REDBLACK
Lookup Table	REDBLACK
Log Table	LSM

KEYWORD Index

This can be created only for varchar and text column of log table. It can be created for only one column.

LSM Index

LSM (Log Structure Merge) Index is an index optimized for storing and searching Big Data. The partitions of the LSM indexes are maintained for each level, and the lower level partitions are merged to move to the upper level. Lower partitions used to create a higher level partition are deleted.

This Index Level Partition Building is performed by Background Thread. The upper level partitions are merged with the lower level partitions and are created as one partition, so there are the following advantages when searching through the index.

1. If the key is duplicated, the disk space for key storage is saved because it is stored only once.
2. Searching for multiple partitions reduces the cost of opening and closing the file when searching for one index partition, and the number of index pages accessed is also reduced.

LSM Index Property

Item	Description
MAX_LEVEL (DEFAULT = 3, MIN = 0, MAX = 3)	The maximum level of the LSM Index, and the current value of 3 is the maximum value. And the maximum number of records of one partition can not exceed 200 million. The partition size of each level is the number of values of the previous partition * 10. For example, if MAX_LEVEL = 3 and PART_VALUE_COUNT is 100,000, then Level 0 = 100,000, Level 1 = 1,000,0000, Level 2 = 10,000,000, and Level 3 = 100,000,000. If the Partition Size of the last level exceeds 200 million, index creation will fail.
PAGE_SIZE (DEFAULT = 512 * 1024, MIN = 32 * 1024, MAX = 1 * 1024 * 1024)	Specifies the size of the page in which the index key value and bitmap value are stored. Default is 512K.
BLOOM_FILTER (DEFAULT = 1, DISABLE = 0, ENABLE(DEFAULT) = non_zero_integer)	Sets whether to set Bloom filter on index. Setting the Bloom filter allows you to quickly search for values that do not exist in the index, but it will increase the time it takes to create the index. If you use only Range conditions, Bloom filters are not available and need not be created. If you do not want to create a Bloom filter, you should set BLOOM_FILTER = 0.
BITMAP_ENCODE (DEFAULT = EQUAL, RANGE)	Sets the bitmap type of the index. If BITMAP_ENCODE = EQUAL (default), generates a bitmap for the same value as the key value. If BITMAP = RANGE, generates a bitmap according to the range of the key value. It is better to set as BITMAP_ENCODE = EQUAL when using = as the query condition, and BITMAP_ENCODE = RANGE when using the specific range value as the query condition. In the case of BITMAP = RANGE, the cost of creation increases slightly compared to EQUAL.

BITMAP Index

This is an index for data analysis and can be created only in the log table. It can be created on all columns except varchar, text, and binary, and can only be created on a single column.

RED-BLACK Index

This is a memory index for real-time data retrieval. It can be created only in the Volatile/Lookup table. It can be created in all columns of this table and can only be created for a single column.

Index Property

The properties that can be applied in the LSM Index are as follows.

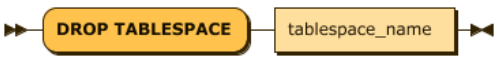
PART_VALUE_COUNT

Indicates the number of rows stored in the Partition of Index.

```
-- Example
-- Index applied to c1 column.
CREATE INDEX index1 on table1 ( c1 )
-- Keyword index applied to var_column of varchar type, and page_size unit is 100000.
CREATE INDEX index2 on table1 (var_column) INDEX_TYPE KEYWORD PAGE_SIZE=100000;
```

DROP TABLESPACE

drop_table_stmt:



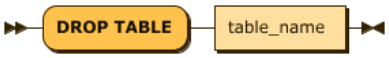
```
drop_table_stmt ::= 'DROP TABLESPACE' tablespace_name
```

Deletes the specified tablespace. However, if the object created in Tablespace exists, deletion fails.

```
-- Example
DROP TABLESPACE TablespaceName;
```

DROP TABLE

drop_table_stmt:



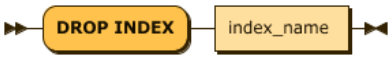

```
drop_table_stmt ::= 'DROP TABLE' table_name
```

Deletes the specified table. However, if there is another session in which the table is being searched, it fails with an error.

```
-- Example
DROP TABLE TableName;
```

DROP INDEX

drop_index_stmt:




```
drop_index_stmt ::= 'DROP INDEX' index_name
```

Deletes the specified index. However, if there is another session in which the table is being searched, it fails with an error.

```
-- Example
DROP INDEX IndexName;
```

ALTER TABLESPACE

The ALTER TABLESPACE statement is used to change the information associated with the specified tablespace.

ALTER TABLESPACE MODIFY DATADISK

This syntax is used to change the properties of DATADISK in Tablespace.

alter_tablespace_stmt:


```
alter_tablespace_stmt ::= 'ALTER TABLESPACE' table_name 'MODIFY DATADISK' disk_name 'SET' 'PARALLEL_IO' '=' value
```

```
-- Example
ALTER TABLESPACE tbs1 MODIFY DATADISK disk1 SET PARALLEL_IO = 10;
```

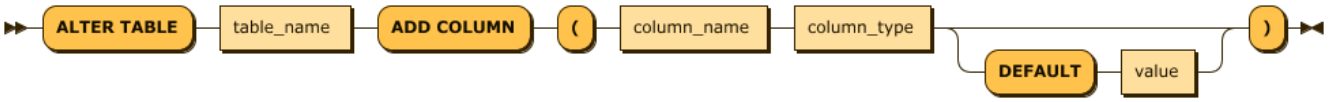
The ALTER TABLE statement is used to change the schema information of the specified table, and only the Log Table is available.

ALTER TABLE SET

This syntax changes the properties of a table. Currently there are no dynamically changeable properties.

ALTER TABLE ADD COLUMN

alter_table_add_stmt:




```
alter_table_add_stmt ::= 'ALTER TABLE' table_name 'ADD COLUMN' '(' column_name column_type ( 'DEFAULT' value )? ')
```

This syntax is the ability to add a specific column to the table in real time. You can add the name and type of the column, and set the default data values through the DEFAULT clause.

```
-- Example-1
alter table atest2 add column (id4 float);

-- Example-2
alter table atest2 add column (id6 double  default 5);
alter table atest2 add column (id7 ipv4  default '192.168.0.1');
alter table atest2 add column (id8 varchar(4) default 'hello');
```

ALTER TABLE DROP COLUMN

alter_table_drop_stmt:


```
alter_table_drop_stmt ::= 'ALTER TABLE' table_name 'DROP COLUMN' '(' column_name ')'
```

This syntax is to delete a specific column in the table in real time.

```
-- Example
alter table atest2 drop column (id4);
alter table atest2 drop column (id8);
```

ALTER TABLE RENAME COLUMN

alter_table_column_rename_stmt:

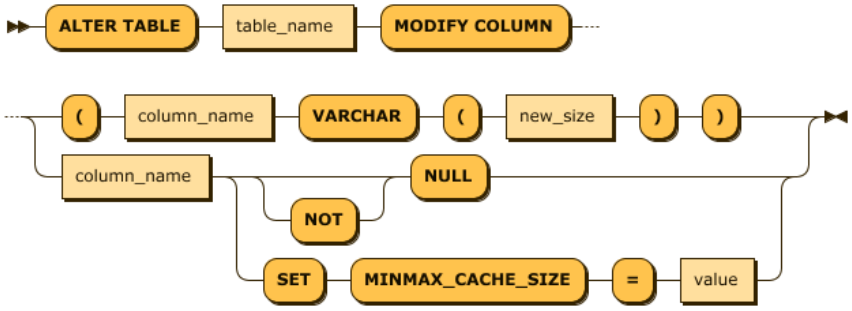

```
alter_table_column_rename_stmt ::= 'ALTER TABLE' table_name 'RENAME COLUMN' old_column_name 'TO' new_column_name
```

This syntax is a function that changes a specific column name in a table.

```
-- Example
alter table atest2 rename column id7 to id7_rename;
```

ALTER TABLE MODIFY COLUMN

alter_table_modify_stmt:




```
alter_table_modify_stmt ::= 'ALTER TABLE' table_name 'MODIFY COLUMN' ( '(' column_name 'VARCHAR' '(' new_size ')'
```

This syntax changes the properties of a particular column of a table. Currently it is possible to modify MINMAX CACHE attributes and NOT NULL constraints for column lengths and other types of VARCHAR types.

VARCHAR SIZE

This syntax supports changing the column length of VARCHAR type only. This operation can not be reduced in length to preserve existing data, and should always be increased.

```
ALTER TABLE table_name MODIFY COLUMN (column_name VARCHAR(new_size));
```

```
-- Example: Assume TABLE is created like this.
-- create table atest5 (id integer, name varchar(5), id3 double, id4 float);

-- Error occurred: Can not change to another type,
alter table atest5 modify column (id varchar(10));

-- Error occurred: VARCHAR length can not be made smaller.
alter table atest5 modify column (name varchar(3));

-- Error occurred: Maximum size of VARCHAR can not exceed 32767.
alter table atest5 modify column (name varchar(32768));

-- Success
alter table atest5 modify column (name varchar(128));
```

MINMAX_CACHE_SIZE

This syntax changes MINMAX_CACHE_SIZE for a particular column.

```
ALTER TABLE table_name MODIFY COLUMN column_name SET MINMAX_CACHE_SIZE=value;
```

```
-- Example: Assume TABLE is created like this.
create table atest9 (id integer, name varchar(100));

-- Error: Does not apply to VARCHAR.
alter table atest9 modify column name set minmax_cache_size=0;
[ERR-02139 : MINMAX CACHE is not allowed for VARCHAR column(NAME).]

-- Change success
alter table atest9 modify column id set minmax_cache_size=10240;
```

NOT NULL

Adds a NOT NULL constraint to the column. If you add a NOT NULL constraint, the DDL operation fails for columns with NULL values.

If you want to allow NULL values in a column, use the MODIFY COLUMN NULL command in the next section.

```
ALTER TABLE table_name MODIFY COLUMN column_name NOT NULL;
```

```
-- Add NOT NULL constraint to t1.c1.
alter table t1 modify column c1 not null;
```

NULL

Releases the NOT NULL constraint. Performance improvement due to min_max cache of LSM index can not be obtained. NULL values can be input.

```
ALTER TABLE table_name MODIFY COLUMN column_name NULL;
```

```
-- Release NOT NULL constraint at t1.c1.
alter table t1 modify column c1 null;
```

ALTER TABLE FLUSH

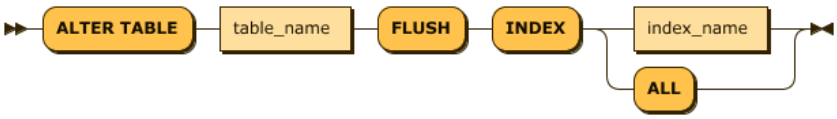
alter_table_flush_stmt:


```
alter_table_flush_stmt ::= 'ALTER TABLE' table_name 'FLUSH'
```

Waits until the input data for the specified table is fully reflected in the data file.

ALTER TABLE FLUSH INDEX

alter_table_flush_index_stmt:




```
alter_table_flush_index_stmt ::= 'ALTER TABLE' table_name 'FLUSH' 'INDEX' ( index_name | 'ALL' )
```

Waits until the index data of the specified table is completely reflected in the index file.

```
-- Wait until data for all indexes of bulktable is reflected in index files.
alter table bulktable flush index all;

-- Wait until data of c1_idx index of bulktable is reflected in index file.
alter table bulktable flush index c1_idx;
```

ALTER TABLE CHECK FORGERY RESULT FILE

alter_table_check_forgery_stmt:


```
alter_table_check_forgery_stmt ::= 'ALTER TABLE' table_name 'CHECK FORGERY RESULTFILE' "'" result_filename "'"
```

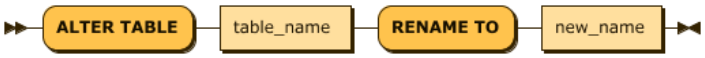
Checks whether the table has changed since it was appended to the specified table (Forgery). Forgery Check is only supported for Disk Column Table, and when creating Disk Column Table, it should be created by setting FORGERY_CHECK attribute to 1. When the Forgery is detected, a file corresponding to result_filename is created under $MACHBASE_HOME/trc and the file shows which part of the table has been changed.

```
-- Perform forgery check on t1 table.
```

```
alter table t1 check forgery resultfile 't1_forgery_check';
```

ALTER TABLE RENAME TO

alter_table_rename_stmt:




```
alter_table_rename_stmt ::= 'ALTER TABLE' table_name 'RENAME TO' new_name
```

Changes the name of the table.

Metatables can not be renamed, and you can not use the $ character in the name to be changed. Table renaming is only possible for log tables.

```
-- Change name of user table to employee.
ALTER TABLE user RENAME TO employee
```

TRUNCATE TABLE

truncate_table_stmt:


```
truncate_table_stmt ::= 'TRUNCATE TABLE' table_name
```

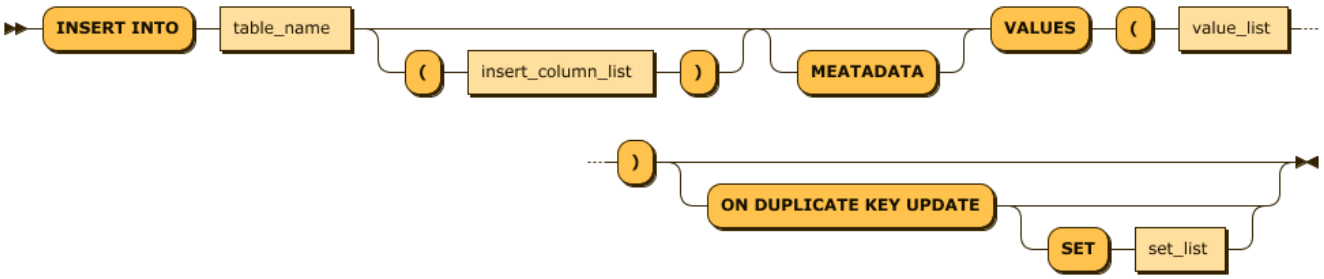
```
-- Delete all data in ctest table.
Mach> truncate table ctest;
Truncated successfully.
```

Deletes all data in the specified table. However, if there is another session in which the table is being searched, it fails with an error.

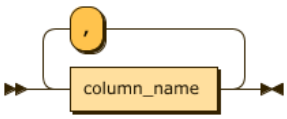
DML

INSERT

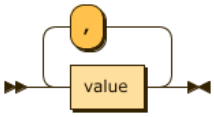
insert_stmt:



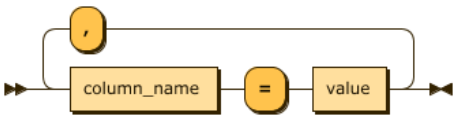
insert_column_list:



value_list:



set_list:




```
insert_stmt ::= 'INSERT INTO' table_name ( '(' insert_column_list ')' )? 'METADATA'? 'VALUES' '(' value_list ')' (
insert_column_list ::= column_name ( ',' column_name )*
value_list ::= value ( ',' value )*
set_list ::= column_name '=' value ( ',' column_name '=' value )*
```

```
create table test (number int,name varchar(20));
Created successfully.
insert into test values (1,"test");
1 row(s) inserted.
insert into test(name,number) values ("test",2);
1 row(s) inserted.
```

This is the syntax for entering values into a specific table. One unusual thing is that columns not specified in Column_List are all filled with NULL values. This is a considering the characteristics of log files adopted for convenience of input and efficiency of storage space.
METADATA is only available for tag tables.

INSERT ON DUPLICATE KEY UPDATE

Machbase supports syntax similar to the commonly known UPSERT function.

A special syntax that can be used when entering a value into a Lookup/Volatile table with a primary key specified. If the table already contains data with a duplicate primary key value, the value of the existing data is changed.
Of course, when there is no duplicated key value data, it is inserted as new data.

To use this syntax, the primary key must be specified in the volatile table.

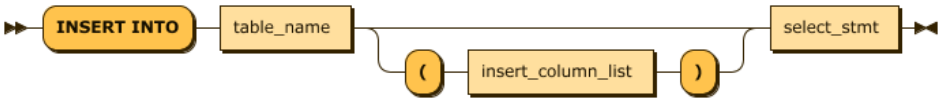
If the column value of the inserted data is different from the column value of the updated data, or if it is desired to update a column value other than the column value of the inserted data, the SET clause can be further input.

- The SET clause consists of 'column = value', each separated by a comma.

- You must not change the default key value in the SET clause.

INSERT SELECT

insert_select_stmt:




```
insert_select_stmt ::= 'INSERT INTO' table_name ( '(' insert_column_list ')' )? select_stmt
```

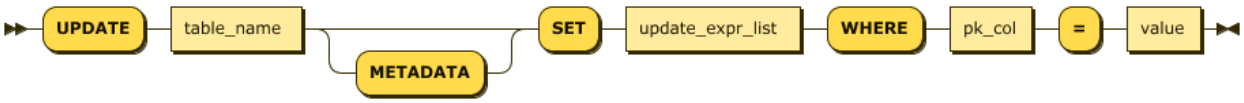
This statement inserts the result of the SELECT statement on a specific table. In basic, it is similar to other DBMSs, but with the following differences.

1. The _ARRIVAL_TIME column value is entered as the time value at the time the INSERT SELECT statement is executed unless specified in the select and INSERT column lists.
2. If the input value to be inserted for a VARCHAR type column is greater than the maximum length of the column, the maximum length of the corresponding column is entered without error.
3. If type conversion is possible (numeric -> numeric), it is inserted according to the input column value.
4. ROLLBACK does not occur if an error occurs during execution.
5. If you insert a value in the _ARRIVAL_TIME column, the new value will not be entered if it has a time before the existing value.

```
create table t1 (i1 integer, i2 varchar(60), i3 varchar(5));
Created successfully.

insert into t1 values (1, 'a', 'ddd' );
1 row(s) inserted.
insert into t1 values (2, 'kkkkkkkkkkkkkkkkkkkk', 'c');
1 row(s) inserted.

insert into t1 select * from t1;
2 row(s) inserted.
create table t2 (i1 integer, i2 varchar(60), i3 varchar(5));

insert into t2 (_arrival_time, i1, i2, i3) select _arrival_time, * from t1;
4 row(s) inserted.
```

UPDATE

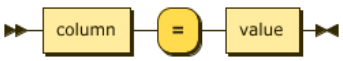
This function is available from 5.5.



update_stmt:

update_expr_list:

update_expr:




```
update_stmt ::= 'UPDATE' table_name ( 'METADATA' )? 'SET' update_expr_list 'WHERE' primary_key_column '=' value
update_expr_list ::= update_expr ( ',' update_expr)*
update_expr ::= column '=' value
```

INSERT ON DUPLICATE UPDATE syntax is also provided, rather than UPSERT via a KEY UPDATE.
You can also use the Primary Key to enter values into the specified Lookup/Volatile table. In the WHERE clause, you must create a matching predicate for the

primary key.

UPDATE METADATA

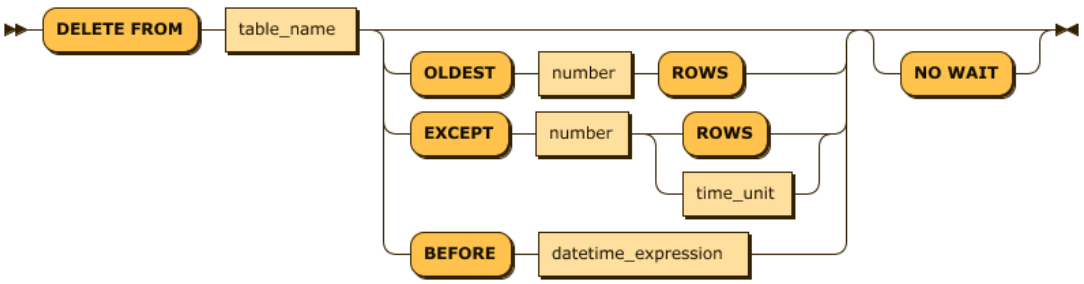
Only for the TAGDATA table, when you want to update the metadata.

```
UPDATE TAG METADATA SET ...
```

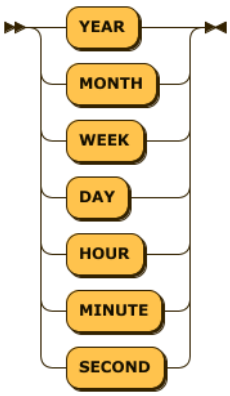
> ⚠ The metadata of the TAGDATA table can not be entered/modified through INSERT ON DUPLICATE KEY UPDATE.

DELETE

delete_stmt:



time_unit:




```
delete_stmt ::= 'DELETE FROM' table_name ( 'OLDEST' number 'ROWS' | 'EXCEPT' number ( 'ROWS' | time_unit ) | 'BEFOR
time_unit ::= 'DURATION' number time_unit ( ( 'BEFORE' | 'AFTER' ) number time_unit )?
```

The DELETE statement in Machbase can be performed on the log table. In addition, it is not possible to delete data in an arbitrary position in the middle, and it is possible to erase consecutively from the arbitrary position to the last (oldest log) record.
This is a policy that takes advantage of the characteristics of log data. It is a DB format representation of the act of deleting a file in order to secure space when it is entered once.

```
-- Delete all.
DELETE FROM devices;

-- Delete oldest last N rows.
DELETE FROM devices OLDEST N ROWS;

-- Delete all except recent N rows.
DELETE FROM devices EXCEPT N ROWS;

-- Delete all except N matches from now on.
DELETE FROM devices EXCEPT N DAY;

-- Delete all data from before June 1, 2014.
DELETE FROM devices BEFORE TO_DATE('2014-06-01', 'YYYY-MM-DD');
```

DELETE WHERE

delete_where_stmt:

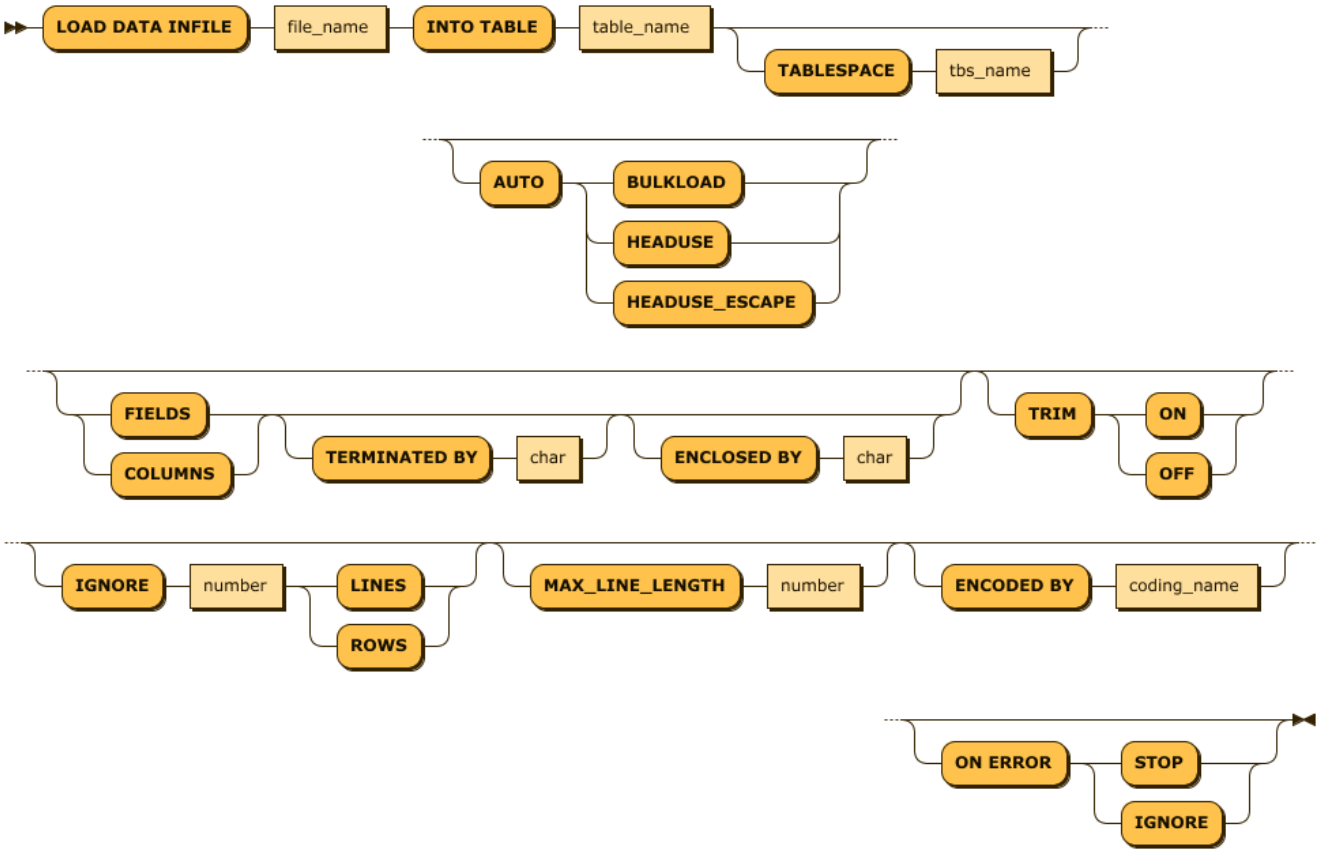

```
delete_where_stmt ::= 'DELETE FROM' table_name 'WHERE' column_name '=' value
```

```
create volatile table t1 (i1 int primary key, i2 int);
Created successfully.
insert into t1 values (2,2);
1 row(s) inserted.
delete from t1 where i1 = 2;
1 row(s) deleted.
```

- You can only delete records that match the conditions created in the WHERE clause, which can only be performed on volatile tables.
- The primary key can only be performed on the specified volatile table.
- The WHERE clause allows only conditions of (primary key column) = (value), and can not be created with other conditions.
- You can not use a column other than the primary key column in the condition.

LOAD DATA INFILE

load_data_infile_stmt:




```
load_data_infile_stmt: 'LOAD DATA INFILE' file_name 'INTO TABLE' table_name ( 'TABLESPACE' tbs_name )? ( 'AUTO' (
```

CSV format data files are read directly from the server, and tables and columns are created directly from the server according to the options and input them. Each option is explained as follows

Options	Description
AUTO mode_string	Creates the corresponding table and automatically generates the column type (varchar type for automatic creation) and the column name.

Options	Description
mode_string = (BULKLOAD | HEADUSE | HEADUSE_ESCAPE)	BULKLOAD: Enters one row of data as one column. It is used for data that can not be divided into columns. HEADUSE: Uses the column name as described in the first line of the data file as the column name of the table, and creates as many columns as there are in the line. HEADUSE_ESCAPE: Similar to the HEADUSE option, but appends a '_' character to the front and back of the column name to avoid errors that can occur if the column name is the same as the reserved word in the DB. If a special character exists in the column name, changes it to '_'.
(FIELDS|COLUMNS) TERMINATED BY 'term_char' ESCAPED BY 'escape_char'	Specifies the delimiter (term_char) and escape character (escape_char) for parsing the data line. For common CSV files, the delimiter is , and the escape character is '.
ENCODED BY coding_name coding_name = { UTF8(default) | MS949 | KSC5601 | EUCJP | SHIFTJIS | BIG5 | GB231280 }	Specifies the encoding options for the data file. The default value is UTF-8.
TRIM (ON | OFF)	Removes or maintains the empty space of the column. The default is ON.
IGNORE number (LINES | ROWS)	Ignores data for a specified number of lines or lines. It is used to ignore header of CSV format file or ignore VCF header.
MAX_LINE_LENGTH	Specifies the maximum length of one line. The default value is 512K. If the data is larger, you can specify a larger value.
ON ERROR (STOP | IGNORE)	Specifies the action to take when an error occurs during input. If it is STOP, input is stopped. If it is IGNORE, the line where error occurred is skipped and input is continued. The default is IGNORE.

```
-- Use default field delimiter(,)  field encloser (") to input data.
LOAD DATA INFILE '/tmp/aaa.csv' INTO TABLE Sample_data ;

-- Create NEWTABLE with one column and enter one line as one column.
LOAD DATA INFILE '/tmp/bbb.csv' INTO TABLE NEWTABLE AUTO BULKLOAD;

-- Create NEWTABLE using first line of csv as column information, and input it into table.
LOAD DATA INFILE '/tmp/bbb.csv' INTO TABLE NEWTABLE AUTO HEADUSE;

-- First line is ignored and field delimiter is ; and enclosing character is specified by '.
LOAD DATA INFILE '/tmp/ccc.csv' INTO TABLE Sample_data FIELDS TERMINATED BY ';' ENCLOSED '\''  IGNORE 1 LINES ON E
```

SELECT

SELECT is a syntax used to find, filter, and manipulate data from various tables in Machbase.

Index

SELECT Syntax

```
select_stmt UNION ALL select_stmt
```

```
SELECT target_list FROM TableList WHERE Condition GROUP BY Expr ORDER BY Expr [
```

SET OPERATOR

Used when receiving the results of multiple Select queries as a single query result. Machbase supports only the UNION ALL set operator. The set operator can be executed only if the left and right Select statements are (1) the same or compatible types, (2) the number of query results is the same, and if any of the two conditions does not match, they are treated as errors.

Data type conversion and compatibility verification are performed based on the following criteria.

- Signed integer types and unsigned integer types are not compatible.
- The integer type is compatible with the real type, and the query result is converted to the real type and returned.
- Character types are compatible with different lengths.
- IPv6 type and IPv4 type are not compatible.
- Of the two SELECT statements, the column name of the left query is always used.

Examples

```
SELECT i1, i2 FROM table_1
UNION ALL
SELECT c1, c2 FROM table_2
```

TARGET LIST

This is a **list of columns or subqueries** targeted by the Select statement .

The subquery used in the target list is treated as an error if it has two or more values or two or more result columns, such as a subquery used in the WHERE clause.

```
SELECT i1, i2 ...
SELECT i1 (Select avg(c1) FROM t1), i2 ...
```

CASE Statement

```
CASE <simple_case_expression|searched_case_expression> [else_clause] END

simple_case_expression ::=
    expr WHEN comparison_expr THEN return_expr
        [WHEN comparison_expr THEN return_expr ...]

searched_case_expression ::=
    WHEN condtion_expr THEN return EXPR [WHEN condtion_expr THEN return EXPR ...]

else_clause ::=
    ELSE else_value_expr
```

This is an expression that supports the IF ... THEN ... ELSE block of a typical programming language. simple_case_expression is executed in the form of return_expr when one column or expression is equal to the value of comparison_expr followed by when, and this when ... then clause can be repeated as

many times as desired.

searched_case_expression does not specify an expression after CASE but describes a conditional clause that includes a comparison operator in the when clause. If the result of each comparison operation is true, then the value of the then clause is returned. The else clause returns else_value if the value of the when clause is not satisfied (even if the expression is NULL).

```
select * from t1;
I1          I2
--------------------------
2           2
1           1
[2] row(s) selected.

select case i1 when 1 then 100 end from t1;
case i1 when 1 then 100 end
------------------------------
NULL
100
[2] row(s) selected.
```

In the simple_case_expression example, if the value of the i1 column is 2, NULL is returned.

```
select case when i1 > 0 then 100 when i1 > 1 then 200 end from t1;
case when i1 > 0 then 100 when i1 > 1 then 200 end
----------------------------------------
100
100
[2] row(s) selected.
```

Since searched_case_expression returns the first condition that satisfies the condition, 100 is returned, and the second condition is not executed.

FROM

You can specify a table name or an Inline view in the FROM clause. To perform a join between tables, lists the table or Inline view separated by a comma (,).

```
FROM table_name
```

Retrieves data in the table specified by table_name.

SUBQUERY(INLINE VIEW)

```
FROM (Select statement)
```

Retrievse data for the contents of the subquery enclosed in parentheses.

> Machbase server does not support correlated subqueries, so you can not reference columns in a subquery in an outer query.

JOIN(INNER JOIN)

```
FROM TABLE_1, TABLE_2
```

Joins two tables, table_1 and table_2. An INNER JOIN can be used when three or more tables are listed, and both the search condition and the conditional clause are described in the WHERE clause.

```
SELECT t1.i1, t2.i1 FROM t1, t2 WHERE t1.i1 = t2.i1 AND t1.i1 > 1 AND t2.i2 = 3;
```

INNER JOIN 및 OUTER JOIN

Supports ANSI style INNER JOIN, LEFT OUTER JOIN, and RIGHT OUTER JOIN. FULL OUTER JOIN is not supported.

```
FROM TABLE_1 [INNER|LEFT OUTER|RIGHT OUTER] JOIN TABLE_2 ON expression
```

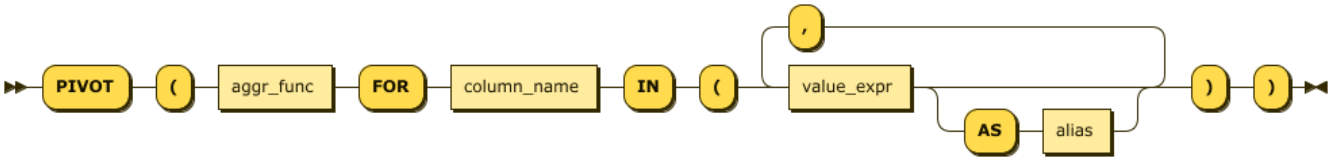
The ON clause of the ANSI-style JOIN clause uses the conditional clause that is performed by the JOIN. If the WHERE clause in the OUTER JOIN query has a clause for an inner table (a table that is filled with NULL if the condition of the ON clause is not satisfied), the query is converted to an INNER JOIN.

```
SELECT t1.i1 t2.i1 FROM t1 LEFT OUTER JOIN t2 ON (t1.i1 = t2.i1) WHERE t2.i2 = 1;
```

The above query is converted to an INNER JOIN by the condition t2.i2 = 1 in the WHERE clause.

PIVOT

> The PIVOT syntax is supported from Machbase version 5.6.



pivot_clause:

The PIVOT statement shows the aggregated results of GROUP BY output as ROW, rearranged into columns.

It is used in conjunction with the Inline view and is performed as follows.

- Performs GROUP BY on columns that are not used in the PIVOT clause of the inline view, and then performs aggregate functions on the values listed in the PIVOT IN clause.
- The resulting grouping column and the aggregation result are rotated and displayed as columns.

For example, aggregate the value of each device from the data collected from various sensors.
The query that should be performed through the CASE statement can be expressed simply through the PIVOT statement.

```
-- w/o PIVOT
SELECT * FROM (
    SELECT
             regtime,
             SUM(CASE WHEN tagid = 'FRONT_AXIS_TORQUE' THEN dvalue ELSE 0 END)  AS front_axis_torque,
             SUM(CASE WHEN tagid = 'REAR_AXIS_TORQUE' THEN dvalue ELSE 0 END)  AS rear_axis_torque,
             SUM(CASE WHEN tagid = 'HOIST_AXIS_TORQUE' THEN dvalue ELSE 0 END)  AS hoist_axis_torque,
             SUM(CASE WHEN tagid = 'SLIDE_AXIS_TORQUE' THEN dvalue ELSE 0 END)  AS slide_axis_torque
    FROM     result_d
    WHERE    regtime BETWEEN TO_DATE('2018-12-07 00:00:00') AND TO_DATE('2018-12-08 05:00:00')
    GROUP BY regtime
) WHERE front_axis_torque >= 40 AND rear_axis_torque >= 20;

-- w/ PIVOT
SELECT * FROM (
    SELECT regtime, tagid, dvalue FROM result_d
    WHERE  regtime BETWEEN TO_DATE('2018-12-07 00:00:00') AND TO_DATE('2018-12-08 05:00:00')
) PIVOT (SUM(dvalue) FOR tagid IN ('FRONT_AXIS_TORQUE', 'REAR_AXIS_TORQUE', 'HOIST_AXIS_TORQUE', 'SLIDE_AXIS_TORQUE
WHERE front_axis_torque >= 40 AND rear_axis_torque >= 20;

-- Result
regtime                         'FRONT_AXIS_TORQUE'         'REAR_AXIS_TORQUE'          'HOIST_AXIS_TORQUE'
-------------------------------------------------------------------------------------------------------------------
2018-12-07 16:42:29 840:000:000 12158                       7244                        NULL
2018-12-07 14:56:26 220:000:000 3308                        663                         NULL
2018-12-07 12:20:13 844:000:000 3804                        113                         NULL
2018-12-07 11:10:01 957:000:000 8729                        5384                        NULL
2018-12-07 17:46:57 812:000:000 7500                        4559                        NULL
2018-12-07 14:30:06 138:000:000 5080                        6817                        NULL
2018-12-07 13:09:20 464:000:000 5233                        1869                        -7253
2018-12-07 15:43:03 539:000:000 7491                        4453                        NULL
...
```

WHERE

Use of SUBQUERY

Subquery can be used for conditional statements. If the subquery returns more than one record in a clause except the IN clause, or if there is more than one result column in the subquery, it is not supported.

```
WHERE i1 = (SELECT MAX(c2) FROM T1)
```

Uses subquery by surrounding parentheses to the right of the conditional operator.

Machbase server does not support correlated subqueries, so you can not reference columns in a subquery in an outer query.

SEARCH Statement

The syntax is the same as for a regular database. However, a keyword index must be registered, and an additional search operation is possible by adding "SEARCH" as an operator keyword for text search.

```
-- drop table realdual;
create table realdual (id1 integer, id2 varchar(20), id3 varchar(20));

create keyword index idx1 on realdual (id2);
create keyword index idx2 on realdual (id3);

insert into realdual values(1, 'time time2', 'series series2');

select * from realdual;

select * from realdual where id2 search 'time';
select * from realdual where id3 search 'series' ;
select * from realdual where id2 search 'time' and id3 search 'series';
```

The results are as follows.

```
Mach> create table realdual (id1 integer, id2 varchar(20), id3 varchar(20));
Created successfully.

Mach> create keyword index idx1 on realdual (id2);
Created successfully.

Mach> create keyword index idx2 on realdual (id3);
Created successfully.

Mach> insert into realdual values(1, 'time time2', 'series series2');
1 row(s) inserted.

Mach> select * from realdual;
ID1         ID2                  ID3
------------------------------------------------------------
1           time time2           series series2
[1] row(s) selected.

Mach> select * from realdual where id2 search 'time';
ID1         ID2                  ID3
------------------------------------------------------------
1           time time2           series series2
[1] row(s) selected.

Mach> select * from realdual where id3 search 'series';
ID1         ID2                  ID3
------------------------------------------------------------
1           time time2           series series2
[1] row(s) selected.

Mach> select * from realdual where id2 search 'time' and id3 search 'series';
ID1         ID2                  ID3
------------------------------------------------------------
1           time time2           series series2
[1] row(s) selected.
```

ESEARCH Statement

The ESEARCH statement is a search keyword that enables extended searches on ASCII text. For this extension, search for the desired pattern is performed using the % character. In this Like operation, if all the records are checked before the %, the advantage of ESEARCH is that the words can be found quickly even in this case. This feature can be very useful when looking for a part of an English string (an error string or code).

```
Example

select id2 from realdual where id2 esearch 'bbb%';
id2
-----------------------------------------
bbb ccc1
aaa bbb1

[2] row(s) selected.

Search pattern 'bbb%' also includes bbb1 in search results.


select id3 from realdual where id3 esearch '%cd%';
id3
-----------------------------------------
cdf def1
bcd/cdf1ad
abc, bcd1
[3] row(s) selected.

% character works in middle of search pattern as well as beginning and end.

select id3 from realdual where id3 esearch '%cd%';
id3
-----------------------------------------
cdf def1
bcd/cdf1ad
abc, bcd1
[3] row(s) selected.
```

NOT SEARCH Statement

NOT SEARCH is a statement that returns true for records other than those found in the SEARCH statement.

NOT ESEARCH can not be used.

```
create table t1 (id integer, i2 varchar(10));
create keyword index t1_i2 on t1(i2);
insert into t1 values (1, 'aaaa');
insert into t1 values (2, 'bbbb');

select id from t1 where i2 not search 'aaaa';

id
-----------------------------------------
2
[1] row(s) selected.
```

REGEXP Statement

The REGEXP statement is used to perform searches on data using regular expressions. In general, patterns of a particular column are filtered using regular expressions.

One thing to keep in mind is that you can not use indexes when using the REGEXP clause, so you must lower the overall search cost by putting index conditions on other columns in order to reduce the overall search scope.
If you want to check a specific pattern, use index by SEARCH or ESEARCH, and then use REGEXP again in a state where the total number of data is small, it helps to improve the efficiency of the whole system.

```
Mach>
create table realdual (id1 integer, id2 varchar(20), id3 varchar(20));
create table dual (id integer);
insert into dual values(1);
insert into realdual values(1, 'time1', 'series1 series21');
insert into realdual values(1, 'time2', 'series2 series22');
insert into realdual values(1, 'time3', 'series3 series32');
```

```
Mach> select * from realdual where id2 REGEXP 'time' ;
ID1         ID2                  ID3
-------------------------------------------------------------
1           time3                series3 series32
1           time2                series2 series22
1           time1                series1 series21
[3] row(s) selected.

Mach> select * from realdual where id2 REGEXP 'time[12]' ;
ID1         ID2                  ID3
-------------------------------------------------------------
1           time2                series2 series22
1           time1                series1 series21
[2] row(s) selected.

Mach> select * from realdual where id2 REGEXP 'time[13]' ;
ID1         ID2                  ID3
-------------------------------------------------------------
1           time3                series3 series32
1           time1                series1 series21
[2] row(s) selected.

Mach> select * from realdual where id2 regexp 'time[13]' and id3 regexp 'series[12]';
ID1         ID2                  ID3
-------------------------------------------------------------
1           time1                series1 series21
[1] row(s) selected.

Mach> select * from realdual where id2 NOT REGEXP 'time[12]';
ID1         ID2                  ID3
-------------------------------------------------------------
1           time3                series3 series32
[1] row(s) selected.

Mach> SELECT 'abcde' REGEXP 'a[bcd]{1,10}e' from dual;
'abcde' REGEXP 'a[bcd]{1,10}e'
--------------------------------
1
[1] row(s) selected.
```

IN Statement

```
column_name IN (value1, value2,...)
```

The IN statement returns TRUE if it is satisfied in the value list. It is the same as the syntax linked by OR.

Use In Statement and SUBQUERY

You can use a subquery to the right of the IN statement in the conditional statement. However, if you specify more than one column on the left side of the IN condition, it treats it as an error and checks whether the result set returned from the right subquery exists in the left column value.

```
WHERE i1 IN (Select c1 from ...)
```

> Machbase server does not support correlated subqueries, so you can not reference columns in a subquery in an outer query.

BETWEEN Statement

```
column_name BETWEEN value1 AND value2
```

The BETWEEN statement returns TRUE if the value of column is in the range of value1 and value2.

RANGE Statement

```
column_name RANGE duration_spec;

-- duration_spec : integer (YEAR | WEEK | HOUR | MINUTE | SECOND);
```

Provides a Range operator that allows you to easily specify a time condition for a given column. The Range operator specifies the time range from the current time as the target of the operation, rather than specifying a specific time (as specified by the BEFORE keyword). With this operator, you can easily retrieve result records within a desired time range.

```
select * from test where id < 2 and c1 range 1 hour;
ID          C1
----------------------------------------------
1           2014-07-25 09:28:53 706:707:001
[1] row(s) selected.
```

GROUP BY / HAVING

The GROUP BY clause is used to group the results of a SELECT statement on a specific column. It is used when sorting by group or by aggregating functions by using aggregate functions. Group means records having the same column value for the column specified in the GROUP BY clause.You can combine the HAVING clause after the GROUP BY clause to set the conditional expression for group selection. That is, of all the groups constituted by the GROUP BY clause, only the group satisfying the conditional expression specified in the HAVING clause is inquired.

```
SELECT ...
GROUP BY { col_name | expr } ,...[ HAVING <search_condition> ]

select id1, avg(id2) from exptab where id2 group by id1 order by id1;
Obtain average value of id2 based on id1 column.
```

ORDER BY

The ORDER BY clause sorts the query results in ascending or descending order. If no sorting options such as ASC or DESC are specified, the ORDER BY clause sorts by default in ascending order. If the ORDER BY clause is not specified, the order of the records to be queried depends on the query.

```
SELECT ...
ORDER BY {col_name | expr} [ASC | DESC]

select id1, avg(id2) from exptab where id2 group by id1 order by id1;
Obtain average value of id2 based on id1 column.
```

SERIES BY

The SERIES BY clause extracts the sorted result set as successive result values satisfying the SERIES BY condition. If the ORDER BY clause is not specified, it generates the sorted result using the _ARRIVAL_TIME column value. Therefore, if you use the GROUP BY clause or the query for a volatile table or lookup table that does not have the _ARRIVAL_TIME column, you must use the ORDER BY clause do.

The result values that satisfy the conditional clause will have the return value of the same SERIESNUM () function.

```
For example, for the following data
CREATE TABLE T1 (C1 INTEGER, C2 INTEGER);
INSERT INTO T1 VALUES (0, 1);

INSERT INTO T1 VALUES (1, 2);

INSERT INTO T1 VALUES (2, 3);

INSERT INTO T1 VALUES (3, 2);

INSERT INTO T1 VALUES (4, 1);

INSERT INTO T1 VALUES (5, 2);

INSERT INTO T1 VALUES (6, 3);
```

```
INSERT INTO T1 VALUES (7, 1);


The following query produces the following output:
SELECT C1,C2 FROM T1 ORDER BY C1 SERIES BY C2>1;
C1          C2
---------------------------
1           2
2           3
3           2
5           2
6           3

If you want to know the RANGE value of C1 where the value of the C2 column is larger than 1, you can determine the
```

LIMIT

The LIMIT clause is used to limit the number of records to be output. You can specify an integer to output from the first row to the last row of the result set

```
LIMIT [offset,] row_count

select id1, avg(id2) from exptab where id2 group by id1 order by id1 LIMIT 10;
```

DURATION

DURATION is a keyword that allows you to easily determine the data retrieval scope based on _arrival_time. Used with the BEFORE statement to set a specific range of data at a specific point in time. By using this DURATION, search performance can be dramatically increased and the system load can be dramatically reduced. For more detailed usage, please refer to the following.

```
DURATION Number TimeSpec [BEFORE/AFTER Number TimeSpec]
TimeSpec : YEAR | MONTH | WEEK |  DAY | HOUR | MINUTE | SECOND
```

```
create table t8(i1 integer);
insert into t8 values(1);
insert into t8 values(2);

select i1 from t8;

# Without BEFORE clause
select i1 from t8 duration 2 second;
select i1 from t8 duration 1 minute;
select i1 from t8 duration 1 hour;
select i1 from t8 duration 1 day;
select i1 from t8 duration 1 week;
select i1 from t8 duration 1 month;
select i1 from t8 duration 1 year;

# Using full DURATION statement
select i1 from t8 duration 1 second before 1 day;
select i1 from t8 duration 1 minute before 1 day;
select i1 from t8 duration 1 hour before 1 day;
select i1 from t8 duration 1 day before 1 day;
select i1 from t8 duration 1 week before 1 day;
select i1 from t8 duration 1 month before 1 day;
select i1 from t8 duration 1 year before 1 day;
```

The results are as follows.

```
Mach> create table t8(i1 integer);
Created successfully.

Mach> insert into t8 values(1);
```

```
1 row(s) inserted.

Mach> insert into t8 values(2);
1 row(s) inserted.

Mach> select i1 from t8;
i1
--------------
2
1
[2] row(s) selected.

# Without BEFORE clause
Mach> select i1 from t8 duration 2 second;
i1
--------------
2
1
[2] row(s) selected.

Mach> select i1 from t8 duration 1 minute;
i1
--------------
2
1
[2] row(s) selected.

Mach> select i1 from t8 duration 1 hour;
i1
--------------
2
1
[2] row(s) selected.

Mach> select i1 from t8 duration 1 day;
i1
--------------
2
1
[2] row(s) selected.

Mach> select i1 from t8 duration 1 week;
i1
--------------
2
1
[2] row(s) selected.

Mach> select i1 from t8 duration 1 month;
i1
--------------
2
1
[2] row(s) selected.

Mach> select i1 from t8 duration 1 year;
i1
--------------
2
1
[2] row(s) selected.

# Using full DURATION statement
Mach> select i1 from t8 duration 1 second before 1 day;
i1
--------------
[0] row(s) selected.

Mach> select i1 from t8 duration 1 minute before 1 day;
i1
```

```
--------------
[0] row(s) selected.

Mach> select i1 from t8 duration 1 hour before 1 day;
i1
--------------
[0] row(s) selected.

Mach> select i1 from t8 duration 1 day before 1 day;
i1
--------------
[0] row(s) selected.

Mach> select i1 from t8 duration 1 week before 1 day;
i1
--------------
[0] row(s) selected.

Mach> select i1 from t8 duration 1 month before 1 day;
i1
--------------
[0] row(s) selected.

Mach> select i1 from t8 duration 1 year before 1 day;
i1
--------------
[0] row(s) selected.
```

SAVE DATA

Saves the results of the query directly into the CSV data file.

```
SAVE DATA INTO 'file_name.csv' [{FIELDS | COLUMNS} [TERMINATED BY 'char'] [ENCLOSED BY 'char'] ] [HEADER ON|OFF] [E
```

The options are described below.

Options	Description
(FIELDS|COLUMNS) TERMINATED BY 'term_char' ENCLOSED BY 'escape_char'	Specifies the column delimiter and escape delimiter of the csv file to be created.
HEADER (ON|OFF)	Decides whether to enter the column name on the first line of the csv file to be created. The default is OFF.
ENCODED BY coding_name coding_name = (UTF8, MS949, KSC5601, EUCJP, SHIFTJIS, BIG5, GB231280)	Specifies the encoding format of the output data file. The default value is UTF8.

```
SAVE DATA INTO '/tmp/aaa.csv' AS select * from t1;
-- Execute select statement and write result to '/tmp/aaa.csv' file in csv format.

SAVE DATA INTO '/tmp/ccc.csv'  FIELDS TERMINATED BY ';' ENCLOSED '\''  HEADER ON ENCODED BY 'MS949' AS select * fr
-- Execute select statement and write result to /tmp/ccc.csv file. Specify field separator and escape separator, re
```

SELECT Hint

Index

PARALLEL

Specifies parallel factor for parallel query execution.

```
SELECT /*+ PARALLEL(table_name, parallel_factor) */ ...
```

```
Mach> EXPLAIN SELECT /*+ PARALLEL(test, 8) */ sensor, frequency, avg(value)
FROM test
WHERE ts >= TO_DATE('2007-07-01', 'YYYY-MM-DD') and ts <= TO_DATE('2007-07-31',
GROUP BY sensor,frequency;

PLAN
------------------------------------------------------------------------------
 QPX_NODE_TYPE_PROJ
  QPX_NODE_TYPE_GRAG
   PARALLEL INDEX SCAN
    *BITMAP RANGE (t:92, c:2, i:94)
[4] row(s) selected.
```

NOPARALLEL

Does not perform in parallel.

```
SELECT /*+ NOPARALLEL(table_name) */ ...
```

```
Mach> EXPLAIN SELECT /*+ NOPARALLEL(test) */ sensor, frequency, avg(value)
FROM test
WHERE ts >= TO_DATE('2007-07-01', 'YYYY-MM-DD') and ts <= TO_DATE('2007-07-31', 'YYYY-MM-DD')
GROUP BY sensor,frequency;

PLAN
------------------------------------------------------------------------------
 QPX_NODE_TYPE_PROJ
  QPX_NODE_TYPE_GRAG
   INDEX SCAN
    *BITMAP RANGE (t:92, c:2, i:94)
[4] row(s) selected.
```

FULL

Does not use INDEX SCAN.

```
SELECT /*+ FULL(table_name) */ ...
```

```
Mach> EXPLAIN SELECT * FROM TEST WHERE I1 = 1;
PLAN
------------------------------------------------------------------------------
 PROJECT
  INDEX SCAN
   *BITMAP RANGE (t:7, c:1, i:8) with BLOOMFILTER
[3] row(s) selected.
Elapsed time: 0.001

Mach> EXPLAIN SELECT /*+ FULL(TEST) */ * FROM TEST WHERE I1 = 1;
PLAN
------------------------------------------------------------------------------
 PROJECT
```

```
  FULL SCAN
[2] row(s) selected.
```

NO_INDEX

Does not use the corresponding INDEX.

```
SELECT /*+ NO_INDEX(table_name,index_name) */ ...
```

```
Mach> EXPLAIN SELECT * FROM TEST WHERE I1 = 1;
PLAN
-------------------------------------------------------------------------------------
 PROJECT
  INDEX SCAN
   *BITMAP RANGE (t:7, c:1, i:8) with BLOOMFILTER
[3] row(s) selected.
Elapsed time: 0.001

Mach> EXPLAIN SELECT /*+ NO_INDEX(TEST,TEST_IDX) */ * FROM TEST WHERE I1 = 1;
PLAN
-------------------------------------------------------------------------------------
 PROJECT
  FULL SCAN
[2] row(s) selected.
```

RID_RANGE

Runs within RID range.

```
SELECT /*+ RID_RANGE(table_name,number,number) */ ...
```

```
Mach> SELECT /*+ RID_RANGE(TEST,45,50) */ _RID, * FROM TEST;
_RID                 I1
-----------------------------------
49                   1
48                   1
47                   1
46                   1
45                   1
[5] row(s) selected.
```

ROLLUP

It is possible to inquire statistical data in hours, minutes and seconds.

```
SELECT /*+ ROLLUP(table_name,(HOUR | MIN | SEC) [, aggr_func]) */ ...
```

```
Mach> EXPLAIN SELECT T_TIME, T_VALUE FROM TAG;
PLAN
-------------------------------------------------------------------------------------
 PROJECT
  TAG READ (RAW)
   KEYVALUE FULL SCAN
[3] row(s) selected.


Mach> EXPLAIN SELECT /*+ ROLLUP(TAG, SEC) */ T_TIME, T_VALUE FROM TAG;
PLAN
-------------------------------------------------------------------------------------
 PROJECT
  GROUP AGGREGATE
   TAG READ (SEC, AVG)
    KEYVALUE FULL SCAN
```

```
[4] row(s) selected.

Mach> EXPLAIN SELECT /*+ ROLLUP(TAG, SEC, MAX) */ T_TIME, T_VALUE FROM TAG;
PLAN
-------------------------------------------------------------------------------------
 PROJECT
  GROUP AGGREGATE
   TAG READ (SEC, MAX)
    KEYVALUE FULL SCAN
[4] row(s) selected.
```

SCAN_FORWARD, SCAN_BACKWARD

Specifies the direction of scanning for TAGDATA table. With SCAN_FORWARD, the oldest record input is retrieved first, whereas with SCAN_BACKWARD, the newest record input is retrieved first.

These hints have no effect on LOG tables.

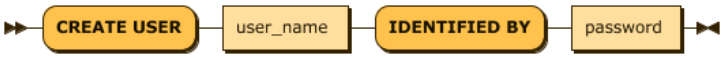
```
SELECT /*+ SCAN_FORWARD(table_name) */ ...
SELECT /*+ SCAN_BACKWARD(table_name) */ ...
```

```
Mach> SELECT /*+ SCAN_FORWARD(tag) */  * FROM tag WHERE t_name='TAG_99' LIMIT 10;
T_NAME                T_TIME                          T_VALUE
-------------------------------------------------------------------------------------
TAG_99                2017-01-01 00:00:49 500:000:000 0
TAG_99                2017-01-01 00:01:39 500:000:000 1
TAG_99                2017-01-01 00:02:29 500:000:000 2
TAG_99                2017-01-01 00:03:19 500:000:000 3
TAG_99                2017-01-01 00:04:09 500:000:000 4
TAG_99                2017-01-01 00:04:59 500:000:000 5
TAG_99                2017-01-01 00:05:49 500:000:000 6
TAG_99                2017-01-01 00:06:39 500:000:000 7
TAG_99                2017-01-01 00:07:29 500:000:000 8
TAG_99                2017-01-01 00:08:19 500:000:000 9
[10] row(s) selected.

Mach> SELECT /*+ SCAN_BACKWARD(tag) */ * FROM tag WHERE t_name='TAG_99' LIMIT 10;
T_NAME                T_TIME                          T_VALUE
-------------------------------------------------------------------------------------
TAG_99                2017-02-27 20:53:19 500:000:000 9
TAG_99                2017-02-27 20:52:29 500:000:000 8
TAG_99                2017-02-27 20:51:39 500:000:000 7
TAG_99                2017-02-27 20:50:49 500:000:000 6
TAG_99                2017-02-27 20:49:59 500:000:000 5
TAG_99                2017-02-27 20:49:09 500:000:000 4
TAG_99                2017-02-27 20:48:19 500:000:000 3
TAG_99                2017-02-27 20:47:29 500:000:000 2
TAG_99                2017-02-27 20:46:39 500:000:000 1
TAG_99                2017-02-27 20:45:49 500:000:000 0
[10] row(s) selected.

Mach>
```

User Management

CREATE USER

create_user_stmt:




```
create_user_stmt ::= 'CREATE USER' user_name 'IDENTIFIED BY' password
```

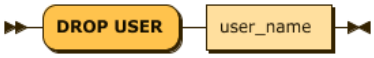
The syntax for creating a user is:

```
-- Example
CREATE USER new_user IDENTIFIED BY password
```

Index

DROP USER

drop_user_stmt:



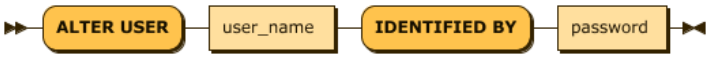

```
drop_user_stmt ::= 'DROP USER' user_name
```

The syntax for deleting a user is as follows. The SYS user can not be deleted, and if there is a table already created by the user to be deleted, an error is displayed.

```
-- Example
DROP USER old_user
```

ALTER USER

alter_user_pwd_stmt:




```
alter_user_pwd_stmt ::= 'ALTER USER' user_name 'IDENTIFIED BY' password
```

The user can change the password through the following syntax.

```
-- Example
ALTER USER user1 IDENTIFIED BY password
```

CONNECT

user_connect_stmt:


```
user_connect_stmt: 'CONNECT' user_name '/' password
```

The user can reconnect to another user via the following syntax without terminating the application.

```
CONNECT user1/password;
```

Managing User Example

Here is an example of the above query and its results.

```
##########################################
# Connect with SYS account
##########################################
Mach> create user demo identified by 'demo';
Created successfully.

Mach> drop user demo;
Dropped successfully.

Mach> create user demo1 identified by 'demo1';
Created successfully.

Mach> create user demo2 identified by 'demo2';
Created successfully.

Mach> alter user demo2 identified by 'demo22';
Altered successfully.

Mach> create table demo1_table (id integer);
Created successfully.

Mach> create bitmap index demo1_table_index1 on demo1_table(id);
Created successfully.

Mach> insert into demo1_table values(99991);
1 row(s) inserted.

Mach> insert into demo1_table values(99992);
1 row(s) inserted.

Mach> insert into demo1_table values(99993);
1 row(s) inserted.

Mach> select * from demo1_table;
ID
--------------
99993
99992
99991
[3] row(s) selected.

#Error: Can't drop the user connected.
Mach> drop user SYS;
[ERR-02083 : Drop user error. You cannot drop yourself(SYS).]

##########################################
# Connect DEMO1
##########################################
Mach> connect demo1/demo1;
Connected successfully.

#Error: can't alter other's account password
Mach> alter user demo2 identified by 'demo22';
[ERR-02085 : ALTER user error. The user(DEMO2) does not have ALTER privileges.]

Mach> alter user demo1 identified by demo11;
Altered successfully.
```

```
Mach> connect demo2/demo22;
Connected successfully.

#Error: wrong password
Mach> connect demo1/demo11234;
 [ERR-02081 : User authentication error. Invalid password (DEMO11234).]

# Correct password
Mach> connect demo1/demo11;
Connected successfully.

Mach> create table demo1_table (id integer);
Created successfully.

Mach> create bitmap index demo1_table_index1 on demo1_table(id);
Created successfully.

Mach> insert into demo1_table values(1);
1 row(s) inserted.

Mach> insert into demo1_table values(2);
1 row(s) inserted.

Mach> insert into demo1_table values(3);
1 row(s) inserted.

Mach> select * from demo1_table;
ID
--------------
3
2
1
[3] row(s) selected.

Mach> select * from demo1.demo1_table;
ID
--------------
3
2
1
[3] row(s) selected.

###############################################
# Connect SYS again
###############################################
Mach> connect SYS/MANAGER;
Connected successfully.

Mach> select * from demo1_table;
ID
--------------
99993
99992
99991
[3] row(s) selected.

Mach> select * from demo1.demo1_table;
ID
--------------
3
2
1
[3] row(s) selected.

Mach> drop user demo1;
[ERR-02084 : DROP user error. The user's tables still exist. Drop those tables first.]

Mach> connect demo1/demo11;
Connected successfully.
```

```
Mach> drop table demo1_table;
Dropped successfully.

Mach> connect SYS/MANAGER;
Connected successfully.

Mach> drop user demo1;
Dropped successfully.
```

Functions

ABS

This function works on a numeric column, converts it to a positive value, and returns the value as a real number.

```
ABS(column_name), ABS(column_name),...
```

```
Mach> CREATE TABLE abs_table (c1 integer, c2 double, c3 varchar(10));
Created successfully.

Mach> INSERT INTO abs_table VALUES(1, 1.0, '');
1 row(s) inserted.

Mach> INSERT INTO abs_table VALUES(2, 2.0, 'sqltest');
1 row(s) inserted.

Mach> INSERT INTO abs_table VALUES(3, 3.0, 'sqltest');
1 row(s) inserted.

Mach> SELECT ABS(c1), ABS(c2) from abs_table;
SELECT ABS(c1), ABS(c2) from abs_table;
ABS(c1)                     ABS(c2)
------------------------------------------------------------
3                           3
2                           2
1                           1
[3] row(s) selected.
```

ADD_TIME

This function performs a date and time operation on a given datetime column. Supports increment/decrement operations up to year, month, day, hour, minute, and second, and does not support operations on milli, micro, and nanoseconds. The Diff format is: "Year/Month/Day Hour:Minute:Second". Each item has a positive or negative value.

```
ADD_TIME(column,time_diff_format)
```

```
Mach> CREATE TABLE add_time_table (id INTEGER, dt DATETIME);
Created successfully.

Mach> INSERT INTO  add_time_table VALUES(1, TO_DATE('1999-11-11 1:2:3 4:5:6'));
1 row(s) inserted.

Mach> INSERT INTO  add_time_table VALUES(2, TO_DATE('2000-11-11 1:2:3 4:5:6'));
1 row(s) inserted.

Mach> INSERT INTO  add_time_table VALUES(3, TO_DATE('2012-11-11 1:2:3 4:5:6'));
1 row(s) inserted.

Mach> INSERT INTO  add_time_table VALUES(4, TO_DATE('2013-11-11 1:2:3 4:5:6'));
1 row(s) inserted.

Mach> INSERT INTO  add_time_table VALUES(5, TO_DATE('2014-12-30 11:22:33 444:55
1 row(s) inserted.

Mach> INSERT INTO  add_time_table VALUES(6, TO_DATE('2014-12-30 23:22:33 444:55
1 row(s) inserted.

Mach> SELECT ADD_TIME(dt, '1/0/0 0:0:0') FROM add_time_table;
```

```
ADD_TIME(dt, '1/0/0 0:0:0')
---------------------------------
2015-12-30 23:22:33 444:555:666
2015-12-30 11:22:33 444:555:666
2014-11-11 01:02:03 004:005:006
2013-11-11 01:02:03 004:005:006
2001-11-11 01:02:03 004:005:006
2000-11-11 01:02:03 004:005:006
[6] row(s) selected.

Mach> SELECT ADD_TIME(dt, '0/0/0 1:1:1') FROM add_time_table;
ADD_TIME(dt, '0/0/0 1:1:1')
---------------------------------
2014-12-31 00:23:34 444:555:666
2014-12-30 12:23:34 444:555:666
2013-11-11 02:03:04 004:005:006
2012-11-11 02:03:04 004:005:006
2000-11-11 02:03:04 004:005:006
1999-11-11 02:03:04 004:005:006
[6] row(s) selected.

Mach> SELECT ADD_TIME(dt, '1/1/1 0:0:0') FROM add_time_table;
ADD_TIME(dt, '1/1/1 0:0:0')
---------------------------------
2016-01-31 23:22:33 444:555:666
2016-01-31 11:22:33 444:555:666
2014-12-12 01:02:03 004:005:006
2013-12-12 01:02:03 004:005:006
2001-12-12 01:02:03 004:005:006
2000-12-12 01:02:03 004:005:006
[6] row(s) selected.

Mach> SELECT ADD_TIME(dt, '-1/0/0 0:0:0') FROM add_time_table;
ADD_TIME(dt, '-1/0/0 0:0:0')
---------------------------------
2013-12-30 23:22:33 444:555:666
2013-12-30 11:22:33 444:555:666
2012-11-11 01:02:03 004:005:006
2011-11-11 01:02:03 004:005:006
1999-11-11 01:02:03 004:005:006
1998-11-11 01:02:03 004:005:006
[6] row(s) selected.

Mach> SELECT ADD_TIME(dt, '0/0/0 -1:-1:-1') FROM add_time_table;
ADD_TIME(dt, '0/0/0 -1:-1:-1')
---------------------------------
2014-12-30 22:21:32 444:555:666
2014-12-30 10:21:32 444:555:666
2013-11-11 00:01:02 004:005:006
2012-11-11 00:01:02 004:005:006
2000-11-11 00:01:02 004:005:006
1999-11-11 00:01:02 004:005:006
[6] row(s) selected.

Mach> SELECT ADD_TIME(dt, '-1/-1/-1 0:0:0') FROM add_time_table;
ADD_TIME(dt, '-1/-1/-1 0:0:0')
---------------------------------
2013-11-29 23:22:33 444:555:666
2013-11-29 11:22:33 444:555:666
2012-10-10 01:02:03 004:005:006
2011-10-10 01:02:03 004:005:006
1999-10-10 01:02:03 004:005:006
1998-10-10 01:02:03 004:005:006
[6] row(s) selected.

Mach> SELECT * FROM add_time_table WHERE dt > ADD_TIME(TO_DATE('2014-12-30 11:2
ID          DT
-----------------------------------------------
6           2014-12-30 23:22:33 444:555:666
5           2014-12-30 11:22:33 444:555:666
```

```
[2] row(s) selected.

Mach> SELECT * FROM add_time_table WHERE dt > ADD_TIME(TO_DATE('2014-12-30 11:2
ID          DT
---------------------------------------------
6           2014-12-30 23:22:33 444:555:666
5           2014-12-30 11:22:33 444:555:666
4           2013-11-11 01:02:03 004:005:006
[3] row(s) selected.

Mach> SELECT ADD_TIME(TO_DATE('2000-12-01 00:00:00 000:000:001'), '-1/0/0 0:0:-
ADD_TIME(TO_DATE('2000-12-01 00:00:00 000:000:001'), '-1/0/0 0:0:-1')
-----------------------------------------
1999-11-30 23:59:59 000:000:001
1999-11-30 23:59:59 000:000:001
1999-11-30 23:59:59 000:000:001
1999-11-30 23:59:59 000:000:001
1999-11-30 23:59:59 000:000:001
1999-11-30 23:59:59 000:000:001
[6] row(s) selected.

Mach> SELECT * FROM add_time_table WHERE dt > ADD_TIME(TO_DATE('2014-12-30 11:2
ID          DT
---------------------------------------------
6           2014-12-30 23:22:33 444:555:666
5           2014-12-30 11:22:33 444:555:666
4           2013-11-11 01:02:03 004:005:006
[3] row(s) selected.
```

AVG

This function is an aggregate function that operates on a numeric column and prints the average value of that column.

```
AVG(column_name)
```

```
Mach> CREATE TABLE avg_table (id1 INTEGER, id2 INTEGER);
Created successfully.

Mach> INSERT INTO avg_table VALUES(1, 1);
1 row(s) inserted.

Mach> INSERT INTO avg_table VALUES(1, 2);
1 row(s) inserted.

Mach> INSERT INTO avg_table VALUES(1, 3);
1 row(s) inserted.

Mach> INSERT INTO avg_table VALUES(2, 1);
1 row(s) inserted.

Mach> INSERT INTO avg_table VALUES(2, 2);
1 row(s) inserted.

Mach> INSERT INTO avg_table VALUES(2, 3);
1 row(s) inserted.

Mach> INSERT INTO avg_table VALUES(null, 4);
1 row(s) inserted.

Mach> SELECT id1, AVG(id2) FROM avg_table GROUP BY id1;
id1         AVG(id2)
-----------------------------------------
2               2
NULL            4
1               2
```

BITAND / BITOR

This function converts two input values to a 64-bit signed integer and returns the result of bitwise and/or. The input value must be an integer and the output value is a 64-bit signed integer.

For integer values less than 0, it is recommended to use only uinteger and ushort types, because different results may be obtained depending on the platform.

```
BITAND (<expression1>, <expression2>)
BITOR (<expression1>, <expression2>)
```

```
Mach> CREATE TABLE bit_table (i1 INTEGER, i2 UINTEGER, i3 FLOAT, i4 DOUBLE, i5 SHORT, i6 VARCHAR(10));
Created successfully.

Mach> INSERT INTO bit_table VALUES (-1, 1, 1, 1, 2, 'aaa');
1 row(s) inserted.

Mach> INSERT INTO bit_table VALUES (-2, 2, 2, 2, 3, 'bbb');
1 row(s) inserted.

Mach> SELECT BITAND(i1, i2) FROM bit_table;
BITAND(i1, i2)
-----------------------
2
1
[2] row(s) selected.

Mach> SELECT * FROM bit_table WHERE BITAND(i2, 1) = 1;
I1          I2          I3                          I4                          I5          I6
-------------------------------------------------------------------------------------------------------------------
-1          1           1                           1                           2           aaa
[1] row(s) selected.

Mach> SELECT BITOR(i5, 1) FROM bit_table WHERE BITOR(i5, 1) = 3;
BITOR(i5, 1)
-----------------------
3
3
[2] row(s) selected.

Mach> SELECT * FROM bit_table WHERE BITOR(i2, 1) = 1;
I1          I2          I3                          I4                          I5          I6
-------------------------------------------------------------------------------------------------------------------
-1          1           1                           1                           2           aaa
[1] row(s) selected.

Mach> SELECT * FROM bit_table WHERE BITAND(i3, 1) = 1;
I1          I2          I3                          I4                          I5          I6
-------------------------------------------------------------------------------------------------------------------
[ERR-02037 : Function [BITAND] argument data type is mismatched.]
[0] row(s) selected.

Mach> SELECT * FROM bit_table WHERE BITAND(i4, 1) = 1;
I1          I2          I3                          I4                          I5          I6
-------------------------------------------------------------------------------------------------------------------
[ERR-02037 : Function [BITAND] argument data type is mismatched.]
[0] row(s) selected.

Mach> SELECT BITAND(i5, 1) FROM bit_table WHERE BITAND(i5, 1) = 1;
BITAND(i5, 1)
-----------------------
1
[1] row(s) selected.

Mach> SELECT * FROM bit_table WHERE BITOR(i6, 1) = 1;
I1          I2          I3                          I4                          I5          I6
-------------------------------------------------------------------------------------------------------------------
[ERR-02037 : Function [BITOR] argument data type is mismatched.]
[0] row(s) selected.
```

```
Mach> SELECT BITOR(i1, i2) FROM bit_table;
BITOR(i1, i2)
-----------------------
-2
-1
[2] row(s) selected.

Mach> SELECT BITAND(i1, i3) FROM bit_table;
BITAND(i1, i3)
-----------------------
[ERR-02037 : Function [BITAND] argument data type is mismatched.]
[0] row(s) selected.

Mach> SELECT BITOR(i1, i6) FROM bit_table;
BITOR(i1, i6)
-----------------------
[ERR-02037 : Function [BITOR] argument data type is mismatched.]
[0] row(s) selected.
```

COUNT

This function is an aggregate function that obtains the number of records in a given column.

```
COUNT(column_name)
```

```
Mach> CREATE TABLE count_table (id1 INTEGER, id2 INTEGER);
Created successfully.

Mach> INSERT INTO count_table VALUES(1, 1);
1 row(s) inserted.

Mach> INSERT INTO count_table VALUES(1, 2);
1 row(s) inserted.

Mach> INSERT INTO count_table VALUES(1, 3);
1 row(s) inserted.

Mach> INSERT INTO count_table VALUES(2, 1);
1 row(s) inserted.

Mach> INSERT INTO count_table VALUES(2, 2);
1 row(s) inserted.

Mach> INSERT INTO count_table VALUES(2, 3);
1 row(s) inserted.

Mach> INSERT INTO count_table VALUES(null, 4);
1 row(s) inserted.

Mach> SELECT COUNT(*) FROM count_table;
COUNT(*)
-----------------------
7
[1] row(s) selected.

Mach> SELECT COUNT(id1) FROM count_table;
COUNT(id1)
-----------------------
6
[1] row(s) selected.
```

DATE_TRUNC

This function returns a given datetime value as a new datetime value that is displayed only up to 'time unit' and 'time range'.

```
DATE_TRUNC (field, date_val [, count])
```

```
Mach> CREATE TABLE trunc_table (i1 INTEGER, i2 DATETIME);
Created successfully.

Mach> INSERT INTO trunc_table VALUES (1, TO_DATE('1999-11-11 1:2:0 4:5:1'));
1 row(s) inserted.

Mach> INSERT INTO trunc_table VALUES (2, TO_DATE('1999-11-11 1:2:0 5:5:2'));
1 row(s) inserted.

Mach> INSERT INTO trunc_table VALUES (3, TO_DATE('1999-11-11 1:2:1 6:5:3'));
1 row(s) inserted.

Mach> INSERT INTO trunc_table VALUES (4, TO_DATE('1999-11-11 1:2:1 7:5:4'));
1 row(s) inserted.

Mach> INSERT INTO trunc_table VALUES (5, TO_DATE('1999-11-11 1:2:2 8:5:5'));
1 row(s) inserted.

Mach> INSERT INTO trunc_table VALUES (6, TO_DATE('1999-11-11 1:2:2 9:5:6'));
1 row(s) inserted.

Mach> INSERT INTO trunc_table VALUES (7, TO_DATE('1999-11-11 1:2:3 10:5:7'));
1 row(s) inserted.

Mach> INSERT INTO trunc_table VALUES (8, TO_DATE('1999-11-11 1:2:3 11:5:8'));
1 row(s) inserted.

Mach> SELECT COUNT(*), DATE_TRUNC('second', i2) tm FROM trunc_table group by tm ORDER BY 2;
COUNT(*)             tm
------------------------------------------------------------
2                    1999-11-11 01:02:00 000:000:000
2                    1999-11-11 01:02:01 000:000:000
2                    1999-11-11 01:02:02 000:000:000
2                    1999-11-11 01:02:03 000:000:000
[4] row(s) selected.

Mach> SELECT COUNT(*), DATE_TRUNC('second', i2, 2) tm FROM trunc_table group by tm ORDER BY 2;
COUNT(*)             tm
------------------------------------------------------------
4                    1999-11-11 01:02:00 000:000:000
4                    1999-11-11 01:02:02 000:000:000
[2] row(s) selected.

Mach> SELECT COUNT(*), DATE_TRUNC('nanosecond', i2, 2) tm FROM trunc_table group by tm ORDER BY 2;
COUNT(*)             tm
------------------------------------------------------------
1                    1999-11-11 01:02:00 004:005:000
1                    1999-11-11 01:02:00 005:005:002
1                    1999-11-11 01:02:01 006:005:002
1                    1999-11-11 01:02:01 007:005:004
1                    1999-11-11 01:02:02 008:005:004
1                    1999-11-11 01:02:02 009:005:006
1                    1999-11-11 01:02:03 010:005:006
1                    1999-11-11 01:02:03 011:005:008
[8] row(s) selected.

Mach> SELECT COUNT(*), DATE_TRUNC('nsec', i2, 1000000000) tm FROM trunc_table group by tm ORDER BY 2; -- same with
COUNT(*)             tm
------------------------------------------------------------
2                    1999-11-11 01:02:00 000:000:000
2                    1999-11-11 01:02:01 000:000:000
2                    1999-11-11 01:02:02 000:000:000
2                    1999-11-11 01:02:03 000:000:000
[4] row(s) selected.
```

The allowable time ranges for time units and time units are as follows.

> nanosecond, microsecond, milisecond supports in and after 5.5.6 version.

Time Unit	Time Range
nanosecond (nsec)	1000000000 (1 sec)
microsecond (usec)	60000000 (60 sec)
milisecond (msec)	60000 (60 sec)
second	86400
minute	1440
hour	24
day	1
month	1
year	1

For example, if you type in `DATE_TRUNC('second', time, 120)`, the value returned will be displayed **every two minutes** and is the same as `DATE_TRUNC('minute', time, 2)`.

DAYOFWEEK

This function returns a natural number representing the day of the week for a given datetime value.

Returns a semantically equivalent value for TO_CHAR (time, 'DAY'), but returns an integer here.

```
DAYOFWEEK(date_val)
```

The returned natural number represents the next day of the week.

Return Value	Day of Week
0	Sunday
1	Monday
2	Tuesday
3	Wednesday
4	Thursday
5	Friday
6	Saturday

DECODE

This function compares the given Column value with Search, and returns the next return value if it is the same. If there is no satisfactory Search value, it returns the default value. If Default is omitted, NULL is returned.

```
DECODE(column, [search, return],.. default)
```

```
Mach> CREATE TABLE decode_table (id1 VARCHAR(11));
Created successfully.

Mach> INSERT INTO decode_table VALUES('decodetest1');
1 row(s) inserted.
```

```
Mach> INSERT INTO decode_table VALUES('decodetest2');
1 row(s) inserted.

Mach> SELECT id1, DECODE(id1, 'decodetest1', 'result1', 'decodetest2', 'result2', 'DEFAULT') FROM decode_table;
id1          DECODE(id1, 'decodetest1', 'result1', 'decodetest2', 'result2', 'DEFAULT')
-------------------------------------------------------------
decodetest2  result2
decodetest1  result1
[2] row(s) selected.

Mach> SELECT id1, DECODE(id1, 'codetest', 2, 99) FROM decode_table;
id1          DECODE(id1, 'codetest', 2, 99)
-----------------------------------------------
decodetest2  99
decodetest1  99
[2] row(s) selected.

Mach> SELECT DECODE(id1, 'decodetest1', 2) FROM decode_table;
DECODE(id1, 'decodetest1', 2)
--------------------------------
NULL
2
[2] row(s) selected.

Mach> SELECT DECODE(id1, 'codetest', 2) FROM decode_table;
DECODE(id1, 'codetest', 2)
----------------------------
NULL
NULL
[2] row(s) selected.
```

FIRST / LAST

This function is an aggregate function that returns the specific value of the highest (or last) record in the sequence in which the 'reference value' in each group is in order.

- FIRST: Returns a specific value from the most advanced record in the sequence
- LAST: Returns a specific value from the last record in the sequence

```
FIRST(sort_expr, return_expr)
LAST(sort_expr, return_expr)
```

```
Mach> create table firstlast_table (id integer, name varchar(20), group_no integer);
Created successfully.
Mach> insert into firstlast_table values (1, 'John', 0);
1 row(s) inserted.
Mach> insert into firstlast_table values (2, 'Grey', 1);
1 row(s) inserted.
Mach> insert into firstlast_table values (5, 'Ryan', 0);
1 row(s) inserted.
Mach> insert into firstlast_table values (4, 'Andrew', 0);
1 row(s) inserted.
Mach> insert into firstlast_table values (7, 'Kyle', 1);
1 row(s) inserted.
Mach> insert into firstlast_table values (6, 'Ross', 1);
1 row(s) inserted.

Mach> select group_no, first(id, name) from firstlast_table group by group_no;
group_no    first(id, name)
------------------------------------
1           Grey
0           John
[2] row(s) selected.


Mach> select group_no, last(id, name) from firstlast_table group by group_no;
group_no    last(id, name)
```

```
------------------------------------
1           Kyle
0           Ryan
```

FROM_UNIXTIME

This function converts a 32-bit UNIXTIME value entered as an integer to a datetime datatype value. (UNIX_TIMESTAMP converts datetime data to 32-bit UNIXTIME integer data.)

```
FROM_UNIXTIME(unix_timestamp_value)
```

```
Mach> SELECT FROM_UNIXTIME(315540671) FROM TEST;
FROM_UNIXTIME(315540671)
---------------------------------
1980-01-01 11:11:11 000:000:000

Mach> SELECT FROM_UNIXTIME(UNIX_TIMESTAMP('2001-01-01')) FROM unix_table;
FROM_UNIXTIME(UNIX_TIMESTAMP('2001-01-01'))
-----------------------------------------
2001-01-01 00:00:00 000:000:000
```

FROM_TIMESTAMP

This function takes a nanosecond value that has passed since 1970-01-01 09:00 and converts it to a datetime data type.

(TO_TIMESTAMP () converts a datetime data type to nanosecond data that has passed since 1970-01-01 09:00.)

```
FROM_TIMESTAMP(nanosecond_time_value)
```

```
Mach> SELECT FROM_TIMESTAMP(1562302560007248869) FROM TEST;
FROM_TIMESTAMP(1562302560007248869)
-----------------------------------
2019-07-05 13:56:00 007:248:869
```

Both sysdate and now represent nanosecond values elapsed since 1970-01-01 09:00 at the current time, so you can use FROM_TIMESTAMP () immediately.

Of course, the results are the same without using them. This can be useful if you have sysdate and now operations in nanoseconds.

```
Mach> select sysdate, from_timestamp(sysdate) from test_tbl;
sysdate                       from_timestamp(sysdate)
-------------------------------------------------------------------
2019-07-05 14:00:59 722:822:443 2019-07-05 14:00:59 722:822:443
[1] row(s) selected.

Mach> select sysdate, from_timestamp(sysdate-1000000) from test_tbl;
sysdate                       from_timestamp(sysdate-1000000)
-------------------------------------------------------------------
2019-07-05 14:01:05 130:939:525 2019-07-05 14:01:05 129:939:525      -- 1 ms (1,000,000 ns) difference happens
[1] row(s) selected.
```

GROUP_CONCAT

This function is an aggregate function that outputs the value of the corresponding column in the group in a string.

```
GROUP_CONCAT(
     [DISTINCT] column
     [ORDER BY { unsigned_integer | column }
     [ASC | DESC] [, column ...]]
     [SEPARATOR str_val]
)
```

- DISTINCT: Duplicate values are not appended if duplicate values are attached.
- ORDER BY: Arranges the sequence of column values to be attached according to the specified column values.
- SEPARATOR: A delimiter string used to append column values. The default value is a comma (,).

The syntax notes are as follows.

- You can specify only one column, and if you want to specify more than one column, you must use the TO_CHAR () function and the CONCAT operator (||) to make one expression.
- ORDER BY can specify other columns besides the columns to be joined, and can specify multiple columns.
- You must enter a string constant in SEPARATOR, and you can not enter a string column.

```
Mach> CREATE TABLE concat_table(id1 INTEGER, id2 DOUBLE, name VARCHAR(10));
Created successfully.

Mach> INSERT INTO concat_table VALUES (1, 2, 'John');
1 row(s) inserted.

Mach> INSERT INTO concat_table VALUES (2, 1, 'Ram');
1 row(s) inserted.

Mach> INSERT INTO concat_table VALUES (3, 2, 'Zara');
1 row(s) inserted.

Mach> INSERT INTO concat_table VALUES (4, 2, 'Jill');
1 row(s) inserted.

Mach> INSERT INTO concat_table VALUES (5, 1, 'Jack');
1 row(s) inserted.

Mach> INSERT INTO concat_table VALUES (6, 1, 'Jack');
1 row(s) inserted.


Mach> SELECT GROUP_CONCAT(name) AS G_NAMES FROM concat_table GROUP BY id2;
G_NAMES
--------------------------------------------------------------------------------
Jack,Jack,Ram
Jill,Zara,John
[2] row(s) selected.

Mach> SELECT GROUP_CONCAT(DISTINCT name) AS G_NAMES FROM concat_table GROUP BY Id2;
G_NAMES
--------------------------------------------------------------------------------
Jack,Ram
Jill,Zara,John
[2] row(s) selected.

Mach> SELECT GROUP_CONCAT(name SEPARATOR '.') G_NAMES FROM concat_table GROUP BY Id2;
G_NAMES
--------------------------------------------------------------------------------
Jack.Jack.Ram
Jill.Zara.John
[2] row(s) selected.

Mach> SELECT GROUP_CONCAT(name ORDER BY id1) G_NAMES, GROUP_CONCAT(id1 ORDER BY id1) G_SORTID FROM concat_table GR
G_NAMES
--------------------------------------------------------------------------------
G_SORTID
--------------------------------------------------------------------------------
Ram,Jack,Jack
2,5,6
John,Zara,Jill
1,3,4
[2] row(s) selected.
```

INSTR

This function returns the index of the number of characters in the string entered together. The index starts at 1.

- If no string pattern is found, 0 is returned.

- If the length of the string pattern to find is 0 or NULL, NULL is returned.

```
INSTR(target_string, pattern_string)
```

```
Mach> CREATE TABLE string_table(c1 VARCHAR(20));
Created successfully.

Mach> INSERT INTO string_table VALUES ('abstract');
1 row(s) inserted.

Mach> INSERT INTO string_table VALUES ('override');
1 row(s) inserted.

Mach> SELECT c1, INSTR(c1, 'act') FROM string_table;
c1                   INSTR(c1, 'act')
-----------------------------------------
override             0
abstract             6
[2] row(s) selected
```

LEAST / GREATEST

Both functions return the smallest value (LEAST) or the largest value (GREATEST) if you specify multiple columns or values as input parameters.

If the input value is 1 or absent, it is treated as an error. If the input value is NULL, NULL is returned. Therefore, if the input value is a column, it must be converted in advance using a function.
If a column (BLOB, TEXT) that can not be compared with the input value is included or type conversion is not possible for comparison, comparison is processed as an error.

```
LEAST(value_list, value_list,...)
GREATEST(value_list, value_list,...)
```

```
Mach> CREATE TABLE lgtest_table(c1 INTEGER, c2 LONG, c3 VARCHAR(10), c4 VARCHAR(5));
Created successfully.

Mach> INSERT INTO lgtest_table VALUES (1, 2, 'abstract', 'ace');
1 row(s) inserted.

Mach> INSERT INTO lgtest_table VALUES (null, 100, null, 'bag');
1 row(s) inserted.

Mach> SELECT LEAST (c1, c2) FROM lgtest_table;
LEAST (c1, c2)
-----------------------
NULL
1
[2] row(s) selected.

Mach> SELECT LEAST (c1, c2, -1) FROM lgtest_table;
LEAST (c1, c2, -1)
-----------------------
NULL
-1
[2] row(s) selected.

Mach> SELECT GREATEST(c3, c4) FROM lgtest_table;
GREATEST(c3, c4)
--------------------
NULL
ace
[2] row(s) selected.

Mach> SELECT LEAST(c3, c4) FROM lgtest_table;
LEAST(c3, c4)
-----------------
NULL
```

```
abstract
[2] row(s) selected.

Mach> SELECT LEAST(NVL(c3, 'aa'), c4) FROM lgtest_table;
LEAST(NVL(c3, 'aa'), c4)
----------------------------
aa
abstract
[2] row(s) selected.
```

LENGTH

This function gets the length of a string column. The obtained value outputs the number of bytes in English.

```
LENGTH(column_name)
```

```
Mach> CREATE TABLE length_table (id1 INTEGER, id2 DOUBLE, name VARCHAR(15));
Created successfully.

Mach> INSERT INTO length_table VALUES(1, 10, 'Around the Horn');
1 row(s) inserted.

Mach> INSERT INTO length_table VALUES(NULL, 20, 'Alfreds Futterkiste');
1 row(s) inserted.

Mach> INSERT INTO length_table VALUES(3, NULL, 'Antonio Moreno');
1 row(s) inserted.

Mach> INSERT INTO length_table VALUES(4, 40, NULL);
1 row(s) inserted.

Mach> select * FROM length_table;
ID1         ID2                         NAME
---------------------------------------------------------------
4           40                          NULL
3           NULL                        Antonio Moreno
NULL        20                          Alfreds Futterk
1           10                          Around the Horn
[4] row(s) selected.

Mach> select id1 * 10 FROM length_table;
id1 * 10
-----------------------
40
30
NULL
10
[4] row(s) selected.

Mach> select * FROM length_table Where id1 > 1 and id2 < 50;
ID1         ID2                         NAME
---------------------------------------------------------------
4           40                          NULL
[1] row(s) selected.

Mach> select name || ' with null concat' FROM length_table;
name || ' with null concat'
-----------------------------------
NULL
Antonio Moreno with null concat
Alfreds Futterk with null concat
Around the Horn with null concat
[4] row(s) selected.

Mach> select LENGTH(name) FROM length_table;
LENGTH(name)
```

```
---------------
NULL
14
15
15
[4] row(s) selected.
```

LOWER

This function converts an English string to lowercase.

```
LOWER(column_name)
```

```
Mach> CREATE TABLE lower_table (name VARCHAR(20));
Created successfully.

Mach> INSERT INTO lower_table VALUES('');
1 row(s) inserted.

Mach> INSERT INTO lower_table VALUES('James Backley');
1 row(s) inserted.

Mach> INSERT INTO lower_table VALUES('Alfreds Futterkiste');
1 row(s) inserted.

Mach> INSERT INTO lower_table VALUES('Antonio MORENO');
1 row(s) inserted.

Mach> INSERT INTO lower_table VALUES (NULL);
1 row(s) inserted.

Mach> SELECT LOWER(name) FROM lower_table;
LOWER(name)
------------------------
NULL
antonio moreno
alfreds futterkiste
james backley
NULL
[5] row(s) selected.
```

LPAD / RPAD

This function adds a character to the left (LPAD) or to the right (RPAD) until the input is of a given length.

The last parameter, char, can be omitted, or a space ' ' character if omitted.
If the input column value is longer than the given length, the characters are not appended but only the length is taken from the beginning.

```
LPAD(str, len, padstr)
RPAD(str, len, padstr)
```

```
Mach> CREATE TABLE pad_table (c1 integer, c2 varchar(15));
Created successfully.

Mach> INSERT INTO pad_table VALUES (1, 'Antonio');
1 row(s) inserted.

Mach> INSERT INTO pad_table VALUES (25, 'Johnathan');
1 row(s) inserted.

Mach> INSERT INTO pad_table VALUES (30, 'M');
1 row(s) inserted.
```

```
Mach> SELECT LPAD(to_char(c1), 5, '0') FROM pad_table;
LPAD(to_char(c1), 5, '0')
-----------------------------
00030
00025
00001
[3] row(s) selected.

Mach> SELECT RPAD(to_char(c1), 5, '0') FROM pad_table;
RPAD(to_char(c1), 5, '0')
-----------------------------
30000
25000
10000
[3] row(s) selected.

Mach> SELECT LPAD(c2, 5) FROM pad_table;
LPAD(c2, 5)
---------------
    M
Johna
Anton
[3] row(s) selected.

Mach> SELECT RPAD(c2, 5) FROM pad_table;
RPAD(c2, 5)
---------------
M
Johna
Anton
[3] row(s) selected.

Mach> SELECT RPAD(c2, 10, '***') FROM pad_table;
RPAD(c2, 10, '***')
-----------------------
M*********
Johnathan*
Antonio***
[3] row(s) selected.
```

LTRIM / RTRIM

This function removes the value corresponding to the pattern string from the first parameter. The LTRIM function checks to see if the characters are in pattern from left to right, the RTRIM function from right to left, and truncates until a character not in pattern is encountered. If all the strings are present in the pattern, NULL is returned.

If you do not specify a pattern expression, use the space character '' as a basis to remove the space character.

```
LTRIM(column_name, pattern)
RTRIM(column_name, pattern)
```

```
Mach> CREATE TABLE trim_table1(name VARCHAR(10));
Created successfully.

Mach> INSERT INTO trim_table1 VALUES ('   smith   ');
1 row(s) inserted.

Mach> SELECT ltrim(name) FROM trim_table1;
ltrim(name)
---------------
smith
[1] row(s) selected.

Mach> SELECT rtrim(name) FROM trim_table1;
rtrim(name)
---------------
   smith
```

```
[1] row(s) selected.

Mach> SELECT ltrim(name, ' s') FROM trim_table1;
ltrim(name, ' s')
---------------------
mith
[1] row(s) selected.

Mach> SELECT rtrim(name, 'h ') FROM trim_table1;
rtrim(name, 'h ')
---------------------
   smit
[1] row(s) selected.

Mach> CREATE TABLE trim_table2 (name VARCHAR(10));
Created successfully.

Mach> INSERT INTO trim_table2 VALUES ('ddckaaadkk');
1 row(s) inserted.

Mach> SELECT ltrim(name, 'dc') FROM trim_table2;
ltrim(name, 'dc')
---------------------
kaaadkk
[1] row(s) selected.

Mach> SELECT rtrim(name, 'dk') FROM trim_table2;
rtrim(name, 'dk')
---------------------
ddckaaa
[1] row(s) selected.

Mach> SELECT ltrim(name, 'dckak') FROM trim_table2;
ltrim(name, 'dckak')
------------------------
NULL
[1] row(s) selected.

Mach> SELECT rtrim(name, 'dckak') FROM trim_table2;
rtrim(name, 'dckak')
------------------------
NULL
[1] row(s) selected.
```

MAX

This function is an aggregate function that obtains the maximum value of a given numeric column.

```
MAX(column_name)
```

```
Mach> CREATE TABLE max_table (c INTEGER);
Created successfully.

Mach> INSERT INTO max_table VALUES(10);
1 row(s) inserted.

Mach> INSERT INTO max_table VALUES(20);
1 row(s) inserted.

Mach> INSERT INTO max_table VALUES(30);
1 row(s) inserted.

Mach> SELECT MAX(c) FROM max_table;
MAX(c)
--------------
30
```

```
[1] row(s) selected.
```

MIN

This function is an aggregate function that obtains the minimum value of a corresponding numeric column.

```
MIN(column_name)
```

```
Mach> CREATE TABLE min_table(c1 INTEGER);
Created successfully.

Mach> INSERT INTO min_table VALUES(1);
1 row(s) inserted.

Mach> INSERT INTO min_table VALUES(22);
1 row(s) inserted.

Mach> INSERT INTO min_table VALUES(33);
1 row(s) inserted.

Mach> SELECT MIN(c1) FROM min_table;
MIN(c1)
--------------
1
[1] row(s) selected.
```

NVL

This function returns value if the value of the column is NULL, or the value of the original column if it is not NULL.

```
NVL(string1, replace_with)
```

```
Mach> CREATE TABLE nvl_table (c1 varchar(10));
Created successfully.

Mach> INSERT INTO nvl_table VALUES ('Johnathan');
1 row(s) inserted.

Mach> INSERT INTO nvl_table VALUES (NULL);
1 row(s) inserted.

Mach> SELECT NVL(c1, 'Thomas') FROM nvl_table;
NVL(c1, 'Thomas')
---------------------
Thomas
Johnathan
```

ROUND

This function returns the result of rounding off the digits of the input value (input digit +1). If no digits are entered, the rounding is done at position 0. It is possible to enter a negative number in `decimals` place to round the decimal place.

```
ROUND(column_name, [decimals])
```

```
Mach> CREATE TABLE round_table (c1 DOUBLE);
Created successfully.

Mach> INSERT INTO round_table VALUES (1.994);
```

```
1 row(s) inserted.

Mach> INSERT INTO round_table VALUES (1.995);
1 row(s) inserted.

Mach> SELECT c1, ROUND(c1, 2) FROM round_table;
c1                          ROUND(c1, 2)
------------------------------------------------------------
1.995                       2
1.994                       1.99
```

ROWNUM

This function assigns a number to the SELECT query result row.

It can be used inside Subquery or Inline View that is used inside SELECT query. If you use ROWNUM () function in Inline View in Target List, you need to give Alias to refer to from outside.

```
ROWNUM()
```

Available Clauses

This function can be used in the target list, GROUP BY, or ORDER BY clause of a SELECT query. However, it can not be used in the WHERE and HAVING clauses of a SELECT query. ROWNUM () If you want to control WHERE or HAVING clause with result number, you can use SELECT query with ROWNUM () in Inline View and refer to it in WHERE or HAVING clause.

Available Clauses	Unavailable Clauses
Target List / GROUP BY / ORDER BY	WHERE / HAVING

```
Mach> CREATE TABLE rownum_table(c1 INTEGER, c2 DOUBLE, c3 VARCHAR(10));
Created successfully.

Mach> INSERT INTO rownum_table VALUES(1, 1.0, '');
1 row(s) inserted.

Mach> INSERT INTO rownum_table VALUES(2, 2.0, 'Second Row');
1 row(s) inserted.

Mach> INSERT INTO rownum_table VALUES(3, 3.3, 'Third Row');
1 row(s) inserted.

Mach> INSERT INTO rownum_table VALUES(4, 4.3, 'Fourth Row');
1 row(s) inserted.

Mach> SELECT INNER_RANK, c3 AS NAME
    2 FROM   (SELECT ROWNUM() AS INNER_RANK, * FROM rownum_table)
    3 WHERE  INNER_RANK < 3;
INNER_RANK           NAME
-----------------------------------
1                    Fourth Row
2                    Third Row
[2] row(s) selected.
```

Altering Results Due to Sorting

If there is an ORDER BY clause in the SELECT query, the result number of ROWNUM () in the target list may not be sequentially assigned. This is because the ROWNUM () operation is performed before the operation of the ORDER BY clause. If you want to give it sequentially, you can use the query containing the ORDER BY clause in Inline View and then call ROWNUM () in the outer SELECT statement.

```
Mach> CREATE TABLE rownum_table(c1 INTEGER, c2 DOUBLE, c3 VARCHAR(10));
Created successfully.

Mach> INSERT INTO rownum_table VALUES(1, 1.0, '');
1 row(s) inserted.
```

```
Mach> INSERT INTO rownum_table VALUES(2, 2.0, 'John');
1 row(s) inserted.

Mach> INSERT INTO rownum_table VALUES(3, 3.3, 'Sarah');
1 row(s) inserted.

Mach> INSERT INTO rownum_table VALUES(4, 4.3, 'Micheal');
1 row(s) inserted.

Mach> SELECT ROWNUM(), c2 AS SORT, c3 AS NAME
    2 FROM   ( SELECT * FROM rownum_table ORDER BY c3 );
ROWNUM()             SORT                        NAME
------------------------------------------------------------------
1                    1                           NULL
2                    2                           John
3                    4.3                         Micheal
4                    3.3                         Sarah
[4] row(s) selected.
```

SERIESNUM

Returns a number indicating how many of the records belong to the series grouped by SERIES BY. The return type is BIGINT type, and always returns 1 if the SERIES BY clause is not used.

```
SERIESNUM()
```

```
Mach> CREATE TABLE T1 (C1 INTEGER, C2 INTEGER);
Created successfully.

Mach> INSERT INTO T1 VALUES (0, 1);
1 row(s) inserted.

Mach> INSERT INTO T1 VALUES (1, 2);
1 row(s) inserted.

Mach> INSERT INTO T1 VALUES (2, 3);
1 row(s) inserted.

Mach> INSERT INTO T1 VALUES (3, 2);
1 row(s) inserted.

Mach> INSERT INTO T1 VALUES (4, 1);
1 row(s) inserted.

Mach> INSERT INTO T1 VALUES (5, 2);
1 row(s) inserted.

Mach> INSERT INTO T1 VALUES (6, 3);
1 row(s) inserted.

Mach> INSERT INTO T1 VALUES (7, 1);
1 row(s) inserted.


Mach> SELECT SERIESNUM(), C1, C2 FROM T1 ORDER BY C1 SERIES BY C2 > 1;
SERIESNUM() C1 C2
-------------------------------------------------
1 1 2
1 2 3
1 3 2
2 5 2
2 6 3
[5] row(s) selected.
```

STDDEV / STDDEV_POP

This function is an aggregate function that returns the (standard) deviation and the population standard deviation of the (input) column. Equivalent to the square root of the VARIANCE and VAR_POP values, respectively.

```
STDDEV(column)
STDDEV_POP(column)
```

```
Mach> CREATE TABLE stddev_table(c1 INTEGER, C2 DOUBLE);

Mach> INSERT INTO stddev_table VALUES (1, 1);
1 row(s) inserted.

Mach> INSERT INTO stddev_table VALUES (2, 1);
1 row(s) inserted.

Mach> INSERT INTO stddev_table VALUES (3, 2);
1 row(s) inserted.

Mach> INSERT INTO stddev_table VALUES (4, 2);
1 row(s) inserted.

Mach> SELECT c2, STDDEV(c1) FROM stddev_table GROUP BY c2;
c2                          STDDEV(c1)
------------------------------------------------------------
1                           0.707107
2                           0.707107
[2] row(s) selected.

Mach> SELECT c2, STDDEV_POP(c1) FROM stddev_table GROUP BY c2;
c2                          STDDEV_POP(c1)
------------------------------------------------------------
1                           0.5
2                           0.5
[2] row(s) selected.
```

SUBSTR

This function truncates the variable string column data from START to SIZE.

- START starts at 1 and returns NULL if it is zero.
- If SIZE is larger than the size of the corresponding string, only the maximum value of the string is returned.

SIZE is optional, and if omitted, it is internally specified by the size of the string.

```
SUBSTRING(column_name, start, [length])
```

```
Mach> CREATE TABLE substr_table (c1 VARCHAR(10));
Created successfully.

Mach> INSERT INTO substr_table values('ABCDEFG');
1 row(s) inserted.

Mach> INSERT INTO substr_table values('abstract');
1 row(s) inserted.

Mach> SELECT SUBSTR(c1, 1, 1) FROM substr_table;
SUBSTR(c1, 1, 1)
--------------------
a
A
[2] row(s) selected.

Mach> SELECT SUBSTR(c1, 3, 3) FROM substr_table;
SUBSTR(c1, 3, 3)
--------------------
```

```
str
CDE
[2] row(s) selected.

Mach> SELECT SUBSTR(c1, 2) FROM substr_table;
SUBSTR(c1, 2)
-----------------
bstract
BCDEFG
[2] row(s) selected.

Mach> drop table substr_table;
Dropped successfully.

Mach> CREATE TABLE substr_table (c1 VARCHAR(10));
Created successfully.

Mach> INSERT INTO substr_table values('ABCDEFG');
1 row(s) inserted.

Mach> SELECT SUBSTR(c1, 1, 1) FROM substr_table;
SUBSTR(c1, 1, 1)
--------------------
A
[1] row(s) selected.

Mach> SELECT SUBSTR(c1, 3, 3) FROM substr_table;
SUBSTR(c1, 3, 3)
--------------------
CDE
[1] row(s) selected.

Mach> SELECT SUBSTR(c1, 2) FROM substr_table;
SUBSTR(c1, 2)
-----------------
BCDEFG
[1] row(s) selected.
```

SUBSTRING_INDEX

Returns the duplicate string until the given delim is found by the count entered. If count is a negative value, it checks the delimiter from the end of the input string and returns it from the position where the delimiter was found to the end of the string.

If you enter count as 0 or there is no delimiter in the string, the function will return NULL.

```
SUBSTRING_INDEX(expression, delim, count)
```

```
Mach> CREATE TABLE substring_table (url VARCHAR(30));
Created successfully.

Mach> INSERT INTO substring_table VALUES('www.machbase.com');
1 row(s) inserted.

Mach> SELECT SUBSTRING_INDEX(url, '.', 1) FROM substring_table;
SUBSTRING_INDEX(url, '.', 1)
---------------------------------
www
[1] row(s) selected.

Mach> SELECT SUBSTRING_INDEX(url, '.', 2) FROM substring_table;
SUBSTRING_INDEX(url, '.', 2)
---------------------------------
www.machbase
[1] row(s) selected.

Mach> SELECT SUBSTRING_INDEX(url, '.', -1) FROM substring_table;
SUBSTRING_INDEX(url, '.', -1)
```

```
---------------------------------
com
[1] row(s) selected.

Mach> SELECT SUBSTRING_INDEX(SUBSTRING_INDEX(url, '.', 2), '.', -1) FROM substring_table;
SUBSTRING_INDEX(SUBSTRING_INDEX(url, '.', 2), '.', -1)
------------------------------------------
machbase
[1] row(s) selected.

Mach> SELECT SUBSTRING_INDEX(url, '.', 0) FROM substring_table;
SUBSTRING_INDEX(url, '.', 0)
---------------------------------
NULL
[1] row(s) selected.
```

SUM

This function is an aggregate function that represents the sum of the numeric columns.

```
SUM(column_name)
```

```
Mach> CREATE TABLE sum_table (c1 INTEGER, c2 INTEGER);
Created successfully.

Mach> INSERT INTO sum_table VALUES(1, 1);
1 row(s) inserted.

Mach> INSERT INTO sum_table VALUES(1, 2);
1 row(s) inserted.

Mach> INSERT INTO sum_table VALUES(1, 3);
1 row(s) inserted.

Mach> INSERT INTO sum_table VALUES(2, 1);
1 row(s) inserted.

Mach> INSERT INTO sum_table VALUES(2, 2);
1 row(s) inserted.

Mach> INSERT INTO sum_table VALUES(2, 3);
1 row(s) inserted.

Mach> INSERT INTO sum_table VALUES(3, 4);
1 row(s) inserted.

Mach> SELECT c1, SUM(c1) from sum_table group by c1;
c1          SUM(c1)
-----------------------------------
2           6
3           3
1           3
[3] row(s) selected.

Mach> SELECT c1, SUM(c2) from sum_table group by c1;
c1          SUM(c2)
-----------------------------------
2           6
3           4
1           6
[3] row(s) selected.
```

SYSDATE / NOW

SYSDATE is a pseudocolumn, not a function, that returns the system's current time.

NOW is the same function as SYSDATE and is provided for user convenience.

```
SYSDATE
NOW
```

```
Mach> SELECT SYSDATE, NOW FROM t1;

SYSDATE                         NOW
------------------------------------------------------------------
2017-01-16 14:14:53 310:973:000 2017-01-16 14:14:53 310:973:000
```

TO_CHAR

This function converts a given data type to a string type. Depending on the type, format_string may be specified, but not for binary types.

```
TO_CHAR(column)
```

TO_CHAR: Default Datatype

The default data types are converted to data in the form of strings as shown below.

```
Mach> CREATE TABLE fixed_table (id1 SHORT, id2 INTEGER, id3 LONG, id4 FLOAT, id5 DOUBLE, id6 IPV4, id7 IPV6, id8 VA
Created successfully.

Mach> INSERT INTO fixed_table values(200, 19234, 1234123412, 3.14, 7.8338, '192.168.0.1', '::127.0.0.1', 'log varch
1 row(s) inserted.

Mach> SELECT '[ ' || TO_CHAR(id1) || ' ]' FROM fixed_table;
'[ ' || TO_CHAR(id1) || ' ]'
--------------------------------------------------------------------------------
[ 200 ]
[1] row(s) selected.

Mach> SELECT '[ ' || TO_CHAR(id2) || ' ]' FROM fixed_table;
'[ ' || TO_CHAR(id2) || ' ]'
--------------------------------------------------------------------------------
[ 19234 ]
[1] row(s) selected.

Mach> SELECT '[ ' || TO_CHAR(id3) || ' ]' FROM fixed_table;
'[ ' || TO_CHAR(id3) || ' ]'
--------------------------------------------------------------------------------
[ 1234123412 ]
[1] row(s) selected.

Mach> SELECT '[ ' || TO_CHAR(id4) || ' ]' FROM fixed_table;
'[ ' || TO_CHAR(id4) || ' ]'
--------------------------------------------------------------------------------
[ 3.140000 ]
[1] row(s) selected.

Mach> SELECT '[ ' || TO_CHAR(id5) || ' ]' FROM fixed_table;
'[ ' || TO_CHAR(id5) || ' ]'
--------------------------------------------------------------------------------
[ 7.833800 ]
[1] row(s) selected.

Mach> SELECT '[ ' || TO_CHAR(id6) || ' ]' FROM fixed_table;
'[ ' || TO_CHAR(id6) || ' ]'
--------------------------------------------------------------------------------
[ 192.168.0.1 ]
[1] row(s) selected.
```

```
Mach> SELECT '[ ' || TO_CHAR(id7) || ' ]' FROM fixed_table;
'[ ' || TO_CHAR(id7) || ' ]'
--------------------------------------------------------------------------------
[ 0000:0000:0000:0000:0000:0000:7F00:0001 ]
[1] row(s) selected.

Mach> SELECT '[ ' || TO_CHAR(id8) || ' ]' FROM fixed_table;
'[ ' || TO_CHAR(id8) || ' ]'
--------------------------------------------------------------------------------
[ log varchar ]
[1] row(s) selected.
```

TO_CHAR : Floating point

Supported since 5.5.6.

This function converts the values of float and double columns into arbitrary strings.

Format expressions cannot be duplicated and must be entered in the form '[character] [number]'.

Format expression	Description
F / f	Specifies the number of decimal places for column values. The maximum input numeric value is 30.
N / n	Specify the number of decimal places for the column value, and enter a comma (,) every three digits for the integer part. The maximum input numeric value is 30

```
Mach> create table float_table (i1 float, i2 double);
Created successfully.

Mach> insert into float_table values (1.23456789, 1234.5678901234567890);
1 row(s) inserted.

Mach> select TO_CHAR(i1, 'f8'), TO_CHAR(i2, 'N9') from float_table;
TO_CHAR(i1, 'f8')       TO_CHAR(i2, 'N9')
------------------------------------------------------------
1.23456788              1,234.567890123
[1] row(s) selected.
```

TO_CHAR: DATETIME Type

A function that converts the value of a datetime column to an arbitrary string. You can use this function to create and combine various types of strings.

If format_string is omitted, the default is "YYYY-MM-DD HH24: MI: SS mmm: uuu: nnn".

Format Expression	Description
YYYY	Converts year to a four-digit number.
YY	Converts year to a two-digit number.
MM	Converts the month to a two-digit number.
MON	Converts the month to a three-digit abbreviated alphabet. (eg JAN, FEB, MAY, ...)
DD	Converts the day to a two-digit number.
DAY	Converts the day of the week to a three-digit abbreviation . (eg SUN, MON, ...)
IW	Converts the week number of a specific year from 1 to 53 (taking into account the day of the week) by the ISO 8601 rule . • The start of one week is Monday. • The first week can be considered as the last week of the previous year. Likewise, the last week can be considered the first week of the next year. See ISO 8601 more information .
WW	Converts week number of the particular year from 1 to 53 (Week Number) not taking into account the day of the week. That is, from January 1 to January 7, it is converted to 1.
W	Converts week number of a given month from 1 to 5 (Number The Week) not taking in to account the day of the week.

Format Expression	Description
	That is, from March 1 to March 7 is converted to 1.
HH	Converts the time to a two-digit number.
HH12	Converts the time to a 2-digit number, from 1 to 12.
HH24	Converts the time to a 2-digit number, from 1 to 23.
HH2, HH3, HH6	Cuts the time to the number following HH. That is, when HH6 is used, 0 is expressed from 0 to 5, and 6 is expressed from 6 to 11. This expression is useful for calculating certain time-series statistics on time series. This value is expressed on a 24-hour basis.
MI	The minute is represented by a two-digit number.
MI2, MI5, MI10, MI20, MI30	The corresponding minute is cut to the number following MI. That is, when MI30 is used, 0 is expressed from 0 to 29 minutes, and 30 is represented from 30 to 59 minutes. This expression is useful for calculating certain time-series statistics on time series.
SS	The second is represented by a two-digit number.
SS2, SS5, SS10, SS20, SS30	The corresponding seconds are truncated to successive digits. That is, when SS30 is used, 0 is expressed from 0 to 29 seconds, and 30 is represented from 30 to 59 seconds. This expression is useful for calculating certain time-series statistics on time series.
AM	The current time is expressed in AM or PM according to AM and PM, respectively.
mmm	The millisecond of the time is represented by a three-digit number. The range of values is 0 to 999.
uuu	The micro second of the time is represented as a three-digit number. The range of values is 0 to 999.
nnn	The nano second of the time is expressed as a three-digit number. The range of values is 0 to 999.

```
Mach> CREATE TABLE datetime_table (id integer, dt datetime);
Created successfully.

Mach> INSERT INTO  datetime_table values(1, TO_DATE('1999-11-11 1:2:3 4:5:6'));
1 row(s) inserted.

Mach> INSERT INTO  datetime_table values(2, TO_DATE('2012-11-11 1:2:3 4:5:6'));
1 row(s) inserted.

Mach> INSERT INTO  datetime_table values(3, TO_DATE('2013-11-11 1:2:3 4:5:6'));
1 row(s) inserted.

Mach> INSERT INTO  datetime_table values(4, TO_DATE('2014-12-30 11:22:33 444:555:666'));
1 row(s) inserted.

Mach> SELECT id, dt FROM datetime_table WHERE dt > TO_DATE('2000-11-11 1:2:3 4:5:0');
id          dt
-----------------------------------------------
4           2014-12-30 11:22:33 444:555:666
3           2013-11-11 01:02:03 004:005:006
2           2012-11-11 01:02:03 004:005:006
[3] row(s) selected.

Mach> SELECT id, dt FROM datetime_table WHERE dt > TO_DATE('2013-11-11 1:2:3') and dt < TO_DATE('2014-11-11 1:2:3')
id          dt
-----------------------------------------------
3           2013-11-11 01:02:03 004:005:006
[1] row(s) selected.

Mach> SELECT id, TO_CHAR(dt) FROM datetime_table;
id          TO_CHAR(dt)
-------------------------------------------------------------------------------------------
4           2014-12-30 11:22:33 444:555:666
3           2013-11-11 01:02:03 004:005:006
```

```
2           2012-11-11 01:02:03 004:005:006
1           1999-11-11 01:02:03 004:005:006
[4] row(s) selected.

Mach> SELECT id, TO_CHAR(dt, 'YYYY') FROM datetime_table;
id          TO_CHAR(dt, 'YYYY')
----------------------------------------------------------------------------------------------
4           2014
3           2013
2           2012
1           1999
[4] row(s) selected.

Mach> SELECT id, TO_CHAR(dt, 'YYYY-MM') FROM datetime_table;
id          TO_CHAR(dt, 'YYYY-MM')
----------------------------------------------------------------------------------------------
4           2014-12
3           2013-11
2           2012-11
1           1999-11
[4] row(s) selected.

Mach> SELECT id, TO_CHAR(dt, 'YYYY-MM-DD') FROM datetime_table;
id          TO_CHAR(dt, 'YYYY-MM-DD')
----------------------------------------------------------------------------------------------
4           2014-12-30
3           2013-11-11
2           2012-11-11
1           1999-11-11
[4] row(s) selected.

Mach> SELECT id, TO_CHAR(dt, 'YYYY-MM-DD TO_CHAR') FROM datetime_table;
id          TO_CHAR(dt, 'YYYY-MM-DD TO_CHAR')
----------------------------------------------------------------------------------------------
4           2014-12-30 TO_CHAR
3           2013-11-11 TO_CHAR
2           2012-11-11 TO_CHAR
1           1999-11-11 TO_CHAR
[4] row(s) selected.

Mach> SELECT id, TO_CHAR(dt, 'YYYY-MM-DD HH24:MI:SS') FROM datetime_table;
id          TO_CHAR(dt, 'YYYY-MM-DD HH24:MI:SS')
----------------------------------------------------------------------------------------------
4           2014-12-30 11:22:33
3           2013-11-11 01:02:03
2           2012-11-11 01:02:03
1           1999-11-11 01:02:03
[4] row(s) selected.

Mach> SELECT id, TO_CHAR(dt, 'YYYY-MM-DD HH24:MI:SS mmm.uuu.nnn') FROM datetime_table;
id          TO_CHAR(dt, 'YYYY-MM-DD HH24:MI:SS mmm.
----------------------------------------------------------------------------------------------
4           2014-12-30 11:22:33 444.555.666
3           2013-11-11 01:02:03 004.005.006
2           2012-11-11 01:02:03 004.005.006
1           1999-11-11 01:02:03 004.005.006
[4] row(s) selected.
```

TO_CHAR: Unsupported Type

Currently, TO_CHAR is not supported for binary types.

This is because it is impossible to convert to plain text. If you want to output it to the screen, you can check it by outputting hexadecimal value through TO_HEX () function.

TO_DATE

This function converts a string represented by a given format string to a datetime type.

If format_string is omitted, the default is "YYYY-MM-DD HH24: MI: SS mmm: uuu: nnn".

```
-- default format is "YYYY-MM-DD HH24:MI:SS mmm:uuu:nnn" if no format exists.
TO_DATE(date_string [, format_string])
```

```
Mach> CREATE TABLE to_date_table (id INTEGER, dt datetime);
Created successfully.

Mach> INSERT INTO  to_date_table VALUES(1, TO_DATE('1999-11-11 1:2:3 4:5:6'));
1 row(s) inserted.

Mach> INSERT INTO  to_date_table VALUES(2, TO_DATE('2012-11-11 1:2:3 4:5:6'));
1 row(s) inserted.

Mach> INSERT INTO  to_date_table VALUES(3, TO_DATE('2014-12-30 11:22:33 444:555:666'));
1 row(s) inserted.

Mach> INSERT INTO  to_date_table VALUES(4, TO_DATE('2014-12-30 23:22:34 777:888:999', 'YYYY-MM-DD HH24:MI:SS mmm:uu
1 row(s) inserted.

Mach> SELECT id, dt FROM to_date_table WHERE dt > TO_DATE('1999-11-11 1:2:3 4:5:0');
id          dt
-----------------------------------------------
4           2014-12-30 23:22:34 777:888:999
3           2014-12-30 11:22:33 444:555:666
2           2012-11-11 01:02:03 004:005:006
1           1999-11-11 01:02:03 004:005:006
[4] row(s) selected.

Mach> SELECT id, dt FROM to_date_table WHERE dt > TO_DATE('2000-11-11 1:2:3 4:5:0');
id          dt
-----------------------------------------------
4           2014-12-30 23:22:34 777:888:999
3           2014-12-30 11:22:33 444:555:666
2           2012-11-11 01:02:03 004:005:006
[3] row(s) selected.

Mach> SELECT id, dt FROM to_date_table WHERE dt > TO_DATE('2012-11-11 1:2:3','YYYY-MM-DD HH24:MI:SS') and dt < TO_D
id          dt
-----------------------------------------------
2           2012-11-11 01:02:03 004:005:006
[1] row(s) selected.

Mach> SELECT id, TO_DATE('1999', 'YYYY') FROM to_date_table LIMIT 1;
id          TO_DATE('1999', 'YYYY')
-----------------------------------------------
4           1999-01-01 00:00:00 000:000:000
[1] row(s) selected.

Mach> SELECT id, TO_DATE('1999-12', 'YYYY-MM') FROM to_date_table LIMIT 1;
id          TO_DATE('1999-12', 'YYYY-MM')
-----------------------------------------------
4           1999.12.01 00:00:00 000:000:000
[1] row(s) selected.

Mach> SELECT id, TO_DATE('1999', 'YYYY') FROM to_date_table LIMIT 1;
id          TO_DATE('1999', 'YYYY')
-----------------------------------------------
4           1999-01-01 00:00:00 000:000:000
[1] row(s) selected.

Mach> SELECT id, TO_DATE('1999-12', 'YYYY-MM') FROM to_date_table LIMIT 1;
id          TO_DATE('1999-12', 'YYYY-MM')
-----------------------------------------------
4           1999-12-01 00:00:00 000:000:000
[1] row(s) selected.

Mach> SELECT id, TO_DATE('1999-12-31 13:12', 'YYYY-MM-DD HH24:MI') FROM to_date_table LIMIT 1;
id          TO_DATE('1999-12-31 13:12', 'YYYY-MM-DD HH24:MI')
```

```
----------------------------------------------------
4           1999-12-31 13:12:00 000:000:000
[1] row(s) selected.

Mach> SELECT id, TO_DATE('1999-12-31 13:12:32', 'YYYY-MM-DD HH24:MI:SS') FROM to_date_table LIMIT 1;
id          TO_DATE('1999-12-31 13:12:32', 'YYYY-MM-DD HH24:MI:SS')
----------------------------------------------------
4           1999-12-31 13:12:32 000:000:000
[1] row(s) selected.

Mach> SELECT id, TO_DATE('1999-12-31 13:12:32 123', 'YYYY-MM-DD HH24:MI:SS mmm') FROM to_date_table LIMIT 1;
id          TO_DATE('1999-12-31 13:12:32 123', 'YYYY-MM-DD HH24:MI:SS mmm')
----------------------------------------------------
4           1999-12-31 13:12:32 123:000:000
[1] row(s) selected.

Mach> SELECT id, TO_DATE('1999-12-31 13:12:32 123:456', 'YYYY-MM-DD HH24:MI:SS mmm:uuu') FROM to_date_table LIMIT 1
id          TO_DATE('1999-12-31 13:12:32 123:456', 'YYYY-MM-DD HH24:MI:SS mmm:uuu')
----------------------------------------------------
4           1999-12-31 13:12:32 123:456:000
[1] row(s) selected.

Mach> SELECT id, TO_DATE('1999-12-31 13:12:32 123:456:789', 'YYYY-MM-DD HH24:MI:SS mmm:uuu:nnn') FROM to_date_table
id           TO_DATE('1999-12-31 13:12:32 123:456:789', 'YYYY-MM-DD HH24:MI:SS mmm:uuu:nnn')
----------------------------------------------------
4           1999-12-31 13:12:32 123:456:789
[1] row(s) selected.
```

TO_DATE_SAFE

Similar to TO_DATE (), but returns NULL without error if conversion fails.

```
TO_DATE_SAFE(date_string [, format_string])
```

```
Mach> CREATE TABLE date_table (ts DATETIME);
Created successfully.

Mach> INSERT INTO date_table VALUES (TO_DATE_SAFE('2016-01-01', 'YYYY-MM-DD'));
1 row(s) inserted.
Mach> INSERT INTO date_table VALUES (TO_DATE_SAFE('2016-01-02', 'YYYY'));
1 row(s) inserted.
Mach> INSERT INTO date_table VALUES (TO_DATE_SAFE('2016-12-32', 'YYYY-MM-DD'));
1 row(s) inserted.

Mach> SELECT ts FROM date_table;
ts
---------------------------------
NULL
NULL
2016-01-01 00:00:00 000:000:000
[3] row(s) selected.
```

TO_HEX

This function returns value if the value of the column is NULL, or the value of the original column if it is not NULL. To ensure consistency of output, convert to BIG ENDIAN type for short, int, and long types.

```
TO_HEX(column)
```

```
Mach> CREATE TABLE hex_table (id1 SHORT, id2 INTEGER, id3 VARCHAR(10), id4 FLOAT, id5 DOUBLE, id6 LONG, id7 IPV4, i
id11 DATETIME);
Created successfully.
```

```
Mach> INSERT INTO hex_table VALUES(256, 65535, '0123456789', 3.141592, 1024 * 1024 * 1024 * 3.14, 13513135446, '192
'binary', TO_DATE('1999', 'YYYY'));
1 row(s) inserted.

Mach> SELECT TO_HEX(id1), TO_HEX(id2), TO_HEX(id3), TO_HEX(id4), TO_HEX(id5), TO_HEX(id6), TO_HEX(id7), TO_HEX(id8)
FROM hex_table;
TO_HEX(id1)  TO_HEX(id2)  TO_HEX(id3)            TO_HEX(id4)  TO_HEX(id5)        TO_HEX(id6)        TO_HEX(id7)
-------------------------------------------------------------------------------------------------------------------
TO_HEX(id8)                         TO_HEX(id9)
-------------------------------------------------------------------------------------------------------------------
TO_HEX(id10)                                                                        TO_HEX(id11)
-------------------------------------------------------------------------------------------------------------
0100   0000FFFF   30313233343536373839   D80F4940   1F85EB51B81EE941   0000000325721556   04C0A80001
0600000000000000000000000C0A80001   74657874657874
62696E617279                                                                        0CB325846E226000
[1] row(s) selected.
```

TO_IPV4 / TO_IPV4_SAFE

This function converts a given string to an IP version 4 type. If the string can not be converted to a numeric value, the TO_IPV4 () function returns an error and terminates the operation.

However, in the case of the TO_IPV4_SAFE () function, NULL is returned in case of error, and the operation can continue.

```
TO_IPV4(string_value)
TO_IPV4_SAFE(string_value)
```

```
Mach> CREATE TABLE ipv4_table (c1 varchar(100));
Created successfully.

Mach> INSERT INTO ipv4_table VALUES('192.168.0.1');
1 row(s) inserted.

Mach> INSERT INTO ipv4_table VALUES('     192.168.0.2    ');
1 row(s) inserted.

Mach> INSERT INTO ipv4_table VALUES(NULL);
1 row(s) inserted.

Mach> SELECT c1 FROM ipv4_table;
c1
-------------------------------------------------------------------------------
NULL
     192.168.0.2
192.168.0.1
[3] row(s) selected.

Mach> SELECT TO_IPV4(c1) FROM ipv4_table;
TO_IPV4(c1)
------------------
NULL
192.168.0.2
192.168.0.1
[3] row(s) selected.

Mach> INSERT INTO ipv4_table VALUES('192.168.0.1.1');
1 row(s) inserted.

Mach> SELECT TO_IPV4(c1) FROM ipv4_table limit 1;
TO_IPV4(c1)
------------------
[ERR-02068 : Invalid IPv4 address format (192.168.0.1.1).]
[0] row(s) selected.

Mach> SELECT TO_IPV4_SAFE(c1) FROM ipv4_table;
TO_IPV4_SAFE(c1)
------------------
```

```
NULL
NULL
192.168.0.2
192.168.0.1
[4] row(s) selected.
```

TO_IPV6 / TO_IPV6_SAFE

This function converts a given string to an IP version 6 type. If the string can not be converted to a numeric type, the TO_IPV6 () function returns an error and terminates the operation.

However, in the case of the TO_IPV6_SAFE () function, NULL is returned in case of error, and the operation can continue.

```
TO_IPV6(string_value)
TO_IPV6_SAFE(string_value)
```

```
Mach> CREATE TABLE ipv6_table (id varchar(100));
Created successfully.

Mach> INSERT INTO ipv6_table VALUES('::0.0.0.0');
1 row(s) inserted.

Mach> INSERT INTO ipv6_table VALUES('::127.0.0.1');
1 row(s) inserted.

Mach> INSERT INTO ipv6_table VALUES('::127.0' || '.0.2');
1 row(s) inserted.

Mach> INSERT INTO ipv6_table VALUES('   ::127.0.0.3');
1 row(s) inserted.

Mach> INSERT INTO ipv6_table VALUES('::127.0.0.4  ');
1 row(s) inserted.

Mach> INSERT INTO ipv6_table VALUES('   ::FFFF:255.255.255.255   ');
1 row(s) inserted.

Mach> INSERT INTO ipv6_table VALUES('21DA:D3:0:2F3B:2AA:FF:FE28:9C5A');
1 row(s) inserted.

Mach> SELECT TO_IPV6(id) FROM ipv6_table;
TO_IPV6(id)
----------------------------------------------------------------
21da:d3::2f3b:2aa:ff:fe28:9c5a
::ffff:255.255.255.255
::127.0.0.4
::127.0.0.3
::127.0.0.2
::127.0.0.1
::
[7] row(s) selected.

Mach> INSERT INTO ipv6_table VALUES('127.0.0.10.10');
1 row(s) inserted.

Mach> SELECT TO_IPV6(id) FROM ipv6_table limit 1;
TO_IPV6(id)
----------------------------------------------------------------
[ERR-02148 : Invalid IPv6 address format.(127.0.0.10.10)]
[0] row(s) selected.

Mach> SELECT TO_IPV6_SAFE(id) FROM ipv6_table;
TO_IPV6_SAFE(id)
----------------------------------------------------------------
NULL
21da:d3::2f3b:2aa:ff:fe28:9c5a
::ffff:255.255.255.255
```

```
::127.0.0.4
::127.0.0.3
::127.0.0.2
::127.0.0.1
::
[8] row(s) selected.
```

TO_NUMBER / TO_NUMBER_SAFE

This function converts a given string to a numeric double. If the string can not be converted to a numeric value, the TO_NUMBER () function returns an error and terminates the operation.

However, in case of TO_NUMBER_SAFE () function, NULL is returned in case of error, and the operation can continue.

```
TO_NUMBER(string_value)
TO_NUMBER_SAFE(string_value)
```

```
Mach> CREATE TABLE number_table (id varchar(100));
Created successfully.

Mach> INSERT INTO number_table VALUES('10');
1 row(s) inserted.

Mach> INSERT INTO number_table VALUES('20');
1 row(s) inserted.

Mach> INSERT INTO number_table VALUES('30');
1 row(s) inserted.

Mach> SELECT TO_NUMBER(id) from number_table;
TO_NUMBER(id)
------------------------------
30
20
10
[3] row(s) selected.

Mach> CREATE TABLE safe_table (id varchar(100));
Created successfully.

Mach> INSERT INTO safe_table VALUES('invalidnumber');
1 row(s) inserted.

Mach> SELECT TO_NUMBER(id) from safe_table;
TO_NUMBER(id)
------------------------------
[ERR-02145 : The string cannot be converted to number value.(invalidnumber)]
[0] row(s) selected.

Mach> SELECT TO_NUMBER_SAFE(id) from safe_table;
TO_NUMBER_SAFE(id)
------------------------------
NULL
[1] row(s) selected.
```

TO_TIMESTAMP

This function converts a datetime data type to nanosecond data that has passed since 1970-01-01 09:00.

```
TO_TIMESTAMP(datetime_value)
```

```
Mach> create table datetime_tbl (c1 datetime);
Created successfully.
```

```
Mach> insert into datetime_tbl values ('2010-01-01 10:10:10');
1 row(s) inserted.

Mach> select to_timestamp(c1) from datetime_tbl;
to_timestamp(c1)
----------------------
1262308210000000000
[1] row(s) selected.
```

TRUNC

The TRUNC function returns the number truncated at the nth place after the decimal point.

If n is omitted, treat it as 0 and delete all decimal places. If n is negative, it returns the value truncated from n before the decimal point.

```
TRUNC(number [, n])
```

```
Mach> CREATE TABLE trunc_table (i1 DOUBLE);
Created successfully.

Mach> INSERT INTO trunc_table VALUES (158.799);
1 row(s) inserted.

Mach> SELECT TRUNC(i1, 1), TRUNC(i1, -1) FROM trunc_table;
TRUNC(i1, 1)                TRUNC(i1, -1)
------------------------------------------------------------
158.7                       150
[1] row(s) selected.

Mach> SELECT TRUNC(i1, 2), TRUNC(i1, -2) FROM trunc_table;
TRUNC(i1, 2)                TRUNC(i1, -2)
------------------------------------------------------------
158.79                      100
[1] row(s) selected.
```

TS_CHANGE_COUNT

This function is an aggregate function that obtains the number of changes to a particular column value.

This function can not be used with 1) Join or 2) Inline view because it can be guaranteed that input data is input in chronological order.
The current version only supports types except varchar.

```
TS_CHANGE_COUNT(column)
```

```
Mach> CREATE TABLE ipcount_table (id INTEGER, ip IPV4);
Created successfully.

Mach> INSERT INTO ipcount_table VALUES (1, '192.168.0.1');
1 row(s) inserted.

Mach> INSERT INTO ipcount_table VALUES (1, '192.168.0.2');
1 row(s) inserted.

Mach> INSERT INTO ipcount_table VALUES (1, '192.168.0.1');
1 row(s) inserted.

Mach> INSERT INTO ipcount_table VALUES (1, '192.168.0.2');
1 row(s) inserted.

Mach> INSERT INTO ipcount_table VALUES (2, '192.168.0.3');
```

```
1 row(s) inserted.

Mach> INSERT INTO ipcount_table VALUES (2, '192.168.0.3');
1 row(s) inserted.

Mach> INSERT INTO ipcount_table VALUES (2, '192.168.0.4');
1 row(s) inserted.

Mach> INSERT INTO ipcount_table VALUES (2, '192.168.0.4');
1 row(s) inserted.

Mach> SELECT id, TS_CHANGE_COUNT(ip) from ipcount_table GROUP BY id;
id          TS_CHANGE_COUNT(ip)
-----------------------------------
2           2
1           4
[2] row(s) selected.
```

UNIX_TIMESTAMP

UNIX_TIMESTAMP is a function that converts a date type value to a 32-bit integer value that is converted by unix's time () system call. (FROM_UNIXTIME is a function that converts integer data to a date type value on the contrary.)

```
UNIX_TIMESTAMP(datetime_value)
```

```
Mach> CREATE table unix_table (c1 int);
Created successfully.

Mach> INSERT INTO unix_table VALUES (UNIX_TIMESTAMP('2001-01-01'));
1 row(s) inserted.

Mach> SELECT * FROM unix_table;
C1
--------------
978274800
[1] row(s) selected.
```

UPPER

This function converts the contents of a given English column to uppercase.

```
UPPER(string_value)
```

```
Mach> CREATE TABLE upper_table(id INTEGER,name VARCHAR(10));
Created successfully.

Mach> INSERT INTO upper_table VALUES(1, '');
1 row(s) inserted.

Mach> INSERT INTO upper_table VALUES(2, 'James');
1 row(s) inserted.

Mach> INSERT INTO upper_table VALUES(3, 'sarah');
1 row(s) inserted.

Mach> INSERT INTO upper_table VALUES(4, 'THOMAS');
1 row(s) inserted.

Mach> SELECT id, UPPER(name) FROM upper_table;
id          UPPER(name)
----------------------------
4           THOMAS
```

```
3           SARAH
2           JAMES
1           NULL
[4] row(s) selected.
```

VARIANCE / VAR_POP

This function is an aggregate function that returns the variance of a given numeric column value. The Variance function returns the variance for the sample, and the VAR_POP function returns the variance for the population.

```
VARIANCE(column_name)
VAR_POP(column_name)
```

```
Mach> CREATE TABLE var_table(c1 INTEGER, c2 DOUBLE);
Created successfully.

Mach> INSERT INTO var_table VALUES (1, 1);
1 row(s) inserted.

Mach> INSERT INTO var_table VALUES (2, 1);
1 row(s) inserted.

Mach> INSERT INTO var_table VALUES (1, 2);
1 row(s) inserted.

Mach> INSERT INTO var_table VALUES (2, 2);
1 row(s) inserted.

Mach> SELECT VARIANCE(c1) FROM var_table;
VARIANCE(c1)
------------------------------
0.333333
[1] row(s) selected.

Mach> SELECT VAR_POP(c1) FROM var_table;
VAR_POP(c1)
------------------------------
0.25
[1] row(s) selected.
```

YEAR / MONTH / DAY

These functions extract the corresponding year, month, and day from the input datetime column value and return it as an integer type value.

```
YEAR(datetime_col)
MONTH(datetime_col)
DAY(datetime_col)
```

```
Mach> CREATE TABLE extract_table(c1 DATETIME, c2 INTEGER);
Created successfully.

Mach> INSERT INTO extract_table VALUES (to_date('2001-01-01 12:30:00 000:000:000'), 1);
1 row(s) inserted.

Mach> SELECT YEAR(c1), MONTH(c1), DAY(c1) FROM extract_table;
year(c1)    month(c1)   day(c1)
---------------------------------------
2001        1           1
```

Support Type of Built-In Function

	Short	Integer	Long	Float	Double	Varchar	Text	Ipv4	Ipv6	D
ABS	o	o	o	o	o	x	x	x	x	x
ADD_TIME	x	x	x	x	x	x	x	x	x	o
AVG	o	o	o	o	o	x	x	x	x	x
BITAND / BITOR	o	o	o	x	x	x	x	x	x	x
COUNT	o	o	o	o	o	o	x	o	o	o
DATE_TRUNC	x	x	x	x	x	x	x	x	x	o
DECODE	o	o	o	o	o	o	x	o	x	o
FIRST / LAST	o	o	o	o	o	o	x	o	o	o
FROM_UNIXTIME	o	o	o	o	o	x	x	x	x	x
FROM_TIMESTAMP	o	o	o	o	o	x	x	x	x	x
GROUP_CONCAT	o	o	o	o	o	o	x	o	o	o
INSTR	x	x	x	x	x	o	o	x	x	x
LEAST / GREATEST	o	o	o	o	o	o	x	x	x	x
LENGTH	x	x	x	x	x	o	o	x	x	x
LOWER	x	x	x	x	x	o	x	x	x	x
LPAD / RPAD	x	x	x	x	x	o	x	x	x	x
LTRIM / RTRIM	x	x	x	x	x	o	x	x	x	x
MAX	o	o	o	o	o	o	x	o	o	o
MIX	o	o	o	o	o	o	x	o	o	o
NVL	x	x	x	x	x	o	x	o	x	x
ROUND	o	o	o	o	o	x	x	x	x	x
ROWNUM	o	o	o	o	o	o	o	o	o	o
SERIESNUM	o	o	o	o	o	o	o	o	o	o
STDDEV / STDDEV_POP	o	o	o	o	o	x	x	x	x	x
SUBSTR	x	x	x	x	x	o	x	x	x	x
SUBSTRING_INDEX	x	x	x	x	x	o	o	x	x	x
SUM	o	o	o	o	o	x	x	x	x	x
SYSDATE / NOW	x	x	x	x	x	x	x	x	x	x

	Short	Integer	Long	Float	Double	Varchar	Text	Ipv4	Ipv6	D
TO_CHAR	o	o	o	o	o	o	x	o	o	o
TO_DATE / TO_DATE_SAFE	x	x	x	x	x	o	x	x	x	x
TO_HEX	o	o	o	o	o	o	o	o	o	o
TO_IPV4 / TO_IPV4_SAFE	x	x	x	x	x	o	x	x	x	x
TO_IPV6 / TO_IPV6_SAFE	x	x	x	x	x	o	x	x	x	x
TO_NUMBER / TO_NUMBER_SAFE	x	x	x	x	x	o	x	x	x	x
TO_TIMESTAMP	x	x	x	x	x	x	x	x	x	o
TRUNC	o	o	o	o	o	x	x	x	x	x
TS_CHANGE_COUNT	o	o	o	o	o	x	x	o	o	o
UNIX_TIMESTAMP	o	o	o	o	o	x	x	x	x	x
UPPER	x	x	x	x	x	o	x	x	x	x
VARIANCE / VAR_POP	o	o	o	o	o	x	x	x	x	x
YEAR / MONTH / DAY	x	x	x	x	x	x	x	x	x	o

ALTER SYSTEM

This statement is the syntax for managing system-wide resources or changing settings.

KILL SESSION

alter_system_kill_session_stmt:


```
alter_system_kill_session_stmt: 'ALTER SYSTEM KILL SESSION' number
```

Terminates a specific session with a SessionID.

However, only the SYS user can execute this statement and can not KILL their own session.

CANCEL SESSION

alter_system_cancel_session_stmt:


```
alter_system_cancel_session_stmt ::= 'ALTER SYSTEM CANCEL SESSION' number
```

Cancels a specific session with a SessionID.

Rather than disconnecting the connection, it cancels the action being performed and returns an error code to the user that the action was aborted. However, like KILL, you can not cancel your own connected sessions.

CHECK DISK_USAGE

alter_system_check_disk_stmt:


```
alter_system_check_disk_stmt ::= 'ALTER SYSTEM CHECK DISK_USAGE'
```

Corrects the value of **DC_TABLE_FILE_SIZE**, which indicates the disk usage of the log table in V$STORAGE.

Disk usage may be inaccurate when process failures or power failures occur. This command reads the correct value from the file system. However, it should be avoided because it can put a considerable load on the file system.

INSTALL LICENSE

alter_system_install_license_stmt:


```
alter_system_install_license_stmt ::= 'ALTER SYSTEM INSTALL LICENSE'
```

Installs the license file in the default location of the license file ($MACHBASE_HOME/conf/license.dat).

It is installed after determining whether the license is suitable for installation.

INSTALL LICENSE (PATH)

alter_system_install_license_path_stmt:


```
alter_system_install_license_path_stmt: ::= 'ALTER SYSTEM INSTALL LICENSE' '='
```

Index

Installs the license file in a specific location.

An error occurs when you enter a license file that does not exist at that location or is incorrect. The path must be an absolute path. It is installed after determining whether the license is suitable for installation.

SET

> Support this after 5.7 version

alter_system_set_stmt:


```
alter_system_set_stmt ::= 'ALTER SYSTEM SET' prop_name '=' value
```

Can modify system property as follows.

- QUERY_PARALLEL_FACTOR
- DEFAULT_DATE_FORMAT
- TRACE_LOG_LEVEL
- PAGE_CACHE_MAX_SIZE

ALTER SESSION

This is the syntax for managing resources or changing settings on a per-session basis.

SET SQL_LOGGING

alter_session_sql_logging_stmt:


```
alter_session_sql_logging_stmt ::= 'ALTER SESSION SET SQL_LOGGING' '=' flag
```

Determines whether to leave a message in the Trace Log of the session.

You can use this message as a Bit Flag with the following values:

- 0x1: Parsing, Validation, Optimization.
- 0x2: It leaves the result of performing DDL.

That is, when the value of the corresponding flag is 2, only the DDL is logged, and when the flag is 3, the error and DDL are logged together.
Below is an example of changing the logging flag of the session and leaving error logging.

```
Mach> alter session set SQL_LOGGING=1;
Altered successfully.
Mach> exit
```

SET DEFAULT_DATE_FORMAT

alter_session_set_defalut_dateformat_stmt:


```
alter_session_set_defalut_dateformat_stmt ::= 'ALTER SESSION SET DEFAULT_DATE_FORMAT' '=' date_format
```

Sets the default format for Datetime data types for this session.

When the server is started, the property **DEFAULT_DATE_FORMAT** is set to the session attribute.
If the property of the property has not changed, the value of the session will also be "YYYY-MM-DD HH24: MI: SS mmm: uuu: nnn".
Use this command to modify the default format of a datetime datatype for a specific user, regardless of the system.
V$session has a default date format set for each session and can be checked. Below is an example of checking and changing the value of the session.

```
Mach> CREATE TABLE time_table (time datetime);
Created successfully.
```

```
Mach> SELECT DEFAULT_DATE_FORMAT from v$session;
default_date_format
-------------------------------------------
YYYY-MM-DD HH24:MI:SS mmm:uuu:nnn
[1] row(s) selected.

Mach> INSERT INTO time_table VALUES(TO_DATE('2016-11-11'));
[ERR-00300 : Invalid date format or input string.([2016-11-11]:[%Y-%m-%d %H:%M:%S %0:%1:%2])]

Mach> ALTER SESSION SET DEFAULT_DATE_FORMAT='YYYY-MM-DD';
Altered successfully.

Mach> SELECT DEFAULT_DATE_FORMAT from v$session;

default_date_format
-------------------------------------------
YYYY-MM-DD
[1] row(s) selected.

Mach> INSERT INTO time_table VALUES(TO_DATE('2016-11-11'));
1 row(s) inserted.

Mach> SELECT * FROM time_table;

TIME
---------------------------------
2016-11-11

[1] row(s) selected.
```

SET SHOW_HIDDEN_COLS

alter_session_set_hidden_column_stmt:


```
alter_session_set_hidden_column_stmt ::= 'ALTER SESSION SET SHOW_HIDDEN_COLS' '=' ( '0' | '1' )
```

Decides whether to output the hidden column (_arrival_time) in the column represented by * when executing the select of the session.

When the server is started, the value of the global property SHOW_HIDDEN_COLS is set to 0 for the session attribute.
If you want to change the default behavior of your session, you can set this value to 1.
V$session has a SHOW_HIDDEN_COLS value set for each session.

```
Mach> SELECT * FROM  v$session;
ID                  CLOSED      USER_ID     LOGIN_TIME                      SQL_LOGGING SHOW_HIDDEN_COLS
-------------------------------------------------------------------------------------------------------------------
DEFAULT_DATE_FORMAT                                                              HASH_BUCKET_SIZE
-------------------------------------------------------------------------------------------------------
1                   0           1           2015-04-29 17:23:56 248:263:000 3           0
YYYY-MM-DD HH24:MI:SS mmm:uuu:nnn                                                20011
[1] row(s) selected.
Mach> ALTER SESSION SET SHOW_HIDDEN_COLS=1;
Altered successfully.
Mach> SELECT * FROM v$session;
_ARRIVAL_TIME                   ID                  CLOSED      USER_ID     LOGIN_TIME                      SQL_LO
-------------------------------------------------------------------------------------------------------------------
SHOW_HIDDEN_COLS DEFAULT_DATE_FORMAT                                                                  HASH_BUCKET_SIZE
-------------------------------------------------------------------------------------------------------------------
1970-01-01 09:00:00 000:000:000 1                   0           1           2015-04-29 17:23:56 248:263:000 3
1           YYYY-MM-DD HH24:MI:SS mmm:uuu:nnn                                                     20011
[1] row(s) selected.
```

SET FEEDBACK_APPEND_ERROR

alter_session_set_feedback_append_err_stmt:


```
alter_session_set_feedback_append_err_stmt ::= 'ALTER SESSION SET FEEDBACK_APPEND_ERROR' '=' ( '0' | '1' )
```

Sets whether to send the session's Append error message to the client program.

Use the following values for the error message.

- 0 = Do not send an error message.
- 1 = Send an error message.

Below is an example of use.

```
mach> ALTER SESSION SET FEEDBACK_APPEND_ERROR=0;
Altered successfully.
```

SET HASH_BUCKET_SIZE

alter_session_set_hash_bucket_size_stmt:


```
alter_session_set_hash_bucket_size_stmt ::= 'ALTER SESSION SET HASH_BUCKET_SIZE' '=' value
```

Sets the size of the hash table used to perform the GROUP BY or Distinct operation of the session.

If this value is set too large, memory usage will be heavy for each Hash operation. If too small, Hash bucket conflict will occur and query performance may be degraded.
It is recommended to specify the number of whole groups * 1.5 to 3.0.

```
Mach> ALTER SESSION SET HASH_BUCKET_SIZE=65536;
Altered successfully.

Mach> SELECT * FROM v$session;
_ARRIVAL_TIME                   ID                   CLOSED      USER_ID     LOGIN_TIME                      SQL_LO
-------------------------------------------------------------------------------------------------------------------
SHOW_HIDDEN_COLS DEFAULT_DATE_FORMAT                                                          HASH_BUCKET_SIZE
-------------------------------------------------------------------------------------------------------------------
1970-01-01 09:00:00 000:000:000 1                    0           1           2015-04-29 17:23:56 248:263:000 3
1           YYYY-MM-DD HH24:MI:SS mmm:uuu:nnn                                                 65536
[1] row(s) selected.
```

SET MAX_QPX_MEM

alter_session_set_max_qpx_mem_stmt:


```
alter_session_set_max_qpx_mem_stmt ::= 'ALTER SESSION SET MAX_QPX_MEM' '=' value
```

Specifies the maximum amount of memory that a single SQL statement in the session will use when performing GROUP BY, DISTINCT, ORDER BY operations.

If you try to allocate more memory than the maximum memory, the system cancels the execution of the SQL statement and treats it as an error.
In case of error, record the error code and error message in machbase.trc including the query.

```
Mach> ALTER SESSION SET MAX_QPX_MEM=1073741824;
Altered successfully.

Mach> SELECT * FROM v$session;
ID                  CLOSED      USER_ID     LOGIN_TIME                      SQL_LOGGING SHOW_HIDDEN_COLS FEEDBACK_
-------------------------------------------------------------------------------------------------------------------
```

```
DEFAULT_DATE_FORMAT                                                                  HASH_BUCKET_SIZE MAX_QPX_MEM
--------------------------------------------------------------------------------------------------------------------
324                 0           1           2015-07-14 10:53:46 124:627:000 11          0           0
YYYY-MM-DD HH24:MI:SS mmm:uuu:nnn                                                    20011       1073741824
[1] row(s) selected.
Mach>
```

SDK

This document describes the MACHBASE SDK (Software Development Kit).

CLI/ODBC

CLI is a software development standard defined in ISO/IEC 9075-3: 2003.

The CLI defines functions and specifications for how to pass SQL to the database and how to receive and analyze the results. This CLI was developed in the early 1990s and was developed exclusively for C and COBOL languages, and its specifications have been maintained to date.

The most widely known standard interface to date is ODBC (Open Database Connectivity), which provides a way for a client program to access a database regardless of the type of database. The current ODBC API version is 3.52 and is defined in ISO and X/Open standards.

Standard CLI Functions

See the following links for usage of the standard functions.

- Wikipedia
- Open Group Document

You can refer to the following function.

SQLAllocConnect	SQLDisconnect	SQLGetDescField	SQLPrepare
SQLAllocEnv	SQLDriverConnect	SQLGetDescRec	SQLPrimaryKeys
SQLAllocHandle	SQLExecDirect	SQLGetDiagRec	SQLStatistics
SQLAllocStmt	SQLExecute	SQLGetEnvAttr	SQLRowCount
SQLBindCol	SQLFetch	SQLGetFunctions	SQLSetConnectAttr
SQLBindParameter+	SQLFreeConnect	SQLGetInfo	SQLSetDescField
SQLColAttribute	SQLFreeEnv	SQLGetStmtAttr	SQLSetDescRec
SQLColumns	SQLFreeHandle	SQLGetTypeInfo	SQLSetEnvAttr
SQLConnect	SQLFreeStmt	SQLNativeSQL	SQLSetStmtAttr
SQLCopyDesc	SQLGetConnectAttr	SQLNumParams	SQLStatistics
SQLDescribeCol	SQLGetData	SQLNumResultCols	SQLTables

Connection String for Connecting

To connect through the CLI, you need to create a connection string. The contents of each are as follows.

Connection String Item Name	Item Description
DSN	Specifies the data source name. ODBC specifies the section name of the file containing the resource, and CLI specifies the server name or IP address.
DBNAME	Describes the DB name of Machbase.
SERVER	Indicates the host name or IP address of the server where Machbase is located.
NLS_USE	Sets the language type to use with each other (currently unused, kept for future expansion).
UID	User ID
PWD	User password

Connection String Item Name	Item Description
PORT_NO	Port number to connect to
PORT_DIR	Specifies the file path to use when connecting to a Unix domain from Unix. (It is specified when modified from the server, and it works even if it is not specified by default.)
CONNTYPE	Specifies the connection method between the client and the server. 1: Connection with TCP / IP INET 2: Connect to Unix Domain
COMPRESS	Indicates whether to compress the Append protocol. If this value is 0, it is transmitted without compression. If this value is any value greater than 0, it is compressed only if the Append record is larger than its value. Ex) COMPRESS = 512 Only when the record size is larger than 512, it is compressed and operates. For remote connection, compression improves transmission performance.
SHOW_HIDDEN_COLS	Decides whether to show the hidden column (_arrival_time) when executing it with select *. If it is 0, it is not shown. If it is 1, information of the corresponding column is output.
CONNECTION_TIMEOUT	Sets how long to wait on the first connection. The default setting is 30 seconds. This value is set higher if the server response on the first connection is slower than 30 seconds.

An example of CLI connection is as follows.

```
sprintf(connStr,"SERVER=127.0.0.1;COMPRESS=512;UID=SYS;PWD=MANAGER;CONNTYPE=1;PORT_NO=%d", MACHBASE_PORT_NO);

if (SQL_ERROR == SQLDriverConnect( gCon, NULL, (SQLCHAR *)connStr, SQL_NTS, NULL, 0, NULL, SQL_DRIVER_NOPROMPT )) {
   ...
}
```

Extension CLI Function (APPEND)

The CLI extension function is a function for implementing the Append protocol provided to input data to the Machbase server at high speed.

This function consists of four functions: channel open, channel data input, channel flush, and channel closing.

Understanding Append Protocol

The Append protocol provided by Machbase works asynchronously. The term asynchronous means that the response to a specific job requested by the client to the server does not completely synchronize with each other but occurs at the moment when an arbitrary event occurs. That is, even if a client has performed an append, you can not immediately get or verify the results of that execution, and you can check it at any time when the server is ready. For this reason, developers who develop applications using the Append protocol should have an understanding of the following internal behaviors. The following discussion is about how and when a client detects asynchronous errors that occur in the server.

Append Data Transfer

In a typical call such as SQLExecute or SQLExecDirect (), Machbase uses a synchronous scheme that returns the results back to the client immediately. However, SQLAppendDataV2 () does not send a request immediately after user data is entered. Instead, it waits until all of the client communication buffers are full, and then it sends the data to the client all at once. The reason for this design is that the input data of the client using Append assumes tens to hundreds of thousands of records per second, so it utilizes the buffering method for high-speed data transmission. For this reason, if the user wants to transmit the contents of the buffer at will, the user can input data explicitly by calling SQLAppendFlush () function.

Append Data Error Check

As mentioned earlier, the Append protocol is buffered and operates asynchronously. In particular, it is very important to understand when and how an error is detected because it takes a method to detect an error only when an error occurs, without receiving any response when an error does not occur in the server. In addition, since the cost of detecting an error is relatively large, it is very inefficient to check each time a record is input, and currently Machbase is designed to detect an error only in the following cases explicitly. When an error is detected, the error callback function set by the user is called every time.

1. Checks after all the transmit buffers are full and the data has been explicitly sent to the server,
2. Checks after explicitly sending data to the server from within SQLAppendFlush ()
3. Checks just before shut down from within SQLAppendClose ()

In other words, it is basically designed to detect errors only in the above three cases, and is designed to minimize the occurrence of I/O.

Additional Options for Checking Server Errors

In order to achieve the maximum performance, the default error detection technique can be more frequently checked and utilized by the user if desired. This can be done by adjusting the last argument to the SQLAppendOpen () function, aErrorCheckCount. When this value is 0, it does not perform any checking operation and operates basically. However, if this value is greater than 0, SQLAppendData () is explicitly checked for errors every time it is called. In other words, if this value is 10, you pay the cost of checking for errors every 10 appends. Therefore, when this value is small, system resources for error detection are used much, so it should be adjusted to an appropriate number.

Leaving Trace Log When Server Error Occurs

If you want to leave a trace log for the append data where an error occurs, set the prepared property DUMP_APPEND_ERROR to 1 on the server. With this setting, the specification of the record that generated the error in the mach.trc file is written to the file. However, if the number of errors is excessive, the amount of system resources used will increase drastically, which may degrade the overall performance of Machbase.

APPEND Function Description

SQLAppendOpen

```
SQLRETURN SQLAppendOpen(SQLHSTMT   aStatementHandle,
                        SQLCHAR   *aTableName,
                        SQLINTEGER aErrorCheckCount );
```

This function opens a channel for the target table. If this channel is not closed afterwards, it is kept open continuously.
A maximum of 1024 statements can be set for one connection. You can use SQLAppendOpen for each statement.

1. aStatementHandle: Represents the handle of the Statement to be appended.
2. aTableName: Indicates the name of the table to which Append will be performed.
3. aErrorCheckCount: Decides whether to check the server for errors whenever several data are input. If this value is 0, no error is checked arbitrarily.

SQLAppendData (deprecated)

```
SQLRETURN  SQLAppendData(SQLHSTMT StatementHandle, void *aData[]);
```

This function is a function that inputs data for the channel.

- aData is an array containing pointers to the data to be input. The number of arrays must match the number of columns held by the table specified at Open.
- The return value can be SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, or SQL_ERROR.
  In particular, if SQL_SUCCESS_WITH_INFO is returned, there may be errors such as a lengthy input column being truncated, so check the result again.

Configuration According to Data Type

Numeric and character types

- Types such as float, double, short, int, long long, and char * work well with pointers to their values.

Address type

- In the case of ipv4, it is passed as an array of 5-byte unsigned char.
- The first byte is set to 4, the next 4 bytes are set to consecutive address values.
- For example, in the case of 127.0.0.1, five byte arrays **0x04, 0x7f, 0x00, 0x00, and 0x01** are entered in order.

```
// For tables with four column information (short (16), int (32), long (64), varchar)

testAppendIPFunc()
{
   short val1 = 0;
   int   val2 = 1;
   long long  val3 = 2;
   char *val4 = "my string";
   void *valueArray[4];

   valueArray[0] = (void *)&val1;
   valueArray[1] = (void *)&val2;
   valueArray[2] = (void *)&val3;
   valueArray[3] = (void *)val4;

   SQLAppendData(aStmt, valueArray);
}
```

Configuration According to Data Type

datetime type

- Since Machbase internally has a nano-unit time resolution value, it must be converted when setting the time on the client, and it is expressed as a 64-bit unsigned integer value.
  Therefore, for proper conversion, you need to add nano values after converting to seconds using the UNIX library mktime.
  ※ Machbase time = (total time (seconds) since January 1, 1970) * 1,000,000,000 + milli-second * 1,000,000 + micro-second * 1000 + nano-second;

```
// Code if Date String is entered as "Year - Month - Date: Minute: Second Millis: Micro: Nano"

testAppendDateStrFunc(char *aDateString)
{
    int yy, int mm, int dd, int hh, int mi, int ss;
    unsigned long t1;
    void *valueArray[5];
    sscanf(aDateString, "%d-%d-%d %d:%d:%d %d:%d:%d",
        &yy, &mm, &dd, &hh, &mi, &ss, &mmm, &uuu, &nnn);
    sTm.tm_year = yy - 1900;
    sTm.tm_mon = mm - 1;
    sTm.tm_mday = dd;
    sTm.tm_hour = hh;
    sTm.tm_min = mi;
    sTm.tm_sec = ss;
    t1 = mktime(&sTm);
    t1 = t1 * 1000000000L;
    t1 = t1 + (mmm*1000000L) + (uuu*1000) + nnn;

    valueArray[4] = &t1;
    SQLAppendData(aStmt, valueArray);
}
```

SQLAppendDataByTime(deprecated)

```
SQLRETURN  SQLAppendDataByTime(SQLHSTMT StatementHandle, SQLBIGINT aTime, void *aData[]);
```

This function is a function to input data for the corresponding channel, and the value of _arrival_time stored in the DB can be set to a specific time value instead of the current time.
For example, you want to enter the date in the log file a month ago as the date.

- aTime is a time value set to _arrival_time.
- aData is an array containing pointers to the data to be input.
- The number of arrays must match the number of columns held by the table specified at Open.

For the rest, refer to the SQLAppendData () function.

```
// For tables with four column information (short (16), int (32), long (64), varchar)

testAppendFuncWithTime()
{
   long long sTime = 1;
   short val1 = 0;
   int   val2 = 1;
   long long  val3 = 2;
   char *val4 = "my string";
   void *valueArray[4];

   valueArray[0] = (void *)&val1;
   valueArray[1] = (void *)&val2;
   valueArray[2] = (void *)&val3;
   valueArray[3] = (void *)val4;

   SQLAppendDataByTime(aStmt, sTime, valueArray);
}
```

SQLAppendDataV2

```
SQLRETURN  SQLAppendDataV2(SQLHSTMT StatementHandle, SQL_APPEND_PARAM *aData);
```

This function is a newly introduced Append function since Machbase 2.0. It is a convenient function that improves the input method inconvenient in existing functions.
In the case of TEXT and BINARY type introduced in 2.0 especially, input is possible only in SQLAppendDataV2 () function.

- Can input NULL for each type
- Can input string length when inputting VARCHAR
- Can input binary and string data when inputting IPv4 or IPv6
- Can specify data length for TEXT, BINARY type

The function arguments are structured as follows.

- aData is a pointer to an array of arguments called SQL_APPEND_PARAM. The number of this array must match the number of columns held by the table specified at Open.
- The return value can be SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, or SQL_ERROR. In particular, if SQL_SUCCESS_WITH_INFO is returned, there may be errors such as a lengthy input column being truncated, so check the result again.

Below is the definition of `SQL_APPEND_PARAM` that will actually be used in V2 , which is included in `machbase_sqlcli.h`.

```
typedef struct machAppendVarStruct
{
    unsigned int mLength;
    void *mData;
} machAppendVarStruct;

/* for IPv4, IPv6 as bin or string representation */
typedef struct machAppendIPSttypedef struct machbaseAppendVarStruct
{
    unsigned int mLength;
    void        *mData;
} machbaseAppendVarStruct;

/* for IPv4, IPv6 as bin or string representation */
typedef struct machbaseAppendIPStruct
{
    unsigned char   mLength; /* 0:null, 4:ipv4, 6:ipv6, 255:string representation */
    unsigned char   mAddr[16];
    char           *mAddrString;
} machbaseAppendIPStruct;

/* Date time*/
typedef struct machbaseAppendDateTimeStruct
{
    long long       mTime;
#if defined(SUPPORT_STRUCT_TM)
    struct tm       mTM;
#endif
    char           *mDateStr;
    char           *mFormatStr;
} machbaseAppendDateTimeStruct;

typedef union machbaseAppendParam
{
    short                       mShort;
    unsigned short              mUShort;
    int                         mInteger;
    unsigned int                mUInteger;
    long long                   mLong;
    unsigned long long          mULong;
    float                       mFloat;
    double                      mDouble;
    machbaseAppendIPStruct      mIP;
    machbaseAppendVarStruct     mVar;     /* for all varying type */
    machbaseAppendVarStruct     mVarchar; /* alias */
    machbaseAppendVarStruct     mText;    /* alias */
    machbaseAppendVarStruct     mBinary;  /* binary */
    machbaseAppendVarStruct     mBlob;    /* reserved alias */
    machbaseAppendVarStruct     mClob;    /* reserved alias */
    machbaseAppendDateTimeStruct mDateTime;
} machbaseAppendParam;
```

```
#define SQL_APPEND_PARAM machbaseAppendParam
```

As you can see from the above, there is a structure in which a shared structure `machbaseAppendParam` which internally contains one argument. The length and value for the data and string can be explicitly entered for each data type. Examples of actual use are as follows.

Fixed-Length Numeric Type Input

Fixed-length numeric types are short, ushort, integer, uinteger, long, ulong, float, and double. This type can be entered by directly assigning a value to the structure member of SQL_APPEND_PARAM.

Database Type	NULL Macro	SQL_APPEND_PARAM Member
SHORT	SQL_APPEND_SHORT_NULL	mShort
USHORT	SQL_APPEND_USHORT_NULL	mUShort
INTEGER	SQL_APPEND_INTEGER_NULL	mInteger
UINTEGER	SQL_APPEND_UINTEGER_NULL	mUInteger
LONG	SQL_APPEND_LONG_NULL	mLong
ULONG	SQL_APPEND_ULONG_NULL	mULong
FLOAT	SQL_APPEND_FLOAT_NULL	mFloat
DOUBLE	SQL_APPEND_DOUBLE_NULL	mDouble

The following is an example of entering actual values.

```
// Assume that the Table Schema consists of eight columns, SHORT, USHORT, INTEGER, UINTEGER, LONG, ULONG, FLOAT, ar

void testAppendExampleFunc()
{
    SQL_APPEND_PARAM sParam[8];

    /* fixed column */
    sParam[0].mShort = SQL_APPEND_SHORT_NULL;
    sParam[1].mUShort = SQL_APPEND_USHORT_NULL;
    sParam[2].mInteger = SQL_APPEND_INTEGER_NULL;
    sParam[3].mUInteger = SQL_APPEND_UINTEGER_NULL;
    sParam[4].mLong = SQL_APPEND_LONG_NULL;
    sParam[5].mULong = SQL_APPEND_ULONG_NULL;
    sParam[6].mFloat = SQL_APPEND_FLOAT_NULL;
    sParam[7].mDouble = SQL_APPEND_DOUBLE_NULL;

    SQLAppendDataV2(Stmt, sParam);

    /* FIXED COLUMN Value */
    sParam[0].mShort = 2;
    sParam[1].mUShort = 3;
    sParam[2].mInteger = 4;
    sParam[3].mUInteger = 5;
    sParam[4].mLong = 6;
    sParam[5].mULong = 7;
    sParam[6].mFloat = 8.4;
    sParam[7].mDouble = 10.9;

    SQLAppendDataV2(Stmt, sParam);
}
```

Date Type Input

Below is an example of inputting data of DATETIME type. Several macros are available for convenience.

Performs operations on the mDateTime member in SQL_APPEND_PARAM. The following macro can specify a date by setting a 64-bit integer value called mTime in the mDateTime structure.

```
typedef struct machbaseAppendDateTimeStruct
{
```

```
    long long       mTime;
#if defined(SUPPORT_STRUCT_TM)
    struct tm       mTM;
#endif
    char           *mDateStr;
    char           *mFormatStr;
} machbaseAppendDateTimeStruct;
```

Macro	Description
SQL_APPEND_DATETIME_NOW	Enters the current client time.
SQL_APPEND_DATETIME_STRUCT_TM	Sets a value to mTM, the struct tm structure of mDateTime, and inputs the value to the database.
SQL_APPEND_DATETIME_STRING	Sets a value for the string type of mDateTime and enters it into the database. mDateStr: real date string value assigned mFormatStr: format string assignment for date string
SQL_APPEND_DATETIME_NULL	Enters the value of the date column as NULL.
Any 64-bit Value	This value is entered as the actual datetime. This value represents an integer value in nanoseconds since January 1, 1970. For example, if this value is 1 billion (1,000,000,000), it represents 0: 1: 1 on January 1, 1970. (GMT)

```
// Assume that the table schema consists of eight columns, SHORT, USHORT, INTEGER, UINTEGER, LONG, ULONG, FLOAT, ar

void testAppendDateTimeFunc()
{
    SQL_mach_PARAM sParam[1];
    /* NULL Insert */
    sParam[0].mDateTime.mTime   = SQL_APPEND_DATETIME_NULL;
    SQLAppendDataV2(Stmt, sParam);

    /* Current Time  */
    sParam[0].mDateTime.mTime      = SQL_APPEND_DATETIME_NOW;
    SQLAppendDataV2(Stmt, sParam);

    /* nano second since 1970/01/01 */
    sParam[0].mDateTime.mTime      = 1234;
    SQLAppendDataV2(Stmt, sParam);

    /*  String format time  */
    sParam[0].mDateTime.mTime      = SQL_APPEND_DATETIME_STRING;
    sParam[0].mDateTime.mDateStr   = "23/May/2014:17:41:28";
    sParam[0].mDateTime.mFormatStr = "DD/MON/YYYY:HH24:MI:SS";
    SQLAppendDataV2(Stmt, sParam);

    /*  struct tm based time  */
    sParam[0].mDateTime.mTime      = SQL_APPEND_DATETIME_STRUCT_TM;
    sParam[0].mDateTime.mTM.tm_year = 2000 - 1900;
    sParam[0].mDateTime.mTM.tm_mon  =  11;
    sParam[0].mDateTime.mTM.tm_mday  = 31;
    SQLAppendDataV2(Stmt, sParam);
}
```

Internet Address Type Input

The following is an example of inputting IPv4 and IPv6 type data. There are also several macros available for your convenience. Performs operations on the mLength member in SQL_APPEND_PARAM.

```
/* for IPv4, IPv6 as bin or string representation */
typedef struct machbaseAppendIPStruct
{
    unsigned char   mLength; /* 0:null, 4:ipv4, 6:ipv6, 255:string representation */
    unsigned char   mAddr[16];
    char           *mAddrString;
} machbaseAppendIPStruct;
```

Macro (Set to mLength)	Description
SQL_APPEND_IP_NULL	Enters a NULL value in the corresponding column
SQL_APPEND_IP_IPV4	mAddr has IPv4
SQL_APPEND_IP_IPV6	mAddr has IPv6
SQL_APPEND_IP_STRING	mAddrString has an address string.

The following is an example of entering actual values for each case.

```
void testAppendIPFunc()
{
    SQL_APPEND_PARAM sParam[1];
    /* NULL */
    sParam[0].mIP.mLength  = SQL_APPEND_IP_NULL;
    SQLAppendDataV2(Stmt, sParam);

    /* Direct array access */
    sParam[0].mIP.mLength  = SQL_APPEND_IP_IPV4;
    sParam[0].mIP.mAddr[0] = 127;
    sParam[0].mIP.mAddr[1] = 0;
    sParam[0].mIP.mAddr[2] = 0;
    sParam[0].mIP.mAddr[3] = 1;
    SQLAppendDataV2(Stmt, sParam);

    /* IPv4 from binary */
    sParam[0].mIP.mLength  = SQL_APPEND_IP_IPV4;
    *(in_addr_t *)(sParam[0].mIP.mAddr) = inet_addr("192.168.0.1");
    SQLAppendDataV2(Stmt, sParam);

    /* IPv4 : ipv4 from string */
    sParam[0].mIP.mLength     = SQL_APPEND_IP_STRING;
    sParam[0].mIP.mAddrString = "203.212.222.111";
    SQLAppendDataV2(Stmt, sParam);

    /* IPv4 : ipv4 from invalid string */
    sParam[0].mIP.mLength     = SQL_APPEND_IP_STRING;
    sParam[0].mIP.mAddrString = "ip address is not valid";
    SQLAppendDataV2(Stmt, sParam);                          // invalid IP value

    /* IPv6 : ipv6 from binary bytes */
    sParam[0].mIP.mLength  = SQL_APPEND_IP_IPV6;
    sParam[0].mIP.mAddr[0]  = 127;
    sParam[0].mIP.mAddr[1]  = 127;
    sParam[0].mIP.mAddr[2]  = 127;
    sParam[0].mIP.mAddr[3]  = 127;
    sParam[0].mIP.mAddr[4]  = 127;
    sParam[0].mIP.mAddr[5]  = 127;
    sParam[0].mIP.mAddr[6]  = 127;
    sParam[0].mIP.mAddr[7]  = 127;
    sParam[0].mIP.mAddr[8]  = 127;
    sParam[0].mIP.mAddr[9]  = 127;
    sParam[0].mIP.mAddr[10] = 127;
    sParam[0].mIP.mAddr[11] = 127;
    sParam[0].mIP.mAddr[12] = 127;
    sParam[0].mIP.mAddr[13] = 127;
    sParam[0].mIP.mAddr[14] = 127;
    sParam[0].mIP.mAddr[15] = 127;
    SQLAppendDataV2(Stmt, sParam);

    sParam[0].mIP.mLength     = SQL_APPEND_IP_STRING;
    sParam[0].mIP.mAddrString = "::127.0.0.1";
    SQLAppendDataV2(Stmt, sParam);

    sParam[0].mIP.mLength     = SQL_APPEND_IP_STRING;
    sParam[0].mIP.mAddrString = "FFFF:FFFF:1111:2222:3333:4444:7733:2123";
    SQLAppendDataV2(Stmt, sParam);
```

```
}
```

ⓘ

Variable Data Types (Character and Binary Data) Input

Variable data types include VARCHAR and TEXT, and BLOB and CLOB. In existing functions, only VARCHAR was supported, and there was no way for the user to enter the length of the string. For that reason, we had to get the length through the strlen () function each time, but from function V2, the user can directly specify the length for the variable data type. Thus, if the user knows the length in advance, data can be input more quickly. Internally, the variable data type is a structure. However, for convenience of development, members are created separately for each data type.

```
typedef struct machAppendVarStruct
{
    unsigned int mLength;
    void *mData;
} machAppendVarStruct;
```

When inputting a variable data type, set the length of the data to mLength and set the primitive data pointer to mData. If mLength is greater than the defined schema, it is automatically truncated. At this time, SQLAppendDataV2 () returns **SQL_SUCCESS_WITH_INFO** and also fills the internal structure with a related warning message. To see this warning message, use SQLError () function.

Database Type	NULL Macro	SQL_APPEND_PARAM Member (mVar is acceptable)
VARCHAR	SQL_APPEND_VARCHAR_NULL	mVarchar
TEXT	SQL_APPEND_TEXT_NULL	mText
BINARY	SQL_APPEND_BINARY_NULL	mBinary
BLOB	SQL_APPEND_BLOB_NULL	mBlob
CLOB	SQL_APPEND_CLOB_NULL	mClob

The following is an example of entering actual values for each environment. Assumes that there is one VARCHAR column.

```
CREATE TABLE ttt (name VARCHAR(10));
```

```
void testAppendVarcharFunc()
{
    SQL_mach_PARAM sParam[1];

    /*  VARCHAR : NULL */
    sParam[0].mVarchar.mLength = SQL_APPEND_VARCHAR_NULL
    SQLAppendDataV2(Stmt, sParam); /* OK */

    /*  VARCHAR : string */
    strcpy(sVarchar, "MY VARCHAR");
    sParam[0].mVarchar.mLength = strlen(sVarchar);
    sParam[0].mVarchar.mData   = sVarchar;
    SQLAppendDataV2(Stmt, sParam); /* OK */

    /*  VARCHAR : Truncation! */
    strcpy(sVarchar, "MY VARCHAR9"); /* Truncation! */
    sParam[0].mVarchar.mLength = strlen(sVarchar);
    sParam[0].mVarchar.mData   = sVarchar;
    SQLAppendDataV2(Stmt, sParam);  /* SQL_SUCCESS_WITH_INFO */
}
```

The following is an example of input for the Text type.

```
CREATE TABLE ttt (doc TEXT);
```

```
void testAppendFunc()
{
    SQL_mach_PARAM sParam[1];

    /*  VARCHAR : NULL */
    sParam[0].mText.mLength = SQL_APPEND_TEXT_NULL
    SQLAppendDataV2(Stmt, sParam); /* OK */

    /*  VARCHAR : string */
    strcpy(sText, "This is the sample document for tutorial.");
    sParam[0].mVar.mLength = strlen(sText);
    sParam[0].mVar.mData   = sText;
    SQLAppendDataV2(Stmt, sParam); /* OK */
}
```

SQLAppendDataByTimeV2

```
SQLRETURN  SQLAppendDataByTimeV2(SQLHSTMT StatementHandle, SQLBIGINT aTime, SQL_APPEND_PARAM  *aData);
```

This function is a function to input data for the corresponding channel, and the value of _arrival_time stored in the DB can be set to a specific time value instead of the current time. For example, you want to enter the date in the log file a month ago as the date.

- aTime is the time value to be set to _arrival_time. *You must enter the nano second value from January 1, 1970 to the present. Also, input values must be sorted in order from the past to the present.*
- aData is an array containing pointers to the data to be input. The number of arrays must match the number of columns held by the table specified at Open.

For the rest, refer to the SQLAppendDataV2 () function.

SQLAppendFlush

```
SQLRETURN SQLAppendFlush(SQLHSTMT StatementHandle);
```

This function immediately sends the data accumulated in the current channel buffer to the Machbase server.

SQLAppendClose

```
SQLRETURN SQLAppendClose(SQLHSTMT   aStmtHandle,
                         SQLBIGINT* aSuccessCount,
                         SQLBIGINT* aFailureCount);
```

This function closes the currently open channel. If an unopened channel exists, an error occurs.

- aSuccessCount: The number of successful Append records.
- aFailureCount: The number of failed Append records.

SQLAppendSetErrorCallback

```
SQLRETURN SQLAppendSetErrorCallback(SQLHSTMT aStmtHandle, SQLAppendErrorCallback aFunc);
```

This function sets the callback function that is called when an error occurs during append. If you do not set this function, the client will ignore any errors that occur in the server.

- aStmtHandle: Specifies a Statement to check for errors.
- aFunc: Specifies the function pointer to call on Append failure.

The prototype for SQLAppendErrorCallback is:

```
typedef void (*SQLAppendErrorCallback)(SQLHSTMT aStmtHandle,
                                       SQLINTEGER aErrorCode,
                                       SQLPOINTER aErrorMessage,
                                           SQLLEN aErrorBufLen,
                                       SQLPOINTER aRowBuf,
                                           SQLLEN aRowBufLen);
```

- aStatementHandle: the statement handle that generated the error
- aErrorCode: 32-bit error code that caused the error
- aErrorMessage: string for the error code
- aErrorBufLen: the length of aErrorMessage
- aRowBuf: a string containing the detailed description of the record that caused the error
- aRowBufLen: length of aRowBuf

Example of Using Error Callback (dumpError)

```
void dumpError(SQLHSTMT    aStmtHandle,
               SQLINTEGER  aErrorCode,
               SQLPOINTER  aErrorMessage,
               SQLLEN      aErrorBufLen,
               SQLPOINTER  aRowBuf,
               SQLLEN      aRowBufLen)
{
    char       sErrMsg[1024] = {0, };
    char       sRowMsg[32 * 1024] = {0, };

    if (aErrorMessage != NULL)
    {
        strncpy(sErrMsg, (char *)aErrorMessage, aErrorBufLen);
    }

    if (aRowBuf != NULL)
    {
        strncpy(sRowMsg, (char *)aRowBuf, aRowBufLen);
    }

    fprintf(stdout, "Append Error : [%d][%s]\n[%s]\n\n", aErrorCode, sErrMsg, sRowMsg);
}


......

    if( SQLAppendOpen(m_IStmt, TableName, aErrorCheckCount) != SQL_SUCCESS )
    {
        fprintf(stdout, "SQLAppendOpen error\n");
        exit(-1);
    }
    // Set callback.
    assert(SQLAppendSetErrorCallback(m_IStmt, dumpError) == SQL_SUCCESS);

    doAppend(sMaxAppend);

    if( SQLAppendClose(m_IStmt, &sSuccessCount, &sFailureCount) != SQL_SUCCESS )
    {
        fprintf(stdout, "SQLAppendClose error\n");
        exit(-1);
    }
}
```

SQLSetConnectAppendFlush

```
SQLRETURN SQL_API SQLSetConnectAppendFlush(SQLHDBC hdbc, SQLINTEGER option)
```

The data input by Append is written to the communication buffer and is sent to the server when the user calls the SQLAppendFlush function in the waiting state or the communication buffer becomes full. You can use this function if you want the user to send data by append to the server at regular intervals even if the buffer is not full. This function computes the difference between the last transmitted time and the current time every 100ms, and transfers the contents of the communication buffer to the server when the specified time (1 second if not set) has passed.

The parameters are:

- hdbc: DB connection handle.
- If option: 0, auto flush is off; otherwise, auto flush is on.

Executing on an unconnected hdbc will result in an error.

SQLSetStmtAppendInterval

```
SQLRETURN SQL_API SQLSetStmtAppendInterval(SQLHSTMT hstmt, SQLINTEGER fValue)
```

Uses SQLSetConnectAppendFlush to turn off automatic flushing or flushing for a particular statement when you turn on flushing on a time unit.

The parameters are:

- hstmt: This is the statement handle that you want to adjust the flush interval.
- fValue: The value to which you want to adjust the flush interval. **If 0, flush is not performed and the unit is ms**. Set to a multiple of 100 since the thread that determines whether to flush every 100ms is executed. It does not automatically flush at exactly the right time. **1000 is the default value** .

Execution of this function will succeed even if time-based flush is not running.

Error Check and Description

This is a description of the code and how to check for errors when using the Append related functions. If the return value in the CLI function is not SQL_SUCCESS, you can check the error message using the following code.

```
SQLINTEGER errNo;
int msgLength;
char sqlState[6];
char errMsg[1024];

if (SQL_SUCCESS == SQLError ( env, con, stmt, (SQLCHAR *)sqlState, &errNo,
                              (SQLCHAR *)errMsg, 1024, &msgLength ))
{
    // (Specify error code value as 5-digit number.)
    printf("ERROR-%05d: %s\n", errNo, errMsg);
}
```

The error message returned from the Append related function is as follows.

Item	Message	Description
SQLAppendOpen	statement is already opened.	Occurs when SQLAppendOpen is executed in duplicate.
	Failed to close stream protocol.	Stream protocol termination failed.
	Failed to read protocol.	A network read error occurred.
	cannot read column meta.	Invalid column meta information structure
	cannot allocate memory.	An internal buffer memory allocation error occurred.
	cannot allocate compress memory.	Compressed buffer memory allocation error occurred.
	invalid return after reading column meta.	Return value has an error.
SQLAppendData	statement is not opened.	Called AppendData without AppendOpen.
	column() truncated :	Occurs when you enter data that is larger than the size specified in the varchar type column.
	Failed to add binary.	Write error in communication buffer occurred.
SQLAppendClose	statement is not opened.	Not in AppendOpen state.
	Failed to close stream protocol.	Stream protocol termination failed.
	Failed to close buffer protocol.	Buffer protocol termination failed.
	cannot read column meta.	The column meta information structure is incorrect.
	invalid return after reading column meta.	Return value has an error.
SQLAppendFlush	statement is not opened.	Not in AppendOpen state
	Failed to close stream protocol.	A network write error occurred.
SQLSetErrorCallback	statement is not opened.	Not in AppendOpen state.
	Protocol Error (not APPEND_DATA_PROTOCOL)	Communication buffer read result is not APPEND_DATA_PROTOCOL value.
SQLAppendDataV2	Invalid date format or date string.	Occurs when the datetime type is wrong.
	statement is not opened.	Not in AppendOpen state
	column() truncated :	This occurs when you enter data that is larger than the size specified in the binary type column.

Item	Message	Description
	column() truncated :	Occurs when you enter data that is larger than the size specified in the varchar and text type column.
	Failed to add stream.	Write error in communication buffer occurred.
	IP address length is invalid.	The mLength value of the IPv4, IPv6 type structure is specified incorrectly.
	IP string is invalid.	Not in IPv4 or IPv6 format.
	Unknown data type has been specified.	Not the data type used by Machbase.

Column wise parameter binding

The SQLAppend function, which is used to enter a large amount of data into Machbase quickly, can be used only when entering a log / tag table, and the SQLAppend function cannot be used to perform a bulk update on a lookup or volatile table.

For this purpose, Machbase 5.5 and later versions support column wise parameter binding. (Row wise format parameter binding is not yet supported.)

Set SQL_ATTR_PARAM_BIND_TYPE in the argument Attribute of the function SQLSetStmtAttr () and SQL_PARAM_BIND_BY_COLUMN in the parameter param.

For each column to bind, set the parameter to an array and the indicator variable to an array. Then call SQLBindParameter () with this parameter.

The figure below shows how columnar binding works for each parameter array.

Column A (parameter A)		Column B (parameter B)		Column C (parameter C)	
Value_Array	Indicator/ length array	Value_Array	Indicator/ length array	Value_Array	Indicator/ length array

The following example inserts a large amount of data using column wise parameter binding.

```
#define DESC_LEN 51
#define ARRAY_SIZE 10
SQLCHAR * Statement = "INSERT INTO Parts (PartID, Description, Price) VALUES (?, ?, ?)";

SQLUINTEGER PartIDArray[ARRAY_SIZE];
SQLCHAR DescArray[ARRAY_SIZE][DESC_LEN];
SQLREAL PriceArray[ARRAY_SIZE];
/* Bind variable array  */
SQLINTEGER PartIDIndArray[ARRAY_SIZE], DescLenOrIndArray[ARRAY_SIZE], PriceIndArray[ARRAY_SIZE];
SQLUSMALLINT i, ParamStatusArray[ARRAY_SIZE];
SQLUINTEGER ParamsProcessed;

// Set the SQL_ATTR_PARAM_BIND_TYPE statement attribute to use
// column-wise binding.
SQLSetStmtAttr(hstmt, SQL_ATTR_PARAM_BIND_TYPE, SQL_PARAM_BIND_BY_COLUMN, 0);
// Specify the number of elements in each parameter array.
SQLSetStmtAttr(hstmt, SQL_ATTR_PARAMSET_SIZE, ARRAY_SIZE, 0);
// Specify an array in which to return the status of each set of
// parameters.
SQLSetStmtAttr(hstmt, SQL_ATTR_PARAM_STATUS_PTR, ParamStatusArray, 0);
// Specify an SQLUINTEGER value in which to return the number of sets of
// parameters processed.
SQLSetStmtAttr(hstmt, SQL_ATTR_PARAMS_PROCESSED_PTR, &ParamsProcessed, 0);
// Bind the parameters in column-wise fashion.
SQLBindParameter(hstmt, 1, SQL_PARAM_INPUT, SQL_C_ULONG, SQL_INTEGER, 5, 0,
    PartIDArray, 0, PartIDIndArray);
SQLBindParameter(hstmt, 2, SQL_PARAM_INPUT, SQL_C_CHAR, SQL_CHAR, DESC_LEN - 1, 0,
    DescArray, DESC_LEN, DescLenOrIndArray);
SQLBindParameter(hstmt, 3, SQL_PARAM_INPUT, SQL_C_FLOAT, SQL_REAL, 7, 0,
    PriceArray, 0, PriceIndArray);
```

CLI/ODBC Examples

Application Development

Check CLI Installation

If the following files are included in include and lib of the directory where Machbase is installed, the environment in which the application can be developed is complete.

```
Mach@localhost:~/machbase_home$ ls -l include lib install/
include:
total 176
-rwxrwxr-x 1 mach mach 31449 Jun 18 19:26 machbase_sqlcli.h

install/:
total 12
-rw-rw-r-- 1 mach mach 1667 Jun 18 19:26 machbase_env.mk

lib:
total 16196
-rw-rw-r-- 1 mach mach  78603 Jun 18 19:26 machbase.jar
-rw-rw-r-- 1 mach mach 964290 Jun 18 19:26 libmachbasecli.a
```

Index

Makefile Creation Guide

```
mach@localhost:~/machbase_home$ cd sample/
mach@localhost:~/machbase_home/sample$ cd cli/
mach@localhost:~/machbase_home/sample/cli$ ls
Makefile sample1_connect.c
```

If the Machbase package was installed, the sample program will be installed in the following path.

```
include $(MACHBASE_HOME)/install/machbase_env.mk
INCLUDES += $(LIBDIR_OPT)/$(MACHBASE_HOME)/include

all : sample1_connect

sample1_connect : sample1_connect.o
$(LD_CC) $(LD_FLAGS) $(LD_OUT_OPT)$@ $< $(LIB_OPT)machbasecli$(LIB_AFT) $(LIBDIR_OPT)$(MACHBASE_HOME)/lib $(LD_LIBS

sample1_connect.o : sample1_connect.c
$(COMPILE.cc) $(CC_FLAGS) $(INCLUDES) $(CC_OUT_OPT)$@ $<

clean :
rm -f sample1_connect
```

Compile and Link

Executing the following for a given sample creates an executable file.

```
mach@localhost:~/machbase_home/sample/cli$ make
gcc -c -g -W -Wall -rdynamic -O3 -finline-functions -fno-omit-frame-pointer -fno-strict-aliasing -m64 -mtune=k8 -g
gcc -m64 -mtune=k8 -L/home/machbase/machbase_home/lib -osample1_connect sample1_connect.o -lmachbasecli -L/home/mac
mach@localhost:~/machbase_home/sample/cli$ ls -al
total 1196
drwxrwxr-x 2 mach mach 4096 Jun 18 20:15 .
drwxrwxr-x 4 mach mach 4096 Jun 18 19:26 ..
-rw-rw-r-- 1 mach mach 483 Jun 18 19:26 Makefile
-rwxrwxr-x 1 mach mach 1196943 Jun 18 20:15 sample1_connect
-rw-rw-r-- 1 mach mach 549 Jun 18 19:26 sample1_connect.c
```

```
-rw-rw-r-- 1 mach mach 8168 Jun 18 20:15 sample1_connect.o
```

You can write your application as necessary by modifying the sample Makefile above

Sample Program

Connection Example

We will create an example program to connect using the CLI.
The sample file name is sample1_connect.c.
MACHBASE_PORT_NO must be the same as the PORT_NO value in the $MACHBASE_HOME/conf/machbase.conf file.

› sample1_connect.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <machbase_sqlcli.h>

#define MACHBASE_PORT_NO 5656

SQLHENV gEnv;
SQLHDBC gCon;
SQLHSTMT gStmt;

void connectDB()
{
    char connStr[1024];
    SQLINTEGER errNo;
    SQLSMALLINT msgLength;
    SQLCHAR errMsg[1024];

    if (SQL_ERROR == SQLAllocEnv(&gEnv)) {
        printf("SQLAllocEnv error!!\n");
        exit(1);
    }
    if (SQL_ERROR == SQLAllocConnect(gEnv, &gCon)) {
        printf("SQLAllocConnect error!!\n");
        SQLFreeEnv(gEnv);
        exit(1);
    }
    sprintf(connStr,"SERVER=127.0.0.1;UID=SYS;PWD=MANAGER;CONNTYPE=1;PORT_NO=%d", MACHBASE_PORT_NO);
    if (SQL_ERROR == SQLDriverConnect( gCon, NULL,
                                       (SQLCHAR *)connStr,
                                       SQL_NTS,
                                       NULL, 0, NULL,
                                       SQL_DRIVER_NOPROMPT ))
    {
        printf("connection error\n");
        if (SQL_SUCCESS == SQLError ( gEnv, gCon, NULL, NULL, &errNo,
                                      errMsg, 1024, &msgLength ))
        {
            printf("mach-%d : %s\n", errNo, errMsg);
        }
        SQLFreeEnv(gEnv);
        exit(1);
    }
    printf("connected ... \n");
}

void disconnectDB()
{
    SQLINTEGER errNo;
    SQLSMALLINT msgLength;
    SQLCHAR errMsg[1024];
```

```
    if (SQL_ERROR == SQLDisconnect(gCon))
    {
        printf("disconnect error\n");

        if( SQL_SUCCESS == SQLError( gEnv, gCon, NULL, NULL, &errNo,
                                     errMsg, 1024, &msgLength ))
        {
            printf("mach-%d : %s\n", errNo, errMsg);
        }
    }

    SQLFreeConnect(gCon);
    SQLFreeEnv(gEnv);
}

int main()
{
    connectDB();
    disconnectDB();
    return 0;
}
```

If you register sample1_connect.c in Makefile, compile and run it, it will appear as follows.

```
[mach@localhost cli]$ make

[mach@localhost cli]$ ./sample1_connect
connected ...
```

Data Input and Output Example

In the example source below, we created a table using the CREATE TABLE statement, arbitrarily create simple data values, input data using the INSERT statement, and output the data using the SELECT statement. You will be able to see how to configure each type when entering and checking values directly.
The sample file name is sample2_insert.c.

› sampe2_insert.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <machbase_sqlcli.h>

#define MACHBASE_PORT_NO 5656

SQLHENV gEnv;
SQLHDBC gCon;
SQLHSTMT gStmt;
SQLCHAR gErrorState[6];

void connectDB()
{
    char connStr[1024];
    SQLINTEGER errNo;
    SQLSMALLINT msgLength;
    SQLCHAR errMsg[1024];
    if (SQL_ERROR == SQLAllocEnv(&gEnv)) {
        printf("SQLAllocEnv error!!\n");
        exit(1);
    }
    if (SQL_ERROR == SQLAllocConnect(gEnv, &gCon)) {
        printf("SQLAllocConnect error!!\n");
        SQLFreeEnv(gEnv);
        exit(1);
    }
    sprintf(connStr,"SERVER=127.0.0.1;UID=SYS;PWD=MANAGER;CONNTYPE=1;PORT_NO=%d", MACHBASE_PORT_NO);
```

```
    if (SQL_ERROR == SQLDriverConnect( gCon, NULL,
                                       (SQLCHAR *)connStr,
                                       SQL_NTS,
                                       NULL, 0, NULL,
                                       SQL_DRIVER_NOPROMPT ))
    {
        printf("connection error\n");
        if (SQL_SUCCESS == SQLError ( gEnv, gCon, NULL, NULL, &errNo,
                                      errMsg, 1024, &msgLength ))
        {
            printf(" mach-%d : %s\n", errNo, errMsg);
        }
        SQLFreeEnv(gEnv);
        exit(1);
    }
    printf("connected ... \n");
}

void disconnectDB()
{
    SQLINTEGER errNo;
    SQLSMALLINT msgLength;
    SQLCHAR errMsg[1024];
    if (SQL_ERROR == SQLDisconnect(gCon)) {
        printf("disconnect error\n");
        if( SQL_SUCCESS == SQLError( gEnv, gCon, NULL, NULL, &errNo,
                                     errMsg, 1024, &msgLength ))
        {
            printf(" mach-%d : %s\n", errNo, errMsg);
        }
    }
    SQLFreeConnect(gCon);
    SQLFreeEnv(gEnv);
}

void outError(const char *aMsg, SQLHSTMT stmt)
{
    SQLINTEGER errNo;
    SQLSMALLINT msgLength;
    SQLCHAR errMsg[1024];
    printf("ERROR : (%s)\n", aMsg);
    if (SQL_SUCCESS == SQLError( gEnv, gCon, stmt, NULL, &errNo,
                                 errMsg, 1024, &msgLength ))
    {
        printf(" mach-%d : %s\n", errNo, errMsg);
    }
    exit(-1);
}

void executeDirectSQL(const char *aSQL, int aErrIgnore)
{
    SQLHSTMT stmt;
    if (SQLAllocStmt(gCon, &stmt) == SQL_ERROR)
    {
        if (aErrIgnore != 0) return;
        outError("AllocStmt error", stmt);
    }
    if (SQLExecDirect(stmt, (SQLCHAR *)aSQL, SQL_NTS) == SQL_ERROR)
    {
        if (aErrIgnore != 0) return;
        printf("sql_exec_direct error[%s] \n", aSQL);
        outError("sql_exec_direct error", stmt);
    }
    if (SQL_ERROR == SQLFreeStmt(stmt, SQL_DROP))
    {
        if (aErrIgnore != 0) return;
        outError("FreeStmt Error", stmt);
    }
}
```

```
void prepareExecuteSQL(const char *aSQL)
{
    SQLHSTMT stmt;
    if (SQLAllocStmt(gCon, &stmt) == SQL_ERROR)
    {
        outError("AllocStmt error", stmt);
    }
    if (SQLPrepare(stmt, (SQLCHAR *)aSQL, SQL_NTS) == SQL_ERROR)
    {
        printf("Prepare error[%s]\n", aSQL);
        outError("Prepare error", stmt);
    }
    if (SQLExecute(stmt) == SQL_ERROR)
    {
        outError("prepared execute error", stmt);
    }
    if (SQL_ERROR == SQLFreeStmt(stmt, SQL_DROP))
    {
        outError("FreeStmt Error", stmt);
    }
}

void createTable()
{
    executeDirectSQL("DROP TABLE CLI_SAMPLE1", 1);
    executeDirectSQL("CREATE TABLE CLI_SAMPLE1(seq short, score integer, total long, percentage float, ratio dou
}

void selectTable()
{
    SQLHSTMT stmt;
    const char *aSQL = "SELECT seq, score, total, percentage, ratio, id, srcip, dstip, reg_date, textlog, image
    int i=0;
    SQLLEN Len = 0;
    short seq;
    int score;
    long total;
    float percentage;
    double ratio;
    char id [11];
    char srcip[16];
    char dstip[40];
    SQL_TIMESTAMP_STRUCT regdate;
    char log [1024];
    char image[1024];
    if (SQLAllocStmt(gCon, &stmt) == SQL_ERROR) {
        outError("AllocStmt Error", stmt);
    }
    if (SQLPrepare(stmt, (SQLCHAR *)aSQL, SQL_NTS) == SQL_ERROR) {
        printf("Prepare error[%s] \n", aSQL);
        outError("Prepare error", stmt);
    }
    if (SQLExecute(stmt) == SQL_ERROR) {
        outError("prepared execute error", stmt);
    }
    SQLBindCol(stmt, 1, SQL_C_SHORT, &seq, 0, &Len);
    SQLBindCol(stmt, 2, SQL_C_LONG, &score, 0, &Len);
    SQLBindCol(stmt, 3, SQL_C_BIGINT, &total, 0, &Len);
    SQLBindCol(stmt, 4, SQL_C_FLOAT, &percentage, 0, &Len);
    SQLBindCol(stmt, 5, SQL_C_DOUBLE, &ratio, 0, &Len);
    SQLBindCol(stmt, 6, SQL_C_CHAR, id, sizeof(id), &Len);
    SQLBindCol(stmt, 7, SQL_C_CHAR, srcip, sizeof(srcip), &Len);
    SQLBindCol(stmt, 8, SQL_C_CHAR, dstip, sizeof(dstip), &Len);
    SQLBindCol(stmt, 9, SQL_C_TYPE_TIMESTAMP, &regdate, 0, &Len);
    SQLBindCol(stmt, 10, SQL_C_CHAR, log, sizeof(log), &Len);
    SQLBindCol(stmt, 11, SQL_C_CHAR, image, sizeof(image), &Len);
    while (SQLFetch(stmt) == SQL_SUCCESS)
    {
        printf("===== %d ========\n", i++);
        printf("seq = %d", seq);
```

```
        printf(", score = %d", score);
        printf(", total = %ld", total);
        printf(", percentage = %.2f", percentage);
        printf(", ratio = %g", ratio);
        printf(", id = %s", id);
        printf(", srcip = %s", srcip);
        printf(", dstip = %s", dstip);
        printf(", regdate = %d-%02d-%02d %02d:%02d:%02d",
               regdate.year, regdate.month, regdate.day,
               regdate.hour, regdate.minute, regdate.second);
        printf(", log = %s", log);
        printf(", image = %s\n", image);
    }
    if (SQL_ERROR == SQLFreeStmt(stmt, SQL_DROP))
    {
        outError("FreeStmt eror", stmt);
    }
}

void directInsert()
{
    int i;
    char query[2 * 1024];
    short seq;
    int score;
    long total;
    float percentage;
    double ratio;
    char id [11];
    char srcip [16];
    char dstip [40];
    char reg_date [40];
    char log [1024];
    char image [1024];
    for(i=1; i<10; i++)
    {
        seq = i;
        score = i+i;
        total = (seq + score) * 10000;
        percentage = (float)score/total;
        ratio = (double)seq/total;
        sprintf(id, "id-%d", i);
        sprintf(srcip, "192.168.0.%d", i);
        sprintf(dstip, "2001:0DB8:0000:0000:0000:0000:1428:%04d", i);
        sprintf(reg_date, "2015-03-31 15:26:%02d", i);
        sprintf(log, "text log-%d", i);
        sprintf(image, "binary image-%d", i);
        memset(query, 0x00, sizeof(query));
        sprintf(query, "INSERT INTO CLI_SAMPLE1 VALUES(%d, %d, %ld, %f, %f, '%s', '%s', '%s',TO_DATE('%s','YYYY-
                seq, score, total, percentage, ratio, id, srcip, dstip, reg_date, log, image);
        prepareExecuteSQL(query);
        printf("%d record inserted\n", i);
    }
}

int main()
{
    connectDB();
    createTable();
    directInsert();
    selectTable();
    disconnectDB();
    return 0;
}
```

If you register sample2_insert.c in the Makefile, compile and run it, it will appear as follows.

```
[mach@localhost cli]$ make
```

```
[mach@localhost cli]$ ./sample2_insert

connected ...
1 record inserted
2 record inserted
3 record inserted
4 record inserted
5 record inserted
6 record inserted
7 record inserted
8 record inserted
9 record inserted
===== 0 ========
seq = 9, score = 18, total = 270000, percentage = 0.00, ratio = 3.3e-05, id = id-9, srcip = 192.168.0.9, dstip = 20
===== 1 ========
seq = 8, score = 16, total = 240000, percentage = 0.00, ratio = 3.3e-05, id = id-8, srcip = 192.168.0.8, dstip = 20
===== 2 ========
seq = 7, score = 14, total = 210000, percentage = 0.00, ratio = 3.3e-05, id = id-7, srcip = 192.168.0.7, dstip = 20
===== 3 ========
seq = 6, score = 12, total = 180000, percentage = 0.00, ratio = 3.3e-05, id = id-6, srcip = 192.168.0.6, dstip = 20
===== 4 ========
seq = 5, score = 10, total = 150000, percentage = 0.00, ratio = 3.3e-05, id = id-5, srcip = 192.168.0.5, dstip = 20
===== 5 ========
seq = 4, score = 8, total = 120000, percentage = 0.00, ratio = 3.3e-05, id = id-4, srcip = 192.168.0.4, dstip = 200
===== 6 ========
seq = 3, score = 6, total = 90000, percentage = 0.00, ratio = 3.3e-05, id = id-3, srcip = 192.168.0.3, dstip = 2001
===== 7 ========
seq = 2, score = 4, total = 60000, percentage = 0.00, ratio = 3.3e-05, id = id-2, srcip = 192.168.0.2, dstip = 2001
===== 8 ========
seq = 1, score = 2, total = 30000, percentage = 0.00, ratio = 3.3e-05, id = id-1, srcip = 192.168.0.1, dstip = 2001
```

Prepare Execute Example

Let's write an example program that binds and INSERTs data.
You can enter a value by binding data in Machbase. When you use this, you need to specify the types of data values clearly. In case of long string types, you must specify the length value.
The following example shows how to bind data for each type.
The file name is sample3_prepare.c.

› sample3_prepare.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <machbase_sqlcli.h>
#include <time.h>

#define MACHBASE_PORT_NO 5656

SQLHENV gEnv;
SQLHDBC gCon;
SQLHSTMT gStmt;
SQLCHAR gErrorState[6];

void connectDB()
{
    char sConnStr[1024];

    SQLINTEGER sErrorNo;
    SQLSMALLINT sMsgLength;
    SQLCHAR sErrorMsg[1024];

    if (SQL_ERROR == SQLAllocEnv(&gEnv)) {
        printf("SQLAllocEnv error!!\n");
        exit(1);
    }

    if (SQL_ERROR == SQLAllocConnect(gEnv, &gCon)) {
```

```
        printf("SQLAllocConnect error!!\n");
        SQLFreeEnv(gEnv);
        exit(1);
    }

    sprintf(sConnStr,"SERVER=127.0.0.1;UID=SYS;PWD=MANAGER;CONNTYPE=1;PORT_NO=%d", MACHBASE_PORT_NO);

    if (SQL_ERROR == SQLDriverConnect( gCon, NULL,
                                       (SQLCHAR *)sConnStr,
                                       SQL_NTS,
                                       NULL, 0, NULL,
                                       SQL_DRIVER_NOPROMPT ))
    {
        printf("connection error\n");

        if (SQL_SUCCESS == SQLError ( gEnv, gCon, NULL, NULL, &sErrorNo,
                                      sErrorMsg, 1024, &sMsgLength ))
        {
            printf(" mach-%d : %s\n", sErrorNo, sErrorMsg);
        }
        SQLFreeEnv(gEnv);
        exit(1);
    }

    printf("connected ... \n");

}

void disconnectDB()
{
    SQLINTEGER sErrorNo;
    SQLSMALLINT sMsgLength;
    SQLCHAR sErrorMsg[1024];

    if (SQL_ERROR == SQLDisconnect(gCon)) {
        printf("disconnect error\n");

        if( SQL_SUCCESS == SQLError( gEnv, gCon, NULL, NULL, &sErrorNo,
                                     sErrorMsg, 1024, &sMsgLength ))
        {
            printf(" mach-%d : %s\n", sErrorNo, sErrorMsg);
        }
    }

    SQLFreeConnect(gCon);
    SQLFreeEnv(gEnv);
}

void outError(const char *aMsg, SQLHSTMT aStmt)
{
    SQLINTEGER sErrorNo;
    SQLSMALLINT sMsgLength;
    SQLCHAR sErrorMsg[1024];

    printf("ERROR : (%s)\n", aMsg);

    if (SQL_SUCCESS == SQLError( gEnv, gCon, aStmt, NULL, &sErrorNo,
                                 sErrorMsg, 1024, &sMsgLength ))
    {
        printf(" mach-%d : %s\n", sErrorNo, sErrorMsg);
    }
    exit(-1);
}

void executeDirectSQL(const char *aSQL, int aErrIgnore)
{
    SQLHSTMT sStmt;

    if (SQLAllocStmt(gCon, &sStmt) == SQL_ERROR)
    {
```

```
        if (aErrIgnore != 0) return;
        outError("AllocStmt error", sStmt);
    }

    if (SQLExecDirect(sStmt, (SQLCHAR *)aSQL, SQL_NTS) == SQL_ERROR)
    {
        if (aErrIgnore != 0) return;
        printf("sql_exec_direct error[%s] \n", aSQL);
        outError("sql_exec_direct error", sStmt);
    }

    if (SQL_ERROR == SQLFreeStmt(sStmt, SQL_DROP))
    {
        if (aErrIgnore != 0) return;
        outError("FreeStmt Error", sStmt);
    }
}

void createTable()
{
    executeDirectSQL("DROP TABLE CLI_SAMPLE", 1);
    executeDirectSQL("CREATE TABLE CLI_SAMPLE(seq short, score integer, total long, percentage float, ratio doub
}

void selectTable()
{
    SQLHSTMT sStmt;
    const char *aSQL = "SELECT seq, score, total, percentage, ratio, id, srcip, dstip, reg_date, tlog, image FRO

    int i=0;
    short sSeq;
    int sScore;
    long sTotal;
    float sPercentage;
    double sRatio;
    char sId [20];
    char sSrcIp[20];
    char sDstIp[50];
    SQL_TIMESTAMP_STRUCT sRegDate;
    char sLog [1024];
    char sImage[1024];
    SQL_LEN sLen;

    if (SQLAllocStmt(gCon, &sStmt) == SQL_ERROR) {
        outError("AllocStmt Error", sStmt);
    }

    if (SQLPrepare(sStmt, (SQLCHAR *)aSQL, SQL_NTS) == SQL_ERROR) {
        printf("Prepare error[%s] \n", aSQL);
        outError("Prepare error", sStmt);
    }

    if (SQLExecute(sStmt) == SQL_ERROR) {
        outError("prepared execute error", sStmt);
    }

    SQLBindCol(sStmt, 1, SQL_C_SSHORT, &sSeq, 0, &sLen);
    SQLBindCol(sStmt, 2, SQL_C_SLONG, &sScore, 0, &sLen);
    SQLBindCol(sStmt, 3, SQL_C_SBIGINT, &sTotal, 0, &sLen);
    SQLBindCol(sStmt, 4, SQL_C_FLOAT, &sPercentage, 0, &sLen);
    SQLBindCol(sStmt, 5, SQL_C_DOUBLE, &sRatio, 0, &sLen);
    SQLBindCol(sStmt, 6, SQL_C_CHAR, sId, sizeof(sId), &sLen);
    SQLBindCol(sStmt, 7, SQL_C_CHAR, sSrcIp, sizeof(sSrcIp), &sLen);
    SQLBindCol(sStmt, 8, SQL_C_CHAR, sDstIp, sizeof(sDstIp), &sLen);
    SQLBindCol(sStmt, 9, SQL_C_TYPE_TIMESTAMP, &sRegDate, 0, &sLen);
    SQLBindCol(sStmt, 10, SQL_C_CHAR, sLog, sizeof(sLog), &sLen);
    SQLBindCol(sStmt, 11, SQL_C_CHAR, sImage, sizeof(sImage), &sLen);

    while (SQLFetch(sStmt) == SQL_SUCCESS)
    {
```

```
        printf("===== %d ========\n", i++);
        printf("seq = %d", sSeq);
        printf(", score = %d", sScore);
        printf(", total = %ld", sTotal);
        printf(", percentage = %.2f", sPercentage);
        printf(", ratio = %g", sRatio);
        printf(", id = %s", sId);
        printf(", srcip = %s", sSrcIp);
        printf(", dstip = %s", sDstIp);
        printf(", regdate = %d-%02d-%02d %02d:%02d:%02d",
               sRegDate.year, sRegDate.month, sRegDate.day,
               sRegDate.hour, sRegDate.minute, sRegDate.second);
        printf(", log = %s", sLog);
        printf(", image = %s\n", sImage);
    }

    if (SQL_ERROR == SQLFreeStmt(sStmt, SQL_DROP))
    {
        outError("FreeStmt eror", sStmt);
    }
}

void prepareInsert()
{
    SQLHSTMT sStmt;
    int i;
    short sSeq;
    int sScore;
    long sTotal;
    float sPercentage;
    double sRatio;
    char sId [20];
    char sSrcIp [20];
    char sDstIp [50];
    long reg_date;
    char sLog [100];
    char sImage [100];
    int sLength[5];

    const char *sSQL = "INSERT INTO CLI_SAMPLE VALUES(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ? )";

    if (SQLAllocStmt(gCon, &sStmt) == SQL_ERROR)
    {
        outError("AllocStmt error", sStmt);
    }

    if (SQLPrepare(sStmt, (SQLCHAR *)sSQL, SQL_NTS) == SQL_ERROR)
    {
        printf("Prepare error[%s]\n", sSQL);
        outError("Prepare error", sStmt);
    }

    for(i=1; i<10; i++)
    {
        sSeq = i;
        sScore = i+i;
        sTotal = (sSeq + sScore) * 10000;
        sPercentage = (float)(sScore+2)/sScore;
        sRatio = (double)(sSeq+1)/sTotal;
        sprintf(sId, "id-%d", i);
        sprintf(sSrcIp, "192.168.0.%d", i);
        sprintf(sDstIp, "2001:0DB8:0000:0000:0000:0000:1428:%04x", i);
        reg_date = i*10000;
        sprintf(sLog, "log-%d", i);
        sprintf(sImage, "image-%d", i);

        if (SQLBindParameter(sStmt,
                             1,
                             SQL_PARAM_INPUT,
                             SQL_C_SSHORT,
```

```
                    SQL_SMALLINT,
                    0,
                    0,
                    &sSeq,
                    0,
                    NULL) == SQL_ERROR)
{
    outError("BindParameter error 1", sStmt);
}

if (SQLBindParameter(sStmt,
                    2,
                    SQL_PARAM_INPUT,
                    SQL_C_SLONG,
                    SQL_INTEGER,
                    0,
                    0,
                    &sScore,
                    0,
                    NULL) == SQL_ERROR)
{
    outError("BindParameter error 2", sStmt);
}

if (SQLBindParameter(sStmt,
                    3,
                    SQL_PARAM_INPUT,
                    SQL_C_SBIGINT,
                    SQL_BIGINT,
                    0,
                    0,
                    &sTotal,
                    0,
                    NULL) == SQL_ERROR)
{
    outError("BindParameter error 3", sStmt);
}

if (SQLBindParameter(sStmt,
                    4,
                    SQL_PARAM_INPUT,
                    SQL_C_FLOAT,
                    SQL_FLOAT,
                    0,
                    0,
                    &sPercentage,
                    0,
                    NULL) == SQL_ERROR)
{
    outError("BindParameter error 4", sStmt);
}

if (SQLBindParameter(sStmt,
                    5,
                    SQL_PARAM_INPUT,
                    SQL_C_DOUBLE,
                    SQL_DOUBLE,
                    0,
                    0,
                    &sRatio,
                    0,
                    NULL) == SQL_ERROR)
{
    outError("BindParameter error 5", sStmt);
}

sLength[0] = strlen(sId);
if (SQLBindParameter(sStmt,
                    6,
                    SQL_PARAM_INPUT,
```

```
                    SQL_C_CHAR,
                    SQL_VARCHAR,
                    0,
                    0,
                    sId,
                    0,
                    (SQLLEN *)&sLength[0]) == SQL_ERROR)
{
    outError("BindParameter error 6", sStmt);
}

sLength[1] = strlen(sSrcIp);
if (SQLBindParameter(sStmt,
                    7,
                    SQL_PARAM_INPUT,
                    SQL_C_CHAR,
                    SQL_IPV4,
                    0,
                    0,
                    sSrcIp,
                    0,
                    (SQLLEN *)&sLength[1]) == SQL_ERROR)
{
    outError("BindParameter error 7", sStmt);
}

sLength[2] = strlen(sDstIp);
if (SQLBindParameter(sStmt,
                    8,
                    SQL_PARAM_INPUT,
                    SQL_C_CHAR,
                    SQL_IPV6,
                    0,
                    0,
                    sDstIp,
                    0,
                    (SQLLEN *)&sLength[2]) == SQL_ERROR)
{
    outError("BindParameter error 8", sStmt);
}

if (SQLBindParameter(sStmt,
                    9,
                    SQL_PARAM_INPUT,
                    SQL_C_SBIGINT,
                    SQL_DATE,
                    0,
                    0,
                    &reg_date,
                    0,
                    NULL) == SQL_ERROR)
{
    outError("BindParameter error 9", sStmt);
}

sLength[3] = strlen(sLog);
if (SQLBindParameter(sStmt,
                    10,
                    SQL_PARAM_INPUT,
                    SQL_C_CHAR,
                    SQL_VARCHAR,
                    0,
                    0,
                    sLog,
                    0,
                    (SQLLEN *)&sLength[3]) == SQL_ERROR)
{
    outError("BindParameter error 10", sStmt);
}
```

```
        sLength[4] = strlen(sImage);
        if (SQLBindParameter(sStmt,
                             11,
                             SQL_PARAM_INPUT,
                             SQL_C_CHAR,
                             SQL_BINARY,
                             0,
                             0,
                             sImage,
                             0,
                             (SQLLEN *)&sLength[4]) == SQL_ERROR)
        {
            outError("BindParameter error 11", sStmt);
        }

        if( SQLExecute(sStmt) == SQL_ERROR) {
            outError("prepare execute error", sStmt);
        }

        printf("%d prepared record inserted\n", i);

    }

    if (SQL_ERROR == SQLFreeStmt(sStmt, SQL_DROP)) {
        outError("FreeStmt", sStmt);
    }
}

int main()
{
    connectDB();
    createTable();
    prepareInsert();
    selectTable();
    disconnectDB();

    return 0;
}
```

If you register sample3_prepare.c in the Makefile, compile and run it, it will appear as follows.

```
[mach@localhost cli]$ make

[mach@localhost cli]$ ./sample3_prepare

connected ...
1 prepared record inserted
2 prepared record inserted
3 prepared record inserted
4 prepared record inserted
5 prepared record inserted
6 prepared record inserted
7 prepared record inserted
8 prepared record inserted
9 prepared record inserted
===== 0 ========
seq = 9, score = 18, total = 270000, percentage = 1.11, ratio = 3.7037e-05, id = id-9, srcip = 192.168.0.9, dstip =
===== 1 ========
seq = 8, score = 16, total = 240000, percentage = 1.12, ratio = 3.75e-05, id = id-8, srcip = 192.168.0.8, dstip = 2
===== 2 ========
seq = 7, score = 14, total = 210000, percentage = 1.14, ratio = 3.80952e-05, id = id-7, srcip = 192.168.0.7, dstip
===== 3 ========
seq = 6, score = 12, total = 180000, percentage = 1.17, ratio = 3.88889e-05, id = id-6, srcip = 192.168.0.6, dstip
===== 4 ========
seq = 5, score = 10, total = 150000, percentage = 1.20, ratio = 4e-05, id = id-5, srcip = 192.168.0.5, dstip = 200
===== 5 ========
seq = 4, score = 8, total = 120000, percentage = 1.25, ratio = 4.16667e-05, id = id-4, srcip = 192.168.0.4, dstip =
===== 6 ========
```

```
seq = 3, score = 6, total = 90000, percentage = 1.33, ratio = 4.44444e-05, id = id-3, srcip = 192.168.0.3, dstip =
===== 7 ========
seq = 2, score = 4, total = 60000, percentage = 1.50, ratio = 5e-05, id = id-2, srcip = 192.168.0.2, dstip = 2001:0
===== 8 ========
seq = 1, score = 2, total = 30000, percentage = 2.00, ratio = 6.66667e-05, id = id-1, srcip = 192.168.0.1, dstip =
```

Extension Function Append Example

In Machbase, the append protocol is provided by reading a large amount of data from a file and inputting it at a high speed. Let's write an example program that uses this Append protocol.
First, let's look at an example of how to append to the various types provided by Machbase. The Append method has its own settings for each type. Therefore, if you know how to use every method, you will be able to write programs more efficiently. All the methods are listed in the example code at the bottom.
The file name is sample4_append1.c.

› sample4_append1.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <machbase_sqlcli.h>
#include <arpa/inet.h>

#if __linux__
#include <sys/time.h>
#endif

#if defined(SUPPORT_STRUCT_TM)
# include <time.h>
#endif

#define MACHBASE_PORT_NO 5656
#define MAX_APPEND_COUNT 0xFFFFFFFF
#define ERROR_CHECK_COUNT 100

#define ERROR -1
#define SUCCESS 0

SQLHENV gEnv;
SQLHDBC gCon;
SQLHSTMT gStmt;
SQLCHAR gErrorState[6];

void connectDB();
void disconnectDB();
void outError(const char *aMsg);
void executeDirectSQL(const char *aSQL, int aErrIgnore);
void createTable();
void appendOpen();
void appendData();
int appendClose();
time_t getTimeStamp();

int main()
{
    unsigned int sCount=0;
    time_t sStartTime, sEndTime;

    connectDB();
    createTable();

    appendOpen();
    sStartTime = getTimeStamp();
    appendData();
    sEndTime = getTimeStamp();
    appendClose();

    printf("timegap = %ld microseconds for %d records\n", sEndTime - sStartTime, sCount);
    printf("%.2f records/second\n", ((double)sCount/(double)(sEndTime - sStartTime))*1000000);

    disconnectDB();
```

```
    return SUCCESS;
}

void connectDB()
{
    char sConnStr[1024];

    if (SQL_ERROR == SQLAllocEnv(&gEnv)) {
        outError("SQLAllocEnv error!!");
    }

    if (SQL_ERROR == SQLAllocConnect(gEnv, &gCon)) {
        outError("SQLAllocConnect error!!");
    }

    sprintf(sConnStr,"SERVER=127.0.0.1;UID=SYS;PWD=MANAGER;CONNTYPE=1;PORT_NO=%d", MACHBASE_PORT_NO);

    if (SQL_ERROR == SQLDriverConnect( gCon, NULL,
                                       (SQLCHAR *)sConnStr, SQL_NTS,
                                       NULL, 0, NULL,
                                       SQL_DRIVER_NOPROMPT ))
    {
        outError("connection error\n");
    }

    if (SQL_ERROR == SQLAllocStmt(gCon, &gStmt) )
    {
        outError("AllocStmt error");
    }

    printf("connected ... \n");
}

void disconnectDB()
{
    if( SQL_ERROR == SQLFreeStmt(gStmt, SQL_DROP) )
    {
        outError("SQLFreeStmt error");
    }

    if (SQL_ERROR == SQLDisconnect(gCon)) {
        outError("disconnect error");
    }
    SQLFreeConnect(gCon);
    SQLFreeEnv(gEnv);
}

void outError(const char *aMsg)
{
    SQLINTEGER sErrorNo;
    SQLSMALLINT sMsgLength;
    SQLCHAR sErrorMsg[1024];

    printf("ERROR : (%s)\n", aMsg);
    if (SQL_SUCCESS == SQLError( gEnv, gCon, gStmt, NULL, &sErrorNo,
                                 sErrorMsg, 1024, &sMsgLength ))
    {
        printf(" mach-%d : %s\n", sErrorNo, sErrorMsg);
    }

    if( gStmt )
    {
        SQLFreeStmt(gStmt, SQL_DROP);
    }

    if( gCon )
    {
        SQLFreeConnect( gCon );
    }
```

```
    if( gEnv )
    {
        SQLFreeEnv( gEnv );
    }
    exit(ERROR);
}

void executeDirectSQL(const char *aSQL, int aErrIgnore)
{
    SQLHSTMT sStmt;

    if (SQLAllocStmt(gCon, &sStmt) == SQL_ERROR)
    {
        if (aErrIgnore != 0) return;
        outError("AllocStmt error");
    }

    if (SQLExecDirect(sStmt, (SQLCHAR *)aSQL, SQL_NTS) == SQL_ERROR)
    {
        if (aErrIgnore != 0) return;
        printf("sql_exec_direct error[%s] \n", aSQL);
        outError("sql_exec_direct error");
    }

    if (SQL_ERROR == SQLFreeStmt(sStmt, SQL_DROP))
    {
        if (aErrIgnore != 0) return;
        outError("FreeStmt Error");
    }
}

void createTable()
{
    executeDirectSQL("DROP TABLE CLI_SAMPLE", 1);
    executeDirectSQL("CREATE TABLE CLI_SAMPLE(short1 short, integer1 integer, long1 long, float1 float, double1
}

void appendOpen()
{
    const char *sTableName = "CLI_SAMPLE";

    if( SQLAppendOpen(gStmt, (SQLCHAR *)sTableName, ERROR_CHECK_COUNT) != SQL_SUCCESS )
    {
        outError("SQLAppendOpen error");
    }

    printf("append open ok\n");
}

void appendData()
{
    SQL_APPEND_PARAM sParam[11];
    char sVarchar[10] = {0, };
    char sText[100] = {0, };
    char sBinary[100] = {0, };

    memset(sParam, 0, sizeof(sParam));

    /* NULL FOR ALL*/
    /* fixed column */
    sParam[0].mShort = SQL_APPEND_SHORT_NULL;
    sParam[1].mInteger = SQL_APPEND_INTEGER_NULL;
    sParam[2].mLong = SQL_APPEND_LONG_NULL;
    sParam[3].mFloat = SQL_APPEND_FLOAT_NULL;
    sParam[4].mDouble = SQL_APPEND_DOUBLE_NULL;
    /* datetime */
    sParam[5].mDateTime.mTime = SQL_APPEND_DATETIME_NULL;
    /* varchar */
    sParam[6].mVarchar.mLength = SQL_APPEND_VARCHAR_NULL;
    /* ipv4 */
```

```
sParam[7].mIP.mLength = SQL_APPEND_IP_NULL;
/* ipv6 */
sParam[8].mIP.mLength = SQL_APPEND_IP_NULL;
/* text */
sParam[9].mText.mLength = SQL_APPEND_TEXT_NULL;
/* binary */
sParam[10].mBinary.mLength = SQL_APPEND_BINARY_NULL;
SQLAppendDataV2(gStmt, sParam);

/* FIXED COLUMN Value */
sParam[0].mShort = 2;
sParam[1].mInteger = 4;
sParam[2].mLong = 6;
sParam[3].mFloat = 8.4;
sParam[4].mDouble = 10.9;
SQLAppendDataV2(gStmt, sParam);

/* DATETIME : absolute value */
sParam[5].mDateTime.mTime = MACH_UINT64_LITERAL(1000000000);
SQLAppendDataV2(gStmt, sParam);

/* DATETIME : current */
sParam[5].mDateTime.mTime = SQL_APPEND_DATETIME_NOW;
SQLAppendDataV2(gStmt, sParam);

/* DATETIME : string format*/
sParam[5].mDateTime.mTime = SQL_APPEND_DATETIME_STRING;
sParam[5].mDateTime.mDateStr = "23/May/2014:17:41:28";
sParam[5].mDateTime.mFormatStr = "DD/MON/YYYY:HH24:MI:SS";
SQLAppendDataV2(gStmt, sParam);

/* DATETIME : struct tm format*/
sParam[5].mDateTime.mTime = SQL_APPEND_DATETIME_STRUCT_TM;
sParam[5].mDateTime.mTM.tm_year = 2000 - 1900;
sParam[5].mDateTime.mTM.tm_mon = 11;
sParam[5].mDateTime.mTM.tm_mday = 31;
SQLAppendDataV2(gStmt, sParam);

/* VARCHAR : string */
strcpy(sVarchar, "MY VARCHAR");
sParam[6].mVar.mLength = strlen(sVarchar);
sParam[6].mVar.mData = sVarchar;
SQLAppendDataV2(gStmt, sParam);

/* IPv4 : ipv4 from binary bytes */
sParam[7].mIP.mLength = SQL_APPEND_IP_IPV4;
sParam[7].mIP.mAddr[0] = 127;
sParam[7].mIP.mAddr[1] = 0;
sParam[7].mIP.mAddr[2] = 0;
sParam[7].mIP.mAddr[3] = 1;
SQLAppendDataV2(gStmt, sParam);

/* IPv4 : ipv4 from binary */
sParam[7].mIP.mLength = SQL_APPEND_IP_IPV4;
*(in_addr_t *)(sParam[7].mIP.mAddr) = inet_addr("192.168.0.1");
SQLAppendDataV2(gStmt, sParam);

/* IPv4 : ipv4 from string */
sParam[7].mIP.mLength = SQL_APPEND_IP_STRING;
sParam[7].mIP.mAddrString = "203.212.222.111";
SQLAppendDataV2(gStmt, sParam);

/* IPv6 : ipv6 from binary bytes */
sParam[8].mIP.mLength = SQL_APPEND_IP_IPV6;
sParam[8].mIP.mAddr[0] = 127;
sParam[8].mIP.mAddr[1] = 127;
sParam[8].mIP.mAddr[2] = 127;
sParam[8].mIP.mAddr[3] = 127;
sParam[8].mIP.mAddr[4] = 127;
sParam[8].mIP.mAddr[5] = 127;
```

```
    sParam[8].mIP.mAddr[6] = 127;
    sParam[8].mIP.mAddr[7] = 127;
    sParam[8].mIP.mAddr[8] = 127;
    sParam[8].mIP.mAddr[9] = 127;
    sParam[8].mIP.mAddr[10] = 127;
    sParam[8].mIP.mAddr[11] = 127;
    sParam[8].mIP.mAddr[12] = 127;
    sParam[8].mIP.mAddr[13] = 127;
    sParam[8].mIP.mAddr[14] = 127;
    sParam[8].mIP.mAddr[15] = 127;
    SQLAppendDataV2(gStmt, sParam);
    sParam[8].mIP.mLength = SQL_APPEND_IP_NULL; /* recover */

    /* TEXT : string */
    memset(sText, 'X', sizeof(sText));
    sParam[9].mVar.mLength = 100;
    sParam[9].mVar.mData = sText;
    SQLAppendDataV2(gStmt, sParam);

    /* BINARY : datas */
    memset(sBinary, 0xFA, sizeof(sBinary));
    sParam[10].mVar.mLength = 100;
    sParam[10].mVar.mData = sBinary;
    SQLAppendDataV2(gStmt, sParam);
}

int appendClose()
{
    int sSuccessCount = 0;
    int sFailureCount = 0;

    if( SQLAppendClose(gStmt, &sSuccessCount, &sFailureCount) != SQL_SUCCESS )
    {
        outError("SQLAppendClose error");
    }

    printf("append close ok\n");
    printf("success : %d, failure : %d\n", sSuccessCount, sFailureCount);
    return sSuccessCount;
}

time_t getTimeStamp()
{
#if _WIN32 || _WIN64

#if defined(_MSC_VER) || defined(_MSC_EXTENSIONS)
#define DELTA_EPOCH_IN_MICROSECS 11644473600000000Ui64
#else
#define DELTA_EPOCH_IN_MICROSECS 11644473600000000ULL
#endif
    FILETIME sFT;
    unsigned __int64 sTempResult = 0;

    GetSystemTimeAsFileTime(&sFT);

    sTempResult |= sFT.dwHighDateTime;
    sTempResult <<= 32;
    sTempResult |= sFT.dwLowDateTime;

    sTempResult -= DELTA_EPOCH_IN_MICROSECS;
    sTempResult /= 10;

    return sTempResult;
#else
    struct timeval sTimeVal;
    int sRet;

    sRet = gettimeofday(&sTimeVal, NULL);

    if (sRet == 0)
```

```
    {
        return (time_t)(sTimeVal.tv_sec * 1000000 + sTimeVal.tv_usec);
    }
    else
    {
        return 0;
    }
#endif
}
```

If you register sample4_append1.c in the Makefile, compile and run it, it will appear as follows.

```
[mach@localhost cli]$ make sample4_append1
gcc -c -g -W -Wall -rdynamic -fno-inline -m64 -mtune=k8 -g -W -Wall -rdynamic -fno-inline -m64 -mtune=k8 -I/home/ma
gcc -m64 -mtune=k8 -L/home/mach/machbase_home/lib -osample4_append1 sample4_append1.o -lmachcli -L/home/mach/machba
[mach@localhost cli]$ ./sample4_append1
connected ...
append open ok
append close ok
success : 13, failure : 0
timegap = 48 microseconds for 13 records
270833.33 records/second
[mach@localhost cli]$

You can check what is inserted after MACH_SQL.

Mach> select * from CLI_SAMPLE;
SHORT1 INTEGER1 LONG1 FLOAT1 DOUBLE1
---------------------------------------------------------------------------------------------------------
DATETIME1 VARCHAR1 IP IP2
---------------------------------------------------------------------------------------------------------------------
TEXT1
-------------------------------------------------------------------------------
BIN1
-------------------------------------------------------------------------------
2 4 6 8.4 10.9
2000-12-31 00:00:00 000:000:000 MY VARCHAR 203.212.222.111 NULL
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXX
FAFAFAFAFAFAFAFAFAFAFAFAFAFAFAFAFAFAFAFAFAFAFAFAFAFAFAFAFAFAFAFAFAFAFAFAFAFAFAFAFAFAFA
FAFAFAFAFAFAFAFAFAFAFAFAFAFAFAFAFAFAFAFAFAFAFAFAFAFAFAFAFAFAFAFAFAFAFAFAFAFAFAFAFAFAFA
FAFAFAFAFAFAFAFAFAFAFAFAFAFAFAFAFAFAFAFA
2 4 6 8.4 10.9
2000-12-31 00:00:00 000:000:000 MY VARCHAR 203.212.222.111 NULL
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXX
NULL
2 4 6 8.4 10.9
2000-12-31 00:00:00 000:000:000 MY VARCHAR 203.212.222.111 7F7F:7F7F:7F7F:7F7F:7F7F:7F7F:7F7F:7F7F:7F7F:7F7F
NULL
NULL
2 4 6 8.4 10.9
2000-12-31 00:00:00 000:000:000 MY VARCHAR 203.212.222.111 NULL
NULL
NULL
2 4 6 8.4 10.9
2000-12-31 00:00:00 000:000:000 MY VARCHAR 192.168.0.1 NULL
NULL
NULL
2 4 6 8.4 10.9
2000-12-31 00:00:00 000:000:000 MY VARCHAR 127.0.0.1 NULL
NULL
NULL
2 4 6 8.4 10.9
2000-12-31 00:00:00 000:000:000 MY VARCHAR NULL NULL
NULL
NULL
2 4 6 8.4 10.9
```

```
2000-12-31 00:00:00 000:000:000 NULL NULL NULL
NULL
NULL
2 4 6 8.4 10.9
2014-05-23 17:41:28 000:000:000 NULL NULL NULL
NULL
NULL
2 4 6 8.4 10.9
2015-04-09 16:44:11 134:256:000 NULL NULL NULL
NULL
NULL
2 4 6 8.4 10.9
1970-01-01 09:00:01 000:000:000 NULL NULL NULL
NULL
NULL
2 4 6 8.4 10.9
1970-01-01 09:00:00 000:000:000 NULL NULL NULL
NULL
NULL
[12] row(s) selected.
```

Now let's use a fast append method using a file. This is an example useful for fast input of large amounts of logs, packets, etc. used in business. The file name is sample4_append2.c.
You have to save the data to be entered in advance in data.txt.

```
./make_data
```

Modifying the given make_data.c gives you the opportunity to create a data.txt file for your situation.

› make_data.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/time.h>
#include <machbase_sqlcli.h>

#define MACHBASE_PORT_NO 5656
#define MAX_APPEND_COUNT 0xFFFFFFFF
#define ERROR_CHECK_COUNT 100

SQLHENV gEnv;
SQLHDBC gCon;
SQLHSTMT gStmt;
SQLCHAR gErrorState[6];

void connectDB();
void disconnectDB();
void outError(const char *aMsg);
void executeDirectSQL(const char *aSQL, int aErrIgnore);
void createTable();
void appendOpen();
int appendData();
void appendClose();
time_t getTimeStamp();

int main()
{
    unsigned int sCount=0;
    time_t sStartTime, sEndTime;

    connectDB();
    createTable();

    appendOpen();
    sStartTime = getTimeStamp();
    sCount = appendData();
    sEndTime = getTimeStamp();
```

```
    appendClose();

    printf("timegap = %ld microseconds for %d records\n", sEndTime - sStartTime, sCount);
    printf("%.2f records/second\n", ((double)sCount/(double)(sEndTime - sStartTime))*1000000);

    disconnectDB();

    return 0;
}

void connectDB()
{
    char sConnStr[1024];

    if (SQL_ERROR == SQLAllocEnv(&gEnv)) {
        outError("SQLAllocEnv error!!");
    }

    if (SQL_ERROR == SQLAllocConnect(gEnv, &gCon)) {
        outError("SQLAllocConnect error!!");
    }

    sprintf(sConnStr,"SERVER=127.0.0.1;UID=SYS;PWD=MANAGER;CONNTYPE=1;PORT_NO=%d", MACHBASE_PORT_NO);

    if (SQL_ERROR == SQLDriverConnect( gCon, NULL,
                                       (SQLCHAR *)sConnStr, SQL_NTS,
                                       NULL, 0, NULL,
                                       SQL_DRIVER_NOPROMPT ))
    {
        outError("connection error!!");
    }

    if( SQL_ERROR == SQLAllocStmt(gCon, &gStmt) )
    {
        outError("SQLAllocStmt error!!");
    }

    printf("connected ... \n");
}

void disconnectDB()
{
    if( SQL_ERROR == SQLFreeStmt(gStmt, SQL_DROP) )
    {
        outError("SQLFreeStmt error");
    }

    if (SQL_ERROR == SQLDisconnect(gCon)) {
        outError("disconnect error");
    }

    SQLFreeConnect(gCon);
    SQLFreeEnv(gEnv);
}

void outError(const char *aMsg)
{
    SQLINTEGER sErrorNo;
    SQLSMALLINT sMsgLength;
    SQLCHAR sErrorMsg[1024];

    printf("ERROR : (%s)\n", aMsg);

    if (SQL_SUCCESS == SQLError( gEnv, gCon, gStmt, NULL, &sErrorNo,
                                 sErrorMsg, 1024, &sMsgLength ))
    {
        printf(" mach-%d : %s\n", sErrorNo, sErrorMsg);
    }
```

```
    if( gStmt )
    {
        SQLFreeStmt( gStmt, SQL_DROP );
    }
    if( gCon )
    {
        SQLFreeConnect( gCon );
    }
    if( gEnv )
    {
        SQLFreeEnv( gEnv );
    }
    exit(-1);
}

void executeDirectSQL(const char *aSQL, int aErrIgnore)
{
    SQLHSTMT sStmt;

    if (SQLAllocStmt(gCon, &sStmt) == SQL_ERROR)
    {
        if (aErrIgnore != 0) return;
        outError("AllocStmt error");
    }

    if (SQLExecDirect(sStmt, (SQLCHAR *)aSQL, SQL_NTS) == SQL_ERROR)
    {
        if (aErrIgnore != 0) return;
        outError("sql_exec_direct error");
    }

    if (SQL_ERROR == SQLFreeStmt(sStmt, SQL_DROP))
    {
        if (aErrIgnore != 0) return;
        outError("FreeStmt Error");
    }
}

void createTable()
{
    executeDirectSQL("DROP TABLE CLI_SAMPLE", 1);
    executeDirectSQL("CREATE TABLE CLI_SAMPLE(seq short, score integer, total long, percentage float, ratio doub

    printf("table created\n");
}

void appendOpen()
{
    const char *sTableName = "CLI_SAMPLE";

    if( SQLAppendOpen(gStmt, (SQLCHAR *)sTableName, ERROR_CHECK_COUNT) != SQL_SUCCESS )
    {
        outError("SQLAppendOpen error!!");
    }

    printf("append open ok\n");
}

int appendData()
{
    FILE *sFp;
    char sBuf[1024];
    int j;
    char *sToken;
    unsigned int sCount=0;
    SQL_APPEND_PARAM sParam[11];

    sFp = fopen("data.txt", "r");
    if( !sFp )
    {
```

```
        printf("file open error\n");
        exit(-1);
    }

    printf("append data start\n");

    memset(sBuf, 0, sizeof(sBuf));

    while( fgets(sBuf, 1024, sFp ) != NULL )
    {
        if( strlen(sBuf) < 1)
        {
            break;
        }

        j=0;
        sToken = strtok(sBuf,",");

        while( sToken != NULL )
        {
            memset(sParam+j, 0, sizeof(sParam));
            switch(j){
                case 0 : sParam[j].mShort = atoi(sToken); break; //short
                case 1 : sParam[j].mInteger = atoi(sToken); break; //int
                case 2 : sParam[j].mLong = atol(sToken); break; //long
                case 3 : sParam[j].mFloat = atof(sToken); break; //float
                case 4 : sParam[j].mDouble = atof(sToken); break; //double
                case 5 : //string
                case 9 : //text
                case 10 : //binary
                        sParam[j].mVar.mLength = strlen(sToken);
                        strcpy(sParam[j].mVar.mData, sToken);
                        break;
                case 6 : //ipv4
                case 7 : //ipv6
                        sParam[j].mIP.mLength = SQL_APPEND_IP_STRING;
                        strcpy(sParam[j].mIP.mAddrString, sToken);
                        break;
                case 8 : //datetime
                        sParam[j].mDateTime.mTime = SQL_APPEND_DATETIME_STRING;
                        strcpy(sParam[j].mDateTime.mDateStr, sToken);
                        sParam[j].mDateTime.mFormatStr = "DD/MON/YYYY:HH24:MI:SS";
                        break;
            }

            sToken = strtok(NULL, ",");

            j++;
        }
        if( SQLAppendDataV2(gStmt, sParam) != SQL_SUCCESS )
        {
            printf("SQLAppendData error\n");
            return 0;
        }
        if ( ((sCount++) % 10000) == 0)
        {
            printf(".");
        }

        if( ((sCount) % 100) == 0 )
        {
            if( SQLAppendFlush( gStmt ) != SQL_SUCCESS )
            {
                outError("SQLAppendFlush error");
            }
        }
        if (sCount == MAX_APPEND_COUNT)
        {
            break;
        }
```

```
    }

    printf("\nappend data end\n");

    fclose(sFp);

    return sCount;
}

void appendClose()
{
    int sSuccessCount = 0;
    int sFailureCount = 0;

    if( SQLAppendClose(gStmt, &sSuccessCount, &sFailureCount) != SQL_SUCCESS )
    {
        outError("SQLAppendClose error");
    }

    printf("append close ok\n");
    printf("success : %d, failure : %d\n", sSuccessCount, sFailureCount);
}

time_t getTimeStamp()
{
    struct timeval tv;
    gettimeofday(&tv, NULL);
    return tv.tv_sec*1000000+tv.tv_usec;
}
```

If you register sample4_append2.c in the Makefile, compile and run it, it will appear as follows.

```
[mach@localhost cli]$ make
gcc -c -g -W -Wall -rdynamic -fno-inline -m64 -mtune=k8 -g -W -Wall -rdynamic -fno-inline -m64 -mtune=k8 -I/home/ma
gcc -m64 -mtune=k8 -L/home/mach/machbase_home/lib -osingle_append2 single_append2.o -lmachcli -L/home/mach/machbase
[mach@localhost cli]$ ./single_append2
connected ...
table created
append open ok
append data start
.....................................................................................................
append data end
append close ok
success : 1000000, failure : 0
timegap = 1641503 microseconds for 1000000 records
609197.79 records/second
```

Acquiring Table Column Information Example

There are a number of ways to obtain table column information, but we will look at how to use SQLDescribeCol and SQLColumns.

SQLDescribeCol

SQLDescribeCol is a function that retrieves the number, name, buffer size, length, and type of table columns.

The sample file name is sample5_describe.c.

› sample5_describe.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <machbase_sqlcli.h>
#include <time.h>

#define MACHBASE_PORT_NO 5656

SQLHENV gEnv;
```

```
SQLHDBC gCon;
SQLHSTMT gStmt;
SQLCHAR gErrorState[6];

void connectDB()
{
    char connStr[1024];

    SQLINTEGER errNo;
    SQLSMALLINT msgLength;
    SQLCHAR errMsg[1024];

    if (SQL_ERROR == SQLAllocEnv(&gEnv)) {
        printf("SQLAllocEnv error!!\n");
        exit(1);
    }

    if (SQL_ERROR == SQLAllocConnect(gEnv, &gCon)) {
        printf("SQLAllocConnect error!!\n");
        SQLFreeEnv(gEnv);
        exit(1);
    }

    sprintf(connStr,"SERVER=127.0.0.1;UID=SYS;PWD=MANAGER;CONNTYPE=1;PORT_NO=%d", MACHBASE_PORT_NO);

    if (SQL_ERROR == SQLDriverConnect( gCon, NULL,
                                       (SQLCHAR *)connStr,
                                       SQL_NTS,
                                       NULL, 0, NULL,
                                       SQL_DRIVER_NOPROMPT ))
    {
        printf("connection error\n");

        if (SQL_SUCCESS == SQLError ( gEnv, gCon, NULL, NULL, &errNo,
                                      errMsg, 1024, &msgLength ))
        {
            printf(" mach-%d : %s\n", errNo, errMsg);
        }
        SQLFreeEnv(gEnv);
        exit(1);
    }

    if (SQLAllocStmt(gCon, &gStmt) == SQL_ERROR)
    {
        outError("AllocStmt error", gStmt);
    }

    printf("connected ... \n");

}

void disconnectDB()
{
    SQLINTEGER errNo;
    SQLSMALLINT msgLength;
    SQLCHAR errMsg[1024];

    if (SQL_ERROR == SQLDisconnect(gCon)) {
        printf("disconnect error\n");

        if( SQL_SUCCESS == SQLError( gEnv, gCon, NULL, NULL, &errNo,
                                     errMsg, 1024, &msgLength ))
        {
            printf(" mach-%d : %s\n", errNo, errMsg);
        }
    }

    SQLFreeConnect(gCon);
    SQLFreeEnv(gEnv);
}
```

```
void outError(const char *aMsg, SQLHSTMT stmt)
{
    SQLINTEGER errNo;
    SQLSMALLINT msgLength;
    SQLCHAR errMsg[1024];

    printf("ERROR : (%s)\n", aMsg);

    if (SQL_SUCCESS == SQLError( gEnv, gCon, stmt, NULL, &errNo,
                                 errMsg, 1024, &msgLength ))
    {
        printf(" mach-%d : %s\n", errNo, errMsg);
    }
    exit(-1);
}

void executeDirectSQL(const char *aSQL, int aErrIgnore)
{
    SQLHSTMT stmt;

    if (SQLAllocStmt(gCon, &stmt) == SQL_ERROR)
    {
        if (aErrIgnore != 0) return;
        outError("AllocStmt error", stmt);
    }

    if (SQLExecDirect(stmt, (SQLCHAR *)aSQL, SQL_NTS) == SQL_ERROR)
    {
        if (aErrIgnore != 0) return;
        printf("sql_exec_direct error[%s] \n", aSQL);
        outError("sql_exec_direct error", stmt);
    }

    if (SQL_ERROR == SQLFreeStmt(stmt, SQL_DROP))
    {
        if (aErrIgnore != 0) return;
        outError("FreeStmt Error", stmt);
    }
}

void createTable()
{
    executeDirectSQL("DROP TABLE CLI_SAMPLE", 1);
    executeDirectSQL("CREATE TABLE CLI_SAMPLE(seq short, score integer, total long, percentage float, ratio doub

}

int main()
{
    char sSqlStr[] = "select * from cli_sample";
    SQLCHAR sColName[32];
    SQLSMALLINT sColType;
    SQLSMALLINT sColNameLen;
    SQLSMALLINT sNullable;
    SQLULEN sColLen;
    SQLSMALLINT sDecimalDigits;
    SQLLEN sOutlen;
    SQLCHAR* sData;
    SQLLEN sDisplaySize;
    int i;

    SQLSMALLINT sColumns;

    connectDB();

    createTable();

    if(SQLPrepare(gStmt, (SQLCHAR*)sSqlStr, SQL_NTS))
    {
```

```
        outError("sql prepare fail", gStmt);
        return -1;
    }

    if(SQLNumResultCols(gStmt, &sColumns) != SQL_SUCCESS )
    {
        printf("get col length error \n");
        return -1;
    }

    printf("----------------------------------------------------------------\n");
    printf("%32s%16s%10s\n","Name","Type","Length");
    printf("----------------------------------------------------------------\n");

    for(i = 0; i < sColumns; i++)
    {
        SQLDescribeCol(gStmt,
                       (SQLUSMALLINT)(i + 1),
                       sColName,
                       sizeof(sColName),
                       &sColNameLen,
                       &sColType,
                       (SQLULEN *)&sColLen,
                       &sDecimalDigits,
                       (SQLSMALLINT *)&sNullable);

        printf("%32s%16d%10d\n",sColName, sColType, sColLen);
    }

    printf("----------------------------------------------------------------\n");

    disconnectDB();

    return 0;
}
```

If you add the above file and run make, you can see the contents of the column as shown below.

```
[mach@localhost cli]$ make

[mach@localhost cli]$ ./sample5_describe
connected ...
----------------------------------------------------------------
Name Type Length
----------------------------------------------------------------
SEQ 5 5
SCORE 4 10
TOTAL -5 19
PERCENTAGE 6 27
RATIO 8 27
ID 12 10
SRCIP 2104 15
DSTIP 2106 60
REG_DATE 9 31
TLOG 2100 67108864
IMAGE -2 67108864
----------------------------------------------------------------
[mach@localhost cli]$
```

SQLColumns

SQLColumns is a function that can find the information of the columns existing in the current table. Machbase also supports the above functions and can be used to find out the information of each column.

The file name is sample6_columns.c.

› sample6_columns.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <machbase_sqlcli.h>

#include <time.h>

#define MACHBASE_PORT_NO 5656

SQLHENV gEnv;
SQLHDBC gCon;
SQLHSTMT gStmt;
SQLCHAR gErrorState[6];

void connectDB()
{
    char connStr[1024];

    SQLINTEGER errNo;
    SQLSMALLINT msgLength;
    SQLCHAR errMsg[1024];

    if (SQL_ERROR == SQLAllocEnv(&gEnv)) {
        printf("SQLAllocEnv error!!\n");
        exit(1);
    }

    if (SQL_ERROR == SQLAllocConnect(gEnv, &gCon)) {
        printf("SQLAllocConnect error!!\n");
        SQLFreeEnv(gEnv);
        exit(1);
    }

    sprintf(connStr,"SERVER=127.0.0.1;UID=SYS;PWD=MANAGER;CONNTYPE=1;PORT_NO=%d", MACHBASE_PORT_NO);

    if (SQL_ERROR == SQLDriverConnect( gCon, NULL,
                                       (SQLCHAR *)connStr,
                                       SQL_NTS,
                                       NULL, 0, NULL,
                                       SQL_DRIVER_NOPROMPT ))
    {
        printf("connection error\n");

        if (SQL_SUCCESS == SQLError ( gEnv, gCon, NULL, NULL, &errNo,
                                      errMsg, 1024, &msgLength ))
        {
            printf(" mach-%d : %s\n", errNo, errMsg);
        }
        SQLFreeEnv(gEnv);
        exit(1);
    }

    if (SQLAllocStmt(gCon, &gStmt) == SQL_ERROR)
    {
        outError("AllocStmt error", gStmt);
    }

    printf("connected ... \n");

}

void disconnectDB()
{
    SQLINTEGER errNo;
    SQLSMALLINT msgLength;
    SQLCHAR errMsg[1024];

    if (SQL_ERROR == SQLDisconnect(gCon)) {
        printf("disconnect error\n");
```

```
        if( SQL_SUCCESS == SQLError( gEnv, gCon, NULL, NULL, &errNo,
                                     errMsg, 1024, &msgLength ))
        {
            printf(" mach-%d : %s\n", errNo, errMsg);
        }
    }

    SQLFreeConnect(gCon);
    SQLFreeEnv(gEnv);
}

void outError(const char *aMsg, SQLHSTMT stmt)
{
    SQLINTEGER errNo;
    SQLSMALLINT msgLength;
    SQLCHAR errMsg[1024];

    printf("ERROR : (%s)\n", aMsg);

    if (SQL_SUCCESS == SQLError( gEnv, gCon, stmt, NULL, &errNo,
                                 errMsg, 1024, &msgLength ))
    {
        printf(" mach-%d : %s\n", errNo, errMsg);
    }
    exit(-1);
}

void executeDirectSQL(const char *aSQL, int aErrIgnore)
{
    SQLHSTMT stmt;

    if (SQLAllocStmt(gCon, &stmt) == SQL_ERROR)
    {
        if (aErrIgnore != 0) return;
        outError("AllocStmt error", stmt);
    }

    if (SQLExecDirect(stmt, (SQLCHAR *)aSQL, SQL_NTS) == SQL_ERROR)
    {
        if (aErrIgnore != 0) return;
        printf("sql_exec_direct error[%s] \n", aSQL);
        outError("sql_exec_direct error", stmt);
    }

    if (SQL_ERROR == SQLFreeStmt(stmt, SQL_DROP))
    {
        if (aErrIgnore != 0) return;
        outError("FreeStmt Error", stmt);
    }
}

void createTable()
{
    executeDirectSQL("DROP TABLE CLI_SAMPLE", 1);
    executeDirectSQL("CREATE TABLE CLI_SAMPLE(seq short, score integer, total long, percentage float, ratio doub
}

int main()
{
    SQLCHAR sColName[32];
    SQLSMALLINT sColType;
    SQLCHAR sColTypeName[16];
    SQLSMALLINT sColNameLen;
    SQLSMALLINT sColTypeLen;
    SQLSMALLINT sNullable;
    SQLULEN sColLen;
    SQLSMALLINT sDecimalDigits;
    SQLLEN sOutlen;
    SQLCHAR* sData;
```

```
    SQLLEN sDisplaySize;
    int i;

    SQLSMALLINT sColumns;

    connectDB();

    createTable();

    if(SQLColumns(gStmt, NULL, 0, NULL, 0, "cli_sample", SQL_NTS, NULL, 0) != SQL_SUCCESS)
    {
        printf("sql columns error!\n");
        return -1;
    }

    SQLBindCol(gStmt, 4, SQL_C_CHAR, sColName, sizeof(sColName), &sColNameLen);
    SQLBindCol(gStmt, 5, SQL_C_SSHORT, &sColType, 0, &sColTypeLen);
    SQLBindCol(gStmt, 6, SQL_C_CHAR, sColTypeName, sizeof(sColTypeName), NULL);
    SQLBindCol(gStmt, 7, SQL_C_SLONG, &sColLen, 0, NULL);

    printf("--------------------------------------------------------------------------------\n");
    printf("%32s%16s%16s%10s\n","Name","Type","TypeName","Length");
    printf("--------------------------------------------------------------------------------\n");

    while( SQLFetch(gStmt) != SQL_NO_DATA )
    {
        printf("%32s%16d%16s%10d\n",sColName, sColType, sColTypeName, sColLen);
    }
    printf("--------------------------------------------------------------------------------\n");

    disconnectDB();

    return 0;
}
```

Add the above file and run make. The results are as follows.

```
[mach@localhost cli]$ make

[mach@localhost cli]$ ./sample6_columns
connected ...
--------------------------------------------------------------------------------
Name Type TypeName Length
--------------------------------------------------------------------------------
_ARRIVAL_TIME 93 DATE 31
SEQ 5 SMALLINT 5
SCORE 4 INTEGER 10
TOTAL -5 BIGINT 19
PERCENTAGE 6 FLOAT 27
RATIO 8 DOUBLE 27
ID 12 VARCHAR 10
SRCIP 2104 IPV4 15
DSTIP 2106 IPV6 60
REG_DATE 93 DATE 31
TLOG 2100 TEXT 67108864
IMAGE -2 BINARY 67108864
--------------------------------------------------------------------------------
```

Multi-Thread Append Example

An example of using multiple threads in one program to append to multiple tables.

The file name is sample8_multi_session_multi_table.c.

› sample8_multi_session_multi_table.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <pthread.h>
```

```c
#include <machbase_sqlcli.h>

#define MACHBASE_PORT_NO       5656
#define ERROR_CHECK_COUNT   100

#define LOG_FILE_CNT        3
#define MAX_THREAD_NUM      LOG_FILE_CNT

#define RC_FAILURE          -1
#define RC_SUCCESS          0

#define UNUSED(aVar) do { (void)(aVar); } while(0)

char *gTableName[LOG_FILE_CNT] = {"table_f1", "table_f2", "table_event"};
char *gFileName[LOG_FILE_CNT] =  {"suffle_data1.txt","suffle_data2.txt","suffle_data3.txt"};

void printError(SQLHENV aEnv, SQLHDBC aCon, SQLHSTMT aStmt, char *aMsg);
int connectDB(SQLHENV *aEnv, SQLHDBC *aCon);
void disconnectDB(SQLHENV aEnv, SQLHDBC aCon);
int executeDirectSQL(SQLHENV aEnv, SQLHDBC aCon, const char *aSQL, int aErrIgnore);
int appendOpen(SQLHENV aEnv, SQLHDBC aCon, SQLHSTMT aStmt, char* aTableName);
int appendClose(SQLHENV aEnv, SQLHDBC aCon, SQLHSTMT aStmt);
int createTables(SQLHENV aEnv, SQLHDBC aCon);

/*
 * error code returned from CLI lib
 */
void printError(SQLHENV aEnv, SQLHDBC aCon, SQLHSTMT aStmt, char *aMsg)
{
    SQLINTEGER      sNativeError;
    SQLCHAR         sErrorMsg[SQL_MAX_MESSAGE_LENGTH + 1];
    SQLCHAR         sSqlState[SQL_SQLSTATE_SIZE + 1];
    SQLSMALLINT     sMsgLength;

    if( aMsg != NULL )
    {
        printf("%s\n", aMsg);
    }

    if( SQLError(aEnv, aCon, aStmt, sSqlState, &sNativeError,
                 sErrorMsg, SQL_MAX_MESSAGE_LENGTH, &sMsgLength) == SQL_SUCCESS )
    {
        printf("SQLSTATE-[%s], Machbase-[%d][%s]\n", sSqlState, sNativeError, sErrorMsg);
    }
}

/*
 * error code returned from Machbase server
 */

void appendDumpError(SQLHSTMT    aStmt,
                     SQLINTEGER  aErrorCode,
                     SQLPOINTER  aErrorMessage,
                     SQLLEN      aErrorBufLen,
                     SQLPOINTER  aRowBuf,
                     SQLLEN      aRowBufLen)
{
    char       sErrMsg[1024] = {0, };
    char       sRowMsg[32 * 1024] = {0, };

    UNUSED(aStmt);

    if (aErrorMessage != NULL)
    {
        strncpy(sErrMsg, (char *)aErrorMessage, aErrorBufLen);
    }

    if (aRowBuf != NULL)
    {
        strncpy(sRowMsg, (char *)aRowBuf, aRowBufLen);
```

```
    }

    fprintf(stdout, "Append Error : [%d][%s]\n[%s]\n\n", aErrorCode, sErrMsg, sRowMsg);
}


int connectDB(SQLHENV *aEnv, SQLHDBC *aCon)
{
    char sConnStr[1024];

    if( SQLAllocEnv(aEnv) != SQL_SUCCESS )
    {
        printf("SQLAllocEnv error\n");
        return RC_FAILURE;
    }

    if( SQLAllocConnect(*aEnv, aCon) != SQL_SUCCESS )
    {
        printf("SQLAllocConnect error\n");

        SQLFreeEnv(*aEnv);
        *aEnv = SQL_NULL_HENV;

        return RC_FAILURE;
    }

    sprintf(sConnStr,"SERVER=127.0.0.1;UID=SYS;PWD=MANAGER;CONNTYPE=1;PORT_NO=%d", MACHBASE_PORT_NO);

    if( SQLDriverConnect( *aCon, NULL,
                          (SQLCHAR *)sConnStr,
                          SQL_NTS,
                          NULL, 0, NULL,
                          SQL_DRIVER_NOPROMPT ) != SQL_SUCCESS
      )
    {

        printError(*aEnv, *aCon, NULL, "SQLDriverConnect error");

        SQLFreeConnect(*aCon);
        *aCon = SQL_NULL_HDBC;

        SQLFreeEnv(*aEnv);
        *aEnv = SQL_NULL_HENV;

        return RC_FAILURE;
    }

    return RC_SUCCESS;
}


void disconnectDB(SQLHENV aEnv, SQLHDBC aCon)
{
    if( SQLDisconnect(aCon) != SQL_SUCCESS )
    {
        printError(aEnv, aCon, NULL, "SQLDisconnect error");
    }

    SQLFreeConnect(aCon);
    aCon = SQL_NULL_HDBC;

    SQLFreeEnv(aEnv);
    aEnv = SQL_NULL_HENV;
}


int executeDirectSQL(SQLHENV aEnv, SQLHDBC aCon, const char *aSQL, int aErrIgnore)
{
    SQLHSTMT sStmt = SQL_NULL_HSTMT;
```

```
    if( SQLAllocStmt(aCon, &sStmt) != SQL_SUCCESS )
    {
        if( aErrIgnore == 0 )
        {
            printError(aEnv, aCon, sStmt, "SQLAllocStmt Error");
            return RC_FAILURE;
        }
    }

    if( SQLExecDirect(sStmt, (SQLCHAR *)aSQL, SQL_NTS) != SQL_SUCCESS )
    {

        if( aErrIgnore == 0 )
        {
            printError(aEnv, aCon, sStmt, "SQLExecDirect Error");

            SQLFreeStmt(sStmt,SQL_DROP);
            sStmt = SQL_NULL_HSTMT;
            return RC_FAILURE;
        }
    }

    if( SQLFreeStmt(sStmt, SQL_DROP) != SQL_SUCCESS )
    {
        if (aErrIgnore == 0)
        {
            printError(aEnv, aCon, sStmt, "SQLFreeStmt Error");
            sStmt = SQL_NULL_HSTMT;
            return RC_FAILURE;
        }
    }
    sStmt = SQL_NULL_HSTMT;

    return RC_SUCCESS;
}


int appendOpen(SQLHENV aEnv, SQLHDBC aCon, SQLHSTMT aStmt, char* aTableName)
{
    if( aTableName == NULL )
    {
        printf("append open wrong table name");
        return RC_FAILURE;
    }

    if( SQLAppendOpen(aStmt, (SQLCHAR *)aTableName, ERROR_CHECK_COUNT) != SQL_SUCCESS )
    {
        printError(aEnv, aCon, aStmt, "SQLAppendOpen error");
        return RC_FAILURE;
    }
    return RC_SUCCESS;
}


int appendClose(SQLHENV aEnv, SQLHDBC aCon, SQLHSTMT aStmt)
{
    int sSuccessCount = 0;
    int sFailureCount = 0;

    if( SQLAppendClose(aStmt, &sSuccessCount, &sFailureCount) != SQL_SUCCESS )
    {
        printError(aEnv, aCon, aStmt, "SQLAppendClose error");
        return RC_FAILURE;
    }

    printf("append result success : %d, failure : %d\n", sSuccessCount, sFailureCount);

    return RC_SUCCESS;
}
```

```
int createTables(SQLHENV aEnv, SQLHDBC aCon)
{
    int      i;
    char    *sSchema[] = { "srcip1 ipv4, srcip2 ipv6, srcport short, dstip1 ipv4, dstip2 ipv6, dstport short, da
        "srcip1 ipv4, srcip2 ipv6, srcport short, dstip1 ipv4, dstip2 ipv6, dstport short, data1 long, data2 lon
        "machine ipv4, err integer, msg varchar(30)"
    };

    char sDropQuery[256];
    char sCreateQuery[256];

    for(i = 0; i < LOG_FILE_CNT; i++)
    {
        snprintf(sDropQuery, 256, "DROP TABLE %s", gTableName[i]);
        snprintf(sCreateQuery, 256, "CREATE TABLE %s ( %s )", gTableName[i], sSchema[i]);

        executeDirectSQL(aEnv, aCon, sDropQuery, 1);
        executeDirectSQL(aEnv, aCon, sCreateQuery, 0);
    }

    return RC_SUCCESS;
}


int appendF1(SQLHENV aEnv, SQLHDBC aCon, SQLHSTMT aStmt, FILE *aFp)
{
    SQL_APPEND_PARAM sParam[8];
    SQLRETURN        sRC;

    SQLINTEGER      sNativeError;
    SQLCHAR         sErrorMsg[SQL_MAX_MESSAGE_LENGTH + 1];
    SQLCHAR         sSqlState[SQL_SQLSTATE_SIZE + 1];
    SQLSMALLINT     sMsgLength;

    char             sData[4][64];

    memset(sParam, 0, sizeof(sParam));

    fscanf(aFp, "%s %s %hd %s %s %hd %lld %lld\n",
           sData[0], sData[1], &sParam[2].mShort,
           sData[2], sData[3], &sParam[5].mShort,
           &sParam[6].mLong, &sParam[7].mLong);

    sParam[0].mIP.mLength = SQL_APPEND_IP_STRING;
    sParam[0].mIP.mAddrString = sData[0];

    sParam[1].mIP.mLength = SQL_APPEND_IP_STRING;
    sParam[1].mIP.mAddrString = sData[1];

    sParam[3].mIP.mLength = SQL_APPEND_IP_STRING;
    sParam[3].mIP.mAddrString = sData[2];

    sParam[4].mIP.mLength = SQL_APPEND_IP_STRING;
    sParam[4].mIP.mAddrString = sData[3];

    sRC = SQLAppendDataV2(aStmt, sParam);
    if( !SQL_SUCCEEDED(sRC) )
    {
        if( SQLError(aEnv, aCon, aStmt, sSqlState, &sNativeError,
                     sErrorMsg, SQL_MAX_MESSAGE_LENGTH, &sMsgLength) != SQL_SUCCESS )
        {
            return RC_FAILURE;
        }

        printf("SQLSTATE-[%s], Machbase-[%d][%s]\n", sSqlState, sNativeError, sErrorMsg);

        if( sNativeError != 9604 &&
            sNativeError != 9605 &&
            sNativeError != 9606 )
```

```
        {
            return RC_FAILURE;
        }
        else
        {
            //data value error in one record, so return success to keep attending
        }
    }
    return RC_SUCCESS;
}


int appendF2(SQLHENV aEnv, SQLHDBC aCon, SQLHSTMT aStmt, FILE* aFp)
{
    SQL_APPEND_PARAM sParam[8];
    SQLRETURN        sRC;

    SQLINTEGER      sNativeError;
    SQLCHAR         sErrorMsg[SQL_MAX_MESSAGE_LENGTH + 1];
    SQLCHAR         sSqlState[SQL_SQLSTATE_SIZE + 1];
    SQLSMALLINT     sMsgLength;

    char             sData[4][64];

    memset(sParam, 0, sizeof(sParam));

    fscanf(aFp, "%s %s %hd %s %s %hd %lld %lld\n",
           sData[0], sData[1], &sParam[2].mShort,
           sData[2], sData[3], &sParam[5].mShort,
           &sParam[6].mLong, &sParam[7].mLong);

    sParam[0].mIP.mLength = SQL_APPEND_IP_STRING;
    sParam[0].mIP.mAddrString = sData[0];

    sParam[1].mIP.mLength = SQL_APPEND_IP_STRING;
    sParam[1].mIP.mAddrString = sData[1];

    sParam[3].mIP.mLength = SQL_APPEND_IP_STRING;
    sParam[3].mIP.mAddrString = sData[2];

    sParam[4].mIP.mLength = SQL_APPEND_IP_STRING;
    sParam[4].mIP.mAddrString = sData[3];

    sRC = SQLAppendDataV2(aStmt, sParam);
    if( !SQL_SUCCEEDED(sRC) )
    {
        if( SQLError(aEnv, aCon, aStmt, sSqlState, &sNativeError,
                     sErrorMsg, SQL_MAX_MESSAGE_LENGTH, &sMsgLength) != SQL_SUCCESS )
        {
            return RC_FAILURE;
        }

        printf("SQLSTATE-[%s], Machbase-[%d][%s]\n", sSqlState, sNativeError, sErrorMsg);

        if( sNativeError != 9604 &&
            sNativeError != 9605 &&
            sNativeError != 9606 )
        {
            return RC_FAILURE;
        }
        else
        {
            //data value error in one record, so return success to keep attending
        }
    }
    return RC_SUCCESS;
}


int appendEvent(SQLHENV aEnv, SQLHDBC aCon, SQLHSTMT aStmt, FILE* aFp)
```

```
{
    SQL_APPEND_PARAM sParam[3];
    SQLRETURN        sRC;

    SQLINTEGER      sNativeError;
    SQLCHAR         sErrorMsg[SQL_MAX_MESSAGE_LENGTH + 1];
    SQLCHAR         sSqlState[SQL_SQLSTATE_SIZE + 1];
    SQLSMALLINT     sMsgLength;

    char             sData[2][20];

    memset(sParam, 0, sizeof(sParam));

    fscanf(aFp, "%s %d %s\n",sData[0], &sParam[1].mInteger, sData[1]);

    sParam[0].mIP.mLength = SQL_APPEND_IP_STRING;
    sParam[0].mIP.mAddrString = sData[0];

    sParam[2].mVarchar.mLength = strlen(sData[1]);
    sParam[2].mVarchar.mData = sData[1];

    sRC = SQLAppendDataV2(aStmt, sParam);
    if( !SQL_SUCCEEDED(sRC) )
    {
        if( SQLError(aEnv, aCon, aStmt, sSqlState, &sNativeError,
                     sErrorMsg, SQL_MAX_MESSAGE_LENGTH, &sMsgLength) != SQL_SUCCESS )
        {
            return RC_FAILURE;
        }

        printf("SQLSTATE-[%s], Machbase-[%d][%s]\n", sSqlState, sNativeError, sErrorMsg);

        if( sNativeError != 9604 &&
            sNativeError != 9605 &&
            sNativeError != 9606 )
        {
            return RC_FAILURE;
        }
        else
        {
            //data value error in one record, so return success to keep attending
        }
    }
    return RC_SUCCESS;
}


void *eachThread(void *aIdx)
{
    SQLHENV    sEnv = SQL_NULL_HENV;
    SQLHDBC    sCon = SQL_NULL_HDBC;
    SQLHSTMT   sStmt[LOG_FILE_CNT] = {SQL_NULL_HSTMT,};

    FILE*      sFp;
    int        i;
    int        sLogType;

    int        sThrNo = *(int *)aIdx;

    // Alloc ENV and DBC
    if( connectDB(&sEnv, &sCon) == RC_SUCCESS )
    {
        printf("[%d]connectDB success.\n", sThrNo);
    }
    else
    {
        printf("[%d]connectDB failure.\n", sThrNo);
        goto error;
    }
```

```
// set timed flush true
if( SQLSetConnectAppendFlush(sCon, 1) != SQL_SUCCESS )
{
    printError(sEnv, sCon, NULL, "SQLSetConnectAppendFlush Error");
    goto error;
}

for( i = 0; i < LOG_FILE_CNT; i++ )
{
    // Alloc stmt
    if( SQLAllocStmt(sCon,&sStmt[i]) != SQL_SUCCESS )
    {
        printError(sEnv, sCon, sStmt[i], "SQLAllocStmt Error");
        goto error;
    }

    if( appendOpen(sEnv, sCon, sStmt[i], gTableName[i]) == RC_FAILURE )
    {
        printError(sEnv, sCon, sStmt[i], "SQLAppendOpen Error");
        goto error;
    }
    else
    {
        printf("[%d-%d]appendOpen success.\n", sThrNo, i);
    }

    if( SQLAppendSetErrorCallback(sStmt[i], appendDumpError) != SQL_SUCCESS )
    {
        printError(sEnv, sCon, sStmt[i], "SQLAppendSetErrorCallback Error");
        goto error;
    }

    // set timed flush interval as 2 seconds
    if( SQLSetStmtAppendInterval(sStmt[i], 2000) != SQL_SUCCESS )
    {
        printError(sEnv, sCon, sStmt[i], "SQLSetStmtAppendInterval Error");
        goto error;
    }
}

sFp = fopen((char*)gFileName[sThrNo], "rt");
if( sFp == NULL )
{
    printf("file open error - [%d][%s]\n", sThrNo, gFileName[sThrNo]);
}
else
{
    printf("file open success - [%d][%s]\n", sThrNo, gFileName[sThrNo]);

    for( i = 0; !feof(sFp); i++ )
    {
        fscanf(sFp, "%d ", &sLogType);
        switch(sLogType)
        {
            case 1://f1
                if( appendF1(sEnv, sCon, sStmt[0], sFp) == RC_FAILURE )
                {
                    goto error;
                }
                break;
            case 2://f2
                if( appendF2(sEnv, sCon, sStmt[1],sFp) == RC_FAILURE )
                {
                    goto error;
                }
                break;
            case 3://event
                if(appendEvent(sEnv, sCon, sStmt[2], sFp) == RC_FAILURE )
                {
                    goto error;
```

```
                    }
                    break;
                default:
                    printf("unknown type error\n");
                    break;
            }

            if( (i%10000) == 0 )
            {
                fprintf(stdout, ".");
                fflush(stdout);
            }
        }
        printf("\n");

        fclose(sFp);
    }

    for( i = 0; i < LOG_FILE_CNT; i++)
    {
        printf("[%d-%d]appendClose start...\n", sThrNo, i);
        if( appendClose(sEnv, sCon, sStmt[i]) == RC_FAILURE )
        {
            printf("[%d-%d]appendClose failure\n", sThrNo, i);
        }
        else
        {
            printf("[%d-%d]appendClose success\n", sThrNo, i);
        }

        if( SQLFreeStmt(sStmt[i], SQL_DROP) != SQL_SUCCESS )
        {
            printError(sEnv, sCon, sStmt[i], "SQLFreeStmt Error");
        }
        sStmt[i] = SQL_NULL_HSTMT;
    }

    disconnectDB(sEnv, sCon);

    printf("[%d]disconnected.\n", sThrNo);

    pthread_exit(NULL);

error:
    for( i = 0; i < LOG_FILE_CNT; i++)
    {
        if( sStmt[i] != SQL_NULL_HSTMT )
        {
            appendClose(sEnv, sCon, sStmt[i]);

            if( SQLFreeStmt(sStmt[i], SQL_DROP) != SQL_SUCCESS )
            {
                printError(sEnv, sCon, sStmt[i], "SQLFreeStmt Error");
            }
            sStmt[i] = SQL_NULL_HSTMT;
        }
    }

    if( sCon != SQL_NULL_HDBC )
    {
        disconnectDB(sEnv, sCon);
    }

    pthread_exit(NULL);
}


int initTables()
{
    SQLHENV     sEnv  = SQL_NULL_HENV;
```

```
    SQLHDBC     sCon  = SQL_NULL_HDBC;

    if( connectDB(&sEnv, &sCon) == RC_SUCCESS )
    {
        printf("connectDB success.\n");
    }
    else
    {
        printf("connectDB failure.\n");
        goto error;
    }

    if( createTables(sEnv, sCon) == RC_SUCCESS )
    {
        printf("createTables success.\n");
    }
    else
    {
        printf("createTables failure.\n");
        goto error;
    }

    disconnectDB(sEnv, sCon);

    return RC_SUCCESS;

error:

    if( sCon != SQL_NULL_HDBC )
    {
        disconnectDB(sEnv, sCon);
    }

    return RC_FAILURE;
}


int main()
{
    pthread_t sThread[MAX_THREAD_NUM];
    int       sNum[MAX_THREAD_NUM];
    int       sRC;
    int       i;

    initTables();

    //
    //eachThread has own ENV,DBC and STMT
    //
    for(i = 0; i < MAX_THREAD_NUM; i++)
    {
        sNum[i] = i;

        sRC = pthread_create(&sThread[i], NULL, (void *)eachThread, (void*)&sNum[i]);
        if ( sRC != RC_SUCCESS )
        {
            printf("Error in Thread create[%d] : %d\n", i, sRC);
            return RC_FAILURE;
        }
    }

    for(i = 0; i < MAX_THREAD_NUM; i++)
    {
        sRC = pthread_join(sThread[i], NULL);
        if( sRC != RC_SUCCESS )
        {
            printf("Error in Thread[%d] : %d\n", i, sRC);
            return RC_FAILURE;
        }
        printf("%d thread join\n", i+1);
```

```
    }

    return RC_SUCCESS;
}
```

Add the make code and run the executable file. Because the threads are used, the output order may be different. The results are as follows.

```
[mach@localhost cli]$ make sample8_multi_session_multi_table
gcc -c -g -W -Wall -rdynamic -fno-inline -m64 -mtune=k8 -g -W -Wall -rdynamic -fno-inline -m64 -mtune=k8 -I/home/ma
gcc -m64 -mtune=k8 -L/home/mach/machbase_home/lib -osample8_multi_session_multi_table sample8_multi_session_multi_t
[mach@localhost cli]$ ./sample8_multi_session_multi_table
connectDB success.
createTables success.
[0]connectDB success.
[1]connectDB success.
[2]connectDB success.
[1-0]appendOpen success.
[0-0]appendOpen success.
[2-0]appendOpen success.
[1-1]appendOpen success.
[2-1]appendOpen success.
[0-1]appendOpen success.
[1-2]appendOpen success.
[2-2]appendOpen success.
file open success - [1][suffle_data2.txt]
file open success - [2][suffle_data3.txt]
[0-2]appendOpen success.
file open success - [0][suffle_data1.txt]
.....................................................................................

[1-0]appendClose start...
..
[0-0]appendClose start...
append result success : 100000, failure : 0
[1-0]appendClose success
[1-1]appendClose start...
append result success : 100000, failure : 0
[1-1]appendClose success
[1-2]appendClose start...
append result success : 100000, failure : 0
[1-2]appendClose success
append result success : 100000, failure : 0
[0-0]appendClose success
[0-1]appendClose start...
.append result success : 100000, failure : 0
[0-1]appendClose success
[0-2]appendClose start...
append result success : 100000, failure : 0
[0-2]appendClose success

[2-0]appendClose start...
append result success : 100000, failure : 0
[2-0]appendClose success
[2-1]appendClose start...
append result success : 100000, failure : 0
[2-1]appendClose success
[2-2]appendClose start...
append result success : 100000, failure : 0
[2-2]appendClose success
[1]disconnected.
[2]disconnected.
[0]disconnected.
1 thread join
2 thread join
3 thread join
```

You can see the result through machsql as below.

```
[mach@localhost cli]$ machsql


================================================================
     Machbase Client Query Utility
     Release Version 3.5.0
     Copyright 2014, Machbase Inc. or its subsidiaries.
     All Rights Reserved.
================================================================
Machbase Server Addr (Default:127.0.0.1) :
Machbase User ID  (Default:SYS)
Machbase User Password : manager
MACH_CONNECT_MODE=INET, PORT=5656
Mach> select count(*) from table_f1;
count(*)
----------------------
300000
[1] Row Selected.
Mach> select count(*) from table_f2;
count(*)
----------------------
300000
[1] row(s) selected.
Mach> select count(*) from table_event;
count(*)
----------------------
300000
[1] row(s) selected.
```

JDBC

JDBC Overview

The set of database manipulation interfaces created in the Java programming language is called JDBC (Java DataBase Connectivity). A set of APIs that provide a consistent interface for a variety of relational databases, defining a set of object-oriented classes of classes that the programmer will use to build SQL requests. That is, if you use a JDBC driver, no matter which database you use, there is an advantage that you can apply it directly without modifying the code.

Standard JDBC Functions

Standard Function Specs 4.0

Extension JDBC Functions

setIPv4

```
void setIpv4(int ind, String ipString)
```

This is a function to input IPv4 address type in PrepareStatement.

Receives column index and IPv4 string as arguments.

setIPv6

```
void setIpv6(int ind, String ipString)
```

This is a function to input IPv6 address type in PrepareStatement.

Receives column index and IPv6 string as arguments.

ExecuteAppendOpen

```
ResultSet executeAppendOpen(String aTableName, int aErrorCheckCount)
```

Opens the protocol to write the Append protocol in the Statement.

The table name and error checking interval are received as arguments. Returns a ResultSet with the result value.

executeAppendData

```
int executeAppendData(ResultSetMetaData rsmd, ArrayList aData)
```

Enters the actual data for the Append protocol in the statement.

Receives the metadata of the ResultSet, which is the result value of executeAppendOpen, and the data to input. When the result value is stored in the transfer buffer, 1 is returned. If the transfer buffer is transferred to Machbase, 2 is returned. Therefore, if 1 or 2 is returned, it is judged as success.

executeAppendDataByTime

```
int executeAppendDataByTime(ResultSetMetaData rsmd, long aTime, ArrayList aData)
```

Enters the actual data for the Append protocol on a time basis in the statement.

Receives the metadata of the ResultSet which is the result value of executeAppendOpen, the time value of the specific time zone to be set, and the data to input as arguments. If the result value is stored in the transmission buffer, 1 is returned.

executeAppendClose

```
int executeAppendClose()
```

Terminates the statement for the Append protocol in the statement.

If the result is a success, it returns 1.

Index

executeSetAppendErrorCallback

```
int executeSetAppendErrorCallback(MachAppendCallback aCallback)
```

Sets a callback function that outputs an error if an error occurs during Append execution.

It takes a callback function that outputs an error log as an argument. If the result is successful, 1 is returned.

getAppendSuccessCount

```
long getAppendSuccessCount()
```

Returns the number of successes for the Append protocol in the Statement.

Returns the number of successful results.

getAppendFailCount

```
long getAppendFailCount()
```

Returns the number of failures for the Append protocol in the Statement.

Returns the number of failures as a result.

Application Development

JDBC Library Installation Check

Verifies that the machbase.jar file exists in the $MACHBASE_HOME/lib directory.

```
[mach@localhost ~]$ cd $MACHBASE_HOME/lib
[mach@localhost lib]$ ls -l machbase.jar
-rw-rw-r-- 1 mach mach 78599 Jun 18 10:00 machbase.jar
[mach@localhost lib]$
```

Makefile Creation Guide

$(MACHBASE_HOME)/lib/machbase.jar must be specified in the classpath. The following is an example of a Makefile.

```
CLASSPATH=".:$(MACHBASE_HOME)/lib/machbase.jar"

SAMPLE_SRC = Sample1Connect.java Sample2Insert.java Sample3PrepareStmt.java Sample4Append.java

all: build

build:
    -@rm -rf *.class
    javac -classpath $(CLASSPATH) -d . $(SAMPLE_SRC)

create_table:
    machsql -s localhost -u sys -p manager -f createTable.sql

select_table:
    machsql -s localhost -u sys -p manager -f selectTable.sql

run_sample1:
    java -classpath $(CLASSPATH) Sample1Connect

run_sample2:
    java -classpath $(CLASSPATH) Sample2Insert

run_sample3:
    java -classpath $(CLASSPATH) Sample3PrepareStmt

run_sample4:
    java -classpath $(CLASSPATH) Sample4Append
```

```
clean:
    rm -rf *.class
```

Compile and Link

Run the make command to compile and link as follows:

```
[mach@localhost jdbc]$ make
javac -classpath ".:/home/machbase/machbase_home/lib/machbase.jar" -d . Sample1Connect.java Sample2Insert.java Samp
[mach@localhost jdbc]$
```

JDBC Sample

Connection Example

Let's write an example program that connects to a Machbase server using a Machbase JDBC driver. Name the source file Sample1Connect.java.

> The _arrival_time column is not displayed by default. Therefore, to display the _arrival_time column, add show_hidden_cols = 1 to the connection string.
>
> You can modify the connection string in the following example source as follows:
> String sURL = "jdbc:machbase://localhost:5656/mhdb?show_hidden_cols=1";

```
import java.util.*;
import java.sql.*;
import mach.jdbc.driver.*;

public class Sample1Connect
{
    public static Connection connect()
    {
        Connection conn = null;
        try
        {
            String sURL = "jdbc:machbase://localhost:5656/mhdb";

            Properties sProps = new Properties();
            sProps.put("user", "sys");
            sProps.put("password", "manager");

            Class.forName("com.machbase.jdbc.driver");
            conn = DriverManager.getConnection(sURL, sProps);
        }
        catch ( ClassNotFoundException ex )
        {
            System.err.println("Exception : unable to load mach jdbc driver class");
        }
        catch ( Exception e )
        {
            System.err.println("Exception : " + e.getMessage());
        }
        return conn;
    }

    public static void main(String[] args) throws Exception
    {
        Connection conn = null;

        try
        {
            conn = connect();
            if( conn != null )
            {
                System.out.println("mach JDBC connected.");
            }
```

```
        }
        catch( Exception e )
        {
            System.err.println("Exception : " + e.getMessage());
        }
        finally
        {
            if( conn != null )
            {
                conn.close();
                conn = null;
            }
        }
    }
}
```

Now compile and run the source code. Use the Makefile you have already created.

```
[mach@localhost jdbc]$ make
javac -classpath ".:/home/machbase/machbase_home/lib/machbase.jar" -d . Sample1Connect.java Sample2Insert.java Samp
[mach@localhost jdbc]$ make run_sample1
java -classpath ".:/home/machbase/machbase_home/lib/machbase.jar" Sample1Connect
mach JDBC connected.
```

Data Input and Output Example (1) Direct I/O

Create and display an example that uses the Machbase JDBC driver to input and output data.

The name of the source file is called Sample2Insert.java.
First, you need to create the necessary tables using the machsql program.
In the example, we used the sample code to create a table called sample_table in advance.

```
[mach@localhost jdbc]$ machsql
=================================================================
     Machbase Client Query Utility
     Release Version 3.5.0.826b8f2.official
     Copyright 2014, Machbase Inc. or its subsidiaries.
     All Rights Reserved.
=================================================================
Machbase server address (Default:127.0.0.1):
Machbase rser ID  (Default:SYS)
Machbase user password: MANAGER
MACHBASE_CONNECT_MODE=INET, PORT=5656
mach> create table sample_table(d1 short, d2 integer, d3 long, f1 float, f2 double, name varchar(20), text text, bi
Created successfully.
mach> exit
[mach@localhost jdbc]$
```

```
import java.util.*;
import java.sql.*;
import mach.jdbc.driver.*;

public class Sample2Insert
{
    public static Connection connect()
    {
        Connection conn = null;
        try
        {

            String sURL = "jdbc:machbase://localhost:5656/mhdb";

            Properties sProps = new Properties();
            sProps.put("user", "sys");
            sProps.put("password", "manager");
```

```
            Class.forName("com.machbase.jdbc.driver");

            conn = DriverManager.getConnection(sURL, sProps);

        }
        catch ( ClassNotFoundException ex )
        {
            System.err.println("Exception : unable to load mach jdbc driver class");
        }
        catch ( Exception e )
        {
            System.err.println("Exception : " + e.getMessage());
        }

        return conn;
    }


    public static void main(String[] args) throws Exception
    {
        Connection conn = null;
        Statement stmt = null;
        String sql;

        try
        {
            conn = connect();
            if( conn != null )
            {
                System.out.println("mach JDBC connected.");

                stmt = conn.createStatement();

                for(int i=1; i<10; i++)
                {
                    sql = "INSERT INTO SAMPLE_TABLE VALUES (";
                    sql += (i - 5) * 6552;//short
                    sql += ","+ ((i - 5) * 429496728);//integer
                    sql += ","+ ((i - 5) * 92233720368547758O);//long
                    sql += ","+ 1.23456789+"e"+((i<=5)?"":"+")+((i-5)*7);//float
                    sql += ","+ 1.23456789+"e"+((i<=5)?"":"+")+((i-5)*61);//double
                    sql += ",'id-"+i+"'";//varchar
                    sql += ",'name-"+i+"'";//text
                    sql += ",'aabbccddeeff'";//binary
                    sql += ",'192.168.0."+i+"'";//ipv4
                    sql += ",'::192.168.0."+i+"'";
                    sql += ",TO_DATE('2014-08-0"+i+"','YYYY-MM-DD')";//dt
                    sql += ")";

                    stmt.execute(sql);

                    System.out.println( i+" record inserted.");
                }

                String query = "SELECT d1, d2, d3, f1, f2, name, text, bin, to_hex(bin), v4, v6, to_char(dt,'YYYY-M
                ResultSet rs = stmt.executeQuery(query);
                while( rs.next () )
                {
                    short d1 = rs.getShort("d1");
                    int d2 = rs.getInt("d2");
                    long d3 = rs.getLong("d3");
                    float f1 = rs.getFloat("f1");
                    double f2 = rs.getDouble("f2");
                    String name = rs.getString("name");
                    String text = rs.getString("text");
                    String bin = rs.getString("bin");
                    String hexbin = rs.getString("to_hex(bin)");
                    String v4 = rs.getString("v4");
                    String v6 = rs.getString("v6");
```

```
                    String dt = rs.getString("dt");

                    System.out.print("d1: " + d1);
                    System.out.print(", d2: " + d2);
                    System.out.print(", d3: " + d3);
                    System.out.print(", f1: " + f1);
                    System.out.print(", f2: " + f2);
                    System.out.print(", name: " + name);
                    System.out.print(", text: " + text);
                    System.out.print(", bin: " + bin);
                    System.out.print(", hexbin: "+hexbin);
                    System.out.print(", v4: " + v4);
                    System.out.print(", v6: " + v6);
                    System.out.println(", dt: " + dt);

                }
                rs.close();
            }
        }
        catch( SQLException se )
        {
            System.err.println("SQLException : " + se.getMessage());
        }
        catch( Exception e )
        {
            System.err.println("Exception : " + e.getMessage());
        }
        finally
        {
            if( stmt != null )
            {
                stmt.close();
                stmt = null;
            }
            if( conn != null )
            {
                conn.close();
                conn = null;
            }
        }
    }
}
```

Now compile and run the source code. Use the Makefile you have already created.

```
[mach@localhost jdbc]$ make
javac -classpath ".:/home/machbase/machbase_home/lib/machbase.jar" -d . Sample1Connect.java Sample2Insert.java Samp
[mach@localhost jdbc]$ make run_sample2
make run_sample2
java -classpath ".:/home/machbase/machbase_home/lib/machbase.jar" Sample2Insert
mach JDBC connected.
1 record inserted.
2 record inserted.
3 record inserted.
4 record inserted.
5 record inserted.
6 record inserted.
7 record inserted.
8 record inserted.
9 record inserted.
d1: 26208, d2: 1717986912, d3: 3689348814741910320, f1: 1.2345679E28, f2: 1.23456789E244, name: id-9, text: name-9,
d1: 19656, d2: 1288490184, d3: 2767011611056432740, f1: 1.2345678E21, f2: 1.23456789E183, name: id-8, text: name-8,
d1: 13104, d2: 858993456, d3: 1844674407370955160, f1: 1.23456788E14, f2: 1.23456789E122, name: id-7, text: name-7,
d1: 6552, d2: 429496728, d3: 922337203685477580, f1: 1.2345679E7, f2: 1.23456789E61, name: id-6, text: name-6, bin
d1: 0, d2: 0, d3: 0, f1: 1.2345679, f2: 1.23456789, name: id-5, text: name-5, bin: aabbccddeeff, hexbin: 616162626
d1: -6552, d2: -429496728, d3: -922337203685477580, f1: 1.2345679E-7, f2: 1.23456789E-61, name: id-4, text: name-4,
d1: -13104, d2: -858993456, d3: -1844674407370955160, f1: 1.2345679E-14, f2: 1.23456789E-122, name: id-3, text: nam
d1: -19656, d2: -1288490184, d3: -2767011611056432740, f1: 1.2345679E-21, f2: 1.23456789E-183, name: id-2, text: na
```

```
d1: -26208, d2: -1717986912, d3: -3689348814741910320, f1: 1.2345679E-28, f2: 1.23456789E-244, name: id-1, text: na
```

Data Input and Output Example (2) PreparedStatement Input Used

Create and view an example that uses a PreparedStatement to input and output data.

The name of the source file is Sample3PrepareStmt.java.

```
import java.util.*;
import java.sql.*;
import java.text.SimpleDateFormat;
import mach.jdbc.driver.*;

public class Sample3PrepareStmt
{
    public static Connection connect()
    {
        Connection conn = null;
        try
        {
            String sURL = "jdbc:machbase://localhost:5656/mhdb";

            Properties sProps = new Properties();
            sProps.put("user", "sys");
            sProps.put("password", "manager");

            Class.forName("com.machbase.jdbc.driver");

            conn = DriverManager.getConnection(sURL, sProps);

        }
        catch ( ClassNotFoundException ex )
        {
            System.err.println("Exception : unable to load mach jdbc driver class");
        }
        catch ( Exception e )
        {
            System.err.println("Exception : " + e.getMessage());
        }
        return conn;
    }

    public static void main(String[] args) throws Exception
    {
        Connection conn = null;
        Statement stmt = null;
        machPreparedStatement preStmt = null;
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss SSS");

        try
        {
            conn = connect();
            if( conn != null )
            {
                System.out.println("mach JDBC connected.");

                stmt = conn.createStatement();
                preStmt = (machPreparedStatement)conn.prepareStatement("INSERT INTO SAMPLE_TABLE VALUES(?, ?, ?, ?,

                String ipStr = null;
                String dateStr = null;
                for(int i=1; i<10; i++)
                {
                    ipStr = String.format("172.16.0.%d",i);
                    dateStr = String.format("2014-08-09 12:23:34 %03d", i);
                    byte[] bin = new byte[20];
                    for(int j=0;j<20;j++){
                        bin[j]=(byte)(Math.random()*255);
```

```
                }
                java.util.Date day = sdf.parse(dateStr);
                java.sql.Date sqlDate = new java.sql.Date(day.getTime());

                preStmt.setShort(1, (i-5) * 3276 );
                preStmt.setInt(2, (i-5) * 214748364 );
                preStmt.setLong(3, (i-5) * 922337203685477580L );
                preStmt.setFloat(4, 1.23456789101112131415*Math.pow(10,i));
                preStmt.setDouble(5, 1.23456789101112131415*Math.pow(10,i*10));
                preStmt.setString(6, String.format("varchar-%d",i));
                preStmt.setString(7, String.format("text-%d",i));
                preStmt.setBytes(8, bin);
                preStmt.setIpv4(9, ipStr);
                preStmt.setIpv6(10, "::"+ipStr);
                preStmt.setDate(11, sqlDate);
                preStmt.executeUpdate();

                System.out.println( i+" record inserted.");
            }

            //date type format : YYYY-MM-DD HH24:MI:SS mmm:uuu:nnnn
            String query = "SELECT d1, d2, d3, f1, f2, name, text, bin, to_hex(bin), v4, v6, to_char(dt,'YYYY-M
            ResultSet rs = stmt.executeQuery(query);
            while( rs.next () )
            {
                short d1 = rs.getShort("d1");
                int d2 = rs.getInt("d2");
                long d3 = rs.getLong("d3");
                float f1 = rs.getFloat("f1");
                double f2 = rs.getDouble("f2");
                String name = rs.getString("name");
                String text = rs.getString("text");
                String bin = rs.getString("bin");
                String hexbin = rs.getString("to_hex(bin)");
                String v4 = rs.getString("v4");
                String v6 = rs.getString("v6");
                String dt = rs.getString("dt");

                System.out.print("d1: " + d1);
                System.out.print(", d2: " + d2);
                System.out.print(", d3: " + d3);
                System.out.print(", f1: " + f1);
                System.out.print(", f2: " + f2);
                System.out.print(", name: " + name);
                System.out.print(", text: " + text);
                System.out.print(", bin: " + bin);
                System.out.print(", hexbin: "+hexbin);
                System.out.print(", v4: " + v4);
                System.out.print(", v6: " + v6);
                System.out.println(", dt: " + dt);
            }
            rs.close();
        }
    }
    catch( SQLException se )
    {
        System.err.println("SQLException : " + se.getMessage());
    }
    catch( Exception e )
    {
        System.err.println("Exception : " + e.getMessage());
    }
    finally
    {
        if( stmt != null )
        {
            stmt.close();
            stmt = null;
        }
        if( conn != null )
```

```
            {
                conn.close();
                conn = null;
            }
        }
    }
}
```

Now compile and run the source code. Use the Makefile you have already created.

It should be noted that the data entered in Sample2Insert.java is output together.

```
[mach@localhost jdbc]$ make
javac -classpath ".:/home/machbase/machbase_home/lib/machbase.jar" -d . Sample1Connect.java
Sample2Insert.java Sample3PrepareStmt.java Sample4Append.java
[mach@localhost jdbc]$ make run_sample3
make run_sample3
java -classpath ".:/home/machbase/machbase_home/lib/machbase.jar" Sample3PrepareStmt
Mach JDBC connected.
1 record inserted.
2 record inserted.
3 record inserted.
4 record inserted.
5 record inserted.
6 record inserted.
7 record inserted.
8 record inserted.
9 record inserted.
d1: 13104, d2: 858993456, d3: 3689348814741910320, f1: 754454.6, f2: 453821.380752063, name:
varchar-9, text: text-9, bin: ?+,??r?J?????S)n?, hexbin:
A4C9A8D491D6728B4AACB39EE5FC5300296EFA9F, v4: 172.16.0.9, v6: 0:0:0:0:0:0:ac10:9, dt:
2014-08-09 12:23:34 009:000:000
?h???a?, hexbin: 6C20F09329ABBA3E7DE501C30DA368D6EFC961EF, v4: 172.16.0.8, v6:
0:0:0:0:0:0:ac10:8, dt: 2014-08-09 12:23:34 008:000:000
d1: 6552, d2: 429496728, d3: 1844674407370955160, f1: 2664182.0, f2: 1357910.1926900472, name:
varchar-7, text: text-7, bin: ????Uls?q?H?I?&(?, hexbin:
B5A0A2EFA185556C73BF719448BD49C92628F8C6, v4: 172.16.0.7, v6: 0:0:0:0:0:0:ac10:7, dt:
2014-08-09 12:23:34 007:000:000
d1: 3276, d2: 214748364, d3: 922337203685477580, f1: 443847.1, f2: 9342855.256576871, name:
varchar-6, text: text-6, bin: ??>x??Eu?? ?Iw??+n, hexbin:
BC973E78F5B44575D6CC15F94977DAE62B6E1D0E, v4: 172.16.0.6, v6: 0:0:0:0:0:0:ac10:6, dt:
2014-08-09 12:23:34 006:000:000
d1: 0, d2: 0, d3: 0, f1: 1283723.1, f2: 1771261.2019240903, name: varchar-5, text: text-5,
bin: &== j?j3?? T??y?
??, hexbin: 263D3D1C6AF56A33F79D0C54A5C479A4030AFE8B, v4: 172.16.0.5, v6: 0:0:0:0:0:0:ac10:5,
dt: 2014-08-09 12:23:34 005:000:000
d1: -3276, d2: -214748364, d3: -922337203685477580, f1: 9447498.0, f2: 7529392.937964935,
name: varchar-4, text: text-4, bin: ?Sw ??)? ?h2?E??/?, hexbin:
C653771DD2DF29CDB30ED96832E745D3D7A52FD2, v4: 172.16.0.4, v6: 0:0:0:0:0:0:ac10:4, dt:
2014-08-09 12:23:34 004:000:000
d1: -6552, d2: -429496728, d3: -1844674407370955160, f1: 9589634.0, f2: 5994172.201347323,
name: varchar-3, text: text-3, bin: 9aB,.????L/?=3,?`?f, hexbin:
3961422C2EA39BE6F2964C2FCD3D332C8960A466, v4: 172.16.0.3, v6: 0:0:0:0:0:0:ac10:3, dt:
2014-08-09 12:23:34 003:000:000
d1: -9828, d2: -644245092, d3: -2767011611056432740, f1: 7409537.5, f2: 2313739.6613546023,
name: varchar-2, text: text-2, bin: _? N?3 ?? ??~H ??= 8, hexbin:
5F84144EF63320F3C718B0FD7E4809A4CB3D1838, v4: 172.16.0.2, v6: 0:0:0:0:0:0:ac10:2, dt:
2014-08-09 12:23:34 002:000:000
d1: -13104, d2: -858993456, d3: -3689348814741910320, f1: 596626.75, f2: 2649492.1936065694,
name: varchar-1, text: text-1, bin: ???d??Wu$v? 7m?-, hexbin:
E8D0C564B4EB57E59B08752476FC07376DBF2D14, v4: 172.16.0.1, v6: 0:0:0:0:0:0:ac10:1, dt:
2014-08-09 12:23:34 001:000:000
d1: 26208, d2: 1717986912, d3: 3689348814741910320, f1: 1.2345679E28, f2: 1.23456789E244,
name: id-9, text: name-9, bin: aabbccddeeff, hexbin: 616162626363646465656666, v4:
192.168.0.9, v6: 0:0:0:0:0:0:c0a8:9, dt: 2014-08-09 00:00:00 000:000:000
d1: 19656, d2: 1288490184, d3: 2767011611056432740, f1: 1.2345678E21, f2: 1.23456789E183,
name: id-8, text: name-8, bin: aabbccddeeff, hexbin: 616162626363646465656666, v4:
192.168.0.8, v6: 0:0:0:0:0:0:c0a8:8, dt: 2014-08-08 00:00:00 000:000:000
d1: 13104, d2: 858993456, d3: 1844674407370955160, f1: 1.23456788E14, f2: 1.23456789E122,
name: id-7, text: name-7, bin: aabbccddeeff, hexbin: 616162626363646465656666, v4:
```

```
192.168.0.7, v6: 0:0:0:0:0:0:c0a8:7, dt: 2014-08-07 00:00:00 000:000:000
d1: 6552, d2: 429496728, d3: 922337203685477580, f1: 1.2345679E7, f2: 1.23456789E61, name:
id-6, text: name-6, bin: aabbccddeeff, hexbin: 616162626363646465656666, v4: 192.168.0.6, v6:
0:0:0:0:0:0:c0a8:6, dt: 2014-08-06 00:00:00 000:000:000
d1: 0, d2: 0, d3: 0, f1: 1.2345679, f2: 1.23456789, name: id-5, text: name-5, bin:
aabbccddeeff, hexbin: 616162626363646465656666, v4: 192.168.0.5, v6: 0:0:0:0:0:0:c0a8:5, dt:
2014-08-05 00:00:00 000:000:000
d1: -6552, d2: -429496728, d3: -922337203685477580, f1: 1.2345679E-7, f2: 1.23456789E-61,
name: id-4, text: name-4, bin: aabbccddeeff, hexbin: 616162626363646465656666, v4:
192.168.0.4, v6: 0:0:0:0:0:0:c0a8:4, dt: 2014-08-04 00:00:00 000:000:000
d1: -13104, d2: -858993456, d3: -1844674407370955160, f1: 1.2345679E-14, f2: 1.23456789E-122,
name: id-3, text: name-3, bin: aabbccddeeff, hexbin: 616162626363646465656666, v4:
192.168.0.3, v6: 0:0:0:0:0:0:c0a8:3, dt: 2014-08-03 00:00:00 000:000:000
d1: -19656, d2: -1288490184, d3: -2767011611056432740, f1: 1.2345679E-21, f2: 1.23456789E-183,
name: id-2, text: name-2, bin: aabbccddeeff, hexbin: 616162626363646465656666, v4:
192.168.0.2, v6: 0:0:0:0:0:0:c0a8:2, dt: 2014-08-02 00:00:00 000:000:000
d1: -26208, d2: -1717986912, d3: -3689348814741910320, f1: 1.2345679E-28, f2: 1.23456789E-244,
name: id-1, text: name-1, bin: aabbccddeeff, hexbin: 616162626363646465656666, v4:
192.168.0.1, v6: 0:0:0:0:0:0:c0a8:1, dt: 2014-08-01 00:00:00 000:000:000
```

Extension Function Append Example

The Machbase JDBC driver supports the Append protocol to quickly upload large numbers of data.

The following is an example of using the Append protocol.
Use the sample_table used in the previous example.
The name of the source file is called Sample4Append.java.
Enter the contents of data.txt into sample_table.
Copy the data.txt file used in the CLI append example.

```java
import java.util.*;
import java.sql.*;
import java.io.*;
import java.text.SimpleDateFormat;
import java.math.BigDecimal;
import mach.jdbc.driver.*;


public class Sample4Append
{
    protected static final String sTableName = "sample_table";
    protected static final int sErrorCheckCount = 100;

    public static Connection connect()
    {
        Connection conn = null;
        try
        {
            String sURL = "jdbc:machbase://localhost:5656/mhdb";

            Properties sProps = new Properties();
            sProps.put("user", "sys");
            sProps.put("password", "manager");

            Class.forName("com.machbase.jdbc.driver");

            conn = DriverManager.getConnection(sURL, sProps);

        }
        catch ( ClassNotFoundException ex )
        {
            System.err.println("Exception : unable to load mach jdbc driver class");
        }
        catch ( Exception e )
        {
            System.err.println("Exception : " + e.getMessage());
        }
        return conn;
    }
```

```
public static void main(String[] args) throws Exception
{
    Connection conn = null;
    MachStatement stmt = null;
    SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
    Calendar cal = Calendar.getInstance();
    String filename = "data.txt";

    try
    {
        conn = connect();
        if( conn != null )
        {
            System.out.println("Mach JDBC connected.");

            stmt = (MachStatement)conn.createStatement();

            ResultSet rs = stmt.executeAppendOpen(sTableName, sErrorCheckCount);
            ResultSetMetaData rsmd = rs.getMetaData();

            System.out.println("append open ok");

            MachAppendCallback cb = new MachAppendCallback() {
                    @Override
                    public void onAppendError(long aErrNo, String aErrMsg, String aRowMsg) {
                         System.out.format("Append Error : [%05d - %s]\n%s\n", aErrNo, aErrMsg, aRowMsg);
                    }
                };

            stmt.executeSetAppendErrorCallback(cb);

            System.out.println("append data start");
            BufferedReader in = new BufferedReader(new FileReader(filename));
            String buf = null;
            int cnt = 0;
            long dt;

            long startTime = System.nanoTime();

            while( (buf = in.readLine()) != null )
            {
                ArrayList<Object> sBuf = new ArrayList<Object>();
                StringTokenizer st = new StringTokenizer(buf,",");
                for(int i=0; st.hasMoreTokens() ;i++ )
                {
                    switch(i){
                        case 7://binary case
                            sBuf.add(new ByteArrayInputStream(st.nextToken().getBytes())); break;
                        case 10://date case
                            java.util.Date day = sdf.parse(st.nextToken());
                            cal.setTime(day);
                            dt = cal.getTimeInMillis()*1000000; //make nanotime
                            sBuf.add(dt);
                            break;
                        default:
                            sBuf.add(st.nextToken()); break;
                    }
                }

                if( stmt.executeAppendData(rsmd, sBuf) != 1 )
                {
                    System.err.println("Error : AppendData error");
                    break;
                }

                if( (cnt++%10000) == 0 )
                {
                    System.out.print(".");
                }
                sBuf = null;
```

```
                }
                System.out.println("\nappend data end");

                long endTime = System.nanoTime();
                stmt.executeAppendClose();
                System.out.println("append close ok");
                System.out.println("Append Result : success = "+stmt.getAppendSuccessCount()+", failure = "+stmt.ge
                System.out.println("timegap " + ((endTime - startTime)/1000) + " in microseconds, " + cnt + " reco

                try {
                    BigDecimal records = new BigDecimal( cnt );
                    BigDecimal gap = new BigDecimal( (double)(endTime - startTime)/1000000000 );
                    BigDecimal rps = records.divide(gap, 2, BigDecimal.ROUND_UP );

                    System.out.println( rps + " records/second" );
                } catch(ArithmeticException ae) {
                    System.out.println( cnt + " records/second");
                }

                rs.close();
            }
        }
        catch( SQLException se )
        {
            System.err.println("SQLException : " + se.getMessage());
        }
        catch( Exception e )
        {
            System.err.println("Exception : " + e.getMessage());
        }
        finally
        {
            if( stmt != null )
            {
                stmt.close();
                stmt = null;
            }
            if( conn != null )
            {
                conn.close();
                conn = null;
            }
        }
    }
}
```

When appending, date type data must be converted to long type nanosecond time.

```
[mach@localhost jdbc]$ make run_sample4
make run_sample4
java -classpath ".:/home/machbase/machbase_home/lib/machbase.jar" Sample4Append;
Mach JDBC connected.
append open ok
append data start
......
append data end
append close ok
Append Result : success = 60000, failure = 0
timegap 6905594 in microseconds, 60000 records
8688.61 records/second
```

Displays the dot (.) every 10,000, and can know the input time.

```
# Use machsql to check number actually entered.
# Confirm that 60018 are entered including those from Sample2Insert and Sample3PrepareStmt.
```

```
[mach@localhost jdbc]$ machsql
=================================================================
     Machbase Client Query Utility
     Release Version 3.0.0
     Copyright 2014, Machbase Inc. or its subsidiaries.
     All Rights Reserved.
=================================================================
Machbase server address (Default:127.0.0.1):
Machbase user ID  (Default:SYS)
Machbase user password: MANAGER
MACH_CONNECT_MODE=INET, PORT=5656
mach> select count(*) from sample_table;
count(*)
-----------------------
60018
[1] row(s) selected.
```

Python

Python Module Usage Overview

Machbase supports Python modules. By installing the module, it provides a class that can exchange values with the Machbase server and the CLI method. You can use this to easily enter and delete values in the query base in Python, create and delete tables, and so on.

Preferences

Simple configuration and library files are required to use this. First, make sure that $MAC_LIBRARY_PATH is registered in the $MACHBASE_HOME/lib directory. The libmachbasecli_dll.so file must exist in the library folder because you are accessing Machbase using the CLI. Then unzip machbaseAPI-1.0.tar.gz in the $MACHBASE_HOME/3rd-party/python-module folder and install the module into the Python you want to use via the python setup.py install command. Python recommends version 2.7. Now you can use the Machbase Python module.

Creating Class

To use the Machbase Python module, you need to declare the class.

The class name is machbase.

```
from machbaseAPI import machbase
```

Index

Connection and Disconnection

machbase.open(aHost, aUser, aPw, aPort)

This is a function to connect to Machbase. When the appropriate parameter value is input, it returns whether connection to the DB is successful or failed. Returns 1 on success and 0 on failure.

machbase.close()

This is a function to close the Machbase connection. Returns 1 on success and 0 on failure.

machbase.isConnected()

This is a function that determines whether the declared class is connected to the server. Returns 1 when connected and 0 when not connected.

Executing Commands and User Convenience Functions

machbase.execute(aSql)

This is a command to send a query to the server when it is connected to the server. It returns 1 when it is normally executed, and 0 when it fails or an error occurs.

You can use any command except UPDATE which is not supported by Machbase.

machbase.append(aTableName, aTypes, aValues, aFormat)

This is a function that can use Append protocol supported by Machbase.

Append can be executed by inputting the table name, the dictionary of the type of each column, and the values to input in JSON format and specifying dateformat.
Returns 1 if executed normally, 0 otherwise.

Type Name	Value
short	4
ushort	104
integer	8
uinteger	108
long	12
ulong	112
float	16

Type Name	Value
double	20
datetime	6
varchar	5
ipv4	32
ipv6	36
text	49
binary	97

machbase.tables()

Returns information about all tables in the connected server. Returns 1 if successful; 0 if unsuccessful.

machbase.columns(aTableName)

Returns information about the columns in the corresponding table on the connected server. Returns 1 if successful; 0 if unsuccessful.

Checking Results

In the Machbase Python module, all result values are returned in JSON.

It is adopted to return the result in a form that is easy to use in various environments.

machbase.result()

The functions described above do not represent the execution results as the return value of the function, but only return success or failure. The result of a function can be obtained only by the return value of this function.

Examples

Let's see how to use the Machbase Python module with simple examples.

You can check using $MACHBASE_HOME/sample/python files. The directory has a Makefile that makes it easy to test and a MakeData.py file that creates the data. Make sure that the value of PYPATH in the Makefile is set to Python with the Machbase Python module installed. The default is specified in Python installed in the Machbase package. Also, you need to make __init__.py file to execute the module in Python independently, so make sure that the file exists in that directory.

```
[mach@localhost]$ cd $MACHBASE_HOME/sample/python
[mach@localhost python]$ ls -l
total 20
-rw-rw-r-- 1 mach mach    0 Oct  7 14:37 __init__.py
-rw-rw-r-- 1 mach mach  764 Oct  7 14:37 MakeData.py
-rw-rw-r-- 1 mach mach  593 Oct  7 14:58 Makefile
-rw-rw-r-- 1 mach mach  664 Oct  7 14:37 Sample1Connect.py
-rw-rw-r-- 1 mach mach 2401 Oct  7 14:37 Sample2Simple.py
-rw-rw-r-- 1 mach mach 1997 Oct  7 14:37 Sample3Append.py
```

Connect

The following example is a simple function that connects to the server, executes the query, and returns the result. If each function returns a failure value (0), it returns an error result. If successful, the number of values in the m $ tables table is returned.

The file name is Sample1Connect.py.

```
from machbaseAPI import machbase
def connect():
    db = machbase()
    if db.open('127.0.0.1','SYS','MANAGER',5656) is 0 :
        return db.result()
    if db.execute('select count(*) from m$tables') is 0 :
        return db.result()
    result = db.result()
    if db.close() is 0 :
        return db.result()
    return result
```

```
if __name__=="__main__":
    print connect()
```

```
[mach@localhost python]$ make run_sample1
/home/machbase/machbase_home/webadmin/flask/Python/bin/python Sample1Connect.py
{"count(*)":"13"}
```

Simple

Using the example below, we simply create a table using Python in Machbase, input the value, and extract the input value to check. The file name is Sample2Simple.py.

```
import re
import json
from machbaseAPI import machbase
def insert():
    db = machbase()
    if db.open('127.0.0.1','SYS','MANAGER',5656) is 0 :
        return db.result()
    db.execute('drop table sample_table')
    db.result()
    if db.execute('create table sample_table(d1 short, d2 integer, d3 long, f1 float, f2 double, name varchar(20),
        return db.result()
    db.result()
    for i in range(1,10):
        sql = "INSERT INTO SAMPLE_TABLE VALUES ("
        sql += str((i - 5) * 6552) #short
        sql += ","+ str((i - 5) * 42949672) #integer
        sql += ","+ str((i - 5) * 92233720368547758L) #long
        sql += ","+ "1.234"+str((i-5)*7) #float
        sql += ","+ "1.234"+str((i-5)*61) #double
        sql += ",'id-"+str(i)+"'" #varchar
        sql += ",'name-"+str(i)+"'" #text
        sql += ",'aabbccddeeff'" #binary
        sql += ",'192.168.0."+str(i)+"'" #ipv4
        sql += ",'::192.168.0."+str(i)+"'" #ipv6
        sql += ",TO_DATE('2015-08-0"+str(i)+"','YYYY-MM-DD')" #date
        sql += ")";
        if db.execute(sql) is 0 :
            return db.result()
        else:
            print db.result()
        print str(i)+" record inserted."
    query = "SELECT d1, d2, d3, f1, f2, name, text, bin, to_hex(bin), v4, v6, to_char(dt,'YYYY-MM-DD') as dt from S
    if db.execute(query) is 0 :
        return db.result()
    result = db.result()
    for item in re.findall('{[^}]+}',result):
        res = json.loads(item)
        print "d1 : "+res.get('d1')
        print "d2 : "+res.get('d2')
        print "d3 : "+res.get('d3')
        print "f1 : "+res.get('f1')
        print "f2 : "+res.get('f2')
        print "name : "+res.get('name')
        print "text : "+res.get('text')
        print "bin : "+res.get('bin')
        print "to_hex(bin) : "+res.get('to_hex(bin)')
        print "v4 : "+res.get('v4')
        print "v6 : "+res.get('v6')
        print "dt : "+res.get('dt')
    if db.close() is 0 :
        return db.result()
    return result
if __name__=="__main__":
    print insert()
```

```
[mach@loclahost python]$ make run_sample2
/home/machbase/machbase_home/webadmin/flask/Python/bin/python Sample2Simple.py
{"EXECUTE RESULT":"Execute Success"}
1 record inserted.
{"EXECUTE RESULT":"Execute Success"}
2 record inserted.
{"EXECUTE RESULT":"Execute Success"}
3 record inserted.
{"EXECUTE RESULT":"Execute Success"}
4 record inserted.
{"EXECUTE RESULT":"Execute Success"}
5 record inserted.
{"EXECUTE RESULT":"Execute Success"}
6 record inserted.
{"EXECUTE RESULT":"Execute Success"}
7 record inserted.
{"EXECUTE RESULT":"Execute Success"}
8 record inserted.
{"EXECUTE RESULT":"Execute Success"}
9 record inserted.
d1 : 26208
d2 : 171798688
d3 : 368934881474191032
f1 : 1.23428
f2 : 1.23424
name : id-9
text : name-9
bin : 616162626363646465656666
to_hex(bin) : 6161626263636464656566666
v4 : 192.168.0.9
v6 : ::192.168.0.9
...
```

Append

Append method to input data at high speed in Machbase can also be used by using Python module. The following example shows how to input data at high speed. We used a method of declaring a connection class db2 for the column information and initialization, and a connection class db2 for the Append, and using each function. The file name is Sample3Append.py.

```
import re
import json
from machbaseAPI import machbase
def append():
#init,columns start
    db = machbase()
    if db.open('127.0.0.1','SYS','MANAGER',5656) is 0 :
        return db.result()
    db.execute('drop table sample_table')
    db.result()
    if db.execute('create table sample_table(d1 short, d2 integer, d3 long, f1 float, f2 double, name varchar(20),
        return db.result()
    db.result()
    tableName = 'sample_table'
    db.columns(tableName)
    result = db.result()
    if db.close() is 0 :
        return db.result()
#init, colums end
#append start
    db2 = machbase()
    if db2.open('127.0.0.1','SYS','MANAGER',5656) is 0 :
        return db2.result()
    types = []
    for item in re.findall('{[^}]+}',result):
        types.append(json.loads(item).get('type'))
    values = []
    with open('data.txt','r') as f:
```

```
        for line in f.readlines():
            v = []
            i = 0
            for l in line[:-1].split(','):
                t = int(types[i])
                if t == 4 or t == 8 or t == 12 or t == 104 or t == 108 or t == 112:
                    #short   integer    long       ushort      uinteger     ulong
                    v.append(int(l))
                elif t == 16 or t == 20:
                    #float      double
                    v.append(float(l))
                else:
                    v.append(l)
                i+=1
            values.append(v)
    db2.append(tableName, types, values, 'YYYY-MM-DD HH24:MI:SS')
    result = db2.result()
    if db2.close() is 0 :
        return db2.result()
#append end
    return result
if __name__=="__main__":
    print append()
```

```
[mach@localhost python]$ make run_sample3
/home/machbase/machbase_home/webadmin/flask/Python/bin/python Sample3Append.py
{"EXECUTE RESULT":"Append success"}
```

RESTful API

RESTful API Overview

Representational State Transfer (REST) is a type of software architecture style that consists of guidelines and best practices for interfaces provided by scalable Web services.

The four methods defined in the HTTP protocol define the CRUD for the resource.

HTTP Method	Meaning
POST	Create
GET	Select
PUT	Update
DELETE	Delete

Machbase is not a standard RESTful API, but rather a RESTful API that handles CRUD using only POST and GET methods.

That is, the POST method is used for data input and the rest is transmitted as a GET Method parameter to the SQL query so that all the operations can be performed.

Index

Using Embedded web server (version 6.5 and after)

From version 6.5, MachBase supports convenient and fast Rest API functions through the web server built into the server.

Machbase Edition support embedded web server

All type of machbase editions are supported. (Edge / Fog / Cluster)

Location of version-specific .conf files

Edge/Fog version

$MACHBASE_HOME/conf/machbase.conf $MACHBASE_HOME/http/conf/http.conf

Cluster version

$EACH_BROKER_HOME/conf/machbase.conf (Modify by Broker) $EACH_BROKER_HOME/http/conf/http.conf (modify all per Broker)

Added properties for embedded web server

machbase.conf (set as PROPERTY = VALUE)

Property	Description
HTTP_ENABLE	Whether to run the embedded web server 0: not driven, 1: driven
HTTP_PORT_NO	Embedded web server connection port number Port range: 0 ~ 65535 Default : 5657
HTTP_MAX_MEM	Maximum memory used by one Web Session Min: 1048576 (1MB) Default : 536870912 (512MB
HTTP_AUTH	Whether to use Basic authentication when using the Embedded Web Server 0: Authentication not used, 1: Authentication enabled

http.conf (set in JSON format)

Property	Description
document_root	html file location based on $MACHBASE_HOME Default : http/html ($MACHBASE_HOME/http/html)
max_request_size	Limit the maximum request byte size for one request
request_timeout_ms	Maximum response latency for one request (millisecond)

Property	Description
enable_auth_domain_check	Whether to enable domain authentication Set to "yes" or "no" value Default: "no"
reverse_proxy	change request url to specific url

sample conf files

machbase.conf

machbase.conf

```
##############################################################################
# Rest-API port
##############################################################################
HTTP_PORT_NO = 5657

##############################################################################
# Maximum memory per web session.
# Default Value: 536870912 (512MB)
##############################################################################
HTTP_MAX_MEM = 536870912

##############################################################################
# Min Value:     0
# Max Value:     1
# Default Value: 0
#
# Enable REST-API service.
##############################################################################
HTTP_ENABLE = 0

##############################################################################
# Min Value:     0
# Max Value:     1
# Default Value: 0
#
# Enable Basic Authentication for Rest-API service
##############################################################################
HTTP_AUTH = 0
```

http.conf

http.conf

```
{
    "document_root":"http/html/",
    "max_request_size": "100000",
    "request_timeout_ms": "10000",
    "enable_auth_domain_check": "no",
    "reverse_proxy" : [["/machbase/tables", "http://127.0.0.1:55657/machbase"],
        ["/self_machbase_proxy", "http://127.0.0.1:55657/machbase"],
        ["/dead_proxy", "http://127.0.0.0/machbase"]]
}
```

DDL / DML / AppendUsing REST API

```
Basic request format

http://addr:port/machbase?q=query&f=dateformat

Response DDL / Append / DML (except Select)
{"error_code":0, "error_message" :"Message", "data":[]}

Response DML (Select)
```

```
{"error_code":0, "error_message" :"Message", "columns":[Columns], "data":[Data]}
```

DDL Sample

```
## Request of creating a table
curl -G "http://127.0.0.1:5657/machbase" --data-urlencode 'q=create table test_table (name varchar(20), time datet

## Normal response
{"error_code":0, "error_message" :"No Error", "data":[]}

## Request of dropping a table
curl -G "http://127.0.0.1:5657/machbase" --data-urlencode 'q=drop table test_table'

## Normal response
{"error_code":0, "error_message" :"No Error", "data":[]}
```

DML Sample

```
## Request Log table data insert
curl -G "http://127.0.0.1:5657/machbase" --data-urlencode 'q=insert into test_table values ("test", "1999-01-01 00

## Response
{"error_code":0, "error_message" :"No Error", "data":[]}

## Request Log table select
curl -G "http://127.0.0.1:5657/machbase" --data-urlencode 'q=select * from test_table'

## Response
{"error_code":0, "error_message": "", "columns" : [{"name":"NAME", "type":5, "length":20},{"name":"TIME", "type":6,
```

Append Sample

```
## Append some data to log table
curl -X POST -H "Content-Type: application/json" "http://127.0.0.1:5657/machbase" -d '{"name":"test_table", "date_t

## Response
{"error_code":0, "error_message" :"No Error", "data":[], "append_success":3, "append_failure":0}
```

In the case of Binary Append, if binary data is encoded in Base64 and transmitted, the server will decode it and store it. When outputting, binary data is returned after b

Input : Binary Data >> Base64 Encoding >> HTTP(POST) >> Base64 Decoding >> Append(BLOB Binary)

Output : BLOB Binary >> Base64 Encoding >> HTTP (GET) >> Base64 Decoding >> Save or View Binary

Binary Append Sample

```
## Example of sending binary data. data should be encoded by Base64.

## Request append to log table
curl  -X POST -H "Content-Type: application/json" "http://127.0.0.1:5657/machbase" -d '{"name":"test_table", "date_

## Result
{"error_code":0, "error_message" :"No Error", "data":[], "append_success":1, "append_failure":0}

## Get data from log table
curl -G "http://127.0.0.1:5657/machbase" --data-urlencode 'q=select * from test_table';

## The Base64 encoded data are displaied
{"error_code" :0, "error_message": "No Error", "columns" : [{"name":"V1", "type":57, "length":67108864}],"data" :[

## Can check data using machsql
select to_hex(v1) from test_table;
to_hex(v1)
```

```
--------------------------------------------------------------------------------
000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F2021222324252627
28292A2B2C2D2E2F303132333435363738393A3B3C3D3E3F404142434445464748494A4B4C4D4E4F
505152535455565758595A5B5C5D5E5F606162636465666768696A6B6C6D6E6F7071727374757677
78797A7B7C7D7E7F808182838485868788898A8B8C8D8E8F909192939495969798999A9B9C9D9E9F
A0A1A2A3A4A5A6A7A8A9AAABACADAEAFB0B1B2B3B4B5B6B7B8B9BABBBCBDBEBFC0C1C2C3C4C5C6C7
C8C9CACBCCCDCECFD0D1D2D3D4D5D6D7D8D9DADBDCDDDEDFE0E1E2E3E4E5E6E7E8E9EAEBECEDEEEF
F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF
[1] row(s) selected.
```

Using HTTP Auth Property

This is an option to set authentication as a normal user by including the string 'Authorization: Basic Base64String' in the Request Header. Base64 string is written in ID

How to create a Basic Base64String for authorize

```
## In case of ID: sys, Password: manager , creating a Base64String
echo -n "sys@localhost:manager" | base64

## Result
c3lzQGxvY2FsaG9zdDptYW5hZ2Vy
```

Using Auth Sample (HTTP_AUTH = 1)

```
## Request of result withoutauthorization clause
curl -G "http://127.0.0.1:5657/machbase" --data-urlencode 'q=select * from v$stmt'

## Error occurred
{"error_code":3118, "error_message" :"There is No Authorization Header.", "data":[]}

## Adding'Authorization:Base64String' at the request header
curl -H "Authorization: Basic c3lzQGxvY2FsaG9zdDptYW5hZ2Vy"  -G "http://127.0.0.1:5657/machbase" --data-urlencode

## Normal response
{"error_code":0, "error_message": "No Error", "columns" : [{"name":"ID", "type":8, "length":4},{"name":"SESS_ID", "
```

Changing floating point precision with s option

Specify how many decimal places of response data to output Set to a value from 0 to 9 (If it is not a range value, it operates as 3)

Sample (s=5)

```
## display result 5 decimal places
curl -G "http://127.0.0.1:5657/machbase" --data-urlencode 'q=select * from test_table' --data-urlencode 's=5';

## Normal response
{"error_code" :0, "error_message": "", "columns" : [{"name":"C1", "type":16, "length":4},{"name":"C2", "type":20, "
```

Changing data Fetch mode (m option)

Decide whether to always display column names in response data (0: display, 1: do not display)

Default Fetch mode Sample (m=0)

```
## Requestfetch mode (m=0)
curl -G "http://127.0.0.1:5657/machbase" --data-urlencode 'q=select * from tag limit 2' --data-urlencode 'm=0';

## Normal response (ConatainsColumn Name in result data)
{"error_code" :0, "error_message": "", "columns" : [{"name":"NAME", "type":5, "length":20},{"name":"TIME", "type":6
"data" :[{"NAME":"tag1", "TIME":"2001-09-09 10:46:40 000:000:000", "VALUE":1000000000.000}, {"NAME":"tag1", "TIME"
```

Advanced Fetch mode Sample (m=1)

```
## Requestfetch mode (m=1)
curl -G "http://127.0.0.1:5657/machbase" --data-urlencode 'q=select * from tag limit 2' --data-urlencode 'm=1';
```

```
## Normal response (Column Namesare not included in results)
{"error_code" :0, "error_message": "", "columns" : [{"name":"NAME", "type":5, "length":20},{"name":"TIME", "type":6
"data" :[["tag1", "2001-09-09 10:46:40 000:000:000", 1000000000.000], ["tag1", "2001-09-09 10:46:41 000:000:000", 1
```

Handling of NULL values
When inserting or appending during DML processing, a NULL value can be entered as null as it is.

JSON Sample for appending null values

```
[["data1", "data2", "data3"],["data11", "data12", "data13"],["data21", "data22", "data23"],[null,null,null]]
```

Sample result containing null values from SELECT

```
[{"C1":null, "C2":null, "C3":null, "C4":null, "C5":null, "C6":null, "C7":null, "C8":null, "C9":null, "C10":null, "C
```

How To Use : Use MWA as a REST gateway(before 6.5 version)

Preferences

If `$MACHBASE_HOME/webadmin/flask/MWAserver start` is used to run Machbase Web Analytics (MWA), you can access the Machbase server using the Restful API.

```
[mach@localhost]$ cd $MACHBASE_HOME/bin
[mach@localhost]$ MWAserver start
```

The Machbase RESTful API can be used in conjunction with the MWA server configuration. When using the RESTful API, specify the server ID or Server name to use as the 's' parameter.
ex) The following example sends the query to the server whose ID is 3 (or server name is sample_server) among the servers set in MWA and receives the result

```
curl -G "http://127.0.0.1:5001/machbase" --data-urlencode 'q=select * from m$sys_tables' --data 's=3'
# or
curl -G "http://127.0.0.1:5001/machbase" --data-urlencode 'q=select * from m$sys_tables' --data 's=sample_server'
```

If it can not find the ID or name of the specified server, it returns the following json with a 404 error.

```
curl -G "http://127.0.0.1:5001/machbase" --data-urlencode 'q=select * from m$sys_tables' --data 's=not_exist'
{
  "error": "Cannot found server."
}
```

Default URL

The default URL for the Restful API is:

- http://hostname:port/machbase
- http://hostname:port/api/machbase

http://hostname: port can be found by running the MWAserver start command to run MWA with the url to connect to the MWA.

Data extraction

Retrieves the data using the HTTP GET method. The return value is a json type. You can pass the query statement you want to execute using the 'q' parameter. The query below brings up a list of tables.

```
curl -G "http://127.0.0.1:5001/machbase" --data-urlencode 'q=select * from m$sys_tables'
```

Data Input

You can use the HTTP POST method to send the input value of the json type as a parameter. Below is an example of entering data into a test_table with three columns.
First, we execute a query that creates a table using the curl GET method.

```
curl -X GET "http://127.0.0.1:5001/machbase" --data-urlencode 'q=create table test_table(c1 short, c2 integer, c3 v
```

Enter data using the POST method using the curl command. The data input json can use four keys, and the 'name' and 'values' keys must be entered.

```
curl -X POST -H "Content-Type: application/json" "http://127.0.0.1:5001/machbase" -d '{"name":"test_table", "values
```

Key	Description	Remarks
name	Input table name	Required
values	Input data to be input as a 2-dimensional array	Required
date_format	Date format to be used	If data type is used
s	Server ID or name	Server defined in MWA

If you enter a date type, you must specify the date format. The date format must use the pattern used in Machbase. If not specified otherwise, YYYY-MM-DD is set to HH24: MI: SS mmm: uuu: nnn Machbase default format and the result value will not be returned properly if date format is not correct. When specifying date and time pattern, you should be careful of case sensitive.

```
curl -X GET "http://127.0.0.1:5001/machbase" --data-urlencode 'q=create table test_date(c1 datetime)'

curl -X POST -H "Content-Type: application/json" "http://127.0.0.1:5001/machbase" -d '{"name":"test_date",  "date_t
```

Deleting Data

Standard RESTful APIs should use the HTTP DELETE method, but Machbase methods use the HTTP GET Method to send a delete query. Below is an example of deleting data from test_table.

```
curl -G "http://127.0.0.1:5001/machbase" --data-urlencode 'q=delete from test_table except 1 rows'
```

Creating, Altering, and Dropping Table

In Machbase, any query can be performed using the HTTP GET method. Therefore, the TABLE CREATE, ALTER, and DROP commands are also available.

```
curl -G "http://127.0.0.1:5001/machbase" --data-urlencode 'q=drop table test_table'
```

.NET Connector

The .NET (C #) Connector library that supports some features of the ADO.NET driver is provided.

The library location is `$MACHBASE_HOME/lib/` provided as a DLL type. It provides different DLLs depending on the .NET version.

- > .NET Framework 4.0: `machNetConnector.dll`
- > .NET Core 2.0: `machNetConnectorCore.dll`

> **GitHub**
> The source code can be found on github site.
>
> https://github.com/MACHBASE/NetConnector

Index

Class

> Features not listed below may not be implemented yet or may not work correctly.
>
> If you call a method or field that is not a named instance, it generates NotImplementedException or a NotSupportedException.

MachConnection : DbConnection

This class is responsible for linking with Machbase. Because it inherits IDisposable like DbConnection, it supports disassociation through Dispose () or automatic disposition of object using using () statement.

Constructor	Description
MachConnection(string aConnectionString)	Creates a MachConnection with a Connection String as input.

Method	Description
Open()	Attempts to connect to the connection string.
Close()	Closes the connection when connecting.
BeginDbTransaction(IsolationLevel isolationLevel)	(Not yet implemented) MACHBASE does not support this object because there is no special transaction.
CreateDbCommand()	(Not yet implemented) Explicitly induces MachCommands to be created
ChangeDatabase(string databaseName)	(Not yet implemented) MACHBASE has no DATABASE classification.

Field	Description
State	Represents a System.Data.ConnectionState value.
StatusString	Indicates the state to be performed by the connected MachCommand. This is used internally to decorate the Error Message and it is not appropriate to check the status of the query with this value because it indicates the state in which the operation started.
Database	(Not yet implemented)
DataSource	(Not yet implemented)
ServerVersion	(Not yet implemented)

> **Connection String**
>
> Each item is separated by a semicolon (;).
>
> Many of the keywords in the same section have the same meaning.

Keyword	Description	Example	Default Value

Keyword	Description	Example	Default Value
SERVER	Hostname	SERVER=192.168.0.1	
PORT PORT_NO	Port No.	PORT=5656	5656
USERID USERNAME USER UID	User ID	USER=SYS	SYS
PASSWORD PWD	User password	PWD=manager	
CONNECT_TIMEOUT ConnectionTimeout connectTimeout	Maximum connection time	CONNECT_TIMEOUT	60 second
COMMAND_TIMEOUT commandTimeout	Maximum time to perform each command	COMMAND_TIMEOUT	60 second

As an example, we can prepare the following string.

```
String sConnString = String.Format("SERVER={0};PORT_NO={1};UID=;PWD=MANAGER;CONNECT_TIMEOUT=10000;COMMAND_TIM
```

MachCommand : DbCommand

A class that performs **SQL commands or APPEND** using MachConnection .
Since it inherits IDisposable like DbCommand, it supports object disposal through Dispose () or automatic disposal of object using using () statement.

Constructor	Description
MachCommand(string aQueryString, MachConnection)	Creates by typing the query to be executed along with the MachConnection object to be connected.
MachCommand(MachConnection)	Creates a MachConnection object to connect to. Use only if there is no query to perform (eg APPEND).

Method	Description
void CreateParameter() / void CreateDbParameter()	Creates a new MachParameter.
void Cancel()	(Not yet implemented)
void Prepare()	(Not yet implemented)

MachAppendWriter
AppendOpen(aTableName, aErrorCheckCount = 0,
MachAppendOption = None)
	Starts APPEND. Returns a MachAppendWriter object.
	• aTableName: Target table name
	• aErrorCheckCount: Each time the cumulative number of records entered by APPEND-DATA matches, it is checked whether it is sent to the server or not.
	In other words, you are setting the automatic APPEND-FLUSH point.
	• MachAppendOption: Currently only one option is provided.
	  • MachAppendOption.None: No options are attached.
	  • MachAppendOption. MicroSecTruncated: When inputting the value of a DateTime object, enter the value expressed only up to microsecond.
	(The Ticks value of a DateTime object is expressed up to 100 nanoseconds.)

void AppendData(MachAppendWriter aWriter, List<object> aDataList)	Through the MachAppendWriter object, it takes a list containing the data and enters it into the database. • In the order of the data in the List, each datatype must match the datatype of the column represented in the table. • If the data in the List is insufficient or overflows, an error occurs. ⓘ When representing a time value with a ulong object, simply do not enter the Tick value of the DateTime object. In that value, you must enter a value that excludes the DateTime Tick value that represents 1970-01-01 .
void AppendDataWithTime(MachAppendWriter aWriter, List<object> aDataList, DateTime aArrivalTime)	Method that explicitly puts an _arrival_time value into a DateTime object in AppendData ().

Method	Description
void AppendDataWithTime(MachAppendWriter aWriter, List<object> aDataList, ulong aArrivalTimeLong)	Method that can explicitly put _arrival_time value into a ulong object in AppendData (). Refer to AppendData () above for problems that may occur when typing a ulong value as an _arrival_time value .
void AppendFlush(MachAppendWriter aWriter)	The data entered by AppendData () is immediately sent to the server to force data insert. The more frequently the call is made, the lower the data loss rate due to the system error and the faster the error check, although the performance is lowered. The less frequently the call is made, the more likely the data loss will occur and the error checking will be delayed, but the performance will increase significantly.
void AppendClose(MachAppendWriter aWriter)	Closes APPEND. Internally, after calling AppendFlush (), the actual protocol is internally finished.
int ExecuteNonQuery()	Performs the input query. Returns the number of records affected by the query. It is usually used when performing queries except SELECT.
object ExecuteScalar()	Performs the input query. Returns the first value of the query targetlist as an object. It is usually used when you want to perform a SELECT query, especially a SELECT (Scalar Query) with only one result, and get the result without a DbDataReader
DbDataReader ExecuteDbDataReader(CommandBehavior aBehavior)	Executes the input query, generates a DbDataReader that can read the result of the query, and returns it.

Field	Description
Connection / DbConnection	Connected MachConnection.
ParameterCollection / DbParameterCollection	The MachParameterCollection to use for the Binding purpose.
CommandText	Query string.
CommandTimeout	The amount of time it takes to perform a particular task, waiting for a response from the server. It follows the values set in MachConnection, where you can only reference values.
FetchSize	The number of records to fetch from the server at one time . The default value is 3000.
IsAppendOpened	Determines if Append is already open when APPEND is at work
CommandType	(Not yet implemented)
DesignTimeVisible	(Not yet implemented)
UpdatedRowSource	(Not yet implemented)

MachDataReader : DbDataReader

This is a class that reads fetch results. Only objects created with MachCommand.ExecuteDbDataReader () that can not be explicitly created are available.

Method	Description
string GetName(int ordinal)	Returns the ordinal column name.
string GetDataTypeName(int ordinal)	Returns the datatype name of the ordinal column.
Type GetFieldType(int ordinal)	Returns the datatype of the ordinal column.
int GetOrdinal(string name)	Returns the index at which the column name is located.
object GetValue(int ordinal)	Returns the ordinal value of the current record.
bool IsDBNull(int ordinal)	Returns whether the ordinal value of the current record is NULL.
int GetValues(object[] values)	Sets all the values of the current record and returns the number.
xxxx GetXXXX(int ordinal)	Returns the ordinal column value according to the datatype (XXXX). • Boolean

Method	Description
	• Byte • Char • Int16 / 32/64 • DateTime • String • Decimal • Double • Float
bool Read()	Reads the next record. Returns False if the result does not exist.
DataTable GetSchemaTable()	(Not supported)
bool NextResult()	(Not supported)

Field	Description
FetchSize	The number of records to fetch from the server at one time. The default is 3000, which can not be modified here.
FieldCount	Number of result columns.
this[int ordinal]	Equivalent to object GetValue (int ordinal).
this[string name]	Equivalent to object GetValue(GetOrdinal(name).
HasRows	Indicates whether the result is present.
RecordsAffected	Unlike MachCommand, here, it represents Fetch Count.

MachParameterCollection : DbParameterCollection

This is a class that binds parameters needed by MachCommand.

If you do this after binding, the values are done together.

> Since the concept of Prepared Statement is not implemented, execution performance after Binding is the same as the performance performed first.

Method	Description
MachParameter Add(string parameterName, DbType dbType)	Adds the MachParameter, specifying the parameter name and type. Returns the added MachParameter object.
int Add(object value)	Adds a value. Returns the index added.
void AddRange(Array values)	Adds an array of simple values.
MachParameter AddWithValue(string parameterName, object value)	Adds the parameter name and its value. Returns the added MachParameter object.
bool Contains(object value)	Determines whether or not the corresponding value is added.
bool Contains(string value)	Determines whether or not the corresponding parameter name is added.
void Clear()	Deletes all parameters.
int IndexOf(object value)	Returns the index of the corresponding value.
int IndexOf(string parameterName)	Returns the index of the corresponding parameter name.
void Insert(int index, object value)	Adds the value to a specific index.

Method	Description
void Remove(object value)	Deletes the parameter including the value.
void RemoveAt(int index)	Deletes the parameter located at the index.
void RemoveAt(string parameterName)	Deletes the parameter with that name.

Field	Description
Count	Number of parameters
this[int index]	Indicates the MachParameter at index.
this[string name]	Indicates the MachParameter of the order in which the parameter names match.

MachParameter : DbParameter

This is a class that contains the information that binds the necessary parameters to each MachCommand.

No special methods are supported.

Field	Description
ParameterName	Parameter name
Value	Value
Size	Value size
Direction	ParameterDirection (Input / Output / InputOutput / ReturnValue) The default value is Input.
DbType	DB Type
MachDbType	MACHBASE DB Type May differ from DB Type.
IsNullable	Whether nullable
HasSetDbType	Whether DB Type is specified

MachException : DbException

This is a class that displays errors that appear in Machbase.

An error message is set, and all error messages can be found in *MachErrorMsg* .

Field	Description
int MachErrorCode	Error code provided by MACHBASE

MachAppendWriter

APPEND is supported as a separate class using MachCommand.

This is a class to support MACHBASE Append Protocol, not ADO.NET standard.
It is created with MachCommand's AppendOpen () without a separate constructor.

Method	Description
void SetErrorDelegator(ErrorDelegateFuncType aFunc)	Specifies the ErrorDelegateFunc to call when an error occurs.

Field	Description
SuccessCount	Number of successful records. Is set after AppendClose ().

Field	Description
FailureCount	The number of records that failed input. Set after AppendClose ().
Option	MachAppendOption received input during AppendOpen()

ErrorDelegateFuncType

```
public delegate void ErrorDelegateFuncType(MachAppendException e);
```

In MachAppendWriter, you can specify a function to detect errors occurring on the MACHBASE server side during APPEND.

In .NET, this function type is specified as a Delegator Function.

MachAppendException : MachException

Same as MachException, except that:

- An error message is received from the server side.
- A data buffer in which an error has occurred can be obtained. (comma-separated) can be used to process and re-append or record data.

The exception is only available within the ErrorDelegateFunc.

Method	Description
GetRowBuffer()	A data buffer in which an error has occurred can be obtained.

MachTransaction

Not supported.

Sample Code

Connection

You can create a MachConnection and use Open () - Close ().

```
String sConnString = String.Format("SERVER={0};PORT_NO={1};UID=;PWD=MANAGER;", SERVER_HOST, SERVER_PORT);
MachConnection sConn = new MachConnection(sConnString);
sConn.Open();
//... do something
sConn.Close();
```

If you use the using statement, you do not need to call Close (), which is a connection closing task.

```
String sConnString = String.Format("SERVER={0};PORT_NO={1};UID=;PWD=MANAGER;", SERVER_HOST, SERVER_PORT);
using (MachConnection sConn = new MachConnection(sConnString))
{
    sConn.Open();
    //... do something
} // you don't need to call sConn.Close();
```

Performing Queries

Create a MachCommand and perform the query.

```
String sConnString = String.Format("SERVER={0};PORT_NO={1};UID=;PWD=MANAGER;", SERVER_HOST, SERVER_PORT);
using (MachConnection sConn = new MachConnection(sConnString))
{
    String sQueryString = "CREATE TABLE tab1 ( col1 INTEGER, col2 VARCHAR(20) )";
    MachCommand sCommand = new MachCommand(sQueryString , sConn)
```

```
    try
    {
        sCommand.ExecuteNonQuery();
    }
    catch (MachException me)
    {
        throw me;
    }
}
```

Again, using the using statement, MachCommand release can be done immediately.

```
String sConnString = String.Format("SERVER={0};PORT_NO={1};UID=;PWD=MANAGER;", SERVER_HOST, SERVER_PORT);
using (MachConnection sConn = new MachConnection(sConnString))
{
    String sQueryString = "CREATE TABLE tab1 ( col1 INTEGER, col2 VARCHAR(20) )";
    using(MachCommand sCommand = new MachCommand(sQueryString , sConn))
    {
        try
        {
            sCommand.ExecuteNonQuery();
        }
        catch (MachException me)
        {
            throw me;
        }
    }
}
```

Performing Select

You can get a MachDataReader by executing a MachCommand with a SELECT query.

You can fetch the records one by one through the MachDataReader.

```
String sConnString = String.Format("SERVER={0};PORT_NO={1};UID=;PWD=MANAGER;", SERVER_HOST, SERVER_PORT);
using (MachConnection sConn = new MachConnection(sConnString))
{
    String sQueryString = "SELECT * FROM tab1;";
    using(MachCommand sCommand = new MachCommand(sQueryString , sConn))
    {
        try
        {
            MachDataReader sDataReader = sCommand.ExecuteReader();
            while (sDataReader.Read())
            {
                for (int i = 0; i < sDataReader.FieldCount; i++)
                {
                    Console.WriteLine(String.Format("{0} : {1}",
                                                    sDataReader.GetName(i),
                                                    sDataReader.GetValue(i)));
                }
            }
        }
        catch (MachException me)
        {
            throw me;
        }
    }
}
```

Parameter Binding

You can create a MachParameterCollection and then link it to a MachCommand.

```
String sConnString = String.Format("SERVER={0};PORT_NO={1};UID=;PWD=MANAGER;", SERVER_HOST, SERVER_PORT);
```

```
using (MachConnection sConn = new MachConnection(sConnString))
{
    string sSelectQuery = @"SELECT *
        FROM tab2
        WHERE CreatedDateTime < @CurrentTime
        AND CreatedDateTime >= @PastTime";

    using (MachCommand sCommand = new MachCommand(sSelectQuery, sConn))
    {
        DateTime sCurrtime = DateTime.Now;
        DateTime sPastTime = sCurrtime.AddMinutes(-1);

        try
        {
            sCommand.ParameterCollection.Add(new MachParameter { ParameterName = "@CurrentTime", Value = sCurrtime
            sCommand.ParameterCollection.Add(new MachParameter { ParameterName = "@PastTime", Value = sPastTime }),

            MachDataReader sDataReader = sCommand.ExecuteReader();

            while (sDataReader.Read())
            {
                for (int i = 0; i < sDataReader.FieldCount; i++)
                {
                    Console.WriteLine(String.Format("{0} : {1}",
                                                    sDataReader.GetName(i),
                                                    sDataReader.GetValue(i)));
                }
            }
        }
        catch (MachException me)
        {
            throw me;
        }
    }
}
```

APPEND

When you run AppendOpen () on a MachCommand, you get a MachAppendWriter object.

Using this object and MachCommand, you can get a list of one input record and perform an AppendData ().
AppendFlush () will reflect the input of all records, and AppendClose () will end the entire Append process.

```
String sConnString = String.Format("SERVER={0};PORT_NO={1};UID=;PWD=MANAGER;", SERVER_HOST, SERVER_PORT);
using (MachConnection sConn = new MachConnection(sConnString))
{
    using (MachCommand sAppendCommand = new MachCommand(sConn))
    {
        MachAppendWriter sWriter = sAppendCommand.AppendOpen("tab2");
        sWriter.SetErrorDelegator(AppendErrorDelegator);

        var sList = new List<object>();
        for (int i = 1; i <= 100000; i++)
        {
            sList.Add(i);
            sList.Add(String.Format("NAME_{0}", i % 100));

            sAppendCommand.AppendData(sWriter, sList);

            sList.Clear();

            if (i % 1000 == 0)
            {
                sAppendCommand.AppendFlush();
            }
        }

        sAppendCommand.AppendClose(sWriter);
        Console.WriteLine(String.Format("Success Count : {0}", sWriter.SuccessCount));
```

```
        Console.WriteLine(String.Format("Failure Count : {0}", sWriter.FailureCount));
    }
}
```

```
private static void AppendErrorDelegator(MachAppendException e)
{
    Console.WriteLine("{0}", e.Message);
    Console.WriteLine("{0}", e.GetRowBuffer());
}
```

Tag Table RESTful API

Overview

Machbase provides a RESTful API for accessing the Tag Table.
If you run Machbase Web Analytics (MWA) using $MACHBASE_HOME/webadmin/flask/MWAserver start, you can use the Tag Table Restful API.

```
[mach@localhost]$ cd $MACHBASE_HOME/bin
[mach@localhost]$ MWAserver start
```

> Unless otherwise noted, all use the GET method.

Index

Check Working Status

Usage

```
{MWA URL}/machiot-rest-api/      --> Latest version api
{MWA URL}/machiot-rest-api/v1/   --> Version 1
```

Example

```
http://192.168.0.31:5003/machiot-rest-api/
http://192.168.0.31:5003/machiot-rest-api/v1/
```

Result

```
{
 "status": "Working",
 "version": "1.0"
}
```

curl Example

```
curl -G 'http://192.168.0.31:5003/machiot-rest-api/'
```

get Calculated Data

Usage

```
{MWA URL}/machiot-rest-api/datapoints/calculated/{TagName}/{Start}/{End}/{CalculationMode}/{Count}/{IntervalType}/

TagName         : Tag Name. Supports multiple tags (separated by ,) -> Passes multiple Tag results to a single Seri
Start, End      : Period, YYYY-MM-DD HH24:MI:SS or YYYY-MM-DD
CalculationMode : min, max, total(sum), count(cnt), average(avg)
Count           : LIMIT, all if 0
IntervalType    : sec(second), min(minute), hour, day
IntervalValue   : Unit of IntervalType (5 sec, 3 min, etc.) Default 1
```

Can also be passed to Get method parameter (duplicate URL takes precedence)

> If TagName is separated by, and multiple tags are specified, it is returned as one Series.
> Use this when you want to combine several tags that have data only in a certain time zone and get results in one series.
> If the data is duplicated, the data of the quick tag is selected in the order by order of TagName.

Example

```
http://192.168.0.31:5003/machiot-rest-api/v1/datapoints/calculated/TAG_193,TAG_194/2018-01-16T14:40:00/2018-01-16T1
```

```
http://192.168.0.31:5003/machiot-rest-api/v1/datapoints/calculated?TagNames=TAG_193,TAG_194&amp;Start=2018-01-16T1
```

Result

```
{
 "ErrorCode": 0,
 "ErrorMessage": "",
 "Data": [
     {"DataType": "LONG",
      "ErrorCode": 0,
      "TagName": "TAG_193,TAG_194",
      "CalculationMode": "cnt",
      "Samples": [
          {"TimeStamp": "2018-01-16 14:40:05 000:000:000", "Value": 6, "Quality": 0},
          {"TimeStamp": "2018-01-16 14:40:06 000:000:000", "Value": 25, "Quality": 0}, ....
      ]
     }
 ]
}
```

curl Example

```
curl -G 'http://192.168.0.31:5003/machiot-rest-api/datapoints/calculated/TAG_193,TAG_194/2018-01-16T14:40:00/2018-0
```

get Raw Data

Usage

```
{MWA URL}/machiot-rest-api/datapoints/raw/{TagName}/{Start}/{End}/{Direction}/{Count}/{Offset}/

TagName    : Tag Name. Supports multiple tags (separated by ,) -> Passes multiple Tag results to a single Series
Start, End : Period, YYYY-MM-DD HH24:MI:SS or YYYY-MM-DD or YYYY-MM-DD HH24:MI:SS,mmm (mmm: millisecond, if omitte
Direction  : 0(not use= default value), 1(descend), 2(ascend)
Count      : LIMIT, all if 0
Offset     : offset (default value = 0)
```

Can also be passed to Get method parameter (duplicate URL takes precedence)

> If TagName is separated by, and multiple tags are specified, it is returned as one Series.
> Use this when you want to combine several tags that have data only in a certain time zone and get results in one series.
> If the data is duplicated, the data of the quick tag is selected in the order by order of TagName.

Example

```
http://192.168.0.31:5003/machiot-rest-api/v1/datapoints/raw/TAG_193,TAG_194/2018-01-16T14:40:00,500/2018-01-16T14:4
http://192.168.0.31:5003/machiot-rest-api/v1/datapoints/raw?TagNames=TAG_193,TAG_194&amp;Start=2018-01-16T14:40:00,
```

Result

```
{
 "ErrorCode": 0,
 "ErrorMessage": "",
 "Data": [
     {"DataType": "DOUBLE",
      "ErrorCode": 0,
      "TagName": "TAG_193,TAG_194",
      "Samples": [
          {"TimeStamp": "2018-01-16 14:40:05 538:012:000", "Value": 100.0, "Quality": 0},
          {"TimeStamp": "2018-01-16 14:40:05 798:813:000", "Value": 26.0, "Quality": 0}, ....
      ]
     }
 ]
```

```
}
```

curl Example

```
curl -G 'http://192.168.0.31:5003/machiot-rest-api/datapoints/raw/TAG_193,TAG_194/2018-01-16T14:40:00,500/2018-01-1
```

INSERT

When called with the POST method, it is appended to the TAG table.
parameter is the same, except that the name (= Table Name) is not present in the POST method parameter of the RESTful API.

Parameter

```
{
 "values":[
     [TAG_NAME, TAG_TIME, VALUE],
     [ .... ]....
 ],
 "date_format":"Date Format"
}

Note: If date_format is omitted, set to 'YYYY-MM-DD HH24: MI: SS mmm: uuu: nnn'
```

Example: GET URL

```
http://192.168.0.31:5003/machiot-rest-api/
```

Example: POST data

```
{
 "values":[
     ["TAG_0","2018-01-11 01:16:37",100],
     ["TAG_1","2018-01-11 01:16:37",100],
     ["TAG_2","2018-01-11 01:16:37",100],
     ["TAG_3","2018-01-11 01:16:37",100],
     ["TAG_4","2018-01-11 01:16:37",100],
     ["TAG_5","2018-01-11 01:16:37",100],
     ["TAG_6","2018-01-11 01:16:37",100],
     ["TAG_7","2018-01-11 01:16:37",100],
     ["TAG_8","2018-01-11 01:16:37",100],
     ["TAG_9","2018-01-11 01:16:37",100]
 ],
 "date_format":"YYYY-MM-DD HH24:MI:SS"
}
```

curl Example

```
curl -X POST -H "Content-Type: application/json" "http://127.0.0.1:5003/machiot-rest-api/" -d '{ "values":[["TAG_0(
```

Get Tag List

Obtain a list of tags.

Usage

```
{MWA URL}/machiot-rest-api/tags/list/
```

Example

```
http://192.168.0.31:5003/machiot-rest-api/tags/list/
```

Result

```
{
 "ErrorCode": 0,
 "ErrorMessage": "",
 "Data": [
     {"NAME": "TAG_00"},
     {"NAME": "TAG_01"},
     {"NAME": "TAG_02"},
     {"NAME": "TAG_03"}, ......
 ]
}
```

curl Example

```
curl -G 'http://192.168.0.31:5003/machiot-rest-api/tags/list'
```

Get Time Range

Obtain the time range of the input data.

Usage

```
{MWA URL}/machiot-rest-api/tags/range/
{MWA URL}/machiot-rest-api/tags/range/{TagName}

TagName    : Tag Name (Currently, multiple Tag Names are not supported)
If there is no TagName, obtains the time range for all data.
```

Example

```
http://192.168.0.31:5003/machiot-rest-api/tags/range/TAG_00
```

Result

```
{
 "ErrorCode": 0,
 "ErrorMessage": "",
 "Data": [
     {"MIN": "2017-01-01 00:00:00 000:000:000", "MAX": "2017-02-02 19:02:13 000:000:000"}
 ]
}
```

curl Example

```
curl -G 'http://192.168.0.31:5003/machiot-rest-api/tags/range/TAG_00'
```

Request Rollup

Force rollup. (Execute 'EXEC ROLLUP_FORCE' command, takes about 6 seconds)

Usage

```
{MWA URL}/machiot-rest-api/rollup/
```

Example

```
http://192.168.0.31:5003/machiot-rest-api/rollup/
```

Result

```
{
```

```
 "ErrorCode": 0,
 "ErrorMessage": "",
 "Data": [
     {"EXECUTE RESULT": "Execute Success"}
 ]
}
```


curl Example

```
curl -G 'http://192.168.0.31:5003/machiot-rest-api/rollup/'
```

부록 Appendix

Needs to be completed

페이지 잠금 상태입니다. 외부에서 공개되지 않고 로그인 사용자만 접근 가능합니다.
이유 : 유일한 페이지인 Github 활용 방법이 공란입니다. 해당 부분이 채워질 때 까지 '부록' 페이지도 사용할 필요가 없습니다.

MACHBASER GitHub

On this site, https://github.com/MACHBASE, you can find many valuable materials in using MACHBASE.