GMA022J4

GridDB Advanced Edition
SQL reference

Introduction

This manual describes how to write a SQL command in the GridDB Advanced Edition.

Please read this manual before using the GridDB Advanced Edition / Vector Edition.

The functions described in this manual can be used exclusively by users with GridDB Advanced Edition / Vector Edition license.

Trademarks

Table of Contents

1. About the GridDB Advanced Edition

An interface (NewSQL interface) that can access GridDB data using SQL is provided in the GridDB Advanced Edition.

This manual explains the SQL commands of the NewSQL interface used to access a database supported by the GridDB Advanced Edition (hereinafter referred to as GridDB AE). Note that the interface is different from the NoSQL interface.

The NewSQL interface can also be used in GridDB Vector Edition (hereinafter referred to as GridDB VE). See “GridDB Advanced Edition JDBC Driver Instructions” for how to use the JDBC driver and “GridDB Vector Edition manual” for the details of GridDB VE extended SQL.

2. SQL description format

This chapter shows the descriptive format of the SQL that can be used in the NewSQL interface.

2.1 Usable operations

In this version, besides the SELECT command, DDL command (Data Definition Language) such as CREATE TABLE, and INSERT/DELETE are also supported. See chapter 3 for details.

2.2 Data types

2.2.1 Data types used in data storage and NULL storage

The data types used for data storage in the NewSQL interface are shown in Table 1. These data type names can be written as a column data type when creating a table.
An INTEGER in NoSQL interface client is a 32-bit positive number but an INTEGER in NewSQL interface is a 64-bit positive number and the range of values that can be stored is different.

Table 1: Data types used in data storage and NULL storage

Data types	Detailed contents	Handling of NULL
BOOL	True or false	-
BYTE	Integer value from -2^7 to 2^7-1 (8 bits)	0
SHORT	Integer value from -2^{15} to $2^{15}-1$ (16 bits)	0
INTEGER	Integer value from -2^{31} to $2^{31}-1$ (32 bits)	0
LONG	Integer value from -2^{63} to $2^{63}-1$ (64 bits)	0
FLOAT	Single-precision data type (32 bits) Floating-point number defined in IEEE754	0.0
DOUBLE	Double-precision data type (64 bits) Floating-point number defined in IEEE754	0.0
TIMESTAMP	Data type expressing the date and time Data format maintained in the database is UTC, and accuracy is in milliseconds	-
STRING	Text that is composed of an arbitrary number of characters using the unicode code point	Text with size 0
BLOB	Data type for binary data such as images and voice, etc. Large objects to be saved directly in the input format. The character x or X can also be added to create a hexadecimal expression such as X'23AB'.	Data of size 0

In addition, this version does not allow NULL to be stored in a table. If you try to store a NULL, the value indicated in Table 1 will be stored. However, NULL can be used in a calculation using SQL equations.

Based on this, note that the result may be different from the SQL specifications when an operator such as “IS NULL”, etc. is used for a value stored in a table.

2.2.2 Expression that can be written as a column data type when creating a table.

In the NewSQL interface, for data type names that are described as column data types when the table was created, even if the name does not match the data type name given in 2.2.1, follow the rules to interpret and determine the data type to use for data storage.

Check the following rules in sequence starting from the top and determine the data type to use for data storage based on the applicable rule. The data type name described when checking the rules and the strings to check using the rules are not case sensitive. If multiple rules apply, the rule ranked higher will be prioritized. If no rules are applicable, an error will occur and table creation will fail.

Rule No.	Specified column name string	Column type of the table
1	Type names listed in 2.2.1	Same as specified type
2	"REAL"	DOUBLE
3	"TINYINT"	BYTE
4	"SMALLINT"	SHORT
5	"BIGINT"	LONG
6	Type name including "INT"	INTEGER
7	Type name including any of "CHAR", "CLOB", "TEXT"	STRING
8	Type name including "BLOB"	BLOB
9	Type name including any of "REAL", "DOUB"	DOUBLE
10	Type name including "FLOA"	FLOAT

An example to determine the data type using this rule is shown.

- Name of written data type is "BIGINTEGER" -> INTEGER (Rule 6)
- Name of written data type is "LONG" -> LONG (Rule 1)
- Name of written data type is "TINYINT" -> BYTE (Rule 3)
- Name of written data type is "FLOAT" -> FLOAT (Rule 1)
- Name of written data type is "VARCHAR" -> STRING (Rule 7)
- Name of written data type is "CHARINT" -> INTEGER (Rule 6)
- Name of written data type is "BIGBLOB" -> BLOB (Rule 8)
- Name of written data type is "FLOATDOUB" -> DOUBLE (Rule 9)
- Name of written data type is "INTREAL" -> INTEGER (Rule 6)
- Name of written data type is "FLOATINGPOINT" -> INTEGER (Rule 6)
- Name of written data type is "DECIMAL" -> error

Write the data type as follows when using the same data type in the NewSQL interface as the data type used in the clients of the NoSQL interface. However, some equivalent data types may not be available as they do not exist.

Table 2: Column data type descriptions which are the same as the data types in a NoSQL interface client

Data type in a NoSQL interface client	Column data type descriptions of a NewSQL interface with the same data types
STRING (string data type)	"STRING" or "Expression to be STRING"
BOOL (Boolean)	"BOOL"

BYTE (8-bit integer)	"BYTE" or "Expression to be BYTE"
SHORT (16-bit integer)	"SHORT" or "Expression to be SHORT"
INTEGER (32-bit integer)	"INTEGER" or "Expression to be INTEGER"
LONG (64-bit integer)	"LONG" or "Expression to be LONG"
FLOAT (32 bitwise floating point number)	"FLOAT" or "Expression to be FLOAT"
DOUBLE (64 bitwise floating point number)	"DOUBLE" or "Expression to be DOUBLE"
TIMESTAMP (time data type)	"TIMESTAMP"
GEOMETRY (spatial data type)	Non-existent (cannot be used)
BLOB	"BLOB" or "Expression to be BLOB"
ARRAY	Non-existent (cannot be used)

2.2.3 Data type when accessing a container as a table and the treatment of the values

The column data types of the container and treatment of the values when accessing a container created in a NoSQL interface client as a table in the NewSQL interface are shown below.

Table 3: Treatment of container data types and values in the NewSQL interface

Column type of container	Data type mapped in NewSQL	Value
STRING	STRING	Same as original value
BOOL	BOOL	Same as original value
BYTE	BYTE	Same as original value
SHORT	SHORT	Same as original value
INTEGER	INTEGER	Same as original value
LONG	LONG	Same as original value
FLOAT	FLOAT	Same as original value
DOUBLE	DOUBLE	Same as original value
TIMESTAMP	TIMESTAMP	Same as original value
GEOMETRY	STRING	All the values are NULL
BLOB	BLOB	Same as original value
ARRAY	STRING	All the values are NULL

2.3 User and database

There are 2 types of GridDB user, an administrator user and a general user, which differ in terms of the functions which can be used. In addition, access can be separated on a user basis by creating a database. See “Technical Reference” for details of the user and database.

Furthermore, to register and search for data with the NewSQL interface, a table needs to be created to store the data. Data to be registered is known as a row and is made up of 1 or more column data.

2.4 Naming rule

A database name, table name, column name and general user name is a string composed of 1 or more ASCII alphanumeric characters, the underscore “_” , the hyphen “-” , the dot “.” , the slash “/” and the equal “=”. However, if the first character of the name is a number or the name contains special characters except the underscore, it is necessary to enclose the name with double quotation.

For table name, the “@” character can also be specified for the node affinity function. See “Technical Reference” for the node affinity function.

Further for the details of naming rule and limitations, see “Naming limitations” of “Technical Reference”.

3. SQL commands supported by GridDB AE

Supported SQL commands are Table 4 as follows.

Table 4: List of supported SQL commands

Command	Overview
CREATE DATABASE	Create a database.
CREATE TABLE	Create a table.
CREATE INDEX	Create an index.
CREATE USER	Create a general user.
DROP DATABASE	Delete a database.
DROP TABLE	Delete a table.
DROP INDEX	Delete an index.
DROP USER	Delete a general user.
SET PASSWORD	Change the password of a general user.
GRANT	Set the access rights to the database for a general user.
REVOKE	Set the access rights to the database for a general user.
INSERT	Insert a row into a table.
DELETE	Delete a row from a table.
UPDATE	Update a row in a table.
SELECT	Get data.
Comment	Add a comment

An explanation for each category of SQL command is given in this chapter.

3.1 Data definition language (DDL)

Composed of CREATE, DROP commands, etc.

3.1.1 CREATE DATABASE

Create a database.

Type
CREATE DATABASE database name;

- Can be executed by an administrator user only.
- Databases with the same name as “public”, “information_schema” cannot be created as these are reserved for internal use in GridDB.
- Nothing will be changed if a database with the same name already exists.
- Only ASCII alphanumeric characters, the underscore mark (“_”), the hyphen (“-“), the dot (“.”), the slash (“/”) and the equal (“=”) can be used in the database name. ASCII characters are not case-sensitive.

3.1.2 CREATE TABLE

Create a table.

Type
CREATE TABLE [IF NOT EXISTS] table name (column definition 1* [, column definition 2 …])

For time series table

CREATE TABLE [IF NOT EXISTS] table name (column name TIMESTAMP PRIMARY KEY [, column definition 2 …]) USING TIMESERIES
[WITH (property key=property value [, property key=property value …])]

column definition: column name, data type name [column constraint 1 [, column constraint 2 …]
*column constraint: [PRIMARY KEY] (only the leading column can be specified)

- If “IF NOT EXISTS” is specified, the specified table can be created only if another table with the same name does not exist.
- The column name and data type name need to be specified in “Column Definition”.
  - See 2.2 Data types for the names of the data types that can be specified.
- “Column Constraint” supports “PRIMARY KEY” and “NOT NULL”. Only the “PRIMARY KEY” of the leading column is supported in “Column Constraint”.
- For time series tables, options about time series can be specified by the format “WITH (property key=property value, …).

Function	Item	Property key	Property value
Expiration	Elapsed time	expiration_time	INTEGER
	Elapsed time unit	expiration_time_unit	STRING (Any of the followings. “DAY” / “HOUR” / “MINUTE” / “SECOND” / “MILLISECOND”)
	Division count	expiration_division_count	INTEGER

See “Technical Reference” for each item.

(Example)

CREATE TABLE *myTimeseries* (*mycolumn1* TIMESTAMP PRIMARY KEY, *mycolumn2* STRING) USING TIMESERIES WITH (expiration_time=*10*,expiration_time_unit='DAY')

- Only ASCII alphanumeric characters, the underscore mark (“_”), the hyphen (“-“), the dot (“.”), the slash (“/”) and the equal (“=”) can be used in the table name and column name. ASCII characters are not case-sensitive.
- A node affinity function can be used by appending “@hint data” to the table name. See “Technical Reference” for the node affinity function.

3.1.2.1 Creating a partitioned table

See “Technical Reference” for the function of each partitioning.

■ Creating a hash partitioned table

Type
CREATE TABLE [IF NOT EXISTS] *table name* (column definition1 [, column definition2 …])
[USING TIMESERIES [WITH (*property key=property value*, ...)]]
PARTITION BY HASH (*column name of partitioning key*) PARTITIONS *division count*

- Create a hash partitioned table by the column name of partitioning key and the value of division count.
- Specify the value from 1 to 1024 for “division count”.
- The column specified as partitioning key cannot be updated.
- See the section “Hash partitioning” in “Technical Reference” for the function of hash partitioning.

(Example)

CREATE TABLE *myHashPartition* (*id* INTEGER PRIMARY KEY, *value* STRING)
PARTITION BY HASH (*id*) PARTITIONS *128*

■ Creating an interval partitioned table

Type
CREATE TABLE [IF NOT EXISTS] *table name* (column definition1 [, column definition2 …])
[USING TIMESERIES [WITH (*property key=property value*, ...)]]
PARTITION BY RANGE (*column name of partitioning key*) EVERY *(interval value [, interval unit])*

- Specify the column which type is BYTE, SHORT, INTEGER, LONG or TIMESTAMP for “column name of partitioning key”.
- The column specified for “column name of partitioning key” is required to be “PRIMARY KEY” or have “NOT NULL” constraint.
- The column specified as partitioning key cannot be updated.
- The following values can be specified as the interval value.

Partitioning key type	Possible interval value
BYTE	1～127
SHORT	1～32767
INTEGER	1～2147483647
LONG	1000～9223372036854775807
TIMESTAMP	1～

- If the column of TIMESTAMP is specified, it is also required to specify the interval unit. 'DAY' is the only value that can be specified as the interval unit.
- The interval unit cannot be specified for any types other than TIMESTAMP.

- See the section "Interval partitioning" in "Technical Reference" for the function of interval partitioning.

(Example)

```
CREATE TABLE myIntervalPartition (date TIMESTAMP PRIMARY KEY, value STRING)
PARTITION BY RANGE (date) EVERY (30, DAY)
```

■ Creating an interval-hash partitioned table

```
Type
  CREATE TABLE [IF NOT EXISTS] table name ( column definition1 [, column definition2 …] )
    [USING TIMESERIES] [WITH (table option)]
    PARTITION BY RANGE (column name of interval partitioning key) EVERY(interval value [, interval unit])
    SUBPARTITION BY HASH(column name of hash partitioning key) SUBPARTITIONS division count
```

- Specify the column which type is BYTE, SHORT, INTEGER, LONG or TIMESTAMP for "column name of interval partitioning key".
- The column specified for "column name of interval partitioning key" is required to be "PRIMARY KEY" or have "NOT NULL" constraint.
- The following values can be specified as the interval value.

Partitioning key type	Possible interval value
BYTE	1～127
SHORT	1～32767
INTEGER	1～2147483647
LONG	1000～9223372036854775807
TIMESTAMP	1～

  - If the column of TIMESTAMP is specified, it is also required to specify the interval unit. 'DAY' is the only value that can be specified as the interval unit.
  - The interval unit cannot be specified for any types other than TIMESTAMP.

- Specify the value from 1 to 1024 for "division count".
- The columns specified as partitioning key cannot be updated.
- See the section "Interval-hash partitioning" in "Technical Reference" for the function of interval-hash partitioning.

(Example)

```
CREATE TABLE myIntervalHashPartition (date TIMESTAMP PRIMARY KEY, value STRING)
PARTITION BY RANGE (date) EVERY (60, 'DAY')
SUBPARTITION BY HASH (value) PARTITIONS 64
```

3.1.3 CREATE INDEX

Create an index.

Type
CREATE INDEX [IF NOT EXISTS] index name ON table name (column name to be indexed);

- An index with the same name as an existing index cannot be created.
- If a transaction under execution exists in a table subject to processing, the system will wait for these to be completed before creating the data.
- Only ASCII alphanumeric characters, the underscore mark ("_"), the hyphen ("-"), the dot ("."), the slash ("/") and the equal ("=") can be used in the index name. ASCII characters are not case-sensitive.
- An index cannot be created on a column of BLOB type.

3.1.4 CREATE USER

Create a general user.

Type
CREATE USER user name IDENTIFIED BY 'password string';

- Can be executed by an administrator user only.
- A user with the same name as an administrator user (admin and system) registered during installation cannot be created.
- Only ASCII alphanumeric characters, the underscore mark ("_"), the hyphen ("-"), the dot ("."), the slash ("/") and the equal ("=") can be used in the user name. ASCII characters are not case-sensitive.
- Only ASCII characters can be used in the password string. The password is case-sensitive.

3.1.5 DROP DATABASE

Delete a database.

Type
DROP DATABASE database name;

- Can be executed by an administrator user only.

- Databases with names starting with “gs#” and those named “public” and “information_schema” cannot be deleted as these are reserved for internal use in GridDB.
- A database containing tables created by a user cannot be deleted.

3.1.6 DROP TABLE

Delete a table.

```
Type
  DROP TABLE table name;
```

- If there is an active transaction involving the table, the table will be deleted only after the transaction is completed.

3.1.7 DROP INDEX

Delete the specified index.

```
Type
  DROP INDEX [IF EXISTS] index name ON table name;
```

- If “IF EXISTS” is specified, nothing will change if no index with the specified name exists.
- If there is an active transaction involving the table, the table will be deleted only after the transaction is completed.

3.1.8 DROP USER

Delete a general user.

```
Type
  DROP USER user name;
```

- Can be executed by an administrator user only.

3.1.9 SET PASSWORD

Change the password of a general user.

```
Type
  SET PASSWORD [FOR user name] = ‘password string’;
```

- An administrator user can change the passwords of all general users.
- A general user can change its own password only.

3.1.10 ALTER TABLE

Delete data partitions created by table partitioning.

```
Type
  ALTER TABLE table name DROP PARTITION FOR (value included in the data partition )
```

- Data partitions can be deleted only for interval and interval-hash partitioning.
- Specify the value included in the data partition to be deleted.
- Data in the range of the once deleted data partition (from the lower limit value to the upper limit value of the data partition) cannot be registered.
- The lower limit value of a data partition can be checked by metatable. The upper limit value is obtained by adding the interval value to the lower limit value. See “4. Metatable” for the details on the metatable.

Example)

```
○Check the lower limit value of the interval partitioned table “myIntervalPartition1” (partitioning key type: TIMESTAMP, interval: 30DAY)

select PARTITION_BOUNDARY_VALUE FROM "#table_partitions"
where TABLE_NAME='myIntervalPartition1' ORDER BY PARTITION_BOUNDARY_VALUE;
PARTITION_BOUNDARY_VALUE
----------------------------------
  2017-01-10T13:00:00Z
  2017-02-09T13:00:00Z
  2017-03-11T13:00:00Z
    ・・・

○Delete unnecessary data partitions

ALTER TABLE myIntervalPartition1 DROP PARTITION FOR ('2017-01-10T13:00:00Z');
```

- For interval-hash partitioned tables, there are multiple data partitions which have the same lower limit value, and the maximum number of those partitions is equal to the hash division count. Those data partitions are deleted simultaneously. Deleted partitions are checked by metatable. See “4. Metatable” for the details of the metatable.

3.2 Data control language (DCL)

3.2.1 GRANT statement

Assign database access rights to a general user.

Type
```
GRANT ALL ON database name TO user name;
```

- Can be executed by an administrator user only.
- Access to a single database is restricted to a single general user only.

3.2.2 REVOKE statement

Revoke database access rights from a general user.

Type
REVOKE ALL ON database name FROM user name;

- Can be executed by an administrator user only.

3.3 Data management language (DML)

3.3.1 INSERT statement

Register a row in a table.

Type
{INSERT|INSERT OR REPLACE|REPLACE} INTO table name
{VALUES ({numerical value 1|string 1} [, {numerical value 2|string 2} …]), … | SELECT text};

3.3.2 DELETE statement

Delete a row from a table.

Type
DELETE FROM table name [WHERE extraction condition];

3.3.3 UPDATE statement

Update the rows existing in a table.

Type
UPDATE table name
SET column name 1 = equation 1 [, column name 2 = equation 2 …]
[WHERE extraction condition];

- For partitioned tables, it is impossible to update a value of partition key to a different value using the UPDATE statement. In such a case, INSERT after DELETE.

  Example:

  create table tab(a integer, b string) partition by hash a partitions 5;

  NG : update tab set a = a * 2;

  [240015:SQL_COMPILE_PARTITIONING_KEY_NOT_UPDATABLE] Partitioning column='a' is not datable on executing statement

  OK: update tab set b = ‘XXX’;

3.3.4 SELECT statement

Get data.

Type
SELECT [{ALL|DISTINCT}] column name 1 [, column name 2 …] [FROM phrase]
[WHERE phrase]
[GROUP BY phrase [HAVING phrase]]
[ORDER BY phrase]
[LIMIT phrase [OFFSET phrase]];

Made up of a variety of phrases such as FROM, WHERE, etc.

3.4 Phrase

3.4.1 FROM phrase

Specify the table name to perform SELECT.

Type
  FROM table name 1 [, table name 2 …]

3.4.2 GROUP BY phrase

Among the results of the phrases specified earlier, rows having the same value in the specified column will be grouped together.

Type
  GROUP BY column name 1 [, column name 2 …]

3.4.3 HAVING phrase

Perform filtering using the search condition on data grouped by the GROUP BY phrase. GROUP BY phrase cannot be omitted.

Type
  HAVING search condition

3.4.4 ORDER BY phrase

Sort search results.

Type
  ORDER BY column name 1 [{ASC|DESC}] [, column name 2 [{ASC|DESC}] …]

3.4.5 WHERE phrase

Apply a search condition on the result of the preceding FROM phrase.

Type
  WHERE search condition

- Search conditions

  Predicates and the SELECT text, etc. can be used for search conditions.

3.4.6 LIMIT phrase/OFFSET phrase

Extract the specified number of data from the specified location.

Type
  LIMIT value 1 OFFSET value 2

Value 1 represents the number of data to extract while value 2 represents the position of the data to extract.

3.5 Predicate

BETWEEN predicate, IN predicate and LIKE predicate can be used except for COMPARE predicate using a comparison operator (=, >, etc.).

3.5.1 BETWEEN predicate

Extract values of the specified range.

Type
Equation 1 [NOT] BETWEEN equation 2 AND equation 3

BETWEEN predicate is true when the following condition is satisfied.

Equation 2 ≦ equation 1 ≦ equation 3

This predicate is true for rows which do not satisfy the condition when NOT is specified.

3.5.2 IN predicate

Extract a set that satisfies that conditions.

Type
Equation 1 [NOT] IN (equation 2 [, equation 3 …])

3.5.3 LIKE predicate

Conduct a comparison of the matching patterns.

Type
Equation [NOT] LIKE character pattern [ESCAPE escape character]

A character pattern is expressed using special characters such as % or _.

%: Any string

_: Any character

If you want to use % and _ as normal characters, write the escape characters in front of the % or _ after specifying the escape characters in the [ESCAPE escape character] format.

3.6 Comment

Comments can be written in a SQL command.
Format: Description at the back of -- (2 hyphens) or enclose with /* */. A new line needs to be returned at the end of the comment.

3.7 Functions

The following functions are available in the SQL commands of GridDB AE.

AVG, GROUP_CONCAT, SUM, TOTAL, EXISTS, ABS, CHAR, COALESCE, IFNULL, INSTR, HEX, LENGTH, LIKE, LOWER, LTRIM, MAX, MIN, NULLIF, PRINTF, QUOTE, RANDOM, RANDOMBLOB, REPLACE, ROUND, RTRIM, SUBSTR, TRIM, TYPEOF, UNICODE, UPPER, ZEROBLOB, NOW, TIMESTAMP, TO_TIMESTAMP_MS, TO_EPOCH_MS, EXTRACT, TIMESTAMPADD, TIMESTAMPDIFF

3.8 Hints

In GridDB AE, specifying the hints indicating the execution plan in the query makes it possible to control the execution plan without changing the SQL statement.

[Points to note]

- This function might change in future release.

3.8.1 Explanation of terms

The following table explains the hint function related terms.

Table 5: List of hint function related terms

Term	Meaning
Hint phrase	Information for controlling the execution plan
Hint	An enumeration of hint phrases

3.8.2 Specifying hints

Write the hint in the block comment of the query to control the execution plan. The block comment for the hint, can only be written immediately before or after the first SELECT (INSERT/UPDATE/DELETE) statement in SQL.

To distinguish a hint comment from regular comments, the block comment for the hint begin with "/*+". The target to give a hint is specified by the object name or alias in parentheses. The targets are separated by either space, tab, or newline.

In the following example, the MaxDegreeOfParallelism hint phrase sets the upper limit of processing thread number to 2, and the Leading hint phrase specifies the table join order.

```
/*+
      MaxDegreeOfParallelism(2)
      Leading(t3 t2 t1)
 */
SELECT *
   FROM t1, t2, t3
      ON t1.x = t2.y and t2.y = t3.z
   ORDER BY t1.x
   LIMIT 10;
```

[Memo]

- In case of using a table with the same name more than once in the query due to a schema difference or multiple use of the same table, distinguish each table by giving an alias to the table.

3.8.3 List of hint phrases

The following table shows the available hint phrases.

Table 6: List of hint phrases

Class	Operation	Description
Parallelism	MaxDegreeOfParallelism(upper limit)	Maximum number of processing threads for one query.

	MaxDegreeOfTaskInput(upper limit)	Maximum number of inputs for one task.
	MaxDegreeOfExpansion(upper limit)	Maximum number of expansion nodes of planning.
Distributed planning	DistributedPolicy(policy)	Selection of distributed planning policy. ‘LOCAL ONLY’ or ‘LOCAL PREFER’ are available.
Scanning method	IndexScan(table)	Index scan is used if possible.
	NoIndexScan(table)	No index scan is used.
Joining method	IndexJoin(table table)	Using index join if possible.
	NoIndexJoin(table table)	No index join is used.
Table joining order	Leading(table table [table …])	Join specified tables in the specified order.
	Leading((table set(*1) table set(*1))) *1: table set = { table or (table set table set) }	Join the first specified table set as the outer table and the second set as the inner table. ('Table set' in expression indicates single table or table set)

3.8.4 Details of hint phrases

This chapter shows details for each category of hint phrases.

3.8.4.1 Parallelism

Control parallelization processing.

- MaxDegreeOfParallelism(upper limit)

  Specify the maximum number of processing threads for one query when processing SQL in a node.
- MaxDegreeOfTaskInput(upper limit)

  Specify the maximum number of inputs for one task.

  It applies to the following processing:

  ➢ UNION ALL processing when scanning the partitioned table
- MaxDegreeOfExpansion(upper limit)

  Specify the maximum number of expansion nodes of planning.

  It applies to the following processing:

  ➢ Push down join optimization processing

3.8.4.2 Distributed planning

Specify distributed planning policy.

- DistributedPolicy(policy)

  Following distributed planning policies are available:

  ➢ 'LOCAL_ONLY'

  The planner uses only the local information of the connected node without distributing. An error occurs if no table found in local.

  ➢ 'LOCAL_PREFER'

The planner gives priority to the local information of the connected node. A remote plan is generated if no table found in local.

3.8.4.3 Scanning method

Specify scanning method.

- IndexScan(table)
  Index scan is used if possible. If it cannot be used, nothing is done.
- NoIndexScan(table)
  No index scan is used.

3.8.4.4 Joining method

Specify which joining method to select for a table combination.

- IndexJoin(table table)
  Index join is used if possible. If it cannot be used, nothing is done.
- NoIndexJoin(table table)
  No index join is used.

3.8.4.5 Table joining order

Specify in what order the tables are joined.

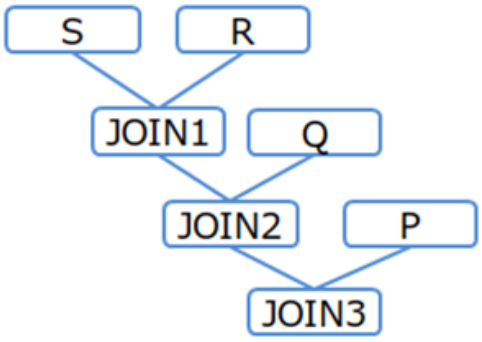
(1) Specify only joining order: Leading(table table [table …])

Specify the table names or aliases in order from the first table to be joined. In this method, using only Left-deep join orders.

An example is shown below:

(Example 1)

```
/*+ Leading(S R Q P) */
SELECT * FROM P,Q,R,S WHERE P.x = Q.x AND ...
```



(No outer or inner table specified)

Figure 1 : Table joining order (Example 1)

(2) Specify joining order and direction: Leading((table set(*1) table set(*1)))

*1: table set = { table or (table set table set) }

In case of specifying only joining order like (1), the joining direction (different for outer table or inner table) may be different from expectation. To fix the joining direction, use the following expression.

In this expression, parentheses can be nested. It joins the first specified table set as the outer table and the second set as the inner table.

Examples are shown below:

(Example 2-1)

```
/*+ Leading(((P Q) R)) */
SELECT * FROM P,Q,R WHERE P.x = Q.x AND ...
```

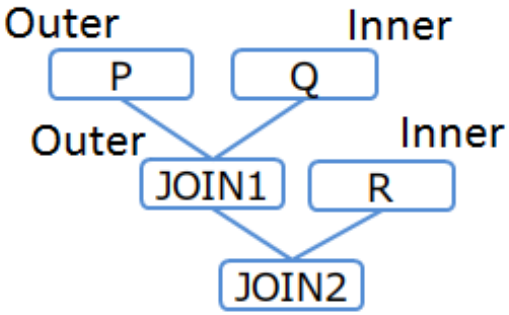


Figure 2 : Table joining order (Example 2-1)

(Example 2-2)

```
/*+ Leading((R (Q P))) */
SELECT * FROM P,Q,R WHERE P.x = Q.x AND ...
```

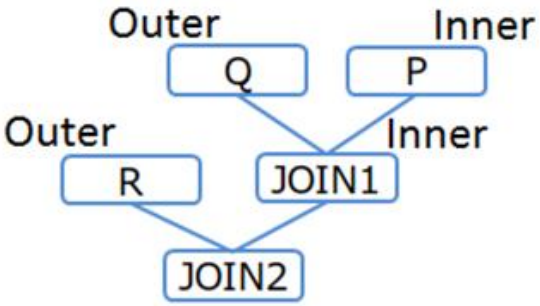


Figure 3 : Table joining order (Example 2-2)

(Example 2-3)

```
/*+ Leading(((P Q) (R S))) */
SELECT * FROM P,Q,R,S WHERE P.x = Q.x AND ...
```

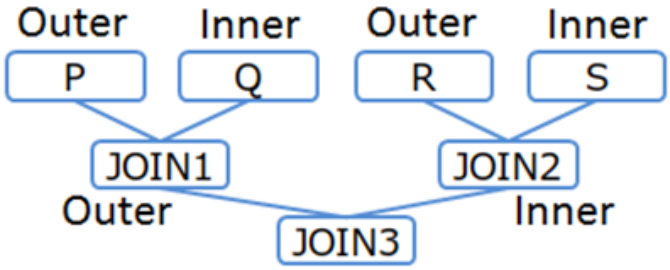


Figure 4 : Table joining order (Example 2-3)

[Memo]

- If three or more tables are joined and there is no join condition between the tables, it is impossible to specify the order by hint.

3.8.5 Error handling

In the following cases, a syntax error occurs.

- Multiple block comments for hints are described
- The hint is described in the wrong position
- There is a syntax error in the description of the hint phrase
- Duplicate hint of the same class are specified for the same table

In the following case, a table specification error occurs:

- The table specification of the hint phrase is incorrect

[Memo]

- When a table specification error occurs, ignore the error hint phrase and execute the query using the others.
- When a syntax error and a table specification error occur at the same time, a syntax error occurs.

4. Metatables

4.1 About metatables

The metatables are tables that are used for checking metadata of data management in GridDB.

[Memo]

・Metatables can only be referred. It is not allowed to register or delete data in the metatables.

[Points to note]

・The schema of metatables may be changed in future version.

4.2 Partitioning data table

Data about partitioned tables can be obtained from this metatable.

Table name

#table_partitions

Schema

Column name	Value	Type
DATABASE_NAME	Database name	STRING
TABLE_NAME	Partitioned table name	STRING
CLUSTER_NODE_ADDRESS	Node address where the internal container (data partition) is placed on. Format Node IP address:Port number	STRING
PARTITION_BOUNDARY_VALUE	The lower limit value of each data partition	STRING

・Each row represents the information of a data partition. For example, when searching rows of a hash partitioned table in which the division count is 128, the number of rows displayed will be 128.

```
//Displaying addresses of nodes where each data partition is placed on.

select DATABASE_NAME, TABLE_NAME, CLUSTER_NODE_ADDRESS from "#table_partitions"

DATABASE_NAME,TABLE_NAME,CLUSTER_NODE_ADDRESS
public,hashTable,10.11.12.1:20001
public,hashTable,10.11.12.2:20001
public,hashTable,10.11.12.4:20001
public,hashTable,10.11.12.1:20001
public,hashTable,10.11.12.3:20001
・・・
```

[Memo]

・In the metatable “#table_partitions”, the other columns may be displayed besides the above columns.

Example)

○Check the number of data partitions

```
select count(*) FROM "#table_partitions"
where TABLE_NAME='myIntervalPartition'
count(*)
-----------------------------------
  8703
```

○Check the lower limit value of each data partition

```
select PARTITION_BOUNDARY_VALUE FROM "#table_partitions"
where TABLE_NAME='myIntervalPartition' ORDER BY PARTITION_BOUNDARY_VALUE;
PARTITION_BOUNDARY_VALUE
-----------------------------------
  2016-10-30T10:00:00Z
  2017-01-29T10:00:00Z
    ・・・
```

[Memo]

It is required to cast the lower limit value to the partitioning key type for sorting by the lower limit value.

Example)

○Check the lower limit value of each data partitions on the interval partitioned table “myIntervalPartition2” (partitioning key type：INTEGER、interval value: 20000)

```
select CAST(PARTITION_BOUNDARY_VALUE AS INTEGER) V from "#table_partitions"
where TABLE_NAME='myIntervalPartition2' ORDER BY V;
PARTITION_BOUNDARY_VALUE
-----------------------------------
-5000
15000
35000
55000
    ・・・
```

A References

- Website of Japan Industrial Standards Committee, http://www.jisc.go.jp/, JISX3005-2 database language SQL, Section 2: Basic Functions (SQL/Foundation)

B Annex: Keywords

The following terms are defined as keywords in the SQL of GridDB AE.

ABORT ACTION AFTER ALL ANALYZE AND AS ASC BEGIN BETWEEN BY CASE CAST COLLATE COLUMN COMMIT CONFLICT CREATE CROSS DATABASE DAY DELETE DESC DISTINCT DROP ELSE END ESCAPE EXCEPT EXCLUSIVE EXISTS EXPLAIN EXTRACT FALSE FOR FROM GLOB GRANT GROUP HASH HAVING HOUR IDENTIFIED IF IN INDEX INITIALLY INNER INSERT INSTEAD INTERSECT INTO IS ISNULL JOIN KEY LEFT LIKE LIMIT MATCH MILLISECOND MINUTE MONTH NATURAL NO NOT NOTNULL NULL OF OFFSET ON OR ORDER OUTER PARTITION PARTITIONS PASSWORD PLAN PRAGMA PRIMARY QUERY RAISE REGEXP RELEASE REPLACE RESTRICT REVOKE RIGHT ROLLBACK ROW SECOND SELECT SET TABLE THEN TIMESTAMPADD TIMESTAMPDIFF TO TRANSACTION TRUE UNION UPDATE USER USING VALUES VIEW VIRTUAL WHEN WHERE WITHOUT XOR YEAR