GMA022J4

GridDB Advanced Edition
SQL リファレンス

東芝デジタルソリューションズ株式会社

は じ め に

本書では、GridDB Advanced Edition における SQL の記述方法および、注意事項について記載しています。
GridDB Advanced Edition / Vector Edition をご使用になる前に、必ずお読みください。
なお、本書で説明する機能は、GridDB Advanced Edition / Vector Edition ライセンスを保有するユーザのみがご利用いただけます。

商標

目次

1. GridDB Advanced Edition とは

GridDB Advanced Edition では、GridDB のデータに SQL でアクセスできるインターフェース(NewSQL インタフェース)を提供します。

本書では、GridDB Advanced Edition（以降　GridDB AE と記載します）のサポートするデータベースにアクセスするための NewSQL インタフェースの SQL について説明します。NoSQL インタフェースとは異なるインタフェースですのでご注意ください。
NewSQL インターフェースは、NoSQL インターフェースで作成したコンテナをテーブルとみなして参照・更新可能です。更新には行の更新だけでなく、コンテナのスキーマや索引の変更を含みます。また、NewSQL インターフェースで作成したテーブルは、コンテナとして NoSQL インターフェースで参照・更新可能です。

なお、NewSQL インタフェースは GridDB Vector Edition（以降　GridDB VE と記載します）でも利用可能です。また、JDBC ドライバの使用方法は、「GridDB Advanced Edition JDBC ドライバ説明書」を、GridDB Vector Edition の拡張 SQL の使用方法は、GridDB Vector Edition のマニュアルをそれぞれ参照してください。

2. SQL 記述形式

本章では、NewSQL インタフェースで使用できる SQL の記述形式について示します。

2.1 使用できる操作

本バージョンでは SELECT 文の他、CREATE TABLE 等の DDL(Data Definition Language、データ定義言語)や INSERT/DELETE 文などをサポートしています。詳細は第 3 章を参照して下さい。

2.2 データ型

2.2.1 データ格納に使用する型

NewSQL インタフェースでデータの格納に使用する型を表 1 に示します。この型名はテーブル作成時にカラム型として記述できます。

表 1：データの格納に使用する型

データ型	内容詳細
BOOL 型	True/False
BYTE 型	-2^7 から 2^7-1 (8 ビット)の整数値
SHORT 型	-2^{15} から $2^{15}-1$ (16 ビット)の整数値
INTEGER 型	-2^{31} から $2^{31}-1$ (32 ビット)の整数値
LONG 型	-2^{63} から $2^{63}-1$ (64 ビット)の整数値
FLOAT 型	単精度型(32 ビット) IEEE754 で定められた浮動小数点数
DOUBLE 型	倍精度型(64 ビット) IEEE754 で定められた浮動小数点数
TimeStamp 型	日付と時刻の組。 日付のみ、時刻のみの値を格納・取得する場合、時刻 00:00:00 または、日付 1970-01-01 が指定されたとみなす。
STRING 型	Unicode コードポイントを文字とする、任意個数の文字の列より構成されるテキスト文字
BLOB 型	画像や音声などのバイナリデータのためのデータ型。入力したままの形式で保存されるラージオブジェクト。文字 x あるいは X をつけて、X'23AB'のような 16 進表現もできます。

また、テーブルに NULL 値を格納することができます。NULL 値に対して“IS NULL”などの演算子を使用すると、SQL 仕様に沿った結果を返却します。

2.2.2　テーブル作成時にカラム型として記述可能な表現

NewSQL インタフェースでは、テーブル作成時にカラム型として記述された型名について、2.2.1 で列挙した型名と一致しなくても、ルールに従って解釈しデータの格納に使用する型を決定します。

以下のルールを上から順にチェックし、合致したルールによってデータ格納に使用する型を決定します。ルールのチェック時には記述した型名およびルールでチェックする文字列の大文字小文字は区別しません。複数のルールに合致した場合はより上にあるルールが優先されます。どのルールにも当てはまらない場合はエラーとなりテーブル作成に失敗します。

ルール No.	テーブル作成時、カラム型として記述した文字列	作成するテーブルのカラム型
1	2.2.1 に列挙した型名	テーブル作成時に指定された型に従う
2	"REAL"	DOUBLE 型
3	"TINYINT"	BYTE 型
4	"SMALLINT"	SHORT 型
5	"BIGINT"	LONG 型
6	"INT"を含む型名	INTEGER 型
7	"CHAR", "CLOB", "TEXT"のどれかを含む型名	STRING 型
8	"BLOB"を含む型名	BLOB 型
9	"REAL", "DOUB"のどれかを含む型名	DOUBLE 型
10	"FLOA"を含む型名	FLOAT 型

上記ルールによるデータ型決定の例を示します。

- 記述した型名が"BIGINTEGER"→INTEGER 型(ルール 6)
- 記述した型名が"LONG"→LONG 型(ルール 1)
- 記述した型名が"TINYINT"→BYTE 型(ルール 3)
- 記述した型名が"FLOAT"→FLOAT 型(ルール 1)
- 記述した型名が"VARCHAR"→STRING 型(ルール 7)
- 記述した型名が"CHARINT"→INTEGER 型(ルール 6)
- 記述した型名が"BIGBLOB"→BLOB 型(ルール 8)
- 記述した型名が"FLOATDOUB"→DOUBLE 型(ルール 9)
- 記述した型名が"INTREAL"→INTEGER 型(ルール 6)
- 記述した型名が"FLOATINGPOINT"→INTEGER 型(ルール 6)
- 記述した型名が"DECIMAL"→エラー

NoSQL インタフェースのクライアントにおけるデータ型と同等の型を NewSQL インタフェースで使用する場合は以下のように記述してください。ただし一部同等の型が存在せず、使用できないものがあります。

表 2：NoSQL インタフェースのクライアントのデータ型と同等のカラム型記述

NoSQL インタフェースのデータ型	同等の型となる NewSQL インタフェースのカラム型記述
STRING(文字列型)	"STRING" または "String 型となる表現"
BOOL(ブール型)	"BOOL"
BYTE(8 ビット整数型)	"BYTE" または "BYTE 型となる表現"
SHORT(16 ビット整数型)	"SHORT" または "SHORT 型となる表現"
INTEGER(32 ビット整数型)	"INTEGER" または "INTEGER 型となる表現"
LONG(64 ビット整数型)	"LONG" または "LONG 型となる表現"
FLOAT(32 ビット単精度浮動小数点数型)	"FLOAT" または "FLOAT 型となる表現"
DOUBLE(64 ビット倍精度浮動小数点数型)	"DOUBLE" または "DOUBLE 型となる表現"
TIMESTAMP(時刻型)	"TIMESTAMP"
GEOMETRY(空間型)	存在しません(使用できません)
BLOB 型	"BLOB" または "BLOB 型となる表現"
配列型	存在しません(使用できません)

2.2.3　コンテナをテーブルとしてアクセスするときのデータ型と値の扱い

NoSQL インタフェースのクライアントで作成したコンテナを、NewSQL インタフェースでテーブルとしてアクセスする場合のコンテナのカラム型および値の扱いを以下に示します。

表 3：コンテナのデータ型と値の NewSQL インタフェースでの扱い

コンテナのカラム型	NewSQL にマッピングされるデータ型	値
STRING 型	STRING 型	元の値と同一
BOOL 型	BOOL 型	元の値と同一
BYTE 型	BYTE 型	元の値と同一
SHORT 型	SHORT 型	元の値と同一
INTEGER 型	INTEGER 型	元の値と同一
LONG 型	LONG 型	元の値と同一
FLOAT 型	FLOAT 型	元の値と同一
DOUBLE 型	DOUBLE 型	元の値と同一
TIMESTAMP 型	TIMESTAMP 型	元の値と同一
GEOMETRY 型	STRING 型	全ての値が NULL
BLOB 型	BLOB 型	元の値と同一
配列型	STRING 型	全ての値が NULL

2.3　ユーザとデータベース

GridDB のユーザには、管理ユーザと一般ユーザの 2 種類があり、利用できる機能に違いがあります。また、データベースを作成することで、利用ユーザ単位にアクセスを分離することができます。ユーザ、データベースの詳細は「テクニカルリファレンス」をご参照ください。

さらに、NewSQL インタフェースでデータを登録、検索するには、データを格納するテーブル（表）を作成する必要があります。登録されるデータは行と呼び、1 個以上の列データから構成されます。

2.4　ネーミングの規則

データベース名・テーブル名・列名および一般ユーザ名は、1 文字以上の ASCII 英数字ならびにアンダースコア「_」、ハイフン「-」、ドット「.」、スラッシュ「/」、イコール「=」の列で構成されます。ただし、名前先頭の文字に数字、または、名前にアンダースコア以外の記号を用いる場合には引用符"で囲んでください。

テーブル名にはノードアフィニティ機能向けに「@」の文字も指定できます。ノードアフィニティ機能については「テクニカルリファレンス」をご参照ください。

ネーミングの規則・制限についての詳細は、「テクニカルリファレンス」の「ネーミングに関する制限」の節をご参照ください。

3. GridDB AE でサポートされる SQL 文

サポートされる SQL 文は、表 4 の通りです。

表 4：サポートされる SQL 文一覧

コマンド	概要
CREATE DATABASE	データベースを作成する。
CREATE TABLE	テーブルを作成する。
CREATE INDEX	索引を作成する。
CREATE USER	一般ユーザを作成する。
DROP DATABASE	データベースを削除する。
DROP TABLE	テーブルを削除する。
DROP INDEX	索引を削除する。
DROP USER	一般ユーザを削除する。
SET PASSWORD	一般ユーザのパスワードを変更する。
GRANT	一般ユーザにデータベースへのアクセス権を設定する。
REVOKE	一般ユーザからデータベースへのアクセス権を削除する。
INSERT	テーブルに行を挿入する。
DELETE	テーブルから行を削除する。
UPDATE	テーブルにある行を更新する。
SELECT	データを取得する。
Comment	コメントを表記する。

本章では、SQL 文の分類ごとに説明を行います。

3.1 データ定義言語(DDL)

CREATE 文、DROP 文などから構成されます。

3.1.1 CREATE DATABASE

データベースを作成します。

形式
```
CREATE DATABASE データベース名;
```

- 管理ユーザのみが実行可能です。
- 「public」、「information_schema」と同一の名前のデータベースは、GridDB の内部用に予約済みのため作成できません。
- 既に同一の名前のデータベースが存在した場合は何も変更しません。
  データベース名に使用できる文字は、ASCII の英数字とアンダースコア「_」、ハイフン「-」、ドット「.」、スラッシュ「/」、イコール「=」です。 ASCII の大文字と小文字は同一視します。

3.1.2 CREATE TABLE

テーブルを作成します。

形式
CREATE TABLE [IF NOT EXISTS] *テーブル名* (列定義※ [, 列定義 …])

○時系列
CREATE TABLE [IF NOT EXISTS] *テーブル名* (*列名* TIMESTAMP PRIMARY KEY [, 列定義 …])
USING TIMESERIES
[WITH (*プロパティキー=プロパティ値* [,*プロパティキー=プロパティ値* …])]

※列定義: 列名 型名 [列制約※]
※列制約: [PRIMARY KEY] (先頭の列のみ指定可)、[NOT NULL]

- "IF NOT EXISTS"が指定された場合、指定したテーブル名と同じ名前のテーブルが存在しないときのみ作成します。
- "列定義"では、列名と型名の指定が必須です。
  - 指定可能な型名は 2.2 データ型を参照してください。
- 列制約は、"PRIMARY KEY"、"NOT NULL"をサポートします。"PRIMARY KEY"は先頭の列のみ指定可能です。
- "USING TIMESERIES"が指定された場合、時系列テーブルを作成します。
  - 先頭の列は TIMESTAMP 型で、"PRIMARY KEY"を指定する必要があります。

  時系列テーブル（時系列コンテナ）については「テクニカルリファレンス」をご参照ください。
- 時系列テーブルの場合、時系列に関するオプションを"WITH (プロパティキー=プロパティ値, ...)"の形式で指定することができます。

機能	内容	プロパティキー	プロパティ値の型
期限解放機能	経過時間	expiration_time	INTEGER 型
	経過単位	expiration_time_unit	STRING 型 (指定可能な値は次の 5 種類。 DAY/ HOUR/ MINUTE/ SECOND / MILLISECOND)
	分割数	expiration_division_count	INTEGER 型

各項目の内容については、「テクニカルリファレンス」をご参照ください。

(例)

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

- テーブル名・列名に使用できる文字は、ASCII の英数字とアンダースコア「_」、ハイフン「-」、ドット「.」、スラッシュ「/」、イコール「=」です。 ASCII の大文字と小文字は同一視されます。
- テーブル名に「@ヒント情報」を付けることでノードアフィニティ機能を使用できます。ノードアフィニティ機能については「テクニカルリファレンス」をご参照ください。

3.1.2.1パーティションテーブルの作成

パーティションテーブルを作成します。

各パーティショニングの機能については、「テクニカルリファレンス」をご参照ください。

■ハッシュパーティションテーブルの作成

形式
```
CREATE TABLE [IF NOT EXISTS] テーブル名 ( 列定義 [, 列定義 …] )
  [USING TIMESERIES [WITH (プロパティキー=プロパティ値, ...)] ]
  PARTITION BY HASH (パーティショニングキーの列名) PARTITIONS 分割数
```

- 指定されたパーティショニングキーの列名と分割数の値により、ハッシュパーティションテーブルを作成します。
- 「分割数」は、1 以上かつ 1024 以下の値を指定してください。
- パーティショニングキーに指定した列の値は、更新できません。
- ハッシュパーティショニングの機能については、「テクニカルリファレンス」の「ハッシュパーティショニング」の節をご参照ください。

(例)

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

■インターバルパーティションテーブルの作成

形式
```
CREATE TABLE [IF NOT EXISTS] テーブル名 ( 列定義 [, 列定義 …] )
  [USING TIMESERIES [WITH (プロパティキー=プロパティ値, ...)] ]
  PARTITION BY RANGE(パーティショニングキーの列名) EVERY(分割幅値 [,単位])
```

- 「パーティショニングキーの列名」には、BYTE 型、SHORT 型、INTEGER 型、LONG 型、TIMESTAMP 型のいずれかの列を指定してください。
- 「パーティショニングキーの列名」に指定する列は、"PRIMARY KEY"または"NOT NULL"制約を指定する必要があります。
- パーティショニングキーに指定した列の値は、更新できません。
- 分割幅値には次の値が指定できます。

パーティショニングキーの型	分割幅値に指定できる値
BYTE 型	1～127
SHORT 型	1～32767
INTEGER 型	1～2147483647
LONG 型	1000～9223372036854775807
TIMESTAMP 型	1 以上

- TIMESTAMP 型の列を指定した場合は、単位を指定する必要があります。単位に指定できる値は、DAY です。
- TIMESTAMP 型以外の列を指定した場合は、単位は指定できません。

- インターバルパーティショニングの機能については、「テクニカルリファレンス」の「インターバルパーティショニング」の節をご参照ください。

(例)

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

■インターバル-ハッシュパーティションテーブルの作成

```
形式
  CREATE TABLE [IF NOT EXISTS] テーブル名 ( 列定義 [, 列定義 …] ) [USING TIMESERIES]
   [WITH (テーブルオプション) ]
   PARTITION BY RANGE(インターバルパーティショニングキーの列名) EVERY(分割幅値 [,単位])
   SUBPARTITION BY HASH(ハッシュパーティショニングキーの列名) SUBPARTITIONS 分割数
```

- 「インターバルパーティショニングキーの列名」には、BYTE 型、SHORT 型、INTEGER 型、LONG 型、TIMESTAMP 型のいずれかの列を指定してください。
- 「インターバルパーティショニングキーの列名」に指定する列には、"PRIMARY KEY"または"NOT NULL"制約を指定する必要があります。
- インターバルパーティショニングの分割幅値には次の値が指定できます。

パーティショニングキーの型	分割幅値に指定できる値
BYTE 型	1～127
SHORT 型	1～32767
INTEGER 型	1～2147483647
LONG 型	1000×ハッシュの分割数～9223372036854775807
TIMESTAMP 型	1 以上

  - TIMESTAMP 型の列を指定した場合は、分割単位を指定する必要があります。分割単位に指定できる値は、'DAY'です。
  - TIMESTAMP 型以外の列を指定した場合は、分割単位は指定できません。

- ハッシュパーティショニングの「分割数」は、1 以上かつ 1024 以下の値を指定してください。
- パーティショニングキーに指定した列の値は、更新できません。
- インターバル-ハッシュパーティショニングの機能については、「テクニカルリファレンス」の「インターバル-ハッシュパーティショニング」の節をご参照ください。

(例)

```
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

索引を作成します。

```
形式
  CREATE INDEX [IF NOT EXISTS] インデックス名 ON テーブル名 ( 索引をつける列名 );
```

- 既に存在する索引と同じ名前の索引は作成できません。
- 処理対象のテーブルにおいて実行中のトランザクションが存在する場合、それらの終了を待機してから作成を行います。
- インデックス名に使用できる文字は、ASCII の英数字とアンダースコア「_」、ハイフン「-」、ドット「.」、スラッシュ「/」、イコール「=」です。 ASCII の大文字と小文字は同一視します。
- BLOB 型の列には索引を作成できません。

3.1.4 CREATE USER

一般ユーザを作成します。

```
形式
  CREATE USER ユーザ名 IDENTIFIED BY ‘パスワード文字列’;
```

- 管理ユーザのみが実行可能です。
- インストール時に登録済の管理ユーザ(admin および system)と同一の名前のユーザは作成できません。
- ユーザ名に使用できる文字は、ASCII の英数字とアンダースコア「_」、ハイフン「-」、ドット「.」、スラッシュ「/」、イコール「=」のみです。 ASCII の大文字と小文字は同一視します。
- パスワード文字列に使用できる文字は、ASCII 文字のみです。大文字と小文字は区別します。

3.1.5 DROP DATABASE

データベースを削除します。

```
形式
  DROP DATABASE データベース名;
```

- 管理ユーザのみが実行可能です。
- 「public」、「information_schema」という名前のデータベース、及び「gs#」で始まる名前のデータベースは、GridDB の内部用に予約済みのため削除できません。
- ユーザが作成したテーブルが存在するデータベースは削除できません。

3.1.6 DROP TABLE

テーブルを削除します。

```
形式
  DROP TABLE テーブル名;
```

- 処理対象のテーブルにおいて実行中のトランザクションが存在する場合、それらの終了を待機してから削除を行います。

3.1.7 DROP INDEX

指定された索引を削除します。

```
形式
  DROP INDEX [IF EXISTS] インデックス名 ON テーブル名;
```

- “IF EXISTS”が指定された場合、指定した名前の索引が存在しない場合は何も変更しません。
- 処理対象のテーブルにおいて実行中のトランザクションが存在する場合、それらの終了を待機してから削除を行います。
- NoSQL インターフェースから名前無しで付与した索引を DROP INDEX で削除することはできません。

3.1.8 DROP USER

一般ユーザを削除します。

```
形式
  DROP USER ユーザ名;
```

- 管理ユーザのみが実行可能です。

3.1.9 SET PASSWORD

一般ユーザのパスワードを変更します。

```
形式
  SET PASSWORD [FOR ユーザ名] = ‘パスワード文字列’;
```

- 管理ユーザは全ての一般ユーザのパスワードを変更可能です。
- 一般ユーザは自身のパスワードのみ変更可能です。

3.1.10 ALTER TABLE

テーブルパーティショニングで作成されたデータパーティションを削除します。

```
形式
  ALTER TABLE テーブル名 DROP PARTITION FOR (削除するデータパーティションの区間(下限値から上限値)に含まれる値)
```

- インターバルとインターバル-ハッシュパーティショニングの場合のみ、データパーティションを削除できます。
- 削除するデータパーティションの区間(下限値から上限値)に含まれる値を指定してください。

- 一度削除したデータパーティションの区間(下限値から上限値)に該当するデータの新規登録はできません。
- データパーティションの下限値は、メタテーブルで確認できます。上限値は、下限値＋分割幅値の値です。メタテーブルの詳細は、「4　メタテーブル」をご参照ください。

  例)

```
〇インターバルパーティショニングのテーブル「myIntervalPartition1」(パーティショニング
キーの型：TIMESTAMP、分割幅値 30DAY)のデータパーティションの下限値一覧を確認する

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
   ・・・

〇不要なデータパーティションを削除する
ALTER TABLE myIntervalPartition1 DROP PARTITION FOR ('2017-01-10T13:00:00Z');
```

- インターバル-ハッシュパーティショニングの場合は、同じ下限値のデータパーティションが、ハッシュの分割数分(最大)存在します。データパーティションを削除する場合は、それらの同じ下限値をもつデータパーティションは同時に削除されます。削除の確認はメタテーブルで行います。メタテーブルの詳細は「4　メタテーブル」をご参照ください。

```
○インターバルパーティショニングのテーブル「myIntervalHashPartition」(インターバルパ
ーティショニングキーの型：TIMESTAMP、分割幅値 90DAY、ハッシュパーティショニングキーの
分割数 3)のデータパーティションの下限値一覧を確認する
select PARTITION_BOUNDARY_VALUE FROM "#table_partitions"
where TABLE_NAME='myIntervalHashPartition' ORDER BY PARTITION_BOUNDARY_VALUE;
PARTITION_BOUNDARY_VALUE
-----------------------------------
 2016-08-01T10:00:00Z     同じ下限値のデータがハッシュされて 3 つの
 2016-08-01T10:00:00Z     データパーティションに分割されている
 2016-08-01T10:00:00Z
 2016-10-30T10:00:00Z
 2016-10-30T10:00:00Z
 2016-10-30T10:00:00Z
 2017-01-29T10:00:00Z
   ・・・

○不要なデータパーティションを削除する
ALTER TABLE myIntervalHashPartition DROP PARTITION FOR ('2016-09-15T10:00:00Z');

○同じ下限値のデータパーティションが削除される
select PARTITION_BOUNDARY_VALUE FROM "#table_partitions"
where TABLE_NAME='myIntervalHashPartition' ORDER BY PARTITION_BOUNDARY_VALUE;
PARTITION_BOUNDARY_VALUE
-----------------------------------
 2016-10-30T10:00:00Z     '2016-09-15T10:00:00Z'を含む区間
 2016-10-30T10:00:00Z     (下限値'2016-08-01T10:00:00Z')の
 2016-10-30T10:00:00Z     3 つのデータパーティションが削除される
 2017-01-29T10:00:00Z
   ・・・
```

3.2 データ制御言語(DCL)

3.2.1 GRANT 文

一般ユーザにデータベースへのアクセス権を付与します。

```
形式
  GRANT ALL ON データベース名 TO ユーザ名;
```

- 管理ユーザのみが実行可能です。
- 1 つのデータベースにアクセス可能な一般ユーザは 1 ユーザに制限されています。

3.2.2 REVOKE 文

一般ユーザからデータベースへのアクセス権を剥奪します。

```
形式
  REVOKE ALL ON データベース名 FROM ユーザ名;
```

- 管理ユーザのみが実行可能です。

3.3　データ操作言語(DML)

3.3.1　INSERT 文

テーブルに行を登録します。

```
形式
  {INSERT|INSERT OR REPLACE|REPLACE} INTO テーブル名
    {VALUES ( {数値 1|文字列 1} [, {数値 2|文字列 2} …] ), … | SELECT 文};
```

3.3.2　DELETE 文

テーブルから行を削除します。

```
形式
  DELETE FROM テーブル名 [WHERE 抽出条件];
```

3.3.3　UPDATE 文

テーブルに存在する行を更新します。

```
形式
  UPDATE テーブル名
    SET 列名 1 = 式 1 [, 列名 2 = 式 2 …]
    [WHERE 抽出条件];
```

- パーティショニングを設定した場合、UPDATE を使ってパーティションキーになっている項目を別の値に更新することは出来ません。このような場合は、DELETE 後に INSERT を行ってください。

  (例)

  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 文

データを取得します。

```
形式
  SELECT [{ALL|DISTINCT}]  列名 1 [, 列名 2 …] [FROM 句]
    [WHERE 句]
    [GROUP BY 句 [HAVING 句]]
    [ORDER BY 句]
    [LIMIT 句 [OFFSET 句]];
```

FROM 句、WHERE 句など様々な句から構成されます。

3.4　句

3.4.1　FROM 句

SELECT を行うテーブル名を指定します。

形式
　FROM テーブル名 1 [, テーブル名 2 …]

3.4.2　GROUP BY 句

前に指定された句の結果の中で、指定された列で同じ値を持った行をグループ化します。

形式
　GROUP BY 列名 1 [,　列名 2 …]

3.4.3　HAVING 句

GROUP BY 句によりグループ化された情報に対して探索条件で絞り込みを行います。GROUP BY 句は省略できません。

形式
　HAVING 探索条件

3.4.4　ORDER BY 句

検索結果の並べ替え（ソート）を行います。

形式
　ORDER BY 列名 1 [{ASC|DESC}]　[, 列名 2 [{ASC|DESC}] …]

3.4.5　WHERE 句

先行する FROM 句の結果に、探索条件を適用します。

形式
　WHERE 探索条件

- 探索条件

  探索条件は述語や SELECT 文などが使用できます。

3.4.6　LIMIT 句/OFFSET 句

指定した位置から指定した件数分のデータを取り出します。

形式
　LIMIT 値 1 OFFSET 値 2

値 1 は取り出すデータ件数を表し、値 2 は取り出すデータ位置を表します。

3.5　　述語

比較演算子(=、>など)を使った比較述語以外に BETWEEN 述語、IN 述語、LIKE 述語を使うことができます。

3.5.1　BETWEEN 述語

指定した範囲の値を取り出します。

```
形式
  式 1 [NOT] BETWEEN 式 2 AND 式 3
```

BETWEEN 述語が真となるのは、次の条件のときです。

式 2 ≦ 式 1 ≦ 式 3

NOT を指定した場合は、条件を満足しない行に対して、この述語は真になります。

3.5.2　IN 述語

条件を満たす集合を取り出します。

```
形式
  式 1 [NOT] IN ( 式 2 [, 式 3 …] )
```

3.5.3　LIKE 述語

パターン一致比較を行います。

```
形式
  式 [NOT] LIKE 文字パターン [ESCAPE エスケープ文字]
```

文字パターンは、%や_の特殊文字を使って表現します。

%：任意の文字列

_：任意の文字

なお、%や_を通常の文字として使いたい場合には、エスケープ文字を「ESCAPE エスケープ文字」形式で指定した上で、エスケープ文字を%や_の前に記述してください。

3.6　　コメント

SQL 文中にコメントを書くことができます。
書式は、--（ハイフンを 2 つ）の後ろに記述するか、/* */で囲みます。コメントの後ろには改行が必要です。

3.7　　関数

GridDB AE の SQL 文には以下の関数が用意されています。

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　ヒント機能

GridDB AE では、実行計画を示すヒントをクエリに指定することで、SQL 文を変えることなく実行計画を制御できます。

【注意事項】

・　本機能は将来のバージョンで変更される可能性があります。

3.8.1　用語

ヒント機能で用いられる用語を表 5 に示します。

表 5：ヒント機能の用語

用語	説明
ヒント句	実行計画を制御するための情報
ヒント	ヒント句を列挙したもの。実行計画を制御するクエリに指定する。

3.8.2　ヒントの指定方法

実行計画を制御するクエリのブロックコメントの中にヒントを記述します。ヒント用のブロックコメントは、SQL 文中の先頭の SELECT(INSERT/UPDATE/DELETE)句の直前または直後のみ記述できます。通常のコメントと区別するため、ヒント用のブロックコメントは「/*+」で始めます。

ヒントの対象は、ヒント句の括弧内にオブジェクト名または別名で指定します。複数のオブジェクト名を指定する場合、スペース、タブ、改行のいずれかで区切って指定します。

以下の例では、MaxDegreeOfParallelism ヒント句により処理スレッド数上限を 2 に設定するとともに、Leading ヒント句により、テーブル結合順序を指定しています。

```
/*+
    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;
```

【メモ】

・　クエリ中に同一名称のテーブルが複数回現れる場合、それぞれのテーブルに別名をつけて区別してください。

3.8.3　ヒント句一覧

指定できるヒント句の一覧を表 6 に示します。

表 6：ヒント句一覧

分類	命令	説明
並列化	MaxDegreeOfParallelism(並列数上限)	1 クエリの処理スレッド数の上限
	MaxDegreeOfTaskInput(上限数)	1 タスクへの入力の上限
	MaxDegreeOfExpansion(上限数)	プランノードの展開数の上限
分散プランニング	DistributedPolicy(分散プラン方針)	分散プラン方針の指定。'LOCAL_ONLY','LOCAL_PREFER'のいずれかが指定可能
スキャン方式	IndexScan(テーブル)	可能な場合はインデックススキャンを選択する
	NoIndexScan(テーブル)	インデックススキャンを選択しない
結合方式	IndexJoin(テーブル テーブル)	可能な場合はインデックスジョインを選択する
	NoIndexJoin(テーブル テーブル)	インデックスジョインを選択しない
結合順序	Leading(テーブル テーブル[テーブル…])	指定したテーブルを指定した順序で結合する
	Leading((テーブル集合<※1> テーブル集合)) <※1>テーブル集合 = { テーブル or (テーブル集合 テーブル集合) }	1 つ目に指定したテーブル集合を外部表、2 つ目に指定したテーブル集合を内部表として結合する

3.8.4　ヒント句詳細

ヒント句の分類ごとに詳細を説明します。

3.8.4.1 並列化

並列化処理の制御を行います。

- MaxDegreeOfParallelism(並列数)
  あるノードで SQL を処理する場合の、1 クエリあたりの最大並列スレッド数を指定します。
- MaxDegreeOfTaskInput(上限数)
  1 タスクへの入力の上限数を指定します。
  以下の処理に適用されます。
  ・ テーブルパーティショニングの設定されたテーブルをスキャンした場合の UNION ALL 処理

・ MaxDegreeOfExpansion(上限数)
プランノードの展開数の上限を指定します。
以下の処理に適用されます。
・プッシュダウンジョイン最適化処理

3.8.4.2 分散プランニング

分散プランニングの方針を指定します。

・ DistributedPolicy(分散プラン方針)
分散プラン方針として、以下のいずれかが指定可能です。
・'LOCAL_ONLY'
分散プランとして、接続したノードだけで分散させずにプランニングします。ローカルになければエラーとなります。
・'LOCAL_PREFER'
分散プランとして、接続したノードのローカル情報を優先してプランニングします。ローカルになければリモートプランを生成します。

3.8.4.3 スキャン方式

テーブルのスキャン方式を指定します。

・ IndexScan(テーブル)
可能な場合はインデックススキャンを選択します。元々インデックススキャンが不可能な場合は何もしません。
・ NoIndexScan(テーブル)
インデックススキャンを選択しません。

3.8.4.4 結合方式

結合方式を指定します。

・ IndexJoin(テーブル テーブル)
可能な場合はインデックスジョインを選択します。 元々インデックスジョインが不可能な場合は何もしません。
・ NoIndexJoin(テーブル テーブル)
インデックスジョインを選択しません。

3.8.4.5 結合順序

テーブルのジョイン処理でどのような順番で結合するかを指定します。

(1) 結合順序のみ指定: Leading(テーブル テーブル[テーブル...])
先に結合するテーブルから順にテーブル名または別名を指定します。この指定方式の場合、常に Left-deep join で結合されます。

以下に例を示します。

(例 1)

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

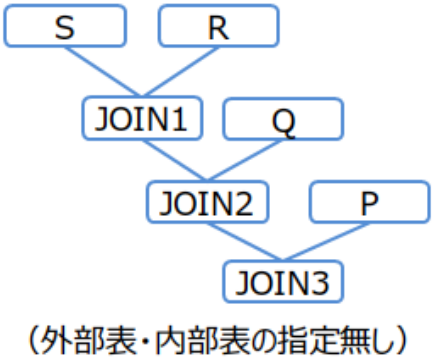

図 1：結合順序(例 1)

(2) 結合方向を含めた指定: Leading((テーブル集合<注 1> テーブル集合<注 1>))

<注 1>テーブル集合 = { テーブル or (テーブル集合 テーブル集合) }

(1)のように結合順序のみを指定した場合、結合方向(外部表/内部表の別)が期待と異なる場合があります。結合方向を固定したい場合は以下の書式を使います。

(書式)

```
/*+ Leading((t1 (t2 t3))) */
SELECT...
```

この書式では、括弧をネストして記述できます。括弧内の 1 つ目に指定したテーブル集合を外部表、2 つ目に指定したテーブル集合を内部表として結合されます。

以下に例を示します。

(例 2-1)

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

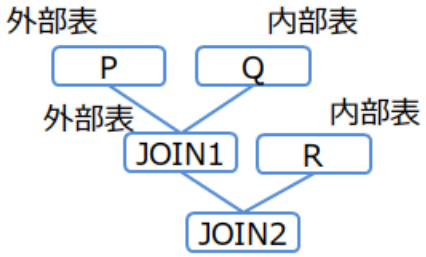

図 2：結合順序(例 2-1)

(例 2-2)

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

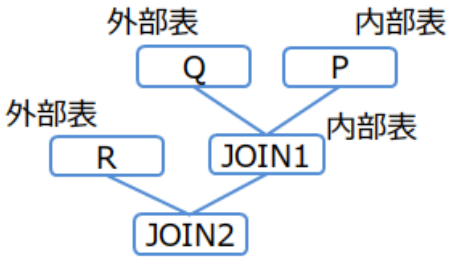

図 3：結合順序(例 2-2)

(例 2-3)

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

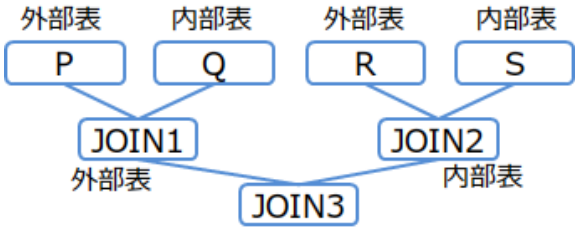

図 4：結合順序(例 2-3)

【メモ】

・ 3 つ以上のテーブルの結合で、テーブル間の結合条件が 1 つもない場合は、ヒントによる順序指定は出来ません。

3.8.5 エラーの扱い

以下の場合は構文エラーとなります。

・ ヒント用のブロックコメントを複数記述した場合
・ ヒントを記述できない位置にヒントを記述した場合
・ ヒント句の記述に構文上の誤りがあった場合
・ 同じテーブルに対して同じ分類のヒント句を重複して指定した場合

以下の場合はテーブル指定エラーとなります。

・ ヒント句対象のテーブル指定に誤りがあった場合

【メモ】

・ テーブル指定エラーが起こった場合、対象のヒント句を無視し、それ以外のヒント句を使ってクエリを実行します。
・ 構文エラーとテーブル指定エラーが同時に起こった場合は構文エラーとなります。

4. メタテーブル

4.1 メタテーブルとは

GridDB の管理用のメタデータを参照することができるテーブル群です。

［メモ］
・メタテーブルは、参照のみ可能です。データの登録や削除はできません。

【注意事項】
・ メタテーブルのスキーマは将来のバージョンで変更される可能性があります。

4.2 パーティショニング情報テーブル

パーティショニングされたテーブルの内部コンテナ(データパーティション)に関する情報を取得することができます。

テーブル名
#table_partitions

スキーマ

列名	内容	型
DATABASE_NAME	データベース名	STRING
TABLE_NAME	パーティショニングされたテーブルの名前	STRING
CLUSTER_NODE_ADDRESS	内部コンテナ(データパーティション)が配置されているノードのアドレス 書式 ノードの IP アドレス:ポート番号	STRING
PARTITION_BOUNDARY_VALUE	データパーティションの下限値	STRING

・ロウ 1 件がひとつのデータパーティションの情報を表します。例えば、分割数 128 のハッシュパーティショニングテーブルの場合、TABLE_NAME にテーブル名を指定して検索するとロウが 128 個表示されます。

例)

```
〇データパーティションの配置ノードのアドレスを表示する
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
・・・
```

［メモ］
・メタテーブル"#table_partitions"では、上記の列以外にも複数の列が表示されます。

例）

```
〇データパーティションの数を確認する
select count(*) FROM "#table_partitions"
where TABLE_NAME='myIntervalPartition';
count(*)
---------------------------------
 8703

〇データパーティションの下限値を確認する
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
  ・・・
```

[メモ]
・下限値でソートする場合、対象のパーティショニングキーの型に合わせてキャストする必要があります

例）

```
〇インターバルパーティショニングのテーブル「myIntervalPartition2」(パーティショニング
キーの型：INTEGER、分割基準値 20000)のデータパーティションの下限値一覧を確認する
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　参考文献

- 日本工業標準調査会ウェブサイト, http://www.jisc.go.jp/, JISX3005-2 データベース言語ＳＱＬ第２部：基本機能（ＳＱＬ／Ｆｏｕｎｄａｔｉｏｎ）

B　付録：予約語

GridDB AE の SQL では、以下の単語が予約語として定義されています。

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

東芝デジタルソリューションズ株式会社