# InfiniDB®
# SQL 構文ガイド

Release : 4.5
Document Version : 4.5-2

InfiniDB SQL 構文ガイド
2014 年 5 月

# 目次

# 1 はじめに

InfiniDBは高パフォーマンスかつスケーラブルなデータウェアハウス用ストレージエンジンです。エンドユーザーおよびアプリケーションによるInfiniDBの使用を有効にするには、InfiniDBをデータベースのフロントエンドと統合する必要があります。このマニュアルでは、InfiniDBおよびMySQL v5.1の複合展開でユーザーが利用可能なSQL構文について説明します。

- データベースの問合せ
- 行の挿入、更新および削除
- オブジェクトの作成、更新および削除

本書では、すべてのMySQLのSQL文について記載しているわけではありません。本書では、InfiniDBを使用するときに有効なデータ型、式および条件について特に説明します。また、InfiniDBに固有のSQL文についても説明します。InfiniDBに固有の文は、本書に記載されていない他のMySQL Oracleコマンドを超える最適なパフォーマンスを実現します。詳細は、「動作モード」の項を参照してください。

## 1.1 対象読者

本書は、InfiniDBを使用してビジネスインテリジェンスレポートに対する問合せおよびデータウェアハウスへの更新を実行するデータベース管理者およびビジネスインテリジェンス管理者を対象としています。

## 1.2 マニュアルリスト

InfiniDB データベースプラットフォームのマニュアルは、様々な読者を対象とした複数のガイドで構成されています。次の表を参照してください。

| マニュアル | 説明 |
|---|---|
| 『InfiniDB 管理者ガイド』 | InfiniDB を管理するための詳細な手順について説明します。 |
| 『InfiniDB 概要』 | 分析用データベース InfiniDB の概要について説明します。 |
| 『InfiniDB 最小推奨仕様ガイド』 | InfiniDB の実装に必要なハードウェアおよびソフトウェアの最小の推奨仕様を示します。 |
| 『InfiniDB インストレーションガイド』 | InfiniDB をインストールするために必要な手順の概要について説明します。 |
| 『InfiniDB マルチ UM 構成ガイド』 | マルチユーザーモジュールの構成情報について説明します。 |
| 『InfiniDB パフォーマンスチューニングガイド』 | 分析用データベース InfiniDB をパラレル化および拡張するためのチューニングに役立つ情報について説明します。 |

## 1.3 マニュアルの入手

英語版のマニュアルは、http://www.infinidb.coで入手することができます。追加の支援が必要な場合はinfinidb_doc@ashisuto.co.jpにご連絡ください。

## 1.4 マニュアルへのフィードバック

マニュアルの改善に向けて、フィードバック、コメントおよび提案をいただけますようお願いいたします。マニュアル名、バージョンおよびページ番号を添えてコメントをinfinidb_doc@ashisuto.co.jpにご送付ください。

## 1.5 追加リソース

InfiniDB のインストールおよびチューニング、または InfiniDB を使用したデータの問合せに関して支援が必要な場合は infinidb_doc@ashisuto.co.jp までご連絡ください。

# 2 SQL の概要

この章では、InfiniDBを使用してDDLおよびDML文を実行するのに必要な情報について説明します。また、オブジェクトのネーミングに関するガイドラインを示します。

## 2.1 注意事項

この項では、SQL文の実行において重要な詳細について説明します。特定のSQL文の詳細は「SQL構文」を参照してください。

## 2.2 ネーミング規則

InfiniDBにおけるMySQLオブジェクトの最大長は次のとおりです。

- スキーマ、表および列の名前:128文字
- InfiniDBの内部使用との完全な互換性を確保するため、すべての表と列の名前の最初の文字には「a-z」の文字を指定してください。
- オブジェクト名では空白はサポートされていません。
- オブジェクト名ではUTF-8文字はサポートされていません。
- SELECT、CHAR、TABLEなどの特定の単語はInfiniDBの予約語であるため、オブジェクトには使用できません。これらの単語がバックティック(’)識別子で囲まれている場合でも、InfiniDBは構文エラーを戻します。
- キーワードDATEは列名として使用できます。

# 3 データ型および関数

この章では、InfiniDBに固有のネーミング規則、データ型および関数について説明します。

## 3.1 データ型

InfiniDBは、次の表に示すANSI-92データ型をサポートします。

表 1:データ型

| データ型 | 列のサイズ | 説明 |
|---|---|---|
| BIGINT | 8 バイト | 大きい整数。スケール 0 の数値。<br><br>最小値/最大値(符号付き):<br>-9,223,372,036,854,775,806 から<br>+9,223,372,036,854,775,807<br>最小値/最大値(符号なし):0 から<br>+18,446,744,073,709,551,613 |
| CHAR | 1、2、4 または 8 バイト | 固定長の文字および特殊文字を保持します。<br>最大長は 255 です。<br>最小サイズは 1 バイト(デフォルト)です。 |
| DATE | 4 バイト | DATE には、年、月および日が含まれます。日付は内部的には 4 バイトの文字列で示されます。YYYY-MM-DD の形式で表示され、最初の 2 バイトは年を、次の 0.5 バイトは月を、次の 0.75 バイトは日を示します。<br><br>サポートされる範囲は 1400-01-01 から 9999-12-31 です。 |
| DATETIME | 8 バイト | 日付および時刻の組合せ。<br><br>サポートされる範囲は 1400-01-01 00:00:00 から 9999-12-31 23:59:59 です。 |
| DECIMAL/NUMERIC | 2、4 または 8 バイト | パック形式の固定小数点数で、特定の総桁数を指定し、小数点の後の桁数を設定できます。指定できる最大精度(総桁数)は 18 です。 |

| | | |
|---|---|---|
| DOUBLE/REAL | 8 バイト | 64 ビットの IEEE-754 浮動小数点形式で格納されます。精度およびスケールを指定する MySQL 拡張はサポートされていません。「REAL」は「DOUBLE」のシノニムです。<br><br>最小値/最大値(符号付き):<br>約-1.797693134862316 +/- e307 から<br>1.797693134862316 +/- e307。<br>最小値/最大値(符号なし):0 から約<br>1.797693134862316 +/- e307。 |
| FLOAT | 4 バイト | 32 ビットの IEEE-754 浮動小数点形式で格納されます。精度およびスケールを指定する MySQL 拡張はサポートされていません。<br><br>最小値/最大値(符号付き):<br>約-3.402823466385289 +/- e38 から<br>3.402823466385289 +/- e38。<br>最小値/最大値(符号なし):0 から約<br>3.402823466385289 +/- e38。 |
| INTEGER/INT | 4 バイト | 通常サイズの整数。スケール 0 の数値。<br><br>最小値/最大値(符号付き):<br>-2,147,483,646 から<br>2,147,483,647<br>最小値/最大値(符号なし):0 から+4294967293 |
| SMALLINT | 2 バイト | 小さい整数。<br><br>最小値/最大値(符号付き):-32,766 から 32,767<br>最小値/最大値(符号なし):0 から 65533 |
| TINYINT | 1 バイト | 非常に小さい整数。スケール 0 の数値。<br><br>最小値/最大値(符号付き):-126 から+127<br>最小値/最大値(符号なし):0 から 253 |
| VARCHAR | 1、2、4 または 8 バイト、あるいは 8 バイトのトークン | 可変長の文字、数字および特殊文字を保持します。<br><br>最大長は 8000 バイトまたは 8000 文字。<br>最小長は 1 バイトまたは 1 文字。 |

| | | |
|---|---|---|
| VARBINARY | 8 バイト | 部分的にサポートします-次の VARBINARY の注意事項を参照してください。<br><br>可変長のバイナリ文字および数字を保持します。<br><br>最大長は 8000 バイト。<br>最小長は 8 バイト。 |

**データ型の注意事項**:

- MyISAMエンジンとは異なり、InfiniDBエンジンは長さが0（ゼロ）の文字をNULL値として扱います。
- InfiniDBエンジンでは、MyISAMエンジンと同様、整数値での「飽和セマンティクス」が有効です。つまり、保持するには大きすぎる値または小さすぎる値（前述の値よりも小さい値または大きい値）を整数フィールドに挿入した場合、InfiniDBは前述に示す適切な最小値または最大値を使用してその値を下限または上限の状態にします。たとえば、符号付きSMALLINT列に32800が指定された場合、実際に挿入される値は32767になります。
- InfiniDBでの符号付きデータ型の最大負数はMySQLでサポートされる数よりも2つ少ないように見えます。これらの2つの最大負数は、InfiniDBによって内部使用に予約されているため、使用できません。たとえば、-128を列に格納する必要がある場合、TINYINTデータ型は使用できません。かわりにSMALLINTデータ型を使用する必要があります。値-128がTINYINT列に挿入されると、InfiniDBはその値を下限である-126にして警告を発行します。
- InfiniDBでの符号なしデータ型の最大正数はMySQLでサポートされる数よりも2つ少ないように見えます。これらの2つの最大正数は、InfiniDBによって内部使用に予約されているため、使用できません。
- DATEおよびDATETIMEデータ型の場合、サポートされる範囲（1400-01-01から9999-12-31）以外の値は、InfiniDBではNULLとして格納されます。
- BIGINT、INTEGER/INT、SMALLINTおよびTINYINTの列に、任意の表示幅を追加できます。MyISAM表と同様、この値はその列の内部ストレージ要件にも、有効な値範囲にも影響しません。
- InfiniDBのすべての列はNULL値をとることができ、すべての列のデフォルト値はNULLです。必要に応じて、任意の列にNOT NULLを指定でき、さらにDEFAULT値も設定できます。
- UTFで文字列の列を作成する場合は、文字数ではなくバイト数で長さを定義します。UTFで10個のマルチバイト文字を格納するには、格納される実際の値に応じて20-30バイトが必要です。詳細は、「UTF-8キャラクタセットの使用」（61ページ）を参照してください。
- InfiniDB では、VARBINARY 列に対する次の機能があります。ここで説明しない動作は、通常の MySQL で同様に動作する場合または動作しない場合があることに注意してください。 InfiniDB VARBINARY の動作は、常に、テスト環境で検証してから、このデータ型を本番環境で使用する必要があります。

  VARBINARY のサポートは、InfiniDB ではデフォルトで無効です。有効にするには、次のコマンドを実行します。

  ```
  /usr/local/Calpont/bin/setConfig WriteEngine AllowVarbinary Yes
  /usr/local/Calpont/bin/calpontConsole RestartSystem
  ```

  VARBINARY 列がある表を作成する場合の構文は、次のとおりです。

  ```
  CREATE TABLE VAR_TABLE (COL1 INT, COL2 VARBINARY(1024)) ENGINE=INFINIDB;
  ```

INSERT を使用して表に VARBINARY データを取り込む場合の構文は、次のとおりです。

```
INSERT INTO VAR_TABLE VALUES (1, 0xABCDEF01);
```

cpimport を使用する場合、インポートファイルの形式は次のようになります。

```
1|ABCDEF01|
```

通常、mysql クライアントを使用して VARBINARY 列を選択して端末に返す場合、バイナリデータが画面に出力されます。このデータを 16 進数形式で表示する場合、SQL プロンプトで次を実行してから select 文を実行します。

```
set infinidb_varbin_always_hex=1;
```

一般的に InfiniDB では、VARBINARY データを表に取り込み、そのまま選択して再度取り出すことしかできません。VARBINARY 列は、結合したり、関数や式に入力して使用したりすることはできません。これらを試みると、通常はエラーメッセージを受信しますが、組込みのエラー確認を回避する方法がある場合があります。この場合、結果は未定義です。

## 3.2 InfiniDB 分散関数

### 3.2.1 分散集計関数

InfiniDBは次の集計関数をサポートします。これらの関数は、SQL文の投影（SELECT）、HAVINGおよびORDER BYの部分で指定できます。

表 2:InfiniDB 分散集計関数

| 関数 | 説明 |
|---|---|
| AVG([DISTINCT] *column*) | 数値データ型列（INT のバリエーション、NUMERIC、DECIMAL）の平均値を戻します。 |
| COUNT (*, [DISTINCT] *column*) | 問合せによって戻される行数。前述のすべてのデータ型がサポートされます。 |
| MAX ([DISTINCT] *column*) | 列の最大値。前述のすべてのデータ型がサポートされます。 |
| MIN ([DISTINCT] *column*) | 列の最小値。前述のすべてのデータ型がサポートされます。 |
| STD()<br>STDDEV()<br>STDDEV_POP() | 数値データ型列（INT のバリエーション、NUMERIC、DECIMAL）の母集団標準偏差を戻します。 |
| STDDEV_SAMP() | 数値データ型列（INT のバリエーション、NUMERIC、DECIMAL）の標本標準偏差を戻します。 |
| SUM([DISTINCT] *column*) | 数値データ型列（INT のバリエーション、NUMERIC、DECIMAL）の合計を戻します。 |
| VARIANCE()<br>VAR_POP() | 数値データ型列（INT のバリエーション、NUMERIC、DECIMAL）の母集団標準分散を戻します。 |
| VAR_SAMP() | 数値データ型列（INT のバリエーション、NUMERIC、DECIMAL）の標本標準分散を戻します。 |

### 3.2.2 分散関数

InfiniDBは次の関数をサポートします。これらの関数は、SQL文の投影（SELECT）、WHERE、GROUP BY、HAVINGおよびORDER BYの部分で指定でき、分散形式で処理されます。

表 3：InfiniDB 分散関数

| 関数 | 説明 |
|---|---|
| &[1] | ビット単位の論理積 |
| ABS()[2] | 絶対値を戻します。 |
| ACOS() | アークコサインを戻します。 |
| ADDTIME() | 時刻を加算します。 |
| ASCII() | 一番左の文字の数値を戻します。 |
| ASIN()[3] | アークサインを戻します。 |
| ATAN()[3] | アークタンジェントを戻します。 |
| BETWEEN...AND... | 値が値範囲内にあるかどうかを確認します。 |
| BIT_AND() | ビット単位の論理積を戻します。 |
| BIT_OR() | ビット単位の論理和を戻します。 |
| BIT_XOR() | ビット単位の排他的論理和を戻します。 |
| CASE() | CASE 演算子。 |
| CAST()<br>CONVERT() | ある型の値をとり、別の型の値を生成します。 |
| CEIL()<br>CEILING() | 引数より大きい値のうち、最小の整数値を戻します。 |
| CHAR() | 渡された各整数の文字を戻します。 |
| CHAR_LENGTH()<br>CHARACTER_LENGTH() | 引数の文字数を戻します。 |
| COALESCE() | 最初の非 NULL 値の引数を戻します。 |
| CONCAT() | 連結した文字列を戻します。 |
| CONCAT_WS() | セパレータのある連結を戻します。 |
| CONV() | 値を異なる進数に変換します。 |
| COS()[3] | コサインを戻します。 |
| COT()[3] | コタンジェントを戻します。 |
| CRC32() | 巡回冗長検査の値を計算します。 |
| DATE() | DATE 式または DATETIME 式の日付部分を抽出します。 |
| DATE_ADD()<br>ADDDATE() | 日付の値に時間の値（間隔）を加算します。 |
| DATE_FORMAT() | 日付を指定したとおりに書式化します。 |
| DATE_SUB()<br>SUBDATE() | 2 つの日付を減算します。 |
| DATEDIFF() | 2 つの日付の差を戻します。 |
| DAY()<br>DAYOFMONTH() | 月の日付（0-31）を戻します。 |
| DAYNAME() | 曜日の名前を戻します。 |
| DAYOFWEEK() | 引数の曜日索引を戻します。 |

| | |
|---|---|
| DAYOFYEAR() | 通日(1-366)を戻します。 |
| DEGREES() | ラジアンを度に変換します。 |
| DIV() | 整数除算。 |
| ELT() | 文字列を索引番号で戻します。 |
| EXP()[3] | 累乗します。 |
| EXTRACT() | 日付の部分を抽出します。 |
| FIND_IN_SET() | 2 番目の引数内にある最初の引数の索引の位置を戻します。 |
| FLOOR() | 引数より小さい値のうち、最大の整数値を戻します。 |
| FORMAT() | 指定した小数点以下の桁数に書式化された数値を戻します。 |
| FROM_DAYS() | 日数を日付に変換します。 |
| FROM_UNIXTIME() | UNIX タイムスタンプを日付で書式化します。 |
| GET_FORMAT() | 日付形式の文字列を戻します。 |
| GREATEST() | 最大の引数を戻します。 |
| GROUP_CONCAT() | 連結した文字列を戻します。 |
| HEX() | 10 進数または文字列の値を 16 進数で戻します。 |
| HOUR() | 時間を抽出します。 |
| IF() | If/else 構文。 |
| IFNULL() | Null If/else 構文。 |
| IN | 値が一連の値内にあるかどうかを確認します。現在、リテラル値のみがサポートされます。 |
| INET_ATON() | IP アドレスに対応する数値を戻します。 |
| INET_NTOA() | 数値に対応する IP アドレスを戻します。 |
| INSERT() | 指定した位置に、指定した文字数まで部分文字列を挿入します。 |
| INSTR() | 最初に出現する部分文字列の索引を戻します。 |
| ISNULL() | 引数が NULL 値かどうかをテストします。 |
| LAST_DAY() | 引数に指定した月の最後の日を戻します。 |
| LCASE() | LOWER()のシノニム。 |
| LEAST() | 最小の引数を戻します。 |
| LEFT() | 左から指定した数の文字を戻します。 |
| LENGTH() | 文字列の長さをバイト単位で戻します。 |
| LIKE | シンプルなパターン一致。 |
| LN()[3] | 引数の自然対数を戻します。 |
| LOCATE() | 最初に出現する部分文字列の位置を戻します。 |
| LOG()[3] | 最初の引数の自然対数を戻します。 |
| LOG2()[3] | 2 を底とする引数の対数を戻します。 |
| LOG10()[3] | 10 を底とする引数の対数を戻します。 |
| LOWER() | 引数を小文字で戻します。 |
| LPAD() | 左側に指定した文字列を埋め込んで文字列の引数を戻します。 |
| LTRIM() | 先頭にある空白を削除します。 |
| MAKEDATE() | 年および通日から日付を戻します。 |

| | |
|---|---|
| MAKETIME() | 時間、分および秒の引数から計算された時間を戻します。 |
| MD5() | MD5 チェックサムを計算します。 |
| MICROSECOND() | 引数のマイクロ秒を戻します。 |
| MID() | 指定した位置から始まる部分文字列を戻します。 |
| MINUTE() | 引数の分を戻します。 |
| MOD() | 剰余を戻します。 |
| MONTH() | 渡された日付の月を戻します。 |
| MONTHNAME() | 月の名前を戻します。 |
| NOW() | 現在の日付および時刻を戻します。 |
| NULLIF() | expr1 が expr2 と等しい場合に NULL を戻します。 |
| PERIOD_ADD() | 年月に期間を加算します。 |
| PERIOD_DIFF() | 期間の月数を戻します。 |
| POSITION() | LOCATE()のシノニム。 |
| POW()/POWER() | 引数を指定した指数に累乗して戻します。 |
| QUARTER() | 日付引数の四半期を戻します。 |
| RADIANS() | 引数をラジアンに変換して戻します。 |
| RAND() | ランダムな浮動小数点の値を戻します。 |
| REGEXP()<br>RLIKE() | 正規表現を使用したパターン一致。 |
| REPEAT() | 指定した回数だけ文字列を繰り返します。 |
| REPLACE() | 指定した文字列を置き換えます。 |
| REVERSE() | 文字列内の文字の順序を逆にします。 |
| RIGHT() | 右から指定した数の文字を戻します。 |
| ROUND() | 引数を四捨五入します。 |
| RPAD() | 指定した回数だけ文字列を追加します。 |
| RTRIM() | 末尾にある空白を削除します。 |
| SECOND() | 秒(0-59)を戻します。 |
| SEC_TO_TIME() | 秒数を「HH:MM:SS」の形式に変換します。 |
| SHA()<br>SHA1() | SHA-1 160 ビットのチェックサムを計算します。 |
| SIGN() | 引数の符号を戻します。 |
| SIN()[3] | 引数のサインを戻します。 |
| SPACE() | 指定した数の空白文字を戻します。 |
| SQRT()[3] | 引数の平方根を戻します。 |
| STR_TO_DATE() | 文字列を日付に変換します。 |
| STRCMP() | 2 つの文字列を比較します。 |
| SUBSTR()<br>SUBSTRING() | 指定した部分文字列を戻します。 |
| SUBSTRING_INDEX() | 指定した数のデリミタまでの文字列から部分文字列を戻します。 |
| SUBTIME() | 時刻を減算します。 |
| SYSDATE() | 関数を実行した時刻を戻します。 |
| TAN()[3] | 引数のタンジェントを戻します。 |

| | |
|---|---|
| TIME() | TIME 式または DATETIME 式の時刻部分を文字列として戻します。 |
| TIMEDIFF() | 2 つの時刻の差を戻します。 |
| TIME_FORMAT() | 時刻として書式化します。 |
| TIME_TO_SEC() | 引数を秒数に変換して戻します。 |
| TIMESTAMPADD() | DATETIME 式に間隔を追加します。 |
| TIMESTAMPDIFF() | DATETIME 式から間隔を減算します。 |
| TO_DAYS() | 日付引数を日数に変換して戻します。 |
| TRIM() | 先頭および末尾にある空白を削除します。 |
| TRUNCATE() | 指定した小数点以下の桁数を切り捨てます。 |
| UCASE() | UPPER()のシノニム。 |
| UNIX_TIMESTAMP() | UNIX タイムスタンプを戻します。 |
| UPPER() | 大文字に変換します。 |
| WEEK() | 週番号を戻します。 |
| WEEKDAY() | 曜日索引を戻します。 |
| WEEKOFYEAR() | 日付の暦週(0-53)を戻します。 |
| XOR() | 排他的論理和。 |
| YEAR() | 年を戻します。 |
| YEARWEEK() | 年および週を戻します。 |

1. 負の値または浮動小数点のデータ型を含む列で使用されるビット単位の論理積では、他のストレージエンジンとは異なる結果が生成される場合があります。
2. これらの機能の詳細は、MySQL関数のリファレンスを参照してください。これらは整数関数であるため、予想可能な結果のみが整数列に生成されます。
3. これらの機能の詳細は、MySQL関数のリファレンスを参照してください。これらは浮動小数点の三角関数であるため、予想可能な結果のみがFLOAT列およびDOUBLE列に生成されます。

### 3.2.3 ウィンドウ関数

ウィンドウ関数は、現在の行と何らかの関係性を持つ行セットに関する計算を実行します。通常は、累積集計、移動集計、センター集計、レポート集計を実施する分析に使用されます。

ウィンドウ関数には他にも次の特性があります。

- 分析関数とも呼ばれます。
- 複数の行が1つの出力行にまとめられることはなく、各行が別個に識別されます。
- 標準のSQL構文の「GROUP BY」句を使用する従来の集計関数とは異なり、ウィンドウ関数は次のように動作します。
    - o 各グループに対して複数の行を戻します。
    - o グループ句に指定された投影リストのすべての列を必要とするわけではありません。

ウィンドウ関数は、問合せで実行される最後の操作セットです(ただし、最後のORDER BY句を除く)。すべての結合およびすべてのWHERE句、GROUP BY句およびHAVING句は、ウィンドウ関数が処理される前に完了されます。このため、ウィンドウ関数はSELECTリストまたはORDER BY句でのみ指定できます。

#### 3.2.3.1 従来の集計とウィンドウ集計

次の図に、従来の集計とウィンドウ集計の相違点をいくつか示します。

| 従来の集計 | ウィンドウ集計 |
|---|---|
| グループを作成して集計 | パーティションとウィンドウフレームを使用して集計 |
| グループごとに 1 行出力 | 入力行ごとに 1 行出力 |
| 各グループを 1 つの方法のみで集計する | 同じパーティションの異なる行が異なるウィンドウフレームを持てる |
| 各 SELECT を 1 つの方法のみでグループ化する | 同じ SELECT による集計で異なるパーティションを使用できる |

#### 3.2.3.2 パーティションおよびフレーム

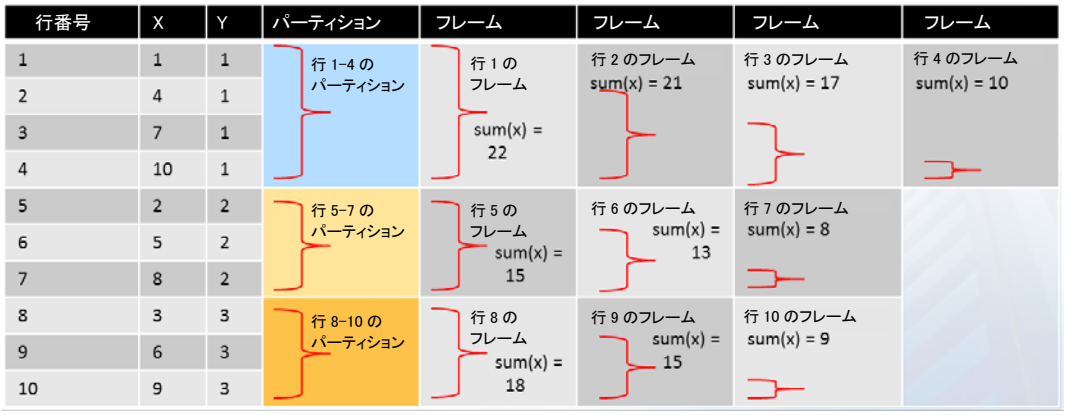
ウィンドウ関数の主な2つの要素は、パーティションとフレームです。

パーティションは、特定の列に、現在の行と同じ値が含まれている行のグループ（ウィンドウ）です。

各行のフレームは、その行のパーティションのサブセットです。各行に対して、スライドする行のフレームが定義されます。このフレームによって、現在の行の計算に使用される行の範囲が決まります。

次の図は、パーティションとフレームのサブセット、および各フレームの結果を示します。次のウィンドウ関数の問合せを前提としています。

```
SELECT x,y,sum(x) OVER (PARTITION BY y RANGE BETWEEN CURRENT ROW AND UNBOUNDED
FOLLOWING) FROM a ORDER BY x, y
```



#### 3.2.3.3 サポートされているウィンドウ関数

次の関数は、SQL文の投影（SELECT）またはORDER BYの部分で指定可能であり、分散形式で処理されます。

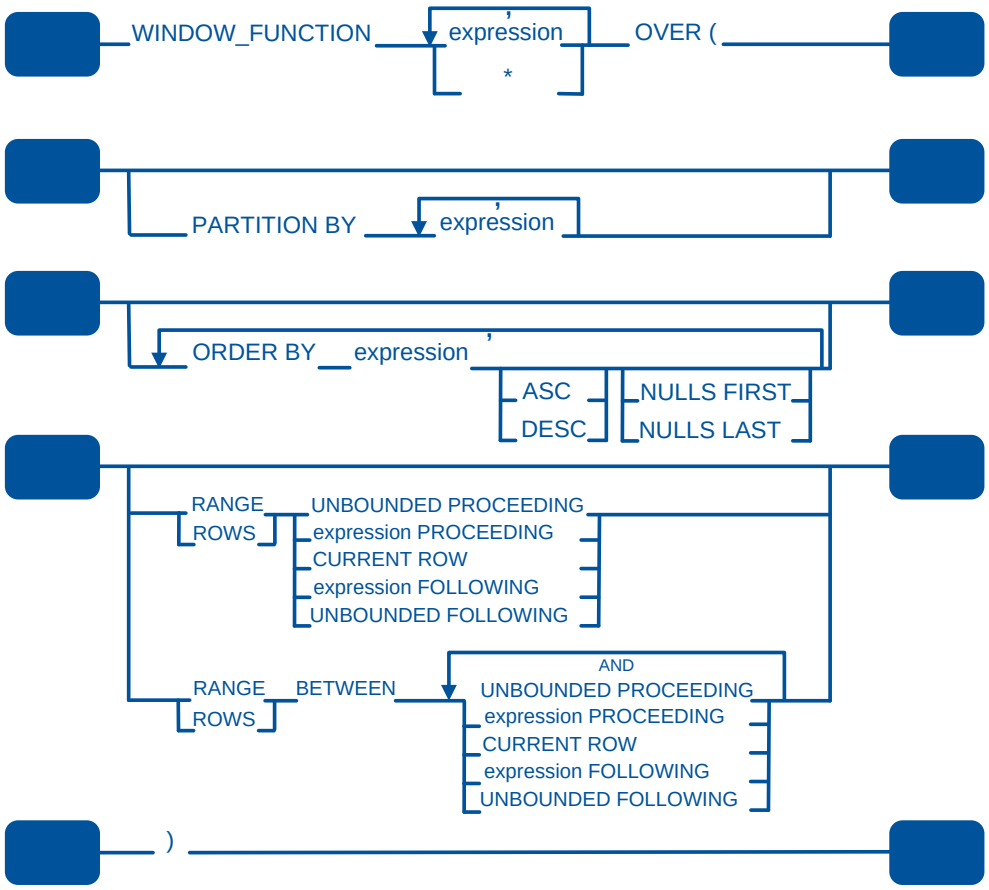
表 4：InfiniDB 分散ウィンドウ関数

| 関数 | 説明 |
|---|---|
| AVG( ) | すべての入力値の平均（算術平均）。 |
| COUNT( ) | 入力行の数。 |
| CUME_DIST( ) | 現在の行から同一パーティション内の他の行までの累積分布（相対ランク）を計算します。ピアまたは先行する行の数 / パーティション内の行の数。 |
| DENSE_RANK( ) | グループ内のアイテムのランク付けをします。同順位がある場合でもランキングシーケンスにギャップは発生しません。 |
| FIRST_VALUE( ) | ウィンドウフレームの最初の行（1からカウントされる）で評価される値。該当する行が存在しない場合はNULLを戻します。 |
| LAG( ) | パーティション内の現在の行より前のオフセット行で評価される値。該当する行がない場合は、デフォルト値を戻します。オフセットとデフォルト値は、どちらも現在の行に対して評価されます。省略すると、オフセットはデフォルトで1になり、デフォルト値はNULLになります。LAGは、自己結合しなくても表の複数の行に同時にアクセスできます。問合せから戻される一連の行およびカーソル位置を指定すると、LAGはその位置より前にある、指定された物理オフセットの行にアクセスします。 |
| LAST_VALUE( ) | ウィンドウフレームの最後の行で評価される値（1からカウントされる）。該当する行が存在しない場合はNULL。 |
| LEAD( ) | 特定の位置より後にある、指定された物理オフセットの行にアクセスします。パーティション内の現在の行より後のオフセット行で評価される値を戻します。該当する行がない場合は、デフォルト値を戻します。オフセットとデフォルト値は、どちらも現在の行に対して評価されます。省略すると、オフセットはデフォルトで1になり、デフォルト値はNULLになります。 |
| MAX( ) | すべての入力値に対する式の最大値。 |
| MEDIAN( ) | 連続分散モデルを前提とした逆分散関数です。数値または日時値をとり、その中央値、または値のソート後に中央値となる補間された値を戻します。計算ではNULLは無視されます。 |
| MIN( ) | すべての入力値に対する式の最小値。 |
| NTH_VALUE( ) | ウィンドウフレームの*n*番目の行で評価される値（1からカウントされる）。該当する行が存在しない場合はNULL。 |
| NTILE( ) | 順序付けられたデータセットを、式で示された数のバケットに分割し、適切なバケット番号を各行に割り当てます。バケットには1から式の値までの番号が付けられます。式の値は、パーティションごとに正の定数に解決される必要があります。1から引数値までの範囲の整数で、パーティションを可能なかぎり均等に分割します。 |
| PERCENT_RANK( ) | 現在の行の相対ランク：（ランク－1）/（合計行数－1）。 |

| | |
|---|---|
| PERCENTILE_CONT() | 連続分散モデルを前提とした逆分散関数です。パーセンタイル値およびソート指定をとり、ソート指定に従ってパーセンタイル値に該当する補間された値を戻します。計算ではNULLは無視されます。<br>構文は、WITHIN GROUP（ORDER BY句）をOVER句よりも前に指定します。 |
| PERCENTILE_DISC() | 不連続分散モデルを前提とした逆分散関数です。パーセンタイル値およびソート指定をとり、そのセットから要素を戻します。計算ではNULLは無視されます。<br>構文は、WITHIN GROUP（ORDER BY句）をOVER句よりも前に指定します。 |
| RANK() | ギャップを含む現在の行のランクで、その最初のピアのROW_NUMBERと同一です。 |
| ROW_NUMBER() | そのパーティションでの現在の行の番号。1からカウントされます。 |
| STDDEV()<br>STDDEV_POP() | 母集団標準偏差を計算し、母集団分散の平方根を戻します。 |
| STDDEV_SAMP() | 標本累積標準偏差を計算し、標本分散の平方根を戻します。 |
| SUM() | すべての入力値に対する式の合計。 |
| VARIANCE()<br>VAR_POP() | 入力値の母集団分散（母集団標準偏差の2乗）。 |
| VAR_SAMP() | 入力値の標本分散（標本標準偏差の2乗）。 |

### 3.2.3.4 ウィンドウ関数の構文

次の図は、ウィンドウ関数で有効な構文を示します。




PARTITION BY

PARTITION BY句の特性:

- 問合せの結果セットを、1つ以上の*expression*に基づいて複数のグループにパーティション化します。
- 同一のSELECT内の複数の分析関数で、異なるパーティションを使用できます。
- *expression*の有効な値は、定数、列、非分析関数、関数式、またはこれらのいずれかを含む式です。
- PARTITION BYで使用される列は、SELECTの投影リストに含まれる必要はありませんが、SELECT問合せのFROM句、WHERE句、GROUP BY句およびHAVING句の適用後に生成される結果セットとして使用できる必要があります。
- PARTITION BYを省略すると、関数は問合せの結果セットのすべての行を単一のグループとして扱います。

ORDER BY

ORDER BY句の特性:

- 分析句でのORDER BYは、データをパーティション内でどのように順序付けるかを指定します。
- パーティション内の値は複数のキーで順序付けることが可能で、それぞれが*expression*で定義され、それぞれが順序付けシーケンスで修飾されます。
- *expression*の有効な値は、定数、列、非分析関数、関数式、またはこれらのいずれかを含む式です。

- ORDER BYで使用される列は、SELECTの投影リストに含まれる必要はありませんが、SELECT問合せのFROM句、WHERE句、GROUP BY句およびHAVING句の適用後に生成される結果セットとして使用できる必要があります。
- 列の別名は、ORDER BYでは使用できません。
- ASC | DESCオプションは、順序付けシーケンス(昇順または降順)を指定します。デフォルトはASCです。
- NULLS FIRST | NULLS LASTオプションは、NULLを含む戻り行を順序付けシーケンスの最初と最後のどちらに出現させるかを指定します。NULLS FIRSTはASC順のデフォルトで、NULLS LASTはDESC順のデフォルトです。

RANGE/ROWS(ウィンドウ句)

RANGE/ROWS句の特性:

- ウィンドウ句は、関数の結果算出に使用するウィンドウ(行の物理セットまたは論理セット)を行ごとに定義します。
- その後、ウィンドウ内のすべての行に対して関数が適用されます。
- ウィンドウは、問合せの結果セットまたはパーティションの先頭から末尾まで移動します。
- ROWS | RANGEはウィンドウを定義します。
    - ORDER BY句が存在する場合にのみ有効です。
    - ROWSは、物理単位(行)でウィンドウを指定します。
        - *expression*は、物理オフセットです。定数または式であり、正の数値に評価される必要があります。
        - *expression*が開始ポイントの一部である場合、終了ポイントの前にある行に評価される必要があります。
- RANGEは、ウィンドウを論理オフセットとして指定します。
    - *expression*は、論理オフセットです。正の数値または期間リテラルに評価される定数または式である必要があります。
    - ORDER BY句で指定できる*expression*は1つだけです。これは、RANGEが列または*expression*に基づく順序に作用するためです。
    - *expression*が数値に評価される場合、ORDER BY *expression*は数値またはDATEデータ型である必要があります。
    - *expression*が期間値に評価される場合、ORDER BY *expression*はDATEデータ型である必要があります。
- BETWEEN ... ANDは、ウィンドウの開始ポイントおよび終了ポイントを定義します。
    - ウィンドウ句でBETWEENを省略して、単一のポイントを指定した場合、このポイントが開始ポイントになり、現在の行が終了ポイントになります。
- UNBOUNDED PRECEDINGは、ウィンドウがパーティションの先頭行から開始されることを示します。これは、終了ポイントの指定としては使用できません。
- UNBOUNDED FOLLOWINGは、ウィンドウがパーティションの最終行で終了することを示します。これは、開始ポイントの指定としては使用できません。
- CURRENT ROWは、ウィンドウが現在の行または値から開始されることを指定します。
- ウィンドウ句を省略すると、デフォルトはRANGE BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWINGになります。

#### 3.2.3.5 ウィンドウ関数の例

例1:関数によるパーティション

各従業員に対して、同じ暦年に入社した他の従業員の数を算出します。

```
SELECT empno, deptno, hiredate,
COUNT(*) OVER (PARTITION BY YEAR(hiredate) ORDER BY hiredate) emp_cnt
FROM emp
ORDER BY hiredate;

EMPNO  DEPTNO  HIREDATE    EMP_CNT
-----  ------  ----------  -------
 7369      20  2008-01-01        2
 7499      30  2008-01-12        2
 7520      30  2009-02-27        9
 7521      30  2009-02-27        9
 7566      20  2009-04-03        9
 7698      30  2009-04-18        9
 7782      10  2009-04-14        9
 7844      30  2009-05-01        9
 7654      30  2009-05-15        9
 7839      10  2009-07-02        9
 7900      30  2009-08-30        9
```

例2:PARTITION BY句の省略

各従業員に対して、部門が30の給与の母集団標準偏差を算出し、入社日で並べ替えます。

```
SELECT last_name, salary,
STDDEV(salary) OVER (ORDER BY hire_date) "StdDev"
FROM employees
WHERE dept_id = 30
ORDER BY last_name, salary, "StdDev“;

LAST_NAME    SALARY      StdDev
-----------  ----------  ----------
Baida              2900    3891.015
Himuro             2600    3891.015
Raphaely          11000    3891.015
```

### 例3:複数の列でのパーティション化

製品の総販売量を、すべての製品およびすべての国に関して、暦年ごとに算出します。また、製品の平均販売量を、すべての製品およびすべての国に関して、すべての年にわたって算出します。

```
SELECT p.prod_id, country_id, calendar_year,
SUM(s.quantity_sold) OVER (PARTITION BY p.prod_id, p.country_id,
   a.calendar_year) units,
AVG(s.amount_sold)  OVER (PARTITION BY p.prod_id, p.country_id)
   avg_sales
FROM sales s, products p
WHERE p.prod_id = s.prod_id;
```

```
PROD     COUNTRY    YEAR     UNITS    AVG_SALES
------   --------   ------   -------  ----------
   147      52782     1999        29      400.02
   147      52782     2000        71      400.02
   147      52785     1999         1     2023.22
   147      52785     2000       139     2023.22
   147      52786     1999         1       12.01
   147      52786     2000         2       12.01
   148      52782     1999       251     6050.14
   148      52782     2000       308     6050.14
   148      52788     1999         4      175.31
   148      52788     2000         8      175.31
```

### 例4:複数のキーでのORDER BY

部門が80の従業員を、給与および歩合に基づいてランク付けします。

```
SELECT department_id, last_name, salary, commission_pct,
RANK() OVER (PARTITION BY department_id
  ORDER BY salary DESC, commission_pct) "Rank"
FROM employees WHERE department_id = 80
ORDER BY department_id, last_name, salary, commission_pct, "Rank";
```

```
DEPARTMENT_ID  LAST_NAME   SALARY   COMMISSION_PCT  Rank
-------------  ----------  -------  --------------  -----
           80  Abel         11000              .3       1
           80  Ande          6400              .1       4
           80  Banda         6200              .1       5
           80  Bates         7300             .15       3
           80  Bates         9500             .25       2
```

### 例5:ROWS UNBOUND PROCEEDING

部門が90で最も給与の低い従業員の名前を取得します。

```
SELECT department_id, last_name, salary,
FIRST_VALUE(last_name)
 OVER (ORDER BY salary ASC ROWS UNBOUNDED PRECEDING) AS lowest_sal
 FROM (SELECT * FROM employees
          WHERE department_id = 90
          ORDER BY employee_id)
 ORDER BY last_name;

DEPARTMENT_ID  LAST_NAME   SALARY    LOWEST_SAL
-------------  ----------  --------  ----------
           90  De Haan        17500     Kochhar
           90  King           24000     Kochhar
           90  Kochhar        17000     Kochhar
```

### 例6:ROWS BETWEEN...AND

従業員表の従業員ごとに、入社日順でその従業員の直前から直後の範囲内に該当し、同じ管理者を上司とする従業員の平均給与を算出します。

```
SELECT manager_id, last_name, hire_date, salary,
  AVG(salary) OVER (PARTITION BY manager_id ORDER BY hire_date
             ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING) AS c_mavg
  FROM employees
  ORDER BY manager_id, hire_date, salary;

MANAGER_ID LAST_NAME                 HIRE_DATE     SALARY     C_MAVG
---------- ------------------------- --------- ---------- ----------
       100 De Haan                   13-JAN-01      17000      14000
       100 Raphaely                  07-DEC-02      11000 11966.6667
       100 Kaufling                  01-MAY-03       7900 10633.3333
       100 Hartstein                 17-FEB-04      13000 9633.33333
       100 Weiss                     18-JUL-04       8000 11666.6667
       100 Russell                   01-OCT-04      14000 11833.3333
       100 Partners                  05-JAN-05      13500 13166.6667
       100 Errazuriz                 10-MAR-05      12000 11233.3333
```

### 3.2.4 情報関数

InfiniDB情報関数は、InfiniDB固有の「メタ」情報を戻す選択可能な疑似関数であり、問合せを特定のノードにローカルに送ることができます。これらの関数は、SQL文の投影（SELECT）、WHERE、GROUP BY、HAVINGおよびORDER BYの部分で指定でき、分散形式で処理されます。

表 5：InfiniDB 情報関数

| 関数 | 説明 |
|---|---|
| `idbBlockId(column)` | 物理行を含むブロックの論理ブロック識別子（LBID）。 |
| `idbDBRoot(column)` | 物理行が存在するDBRoot。 |
| `idbExtentId(column)` | 物理行を含むエクステントの最初のブロックの論理ブロック識別子（LBID）。これは、`editem`ユーティリティで報告される、エクステントの最初の番号と同じです。 |
| `idbExtentMax(column)` | 物理行を含むエクステントのエクステントマップエントリの最大値。 |
| `idbExtentMin(column)` | 物理行を含むエクステントのエクステントマップエントリの最小値。 |
| `idbExtentRelativeRid(column)` | 列のエクステントの列ID（1から8,388,608）。 |
| `idbLocalPm()` | 問合せが開始されたPM。問合せがスタンドアロンのUMから開始された場合、この関数はNULLを戻します。 |
| `idbPartition(column)` | calShowPartitionsなどで使用されるような3つの部分のパーティションID（Directory.SegmentDBRoot）。 |
| `idbPm(column)` | 物理行が存在するPM。 |
| `idbSegmentDir(column)` | 物理行を含む列ファイルの最下位レベルのディレクトリID。 |
| `idbSegment(column)` | 物理行を含むセグメントファイルの数。 |

これらの関数のいくつかの使用例については、『InfiniDB マルチ UM 構成ガイド』の「ローカル PM 問合せの例」の項を参照してください。

## 3.3 ユーザー定義の分散関数

InfiniDBはユーザー定義の分散関数をサポートします。これらの関数は、SQL文の投影（SELECT）、WHERE、GROUP BY、HAVINGおよびORDER BYの部分で指定でき、分散形式で処理されます。

ユーザー定義の関数SDKは、https://github.com/infinidbからInfiniDBソースをダウンロードすることで取得できます。ソースtarファイルを解凍した後、InfiniDBのUDF SDK、手順および例は`utils/udfsdk`ディレクトリに存在します。

## 3.4 非分散の後処理関数

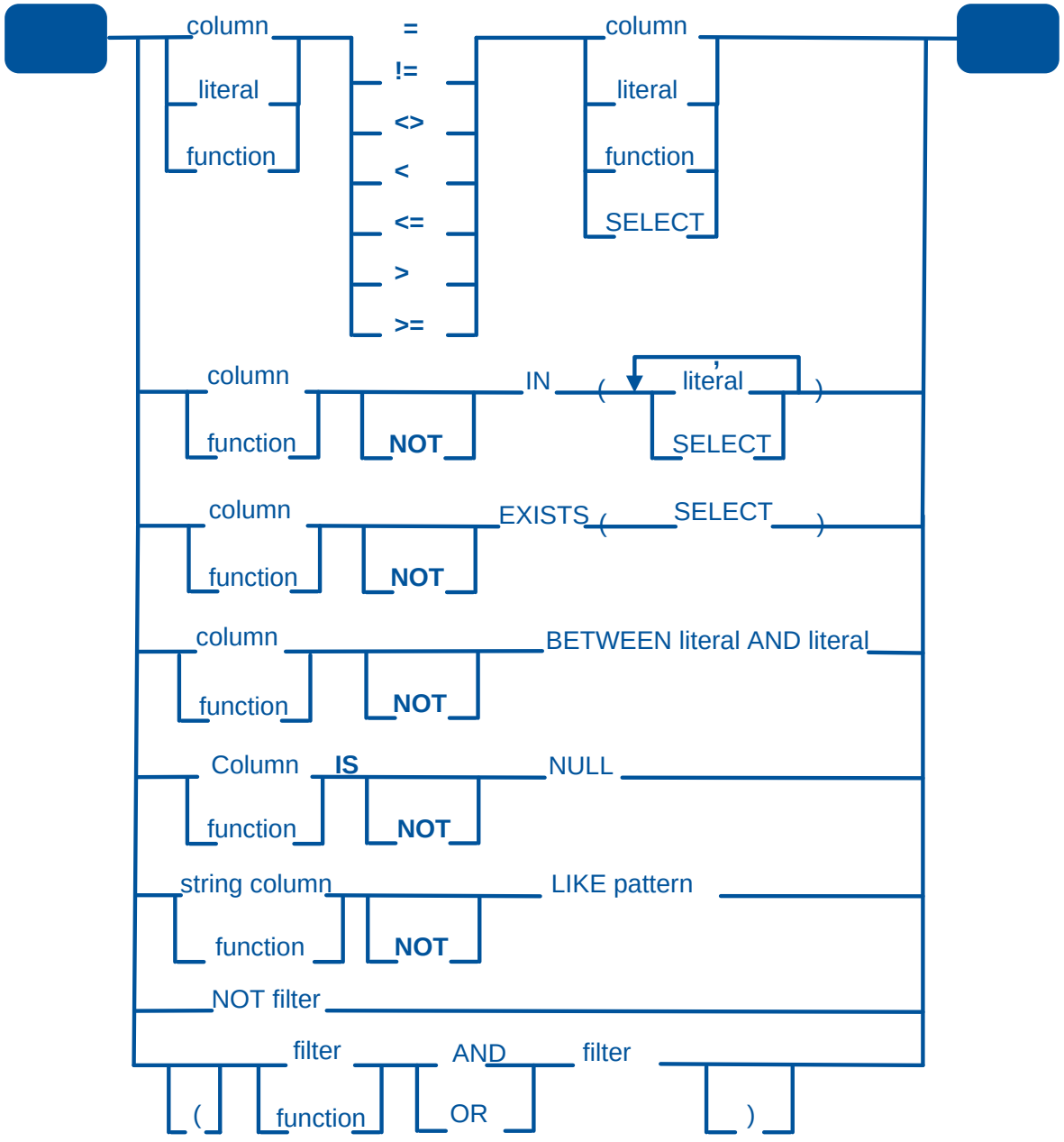
すべてのMySQL関数において、最初にInfiniDBがデータを戻し、MySQLがその戻されたデータに対して関数を実行するという後処理方式を利用できます。これらの関数は、現在、SQL文の投影（SELECT）およびORDER BYの部分でのみサポートされています。

# 4 条件

条件は式および演算子の組合せで、TRUE、FALSEまたはNULLを戻します。

## 4.1 filter

次の図は、InfiniDBで使用可能な条件を示します。




**注意**:「literal」は定数(3など)または定数の値を求める式(100 - (27 * 3))です。日付列では、式のすべての構成要素が定数である場合は、SQLの「interval」構文を使用して日付演算を実行できます('1998-12-01' - interval '1' yearなど)。

### 4.1.1 文字列の比較

MyISAMエンジンとは異なり、InfiniDBエンジンでは、フィルタに使用される文字列の比較で大文字と小文字が区別されます。

最も正確な結果を出力し、不明瞭な結果を回避するため、文字列のフィルタの定数が列幅よりも長くならないようにしてください。

### 4.1.2 パターン一致

LIKE条件で説明するとおり、パターン一致では、任意の1文字に一致させる場合は「_」を、任意の数の文字に一致させる場合は「%」を使用できます。ワイルドカード文字(「%」または「_」)のリテラルインスタンスをテストするには、その前に「¥」記号を付けます。

### 4.1.3 OR 処理

OR処理には次の制限があります。

- ORと組み合せて利用できるのは、リテラルとの列比較のみです。次の問合せは、すべての比較がリテラルに対して行われるため有効です。

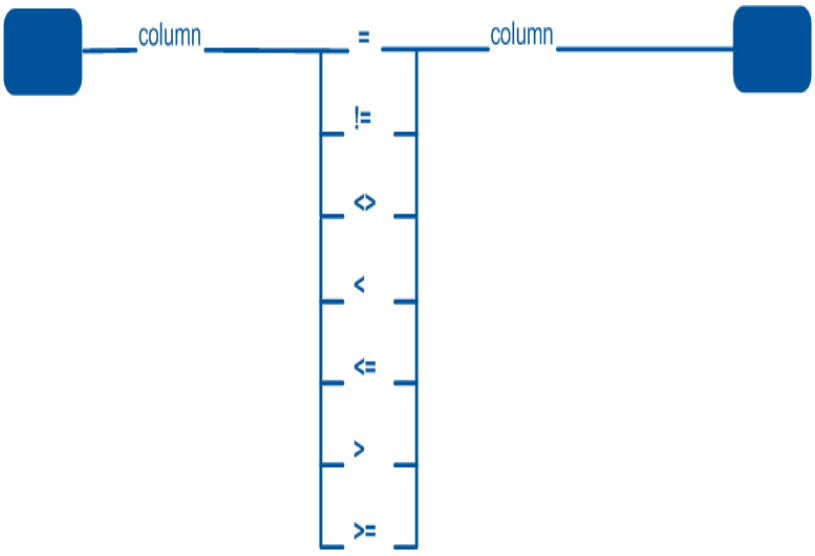
```
SELECT count(*) from lineitem WHERE l_partkey < 100 OR l_linestatus = 'F';
```

- 他のSQLパーサーと同様、InfiniDBではORよりもANDの方がより密にバインドされます。そのため、他のSQLパーサーと同様、OR関係は丸カッコで囲む必要があります。

```
SELECT count(*) FROM orders, lineitem WHERE (lineitem.l_orderkey < 100 OR
lineitem.l_linenumber > 10) AND lineitem.l_orderkey = orders.o_orderkey;
```

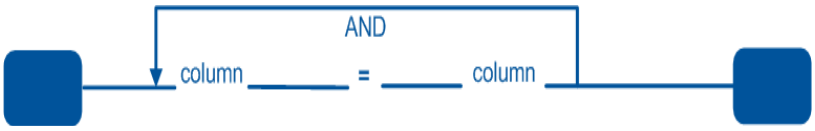
## 4.2 table_filter

次の図は、2つの列に対する条件を実行する場合に使用する条件を示しています。同じ表の列である必要があります。

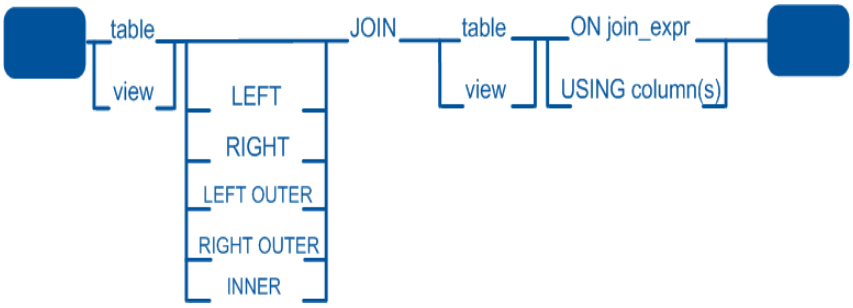


## 4.3 join

次の図は、2つの表で結合を実行する場合に使用する条件を示しています。




## 4.4 join table




**結合の注意事項**:

- InfiniDB表は、非InfiniDB表（MyISAM表）と結合できます。詳細は、『InfiniDB管理者ガイド』のクロスエンジン表アクセスの項を参照してください。
- InfiniDBでは、FROM句の各表の組に対してWHERE句の結合が必要です。デカルト積の問合せは使用できません。
- InfiniDBでは、同じデータ型に対して結合を行う必要があります。また、数値データ型（INTのバリエーション、NUMERIC、DECIMAL）のスケールが同じである場合は、結合で同時に指定できます。
- InfiniDBでは、循環結合はサポートされていません。詳細は、『InfiniDB管理者ガイド』のトラブルシューティングの項を参照してください。

# 5 SQL 構文

InfiniDBはSQL構文をサポートする高パフォーマンスのSQLエンジンです。この章では、INSERT、UPDATEまたはDELETE操作を実行するときに従う必要のある構文について説明します。

本書では、InfiniDBに固有のSELECT構文を示し、非常に高速な問合せの実行について説明します。

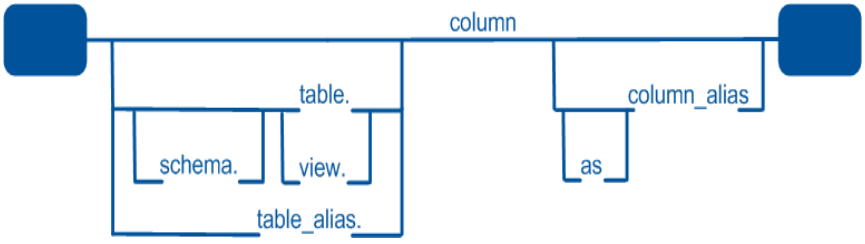
## 5.1 表または列の参照

### 5.1.1 表

次の図は、SQL文で表を参照するときのガイドラインを示します。


### 5.1.2 列

次の図は、SQL文で列を参照するときのガイドラインを示します。

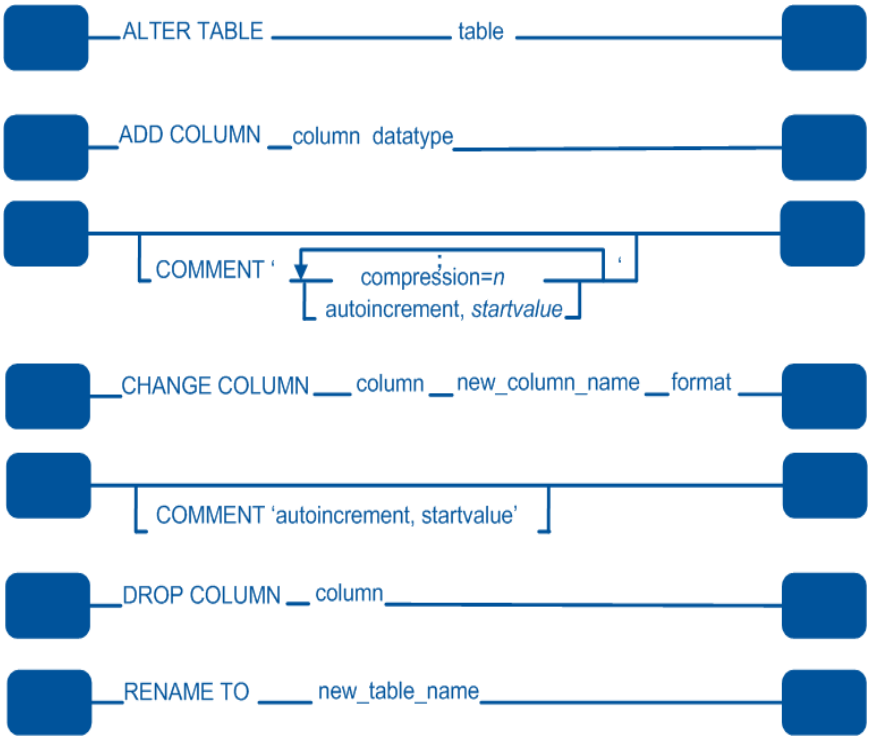


## 5.2 DDL 文

DDL文はデータベースの構造を定義します。この構造には、列および表が含まれます。以下に、サポートされているDDL文を、説明および構文の例とともにアルファベット順で示します。

### 5.2.1 ALTER TABLE

ALTER TABLE文は既存の表を変更します。列の追加、削除および名前の変更と、表の名前の変更が含まれます。




#### 5.2.1.1 ADD

ADD句を使用して表に列を追加します。列名の後にデータ型を指定する必要があります。

ALTER TABLE ADD COLUMNに関する注意事項を次に示します。

- 列レベルの圧縮コメントは、システム、セッションおよび表のデフォルトよりも優先されます。値は次のとおりです。
  0:表または列を圧縮しない
  1:表または列を圧縮する

  圧縮コメントなしで列を追加する場合、表レベルの圧縮コメントが圧縮のデフォルトになります。表レベルの圧縮コメントが存在しない場合は、セッションおよびシステムのデフォルトがそれぞれデフォルトになります。
- 列定義または表レベルのいずれかに「autoincrement」コメントを指定すると、InfiniDBのautoincrementの列が作成されます。autoincrementの列は、各表に1つだけ定義できます。autoincrementの処理の詳細は、「InfiniDBでのautoincrementの使用」の項を参照してください。*startvalue*が指定されていない場合、デフォルトは1です。

次の文は、orders表にINTEGERデータ型のpriority列を追加します。

```
ALTER TABLE orders ADD COLUMN priority INTEGER;
```

オンラインのALTER TABLE ADD COLUMN

InfiniDBエンジンではオンラインDDL（あるセッションで表の問合せを実行中に、別のセッションでその表に列を追加することが可能）を完全にサポートしています。MySQLではサポートされていません。MySQLサーバーでこれが解決されるまでの回避方法を次に示します。この回避方法は、表に列を1つずつ追加する操作を対象としています。その他の目的では使用しないでください。可能なかぎり以下の例に従ってください。

シナリオ：col7という名前のINT列を既存の表fooに追加する

```
select calonlinealter('alter table foo add column col7 int;');
alter table foo add column col7 int comment 'schema sync only';
```

その時点で表に含まれている行数によっては、SELECT文の実行に数十秒かかる場合があります。これを実行している間も、他のセッションはその表に対してSELECT文を実行できます（ただし、新しい列を参照できません）。ALTER TABLE文の実行にかかる時間は1秒未満（MySQLの動作状態に依存）で、この短い間、他のセッションによる表の読取りは実行されません。

#### 5.2.1.2 CHANGE

CHANGE句を使用して表の列の名前を変更します。

CHANGE COLUMNに関する注意事項を次に示します。

- CHANGE COLUMNを使用して列の定義を変更することはできません。符号付きから符号なしへの変更、またはその逆は実行できません。
- 一度に1つの列のみ変更できます。
- 列定義または表レベルのいずれかに「autoincrement」コメントを指定すると、InfiniDBのautoincrementの列が作成されます。autoincrementの列は、各表に1つだけ定義できます。autoincrementの処理の詳細は、「InfiniDBでのautoincrementの使用」を参照してください。autoincrementコメントなしでCHANGE COLUMNを実行するとautoincrement機能が無効になります。

次の例では、orders表のorder_qtyフィールドの名前をquantityに変更します。

```
ALTER TABLE orders CHANGE COLUMN order_qty quantity INTEGER;

ALTER TABLE tpch ADD (priority INT);
```

#### 5.2.1.3 DROP

DROP句を使用して列を削除します。列を削除すると、関連付けされたすべてのデータが削除されます。DROP COLUMN (*column_name*)を実行できます。

次の例では、priority列を削除するようにorders表を変更します。

```
ALTER TABLE orders DROP COLUMN priority;
```

#### 5.2.1.4 RENAME

RENAME句を使用して表の名前を変更します。

次の例ではorders表の名前を変更します。

```
ALTER TABLE orders RENAME TO customer_orders;
```

### 5.2.2 ALTER VIEW

InfiniDBビューの定義を変更します。InfiniDBビューの定義を変更するためにCREATE VIEWまたはREPLACE VIEWを使用することもできます。


次の文は、オープン状態のordersのみ戻すようにv_cust_ordersビューの定義を変更します。

```
ALTER VIEW v_cust_orders (cust_name, order_number, order_status) as select
c.cust_name, o.ordernum, o.status from customer c, orders o where c.custnum =
o.custnum WHERE o.status = 'O';
```

### 5.2.3 COMMIT

COMMIT文は表への変更を確定します。変更したデータの整合性を検証した後に変更をコミットするだけです。

データを一度コミットすると、ROLLBACK文を使用しても元に戻せません。データベースを元の状態に戻すには、バックアップからデータをリストアする必要があります。「ROLLBACK」を参照してください。

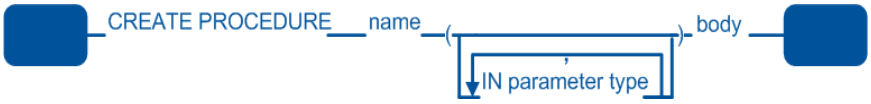

### 5.2.4 CREATE DATABASE

InfiniDBデータベースに、データベースまたはスキーマを作成します。

### 5.2.5 CREATE PROCEDURE

InfiniDBデータベースにストアドルーチンを作成します。

**注意**:現在InfiniDBでは、動作モードが1(VTABLEモード)の場合に入力引数および単一問合せのみのストアドプロシージャを定義できます。ただし、動作モードが0(TABLEモード)の場合は、定義が複雑なストアドプロシージャ(出力パラメータ、宣言、カーソルなど)を処理できます。動作モードの詳細は、「動作モード」を参照してください。




次の文はsp_complex_variableストアドプロシージャを作成して呼び出します。

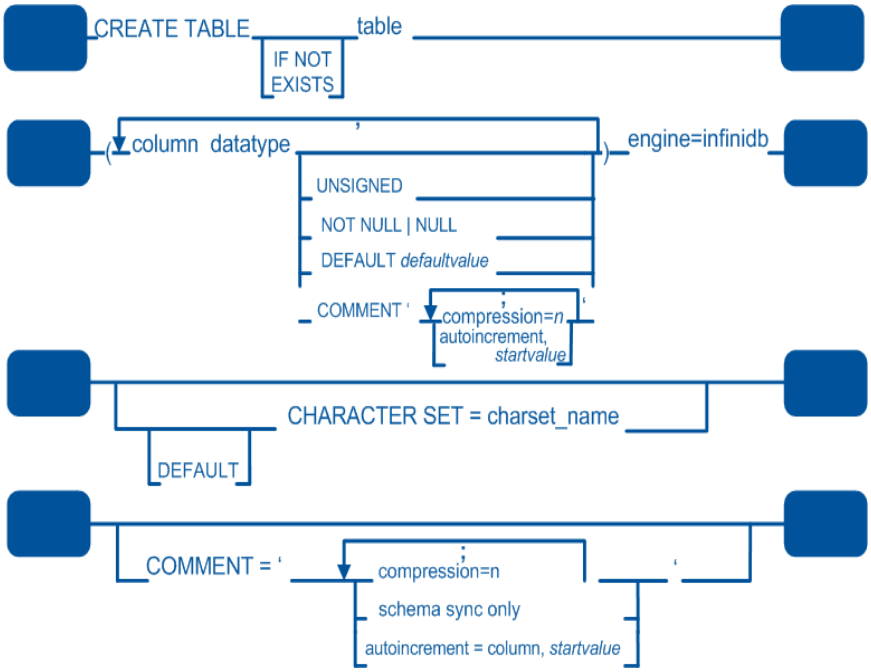
```
delimiter $$
CREATE PROCEDURE sp_complex_variable(in arg_key int, in arg_date date)
    begin
         Select *
         from lineitem, orders
         where o_custkey < arg_key
         and     l_partkey < 10000
         and     l_shipdate>arg_date
         and     l_orderkey = o_orderkey
         order by l_orderkey, l_linenumber;
    end
$$

delimiter ;

call sp_complex_variable(1000, '1998-10-10');
```

### 5.2.6 CREATE TABLE

データベースはユーザーデータを格納する表で構成されます。CREATE TABLE文で複数の列を作成できます。列を追加するときはデータ型と、その後に列名を指定します。




CREATE TABLEに関する注意事項を次に示します。

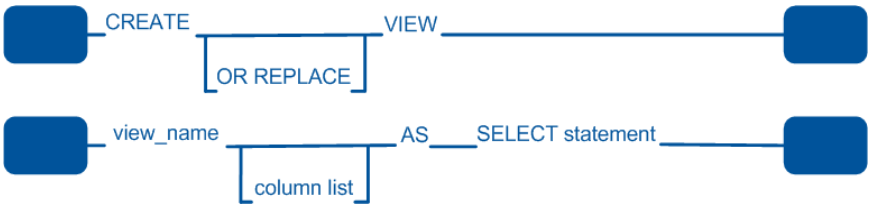
- InfiniDBの表は、mysqlまたはinformation_schemaデータベースに作成しないでください。
- すべてのオブジェクト名はInfiniDB内で小文字で格納されます。
- 現在、CREATE TABLE AS SELECTはInfiniDBでサポートされていません。この構文を使用することはできますが、現在のデフォルトエンジン（MyISAMなど）を使用して表が作成されます。
- 表レベルの「compression」コメントは、システムおよびセッションのデフォルトよりも優先されます。列レベルの「compression」コメントは、システム、セッションおよび表のデフォルトよりも優先されます。値は次のとおりです。
  0：表または列を圧縮しない
  1：表または列を圧縮する
- 「schema sync only」コメントを使用すると、フロントエンドにのみ表を作成することができます。これは、何らかの理由で表がデータベースのバックエンドに存在するがフロントエンドに存在しない場合に使用されます。通常これは、表があるUMで作成されていて、他のUMで同期をとる必要があるという、複数ユーザーモジュール（UM）のシナリオで使用されます。
- 列定義または表レベルのいずれかに「autoincrement」コメントを指定すると、InfiniDBのautoincrementの列が作成されます。autoincrementの列は、各表に1つだけ定義できます。autoincrementの処理の詳細は、「InfiniDBでのautoincrementの使用」を参照してください。*startvalue*が指定されていない場合、デフォルトは1です。
- 外部ツールとの互換性を最大化するために、InfiniDBで次の表宣言を使用できますが、InfiniDB内では実装されません。
  `MIN_ROWS`
  `MAX_ROWS`
  `AUTO_INCREMENT`
- 列のDEFAULT値は、InfiniDBでは64文字に制限されています。

次の文は、データ型が符号なしのINTEGERでNOT NULLを定義したorderkey、データ型がVARCHARのcustomerの2つの列があるordersという表を作成します。値1で始まるautoincrementの列としてorderkeyも定義します。

```
CREATE TABLE orders
    (orderkey INTEGER UNSIGNED NOT NULL,
     customer VARCHAR(45)
    ) ENGINE=INFINIDB
      COMMENT='autoincrement=orderkey,1';
```

### 5.2.7 CREATE VIEW

InfiniDBデータベースにストアドクエリーを作成します。




CREATE VIEWに関する注意事項を次に示します。

- InfiniDBでビューを記述する場合、レポートされる列の型が、元になる表の実際の列の型と一致しない場合があります。これは特に問題ではなく、無視してかまいません。

次の文は、注文および状態を表示するカスタムビューを作成します。

```
CREATE VIEW v_cust_orders (cust_name, order_number, order_status) as
select c.cust_name, o.ordernum, o.status from customer c, orders o where
c.custnum = o.custnum;
```

### 5.2.8 DROP DATABASE

InfiniDBデータベースのデータベースまたはスキーマを削除します。

### 5.2.9 DROP PROCEDURE

DROP PROCEDURE文はInfiniDBデータベースからストアドプロシージャを削除します。


次の文は、sp_complex_variableプロシージャを削除します。

```
DROP PROCEDURE sp_complex_variable;
```

### 5.2.10 DROP TABLE

DROP TABLE文はInfiniDBデータベースから表を削除します。


DROP TABLEに関する注意事項を次に示します。

- RESTRICTを使用すると、フロントエンドの表のみ削除することができます。これは、何らかの理由で表がデータベースのバックエンドに存在しないがフロントエンドに存在する場合に使用されます。通常これは、あるUMで表が削除されていて、他のUMで同期をとる必要があるという、複数ユーザーモジュール(UM)のシナリオで使用されます。

次の文はordersを削除します。

```
DROP TABLE orders;
```

### 5.2.11 DROP VIEW

DROP VIEW文はInfiniDBデータベースからビューを削除します。


次の文はv_cust_ordersビューを削除します。

```
DROP VIEW v_cust_orders;
```

### 5.2.12 RENAME TABLE

RENAME TABLE文はInfiniDBデータベース内の1つ以上の表の名前を変更します。


RENAME TABLEに関する注意事項を次に示します。

- 現在、RENAME TABLEを使用して、あるデータベースから他のデータベースに表を移動できません。
- RENAME TABLEの代替方法については、ALTER TABLEの構文を参照してください。

次の文はorders表の名前を変更します。

```
RENAME TABLE orders TO customer_order;
```

次の文は、orders表およびcustomer表の両方の名前を変更します。

```
RENAME TABLE orders TO customer_orders,
customer TO customers;
```

表を入れ替えるためにRENAME TABLEを使用することもできます。この例では、customer表とvendor表を入れ替えます(temp_tableが存在しないことを想定しています)。

```
RENAME TABLE customer TO temp_table,
vendor TO customer,
temp_table to vendor;
```

### 5.2.13 ROLLBACK

ROLLBACK文は、COMMIT文によってデータベースに永続的に保存されていないトランザクションを元に戻します。

ALTER TABLE、CREATE TABLE、DROP TABLEまたはTRUNCATE TABLE文など、表のプロパティへの変更は元に戻せません。


### 5.2.14 TRUNCATE TABLE

TRUNCATE TABLE文は、InfiniDBデータベースから表を完全に空にします。TRUNCATEは、論理的にDROP TABLEおよびCREATE TABLE文と同じで、表全体を高速に削除できます。


TRUNCATE TABLEに関する注意事項を次に示します。

- このコマンドを使用するユーザーには、DROP権限が必要です。

次の文はorders表を空にします。

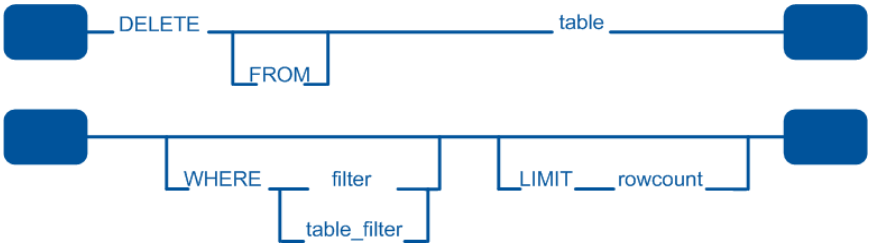
```
TRUNCATE TABLE orders;
```

## 5.3 DML 文

次のDML文は、アプリケーションユーザースキーマから実行されます。DML文は、表内のデータを操作するために使用されます。データの削除、挿入および更新が含まれます。以下に、InfiniDBに固有のDML文を、説明および構文の例とともにアルファベット順で示します。**注意**:現在、LIMITはDML文でサポートされていません。

### 5.3.1 DELETE

DELETE文を使用して、表から行を削除します。




DELETEに関する注意事項を次に示します。

- DELETEは、削除した行によって得られるディスク領域をリカバリしません。消費したディスク領域をリカバリする場合は、TRUNCATE、DROP PARTITION、行の再ロードの組合せ(CREATE TABLEと保持対象の行のみのインポート/挿入、古い表の削除と新しい表の名前の変更)などを使用する、別のオプションがあります。
- DELETEにLIMITオプションを指定すると、削除対象の行の最大数がInfiniDBに通知されてから、制御がクライアントに戻されます。これを利用して、特定のDELETE文の実行時間が長くなりすぎないようにできます。影響を受ける行の数がLIMIT値未満であるまでは、そのDELETE文を単純に繰り返すことができます。

次の文は、顧客キーIDが1001から1999の顧客のレコードを削除します。

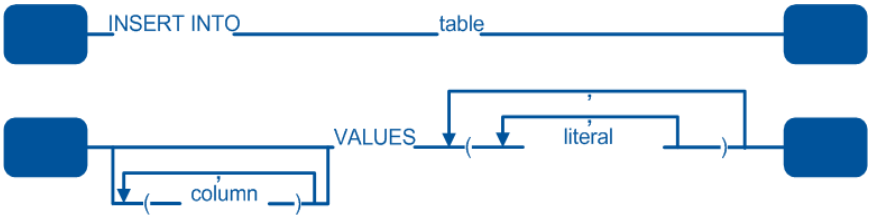
```
DELETE FROM customer WHERE custkey > 1000 and custkey < 2000;
```

次の文は名前が「Widgets」で始まるすべての顧客を削除します。

```
DELETE FROM customer WHERE SUBSTR(name, 1, 7) = 'Widgets';
```

### 5.3.2 INSERT

INSERT文を使用して表にデータを追加します。




次の文は、列値をすべて指定した状態の1行をcustomer表に挿入します。

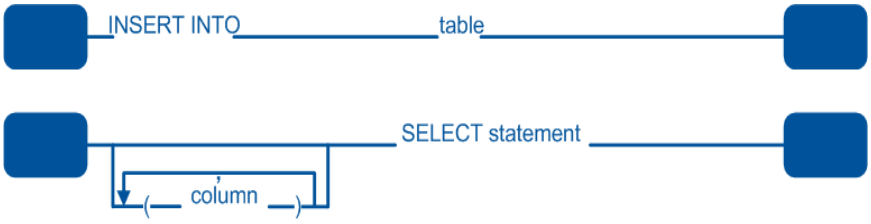
```
INSERT INTO customer (custno, custname, custaddress, phoneno,
cardnumber, comments) VALUES (12, 'John Smith', '100 First Street,
Dallas', '(214) 555-1212', 100, 'On Time');
```

次の文は、列値をすべて指定した状態の2行をcustomer表に挿入します。

```
INSERT INTO customer (custno, custname, custaddress, phoneno,
cardnumber, comments) VALUES (12, 'John Smith', '100 First Street,
Dallas', '(214) 555-1212', 100, 'On Time'), (13, 'John Q Public', '200
Second Street, Dallas', '(972) 555-1234', 200, 'Late Payment');
```

#### 5.3.2.1 INSERT…SELECT

INSERT…SELECT文は1つ以上の表の多数の行を、表に挿入します。




InfiniDBのautoincrementの列は通常通り動作します。
ON DUPLICATE KEY句はInfiniDBで無視されます。

次の文は定常的な顧客をcustomer_ontime表に挿入します。

```
INSERT INTO customer_ontime (custno, custname, custaddress) SELECT
custno, custname, custaddress from customer where comments = 'On Time';
```

### 5.3.3 LOAD DATA INFILE

LOAD DATA INFILE文はテキストファイルから表に高速で行を読み取ります。ファイル名をリテラル文字列として指定する必要があります。

```
LOAD DATA INFILE 'file_name'
    INTO TABLE tbl_name
    [CHARACTER SET charset_name]
    [{FIELDS | COLUMNS}
        [TERMINATED BY 'string']
        [[OPTIONALLY] ENCLOSED BY 'char']
        [ESCAPED BY 'char']
    ]
    [LINES
        [STARTING BY 'string']
        [TERMINATED BY 'string']
    ]
```

次の例では、5列のシンプルな表にデータをロードします。

simpletable.tblという名前のファイルには次のデータが含まれています。

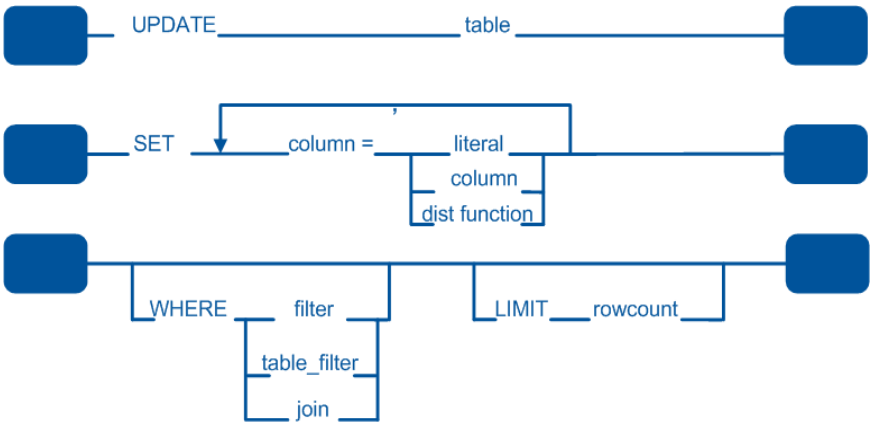
```
1|100|1000|10000|Test Number 1|
2|200|2000|20000|Test Number 2|
3|300|3000|30000|Test Number 3|
```

次の構文を使用してsimpletable表にデータをロードできます。

```
LOAD DATA INFILE 'simpletable.tbl' INTO TABLE simpletable
  FIELDS TERMINATED BY '|';
```

### 5.3.4 UPDATE

UPDATE文は行に格納されているデータを変更します。




**注意**:UPDATEにLIMITオプションを指定すると、更新対象の行の最大数がInfiniDBに通知されてから、制御がクライアントに戻されます。これを利用して、特定のUPDATE文の実行時間が長くなりすぎないようにできます。影響を受ける行の数がLIMIT値未満であるまでは、そのUPDATE文を単純に繰り返すことができます。

次の文は、supplier表内の「WidgetFactory」というサプライヤ名を「WidgetsInc」に更新します。

```
UPDATE supplier SET name = 'WidgetsInc.'WHERE name = 'WidgetFactory';
```

次の文は、予定していた出荷日よりも前に出荷された商品に対し、orders表のdelivery_metフラグを更新します。

```
UPDATE orders SET delivery_met = 'Y' WHERE shipdate < estimated_shipdate;
```

次の文は、「Customer_」と顧客キーを結合したリテラルを顧客名のデフォルトとして設定します。

```
UPDATE customer set name = concat('Customer_', custkey);
```

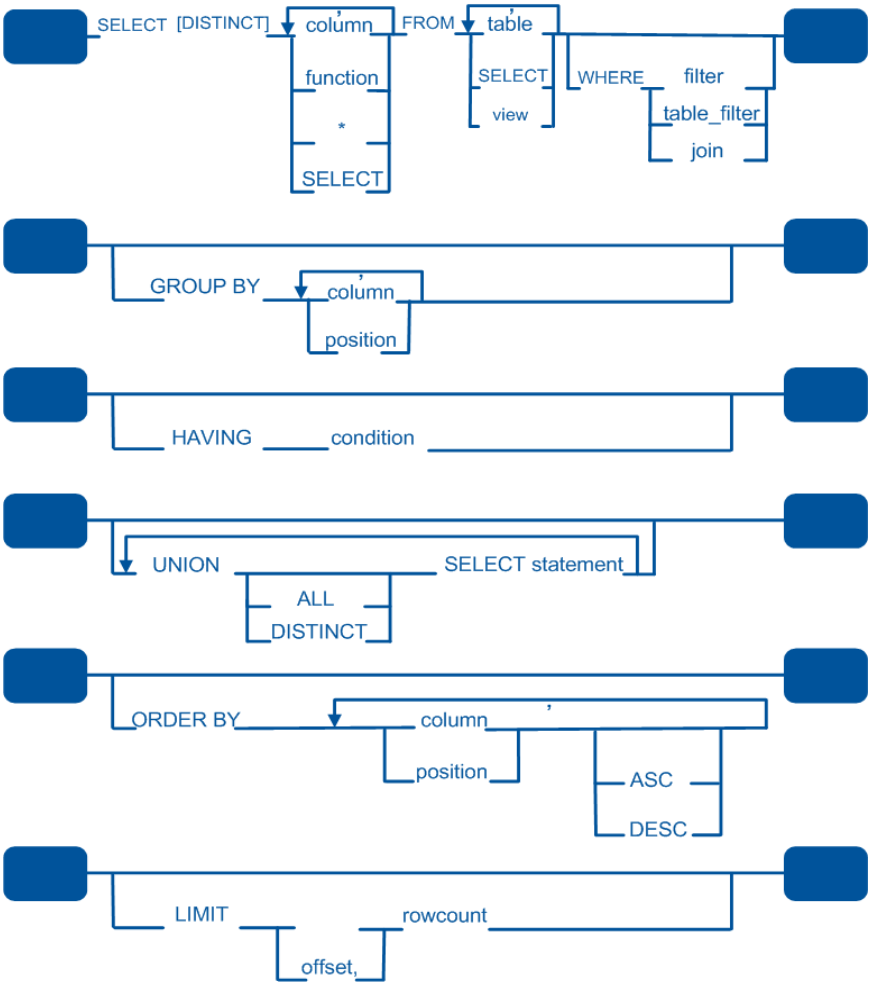
**複数表の構文を使用した UPDATE**

次の例では、複数表の構文形式で表を更新します。

```
update table2, table1 set table2.name=table1.name where table1.id =
   table2.id;
update table2 set name = (select name from table1 where table1.id =
   table2.id);
update table1 join table2 on table1.id = table2.id set table1.name =
   table2.name where table1.id = table2.id;
```

## 5.4 SELECT

SELECT文を使用して、データベースへの問合せを行い、表のデータを表示します。データをフィルタ処理するための多数の句を追加できます。




### 5.4.1 射影リスト(SELECT)の注意事項

射影リストで同じ列を2回以上参照する必要がある場合、各列に、列の別名を使用した固有の名前を指定する必要があります。

列名の合計の長さには関数の長さが含まれるため、射影リスト内では64文字以下にする必要があります。

### 5.4.2 WHERE

WHERE句を使用して、取得されたデータを条件に基づいてフィルタ処理します。**注意**:WHERE句で列の別名は使用できません。

次の文は、region表内の、**region**が「ASIA」の行を戻します。

```
SELECT * FROM region WHERE name = 'ASIA';
```

### 5.4.3 GROUP BY

GROUP BYは1つ以上の特定の列の値に基づいてデータをグループ化します。

次の文はlineitem表内のorderkeyが1000000未満の行を戻し、数量に基づいてグループ化します。

```
SELECT quantity, count(*) FROM lineitem WHERE orderkey < 1000000 GROUP
BY quantity;
```

次の文は、orders表およびcustomer表から戻されるcustkey行を戻します。

```
SELECT custkey FROM orders INTERSECT SELECT custkey FROM customer;
```

### 5.4.4 HAVING

HAVINGは、GROUP BY句と組み合せて使用します。SELECT文で、GROUP BYによって戻されたレコードをフィルタ処理するために使用します。

次の文は、数量が2500以上の商品の出荷日および数量を戻します。

```
SELECT shipdate, count(*) FROM lineitem GROUP BY shipdate HAVING count(*)
>= 2500;
```

次の文は、partsupp表のpartkeyから、part表のpartkeyを除外して、一意の行を戻します。

```
SELECT partkey FROM partsupp MINUS SELECT partkey FROM part;
```

### 5.4.5 ORDER BY

ORDER BY句は、結果を特定の順序で表示します。

注意:ORDER BY句はMySQLで後処理される文です。ORDER BYは、MySQLによって後処理される句のため、問合せの最も外側に位置します。ORDER BY句を含む副問合せでは正しいデータを戻しますが、必ずしも正しい順序で戻しません。

次の文は、lineitem表のquantity列を並べ替えて戻します。

```
SELECT quantity FROM lineitem WHERE orderkey < 1000000 order by quantity;
```

次の文は、lineitem表のshipmode列を並べ替えて戻します。

```
Select shipmode from lineitem where orderkey < 1000000  order by 1;
```

### 5.4.6 UNION

複数のSELECT文の結果を、1つの結果セットに結合するために使用します。

UNION句またはUNION DISTINCT句は、複数の問合せの結果を1つにまとめ、重複した結果を排除します。UNION ALL句は複数の問合せの結果を表示しますが、重複した結果を排除しません。

次の文はpart表およびpartno表のp_name行を戻し、重複した結果を排除します。

```
SELECT p_name FROM part UNION select p_name FROM partno;
```

次の文は、part表およびpartno表のすべてのp_name行を戻します。

```
SELECT p_name FROM part UNION ALL select p_name FROM partno;
```

### 5.4.7 LIMIT

SELECT文から戻される行数を制限するために使用します。LIMITには、引数を2つまで指定できます。LIMITではrowcountを必ず指定し、必要に応じて、戻す最初の行のオフセットを指定します（初期値の行は0です）。

次の文は、customer表から5つの顧客キーを戻します。

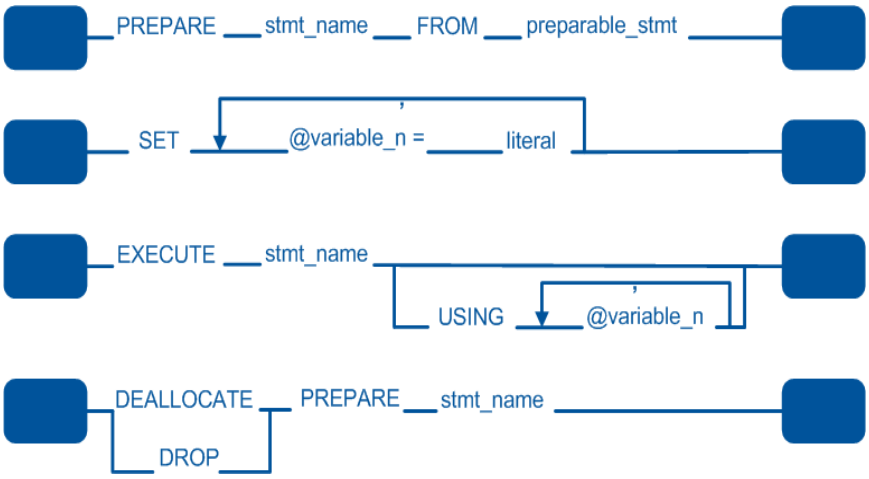
```
SELECT custkey from customer limit 5;
```

次の文は、customer表のオフセット1000から開始して5つの顧客キーを戻します。

```
SELECT custkey from customer limit 1000,5;
```

## 5.5 準備された文

準備された文を使用して、SQL文を、実行先のサーバーに送信します。準備された文のSQL構文は、PREPARE、SET、EXECUTEおよびDEALLOCATE PREPARE（またはDROP PREPARE）の4つのSQL文で構成されます。



### 5.5.1 PREPARE

PREPARE文は文を準備して、後でその文を参照するときに使用するstmt_nameという名前に割り当てます。1つのSQL文で表します。文の中で、「?」文字は後で文を実行するときに問合せにバインドされるデータ値を示すパラメータとして使用されます。「?」文字は、文字列値にバインドする場合でも引用符で囲まないでください。

```
PREPARE test1 FROM
  "SELECT
     SUM(L_EXTENDEDPRICE*L_DISCOUNT) AS REVENUE
   FROM
     LINEITEM
   WHERE
     L_SHIPDATE >= ?AND
     L_SHIPDATE < ?+ interval '1' year AND
     L_DISCOUNT BETWEEN ?- 0.01 AND ?+ 0.01 AND
     L_QUANTITY < ?;";
```

### 5.5.2 SET

SET文はPREPARE文で使用される変数を初期化します。

```
SET @v1=date '1994-01-01';
SET @v2=0.06;
SET @v3=24;
```

### 5.5.3 EXECUTE

PREPAREを使用して文を準備した後、その文の名前を参照するEXECUTE文を使用して実行します。準備された文にパラメータが含まれる場合、パラメータにバインドされる値を含むユーザー変数を指定するUSING句を使用する必要があります。パラメータ値はユーザー変数によってのみ指定されます。文のUSING句では、多数の変数をパラメータマーカーの番号で正確に指定する必要があります。

準備された文を複数回実行できます。SET文を使用して、各実行前にパラメータ値を変更することも、前回設定されたパラメータ値を使用することもできます。

```
EXECUTE test1 using @v1, @v1, @v2, @v2, @v3;
```

### 5.5.4 DEALLOCATE PREPARE または DROP PREPARE

準備された文を削除します。

```
DROP PREPARE test1;
```

# 6 動作モード

InfiniDBは、動作モードを介してすべてのMySQL問合せ構文をサポートします。この動作モードは、インスタンスに対してデフォルトとして設定することも、セッションレベルで設定することもできます。

デフォルトの動作モードの設定については、『InfiniDB管理者ガイド』を参照してください。

セッションレベルで動作モードを設定するには、次のコマンドを使用します。セッションが終了すると、後続のすべてのセッションはインスタンスのデフォルトに戻ります。

```
set infinidb_vtable_mode = n
```

*n*は次のとおりです。

- 0:汎用的で互換性の高い行単位の処理モード。一部のWHERE句コンポーネントはInfiniDBによって処理されますが、結合はネステッドループ結合メカニズムを使用して`mysqld`によってすべて処理されます。
- 1:(デフォルト)InfiniDBによって分散実行との互換性について問合せ構文が評価され、互換性のない問合せは拒否されます。このモードで実行された問合せは分散実行を利用するため、通常、より高いパフォーマンスが確保されます。
- 2:自動スイッチモード。InfiniDBは問合せを内部で処理しようとします。内部で処理できない場合は、行単位のモードで実行するように問合せを自動的に切り替えます。

**注意**:モード0および2でサポートされる問合せ構文の詳細は、『MySQL 5.1リファレンスマニュアル』を参照してください。

# 7 double 演算用の 10 進数

InfiniDB には、中間の 10 進数の算術結果を decimal 型から double 型に変更する機能があります。decimal 型には約 17-18 桁の精度がありますが最大範囲は小さく、double 型には約 15-16 桁の精度がありますが最大範囲はかなり大きくなります（詳細は、前述の「データ型」の項を参照してください）。そのため、適切な設定は用途によって異なります。一般的な数学および科学アプリケーションでは、double 演算を使用して中間結果のオーバーフローを回避する機能の方が、精度が 2 桁上がるよりメリットがあります。ただし、財務アプリケーションでは、デフォルトの 10 進数設定のままにして最下位桁の正確度を確保する方が適切な場合があります。

*infinidb_double_for_decimal_math* 変数が InfiniDB では使用され、10 進数の中間結果のデータ型を制御します。この double 演算用の 10 進数は、この変数のオンとオフを切り替えて、インスタンスに対してデフォルトとして設定したり、セッションレベルで設定したり、文レベルで設定したりすることができます。

インスタンスレベルでのデフォルトのdouble演算用の10進数の設定については、『InfiniDB管理者ガイド』を参照してください。

## 7.1 double 演算用の 10 進数の有効化または無効化

セッションレベルでdouble演算用の10進数の使用を有効または無効にするには、次のコマンドを使用します。セッションが終了すると、後続のすべてのセッションはインスタンスのデフォルトに戻ります。

```
set infinidb_double_for_decimal_math = n
```

*n*は次のとおりです。

- 0（無効）
- 1（有効）

# 8 10 進法

InfiniDBには、10進法の計算で様々な内部精度をサポートする機能があります。

`infinidb_decimal_scale`はInfiniDBエンジンで内部的に使用され、計算された列に対するサブ操作で保持される、小数点の右側の有効桁数を制御します。問合せを実行しているときに「aggregate overflow」というメッセージを受信した場合は、`infinidb_decimal_scale`を減らして問合せの再実行を試みます。`infinidb_decimal_scale`を減らすと、戻された計算列の最下位桁の正確度が低くなる場合があることに注意してください。

`infinidb_use_decimal_scale`は、内部精度での使用の有効および無効を切り替えるためにInfiniDBエンジンによって内部的に使用されます。

この2つのシステム変数は、インスタンスに対してデフォルトとして設定することも、セッションレベルで設定することもできます。

デフォルトの10進法の設定については、『InfiniDB管理者ガイド』を参照してください。

## 8.1 10 進法の有効化または無効化

セッションレベルで10進法の使用を有効または無効にするには、次のコマンドを使用します。セッションが終了すると、後続のすべてのセッションはインスタンスのデフォルトに戻ります。

```
set infinidb_use_decimal_scale = n
```

*n*は次のとおりです。

- 0（無効）
- 1（有効）

## 8.2 10 進法のレベルの設定

セッションレベルで10進法を設定するには、次のコマンドを使用します。セッションが終了すると、後続のすべてのセッションはインスタンスのデフォルトに戻ります。

```
set infinidb_decimal_scale = n
```

*n*は、計算で必要とされる精度です。

# 9 圧縮モード

InfiniDBにはデータを圧縮する機能があり、圧縮モードで制御されます。この圧縮モードは、インスタンスに対してデフォルトとして設定することも、セッションレベルで設定することもできます。

インスタンスレベルでのデフォルトの圧縮モードの設定については、『InfiniDB管理者ガイド』を参照してください。

セッションレベルで圧縮モードを設定するには、次のコマンドを使用します。セッションが終了すると、後続のすべてのセッションはインスタンスのデフォルトに戻ります。

```
set infinidb_compression_type = n
```

*n*は次のとおりです。

- 0:圧縮は無効です。後続の表作成の文の実行では、文の上書きが実行されていないかぎり、その表に対して圧縮は無効になります。列を追加するためにALTER文を実行すると、文の上書きが実行されていないかぎり、その列に対して圧縮は無効になります。
- 1または2:圧縮は有効です。後続の表作成の文の実行では、文の上書きが実行されていないかぎり、その表に対して圧縮は有効になります。列を追加するためにALTER文を実行すると、文の上書きが実行されていないかぎり、その列に対して圧縮は有効になります。次の注意事項を参照してください。

**圧縮値に関する注意事項**:InfiniDBバージョン3.0.6、3.5.1以上(および今後のすべてのバージョン)では、圧縮タイプ1が廃止されました。このアルゴリズムでデータを圧縮することはできません。この変更によって顧客が受ける影響を最小限に抑えるために、InfiniDBでは圧縮タイプ1のリクエストは、タイプ2のリクエストとして処理されます。既存データがタイプ1で圧縮されている場合でも、引き続き読取り可能です。ただし、このデータを変更すると圧縮タイプ2で再圧縮されます。これはすべて、*infinidb_compression_type*変数の設定、およびcalpontsys.syscolumnに格納されている値に関係なく行われます。*infinidb_compression_type*を0に設定して圧縮を無効にすることはできますが、新しいデータを圧縮タイプ1で圧縮することはできません。圧縮タイプ1の使用を試みると、InfiniDBは警告することなく、それを圧縮タイプ2のリクエストとして処理します。

つまり、*infinidb_compression_type*を0に設定すると圧縮を無効にできます。*infinidb_compression_type*を1または2に設定すると圧縮を有効にできますが、InfiniDBでは*infinidb_compression_type*の設定が1であるか、2であるかに関係なく、新しいデータの圧縮には必ずタイプ2のアルゴリズムが使用されます。

これは、圧縮の有効または無効のみを選択できた前バージョンのInfiniDBとは基本的に違いはありません。現在でも圧縮の有効または無効のみを選択可能である点は同じですが、InfiniDBで新しいアルゴリズム(タイプ2)が使用される点が異なります。

# 10 ローカル PM 問合せ

InfiniDBには、UMのデータベース全体ではなく、単一のPMのみでデータの問合せを実行する機能があります。これには、`my.cnf`構成ファイルの*`infinidb_local_query`*変数を使用して、インスタンスのデフォルトとして設定するか、またはセッションレベルで設定します。この変数は、個々のPMでの問合せの実行にのみ適用され、UMで実行された場合はエラーが発生します。PMは、インストール時にローカル問合せオプションで設定する必要があります。

インスタンスレベルでのデフォルトのローカルPM問合せの設定については、『InfiniDB管理者ガイド』を参照してください。

## 10.1 ローカル PM 問合せの有効化または無効化

セッションレベルでローカルPM問合せの使用を有効または無効にするには、次のコマンドを使用します。セッションが終了すると、後続のすべてのセッションはインスタンスのデフォルトに戻ります。

```
set infinidb_local_query = n
```

*n*は次のとおりです。

- 0（無効）
- 1（有効）

# 11 パーティション管理

InfiniDBには、パーティションの無効化および削除を管理することによって、より効率的にデータ削除を管理する機能があります。 InfiniDBのパーティションの詳細は、『InfiniDB概要』のInfiniDBのストレージ概念に関する説明を参照してください。

データが破損する恐れがあるため、これらのコマンドは注意して使用してください。

パーティションの削除または無効化を管理するには、次の2つの方法があります。

- 列の値
- パーティション番号

## 11.1 列の値によるパーティション管理

以降に示す関数では、列の特定の値を使用してパーティションを管理できます。

### 11.1.1 列の値によるパーティション情報の表示

特定の列に対するパーティション情報を表示するには、calShowPartitionsbyValue関数を使用します。この関数によって出力される情報に、最小値と最大値が入力した値の範囲に完全に含まれているパーティションと、そのパーティションの状態が示されます。この情報を、必要に応じて、無効化、再有効化または削除するパーティションを判断するために使用します。

- あるエクステントの列の最小値/最大値がstartValとendValの範囲に完全に含まれている場合、そのエクステントに対して該当する処理が行われます。

- このような「値」によるパーティション関数では、パーティション列の比較的単純な型（最大8バイトのINTEGER、DECIMAL、DATE、DATETIME、CHARと、最大7バイトのVARCHAR）のみがサポートされます。


calShowPartitionsbyValueを次のように実行すると、orderdate列のデータが入力した範囲に含まれている3つのパーティションが表示されます。

```
mysql> select calShowPartitionsByValue('orders','o_orderdate', '1992-01-01', '20
10-10-31');
+-------------------------------------------------------------------------------
-------------------------------------------------------------------------------+

| calShowPartitionsByValue('orders','o_orderdate', '1992-01-01', '2010-10-31')
                                                                               |

mysql> select calShowPartitionsByValue('orders','orderdate', '1992-01-01',
'2010-07-24');
+-------------------------------------------------------------+
|calShowPartitionsbyvalue('orders','orderdate', '1992-01-02', '2010-07-24')|
+-------------------------------------------------------------+
| Part#      Min             Max             Status
  0.0.1      1992-01-01      1998-08-02      Enabled
  0.1.2      1998-08-03      2004-05-15      Enabled
  0.2.3      2004-05-16      2010-07-24      Enabled          |
+-------------------------------------------------------------+
1 row in set (0.05 sec)
```

### 11.1.2 値によるパーティションの無効化

calShowPartitionsbyValueのパーティション情報の分析後、パーティションを削除せずに無効にする場合は、calDisablePartitionsbyValue関数を使用します。無効にしたパーティションは、ファイルシステム上に残りますが、問合せ、DMLまたはインポートアクティビティには含まれません。

- あるエクステントの列の最小値/最大値がstartValとendValの範囲に完全に含まれている場合、そのエクステントに対して該当する処理が行われます。

- このような「値」によるパーティション関数では、パーティション列の比較的単純な型（最大8バイトのINTEGER、DECIMAL、DATE、DATETIME、CHARと、最大7バイトのVARCHAR）のみがサポートされます。


calDisablePartitionsbyValueを次のように実行すると、最初のパーティションが無効になります。

```
mysql> select calDisablePartitionsByValue('orders','orderdate', '1992-01-01',
'2004-05-15');
+-------------------------------------------------------------------------------+
| caldisablepartitionsbyvalue ('orders', 'o_orderdate','1992-01-01','1998-08-02') |
+-------------------------------------------------------------------------------+
| Partitions are disabled successfully                                          |
+-------------------------------------------------------------------------------+
1 row in set (0.28 sec)
```

最初のパーティションが無効になります。

```
mysql> select calShowPartitionsByValue('orders','orderdate', '1992-01-01',
'2010-07-24');
+-------------------------------------------------------------+
|calShowPartitionsbyvalue('orders','orderdate', '1992-01-02','2010-07-24' )|
+-------------------------------------------------------------+
| Part#      Min             Max               Status
  0.0.1      1992-01-01      1998-08-02        Disabled
  0.1.2      1998-08-03      2004-05-15        Enabled
  0.2.3      2004-05-16      2010-07-24        Enabled          |
+-------------------------------------------------------------+
1 row in set (0.05 sec)
```

### 11.1.3 値によるパーティションの有効化

calShowPartitionsbyValueのパーティション情報の分析後、パーティションを有効にする場合は、calEnablePartitionsbyValue関数を使用します。

- あるエクステントの列の最小値/最大値がstartValとendValの範囲に完全に含まれている場合、そのエクステントに対して該当する処理が行われます。

- このような「値」によるパーティション関数では、パーティション列の比較的単純な型（最大8バイトのINTEGER、DECIMAL、DATE、DATETIME、CHARと、最大7バイトのVARCHAR）のみがサポートされます。


calEnablePartitionsbyValueを次のように実行すると、最初のパーティションが有効になります。

```
mysql> select calEnablePartitionsByValue('orders','orderdate', '1992-01-01',
'2004-05-15');
+-----------------------------------------------------------------------------+
| calenablepartitionsbyvalue ('orders', 'o_orderdate','1992-01-01','1998-08-02')|
+-----------------------------------------------------------------------------+
| Partitions are enabled successfully                                         |
+-----------------------------------------------------------------------------+
1 row in set (0.28 sec)
```

最初のパーティションが有効になります。

```
mysql> select calShowPartitionsByValue('orders','orderdate', '1992-01-01',
'2010-07-24');
+-------------------------------------------------------------------+
|calShowPartitionsbyvalue('orders','orderdate', '1992-01-02','2010-07-24' )|
+-------------------------------------------------------------------+
| Part#      Min             Max                Status
  0.0.1      1992-01-01      1998-08-02         Enabled
  0.1.2      1998-08-03      2004-05-15         Enabled
  0.2.3      2004-05-16      2010-07-24         Enabled          |
+-------------------------------------------------------------------+
2 rows in set (0.05 sec)
```

### 11.1.4 値によるパーティションの削除

calShowPartitionsbyValueのパーティション情報の分析後、パーティションを削除する場合は、calDropPartitionsbyValue関数を使用します。有効または無効のどちらの状態の場合でも、パーティションは削除できます。

- あるエクステントの列の最小値/最大値がstartValとendValの範囲に完全に含まれている場合、そのエクステントに対して該当する処理が行われます。

- このような「値」によるパーティション関数では、パーティション列の比較的単純な型（最大8バイトのINTEGER、DECIMAL、DATE、DATETIME、CHARと、最大7バイトのVARCHAR）のみがサポートされます。


calDropPartitionsbyValueを次のように実行すると、最初のパーティションが削除されます。

```
mysql> select calDropPartitionsByValue('orders','orderdate', '1992-01-01',
'2004-05-15');
+-------------------------------------------------------------------------+
| caldroppartitionsbyvalue ('orders', 'o_orderdate','1992-01-01','1998-08-02') |
+-------------------------------------------------------------------------+
| Partitions are enabled successfully                                     |
+-------------------------------------------------------------------------+
1 row in set (0.28 sec)
```

最初のパーティションが削除されます。

```
mysql> select calShowPartitionsByValue('orders','orderdate', '1992-01-01',
'2010-07-24');
+-----------------------------------------------------------------+
|calShowPartitionsbyvalue('orders','orderdate', '1992-01-02','2010-07-24' )|
+-----------------------------------------------------------------+
| Part#      Min             Max                Status
  0.1.2      1998-08-03      2004-05-15         Enabled
  0.2.3      2004-05-16      2010-07-24         Enabled          |
+-----------------------------------------------------------------+
1 row in set (0.05 sec)
```

## 11.2 パーティション番号によるパーティション管理

以降に示す関数では、列のパーティション番号を使用してパーティションを管理できます。

### 11.2.1 パーティション番号によるパーティション情報の表示

特定の列に対するパーティション情報を表示するには、calShowPartitions関数を使用します。この関数によって出力される情報に、その列の最小値および最大値と、そのパーティションの状態が示されます。最小値および最大値は、必要に応じて、無効化、再有効化または削除するパーティションの決定に使用されます。


calShowPartitionsを次のように実行すると、orderdate列に対するデータが3つのパーティションで表示されます。

```
mysql> select calShowPartitions('orders','orderdate');
+-------------------------------------------------------------+
| calShowPartitions('orders','orderdate')                     |
+-------------------------------------------------------------+
| Part#      Min             Max                 Status
  0.0.1      1992-01-01      1998-08-02          Enabled
  0.1.2      1998-08-03      2004-05-15          Enabled
  0.2.3      2004-05-16      2010-07-24          Enabled     |
+-------------------------------------------------------------+
1 row in set (0.05 sec)
```

### 11.2.2 パーティション番号によるパーティションの無効化

calShowPartitionsのパーティション情報の分析後、パーティションを削除せずに無効にする場合は、calDisablePartitions関数を使用します。無効にしたパーティションは、ファイルシステム上に残りますが、問合せ、DMLまたはインポートアクティビティには含まれません。


- パーティション番号は、単一のパーティション('0.0.1')またはパーティションリスト('0.0.1,0.1.2')として指定できます。

calDisablePartitionsを次のように実行すると、パーティション0.0.1と0.1.2が無効になります。

```
mysql> select calDisablePartitions ('mydb','orders', '0.0.1, 0.1.2');
+-----------------------------------------------------+
| calDisablePartitions('mydb','orders', '0.0.1, 0.1.2')|
+-----------------------------------------------------+
| Partitions are disabled.                            |
+-----------------------------------------------------+
1 row in set (0.28 sec)
```

パーティション0.0.1と0.1.2が無効になります。

```
mysql> select calShowPartitions ('orders','orderdate');
+-------------------------------------------------------------------+
| calShowPartitions('orders','orderdate')                           |
+-------------------------------------------------------------------+
| Part#      Min             Max                 Status             |
  0.0.1      1992-01-01      1998-08-02          Disabled
  0.1.2      1998-08-03      2004-05-15          Disabled
  0.2.3      2004-05-16      2010-07-24          Enabled     |
+-------------------------------------------------------------------+
3 rows in set (0.05 sec)
```

### 11.2.3 パーティション番号によるパーティションの有効化

calShowPartitionsのパーティション情報の分析後、パーティションを有効にする場合は、calEnablePartitions関数を使用します。


- パーティション番号は、単一のパーティション('0.0.1')またはパーティションリスト('0.0.1,0.1.2')として指定できます。

calEnablePartitionsを次のように実行すると、パーティション0.0.1と0.1.2が有効になります。

```
mysql> select calEnablePartitions ('mydb','orders', '0.0.1,0.1.2');
+-------------------------------------------------+
| calEnablePartitions('mydb','orders','0.0.1,0.1.2'|
+-------------------------------------------------+
| Partitions are enabled.                         |
+-------------------------------------------------+
1 row in set (0.28 sec)
```

パーティション0.0.1と0.1.2が有効になります。

```
mysql> select calShowPartitions('orders','orderdate');
+-------------------------------------------------------------+
| calShowPartitions('orders','orderdate')                     |
+-------------------------------------------------------------+
| Part#      Min             Max                Status
  0.0.1      1992-01-01      1998-08-02         Enabled
  0.1.2      1998-08-03      2004-05-15         Enabled
  0.2.3      2004-05-16      2010-07-24         Enabled     |
+-------------------------------------------------------------+
3 rows in set (0.05 sec)
```

### 11.2.4 パーティション番号によるパーティションの削除

calShowPartitionsのパーティション情報の分析後、パーティションを削除する場合は、calDropPartitions関数を使用します。有効または無効のどちらの状態の場合でも、パーティションは削除できます。


- パーティション番号は、単一のパーティション('0.0.1')またはパーティションリスト('0.0.1,0.1.2')として指定できます。

calDropPartitionsを次のように実行すると、パーティション0.0.1と0.1.2が削除されます。

```
mysql> select calDropPartitions ('orders', '0.0.1,0.1.2');
+--------------------------------------------+
| calDropPartitions('orders', '0.0.1,0.1.2')  |
+--------------------------------------------+
| Partitions are dropped.                    |
+--------------------------------------------+
1 row in set (0.28 sec)
```

パーティション0.0.1と0.1.2がデータベースから削除されます。

```
mysql> select calShowPartitions('orders','orderdate');
+-------------------------------------------------------------+
| calShowPartitions('orders','orderdate')                     |
+-------------------------------------------------------------+
| Part#      Min             Max                Status
  0.2.3      2004-05-16      2010-07-24         Enabled     |
+-------------------------------------------------------------+
3 rows in set (0.05 sec)
```

# 12 InfiniDB での autoincrement の使用

InfiniDBのautoincrementの列の属性は、MySQLの類似した列の属性と名前を共有していますが、その動作はMySQLの実装と同じではありません。autoincrementの列がInfiniDBでどのように動作するのかを注意して読み、理解する必要があります。そうしない場合、この属性の使用によって予期しない結果を得る確率が高くなります。

1つの表内で1つの列をautoincrementとして定義することができます。整数型の列を指定する必要があります。InfiniDBは、次に示す各行のこの列に一意の値を割り当てます。

- INSERT文、LOAD DATA INFILE文またはcpimportを使用して挿入されていて明示的または暗黙的にNULLとしてコード化されている値を持っているか、または明示的に0（ゼロ）としてコード化されている。
- NULLまたは0（ゼロ）の明示的な値に更新されている。別の列の値または式の結果を使用して列を更新する場合、最終的な評価がNULLまたは0（ゼロ）かどうかにかかわらず、列はその値に更新されることに注意してください。

autoincrementの列を手動で更新しない場合、InfiniDBは常に一意の値を使用します。これらの値は一意というだけで、継続的または均一に増加していくとはかぎりません。ただし、autoincrementの列は一意の値に制約されません。行を挿入または更新して明示的に値（NULLまたは0（ゼロ）以外）をコード化する場合、InfiniDBは、（そのデータ型のその他すべての要件を満たしている場合）列で値が重複することになってもその値を挿入し、エラーまたは警告を発行しません。

表レベルのautoincrementの開始値を、表の最大値よりも大きい値に変更する場合、InfiniDBは次の挿入または更新でその新しい値の使用を開始します。これにより、数字間でギャップが生じます。表内の最小値よりも小さい値に変更する場合、InfiniDBは後続の挿入および更新でこの新しい値の使用を開始します。これにより重複が作成されます。どちらの場合にもエラーまたは警告は発行されません。

InfiniDBシステムで、データ型が適合しない（autoincrementの列がTINYINTデータ型なのに127を超える行を挿入しようとするなど）ために順序番号を生成できない場合、文またはジョブは失敗します。この場合、autoincrementの設定をリセットするか、autoincrementの列を削除して、範囲の広いデータ型の新しいautoincrementの列を追加する必要があります。

LOAD DATA INFILEは、ソースファイルのautoincrementの列にNULL値があるとMySQLの警告を発生させます。この警告を回避するには、autoincrementの列にNULLのかわりに0（ゼロ）を使用します。

# 13 UTF-8 キャラクタセットの使用

## 13.1 UTF-8 キャラクタセット

InfiniDBには、UTF-8キャラクタセットをサポートする機能があります。このプロファイルは、インスタンスに対してデフォルトとして設定することも、セッションレベルで設定することもできます。

インスタンスレベルでのUTF-8プロファイルの設定については、『InfiniDB管理者ガイド』を参照してください。

セッションレベルでUTF-8プロファイルを設定するには、次のコマンドを使用します。セッションが終了すると、後続のすべてのセッションはインスタンスのデフォルトに戻ります。

```
set names 'utf8' collate value;
```

*value*は、有効なUnicodeキャラクタセットです。有効な値については、次のMySQLリファレンスを参照してください。

http://dev.mysql.com/doc/refman/5.1/en/charset-unicode-sets.html

## 13.2 UTF-8 のオブジェクト名

UTFで文字列の列を作成する場合は、文字数ではなくバイト数で長さを定義します。UTFで10個のマルチバイト文字を格納するには、格納される実際の値に応じて20-30バイトが必要です。

## 13.3 既知の問題および制限

- インスタンスをUTF-8プロファイルで設定した場合は、表レベルでUTF-8を宣言する必要があります。一致しないキャラクタセットで表を作成すると、予測できない結果が発生します。
- SQLの出力の表示は、UTF-8キャラクタセットをサポートするクライアントソフトウェアを使用して実行する必要があります。
- オブジェクト名ではUTF-8文字はサポートされていません。