Atitit import sql fun 重要的sql功能扩展 ext

目录

[1.1. Insert merge 1](#_Toc9740)

[1.2. Insert set 1](#_Toc20107)

[1.2.1. 13.2.5. LOAD DATA INFILE语法 1](#_Toc23210)

[1.2.2. 13.2.6. REPLACE语法 2](#_Toc10857)

[2. 13.2.10. UPDATE语法 3](#_Toc4107)

[2.1. ORDER BY子句，则按照被指定的顺序对行进行更新。 4](#_Toc3317)

[2.2. LIMIT子句用于给定一个限值，限制可以被更新的行的数目。 4](#_Toc18297)

[3. 4](#_Toc5141)

[3.1.1. 13.5.3. SET语法 4](#_Toc18941)

[3.2. 13.7. 用于预处理语句的SQL语法 5](#_Toc25724)

## Insert merge

## Insert set

### **13.2.5. LOAD DATA INFILE语法**

LOAD DATA [LOW\_PRIORITY | CONCURRENT] [LOCAL] INFILE '*file\_name*.txt'

    [REPLACE | IGNORE]

    INTO TABLE *tbl\_name*

    [FIELDS

        [TERMINATED BY '*string*']

        [[OPTIONALLY] ENCLOSED BY '*char*']

        [ESCAPED BY '*char*' ]

    ]

    [LINES

        [STARTING BY '*string*']

        [TERMINATED BY '*string*']

    ]

    [IGNORE *number* LINES]

    [(*col\_name\_or\_user\_var*,...)]

    [SET *col\_name* = *expr*,...)]

LOAD DATA INFILE语句用于高速地从一个文本文件中读取行，并装入一个表中。文件名称必须为一个文字字符串。

如果您指定IGNORE，则把原有行复制到唯一关键字值的输入行被跳过。如果您这两个选项都不指定，则运行情况根据LOCAL关键词是否被指定而定。不使用LOCAL时，当出现重复关键字值时，会发生错误，并且剩下的文本文件被忽略。使用LOCAL时，默认的运行情况和IGNORE被指定时的情况相同；这是因为在运行中间，服务器没有办法中止文件的传输。

如果您希望在载入运行过程中忽略外键的限制，您可以在执行LOAD DATA前发送一个SET FOREIGN\_KEY\_CHECKS=0语句。

### **13.2.6. REPLACE语法**

REPLACE [LOW\_PRIORITY | DELAYED]

    [INTO] *tbl\_name* [(*col\_name*,...)]

    VALUES ({*expr* | DEFAULT},...),(...),...

或：

REPLACE [LOW\_PRIORITY | DELAYED]

    [INTO] *tbl\_name*

    SET *col\_name*={*expr* | DEFAULT}, ...

或：

REPLACE [LOW\_PRIORITY | DELAYED]

    [INTO] *tbl\_name* [(*col\_name*,...)]

    SELECT ...

REPLACE的运行与INSERT很相像。只有一点除外，如果表中的一个旧记录与一个用于PRIMARY KEY或一个UNIQUE索引的新记录具有相同的值，则在新记录被插入之前，旧记录被删除。请参见[13.2.4节，“INSERT语法”](http://download.nust.na/pub6/mysql/doc/refman/5.1/zh/sql-syntax.html" \l "insert" \o "13.2.4. INSERT Syntax)。

注意，除非表有一个PRIMARY KEY或UNIQUE索引，否则，使用一个REPLACE语句没有意义。该语句会与INSERT相同，因为没有索引被用于确定是否新行复制了其它的行。

# 13.2.10. UPDATE语法

Single-table语法：

UPDATE [LOW\_PRIORITY] [IGNORE] *tbl\_name*

    SET *col\_name1*=*expr1* [, *col\_name2*=*expr2* ...]

    [WHERE *where\_definition*]

    [ORDER BY ...]

    [LIMIT *row\_count*]

Multiple-table语法：

UPDATE [LOW\_PRIORITY] [IGNORE] *table\_references*

    SET *col\_name1*=*expr1* [, *col\_name2*=*expr2* ...]

    [WHERE *where\_definition*]

UPDATE语法可以用新值更新原有表行中的各列。SET子句指示要修改哪些列和要给予哪些值。WHERE子句指定应更新哪些行。如果没有WHERE子句，则更新所有的行。如果指定了

## ORDER BY子句，则按照被指定的顺序对行进行更新。

## LIMIT子句用于给定一个限值，限制可以被更新的行的数目。

# 

### **13.5.3. SET语法**

SET *variable\_assignment* [, *variable\_assignment*] ...

*variable\_assignment*:

*user\_var\_name* = *expr*

    | [GLOBAL | SESSION] *system\_var\_name* = *expr*

    | @@[global. | session.]*system\_var\_name* = *expr*

SET用于设置不同类型的变量。这些变量会影响服务器或客户端的操作。SET可以用于向用户变量或系统变量赋值。

用于分配账户密码的SET PASSWORD语句在[13.5.1.5节，“SET PASSWORD语法”](http://download.nust.na/pub6/mysql/doc/refman/5.1/zh/sql-syntax.html" \l "set-password" \o "13.5.1.5. SET PASSWORD Syntax)中进行了讨论。

多数系统变量可以在运行时间被更改。可以被动态设置的系统变量在[5.3.3.1节，“动态系统变量”](http://download.nust.na/pub6/mysql/doc/refman/5.1/zh/database-administration.html" \l "dynamic-system-variables" \o "5.3.3.1. Dynamic System Variables)中进行了讨论。

注释：旧版本的MySQL采用SET OPTION作为这个命令，但是由于有了SET，现在不赞成使用SET OPTION。

以下例子显示了您可以用于设置变量的不同语法。

用户变量可以被写作@*var\_name*，并可以进行如下设置：

SET @*var\_name* = *expr*;

在[9.3节，“用户变量”](http://download.nust.na/pub6/mysql/doc/refman/5.1/zh/language-structure.html" \l "variables" \o "9.3. User Variables)中给出了有关用户变量的更多信息。

系统变量可以被作为*var\_name*引用到SET语句中。在名称的前面可以自选地添加GLOBAL或@@global，以明确地指示该变量是全局变量。或者在名称前面添加SESSION, @@session，或@@，以指示它是一个会话变量。LOCAL和@@local是SESSION和@@session地同义词。如果没有修改符，则SET设置会话变量。

支持系统变量的@@*var\_name*语法，以便使MySQL语法与其它数据库系统相兼容。

## **13.7. 用于预处理语句的SQL语法**

MySQL 5.1对服务器一方的预制语句提供支持。如果您使用合适的客户端编程界面，则这种支持可以发挥在MySQL 4.1中实施的高效客户端/服务器二进制协议的优势。候选界面包括MySQL C API客户端库（用于C程序）、MySQL Connector/J（用于Java程序）和MySQL Connector/NET。例如，C API可以提供一套能组成预制语句API的函数调用。请参见[25.2.4节，“C API预处理语句”](http://download.nust.na/pub6/mysql/doc/refman/5.1/zh/apis.html" \l "c-api-prepared-statements" \o "25.2.4. C API Prepared Statements)。其它语言界面可以对使用了二进制协议（通过在C客户端库中链接）的预制语句提供支持。有一个例子是[PHP 5.0中的mysqli扩展](http://php.net/mysqli" \t "http://download.nust.na/pub6/mysql/doc/refman/5.1/zh/_top)。

对预制语句，还有一个SQL界面可以利用。与在整个预制语句API中使用二进制协议相比，本界面效率没有那么高，但是它不要求编程，因为在SQL层级，可以直接利用本界面：

·         当您无法利用编程界面时，您可以使用本界面。

·         有些程序允许您发送SQL语句到将被执行的服务器中，比如mysql客户端程序。您可以从这些程序中使用本界面。

·         即使客户端正在使用旧版本的客户端库，您也可以使用本界面。唯一的要求是，您能够连接到一个支持预制语句SQL语法的服务器上。

预制语句的SQL语法在以下情况下使用：

·         在编代码前，您想要测试预制语句在您的应用程序中运行得如何。或者也许一个应用程序在执行预制语句时有问题，您想要确定问题是什么。

·         您想要创建一个测试案例，该案例描述了您使用预制语句时出现的问题，以便您编制程序错误报告。

·         您需要使用预制语句，但是您无法使用支持预制语句的编程API。

预制语句的SQL语法基于三个SQL语句：

PREPARE *stmt\_name* FROM *preparable\_stmt*;

EXECUTE *stmt\_name* [USING @*var\_name* [, @*var\_name*] ...];

{DEALLOCATE | DROP} PREPARE *stmt\_name*;

PREPARE语句用于预备一个语句，并赋予它名称stmt\_name，借此在以后引用该语句。语句名称对案例不敏感。preparable\_stmt可以是一个文字字符串，也可以是一个包含了语句文本的用户变量。该文本必须展现一个单一的SQL语句，而不是多个语句。使用本语句，‘?’字符可以被用于制作参数，以指示当您执行查询时，数据值在哪里与查询结合在一起。‘?’字符不应加引号，即使您想要把它们与字符串值结合在一起，也不要加引号。参数制作符只能被用于数据值应该出现的地方，不用于SQL关键词和标识符等。

如果带有此名称的预制语句已经存在，则在新的语言被预备以前，它会被隐含地解除分配。这意味着，如果新语句包含一个错误并且不能被预备，则会返回一个错误，并且不存在带有给定名称语句。

预制语句的范围是客户端会

MySQL :: MySQL 5.1参考手册 :: 13. SQL语句语法

MySQL :: MySQL 5.1参考手册 :: 13. SQL语句语法