***在语句末尾分号前加\G可以以网格形式呈现***

***查询表名为tableName的auto\_increment值：***

***SELECT AUTO\_INCREMENT FROM information\_schema.tables***

***WHERE table\_name="tableName";***

***修改表名为tableName的auto\_increment值：***

***ALTER TABLE tableName auto\_increment=number ;***

第1章 初涉MySQL

1-1 MySQL概述

1-2 MySQL的安装与配置

1-3 启动与停止MySQL服务

·启动服务（所有的window任务都可以通过net start和net work来启动与停止）

启动：net start mysql

停止：net stop mysql

1-4 MySQL登录与退出



·MySQL退出

mysql > exit;

mysql > quit;

mysql > \q;

1-5 修改MySQL提示符

·连接客户端时通过参数指定

shell>mysql -u root -p密码 --prompt 提示符//密码与p之间不能有空格

·连接上客户端后，通过prompt命令修改

mysql>prompt 提示符



1-6 MySQL常用命令以及语法规范

·显示当前服务器版本

**SELECT VERSION();**

·显示当前日期时间

**SELECT NOW();**

·显示当前用户

**SELECT USER();**

·MySQL语句的规范

·关键字与函数名称全部大写

·数据库名称、表名称、字段名称全部小写

·SQL语句必须以分号结尾

1-7 操作数据库

·花括号内的为必选项，中括号内的为可选项

·创建数据库

**CREATE {DATABASE | SCHEMA} [IF NOT EXISTS] db\_name**

**[DEFAULT] CHARACTER SET [=] charset\_name** //设置编码方式

例：CREATE DATABASE IF NOT EXISTS t2 CHARACTER SET gbk;

·查看当前服务器下的数据表列表

**SHOW {DATABASES | SCHEMAS} [LIKE 'pattern' | WHERE expr]**

·**SHOW WARNINGS;** //查看警告信息

·**SHOW CREATE DATABASE t1;** //显示创建数据库时所使用的指令，若修改过字符集则显示修改后的

·修改数据库

**ALTER {DATABASE SCHEMA} [db\_name]**

**[DEFAULT] CHARACTER SET [=] charset\_name**

·删除数据库

**DROP {DATABASE | SCHEMA} [IF EXISTS] db\_name**

第2章 数据类型与操作数据表

2-1 内容回顾

·MySQL默认的端口号是多少？ 3306

·MySQL中的超级用户叫什么？ root

·创建数据库 CREATE DATABASE

·修改数据库 ALTER DATABASE

·删除数据库 DROP DATABASE

2-2 MySQL数据类型之整型

·数据类型是指列、存储过程参数、表达式和局部变量的数据特征，他决定了数据的存储格式，代表了不同的信息类型。



2-3 MySQL数据类型之浮点型



2-4 MySQL数据类型之日期时间型



DATE 1000年1月1号~9999年12月31日

DATETIME 1000年1月1号00：00：00~9999年12月31日23：59：59

TIMESTAMP 1970年1月1日0点~2037年

TIME -8385959~8385959

YEAR

2-5 MySQL数据类型之字符型



2-6 MySQL创建数据表

·数据表（或称表）是数据库最重要的组成部分致意，是其他对象的基础。

·USE

打开数据库

USE 数据库名称

·SELECT DATABASE();显示当前用户所打开的数据库

·创建数据表

**CREATE TABLE [IF NOT EXISTS] table\_name(**

**column\_name data\_type**

**...**

**)**

2-7 MySQL查看数据表

·查看数据表列表

**SHOW TABLES [FROM db\_name]**

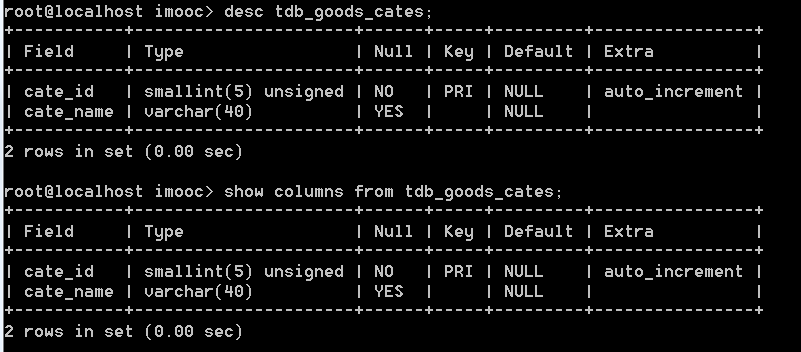
**[LIKE 'pattern' | WHERE expr]**

2-8 MySQL查看数据表结构

·查看数据表结构

**SHOW COLUMNS FROM tbl\_name**

**DESC tbl\_name**



2-9 MySQL记录的插入与查找

·记录插入

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

·记录查找

**SELECT expr,... FROM tbl\_name**

2-10 MySQL空值与非空

·NULL / NOT NULL

2-11 MySQL自动编号

·AUTO\_INCREMENT //要想使用必须先设为主键，必须为数值型，若是浮点数小数位数为0

自动编号，且必须与主键组合使用

使主键在默认情况下，起始值为1，每次的增量都为1

2-12 MySQL初涉主键约束

·PRIMARY KEY

主键约束

每张表只能存在一个主键

主键保证记录的唯一性

主键自动为NOT NULL

2-13 MySQL初涉唯一约束

·UNIQUE EKY

唯一约束

唯一约束可以保证记录的唯一性

唯一约束的字段可以为空值（NULL）

每张数据表可以存在多个唯一约束

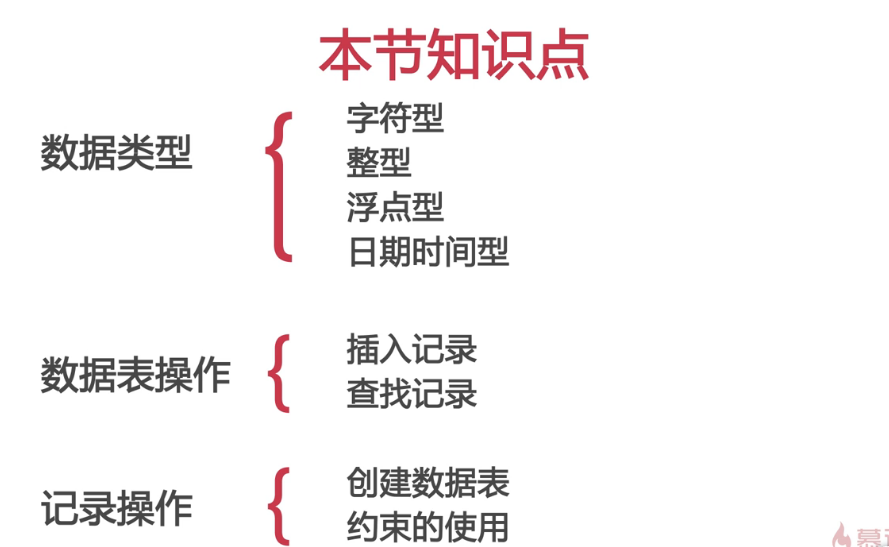
2-14 MySQL初涉默认约束

·DEFAULT

默认值

当插入记录时，如果没有明确为字段复制，则自动赋予默认值

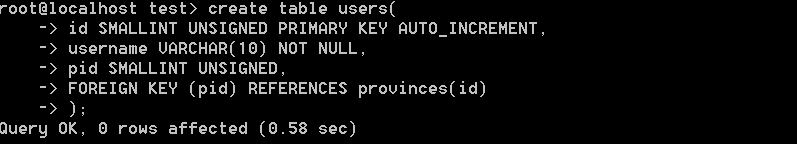
2-15 总结



第3章 约束以及修改数据表

3-1 回顾和概述

3-2 MySQL 外键约束的要求解析



·约束保证数据的完整性和一致性

·约束分为表级约束和列级约束

·约束类型包括：

NOT NULL(非空约束)

PRIMARY KEY(主键约束)

UNIQUE KEY(唯一约束)

DEFAULT(默认约束)

FOREIGN KEY(外键约束

)

·FOREIGN KEY

保持数据一致性、完整性

实现一对一或一对多关系

·外键约束的要求

·父表和子表必须使用相同的存储引擎，而且禁止使用临时表。

·**数据表的存储引擎只能为InnoDB**。

·外键列和参照列必须具有相似的数据类型。其中数字的长度或是否有符号位必须相同； 而字符的长度则可以不同。

·外键列和参照列必需创建索引。如果外键列不存在索引的话，MySQL将自动创建索引。

·查看索引

**SHOW INDEXES FROM tbl\_name;**

·编辑数据表的默认存储引擎

·MySQL配置文件

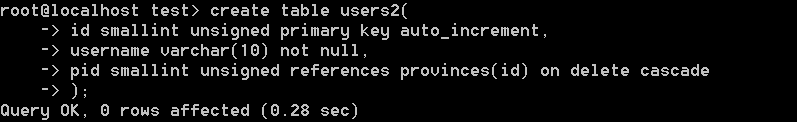
default-storage-engine=INNODB

·查看当前数据库的默认引擎

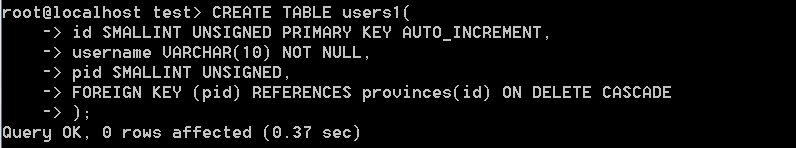
SHOW VARIABLES LIKE 'default\_storage\_engine';

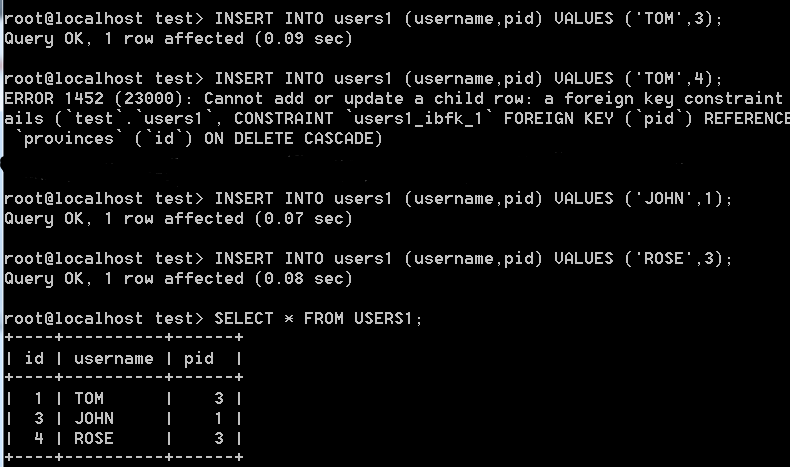
3-3 MySQL 外键约束的参照操作

·CASCADE:从父表删除或更新且自动删除或更新子表中匹配的行



或







·SET NULL:从父表删除或更新行，并设置子表中的外键列

·RESTRICT:拒绝对父表的删除或更新操作

·NO ACTION:标准SQL的关键字，在MySQL中与RESTRICT相同

3-4 MySQL 表级约束与列级约束

·对一个数据列建立的约束，称为列级约束。

·读多个数据列建立的约束，称为表级约束。

·列级约束既可以在列定义时声明，也可以在列定义后声明。

·表级约束只能在列定义后声明

·主键约束、外键约束和唯一约束可以存在表级约束和列级约束

·NOT NULL约束和DEFAULT约束不存在表级约束

3-5 MySQL 修改数据表-添加/删除列

·添加单列

**ALTER TABLE tbl\_name ADD [COLUMN] col\_name**

**column\_definition [FIRST |AFTER col\_name]** //默认加在最后

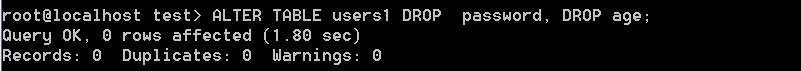
·添加多列

**ALTER TABLE tbl\_name ADD [COLUMN]**

**(col\_name column\_definition,...)** //不能指定插入位置

·删除列

**ALTER TABLE tbl\_name DROP [COLUMN] col\_name**



3-6 MySQL 修改数据表--添加约束

·添加主键约束

**ALTER TABLE tbl\_name ADD [CONSTRAINT [symbol]]**

**PRIMARY KEY [index\_type] (index\_col\_name,...)**

·添加唯一约束

**ALTER TABLE tbl\_name ADD [CONSTRAINT [symbol]]**

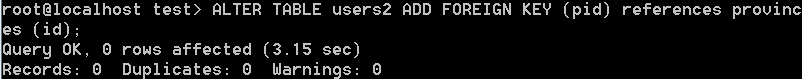
**UNIQUE [INDEX|KEY] [index\_name] [index\_type] (index\_col\_name,...)**

·添加外键约束

**ALTER TABLE tbl\_name ADD [CONSTRAINT [symbol]]**

**FOREIGN KEY [index\_name] (index\_col\_name,...)**

**reference\_definition**



·添加/删除默认约束

**ALTER TABLE tbl\_name ALTER [COLUMN] col\_name**

**{SET DEFAULT literal | DROP DEFAULT}**

3-7 MySQL 修改数据表--删除约束

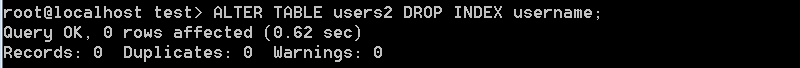
·删除主键约束

**ALTER TABLE tbl\_name DROP PRIMARY KEY**

//任何一个数据表有且仅有一个主键，不需要指定名称

·删除唯一约束

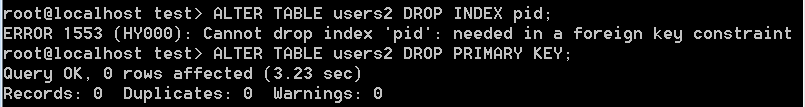
**ALER TABLE tbl\_name DROP {INDEX|KEY} index\_name**//通过索引删除

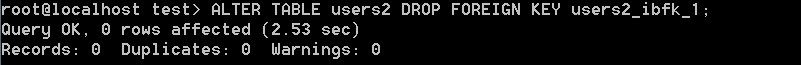


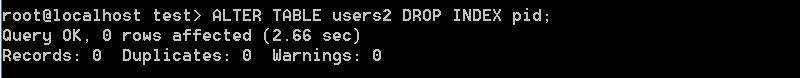
·删除外键约束

**ALTER TABLE tbl\_name DROP FOREIGN KEY fk\_symbol**

不可直接删除索引







3-8 MySQL 修改数据表--修改列定义和更名数据表

·修改列定义

**ALTER TABLE tbl\_name MODIFY [COLUMN] col\_name**

**column\_definition [FIRST | AFTER col\_name]**

//修改列的数据类型或数据类型和位置

//可能会造成数据丢失

·修改列名称//与oracle不同

**ALTER TABLE tbl\_name CHANGE [COLUMN] old\_col\_name**

**new\_col\_name column\_definition [FIRST|AFTER col\_name]**

//比modify多了改名功能

·数据表更名

方法1

**ALTER TABLE tbl\_name RENAME [TO|AS] new\_tbl\_name**

方法2

**RENAME TABLE bl\_name TO new\_tbl\_name**

**[, tblname2 TO new\_tbl\_name2]...**

·尽量少使用数据表或数据列的更名。当我们创建了索引或使用过视图或使用过存储过程的话，在表名或列名曾经被引用的情况下，更名可能会导致其无法正常使用

3-9 小结

·本节知识点

·约束

·按功能划为：NOT NULL, PRIMARY KEY, UNIQUE KEY, DEFAULT, FOREIGN KEY, ·按数据列的数目划为：表级约束， 列级约束

·修改数据表

·针对字段的操作：添加/删除字段、修改列定义，修改列名称等

·针对约束的操作：添加、删除各种约束

·针对数据表的操作：数据表更名（两种方式）



记录INSERT

·**INSERT [INTO] tbl\_name[(col\_name)] {VALUES|VALUE}**

**({EXPR|DEFAULT},...),(...),...** //可插入多条记录

第4章 操作数据表中的记录

4-1 回顾和概述

4-2 MySQL 插入

· 如果把一个NULL插入到一个AUTO\_INCREMENT数据列里去，MySQL(和PHP搭配之最佳组合)将自动生成下一个序列编号。编号从1开始，并1为基数递增。把0插入MySQL AUTO\_INCREMENT数据列的效果与插入NULL值一样。但不建议这样做，当入记录时，没有为AUTO\_INCREMENT明确指定值，则等同插入NULL值。

·可在建表时可用“MySQL AUTO\_INCREMENT=n”选项来指定一个自增的初始值。可用alter table table\_name AUTO\_INCREMENT=n命令来重设自增的起始值。

·若存在默认值，也可以用DEFAULT代替NULL

4-3 MySQL 插入记录INSERT SET-SELECT

·插入记录

**·INSERT[INTO] tbl\_name [(col\_name,...)] {VALUES|VALUE}**

**({EXPR|default},...),(...),...**

·**INSERT [INTO] tbl\_name SET col\_name1={expr|DEFAULT},col\_name2=...**

//与第一种方式的区别在于，此方法可以使用子查询（SubQuery），一次仅能插入一条



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

//此方法可将查询结果插入到指定数据表



4-4 MySQL 单表更新记录UPDATE

·更新记录（单表更新）

**UPDATE [LOW\_PRIMARY] [IGNAORE] table\_reference SET**

**col\_name1={expr1|DEFAULT} [,col\_name2={expr2|DEFAULT}]...**

**[WHERE where\_conditon]**

4-5 MySQL 单表删除记录DELETE

·删除记录（单表删除）

**DELETE FROM tbl\_name [WHERE where\_condition]**

4-6 MySQL 查询表达式解析

**·SELECT select\_expr [,select\_expr...]**

**[**

**FROM table\_references**

**[WHERE where\_conditon]**

**[GROUP BY{col\_name|position} [ASC|DESC],...]**

**[HAVAING where\_conditon]**

**[ORDER BY {col\_name|expr|position} [ASC|DESC],...]**

**[LIMIT{[offset,] row\_count|row\_count OFFST offset}]**

**]**

·select\_expr

查询表达式

每一个表达式表示想要的一列，必须有至少一个。

多个列之间以英文逗号分隔。

星号（\*）表示所有列。tbl\_name.\*可以表示命名表的所有列。

查询表达式可以使用[AS] alias\_name 为其赋予别名。

别名可用于GTOUP BY, ORDER BY或HAVING子句

4-7 MySQL where语句进行条件查询

·条件表达式

对记录进行过滤，如果没有指定WHERE子句，则显示所有记录

在WHERE表达式中，可以使用MySQL支持的函数或运算符

4-8 MySQL group by语句对查询结果分组

·查询结果分组

**[GROUP BY {col\_name|position} [ASC|DESC],...]**

position=1,2,3...

4-9 having语句设置分组条件

·分组条件

**HAVING [where\_condition]**

4-10 order by语句对查询结果排序

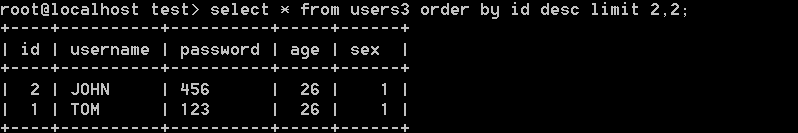
**·[ORDER BY {col\_name|expr|position} [ASC|DESC],...]**

4-11 limit语句限制查询数量

·限制查询结果返回的数量

**[LIMIT {[offset,] row\_count|row\_count OFFSET offset}]**

//从0开始编号



先排序再限制

4-12 小结

·记录操作

·INSERT

·UPDATE

·DELETE

·SELECT

第5章 子查询与连接

5-1 数据准备

·SET NAMES charset;

暂时在客户端以charset编码显示数据

5-2 MySQL 子查询简介

·子查询（Subquery）是指出现在其他SQL语句内的SELECT子句。

例如：SELECT \* FROM t1 WHERE col1 = (SELECT col2 FROM t2);

其中SELECT \* FROM t1,称为Outer Query/Outer Statement

SELECT col2 FROM t2,称为SubQuery

·子查询指镶嵌在查询内部，且必须始终出现在圆括号内。

·子查询可以包含多个关键字或条件，

如DISTINCT、GROUP BY、ORDER BY,LIMIT,函数等。

·子查询的外层查询可以是：DELECT,INSERT,UPDATE,SET或DO.

·子查询可以返回标量、一行、一列或子查询。

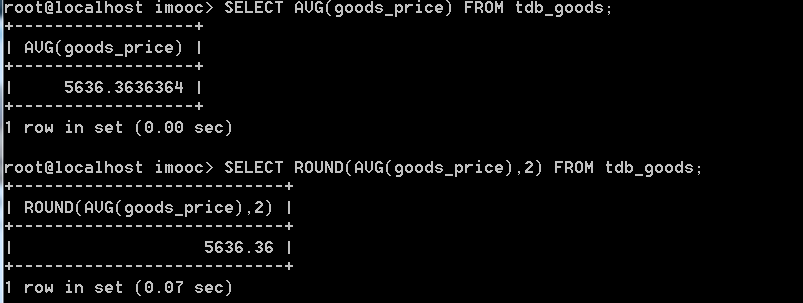
5-3 由比较运算符引发的子查询

·使用比较运算符的子查询

=、>、<、>=、<=、<>、!=、<=>

·语法结构

operand comparison\_operator subquery





·用 ANY、SOME、或ALL修饰的比较运算符

**operand comparison\_operator ANY|SOME|ALL (subquery)**

此处ANY与SOME等价



5-4 由[NOT] IN/EXISTS引发的子查询

·使用[NO] IN的子查询

operand comparison\_operator [NOT] IN (subquery)

=ANY 运算符与IN等效。

！=ALL或<>ALL运算符与NOT IN等效

·使用[NOT] EXISTS的子查询

如果子查询返回任何行，EXISTS将返回TRUE；否则为FALSE

5-5 使用INSERT...SELECT插入记录

·将查询结果写入数据表

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

**SELECT...**

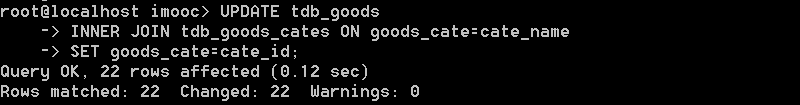
5-6 多表更新

·**UPDATE table\_references**

**SET col\_name1={expr1|DEFAULT}**

**[.col\_name2={expr2|DEFAULT}]...**

**[WHERE where\_condition]**



·MYSQL执行如下语句报错：

UPDATE sc SET grade =grade\*1.05 WHERE grade < (SELECT AVG(grade) AS avg\_grade FROM sc)

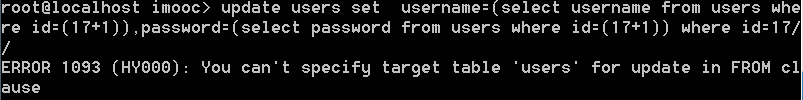
报错信息如下：

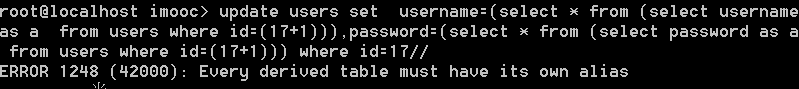
　　 错误代码： 1093  
　　 You can't specify target table 'sc' for update in FROM clause

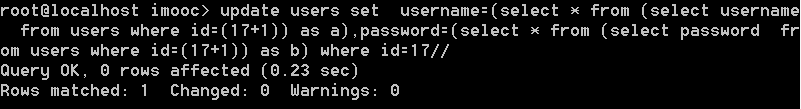
意思是不能在同一语句中更新select出的同一张表元组的属性值

解决方法：将select出的结果通过中间表再select一遍即可。

UPDATE sc SET grade =grade\*1.05 WHERE grade < (SELECT avg\_grade FROM (SELECT AVG(grade) AS avg\_grade FROM sc) AS temp)







5-7 多表更新之一步到位

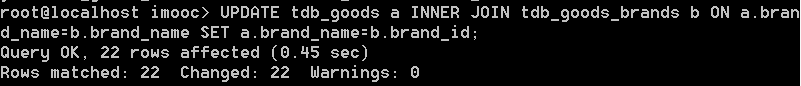
·创建数据表同时将查询结果写入到数据表

**CREATE TABLE [IF NOT EXISTS] tbl\_name**

**[(create\_definition,...)]**

**select\_statement**





5-8 连接的语法结构

·MYSQL在SELECT语句、多表更新、多表删除语句中支持JOIN操作

·table\_reference

**{[INNER|CROSS] JOIN | {LEFT|RIGHT} [OUTER] JOIN}**

**table\_reference**

**ON conditional\_expr**

·数据表参照

table\_reference

**tbl\_name [[AS] alias] | table\_subquery [AS] alias**

数据表可以使用tbl\_name alias\_name赋予别名。

table\_subquery可以作为子查询使用在FROM子句中，

这样的子查询必须为其赋予别名

5-9 内连接INNER JOIN

·INNER JOIN,内连接//交集

在MySQL中，JOIN,CROSS JOIN和INNER JOIN是等价的

·LEFT [OUTER] JOIN,左外连接

·RIGHT [OUTER] JOIN,右外连接

·连接条件

·使用ON关键字来设定连接条件，也可以使用WHERE来代替

·通常使用ON关键字来设定连接条件，使用WHERE关键字进行结果集记录的过滤

5-10 外连接OUTER JOIN

·左外连接

显示左表的全部记录及右表符合连接条件的记录

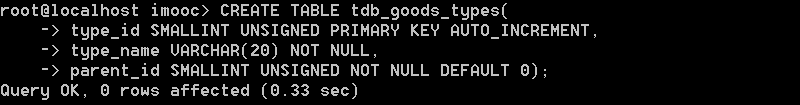
·右外连接

显示右表的全部记录及左表符合连接条件的记录

5-11 多表连接

5-12 关于连接的几点说明

5-13 无限级分类表设计



·通过自身连接来实现查找

·自身连接

同一个数据表对其自身进行连接

5-14 多表删除

**·DELETE tbl\_name[.\*] [,tbl\_name[.\*]]...**

**FROM table\_references**

**[WHERE where\_condition]**

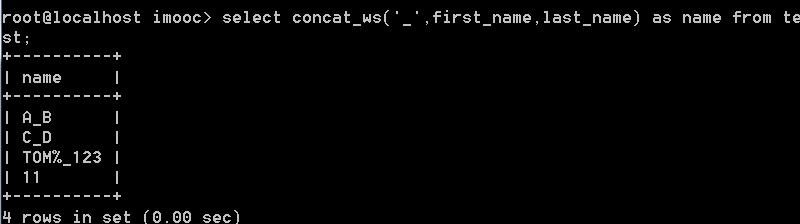
第6章 运算符和函数

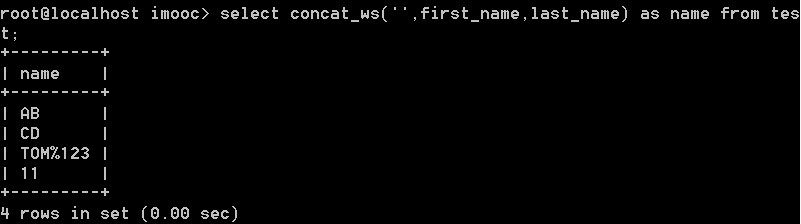
6-1 回顾和概述

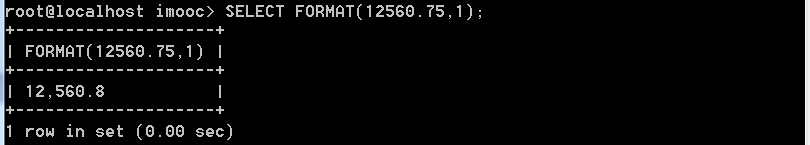
6-2 MySQL 字符函数





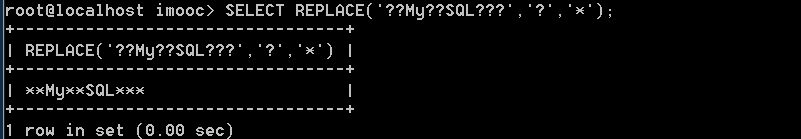
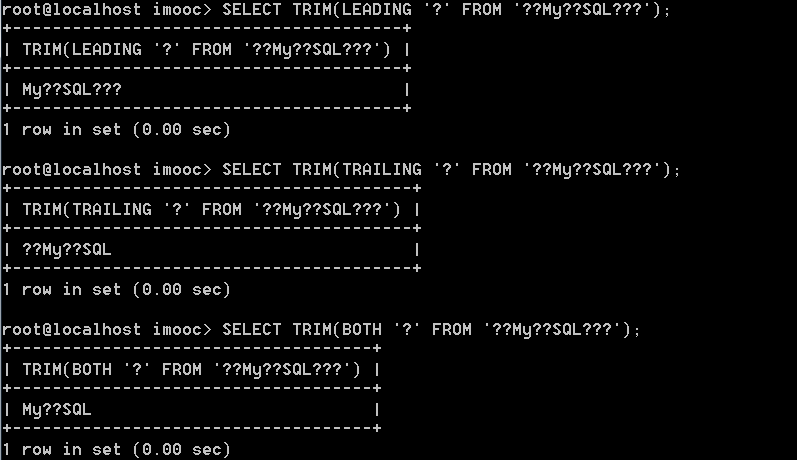


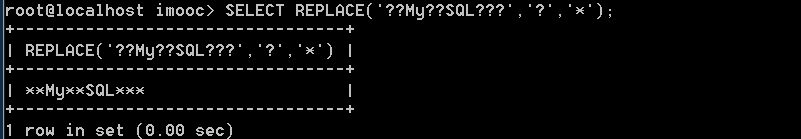








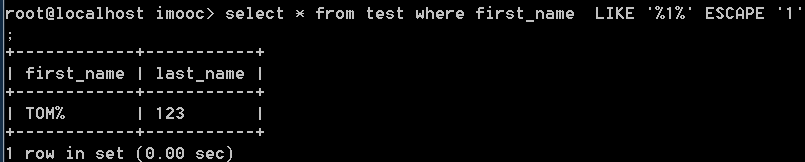








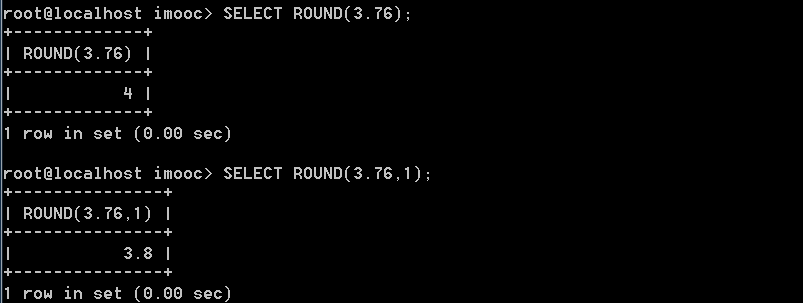
//MySQL编号从1开始

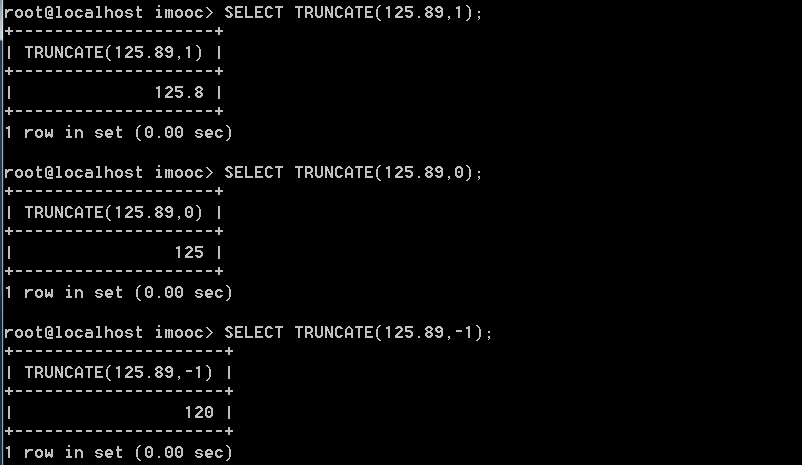


//%任意个字符，\_任意一个字符。1后面的百分号不需要解析。

6-3 MySQL 数值运算符和函数





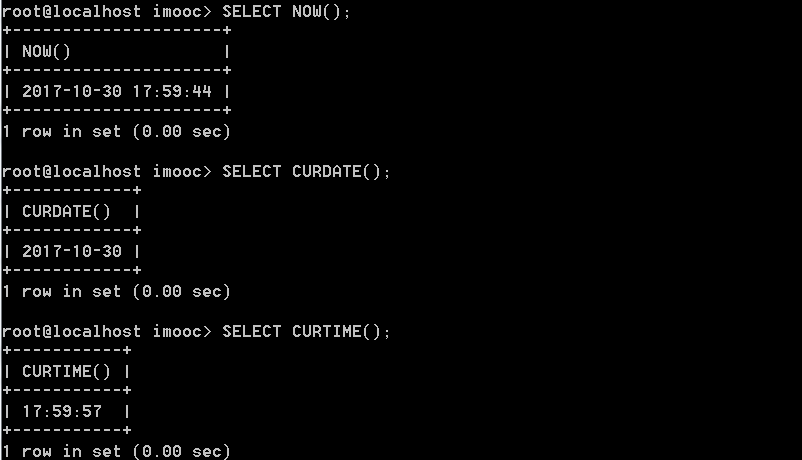


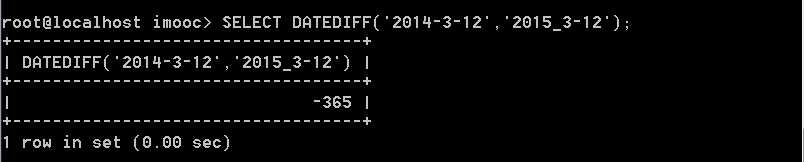
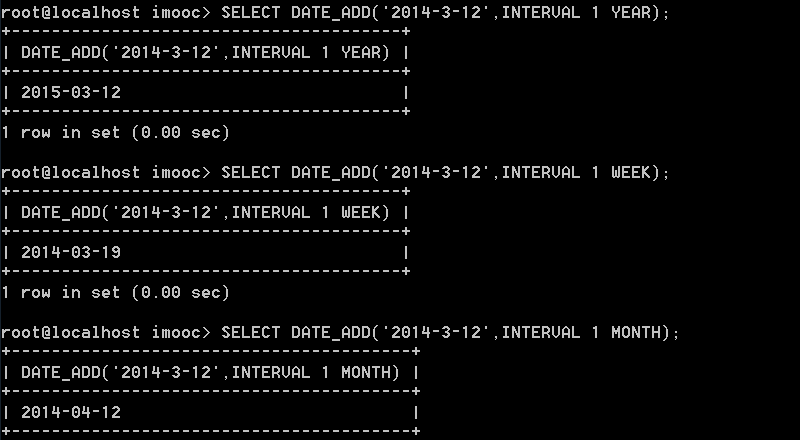
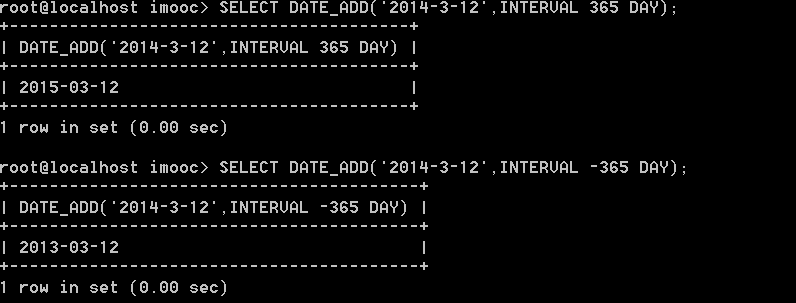
6-4 MySQL 比较运算符和函数

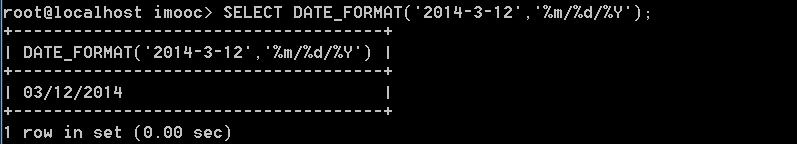


6-5 MySQL 日期时间函数









6-6 MySQL 信息函数



·LAST\_INSERT\_ID()函数需要插入表有AUTO\_INCREMENT的主键的ID字段。若同时插入多条记录，返回第一条记录的ID。

6-7 MySQL 聚合函数



6-8 MySQL 加密函数





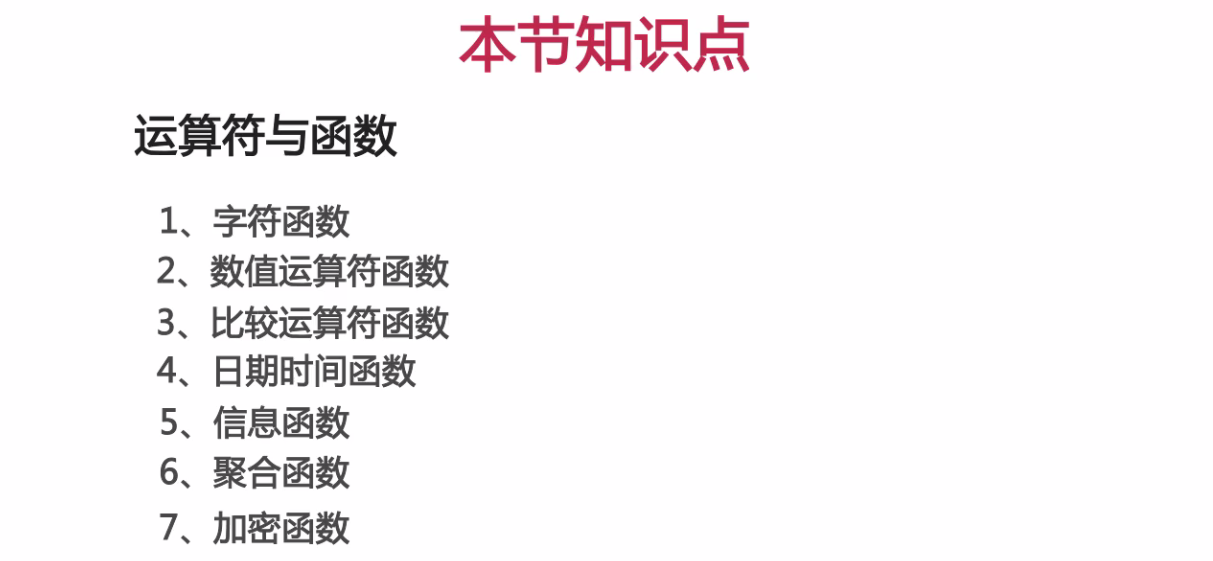
·如果是为了web页面做准备，尽量用MD5()

·PASSWORD()在修改密码时使用

SET PASSWORD()=PASSWORD("new\_password");

第7章 自定义函数

7-1 回顾和概述



7-2 MySQL 自定义函数简介

·自定义函数

用户自定义函数（user-defined function, UDF）是一种对MySQL扩展的途径，其用法与内置函数相同。

·自定义函数的两个必要条件：

（1）参数

（2）返回值

·函数可以返回任意类型的值，同样可以接受这些类型的参数

·**CREATE FUNCTION function\_name**

**RETURNS**

**{STRING|INTEGER|REAL|DECIMAL}**

**routine\_body**

·关于函数体

·函数体由合法的SQL语句构成;

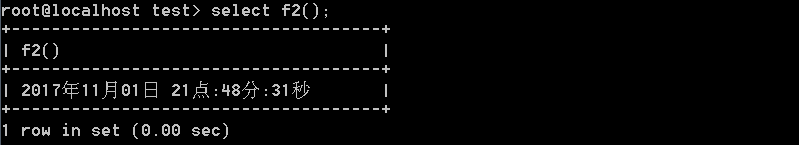
·函数体可以是简单的SELECT或INSERT语句;

·函数体如果为复合结构则使用BEGIN...END语句;

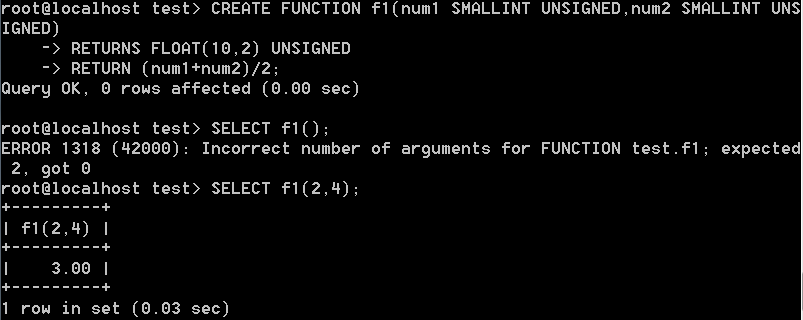
·复合结构可以包含声明、循环、控制结构；

7-3 MySQL 创建不带参数的自定义函数





7-4 MySQL 创建带有参数的自定义函数

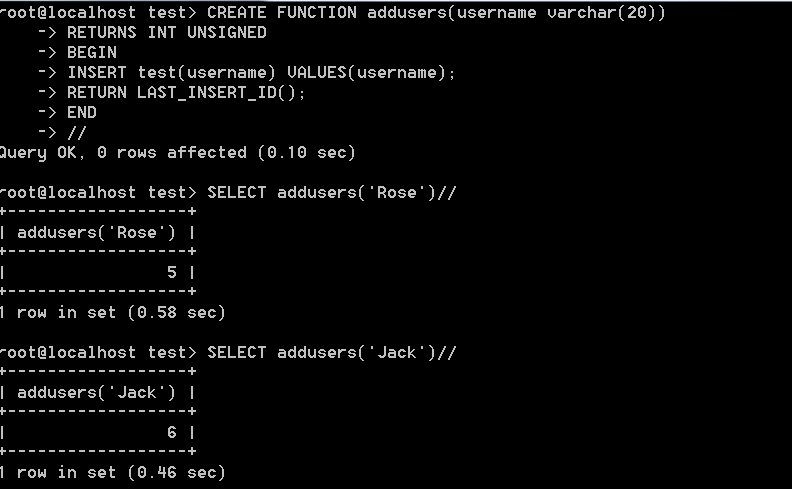


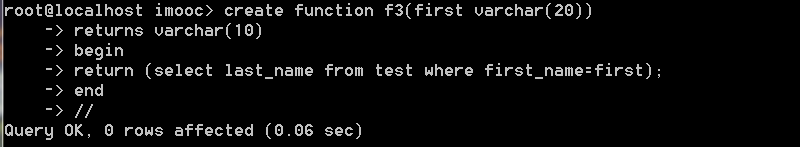
7-5 MySQL 创建具有复合结构函数体的自定义函数

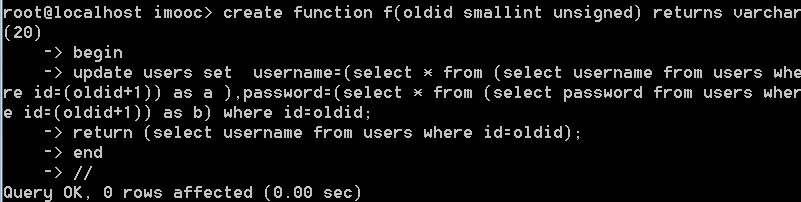
·**DELIMITER symbol**

设置symbol为MySQL语句结束符号









·关于函数体

·函数体由合法的SQL语句构成；

·函数体可以是简单的SELECT或INSERT语句；

·函数体如果为复合结构则使用BEGIN...END语句；

·符合结构可以包含声明，循环，控制结构

·删除函数

**DROP FUNCTION [IF EXISTS] function\_name**

·在mysql的trigger和function中不能出现select \* from table形式的查询，因为其会返回一个结果集；而这在mysql的trigger和function中是不可接受的，但是在存储过程中可以。在trigger和function中可以使用select ... into ...形式的查询。

第8章 MySQL存储过程

8-1 课程回顾

·自定义函数：简称UDF；是对MySQL拓展的一种途径

·创建自定义函数：CREATE FUNCTION......

·自定义函数两个必要条件

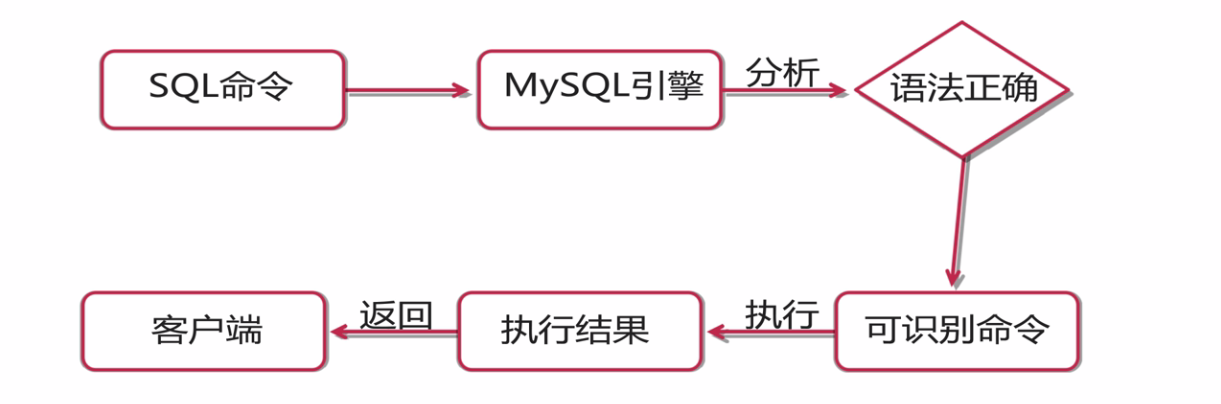
参数：可以由零个或多个

返回值：只能有一个返回值

·具有复合结构的函数体需要使用BEGIN...END来包含

8-2 MySQL 存储过程简介

·MySQL语句执行流程



·存储过程是SQL语句和控制语句的预编译集合，以一个名称存储并作为一个单元处理

跳过语法分析和编译

·存储过程的优点

·增强SQL语句的功能和灵活性

因为在存储过程内可以写控制语句，name就有很强的灵活性，可以完成复杂的判 断以及较复杂的运算

·实现较快的执行速度

如果执行某一个复杂的操作，包含大量的sql语句，那么这些语句都将被mysql 的引擎 执行语法分析、编译、以及执行的过程，所以执行效率相对较低。而存储过程 是预编译的，当客户端第一次调用这个存储过程的时候，mysql的引擎将对它进行 语 法分析、编译等操作，然后把这个编译的结果存储到内存当中，所以，第一次的的执行 效率适合以前一样。但是以后客户端再次调用这个存储过程的时候便直接从内存当中来 执行，所以说效率比较高，速度比较快。

·减少网络流量

如果我们通过客户端每个单独发送sql语句让服务器来执行，那么通过http协议 所提交的数据量相对而言比较大。假设我们准备删除user表中id为3的记录 delete from user where id=3;这句话的字符量大约超过了30个字符，如果我们把他存储成存 储过程的话，假设有一个存储过程叫delete user，我们只需要调用delete user吧id 传过去就可以完成了删除，所以我们会发现，我们只需要来传递存储过程的名字以及要 删除的那个id的号码就可以了。所以他提交给服务器的数量就会少很多。减少了网络 流量

8-3 MySQL 存储过程语法结构解析

·创建存储过程

**CREATE**

**[DEFINER = { user | CURRENT\_USER }]**

**PROCEDURE sp\_name([proc\_parameter[,...]])**

**[characteristic...] routine\_body**

·**proc\_parameter:[IN | OUT | INOUT] param\_name type**

·IN, 表示该参数的值必须在调用存储过程时指定

·OUT, 表示该参数的值可以被存储过程改变，并且可以返回

·INOUT，表示该参数的调用时指定，并且可以被改变和返回。

·特性

·COMMENT:注释

·CONTAINS SQL:包含SQL语句，但不包含读或写数据的语句

·NO SQL:不包含SQL语句

·READS SQL DATA:包含读数据的语句

·MODIFIES SQL DATA:包含写数据的语句

·SQL SECURITY { DEFINER| INVOKER }指明谁有权限来执行

·过程体

·过程由合法的SQL语句构成

·过程体可以是任意SQL语句

·过程体如果为复合结构则使用BEGIN..END语句

·复合结构可以包含声明，循环，控制结构

·注意：不能在一个存储过程中删除另一个存储过程，只能调用另一个存储过程

·区块，条件，循环   
 ·区块定义，常用

**BEGIN**

**......END;**

也可以给区块起别名，如：

**lable:BEGIN**

**...........END lable;**

可以用leave lable;跳出区块，执行区块以后的代码

·条件语句

**IF condition THEN**

**statement**

**ELSE**

**statement**

**END IF;**

·循环语句   
 ·WHILE循环

**[label:] WHILE expression DO**

**statements**

**END WHILE [label] ;**

·LOOP循环

**[label:] LOOP**

**statements**

**END LOOP [label];**

·REPEAT UNTIL循环

**[label:] REPEAT**

**statements**

**UNTIL expression**

**END REPEAT [label] ;**

8-4 MySQL 创建不带参数的存储过程

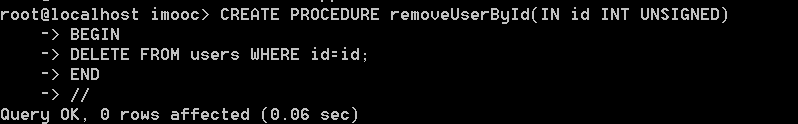


·调用存储过程

·**CALL sp\_name([parameter[,...]])**

·**CALL sp\_name[()]**

8-5 MySQL 创建带有IN类型参数的存储过程



·修改存储过程

**ALTER PROCEDURE sp\_name [characteristic ...]**

**COMMENT 'string'**

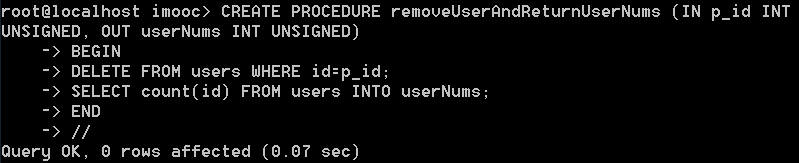
**|{CONTEAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA}**

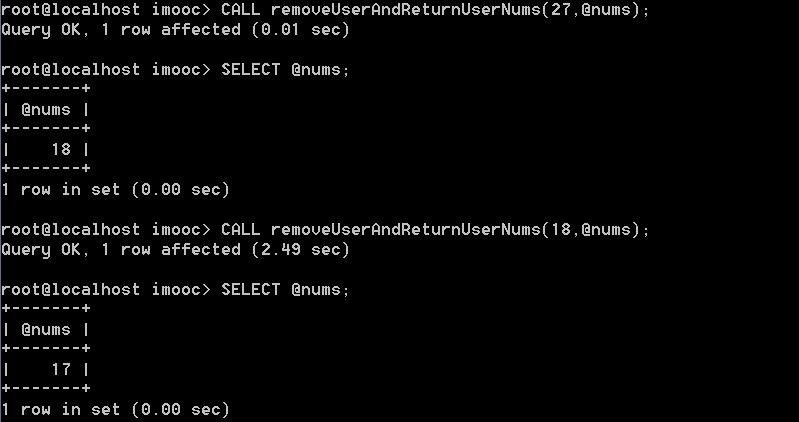
**|SQL SECURITY { DEFINER | INVOKER }**

·删除存储过程

**DROP PROCEDURE [IF EXISTS] sp\_name**

8-6 MySQL 创建带有IN和OUT类型参数的存储过程





·用户变量

跟MySQL的客户端绑定



·mysql变量的术语分类：

·用户变量：以"@"开始，形式为"@变量名"

用户变量跟mysql客户端是绑定的，设置的变量，只对当前用户使用的客户端生效

·全局变量：定义时，以如下两种形式出现，set GLOBAL 变量名 或者 set @@global. 变量名

对所有客户端生效。只有具有super权限才可以设置全局变量

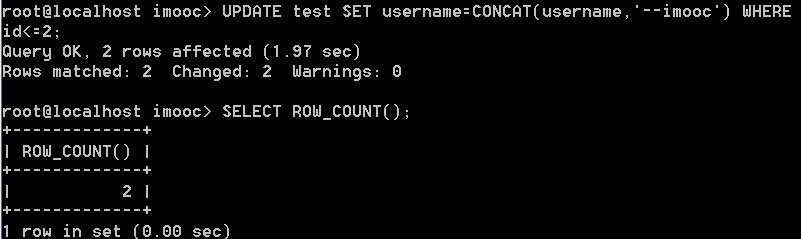
·会话变量：只对连接的客户端有效。

·局部变量：作用范围在begin到end语句块之间。在该语句块里设置的变量 declare可以声明变量，比如 declare @a int，只不过声明全局变量可以省略 declare，而在begin--end里面声明需要用declare，而且要放在第一行。set语 句 是设置不同类型的变量，包括会话变量和全局变量

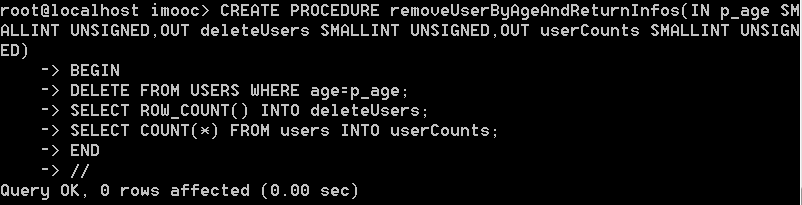
8-7 MySQL 创建带有多个OUT类型参数的存储过程

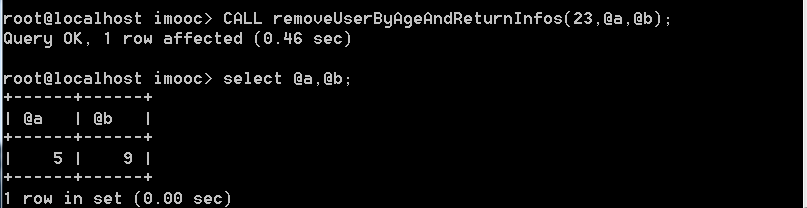
·SELECT ROW\_COUNT()

返回修改的行的数量



·





8-8 MySQL 存储过程与自定义函数的区别

·存储过程与自定义函数的区别

·存储过程实现的功能要复杂一些；而函数的针对性更强

·存储过程可以返回多个值；函数只能有一个返回值

·存储过程一般独立的来执行；而函数可以作为其他SQL语句的组成部分来出现

第9章 MySQL存储引擎

9-1 课程回顾

·存储过程：是SQL语句和控制语句的预编译集合，以一个名称存储并作为一个单元处理

·参数：输入类型 输出类型 输入&&输出

·创建：CREATE...PROCEDURE...

·注意事项：

·创建存储过程或者自定义函数时需要通过DELIMITER语句修改定界符

·如果函数体或过程有多个语句，需要包含在BEGIN...END语句块中

·存储过程通过call来调用。

9-2 MySQL 存储引擎简

·查看数据表的创建命令：

SHOW CREATE table tbl\_name

·MySQL可以将数据以不同的技术存储在文件（内存）中，这种技术就称为存储引擎

·每一种存储引擎使用不同的存储机制、索引技巧、锁定水平，最终提供广泛且不同的功能。

·存储引擎

·MyISAM

·InnoDB

·Memory

·CSV

·Archive

9-3 MySQL 相关知识点之并发处理

·并发控制

当多个连接对记录进行修改时保证数据的一致性和完整性

·锁

·共享锁（读锁）：在同一时间段内，多个用户可以读取同一个资源，读取过程中数据 不会发生任何变化。

·排他锁（写锁）：在任何时候只能有一个用户写入资源，当进行写锁时会阻塞其他的 读锁或者写锁操作。

·锁颗粒

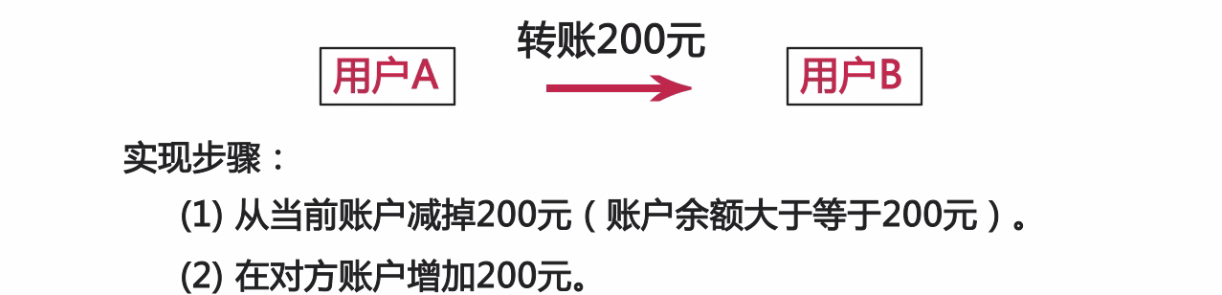
·表锁，是一种开销最小的锁策略

·行锁，是一种开销最大的锁策略

9-4 MySQL 相关知识点之事务处理

·事务

·事务用于保证数据库的完整性



·事务的特性

·原子性（Atomicity）

·一致性（Consistency）

·隔离性（Isolation）

·持久性（Durability）

9-5 MySQL 相关知识点之外键和索引

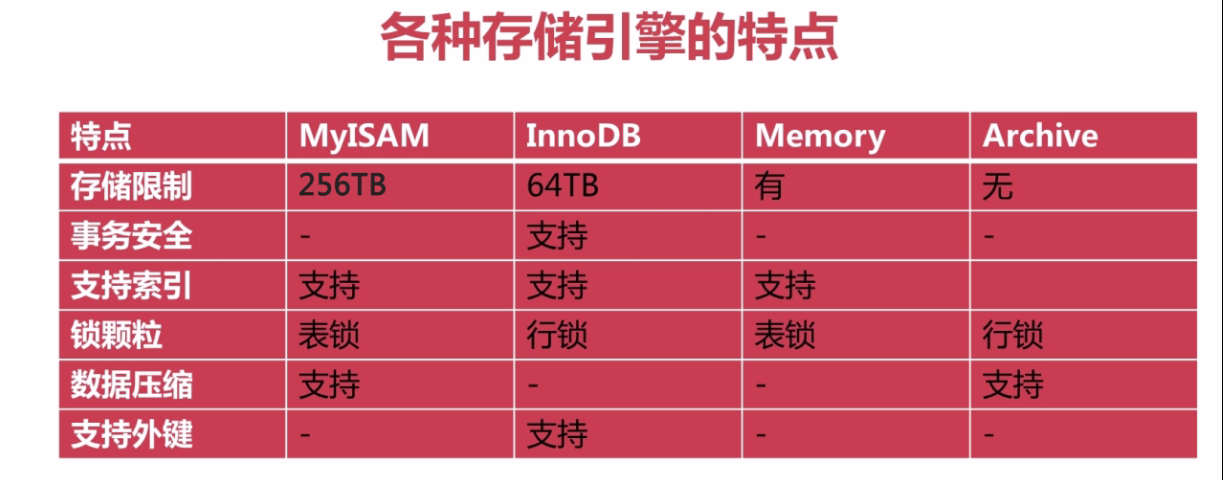
·外键

是保证数据一致性的策略

·索引

是对数据表中一列或多列的值进行排序的一种结构

9-6 MySQL 各个存储引擎特点



·MyISAM：适用于事务的处理不多的情况

·InnoDB：适用于事务处理比较多，需要有外键支持的情况。

9-7 MySQL 设置存储引擎

·修改存储引擎的方法

·通过修改MySQL配置文件实现

default-storage-engine=engine

·通过创建数据表命令实现

**CREATE TABLE table\_name(**

**...**

**...**

**)ENGINE=engine;**

·修改存储类型的方法

·通过修改数据表命令实现

**ALTER TABLE table\_name ENGINE [=] engine\_name;**