MySQL基础

[MySQL基础 1](#_Toc10868)

[一 配置 3](#_Toc17051)

[1. 目录结构 3](#_Toc29580)

[2. 登录参数 3](#_Toc21942)

[3. 退出参数 3](#_Toc27829)

[4.修改提示符 4](#_Toc23032)

[5. 常用基本命令 4](#_Toc31584)

[6. 语句规范 4](#_Toc7103)

[7.数据库操作 4](#_Toc11335)

[1.创建数据库 4](#_Toc26481)

[2.查看警告：SHOW WARNINGS; 4](#_Toc27293)

[3.查看数据库：SHOW DATABASES; 4](#_Toc4684)

[4.查看编码方式:SHOW CREATE DATABASE db\_name; 4](#_Toc23065)

[5.修改编码方式 4](#_Toc3796)

[6. 删除数据库 4](#_Toc12850)

[三 数据类型 4](#_Toc4510)

[1. 整形 5](#_Toc11317)

[2. 浮点型 5](#_Toc12683)

[3. 日期类型 5](#_Toc25383)

[4. 字符类型 6](#_Toc1706)

[四 数据表操作 6](#_Toc15243)

[1. 打开数据库 6](#_Toc9563)

[2. 创建数据表 6](#_Toc23171)

[3. 查看数据表——SHOW TABLES 7](#_Toc32623)

[4. 查看数据表结构 7](#_Toc12016)

[5. 插入记录 7](#_Toc17234)

[6. 记录查找 7](#_Toc25710)

[7.空与非空设置 7](#_Toc20870)

[8. 自动递增编号——AUTO\_INCREMENT 7](#_Toc5990)

[9. 主键约束——PRIMARY KEY 7](#_Toc25338)

[10. 唯一约束——UNIQUE KEY 7](#_Toc6803)

[五 约束 9](#_Toc3835)

[1.外键约束——FOREIGN KEY 9](#_Toc24990)

[六 操作数据表中记录 12](#_Toc5587)

[1. 插入记录 12](#_Toc456)

[2. 更新记录 12](#_Toc15707)

[1. 单表更新 12](#_Toc187)

[2. 单表删除记录 12](#_Toc14080)

[3.查询 12](#_Toc10993)

[七 子查询与连接 14](#_Toc782)

[1. 子查询：指出现在其他SQL语句内的SELECT子句 14](#_Toc13913)

[2. 使用比较运算符的子查询 14](#_Toc9025)

[3. 使用in或者not in的子查询 15](#_Toc23329)

[4. 使用INSERT...SELECT...插入 16](#_Toc16103)

[5. 多表更新 16](#_Toc7118)

[1.语法 16](#_Toc11553)

[2. 参照关系 16](#_Toc32515)

[3.实例 16](#_Toc15866)

[4. 创建并插入查询结果CREATE...SELECT 16](#_Toc20655)

[5. 数据表参照 16](#_Toc8561)

[6. 连接类型 17](#_Toc28025)

[7. 多表连接 18](#_Toc942)

[8. 无限级分类 19](#_Toc29335)

[9.删除重复记录 19](#_Toc28660)

[八 运算符和函数 20](#_Toc15746)

[1. 字符函数 20](#_Toc22547)

[1.CONCAT(连接内容，连接内容) 20](#_Toc16931)

[2.CONCAT\_WS(‘指定分隔符’,连接内容,连接内容) 20](#_Toc19548)

[3.FORMAT(数字，保留小数位) 20](#_Toc13153)

[4. LOWER(字符串); 和 UPPER(字符串); 20](#_Toc4213)

[6. LENGTH(‘string’); //计算字符长度，并且包括空格 21](#_Toc6465)

[2. 数值运算符函数 21](#_Toc23413)

[1. CEIL() 和 FLOOR();和ROUND(); 21](#_Toc2300)

[3. 比较运算符函数 22](#_Toc17969)

[4. 日期时间函数 22](#_Toc5803)

[1. DATE\_ADD(); 22](#_Toc16545)

[2. DATEDIFF(); 23](#_Toc5506)

[3. DATE\_FORMAT(); 23](#_Toc28615)

[5. 信息函数 23](#_Toc8878)

[6. 聚合函数 23](#_Toc8068)

[7. 加密函数 23](#_Toc6224)

[九 自定义函数 24](#_Toc4145)

[1.必要条件 24](#_Toc12592)

[2.创建自定义函数 24](#_Toc26112)

[3.函数体 24](#_Toc11030)

[4. 不带参数的自定义函数 24](#_Toc28787)

[5.带参数的自定义函数 24](#_Toc25841)

[十 存储过程 24](#_Toc29284)

[1.语法结构 24](#_Toc7989)

[2.存储过程调用 25](#_Toc6004)

[3.不带参数的存储过程 25](#_Toc30737)

[4.带IN类型参数的存储过程 26](#_Toc27104)

[5.带IN类型参数和OUT类型参数的存储过程 26](#_Toc19052)

[6.带IN类型参数和多个OUT类型参数的存储过程 26](#_Toc29708)

一 配置

1. 目录结构

●bin目录：存储可执行文件

●data目录：存储数据文件

●docs目录：文档

●include目录：存储包含的头文件

●lib目录：存储库文件

●share：储存错误信息和字符集文件

二 数据库操作

1. 登录参数





1. 退出参数

●exid;

●quit;

●\q;

●cls:清屏命令

4.修改提示符



···

Mysql> prompt \u@\h \d>

//修改为用户名@服务器 数据库名>

root@localhost (none)> use test //切换数据库

root@localhost (test)> //切换数据库

···

1. 常用基本命令
2. SELECT VERSION(); //显示当前服务器版本，注意分号，不区分大小写
3. SELECT NOW(); //显示当前日期时间
4. SELECT USER(); //显示当前用户
5. 语句规范
6. 关键字和函数名称全部大写；
7. 数据库名称、表名称、字段名称全部小写
8. SQL语句必须分号结尾；

7.数据库操作

1.创建数据库

CREATE { DATABASE | SCHEMA} [IF NOT EXISTS] db\_name [DEFAULT] CHARACTER SET [=] character\_name;

●{}为必选项，DATABASE为数据库；SCHEMA为模型；选哪个都一样

●[]为可选项，IF NOT EXISTS为忽略错误信息；

●[DEFAULT] CHARACTER SET [=] character\_name：设置数据编码

2.查看警告：SHOW WARNINGS;

3.查看数据库：SHOW DATABASES;

4.查看编码方式:SHOW CREATE DATABASE db\_name;

5.修改编码方式

ALTER { DATABASE | SCHEMA} [db\_name] [DEFAULT] CHARACTER SET [=] character\_name;

1. 删除数据库

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

三 数据类型

1. 整形

符号位：0为正，1为负



1. 浮点型



1. 日期类型



1. 字符类型

ENUM :单选类型

SET：多选类型（集合）



四 数据表操作

1. 打开数据库

●USE db\_name;

●SELECT DATABASE();查看当前打开的数据库

1. 创建数据表

···

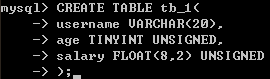
CREATE TABLE [IF NOT EXISTS] table\_name(

Column\_name data\_type,

...

);

// UNSIGNED 无符号



···

1. 查看数据表——SHOW TABLES

···

SHOW TABLES [FROM db\_name] [LIKE ‘pattern’ | WHERE expr]

···

1. 查看数据表结构

···

SHOW COLUMNS FROM table\_name;

···

1. 插入记录

···

INSERT [INTO] tbl\_name [(column\_name,...)] VALUES (val,...);

//字段名选填，如果不书写字段名，值必须一一对应每个字段

···

1. 记录查找

···

SELECT expr,... FROM tbl\_name;

//SELECT \* FROM tb\_1;

···

7.空与非空设置

···

CREATE TABLE tb\_2 (

Username VARCHAR(20) NOT NULL; //不允许为空

Age TINYINT UNSIGNED NULL; //非符号，允许为空

)

SHOW COLUMNS FROM tb\_2;

INSERT tb\_2 VALUES (“LIN”,NULL); //正常插入

INSERT tb\_2 VALUES(NULL,25); //提示错误

···

1. 自动递增编号——AUTO\_INCREMENT

●自动编号，且必须与主键组合使用

●默认情况，初始值为1，每次增量为1

1. 主键约束——PRIMARY KEY

●主键非必须与AUTO\_INCREMENT一起使用

●每张表只能存在一个主键

●主键保证记录唯一性

●主键自动为NOT NULL

1. 唯一约束——UNIQUE KEY

●唯一约束保证记录的唯一性

●唯一约束的字段可以为NULL

●每张数据表可以存在多个唯一约束

1. 添加单列数据表

···

ALTER TABLE tb\_name ADD [COLUMN] col\_name col\_definition [FIRST | AFTER col\_name]

···

1. 添加多列数据表

···

ALTER TABLE tb\_name ADD [COLUMN] (col\_name col\_definition,...);

···

1. 删除列

···

ALTER TABLE tb\_name DROP[COLUMN] col\_name,DROP[COLUMN] col\_name;

···

1. 添加/删除主键约束

···

ALTER TABLE tb\_name ADD [CONSTRAINT[symbol]] PRIMARY KEY [index\_type] (index\_col\_name,...);

···

···

ALTER TABLE tb\_name DROP PRIMARY KEY;

···

1. 添加/删除唯一约束

···

ALTER TABLE tb\_name ADD [CONSTRAINT[symbol]] UNIQUE [INDEX | KEY] [index\_name][inde\_type] (index\_col\_name,...);

···

···

ALTER TABLE tb\_name DROP {INDEX | KEY} index\_name;

//注意:此处是删除约束，所以是删除索引名

//SHOW INDEXES FROM tb\_name； 查看索引名

···

1. 添加/删除外键约束

···

ALTER TABLE tb\_name ADD [CONSTRAINT[symbol]] FOREIGN KEY [index\_name](index\_col\_name,...) reference\_definition;

···

···

ALTER TABLE tb\_name DROP FOREIGN KEY fk\_name(外键名);

//注意:此处是删除约束，所以是删除外键名

//SHOW CREATE TABLE tb\_name； 查看contraint名

···

1. 添加/删除默认约束

···

ALTER TABLE tb\_name ALTER [COLUMN] col\_name {SET DEFAULT literal | DROP DEFAULT};

···

1. 修改列定义

···

ALTER TABLE tb\_name MODIFY [COLUMN] col\_name col\_definition [FIRST | AFTER col\_name]

···

1. 修改列名称

···

ALTER TABLE tb\_name CHANGE [COLUMN] old\_col\_name new\_col\_name col\_definition [FIRST | AFTER col\_name];

···

1. 修改数据表名称

···

ALTER TABLE tb\_name RENAME [TO|AS] new\_tb\_name;

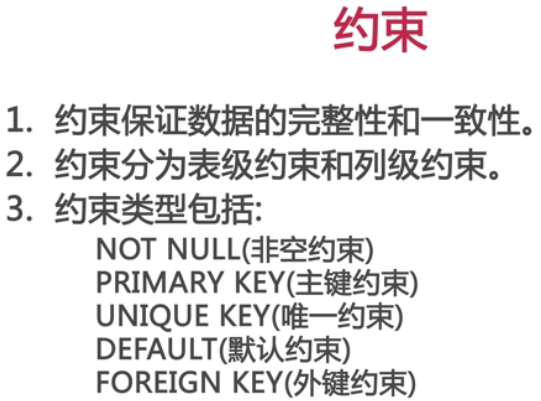
···

···

RENAME TABLE tb\_name TO new\_tb\_name[,tb\_name TO new\_tb\_name,...]

···

五 约束



1.外键约束——FOREIGN KEY

●保持数据一致性和完整性

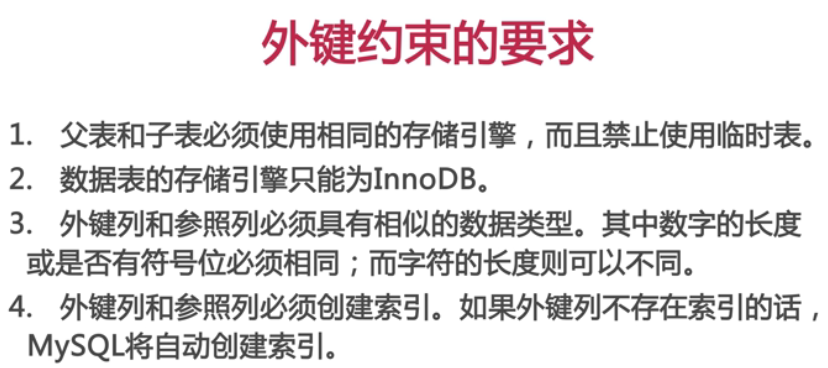
●实现一对一和一对多的关系

●父表：子表所参照的表为父表

●子表：具有外键列的表为子表

●外键列：添加FOREIGN KEY 的列（字段名）

●参照列：



●修改数据表的存储引擎，配置文件：default-storage-engine=InnoDB

●外键格式

···

FOREIGN KEY (col\_name) REFERENCES tb\_name (col\_name);

···

···

CREATE TABLE provinces（

Id SMALLINT UNSIGNED PRIMARY KEY AUTO\_INCREMENT,

Pname VARCHAR(20) NOT NULL

）;

CAEATE TABLE users(

Id SMALLINT UNSIGNED PRIMARY KEY AUTO\_INCREMENT,

Username VARCHAR(10) NOT NULL,

Pid SMALLINT UNSIGNED,

FOREIGN KEY (pid) REFERENCES provinces (id)

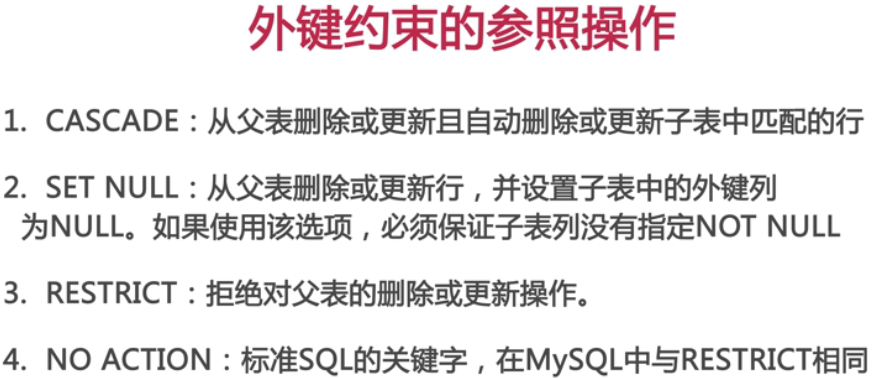
);

SHOW INDEXES FROM provinces\G;

SHOW INDEXES FROM users\G; //\G:以网格形式输出展示

···

1. 外键约束的参照操作



···

CREATE TABLE provinces（

Id SMALLINT UNSIGNED PRIMARY KEY AUTO\_INCREMENT,

Pname VARCHAR(20) NOT NULL

）;

CAEATE TABLE users(

Id SMALLINT UNSIGNED PRIMARY KEY AUTO\_INCREMENT,

Username VARCHAR(10) NOT NULL,

Pid SMALLINT UNSIGNED,

FOREIGN KEY (pid) REFERENCES provinces (id) ON DELETE CASCADE;

);

INSERT provinces (pname) VALUES (‘A’);

INSERT provinces (pname) VALUES (‘B’);

INSERT provinces (pname) VALUES (‘C’);

INSERT users (username,pid) VALUES (‘L’,1);

INSERT users (username,pid) VALUES (‘L’,3);

INSERT users (username,pid) VALUES (‘L’,3);

//现在如果删除Provinces表的id=3，users表中pid为3的记录也会删除

DELETE FROM provinces WHERE id=3；

SELECT \*FROM users； //验证是否正确

···

//在实际的开发过程中，我们很少使用物理的外键约束，很多都去使用逻辑的外键约束，因为物理的外键约束只有INNODB这种引擎才会支持，像我们另外的一种引擎MYISAM的引擎则不支持，反过来说，如果我想创建的数据表，假设存储引擎为MYISAM，而且又想使用外键约束的话，其实是不可能实现的，所以说，我们在实际的项目开发中，我们不去定义物理的外键，所谓的逻辑外键指的是就是我们在定义两张表的结构的时候，我们是按照存在的某种结构的方式去定义，但是不去使用FOREIGN KEY这个关键词来定义。



六 操作数据表中记录

1. 插入记录

···

INSERT [INTO] tb\_name [(col\_name,...)] {VALUES | VALUE} ({expr | default},...),(...),...;

//一次可插入多条语句

···

···

INSERT [INTO] tb\_name SET col\_name = {expr | default},...

//与第一种方法区别，此方法可以使用子查询，并且只能插入一条语句

···

···

INSERT [INTO] tb\_name [(col\_name,...)] SELECT ...

//此方法可以将查询结果插入到指定数据表

INSERT users1(username) SELECT username FROM users WHERE age > 30;

···

1. 更新记录
2. 单表更新

···

UPDATE [LOW\_PRIORITY] [IGNORE] tb\_reference SET col\_name1 = {expr | default} [,col\_name2 = {expr | default},...] [WHERE where\_condition]

UPDATE users SET age = age +1; //更新所有age列的值 +1

UPDATE users SET age = age + id,sex = 0;

//更新所有age列的值+id字段的值；sex字段的值全部置0

UPDATE users SET age = age +10 WHERE id%2=0;

//更新id为偶数位的age列的记录值+10；

···

1. 单表删除记录

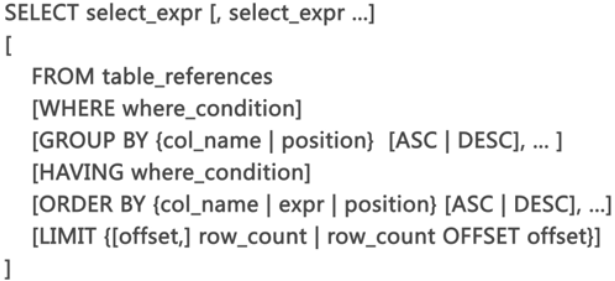
···

DELETE FROM tb\_name [WHERE where\_condition];

//删除记录不影响自增量递增；仍以最大量递增

···

3.查询



1. 每一个表达式表示想要查找的一列，必须至少有一个

···

SELECT id FROM users;

···

1. 多个列之间以英文逗号隔开

···

SELECT id,username FROM users;

···

3.星号(\*)表示所有列。Tb\_name.\*可以表示命名表的所有列。

···

SELECT \* FROM users;

SELECT users.id,users.username FROM users;

SELECT users.username,users.id FROM users;

//表达式顺序影响查询的显示结果

···

1. 查询表达式可以使用[AS] alias\_name为其赋予别名。

···

SELECT id AS user\_id,username\_uname FROM users;

//查询结果字段名会显示别名

···

1. 别名可用于GROUP BY,ORDER BY 或HAVING子句。
2. Where

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支持的函数和运算符。

1. 查询分组

···

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

SELECT sex FROM users GROUP BY sex;

···

1. 设置分组条件

···

[HAVING where\_condition]

SELECT sex,age FROM users GROUP BY sex HAVING age>35;

//运用HAVING分组条件必须满足条件字段在sql语句中或者使用聚合函数（只返回一个结果;平均值，最大值，最小值，求和，计数等）；

···

1. 对结构进行排序

···

[ORDER BY {col\_name | expr | position} [ASE | DESC]]

SELECT \* FROM users ORDER by id DESC;

SELECT \* FROM users ORDER by age,id DESC;

···

1. 限制返回结果的数量

···

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

SELECT \* FROM users LIMIT 2; //显示结果前两行

SELECT \* FROM users LIMIT 2,2; //跳过结果前两行，显示3、4

//分页偏移量：（当前页面-1）\*每页显示量

···

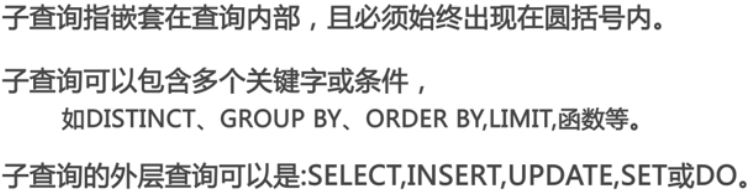
七 子查询与连接

1. 子查询：指出现在其他SQL语句内的SELECT子句

···

SELECT \* FROM t1 WHERE col1 = (SELECT \* FROM T2);

···



1. 使用比较运算符的子查询

···

SELECT goods\_id,goods\_name,goods\_price FROM tdb\_goods WHERE goods\_price > ALL (SELECT goods\_price FROM tdb\_goods WHERE goods\_name = ‘超级本’);

···







1. 使用in或者not in的子查询

···

SELECT goods\_id,goods\_name,goods\_price FROM tdb\_goods WHERE goods\_price IN (SELECT goods\_price FROM tdb\_goods WHERE goods\_name = ‘超级本’);

SELECT goods\_id,goods\_name,goods\_price FROM tdb\_goods WHERE goods\_price NOT IN (SELECT goods\_price FROM tdb\_goods WHERE goods\_name = ‘超级本’);

···



1. 使用INSERT...SELECT...插入

···

INSERT tdb\_goods\_cates(cate\_name) SELECT goods\_cate FROM tdb\_goods GROUP BY goods\_cate;

···

1. 多表更新

1.语法

···

UPDATE [LOW\_PRIORITY] [IGNORE] tb\_reference SET col\_name1 = {expr | default} [,col\_name2 = {expr | default},...] [WHERE where\_condition]

···

1. 参照关系

···

Tbl\_reference

{ [INNER | CROSS] JOIN | {LEFT | RIGGHT} [OUTER] JOIN }

Tbl\_reference

ON conditional\_expr //ON之后跟随连接条件

···

3.实例

···

UPDATE tdb\_goods INNER JOIN tdb\_goods\_cates ON goods\_cate = cate\_name SET goods\_cate = cate\_id;

···

···

UPDATE tdb\_goods AS g INNER JOIN tdb\_goods\_brands AS b ON g.brand\_name = b.brand\_name SET g.brand\_name = b.brand\_id;

···

1. 创建并插入查询结果CREATE...SELECT

···

CREATE TABLE tdb\_goods\_brands (

brand\_id SMALLINT UNSIGNED PRIMARY KEY AUTO\_INCREMENT,

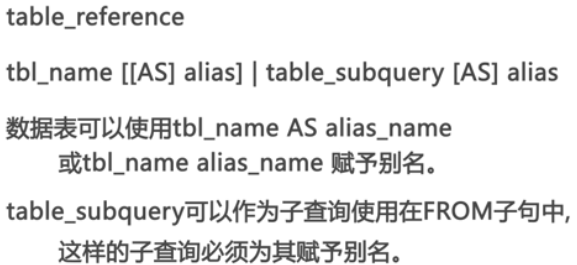
brand\_name VARCHAR(40) NOT NULL

) SELECT brand\_name FROM tdb\_goods GROUP BY brand\_name;

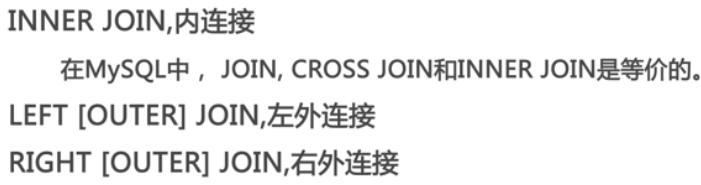
UPDATE tdb\_goods AS g INNER JOIN tdb\_goods\_brands AS b ON g.brand\_name = b.brand\_name SET g.brand\_name = b.brand\_id;

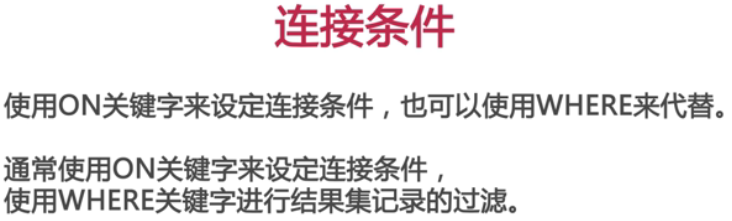
···

1. 数据表参照

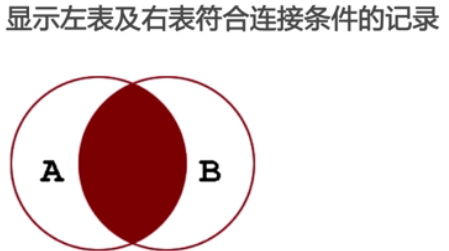


1. 连接类型





●内连接：显示连接表符合连接条件的记录



···

mysql> SELECT goods\_id,goods\_name,brand\_name FROM tdb\_goods AS g INNER JOIN tdb\_goods\_cates AS c ON g.cate\_id = c.cate\_id;

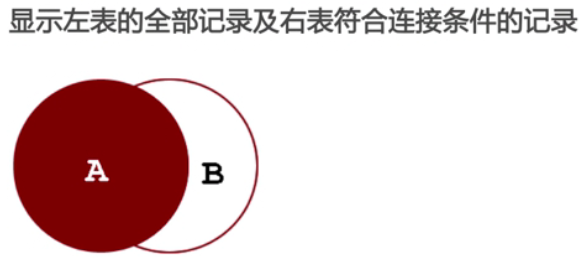
···

●左外连接

···

mysql> SELECT goods\_id,goods\_name,brand\_name FROM tdb\_goods AS g LEFT JOIN tdb\_goods\_cates AS c ON g.cate\_id = c.cate\_id;

···



●右外连接

···

mysql> SELECT goods\_id,goods\_name,brand\_name FROM tdb\_goods AS g RIGHT JOIN tdb\_goods\_cates AS c ON g.cate\_id = c.cate\_id;

···



1. 多表连接

···

mysql> SELECT goods\_id,goods\_name,cate\_name,brand\_name FROM tdb\_goods AS g INNER JOIN tdb\_goods\_cates AS c ON g.cate\_id = c.cate\_id INNER JOIN tdb\_goods\_brands AS b ON g.brand\_id = b.brand\_id;

···

1. 无限级分类

···

CREATE TABLE tdb\_goods\_types(

type\_id SMALLINT UNSIGNED PRIMARY KEY AUTO\_INCREMENT,

type\_name VARCHAR(20) NOT NULL,

parent\_id SMALLINT UNSIGNED NOT NULL DEFAULT 0

);

INSERT tdb\_goods\_types(type\_name,parent\_id) VALUES('家用电器',DEFAULT);

INSERT tdb\_goods\_types(type\_name,parent\_id) VALUES('电脑、办公',DEFAULT);

INSERT tdb\_goods\_types(type\_name,parent\_id) VALUES('大家电',1);

INSERT tdb\_goods\_types(type\_name,parent\_id) VALUES('生活电器',1);

INSERT tdb\_goods\_types(type\_name,parent\_id) VALUES('平板电视',3);

INSERT tdb\_goods\_types(type\_name,parent\_id) VALUES('空调',3);

INSERT tdb\_goods\_types(type\_name,parent\_id) VALUES('电风扇',4);

INSERT tdb\_goods\_types(type\_name,parent\_id) VALUES('饮水机',4);

INSERT tdb\_goods\_types(type\_name,parent\_id) VALUES('电脑整机',2);

INSERT tdb\_goods\_types(type\_name,parent\_id) VALUES('电脑配件',2);

INSERT tdb\_goods\_types(type\_name,parent\_id) VALUES('笔记本',9);

INSERT tdb\_goods\_types(type\_name,parent\_id) VALUES('超级本',9);

INSERT tdb\_goods\_types(type\_name,parent\_id) VALUES('游戏本',9);

INSERT tdb\_goods\_types(type\_name,parent\_id) VALUES('CPU',10);

INSERT tdb\_goods\_types(type\_name,parent\_id) VALUES('主机',10);

···

···

//查找所有分类及其父类

SELECT s.type\_id,s.type\_name,p.type\_name AS parent\_name FROM tdb\_goods\_types AS s LEFT JOIN tdb\_goods\_types AS p ON s.parent\_id = p.type\_id;

···

···

//查找分类及其子类

SELECT s.type\_id,s.type\_name,p.type\_name AS child\_name FROM tdb\_goods\_types AS s LEFT JOIN tdb\_goods\_types AS p ON s.type\_id = p.parent\_id WHERE s.type\_name = "家用电器";

···

···

//查找子类个数

SELECT s.type\_id,s.type\_name,COUNT(p.type\_name) AS child\_count FROM tdb\_goods\_types AS s LEFT JOIN tdb\_goods\_types AS p ON s.type\_id = p.parent\_id GROUP BY s.type\_name ORDER BY type\_id;

···

9.删除重复记录

···

DELETE t1 FROM tdb\_goods AS t1 LEFT JOIN (SELECT goods\_id,goods\_name FROM tdb\_goods GROUP BY goods\_name HAVING COUNT(goods\_name) >=2) AS t2 ON t1.goods\_name = t2.goods\_name WHERE t1.goods\_id > t2.goods\_id;

···

八 运算符和函数

1. 字符函数





1.CONCAT(连接内容，连接内容)

···

SELECT CONCAT(goods\_id,'-',goods\_name) AS concat\_name FROM tdb\_goods;

···

2.CONCAT\_WS(‘指定分隔符’,连接内容,连接内容)

···

SELECT CONCAT\_WS('|',goods\_id,goods\_name) FROM tdb\_goods;

···

3.FORMAT(数字，保留小数位)

···

SELECT FORMAT(123456.75，1); //保留一位小数

···

1. LOWER(字符串); 和 UPPER(字符串);

···

SELECT LOWER(“MySQL”);

SELECT UPPER(“MySQL”);

···

1. LEFT(‘字符串’,获取位数); 和 RIGHT(‘字符串’,获取位数);和SUBSTRING(‘string’,’开始位置’,’个数’);

···

SELECT LEFT(‘MySQL’,2);

SELECT LOWER(RIGHT(‘MySQL’,3));

SELECT SUBSTRING(‘MYSQL’,2,4); //位置从1开始；-1为倒数操作

···

1. LENGTH(‘string’); //计算字符长度，并且包括空格
2. LTRIM();和RTRIM();和TRIM();

···

SELECT LTRIM(‘ MYSQL ’);

SELECT RTRIM(‘ MYSQL ’);

SELECT TRIM(‘ MYSQL ’);

SELECT TRIM(LEADING ‘?’ FROM ‘???MYSQL???’); //删除前导‘？’

SELECT TRIM(TRAILING ‘?’ FROM ‘???MYSQL???’); //删除后导‘？’

SELECT TRIM(BOTH ‘?’ FROM ‘???MYSQL???’); //删除全部‘？’

SELECT TRIM(BOTH ‘?’ FROM ‘???MY???SQL???’); //中间无法删除

···

1. REPLACE(‘string’,’要替换的内容’,’替换的内容’);

···

SELECT REPLACE('???my??sql??','?','');

SELECT REPLACE('???my??sql??','?','--');

···

1. [NOT] LIKE;——包含

···

//%：表示任意个字符

//\_：表示任意一个字符

SELECT \* FROM test WHERE frist\_name LINK ‘%o%’; //查找 frist\_name 包含‘o’的记录

SELECT \* FROM test WHERE frist\_name LINK ‘%1%%’ ESCAPE ‘1’; // ESCAPE 表示1后面不需要进行解析

···

1. 数值运算符函数



1. CEIL() 和 FLOOR();和ROUND();

···

SELECT CEIL(123.56);

SELECT FLOOR(123.56);

ROUND(123.56，1);

ROUND(123.56，-2); //四舍五入到百位数

···

2.DIV

···

SELECR 3 DIV 4;

···

3.MOD(%)

···

SELECT 5 MOD 3;

···

1. POWER()

···

SELECT POWER(3,3); //3的3次方

···

1. TRUNCATE()

···

SELECT TRUNCATE(123.59,1); //截取小数点最后一位位

SELECT TRUNCATE(123.59,-1); //截取个位

···

1. 比较运算符函数



1. 日期时间函数



1. DATE\_ADD();

···

SELECT DATE\_ADD('2016-10-8',INTERVAL 365 DAY);

SELECT DATE\_ADD('2016-10-8',INTERVAL -365 DAY);

SELECT DATE\_ADD('2016-10-8',INTERVAL 1 YEAR);

SELECT DATE\_ADD('2016-10-31',INTERVAL 1 MONTH);

SELECT DATE\_ADD('2016-10-8',INTERVAL 3 WEEK);

···

1. DATEDIFF();

···

SELECT DATEDIFF(‘2016-10-9’,’2016-10-7’);

···

1. DATE\_FORMAT();

···

SELECT DATE\_FORMAT(‘2016-3-2’,’%m/%d/%y’);

//%:前导0；

···

1. 信息函数



1. 聚合函数

只有一个返回值



1. 加密函数

修改数据库密码。平时MD5()用于WEB 页面上的密码加密，但是PASSWORD主要用于登录信息的加密，比如数据库



···

SET PASSWORD=PASSWORD(‘root’);

···

九 自定义函数

1.必要条件

1.参数

2.返回值

3.函数可以返回任意类型的值，同样可以接收这些类型的参数

4.参数的数量理论上不大于1024个

2.创建自定义函数

···

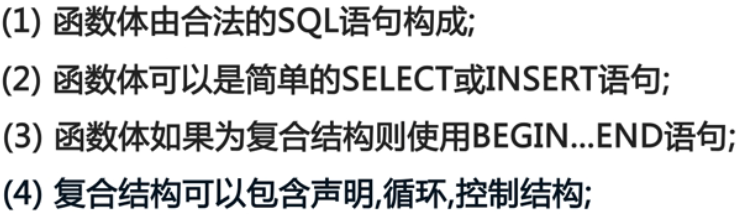
CREATE FUNCTION function\_name RETURNS

{STRING | INTEGER | REAL | DECIMAL}

Routine\_body

···

3.函数体



1. 不带参数的自定义函数

···

CREATE FUNCTION f1() RETURNS VARCHAR(30) RETURN DATE\_FORMAT(NOW(),’%Y年%m月%d日 %H时:%i分:%s秒’)

···

5.带参数的自定义函数

···

CREATE FUNCTION f2(num1 SMALLINT UNSIGNED,num1 SMALLINT UNSIGNED)

RETURNS FLOAT(10,2) UNSIGNED

RETURN (num1+num2)/2;

···

十 存储过程

1.语法结构

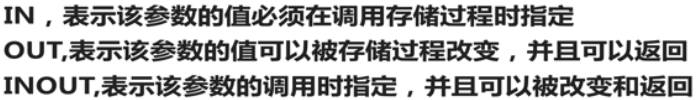
···

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

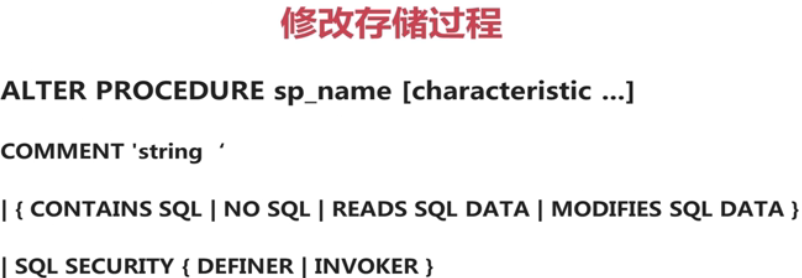
PROCEDURE sp\_name [proc\_parameter[,...]] [characteristic] rountine\_body

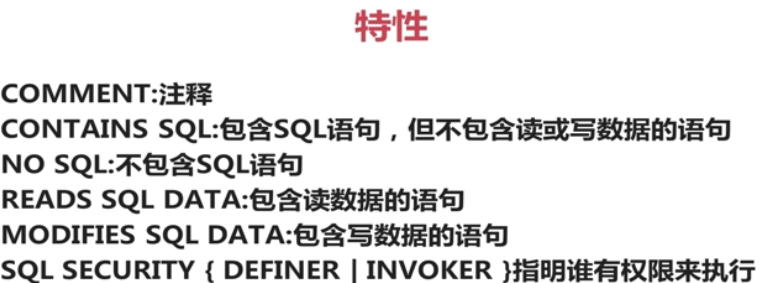
Pro\_parameter

[IN | OUT |INOUT] para\_name type



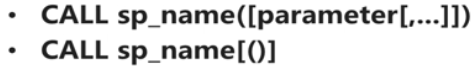
···





2.存储过程调用

···



···

3.不带参数的存储过程

···

CREATE PROCEDURE sq1 SELECT VERSION();

···

4.带IN类型参数的存储过程

···

DELIMITER // //修改结束符

CREATE PROCEDURE userRemoveByID( IN u\_id INT UNSIGNED) BEGIN DELETE FROM tdb\_users WHERE id = u\_id; END;

···

5.带IN类型参数和OUT类型参数的存储过程

···

CREATE PROCEDURE removeUserAndReturnUserNums(IN u\_id INT UNSIGNED,OUT u\_nums INT UNSIGNED) BEGIN

DELETE FROM tdb\_users WHERE id = u\_id;

SELECT COUNT(id) FROM tdb\_users INTO u\_nums;

END;

CALL removeUserAndReturnUserNums(10,@num);

SELECT COUNT(id) FROM tdb\_users INTO u\_nums;/\*\* 该语句中的 INTO 含义就是将 SELECT 语句结果的表达式返回到 u\_nums变量中 \*/

CALL removerUserAndReturnUserName(10,@nums); /\*\* @nums 所代表的就是用户变量，可用 SELECT @nums 输出 \*/

用 DECLARE 声明的变量是局部变量，局部变量只能存在于 BEGIN...END 之间，且声明时必须置于 BEGIN...END 的第一行

而通过 SELECT...INTO.../SET @id = 07 这种方法设置的变量我们称之为用户变量，只能存在于当前用户所使用的客户端有效。

CALL rmUserAndRtUserNums(27, @nums);

SELECT @nums; //@nums - 就是用户变量

DECLARE声明的变量都是在BEGIN与END之间，是局部变量

SET @i = 7; //通过@或SET设置的变量称为用户变量

···

6.带IN类型参数和多个OUT类型参数的存储过程

···

// ROW\_COUNT(); //当前被影响的行数

CREATE PROCEDURE removeUserAge(IN u\_age INT UNSIGNED,OUT delNums SMALLINT UNSIGNED,OUT userNums SMALLINT UNSIGNED) BEGIN

DELETE FROM tdb\_users WHERE age = u\_age;

SELECT ROW\_COUNT() INTO delNums;

SELECT COUNT(id) FROM tdb\_users INTO userNums;

END

CALL removeUserAge(21,@a,@b);

SELECT @a;

SELECT @b;

···

十一 存储引擎



