

## MySQL

### 与MySQL的第一次亲密接触

#### 数据库的相关概念

##### 一、数据库的好处

- 1、可以持久化数据到本地
- 2、结构化查询

##### 二、数据库的常见概念 ★

- 1、DB：数据库，存储数据的容器
- 2、DBMS：数据库管理系统，又称为数据库软件或数据库产品，用于创建或管理DB
- 3、SQL：结构化查询语言，用于和数据库通信的语言，不是某个数据库软件特有的，而是几乎所有的主流数据库软件通用的语言

##### 三、数据库存储数据的特点

- 1、数据存放到表中，然后表再放到库中
- 2、一个库中可以有多张表，每张表具有唯一的表名用来标识自己
- 3、表中有一个或多个列，列又称为“字段”，相当于java中“属性”
- 4、表中的每一行数据，相当于java中“对象”

##### 四、常见的数据库管理系统

mysql、oracle、db2、sqlserver

## MySQL的介绍

### 一、MySQL的背景

前身属于瑞典的一家公司，MySQL AB

08年被sun公司收购

09年sun被oracle收购

### 二、MySQL的优点

- 1、开源、免费、成本低
- 2、性能高、移植性也好
- 3、体积小，便于安装

### 三、MySQL的安装

属于c/s架构的软件，一般来讲安装服务端

企业版

社区版

5.5

5.6

5.7

8.0

### 四、MySQL服务的启动和停止

方式一：通过命令行

```
net start 服务名
```

```
net stop 服务名
```

方式二：计算机——右击——管理——服务

## 五、MySQL服务的登录和退出

**登录:** `mysql` 【-h 主机名 -P 端口号】 -u 用户名 -p密码

**退出:** `exit`或`ctrl+C`

## DQL语言

### 基础查询

#### 一、语法

`select` 查询列表

`from` 表名;

#### 二、特点

- 1、查询列表可以是字段、常量、表达式、函数，也可以是多个
- 2、查询结果是一个虚拟表

#### 三、示例

##### 1、查询单个字段

`select` 字段名 `from` 表名;

##### 2、查询多个字段

`select` 字段名, 字段名 `from` 表名;

##### 3、查询所有字段

`select *` `from` 表名

##### 4、查询常量

`select` 常量值;

**注意:** 字符型和日期型的常量值必须用单引号引起来，数值型不需要

#### 5、查询函数

`select 函数名(实参列表);`

#### 6、查询表达式

`select 100/1234;`

#### 7、起别名

①as

②空格

#### 8、去重

`select distinct 字段名 from 表名;`

#### 9、+

作用：做加法运算

`select 数值+数值; 直接运算`

`select 字符+数值;`先试图将字符转换成数值，如果转换成功，则继续运算；否则转换成0，再做运算

`select null+值;`结果都为null

#### 10、【补充】concat函数

功能：拼接字符

`select concat(字符1, 字符2, 字符3,...);`

#### 11、【补充】ifnull函数

功能：判断某字段或表达式是否为null，如果为null 返回指定的值，否则返回原本的值

`select ifnull(commission_pct,0) from employees;`

#### 12、【补充】isnull函数

功能：判断某字段或表达式是否为null，如果是，则返回1，否则返回0

## 条件查询

## 一、语法

select 查询列表

from 表名

where 筛选条件

## 二、筛选条件的分类

### 1、简单条件运算符

> < = <> != >= <= <=>安全等于

### 2、逻辑运算符

&& and

|| or

! not

### 3、模糊查询

like:一般搭配通配符使用，可以判断字符型或数值型

通配符：%任意多个字符，\_任意单个字符

between and

in

is null /is not null: 用于判断null值

is null PK <=>

	普通类型的数值	null值	可读性
is null	×	√	√
<=>	√	√	×

## 排序查询

## 一、语法

```
select 查询列表  
from 表  
where 筛选条件  
order by 排序列表 【asc}desc】
```

## 二、特点

- 1、asc : 升序，如果不写默认升序  
desc: 降序
- 2、排序列表 支持 单个字段、多个字段、函数、表达式、别名
- 3、order by的位置一般放在查询语句的最后（除limit语句之外）

## 常见函数

### 一、概述

功能：类似于java中的方法

好处：提高重用性和隐藏实现细节

调用：select 函数名(实参列表);

### 二、单行函数

#### 1、字符函数

concat:连接

substr:截取子串

upper:变大写

lower: 变小写

replace: 替换

length: 获取字节长度

trim:去前后空格

lpad: 左填充

rpadd: 右填充

instr: 获取子串第一次出现的索引

## 2、数学函数

ceil: 向上取整

round: 四舍五入

mod: 取模

floor: 向下取整

truncate: 截断

rand: 获取随机数, 返回0-1之间的小数

## 3、日期函数

now: 返回当前日期+时间

year: 返回年

month: 返回月

day: 返回日

date\_format: 将日期转换成字符

curdate: 返回当前日期

str\_to\_date: 将字符转换成日期

curtime: 返回当前时间

hour: 小时

minute: 分钟

second: 秒

datediff: 返回两个日期相差的天数

monthname: 以英文形式返回月

## 4、其他函数

version 当前数据库服务器的版本

database 当前打开的数据库

user当前用户

password('字符'): 返回该字符的密码形式

md5('字符'): 返回该字符的md5加密形式

## 5、流程控制函数

①if(条件表达式, 表达式1, 表达式2): 如果条件表达式成立, 返回表达式1, 否则返回表达式2

②case情况1

case 变量或表达式或字段

when 常量1 then 值1

when 常量2 then 值2

...

else 值n

end

③case情况2

case

when 条件1 then 值1

when 条件2 then 值2

...

else 值n

end

## 三、分组函数

### 1、分类

max 最大值

min 最小值



sum 和

avg 平均值

count 计算个数

## 2、特点

### ①语法

select max(字段) from 表名;

### ②支持的类型

sum和avg一般用于处理数值型

max、min、count可以处理任何数据类型

### ③以上分组函数都忽略null

### ④都可以搭配distinct使用，实现去重的统计

select sum(distinct 字段) from 表;

### ⑤count函数

count(字段): 统计该字段非空值的个数

count(\*):统计结果集的行数

案例：查询每个部门的员工个数

1 xx 10

2 dd 20

3 mm 20

4 aa 40

5 hh 40

count(1):统计结果集的行数

效率上:

MyISAM存储引擎，count(\*)最高

InnoDB存储引擎，count(\*)和count(1)效率>count(字段)

⑥ 和分组函数一同查询的字段，要求是group by后出现的字段

## 分组查询

### 一、语法

select 分组函数，分组后的字段

from 表

【where 筛选条件】

group by 分组的字段

【having 分组后的筛选】

【order by 排序列表】

### 二、特点

	使用关键字	筛选的表	位置
分组前筛选	where	原始表	group by的前面
分组后筛选	having	分组后的结果	group by 的后面

## 连接查询

### 一、含义

当查询中涉及到了多个表的字段，需要使用多表连接

select 字段1，字段2

from 表1，表2,...;

笛卡尔乘积：当查询多个表时，没有添加有效的连接条件，导致多个表所有行实现完

## 全连接

如何解决：添加有效的连接条件

## 二、分类

按年代分类：

sql92：

等值

非等值

自连接

也支持一部分外连接（用于oracle、sqlserver，mysql不支持）

sql99【推荐使用】

内连接

等值

非等值

自连接

外连接

左外

右外

全外（mysql不支持）

交叉连接

### 三、SQL92语法

#### 1、等值连接

语法：

```
select 查询列表  
from 表1 别名,表2 别名  
where 表1.key=表2.key  
【and 筛选条件】  
【group by 分组字段】  
【having 分组后的筛选】  
【order by 排序字段】
```

特点：

- ① 一般为表起别名
- ② 多表的顺序可以调换
- ③ n表连接至少需要n-1个连接条件
- ④ 等值连接的结果是多表的交集部分

#### 2、非等值连接

语法：

```
select 查询列表  
from 表1 别名,表2 别名  
where 非等值的连接条件  
【and 筛选条件】  
【group by 分组字段】  
【having 分组后的筛选】  
【order by 排序字段】
```

#### 3、自连接

语法：

```
select 查询列表  
from 表 别名1,表 别名2  
where 等值的连接条件  
【and 筛选条件】  
【group by 分组字段】  
【having 分组后的筛选】  
【order by 排序字段】
```

#### 四、SQL99语法

##### 1、内连接

语法:

```
select 查询列表  
from 表1 别名  
【inner】 join 表2 别名 on 连接条件  
where 筛选条件  
group by 分组列表  
having 分组后的筛选  
order by 排序列表  
limit 子句;
```

特点:

- ①表的顺序可以调换
- ②内连接的结果=多表的交集
- ③n表连接至少需要n-1个连接条件

分类:

等值连接

非等值连接

自连接

## 2、外连接

语法：

```
select 查询列表  
from 表1 别名  
left|right|full【outer】 join 表2 别名 on 连接条件  
where 筛选条件  
group by 分组列表  
having 分组后的筛选  
order by 排序列表  
limit 子句;
```

特点：

①查询的结果=主表中所有的行，如果从表和它匹配的将显示匹配行，如果从表没有匹配的则显示null

②left join 左边的就是主表，right join 右边的就是主表

full join 两边都是主表

③一般用于查询除了交集部分的剩余的不匹配的行

## 3、交叉连接

语法：

```
select 查询列表  
from 表1 别名  
cross join 表2 别名;
```

特点：

类似于笛卡尔乘积

## 子查询

### 一、含义

嵌套在其他语句内部的select语句称为子查询或内查询，

外面的语句可以是insert、update、delete、**select**等，一般select作为外面语句较多

外面如果为select语句，则此语句称为外查询或主查询

### 二、分类

#### 1、按出现位置

select后面：

    标量子查询

from后面：

    表子查询

where或having后面：

    标量子查询

    列子查询

    行子查询

exists后面：

    标量子查询

    列子查询

    行子查询

    表子查询

#### 2、按结果集的行列

标量子查询（单行子查询）：结果集为一行一列

列子查询（多行子查询）：结果集为多行一列

行子查询：结果集为多行多列

表子查询：结果集为多行多列

### 三、示例

where或having后面

#### 1、标量子查询

案例：查询最低工资的员工姓名和工资

##### ①最低工资

```
select min(salary) from employees
```

##### ②查询员工的姓名和工资，要求工资=①

```
select last_name, salary
from employees
where salary=(
    select min(salary) from employees
);
```

#### 2、列子查询

案例：查询所有是领导的员工姓名

##### ①查询所有员工的 manager\_id

```
select manager_id
from employees
```

##### ②查询姓名，employee\_id属于①列表的一个

```
select last_name
from employees
where employee_id in(
    select manager_id
    from employees
);
```



## 分页查询

### 一、应用场景

当要查询的条目数太多，一页显示不全

### 二、语法

select 查询列表

from 表

limit 【offset, 】 size;

注意:

offset代表的是起始的条目索引，默认从0卡死

size代表的是显示的条目数

公式:

假如要显示的页数为page，每一页条目数为size

select 查询列表

from 表

limit (page-1)\*size,size;

## 联合查询

### 一、含义

union: 合并、联合，将多次查询结果合并成一个结果

### 二、语法

查询语句1

union 【all】

查询语句2

union 【all】

...

### 三、意义

- 1、将一条比较复杂的查询语句拆分成多条语句
- 2、适用于查询多个表的时候，查询的列基本是一致

### 四、特点

- 1、要求多条查询语句的查询列数必须一致
- 2、要求多条查询语句的查询的各列类型、顺序最好一致
- 3、union 去重，union all包含重复项

## 查询总结

语法：

select 查询列表      ⑦  
from 表1 别名      ①  
连接类型 join 表2      ②  
on 连接条件      ③  
where 筛选      ④  
group by 分组列表      ⑤  
having 筛选      ⑥  
order by 排序列表      ⑧  
limit 起始条目索引， 条目数； ⑨

## DML语言

### 插入

#### 一、方式一

语法:

```
insert into 表名(字段名,...) values(值,...);
```

特点:

- 1、要求值的类型和字段的类型要一致或兼容
- 2、字段的个数和顺序不一定与原始表中的字段个数和顺序一致  
但必须保证值和字段一一对应
- 3、假如表中有可以为null的字段，注意可以通过以下两种方式插入null值
  - ①字段和值都省略
  - ②字段写上，值使用null
- 4、字段和值的个数必须一致
- 5、字段名可以省略，默认所有列

## 二、方式二

语法:

```
insert into 表名 set 字段=值, 字段=值, ...;
```

### 两种方式 的区别:

1. 方式一支持一次插入多行，语法如下:

```
insert into 表名【(字段名,...)】 values(值, ..), (值, ...), ...;
```

2. 方式二支持子查询，语法如下:

```
insert into 表名
```

查询语句;

## 修改

### 一、修改单表的记录 ★

语法: update 表名 set 字段=值, 字段=值 【where 筛选条件】;

## 二、修改多表的记录【补充】

语法:

update 表1 别名

left|right|inner join 表2 别名

on 连接条件

set 字段=值, 字段=值

【where 筛选条件】;

## 删除

方式一：使用delete

### 一、删除单表的记录★

语法: delete from 表名 【where 筛选条件】 【limit 条目数】

### 二、级联删除[补充]

语法:

delete 别名1, 别名2 from 表1 别名

inner|left|right join 表2 别名

on 连接条件

【where 筛选条件】

方式二：使用truncate

语法: truncate table 表名

### 两种方式的区别【面试题】★

1. truncate删除后，如果再插入，标识列从1开始

delete删除后，如果再插入，标识列从断点开始

2. delete可以添加筛选条件

truncate不可以添加筛选条件

3. truncate效率较高

4. truncate没有返回值

delete可以返回受影响的行数

5. truncate不可以回滚

delete可以回滚

## DDL语言

### 库的管理

#### 一、创建库

```
create database 【if not exists】 库名 【 character set 字符集名】 ;
```

#### 二、修改库

```
alter database 库名 character set 字符集名;
```

#### 三、删除库

```
drop database 【if exists】 库名;
```

### 表的管理

#### 一、创建表 ★

```
create table 【if not exists】 表名(
```

```
    字段名 字段类型 【约束】, 
```

```
    字段名 字段类型 【约束】, 
```

```
    . . .
```

```
    字段名 字段类型 【约束】
```

```
)
```

## 二、修改表

### 1. 添加列

alter table 表名 add column 列名 类型 【first|after 字段名】;

### 2. 修改列的类型或约束

alter table 表名 modify column 列名 新类型 【新约束】;

### 3. 修改列名

alter table 表名 change column 旧列名 新列名 类型;

### 4. 删除列

alter table 表名 drop column 列名;

### 5. 修改表名

alter table 表名 rename 【to】 新表名;

## 三、删除表

drop table 【if exists】 表名;

## 四、复制表

### 1、复制表的结构

create table 表名 like 旧表;

### 2、复制表的结构+数据

create table 表名

select 查询列表 from 旧表 【where 筛选】;

## 数据类型

### 一、数值型

#### 1、整型

tinyint、smallint、mediumint、int/integer、bigint

1            2            3            4            8

特点:

- ①都可以设置无符号和有符号，默认有符号，通过unsigned设置无符号
- ②如果超出了范围，会报out or range异常，插入临界值
- ③长度可以不指定，默认会有一个长度

长度代表显示的最大宽度，如果不够则左边用0填充，但需要搭配zerofill，并且默认变为无符号整型

## 2、浮点型

定点数: decimal (M, D)

浮点数:

float (M, D)    4

double (M, D)   8

特点:

- ①M代表整数部位+小数部位的个数，D代表小数部位
- ②如果超出范围，则报out or range异常，并且插入临界值
- ③M和D都可以省略，但对于定点数，M默认为10，D默认为0
- ④如果精度要求较高，则优先考虑使用定点数

## 二、字符型

char、varchar、binary、varbinary、enum、set、text、blob

char: 固定长度的字符，写法为char (M)，最大长度不能超过M，其中M可以省略，默认为1

varchar: 可变长度的字符，写法为varchar (M)，最大长度不能超过M，其中M不可以省略

## 三、日期型

year年

date日期

time时间

datetime 日期+时间 8

timestamp 日期+时间 4 比较容易受时区、语法模式、版本的影响，更能反映当前时区的真实时间

## 常见的约束

### 一、常见的约束

NOT NULL：非空，该字段的值必填

UNIQUE：唯一，该字段的值不可重复

DEFAULT：默认，该字段的值不用手动插入有默认值

CHECK：检查，mysql不支持

PRIMARY KEY：主键，该字段的值不可重复并且非空 unique+not null

FOREIGN KEY：外键，该字段的值引用了另外的表的字段

### 主键和唯一

#### 1、区别：

- ①、一个表至多有一个主键，但可以有多个唯一
- ②、主键不允许为空，唯一可以为空

#### 2、相同点

都具有唯一性

都支持组合键，但不推荐

### 外键：

- 1、用于限制两个表的关系，从表的字段值引用了主表的某字段值
- 2、外键列和主表的被引用列要求类型一致，意义一样，名称无要求
- 3、主表的被引用列要求是一个key（一般就是主键）
- 4、插入数据，先插入主表
- 删除数据，先删除从表



可以通过以下两种方式来删除主表的记录

#方式一：级联删除

```
ALTER TABLE stuinfo ADD CONSTRAINT fk_stu_major FOREIGN KEY(majorid)
REFERENCES major(id) ON DELETE CASCADE;
```

#方式二：级联置空

```
ALTER TABLE stuinfo ADD CONSTRAINT fk_stu_major FOREIGN KEY(majorid)
REFERENCES major(id) ON DELETE SET NULL;
```

## 二、创建表时添加约束

create table 表名(

    字段名 字段类型 not null, #非空

    字段名 字段类型 primary key, #主键

    字段名 字段类型 unique, #唯一

    字段名 字段类型 default 值, #默认

    constraint 约束名 foreign key(字段名) references 主表(被引用列)

)

注意：

	支持类型	可以起约束名
列级约束	除了外键	不可以
表级约束	除了非空和默认	可以，但对主键无效

列级约束可以在一个字段上追加多个，中间用空格隔开，没有顺序要求

## 三、修改表时添加或删除约束

### 1、非空

添加非空

```
alter table 表名 modify column 字段名 字段类型 not null;
```

删除非空

```
alter table 表名 modify column 字段名 字段类型 ;
```

## 2、默认

添加默认

```
alter table 表名 modify column 字段名 字段类型 default 值;
```

删除默认

```
alter table 表名 modify column 字段名 字段类型 ;
```

## 3、主键

添加主键

```
alter table 表名 add 【 constraint 约束名】 primary key(字段名);
```

删除主键

```
alter table 表名 drop primary key;
```

## 4、唯一

添加唯一

```
alter table 表名 add 【 constraint 约束名】 unique(字段名);
```

删除唯一

```
alter table 表名 drop index 索引名;
```

## 5、外键

添加外键

```
alter table 表名 add 【 constraint 约束名】 foreign key(字段名) references  
主表（被引用列）;
```

删除外键

```
alter table 表名 drop foreign key 约束名;
```

## 四、自增长列

特点:

1、不用手动插入值，可以自动提供序列值，默认从1开始，步长为1

```
auto_increment_increment
```

如果要更改起始值：手动插入值

如果要更改步长：更改系统变量

```
set auto_increment_increment=值;
```

2、一个表至多有一个自增长列

3、自增长列只能支持数值型

4、自增长列必须为一个key

### 一、创建表时设置自增长列

```
create table 表(  
    字段名 字段类型 约束 auto_increment  
)
```

### 二、修改表时设置自增长列

```
alter table 表 modify column 字段名 字段类型 约束 auto_increment
```

### 三、删除自增长列

```
alter table 表 modify column 字段名 字段类型 约束
```

## TCL语言

## 事务

### 一、含义

事务：一条或多条sql语句组成一个执行单位，一组sql语句要么都执行要么都不执行

### 二、特点（ACID）

A 原子性：一个事务是不可再分割的整体，要么都执行要么都不执行

C 一致性：一个事务可以使数据从一个一致状态切换到另外一个一致的状态

I 隔离性：一个事务不受其他事务的干扰，多个事务互相隔离的

D 持久性：一个事务一旦提交了，则永久的持久化到本地

### 三、事务的使用步骤 ★

了解：

隐式（自动）事务：没有明显的开启和结束，本身就是一条事务可以自动提交，比如 insert、update、delete

显式事务：具有明显的开启和结束

使用显式事务：

①开启事务

set autocommit=0;

start transaction;#可以省略

②编写一组逻辑sql语句

注意：sql语句支持的是insert、update、delete

设置回滚点：

savepoint 回滚点名;

③结束事务

提交：commit;

回滚：rollback;

回滚到指定的地方：rollback to 回滚点名;

#### 四、并发事务

##### 1、事务的并发问题是如何发生的？

多个事务 同时 操作 同一个数据库的相同数据时

##### 2、并发问题都有哪些？

脏读：一个事务读取了其他事务还没有提交的数据，读到的是其他事务“更新”的数据

不可重复读：一个事务多次读取，结果不一样

幻读：一个事务读取了其他事务还没有提交的数据，只是读到的是 其他事务“插入”的数据

##### 3、如何解决并发问题

通过设置隔离级别来解决并发问题

##### 4、隔离级别

	脏读	不可重复读	幻读
read uncommitted: 读未提交	×	×	×
read committed: 读已提交	✓	×	×
repeatable read: 可重复读	✓	✓	×
serializable: 串行化	✓	✓	✓

## 其他

## 视图

### 一、含义

mysql5.1版本出现的新特性，本身是一个虚拟表，它的数据来自于表，通过执行时动态生成。

好处：

- 1、简化sql语句
- 2、提高了sql的重用性
- 3、保护基表的数据，提高了安全性

### 二、创建

```
create view 视图名  
as  
查询语句;
```

### 三、修改

方式一：

```
create or replace view 视图名  
as  
查询语句;
```

方式二：

```
alter view 视图名
```

```
as
```

```
查询语句
```

#### 四、删除

```
drop view 视图1, 视图2,...;
```

#### 五、查看

```
desc 视图名;
```

```
show create view 视图名;
```

#### 六、使用

##### 1. 插入

```
insert
```

##### 2. 修改

```
update
```

##### 3. 删除

```
delete
```

##### 4. 查看

```
select
```

**注意：**视图一般用于查询的，而不是更新的，所以具备以下特点的视图都不允许更新

①包含分组函数、group by、distinct、having、union、

②join

③常量视图

④where后的子查询用到了from中的表

⑤用到了不可更新的视图

#### 七、视图和表的对比

	关键字	是否占用物理空间	使用
视图	view	占用较小，只保存sql逻辑	一般用于查询
表	table	保存实际的数据	增删改查

## 变量

### 分类

#### 一、系统变量

**说明：**变量由系统提供的，不用自定义

语法：

##### ①查看系统变量

show 【global|session】variables like ''；如果没有显式声明global还是session，则默认是session

##### ②查看指定的系统变量的值

select @@【global|session】.变量名；如果没有显式声明global还是session，则默认是session

##### ③为系统变量赋值

方式一：

set 【global|session】 变量名=值；如果没有显式声明global还是session，则默认是session

方式二：

set @@global.变量名=值；

set @@变量名=值；

#### 1、全局变量

服务器层面上的，必须拥有super权限才能为系统变量赋值，作用域为整个服务器，也就是针对于所有连接（会话）有效

#### 2、会话变量

服务器为每一个连接的客户端都提供了系统变量，作用域为当前的连接（会话）

## 二、自定义变量

说明：

### 1、用户变量

作用域：针对于当前连接（会话）生效

位置：begin end里面，也可以放在外面

使用：

#### ①声明并赋值：

set @变量名=值;或

set @变量名:=值;或

select @变量名:=值;

#### ②更新值

方式一：

set @变量名=值;或

set @变量名:=值;或

select @变量名:=值;

方式二：

select xx into @变量名 from 表;

#### ③使用

select @变量名;

### 2、局部变量

作用域：仅仅在定义它的begin end中有效



**位置：**只能放在begin end中，而且只能放在第一句

**使用：**

①声明

declare 变量名 类型 【default 值】;

②赋值或更新

方式一：

set 变量名=值;或

set 变量名:=值;或

select @变量名:=值;

方式二：

select xx into 变量名 from 表;

③使用

select 变量名;

## 存储过程和函数

**说明：**都类似于java中的方法，将一组完成特定功能的逻辑语句包装起来，对外暴露名字

**好处：**

- 1、提高重用性
- 2、sql语句简单
- 3、减少了和数据库服务器连接的次数，提高了效率

## 存储过程

### 一、创建 ★

create procedure 存储过程名(参数模式 参数名 参数类型)

begin

存储过程体

end

注意:

1. 参数模式: in、out、inout, 其中in可以省略
2. 存储过程体的每一条sql语句都需要用分号结尾

## 二、调用

call 存储过程名(实参列表)

举例:

调用in模式的参数: call sp1 ( '值' );

调用out模式的参数: set @name; call sp1(@name);select @name;

调用inout模式的参数: set @name=值; call sp1(@name); select @name;

## 三、查看

show create procedure 存储过程名;

## 四、删除

drop procedure 存储过程名;

# 函数

## 一、创建

create function 函数名(参数名 参数类型) returns 返回类型

begin

    函数体

end

注意: 函数体中肯定需要有return语句

## 二、调用

select 函数名(实参列表);

## 三、查看

show create function 函数名;

#### 四、删除

drop function 函数名;

### 流程控制结构

说明:

顺序结构: 程序从上往下依次执行

分支结构: 程序按条件进行选择执行, 从两条或多条路径中选择一条执行

循环结构: 程序满足一定条件下, 重复执行一组语句

### 分支结构

特点:

#### 1、if函数

功能: 实现简单双分支

语法:

if(条件, 值1, 值2)

位置:

可以作为表达式放在任何位置

#### 2、case结构

功能: 实现多分支

语法1:

case 表达式或字段

when 值1 then 语句1;

when 值2 then 语句2;

..

else 语句n;

end [case];

**语法2:**

```
case  
when 条件1 then 语句1;  
when 条件2 then 语句2;  
..  
else 语句n;  
end [case];
```

**位置:**

可以放在任何位置,

如果放在begin end 外面, 作为表达式结合着其他语句使用

如果放在begin end 里面, 一般作为独立的语句使用

### 3、if结构

功能: 实现多分支

语法:

```
if 条件1 then 语句1;  
elseif 条件2 then 语句2;  
...  
else 语句n;  
end if;
```

位置:

只能放在begin end中

## 循环结构

**位置:**

只能放在begin end中

特点：都能实现循环结构

对比：

①这三种循环都可以省略名称，但如果循环中添加了循环控制语句（leave或iterate）则必须添加名称

②

loop 一般用于实现简单的死循环

while 先判断后执行

repeat 先执行后判断，无条件至少执行一次

### 1、while

语法：

```
【名称：】while 循环条件 do
    循环体
end while 【名称】；
```

### 2、loop

语法：

```
【名称：】loop
    循环体
end loop 【名称】；
```

### 3、repeat

语法：

```
【名称：】repeat
    循环体
until 结束条件
end repeat 【名称】；
```

## 二、循环控制语句

**leave:** 类似于break，用于跳出所在的循环

**iterate:** 类似于continue，用于结束本次循环，继续下一次