MySQL复习资料

**使用数据库保存数据的三大原因：**

1.大批量读取和操作数据效率高；

2.数据共享方便；

3.数据损坏不易丢失，易修复。

**数据定义语言：DDL （definition：定义） (常用)数据操纵语言：DML （manipulation：操纵）/ 数据控制语言：DCL （control：控制）。**

**Mysql常用命令：**

net start mysql ——启动MySQL

net stop mysql ——停止MySQL

mysql -u 用户名 -p ——登录MySQL

mysql -b IP地址 -u -P 3306——登录MySQL

\c:清除命令

\q:退出MySQL

help：帮助命令

\s：查看当前信息

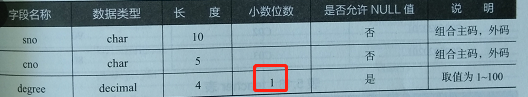
Client：客户端

**注意：**

分号表示语句的结束

数据库不能保存数据，只有表能保存数据

主键：唯一确定一条记录的字段或者字段组合，主键值在表中不能为空且须唯一。一张合格的表必须有主键，且只能有一个主键（注意：不是指只能有一个字段充当主键）



这个小数位数怎么设置

小数位是括号四，然后逗号1, （4，1）

说明主码就是设置主键

设置主键在后面加primary key

默认为空，主键默认为非空

**BLOB类型的字段用于存储二进制数据**

MySQL中，BLOB是个类型系列，包括：TinyBlob、Blob、MediumBlob、LongBlob，这几个类型之间的唯一区别是在存储文件的最大大小上不同。

MySQL的四种BLOB类型  
类型 大小(单位：字节)  
TinyBlob 最大 255  
Blob 最大 65K  
MediumBlob 最大 16M  
LongBlob 最大 4G

# [mysql中tinyint、smallint、int、bigint的区别](https://www.cnblogs.com/baizhanshi/p/8482068.html)

**tinyint**

从 -2^7 (-128) 到 2^7 - 1 (123) 的整型数据。存储大小为 1 个字节。

unsigned 是从 0 到 255 的整型数据。

所以建表的时候 只能是tinyint(3),哪怕你建tinyint(100)，他最大还是3位这么多。

**smallint**

从 -2^15 (-32,768) 到 2^15 - 1 (32,767) 的整型数据。存储大小为 2 个字节。

unsigned 是从 0 到 65535 的整型数据。

所以建表的时候 只能是smallint(5),哪怕你建smallint(100)，他最大还是5位这么多。

**int**

从 -2^31 (-2,147,483,648) 到 2^31 - 1 (2,147,483,647) 的整型数据（所有数字）.存储大小为 4 个字节。

unsigned 是从 0 到 4294967296 的整型数据。

所以建表的时候 只能是int(10),哪怕你建int(100)，他最大还是10位这么多。

**bigint**

从 -2^63 (-9,223,372,036,854,775,808) 到 2^63-1 (9,223,372,036,854,775,807) 的整型数据（所有数字）。存储大小为 8 个字节。

555是(自己算吧)

所以建表的时候 只能是bigint(20),哪怕你建bigint(100)，他最大还是20位这么多.

**mysql数据类型**

**数值型：**

**整数类型**

**int （4字节）**

**smallint（2字节）**

**bigint（8字节）**

**浮点数类型**

**定点数类型**

**float**

**double**

**浮点数会有误差**

**日期与时间类型**

**day**

**3字节**

**1000-01-01~9999-12-31**

**daytime**

**8字节**

**1000-01-01 00:00:00:00~9999-12-31**

**字符串类型：**

**char：character 固定长度的字符串**

**varchar：variable character 可变长度的字符串**

**char（5）：输入'123456’->'12345’ 输入'123’->' 123’**

**在数据库中，除了数值型数据外，字符串和日期型数据都要放在一对单引号中**

**varchar（5）：5表示最大长度**

**输入'123456’->'12345’ 输入'123’->'123’**

**表的结构由字段组成，在显示表内容时，每列叫做字段（field），每一行具体数据叫做记录（record），设计表，就是要确定这张表有多少列，每列的数据类型是什么。**

一、操作数据库

**1.创建数据库**

**create table employee**

**(**

**id int,**

**ename varchar(20), ——姓名**

**gender char(1), ——性别**

**birthday date, ——生日**

**entry\_date date, ——工龄**

**job varchar(50), ——岗位**

**salary double, ——工资**

**resume text ——简介**

**);**

**2.查看数据库**

查看表结构的命令：DESCRIBE 表名 ——desc表示描述

查询当前数据库中所有表：show tables

查看创建表的语句：show create table 表名

查看表的结构：desc book

**3.修改数据库**

alter table 表名

修改选项的语法格式如下：

## { ADD COLUMN <列名> <类型> 添加字段

## ADD COLUMN <字段> <类型> first 第一列/after <字段> 在<字段>后面

## 例：

使用 ALTER TABLE 修改表 tb\_emp1 的结构，在表的第一列添加一个 int 类型的字段 col1，输入的 SQL 语句和运行结果如下所示。

mysql> ALTER TABLE tb\_emp1

-> ADD COLUMN col1 INT FIRST;

Query OK, 0 rows affected (0.94 sec)

Records: 0 Duplicates: 0 Warnings: 0

mysql> DESC tb\_emp1;

+--------+-------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+--------+-------------+------+-----+---------+-------+

| col1 | int(11) | YES | | NULL | |

| id | int(11) | YES | | NULL | |

| name | varchar(25) | YES | | NULL | |

| deptId | int(11) | YES | | NULL | |

| salary | float | YES | | NULL | |

+--------+-------------+------+-----+---------+-------+

5 rows in set (0.00 sec)

## 例：

## 使用 ALTER TABLE 修改表 tb\_emp1 的结构，在一列 name 后添加一个 int 类型的字段 col2，输入的 SQL 语句和运行结果如下所

mysql> ALTER TABLE tb\_emp1

-> ADD COLUMN col2 INT AFTER name;

Query OK, 0 rows affected (0.50 sec)

Records: 0 Duplicates: 0 Warnings: 0

mysql> DESC tb\_emp1;

+--------+-------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+--------+-------------+------+-----+---------+-------+

| col1 | int(11) | YES | | NULL | |

| id | int(11) | YES | | NULL | |

| name | varchar(25) | YES | | NULL | |

| col2 | int(11) | YES | | NULL | |

| deptId | int(11) | YES | | NULL | |

| salary | float | YES | | NULL | |

+--------+-------------+------+-----+---------+-------+

6 rows in set (0.00 sec)  
| CHANGE COLUMN <旧列名> <新列名> <新列类型> 修改字段名称

使用 ALTER TABLE 修改表 tb\_emp1 的结构，将 col1 字段名称改为 col3，同时将数据类型变为 CHAR(30)，输入的 SQL 语句和运行结果如下所示。

mysql> ALTER TABLE tb\_emp1

-> CHANGE col1 col3 CHAR(30);

Query OK, 0 rows affected (0.76 sec)

Records: 0 Duplicates: 0 Warnings: 0

mysql> DESC tb\_emp1;

+--------+-------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+--------+-------------+------+-----+---------+-------+

| col3 | char(30) | YES | | NULL | |

| id | int(11) | YES | | NULL | |

| name | varchar(30) | YES | | NULL | |

| deptId | int(11) | YES | | NULL | |

| salary | float | YES | | NULL | |

+--------+-------------+------+-----+---------+-------+

5 rows in set (0.01 sec)

| ALTER COLUMN <列名> { SET DEFAULT <默认值> | DROP DEFAULT }   
| MODIFY COLUMN <列名> <类型> 修改字段数据类型

mysql> ALTER TABLE tb\_emp1

-> MODIFY name VARCHAR(30);

Query OK, 0 rows affected (0.15 sec)

Records: 0 Duplicates: 0 Warnings: 0

mysql> DESC tb\_emp1;

+--------+-------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+--------+-------------+------+-----+---------+-------+

| col1 | int(11) | YES | | NULL | |

| id | int(11) | YES | | NULL | |

| name | varchar(30) | YES | | NULL | |

| col2 | int(11) | YES | | NULL | |

| deptId | int(11) | YES | | NULL | |

| salary | float | YES | | NULL | |

+--------+-------------+------+-----+---------+-------+

6 rows in set (0.00 sec)

使用 ALTER TABLE 修改表 tb\_emp1 的结构，将 name 字段的数据类型由 VARCHAR(22) 修改成 VARCHAR(30)，输入的 SQL 语句和运行结果如下所示。  
| DROP COLUMN <列名> 删除字段

使用 ALTER TABLE 修改表 tb\_emp1 的结构，删除 col2 字段，输入的 SQL 语句和运行结果如下所示。

mysql> ALTER TABLE tb\_emp1

-> DROP col2;

Query OK, 0 rows affected (0.53 sec)

Records: 0 Duplicates: 0 Warnings: 0

mysql> DESC tb\_emp1;

+--------+-------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+--------+-------------+------+-----+---------+-------+

| col1 | int(11) | YES | | NULL | |

| id | int(11) | YES | | NULL | |

| name | varchar(30) | YES | | NULL | |

| deptId | int(11) | YES | | NULL | |

| salary | float | YES | | NULL | |

+--------+-------------+------+-----+---------+-------+

5 rows in set (0.00 sec)

| RENAME TO <新表名> } 修改表名

使用 ALTER TABLE 将数据表 tb\_emp1 改名为 tb\_emp2，输入的 SQL 语句和运行结果如下所示。

mysql> ALTER TABLE tb\_emp1

-> RENAME TO tb\_emp2;

mysql> SHOW TABLES;

+--------------------+

| Tables\_in\_test\_db |

+--------------------+

| tb\_emp2 |

+--------------------+

1 rows in set (0.00 sec)

**4.删除数据库**

drop database 数据库名

1. 选择数据库

Use 数据库名

二、表

（1）操作表记录

（增删改查，Create Read Update Delete）

1.增加表数据

**insert into 表名[(字段列表)]**

**values(值列表)**

中括号中的字段列表可以省略，省略之后，值列表必须全部给值，并且值的顺序和表结构一一对应。

例：

insert into student(sno,sname,ssex)

values('10000000001','张三',null，null，null)

2.修改表数据

**Update 表名**

**set 字段名=新值，字段名=新值**

**[where 条件表达式]**

例：把张飞的生日改成1999-1-1，入职日期改成2019-1-1

update employee

set birthday='1999-1-1',

entry\_date='2019-1-1'

where ename='张飞'（执行此句，表示只修改一部分的数值）3

由于薪水的不合理，我们如下调整，张飞工资加一万，刘备工资减一万

[注意：set子句的字段指的是同一行]

update employee

set salary=salary+10000

where ename='张飞'

update employee

set salary=salary-10000

where ename='刘备'

练习：2000年以后工资减1000，2000年以前工资加1000

**UPDATE employee22**

**set salary=salary-1000**

**WHERE entry\_date>='2000-1-1'**

**UPDATE employee**

**set salary=salary+1000**

**WHERE entry\_date<'2000-1-1'**

**删除表：**

**Delete from employee where ename='张飞'——删除一条记录**

**delete from employee； ——删除表中所有记录**

三．查询

查询分为单表查询和多表查询

**Select \*|字段列表 from 表名 //对列（垂直方向）**

**Where 条件表达式 //对行（水平方向）进行筛选**

**|表示’或’**

查询整张表：**select \* from 表名**

**例：**

查找所有学生的姓名和英语的数据：**select sname,english from exam**

查询英语数据，去除重复值：**select distinct english from exam（Distinct：去除结果表中的重复记录）**

每科加10分：**select sname,math+10,english+10,chinese+10 from exam**

查询学生三科的总分：**select sname,math+english+chinese from exam**

输出学生表中的前2条记录：**select \* from student LIMIT 2**

输出学生表中第二条记录后的五条记录：**select \* from student limit 2,5**

**curdate():当前日期->今天的日期**

**例：SELECT stuid,stuname,YEAR(CURDATE())-YEAR(birthday) age from student**

**计算年龄，并更改别名为age**

**在查询结果中，用表达式计算出列，叫做计算列，为增强可读性，一般都要为它们设置别名**

**运算符：<>：不等于号；!=：不等于号；!<：不小于；!>：不大于。**

**between运算符：**

**between 低值 and 高值：介于低值和高值之间（含高低值）**

**例：**

**SELECT \* from exam**

**WHERE score BETWEEN 70 and 90;**

**||**

**WHERE score >=70 and score <90**

**例：**

**SELECT \* from exam**

**WHERE score >70 AND score <=80**

**查看大于70，小于等于80的成绩记录，不能用between，条件不同**

**SELECT stuid,stuname,sex,clsname from student**

**WHERE birthday BETWEEN '1990-01-01' and '1999-12-31'**

**另一种条件表达式的写法：**

**where year（birthday） between 1990 and 1999**

**Or运算符：**

**例：**

**查找100,95,90分的成绩记录**

**SELECT \* from exam**

**WHERE score=100 or score=95 or score=90**

**||**

**WHERE score in(100,95,90)**

**like运算符：模糊查询，必须配合通配符使用，分别是%和\_**

**%表示零个或任意多个字符，\_表示1个字符**

**例一：查找信息班所有学生的记录**

**SELECT \* FROM student**

**WHERE clsname LIKE '信息%'**

**例二：查找姓李名字两个字的学生的记录**

**SELECT \* FROM student**

**WHERE stuname LIKE '李\_'**

**例三：查找姓李名字三个字的学生的记录**

**SELECT \* FROM student**

**WHERE stuname LIKE '李\_\_'**

**空值查询：null NULL**

**空值查询：null 空值代表没有值，不存在值，他和空字符串''不同例：**

**SELECT \* FROM student**

**WHERE address is null（注意：null，不能用等于，要用is）**

**聚合函数：**

1. **count()聚合函数，用来统计行数**

**例：**

**查找女同学**

**SELECT count(\*) FROM student**

**WHERE sex='女'**

1. **第二类聚合函数：sum avg max min，这些函数的参数必须是数值型**

**SUM 计算一列中值的总和**

**AVG 计算一列中值的平均值**

**MAX 计算一列中值的最大值**

**MIN 计算一列中值的最小值**

**例：统计单位所有职工的基本工资总和**

**SELECT SUM(base\_sal) 单位总基本工资 FROM emp**

**例：统计单位平均的基本工资**

**SELECT avg(base\_sal) 平均基本工资 FROM emp**

**例：统计最高基本工资和最低基本工资**

**SELECT max(base\_sal) 最高基本工资,min(base\_sal) 最低基本工资 FROM emp**

**【注意】当统计的记录完全相同时，不同的统计结果可以写在同一句中**

**Group By：分组**

**例：统计每个班的学生人数**

**SELECT clsname,count(\*) 人数**

**FROM student**

**GROUP BY clsname**

**having子句：对分组后的记录进行二次筛选，它和where的区别是，where是对表的原始记录进行初次筛选**

**例：查找超过1个人的班级，显示班级名称和该班人数**

**SELECT clsname,count(\*) 人数 FROM student**

**GROUP BY clsname**

**HAVING count(\*) > 1 and clsname not LIKE '信息%'**

**order by子句：用于对最终结果进行排序，他只影响记录的显示顺序，并不影响记录结果，增加第二，第三排序字段，要在后面加分号**

**降序使用desc，升序不用**

**limit：相当于第三次筛选**

**limit m：从结果中取前m条记录**

**limit n,m:跳过结果中的前n条记录，取之后的m条记录（LIMIT M,3**

**表示从第M+1行开始，结果不包含第M行）**

**例一：查找C001这门课考第二名的同学，显示他的学号和分数**

**SELECT stuid,score FROM exam**

**WHERE cid ='C001'**

**ORDER BY score desc LIMIT 1,1**

**例二：找出年龄最大的两位学生，显示所有字段**

**SELECT \* FROM student**

**WHERE birthday is not null**

**ORDER BY birthday,stuid**

**LIMIT 2**

**语句格式：**

1. **select..from**
2. **where…**
3. **group by… (分组)**
4. **having… (对查询后的表进行二次筛选)**
5. **order by… +desc降序(升序和降序)**
6. **limit… (限定输出结果)**

**多表连接查询：在多表连接查询的最初结果中，是全部一一匹配的笛卡尔积，不合逻辑的错误记录很多，为了只留下正确匹配的记录，必须加上筛选条件，表达式为：外表.外键=主表.主键，这个条件可以认为是语法格式要求，它和剩下的逻辑条件之间用and运算符连接，如果用逻辑条件有多个表达式，要把所有逻辑条件用括号括起来**

**以上的连接查询也叫内连接**

**例：显示员工编号，姓名，入职日期，所在部门名称**

**SELECT empid,ename,emp.deptid,dept.deptid,dname FROM emp,dept**

**WHERE emp.deptid=dept.deptid**

**多表连接查询的三个要点：1.连接条件（外表.外键=主表.主键） 2.and 3.括号**

**产生笛卡尔积的查询也叫交叉连接，使用了连接条件产生的正确结果的查询也叫内连接**

**例一：找出业务部的李姓员工，显示工号，姓名，部门号，部门名称**

**SELECT empid,ename,emp.deptid,dname FROM emp,dept**

**WHERE emp.deptid=dept.deptid and (ename LIKE '李%' and dname ='业务部')**

**例二：找出在五楼办公的员工，显示工号、姓名、工作、部门号、部门名称、办公位置字段**

**SELECT empid,ename,job,emp.deptid,dname,address FROM emp,dept**

**WHERE emp.deptid=dept.deptid and (address='五楼')**

**表之间的记录对应关系可分为三类：**

**对于1对多：我们可以在多的一方设计外键保存为一的一方的主键**

**对于1对1：我们可以在任意一方设计外键，保存另一方的主键作为外键**

**对于多对多：新建一张第三方关系表，保存两张表的主键作为外键，存储两张表的主键主键之间的对应关系，来保存两张表之间的关系**

**主键和外键的区别**

**（1）主键是唯一标识一条记录，不能有重复，不允许为空；而外键可以重复，可以是空值**

**（2）主键是用来保持数据完整性，外键是用来建立与其他表联系用的**

**（3）主键只有一个，外键可以有多个**

**外键：放在多表那一边，可以明确的声明表示表和表之间关系的字段的参照关系，使数据库帮我们维护这种关系，这种键就叫做在表和表之间声明了一个外键（某个表的某个字段是从外面得来的）**

**表的参照完整性：由外键实现参照完整性的对象**

**建表的同时建外键：**

**CREATE TABLE dept**

**(**

**deptid char(3) PRIMARY KEY,**

**dname VARCHAR(10)**

**)**

**CREATE TABLE emp**

**(**

**empid char(3) PRIMARY KEY,**

**ename VARCHAR(10),**

**deptid char(3),**

**FOREIGN KEY(deptid) REFERENCES dept(deptid)**

**)**

**DELETE FROM emp WHERE deptid='D03'**

**DELETE FROM dept WHERE deptid='D03'**

**先建表后建外键：**

**ALTER TABLE emp**

**ADD CONSTRAINT fk\_deptid**

**FOREIGN KEY (deptid)**

**REFERENCES dept(deptid)**

**【注意】在多表查询时，如果有两张以上的表，则连接条件要写多个，一般来讲，如果N个表连接查询，最少要有N-1个连接条件，这些连接条件之间，必须用and运算符连接**

**例：查询信息51班网络营销课程的成绩，显示学号、姓名、课程号、课程名、成绩**

**SELECT exam.stuid,stuname,exam.cid,cname,score FROM student,course,exam**

**WHERE exam.stuid=student.stuid and exam.cid=course.cid and (clsname='信息51' and cname='网络营销')**

**例：查询有不及格科目的学生的学号、姓名、课程号、课程名及成绩**

**SELECT exam.stuid,stuname,exam.cid,cname,score FROM student,course,exam**

**WHERE (exam.stuid=student.stuid and exam.cid=course.cid) and score < 60**

**外连接：根据主表的前后位置不同，分为左外和右外连接，所谓外连接中的主表，就是不管连接的其他表有没有对应记录，主表中的记录全部都显示，主外就是主表在左，右外就是主表在右**

**例：**

**CREATE TABLE family**

**(**

**sname VARCHAR(20),**

**fname VARCHAR(20)**

**)**

**SELECT 左.sname 孙子,右.fname 爷爷 FROM family 左,family 右**

**WHERE 左.fname=右.sname**

**例：显示所有学生的学号、姓名、年龄、性别、班级名，如果有选修课程成绩的学生，接着显示他们的课程号、成绩**

**SELECT student.stuid,stuname,year(CURDATE())-year(birthday)年龄，sex,clsname,cid,score FROM student**

**left outer JOIN exam**

**on exam.stuid=student.stuid**

**<any等效于 <MAX**

**<all等效于 <MIN**

**>any等效于 >MIN**

**>all等效于 >MAX**

**!=any或<>any不等于子查询结果中的某个值；**

**!=all或<>all不等于子查询结果中的任何一个值；**

**在相关子查询中，子查询的执行依赖于外部查询，即子查询的查询条件依赖于外部查询的某个属性值。**

**带有exists关键字的查询**

**他的执行方式是查看exists后的子查询有无结果，如果有，外层父查询才返回结果，相当于外层查询有where true的条件**

**例：**

**SELECT stuname,year(CURDATE())-year(birthday) age from student**

**WHERE YEAR(CURDATE())-year(birthday) < (**

**SELECT max(year(CURDATE())-year(birthday)) FROM student**

**WHERE clsname='信息51'**

**) and clsname <> '信息51'**

**例：查询月薪有5000元以上的员工的部门，显示部门号和部门名**

**SELECT deptid,dname FROM dept**

**WHERE exists(SELECT \* FROM emp WHERE base\_sal+commision>5000 and deptid=dept.deptid)**

**以上写法，在子查询中，引用了父查询的当前行的值，这种相关联的子查询**

**索引一旦创建，将由数据库自动管理与维护。**

**书本172**

**索引类型：普通索引（可重复，可有空值）和唯一索引（不可重复，可有空值）**

**单列索引和组合索引**

**索引的设计原则：**

**1.索引并非越多越好**

**2.避免对经常更新的表建立过多的索引**

**3.数据量少的表最好不要使用索引**

**4.在不同值上的列上不要建立索引**

**5.指定唯一索引是由某种数据本身的特征**

**6.为经常需要排序、分组和联合操作的字段建立索引**

-- 第二种创建索引

ALTER TABLE student2

add INDEX stuname2(stuname)

-- 第三种创建索引

CREATE INDEX i\_stuname

on student2(stuname)

**视图是从一个或者几个基本表或视图中导出的虚拟表**

**在定义一个视图中，只是把其定义存放在数据库中，并不直接存储视图对应的数据，直到用户使用视图时，才查找对应的数据**

**视图是一张假的表，它的数据来自于真实表，往往把可以给某些用户看的数据做成视图，给这些用户授权使用视图而不是表，达到只让某些用户看部分数据的目的，视图还可以简化查询。**

**练习：查找学生成绩，显示学号、姓名、班级、课程编号、课程名、分数**

**CREATE VIEW v\_exam**

**AS**

**SELECT student.stuid,stuname,clsname,course.cid,cname,score FROM student,course,exam**

**WHERE student.stuid=exam.stuid and course.cid=exam.cid**

**SELECT \* FROM v\_exam**

**结构化查询语言（Structred Query Language,SQL）**

**常量：称为文字值或标量值，是指程序运行中值始终不改的值**

**变量：由变量名和变量值构成，其类型与常量一样，变量名不能与命令和函数名相同**

**存在3种类型的变量：系统变量、用户变量和局部变量**

**其系统变量又分为全局（global）变量和会话（session）变量两种**

**用户变量：用户可以在表达式中使用自己定义的变量，这样的变量叫做用户变量**

**用户可以先在用户变量中保护值，在引用它，这样可以将值从一个语句传递到另一个语句。用户变量在使用前必须定义和初始化。如果使用没有初始化的值，则它的值为null，为空。**

**一个客户端定义的变量不能被其他客户端看到或使用，当客户端退出时，该客户端连接的所有变量将自动释放。用户变量被引用时要在其名称前加上标志@。**

**例：**

**set @user1=1,@user2=2,@user3=3;**

**SELECT @user1,@user2,@user3**

**SELECT @user:=11,@user2:=22,@user3:=33;**

**if语句：**

**例：通过存储过程的输入参数得到学生姓名和课程名。判断学生的成绩等级：不及格，及格，良好，优秀**

**CREATE PROCEDURE p4(in sn VARCHAR(20),in cn VARCHAR(30))**

**BEGIN**

**DECLARE sc int;**

**set sc = (**

**SELECT score FROM student,course,exam**

**WHERE (student.stuid=exam.stuid and course.cid=exam.cid) and stuname=sn and cname=cn);**

**if sc is null THEN**

**SELECT '查无此成绩' as '查询结果';**

**ELSEIF sc < 60 THEN**

**SELECT concat(sn,cn,'考了不及格') as '考试等级';**

**ELSEIF sc >= 60 and sc < 80 THEN**

**SELECT concat(sn,cn,'考了及格') as '考试等级';**

**ELSEIF sc >= 80 and sc < 90 THEN**

**SELECT concat(sn,cn,'考了良好') as '考试等级';**

**ELSE**

**SELECT concat(sn,cn,'考了优秀') as '考试等级';**

**end if;**

**END;**

**调用函数：use qq**

**mysql> use qq**

**Database changed**

**mysql> call p4('张宇蛟','C语言');**

**循环语句：**

**while... end while**

**Boolean\_expression:后面接表达式**

**LEAVE begin\_label:用于从while循环中退出，将执行出现在END while 关键字后面的任何语句块，end while 关键字为循环结束标记**

**ITERATE begin\_label:用于跳出本次循环，然后直接进入下一次循环，忽略ITERATE语句后的任何语句**

**例：**

**CREATE PROCEDURE p5()**

**BEGIN**

**DECLARE i,sum INT;**

**SET i = 1,sum = 0;**

**WHILE i <= 100 DO**

**SET sum = sum +i;**

**SET i = i + 1;**

**END WHILE;**

**SELECT sum AS '1到100的和是';**

**END ;**

**嵌套查询**

**例：**

**SELECT stuname FROM student**

**WHERE (YEAR(CURDATE())-YEAR(birthday)) >**

**(SELECT avg(year(CURDATE())-year(birthday)) FROM student**

**WHERE birthday is not NULL)**

**例：**

**CREATE PROCEDURE p5()**

**BEGIN**

**DECLARE i,sum int;**

**set i=1,sum=0;**

**WHILE i<=100 do**

**set sum=sum+i;**

**set i=i+1;**

**end WHILE;**

**SELECT sum as '1到100的和是';**

**end;**

**call p5()**

**游标：简单来说就是指查询结果中的一条记录**

**1.定义游标**

**declare cur\_name cursor for 查询语句**

**2.打开游标：其实就是把查询语句执行一次**

**open cur\_name**

**3.获取当前游标记录（经常要循环）**

**fetch cur\_name into 变量列表**

**4.关闭游标**

**close cur\_name**

**例：找5位男同学，按年龄由大到小排序 分配这5位同学按楼层做值日，显示这些同学的姓名和值日楼层**

**create procedure proc5()**

**begin**

**declare sname varchar(20);**

**declare flr int default 0;**

**declare cur\_test cursor for**

**select stuname from student where sex = '男' order by timestampdiff(year,birthday,curdate()) desc,stuid limit 5;**

**open cur\_test;**

**fetch cur\_test into sname;**

**set flr = flr + 1;**

**select sname as '姓名',flr as '值日楼层';**

**fetch cur\_test into sname;**

**set flr = flr + 1;**

**select sname as '姓名',flr as '值日楼层';**

**fetch cur\_test into sname;**

**set flr = flr + 1;**

**select sname as '姓名',flr as '值日楼层';**

**fetch cur\_test into sname;**

**set flr = flr + 1;**

**select sname as '姓名',flr as '值日楼层';**

**fetch cur\_test into sname;**

**set flr = flr + 1;**

**select sname as '姓名',flr as '值日楼层';**

**end;**

**call proc5();**

**【注意】mysql的游标是向前只读的，也就是说，只能顺序的从开始往后读取结果集，不能从后往前，也不能直接跳到中间的记录**

**【重点】MySQL常用函数（书本200页）：**

**1.字符串函数**

**CHAR\_LENGTH(str) ——字符串长度**

**CONCAT(‘1’,‘2’,‘3’) ——合并字符串**

**REVERSE('123456') ——替换函数，把函数转换**

**2.数学函数**

**ABS（x）——取绝对值**

**3.日期时间函数**

**CURDATE（） ——日期型**

**事务：就是用户定义的一个数据库操作序列，这些操作要么全做，要么全不做，是一个不可分割的工作单位。**

**1.原子性**

**2.一致性**

**3.隔离性**

**4.持久性**

**start transaction；——开启事务，在这条语句之后的mysql将处在同一事务中，并不会立即影响数据库**

**commit；——提交事务，让这个事务中的mysql对数据库的影响立即发生**

**rollback；——回滚事务，取消这个事务，一旦提交了事务，则不可回滚**

**例：**

**START TRANSACTION;**

**UPDATE account**

**set money=money-5000**

**WHERE id=1;**

**转钱**

**UPDATE account**

**set money=money+5000**

**WHERE id=2;**

**系统崩溃，代码回滚**

**COMMIT;**

**SELECT \* FROM account**

**user表：user表是MySQL中最重要的一个权限表，记录允许连接到服务器的账号信息。**

**创建用户，设置用户名和密码：**

**CREATE USER**

**'king'@'localhost' IDENTIFIED by '123456';**

**设置用户权限：**

**GRANT SELECT ON qq.\* TO 'king'@'localhost';**

**高级操作：**

**备份：如果要备份存储过程，则必须在书上的命令基础上增加一个大写R的参数，表示routines（复数），如下所示：**

**mysqldump -uroot -p123456 qq -R > c:/myqq.sql 新建备份**

**mysqldump -u root -p 123456 qq < c:/myqq.sql 找回备份**