# Mysql数据库

## 1.1 数据库的特点

1. 可以存储大量的数据。
2. 方便检索数据。
3. 可以实现数据共享。
4. 维护了数据的完整性与一致性。
5. 通过组合查询可以产生新的数据。

## 1.2 市场上常见的数据库

1.Oracle 公司名字|产品名字(甲骨文) 收费的， 按照cpu核数收费， 一个核终生是20w年， 5核\* 20 = 100w 。 以后每年缴纳20%服务费。

2.Mysql 一开始是免费，现在从6.0开始收费的。 瑞典AB公司 ----sun--->Oracle

3.SqlServer 微软的, .net、 C#程序员使用

4.DB2 IBM公司，收费，性能比oracle差，但是收费核oracle差不多。

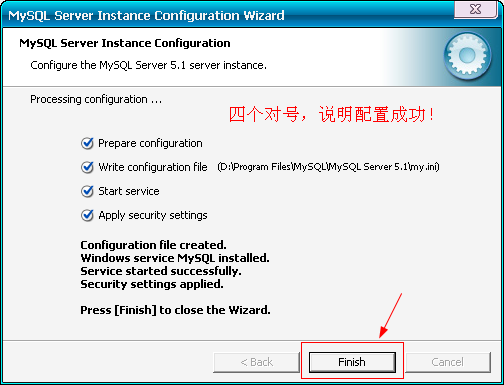
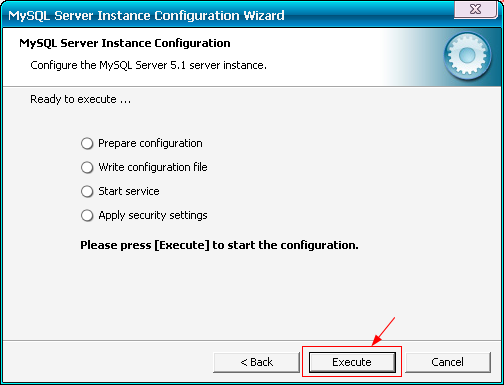
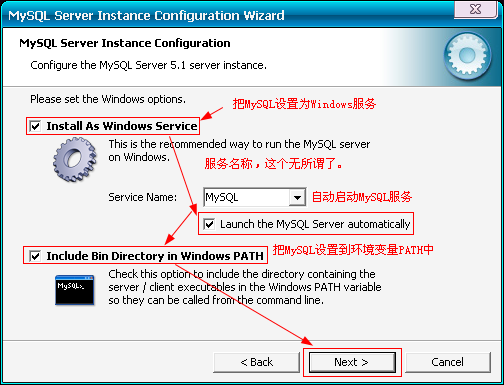
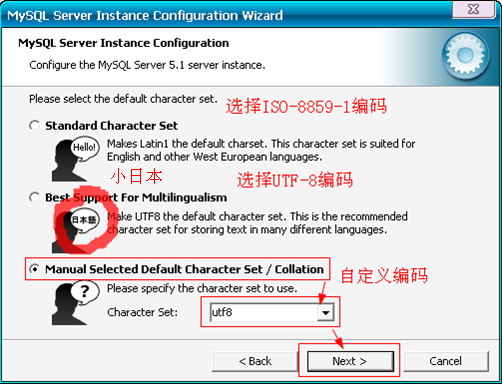
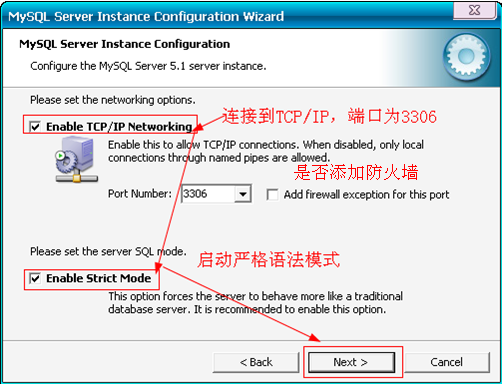
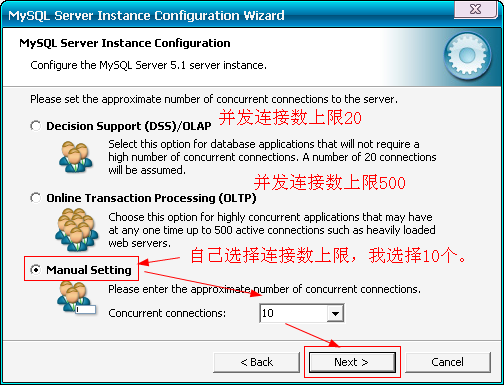
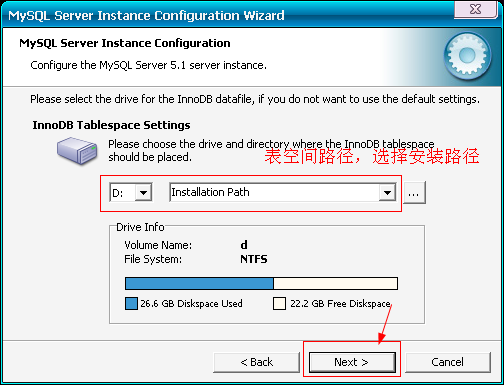
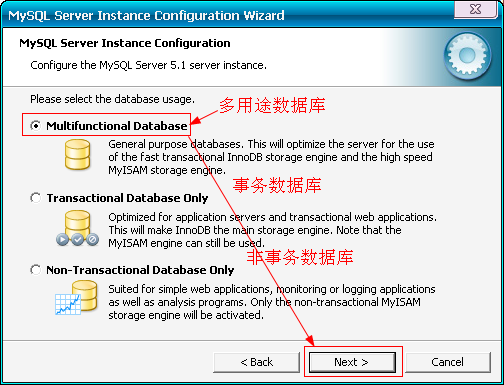
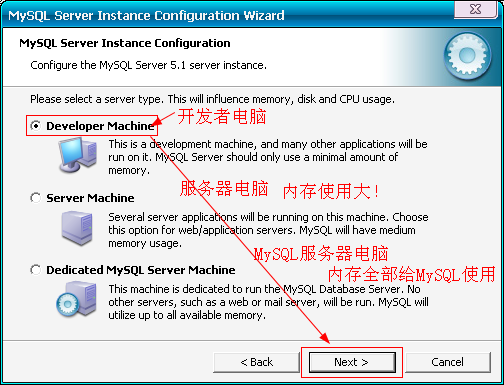
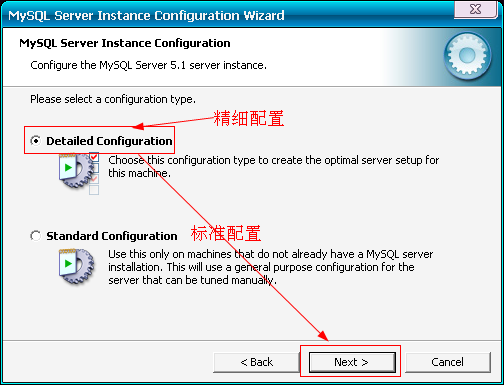
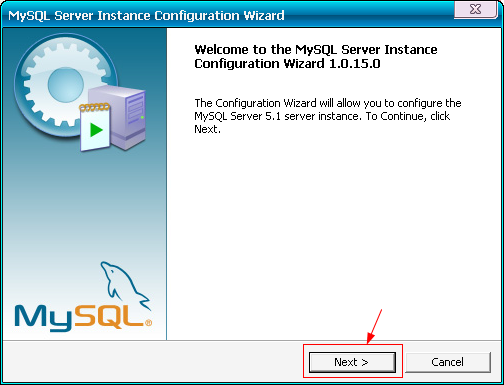
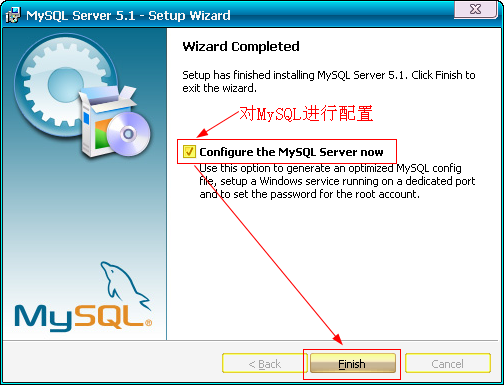
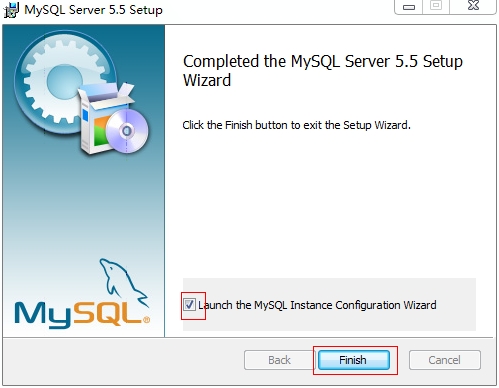
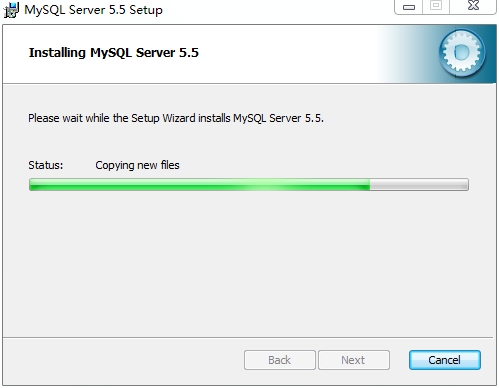
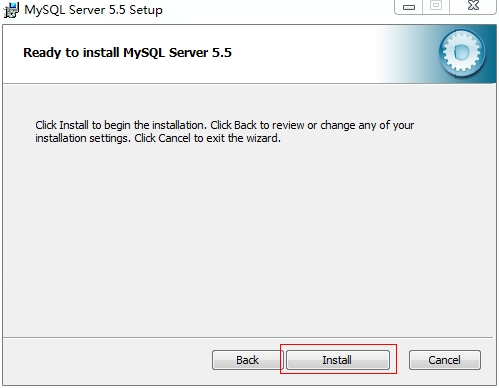
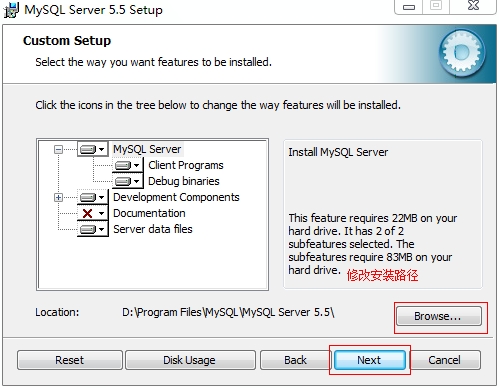
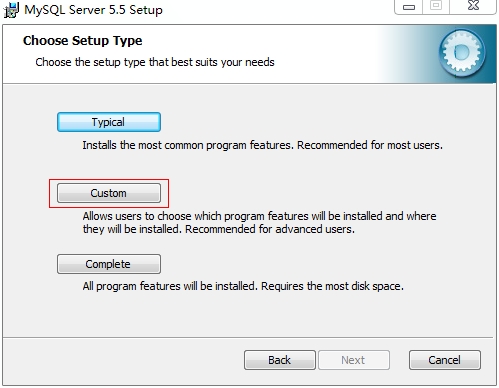
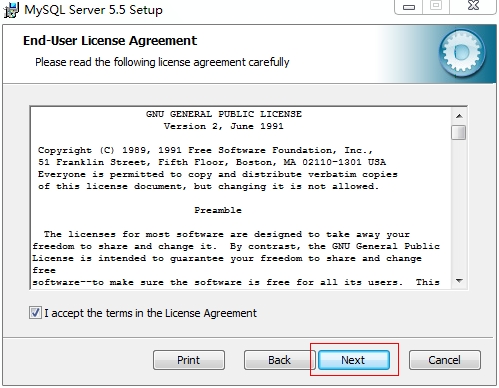
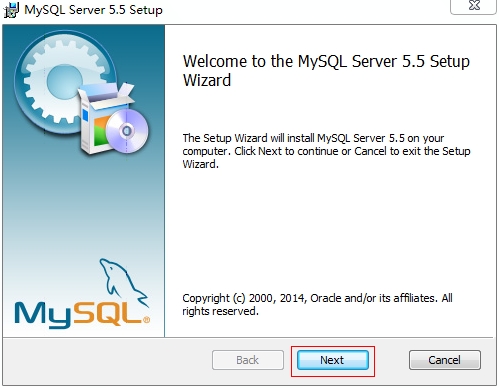
5.Sqllite 安卓系统自带数据库

## 1.3 Mysql数据库服务器的安装

### 1.3.1 安装的注意事项

**安装文件存放路径：不能有中文和空格！**

### 1.3.2 安装图解



### 13.3.3 安装后的校验

打开控制台cmd

Mysql -uroot -p密码--------->Mysql -uroot -p123

## 1.4 Mysql的卸载

1. 停止mysql服务。

A. 控制面板停止

B.通过控制台的命令去停止

Net stop mysql ----------->停止mysql服务。

Net start mysql--------->开启mysql服务.

1. 卸载mysql的软件。

3. 删除mysql的安装目录。

4. 删除mysql原有的安装数据。

C:\Documents and Settings\All Users\Application Data\MySQL

C:\ProgramData\MySQL

1. 清理注册表的数据()

regedit

HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services

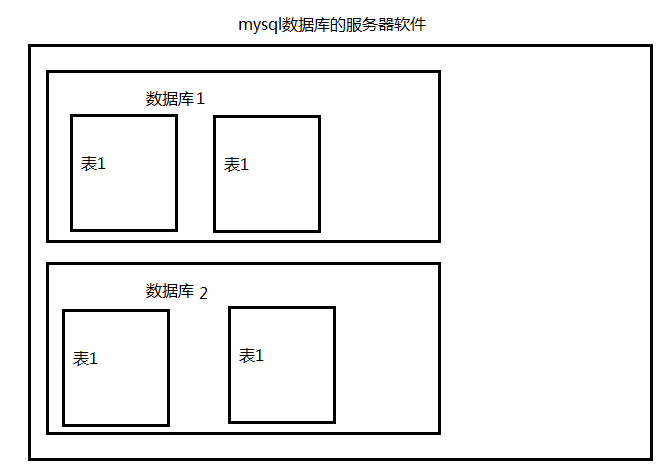
HKEY\_LOCAL\_MACHINE\SYSTEM\ControlSet001\Services

HKEY\_LOCAL\_MACHINE\SYSTEM\ControlSet002\Services

1. 重启电脑。

7.重新安装。

## 1.5 数据库管理系统与数据库与表之间的关系



数据库服务器的软件---------可以创建很多数据库----------一个数据库可以创建很多张表------一张表可以存储很多的数据

## 1.6 如何链接Mysql的服务器

### 1.6.1 方式一：可以使用控制台的方式去连接

|  |  |
| --- | --- |
| **方式一：用户名一般是root** | **Mysql -u用户名 -p密码** |
| **方式二：这个密码是隐藏的** | **Mysql -u用户名 -p回车 输入密码** |
| **方式三：链接别人的数据库服务器** | **Mysql -uroot -p密码 -h(别人的IP地址)** |

### 1.6.2 方式二：图形化界面

**就可以使用图形化界面工具链接**

## 1.7 sql语句

### 1.7.1 sql语句的作用

指挥Mysql服务器干活的命令

我们现在使用的是**sql03**标准法语句

### 1.7.2 sql语句的分类

|  |  |
| --- | --- |
| **DDL(data definition language)** | **数据定义语言，主要用于创建数据库、修改数据库与创建表、修改表的使用的** |
| **DML(Data manipulation language )** | **操作表中的数据** |
| **DQL(Data query language )** | **数据查询语言** |
| **DCL(Data Control Language)** | **控制权限，用于控制哪些用户可以操作数据** |

## 1.8 DDL语句(database的操作)

### 1.8.1 DDL的注意事项

DDL语句的作用就是用于操作数据库与操作表结构的。注意DDL语句并不是用于操作数据。

### 1.8.2 创建数据库

|  |  |
| --- | --- |
| **方式一：直接创建数据库** | **Create database 数据库名；** |
| **方式二：先判断是否存在指定的数据** | **Create database if not exists 数据库名;** |
| **方式三： 指定数据库使用的码表（字符集）创建数据库** | **Create database if not exists 数据库名 character set 码表** |

注意：A.后面一定要有分号，分号表示结束的意思

B.安装数据库服务器的时候默认使用的码表是utf-8

### 1.8.3 查看数据库

|  |  |
| --- | --- |
| **显示所有的数据库** | **Show databases；** |
| **查看创建数据库时的语句** | **Show create database 数据库名 ；** |

### 1.8.4 删除数据库

|  |  |
| --- | --- |
| **直接删除数据库** | **Drop database 数据库名；** |
| **先判断是否存在再删除** | **Drop database if exists 数据库名** |

### 1.8.5 修改数据库

|  |  |
| --- | --- |
| **修改数据库的码表** | **Alter database 数据库名 character set ‘码表’** |

### 1.8.6 指定在一个数据库中操作

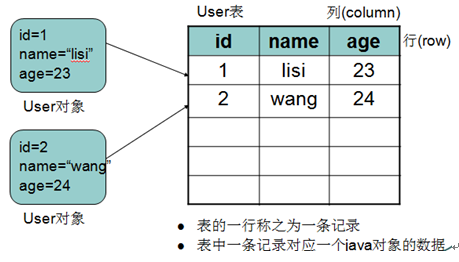
|  |  |
| --- | --- |
| **选用指定的数据库进行操作** | **Use 数据库名** |

## 1.9 DDL语句(table的操作)

### 1.9.1 表数据

根据表字段所规定的数据类型，我们可以向其中填入一条条的数据，而表中的每条数据类似类的实例对象。表中的一行一行的信息我们称之为记录。

* 表记录与java类对象的对应关系



### 1.9.2 创建表

|  |  |
| --- | --- |
| 直接创建表 | **Create table 表名(列名1 数据类名1，**  **列名2 数据类型2，....);** |

**例子：**

create table Person(

id int,

name varchar(6),

birthday timestamp

);

### 1.9.3 mysql的数据类型

|  |  |  |
| --- | --- | --- |
| **整数** | **tinyInt** | 一个字节 2\*8 -128~127 |
| **smallInt** | 2个字节 2^16 |
| **int** | **4个字节 （常用）** |
| **Bigint** | 8个字节 |
| **小数** | **Float(单精度浮点型)** | 最多能够保留7个小数位 |
| **Double(双精度浮点型)** | 最多可以保留15个小数位 |
| **字符型** | **Char(固定长度)** | 固长 , 最多可以存储256个字符 |
| **Varchar(长度可变)** | 可变长, 最多可以存储65536个字符。 |
| **布尔类型** | **boolean** | True和false |
| **时间类型** | **date** | 只会存储日期 |
| **time** | 只会存储时间 |
| **timestamp** | 日期与时间都存储，如果存储的时候没有指定值，那么默认会使用当前的系统时间。 |

**Double用法：**

**方式一： 可以不指定double的小数位,对小数位没有限制**

Create table person(

Id int,

Salary double

);

**方式二：指定double的小数位。**

Create table person(

Id int,

Salary double(5,2)

);

**字符数据**

**Char(字符的长度) 固长 , 最多可以存储256个字符**

**Varchar(字符串长度) 可变长, 最多可以存储65536个字符。**

**选用的原则： 固长的时候用固长，变长的时候用变长。**

**固长的数据： 手机号码(11) ，身份证号码（18），生日**

**变长**： 姓名。

### 1.9.4 查看表

|  |  |
| --- | --- |
| **查看所有的表** | **Show tables；** |
| **查看创建表的语句** | **Show create table 表名；** |
| **查看表的结构** | **Desc 表名；** |

### 1.9.5 删除表

|  |  |
| --- | --- |
| **直接删除表** | **Drop table 表名** |

### 1.9.6 修改表

|  |  |
| --- | --- |
| **添加列** | **Alter table 表名 add 列名 数据类型**  例子：alter table person add salary int; |
| **修改列类型** | **Alter table表名 modify 列名 新的数据类型**  alter table person modify salary double; |
| **更改列名** | **Alter table表名 change 旧列名 新列名 数据类型;**  alter table person change salary money int; |
| **修改表名** | **Alter table表名 rename to 新表名**  alter table person rename to student; |
| **删除列** | **Alter table表名 drop 列名**  alter table student drop money; |

## 1.10 DML操作表中数据

### 1.10.1 查询表中所有的数据

|  |  |
| --- | --- |
| **查询表中所有的数据** | **Select \* from 表名** |

### 1.10.2 表中插入字段

|  |  |
| --- | --- |
| **插入表中所有字段** | **Insert into表名 values(数据1，数据2，数据3...)；**  Insert into student values(1,’gouwa’); |
| **插入表中的指定字段** | **Insert into 表名(字段名1，字段名2...) values(数据1，数据2，数据3.....)；**  insert into person(name) values(‘gousheng’); |

### 1.10.3 更新数据

|  |  |
| --- | --- |
| **修改数据**  Update person set id = 110 where name=’yuling’; | **Update 表名 set 字段名1= 数据1, 字段名2=数据2 where 条件（指定id对应的name）；** |

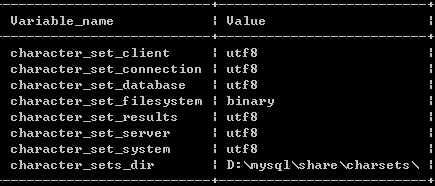
update emps set name=?,age=?,salary=? where id=?

### 1.10.4 删除数据

|  |  |
| --- | --- |
| **清空表的数据(一条一条数据删除，效率低).** | **Delete from 表名 where 条件;** |
| **摧毁表结构进行清空表数据** | **Truncate 表名** |
| **删除指定的数据** | **Delete from 表名 where 条件** |

## 1.11 解决中文问题

show variables like '%charac%';  **把mysql与字符相关的变量全部显示出来。**



**Character\_set\_client: mysql服务器会认为客户端使用的码表是utf8.**

**Character\_set\_results：对结果使用的码表。**

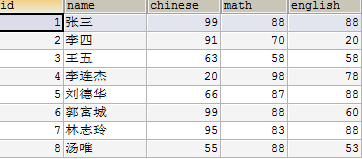
Character\_set\_connnection

**一劳永逸的方法： 找到mysql的安装目录，然后找到my.ini文件，然后把default\_character\_set改成gbk即可。**

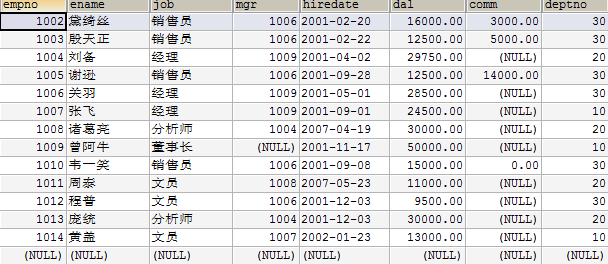
# DQL查询语句

**\*表示所有字段，顺序与表结构相同，也就是所有的列名**

**学生表**



**员工表**



## 2.1 普通查询的格式

|  |  |
| --- | --- |
| **1.查询学生表中的所有学生信息** | **SELECT \* FROM 表名;**  SELECT \* FROM student; |
| **2.查询指定列的学生信息(学生的语文成绩和名字)** | **Select 列名1,列名2 from 表名;**  **Select name,chinese from student;** |
| **3.查询的时候使用别名，即将第一行的要查询的名称改成中文的** | **Select name as姓名,math as 数学from student;------>方式一，标准** |
| **Select name 姓名,math 数学from student;----->方式二，方言** |

### 2.1.1 distinct去除重复数据

|  |  |
| --- | --- |
| **1.查找工作岗位**  **Distinct是去除掉重复的数据** | **Select distinct 列名 from 表名；**  SELECT DISTINCT job FROM emp; |

### 2.1.2 在查询结果上运算

|  |  |
| --- | --- |
| **1.在所有学生分数上加10分特长分(查询的结果是可以运算)** | **SELECT NAME ,chinese+10 语文, math+10 数学,english+10 英语 FROM students;** |

## 2.2 条件筛选查询(where)

|  |  |
| --- | --- |
| **1.查找李连杰的所有信息，\*表示所有的信息** | **Select \* from 表名 where name=’李连杰’;**  **Select \* from student where name=’李连杰’;** |
| **2.查询英语成绩大于90分的同学** | SELECT \* FROM students WHERE english >= 90; |
| **3.查询总分大于200分的所有同学** | SELECT NAME,(chinese+math+english) 总分 FROM student WHERE (chinese+math+english)>200; |
| **4.查询数学分数为89或者90或者91的同学** | SELECT NAME,math FROM student WHERE math **IN**(89,90,91);**---->in语句** |
| SELECT NAME , math FROM students WHERE math=89 **OR** math=90 **OR** math=91;**---->or语句** |
| **5.查询英语分数在 80－90之间的同学，包含80和90** | SELECT NAME ,english FROM student WHERE **english>=80 AND english<=90**; |
| SELECT NAME, english FROM student WHERE english **BETWEEN 80 AND 90**;**----->between语句** |

## 2.3 模糊查询(like)

**查询所有姓'李'的学生成绩，%可以匹配0个字符或多个字符**

**下划线\_表示1个字符**

|  |  |
| --- | --- |
| 查询所有名'李'字的学生成绩 | SELECT \* FROM students WHERE NAME **LIKE '\_%李%'**; |
| 查询所有姓李的学生，并且姓名只有两个字 | SELECT \* FROM students WHERE NAME LIKE '李\_'; |
| 查询所有姓'李'的学生成绩,但姓名必须是三个字符，\_表示1个字符 | SELECT \* FROM students WHERE NAME LIKE '李\_\_'; |

## 2.4 排序查询(**order by 语句**)

**降序： desc 升序： asc 默认是升序**

|  |  |
| --- | --- |
| 对数学成绩排序(降序)后输出 | SELECT NAME,(IFNULL(math,0)) FROM student **ORDER BY math DESC**; |
| 对数学成绩进行首先排名，如果数学成绩一样，根据语文成绩进行排名 | SELECT NAME,math,chinese FROM student ORDER BY math DESC,chinese DESC; |
| 对姓'李'的学生总分排序(降序)输出 , **order by语句是要在最后位置** | SELECT NAME,((IFNULL(math,0))+(IFNULL(chinese,0))+(IFNULL(english,0))) 总分 FROM student WHERE NAME LIKE '李%' ORDER BY 总分 DESC; |

**同时进行排序**

**ORDER BY sal DESC,empno ASC;**

## 2.5 聚合函数(count、sum、avg)

### 2.5.1 count的用法

|  |  |
| --- | --- |
| 统计一个班级共有多少学生 count() 统计行数。**----->建议括号里面用列名** | Select **count(列名|\*)** from students;  两种方式，括号里面可以是列名或\* |
| 统计数学成绩大于80的学生有多少个  符合条件之后再统计 | SELECT COUNT(math) FROM student WHERE math>80; |
| 统计总分大于250的人数有多少人 | SELECT COUNT(math) FROM student WHERE (math+chinese+english)>250; |

**注意：如果是null值会被 排除统计的**

### 2.5.2 sum的用法

|  |  |
| --- | --- |
| 统计这个班级数学的总成绩 | SELECT **SUM(math)** FROM student; |
| 统计一个班级语文、英语、数学各科的总成绩 | SELECT SUM(math),SUM(chinese),SUM(english) FROM student; |
| 统计一个班级语文、英语、数学的成绩总和 | SELECT SUM(math+english+chinese) FROM student; |

### 2.5.3 vag的用法

|  |  |
| --- | --- |
| 统计一个班级语文成绩平均分 avg | SELECT AVG(chinese) FROM student; |
| 计算一个班级的数学平均分 | SELECT AVG(IFNULL(math,0)) FROM students; |

### 2.5.4 最高和最小的用法

|  |  |
| --- | --- |
| 求班级上数学的最高分 | SELECT MAX(IFNULL(math,0)) FROM student; |
| 求班级上数学的最低分 | SELECT MIN(IFNULL(math,0)) FROM students; |

## 2.6 分组查询(group by)

分组查询是把记录使用某一列进行分组，然后查询组信息。

|  |  |
| --- | --- |
| **统计每个工作岗位的人数** | SELECT job, COUNT(job) 人数 FROM emp **GROUP BY** job; |
| **统计工资大于15000的每个工作岗位人数。** | SELECT job, COUNT(job) FROM emp WHERE dal>15000 GROUP BY job; |
| 统计工资大于15000的每个工作岗位人数，并且该工作岗位的人数大于等于3. | SELECT job,COUNT(job) FROM emp WHERE dal>15000 GROUP BY job **HAVING COUNT(job)>=3;** |

统计工资大于15000的每个工作岗位人数，并且该工作岗位的人数大于等于3. ， 分**组之后的条件不能使用WHERE语句，要使用HAVING语句**。

## 2.7 limit句子(方言)

LIMIT用来限定查询结果的起始行，以及总行数。

例如：查询起始行为第5行，一共查询3行记录

**SELECT \* FROM emp LIMIT 开始查询的索引,查询的条数;**

SELECT \* FROM students LIMIT 0 ,3;

# DCL权限控制语言

## 3.1 创建新用户

**CREATE USER 用户名@华为总部ip地址 IDENTIFIED BY ‘密码’；**

Create user [aaa@'](mailto:aaa@192.168.1.1)%’ identified by abc123;

**例子：** CREATE USER aaa@localhost IDENTIFIED BY 'abc123'

**注意：** 新创建的用户是不具备任何一个数据库的权限的。

## 3.2 给用户授权

权限： select ， update 、 insert \delete 所有权限 all

**格式：**

**Grant 权限1,权限2 on 数据库名.\* to 用户名@ip地址;**

**例子：** GRANT UPDATE ON test.\* TO aaa@localhost;

## 3.3 查看用户目前的权限

**Show grants for 用户名@ip地址；**

**例子：** SHOW GRANTS FOR aaa@localhost;

## 3.4删除权限

**Revoke 权限1，权限2 on 数据库名.\* from 用户名@ip地址;**

**例子：** REVOKE DELETE ,UPDATE ON test.\* FROM aaa@localhost;

### 3.4.1修改管理员密码(非登录的情况下去使用)

**Mysqladmin -u用户名 -p password 新密码 回车**

**输入原密码**

### 3.4.2 修改普通用户的密码

//修改普通用户权限

**UPDATE USER SET PASSWORD = PASSWORD('用户密码') WHERE USER='用户名' AND HOST='用户的IP地址';**

//刷新当前的权限

**FLUSH PRIVILEGES;**

### 3.5 删除用户

**Drop user 用户名@IP地址**

# 第4章 数据库的备份

## 4.1 mysqldump工具, 在不登录情况下去使用

**使用格式：**

**Mysqldump -u用户名 -p密码 数据库名>路径(存放的路径)**

**注意：路径后面不要加分号**

## 4.2 数据库的还原

**还原的方式（登录情况并且进入到指定的数据库下）：**

**Source 备份文件的路径**

注意：

1.路径后面不要加分号

2.在还原之前，一定要先创建一个与导入一样名字的数据库

## 4.3 图形化界面

图形化界面直接备份

# 数据库的约束

## 5.1主键约束(primary key)

在数据库中每一条数据都应该有一个唯一的标识，而这个唯一的标识就是使用主键表示。

在现实开发中任何一张表都应该有一个主键字段，因为需要确保每条数据的唯一性。

主键字段往往都是不具备任何业务含义的字段。

### 5.1.1 主键具备的特性

1. 唯一。
2. 非空
3. 被引用。

## 5.2 创建表时添加主键

**添加创建表添加主键:**

CREATE TABLE person(

**id INT PRIMARY KEY,**

NAME VARCHAR(6),

age INT

);

## 5.3后期添加主键

**后期添加主键**

ALTER TABLE person **ADD PRIMARY KEY(id);**

## 5.4 删除主键

**Alter table 表名 drop primary key;**

## 5.5 主键自增长

**添加自增长:**

CREATE TABLE person(

id INT PRIMARY KEY **AUTO\_INCREMENT,**

NAME VARCHAR(6),

age INT

);

例子：

INSERT INTO person(NAME,age) VALUES('狗娃',12);

INSERT INTO person(NAME,age) VALUES('狗剩',12);

**//主键已经是自增长的情况下，我是否可以再插入主键**

INSERT INTO person(id,NAME,age) VALUES(10,'铁蛋',12);

**//删除数据是不会影响到自增长的基数的。 除非是TRUNCATE表。**

INSERT INTO person(NAME,age) VALUES('旺财',12);

## 5.6 非空约束 not null

CREATE TABLE person(

id INT PRIMARY KEY AUTO\_INCREMENT,

NAME VARCHAR(6),

age INT **NOT NULL**

);

## 5.7 唯一约束 unique

CREATE TABLE person(

id INT PRIMARY KEY AUTO\_INCREMENT,

NAME VARCHAR(6) NOT NULL **UNIQUE**,

age INT NOT NULL

);

这种情况下名字是唯一的

## 5.8 默认值

CREATE TABLE person(

id INT PRIMARY KEY AUTO\_INCREMENT,

NAME VARCHAR(6) NOT NULL UNIQUE,

age INT NOT NULL,

gender CHAR(1) **DEFAULT** '男'

);

## 5.9 外键的约束

**作用：约束两种表的数据的完整性**

**Create table 表名(**

**字段 属性1,**

**.....**

**Constraint 外键名字 foreign key (本表的外键字段) references 其他表（字段）**

**)**

/\*创建部门表 \*/

CREATE TABLE dept(

id INT PRIMARY KEY AUTO\_INCREMENT,

NAME VARCHAR(6) NOT NULL

);

/\* 员工表\*/

CREATE TABLE emp(

id INT PRIMARY KEY AUTO\_INCREMENT,

NAME VARCHAR(6) NOT NULL,

salary DOUBLE,

deptId INT,

/\*建立外键的关系 \*/

**CONSTRAINT d\_e\_fk FOREIGN KEY (deptId) REFERENCES dept(id));**

### 5.9.1 删除外键

**Allter table 表名 drop foreign key 外键的名字;**

**例子：**

ALTER TABLE emp DROP FOREIGN KEY d\_e\_fk;

### 5.9.2 后期添加外键

**Alter table 表名 add constraint 外键名字 foreign key (本表字段) references 其他表名(其他表的字段)；**

**例子：** ALTER TABLE emp ADD CONSTRAINT d\_e\_fk FOREIGN KEY (deptId) REFERENCES dept(id);

### 5.9.3 建立外键后的注意事项

**一旦建立了外键的关系我们操作主表与从表的顺序要按照以下顺序：**

**添加数据的时候：** 先添加主表数据，然后再添加从表数据。

**删除数据的时候：** 先删除从表数据，再删除主表数据**。**

**修改数据的时候：** 先修改从表的数据，再修改主表的数据。

## 5.10 外键级联

### 5.10.1 什么是外键级联

就是从表或者说副表与主表建立了外键关系后，那么当主表的数据发生改变的时候会自动对从表的数据进行相应的处理。

### 5.10.2 外键级联的格式

修改级联

On update cascade;----------->放在建立外键时的后面

删除级联

On delete cascade;----------->放在建立外键时的后面

例子：

员工表：

Create table emp(

Id int primary key auto\_increment,

Name varcharacter(6) notnull,

Salary double(5,3),

Deptid int,

Constraint empdept\_fk foreign key (deptid) references dept(id) on update cascade on delete cascade);

部门表：

Create table dept(

Id int primary key auto\_increment,

Name varcharacrer(6) notnull);

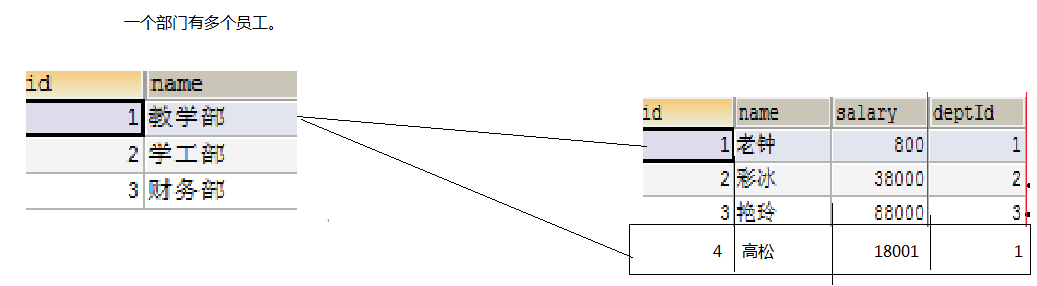
**注意：**

**1.on update cascade on delete cascade两个一起写的时候是没有逗号分开的**

**2.**On delete cascade的运用，只要删除了主表被引用的数据，那么从表所对应的数据也会被删除

# 第6章 表与表的关系及解决方案

## 6.1 一对多(出现次数最多)



### 6.1.1 一对多的解决方案

**在多的一方建立外键，然后引用着主表的数据。**

**例子：**

CREATE TABLE dept(

id INT PRIMARY KEY AUTO\_INCREMENT,

NAME VARCHAR(6) NOT NULL

);

/\* 员工表\*/

CREATE TABLE emp(

id INT PRIMARY KEY AUTO\_INCREMENT,

NAME VARCHAR(6) NOT NULL,

salary DOUBLE,

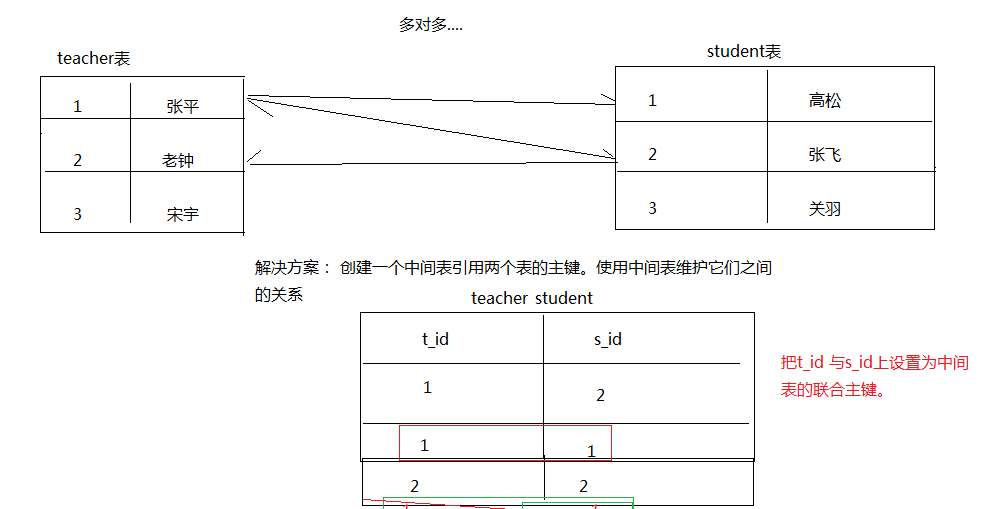
deptId INT,

/\*建立外键的关系 \*/

CONSTRAINT d\_e\_fk FOREIGN KEY (deptId) REFERENCES dept(id) ON DELETE CASCADE ON UPDATE CASCADE

);

## 6.2 多对多(出现较多)



### 6.2.1 多对多的解决方案

1.创建一张中间表引用两个表的主键。

2.把中间表的外键字段设置为联合主键即可

CREATE TABLE teacher(

id INT PRIMARY KEY AUTO\_INCREMENT,

NAME VARCHAR(6)

);

CREATE TABLE student(

id INT PRIMARY KEY AUTO\_INCREMENT,

NAME VARCHAR(6)

);

/\*创建中间表\*/

CREATE TABLE teacher\_student(

t\_id INT ,

s\_id INT,

/\*设置外键 \*/

CONSTRAINT teacher\_fk FOREIGN KEY(t\_id) REFERENCES teacher(id),

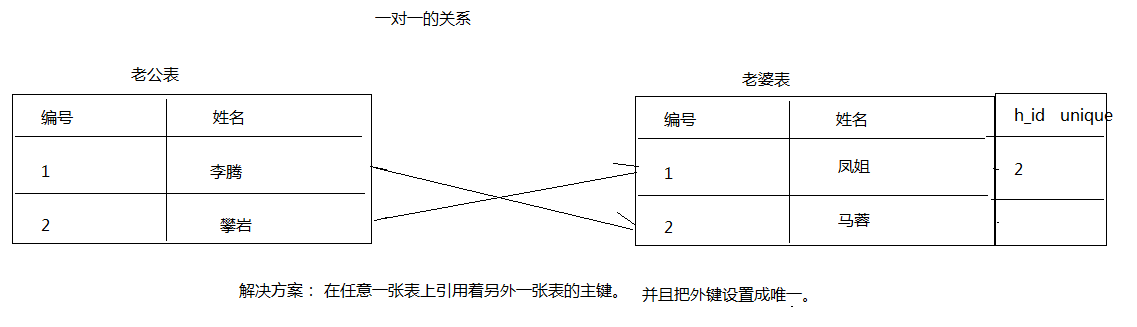
CONSTRAINT student\_fk FOREIGN KEY(s\_id) REFERENCES student(id),

/\*设置联合主键 \*/

PRIMARY KEY(t\_id , s\_id)

);

## 6.3 一对一



### 6.3.1 一对一的解决方案

**在任意一张表上引用着另一张表的主键，并且把外键设置成唯一。**

CREATE TABLE husband(

id INT PRIMARY KEY AUTO\_INCREMENT,

NAME VARCHAR(6)

);

CREATE TABLE wife(

id INT PRIMARY KEY AUTO\_INCREMENT,

NAME VARCHAR(6),

h\_id INT **UNIQUE** , /\*设置唯一 \*/

**CONSTRAINT h\_w\_fk FOREIGN KEY (h\_id) REFERENCES husband(id)** /\* 设置外键引用\*/

);

# 数据库的三范式

数据库的三范式的作用： 以后大家设计表的时候要遵守的一些规则

## 7.1 第一范式： 一个表的任意一个字段都是不可再分割的单元

**人表：错误示范**

**编号 （name）姓名**

**1 张三|张平**

**2 小飞|小三**

需求： 查询曾用名带“三”字的人。

Select \* from Person where 曾用名like “%三%’;

**正确示范**

**编号 现用名 曾用名**

**1 张三 张平**

**2 小飞 小三**

## 7.2 第二范式 : 在第一范式的基础上，不同业务含义的字段应该划分到不同表中。 (一个事物直接属性应该设置该事物上，如果是间接属性就不应该保留到该事物上)

**错误示范：**

员工表 部门编号 部门名称

1 狗娃 1000 1 保洁部

2 狗剩 1200 1 保洁部

问题： 出现大量的数据冗余

**解决方案：**

员工表 部门id

1 狗娃 1000 1

2 狗剩 1200 1

部门编号 部门名称

1 保洁部

1 保洁部

## 7.3 第三范式: 在第二范式的基础上，一张表的主键除外的其他字段都应该跟主键字段是直接决定关系！

员工表 部门编号 部门名称

1 狗娃 1000 1 保洁部

2 狗剩 1200 1 保洁部

-------------------------------------------------------------------------------------------------------------------

员工表 部门id

1 狗娃 1000 1

2 狗剩 1200 1

部门编号 部门名称

1 保洁部

1 保洁部

第三范式强调的是分离表的基础上，我们要让表与表之间的建立关系。

**其实第二范式跟第三范式差不多**

# 多表查询

## 8.1 笛卡尔积现象

多表查询会出现笛卡尔积现象，

笛卡尔积现象：多表查询的时候一张表的每一条数据都会与另外一张表的每一条数据进行组合产生新的数据。

案例：

Create table dept: Create table emp:

**需要：查看员工表中对应的部门以及所有的信息**

**Select \* from dept,emp;**

查询结果如下：



**这种现象就是笛卡尔积现象**

### 8.1.1 清除笛卡尔积现象的方法

**SELECT \* FROM emp e , dept d WHERE e.deptId = d.id;**

添加过滤后的条件后结果是：



**这种方法只能打印出互相有对应的内容，如dept表中的总监部在emp表中是没有内容相互对应的，所以打印不出来总监部所对应的那行。**

## 8.2 内连接查询

### 8.2.1 内连接查询的语句

|  |  |
| --- | --- |
| **标准语法** | **Select \* from 表名1 inner join 表名2 on 清除笛卡尔积现象的条件**  SELECT \* FROM emp AS e INNER JOIN dept AS d ON e.deptId= d.id; |
| **方言语句** | **Select \* from 表名1 , 表名2 where 清除笛卡尔积现象的条件**  SELECT \* FROM emp e , dept d WHERE e.deptId = d.id; |

SELECT \* FROM emp e INNER JOIN dept d ON d.id=e.deptid;

运行结果如下：



其实内连接和清除笛卡尔积现象的结果是一样的，只有员工和部门都对应的才打印出来

## 8.3 外连接查询

### 8.3.1 左外连接语句

|  |  |
| --- | --- |
| **以左表为主不管条件是否满足条件，左表的数据都会全部显示出来** | **Select \* from 表名1 left outer join 表名2 on 清除笛卡尔积的条件** |

**SELECT \* FROM emp e LEFT OUTER JOIN dept d ON e.deptid=d.id;**

员工表为左边，不管员工表中的员工有没有对应的部门都会答应出来

结果如下：



### 8.3.2 右外连接语句

|  |  |
| --- | --- |
| **以右边表为主，不管条件是否满足，右边的表的数据全部都会显示出来** | Select \* from 表名1 **right outer join** 表名2 on 清除笛卡尔积 |

SELECT \* FROM emp e RIGHT OUTER JOIN dept d ON e.deptid=d.id;

结果如下：



一般情况下，打印出的表格姓名是在左边，所以再根据需求选择左边还是右边

## 8.4 合并结果集(union)

合并结果集就是将左边的表格和右边的表格的所有数据都打印出来

**UNION 合并结果集，如果数据是同一条，那么只会显示一条**

### 8.4.1 合并结果集的格式

|  |  |
| --- | --- |
| **两边的所有结果都打印出来** | SELECT \* FROM emp e LEFT OUTER JOIN dept d ON e.deptId = d.id  **Union------------------>注意：用union连接，不用括号**  SELECT \* FROM emp e RIGHT OUTER JOIN dept d ON e.deptId = d.id; |

**结果如下：**



## 8.5 子查询

### 8.5.1 什么是子查询

一个sql在执行的时候需要使用到另外一个sql的查询结果作为数据

### 8.5.2 子查询的结果可以存放的位置

1.Where的条件

2.From 后面，作为表数据

### 8.5.3 子查询的结果类

**1. 子查询的结果只有一个数据：** = > < 子查询的结果只能放在where作为调节。

2. **子查询的结果是单列多行**： In 子查询的结果只能放在where后面。

3. **子查询的结果出现多行多列 放在from后面作为 临时表使用**

## 8.6 综合练习

CREATE DATABASE constcompany;

USE constcompany;

/\*-- 创建工程表： 工程号(主键,字符串),工程名称\*/

CREATE TABLE project (

id VARCHAR(10) PRIMARY KEY,

NAME VARCHAR(10)

);

/\*-- 插入工程数据\*/

INSERT INTO project VALUES ('A1','花园大厦');

INSERT INTO project VALUES ('B2','立交桥');

INSERT INTO project VALUES ('C3','临江饭店');



/\*-- 2) 职务表: 职务编号(主键)，职务,小时工资率\*/

CREATE TABLE job(

id INT PRIMARY KEY AUTO\_INCREMENT,

NAME VARCHAR(10),

salary\_rate INT

)

/\*-- 插入职务表\*/

INSERT INTO job (NAME, salary\_rate) VALUES ('工程师',65),('技术员',60),('律师',100),('工人',55);



/\*-- 3) 员工表: 职工号(主键，从1001开始),姓名,职务编号(外键)\*/

CREATE TABLE emp (

id INT PRIMARY KEY AUTO\_INCREMENT,

NAME VARCHAR(10),

job\_id INT,

FOREIGN KEY(job\_id) REFERENCES job(id)

);

INSERT INTO emp(NAME,job\_id) VALUES ('杨国明',1),('班建斌',2),('伍岳林',3),('鞠明亮',4);



/\*-- 4) 工时表: 工时编号(主键)，工程号(外键),职工号(外键),工时\*/

CREATE TABLE work\_time(

id INT PRIMARY KEY AUTO\_INCREMENT,

project\_id VARCHAR(10),

emp\_id INT,

hours INT,

FOREIGN KEY (project\_id) REFERENCES project(id),

FOREIGN KEY (emp\_id) REFERENCES emp(id)

);

INSERT INTO work\_time (project\_id,emp\_id,hours) VALUES ('A1',1001,13);

INSERT INTO work\_time (project\_id,emp\_id,hours) VALUES ('A1',1002,16);

INSERT INTO work\_time (project\_id,emp\_id,hours) VALUES ('A1',1004,19);

INSERT INTO work\_time (project\_id,emp\_id,hours) VALUES ('B2',1001,13);

INSERT INTO work\_time (project\_id,emp\_id,hours) VALUES ('B2',1003,17);

INSERT INTO work\_time (project\_id,emp\_id,hours) VALUES ('C3',1002,18);

INSERT INTO work\_time (project\_id,emp\_id,hours) VALUES ('C3',1004,14);



**需求一：查询所有的人员的信息**

SELECT p.id,p.name,e.name,j.name,j.salary\_rate,wt.hours FROM emp e,job j,project p,work\_time wt WHERE wt.project\_id=p.id AND e.job\_id=j.id AND wt.emp\_id=e.id;



**需求二：查找花园大厦的所有人员信息**

SELECT pro.name,e.name,j.name,w.hours FROM project pro,emp e,work\_time w,job j WHERE pro.name='花园大厦' AND pro.id=w.project\_id AND e.job\_id=j.id AND w.emp\_id=e.id;



**需求三：查找工程师所对应的所有信息**

SELECT p.id,p.name,e.id,e.name,j.name,j.salary\_rate ,wt.hours FROM project p,emp e, job j,work\_time wt WHERE j.name='工程师'AND e.job\_id=j.id AND wt.emp\_id=e.id AND p.id=wt.project\_id;



# 总结

## 9.1 DDL

1.如果是对列名或者列名的数据类型进行修改用alter(**alter table 表名** modify(修改列名的数据类型)或者change(修改列名)或者rename to(修改表名)或者add(添加列名))，如果是直接对表格进行删除，用(**drop table 表名**)，如果是对表格中的数据进行删除，用(**delete from 表名 where 条件**)

## 9.2 DML

1. 查看：**select \* from 表名**；
2. 添加数据：**insert into** 表名(需要添加数据对应的列名) **values**(添加对应的值)
3. 对表格里面的数据进行修改：**update 表名 set** 要操作的列名=设置的值 **where 条件**；

4.完全摧毁表格，里面的内容全部删掉：truncate 表名；

## 9.3 DQL

1. 清除重复的：select **distinct** job from emp;
2. 模糊查询：**like**; 又涉及到了**%**和下划线**\_**，%表示能够和任意字符匹配，或者是代表任意的字符。下划线\_表示一个字符。这个一般运用在像查询姓李之类的。
3. 排序查询：(**order by desc需要排序的列名**)表示降序，order by asc表示升序，**这个语句放在最后面**。如果是需要排序两个列名的内容,要用逗号如：order by math desc,chinese desc;

4.聚合函数排序：**count/sum/vag/max/min** 注意：如果里面有内容为空null的话，统计人数的时候会忽略不计的，所以我们要把变成0，如：ifnull(math,0)

5.分组查询：**group by** 注意：group by 后面不能使用where条件

6.限制查询行数：(**limit** 查询起始行，查询终点行)

## 9.4 DCL(了解)

1. 创建新用户：create user 用户名@新用户使用的ip地址 identified 密码
2. 给新用户名授权：grant update on 数据库名.\* to 用户名@ip地址
3. 查看用户目前的权限：show grants for 用户名@ip地址
4. 删除用户：

## 9.5 约束

1. 外键级联：**on update cascade**------>更新，更新主表中的主键后，从表对应的外键也会自动更新

**On delete cascade**----->删除，删除主表中某个主键后，从表中对应的外键的那行也会被删除

2.外键：**constrain** 外键名 **foreign key** (本表外键字段) references 主表名(主表主键)；

## 9.6 表与表的关系

1.**一对多**：我们在创建表的时候，一张表跟另外一张表有一对多的话，需要在多的那方设置一个外键，引用着另外一张表的主键

2.**多对多**：如果有多对多的话，需要创建另外一张表，这张表要创建两个外键，并且引用着另外两张表的主键，并且把本表中的两个外键联合创建成本表中的主键

3.**一对一**：在其中的一张表上引用着另外一张表的主键并且设置成唯一

## 9.7 多表查询

1. 笛卡尔积现象：如果没有条件限制的话，直接查询，一张表会与另外一张表的数据进行匹配，全部打印出来
2. 内连接查询：inner join
3. 左外连接：left outer join
4. 右外连接：right outer join