# MYSQL

# 一、数据库及相关概念

## 1、数据库概念

长期存储在计算机内部的、有组织的、有共享的、统一管理的数据集合。

**数据库包括两层含义：保存数据的仓库、管理的方法和技术**

## 2、表的概念

用来存储数据和操作数据的逻辑结构，是一系列二维数组的集合。它是由横向的行和纵向的列构成。

**每一行叫做“记录”。**

每一列叫做“字段”。每一列表示记录的一个属性，包含有描述信息（如数据类型、数据宽度等）。

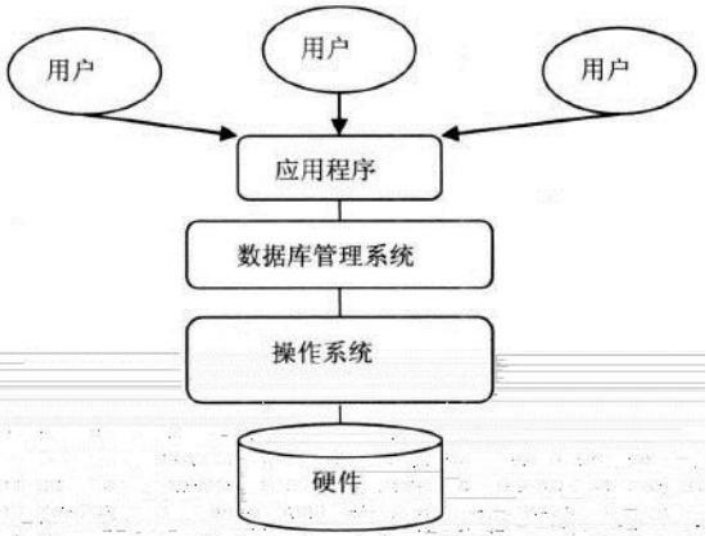
## 3、数据库系统

数据库系统包含有三个主要组成部分：

数据库：存储数据的地方

数据库管理系统：管理数据库的软件

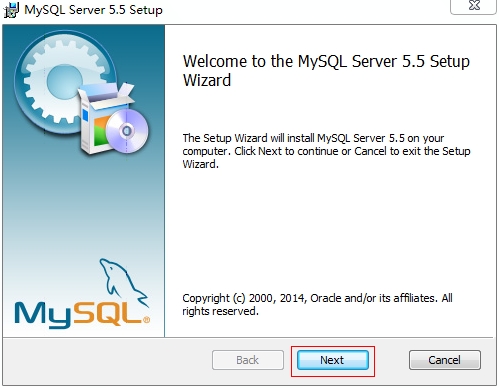
数据库应用程序：为了提高数据库系统的处理能力所使用的管理数据库软件的补充。

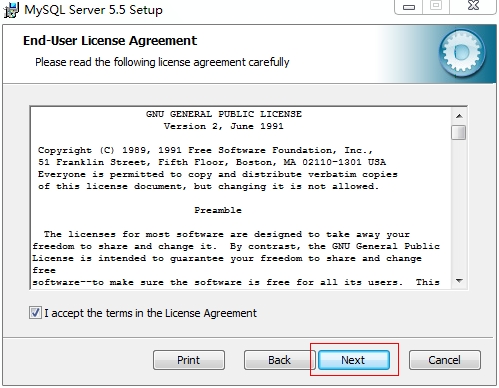


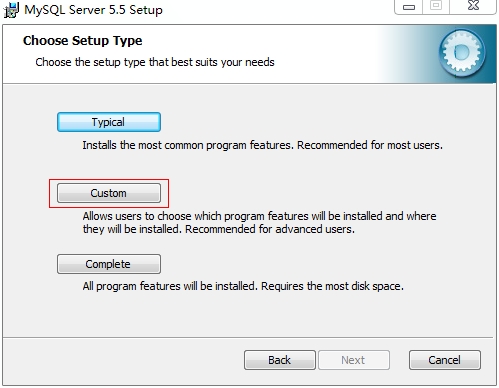
## 4、目前主流数据库产品

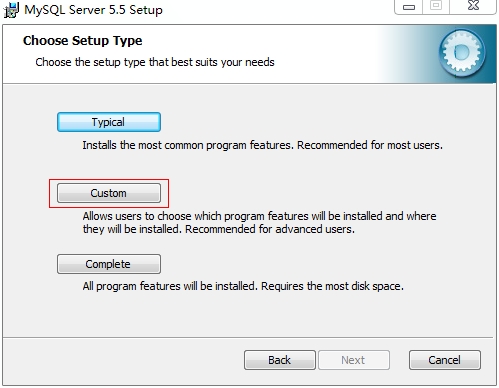
**Oracle、DB2**、SQLServer、**MySQL**等

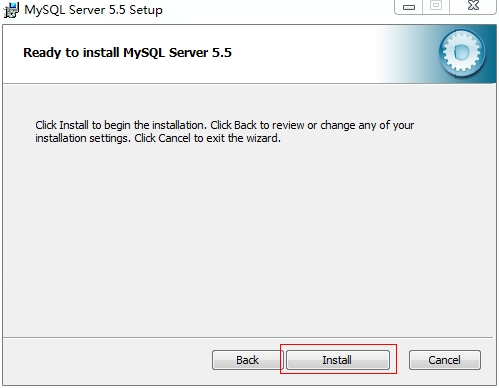
# 二、MYSQL数据库的安装

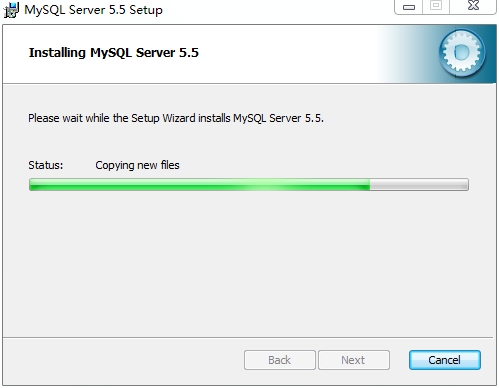


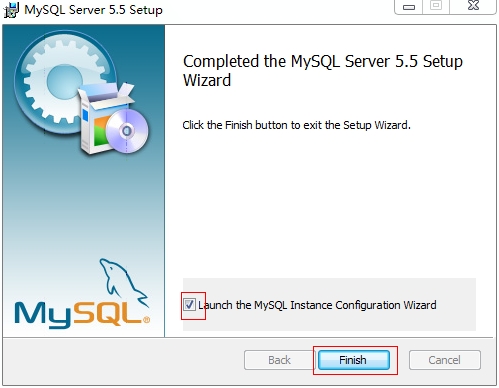


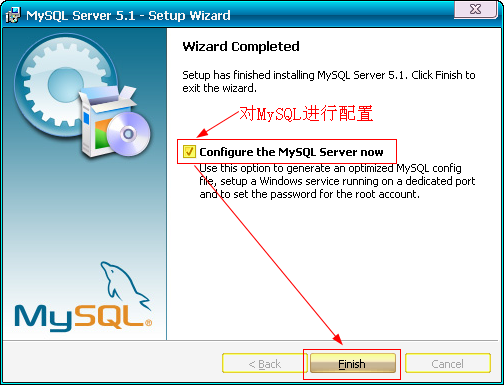


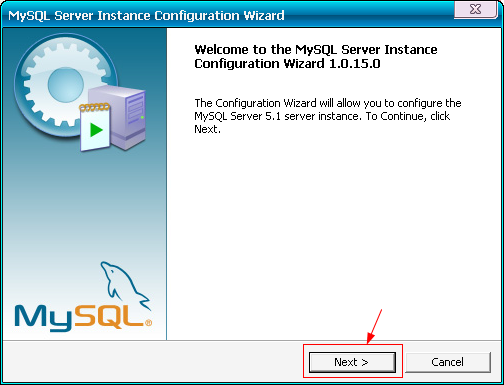


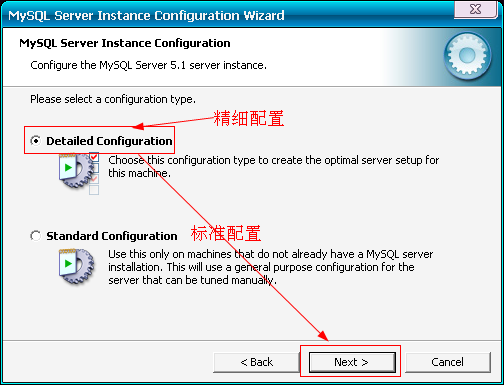


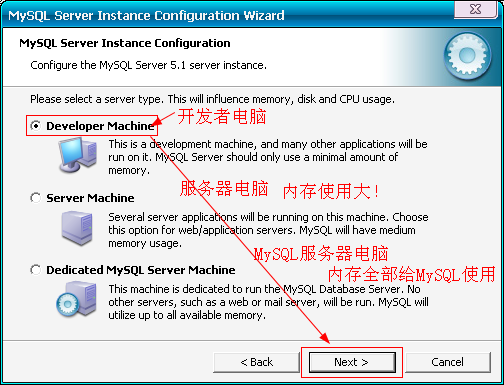


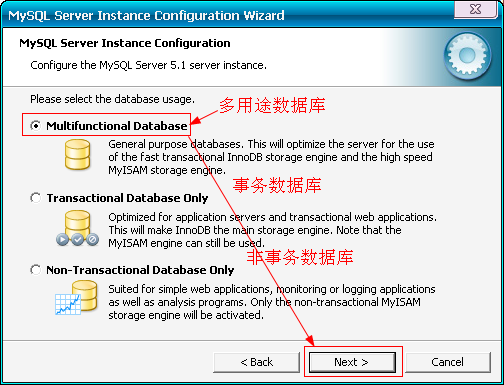


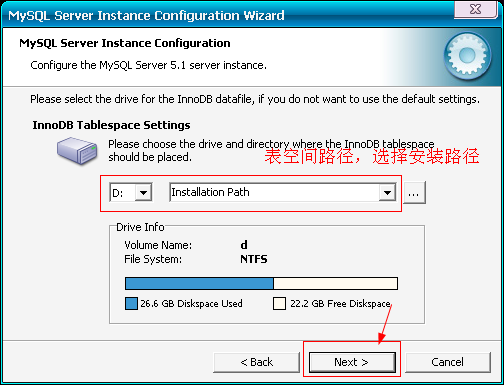


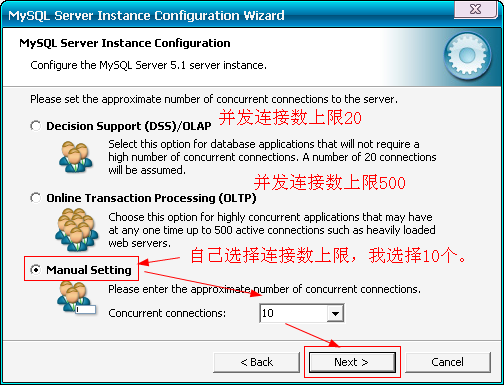


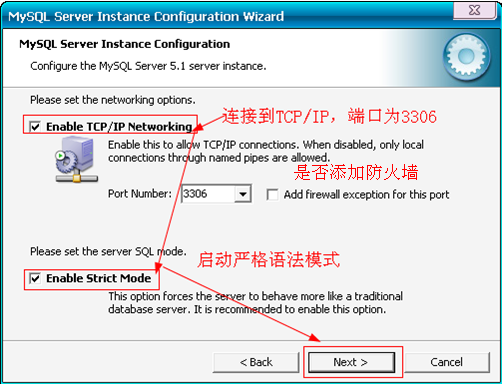


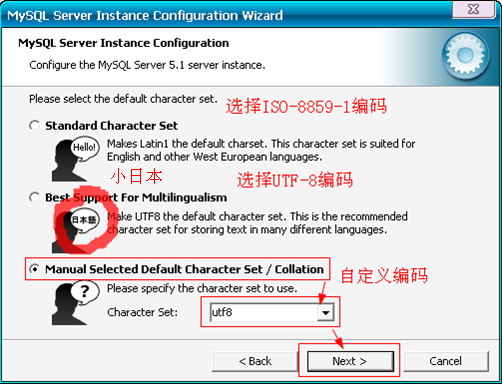


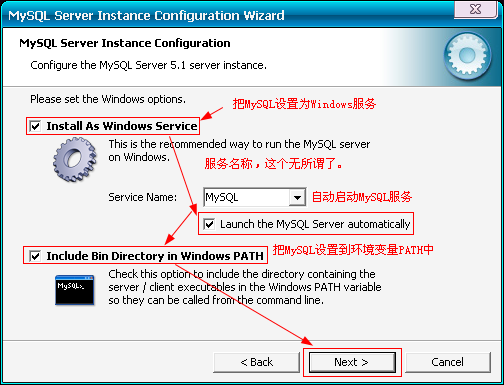




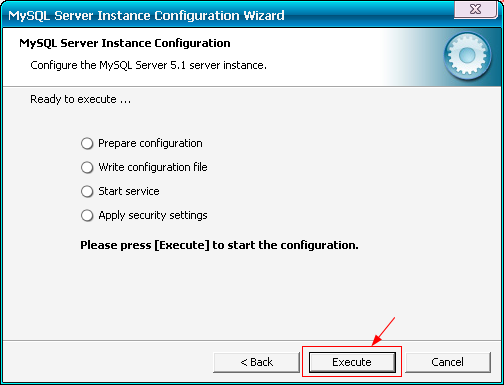


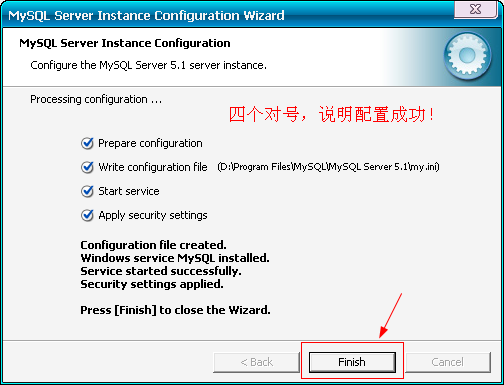






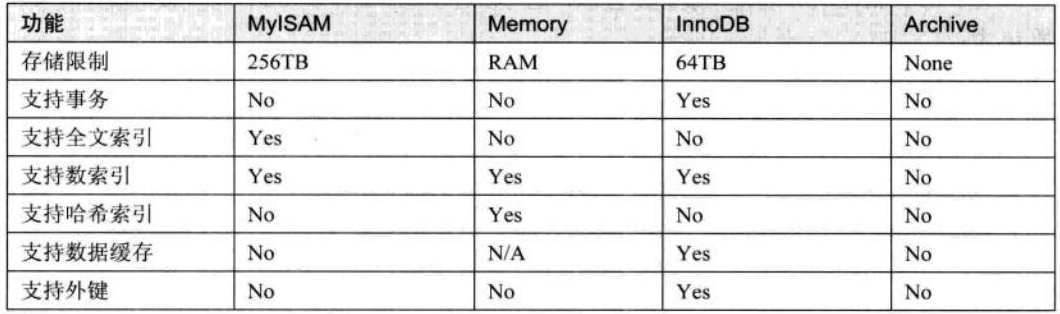








* 数据库存储引擎说明：



# 三、MYSQL数据库基本使用

## 1、登录MYSQL数据库

* 登录方式一：

在windows的command line下登录：

mysql -h 127.0.0.1 -u root -p

* 登录方式二：

使用mysql command line client登录。

## 2、MYSQL配置设置

* 使用图形用户界面的方式进行配置

在mysql的安装路径中，进入到bin目录，运行MySQLInstanceConfig.exe文件进入图形 配置界面进行配置。

* 在mysql的安装路径中，打开my.ini文件进行配置。**（了解）**

## 3、MYSQL基本命令

### 1)、查看系统中的数据库

show databases;

### 2)、切换数据库

use 数据库名称;

### 3)、查看数据库中的表

show tables;

### 4)、查看表结构

Desc 表名称；

查看建表详细结构

show crerate table 表单名称;

## 4、停止和启动MYSQL服务

Net stop mysql //停止服务

Net start mysql //启动服务

## 5、修改数据库密码（掌握）

步骤一：停止MYSQL服务

步骤二：在CMD下输入以下命令

mysqld--skip-grant-tables

步骤三：新打开cmd 输入mysql -u root -p不需要密码

步骤四：在CMD下输入如下两行命令：

use mysql;

update user set password=password(‘新密码’) WHERE User='root';

步骤五：关闭两个cmd窗口 在任务管理器结束mysqld 进程

步骤六：在服务管理页面 重启mysql 服务

## 6、卸载MYSQL

停止mysql服务

卸载mysql

找到mysql 安装目录下的 my.ini datadir="C:/ProgramData/MySQL/MySQL Server 5.5/Data/"

# 四、MYSQL图形用户工具使用

MYSQLWorkbench、phpMyAdmin、navicat

# 五、DDL语句

## 1、SQL语言分类

* **数据定义语句(DDL):**  DROP/CREATE/ALTER。
* **数据操作语言(DML):**INSERT、UPDATE、DELETE。
* **数据查询语句(DQL):SELECT**
* 数据控制语言(DCL):GRANT、REVOKE、COMMIT、ROLLBACK等语句。

## 2、创建数据库

CREATE DATABASE 数据库名称;

## 3、删除数据库

DROP DATABASE 数据库名称;

## 4、创建数据表

* 语法：

CREATE TABLE <表名>**(**

字段名1,数据类型[列级别约束条件][默认值]**,**

字段名1,数据类型[列级别约束条件][默认值]**,**

……

[表级约束条件] //**最后一条语句是不要使用,**

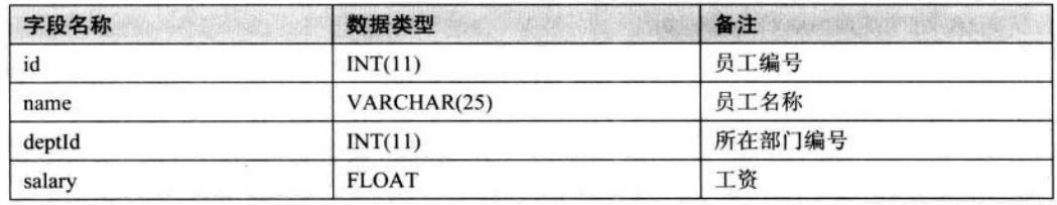
**)**

**使用注意：**

创建表时，表名称不区分大小写。

如果创建表中有多个字段，则字段之间应该使用逗号将其隔开。

例：建立如下的表



建表语句：

CREATE TABLE employee(

id INT(11),

name VARCHAR(25),

deptId INT(11),

salary FLOAT

);

#### 使用约束条件

##### 主键约束

要求主键列数据**唯一**，并且**不能为空**。主键约束分为两种：单字段主键和联合主键

###### 单字段主键

例：在以上的表中，在建表时为id添加主键约束。

建表语句：

CREATE TABLE employee2(

id INT(11) PRIMARY KEY,

name VARCHAR(25),

deptId INT(11),

salary FLOAT

);

或者：

CREATE TABLE employee3(

id INT(11),

name VARCHAR(25),

deptId INT(11),

salary FLOAT,

PRIMARY KEY(id)

);

或者：

ALTER TABLE employee3 ADD PRIMARY KEY(id);

###### 联合主键

语法：

PRIMARY KEY（字段1，字段2……）

例：使用如上表结构建立表，将name和deptId设置为联合主键

建表语句：

CREATE TABLE employee4(

id INT(11),

name VARCHAR(25),

deptId INT(11),

salary FLOAT,

PRIMARY KEY(name,deptId)

);

##### 外键约束(重点内容)

* 外键约束一般指的是在两个表之间建立链接，可以是一列或者是多列，**一个表可以有一个或者多个外键**。
* 首先他可以不是本表的主键，但是对应的另外一个表的主键，主要作用是用来保证数据的完整性。
* 主表：两个表外键关联关系中，主键所在的那个表叫主表。
* 从表：两个表外键关联关系中，外键所在的表叫做从表。

使用语法规则：

CONSTRAINT 外键名 FOREIGN KEY 字段1[,字段2,...]

REFERENCES <主表名> 主键列1[,主键2,...]

**例：**按照如下表格结构要求建立表

Employee表：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 数据类型 | 约束条件 | 描述 |
| id | int(11) | 主键 |  |
| EmpName | varchar(25) |  |  |
| deptId | int(11) | 外键 |  |
| salar | float |  |  |

Deptment表：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 数据类型 | 约束条件 | 描述 |
| id | int(11) | 主键 |  |
| deptName | varchar(25) |  |  |
| Location | Varchar(50) |  |  |

建表语句如下：

CREATE TABLE dept5(

id INT(11) PRIMARY KEY,

deptName VARCHAR(25),

location VARCHAR(50)

);

CREATE TABLE employee5(

id INT(11) PRIMARY KEY,

empName VARCHAR(25),

deptId INT(11),

salar FLOAT,

**CONSTRAINT fk\_employee5\_dept5 FOREIGN KEY (deptId) REFERENCES dept5(id)**

);

-- 单独设置表的外键

ALTER TABLE employee5 ADD FOREIGN KEY(deptId) REFERENCES dept5(id);

##### 使用非空约束

指字段不能为空。如果用户在指定了非空约束，用户在输入值的时候没有输入，则系统会报错。

**语法：**

字段名称 字段类型 not null;

Employee表：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 数据类型 | 约束条件 | 描述 |
| id | int(11) | 主键 |  |
| EmpName | varchar(25) | **非空 约束** |  |
| deptId | int(11) | 外键 |  |
| salar | float |  |  |

建表语句：

CREATE TABLE employee6(

id INT(11) PRIMARY KEY,

empName VARCHAR(25) NOT NULL,

deptId INT(11),

salary FLOAT

);

##### 唯一约束

唯一约束要求该列唯一，**允许为空，但是只能出现一个空值。**

语法：

* 在定义完列后直接指定唯一约束。

字段名 字段数据类型 UNIQUE;

例：

CREATE TABLE employee7(

id INT(11) PRIMARY KEY,

empName VARCHAR(25) **UNIQUE,**

deptId INT(11),

salary FLOAT

);

* 在定义完后指定唯一约束。

CONSTRAINT 约束名 UNIUQE 字段名；

例：

CREATE TABLE employee8(

id INT(11) PRIMARY KEY,

empName VARCHAR(25),

deptId INT(11),

salary FLOAT,

**CONSTRAINT uq\_empName UNIQUE(empName)**

);

##### 默认约束

在一个字段指定了默认值的约束条件后，在一条记录增加的时候，如果没有为该字段输入值，则系统会按照默认约束给定一个指定值。

语法：

字段名 数据类型 DEFAULT 默认值；

例：

CREATE TABLE employee9(

id INT(11) PRIMARY KEY,

empName VARCHAR(25),

deptId INT(11),

salary FLOAT DEFAULT 0

);

##### 自动递增

自动递增约束条件，系统会自动为该字段创建值，可以不 需要用户进行维护。

语法：

字段名 数据类型 auto\_increment;

例：

CREATE TABLE employee10(

id INT(11) PRIMARY KEY **AUTO\_INCREMENT**,

empName VARCHAR(25),

deptId INT(11),

salary FLOAT DEFAULT 0

);

## 5、修改数据表

### 修改表名称

语法：

ALTER TABLE 旧表名称 RENAME 新表名称；

ALTER TABLE employee RENAME emp;

### 修改字段的数据类型

语法：

ALTER TABLE 表名 MODIFY 字段名称 新数据类型；

**ALTER TABLE emp MODIFY salary INT(11);**

### 修改字段名称

语法：

ALTER TABLE 表名 **CHANGE** 旧字段名称 新字段名称 新数据类型；

ALTER TABLE emp CHANGE name empName VARCHAR(20);

### 添加字段

语法：

ALTER TABLE 表名 ADD 新字段名称 数据类型

[约束条件 FIRST|AFTER 存在字段名称]

注意：在使用FIRST的时候只能单一的使用，后面不需要添加已经存在的字段名称，表示添加在第一列。

ALTER TABLE emp ADD address VARCHAR(50); //默认添加在了最后的位置。

ALTER TABLE emp ADD tel VARCHAR(11) FIRST; //添加在第一个字段

ALTER TABLE emp ADD sex VARCHAR(2) AFTER empName;//添加在empName后的字段

## 6、删除字段

语法：

ALTER TABLE 表名 DROP 字段名；

ALTER TABLE emp DROP sex;

## 7、删除外键约束(解除关联关系)

语法：

ALTER TABLE 表名 DROP FOREIGN KEY 外键约束名称；

ALTER TABLE employee5 DROP FOREIGN KEY fk\_employee5\_dept5;

## 8、删除数据表

语法：

DROP TABLE [IF EXISTS] 表名1，表名2……;

注意：如果要删除两个建立了外键关系的表

**首先接触表之间的关联关系，然后再删除表。**

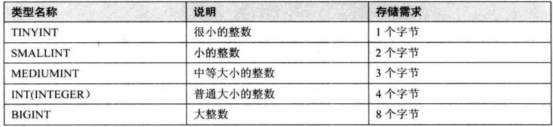
**要先删除从表，再删除主表。**

DROP TABLE emp; //删除数据表，慎重使用。

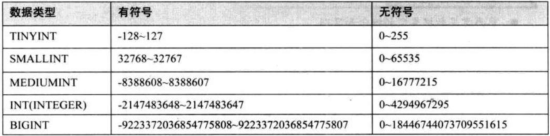
# 六、MYSQL中的数据类型

## 整数类型

**数据类型如下表所示：**

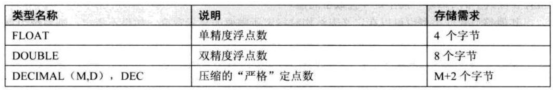


**取值范围如下表所示：**

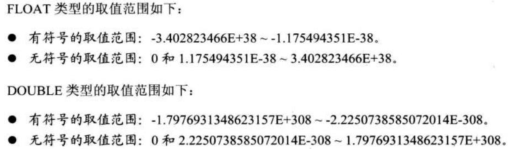


## 浮点类型和定点数据类型

**数据类型如下表所示：**

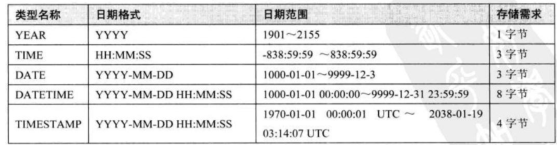


**取值范围：**



## 日期和时间类型

数据类型如下表所示：



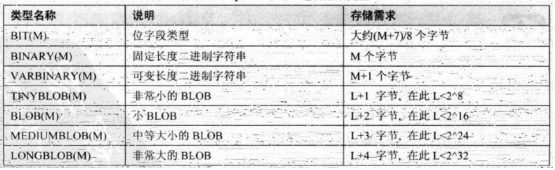
## 字符串类型

字符串类型如下图所示：



## 二进制类型

二进制类型如下表所示：



# 七、运算符

算术运算符如下表所示：



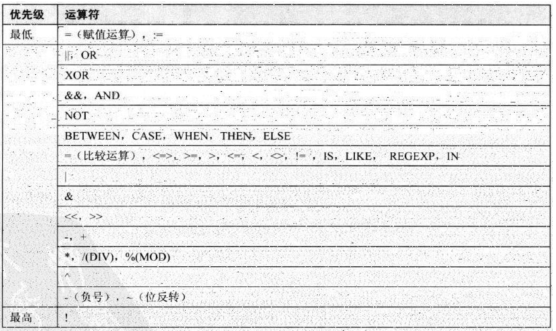
关系运算符如下表所示：



逻辑运算符如下表所示：



各类运算符优先级别如下表所示：



# 八、DQL语句

## 1、表说明

EMP（员工表）：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 类型 | 约束条件 | 说明 |
| EMPNO | Int（11） | 主键 | 雇员编号 |
| ENAME | Varchar(10) |  | 雇员姓名 |
| JOB | Varchar(9) |  | 雇员职位 |
| MGR | Int(11) | 外键 | 雇员领导编号 |
| HIREDATE | Date |  | 雇佣日期 |
| SAL | Double |  | 月薪 |
| COMM | Double |  | 奖金 |
| DEPTNO | Int(11) | 外键 | 部门编号 |

建表语句：

CREATE TABLE Emp(

empNo INT(11) PRIMARY KEY,

eName VARCHAR(10),

job VARCHAR(9),

mgr INT(11),

hiredate DATE,

sal DOUBLE,

comm DOUBLE,

deptNo INT(11),

CONSTRAINT fk\_emp FOREIGN KEY (mgr) REFERENCES emp(empNo),

CONSTRAINT fk\_emp\_dept FOREIGN KEY (deptNo) REFERENCES dept(deptNo)

);

dept（部门表）:

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 类型 | 约束条件 | 说明 |
| DEPTNO | Int(11) | 主键 | 部门编号 |
| DNAME | Varchar(14) |  | 部门名称 |
| LOC | Varchar(13) |  | 部门地址 |

建表语句：

CREATE TABLE Dept(

deptNo INT(11) PRIMARY KEY,

dName VARCHAR(14),

loc VARCHAR(13)

);

Salgrade（工资等级表）：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 类型 | 约束条件 | 说明 |
| GRADE | Int（11） |  | 工资等级 |
| LOSAL | double |  | 此等级最低工资 |
| HISAL | double |  | 此等级最高工资 |

建表语句：

CREATE TABLE Salgrade(

grade INT(11),

losal DOUBLE,

hisal DOUBLE

);

Bonus（奖金表）：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 类型 | 约束条件 | 说明 |
| ENAME | Varchar（10） |  | 雇员姓名 |
| JOB | Varchar（9） |  | 雇员工作 |
| SAL | double |  | 雇员工资 |
| COMM | double |  | 雇员奖金 |

建表语句：

CREATE TABLE Bonus(

ename VARCHAR(10),

job VARCHAR(9),

sal DOUBLE,

comm DOUBLE

);

## 2、简单查询

语法格式：

SELECT \* |  

FROM  ;

要求：查询全部：

SELECT \* FROM emp; //\* ，通配符，返回查询表中所有的字段。

要求：查询所有雇员的编号、姓名、工作。

SELECT empNo,ename,job FROM emp;

要求：在以上的查询中为每列添加一个别名。

SELECT empNo 编号,ename 姓名,job 工作 FROM emp;

SELECT empNo AS 编号,ename AS 姓名,job AS 工作 FROM emp;//推荐使用方式

要求：查询工作列

SELECT job FROM emp;

在以上的查询结果中，有重复字段出现，如果要消除查询的重复结构可以使用DISTINCT关键字消除重复记录。

语法：

SELECT {DISTINCT} \* |   FROM 

SELECT DISTINCT job FROM emp;

如果执行的是以下的SQL语句：

SELECT DISTINCT EMPNO,job FROM emp;

发现JOB仍然重复，是因为部门编号没有重复，所以DISTINCT关键字是消除重复的**记录**。

要求：查询出每个雇员的年薪及姓名

SELECT ename,**sal\*12** AS income FROM emp;

**注意：在查询语句中可以使用四则运算。**

## 3、限定查询(条件查询)

所谓限定查询就是根据制定的制定条件，讲结果查询出来。

语法：



要求：查出工资大于1500的所有员工信息。

分析：查询条件是sal > 1500

SELECT \* FROM emp

**WHERE SAL > 1500;**

要求：查询出可以拿到奖金的雇员信息。

分析：如果一个字段有内容就不空(null)，如果存在内容就会显示具体的值。

**不为空表示：字段名称 IS NOT NULL**

SELECT \* FROM emp

**WHERE COMM IS NOT NULL;**

同理，如果要显示出拿不到奖金的员工信息。

SELECT \* FROM emp

**WHERE COMM IS NULL;**

要求：要求查询出工资大于1500，同时可以领取奖金的雇员信息。

分析：在条件如果存在多个条件可以使用逻辑运算符进行连接，AND逻辑与，OR逻辑活。

SELECT \* FROM emp

WHERE SAL > 1500 AND COMM IS NOT NULL;

同理，如果要求要求查询出工资大于1500，或可以领取奖金的雇员信息。

SELECT \* FROM emp

WHERE SAL > 1500 OR COMM IS NOT NULL;

要求：要求查询出，基本工资不大于1500，同时不可以领取奖金的雇员信息。

分析：如果要求条件取反可以使用逻辑运算符中NOT运算符。

SELECT \* FROM emp

WHERE NOT(SAL > 1500 OR COMM IS NOT NULL);

要求：查询基本工资大于1500，但是小于3000的全部员工信息。

分析：sal >1500 and sal < 3000

SELECT \* FROM emp

WHERE sal > 1500 and sal < 3000;

说明：在以上的条件中是一个范围，在SQL中可以也可以使用指定范围的语句**BETWEEN…AND…**

使用语法格式：（**包含了最小值和最大值**）

 BETWEEN  AND 

以上语句改写：

SELECT \* FROM emp

WHERE sal BETWEEN 1500 AND 3000;

要求：查询出在1981年雇佣的雇员信息。

分析：在1981年雇佣员工的是在1981-01-01至1981-12-31，从数据中可以看到hiredate日期的格式为：年-月-日。此时在书写条件时，可以采用日期格式字符串。

SELECT \* FROM emp

WHERE HIREDATE BETWEEN '1981-01-01' AND '1981-12-31';

要求：查询出姓名时’SMITH’的员工信息。

SELECT \* from emp

WHERE ename='SMITH';

同时运行一下语句：

SELECT \* from emp

WHERE ename='smith';

也同样可以看到结果，**说明在MySQL中时不区分大小写的。**

要求：查询出雇员编号时7396,7499,7521的雇员基本信息。

分析：可以使用or运算符进行条件的连接。

SELECT \* FROM emp

WHERE EMPNO=7396 OR EMPNO=7499 OR EMPNO=7521;

在查询中除了可以查询范围，也可以使用**IN运算符**表示判断是否在一个范围内

语法：

值 IN（值1，值2，…） //判断是否在指定的范围

如果判断内容不在某个范围可以使用以下语法：

值 NOT IN（值1，值2，…） //判断是否在指定的范围

以上案例进行改写：

SELECT \* FROM emp

WHERE EMPNO IN (7396,7499,7521);

如果要查询出不是7396,7499,7521的雇员基本信息

SELECT \* FROM emp

WHERE EMPNO NOT IN (7396,7499,7521);

要求：要求查询出姓名是’SMITH’,’ALLEN’,’KING’的雇员信息。

SELECT \* FROM emp

WHERE ENAME in ('SMITH','ALLEN','KING');

**通过以上案例说明，in运算符也可以用在字符串上。**

要求：查询出所有雇员姓名中第二个字母包含了”M”的雇员信息。

分析：

在SQL中进行字符串匹配可以使用**LIKE运算符。**

在进行字符串进行匹配的时候可以使用如下两种通配符：

**“%”：表示任意长度的内容。**

**“\_”：表示一个长度的任意内容。**

分析：条件表单式’\_M%’

SELECT \* from emp

WHERE ENAME LIKE '\_M%';

注意：在LIKE运算符使用过程中，如果没有指定关键字，则表示查询全部。

SELECT \* from emp

WHERE ENAME LIKE '%';

要求： 查询出在1981年雇佣的所有员工信息。

分析：

LIKE运算符也可以使用在日期中。

分析：条件表达式：”1981%”

SELECT \* from emp

WHERE **HIREDATE LIKE '1981%';**

要求：查询出雇员编号不是7369的所有的雇员信息。

分析：

在SQL中的条件表达式中可以使用条件运算符，>,<,>=,<=,**<>,!=**

**<>,!=都可以表示不等于符号。**

SELECT \* FROM emp

WHERE EMPNO **!=** 7369;

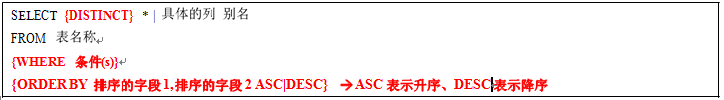
SELECT \* FROM emp

WHERE EMPNO **<>** 7369;

## 4、排序查询结果

在SQL中可以使用ORDER BY进行查询结果排序。

语法：



要求：按照工资由低到高进行排序。

SELECT \* FROM emp

ORDER BY SAL ASC;

SELECT \* FROM emp

ORDER BY SAL;

**注意：**运行结果，发现结果是一样的，说明在没有指定排序规则情况下，**默认使用升序方式进行结果排序**。

要求：按照工资由高到低进行排序。

SELECT \* FROM emp

ORDER BY SAL DESC;

要求：查询出10部门中所有雇员信息，查询信息按照工资由高到低排序，如果工资相等，则按照编号由低到高排序。

分析：

以上要求中存在两个排序条件。如果存在两个排序条件，就是存在所谓第一关键字和第二关键字排序，两个关键字中可以使用”,”进行分隔。

SELECT \* FROM emp

WHERE DEPTNO=10

ORDER BY SAL DESC , EMPNO ASC;

## 5、单行函数

### 1）、字符函数

要求：将所有的小写字符转化成为大写字母。

SELECT **UPPER('smith')**

FROM DUAL;

注意：

在以上的语句中，UPPER(字符串)是将小写转化为大写的字符函数。

DUAL是MYSQL中存在的一个伪表。

同理，在MYSQL中存在一个LOWER(字符串)函数是将大写转化为大写。

SELECT **LOWER('SMITH')**

FROM DUAL;

注意：在MYSQL中还存在**lcase(), ucase()**：函数也可以进行字符大小写转化，但是建议使用**UPPER(字符串)，LOWER(字符串)函数，这样可以提高数据库兼容性。**

要求：将”hello”,”world”字符串进行连接。

分析：

在MYSQL中使用**CONCAT(**字符串1，字符串2)函数进行字符串连接。

SELECT CONCAT('hello','world')

FROM DUAL;

要求：将每个雇员姓名的长度输出。

分析：

求字符串长度的函数为LENGTH(字符串)。

SELECT LENGTH('ucai')

FROM DUAL;

**//注意区分：空串（null），空白串（包含了一个或者多个空格的字符串） 0长度的字符串(‘’)**

要求：要求显示所有雇员姓名及姓名后的三个字符。

分析：

求字串函数SUBSTR(字符串，起始位置，长度)。

SELECT ENAME,SUBSTR(ENAME,LENGTH(ENAME)-2,3)

FROM emp;

注意：

如果SUBSTR函数中“长度”参数没有，表示截取到最后一个字符。

SELECT ENAME,SUBSTR(ENAME,LENGTH(ENAME)-2)

FROM emp;

SUBSTR如果要反向截取，可以“起始位置”参数使用负数。

SELECT ename,SUBSTR(ename,-3,3)

FROM emp; // 反向截取

要求：显示所有雇员姓名前三位字符。

分析：

在MYSQL中如果从左边第一个字符开始截取指定长度的字符串可以使用**LEFT(字符串，长度)函数，同理也存在RIGHT(字符串，长度)函数**

SELECT ename,LEFT(ENAME,3)

FROM emp;

### 2）、数值函数

要求：四舍五入函数

分析：

MYSQL中四舍五入函数**ROUND(浮点数值)**

SELECT ROUND(756.578)

FROM DUAL;

要求：四舍五入保留两位小数。

分析：

保留制定的小数位函数**ROUND(浮点数，保留小数位数)**。

SELECT ROUND(756.578**,2)**

FROM DUAL;

注意：如果“保留小数位数”为负数，则是对整数位进行四舍五入

SELECT ROUND(756.578**,-2)**

FROM DUAL;

要求：**截断**保留两位小数

分析：

TRUNCATE(浮点类型数值,保留小数位)函数实现截断数值（直接去除）

SELECT TRUNCATE(756.578,2)

FROM DUAL;

SELECT TRUNCATE(756.578,-2)

FROM DUAL;

**// TRUNCATE是只能使用两个参数，如果使用的是一个参数，就会报错。**

要求：四舍五入，并且整数超过三位，使用“，”显示。

分析：

可以使用**FORMAT(浮点类型数值，小数位数)**来实现。

SELECT FORMAT(1254.768,2)

FROM DUAL; //一般是在货币上的显示会比较多。1，254.77

要求：模运算

分析：

MYSQL中模运算函数MOD(数值1，数值2)。//10%3

SELECT MOD(10,3)

FROM DUAL;

### 3）、日期函数

要求：显示10部门中雇员进入员工的星期数。

分析：

在MYSQL中获取系统当前日期时间函数如下：

NOW():日期+时间

SYSDATE(): 日期+时间

CURDATE():日期

CURTIME():时间

在日期类型数据中，可以进行算数运算，如下：

 –  = 

 +  = 

 –  = 

SELECT ename,ROUND((CURDATE() - HIREDATE)/7)

FROM emp

WHERE DEPTNO = 10;

注意：在MYSQL中还存在以下日期函数，方便用户操作：

dayofweek(),返回在一星期中的位置。 //返回数值

dayofmonth(),返回在一个月中的位置 //返回数值

dayofyear()返回在一年中的为位置。 //返回数值

last\_day(),返回指定月份中的制定一天 //返回日期

要求：将“08/09/2008”格式字符串转化为日期。

在MYSQL中日期的格式字符串，我们可以使用一些常量来表示日期成分：

**m:表示月份**

**d:表示日期**

**Y:表示年份**

**h:表示小时**

**i:表示分钟**

**s:表示秒数**

分析：

可以使用**str\_to\_date**(日期字符串，格式字符串)，进行转化。

SELECT STR\_TO\_DATE('08/09/2008','%m/%d/%**Y**') //大写字符Y表示年份

FROM DUAL;

要求：将“08.09.2008 08:09:30”转化为日期。

SELECT STR\_TO\_DATE('08.09.2008 08:09:30','%m.%d.%Y %h:%i:%s')

FROM DUAL;

注意：date\_format(日期,格式字符串)函数可以将日期格式转化为制定格式字符串。

## 练习：

1、查询部门30中所有的员工。

SELECT \* FROM emp

WHERE DEPTNO=30;

2、查询出所有CLERK的姓名，编号和部门。

SELECT ENAME,EMPNO,DEPTNO FROM emp

WHERE JOB='CLERK';

3、查询出所有佣金高于薪金的员工。

SELECT \* FROM emp

WHERE COMM > SAL;

4、查询出佣金高于薪金60%的员工。

SELECT \* FROM emp

WHERE COMM > (SAL\*0.6);

5、查询出**部门10**中所有的**经理**（MANAGER）**和** 部门20中所有的办事员(CLERK)的资料。

SELECT \* FROM emp

WHERE (DEPTNO=10 AND JOB='MANAGER') OR (DEPTNO=20 AND JOB='CLERK');

6、找出部门10中所有的经理（MANAGER）,部门20中所有办事员(CLERK),即不是经理又不是办事员，但其薪金大于或等于2000的所有员工的详细资料。

SELECT \* FROM emp

WHERE ((DEPTNO=10 AND JOB='MANAGER')

OR (DEPTNO=20 AND JOB='CLERK'))

OR ((JOB NOT IN('MANAGER','CLERK'))

AND SAL >=2000);

7、找出收取佣金的员工不同工作。

SELECT DISTINCT JOB FROM EMP

WHERE COMM IS NOT NULL;

8、找出不收取佣金或者收取佣金低于100的员工

SELECT \* FROM EMP

WHERE COMM IS NULL OR COMM<100;

9、找出各月倒数3天被雇佣的员工。

SELECT \* FROM emp

WHERE HIREDATE = LAST\_DAY(HIREDATE)-2;

10、查询出名字正好为5个字符的员工姓名。

SELECT \* FROM emp

WHERE LENGTH(ENAME) = 5;

11、查询出不带有”R”的员工姓名。

SELECT \* FROM emp

WHERE ename NOT LIKE '%R%';

12、显示所有员工的姓名，将”a”替换成所有的”Z”

提示：字符串替换函数REPLACE(字符串，原字符串，目标字符串)

SELECT REPLACE(ENAME,'A','Z') FROM emp;

13、显示员工详细资料，按将入职最早的员工排在最前面。

SELECT \* FROM emp ORDER BY HIREDATE ASC;

14、显示所有员工的姓名、工作和薪金，按工作的降序排序，若工作相同按照薪金排序。

SELECT ename,job,sal

FROM emp

ORDER BY JOB DESC , sal;

15、显示在一个月为30天的情况所有员工的日薪金，忽略余数。

SELECT ename,ROUND(sal/30) FROM emp;

16、显示姓名字段的任何位置包含由”A”的所有员工姓名。

SELECT ename FROM emp

WHERE ENAME LIKE '%A%';

## 6、多表查询

如果在一次查询中，使用了一张以上的表，则就叫做多表查询。

语法：



要求：同时查询EMP表和DEPT表。

select \* from emp,dept;

以上的查询中使用了两张表，但是返回的记录是56条记录。但是emp表是14条记录，dept表是4条记录。所以通过以上的查询结果我们知道多表查询的时候会差生**笛卡尔积**。如果要出去笛卡尔积就需要进行字段关联的操作。

select \* from emp,dept

**WHERE emp.DEPTNO = dept.DEPTNO;**

注意：在使用多表查询的时候，为了方便书写，我们可以为查询表添加别名：

select \* **from emp e,dept p**

**WHERE e.DEPTNO = p.DEPTNO;**

要求：查询出雇员的编号、雇员的姓名、部门编号、部门的名称及位置。

分析：

雇员的编号、雇员的姓名、部门编号在emp表

部门的名称及位置在DEPT表中

所以应该要使用多表查询进行记录的字读整合。

SELECT e.empno , e.ename, e.deptno , d.dname,d.loc

FROM emp e , dept d

WHERE e.DEPTNO=d.DEPTNO;

要求：查询出每个雇员的姓名、工作、直接上级领导的姓名。

分析：在EMP表中MGR字段是用来表示直接上级领导的编号。**所以可以将EMP表和自身进行关联进行多表查询。**

SELECT e.ename,e.job,p.ename

FROM emp e,emp p

W*HERE e.MGR = p.EMPNO;*

要求：在以上的查询中，将雇员所在部门的名称同时列出。

分析：部门名称在DEPT表中。

SELECT e.ename,e.job,p.ename,d.dname

FROM emp e,emp p ,dept d

WHERE e.MGR = p.EMPNO AND e.deptno = d.deptno;

要求：查询出每个雇员的姓名、工资、部门名称，工资在公司的等级，及领导的姓名及工资所在公司的等级。

分析：

步骤一：先确定工资等级表中的内容

SELECT \* from salgrade;

步骤二：查询出每个雇员的姓名、工资、部门名称、工资在公司的等级。

SELECT e.ENAME,e.sal,d.DNAME,s.GRADE

FROM emp e,dept d,salgrade s

WHERE

e.DEPTNO = d.DEPTNO AND e.SAL BETWEEN s.LOSAL AND s.HISAL;

步骤三：领导的姓名及工资所在公司的等级

SELECT e.ENAME,e.sal,d.DNAME,s.GRADE,m.ename,m.sal,ms.grade

FROM emp e,dept d,salgrade s,emp m,salgrade ms

WHERE

e.DEPTNO = d.DEPTNO AND e.SAL BETWEEN s.LOSAL AND s.HISAL

AND e.mgr = m.empno and m.SAL BETWEEN ms.LOSAL AND ms.HISAL;

要求：将emp表和dept表进行关联查询。

SELECT e.EMPNO,e.ENAME,e.DEPTNO,d.DNAME,d.LOC

FROM emp e,dept d

WHERE e.DEPTNO=d.DEPTNO;

以上的这种查询方式就是**内连接查询（99查询），**并不是SQL标准中的查询方式，可以简单理解为方言。SQL标准的内连接为：

SELECT 字段1，字段2,…

FROM 表1

**INNER JOIN 表2**

ON 条件

注意：ON子语句只能做主外键关联，不能做其他的条件书写。

SELECT e.EMPNO,e.ENAME,e.DEPTNO,d.DNAME,d.LOC

FROM emp e

INNER JOIN

dept d

on e.DEPTNO=d.DEPTNO;

**简单总结：**内连接实际就是在两个表中的关联字段都有对应值才会显示在查询结果中。

## 7、外连接查询（左、右连接）

查询出的结果不满足条件的可能。

### 左连接

左连接是先查询出左表（即以左表为主），然后查询右表，右表中满足条件的显示出来，不满足条件的显示NULL。

要求：将EMP表和DEPT表进行左连接查询。

SELECT e.ENAME,d.DEPTNO ,d.DNAME

FROM emp e

LEFT OUTER JOIN dept d

ON e.DEPTNO = d.DEPTNO;

在以上查询结果中，部门表中一共有四个部门信息，但是结果中只是列出来了3个部门，部门40并没有显示，**因为部门40表中并没有员工**。

### 右连接

右连接就是先把右表中所有记录都查询出来，然后左表满足条件的显示，不满足显示NULL。

SELECT e.ENAME,d.DEPTNO ,d.DNAME

FROM emp e

RIGHT OUTER JOIN dept d

ON e.DEPTNO = d.DEPTNO;

## 8、自然连接

连接查询会产生无用笛卡尔积，我们通常使用主外键关系等式来去除它。而自然连接无需你去给出主外键等式，它会自动找到这一等式

要求：将EMP表和DEPT表进行左连接查询。

分析：两张连接的表中名称和类型完全一致的列作为条件，例如emp和dept表都存在deptno列，并且类型一致，所以会被自然连接找到

SELECT \* FROM emp

NATURAL JOIN dept;

## 9、组函数分组统计

### 1）组函数

MYSQL中存在以下几个组函数

COUNT():求出全部记录总数

MAX():求出一组中最大的值

MIN():求出一组中最小的值

AVG():求平均值

SUM():求和

要求：查询EMP表中记录总数。

SELECT COUNT(\*) FROM emp;

要求：求出所有员工的最低工资。

SELECT MIN(sal) FROM emp;

要求：求出所有员工的最高工资。

SELECT MAX(sal) FROM emp;

要求：查询出部门20中的工资总和。

SELECT sum(sal) FROM emp

WHERE DEPTNO =20;

要求：查询出部门20的平均工资。

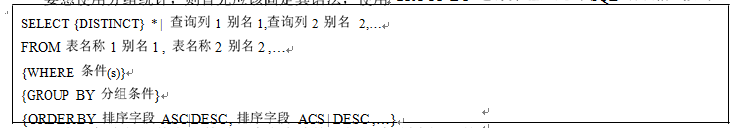
SELECT avg(sal) FROM emp

WHERE DEPTNO =20;

### 2）分组统计

分组统计需要使用**GROUP BY**进行分组。

语法：



要求：球部每个部门员工的数量。

SELECT deptno,count(EMPNO)

FROM emp

GROUP BY deptno;

要求：求出每个部门的平均工资。

SELECT deptno,AVG(sal)

FROM emp

GROUP BY deptno;

要求：按照部门分组，并显示部门名称，及每个部门的员工数量

SELECT d.DNAME,COUNT(e.EMPNO)

FROM emp e,dept d

**WHERE e.DEPTNO = d.DEPTNO**

GROUP BY e.DEPTNO;

要求：显示出平均工资大于2000的部门编号和平均工资。

分析：

条件是AVG(sal)>2000;

SELECT deptno,avg(sal)

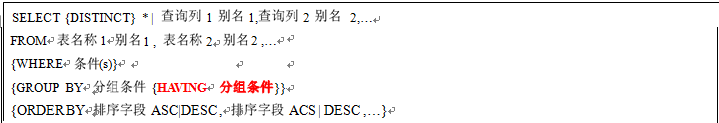
FROM emp

**WHERE avg(sal)>2000 //分组条件应该写在having子语句中，书写位置错了**

GROUP BY DEPTNO;

注意：以上的语句在运行过程中会出现错误，因为分组条件只能写在**HAVING子语句中**。

语法格式：



SELECT deptno,avg(sal)

FROM emp

GROUP BY DEPTNO

**HAVING avg(sal)>2000;**

要求：查询非销售人员的工作名称以及从事同一工作的雇员的的月工资综合，并且要满足从事同一工作的雇员月工资合计大于5000，输出按照月工资合计升序排序。

步骤一：

显示全部的非销售人员

SELECT \*

FROM emp

WHERE JOB<>'SALESMAN';

步骤二：按照工作分组，同事求出工资的总和。

SELECT JOB,SUM(sal)

FROM emp

WHERE JOB<>'SALESMAN'

GROUP BY JOB;

步骤三：

对分组条件进行限制，工资总和大于5000.

SELECT JOB,SUM(sal)

FROM emp

WHERE JOB<>'SALESMAN'

GROUP BY JOB

HAVING SUM(sal)>5000;

步骤四：使用排序。

SELECT JOB,SUM(sal) su

FROM emp

WHERE JOB<>'SALESMAN'

GROUP BY JOB

HAVING SUM(sal)>5000;

ORDER BY **su;**

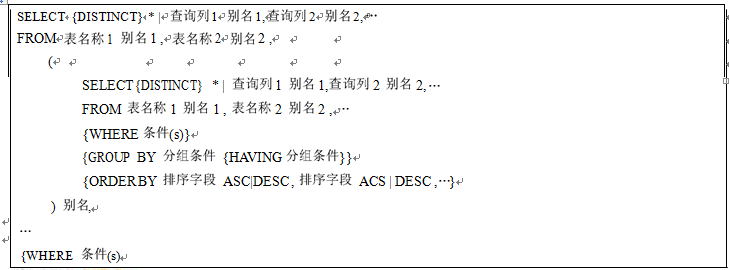
简单总结：

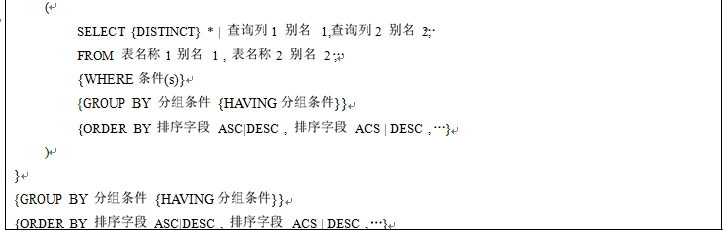
分组的原则:只要一列上存在重复的内容才可能考虑使用分组。

## 10、子查询

在一个查询内部中还包括另外一个查询，则称为子查询。

语法：





要求：查询出比7654工资还要高的全部员工信息。

分析：

首先要查询出7654雇员的工资。

SELECT sal FROM emp

where EMPNO=7654;

之后要以上查询结果作为条件，进行查询。

SELECT \* from emp

WHERE sal > (**SELECT sal FROM emp**

**where EMPNO=7654**);

要求：查询出工资比7654还要高，同时与7788从事相同工作的全部员工信息。

分析：

查询7654雇员的工资。

SELECT sal FROM emp WHERE empno=7654;

查询出7788雇员的额工作。

SELECT JOB FROM emp WHERE EMPNO=7788;

以上两个查询结果进行综合。

SELECT \* FROM emp

WHERE

sal > (SELECT sal FROM emp WHERE empno=7654)

AND

job = (SELECT JOB FROM emp WHERE EMPNO=7788);

要求：查询出工资最低的员工姓名、工作、工资

分析：

求出最低工资。

SELECT min(sal) FROM emp;

使用最低工资进行下一步查询。

SELECT \* FROM emp

WHERE sal = (SELECT min(sal) FROM emp);

**练习：**

要求查询出部门名称、部门员工数量、部门平均工资、部门最低收入的雇员姓名。

分析：

需要关联两张表DEPT\EMP

如果要查询出每个部门员工数量，平均工资，则要使用DEPTNO进行分组。

SELECT DEPTNO,COUNT(EMPNO),AVG(sal)

FROM emp

GROUP BY DEPTNO;

如果要查询出部门名称，则需要与DEPT表进行关联。

SELECT d.DNAME,ed.c,ed.a

FROM dept d,(SELECT DEPTNO,COUNT(EMPNO) c,AVG(sal) a

FROM emp

GROUP BY DEPTNO

) ed

WHERE d.DEPTNO = ed.DEPTNO;

查询出工资最低的员工姓名

SELECT d.DNAME,ed.c,ed.a,e.ENAME

FROM dept d,(SELECT DEPTNO,COUNT(EMPNO) c,AVG(sal) a,MIN(sal) min

FROM emp

GROUP BY DEPTNO

) ed,emp e

WHERE d.DEPTNO = ed.DEPTNO AND e.sal = ed.min;

注意：在子查询中存在以下三种查询操作符号。

**IN：表示一个查询指定范围。**

**ANY：**

**=ANY：与IN操作符的功能完全一致。**

**>ANY：比里面最小的还要大。**

**<ANY：比里面最大的值要小。**

**ALL：**

**>ALL：比里面最大的还要大。**

**<ALL：比里面最小的还要小。**

要求：求出每个部门的最低工资雇员信息。

SELECT \* FROM emp

WHERE sal IN (**SELECT min(sal) FROM emp GROUP BY DEPTNO**);

要求：=ANY

SELECT \* FROM emp

WHERE sal =Any(SELECT min(sal) FROM emp GROUP BY DEPTNO);

要求：>ANY

SELECT \* FROM emp

WHERE sal >Any(SELECT min(sal) FROM emp GROUP BY DEPTNO);

要求：<ANY

SELECT \* FROM emp

WHERE sal <Any(SELECT min(sal) FROM emp GROUP BY DEPTNO);

要求：>ALL

SELECT \* FROM emp

WHERE sal >all(SELECT min(sal) FROM emp GROUP BY DEPTNO);

要求：<ALL

SELECT \* FROM emp

WHERE sal <all(SELECT min(sal) FROM emp GROUP BY DEPTNO);

# 九、DML语句

## 1、添加数据

语法：



复制一张新表为newemp;

要求：在newemp表中插入一条数据。

INSERT INTO newemp

(empno,ename,job,mgr,hiredate,sal,comm,deptno)

VALUES

(7933,'zhangsan','CLERK',7902,'1982-01-24',2000,NULL,30);

在表中增加一个记录，只是部分字段：

INSERT INTO newemp

(empno,ename,job,mgr,sal,comm,deptno)

VALUES

(7936,'ls','CLERK',7902,2000,NULL,30);

如果添加的记录是所有的字段都包含了，我们可以采用如下的写法，（去除字读名称的书写部分。）

INSERT INTO newemp

VALUES

(7946,'ww','CLERK',7902,'1982-01-24',2000,NULL,30); //在开发的过程中是不提倡的。

## 2、修改数据

语法：



修改全部：

UPDATE newemp SET comm=1000; //实际开发的时候没有人会这么去用。

要求：将雇员表中7654员工的工资修改为5000

UPDATE newemp SET sal=5000 **WHERE empno=7654;**

## 3、删除数据

语法：



要求：删除编号为7899雇员的记录。

DELETE FROM newemp WHERE empno=7946;

# 练习

1、查询出至少有一个员工的所有部门

分析：

求出所有部门员工的数量

SELECT deptno,COUNT(empno) FROM emp

GROUP BY deptno ;

列出部门人数大于1的所有部门编号。

SELECT deptno,COUNT(empno) FROM emp

GROUP BY deptno **HAVING COUNT(empno)>1 ;**

通过部门表查询出部门信息。

**SELECT d.\*,ed.cou**

**FROM dept d**,(SELECT deptno,COUNT(empno) **cou** FROM emp

GROUP BY deptno HAVING COUNT(empno)>1) **ed**

**WHERE d.deptno=ed.deptno ;**

2、列出薪金比“SMITH“还要多的所有员工。

分析：

求出SMITH工资。

SELECT sal FROM emp WHERE ename='SMITH' ;

查询出符合条件的所有员工信息。

**SELECT \* FROM emp**

**WHERE sal>**(SELECT sal FROM emp WHERE ename='SMITH') ;

3、列出所有员工的姓名及直接上级的姓名。

SELECT e.ename,m.ename FROM emp e,emp m WHERE e.mgr=m.empno ;

4、列出受雇佣日期早与直接上级的所有员工编号、姓名、部门名称。

分析：

自身关联进行查询。

SELECT e.empno,e.ename

FROM emp e,emp m

WHERE e.mgr=m.empno AND e.hiredate<m.hiredate ;

使用DEPT表进行关联查询。

SELECT e.empno,e.ename,**d.dname**

FROM emp e,emp m**,dept d**

WHERE e.mgr=m.empno AND e.hiredate<m.hiredate **AND e.deptno=d.deptno ;**

5、列出部门名称和这些部门的员工信息，同事列出那些没有员工的部门。

SELECT d.deptno,d.dname,e.empno,e.ename

FROM dept d

LEFT OUTER JOIN emp e

ON d.deptno=e.deptno ;

6、列出所有的CLERK的姓名和部门名称及人数。

分析：

找出所有的办事员的姓名及部门呢编号。

SELECT ename,deptno

FROM emp

WHERE job='CLERK' ;

使用DEPT表，找出部门名称

SELECT **e.ename,d.dname**

FROM emp **e,dept d**

WHERE job='CLERK' **AND e.deptno=d.deptno ;**

使用分组统计部门人数

SELECT e.ename,d.dname,**ed.cou**

FROM emp e,dept d,(**SELECT deptno,COUNT(empno) cou FROM emp GROUP BY deptno) ed**

WHERE job='CLERK' AND e.deptno=d.deptno AND ed.deptno=e.deptno ;

7、查询出薪金高于公司平均薪金的所有员工、所在部门、上级领导、公司的工资等级。

分析：

求出公司平均工资。

SELECT AVG(sal) FROM emp ;

求出高于平均工资的所有雇员信息。

**SELECT \* FROM emp**

**WHERE sal>**(SELECT AVG(sal) FROM emp) ;

与DEPT表关联，查询所在的部门信息。

SELECT **e.\*,d.dname,d.loc**  FROM **emp e,dept d**

**WHERE sal**>(SELECT AVG(sal) FROM emp) **AND e.deptno=d.deptno ;**

与EMP表关联，查询上级领导。

SELECT e.empno,e.ename,m.empno,m.ename,d.deptno,d.dname,d.loc

FROM emp e,dept d,emp m

WHERE e.sal>(SELECT AVG(sal) FROM emp) AND e.deptno=d.deptno

AND e.mgr=m.empno ;

求出雇员工资等级。

SELECT e.empno,e.ename,s.grade,m.empno,m.ename,d.deptno,d.dname,d.loc

FROM emp e,dept d,emp m,salgrade s

WHERE e.sal>(SELECT AVG(sal) FROM emp) AND e.deptno=d.deptno

AND e.mgr=m.empno

AND e.sal BETWEEN s.losal AND s.hisal;

8、查询出与SCOTT从事相同工作的所有员工及部门名称。

分析：

找到SCOTT工作。

SELECT job FROM emp WHERE ename='SCOTT' ;

找出与其工作相同雇员。

SELECT empno,ename,job,sal

FROM emp

WHERE job=(SELECT job FROM emp WHERE ename='SCOTT') ;

以上查询结果中出现了SCOTT，除去SCOTT。

SELECT empno,ename,job,sal

FROM emp

WHERE job=(SELECT job FROM emp WHERE ename='SCOTT') AND ename!='SCOTT' ;

与部门表关联，查询出部门的名称。

SELECT e.empno,e.ename,e.job,e.sal,d.dname,d.loc

FROM emp e,dept d

WHERE job=(SELECT job FROM emp WHERE ename='SCOTT') AND ename!='SCOTT'

AND e.deptno=d.deptno ;

9、列出薪金高于在部门30工作的所有员工姓名和薪金和部门名称。

分析：

使用>ALL

SELECT ename,sal

FROM emp WHERE sal >ALL

(SELECT sal FROM emp WHERE deptno=30) AND deptno!=30 ;

与DEPT表关联，求出部门名称。

SELECT e.ename,e.sal,d.dname,d.loc

FROM emp e,dept d

WHERE e.sal >ALL (

SELECT sal FROM emp WHERE deptno=30) AND e.deptno!=30 AND e.deptno=d.deptno ;

10、列出各个部门的MANAGER的最低薪金。

SELECT deptno,MIN(sal) FROM emp

WHERE job='MANAGER'

GROUP BY deptno ;

# 十、MYSQL数据库的备份与恢复

## 1）备份数据库

在控制台使用mysqldump命令可以用来生成指定数据库的脚本文本，但要注意，脚本文本中只包含数据库的内容，而不会存在创建数据库的语句！所以在恢复数据时，还需要自已手动创建一个数据库之后再去恢复数据。

|  |
| --- |
| mysqldump –u用户名 –p密码 数据库名>生成的脚本文件路径 |



## 2）恢复数据库

**执行SQL脚本需要登录mysql**，然后进入指定数据库，才可以执行SQL脚本！！！

执行SQL脚本不只是用来恢复数据库，也可以在平时编写SQL脚本，然后使用执行SQL 脚本来操作数据库！大家都知道，在黑屏下编写SQL语句时，就算发现了错误，可能也不能修改了。所以我建议大家使用脚本文件来编写SQL代码，然后执行之！

|  |
| --- |
| SOURCE C:\ucai.sql |

注意：由于在我们生成SQL脚本中是没有建立数据库的SQL语句，建议是各位同学生成了你的脚本以后在文件的头部添加上如下的SQL语句：

DROP DATABASE IF EXISTS UCAI;

CREATE DATABASE UCAI;

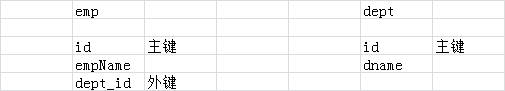
USE UCAI;

# 十一、表与表之间的关联关系

1、一对多的关联关系

例：emp dept

关系模型：



建表语句：

CREATE TABLE dept(

id INT(11) PRIMARY KEY,

dName VARCHAR(25)

);

CREATE TABLE emp(

id INT(11) PRIMARY KEY,

empName VARCHAR(25),

dept\_id INT(11),

CONSTRAINT fk\_emp\_dept FOREIGN KEY (dept\_id) REFERENCES dept(id)

)

2、一对一关联关系

例：person card

关系模型1：



建表语句：

CREATE TABLE person(

id INT(11) PRIMARY KEY,

name varchar(20)

);

CREATE TABLE card(

id INT(11) PRIMARY KEY,

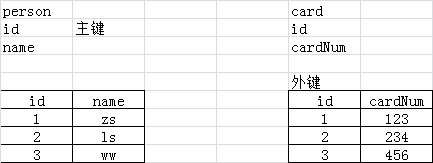
cardNum VARCHAR(16),

person\_id INT(11) UNIQUE,

CONSTRAINT fk\_card\_person FOREIGN KEY (person\_id) REFERENCES person(id)

);

关系模型2：



建表语句：

CREATE TABLE person(

id INT(11) PRIMARY KEY,

name varchar(20)

);

CREATE TABLE card(

id INT(11) PRIMARY KEY,

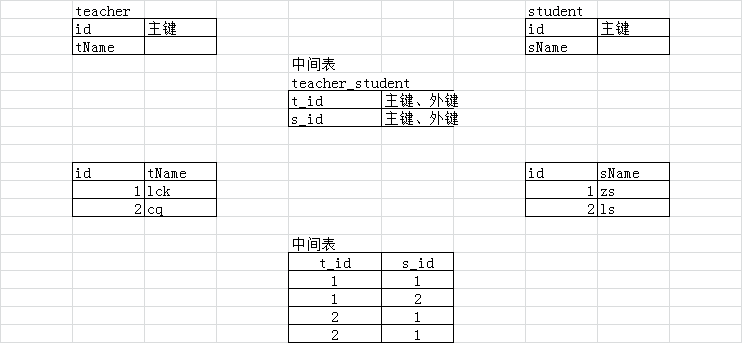
cardNum VARCHAR(16),

CONSTRAINT fk\_card\_person FOREIGN KEY (id) REFERENCES person(id)

);

3、多对多关联关系

例：Teacher student



建表语句：

CREATE TABLE teacher(

id INT(11) PRIMARY KEY,

tName VARCHAR(25)

);

CREATE TABLE student(

id INT(11) PRIMARY KEY,

sName VARCHAR(25)

);

CREATE TABLE teacher\_student(

t\_id INT(11) ,

s\_id INT(11) ,

PRIMARY KEY(t\_id,s\_id),

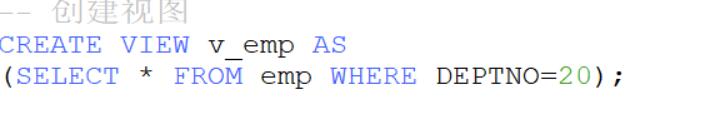
CONSTRAINT fk\_m\_teacher FOREIGN KEY (t\_id) REFERENCES teacher(id),

CONSTRAINT fk\_m\_student FOREIGN KEY (s\_id) REFERENCES student(id)

);

# 十二、视图

自己百度去

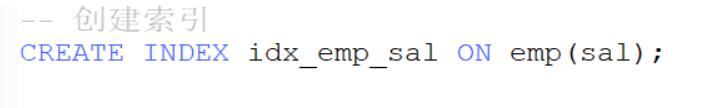


Create view 视图名 as sql语句;

为常使用到的表中字段创建一个视图，使用视图和表一样

# 十三、索引

自己百度去



**索引的作用：**当数据库中的数据非常大时，查找某张表某字段效率低，为常使用到的表和字段创建索引，提高效率。

# 十四、事务和保存点

自己百度去

