第06天 MySQL

**【学习目标】理解、了解、应用、记忆**

通过今天的学习，参训学员能够：（解释的时候说出二级目标的掌握程度）

1. **【理解】多表关系实战练习**
2. 【应用】独立编写SQL实现省市之间的一对多关系
3. 【应用】独立编写SQL建立用户与角色之间的多对多关系
4. 【应用】独立编写SQL建立角色与权限之间的多对多关系
5. 【应用】独立编写SQL建立客户与联系人之间的多对多关系
6. **【应用】多表查询操作**
7. 【应用】独立编写SQL录入多表查询的准备数据
8. 【应用】独自编写SQL使用内连接查询出哪些商品的分类已经上架
9. 【应用】独立编写SQL使用外连接统计所有分类下商品个数
10. 【应用】独立编写SQL完成相关子查询的操作
11. **【应用】SQL查询练习**
12. 独立完成单表查询的练习
13. 独立完成多表查询的练习

# 多表关系实战练习

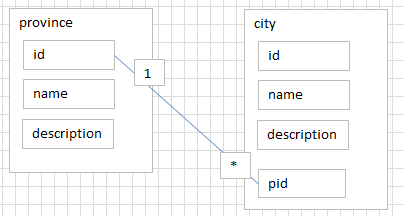
## 多表关系--实战1-- 省和市

### 需求分析

在数据库中建立表关系，一个省对应多个市，因此省和市在数据库中的关系是一对多

### 方案1：

* 多张表，一对多

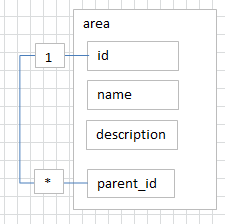


### 案例练习一

|  |
| --- |
| #实战1：省和市  ###方案1  ##创建省表  CREATE TABLE province(  id VARCHAR(32) PRIMARY KEY,  NAME VARCHAR(50),  description VARCHAR(100)  );  ##创建市表  CREATE TABLE city(  id VARCHAR(32) PRIMARY KEY,  NAME VARCHAR(50),  description VARCHAR(100),  pid VARCHAR(32) #对应省表 外键  );  ##主外键关系  ALTER TABLE city ADD CONSTRAINT city\_fk FOREIGN KEY (pid) REFERENCES province(id); |

### 方案2：

* 一张表，自关联一对多



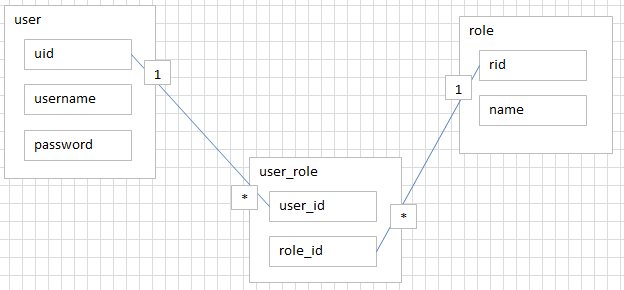
### 案例练习二

|  |
| --- |
| ###方案2：  CREATE TABLE AREA(  id VARCHAR(32) PRIMARY KEY,  NAME VARCHAR(50),  description VARCHAR(100),  parent\_id VARCHAR(32)  );  ##自关联一对多  ALTER TABLE AREA ADD CONSTRAINT area\_fk FOREIGN KEY (parent\_id) REFERENCES AREA(id); |

## 多表关系--实战2--用户和角色

### 需求分析

系统中，一个用户对应多个角色，比如张三用户可以是管理员角色，可以是普通用户角色；一个角色可以分配个多个用户。因此在表中建立用户与角色的多对多关系如下图。



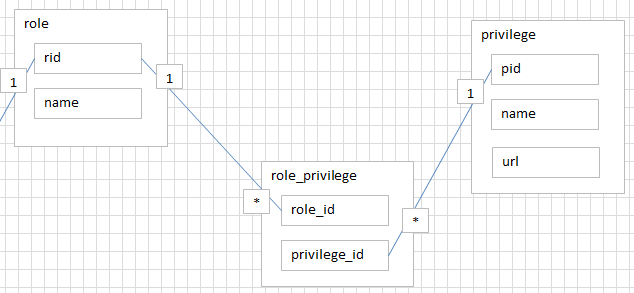
### 案例练习三

|  |
| --- |
| ##实战2：  #创建用户表  create table user(  uid VARCHAR(32) PRIMARY key,  username VARCHAR(50),  password VARCHAR(32)  );  #创建角色表  create table role(  rid VARCHAR(32) PRIMARY key,  name VARCHAR(50)  );  #实现用户与角色的多对多关系：口诀：创建一个关系表，将一方的主键拿到关系表中当外键。  #创建关系表  create table user\_role(  user\_id VARCHAR(32),  role\_id VARCHAR(32)  );  #添加主外键关系  #user\_id是用户表中uid的外键  alter TABLE user\_role add CONSTRAINT ur\_user\_fk FOREIGN key(user\_id) REFERENCES user(uid);  #role\_id是角色表中rid的外键  alter table user\_role add CONSTRAINT ur\_role\_fk FOREIGN key(role\_id) REFERENCES role(rid); |

## 多表关系--实战3--角色和权限

### 需求分析

系统中一个角色可以有多个权限，同一个权限也会给多个角色，因此在开发中角色和权限是一个多对多的关系，在数据库表结构中建立的关系如下图



角色和权限是多对多关系。

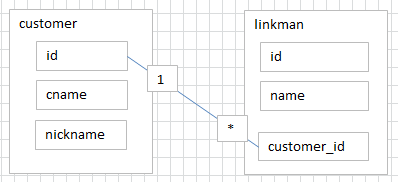
### 案例练习四

|  |
| --- |
| ##实战3：角色和权限  ##创建角色表[存在]  ##创建权限表  #drop table privilege;  CREATE TABLE privilege(  pid VARCHAR(32) PRIMARY KEY ,  NAME VARCHAR(50),  url VARCHAR(100) #之后开发中，每一个功能对应一个权限，对应一个访问路径。  );  ##创建中间表  #DROP TABLE role\_privilege;  CREATE TABLE role\_privilege(  role\_id VARCHAR(32),  privilege\_id VARCHAR(32)  );  ##添加主外键关系  ALTER TABLE role\_privilege ADD CONSTRAINT rp\_role\_fk FOREIGN KEY (role\_id) REFERENCES role (rid);  ALTER TABLE role\_privilege ADD CONSTRAINT rp\_privilege\_fk FOREIGN KEY (privilege\_id) REFERENCES privilege (pid); |

## 多表关系--实战4--客户和联系人

### 需求分析

一个客户可以有多个联系人，一个联系人可以对应多个客户。因此客户与练习人在数据库中建立的是多对多的关系，如下图(改成多对多关系)



### 案例练习五

|  |
| --- |
| ##实战4：客户和联系人  ##创建客户表  CREATE TABLE customer(  cid VARCHAR(32) PRIMARY KEY,  cname VARCHAR(50),  nickname VARCHAR(50)  );  ##创建联系人表  CREATE TABLE linkman(  id VARCHAR(32) PRIMARY KEY,  NAME VARCHAR(50),  customer\_id VARCHAR(32)  );  ##添加主外键关系  ALTER TABLE linkman ADD CONSTRAINT linkman\_fk FOREIGN KEY (customer\_id) REFERENCES customer (cid); |

# 多表查询

## 多表查询--数据初始化

### 准备数据步骤

1. 创建数据库
2. 创建分类表
3. 创建商品表
4. 分类表和商品表数据初始化

### 案例练习六

|  |
| --- |
| #创建数据库  CREATE DATABASE webdb\_3;  USE webdb\_3;  ##创建分类表  CREATE TABLE category (  cid VARCHAR(32) PRIMARY KEY ,  cname VARCHAR(50)  );  ##创建商品表  CREATE TABLE products(  pid VARCHAR(32) PRIMARY KEY ,  pname VARCHAR(50),  price INT,  flag VARCHAR(2), #是否上架标记为：1表示上架、0表示下架  category\_id VARCHAR(32),  CONSTRAINT products\_fk FOREIGN KEY (category\_id) REFERENCES category (cid)  );  ### 初始化数据  #分类  INSERT INTO category(cid,cname) VALUES('c001','家电');  INSERT INTO category(cid,cname) VALUES('c002','服饰');  INSERT INTO category(cid,cname) VALUES('c003','化妆品');  #商品  INSERT INTO products(pid, pname,price,flag,category\_id) VALUES('p001','联想',5000,'1','c001');  INSERT INTO products(pid, pname,price,flag,category\_id) VALUES('p002','海尔',3000,'1','c001');  INSERT INTO products(pid, pname,price,flag,category\_id) VALUES('p003','雷神',5000,'1','c001');  INSERT INTO products (pid, pname,price,flag,category\_id)  VALUES('p004','JACK JONES',800,'1','c002');  INSERT INTO products (pid, pname,price,flag,category\_id) VALUES('p005','真维斯',200,'1','c002');  INSERT INTO products (pid, pname,price,flag,category\_id) VALUES('p006','花花公子',440,'1','c002');  INSERT INTO products (pid, pname,price,flag,category\_id) VALUES('p007','劲霸',2000,'1','c002');  INSERT INTO products (pid, pname,price,flag,category\_id) VALUES('p008','香奈儿',800,'1','c003');  INSERT INTO products (pid, pname,price,flag,category\_id) VALUES('p009','相宜本草',200,'1','c003'); |

## 多表查询--交叉连接和内连接

### 交叉查询(笛卡尔积)

交叉连接查询(基本不会使用-得到的是两个表的乘积) [了解]

* + 语法：select \* from A,B;

### 案例练习七

|  |
| --- |
| ###多表查询  #1 交叉连接查询（笛卡尔积），两个表的成绩  ##格式：select \* from A,B;  ## 开发中没有实际意义，是所有查询最大集  SELECT \* FROM category , products;  SELECT \* FROM category c, products p; |

### 内连接查询

内连接查询(使用的关键字 inner join -- inner可以省略)

* + 隐式内连接：select \* from A,B where 条件;
  + 显示内连接：select \* from A inner join B on 条件;

### 案例练习八

|  |
| --- |
| #2 内连接查询  #2.1隐示内连接  ##格式：select \* from A , B where 连接条件  ##实例：查询那些分类的商品已经上架  SELECT \* FROM category;  SELECT \* FROM products WHERE flag = '1';  SELECT \* FROM category,products WHERE cid = category\_id AND flag = '1';  SELECT cname FROM category,products WHERE cid = category\_id AND flag = '1';  SELECT DISTINCT cname  FROM category,products  WHERE cid = category\_id AND flag = '1';  #2.2显示内连接  ##格式：select \* from A inner join B on 连接条件 where ...  SELECT \* FROM category  INNER JOIN products ON cid = category\_id  WHERE flag = '1';    SELECT DISTINCT cname FROM category  INNER JOIN products ON cid = category\_id  WHERE flag = '1';  ##建议多表查询时，使用别名  SELECT DISTINCT c.cname FROM category c  INNER JOIN products p ON c.cid = p.category\_id  WHERE p.flag = '1'; |

## 多表查询--外连接

### 外链接概述

外连接查询(使用的关键字 outer join -- outer可以省略)

* + 左外连接：left outer join
    - select \* from A left outer join B on 条件;
  + 右外连接：right outer join
    - select \* from A right outer join B on 条件;

### 案例练习九

|  |
| --- |
| #3 外连接  #3.1 左外连接：查询左表(A表)所有的数据，关联的右表(B表)数据是否显示，取决于条件是否成立，如果成立显示数据，如果不成立显示null。  ##格式：select \* from A left outer join B on 连接条件  #3.2 右外连接：查询右表(B表)所有的数据，关联的左表(A表)数据是否显示，取决于条件是否成立，如果成立显示数据，如果不成立显示null。  ##格式：select \* from A right outer join B on 连接条件  ##实例：查询所有分类商品的个数  ##思考：如果分类下没有商品，是否需要显示当前分类？需要显示，只是显示个数为0  INSERT INTO category(cid,cname) VALUES('c004','奢侈品');  SELECT \* FROM category c  LEFT OUTER JOIN products p ON c.cid = p.category\_id;  SELECT c.cname,COUNT(category\_id) AS num FROM category c  LEFT OUTER JOIN products p ON c.cid = p.category\_id  GROUP BY cname; |

## 多表查询--子查询

### 概述

子查询：一条select语句结果作为另一条select语法一部分（查询条件，查询结果，表等）。

select ....查询字段 ... from ... 表.. where ... 查询条件

### 需求分析1

查询“化妆品”分类上架商品详情

### 案例练习十

|  |
| --- |
| #子查询  ##1 查询“化妆品”为分类的上架商品详情  #方案1：内连接  SELECT \* FROM products p , category c WHERE c.cid = p.category\_id AND c.cname='化妆品' AND p.flag = '1';  #方案2：--子查询 --查询条件  SELECT cid FROM category WHERE cname = '化妆品' ;  SELECT \* FROM products p WHERE p.category\_id = 'c003';  # 查询条件  SELECT \* FROM products p WHERE p.category\_id = (SELECT cid FROM category WHERE cname = '化妆品')  #方案3：--子查询 --表  # 表  SELECT \* FROM category WHERE cname = '化妆品' ;    SELECT \* FROM products p , (SELECT \* FROM category WHERE cname = '化妆品') c WHERE c.cid = p.category\_id; |

### 需求分析2

查询“化妆品”和“家电”两个分类上架商品详情

### 案例练习十一

|  |
| --- |
| ##2 查询“化妆品”和“家电”两个分类上架商品详情  SELECT \* FROM category WHERE cname='化妆品' OR cname='家电';  SELECT \* FROM products WHERE category\_id IN ('c001','c003')  SELECT \* FROM products WHERE category\_id IN (SELECT cid FROM category WHERE cname='化妆品' OR cname='家电')  SELECT p.\* FROM products p , (SELECT \* FROM category WHERE cname='化妆品' OR cname='家电') c WHERE c.cid = p.category\_id; |

# SQL的练习

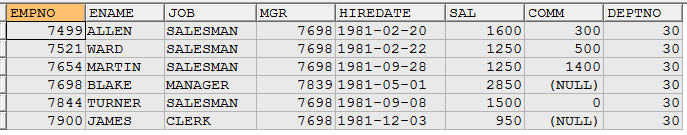
## SQL练习表结构

|  |
| --- |
| DROP DATABASE IF EXISTS test1;  CREATE DATABASE test1;  USE test1;  ##部门表  #DROP IF EXISTS TABLE DEPT;  CREATE TABLE DEPT(  DEPTNO int PRIMARY KEY,##部门编号  DNAME VARCHAR(14) , ##部门名称  LOC VARCHAR(13) ##部门地址  ) ;  INSERT INTO DEPT VALUES (10,'ACCOUNTING','NEW YORK');  INSERT INTO DEPT VALUES (20,'RESEARCH','DALLAS');  INSERT INTO DEPT VALUES (30,'SALES','CHICAGO');  INSERT INTO DEPT VALUES (40,'OPERATIONS','BOSTON');    ##员工表  #DROP IF EXISTS TABLE EMP;  CREATE TABLE EMP(  EMPNO int PRIMARY KEY, #员工编号  ENAME VARCHAR(10), #员工姓名  JOB VARCHAR(9), #员工工作  MGR int, #员工直属领导编号  HIREDATE DATE, #入职时间  SAL double, #工资  COMM double, #奖金  DEPTNO int #对应dept表的外键  );  ## 添加 部门 和 员工 之间的主外键关系  ALTER TABLE EMP ADD CONSTRAINT FOREIGN KEY EMP(DEPTNO) REFERENCES DEPT (DEPTNO);  INSERT INTO EMP  VALUES(7369,'SMITH','CLERK',7902,"1980-12-17",800,NULL,20);  INSERT INTO EMP  VALUES(7499,'ALLEN','SALESMAN',7698,'1981-02-20',1600,300,30);  INSERT INTO EMP  VALUES(7521,'WARD','SALESMAN',7698,'1981-02-22',1250,500,30);  INSERT INTO EMP  VALUES(7566,'JONES','MANAGER',7839,'1981-04-02',2975,NULL,20);  INSERT INTO EMP  VALUES(7654,'MARTIN','SALESMAN',7698,'1981-09-28',1250,1400,30);  INSERT INTO EMP  VALUES(7698,'BLAKE','MANAGER',7839,'1981-05-01',2850,NULL,30);  INSERT INTO EMP  VALUES(7782,'CLARK','MANAGER',7839,'1981-06-09',2450,NULL,10);  INSERT INTO EMP  VALUES(7788,'SCOTT','ANALYST',7566,'1987-07-03',3000,NULL,20);  INSERT INTO EMP  VALUES(7839,'KING','PRESIDENT',NULL,'1981-11-17',5000,NULL,10);  INSERT INTO EMP  VALUES(7844,'TURNER','SALESMAN',7698,'1981-09-08',1500,0,30);  INSERT INTO EMP  VALUES(7876,'ADAMS','CLERK',7788,'1987-07-13',1100,NULL,20);  INSERT INTO EMP  VALUES(7900,'JAMES','CLERK',7698,'1981-12-03',950,NULL,30);  INSERT INTO EMP  VALUES(7902,'FORD','ANALYST',7566,'1981-12-03',3000,NULL,20);  INSERT INTO EMP  VALUES(7934,'MILLER','CLERK',7782,'1981-01-23',1300,NULL,10);  #工资等级表  #DROP IF EXISTS TABLE SALGRADE;  CREATE TABLE SALGRADE(  GRADE int, #等级  LOSAL double, #最低工资  HISAL double ); #最高工资  INSERT INTO SALGRADE VALUES (1,700,1200);  INSERT INTO SALGRADE VALUES (2,1201,1400);  INSERT INTO SALGRADE VALUES (3,1401,2000);  INSERT INTO SALGRADE VALUES (4,2001,3000);  INSERT INTO SALGRADE VALUES (5,3001,9999); |

## 3.2单表查询

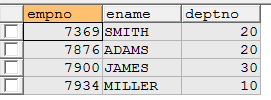
1. 查找部门30中员工的详细信息。

|  |
| --- |
| 结果：select \* from emp where deptno = 30; |



1. 找出从事clerk工作的员工的编号、姓名、部门号。

|  |
| --- |
| 结果：select empno,ename,deptno from emp where job = 'clerk'; |



1. 检索出奖金多于基本工资的员工信息。

|  |
| --- |
| 结果：select \* from emp where comm > sal; |



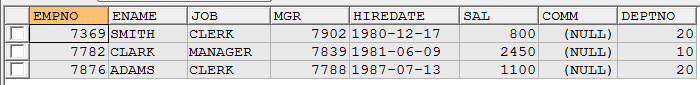
1. 检索出奖金多于基本工资60%的员工信息。

|  |
| --- |
| 结果：select \* from emp where comm > sal \* 0.6; |



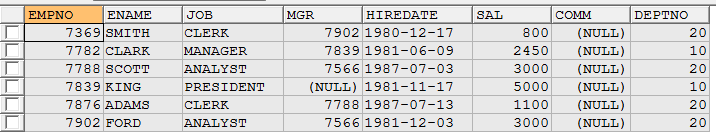
1. 找出10部门的经理、20部门的职员 的员工信息。

|  |
| --- |
| 结果：select \* from emp where deptno = 10 and job='MANAGER' or deptno = 20 and job = 'CLERK'; |



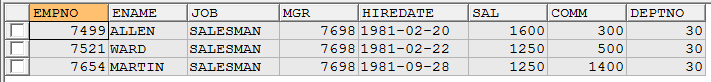
1. 找出10部门的经理、20部门的职员 或者既不是经理也不是职员但是工资高于2000元的员工信息。

|  |
| --- |
| 结果：select \* from emp  where (deptno = 10 and job='MANAGER')  or (deptno = 20 and job = 'CLERK')  or (job!='MANAGER' and job != 'CLERK' and sal > 2000) ;  ##or (job not in ('MANAGER','CLERK' and sal > 2000); |



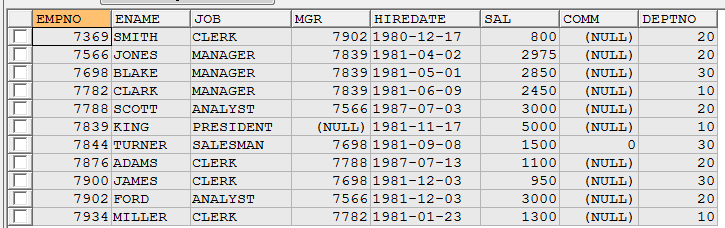
1. 找出获得奖金的员工的工作。

|  |
| --- |
| 结果：select \* from emp where comm > 0; |



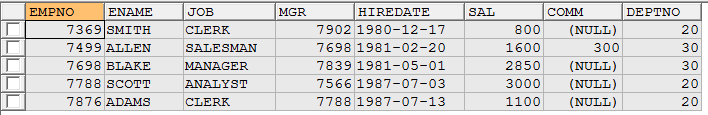
1. 找出奖金少于100或者没有获得奖金的员工的信息。

|  |
| --- |
| 结果：select \* from emp where comm < 100 or comm is null; |



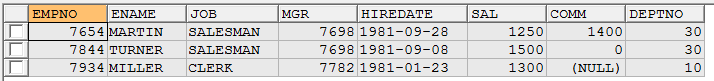
1. 找出姓名以A、B、S开始的员工信息。

|  |
| --- |
| 结果：select \* from emp where ename like 'A%' or ename like 'B%' or ename like 'S%'; |



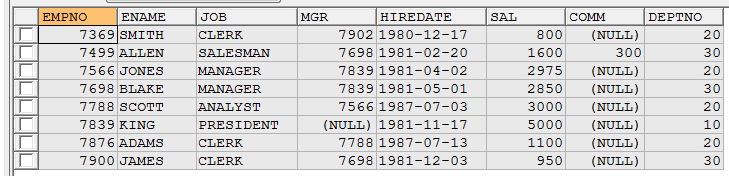
1. 找到名字长度为6个字符的员工信息。

|  |
| --- |
| 结果：select \* from emp where length(ename) = 6;  #select \* from emp where ename like '\_\_\_\_\_\_'; |



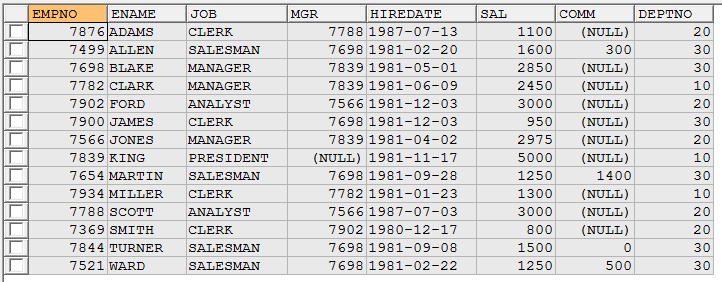
1. 名字中不包含R字符的员工信息。

|  |
| --- |
| 结果：select \* from emp where ename not like '%R%'; |



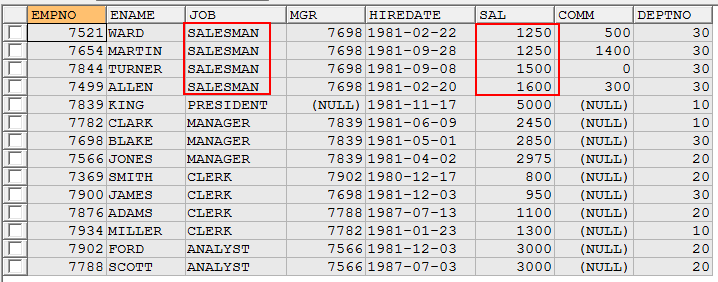
1. 返回员工的详细信息并按姓名排序。

|  |
| --- |
| 结果：select \* from emp order by ename asc; |



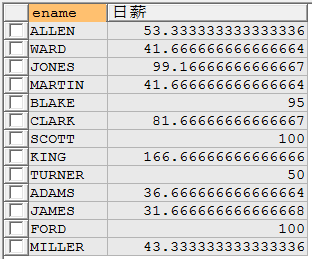
1. 返回员工的信息并按工作降序工资升序排列。

|  |
| --- |
| 结果：select \* from emp order by job desc,sal asc; |

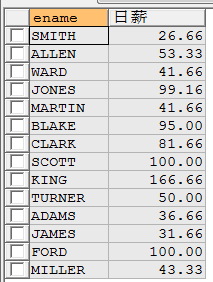


1. 计算员工的日薪(按30天)。

|  |
| --- |
| 结果：select ename,sal/30 as '日薪' from emp;  select ename,truncate(sal/30,2) '日薪' from emp; |

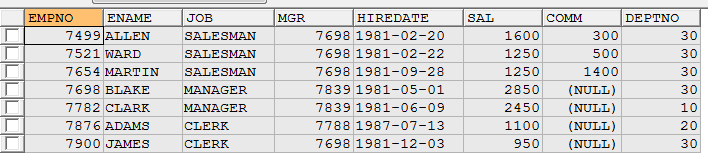


扩展：参考《MySQL\_5.1\_zh》-- 第12章：函数和操作符



1. 找出姓名中包含A的员工信息。

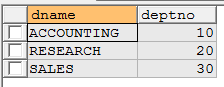
|  |
| --- |
| 结果：select \* from emp where ename like '%A%'; |



## 3.3多表查询

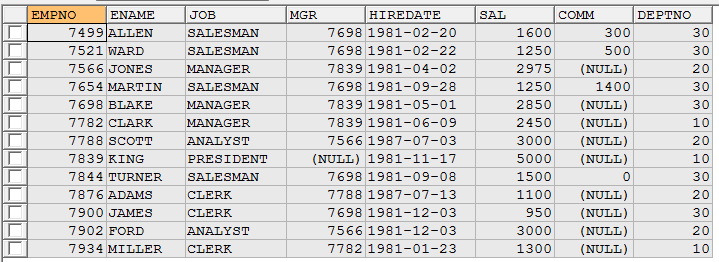
1. 返回拥有员工的部门名、部门号。

|  |
| --- |
| 结果：select distinct d.dname, d.deptno from dept d,emp e where d.deptno = e.deptno; |



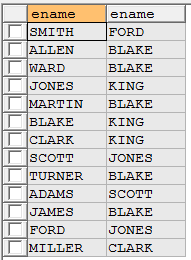
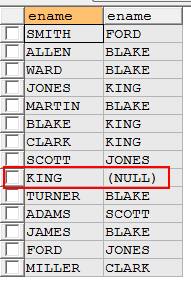
1. 工资水平多于smith的员工信息。

|  |
| --- |
| 结果：select \*from emp where sal > (select sal from emp where ename = 'smith'); |



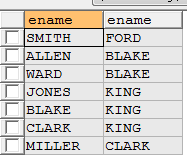
1. 返回员工和所属经理的姓名。

|  |
| --- |
| 结果：select e.ename,m.ename from emp e  left outer join emp m on e.mgr = m.empno;  select e.ename ,(select m.ename from emp m where m.empno = e.mgr) ename from emp e;  select e.ename , m.ename from emp e , emp m where e.mgr = m.empno; |

 or  （没有经理是否显示）

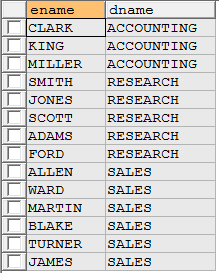
1. 返回雇员的雇佣日期早于其经理雇佣日期的员工及其经理姓名

|  |
| --- |
| 结果：select e.ename,m.ename from emp e  inner join emp m on e.mgr = m.empno  where e.hiredate < m.hiredate;  select e.ename,m.ename from emp e,emp m  where e.mgr=m.empno  and e.hiredate < m.hiredate; |



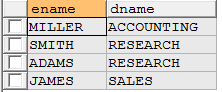
1. 返回员工姓名及其所在的部门名称。

|  |
| --- |
| 结果：select e.ename,d.dname from emp e , dept d where e.deptno = d.deptno; |



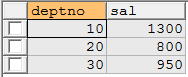
1. 返回从事clerk工作的员工姓名和所在部门名称。

|  |
| --- |
| 结果：select e.ename,d.dname  from emp e , dept d  where e.deptno = d.deptno and e.job = 'CLERK'; |



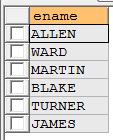
1. 返回部门号及其本部门的最低工资。

|  |
| --- |
| 结果：select deptno ,min(sal) sal  from emp  group by deptno |



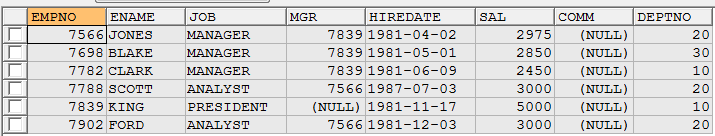
1. 返回销售部(sales)所有员工的姓名。

|  |
| --- |
| 结果：select e.ename from emp e,dept d  where e.deptno = d.deptno and d.dname = 'sales'; |



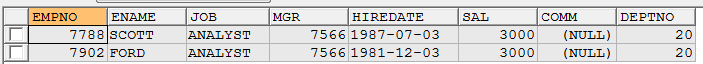
1. 返回工资水平多于平均工资的员工。

|  |
| --- |
| 结果：select \* from emp e  where e.sal > (select avg(sal) from emp); |



1. 返回与SCOTT从事相同工作的员工。

|  |
| --- |
| 结果：select \* from emp  where job = (select job from emp where ename = 'scott');  select e1.\* from emp e1 , (select empno,job from emp where ename = 'scott') e2  where e1.job = e2.job and e1.empno != e2.empno; |

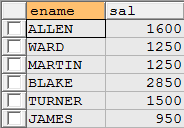


如果不包含自己



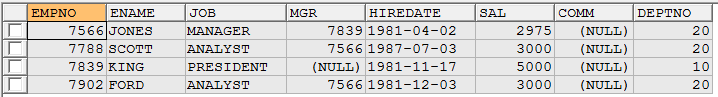
1. 返回与30部门员工工资水平相同的员工姓名与工资。

|  |
| --- |
| 结果：select ename,sal from emp  where sal in (select sal from emp where deptno = 30); |



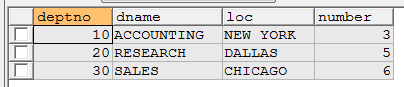
1. 返回工资高于30部门所有员工工资水平的员工信息。

|  |
| --- |
| 结果：select \* from emp  where sal > all(select sal from emp where deptno = 30); |



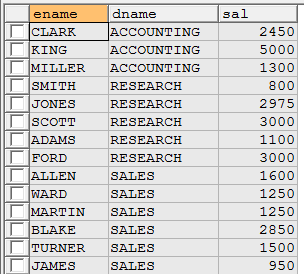
1. 返回部门号、部门名、部门所在位置及其每个部门的员工总数。

|  |
| --- |
| 结果：select dept.deptno,dept.dname,dept.loc,count(emp.deptno) number  from dept,emp  where dept.deptno = emp.deptno  group by emp.deptno; |



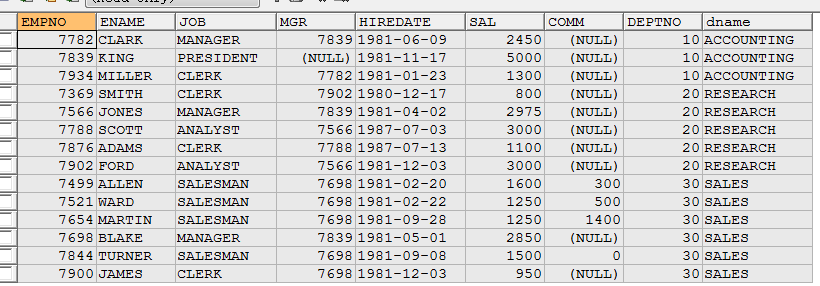
1. 返回员工的姓名、所在部门名及其工资。

|  |
| --- |
| 结果：select ename,dname,sal from emp ,dept  where emp.deptno = dept.deptno; |



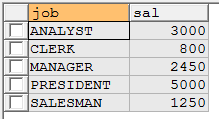
1. 返回员工的详细信息。(包括部门名)

|  |
| --- |
| 结果：select e.\* , d.dname from emp e, dept d  where e.deptno = d.deptno; |



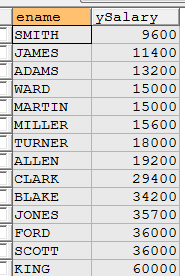
1. 返回员工工作及其从事此工作的最低工资。

|  |
| --- |
| 结果：select job , min(sal) sal from emp  group by job |



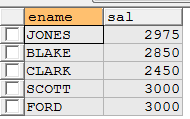
1. 计算出员工的年薪，并且以年薪排序。

|  |
| --- |
| 结果：select ename, sal \* 12 as ySalary from emp order by ySalary; |



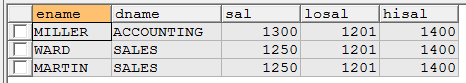
1. 返回工资处于第四级别的员工的姓名。

|  |
| --- |
| 结果：select ename,sal from emp e ,salgrade s  where e.sal >= s.losal and e.sal <= s.hisal  and s.grade = 4;  select emp.ename,emp.sal from  emp ,(select losal,hisal from salgrade where grade=4) g  where emp.sal between g.losal and g.hisal; |



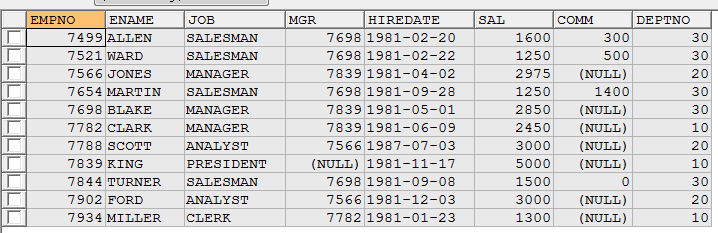
1. 返回工资为二等级的职员名字、部门所在地、和二等级的最低工资和最高工资

|  |
| --- |
| 结果：select ename ,dname ,sal ,losal,hisal from emp,dept,salgrade  where emp.deptno = dept.deptno and grade = 2  and sal >= losal and sal < hisal; |



1. 工资等级多于smith的员工信息。

|  |
| --- |
| 结果：select grade from salgrade s ,emp e  where s.losal < e.sal and s.hisal > e.sal and e.ename = 'smith';  select e.\* from emp e, salgrade s  where s.hisal < e.sal and s.grade = 1;  select e.\* from emp e, salgrade s  where s.hisal < e.sal and s.grade =  (select grade from salgrade s ,emp e  where s.losal < e.sal and s.hisal > e.sal and e.ename = 'smith'); |



重点和总结

1、理解多表关系和应用

2、了解多表查询的交叉查询

3、熟练掌握多表查询的内连接查询和外连接查询

4、熟练掌握子查询的操作