# 第01章_数据库概述

讲师：尚硅谷-宋红康（江湖人称：康师傅）

官网： http://www.atguigu.com

---

1.说说你了解的常见的数据库

Oracle、MySQl、SQL Server、DB2、PGSQL；Redis、MongoDB、ES.....

2.谈谈你对MySQL历史、特点的理解

- 历史：
  - 由瑞典的MySQL AB 公司创立，1995开发出的MySQL
  - 2008年，MySQL被SUN公司收购
  - 2009年，Oracle收购SUN公司，进而Oracle就获取了MySQL
  - 2016年，MySQL8.0.0版本推出
- 特点：
  - 开源的、关系型的数据库
  - 支持千万级别数据量的存储，大型的数据库

3.说说你对DB、DBMS、SQL的理解

DB：database，看做是数据库文件。 （类似于：.doc、.txt、.mp3、.avi、。。。）

DBMS：数据库管理系统。（类似于word工具、wps工具、记事本工具、qq影音播放器等）

MySQL数据库服务器中安装了MySQL DBMS,使用MySQL DBMS 来管理和操作DB，使用的是SQL语言。

4.你知道哪些非关系型数据库的类型呢？（了解）

- 键值型数据库：Redis
- 文档型数据库：MongoDB
- 搜索引擎数据库：ES、Solr
- 列式数据库：HBase
- 图形数据库：InfoGrid

5.表与表的记录之间存在哪些关联关系？

- ORM思想。（了解）
- 表与表的记录之间的关系：一对一关系、一对多关系、多对多关系、自关联 （了解）

# 第02章_MySQL环境搭建

讲师：尚硅谷-宋红康（江湖人称：康师傅）

官网： http://www.atguigu.com

---

1.安装好MySQL之后在windows系统中哪些位置能看到MySQL?

- MySQL DBMS软件的安装位置。 `D:\develop_tools\MySQL\MySQL Server 8.0`
- MySQL 数据库文件的存放位置。 `C:\ProgramData\MySQL\MySQL Server 8.0\Data`
- MySQL DBMS 的配置文件。 `C:\ProgramData\MySQL\MySQL Server 8.0\my.ini`
- MySQL的服务（要想通过客户端能够访问MySQL的服务器，必须保证服务是开启状态的）
- MySQL的path环境变量

2.卸载MySQL主要卸载哪几个位置的内容?

- 使用控制面板的软件卸载，去卸载MySQL DBMS软件的安装位置。 `D:\develop_tools\MySQL\MySQL Server 8.0`
- 手动删除数据库文件。 `C:\ProgramData\MySQL\MySQL Server 8.0\Data`
- MySQL的环境变量
- MySQL的服务进入注册表删除。 （ `regedit` ）
- 务必重启电脑

3.能够独立完成MySQL8.0、MySQL5.7版本的下载、安装、配置 （**掌握**）

4.MySQL5.7在配置完以后，如何修改配置文件?

- 为什么要修改my.ini文件? 默认的数据库使用的字符集是latin1。我们需要修改为：utf8
- 修改哪些信息?

```
[mysql]  #大概在63行左右，在其下添加
...
default-character-set=utf8  #默认字符集

[mysqld]  # 大概在76行左右，在其下添加
...
character-set-server=utf8
collation-server=utf8_general_ci
```

修改完以后，需要重启服务。

```
net stop mysql服务名;

net start mysql服务名;
```

5.熟悉常用的数据库管理和操作的工具

- 方式1：windows自带的cmd
- 方式2：mysql数据库自带的命令行窗口
- 方式3：图形化管理工具：Navicat、SQLyog、dbeaver等。

# 第03章_基本的SELECT语句

讲师：尚硅谷-宋红康（江湖人称：康师傅）

官网： http://www.atguigu.com

---

```
【题目】
# 1.查询员工12个月的工资总和，并起别名为ANNUAL SALARY

# 2.查询employees表中去除重复的job_id以后的数据

# 3.查询工资大于12000的员工姓名和工资

# 4.查询员工号为176的员工的姓名和部门号

# 5.显示表 departments 的结构，并查询其中的全部数据
```

## 1.查询员工12个月的工资总和，并起别名为ANNUAL SALARY

```
SELECT employee_id , last_name,salary * 12  "ANNUAL  SALARY"
FROM employees;

SELECT employee_id,last_name,salary * 12 * (1 + IFNULL(commission_pct,0)) "ANNUAL
SALARY"
FROM employees;
```

## 2.查询employees表中去除重复的job_id以后的数据

```
SELECT DISTINCT job_id
FROM employees;
```

## 3.查询工资大于12000的员工姓名和工资

```
SELECT last_name, salary
FROM employees
WHERE salary > 12000;
```

## 4.查询员工号为176的员工的姓名和部门号

```
SELECT last_name, department_id
FROM employees
WHERE employee_id = 176;
```

## 5.显示表 departments 的结构，并查询其中的全部数据

```
DESC departments;

SELECT * FROM departments;
```

# 第04章_运算符

讲师：尚硅谷-宋红康（江湖人称：康师傅）

官网： http://www.atguigu.com

---

```
【题目】
# 1.选择工资不在5000到12000的员工的姓名和工资

# 2.选择在20或50号部门工作的员工姓名和部门号

# 3.选择公司中没有管理者的员工姓名及job_id

# 4.选择公司中有奖金的员工姓名，工资和奖金级别

# 5.选择员工姓名的第三个字母是a的员工姓名

# 6.选择姓名中有字母a和k的员工姓名

# 7.显示出表 employees 表中 first_name 以 'e'结尾的员工信息

# 8.显示出表 employees 部门编号在 80-100 之间的姓名、工种

# 9.显示出表 employees 的 manager_id 是 100,101,110 的员工姓名、工资、管理者id
```

## 1.选择工资不在5000到12000的员工的姓名和工资

```
SELECT last_name, salary
FROM employees
WHERE salary < 5000 OR salary > 12000;
```

```
SELECT last_name, salary
FROM employees
WHERE salary NOT BETWEEN 5000 AND 12000;
```

## 2.选择在20或50号部门工作的员工姓名和部门号

```
SELECT last_name, department_id
FROM employees
WHERE department_id = 20 OR department_id = 50;
```

```
SELECT last_name, department_id
FROM employees
WHERE department_id IN(20, 50);
```

### 3.选择公司中没有管理者的员工姓名及job_id

```
SELECT last_name, job_id
FROM employees
WHERE manager_id IS NULL;
```

### 4.选择公司中有奖金的员工姓名，工资和奖金级别

```
SELECT last_name, salary, commission_pct
FROM employees
WHERE commission_pct IS NOT NULL;
```

### 5.选员工姓名的第三个字母是a的员工姓名

```
SELECT last_name
FROM employees
WHERE last_name LIKE '__a%';
```

### 6.选择姓名中有字母a和k的员工姓名

```
SELECT last_name
FROM employees
WHERE last_name LIKE '%a%k%' OR last_name LIKE '%k%a%';
```

### 7.显示出表 employees 表中 first_name 以 'e'结尾的员工信息

```
SELECT employee_id,first_name,last_name
FROM employees
WHERE first_name LIKE '%e';
```

```
SELECT employee_id,first_name,last_name
FROM employees
WHERE first_name REGEXP 'e$';
```

### 8.显示出表 employees 部门编号在 80-100 之间的姓名、工种

```
SELECT last_name,job_id
FROM employees
#where department_id in (80,90,100);
WHERE department_id BETWEEN 80 AND 100;
```

### 9.显示出表 employees 的 manager_id 是 100,101,110 的员工姓名、工资、管理者id

```
SELECT last_name,salary,manager_id
FROM employees
WHERE manager_id IN (100,101,110);
```

# 第05章_排序与分页

讲师：尚硅谷-宋红康（江湖人称：康师傅）

官网： http://www.atguigu.com

---

题目：

```
#1. 查询员工的姓名和部门号和年薪，按年薪降序,按姓名升序显示

#2. 选择工资不在 8000 到 17000 的员工的姓名和工资，按工资降序，显示第21到40位置的数据

#3. 查询邮箱中包含 e 的员工信息，并先按邮箱的字节数降序，再按部门号升序
```

答案：

## 1. 查询员工的姓名和部门号和年薪，按年薪降序 按姓名升序显示

```
SELECT last_name,department_id,salary * 12 annual_sal
FROM employees
ORDER BY annual_sal DESC,last_name ASC;
```

## 2. 选择工资不在 8000 到 17000 的员工的姓名和工资，按工资降序，显示第21到40位置的数据

```
SELECT last_name,salary
FROM employees
WHERE salary NOT BETWEEN 8000 AND 17000
ORDER BY salary DESC
LIMIT 20,20;
```

## 3. 查询邮箱中包含 e 的员工信息，并先按邮箱的字节数降序，再按部门号升序

```
SELECT last_name,email,department_id
FROM employees
#where email like '%e%'
WHERE email REGEXP '[e]'
ORDER BY LENGTH(email) DESC,department_id ASC;
```

# 第06章_多表查询

讲师：尚硅谷-宋红康（江湖人称：康师傅）

官网： http://www.atguigu.com

---

## 多表查询-1

```
【题目】
# 1.显示所有员工的姓名，部门号和部门名称。

# 2.查询90号部门员工的job_id和90号部门的location_id

# 3.选择所有有奖金的员工的 last_name , department_name , location_id , city

# 4.选择city在Toronto工作的员工的 last_name , job_id , department_id , department_name

# 5.查询员工所在的部门名称、部门地址、姓名、工作、工资，其中员工所在部门的部门名称为'Executive'

# 6.选择指定员工的姓名，员工号，以及他的管理者的姓名和员工号，结果类似于下面的格式
employees   Emp#    manager Mgr#
kochhar     101 king    100

# 7.查询哪些部门没有员工

# 8. 查询哪个城市没有部门

# 9. 查询部门名为 Sales 或 IT 的员工信息
```

### 1.显示所有员工的姓名，部门号和部门名称

```
SELECT last_name, e.department_id, department_name
FROM employees e
LEFT OUTER JOIN departments d
ON e.`department_id` = d.`department_id`;
```

### 2.查询90号部门员工的job_id和90号部门的location_id

```
SELECT job_id, location_id
FROM employees e, departments d
WHERE e.`department_id` = d.`department_id`
AND e.`department_id` = 90;
```

或

```
SELECT job_id, location_id
FROM employees e
JOIN departments d
ON e.`department_id` = d.`department_id`
WHERE e.`department_id` = 90;
```

### 3.选择所有有奖金的员工的 last_name , department_name , location_id , city

```
SELECT last_name , department_name , d.location_id , city
FROM employees e
LEFT OUTER JOIN departments d
ON e.`department_id` = d.`department_id`
LEFT OUTER JOIN locations l
ON d.`location_id` = l.`location_id`
WHERE commission_pct IS NOT NULL;
```

### 4.选择city在Toronto工作的员工的 last_name , job_id , department_id , department_name

```
SELECT last_name , job_id , e.department_id , department_name
FROM employees e, departments d, locations l
WHERE e.`department_id` = d.`department_id`
AND d.`location_id` = l.`location_id`
AND city = 'Toronto';
```

或

```
SELECT last_name , job_id , e.department_id , department_name
FROM employees e
JOIN departments d
ON e.`department_id` = d.`department_id`
JOIN locations l
ON l.`location_id` = d.`location_id`
WHERE l.`city` = 'Toronto';
```

### 5．查询员工所在的部门名称、部门地址、姓名、工作、工资，其中员工所在部门的部门名称为’Executive’

```
SELECT department_name, street_address, last_name, job_id, salary
FROM employees e JOIN departments d
ON e.department_id = d.department_id
JOIN locations l
ON d.`location_id` = l.`location_id`
WHERE department_name = 'Executive'
```

### 6. 选择指定员工的姓名，员工号，以及他的管理者的姓名和员工号，结果类似于下面的格式

```
employees    Emp#      manager      Mgr#
kochhar      101        king        100
```

```
SELECT emp.last_name employees, emp.employee_id "Emp#", mgr.last_name manager,
mgr.employee_id "Mgr#"
FROM employees emp
LEFT OUTER JOIN employees mgr
ON emp.manager_id = mgr.employee_id;
```

### 7. 查询哪些部门没有员工

```
#方式1：
SELECT d.department_id
FROM departments d LEFT JOIN employees e
ON e.department_id = d.`department_id`
WHERE e.department_id IS NULL

#方式2：
SELECT department_id
FROM departments d
WHERE NOT EXISTS (
        SELECT *
        FROM employees e
        WHERE e.`department_id` = d.`department_id`
         )
```

### 8. 查询哪个城市没有部门

```
SELECT l.location_id,l.city
FROM locations l LEFT JOIN departments d
ON l.`location_id` = d.`location_id`
WHERE d.`location_id` IS NULL
```

### 9. 查询部门名为 Sales 或 IT 的员工信息

```
SELECT employee_id,last_name,department_name
FROM employees e,departments d
WHERE e.department_id = d.`department_id`
AND d.`department_name` IN ('Sales','IT');
```

## 多表查询-2

```
储备：建表操作：
CREATE TABLE `t_dept` (
 `id` INT(11) NOT NULL AUTO_INCREMENT,
 `deptName` VARCHAR(30) DEFAULT NULL,
 `address` VARCHAR(40) DEFAULT NULL,
 PRIMARY KEY (`id`)
) ENGINE=INNODB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8;

CREATE TABLE `t_emp` (
 `id` INT(11) NOT NULL AUTO_INCREMENT,
 `name` VARCHAR(20) DEFAULT NULL,
  `age` INT(3) DEFAULT NULL,
 `deptId` INT(11) DEFAULT NULL,
empno int  not null,
 PRIMARY KEY (`id`),
 KEY `idx_dept_id` (`deptId`)
 #CONSTRAINT `fk_dept_id` FOREIGN KEY (`deptId`) REFERENCES `t_dept` (`id`)
) ENGINE=INNODB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8;
```

```
INSERT INTO t_dept(deptName,address) VALUES('华山','华山');
INSERT INTO t_dept(deptName,address) VALUES('丐帮','洛阳');
INSERT INTO t_dept(deptName,address) VALUES('峨眉','峨眉山');
INSERT INTO t_dept(deptName,address) VALUES('武当','武当山');
INSERT INTO t_dept(deptName,address) VALUES('明教','光明顶');
INSERT INTO t_dept(deptName,address) VALUES('少林','少林寺');
INSERT INTO t_emp(NAME,age,deptId,empno) VALUES('风清扬',90,1,100001);
INSERT INTO t_emp(NAME,age,deptId,empno) VALUES('岳不群',50,1,100002);
INSERT INTO t_emp(NAME,age,deptId,empno) VALUES('令狐冲',24,1,100003);
INSERT INTO t_emp(NAME,age,deptId,empno) VALUES('洪七公',70,2,100004);
INSERT INTO t_emp(NAME,age,deptId,empno) VALUES('乔峰',35,2,100005);
INSERT INTO t_emp(NAME,age,deptId,empno) VALUES('灭绝师太',70,3,100006);
INSERT INTO t_emp(NAME,age,deptId,empno) VALUES('周芷若',20,3,100007);
INSERT INTO t_emp(NAME,age,deptId,empno) VALUES('张三丰',100,4,100008);
INSERT INTO t_emp(NAME,age,deptId,empno) VALUES('张无忌',25,5,100009);
INSERT INTO t_emp(NAME,age,deptId,empno) VALUES('韦小宝',18,null,100010);

【题目】
#1.所有有门派的人员信息
（ A、B两表共有）

#2.列出所有用户，并显示其机构信息
 （A的全集）

#3.列出所有门派
（B的全集）

#4.所有不入门派的人员
（A的独有）

#5.所有没人入的门派
（B的独有）

#6.列出所有人员和机构的对照关系
(AB全有)
#MySQL Full Join的实现 因为MySQL不支持FULL JOIN,下面是替代方法
#left join + union(可去除重复数据)+ right join

#7.列出所有没入派的人员和没人入的门派
（A的独有+B的独有）
```

## 1. 所有有门派的人员信息

（ A、B两表共有）

```
select *
from t_emp a inner join t_dept b
on a.deptId = b.id;
```

## 2. 列出所有用户，并显示其机构信息

(A的全集)

```
select *
from t_emp a left join t_dept b
on a.deptId = b.id;
```

## 3. 列出所有门派

(B的全集)

```
select *
from  t_dept  b;
```

## 4. 所有不入门派的人员

(A的独有)

```
select *
from t_emp a left join t_dept b
on a.deptId = b.id
where b.id is null;
```

## 5. 所有没人入的门派

(B的独有)

```
select *
from t_dept b left join  t_emp a
on a.deptId = b.id
where a.deptId is null;
```

## 6. 列出所有人员和机构的对照关系

(AB全有)

```
#MySQL Full Join的实现 因为MySQL不支持FULL JOIN,下面是替代方法
#left join + union(可去除重复数据)+ right join

SELECT *
FROM t_emp A LEFT JOIN t_dept B
ON A.deptId = B.id
UNION
SELECT *
FROM t_emp A RIGHT JOIN t_dept B
ON A.deptId = B.id
```

## 7. 列出所有没入派的人员和没人入的门派

(A的独有+B的独有)

```sql
SELECT *
FROM t_emp A LEFT JOIN t_dept B
ON A.deptId = B.id
WHERE B.`id` IS NULL
UNION
SELECT *
FROM t_emp A RIGHT JOIN t_dept B
ON A.deptId = B.id
WHERE A.`deptId` IS NULL;
```

# 第07章_单行函数

讲师：尚硅谷-宋红康（江湖人称：康师傅）

官网： http://www.atguigu.com

---

```
【题目】
# 1.显示系统时间(注：日期+时间)

# 2.查询员工号，姓名，工资，以及工资提高百分之20%后的结果（new salary）

# 3.将员工的姓名按首字母排序，并写出姓名的长度（length）

# 4.查询员工id,last_name,salary，并作为一个列输出，别名为OUT_PUT

# 5.查询公司各员工工作的年数、工作的天数，并按工作年数的降序排序

# 6.查询员工姓名，hire_date , department_id，满足以下条件：雇用时间在1997年之后，department_id
为80 或 90 或110, commission_pct不为空

# 7.查询公司中入职超过10000天的员工姓名、入职时间

# 8.做一个查询，产生下面的结果
<last_name> earns <salary> monthly but wants <salary*3>
```

| Dream Salary |
| --- |
| King earns 24000 monthly but wants 72000 |

```
# 9.使用case-when，按照下面的条件：
job                  grade
AD_PRES              A
ST_MAN               B
IT_PROG              C
SA_REP               D
ST_CLERK             E

产生下面的结果：
```

| Last_name | Job_id | Grade |
| --- | --- | --- |
| king | AD_PRES | A |

**答案：**

## 1.显示系统时间(注：日期+时间)

```
SELECT NOW()
FROM DUAL;
```

## 2.查询员工号，姓名，工资，以及工资提高百分之20%后的结果（new salary）

```
SELECT employee_id, last_name, salary, salary * 1.2 "new salary"
FROM employees;
```

## 3.将员工的姓名按首字母排序，并写出姓名的长度（length）

```
SELECT last_name, LENGTH(last_name)
FROM employees
ORDER BY last_name DESC;
```

## 4.查询员工id,last_name,salary，并作为一个列输出，别名为OUT_PUT

```
SELECT CONCAT(employee_id, ',' , last_name , ',', salary) OUT_PUT
FROM employees;
```

## 5.查询公司各员工工作的年数、工作的天数，并按工作年数的降序排序。

```
SELECT DATEDIFF(SYSDATE(), hire_date) / 365 worked_years, DATEDIFF(SYSDATE(),
hire_date) worked_days
FROM employees
ORDER BY worked_years DESC
```

## 6.查询员工姓名，hire_date , department_id，满足以下条件：雇用时间在1997年之后，department_id 为80 或 90 或110, commission_pct不为空

```
SELECT last_name, hire_date, department_id
FROM employees
#WHERE hire_date >= '1997-01-01'
#WHERE hire_date >= STR_TO_DATE('1997-01-01', '%Y-%m-%d')
WHERE DATE_FORMAT(hire_date,'%Y') >= '1997'
AND department_id IN (80, 90, 110)
AND commission_pct IS NOT NULL
```

## 7.查询公司中入职超过10000天的员工姓名、入职时间

```
SELECT last_name,hire_date
FROM employees
#WHERE TO_DAYS(NOW()) - to_days(hire_date) > 10000;
WHERE DATEDIFF(NOW(),hire_date) > 10000;
```

## 8.做一个查询，产生下面的结果

```
-- <last_name> earns `<salary>` monthly but wants <salary*3>
-- Dream Salary
-- King earns 24000 monthly but wants 72000
```

```
SELECT CONCAT(last_name, ' earns ', TRUNCATE(salary, 0) , ' monthly but wants ',
TRUNCATE(salary * 3, 0)) "Dream Salary"
FROM employees;
```

## 9.使用CASE-WHEN，按照下面的条件：

```
-- job                 grade
-- AD_PRES             A
-- ST_MAN              B
-- IT_PROG             C
-- SA_REP              D
-- ST_CLERK            E

-- 产生下面的结果
-- Last_name    Job_id  Grade
-- king         AD_PRES A
```

```
SELECT last_name Last_name, job_id Job_id, CASE job_id WHEN 'AD_PRES' THEN 'A'
                                                       WHEN 'ST_MAN' THEN 'B'
                                                       WHEN 'IT_PROG' THEN 'C'
                                                       WHEN 'SA_REP' THEN 'D'
                                                       WHEN 'ST_CLERK' THEN 'E'
                                                       ELSE 'F'
                                                       END "grade"
FROM employees;
```

# 第08章_聚合函数

讲师：尚硅谷-宋红康（江湖人称：康师傅）

官网： http://www.atguigu.com

---

```
【题目】
#1.where子句可否使用组函数进行过滤?

#2.查询公司员工工资的最大值，最小值，平均值，总和

#3.查询各job_id的员工工资的最大值，最小值，平均值，总和

#4.选择具有各个job_id的员工人数

# 5.查询员工最高工资和最低工资的差距（DIFFERENCE）

# 6.查询各个管理者手下员工的最低工资，其中最低工资不能低于6000，没有管理者的员工不计算在内

# 7.查询所有部门的名字，location_id，员工数量和平均工资，并按平均工资降序

# 8.查询每个工种、每个部门的部门名、工种名和最低工资
```

## #1.where子句可否使用组函数进行过滤?

```
No!
```

## #2.查询公司员工工资的最大值，最小值，平均值，总和

```
SELECT MAX(salary), MIN(salary), AVG(salary), SUM(salary)
FROM employees;
```

## #3.查询各job_id的员工工资的最大值，最小值，平均值，总和

```
SELECT job_id, MAX(salary), MIN(salary), AVG(salary), SUM(salary)
FROM employees
GROUP BY job_id;
```

## #4.选择具有各个job_id的员工人数

```
SELECT job_id, COUNT(*)
FROM employees
GROUP BY job_id;
```

## 5.查询员工最高工资和最低工资的差距（DIFFERENCE）

```
SELECT MAX(salary), MIN(salary), MAX(salary) - MIN(salary) DIFFERENCE
FROM employees;
```

## 6.查询各个管理者手下员工的最低工资，其中最低工资不能低于6000，没有管理者的员工不计算在内

```
SELECT manager_id, MIN(salary)
FROM employees
WHERE manager_id IS NOT NULL
GROUP BY manager_id
HAVING MIN(salary) > 6000;
```

## 7.查询所有部门的名字，location_id，员工数量和平均工资，并按平均工资降序

```
SELECT department_name, location_id, COUNT(employee_id), AVG(salary) avg_sal
FROM employees e RIGHT JOIN departments d
ON e.`department_id` = d.`department_id`
GROUP BY department_name, location_id
ORDER BY avg_sal DESC;
```

## 8.查询每个工种、每个部门的部门名、工种名和最低工资

```
SELECT department_name,job_id,MIN(salary)
FROM departments d LEFT JOIN employees e
ON e.`department_id` = d.`department_id`
GROUP BY department_name,job_id
```

# 第09章_子查询

讲师：尚硅谷-宋红康（江湖人称：康师傅）

官网： http://www.atguigu.com

---

【题目】

```
#1.查询和Zlotkey相同部门的员工姓名和工资

#2.查询工资比公司平均工资高的员工的员工号，姓名和工资。

#3.选择工资大于所有JOB_ID = 'SA_MAN'的员工的工资的员工的last_name, job_id, salary

#4.查询和姓名中包含字母u的员工在相同部门的员工的员工号和姓名

#5.查询在部门的location_id为1700的部门工作的员工的员工号

#6.查询管理者是King的员工姓名和工资

#7.查询工资最低的员工信息: last_name, salary

#8.查询平均工资最低的部门信息

#9.查询平均工资最低的部门信息和该部门的平均工资（相关子查询）

#10.查询平均工资最高的 job 信息

#11.查询平均工资高于公司平均工资的部门有哪些?

#12.查询出公司中所有 manager 的详细信息

#13.各个部门中 最高工资中最低的那个部门的 最低工资是多少?

#14.查询平均工资最高的部门的 manager 的详细信息: last_name, department_id, email, salary

#15. 查询部门的部门号，其中不包括job_id是"ST_CLERK"的部门号

#16. 选择所有没有管理者的员工的last_name

#17. 查询员工号、姓名、雇用时间、工资，其中员工的管理者为 'De Haan'

#18.查询各部门中工资比本部门平均工资高的员工的员工号, 姓名和工资（相关子查询）

#19.查询每个部门下的部门人数大于 5 的部门名称（相关子查询）

#20.查询每个国家下的部门个数大于 2 的国家编号（相关子查询）
```

## 1.查询和Zlotkey相同部门的员工姓名和工资

```
SELECT last_name, salary
FROM employees
WHERE department_id = (
    SELECT department_id
    FROM employees
    WHERE last_name = 'Zlotkey'
)
```

## 2.查询工资比公司平均工资高的员工的员工号，姓名和工资

```
SELECT employee_id, last_name, salary
FROM employees
WHERE salary > (
    SELECT AVG(salary)
    FROM employees
)
```

## 3.选择工资大于所有JOB_ID = 'SA_MAN'的员工的工资的员工的last_name, job_id, salary

```
SELECT last_name,job_id,salary
FROM employees
WHERE salary > ALL (
            SELECT salary
            FROM employees
            WHERE job_id = 'SA_MAN'
            );
```

## 4.查询和姓名中包含字母u的员工在相同部门的员工的员工号和姓名

```
SELECT employee_id, last_name
FROM employees
WHERE department_id = ANY(
    SELECT DISTINCT department_id
    FROM employees
    WHERE last_name LIKE '%u%'
)
```

## 5.查询在部门的location_id为1700的部门工作的员工的员工号

```
SELECT employee_id
FROM employees
WHERE department_id IN (
    SELECT department_id
    FROM departments
    WHERE location_id = 1700
)
```

## 6.查询管理者是King的员工姓名和工资

```
SELECT last_name, salary
FROM employees
WHERE manager_id IN (
    SELECT employee_id
    FROM employees
    WHERE last_name = 'King'
)
```

## 7. 查询工资最低的员工信息: last_name, salary

```
SELECT last_name,salary
FROM employees
WHERE salary = (
        SELECT MIN(salary)
        FROM employees
           );
```

## 8. 查询平均工资最低的部门信息

```
#方式一：
SELECT *
FROM departments
WHERE department_id = (
            SELECT department_id
            FROM employees
            GROUP BY department_id
            HAVING AVG(salary) = (
                        SELECT MIN(dept_avgsal)
                        FROM (
                            SELECT AVG(salary) dept_avgsal
                            FROM employees
                            GROUP BY department_id
                            ) avg_sal
                        )
            );
```

```
#方式二：
SELECT *
FROM departments
WHERE department_id = (
            SELECT department_id
            FROM employees
            GROUP BY department_id
            HAVING AVG(salary) <= ALL(
                        SELECT AVG(salary) avg_sal
                        FROM employees
                        GROUP BY department_id
                        )
            );
```

```
#方式三：
SELECT *
FROM departments
WHERE department_id = (
```

```
            SELECT department_id
            FROM employees
            GROUP BY department_id
            HAVING AVG(salary) = (
                        SELECT AVG(salary) avg_sal
                        FROM employees
                        GROUP BY department_id
                        ORDER BY avg_sal
                        LIMIT 0,1
                       )
            )
```

```
#方式四：
SELECT d.*
FROM departments d,(
    SELECT department_id,AVG(salary) avg_sal
    FROM employees
    GROUP BY department_id
    ORDER BY avg_sal
    LIMIT 0,1) dept_avg_sal
WHERE d.department_id = dept_avg_sal.department_id
```

## 9.查询平均工资最低的部门信息和该部门的平均工资（相关子查询）

```
#方式一：
SELECT d.*,(SELECT AVG(salary) FROM employees WHERE department_id = d.department_id)
avg_sal
FROM departments d
WHERE department_id = (
            SELECT department_id
            FROM employees
            GROUP BY department_id
            HAVING AVG(salary) = (
                        SELECT MIN(dept_avgsal)
                        FROM (
                            SELECT AVG(salary) dept_avgsal
                            FROM employees
                            GROUP BY department_id
                            ) avg_sal
                        )
            );
```

```
#方式二：
SELECT d.*,(SELECT AVG(salary) FROM employees WHERE department_id = d.`department_id`)
avg_sal
FROM departments d
WHERE department_id = (
            SELECT department_id
            FROM employees
            GROUP BY department_id
            HAVING AVG(salary) <= ALL(
                        SELECT AVG(salary) avg_sal
                        FROM employees
                        GROUP BY department_id
                        )
            );
```

```
#方式三：
SELECT d.*,(SELECT AVG(salary) FROM employees WHERE department_id = d.department_id)
avg_sal
FROM departments d
WHERE department_id = (
            SELECT department_id
            FROM employees
            GROUP BY department_id
            HAVING AVG(salary) = (
                        SELECT AVG(salary) avg_sal
                        FROM employees
                        GROUP BY department_id
                        ORDER BY avg_sal
                        LIMIT 0,1
                       )
            )
```

```
#方式四：
SELECT d.*,dept_avg_sal.avg_sal
FROM departments d,(
    SELECT department_id,AVG(salary) avg_sal
    FROM employees
    GROUP BY department_id
    ORDER BY avg_sal
    LIMIT 0,1) dept_avg_sal
WHERE d.department_id = dept_avg_sal.department_id
```

## 10. 查询平均工资最高的 job 信息

```
#方式一：
SELECT *
FROM jobs
WHERE job_id = (
        SELECT job_id
        FROM employees
        GROUP BY job_id
        HAVING AVG(salary) = (
                    SELECT MAX(avg_sal)
                    FROM(
                        SELECT AVG(salary) avg_sal
                        FROM employees
                        GROUP BY job_id
                        ) job_avgsal
                    )
        );
```

```
#方式二：
SELECT *
FROM jobs
WHERE job_id = (
        SELECT job_id
        FROM employees
        GROUP BY job_id
        HAVING AVG(salary) >= ALL(
                    SELECT AVG(salary)
                    FROM employees
                    GROUP BY job_id
                    )
        );
```

```
#方式三：
SELECT *
FROM jobs
WHERE job_id = (
        SELECT job_id
        FROM employees
        GROUP BY job_id
        HAVING AVG(salary) = (
                    SELECT AVG(salary) avg_sal
                    FROM employees
                    GROUP BY job_id
                    ORDER BY avg_sal DESC
                    LIMIT 0,1
                              )
         );
```

```
#方式四：
SELECT j.*
FROM jobs j,(
        SELECT job_id,AVG(salary) avg_sal
        FROM employees
        GROUP BY job_id
        ORDER BY avg_sal DESC
        LIMIT 0,1 ) job_avg_sal
WHERE j.job_id = job_avg_sal.job_id
```

## 11. 查询平均工资高于公司平均工资的部门有哪些?

```
SELECT department_id
FROM employees
WHERE department_id IS NOT NULL
GROUP BY department_id
HAVING AVG(salary) > (
              SELECT AVG(salary)
              FROM employees
            );
```

## 12. 查询出公司中所有 manager 的详细信息.

```
#方式1：
SELECT employee_id,last_name,salary
FROM employees
WHERE employee_id IN (
              SELECT DISTINCT manager_id
              FROM employees
            );
```

```
#方式2：
SELECT DISTINCT e1.employee_id, e1.last_name, e1.salary
FROM   employees e1 JOIN employees e2
WHERE e1.employee_id = e2.manager_id;
```

```
#方式3：
SELECT employee_id, last_name, salary
FROM   employees e1
WHERE  EXISTS ( SELECT *
                FROM   employees e2
                WHERE  e2.manager_id = e1.employee_id);
```

## 13. 各个部门中 最高工资中最低的那个部门的 最低工资是多少?

```
#方式1：
SELECT MIN(salary)
FROM employees
WHERE department_id = (
            SELECT department_id
            FROM employees
            GROUP BY department_id
            HAVING MAX(salary) = (
                        SELECT MIN(max_sal)
                        FROM (
                        SELECT MAX(salary) max_sal
                        FROM employees
                        GROUP BY department_id) dept_max_sal
                        )
            );

SELECT *
FROM employees
WHERE department_id = 10;
```

```
#方式2：
SELECT MIN(salary)
FROM employees
WHERE department_id = (
            SELECT department_id
            FROM employees
            GROUP BY department_id
            HAVING MAX(salary) <= ALL(
                        SELECT MAX(salary) max_sal
                        FROM employees
                        GROUP BY department_id
                        )
            );
```

```
#方式3：
SELECT MIN(salary)
FROM employees
WHERE department_id = (
            SELECT department_id
            FROM employees
            GROUP BY department_id
            HAVING MAX(salary) = (
                        SELECT MAX(salary) max_sal
                        FROM employees
                        GROUP BY department_id
                        ORDER BY max_sal
                        LIMIT 0,1
                         )
            )
```

```
#方式4：
SELECT employee_id,MIN(salary)
FROM employees e,
                (SELECT department_id,MAX(salary) max_sal
                FROM employees
                GROUP BY department_id
                ORDER BY max_sal
                LIMIT 0,1) dept_max_sal
WHERE e.department_id = dept_max_sal.department_id
```

## 14. 查询平均工资最高的部门的 manager 的详细信息: last_name, department_id, email, salary

```
#方式一：
SELECT employee_id,last_name, department_id, email, salary
FROM employees
WHERE employee_id IN (
            SELECT DISTINCT manager_id
            FROM employees
            WHERE department_id = (
                        SELECT department_id
                        FROM employees
                        GROUP BY department_id
                        HAVING AVG(salary) = (
                                    SELECT MAX(avg_sal)
                                    FROM(
```

```
                                        SELECT AVG(salary) avg_sal
                                        FROM employees
                                        GROUP BY department_id
                                        ) dept_sal
                        )
            )

);
```

```
#方式二：
SELECT employee_id,last_name, department_id, email, salary
FROM employees
WHERE employee_id IN (
                    SELECT DISTINCT manager_id
                    FROM employees
                    WHERE department_id = (
                                        SELECT department_id
                                        FROM employees e
                                        GROUP BY department_id
                                        HAVING AVG(salary)>=ALL(
                                                            SELECT AVG(salary)
                                                            FROM employees
                                                            GROUP BY department_id
                                                                )
                                            )
                        );
```

```
#方式三：
SELECT *
FROM employees
WHERE employee_id IN (
            SELECT DISTINCT manager_id
            FROM employees e,(
                    SELECT department_id,AVG(salary) avg_sal
                    FROM employees
                    GROUP BY department_id
                    ORDER BY avg_sal DESC
                    LIMIT 0,1) dept_avg_sal
            WHERE e.department_id = dept_avg_sal.department_id
            )
```

## 15. 查询部门的部门号，其中不包括job_id是"ST_CLERK"的部门号

```
#方法一：
SELECT department_id
FROM departments d
WHERE department_id NOT IN (
        SELECT DISTINCT department_id
        FROM employees
        WHERE job_id = 'ST_CLERK'
        );
```

```
#方法二：
SELECT department_id
FROM departments d
WHERE NOT EXISTS (
        SELECT *
        FROM employees e
        WHERE d.`department_id` = e.`department_id`
        AND job_id = 'ST_CLERK'
        );
```

## 16. 选择所有没有管理者的员工的last_name

```
SELECT last_name
FROM employees e1
WHERE NOT EXISTS (
        SELECT *
        FROM employees e2
        WHERE e1.manager_id = e2.employee_id
        );
```

## 17．查询员工号、姓名、雇用时间、工资，其中员工的管理者为 'De Haan'

```
#方式1：
SELECT employee_id, last_name, hire_date, salary
FROM employees
WHERE manager_id = (
    SELECT employee_id
    FROM employees
    WHERE last_name = 'De Haan'
)
```

```
#方式2：
SELECT employee_id, last_name, hire_date, salary
FROM employees e1
WHERE EXISTS (
        SELECT *
        FROM employees e2
        WHERE e2.`employee_id` = e1.manager_id
        AND e2.last_name = 'De Haan'
        );
```

## 18.查询各部门中工资比本部门平均工资高的员工的员工号, 姓名和工资(难)

```
#方式一：相关子查询
SELECT employee_id,last_name,salary
FROM employees e1
WHERE salary > (
        # 查询某员工所在部门的平均
        SELECT AVG(salary)
        FROM employees e2
        WHERE e2.department_id = e1.`department_id`
        );
```

```
#方式二：
SELECT employee_id,last_name,salary
FROM employees e1,
(SELECT department_id,AVG(salary) avg_sal
FROM employees e2 GROUP BY department_id
) dept_avg_sal
WHERE e1.`department_id` = dept_avg_sal.department_id
AND e1.`salary` > dept_avg_sal.avg_sal;
```

## 19.查询每个部门下的部门人数大于 5 的部门名称

```
SELECT department_name,department_id
FROM departments d
WHERE 5 < (
    SELECT COUNT(*)
    FROM employees e
    WHERE d.`department_id` = e.`department_id`
    );
```

## 20.查询每个国家下的部门个数大于 2 的国家编号

```
SELECT country_id
FROM locations l
WHERE 2 < (
    SELECT COUNT(*)
    FROM departments d
    WHERE l.`location_id` = d.`location_id`
    );
```

# 第10章_创建和管理表

讲师：尚硅谷-宋红康（江湖人称：康师傅）

官网： http://www.atguigu.com

---

## 练习1

题目：

```
#1. 创建数据库test01_office,指明字符集为utf8。并在此数据库下执行下述操作

#2. 创建表dept01
/*
字段       类型
id        INT(7)
NAME      VARCHAR(25)
*/

#3. 将表departments中的数据插入新表dept02中

#4. 创建表emp01
/*
字段          类型
id           INT(7)
first_name   VARCHAR (25)
last_name    VARCHAR(25)
dept_id      INT(7)
*/

#5. 将列last_name的长度增加到50

#6. 根据表employees创建emp02

#7. 删除表emp01

#8. 将表emp02重命名为emp01

#9.在表dept02和emp01中添加新列test_column，并检查所作的操作

#10.直接删除表emp01中的列 department_id
```

答案：

```
#1. 创建数据库test01_office,指明字符集为utf8。并在此数据库下执行下述操作

CREATE DATABASE IF NOT EXISTS test01_office CHARACTER SET 'utf8';

USE test01_office;
```

```
#2. 创建表dept01
/*
字段       类型
id        INT(7)
NAME      VARCHAR(25)
*/

CREATE TABLE dept01(
    id INT(7),
    NAME VARCHAR(25)
);
```

```
#3. 将表departments中的数据插入新表dept02中

CREATE TABLE dept02
AS
SELECT *
FROM atguigudb.departments;
```

```
#4. 创建表emp01
/*
字段          类型
id           INT(7)
first_name   VARCHAR (25)
last_name    VARCHAR(25)
dept_id      INT(7)
*/

CREATE TABLE emp01(
id INT(7),
first_name VARCHAR(25),
last_name VARCHAR(25),
dept_id INT(7)
);
```

```
#5. 将列last_name的长度增加到50
DESC emp01;

ALTER TABLE emp01
MODIFY last_name VARCHAR(50);
```

```
#6. 根据表employees创建emp02
CREATE TABLE emp02
AS
SELECT *
FROM atguigudb.`employees`;

SHOW TABLES FROM test01_office;
```

```
#7. 删除表emp01
DROP TABLE IF EXISTS emp01;
```

```
#8. 将表emp02重命名为emp01
#ALTER TABLE emp02 RENAME TO emp01;
RENAME TABLE emp02 TO emp01;
```

```
#9. 在表dept02和emp01中添加新列test_column，并检查所作的操作
ALTER TABLE emp01 ADD test_column VARCHAR(10);

DESC emp01;

ALTER TABLE dept02 ADD test_column VARCHAR(10);

DESC dept02;
```

```
#10.直接删除表emp01中的列 department_id
ALTER TABLE emp01
DROP COLUMN department_id;
```

## 练习2

```
# 1、创建数据库 test02_market

# 2、创建数据表 customers
```

| 字段名 | 数据类型 |
| --- | --- |
| c_num | int |
| c_name | varchar(50) |
| c_contact | varchar(50) |
| c_city | varchar(50) |
| c_birth | date |

```
# 3、将 c_contact 字段移动到 c_birth 字段后面

# 4、将 c_name 字段数据类型改为 varchar(70)

# 5、将c_contact字段改名为c_phone

# 6、增加c_gender字段到c_name后面，数据类型为char(1)

# 7、将表名改为customers_info

# 8、删除字段c_city
```

答案：

```
#1、创建数据库 test02_market
CREATE DATABASE IF NOT EXISTS test02_market CHARACTER SET 'utf8';

#指定对哪个数据库进行操作
USE test02_market;

#2、创建数据表 customers
CREATE TABLE customers(
    c_num INT ,
    c_name VARCHAR(50),
    c_contact VARCHAR(50),
```

```
    c_city VARCHAR(50),
    c_birth DATE
);

#3、将c_contact字段插入到c_birth字段后面
ALTER TABLE customers MODIFY c_contact VARCHAR(50) AFTER c_birth;

#4、将c_name字段数据类型改为 varchar(70)
ALTER TABLE customers MODIFY c_name VARCHAR(70);

#5、将c_contact字段改名为c_phone
ALTER TABLE customers CHANGE c_contact c_phone VARCHAR(50);

#6、增加c_gender字段到c_name后面，数据类型为char(1)
ALTER TABLE customers ADD c_gender CHAR(1) AFTER c_name;

#7、将表名改为customers_info
RENAME TABLE customers TO customers_info;
#ALTER TABLE customers RENAME TO customers_info;

#8、删除字段c_city
ALTER TABLE customers_info DROP COLUMN c_city ;
```

## 练习3

```
# 1、创建数据库test03_company

# 2、创建表offices
```

| 字段名 | 数据类型 |
| --- | --- |
| officeCode | int |
| city | varchar(30) |
| address | varchar(50) |
| country | varchar(50) |
| postalCode | varchar(25) |

```
# 3、创建表employees
```

| 字段名 | 数据类型 |
| --- | --- |
| empNum | int |
| lastName | varchar(50) |
| firstName | varchar(50) |
| mobile | varchar(25) |
| code | int |
| jobTitle | varchar(50) |
| birth | date |
| note | varchar(255) |
| sex | varchar(5) |

```
# 4、将表employees的mobile字段修改到code字段后面

# 5、将表employees的birth字段改名为birthday

# 6、修改sex字段，数据类型为char(1)

# 7、删除字段note

# 8、增加字段名favoriate_activity，数据类型为varchar(100)

# 9、将表employees的名称修改为 employees_info
```

答案：

```
#1、创建数据库test03_company
CREATE DATABASE IF NOT EXISTS test03_company CHARACTER SET 'utf8';

#指定使用哪个数据库，即下面的sql语句是针对哪个数据库的
USE test03_company;

# 2、创建表offices
CREATE TABLE offices(
    officeCode INT,
    city VARCHAR(30),
    address VARCHAR(50),
    country VARCHAR(50) ,
    postalCode VARCHAR(25)
);

# 3、创建表employees
CREATE TABLE IF NOT EXISTS employees(
    empNum INT,
    lastName VARCHAR(50),
    firstName VARCHAR(50),
    mobile VARCHAR(25),
    `code` INT ,
    jobTitle VARCHAR(50),
    birth DATE,
```

```
    note VARCHAR(255),
    sex VARCHAR(5)
);

# 4、将表employees的mobile字段修改到code字段后面
ALTER TABLE employees
MODIFY mobile VARCHAR(20) AFTER `code`;


# 5、将表employees的birth字段改名为birthday
ALTER TABLE employees CHANGE birth birthday DATE;

# 6、修改sex字段，数据类型为char(1)
ALTER TABLE employees MODIFY sex CHAR(1) ;

# 7、删除字段note
ALTER TABLE employees DROP COLUMN note;


# 8、增加字段名favoriate_activity,数据类型为varchar(100)
ALTER TABLE employees ADD COLUMN favoriate_activity VARCHAR(100);


# 9、将表employees的名称修改为 employees_info
RENAME TABLE employees TO employees_info;

DESC employees_info;
```

# 第11章_数据处理之增删改

讲师：尚硅谷-宋红康（江湖人称：康师傅）

官网： http://www.atguigu.com

---

## 练习1

题目：

```
#1. 创建数据库dbtest11
CREATE DATABASE IF NOT EXISTS dbtest11 CHARACTER SET 'utf8';

#2. 运行以下脚本创建表my_employees
USE dbtest11;
CREATE TABLE my_employees(
    id INT(10),
    first_name VARCHAR(10),
    last_name VARCHAR(10),
    userid VARCHAR(10),
    salary DOUBLE(10,2)
);

CREATE TABLE users(
    id INT,
    userid VARCHAR(10),
    department_id INT
);
#3. 显示表my_employees的结构

#4. 向my_employees表中插入下列数据
ID  FIRST_NAME  LAST_NAME   USERID      SALARY
1   patel       Ralph       Rpatel      895
2   Dancs       Betty       Bdancs      860
3   Biri        Ben         Bbiri       1100
4   Newman      Chad        Cnewman     750
5   Ropeburn    Audrey      Aropebur    1550

#5. 向users表中插入数据
1   Rpatel      10
2   Bdancs      10
3   Bbiri       20
4   Cnewman     30
5   Aropebur    40

#6. 将3号员工的last_name修改为“drelxer”

#7. 将所有工资少于900的员工的工资修改为1000

#8. 将userid为Bbiri的user表和my_employees表的记录全部删除

#9. 删除my_employees、users表所有数据

#10. 检查所作的修正
```

```
#11. 清空表my_employees
```

答案：

```
#1. 创建数据库dbtest11
CREATE DATABASE IF NOT EXISTS dbtest11 CHARACTER SET 'utf8';

#2. 运行以下脚本创建表my_employees
USE dbtest11;
CREATE TABLE my_employees(
    id INT(10),
    first_name VARCHAR(10),
    last_name VARCHAR(10),
    userid VARCHAR(10),
    salary DOUBLE(10,2)
);

CREATE TABLE users(
    id INT,
    userid VARCHAR(10),
    department_id INT

);
#3. 显示表my_employees的结构
DESC my_employees;

#4. 向my_employees表中插入下列数据
ID  FIRST_NAME  LAST_NAME   USERID     SALARY
1   patel       Ralph       Rpatel     895
2   Dancs       Betty       Bdancs     860
3   Biri        Ben         Bbiri      1100
4   Newman      Chad        Cnewman    750
5   Ropeburn    Audrey      Aropebur   1550

#方式一：
INSERT INTO my_employees
VALUES(1,'patel','Ralph','Rpatel',895),
(2,'Dancs','Betty','Bdancs',860),
(3,'Biri','Ben','Bbiri',1100),
(4,'Newman','Chad','Cnewman',750),
(5,'Ropeburn','Audrey','Aropebur',1550);

#DELETE FROM my_employees;

#方式二：
INSERT INTO my_employees
SELECT 1,'patel','Ralph','Rpatel',895 UNION ALL
SELECT 2,'Dancs','Betty','Bdancs',860 UNION ALL
SELECT 3,'Biri','Ben','Bbiri',1100 UNION ALL
SELECT 4,'Newman','Chad','Cnewman',750 UNION ALL
SELECT 5,'Ropeburn','Audrey','Aropebur',1550;


#5. 向users表中插入数据
1   Rpatel      10
2   Bdancs      10
```

```
3   Bbiri       20
4   Cnewman     30
5   Aropebur    40

INSERT INTO users VALUES
(1,'Rpatel',10),
(2,'Bdancs',10),
(3,'Bbiri',20),
(4,'Cnewman',30),
(5,'Aropebur',40)

#6. 将3号员工的last_name修改为“drelxer”
UPDATE my_employees SET last_name='drelxer' WHERE id = 3;


#7. 将所有工资少于900的员工的工资修改为1000
UPDATE my_employees SET salary=1000 WHERE salary<900;


#8. 将userid为Bbiri的user表和my_employees表的记录全部删除
DELETE u,e
FROM users u
JOIN my_employees e ON u.`userid`=e.`Userid`
WHERE u.`userid`='Bbiri';


#9. 删除my_employees、users表所有数据
DELETE FROM my_employees;

DELETE FROM users;


#10. 检查所作的修正
SELECT * FROM my_employees;
SELECT * FROM users;

#11. 清空表my_employees
TRUNCATE TABLE my_employees;
```

## 练习2

```
# 1. 使用现有数据库dbtest11

# 2. 创建表格pet
```

| 字段名 | 字段说明 | 数据类型 |
| --- | --- | --- |
| name | 宠物名称 | VARCHAR(20) |
| owner | 宠物主人 | VARCHAR(20) |
| species | 种类 | VARCHAR(20) |
| sex | 性别 | CHAR(1) |
| birth | 出生日期 | YEAR |
| death | 死亡日期 | YEAR |

```
# 3. 添加记录
```

| name | owner | species | sex | birth | death |
| --- | --- | --- | --- | --- | --- |
| Fluffy | harold | Cat | f | 2003 | 2010 |
| Claws | gwen | Cat | m | 2004 | |
| Buffy | | Dog | f | 2009 | |
| Fang | benny | Dog | m | 2000 | |
| bowser | diane | Dog | m | 2003 | 2009 |
| Chirpy | | Bird | f | 2008 | |

```
# 4. 添加字段:主人的生日owner_birth DATE类型。

# 5. 将名称为Claws的猫的主人改为kevin

# 6. 将没有死的狗的主人改为duck

# 7. 查询没有主人的宠物的名字;

# 8. 查询已经死了的cat的姓名，主人，以及去世时间;

# 9. 删除已经死亡的狗

# 10. 查询所有宠物信息
```

答案:

```
# 1. 使用现有数据库dbtest11
USE dbtest11;

# 2. 创建表格pet
CREATE TABLE pet(
    `name` VARCHAR(20),
    `owner` VARCHAR(20),
    species VARCHAR(20),
    sex CHAR(1),
    birth YEAR,
    death YEAR
);
```

```
# 3. 添加记录
INSERT INTO pet VALUES('Fluffy','harold','Cat','f','2013','2010');
INSERT INTO pet(`name`,`owner`,species,sex,Birth)
VALUES('Claws','gwen','Cat','m','2014');
INSERT INTO pet(`name`,species,sex,Birth) VALUES('Buffy','Dog','f','2009');
INSERT INTO pet(`name`,`owner`,species,sex,Birth)
VALUES('Fang','benny','Dog','m','2000');
INSERT INTO pet VALUES('bowser','diane','Dog','m','2003','2009');
INSERT INTO pet(`name`,species,sex,birth) VALUES('Chirpy','Bird','f','2008');

# 4. 添加字段:主人的生日owner_birth DATE类型
ALTER TABLE pet ADD COLUMN owner_birth DATE;

# 5. 将名称为Claws的猫的主人改为kevin
UPDATE pet SET `owner`='kevin' WHERE `name`='Claws' AND species='Cat';

# 6. 将没有死的狗的主人改为duck
UPDATE pet SET `owner`='duck' WHERE species='Dog' AND death IS NULL;

# 7. 查询没有主人的宠物的名字
SELECT `name` FROM pet WHERE `owner` IS NULL;

# 8. 查询已经死了的cat的姓名，主人，以及去世时间
SELECT `name`,`owner`,death FROM pet WHERE death IS NOT NULL AND species = 'Cat';

# 9. 删除已经死亡的狗
DELETE FROM pet WHERE death IS NOT NULL and species = 'Dog';

# 10. 查询所有宠物信息
SELECT * FROM pet;
```

## 练习3

```
# 1. 使用已有的数据库dbtest11

# 2. 创建表employee，并添加记录
```

| id | name | sex | tel | addr | salary |
|---|---|---|---|---|---|
| 10001 | 张一一 | 男 | 13456789000 | 山东青岛 | 1001.58 |
| 10002 | 刘小红 | 女 | 13454319000 | 河北保定 | 1201.21 |
| 10003 | 李四 | 男 | 0751-1234567 | 广东佛山 | 1004.11 |
| 10004 | 刘小强 | 男 | 0755-5555555 | 广东深圳 | 1501.23 |
| 10005 | 王艳 | 女 | 020-1232133 | 广东广州 | 1405.16 |

```
# 3. 查询出薪资在1200~1300之间的员工信息。

# 4. 查询出姓“刘”的员工的工号，姓名，家庭住址。

# 5. 将“李四”的家庭住址改为“广东韶关”

# 6. 查询出名字中带“小”的员工
```

答案:

```
# 1. 使用dbtest11数据库
USE dbtest11;

# 2. 创建employee表
CREATE TABLE employee(
    id INT,
    `name` VARCHAR(20),
    sex VARCHAR(2),
    tel VARCHAR(20),
    addr VARCHAR(50),
    salary DOUBLE
);

# 添加信息
INSERT INTO employee(id,`name`,sex,tel,addr,salary)VALUES
(10001,'张一一','男','13456789000','山东青岛',1001.58),
(10002,'刘小红','女','13454319000','河北保定',1201.21),
(10003,'李四','男','0751-1234567','广东佛山',1004.11),
(10004,'刘小强','男','0755-5555555','广东深圳',1501.23),
(10005,'王艳','男','020-1232133','广东广州',1405.16);


# 3. 查询出薪资在1200~1300之间的员工信息
SELECT * FROM employee WHERE salary BETWEEN 1200 AND 1300;

# 4. 查询出姓“刘”的员工的工号，姓名，家庭住址
SELECT id,name,addr FROM employee WHERE `name` LIKE '刘%';

# 5. 将“李四”的家庭住址改为“广东韶关”
UPDATE employee SET addr='广东韶关' WHERE `name`='李四';

# 6. 查询出名字中带“小”的员工
SELECT * FROM employee WHERE `name` LIKE '%小%';
```

# 第12章_MySQL数据类型精讲

讲师：尚硅谷-宋红康（江湖人称：康师傅）

官网： http://www.atguigu.com

## 1. 掌握MySQL规范的各种数据类型

## 2. 熟悉数据类型常设置的属性

## 3. 掌握不同的类型的使用场景

# 第13章_约束

讲师：尚硅谷-宋红康（江湖人称：康师傅）

官网： http://www.atguigu.com

---

## 基础练习：

### 练习1

已经存在数据库test04_emp，两张表emp2和dept2

```
CREATE DATABASE test04_emp;

use test04_emp;

CREATE TABLE emp2(
id INT,
emp_name VARCHAR(15)
);

CREATE TABLE dept2(
id INT,
dept_name VARCHAR(15)
);
```

题目：

```
#1.向表emp2的id列中添加PRIMARY KEY约束

#2. 向表dept2的id列中添加PRIMARY KEY约束

#3. 向表emp2中添加列dept_id，并在其中定义FOREIGN KEY约束，与之相关联的列是dept2表中的id列。
```

答案：

```
#1.向表emp2的id列中添加PRIMARY KEY约束

ALTER TABLE emp2 MODIFY COLUMN id INT PRIMARY KEY;
ALTER TABLE emp2 ADD PRIMARY KEY(id);

#2. 向表dept2的id列中添加PRIMARY KEY约束
ALTER TABLE dept2 MODIFY COLUMN id INT PRIMARY KEY;
ALTER TABLE dept2 ADD PRIMARY KEY(id);

#3. 向表emp2中添加列dept_id，并在其中定义FOREIGN KEY约束，与之相关联的列是dept2表中的id列。
ALTER TABLE emp2 ADD COLUMN dept_id INT;
ALTER TABLE emp2 ADD CONSTRAINT fk_emp2_deptid FOREIGN KEY(dept_id) REFERENCES
dept2(id);
```

## 练习2

承接《第11章_数据处理之增删改》的综合案例。

```
# 1、创建数据库test01_library

# 2、创建表 books，表结构如下：
```

| 字段名 | 字段说明 | 数据类型 |
| --- | --- | --- |
| id | 书编号 | INT |
| name | 书名 | VARCHAR(50) |
| authors | 作者 | VARCHAR(100) |
| price | 价格 | FLOAT |
| pubdate | 出版日期 | YEAR |
| note | 说明 | VARCHAR(100) |
| num | 库存 | INT |

```
# 3、使用ALTER语句给books按如下要求增加相应的约束
```

| 字段名 | 字段说明 | 数据类型 | 主键 | 外键 | 非空 | 唯一 | 自增 |
| --- | --- | --- | --- | --- | --- | --- | --- |
| id | 书编号 | INT(11) | 是 | 否 | 是 | 是 | 是 |
| name | 书名 | VARCHAR(50) | 否 | 否 | 是 | 否 | 否 |
| authors | 作者 | VARCHAR(100) | 否 | 否 | 是 | 否 | 否 |
| price | 价格 | FLOAT | 否 | 否 | 是 | 否 | 否 |
| pubdate | 出版日期 | YEAR | 否 | 否 | 是 | 否 | 否 |
| note | 说明 | VARCHAR(100) | 否 | 否 | 否 | 否 | 否 |
| num | 库存 | INT(11) | 否 | 否 | 是 | 否 | 否 |

答案：

```
# 1、2、略

# 3、使用ALTER语句给books按如下要求增加相应的约束
#给id增加主键约束
ALTER TABLE books ADD PRIMARY KEY(id);

#给id字段增加自增约束
ALTER TABLE books MODIFY id INT AUTO_INCREMENT;

#给name等字段增加非空约束
ALTER TABLE books name VARCHAR(50) NOT NULL;
ALTER TABLE books `authors` VARCHAR(100) NOT NULL;
ALTER TABLE books price FLOAT NOT NULL;
ALTER TABLE books pubdate DATE NOT NULL;
```

```
ALTER TABLE books num INT NOT NULL;
```

## 练习3

题目：

```
#1. 创建数据库test04_company

#2. 按照下表给出的表结构在test04_company数据库中创建两个数据表offices和employees
```

- offices表：

| 字段名 | 数据类型 | 主键 | 外键 | 非空 | 唯一 | 自增 |
| --- | --- | --- | --- | --- | --- | --- |
| officeCode | INT(10) | 是 | 否 | 是 | 是 | 否 |
| city | VARCHAR(50) | 否 | 否 | 是 | 否 | 否 |
| address | VARCHAR(50) | 否 | 否 | 否 | 否 | 否 |
| country | VARCHAR(50) | 否 | 否 | 是 | 否 | 否 |
| postalCode | VARCHAR(15) | 否 | 否 | 否 | 是 | 否 |

- employees表：

| 字段名 | 数据类型 | 主键 | 外键 | 非空 | 唯一 | 自增 |
| --- | --- | --- | --- | --- | --- | --- |
| employeeNumber | INT(11) | 是 | 否 | 是 | 是 | 是 |
| lastName | VARCHAR(50) | 否 | 否 | 是 | 否 | 否 |
| firstName | VARCHAR(50) | 否 | 否 | 是 | 否 | 否 |
| mobile | VARCHAR(25) | 否 | 否 | 否 | 是 | 否 |
| officeCode | INT(10) | 否 | 是 | 是 | 否 | 否 |
| jobTitle | VARCHAR(50) | 否 | 否 | 是 | 否 | 否 |
| birth | DATETIME | 否 | 否 | 是 | 否 | 否 |
| note | VARCHAR(255) | 否 | 否 | 否 | 否 | 否 |
| sex | VARCHAR(5) | 否 | 否 | 否 | 否 | 否 |

```
#3. 将表employees的mobile字段修改到officeCode字段后面

#4. 将表employees的birth字段改名为employee_birth

#5. 修改sex字段，数据类型为CHAR(1)，非空约束

#6. 删除字段note

#7. 增加字段名favoriate_activity，数据类型为VARCHAR(100)

#8. 将表employees名称修改为employees_info
```

答案：

```
#1. 创建数据库test04_company
CREATE DATABASE test04_company;

#2. 按照下表给出的表结构在test04_company数据库中创建两个数据表offices和employees
USE test04_company;

CREATE TABLE offices(
officeCode INT(10) ,
city VARCHAR(50) NOT NULL,
```

```
address VARCHAR(50),
country VARCHAR(50) NOT NULL,
postalCode VARCHAR(15) UNIQUE,
PRIMARY KEY(officeCode)
);

CREATE TABLE employees(
employeeNumber INT(11) PRIMARY KEY AUTO_INCREMENT,
lastName VARCHAR(50) NOT NULL,
firstName VARCHAR(50) NOT NULL,
mobile VARCHAR(25) UNIQUE,
officeCode INT(10) NOT NULL,
jobTitle VARCHAR(50) NOT NULL,
birth DATETIME NOT NULL,
note VARCHAR(255),
sex VARCHAR(5),
CONSTRAINT fk_emp_ofCode FOREIGN KEY(officeCode) REFERENCES offices(officeCode)
);

#3. 将表employees的mobile字段修改到officeCode字段后面
ALTER TABLE employees MODIFY mobile VARCHAR(25) AFTER officeCode;

#4. 将表employees的birth字段改名为employee_birth
ALTER TABLE employees CHANGE birth employee_birth DATETIME;

#5. 修改sex字段，数据类型为CHAR(1)，非空约束
ALTER TABLE employees MODIFY sex CHAR(1) NOT NULL;

#6. 删除字段note
ALTER TABLE employees DROP COLUMN note;

#7. 增加字段名favoriate_activity，数据类型为VARCHAR(100)
ALTER TABLE employees ADD favoriate_activity VARCHAR(100);

#8. 将表employees名称修改为employees_info
ALTER TABLE employees RENAME employees_info;
```

# 拓展练习：

## 练习1

创建数据库test04_Market，在test04_Market中创建数据表customers。customers表结构如下所示，按以下要求进行操作。

| 字段名 | 数据类型 | 主键 | 外键 | 非空 | 唯一 | 自增 |
|---|---|---|---|---|---|---|
| c_num | INT(11) | 是 | 否 | 是 | 是 | 是 |
| c_name | VARCHAR(50) | 否 | 否 | 否 | 否 | 否 |
| c_contact | VARCHAR(50) | 否 | 否 | 否 | 否 | 否 |
| c_city | VARCHAR(50) | 否 | 否 | 否 | 否 | 否 |
| c_birth | DATETIME | 否 | 否 | 是 | 否 | 否 |

（1）创建数据库test04_Market。 （2）创建数据表customers，在c_num字段上添加主键约束和自增约束，在c_birth字段上添加非空约束。 （3）将c_contact字段插入c_birth字段后面。 （4）将c_name字段数据类型改为VARCHAR(70)。 （5）将c_contact字段改名为c_phone。 （6）增加c_gender字段，数据类型为CHAR(1)。 （7）将表名修改为customers_info。 （8）删除字段c_city。

在test04_Market中创建数据表orders。orders表结构如下所示，按以下要求进行操作。

| 字段名 | 数据类型 | 主键 | 外键 | 非空 | 唯一 | 自增 |
| --- | --- | --- | --- | --- | --- | --- |
| o_num | INT(11) | 是 | 否 | 是 | 是 | 是 |
| o_date | DATE | 否 | 否 | 否 | 否 | 否 |
| c_id | INT(11) | 否 | 是 | 否 | 否 | 否 |

（1）创建数据表orders，在o_num字段上添加主键约束和自增约束，在c_id字段上添加外键约束，关联customers表中的主键c_num。 （2）删除orders表的外键约束，然后删除表customers。

## 练习2

创建数据表pet，并对表进行插入、更新与删除操作。pet表结构如下表所示。 （1）首先创建数据表pet，使用不同的方法将表记录插入到pet表中。 （2）使用UPDATE语句将名称为Fang的狗的主人改为Kevin。 （3）将没有主人的宠物的owner字段值都改为Duck。 （4）删除已经死亡的宠物记录。 （5）删除所有表中的记录。

pet表结构：

| 字段名 | 字段说明 | 数据类型 | 主键 | 外键 | 非空 | 唯一 | 自增 |
| --- | --- | --- | --- | --- | --- | --- | --- |
| name | 宠物名称 | VARCHAR(20) | 否 | 否 | 是 | 否 | 否 |
| owner | 宠物主人 | VARCHAR(20) | 否 | 否 | 否 | 否 | 否 |
| species | 种类 | VARCHAR(20) | 否 | 否 | 是 | 否 | 否 |
| sex | 性别 | CHAR(1) | 否 | 否 | 是 | 否 | 否 |
| birth | 出生日期 | YEAR | 否 | 否 | 是 | 否 | 否 |
| death | 死亡日期 | YEAR | 否 | 否 | 否 | 否 | 否 |

pet表中记录：

| name | owner | species | sex | birth | death |
| --- | --- | --- | --- | --- | --- |
| Fluffy | Harold | cat | f | 2003 | 2010 |
| Claws | Gwen | cat | m | 2004 | NULL |
| Buffy | NULL | dog | f | 2009 | NULL |
| Fang | Benny | dog | m | 2000 | NULL |
| Bowser | Diane | dog | m | 2003 | 2009 |
| Chirpy | NULL | bird | f | 2008 | NULL |

## 练习3

1、创建数据库：test_company

2、在此数据库下创建如下3表，数据类型，宽度，是否为空根据实际情况自己定义。

A. `部门表（department）`：部门编号（depid），部门名称（depname），部门简介（deinfo）；其中部门编号为主键。

B. `雇员表（emoloyee）`：雇员编号（empid），姓名（name），性别（sex），职称（title），出生日期（birthday），所在部门编号（depid）；其中

- 雇员编号为主键；
- 部门编号为外键，外键约束等级为（on update cascade 和on delete set null）；

- 性别默认为男；

C. `工资表（salary）`：雇员编号（empid），基本工资（basesalary），职务工资（titlesalary），扣除（deduction）。其中雇员编号为主键。

3、给工资表（salary）的雇员编号（empid）增加外键约束，外键约束等级为（on update cascade 和on delete cascade）

4、添加数据如下：

部门表：

| 部门编号 | 部门名称 | 部门简介 |
| --- | --- | --- |
| 111 | 生产部 | Null |
| 222 | 销售部 | Null |
| 333 | 人事部 | 人力资源管理 |

雇员表：

| 雇员编号 | 姓名 | 性别 | 职称 | 出生日期 | 所在部门编号 |
| --- | --- | --- | --- | --- | --- |
| 1001 | 张三 | 男 | 高级工程师 | 1975-1-1 | 111 |
| 1002 | 李四 | 女 | 助工 | 1985-1-1 | 111 |
| 1003 | 王五 | 男 | 工程师 | 1978-11-11 | 222 |
| 1004 | 张六 | 男 | 工程师 | 1999-1-1 | 222 |

工资表：

| 雇员编号 | 基本工资 | 职务工资 | 扣除 |
| --- | --- | --- | --- |
| 1001 | 2200 | 1100 | 200 |
| 1002 | 1200 | 200 | NULL |
| 1003 | 2900 | 700 | 200 |
| 1004 | 1950 | 700 | 150 |

5、查询出每个雇员的雇员编号，姓名，职称，所在部门名称，应发工资（基本工资+职务工资），实发工资（基本工资+职务工资-扣除）。

6、查询销售部门的雇员姓名及其基本工资

7、查询姓“张”且年龄小于40的员工的全部信息和年龄

8、查询所有男员工的基本工资和职务工资

9、查询基本工资低于2000的员工姓名和职称、所在部门名称

10、查询员工总数

11、查询部门总数

12、查询应发工资的平均工资和最高工资、最低工资

13、按照部门统计应发工资的平均工资

14、找出部门基本工资的平均工资低于2000的

15、按照员工编号、姓名、基本工资、职务工资、扣除，并按照职务升序排列，如果职务工资相同，再按照基本工资升序排列

16、查询员工编号、姓名，出生日期，及年龄段。其中，如果80年之前出生的，定为”老年“；80后定为”中年“，90后定为”青壮年“

17、查询所有的员工信息，和他所在的部门名称

18、查询所有部门信息，和该部门的员工信息

19、查询所有职位中含“工程师”的男员工的人数

20、查询每个部门的男生和女生的人数和平均基本工资

```
#创建数据库：test_company
CREATE DATABASE test_company;

#使用数据库test_company
USE test_company;

#创建部门表（department）
CREATE TABLE department(
    depid INT PRIMARY KEY,
    depname VARCHAR(20) NOT NULL,
    deinfo VARCHAR(50)
);

#创建雇员表（emoloyee）
CREATE TABLE employee(
    empid INT PRIMARY KEY,
    `name` VARCHAR(20) NOT NULL,
    sex CHAR NOT NULL DEFAULT '男',
    title VARCHAR(20) NOT NULL,
    birthday DATE,
    depid INT,
    FOREIGN KEY(depid) REFERENCES department(depid) ON UPDATE CASCADE ON DELETE SET
NULL
);

#创建工资表（salary）
CREATE TABLE salary(
    empid INT PRIMARY KEY,
    basesalary DOUBLE,
    titlesalary DOUBLE,
    deduction DOUBLE
);

#给工资表（salary）的雇员编号（empid）增加外键约束，外键约束等级为（on update cascade 和on
delete cascade）
ALTER TABLE salary ADD FOREIGN KEY empid REFERENCES employee(empid) ON UPDATE CASCADE
ON DELETE CASCADE;

#添加部门表数据
INSERT INTO department VALUES
(111,'生产部',NULL),
(222,'销售部',NULL),
```

```
(333,'人事部','人力资源管理');

#添加雇员表
INSERT INTO employee VALUES
(1001,'张三',DEFAULT,'高级工程师','1975-1-1',111),
(1002,'李四','女','助工','1985-1-1',111),
(1003,'王五','男','工程师','1978-11-11',222),
(1004,'张六',DEFAULT,'工程师','1999-1-1',222);

#添加工资表
INSERT INTO salary VALUES
(1001,2200,1100,200),
(1002,1200,200,NULL),
(1003,2900,700,200),
(1004,1950,700,150);


/*
查询出每个雇员的雇员编号，姓名，职称，所在部门名称，
应发工资（基本工资+职务工资），
实发工资（基本工资+职务工资-扣除）。
*/
SELECT employee.empid,`name`,title,depname,
basesalary+titlesalary AS "应发工资",
basesalary+titlesalary-IFNULL(deduction,0) AS "实发工资"
FROM department INNER JOIN employee INNER JOIN salary
ON department.depid = employee.depid AND employee.empid = salary.empid;

#查询销售部门的雇员姓名及其基本工资
SELECT `name`,basesalary
FROM department INNER JOIN employee INNER JOIN salary
ON department.depid = employee.depid AND employee.empid = salary.empid
WHERE department.depname = '销售部';

#查询姓“张”且年龄小于40的员工的全部信息和年龄
SELECT *,YEAR(CURRENT_DATE())-YEAR(birthday) AS "年龄"
FROM employee
WHERE `name` LIKE '张%' AND YEAR(CURRENT_DATE())-YEAR(birthday)<40;

#查询所有男员工的基本工资和职务工资
SELECT basesalary,titlesalary
FROM employee INNER JOIN salary
ON employee.empid = salary.empid
WHERE employee.sex = '男';

#查询基本工资低于2000的员工姓名和职称、所在部门名称
SELECT `name`,title,depname
FROM department INNER JOIN employee INNER JOIN salary
ON department.depid = employee.depid AND employee.empid = salary.empid
WHERE basesalary < 2000;

#查询员工总数
SELECT COUNT(*) FROM employee;

#查询部门总数
SELECT COUNT(*) FROM department;

#查询应发工资的平均工资和最高应发工资、最低应发工资
```

```sql
SELECT AVG(basesalary+titlesalary) AS "平均应发工资",
    MAX(basesalary+titlesalary) AS "最高应发工资",
    MIN(basesalary+titlesalary) AS "最低应发工资"
FROM salary;

#按照部门统计应发工资的平均工资
SELECT depid,AVG(basesalary+titlesalary)
FROM employee INNER JOIN salary
ON employee.`empid` = salary.`empid`
GROUP BY employee.`depid`;

#找出部门基本工资的平均工资低于2000的
SELECT depid,AVG(basesalary)
FROM employee INNER JOIN salary
ON employee.`empid` = salary.`empid`
GROUP BY employee.`depid`
HAVING AVG(basesalary)<2000;


#按照员工编号、姓名、基本工资、职务工资、扣除，
#并按照职务升序排列，如果职务工资相同，再按照基本工资升序排列
SELECT emp.empid,`name`,basesalary,titlesalary,deduction
FROM employee emp INNER JOIN salary
ON emp.`empid` = salary.`empid`
ORDER BY emp.`title` ASC , basesalary ASC;

#查询员工编号、姓名，出生日期，及年龄段，其中
·#如果80年之前出生的，定为”老年“；80后定为”中年“，90后定为”青壮年“
SELECT empid,`name`,birthday,
 CASE WHEN YEAR(birthday)<1980 THEN '老年'
      WHEN YEAR(birthday)<1990 THEN '中年'
 ELSE '青壮年' END "年龄段"
FROM employee;

#查询所有的员工信息，和他所在的部门名称
SELECT emp.*,depname
FROM employee emp LEFT JOIN department dep
ON emp.`depid` = dep.`depid`;

#查询所有部门信息，和该部门的员工信息
SELECT dep.*,emp.*
FROM employee emp RIGHT JOIN department dep
ON emp.`depid` = dep.`depid`;


#查询所有职位中含“工程师”的男员工的人数
SELECT COUNT(*) FROM employee WHERE sex='男' AND title LIKE '%工程师%';

#查询每个部门的男生和女生的人数和平均基本工资
SELECT dep.depid,sex,COUNT(*),AVG(basesalary)
FROM department dep INNER JOIN employee INNER JOIN salary
ON dep.depid = employee.depid AND employee.empid = salary.empid
GROUP BY dep.depid,sex;
```

## 练习4

1、创建一个数据库：test_school

2、创建如下表格

表1 Department表的定义

| 字段名 | 字段描述 | 数据类型 | 主键 | 外键 | 非空 | 唯一 |
| --- | --- | --- | --- | --- | --- | --- |
| DepNo | 部门号 | int(10) | 是 | 否 | 是 | 是 |
| DepName | 部门名称 | varchar(20) | 否 | 否 | 是 | 否 |
| DepNote | 部门备注 | Varchar(50) | 否 | 否 | 否 | 否 |

表2 Teacher表的定义

| 字段名 | 字段描述 | 数据类型 | 主键 | 外键 | 非空 | 唯一 |
| --- | --- | --- | --- | --- | --- | --- |
| Number | 教工号 | int | 是 | 否 | 是 | 是 |
| Name | 姓名 | varchar(30) | 否 | 否 | 是 | 否 |
| Sex | 性别 | varchar(4) | 否 | 否 | 否 | 否 |
| Birth | 出生日期 | date | 否 | 否 | 否 | 否 |
| DepNo | 部门号 | int | 否 | 是 | 否 | 否 |
| Salary | 工资 | float | 否 | 否 | 否 | 否 |
| Address | 家庭住址 | varchar(100) | 否 | 否 | 否 | 否 |

3、添加记录

| DepNo | DepName | DepNote |
| --- | --- | --- |
| 601 | 软件技术系 | 软件技术等专业 |
| 602 | 网络技术系 | 多媒体技术等专业 |
| 603 | 艺术设计系 | 广告艺术设计等专业 |
| 604 | 管理工程系 | 连锁经营管理等专业 |

| Number | Name | Sex | Birth | DepNo | Salary | Address |
| --- | --- | --- | --- | --- | --- | --- |
| 2001 | Tom | 女 | 1970-01-10 | 602 | 4500 | 四川省绵阳市 |
| 2002 | Lucy | 男 | 1983-12-18 | 601 | 2500 | 北京市昌平区 |
| 2003 | Mike | 男 | 1990-06-01 | 604 | 1500 | 重庆市渝中区 |
| 2004 | James | 女 | 1980-10-20 | 602 | 3500 | 四川省成都市 |
| 2005 | Jack | 男 | 1975-05-30 | 603 | 1200 | 重庆市南岸区 |

4、用SELECT语句查询Teacher表的所有记录。

5、找出所有其家庭地址中含有“北京”的教师的教工号及部门名称，要求显示结果中各列标题用中文别名表示。

6、获得Teacher表中工资最高的教工号和姓名。

7、找出所有收入在2500～4000之间的教工号。

8、查找在网络技术系工作的教师的姓名、性别和工资。

```
#创建一个数据库：test_school
CREATE DATABASE test_school;

#使用数据库
USE test_school;

#创建表格
-- 部门信息表Department
CREATE TABLE Department(
    DepNo INT(10) PRIMARY KEY,
    DepName VARCHAR(20) NOT NULL,
    DepNote VARCHAR(50)
);
-- 创建数据表Teacher
CREATE TABLE Teacher(
    Number INT PRIMARY KEY,
    `Name` VARCHAR(30) UNIQUE,
    Sex VARCHAR(4),
    Birth DATE,
    DepNo INT,
    Salary FLOAT,
    Address VARCHAR(100),
    FOREIGN KEY (DepNo) REFERENCES Department(DepNo)
);
-- 将表4的内容插入Department表中
INSERT INTO Department VALUES (601,'软件技术系','软件技术等专业');
INSERT INTO Department VALUES (602,'网络技术系','多媒体技术等专业');
INSERT INTO Department VALUES (603,'艺术设计系','广告艺术设计等专业');
INSERT INTO Department VALUES (604,'管理工程系','连锁经营管理等专业');
-- 将表3的内容插入Teacher表中。
INSERT INTO Teacher VALUES(2001,'Tom','女','1970-01-10',602,4500,'四川省绵阳市');
INSERT INTO Teacher VALUES(2002,'Lucy','男','1983-12-18',601,2500,'北京市昌平区');
INSERT INTO Teacher VALUES(2003,'Mike','男','1990-06-01',604,1500,'重庆市渝中区');
INSERT INTO Teacher VALUES(2004,'James','女','1980-10-20',602,3500,'四川省成都市');
INSERT INTO Teacher VALUES(2005,'Jack','男','1975-05-30',603,1200,'重庆市南岸区');

#用SELECT语句查询Teacher表的所有记录。
SELECT * FROM teacher;

#找出所有其家庭地址中含有“北京”的教师的教工号及部门名称，要求显示结果中各列标题用中文表示。
SELECT number AS 教工号,Teacher.depno AS 部门名称
FROM Teacher INNER JOIN Department
ON Teacher.DepNo = Department.DepNo
WHERE address LIKE '%北京%';

#获得Teacher表中工资最高的教工号和姓名。
SELECT number,`name` FROM teacher WHERE salary = (SELECT MAX(salary) FROM teacher);
SELECT number,`name` FROM teacher ORDER BY salary DESC LIMIT 0,1;
```

```
#找出所有收入在2500～4000之间的教工号。
SELECT number FROM teacher WHERE salary BETWEEN 2500 AND 4000;

#查找在网络技术系工作的教师的姓名、性别和工资。
SELECT `name`,sex,salary FROM teacher
WHERE depno=(SELECT depno FROM department WHERE depname='网络技术系');

SELECT `name`,sex,salary
FROM teacher INNER JOIN department
ON teacher.depno = department.depno
WHERE depname ='网络技术系';
```

## 练习5

1、建立数据库test_student

2、建立以下三张表，并插入记录

Table:Classes

| 专业 | 班级 | 姓名 | 姓别 | 座位 |
|---|---|---|---|---|
| 计算机网络 | 1班 | 张三 | 男 | 8 |
| 软件工程 | 2班 | 李四 | 男 | 12 |
| 计算机维护 | 1班 | 王五 | 男 | 9 |
| 计算机网络 | 2班 | LILY | 女 | 15 |
| 软件工程 | 1班 | 小强 | 男 | 20 |
| 计算机维护 | 1班 | CoCo | 女 | 18 |

Table:Score

| 姓名 | 英语 | 数学 | 语文 |
|---|---|---|---|
| 张三 | 65 | 75 | 98 |
| 李四 | 87 | 45 | 86 |
| 王五 | 98 | 85 | 65 |
| LILY | 75 | 86 | 87 |
| 小强 | 85 | 60 | 58 |
| CoCo | 96 | 87 | 70 |

Table: Records

| 姓名 | 记录 |
| --- | --- |
| 小强 | 迟到 |
| 小强 | 事假 |
| 李四 | 旷课 |
| 李四 | 旷课 |
| 李四 | 迟到 |
| CoCo | 病假 |
| LILY | 事假 |

3、写出将张三的语文成绩修改为88的SQL语句。

4、搜索出计算机维护1班各门课程的平均成绩。

5、搜索科目有不及格的人的名单。

6、查询记录2次以上的学生的姓名和各科成绩。

```
#1、建立数据库test_student
CREATE DATABASE test_student;

#使用数据库
USE test_student;

#2、创建表格并添加记录
CREATE TABLE Classes(
    Pro_name VARCHAR(20) NOT NULL,
    Grade VARCHAR(10) NOT NULL,
    `name` VARCHAR(10) NOT NULL,
    sex VARCHAR(4) NOT NULL,
    seat INT(10) NOT NULL UNIQUE
);
CREATE TABLE Score(
    `name` VARCHAR(10) NOT NULL,
    En_score INT(10) NOT NULL,
    Ma_score INT(10) NOT NULL,
    Ch_score INT(10) NOT NULL
);
CREATE TABLE Records(
    `name` VARCHAR(10) NOT NULL,
    record VARCHAR(10)
);

-- 向classes中添加数据
INSERT INTO classes VALUES('计算机网络','1班','张三','男',8);
INSERT INTO classes VALUES('软件工程','2班','李四','男',12);
INSERT INTO classes VALUES('计算机维护','1班','王五','男',9);
INSERT INTO classes VALUES('计算机网络','2班','LILY','女',15);
INSERT INTO classes VALUES('软件工程','1班','小强','男',20);
INSERT INTO classes VALUES('计算机维护','1班','CoCo','女',18);

-- 向score中添加数据
INSERT INTO Score VALUES('张三',65,75,98);
```

```
INSERT INTO Score VALUES('李四',87,45,86);
INSERT INTO Score VALUES('王五',98,85,65);
INSERT INTO Score VALUES('LILY',75,86,87);
INSERT INTO Score VALUES('小强',85,60,58);
INSERT INTO Score VALUES('CoCo',96,87,70);

-- 向records中添加数据
INSERT INTO records VALUES('小强','迟到');
INSERT INTO records VALUES('小强','事假');
INSERT INTO records VALUES('李四','旷课');
INSERT INTO records VALUES('李四','旷课');
INSERT INTO records VALUES('李四','迟到');
INSERT INTO records VALUES('CoCo','病假');
INSERT INTO records VALUES('LILY','事假');

#3、写出将张三的语文成绩修改为88的SQL语句。
UPDATE score SET ch_score=88 WHERE `name`='张三';

#4、搜索出计算机维护1班各门课程的平均成绩。
SELECT AVG(en_score),AVG(ma_score),AVG(ch_score) FROM score
WHERE `name` IN (SELECT `name` FROM classes WHERE Pro_name='计算机维护' AND grade='1
班');

#5、搜索科目有不及格的人的名单。
SELECT `name` FROM score WHERE en_score<60 OR ma_score<60 OR ch_score<60;

#6、查询记录2次以上的学生的姓名和各科成绩。
SELECT *
FROM score INNER JOIN
(SELECT `name`,COUNT(*) FROM Records GROUP BY `name` HAVING COUNT(*)>2) temp
ON score.name = temp.name;
```

## 练习6

1、建立数据库：test_xuankedb

2、建立如下三张表：

学生表Student由学号(Sno)、姓名(Sname)、性别(Ssex)、年龄(Sage)、所在系(Sdept)五个字段，Sno 为关键字。

课程表Course由课程号(Cno)、课程名(Cname)、选修课号(Cpno)、学分(Ccredit)四个字段，Cno为关键字。

成绩表SG由学号(Sno)、课程号(Cno)、成绩(Grade)三个字段，(SNO, CNO)为关键字。

3、向Student表增加“入学时间(Scome)”列，其数据类型为日期型。

4、查询选修了3号课程的学生的学号及其成绩，查询结果按分数的降序排列。

5、查询学习1号课程的学生最高分数、平均成绩。

6、查询与“李洋”在同一个系学习的学生。

7、将计算机系全体学生的成绩置零。

8、删除学生表中学号为05019的学生记录。

9、删除计算机系所有学生的成绩记录。

```sql
-- 1、创建一个数据库：test_xuankedb
CREATE DATABASE test_xuankedb;

-- 使用数据库
USE test_xuankedb;

-- 2、创建学生表
CREATE TABLE student(
    sno INT(10) PRIMARY KEY,
    sname VARCHAR(10),
    ssex VARCHAR(10),
    sage INT(10),
    sdept VARCHAR(40)
);

-- 创建课程表
CREATE TABLE course(
    cno INT(10) PRIMARY KEY,
    cname VARCHAR(20),
    cpno VARCHAR(40),
    ccredit INT(20)
);

-- 创建成绩表
CREATE TABLE sg(
    sno INT(10),
    cno INT(10),
    grade INT(3),
    PRIMARY KEY(sno,cno),
    CONSTRAINT stu_s_sno_fk FOREIGN KEY (sno) REFERENCES student(sno),
    CONSTRAINT cou_s_sno_fk FOREIGN KEY (cno) REFERENCES course(cno)
);

#3、向Student表增加“入学时间(Scome)”列，其数据类型为日期型。
ALTER TABLE student ADD COLUMN scome DATE;

#4、查询选修了3号课程的学生的学号及其成绩，查询结果按分数的降序排列。
SELECT sno,grade FROM sg WHERE cno=3 ORDER BY grade DESC;

#5、查询学习1号课程的学生最高分数、平均成绩。
SELECT MAX(grade),AVG(grade) FROM sg WHERE cno=1;

#6、查询与“李洋”在同一个系学习的学生。
SELECT * FROM student WHERE sdept=(SELECT sdept FROM student WHERE sname='李洋');

#7、将计算机系全体学生的成绩置零。
UPDATE sg SET grade=0 WHERE sno IN (SELECT sno FROM student WHERE sdept='计算机系')

#8、删除学生表中学号为05019的学生记录。
DELETE FROM student WHERE sno=05019;

#9、删除计算机系所有学生的成绩记录。
DELETE FROM sg WHERE sno IN (SELECT sno FROM student WHERE sdept='计算机系');
```

## 练习7

1、建立数据库：test_library

2、建立如下三个表： 表一：press 出版社 属性：编号pressid(int)、名称pressname(varchar)、地址address(varchar)

表二：sort 种类 属性：编号sortno(int)、数量scount(int)

表二：book图书 属性：编号bid(int)、名称 bname(varchar)、种类bsortno(int)、出版社编号pressid(int)

3、给sort表中添加一列属性：描述describes(varchar)

4、向三个表中各插入几条数据

```
mysql> select * from press;
+---------+------------+---------+
| pressid | pressname  | address |
+---------+------------+---------+
|     100 | 外研社     | 上海    |
|     101 | 北大出版社 | 北京    |
|     102 | 教育出版社 | 北京    |
+---------+------------+---------+
```

```
mysql> select * from sort;
+--------+--------+-----------+
| sortno | scount | describes |
+--------+--------+-----------+
|     11 |     50 | 小说      |
|     12 |    300 | 科幻      |
|     13 |    100 | 神话      |
+--------+--------+-----------+
```

```
mysql> select * from book;
+------+----------+---------+---------+
| bid  | bname    | bsortno | pressid |
+------+----------+---------+---------+
|    1 | 红与黑   |      11 |     100 |
|    2 | 幻城     |      12 |     102 |
|    3 | 希腊神话 |      13 |     102 |
+------+----------+---------+---------+
```

5、查询出版社id为100的书的全部信息

6、查询出版社为外研社的书的全部信息

7、查询图书数量（scount）大于100的种类

8、查询图书种类最多的出版社信息

```sql
-- 1、建立数据库：test_library
CREATE DATABASE test_library;

-- 使用数据库
USE test_library;

-- 2、创建出版社表
CREATE TABLE press(
    pressid INT(10) PRIMARY KEY,
    pressname VARCHAR(30),
    address VARCHAR(50)
);
```

```sql
-- 创建一个种类表
CREATE TABLE sort(
    sortno INT(10) PRIMARY KEY,
    scount INT(10)
);

-- 创建图书表
CREATE TABLE book(
    bid INT(10) PRIMARY KEY,
    bname VARCHAR(40),
    bsortno INT(10),
    pressid INT(10),
    CONSTRAINT p_b_pid_fk FOREIGN KEY (pressid) REFERENCES press(pressid),
    CONSTRAINT s_b_sno_fk FOREIGN KEY (bsortno) REFERENCES sort(sortno)
);

-- 3、添加一列属性
ALTER TABLE sort ADD COLUMN describes VARCHAR(30);

-- 4、添加数据
INSERT INTO press VALUES(100,'外研社','上海');
INSERT INTO press VALUES(101,'北大出版社','北京');
INSERT INTO press VALUES(102,'教育出版社','北京');

-- 添加数据
INSERT INTO sort(sortno,scount,describes)VALUES(11,50,'小说');
INSERT INTO sort(sortno,scount,describes)VALUES(12,300,'科幻');
INSERT INTO sort(sortno,scount,describes)VALUES(13,100,'神话');

-- 添加数据
INSERT INTO book VALUES(1,'红与黑',11,100);
INSERT INTO book VALUES(2,'幻城',12,102);
INSERT INTO book VALUES(3,'希腊神话',13,102);
INSERT INTO book VALUES(4,'一千零一夜',13,102);

#5、查询出版社id为100的书的全部信息
SELECT * FROM book WHERE pressid=100;

#6、查询出版社为外研社的书的全部信息
SELECT * FROM book WHERE pressid=(SELECT pressid FROM press WHERE pressname='外研社');

#7、查询图书数量（scount）大于100的种类
SELECT * FROM sort WHERE scount>100;

#8、查询图书种类最多的出版社信息
SELECT * FROM press WHERE pressid=(
    SELECT temp.pressid FROM
 (SELECT pressid,MAX(t.c) FROM (SELECT pressid,COUNT(*) AS c FROM book GROUP BY
pressid ORDER BY c DESC ) AS t) AS temp);


SELECT * FROM press WHERE pressid=(
SELECT pressid
FROM (SELECT pressid,bsortno FROM book GROUP BY pressid,bsortno) temp
GROUP BY pressid
ORDER BY COUNT(*) DESC
LIMIT 0,1)
```

## 练习8

1、建立数据库：test_tour

2、建立如下两个表：

agency旅行社表：

| 列名（英文名） | 列名（中文名） | 数据类型 | 允许空值 | 说明 |
| --- | --- | --- | --- | --- |
| Id | 旅行社编号 | int | no | 主键 |
| Name | 旅行社名 | varchar | no | |
| Address | 旅行社地址 | varchar | no | |
| Areaid | 所属区域Id | Int | yes | |

travel旅行线路表：

| 列名（英文名） | 列名（中文名） | 数据类型 | 允许空值 | 说明 |
| --- | --- | --- | --- | --- |
| Tid | 旅行线路编号 | int | no | 主键 |
| Time | 所需时间 | varchar | no | |
| Position | 目的地 | varchar | no | |
| Money | 花费 | Float | yes | |
| Aid | 所属旅行社id | Int | no | 外键 |
| Count | 报名人数 | Int | yes | |

3、添加记录

agency表数据

| id | name | address |
| --- | --- | --- |
| 101 | 青年旅行社 | 北京海淀 |
| 102 | 天天旅行社 | 天津海院 |

travel表数据

| tid | time | position | money | aid | rcount |
| --- | --- | --- | --- | --- | --- |
| 1 | 5天 | 八达岭 | 3000 | 101 | 10 |
| 2 | 7天 | 水长城 | 5000 | 101 | 14 |
| 3 | 8天 | 水长城 | 6000 | 102 | 11 |

4、查出旅行线路最多的旅社

5、查出最热门的旅行线路(也就是查询出报名人数最多的线路)

6、查询花费少于5000的旅行线路

7、找到一次旅行花费最昂贵的旅行社名

8、查出青年旅社所有的旅行线路都玩一遍需要多少时间。

```
#1、建立数据库：test_tour
CREATE DATABASE test_tour;

#使用数据库
USE test_tour;

#2、
CREATE TABLE agency(
    id INT PRIMARY KEY NOT NULL,
    NAME VARCHAR(20) NOT NULL,
    address VARCHAR(100) NOT NULL,
    areaid INT
);

CREATE TABLE trval(
    tid INT PRIMARY KEY NOT NULL,
    TIME VARCHAR(50) NOT NULL,
    POSITION VARCHAR(100) NOT NULL,
    money FLOAT,
    aid INT NOT NULL,
    rcount INT,
    CONSTRAINT bk_aid FOREIGN KEY trval(aid) REFERENCES agency(id)
);

#3、
INSERT INTO agency(id,NAME,address) VALUES (101,'青年旅行社','北京海淀');
INSERT INTO agency(id,NAME,address) VALUES (102,'天天旅行社','天津海院');

INSERT INTO trval(tid,TIME,POSITION,money,aid,rcount) VALUES (1,'5天','八达
岭',3000,101,10);
INSERT INTO trval(tid,TIME,POSITION,money,aid,rcount) VALUES (2,'7天','水长
城',5000,101,14);
INSERT INTO trval(tid,TIME,POSITION,money,aid,rcount) VALUES (3,'8天','水长
城',6000,102,11);

SELECT * FROM agency;
SELECT * FROM trval;

#4、查出旅行线路最多的旅社
SELECT *
FROM agency INNER JOIN
(SELECT t.aid,MAX(t.c) FROM (SELECT aid,COUNT(*) AS c FROM trval GROUP BY aid)  AS
t)temp
ON agency.id = temp.aid

#5、查出最热门的旅行线路(也就是查询出报名人数最多的线路)
SELECT * FROM trval WHERE rcount=(SELECT MAX(rcount) FROM trval);

#6、查询花费少于5000的旅行线路
SELECT * FROM trval WHERE money<5000;
```

```sql
#7、找到一次旅行花费最昂贵的旅行社名
SELECT NAME FROM agency WHERE id =
(SELECT aid FROM trval WHERE money =(SELECT MAX(money) FROM trval ));

#8、查出青年旅社所有的旅行线路都玩一遍需要多少时间。
SELECT SUM(TIME) FROM trval WHERE aid=(SELECT id FROM agency WHERE NAME='青年旅行社');
```

# 第14章_视图

讲师：尚硅谷-宋红康（江湖人称：康师傅）

官网： http://www.atguigu.com

---

## 练习1

题目：

```
#1. 使用表employees创建视图employee_vu，其中包括姓名（LAST_NAME），员工号（EMPLOYEE_ID），部门号(DEPARTMENT_ID)

#2. 显示视图的结构

#3. 查询视图中的全部内容

#4. 将视图中的数据限定在部门号是80的范围内
```

答案：

```
#1. 使用表employees创建视图employee_vu，其中包括姓名（LAST_NAME），员工号（EMPLOYEE_ID），部门号(DEPARTMENT_ID)
CREATE OR REPLACE VIEW employee_vu
AS
SELECT last_name,employee_id,department_id
FROM employees

#2. 显示视图的结构
DESC employee_vu;

#3. 查询视图中的全部内容
SELECT * FROM employee_vu;

#4. 将视图中的数据限定在部门号是80的范围内
CREATE OR REPLACE VIEW employee_vu
AS
SELECT last_name,employee_id,department_id
FROM employees
WHERE department_id = 80;
```

## 练习2

题目：

```
CREATE TABLE emps
AS
SELECT * FROM atguigudb.employees;

#1. 创建视图emp_v1,要求查询电话号码以‘011’开头的员工姓名和工资、邮箱
```

```
#2. 要求将视图 emp_v1 修改为查询电话号码以‘011’开头的并且邮箱中包含 e 字符的员工姓名和邮箱、电话号码

#3. 向 emp_v1 插入一条记录，是否可以？

#4. 修改emp_v1中员工的工资，每人涨薪1000

#5. 删除emp_v1中姓名为Olsen的员工

#6. 创建视图emp_v2，要求查询部门的最高工资高于 12000 的部门id和其最高工资

#7. 向 emp_v2 中插入一条记录，是否可以？

#8. 删除刚才的emp_v2 和 emp_v1
```

答案：

```
CREATE TABLE emps
AS
SELECT * FROM atguigudb.employees;

#1. 创建视图emp_v1,要求查询电话号码以‘011’开头的员工姓名和工资、邮箱
CREATE OR REPLACE VIEW emp_v1
AS
SELECT last_name,salary,email
FROM emps
WHERE phone_number LIKE '011%';

SELECT *
FROM emp_v1;

#2. 要求将视图 emp_v1 修改为查询电话号码以‘011’开头的并且邮箱中包含 e 字符的员工姓名和邮箱、电话号码
CREATE OR REPLACE VIEW emp_v1
AS
SELECT last_name,salary,email,phone_number
FROM emps
WHERE phone_number LIKE '011%'
AND email LIKE '%e%';

#3. 向 emp_v1 插入一条记录，是否可以？
DESC emps;
DESC emp_v1;

INSERT INTO emp_v1(last_name,salary,email,phone_number)
VALUES('Tom',2300,'tom@126.com','1322321312');

#实测不可以

#4. 修改emp_v1中员工的工资，每人涨薪1000
UPDATE emp_v1
SET salary = salary + 1000;

#5. 删除emp_v1中姓名为Olsen的员工
DELETE FROM emp_v1
WHERE last_name = 'Olsen';

#6. 创建视图emp_v2，要求查询部门的最高工资高于 12000 的部门id和其最高工资
```

```sql
CREATE OR REPLACE VIEW emp_v2
AS
SELECT department_id,MAX(salary) max_sal
FROM emps
GROUP BY department_id
HAVING MAX(salary)>12000;


SELECT *
FROM emp_v2;

#7. 向 emp_v2 中插入一条记录，是否可以？
INSERT INTO emp_v2
VALUES(400,18000);

#8. 删除刚才的emp_v2 和 emp_v1

DROP VIEW IF EXISTS emp_v1,emp_v2;
```

# 第15章_存储过程与函数

讲师：尚硅谷-宋红康（江湖人称：康师傅）

官网： http://www.atguigu.com

## 存储过程练习

```
#0.准备工作
CREATE DATABASE test15_pro_func;

USE test15_pro_func;

#1. 创建存储过程insert_user(),实现传入用户名和密码，插入到admin表中
CREATE TABLE admin(
id INT PRIMARY KEY AUTO_INCREMENT,
user_name VARCHAR(15) NOT NULL,
pwd VARCHAR(25) NOT NULL

);


#2. 创建存储过程get_phone(),实现传入女神编号，返回女神姓名和女神电话

CREATE TABLE beauty(
id INT PRIMARY KEY AUTO_INCREMENT,
NAME VARCHAR(15) NOT NULL,
phone VARCHAR(15) UNIQUE,
birth DATE
);

INSERT INTO beauty(NAME,phone,birth)
VALUES
('朱茵','13201233453','1982-02-12'),
('孙燕姿','13501233653','1980-12-09'),
('田馥甄','13651238755','1983-08-21'),
('邓紫棋','17843283452','1991-11-12'),
('刘若英','18635575464','1989-05-18'),
('杨超越','13761238755','1994-05-11');

SELECT * FROM beauty;

#3. 创建存储过程date_diff()，实现传入两个女神生日，返回日期间隔大小


#4. 创建存储过程format_date(),实现传入一个日期，格式化成xx年xx月xx日并返回


#5. 创建存储过程beauty_limit()，根据传入的起始索引和条目数，查询女神表的记录


#创建带inout模式参数的存储过程
```

```
#6. 传入a和b两个值，最终a和b都翻倍并返回


#7. 删除题目5的存储过程


#8. 查看题目6中存储过程的信息
```

答案:

```
#0.准备工作
CREATE DATABASE test15_pro_func;

#1. 创建存储过程insert_user(),实现传入用户名和密码，插入到admin表中
CREATE TABLE admin(
id INT PRIMARY KEY AUTO_INCREMENT,
user_name VARCHAR(15) NOT NULL,
pwd VARCHAR(25) NOT NULL

);


DELIMITER //

CREATE PROCEDURE insert_user(IN username VARCHAR(20),IN loginPwd VARCHAR(20))
BEGIN
    INSERT INTO admin(user_name,pwd)
    VALUES(username,loginpwd);
END //

DELIMITER ;


#2. 创建存储过程get_phone(),实现传入女神编号，返回女神姓名和女神电话

DELIMITER //

CREATE PROCEDURE get_phone(IN id INT,OUT NAME VARCHAR(20),OUT phone VARCHAR(20))

BEGIN
    SELECT b.name ,b.phone INTO NAME,phone
    FROM beauty b
    WHERE b.id = id;

END //

DELIMITER ;

#调用
CALL get_phone(1,@name,@phone);
SELECT @name,@phone;

#3. 创建存储过程date_diff()，实现传入两个女神生日，返回日期间隔大小
DELIMITER //

CREATE PROCEDURE date_diff(IN birth1 DATETIME,IN birth2 DATETIME,OUT result INT)
BEGIN
```

```
    SELECT DATEDIFF(birth1,birth2) INTO result;
END //
DELIMITER ;

#调用
SET @birth1 = '1992-09-08';
SET @birth2 = '1989-01-03';
CALL date_diff(@birth1,@birth2,@result);

SELECT @result;

#4. 创建存储过程format_date(),实现传入一个日期，格式化成xx年xx月xx日并返回
DELIMITER //

CREATE PROCEDURE format_date(IN mydate DATETIME,OUT strdate VARCHAR(50))
BEGIN
    SELECT DATE_FORMAT(mydate,'%y年%m月%d日') INTO strDate;
END //

DELIMITER ;

#调用
SET @mydate = '1992-09-08';
CALL format_date(@mydate,@strdate);

SELECT @strdate;


#5. 创建存储过程beauty_limit()，根据传入的起始索引和条目数，查询女神表的记录

DELIMITER //

CREATE PROCEDURE beauty_limit(IN startIndex INT,IN size INT)
BEGIN
    SELECT * FROM beauty LIMIT startIndex,size;
END //

DELIMITER ;

#调用
CALL beauty_limit(1,3);

#创建带inout模式参数的存储过程
#6. 传入a和b两个值，最终a和b都翻倍并返回
DELIMITER //

CREATE PROCEDURE add_double(INOUT a INT ,INOUT b INT)
BEGIN
    SET a = a * 2;
    SET b = b * 2;
END //

DELIMITER ;

#调用
SET @a = 3,@b = 5;
CALL add_double(@a,@b);
```

```
SELECT @a,@b;


#7. 删除题目5的存储过程
DROP PROCEDURE beauty_limit;

#8. 查看题目6中存储过程的信息

SHOW CREATE PROCEDURE add_double;
SHOW PROCEDURE STATUS LIKE 'add_double';
```

# 存储函数练习

题目:

```
#0. 准备工作
USE test15_pro_func;

CREATE TABLE employees
AS
SELECT * FROM atguigudb.`employees`;

CREATE TABLE departments
AS
SELECT * FROM atguigudb.`departments`;

#无参有返回
#1. 创建函数get_count(),返回公司的员工个数


#有参有返回
#2. 创建函数ename_salary(),根据员工姓名，返回它的工资


#3. 创建函数dept_sal() ,根据部门名，返回该部门的平均工资


#4. 创建函数add_float()，实现传入两个float，返回二者之和
```

答案:

```
#0. 准备工作
USE testtesttest;

CREATE TABLE employees
AS
SELECT * FROM atguigudb.`employees`;

CREATE TABLE departments
AS
SELECT * FROM atguigudb.`departments`;

#无参有返回
```

```mysql
#1. 创建函数get_count(),返回公司的员工个数
DELIMITER //

CREATE FUNCTION get_count() RETURNS INT
BEGIN
    RETURN (SELECT COUNT(*) FROM employees);

END //
DELIMITER ;

#调用
SELECT get_count();


#有参有返回
#2. 创建函数ename_salary(),根据员工姓名，返回它的工资
DELIMITER //

CREATE FUNCTION ename_salary(emp_name VARCHAR(20)) RETURNS DOUBLE
BEGIN
    RETURN (
        SELECT salary
        FROM employees
        WHERE last_name = emp_name
        );

END //

DELIMITER ;

#调用
SELECT ename_salary('Abel');

#3. 创建函数dept_sal() ,根据部门名，返回该部门的平均工资
DELIMITER //

CREATE FUNCTION dept_sal(dept_name VARCHAR(20)) RETURNS DOUBLE
BEGIN
    RETURN (
        SELECT AVG(salary)
        FROM employees e JOIN departments d
        ON e.department_id = d.department_id
        WHERE d.department_name = dept_name
        );
END //

DELIMITER ;

#调用
SELECT dept_sal('Marketing');


#4. 创建函数add_float()，实现传入两个float，返回二者之和
DELIMITER //

CREATE FUNCTION add_float(num1 FLOAT,num2 FLOAT) RETURNS FLOAT
BEGIN
    RETURN (SELECT num1 + num2 );
```

```
END //

DELIMITER ;

#调用
SET @num1 := 1.2,@num2 = 3.2;
SELECT add_float(@num1,@num2);
```

# 第16章_变量、流程控制与游标

讲师：尚硅谷-宋红康（江湖人称：康师傅）

官网： http://www.atguigu.com

## 1. 变量

题目：

```
#0.准备工作
CREATE DATABASE test16_var_cur;

use test16_var_cur;

CREATE TABLE employees
AS
SELECT * FROM atguigudb.`employees`;

CREATE TABLE departments
AS
SELECT * FROM atguigudb.`departments`;

#无参有返回
#1. 创建函数get_count(),返回公司的员工个数


#有参有返回
#2. 创建函数ename_salary(),根据员工姓名，返回它的工资


#3. 创建函数dept_sal() ,根据部门名，返回该部门的平均工资


#4. 创建函数add_float()，实现传入两个float，返回二者之和
```

答案：

```
#0.准备工作
CREATE DATABASE test16_var_cur;

use test16_var_cur;

CREATE TABLE employees
AS
SELECT * FROM atguigudb.`employees`;

CREATE TABLE departments
AS
SELECT * FROM atguigudb.`departments`;
```

```sql
#无参有返回
#1. 创建函数get_count(),返回公司的员工个数
DELIMITER //

CREATE FUNCTION get_count() RETURNS INT
BEGIN

    DECLARE c INT DEFAULT 0;#定义局部变量
    SELECT COUNT(*) INTO c#赋值
    FROM employees;
    RETURN c;

END //
DELIMITER ;

#调用
SELECT get_count();


#有参有返回
#2. 创建函数ename_salary(),根据员工姓名，返回它的工资
DELIMITER //

CREATE FUNCTION ename_salary(emp_name VARCHAR(15))
RETURNS DOUBLE
BEGIN
    SET @sal=0;#定义用户变量
    SELECT salary INTO @sal   #赋值
    FROM employees
    WHERE last_name = emp_name;

    RETURN @sal;
END //

DELIMITER ;

#调用
SELECT ename_salary('Abel');

#3. 创建函数dept_sal() ,根据部门名，返回该部门的平均工资
DELIMITER //

CREATE FUNCTION dept_sal(dept_name VARCHAR(15))
RETURNS DOUBLE
BEGIN
    DECLARE avg_sal DOUBLE ;
    SELECT AVG(salary) INTO avg_sal
    FROM employees e
    JOIN departments d ON e.department_id = d.department_id
    WHERE d.department_name=dept_name;

    RETURN avg_sal;
END //

DELIMITER ;

#调用
SELECT dept_sal('Marketing');
```

```
#4. 创建函数add_float()，实现传入两个float，返回二者之和
DELIMITER //

CREATE FUNCTION add_float(value1 FLOAT,value2 FLOAT)
RETURNS FLOAT
BEGIN
    DECLARE SUM FLOAT;
    SET SUM=value1+value2;
    RETURN SUM;
END //

DELIMITER ;

#调用
SET @v1 := 12.2;
SET @v2 = 2.3;
SELECT add_float(@v1,@v2);
```

## 2. 流程控制

题目：

```
#1. 创建函数test_if_case()，实现传入成绩，如果成绩>90,返回A，如果成绩>80,返回B，如果成绩>60,返回C，否则返回D
#要求：分别使用if结构和case结构实现

#2. 创建存储过程test_if_pro()，传入工资值，如果工资值<3000,则删除工资为此值的员工，如果3000 <= 工资值 <= 5000,则修改此工资值的员工薪资涨1000，否则涨工资500

#3. 创建存储过程insert_data(),传入参数为 IN 的 INT 类型变量 insert_count,实现向admin表中批量插入insert_count条记录

CREATE TABLE admin(
id INT PRIMARY KEY AUTO_INCREMENT,
user_name VARCHAR(25) NOT NULL,
user_pwd VARCHAR(35) NOT NULL
);

SELECT * FROM admin;
```

答案：

```
#1. 创建函数test_if_case()，实现传入成绩，如果成绩>90,返回A，如果成绩>80,返回B，如果成绩>60,返回C，否则返回D
#要求：分别使用if结构和case结构实现
#方式1：
DELIMITER //

CREATE FUNCTION test_if_case1(score DOUBLE)
RETURNS CHAR
BEGIN
    DECLARE ch CHAR;

    IF score>90
```

```
        THEN SET ch='A';
    ELSEIF score>80
        THEN SET ch='B';
    ELSEIF score>60
        THEN SET ch='C';
    ELSE SET ch='D';
    END IF;

    RETURN ch;

END //

DELIMITER ;



#调用
SELECT test_if_case1(87);

#方式2：
DELIMITER //

CREATE FUNCTION test_if_case2(score DOUBLE)
RETURNS CHAR
BEGIN
    DECLARE ch CHAR;

    CASE
    WHEN score>90 THEN SET ch='A';
    WHEN score>80 THEN SET ch='B';
    WHEN score>60 THEN SET ch='C';
    ELSE SET ch='D';
    END CASE;

    RETURN ch;
END //

DELIMITER ;

#调用
SELECT test_if_case2(67);

#2. 创建存储过程test_if_pro()，传入工资值，如果工资值<3000,则删除工资为此值的员工，如果3000 <= 工资值 <= 5000,则修改此工资值的员工薪资涨1000，否则涨工资500
DELIMITER //

CREATE PROCEDURE test_if_pro(IN sal DOUBLE)
BEGIN
    IF sal<3000
        THEN DELETE FROM employees WHERE salary = sal;
    ELSEIF sal <= 5000
        THEN UPDATE employees SET salary = salary+1000 WHERE salary = sal;
    ELSE
        UPDATE employees SET salary = salary+500 WHERE salary = sal;
    END IF;

END //
```

```
DELIMITER ;

SELECT * FROM employees;

#调用
CALL test_if_pro(3100);



#3. 创建存储过程insert_data(),传入参数为 IN 的 INT 类型变量 insert_count,实现向admin表中批量插入insert_count条记录

DELIMITER //

CREATE PROCEDURE insert_data(IN insert_count INT)
BEGIN
    DECLARE i INT DEFAULT 1;
    WHILE i <= insert_count DO
        INSERT INTO admin(user_name,user_pwd) VALUES(CONCAT('Rose-',i),ROUND(RAND() * 100000));
        SET i=i+1;
    END WHILE;

END //

DELIMITER ;

#调用
CALL insert_data(100);
```

## 3. 游标的使用

创建存储过程update_salary()，参数1为 IN 的INT型变量dept_id，表示部门id；参数2为 IN的INT型变量change_sal_count，表示要调整薪资的员工个数。查询指定id部门的员工信息，按照salary升序排列，根据hire_date的情况，调整前change_sal_count个员工的薪资，详情如下。

| hire_date | salary |
| --- | --- |
| hire_date < 1995 | salary = salary*1.2 |
| hire_date >=1995 and hire_date <= 1998 | salary = salary*1.15 |
| hire_date > 1998 and hire_date <= 2001 | salary = salary *1.10 |
| hire_date > 2001 | salary = salary * 1.05 |

答案：

```
DELIMITER //

CREATE PROCEDURE update_salary(IN dept_id INT,IN change_sal_count INT)
BEGIN
```

```
    #声明变量
    DECLARE int_count INT DEFAULT 0;
    DECLARE salary_rate DOUBLE DEFAULT 0.0;

    DECLARE emp_id INT;
    DECLARE emp_hire_date DATE;


    #声明游标
    DECLARE emp_cursor CURSOR FOR SELECT employee_id,hire_date FROM employees
    WHERE department_id = dept_id ORDER BY salary ;
    #打开游标
    OPEN emp_cursor;

    WHILE int_count < change_sal_count DO
        #使用游标
        FETCH emp_cursor INTO emp_id,emp_hire_date;

        IF(YEAR(emp_hire_date) < 1995)
            THEN SET salary_rate = 1.2;
        ELSEIF(YEAR(emp_hire_date) <= 1998)
            THEN SET salary_rate = 1.15;
        ELSEIF(YEAR(emp_hire_date) <= 2001)
            THEN SET salary_rate = 1.10;
        ELSE SET salary_rate = 1.05;
        END IF;

        #更新工资
        UPDATE employees SET salary = salary * salary_rate
        WHERE employee_id = emp_id;

        #迭代条件
        SET int_count = int_count + 1;

    END WHILE;

    #关闭游标
    CLOSE emp_cursor;

END //

DELIMITER ;

# 调用
CALL update_salary(50,2);
```

# 第17章_触发器

讲师：尚硅谷-宋红康（江湖人称：康师傅）

官网： http://www.atguigu.com

---

## 练习1

- 题目

```
#0. 准备工作
CREATE TABLE emps
AS
SELECT employee_id,last_name,salary
FROM atguigudb.`employees`;

#1. 复制一张emps表的空表emps_back，只有表结构，不包含任何数据


#2. 查询emps_back表中的数据


#3. 创建触发器emps_insert_trigger，每当向emps表中添加一条记录时，同步将这条记录添加到emps_back表中


#4. 验证触发器是否起作用
```

- 答案

```
#0. 准备工作
CREATE TABLE emps
AS
SELECT employee_id,last_name,salary
FROM atguigudb.`employees`;

#1. 复制一张emps表的空表emps_back，只有表结构，不包含任何数据
CREATE TABLE emps_back
AS
SELECT * FROM emps
WHERE 1=2;

#2. 查询emps_back表中的数据
SELECT * FROM emps_back;

#3. 创建触发器emps_insert_trigger，每当向emps表中添加一条记录时，同步将这条记录添加到emps_back表中
DELIMITER //

CREATE TRIGGER emps_insert_trigger
AFTER INSERT ON emps
FOR EACH ROW
```

```
BEGIN
    INSERT INTO emps_back(employee_id,last_name,salary)
    VALUES(NEW.employee_id,NEW.last_name,NEW.salary);

END //

DELIMITER ;

#4. 验证触发器是否起作用
INSERT INTO emps
VALUES(300,'Tom',5600);

SELECT * FROM emps_back;
```

## 练习2

- 题目

```
#0. 准备工作：使用练习1中的emps表

#1. 复制一张emps表的空表emps_back1，只有表结构，不包含任何数据


#2. 查询emps_back1表中的数据


#3. 创建触发器emps_del_trigger，每当向emps表中删除一条记录时，同步将删除的这条记录添加到
emps_back1表中


#4. 验证触发器是否起作用
```

- 答案

```
#0. 准备工作：使用练习1中的emps表

#1. 复制一张emps表的空表emps_back1，只有表结构，不包含任何数据
CREATE TABLE emps_back1
AS
SELECT * FROM emps
WHERE 1=2;

#2. 查询emps_back1表中的数据
SELECT * FROM emps_back1;

#3. 创建触发器emps_del_trigger，每当向emps表中删除一条记录时，同步将删除的这条记录添加到
emps_back1表中
DELIMITER //

CREATE TRIGGER emps_del_trigger
BEFORE DELETE ON emps
FOR EACH ROW
```

```
BEGIN
    INSERT INTO emps_back1(employee_id,last_name,salary)
    VALUES(OLD.employee_id,OLD.last_name,OLD.salary);

END //

DELIMITER ;

#4. 验证触发器是否起作用
#测试1
DELETE FROM emps
WHERE employee_id = 100;

SELECT * FROM emps_back1;
#测试2
DELETE FROM emps;

SELECT * FROM emps_back1;
```

# 第18章_MySQL8其它新特性

讲师：尚硅谷-宋红康（江湖人称：康师傅）

官网： http://www.atguigu.com

---

题目：

```
#1. 创建students数据表，如下

CREATE TABLE students(
id INT PRIMARY KEY AUTO_INCREMENT,
student VARCHAR(15),
points TINYINT
);

#2. 向表中添加数据如下
INSERT INTO students(student,points)
VALUES
('张三',89),
('李四',77),
('王五',88),
('赵六',90),
('孙七',90),
('周八',88);

#3. 分别使用RANK()、DENSE_RANK() 和 ROW_NUMBER()函数对学生成绩降序排列情况进行显示
```

答案：

```
#1. 创建students数据表，如下

CREATE TABLE students(
id INT PRIMARY KEY AUTO_INCREMENT,
student VARCHAR(15),
points TINYINT
);

#2. 向表中添加数据如下
INSERT INTO students(student,points)
VALUES
('张三',89),
('李四',77),
('王五',88),
('赵六',90),
('孙七',90),
('周八',88);

#3. 分别使用RANK()、DENSE_RANK() 和 ROW_NUMBER()函数对学生成绩降序排列情况进行显示
SELECT student,points,
RANK() OVER w AS 排序1,
DENSE_RANK() OVER w AS 排序2,
ROW_NUMBER() OVER w AS 排序3
FROM students
```

```
WINDOW w AS (ORDER BY points DESC);
```