

## 多表查询：

\* 查询语法：

```
select
    列名列表
from
    表名列表
where....
```

\* 准备sql

# 创建部门表

```
CREATE TABLE dept(
    id INT PRIMARY KEY AUTO_INCREMENT,
    NAME VARCHAR(20)
);
INSERT INTO dept (NAME) VALUES ('开发部'),('市场部'),('财务部');
```

# 创建员工表

```
CREATE TABLE emp (
    id INT PRIMARY KEY AUTO_INCREMENT,
    NAME VARCHAR(10),
    gender CHAR(1), -- 性别
    salary DOUBLE, -- 工资
    join_date DATE, -- 入职日期
    dept_id INT,
    FOREIGN KEY (dept_id) REFERENCES dept(id) -- 外键，关联部门表(部门表的主键)
);
INSERT INTO emp(NAME,gender,salary,join_date,dept_id) VALUES('孙悟空','男',7200,'2013-02-24',1);
INSERT INTO emp(NAME,gender,salary,join_date,dept_id) VALUES('猪八戒','男',3600,'2010-12-02',2);
INSERT INTO emp(NAME,gender,salary,join_date,dept_id) VALUES('唐僧','男',9000,'2008-08-08',2);
INSERT INTO emp(NAME,gender,salary,join_date,dept_id) VALUES('白骨精','女',5000,'2015-10-07',3);
INSERT INTO emp(NAME,gender,salary,join_date,dept_id) VALUES('蜘蛛精','女',4500,'2011-03-14',1);
```

\* 笛卡尔积：

\* 有两个集合A,B .取这两个集合的所有组成情况。

\* 要完成多表查询，需要消除无用的数据

\* 多表查询的分类：

1. 内连接查询：

1. 隐式内连接：使用where条件消除无用数据

\* 例子：

-- 查询所有员工信息和对应的部门信息

```
SELECT * FROM emp,dept WHERE emp.`dept_id` = dept.`id`;
```

-- 查询员工表的名称，性别。部门表的名称

```
SELECT emp.name,emp.gender,dept.name FROM emp,dept WHERE emp.`dept_id` = dept.`id`;
```

```
SELECT
```

```

        t1.name, -- 员工表的姓名
        t1.gender, -- 员工表的性别
        t2.name -- 部门表的名称
FROM
    emp t1,
    dept t2
WHERE
    t1.`dept_id` = t2.`id`;

```

## 2. 显式内连接:

- \* 语法: select 字段列表 from 表名1 [inner] join 表名2 on 条件
- \* 例如:
  - \* SELECT \* FROM emp INNER JOIN dept ON emp.`dept\_id` = dept.`id`;
  - \* SELECT \* FROM emp JOIN dept ON emp.`dept\_id` = dept.`id`;

## 3. 内连接查询:

1. 从哪些表中查询数据
2. 条件是什么
3. 查询哪些字段

## 2. 外链连接查询:

### 1. 左外连接:

- \* 语法: select 字段列表 from 表1 left [outer] join 表2 on 条件;
- \* 查询的是左表所有数据以及其交集部分。
- \* 例子:
  - 查询所有员工信息, 如果员工有部门, 则查询部门名称, 没有部门, 则不显示部门名称
  - SELECT t1.\*, t2.`name` FROM emp t1 LEFT JOIN dept t2 ON t1.`dept\_id` =

t2.`id`;

### 2. 右外连接:

- \* 语法: select 字段列表 from 表1 right [outer] join 表2 on 条件;
- \* 查询的是右表所有数据以及其交集部分。
- \* 例子:
  - SELECT \* FROM dept t2 RIGHT JOIN emp t1 ON t1.`dept\_id` = t2.`id`;

## 3. 子查询:

- \* 概念: 查询中嵌套查询, 称嵌套查询为子查询。

- 查询工资最高的员工信息
- 1 查询最高的工资是多少 9000
- SELECT MAX(salary) FROM emp;

- 2 查询员工信息, 并且工资等于9000的
- SELECT \* FROM emp WHERE emp.`salary` = 9000;

- 一条sql就完成这个操作。子查询
- SELECT \* FROM emp WHERE emp.`salary` = (SELECT MAX(salary) FROM emp);

- \* 子查询不同情况

### 1. 子查询的结果是单行单列的:

- \* 子查询可以作为条件, 使用运算符去判断。 运算符: > >= < <= =
- \*

- 查询员工工资小于平均工资的人

- SELECT \* FROM emp WHERE emp.salary < (SELECT AVG(salary) FROM emp);

### 2. 子查询的结果是多行单列的:

- \* 子查询可以作为条件, 使用运算符in来判断

- 查询'财务部'和'市场部'所有的员工信息

```

SELECT id FROM dept WHERE NAME = '财务部' OR NAME = '市场部';
SELECT * FROM emp WHERE dept_id = 3 OR dept_id = 2;
-- 子查询
SELECT * FROM emp WHERE dept_id IN (SELECT id FROM dept WHERE NAME = '财务部'
OR NAME = '市场部');

```

### 3. 子查询的结果是多行多列的:

```

* 子查询可以作为一张虚拟表参与查询
-- 查询员工入职日期是2011-11-11日之后的员工信息和部门信息
-- 子查询
SELECT * FROM dept t1 ,(SELECT * FROM emp WHERE emp.`join_date` > '2011-11-
11') t2

WHERE t1.id = t2.dept_id;

-- 普通内连接
SELECT * FROM emp t1,dept t2 WHERE t1.`dept_id` = t2.`id` AND t1.`join_date` >
'2011-11-11'

```

### \* 多表查询练习

```

-- 部门表
CREATE TABLE dept (
  id INT PRIMARY KEY PRIMARY KEY, -- 部门id
  dname VARCHAR(50), -- 部门名称
  loc VARCHAR(50) -- 部门所在地
);

```

```

-- 添加4个部门
INSERT INTO dept(id,dname,loc) VALUES
(10,'教研部','北京'),
(20,'学工部','上海'),
(30,'销售部','广州'),
(40,'财务部','深圳');

```

### -- 职务表, 职务名称, 职务描述

```

CREATE TABLE job (
  id INT PRIMARY KEY,
  jname VARCHAR(20),
  description VARCHAR(50)
);

```

### -- 添加4个职务

```

INSERT INTO job (id, jname, description) VALUES
(1, '董事长', '管理整个公司, 接单'),
(2, '经理', '管理部门员工'),
(3, '销售员', '向客人推销产品'),
(4, '文员', '使用办公软件');

```

### -- 员工表

```

CREATE TABLE emp (
  id INT PRIMARY KEY, -- 员工id
  ename VARCHAR(50), -- 员工姓名

```

```

    job_id INT, -- 职务id
    mgr INT , -- 上级领导
    joindate DATE, -- 入职日期
    salary DECIMAL(7,2), -- 工资
    bonus DECIMAL(7,2), -- 奖金
    dept_id INT, -- 所在部门编号
    CONSTRAINT emp_jobid_ref_job_id_fk FOREIGN KEY (job_id) REFERENCES job (id),
    CONSTRAINT emp_deptid_ref_dept_id_fk FOREIGN KEY (dept_id) REFERENCES dept (id)
);

-- 添加员工
INSERT INTO emp(id,ename,job_id,mgr,joindate,salary,bonus,dept_id) VALUES
(1001,'孙悟空',4,1004,'2000-12-17','8000.00',NULL,20),
(1002,'卢俊义',3,1006,'2001-02-20','16000.00','3000.00',30),
(1003,'林冲',3,1006,'2001-02-22','12500.00','5000.00',30),
(1004,'唐僧',2,1009,'2001-04-02','29750.00',NULL,20),
(1005,'李逵',4,1006,'2001-09-28','12500.00','14000.00',30),
(1006,'宋江',2,1009,'2001-05-01','28500.00',NULL,30),
(1007,'刘备',2,1009,'2001-09-01','24500.00',NULL,10),
(1008,'猪八戒',4,1004,'2007-04-19','30000.00',NULL,20),
(1009,'罗贯中',1,NULL,'2001-11-17','50000.00',NULL,10),
(1010,'吴用',3,1006,'2001-09-08','15000.00','0.00',30),
(1011,'沙僧',4,1004,'2007-05-23','11000.00',NULL,20),
(1012,'李逵',4,1006,'2001-12-03','9500.00',NULL,30),
(1013,'小白龙',4,1004,'2001-12-03','30000.00',NULL,20),
(1014,'关羽',4,1007,'2002-01-23','13000.00',NULL,10);

```

```

-- 工资等级表
CREATE TABLE salarygrade (
    grade INT PRIMARY KEY, -- 级别
    losalary INT, -- 最低工资
    hisalary INT -- 最高工资
);

-- 添加5个工资等级
INSERT INTO salarygrade(grade,losalary,hisalary) VALUES
(1,7000,12000),
(2,12010,14000),
(3,14010,20000),
(4,20010,30000),
(5,30010,99990);

-- 需求:

-- 1.查询所有员工信息。查询员工编号, 员工姓名, 工资, 职务名称, 职务描述
/*
    分析:
    1.员工编号, 员工姓名, 工资, 需要查询emp表    职务名称, 职务描述 需要查询job表
    2.查询条件 emp.job_id = job.id

*/
SELECT
    t1.`id`, -- 员工编号

```

```

        t1.`ename`, -- 员工姓名
        t1.`salary`,-- 工资
        t2.`jname`, -- 职务名称
        t2.`description` -- 职务描述
FROM
    emp t1, job t2
WHERE
    t1.`job_id` = t2.`id`;

```

```

-- 2.查询员工编号, 员工姓名, 工资, 职务名称, 职务描述, 部门名称, 部门位置
/*

```

分析:

1. 员工编号, 员工姓名, 工资 emp 职务名称, 职务描述 job 部门名称, 部门位置 dept
2. 条件: emp.job\_id = job.id and emp.dept\_id = dept.id

```

*/

```

```

SELECT
    t1.`id`, -- 员工编号
    t1.`ename`, -- 员工姓名
    t1.`salary`,-- 工资
    t2.`jname`, -- 职务名称
    t2.`description`, -- 职务描述
    t3.`dname`, -- 部门名称
    t3.`loc` -- 部门位置
FROM
    emp t1, job t2,dept t3
WHERE
    t1.`job_id` = t2.`id` AND t1.`dept_id` = t3.`id`;

```

```

-- 3.查询员工姓名, 工资, 工资等级
/*

```

分析:

- 1.员工姓名, 工资 emp 工资等级 salarygrade
- 2.条件 emp.salary >= salarygrade.losalary and emp.salary <=

```

salarygrade.hisalary
        emp.salary BETWEEN salarygrade.losalary and salarygrade.hisalary

```

```

*/

```

```

SELECT
    t1.ename ,
    t1.`salary`,
    t2.*
FROM emp t1, salarygrade t2
WHERE t1.`salary` BETWEEN t2.`losalary` AND t2.`hisalary`;

```

```

-- 4.查询员工姓名, 工资, 职务名称, 职务描述, 部门名称, 部门位置, 工资等级
/*

```

分析:

1. 员工姓名, 工资 emp , 职务名称, 职务描述 job 部门名称, 部门位置, dept 工资等级 salarygrade
2. 条件: emp.job\_id = job.id and emp.dept\_id = dept.id and emp.salary BETWEEN salarygrade.losalary and salarygrade.hisalary

```

*/
SELECT
    t1.`ename`,
    t1.`salary`,
    t2.`jname`,
    t2.`description`,
    t3.`dname`,
    t3.`loc`,
    t4.`grade`
FROM
    emp t1,job t2,dept t3,salarygrade t4
WHERE
    t1.`job_id` = t2.`id`
    AND t1.`dept_id` = t3.`id`
    AND t1.`salary` BETWEEN t4.`losalary` AND t4.`hisalary`;

```

-- 5.查询出部门编号、部门名称、部门位置、部门人数

```

/*
    分析：
    1.部门编号、部门名称、部门位置 dept 表。 部门人数 emp表
    2.使用分组查询。按照emp.dept_id完成分组，查询count(id)
    3.使用子查询将第2步的查询结果和dept表进行关联查询

*/
SELECT
    t1.`id`,t1.`dname`,t1.`loc` , t2.total
FROM
    dept t1,
    (SELECT
        dept_id,COUNT(id) total
    FROM
        emp
    GROUP BY dept_id) t2
WHERE t1.`id` = t2.dept_id;

```

-- 6.查询所有员工的姓名及其直接上级的姓名,没有领导的员工也需要查询

```

/*
    分析：
    1.姓名 emp, 直接上级的姓名 emp
      * emp表的id 和 mgr 是自关联
    2.条件 emp.id = emp.mgr
    3.查询左表的所有数据, 和 交集数据
      * 使用左外连接查询

*/
/*
select
    t1.ename,
    t1.mgr,

```

```

        t2.`id`,
        t2.ename
    from emp t1, emp t2
    where t1.mgr = t2.`id`;

*/

SELECT
    t1.ename,
    t1.mgr,
    t2.`id`,
    t2.`ename`
FROM emp t1
LEFT JOIN emp t2
ON t1.`mgr` = t2.`id`;

```

## 事务

### 1. 事务的基本介绍

#### 1. 概念:

- \* 如果一个包含多个步骤的业务操作，被事务管理，那么这些操作要么同时成功，要么同时失败。

#### 2. 操作:

1. 开启事务: start transaction;
2. 回滚: rollback;
3. 提交: commit;

#### 3. 例子:

```

CREATE TABLE account (
    id INT PRIMARY KEY AUTO_INCREMENT,
    NAME VARCHAR(10),
    balance DOUBLE
);
-- 添加数据
INSERT INTO account (NAME, balance) VALUES ('zhangsan', 1000), ('lisi', 1000);

```

```

SELECT * FROM account;
UPDATE account SET balance = 1000;
-- 张三给李四转账 500 元

-- 0. 开启事务
START TRANSACTION;
-- 1. 张三账户 -500

UPDATE account SET balance = balance - 500 WHERE NAME = 'zhangsan';
-- 2. 李四账户 +500
-- 出错了...
UPDATE account SET balance = balance + 500 WHERE NAME = 'lisi';

-- 发现执行没有问题，提交事务
COMMIT;

```

-- 发现问题了，回滚事务

ROLLBACK;

#### 4. MySQL数据库中事务默认自动提交

\* 事务提交的两种方式:

\* 自动提交:

\* mysql就是自动提交的

\* 一条DML(增删改)语句会自动提交一次事务。

\* 手动提交:

\* Oracle 数据库默认是手动提交事务

\* 需要先开启事务，再提交

\* 修改事务的默认提交方式:

\* 查看事务的默认提交方式: `SELECT @@autocommit;` -- 1 代表自动提交 0 代表手动提交

\* 修改默认提交方式: `set @@autocommit = 0;`

#### 2. 事务的四大特征:

1. 原子性: 是不可分割的最小操作单位, 要么同时成功, 要么同时失败。

2. 持久性: 当事务提交或回滚后, 数据库会持久化的保存数据。

3. 隔离性: 多个事务之间。相互独立。

4. 一致性: 事务操作前后, 数据总量不变

#### 3. 事务的隔离级别 (了解)

\* 概念: 多个事务之间隔离的, 相互独立的。但是如果多个事务操作同一批数据, 则会引发一些问题, 设置不同的隔离级别就可以解决这些问题。

\* 存在问题:

1. 脏读: 一个事务, 读取到另一个事务中没有提交的数据

2. 不可重复读(虚读): 在同一个事务中, 两次读取到的数据不一样。

3. 幻读: 一个事务操作(DML)数据表中所有记录, 另一个事务添加了一条数据, 则第一个事务查询不到自己的修改。

\* 隔离级别:

1. read uncommitted: 读未提交

\* 产生的问题: 脏读、不可重复读、幻读

2. read committed: 读已提交 (Oracle)

\* 产生的问题: 不可重复读、幻读

3. repeatable read: 可重复读 (MySQL默认)

\* 产生的问题: 幻读

4. serializable: 串行化

\* 可以解决所有的问题

\* 注意: 隔离级别从小到大安全性越来越高, 但是效率越来越低

\* 数据库查询隔离级别:

\* `select @@tx_isolation;`

\* 数据库设置隔离级别:

\* `set global transaction isolation level 级别字符串;`

\* 演示:

`set global transaction isolation level read uncommitted;`

`start transaction;`

-- 转账操作

`update account set balance = balance - 500 where id = 1;`

`update account set balance = balance + 500 where id = 2;`



# DCL:

## \* SQL分类:

1. DDL: 操作数据库和表
2. DML: 增删改表中数据
3. DQL: 查询表中数据
4. DCL: 管理用户, 授权

## \* DBA: 数据库管理员

## \* DCL: 管理用户, 授权

### 1. 管理用户

#### 1. 添加用户:

\* 语法: CREATE USER '用户名'@'主机名' IDENTIFIED BY '密码';

#### 2. 删除用户:

\* 语法: DROP USER '用户名'@'主机名';

#### 3. 修改用户密码:

```
UPDATE USER SET PASSWORD = PASSWORD('新密码') WHERE USER = '用户名';
```

```
UPDATE USER SET PASSWORD = PASSWORD('abc') WHERE USER = 'lisi';
```

```
SET PASSWORD FOR '用户名'@'主机名' = PASSWORD('新密码');
```

```
SET PASSWORD FOR 'root'@'localhost' = PASSWORD('123');
```

## \* mysql中忘记了root用户的密码?

### 1. cmd -- > net stop mysql 停止mysql服务

\* 需要管理员运行该cmd

### 2. 使用无验证方式启动mysql服务: mysqld --skip-grant-tables

### 3. 打开新的cmd窗口,直接输入mysql命令, 敲回车。就可以登录成功

### 4. use mysql;

### 5. update user set password = password('你的新密码') where user = 'root';

### 6. 关闭两个窗口

### 7. 打开任务管理器, 手动结束mysqld.exe 的进程

### 8. 启动mysql服务

### 9. 使用新密码登录。

### 4. 查询用户:

-- 1. 切换到mysql数据库

```
USE mysql;
```

-- 2. 查询user表

```
SELECT * FROM USER;
```

\* 通配符: % 表示可以在任意主机使用用户登录数据库

## 2. 权限管理:

### 1. 查询权限:

-- 查询权限

```
SHOW GRANTS FOR '用户名'@'主机名';
```

```
SHOW GRANTS FOR 'lisi'@'%';
```

### 2. 授予权限:

-- 授予权限

```
grant 权限列表 on 数据库名.表名 to '用户名'@'主机名';  
-- 给张三用户授予所有权限，在任意数据库任意表上
```

```
GRANT ALL ON *.* TO 'zhangsan'@'localhost';
```

### 3. 撤销权限:

```
-- 撤销权限:  
revoke 权限列表 on 数据库名.表名 from '用户名'@'主机名';  
REVOKE UPDATE ON db3.`account` FROM 'lisi'@'%';
```