**第04章 多表查询与组函数**

**本章内容**

* 组函数
* group by分组
* 使用having过滤
* 子查询
* 连接查询
* 自关联

# 组函数

也叫聚合函数，分组函数作用于一组数据，并对一组数据返回一个值

--1.MAX()：求最大值,求员工的最高工资

select max(sal) from emp;

--2.MIN()：求最小值,求员工的最低工资

select min(sal) from emp;

--3.SUM()：求和,求员工的总工资

select sum(sal) from emp;

--4.AVG()：求平均值,求员工的平均工资

select avg(sal) from emp;

--5.COUNT()：求总数,求员工的总数

--注意：这个\*号可以用任意数字替换

select count(\*) from emp;

# group by分组

分组查询语法：

select 分组条件【分组条件1，分组条件2.......】

from 表名

where 条件 --可选语句

group by 分组条件【分组条件1，分组条件2......】 --分组必须语句

having 对分组进行过滤 --可选语句

order by 排序条件 --可选语句

注意：如果分组了,select后面的字段要么是分组的条件，要么是聚合函数之一

--需求1:求每个部门的平均薪水

select deptno,avg(sal) as avg\_sal

from emp

group by deptno;

--需求2：每个部门同一个职位的最大工资

select deptno ,job, max(sal)

from emp

group by deptno,job;

# 使用having过滤

使用having对分组以后的数据进行再过滤

--需求1:按照部门编号进行分组，要求显示平均薪水大于2000的部门的部门编号和平均薪水

select deptno,avg(sal)

from emp

group by deptno

having avg(sal)>2000;

--需求2:薪水大于1200的雇员，按照部门编号进行分组，

--分组之后平均薪水必须大于1500,求分组内的平均工资，平均工资按倒序排列

--注意：排序的字段必须是查询字段之中

select deptno,avg(sal)

from emp

where sal>1200

group by deptno

having avg(sal)>1500

order by avg(sal) desc;

# 子查询

子查询(也叫嵌套查询)查多次，多个select嵌套出现,第一次的查询结果可以作为第二次的查询条件 或 表名

--需求1:求最高工资那个人的姓名和薪水

--分解第一步:求最高工资

select max(sal) from emp;

--分解第二步:求最高工资的那个员工叫什么

select ename,sal from emp where sal=5000;

--分解第三步：合并

select ename,sal from emp where sal=(select max(sal) from emp);

--需求2:求工资大于平均工资的员工

--分解第一步：求平均工资

select avg(sal) from emp;

--分解第二步：工资大于平均工资的 人的姓名、工资

select ename,sal from emp where sal>2073.21428571429;

--分解第三步：合并

select ename,sal from emp where sal>(select avg(sal) from emp);

--需求3:分页查询

select \* from (select e.\*, rownum rn from(select ename,job from emp) e

where rownum<=10)where rn>=5;

# 连接查询(多表查询)

--1、内连接

--语法： select \* from 表1 inner join 表2 on (表1.字段=表2.字段)

--需求:查询员工姓名、薪水、部门名称

select e.ename,e.sal,d.dname

from dept d inner join emp e

on (d.deptno = e.deptno);

select e.ename,e.sal,d.dname

from dept d,emp e

where (d.deptno = e.deptno);

/\*外连接分类:左外连接、右外连接、全外连接\*/

--2、左外连接

--语法： select \* from 表1 left outer join 表2 on (表1.字段=表2.字段)

--需求：在内连接的基础上,保证左表的数据都有 (右表的字段用空补全)

--注意：外连接语句中的outer关键字可以省略

select \* from dept left outer join emp

on (dept.deptno = emp.deptno);

--3、右外连接

--语法： select \* from 表1 right outer join 表2 on (表1.字段=表2.字段)

--需求：在内连接的基础上,保证右表的数据都有 (左表的字段用空补全)

select \* from dept right outer join emp

on (dept.deptno = emp.deptno);

--4、全外连接

--语法： select \* from 表1 full outer join 表2 on (表1.字段=表2.字段)

--需求：在内连接的基础上保证左右表的数据都有 (左连接和右连接的并集)

select \* from dept full outer join emp

on (dept.deptno = emp.deptno);

--5、交叉连接(笛卡尔积)和自然连接

--语法： select \* from 表1 cross join 表2

select \* from emp;-- sql 15条

select \* from dept;-- sql 9条

select ename,dname from emp,dept;--？sql 135条

select ename,dname from dept cross join emp;--？sql 135条

--左表15条\*右表9=查询结果共135条

--6、自然连接

--自然连接是在两张表中寻找那些数据类型和列名都相同的字段，然后自动地将他们连接起来，--并返回所有符合条件按的结果

--语法： select \* from 表1 natural join 表2

select \* from dept natural join emp;

--特价于

select emp.ename,dept.dname

from emp join dept on emp.deptno = dept.deptno;

# 自关联

--需求1:SMITH这个人的名字 以及他的经理人的名字

select employee.ename,manager.ename

from emp manager inner join emp employee

on(manager.empno=employee.mgr)

where employee.ename='SMITH';

--需求2:查询所有员工的名字和上司的名字

select employee.ename,manager.ename

from emp manager right join emp employee

on(manager.empno=employee.mgr);

# 面试题

*--技术点*

1、行变列

2、排名

*--创建表*

create table student

(

stuID number(8),

stuName varchar2(20),

cName varchar2(20),

score number(8)

);

*--插入数据*

insert into student

select 1,'赵雷','语文',60 from dual union

select 1,'赵雷','数学',70 from dual union

select 1,'赵雷','英语',80 from dual union

select 2,'钱电','语文',85 from dual union

select 2,'钱电','数学',77 from dual union

select 2,'钱电','英语',69 from dual union

select 3,'孙风','语文',85 from dual union

select 3,'孙风','数学',72 from dual union

select 3,'孙风','英语',98 from dual;

*--列变成行显示*

select

stuID 编号,stuName 姓名,

max(case cName when '语文' then score else null end) 语文,

max(case cName when '数学' then score else null end) 数学,

max(case cName when '英语' then score else null end) 英语

from student

group by stuID,stuName;

*--各科成绩排名*

select student.\*,dense\_rank() over(partition by cName order by score desc) 排名

from student;

*--总分排名*

select stuid,stuName,sum(score),dense\_rank() over(order by sum(score) desc) 排名 from student group by stuid,stuName;