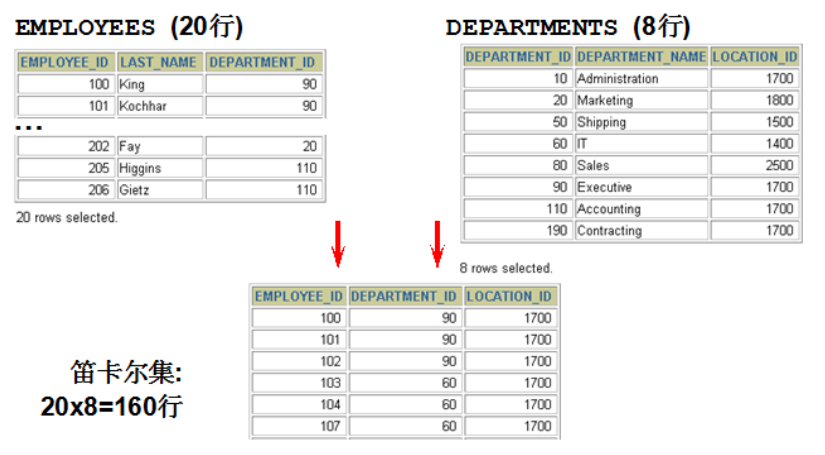
# Oracle第二天

# 多表查询

## 什么是笛卡尔积？



## Oracle的连接条件的类型

* 等值连接
* 不等值连接
* 外连接
* 自连接

## Oracle多表连接示例

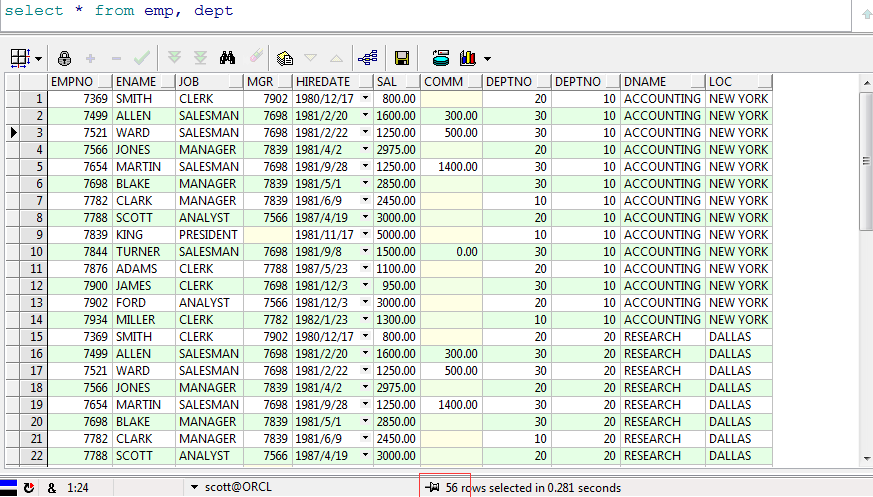
### 1.多表连接基本查询

使用一张以上的表做查询就是多表查询

语法： SELECT {DISTINCT} \*|列名.. FROM 表名 别名，表名1 别名

{WHERE 限制条件 ORDER BY 排序字段 ASC|DESC...}

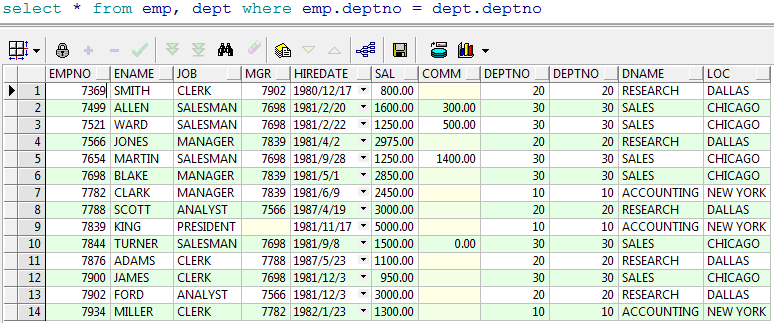
范例：查询员工表和部门表



我们发现产生的记录数是56条，我们还会发现emp表是14条，dept表是4条，56正是emp表和dept表的记录数的乘积，我们称其为笛卡尔积。

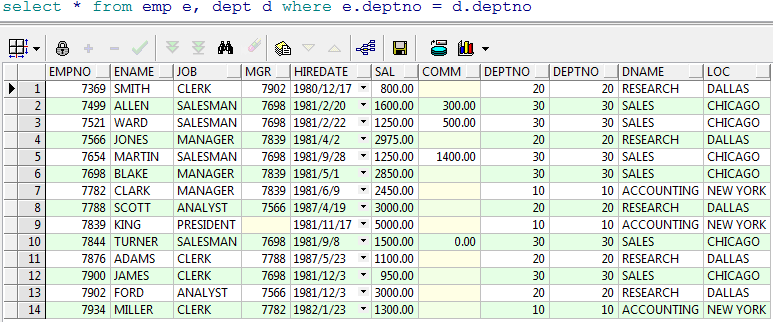
如果多张表进行一起查询而且每张表的数据很大的话笛卡尔积就会变得非常大，对性能造成影响，想要去掉笛卡尔积我们需要关联查询。

在两张表中我们发现有一个共同的字段是depno，depno就是两张表的关联的字段，我们可以使用这个字段来做限制条件，两张表的关联查询字段一般是其中一张表的主键，另一张表的外键。

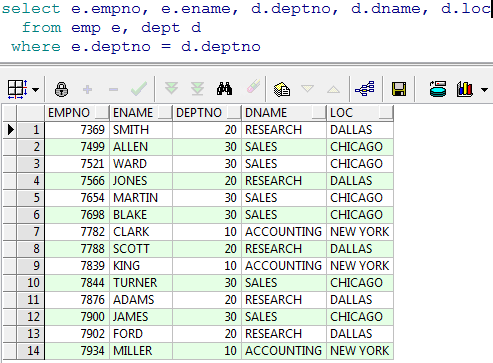


关联之后我们发现数据条数是14条，不在是56条。

多表查询我们可以为每一张表起一个别名

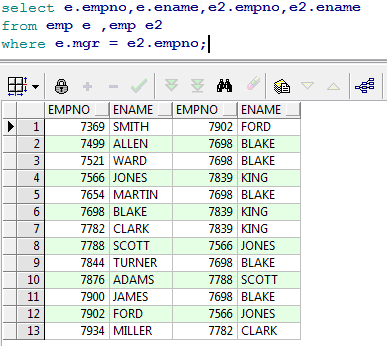


范例：查询出雇员的编号，姓名，部门的编号和名称，地址



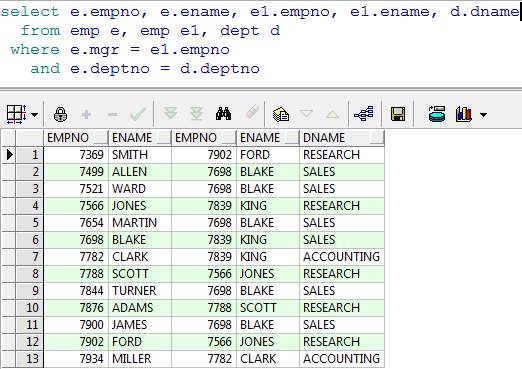
范例：查询出每个员工的上级领导

分析：emp表中的mgr字段是当前雇员的上级领导的编号，所以该字段对emp表产生了自身关联，可以使用mgr字段和empno来关联



范例:在上一个例子的基础上查询该员工的部门名称

分析：只要在上一个例子基础上再加一张表的关联，使用deptno来做关联字段即可



范例：查询出每个员工编号，姓名，部门名称，工资等级和他的上级领导的姓名，工资等级

select e.empno,

e.ename,

decode(s.grade,

1,'一级',

2,'二级',

3,'三级',

4,'四级',

5,'五级') grade,

d.dname,

e1.empno,

e1.ename,

decode(s1.grade,

1,'一级',

2,'二级',

3,'三级',

4,'四级',

5,'五级') grade

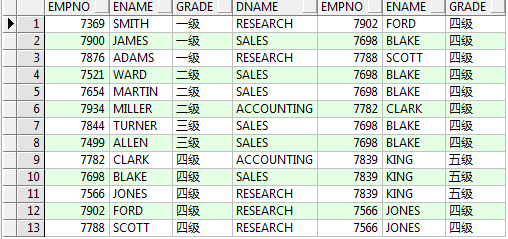
from emp e, emp e1, dept d, salgrade s, salgrade s1

where e.mgr = e1.empno

and e.deptno = d.deptno

and e.sal between s.losal and s.hisal

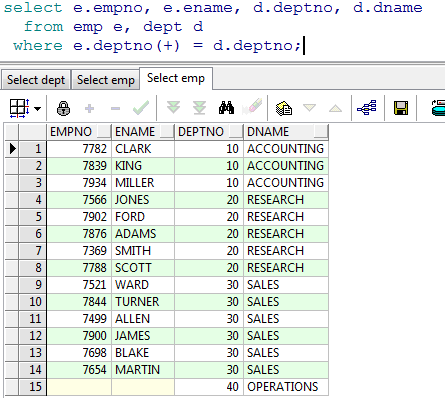
and e1.sal between s1.losal and s1.hisal



### 2.外连接（左右连接）

1. 右连接

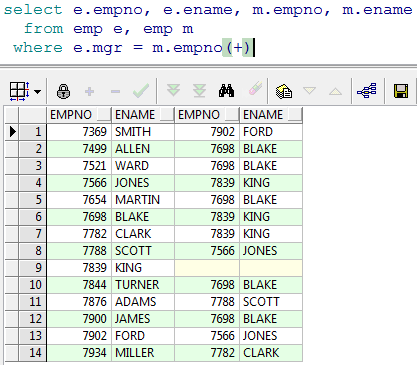
当我们在做基本连接查询的时候，查询出所有的部门下的员工，我们发现编号为40的部门下没有员工，但是要求把该部门也展示出来，我们发现上面的基本查询是办不到的



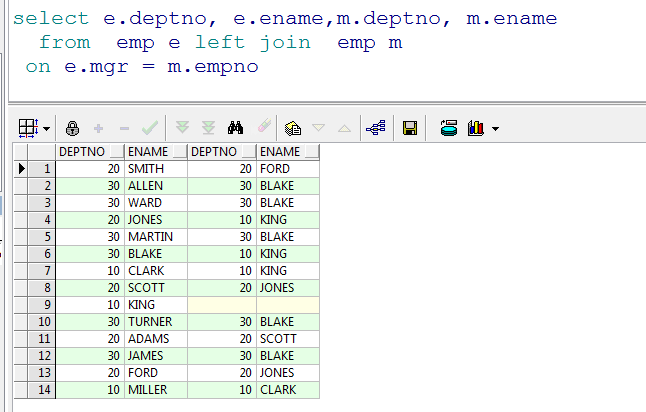
使用(+)表示左连接或者右连接。

范例：查询出所有员工的上级领导

分析：我们发现使用我们以前的做法发现KING的上级领导没有被展示，我们需要使用外连接把他查询出来



如果用left join实现：

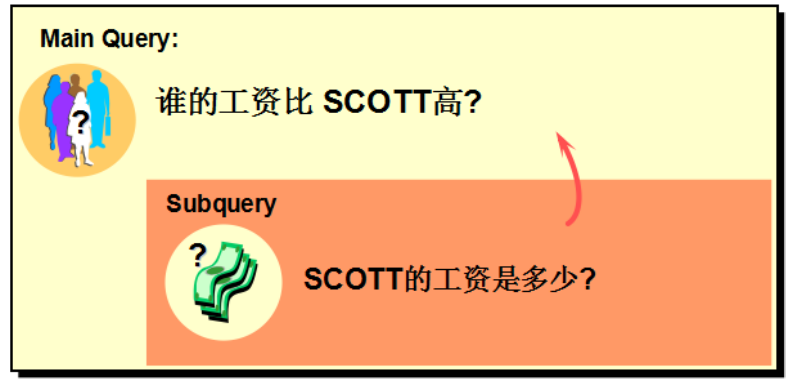


因为（+）这种形式是oracle数据库独有的，所以要求大家一定要掌握left join 或right join方式的写法。

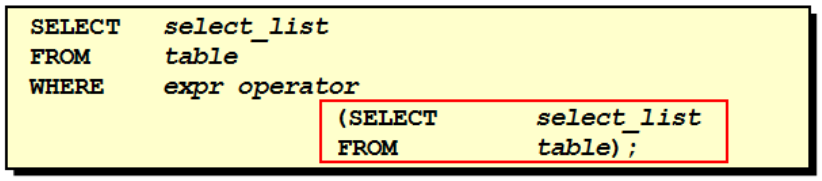
# 子查询

## 什么是子查询？

使用子查询解决问题：谁的工资比SCOTT高？

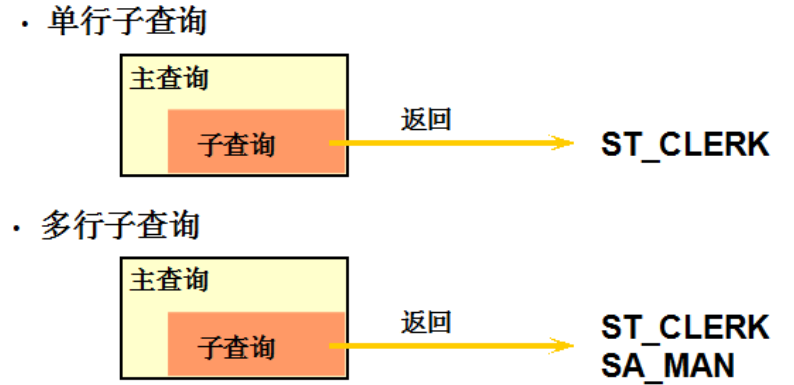


## 子查询的语法



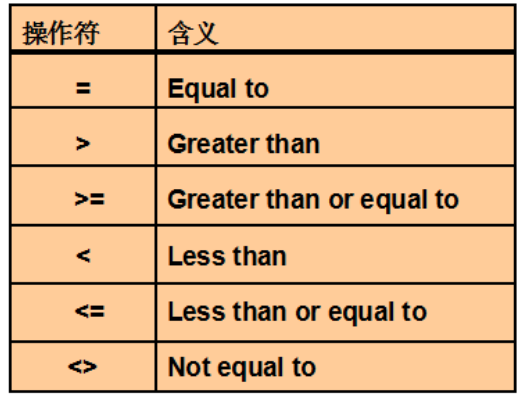
* 子查询 (内查询) 在主查询之前一次执行完成。
* 子查询的结果被主查询使用 (外查询)。

## 子查询的类型

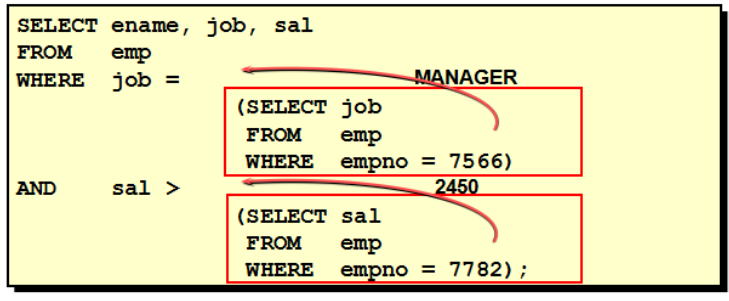


## 单行子查询

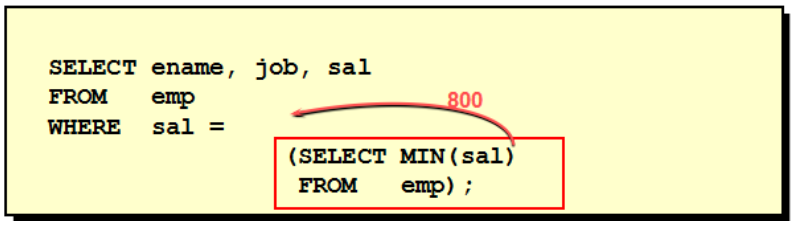
* 只返回一条记录
* 单行操作符



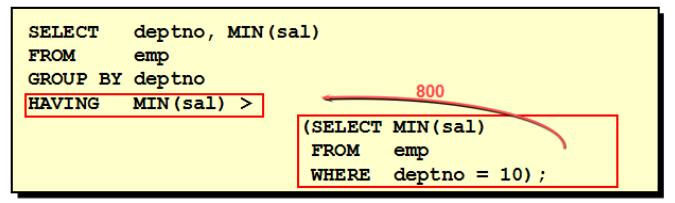
* 单行子查询示例1



* 单行子查询示例2

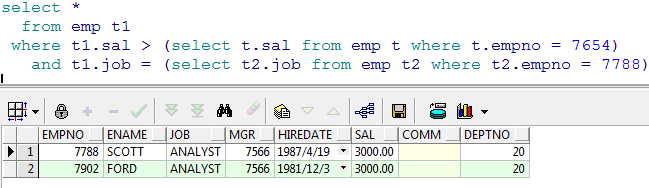


* 单行子查询示例3

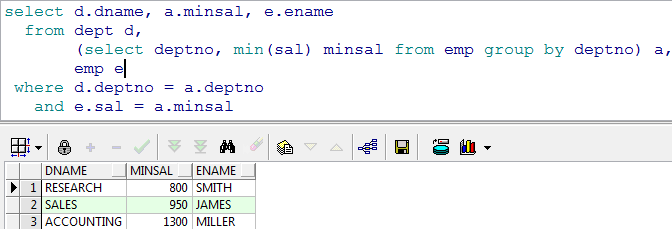


* 单行子查询示例4

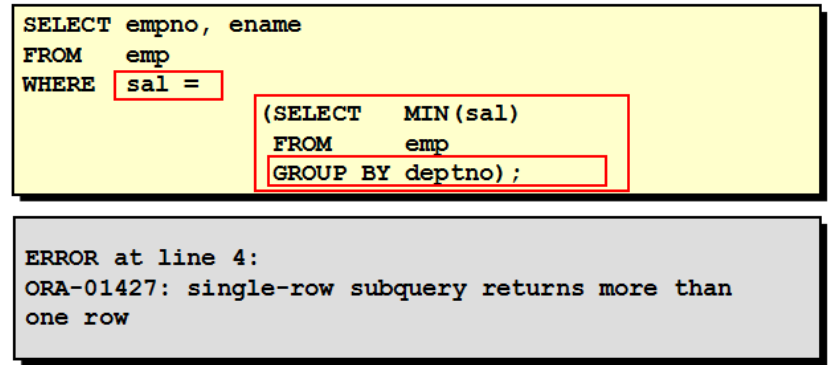
范例：查询出比雇员7654的工资高，同时从事和7788的工作一样的员工



范例：要求查询每个部门的最低工资和最低工资的雇员和部门名称



* 非法使用单行子查询示例

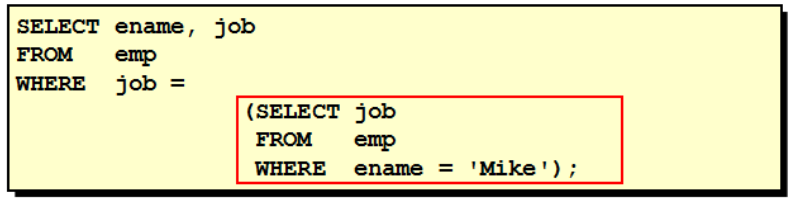


## 多行子查询

* 返回了多条记录
* 多行操作符

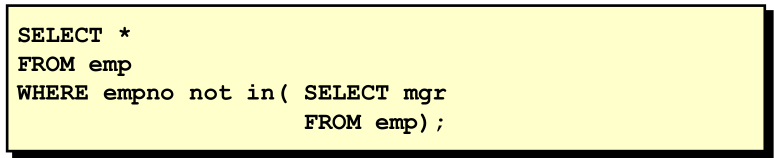
## 子查询中的null值问题

* + 单行子查询中的null值问题

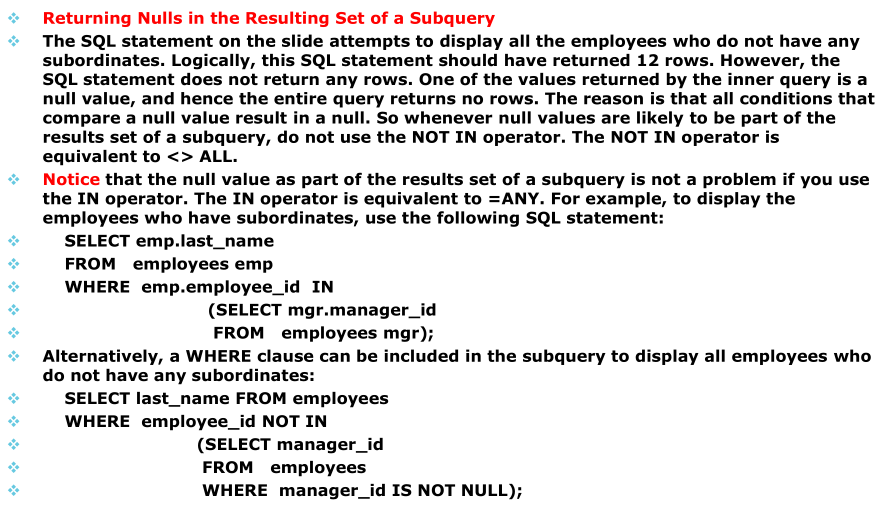


* + 多行子查询中的null值问题

示例：查询不是老板的员工



多行子查询中null值需要注意的问题：



## Exists用法：

语法：

exists(sql查询语句)

用法一：判断exists后面的sql语句是否为真

sql查询语句为空 返回值是false

sql查询语句有值 返回值就是true

select \* from emp where exists (select \* from dept where deptno=1)

等同于：select \* from emp where 1=2

select \* from emp where exists (select \* from dept where deptno=10)

等同于：select \* from emp where 1=1

用法二：判断一个表中是否包含另一张表中外键的记录

范例：查询有员工的部门

select \* from dept d

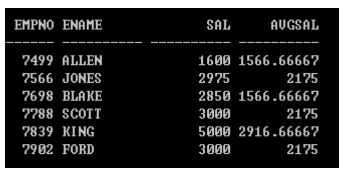
where exists (select \* from emp e where e.deptno = d.deptno)

# 课堂练习

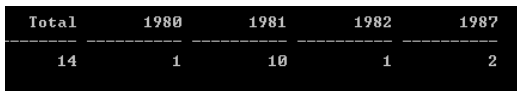
## 找到员工表中工资最高的前三名，如下格式：



## 找到员工表中薪水大于本部门平均薪水的员工。



## 统计每年入职的员工个数



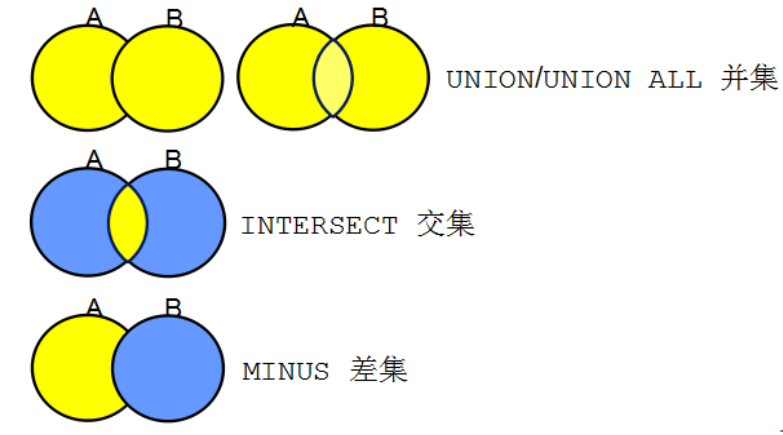
## 补充知识点：Oracle中的分页查询

ROWNUM:表示行号，实际上此是一个列,但是这个列是一个伪列,此列可以在每张表中出现。

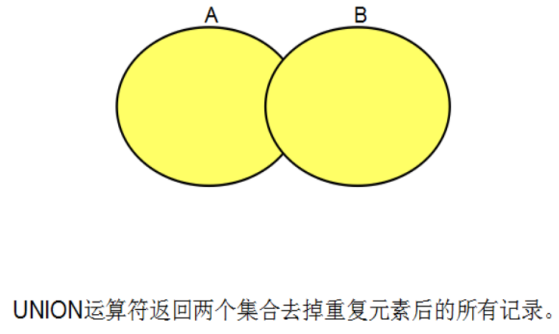
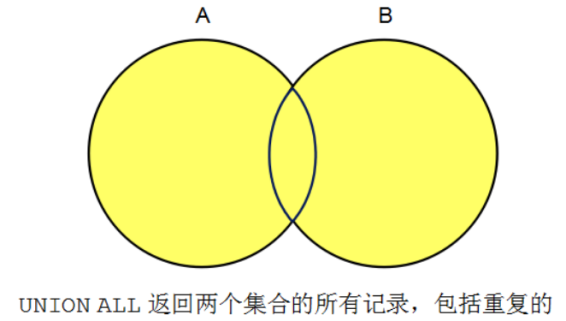
ROWID:表中没行数据指向磁盘上的物理地址。

# 集合运算

## 什么是集合运算？



## 并集

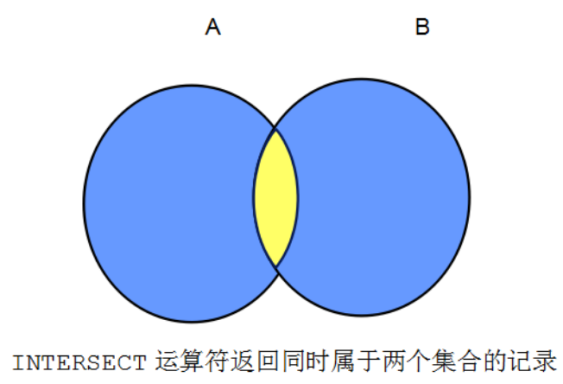
范例：工资大于1500，或者是20号部门下的员工（并集）

select \* from emp where sal>1500

union -- 或union all

select \* from emp where deptno=20;

## 交集



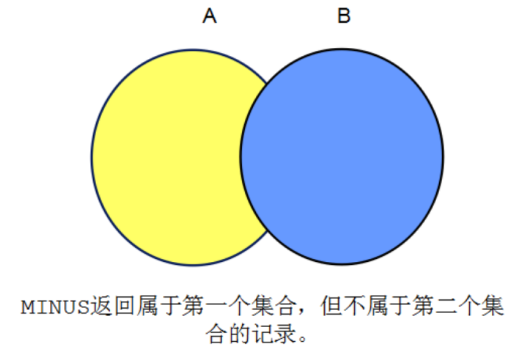
范例：工资大于1500，并且是20号部门下的员工（交集）

select \* from emp where sal>1500

intersect *--交集*

select \* from emp where deptno=20;

## 差集



--范例：1981年入职的普通员工（不包括总裁和经理）（差集）

select \* from emp where to\_char(hiredate,'yyyy')='1981'

minus

select \* from emp where job='PRESIDENT' or job='MANAGER'

## 集合运算的特征

集合运算两边查询的字段数量、字段类型、顺序必须一致