# Oracle基础

## Oracle简介

Oracle公司成立于1977年，总部位于美国加州，是世界领先的信息管理软件开发商，因其复杂的关系数据库产品而闻名。Oracle数据库产品为财富排行榜上的前1000家公司所采用，许多大型网站也选用了Oracle系统。

Oracle数据库是Oracle（中文名称叫甲骨文）公司的核心产品，Oracle数据库是一个适合于大中型企业的数据库管理系统。在所有的数据库管理系统中（比如：微软的SQL Server，IBM的DB2等），Oracle的主要用户涉及面非常广，包括：银行、电信、移动通信、航空、保险、金融、电子商务和跨国公司等。

Oracle公司成立以来，从最初的数据库版本到Oracle7、Oracle8i、Oracle9i，Oracle10g到Oracle11g，虽然每一个版本之间的操作都存在一定的差别，但是Oracle对数据的操作基本上都遵循SQL标准。因此对Oracle开发来说各版本之间的差别不大。

2008年1月16日，全球最大的数据库软件公司甲骨文（Oracle）宣布已经同BEA达成协议，以85亿美元收购BEA。

2008年1月16日，Sun宣布已经与MySQL AB达成协议，以大约10亿美元收购MySQL AB。

2009年04月20日，甲骨文宣布，该公司将以每股9.5美元的价格收购Sun。该交易价值约为74亿美元。

Oracle的官方网站：<http://www.oracle.com>

## Oracle相关的参考文档

文档：

<http://www.oracle.com/technetwork/database/database10g/documentation/index.html>

在线：

<http://www.oracle.com/pls/db102/homepage>

## Oracle中的一些概念

### Oracle数据库

位于硬盘上实际存放数据的文件，这些文件组织在一起，成为一个逻辑整体，即为 Oracle 数据库。因此在Oracle 看来，“数据库”是指硬盘上文件的逻辑集合，必须要与内存里实例合作，才能对外提供数据管理服务。

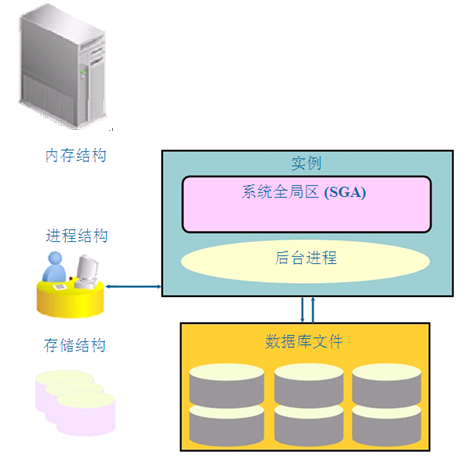
### Oracle实例

**位于物理内存里的数据结构。它由一个共享的内存池和多个后台进程所组成，共享的内存池可以被所有进程访问。**用户如果要存取数据库(也就是硬盘上的文件) 里的数据，必须通过实例才能实现，不能直接读取硬盘上的文件。

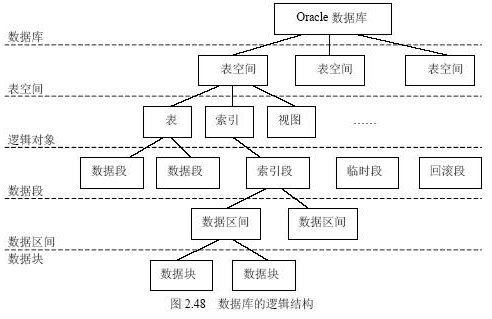
### Oracle服务器

一个Oracle服务器：是一个数据管理系统（RDBMS），它提供开放的，全面的，近乎完整的信息管理。由一个Oracle实例 和一个Oracle数据库组成。

实例可以操作数据库，在任何时刻一个实例只能与一个数据库关联。大多数情况下，一个数据库上只有一个实例对其进行操作（也可以有多个实例）。



### 数据库的逻辑和物理结构



# 搭建Oracle环境（安装与配置）

## 系统需求

内存需求：

1 GB

磁盘空间需求：

Oracle软件需要 1.5 GB 到 3.5 GB

操作系统：

根据手册文档而定

## 安装Oracle服务器端

请参见相关文档：

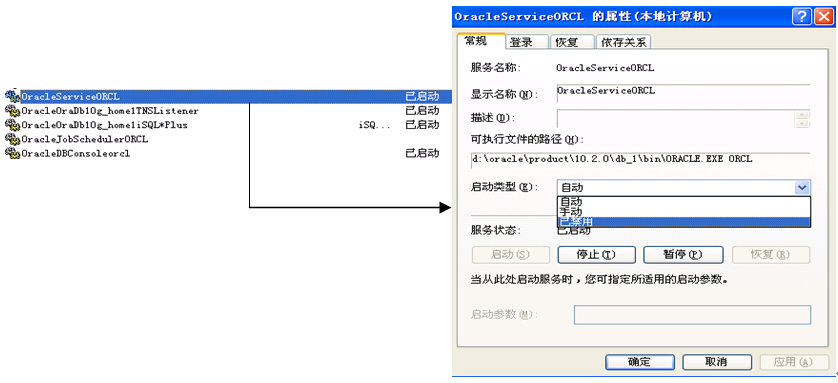
1. 《在Windows XP中安装Oracle10g》
2. 《在Win7或Vista中安装Oracle11g》
3. 《在Linux中安装Oracle》
4. 《虚拟机中安装Oracle》

全局数据库名是数据库在服务器网络中的唯一标识。

数据库创建完毕后，需要设置数据库的默认用户。Oracle中为管理员预置了两个用户分别是SYS和SYSTEM。同时Oracle为程序测试提供了一个普通用户scott，口令管理中，可以对数据库用户设置密码，设置是否锁定。 Oracle客户端使用用户名和密码登录Oracle系统后才能对数据库操作。

默认的用户中，SYS和SYSTEM用户是没有锁定的，安装成功后可以直接使用，SCOTT用户默认为锁定状态，因此不能直接使用，需要把SCOTT用户设定为非锁定状态才能正常使用。

Oracle 数据库是一个庞大的软件，启动它会占有大量的内存和CPU资源。如果不想让Oracle数据库自动启动，可做如下设置：



虽然一个Oracle数据库服务器中可以安装多个数据库，但是一个数据库需要占用非常大的内存空间，因此一般一个服务器只安装一个数据库。每一个数据库可以有很多用户，不同的用户拥有自己的数据库对象（比如：数据库表），一个用户如果访问其他用户的数据库对象，必须由对方用户授予一定的权限。不同的用户创建的表，只能被当前用户访问。因此在Oracle开发中，不同的应用程序只需使用不同的用户访问即可。

### SQL语句说明

SQL两类大小部不敏感

SQL语句可以分为一行或者多行数据 使用时最后意分号结束

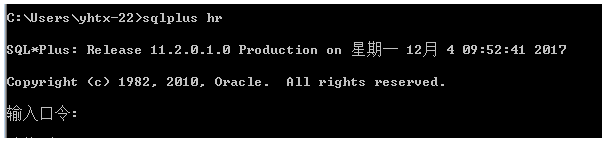
### 一些SQL\*Plus命令

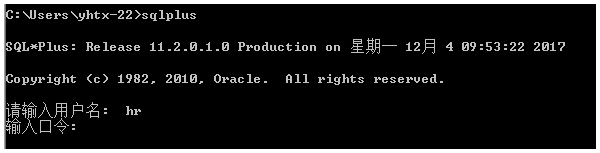
**说明：命令不区分大小写。**

#### 登录、注销

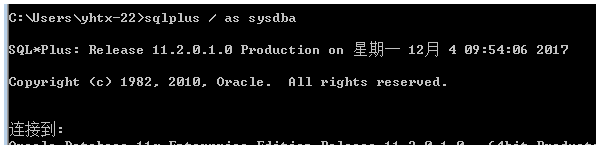
* 登录普通用户：

C:\Users\yhtx-22>sqlplus scott/scott





* 登录管理员：



* 退出：

exit

**说明：用户名不区分大小写，密码区分。**

#### 用户锁定、解锁、修改密码

以管理员的方式进行登录和相关的操作

* 解锁用户：

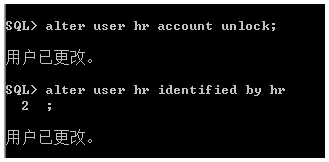
Ater user 用户名 account unlock;

* 锁定用户：

Alter user 用户名 account lock;

* 修改密码：

Alter user 用户名 identified by 新密码



* 修改管理员密码：

Alter user sys identified by 新密码

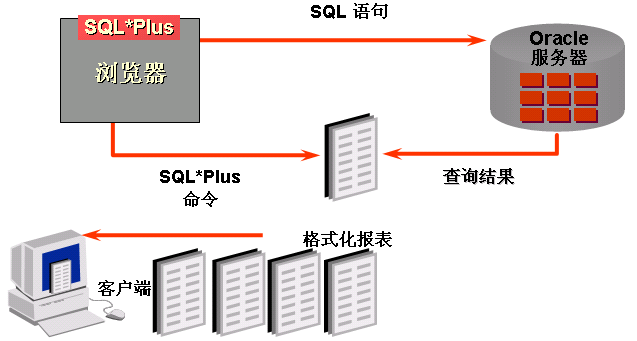
#### 中止正在执行的命令

在命令行的SqlPlus中，中止一个正在执行的命令是 Ctrl+/, Ctrl + C，如果直接按Ctrl+C会退出SqlPlus程序。

在sqlplus.exe（单独运行的程序）中，中止一个正在执行的命令是 Ctrl + C。

#### Oracle启动和关闭

### SQL语句与SqlPlus命令



SQL

* + 一种语言
  + ANSI 标准
  + 关键字不能缩写
  + 使用语句控制数据库中的表的定义信息和表中的数据

SQL\*Plus

* + 一种环境
  + Oracle的特性之一
  + 关键字可以缩写
  + 命令不能改变数据库中的数据的值

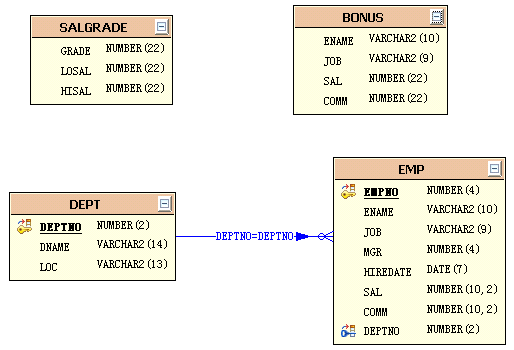
## Oracle相关的服务

**OracleService+服务名**，该服务是数据库启动的基础，只有该服务启动了，Oracle数据库才能正常启动。这是必须启动的服务。

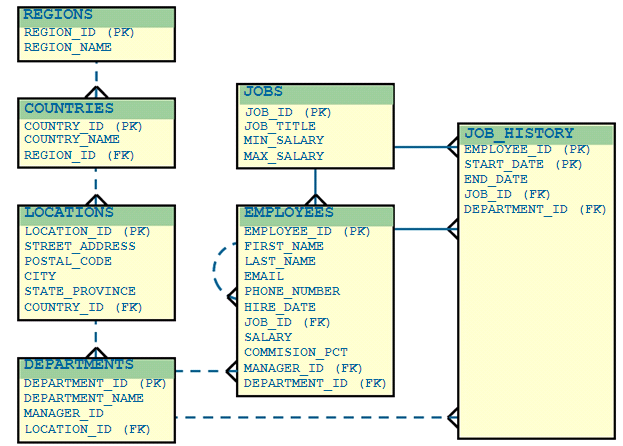
**OracleOraDb11g\_home1TNSListener**，该服务是服务器端为客户端提供的监听服务，只有该服务在服务器上正常启动，客户端才能连接到服务器。该监听服务接收客户端发出的请求，然后将请求传递给数据库服务器。一旦建立了连接，客户端和数据库服务器就能直接通信了。

## 贯穿这门课程的方案（scott方案）

**Scott方案：**



**Hr方案：**

****

1. **查询数据（基本查询）**
   1. **查询（select .. form ..）**
      1. **语法**

**SELECT \*|{[DISTINCT] *column*|*expression* [*alias*],...}**

**FROM *table;***

* **SELECT 标识 选择哪些列。**
* **FROM 标识从哪个表中选择。**
  + 1. **简单查询（全部列与指定列）**
       1. **选择全部列**

select \* from employees

* + - 1. **选择特定的列**

select last\_name,email,salary from employees

* + 1. **使用数学运算符**



* 乘除的优先级高于加减
* 优先级相同时，按照从左至右运算
* 可以使用括号改变优先级
  + 1. **空值（null）与对空值的处理**

空值是无效的 未指定的 未知的 不可预知的值

空值不是0 也不是空格

任何包含空值的数学表达式运算结果都是空值

*--任何和空值的运算都是空值*

select last\_name, job\_id,salary ,commission\_pct , 15\*(1+commission\_pct)\*salary+20000

from employees

* + 1. **列的别名**

As 之后跟别名 其中as关键字也可以省略

select last\_name sname, job\_id,salary ,commission\_pct , 15\*(1+commission\_pct)\*salary+20000 as annsal

from employees

* + 1. **使用连接符（连接字符串或列）**

|| 完成字符的拼接

select first\_name ,last\_name ,salary from employees

select first\_name||last\_name as sname,salary from employees

* + 1. **对字符串的处理**

在这里字符串只能出现在单引号内

select first\_name|| ' \_\_\_ is a \_\_ ' ||last\_name as sname,salary from employees

* + 1. **删除重复的行**

*--查询员工所在的部门有哪些 不需要重复的数据*

select distinct department\_id from employees

* 1. **过滤（Where）**
     1. **语法**

**SELECT \*|{[DISTINCT] *column|expression* [*alias*],...}**

**FROM *table***

**[WHERE *condition(s)*];**

* 使用WHERE 子句，将不满足条件的行过滤掉。
* WHERE 子句紧随 FROM 子句。
  + 1. **处理字符串与日期**

**--字符串大小写是敏感的 日期格式也是敏感**

**select last\_name, department\_id**

**from employees**

**where last\_name='Ernst'**

* + 1. **比较运算**
       1. **简单运算符**



*--查询工作低于3000的*

select last\_name ,salary

from employees

where salary<3000

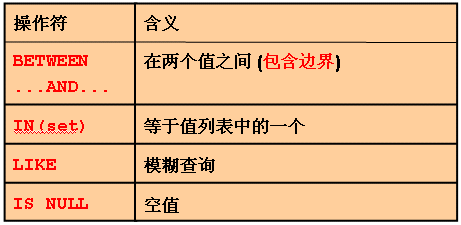
*--查询工资大于20000*

select last\_name ,salary

from employees

where salary>=10000

* + - 1. **其他运算符**



* + - * 1. **BETWEEN**

*--查询工资在8000-16000之间的员工*

select last\_name ,salary

from employees

where salary between 8000 and 16000

* + - * 1. **IN**

*-- 查询员工所在部门是 100 101 201 的所有员工*

select last\_name ,salary ,department\_id

from employees

where department\_id in(90,100,50)

* + - * 1. **LIKE**

% 代表任意个数的任意字符

\_ 代表一个任意字符

*--查询员工的姓名是以S开头的所有员工*

select first\_name ,salary

from employees

where first\_name like 'S%'

*--查询员工 的姓名是的第二个字符是o的所有的员工*

select first\_name ,salary

from employees

where first\_name like '\_o%'

*--查询 员工的姓名中包含 oo的所有的员工*

select first\_name ,salary

from employees

where first\_name like '%ee%'

* + 1. **对空值（null）的处理**

对空值的判断使用is null进行运算

*--查询这个公司的老板是谁*

select \* from employees

select last\_name, salary ,manager\_id

from employees

where manager\_id is null

* + 1. **逻辑运算**

****

* + - 1. **AND**
* **AND 要求并的关系为真。**

select last\_name ,salary ,job\_id

from employees

where salary>=10000 and job\_id like '%MAN%'

* + - 1. **OR**
* **OR 要求或关系为真**

select last\_name ,salary ,job\_id

from employees

where salary>=10000 or job\_id like '%MAN%'

* + - 1. **NOT**

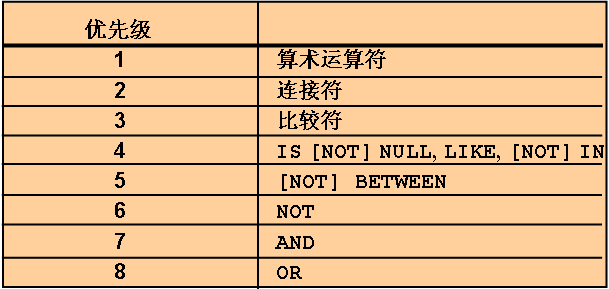
*--3.2.5.2. NOT*

select last\_name ,salary ,job\_id

from employees

where job\_id not in ('PU\_MAN','AD\_VP','ST\_MAN')

* + 1. **优先级**

****

* **可以使用括号改变优先级顺序**
  1. **排序**
     1. **语法**

使用order by 子句进行排序

asc 默认方式 进行升序排序

desc 降序排序

*--按照员工的雇佣日期进行排序 升序排序*

select last\_name,job\_id department\_id ,hire\_date

from employees

order by hire\_date asc

*--按照员工的雇佣日期进行排序 降序排序*

select last\_name,job\_id department\_id ,hire\_date

from employees

order by hire\_date desc

* + 1. **按别名排序**

*--按照员工的年薪进行排序*

select \* from employees

select last\_name,job\_id,department\_id,salary\*(1+0.2)\*15 as annsal

from employees

order by annsal desc

* + 1. **多个列排序**

按照员工的部门进行排序，如果部门号相同则按照工资进行排序

*--按照员工的部门进行排序，如果部门号相同则按照工资进行排序*

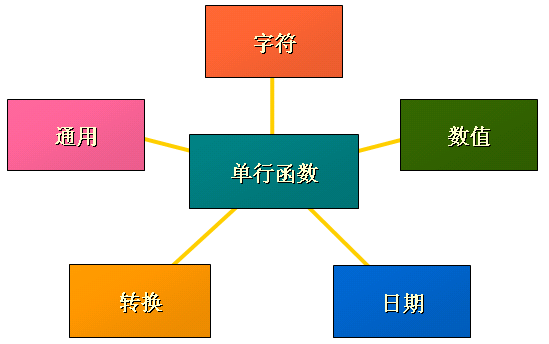
select last\_name,department\_id ,salary

from employees

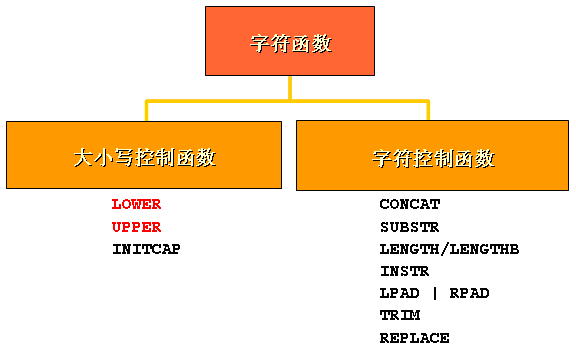
order by department\_id desc, salary desc

* 1. **单行函数**
     1. **单行函数说明**

单行函数：

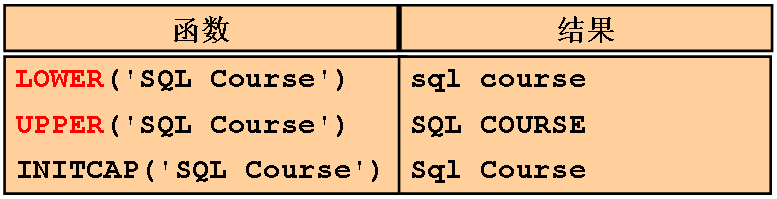


* + 1. **字符函数**



#### 3.4.2.1 大小写控制函数

* 这类函数用于改变字符的大小写。



*--dual 是虚拟的万能表*

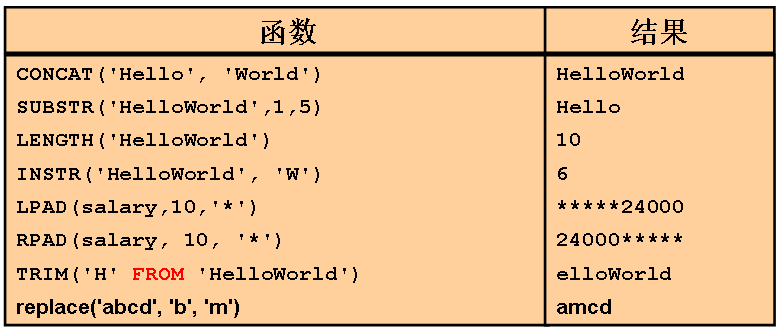
select lower('SqL CourSE') from dual;

select upper('SqL CourSE') from dual;

select initcap('SqL CourSE') from dual;

#### 3.4.2.2字符控制函数

**这类函数用于控制字符：**



select employee\_id,concat(first\_name,last\_name) as sname,job\_id,length(first\_name) length,

instr(last\_name,'a'),lpad(salary,10,'\*'),substr(last\_name,2,3),salary

from employees

where substr(job\_id,4,3)=upper('rep')

order by salary desc

* + 1. **数字函数**
* **ROUND: 四舍五入**

**例：ROUND(45.926, 2) ，结果45.93**

* **TRUNC: 截断**

**例：TRUNC(45.926, 2) ，结果45.92**

* **MOD: 求余**

**例：MOD(1600, 300) ，结果100**

select round(36.973,2),round(36.966,2),trunc(36.979,2),trunc(36.979,-2),trunc(362.979,-2),mod(1500,300)

from dual;

* + 1. **日期**
* Oracle 中的日期型数据实际含有两个值: 日期和时间。
* 默认的日期格式是 DD-MON-RR，例如下面的hire\_date的值：

*--获取系统时间*

select sysdate from dual

select first\_name, last\_name,hire\_date

from employees

#### 3.4.4.1日期的数学运算

* 在日期上加上或减去一个数字结果仍为日期。
* **两个日期相减返回日期之间相差的天数**。
* 可以用数字除24来向日期中加上或减去小时。

*--两个日期相减得到的是相差的天数*

select e.first\_name,e.last\_name, trunc((sysdate-e.hire\_date)/7) as weekend

from employees e

where e.department\_id=90

#### 3.4.4.2日期函数



*--系统版本的差别 有的默认是中文 有的默认是英文*

*--select months\_between('04-11月-2017','12-9月-2017') from dual;*

select months\_between('04-January-2017','12-March-2017') from dual;

select months\_between('04-jan-2017','12-mar-2017') from dual;

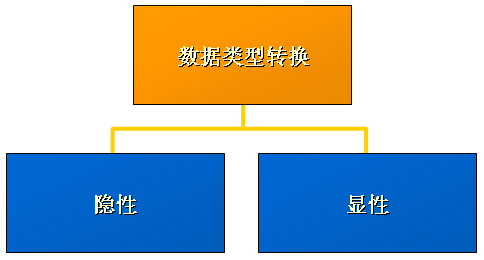
select add\_months('4-December-2017',6) from dual

*--下一个指定的日期*

select next\_day('4-December-2017','sat') from dual

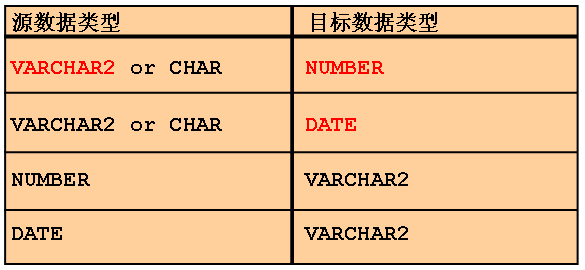
select last\_day('4-December-2017') from dual

* + 1. **转换函数**

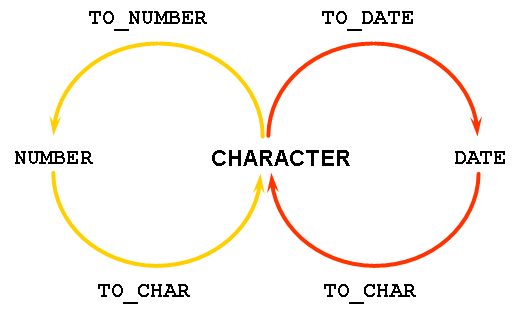


* **隐式数据类型转换**

Oracle 自动完成下列转换：



* **显式数据类型转换**



#### 3.4.5.1 TO\_CHAR 函数对日期的转换

**语法：TO\_CHAR(*date,* '*format\_model*')**

格式：

* **必须包含在单引号中而且大小写敏感**。
* 可以包含任意的**有效的日期格式**。
* **日期之间用逗号隔开**。
* **日期格式的元素**

|  |  |  |
| --- | --- | --- |
| **格式** | **说明** | **举例** |
| YYYY | Full year in numbers | 2011 |
| YEAR | Year spelled out(年的英文全称) | twenty eleven |
| MM | Two-digit value of month 月份（两位数字） | 04 |
| MONTH | Full name of the month（月的全称） | 4月 |
| DY | Three-letter abbreviation of the day of the week(星期几) | 星期一 |
| DAY | Full name of the day of the week | 星期一 |
| DD | Numeric day of the month | 02 |

*--to\_char*

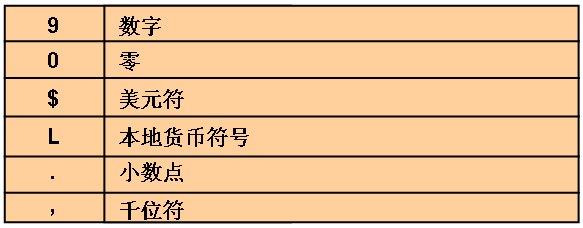
select e.first\_name,e.last\_name ,to\_char(e.hire\_date,'YYYY--MM--DD') as da

from employees e

#### 3.4.5.2TO\_CHAR 函数对数字的转换

**语法：TO\_CHAR(*number,* '*format\_model*')**

下面是在TO\_CHAR 函数中经常使用的几种格式:



select e.first\_name,to\_char(e.salary,'L99,999,999.00') as sal

from employees e

* + 1. **通用函数**

这些函数**适用于任何数据类型，同时也适用于空值**：

* NVL (expr1, expr2)
* NVL2 (expr1, expr2, expr3)
* NULLIF (expr1, expr2)
* COALESCE (expr1, expr2, ..., expr*n*)

**NVL函数将空值转换成一个已知的值**：

* 可以使用的数据类型有日期、字符、数字。
* 函数的一般形式:
  + NVL(commission\_pct,0)
  + NVL(hire\_date,'01-JAN-97')
  + NVL(job\_id,'No Job Yet')

select e.first\_name,e.salary,nvl(e.commission\_pct,0) commi,e.salary\*(1+nvl(e.commission\_pct,0))\*15+20000 as annsal

from employees e

**NVL2 (expr1, expr2, expr3) : expr1不为NULL，返回expr2；为NULL，返回expr3。**

*--NVL2 (expr1, expr2, expr3) : expr1不为NULL，返回expr2；为NULL，返回expr3。*

select e.first\_name,e.salary,e.commission\_pct, nvl2(e.commission\_pct,'sal+comm','sal')

from employees e

where e.department\_id in (70,80,90)

**NULLIF (expr1, expr2) : 相等返回NULL，不等返回expr1**

select e.first\_name, length(e.first\_name) ,e.last\_name,length(e.last\_name),nullif(length(e.first\_name),length(e.last\_name))

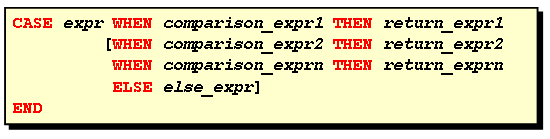
from employees e

* **使用 COALESCE 函数**
* COALESCE 与 NVL 相比的优点在于 COALESCE 可以同时处理交替的多个值。
* 如果第一个表达式为空,则返回下一个表达式，对其他的参数进行COALESCE 。
* **即：找第一个不为空的值。**

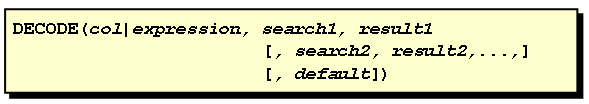
select e.first\_name, e.commission\_pct,e.department\_id,e.salary, coalesce(e.commission\_pct,e.department\_id,e.salary,10)

from employees e

* + 1. **条件表达式**
* 在 SQL 语句中使用IF-THEN-ELSE 逻辑
* 使用两种方法:
  + CASE 表达式：SQL99的语法，类似Basic，比较繁琐
  + DECODE 函数：Oracle自己的语法，类似Java，比较简介
* **CASE 表达式**



* **DECODE 函数**



select e.last\_name,e.job\_id,e.salary,

case e.job\_id when 'IT\_PROG' then 2.0\*salary

when 'ST\_CLERK' then 1.15\*salary

when 'SA\_REP' then 1.01\*salary

else e.salary

end "rev\_salary"

from employees e

select e.last\_name,e.job\_id,e.salary,

decode (e.job\_id , 'IT\_PROG' , 2.0\*salary ,

'ST\_CLERK' , 1.15\*salary,

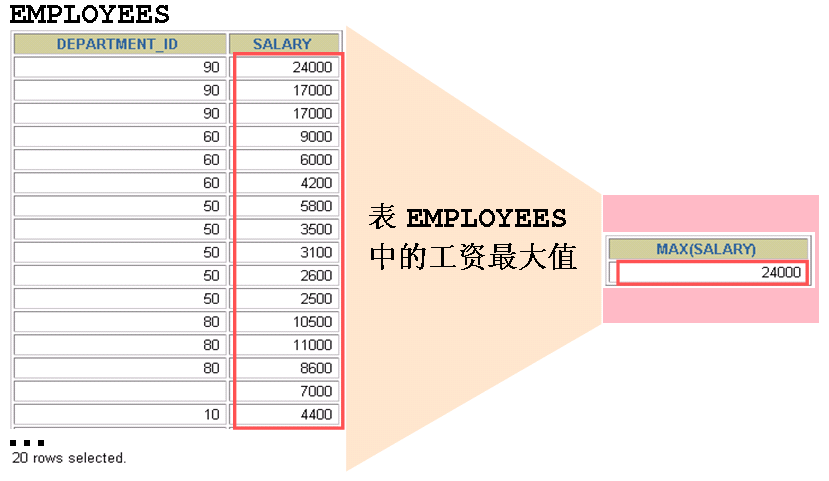
'SA\_REP' , 1.01\*salary,

e.salary )"rev\_salary"

from employees e

* 1. **分组函数（多行函数）**
     1. **什么是分组函数**

分组函数作用于一组数据，并对一组数据返回一个值。



* + 1. **组函数使用**
* **类型：**
* **AVG 平均值**
* **COUNT 数量**
* **MAX 最大值**
* **MIN 最小值**
* **SUM 总和**

select count(\*) from employees

select avg(salary) from employees

select max(salary) from employees

select min(salary) from employees

select sum(salary) from employees

* + 1. **AVG（平均值）和 SUM （合计）函数**
    2. **MIN（最小值） 和 MAX（最大值）函数**

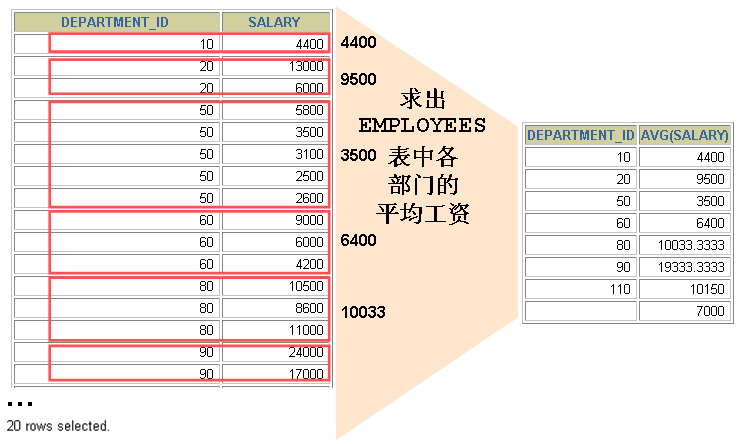
可以对任意数据类型的数据使用 MIN 和 MAX 函数。

* + 1. **COUNT（计数）函数**
* COUNT(\*) 返回表中记录总数。
  + 1. **组函数忽略空值。**
* **组函数忽略空值。**
* **在组函数中使用NVL函数，NVL函数使分组函数无法忽略空值。**

select avg(e.commission\_pct) from employees e

select avg( nvl(e.commission\_pct,0)) from employees e

* + 1. **分组数据**
* **EMPLOYEES**



* **GROUP BY 子句语法**

**可以使用GROUP BY 子句将表中的数据分成若干组**

* **在SELECT 列表中所有未包含在组函数中的列都应该包含在 GROUP BY 子句中。**
* **包含在 GROUP BY 子句中的列不必包含在SELECT 列表中**

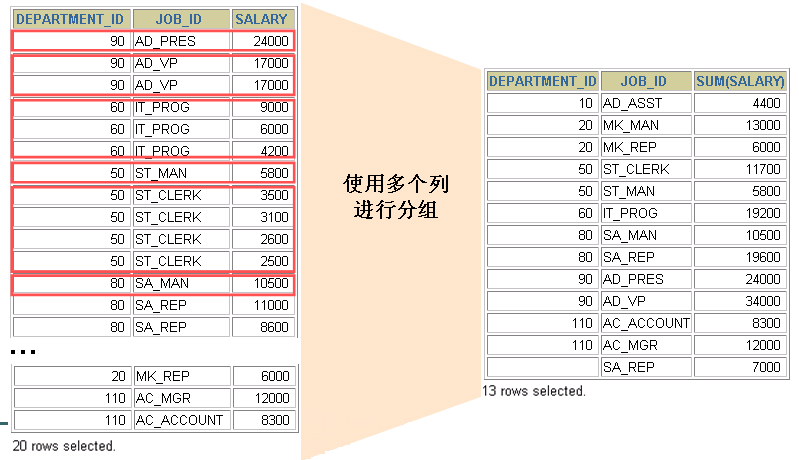
select e.department\_id, avg(salary)

from employees e

group by e.department\_id

order by e.department\_id

* **使用多个列分组**

****

* **在GROUP BY 子句中包含多个列**

*---根据部门编号和job\_id进行分组 求平均工资*

select e.department\_id,e.job\_id,avg(salary),max(salary) ,min(salary)

from employees e

group by e.department\_id,e.job\_id

order by e.department\_id

* **非法使用组函数**

不能在where子句使用组函数

可以使用having子句

下面的代码会报错。

select e.department\_id,e.job\_id,avg(salary),max(salary) ,min(salary)

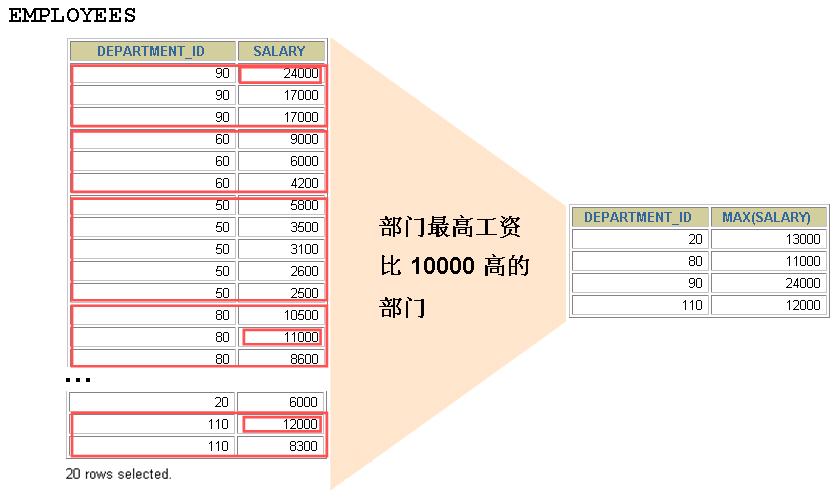
from employees e

where avg(salary)>8000

group by e.department\_id,e.job\_id

order by e.department\_id

* + 1. **过滤分组（Having）**



*-------根据部门编号和job\_id进行分组 求平均工资*

select e.department\_id,e.job\_id,avg(salary),max(salary) ,min(salary)

from employees e

group by e.department\_id,e.job\_id

having avg(salary)>8000 *--在分组以后使用过滤操作*

order by e.department\_id

*---根据部门分组 求 最大工资高于12000的部门*

select e.department\_id,max(e.salary),min(e.salary)

from employees e

group by e.department\_id

having max(e.salary)>12000

* + 1. **组函数嵌套**

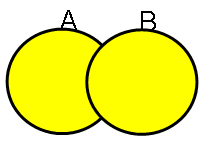
*--求这个公司每个部门的平均工资的最大值*

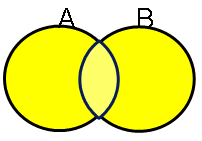
select max(avg(salary))

from employees e

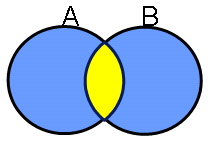
group by e.department\_id

1. **集合运算**
   1. **集合运算的类型与集合运算符**
      1. **UNION/UNION ALL 并集**

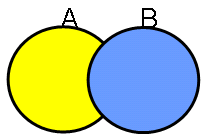




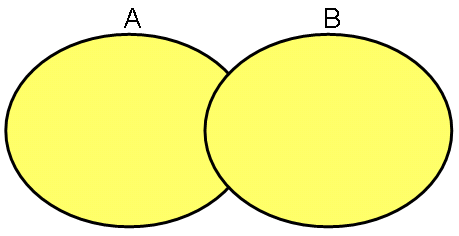
* + 1. **INTERSECT 交集**



* + 1. **MINUS 差集**

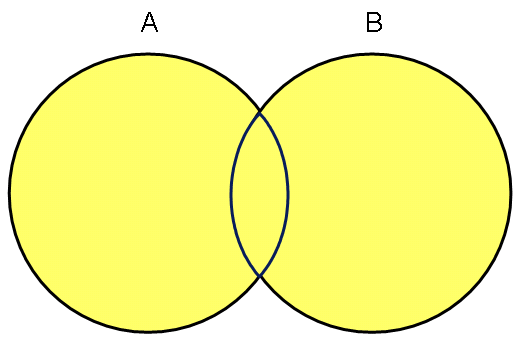


* 1. **并集（Union与Union All）**
     1. **并集（Union）**



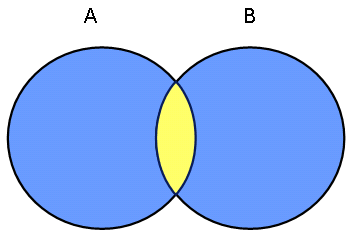
**UNION运算符返回两个集合去掉重复元素后的所有记录。**

* + 1. **并集（Union All）**



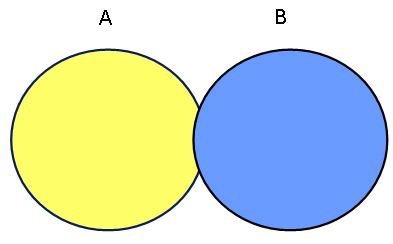
**UNION ALL 返回两个集合的所有记录，包括重复的。**

* 1. **交集（Intersect）**



**INTERSECT 运算符返回同时属于两个集合的记录。**

* 1. **差集（Minus）**



**MINUS返回属于第一个集合，但不属于第二个集合的记录。**

* 1. **集合运算的注意事项**

*--显示当前员工 工作情况和之前的工作情况*

select e.employee\_id, e.job\_id from employees e

union

select j.employee\_id ,j.job\_id from job\_history j

select e.employee\_id, e.job\_id from employees e

union all

select j.employee\_id ,j.job\_id from job\_history j

select e.employee\_id, e.job\_id from employees e

INTERSECT

select j.employee\_id ,j.job\_id from job\_history j

select e.employee\_id, e.job\_id from employees e

minus

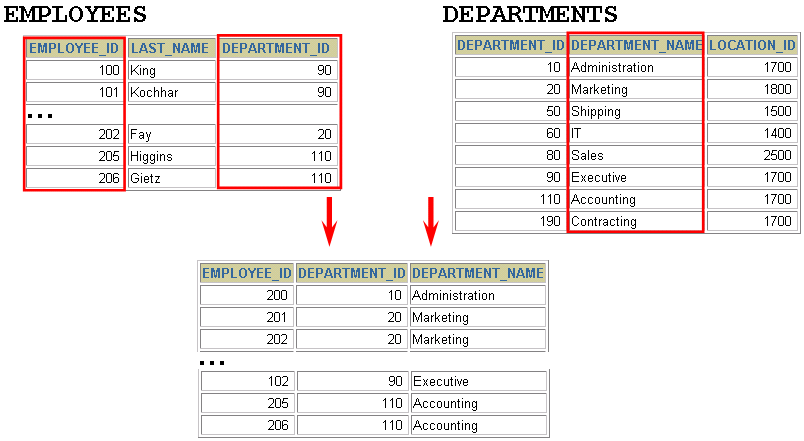
select j.employee\_id ,j.job\_id from job\_history j

select j.employee\_id ,j.job\_id from job\_history j

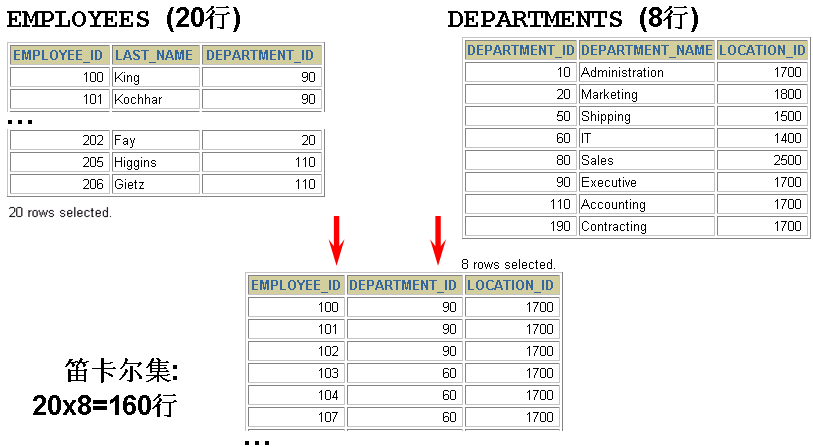
minus

select e.employee\_id, e.job\_id from employees e

1. **多表查询**
   1. **基础（相关说明与笛卡尔积）**
      1. **从多个表中获取数据**



* + 1. **笛卡尔集**



省略链接条件 表中的数据 所有的行相互链接

*--笛卡尔积 没有连接条件*

select e.employee\_id, e.first\_name,d.department\_name

from employees e,departments d

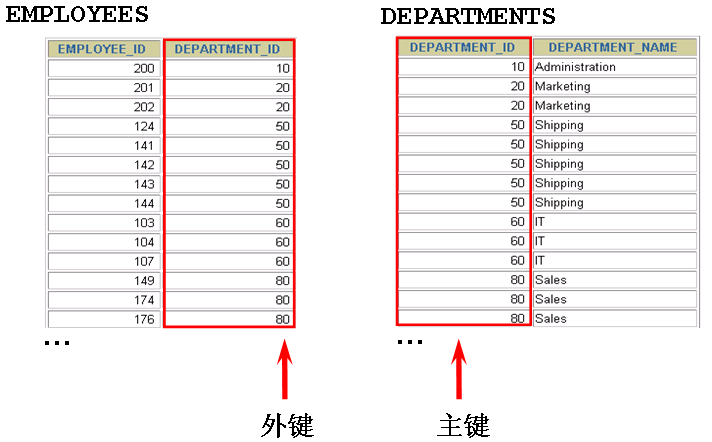
* 1. **连接的类型**
     1. **Oracle 连接:**
* 等值连接
* 不等值连接
* 外连接
* 自连接
  + 1. **SQL: 1999**
* Cross join
* Natural join
* Using caluse
* Full or two side outer join
  1. **Oracle的连接查询**
     1. **语法**

Select table1.column, table2.column

From table1,table2

Where tabl1.column1=table2.column2

* + 1. **等值连接**



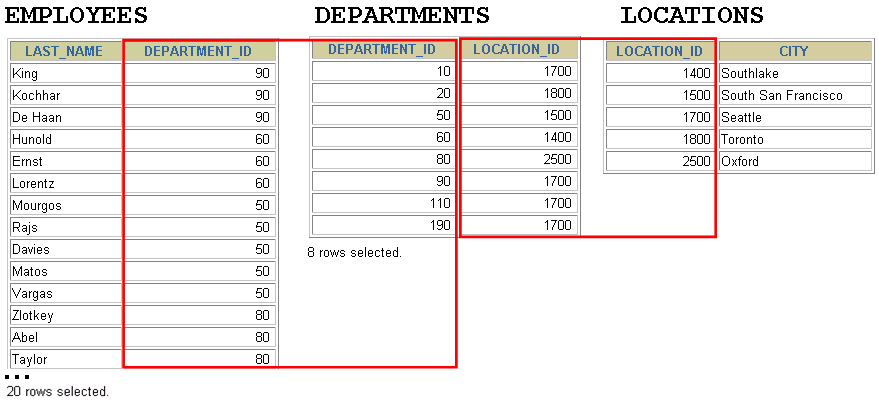
*--106条记录 有一个员工是没有部门编号的*

select e.employee\_id, e.first\_name,d.department\_name

from employees e,departments d

where e.department\_id=d.department\_id

* + 1. **多个连接条件与 AND 操作符**
    2. **连接多个表**



*--查询员工的姓名 员工的编号 员工的部门名称 员工部门所在城市的名字*

select e.employee\_id, e.first\_name,d.department\_name,l.city

from employees e, departments d,locations l

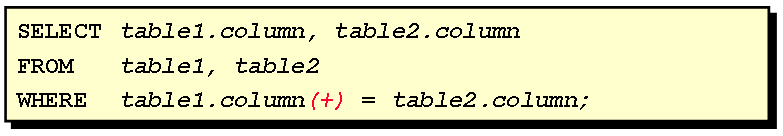
where e.department\_id=d.department\_id and d.location\_id=l.location\_id

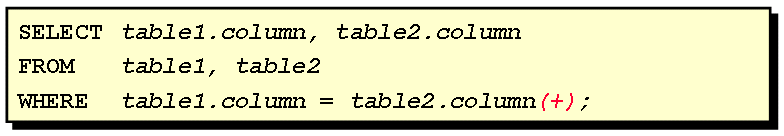
如果是n张表进行关联 至少需要n-1个条件

* + 1. **外连接**

外链接是指将不满足条件的内容也筛选出来

外链接的符号是(+)

****



*--查询员工的id 姓名 部门名称 要求 显示所有员工信息 没有部门编号的员工信息也要显示出来*

*--要显示那个表的全部信息 将+ 写在对方的位置*

select e.employee\_id, e.first\_name,d.department\_name

from employees e,departments d

where e.department\_id=d.department\_id(+)

*----查询员工的id 姓名 部门名称 要求 显示所有部门的名称 有的部门没有员工 也要显示出来*

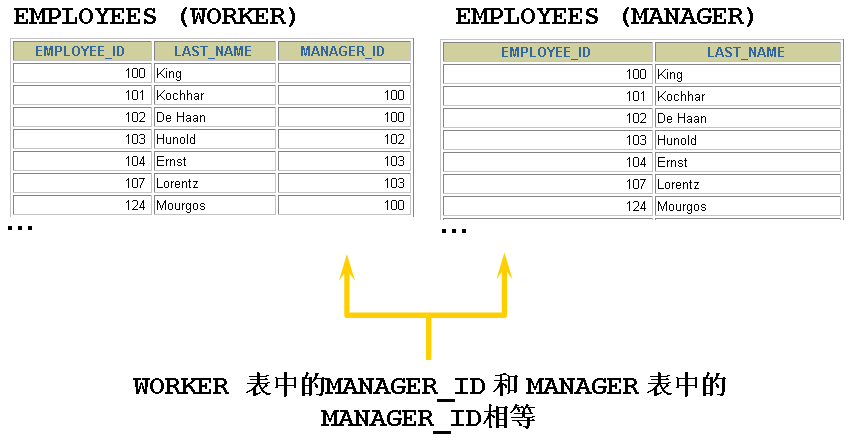
select \* from departments

select e.employee\_id, e.first\_name,d.department\_name

from employees e,departments d

where e.department\_id(+)=d.department\_id

* + 1. **自连接**



select e.employee\_id,e.first\_name,ee.first\_name

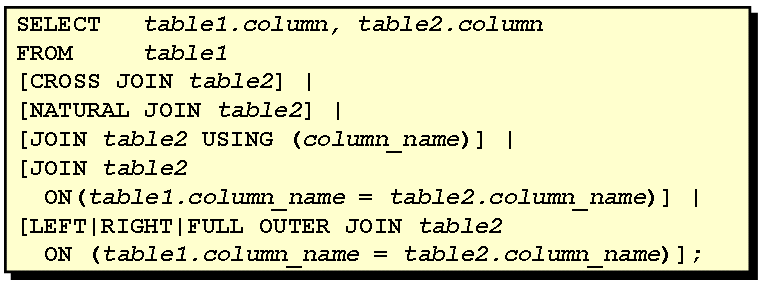
from employees e ,employees ee

where e.manager\_id=ee.employee\_id

order by e.employee\_id

* 1. **SQL99标准的连接查询**
     1. **使用连接从多个表中查询数据**

使用连接从多个表中查询数据：



select e.employee\_id,e.first\_name,d.department\_name

from employees e join departments d

on e.department\_id =d.department\_id\

* + 1. **叉集**

和oracle语法中的笛卡尔积是完全一致的

select e.employee\_id,e.first\_name,d.department\_name

from employees e cross join departments d

* + 1. **自然连接**

Natural join 子句 会以两个表中相同名字的列作为连接条件 如果仅仅是列名相同，数据类型不同 会报错

select d.department\_id ,d.department\_name,l.city

from departments d natural join locations l

* + 1. **使用 USING 子句创建连接**

可以在自然连接的时候指定使用哪个列进行连接

select e.employee\_id,e.first\_name, d.department\_name

from employees e join departments d

using(department\_id)

* + 1. **使用ON 子句创建连接**

*--查询员工的id 员工的姓名 员工部门的名称 员工部门所在的城市*

select e.employee\_id,e.first\_name,d.department\_name,l.city

from employees e join departments d on e.department\_id=d.department\_id

join locations l on d.location\_id=l.location\_id

* + 1. **内连接和外连接(2)**
* 在SQL: 1999中，内连接只返回满足连接条件的数据
* 两个表在连接过程中除了返回满足连接条件的行以外还返回左（或右）表中不满足条件的行 ，这种连接称为左（或右） 外联接。
* 两个表在连接过程中除了返回满足连接条件的行以外还返回两个表中不满足条件的行 ，这种连接称为**满外联接**。
  + 1. **左外联接**

*--左外连接 右外连接 sql99标准中还有一个满外连接*

*--查询员工的id 姓名 部门名称 要求 显示所有员工信息 没有部门编号的员工信息也要显示出来*

*--sql99标准*

select e.employee\_id, e.first\_name,d.department\_name

from employees e,departments d

where e.department\_id=d.department\_id(+)

select e.employee\_id, e.first\_name,d.department\_name

from employees e left outer join departments d

on e.department\_id=d.department\_id

*----查询员工的id 姓名 部门名称 要求 显示所有部门的名称 有的部门没有员工 也要显示出来*

select \* from departments

select e.employee\_id, e.first\_name,d.department\_name

from employees e,departments d

where e.department\_id(+)=d.department\_id

select e.employee\_id, e.first\_name,d.department\_name

from employees e right outer join departments d

on e.department\_id=d.department\_id

*----查询员工的id 姓名 部门名称 要求 显示所有的部门 所有的员工*

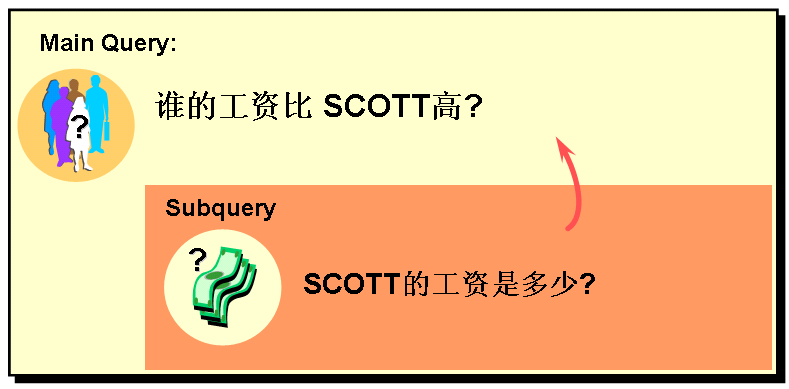
select e.employee\_id, e.first\_name,d.department\_name

from employees e full outer join departments d

on e.department\_id=d.department\_id

1. **子查询**

谁的工资比 SCOTT高?

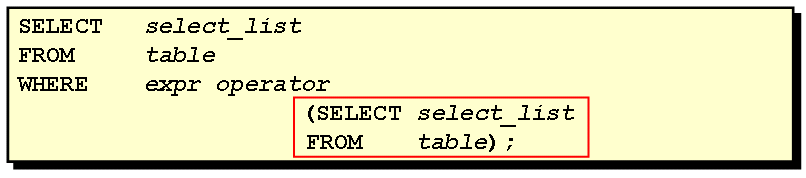


*--谁的工资比 Hunold?*

select e.first\_name ,e.salary from employees e where e.salary>(

select salary from employees e where lower(e.last\_name)='hunold')

* 1. **子查询语法**



* 1. **注意事项**

子查询要包含在括号内

将子查询放在比较条件的右侧

单行操作符对应单行子查询，多行操作符对应多行子查询

* 1. **单行子查询**

只返回一个行数据

使用的是单行比较付

* **在子查询中使用组函数**

*--TOP-N分析 查询工资最高的四个人*

*--oracle中为每个对象都有一个隐藏的rowid 和rownum*

select rowid ,rownum, e.\*

from employees e

*--固化排序完成的表*

select emp.\* from

(select e.first\_name,e.salary

from employees e

order by e.salary desc) emp

where rownum<=4

*--TOP-N分析 查询工资第七名 到第15名*

*--下面的代码查询无数据*

select rownum, emp.\* from

(select e.first\_name,e.salary

from employees e

order by e.salary desc) emp

where rownum>=7 and rownum<=15

*--rownum是从第一个数据开始 不能从中间的数据开始 否则查询无数据 将rownum固化普通的列*

select t2.first\_name,t2.salary from

(select rownum rn,t1.\* from

(select \* from employees e order by e.salary desc)t1)t2

where rn>=7 and rn<=15

*--分页查询*

*--每页显示数据10条 查询第 7页的数据*

*--1页 1-10*

*--2页 11-20 7页 71-80*

select \* from employees e order by e.salary desc

select t2.first\_name,t2.salary from

(select rownum rn,t1.\* from

(select \* from employees e order by e.salary desc)t1)t2

where rn>=71 and rn<=80

*--在子查询中使用组函数*

select e.first\_name,e.salary

from employees e

where e.salary=( select min(salary) from employees)

* **子查询中的 HAVING 子句**
* 首先执行子查询。
* 向主查询中的HAVING 子句返回结果。

*--查询最低工资比60号部门最低工资还要低的部门有哪些*

select department\_id,min(salary)

from employees

group by department\_id

having min(salary)<(

select min(salary) from employees e where e.department\_id=60)

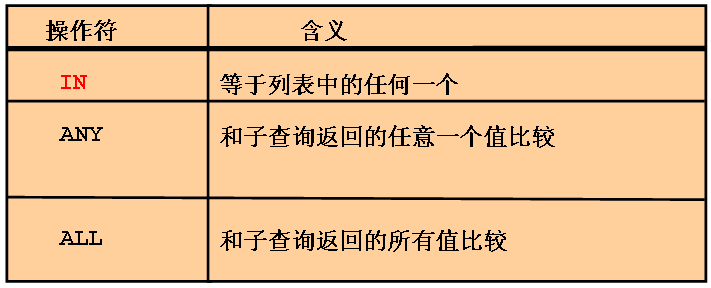
* 1. **非法使用子查询**
* **多行子查询使用单行比较符**

select e.salary

from employees e where e.salary>(

select ee.salary from employees ee where ee.salary<10000)

* 1. **多行子查询**
* 返回多行。
* 使用多行比较操作符。



* **在多行子查询中使用 in操作符**
* **在多行子查询中使用 Any 操作符**
* **在多行子查询中使用 Any 操作符**

*--in 等于子查询列表中的每一个数据*

select e.salary

from employees e where e.salary in (

select ee.salary from employees ee where ee.salary<10000)

*--any 大于指定列表中的任意一个数据即满足条件*

select e.salary

from employees e where e.salary > any (

select ee.salary from employees ee where ee.salary<10000)

*--all 大于指定列表中的每一个数据*

select e.salary

from employees e where e.salary > all(

select ee.salary from employees ee where ee.salary<10000)

* 1. **分级查询**

*--可以明确的看到上下级关旭 分级查询可以从上到下查询，也可以从下往上查询*

select employee\_id,first\_name,job\_id,manager\_id

from employees

start with employee\_id=110

connect by prior manager\_id =employee\_id

*--从顶部到底部继续查询*

select last\_name || ' reports to ' || prior last\_name

from employees e

start with last\_name='King'

connect by prior employee\_id=manager\_id

*--使用分层查询 level*

select lpad(last\_name,length(last\_name)+(level\*2)-2,'\_') as org\_chart

from employees e

start with e.employee\_id=100

connect by prior employee\_id =manager\_id