

Oracle 应用开发实战

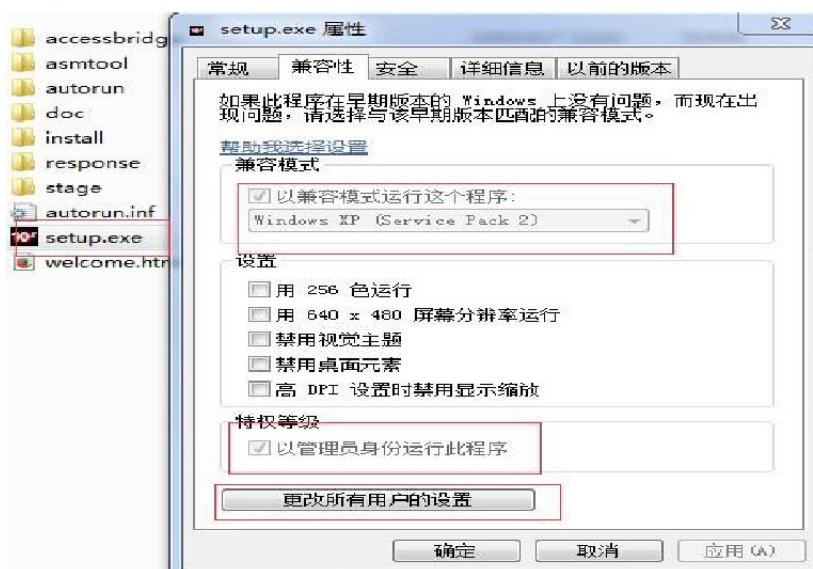
一、Oracle 的基本概念和安装

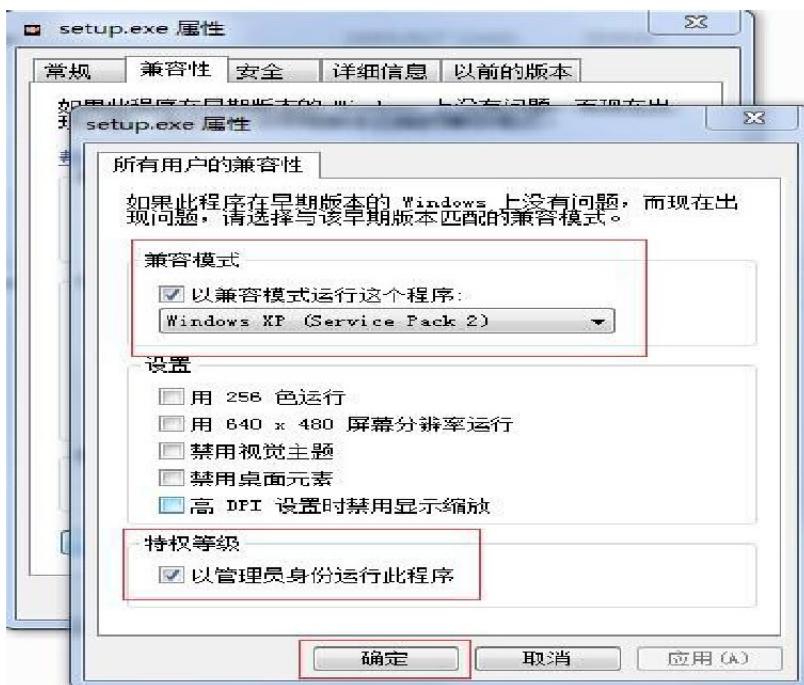
● Oracle 简介

ORACLE 数据库系统是美国 ORACLE 公司（甲骨文）提供的以分布式数据库为核心的一组软件产品，是目前最流行的客户/服务器(CLIENT/SERVER)或 B/S 体系结构的数据库之一。比如 SilverStream 就是基于数据库的一种中间件。ORACLE 数据库是目前世界上使用最为广泛的数据库管理系统，作为一个通用的数据库系统，它具有完整的数据管理功能；作为一个关系数据库，它是一个完备关系的产品；作为分布式数据库它实现了分布式处理功能。但它所有的知识，只要在一种机型上学习了 ORACLE 知识，便能在各种类型的机器上使用它。

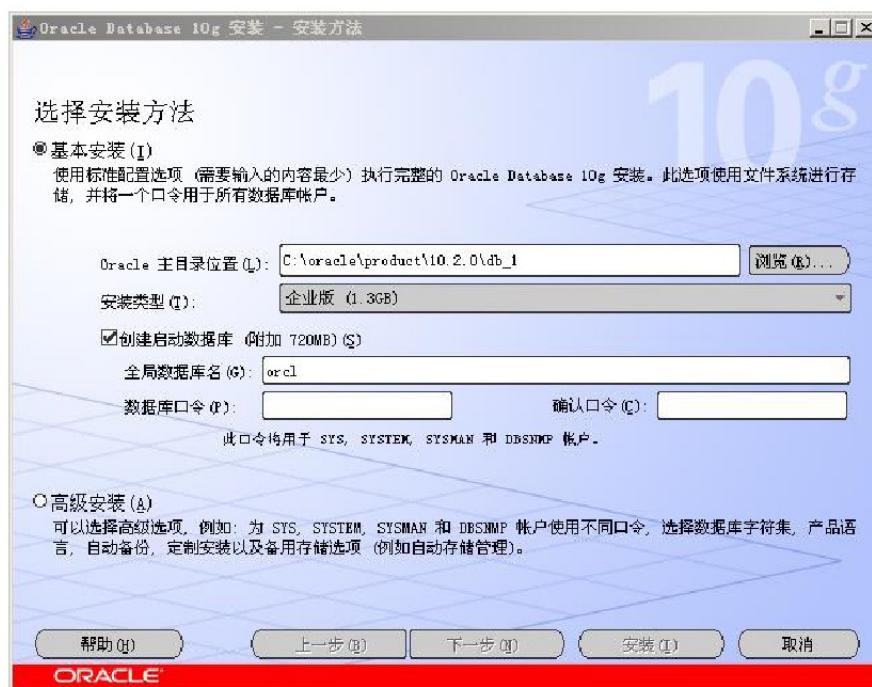
● Oracle10g 的安装

1. 解压 oracle 数据库安装包，**如果是 win7 或者 win8 系统**右键点击 setup.exe 选择兼容性，以 xp 方式，并且以管理员方式运行，以及其他所有用户都按着此规则如图



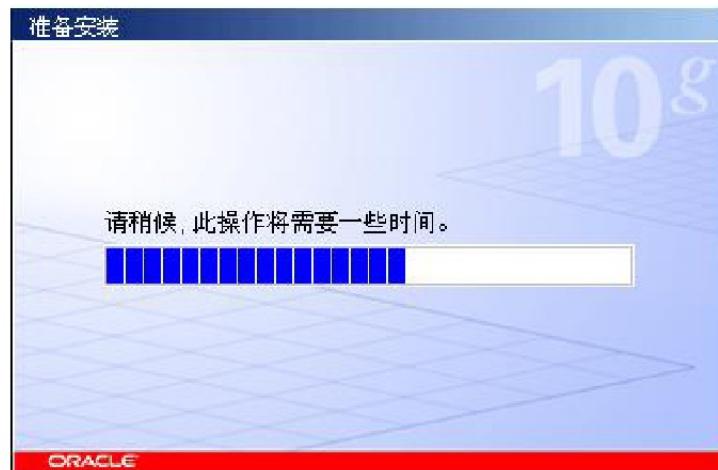


2. 如果是 xp 系统可以直接并双击解压目录下的 setup.exe，出现安装界面，如下：

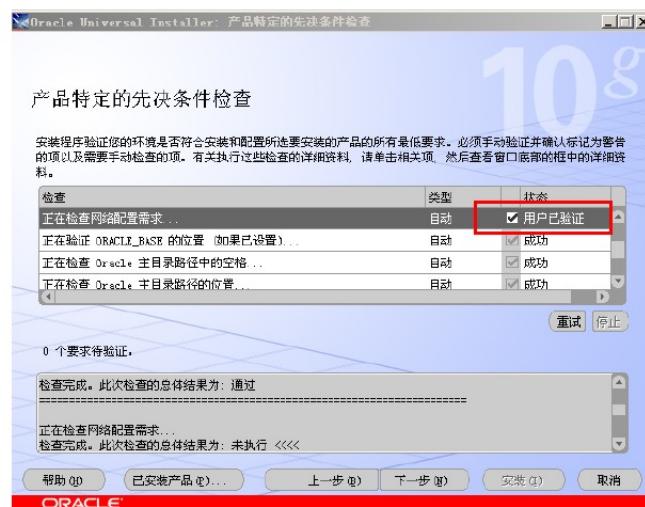


3. 输入口令和确认口令，如：itcast，点击下一步，出现如下进度条，

注：此口令即是管理员密码。



4. 检查先决条件，选中红框所示的选择框，如下图：



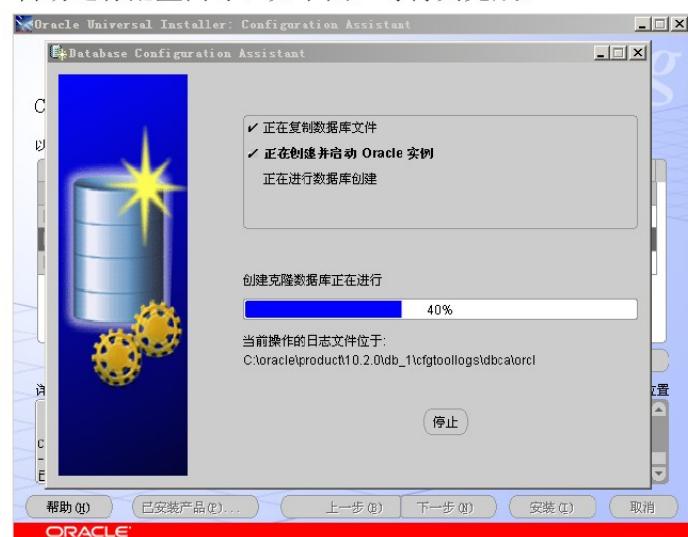
5. 点击“下一步”，出现“概要”界面，点击“安装”。



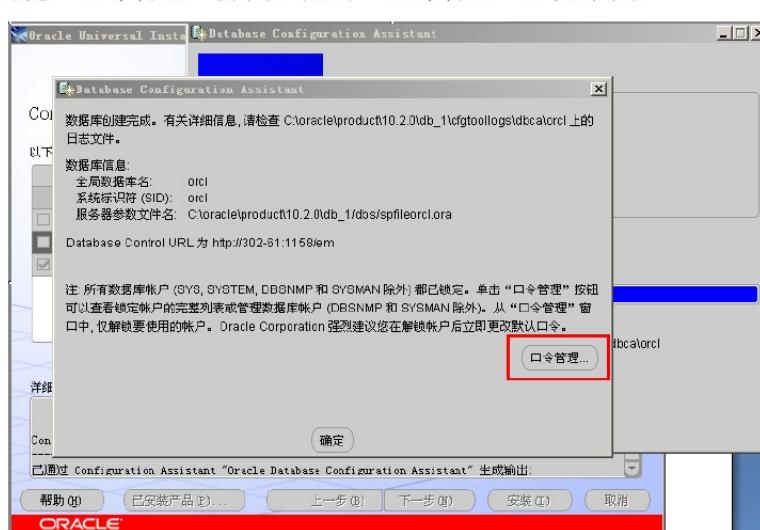
6. 出现安装进度条，等待安装完成，如下图：



7. 安装完成后，自动运行配置向导，如下图，等待其完成：



8. 完成后，出现“口令管理”界面，点击“口令管理”，如下图：



9. 将 SCOTT 和 HR 用户的勾去掉（解锁这两个账户），如下图所示，点击“确定”：



10. 回到“口令管理”界面，点击“确定”，如下图：



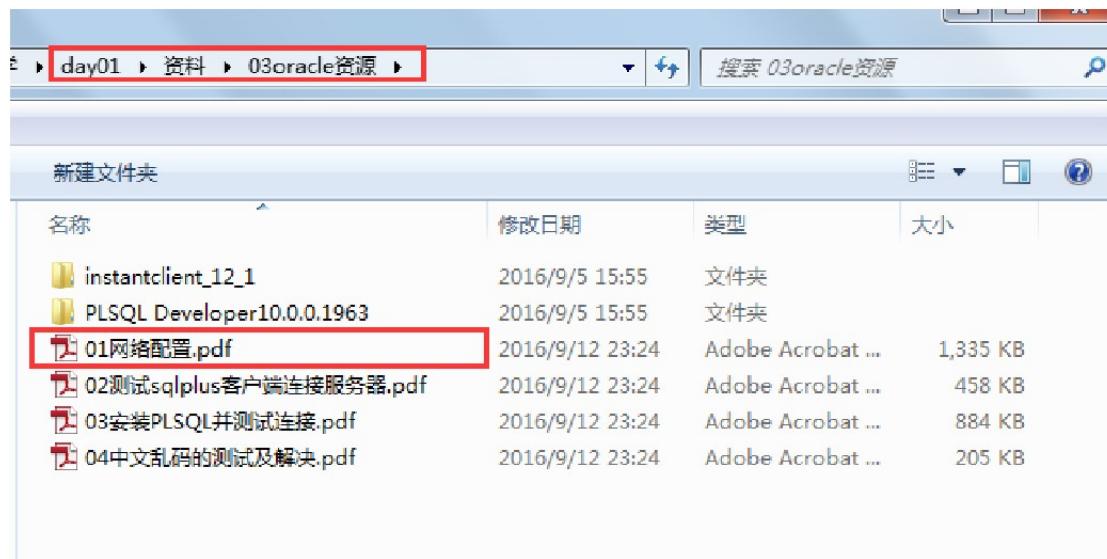
11. 安装结束，点击“退出”。





● 虚拟网卡设置

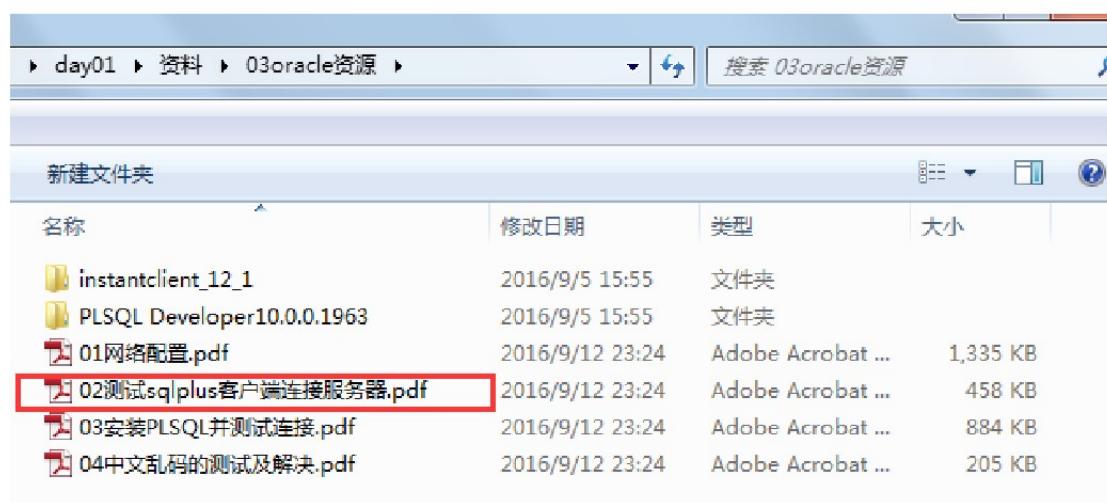
本机和虚拟机之间能相互访问，它们的IP段必须相同，但是本机将会连接不同的网络环境（比如教室、宿舍、家庭），那么本机的IP段会产生变化就连不上虚拟机了，为了避免这种情况我们让本机和虚拟机之间用虚拟网卡的方式互相通信，配置方式参考如下文档：



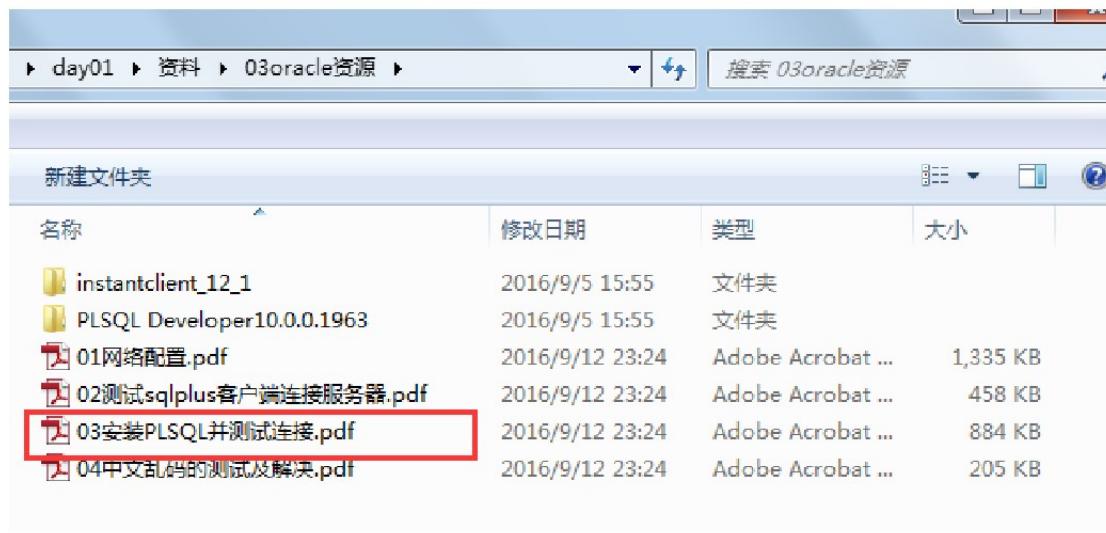
● PLSQL Developer 客户端工具的安装

1. 网络的测试

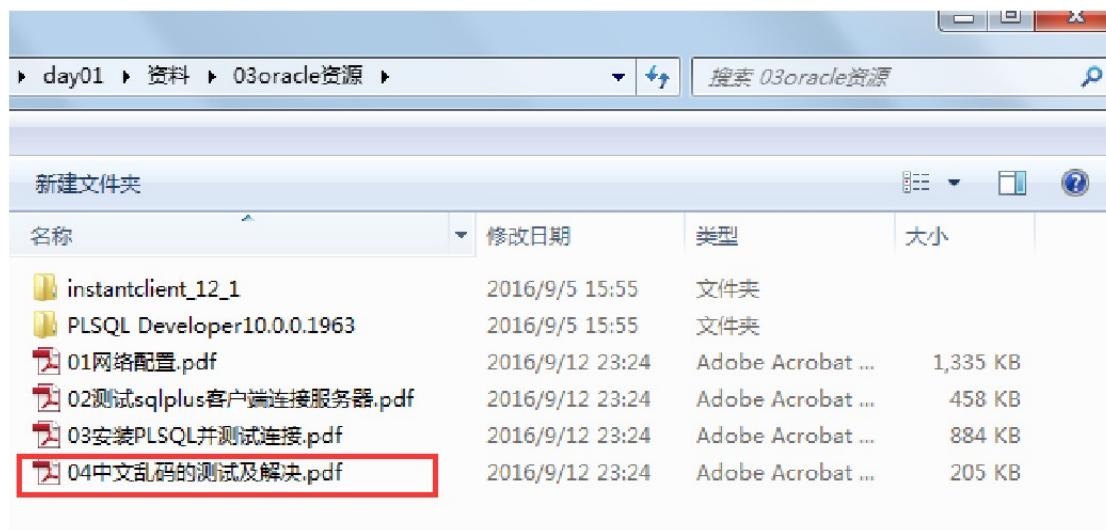
参考：



2. 安装 PLSQL Developer 客户端



3. 中文乱码的处理



● Oracle 数据库的体系结构

● 数据库: database 在硬盘中

Oracle 数据库是数据的物理存储。这就包括（数据文件 ORA 或者 DBF、控制文件、联机日志、参数文件）。其实 Oracle 数据库的概念和其它数据库不一样，这里的数据库是一个操作系统只有一个库。可以看作是 Oracle 就只有一个大数据库。

● 实例: 在内存中

一个 Oracle 实例（Oracle Instance）有一系列的后台进程（Background Processes)和内存结构（Memory Structures)组成。一个数据库可以有 n

个实例。

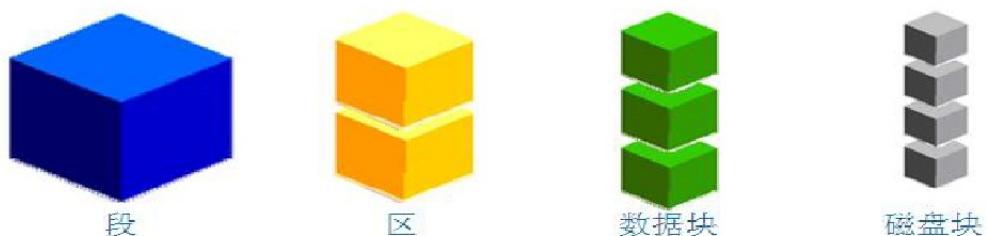
- 数据文件（dbf）：

数据文件是数据库的物理存储单位。数据库的数据是存储在表空间中的，真正是在某一个或者多个数据文件中。而一个表空间可以由一个或多个数据文件组成，一个数据文件只能属于一个表空间。一旦数据文件被加入到某个表空间后，就不能删除这个文件，**如果要删除某个数据文件，只能删除其所属于的表空间才行。**

- 表空间：

表空间是 Oracle 对物理数据库上相关数据文件（ORA 或者 DBF 文件）的逻辑映射。一个数据库在逻辑上被划分成一到若干个表空间，每个表空间包含了在逻辑上相关联的一组结构。每个数据库至少有一个表空间（称之为 system 表空间）。

每个表空间由同一磁盘上的一个或多个文件组成，这些文件叫数据文件（datafile）。**一个数据文件只能属于一个表空间。**

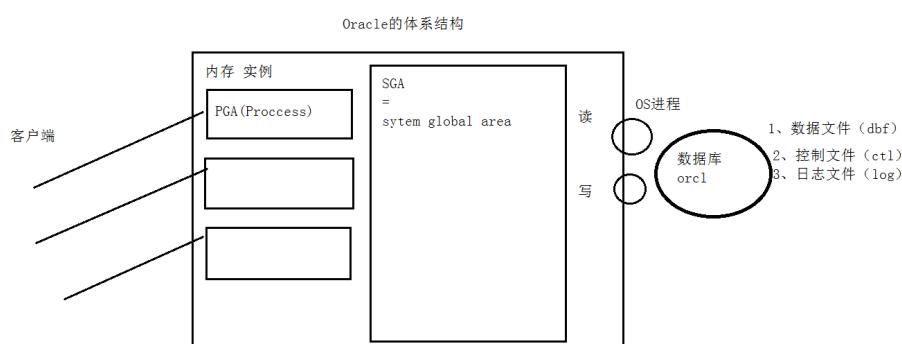


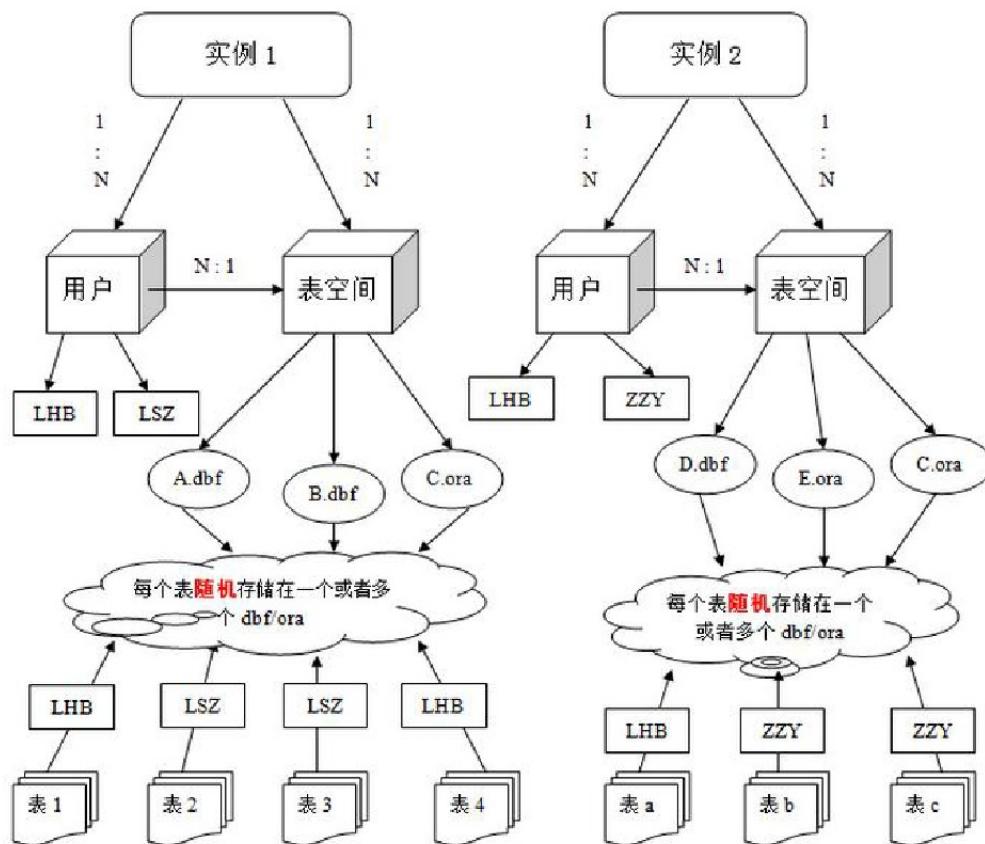
- 用户：

用户是在实例下建立的。不同实例中可以建相同名字的用户。

注： 表的数据，是有用户放入某一个表空间的，而这个表空间会随机把这些表数据放到一个或者多个数据文件中。

由于 oracle 的数据库不是普通的概念，oracle 是有用户和表空间对数据进行管理和存放的。但是表不是有表空间去查询的，而是由用户去查的。因为不同用户可以在同一个表空间建立同一个名字的表！这里区分就是用户了！

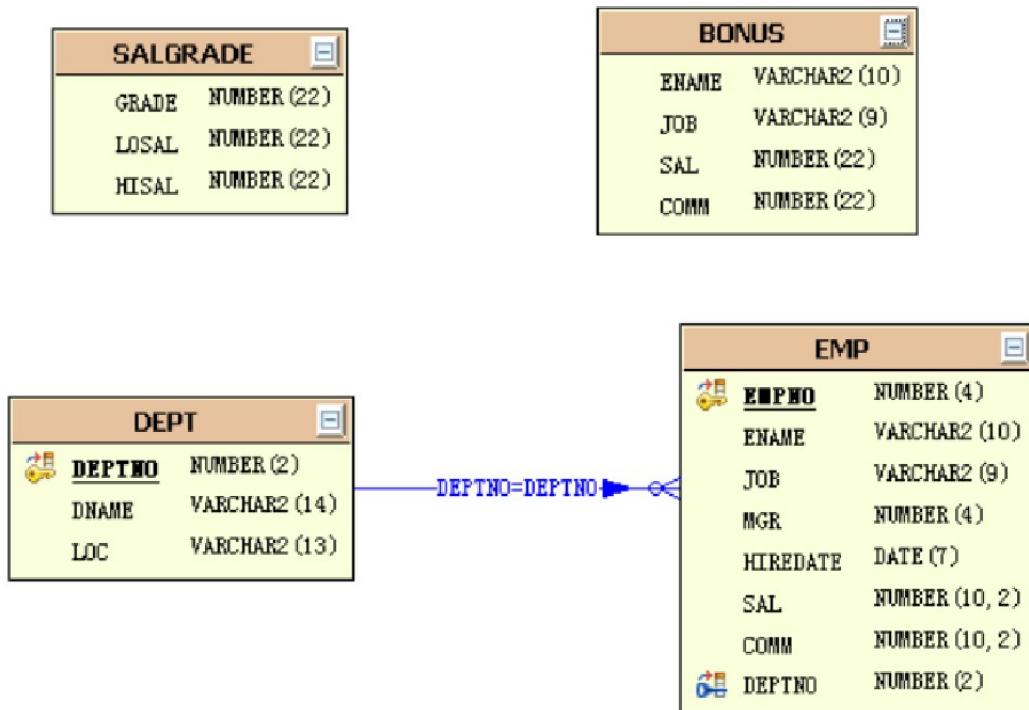




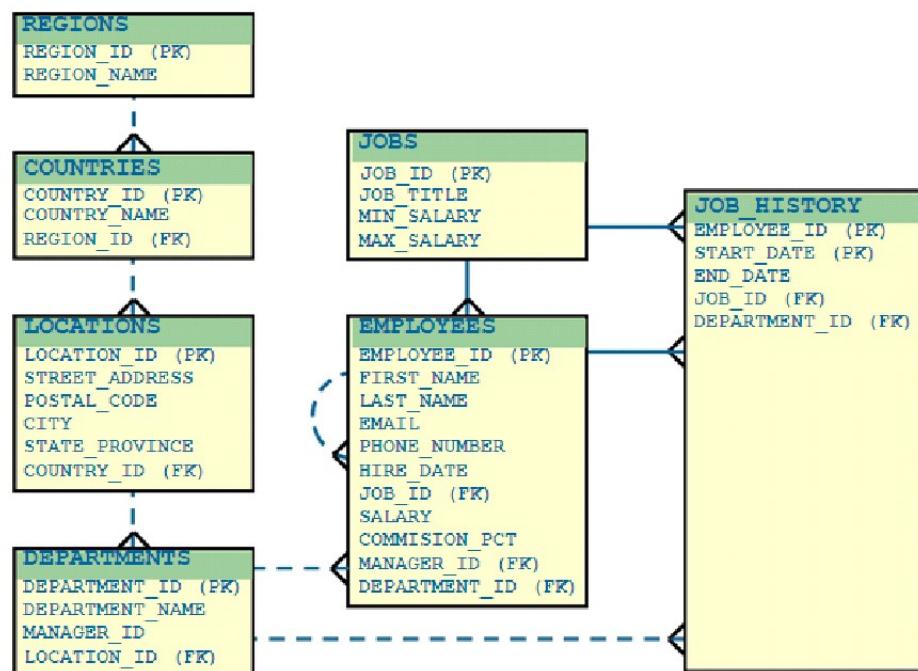
● SCOTT 用户和 HR 用户

Oracle 为了让学习者更好的进行学习，在安装成功后，也创建了初始的用户，其中 SCOTT 与 HR 就是初始的普通用户。这些用户下面都默认存在了表结构，我们重点掌握 SCOTT 用户下的所有表，如下所示：

SCOTT 用户下的表



HR 用户下的表



二、基本查询

● SQL 简介

结构化查询语言(Structured Query Language)简称 SQL(发音： /'es kju: 'el/ "S-Q-L")，结构化查询语言是一种数据库查询和程序设计语言，用于存取数据以及查询、更新和管理关系数据库系统；同时也是数据库脚本文件的扩展名。结构化查询语言是高级的非过程化编程语言，允许用户在高层数据结构上工作。它不要求用户指定对数据的存放方法，也不需要用户了解具体的数据存放方式，所以具有完全不同底层结构的不同数据库系统，可以使用相同的结构化查询语言作为数据输入与管理的接口。结构化查询语言语句可以嵌套，这使它具有极大的灵活性和强大的功能。

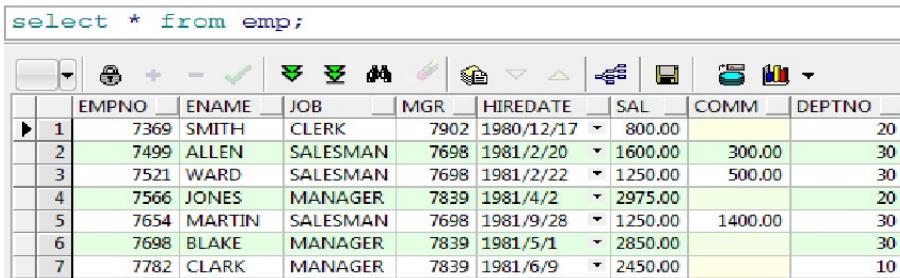
- **DML**(数据库操作语言): 其语句包括动词 `INSERT`, `UPDATE` 和 `DELETE`。它们分别用于添加，修改和删除表中的行。也称为动作查询语言。
- **DDL**(数据库定义语言): 其语句包括动词 `CREATE` 和 `DROP`。在数据库中创建新表或删除表（`CREATE TABLE` 或 `DROP TABLE`）；为表加入索引等。DDL 包括许多与人数据库目录中获得数据有关的保留字。它也是动作查询的一部分。
- **DCL**(数据库控制语言): 它的语句通过 `GRANT` 或 `REVOKE` 获得许可，确定单个用户和用户组对数据库对象的访问。某些 RDBMS 可用 `GRANT` 或 `REVOKE` 控制对表单个列的访问。

● Select 语句的语法格式和示例

```
SELECT * | { [DISTINCT] column|expression [alias], ... }  
FROM table;
```

1. 查询语法

Select * | 列名 from 表名



The screenshot shows the MySQL Workbench interface with a query editor containing the command: `select * from emp;`. Below the editor is a table with 7 rows of employee data. The columns are: EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, and DEPTNO. The data includes entries for Smith, Allen, Ward, Jones, Martin, Blake, and Clark.

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
1	7369	SMITH	CLERK	7902	1980/12/17	800.00		20
2	7499	ALLEN	SALESMAN	7698	1981/2/20	1600.00	300.00	30
3	7521	WARD	SALESMAN	7698	1981/2/22	1250.00	500.00	30
4	7566	JONES	MANAGER	7839	1981/4/2	2975.00		20
5	7654	MARTIN	SALESMAN	7698	1981/9/28	1250.00	1400.00	30
6	7698	BLAKE	MANAGER	7839	1981/5/1	2850.00		30
7	7782	CLARK	MANAGER	7839	1981/6/9	2450.00		10

```
1. show user; --当前用户  
2. select * from tab; --当前用户下的表  
3. desc emp; --员工表的结构  
4. host cls; --清屏  
5. show linesize--显示行宽  
6. set linesize 120--设置行宽  
7. col ename for a8 (a8代表8个字符的宽度) --设置列宽  
8. col sal for 9999 (9999代表4个数字的宽度)  
9. SQL优化的原则一：尽量使用列名  
10. set pagesize 20 --设置一页显示的数据条数
```

```
select empno, ename, job, mgr, hiredate, sal, comm, deptno from emp;
```

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
▶ 1	7369	SMITH	CLERK	7902	1980/12/17	800.00		20
2	7499	ALLEN	SALESMAN	7698	1981/2/20	1600.00	300.00	30
3	7521	WARD	SALESMAN	7698	1981/2/22	1250.00	500.00	30
4	7566	JONES	MANAGER	7839	1981/4/2	2975.00		20
5	7654	MARTIN	SALESMAN	7698	1981/9/28	1250.00	1400.00	30
6	7698	BLAKE	MANAGER	7839	1981/5/1	2850.00		30
7	7782	CLARK	MANAGER	7839	1981/6/9	2450.00		10
8	7788	SCOTT	ANALYST	7566	1987/4/19	3000.00		20
9	7839	KING	PRESIDENT		1981/11/17	5000.00		10
10	7844	TURNER	SALESMAN	7698	1981/9/8	1500.00	0.00	30
11	7876	ADAMS	CLERK	7788	1987/5/23	1100.00		20
12	7900	JAMES	CLERK	7698	1981/12/3	950.00		30
13	7902	FORD	ANALYST	7566	1981/12/3	3000.00		20
14	7934	MILLER	CLERK	7782	1982/1/23	1300.00		10

2. 别名用法

在查询的结果列中可以使用别名

别名不能用单引号

```
Select 列名 别名, 列名别名, ... from emp;
```

别名中，有没有双引号的区别就在于别名中有没有特殊的符号或者关键字。纯数字也不行

```
select empno 部门编号, ename 部门名称 from emp;
```

	部门编号	部门名称
▶ 1	7369	SMITH
2	7499	ALLEN
3	7521	WARD
4	7566	JONES
5	7654	MARTIN
6	7698	BLAKE
7	7782	CLARK

3. 消除重复的数据

```
Select distinct * | 列名, ... from emp;
```

```
select distinct job from emp;
```

	JOB
▶ 1	CLERK
2	SALESMAN
3	PRESIDENT
4	MANAGER
5	ANALYST

使用 `distinct` 可以消除重复的行，如果查询多列的必须保证多列都重复才能去掉重复

--`distinct`作用于后面所有的列(只要列的组合不重复即可)：
如：`select distinct deptno, job from emp;` (只要deptno和job组合不是一样的就不算是重复)

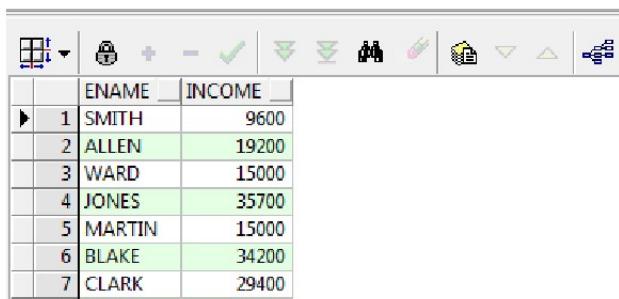
4. 查询中四则运算

查询每个雇员的年薪

```
select ename, sal*12 from emp;
```

```
select ename, sal*12 income from emp;
```

```
Select ename, sal*12 income from emp;
```



	ENAME	INCOME
1	SMITH	9600
2	ALLEN	19200
3	WARD	15000
4	JONES	35700
5	MARTIN	15000
6	BLAKE	34200
7	CLARK	29400

Sql 中支持四则运算 “+, -, *, /”

● 什么是空值？

- 空值是无效的，未指定的，未知的或不可预知的值
- 空值不是空格或者 0 。

注意： *、包含 null 的表达式都为 null null+任何值都为null

使用判空函数： nvl(a, b) 或 nvl2

*、空值永远不等于空值 (而是 is null)

补充：sql、sql Plus、i sql plus的区别

❖ 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
- 集中运行

i Sql Plus相当于sql Plus的网页版。

● 连接符 ||

字符串连接查询

Mysql 中实现方法：

```
mysql> select concat(id,'的学号',cardno,'的卡号') from card;
+-----+
| concat(id,'的学号',cardno,'的卡号') |
+-----+
| bce7009a-53bb-1034-9c6e-87f538376a72的学号 123456789的卡号 |
| bcea5c35-53bb-1034-9c6e-87f538376a72的学号 123456的卡号 |
+-----+
2 rows in set (0.00 sec)
```

查询雇员编号，姓名，工作

编号是：7369 的雇员，姓名是：smith，工作是：clerk

```
select '编号是：' || empno || '的雇员，姓名是：' || ename || '，工作是：' || job
| from emp;
```

字符串	
1	编号是：7369的雇员，姓名是：SMITH,工作是：CLERK
2	编号是：7499的雇员，姓名是：ALLEN,工作是：SALESMAN
3	编号是：7521的雇员，姓名是：WARD,工作是：SALESMAN
4	编号是：7566的雇员，姓名是：JONES,工作是：MANAGER
5	编号是：7654的雇员，姓名是：MARTIN,工作是：SALESMAN
6	编号是：7698的雇员，姓名是：BLAKE,工作是：MANAGER
7	编号是：7782的雇员，姓名是：CLARK,工作是：MANAGER
8	编号是：7788的雇员，姓名是：SCOTT,工作是：ANALYST
9	编号是：7839的雇员，姓名是：KING,工作是：PRESIDENT
10	编号是：7844的雇员，姓名是：TURNER,工作是：SALESMAN
11	编号是：7876的雇员，姓名是：ADAMS,工作是：CLERK
12	编号是：7900的雇员，姓名是：JAMES,工作是：CLERK
13	编号是：7902的雇员，姓名是：FORD,工作是：ANALYST
14	编号是：7934的雇员，姓名是：MILLER,工作是：CLERK

字符串的连接使用 ‘||’

- ❖ 字符串可以是 SELECT 列表中的一个字符,数字,日期。
- ❖ 日期和字符只能在 **单引号** 中出现。
- ❖ 每当返回一行时，字符串被输出一次。

三、条件查询和排序

● 使用 where 语句对结果进行过滤

```
SELECT * | { [DISTINCT] column|expression [alias], ... }
FROM table
[WHERE condition(s)];
```

● 比较运算符

操作符	含义
=	等于 (不是 ==)
>	大于
>=	大于、等于
<	小于
<=	小于、等于
<>	不等于 (也可以是 !=)

赋值使用 := 符号

注意：
1. 日期格式敏感
2. 字符串大小写敏感

● 其他比较运算符

如果条件是一个时间范围：
between to_date('2018-01-01', 'YyYy-Mm-Dd') and to_date
(..略);
注：时间格式：YYYY-mm-dd HH24:mi:ss, oracle是对于时间格式是不区分大小写的，注意分钟是mi即可。

操作符	含义
BETWEEN ... AND ...	在两个值之间 (包含边界) 小值在前 大值在后
IN (set)	等于值列表中的一个
LIKE	模糊查询
IS NULL	空值

● 逻辑运算符

操作符	含义
AND	逻辑并
OR	逻辑或
NOT	逻辑否

● Where语句示例

1. 非空和空的限制

1. SQL优化二：where解析顺序：右 ---> 左（所以右边的尽量能尽快地、大范围地选出满足条件的数据）
where ... and ... : false的放右边（辅助记忆：AF）
where ... or ... : true的放右边

➤ 示例：查询每月能得到奖金的雇员

分析：只要字段中存在内容表示不为空，如果不存在内容就是 null，

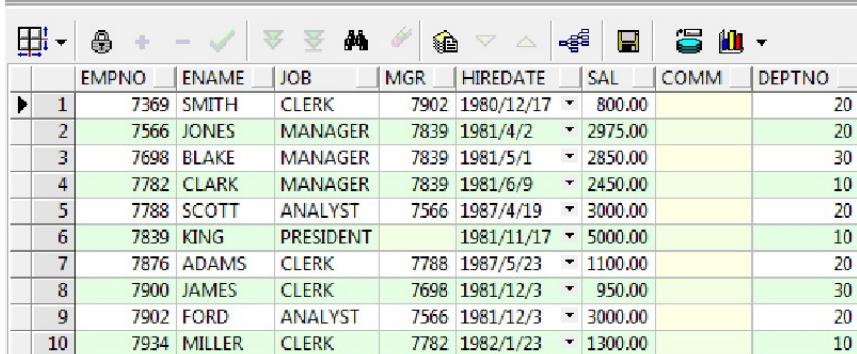
语法：列名 **IS NOT NULL**

为空 列名 **IS NULL**

```
select * from emp where comm is not null
```

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
▶ 1	7499	ALLEN	SALESMAN	7698	1981/2/20	1600.00	300.00	30
2	7521	WARD	SALESMAN	7698	1981/2/22	1250.00	500.00	30
3	7654	MARTIN	SALESMAN	7698	1981/9/28	1250.00	1400.00	30
4	7844	TURNER	SALESMAN	7698	1981/9/8	1500.00	0.00	30

```
select * from emp where comm is null
```

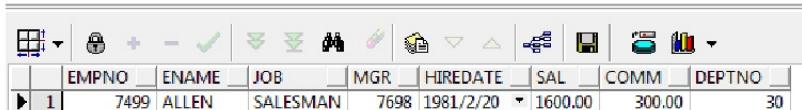


	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
▶ 1	7369	SMITH	CLERK	7902	1980/12/17	800.00		20
2	7566	JONES	MANAGER	7839	1981/4/2	2975.00		20
3	7698	BLAKE	MANAGER	7839	1981/5/1	2850.00		30
4	7782	CLARK	MANAGER	7839	1981/6/9	2450.00		10
5	7788	SCOTT	ANALYST	7566	1987/4/19	3000.00		20
6	7839	KING	PRESIDENT		1981/11/17	5000.00		10
7	7876	ADAMS	CLERK	7788	1987/5/23	1100.00		20
8	7900	JAMES	CLERK	7698	1981/12/3	950.00		30
9	7902	FORD	ANALYST	7566	1981/12/3	3000.00		20
▶ 10	7934	MILLER	CLERK	7782	1982/1/23	1300.00		10

- 范例：查询工资大于 1500 并且有奖金领取的雇员

分析：多个查询条件同时满足之间使用 ‘AND’

```
select * from emp where comm is not null and sal > 1500
```

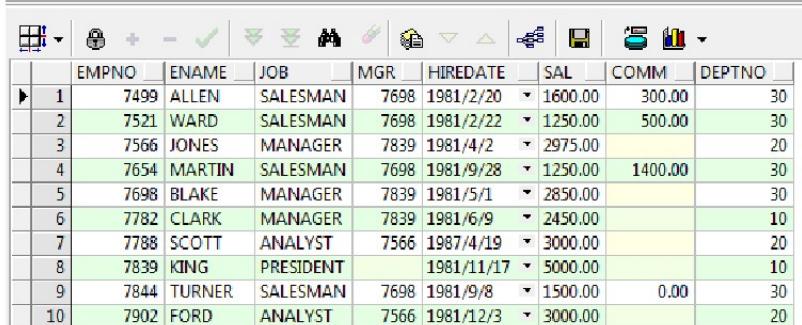


	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
▶ 1	7499	ALLEN	SALESMAN	7698	1981/2/20	1600.00	300.00	30

- 范例：查询工资大于 1500 或者有奖金的雇员

分析：多个查询条件或满足，条件之间使用 “OR”

```
select * from emp where comm is not null or sal > 1500
```

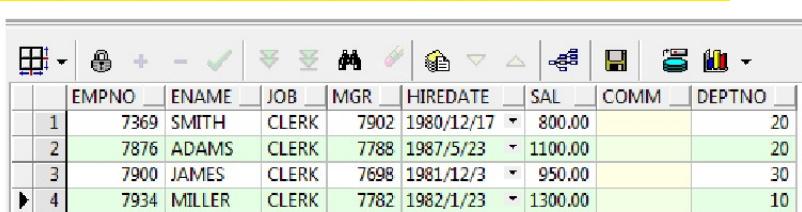


	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
▶ 1	7499	ALLEN	SALESMAN	7698	1981/2/20	1600.00	300.00	30
2	7521	WARD	SALESMAN	7698	1981/2/22	1250.00	500.00	30
3	7566	JONES	MANAGER	7839	1981/4/2	2975.00		20
4	7654	MARTIN	SALESMAN	7698	1981/9/28	1250.00	1400.00	30
5	7698	BLAKE	MANAGER	7839	1981/5/1	2850.00		30
6	7782	CLARK	MANAGER	7839	1981/6/9	2450.00		10
7	7788	SCOTT	ANALYST	7566	1987/4/19	3000.00		20
8	7839	KING	PRESIDENT		1981/11/17	5000.00		10
9	7844	TURNER	SALESMAN	7698	1981/9/8	1500.00	0.00	30
▶ 10	7902	FORD	ANALYST	7566	1981/12/3	3000.00		20

- 范例：查询工资不大于 1500 和没有奖金的人

语法：NOT(查询条件)

```
select * from emp where comm is null and not(sal>1500);
```



	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
1	7369	SMITH	CLERK	7902	1980/12/17	800.00		20
2	7876	ADAMS	CLERK	7788	1987/5/23	1100.00		20

2. 范围限制

范例：基本工资大于 1500 但是小于 3000 的全部雇员

分析：sal>1500, sal<3000

```
select * from emp where sal > 1500 and sal < 3000
```

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
▶ 1	7499	ALLEN	SALESMAN	7698	1981/2/20	1600.00	300.00	30
2	7566	JONES	MANAGER	7839	1981/4/2	2975.00		20
3	7698	BLAKE	MANAGER	7839	1981/5/1	2850.00		30
4	7782	CLARK	MANAGER	7839	1981/6/9	2450.00		10

Between and 等于 $\text{sal} \geq 1500 \text{ and } \text{sal} \leq 3000$

```
select * from emp where sal between 1500 and 3000
```

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
▶ 1	7499	ALLEN	SALESMAN	7698	1981/2/20	1600.00	300.00	30
2	7566	JONES	MANAGER	7839	1981/4/2	2975.00		20
3	7698	BLAKE	MANAGER	7839	1981/5/1	2850.00		30
4	7782	CLARK	MANAGER	7839	1981/6/9	2450.00		10
5	7788	SCOTT	ANALYST	7566	1987/4/19	3000.00		20
6	7844	TURNER	SALESMAN	7698	1981/9/8	1500.00	0.00	30
7	7902	FORD	ANALYST	7566	1981/12/3	3000.00		20

范例：查询 1981-1-1 到 1981-12-31 号入职的雇员

分析：between and 不仅可以使用在数值之间，也可以用在日期的区间

```
select * from emp where hiredate between '1-1月-1981' and '31-12月-1981';
```

还是那句话：单引号中可以是字符，数字，时间

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
▶ 1	7499	ALLEN	SALESMAN	7698	1981/2/20	1600.00	300.00	30
2	7521	WARD	SALESMAN	7698	1981/2/22	1250.00	500.00	30
3	7566	JONES	MANAGER	7839	1981/4/2	2975.00		20
4	7654	MARTIN	SALESMAN	7698	1981/9/28	1250.00	1400.00	30
5	7698	BLAKE	MANAGER	7839	1981/5/1	2850.00		30
6	7782	CLARK	MANAGER	7839	1981/6/9	2450.00		10
7	7839	KING	PRESIDENT		1981/11/17	5000.00		10
8	7844	TURNER	SALESMAN	7698	1981/9/8	1500.00	0.00	30
9	7900	JAMES	CLERK	7698	1981/12/3	950.00		30
10	7902	FORD	ANALYST	7566	1981/12/3	3000.00		20

范例：查询雇员名字叫 smith 的雇员

在 oracle 中的查询条件中查询条件的值是区分大小写的

```
select * from emp where ename = 'smith'
```

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
▶ 1	7499	ALLEN	SALESMAN	7698	1981/2/20	1600.00	300.00	30

```
select * from emp where ename = 'SMITH'
```

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
▶ 1	7369	SMITH	CLERK	7902	1980/12/17	800.00		20

范例：查询雇员编号是 7369,7499,7521 的雇员编号的具体信息

如果使用之前的做法可以使用 OR 关键字



```
select * from emp where empno = 7369 or empno = 7499 or empno = 7521
```

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
▶	1	7369	SMITH	CLERK	7902 1980/12/17	800.00		20
▶	2	7499	ALLEN	SALESMAN	7698 1981/2/20	1600.00	300.00	30
▶	3	7521	WARD	SALESMAN	7698 1981/2/22	1250.00	500.00	30

实际上，此时指定了查询范围，那么 sql 可以使用 **IN** 关键字

语法: 列名 IN (值 1, 值 2, ...)

列名 NOT IN (值 1, 值 2, ...)

其中的值不仅可以是数值类型也可以是字符串

```
select * from emp where empno in (7369,7499,7521)
```

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
▶	1	7369	SMITH	CLERK	7902 1980/12/17	800.00		20
▶	2	7499	ALLEN	SALESMAN	7698 1981/2/20	1600.00	300.00	30
▶	3	7521	WARD	SALESMAN	7698 1981/2/22	1250.00	500.00	30

范例: 查询雇员姓名是'SMITH','ALLEN','WARD'的雇员具体信息

```
select * from emp where ename in ('SMITH','ALLEN','WARD')
```

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
▶	1	7369	SMITH	CLERK	7902 1980/12/17	800.00		20
▶	2	7499	ALLEN	SALESMAN	7698 1981/2/20	1600.00	300.00	30
▶	3	7521	WARD	SALESMAN	7698 1981/2/22	1250.00	500.00	30

3. 模糊查询

在常用的站点中经常会有模糊查询，即：输入一个关键字，把符合的内容全部的查询出来，在 sql 中使用 **LIKE** 语句完成。

在 **LIKE** 中主要使用以下两种通配符

“%”：可以匹配任意长度的内容

“_”：可以匹配一个长度的内容

范例：查询出所有雇员姓名中第二个字符包含“M”的雇员

```
select * from emp where ename like '_M%'
```

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
▶	1	7369	SMITH	CLERK	7902 1980/12/17	800.00		20

在 **LIKE** 中如果没有关键字表示查询全部

1. 模糊查询细节：

--查询名字中含有下划线的员工

这里需要使用 **escape** 关键字定义转义符（可以是任意字符），当转义符置于通配符之前时，该通配符就解释为普通字符

修改后：where ename like '%_%' escape '\';

```
select * from emp where ename like '%s'
```

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
1	7369	SMITH	CLERK	7902	1980/12/17	800.00		20
2	7499	ALLEN	SALESMAN	7698	1981/2/20	1600.00	300.00	30
3	7521	WARD	SALESMAN	7698	1981/2/22	1250.00	500.00	30
4	7566	JONES	MANAGER	7839	1981/4/2	2975.00		20
5	7654	MARTIN	SALESMAN	7698	1981/9/28	1250.00	1400.00	30
6	7698	BLAKE	MANAGER	7839	1981/5/1	2850.00		30
7	7782	CLARK	MANAGER	7839	1981/6/9	2450.00		10
8	7788	SCOTT	ANALYST	7566	1987/4/19	3000.00		20
9	7839	KING	PRESIDENT		1981/11/17	5000.00		10
10	7844	TURNER	SALESMAN	7698	1981/9/8	1500.00	0.00	30
11	7876	ADAMS	CLERK	7788	1987/5/23	1100.00		20
12	7900	JAMES	CLERK	7698	1981/12/3	950.00		30
13	7902	FORD	ANALYST	7566	1981/12/3	3000.00		20
14	7934	MILLER	CLERK	7782	1982/1/23	1300.00		10

查询名字中带有“M”的雇员

```
select * from emp where ename like '%M%'
```

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
1	7369	SMITH	CLERK	7902	1980/12/17	800.00		20
2	7654	MARTIN	SALESMAN	7698	1981/9/28	1250.00	1400.00	30
3	7876	ADAMS	CLERK	7788	1987/5/23	1100.00		20
4	7900	JAMES	CLERK	7698	1981/12/3	950.00		30
5	7934	MILLER	CLERK	7782	1982/1/23	1300.00		10

在 oracle 中不等号的用法可以有两种形式“<>”和“!=”

范例：查询雇员编号不是 7369 的雇员信息

```
select * from emp where empno <> 7369;
```

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
1	7499	ALLEN	SALESMAN	7698	1981/2/20	1600.00	300.00	30
2	7521	WARD	SALESMAN	7698	1981/2/22	1250.00	500.00	30
3	7566	JONES	MANAGER	7839	1981/4/2	2975.00		20
4	7654	MARTIN	SALESMAN	7698	1981/9/28	1250.00	1400.00	30
5	7698	BLAKE	MANAGER	7839	1981/5/1	2850.00		30
6	7782	CLARK	MANAGER	7839	1981/6/9	2450.00		10
7	7788	SCOTT	ANALYST	7566	1987/4/19	3000.00		20

```
select * from emp where empno != 7369;
```

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
1	7499	ALLEN	SALESMAN	7698	1981/2/20	1600.00	300.00	30
2	7521	WARD	SALESMAN	7698	1981/2/22	1250.00	500.00	30
3	7566	JONES	MANAGER	7839	1981/4/2	2975.00		20
4	7654	MARTIN	SALESMAN	7698	1981/9/28	1250.00	1400.00	30
5	7698	BLAKE	MANAGER	7839	1981/5/1	2850.00		30
6	7782	CLARK	MANAGER	7839	1981/6/9	2450.00		10
7	7788	SCOTT	ANALYST	7566	1987/4/19	3000.00		20
8	7839	KING	PRESIDENT		1981/11/17	5000.00		10



● 使用 `order by` 对结果排序

1. 排序的语法

在 SQL 中可以使用 ORDER BY 对查询结果进行排序

语法：SELECT * | 列名 FROM 表名 {WHERE 查询条件} ORDER BY 列名 1 ASC|DESC, 列名 2...ASC|DESC

范例：查询雇员的工资从低到高

分析：ORDER BY 列名 默认的排序规则是升序排列，可以不指定 ASC，如果按着降序排列必须指定 DESC

```
select * from emp order by sal
```

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
▶	1	7369	SMITH	CLERK	7902 1980/12/17	800.00		20
	2	7900	JAMES	CLERK	7698 1981/12/3	950.00		30
	3	7876	ADAMS	CLERK	7788 1987/5/23	1100.00		20
	4	7521	WARD	SALESMAN	7698 1981/2/22	1250.00	500.00	30
	5	7654	MARTIN	SALESMAN	7698 1981/9/28	1250.00	1400.00	30
	6	7934	MILLER	CLERK	7782 1982/1/23	1300.00		10
	7	7844	TURNER	SALESMAN	7698 1981/9/8	1500.00	0.00	30
	8	7499	ALLEN	SALESMAN	7698 1981/2/20	1600.00	300.00	30

1.--多个列的排序：

--1.order by 作用于后面所有的列，先按照第一个列排序，再依次按后面的列排序；
--2.多个排序字段用逗号隔开，desc/asc只作用所描述的列



```
select * from emp order by sal desc
```

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
▶	1	7839	KING	PRESIDENT	1981/11/17	5000.00		1
	2	7902	FORD	ANALYST	7566 1981/12/3	3000.00		2
	3	7788	SCOTT	ANALYST	7566 1987/4/19	3000.00		2
	4	7566	JONES	MANAGER	7839 1981/4/2	2975.00		2
	5	7698	BLAKE	MANAGER	7839 1981/5/1	2850.00		3
	6	7782	CLARK	MANAGER	7839 1981/6/9	2450.00		1
	7	7499	ALLEN	SALESMAN	7698 1981/2/20	1600.00	300.00	3
	8	7844	TURNER	SALESMAN	7698 1981/9/8	1500.00	0.00	3

如果存在多个排序字段可以用逗号分隔

```
select * from emp order by sal asc, hiredate desc
```

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
▶	1	7369	SMITH	CLERK	7902 1980/12/17	800.00		20
	2	7900	JAMES	CLERK	7698 1981/12/3	950.00		30
	3	7876	ADAMS	CLERK	7788 1987/5/23	1100.00		20
	4	7654	MARTIN	SALESMAN	7698 1981/9/28	1250.00	1400.00	30
	5	7521	WARD	SALESMAN	7698 1981/2/22	1250.00	500.00	30

注意 ORDER BY 语句要放在 sql 的最后执行。

2. 排序中的空值问题

当排序时有可能存在 null 时就会产生问题，我们可以用 `nulls first`, `nulls last` 来指定 null 值显示的位置。

--查询雇员的工资从低到高

```
select * from emp order by sal nulls first;
```

```
select * from emp order by sal desc nulls last ;
```



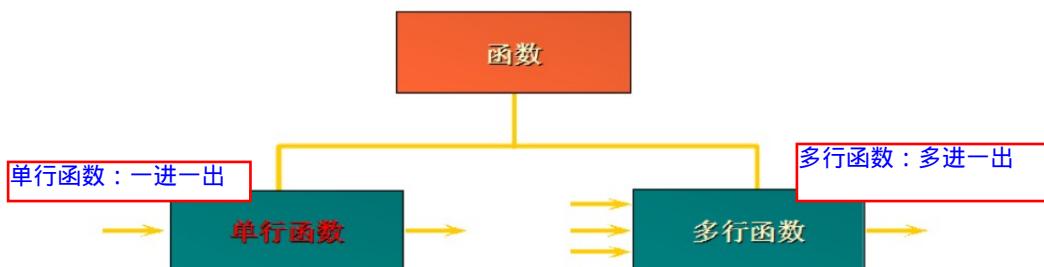
四、单行函数

● 什么是 SQL 的函数？

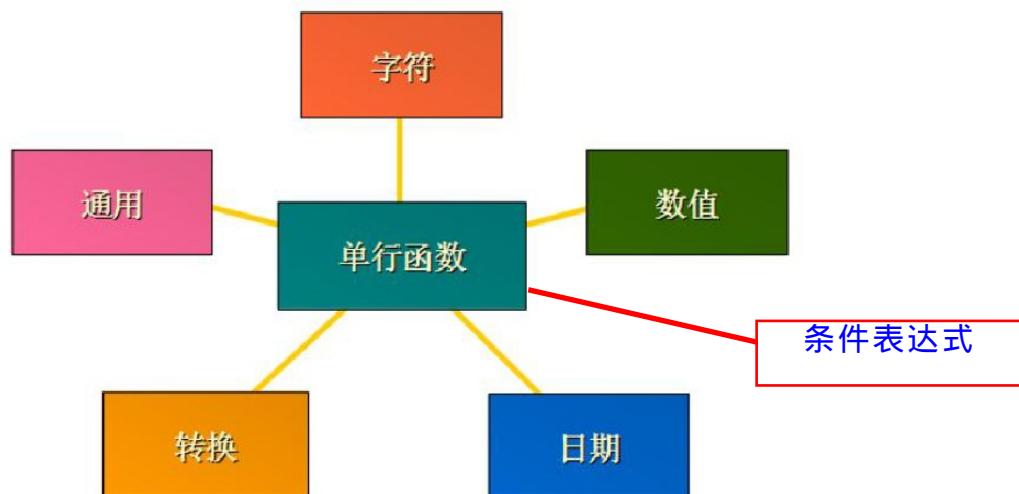


注意：函数可以没有参数，但必须要有返回值

● 函数的类型

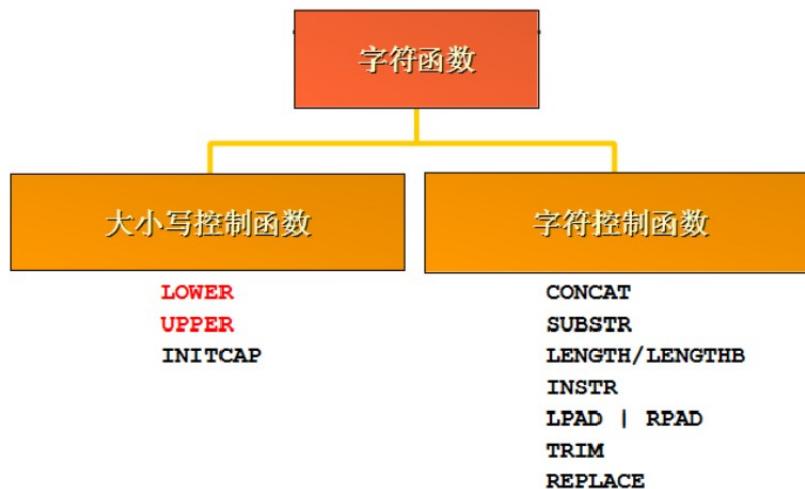


● 单行函数





✓ 字符函数



示例：

接收字符输入返回字符或者数值，**dual** 是伪表

1. 把小写的字符转换成大写的字符

```
upper('smith')
```

```
select upper('smith') from dual
```

2. 把大写字符变成小写字母

```
lower('SMITH')
```

```
select lower('SMITH') from dual
```

3. 把首字符大写

```
initcap('smith')
```

```
select initcap('smith') from dual
```

4. 字符串的连接可以使用 concat 可以使用 “||” 建议使用 “||”

```
concat('hello', 'world')
```

```
select concat('hello', 'world') from dual
```

CONCAT('HELLO','WORLD')											
1 helloworld											

5. 字符串的截取，使用 `substr`，第一个参数是源字符串，第二个参数是开始索引，第三个参数长度，开始的索引使用 1 和 0 效果相同

索引是从1开始。

```
substr('hello', 1,3)
```

```
select substr('hello', 0,3) from dual;
```

Select dual	Select dual	Select dual
SUBSTR('HELLO',0,3)		

6. 获取字符串的长度

`length('hello')` --length 字符数 lengthb 字节数 (一个汉字=一个字符=两个字节)

```
select length('hello') from dual;
```

Select dual	Select dual	Select dual
LENGTH('HELLO')		

7. 字符串替换，第一个参数是源字符串，第二个参数被替换的字符串，第三个是替换字符串

```
replace('hello', 'l','x')
```

```
select replace('hello', 'l','x') from dual;
```

Select dual	Select dual	Select dual
REPLACE('HELLO','L','X')		

✓ 数值函数

❖ ROUND: 四舍五入

`ROUND(45.926, 2)` 45.93

❖ TRUNC: 截断

`TRUNC(45.926, 2)` 45.92

❖ MOD: 求余

`MOD(1600, 300)` 100

✓ 日期函数

■ Oracle 中的日期：

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 。

■ 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

```
select (sysdate-1) 昨天, sysdate 今天, (sysdate+1) 明天  
from dual;
```

■ 日期函数

函数	描述
MONTHS_BETWEEN	两个日期相差的月数
ADD_MONTHS	向指定日期中加上若干月数
NEXT_DAY	指定日期的下一个日期
LAST_DAY	本月的最后一天
ROUND	日期四舍五入
TRUNC	日期截断

months_between(大的日期 , 小的日期)

next_day 的作用举例：每星期一自动备份数据库。

■ 日期的四舍五入

❖ 假设 SYSDATE = '25-JUL-95' :

ROUND(SYSDATE, 'MONTH') → 01-AUG-95

ROUND(SYSDATE, 'YEAR') → 01-JAN-96

TRUNC(SYSDATE, 'MONTH') → 01-JUL-95

TRUNC(SYSDATE, 'YEAR') → 01-JAN-95

■ 日期函数示例

1. 范例：查询雇员的进入公司的周数。

分析：查询雇员进入公司的天数(sysdate - 入职日期)/7 就是周数

```
select ename, round((sysdate - hiredate)/7) from emp
```

	ENAME	ROUND((SYSDATE-HIREDATE)/7)
1	SMITH	1772
2	ALLEN	1762
3	WARD	1762
4	JONES	1756
5	MARTIN	1731
6	BLAKE	1752
7	CLARK	1747
8	SCOTT	1441
9	KING	1724
10	TURNER	1734
11	ADAMS	1436

2. 获得两个时间段中的月数: MONTHS_BETWEEN()

范例：查询所有雇员进入公司的月数

```
select ename, round(months_between(sysdate, hiredate)) from emp
```

	ENAME	ROUND(MONTHS_BETWEEN(SYSDATE,HIREDATE))
1	SMITH	407
2	ALLEN	405
3	WARD	405
4	JONES	404
5	MARTIN	398
6	BLAKE	403
7	CLARK	402
8	SCOTT	331
9	KING	396
10	TURNER	399
11	ADAMS	330
12	JAMES	396
13	FORD	396
14	MILLER	394

3. 获得几个月后的日期: ADD_MONTHS()

范例：求出三个月后的日期

```
select add_months(sysdate, 3) from dual
```

	ADD_MONTHS(SYSDATE,3)
1	2015/2/28 16:02:30

✓ 转换函数

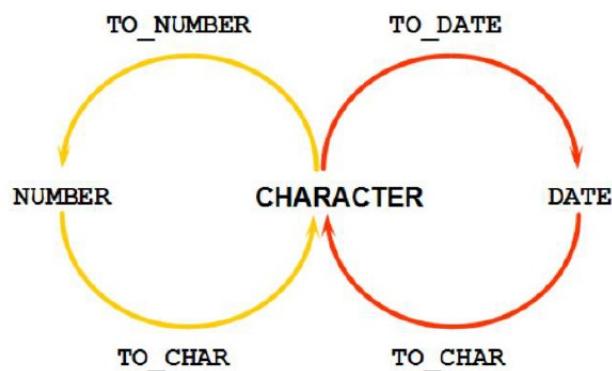


■ 隐式数据类型转换

Oracle 自动完成数据类型的转换

源数据类型	目标数据类型
VARCHAR2 or CHAR	NUMBER
VARCHAR2 or CHAR	DATE
NUMBER	VARCHAR2
DATE	VARCHAR2

■ 显式数据类型转换



■ TO_CHAR 函数对日期的转换

TO_CHAR(date, 'format_model')

日期的格式:

大小写无所谓

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 函数对数字的转换

TO_CHAR(number, 'format_model')

数字转换的格式：

9	数字
0	零
\$	美元符
L	本地货币符号
.	小数点
,	千位符

■ TO_NUMBER 和 TO_DATE 函数

❖ 使用 **TO_NUMBER** 函数将字符转换成数字**TO_NUMBER(char[, 'format_model'])**❖ 使用 **TO_DATE** 函数将字符转换成日期**TO_DATE(char[, 'format_model'])**

■ 示例：

1. TO_CHAR:字符串转换函数

范例：查询所有的雇员将将年月日分开，此时可以使用 TO_CHAR 函数来拆分
拆分时需要使用通配符

年：y, 年是四位使用 yyyy

月：m, 月是两位使用 mm

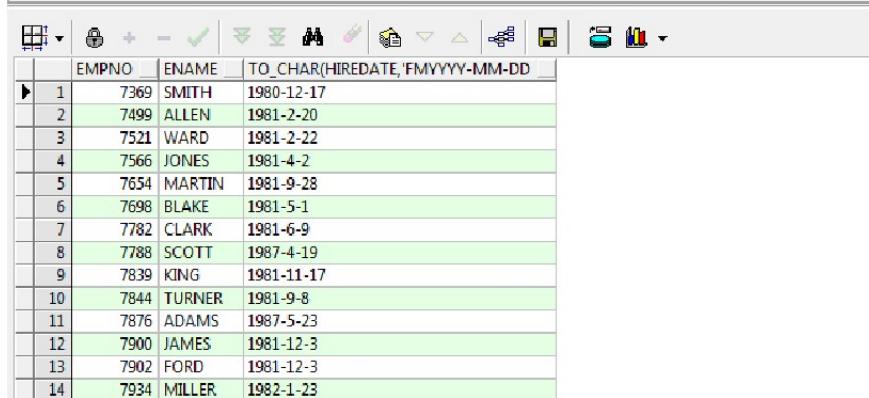
日：d, 日是两位使用 dd

```
select empno, ename, to_char(hiredate, 'yyyy-mm-dd') from emp;
```

	EMPNO	ENAME	TO_CHAR(HIREDATE, 'YYYY-MM-DD')
1	7369	SMITH	1980-12-17
2	7499	ALLEN	1981-02-20
3	7521	WARD	1981-02-22
4	7566	JONES	1981-04-02
5	7654	MARTIN	1981-09-28
6	7698	BLAKE	1981-05-01
7	7782	CLARK	1981-06-09
8	7788	SCOTT	1987-04-19
9	7839	KING	1981-11-17
10	7844	TURNER	1981-09-08

在结果中 10 以下的月前面被被补了前导零，可以使用 fm 去掉前导零

```
select empno, ename, to_char(hiredate, 'fmYYYY-mm-dd') from emp;
```

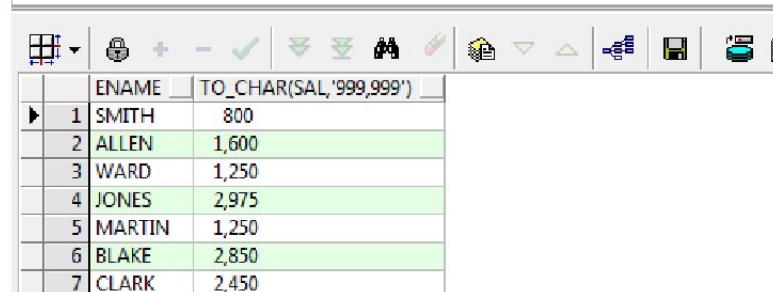


	EMPNO	ENAME	TO_CHAR(HIREDATE, 'FMYYYY-MM-DD')
▶	1	7369	SMITH 1980-12-17
	2	7499	ALLEN 1981-2-20
	3	7521	WARD 1981-2-22
	4	7566	JONES 1981-4-2
	5	7654	MARTIN 1981-9-28
	6	7698	BLAKE 1981-5-1
	7	7782	CLARK 1981-6-9
	8	7788	SCOTT 1987-4-19
	9	7839	KING 1981-11-17
	10	7844	TURNER 1981-9-8
	11	7876	ADAMS 1987-5-23
	12	7900	JAMES 1981-12-3
	13	7902	FORD 1981-12-3
	14	7934	MILLER 1982-1-23

TO_CHAR 还可以给数字做格式化

范例：把雇员的工资按三位用“,”分隔，在 oracle 中“9”代表一位数字

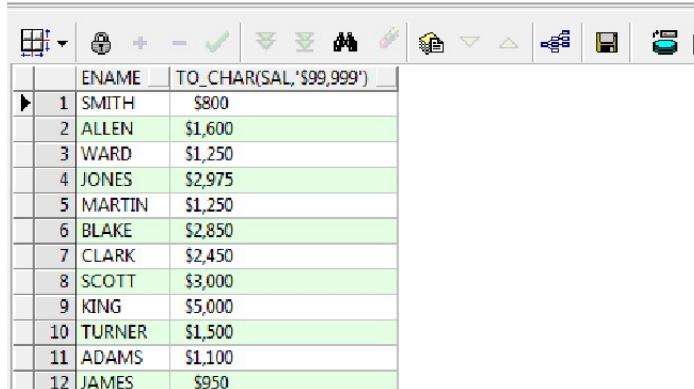
```
select ename, to_char(sal, '99,999') from emp;
```



	ENAME	TO_CHAR(SAL, '99,999')
▶	1	SMITH 800
	2	ALLEN 1,600
	3	WARD 1,250
	4	JONES 2,975
	5	MARTIN 1,250
	6	BLAKE 2,850
	7	CLARK 2,450

如果在钱的前面加上国家的符号可以使用“\$”代表是美元，如果要使用本地的钱的单位使用“L”

```
select ename, to_char(sal, '$99,999') from emp
```



	ENAME	TO_CHAR(SAL, '\$99,999')
▶	1	SMITH \$800
	2	ALLEN \$1,600
	3	WARD \$1,250
	4	JONES \$2,975
	5	MARTIN \$1,250
	6	BLAKE \$2,850
	7	CLARK \$2,450
	8	SCOTT \$3,000
	9	KING \$5,000
	10	TURNER \$1,500
	11	ADAMS \$1,100
	12	JAMES \$950

select ename, to_char(sal, 'L99,999') from emp;	
	ENAME TO_CHAR(SAL,'L99,999')
1	SMITH ¥800
2	ALLEN ¥1,600
3	WARD ¥1,250
4	JONES ¥2,975
5	MARTIN ¥1,250
6	BLAKE ¥2,850
7	CLARK ¥2,450
8	SCOTT ¥3,000
9	KING ¥5,000
10	TURNER ¥1,500
11	ADAMS ¥1,100
12	JAMES ¥950
13	FORD ¥3,000
14	MILLER ¥1,300

2. TO_NUMBER:数值转换函数

TO_NUMBER 可以把字符串转换成数值

```
select to_number('10')+to_number('10') from dual
```

TO_NUMBER('10')+TO_NUMBER('10')	
1	20

3. TO_DATE:日期转换函数

TO_DATE 可以把字符串的数据转换成日期类型

```
select to_date('1985-04-22', 'yyyy-mm-dd') from dual
```

TO_DATE('1985-04-22','YYYY-MM-DD')	
1	1985/4/22

✓ 通用函数

■ 什么是通用函数?

这些函数适用于任何数据类型，同时也适用于空值

■ 常用的通用函数

- **NVL (expr1, expr2)**
- **NVL2 (expr1, expr2, expr3)**
- **NULLIF (expr1, expr2)**
- **COALESCE (expr1, expr2, ..., exprn)**

■ 通用函数示例

1. 空值处理 nvl

范例：查询所有的雇员的年薪

--nvl2(a,b,c) 当a=null的时候，返回c；否则返回b
--nullif(a,b) 当a=b的时候，返回null；否则返回a
--coalesce 从左到右 找到第一个不为null的值

	ENAME	SAL*12+COMM
1	SMITH	
2	ALLEN	19500
3	WARD	15500
4	JONES	
5	MARTIN	16400
6	BLAKE	
7	CLARK	
8	SCOTT	
9	KING	
10	TURNER	18000
11	ADAMS	
12	JAMES	
13	FORD	
14	MILLER	

我们发现很多员工的年薪是空的，原因是很多员工的奖金是 null，null 和任何数值计算都是 null，这时我们可以使用 nvl 来处理。

```
select ename, nvl(comm,0), sal*12+ nvl(comm, 0) from emp;
```

	ENAME	NVL(COMM,0)	SAL*12+NVL(COMM,0)
1	SMITH	0	9600
2	ALLEN	300	19500
3	WARD	500	15500
4	JONES	0	35700
5	MARTIN	1400	16400
6	BLAKE	0	34200
7	CLARK	0	29400
8	SCOTT	0	36000
9	KING	0	60000
10	TURNER	0	18000
11	ADAMS	0	13200
12	JAMES	0	11400
13	FORD	0	36000
14	MILLER	0	15600

✓ 条件表达式

非常重要！！！可以简化在java代码上的逻辑判断。并且效率更高。

■ 什么是条件表达式？

在 SQL 语句中使用 IF-THEN-ELSE

■ 实现的方式：

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

```
CASE expr WHEN comparison_expr1 THEN return_expr1
            [WHEN comparison_expr2 THEN return_expr2
            WHEN comparison_exprn THEN return_exprn
            ELSE else_expr]
```

END

中括号中的代表可以省略。

```
SQL> select ename, job, sal 涨前,
2      case job when 'PRESIDENT' then sal+1000
3          when 'MANAGER' then sal+800
4          else sal+400
5      end 涨后
6  from emp;
```

■ DECODE 函数

```
DECODE(col|expression, search1, result1
      [, search2, result2,...,]
      [, default])
```

```
SQL> select ename, job, sal 涨前,
2      decode(job, 'PRESIDENT', sal+1000,
3              'MANAGER', sal+800,
4              sal+400) 涨后
5  from emp;
```

- 条件表达式示例：根据 10 号部门员工的工资，显示税率

```
SELECT ename, sal,
       DECODE (TRUNC(sal/2000, 0),
                0, 0.00,
                1, 0.09,
                2, 0.20,
                3, 0.30,
                4, 0.40,
                5, 0.42,
                6, 0.44,
                0.45) TAX_RATE
  FROM emp
 WHERE deptno = 10;
```

五、多行函数

- 什么是多行函数？

分组函数作用于一组数据，并对一组数据返回一个值。

也叫：组函数、分组函数

组函数会忽略空值；**NVL** 函数使分组函数无法忽略空值

- 常用的多行函数

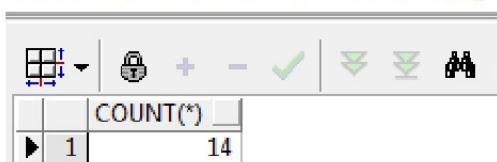
- ❖ **AVG**
- ❖ **COUNT**
- ❖ **MAX**
- ❖ **MIN**
- ❖ **SUM**

- 多行函数示例

1. 统计记录数 **count()**

范例：查询出所有员工的记录数

```
select count(*) from emp
```



	COUNT(*)
▶ 1	14

不建议使用 **count(*)**，可以使用一个具体的列以免影响性能。



select count(ename) from emp	
COUNT(ENAME)	14

2.最小值查询 min()

范例：查询出来员工最低工资

select min(sal) from emp	
MIN(SAL)	800

3.最大值查询 max()

范例：查询出员工的最高工资

select max(sal) from emp	
MAX(SAL)	5000

4.查询平均值 avg()

范例：查询出员工的平均工资

select avg(sal) from emp	
AVG(SAL)	2073.21428571429

5.求和函数 sum()

范例：查询出 20 号部门的员工的工资总和



select sum(sal) from emp t where t.deptno = 20

	SUM(SAL)
1	10875

● 分组数据

```
SELECT      column, group_function(column)
FROM        table
[WHERE      condition]
[GROUP BY   group_by_expression]
[ORDER BY   column];
```

可以使用**GROUP BY** 子句将表中的数据分成若干组

范例：查询每个部门的人数

```
select deptno, count(ename) from emp group by deptno
```

	DEPTNO	COUNT(ENAME)
1	30	6
2	20	5
3	10	3

范例：查询出每个部门的平均工资

```
select deptno, avg(sal) from emp group by deptno
```

	DEPTNO	AVG(SAL)
1	30	1566.666666666667
2	20	2175
3	10	2916.666666666667

范例：查询出来部门编号，和部门下的人数

```
select deptno, count(ename) from emp
```



The screenshot shows a SQL query in the editor and an 'Error' dialog box. The dialog contains the message 'ORA-00937: 不是单组分组函数' (ORA-00937: not a grouped function). There are three buttons: 'OK', 'Cancel', and 'Help'.

我们发现报了一个 ORA-00937 的错误

注意：

1. 如果使用分组函数，SQL 只可以把 GROUP BY 分组条件字段和分组函数查询出来，不能有其他字段。

2. 如果使用分组函数，不使用 GROUP BY 只可以查询出来分组函数的值

这很好理解，比如根据部门号分组，如果选择列表中出现员工姓名，那10号部门该用什么名字匹配呢！？这明显就出现错误了。

即：多行函数/组函数

```
select deptno, ename, count(ename) from emp group by deptno;
```

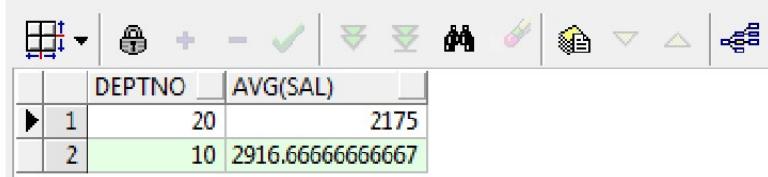
group by a, b, c : 多列分组时，先按a, 再按b...依次类推。



● 过滤分组数据

范例：查询出部门平均工资大于 2000 的部门

```
select deptno, avg(sal) from emp
group by deptno having avg(sal) > 2000
```



The screenshot shows the results of the query in a grid format. The columns are DEPTNO and AVG(SAL). The data is as follows:

	DEPTNO	AVG(SAL)	
▶	1	20	2175
	2	10	2916.666666666667

● WHERE 和 HAVING 的区别

最大区别在于：where 后面不能有组函数

如果条件中没有含有分组函数，having 和where都可以使用。只不过where在group之前，having在group之后。
sql 优化：尽量使用where。

```
SELECT    deptno, AVG(sal)
FROM      emp
WHERE     AVG(sal) > 8000
GROUP BY deptno;
```

```
WHERE    AVG(sal) > 8000
*
ERROR at line 3:
ORA-00934: group function is not allowed here
```

```
无标题 - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
select deptno, job, sum(sal) from emp group by deptno, job
+
select deptno, sum(sal) from emp group by deptno
+
select sum(sal) from emp
=====
select deptno, job, sum(sal) from emp group by rollup(deptno, job)
抽象
group by rollup(a, b)
=
group by a, b
+
group by a
+
没有group by
```

group by增强--roll up (了解)

六、多表查询

● 什么是笛卡尔积？

EMPLOYEES (20行)

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
100	King	90
101	Kochhar	90
...		
202	Fay	20
205	Higgins	110
206	Gietz	110

20 rows selected.

DEPARTMENTS (8行)

DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID
10	Administration	1700
20	Marketing	1800
50	Shipping	1500
60	IT	1400
80	Sales	2500
90	Executive	1700
110	Accounting	1700
190	Contracting	1700

8 rows selected.

笛卡尔积：
 $20 \times 8 = 160$ 行

EMPLOYEE_ID	DEPARTMENT_ID	LOCATION_ID
100	90	1700
101	90	1700
102	90	1700
103	60	1700
104	60	1700
107	60	1700

● Oracle 的连接条件的类型

- 等值连接
- 不等值连接
- 外连接
- 自连接

总而言之，多表连接查询，
大体上就分为这三个：
1. 内连接：等值和不等值
2. 外连接：左外和右外
3. 自连接：一张表当做多张
表来看
(交叉连接就是笛卡尔积)

● Oracle 多表连接示例

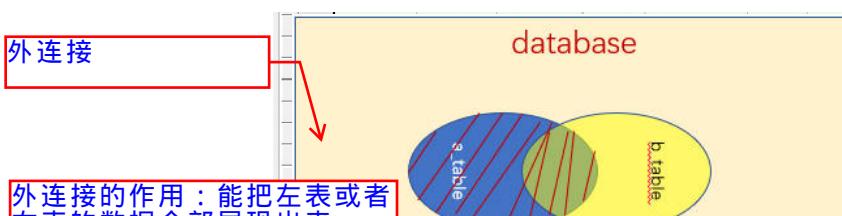
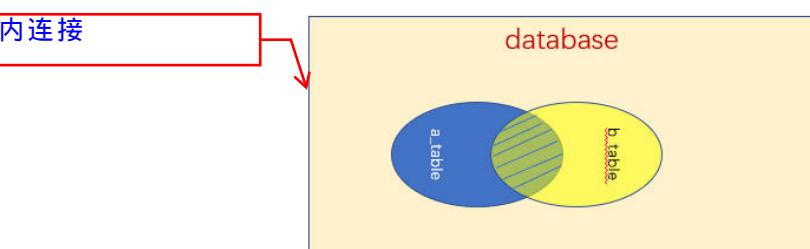
1. 多表连接基本查询

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

说明：组合两个表中的记录，返回关联字段相符的记录，也就是返回两个表的交集（阴影）部分。



外连接的作用：能把左表或者右表的数据全部展现出来。
使用场景：当某数据为null时，使用内连接查询导致数据查询丢失，此时就需要使用外连接。

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。

```
select * from emp, dept where emp.deptno = dept.deptno
```

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO	DEPTNO	DNAME	LOC
► 1	7369	SMITH	CLERK	7902	1980/12/17	800.00		20	20	RESEARCH	DALLAS
2	7499	ALLEN	SALESMAN	7698	1981/2/20	1600.00	300.00	30	30	SALES	CHICAGO
3	7521	WARD	SALESMAN	7698	1981/2/22	1250.00	500.00	30	30	SALES	CHICAGO
4	7566	JONES	MANAGER	7839	1981/4/2	2975.00		20	20	RESEARCH	DALLAS
5	7654	MARTIN	SALESMAN	7698	1981/9/28	1250.00	1400.00	30	30	SALES	CHICAGO
6	7698	BLAKE	MANAGER	7839	1981/5/1	2850.00		30	30	SALES	CHICAGO
7	7782	CLARK	MANAGER	7839	1981/6/9	2450.00		10	10	ACCOUNTING	NEW YORK
8	7788	SCOTT	ANALYST	7566	1987/4/19	3000.00		20	20	RESEARCH	DALLAS
9	7839	KING	PRESIDENT		1981/11/17	5000.00		10	10	ACCOUNTING	NEW YORK
10	7844	TURNER	SALESMAN	7698	1981/9/8	1500.00	0.00	30	30	SALES	CHICAGO
11	7876	ADAMS	CLERK	7788	1987/5/23	1100.00		20	20	RESEARCH	DALLAS
12	7900	JAMES	CLERK	7698	1981/12/3	950.00		30	30	SALES	CHICAGO
13	7902	FORD	ANALYST	7566	1981/12/3	3000.00		20	20	RESEARCH	DALLAS
14	7934	MILLER	CLERK	7782	1982/1/23	1300.00		10	10	ACCOUNTING	NEW YORK

关联之后我们发现数据条数是 14 条，不再是 56 条。

多表查询我们可以为每一张表起一个别名

```
select * from emp e, dept d where e.deptno = d.deptno
```

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO	DEPTNO	DNAME	LOC
1	7369	SMITH	CLERK	7902	1980/12/17	800.00		20	20	RESEARCH	DALLAS
2	7499	ALLEN	SALESMAN	7698	1981/2/20	1600.00	300.00	30	30	SALES	CHICAGO
3	7521	WARD	SALESMAN	7698	1981/2/22	1250.00	500.00	30	30	SALES	CHICAGO
4	7566	JONES	MANAGER	7839	1981/4/2	2975.00		20	20	RESEARCH	DALLAS
5	7654	MARTIN	SALESMAN	7698	1981/9/28	1250.00	1400.00	30	30	SALES	CHICAGO
6	7698	BLAKE	MANAGER	7839	1981/5/1	2850.00		30	30	SALES	CHICAGO
7	7782	CLARK	MANAGER	7839	1981/6/9	2450.00		10	10	ACCOUNTING	NEW YORK
8	7788	SCOTT	ANALYST	7566	1987/4/19	3000.00		20	20	RESEARCH	DALLAS
9	7839	KING	PRESIDENT		1981/11/17	5000.00		10	10	ACCOUNTING	NEW YORK
10	7844	TURNER	SALESMAN	7698	1981/9/8	1500.00	0.00	30	30	SALES	CHICAGO
11	7876	ADAMS	CLERK	7788	1987/5/23	1100.00		20	20	RESEARCH	DALLAS
12	7900	JAMES	CLERK	7698	1981/12/3	950.00		30	30	SALES	CHICAGO
13	7902	FORD	ANALYST	7566	1981/12/3	3000.00		20	20	RESEARCH	DALLAS
14	7934	MILLER	CLERK	7782	1982/1/23	1300.00		10	10	ACCOUNTING	NEW YORK

范例：查询出雇员的编号，姓名，部门的编号和名称，地址

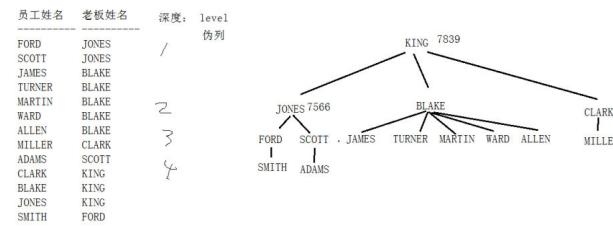
```
select e.empno, e.ename, d.deptno, d.dname, d.loc
from emp e, dept d
where e.deptno = d.deptno
```

	EMPNO	ENAME	DEPTNO	DNAME	LOC
1	7369	SMITH	20	RESEARCH	DALLAS
2	7499	ALLEN	30	SALES	CHICAGO
3	7521	WARD	30	SALES	CHICAGO
4	7566	JONES	20	RESEARCH	DALLAS
5	7654	MARTIN	30	SALES	CHICAGO
6	7698	BLAKE	30	SALES	CHICAGO
7	7782	CLARK	10	ACCOUNTING	NEW YORK
8	7788	SCOTT	20	RESEARCH	DALLAS
9	7839	KING	10	ACCOUNTING	NEW YORK
10	7844	TURNER	30	SALES	CHICAGO
11	7876	ADAMS	20	RESEARCH	DALLAS
12	7900	JAMES	30	SALES	CHICAGO
13	7902	FORD	20	RESEARCH	DALLAS
14	7934	MILLER	10	ACCOUNTING	NEW YORK

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

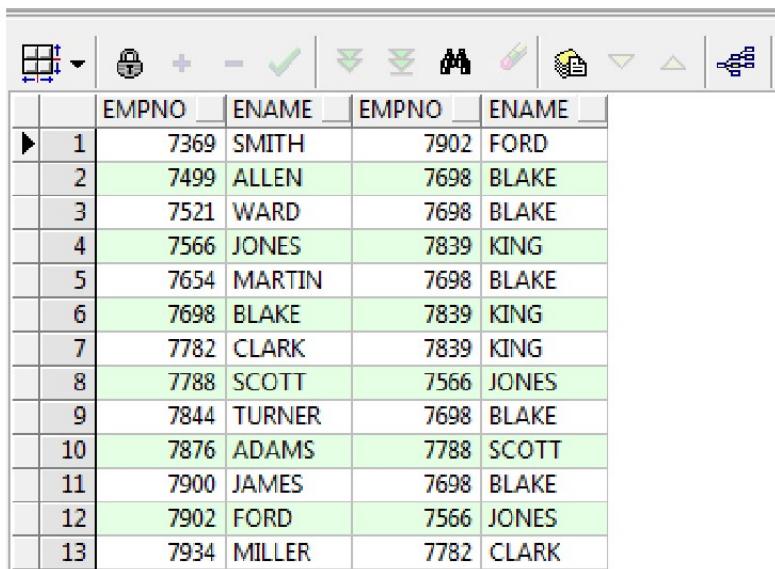
除了使用自连接查询，还可以使用层次查询：



```
select level, empno, ename, mgr
from emp
connect by prior empno=mgr
start with mgr is null
order by 1;
```

1. start with...也可以放在connect by前面；
2. prior在等号哪边，表示哪边是“我的”，即：我的empno=别人的上司；（找下属）
3. level伪列只能和connect by子句结合使用

```
select e.empno, e.ename, e2.empno, e2.ename
from emp e ,emp e2
where e.mgr = e2.empno;
```

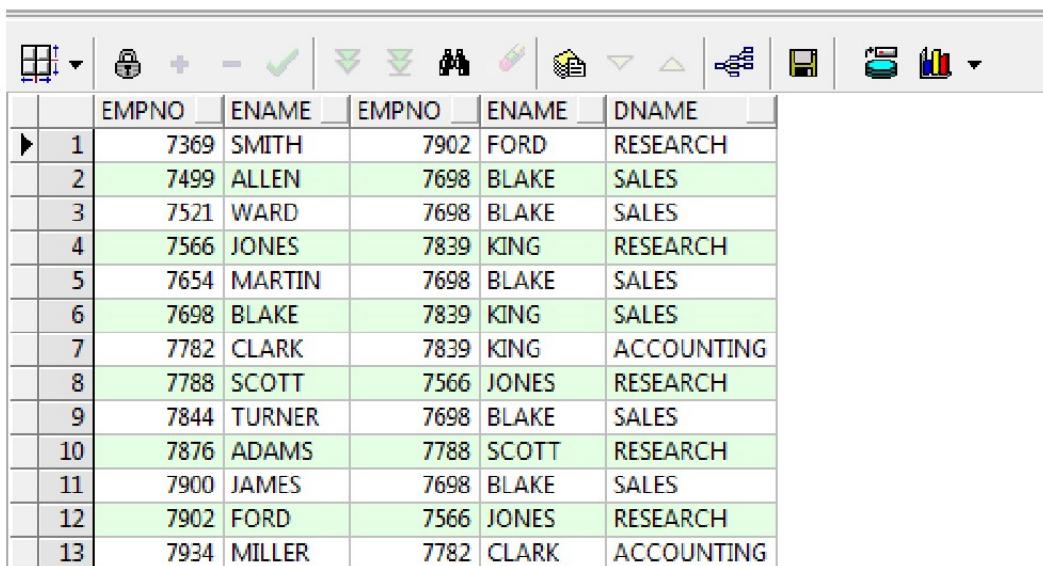


	EMPNO	ENAME	EMPNO	ENAME
▶ 1	7369	SMITH	7902	FORD
2	7499	ALLEN	7698	BLAKE
3	7521	WARD	7698	BLAKE
4	7566	JONES	7839	KING
5	7654	MARTIN	7698	BLAKE
6	7698	BLAKE	7839	KING
7	7782	CLARK	7839	KING
8	7788	SCOTT	7566	JONES
9	7844	TURNER	7698	BLAKE
10	7876	ADAMS	7788	SCOTT
11	7900	JAMES	7698	BLAKE
12	7902	FORD	7566	JONES
13	7934	MILLER	7782	CLARK

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

```
select e.empno, e.ename, e1.empno, e1.ename, d.dname
from emp e, emp e1, dept d
where e.mgr = e1.empno
and e.deptno = d.deptno
```



	EMPNO	ENAME	EMPNO	ENAME	DNAME
▶ 1	7369	SMITH	7902	FORD	RESEARCH
2	7499	ALLEN	7698	BLAKE	SALES
3	7521	WARD	7698	BLAKE	SALES
4	7566	JONES	7839	KING	RESEARCH
5	7654	MARTIN	7698	BLAKE	SALES
6	7698	BLAKE	7839	KING	SALES
7	7782	CLARK	7839	KING	ACCOUNTING
8	7788	SCOTT	7566	JONES	RESEARCH
9	7844	TURNER	7698	BLAKE	SALES
10	7876	ADAMS	7788	SCOTT	RESEARCH
11	7900	JAMES	7698	BLAKE	SALES
12	7902	FORD	7566	JONES	RESEARCH
13	7934	MILLER	7782	CLARK	ACCOUNTING

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
```

	EMPNO	ENAME	GRADE	DNAME	EMPNO	ENAME	GRADE
▶ 1	7369	SMITH	一级	RESEARCH	7902	FORD	四级
2	7900	JAMES	一级	SALES	7698	BLAKE	四级
3	7876	ADAMS	一级	RESEARCH	7788	SCOTT	四级
4	7521	WARD	二级	SALES	7698	BLAKE	四级
5	7654	MARTIN	二级	SALES	7698	BLAKE	四级
6	7934	MILLER	二级	ACCOUNTING	7782	CLARK	四级
7	7844	TURNER	三级	SALES	7698	BLAKE	四级
8	7499	ALLEN	三级	SALES	7698	BLAKE	四级
9	7782	CLARK	四级	ACCOUNTING	7839	KING	五级
10	7698	BLAKE	四级	SALES	7839	KING	五级
11	7566	JONES	四级	RESEARCH	7839	KING	五级
12	7902	FORD	四级	RESEARCH	7566	JONES	四级
13	7788	SCOTT	四级	RESEARCH	7566	JONES	四级

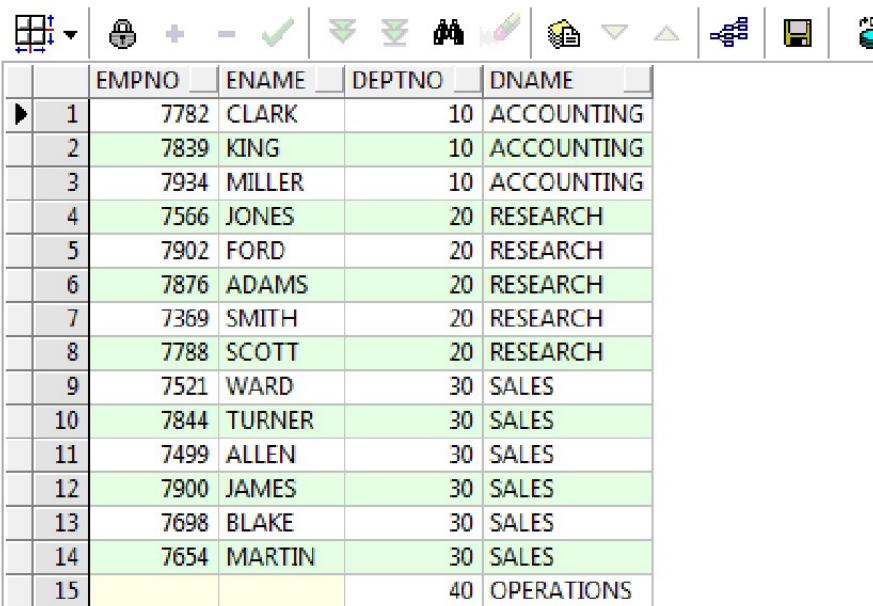
2. 外连接（左右连接）

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

```
select e.empno, e.ename, d.deptno, d.dname
  from emp e, dept d
 where e.deptno (+) = d.deptno;
```

Select dept | Select emp | Select emp



	EMPNO	ENAME	DEPTNO	DNAME
▶ 1	7782	CLARK	10	ACCOUNTING
2	7839	KING	10	ACCOUNTING
3	7934	MILLER	10	ACCOUNTING
4	7566	JONES	20	RESEARCH
5	7902	FORD	20	RESEARCH
6	7876	ADAMS	20	RESEARCH
7	7369	SMITH	20	RESEARCH
8	7788	SCOTT	20	RESEARCH
9	7521	WARD	30	SALES
10	7844	TURNER	30	SALES
11	7499	ALLEN	30	SALES
12	7900	JAMES	30	SALES
13	7698	BLAKE	30	SALES
14	7654	MARTIN	30	SALES
15			40	OPERATIONS

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

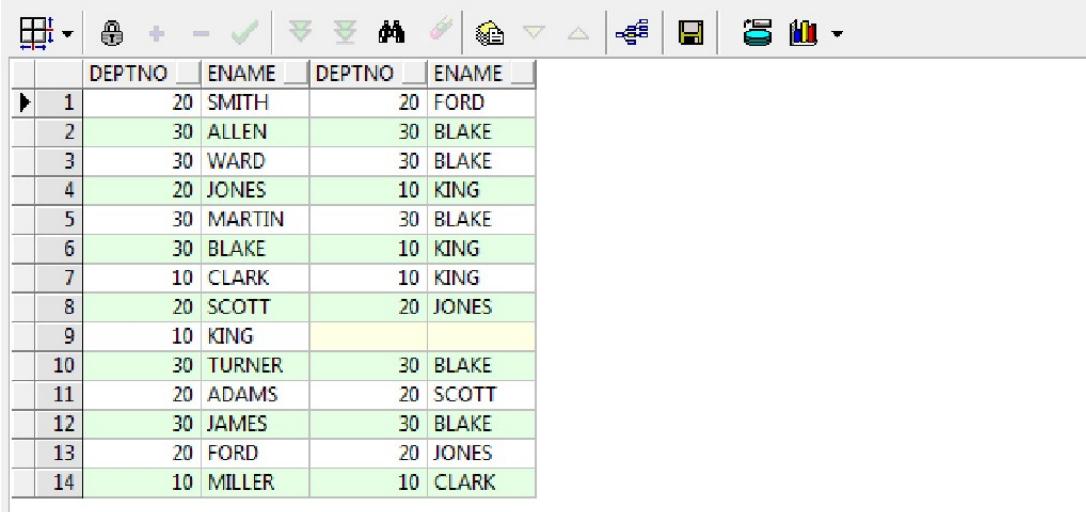
```
select e.empno, e.ename, m.empno, m.ename
  from emp e, emp m
 where e.mgr = m.empno(+)
```



	EMPNO	ENAME	EMPNO	ENAME
▶ 1	7369	SMITH	7902	FORD
2	7499	ALLEN	7698	BLAKE
3	7521	WARD	7698	BLAKE
4	7566	JONES	7839	KING
5	7654	MARTIN	7698	BLAKE
6	7698	BLAKE	7839	KING
7	7782	CLARK	7839	KING
8	7788	SCOTT	7566	JONES
9	7839	KING		
10	7844	TURNER	7698	BLAKE
11	7876	ADAMS	7788	SCOTT
12	7900	JAMES	7698	BLAKE
13	7902	FORD	7566	JONES
14	7934	MILLER	7782	CLARK

如果用 left join 实现：

```
select e.deptno, e.ename, m.deptno, m.ename
  from emp e left join emp m
  on e.mgr = m.empno
```



	DEPTNO	ENAME	DEPTNO	ENAME
▶	1	20 SMITH	20	FORD
	2	30 ALLEN	30	BLAKE
	3	30 WARD	30	BLAKE
	4	20 JONES	10	KING
	5	30 MARTIN	30	BLAKE
	6	30 BLAKE	10	KING
	7	10 CLARK	10	KING
	8	20 SCOTT	20	JONES
	9	10 KING		
	10	30 TURNER	30	BLAKE
	11	20 ADAMS	20	SCOTT
	12	30 JAMES	30	BLAKE
	13	20 FORD	20	JONES
	14	10 MILLER	10	CLARK

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。