



第1章 Oracle 第一天

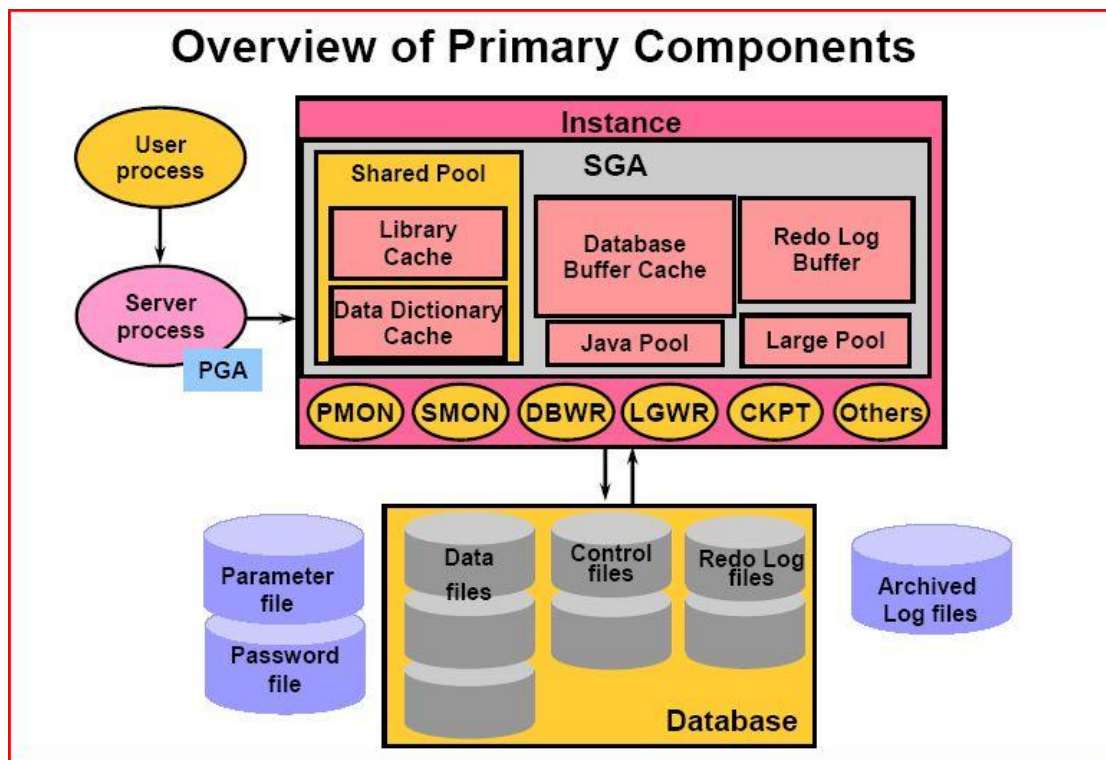
一、oracle介绍[了解]

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。但它的所有知识，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。

二、Oracle安装[了解]

课上已准备好了一个安装全套 Oracle 软件的 XP 虚拟机，我们直接在虚拟机中学习，如果自己想把软件安装到自己电脑请参考文档《Oracle 安装.docx》

三、 Oracle体系结构[理解]



1. 数据库

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。

2. 实例

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。

3. 用户

用户是在实例下建立的。不同实例可以建相同名字的用户。



4. 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。

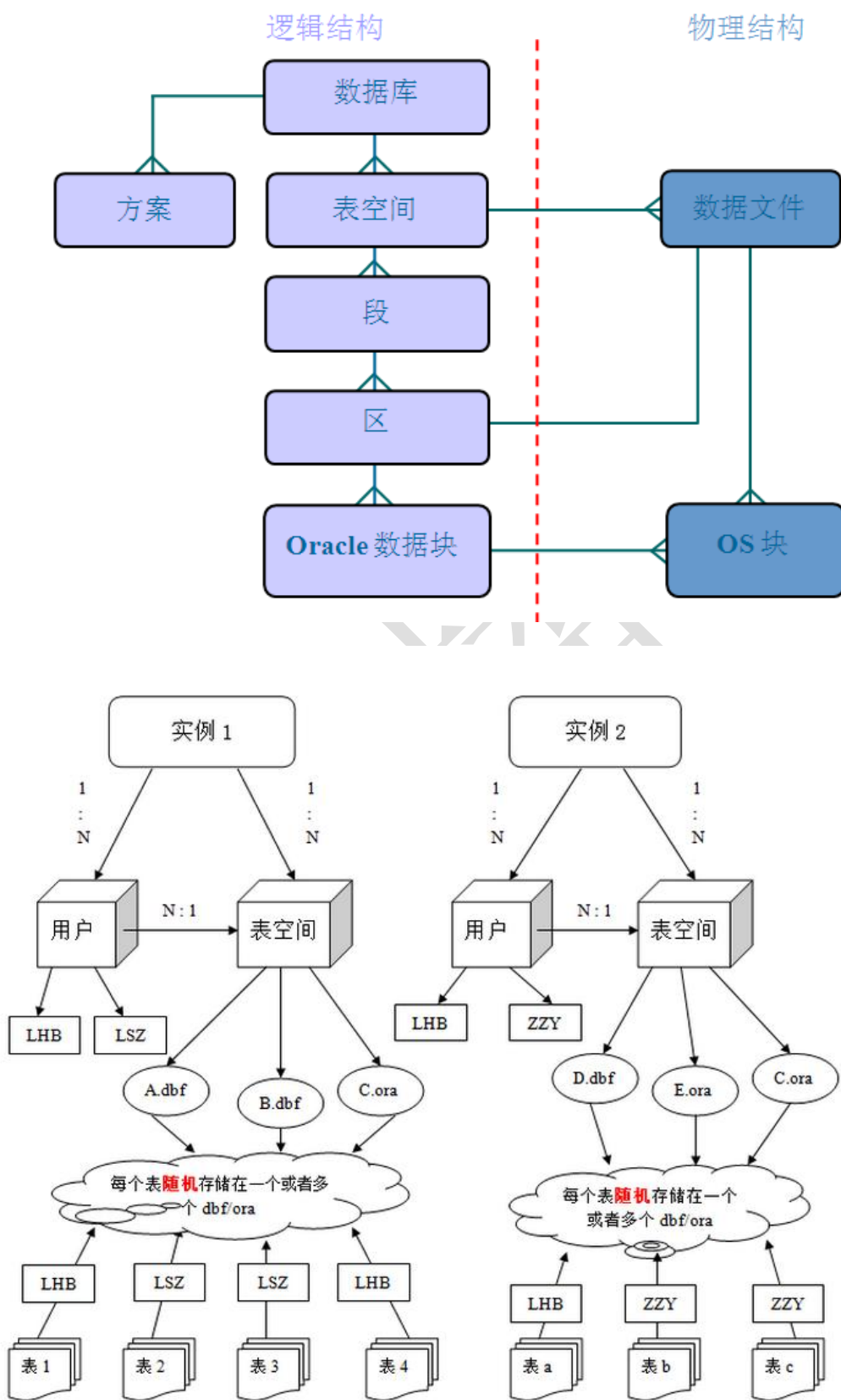


5. 数据文件 (dbf ora)

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。

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一个名字的表！这里区分就是用户了！





四、 创建表空间[理解]

表空间？ ORACLE 数据库的逻辑单元。 数据库---表空间 一个表空间可以与多个数据文件（物理结构）关联

一个数据库下可以建立多个表空间，一个表空间可以建立多个用户、一个用户下可以建立多个表。

```
create tablespace itcast
datafile 'c:\itcast.dbf'
size 100m
autoextend on
next 10m
```

itcast 为表空间名称

datafile 指定表空间对应的数据文件

size 后定义的是表空间的初始大小

autoextend on 自动增长，当表空间存储都占满时，自动增长

next 后指定的是一次自动增长的大小。

五、 用户[理解]

6. 创建用户

```
create user itcastuser
identified by itcast
default tablespace itcast
```

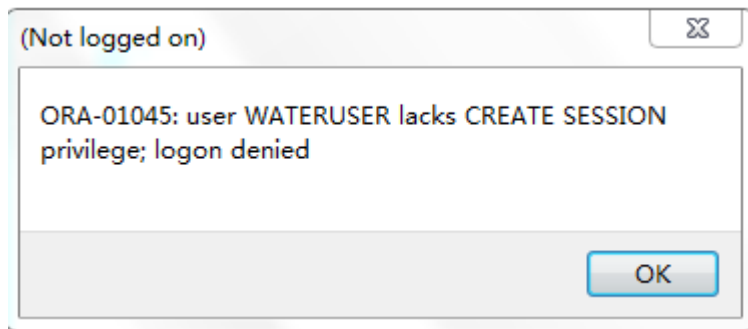
identified by 后边是用户的密码

default tablespace 后边是表空间名称

oracle 数据库与其它数据库产品的区别在于，表和其它的数据库对象都是存储在用户下的。

7. 用户赋权限

新创建的用户没有任何权限，登陆后会提示



Oracle 中已存在三个重要的角色：connect 角色，resource 角色，dba 角色。

CONNECT 角色： --是授予最终用户的典型权利，最基本的

ALTER SESSION --修改会话

CREATE CLUSTER --建立聚簇

CREATE DATABASE LINK --建立数据库链接

CREATE SEQUENCE --建立序列

CREATE SESSION --建立会话

CREATE SYNONYM --建立同义词

CREATE VIEW --建立视图

RESOURCE 角色： --是授予开发人员的

CREATE CLUSTER --建立聚簇

CREATE PROCEDURE --建立过程

CREATE SEQUENCE --建立序列

CREATE TABLE --建表

CREATE TRIGGER --建立触发器

CREATE TYPE --建立类型

DBA 角色：拥有全部特权，是系统最高权限，只有 DBA 才可以创建数据库结构，并且系统权限也需要 DBA 授出，且 DBA 用户可以操作全体用户的任意基表，包括删除

```
grant dba to itcastuser
```

进入 system 用户下给用户赋予 dba 权限，否则无法正常登陆

六、 Oracle数据类型[应用]

No	数据类型	描述
1	Varchar, varchar2	表示一个字符串
2	NUMBER	NUMBER(n)表示一个整数，长度是 n NUMBER(m,n):表示一个小数，总长度是 m，小数是 n，整数是 m-n
3	DATA	表示日期类型
4	CLOB	大对象，表示大文本数据类型，可存 4G
5	BLOB	大对象，表示二进制数据，可存 4G

七、 表的管理[应用]

1.1 建表

语法:

Create table 表名 (

字段 1 数据类型 [default 默认值],

字段 2 数据类型 [default 默认值],

...

字段 n 数据类型 [default 默认值]

);

范例: 创建 person 表

```
create table person(  
    pid      number(10),  
    name     varchar2(10),  
    gender   number(1) default 1,  
    birthday date  
);
```

```
insert into person(pid, name, gender, birthday)
```

```
values(1, '张三', 1, to_date('1999-12-22', 'yyyy-MM-dd'));
```

2.1 表删除

语法: DROP TABLE 表名



3.1 表的修改

在 sql 中使用 alter 可以修改表

- 添加语法: ALTER TABLE 表名称 ADD(列名 1 类型 [DEFAULT 默认值], 列名 1 类型 [DEFAULT 默认值]...)
- 修改语法: ALTER TABLE 表名称 MODIFY(列名 1 类型 [DEFAULT 默认值], 列名 1 类型 [DEFAULT 默认值]...)
- 修改列名: ALTER TABLE 表名称 RENAME 列名 1 TO 列名 2

范例: 在 person 表中增加列 address

```
alter table person add(address varchar2(10));
```

范例: 把 person 表的 address 列的长度修改成 20 长度

```
alter table person modify(address varchar2(20));
```

4.1 数据库表数据的更新

1. INSERT (增加)

标准写法:

```
INSERT INTO 表名[(列名 1, 列名 2, ...)]VALUES(值 1, 值 2, ...)
```

简单写法 (不建议)

```
INSERT INTO 表名 VALUES(值 1, 值 2, ...)
```

注意: 使用简单的写法必须按照表中的字段的顺序来插入值, 而且如果有为空的字段使用 null

```
insert into person
```

```
values (2, '李四', 1, null, '北京育新');
```

2. UPDATE (修改)

全部修改: UPDATE 表名 SET 列名 1=值 1, 列名 2=值 2, ...

局部修改: UPDATE 表名 SET 列名 1=值 1, 列名 2=值 2, ...WHERE 修改条件;

全部更新

局部更新

3. DELETE (删除)

语法 : DELETE FROM 表名 WHERE 删除条件;



在删除语句中如果不指定删除条件的话就会删除所有的数据

因为 oracle 的事务对数据库的变更的处理,我们必须做提交事务才能让数据真正的插入到数据库中,在同样在执行完数据库变更的操作后还可以把事务进行回滚,这样就不会插入到数据库。如果事务提交后则不可以再回滚。

提交: commit

回滚: rollback

5.1 序列

在很多数据库中都存在一个自动增长的列,如果现在要想在 oracle 中完成自动增长的功能,

则只能依靠序列完成,所有的自动增长操作,需要用户手工完成处理。

语法: CREATE SEQUENCE 序列名

[INCREMENT BY n]

[START WITH n]

[{MAXVALUE/ MINVALUE n|NOMAXVALUE}]

[{CYCLE|NOCYCLE}]

[{CACHE n|NOCACHE}];

范例:创建一个 seqpersonid 的序列,验证自动增长的操作

```
CREATE SEQUENCE seqpersonid;
```

序列创建完成之后,所有的自动增长应该由用户自己处理,所以在序列中提供了以下的两种操作:

nextval :取得序列的下一个内容

currval :取得序列的当前内容

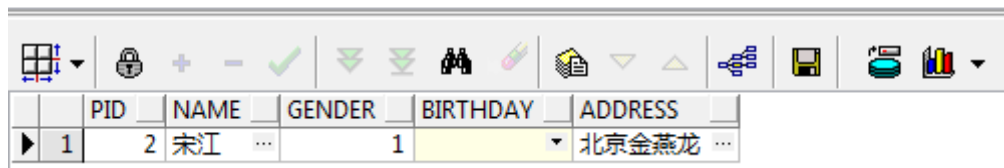
```
select seqpersonid.nextval from dual;
```

```
select seqpersonid.currval from dual;
```

在插入数据时需要自增的主键中可以这样使用

```
insert into person
```

```
values(seqpersonid.nextval,'宋江',1,null,'北京金燕龙');
```



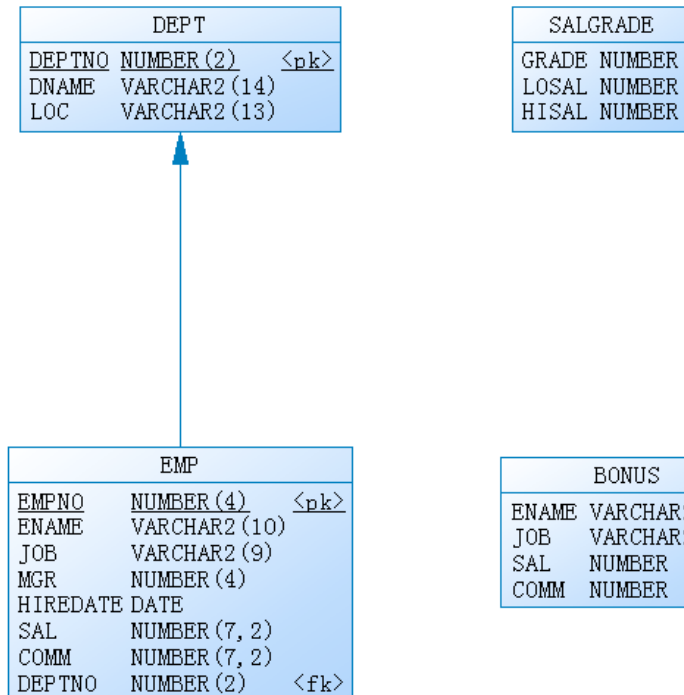
	PID	NAME	GENDER	BIRTHDAY	ADDRESS
1	2	宋江	1		北京金燕龙

在实际项目中每一张表会配一个序列,但是表和序列是没有必然的联系的,一个序列被哪一张表使用都可以,但是我们一般都是用一张表用一个序列。

序列的管理一般使用工具来管理。



八、 Scott用户下的表结构[了解]



雇员表 (EMP)			
No.	字段	类型	描述
1	EMPNO	NUMBER(4)	表示雇员编号，是唯一编号
2	ENAME	VARCHAR2(10)	表示雇员姓名
3	JOB	VARCHAR2(9)	表示工作职位
4	MGR	NUMBER(4)	表示一个雇员的领导编号
5	HIREDATE	DATE	表示雇佣日期
6	SAL	NUMBER(7,2)	表示月薪，工资
7	COMM	NUMBER(7,2)	表示奖金，或者称为佣金
8	DEPTNO	NUMBER(2)	部门编号

部门表 (dept)			
No.	字段	类型	描述
1	DEPTNO	NUMBER(2)	部门编号，是唯一编号
2	DNAME	VARCHAR2(14)	部门名称
3	LOC	VARCHAR2(13)	部门位置

工资等级表 (SALGRADE)			
No.	字段	类型	描述
1	GRADE	NUMBER	等级名称
2	LOSAL	NUMBER	此等级的最低工资
3	HISAL	NUMBER	此等级的最高工资



奖金表 (BONUS)			
No.	字段	类型	描述
1	ENAME	VARCHAR2(10)	雇员姓名
2	JOB	VARCHAR2(9)	雇员工作
3	SAL	NUMBER	雇员工资
4	COMM	NUMBER	雇员奖金 (佣金)

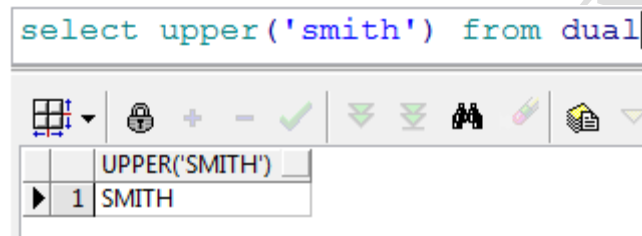
九、 单行函数[应用]

1. 字符函数

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小的字符

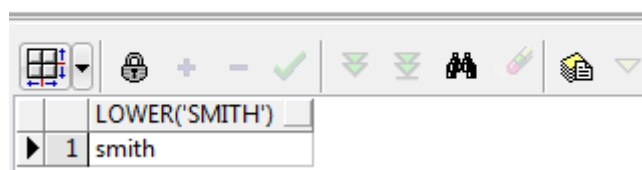
```
upper('smith')
```



2. 把大写字符变成小写字符

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

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



3. 作业

根据 API 学习：首字母大写函数、字符串链接函数、字符串截取函数、字符串替换函数、获取字符串长度函数等。

2. 数值函数

1. 四舍五入函数：ROUND()

默认情况下 ROUND 四舍五入取整，可以自己指定保留的位数。



```
select round(12.534) from dual
```

ROUND(12.534)	
1	13

```
select round(12.534, 2) from dual
```

ROUND(12.534,2)	
1	12.53

2. 作业

根据 API 学习：数值截取函数、取余函数等

1.1.1 3.日期函数

Oracle 中提供了很多和日期相关的函数，包括日期的加减，在日期加减时有一些规律

日期 - 数字 = 日期

日期 + 数字 = 日期

日期 - 日期 = 数字

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,H
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

1.1.2 4.转换函数

1. TO_CHAR:字符串转换函数

范例：查询所有的雇员将年月日分开，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') 年,
       to_char(hiredate, 'mm') 月,
       to_char(hiredate, 'dd') 日
from emp;
```

	EMPNO	ENAME	年	月	日
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
11	7876	ADAMS	1987	05	23
12	7900	JAMES	1981	12	03

```
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

2. 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

1.1.3 5.通用函数

1. 空值处理 nvl

范例：查询所有的雇员的年薪

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

	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

2.Decode 函数

该函数类似 if....else if...esle

语法: DECODE(col/expression, [search1,result1],[search2, result2]....[default])

Col/expression:列名或表达式

Search1, search2....用于比较的条件

Result1, result2....返回值

如果 col/expression 和 Searchi 匹配就返回 resulti,否则返回 default 的默认值

```
select decode(2, 1, '我是1', 2, '我是2', '我是无名') from dual
```

	DECODE(2,1,'我是1',2,'我是2','')
1	我是2

范例：查询出所有雇员的职位的中文名



```
select ename,
       decode(job,
              'CLERK','业务员',
              'SALESMAN','销售',
              'PRESIDENT','总裁',
              'MANAGER','分析员',
              'MANAGER','经理',
              '无业'
       ) from emp;
```

	ENAME	DECODE(JOB,'CLERK','业务员','S
1	SMITH	业务员
2	ALLEN	销售
3	WARD	销售
4	JONES	分析员
5	MARTIN	销售
6	BLAKE	分析员
7	CLARK	分析员
8	SCOTT	无业
9	KING	总裁
10	TURNER	销售
11	ADAMS	业务员
12	JAMES	业务员
13	FORD	无业

3.case when

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

END

```
select t.empno,
       t.ename,
       case
         when t.job = 'CLERK' then
           '业务员'
         when t.job = 'MANAGER' then
           '经理'
         when t.job = 'ANALYST' then
           '分析员'
         when t.job = 'PRESIDENT' then
           '总裁'
         when t.job = 'SALESMAN' then
           '销售'
```



```
else
    '无业'

end
from emp t
```

	EMPNO	ENAME	CASEWHENT.JOB='CLERK'THEN'业务
1	7369	SMITH	业务员
2	7499	ALLEN	销售
3	7521	WARD	销售
4	7566	JONES	经理
5	7654	MARTIN	销售
6	7698	BLAKE	经理
7	7782	CLARK	经理
8	7788	SCOTT	分析员
9	7839	KING	总裁
10	7844	TURNER	销售
11	7876	ADAMS	业务员
12	7900	JAMES	业务员
13	7902	FORD	分析员
14	7934	MILLER	业务员

十、 多行函数（聚合函数）

1.1.3.1 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)
1	14



1.1.3.2 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)
▶	1	800

1.1.3.3 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)
▶	1	5000

1.1.3.4 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)
▶	1	2073.21428571429

1.1.3.5 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



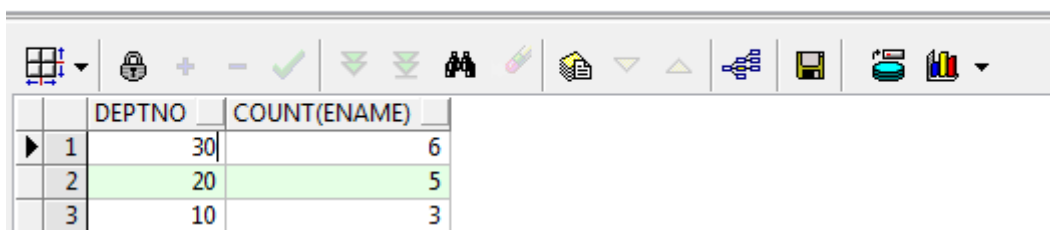
十一、 分组统计

分组统计需要使用 GROUP BY 来分组

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

范例：查询每个部门的人数

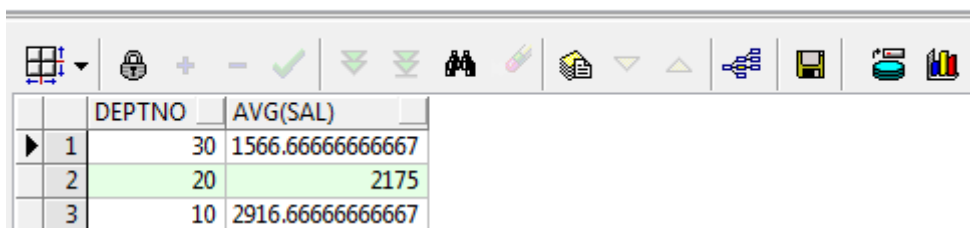
```
select deptno, count(ename) from emp group by deptno
```



	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

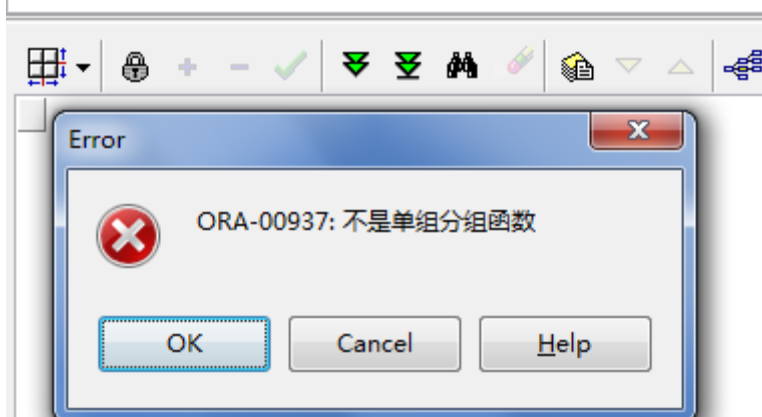
```
select deptno, avg(sal) from emp group by deptno
```



	DEPTNO	AVG(SAL)
▶ 1	30	1566.66666666667
2	20	2175
3	10	2916.66666666667

如果我们想查询出来部门编号，和部门下的人数

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



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



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



范例：按部门分组，查询出部门名称和部门的员工数量

```
select d.deptno, d.dname, count(ename)
  from emp e, dept d
 where e.deptno = d.deptno
 group by d.deptno, d.dname
```

	DEPTNO	DNAME	COUNT(ENAME)
1	10	ACCOUNTING	3
2	20	RESEARCH	5
3	30	SALES	6

范例：查询出部门人数大于 5 人的部门

分析：需要给 count(ename)加条件，此时在本查询中不能使用 where，可以使用 HAVING

```
select d.deptno, d.dname, count(ename)
  from emp e, dept d
 where e.deptno = d.deptno
 group by d.deptno, d.dname
 having count(ename) > 5
```

	DEPTNO	DNAME	COUNT(ENAME)
1	30	SALES	6

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
```

	DEPTNO	AVG(SAL)
1	20	2175
2	10	2916.6666666667

十二、多表查询[应用]

1.1.4 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

```
select * from emp, dept
```

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO	DEPTNO	DNAME	LOC
1	7369	SMITH	CLERK	7902	1980/12/17	800.00		20	10	ACCOUNTING	NEW YORK
2	7499	ALLEN	SALESMAN	7698	1981/2/20	1600.00	300.00	30	10	ACCOUNTING	NEW YORK
3	7521	WARD	SALESMAN	7698	1981/2/22	1250.00	500.00	30	10	ACCOUNTING	NEW YORK
4	7566	JONES	MANAGER	7839	1981/4/2	2975.00		20	10	ACCOUNTING	NEW YORK
5	7654	MARTIN	SALESMAN	7698	1981/9/28	1250.00	1400.00	30	10	ACCOUNTING	NEW YORK
6	7698	BLAKE	MANAGER	7839	1981/5/1	2850.00		30	10	ACCOUNTING	NEW YORK
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	10	ACCOUNTING	NEW YORK
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	10	ACCOUNTING	NEW YORK
11	7876	ADAMS	CLERK	7788	1987/5/23	1100.00		20	10	ACCOUNTING	NEW YORK
12	7900	JAMES	CLERK	7698	1981/12/3	950.00		30	10	ACCOUNTING	NEW YORK
13	7902	FORD	ANALYST	7566	1981/12/3	3000.00		20	10	ACCOUNTING	NEW YORK
14	7934	MILLER	CLERK	7782	1982/1/23	1300.00		10	10	ACCOUNTING	NEW YORK
15	7369	SMITH	CLERK	7902	1980/12/17	800.00		20	20	RESEARCH	DALLAS
16	7499	ALLEN	SALESMAN	7698	1981/2/20	1600.00	300.00	30	20	RESEARCH	DALLAS
17	7521	WARD	SALESMAN	7698	1981/2/22	1250.00	500.00	30	20	RESEARCH	DALLAS
18	7566	JONES	MANAGER	7839	1981/4/2	2975.00		20	20	RESEARCH	DALLAS
19	7654	MARTIN	SALESMAN	7698	1981/9/28	1250.00	1400.00	30	20	RESEARCH	DALLAS
20	7698	BLAKE	MANAGER	7839	1981/5/1	2850.00		30	20	RESEARCH	DALLAS
21	7782	CLARK	MANAGER	7839	1981/6/9	2450.00		10	20	RESEARCH	DALLAS
22	7788	SCOTT	ANALYST	7566	1987/4/19	3000.00		20	20	RESEARCH	DALLAS

56 rows selected in 0.281 seconds

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 deptno，deptno 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，不在是 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

```
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

1.1.5 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，当(+)在左边表的关联条件字段上时是左连接，如果是在右边表的关联条件字段上就是右连接。

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，我们需要使用左右连接把他查询出来



```
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

十三、子查询[应用]

子查询：在一个查询的内部还包括另一个查询，则此查询称为子查询。

Sql的任何位置都可以加入子查询。

范例：查询比 7654 工资高的雇员

分析：查询出 7654 员工的工资是多少，把它作为条件

```
select *
  from emp t1
 where t1.sal > (select t.sal from emp t where t.empno = 7654)
```

	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	7839	KING	PRESIDENT		1981/11/17	5000.00		10
7	7844	TURNER	SALESMAN	7698	1981/9/8	1500.00	0.00	30
8	7902	FORD	ANALYST	7566	1981/12/3	3000.00		20
9	7934	MILLER	CLERK	7782	1982/1/23	1300.00		10

子查询在操作中有三类：

单列子查询：返回的结果是一列的一个内容



单行子查询：返回多个列，有可能是一个完整的记录

多行子查询：返回多条记录

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

```
select *
  from emp t1
 where t1.sal > (select t.sal from emp t where t.empno = 7654)
    and t1.job = (select t2.job from emp t2 where t2.empno = 7788)
```

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
1	7788	SCOTT	ANALYST	7566	1987/4/19	3000.00		20
2	7902	FORD	ANALYST	7566	1981/12/3	3000.00		20

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

```
select d.dname, a.minsal, e.ename
  from dept d,
       (select deptno, min(sal) minsal from emp group by deptno) a,
       emp e
 where d.deptno = a.deptno
    and e.sal = a.minsal
```

	DNAME	MINSAL	ENAME
1	RESEARCH	800	SMITH
2	SALES	950	JAMES
3	ACCOUNTING	1300	MILLER

在返回多条记录的子查询可以把它的结果集当做一张表，给起个别名，如图中的 a。

十四、Rownum与分页查询[应用]

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。

范例：查询 emp 表带有 rownum 列

```
select rownum, t.* from emp t
```

ROWNUM	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

我们可以根据 rownum 来取结果集的前几行，比如前 5 行



```
select rownum, t.* from emp t where rownum < 6
```

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

但是我们不能取到中间几行，因为rownum不支持大于号，只支持小于号，如果想实现我们的需求怎么办呢？答案是使用子查询，也正是oracle分页的做法。

第一种写法：

```
select *
  from (select rownum rm, a.* from (select * from emp) a where rownum < 11) b where b.rm > 5
```

第二种写法：

```
select *
  from (select rownum r ,emp.* from emp) b
 where b.r >5 and b.r <11
```