



北京动力节点教育科技有限公司

# 动力节点课程讲义

DONGLIJIEDIANKECHENGJIANGYI

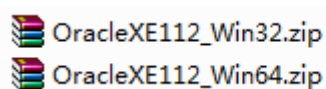
[www.bjpowernode.com](http://www.bjpowernode.com)

# Oracle 讲义

## 第1章 Oracle 数据库的安装

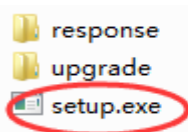
### 1.1 第 1 步

根据自己操作系统的版本选择合适的数据库

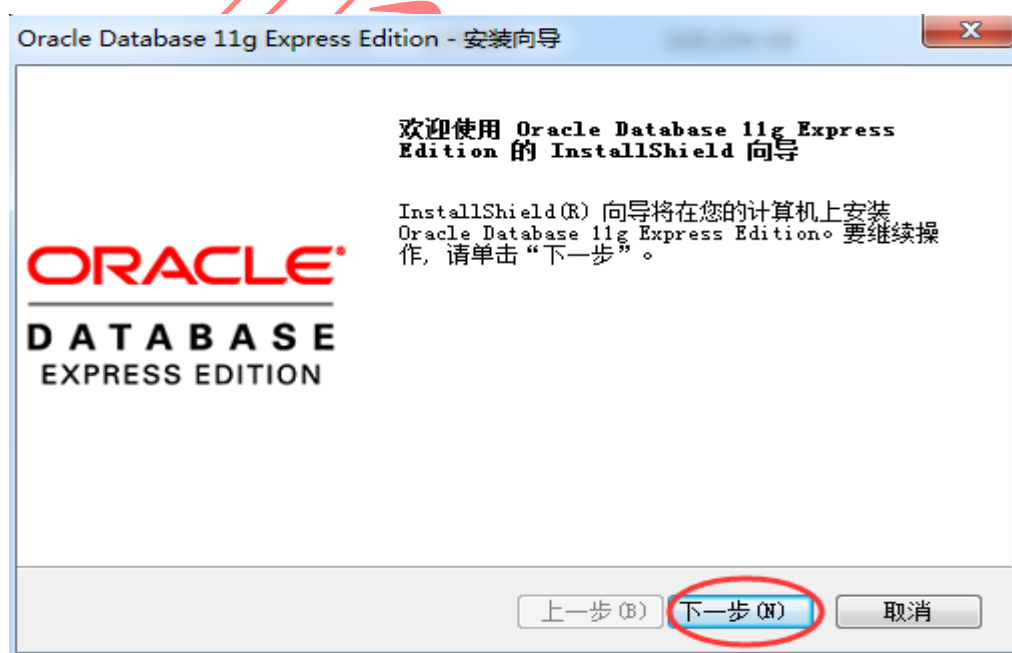


### 1.2 第 2 步

将文件解压后,执行 setup.exe 安装程序



### 1.3 第 3 步



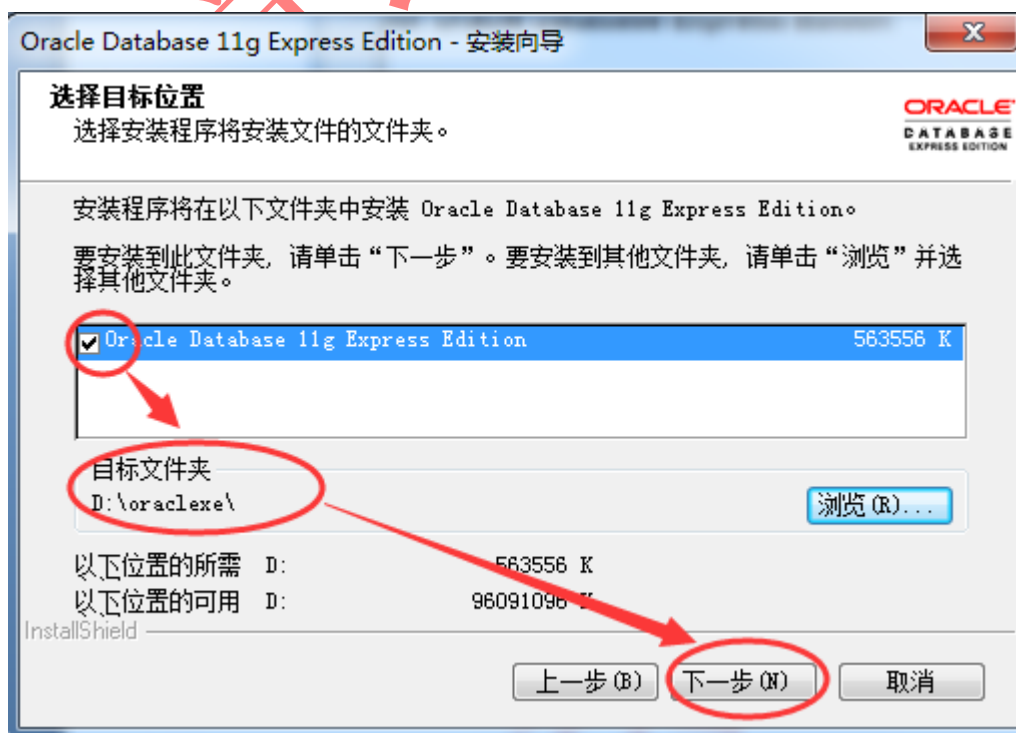
## 1.4 第 4 步

接受许可协议条款,点击下一步



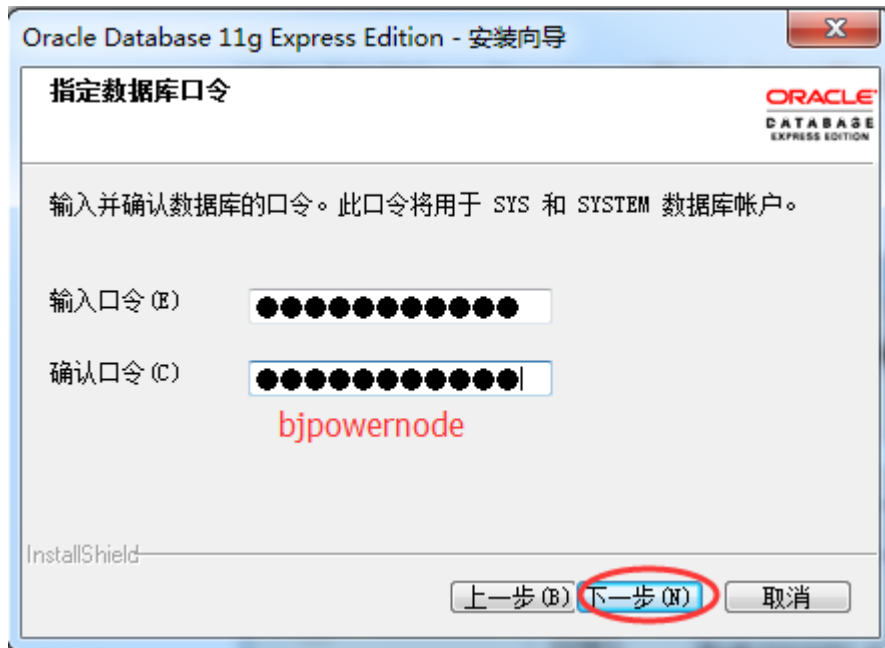
## 1.5 第 5 步

根据自己平时安装软件的习惯,选择安装目录



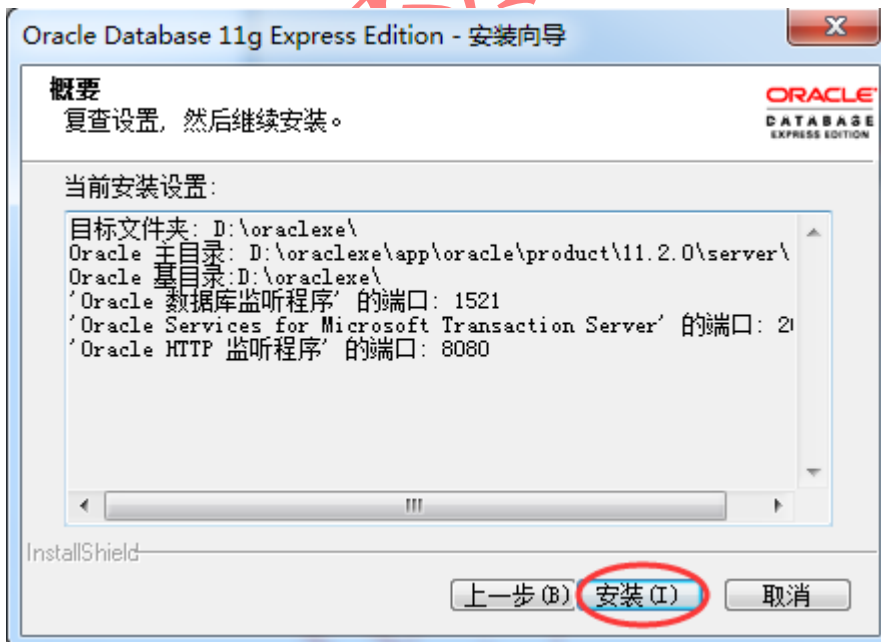
## 1.6 第 6 步

为默认系统用户 SYS 和 SYSTEM 设置密码,我们这里统一使用 bjpowernode



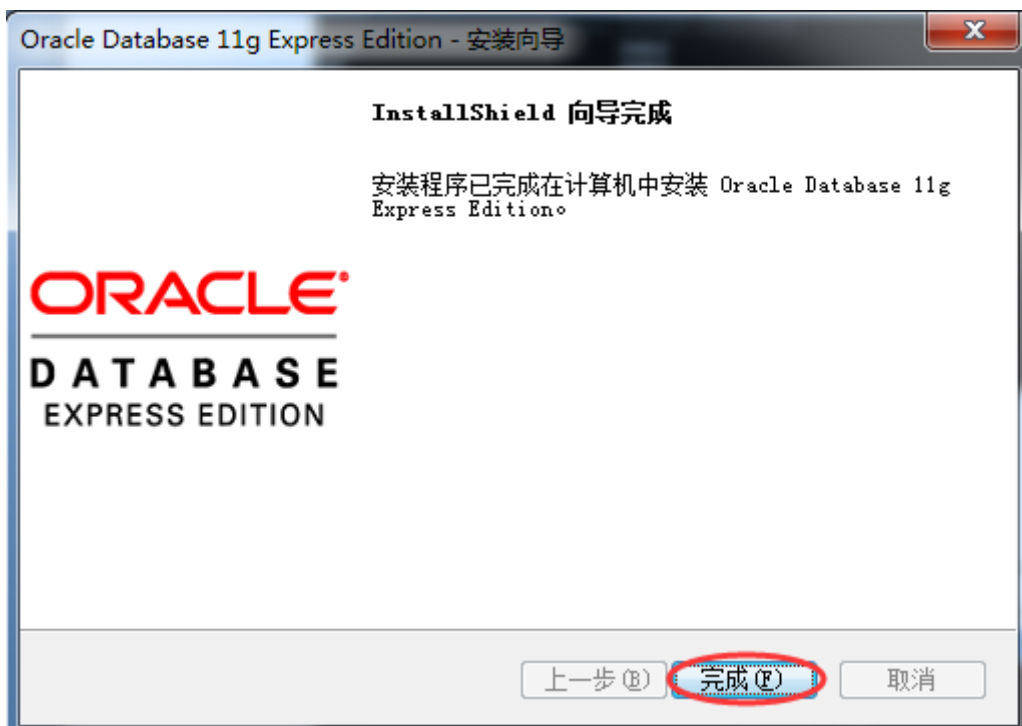
## 1.7 第 7 步

开始安装,安装过程如果有杀毒软件阻止,请选择允许



## 1.8 第 8 步

出现如下界面,安装完成



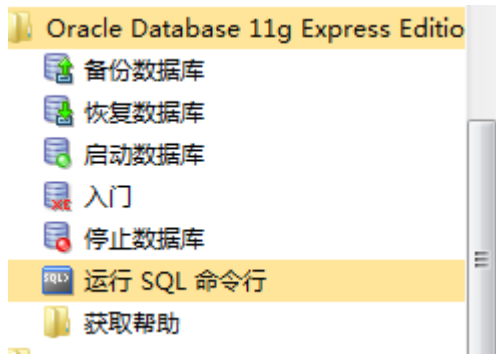
## 1.9 第 9 步

安装完毕后,在系统的环境变量下会有 oracle 相关的配置



## 1.10 第 10 步

安装完毕之后,可以在开始菜单上看到 oracle 的内容



### 1.11 第 11 步

安装完毕之后,可以在服务中看到如下 oracle 服务,其中红色圈起的部分为必须启动项目

OracleJobSchedulerXE	禁用
OracleMTSRecoveryService	手动
OracleServiceXE	已启动 自动
OracleXEClrAgent	手动
OracleXETNSListener	已启动 自动

### 1.12 第 12 步

安装完毕后,可以在安装目录中看一下具体结构

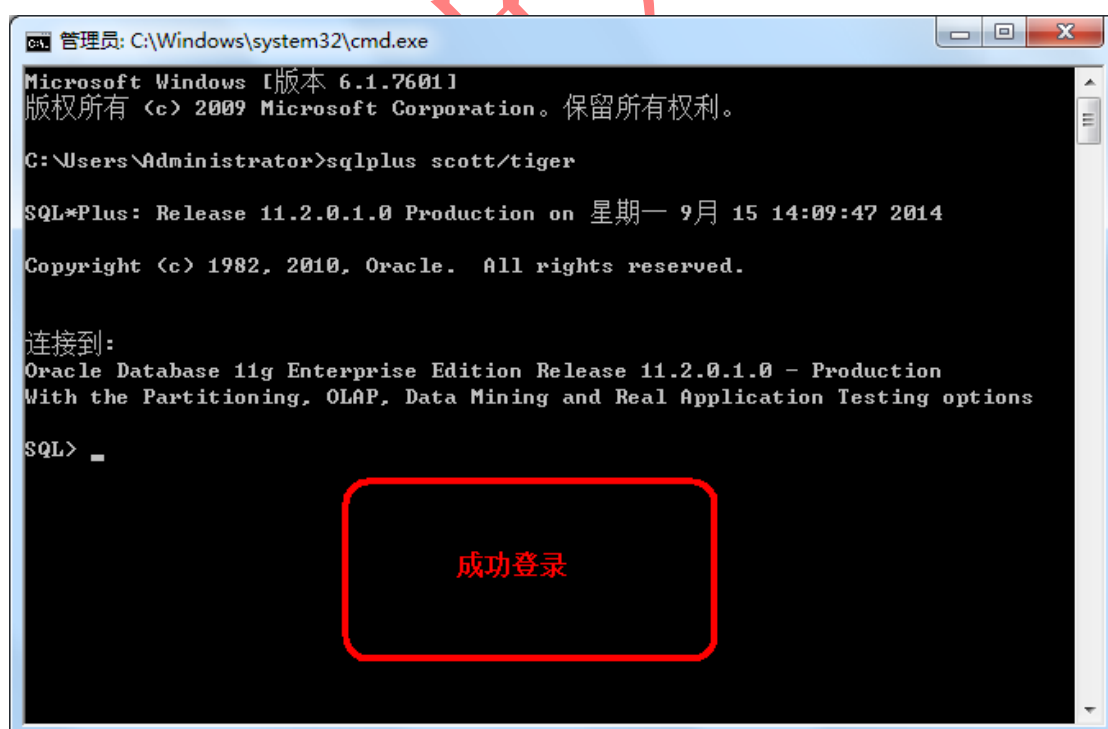
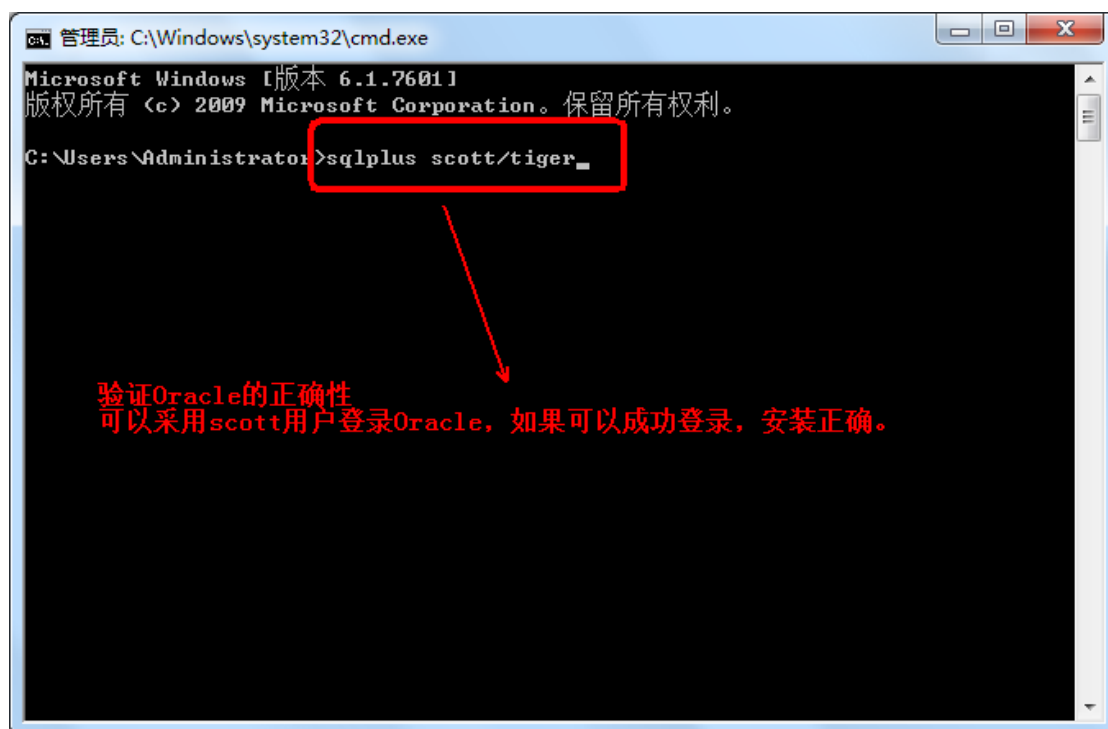
admin	2016/4/13 19:27	文件夹
diag	2016/4/13 19:27	文件夹
fast_recovery_area	2016/4/13 19:28	文件夹
oradata	2016/4/13 19:27	文件夹
product	2016/4/13 19:24	文件夹

### 1.13 第 13 步

如果是 XE 版本,默认是没有 scott 用户的,我们可以执行安装目录创建 scott 用户,初始密码为 tiger,并且初始化数据库 scott 用户中的基本表数据,

@+D:\oraclexe\app\oracle\product\11.2.0\server\rdbms\admin\utlsampl.sql

## 1.14 第 14 步



注意:oracleXE 的监听服务会占用 apache tomcat 的 8080 端口

## 第2章 讲解内容

### 2.1 SQL 概述

SQL，一般发音为 sequel，SQL 的全称 Structured Query Language)，SQL 用来和数据库打交道，完成和数据库的通信，SQL 是一套标准。

### 2.2 什么是数据库

数据库（Data Base、DB），通常是一个或一组文件，保存了一些符合特定规格的数据。数据库软件称为数据库管理系统(DBMS)，全称为 DataBase Management System，如：Oracle、SQL Server、MySQL、Sybase、informix、DB2、interbase、PostgreSql

### 2.3 表

表是一种结构化的文件，可以用来存储特定类型的数据，如：学生信息，课程信息，都可以放到表中。另外表都有特定的名称，而且不能重复。表中具有几个概念：列、行、主键。

学生信息表

学号（主键）	姓名	性别	年龄
00001	张三	男	20
00002	李四	女	20

主键，主键是由列构成的，表中的每一行通常都有一个标识，主键可以由一个字段或多个字段构成，一个字段构成的主键称为单一主键，多个字段构成的主键称为复合主键，主键通常是不能修改的

行，也叫记录，表中的数据是按行(记录)存储的，表里可以有 0 条或多条记录

列，通常也叫字段，表是由列构成的，列是具有类型的

### 2.4 SQL 的分类

数据查询语言（DQL），只有一个 select

数据操纵语言（DML），主要包括：insert/update/delete



数据定义语言（DDL），主要包括：create/drop/alter

事务控制语言（TCL→Transaction Control Language），主要包括：commit/rollback

数据控制语言（DCL），主要包括授权等等

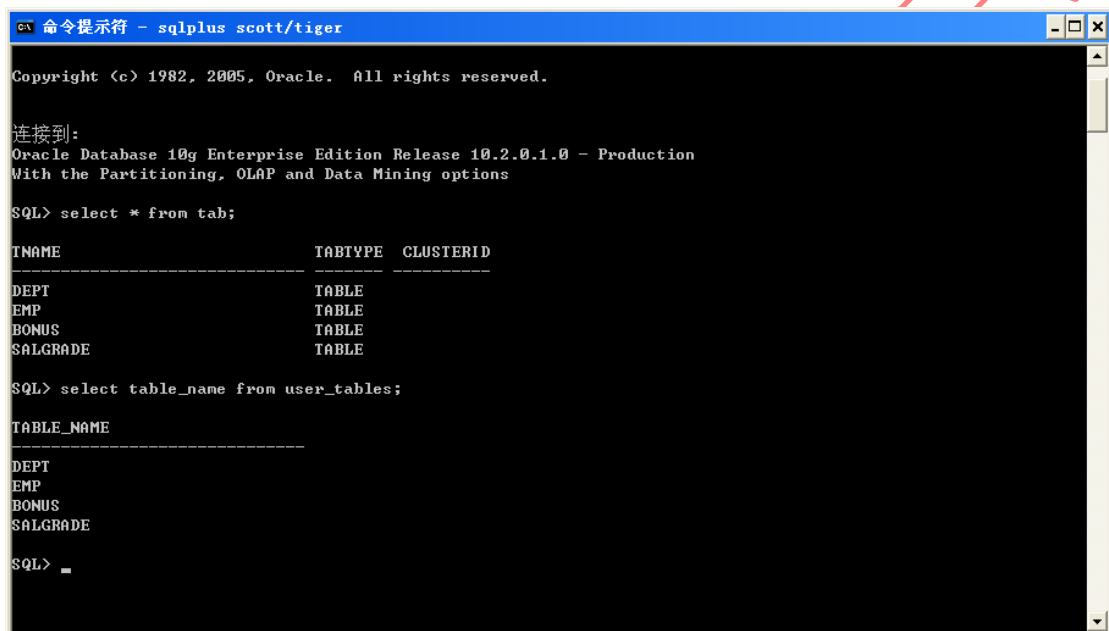
## 2.5 演示数据的结构

### 2.5.1 如何取得演示数据的表

```
select * from tab;
```

或

```
select table_name from user_tables;
```



```
命令提示符 - sqlplus scott/tiger

Copyright (c) 1982, 2005, Oracle. All rights reserved.

连接到:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options

SQL> select * from tab;

TABLENAME          TABTYPE CLUSTERID
-----
DEPT                TABLE
EMP                 TABLE
BONUS               TABLE
SALGRADE            TABLE

SQL> select table_name from user_tables;

TABLE_NAME
-----
DEPT
EMP
BONUS
SALGRADE

SQL>
```

### 2.5.2 查看表结构

```
desc dept
```



```
命令提示符 - sqlplus scott/tiger

SQL> desc dept;
名称
-----
DEPTNO          NOT NULL  NUMBER(2)
DNAME           VARCHAR2(14)
LOC             VARCHAR2(13)

SQL>
```

表结构是由字段构成的，字段是有类型的

### 2.5.3 表结构描述

表名称: dept

描述: 部门信息表

英文字段名称	中文描述	类型
DEPTNO	部门编号	NUMBER(2)
DNAME	部门名称	VARCHAR2(14)
LOC	位置	VARCHAR2(13)

表名称: emp

描述: 员工信息表

英文字段名称	中文描述	类型
EMPNO	员工编号	NUMBER(4)
ENAME	员工姓名	VARCHAR2(10)
JOB	工作岗位	VARCHAR2(9)
MGR	上级经理	NUMBER(4)
HIREDATE	入职日期	DATE
SAL	薪水	NUMBER(7,2)
COMM	津贴	NUMBER(7,2)
DEPTNO	部门编号	NUMBER(2)

注: DEPTNO 字段是外键, DEPTNO 的值来源于 dept 表的主键, 起到了约束的作用

表名称: salgrade

描述: 薪水登记信息表

英文字段名称	中文描述	类型
GRADE	等级	NUMBER
LOSAL	最低薪水	NUMBER
HISAL	最高薪水	NUMBER

### 2.5.4 Oracle 中常见的数据类型

关于 Oracle 数据库中常见的字段数据类型

number

数字类型: 整数型和浮点型数据都是 number

number(3) 表示该字段只能填整数型, 最大值 999

number(3,2) 表示该字段可以填写浮点型数据, 3 表示有效数字的个数, 2 表示两个小数位。

思考: create table t\_a(id number(3,2));

insert into t\_a values(9);可以

insert into t\_a values(10)//不行

### varchar

字符类型：可变长度字符串。

varchar 是 SQL 语句规范中的标准。

通用的。

Varchar 默认采用一个字节编码，~~汉字、全角等采用两个。~~

### varchar2

字符类型：可变长度字符串。

varchar2 只有 Oracle 数据库采用，

不是通用的，但是 varchar2 采用默认采用两个字节编码，可以容纳更多种类的语言。

兼容性更好一些。~~和数据库的字符集有关系~~

### char

字符类型：定长字符串，不可变长度字符串。

在 Oracle 数据库中：char(10)，假如传递的数据是'jack'，底层实际存储的是'jack+6 个空格'

### date

日期类型

### blob

Binary Large Object(二进制大对象)

通常存储：图片、声音、视频等二进制数据。

### clob

Character Large Object(字符大对象)

通常存储：大文本

.....

思考：char 和 varchar2 的选择？

char:该字段中的数据如果是定长的，建议使用 char，例如：'性别'字段。

varchar:该字段中的数据是变化的，建议使用 varchar，例如：'简介'字段。

### t\_movie

id(number)    上映时间(date/varchar)    海报(blob)

-----

Number 类型数据默认的长度是 38，小数长度为 7 → NUMBER(38,7)

VARCHAR2 → VARCHAR

VARCHAR2 是 Oracle 特有的数据类型, 最大长度是 4000(字节)

CHAR 类型和 VARCHAR2 类型在 ORACLE 中都可以用来存储字符串

CHAR 和 VARCHAR2 类型的区别?

CHAR 存储定长字符串。

VARCHAR2 存储可变长字符串。

DATE 日期类型在不同的数据库的处理是不同的, 所以在工作中用的少。

BLOB (Binary Large Object) 二进制大对象 (图片, 视频, 声音)

CLOB (Character Large Object) 字符大对象 (大文本)

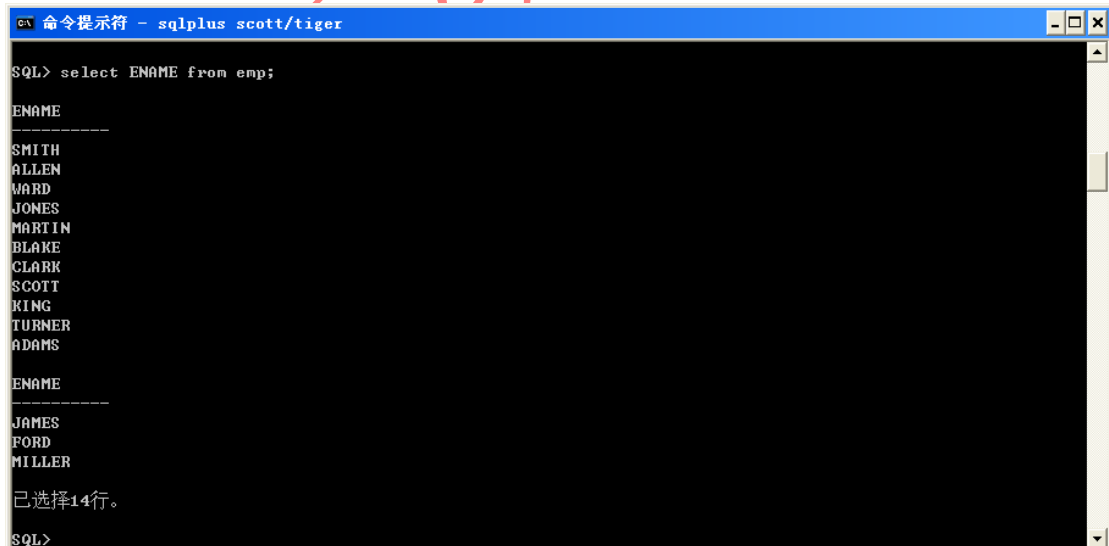
## 第3章 简单查询及 SQLPLUS 常用命令

### 3.1 简单查询

#### 3.1.1 查询一个字段

- 查询员工姓名

```
select ENAME from emp;
```



```
命令提示符 - sqlplus scott/tiger

SQL> select ENAME from emp;

ENAME
-----
SMITH
ALLEN
WARD
JONES
MARTIN
BLAKE
CLARK
SCOTT
KING
TURNER
ADAMS

ENAME
-----
JAMES
FORD
MILLER

已选择14行。

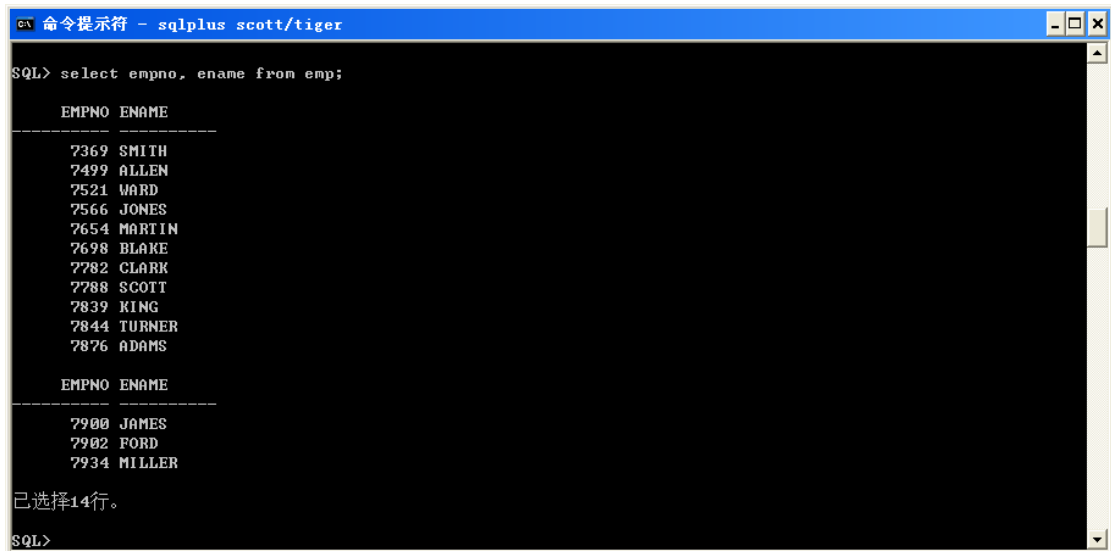
SQL>
```

Select 语句后面跟的是字段名称, select 是关键字, select 和字段名称之间采用空格隔开, from 表示将要查询的表, 它和字段之间采用空格隔开

### 3.1.2 查询多个字段

- 查询员工的编号和姓名(SQL 语句不区分大小写)

```
select empno, ename from emp;
```



```
SQL> select empno, ename from emp;

  EMPNO ENAME
-----
    7369 SMITH
    7499 ALLEN
    7521 WARD
    7566 JONES
    7654 MARTIN
    7698 BLAKE
    7782 CLARK
    7788 SCOTT
    7839 KING
    7844 TURNER
    7876 ADAMS

  EMPNO ENAME
-----
    7900 JAMES
    7902 FORD
    7934 MILLER

已选择14行。

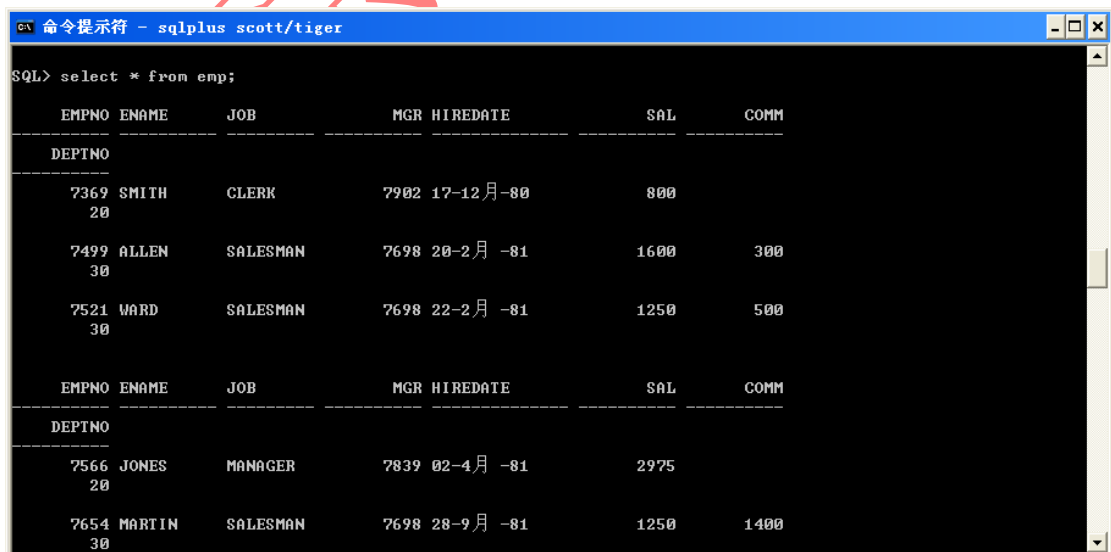
SQL>
```

查询多个字段，需要放到 select 语句的后面，字段之间采用逗号隔开，最后一个字段和 from 不能加逗号

### 3.1.3 查询所有字段

可以将所有字段采用逗号隔开都放到 select 语句后面，但这样不是很方便，所以可以采用如下方式

```
select * from emp;
```



```
SQL> select * from emp;

  EMPNO ENAME      JOB      MGR HIREDATE          SAL        COMM
-----
  DEPTNO
    7369 SMITH      CLERK      7902 17-12月-80         800          0
    7499 ALLEN      SALESMAN   7698 20-2月-81        1600         300
    7521 WARD      SALESMAN   7698 22-2月-81        1250         500
    7566 JONES      MANAGER    7839 02-4月-81        2975          0
    7654 MARTIN     SALESMAN   7698 28-9月-81        1250        1400
    7698 BLAKE      SALESMAN   7698 13-4月-81        2850          0
    7782 CLARK      ANALYST    7788 09-7月-81        2450          0
    7788 SCOTT      ANALYST    7566 04-7月-81        3000          0
    7839 KING      MANAGER    7839 12-12月-81        5000          0
    7844 TURNER     SALESMAN   7844 08-9月-81        1500          0
    7876 ADAMS      CLERK      7876 23-8月-81         750          0
    7900 JAMES      MANAGER    7900 03-12月-81        9500          0
    7902 FORD      ANALYST    7902 04-11月-81        3000          0
    7934 MILLER     CLERK      7934 24-6月-81         1200          0

已选择14行。

SQL>
```

一般建议不使用\*号，使用\*号不明确，建议将相关的字段写到 select 语句的后面，使用\*号的效率比较低

### 3.1.4 计算员工的年薪

- 列出员工的编号，姓名和年薪

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

```
SQL> select empno, ename, sal*12 from emp;
```

EMPNO	ENAME	SAL*12
7369	SMITH	9600
7499	ALLEN	19200
7521	WARD	15000
7566	JONES	35700
7654	MARTIN	15000
7698	BLAKE	34200
7782	CLARK	29400
7788	SCOTT	36000
7839	KING	60000
7844	TURNER	18000
7876	ADAMS	13200

EMPNO	ENAME	SAL*12
7900	JAMES	11400
7902	FORD	36000
7934	MILLER	15600

已选择14行。

```
SQL>
```

在 select 语句中可以使用运算符，以上存在一些问题，年薪的字段名称不太明确

### 3.1.5 将查询出来的字段显示为中文

```
select empno as 员工编号, ename as 员工姓名, sal*12 as 年薪 from emp;
```

```
SQL> select empno as 员工编号, ename as 员工姓名, sal*12 as 年薪 from emp;
```

员工编号	员工姓名	年薪
7369	SMITH	9600
7499	ALLEN	19200
7521	WARD	15000
7566	JONES	35700
7654	MARTIN	15000
7698	BLAKE	34200
7782	CLARK	29400
7788	SCOTT	36000
7839	KING	60000
7844	TURNER	18000
7876	ADAMS	13200

员工编号	员工姓名	年薪
7900	JAMES	11400
7902	FORD	36000
7934	MILLER	15600

已选择14行。

```
SQL>
```

可以采用 as 命名别名，as 可以省略，如：

```
select empno 员工编号, ename 员工姓名, sal*12 年薪 from emp;
```

如果字段重命名之后，新名字中含有空格，那么这个新名字需要用双引号括起来。这是

ORACLE 数据库唯一一个用双引号的地方。(Oracle 中所有的字符串都使用单引号括起来)

## 3.2 SQL Plus 常用命令

### 3.2.1 set linesize 200

set linesize 可以设置一行的字符数，默认为 80 个字符

set linesize 200,表示设置一行为 200 个字符

```

命令提示符 - sqlplus scott/tiger
已选择14行。
SQL> set linesize 200;
SQL> select * from emp;

  EMPNO  ENAME      JOB              MGR HIREDATE          SAL       COMM     DEPTNO
-----
  7369 SMITH        CLERK             7902 17-12月-80          800           0         20
  7499 ALLEN        SALESMAN          7698 20-2月-81          1600          300         30
  7521 WARD           SALESMAN          7698 22-2月-81          1250          500         30
  7566 JONES         MANAGER           7839 02-4月-81          2975           0         20
  7654 MARTIN       SALESMAN          7698 28-9月-81          1250          1400         30
  7678 BLAKE        MANAGER           7839 01-5月-81          2850           0         30
  7782 CLARK        MANAGER           7839 09-6月-81          2450           0         10
  7788 SCOTT        ANALYST           7566 19-4月-87          3000           0         20
  7839 KING           PRESIDENT         17-11月-81          5000           0         10
  7844 TURNER       SALESMAN          7698 08-9月-81          1500           0         30
  7876 ADAMS        CLERK             7788 23-5月-87          1100           0         20

  EMPNO  ENAME      JOB              MGR HIREDATE          SAL       COMM     DEPTNO
-----
  7900 JAMES        CLERK             7698 03-12月-81          950           0         30
  7902 FORD          ANALYST           7566 03-12月-81          3000           0         20
  7934 MILLER       CLERK             7782 23-1月-82          1300           0         10

已选择14行。
SQL>

```

### 3.2.2 l(List)

可以显示缓存区中的最后执行的内容

```

命令提示符 - sqlplus scott/tiger
SQL> l
  1* select * from emp
SQL> list
  1* select * from emp
SQL>

```

### 3.2.3 run ,/, r

以上三个命令功能是一致的，重新运行缓存区中的语句

```
SQL> /
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-12月-80	800		20
7499	ALLEN	SALESMAN	7698	20-2月-81	1600	300	30
7521	WARD	SALESMAN	7698	22-2月-81	1250	500	30
7566	JONES	MANAGER	7839	02-4月-81	2975		20
7654	MARTIN	SALESMAN	7698	28-9月-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-5月-81	2850		30
7782	CLARK	MANAGER	7839	09-6月-81	2450		10
7788	SCOTT	ANALYST	7566	19-4月-87	3000		20
7839	KING	PRESIDENT		17-11月-81	5000		10
7844	TURNER	SALESMAN	7698	08-9月-81	1500	0	30
7876	ADAMS	CLERK	7788	23-5月-87	1100		20

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7900	JAMES	CLERK	7698	03-12月-81	950		30
7902	FORD	ANALYST	7566	03-12月-81	3000		20
7934	MILLER	CLERK	7782	23-1月-82	1300		10

已选择14行。

```
SQL>
```

### 3.2.4 save

save 可以将最后一次在缓存区中执行的语句保存到文件

```
SQL> save c:\emp.sql
已创建 file c:\emp.sql
SQL>
```

### 3.2.5 get

get 可以将文件中的 sql 语句放到缓存区中，采用 / 或 r 或 run，可以执行

```
SQL> get c:\emp.sql;
1* select empno, ename from emp
SQL> /
```

EMPNO	ENAME
7369	SMITH
7499	ALLEN
7521	WARD
7566	JONES
7654	MARTIN
7698	BLAKE
7782	CLARK
7788	SCOTT
7839	KING
7844	TURNER
7876	ADAMS

EMPNO	ENAME
7900	JAMES
7902	FORD
7934	MILLER

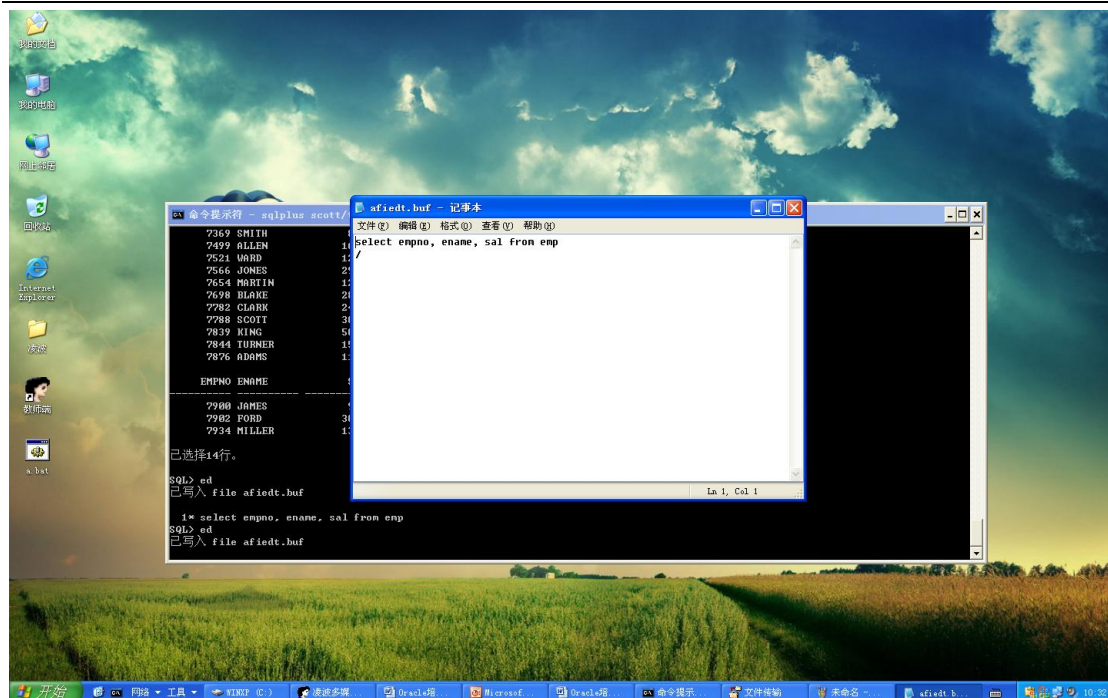
已选择14行。

```
SQL>
```

### 3.2.6 ed(edit)

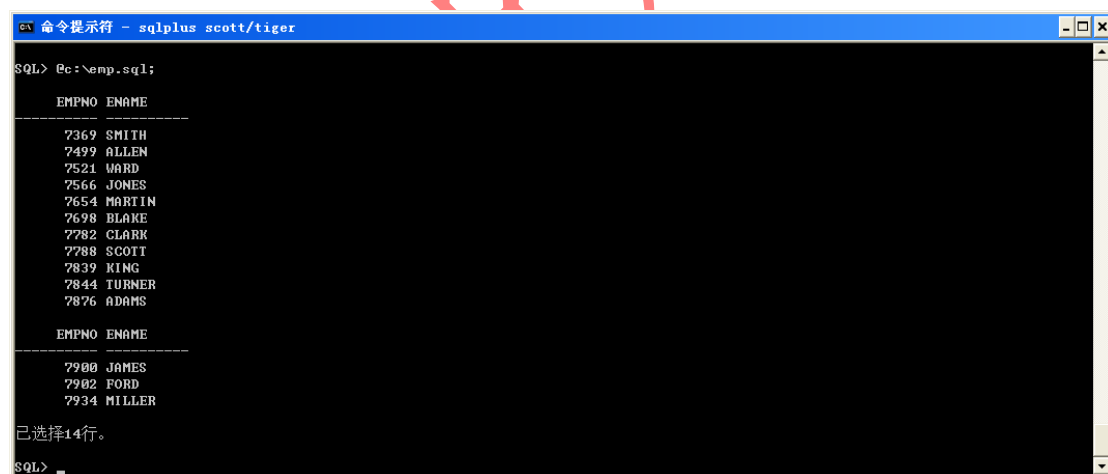
ed 可以采用记事本来编辑缓存区中的内容





### 3.2.7 如何直接执行 sql 脚本

@c:\emp.sql;



## 第4章 条件查询

### 4.1 条件查询

条件查询需要用到 where 语句，where 必须放到 from 语句表的后面  
支持如下运算符

运算符	说明
=	等于
<>或!=	不等于
<	小于
<=	小于等于
>	大于
>=	大于等于
between ... and .... (等同于>= and <=)	两个值之间
is null	为 null
is not null	不是 null

### 4.1.1 等号操作符

- 查询薪水为 5000 的员工
- 如果是字符类型的数据进行比较的时候，是区分大小写的。

```
select empno, ename, sal from emp where sal=5000;
```

```

命令提示符 - sqlplus scott/tiger

SQL> select empno, ename, sal from emp where sal=5000;

   EMPNO  ENAME      SAL
-----
    7839  KING         5000

SQL>

```

### 4.1.2 <>操作符

- 查询薪水不等于 5000 的员工

```
select empno, ename, sal from emp where sal <> 5000;
```

```

命令提示符 - sqlplus scott/tiger

SQL> select empno, ename, sal from emp where sal <> 5000;

   EMPNO  ENAME      SAL
-----
    7369  SMITH         800
    7499  ALLEN        1600
    7521  WARD         1250
    7566  JONES        2975
    7654  MARTIN       1250
    7698  BLAKE        2850
    7782  CLARK        2450
    7788  SCOTT        3000
    7844  TURNER       1500
    7876  ADAMS        1100
    7900  JAMES         950

   EMPNO  ENAME      SAL
-----
    7902  FORD        3000
    7934  MILLER      1300

已选择13行。

SQL>

```

- 查询工作岗位不等于 MANAGER 的员工

```
select empno, ename, sal from emp where job <> 'MANAGER';
```

```
命令提示符 - sqlplus scott/tiger
SQL> select empno, ename, sal from emp where job <> 'manager';

EMPNO  ENAME      SAL
-----
7369 SMITH      800
7499 ALLEN     1600
7521 WARD      1250
7566 JONES     2975
7654 MARTIN    1250
7698 BLAKE     2850
7782 CLARK     2450
7788 SCOTT     3000
7839 KING      5000
7844 TURNER    1500
7876 ADAMS     1100

EMPNO  ENAME      SAL
-----
7900 JAMES      950
7902 FORD      3000
7934 MILLER    1300

已选择14行。
SQL> select empno, ename, sal from emp where job <> 'MANAGER';
```

在 sql 语句中如果是字符串采用单引号，引起来，不同于 java 中采用双引号，如果是数值型也可以引起来,只不过是数值类型数据当成字符串来处理

### 4.1.3 between ... and ...操作符

- 查询薪水为 1600 到 3000 的员工(第一种方式，采用>=和<=)

```
select empno, ename, sal from emp where sal >=1600 and sal <=3000;
```

```
命令提示符 - sqlplus scott/tiger
SQL> select empno, ename, sal from emp where sal >=1600 and sal <=3000;

EMPNO  ENAME      SAL
-----
7499 ALLEN     1600
7566 JONES     2975
7698 BLAKE     2850
7782 CLARK     2450
7788 SCOTT     3000
7902 FORD      3000

已选择6行。
SQL>
```

- 查询薪水为 1600 到 3000 的员工(第二种方式，采用 between ... and ...)

```
select empno, ename, sal from emp where sal between 1600 and 3000;
```

```
命令提示符 - sqlplus scott/tiger
SQL> select empno, ename, sal from emp where sal between 1600 and 3000;

EMPNO  ENAME      SAL
-----
7499 ALLEN     1600
7566 JONES     2975
7698 BLAKE     2850
7782 CLARK     2450
7788 SCOTT     3000
7902 FORD      3000

已选择6行。
SQL>
```

between ....and ..., 包含最大值和最小值

between ....and ..., 不仅仅可以应用在数值类型的数据上，还可以应用在字符数据类型上

between ....and ..., 对于两个参数的设定是有限制的，小的数在前，大的数在后

#### 4.1.4 is null

Null 为空，但不是空串，为 null 可以设置这个字段不同填值，如果查询为 null 的字段，采用 is null

- 查询津贴为空的员工

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

```
SQL> select * from emp where comm is null;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-12月-80	800		20
7566	JONES	MANAGER	7839	02-4月-81	2975		20
7698	BLAKE	MANAGER	7839	01-5月-81	2850		30
7782	CLARK	MANAGER	7839	09-6月-81	2450		10
7788	SCOTT	ANALYST	7566	19-4月-87	3000		20
7839	KING	PRESIDENT		17-11月-81	5000		10
7876	ADAMS	CLERK	7788	23-5月-87	1100		20
7900	JAMES	CLERK	7698	03-12月-81	950		30
7902	FORD	ANALYST	7566	03-12月-81	3000		20
7934	MILLER	CLERK	7782	23-1月-82	1300		10

已选择10行。

- 查询津贴不为空的员工

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

```
SQL> select * from emp where comm is not null;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7499	ALLEN	SALESMAN	7698	20-2月-81	1600	300	30
7521	WARD	SALESMAN	7698	22-2月-81	1250	500	30
7654	MARTIN	SALESMAN	7698	28-9月-81	1250	1400	30
7844	TURNER	SALESMAN	7698	08-9月-81	1500	0	30

#### 4.1.5 and

and 表示并且的含义，表示所有的条件必须满足

- 工作岗位为 MANAGER,薪水大于 2500 的员工

```
select empno, ename, sal from emp where job='MANAGER' and sal>2500;
```

```
SQL> select empno, ename, job, sal from emp where job = 'MANAGER' and sal > 2500;
```

EMPNO	ENAME	JOB	SAL
7566	JONES	MANAGER	2975
7698	BLAKE	MANAGER	2850

#### 4.1.6 or

or, 只要满足条件即可,相当于或者

- 查询出 job 为 manager 和 job 为 salesman 的员工

```
select * from emp where job='MANAGER' or job='SALESMAN';
```

```
SQL> select * from emp where job='MANAGER' or job='SALESMAN';
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7499	ALLEN	SALESMAN	7698	20-2月-81	1600	300	30
7521	WARD	SALESMAN	7698	22-2月-81	1250	500	30
7566	JONES	MANAGER	7839	02-4月-81	2975		20
7654	MARTIN	SALESMAN	7698	28-9月-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-5月-81	2850		30
7782	CLARK	MANAGER	7839	09-6月-81	2450		10
7844	TURNER	SALESMAN	7698	08-9月-81	1500	0	30

已选择7行。

## 4.1.7 表达式的优先级

- 查询薪水大于 1800，并且部门代码为 20 或 30 的（错误的写法）

```
select * from emp where sal>1800 and deptno=20 or deptno=30;
```

```
SQL> select * from emp where sal>1800 and deptno=20 or deptno=30;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7499	ALLEN	SALESMAN	7698	20-2月-81	1600	300	30
7521	WARD	SALESMAN	7698	22-2月-81	1250	500	30
7566	JONES	MANAGER	7839	02-4月-81	2975		20
7654	MARTIN	SALESMAN	7698	28-9月-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-5月-81	2850		30
7788	SCOTT	ANALYST	7566	19-4月-87	3000		20
7844	TURNER	SALESMAN	7698	08-9月-81	1500	0	30
7900	JAMES	CLERK	7698	03-12月-81	950		30
7902	FORD	ANALYST	7566	03-12月-81	3000		20

已选择9行。

以上输出是错误的，由于表达式优先级导致的，首先查询出 `sal>1800 and deptno=20` 的员工，在和并上 `deptno=30` 的员工

- 查询薪水大于 1800，并且部门代码为 20 或 30 的（正确的写法）

```
select * from emp where sal>1800 and (deptno=20 or deptno=30);
```

```
SQL> select * from emp where sal>1800 and (deptno=20 or deptno=30);
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7566	JONES	MANAGER	7839	02-4月-81	2975		20
7698	BLAKE	MANAGER	7839	01-5月-81	2850		30
7788	SCOTT	ANALYST	7566	19-4月-87	3000		20
7902	FORD	ANALYST	7566	03-12月-81	3000		20

已选择4行。

关于运算符的问题：不用记，没有把握尽量采用括号

## 4.1.8 in（忽略空值）

in 表示包含的意思，完全可以采用 or 来表示，采用 in 会更简洁一些

- 查询出 job 为 manager 和 job 为 salesman 的员工

```
select * from emp where job in('MANAGER','SALESMAN');
```

命令提示符 - sqlplus scott/tiger

```
SQL> select * from emp where job in('MANAGER','SALESMAN');
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7499	ALLEN	SALESMAN	7698	20-2月-81	1600	300	30
7521	WARD	SALESMAN	7698	22-2月-81	1250	500	30
7566	JONES	MANAGER	7839	02-4月-81	2975		20
7654	MARTIN	SALESMAN	7698	28-9月-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-5月-81	2850		30
7782	CLARK	MANAGER	7839	09-6月-81	2450		10
7844	TURNER	SALESMAN	7698	08-9月-81	1500	0	30

已选择7行。

```
SQL>
```

#### 4.1.9 not in (不忽略空值)

- 查询 job 不等于 MANAGER 并且不等与 SALESMAN 的员工 (第一种写法)

```
select * from emp where job <> 'MANAGER' and job <> 'SALESMAN';
```

命令提示符 - sqlplus scott/tiger

```
SQL> select * from emp where job <> 'MANAGER' and job <> 'SALESMAN';
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-12月-80	800		20
7788	SCOTT	ANALYST	7566	19-4月-87	3000		20
7839	KING	PRESIDENT		17-11月-81	5000		10
7876	ADAMS	CLERK	7788	23-5月-87	1100		20
7900	JAMES	CLERK	7698	03-12月-81	950		30
7902	FORD	ANALYST	7566	03-12月-81	3000		20
7934	MILLER	CLERK	7782	23-1月-82	1300		10

已选择7行。

```
SQL>
```

- 查询 job 不等于 MANAGER 并且不能与 SALESMAN 的员工 (第二种写法)

```
select * from emp where job not in('MANAGER','SALESMAN');
```

命令提示符 - sqlplus scott/tiger

```
SQL> select * from emp where job not in('MANAGER','SALESMAN');
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-12月-80	800		20
7788	SCOTT	ANALYST	7566	19-4月-87	3000		20
7839	KING	PRESIDENT		17-11月-81	5000		10
7876	ADAMS	CLERK	7788	23-5月-87	1100		20
7900	JAMES	CLERK	7698	03-12月-81	950		30
7902	FORD	ANALYST	7566	03-12月-81	3000		20
7934	MILLER	CLERK	7782	23-1月-82	1300		10

已选择7行。

```
SQL>
```

#### 4.1.10 like

Like 可以实现模糊查询, like 支持%和下划线匹配

- 查询姓名以 M 开头所有的员工

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

```
SQL> select * from emp where ename like 'M%';
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7654	MARTIN	SALESMAN	7698	28-9月-81	1250	1400	30
7934	MILLER	CLERK	7782	23-1月-82	1300		10

- 查询姓名以 T 结尾的所有的员工

```
select * from emp where ename like '%T';
```

```
SQL> select * from emp where ename like '%T';
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7788	SCOTT	ANALYST	7566	19-4月-87	3000		20

- 查询姓名中包含 O 的所有的员工

```
select * from emp where ename like '%O%';
```

```
SQL> select * from emp where ename like '%O%';
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7566	JONES	MANAGER	7839	02-4月-81	2975		20
7788	SCOTT	ANALYST	7566	19-4月-87	3000		20
7902	FORD	ANALYST	7566	03-12月-81	3000		20

- 查询姓名中第二个字符为 A 的所有员工

```
select * from emp where ename like '_A%';
```

```
SQL> select * from emp where ename like '_A%';
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7521	WARD	SALESMAN	7698	22-2月-81	1250	500	30
7654	MARTIN	SALESMAN	7698	28-9月-81	1250	1400	30
7900	JAMES	CLERK	7698	03-12月-81	950		30

Like 中%和下划线的差别?

%匹配任意字符出现任意次数

下划线只匹配一个任意字符出现一次

Like 语句是可以应用在数值类型的数据上的,但是如果没有使用引号括起来的话,那么不能使用%和下划线。类似于等号的操作,如果使用引号括起来的话,那么可以使用%和下划线,将数值类型的数据转换为字符类型后进行处理。

查询出系统中表名含有“下划线”的表。

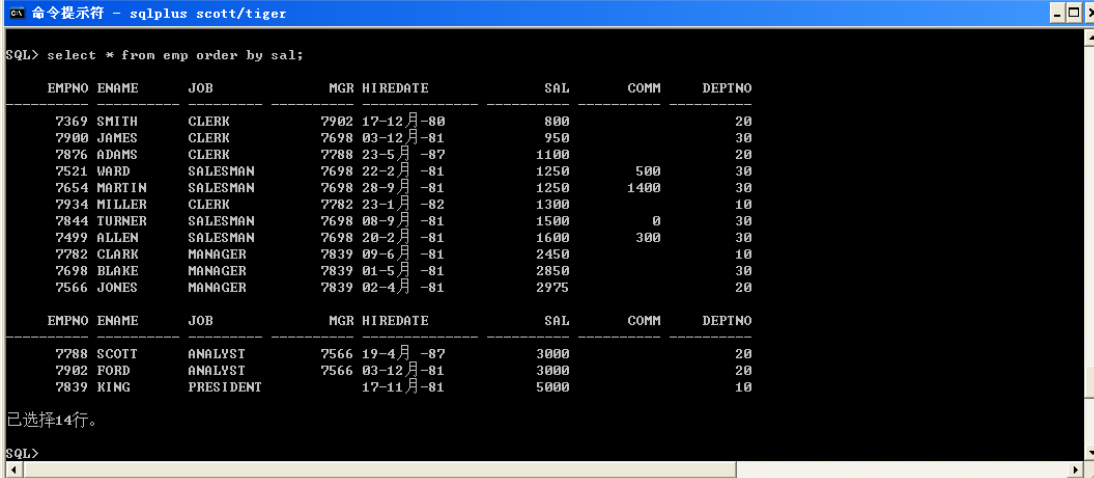
```
Select table_name
From user_tables
Where table_name like '%@_%' escape '@';
```

## 第5章 排序

### 5.1 单一字段排序

- 按照薪水由小到大排序

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



EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-12月-80	800		20
7900	JAMES	CLERK	7698	03-12月-81	950		30
7876	ADAMS	CLERK	7788	23-5月-87	1100		20
7521	WARD	SALESMAN	7698	22-2月-81	1250	500	30
7654	MARTIN	SALESMAN	7698	28-9月-81	1250	1400	30
7934	MILLER	CLERK	7782	23-1月-82	1300		10
7844	TURNER	SALESMAN	7698	08-9月-81	1500	0	30
7499	ALLEN	SALESMAN	7698	20-2月-81	1600	300	30
7782	CLARK	MANAGER	7839	09-6月-81	2450		10
7698	BLAKE	MANAGER	7839	01-5月-81	2850		30
7566	JONES	MANAGER	7839	02-4月-81	2975		20
7788	SCOTT	ANALYST	7566	19-4月-87	3000		20
7902	FORD	ANALYST	7566	03-12月-81	3000		20
7839	KING	PRESIDENT		17-11月-81	5000		10

已选择14行。

排序采用 order by 子句，order by 后面跟上排序字段，排序字段可以放多个，多个采用逗号间隔，order by 默认采用升序，如果存在 where 子句那么 order by 必须放到 where 语句的后面，错误方式

```
select * from emp order by sal where sal > 1500;
```

正确的方式

```
select * from emp where sal > 1500 order by sal;
```

### 5.2 手动指定排序顺序

- 手动指定按照薪水由小到大排序

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



```
命令提示符 - sqlplus scott/tiger

SQL> select * from emp order by sal asc;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-12月-80	800		20
7900	JAMES	CLERK	7698	03-12月-81	950		30
7876	ADAMS	CLERK	7788	23-5月-87	1100		20
7521	WARD	SALESMAN	7698	22-2月-81	1250	500	30
7654	MARTIN	SALESMAN	7698	28-9月-81	1250	1400	30
7934	MILLER	CLERK	7782	23-1月-82	1300		10
7844	TURNER	SALESMAN	7698	08-9月-81	1500	0	30
7499	ALLEN	SALESMAN	7698	20-2月-81	1600	300	30
7782	CLARK	MANAGER	7839	09-6月-81	2450		10
7698	BLAKE	MANAGER	7839	01-5月-81	2850		30
7566	JONES	MANAGER	7839	02-4月-81	2975		20
已选择14行。							
7788	SCOTT	ANALYST	7566	19-4月-87	3000		20
7902	FORD	ANALYST	7566	03-12月-81	3000		20
7839	KING	PRESIDENT		17-11月-81	5000		10

```
SQL>
```

- 手动指定按照薪水由大到小排序

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

```
命令提示符 - sqlplus scott/tiger

SQL> select * from emp order by sal desc;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		17-11月-81	5000		10
7902	FORD	ANALYST	7566	03-12月-81	3000		20
7788	SCOTT	ANALYST	7566	19-4月-87	3000		20
7566	JONES	MANAGER	7839	02-4月-81	2975		20
7698	BLAKE	MANAGER	7839	01-5月-81	2850		30
7782	CLARK	MANAGER	7839	09-6月-81	2450		10
7499	ALLEN	SALESMAN	7698	20-2月-81	1600	300	30
7844	TURNER	SALESMAN	7698	08-9月-81	1500	0	30
7934	MILLER	CLERK	7782	23-1月-82	1300		10
7521	WARD	SALESMAN	7698	22-2月-81	1250	500	30
7654	MARTIN	SALESMAN	7698	28-9月-81	1250	1400	30
已选择14行。							
7876	ADAMS	CLERK	7788	23-5月-87	1100		20
7900	JAMES	CLERK	7698	03-12月-81	950		30
7369	SMITH	CLERK	7902	17-12月-80	800		20

```
SQL>
```

### 5.3 多个字段排序(按照员工薪资的降序排列,如果薪资一样,再按照员工名字的将序排列。)

- 按照薪水和姓名倒序

```
select * from emp order by sal desc ,ename desc;
```

```

SQL> select * from emp order by sal desc ,ename desc;

  EMPNO  ENAME      JOB              MGR HIREDATE          SAL     COMM     DEPTNO
-----
   7839  KING      PRESIDENT              17-11月-81      5000             10
   7788  SCOTT     ANALYST               7566 19-4月-87       3000             20
   7902  FORD      ANALYST               7566 03-12月-81      3000             20
   7566  JONES     MANAGER               7839 02-4月-81       2975             20
   7698  BLAKE     MANAGER               7839 01-5月-81       2850             30
   7782  CLARK     MANAGER               7839 09-6月-81       2450             10
   7499  ALLEN     SALESMAN              7698 20-2月-81       1600             30
   7844  TURNER    SALESMAN              7698 08-9月-81       1500              0
   7934  MILLER    CLERK                  7782 23-1月-82       1300             10
   7521  WARD      SALESMAN              7698 22-2月-81       1250             30
   7654  MARTIN    SALESMAN              7698 28-9月-81       1250            1400

  EMPNO  ENAME      JOB              MGR HIREDATE          SAL     COMM     DEPTNO
-----
   7876  ADAMS     CLERK                  7788 23-5月-87       1100             20
   7900  JAMES     CLERK                  7698 03-12月-81       950              30
   7369  SMITH     CLERK                  7902 17-12月-80       800              20

已选择14行。

SQL>

```

如果采用多个字段排序，如果根据第一个字段排序重复了，会根据第二个字段排序

## 5.4 使用字段的位置来排序

- 按照薪水升序

```
select * from emp order by 6;
```

```

SQL> select * from emp order by 6;

  EMPNO  ENAME      JOB              MGR HIREDATE          SAL     COMM     DEPTNO
-----
   7369  SMITH     CLERK                  7902 17-12月-80       800              20
   7900  JAMES     CLERK                  7698 03-12月-81       950              30
   7876  ADAMS     CLERK                  7788 23-5月-87       1100             20
   7521  WARD      SALESMAN              7698 22-2月-81       1250             30
   7654  MARTIN    SALESMAN              7698 28-9月-81       1250            1400
   7934  MILLER    CLERK                  7782 23-1月-82       1300             10
   7844  TURNER    SALESMAN              7698 08-9月-81       1500              0
   7499  ALLEN     SALESMAN              7698 20-2月-81       1600             30
   7782  CLARK     MANAGER               7839 09-6月-81       2450             10
   7698  BLAKE     MANAGER               7839 01-5月-81       2850             30
   7566  JONES     MANAGER               7839 02-4月-81       2975             20

  EMPNO  ENAME      JOB              MGR HIREDATE          SAL     COMM     DEPTNO
-----
   7788  SCOTT     ANALYST               7566 19-4月-87       3000             20
   7902  FORD      ANALYST               7566 03-12月-81      3000             20
   7839  KING      PRESIDENT              17-11月-81      5000             10

已选择14行。

SQL>

```

不建议使用此种方式，采用数字含义不明确，程序不健壮

## 第6章 单行函数

### 6.1 lower

- 查询员工，将员工姓名全部转换成小写

```
select lower(ename) from emp;
```

```

SQL> select lower(ename) from emp;

LOWER(ENAME)
-----
smith
allen
ward
jones
martin
blake
clark
scott
king
turner
adams

LOWER(ENAME)
-----
james
ford
miller

已选择14行。

SQL>

```

## 6.2 upper

- 查询 job 为 manager 的员工

```
select * from emp where job=upper('manager');
```

```

SQL> select * from emp where job=upper('manager');

EMPNO  ENAME      JOB            MGR HIREDATE          SAL      COMM      DEPTNO
-----
7566 JONES      MANAGER        7839 02-4月 -81      2975      20
7698 BLAKE      MANAGER        7839 01-5月 -81      2850      30
7782 CLARK      MANAGER        7839 09-6月 -81      2450      10

SQL>

```

## 6.3 substr

- 查询姓名以 M 开头所有的员工
- 方法的第二个参数表示的是查询字符的位置，0,1 都表示第一个字符，负数表示从结尾开始的位置，第三个参数表示截取字符串的长度。
- Substr('被截取的字符串',从哪一位开始截取,截取几位);
  - a) 从哪一位开始截取，有正数也有负数，正数表示从左边开始数。负数表示从右边开始数。截取的时候一定是从左向右截取。

```
select * from emp where substr(ename, 1,1)='M';
```

```

SQL> select * from emp where substr(ename, 1,1)='M';

EMPNO  ENAME      JOB            MGR HIREDATE          SAL      COMM      DEPTNO
-----
7654 MARTIN    SALESMAN       7698 28-9月 -81      1250      1400      30
7934 MILLER    CLERK          7782 23-1月 -82      1300      10

SQL> select * from emp where upper(substr(ename, 1,1))='M';

EMPNO  ENAME      JOB            MGR HIREDATE          SAL      COMM      DEPTNO
-----
7654 MARTIN    SALESMAN       7698 28-9月 -81      1250      1400      30
7934 MILLER    CLERK          7782 23-1月 -82      1300      10

SQL>

```

## 6.4 length

- 取得员工姓名的长度

```
select length(ename) from emp;
```

```

SQL> select length(ename) from emp;

LENGTH(ENAME)
-----
5
5
4
5
6
5
5
5
4
6
5

LENGTH(ENAME)
-----
5
4
6

已选择14行。
SQL>

```

## 6.5 trim

trim 会去首尾空格，不会去除中间的空格

- 取得工作岗位为 MANAGER 的所有员工

```
select * from emp where job=trim('MANAGER');
```

```

SQL> select * from emp where job=trim('MANAGER ');

EMPNO  ENAME      JOB      MGR HIREDATE          SAL      COMM      DEPTNO
-----
7566 JONES      MANAGER  7839 02-4月 -81      2975             20
7698 BLAKE      MANAGER  7839 01-5月 -81      2850             30
7782 CLARK      MANAGER  7839 09-6月 -81      2450             10

SQL>

```

## 6.6 to\_date

- 查询 1981-02-20 入职的员工（第一种方法，与数据库的格式匹配上）

```
select * from emp where HIREDATE='20-2月 -81';
```

- 查询 1981-02-20 入职的员工（第二种方法，将字符串转换成 date 类型）

```
select * from emp where hiredate=to_date('1981-02-20 00:00:00', 'YYYY-MM-DD HH24:MI:SS');
```

```

SQL> select * from emp where hiredate=to_date('1981-02-20 00:00:00', 'YYYY-MM-DD HH24:MI:SS');

EMPNO  ENAME      JOB      MGR HIREDATE          SAL      COMM      DEPTNO
-----
7499 ALLEN      SALESMAN  7698 20-2月 -81      1600             30

SQL>

```

to\_date 可以将字符串转换成日期，具体格式 to\_date(字符串, 匹配格式)

## 日期格式的说明

控制符	说明
YYYY	表示年
MM	表示月
DD	表示日
HH12,HH24	表示 12 小时制, 表示 24 小时制
MI	表示分
SS	表示秒

说明: Oracle 默认日期格式是: DD-MON-YY

Insert into customer(birth) values('11-10 月-85'); //系统默认调用 to\_date 函数

Insert into customer(birth) values(to\_date('11-10 月-85','DD-MON-YY'));

Select birth from customer; //系统默认调用 to\_char

Select to\_char(birth,'DD-MON-YY') birth from customer;

获取 ORACLE 系统当前日期:

Select to\_char(sysdate,'YYYY-MM-DD HH24:MI:SS') nowTime from dual;

Select sysdate from dual; //默认调用系统的 to\_char 函数

如何修改 ORACLE 的默认日期格式? (DDL)

alter session set nls\_date\_format = 'YYYY-MON-DD HH24:MI:SS';

## 6.7 to\_char

- 查询 1981-02-20 以后入职的员工, 将入职日期格式化成 yyyy-mm-dd hh:mm:ss

```
select empno, ename, to_char(hiredate, 'yyyy-mm-dd hh24:mi:ss') from emp where
hiredate>to_date('1981-02-20 00:00:00', 'YYYY-MM-DD HH24:MI:SS');
```

```
SQL> select empno, ename, to_char(hiredate, 'yyyy-mm-dd hh24:mi:ss') from emp where hiredate>to_date('1981-02-20 00:00:00', 'YYYY-MM-DD HH24:MI:SS');
```

EMPNO	ENAME	TO_CHAR(HIREDATE, 'Y
7521	WARD	1981-02-22 00:00:00
7566	JONES	1981-04-02 00:00:00
7654	MARTIN	1981-09-28 00:00:00
7698	BLAKE	1981-05-01 00:00:00
7782	CLARK	1981-06-09 00:00:00
7788	SCOTT	1987-04-19 00:00:00
7839	KING	1981-11-17 00:00:00
7844	TURNER	1981-09-08 00:00:00
7876	ADAMS	1987-05-23 00:00:00
7900	JAMES	1981-12-03 00:00:00
7902	FORD	1981-12-03 00:00:00
7934	MILLER	1982-01-23 00:00:00

已选择12行。

```
SQL>
```

- 查询员工薪水加入千分位

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

```
SQL> select empno, ename, to_char(sal, '$999,999') from emp;
```

EMPNO	ENAME	TO_CHAR(S)
7369	SMITH	\$800
7499	ALLEN	\$1,600
7521	WARD	\$1,250
7566	JONES	\$2,975
7654	MARTIN	\$1,250
7698	BLAKE	\$2,850
7782	CLARK	\$2,450
7788	SCOTT	\$3,000
7839	KING	\$5,000
7844	TURNER	\$1,500
7876	ADAMS	\$1,100
7900	JAMES	\$950
7902	FORD	\$3,000
7934	MILLER	\$1,300

已选择14行。

- 查询员工薪水加入千分位和保留两位小数

```
select empno, ename, to_char(sal, '$999,999.00') from emp;
```

```
SQL> select empno, ename, to_char(sal, '$999,999.00') from emp;
```

EMPNO	ENAME	TO_CHAR(SAL)
7369	SMITH	\$800.00
7499	ALLEN	\$1,600.00
7521	WARD	\$1,250.00
7566	JONES	\$2,975.00
7654	MARTIN	\$1,250.00
7698	BLAKE	\$2,850.00
7782	CLARK	\$2,450.00
7788	SCOTT	\$3,000.00
7839	KING	\$5,000.00
7844	TURNER	\$1,500.00
7876	ADAMS	\$1,100.00
7900	JAMES	\$950.00
7902	FORD	\$3,000.00
7934	MILLER	\$1,300.00

已选择14行。

将数字转换成字符串，格式

控制符	说明
9	表示一位数字
0	位数不够可以补零
\$	美元符
L	本地货币符号
.	显示小数
,	显示千分位

## 6.8 to\_number

将字符串转换成数值

```
select * from emp where sal > to_number('1,500', '999,999');
```

```
SQL> select * from emp where sal>to_number('1,500', '999,999');
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7499	ALLEN	SALESMAN	7698	20-2月-81	1600	300	30
7566	JONES	MANAGER	7839	02-4月-81	2975		20
7698	BLAKE	MANAGER	7839	01-5月-81	2850		30
7782	CLARK	MANAGER	7839	09-6月-81	2450		10
7788	SCOTT	ANALYST	7566	19-4月-87	3000		20
7839	KING	PRESIDENT		17-11月-81	5000		10
7902	FORD	ANALYST	7566	03-12月-81	3000		20

已选择7行。

## 6.9 nvl (oracle 中将空值当做无穷大)

- 取得员工的全部薪水，薪水+津贴

```
select empno, ename, sal, comm, sal+comm from emp;
```

```
SQL> select empno, ename, sal, comm, sal+comm from emp;
```

EMPNO	ENAME	SAL	COMM	SAL+COMM
7369	SMITH	800		
7499	ALLEN	1600	300	1900
7521	WARD	1250	500	1750
7566	JONES	2975		
7654	MARTIN	1250	1400	2650
7698	BLAKE	2850		
7782	CLARK	2450		
7788	SCOTT	3000		
7839	KING	5000		
7844	TURNER	1500	0	1500
7876	ADAMS	1100		
7900	JAMES	950		
7902	FORD	3000		
7934	MILLER	1300		

已选择14行。

以上结果不正确，主要原因是津贴（comm）字段为 null，所以无法计算，所以正确的做法是将津贴先转换成 0，再计算。可以使用 Oracle 提供的 nvl，该函数的语法格式为：nvl(表达式 1，表达式 2)，表达式 1：指的是字段名称；表达式 2：指的是将该字段的 null 转换成的值

- 采用 nvl 函数，取得员工的全部薪水，薪水+津贴

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

```

SQL> select empno, ename, sal, comm, sal+nvl(comm,0) from emp;

  EMPNO  ENAME      SAL       COMM  SAL+NVL(COMM,0)
-----
   7369  SMITH          800         0         800
   7499  ALLEN         1600        300        1900
   7521  WARD          1250        500        1750
   7566  JONES         2975         0         2975
   7654  MARTIN        1250       1400        2650
   7698  BLAKE         2850         0         2850
   7782  CLARK         2450         0         2450
   7788  SCOTT         3000         0         3000
   7839  KING          5000         0         5000
   7844  TURNER        1500         0         1500
   7876  ADAMS         1100         0         1100

  EMPNO  ENAME      SAL       COMM  SAL+NVL(COMM,0)
-----
   7900  JAMES          950         0          950
   7902  FORD          3000         0         3000
   7934  MILLER        1300         0         1300

已选择14行。
SQL>

```

以上结果是正确的，在做表设计的时候，关于数值字段最好不允许为 null，可以设置缺省值

## 6.10 case ... when ... then ...when...then...end

- 如果 job 为 MANAGER 薪水上涨 10%,如果 job 为 SALESMAN 工资上涨 50%

```
select empno, ename, job, sal, (case job when 'MANAGER' then sal*1.1 when 'SALESMAN' then
sal*1.5 end) as newsal from emp;
```

```

SQL> select empno, ename, job, sal, case job when 'MANAGER' then sal*1.1 when 'SALESMAN' then sal*1.5 end as newsal from emp;

  EMPNO  ENAME      JOB       SAL       NEWSAL
-----
   7369  SMITH      CLERK          800         800
   7499  ALLEN     SALESMAN       1600        2400
   7521  WARD     SALESMAN       1250        1875
   7566  JONES     MANAGER       2975        3272.5
   7654  MARTIN   SALESMAN       1250        1875
   7698  BLAKE     MANAGER       2850        3135
   7782  CLARK     MANAGER       2450        2695
   7788  SCOTT    ANALYST       3000
   7839  KING    PRESIDENT       5000
   7844  TURNER   SALESMAN       1500        2250
   7876  ADAMS     CLERK          1100
   7900  JAMES     CLERK           950
   7902  FORD    ANALYST       3000
   7934  MILLER     CLERK         1300

已选择14行。
SQL>

```

## 6.11 decode

同 case ...when ...then ... end

- 如果 job 为 MANAGER 薪水上涨 10%,如果 job 为 SALESMAN 工资上涨 50%

```
select empno, ename, job, sal, decode(job, 'MANAGER', SAL*1.1, 'SALESMAN', sal*1.5) as newsal
from emp;
```



```

SQL> select empno, ename, job, sal, decode(job, 'MANAGER', sal*1.1, 'SALESMAN', sal*1.5) as newsal from emp;

   EMPNO  ENAME      JOB              SAL      NEWSAL
-----
   7369  SMITH        CLERK              800         2400
   7499  ALLEN        SALESMAN          1600         2400
   7521  WARD         SALESMAN          1250         1875
   7566  JONES        MANAGER           2975         3272.5
   7654  MARTIN       SALESMAN          1250         1875
   7698  BLAKE        MANAGER           2850         3135
   7782  CLARK        MANAGER           2450         2695
   7788  SCOTT        ANALYST           3000
   7839  KING         PRESIDENT         5000
   7844  TURNER       SALESMAN          1500         2250
   7876  ADAMS        CLERK              1100
   7900  JAMES        CLERK               950
   7902  FORD         ANALYST           3000
   7934  MILLER       CLERK              1300
  
```

## 6.12 round

四舍五入

```
select round(1234567.4567, 2) from dual;
```

```

SQL> select round(1234567.4567, 2) from dual;

ROUND(1234567.4567,2)
-----
1234567.46

SQL> desc dual;
 名称                是否为空? 类型
-----
DUMMY                UARCHAR2(1)
  
```

Dual 是 oracle 提供的，主要为了方便使用，因为 select 的时候需要用 from

单行函数：一个输入对应一个输出

多行函数(聚合函数)：多个输入对应一个输出

## 第7章 分组函数及分组查询

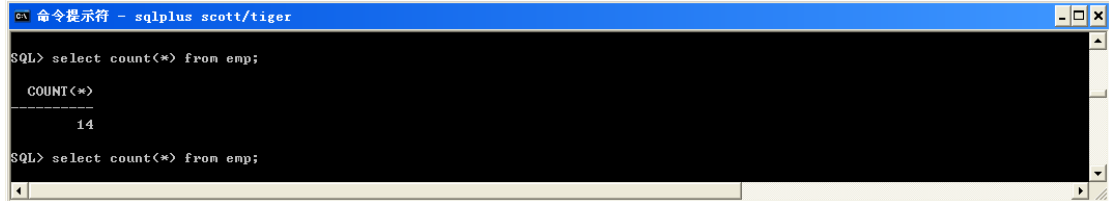
### 7.1 分组函数

count(comm)	count(*)	取得记录数
sum		求和
avg		取平均
max		取最大的数
min		取最小的数

### 7.1.1 count

- 取得所有的员工数

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

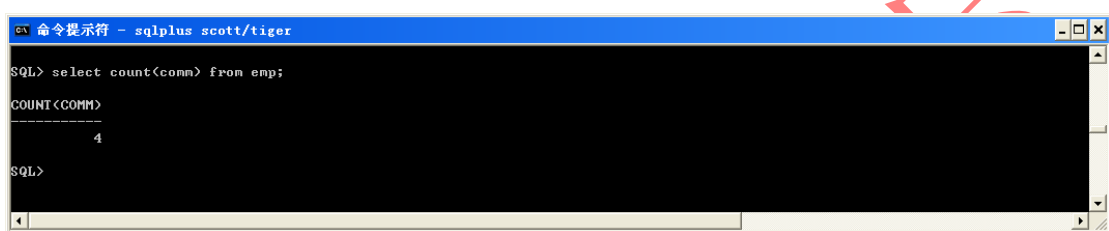


```
命令提示符 - sqlplus scott/tiger
SQL> select count(*) from emp;
COUNT(*)
         14
SQL> select count(*) from emp;
```

Count (\*) 表示取得所有记录，忽略 null，为 null 值也会取得

- 取得津贴不为 null 员工数

```
select count(comm) from emp;
```

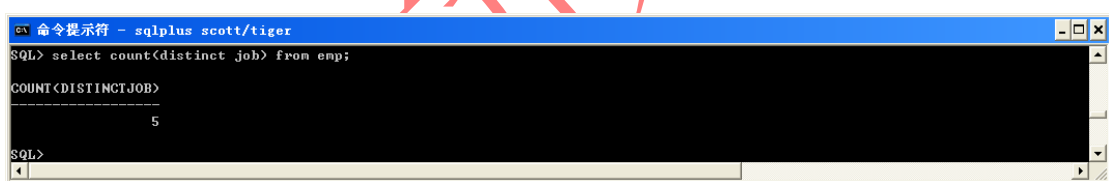


```
命令提示符 - sqlplus scott/tiger
SQL> select count(comm) from emp;
COUNT(COMM)
            4
SQL>
```

采用 count(字段名称)，不会取得为 null 的记录

- 取得工作岗位的个数

```
select count(distinct job) from emp;
```



```
命令提示符 - sqlplus scott/tiger
SQL> select count(distinct job) from emp;
COUNT(DISTINCT JOB)
                    5
SQL>
```

Distinct 可以去除重复的记录

联合去重：

Select distinct job,deptno from emp; (job 和 deptno 联合去重)

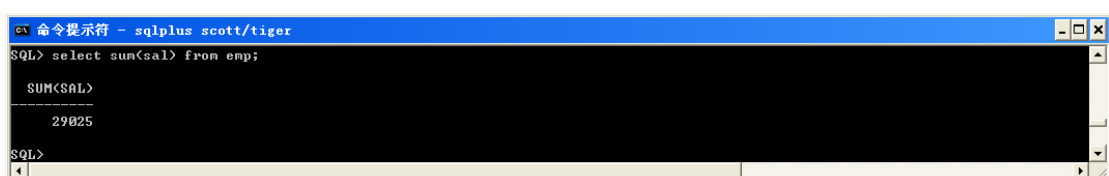
Distinct 关键字必须出现在所有字段的前面。

### 7.1.2 sum

Sum 可以取得某一个列的和，如果是 null 会略

- 取得薪水的合计

```
select sum(sal) from emp;
```



```
命令提示符 - sqlplus scott/tiger
SQL> select sum(sal) from emp;
SUM(SAL)
      29025
SQL>
```

- 取得薪水的合计 (sal+comm)

```
select sum(sal+comm) from emp;
```

```

命令提示符 - sqlplus scott/tiger

SQL> select sum(sal+comm) from emp;

SUM(SAL+COMM)
-----
       7800
    
```

从以上结果来看，不正确，原因在于 comm 字段有 null 值，所以无法计算，sum 会忽略掉，正确的做法是将 comm 字段转换成 0

```
select sum(sal+nvl(comm, 0)) from emp;
```

```

命令提示符 - sqlplus scott/tiger

SQL> select sum(sal+nvl(comm, 0)) from emp;

SUM(SAL+NVL(COMM,0))
-----
       31225

SQL>
    
```

### 7.1.3 avg

取得某一列的平均值

- 取得平均薪水

```
select avg(sal) from emp;
```

```

命令提示符 - sqlplus scott/tiger

SQL> select avg(sal) from emp;

AVG(SAL)
-----
2073.21429

SQL>
    
```

### 7.1.4 max

取得某个一列的最大值

- 取得最高薪水

```
select max(sal) from emp;
```

```

命令提示符 - sqlplus scott/tiger

SQL> select max(sal) from emp;

MAX(SAL)
-----
       5000

SQL>
    
```

- 取得最晚入职得员工

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

```
命令提示符 - sqlplus scott/tiger

SQL> select max(to_char(hiredate, 'yyyy-mm-dd')) from emp;

MAX(TO_CHAR
1987-05-23
SQL>
```

### 7.1.5 min

取得某个一列的最小值

- 取得最低薪水

```
select min(sal) from emp;
```

```
命令提示符 - sqlplus scott/tiger

SQL> select min(sal) from emp;

MIN(SAL)
-----
800
SQL>
```

- 取得最早入职得员工

```
select min(hiredate) from emp;
```

```
命令提示符 - sqlplus scott/tiger

SQL> select min(hiredate) from emp;

MIN(HIREDATE)
-----
17-12月-80
SQL>
```

### 7.1.6 组合分组函数

可以将这些分组函数都放到 select 中一起使用

```
select count(*), sum(sal), avg(sal), max(sal), min(sal) from emp;
```

```
命令提示符 - sqlplus scott/tiger

SQL> select count(*), sum(sal), avg(sal), max(sal), min(sal) from emp;

COUNT(*)  SUM(SAL)  AVG(SAL)  MAX(SAL)  MIN(SAL)
-----
14         29025  2073.21429  5000      800
SQL>
```

## 7.2 分组查询

分组查询主要涉及到两个子句，分别是：group by 和 having

## 7.2.1 group by

- 取得每个工作岗位的工资合计，要求显示岗位名称和工资合计

```
select job, sum(sal) from emp group by job;
```

```
SQL> select job, sum(sal) from emp group by job;

JOB                SUM(SAL)
-----
CLERK               4150
SALESMAN            5600
PRESIDENT           5000
MANAGER             8275
ANALYST             6000

SQL>
```

采用 group by，非聚合函数所使用的字段必须参与分组

在 select 语句中，如果有 group by 语句，那么 select 后面只能跟参加分组的字段和分组函数。

如果使用了 order by，order by 必须放到 group by 后面

```
SQL> select job, sum(sal) from emp order by job group by job;

ORA-00933: SQL 命令未正确结束
```

- 按照工作岗位和部门编码分组，取得的工资合计

### ■ 原始数据

```
SQL> select * from emp;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-12 月 -80	800		20
7499	ALLEN	SALESMAN	7698	20-2 月 -81	1600	300	30
7521	WARD	SALESMAN	7698	22-2 月 -81	1250	500	30
7566	JONES	MANAGER	7839	02-4 月 -81	2975		30
7654	MARTIN	SALESMAN	7698	28-9 月 -81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-5 月 -81	2850		30
7782	CLARK	MANAGER	7839	09-6 月 -81	2450		10
7788	SCOTT	ANALYST	7566	19-4 月 -87	3000		20
7839	KING	PRESIDENT		17-11 月 -81	5000		10
7844	TURNER	SALESMAN	7698	08-9 月 -81	1500	0	30
7876	ADAMS	CLERK	7788	23-5 月 -87	1100		20
7900	JAMES	CLERK	7698	03-12 月 -81	950		30
7902	FORD	ANALYST	7566	03-12 月 -81	3000		20
7934	MILLER	CLERK	7782	23-1 月 -82	1300		10

已选择 14 行。

### ■ 分组语句

```
select job,deptno, sum(sal) from emp group by job,deptno;
```

分组后的数据

```
SQL> select job,deptno, sum(sal) from emp group by job,deptno;
```

JOB	DEPTNO	SUM(SAL)
PRESIDENT	10	5000
CLERK	10	1300
SALESMAN	30	5600
ANALYST	20	6000
MANAGER	30	5825
MANAGER	10	2450
CLERK	30	950
CLERK	20	1900

已选择 8 行。

Group by 中不能使用聚合函数

## 7.2.2 having(对分组之后的数据进行过滤)

如果想对分组数据再进行过滤需要使用 having 子句  
取得每个岗位的平均工资大于 2000

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

```
SQL> select job, avg(sal) from emp group by job having avg(sal) >2000;
```

JOB	AUG(SAL)
PRESIDENT	5000
MANAGER	2758.3333
ANALYST	3000

分组函数的执行顺序:

- 1、根据条件查询数据
- 2、分组
- 3、采用 having 过滤，取得正确的数据

原则：可以在 where 语句中过滤的数据，不要使用 having 过滤。

## 7.2.3 select 语句总结

一个完整的 select 语句格式如下

```
select 字段
from 表名
where .....
group by .....
```

having .....

order by .....

以上语句的执行顺序

首先执行 **where** 语句过滤原始数据

执行 **group by** 进行分组

执行 **having** 对分组数据进行操作

执行 **select** 选出数据

执行 **order by** 排序

## 第8章 连接查询

### 8.1 什么是连接查询

我们现在使用的数据库是关系型数据库，表和表之间存在关联关系，通常的业务中要求我们多张表联合起来取得有效数据，这种多张表联合查询被称作连接查询。(从单张表中取数据的情况比较少)

### 8.2 连接查询的分类：

按照连接查询语法出现的年代分：

SQL92

SQL99(重点掌握)

按照连接查询的连接方式分：

内连接：等值连接、非等值连接、自连接

内连接：**a** 和 **b** 两张表进行连接查询，只查询两张表能够完全匹配的记录，这种查询叫做内连接

外连接：左(外)连接、右(外)连接

外连接：在内连接的(完全匹配的)基础之上，将其中一张表的记录完全展示，另一张表肯定会有一些记录无法与其匹配，此时会自动模拟出空值与其匹配。这种连接查询叫做外连接。

### 8.3 分析：两张表连接查询的时候如果没有条件限制会出现什么现象？

案例：查询每一个员工所在的部门名称。要求显示员工名和对应的部门名称。

SQL> select ename,deptno from emp; (e 表)

ENAME	DEPTNO
SMITH	20
ALLEN	30
WARD	30
JONES	20
MARTIN	30
BLAKE	30
CLARK	10
SCOTT	20
KING	10
TURNER	30
ADAMS	20
JAMES	30
FORD	20
MILLER	10

SQL> select \* from dept; (d 表)

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

select e.ename,d.dname from emp e,dept d;

结论：两张表连接查询如果没有条件限制，会进行任意匹配，查询结果条数是两张表记录条数的乘积，这种现象叫做笛卡尔积现象。



## 8.4 如何避免笛卡尔积现象的发生

在查询的时候添加条件进行限制。查询匹配的次数并没有减少

案例：查询每一个员工所在的部门名称。要求显示员工名和对应的部门名称。

SQL> select ename,deptno from emp; (e 表)

ENAME	DEPTNO
SMITH	20
ALLEN	30
WARD	30
JONES	20
MARTIN	30
BLAKE	30
CLARK	10
SCOTT	20
KING	10
TURNER	30
ADAMS	20
JAMES	30
FORD	20
MILLER	10

SQL> select \* from dept; (d 表)

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

SQL92 语法: (内连接中的等值连接)

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

SQL99 语法: (内连接中的等值连接)

```
select e.ename,d.dname from emp e inner join dept d on e.deptno=d.deptno;
```

```
select e.ename,d.dname from emp e join dept d on e.deptno=d.deptno; //inner 可以省略
```

**Sql92 语法和 sql99 语法的区别：**99 语法可以做到表的连接条件和查询条件分离，特别是多个表进行连接的时候，会比 sql92 更清晰

ENAME	DNAME
SMITH	RESEARCH
ALLEN	SALES
WARD	SALES
JONES	RESEARCH
MARTIN	SALES
BLAKE	SALES
CLARK	ACCOUNTING
SCOTT	RESEARCH
KING	ACCOUNTING
TURNER	SALES
ADAMS	RESEARCH

ENAME	DNAME
JAMES	SALES
FORD	RESEARCH
MILLER	ACCOUNTING

## 8.5 案例：查询每一个员工的工资等级,要求显示员工的薪水，以及对应的等级

SQL> select ename,sal from emp;

ENAME	SAL
SMITH	800
ALLEN	1600
WARD	1250
JONES	2975
MARTIN	1250
BLAKE	2850
CLARK	2450
SCOTT	3000
KING	5000
TURNER	1500
ADAMS	1100
JAMES	950

FORD	3000
MILLER	1300

GRADE	LOSAL	HISAL
1	700	1200
2	1201	1400
3	1401	2000
4	2001	3000
5	3001	9999

SQL92 语法: (内连接中的非等值连接)

select e.ename,e.sal,s.grade from emp e,salgrade s where e.sal between s.losal and s.hisal;

SQL99 语法: (内连接中的非等值连接)

select e.ename,e.sal,s.grade from emp e inner join salgrade s on e.sal between s.losal and s.hisal;

select e.ename,e.sal,s.grade from emp e join salgrade s on e.sal between s.losal and s.hisal;

//inner 可以省略

## 8.6 案例: 查询出每一个员工的上级领导, 要求显示员工姓名以及对应的领导名称。

SQL> select empno,ename,mgr from emp;

(emp a 表: 员工表)

EMPNO	ENAME	MGR
7369	SMITH	7902
7499	ALLEN	7698
7521	WARD	7698
7566	JONES	7839
7654	MARTIN	7698
7698	BLAKE	7839
7782	CLARK	7839
7788	SCOTT	7566
7839	KING	
7844	TURNER	7698
7876	ADAMS	7788
7900	JAMES	7698
7902	FORD	7566
7934	MILLER	7782

(emp b 表: 领导表)

```
EMPNO ENAME
-----
7566 JONES
7698 BLAKE
7782 CLARK
7788 SCOTT
7839 KING
7902 FORD
```

SQL92 语法: (内连接中的自连接)

```
select a.ename 员工,b.ename 领导 from emp a , emp b where a.mgr=b.empno;
```

SQL99 语法: (内连接中的自连接)

```
select a.ename 员工,b.ename 领导 from emp a inner join emp b on a.mgr=b.empno;
```

```
select a.ename 员工,b.ename 领导 from emp a join emp b on a.mgr=b.empno;
```

员工	领导
FORD	JONES
SCOTT	JONES
JAMES	BLAKE
TURNER	BLAKE
MARTIN	BLAKE
WARD	BLAKE
ALLEN	BLAKE
MILLER	CLARK
ADAMS	SCOTT
CLARK	KING
BLAKE	KING
JONES	KING
SMITH	FORD

## 8.7 案例: 查询员工所在的部门, 要求显示员工名和对应的部门名称 (要求部门名称全部显示)

SQL92 语法: (外连接中的右(外)连接)

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

SQL92 语法: (外连接中的左(外)连接)

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

SQL99 语法: (外连接中的右(外)连接)

```
select e.ename,d.dname from emp e right outer join dept d on e.deptno=d.deptno;
```

```
select e.ename,d.dname from emp e right join dept d on e.deptno=d.deptno; //outer 可以省略
```

SQL99 语法: (外连接中的左(外)连接)

```
select e.ename,d.dname from dept d left outer join emp e on e.deptno=d.deptno;
```

```
select e.ename,d.dname from dept d left join emp e on e.deptno=d.deptno; //outer 可以省略
```

任何一个左外连接都有对应的右外连接。

ENAME	DNAME
SMITH	RESEARCH
ALLEN	SALES
WARD	SALES
JONES	RESEARCH
MARTIN	SALES
BLAKE	SALES
CLARK	ACCOUNTING
SCOTT	RESEARCH
KING	ACCOUNTING
TURNER	SALES
ADAMS	RESEARCH
JAMES	SALES
FORD	RESEARCH
MILLER	ACCOUNTING
	OPERATIONS

## 8.8 案例：查询出哪个部门没有员工

第一种方式:

```
SQL> select * from dept where deptno not in(select distinct deptno from emp);
```

DEPTNO	DNAME	LOC
40	OPERATIONS	BOSTON

第二种方式:

SQL> select e.ename,d.dname from dept d left join emp e on e.deptno=d.deptno where e.ename is null;

ENAME	DNAME
-----	
	OPERATIONS

## 8.9 案例：查询出"所有"员工对应的上级领导名称

select a.ename 员工,b.ename 领导 from emp a left join emp b on a.mgr=b.empno;

select a.ename 员工,nvl(b.ename,'这是老板') 领导 from emp a left join emp b on a.mgr=b.empno;

员工	领导
-----	
FORD	JONES
SCOTT	JONES
JAMES	BLAKE
TURNER	BLAKE
MARTIN	BLAKE
WARD	BLAKE
ALLEN	BLAKE
MILLER	CLARK
ADAMS	SCOTT
CLARK	KING
BLAKE	KING
JONES	KING
SMITH	FORD
KING	

员工	领导
-----	
FORD	JONES
SCOTT	JONES
JAMES	BLAKE
TURNER	BLAKE
MARTIN	BLAKE
WARD	BLAKE

ALLEN	BLAKE
MILLER	CLARK
ADAMS	SCOTT
CLARK	KING
BLAKE	KING
JONES	KING
SMITH	FORD
KING	这是老板

## 8.10 三张表如何表连接 a join b join c → a 先和 b 关联, a 再和 c 关联

学生选课

学生表 s  
t\_stu

sid(pk)	sname
1	张三
2	李四
3	王五

课程表 c  
t\_cour

cid(pk)	cname
1	C++
2	.NET
3	Java

学生选课表 sc  
t\_stu\_cour

sid(fk)	cid(fk) (sid 和 cid 是联合主键、复合主键)
1	1
1	2
1	3

2	2
2	3
3	1
3	3

要求：查询出 2 号学生所选课程，要求显示学生姓名以及对应的课程名称

```
select
    s.sname,c.cname
from
    t_stu_cour sc
join
    t_stu s
on
    sc.sid=s.sid
join
    t_cour c
on
    sc.cid=c.cid
where
    s.sid=2;
```

## 第9章 子查询

### 9.1 子查询定义

select 语句中嵌套 select 语句

### 9.2 子查询使用场景

```
select..(select)..
from...(select).
where..(select)..
```



### 9.3 在 where 中使用子查询

查询员工信息，查询哪些人是管理者，要求显示出其员工编号和员工姓名

第一步：

```
SQL> select distinct mgr from emp;
```

MGR
7839
7782
7698
7902
7566
7788

已选择 7 行。

第二步：使用 in，在上面的查询结果的范围中的

```
SQL> select empno,ename from emp where empno in(select distinct mgr from emp);
```

EMPNO	ENAME
7902	FORD
7698	BLAKE
7839	KING
7566	JONES
7788	SCOTT
7782	CLARK

已选择 6 行。

注意：in(有空值可以自动忽略，不需要手动过滤)

注意：not in(不忽略空值，需要手动过滤)

查询员工信息，查询哪些人不是管理者，要求显示出其员工编号和员工姓名

```
SQL> select empno,ename from emp where empno not in(select distinct mgr from emp);
```

未选定行

```
SQL> select empno,ename from emp where empno not in(select distinct mgr from emp where mgr is not null);
```

EMPNO ENAME

-----

```
7844 TURNER
7521 WARD
7654 MARTIN
7499 ALLEN
7934 MILLER
7369 SMITH
7876 ADAMS
7900 JAMES
```

## 9.4 在 from 后面使用子查询(要点: 将子查询当做临时表)

查询各个部门的平均薪水所属等级, 需要显示部门编号, 平均薪水, 等级编号

第一步: 查询各个部门的平均薪水

```
SQL> select deptno,avg(sal) avgсал from emp group by deptno;
```

DEPTNO	AVGSAL
30	1566.66667
20	2175
10	2916.66667

第二步: 将上面的查询结果当做临时表 T, t 表和 salgrade 表连接, 条件: t.avgсал between s.losal and s.hisal

```
select
    t.deptno,t.avgсал,s.grade
from
    (select deptno,avg(sal) avgсал from emp group by deptno) t
join
    salgrade s
on
    t.avgсал between s.losal and s.hisal;
```

DEPTNO	AVGSAL	GRADE
30	1566.66667	3
20	2175	4
10	2916.66667	4

## 9.5 了解(select..(select)..)

## 9.6 总结

子查询和连接查询的取舍：子查询管有多少，最终都是基于一个基表的展现，而连接查询可以显示出多表的信息。

# 第10章 Rownum/rowId/union/minus

## 10.1 rownum

### 10.1.1 什么是 rownum?

rownum 是 Oracle 数据库特有的机制

在 Oracle 数据库中任何一张表都有 rownum 这个字段，它是一个隐含字段。

rownum 为查询结果集维护一个自增的行号，行号从 1 开始，以 1 递增。

### 10.1.2 不同的数据库分页的 SQL 语句

mysql 中使用: limit

Oracle 中使用: rownum

Hibernate 中只要指定了不同的数据库方言(dialect)，就会生成不同的 SQL 语句。

### 10.1.3 分析: rownum 和表中记录是否存在绑定关系

SQL> select ename,rownum from emp;

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

SQL> select ename,rownum from emp where sal>=3000;

ENAME	ROWNUM
SCOTT	1
KING	2
FORD	3

注意：rownum 和表中记录不存在绑定关系

#### 10.1.4 Oracle 数据库中的 rownum 只支持哪些操作？

注意：rownum 在 select 语句执行之后才有值。如何理解：where 条件后的 rownum 只是先指定一个查询条件，执行了 select 语句后，才会从 1 开始为其赋值；所以如果在 where 中指定 rownum>=某值的话，不会有记录。

SQL> select ename,rownum from emp where rownum=2;

未选定行

SQL> select ename,rownum from emp where rownum=1;

ENAME	ROWNUM
SMITH	1

SQL> select ename,rownum from emp where rownum>2;

未选定行

SQL> select ename,rownum from emp where rownum<3;

ENAME	ROWNUM
SMITH	1
ALLEN	2

结论：Oracle 中的 rownum 只支持这些操作：=1、<、<=

### 10.1.5 案例：查询 emp 表的前 5 条记录

select ename,rownum from emp where rownum<=5;

ENAME	ROWNUM
SMITH	1
ALLEN	2
WARD	3
JONES	4
MARTIN	5

### 10.1.6 案例：查询工资排名在前 5 名的员工

第一步：先按照工资降序排列

select ename,sal from emp order by sal desc;

ENAME	SAL
KING	5000
FORD	3000

SCOTT	3000
JONES	2975
BLAKE	2850
CLARK	2450
ALLEN	1600
TURNER	1500
MILLER	1300
WARD	1250
MARTIN	1250
ADAMS	1100
JAMES	950
SMITH	800

第二步：将上面的查询结果当作临时表取前 5 条记录

```
select ename,sal from (select ename,sal from emp order by sal desc) where rownum<=5;
```

ENAME	SAL
-----	
KING	5000
SCOTT	3000
FORD	3000
JONES	2975
BLAKE	2850

错误的写法：select ename,sal from emp where rownum<=5 order by sal desc;(注意 SQL 语句的执行顺序。Rownum 的生成在 select 后，orderby 前)

### 10.1.7 案例：查询工资排名在[3-9]名的员工

第一步：查询工资排名在前 9 名的员工

```
select ename,sal,rownum as linenum from (select ename,sal from emp order by sal desc) where rownum<=9;
```

ENAME	SAL	LINENUM
-----		
KING	5000	1
SCOTT	3000	2
FORD	3000	3
JONES	2975	4

BLAKE	2850	5
CLARK	2450	6
ALLEN	1600	7
TURNER	1500	8
MILLER	1300	9

```
select
    ename,sal,linenum
from
    (select
        ename,sal,rownum as linenum
    from
        (select ename,sal from emp order by sal desc)
    where
        rownum<=9)
where
    linenum>=3;
```

ENAME	SAL	LINENUM
-----		
FORD	3000	3
JONES	2975	4
BLAKE	2850	5
CLARK	2450	6
ALLEN	1600	7
TURNER	1500	8
MILLER	1300	9

### 10.1.8 通用分页 SQL:

每页显示 3 条记录

第 1 页: (0~3]

第 2 页: (3~6]

第 3 页: (6~9]

每页显示 pageSize 条记录

第 pageNo 页: ( (pageNo-1) \* pageSize ~ pageNo \* pageSize ]

通用的:

```
select
  t1.*
from
  (select
    t.*,rownum as linenum
  from
    (业务 SQL) t
  where
    rownum<=(pageNo * pageSize)) t1
where
  linenum>((pageNo-1) * pageSize);
```

## 10.2 Rowid

### 10.2.1 定义

rowid 是 Oracle 数据库特有的，rowid 是表中该行在硬盘上存储的真实的物理地址。

通过 rowid 查询表中记录的时候不需要进行表扫描，直接通过物理地址定位，效率较高。

### 10.2.2 Oracle 查询表中记录的时候有两种方式：

第一种方式：全表扫描

第二种方式：通过索引(Oracle 的索引使用了 Oracle 中的 rowid 机制)

```
SQL> select ename,rowid from emp;
```

ENAME	ROWID
SMITH	AAAM9WAAEAAAAHdAAA
ALLEN	AAAM9WAAEAAAAHdAAB
WARD	AAAM9WAAEAAAAHdAAC
JONES	AAAM9WAAEAAAAHdAAD
MARTIN	AAAM9WAAEAAAAHdAAE
BLAKE	AAAM9WAAEAAAAHdAAF
CLARK	AAAM9WAAEAAAAHdAAG
SCOTT	AAAM9WAAEAAAAHdAAH
KING	AAAM9WAAEAAAAHdAAI
TURNER	AAAM9WAAEAAAAHdAAJ



ADAMS	AAAM9WAAEAAAAHdAAK
JAMES	AAAM9WAAEAAAAHdAAL
FORD	AAAM9WAAEAAAAHdAAM
MILLER	AAAM9WAAEAAAAHdAAN

```
SQL> select ename from emp where rowid='AAAM9WAAEAAAAHdAAH';
```

```
ENAME
```

```
-----
```

```
SCOTT
```

### 10.2.3 面试题：删除表中的重复记录(使用 rowid)

```
t_stu
```

```
name
```

```
-----
```

```
jack
```

```
jack
```

```
jack
```

```
sun
```

```
sun
```

```
sun
```

```
delete from t_stu where rowid not in(select min(rowid) from t_stu group by name);
```

### 10.2.4 union 可以合并集合（相加）

```
select * from emp where job='MANAGER'
union
select * from emp where job='SALESMAN'
```

```

SQL> ed
已写入 file afiedt.buf

 1 select * from emp where job='MANAGER'
 2 union
 3 select * from emp where job='SALESMAN'
SQL> /

      EMPNO ENAME      JOB              MGR HIREDATE          SAL          COMM          DEPTNO
-----
 7499 ALLEN      SALESMAN        7698 20-2月 -81      1600          300           30
 7521 WARD      SALESMAN        7698 22-2月 -81      1250          500           30
 7566 JONES      MANAGER        7839 02-4月 -81      2975          0            20
 7654 MARTIN     SALESMAN        7698 28-9月 -81      1250          1400          30
 7698 BLAKE      MANAGER        7839 01-5月 -81      2850          0            30
 7782 CLARK      MANAGER        7839 09-6月 -81      2450          0            10
 7844 TURNER     SALESMAN        7698 08-9月 -81      1500          0            30

已选择7行。

SQL>

```

Union 在某些场合下相当于 or 或 in，等同于如下代码：

select \* from emp where job in('MANAGER','SALESMAN');

或

select \* from emp where job='MANAGER' or job='SALESMAN';

并不是所有的查询操作都可以使用 union 操作

select \* from emp union select \* from dept

使用 union 操作必须保证多个查询结果结合需要具有相同列数

使用 union 操作保证查询结果的（字段含义，数据类型）相同

## 10.2.5 minus 可以移出集合（相减）

- 查询部门编号为 10 和 20 的，去除薪水大于 2000 的（第一种方法）

select \* from emp where deptno in(10, 20) and sal <=2000;

- 查询部门编号为 10 和 20 的，去除薪水大于 2000 的（第二种方法，使用 minus）

select \* from emp where deptno in(10, 20)

minus

select \* from emp where sal>2000

```

SQL> select * from emp where deptno in(10, 20) and sal <=2000;

      EMPNO ENAME      JOB              MGR HIREDATE          SAL          COMM          DEPTNO
-----
 7369 SMITH      CLERK            7902 17-12月 -80      800          0            20
 7876 ADAMS      CLERK            7788 23-5月 -87      1100         0            20
 7934 MILLER     CLERK            7782 23-1月 -82      1300         0            10

SQL> select * from emp where deptno in(10, 20)
 2 minus
 3 select * from emp where sal>2000;

      EMPNO ENAME      JOB              MGR HIREDATE          SAL          COMM          DEPTNO
-----
 7369 SMITH      CLERK            7902 17-12月 -80      800          0            20
 7876 ADAMS      CLERK            7788 23-5月 -87      1100         0            20
 7934 MILLER     CLERK            7782 23-1月 -82      1300         0            10

SQL> ed
已写入 file afiedt.buf

```

处理方式需要注意的地方可以参考 union

## 第11章 对数据库中表及其它数据库对象的操作

### 11.1 添加、修改和删除表中记录

#### 11.1.1 insert

添加、修改和删除都属于 DML，主要包含的语句：insert、update、delete

- Insert 语法格式

Insert into 表名(字段,.....) values(值,.....)

- 省略字段的插入

insert into emp\_01 values(9999, 'zhangsan', 'MANAGER', NULL, NULL, 200, 100, 10);

```

SQL> insert into emp values(9999, 'zhangsan', 'MANAGER', NULL, NULL, 200, 100, 10);
已创建 1 行。

SQL> SELECT * FROM EMP;

   EMPNO  ENAME      JOB              MGR HIREDATE          SAL     COMM     DEPTNO
-----  -
  9999 zhangsan    MANAGER              7902 17-12月-80         200      100        10
  7369 SMITH      CLERK              7698 20-2月-81         800          0        20
  7499 ALLEN     SALESMAN           7698 20-2月-81        1600      300        30
  7521 WARD      SALESMAN           7698 22-2月-81        1250      500        30
  7566 JONES     MANAGER           7839 02-4月-81        2975          0        20
  7654 MARTIN  SALESMAN           7698 28-9月-81        1250     1400        30
  7698 BLAKE    MANAGER           7839 01-5月-81        2850          0        30
  7782 CLARK    MANAGER           7839 09-6月-81        2450          0        10
  7788 SCOTT    ANALYST           7566 19-4月-87        3000          0        20
  7839 KING      PRESIDENT          7566 17-11月-81        5000          0        10
  7844 TURNER   SALESMAN           7698 08-9月-81        1500          0        30
  7876 ADAMS     CLERK             7788 23-5月-87        1100          0        20
  7900 JAMES     CLERK             7698 03-12月-81         950          0        30
  7902 FORD      ANALYST           7566 03-12月-81        3000          0        20
  7934 MILLER   CLERK             7782 23-1月-82        1300          0        10

已选择15行。
  
```

不建议使用此种方式，因为当数据库表中的字段位置发生改变的时候会影响到 insert 语句

- 指定字段的插入(建议使用此种方式)

SQL> insert into emp\_01(empno, ename, job, mgr, hiredate, sal, comm, deptno) values(9999, 'zhangsan', 'MANAGER', NULL, NULL, 200, 100, 10);

insert into emp\_01(empno, ename, job, mgr, hiredate, sal, comm, deptno) values(9999, 'zhangsan', 'MANAGER', NULL, NULL, 200, 100, 10)

\*

第 1 行出现错误:

ORA-00001: 违反唯一约束条件 (SCOTT.PK\_EMP)

主键不能重复

insert into emp\_01(empno, ename, job, mgr, hiredate, sal, comm, deptno) values(8887, 'zhangsan', 'MANAGER', null, sysdate, 200, 100, 10);

```
insert into emp_01(empno, ename, job, mgr, hiredate, sal, comm, deptno) values(8888,
'zhangsan', 'MANAGER', null, to_date('2001-01-01', 'yyyy-mm-dd'), 200, 100, 10);
```

注意 sysdate 或 to\_date

- 表内容的复制（克隆表只能克隆表的内容，约束无法克隆）

```
create table emp_01 as select * from emp;
```

```
SQL> select * from emp_bak;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
9999	zhangsan	MANAGER			200	100	10
8888	zhangsan	MANAGER		01-1月-01	200	100	10
8887	zhangsan	MANAGER		30-9月-09	200	100	10
7369	SMITH	CLERK	7902	17-12月-80	800		20
7499	ALLEN	SALESMAN	7698	20-2月-81	1600	300	30
7521	WARD	SALESMAN	7698	22-2月-81	1250	500	30
7566	JONES	MANAGER	7839	02-4月-81	2975		20
7654	MARTIN	SALESMAN	7698	28-9月-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-5月-81	2850		30
7782	CLARK	MANAGER	7839	09-6月-81	2450		10
7788	SCOTT	ANALYST	7566	19-4月-87	3000		20
7839	KING	PRESIDENT		17-11月-81	5000		10
7844	TURNER	SALESMAN	7698	08-9月-81	1500	0	30
7876	ADAMS	CLERK	7788	23-5月-87	1100		20
7900	JAMES	CLERK	7698	03-12月-81	950		30
7902	FORD	ANALYST	7566	03-12月-81	3000		20
7934	MILLER	CLERK	7782	23-1月-82	1300		10

已选择17行。

以上的语句会自动创建一张表，将所有数据复制到新表中

- 如何将查询的数据直接放到已经存在的表中，可以使用条件？

```
insert into emp_01 select * from emp where job='MANAGER';
```

```
SQL> insert into emp_bak select * from emp where job='MANAGER';
```

已创建6行。

```
SQL> SELECT * FROM EMP_BAK;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
9999	zhangsan	MANAGER			200	100	10
8888	zhangsan	MANAGER		01-1月-01	200	100	10
8887	zhangsan	MANAGER		30-9月-09	200	100	10
7566	JONES	MANAGER	7839	02-4月-81	2975		20
7698	BLAKE	MANAGER	7839	01-5月-81	2850		30
7782	CLARK	MANAGER	7839	09-6月-81	2450		10

已选择6行。

## 11.1.2 update

可以修改数据，可以根据条件修改数据

- 语法规则：

```
update 表名 set 字段名称 1=需要修改的值 1, 字段名称 2=需要修改的值 2 where .....
```

- 将 job 为 manager 的员工的工资上涨 10%

```
update emp set sal=sal+sal*0.1 where job='MANAGER';
```

## 11.1.3 delete

可以删除数据，可以根据条件删除数据

- 语法规则：

```
Delete from 表名 where .....
```

- 删除津贴为 300 的员工

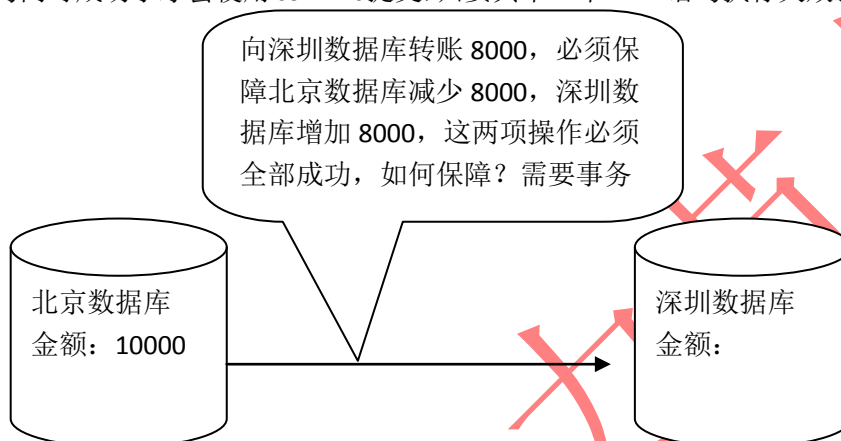
```
delete from emp where comm=300;
```

- 删除津贴为 null 的员工

```
delete from emp where comm is null;
```

### 11.1.4 事务概述

什么是事务？不可再分的最小的工作单元。宏观角度表示一个完整的不可分割的业务。其实是批量 DML 语句必须同时成功，或者同时失败。只要是 DML 语句开始执行就表示开启了一个事务，只要 commit 语句执行，就标识着这个事务结束。在同一个事务中，所有的 DML 语句同时成功了才会使用 commit 提交，只要其中一个 DML 语句执行失败就要 rollback 回滚。



事务可以保证多个操作原子性，要么全成功，要么全失败。对于数据库来说事务保证批量的 sql 要么全成功，要么全失败。事务具有四个特征 ACID

- a) 原子性
- b) 一致性
- c) 隔离性
  - 隔离级别：
    1. 读未提交（脏读）(READ UNCOMMITTED)
    2. 读提交 (READ COMMITTED)
    3. 可重复读 (REPEATABLE READ)
    4. 序列化 (SERIALIZABLE)

Oracle 中只支持：READ COMMITTED 和 SERIALIZABLE

设置事务的隔离级别：set transaction isolation level SERIALIZABLE;

- d) 持久性

事务控制语言（TCL）

事务中存在一些概念：

- a) 事务（Transaction）：一批操作（一组 DML sql）
- b) 开启事务（Begin Transaction）
- c) 回滚事务（rollback Transaction）--Oracle---rollback
- d) 提交事务（commit transaction）----Oracle--commit

当执行 DML 语句时其实就是开启一个事务

关于事务的回滚需要注意：只能回滚 insert、delete 和 update 语句，不能回滚 select（回滚

select 没有任何意义), 对于 create、drop、alter 这些无法回滚

Delete 和 truncate 都可以删除表中的数据。

Delete 语句删除数据之后还可以回滚

Truncate 语句删除数据之后不可以回滚

Delete 语句是 DML 语句

Truncate 语句是 DDL 语句

Truncate 语句是将表截断。

Delete from emp\_bak;

Truncate table emp\_bak;

## 11.2 表的操作

### 11.2.1 创建表

- 语法规则

```
create table 表名 (
    字段名 类型,
    .....
    .....
);
```

- Oracle 常用数据类型

类型	描述
Char(长度)	定长字符串, 存储空间大小固定, 适合作为主键或外键 如果数据不足位数, 那么数据库会自动使用空格补足位数
Varchar2(长度)	变长字符串, 存储空间等于实际数据空间 如果数据不足位数, 那么数据库不会自动使用空格补足位数
Number(有效数字的个数, 小数位)	数值型
Date	日期型
BLOB (Binary Large Object)	二进制大对象
CLOB (Character Large Object)	字符大对象

- 建立学生信息表, 字段包括: 学号、姓名、性别、出生日期、email、班级标识

```
create table t_student(
    student_id    number(10),
    student_name  varchar2(30), → 32*n
    sex          char(2), → 1 or 0
    birthday      date, → CHAR(10)
    email         varchar2(30), → 32*n
```

```
classes_id    number(3)
)
```

命令提示符 - sqlplus scott/tiger

```
SQL> desc t_student;
```

名称	是否为空? 类型
STUDENT_ID	NUMBER(10)
STUDENT_NAME	VARCHAR2(30)
SEX	CHAR(2)
BIRTHDAY	DATE
EMAIL	VARCHAR2(30)
CLASSES_ID	NUMBER(3)

SQL>

- 建立学生信息表，字段包括：学号、姓名、性别、出生日期、email、班级标识，性别加入默认值为“男”，出生日期默认为当前日期

```
create table t_student(
    student_id    number(10),
    student_name  varchar2(30),
    sex           char(2) default '男',
    birthday      date default sysdate,
    email         varchar2(30),
    classes_id    number(3)
)
```

如何插入数据

```
insert into t_student(student_id, student_name, email, classes_id) values(1111, 'lisi', 'lisi@152.net', 10);
```

注：sex 和 birthday 会使用默认值

### 11.2.2 创建表加入约束(保证表中的数据合法有效)

- 常见的约束
  - a) 非空约束，not null
  - b) 唯一约束，unique
  - c) 主键约束，primary key
  - d) 外键约束，foreign key
  - e) 自定义检查约束，check（不建议使用）
- 非空约束，not null

非空约束，针对某个字段设置其值不为空，如：学生的姓名不能为空

```
create table t_student(
    student_id    number(10),
    student_name  varchar2(30) not null,
    sex           char(2) default '男',
    birthday      date default sysdate,
    email         varchar2(30),
    classes_id    number(3)
)
```

以上，我可以自己起约束名称，如：

```
create table t_student(  
    student_id    number(10),  
    student_name  varchar2(30) constraint student_name_not_null not null,  
    sex          char(2) default '男',  
    birthday      date default sysdate,  
    email         varchar2(30),  
    classes_id    number(3)  
)
```

系统表：user\_constraints; 描述当前数据库用户下所有的约束信息。

- 唯一约束，unique key

唯一性约束，它可以使某个字段的值不能重复，如：email 不能重复：

```
create table t_student(  
    student_id    number(10),  
    student_name  varchar2(30),  
    sex          char(2) default '男',  
    birthday      date default sysdate,  
    email         varchar2(30) unique,  
    classes_id    number(3)  
)
```

同样可以为唯一约束起一个名称，如：

```
create table t_student(  
    student_id    number(10),  
    student_name  varchar2(30),  
    sex          char(2) default '男',  
    birthday      date default sysdate,  
    email         varchar2(30) constraint email_unique unique,  
    classes_id    number(3)  
)
```

以上约束放到字段上了，也成为字段级的约束，还有一种约束叫表级约束，也就是说可以把约束信息放到字段的后面（注意：not null 约束只有列级定义方式）

```
create table t_student(  
    student_id    number(10),  
    student_name  varchar2(30),  
    sex          char(2) default '男',  
    birthday      date default sysdate,  
    email         varchar2(30),  
    classes_id    number(3),  
    constraint email_unique unique(email)  
)
```

- 主键约束，primary key

每个表应该具有主键，主键可以标识记录的唯一性，主键分为单一主键和复合（联合）主键，单一主键是由一个字段构成的，复合（联合）主键是由多个字段构成的



```
create table t_student(
    student_id    number(10) primary key,
    student_name  varchar2(30),
    sex          char(2) default '男',
    birthday      date default sysdate,
    email         varchar2(30),
    classes_id    number(3)
)
```

也可以采用表级约束

```
create table t_student(
    student_id    number(10),
    student_name  varchar2(30),
    sex          char(2) default '男',
    birthday      date default sysdate,
    email         varchar2(30),
    classes_id    number(3),
    constraint student_id_pk primary key(student_id)
)
```

复合主键，如：学生代码和学生姓名构成主键

```
create table t_student(
    student_id    number(10),
    student_name  varchar2(30),
    sex          char(2) default '男',
    birthday      date default sysdate,
    email         varchar2(30),
    classes_id    number(3),
    constraint student_id_name_pk primary key(student_id,student_name)
)
```

#### ● 外键约束, foreign key

外键主要是维护表之间的关系的，主要是为了保证参照完整性，如果表中的某个字段为外键字段，那么该字段的值必须来源于参照的表的主键，如：emp 中的 deptno 值必须来源于 dept 表中的 deptno 字段值。

建立学生和班级表之间的连接

首先建立班级表 t\_classes

```
create table t_classes(
    classes_id number(3),
    classes_name varchar2(30),
    constraint t_classes_classes_id_pk primary key(classes_id)
)
```

对 t\_student 表中的 classes\_id 建立外键关系（建立外键其实就是建立两个表的父子关系，存在外键的表是子表，被引用的表为父表）

```
create table t_student(
    student_id    number(10),
```

```
student_name    varchar2(30),
sex             char(2) default '男',
birthday        date default sysdate,
email           varchar2(30),
classes_id      number(3) references t_classes(classes_id),
constraint student_id_pk primary key(student_id)
)
```

关于外键的删除，先删除子再删除父

```
SQL> delete from t_classes;
```

```
delete from t_classes
```

```
*
```

第 1 行出现错误:

ORA-02292: 违反完整约束条件 (SCOTT.SYS\_C005440) - 已找到子记录

建立外键还可以使用表级约束

```
create table t_student(
    student_id    number(10),
    student_name  varchar2(30),
    sex           char(2) default '男',
    birthday      date default sysdate,
    email         varchar2(30),
    classes_id    number(3),
    constraint student_id_pk primary key(student_id),
    constraint student_fk_classes_id foreign key(classes_id) references t_classes(classes_id)
)
```

- 自定义检查约束，check（不建议使用）

使用 check 可以检查表中的字段，是否符合某一个表达式，如：性别只能为“男”和“女”

```
create table t_student(
    student_id    number(10),
    student_name  varchar2(30),
    sex           char(2) default '男',
    birthday      date default sysdate,
    email         varchar2(30),
    classes_id    number(3) not null references t_classes(classes_id),
    constraint student_id_pk primary key(student_id),
    constraint student_check_sex check(sex in('男','女'))
)
```

Check 不建议使用，关于验证一般都放到应用程序中作，而不放到数据库中作

### 11.2.3 t\_student 完整示例

```
create table t_student(
    student_id    number(10),
```

```
student_name    varchar2(30) not null,
sex             char(2) not null,
birthday        date default sysdate,
email           varchar2(30) unique,
classes_id      number(3),
constraint student_id_pk primary key(student_id),
constraint student_fk_classes_id foreign key(classes_id) references t_classes(classes_id),
constraint student_check_sex check(sex in('男','女'))
)
```

### 11.2.4 增加/删除/修改表结构

采用 alter table 来增加/删除/修改表结构，不影响表中的数据

- 添加字段

如：需求发生改变，需要向 t\_student 中加入联系电话字段，字段名称为：contact\_tel 类型为 varchar2(40)

```
alter table t_student add(contact_tel varchar2(40));
```

```
SQL> alter table t_student add(contact_tel varchar2(40));
表已更改。
SQL> desc t_student;
 名称                                           是否为空? 类型
-----
STUDENT_ID                                     NOT NULL   NUMBER(10)
STUDENT_NAME                                  NOT NULL   VARCHAR2(30)
SEX                                             NOT NULL   CHAR(2)
BIRTHDAY                                       DATE
EMAIL                                          VARCHAR2(30)
CLASSES_ID                                    NUMBER(3)
CONTACT_TEL                                   VARCHAR2(40)
```

```
SQL> select * from t_student;

STUDENT_ID STUDENT_NAME      SE BIRTHDAY      EMAIL              CLASSES_ID CONTACT_TEL
-----
2222 wangwu                女 04-10月-09     zhangsan@163.net    10
```

- 修改字段

如：std\_name 无法满足需求，长度需要更改为 50

注：如果数值类型的数据需要减小长度，那么该列必须为空，如果字符类型的数据需要减小长度，那么只要减小的长度大于已有数据的最大长度就可以了

```
alter table t_student modify(student_name varchar2(50));
```

```

SQL> alter table t_student modify(student_name varchar2(50));
表已更改。

SQL> desc t_student;
名称                                是否为空? 类型
-----
STUDENT_ID                          NOT NULL  NUMBER(10)
STUDENT_NAME                        NOT NULL  VARCHAR2(50)
SEX                                  NOT NULL  CHAR(2)
BIRTHDAY                             DATE
EMAIL                                VARCHAR2(30)
CLASSES_ID                          NUMBER(3)
CONTACT_TEL                          VARCHAR2(40)

SQL> select * from t_student;

STUDENT_ID STUDENT_NAME                                SE BIRTHDAY     EMAIL                CLASSES_ID CONTACT_TEL
-----
2222 wangwu                                女 04-10月-09     zhangsan@163.net     10
  
```

### ● 删除字段

如：删除联系电话字段

```

alter table t_student drop(contact_tel);
或者
alter table t_student drop column contact_tel;
  
```

## 11.2.5 增加/删除/修改表约束

### ● 删除约束

将 t\_student 中的 classes\_id 外键约束删除

```
alter table t_student drop constraint STUDENT_FK_CLASSES_ID;
```

### ● 添加约束

将 t\_student 中的 classes\_id 加入外键约束

```
alter table t_student add constraint t_classes_fk_classes_id foreign key(classes_id) references
t_classes(classes_id);
```

### ● 修改约束

```
alter table t_student modify(student_name varchar2(50) null);
```

## 11.2.6 删除表

```
drop table t_classes;
```

如果存在父子表（存在外键关系），先删除子再删除父

## 11.3 索引(index)

索引的目的是提高查询数据的速度，索引一本书的目录一样，索引的建立原则，比较少的DML（insert、update、delete），经常出现在 where 语句中的字段

### 11.3.1 建立索引

如经常根据 birthday 进行查询，并且遇到了性能瓶颈，首先查看程序是否存算法问题，再考虑对 birthday 建立索引，建立索引如下：

```
create index t_student_birthday on t_student(birthday);
```

### 11.3.2 删除索引

```
drop index T_STUDENT_BIRTHDAY;
```

主键建立后，会相应的为主键建立索引，所以根据主键查询，通常比普通字段快

## 11.4 视图(view)

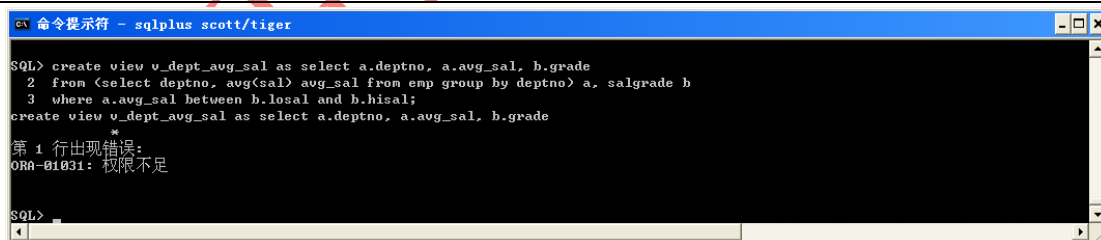
如下示例：

```
select a.deptno, a.avg_sal, b.grade
from (select deptno, avg(sal) avg_sal from emp group by deptno) a, salgrade b
where a.avg_sal between b.losal and b.hisal;
```

为什么使用视图？因为需求决定以上语句需要在多个地方使用，如果频繁的拷贝以上代码，会给维护带来成本，视图可以解决这个问题

### 11.4.1 创建视图

```
create view v_dept_avg_sal as select a.deptno, a.avg_sal, b.grade from (select deptno, avg(sal)
avg_sal from emp group by deptno) a, salgrade b where a.avg_sal between b.losal and b.hisal;
```



出现错误，权限不够，如何查询某个用户拥有的权限？

```
select * from session_privs;
```

如何切换用户？

```
conn system/bjpowernode@192.168.1.23/bjpowernode.com
```

如何对 scott 用户授权？

切换到 system 用户

```
conn system/bjpowernode@192.168.1.23/bjpowernode.com
```

在 system 用户下为 scott 授权，授予 scott 创建视图的权利

```
grant create view to scott; (撤销权限: revoke create view,create session from scott)
```

再次切换到 scott 用户下，查看是否拥有创建视图的权利

conn scott/tiger@192.168.1.23/bjpowernode.com

select \* from session\_privs;

```
命令提示符 - sqlplus scott/tiger
SQL> select * from session_privs;

PRIVILEGE
-----
CREATE SESSION
UNLIMITED TABLESPACE
CREATE TABLE
CREATE CLUSTER
CREATE VIEW
CREATE SEQUENCE
CREATE PROCEDURE
CREATE TRIGGER
CREATE TYPE
CREATE OPERATOR
CREATE INDEXTYPE

已选择11行。

SQL>
```

Scott 用户已经拥有了创建视图的权利

开始创建视图

```
命令提示符 - sqlplus scott/tiger
SQL> select view_name from user_views;

VIEW_NAME
-----
U_DEPT_AVG_SAL

SQL> desc U_DEPT_AVG_SAL;
名称 是否为空? 类型
-----
DEPTNO          NUMBER(2)
AUG_SAL         NUMBER
GRADE          NUMBER

SQL>
```

如何使用视图?

视图的使用和表的使用是一致的，但是视图不能进行增删改，因为视图是表的结果，采用视图主要是为了操作的方便性，重复使用的结果集考虑建成视图，如果表的结构可能会频繁发生变化，那么最好设置视图

```
命令提示符 - sqlplus scott/tiger
SQL> select * from U_DEPT_AVG_SAL;

DEPTNO    AUG_SAL    GRADE
-----
30 1566.66667 3
20 2175 4
10 2916.66667 4

SQL>
```

## 11.4.2 删除视图

drop view V\_DEPT\_AVG\_SAL;

```
命令提示符 - sqlplus scott/tiger
SQL> drop view U_DEPT_AVG_SAL;

视图已删除。

SQL> select * from user_views;

未选定行

SQL>
```

## 11.5 序列(Sequence)

序列是 Oracle 特有的，它可以维护一个自增的数字序列，通常从 1 开增长，但可以设置，例如：学生表 `t_student` 中的编号，可以采用 Oracle 的序列的方式来维护  
还有一种经常使用的生成策略是 Identity，如：Mysql/MS SQL Server

序列的两个属性：`nextval` 和 `currval`。在 Oracle 中序列 `sequence` 对象是共享的。  
Sequence 作用：帮助自动给表生成主键。

### 11.5.1 创建序列

```
create sequence seq_student_id start with 1 increment by 1;
```

创建一个完整的序列：

```
create sequence my_seq_01
```

```
start with 100
```

```
minvalue 100
```

```
maxvalue 1000
```

```
increment by 10
```

```
cycle
```

或者 no cycle

```
cache 10;
```

```

SQL> create sequence seq_student_id start with 1 increment by 1;
序列已创建。
SQL> select sequence_name from user_sequences;
SEQUENCE_NAME
-----
SEQ_STUDENT_ID
SQL>
  
```

### 11.5.2 使用序列

向 `t_student` 中加入数据

```
insert into t_student(student_id, student_name, sex, email, classes_id)
values(SEQ_STUDENT_ID.nextval, '女', 'zhangsan442@163.net', 10);
```

```

SQL> insert into t_student(student_id, student_name, sex, email, classes_id) values(SEQ_STUDENT_ID.nextval, '女', 'zhangsan442@163.net', 10);
已创建 1 行。
SQL> select * from t_student;
STUDENT_ID STUDENT_NAME      SE BIRTHDAY      EMAIL                                CLASSES_ID
-----
1          女 04-10月-09     zhangsan11@163.net                10
3          女 04-10月-09     zhangsan12@163.net                10
4          女 04-10月-09     zhangsan331@163.net                10
6          女 04-10月-09     zhangsan442@163.net                10
SQL>
  
```

Sequence 如果出现错误会断号

### 11.5.3 删除序列

```
drop sequence SEQ_STUDENT_ID;
```

## 11.6 存储过程、触发器和游标

### 11.6.1 存储过程

存储过程最直接的理解：就是保存了批量的 sql (select,insert,if for)，以后可以通过一个名字把这些批量的 sql 执行，使用存储过程在大批量数据查询或计算时会带来高性能，存储过程编写和调试比较复杂，不同数据库产品存储过程差异非常大，很难实现平滑移植

- 建立存储过程

```
create or replace procedure proc_test(in_var number,out_var out sys_refcursor)
as
begin
    open out_var for select * from emp where deptno=in_var;
end;
```

- 执行存储过程

```
var ret refcursor
exec proc_test(20,:ret)
print :ret
```

### 11.6.2 触发器

触发器是特殊的存储过程，它与数据库的 insert、update 和 delete 相关联，如定义完成触发器之后，会在 insert、update 或 delete 语句执行前或执行后自动执行触发器中的内容

触发器示例，向 emp 表中加入数据，采用触发器自动再向 t\_log 表里加入一条数据

- 首先建立 t\_log 表

```
create table t_log (
    log_id number(10) primary key,
    log_time date
)
```

- 为建立 t\_log 的主键建立 sequence

```
create sequence seq_log_id start with 1 increment by 1;
```

- 建立触发器

```
create or replace trigger tri_test
after insert on emp
begin
```



```
insert into t_log(log_id, log_time) values(seq_log_id.nextval, sysdate);
end;
```

- 向 emp 表中加入数据

```
insert into emp(empno, deptno) values(7777, 10);
```

SQL> select \* from emp;

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
8888	lisi	MANAGER			4000		20
9999	wangwu						
7369	SMITH	CLERK	7902	17-12月-80	800		20
7499	ALLEN	SALESMAN	7698	20-2月-81	1600	300	30
7521	WARD	SALESMAN	7698	22-2月-81	1250	500	30
7566	JONES	MANAGER	7839	02-4月-81	2975		20
7654	MARTIN	SALESMAN	7698	28-9月-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-5月-81	2850		30
7782	CLARK	MANAGER	7839	09-6月-81	2450		10
7788	SCOTT	ANALYST	7566	19-4月-81	3000		20
7839	KING	PRESIDENT		17-11月-81	5000		10
7844	TURNER	SALESMAN	7698	08-9月-81	1500	0	30
7876	ADAMS	CLERK	7788	23-5月-81	1100		20
7900	JAMES	CLERK	7698	03-12月-81	950		30
7902	FORD	ANALYST	7566	03-12月-81	3000		20
7934	MILLER	CLERK	7782	23-1月-82	1300		10
7777							10

已选择17行。

SQL> select \* from t\_log;

LOG_ID	LOG_TIME
1	06-6月-09

SQL>

在 emp 中多了一条数据 empno 为 7777，在 t\_log 中自动加入了一条数据，这就是触发器的作用。

## 11.7 游标

我们有时采用 select 会返回一个结果集，使用简单的 select 无法得到上一行，下一行，后 5 行，后 10 行，如果要做到这一点必须使用游标，游标是存储在数据库服务器上的一个数据库查询，它不是一条 select 语句，他是一个结果集，有了游标就可以根据需要滚动浏览数据了

下面通过一个示例，根据岗位加工资，如果是 MANAGER 增加 20% 的工资，如果是 SALESMAN 增加 10% 的工资，其他的增加 5% 的工资

(for update 是将数据库表的数据进行锁定的操作，不让其他的事务可以修改。在 Oracle 中的这种锁定是对查询的结果数据进行加锁，其他的数据不会被加锁。我们把这样的锁定方式叫行级锁)

```
create or replace procedure proc_sal
is
    cursor c is
        select * from emp for update;
begin
    for v_emp in c loop
        if (v_emp.job = 'MANAGER') then
```

```
        update emp set sal = sal + sal*0.2 where current of c;  
    elsif (v_emp.job = 'SALESMAN') then  
        update emp set sal = sal + sal*0.1 where current of c;  
    else  
        update emp set sal = sal + sal*0.05 where current of c;  
    end if;  
  
end loop;  
commit;  
end;
```

执行存储过程

```
exec proc_sal;
```

## 第12章 常用的 DBA 命令及设计三范式

### 12.1.1 查看用户拥有的数据库对象

```
select object_name from user_objects;
```

系统表总结:

```
USER_TABLES  
USER_SEQUENCES  
USER_VIEWS  
USER_INDEXES  
USER_CONSTRAINTS  
USER_OBJECTS  
SESSION_PRIVS
```

### 12.1.2 查看约束信息

```
select constraint_name from user_constraints;
```

### 12.1.3 查看用户拥有的表

```
select table_name from user_tables;
```

#### 12.1.4 查看用户拥有的视图

```
select view_name from user_views;
```

#### 12.1.5 查看用户拥有的触发器

```
select trigger_name from user_triggers;
```

#### 12.1.6 查看用户拥有的序列

```
select sequence_name from user_sequences;
```

#### 12.1.7 查看用户拥有的存储过程

```
select object_name from user_procedures;
```

#### 12.1.8 查看用户拥有的索引

```
select index_name from user_indexes;
```

#### 12.1.9 显示当前用户

show user 不是 SQL 语句，是一个 oracle 提供的 sqlplus 命令

#### 12.1.10 切换用户

```
conn system/bjpowernode@IP/全局数据库名
```

#### 12.1.11 以数据库管理员的身份登录

```
conn sys/bjpowernode@IP/全局数据库名 as sysdba;
```

### 12.1.12 查看所有的用户

```
select username from dba_users; //dba_users 只有管理员才有这张表
```

### 12.1.13 查看用户拥有的权限

```
select * from session_privs;
```

常用权限

CREATE SESSION	连接数据库
CREATE TABLE	创建表
CREATE VIEW	创建视图
CREATE SEQUENCE	创建序列
CREATE PROCEDURE	创建存储过程
CREATE TRIGGER	创建触发器
CREATE INDEXTYPE	创建索引
UNLIMITED TABLESPACE	对表空间的使用

### 12.1.14 给用户加锁

```
alter user scott account lock;
```

### 12.1.15 给用户解锁

```
alter user scott account unlock;
```

### 12.1.16 修改用户密码(数字不能开头)

```
alter user scott identified by tiger123;
```

### 12.1.17 新建用户

```
create user epay identified by bjpowernode;
```

### 12.1.18 删除用户及相关对象

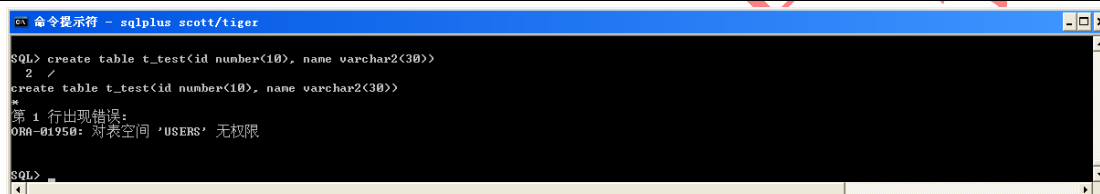
```
drop user epay cascade;
```

### 12.1.19 给用户授权（多个采用逗号间隔）

```
grant create session, create table to epay;
```

### 12.1.20 分配空间 xxx 给用户

```
create table t_test(id number(10), name varchar2(30))
```

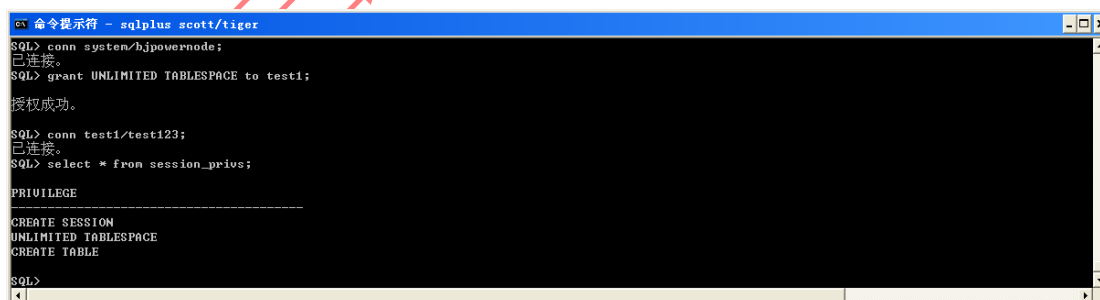


以上出现无法创建表，主要原因在于没有分配表空间，也就是我们新建的表不知道放到什么地方。

```
alter user epay default tablespace xxx;
```

### 12.1.21 授权表空间给用户

```
grant UNLIMITED TABLESPACE to epay;
```



### 12.1.22 一个完整的过程,创建用户、创建表空间、授权、建表

- 创建用户

```
create user epay identified by bjpowernode;
```

- 创建表空间

```
create tablespace epay_tablespace datafile 'D:\oraclexe\app\oracle\oradata\XE\epay.DBF' size 50m;
```

- 将表空间分配给用户

```
alter user epay default tablespace epay_tablespace;
```

- 给用户授权

```
grant create session, create table, unlimited tablespace,create sequence to epay;
```

- 以 epay 登陆建立表,tt\_test

```
create table tt_test(id number(10));
```

### 12.1.23 导入和导出命令 imp、exp

Export

```
exp scott/tiger file=C:\EMP.DMF tables=e mp,dept,salgrade
```

Import

```
imp scott/tiger file=C:\EMP.DMF
```

## 12.2 数据库设计的三范式

### 12.2.1 第一范式

数据库表中不能出现重复记录，每个字段是原子性的不能再分  
不符合第一范式的示例

学生编号	学生姓名	联系方式
1001	张三	zs@gmail.com,1359999999
1002	李四	ls@gmail.com,13699999999
1001	王五	ww@163.net,13488888888

存在问题：

- 最后一条记录和第一条重复（不唯一，没有主键）
- 联系方式字段可以再分，不是原子性的

学生编号(pk)	学生姓名	email	联系电话
1001	张三	zs@gmail.com	1359999999
1002	李四	ls@gmail.com	13699999999
1003	王五	ww@163.net	13488888888

关于第一范式，每一行必须唯一，也就是每个表必须有主键，这是我们数据库设计的最基本要求，主要通常采用数值型或定长字符串表示，关于列不可再分，应该根据具体的情况来决定。如联系方式，为了开发上的便利行可能就采用一个字段了。

### 12.2.2 第二范式

第二范式是建立在第一范式基础上的，另外要求所有非主键字段完全依赖主键，不能产生部分依赖

示例：

学生编号	学生姓名	教师编号	教师姓名
1001	张三	001	王老师
1002	李四	002	赵老师
1003	王五	001	王老师
1001	张三	002	赵老师

确定主键：

学生编号(PK)	教师编号(PK)	学生姓名	教师姓名
1001	001	张三	王老师
1002	002	李四	赵老师
1003	001	王五	王老师
1001	002	张三	赵老师

以上虽然确定了主键，但此表会出现大量的冗余，主要涉及到的冗余字段为“学生姓名”和“教师姓名”，出现冗余的原因在于，学生姓名部分依赖了主键的一个字段学生编号，而没有依赖教师编号，而教师姓名部门依赖了主键的一个字段教师编号，这就是第二范式部分依赖。

解决方案如下：

学生信息表

学生编号 (PK)	学生姓名
1001	张三
1002	李四
1003	王五

教师信息表

教师编号 (PK)	教师姓名
001	王老师
002	赵老师

教师和学生的关系表

学生编号(FK)	教师编号(FK)
1001	001
1002	002
1003	001
1001	002

如果一个表是单一主键，那么它就复合第二范式，部分依赖和主键有关系

### 12.2.3 第三范式

建立在第二范式基础上的，非主键字段不能传递依赖于主键字段。

学生编号 (PK)	学生姓名	班级编号	班级名称
-----------	------	------	------

1001	张三	01	一年一班
1002	李四	02	一年二班
1003	王五	03	一年三班

从上表可以看出，班级名称字段存在冗余，因为班级名称字段没有直接依赖于主键，班级名称字段依赖于班级编号，班级编号依赖于学生编号，那么这就是传递依赖，解决的办法是将冗余字段单独拿出来建立表，如：

学生信息表

学生编号 (PK)	学生姓名	班级编号
1001	张三	01
1002	李四	02
1003	王五	03

班级信息表

班级编号	班级名称
01	一年一班
02	一年二班
03	一年三班

#### 12.2.4 三范式总结

第一范式：有主键，具有原子性，字段不可分割

第二范式：完全依赖，没有部分依赖

第三范式：没有传递依赖

数据库设计尽量遵循三范式，但是还是根据实际情况进行取舍，有时可能会拿冗余换速度，最终目的要满足客户需求。

## 第13章 课后练习题

### 13.1 取得每个部门最高薪水的人员名称

第一步：取得每个部门的最高薪水

```
Select deptno, max(sal) from emp group by deptno
```

第二步：根据第一步的结果和员工表进行关联，获取人员名称、

```
Select e.ename, e.sal, e.deptno from emp e join (Select deptno, max(sal) maxSal from emp group by deptno) t on e.deptno = t.deptno and e.sal = t.maxSal
```



```

C:\ 命令提示符 - sqlplus scott/tiger

ENAME          SAL          DEPTNO
-----
BLAKE          2850           30
SCOTT          3000           20
KING           5000           10
FORD           3000           20

SQL>

```

## 13.2 哪些人的薪水在部门的平均薪水之上

第一步：获取每个部门的平均薪水

```
Select deptno, avg(sal) avgSal from emp group by deptno
```

第二步：根据第一步的结果和员工表进行关联，获取人员名称

```
Select e.ename, e.sal, e.deptno from emp e join (Select deptno, avg(sal) avgSal from emp group
by deptno) t on e.deptno = t.deptno and e.sal > t.avgSal
```

```

C:\ 命令提示符 - sqlplus scott/tiger

ENAME          SAL
-----
ALLEN          1600
JONES          2975
BLAKE          2850
SCOTT          3000
KING           5000
FORD           3000

已选择6行。

SQL>

```

## 13.3 取得部门中（所有人的）平均的薪水等级，如下：

第一步：获取每个员工的薪水等级

```
Select e.empno, e.ename, e.deptno, g.grade from emp e join salgrade g on e.sal between g.losal
and g.hisal
```

第二步：将第一步的结果用部门进行分组，然后获取等级的平均值

```
Select deptno, avg(grade) from (Select e.empno, e.ename, e.deptno, g.grade from emp e join
salgrade g on e.sal between g.losal and g.hisal) group by deptno
```

```

C:\ 命令提示符 - sqlplus scott/tiger
DEPTNO AUG<GRADE>
-----
30      2.5
20      2.8
10      3.66666667
SQL>

```

### 13.4 不准用组函数（Max），取得最高薪水（给出两种解决方案）

第一种方法：(Rownum)

1. 将员工薪水降序排列

```
Select sal from emp order by sal desc
```

- 2 取得第一条数据

```
Select sal from (Select sal from emp order by sal desc) where rownum = 1
```

第二种方法：（自关联）

1. 将 emp 表当做 2 张表来处理，使用的笛卡尔乘积的方法进行比较，得到最大值以外的值

```
Select distinct e.sal from emp e join emp t on e.sal < t.sal
```

2. 获取最大值

```
Select sal from emp where sal not in (Select distinct e.sal from emp e join emp t on e.sal < t.sal)
```

```

C:\ 命令提示符 - sqlplus scott/tiger
SAL
---
5000
SQL>

```

### 13.5 取得平均薪水最高的部门的部门编号（至少给出两种解决方案）

第一种方法：

1. 取得部门的平均薪水

```
Select deptno, avg(sal) avgSal from emp group by deptno
```

2. 取得最高的平均薪水

```
Select max(avgSal) from (Select deptno, avg(sal) avgSal from emp group by deptno)
```

3. 将第一步和第二步获取的结果进行关联

```
Select deptno from (Select deptno, avg(sal) avgSal from emp group by deptno) t join (Select
max(avgSal) maxAvgSal from (Select deptno, avg(sal) avgSal from emp group by deptno)) t1 on t.
avgSal = t1.maxAvgSal
```

第二种方法:

```
Select deptno from (Select deptno, avg(sal) avgSal from emp group by deptno order by avgSal
desc ) where rownum = 1
```

第三种方式:

```
Select deptno from emp group by deptno having avg(sal) = ( select max(avg(sal)) from emp group
by deptno )
```

```
命令提示符 - sqlplus scott/tiger
SQL> /
DEPTNO
-----
10
SQL>
```

## 13.6 取得平均薪水最高的部门的部门名称

第一步: 获取平均薪水最高的部门编号 (参考上一题)

```
Select deptno from emp group by deptno having avg(sal) = ( select max(avg(sal)) from emp group
by deptno )
```

第二步: 根据部门编号获取部门名称

```
Select dname from dept where deptno = (Select deptno from emp group by deptno having
avg(sal) = ( select max(avg(sal)) from emp group by deptno ))
```

```
命令提示符 - sqlplus scott/tiger
SQL> /
DNAME
-----
ACCOUNTING
SQL>
```

## 13.7 求平均薪水的等级最低的部门的部门名称

第一步: 获取每个部门的平均薪水

```
Select deptno, avg(sal) avgSal from emp group by deptno
```

第二步: 获取部门的平均薪水等级

```
Select t.*, g.grade from salgrade g join (Select deptno, avg(sal) avgSal from emp group by deptno)
```

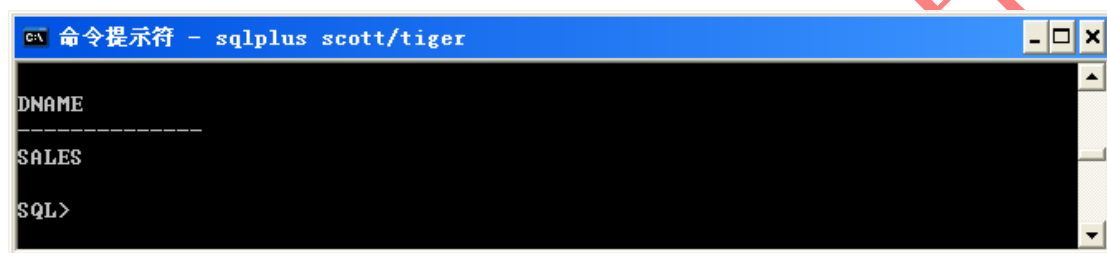
t on t.avgSal between g.losal and g.hisal

第三步：取得最低的等级及部门编号

Select deptno from (Select t.\*, g.grade from salgrade g join (Select deptno, avg(sal) avgSal from emp group by deptno) t on t.avgSal between g.losal and g.hisal order by grade) where rownum = 1

第四步：根据编号获取名称

Select dname from dept where deptno = (Select deptno from (Select t.\*, g.grade from salgrade g join (Select deptno, avg(sal) avgSal from emp group by deptno) t on t.avgSal between g.losal and g.hisal order by grade) where rownum = 1)



## 13.8 取得比普通员工(员工代码没有在 mgr 字段上出现的)的最高薪水还要高的经理人姓名

第一步：获取经理的员工编号

Select distinct mgr from emp where mgr is not null

第二步：获取普通员工的最高薪水

Select max(sal) maxSal from emp where empno not in (Select distinct mgr from emp where mgr is not null)

第三步：将第一步和第二步关联，获取结果

Select m.ename, m.sal from emp m where m.empno in (Select distinct mgr from emp where mgr is not null) and m.sal > (Select max(sal) maxSal from emp where empno not in (Select distinct mgr from emp where mgr is not null))

```

C:\ 命令提示符 - sqlplus scott/tiger
-----
ENAME          SAL
-----
JONES          2975
BLAKE          2850
CLARK          2450
SCOTT          3000
KING           5000
FORD           3000

已选择6行。

SQL>

```

### 13.9 取得薪水最高的前五名员工

```

select *
from
(
  select rownum r, t.*
  from
  (
    select ename, sal from emp order by sal desc
  ) t
  where rownum <=5
)where r> 0

```

```

C:\ 命令提示符 - sqlplus scott/tiger
-----
ROWNUM ENAME          SAL
-----
1 KING           5000
2 SCOTT          3000
3 FORD           3000
4 JONES          2975
5 BLAKE          2850

SQL>

```

### 13.10 取得薪水最高的第六到第十名员工

```

select *
from
(
  select rownum r, t.*
  from

```

```
(
  select ename, sal from emp order by sal desc
) t
where rownum <=10
)where r> 5
```

命令提示符 - sqlplus scott/tiger

ENAME	SAL
CLARK	2450
ALLEN	1600
TURNER	1500
MILLER	1300
WARD	1250

SQL>

### 13.11 3.11、取得最后入职的 5 名员工

Select \* from ( Select ename, hiredate from emp order by hiredate desc) where rownum <=5

命令提示符 - sqlplus scott/tiger

ENAME	HIREDATE
ADAMS	23-5月 -87
SCOTT	19-4月 -87
MILLER	23-1月 -82
JAMES	03-12月 -81
FORD	03-12月 -81

SQL>

### 13.12 3.12、取得每个薪水等级有多少员工

第一步：获取每个员工的薪水等级

Select grade from emp e join salgrade g on e.sal between g.losal and g.hisal

第二步：根据薪水等级进行分组，然后获取数量

Select grade, count(\*) from (Select grade from emp e join salgrade g on e.sal between g.losal and g.hisal) group by grade

命令提示符 - sqlplus scott/tiger

GRADE	COUNT(*)
1	3
2	3
4	5
5	1
3	2

SQL>

### 13.13 3.13、面试题

有 3 个表 S, C, SC

S (SNO, SNAME) 代表 (学号, 姓名)

C (CNO, CNAME, CTEACHER) 代表 (课号, 课名, 教师)

SC (SNO, CNO, SCGRADE) 代表 (学号, 课号, 成绩)

问题:

1, 找出没选过“黎明”老师的所有学生姓名。

2, 列出 2 门以上 (含 2 门) 不及格学生姓名及平均成绩。

3, 即学过 1 号课程又学过 2 号课所有学生的姓名。

请用标准 SQL 语言写出答案, 方言也行 (请说明是使用什么方言)。Dialect

CREATE TABLE SC

```
(
  SNO      VARCHAR2(200 BYTE),
  CNO      VARCHAR2(200 BYTE),
  SCGRADE  VARCHAR2(200 BYTE)
);
```

CREATE TABLE S

```
(
  SNO      VARCHAR2(200 BYTE),
  SNAME    VARCHAR2(200 BYTE)
);
```

CREATE TABLE C

```
(
  CNO      VARCHAR2(200 BYTE),
  CNAME    VARCHAR2(200 BYTE),
  CTEACHER VARCHAR2(200 BYTE)
);
```

```
INSERT INTO C ( CNO, CNAME, CTEACHER ) VALUES ( '1', '语文', '张');
INSERT INTO C ( CNO, CNAME, CTEACHER ) VALUES ( '2', '政治', '王');
INSERT INTO C ( CNO, CNAME, CTEACHER ) VALUES ( '3', '英语', '李');
INSERT INTO C ( CNO, CNAME, CTEACHER ) VALUES ( '4', '数学', '赵');
INSERT INTO C ( CNO, CNAME, CTEACHER ) VALUES ( '5', '物理', '黎明');
commit;
```

```
INSERT INTO S ( SNO, SNAME ) VALUES ( '1', '学生 1');
INSERT INTO S ( SNO, SNAME ) VALUES ( '2', '学生 2');
INSERT INTO S ( SNO, SNAME ) VALUES ( '3', '学生 3');
INSERT INTO S ( SNO, SNAME ) VALUES ( '4', '学生 4');
commit;
```

```
INSERT INTO SC ( SNO, CNO, SCGRADE ) VALUES ( '1', '1', '40');
INSERT INTO SC ( SNO, CNO, SCGRADE ) VALUES ( '1', '2', '30');
INSERT INTO SC ( SNO, CNO, SCGRADE ) VALUES ( '1', '3', '20');
INSERT INTO SC ( SNO, CNO, SCGRADE ) VALUES ( '1', '4', '80');
INSERT INTO SC ( SNO, CNO, SCGRADE ) VALUES ( '1', '5', '60');
INSERT INTO SC ( SNO, CNO, SCGRADE ) VALUES ( '2', '1', '60');
INSERT INTO SC ( SNO, CNO, SCGRADE ) VALUES ( '2', '2', '60');
INSERT INTO SC ( SNO, CNO, SCGRADE ) VALUES ( '2', '3', '60');
INSERT INTO SC ( SNO, CNO, SCGRADE ) VALUES ( '2', '4', '60');
INSERT INTO SC ( SNO, CNO, SCGRADE ) VALUES ( '2', '5', '40');
INSERT INTO SC ( SNO, CNO, SCGRADE ) VALUES ( '3', '1', '60');
INSERT INTO SC ( SNO, CNO, SCGRADE ) VALUES ( '3', '3', '80');
commit;
```

问题 1:找出没选过“黎明”老师的所有学生姓名。

第一步：找到黎明老师教授的课程编号

```
Select cno from c where cteacher = '黎明'
```

第二步：找到选择了第一步获取的课程编号的学生

```
Select sno from sc where cno in (Select cno from c where cteacher = '黎明')
```

第三步：获取结果

```
Select sname from s where sno not in (Select sno from sc where cno in (Select cno from c where cteacher = '黎明'))
```

问题 2:列出 2 门以上（含 2 门）不及格学生姓名及平均成绩。

第一步：获取 2 门以上（含 2 门）不及格学生编号

```
Select sno from sc where SCGRADE < 60 group by sno having count(*) >=2
```

第二步：获取学生的平均成绩

```
Select sno, avg(SCGRADE) from sc where sno in (Select sno from sc where SCGRADE < 60 group by sno having count(*) >=2) group by sno
```

第三步：得到结果

```
Select sname, t. avgGrade from s join (Select sno, avg(SCGRADE) avgGrade from sc where sno in
```



```
(Select sno from sc where SCGRADE < 60 group by sno having count(*) >=2) group by sno) t on  
s.sno = t.sno
```

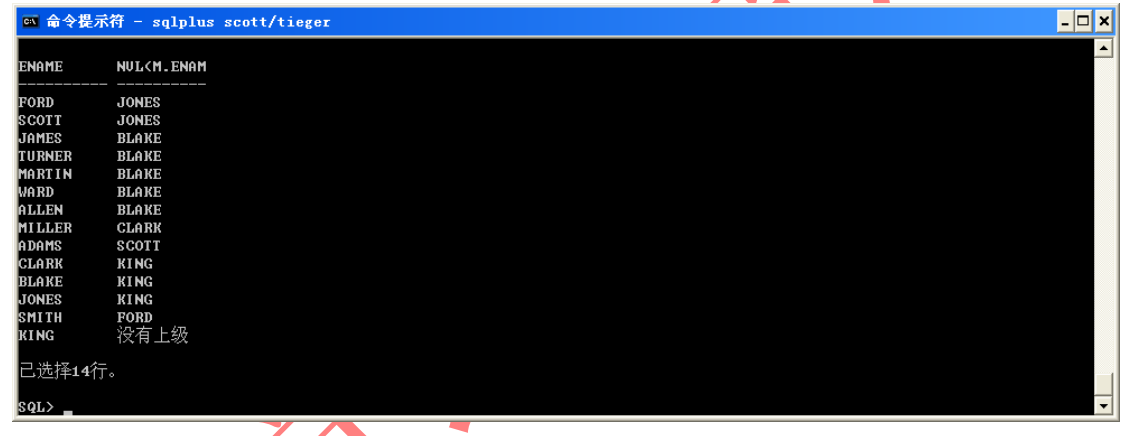
问题 3:既学过 1 号课程又学过 2 号课所有学生的姓名。

```
Select sname from s where sno in (Select sno from sc where cno = 2 and sno in (Select sno from  
sc where cno = 1))
```

### 13.14 3.14、列出所有员工及直接上级的姓名

```
Select e.ename, nvl (m.ename, '没有上级') mname from emp e left join emp m on e.mgr =  
m.empno (SQL99)
```

```
Select e.ename, nvl (m.ename, '没有上级') mname from emp e, emp m where e.mgr =  
m.empno(+) (SQL92)
```

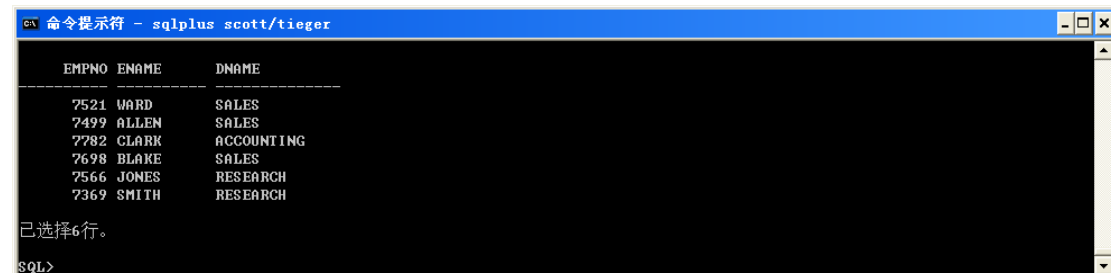


ENAME	NULCM. ENAM
FORD	JONES
SCOTT	JONES
JAMES	BLAKE
TURNER	BLAKE
MARTIN	BLAKE
WARD	BLAKE
ALLEN	BLAKE
MILLER	CLARK
ADAMS	SCOTT
CLARK	KING
BLAKE	KING
JONES	KING
SMITH	FORD
KING	没有上级

已选择14行。  
SQL>

### 13.15 3.15、列出受雇日期早于其直接上级的所有员工的编号,姓名, 部门名称

```
Select e.empno, e.ename, d.dname from emp e join emp m on e.hiredate < m.hiredate and  
e.mgr = m.empno join dept d on e.deptno = d.deptno
```



EMPNO	ENAME	DNAME
7521	WARD	SALES
7499	ALLEN	SALES
7782	CLARK	ACCOUNTING
7698	BLAKE	SALES
7566	JONES	RESEARCH
7369	SMITH	RESEARCH

已选择6行。  
SQL>

### 13.16 3.16、列出部门名称和这些部门的员工信息,同时列出那些没有员工的部门.

Select d.dname, e.\* from emp e right join dept d on d.deptno = e.deptno

命令提示符 - sqlplus scott/tieger

DNAME	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
RESEARCH	7369	SMITH	CLERK	7902	17-12月-80	800		20
SALES	7499	ALLEN	SALESMAN	7698	20-2月-81	1600	300	30
SALES	7521	WARD	SALESMAN	7698	22-2月-81	1250	500	30
RESEARCH	7566	JONES	MANAGER	7839	02-4月-81	2975		20
SALES	7654	MARTIN	SALESMAN	7698	28-9月-81	1250	1400	30
SALES	7698	BLAKE	MANAGER	7839	01-5月-81	2850		30
ACCOUNTING	7782	CLARK	MANAGER	7839	09-6月-81	2450		10
RESEARCH	7788	SCOTT	ANALYST	7566	19-4月-87	3000		20
ACCOUNTING	7839	KING	PRESIDENT		17-11月-81	5000		10
SALES	7844	TURNER	SALESMAN	7698	08-9月-81	1500	0	30
RESEARCH	7876	ADAMS	CLERK	7788	23-5月-87	1100		20
SALES	7900	JAMES	CLERK	7698	03-12月-81	950		30
RESEARCH	7902	FORD	ANALYST	7566	03-12月-81	3000		20
ACCOUNTING	7934	MILLER	CLERK	7782	23-1月-82	1300		10
OPERATIONS								

已选择15行。  
SQL>

### 13.17 3.17、列出至少有一个员工的所有部门

Select dname, count(\*) from emp e join dept d on e.deptno = d.deptno group by dname

命令提示符 - sqlplus scott/tieger

DNAME	COUNT(*)
ACCOUNTING	3
RESEARCH	5
SALES	6

SQL>

### 13.18 3.18、列出薪金比"SMITH"多的所有员工信息.

Select \* from emp where sal > (Select sal from emp where ename = 'SMITH')

命令提示符 - sqlplus scott/tieger

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7499	ALLEN	SALESMAN	7698	20-2月-81	1600	300	30
7521	WARD	SALESMAN	7698	22-2月-81	1250	500	30
7566	JONES	MANAGER	7839	02-4月-81	2975		20
7654	MARTIN	SALESMAN	7698	28-9月-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-5月-81	2850		30
7782	CLARK	MANAGER	7839	09-6月-81	2450		10
7788	SCOTT	ANALYST	7566	19-4月-87	3000		20
7839	KING	PRESIDENT		17-11月-81	5000		10
7844	TURNER	SALESMAN	7698	08-9月-81	1500	0	30
7876	ADAMS	CLERK	7788	23-5月-87	1100		20
7900	JAMES	CLERK	7698	03-12月-81	950		30
7902	FORD	ANALYST	7566	03-12月-81	3000		20
7934	MILLER	CLERK	7782	23-1月-82	1300		10

已选择13行。  
SQL>

### 13.19 3.19、列出所有"CLERK"(办事员)的姓名及其部门名称,部门的人数.

第一步: 获取工作岗位为办事员的员工信息

```
Select ename, deptno from emp where job = 'CLERK'
```

第二步: 获取部门名称

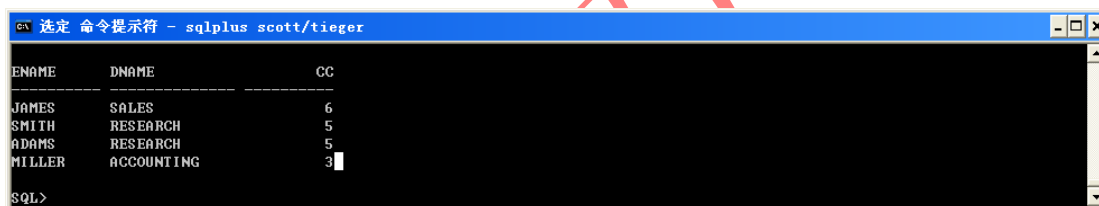
```
Select t.ename, t.deptno, d.dname from dept d join (Select ename, deptno from emp where job = 'CLERK') t on d.deptno = t.deptno
```

第三步: 获取每个部门的员工数量

```
Select dname, count(*) cc from emp e join dept d on e.deptno = d.deptno group by dname
```

第四步: 显示结果

```
Select t.ename, t.dname, t1.cc from (Select t.ename, t.deptno, d.dname from dept d join (Select ename, deptno from emp where job = 'CLERK') t on d.deptno = t.deptno) t join (Select dname, count(*) cc from emp e join dept d on e.deptno = d.deptno group by dname) t1 on t.dname = t1.dname
```



ENAME	DNAME	CC
JAMES	SALES	6
SMITH	RESEARCH	5
ADAMS	RESEARCH	5
MILLER	ACCOUNTING	3

### 13.20 3.20、列出最低薪金大于 1500 的各种工作及从事此工作的全部雇员人数.

第一步: 获取最低薪水大于 1500 的工作

```
Select job from emp group by job having min(sal) > 1500
```

第二步: 获取每个工作的员工数量

```
Select job , count(*) cc from emp group by job
```

第三步: 将前面 2 步数据进行关联, 获取结果

```
Select t.job, t1.cc from (Select job from emp group by job having min(sal) > 1500) t join (Select job , count(*) cc from emp group by job) t1 on t.job = t1.job
```



JOB	COUNT(E.EMPNO)
PRESIDENT	1
MANAGER	3
ANALYST	2

### 13.21 3.21、列出在部门"SALES"<销售部>工作的员工的姓名,假定不知道销售部的部门编号.

```
Select * from emp where deptno = (Select deptno from dept where dname = 'SALES')
```

### 13.22 3.22、列出薪金高于公司平均薪金的所有员工,所在部门,上级领导,雇员的工资等级.

第一步：获取公司的平均薪水

```
Select avg(sal) avgSal from emp
```

第二步：获取大于平均薪水的员工

```
Select sal, mgr, ename, deptno from emp where sal > (Select avg(sal) avgSal from emp)
```

第三步：获取部门名称

```
Select t.ename, d.dname from dept d join (Select ename, deptno from emp where sal > (Select avg(sal) avgSal from emp)) t on d.deptno = t.deptno
```

第四步：获取领导名称

```
Select t.ename, d.dname, nvl(m.ename, '无') from dept d join (Select mgr, ename, deptno from emp where sal > (Select avg(sal) avgSal from emp)) t on d.deptno = t.deptno left join emp m on t.mgr = m.empno
```

第五步：获取薪水等级 (字段别名是不需要加引号，因为不是字符串)

```
Select t.ename 姓名, d.dname 部门名称, nvl(m.ename, '无') 上级领导, g.grade 工资等级
from dept d join (Select sal, mgr, ename, deptno from emp where sal > (Select avg(sal) avgSal
from emp)) t on d.deptno = t.deptno left join emp m on t.mgr = m.empno join salgrade g on t.sal
between g.losal and g.hisal
```

姓名	部门名称	上级领导	工资等级
JONES	RESEARCH	KING	4
BLAKE	SALES	KING	4
CLARK	ACCOUNTING	KING	4
SCOTT	RESEARCH	JONES	4
KING	ACCOUNTING	无	5
FORD	RESEARCH	JONES	4

已选择6行。

SQL>

### 13.23 3.23、列出与"SCOTT"从事相同工作的所有员工及部门名称.

```
Select e.ename, d.dname from emp e, dept d where e.deptno = d.deptno and e.job = (Select job
from emp where ename = 'SCOTT') and e.ename != 'SCOTT'
```

```

命令提示符 - sqlplus scott/tiger
ENAME      DNAME
-----
FORD        RESEARCH
SQL>

```

13.24 3.24、列出薪金等于部门 30 中员工的薪金的所有员工的姓名和薪金.

Select ename, sal from emp where sal in (Select distinct sal from emp where deptno = 30)

```

SQL> Select ename, sal from emp where sal in (Select distinct sal from emp where deptno = 30);

ENAME      SAL
-----
ALLEN       1600
MARTIN      1250
WARD        1250
BLAKE       2850
TURNER      1500
JAMES       950
已选择6行。

```

13.25 3.25、列出薪金高于在部门 30 工作的所有员工的薪金的员工姓名和薪金.部门名称.

Select ename, sal from emp where sal > (Select max(sal) maxSal from emp where deptno = 30) and deptno != 30

Select e.ename, e.sal, d.dname from emp e join dept d on e.deptno = d.deptno where e.sal > (Select max(sal) maxSal from emp where deptno = 30) and e.deptno != 30

```

命令提示符 - sqlplus scott/tiger
ENAME      SAL DNAME
-----
JONES       2975 RESEARCH
SCOTT       3000 RESEARCH
KING        5000 ACCOUNTING
FORD        3000 RESEARCH
SQL>

```

13.26 3.26、列出在每个部门工作的员工数量,平均工资和平均服务期限.

Select d.dname 部门名称, count(e.empno) 员工数量, avg(e.sal) 平均工资 from emp e, dept d where e.deptno = d.deptno group by d.dname

Select d.dname 部门名称, count(e.empno) 员工数量, round(avg(e.sal),2) 平均工资, round(avg((sysdate-hiredate)/365), 0) 平均服务期限 from emp e, dept d where e.deptno = d.deptno group by d.dname

命令提示符 - sqlplus scott/tieger

部门名称	员工数量	平均工资	服务期限
ACCOUNTING	3	2916.67	28
RESEARCH	5	2175	26
SALES	6	1566.67	28

SQL>

### 13.27 3.27、列出所有员工的姓名、部门名称和工资。

Select e.ename, d.dname, e.sal from emp e, dept d where e.deptno = d.deptno

命令提示符 - sqlplus scott/tieger

ENAME	DNAME	SAL
SMITH	RESEARCH	800
ALLEN	SALES	1600
WARD	SALES	1250
JONES	RESEARCH	2975
MARTIN	SALES	1250
BLAKE	SALES	2850
CLARK	ACCOUNTING	2450
SCOTT	RESEARCH	3000
KING	ACCOUNTING	5000
TURNER	SALES	1500
ADAMS	RESEARCH	1100
JAMES	SALES	950
FORD	RESEARCH	3000
MILLER	ACCOUNTING	1300

已选择14行。  
SQL>

### 13.28 3.28、列出所有部门的详细信息和人数

Select d.\*, (select count(e.empno) from emp e where e.deptno = d.deptno) 人数 from dept d

命令提示符 - sqlplus scott/tieger

DEPTNO	DNAME	LOC	人数
10	ACCOUNTING	NEW YORK	3
20	RESEARCH	DALLAS	5
30	SALES	CHICAGO	6
40	OPERATIONS	BOSTON	0

SQL>

### 13.29 3.29、列出各种工作的最低工资及从事此工作的雇员姓名

命令提示符 - sqlplus scott/tieger

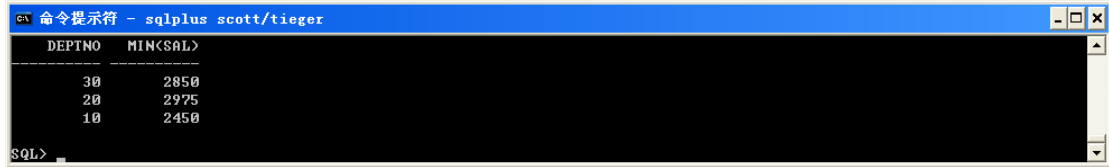
SQL> select e.\* from emp e join (select job,min(sal) sal from emp group by job) d on e.job=d.job and e.sal=d. sal;

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-12月-80	800		20
7654	MARTIN	SALESMAN	7698	28-9月-81	1250	1400	30
7521	WARD	SALESMAN	7698	22-2月-81	1250	500	30
7839	KING	PRESIDENT		17-11月-81	5000		10
7782	CLARK	MANAGER	7839	09-6月-81	2450		10
7902	FORD	ANALYST	7566	03-12月-81	3000		20
7788	SCOTT	ANALYST	7566	19-4月-87	3000		20

已选择7行。  
SQL>

### 13.30 3.30、列出各个部门的 MANAGER(经理)的最低薪金

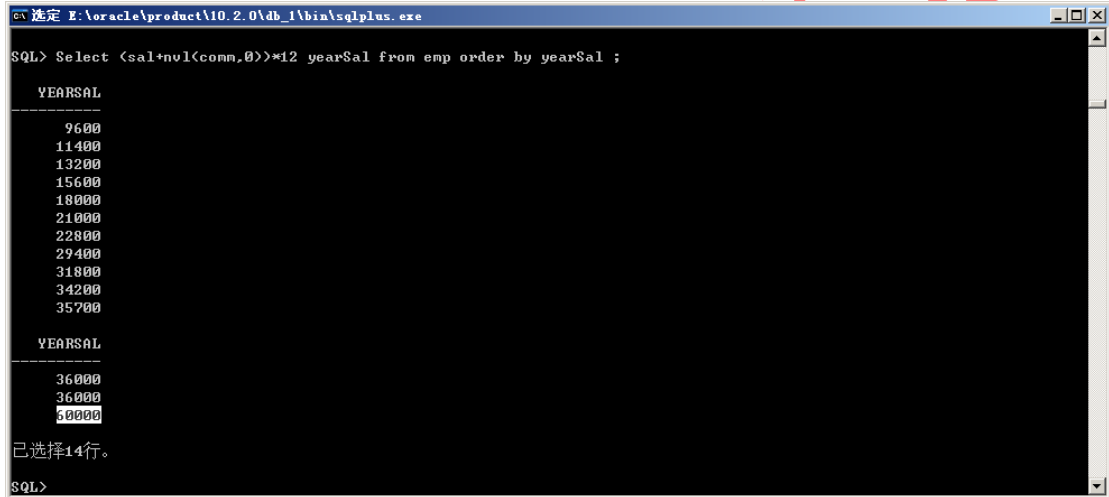
```
Select m.deptno, min(m.sal) from emp e, emp m where e.mgr = m.empno group by m.deptno
```



DEPTNO	MIN(SAL)
30	2850
20	2975
10	2450

### 13.31 3.31、列出所有员工的年工资,按年薪从低到高排序

```
Select (sal+nvl(comm,0))*12 yearSal from emp order by yearSal
```




YEARSAL
9600
11400
13200
15600
18000
21000
22800
29400
31800
34200
35700
36000
36000
50000

已选择14行。

### 13.32 3.32、查出某个员工的上级主管,并要求这些主管中的薪水超过 3000.

```
Select distinct m.ename, m.sal from emp e, emp m where e.mgr = m.empno and m.sal > 3000
```

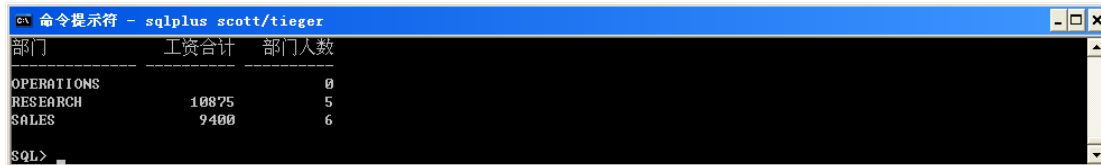


主管	主管工资
KING	5000

### 13.33 3.33、求出部门名称中,带'S'字符的部门, 员工的工资合计、部门人数.

```
Select d.dname, sum(e.sal), count(e.empno) from emp e right join dept d on e.deptno = d.deptno
```

where d.dname like '%S%' group by d.dname



命令提示符 - sqlplus scott/tiger

部门	工资合计	部门人数
OPERATIONS		0
RESEARCH	10875	5
SALES	9400	6

SQL>

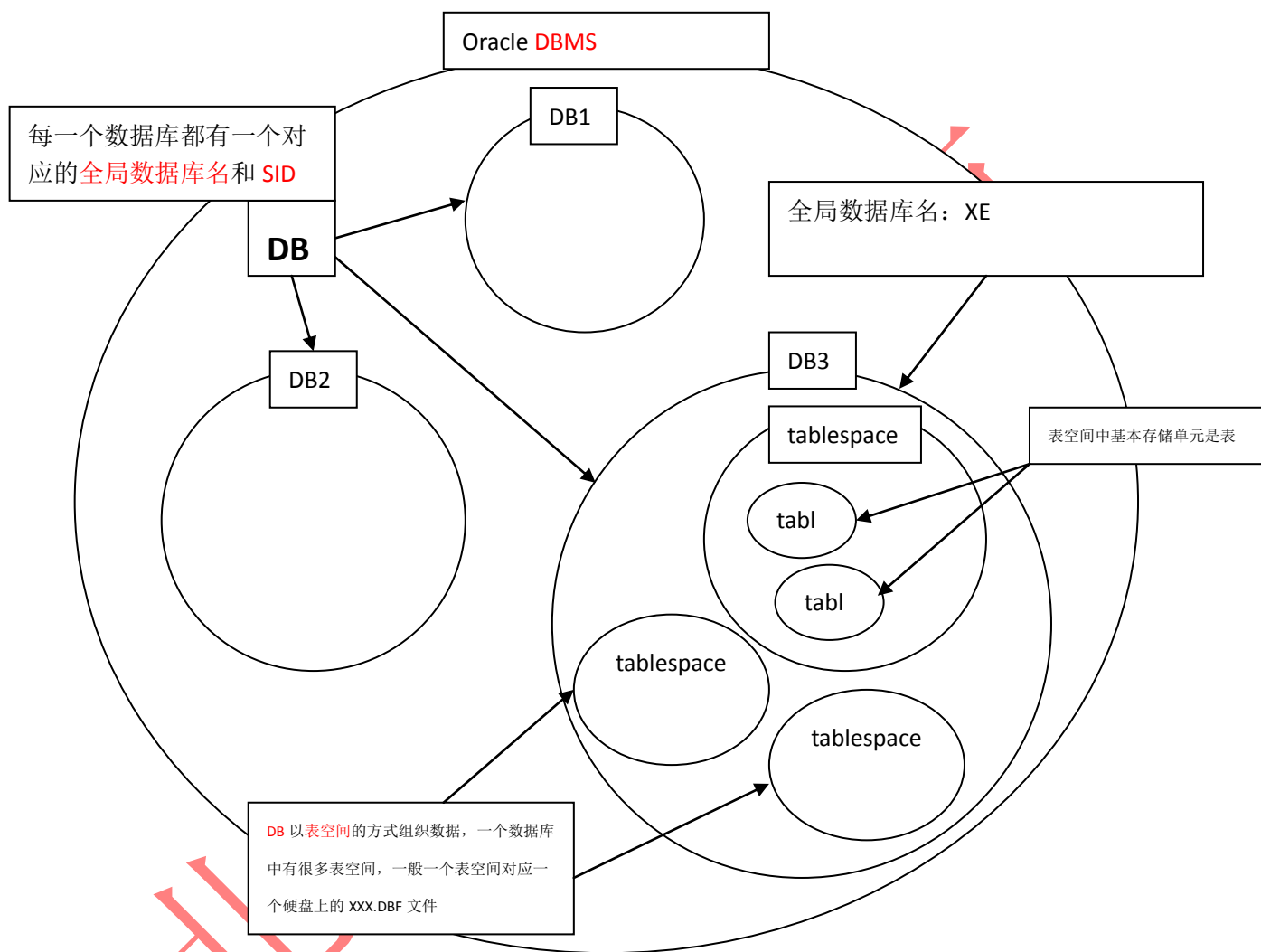
### 13.34 3.34、给任职日期超过 25 年的员工加薪 10%.

Select \* from emp\_bak where months\_between(sysdate, hiredate)/12 > 25

Update emp\_bak set sal = sal\*1.1 where months\_between(sysdate, hiredate)/12 > 25



## 附录 A Oracle 数据库管理系统结构分析



- 一个 DBMS 可以管理多个 DB
- 每一个 DB 都有自己的全局数据库名和 SID
- 一个 DB 中有多个表空间, 一个表空间通常对应一个 XXX.DBF 文件
- 在表空间中存储了表, 表中存储了数据
- Oracle 中通常会有一个默认的表空间, 这个表空间的逻辑名称是 USERS, 新建的用户如果没有指定他去使用哪个表空间, 会使用这个缺省的表空间。
- 通常开发初期数据库的准备需要完成以下操作:
  - 创建该项目专属的数据库
  - 在数据库中创建该项目专属的表空间
  - 创建用户
  - 让新建用户使用该表空间

---

■ 给新建用户授权

只要开发人员以新建的用户登录该数据库，该开发人员对该数据库的操作都在此表空间中

北京动力节点