

## 知识点列表

编号	名称	描述	级别
1	建立一个连接	掌握连接服务器命令 telnet，以及使用 Sqlplus 工具进入数据库	**
2	建数据表	掌握建表语句 create table	**
3	插入数据	掌握插入数据语句 insert	**
4	Sqlplus 命令	了解常用的 Sqlplus 命令	*
5	部门表结构	理解并记住部门表结构	*
6	员工表结构	理解并记住员工表结构	*
7	建表语句	掌握建表语句	***
8	学习查询语句	重点学习并掌握每一个查询案例	***

✓    "\*"理解级别    "\*\*\*"掌握级别    "\*\*\*\*"应用级别

## 目录

数据库简介 .....	4
【练习 1】把用户数据存入数据库的表中，并做查询.....	5
建立一个连接 ** .....	5
建数据表 ** .....	6
将用户数据插入数据表中 .....	7
【练习 2】Dept 表 && Emp 表数据准备.....	13
部门表结构 ** .....	13
员工表结构 ** .....	13
建表语句 ** .....	13
【练习 3】学习查询语句 *** .....	15
列的别名 .....	15
Sql 语句中函数的使用 .....	16
指定字段插入值 .....	18
复制表 .....	19
distinct 关键字 .....	20
where 条件查询 .....	20
大小写问题 .....	20
lower()函数 .....	20
upper()函数 .....	21
between ...and... 关键字 .....	21

in( 列表 ).....	22
模糊匹配 like %.....	22
is null.....	23
查询条件中的否定形式.....	24
SQL 语言的分类 *.....	24
数据定义语言.....	24
数据操纵语言.....	25
数据查询语言.....	25
事务控制语句.....	25

## 1. 数据库简介

### 1) 数据库( DataBase )

数据库是一种软件产品，是用于存放数据，管理数据的存储仓库，是有效组织在一起的数据集合。

### 2) 常用数据库软件

#### ■ 大型数据库

- ✓ **Oracle** Oracle 是著名的 Oracle(甲骨文)公司的数据库产品，它是世界上第一个商品化的关系型数据库管理系统，也是第一个推出和数据库结合的第四代语言开发工具的数据库产品。

Oracle 公司的软件产品丰富，包括 Oracle 服务器产品，Oracle 开发工具和 Oracle 应用软件。其中最著名的就是 Oracle 数据库，目前最新的版本是 Oracle 11g。

- ✓ **DB2** DB2 是 IBM 的关系型数据库管理系统，DB2 有很多不同的版本，可以运行在从掌上产品到大型机不同的终端机器上。DB2 在高端数据库的主要竞争对手是 Oracle。
- ✓ **Sybase** Sybase 是美国 Sybase 公司研制的一种关系型数据库系统，是较早采用 C/S 技术的数据库厂商，是一种典型的 UNIX 或 Windows NT 平台上客户机/服务器环境下的大型数据库系统，在国内大中型系统中具有广泛的应用。

#### ■ 中小型数据库

- ✓ **Sql Server** Microsoft SQL Server 是运行在 Windows NT 服务器上，支持 C/S 结构的数据库管理系统。它采用标准 SQL 语言。
- ✓ **Mysql** MySQL 是一个小型关系型数据库管理系统，开发者为瑞典 MySQL AB 公司。在 2008 年 1 月 16 号被 Sun 公司收购。而 2009 年 SUN 又被 Oracle 收购。MySQL 体积小、速度快、总体拥有成本低，尤其是开放源码，许多中小型网站为了降低网站总体拥有成本而选择了 MySQL 作为网站数据库。

#### ■ 小型数据库

- ✓ **Access** Microsoft Office Access(前名 Microsoft Access)是由微软发布的关联式数据库管理系统，是 Microsoft Office 的成员之一。

### 3) RDBMS( Relational Database Management System )

关系型数据库管理系统是数据库软件中用来操纵和管理数据库的部分，用于建立、使用和维护数据库，简称 rdbms。它对数据进行统一的管理和控制，以保证数据的安全性和完整性。

### 4) SQL( Structured Query Language )

SQL(Structured Query Language)语言是用来在关系数据库上执行数据操作、检索及维护所使用的标准语言，是一个综合的、通用的关系数据库语言。大多数数据库都使用相同或者相似的语言来操作和维护数据库。

SQL 语言可以用来查询数据，操纵数据，定义数据，控制数据，使软件开发人员、数据库管理员都可以通过 SQL 语言对数据库执行特定的操作。

5) **DBA**( Database Administrator )

数据库管理员

6) **Table**( 表 )

表是数据库存储的基本单元，对应于现实世界中的实体对象，比如部门、职员等，表是一个二维结构，由行和列组成，横向为行(Row)，也叫记录(Record)，用来表示实体的数据，比如一个职员的相关信息。纵向为列(Column)，也叫作字段(Field)，用来表示实体的属性，比如职员的薪水。

在软件开发技术比如 Java 中，现实世界的实体使用对象来描述，所以数据表和对象之间存在一种对应关系。数据表相当于类(Class)，数据表的行就是某个对象的实例(Instance)，其中每个列都是对象实例的属性(Field)。

## 2. 【练习 1】把用户数据存入数据库的表中，并做查询

### 2.1. 建立一个连接 \*\*

1) 数据库所在的服务器的 IP 地址( 现场班 ) : **telnet** 192.168.0.26

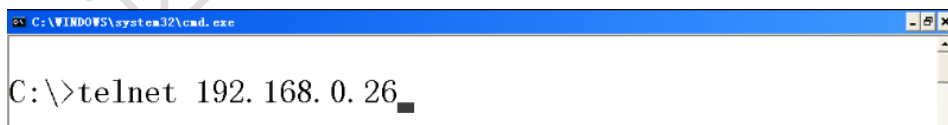
服务器的帐号/密码 : openlab/open123

2) 数据库访问用户/密码 : openlab/open123

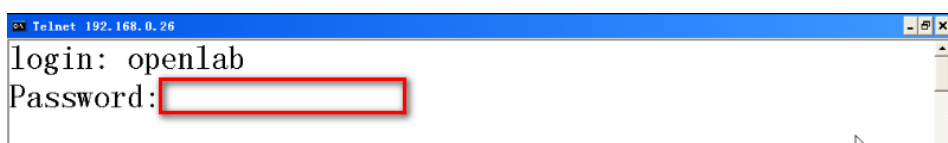
oracle 数据库的连接工具 **SQLPlus**

SQLPlus 是 Oracle 数据库提供的客户端工具。

#### 利用 telnet 工具连接服务器



输入用户名、密码( 密码不显示 )注意此步骤输入的是服务器的帐号和密码，不是数据库的。



登录成功后，用 Sql Plus 工具连接 Oracle 数据库

```
Telnet 192.168.0.26
login: openlab
Password:
Last login: Mon Oct 31 10:41:40 from 172.16.5.72
Sun Microsystems Inc. SunOS 5.10 Generic January
bash-3.00$ sqlplus openlab/open123
```

如果提示 SQL>表示 Oracle 数据库连接成功

```
Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.1
With the Partitioning, OLAP and Data Mining options

SQL>
```

## 2.2. 建数据表 \*\*

2.2.1. 定义数据结构( 这里 xxx 是代称，请改为自己能识别的字符串，比如：user\_007 )

**【案例 1】建表( User\_xxx )并插入数据**

```
-- 表名不能超过 30 个字符
-- 表名、列名是自由定义的
-- 所有的 SQL 语句都是以 “ ; ” 结尾

SQL> create table user_xxx(
        id          number(4) ,
        password    char(4) ,
        name        char(20) ,
        phone       char(20) ,
        email       varchar2(50)
    );
```

- ✓ 提示 **Table Created.** 表示表已成功创建
- ✓ 数据类型

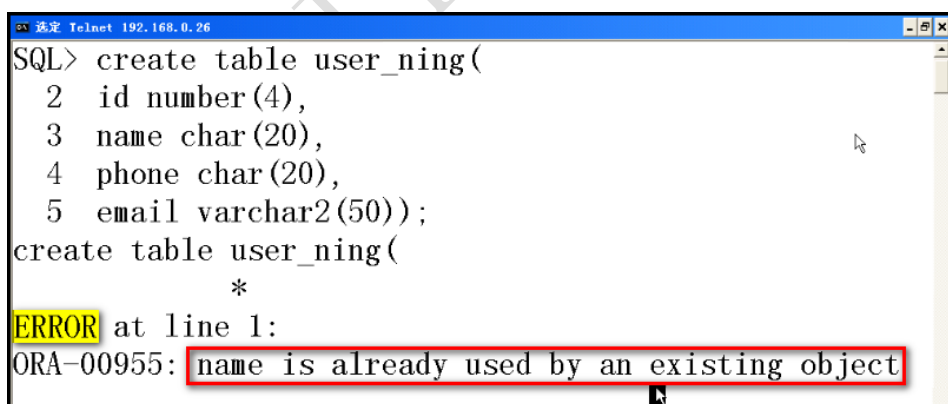
- **数字**
  - ✓ **number(n)**      数字( 最长 n 位 )
  - ✓ **number(n, m)**    浮点数( 总长 n 为 , 小数点后 m 位 )
  - 例 : number(7,2)    表示最大数为 99999.99
- **字符串**
  - ✓ **char(n)**      表示定长字符串( **方便查询** )  
最长放入 n 个字符 , 放入的数据如果不够 n 个字符则补空格 , 无论如何都占 n 个字符长度。
  - ✓ **varchar(n)**    表示变长字符串( **节省空间** )  
最长放入 n 个字符 , 放入的数据是几个长度就占多大空间
  - ✓ **varchar2(n)**    Oracle 自己定义的变长字符串
- **日期**
  - ✓ **date**      日期

### 2.2.2. 清屏命令

```
SQL> clear scr      -- scr 代表 screen
```

### 2.2.3. 常见错误

数据库中已经有了同名的表



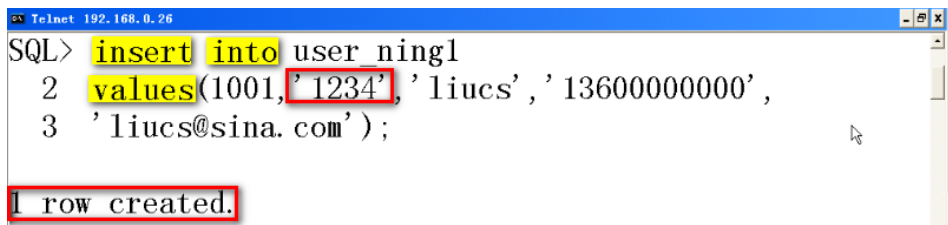
```
SQL> create table user_ning(
2  id number(4),
3  name char(20),
4  phone char(20),
5  email varchar2(50));
create table user_ning(
      *
ERROR at line 1:
ORA-00955: name is already used by an existing object
```

## 2.3. 将用户数据插入数据表中

**【案例 2】插入数据 , 并查找 id=1001 && passwd=1234 考生的名字**

### 2.3.1. 插入数据 \*\*

```
SQL> insert into user_xxx
      values(1001, '1234', 'liucs', '13600000000', 'liucs@sina.com');
```



```
SQL> insert into user_ning1
2 values(1001, '1234', 'liucs', '13600000000',
3 'liucs@sina.com');
```

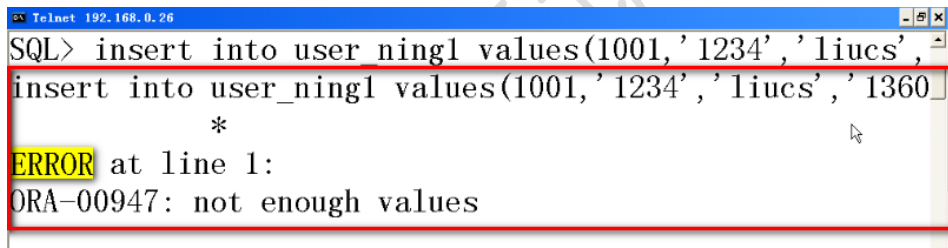
1 row created.

✓ 数据库中字符用单引号表示，数字不用。

### 2.3.2. 常见错误

#### 1) values 少了一项

```
SQL> insert into user_ning1 values(1001, '1234', 'liucs', '13600000000');
```

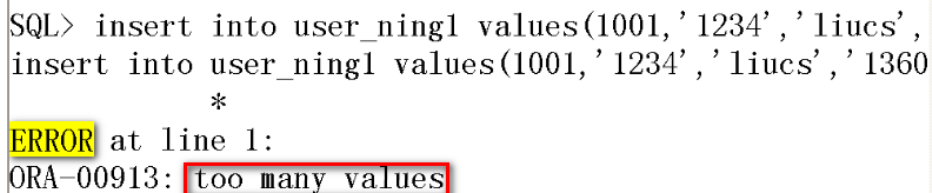


```
SQL> insert into user_ning1 values(1001, '1234', 'liucs',
insert into user_ning1 values(1001, '1234', 'liucs', '1360
*
```

ERROR at line 1:  
ORA-00947: not enough values

#### 2) values 多了一项

```
SQL> insert into user_ning1
      values(1001, '1234', 'liucs', '13600000000', 'my email', 25);
```



```
SQL> insert into user_ning1 values(1001, '1234', 'liucs',
insert into user_ning1 values(1001, '1234', 'liucs', '1360
*
```

ERROR at line 1:  
ORA-00913: too many values

### 2.3.3. 查询数据



```
SQL> select * from user_ning1;
```

ID	PASS	NAME	PHONE
1001	1234	liucs	13600000000

EMAIL  
liucs@sina.com

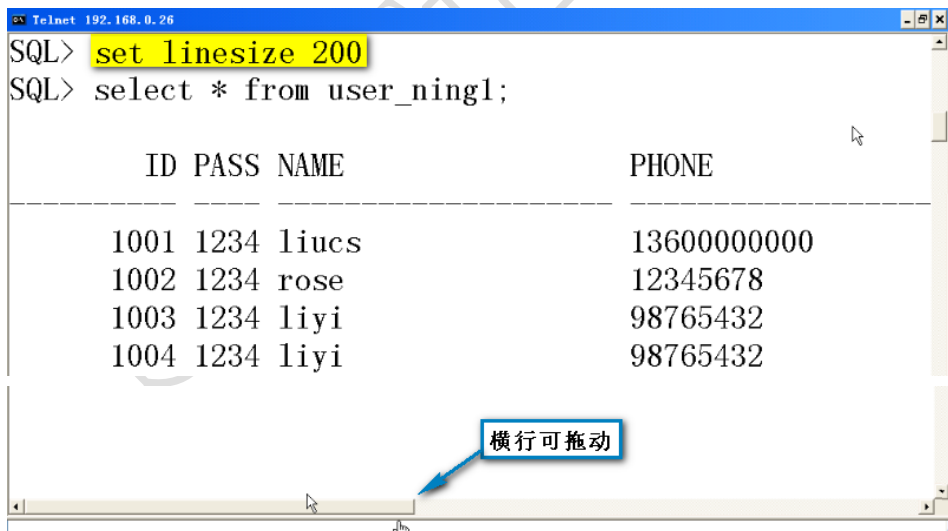
#### 2.3.4. 三种 SQL 语句总结

- 1) **建表**                create table 表名(列名 列类型 ,.....)
- 2) **插入数据**        insert into 表名 values(列值 ,.....)
- 3) **查询**              select \* from 表名 ;

#### 2.3.5. Sqlplus 命令 \*

##### 1) 设置行的长度

显示 200 个字符再换行



```
Telnet 192.168.0.26
SQL> set linesize 200
SQL> select * from user_ning1;
```

ID	PASS	NAME	PHONE
1001	1234	liucs	13600000000
1002	1234	rose	12345678
1003	1234	liyi	98765432
1004	1234	liyi	98765432

横行可拖动

- ✓ **注意：** Sqlplus 命令不需要写 “;” 分号
- ✓ SQL 语句必须以分号结束，Sqlplus 命令可加可不加分号

##### 2) 设置列的宽度

- 如果要设置的列为字符

设置列( Column )name 的宽度 6 个字符( a6 表示 6 个字符 )

```
SQL> column name format a6
```

ID	PASS	NAME	PHONE	EMAIL
1001	1234	liucs	13600000000	liucs@sina.

**注意：**此 Sqlplus 命令不能设置数字格式( 如 id )，字符和数字在 Oracle 中是区别对待的

**错误做法** SQL> column id format a6

设置 id 属性，会显示为 “#” 符号

```
SQL> select * from user_ning1;
```

ID	PASS	NAME	PHONE	EMAIL
#####	1234	liucs	13600000000	liucs@sina.com

### ■ 属性列为数字

显示 4 个数字，格式定义为 4 个 9.

```
SQL> column id format 9999
SQL> col id for 9999
SQL> select * from user_ning1;
```

ID	PASS	NAME	PHONE	EMAIL
1001	1234	liucs	13600000000	liucs@sina.com

✓ column 可简写为 col ; format 可简写为 for

### 3) 查看表结构

```
SQL> desc user_xxx --desc 表名
```

注意：设置的命令仅当前窗口有效，关闭窗口重新连接仍按默认方式显示

4) “/” 表示执行上一条 Sql 语句，忽略 sql 命令。

```
SQL> set linesize 200
SQL> select * from user_nginx;

      ID PASS NAME                PHONE
-----
1001 1234 liucs                13600000000
1002 1234 rose                 12345678
1003 1234 liyi                 98765432
1004 1234 liyi                 98765432

SQL> column name format a6
SQL> /
```

5) 设置分页显示

```
SQL> set pagesize 100 --每 100 行数据分页显示
SQL> set pages 0 --pagesize 可以简写为 pages,设置为 0 表示不分页
```

### 2.3.6. 查询语句

```
SQL> select * from user_xxx ; -- "*" 星号表示查询全部列
SQL> select name , email from user_xxx ; -- 查询指定列的数据
```

### 2.3.7. 条件查询

查找 id=1001 且 passwd=1234 考生的名字

1) 查询所有人的 name 和 email

```
SQL> select name , email from user_xxx ; -- 查询指定列的数据
```

NAME	EMAIL
liucs	liucs@sina.com
rose	rose@163.com
liyi	liyi@sina.com
liyi	liyi@sina.com

2) 查找 id=1001 且 passwd=1234 考生的名字

```
SQL> select name , email from user_xxx
      where id = 1001 and password = '1234' ;
```

NAME	EMAIL
liucs	liucs@sina.com

**Select**

select ename,sal  
from emp;

→

select \*  
from emp  
where deptno = 10;

→

### 3. 【练习 2】Dept 表 && Emp 表数据准备

#### 3.1. 部门表结构 \*\*

部门表 Dept XXX		
部门编码 deptno Number(2)	部门名字 dname Char(20)	部门所在位置 Location char(20)
10	研发部	北京
20	财务部	上海
30	销售部	广州
40	后勤部	天津

#### 3.2. 员工表结构 \*\*

员工表 Emp XXX							
编码	姓名	职位	薪水	奖金	入职时间	经理	所在部
1001	张无忌	Manager	10000	2000	12-MAR-10	1005	10
1002	刘苍松	Analyst	8000	1000	01-APR-11	1001	10
1003	李翊	Analyst	9000	1000	11-APR-10	1001	10
1004	郭芙蓉	Programmer	5000		01-JAN-11	1001	10
1005	张三丰	President	15000		15-MAY-08		20
1006	燕小六	Manager	5000	400	01-FEB-09	1005	20
1007	陆无双	Clerk	4000	500	01-FEB-09	1006	20
1008	黄蓉	Manager	5000	800	01-MAY-09	1005	30
1009	韦小宝	Salesman	4000		20-FEB-09	1008	30
1010	郭靖	Salesman	4500		10-MAY-09	1008	30

#### 3.3. 建表语句 \*\*

**【案例 3】建表 Dept\_xxx 和表 Emp\_xxx(注意把 xxx 改成自己能识别的字符串，比如 dept\_007)****1) Dept 表**

```
SQL> drop table dept_xxx; --删除语句( 注意请不要删除其他同学创建的表 )
SQL> create table dept_xxx(
            deptno    number(2),
            dname      char(20),
            location    char(20));

SQL> insert into dept_xxx values(10, 'developer', 'beijing');
SQL> insert into dept_xxx values(20, 'account', 'shanghai');
SQL> insert into dept_xxx values(30, 'sales', 'guangzhou');
SQL> insert into dept_xxx values(40, 'operations', 'tianjin');
SQL> commit; --事务控制语句
SQL> select * from dept_xxx;
```

**2) Emp 表**

```
SQL> create table emp_xxx(
            empno      number(4),
            ename       varchar2(20),
            job         varchar2(15),
            salary      number(7, 2),
            bonus       number(7, 2),
            hiredate    date,
            mgr         number(4),
            deptno      number(10)
        );

SQL> insert into emp_xxx values(
            1001, '张无忌', 'Manager',
            10000, 2000, '12-MAR-10', 1005, 10);
SQL> insert into emp_xxx values(
            1002, '刘苍松', 'Analyst',
            8000, 1000, '01-APR-11', 1001, 10);
SQL> insert into emp_xxx values(
            1003, '李翊', 'Analyst',
            9000, 1000, '11-APR-10', 1001, 10);
```

```
SQL> insert into emp_xxx values(
        1004 , '郭芙蓉' , 'Programmer' ,
        5000 , null , '01-JAN-11' , 1001 , 10) ;
SQL> insert into emp_xxx values(
        1005 , '张三丰' , 'President' ,
        15000 , null , '15-MAY-08' , null , 20) ;
SQL> insert into emp_xxx values(
        1006 , '燕小六' , 'Manager' ,
        5000 , 400 , '01-FEB-09' , 1005 , 20) ;
SQL> insert into emp_xxx values(
        1007 , '陆无双' , 'clerk' ,
        3000 , 500 , '01-FEB-09' , 1006 , 20) ;
SQL> insert into emp_xxx values(
        1008 , '黄蓉' , 'Manager' ,
        5000 , 500 , '1-MAY-09' , 1005 , 30) ;
SQL> insert into emp_xxx values(
        1009 , '韦小宝' , 'salesman' ,
        4000 , null , '20-FEB-09' , 1008 , 30) ;
SQL> insert into emp_xxx values(
        1010 , '郭靖' , 'salesman' ,
        4500 , 500 , '10-MAY-09' , 1008 , 30) ;

SQL> set linesize 150          -- 设置行长度为 150
SQL> col empno for 9999        -- 设置 empno 显示 4 位
SQL> col mgr for 9999          -- 设置 mgr 显示 4 位
SQL> col deptno for 99         -- 设置 deptno 显示 2 位
SQL> col salary for 99999.99   -- 设置 salary 显示格式

SQL> select * from dept_xxx ;
SQL> select * from emp_xxx ;
```

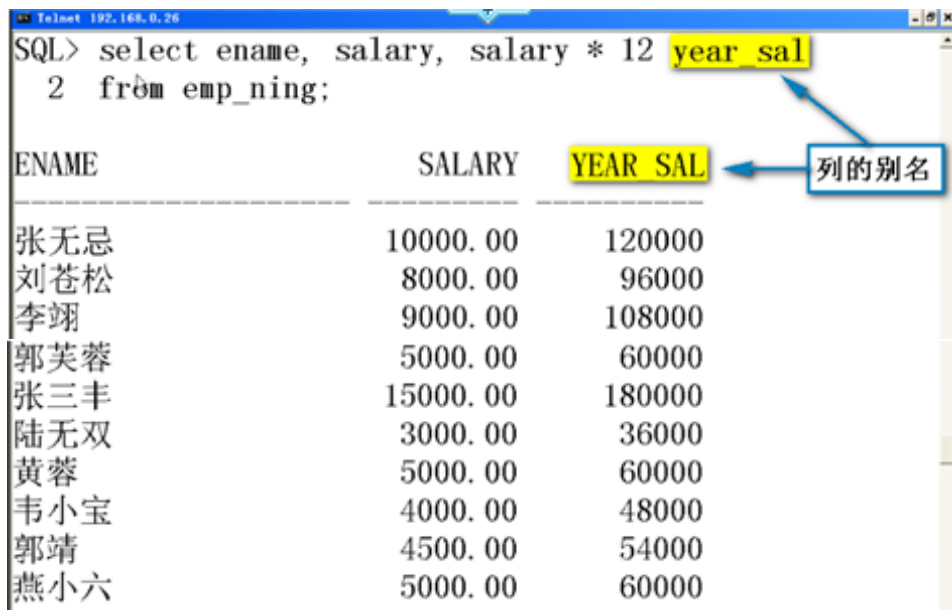
## 4. 【练习 3】学习查询语句 \*\*\*

### 4.1. 列的别名

#### 【案例 4】计算员工的名字、月薪和年薪？

```
SQL> select ename , salary , salary * 12  year_sal
      from emp_xxx ;
```

-- year\_sal 是 salary\*12 的别名



ENAME	SALARY	YEAR SAL
张无忌	10000.00	120000
刘苍松	8000.00	96000
李翊	9000.00	108000
郭芙蓉	5000.00	60000
张三丰	15000.00	180000
陆无双	3000.00	36000
黄蓉	5000.00	60000
韦小宝	4000.00	48000
郭靖	4500.00	54000
燕小六	5000.00	60000

## 4.2. Sql 语句中函数的使用

### 4.2.1. Oracle 中空值的概念：null

- 1) 任何数据类型都可以取值 null
- 2) 空值和任何数据做算数运算，结果都是 null。
- 3) 空值和字符串类型做连接操作，结果相当于空值不存在。

#### 【案例 5】计算员工的月收入？

**注意：**空值和任何数据做算数运算，结果为空(null)

■ 下列写法得不到正确的结果：

```
SQL> select ename , salary , bonus , salary+bonus  month_sal
      from emp_xxx ;
```

--错误的写法



ENAME	SALARY	BONUS	MONTH_SAL
张无忌	10000.00	2000	12000
刘苍松	8000.00	1000	9000
李翊	9000.00	1000	10000
郭芙蓉	5000.00		
张三丰	15000.00		
陆无双	3000.00	500	3500
黄蓉	5000.00	500	5500
韦小宝	4000.00		
郭靖	4500.00	500	5000
燕小六	5000.00	400	5400

#### 4.2.2. 处理空值的函数 nvl，使用方式：nvl(bonus, 0)

- **正确写法：**如果 bonus 的值是 null，则取 0

```
SQL> select ename, salary, bonus, salary + nvl(bonus, 0) month_sal
      from emp_xxx;
```

ENAME	SALARY	BONUS	MONTH_SAL
张无忌	10000.00	2000	12000
刘苍松	8000.00	1000	9000
李翊	9000.00	1000	10000
郭芙蓉	5000.00		5000
张三丰	15000.00		15000
陆无双	3000.00	500	3500
黄蓉	5000.00	500	5500
韦小宝	4000.00		4000
郭靖	4500.00	500	5000
燕小六	5000.00	400	5400

- ✓ nvl(d1, d2)方法演示：如果 d1 为 null 则用 d2 替代
- ✓ nvl 函数的两个参数可以是数字、字符或日期，但两个参数的数据类型必须一致。
- ✓ 以下 java 程序只是说明 nvl 函数的功能，它和 java 中的方法功能类似，可以有三种数据类型的参数，类似于 java 中的方法重载。但实际上这个函数并非使用 java 写的。

```

Oracle.java X
1 package tarena.elts;
2 import java.util.Date;
3
4 //nvl() 方法的作用演示
5 public class Oracle {
6     public double nvl(Double d1, Double d2){
7         return d1 != null ? d1 : d2 ;
8     }
9     public String nvl(String s1,String s2)
10    {
11        return s1 != null ? s1 : s2 ;
12    }
13    public Date nvl(Date d1, Date d2){
14        return (d1 != null) ? d1 : d2 ;
15    }
16 }

```

#### 4.3. 指定字段插入值

【案例 6】插入一条 ID 为 1011，姓名为 '余泽成'，其余字段为 null 的数据

繁琐

```
SQL> insert into emp_xxx
      values( 1011, '余泽成', null, null, null, null, null, null );
```

简写

```
SQL> insert into emp_xxx( empno, ename )
      values( 1011, '余泽成' );
```

【案例 7】查询 emp\_xxx 表，如果没有职位，显示 'no position'，如果有职位，显示员工的职位

```
SQL> select ename, nvl(job, 'no position') from emp_xxx ;
```

姓名	职位
张无忌	Manager
刘苍松	Analyst
李翊	Analyst
郭芙蓉	Programmer
张三丰	President

陆无双	clerk
黄蓉	Manager
韦小宝	salesman
郭靖	salesman
燕小六	Manager
余泽成	no position

#### 4.3.1. 日期类型 `nvl(hiredate, '10-OCT-11')`

**【案例 8】** 查询 `emp_xxx` 表，如果没有入职时间，显示为 2011 年 10 月 10 日，否则原样显示

```
SQL> select ename , nvl( hiredate , '10-OCT-11' ) from emp_xxx ;
```

#### 【扩展练习 1】

```
SQL> select empno, ename || job detail from emp_xxx ;
```

```
-- "||" 符号表示两个数据串接起来，类似于 Java 中的两个字符串之间的+号
-- 上例是把 ename 和 job 两个字符串连接起来
-- 如果 job 是 null，相当于不串接任何字符
```

#### 【扩展练习 2】

```
SQL> select empno , ename || ' work as ' || job detail from emp_xxx ;
-- 串接的两个字段之间有常量字符，每行显示一次
```

### 4.4. 复制表

**【案例 9】** 复制表 `emp_YYY` 为 `emp_XXX`

```
SQL> create table emp_xxx as select * from emp_yyy ;
```

## 4.5. distinct 关键字

**注意：**distinct 必须( 只能 )跟在 select 后边

### 【案例 10】机构中有多少种职位？

```
SQL> select distinct job from emp_xxx ;
```

### 【案例 11】员工分布在哪些部门？

```
SQL> select distinct deptno from emp_xxx ;
```

### 【扩展练习】查询每个部门不重复的职位

```
SQL> select distinct deptno, job from emp_xxx ;  
--distinct 指所有列的唯一组合
```

## 4.6. where 条件查询

### 【案例 12】薪水高于 10000 元的员工数据？

```
SQL> select * from emp_xxx where salary > 10000 ;
```

## 4.7. 大小写问题

### 【案例 13】职位是 Analyst 的员工数据？

```
SQL> select * from emp_xxx where job = 'Analyst' ;  
--如果数据是 analyst，查不出结果
```

**注意：**SQL 语句大小写不敏感，数据大小写敏感

## 4.8. lower()函数

将字符数据转换为小写( 如果不知道职位的大小写形式，可以使用 lower 函数，忽略大小写 )

#### 【案例 14】查找职位为 “analyst” 的员工

```
SQL> select * from emp_xxx where lower(job) = 'analyst' ;
```

### 4.9. upper()函数

将数据转换为大写

#### 【案例 15】忽略大小写，查找职位为 “analyst” 的员工

```
SQL> select * from emp_xxx where upper(job) = 'ANALYST' ;
```

### 4.10. between ...and... 关键字

- 1) 在区间中：between 低值 and 高值
- 2) 闭区间：[低值，高值]

#### 【案例 16】薪水大于 5000 并且小于 10000 的员工数据？

```
SQL> select * from emp_xxx where salary >= 5000 and salary <= 10000 ;
```

等同于

```
SQL> select * from emp_xxx where salary between 5000 and 10000 ;
```

EMPNO	ENAME	JOB	SALARY
1001	张无忌	Manager	10000.00
1002	刘苍松	Analyst	8000.00
1003	李翊	Analyst	9000.00
1004	郭芙蓉	Programmer	5000.00
1008	黄蓉	Manager	5000.00
1006	燕小六	Manager	5000.00

6 rows selected.

**【案例 17】入职时间在 2011 年的员工？**

```
SQL> select * from emp_xx
      where hiredate between '01-JAN-11' and '31-DEC-11';
      --闭区间( 包括边界值 ) : ['01-JAN-11', '31-DEC-11']
```

EMPNO	ENAME	JOB	SALARY
1002	刘苍松	Analyst	8000. 00
1004	郭芙蓉	Programmer	5000. 00

**4.11. in( 列表 )**

**【案例 18】列出职位是 Manager 或者 Analyst 的员工**

```
SQL> select * from emp_xxx
      where job = 'Manager' or job = 'Analyst';
```

等同于

```
SQL> select * from emp_xxx
      where job in ('Manager', 'Analyst');
```

**4.12. 模糊匹配 like %**

- 1) “%” 表示 0 到多个字符，跟 like 配合使用
- 2) “\_” 下划线表示一个字符

**【案例 19】列出职位中包含有 sales 字符的员工数据？**

```
SQL> select * from emp_xxx where lower(job) like '%sales%';
```

EMPNO	ENAME	JOB	SALARY
1009	韦小宝	salesman	4000. 00
1010	郭靖	salesman	4500. 00

**【案例 20】列出职位中第二个字符是 a 的员工数据？**

```
SQL> select * from emp_xxx where job like '_a%';
```

**【案例 21】查询数据库中有多少个名字中包含 'EMP' 的表？**

```
SQL> select count(*) from user_tables where table_name like '%EMP%';
```

**【案例 22】查询数据库中有多少个名字中以 'S\_' 开头的表？**

```
SQL> select count(*) from user_tables
      where table_name like 'S\_%' escape '\';

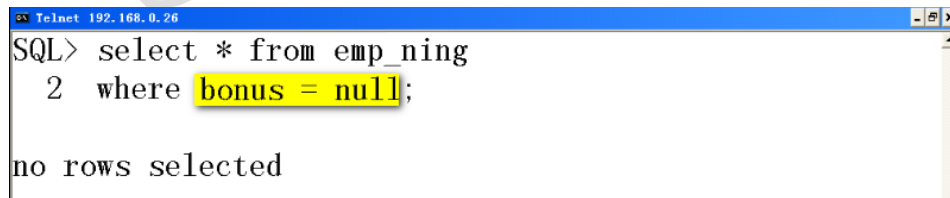
-- 如果要查询的数据中有特殊字符( 比如_或% ),
-- 在做模糊查询时 ,
-- 需要加上\符号表示转义 , 并且用 escape 短语指明转义字符\
```

### 4.13. is null

数据库语言判断 null 值的方法。

**【案例 23】查询哪些员工没有奖金？**

■ **错误写法**



```
SQL> select * from emp_ning
2  where bonus = null;

no rows selected
```

■ **正确写法**

```
SQL> select * from emp_xxx where bonus is null ;
```

EMPNO	ENAME	JOB	SALARY
1004	郭芙蓉	Programmer	5000.00
1005	张三丰	President	15000.00
1009	韦小宝	salesman	4000.00
1011	余泽成		

## 4.14. 查询条件中的否定形式

### 4.14.1. is not null

数据库语言判断不是 null 值的方法。

#### 【案例 24】哪些员工有奖金？

```
SQL> select * from emp_xxx where bonus is not null ;
```

### 4.14.2. not between 低值 and 高值

#### 【案例 25】薪水不在 5000 至 8000 的员工？

```
SQL> select * from emp_xxx where salary not between 5000 and 8000 ;
```

### 4.14.3. not in (list) : 不在列表中

#### 【案例 26】不是部门 20 和部门 30 的员工？

```
SQL> select * from emp_xxx where depno not in( 20 , 30 ) ;
```

## 5. SQL 语言的分类 \*

### 5.1. 数据定义语言



数据定义语言 **DDL**( Data Definition Language )，是 SQL 语言集中负责数据结构定义与数据库对象定义的语言，主要有 create、alter、drop 和 truncate 四种常用语句。

DDL 对**数据结构**起作用。

- ✓ **create**                      数据库对象的创建
- ✓ **alter**                      修改数据库对象
- ✓ **drop**                      删除数据库对象
- ✓ **truncate**                清空表数据

## 5.2. 数据操纵语言

数据操纵语言 **DML**( Data Manipulation Language )，用户通过它可以实现对数据表的基本操作，即对表中数据的增、删、改。

DML 对**数据**起作用。

- **insert**                    插入操作
- **update**                  更新操作
- **delete**                  删除操作

## 5.3. 数据查询语言

数据查询语言 **DQL**( Data Query Language )，用户主要通过它实现对数据的查询操作。

- **select**                    查询操作

## 5.4. 事务控制语句

事务控制语句是用来对 DML 操作进行确认的。

- 3) **commit**                提交数据
- 4) **rollback**              数据回滚
- 5) **savepoint**            保存点