oracle

一：安装介绍

1、版本

> Oracle9i **10g** 11g 12c

2、安装服务器

```
oracle10g服务器在win7下安装的注意点：
单击setup.exe，运行时需要使用管理员的方式，并且以兼容xp的模式
```

3、使用终端测试一下数据库是否安装成功

```
在终端运行sqlplus sys as sysdba  pwd
-->输入sql语句
select sysdate from  dual;

SYSDATE
--------------
07-8月 -18

SQL>
--表示安装成功
```

4、与mysql数据库的区别

```
oracle通过不同的账号和表空间来划分不同的业务/项目数据
而mysql是每一个项目对应不同的数据库
oracle只有一个数据库，这个数据库叫全局数据库实例:orcl
```

5、oracle的账号

```
安装时有两个特殊的账号:sys    system
sys:数据库的创建者/拥有者
                  可以创建管理员
                  卸载、安装
                  停止、启动
                  修改数据库的基本配置信息

system：数据库的管理员
                  不能复制自己的身份
                  只能管理账号下的表,视图，函数等等数据组件
```

6、2个重要的服务

```
1:oracleServiceXXX(XXX就是数据库实例名)
2:oracleService-home01TNSListener(远程监听连接器--通过主机名:端口号访问)
如果这个服务不启动，那么该数据库只能本机访问，不能远程访问
```

7、相关的数据库文件

```
1:数据文件
        表空间名+序号.DBF   （database file）
2:日志文件
    .LOG
3:控制文件
        .CTL   （controll）
```

8、各种数据库的比较

```
大型
        oracle(Oracle)
        db2(IBM  International Business Machine)

中型
        MSsql(Microsoft)
        MySql(oracle)

小型
        Access(Microsoft)
```

二：用户权限

1、登录

```
在终端输入：sqlplus  sys as sysdba  pwd

--如何查看环境变量path的值？
win:echo %path%
linux  echo $path
```

2、退出

```
exit;  退出sqlplus,回到操作系统界面
disc;  退出当前会话，切换了身份，show user 显示用户为""
```

3、数据字典表

```
如何查看与数据库相关的一些系统信息？---> 使用数据字典表
数据字典表：用于存储系统数据的表
分类
dba_      具有dba角色的用户可以查看到的全局数据对象信息
user_     普通账号可以查看自己账号下的相关数据信息
```

4、在sql命令行下连接数据库

> 使用**conn**命令

```
SQL> conn sys as sysdba
输入口令:
已连接。
SQL>
```

5、数据字典表----用户信息表

```
dba_users---用户信息表
SQL> desc dba_users;
 名称                                      是否为空? 类型
 ----------------------------------------- -------- ----------------
 USERNAME                                  NOT NULL VARCHAR2(30)
 USER_ID                                   NOT NULL NUMBER
 PASSWORD                                           VARCHAR2(30)
 ACCOUNT_STATUS                            NOT NULL VARCHAR2(32)
 LOCK_DATE                                          DATE
 EXPIRY_DATE                                        DATE
 DEFAULT_TABLESPACE                        NOT NULL VARCHAR2(30)
 TEMPORARY_TABLESPACE                      NOT NULL VARCHAR2(30)
 CREATED                                   NOT NULL DATE
 PROFILE                                   NOT NULL VARCHAR2(30)
 INITIAL_RSRC_CONSUMER_GROUP                        VARCHAR2(30)
 EXTERNAL_NAME                                      VARCHAR2(4000)

SQL> select username,account_status from dba_users;

USERNAME                       ACCOUNT_STATUS
------------------------------ ------------------------
MGMT_VIEW                      OPEN
SYS                            OPEN
SYSTEM                         OPEN
DBSNMP                         OPEN
SYSMAN                         OPEN
SCOTT                          OPEN
MIKE_                          OPEN
OUTLN                          EXPIRED & LOCKED
MDSYS                          EXPIRED & LOCKED
ORDSYS                         EXPIRED & LOCKED
EXFSYS                         EXPIRED & LOCKED
```

- locked:表示该账号被锁定，不能被访问
- expired:表示该账号已过期，登录后必须立即修改密码

6、加锁

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

7、解锁

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

8、强制密码过期

```
alter user scott password expire;
```

9、当前账号强制修改密码

```
password  必须在当前账号下调用
```

10、dba修改密码

```
alter user scott identified by abc;
```

11、创建用户

```
create user mike identified by abc;
```

```
SQL> conn mike/abc;
ERROR:
ORA-01045: user MIKE lacks CREATE SESSION privilege;
```

问题：

创建的mike用户，登录后缺少create session权限

也就是说光有账号还不行，还必须有登录权限

12、授权登录

```
grant create session to mike;
```

```
已连接。
SQL> grant create session to mike;

授权成功。

SQL> conn mike/abc@192.168.7.8:1521/orcl;
已连接。
SQL>
```

13、创建表

```
SQL> conn mike/abc;
已连接。
SQL> create table t_user(id int,name varchar(10));
create table t_user(id int,name varchar(10))
*
第 1 行出现错误:
ORA-01031: 权限不足
```

问题：无法创建表，没有创建表的权限

解决方案：赋予mike账号创建表的权限

```
grant create table to mike;
```

```
SQL> conn mike/abc;
已连接。
SQL> create table t_user(id int,name varchar(10));
create table t_user(id int,name varchar(10))
*
第 1 行出现错误:
ORA-01950: 对表空间 'USERS' 无权限
```

问题：缺少表空间

解决方案:授予表空间权限给mike

```
alter user mike quota unlimited on users;
```

```
SQL> create table t_user(id int,name varchar(10));

表已创建。
```

```
SQL> insert into t_user values(1,'mike');

已创建 1 行。

SQL> select * from t_user;

        ID NAME
---------- ----------
         1 mike
```

14、角色

角色：是一组已经定义好的权限的集合

我们只要授予这个角色给用户，就相当于将该角色下的所有权限授予该用户

- 创建角色

```
create role r1;
```

- 归纳一组权限到该角色下

```
grant create session ,create table to r1;
```

- 再将该角色授予用户

```
SQL> create user mike1 identified by abc;

用户已创建。

SQL> grant r1 to mike1;

授权成功。

SQL> conn mike1/abc;
已连接。
SQL> create table t_1(id int);
create table t_1(id int)
*
第 1 行出现错误:
ORA-01950: 对表空间 'USERS' 无权限
```

- 系统默认提供的角色

  系统中有一组已经预定义好的角色，可以直接让管理员使用

  connect

  resource

  ```
  grant connect,resource to mike2;
  ```

```
SQL> conn sys as sysdba
输入口令:
已连接。
SQL> create user mike2 identified by abc;

用户已创建。

SQL> grant connect,resource to mike2;

授权成功。

SQL> conn mike2/abc
已连接。
SQL> create table t_1(id int,name varchar(1

表已创建。

SQL> insert into t_1 values(1,'mike');

已创建 1 行。
```

SQL> select * from t_1;

```
        ID NAME
```

---------- ----------

```
         1 mike
```

* 撤销(收回)相关的系统权限

```sql
revoke resource from mike2;
```

```
SQL> revoke resource from mike2;

撤销成功。

SQL> conn mike2/abc
已连接。
SQL> create table t_2(id int);
create table t_2(id int)
*
第 1 行出现错误:
ORA-01031: 权限不足
```

注意：在回收权限时，如果是角色授予，也要相应的回收该角色，而不是回收该角色中的某一个权限.

15、对象权限

刚才讨论的是系统权限，下面讨论基于数据对象(例如表.....)的权限

```
SQL> conn mike/abc
已连接。
SQL> select * from scott.emp;
select * from scott.emp
                    *
第 1 行出现错误:
ORA-00942: 表或视图不存在
```

```
grant select,insert,update,delete on emp to mike;
grant all on emp to mike;
```

```
SQL> conn scott/abc
已连接。
SQL> grant select on emp to mike;

授权成功。

SQL> conn mike/abc
已连接。
SQL> select * from scott.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                    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。
```

16、常见的数据字典表

```
dba_users  --当前数据库下的账号表(dba访问)

user_tab_privs  --当前账号下的对象权限表

user_role_privs  ---当前账号下角色表

user_sys_privs;  ---当前账号下的系统权限表

user_tables   ---当前账号下有哪些表

user_tab_comments ---当前账号下的数据对象的注释

user_col_comments ---当前账号下的表的字段的注释

user_sequences   --当前账号下的序列组件

user_开头的表是用户级别的表，当前账号可以使用
dba_开头的表是dba级别的表，必须只有dba(数据库管理员)才能访问


select * from user_tab_privs;

select * from scott.salgrade;

select * from user_role_privs;

select * from user_sys_privs;

select * from user_tables;

select * from user_tab_comments;

--如何在sys账号下删除某个账号
drop user mike2 cascade;
--在sys账号下查看所有账号
select username from dba_users;
```

三：oracle客户端

```
sqldeveloper4.*
```

- windows下运行

  直接双击：sqldeveloper.exe

- linux下运行

```
解压zip包
让sqldeveloper.sh变为一个可执行命令
cd sqldeveloper.sh
chmod u+x sqldeveloper.sh
./sqldeveloper.sh  &
./sqldeveloper.sh
```

四：数据类型

```
number
    int
    float
    number(5,2)--范围[-999.99-999.99]

varchar(最多可存放4000个字节)
字符串的精度
char(10)
varchar(10)--10个字节
varchar2(10)==varchar(10)==varchar2(10 byte)
varchar2(10 char)--10个字符

date(mysql有3个日期类型：date,datetime,time)

clob(大字符串)
blob(大二进制)
图片、文档、流对象、java对象.....
```

测试数据

```
/*
  多行注释


  ddf

  dfdf
*/

--单行注释




drop table t_user;
create table t_user(id int,name varchar2(10 byte));
desc t_user;
insert into t_user values(1,'aaaaaaaaa中');--插入失败



drop table t_user;
create table t_user(id int,name varchar2(10 char));
insert into t_user values(1,'aaaaaaaaa中1');
select * from t_user;


drop table t_user;
drop table "user";
create table "user"("uid" int,name varchar2(10 char),birth date);
create table t_user(id int,name varchar2(10 char),birth date);
--在oracle中字符串用'',""只在字段或者表的命名的时候使用
--可以使用to_date(字符串)函数，将一个字符串转换成date类型
insert into t_user values(1,'mike','2018-8-8');
insert into t_user values(1,'mike',sysdate);
--可以使用to_char(日期型数据)函数，将一个日期型数据转换成指定格式的字符串，类似java中的
SimpleDateFormat
select * from t_user;

drop table t_user;
create table t_user(id int,name varchar2(10 char),birth date,blog clob);

insert into t_user
values(1,'mike',sysdate,'dsfdfffffffffffffffffffffffsssssfffffffffffffffffffffffffffffff
');
```

五、表结构

```
desc t_user;
--重命名表
rename t_user to t_user1;
select * from t_user;

alter table t_user1 rename to t_user;

--重命名列
alter table t_user rename column id to sid;
drop table "user";
--对数据对象命名规范的要求
create table user(id int);--创建失败  user是关键字
create table "user"(id int);
create table "user"(uid int);--创建失败,uid是关键字
create table "user"("uid" int);

select "uid" from "user";

--表结构的复制
select * from scott.emp;

drop table emp;

select * from emp;

delete from emp;
--只复制表结构
create table emp as select * from scott.emp where 1=2;
--不仅复制表结构，同时复制其中的数据
create table emp as select * from scott.emp;

create table emp as select * from scott.emp where deptno=20;

--表名和列名不能以_开头
create table _emp1 (id int);--报错
create table emp1 (_id int);--报错
create table emp1 (id int);--正常

--行和列的复制
drop table emp;
create table emp as select job from scott.emp where deptno=20;
--这种复制的前提是该表不存在
select * from emp;
--在现有表的基础上追加复制
--前提是emp表中对应的字段必须存在
insert into emp(sal) select sal from scott.emp where deptno=20;--报错，sal字段并不存在

--给表和列加注释
comment on table t_user is '这是一张测试用的表';
comment on column t_user.sid is '这是学号，不是身份证号';
comment on column t_user.birth is '这是生日字段';

--查看表和列上的注释
```

```
select * from user_tab_comments;
select * from user_col_comments;


--对表结构的修改
--增加列
desc t_user;
alter table t_user add address varchar(20);
alter table t_user add (phone char(11),email varchar(20));
--修改列
alter table t_user modify address varchar(30);
alter table t_user modify(phone char(15),email varchar(80));

--删除列
alter table t_user drop column address;--需要关键字column
alter table t_user drop (phone,email);--不需要关键字column

--查询时区分大小写
select * from user_tables where table_name='T_USER';--查询条件大小写区分
--查询表名中出现_的表的记录
--oracle中查询条件中的_是特殊字符，代表的是任意一个字符，需要转义
select * from user_tables where table_name like '%\_%' escape '\';


select * from t_user;
--oracle不支持这种简写的多行插入，mysql是可以的
insert into t_user(sid,name)values(2,'rose')(3,'jack');
```

六：函数

1、字符串相关

```
drop table t_user;
create table t_user(name varchar(30));
insert into t_user values('mike');
insert into t_user values('p&g');
--&g：此时g作为一个变量，要求对话框输入相应的值，进行替换
--oracle中字符串的连接符是||
--dual是一个临时内存表
select 'a'||'b' from dual;
insert into t_user values('p'||'&'||'g');
select * from t_user;

--ascii--->将字符串转成ascii码
select ascii('A') from dual;

--chr--->将ascii码转成字符
select chr(65) from dual;

--vsize---->求一个字符串的字节，一个汉字2个字节
select vsize('hello中') from dual;

--lower:转小写
--upper:转大写
--initcap:单词首字母大写
select lower('HELLO') a,upper('Hello')b,initcap('HELLO WORLD')C from dual;

--字符串拼接||
select 'a'||'b'||'c'||'mike' from dual;

--字符串拼接concat
--只能适用于两个参数的拼接
select concat('a','b') from dual;

--substr-->截取字符串
--0,1都是从第一个位置开始
select substr('helloworld',0,2) from dual;
select substr('helloworld',1,2) from dual;
select substr('helloworld',2,2) from dual;
select substr('helloworld',-7,2) from dual;--lo--从后往前数7位，再向后取2位


--instr-->返回字符首次出现的位置(从位置1开始查找,找不到返回0)
select instr('hello world','el') from dual;--2
select instr('hello world','ele') from dual;--0
select instr('hello woreld','el',5) from dual;--从第5个位置开始查找

--trim--->两边去空格
--ltrim-->左边去空格
--rtrim-->右边去空格
select '['||'   abc   '||']' from dual;
select '['||trim('   abc   ')||']' from dual;
select '['||ltrim('   abc   ')||']' from dual;
select '['||rtrim('   abc   ')||']' from dual;

```

```
--rpad--->右边补空格,默认补空格
select '['||rpad('hello',10)||']' from dual;
--rpad--->右边补空格,补*
select '['||rpad('hello',10,'*')||']' from dual;

----lpad--->左边补空格,默认补空格
select '['||lpad('hello',10)||']' from dual;
--lpad--->左边补空格,补*
select '['||lpad('hello',10,'*')||']' from dual;

--replace-->替换,可以替换所有匹配的字符
select replace('hello','l','H') from dual;

drop table emp;
create table emp as select * from scott.emp;
select * from emp;

select * from emp where ename=upper('smith');

--正则表达式
--匹配ename=smith,不区分大小写
select * from emp where regexp_like(ename,'smith','i');
--匹配ename为5个字符的记录
select * from emp where regexp_like(ename,'^\w{5}$');
--匹配sal为3位数字的记录
select * from emp where regexp_like(sal,'^\d{3}$');

--将薪水的前两位遮盖
select regexp_replace(sal,'^\d{2}','**')from emp;
```

2、数值相关

```
--mod  取余
select mod(10,3) from dual;

--abs  取绝对值
select abs(-1) from dual;

--power:取指数方
select power(3,4)from dual;

--sqrt:取平方根
select sqrt(10) from dual;

--ceil:向上取整，取大于它的最小整数
select ceil(3.1) from dual;

--floor:向下取整，取小于它的最大整数
select floor(3.9)from dual;

--round:四舍五入
select round(3.5) from dual;--默认取整数
select round(100.125,2) from dual;--精度2位
select round(2150.125,-3) from dual;--精度为负数，小数点处反向查找,截取小数位

--trunc:截取小数位
select trunc(100.123),trunc(100.567) from dual;--类似于floor
select trunc(100.123,2)from dual;--将小数位保留2位后截取
select trunc(150.123,-2)from dual;--小数位反向查找，截取，小数位前补0
```

3、日期相关

```
--to_char:将日期型数据转换成字符串
select * from emp;
select hiredate from emp;
select to_char(hiredate,'yyyy-mm-dd hh:mi:ss')from emp;
select to_char(sysdate,'yyyy-mm-dd hh:mi:ss')from dual;--12时
select to_char(sysdate,'yyyy-mm-dd HH:mi:ss')from dual;--12时
select to_char(sysdate,'yyyy-mm-dd hh24:mi:ss')from dual;--24时
select to_char(sysdate,'yyyy年mm月dd日 hh24时mi分ss秒')from dual;--无法识别，oracle只识别-/:分隔的时间，需要中文分隔时，使用""包含
select to_char(sysdate,'yyyy"年"mm"月"dd"日" hh24"时"mi"分"ss"秒"')from dual;

select to_char(sysdate,'yyyy-mm-dd HH:mi:ss am')from dual;--上午/下午12时
--星期几
select to_char(sysdate,'day') from dual;

--星期的序号：1--星期天，2--星期一,...
select to_char(sysdate,'d') from dual;

--to_date--->将一个字符串转换成日期型数据
select to_date('2018-8-8 9:9','yyyy-mm-dd hh24:mi') from dual;
--课堂练习:--计算今年圣诞节是星期几?
--2018-12-25  --->星期二
select to_char(to_date('2018-12-25','yyyy-mm-dd'),'day') from dual;
```

4、系统函数

```
--sysdate:年月日时分秒
select to_char(sysdate,'yyyy-mm-dd hh24:mi:ss') from dual;
--systimestamp:年月日时分秒毫秒
select to_char(systimestamp,'yyyy-mm-dd hh24:mi:ss:ff3') from dual;

--几个关键字：user,uid
--uid:当前账号的id
--user:当前账号名称
select uid,user from dual;
```

七：作业

八：自增序列组件

```
--需要创建sequence序列组件的权限，在resource角色中有
SQL> select * from dba_sys_privs where grantee= upper('resource');

GRANTEE                        PRIVILEGE                                ADM
------------------------------ ---------------------------------------- ---
RESOURCE                       CREATE TRIGGER                           NO
RESOURCE                       CREATE SEQUENCE                          NO
RESOURCE                       CREATE TYPE                              NO
RESOURCE                       CREATE PROCEDURE                         NO
RESOURCE                       CREATE CLUSTER                           NO
RESOURCE                       CREATE OPERATOR                          NO
RESOURCE                       CREATE INDEXTYPE                         NO
RESOURCE                       CREATE TABLE                             NO
```

```
create sequence seq1;
--获取该序列的下一个自增值
select seq1.nextval from dual;

--删除序列组件
drop sequence seq1;

--获取当前值
select seq1.currval from dual;

--可以控制初始值和自增量
create sequence seq2 start with 10 increment by 3;--初始值为10，自增量为3
select seq2.nextval from dual;
select * from t_student;
insert into t_student (sid,name)values(seq2.nextval,'mike');

```

完善java代码，

- sid字段使用序列主键自己维护
- 传入字符串，通过to_date函数转成日期型数据，插入到数据库

```java
@Test
    public void writeToDB() throws Exception {

        // System.out.println(11);
        Class.forName("oracle.jdbc.driver.OracleDriver");
        Connection conn = DriverManager.getConnection(
                "jdbc:oracle:thin:@127.0.0.1:1521:orcl", "mike_", "mike_");
        // Connection conn=DriverManager.getConnection
        // ("jdbc:oracle:thin:mike1/abc@127.0.0.1:1521:orcl");
        // System.out.println(conn);
        PreparedStatement pstmt = conn
                .prepareStatement("insert into t_student
values(seq1.nextval,?,to_date(?,'yyyy-mm-dd hh24:mi:ss'),?,?,?)");
        //pstmt.setInt(1, 14);
        pstmt.setString(1, "mike");
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
        String date = sdf.format(new Date());
//      pstmt.setTimestamp(2, Timestamp.valueOf(date));
        pstmt.setString(2,date);
        String[] hobbys = { "读书", "打球" };
        // 对象字节流的处理
        pstmt.setBytes(3, convertObject2Byte(hobbys));

        // 图片字节流的处理
        pstmt.setBytes(4, readImage("d:/Desert.jpg"));

        // 大文本数据的处理
        // 使用FileReader流存储clob数据
        int len = new FileInputStream("d:/JDBCTest.java").available();
        pstmt.setCharacterStream(5, new FileReader("d:/JDBCTest.java"), len);

        int result = pstmt.executeUpdate();
        System.out.println(result);
        // 关闭资源
        pstmt.close();
        conn.close();
    }
```

```sql
--通过数据字典表可以查看自己创建的序列组件
select * from user_sequences;
```

```sql
select * from emp;

--伪列--在表中不存在，只在查询中动态生成
--rowid:用来标志当前行在数据库中的唯一引用
--它是一个字符串序列，根据用户名称，表空间名，表名等等通过某种算法自动生成
select rowid,empno,ename,sal from emp;

select * from emp where empno=1;

select * from emp where rowid='AAAM7uAAEAAAAo8AAA';
```

> 获取自增序列的值

```
//插入一条记录后获取自增序列的值（主键值）
@Test
public void testSequence() throws Exception{
  Class.forName("oracle.jdbc.driver.OracleDriver");
  Connection conn = DriverManager.getConnection(
    "jdbc:oracle:thin:@127.0.0.1:1521:orcl", "mike_", "mike_");
  PreparedStatement pstmt = conn.prepareStatement
    ("insert into t_student(sid,name)values(seq1.nextval,?)",
     Statement.RETURN_GENERATED_KEYS);
  pstmt.setString(1,"张三2");
  int cnt = pstmt.executeUpdate();//默认的返回值是影响的行数
  //System.out.println(cnt);
  //获取自增值
  ResultSet keys = pstmt.getGeneratedKeys();
  if(keys.next()){
    ROWID sid=(ROWID)keys.getObject(1);
    System.out.println(sid.stringValue());

    pstmt=conn.prepareStatement
      ("select sid from t_student where rowid=?");
    pstmt.setString(1,sid.stringValue());
    ResultSet rs = pstmt.executeQuery();
    if(rs.next()){
      System.out.println(rs.getInt(1));
      rs.close();
    }

  }
  pstmt.close();
  conn.close();
}
```

九：判断函数

在oracle中有三个判断函数，用于对null值或者给定值进行判断

- nvl,nvl2：对null值进行判断

```
select * from emp;
--nvl(列值，给定值)：如果列值为null,显示给定值，否则显示该列值
select sal,nvl(sal,0) from emp;
--nvl2(列值，给定值1，给定值2)：如果列值为null,显示给定值2，否则显示给定值1
select sal,nvl(sal,0),nvl2(sal,sal,0) from emp;
--nvl2(列值，列值，给定值)==nvl(列值，给定值)
--例子：如果没有奖金(null)，发奖金100，否则奖金多发10%
select sal,comm,nvl2(comm,comm*1.1,100)from emp;
```

注意：**nvl,nvl2值只是对null值进行判断，如果需要更强大的判断，需要使用decode进行判断**

- decode：对给定值进行判断

```
--decode(列值，值1，给定值1，值2，给定值2，值3，给定值3。。。,默认值)
create table dept as select * from scott.dept;
select * from dept;
select empno,ename,deptno,decode(deptno,10,'研发部',20,'销售部',30,'行政部','其它')from emp;

select * from emp;

insert into emp(empno,ename,sal,deptno)values(1,'mike',1000,40);

--将10号部门员工的薪水+10%，20号部门员工的薪水+15%，10号部门员工的薪水+20%，其余部门员工的薪水不加
update emp set sal=decode(deptno,10,sal*1.1,20,sal*1.15,30,sal*1.2,sal);
```

十：分支语句

1、case end

语法1：

- 等值比较===decode函数

case 列值

when 等值1 then 结果1

when 等值2 then 结果2

when 等值3 then 结果3

else 结果4

end;

```
select empno,ename,deptno,
case deptno
    when 10  then '研发部'
    when 20  then '销售部'
    when 30  then '行政部'
    else '其它'
end "部门"
from emp;


update emp set sal=
case deptno
    when 10 then sal*1.1
    when 20 then sal*1.15
    when 30 then sal*1.2
    else sal

end;
```

- 算法比较

  语法2：

case

when 列值算法1 then 结果1

when 列值算法2 then 结果2

when 列值算法3 then 结果3

else 结果4

end

```
--根据薪水的范围加薪
--0-999--->加100
--1000-1999---->加150
--2000-2999---->加200
-->其它情况不加
select * from emp;

update emp set sal=
case
    when sal between 0 and 999 then sal+100
    when sal between 1000 and 1999 then sal+150
    when sal between 2000 and 2999 then sal+200
    else sal
end;
```

十一：事务

所有的数据操作语言(DML)都必须要显示提交，数据才能真正保存到数据库中，否则这个数据仅仅保存在SGA（全局缓冲区）容器中，没有提交的数据，其它会话是无法看到的。

```
select * from emp;
insert into emp(empno,ename)values(5,'jack1');

commit;
rollback;
```

只有你提交或者回滚，之后的数据才能被其它会话看到

因为oracle默认采用的事务隔离级别是提交读（只有你提交了，别人才能看见）

mysql默认采用的事务隔离级别是未提交读，只要你操作了，其它会话就能读到

oracle只支持2种事务隔离级别，1：提交读，2：串行读(所有的更新都必须先后执行，不能并发)

- oracle的4种语言

  1.DML :data manipulation language 数据操语言

  ```
  select update  delete  insert
  ```

  2.DDL：data definition language 数据定义语言

```
create drop  alter  truncate
```

3.DCL：data controll language 数据控制语言

```
grant  revoke
```

4.DTL：data transaction language 数据事务语言

```
commit rollback  savepoint
```

注意：只有dml语言支持事务，其它都不支持事务

- 在以下三种情况下，事务能够自动提交
  - 会话超时
  - disc手动断开连接

```
在命令行输入
insert into emp(empno,ename)values(6,'taylor');
disc:--断开连接

--在其它会话能看见该记录
```

  - 当你执行了一条ddl或者dcl语句后，之前未被提交的数据会自动提交

```
在命令行输入
insert into emp(empno,ename)values(6,'taylor');
create table t1(id int);--执行一条ddl或者dcl语句

--在其它会话能看见该记录
```

- 事务保存点(savepoint)

```

select * from emp;
update emp set ename=lower(ename) where deptno=10;
savepoint sp1;
update emp set ename=initcap(ename) where deptno=20;
savepoint sp2;
rollback;--回退到上次事务提交时的状态
--回退到保存点sp1时的状态(但未提交)
rollback to sp1;
commit;
```

- 在java代码中实现oracle的事务

```java
@Test
public void testTransaction() throws Exception{
  Class.forName("oracle.jdbc.driver.OracleDriver");
  Connection conn = DriverManager.getConnection(
    "jdbc:oracle:thin:@127.0.0.1:1521:orcl", "mike_", "mike_");
  conn.setAutoCommit(false);
  try {
    Statement stmt = conn.createStatement();
    stmt.executeUpdate("insert into emp(empno,ename)values(10,'Swarn1')");
    //           stmt.executeUpdate("insert into
emp(empno,ename)values(9,'Swarnaaaaaaaaaaa')");
    //conn.commit();
  } catch (Exception e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
    //conn.rollback();

  }

  conn.close();//只要连接关闭，就会帮你提交，与你是否设置自动提交无关
}
```

```java
@Test
public void testSavePoint() throws Exception{
  Class.forName("oracle.jdbc.driver.OracleDriver");
  Connection conn = DriverManager.getConnection(
    "jdbc:oracle:thin:@127.0.0.1:1521:orcl", "mike_", "mike_");
  conn.setAutoCommit(false);
  Savepoint sp1=null;
  try {
    Statement stmt = conn.createStatement();
    stmt.executeUpdate("insert into emp(empno,ename)values(11,'Swarn2')");
    sp1=conn.setSavepoint();
    stmt.executeUpdate("insert into emp(empno,ename)values(12,'Swarn3aaaaaaaaaaa')");
    conn.commit();
  } catch (Exception e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
    conn.rollback(sp1);
    conn.commit();

  }
  //conn.close();//只要连接关闭，就会帮你提交，与你是否设置自动提交无关
}
```

十二：查询

多表查询

- where
- join

- 子查询

1、where查询

where N表连接(连接条件至少N-1)

- 连接关系：等值连接

```
select * from dept;
select * from emp;
--查询部门所对应的部门名称
select e.ename,e.deptno,d.dname
from emp e,dept d
where e.deptno=d.deptno;

delete from emp where empno<20;

insert into dept values(50,'research','seatle');

--左外连接(where条件的左侧表的数据都出现，右侧表没有对应的数据，显示null)
select ename,dname
from emp,dept
where emp.deptno=dept.deptno(+);
--右外连接(where条件的右侧表的数据都出现，左侧表没有对应的数据，显示null)
select ename,dname
from emp,dept
where emp.deptno(+)=dept.deptno;

--自连接(同一个表，不同列之间)
--应用场景：组织关系，上下级关系，商品目录等等

--找jones的下属
select e.ename,m.ename
from emp e,emp m
where e.mgr=m.empno
and m.ename='JONES';

--查找资历比上级还老的雇员信息
select e.ename,e.hiredate,m.ename,m.hiredate
from emp e,emp m
where e.mgr=m.empno
and e.hiredate<m.hiredate;
```

- 连接关系：非等值连接

```
--查找雇员的收入以及对应的等级
create table salgrade as select * from scott.salgrade;
select * from salgrade;

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

- 三张表(两个关系，一个等值关系，一个非等值关系)

```
--查询雇员的姓名和所在的部门名称以及收入所处的等级
select e.ename,d.dname,e.sal,s.grade
from emp e,dept d,salgrade s
where e.deptno=d.deptno
and e.sal between s.LOSAL and s.HISAL;
```

2、join查询

所有的where查询都可以使用join查询，join查询还可以有自己的查询方式

- 连接关系：等值连接

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

```
insert into emp(empno,ename,deptno)values(2,'mike1',70);
select * from dept;
select ename,dname
from emp
left join dept
on emp.deptno=dept.deptno;

select ename,dname
from emp
right join dept
on emp.deptno=dept.deptno;


--找jones的下属
select e.ename,m.ename
from emp e
join emp m
on e.mgr=m.empno
and m.ename='JONES';


--查找资历比上级还老的雇员信息
select e.ename,e.hiredate,m.ename,m.hiredate
from emp e
join emp m
on e.mgr=m.empno
and e.hiredate<m.hiredate;

```

- 连接关系：非等值连接

```sql
--查找雇员的收入以及对应的等级
select e.ename,e.sal,s.grade
from emp e
join salgrade s
on e.sal between s.losal and s.HISAL;
```

- 三张表(两个关系，一个等值关系，一个非等值关系)

```sql
--查询雇员的姓名和所在的部门名称以及收入所处的等级
--join查询的查询条件可以任意组合(3!(6种写法))
--方式1
select e.ename,d.dname,e.sal,s.grade
from emp e
join dept d
on e.deptno=d.deptno
join salgrade s
on e.sal between s.LOSAL and s.HISAL;

--方式2
select e.ename,d.dname,e.sal,s.grade
from emp e
join salgrade s
on e.sal between s.LOSAL and s.HISAL
join dept d
on e.deptno=d.deptno;

--方式3
select e.ename,d.dname,e.sal,s.grade
from salgrade s
join emp e
on e.sal between s.LOSAL and s.HISAL
join dept d
on e.deptno=d.deptno;
```

十三：三种数据关系模型

- 一对一
- 一对多
- 多对多

1、一对一

```sql
--一对一
--有外键，外键有唯一性约束
--创建身份证表
create table t_idcard(idcard char(5) primary key ,
name varchar(20));
select * from t_idcard;
insert into t_idcard values('a0001','mike');
insert into t_idcard values('a0002','rose');

--创建护照表
create table t_password(passid int primary key,
pass_idcard char(5) references t_idcard(idcard)unique
);
insert into t_password values(10001,'a0001');
insert into t_password values(10002,'a0002');

select * from t_password;
```

2、一对多

```sql

--一对多
--与一对一类似，区别在于外键上没有唯一性约束
```

3、多对多

```sql
--多对多
--创建学生表
create table t_stu(sid int primary key,sname varchar(20));

insert into t_stu values(1,'mike');
insert into t_stu values(2,'张三');
insert into t_stu values(3,'李四');
select * from t_stu;

--创建课程表
select * from t_course;
create table t_course(cid int primary key,cname varchar(20));
insert into t_course values(10,'java');
insert into t_course values(20,'mysql');
insert into t_course values(30,'jsp');

--将外键单独抽离，形成一张中间表
create table t_rel_stu_cur(
rsid  int,--关联学生表
rcid  int--关联课程表
);

--给这两个外键创建约束条件
alter table t_rel_stu_cur add constraint fk_rsid foreign key(rsid)references t_stu(sid);
alter table t_rel_stu_cur add constraint fk_csid foreign key(rcid)references t_course(cid);
--给中间表加数据
insert into t_rel_stu_cur values(1,10);
insert into t_rel_stu_cur values(1,20);
insert into t_rel_stu_cur values(1,30);
insert into t_rel_stu_cur values(2,10);
insert into t_rel_stu_cur values(3,10);
insert into t_rel_stu_cur values(3,20);
select * from t_rel_stu_cur;

--查询出学生名称，以及学习的课程名称
--where查询
select s.sname,c.cname
from t_stu s,t_course c,t_rel_stu_cur r
where r.rsid=s.sid and r.RCID=c.CID;

--join查询
select s.sname,c.cname
from t_rel_stu_cur r
join t_course c
on  r.rcid=c.cid
join t_stu s
on r.RsID=s.sid;
```

十四：子查询

1. 将一个查询的结果(单行单列),作为一个值再嵌入到其它的查询语句的where 条件中

```sql
--查询比jones薪水高的雇员的信息
select * from emp;
select sal from emp where ename=upper('jones');

select * from emp where sal >(select sal from emp where ename=upper('jones'));

--给新入职的雇员的薪水和jones的薪水一致
update emp set sal=(select sal from emp where ename=upper('jones')) where empno=1;
```

2.将一个查询的结果(多行多列)作为一个逻辑表和其它表进行关联查询

```sql
--查询每个部门下拿最低工资雇员的信息
--分组查询：group by
--集合函数：min max sum count
--分组查询的特点：只要查询中出现聚合函数，就必须进行分组(group by)
--group by 的条件至少包含查询中的非聚合字段
--查找每个部门下的最低工资
select deptno,min(sal)
from emp
group by deptno
order by deptno;

--将查询到的部门编号和最低工资作为条件对emp表进行查询
select * from emp where deptno=10 and sal=1723;

select * from emp e,(select deptno,min(sal)min_sal
from emp
group by deptno
order by deptno)t
where e.deptno=t.deptno
and e.sal=t.min_sal;

--进一步简化
--如果查询出的记录是多行多列，使用in
select * from emp where (deptno,sal)in(select deptno,min(sal)
from emp
group by deptno);
```

3.对于多行结果，还可以使用> < all any组合

```
>any:比结果中的最大值小
<any:比结果中的最小值大
>all:比结果中的最大值大
<all:比结果中的最小值小
```

```sql
--查询哪些雇员的薪水是比整个20号部门的雇员的薪水还要高的雇员的信息
select max(sal) from emp where deptno=20;

select * from emp where sal>(select max(sal) from emp where deptno=20);
--另一种写法

select * from emp where sal >all(select sal from emp where deptno=20);
```

```
--练习
--创建销售员表
create table t_seller(
    sid int primary key,--编号
    sname varchar(20),--姓名
    sal number(8,2)--收入
);

insert into t_seller values(1,'mike',5500);
insert into t_seller values(2,'rose',6500);
insert into t_seller values(3,'张三',7500);
select * from t_seller;

--创建销售业绩表
create table t_sale(
    id int primary key,--编号
    amount float,  --销售额
    sid int references t_seller(sid)
);
insert into t_sale values(1,4500,1);
insert into t_sale values(2,5500,2);
insert into t_sale values(3,5500,2);
insert into t_sale values(4,4500,3);
select * from t_sale;

--如果这个月的销售总额>19000元，月底每人加薪10%,否则不加
select sum(amount) from t_sale;
update t_seller set sal=sal*1.1 where (select sum(amount) from t_sale)>20000;

select * from t_seller;
```

```
--练习：查询哪些部门的雇员最低薪水和最高薪水的差额超过2000元的部门的信息
select * from dept where deptno in(
select deptno from(
select deptno,min(sal)min_sal,max(sal)max_sal from emp
group by deptno)
where max_sal-min_sal>2000
);
```

4.exists

select/update/delete where exists(子查询)

如果子查询有返回结果，那么就执行exists左侧的语句

```
--如果这个月的销售总额>19000元，月底每人加薪10%,否则不加
update t_seller set sal=sal*1.1
where exists (select sum(amount) from t_sale having sum(amount)>=20000);

select * from t_seller;
```

十五：分组聚合

min max avg count sum

clob blob不支持以上聚合函数

date不支持avg sum

对于null值也不支持集合统计

```
--求奖金的平均值,只有4个雇员有值，所以平均值为这4个记录的平均值,null值不参与计算
select avg(comm) from emp;
--统计所有员工奖金的平均值
select count(*)from emp;

select sum(comm) from emp;

select sum(comm)/count(*) from emp;

--另外一种写法
select avg(nvl(comm,0)) from emp;
```

> 按照部门进行分组，然后再进行统计

```sql
select deptno,min(sal),max(sal)from emp group by deptno;

drop table emp;
create table emp as select * from scott.emp;
select * from emp;
delete from emp where empno=1;
--统计每一个职位的雇员中的最低薪水和最高薪水
select job ,min(sal),max(sal) from emp group by job;

--统计每个部门中最低薪水和最高薪水差额超过2000的记录
--语法是错误的，因为where是在group by 之前进行的筛选
select deptno,min(sal)min_sal,max(sal)max_sal from emp group by deptno
where max_sal-min_sal>2000;

--使用having,注意having子句必须使用聚合函数,having是在group by语句后执行的筛选
select deptno,min(sal)min_sal,max(sal)max_sal from emp group by deptno
having max(sal)-min(sal)>2000;

select * from emp;
--进一步进行筛选，当薪水低于1000元时，是临时工，不参与统计
select deptno,min(sal)min_sal,max(sal)max_sal from emp group by deptno
having max(sal)-min(sal)>2000
and  min(sal)>1000;

select deptno,min(sal)min_sal,max(sal)max_sal from emp
where sal>1000
group by deptno
having max(sal)-min(sal)>2000;

--总结:
--1:数据库统计的顺序：先通过where进行筛选，然后通过group by进行站队，然后通过统计函数进行统计
--最后通过having对站队统计后的数据再次进行筛选
--2:where的条件可以是任何列，having的条件只能是聚合函数
```

> 对重复记录的统计

```sql
create table t_stu1(name varchar(20));

insert into t_stu1 values('mike');
insert into t_stu1 values('mike1');
insert into t_stu1 values('mike1');
insert into t_stu1 values('mike2');
insert into t_stu1 values('mike');
insert into t_stu1 values('mike3');
insert into t_stu1 values('mike');
insert into t_stu1 values('mike4');

select * from t_stu1;

--显示姓名(重复的姓名只出现一次)
select distinct name from t_stu1;
select name from t_stu1 group by name;--告诉我们一个重要的道理，如果想得到一个没有重复的行，只要对所有列进行分组即可

--显示有重复的姓名（哪些姓名出现多次）
select name,count(name)from t_stu1 group by name
having count(name)>1;

--显示没有重复的姓名(哪些姓名只出现一次)
select name,count(name)from t_stu1 group by name
having count(name)=1;


create table t_stu2(
    id int primary key,
    name varchar(20),
    addr varchar(30)
);
insert into t_stu2 values(1,'mike','Boston');
insert into t_stu2 values(2,'mike1','Boston');
insert into t_stu2 values(3,'mike2','Boston');
insert into t_stu2 values(4,'mike','Boston');
insert into t_stu2 values(5,'mike3','Boston');
insert into t_stu2 values(6,'mike3','Boston');
insert into t_stu2 values(7,'mike1','Boston');
insert into t_stu2 values(8,'mike','Boston');

select * from t_stu2;
--查询姓名有重复的姓名及出现的次数
select name,count(name) from t_stu2 group by name having count(name)>1;

--查询出整行都重复的记录
select name,addr,count(*) from t_stu2 group by name,addr having count(*)>1;

--删除重复的行，重复的行只留一行
select rowid,t.* from t_stu2 t;

--重复的行只留一行

--取出的就是不重复的行的rowid
```

```
select max(rowid) from t_stu2 group by name,addr;

--删除不是这些记录的记录，也就是重复行,剩下的就是非重复行
delete from t_stu2 where rowid not in
(select max(rowid) from t_stu2 group by name,addr);

--查询在1980-1981年入职的雇员的人数
select empno,to_char(hiredate,'yyyy') from emp;

select
  count(decode(to_char(hiredate,'yyyy'),'1980',empno,null))"1980",
  count(decode(to_char(hiredate,'yyyy'),'1981',empno,null))"1981"
from emp;
```

十六：rownum

> rownum:查询出来的行的序号

rownum的特点

1. rownum在做where查询时，只支持
   - =1
   - <n
   - <=n
   - between 1 and n

```
select rownum,e.* from emp e
where rownum between 1 and  6;
```

2. 如果和order by组合，先给序号再排序(序号就乱了)

```
--乱序
select rownum,e.*
from emp e
order by sal;

--排序
select rownum,t.* from(
select e.*
from emp e
order by sal) t;

--分页
--获取第11-15条记录
select tt.* from(
select rownum rn,t.* from(
select e.*
from emp e
order by sal) t)tt
where tt.rn between 11 and 15;
```

十七：投影查询

union

> 将两个查询结果合并，去掉重复行

```
select * from emp where deptno=10
union
select * from emp where sal>3000;
```

union all

> 将两个查询结果合并，重复行保留

```
select * from emp where deptno=10
union all
select * from emp where sal>3000;
```

intersect

> 取交集

```
select * from emp where deptno=10
intersect
select * from emp where sal>3000;
```

minus

> 取差集

> A minus B--->A表中存在但B表中不存在
>
> B minus A--->B表中存在但A表中不存在

```
select * from emp where deptno=10
minus
select * from emp where sal>3000;


select * from emp where sal>3000
minus
select * from emp where deptno=10;
```

十八：作业

```sql
create table t_air(
    aid int,
    aname varchar(30)
);

insert into t_air values(1,'南方航空');
insert into t_air values(2,'东方航空');
insert into t_air values(3,'中国国航');
select * from t_air;

create table t_city(
    cid int,
    cname varchar(30)
);
insert into t_city values(1,'北京');
insert into t_city values(2,'南京');
insert into t_city values(3,'成都');
insert into t_city values(4,'上海');
select * from t_city;

create table t_shedule(
    sid int,
    aid int,
    f_id int,
    t_id int
);
insert into t_shedule values(1,1,2,3);
insert into t_shedule values(2,1,2,4);
insert into t_shedule values(3,2,1,3);
insert into t_shedule values(4,3,2,1);

select * from t_shedule;

select s.sid,a.aname,c.cname,c1.cname
from t_shedule s
join t_air a
on s.AID=a.AID
join t_city c
on s.F_ID=c.CID
join t_city c1
on s.T_ID=c1.CID
order by s.sid;
```

```sql
--1.查询每个部门的名称，雇员数，平均工资，最低工资和拿最低工资的雇员名称

--查询每个部门的编号，雇员数，平均工资，最低工资和拿最低工资
select deptno,count(empno)cnt_empno,round(avg(nvl(sal,0)))avg_sal,
round(min(nvl(sal,0)))min_sal
from emp
group by deptno
order by deptno;


--查询每个部门的名称，雇员数，平均工资，最低工资和拿最低工资的雇员名称
select t.deptno,t.cnt_empno,t.avg_sal,t.min_sal,e.ename,d.dname
from(
select deptno,count(empno)cnt_empno,round(avg(nvl(sal,0)))avg_sal,
round(min(nvl(sal,0)))min_sal
from emp
group by deptno)t,
dept d,emp e
where
e.deptno=d.deptno
and e.sal=t.min_sal
order by deptno;
```

```sql
desc TBLEXAMINATION;
desc TBLANSWER;

select * from TBLEXAMINATION;

select * from tblanswer;

--先找到对应的结果
select e.id,e.ans,a.ans,e.sid
from tblexamination e,tblanswer a
where e.id=a.id;

--进一步，进行比较
--判断e.ans和a.ans的值是否相同，相同-->1  不同-->0
--别名默认大写，加了""后，别名就是“”内的值
select e.id,
case
    when e.ans=a.ans then 1
    else 0
end  "score"


from tblexamination e,tblanswer a
where e.id=a.id;

--进一步完善，进行统计
select e.sid,
sum(case
    when e.ans=a.ans then 1
    else 0
end) "score"


from tblexamination e,tblanswer a
where e.id=a.id
group by e.sid;


select e.sid,
sum(case
    when e.ans=a.ans then 1
    else 0
end)/(select count(*) from TBLANSWER)*100||'%' "score"


from tblexamination e,tblanswer a
where e.id=a.id
group by e.sid;
```

```
select * from t_student3;

select id,name,
sum(decode(course,'java',score))java,
sum(decode(course,'sql',score))sql,
sum(decode(course,'c++',score))"c++"
from t_student3
group by id,name
order by id;
```