Oracle10g 学习笔记

一、Oracle 简介

甲骨文，Oracle 公司依靠 IBM 公司

1970 年 6 月，IBM 公司研究员埃德加.考特在 Communications of ACM 上发表著名《大型共享数据库数据的关系模型》的论文。

1977 年 6 月，Larry Ellison 与 Bob Miner 和 Ed Oates 在硅谷共同创办了一家名为软件开发实验室（Software Development Laboratories,SDL）的计算机公司（Oracle 公司的前身），SDL 开始策划用构建可商用的关系型数据库管理系统（RDBMS）。

Bruce Scott ：离开公司，自己开发一套数据库 PointBase。

主要版本：

·Oracle 8

·Oracle 8i ： i ， internet，表示此时 Oracle 公司正式开始进军互联网。

·Oracle 9i ：与 8i 产品非常相似

·Oracle 10g ： g,表网格技术

二、Oracle 的安装

安装盘镜像文件：

[Oracle.10g.For.Windows].Oracle.10g.for.Windows.ISO

运行 ： oracle 10g \ install \ setup.exe 进行安装

1、选择企业版

2 、 全 局 数 据 库 名 称 ： ACCP （ 为 服 务 选 项 ， OracleServiceACCP）

3、选中创建带样本方案的数据库

4、设置同一口令 oracleadmin/oracleadmin

5、打开口令管理

解锁普通用户 scott ，设置密码 tiger

普通管理员 system/manager

超级管理员 sys/change_on_install

三、sqlplusw 的使用

企业管理器 em ： http://localhost:5500/em

sqlplusw scott/tiger 以用户 scott 身份进行登录

sqlplusw /nolog 无连接登录

要运行 Oracle，必须打开两个服务

1、**监听 Listener** ：

net start oracleoradb10g_home1tnslistener

2、**服务 Service** ：net start oracleserviceaccp

sqlplus ：oracle 提供的一个命令行执行工具软件，一般不常用

sqlplusw ：oracle 提供的窗口形式的命令行工具

主机字符串：如果一台机器上有多个数据库，则要输入数据库的名称

sqlplusw 的使用：

1、登录 sqlplusw scott/tiger

2、设置环境变量

```
SQL>set linesize 300  设置行大小
SQL>set pagesize 30 设置每页显示的行数
```

3、编辑 ed

由于 sqlplusw 中无法修改，因此借助于本机记事本进行编辑，输入 "ed 文件名"

```
SQL>ed a ;
```

输入之后，如果此文件不存在，则提示创建此文件 a.sql

编辑完之后，可以通过 ed a 再次凋出修改内容

```
SQL>@a ;
```

执行文件 a.sql 中所有的内容，当然，可以执行任何文件

```
SQL>ed d:\demo.txt
SQL>@d:\demo.txt
```

可以省略路径中的"\"，如果文件后缀名为 sql 则可以省略不写

4、查看当前用户

```
SQL>show user
```

5、更改用户

```
SQL>conn system/manager
SQL>conn sys/change_on_install as sysdba
```

6、显示当前用户下的所有表

```
SQL>select * from tab ;
```

7、查看表结构

```
SQL>desc emp ;
```

8、继续使用上次正确的指令

```
SQL>/
```

9、查看当前所有的环境变量

```
SQL>show all ;
```

10、显示所有错误

```
SQL>show errors ;
```

11、执行 SQL 脚本

```
SQL>start fileName
SQL>@fileName
```

12、执行 window 命令

```
SQL>$calc ;
```

13、将显示的内容输出到文件

```
SQL>spool d:\out.txt
SQL>...
SQL>spool off
```

将省略号部分的显示内容存储到文件 out.txt 中，直到 spool off 关闭输出后，文件中才会一次性保存所有内容

14、利用左键和右键组合，将所选内容快速复制到最后一行

左键选中一部分内容，不放，再按下右键

15、清屏

```
SQL>clear scr ;
```

16、&与&& 接收用户输入

```
SQL>select '&name' from dual;
SQL>select '&&name' from dual;
```

四、简单 SQL 语句

```
/*
  先来熟悉scott用户下的四张表
  1 雇员表 emp
    empno number(4) 表示雇员编号
    ename varchar2(10) 表示雇员姓名
    job varchar2(9) 表示工作职位
    mgr number(4) 表示领导编号
    hiredate date 表示雇佣日期
    sal number(7,2) 表示月薪工资
    comm number(7,2) 表示奖金,佣金
    deptno number(2) 部门编号
  2 部门表dept
    deptno number(2) 部门编号
    dname varchar2(14) 部门名称
    loc varchar2(13) 部门位置
  3 工资等级表 salgrade
    grade number 等级名称
    losal number 此等级最低工资
    hisal number 此等级最高工资
  4 奖金表 bonus
    ename varchar2(10) 雇员姓名
    job varchar2(9) 雇员工作
    sal number 雇员工资
    comm number 雇员资金
*/
--1 查询当前用户下的所有表
select * from tab ;
--2 查询雇员表中所有信息
select * from emp ;
--3 查询雇员编号,姓名,工作,工资
select empno,ename,job,sal from emp ;
--4 查询雇员编号,姓名,工作,工资,并显示中文
select empno 编号,ename 姓名,job 工作,sal 工资 from emp ;
--5 消除重复列,查询雇员工作种类
select distinct job from emp ;          --可以消除多余的列
select distinct * from emp ;            --不可以,查询多列时只有同时重复才能消除
--6 字符串连接操作
--查询雇员编号,姓名,工作.按以下格工显示:编号:7369,姓名:Smith,工作:Clerk
select '编号:'||empno||',姓名:'||ename||',工作:'||job 雇员资料 from emp ;
--7 查询列支持四则运算
--查询雇员编号,姓名,工作,年薪
select empno 编号,ename 姓名,job 工作,sal 月薪,sal*12 年薪 from emp ;
--8 Where条件查询
-- 查询工资大于1500的所有雇员
select empno 编号,ename 姓名,job 工作,sal 月薪 from emp where sal>1500 ;
--查询可以得到奖金的所有雇员
select empno 编号,ename 姓名,job 工作,comm 奖金 from emp
where comm is not null ;
--查询工资大于1500或可以得到奖金的雇员
select empno 编号,ename 姓名,job 工作,sal 月薪,comm 奖金 from emp
where sal>1500 or comm is not null ;
--查询工资大于1500并且可以领取奖金的雇员
select empno 编号,ename 姓名,job 工作,sal 月薪,comm 奖金 from emp
where sal>1500 and comm is not null ;
--查询工资不大于1500或者不可以领取奖金的雇员
select empno 编号,ename 姓名,job 工作,sal 月薪,comm 奖金 from emp
where not (sal>1500 and comm is not null) ;
--查询工资在1500到3000的所有雇员信息
```

```
select empno 编号,ename 姓名,job 工作,sal 月薪,comm 奖金 from emp
where sal between 1500 and 3000 ;
--查询在1981年雇用的员工信息
select empno 编号,ename 姓名,job 工作,sal 月薪,hiredate 雇用日期
from emp
where hiredate between '01-1月-1981' and '31-12月-1981' ;
--用like必写上面
select empno 编号,ename 姓名,job 工作,sal 月薪,hiredate 雇用日期
from emp
where hiredate like '%81%' ;
--查询雇员姓名中第二个字母为"M"的雇员
select empno 编号,ename 姓名,job 工作,sal 月薪,hiredate 雇用日期
from emp
where ename like '_M%' ;
--查询雇员工资中带8这个数字的
select empno 编号,ename 姓名,job 工作,sal 月薪,hiredate 雇用日期
from emp
where sal like '%8%' ;
--查询编号是7369,7499,7521,7799的雇员信息
select empno 编号,ename 姓名,job 工作,sal 月薪,hiredate 雇用日期
from emp
where empno in (7369,7499,7521,7799) ;
--查询雇员编号不是7369,7499,7521,7799的所有雇员信息
select empno 编号,ename 姓名,job 工作,sal 月薪,hiredate 雇用日期
from emp
where empno not in (7369,7499,7521,7799) ;
--查询雇员编号为7369的雇员信息
select empno 编号,ename 姓名,job 工作,sal 月薪,hiredate 雇用日期
from emp
where empno = 7369 ;
--查询雇员编号不为7369的雇员信息
select empno 编号,ename 姓名,job 工作,sal 月薪,hiredate 雇用日期
from emp
where empno<>7369 ;
--查询雇员信息,按工资由低到高排序
select empno 编号,ename 姓名,job 工作,sal 月薪,hiredate 雇用日期
from emp
order by sal asc ;
--查询雇员信息,按工资由高到低排序
select empno 编号,ename 姓名,job 工作,sal 月薪,hiredate 雇用日期
from emp
order by sal desc ;
```

五、单行函数

```
--单行函数
/*
   字符函数
*/
--小写转大写 upper()
select upper('hello accp!') from dual ;
--大写转小写 lower()
select lower('HELLO ACCP!') from dual ;
--单词首字母大写,其余小写 initcap()
select initcap('hello accp!') from dual ;
--返回指定字符串的十进制数 ascii()
select ascii('\') ,ascii('n') n from dual ;
--返回指定整数对应的字符串 chr()
select chr(54740) zhao from dual ;
--字符串连接 concat()
select concat('0512-','88889999') from dual ;
--搜索指定字符串 instr(content,search,startindex,index)
select instr('hello accp welcome to you !','o',6,2) from dual ;
--返回字符串的长度 length()
select length('hello accp!') from dual ;
--粘贴字符 lpad(),rpad() 在左边或右边插入指定字符(根据指定返回结果长度循环插入指定字符)
select lpad(rpad('gao',10,'*'),17,'*') from dual ; --结果 *******gao*******
select rpad('hello',15,'accp') from dual ; --结果 helloaccpaccpac
--删除字符串 ltrim() rtrim() trim()
select '   accp   ' str from dual ;
select ltrim('   accp   ') str from dual ;
select rtrim('   accp   ') str from dual ;
select ltrim(rtrim('   accp   ',' '),' ') str from dual ;
select trim(' ' from '   accp   ') str from dual ;
--截取字符串 substr(str,startIndex,length)
select substr('hello accp!',1,7) str1,substr('hello accp!',0,7) str2 from dual ;
--结果一样都是hello a
--查询雇员姓名的最后三个字母
select substr(ename,-3,3) ename from emp ;
--字符串内容替换 replace()
select replace('   accp   ',' ') str from dual ;          --accp
select replace('   accp   ',' ','*') str from dual ;   --**accp**
/*
          数值函数
*/
--返回大于指定值的最小的的整数 ceil()
select ceil(68.49),ceil(-68.49) from dual ; --69,-68
--返回小于指定值的最大整数
select floor(68.49),floor(-68.49) from dual ; --68,-69
--四舍五入 round()
select   round(789.536),round(789.536,2),round(789.536,-2)   from   dual   ;        --790,789.54,800
--截断小数位 trunc()
select   trunc(789.536),trunc(789.536,2),trunc(789.536,-2)   from   dual   ;        --789,789.53,700
--取余mod()
select mod(10,3) from dual ; --1
--返回一个数值的符合 sign()
select sign(5),sign(0),sign(-5) from dual ;   --1,0,-1

/*
          日期函数
*/
--当前日期 sysdate
select sysdate from dual ;
--查询10部门雇员进入公司的星期数
select floor((sysdate-hiredate)/7) "weeks" from emp ;
--求出给定日期范围的月数 months_between()
select floor(months_between(sysdate,'27-2月 -83')) "months" from dual ;
--在指定的日期上加上指定的月数求出之后的日期
select add_months(sysdate,12) "new_date" from dual ;
--求出下一个指定星期对应的日期
select next_day(sysdate,'星期五') from dual ;
--求出指定日期的最后一天 last_day()
select last_day(sysdate) from dual ;

/*
          转换函数
*/
--转换成字符串 to_char()
/*
YYYY：四位表示的年份
YYY，YY，Y：年份的最后三位、两位或一位，缺省为当前世纪
MM：01~12的月份编号
MONTH：九个字符表示的月份，右边用空格填补
MON：三位字符的月份缩写
WW：一年中的星期
D：星期中的第几天
DY/DAY:显示星期几
DD：月份中的第几天
DDD：年所中的第几天
DAY：九个字符表示的天的全称，右边用空格补齐
HH，HH12：一天中的第几个小时，12进制表示法
HH24：一天中的第几个小时，取值为00~23
MI：一小时中的分钟
SS：一分钟中的秒
SSSS：从午夜开始过去的秒数
*/
select to_char(hiredate,'yyyy/mm/dd') "date" from emp ;      --格式化日期到字符串
select to_char(hiredate,'fmyyyy/mm/dd') "date" from emp ;   --fm去前导零
select to_char(sysdate,'fmyyyy/mm/dd hh24:mi:ss') "date" from emp ;
select to_char(sal,'L99,999') "sal" from emp ;                    --用当地货币表示法
select to_char(sal,'$99,999') "sal" from emp ;                    --指定美元货币表示法
/*
to_char函数特殊用法
to_char(sysdate,'d')  每周第几天
to_char(sysdate,'dd')  每月第几天
to_char(sysdate,'ddd')  每年第几天
to_char(sysdate,'ww')  每年第几周
to_char(sysdate,'mm')  每年第几月
to_char(sysdate,'q')  每年第几季
to_char(sysdate,'yyyy')  年
*/
select to_char(sysdate,'ww') from dual ;    --今年第38周
```

```
--将字符串转换成日期  to_date(string,'format')
select to_date('2009-9-18','yyyy-mm-dd') from dual ;
--将字符串转换成数字  to_number(string)
select to_number('123')+to_number('123') from dual ;
--1  查询部门30中的所有员工
select * from emp where deptno=30;
--2  列出所有办事员(CLERK)的姓名,编号和部门编号
select ename,empno,deptno from emp where lower(job)='clerk';
--3  找出佣金高于薪金的员工
/*
   truncate table emp ;
   select * from emp;
   insert into emp select * from scott.emp;
*/
select * from emp where comm > sal;
--求出每个雇员的年薪
/*
    nvl(arg,value)  如果前面的arg值为null，那么返回后面的value值
*/
select (sal+nvl(comm,0))*12 income ,ename from emp;
--4  找出佣金高于薪金的60%的员工
select * from emp where comm>sal*0.6;
--5  找出部门10中所有经理(MANAGER)和部门20中所有办事员(CLERK)的详细资料
select * from emp
where (deptno=10 and lower(job)='manager') or (deptno=20 and lower(job)='clerk');
--6  找出部门10中所有经理(MANAGER),部门20中的所有办事员(CLEAK),
--   既不是经理又不是办事员但薪金大于或等于2000的所有员工的详细资料
select * from emp
where (deptno=10 and lower(job)='manager') or
         (deptno=20 and lower(job)='clerk') or
         (sal>=2000 and lower(job) not in ('manager','clerk'));
--7  找出收取佣金的员工的不同工作
/*
      null类型数据要用  is null和is not null来判断,其它的任何判断都是错误的
      如=null,<null等.
      select nvl(comm,0) from emp;
      select * from emp where comm is null;
      select * from emp where comm is not null ;
*/
select distinct job from emp
where comm is not null ;
--8  找出不收取佣金或收取佣金低于100的员工
select * from emp
where comm is null or comm<100;
--9  找出各月倒数第3天受雇的所有员工
select * from emp where last_day(hiredate)-2=hiredate ;
--10  找出早于12年前受雇的员工
select * from emp where months_between(sysdate,hiredate)/12 > 12 ;
--11  以首字母大写的方式显示所有员工的姓名
select upper(ename) from emp;
--12  显示正好为5个字符的员工的姓名
select ename from emp where ename like '_____';
--13  显示不带有"R"的员工的姓名
select ename from emp where ename not like '%R%';
--14  显示所有员工的姓名的前三个字符
select substr(ename,1,3) ename from emp ;
--15  显示所有员工的姓名,用"a"代替所有的"A"
select replace(ename,'A','a') ename from emp ;
--16  显示满10年服务年限的员工的姓名和受雇日期
select ename,hiredate from emp where months_between(sysdate,hiredate)/12 > 10 ;
--17  显示员工详细信息,按姓名排序
select * from emp order by ename;
--18  显示员工的姓名和受雇日期,根据其服务年限,将最老的员工排在前面
select ename,hiredate from emp order by hiredate asc ;
--19  显示所有员工的姓名,工作和薪金,按工作的降序排序,若工作相同则薪金排序
select ename,job,sal from emp order by job desc,sal;
--20  显示所有员工的姓名,加入公司的年份和月份,按受雇日期所在月排序,若月份相同则将
--最早年份的员工排在前面
select  ename,to_char(hiredate,'YYYY') year,to_char(hiredate,'MM') month from emp
order by month,year;
--21  显示在一个月为30天的情况,所有员工的日薪金,忽略余数
select round(sal/30) from emp ;
--22  找出在(任何年份的)2月受雇的所有员工
select * from emp where to_char(hiredate,'mm')=2;
--23  对于每个员工,显示其加入公司的天数
select round(sysdate-hiredate) day_num,ename from emp;
--24  显示姓名字段的任何位置包含"A"的所有员工的姓名
select ename from emp where ename like '%A%' ;
--25  以年月日的方式显示所有员工的服务年限(大概)
select    trunc(temp.days/365) years,
             trunc((temp.days-trunc(temp.days/365)*365)/30) months,
             trunc(temp.days-(trunc(temp.days/365)*365+trunc((temp.days-
trunc(temp.days/365)*365)/30)*30)) days
        from    (select ceil(sysdate-hiredate) days from emp ) temp ;
```

六、复杂查询

```
/*
   多表查询
*/
--多表查询时,将产生笛卡尔积(多表行相乘)
select * from emp,dept ;
--加where条件过滤笛卡尔积
select * from emp,dept where emp.deptno=dept.deptno ;
--为表取别名
select * from emp e,dept d where e.deptno=d.deptno ;
--查询雇员姓名,所在部门编号和名称
select e.ename  姓名,d.deptno  部门编号,d.dname  部门名称
from emp e,dept d
where e.deptno=d.deptno ;
--查询雇员姓名,工作,领导的姓名
select e1.ename  姓名,e1.job  工作,e2.ename  领导
from emp e1,emp e2
where e1.mgr=e2.empno ;
--查询雇员姓名,工作,领导姓名及部门名称
select e1.ename  姓名,e1.job  工作,e2.ename  领导,d.dname  部门名称
from emp e1,emp e2,dept d
where e1.mgr=e2.empno and e1.deptno=d.deptno ;
--查询雇员姓名,工作,工资及工资等级
select e.ename  姓名,e.job  工作,e.sal  工资,s.grade  工资等级
from emp e,salgrade s
where e.sal between s.losal and s.hisal ;
--查询雇员姓名,工作,工资及工资等级,要求工资等级显示为
--A B C D E
select e.ename  姓名,e.job  工作,e.sal  工资,
          decode(s.grade,1,'E',2,'D',3,'C',4,'B',5,'A')  工资等级
from emp e,salgrade s
where e.sal between s.losal and s.hisal ;
/*
        左右连接
*/
--查询雇员姓名,工作,领导的姓名  (左连接  =(+) :以左表为主)
select e1.ename  姓名,e1.job  工作,e2.ename  领导
from emp e1,emp e2
where e1.mgr=e2.empno(+) ;
/*
        SQL:1999语法对多表连接的支持
*/
--交叉连接:cross join  产生笛卡尔积
select * from emp cross join dept ;
--自然连接:natural join   自动进行关联字的匹配
select * from emp natural join dept ;
--指定关联操作列  join ... using(column)
select * from emp join dept using(deptno) ;
--自己编写条件  join...on(子句)
select * from emp join dept on emp.deptno=dept.deptno ;
--左连接(左外连接) left outer join ... on
select * from emp left outer join dept on emp.deptno=dept.deptno ;   --以雇员表为主,
有14条记录
--右连接(右外连接) right outer join ... on
select * from emp right outer join dept on emp.deptno=dept.deptno ;    --以部门表为
主,有15条记录
/*
          排序\分组
*/
--查询雇员姓名,年薪,按年薪从高到低排序
select ename,sal*12 income from emp
order by income desc ;
--查询每个部门中工资最高的雇员姓名,工作,工资,部门名称,最后按工资从高到低
排序
select e.ename,e.job,e.sal,d.deptno,d.dname
from emp e,dept d,
       (select max(e.sal) sal
```

```
        from emp e,dept d
        where e.deptno=d.deptno
        group by d.deptno ) temp
where e.deptno=d.deptno and e.sal=temp.sal
order by e.sal desc ;
/*
        分组函数
        count() -- 全部的记录数
        max()  -- 一组中最大值
        min()  -- 一组中最小值
        avg()  -- 一组中平均值
        sum()  -- 一组中求和
        若程序中使用了分组函数，则有两种情况可以使用:
        ¡¤程序使用了group by，并指定了分组条件，这样可以将分组条件一起查询出来
        ¡¤如果不用分组条件，则只能单独使用分组函数
        注意：
        ¡¤使用分组函数的时候，不能出现分组函数和分组条件以外的的字段
        ¡¤where条件中不允许使用分组函数，分组条件可以用having表示
*/

--查询每个部门的雇员数量
select count(empno),deptno from emp group by deptno;
--求出每个部门的平均工资
select avg(sal),deptno from emp group by deptno;
--按部门分组，并显示部门的名称，以及每个部门的员工数
select d.dname,emp_temp.count_empno
from dept d,(select count(empno) count_empno,deptno from emp group by deptno) emp_temp
where d.deptno=emp_temp.deptno;
--要求显示平均工资大于2000的部门编号和平均工资
select deptno,avg(sal)
from emp
group by deptno
having avg(sal)>2000;
--显示非销售人员工作名称以及从事同一工作雇员的月工资的总和，并且要满足从事同一工作的雇员
--的月工资大于$1500，输出结果按月工资的合计升序排列
select emp_temp.job, sum(emp_temp.sal) sum_sal
from ( select job,sal from emp where lower(job)<>'salesman') emp_temp
where emp_temp.sal>1500
group by emp_temp.job
order by sum_sal asc;
--求出平均工资最高的部门
/*
        分组函数可以嵌套,但不能再出现分组条件的列
        如下面的代码是错误的:
        select deptno,max(avg(sal)) from emp group by deptno
*/
select max(avg(sal)) from emp group by deptno;
/*
        子查询
        in : 指定查询范围
        any:
                =any 与 in 作用一样
                >any : 查询大于指定范围中的最低值
                <any : 查询小于指定范围中的最大值
        all:
                >all : 大于指定范围中最大的值
                <all : 小于指定范围中最小的值
*/

--要求查询出比7654工资要高的全部雇员的信息
select *
from emp
where sal > (select sal from emp where empno=7654) ;
--要求查询工资比7654高，与7788从事相同工作的全部雇员信息
select *
from emp
where sal > (select sal from emp where empno=7654) and
        job = (select job from emp where empno=7788);
--查询出工资最低的雇员姓名，工作，工资
select ename,job,sal
from emp
where sal = (select min(sal) from emp);
--要求查询出部门名称，部门的员工数，部门的平均工资，部门的最低收入雇员姓名
select  result_temp.dname  ,  result_temp.count_empno  , result_temp.avg_sal  , result_temp.min_sal , e.ename
from emp e,
  (select d.dname dname,emp_temp.count_empno count_empno,emp_temp.avg_sal avg_sal,emp_temp.min_sal min_sal
                from dept d,(select deptno, count(empno) count_empno,avg(sal) avg_sal,min(sal) min_sal from emp group by deptno) emp_temp
                where d.deptno = emp_temp.deptno
    ) result_temp
where e.sal=result_temp.min_sal;

/*
        另一种方法
*/
select d.dname,ed.c,ed.a,ed.min,e.ename
from dept d,emp e,
        (select deptno,count(empno) c,avg(sal) a,min(sal) min
                from emp
                group by deptno ) ed
where d.deptno=ed.deptno and e.sal=ed.min;
```

七、表的管理

```
/*
  目前主流数据库:
  微软:SQL Server和Access
  瑞典MySQL AB公司mysql
  IBM公司:db2
  美国Sybase公司:Sybase
  IBM公司:informix
  美国Oracle公司:oracle
*/

--查询所有表
select * from tab ;
--创建表person
create table person
(
        id varchar2(18) not null,
        name varchar(10) not null,
        sex varchar2(2) default('男'),
        age number(3) ,
        birthday date
);
--default写法有两种
create table person2
(
        id varchar2(18) not null,
        name varchar(10) not null,
        sex varchar2(2) default '男',
        age number(3) ,
        birthday date
);
--插入数据
insert into person(id,name,sex,age,birthday) values(100011,'张三','女',20,to_date('2009-10-10','yyyy-mm-dd'));
insert into person(id,name,sex,age,birthday) values(100012,'李四',default,28,to_date('2009-10-10','yyyy-mm-dd'));
insert into person(id,name,age,birthday) values('100013','王五',30,to_date('2009-10-10','yyyy-mm-dd'));
--插入数据 (可以从其它表中将数据复制插入)
/*
        当前用户为dboy/pass,复制scott/tiger中的emp表中一条记录
*/
insert into emp select * from scott.emp where empno=7369;
--查询表
select * from emp ;
select * from person;
--删除表
drop table person ;
--复制表
create table temp as select * from person2 ;
--修改表
alter table temp add address varchar(50) ;
select * from temp ;
alter table temp modify address varchar(100)  ;
alter table person2 add address varchar(100) default '暂时无地址';
alter table person2 drop column address ;
--重命名表
```

```
rename temp to newtemp;
select * from tab ;

--初始化表
truncate table newtemp ;

--约束
--1  主键约束  primary key
create table t1(name varchar(10) primary key);
insert into t1(name) values('person');
insert into t1(name) values(null);            --默认不为空
insert into t1(name) values('person');     --违反了唯一约束条件
create table t2(name varchar(10),constraint pk_name primary key(name) ) ;
create table t3(name varchar(10));
alter table t3 add constraint pk_name2 primary key(name);

--2  非空约束  not null
create table t4(name varchar(10) not null,sex varchar(2) not null);

insert into t4(name,sex) values(null,'男');
insert into t4(sex) values('女');

--3  唯一约束  unique

create table t5(name varchar(10) not null unique,tel varchar(12) not null unique);
insert into t5(name,tel) values('zs','13218102560');
insert into t5(name,tel) values('zs','13218102560');                --违反了唯一约束条件

create table t6(name varchar(10) not null,tel varchar(12) not null,constraint uk_name
unique(name));
insert into t6(name,tel) values('zs','13218102560');
insert into t6(name,tel) values('zs','13218102560');

--4  检查约束  check
create table t7(age int not null,sex varchar(2) not null);

alter table t7 add constraint chk_age check(age between 0 and 120);
alter table t7 add constraint chk_sex check(sex in('男','女'));

insert into t7(age,sex) values(150,'男');
insert into t7(age,sex) values(50,'中');

--5  外键约束  foreign key
drop table t8 ;
create table t8 (id int not null primary key,addres varchar2(50));
create table t9 (id int not null ,infoid int not null,name varchar2(50),constraint
fk_infoid_t8 foreign key(infoid) references t8(id));
insert into t8(id,addres)
select 1,'苏州' from dual union all
select 2,'浙江' from dual union all
select 3,'安徽' from dual ;

insert into t9(id,infoid,name)
select 1,1,'好' from dual union all
select 2,2,'较好' from dual union all
select 3,3,'最好' from dual    ;

select * from t9;

--查看约束
select constraint_name from user_constraints;
select        constraint_name,table_name        from        user_constraints        where
lower(constraint_name) like 'fk_%';

--删除约束
alter table t9 drop constraint fk_infoid_t8 ;

--查看表结构
--desc table_name只能用在sqlplus/sqlplusw中
select            column_name,data_type,data_length,nullable,data_default            from
all_tab_columns where lower(table_name)='t9' ;

--查看当前用户有多少表
select object_name,object_type from user_objects where lower(object_type)='table' ;

--查看数据库全部表
select * from all_tables;

--查看所有用户
select * from all_users;
```

八、视图

```
/*
   视图  view :
   1  建议命名规则:v$_开始
   2  视图可以简化查询
   3  视图查询安全性更高,隐藏不想显示的列
   4  表结构修改后,视图就得修改,增加了维护的难度
   5  可以通过视图更新表数据,但不建议使用
*/

--创建表
drop table t10;
create table t10 (id number(2),name varchar2(10));
--创建视图
create view v$myview as select name from t10 ;
--查看所有视图
select * from user_views;
--利用视图向表插入数据
insert into v$myview(name) values('张三');
insert into v$myview(name) values('李四');
insert into v$myview(name) values('王五');
--查询数据
select * from v$myview ;
--更改视图依赖的表结构
alter table t10 modify    id number(2)    default 8 ;
alter table t10 add tel char(11) default '13218102560';
alter table t10 add address varchar2(50) default 'no' not null ;
--删除视图
drop view v$myview;
--再插入数据将出错
insert into v$myview(name) values('小六');
```

九、序列同义词

```
/*
   序列  sequence :
       序列号,在每次取的时候自动增加(第一次使用是初始值)
       start with number :  从多少开始增长
       increment by number :  每次增长多大
       nomaxvalue :  没有最大值,不限制最在值
       nocycle :  不循环
       cache number :  预先在内存里存放一些sequence ,  这样存取速度快.
       nocache:  数据库突然不正常down掉(shutdown abort),  cache中的sequence就
会丢失。  所以可以在create sequence的时候用nocache防止这种情况

   使用序场所:
   1. insert语句  values
   2. update语句  set

*/

--创建序列
/*需要有  create sequence  或  create any sequence权限*/
create sequence seq_up1 start with 1 increment by 1 nomaxvalue nocycle    cache 10;
create sequence seq_up2 start with 10 increment by 2 maxvalue 100 cycle nocache;
--使用序列
select seq_up1.nextval from dual ;    --让序列增长,并返回下一个值
select seq_up1.currval from dual ;    --返回序列当前值
select seq_up2.nextval from dual ;
select seq_up2.currval from dual ;
insert into t10(id,name,address) values(seq_up2.nextval,'张二','不详');
--修改序列
alter sequence seq_up2 increment by 1 maxvalue 1000 ;
--删除序列
drop sequence seq_up2 ;

/*
       同义词  synonyms
       优点：节省大量的数据库空间，对不同用户的操作同一张表没有多少差别;
           扩展的数据库的使用范围，能够在不同的数据库用户之间实现无缝交互;
           同义词可以创建在不同一个数据库服务器上，通过网络实现连接。
*/
--创建同义词
conn sys/change_on_install ;    --以sys/change_on_install身份登录进去
select * from emp ;                --查看emp提示无此表或视图
```

```
select * from scott.emp ;        --可以查询,可知emp表是属于scott的
create synonym emp for scott.emp ;   --创建同义词emp,它的作用是scott.emp的别名
select * from emp ;                  --利用创建的同义词(表的别名)来查询,ok
--查看所有同义词
select * from user_synonyms ;
--删除同义词
drop synonym emp ;
--创建公共同义词
create public synonym dept for scott.dept;
--删除公共同义词
drop public synonym dept ;
```

十、用户管理

```
/*
   用户管理
用户创建后，系统会自动创建一个同名的方案，该方案中包括表、视图、索引触发表器等数据库对象，该用户创建的对象都保存在自己的方案中。
*/
--显示当前用户
show user;
select user from dual ;
--查询所有用户
select * from all_users;
--备份用户资源
exp
/*
        cmd --> 切换到导出目录(如：e:\temp)
        这里是导出scott下的资源，所以在用户名中输入scott/tiger
        接下来全部默认即可
*/
--创建用户  dboy/pass
conn sys/change_on_install;
create user dboy identified by pass default tablespace users quota 10m on users ;
commit;
--给新用户授权
grant create session,create table,create view,create sequence to dboy;
commit;
--赋予新用户角色  (一次性赋予更多授权)
grant connect,resource to dboy ;
--授于访问其它用户表的访问权限
grant select,delete,update on scott.emp to dboy;
--回收访问其它用户表的访问权限
revoke select,delete,update on scott.emp from dboy;
--让用户在登录时修改密码
alter user dboy password expire;
--查看所有用户  (在sysdba权限下)
select * from dba_users;
--导入数据
imp
/*
        cmd --> 同样还是在导出目录下操作
        用户名中输入新用户名和密码(如dboy/pass)
        接下来一路默认
        再要求输入用户名时输入：scott(即导出的是谁的资源，注意这里不需要输入scott的密码)
*/
--查看当前用户下的表
select * from user_tables;
--解锁用户
alter user dboy account lock ;
alter user dboy account unlock ;
--修改用户密码
alter user dboy identified by dboy ;
--还可以
password dboy;
--删除用户(如果该用户创建了对象就需要指定cascade级联删除其对象)
drop user dboy cascade;

/*
        profile管理用户口令,profile是口令限制,资源限制的命令集合,当建立数据库时,oracle会自动工建立名称为
        default的profile.当建立用户没有指定profile选项,那oracle就会将default分配给用户.
        1. 账户锁定
          指定tea这个用户最多只尝试3次登录,锁定时间为2天,让我们看看怎么实现.
        create profile lock_account limit failed_login_attempts 3 password_lock_time 2;
            alter user tea profile lock_account;
        2. 终止口令
            要求用户tea每隔10天要修改自家的登录密码,宽限期为2天
            create profile myprofile limit password_life_time 10 password_grace_time 2;
            alter user tea profile myprofile
        3. 口令历史
            用户在修改密码时,不能使用以前使用过的密码,可使用口令历史
            create profile password_history limit password_life_time 10 password_grace_time 2 password_reuse_time 10
            password_reuse_tim 指定口令可重用时间即10天后就可以重用
        4. 删除profile
            drop profile password_history cascade;
*/
```

十一、 嵌套表 可变数组

```
/*
        嵌套表: 表中的一个字段的数据类型为一张表
*/
--1、创建类型：定义嵌套表数据类型
create type project_ty as object
(
        proid number(4),
        proname varchar2(50),
        prodate date
);
/
--2、指定类型名称，类型创建完成后，必须指定使用名称才能使用
create type project_nt as table of project_ty;
/

--3、创建嵌套表
create table department
(
        deptno number(2) primary key not null,
        dname varchar2(50) not null,
        projects project_nt
) nested table projects store as project_nt_tab_temp;

--显示错误
show errors;

--4、插入数据
insert into department(deptno,dname,projects)
values(1,'技术部',
        project_nt(
                project_ty(1001,'ERP',sysdate),
                project_ty(1002,'CRM',sysdate),
                project_ty(1003,'OA',sysdate)
        )
);

--5、查询记录
select * from department;
--查询一个部门中的全部项目
select * from table(select projects from department where deptno=1);

--6、更新项目编号为1001的名称为"测试项目"
update table (select projects from department where deptno=1) pro
set value(pro)=project_ty(1001,'测试项目',to_date('2009-10-12','yyyy-mm-dd')) where pro.proid=1001;

/*
        可变数组:嵌套表的升级版
*/

--1、定义工人数据类型
create type worker_info as object
(
        id number,
        name varchar2(50),
        sex varchar2(6)
);
/
--2、指定数组类型
create type worker_info_list as varray(10) of worker_info;
/
--3、定义部门表,一个部门中可能有多个工人
drop table department;
create table department
```

```
(
        deptno number(2) primary key not null,
        dname varchar2(50) not null,
        workers worker_info_list
);

--4、插入数据
insert into department(deptno,dname,workers)
values(20,'后勤部',
        worker_info_list(
                worker_info(1,'张三','男'),
                worker_info(2,'李四','女'),
                worker_info(3,'王五','男')
        )
);
--5、查询数据
select * from department;
```

十二、PLSQL 语法

```
/*
   PL/SQL : Oracle内部语言
            SQLServer的是:TSQL
   Procedural Language/SQL  是oracle在标准的SQL语言上的扩展,PL/SQL不仅允许嵌入SQL语言,还可以
   定义变量和常量,允许使用条件语句和循环语句,允许使用例外处理各种错误,这样使得它的功能变得更加的强大.
  优点:提高应用程序的运行性能,模块化的设计思想,减少网络传输量,提高安全性
SQL语言:有两种SQL1992/SQL1996   没有循环控制结构
   PL/SQL块由三个部分组成:定义部分,执行部分,例外处理部分
   declare
          定义部分 -- 定义常量,变量,游标,例外,复杂数据类型
   begin
          执行部分 -- 要执行的PL/SQL语句和SQL语句
   exception
          例外部分 -- 处理运行的各种错误
   end;

   调用过程
   exec  过程名
   call  过程名
*/
--简单打印一句话
--打开显示
set serveroutput on ;
begin
   dbms_output.put_line('Hello PL/SQL!');
end ;
/
/*
      变量声明的规则
      1 变量名不能够使用保留字
      2 第一个字符必须是字母
      3 变量名最多包含30个字符
      4 不要与数据库的表或者列同名
      5 每一行只能声明一个变量

      常用的变量类型
      1 binary_integer : 整数,主要用来计数而不是用来表示字段类型
      2 number: 数字字符
      3 char : 定长字符串
      4 varchar2 : 变长字符串
      5 data : 日期
      6 long : 长字符串最长2GB
      7 boolean : 布尔类型,可以取值true|false和null值

*/
--声明变量
declare v_name varchar2(20) ;
declare
        v_temp number(1);
        v_count binary_integer :=0;
        v_sal number(7,2) :=4000.00;
        v_date date :=sysdate ;
        v_pi constant number(3,2) :=3.14;
        v_valid boolean :=false;
        v_name varchar2(20) not null :='accp' ;
begin
        dbms_output.put_line('v_temp value :'||v_temp);
end;

--根据表中字段类型声明变量类型的
declare
        v_empno number(4);
        v_empno2 emp.empno%type;
        v_empno3 v_empno2%type;
begin
        dbms_output.put_line('test');
end;

/*
   复杂变量
   Table变量类型 : 集合  (表示集合时要自己定义的一个新的数据类型)
   Record变量类型 : 类 (代表实体,一条记录)
*/
--定义Table变量类型
--Table  变量类型(数组)
declare
        type type_newname is table of emp.empno%type index by binary_integer; --声明一个数组类型
                v_empnos type_newname ;   --用数组类型定义变量
begin
        v_empnos(0) :=7369;
        v_empnos(2) :=7839;
        v_empnos(-1):=9999;
        dbms_output.put_line(v_empnos(-1));
end;

--Record变量类型
declare
        type type_record_dept is record
                (
                        deptno dept.deptno%type,
                        dname dept.dname%type,
                        loc dept.loc%type
                );
        v_temp type_record_dept ;
begin
        v_temp.deptno :=50 ;
        v_temp.dname    :='aaa';
        v_temp.loc      :='bj';
        dbms_output.put_line(v_temp.deptno || ' ' || v_temp.dname);
end;

--使用%rowtype  声明record变量
declare
        v_temp dept%rowtype;     --直接用一张表的行来定义record,表结构变化时record也在变
begin
        v_temp.deptno :=50;
        v_temp.dname    :='aaa';
        v_temp.loc      :='bj';
        dbms_output.put_line(v_temp.deptno||' '||v_temp.dname);
end ;

/*
      PL/SQL中的SQL语句
      select  语句返回有且只有一条记录
*/
--select语句
declare
        v_ename emp.ename%type;
        v_sal emp.sal%type;
begin
        select ename,sal into v_ename,v_sal from emp where empno =7369;
        dbms_output.put_line(v_ename || ' '||v_sal);
end;

declare
        v_temp emp%rowtype;
begin
        select * into v_emp from emp where empno=7369;
        dbms_output.put_line(v_temp.ename);
end;
```

```
--insert 语句
declare
        v_deptno dept.deptno%type :=10;
        v_dname dept.dname%type :='aaa';
        v_loc dept.loc%type :='beijing' ;
begin
        insert into dept2 values(v_deptno,v_dname,v_loc);
        commit;
end;

--update语句
declare
        v_deptno emp2.deptno%type :=10 ;
        v_count number;
begin
        update emp2 set sal = sal/2 where deptno = v_deptno ;
        --select deptno into v_deptno from emp2 where empno = 7369;
        --select count(*) into v_count from emp2 ;
        dbms_output.put_line(sql%rowcount||'条记被影响');
        commit ;
end;

--DDL语句
begin
   execute immediate 'create table t_temp(nnn varchar2(20) default "aaa")' ;
end ;

--语句块
declare
        v_name varchar2(20);
begin
        v_name := 'jack' ;
        dbms_output.put_line(v_name);
end;

--带异常捕获的语句块
declare
        v_num number :=0 ;
begin
        v_num :=2/v_num ;
        dbms_output.put_line(v_num);
exception
        when others then
                dbms_output.put_line('error');
end;

--分支语句
--if 取出7369的薪水，如果<1200，则输出'low'，如果<2000则输出'middle'，否则
'high'
declare
        v_sal emp.sal%type;
begin
        select sal into v_sal from emp where empno=7369;
        if(v_sal<1200) then
                dbms_output.put_line('low');
        elsif(v_sal<2000) then
                dbms_output.put_line('middle');
        else
                dbms_output.put_line('high');
        end if;
end;

--循环   (相当于do...while)
declare
        i binary_integer :=1;
begin
        loop
                dbms_output.put_line(i);
                        i := i+1 ;
                        exit when(i>=11);
        end loop;
end;

declare
        j binary_integer :=1;
begin
        while j<11 loop
                dbms_output.put_line(j);
                        j := j+1 ;
        end loop ;
end;

begin
        for k in 1..10 loop
                dbms_output.put_line(k);
        end loop;
        for k in reverse 1..10 loop
                dbms_output.put_line(k);
        end loop;
end;

--错误处理
declare
        v_temp number(4);
begin
        select empno into v_temp from emp where deptno =10 ;
exception
        when too_many_rows then
                dbms_output.put_line('太多记录了');
        when others then
                dbms_output.put_line('error');
end;

declare
        v_temp number(4);
begin
        select empno into v_temp from emp where empno=2222;
exception
        when no_data_found then
                dbms_output.put_line('没数据');
end;

--错误记录
create table errorlog
(
        id number primary key,
        errcode number,
        errmsg varchar2(1024),
        errdate date
);

create sequence seq_errorlog_id start with 1 increment by 1;

declare
        v_deptno dept.deptno%type :=10 ;
        v_errcode number ;
        v_errmsg varchar2(1024) ;
begin
        delete from dept where deptno=v_deptno;
        commit;
exception
        when others then
                rollback;
                v_errcode :=SQLCODE;
                v_errmsg    :=SQLERRM;
                insert into errorlog
                values(seq_errorlog_id.nextval,v_errcode,v_errmsg,sysdate);
                commit;
end;

select * from errorlog;
```

十三、 索引

```
/*
   主键约束/唯一约束 ： 系统会自动建立索引,以SYS_开始
   1. 为一个表的列或组合列建立索引后,读取的速度加快
   2. 但写的速度却减慢了,因为插入,修改和删除数据后,还要更新索引
   3. 索引也需要空间,系统要占用大约为表1.2倍的硬盘和内存空间来保存索引,
增加了空间负担

   索引创建的原则
   1. 在大表上建立索引才有意义
   2. 在where子句或是连接条件上经常引用的列上建立索引
   3. 索引的层次不要超过4层
```

```
   提高查询效率是以消耗一定的系统资源为代价的,索引不能盲目的建立,这是考验一个dba是否优秀的很重要的指标

   索引分类:
   1. 按照数据存储方式,分为B*树,反向索引,位图索引
   2. 按照索引列的个数分为,单列索引,复合索引
   3. 按照索引列值的唯一性,分为唯一索引和非唯一索引
   此外还有函数索引,全局索引,分区索引等.
*/

--查看有多少张表
select * from user_tables;

--查看表t10结构
select column_name,data_type,data_length,nullable,data_default
from all_tab_columns where lower(table_name)='t10';
--查看表t10内容
select * from t10 ;
--建立索引
create index idx_t10_name on t10(name);
--删除索引
drop index idx_t10_name;

--查看所有索引(从字典数据表中查询)
select index_name,table_name from user_indexes
where lower(index_name)='idx_t10_id_name' ;
--建立复合索引
create index idx_t10_id_name on t10(id,name);
```

十四、游标

```
/*
   游标  cursor
*/
declare
   cursor c is      --声明游标时并不从数据库查询数据
      select * from emp ;
   v_emp c%rowtype;
begin
   open c;    --open游标才真正的查询数据保存到内存
   loop
      fetch c into v_emp;
      exit when (c%notfound);
      dbms_output.put_line(v_emp.ename);
   end loop;
         close c;    --关闭游标
end;
/
declare
   cursor c is
      select * from emp ;
   v_emp c%rowtype;
begin
   open c;
   fetch c into v_emp;
   while(c%found) loop
      dbms_output.put_line(v_emp.ename);
      fetch c into v_emp ;
   end loop;
         close c;
end;
/

declare
   cursor c is
      select * from emp ;
begin
   for v_emp in c loop
         dbms_output.put_line(v_emp.ename);
   end loop;
end;
/

--带参数游标
declare
         cursor c(v_deptno emp.deptno%type,v_job emp.job%type) is
                  select ename,sal from emp where deptno=v_deptno and job=v_job;
         --v_temp c%rowtype;
begin
         for v_temp in c(30,'CLERK') loop
                  dbms_output.put_line(v_temp.ename);
         end loop;
end;
/
--可更新游标
declare
         cursor c is
                  select * from emp2 for update;
begin
         for v_temp in c loop
                  if(v_temp.sal<2000) then
                           update emp2 set sal=sal*2 where current of c ;
                  elsif(v_temp.sal=5000) then
                           delete from emp2 where current of c;
                  end if;
         end loop;
         commit;
end;
/
```

十五、 存储过程

```
/*
   存储过程  Procedure
   创建存储过程必须有create procedure权限
*/
--打开显示
set serveroutput on;
--创建一个存储过程
create or replace procedure p_update_emp
as
         cursor c is
                  select * from emp for update;
begin
         for v_emp in c loop
                  if(v_emp.deptno = 10) then
                           update emp set sal=sal+10 where current of c;
                  elsif(v_emp.deptno = 20) then
                           update emp set sal=sal+20 where current of c;
                  else
                           update emp set sal=sal+50 where current of c;
                  end if ;
         end loop;
         commit;
end;
/
--查询所有存储过程
/*
   这里比较好玩:根据一般数据字典表中的列命名,这里p_update应为procedure_name
   但是不是,它是object_name,procedure_name为空
*/
select * from user_procedures;
--执行存储过程 (方式一)
begin
   p_update_emp;
end;
--执行存储过程 (方式二)
exec p_update_emp;

--创建带参数存储过程
/*
         in : 表输入参数
         out: 表输出参数
         不加关键字的表示默认输入参数
         两个都加的既表示输入参数又表示输出参数

         注意:存储过程的参数只能指定类型,而不能指定大小
         如   v_a number(2)是错误的,正确为v_a number
*/
create or replace procedure p_in_out
         (v_a in number,v_b number,v_ret out number,v_temp in out number)
as
         begin
                  if(v_a > v_b) then
```

```
                v_ret := v_a ;
            else
                v_ret := v_b ;
            end if ;
            v_temp := v_temp +1 ;
        end;
/

--执行带参存储过程
declare
        v_a number := 3 ;
        v_b number := 4 ;
        v_ret number;
        v_temp number := 5 ;
begin
        p_in_out(v_a,v_b,v_ret,v_temp);
        dbms_output.put_line(v_ret);
        dbms_output.put_line(v_temp);
end;
/
--结果打印 4 和 6

--删除存储过程
drop procedure p_update_emp;
```

十六、函数

```
/*
   函数 function
      可以理解为函数是有返回值的存储过程
*/

--创建函数
create or replace function f_calc_add
        (v_a number,v_b number)
        return number
is
        begin
                return v_a + v_b ;
        end;

--使用函数
select f_calc_add(10,20) from dual ;
--打印结果: 30
declare v_result number := f_calc_add(30,40);
begin
        dbms_output.put_line(v_result);
end;
--打印结果: 70
--查询函数
select * from user_procedures;
--删除函数
drop function f_calc_add;
```

十七、触发器

```
/*
   触发器 trigger
        触发器必须依附于表才能起作用
*/

--创建日志表 emp_log,用于记录对表emp的所有操作日志
create table emp_log
(
        uname varchar2(20),
        action varchar2(10),
        atime date
);

--创建触发器
/*
        触发条件:insert|delete|update 对某张表做插入|删除|更新的操作
        触发时间:after|before 操作之后|操作之前
        触发行:each row 操作影响一行触发一次，不写表求一次操作触发一次
*/
create or replace trigger trig
        after insert or delete or update on emp for each row
begin
        if inserting then
                insert into emp_log(uname,action,atime) values(USER,'insert',sysdate);
        elsif updating then
                insert into emp_log(uname,action,atime) values(USER,'update',sysdate);
        elsif deleting then
                insert into emp_log(uname,action,atime) values(USER,'delete',sysdate);
        end if ;
end;
/
--触发
update emp set sal=sal*2 where deptno=30;
delete from emp where empno = 7369 ;
--查看日志表
select * from emp_log;
--? : 有外键关系时,主表中的主键存在外表的引用关系,因此不能随意更新,但可以用触发器解决这个问题
      --如: update dept set deptno=99 where deptno=10
--删除触发器
drop trigger trig;
create or replace trigger trig
        after update on dept for each row
begin
        update emp set deptno = :NEW.deptno where deptno = :OLD.deptno ;
end;
/
/*
通常一条update语句会产生新旧两个状态 :NEW代表新状态 :OLD代表旧状态
*/
update dept set deptno=99 where deptno=10;
```

十八、设计范式

```
/*
   数据库设计三范式
   数据库设计应该遵循的规则,给一个姓范的人制定的,所以叫范式
   第一范式: 1NF Normal Formate
            1. 要有主键,任何表都要求有主键
            2. 列不可分
            确保每列的原子性,即实体中的某个属性不能有多个值或不能有重复值

            产品号;包含多个信息,但只设计一列,就打破了第一范式

   第二范式: 2NF
            目标是确保表中的每列都和主键相关,要求每个表只描述一件事件,数据库中表的每个实例或行必须可以被唯一区分
            非主键列不能依赖于组合主键列的一部分
            消除部分依赖,在满足1NF范式的情况下,只要不是组合主键都满足第二范式

   第三范式: 3NF
            确保每列都和主键直接相关,而不是间接相关.要求一个数据库表中不包含其他表中包含的非主键的信息
            消除传递依赖
            例如:学生表:学号,班级编号,班级名称
            1) 班级编号    依赖    学号(主键)
            2) 班级名称    依赖    班级编号
            3) 从而推断出    班级名称    依赖    学号    (这就是传递依赖,不是直接依赖,所以这张表就要拆分)
*/
/*
        1NF :列的原子性和要有主键
        数据库表中的字段都有是单一属性的，不可再分，这个单一属性由基本类型构成，包括整型、实数
        字符型、逻辑型、日期型等。
*/
create table person
(
        pid number(4) primary key not null, --人员编号
        name varchar2(10),    --人员姓名
        info varchar2(1000)   --人员信息
);
insert into person(pid,name,info) values(1111,'张三','1983年11月23日出生，现住地址为北京市西城区...');
/*
        可见上表中的info字段由以下几部分信息组成：
        ¡ñ生日：1983年11月23日
        ¡ñ省市：北京
        ¡ñ地区：西城区
        ¡ñ更加详细的信息：...
```

```
        所以违背了第一范式，必须修改为：
*/
drop table person;
create table person
(
        pid number(4) primary key not null,--编号
        name varchar2(10),    --姓名
        birthday date,  --生日
        area varchar2(100),    --省市
        subarea varchar2(100),        --区域
        info varchar2(1000)   --更加详细信息
);
insert into person values(1111,'张三',to_date('1983-11-23','yyyy-mm-dd'),'北京','西城区','...');
/*
        2NF:每列都和主键相关，不相关的放入其它表，一个表只能描述一件事情
        数据库中不存在非关键字段对任一候选关键字段的部分函数依赖(部分函数依赖指的是存在组合键
        关键字中的某些字段是非关键字的情况),也即所有非关键字段都完全依赖于任意一组候选关键字。
*/
create table selectcourse
(
        stuno varchar2(50),    --学生编号
        stuname varchar2(50),        --学生姓名
        stuage number,                --学生年龄
        cname varchar2(50),   --课程名称
        grade number,         --成绩
        credit number         --学分
);
insert into selectcourse values('s001','张三',20,'java',89.0,3);
insert into selectcourse values('s002','李四',21,'java',79.0,3);
insert into selectcourse values('s003','王五',22,'java',69.0,3);

/*
        由以上学生选课表的设计来看，课程名称和学分明显示复复，出现数据冗余
        根据第二范式修改为：
*/
create table student
(
        stuno varchar2(50) primary key ,--学生编号
        stuname varchar2(50),         --学生姓名
        stuage number,                --学生年龄
);
create table course
(
        cid number primary key,       --课程编号
        cname varchar2(50),   --课程名称
        credit number         --课程学分
);
create table selectcourse
(
        stuno varchar2(50) ,
        cid number,
        grade number,
        constraint fk_stuno_Student foreign key(stuno) references student(stuno),
        constraint fk_cid_course foreign key(cid) references course(cid)
);
insert into student(stuno,stuname,stuage) values('s001','张三',20);
insert into student(stuno,stuname,stuage) values('s002','李四',21);
insert into student(stuno,stuname,stuage) values('s003','王五',22);
insert into course(cid,cname,credit) values('ca','java',3);
insert into course(cid,cname,credit) values('cb','oracle',5);
insert into selectcourse(sutno,cid,grade) values('s001','ca',89.0);
insert into selectcourse(sutno,cid,grade) values('s002','ca',79.0);
insert into selectcourse(sutno,cid,grade) values('s003','ca',69.0);
commit;

/*
        3NF:每列都和键直接相关，而非间接相关
        现在设计这样一张表
        学生表：学号、姓名、年龄、所在学院、学院地址、学院电话
        根据二级范式，里面包含三个事情，学生信息、学院信息、关系信息
*/
create table collage
(
        cid number primary key ,      --学院编号
        cname varchar2(50),           --学院名称
        caddress varchar2(100),               --学院地址
        ctel varchar2(20)             --学院电话
);
create table student
(
        stuno varchar2(50) primary key ,--学生编号
        stuname varchar2(50),         --学生姓名
        stuage number,                --学生年龄
);
create table student_collage
(
        stuno varchar2(50) ,   --学生编号
        cid number    ,        --学院编号
        constraint fk_stuno_student foreign key(stuno) references student(stuno),
        constraint fk_cid_collage foreign key(cid) references collage(cid)
);

/*
        但实际生活中，却是一个学院对应多个学生，一个学生只属于一个学院，
        因此上面根据第二范式设计不合理，根据第三范式修改为：
*/

create table collage
(
        cid number primary key ,      --学院编号
        cname varchar2(50),           --学院名称
        caddress varchar2(100),               --学院地址
        ctel varchar2(20)             --学院电话
);
create table student
(
        stuno varchar2(50) primary key ,--学生编号
        stuname varchar2(50),         --学生姓名
        stuage number,                --学生年龄
        cid number,             --学院编号
        constraint fk_cid_collage foreign key(cid) references collage(cid)
);

/*
        数据库设计工具：PowerDesigner(sybase)
*/
```

十九、数据库管理

数据库管理员　DBA (DataBase Administrator)

职责：

- ·安装和升级 oracle 数据库
- ·建库、表空间、表、视图、索引……
- ·制定并实施备份和恢复计划
- ·数据库权限管理，调优，故障排除
- ·对于高级 dba，要求能参与项目开发，会编写 sql 语句，存储过程，触发器，规则、约束和包

管理数据库的用户主要是 sys 和 system

sys:

所有 oracle 的数据字典的基表和视图都存放在 sys 用户中，这此基表和视图对于 oracle 的运行是至关重要的，

由数据库自己维护，任何用户不能手动更新。sys 用户拥有 dba、sysdba 和 sysoper 的角色或权限，

是 oracle 权限最高的用户。

sys 必须以 as sysdba 或 as sysoper 形式登录，不能以 normal 方式登录

system:

用于存放一级的内部数据，如 oracle 的一些的内部数据，如 oracle 的一些特性或工具的管理信息.

system 用户拥有 dba，sysdba 角色或系统权限。system 如果是以 normal 形式登录，它就是一个普通

用户，如果以 as sysdba 形式登录，实际上它就是作为 sys 用户登录。

sysdba 和 sysoper 权限的差别：

startup(启动数据库)　也可以

shutdown(关闭数据库)　也可以

alter database open/mount/backup　也可以

改变字符集　不可以

create database(创建数据库)　不可以

drop database　不可以

create spfile　可以

alter database archivlog(归档日志)　可以

alter database recover(恢复数据库) 只能完全恢复，不能执行部分恢复
拥有 restricted session(会话限制权限) 可以
可以让用户作为 sys 用户连接 可以进行一些基本操作，但不能查看用户数据
登录之后用户是 sys 登录之后用户是 public

dba 登录后可以关闭和启动数据库，数据库关闭后，普通用户无法登录，但 sysdba 可以登录
shutdown ; //关闭数据库
startup ; //启动数据库

管理初始化参数(用于设置实例或是参数特征)
show parameter ;
E:\Oracle10g\product\10.1.0\admin\accp\pfile\init.ora

数据库备份
逻辑备份：使用 export（命令为 exp）工具将数据对象的结构和数据导出到文件的过程
逻辑恢复：指当前数据库对象被误操作后利用工具 import（命令为 imp）把备份文件中的数据对象导入到数据库的过程

物理备份既可在数据库 open 状态下进行，也可以在关闭数据库后进行，但是逻辑备份和恢复只能在 open 状态下进行。

导出：分为三种：导出表、导出方案、导出数据库
exp 命令
userid : 用于指定执行导出操作的用户名，口令和连接字符串
tables : 用于指定执行导出操作的表
owner : 用于指定执行导出操作的方案
full=y : 用于指定执行导出操作的是数据库
inctype : 用于指定执行导出操作的增量类型
rows : 用于指定导出操作是否要导出表中的数据 ,当 rows=n 只导出表结构
file : 用于指定导出文件
导出表：
·导出自定的表
exp userid=scott/tiger@accp tables=(emp,dept) file=e:\temp\e1.dmp
·导出其它方案 必须为 sysdba 用户或具有 exp_full_database 权限
exp userid=system/manager@accp tables=(scott.emp) file=e:\temp\e2.dmp
·导出表结构
exp userid=scott/tiger@accp tables=(emp) file=e:\temp\e3.dmp rows=n
·直接导出方式 direct=y
exp userid=scott/tiger@accp tables=(emp) file=e:\temp\e4.dmp direct=y
这种方式比常规方式速度要快，当数据量大时，可以考虑此种方式
需要数据库的字符集与客户端字符集完全一致，否则会报错
导出方案
·导出自己的方案
exp userid=scott/tiger@accp owner=scott file=e:\temp\e5.dmp
·导出其它人的方案
exp userid=system/manager@accp owner=(dboy,scott) file=e:\temp\e6.dmp

导出数据库
exp userid=system/manager@accp full=y file=e:\temp\e7.dmp

恢复
导入表
imp userid=scott/tiger@accp tables=(emp,dept) file=e:\temp\e1.dmp
导入方案
imp userid=scott/tiger@accp file=e:\temp\e5.dmp
imp userid=system/manager@accp fromuser=system touser=(scott,dboy) file=e:\temp\e6.dmp
导入数据库
imp userid=system/manager@accp full=y file=e:\temp\e7.dmp

数据字典：
oracle 数据库中最重要的组成部分，它提供了数据库一些系统信息
它是只读表和视图的集合，所有者为 sys 用户，普通用户只能在数据字典上执行查询，而其
维护和修改是由系统自己自动完成的。
分为：
·数据字典基表 ： 存储数据库的基本信息 -- 不能直接访问
·数据字典视图 ： 基于基表所建立的视图，普通用户可以查询，分为 user_xxx、all_xxx 和 dba_xxx 三种类型
例如：
user_tables 可查询用户自己的所有表，
all_tables 可查询用户自己的所有表和可以访问的其它表 ，
dba_tables 可查询数据库中所有方案的表，它要求当前用户为 sysdba 或有 select any table 权限

动态性能视图 ： 记载了例程启动后的相关信息

用户、角色、权限
dba_users 显示数据库中用户的详细信息
dba_sys_privs 显示用户所具有的系统权限
dba_tab_privs 显示用户所具有的对象权限
dba_col_privs 显示用户具有有的列权限
dba_role_privs 显示用户所具有的角色

查询 oracle 中所有的系统权限
select * from system_privilege_map ;
查询 oracle 中所有的角色
select * from dba_roles;
查询 oracle 中所有的对象权限
select distinct privilege from dba_tab_privs;

表空间
数据库的逻辑组成部分，从物理上讲，数据库存放在数据文件中，从逻辑上讲，数据库存放在表空间中，
表空间由一个或多个数据文件组成。
数据库的逻辑结构：oracle 数据库的逻辑结构包括 表空间(tablespace)、段(segment)、区(extent)和块(block)
逻辑结构:
块---最小的单元，由一个或多个操作系统块组成，不可更改。
区---存储空间分配，回收和管理的基本单位。
段---对象(表,索引)创建时同时创建，一个对象只拥有一个段
表空间---最高级逻辑存储结构物理结构:
数据文件(DBF)---记录用户数据，包括:系统数据文件，回滚数据文件，临时数据文件， 用户数据文件。
控制文件(CTL)---记录数据库的物理结构
重做日志文件(LOG)---至少需要两个日志组，每个日志组至少有一个日志成员， 一个日志组中的成员是镜像关系，防止日志文件的损坏。
*归档情况下，需要归档的日志来不及归档，联机日志又需要被重新利用(增加日 志组解决)
*检查点事件还没有完成(日志切换引起检查点)，而联机日志需要被重新利用(增 大日志文件成员大小解决)
数据字典---存放数据库信息的地方，描述数据库中数据信息。字典中的表不能直接被访问 ，可以访问的只是数据字典中的视图。
*静态数据字典--用户访问数据字典时不会改变
*动态数据字典--依赖数据库运行的性能，反映数据库运行的一些内在信息软件结构:
实例---驻留在内存中的结构和一系列的进程，一个数据库可以对应多个实例，一个实例只 能对应一个数据库。
SGA 区---系统全局区，包括:数据库缓冲，重做日志文件，共享池，JAVA 池，大型池，其 他控制信息的结构
PGA 区---保存与用户进程相关的内存段
其他后台进程：
DBWR--数据写，将修改过的数据缓冲区的数据写入数据文件
LGWR--日志写，将重做日志缓冲区的数据写入重做日志文件
SMON--实例启动时对数据库进行恢复操作
PMON--清除失效的用户进程，释放用户进程所用的资源
CKPT--同步数据文件，日志文件和控制文件

数据库逻辑上是由一个或是多个表空间组成，通过表空间可以达到以下目的：
·控制数据库占用的磁盘空间
·dba 可以将不同的数据类型部署到不同的位置，有利于提高 I/O 性能，同时利于备份和恢复等管理操作

建立表空间 (要求有 create tablespace 权限)
create tablespace tp01
datafile
'e:\temp\data01_1.dbf' size 20m,
'e:\temp\data01_2.dbf' size 20m,
'e:\temp\data01_3.dbf' size 1m autoextend on maxsize 10m uniform size 128k;

使用表空间
create table mytemp(name varchar2(50)) tablespace tp01;

改变表空间状态 online(连机)|offline(脱机)|read only(只读)|read write(读写)
alter tablespace tp01 online;
表空间状态发生改变时，其相应的数据文件访问状态也会发生变化

1、知道表空间名，显示该表空间包括所的所有表
select * from all_tables where tablespace_name='tp01'

2、知道表名，查看该表所属的表空间
select tablespace_name,table_name from user_tables where table_name='mytemp'

扩展表空间
·增加数据文件
alter tablespace tp01 add datafile 'e:\temp\data01_4.dbf' size 3m;
·更改原有数据文件大小
alter database datafile 'e:\temp\data01_1.dbf' resize 30m
·设置数据文件自动增长
alter database datafile 'e:\temp\data01_2.dbf' autoextend on next 10m maxsize 100m

删除表空间　(连带也删除该空间的所有数据库对象和数据文件)
drop tablespace tp01 including contents and datafiles ;

移动数据文件
1、确定数据文件所有的表空间
select tablespace_name form dba_data_files where file_name='e:\temp\data01_1.dbf';
2、使表空间脱机
alter tablespace tp01 offline;
3、移动数据文件
拷贝数据文件到其它盘
4、物理上移动数据文件后，表空间中相应的数据文件位置也要进行逻辑修改
alter tablespace tp01 rename datafile 'e:\temp\data01_1.dbf' to 'c:\data01_1.dbf';
5、重新联机表空间
alter tablespace tp01 online;

查询数据库的表空间
select tablespace_name from dba_tablespaces;
/*
　dba 数据库管理员
*/

--查询所有的系统权限
select * from system_privilege_map;
--查询所有的角色
select * from dba_roles;
--确定角色的权限
select * from role_role_privs where role='DBA';　　-- 角色dba中的角色
select role,privilege from role_tab_privs where role='DBA';　-- 角色dba的对象权限
select role,privilege from role_sys_privs where role='DBA';　-- 角色dba的系统权限
--查询用户dboy所拥有的角色
select * from dba_role_privs where grantee='DBOY';　　-- 用户dboy的角色
select * from dba_sys_privs where grantee='DBOY';　　-- 用户dboy的系统权限
select * from dba_tab_privs where grantee='DBOY';　　-- 用户dboy的对象权限

--查询一个角色所对应的权限
select role,privilege from role_tab_privs where role='DBA';
--查询当前用户权限
select * from session_privs;

/*
　授予系统权限给用户,首先当前用户必须为DBA
　　with admin option 转授系统权限选项,但是将角色授予角色就不允许带此选项
　系统权限不会被级联回收
*/
-- 授予权限给DBOY
conn sys/chage_on_install as sysdba;　　-- DBA身份登录
create user ken identified by my123;　　-- 创建用户ken/my123
create user tom identified by my123;　　-- 创建用户tom/my123
grant create session to ken with admin option;　-- 授予用户ken登录权限,并且附带将这种权限授予ken
grant create table to ken ;　　-- 授予用户ken创建表的权限

conn tom/my123 ;　--登录失败,无登录权限
conn ken/my123 ;　--登录成功,因为有了登录权限
grant create session to tom ;　　--ken用户可以授予create session权限给用户tom,因为dba将此种权限授予了ken
grant create table to tom ; --失败,因为ken用户没有授予create table给别人的权限
conn tom/my123 ;　--登录成功,因为ken给了它登录权限

-- 回收权限
conn sys/change_on_install as sysdba ;　　-- DBA身份登录
revoke create session from tom ;　　-- 回收tom用户登录权限
conn ken/my123 ;　　-- 登录成功,权限回收不级联,虽然回收了tom用户的权限,但不影响ken用户

/*
　授予对象权限
　with grant option 转授对象权限选项
　对象权限会被级联回收
*/
conn sys/chagen_on_install as sysdba;　　-- DBA身份登录
create user user1 identified by my123;　　-- 创建用户user1/my123
grant create session to user1;　　-- 授予登录权限给用户user1
grant select on scott.emp to user1 ;　　-- 授予访问对象scott.emp的权限给用户user1
conn user1/my123 ;　　-- 用户user1登录,成功
select * from scott.emp;　　-- 查询成功!

grant update on scott.emp(sal) to user1 ;　　-- 只授予修改表emp中sal列的权限给用户user1
grant select on scott.emp(ename,job) to user1; -- 只授予访问表emp中的ename,job列的权限给用户user1

/*
　　角色 : 为了简化对权限的管理,oracle事先将一系列的权限打包到一个角色中,所以角色是相关权限的命令集合.
　　connect角色:具有一般开发人员需要的大部分权限,connect角色具有下列系统权限:
　　alter session
　　create cluster
　　create database link
　　create session
　　create table
　　create view
　　create sequence
　一般新建一个用户后,只需要将connect角色和resource角色赋予访用户就够了.

　　resouce角色:具有应用开发人员所需要的其它权限,比如建立存储过程,触发器等.
　　隐含了unlimited tablespace系统权限,具体包括:
　　create cluster
　　create indextype
　　create table
　　create sequence
　　create type
　　create procedure
　　create trigger
　dba角色:具有所有的系统权限,及with admin option选项,可以将任何系统权限授予其它用户
　但要注意dba角色不具备sysdba和sysoper的特权(启动和关闭数据库)

　自定义角色
　not identified 公用角色,可以采取不验证的方式建立角色
　identified by shunping 数据库验证角色,当激活访角色时要提供密码

　给角色授权与给用户授权没有太多区别,但要注意系统权限unlimited tablespace和对象权限
　with grant option选项不能授予角色,但可以授予用户.

*/

-- 自定义角色
create role my_role;　　-- 创建一个角色
grant select any table to my_role;　-- 授予查看任何表的权限给自定义的角色my_role
grant my_role to tom;　　-- 授予角色my_role给tom用户,这样tom用户就可以查看任何表了

-- 删除角色
drop role my_role;