



Oracle期末考试复习资料

原创：林大夕可

 [阅读原文](#) 

2020.05.26

前言

一、简介

1. 登录
2. 授权
3. 修改
4. 启用与禁用
5. 删除
6. 索引、序列、角色、用户

二、表

1. 创建表
 - 1.1 子查询创建表
 - 1.2 定义约束!!!
 - 1.3 管理约束
2. 更改表
 - 2.1 更改表名称
3. 删除表
4. 截断表
5. 注释表

三、视图

1. 创建视图
2. 重定义视图
3. 删除视图
4. 更改视图

四、DML

1. Insert!!!
 - 1.1 插入多行数据
2. Update!!!
 - 2.1 更新多列数据
3. Delete

五. 查询

1. 消除重复行
2. 比较

3. 日期相减
4. 数值转换
5. 空值置换
6. 条件表达式! ! !
7. 连接查询（高级查询）
8. 递归查询! ! !
9. 分组函数
10. 子查询
11. 嵌套查询

六. PL/SQL

1. PL/SQL编程基础
 - 1.1 符号
 - 1.2 声明变量
 - 1.3 引用
 - 1.4 转换员工姓名为小写
 - 1.5 PL/SQL中的select
 - 1.6 PL/SQL中的DML
 - 1.7 PL/SQL中的Commit & Rollback
 - 1.8 if
 - 1.9 loop/for
 - 1.10 游标Cursor
2. PL/SQL程序设计
 - 2.1 触发器
 - 2.2 过程函数

前言

这是我上学期Oracle期末考试前整理的一些资料，今天调整了部分格式后分享出来。希望能帮到要考Oracle（以写SQL语句为主的考试）的小🔥伴儿们~



一、简介

1. 登录

结尾无分号

```
sqlplus sys/11111 as sysdba
```

- 切换用户（关键词： Connect ）

```
connect scott/tiger
```

- 用户表中可使用的磁盘空间大小（关键词： Quota ）[了解]

```
create user linda
identified by 111111
default tablespace users
quota 10M on users;
```

- 用户默认密码已过期，用户登录前需修改（关键词： password expire ）[了解]

```
create user linda
identified by 111111
password expire;
```

2. 授权

- 建立会话（登录）系统权限：grant **create session** to linda;
- 创建数据表的系统权限：grant **create table** to linda;
- 授权和数据库建立连接的角色：grant **connect** to linda;

3. 修改

关键词：Alter

- 修改密码

```
alter user linda  
identified by 111112;
```

4. 启用与禁用

alter user linda **account lock**;

alter user linda **account unlock**;

5. 删除

若用户方案中存在对象，则需要用**cascade**

drop user linda **cascade**;

6. 索引、序列、角色、用户

- 索引是表的一个微型拷贝

```
create role hr_clerk  
identified by password;
```

注：**identified by**后为密码

```
create user user_name  
identified by password;
```

二、表

1. 创建表

```
create table haha  
(deptno number(2),  
  dname varchar2(14),  
  loc varchar2(13));
```

查看表信息：**describe** haha

1.1 子查询创建表

```
create table dept30  
as  
select empno, ename, sal*12 annsal  
from emp  
where deptno=30;
```

1.2 定义约束!!!

关键词： `constraint`

- 非空： `not null`

```
create table haha  
(empno number(4),  
  ename varchar2(10),  
  deptno number(7, 2) not null,  
  constraint emp_empno_pk primary key (empno));
```

- 唯一码约束: `unique`
- 主键约束: `primary key`
- 外键约束: `foreign key` , **references**
- Check约束: `check`

```
create table haha
(empno number(2),
  ename varchar2(10) not null,
  job varchar2(9),
  sal number(7,2),
  comm number(7,2),
  mgr number(4),
  hiredate date,
  deptno number(7,2) not null,
  constraint haha_ename_uk unique (ename),
  constraint haha_empno_pk primary key (empno),
  constraint haha_deptno_fk foreign key (deptno)
    references dept (deptno)
  constraint haha_deptno_ck check
    (deptno between 10 and 99));
```

Q: `number(7,2)`含义?

A: 7位有效数字, 其中包含2位小数

1.3 管理约束

1. 增加

```
alter table haha
add constraint haha_mgr_fk foreign key (mgr)
  references emp (empno);
```

2. 删除

```
alter table haha
drop constraint haha_mgr_fk;
```

```
alter table haha  
drop primary key cascade;
```

3. 禁用

```
alter table haha  
enable constraint haha_empno_pk;
```

```
alter table haha  
disable constraint haha_empno_pk cascade;
```

2. 更改表

关键词: alter , add , modify , drop column

```
alter table haha  
add (job varchar2(9));
```

```
alter table haha  
modify (job varchar2(10));
```

```
alter table haha  
drop column job;
```

```
alter table haha  
drop unused column;
```

2.1 更改表名称

关键词: rename ... to ...

```
rename haha to fafa;
```

3. 删除表

```
drop table haha;
```

若删除的表中包含有被其它表外部码引用的码，并希望删除表的同时删除其它表中的相关外部码约束，则需要加**cascade**语句

```
drop table haha cascade constraints;
```

4. 截断表

关键词： `truncate`

```
truncate table fafa;
```

5. 注释表

关键词： `comment`

```
comment on table emp is 'Employee Information';
```

三、视图

1. 创建视图

- 关键词： `create view ... as ...`

```
create view emp_view
as
select empno, ename, sal
from emp;
```


scott账户下创建视图会显示权限不足。解决方法如下

1. 以system/pw登录oracle;
2. 输入

```
grant create any view to scott;
```

3. 重新登录scott账户，即可创建视图

2. 重定义视图

- 关键词：create **or replace** view ...

```
create or replace view emp_view
as
select empno, ename, sal, loc
from emp, dept
where emp.deptno = dept.deptno;
```

3. 删除视图

- 关键词：drop

```
drop view emp_view;
```

4. 更改视图

```
create or replace view empvu10 (employee_number, employee_name, job_title)
as
select empno, ename, job
from emp
where deptno = 10;
```

思考：如何修改视图中某列的名称？

四、DML

1. Insert! ! !

关键词: insert into... **values**...

注意括号!

```
insert into emp (empno, ename, job, mgr, hiredate, sal, comm, deptno)
values (9000, 'GREEN', 'SALESMAN', 7782, sysdate, 2000, null, 10);
```

1.1 插入多行数据

```
insert into salesman (id, name, job, hiredate)
select empno, ename, job, hiredate
from emp
where job = 'SALESMAN';
```

2. Update! ! !

关键词: update... **set**... where...

```
update emp
set sal=8000
where ename='Anne';
```

2.1 更新多列数据

```
update emp
set (job, deptno) = (select job, deptno
                    from emp
                    where empno = 9000)
where empno = 7369;
```

3. Delete

关键词: delete **from**... where

```
delete from dept
where deptno = 50;
```

五. 查询

1. 消除重复行

关键词: `distinct`

```
select distinct deptno
from emp;
```

2. 比较

注意: is后接NULL或者not NULL, 不用来做其它比较, 如job = 'SALESMAN'不可将=改为is

1. in(list) [list列中成员匹配]

```
select empno, ename, sal, mgr
from emp
where mgr in (7902, 7566, 7788)
```

1.1 not

```
select ename, job
from emp
where job not in ('CLERK', 'MANAGER', 'ANALYST');
```

2. like [字符匹配]

- % : 匹配0或多个字符
- _ : 匹配单一字符

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

3. is null

```
select ename, mgr
from emp
where mgr is null;
```

3. 日期相减

- select empno, ename, job, **sysdate-hiredate** hiredays from emp;

4. 数值转换

- select ename, to_char(sal, '\$0,999.00') salary from emp;

5. 空值置换

1. NVL(expr1, expr2)

若expr1 = NULL, 返回expr2; 否则, 返回expr1

```
select ename, sal, (sal*12) + NVL(comm, 0) annual_sal from emp;
```

```
select ename, NVL(to_char(mgr), 'No Manager') status from emp where mgr is null;
```

2. NVL2(expr1, expr2, expr3)

若expr1 = NULL, 返回expr3; 否则, 返回expr2

```
select ename, sal, (sal*12) + NVL2(comm, 3000+comm, 3000) annual_sal from emp;
```

6. 条件表达式!!!

1. CASE

```
select ename, job, sal,  
       case job when 'ANALYST' then 1.10*sal  
                when 'CLERK' then 1.15*sal  
                when 'SALESMAN' then 1.20*sal  
       else sal  
       end  
revised_salary from emp;
```

2. DECODE

```
select ename, job, sal,  
       decode(job, 'ANALYST', 1.10*sal,  
                'CLERK', 1.15*sal,  
                'SALESMAN', 1.20*sal,  
                sal)  
revised_salary from emp;
```

使用 decode 计算个税

```
select ename, job, sal,  
       decode(sal, sal < 1500, 0.97*sal,  
                1500 < sal < 4500, 0.9*sal-105,  
                4500 < sal < 9000, 0.8*sal-555,  
                sal > 9000, 0.75*sal-1005,  
                sal)  
revised_salary from emp;
```

7. 连接查询（高级查询）

1. 相等连接

- 使用 “=” 连接

```
select emp.empno, emp.ename, emp.deptno,  
       dept.deptno, dept.dname, dept.loc  
from emp, dept  
where emp.deptno=dept.deptno;
```

2. 不相等连接

- 使用 between 连接
- 简化 emp 表示的方法：在where后用e代替emp

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

3. 外连接!!!

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

- 使用 “(+)” 连接 （在表中加入一个空行与没有直接匹配行的数据经行匹配）
- 完全外部连接

```
select ename, dname from emp full outer join dept on dept.deptno = emp.deptno;
```

4. 自连接

```
select w.ename "Worker", m.ename "Manager"  
from emp w, emp m  
where w.mgr = m.empno;
```

8. 递归查询!!!

- 关键词: Start with , Connect by prior

查询Jones的所有下属员工

```
SELECT empno, ename, level
FROM emp
START WITH ename = 'JONES'
CONNECT BY PRIOR empno = mgr;
```

查询Jones的上级员工时将上方SQL最后改为: mgr=empno

可以理解为: 查谁谁在前

9. 分组函数

1. avg , max , min , sum 在计算时会忽略NULL行; 但若数值全为NULL, 则计算结果为NULL

```
select avg(sal), max(sal), sum(sal)
from emp
where job like 'SALES%';
```

```
select avg(comm), max(comm), sum(comm)
from emp
where job like 'SALES%';
```

1. count

- count(*)所有行数

```
select count(*) from emp where job = 'SALESMAN';
```

- count(expr): 返回非NULL行数

2. group by

- 分一组

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

- 分多组

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

- 限制选择组 (使用 **having**)

```
select deptno, max(sal)
from emp
group by deptno
having max(sal)>1500;
```

在 Where 子句中**不能**直接使用组函数

10. 子查询

1. 单行子查询

```
select ename, job
from emp
where job =
    (select job
     from emp
     where empno = 7369)
and sal >
    (select sal
     from emp
     where empno = 7876);
```

总结：子查询select后的项目与操作符前的相同

```
select deptno, min(sal)
from emp
group by deptno
having min(sal) >
    (select min(sal)
     from emp
     where deptno = 20);
```


2. 多行子查询

- Any

```
select empno, ename, job, sal
from emp
where sal < any
  (select sal
   from emp
   where job = 'CLERK')
and job <> 'CLERK';
```

- All

```
select empno, ename, job, sal
from emp
where sal < all
  (select avg(sal)
   from emp);
```

- Exists

```
select e.empno, e.ename
from emp e
where exists
  (select 'X'
   from dept d
   where e.deptno = d.deptno
   and d.loc = 'NEW YORK');
```

- In

```
select e.empno, e.ename, d.deptno
from emp e
where e.deptno in
  (select d.deptno
   from dept d
   where d.loc = 'NEW YORK');
```

11. 嵌套查询

```
create or replace view dept_sum_vu
as select d.dname "部门名称", e.minsal "最低工资", e.maxsal "最高工资", e.avgsal "平均工资"
from dept d,
    (select deptno, min(sal) minsal, max(sal) maxsal, avg(sal) avgsal
     from emp
     group by deptno) e
where d.deptno = e.deptno;
```

六. PL/SQL

1. PL/SQL编程基础

1.1 符号

三种不等于: `<>` , `!=` , `^=`

赋值: `:=`

注释: `--` , `/* */`

PL/SQL块结构

```
declare --可选
    /* 变量, 游标, 用户定义异常 */
begin --必须
    /* SQL或PL/SQL */
exception --可选

end; --必须
```

PL/SQL块种类

- 匿名块: Declare... Begin... Exception... End;
- 储存过程: Procedure name is... Begin... Exception... End **name**;
- 函数: Function name... Return datatype is... Begin...return value; Exception... End **name**;

1.2 声明变量

- v=variable 变量
- c=constant 常量, 定义常量时必须同时为它赋值, 否则会出现错误
- boolean变量的三种值: True, False, Null; boolean一般和and, or, not一起使用
- 'YYYY/MM/DD HH24:MI:SS'

```
declare
  v_ename varchar2(10);
  v_sal number(6,2);
  c_tax_rate constant number(3,2) := 5.5;
  v_hiredate date;
  v_valid boolean not null default false;
```

1.3 引用

- %type: 定义简单变量

前缀为¹数据库的表.列名, ²先前声明的变量名

```
v_ename emp.ename%type;
v_balance number(7,2);
v_min_balace v_balance%type;
```

- %rowtype: 定义记录变量

record_variable [schema.]table_name%rowtype

1.4 转换员工姓名为小写

```
v_ename := lower(v_ename);
```

1.5 PL/SQL中的select

```
declare
    v_ename emp.ename%type;
    v_empno emp.empno%type;
begin
    select ename, empno
    into v_ename, v_empno
    from emp
    where empno = 9000;
    dbms_output.put_line(v_ename || ' ' || v_empno);
end;
```

```
DECLARE
    v_empRecord emp%ROWTYPE;
BEGIN
    -- 从emp表中检索一条记录并存储到v_empRecord记录变量中。
    SELECT *
    INTO v_empRecord
    FROM emp
    WHERE empno = 7369;
    dbms_output.put_line(v_empRecord.empno || v_empRecord.ename || v_empRecord.sal);
END;
```

```

declare
    v_sum_sal emp.sal%type;
    v_deptno number not null := 10;
begin
    select sum(sal)    --组函数
        into v_sum_sal
    from emp
    where deptno = v_deptno;
    dbms_output.put_line(v_sum_sal);
end;

```

1.6 PL/SQL中的DML

1. 插入

关键点: v_deptno := &no v_dname := '&name'

```

declare
    v_deptno dept.deptno%type;
    v_dname dept.dname%type;
begin
    v_deptno := &no;    --表示替代变量，当程序被执行时，系统会提示为其输入值
    v_dname := '&name';  --同上
    insert into dept (deptno, dname)
    values (v_deptno, v_dname);
end;

```

2. 更新

```

declare
    v_sal_increase emp.sal%type := 180;
begin
    update emp
    set sal = sal + v_sal_increase
    where job = 'SALESMAN';
end;

```

3. 删除

```

declare
    v_deptno emp.deptno%type := 10;
begin
    delete from emp
    where deptno = v_deptno;
end;

```

1.7 PL/SQL中的Commit & Rollback

```

declare
    v_sal number(10,2) := &salary;
    v_ename varchar2(20) := '&ename';
begin
    update emp
    set sal = v_sal
    where ename = v_ename;
    commit;
exception
    when others then
        rollback;
end;

```

1.8 if

```

declare
    v_sal emp.sal%type;
    v_ename emp.ename%type := '&ename';
begin
    select sal into v_sal from scott.emp
    where lower(ename) = lower(v_ename);
    if v_sal < 2000 then
        update emp
        set sal := v_sal + 200
        where lower(ename) = lower(v_ename);
    end if;
end;

```

注意处理null的方法

```
declare
  v_comm emp.comm%type;
  v_empno emp.empno%type := &no;
begin
  select nvl(comm, 0) into v_comm from scott.emp
  where empno = v_empno;
  if v_comm <> 0 then
    update emp
    set comm = v_comm + 200
    where empno = v_empno;
  else
    update emp
    set comm = 100
    where empno = v_empno;
  end if;
end;
```

思考: *select... into...* 含义?

1.9 loop/for

```
declare
  v_num number(2) := &num;
  v_pro number(20) := 1;
  i number(2) := 1;
begin
  if v_num = 0 then
    v_pro := 1;
  else
    loop
      v_pro := v_pro*i;
      i := i+1;
      exit when i>v_num;
    end loop;
  end if;
  dbms_output.put_line('num: '||v_num||' factorial: '||v_pro);
end;
```

- for循环中不声明计数器变量；计数器不能在loop循环体外部定义

```
declare
    v_num number(2) := &num;
    v_pro number(20) := 1;
begin
    if v_num = 0 then
        v_pro := 1;
    else
        for i in 1..v_num loop
            v_pro := v_pro*i;
        end loop;
    end if;
    dbms_output.put_line('num: '||v_num||' factorial'||v_pro);
end;
```

1.10 游标Cursor

- 定义的sql语句必须只包含select语句，且不能使用 insert , update , delete 等关键字
- 显示游标应用场景：select语句返回零或多于一行
- 隐式游标应用场景：select语句只返回一行

1. 声明游标 (不能包含into语句)

关键词：cursor... is...

```
declare
    cursor c1 is
        select empno, ename
        from emp;
    cursor c2 is
        select *
        from dept
        where deptno = 10;
    ...
```

2. 从游标中提取数据

- 反复使用fetch来提取每一行数据


```

declare
    cursor c1 is
        select * from emp;
        emp_rec emp%rowtype;  --定义一个和表结构完全一致的记录变量
begin
    open c1;
    fetch c1 into emp_rec;
    dbms_output.put_line('姓名是: '||emp_rec.ename||' 工作是: '||emp_rec.job||' 工资
是: '||emp_rec.sal);
    close c1;
end;

```

3. 游标提取basic loop

```

declare
    cursor emp_cursor is
        select ename, sal
        from emp
        where deptno=10;
    emp_record emp%rowtype;
begin
    open emp_cursor;
    loop
        fetch emp_cursor into emp_record.ename, emp_record.sal;
        exit when emp_cursor%notfound;
        dbms_output.put_line('ename: '||emp_record.ename||' sal:'||emp_record.sal);
    end loop;
    dbms_output.put_line('row count: '||emp_cursor%rowcount);
    close emp_cursor;
end;

```

4. 游标提取while loop

```

declare
    cursor emp_c is
        select ename, sal
        from emp
        where deptno=10;
    emp_rec emp%rowtype;
begin
    open emp_c;
    fetch emp_c into emp_rec.ename, emp_rec.sal;
    while emp_c%found loop

```

```

        dbms_output.put_line('ename: '||emp_rec.ename||' sal: '||emp_rec.sal);
    end loop;
    dbms_output.put_line('row count: '||emp_c%rowcount);
    close emp_c;
end;

```

5. 游标提取for loop

- for前系统**自动open**游标，for中系统**自动fetch**游标，for后系统**自动close**游标
- for循环中的循环控制变量不需要事先定义

```

declare
    cursor emp_c is
        select ename, sal
        from emp
        where deptno=10;
begin
    for emp_rec in emp_c loop
        dbms_output.put_line('ename: '||emp_rec.ename||' sal: '||emp_rec.sal);
    end loop;
end;

```

```

declare
    cursor c1 is
        select *
        from emp
        order by sal desc;
begin
    for rec in c1 loop
        if c1%rowcount <= 5 then
            dbms_output.put_line('ename: '||rec.ename||' sal: '||rec.sal);
        else
            exit;
        end if;
    end loop;
end;

```

2. PL/SQL程序设计

2.1 触发器

```
create or replace trigger secure_emp
before insert on emp
begin
    if(to_char(sysdate, 'DY')in('星期六','星期天'))
        or(to_char(sysdate, 'HH24:MI')not between '08:00' and '18:00')
    then
        if deleting then
            raise_application_error(-20502, 'You may delete from emp table only during
business hours.');
```

```
        elsif inserting then
            raise_application_error(-20500, 'You may insert into emp table only during
business hours.');
```

```
        elsif updating('SAL') then
            raise_application_error(-20503, 'You may update sal only during business
hours.');
```

```
        else
            raise_application_error(-20504, 'You may update emp table only during normal
hours.');
```

```
        end if;
    end if;
end
```

- 行级触发器：使用 :old 和 :new

2.2 过程函数

1. 过程与匿名块相比

- 无**declare**关键字
- 再end后可以加过程名，作为定义结束的标志
- 定义完成后需要**调用**才能执行过程内部代码

```
create or replace procedure display_time
is
begin
    dbms_output.put_line(systimestamp);
end display_time;
```

调用: **call** display_time();

```
create or replace procedure query_emp
(p_id in emp.empno%type,
 p_name out emp.ename%type,
 p_salary out emp.sal%type,
 p_comm out emp.comm%type)
is
begin
    select ename, sal, comm
    into p_name, p_salary, p_comm
    from emp
    where empno = p_id;
end query_emp;
```

```
create or replace function select_sal_name
(v_no in scott.emp.empno%type,
 v_sal out scott.emp.sal%type,
 v_name out scott.emp.ename%type
)
return number
is
    v_result number;
begin
    select sal, ename into v_sal, v_name
    from emp
    where empno = v_no;
    return v_result;
exception
    when NO_DATA_FOUND then
        dbms_output.put_line('无符合要求的记录');
        v_result := 0;
        v_name := '';
end;
```

那么...

与君歌一曲，请君为我倾耳听~

我的其它相关文章

- [Python期末考试总复习资料](#)
- [Linux服务器\(Ubuntu 18.04\)安装JDK、Hadoop、Hbase以及图形界面的全过程](#)
- [嵌入式期末复习重点](#)
- [《软件测试与质量保证》期末复习重点](#)
- [离散数学期末复习笔记【精华版】](#)