超管登录：sqlplus / as sys(用户名)dba(角色)

解锁用户:alter user scott account unlock;

设置密码:alter user scott identified by tiger;

解锁用户并设置密码:alter user scott identified by tiger account unlock;

显示表结构: desc 表名;

普通用户登录：sqlplus scott(用户名)/tiger(密码)

显示当前用户:show user

更改密码：password

退出sqlplus:exit

打开实时监控时间

set time on;

set time off;

打开sql每执行一次命令时间

set timing on;

set timing off;

sqlplus默认不显示程序执行结果，set serveroutput on/off

设置显示列宽:column 列名 format 9;(表示数字型宽度，一个9一个宽度)

column 列名 format a12;(忽略大小写，表示字符型宽度，12代表12个宽度)

设置一页最多显示80条数据记录:set pagesize 80;

执行最近一次sql语句：/

清屏(sqlplus工具)：host cls；

结果为null，sqlplus工具不显示，null与数值进行运算，结果为null。

解决null的问题：使用NVL(a,b)，a为null，b替代。反之亦然。

列别名：as 别名。（as忽略大小写，不加双引号的别名中间不能空格。添加双引号则可以。不能添加单引号，

因为oracle单引号代表字符串或日期）

dual：表示哑表或者伪表

||符号：连接字符串

使用哑表和||输出helloword，oracle中from必须写。select 'HELLO'||'WORLD' from dual;

显示本地时间：select sysdate from dual;只显示日期，不显示时间，默认显示格式为：30-3月-16

ename的薪水为：salary美元

select ename || '的薪水为：' || salary || '美元' from tablename;

使用sqlplus工具命令，保存sql语句到硬盘文件e：/oracle.sql

spool e:/oracle.sql;

写入e：/oracle.sql;

spool off;

使用命令，读取sql文件到orcl实例中，并执行。

@e：/oracle.sql;

注释://单行注释 , /\*多行注释\*/

查看表结构：

desc 表名

## SQL语句的特点

1）是SQL92/99的ANSI官方标准，只要按照该标准来写，在任何的关系型数据库中都可以直接执行

2）SQL语句的关健字不能简写，例如：select，where，from

3）大小写不敏感，提倡大写

4）能够对表数据进行增删改查操作

5）必须以分号结束

6）通常称做语句

SQLPLUS命令的特点

1）是oracle自带的一款工具，在该工具中执行的命令叫SQLPLUS命令

2）SQLPLUS工具的命令中的关健字可以简写，也可以不简写，例如：col ename for a10;

3）大小写不敏感，提倡大写

4）不能够对表数据进行增删改查操作，只能完成显示格式控制，例如：设置显示列宽，清屏，记录执行结果

5）可以不用分号结束，也可以用分号结束，个人提倡不管SQL或SQLPLUS，都以分号结束

6）通常称做命令，是SQLPLUS工具中的命令

注意：SQLPLUS命令是SQLPLUS工具中特有的语句

单引号出现地方：

1:字符型：'hello'||'world'

2:日期型：'30-3月-16'

3:to\_char/to date(日期,'yyyy-mm-dd hh12:mi:ss')

双引号出现地方：

1:列别名

2:to\_char//to date(日期,'yyyy"年"mm"月"dd"日"hh12:mi:ss')

select \* from table\_Name where ename='大小写敏感'

！=和<>意思一样

对于日期和数值型，where时，小数值在前，大数值在后。

模糊查询

like '%n':查询以n结尾

like 'n%':查询以n开始

like 'm%n':查询第一个字母为m，最后一个字母为n

like '\_n\_\_'：查询长度是4个字符，且第二个为n的员工，一个\_表示一个字符

查询table所有信息。

select \* from table;

select \* from table where column\_Name like '%';

select \* from table where column\_Name like '%\_%';

查询员工姓名中含有'\_'的员工,'%\\_%' escape '\':让\(可任意符号)后面的字符恢复本来意思

select \* from table where ename like '%\\_%' escape '\';

插入姓名为‘的员工,2个单引号表示一个单引号

insert into table(name) values ('''')

使用order by语句时null看成最大值

列是数值型，过滤条件也要是数值型，字符串里面是数值可隐式转换。

列是字符型，过滤条件也要字符型。

## 单行函数

lower(全小写)/upper(全大写)/initcap(首字母大写)

select upper('HeLlo') from dual;

select lower('HeLlo') from dual;

select initcap('heLlo.cOm') from dual;

concat(连接字符，只能连接2个)/substr(截取字符)

select concat('hello','world') from dual;

select substr('helloWorld',6,5)from dual;//从第6位开始，取5位字符，中英文统一处理。

length/lengthb,测试字符(字节)长度。

select length('你好World') from dual;//7,字符长度。

select lengthb('你好World') from dual;//11，字节长度，与编码有关系，gbk：2字节/utf-8：3字节

instr/lpad/rpad

select instr('hellooo','o')from dual;查询o第一次首次出现位置。找不到返回0，大小写敏感。

select lpad('hello',10,'\*')from dual;//长度不足10位，从左到右用\*补到10位,rpad则从右边补。

select rpad('hello',3,'\*')from dual;//长度不足补，足则从左截取到3位。

trim/replace

select trim(' ' from ' he llo ') from dual;//只去掉左右2边指定的，中间不去掉。

select replace('hello','l','L') from dual;//全部替换为LL，没有找到相同的则原样输出.

round/trunc/mod

作用数值:

select round(3.145,2) from dual;//截取2位小数，四舍五入。

select trunc(3.145,2) from dual;//截取2位小数。

select mod (10,3) from dual;//返回取模的值

作用日期:

round//1-6上半年，7-12下半年。1-15上半月，16-月末下半月。

2016-7-14

舍年，看月。属于下半年，2017-1-01

舍月，看日。属于上半月,2016-7-01

2016-3-30

属于上半年，2016-1-01

属于下半月,2016-4-01

select round(sysdate,'year')from dual;

select round(sysdate,'month')from dual;

trunc//

2016-7-14

舍年，看月。更改为当前年第一天。

舍月，看日。更改为当前月第一天。

select trunc(sysdate,'year')from dual;

select trunc(sysdate,'month')from dual;

显示昨天今天明天

select sysdate-1 "昨天",sysdate "今天",sysdate+1 "明天" from dual;

日期也能进行运算

select (sysdate - sysdate+1) from dual;

months\_between:精确运算相差多少天,注意months\_between(a,b)表示b到a相差多少天。a>b则正数，反之负数。

select months\_between('31-12月-16',sysdate)from dual;//今天距离今年最后一天相差多少天。

add\_months:表示几月前/后今天多少号,2月无30号。

select add\_months(sysdate,-1)from dual;

next\_day:从今天起，下个星期几是多少号(支持中文平台)

select next\_day(next\_day(sysdate,'星期三') ,'星期日')from dual;

last\_day:本月最后一天是多少号

select last\_day(sysdate)from dual;

日期-日期=天数

日期+-天数=日期

## 三大类型转换

字符串(char/varchar)，日期(date)，数值(number)

除了数值不能转换日期，其他都能相互转换。

oracle如何隐式转换

1：=号2边类型是否相同

2：如果=号2边类型不相同，尝试转换。

3：转换时，确保合理，否则失败。如12月不会有37天，一年不会有13月。

日期->字符串:to\_char('日期','格式')year(年的英文全称)/month(月全称,几月)/dd和day代表星期

select to\_char(sysdate,'yyyy "年" mm "月" dd "日" day')from dual;//2016 年 03 月 30 日 星期三

select to\_char(sysdate,'yyyy-mm-dd "今天是"day hh24:mi:ss AM')from dual;//2016-03-30 今天是星期三 18:32:31,hh表示24/12进制

数值->字符串:to\_char('数值','格式')

9:0-9任意值

0:0

$:美元符号

L:本地货币符号

.:小数点

,:千位符

select to\_char(100,'999') from dual;

select to\_char(1.00,'L9.99') from dual;//￥1.00

select to\_char(100,'$9,99') from dual;// $1,00

字符串->日期:to\_date('字符串','格式')

select \* from table\_Name where column\_Name=to\_date('1996年12月2日','yyyy"年"mm"月"dd"日"');

select \* from table\_Name where column\_Name=to\_date('1996-12-02','yyyy-mm-dd');

select to\_date('1996-12-2','yyyy-mm-dd') from dual;

select to\_date('1996年12月2日','yyyy"年"mm"月"dd"日"') from dual;

select to\_date('1996\*12\*2\*','yyyy"\*"mm"\*"dd"\*"') from dual;

select to\_date('1996\*12\*2\*','yyyy\*mm\*dd\*') from dual;//添不添加双引号不影响

字符串->数值:to\_number('字符串');

select to\_number('120')from dual;

面试题：

select 132+'123' from dual;//255,四则运算

select 123||'132'from dual;//123132,字符串连接

## 通用函数

NVL(a,b):a为null取b值，否则a值。

NVL2(a,b,c):a不为null取b，否则取c

NULLIF(a,b):a=b返回null，否则a，大小写和数据类型敏感。

coalesce(expr1,expr2)：返回该函数中第一个不为null的表达式的值

流程控制

sql99:case语句可看做一个字段

case 字段

when 条件1 then 表达式1

when 条件2 then 表达式2

else 表达式3

end (可放列名)

select

case job

when 'BOSS' then salary+1000

when 'TFBOSS' then salary+8000

else sal+400;

end "涨后工资"

from table\_Name;

oracle特有条件运算:decode，判断job是否是boss，是返回salary+1000，不是，判断是否tfboos...都不相等，默认返回default

select decode(job, 'BOSS' ,salary+1000, 'TFBOSS' ,salary+8000, sal+400)"涨后工资" from table\_Name;

## 多行函数

单行函数：输入一个参数，输出一个结果。如：upper

多行函数：输入多个参数或内部扫面多次，输出一个结果。如：count，avg，sum

多行函数不统计null值

max/min可比较日期

count/avg/sum/max/min/group by having

函数：oracle服务器事先写好的一段具有一定功能的程序片段，内置于oracle服务器，供用户调用。

统计emp员工表总人数

select count(\*)from emp;

\*号适合用于表字段较少的情况，如果字段较多，扫描时间长，效率低。推荐某一个非null唯一的字段，通常主键。

统计不重复的部门

select count(distinct deptno)from emp;

having和where区别

where

1:行过滤器

2：针对原始的记录

3：跟在from后面

4：where可省

5：先执行

having

1：组过滤器

2：针对分组后的记录

3：跟在group by后面

4：可省

5:后执行

oracle综合语法

1：select子句必须

2：from必须

3：where可选

4：group by可选

5：having可选

6：order by可选

group by细节：

1：在selece子句中出现的非多行函数的所有列，必须出现在group by后面。

2：在group by子句中出现的所有列，可不出现在select子句中。

## 多表查询

笛卡尔积：多表查询理论基础

多张表

内连接(只能查询符合条件的记录):

1:等值连接,只能用=号

2：非等值连接,非=号其他符号

外连接(能查出符合条件的记录，也能根据一方强行将另一方查询出来):左/右外连接

单张表:自连接

内连接

等值连接(=):select a.\* ,b.\* from a.deptno=b.deptno;

非等值连接(除=外所有符号，>=,<=,<>,!=,betwen,and等):

select a.\*,b.\* from a.salary betwen b.区间1 and b.区间2;

外连接

+出现在=号左边，右外连接，反之亦然。本质一样，只是+号位置不同，且+只能出现一次。oracle专用

+放在少的一边。

按部门号查出部门总人数。

select dept.deptNo "部门号", deptno.dname "部门名",count(emp.empNo) "人数"

from dept,emp

where dept.deptNo=emp.deptNo(+)

group by dept.deptno,deptno.dname;

自连接,也用到内连接+外连接

select user.name,boss.name

from emp user,emp boss

where user.mgr=boss.empno;

## 子查询

子查询：查询条件未知的事物

子查询别名不能添加引号

子查询细节：

1）子查询与父查询可以针对同一张表

2）子查询与父查询可以针对不同张表

3) 子查询与父查询在传统参数时，数量要相同

4) 子查询与父查询在传统参数时，类型要相同

5) 子查询与父查询在传统参数时，含义要相同

单行子查询：子查询返回一个结果，父查询用= < <= >= 来比较

多行子查询：子查询返回多个结果，父查询用any/in/all

select \* from emp where emp.id in(10,20);

select \* from emp where emp.salary <any(100,2000,100,200);//<集合最大值

select \* from emp where emp.salary <all(100,2000,100,200);//<集合最小值

多表查询(索引帮助)>子查询>集合查询

## 集合查询

union:2集合中有相同取其一

union all:2集合中有相同取全部

intersect:交集，输出2集合相交的记录

minus:差集,减

注意

1：确保列数相等

2：类型相等

3：a union b union c=c union b union a;

4:多个集合操作时，别名由第一个集合列名确定。

## 分页

什么是rownum，有何特点

1）rownum是oracle专用的关健字

2）rownum与表在一起，表亡它亡,表在它在

3）rownum在默认情况下，从表中是查不出来的

4）只有在select子句中，明确写出rownum才能显示出来

5）rownum是number类型，且唯一连续

6）rownum最小值是1，最大值与你的记录条数相同

7）rownum也能参与关系运算

\* rownum = 1 有值

\* rownum < 5 有值

\* rownum <=5 有值

\* rownum > 2 无值

\* rownum >=2 无值

\* rownum <>2 有值 与 rownum < 2 相同

\* rownum = 2 无值

8）基于rownum的特性，我们通常rownum只用于<或<=关系运算

显示emp表3-8条记录

method1:select rownum ,emp.\* from emp where rownum<=8

minus

select rownum, emp.\* from emp where rownum<=2;

method2(不能用rownum):select xx.\*

from (

select rownum id, emp.\* from emp where rownum<=8

) xx

where id>2;

## 创建表和约束

create table users(

id number(5) primary key,

name varchar2(8) not null unique,

sal number(6,2) not null,

birthday date default sysdate

);

查询该用户下有多少表：select \* from tab;

进入回收站：drop table 表名;添加cascade constraints级联删除

彻底删除:drop table 表名 purge;

查询回收站中的对象

show recyclebin;

清除回收站

purge recyclebin;

闪回，回收站还原,也可以根据回收站名还原// "BIN$bdkOnSQKS9yndzNUuvTJOA==$0" 要加双引号

flashback table 表名 to before drop;

flashback table 表名 to before drop rename to 新表名

number(5) :表示最多存99999

number(6,2):小数+整数不<=6，最多显示2位小数，采用四舍五入，不足补0，同时要设置col column format 9999.99；

varchar2(8) :8字节，跟编码有关。

date:默认格式:'31-3月-16'

clob(character large object):大文本对象，超过65565字节的数据对象，最多4G

blob(binary large object)： 大二进制对象，图片，mp3,MP4，最多4G

alter table 表名

add 列名 类型(宽度) //添加列

modify 列名 类型(宽度) not null //修改列类型宽度or约束

drop column 列名 //删除列

rename column 原列名 to 新列名 //重命名列名

rename 原表名 to 新表名 //更改表名

表很大时，删除列非常耗时，针对这种，设置为无用列。一旦设置无法恢复

并没有删除该列，只是让用户无法查看该列数据。可以空闲时删除

Alter table tablename set unused colunmn columnname cascade constraints;

删除:Alter table tablename Drop unused columns;

面试题：1000亿数据，高效将薪水字段清0，其他字段不变

//删除列

alter table table\_Name

drop column salary;

//添加列，默认值为0

alter table table\_Name

add salary number(6) default 0;

--级联,约束

create table curstoms(

id number(3) primary key,

namr varchar2(8) not null unique

);

create table orders(

id number(3) primary key,

cid number(3),

salary number(6) not null check (salary between 6000 and 8000),//薪水介于6000-8000

gender varchar2(2) not null check(gender in ('男' ,女'')),//性别男或女

--constraint cid\_CK foreign key(cid) references cursoms(id) on delete cascade,//删除curtoms表时，本表记录清空

--constraint cid\_CK foreign key(cid) references cursoms(id) on delete cascade set null //删除curtoms表时，本表记录null

);

## 事务，视图，索引，同义词

&占位符，范围DML语句，&是sqlpuls工具中提供的占位符。

可在value,from ,select ,where , group by ,having 使用

插入数据时，在字符串和日期中要添加''符号。

select &number from &table where salary >&sal

批量插入(表2数据复制到表1中):insert into 表1 select \* from 表2。

复制表1结构：create table 新表

as

select \* from 表1 where 1<>1;

复制表1结构+数据(where不写默认true):create table 新表

as

select \* from 表1 where 1=1;

截断表:truncate table 表名

drop table ||truncate table

1:属于ddl

2:不可回滚

3：不可where

4：前者删除表内容+结构，后者删除表内容

5：前者速度快，后者次之。

delete from

1:属于dml

2：可回滚

3：可带where

4：结构存在，表内容要看where执行情况

5：删除速度慢，逐条删除。

## 事务

oracle的事务只针对dml操作select/insert/delete/update

oracle事务开启：第一条dml操作为事务开启

提交事务(提交的是事务开始到提交中间的内容)：

1:显示提交，commit

2：隐式提交：ddl/dcl/exit(sqlplus工具)

回滚(回滚到事务开始的地方)：

显示回滚：rollback

隐式回滚：关闭窗口(sqlplus工具)，死机，掉电

回滚点：savapoint a;

回滚到a:rollback to savapoint a;

commit清空回滚点。

Oracle之所以能回滚的原因是？

主要机制是实例池

回顾MySQL支持的四种事务隔离级别及能够解决的问题

（1）read uncommitted -- 不能解决任何缺点

（2）read committed -- 脏读，Oracle默认,只能读到对方提交的数据

（3）reapatable read -- 不可重复读，脏读，MySQL默认

（4）serializable -- 幻读，不可重复读，脏读，效率低

脏读，读取到了对方没提交的数据

不可重复读：读取2次内容不一样

幻读：读取2次数目不一样

注意：jdbc/dbutils速度快，但书写烦

mybaits速度中等，但书写"中等"

hibernate速度慢，但书写"爽"

Oracle支持的二种事务隔离级别及能够解决的问题

Oracle支持：read committed 和 serializable

Oracle中设置事务隔离级别为serializable

set transaction isolation level serializable;

演示二个用户同时操作emp表，删除KING这条记录，会有什么后果？

因为有隔离级别的存在，所以不会出现二个用户都删除了KING这条记录，

一定是一个用户删除KING成功,在该用户没有提交的情况下，另一个用户等待

## 访问其他用户下的对象

scott或hr：用户名/方案名/空间名

查询scott表空间下所有对象,scott可加可不加

select \* from scott.表名;

以scott身份查询hr表空间的所有表时，需添加用户名

select \* from hr.表名;

默认情况下，每个用户只能查询自己空间下的对象，想插叙其他用户的表，需要授权

以sysdba角色，授予scott用户查看所有用户空间下的对象权限。

grant select any table to scott;

撤销权限

revoke select any table from scott;

查看自己拥有权限

select \* from user\_sys\_privs;

从scott用户空间导航到hr

conn hr/lion

## 视图

什么是视图【View】

（1）视图是一种虚表

（2）视图建立在已有表的基础上, 视图赖以建立的这些表称为基表

（3）向视图提供数据内容的语句为 SELECT 语句,可以将视图理解为存储起来的 SELECT 语句

（4）视图向用户提供基表数据的另一种表现形式

（5）视图没有存储真正的数据，真正的数据还是存储在基表中

（6）程序员虽然操作的是视图，但最终视图还会转成操作基表

（7）一个基表可以有0个或多个视图

什么情况下会用到视图

（1）如果你不想让用户看到所有数据（字段，记录），只想让用户看到某些的数据时，此时可以使用视图

（2）当你需要减化SQL查询语句的编写时，可以使用视图，但不提高查询效率

视图应用领域

（1）银行，电信，金属，证券军事等不便让用户知道所有数据的项目中

视图的作用

（1）限制数据访问

（2）简化复杂查询

（3）提供数据的相互独立

（4）同样的数据，可以有不同的显示方式

创建视图

create view 视图名

as

select 列1..列2 from 表名;

创建或者替换视图

create or replace view 视图名

as

select 列1..列2 from 表名;

默认情况下，没创建视图权限，需要授权

分配创建视图权限

grant create view to scott;

删除视图权限

revoke create view from scott;

视图设置只读

create or replace view 视图名

as

select 列1..列2 from 表名

with read only;

对视图进行更新语句操作也会对应更新到基表中。

删除视图不影响表

删除视图不进入回收站

删除基表会影响视图

闪回视图基表正常工作。

删除基表后，重建跟基表结构名字一样的表，视图不工作。

## 同义词

什么是同义词【Synonym】

（1）对一些比较长名字的对象(表，视图，索引，序列，。。。）做减化，用别名替代

同义词的作用

（1）缩短对象名字的长度

（2）方便访问其它用户的对象

授权

grant create synonym to scott;

回收

revoke create synonym from scott;

创建同义词

create synonym 同义词名 for 表名/视图/other

删除同义词

drop synonym 同义词;

删除同义词不影响基表

删除基表，不会影响同义词

删除基表，闪回后，同义词依旧可用。

## 序列

什么是序列【Sequence】

（1）类似于MySQL中的auto\_increment自动增长机制，但Oracle中无auto\_increment机制

（2）是oracle提供的一个产生唯一数值型值的机制

（3）通常用于表的主健值

（4）序列只能保证唯一，不能保证连续

声明：oracle中，只有rownum永远保持从1开始，且继续

（5）序列值，可放于内存，取之较快

题问：为什么oracle不直接用rownum做主健呢？

rownum=1这条记录不能永远唯一表示SMITH这个用户

但主键=1确可以永远唯一表示SMITH这个用户

为什么要用序列

（1）以前我们为主健设置值，需要人工设置值，容易出错

（2）以前每张表的主健值，是独立的，不能共享

创建序列

create sequence 序列名

start with 1//1开始

increment by 1 //步长

maxvalue 1|nomaxvalue //设置最大值

minvalue 1|nominvalue//设置最小值

cycle|nocycle //假设1开始，到100结束。开启则重新由1在循环到100

cache|nocache 20 //缓存

当大量语句发生请求，申请序列时，为了避免序列在运用层实现序列而引起的性能瓶颈。

Oracle序列允许将序列提前生成 cache x个先存入内存，在发生大量申请序列语句时，

可直接到运行最快的内存中去得到序列。但cache个数也不能设置太大，因为在数据库重启时，

会清空内存信息，预存在内存中的序列会丢失，当数据库再次启动后，

序列从上次内存中最大的序列号+1 开始存入cache x个

删除序列

drop sequence 序列名;

获取下一个系列值,第一次使用序列，必须用nextval

select 序列名.nextval from dual;

获取当前序列值

select 序列名.currval from dual;

修改序列步长为20

alter sequence 序列名 increment by20;

创建序列后，无法更改序列启动值

删除表或序列，相互不影响。

inisert into (列1) values (sequence.currrval);

## 索引

select rowid "索引" 表名.\* from 表名

索引表，包括索引列+rowid

索引只提高查询速度，与insert/delete/update无关

什么是索引【Index】

（1）是一种快速查询表中内容的机制，类似于新华字典的目录

（2）运用在表中某个/些字段上，但存储时，独立于表之外

为什么要用索引

（1）通过指针加速Oracle服务器的查询速度

（2）通过rowid快速定位数据的方法，减少磁盘I/O

rowid是oracle中唯一确定每张表不同记录的唯一身份证

rowid的特点

（1）位于每个表中，但表面上看不见，例如：desc emp是看不见的

（2）只有在select中，显示写出rowid，方可看见

（3）它与每个表绑定在一起，表亡，该表的rowid亡，二张表rownum可以相同，但rowid必须是唯一的

（4）rowid是18位大小写加数字混杂体，唯一表代该条记录在DBF文件中的位置

（5）rowid可以参与=/like比较时，用''单引号将rowid的值包起来，且区分大小写

（6）rowid是联系表与DBF文件的桥梁

索引的特点

（1）索引一旦建立, Oracle管理系统会对其进行自动维护, 而且由Oracle管理系统决定何时使用索引

（2）用户不用在查询语句中指定使用哪个索引

（3）在定义primary key或unique约束后系统自动在相应的列上创建索引

（4）用户也能按自己的需求，对指定单个字段或多个字段，添加索引

什么时候【要】创建索引

（1）表经常进行 SELECT 操作

（2）表很大(记录超多)，记录内容分布范围很广

（3）列名经常在 WHERE 子句或连接条件中出现

注意：符合上述某一条要求，都可创建索引，创建索引是一个优化问题，同样也是一个策略问题

什么时候【不要】创建索引

（1）表经常进行 INSERT/UPDATE/DELETE 操作

（2）表很小(记录超少)

（3）列名不经常作为连接条件或出现在 WHERE 子句中

同上注意

创建索引

单列索引：create index 索引名 on 表(列名);

多列索引/联合索引:create index 索引名 on 表(列1，列2);查找的包括列1使用索引

删除索引

drop index 索引名;

## PLSQL

SQL99是什么

（1）是操作所有关系型数据库的规则

（2）是第四代语言

（3）是一种结构化查询语言

（4）只需发出合法合理的命令，就有对应的结果显示

SQL的特点

（1）交互性强，非过程化

（2）数据库操纵能力强，只需发送命令，无需关注如何实现

（3）多表操作时，自动导航简单，例如：

select emp.empno,emp.sal,dept.dname

from emp,dept

where emp.deptno = dept.deptno

（4）容易调试，错误提示，直接了当

（5）SQL强调结果

PLSQL是什么

是专用于Oracle服务器，在SQL基础之上，添加了一些过程化控制语句，叫PLSQL

过程化包括有：类型定义，判断，循环，游标，异常或例外处理。。。

PLSQL强调过程

为什么要用PLSQL

因为SQL是第四代命令式语言，无法显示处理过程化的业务，所以得用一个过程化程序设计语言来弥补SQL的不足之处，

SQL和PLSQL不是替代关系，是弥补关系

PLSQL程序的完整组成结构如下：

[declare]

变量声明;

变量声明;

begin

DML/TCL操作;

DML/TCL操作;

[exception]

例外处理;

例外处理;

end;

/

注意：在PLSQL程序中，；号表示每条语句的结束，/表示整个PLSQL程序结束

书写PLSQL的工具有：

（1）SQLPLUS工具

（2）SQLDeveloper工具

（3）第三方工具（PLSQL & 其它）

PLSQL与SQL执行有什么不同：

（1）SQL是单条执行的

（2）PLSQL是整体执行的，不能单条执行，整个PLSQL结束用/，其中每条语句结束用；号

[declare]

声明变量，例外。

begin

dml和事务控制语句

[exception]

例外处理语句

end;

/ 结束plsql程序

输出helloworld字符串

begin

dbms\_output.put\_line('hello world');

end;

/

注意：

dbms\_output：输出对象

put\_line：上述对象的一个方法，输出一个字符串并换行

sql单条执行

plsql，整体执行，结束用/，每条语句结束用;符号

常量，变量说明

i number(2); //最多99

mysum nubmer(3):100;//最多999，并赋值100

msg varchar(2):='结果是';//2字节

pename emp.ename%type;//单列字段

emp\_re emp%rowtype;//表所有字段

求10+100的值

declare

mysum number(3) :=0;

tip varchar2(13) :='结果是：';

begin

mysum:=10+100;

dbms\_output.put\_line(tip||mysum);

end;

/

输出100号员工的姓名和薪水

declare

pname emp.ename%type;

salary emp.sal%type;

begin

select ename , sal into pname ,salary from emp where empno=100;

dbms\_output.put\_line('100号员工的'||pname||'的薪水是'||salary);

end;

/

输出100号员工的姓名和薪水

declare

emp\_re emp%rowtype;

begin

select \* init emp\_re from emp where empno=100;

dbms\_output.put\_line(emp\_re.ename||emp\_re.sal);

end;

/

当定义变量时，该变量的类型与表中某字段的类型相同时，用%type

当定义变量时，该变量与整个表结构完全相同时，可以使用row%type，可以通过变量名.字段名，取对应值

## PL判断

declare

pday varchar2(10);

begin

select to\_char(sysdate,'day')into pday from dual;

if(pday in('星期六','星期日')) then

dbms\_output.put\_line('周末');

elsif(pday in('星期四')) then

dbms\_output.put\_line('周四');

else

dbms\_output.put\_line('工作日');

end if;

end;

/

## PL循环

循环

method1:

declare

i number(2):=1;

begin

loop

--大于10退出

exit when i>10;

dbms\_output.put\_line(i);

--自增

i:=i+1;

end loop;

end;

/

method2:

declare

i number(2):=1;

begin

--小于11循环

while (i<11) --可省略条件括号

loop

dbms\_output.put\_line(i);

--自增

i:=i+1;

end loop;

end;

/

for不能修改步长

method3:

declare

i number(2):=1;

begin

for i in 20 .. 30

loop

dbms\_output.put\_line(i);

end loop;

end;

/

## PL游标

什么是光标/游标/cursor

类似于JDBC中的ResultSet对象的功能，从上向下依次获取每一记录的内容

使用无参游标，输出员工表姓名和工资

declare

--定义游标

cursor cemp is select ename,sal from emps;

vename emp.ename%type;

salary emp.sal%type;

begin

--开启游标

open cemp;

loop

--向下移动游标一次

fetch cemp into vename,salary;

//游标找不到记录，退出循环

exit when cemp%notfound;

dbms\_output.put\_line(vename||salary);

end loop

--关闭游标

close cemp;

end;

/

使用带参游标，输出10号部门员工姓名和工资

declare

cursor cemp(deNo emps%pdeNo) is select ename,sal from emps where pdeNo=deNo;

vename emp.ename%type;

salary emp.sal%type;

begin

open cemp(&deNo);

loop

--向下移动游标一次

fetch cemp into vename,salary;

//游标找不到记录，退出循环

exit when cemp%notfound;

dbms\_output.put\_line(vename||salary);

end loop

close cemp;

end;

/

使用无参光标cursor，真正给员工涨工资，ANALYST涨1000，MANAGER涨800，其它涨400，

要求显示编号，姓名，职位，薪水

declare

cursor cemp is select empno,ename,job,sal from emp;

pempno emp.empno%type;

pename emp.ename%type;

pjob emp.job%type;

psal emp.sal%type;

begin

open cemp;

loop

fetch cemp into pempno,pename,pjob,psal;

--循环退出条件一定要写

exit when cemp%notfound;

if pjob='ANALYST' then

update emp set sal = sal + 1000 where empno = pempno;

elsif pjob='MANAGER' then

update emp set sal = sal + 800 where empno = pempno;

else

update emp set sal = sal + 400 where empno = pempno;

end if;

end loop;

commit;

close cemp;

end;

/

## PL例外

例外好处：增加程序健壮性，容错性。

除0例外

declare

result number;--不写默认6位

begin

result:=1/0;

dbms\_output.put\_line(result);

exception

when zero\_divide then

dbms\_output.put\_line('除数不能为0');

--后面还可以接sql语句

end;

/

没找到数据例外

declare

pname varchar2(20)

begin

select ename into pname from emp where deptno=20;

dbms\_output.put\_line(pname);

exception

when no\_data\_fount then

dbms\_output.put\_line('没找到数据');

end;

/

自定义例外

declare

result number;--不写默认6位

no\_ZERO exception;--创建例外对象

begin

raise no\_ZERO;--抛出例外

dbms\_output.put\_line(result);

exception

when no\_ZERO then --捕获例外

dbms\_output.put\_line('除数不能为0');

end;

/

## 存储过程

什么是存储过程【procedure】？

为什么要用存储过程？

（1）PLSQL每次执行都要整体运行一遍，才有结果

（2）PLSQL不能将其封装起来，长期保存在oracle服务器中

（3）PLSQL不能被其它应用程序调用，例如：Java

存储过程与PLSQL是什么关系？

存储过程是plsql一个方面的应用，即plsql是存储过程的基础

创建无参存储过程

create or replace procedure hello

as

begin

dbms\_output.put\_line('hello');

end;

/

删除存储过程

drop procedure hello;

执行存储过程

exec hello;

plsql执行

begin

hello;

end;

/

创建有参存储过程,大小写不敏感，in可不添加，默认。

create or replace procedure hello(pnum in number)

as

begin

dbms\_output.put\_line(pnum);

end;

/

exec getResult(10);

创建带参有返回值的存储过程

create or replace procedure hello(pnum1 in number ,pnum2 in number , resu out number)

as

begin

resu:=pnum1+pnum2;

end;

/

----

declare

re number;

begin

hello(10,20,re);

dbms\_output.put\_line(re||'哈哈');

end;

/

exec适合过程无返回值

plsql适合过程有返回值，不限个数。

## 存储函数

必须带返回值，已经返回类型。

创建无参函数

create or replace function ger\_Result return number

as

begin

return 10;

end;

/

删除函数

drop function get\_Result;

调用函数

method1:

declare

re number;

begin

re:=ger\_Result();--括号可省略

dbms\_output.put\_line(re);

end;

/

method2:其他类似java程序

创建有参函数

create or replace function ger\_Result(num1 in number) return number

as

begin

return num1;

end;

/

------

declare

re number;

begin

re:=ger\_Result(10);

dbms\_output.put\_line(re);

end;

/

创建有参，多返回值函数

create or replace function ger\_Result(num1 in number,num2 in number,resu out number) return varchar2

as

te varchar2(9):='结果是';

begin

resu:=num1+num2;

return te;

end;

/

declare

re number;

tem varchar2(12);

begin

tem:=ger\_Result(10,20,re);

dbms\_output.put\_line(tem||re);

end;

/

## 过程函数适合场景

声明：适合不是强行要你使用，只是优先考虑

什么情况下【适合使用】存储过程？什么情况下【适合使用】存储函数？

【适合使用】存储过程：

【适合使用】存储函数：

什么情况【适合使用】过程函数，什么情况【适合使用】SQL？

【适合使用】过程函数：

》需要长期保存在数据库中

》需要被多个用户重复调用

》业务逻辑相同，只是参数不一样

》批操作大量数据，例如：批量插入很多数据

【适合使用】SQL：

》凡是上述反面，都可使用SQL

》对表，视图，序列，索引，等这些还是要用SQL

## 触发器

create or replace trigger 触发器名

before/after --操作前/操作后

insert/delete --语句级，拦截那个操作

update of 列名 --行级，拦截那个操作的列

on 表名

for each row --扫描每行

plsql块

创建触发器，对表增加，显示hello

create or replace trigger insertEmp

before

insert

on emp

begin

dbms\_output.put\_line('hello');

end;

/

删除触发器

drop trigger 触发器名

一次插入或删除多条时，也只会拦截一次

删除表后，显示hello

create or replace trigger deletetEmp

after

delete

on emp

begin

dbms\_output.put\_line('hello');

end;

/

星期一到星期五，且7-23点能向数据库插入数据，否则异常

create or replace trigger insertEmp

before

insert

on emp

delare

pday varchar2(10);

phour number(2);

begin

--获取星期

select to\_char(sysdate,'day')into pday from dual;

--获取时间

select to\_char(sysdate,'hh24')into phour from dual;

if pday in ('星期六'，'星期日') or hour not between 7 and 23 then

--抛出例外，参数1编号，参数2例外原因

raise\_application\_error('-2000000','非工作时间，添加失败');

end if;

end;

/

创建行级触发器，确保涨后工资>涨前工资

触发语句 ：old ：new

insert 所有字段都空null 将要插入的数据

update 更新以前该行的值 更新后该行的值

delete 删除以前该行的值 所有字段都空null

create or replace trigger checkSalary

after

update of sal

on emp

for each row

begin

if :new.sal<= :old.sal the

raise\_application\_error('-222200','工资不能越涨越低');

end if;

end;

/

表丢到回收站，闪回后，触发器依旧正常工作。

彻底删除表，触发器也删。新的同样名字表，不会继承原来的触发器。

笔试题3：有一个员工表empinfo结构如下

create table empinfo(

fempno varchar2(10) primary key,

fempname varchar2(20) not null,

fage number(2) not null,

fsalary number(10,2) not null

);

insert into empinfo(fempno,fempname,fage,fsalary) values('1','AA',30,7000);

insert into empinfo(fempno,fempname,fage,fsalary) values('2','BB',31,8000);

insert into empinfo(fempno,fempname,fage,fsalary) values('3','CC',32,9000);

insert into empinfo(fempno,fempname,fage,fsalary) values('4','DD',33,10000);

insert into empinfo(fempno,fempname,fage,fsalary) values('5','EE',34,11000);

insert into empinfo(fempno,fempname,fage,fsalary) values('6','FF',35,12000);

insert into empinfo(fempno,fempname,fage,fsalary) values('7','GG',36,13000);

insert into empinfo(fempno,fempname,fage,fsalary) values('8','FF',37,14000);

假如该表有大约1000万条记录，写一句最高效的SQL语句，计算以下4种人中每种员工的数量

第1种人：fsalary>9999 and fage>35

第2种人：fsalary>9999 and fage<35

第3种人：fsalary<9999 and fage>35

第4种人：fsalary<9999 and fage<35

提示：只用一条SQL搞定

\*/

select

sum(case when e.fsalary>9999 and e.fage>35 then 1 else 0 end) "第1种人",

sum(case when e.fsalary>9999 and e.fage<35 then 1 else 0 end) "第2种人",

sum(case when e.fsalary<9999 and e.fage>35 then 1 else 0 end) "第3种人",

sum(case when e.fsalary<9999 and e.fage<35 then 1 else 0 end) "第4种人"

from empinfo e;