Oracle数据库

1、什么是数据库

1.1数据库：数据的仓库

以前存放数据：内存、文件

内存： int num = 10 ; 问题：临时

文件： 解析/操作起来麻烦

-》永久存放，方便解析/管理 ->数据库

1.2、主流的关系型数据库：

oracle：产品免费，服务收费，强大稳定性 安全性

mysql ：MySQL AB开源， 2008被SUN收购， 2009年被Oracle收购； 版本：社 区Community免费,企业版收费 ；产品免费，服务收费

sql server:微软开发，强大的图形化工具，方便使用

db2：IBM，多个操作系统、多个硬件

1.3、oracle版本：

oracle8i/9i:internet ,开始走向网络

oracle10g/11g:grid，网格计算，提高访问速度，避免舍近取远的问题

oracle12c:cloud，云计算

各个版本对于学习、研发 没有区别，只是在 最终的部署、运维时差异较大

oracle11g：目前主流

1.4数据库服务器：将数据库部署在服务器上

oracle服务器

1.基于关系型的数据库（RDBMS）: oracle mysql sqlserver db2....

关系型->二维表

非关系型：

NoSQL:not only sql ,redis/mongodb ：基于key-value结构 person.name

2.组成结构

一个PGA对应于一个客户端：

两阶段提交：

pga->sga

sg-a数据库（此阶段，会将一些重复/冗余的工作 进行合并，从而减少数据访问次数）

select \*from 表名; 查询语句

2、数据库表结构

2.1基本概念：

实体：java中的类

记录：java的对象 (per(zs,23,170...)) ,行

字段：java的属性，列

表：同一个实体中，所有的记录、字段组合起来 就是一张表

2.2列的计算：

select empno,ename ,sal ,sal\*12 from emp;

2.3、SQL语法解析：

什么控制行？什么控制列？

什么控制列？select empno,ename from emp ;

什么控制行？where

字符串/字符 、日期 ： 单引号

大小写问题：

a.命令/关键字：不敏感（不区别）

b.数据：敏感（区分

2.4、运算符和连接符：

操作运算符： + - / \* %

关系运算符： > >= < <= = !=或<>

如果是null，必须用is ，或is not

逻辑运算福： or and not

select\* from emp where not (mgr = 7788 and job = 'CLERK')

select\* from emp where mgr = 7788 and job = 'CLERK';

where执行顺序：右->左

连接符： java: "hello"+"world"->"helloworld"

oracle:

concat ||

dual：oracle提供的 学习时使用 临时表：单行单列

select 'hello'||'world' from dual;

2.5、null的特殊处理:

is/is not

null的计算：

任何数字 和null结算，结果为null

需要对null进行处理：null->0

nvl:if

nvl(comm,0 )

nvl2:if...else

nvl2(comm,comm,0)

if(comm==null) return 0

else return comm

2.6、distinct：结果集去重：

select distinct deptno from emp

2.7、修改oracle默认的日期格式

默认：DD-MON-RR

修改：

alter session set NLS\_DATE\_FORMAT = 'yyyy-mm-dd' ;

alter session set NLS\_DATE\_FORMAT = 'DD-MON-RR' ;

---

2.8、范围查询： 数字/日期

between 小 and 大

>=小 and <=大

2.9、模糊查询：

like

配合通配符使用：\_ ：一个字符

% ：任意个字符

ename like ..

数字、日期：like

姓名中第二个字母是M的员工信息：

select \*from emp where ename like '\_M%' ;

姓名中包含M的员工信息：

select \*from emp where ename like '%M%' ;

姓名长度>6的员工信息： >6 >=7

select \*from emp where ename like '\_\_\_\_\_\_\_%' ;

姓名中包含下划线的

zhang\_san

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

not in 不能出现null：如果出现了null，结果为null

select \*from emp where deptno not in(10,20,30,null) ;

select \*from emp where deptno not in(10,20,30,null) ;

select \*from emp where deptno != 10 and deptno != 20 and deptno != 30 and deptno != null

select \*from emp where mgr not in

(select mgr from emp);

2.10、排序：

order by 字段名|表达式|序号

select \*from emp order by sal desc ;

select \*from emp order by sal asc ;默认

select \*from emp order by sal+10000 desc ;

null默认是最大值

多列排序 ：

sal, hirdate

3、数据库函数

3.1、函数分类：

单行函数：一次操作一行

多行函数 ：以此操作多行

3.2、单行函数

3.2.1字符函数 ： lower upper initcap

dual ：单行单列 ->单行

substr(str,begin,len) ：从1开始数

length字符数 / lengthb 字节数

英文/数字

如果中文/符号：

utf-8编码格式下： 一个汉字/符号 占3个字节

gbk： 一个汉字/符号 占2个字节

查看当前系统编码格式：

select \* from nls\_database\_parameters ;

insrt(str,substr)

lpad/rpad：填充

trim：去掉任意字符

replace

3.2.2、数值函数:

round(数字,n位数):四舍五入 ，保留n位小数

trunc(数字,n位数):舍尾，保留n位小数

mod()

3.2.3、日期函数：

sysdate：当前时间

格式化： 日期-》字符

to\_char(日期,格式)

日期+-数字 （默认是天）

日期 - 日期

计算员工工龄 ：入职日期 天 星期 月 年

select ename ,hiredate , (sysdate - hiredate) , (sysdate - hiredate)/7 , (sysdate - hiredate)/30, (sysdate - hiredate)/365 from emp;

months\_between(日期1,日期2) ： 日期1-日期2

add\_months(日期,月数)：

当前最大是第几天 last\_day

下一个星期n是哪一天next\_day

select next\_day(sysdate,'星期五') from dual ;

round

trunc

if

else if

else if.

...

else

3.2.4、通用函数 ：

a. nvl/nvl2

b. nullif(a,b) :a=b,null ,否则返回

c coalesce :从左往后 找到第一个不为null的值

d.条件判断函数

decode(字段,条件1，返回值1，条件2，返回2，....,最后表达式)

select ename,job , sal 涨前, decode(job, 'PRESIDENT',sal+1000,'MANAGER',sal+500,sal+300) 涨后 FROM EMP;

case表达式

select ename ,job ,sal 涨前,case job

when 'PRESIDENT' then sal+1000

when 'MANAGER' then sal+500

else sal + 300 end

涨后

from emp ;

case

when ... then

when ... then

else

end ;

3.2.5、转换函数：

隐式转换/显式转换

隐式转换:

字符-数字

数字-字符

字符-日期

日期-字符

显式转换：

字符->字数

select to\_number('￥123,456.7', 'L999,999.9') from dual ;

字数->字符

字符-日期

日期-字符

to\_char

3.3、多行函数：组函数、 聚合函数

count(\*) -> 19

求有几个部门

count() 自动排除null

max/min/avg/sum

分组：

各个部门总工资： 对部门分组 dept 10 20 30 40

select deptno,avg(sal) from emp 【group by deptno】 ;

select deptno,ename , avg(sal) from emp group by deptno ;

10 8000

20 6000

30 5000

40 9000

分组查询时，不在组函数(多行函数)中的列，必须在group by中。

先根据部门分组 ，再根据job分组

--各个部门中各个工作 的平均工资

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

对行筛选用where

对组进行筛选用Having

可以在Having使用多行函数count min avg

但是 不能在where中使用多行函数

3.4、多表连接查询：

3.4.1、交叉连接（笛卡尔积）:所有情况的组合 ，不推荐使用

select \* from emp ,dept ;

3.4.2、内连接 ：多张表通过 相同字段进行匹配，只显示匹配成功的数据

a.

select \* from emp e ,dept d

where e.deptno = d.deptno ;

b.

select \* from emp e

inner join dept d

on e.deptno = d.deptno

不等值连接（一般不用）

select \* from emp e ,dept d

where e.deptno <= d.deptno ;

3.4.3、外连接

左外连接：以左表为基准（左表数据全部显示），去匹配右表数据，如果匹配成功 则全部显示；匹配不成功，显示部分(无数据部分 用NULL填充)

a.（oracle独有）

select \* from emp e ,dept d

where e.deptno = d.deptno(+) ;

b

select \* from emp e

left outer join dept d

on e.deptno = d.deptno

右外连接

右外连接：以右表为基准（右表数据全部显示），去匹配左表数据，如果匹配成功 则全部显示；匹配不成功，显示部分(无数据部分 用NULL填充)

a.（oracle独有）

select \* from emp e ,dept d

where e.deptno(+) = d.deptno;

b

select \* from emp e

right outer join dept d

on e.deptno = d.deptno

全外连接 = 左外 + 右外连接 - 去重

自连接：将一张表 通过别名 “视为”不同的表

查询 员工姓名，以及 该员工的领导姓名

select e.ename ,b.ename from emp e,emp b

where e.mgr =b.empno;

//员工表的领导编号mgr = 领导表的 员工编号号empno

3.4.4、自连接

一个表内两个字段连接查询，相当于交叉连接

费性能： emp -> e,b

3.4.5、层次连接

优化自连接

层次连接：

select level ,empno, ename ,mgr from emp

connect by prior empno=mgr

start with mgr is null

order by level ;

---

3.5、子查询

比SCOTT工资高的 员工信息

1：scott的工资->5200

select SAL from emp where ename = 'SCOTT' ;

2.>5200的员工信息

select \*from emp where sal > ;

->合二为一

select \*from emp where sal > (select SAL from emp where ename = 'SCOTT' )

①子查询可以出现的位置：where、select、having、from ；不能写在group by 后面。

where

select： 单行列 （常量列）

select empno 第一列,ename 第二列,(select job from emp where empno =7369) 第三列 from emp ;

having:

查询最低工资比10号部门的最低工资高的部门编号。

分组

select deptno,min(sal) from emp

group by deptno

having min(sal) > ( select min(sal) from emp where deptno =10 );

from:相当于修改了表结构

select \* from emp;

select \* from ( select empno,ename, sal\*12 from emp) ;

②主查询和子查询 可以是，也可以不同同一张表

查询销售部的员工信息

1.现根据“销售部”查询 销售部的部门编号30

select deptno from dept where dname = 'DNAME' ;

2.根据部门编号30 查询员工信息

select \* from emp where deptno = (select deptno from dept where dname = 'SALES' );

③子查询可以使用 单行操作符（=,<），多行操作符(in)

查询工资比30号部门中 任意其中一个员工高的（存在） 员工信息

"只需要满足一个即可，存在一个就可以"->any

select \*from emp where sal > any(select sal from emp) ;

select \*from emp where sal > (select min(sal) from emp) ;

查询工资比30号部门中 全部员工高的（存在） 员工信息

“所有、全部”->all

select \*from emp where sal > all(select sal from emp) ;

select \*from emp where sal > (select max(sal) from emp) ;

多行操作符:

查询销售部，财务部的员工信息

select \* from emp where deptno in

(select deptno from dept where dname = 'SALES' or dname='ACCOUNTING');

any:只要有一个

all：全部

④子查询中的null ：子查询的结果中不要有NULL！！

select \*from emp where mgr in (7566,7698);

select \*from emp where mgr in (7566,7698,NULL);

in : = or = or

select \*from emp where mgr =7566 or mgr=7698 or mgr = NULL;

select \*from emp where mgr not in (7566,7698,NULL);

select \*from emp where mgr not in (7566,7698,NULL);

select \*from emp where mgr not in (7566,7698,NULL);

select \*from emp where mgr !=7566 and mgr!=7698 and mgr != NULL ;

NULL:自身特性： 如果!=NULL则无法查询出任何数据

is null

is not null

=null

!= null

查询 不是领导的员工信息（子查询时排除NULL）

不是领导：判断empno 是否存在于mgr中

select \* from emp

where empno not in (select mgr from emp where mgr is not null )

⑤一般不在子查询中排序，除非TOP-N问题（分页）

集合运算细节：

各个集合的列数、类型必须保持一致

select empno ,ename from emp

union

select deptno,job from emp;

--报表

查询总工资、各个部门的共工资、各个部门中各个工作的总工资

各个部门中各个工作：

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

10 sales 7000

10 dev 8000

20 xxx 6500

各个部门的共工资、

select deptno,sum(sal) from emp group by deptno ;

总工资

select sum(sal) from emp ;

NULL：常量列

select deptno,job ,sum(sal) from emp group by deptno,job

union

select deptno, NULL ,sum(sal) from emp group by deptno

union

select NULL,NULL ,sum(sal) from emp ;

增强group by：rollup()

select deptno,job ,sum(sal) from emp group by rollup( deptno,job) ;

group by rollup( a,b)相当于：

group by a,b

group by a,

group by null

---

4、数据库操作

4.1、SQL语句的类型：

DQL：数据查询语言select

DML：数据操作语言,insert delete update --》可以回退（可以进行事务操作）

DDL ：数据定义语言 create/drop /truncate /alter table

DCL:数据控制语言 grant, revoke

DML:

4.2、增加数据insert

insert into 表名(字段名1,字段名2,...,) values(字段值1,字段值2,...)

字段名和字段值一一对象：数据类型、个数、顺序

insert into emp(EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO)

values(2222,'zhangsan','MANAGER',7788,'19-9月 -88',9998,1000,10);

可以省略字段名：

1.values插入的数据 必须是 完整的字段，并且顺序和默认顺序一致

2.目前使用的SQL99标准，可以省略字段名（如果是SQL92则不能省略,MyCat）

insert into emp

values(3322,'LISI','MANAGER',7788,'19-9月 -88',9998,1000,10);

如果插入的数据不完整，可以协商 部分字段名

insert into emp(EMPNO,ENAME,JOB)

values(444,'zhan','MANAGER');

动态输入插入的值(&) Scanner input = new Scanner(System.in); input.next();

insert into emp(EMPNO,ENAME,JOB)

values(&empno,&xxx,&job);

如果是字符、日期： 仍然需要加' '

insert into emp(EMPNO,ENAME,&otherName)

values('5555','kkk',&otherValue);

--

4.3、批量插入数据

1.创建新表(批量插入之前不存在)

emp ->复制 mytab

create table mytab

as

select \*from emp;

create table mytab2

as

select empno,ename ,job from emp;

create table mytab3

as

select empno,ename ,job from emp

where sal < 6000;

还可以用于快速创建表结构：

mytab4 -> emp

create table mytab4

as

select \*from emp where 1=0 ;

2.在旧表中插入(已存在的表)

insert into mytab4(empno,ename,sal)

select empno,ename ,sal from emp;

3. begin ...end /

begin

insert into emp

values(1221,'LISI','MANAGER',7788,'19-9月 -88',9998,1000,10);

insert into emp

values(1223,'LISI','MANAGER',7788,'19-9月 -88',9998,1000,10);

end ;

4.4、海量数据： 数据泵 \ SQL Loader\ 外部表

4.5、delete、truncate和update

delete from 表名 ;

delete from emp where empno >7900;

1.加where

2. delete from 表名 ;

全表删除：

1. delete from emp ; 可以回退

truncate table emp ;不能回退

原因： DML：insert update delete ->可以回退

2.测试二者执行时间

打开执行时间：

set timing on/off

对于少量数据： delete 效率高 ，一行一行删除

对于海量数据：truncate效率高 ， a.drop table 丢弃整张表 ，b.重新创建表

3.delete支持闪回， truncate不支持闪回

4.delete不会释放空间 （换两个地方存储数据[undo空间]），trucante会

5.delete会产生碎片，trunate不会

如果碎片太多，需要整理碎片：a. alter table 表名 move ; b.导出导入

---

update/delete :where

修改update

update 表名 set 字段名1=字段名1 , 字段名2=字段名2,字段名3=字段名3... where ...

update emp set ename = 'x' ,job ='y' where empno>7900;

--DDL： create/drop/truncate/alter

创建表

create table mytab6

(

id number ,

name varchar(10),

age number

)

;

注意事项：

1.权限和空间问题

2.表名的规定：

a.必须以字母开头

b.表名只能包含： 大小写字母、数字、\_、$、#

c.长度 1-30个字符

d.不能与数据库中其他对象重名(表，视图、索引、触发器、存储过程....)

e.不能与 保留字重名

4.6、rownum/rowid：删除重复数据

rowid：根据插入的顺序 依次递增

rownum：逻辑伪列

rowid:物理伪列，18位：

前6位： 数据对象编号

依次往后数3位：数据文件编号

依次往后数6位：数据块编号

依次往后数3：行号

思路：

根据编号分组（将重复的数据 放到一组） ，然后在每组中只保留一个

保留：rowid最小的

delete from mystudent where rowid not in (select min(rowid) from mystudent group by stuno );

5、数据库约束和对象

5.1、约束

5.1.1、常见6个约束

检查约束（check） name > 4

唯一约束（Unique） id:1 2 3 4 null

主键约束（Primary key） 类似唯一约束(唯一)

外键约束（Foreign Key） 两张表 学生表 课程表(1 2 3)

非空约束(Not null) 不能为null

默认约束（Default） address:西安

主键和唯一的区别：

a.【主键不能为Null】，唯一 可以为null

b.主键可以是复合主键，也可以是单值主键（id）

c.一张表中 只能设置一次主键（复合主键），但唯一键可以设置多次

5.1.2、约束分类：

列级约束 表级约束

作用于一个列 作用于一个列/多列

列的后面 表的后面

可以有多个约束（空格） 逗号,

全部的6个 4个（主键 外键 唯一 检查）

列级约束:

create table student(

stuno number(3) primary key ,

stuname varchar2(10) not null unique ,

stuaddress varchar2(20) default '陕西西安' check(length(stuaddress)>2),

stubid number(3)

);

表级约束

create table student2(

stuno number(3) ,

stuname varchar2(10) ,

stuaddress varchar2(20) ,

stubid number(3),

constraint PK\_sno primary key(stuno) ,

constraint UQ\_sname\_subid unique(stuname,stubid),

constraint CK\_saddress check( length(stuAddress)>2)

);

注意事项：

a.报错：违反唯一约束条件 可能主键报错 也可能唯一约束报错

b.如果有多个约束，default必须放在第一位

c.check约束： 如何编写 ：和使用where完全相同

d.唯一约束： 可以是Null,但不适用于Null(可以有多个null)

5.1.3、外键

外键含义： A.a ->B.b字段， a中的数据 必须来自于b中。

如果删除父表中 外键所指向的列， 2个策略：级联删除|级联置空

级联删除：当删除父表中的数据时，子表 会跟着删除相对应的数据；

级联置空：当删除父表中的数据时，子表 会将 相对应的 那一字段的值设置为Null，其他字段不影响；

外键使用建议：

1.当父表中没有相对应数据时，不要向子表增加数据(如果sub表没有编号为2的课程，那么子表student不要去选择2号课程)

2.不要更改父表的数据，导致子表孤立

3.建议：在创建外键时 直接设置成 级联删除 或级联置空

4.删除表？ 先删除子表，再删除父表

给表加外键：

创建表的同时 增加外键

create table student3(

stuno number(3) ,

stuname varchar2(10) ,

stuaddress varchar2(20) ,

subid number(3) ,

constraint FK\_student3\_sub foreign key(subid) references sub(sid)

);

其中，解释： constraint FK\_student3\_sub foreign key(subid) references sub(sid) ;

直接创建外建时报错：此列列表的【唯一关键字或主键】不匹配 ，含义是：外键所指向的 字段 必须先是 主键 或者唯一约束的键。

定义外键 需要4个参数： 两个表名 ，两个字段名

级联删除

drop table student3;

create table student3(

stuno number(3) ,

stuname varchar2(10) ,

stuaddress varchar2(20) ,

subid number(3) ,

constraint FK\_student3\_sub foreign key(subid) references sub(sid) on delete cascade

);

--

级联置空

drop table student3;

create table student3(

stuno number(3) ,

stuname varchar2(10) ,

stuaddress varchar2(20) ,

subid number(3) ,

constraint FK\_student3\_sub foreign key(subid) references sub(sid) on delete set null

);

5.1.4、约束命名

规范：约束类型\_字段名

主键： PK\_stuno

检查约束： CK\_字段名

唯一约束： UQ\_字段名

非空约束： NN\_字段名

外键约束： FK\_子表\_父表

默认约束： 一般不需要命名

加约束名： constraint 约束名

create table student(

stuno number(3) constraint PK\_stuno primary key ,

stuname varchar2(10) constraint NN\_stuname not null constraint UQ\_stuname unique ,

stuaddress varchar2(20) default '陕西西安' constraint CK\_stuaddress check(length(stuaddress)>2),

stubid number(3)

);

注意事项：

约束名是 多个表公用的 (多个表中的约束 不能重名)

5.1.5、追加约束

在创建表示忘了加约束，后续可以追加约束

1.唯一、主键、检查、外键约束

alter table 表名 add constraint 约束名 约束类型I

create table student4(

stuno number(3) ,

stuname varchar2(10) ,

stuaddress varchar2(20) ,

subid number(3)

);

alter table student4 add constraint UQ\_stuaddress4 unique(stuaddress);

alter table student4 add constraint PK\_stuno4 primary key (stuno );

alter table student4 add constraint CK\_stuname4 check(length(stuname)>2);

alter table student4 add constraint FK\_student4\_sub foreign key(subid) references sub(sid);

不适用于 默认、非空

alter table student4 add constraint NN\_stuname not null(stuname);

alter table student4 add constraint DF\_stuname default 'hello' ;

2. 默认、非空

alter table 表名 motidy 字段名 constraint 约束名 约束类型

非空

alter table student4 modify stuname constraint NN\_stuname4 not null ;

默认（建议默认约束不起名字,不写constraint）

alter table student4 modify stuname default '没名字' ;

删除约束：

alter table 表名 drop constraint 约束名;

alter table student4 drop constraint UQ\_stuaddress4;

alter table student4 drop constraint PK\_stuno4;

alter table student4 drop constraint CK\_stuname4;

alter table student4 drop constraint FK\_student4\_sub;

alter table student4 drop constraint NN\_stuname4;

特殊情况：

默认约束（删除默认约束： 将默认约束置为null）

alter table student4 modify stuname default null ;

5.1.6、完整性约束

保证数据的正确性、相容性、防止数据冗余等。

域完整性： 列。数据类型，非空约束,检查约束，外键约束

实体完整性：行。主键约束、唯一约束

引用完整性：不同表之间。外键

自定义完整性：触发器(当执行换一个操作时，会自动触发另一个操作)。例如：自定义需求 学生的上学时间 必须在出生日期之后。

5.2、视图

属于数据库对象之一

表、视图、序列、索引、同义词、触发器

视图，是一个虚表。建立在表的基础之上

注意：一张表，或多张表

5.2.1、创建视图

create view 视图名

as

select ...语句

with check option ;

with check option :限制对视图操作时，必须满足where子句

drop view myempview ;

create view myempview

as

select empno ,ename ,deptno from emp where deptno =20 with check option;

b.强烈建议

with read only ;

视图只建议查看，不建议DML。因为对视图的操作，会影响原表。

好处：

a.简化查询

原始SQL:

select \*from X ,Y where X.x = Y.y ;

select \*from emp e ,dept d where e.deptno = d.deptno;

select d.deptno 部门编号,e.empno,e.ename,e.sal,e.comm ,d.dname from emp e ,dept d where e.deptno =d.deptno and d.deptno=20;

用视图封装以上的查询结果：

create view myempview

as

select d.deptno 部门编号,e.empno,e.ename,e.sal,e.comm ,d.dname from emp e ,dept d where e.deptno=d.deptno and d.deptno=20;

5.2.2、修改视图

单表 可以任意更新 ； 一次性更新多张表，则必须唯一；

update myempview set ename = 'hello' where empno=7934 ;

注意： 修改的视图(DML)，会影响原表。一般建议 对视图 仅仅进行查询操作，不要进行DML.

建议： 如果update操作涉及了多张表，则很容易出现 异常；因此建议，update只对单表进行操作

5.2.3、删除视图

drop view myempview ;

5.3、序列

序列：模拟自增

5.3.1、序列的属性

本质就是内存中的数组 （20）

[1,2,3,...,20] [21,22,...,40]

序列有2个属性：

nextval:下一个值

currval:当前值

5.3.2、创建序列

create sequence 序列名

increment by 步长

start with 起始值

maxvalue | nomaxvalue

minvalue | nominvalue

cycle | nocycle

cache n | no cache ;

实例：

create sequence myseq;

insert into person values(myseq.nextval, 'ls4');

insert into person values(myseq.nextval, 'ws4');

insert into person values(myseq.nextval, 'zs4');

create sequence myseq1

increment by 2

start with 1000

;

create sequence myseq2

increment by 2

start with 1

maxvalue 9

minvalue 1

cycle

cache 3 ;

[1 3 5] 7 9

cache元素的个数 <= 循环元素个数

5.3.3、修改序列

只对修改之后的 操作有效

alter sequence myseq

increment by 2;

5.3.4、删除序列

drop sequence 序列名 ;

drop sequence myseq2;

5.4、索引

5.4.1、创建索引

索引类型 默认B树索引（默认）、位图索引

什么时候 适合建立索引：

数据集中的列,经常在where中使用的列， 数据量大、数据集中的列：主 键列(empno,id)不集中，但是因为 会被频繁使用 ，因此也适合建索引

create index 索引名

create index myindex on emp(deptno) ;

主键默认 就是索引

create index myindex on emp(deptno ,sal ) ;

5.4.2、删除索引

drop index 索引名

drop index myindex ;

5.5、同义词

数据库对象（表 视图 索引...）起别名 （默认私有/专用）

5.5.1、管理员赋权限

授权：

grant xxx to 用户名;

revoke xxx from 用户名;

grant create synonym to scott ;

5.5.2、创建私有、公有同义词

两种同义词需要管理员赋2种权限

grant create synonym to scott ;

grant create public synonym to scott ;

=================

create synonym hremp2 for hr.employe

create public synonym hremp2 for hr.employe

1. 事务

6.1、事务的概念

概念：作为单个逻辑工作单元执行的一系列操作

四大特性:ACID

Atomicity原子性：要么都成功，要么都失败。

Consistency:一致性 :事务执行前后 ，总量保持一致

Isolation隔离性:各个事务并发执行时，彼此独立

Durability:持久性：持久化操作。

6.2、事务的生命周期

事务的生命周期：

（MySQL: 自动提交，自动将每一条DML语句直接commit ）

Oracle:手工提交

事务的开始标识： 第一条DML

事务的中间过程： 各种DML操作

结束：

a.提交

i.显示提交：commit

ii.隐式提交（自动提交）：正常退出exit（ctrl+c）、DCL(grant ....to..., revoke ..from )、DDL(create ... ,drop ....)

b.回滚

i.显示回滚：rollback

ii.隐式回滚：异常退出（宕机、断电）

6.3、保存点savepoint:

打游戏： 10 ： 1 ,2(savepoint) , 3,4,5 (savepoint) ,6,7,8 -->rollba

语法： savepoint 保存点名字

insert into xx values(1,'zs');

insert into xx values(2,'ls');

savepoint initdate ;

insert into xx values(3,'ww');

回滚到特定保存点

rollback to savepoint initdate ;

6.4、事务的隔离级别：

多个事务会产生很多并发问题：隔离级别用来处理并发问题

1.脏读：当一个事务正在访问数据，并对此数据进行了修改（1->2），但是这种修改【还没有提交到数据库(commit)】； 此时，另一个事务也在访问这个数据 。本质： 某个事务（客户端）读取到的数据是 过时的。

2.不可重复读： 在一个事务内（客户端）内，多次读取同一个数据，但结果不同。

本质：就是事务A拿到了 被其他事务B修改并提交后的数据

3.幻读（虚读）：在一个事务内（客户端）内，多次读取同一批数据，但结果不同。

不可重复读和幻读的区别：

a.不可重复读指的是对于“同一条”数据的查询操作 a ->b

幻读对于“多条数据”的查询操作，数据量数： 20条 -> 18条

b.不可重复读：update

幻读：insert|delete

四种隔离级别的程度 依次递进（解决 并发的效果，越来越 稳定） ，但是性能越来越低。

并发性 、可用性 本身就是矛盾的。

Oracle只支持其中两种：Read Committed（默认），Serializable

（oracle自身扩种了一种 read only，实际 read only隶属于 Serializable级别 ）

6.4.1、切换四种隔离级别：

必须在开启事务的开始就切换隔离级别。Oracle99%事务级别都是read committed

set transaction isolation level Serializable;

切换read only：

set transaction read only ;

MySQL 支持全部的四种

1. PLSQL

对SQL进行编程

7.1、开发工具

plsql developer

oracle sql developer

7.2、plsql定义变量

declare

变量、常量、光标（游标）、例外（自定义异常）

begin

...

exception

...

end ;

示例

set serveroutput on ; 打开文本输出按钮

declare

--变量、常量、光标（游标）、例外（自定义异常）

psex char(3) := '男' ;

pname varchar2(10) ;

begin

select ename into pname from emp where empno =7499 ;

dbms\_output.put\_line('hello world:' || psex ||'---' || pname);

end ;

pname emp.ename%type ;--引用型变量，推荐

记录型变量:java 对象,可以用于同时保存多个变量值

示例

set serveroutput on ;

declare

--变量、常量、光标（游标）、例外（自定义异常）

emp\_info emp%rowtype ;

begin

select \* into emp\_info from emp where empno =7788;

dbms\_output.put\_line( emp\_info.empno || '---' || emp\_info.ename ||'--- '|| emp\_info.job);

end ;

7.3、plsql逻辑判断语法

if:

1.

if 条件 then ... ;

end if;

2.

if 条件 then ...;

else ....

end if;

3.

if 条件 then ..;

elsif 条件 then ...

else ..

end if ;

set serveroutput on ;

declare

pnum number := 3 ;

begin

if pnum=1 then dbms\_output.put\_line('一');

elsif pnum=2 then dbms\_output.put\_line('二');

else dbms\_output.put\_line('其他');

end if ;

end ;

循环while do..while for

do{}while(i<5)

1.

while 条件

loop

...

end loop ;

2.

loop

...

exit when i>5 ;

end loop;

3

for i in 1 .. 10

loop

..

end loop ;

示例

set serveroutput on ;

declare

begin

for x in 1 .. 5

loop

dbms\_output.put\_line(x);

end loop ;

end ;

--1-5之和

declare

pnum number:=1 ;

psum number:= 0 ;

begin

loop

exit when pnum >5 ;

psum := psum + pnum ; --sum+= i ;

pnum := pnum +1 ;

end loop ;

dbms\_output.put\_line(psum);

end ;

7.4、游标（光标）

语法：

定义

cursor 光标名(参数列表)

is

select ....

光标的属性

%isopen %rowcount %found %notfound

是否打开 当前指向行 是否存在下一行 是否不存在下一行

--查询并打印全部员工的姓名、薪水

set serveroutput on ;

declare

--变量、常量、光标（游标）、例外（自定义异常）

cursor cemp is select ename ,sal from emp ;

pename emp.ename%type ;

psal emp.sal%type ;

begin

open cemp ;--1.打开光标

loop --2.循环 准备获取每一行数据

fetch cemp into pename,psal ;--3.一行一行获取光标的值

exit when cemp%notfound ;

dbms\_output.put\_line(pename||'的工资是：'||psal);

end loop ;

close cemp ;--关闭光标

end ;

--总工资： president 1000 ，manager 800， 其他 400

set serveroutput on ;

declare

cursor cemp is select empno, job from emp ;

pempno emp.empno%type ;

pjob emp.job%type ;

begin

open cemp ;

loop

fetch cemp into pempno,pjob ;

exit when cemp%notfound ;

--判断职位

if pjob = 'PRESIDENT'

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 ;

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

close cemp ;

commit ; --ACID oracle read commit ，一边(终端)不提交，另一个访问不到

end ;

自定义例外

MyException

try{

if(xxx) throw new MyException() ;

}catch(){}

catch(){}

catch(){}

set serveroutput on ;

declare

myexc exception ;

pnum number := 1 ;

begin

if pnum =1 then raise myexc ;

end if;

exception

end ;

是否存在编号50的部门，如果不存在 ，抛出一个例外；如果存在，将该部门的员工姓名 打印。。

set serveroutput on ;

declare

cursor cemp(dno number) is select ename from emp where deptno = dno ;

pename emp.ename%type ;

no\_emp\_found exception ;

begin

open cemp(50);

fetch cemp into pename ;

if cemp%notfound then raise no\_emp\_found ;

else

loop

exit when cemp%notfound ;

fetch cemp into pename ;

end loop ;

end if ;

exception

when no\_emp\_found then dbms\_output.put\_line('自定义例外,没有此号部门...');

close cemp;

end ;

7.4.1、习题

1.统计每年的入职人数

sql:

select to\_char(HIREDATE,'yyyy' ),count(1) from emp group by to\_char(HIREDATE,'yyyy' );

plsql：

java:每个人数据全获取， 判断入职年份， 如果1980 , +1 ;+1

set serveroutput on ;

declare

cursor cemp is select to\_char(HIREDATE,'yyyy' ) from emp ;

phiredate varchar(4) ;

count80 number:=0 ;

count81 number:=0 ;

count82 number:=0 ;

count87 number:=0 ;

begin

open cemp ;

loop

fetch cemp into phiredate ;

exit when cemp%notfound ;

if phiredate = '1980' then count80 := count80+1 ;

elsif phiredate = '1981' then count81 := count81+1 ;

elsif phiredate = '1982' then count82 := count82+1 ;

else count87 := count87+1 ;

end if ;

end loop ;

close cemp ;

dbms\_output.put\_line('1980'||count80);

dbms\_output.put\_line('1981'||count81);

dbms\_output.put\_line('1982'||count82);

dbms\_output.put\_line('1987'||count87);

dbms\_output.put\_line('总人数'||(count80+count81+count82+count87));

end ;

2.涨工资。 每个10%， 按入职时间顺序涨工资，且涨后的总工资不能超过5万。计算 需要涨工资的人个数 以及涨后的工资总额。

set serveroutput on ;

declare

cursor cemp is select empno,sal from emp order by hiredate asc ;

pempno emp.empno%type ;

psal emp.sal%type ;

countEmp number :=0 ;

salTotal number :=0 ;

begin

OPEN cemp ;

loop

--exit when salTotal > 50000 ;-- 5.1

fetch cemp into pempno ,psal ;

exit when cemp%notfound ;

if salTotal + psal\*1.1 < 50000

--涨工资

then update emp set sal = sal\*1.1 where empno = pempno;

countEmp := countEmp +1;

--salTotal: 50 ,51

salTotal := salTotal + psal\*1.1 ; --psal 是变量， sal是表的字段

else

dbms\_output.put\_line('人数'||countEmp ||'--'|| '涨后的总额'||salTotal);

exit ;

end if ;

end loop ;

close cemp ;

end ;

3.统计各部门的工资情况。格式如下

部门编号 <2000的人数 2000-4000人数 >4000人数 工资总额

count1 count2 count3 salTotal

光标保存各个部门：10 20 30

set serveroutput on ;

declare

cursor cdept is select deptno from dept ;--10 20 30 40

pdeptno dept.deptno%type ;

--部门中员工的所有工资

cursor cemp(dno number) is select sal from emp where deptno = dno;

psal emp.sal%type ;

count1 number;

count2 number;

count3 number ;

--各部门工资总额

salTotal number ;

begin

open cdept;

loop --外层循环：遍历所有的部门 编号；；内层循环：遍历某个部门的所有工资

fetch cdept into pdeptno ;

exit when cdept%notfound ;

count1 :=0 ;

count2 :=0 ;

count3 :=0 ;

select sum(sal) into salTotal from emp where deptno =pdeptno ;

open cemp(pdeptno) ;

loop

fetch cemp into psal ;

exit when cemp%notfound ;

if psal<2000 then count1:=count1+1;

elsif psal>=2000 and psal<4000 then count2:=count2+1;

else count3:= count3+1 ;

end if ;

end loop ;

close cemp ;

dbms\_output.put\_line(pdeptno||' '||count1||' '||count2||' '||count3||' '||salTotal);

end loop;

close cdept;

end ;

7.5、存储过程/存储函数

7.5.1、无参存储过程语法:

语法：

create or reaplace procedure 过程名(参数列表)

as

PLSQL语句 ；

具体语法：

无参

create procedure 过程名(参数列表)

as

begin

end ;

7.5.2、无参存储过程示例：

示例

set serveroutput on ;

create or replace procedure test1

as

pnum number:=10 ;

begin

dbms\_output.put\_line('hello:'||pnum);

end ;

7.5.3、存储过程调用方法：

1、exec test1();

2、 begin

test1();

test1();

test1();

end ;

7.5.4、有参存储过程创建语法

有参：

输入参数in

输出参数out （返回值）

--传入一个员工编号，给该员工涨500

create or replace procedure raiseSalary(pid in number)

as

psal emp.sal%type ;

begin

select sal into psal from emp where empno=pid ;

update emp set sal = sal + 500 where empno=pid ;

dbms\_output.put\_line(psal || '--'||psal+500);

end ;

存储函数:与存储过程的最大区别： 存储函数必须有return

存储过程用in、out输出

7.5.5、调试debug和存储过程存储函数的选择

调试需要为角色赋予权限

grant DEBUG CONNECT SESSION to scott

grant DEBUG ANY PROCEDURE to scott;

存储过程/存储函数？

建议:只有一个返回值 存储函数 return

没有、多个返回值 ，存储过程 out out out

7.5.6、光标来保存集合数据

-查询某个部门的所有员工信息

select \* from emp where deptno = 10 ;

光标来保存集合数据 ：out

返回值/输出参数如果是光标类型： 包

包 = 包头+包体

public void aa() 包头

{

包体

}

包头：右键程序包：

CREATE OR REPLACE

PACKAGE BODY MYPACKAGE AS

procedure queryEmpList(dno in number , empList out empcursor) AS

BEGIN

END queryEmpList;

END MYPACKAGE;

包体： 右键包- 创建体

创建示例：

CREATE OR REPLACE

PACKAGE BODY MYPACKAGE AS

procedure queryEmpList(dno in number , empList out empcursor) AS

BEGIN

open empList for

select \*from emp where deptno = dno;

END queryEmpList;

END MYPACKAGE;

验证：desc mypackage ;

SQLdeveloper无法执行测试package

7.6、触发器

触发器：

与表相关联的，PLSQL程序

当执行DML，自动执行触发器

语法

create or replace trigger 触发器名

before|after

delete|insert |update [of 列名]

on 表

for each row [ when (条件)]

...plsql 代码

/

当插入一个数据时，自动打印“成功”

select \*from student;

create trigger logStudent

after

insert

on student

declare

begin

dbms\_output.put\_line('增加成功');

end ;

/

测试

set serveroutput on ;

insert into student(stuno,stuname,stuage) values(22,'zs22',222);

set serveroutput on ;

create trigger logupdateStudent

after

update of stuname

on student

declare

begin

dbms\_output.put\_line('您修改了姓名...');

end ;

/

无论修改多少行，触发器只执行一次。

原因：默认是 语句级触发器，作用于表，只执行一次。

语句级触发器：

作用于表，只执行一次。

行级触发器：

作用于每一行，每满足一次条件 都执行一次；可以执行多次触发器

for each row [when 条件]

行级触发器：

create or replace trigger logaddStudent

after

insert

on student

for each row

declare

begin

dbms\_output.put\_line('增加成功！');

end;

/

insert into student(stuno,stuname)

select empno,ename from emp;

--不能在非工作时间插入员工信息

周一 - 周五 9：00 - 18:00

星期几：select to\_char(sysdate, 'day') from dual;

小时：select to\_char(sysdate, 'hh24') from dual;

create or replace trigger securityStudent

before insert

on student

begin

--校验 --不正常

if to\_char(sysdate,'day') in ('星期四','星期日') or to\_number( to\_char(sysdate,'hh24') ) not between 9 and 18

then

--禁止插入，例外

raise\_application\_error(-20001,'禁止非工作时间插入学生' );

end if ;

end ;

/

使用触发器确保：涨工资，涨后的工资 不能少于涨前的工资

create or replace trigger checkSalary

before update

on emp

for each row

begin --new old

if :new.sal < :old.sal --3000 - 5000

then

raise\_application\_error(-20002 , '涨后的工资不能小于涨前的！');

end if ;

end ;

/

raise\_application\_error中编号的范围 -20000 - -20999

数据校验: web前端 onsubmit onblur --> 控制器 Servlet/Spring MVC if()

->数据库 触发器

7.7、数据字典

数据字典（了解） ：元数据， 数据库的各种描述信息，系统自带很多表

dictionary、user\_objects 、user\_tables、user\_tab\_columns ;

all\_tables

数据字典的命名规范：

user：当前用户能够使用的

all：系统中全部的

dba:管理员

v$:性能相关

user\_sequences ;

user\_synonyms;

user\_关键字s;

user\_tab\_comments

all\_tab\_comments

7.8、闪回

作用： 将错误的DML 并且commit， 进行撤销 （撤销已提交的事务）；

还原删除的表 drop table xxx ;

获取表上的历史记录

类型：

闪回表中的数据：将表中的数据 回退到历史的某一个点

闪回删除的表：还原表

闪回事务查询： undo\_sql

闪回数据库（了解）：将数据库 中的数据 回退到历史的某一个点

闪回归档日志（了解）

7.8.1、闪回表中的数据：将表中的数据 回退到历史的某一个点

必须sys

show parameter undo;

假设闪存区3G

undo\_retention：900 ，超过闪存区的数据 必须在900秒内闪回

alter system set undo\_retention=1200 scope = both ;

both| memory 当前数据库有效，重启无效| spfile 当前数据库无效，重启有效

闪回：时间点SCN

闪回：必须知道 回退的时间点 或者SCN ,时间->SCN

时间->SCN ：

select timestamp\_to\_scn(sysdate) scn from dual ;

select timestamp\_to\_scn(to\_date('2021-03-15 23:30:20','yyyy-MM-dd HH24:mi:ss')) scn from dual

flashback

create table testfb

(

id number ,

name varchar(10)

);

insert into testfb values(1,'a');

insert into testfb values(2,'b');

insert into testfb values(1,'c');

commit ;

此时SCN:3607794

删除id=1

commit ;

闪回：

flashback table testfb to scn 3607794 ;

如果某个用户在闪回时权限不足，需要授权：

sys:

grant flashback any table to scott;

开启表的行移动功能

alter table testfb enable row movement;

闪回 可以跨越commit 回退。

7.8.2、闪回删除的表（还原oracle回收站中的表 ）

查看回收站

show recyclebin;

默认情况下：如果回收站中重名，闪回的是最近一次删除的表

如果闪回的表 和库中目前的表 重名 ，则冲突，必须 重命名rename to..

flashback table 表名 to before drop ;

flashback table "回收站中的名字" to before drop ;

清空回收站 purge recyclebin;

彻底删除某一张表（不过回收站）： drop table testfb2 purge ;

普通用户： 删除->回收站 -> 清空|还原

管理员：删除表 不过回收站，直接彻底删除。

是否闪回表中的触发器：

A(触发器) -> 删除->回收站 ->闪回表时 是否连同触发器一起闪回

flashback table 表明 to before drop [rename to 新表名] enable|disable（默认） triggers ;

create table testfb4

(

id number ,

name varchar(10)

);

flashback table enable triggers testfb4 to before drop ;

7.8.3、闪回事务查询

区别：

闪回dml+commit：的不同之处：

闪回dml+commit ： 依赖时间 ->scn

闪回事务：依赖的是 提交次数（undo\_sql）

a.提交次数 ？ 闪回版本查询(事务编号)

坑： （ bug）dml操作数据时 不要太快（批量复制 ）

凡是能够撤销的 事务，都必须提前开日日志：在sys中执行alter database add supplemental log data;

create table tb\_version(id number,name varchar2(10)) ;

insert into tb\_version values(11,'a1');

insert into tb\_version values(12,'a1');

insert into tb\_version values(13,'a1');

commit;

insert into tb\_version values(21,'b1');

insert into tb\_version values(22,'a1');

insert into tb\_version values(23,'a1');

commit;

insert into tb\_version values(31,'c1');

insert into tb\_version values(32,'a1');

insert into tb\_version values(33,'a1');

commit;

--bug ：如果插入数据且commit 速度太快，可能会丢失 事务信息

闪回版本查询:

版本信息：伪列

select id,name ,versions\_xid,versions\_operation,versions\_starttime,versions\_endtime from tb\_version versions between timestamp minvalue and maxvalue;

"2019-05-19" ->timestamp 类型

"2019-05-21"

select id,name,versions\_xid,versions\_operation,versions\_starttime,versions\_endtime from tb\_version versions between timestamp minvalue and maxvalue ;

事务闪回的核心： 查询undo\_sql ，执行即可

需要授权： grant select any transaction to scott;

undo\_sql存在于表： desc flashback\_transaction\_query;

查询undo\_sql前 需要在sys中修改设置：alter database add supplemental log data;

select OPERATION ,UNDO\_SQL from flashback\_transaction\_query where XID ='050013005F080000';

select XID,OPERATION ,UNDO\_SQL from flashback\_transaction\_query ;

7.9、导入导出

导出：exp

exp不是sql语句，是一个工具，直接在cmd中执行

a.表方式

exp scott/tiger@127.0.0.1/ORCL file=d:/back/bk.dmp log=d:/back/log.log tables=emp,dept

b.用户方式：

exp scott/tiger@127.0.0.1/ORCL file=d:/back/bk2.dmp log=d:/back/log2.log

c.全库方式:DBA角色的 （sys / system ）

exp system/manager@127.0.0.1/ORCL file=d:/back/bk2.dmp log=d:/back/log2.log full=y

导入

a.表方式

imp hr/hr@127.0.0.1/ORCL file=d:/back/bk.dmp log=d:/back/log.log tables=emp,dept fromuser=scott touser=hr commit=y ignore=y

b.用户方式：(DBA角色)

imp system/manager@127.0.0.1/ORCL file=d:/back/bk2.dmp log=d:/back/log2.log fromuser=scott touser=hr commit=y ignore=y

c.全库方式:

imp system/manager@127.0.0.1/ORCL file=d:/back/bk2.dmp log=d:/back/log2.log commit=y ignore=y destroy=y

1. 分布式数据库

大->小

物理上分开存放，逻辑上一个整体

独立性：客户端不必关心数据如何分割和存储 ，只需要关心数据本身即可

分布式数据库三种操作：

a.本地操作

b.远程操作 sqlplus scott/tiger@192.168.2.128/orcl

注意： i. 关闭远程防火墙

ii.远程： 将2个文件（tnsnames.ora、listener.ora） 的Host值改成 IP地址或计算机名

c.分布操作

借助于数据库链路：

其中l2vm 是链路名

remoteORCL是服务名 ，用于连接远程的服务（orcl）

创建服务名： net manager

创建链路：（链路的单向的）

create database link l2vm connect to scott identified by tiger using 'remoteORCL';

同时操作本地和远端 ，关联查询

select ename,dname from emp@l2vm e,dept d where e.deptno = d.deptno ;

emp@l2vm-> remoteemp

select ename,dname from emp@l2vm e,dept d where e.deptno = d.deptno ;

创建同义词 屏蔽掉分布式访问的 关键字

create synonym remoteemp for emp@l2vm ;

select ename,dname from remoteemp e,dept d where e.deptno = d.deptno ;

视图

create view empremoteview

as

select ename,dname from remoteemp e,dept d where e.deptno = d.deptno ;

select \* from empremoteview;

查看星期的显示格式： select to\_char(sysdate,'day') from dual;

快照：可以备份远程数据库（快照有一定延迟）

create snapshot empdeptss

refresh start with sysdate

next next\_day(sysdate,'星期三')

as

select \* from emp@l2vm

;

触发器：（实时备份）

工资问题：如果本地数据库中修改工资了，则同步到远程

create or replace trigger update\_emp

after update on emp

for each row

begin

update emp@l2vm set sal=:new.sal where empno=:new.empno ;

end ;

/

update后注意commit;