# 体系结构

## 创建表过程

数据库可以比作类，实例就是数据库的实例化对象

oracle数据库 ---> 数据库实例 ---> 表空间(逻辑单位)(用户) ---> 数据文件(物理单位)

地球 ---> 国家 ---> 省份(逻辑单位)(公民) ---> 山川河流(物理单位)

第一步 了解数据库:database.(可以认为它就是一个系统或java中的一个类，也是唯一的类)

Oracke 数据库是数据的物理存储。这就包括(数据文件ORA或者DBF\控制文件、联机日志、参数文件)。

其实Oracle数据库的概念和其它数据库不一样，这里的数据库是一个操作系统只有一个库。

Qracle就只有一个大数据库

第二步 创建实例,默认实例orcl一个

oracle只有一个数据库，但是可以新建很多个实例(mysql是数据库)，但是一般情况下，数据库=实例。默认实例：oracle，需要解锁学习的实例：scott\hr

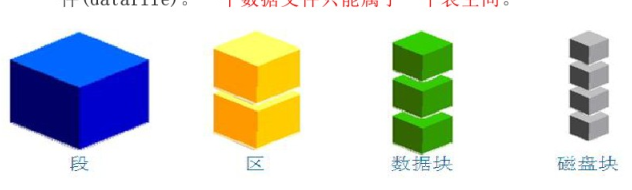
一个oracle实例(Oracle instance)有一系列的后台进程（Backguound Processes）和内存结构（Memory Structures）组成

Java连接数据库就是连接实例：如创建了一个实例wwl；下面是连接url：

jdbc:oracle:thin:@localhost:1521:wwl

第三步 创建表空间

表空间是oracle对物理数据库上相关数据文件(ORA或者DBF文件)的逻辑映射。一个数据库在逻辑上呗划分成一到若干个表空间，每个表空间包含了在逻辑上相关联的一组结构。每个数据库至少有一个表空间(称之为system表空间)每个表空间由同一磁盘上的一个或多个文件组成。这些文件叫数据文件（datafile）。一个数据文件只能属于一个表空间



创建表空间: 逻辑单位, 通常我们新建一个项目,就会去新建表空间,在表空间中创建用户来创建表

语法:

create tablespace 表空间的名称

datafile '文件的路径(服务器上)'

size 大小

autoextend on 自动扩展

next 每次扩展的大小

--创建一个表空间 --- 汉东, (切换到system帐号下创建)

create tablespace handong

datafile 'c:\handong.dbf'

size 100m

autoextend on

next 10m;

--删除表空间(只会删除表空间逻辑，不会删除物理磁盘dbf文件，需要手工删除)

drop tablespace handong;

第四步 数据库文件(dbf)：创建了表空间就会生成一个数据库文件

数据文件是数据库的物理存储单位。数据库的数据是存储在表空间中的，真正是在某一个或者多个数据文件中。而一个表空间可以由一个或多个数据文件组成，一个数据文件只能属于一个表空间。一旦数据文件被加入到某个表空间后，就不能删除这个文件，如果要删除某个数据文件，只能删除其所属于的表空间才行

第五步 用户：需要先表空间创建用户

用户是在实例下建立的。不同实例中可以建相同名字的用户

create user 用户名

identified by 密码

default tablespace 表空间的名称

create user dakang

identified by dakang

default tablespace handong

授权 grant 角色 | 权限 to 用户 (oracle存在三个重要角色：connect,resource,dba)

Connect：是授予最终用户的典型权利，最基本的，没有建表权限

Resource：--是授予开发人员的有增删改查，有建表权限等

Dba角色：只有dba才可以创建数据库结构，并且系统权限也需要DBA授出，且Dba用户可以操作全体用户的任意基表，包括删除

grant connect to dakang;

--授予 dba的角色

grant dba to dakang;

第六步 创建表 通过用户创建表

创建表:关键字create

create table 表名(

列名 列的类型 [列的约束],

列名 列的类型 [列的约束]

);

列的类型:

varchar ,在Oracle中,目前是支持的, 但是不保证以后还支持

varchar2(长度) 可变字符长度 varchar2(10) hello 占5个字符

char(长度) 固定长度字符 char(10) hello 占10个字符,用空格填充

number(总长度,小数长度) 数字类型 --小数长度不能大于等于总长度

date 年月日时分秒 2017/4/13 9:43:49

timestamp 时间戳, 比date类型更加精确 13-APR-17 09.44.08.272000 AM +08:00

LONG/CLOB : 存放一本小说

BLOB : 存放电影 java 存进去, 再读取出来

复制表 使用子查询的方式创建表

create table 表名 as 查询语句;

注意: 只会复制表结构和表中的数据,不会复制列的约束

如果查询语句有结果, 就是复制 表结构和数据

如果查询语句没有结果, 就是复制 表结构

--复制scott账户中表emp数据操作

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

--如果查询语句是没有任何的结果的：只复制表结构，没有数据

select \* from scott.emp where 1=2;

create table emp1 as select \* from scott.emp where 1=2;

第七步 约束

表的五大约束：主要是用来约束表中数据的规则

主键约束: primary key 不能为空, 必须唯一

非空约束: not null

唯一约束: unique

检查约束 check(条件) 在mysql中是可以写的,但是mysql直接忽略了检查约束 check( gender in ('男','女','人妖'))

create table student(

stuid number primary key, #主键约束

sname varchar2(10) unique, #唯一约束

age varchar2(10) not null,#非空约束

gender varchar2(4) check( gender in ('男','女','人妖')) #检查约束 只能输入三种字符男 女 人妖

);

外键约束:主要是用来约束从表A中的记录,必须是存在于主表B中

--添加外键约束语句

alter table wwl1 add foreign key(cno) references ww2(cid); #wwl1是子表，wwl2是主表

--级联删除 : 首先去从表中找有没有 关联数据, 如果在从表中找到关联数据,先删除从表中关联数据,然后再删除表中的数据

oracle与MySQL区别

新建一个项目区别:

MYSQL : 创建一个数据库,创建相应的表

Oracle: 数据库 ---> 数据库实例ORCL ---> 表空间 (用户里面的创建表) ---> 数据文件

Oracle是多用户的, MYSQL是多数据库的

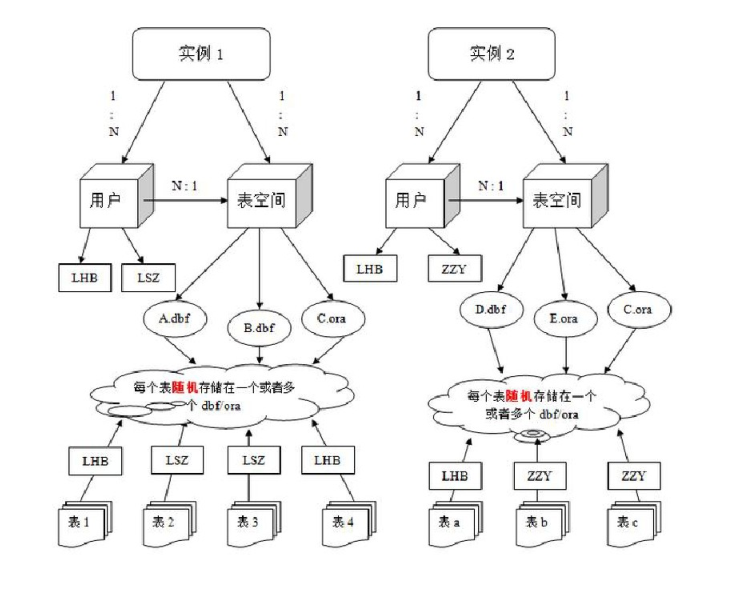
1. 遵循SQL标准

2. 不同厂商,不同的数据库产品,但是有自己的方言

3. 使用自己的方言,也能够完成相同的功能

4. Oracle安全级别要高,MYSQL开源免费

创建过程图



## 视图

视图: 是对查询结果的一个封装 49期oracle(4天)的第三天14节

视图里面所有的数据,都是来自于它查询的那张表,视图本身不存储任何数据

语法: create [or replace] view 视图的名称 as 查询语句 [ with read only]

1、能够封装复杂的查询结果：（视图封装复杂的查询语句）

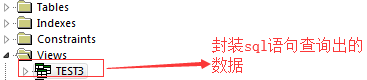
第一步：准备复杂语句

Select sum(cc) "TOTAL",sum(case yy when '1980' then cc end) "1980" from (select to\_char(hiredate,'yyyy') yy,count(1) cc from emp group by to\_char(hiredate,'yyyy')) tt;

第二步：封装语句操作 create view test3 as (test3可以认为是一张表)

create view test3 as select \*\*\*\*\*from (select to\_char(hiredate,'yyyy') yy,count(1) cc from emp group by to\_char(hiredate,'yyyy')) tt;

第三步：执行完以后在plsql工具中view就会有一张虚表test3

注意 真正的表中数据变了，这里的数据也会跟着变化

第四步：进行查询封装后的test3虚表即可

Select \* from test3

2、屏蔽表中的细节: 老板不信任你，屏蔽真正表的核心关键数据，复制一个表(关键信息屏蔽了)给你

如员工表：有工资，不希望你看到工资信息，就可以屏蔽掉工资字段，生成一个虚表让你去查询此张表即可

--创建一个只读视图 with read only

create or replace view view\_test2 as select ename,job,mgr from emp with read only;

注意: 通常不要通过视图去修改,视图创建的时候,通常要加上with read only

## 序列

序列: 生成类似于 auto\_increment(这是mysql主键自增长) 这种ID自动增长 1,2,3,4,5....

语法:

create sequence 序列的名称

start with 从几开始

increment by 每次增长多少

maxvalue 最大值 | nomaxvalue

minvalue 最小值 | nominvalue

cycle | nocycle 是否循环 1,2,3,1,2,3

cache 缓存的数量3 | nocache 1,2,3,4,5,6

如何从序列获取值

currval : 当前值

nextval : 下一个值

注意: currval 需要在调用nextval之后才能使用 永不回头,往下取数据, 无论发生异常, 回滚

--第一步 创建一个 1,3,5,7,9......30

create sequence seq\_test1

start with 1

increment by 2

maxvalue 30

cycle

cache 10;

第二步 执行上面语句后，进行运行和取值

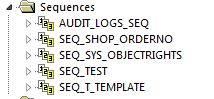
select seq\_test1.nextval from dual; #运行

select seq\_test1.currval from dual; #取值

--序列企业用的最多的一种写法：就是全部用默认：不设置最大值，每次增长1，从1开始

create sequence seq\_test2;

生成序列可以在工具pl/sql中sequence查看



注意

## 索引

索引:相当于是一本书的目录,能够提高我们的查询效率

如果某一列,你经常用来作为查询条件,那么就有必要创建索引,数据量比较的情况

语法: create index 索引的名称 on 表名(列)

注意:主键约束自带主键索引, 唯一约束自带唯一索引

使用步骤

第一：先有数据很多的表，没有可自己使用函数创建一张数据很多的表

--插入500000万条数据

declare

begin

for i in 1..5000000 loop --fro循环语法

insert into user values('姓名'||i,'地址'||i);

end loop;

commit;

end;

第二： 使用索引和没有索引区别 创建单索引与复合索引 语法 create index 索引名字 on 表名字(表字段,复合索引字段);

--在没有添加索引的情况下,去查询 name='姓名3000000' --2.985

select \* from wwl where name='姓名3000000';

create index ind\_wwl on wwl(name); - -创建索引

select \* from wubaiwan where name='姓名3000000'; --创建索引 name 再去查询 name='姓名3000000' 用时：--0.016

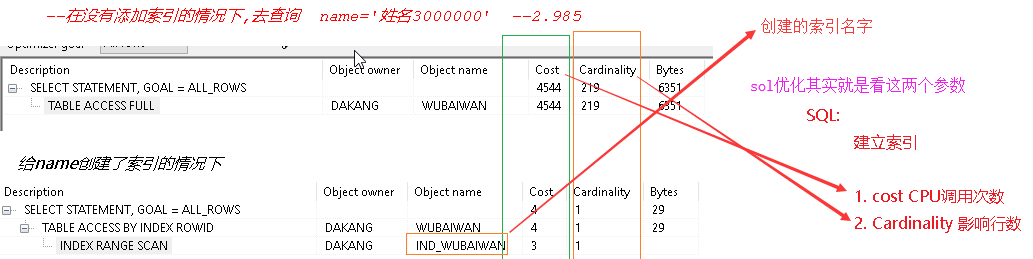
--在没有添加复合索引的情况下,再去查询 name='姓名3000000' and '地址3000000'

select \* from wubaiwan where name='姓名3000000' and address='地址3000000'; --0.032

create index ind\_wubaiwan2 on wubaiwan(name,address); --创建复合索引的情况下, 再去查询

select \* from wubaiwan where name='姓名3000000' and address='地址3000000'; --0.015

第三： sql运行完后：查询运行效率 按f5



索引原理

索引原理: btree balance Tree 平衡二叉树

如果某列作为查询条件的时候,可以提高查询效率,但是修改的时候,会变慢

索引创建好之后,过了一段时间,DBA都会去做重构索引

# 数据库操作

DDL : 数据定义语言 create alter drop truncate

DML : 数据操纵语言 insert update delete

DCL : 数据控制语言 安全 授权 grant revoke

DQL : 数据查询语言 select from子句 where子句

## 基本查询

编写顺序和执行顺序

Sql的编写顺序

Select……from ….where ….group by ……having….order by

Sql的执行顺序

--执行顺序 ：(from)先找目标表，(where)进行条件刷选获得结果集，(group by)根据列有重复的进行分组，(having)分组完后进行过滤得到最后的结果集，(select)查询需要显示的列，(order by)最后进行排序

From .. where ..group by ….having……select…order by

查询语句的结构:

select [列名] [\*] from 表名 [where 条件] [group by 分组条件] [having 过滤] [order by 排序]

### 伪表(dual)

dual : oracle中的虚表 ,伪表, 主要是用来补齐语法结构

select 1+1; --在Oracle等于报错 ,在MYSQL中输出结果是2

select 1+1 from dual;

### 去除重复(distinct)

单列（字段）

如：scott表中的emp(员工表)部门编号字段(deptno)是有重复的

select deptno from emp; //查询到有重复的

select distinct deptno from emp //使用关键字distinct去除重复的

多列

多列去除重复: 每一列都一样才能够算作是重复

如:还是emp表：现在查询两个字段： job,deptno

select distinct deptno from emp：//会显示有：这两个字段都相同的

select distinct job,deptno from emp; //排除了两个字段都一致的

### 四则运算

select 1+1 from dual;

--查询员工年薪 = 月薪\* 12

select sal\*12 from emp;

--查询员工年薪+奖金：这里comm会有null值

注意: null值 , 代表不确定的 不可预知的内容 , 不可以做四则运算

select sal\*12 + comm from emp; //会出现很多空值行

使用--nvl 函数 : 如果参数1为null 就返回参数2；这样就能解决问题

select sal\*12 + nvl(comm,0) from emp;

### 字符串拼接(||)

oracle数据库拼接是：||(oracle特有的) 而不是+

在Oracle 中 ,双引号主要是别名的时候使用, 单引号是使用的值, 是字符

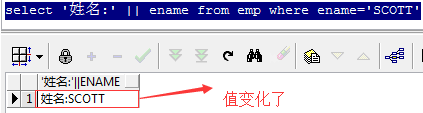
查询员工姓名 : 姓名为:SCOTT

select ename from emp where ename='SCOTT'

使用oracle特有的拼接字符，查询出的值带”姓名：”

select “姓名：” ename from emp where ename='SCOTT'

查询结果值：



注意：Mysql是没有的，使用函数：concat(str1,str2) 函数效果和上面一样, 在mysql和Oracle中都有

### 条件查询(where)

条件查询 : [where后面的写法]

where和having区别

where 后面不能接聚合(多行)函数或(子查询是或多行数据)，可以接单行函数

having 是用在group by之后执行，可以接聚合(多行)函数

关系运算符：就是和Java差不多的

关系运算符: > >= = < <= != <>

三个逻辑运算符:、

And 并且条件，如两个条件，这两个条件必须都符合才显示出来

or、或条件，如两个条件：这两个条件哪一个成立都显示出来，有中文字和、或 一般都会用到or

--找出部门10中所有经理和部门20中所有办事员的详细资料。

select \* from emp where (deptno='10'and job='MANAGER') or (deptno='20' and job='CLERK')

not in(value1,value2) 如同非等于

--显示除了编号为10的部门所有员工信息

select \* from emp where deptno not in(10); 第一种写法

select \* from emp where deptno != 10; 第二种写法 使用关系运算符 !=

其它运算符:

1、like 模糊查询

select \* from emp where ename like '%K%'

2、in(set) 在某个集合内 相当于Java集合

字段名 in(‘value’,’value2’,’value3’)

--查询名字在某个范围的员工信息 in ('JONES','SCOTT','FORD')

select \* from emp where ename in ('JONES','SCOTT','FORD');

3、between..and.. 在某个区间内

--查询工资在1500--3000之间的员工信息

select \* from emp where sal between 1500 and 3000; 第一种方式：between是英语当中的一个介词，意思为在......之间，常与and连用,如between A and B

select \* from emp where sal >= 1500 and sal <= 3000; 第二种方式

4、is null 判断为空

--查询每月能得到奖金的员工信息

select \* from emp where comm is not null;

5、is not null 判断不为空

select \* from emp where comm is not null

select \* from emp where comm !=null; //这个是不正确的 但也不会报错，

### 模糊查询

--查询员工姓名第三个字符是O的员工信息

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

--查询员工姓名中,包含%的员工信息

%需要转义：需要关键字配合：escape：告诉oracle这是转义字符

select \* from emp where ename like '%\%%' escape '\'; //告诉oracle转义字符是\

select \* from emp where ename like '%#%%' escape '#'; //告诉oracle转义字符是#

### 排序

排序 : order by

1默认升序: asc ascend降序: desc descend

2排序注意null问题 : nulls first | last

3同时排列多列, 用逗号隔开，注意 排序完后若有重复的才会有效果 看下面例子2

重要例子：有null，和多个排序

例子1查询员工信息,按照奖金由高到低排序：因此字段有null，需要用到关键字nulls

select \* from emp order by comm desc nulls last; //如果有null，默认是把null排在前面

例子2查询部门编号和按照工资 按照部门升序排序, 部门中的工资降序排序,然后按入职时间顺序来排序

select \* from emp e order by e.deptno asc,e.sal desc ,HIREDATE asc; //asc（升序）是默认的可以不写

-- 先进行部门排序(升序)完后如果有重复的，再进行工资排序(降序)，排序完后若工资有相同的，再进行入职排序(升序)

### rownum(伪列)

常用与：oracle的分页查询操作

rownum : 伪列, 系统自动生成的一列, 用来表示行号

rownum是Oracle中特有的用来表示行号的, 默认值/起始值是 1 ,在每查询出结果之后,再添加1

rownum最好不要做大于号判断(但是可以做大于0),可以做小于号判断

select rownum,e1.\* from emp e1 where rownum > 2 --没有任何记录，因为语句执行顺序问题：rownum关键字是先查询完了才生成一个编号

select rownum,e1.\* from emp e1 where rownum > =1 –这才有结果

select rownum,e1.\* from emp e1 where rownum > 0 --也有结果

select rownum,e1.\* from emp e1 where rownum < 8000 –有结果；查询7999之前的显示

--只查询员工表中工资最高的前三名信息，其它不需要

select rownum,t.\* from (select \* from emp order by sal desc)t where rownum<4; //这里要注意：要先排完顺序，在主查询显示rownum，不然是乱的

--分页查询emp表：下面是查询包括6到包括9的数据

第一种方式：自己想出来的(仿照网上)

select \* from (select rownum rowno,t.\* from emp t where rownum<=9) e where rowno>=6 //先查询出最终要到几的数据，然后主查询进行判断rownum的数字

第二种方式：老师方式

select \* from (select rownum rowno, emp.\* from emp) t where t.rowno between 6 and 9; //使用between关键字方式

### rowid

rowid : 伪列 每行记录所存放的真实物理地址

--去除表中重复记录 ：老师说这是华为面试的

delete from p p1 where rowid > (select min(rowid) from p p2 where p1.name = p2.name); //先查询出重复的最小的物理地址，删除剩余重复的物理地址

## 多表查询

### 内连接

隐式内联接:

等值内联接: where e1.deptno = d1.deptno;

不等值内联接: where e1.deptno <> d1.deptno; 没有意义

自联接: 自己连接自己：select \* from emp1,emp2 where

Scott账号中的emp表：--查询员工编号,员工姓名,此员工经理的编号, 此员工经理的姓名(都在一张表中记录)

select e1.empno,e1.ename,e1.mgr,m1.ename from emp e1, emp m1 where e1.mgr= m1.empno;

显示内联接:

select \* from 表1 inner join 表2 on 连接条件 // inner 关键字可以省略

--查询员工姓名和员工部门所处的位置

select e1.ename,d1.loc from emp e1,dept d1 where e1.deptno = d1.deptno; // 隐式

select \* from emp e1 inner join dept d1 on e1.deptno = d1.deptno; //显式

两个区别

当只是两张表时，显示和隐式都可以使用，而当需要多张表进行内链接时，就必须使用显示内连接的格式了

显式连接可以减少字段的扫描，有更快的执行速度。这种速度优势在3张或更多表连接时比较明显

显式条件需要用到是：on 不是where

### 外连接

外连接: (标准,通用写法)

左外连接: left outer join 左表中所有的记录,如果右表没有对应记录,就显示空

右外连接: right outer join 右表中的所有记录,如果左表没有对应记录,就显示空

outer 关键字可以省略

Oracle中的独有写法在内连接基础上加 (+) 实际上是如果没有对应的记录就加上空值：相当于就是左外连接

select \* from emp e1,dept d1 where e1.deptno = d1.deptno(+); //这是一个普通的内连接查询，但是加上(+)就变成右外连接了

### 子查询

子查询: 查询语句中嵌套查询语句; 用来解决复杂的查询语句

1、单行查询结果用在where条件判断语句中

查询出最高工资的人(很多实现方式，这只是告诉你子查询使用方式)

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

--查询出比雇员7654的工资高,同时和7788从事相同工作的员工信息

如果不使用子查询这个需要3步完成

--1.先查询出雇员7654的工资 1250

select sal from emp where empno = 7654;

--2.然后查询出7788从事的工作 ANALYST

select job from emp where empno = 7788;

--3.最后两个条件合并得出结果

select \* from emp where sal > 1250 and job = 'ANALYST';

使用嵌套子查询，一条语句查询出来：

select \* from emp where sal > (select sal from emp where empno = 7654) and job = (select job from emp where empno = 7788);

2、有多行查询结果用在from后使用（当作一张表）

--查询每个部门最低工资的员工信息

1、首先查询每个部门最低工资员工信息：涉及到根据部门进行分组

select deptno,min(sal) m from emp group by deptno

2、进行整合：把第一次查询到当作一张表放入from下：进行条件查询

select \* from emp t,(select deptno,min(sal) m from emp group by deptno) t2 where t.sal=t2.m and t.deptno=t2.deptno

3、子查询注意事项

1. 空值问题通常情况下, 数据库中不要出现null 最好的做法加上Not null( null值并不代表不占空间, char(100) null 100个字符)

--查询不是领导的信息

select \* from emp where empno not in (select mgr from emp); --错误：查询出来的空，因为此字段有null，

select \* from emp where empno not in (select mgr from emp where mgr is not null); --正确 子查询使用is not null排除了null值

### exists(查询语句)

exists(查询语句) : 存在的意思,判断一张表里面的记录是否存在与另外一张表中

1. 当作布尔值来处理(用的比较少，一般where a=b就可以解决):当查询语句有结果的时候, 就是返回true否则返回的是false

--使用exists判断子查询是否有值：有就是true执行总查询

select \* from emp where exists(select \* from emp where deptno = 1234567);--查看有没有员工为：1234567 没有返回false就不进行主查询了

select \* from emp where 3=4; //上面语句相当于这条，所以恨少用布尔值来处理

1. 数据量比较大的时候是非常高效的

--查询有员工的部门的信息

select \* from dept d1 where exists(select \* from emp e1 where e1.deptno = d1.deptno ); --查询到数据：拿dept表中deptno字段去emp中进行循环判断

## 判断时间

使用函数to\_char(字段名,’yyyy’)转换为年份

--查询入职日期是在1981(不包括1981)年以后的员工

--使用[转换函数](#_4、转换函数)把时间转换为字符类型类型进行对比

select \* from emp e where to\_char(e.hiredate,'yyyy')>1981 //查询结果有3条 这个是正确的

--使用[转换函数](#_4、转换函数)把字符转换为时间进行对比；注意问题

select \* from emp e where e.hiredate > to\_date('1981','yyyy') //查询结果有8条是不正确的，原因如下：

select to\_date('1981','yyyy') from dual;//结果是1981/7/1：此函数值：年份+本月份+1()： 我是7月份进行查询的所以是7月1日，所以判断以7月1日开始

select \* from emp e where e.hiredate > to\_date('1981-12-31','yyyy-mm-dd') //查询结果有3条 这样才正确

## 集合运算

集合运算:

注意事项

1.每个表查询结果列的类型要一致

2.每张表查询查询结果列要按照顺序匹配

3.每张表查询结果列的数量要一致,如果不足,用空值填充

并集: 将两个查询结果进行合并,关键字：unio union all

union : 去除重复的,并且排序

union all : 不会去除重复的

一般用在两张不同表以上：进行合并信息

查询工资大于1500和部门为20的所有员工信息：

select \* from emp where sal > 1500

union

select \* from emp where deptno = 20

交集:查询A表和B表中有关联的信息；关键字intersect

--工资大于1500,并且是20号部门下的员工

select \* from emp where sal > 1500; //查询到所有员工1500元以上的工资

select \* from emp where deptno = 20; //查询到全部20部门编号的员工

--进行交集：就获得编号为20号部门，并且大于1500元的员工信息

select \* from emp where sal > 1500

intersect

select \* from emp where deptno = 20;

差集运算:两个结果相减；如：A表和B表：获取A表信息所有信息(排除跟B表的重复信息) 关键字minus

--1981年入职员工(不包括总裁和经理)

--1981年入职员工

select \* from emp where to\_char(hiredate,'yyyy')='1981'; //查询的结果这里包括总裁和经理的信息

--查询总裁和经理信息

select \* from emp where job = 'PRESIDENT' or job = 'MANAGER';

--进行差集关键字minus，就获得去除总裁和经理后1981年入职员工信息

select \* from emp where to\_char(hiredate,'yyyy')='1981'

minus

select \* from emp where job = 'PRESIDENT' or job = 'MANAGER';

# 表达式

## 条件表达式(decode)

条件判断表达式：跟if else if else 效果一样

通用表达式：case ‘字段’ when 值1 then 值 when 值2 then 值 else 值 end “双引号取别名”

Scott账号：select case ename when 'SMITH' then '孙悟空' when 'ALLEN' then '人渣' else '傻帽' end "昵称" from emp

Oracle独有表达式：decode(字段,if1,值1,if2,值2,其它值)

select decode(ename,'SMITH','孙悟空','ALLEN','人渣','傻帽') from emp

## 分组表达式(group by)

select 分组的条件(这里只能写分组条件，也可以不写),分组之后的操作 from 表名 group by 分组的条件 having 条件过滤

注意事项：

1. Select与from之间：只能写1、group by 的条件字段， 2、分组后的操作：如统计count(1)、求平均值等avg(这里可以别的字段)
2. having放在group by 的后面（这不是废话吗）
3. group by 后面只能放非聚合函数的列
4. having 子句的作用是筛选满足条件的组，即在分组之后过滤数据，条件中经常包含聚组函数，使用having 条件显示特定的组，也可以使用多个分组标准进行分组。
5. having 子句被限制子已经在SELECT语句中定义的列和聚合表达式上。通常，你需要通过在HAVING子句中重复聚合函数表达式来引用聚合值，就如你在SELECT语句中做的那样

//统计emp表中的部门各有多少人

select deptno,count(1) from emp group by deptno //这样是可以的，因为deptno就是分组的条件，

----错误的示范：添加了一个字段ename

select ename, deptno,count(1) from emp group by deptno //希望显示名字，这样是不正确的，

1. Group by 不能在having过滤中使用别名来操作：因为执行语句顺序：having是在select之前运行的 看[语句执行顺序](#_基本查询)

--分组统计所有部门的平均工资

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

--分组统计所有部门的平均工资，找出平均工资大于2000的部门

select deptno,avg(sal) from emp group by deptno having avg(sal)>2000

----错误的示范：表达式中使用别名 tt

select deptno,avg(sal) tt from emp group by deptno having tt>2000 // 这是错误的因为having是在select之前运行的，在这之前无法得到别名值

# 函数

函数: 必须要有返回值

## 单行函数

单行函数: 对某一行中的某个值进行处理

### 1、数值函数

--取大于等于数值value的最小整数：ceil(value)

select ceil(45.926) from dual; --46

--取小于等于数值value的最大整数：floor(value)

select floor(45.926) from dual; --45

--四舍五入：round(value,要取小数点第几位)

select round(45.926,2) from dual; --45.93

select round(45.926,1) from dual; -- 45.9

select round(45.926,0) from dual; --46

select round(45.926,-1) from dual; --50

select round(45.926,-2) from dual; --0 为啥0，-2意思就是看十位数：十位数是4所以四舍五入进位0

select round(65.926,-2) from dual; --100：为啥100，-2意思就是看十位数：十位数是6所以四舍五入进位100

--截断：trunc(value, 要取小数点第几位)：不会进行四舍五入

select trunc(45.926,2) from dual; --45.92

select trunc(45.926,1) from dual; -- 45.9

select trunc(45.926,0) from dual; --45

select trunc(45.926,-1) from dual; --40

select trunc(45.926,-2) from dual; --0

select trunc(65.926,-2) from dual; --0

--求余

select mod(9,3) from dual; --0

select mod(9,4) from dual; --1

### 2、字符函数

-- substr(str1,起始索引,长度) 截取字符

--注意: 起始索引不管写 0 还是 1 都是从第一个字符开始截取

select substr('abcdefg',0,3) from dual; --abc

select substr('abcdefg',1,3) from dual; --abc

select substr('abcdefg',2,3) from dual; --bcd

--获取字符串长度 24 28

select length('abcdefg') from dual;

--去除字符左右两边的空格

select trim(' hello ') from dual;

--替换字符串

Select replace('hello','l','a') from dual;

### 3、日期函数

--查询今天的日期

select sysdate from dual;

--查询3个月后的今天的日期

select add\_months(sysdate,3) from dual;

--查询3天后的日期

select sysdate + 3 from dual;

--查询员工入职的天数

select sysdate - hiredate from emp;

select ceil(sysdate - hiredate) from emp;

--查询员工入职的周数

select (sysdate - hiredate)/7 from emp;

--查询员工入职的月数

select months\_between(sysdate,hiredate) from emp;

--查询员工入职的年份

select months\_between(sysdate,hiredate)/12 from emp;

### 4、转换函数

--字符转数值 to\_number(str) 鸡肋：看下面列子就知道是鸡肋了

select 100+'10' from dual; --110 默认已经帮我们转换

select 100 + to\_number('10') from dual; --110

--数值转字符 to\_char(sal,'$9,999.99')

select to\_char(sal,'$9,999.99') from emp;

select to\_char(sal,'L9,999.99') from emp;//L是代表本地符号

--日期转字符 to\_char()

select to\_char(sysdate,'yyyy-mm-dd hh:mi:ss') from dual;

select to\_char(sysdate,'yyyy-mm-dd hh24:mi:ss') from dual;

select to\_char(sysdate,'yyyy') from dual; --只想要年 --2017

select to\_char(sysdate,'d') from dual; --2 参数一个d 代表一个星期中第几天

select to\_char(sysdate,'dd') from dual; --10 参数两个d 代表一个月中的第几天

select to\_char(sysdate,'ddd') from dual; --100参数三个d 代表一年中的第几天

select to\_char(sysdate,'day') from dual; --星期一；当天日期

select to\_char(sysdate,'dy') from dual; --当前日期 星期的简写

--字符转日期

select to\_date('2017-04-10','yyyy-mm-dd') from dual;

--查询1981年 -- 1985年入职的员工信息

select \* from emp where hiredate between to\_date('1981','yyyy') and to\_date('1985','yyyy');

### 5、通用函数(nvl)

nvl(参数1,参数2) 如果参数1 = null 就返回参数2

nvl2(参数1,参数2,参数3) 如果参数1 = null ,就返回参数3, 否则返回参数2

select nvl2(null,5,6) from dual; --6;

select nvl2(1,5,6) from dual; --5;

nullif(参数1,参数2) 如果参数1 = 参数2 那么就返回 null , 否则返回参数1

select nullif(5,6) from dual; --5

select nullif(6,6) from dual; --null

coalesce: 返回第一个不为null的值

select coalesce(null,null,3,5,6) from dual; --3

## 多行函数

多行函数: 对某一列的所有行进行处理

1、max()

2、min

3、count

4、sum

5、avg

例子

--统计员工工资总和

select sum(sal) from emp;

--统计员工奖金总和 2200

select sum(comm) from emp;

--统计员工人数 14

select count(1) from emp;

--统计员工的平均奖金 550 错误 2200/14 =

select avg(comm) from emp; // comm字段有空值

--统计员工的平均奖金 157.

select sum(comm)/count(1) from emp;

select ceil(sum(comm)/count(1)) from emp;

# plsql编程

## 基础

Oracle特有的，PLSQL编程 : procedure Language 过程语言 Oracle对SQL的一个扩展

让我们能够像在java中一样写 if else else if 条件, 还可以编写循环逻辑 for while

语法

declare

--声明变量

变量名 变量类型;

变量名 变量类型 := 初始值;

vsal emp.sal%type; --引用型的变量

vrow emp%rowtype; --声明记录型变量

begin

--业务逻辑

end;

dbms\_output.put\_line()相当于java中 System.out.pringln();

初级例子

--最简单例子：声明变量 i，打印

declare

i varchar2(10) := '张三'; --声明变量

begin

dbms\_output.put\_line(i); --打印

end;

--查询7369的工资,并且打印出来 例子：声明引用型的变量：查单个字段

declare

vsal emp.sal%type; --声明引用型的变量 vsal 让emp中的sal类型赋予vsal

begin

--将查询出的结果赋值给vsal

select sal into vsal from emp where empno = 7369;

dbms\_output.put\_line(vsal);

end;

--查询scott账号的表emp 7369的员工信息, 并且打印出来 --声明记录型变量，查询出来一行记录，必须一条

declare

vrow emp%rowtype; --声明记录型变量 vsal 让emp中的sal类型赋予vsal

begin

select \* into vrow from emp where empno = 7369; --查询出一条记录

dbms\_output.put\_line('姓名:'||vrow.ename || '工资'|| vrow.sal); --可以输出一行所有字段记录

end;

### 变量

使用范围

declare

用于此之间

begin

end;

定义变量：3种方式

最基本定义变量方式

结构：变量名 类型； //和java相反

a Number :=10; //定义一个类型a变量，类型是number,值是10

引用型变量(字段类型) --🡪意思是引用表中的字段类型

结构：变量名 表名.字段名%type //这个和java引用类型概念不一样，它的意思是获取指定表中这个字段的类型

My\_ename emp.ename%type; -- 定义变量名为My\_ename ，它的类型是emp表中的字段中ename的类型

记录型变量(表中的一行类型) --🡪 可以认为java的数组把记录表中一行的字段类型

结构：变量名 表名%rowtype //引用指定表一行的类型(其实就是所有字段的类型)

My\_emp emp%rowtype; --定义变量名为my\_emp,它的类型是emp中所有字段类型，用于获取emp查询说的一行数据

### If语句

语法

if 条件语句 then

elsif条件语句then

else

end if;

使用

--根据不同年纪,输出相关内容

declare

age number := 20;

begin

if age < 18 then

dbms\_output.put\_line('小屁孩');

elsif age>=18 and age <=24 then

dbms\_output.put\_line('年轻人');

elsif age>24 and age < 40 then

dbms\_output.put\_line('老司机');

else

dbms\_output.put\_line('老年人');

end if;

end;

### 循环

有3种方式

loop循环

loop

exit when 条件

end loop;

例子

--输出1~10 例子3

declare

i number :=1;

begin

loop

exit when i>10;

dbms\_output.put\_line(i);

i := i+1;

end loop;

end;

while 循环 跟java的while几乎一样

while 条件 loop

end loop;

例子

--输出1~10 例子1

declare

i number :=1;

begin

while i<=10 loop

dbms\_output.put\_line(i);

i := i+1;

end loop;

end;

for循环 跟java 的for循环几乎一样

for 变量 in reverse 1..3 起始值..结束值 loop

end loop;

例子

--输出1~10

declare

begin

for i in reverse 1..10 loop

dbms\_output.put\_line(i);

end loop;

end;

--使用PLSQL输出菱形

declare

m number := 10;

begin

for x in -m..m loop

for y in -m..m loop

if abs(y) + abs(x) <= m then

dbms\_output.put('\*');

else

dbms\_output.put(' ');

end if;

end loop;

dbms\_output.new\_line();

end loop;

end;

--使用PLSQL输出三角形,只要是三个角

declare

m number := 10;

begin

for x in reverse -m..m loop

for y in -m..m loop

if abs(y) + abs(x) <= m and x>=0 then

dbms\_output.put('\*');

else

dbms\_output.put(' ');

end if;

end loop;

dbms\_output.new\_line();

end loop;

end;

## 例外(异常)

异常时程序设计语言提供的一种功能，用来增强程序的健壮性和容错性。

使用步骤 语法

Exception ---关键字exception

when 异常变量名1 when 异常名字 then 捕获后处理流程 when相当于java的 ‘（’，then相当于java的 ’）’

then 处理流程 --捕获后的异常进行打印处理

when 异常变量名2 --可以和java一样可以再进行捕获异常

then 处理流程

### 系统例外

No\_data\_found（没有找到数据）

Too\_many\_rows （select …into 语句匹配多个行）

Zero\_divide （被零除）

Value\_error （算术或转换错误）

Timeout\_on\_resource （在等待资源时发生超时）

Others 其它例外，相当于捕获所有异常

示例

----算术错误：除以0

declare

n number(20); --设置变量

begin

n :=9/0; --错误根源，跟java一样一旦出错，捕获异常跳转到异常

--例外操作开始

exception ---关键字exception

when zero\_divide when 异常名字 then 捕获后处理流程 when相当于java的 ‘（’，then相当于java的 ’）’

then dbms\_output.put\_line('测试零除'); --捕获后的异常进行打印处理

when others --其它例外

then dbms\_output.put\_line('其它例外');

end;

### 自定义例外

语法

Declare

自定义异常名 exception; --定义异常变量

Begin

raise 自定义异常名; --抛出例外

exception --处理异常

when 自定义异常名then 处理流程; --捕获异常

when others then dbms\_output.put\_line('其它例外');

end;

End;

自定义

--查询xxx号部门的员工，没有此部门就进入例外流程进行处理

declare

cursor cemp(bumen number) is select ename from emp where deptno= bumen; --定义游标

pename emp.ename%type; --定义引用类型

no\_emp\_found exception; --自定义例外变量

begin

open cemp(50);

fetch cemp into pename; ---获取50号部门的所有员工(表里没有50号部门)

if cemp%notfound then --使用if判断游标如果是notfound ，进行抛出例外

raise no\_emp\_found; -- --抛出例外 关键字raise

end if;

close cemp; --这里游标关闭不了了，是没有问题的，抛出例外 有Pmon进程会进行关闭操作

exception

when no\_emp\_found then dbms\_output.put\_line('没有找到员工');

when others then dbms\_output.put\_line('触发其它例外');

end;

## 游标(cursor)

游标(光标): 是用来操作查询结果集,相当于是JDBC中ResultSet(获取多行数据)

### 自定义游标

显式游标

语法

开发步骤:

1. 声明游标 语法: cursor 游标名[(参数名 参数类型)] is 查询结果集

2. 打开游标 关键字open 游标名

3. 从游标中取数据 fetch 游标名 into 变量；fetch关键字是获取下一行的数据和jdbc的next()方法一样它的底层就是使用游标

4.判断数据 4-1、游标名%found :找到数据

4-2、游标名%notfound : 没有找到数据

4. 关闭游标 close 游标名

无参数自定义游标

declare

cursor a is select \* from emp; -- 自定义声明游标:使用关键字cursor 游标名 is

b emp%rowtype; --声明行变量 用于存储提取游标的数据

begin

open a; --打开游标

loop --从游标中提取数据放入行数据变量中

fetch a into b; --fetch fetch关键字是获取下一行的数据，和jdbc的next()方法一样它的底层就是使用游标

exit when a%notfound; --判断游标a是空跳出loop循环

dbms\_output.put\_line(b.ename);

end loop;

close a; --关闭游标

end;

带参数自定义游标

declare

cursor a(c number) is select \* from emp where empno=c; --声明带参数的游标

b emp%rowtype; --声明行数据变量

begin

open a(7788); --打开游标:这里带参数

loop

fetch a into b; --从游标中提取数据放入行数据变量中

exit when a%notfound;

dbms\_output.put\_line(b.ename);

end loop;

close a;

end;

隐式游标

语法

for循环遍历游标:

不需要声明额外变量

不需要打开游标

不需要关闭游标

使用循环 for 存储变量 in 游标 loop…..end loop

示例

--使用游标获取所有用户的信息 ---无参数

declare

cursor aa is select \* from emp;

begin

for a in aa loop

dbms\_output.put\_line('姓名:'||a.ename ||' 工资: ' || a.sal || '工作:'|| a.job);

end loop;

end;

--使用游标获取指定员工的信息 --有参数

declare

cursor aa(n number) is select \* from emp where empno=n;

begin

for a in aa(7788) loop

dbms\_output.put\_line('姓名:'||a.ename ||' 工资: ' || a.sal || '工作:'|| a.job);

end loop;

end;

### 使用系统游标

declare

a sys\_refcursor; --声明系统游标

b emp%rowtype; --声明行变量 存储游标的数据

begin

open a for select \* from emp; --打开游标

loop

fetch a into b; --从游标中提取数据放入行数据变量中

exit when a%notfound;

dbms\_output.put\_line(b.ename);

end loop;

close a;

end;

## 存储过程|函数

查询此表空间有多少函数和过程

select \* from user\_triggers

### 存储过程

存储过程: 实际上是封装在服务器上一段PLSQL代码片断,(自定义的写的内容)已经编译好了的代码

1.客户端取调用存储过程,执行效率就会非常高效

语法:

create [or replace] procedure 存储过程的名称(参数名 in|out 参数类型,参数名 in|out 参数类型)

is | as

--声明变量部分

begin

--业务逻辑

end;

调用 看使用步骤

语法说明：

1、如果带or replace 覆盖原有的相同的存储过程名字，(建议带上)

2、创建存储过程的关键字为procedure。

3、传参列表中的参数可以用in，out，in out修饰，参数类型一定不能写大小。列表中可以有多个输入输出参数。

4、As和is没区别，没有参数不要写上()

5.调用带输出参数的过程必须要声明变量来接收输出参数值。

6.执行存储过程有两种方式，一种是使用call，另一种是用begin和end包住。

存储过程虽然有很多优点，但是它却不能使用return返回值。当需要使用return返回值时，我们可以使用函数。

Create or replace procedure 如果没有写的话，执行编译好的不正确语句，改好后再执行会爆错已有相同的名字，然后又会不知道哪里找此名字进行删除

Create or replace procedure(fanhuizhi out number) 返回值也是写在这个括号里：返回值名称 out 参数类型

is | as 在这里是没有任何区别的，随便写

使用步骤

存储过程功能： --根据传入的编号，返回员工姓名

第一步 编写存储过程

create or replace procedure ceshi(bianhao number, name out varchar2) -- 参数1：需要传参number类型的编号，参数二：返回结果出去的参数

is

begin

select ename into name from emp where empno=bianhao; - 查询出结果后：返回结果使用oracle语法赋值到name中即可

end; --注意需要带分号

第二步 执行完后，在pl/sql工具中就会有一个存储过程的名字

C:\Users\WWL\AppData\Local\Microsoft\Windows\INetCache\Content.Word\30.png

注意：如果没写or replace执行语法不正确的不会这里出现，再执行会报已有此函数的名称

第三步 调用此存储过程方法

第一种方式：貌似只能调用没有返回值的，没有深究

call ceshi(); 关键字 call

第二种方式 是常用的

Declare --这里不一样使用的是declare ，创建存储过程使用的是 is或as

vv varchar2(500); -- 声明一个变量 如果是字符需要指定长度 ，number数字貌似不用

begin

ceshi(7788,vv); --自己写的存储函数方法：ceshi：第一参数编号，第二个需要变量名字传入

dbms\_output.put\_line(vv); -- 打印结果

end;

其它例子

--给指定员工涨薪,并打印涨薪前和涨薪后的工资

create or replace procedure proc\_updatesal(vempno in number,vnum in number) – 参数：员工编号 参数2 需要加工资多少

is

vsal number; --声明变量.记录当前工资

begin

select sal into vsal from emp where empno = vempno; --查询当前的工资 传入员工编号，把查询结果赋值到声明的变量中

dbms\_output.put\_line('涨薪前:'||vsal); --输出涨薪前的工资

update emp set sal = vsal + vnum where empno = vempno; --更新工资 传入的vnum+vsal

dbms\_output.put\_line('涨薪后:'||(vsal+vnum)); --输出涨薪后的工资

commit; --有更新操作需要提交

end;

示例一：无参无返

创建存储过程语句

create or replace procedure p1

--or replace代表创建该存储过程时，若存储名存在，则替换原存储过程，重新创建

--无参数列表时，不需要写()

as

begin

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

end;

调用过程

--执行存储过程方式1 没有参数可以省略声明变量的关键字：declare

begin

p1();

end;

--执行存储过程方式2

call p1();

示例二：有参有返

创建存储过程语句

create or replace procedure p2(name in varchar2,age int,msg out varchar2)

--参数列表中，声明变量类型时切记不能写大小，只写类型名即可，例如参数列表中的name变量的声明

--参数列表中，输入参数用in表示，输出参数用out表示，不写时默认为输入参数。

------------输入参数不能携带值出去，输出参数不能携带值进来，当既想携带值进来，又想携带值出去，可以用in out

as

begin

msg:='姓名'||name||',年龄'||age;

--赋值时除了可以使用：=，还可以用into来实现

--上面子句等价于select '姓名'||name||',年龄'||age into msg from dual;

end;

调用过程：带参只有一种调用方式

declare

msg varchar2(100);

begin

p2('张三',23,msg);

dbms\_output.put\_line(msg);

end;

示例三：参数列表中有in out参数

创建存储过程语句

create or replace procedure p3(msg in out varchar2)

--当既想携带值进来，又想携带值出去，可以用in out

as

begin

dbms\_output.put\_line(msg); --输出的为携带进来的值

msg:='我是从存储过程中携带出来的值';

end;

调用过程：带参只有一种调用方式

--执行存储过程

declare

msg varchar2(100):='我是从携带进去的值';

begin

p3(msg);

dbms\_output.put\_line(msg); //打印：我是从存储过程中携带出来的值

end;

示例四：存储过程中定义参数

创建存储过程语句

create or replace procedure p4

as

--存储过程中定义的参数列表

name varchar(50);

begin

name := 'hello world';

dbms\_output.put\_line(name);

end;

调用过程：

---执行存储过程

Call p4();

示例五：获取多行数据：使用游标

创建存储过程语句

create or replace procedure aa(youbiao out SYS\_REFCURSOR) -- 系统游标类型

is

begin

OPEN youbiao FOR -- 这里定义参数即可

select \* from emp;

end;

调用过程

declare

vv SYS\_REFCURSOR; --声明游标变量

emps emp%ROWTYPE; --定义一个结构和函数中aa表一样的记录类型变量

begin

aa(vv); --使用自定义存储过程方法aa(vv)

loop --使用loop进行循环

FETCH vv INTO emps; --从游标中获取数据

dbms\_output.put\_line(emps.ename); --获取数据

EXIT WHEN vv%NOTFOUND; --如果vv为null 结束循环

end loop;

close vv; --关闭进程

end;

#### java调用存储过程

示例1 调用游标(多行数据)操作

调用的存储过程方法

create or replace procedure aa(youbiao out SYS\_REFCURSOR) -- 系统游标类型

is

begin

OPEN youbiao FOR -- 这里定义参数即可

select \* from emp;

end;

Java的jdbc操作

连接数据库操作

Class.forName("oracle.jdbc.driver.OracleDriver");

Connection connection=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:orcl", "scott", "123456");

这里开始和普通的获取使用类不一样了：使用call方式调用存储过程方法

CallableStatement call = connection.prepareCall("call aa(?)"); //调用存储过程aa的方法

//此存储过程aa只有输出类型，所以进行设置(注册)输出参数第一个，还有类型

call.registerOutParameter(1, OracleTypes.CURSOR);

//执行存储过程语句

call.execute();

//获取结果：这里因为上面注册的输出类型是第一个参数，这里也就是第一个

//另一个方式:使用oracle的驱动中的类获取集合

//ResultSet r = ((OracleCallableStatement)call).getCursor(1);

ResultSet r = (ResultSet)call.getObject(1); //因为是集合数据可以强制转成ResultSet

while(r.next()) {

System.out.println(r.getObject(1)+":"+r.getObject(2)+":"+r.getObject(3)+":"+r.getObject(4)+":"+r.getObject(5)+":"+r.getObject(6)+":"+r.getObject(7)+":"+r.getObject(8));

}

示例2 调用存储过程参数有输入类型和输出类型的方法

调用的存储过程方法

根据传入的编号，获取姓名

create or replace procedure bb(bianhao number,e out varchar2)

is

begin

select ename into e from emp where empno=bianhao;

end;

Java的jdbc操作

连接数据库操作

Class.forName("oracle.jdbc.driver.OracleDriver");

Connection connection=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:orcl", "scott", "123456");

CallableStatement p = connection.prepareCall("call bb(?,?)");

p.setInt(1, 7788); --设置参数

p.registerOutParameter(2, OracleTypes.VARCHAR);--注册输出参数

p.execute();

String str = p.getString(2); //获取结果：这里因为上面注册的输出类型是第二个参数，这里也就是第二个

System.out.println(str);

### 存储函数

存储函数: 自定义存储函数封装到oracle的一段PLSQL代码片断

函数与存储过程的结构类似，但是函数必须有一个return子句，用于返回函数值

存储函数只能返回一个结果值

语法

1. 和存储过程不一样的地址后缀带return 参数类型
2. 关键字是function

语法:

create [or replace] function 存储函数的名称(参数名 in|out 参数类型,参数名 in|out 参数类型) return 参数类型

is | as

begin

end;

使用步骤

第一步 编写存储函数

create or replace function dd return varchar2 --必须有返回值，且声明返回值类型时不需要加大小

as

msg varchar(50);

begin

msg := 'hello world';

return msg; --必须要return 也就是返回值

end;

第二步 执行完后，在pl/sql工具中就会有一个存储函数的名字

C:\Users\WWL\AppData\Local\Microsoft\Windows\INetCache\Content.Word\52.png

第三步 调用此存储过程方法

--执行函数方式1

select dd() from dual;

--执行函数方式2

begin

dbms\_output.put\_line(dd());

end;

示例1：带输入参数

创建存储函数语句

create or replace function aa(bianhao number)return varchar2 --返回类型不能声明大小

as

vv varchar2(200);

begin

select ename into vv from emp t where t.empno=bianhao;

return vv;

end;

调用操作

declare

e varchar2(200);

begin

e:=aa(7788); -- 使用 变量接受，赋值符号是：:= 不是Java的 =

dbms\_output.put\_line(e);

end;

#### java调用存储函数

其实和存储过程都差不多：这里举个集合例子

获取所有员工

存储函数方法

create or replace function ceshi4 return sys\_refcursor

is

--定义系统游标游标

aaa sys\_refcursor;

begin

open aaa for select \* from emp ;

return aaa;

close aaa;

end ceshi4;

jdbc调用

……

String sql="{?= call ceshi4()}"; //为啥要大括号呢?oracle 的两个调用语句call和exec区别

CallableStatement call = connection.prepareCall(sql);

call.registerOutParameter(1, OracleTypes.CURSOR); //类型是CURSOR【光标】

call.execute();

//另一个方式:使用oracle的驱动中的类

ResultSet object = ((OracleCallableStatement)call).getCursor(1);

ResultSet object = (ResultSet) call.getObject(1); //强转成resultset

while(object.next()) {

System.out.println(object.getObject(1));

}

### 存储过程和函数的区别

其实函数已经废了,自从oracle升级版本后有了out参数 就已经没啥用了

49期老师告知的区别

1.它们本质上没有区别

2.函数存在的意义是给过程调用 存储过程里面调用存储函数

3.函数可以在sql语句里面直接调用

Select \* from emp where ename=cc return varchar2(500)

4.存储过程能实现的,存储函数也能实现,存储函数能实现的,过程也能实现

网上找到的区别1

使用步骤

1. 返回值的区别：函数有1个返回值,而存储过程是通过参数返回的,可以有多个或者没有

2.调用的区别：函数可以在查询语句中直接调用,而存储过程必须单独调用.

3.使用场景：

函数一般情况下是用来计算并返回一个计算结果

存储过程一般是用来完成特定的数据操作（比如修改、插入数据库表或执行某些DDL语句等等）

网上找到的区别2

相同点：

1.创建语法结构相似，都可以携带多个传入参数和传出参数。

2.都是一次编译，多次执行。

不同点：

1.存储过程定义关键字用procedure，函数定义用function。

2.存储过程中不能用return返回值，但函数中可以，而且函数中必须有return子句。

3.执行方式略有不同，存储过程的执行方式有两种（1.使用call 2.使用begin和end），函数除了存储过程的两种方式外，还可以当做表达式使用，例如放在select中（select f1() form dual;）。

总结：如果只有一个返回值，用存储函数，否则，一般用存储过程。

看图



## 触发器

触发器相当于java的监听器

语法

Create [or replace] trigger 触发器名

{before |after} -- before执行前进行操作，after执行完以后进行触发

{delete|insert|update[of 列名]} --没有select的，只有增删改三个语句触发，修改可以指定到列名触发

On 表名 --那张表

[for each row[when 条件]] -- 触发器的类型 ：没有此语句是：语句级触发器，for each row是行级触发器，表级触发器

Plsql块 -- declare begin end;

End 触发器名;

触发器类型

语句触发(针对表)： 默认是语句触发器，在on表名 后写，没有写就是语句触发器

在指定的操作语句操作之前或之后执行一次，不管这条语句影响了多少行

行级触发器 (for each row)：

触发语句作用的每条记录都被触发，在行级触发器中使用:old和:new伪记录变量，识别值得状态

:old 字段旧数据

:new 字段新数据

使用步骤

例子 插入员工表数据前打印一句话

第一步 创建触发器语句

create or replace trigger insertceshi --关键字trigger

before --执行前关键字before

insert --只对插入触发

on emp --针对的表是emp

begin --plsql编程块

dbms\_output.put\_line('进行插入数据。。。。。。。');

end;

第二步 执行成功后在：plsql工具中有，或者使用查询语句

查询语句：select \* from user\_triggers

Plsql工具中可以查看：



第三步：测试

进行插入操作：insert into emp(empno,ename) values('99999','www');

![C:\Users\Administrator\AppData\Roaming\Tencent\Users\531688079\QQ\WinTemp\RichOle\US_8VDX$[~75](U45V}UY`I.png](data:image/png;base64,)

### 作用

1数据确认

2实施复杂的安全性检查

语句触发(针对表)

案例：禁止非工作时间(星期六、星期天)插入数据

create or replace trigger xianzhicahru

before insert --插入触发

on emp --触发表：emp

begin

if to\_char(sysdate,'day') in('星期六','星期天') or -- to\_char(sysdate,'day')是获得今天星期几，判断今天是星期几，

to\_number(to\_char(sysdate,'hh24')) not between 9 and 17 then --时间如果不是在9点到17点的话触发 ，to\_number(to\_char(sysdate,'hh24'))当前整点的数字

raise\_application\_error(-20001,'禁止在非工作时间进行添加新员工'); --处理方式是抛出应用层的错误使用系统定义好的函数(参数一取值范围-20000到-20999之间，参数二：字符串)

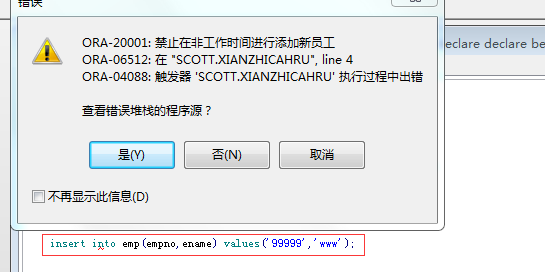
end if;

end xianzhicahru;

执行成功后会在plsql工具中显示



测试 ：在星期六的时间进行插入操作就进行触发



2、行级触发器案例：数据的确认

涨薪之前检查工资：不能越来越少

3做审计，跟踪表上所做的数据操作等

审计可以认为就是日志，用于记录什么人操作了这张表，插入还是修改，记录到一张表中

强制审计

标准审计(配置)

基于值得审计

细粒度审计

管理员审计

4数据的备份和同步

create table emp2 as select \* from emp where 1=2; 复制表结构，要复制全部数据去掉where 1=2

# SQL调优

查询语句是否更好

1.运行完sql以后 使用sql/pl工具查看执行计划F5，也有sql语句操作：自己百度吧：“sql执行计划”

2. 分析里面的cost 和 影响行数, 想办法降低