**1、本章面试题**

     什么是游标，游标如何使用（显式游标）

什么是触发器，如何使用

**2、知识点**

**2.1、课程回顾**

语句块： 函数function, 存储过程procedure, 包 package

**2.2、本章重点**

游标（概念，属性，分类，隐含游标示例，显式游标示例）

触发器（概念，语法，示例）

**3、具体内容**

**3.1 游标**

**3.1.1 概念**

游标是由select语句产生的结果集（数据库表的副本）,还可以充当指针的作用，一次指向结果集中的一条数据，使用它可以对业务功能中，处理一个结果集有帮助。

**3.1.2 游标属性**

游标名称%的格式调用

**%isopen**

判断游标是否打开，打开时返回true 否则返回false

**%found**

检查游标是否提取到了数据，提取到返回true 否则false

**%notfound**

与%found相反， 检查游标是否提取到了数据，提取到返回false 否则true

**%rowcount**

获取当前游标已经执行到实际行数

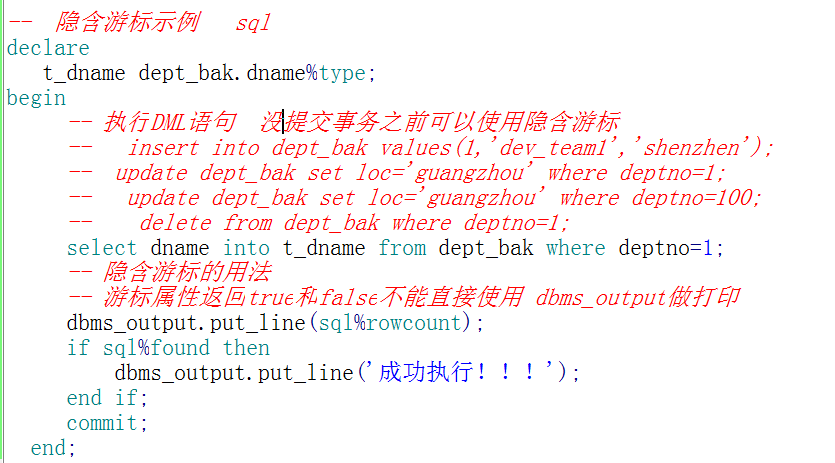
**3.1.3 游标分类**

**隐含游标**,不需要自定义，当执行DML(insert update delete操作)或者select

into操作时，系统会默认调用sql游标，对操作数据的结果会有记录。

**显式游标,**需要自定义，是select产生一个结果集

**3.1.4 隐含游标示例**



**3.1.5 显式游标的示例**

显式游标实用步骤：

定义游标 打开游标 提取数据 关闭游标

**1,示例1 （** 使用游标提取部门表数据，并且打印结果）

declare

-- 定义游标 定义部分（变量 常量constant 自定义异常 游标）

cursor csr\_dept is select \* from dept;

-- 定义部门表的行变量，用于接受fetch结果

t\_dept\_row dept%rowtype;

begin

-- 打开游标

open csr\_dept;

-- 提取数据

-- 循环

loop

fetch csr\_dept into t\_dept\_row; -- fetch一次获取一条数据

-- 判断是否还有数据，方便退出循环

if csr\_dept%notfound then -- %notfound 没有数据返回true

exit; -- 退出 相当于java中break

end if;

-- 更简单的写法

-- exit when csr\_dept%notfound;

-- 打印

dbms\_output.put\_line('部门名称：'||t\_dept\_row.dname

||'位置：'||t\_dept\_row.loc );

end loop;

-- 关闭游标

close csr\_dept;

end;

**2, 示例2 （bulk collect 用法）**

-- bulk collect 一次性取出游标数据，不用循环一条条取出，提高执行效率

-- 使用bulk collect 提取员工表数据（复习table用法）

declare

-- 定义游标

cursor csr\_emp is select \* from emp;

-- 定义table类型 Class table\_emp

type table\_emp is table of emp%rowtype index by binary\_integer;

-- 定义table变量用来接受一次性取出的数据 table\_emp t\_table\_emp = new table\_emp();

t\_table\_emp table\_emp;

begin

-- 打开游标

open csr\_emp;

-- 提取数据(使用bulk collect 一次性取出)

fetch csr\_emp bulk collect into t\_table\_emp;

-- 关闭游标

close csr\_emp;

-- 打印table变量的数据 first开始值 last 结束值 count 总长度

-- 使用for循环 1,loop end loop 2, while 条件 loop end loop 3,for in loop end loop

dbms\_output.put\_line('first:'||t\_table\_emp.first||',last:'||t\_table\_emp.last

||',count:'||t\_table\_emp.count);

for t\_i in t\_table\_emp.first..t\_table\_emp.last loop

dbms\_output.put\_line('姓名：'||t\_table\_emp(t\_i).ename||',工资：'||

t\_table\_emp(t\_i).sal||',雇佣日期：'||

to\_char(t\_table\_emp(t\_i).hiredate,'yyyy-mm-dd'));

end loop;

end;

**3, 示例3(参数游标)**

-- 根据部门编号查询员工,并打印

declare

-- 定义游标

cursor csr\_emp(v\_deptno number) is select \* from emp where deptno=v\_deptno;

-- 定义行变量

t\_emp\_row emp%rowtype;

begin

-- 打开游标

open csr\_emp(&deptno);

-- 提取数据

loop

fetch csr\_emp into t\_emp\_row;

--退出

exit when csr\_emp%notfound;

-- 打印

dbms\_output.put\_line('姓名：'||t\_emp\_row.ename||',工资：'||

t\_emp\_row.sal||',雇佣日期：'||to\_char(t\_emp\_row.hiredate,'yyyy-mm-dd'));

end loop;

-- 关闭游标

close csr\_emp;

end;

**4，示例4** （游标变量一次定义多次使用）

-- 定义一个游标，既可以取部门表数据，又可以取员工表数据

declare

-- 定义游标（定义时不指定集合）

-- 定义游标类型

type type\_csr is ref cursor;

-- 定义变量属于上面类型

t\_type\_csr type\_csr;

-- 定义部门行数据变量

t\_dept\_row dept%rowtype;

t\_emp\_row emp%rowtype;

begin

dbms\_output.put\_line('---------第一次使用---------------------');

-- 打开游标(打开游标时使用for指定集合)

open t\_type\_csr for select \* from dept;

-- 提取数据

loop

fetch t\_type\_csr into t\_dept\_row;

exit when t\_type\_csr%notfound;

dbms\_output.put\_line('部门名称'||t\_dept\_row.dname

||',位置:'||t\_dept\_row.loc );

end loop;

-- 关闭游标

close t\_type\_csr;

----------------------- 再次使用

dbms\_output.put\_line('---------再次使用---------------------');

-- 打开游标 让同一个游标变量指向另外一个集合

open t\_type\_csr for select \* from emp;

-- 提取数据

loop

fetch t\_type\_csr into t\_emp\_row;

--退出

exit when t\_type\_csr%notfound;

-- 打印

dbms\_output.put\_line('姓名：'||t\_emp\_row.ename||',工资：'||

t\_emp\_row.sal||',雇佣日期：'||to\_char(t\_emp\_row.hiredate,'yyyy-mm-dd'));

end loop;

-- 关闭游标

close t\_type\_csr;

end;

**5 示例5(简化游标)**

-- 示例5(使用for简化游标)

declare

-- 定义游标

cursor csr\_dept is select \* from dept;

begin

-- 省去了打开和关闭的过程

for t\_dept\_row in csr\_dept loop

dbms\_output.put\_line('部门名称'||t\_dept\_row.dname

||',位置:'||t\_dept\_row.loc);

end loop;

end;

declare

-- 定义游标

--cursor csr\_dept is select \* from dept;

begin

-- 省去了打开和关闭的过程

for t\_dept\_row in (select \* from dept) loop

dbms\_output.put\_line('部门名称'||t\_dept\_row.dname

||',位置:'||t\_dept\_row.loc);

end loop;

end;

**3.2 触发器**

**3.2.1 概念**

触发是在事件发生时(DML或者登陆/退出数据库)自动的调用语句块（执行的业务）的一中oracle数据库对象.

**3.2.2 类型**

DML触发器（进行insert update delete 操作时，隐含自动的执行语句块）

系统触发器（当登陆或者退出数据库时，隐含自动的执行语句块）

**3.2.3 语法**

create [or replace] trigger 触发器名称

before/after 触发时间（在事件之前或者之后操作）

insert/update/delete 触发事件(DML操作，多个操作时可以使用逗号隔开)

on 表名 触发对象（数据库表）

[for each row] 行级触发器时使用

begin

--执行业务

end;

    **RAISE\_APPLICATION\_ERROR用法：**

      RAISE\_APPLICATION\_ERROR(errorNumber,errorString)

             errorNumber是数值在-20000到-20999之间，errorString为自定义的错误信息。

    **3.2.5举例（周4,6不能修改emp表）**



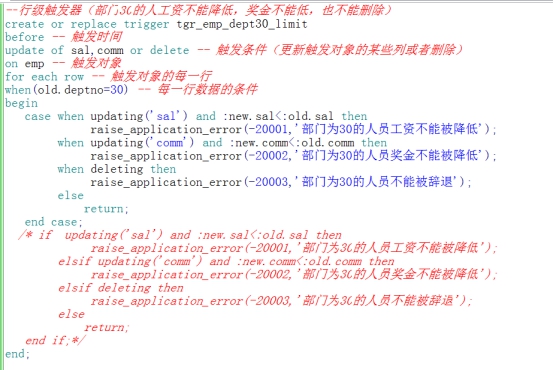
 --条件谓词(inserting, updating, deleting)  触发器触发时，可以监测到到底是执行的什么操作

-- 对部门表进行增删改时，加入触发器

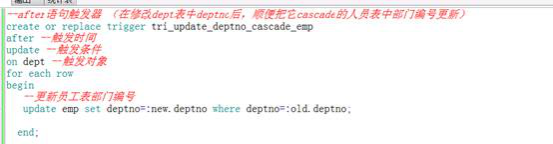


**--行级触发器（部门30的人工资不能降低，奖金不能低，也不能删除）**

--  for each row   :new修改后记录，:old修改前记录  before update of sal,comm

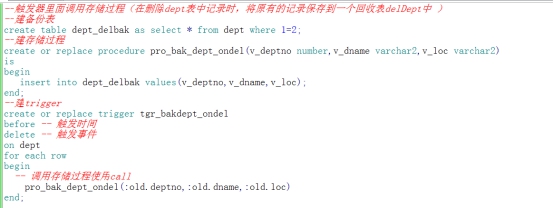


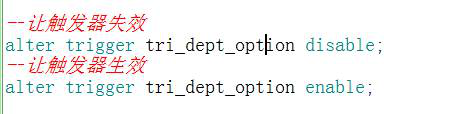
**--after语句触发器 （在修改dept表中deptno后，顺便把它cascade的人员表中部门编号更新）**



**--触发器里面调用存储过程（在删除dept表中记录时，将原有的记录保存到一个回收表dept\_bak中 ）**

建备份表，建存储过程，建trigger





**4、本章总结**

**4.1 总结本章知识点**

**4.2 面试题答案**

-- 全部删除表数据

delete from dept\_bak where deptno>10;

truncate table dept\_bak;

-- 都可以删除表数据

-- 1，delete 删除时可以加条件 truncat 不可以

-- 2，delete 删除时执行速度慢，删除数据时记录日志，使用delete误删数据

-- 在一定时间内可以通过快照找回，truncat 删除速度快 不记录日志，数据一般

-- 无法找回

-- 3，在mysql 使用delete删除数据后，自增会不会重置，使用truncate删除

--- 自增会重置（原来有3条数据，自增的id为3当删除后再次添加数据时，重置

-- 从1开始，不重置从4开始）

**4.3 预习下一章重点**