**一、触发器简介**

触发器的定义就是说某个条件成立的时候，触发器里面所定义的语句就会被自动的执行。

因此触发器不需要人为的去调用，也不能调用。

然后，触发器的触发条件其实在你定义的时候就已经设定好了。

这里面需要说明一下，触发器可以分为语句级触发器和行级触发器。

详细的介绍可以参考网上的资料，简单的说就是语句级的触发器可以在某些语句执行前或执行后被触发。而行级触发器则是在定义的了触发的表中的行数据改变时就会被触发一次。

具体举例：

1、 在一个表中定义的语句级的触发器，当这个表被删除时，程序就会自动执行触发器里面定义的操作过程。这个就是删除表的操作就是触发器执行的条件了。  
2、 在一个表中定义了行级的触发器，那当这个表中一行数据发生变化的时候，比如删除了一行记录，那触发器也会被自动执行了。

**二、触发器语法**

触发器的语法：

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | create [or replace] tigger 触发器名 触发时间 触发事件  on 表名  [for each row]  begin   pl/sql语句  end |

其中：

触发器名：触发器对象的名称。由于触发器是数据库自动执行的，因此该名称只是一个名称，没有实质的用途。  
触发时间：指明触发器何时执行，该值可取：  
before：表示在数据库动作之前触发器执行;  
after：表示在数据库动作之后触发器执行。  
触发事件：指明哪些数据库动作会触发此触发器：  
insert：数据库插入会触发此触发器;  
update：数据库修改会触发此触发器;  
delete：数据库删除会触发此触发器。  
表 名：数据库触发器所在的表。  
for each row：对表的每一行触发器执行一次。如果没有这一选项，则只对整个表执行一次。

触发器能实现如下功能：

**功能**：

1、 允许/限制对表的修改  
2、 自动生成派生列，比如自增字段  
3、 强制数据一致性  
4、 提供审计和日志记录  
5、 防止无效的事务处理  
6、 启用复杂的业务逻辑

举例

1)、下面的触发器在更新表tb\_emp之前触发，目的是不允许在周末修改表：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | create or replace trigger auth\_secure before insert or update or DELETE  on tb\_emp  begin    IF(to\_char(sysdate,'DY')='星期日') THEN      RAISE\_APPLICATION\_ERROR(-20600,'不能在周末修改表tb\_emp');    END IF;  END;  / |

2)、使用触发器实现序号自增

创建一个测试表：

|  |  |
| --- | --- |
| 1  2  3  4  5 | create table tab\_user(    id number(11) primary key,    username varchar(50),    password varchar(50)  ); |

创建一个序列：

复制代码 代码如下:

create sequence my\_seq increment by 1 start with 1 nomaxvalue nocycle cache 20;

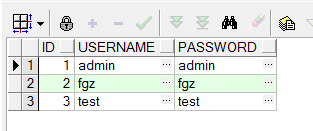
创建一个触发器：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | CREATE OR REPLACE TRIGGER MY\_TGR   BEFORE INSERT ON TAB\_USER   FOR EACH ROW--对表的每一行触发器执行一次  DECLARE   NEXT\_ID NUMBER;  BEGIN   SELECT MY\_SEQ.NEXTVAL INTO NEXT\_ID FROM DUAL;   :NEW.ID := NEXT\_ID; --:NEW表示新插入的那条记录  END; |

向表插入数据:

|  |  |
| --- | --- |
| 1  2  3  4 | insert into tab\_user(username,password) values('admin','admin');  insert into tab\_user(username,password) values('fgz','fgz');  insert into tab\_user(username,password) values('test','test');  COMMIT; |

查询表结果：SELECT \* FROM TAB\_USER;



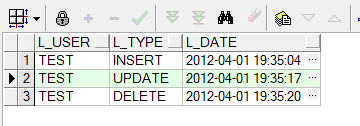
3)、当用户对test表执行DML语句时，将相关信息记录到日志表

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | --创建测试表  CREATE TABLE test(    t\_id  NUMBER(4),    t\_name VARCHAR2(20),    t\_age NUMBER(2),    t\_sex CHAR  );  --创建记录测试表  CREATE TABLE test\_log(    l\_user  VARCHAR2(15),    l\_type  VARCHAR2(15),    l\_date  VARCHAR2(30)  ); |

创建触发器：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31 | --创建触发器  CREATE OR REPLACE TRIGGER TEST\_TRIGGER   AFTER DELETE OR INSERT OR UPDATE ON TEST  DECLARE   V\_TYPE TEST\_LOG.L\_TYPE%TYPE;  BEGIN   IF INSERTING THEN    --INSERT触发    V\_TYPE := 'INSERT';    DBMS\_OUTPUT.PUT\_LINE('记录已经成功插入，并已记录到日志');   ELSIF UPDATING THEN    --UPDATE触发    V\_TYPE := 'UPDATE';    DBMS\_OUTPUT.PUT\_LINE('记录已经成功更新，并已记录到日志');   ELSIF DELETING THEN    --DELETE触发    V\_TYPE := 'DELETE';    DBMS\_OUTPUT.PUT\_LINE('记录已经成功删除，并已记录到日志');   END IF;   INSERT INTO TEST\_LOG   VALUES    (USER, V\_TYPE, TO\_CHAR(SYSDATE, 'yyyy-mm-dd hh24:mi:ss')); --USER表示当前用户名  END;  /  --下面我们来分别执行DML语句  INSERT INTO test VALUES(101,'zhao',22,'M');  UPDATE test SET t\_age = 30 WHERE t\_id = 101;  DELETE test WHERE t\_id = 101;  --然后查看效果  SELECT \* FROM test;  SELECT \* FROM test\_log; |

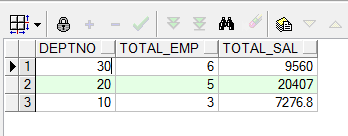
运行结果如下：



3)、创建触发器，它将映射emp表中每个部门的总人数和总工资

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26 | --创建映射表  CREATE TABLE dept\_sal AS  SELECT deptno, COUNT(empno) total\_emp, SUM(sal) total\_sal  FROM scott.emp  GROUP BY deptno;  --创建触发器  CREATE OR REPLACE TRIGGER EMP\_INFO   AFTER INSERT OR UPDATE OR DELETE ON scott.EMP  DECLARE   CURSOR CUR\_EMP IS    SELECT DEPTNO, COUNT(EMPNO) AS TOTAL\_EMP, SUM(SAL) AS TOTAL\_SAL FROM scott.EMP GROUP BY DEPTNO;  BEGIN   DELETE DEPT\_SAL; --触发时首先删除映射表信息   FOR V\_EMP IN CUR\_EMP LOOP    --DBMS\_OUTPUT.PUT\_LINE(v\_emp.deptno || v\_emp.total\_emp || v\_emp.total\_sal);    --插入数据    INSERT INTO DEPT\_SAL    VALUES     (V\_EMP.DEPTNO, V\_EMP.TOTAL\_EMP, V\_EMP.TOTAL\_SAL);   END LOOP;  END;  --对emp表进行DML操作  INSERT INTO emp(empno,deptno,sal) VALUES('123','10',10000);  SELECT \* FROM dept\_sal;  DELETE EMP WHERE empno=123;  SELECT \* FROM dept\_sal; |

显示结果如下：



4)、创建触发器，用来记录表的删除数据

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | --创建表  CREATE TABLE employee(    id  VARCHAR2(4) NOT NULL,    name VARCHAR2(15) NOT NULL,    age NUMBER(2)  NOT NULL,    sex CHAR NOT NULL  );  --插入数据  INSERT INTO employee VALUES('e101','zhao',23,'M');  INSERT INTO employee VALUES('e102','jian',21,'F');  --创建记录表(包含数据记录)  CREATE TABLE old\_employee AS SELECT \* FROM employee;  --创建触发器  CREATE OR REPLACE TRIGGER TIG\_OLD\_EMP   AFTER DELETE ON EMPLOYEE   FOR EACH ROW --语句级触发，即每一行触发一次  BEGIN   INSERT INTO OLD\_EMPLOYEE VALUES (:OLD.ID, :OLD.NAME, :OLD.AGE, :OLD.SEX); --:old代表旧值  END;  /  --下面进行测试  DELETE employee;  SELECT \* FROM old\_employee; |

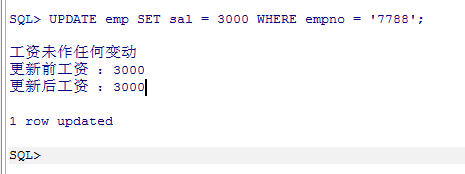
5)、创建触发器，利用视图插入数据

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | --创建表  CREATE TABLE tab1 (tid NUMBER(4) PRIMARY KEY,tname VARCHAR2(20),tage NUMBER(2));  CREATE TABLE tab2 (tid NUMBER(4),ttel VARCHAR2(15),tadr VARCHAR2(30));  --插入数据  INSERT INTO tab1 VALUES(101,'zhao',22);  INSERT INTO tab1 VALUES(102,'yang',20);  INSERT INTO tab2 VALUES(101,'13761512841','AnHuiSuZhou');  INSERT INTO tab2 VALUES(102,'13563258514','AnHuiSuZhou');  --创建视图连接两张表  CREATE OR REPLACE VIEW tab\_view AS SELECT tab1.tid,tname,ttel,tadr FROM tab1,tab2 WHERE tab1.tid = tab2.tid;  --创建触发器  CREATE OR REPLACE TRIGGER TAB\_TRIGGER   INSTEAD OF INSERT ON TAB\_VIEW  BEGIN   INSERT INTO TAB1 (TID, TNAME) VALUES (:NEW.TID, :NEW.TNAME);   INSERT INTO TAB2 (TTEL, TADR) VALUES (:NEW.TTEL, :NEW.TADR);  END;  /  --现在就可以利用视图插入数据  INSERT INTO tab\_view VALUES(106,'ljq','13886681288','beijing');  --查询  SELECT \* FROM tab\_view;  SELECT \* FROM tab1;  SELECT \* FROM tab2; |

6)、创建触发器，比较emp表中更新的工资

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | --创建触发器  set serveroutput on;  CREATE OR REPLACE TRIGGER SAL\_EMP   BEFORE UPDATE ON EMP   FOR EACH ROW  BEGIN   IF :OLD.SAL > :NEW.SAL THEN    DBMS\_OUTPUT.PUT\_LINE('工资减少');   ELSIF :OLD.SAL < :NEW.SAL THEN    DBMS\_OUTPUT.PUT\_LINE('工资增加');   ELSE    DBMS\_OUTPUT.PUT\_LINE('工资未作任何变动');   END IF;   DBMS\_OUTPUT.PUT\_LINE('更新前工资 ：' || :OLD.SAL);   DBMS\_OUTPUT.PUT\_LINE('更新后工资 ：' || :NEW.SAL);  END;  /  --执行UPDATE查看效果  UPDATE emp SET sal = 3000 WHERE empno = '7788'; |

运行结果如下：



7)、创建触发器，将操作CREATE、DROP存储在log\_info表

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36 | --创建表  CREATE TABLE log\_info(    manager\_user VARCHAR2(15),    manager\_date VARCHAR2(15),    manager\_type VARCHAR2(15),    obj\_name   VARCHAR2(15),    obj\_type   VARCHAR2(15)  );  --创建触发器  set serveroutput on;  CREATE OR REPLACE TRIGGER TRIG\_LOG\_INFO   AFTER CREATE OR DROP ON SCHEMA  BEGIN   INSERT INTO LOG\_INFO   VALUES    (USER,     SYSDATE,     SYS.DICTIONARY\_OBJ\_NAME,     SYS.DICTIONARY\_OBJ\_OWNER,     SYS.DICTIONARY\_OBJ\_TYPE);  END;  /  --测试语句  CREATE TABLE a(id NUMBER);  CREATE TYPE aa AS OBJECT(id NUMBER);  DROP TABLE a;  DROP TYPE aa;  --查看效果  SELECT \* FROM log\_info;  --相关数据字典-----------------------------------------------------  SELECT \* FROM USER\_TRIGGERS;  --必须以DBA身份登陆才能使用此数据字典  SELECT \* FROM ALL\_TRIGGERS;SELECT \* FROM DBA\_TRIGGERS;  --启用和禁用  ALTER TRIGGER trigger\_name DISABLE;  ALTER TRIGGER trigger\_name ENABLE; |