**Trigger summarize**

## 1 触发器类型

触发器组成:

        触发事件：引起触发器被触发的事件。 例如：DML语句(INSERT, UPDATE, DELETE语句对表或视图执行数据处理操作)、DDL语句（如CREATE、ALTER、DROP语句在数据库中创建、修改、删除模式对象）、数据库系统事件（如系统启动或退出、异常错误）、用户事件（如登录或退出数据库）。

        触发时间：即该TRIGGER 是在触发事件发生之前（BEFORE）还是之后(AFTER)触发，也就是触发事件和该TRIGGER的操作顺序。

        触发操作：即该TRIGGER 被触发之后的目的和意图，正是触发器本身要做的事情。 例如：PL/SQL块。

        触发对象：包括表、视图、模式、数据库。只有在这些对象上发生了符合触发条件的触发事件，才会执行触发操作。

        触发条件：由WHEN子句指定一个逻辑表达式。只有当该表达式的值为TRUE时，遇到触发事件才会自动执行触发器，使其执行触发操作。

        触发频率：说明触发器内定义的动作被执行的次数。即语句级(STATEMENT)触发器和行级(ROW)触发器。

语句级(STATEMENT)触发器：是指当某触发事件发生时，该触发器只执行一次；

行级(ROW)触发器：是指当某触发事件发生时，对受到该操作影响的每一行数据，触发器都单独执行一次。

**编写触发器时，需要注意以下几点：**

        触发器不接受参数。

        一个表上最多可有12个触发器，但同一时间、同一事件、同一类型的触发器只能有一个。并各触发器之间不能有矛盾。

        在一个表上的触发器越多，对在该表上的DML操作的性能影响就越大。

        触发器最大为32KB。若确实需要，可以先建立过程，然后在触发器中用CALL语句进行调用。

        **在触发器的执行部分只能用DML语句（SELECT、INSERT、UPDATE、DELETE），不能使用DDL语句（CREATE、ALTER、DROP）**。

        触发器中不能包含事务控制语句(COMMIT，ROLLBACK，SAVEPOINT)。因为触发器是触发语句的一部分，触发语句被提交、回退时，触发器也被提交、回退了。

        在触发器主体中调用的任何过程、函数，都不能使用事务控制语句。

        在触发器主体中不能申明任何Long和blob变量。新值new和旧值old也不能向表中的任何long和blob列。

        不同类型的触发器(如DML触发器、INSTEAD OF触发器、系统触发器)的语法格式和作用有较大区别。

**2 创建触发器**

**创建触发器的一般语法是:**

CREATE [OR REPLACE] TRIGGER trigger\_name  
{BEFORE | AFTER }  
{INSERT | DELETE | UPDATE [OF column [, column …]]}  
[OR {INSERT | DELETE | UPDATE [OF column [, column …]]}...]  
ON [schema.]table\_name | [schema.]view\_name   
[REFERENCING {OLD [AS] old | NEW [AS] new| PARENT as parent}]  
[FOR EACH ROW ]  
[WHEN condition]  
PL/SQL\_BLOCK | CALL procedure\_name;

其中：

BEFORE 和AFTER指出触发器的触发时序分别为前触发和后触发方式，前触发是在执行触发事件之前触发当前所创建的触发器，后触发是在执行触发事件之后触发当前所创建的触发器。

       FOR EACH ROW选项说明触发器为行触发器。行触发器和语句触发器的区别表现在：行触发器要求当一个DML语句操走影响数据库中的多行数据时，对于其中的每个数据行，只要它们符合触发约束条件，均激活一次触发器；而语句触发器将整个语句操作作为触发事件，当它符合约束条件时，激活一次触发器。当省略FOR EACH ROW 选项时，BEFORE 和AFTER 触发器为语句触发器，而INSTEAD OF 触发器则只能为行触发器。

            REFERENCING 子句说明相关名称，在行触发器的PL/SQL块和WHEN 子句中可以使用相关名称参照当前的新、旧列值，默认的相关名称分别为OLD和NEW。触发器的PL/SQL块中应用相关名称时，必须在它们之前加冒号(:)，但在WHEN子句中则不能加冒号。

WHEN 子句说明触发约束条件。Condition 为一个逻辑表达时，其中必须包含相关名称，而不能包含查询语句，也不能调用PL/SQL 函数。WHEN 子句指定的触发约束条件只能用在BEFORE 和AFTER 行触发器中，不能用在INSTEAD OF 行触发器和其它类型的触发器中。

    当一个基表被修改( INSERT, UPDATE, DELETE)时要执行的存储过程，执行时根据其所依附的基表改动而自动触发，因此与应用程序无关，用数据库触发器可以保证数据的一致性和完整性。

### 2.1 触发器触发次序

1.        执行 BEFORE语句级触发器;

2.        对与受语句影响的每一行：

         执行 BEFORE行级触发器

         执行 DML语句

         执行 AFTER行级触发器

3.        执行 AFTER语句级触发器

### 2.2 创建DML触发器

**DML触发器的限制**

         CREATE TRIGGER语句文本的字符长度不能超过32KB；

         触发器体内的SELECT 语句只能为SELECT … INTO …结构，或者为定义游标所使用的SELECT 语句。

         触发器中不能使用数据库事务控制语句 COMMIT; ROLLBACK, SVAEPOINT 语句；

         由触发器所调用的过程或函数也不能使用数据库事务控制语句；

         触发器中不能使用LONG, LONG RAW 类型；

         触发器内可以参照LOB 类型列的列值，但不能通过 :NEW 修改LOB列中的数据；

**DML触发器基本要点**

         **触发时机：**指定触发器的触发时间。如果指定为BEFORE，则表示在执行DML操作之前触发，以便防止某些错误操作发生或实现某些业务规则；如果指定为AFTER，则表示在执行DML操作之后触发，以便记录该操作或做某些事后处理。

         **触发事件：**引起触发器被触发的事件，即DML操作（INSERT、UPDATE、DELETE）。既可以是单个触发事件，也可以是多个触发事件的组合（只能使用OR逻辑组合，不能使用AND逻辑组合）。

         **条件谓词：**当在触发器中包含多个触发事件（INSERT、UPDATE、DELETE）的组合时，为了分别针对不同的事件进行不同的处理，需要使用ORACLE提供的如下条件谓词。

1）**INSERTING：**当触发事件是INSERT时，取值为TRUE，否则为FALSE。

2）**UPDATING [（column\_1,column\_2,…,column\_x）]：**当触发事件是UPDATE 时，如果修改了column\_x列，则取值为TRUE，否则为FALSE。其中column\_x是可选的。

3）**DELETING：**当触发事件是DELETE时，则取值为TRUE，否则为FALSE。

**解发对象：**指定触发器是创建在哪个表、视图上。

         **触发类型：**是语句级还是行级触发器。

         **触发条件：**由WHEN子句指定一个逻辑表达式，只允许在行级触发器上指定触发条件，指定UPDATING后面的列的列表。

问题：当触发器被触发时，要使用被插入、更新或删除的记录中的列值，有时要使用操作前、 后列的值.

实现:  :NEW 修饰符访问操作完成后列的值

         :OLD 修饰符访问操作完成前列的值

|  |  |  |  |
| --- | --- | --- | --- |
| 特性 | INSERT | UPDATE | DELETE |
| OLD | NULL | 实际值 | 实际值 |
| NEW | 实际值 | 实际值 | NULL |

### 2.3 创建替代(INSTEAD OF)触发器

**创建触发器的一般语法是:**

CREATE [OR REPLACE] TRIGGER trigger\_name  
INSTEAD OF  
{INSERT | DELETE | UPDATE [OF column [, column …]]}  
[OR {INSERT | DELETE | UPDATE [OF column [, column …]]}...]  
ON [schema.] view\_name --只能定义在视图上  
[REFERENCING {OLD [AS] old | NEW [AS] new| PARENT as parent}]  
[FOR EACH ROW ] --因为INSTEAD OF触发器只能在行级上触发,所以没有必要指定  
[WHEN condition]  
PL/SQL\_block | CALL procedure\_name;

其中：

            INSTEAD OF选项使ORACLE激活触发器，而不执行触发事件。只能对视图和对象视图建立INSTEAD OF触发器，而不能对表、模式和数据库建立INSTEAD OF触发器。

INSTEAD\_OF用于对视图的DML触发，由于视图有可能是由多个表进行联结(join)而成，因而并非是所有的联结都是可更新的。但可以按照所需的方式执行更新，

Example 1:

CREATE OR REPLACE VIEW emp\_view AS   
SELECT deptno, count(\*) total\_employeer, sum(sal) total\_salary   
FROM emp GROUP BY deptno;

CREATE OR REPLACE TRIGGER emp\_view\_delete  
   INSTEAD OF DELETE ON emp\_view FOR EACH ROW  
BEGIN  
   DELETE FROM emp WHERE deptno= :old.deptno;  
END emp\_view\_delete;   
DELETE FROM emp\_view WHERE deptno=**10**;   
**创建INSTEAD OF触发器需要注意以下几点：**

         只能被创建在视图上，并且该视图没有指定WITH CHECK OPTION选项。

         不能指定BEFORE或AFTER选项。

         FOR EACH ROW子可是可选的，即INSTEAD OF触发器只能在行级上触发、或只能是行级触发器，没有必要指定。

         没有必要在针对一个表的视图上创建INSTEAD OF触发器，只要创建DML触发器就可以了。

### 2.3 创建系统事件触发器

    ORACLE10G提供的系统事件触发器可以在DDL或数据库系统上被触发。DDL指的是数据定义语言，如CREATE 、ALTER及DROP 等。而数据库系统事件包括数据库服务器的启动或关闭，用户的登录与退出、数据库服务错误等。创建系统触发器的语法如下：

**创建触发器的一般语法是:**

CREATE OR REPLACE TRIGGER [sachema.]trigger\_name  
{BEFORE|AFTER}   
{ddl\_event\_list | database\_event\_list}  
ON { DATABASE | [schema.]SCHEMA }  
[WHEN condition]  
PL/SQL\_block | CALL procedure\_name;

其中: ddl\_event\_list：一个或多个DDL 事件，事件间用 OR 分开；

         database\_event\_list：一个或多个数据库事件，事件间用 OR 分开；

            系统事件触发器既可以建立在一个模式上，又可以建立在整个数据库上。当建立在模式(SCHEMA)之上时，只有模式所指定用户的DDL操作和它们所导致的错误才激活触发器, 默认时为当前用户模式。当建立在数据库(DATABASE)之上时，该数据库所有用户的DDL操作和他们所导致的错误，以及数据库的启动和关闭均可激活触发器。要在数据库之上建立触发器时，要求用户具有ADMINISTER DATABASE TRIGGER权限。

下面给出系统触发器的种类和事件出现的时机（前或后）：

|  |  |  |
| --- | --- | --- |
| **事件** | **允许的时机** | **说明** |
| STARTUP | AFTER | 启动数据库实例之后触发 |
| SHUTDOWN | BEFORE | 关闭数据库实例之前触发（非正常关闭不触发） |
| SERVERERROR | AFTER | 数据库服务器发生错误之后触发 |
| LOGON | AFTER | 成功登录连接到数据库后触发 |
| LOGOFF | BEFORE | 开始断开数据库连接之前触发 |
| CREATE | BEFORE，AFTER | 在执行CREATE语句创建数据库对象之前、之后触发 |
| DROP | BEFORE，AFTER | 在执行DROP语句删除数据库对象之前、之后触发 |
| ALTER | BEFORE，AFTER | 在执行ALTER语句更新数据库对象之前、之后触发 |
| DDL | BEFORE，AFTER | 在执行大多数DDL语句之前、之后触发 |
| GRANT | BEFORE，AFTER | 执行GRANT语句授予权限之前、之后触发 |
| REVOKE | BEFORE，AFTER | 执行REVOKE语句收权限之前、之后触犯发 |
| RENAME | BEFORE，AFTER | 执行RENAME语句更改数据库对象名称之前、之后触犯发 |
| AUDIT **/**NOAUDIT | BEFORE，AFTER | 执行AUDIT**或**NOAUDIT进行审计或停止审计之前、之后触发 |

### 2.4 系统触发器事件属性

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 事件属性\事件 | Startup/Shutdown | Servererror | Logon/Logoff | DDL | DML |
| 事件名称 | **⎫＊** | **⎫＊** | **⎫＊** | **⎫＊** | **＊** |
| 数据库名称 | **⎫＊** |  |  |  |  |
| 数据库实例号 | **⎫＊** |  |  |  |  |
| 错误号 |  | **⎫＊** |  |  |  |
| 用户名 |  |  | **⎫＊** | **＊** |  |
| 模式对象类型 |  |  |  | **⎫＊** | **＊** |
| 模式对象名称 |  |  |  | **⎫＊** | **＊** |
| 列 |  |  |  |  | **⎫＊** |

 除DML语句的列属性外，其余事件属性值可通过调用ORACLE定义的事件属性函数来读取。

|  |  |  |
| --- | --- | --- |
| 函数名称 | 数据类型 | 说    明 |
| Ora\_sysevent | VARCHAR2（20） | 激活触发器的事件名称 |
| Instance\_num | NUMBER | 数据库实例名 |
| Ora\_database\_name | VARCHAR2（50） | 数据库名称 |
| Server\_error(posi) | NUMBER | 错误信息栈中posi指定位置中的错误号 |
| Is\_servererror(err\_number) | BOOLEAN | 检查err\_number指定的错误号是否在错误信息栈中，如果在则返回TRUE，否则返回FALSE。在触发器内调用此函数可以判断是否发生指定的错误。 |
| Login\_user | VARCHAR2(30) | 登陆或注销的用户名称 |
| Dictionary\_obj\_type | VARCHAR2(20) | DDL语句所操作的数据库对象类型 |
| Dictionary\_obj\_name | VARCHAR2(30) | DDL语句所操作的数据库对象名称 |
| Dictionary\_obj\_owner | VARCHAR2(30) | DDL语句所操作的数据库对象所有者名称 |
| Des\_encrypted\_password | VARCHAR2(2) | 正在创建或修改的经过DES算法加密的用户口令 |

### 2.5 使用触发器谓词

    ORACLE 提供三个参数INSERTING, UPDATING, DELETING 用于判断触发了哪些操作。

|  |  |
| --- | --- |
| 谓词 | 行为 |
| INSERTING | 如果触发语句是 INSERT 语句，则为TRUE,否则为FALSE |
| UPDATING | 如果触发语句是 UPDATE语句，则为TRUE,否则为FALSE |
| DELETING | 如果触发语句是 DELETE 语句，则为TRUE,否则为FALSE |

### 2.6 重新编译触发器

如果在触发器内调用其它函数或过程，当这些函数或过程被删除或修改后，触发器的状态将被标识为无效。当DML语句激活一个无效触发器时，ORACLE将重新编译触发器代码，如果编译时发现错误，这将导致DML语句执行失败。

在PL/SQL程序中可以调用ALTER TRIGGER语句重新编译已经创建的触发器，格式为：

ALTER TRIGGER [schema.] trigger\_name COMPILE [ DEBUG]

       其中：DEBUG 选项要器编译器生成PL/SQL 程序条使其所使用的调试代码。

**3 删除和使能触发器**

         **删除触发器：**

DROP TRIGGER trigger\_name;

当删除其他用户模式中的触发器名称，需要具有DROP ANY TRIGGER系统权限，当删除建立在数据库上的触发器时，用户需要具有ADMINISTER DATABASE TRIGGER系统权限。

此外，当删除表或视图时，建立在这些对象上的触发器也随之删除。

         **禁用或启用触发器**

数据库TRIGGER 的状态：

有效状态(ENABLE)：当触发事件发生时，处于有效状态的数据库触发器TRIGGER 将被触发。

无效状态(DISABLE)：当触发事件发生时，处于无效状态的数据库触发器TRIGGER 将不会被触发，此时就跟没有这个数据库触发器(TRIGGER) 一样。

数据库TRIGGER的这两种状态可以互相转换。格式为：

ALTER TIGGER trigger\_name [DISABLE | ENABLE ];

ALTER TRIGGER语句一次只能改变一个触发器的状态，而ALTER TABLE语句则一次能够改变与指定表相关的所有触发器的使用状态。格式为：

ALTER TABLE [schema.]table\_name {ENABLE|DISABLE} ALL TRIGGERS;

**4 触发器和数据字典**

相关数据字典：USER\_TRIGGERS、ALL\_TRIGGERS、DBA\_TRIGGERS

SELECT TRIGGER\_NAME, TRIGGER\_TYPE, TRIGGERING\_EVENT,  
 TABLE\_OWNER, BASE\_OBJECT\_TYPE, REFERENCING\_NAMES,  
 STATUS, ACTION\_TYPE  
 FROM user\_triggers;

## 8.5   数据库触发器的应用举例

**例1：**创建一个DML语句级触发器，当对emp表执行INSERT, UPDATE, DELETE 操作时，它自动更新dept\_summary 表中的数据。由于在PL/SQL块中不能直接调用DDL语句，所以，利用ORACLE内置包DBMS\_UTILITY中的EXEC\_DDL\_STATEMENT过程，由它执行DDL语句创建触发器。

CREATE TABLE dept\_summary(  
 Deptno NUMBER(**2**),  
 Sal\_sum NUMBER(**9**, **2**),  
 Emp\_count NUMBER);   
  
INSERT INTO dept\_summary(deptno, sal\_sum, emp\_count)  
 SELECT deptno, SUM(sal), COUNT(\*)   
FROM emp   
GROUP BY deptno;  
  
--创建一个PL/SQL过程disp\_dept\_summary  
--在触发器中调用该过程显示dept\_summary标中的数据。  
CREATE OR REPLACE PROCEDURE disp\_dept\_summary  
IS  
 Rec dept\_summary%ROWTYPE;  
 CURSOR c1 IS SELECT \* FROM dept\_summary;  
BEGIN  
 OPEN c1;  
 FETCH c1 INTO REC;  
 DBMS\_OUTPUT.PUT\_LINE('deptno    sal\_sum    emp\_count');  
 DBMS\_OUTPUT.PUT\_LINE('-------------------------------------');  
 WHILE c1%FOUND LOOP  
    DBMS\_OUTPUT.PUT\_LINE(RPAD(rec.deptno, **6**)||  
      To\_char(rec.sal\_sum, '$999,999.99')||  
      LPAD(rec.emp\_count, **13**));  
    FETCH c1 INTO rec;  
 END LOOP;  
 CLOSE c1;  
END;  
BEGIN  
 DBMS\_OUTPUT.PUT\_LINE('插入前');  
 Disp\_dept\_summary();  
 DBMS\_UTILITY.EXEC\_DDL\_STATEMENT('  
    CREATE OR REPLACE TRIGGER trig1  
      AFTER INSERT OR DELETE OR UPDATE OF sal ON emp  
    BEGIN  
      DBMS\_OUTPUT.PUT\_LINE(''正在执行trig1 触发器…'');  
      DELETE FROM dept\_summary;  
      INSERT INTO dept\_summary(deptno, sal\_sum, emp\_count)  
      SELECT deptno, SUM(sal), COUNT(\*)   
      FROM emp GROUP BY deptno;  
    END;  
 ');  
  
  
 INSERT INTO dept(deptno, dname, loc)   
 VALUES(**90**, ‘demo\_dept’, ‘none\_loc’);  
 INSERT INTO emp(ename, deptno, empno, sal)  
 VALUES(USER, **90**, **9999**, **3000**);  
  
 DBMS\_OUTPUT.PUT\_LINE('插入后');  
 Disp\_dept\_summary();  
  
 UPDATE emp SET sal=**1000** WHERE empno=**9999**;  
 DBMS\_OUTPUT.PUT\_LINE('修改后');  
 Disp\_dept\_summary();  
  
 DELETE FROM emp WHERE empno=**9999**;  
 DELETE FROM dept WHERE deptno=**90**;  
  
 DBMS\_OUTPUT.PUT\_LINE('删除后');  
 Disp\_dept\_summary();   
 DBMS\_UTILITY.EXEC\_DDL\_STATEMENT(‘DROP TRIGGER trig1’);  
EXCEPTION  
   WHEN OTHERS THEN  
      DBMS\_OUTPUT.PUT\_LINE(SQLCODE||'---'||SQLERRM);  
  
END;

**Note:**

The INSERTING, DELETING, and UPDATING conditional predicates cannot be used for the CALL procedures; they can only be used in a PL/SQL block.

### Accessing Column Values in Row Triggers

Two correlation names exist for every column of the table being modified: one for the old column value, and one for the new column value. Depending on the type of triggering statement, certain correlation names might not have any meaning.

* A trigger fired by an INSERT statement has meaningful access to new column values only. Because the row is being created by the INSERT, the old values are null.
* A trigger fired by an UPDATE statement has access to both old and new column values for both BEFORE and AFTER row triggers.
* A trigger fired by a DELETE statement has meaningful access to :old column values only. Because the row no longer exists after the row is deleted, the :new values are NULL. However, you cannot modify :new values: ORA-4084 is raised if you try to modify :new values.
* Old and new values are available in both BEFORE and AFTER row triggers. A new column value can be assigned in a BEFORE row trigger, but not in an AFTER row trigger (because the triggering statement takes effect before an AFTER row trigger is fired). If a BEFORE row trigger changes the value of new.column, then an AFTER row trigger fired by the same statement sees the change assigned by the BEFORE row trigger.
* Correlation names can also be used in the Boolean expression of a WHEN clause. A colon (:) must precede the old and new qualifiers when they are used in a trigger body, but a colon is not allowed when using the qualifiers in the WHEN clause or the REFERENCING option.