Programmieren mit PL/SQL

Trigger

- Thomas Stütz -

Trigger

Trigger sind eine spezielle Form von **PL-SQL Prozeduren**. Sie werden nicht durch expliziten Aufruf, sondern durch das **Eintreten** eines bestimmten **Ereignisses** ausgeführt. Ein Trigger kann sich auf die Ereignisse **insert, update, delete** beziehen und **vor** oder **nach** dem Ereignis. Sie dienen genauso wie CONSTRAINTS zur Sicherung der Integrität.

Beispiele für den Einsatz von Triggern

- ☞ Über Trigger kann sichergestellt werden, daß bestimmte Benutzer nur zwischen 08:00 und 16:00 Uhr Daten eingeben.
- ☞ Trigger werden nur bei den Aktionen aufgeführt für die sie definiert sind. CONSTRAINTS hingegen bei allen Ereignissen Alle Ereignisse sind **insert**, **update**, **delete**.
- ☞ Trigger können zur Integritätssicherung verteilter Datenbestände eingesetzt werden.

☞ Trotzdem soll versucht werden die Integrität über Constraints zu sichern.

Überblick

- INSTEAD-OF Trigger
  - wird zu einer VIEW definiert
- System-Trigger
  - Schema-Trigger (DDL-Trigger): Die Auslöser sind von Aktionen des Schema-Owners abhängig
  - Database-Trigger: z.B. bei STARTUP oder SHUTDOWN oder SERVERERROR
- DML-Trigger
  - Simple-Trigger
  - Compound-Trigger: kann zu verschiedenen Zeitpunkten „feuern“.

Anlegen eines DML - Triggers

Triggerkomponenten	entsprechende Syntax
❶ **Trigger-Name**	**CREATE OR REPLACE TRIGGER** <Name>
❷ **Trigger-Zeitpunkt**	**BEFORE | AFTER**
❸ **Trigger-Ereignis**	**INSERT OR UPDATE [OF** <spalte1,..**] OR DELETE ON** <Tabellenname>
❹ **Trigger-Typ**	**[FOR EACH ROW]** oder befehlsorientiert
❺ **Trigger-Restriktion**	**WHEN** <PRÄDIKAT>
❻ **Trigger Rumpf**	PL-SQL Bock

Prinzipiell ist einTrigger **befehlsorientiert** (i.S.v. SQL-Statement) und wird **einmal** zum entsprechenden Zeitpunkt per SQL-Statement ausgeführt .

Ein **zeilenorientierter Trigger** (for each row) wird bei **jedem zu bearbeitenden Datensatz** zum entsprechenden Zeitpunkt ausgeführt. **Wird FOR EACH ROW nicht angegeben ist der Trigger ein befehlsorientierter Trigger**.

Bei zeilenorientierten Triggern ist es möglich mit :OLD- bzw. :NEW.spaltenname auf die alten bzw. neuen Werte zuzugreifen

Syntax eines Triggers ab Oracle 11g

Syntax ab Oracle 11g

```
CREATE  [ OR REPLACE TRIGGER ] <trigger_name>
{ BEFORE | AFTER | INSTEAD OF | FOR } <trigger_event>
INSERT OR UPDATE [OF <spalte1,...> ] OR DELETE ON
<table_name>
[ FOR EACH ROW ]
[ { FORWARD | REVERSE } CROSSEDITION ]
[ { FOLLOWS | PRECEDES } <schema.other_trigger> ]
[ { ENABLE | DISABLE } ]
[ WHEN <trigger_condition> ]
BEGIN
   <trigger_body>
END <trigger_name>
```

Arten von DML - Triggern

- Unterscheidung nach Anzahl der Ausführung
  - Nicht-Datensatzbezogene Trigger (**Statement-Trigger**); zeilenorientierter Trigger
  - Datensatzbezogene Trigger (**Row-Trigger**, ForEachRow-Trigger); befehlsorientierter Trigger
- Unterscheidung nach Ausführungszeitpunkt
  - Vorab-Trigger (**Before-Trigger**)
  - Danach-Trigger (**After-Trigger**)
- Unterscheidung nach Komplexität (ab Oracle DB 11g)
  - **Simple Trigger**
  - **Compound Trigger**: ist Statement- und Row-Trigger sowie Ausführung sowohl bevor als auch nach dem Statement und/oder der Datensatzoperation.

Beispiel eines Triggers

Beispiele für den Einsatz von Triggern

```
CREATE OR REPLACE TRIGGER Teil_LegalEntryTime
BEFORE DELETE OR INSERT OR UPDATE ON Teil
WHEN (USER!='SYSTEM')
DECLARE
  not_a_legal_entry EXCEPTION;
BEGIN
  IF TO_CHAR (SYSDATE, 'HH24:MI') NOT BETWEEN '08:00' AND '17:00' THEN
    RAISE not_a_legal_entry;
  EXEPTION
  WHEN  not_a_legal_enty THEN
    RAISE APPLICATION_ERROR (-20000, 'Entries only between 08:00 and 17:00');
END;
```

Constraints und Trigger im Zusammenspiel

❶ Nachdem ein Update,- Insert- oder Delete Befehl abgesetzt wurde, werden zunächst sämtliche Sperren und Kennungen gesetzt.

❷ Danach werden befehlsorientierte Before-Trigger ausgeführt.

❸ Für jede angesprochene Zeile wird
- ① der jeweilige zeilenorientierte Before-Trigger ausgeführt
- ② die Datenänderung durchgeführt und anhand der definierten Constraints überprüft sowie
- ③ der jeweilige zeilenorientierte After-Trigger ausgeführt.

❹ Dann wird der befehlsorientierte After-Trigger ausgeführt.

❺ Die Abarbeitung wird durch ein nochmaliges Prüfen der in den Constraints definierten Integritätsbedingungen und das Aufheben der Sperren und Kennungen beendet.

Wird während des Ablaufs ein Fehler festgestellt und daher zu irgend einem Zeitpunkt abgebrochen, wird der Ablauf mit einem ROLLBACK beendet.

Hinweis: Für mehrere Trigger kann mit der FOLLOWS-Klausel die Reihenfolge festgelegt werden

INSTEAD-OF Trigger

- Problemstellung
  - Man möchte gewissen Benutzern nur teilweisen Zugriff auf Tabellen geben. z.B. Einschränkung nach Abteilungszugehörigkeit.
- Lösung
  - Man erstellt eine View.
  - Möchte man jedoch auch INSERTs und UPDATEs auf eine View mit mehreren Tabellen absetzen, so benötigt man einen Mechanismus, der die Daten auf die Tabellen aufteilt: den INSTEAD-OF Trigger

Beispiel eines INSTEAD-OF Triggers

1. Anlegen einer View

```
create or replace view outemp as
select  EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,dname
from emp e ,dept d where d.deptno = e.deptno
```

Beispiel eines INSTEAD-OF Triggers

2. Den Trigger erstellen

```
create or replace trigger update_emp_thru_outemp_view
 instead of update on outemp
 referencing new as new
 begin
     update emp
      set ename = :new.ename,
        empno = :new.empno,
        job = :new.job,
        mgr = :new.mgr,
        hiredate = :new.hiredate,
        sal = :new.sal,
        comm = :new.comm,
        deptno = ( select deptno from dept where dname = :new.dname )
        where empno = :old.empno;
     if ( sql%rowcount = 0 )
       then
         raise_application_error
          ( -20001, 'Error updating the outemp view !!!' );
     end if;
 end;
```

VORSICHT: Der Trigger Assistent des SQLDevelopers ist (noch) nicht fehlerfrei. Es ist daher vorteilhafter des SQL-Worksheet zu verwenden.

Beispiel eines INSTEAD-OF Triggers

3. Das UPDATE-Statement absetzen

```
update outemp set ename = 'C TURNER' where ename = 'TURNER';
```

4. Ergebnis

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7844	C TURNER	SALESMAN	7698	08.09.81	1500	0	30
7876	ADAMS	CLERK	7788	12.01.83	1100	(null)	20
7900	JAMES	CLERK	7698	03.12.81	950	(null)	30
7902	FORD	ANALYST	7566	03.12.81	3000	(null)	20

HTL LEONDING

Schön, hier zu lernen.