Lucrarea 12.
Tratarea excepțiilor. Triggere.

Tratarea excepțiilor

Excepțiile se datorează unor erori de programare sau unor date introduse greşit.
Exceptiile pot fi:

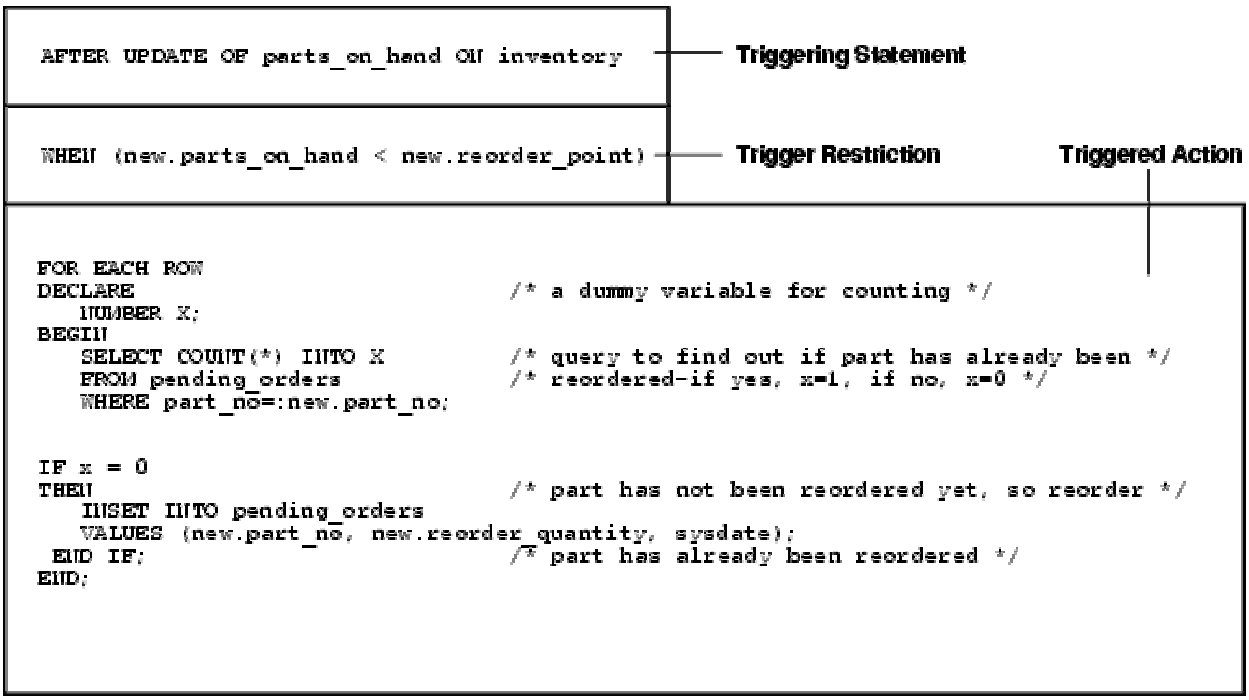
- Exceptii recunoscute ca erori Oracle si tratate automat, ex: daca intr-un Select nu se gaseste o înregistrare care sa îndeplinească conditia va fi afisat mesajul ORA-01403 - NO_DATA_FOUND.
- Exceptii standard recunoscute de Oracle dar tratate de utilizator in sectiunea EXCEPTION. Din aceasta categorie fac parte:
  - No_Data_Found cand nu s-a selectat nici o înregistrare
  - Too_Many_Rows cand in SELECT …INTO var1,var2,… s-au selectat mai mult de o înregistrare datorita conditiei prea putin restrictive.
  - Invalid_Cursor operație de cursor ilegala
  - Zero_Divide impartire cu zero
- Exceptii declarate de utilizator in sectiunea DECLARE si definite in sectiunea EXCEPTION

Daca se trateaza exceptiile structura unei proceduri sau bloc PL/SQL este:

```
DECLARE
Vnume        Emp.Ename%Type;   -- declaratii de variabile locale
Vsal         Emp.Sal%Type;
ErSal  Exception;              -- declaratii de exceptie utilizator – salar eronat
       - declararea de proceduri locale
       - declaratii de cursoare
BEGIN
       - ……
       SELECT Ename,Sal INTO Vnume,Vsal FROM EMP WHERE
       Empno=&Cod_salariat;
Dbms_output.put_line( ‘ Nume salariat: ‘||Vnume||’  Salariu:  ‘||Vsal);
       IF Vsal>2000   THEN          -- exceptie utilizator declarata
Raise ErSal
       END IF;
….
COMMIT;
EXCEPTION
WHEN No_Data_Found THEN       -- exceptie standard tratata in program
RAISE_APPLICATION_ERROR (-20201,’Nu exista angajatul ’);
WHEN ErSal           -- exceptie declarata   - eroare de salar
Dbms_output.put_line(‘  Salar eronat:  ’||Vsal|| ‘  la salariatul:  ‘||VNume);
WHEN OTHERS
Dbms_output.put_line(‘ Eroare necunoscuta   a fost intalnita ‘||
’ Cod eroare: ’||SQLCode|| ’    ’|| SQLERRM);
ROLLBACK;
END;
/
```

Triggere (Declansatoare)

REORDER Trigger



Evenimentul sau comanda declansatoare (Triggering Event)

Un eveniment sau o comanda declansatoare este comanda SQL, evenimentul asupra bazei de date sau a utilizatorului care provoaca declansarea. Aceste evenimente pot fi de tipurile urmatoare:

- Un INSERT, UPDATE, sau DELETE pe o anume tabela (sau view, in unele cazuri) Un CREATE, ALTER, sau DROP pe o schema
- Database startup sau instance shutdown
- Un anume mesaj de eroare sau orice mesaj de eroare
- Un user logon or logoff

Trigger Restriction

O restrictie de trigger specifica o expresie Booleana care trebuie sa fie adevarata (TRUE) pentru a declansa trigger-ul. Actiunea trigger-ului nu este executata daca restrictia e evaluata ca FALSE sau UNKNOWN. In exemplu, restrictia e:

new.parts_on_hand < new.reorder_point

Asadar trigger-ul nu se declanseaza decat daca numarul de piese disponibile este mai mic decat o valoare de la care se face reordonarea.

Trigger Action

Actiunea trigger-ului este procedura (bloc PL/SQL , program Java sau rutina C) ce contine comenzile SQL si codul ce trebuie executat in cazurile urmatoare:

- O comanda declansatoare este data

- Restrictia trigger-ului este evaluata pe TRUE.

Tipuri de Triggere

- Row Triggers si Statement Triggers
- BEFORE si AFTER Triggers
- INSTEAD OF Triggers
- Triggere pentru System Events si User Events

```
/*  Example I  Trigger1
 This example creates a BEFORE statement
trigger named EMP_PERMIT_CHANGES
in the schema SCOTT. This trigger ensures that changes to employee records are
only made during business hours on working days: */
CREATE TRIGGER scott.emp_permit_changes
        BEFORE
        DELETE OR INSERT OR UPDATE
        ON scott.emp
        DECLARE
                dummy           INTEGER;
        BEGIN
                /* If today is a Saturday or Sunday, then return an error.*/
                IF (TO_CHAR(SYSDATE, 'DY') = 'SAT'
OR TO_CHAR(SYSDATE, 'DY') = 'SUN')
                        THEN raise_application_error( -20501,
        'May not change employee table during the weekend');
                END IF;
        /* Compare today's date with the dates of all company holidays.
If today is a company holiday,  then return an error.  */
        /*              SELECT COUNT(*)
                        INTO dummy
                        FROM company_holidays
                        WHERE day = TRUNC(SYSDATE);
                IF dummy > 0
                        THEN raise_application_error( -20501,
'May not change employee table during a holiday');
                END IF;         */
        /*      If the current time is before 8:00AM or after
                6:00PM, then return an error.*/
        IF (TO_CHAR(SYSDATE, 'HH24') < 8 OR
                TO_CHAR(SYSDATE, 'HH24') >= 18)
                THEN raise_application_error( -20502,
        'May only change employee table during working hours');
        END IF;
        END;
/
/*Oracle7 fires this trigger whenever a DELETE, INSERT, or UPDATE statement
affects the EMP table in the schema SCOTT.
Since EMP_PERMIT_CHANGES is a BEFORE statement trigger, Oracle7 fires it
once before executing the triggering statement.        */
ALTER TRIGGER emp_permit_changes disable;    --enable
```

/* **Example II TRIGGER2_Sal**

This example creates a BEFORE row trigger named SALARY_CHECK in the schema SCOTT.
Whenever a new employee is added to the employee table or an existing employee's salary or job is changed, this trigger guarantees that the employee's salary falls within the established salary range for the employee's job: */

```
CREATE TRIGGER scott.salary_check
        BEFORE
        INSERT OR UPDATE OF sal, job ON scott.pers
        FOR EACH ROW
        WHEN (:new.job <> 'PRESIDENT')
        DECLARE
        minsal                  NUMBER(7);
        maxsal                  NUMBER(7);
        BEGIN
                /* Get the minimum and maximum salaries for the
                  employee's job from the SAL_GUIDE table.          */
SELECT minsal, maxsal
                        INTO minsal, maxsal  FROM sal_guide
                        WHERE job = :new.job;

                /* If the employee's salary is below the minimum or */
                /* above the maximum for the job, then generate an error.    */

                IF (:new.sal < minsal OR :new.sal > maxsal)
THEN raise_application_error  ( -20601,  'Salary ' || :new.sal || ' out of range for job '
                        || :new.job || ' for employee ' || :new.ename );
                END IF;
END;
/
/*  Oracle7 fires this trigger whenever one of the following statements is issued:
· an INSERT statement that adds rows to the EMP table
· an UPDATE statement that changes values of the SAL or JOB columns of the EMP
Since SALARY_CHECK is a BEFORE row trigger, Oracle7 fires it before changing each row that is updated by the UPDATE statement or before adding each row that is inserted by the INSERT statement.
SALARY_CHECK has a trigger restriction that prevents it from checking the salary of the company president.
```