Curs 14

Declanșatori (Triggers)

Declanșatori în PL/SQL

1. Declanșatori (Triggers). Noțiuni introductive
2. Crearea declansatorilor DML
3. Folosirea predicatelor conditionale

Database Trigger

Application Trigger

1. Declanșatori (Triggers). Noțiuni introductive

- *Declansatorii permit executarea automata a unor operatii specifice intr-o baza de date, fara a fi nevoie sa scriem un cod de aplicatie suplimentar.*

- Declansatorii sporesc puterea bazei de date si puterea aplicatiei voastre.

Necesitatea unui declansator

Incepem cu un exemplu:

- O regula a afacerii specifica faptul ca ori de cate ori salariul unui angajat este modificat, modificarea trebuie inregistrata intr-o tabela de logare.
- Puteti crea doua proceduri pentru a realiza acest lucru:
  - UPD_EMP_SAL pentru a actualiza salariul
  - si LOG_SAL_CHANGE pentru a insera un rand in tabela de logare.
- Si puteti apela LOG_SAL_CHANGE din cadrul UPD_EMP_SAL sau puteti apela LOG_SAL_CHANGE separat de mediul apelant.
- Dar nu este nevoie sa faceti acest lucru.
- In loc de aceasta, puteti sa creati un declansator.

Exemplu - declansator simplu

```
CREATE OR REPLACE TRIGGER
log_sal_change_trigg
AFTER UPDATE OF sal ON emp
BEGIN
        INSERT INTO log_table (user_id,
logon_date)
        VALUES (USER, SYSDATE);
END;
```

Exemplu - declansator simplu

- Dupa crearea acestui declansator, ori de cate ori o instructiune SQL actualizeaza un salariu, acest declansator se executa automat, inserand un rand in tabela de logare.
- Se spune ca *declansatorul se aprinde automat* (aceasta inseamna executare) ori de cate ori apare evenimentul declansatorului (modificarea unui salariu).
- *Cauza si efect: apare evenimentul si declansatorul se executa(se aprinde).*

1. Ce este un declansator?

- *Un declansator intr-o baza de date*:
  - Este un bloc PL/SQL asociat cu actiunea specifica (un eveniment) asupra unui obiect din baza de date cum ar fi o tabela, vizualizare sau un eveniment la nivel de baza de date – de exemplu o logare cu succes facuta de catre utilizator
  - Este stocat in baza de date
  - Se executa automat ori de cate ori apare actiunea asociata
  - In exemplul anterior declansatorul se asociaza cu actiunea: UPDATE OF sal ON emp

1. Ce este un declansator?

- *Declansatoarele bazei de date* se executa automat ori de cate ori apare in schema sau in baza de date un eveniment de date (cum ar fi DML sau DDL) sau un eveniment de sistem (cum ar fi conectarea unui utilizator sau inchiderea bazei de date de catre administratorul bazei de date)

Database Trigger

Application Trigger

- *Declansatoarele de aplicatie* se executa automat ori de cate ori apare un eveniment particular intr-o aplicatie. Declansatoarele de aplicatie sunt folosite extins in aplicatii dezvoltate cu Oracle Forms Developer.

Ce evenimente pot determina aprinderea unui declansator al bazei de date?

1. Operatii DML asupra unei tabele
2. Operatii DML asupra unei vizualizari cu un declansator INSTEAD OF
3. Instructiuni DDL cum ar fi CREATE si ALTER
4. Evenimente sistem ale bazei de date cum ar fi logarea unui utilizator sau inchiderea bazei de date de catre administratorul bazei de date.

Utilizari posibile ale declansatoarelor

- Consolidarea normelor de securitate intr-o baza de date complexa
- Crearea automata a inregistrarilor de audit
- Punerea in aplicare a normelor complexe de integritate a datelor
- Crearea automata a inregistrarilor de logare
- Prevenirea stergerii accidentale a tabelelor
- Prevenirea aparitiei tranzactiilor DML nevalide
- Generarea automata a valorilor derivate din coloane
- Mentinerea replicarii tabelului sincron

Exemplu

Exemplu – crearea automata a inregistrarilor de logare

- Administratorul bazei de date vrea sa pastreze o inregistrare automata (intr-o tabela a bazei de date) a celor care se logheaza la baza de date si cand se logheaza.
- Acesta poate crea tabela de logare si declansatorul potrivit, cum ar fi:

Exemplu

```
CREATE TABLE log_table
(user_id VARCHAR2(30),
logon_date DATE);

CREATE OR REPLACE TRIGGER logon_trigg
AFTER LOGON ON DATABASE
BEGIN
        INSERT INTO log_table (user_id,
logon_date)
        VALUES (USER, SYSDATE);
END;
```

Reguli pentru folosirea declansatoarelor

1. Nu folositi declansatoarele pentru a duplica sau inlocui operatii pe care le puteti face mai usor in alt mod.

- De exemplu, pentru a implementa regulile simple de integritate a datelor folositi constrangerile, nu declansatoarele.

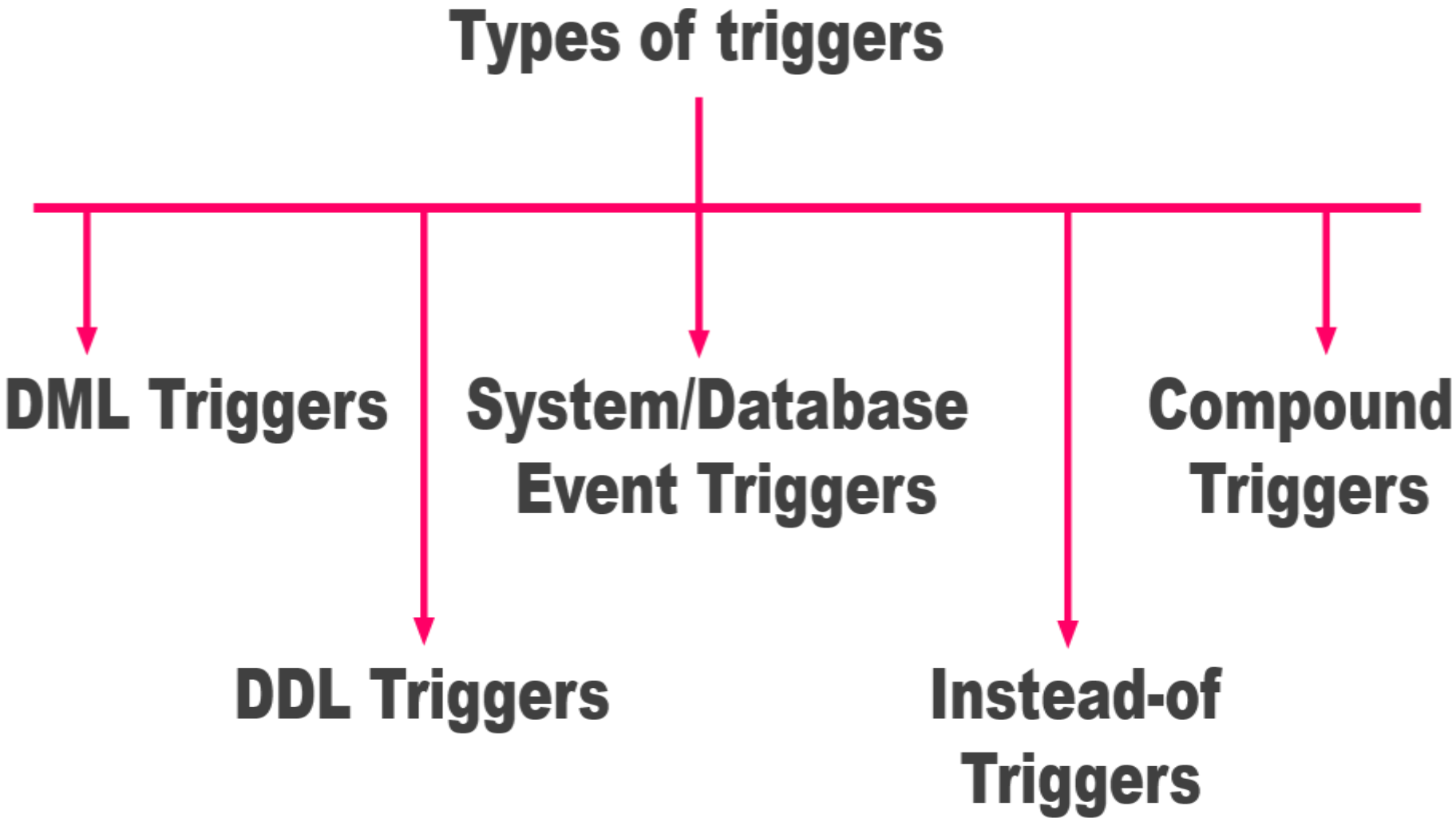
Reguli pentru folosirea declansatoarelor (continuare)

2. Folosirea excesiva a declansatoarelor poate avea ca rezultat o interdependenta complexa care poate fi dificil de intretinut.
   - Folositi declansatoarele doar daca este necesar si fiti atenti la efectele in cascada.

3. Evitati declansatoarele de mari dimensiuni, prin crearea de proceduri stocate sau proceduri impachetate care vor fi apelate in corpul declansatorului.

Comparatie intre declansatoarele bazei de date si procedurile stocate

DECLANSATOARE	PROCEDURI
Se definesc cu CREATE TRIGGER	Se definesc cu CREATE PROCEDURE
Data Dictionary contine codul sursa in USER_TRIGGERS	Data Dictionary contine codul sursa in USER_SOURCE
Apelate implicit	Apelate explicit
Nu sunt permise COMMIT, SAVEPOINT si ROLLBACK	Sunt permise COMMIT, SAVEPOINT si ROLLBACK


Types of triggers
DML Triggers
DDL Triggers
System/Database Event Triggers
Instead-of Triggers
Compound Triggers

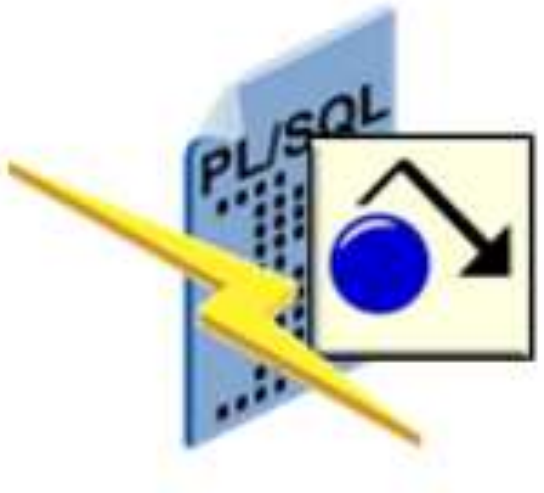
Declanșatori în PL/SQL

1. Declanșatori (Triggers). Noțiuni introductive
2. Crearea declansatorilor DML
3. Folosirea predicatelor conditionale

2. Crearea declansatoarelor DML

- *Un declansator DML (Data Manipulation Language) este un declansator care se executa (se aprinde) automat ori de cate ori se executa o instructiune SQL DML:*

1. ***INSERT***
2. ***UPDATE***
3. ***DELETE***



2. Crearea declansatoarelor DML

Declansatoarele DML se pot clasifica in doua moduri:

1. *Dupa locul in care se executa:*
   1. BEFORE (INAINTE)
   2. AFTER (DUPA)
   3. INSTEAD OF (IN LOC DE)
2. *Dupa numarul de executii*:
   - o singura data pentru intreaga instructiune DML (un *declansator de instructiune*)
   - sau o data pentru fiecare rand afectat de instructiunea DML (*un declansator de rand*).

2. Crearea declansatoarelor DML

Crearea declansatoarelor instructiunilor DML

```
CREATE [OR REPLACE] TRIGGER
trigger_name
        timing
        event1
        [OR event2,
        OR  event3]
        ON    object_name
trigger_body
```

2. Crearea declansatoarelor DML

Unde:

- *timing* – atunci cand se aprinde un declansator in relatie cu evenimentul care-l produce; valorile sunt: BEFORE, AFTER, INSTEAD OF
- *event* – ce operatie DML provoaca aprinderea declansatorului; valorile sunt INSERT, UPDATE [OF column] si DELETE
- *object_name* – tabela sau vizualizarea asociata declansatorului
- *trigger_body* – actiunea (actiunile) executate de catre declansator sunt definite intr-un bloc anonim

2. Crearea declansatoarelor DML

Cand se aprinde un declansator?

1. **BEFORE** - corpul declansatorului se executa inainte de evenimentul DML corespunzator asupra tabelei
2. **AFTER** - corpul declansatorului se executa dupa evenimentul DML
3. **INSTEAD OF** - corpul declansatorului se executa asupra vizualizarii in loc de evenimentul DML

- Cerintele de programare vor stabili care din acestea se va folosi.

2. Crearea declansatoarelor DML

Exemple:

1) Declansatorul se executa imediat inainte ca salariul unui angajat sa fie modificat

```
CREATE OR REPLACE TRIGGER
sal_upd_trigg
        BEFORE UPDATE OF sal ON emp
BEGIN
        ...
END;
```

2. Crearea declansatoarelor DML

Exemple:

2) Declansatorul se executa imediat dupa stergerea unui angajat

```
CREATE OR REPLACE TRIGGER emp_del_trigg
        AFTER DELETE ON emp
BEGIN
        ...
END;
```

2. Crearea declansatoarelor DML

Exemple:

3) Puteti restrictiona un declansator UPDATE pentru a actualiza o anumita coloana sau anumite coloane

```
CREATE OR REPLACE TRIGGER sal_upd_trigg
  BEFORE UPDATE OF sal, comm ON emp
BEGIN
        ...
END;
```

2. Crearea declansatoarelor DML

Exemple:

4) Un declansator poate avea mai multe evenimente declansatoare

```
CREATE OR REPLACE TRIGGER emp_del_trigg
 AFTER INSERT OR DELETE OR UPDATE ON emp
BEGIN
        ...
END;
```

2. Crearea declansatoarelor DML

Cat de des se aprinde o instructiune declansator?

- Se aprinde doar o data pentru fiecare executie a instructiunii declansatorului
- Este tipul implicit de declansator DML
- Se aprinde o data, chiar daca nu este afectat nici un rand

2. Crearea declansatoarelor DML

```
CREATE OR REPLACE TRIGGER
log_emp_changes
AFTER UPDATE ON emp

BEGIN
   INSERT INTO log_emp_table (who, when)
   VALUES (USER, SYSDATE);
END;
```

2. Crearea declansatoarelor DML

Cand se aprinde o instructiune declansator?

- Urmatorul exemplu prezinta secventa de aprindere pentru o instructiune declansator asociata cu evenimentul INSERT INTO dept.

INSERT INTO dept (deptno, dname, loc)
VALUES (400, 'CONSULTING', 2500);

2. Crearea declansatoarelor DML

- Inaintea instructiunii declansator

DEPTNO	DNAME	LOC
10	Administration	1700
20	Marketing	1800
50	Shipping	1500

- Dupa instructiunea declansator

DEPTNO	DNAME	LOC
400	CONSULTING	2500

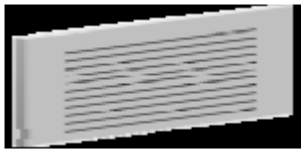
2. Crearea declansatoarelor DML

Exemple – instructiune declansator DML

1) Application Express:
INSERT INTO EMP...;

EMP table

EMPNO	ENAME	JOB
100	King	…
101	Kochhar	…
102	De Haan	…



LOG_EMP trigger

```
CREATE OR REPLACE TRIGGER
log_emp
AFTER INSERT ON emp
BEGIN
INSERT INTO log_emp_table (who, when)
VALUES (USER, SYSDATE);
END;
```

Aceasta instructiune declansator insereaza automat un rand intr-o tabela de logare ori de cate ori unul sau mai multe randuri sunt inserate cu success in tabela emp.

2. Crearea declansatoarelor DML

2)
Urmatoarea instructiune declansator insereaza automat un rand intr-o tabela de logare ori de cate ori se executa cu success o operatie DML pe tabela DEPT.

```
CREATE OR REPLACE TRIGGER log_dept_changes
AFTER INSERT OR UPDATE OR DELETE ON DEPT
BEGIN
        INSERT INTO log_dept_table (which_user,
when_done)
        VALUES (USER, SYSDATE);
END;
```

2. Crearea declansatoarelor DML

3)

Acest exemplu prezinta cum puteti folosi un declansator DML pentru a pune in aplicare reguli complexe ale activitatii, care nu pot fi realizate printr-o constrangere.

- Vreti sa permiteti inserari in tabela EMP in zile obisnuite de lucru (de luni pana vineri), dar sa preveniti inserarile in weekend (sambata sau duminica).
- Daca un utilizator incearca sa insereze un rand in tabela EMP in timpul week-endului, atunci acesta vede un mesaj de eroare, declansatorul esueaza si instructiunea declansatoare revine.

2. Crearea declansatoarelor DML

```
CREATE OR REPLACE TRIGGER secure_emp
BEFORE INSERT ON emp

BEGIN
      IF TO_CHAR(SYSDATE,'DY') IN
('SAT','SUN') THEN
      RAISE_APPLICATION_ERROR(-
20500,'You may insert into EMP  '||'  table only
during business hours');
      END IF;
END;
```

2. Crearea declansatoarelor DML

Testarea declansatorului secure_emp

Un utilizator incearca sa insereze un rand in weekend:

```
INSERT INTO emp (empno, ename, job,
hire_date, sal, deptno, mgr, comm)
VALUES (300, 'Badea', 'IT_PROG', SYSDATE,
4500, 102, 123, 50);
```

2. Crearea declansatoarelor DML

Se vor afisa:

ORA-20500: You may insert into EMP table only during business hours.
ORA-06512: at
“RO_ORACLE_STUDENTI_SQL01_T01.SECURE_EMP”, line 4
ORA_04088: error during execution of trigger
“RO_ORACLE_STUDENTI_SQL01_T01.SECURE_EMP “
2. VALUES (300, 'Badea', 'IT_PROG', SYSDATE, 4500, 102, 123, 50);

2. Crearea declansatoarelor DML

4)
Urmatorul declansator nu se compileaza cu succes. De ce nu?

```
CREATE OR REPLACE TRIGGER log_dept_changes
AFTER INSERT OR UPDATE OR DELETE ON DEPT
BEGIN
        INSERT INTO log_dept_table (which_user,
when_done)
        VALUES (USER, SYSDATE);
        COMMIT;
END;
```

Declanșatori în PL/SQL

1. Declanșatori (Triggers). Noțiuni introductive
2. Crearea declansatorilor DML
3. Folosirea predicatelor conditionale

3. Folosirea predicatelor conditionale

- Am studiat declansatorul care impiedica inserarile in tabela EMP in weekend.

```
CREATE OR REPLACE TRIGGER secure_emp
BEFORE INSERT ON emp
BEGIN
   IF TO_CHAR(SYSDATE,'DY') IN ('SAT','SUN')
    THEN
       RAISE_APPLICATION_ERROR(-20500,'You may
insert into EMP '||' table only during business hours');
   END IF;
END;
```

3. Folosirea predicatelor conditionale

- Sa presupunem ca vreti sa impiedicati orice orice operatie DML in tabela EMP in week-end, cu mesaje diferite de eroare pentru INSERT, UPDATE si DELETE.
- Puteti crea trei declansatoare separate
- Sau puteti face acest lucru cu un singur declansator

3. Folosirea predicatelor conditionale

Exemplu:

SECURE_EMP

Object Details | Code | Errors | SQL

Save & Compile | Find & Replace | Undo | Redo | Download Source | Drop

```
1  create or replace TRIGGER secure_emp
2  BEFORE INSERT OR UPDATE OR DELETE ON emp
3  BEGIN
4  IF TO_CHAR(SYSDATE,'DY') IN ('SAT','SUN') THEN
5  IF DELETING THEN RAISE_APPLICATION_ERROR
6  (-20501,'You may delete from EMP'||' table only during business hours');
7  ELSIF INSERTING THEN RAISE_APPLICATION_ERROR
8  (-20502,'You may insert into EMP'||' table only during business hours');
9  ELSIF UPDATING THEN RAISE_APPLICATION_ERROR
10 (-20503,'You may update EMP'||' table only during business hours');
11     END IF;
12 END IF;
13 END;
14
```

3. Folosirea predicatelor conditionale

Puteti folosi predicate conditionale pentru a testa UPDATE pe o anumita coloana.

SECURE_EMP

Object Details | Code | Errors | SQL

Save & Compile | Find & Replace | Undo | Redo | Download Source | Drop

PL/SQL code successfully compiled (15:16:03)

```
1  create or replace TRIGGER secure_emp
2  BEFORE UPDATE ON emp
3  BEGIN
4  IF UPDATING('SALARY') THEN
5      IF TO_CHAR(SYSDATE,'DY') IN ('SAT','SUN')THEN
6       RAISE_APPLICATION_ERROR (-20501,'You may update SALARY'||' only during business hours');
7      END IF;
8      ELSIF UPDATING('JOB_ID') THEN
9         IF TO_CHAR(SYSDATE,'DY') = 'SUN' THEN
10           RAISE_APPLICATION_ERROR (-20502,'You may not update JOB_ID on Sunday');
11        END IF;
12     END IF;
13 END;
```

3. Folosirea predicatelor conditionale

Intelegerea declansatoarelor de rand

- Sa ne reamintim ca o instructiune declansator se executa doar o data pentru fiecare instructiune DML declansatoare.

```
CREATE OR REPLACE TRIGGER log_emps
AFTER UPDATE OF sal ON emp
BEGIN
        INSERT INTO log_emp_table (who, when)
        VALUES (USER, SYSDATE);
END;
```

- Acest declansator insereaza exact un rand in tabela de logare indiferent daca instructiunea de declansare actualizeaza un angajat, mai multi angajati sau nici unul.

3. Folosirea predicatelor conditionale

- Sa presupunem ca vreti sa inserati un rand intr-o tabela de logare pentru fiecare angajat actualizat.
- De exemplu, daca patru angajati au fost actualizati, inserati patru randuri in tabela de logare.
- Avem nevoie de un *declansator de rand*.

3. Folosirea predicatelor conditionale

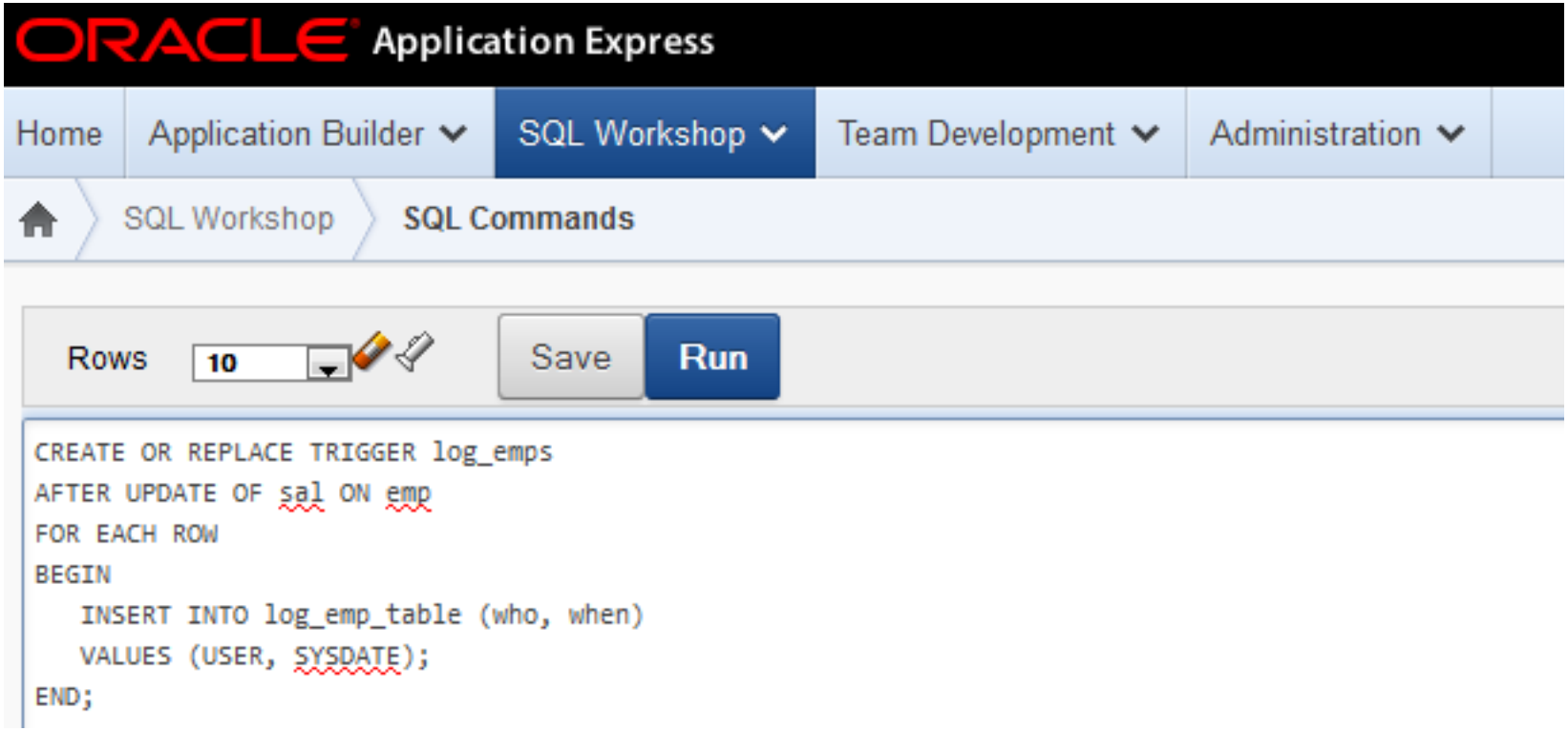
Secventa de aprindere a unui declansator de rand

- Un declansator de rand se aprinde (se executa) o data pentru fiecare rand afectat de instructiunea DML declansatoare, fie chiar inainte ca randul sa fie procesat, sau dupa aceasta.
- Daca sunt 5 de angajati in departamentul 50, declansatorii de rand se executa fiecare de 5 ori.

```
UPDATE emp
SET sal = sal * 1.1
WHERE deptno = 50;
```

3. Folosirea predicatelor conditionale

Crearea unui declansator de rand



Se specifica folosirea unui declansator de rand **FOR EACH ROW**.

3. Folosirea predicatelor conditionale

Calificativele **:OLD** si **:NEW**

- Numai intr-un declansator de rand puteti referi si folosi atat valorile noi cat si cele vechi de coloane dintr-un rand recent actualizat al tabelei EMP.

- Folositi
  - :OLD.column_name pentru a referi o valoare dinainte de actualizare
  - si :NEW.column_name pentru a referi o valoare dupa actualizare.

3. Folosirea predicatelor conditionale

Calificativele **:OLD** si **:NEW**

- De exemplu, daca instructiunea UPDATE modifica salariul unui angajat de la 10000 la 11000, atunci :OLD.sal are valoarea 10000, iar :NEW.sal are valoarea 11000.

- Acum pot fi inserate datele necesare in tabela de logare.

3. Folosirea predicatelor conditionale (exemple)

1)

LOG_EMPS

Object Details | Code | Errors | SQL

Save & Compile | Find & Replace | Undo | Redo | Download Source | Drop

```
1 create or replace TRIGGER log_emps
2 AFTER UPDATE OF sal ON emp
3 FOR EACH ROW
4 BEGIN
5 INSERT INTO log_emp_table (who, when, which_employee, old_salary, new_salary)
6 VALUES (USER, SYSDATE, :OLD.empno, :OLD.sal, :NEW.sal);
7 END;
```

3. Folosirea predicatelor conditionale (exemple)

2)

AUDIT_EMP_VALUES

Object Details | Code | Errors | SQL

Save & Compile | Find & Replace | Undo | Redo | Download Source | Drop

```
1 create or replace TRIGGER audit_emp_values
2 AFTER DELETE OR INSERT OR UPDATE ON emp
3 FOR EACH ROW
4 BEGIN
5    INSERT INTO audit_emp(user_name, time_stamp, id, old_last_name, new_last_name, old_title, new_title, old_salary, new_salary)
6    VALUES (USER, SYSDATE, :OLD.empno, :OLD.ename, :NEW.ename, :OLD.job, :NEW.job, :OLD.sal, :NEW.sal);
7 END;
8
```

3. Folosirea predicatelor conditionale (exemple)

- Testarea declansatorului audit_emp_values

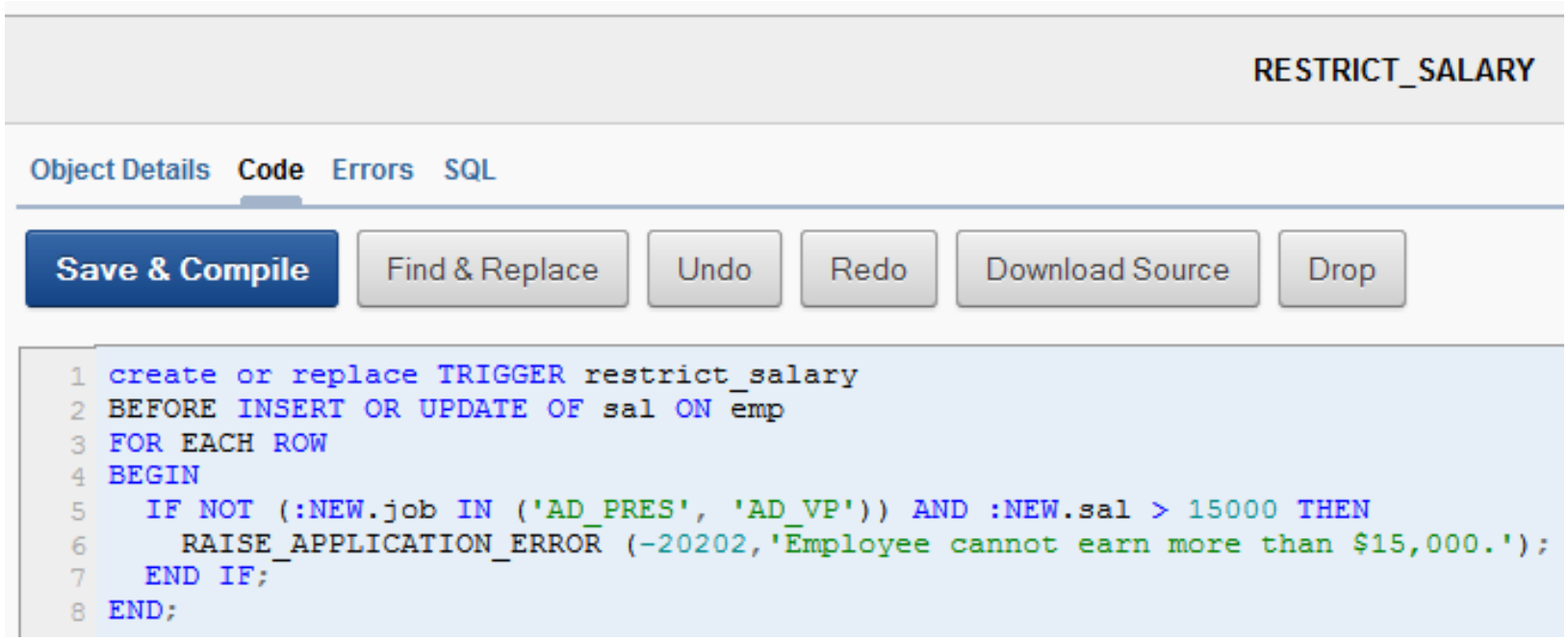
```
INSERT INTO emp (empno, ename, job, salary, ...)
VALUES (999, 'Temp emp', 'SA_REP', 1000,...);

UPDATE emp
SET sal = 2000, ename = 'Smith'
WHERE emp = 999;

SELECT user_name, time_stamp, ...
FROM audit_emp;
```

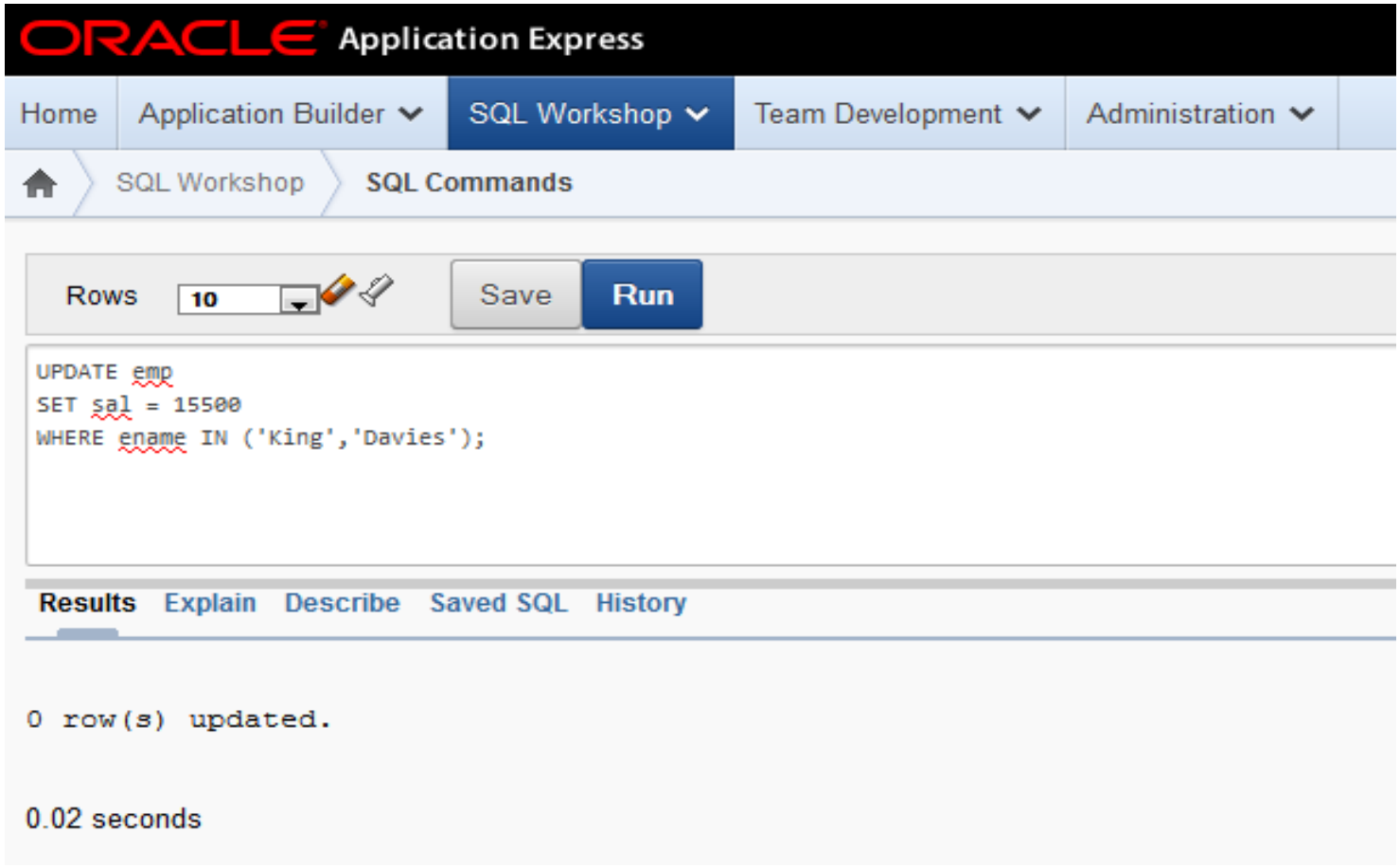
3. Folosirea predicatelor conditionale (exemple)

3)
Sa presupunem ca vrem sa nu permitem ca angajatii care nu sunt presedinte sau vicepresedinte sa aiba salarii mai mari de 15000$



3. Folosirea predicatelor conditionale (exemple)

- Testarea declansatorului restrict_salary



3. Folosirea predicatelor conditionale (exemple)

- King este vicepresedinte, dar Davies nu este.

- Instructiunea UPDATE anterioara produce urmatoarea eroare:

ORA-20202: Employee cannot earn more than $15,000.
ORA-06512: at
“RO_ORACLE_STUDENTI_SQL01_T01.RESTRICT_SALARY”, line 4
ORA-04088: error during execution of trigger
‘RO_ORACLE_STUDENTI_SQL01_T01.RESTRICT_SALARY’
2. WHERE ename IN (‘King’, ‘Davies’);

- Nici un rand al tabelei EMP nu este actualizat deoarece instructiunea UPDATE trebuie sa reuseasca complet sau deloc.

3. Folosirea predicatelor conditionale (exemple)

4) *Implementarea unei constrangeri de integritate cu un declansator*

Tabela EMP are o constrangere de tip **foreign key** (cheie straina) pe coloana DEPTNO a tabelei DEPT.

Valoarea 999 pentru DEPTNO nu exista, deci aceasta instructiune DML incalca constrangerea si randul nu este actualizat.

3. Folosirea predicatelor conditionale (exemple)

```
UPDATE emp
SET deptno = 999
WHERE empno = 124;
```

Puteti folosi un declansator pentru a crea in mod automat un nou departament.

3. Folosirea predicatelor conditionale (exemple)

EMPLOYEE_DEPT_FK_TRG

Object Details | Code | Errors | SQL

Save & Compile | Find & Replace | Undo | Redo | Download Source | Drop

```
1  create or replace TRIGGER employee_dept_fk_trg
2  BEFORE UPDATE OF deptno ON emp
3  FOR EACH ROW
4  DECLARE
5  v_dept_id dept.deptno%TYPE;
6  BEGIN
7     SELECT deptno INTO v_dept_id
8     FROM dept
9     WHERE deptno = :NEW.deptno;
10    EXCEPTION
11      WHEN NO_DATA_FOUND THEN
12           INSERT INTO dept
13           VALUES(:NEW.deptno,'Dept '||:NEW.deptno, NULL, NULL);
14 END;
```

ORACLE Application Express

Home | Application Builder | SQL Workshop | Team Development | Administration

SQL Workshop > SQL Commands

Rows 10 | Save | Run

```
UPDATE emp
SET deptno = 999
WHERE empno = 124;
```

Întrebări?