

# Trighere

- Dependente legate de trighere:

Fie TR un trigher ce utilizeaza chemari de functii sau proceduri. Fie S o astfel de functie sau procedura. Daca S sufera o modificare, atunci TR devine invalid. La o noua lansare a lui TR, acesta se recompileaza.

```
SELECT NAME, REFERENCED_OWNER, REFERENCED_NAME,  
REFERENCED_TYPE FROM ALL_DEPENDENCIES WHERE OWNER =  
'SCOTT' and TYPE = 'TRIGGER';
```

- VALID WITH ERRORS
- ALTER TRIGGER Print\_salary\_changes COMPILE;
- ALTER TRIGGER Reorder ENABLE;
- ALTER TABLE Inventory ENABLE ALL TRIGGERS;
- ALTER TRIGGER Reorder DISABLE;
- **ALTER TABLE Inventory DISABLE ALL TRIGGERS;**

**USER\_TRIGGERS**

**ALL\_TRIGGERS**

**DBA\_TRIGGERS**

# Trighere

- CREATE OR REPLACE TRIGGER Reorder AFTER UPDATE OF Parts\_on\_hand ON Inventory  
FOR EACH ROW WHEN(new.Parts\_on\_hand < new.Reorder\_point)  
DECLARE x NUMBER;  
BEGIN SELECT COUNT(\*) INTO x FROM Pending\_orders WHERE  
Part\_no = :new.Part\_no;  
IF x = 0 THEN  
INSERT INTO Pending\_orders VALUES (:new.Part\_no,  
:new.Reorder\_quantity, sysdate);  
END IF;  
END;  
  
SELECT Trigger\_type, Triggering\_event, Table\_name FROM  
USER\_TRIGGERS WHERE Trigger\_name = 'REORDER';

# Trighere

```
TYPE TRIGGERING_STATEMENT TABLE_NAME
-----
AFTER EACH ROW UPDATE INVENTORY
SELECT Trigger_body FROM USER_TRIGGERS WHERE
Trigger_name = 'REORDER';
TRIGGER_BODY
-----
DECLARE x NUMBER;
BEGIN
    SELECT COUNT(*) INTO x FROM Pending_orders WHERE
        Part_no = :new.Part_no;
    IF x = 0 THEN INSERT INTO Pending_orders VALUES
        (:new.Part_no, :new.Reorder_quantity, sysdate);
    END IF;
END;
```

# Trighere

- CREATE OR REPLACE PACKAGE Auditpackage AS Reason VARCHAR2(10);

```
PROCEDURE Set_reason(Reason VARCHAR2);
```

```
END;
```

```
CREATE TABLE Emp99 ( Empno NOT NULL NUMBER(4), Ename VARCHAR2(10), Job VARCHAR2(9), Mgr NUMBER(4), Hiredate DATE, Sal NUMBER(7,2), Comm NUMBER(7,2), Deptno NUMBER(2), Bonus NUMBER, Ssn NUMBER, Job_classification NUMBER);
```

```
CREATE TABLE Audit_employee ( Oldssn NUMBER, Oldname VARCHAR2(10), Oldjob VARCHAR2(2), Oldsal NUMBER, Newssn NUMBER, Newname VARCHAR2(10), Newjob VARCHAR2(2), Newsal NUMBER, Reason VARCHAR2(10), User1 VARCHAR2(10), Systemdate DATE);
```

## Trighere

- CREATE OR REPLACE TRIGGER Audit\_employee AFTER INSERT OR DELETE OR UPDATE ON Emp99 FOR EACH ROW BEGIN  
/\* AUDITPACKAGE is a package with a public package variable REASON. REASON could be set by the application by a command such as EXECUTE AUDITPACKAGE.SET\_REASON(reason\_string). Note that a package variable has state for the duration of a session and that each session has a separate copy of all package variables. \*/  
IF Auditpackage.Reason IS NULL THEN Raise\_application\_error(-20201, 'Must specify reason' || ' with AUDITPACKAGE.SET\_REASON(Reason\_string)');  
END IF;  
INSERT INTO Audit\_employee VALUES (:old.Ssn, :old.Ename, :old.Job\_classification, :old.Sal, :new.Ssn, :new.Ename, :new.Job\_classification, :new.Sal, auditpackage.Reason, User, Sysdate );  
END;

# Trighere

- CREATE OR REPLACE TRIGGER Audit\_employee\_reset AFTER  
INSERT OR DELETE OR UPDATE ON Emp\_tab  
BEGIN  
auditpackage.set\_reason(NULL);  
END;  
  
CREATE TABLE Audit\_table ( Seq NUMBER, User\_at  
VARCHAR2(10), Time\_now DATE, Term VARCHAR2(10), Job  
VARCHAR2(10), Proc VARCHAR2(10), enum NUMBER);  
CREATE SEQUENCE Audit\_seq;  
  
CREATE TABLE Audit\_table\_values ( Seq NUMBER, Dept  
NUMBER, Dept1 NUMBER, Dept2 NUMBER);  
  
CREATE OR REPLACE TRIGGER Audit\_emp AFTER INSERT  
OR UPDATE OR DELETE ON Emp\_tab  
FOR EACH ROW  
DECLARE Time\_now DATE;  
Terminal CHAR(10);

# Trighere

- BEGIN -- get current time, and the terminal of the user:

```
Time_now := SYSDATE;
```

```
Terminal := USERENV('TERMINAL'); -- record new employee  
-- primary key
```

```
IF INSERTING THEN INSERT INTO Audit_table VALUES  
(Audit_seq.NEXTVAL, User, Time_now, Terminal, 'Emp_tab',  
'INSERT', :new.Empno); -- record primary key of the deleted row:  
ELSIF
```

```
  DELETING THEN INSERT INTO Audit_table VALUES  
(Audit_seq.NEXTVAL, User, Time_now, Terminal, 'Emp_tab',  
'DELETE', :old.Empno); -- for updates, record the primary key  
                        -- of the row being updated:
```

```
INSERT INTO Audit_table VALUES (audit_seq.NEXTVAL, User,  
Time_now, Terminal, 'Emp_tab', 'UPDATE', :old.Empno);
```

```
-- and for SAL and DEPTNO, record old and new values:
```

# Trighere

```
IF UPDATING ('SAL') THEN
  INSERT INTO Audit_table_values VALUES
(Audit_seq.CURRVAL, 'SAL', :old.Sal, :new.Sal);
  ELSIF
  UPDATING ('DEPTNO') THEN
  INSERT INTO Audit_table_values VALUES
(Audit_seq.CURRVAL, 'DEPTNO', :old.Deptno, :new.DEPTNO);
END IF;
END IF;
END;
```