Laborator 10
Problema pregatire examen

Se considera urmatoarele doua tabele:

Tabela1, avand campurile:
- id INTEGER PRIMARY KEY,
- nume VARCHAR(20) NOT NULL,
- salar NUMBER NOT NULL.

Tabela2, avand campurile:
- data_modificarii DATE DEFAULT SYSDATE NOT NULL,
- id INTEGER NOT NULL,
- salar_vechi NUMBER,
- salar_nou NUMBER.

```
CREATE TABLE Tabela1(
  id INTEGER PRIMARY KEY,
  nume VARCHAR(20)NOT NULL,
  salar NUMBER NOT NULL);

CREATE TABLE Tabela2(
  data_modificarii DATE DEFAULT SYSDATE NOT NULL,
  id INTEGER NOT NULL,
  salar_vechi NUMBER,
  salar_nou NUMBER);
```

a) Sa se scrie un trigger, numit HISTORY, care salveaza in *Tabela2* orice modificare are loc pe *Tabela1* (adaugare, stergere de noi linii, respectiv o modificare de salar).

```
CREATE OR REPLACE TRIGGER history
AFTER  INSERT OR UPDATE of salar OR DELETE ON Tabela1
FOR EACH ROW
BEGIN
-- verificare pentru adaugare/modificare
If (:NEW.id IS NOT NULL) THEN
INSERT  INTO  Tabela2  (id,  salar_vechi,  salar_nou)  VALUES(:NEW.id,
:OLD.salar, :NEW.salar);
-- cazul stergere
ELSE
INSERT  INTO  Tabela2  (id,  salar_vechi,  salar_nou)  VALUES(:OLD.id,
:OLD.salar, :NEW.salar);
END IF;
END;
/
```

Pentru testarea trigger-ului:

```
INSERT INTO Tabela1 VALUES (1,'ion', 200);
INSERT INTO Tabela1 VALUES (2,'vali', 100);

SELECT * FROM Tabela1;
SELECT * FROM Tabela2;

UPDATE Tabela1 SET salar=100 WHERE id=1;
SELECT * FROM Tabela1;
SELECT * FROM Tabela2;

DELETE FROM Tabela1 WHERE id=1;
SELECT * FROM Tabela1;
SELECT * FROM Tabela2;
```

b) Indicati modul de interogare al tabelei *Tabela2* pentru a vedea angajatii concediati (stersi din *Tabela1*). Sa se afiseze id–ul lor si data stergerii.

-Afisam angajatii concediati (care au campul salar_nou NULL):

```
SELECT id, data_modificarii FROM Tabela2
WHERE salar_nou is NULL
ORDER BY data_modificarii;
```

c) Indicati modul de interogare al celor doua tabele pentru a vedea angajatii care nu au beneficiat de nicio modificare de salar (fiind inca angajati). Sa se afiseze id-ul si numele lor.

-Afisam id angajati noi (care apar in *Tabela2* o singura data, la angajare):

```
SELECT id FROM Tabela2
HAVING COUNT(id)=1 GROUP BY id;
```

-Afisam si numele lor:

```
SELECT id, nume FROM Tabela1
WHERE id IN (SELECT id FROM Tabela2
HAVING COUNT(id)=1 GROUP BY id);
```

d) Indicati modul de interogare al celor doua tabele pentru a vedea angajatii penalizati (care au avut scaderi de salar, fiind inca angajati). Sa se afiseze id-ul si numele lor.

- Afisam toti angajatii penalizati (cei care au avut scaderi de salar):

```
SELECT id FROM Tabela2 WHERE salar_nou<salar_vechi;
```

-Afisam doar angajatii care nu au fost concediati (deci exista in tabela 1) si au avut scaderi de salar:

```
SELECT id, nume FROM Tabela1
WHERE id IN
(SELECT id FROM Tabela2 WHERE salar_nou<salar_vechi);
```

e) Afisati id-ul, respectiv data angajarii si a concedierii, pentru toti angajatii concediati.

-Afisam angajatii concediati (id, datele angajarii, respectiv datele concedierii): *auto-join*

```
SELECT A.id, A.data_modificarii AS angajat, B.data_modificarii AS concediat
FROM Tabela2 A, Tabela2 B
WHERE A.id=B.id AND A.salar_vechi  IS NULL AND B.salar_nou IS NULL;
```

f) Afisati id-ul, respectiv data angajarii si a concedierii pentru toti angajatii (atat existenti cat si concediati, ca o lista comuna). Pentru cei existenti (neconcediati) la data concedierii se va afisa NULL.

-Afisam toti angajatii - existenti + concediati (id, data angajarii si data concedierii) :

- angajatii concediati – vezi punctul e).
- angajatii neconcediati:

```
SELECT A.id, min(B.data_modificarii) AS angajat, to_date(NULL) AS concediat
FROM tabela1 A, tabela2  B
WHERE A.id=B.id
GROUP BY A.id;
```

```
-UNION face lista comuna:
```

```
SELECT A.id, A.data_modificarii AS angajat, B.data_modificarii AS concediat
FROM Tabela2 A, Tabela2 B
WHERE A.id=B.id AND A.salar_vechi  IS NULL AND B.salar_nou IS NULL
UNION
SELECT A.id, min(B.data_modificarii) AS angajat, to_date(NULL) AS concediat
FROM Tabela1 A, Tabela2  B
WHERE A.id=B.id
GROUP BY A.id;
```

Observatii:

Operatorul **UNION** permite o "alipire pe verticală" a datelor din mai multe tabele, folosind o sintaxă de genul:

```
(comanda_1_SELECT) UNION (comanda_2_SELECT);
```

Operaţia este posibilă în cazul în care cele două comenzi SELECT returnează acelaşi număr de coloane, iar coloanele (corespondente ca poziţie) fac parte din aceeaşi clasă de tipuri (şir de caractere, numeric, dată calendaristică).