Baze de date
Laborator 10

Limbajul de definire a datelor (LDD) – partea II:

Definirea vizualizărilor, secvenţelor, indecsilor.

I. Definirea vizualizărilor (*view*)

- Vizualizările sunt tabele virtuale construite pe baza unor tabele sau a altor vizualizări, denumite tabele de bază.
- Vizualizările nu conţin date, dar reflectă datele din tabelele de bază.
- Vizualizările sunt definite de o cerere *SQL*, motiv pentru care mai sunt denumite cereri stocate.

➢ **Avantajele** utilizării vizualizărilor:
- restricţionarea accesului la date;
- simplificarea unor cereri complexe;
- prezentarea diferitelor imagini asupra datelor;

➢ **Crearea vizualizărilor** se realizează prin comanda *CREATE VIEW*, a cărei sintaxă simplificată este:

CREATE [OR REPLACE] [FORCE | NOFORCE] VIEW
nume_vizualizare [(alias, alias, ..)]
AS *subcerere*
[WITH CHECK OPTION [CONSTRAINT *nume_constrangere]]***
[WITH READ ONLY [CONSTRAINT *nume_constrangere]];***

- *OR REPLACE* se utilizează pentru a schimba definiţia unei vizualizări fără a mai reacorda eventualele privilegii.
- Opţiunea *FORCE* permite crearea vizualizării inainte de definirea tabelelor, ignorând erorile la crearea vizualizării.
- Subcererea poate fi oricât de complexă dar nu poate conţine clauza *ORDER BY*. Dacă se doreşte ordonare se utilizează *ORDER BY* la interogarea vizualizării.
- *WITH CHECK OPTION* permite inserarea şi modificarea prin intermediul vizualizării numai a liniilor ce sunt accesibile vizualizării. Dacă lipseşte numele constrângerii atunci sistemul asociază un nume implicit de tip SYS_C*n* acestei constrangeri (*n* este un număr generat astfel încât numele constrângerii să fie unic).
- *WITH READ ONLY* asigură că prin intermediul vizualizarii nu se pot executa operaţii *LMD*.

➢ **Modificarea vizualizărilor** se realizează prin recrearea acestora cu ajutorul opţiunii *OR REPLACE*. Totuşi, începând cu *Oracle9i*, este posibilă utilizarea comenzii *ALTER VIEW* pentru adăugare de constrângeri vizualizării.

➢ **Suprimarea vizualizărilor** se face cu comanda *DROP VIEW* :
DROP VIEW *nume_vizualizare;*

➢ Informaţii despre vizualizări se pot găsi în dicţionarul datelor interogând vizualizările: *USER_VIEWS*, *ALL_VIEWS* . Pentru aflarea informaţiilor despre coloanele actualizabile, este utilă vizualizarea *USER_UPDATABLE_COLUMNS*.

➢ Subcererile însoţite de un alias care apar în comenzile *SELECT, INSERT, UPDATE, DELETE, MERGE* se numesc **vizualizări *inline***. Spre deosebire de vizualizările propriu zise, acestea nu

sunt considerate obiecte ale schemei ci sunt entităţi temporare (valabile doar pe perioada execuţiei instrucţiunii LMD respective).

- **Operaţii LMD asupra vizualizărilor**
  - Vizualizările se pot împărţi în **simple** şi **complexe**. Această clasificare este importantă pentru că asupra vizualizărilor simple se pot realiza operaţii ***LMD*****,** dar în cazul celor complexe acest lucru nu este posibil intotdeauna (decât prin definirea de *triggeri* de tip *INSTEAD OF*).
    - **Vizualizările *simple*** sunt definite pe baza unui singur tabel şi **nu conţin funcţii** sau **grupări de date**.
    - **Vizualizările *compuse*** sunt definite pe baza mai multor tabele sau conţin funcţii sau grupări de date.
  - **Nu se pot realiza operaţii LMD** în vizualizări ce conţin:
    - funcţii grup;
    - clauzele ***GROUP BY*****,** ***HAVING****, START WITH, CONNECT BY*;
    - cuvântul cheie *DISTINCT*;
    - pseudocoloana *ROWNUM*;
    - operatori pe mulţimi;
  - **Nu se pot actualiza**:
    - coloane ale căror valori rezultă prin calcul sau definite cu ajutorul funcţiei *DECODE*;
    - coloane care nu respectă constrângerile din tabelele de bază;
  - **Pentru vizualizările bazate pe mai multe tabele**, orice operaţie *INSERT*, *UPDATE* sau *DELETE* poate modifica datele doar din unul din tabelele de bază. Acest tabel este cel protejat prin cheie (*key preserved*). În cadrul unei astfel de vizualizări, un tabel de bază se numeşte ***key-preserved*** dacă are proprietatea că fiecare valoare a cheii sale primare sau a unei coloane având constrângerea de unicitate, este unică şi în vizualizare.
    Prima condiţie ca o vizualizare a cărei cerere conţine un *join* să fie modificabilă este ca instrucţiunea *LMD* să afecteze un singur tabel din operaţia de *join*.

- **Reactualizarea** tabelelor implică reactualizarea corespunzătoare a vizualizărilor!!!
  Reactualizarea vizualizărilor implică reactualizarea tabelelor de bază? NU! Există restricţii care trebuie respectate!!!

Exerciţii [I]

1. Să se creeze o vizualizare *VIZ_EMP30_PNU*, care conţine codul, numele, email-ul si salariul angajaţilor din departamentul 30. Să se analizeze structura şi conţinutul vizualizării. Ce se observă referitor la constrângeri? Ce se obţine de fapt la interogarea conţinutului vizualizării? Inseraţi o linie prin intermediul acestei vizualizări; comentaţi.

```
CREATE OR REPLACE VIEW VIZ_EMP30_PNU AS
                (SELECT employee_id, last_name, email, salary
                 FROM emp_pnu
                 WHERE department_id = 30
                 );

DESC VIZ_EMP30_PNU;
SELECT * FROM VIZ_EMP30_PNU;


INSERT INTO VIZ_EMP30_PNU
                VALUES(559,'last_name','eemail',10000);

DROP VIEW VIZ_EMP30_PNU;
```

2. Modificaţi *VIZ_EMP30_PNU* astfel încât să fie posibilă inserarea/modificarea conţinutului tabelului de bază prin intermediul ei. Inseraţi şi actualizaţi o linie (cu valoarea 601 pentru codul angajatului) prin intermediul acestei vizualizări.

Obs: Trebuie introduse neapărat în vizualizare **coloanele care au constrângerea *NOT NULL*** în tabelul de bază (altfel, chiar dacă tipul vizualizării permite operaţii *LMD*, acestea nu vor fi posibile din cauza nerespectării constrângerilor *NOT NULL*).

```
CREATE OR REPLACE VIEW VIZ_EMP30_PNU AS
                (SELECT employee_id, last_name, email, salary, hire_date, job_id, department_id
                 FROM emp_pnu
                 WHERE department_id = 30
                 );

DESC VIZ_EMP30_PNU;
SELECT * FROM VIZ_EMP30_PNU;
SELECT * FROM EMP_PNU;


INSERT INTO VIZ_EMP30_PNU
                VALUES(601, 'last_name', 'eemail', 10000, SYSDATE, 'IT_PROG', 30);

SELECT * FROM VIZ_EMP30_PNU;
SELECT * FROM EMP_PNU;
```

Unde a fost introdusă linia? Mai apare ea la interogarea vizualizării?

Ce efect are următoarea operaţie de actualizare?

```
UPDATE viz_emp30_pnu
SET hire_date=hire_date-15
WHERE employee_id=601;
```

Ştergeţi angajatul având codul 601 prin intermediul vizualizării. Analizaţi efectul asupra tabelului de bază.

```
DELETE FROM viz_emp30_pnu
WHERE employee_id = 601;


COMMIT;
```

3. Să se creeze o vizualizare, *VIZ_EMPSAL50_PNU*, care contine coloanele **cod_angajat**, **nume**, **email**, **functie**, **data_angajare** si **sal_anual** corespunzătoare angajaţilor din departamentul 50. Analizaţi structura şi conţinutul vizualizării.

```
CREATE OR REPLACE VIEW VIZ_EMPSAL50_PNU AS
        SELECT employee_id, last_name, email, job_id, hire_date, salary*12 sal_anual
        FROM emp_pnu
        WHERE department_id = 50;

DESC VIZ_EMPSAL50_PNU;
SELECT * FROM VIZ_EMPSAL50_PNU;
```

4. a) Inseraţi o linie prin intermediul vizualizării precedente. Comentaţi.

```
INSERT INTO VIZ_EMPSAL50_PNU(employee_id, last_name, email, job_id, hire_date)
        VALUES(567, 'last_name', 'email000', 'IT_PROG', sysdate);
```

b) Care sunt coloanele actualizabile ale acestei vizualizări? Verificaţi răspunsul în dicţionarul datelor (***USER_UPDATABLE_COLUMNS***).

c) Analizaţi conţinutul vizualizării *viz_empsal50_pnu* şi al tabelului *emp_pnu*.

5. a) Să se creeze vizualizarea *VIZ_EMP_DEP30_PNU*, astfel încât aceasta să includă coloanele vizualizării *VIZ_EMP30_PNU*, precum şi numele şi codul departamentului. Să se introducă aliasuri pentru coloanele vizualizării.

! Asiguraţi-vă că există constrângerea de cheie externă între tabelele de bază ale acestei vizualizări.

```
CREATE OR REPLACE VIEW VIZ_EMP_DEP30_PNU AS
SELECT v.*,d.department_name
FROM VIZ_EMP30_PNU v JOIN departments d ON(d.department_id = v.department_id);
```

b) Inseraţi o linie prin intermediul acestei vizualizări.

```
INSERT  INTO  VIZ_EMP_DEP30_PNU
        (employee_id,last_name,email,salary,job_id,hire_date,department_id)
        VALUES (358, 'lname', 'email', 15000, 'IT_PROG', sysdate, 30);

SELECT * FROM VIZ_EMP_DEP30_PNU;
SELECT * FROM VIZ_EMP30_PNU;
```

c) Care sunt coloanele actualizabile ale acestei vizualizări? Ce fel de tabel este cel ale cărui coloane sunt actualizabile?

d) Ce efect are o operaţie de ştergere prin intermediul vizualizării *viz_emp_dep30_pnu*? Comentaţi.

DELETE FROM VIZ_EMP_DEP30_PNU WHERE employee_id = 358;

6. Să se creeze vizualizarea *VIZ_DEPT_SUM_PNU*, care conţine codul departamentului şi pentru fiecare departament salariul minim, maxim si media salariilor. Ce fel de vizualizare se obţine (complexa sau simpla)? Se poate actualiza vreo coloană prin intermediul acestei vizualizări?

```
CREATE OR REPLACE VIEW VIZ_DEPT_SUM_PNU AS
        (SELECT department_id, MIN(salary) min_sal, MAX(salary) max_sal,AVG(salary)  med_sal
         FROM employees RIGHT JOIN departments USING (department_id)
         GROUP BY department_id);

SELECT * FROM VIZ_DEPT_SUM_PNU;
```

7. Modificaţi vizualizarea *VIZ_EMP30_PNU* astfel încât să nu permită modificarea sau inserarea de linii ce nu sunt accesibile ei. Vizualizarea va selecta şi coloana *department_id*. Daţi un nume constrângerii şi regăsiţi-o în vizualizarea *USER_CONSTRAINTS* din dicţionarul datelor. Încercaţi să modificaţi şi să inseraţi linii ce nu îndeplinesc condiţia *department_id* = 30.

```
CREATE OR REPLACE VIEW VIZ_EMP30_PNU  AS
        (SELECT employee_id, last_name, email, salary, hire_date, job_id, department_id
         FROM emp_pnu
         WHERE department_id = 30)
WITH READ ONLY CONSTRAINT verific;

INSERT INTO VIZ_EMP30_PNU
        VALUES(600, 'last_name', 'eemail', 10000, SYSDATE, 'IT_PROG', 50);
```

8. Să se consulte informaţii despre vizualizările utilizatorului curent. Folosiţi vizualizarea dicţionarului datelor *USER_VIEWS* (coloanele *VIEW_NAME* şi *TEXT*).

```
SELECT view_name, text
FROM user_views
WHERE view_name LIKE '%PNU';
```

9. Să se selecteze numele, salariul, codul departamentului şi salariul maxim din departamentul din care face parte, pentru fiecare angajat. Este necesară o vizualizare *inline*?

```
SELECT last_name, salary, department_id, (SELECT MAX(salary)
                                          FROM employees
                                          WHERE department_id = E.department_id) max_salary
FROM employees E;
```

10. Să se creeze o vizualizare *VIZ_SAL_PNU*, ce conţine numele angajaţilor, numele departamentelor, salariile şi locaţiile (oraşele) pentru toţi angajaţii. Etichetaţi sugestiv coloanele. Consideraţi ca tabele de bază tabelele originale din schema HR. Care sunt coloanele actualizabile?

```
CREATE OR REPLACE VIEW VIZ_SAL_PNU AS
        (SELECT last_name, department_name, salary, city
         FROM employees JOIN departments USING(department_id)
                JOIN LOCATIONS USING(location_id)
         );

SELECT * FROM VIZ_SAL_PNU;
```

11. Să se implementeze constrângerea ca numele angajaţilor să nu înceapă cu şirul de caractere „Wx”.

```
ALTER TABLE emp_pnu
ADD CONSTRAINT ck_name_emp_pnu
CHECK (UPPER(last_name) NOT LIKE 'WX%');
```

II. Definirea secvenţelor

- Secvenţa este un obiect al bazei de date ce permite generarea de întregi unici pentru a fi folosiţi ca valori pentru cheia primară sau coloane numerice unice. Secvenţele sunt independente de tabele, aşa că aceeaşi secvenţă poate fi folosită pentru mai multe tabele.

- **Crearea secvenţelor** se realizează prin comanda ***CREATE SEQUENCE***, a cărei sintaxă este:

CREATE SEQUENCE *nume_secv*
*[**INCREMENT BY** n]*
*[**START WITH** n]*
*[{**MAXVALUE** n | **NOMAXVALUE**}]*
*[{**MINVALUE** n | **NOMINVALUE**}]*
*[{**CYCLE** | **NOCYCLE**}]*
*[{**CACHE** n | **NOCACHE**}]*

La definirea unei secvenţe se pot specifica:
- numele secvenţei
- diferenţa dintre 2 numere generate succesiv, implicit fiind 1 (*INCREMENT BY*);
- numărul initial, implicit fiind 1 (*START WITH*);
- valoarea maximă, implicit fiind 10^{27} pentru o secvenţă ascendentă şi –1 pentru una descendentă;
- valoarea minimă, implicit fiind 1 pentru o secvenţă ascendentă şi -10^{27} pentru o secvenţă descendentă;
- dacă secvenţa ciclează după ce atinge limita; (*CYCLE*)
- câte numere să încarce în *cache server*, implicit fiind încărcate 20 de numere (*CACHE*).

- Informaţii despre secvenţe găsim în dicţionarul datelor. Pentru secvenţele utilizatorului curent, interogăm *USER_SEQUENCES*. Alte vizualizări utile sunt *ALL_SEQUENCES* şi *DBA_SEQUENCES*.

- **Pseudocoloanele *NEXTVAL* şi *CURRVAL*** permit lucrul efectiv cu secvenţele.
  - *Nume_secv.NEXTVAL* - returnează următoarea valoare a secvenţei, o valoare unică la fiecare referire. Trebuie aplicată cel puţin o dată înainte de a folosi *CURRVAL*;
  - *Nume_secv.CURRVAL* – obţine valoarea curentă a secvenţei.

Obs: Pseudocoloanele **se pot** utiliza în:
- lista *SELECT* a comenzilor ce nu fac parte din subcereri;
- lista *SELECT* a unei cereri ce apare într-un *INSERT*;
- clauza *VALUES* a comenzii *INSERT*;
- clauza *SET* a comenzii *UPDATE*.

Obs: Pseudocoloanele **nu se** pot utiliza:
- în lista *SELECT* a unei vizualizări;
- într-o comanda *SELECT* ce conţine *DISTINCT*, *GROUP BY*, *HAVING* sau *ORDER BY*;
- într-o subcerere în comenzile *SELECT*, *UPDATE*, *DELETE;*
- în clauza *DEFAULT* a comenzilor *CREATE TABLE* sau *ALTER TABLE*;

- **Ştergerea secvenţelor** se face cu ajutorul comenzii *DROP SEQUENCE*.
  DROP SEQUENCE *nume_secventa;*

12. Creaţi o secvenţă pentru generarea codurilor de departamente, *SEQ_DEPT_PNU*. Secvenţa va începe de la 400, va creşte cu 10 de fiecare dată şi va avea valoarea maximă 10000, nu va cicla şi nu va încărca nici un număr înainte de cerere.

13. Ştergeţi secvenţa *SEQ_DEPT_PNU*.

```
CREATE SEQUENCE SEQ_test
INCREMENT BY 10
START WITH 400
MAXVALUE 10000
NOCYCLE
NOCACHE;

SELECT * FROM dept_pnu;

INSERT INTO dept_pnu
VALUES (SEQ_test.nextval, 'DeptNou', null, null);

DELETE FROM dept_pnu
WHERE DEPARTMENT_ID = SEQ_test.currval;  -- nu merge in delete

DELETE FROM dept_pnu
WHERE DEPARTMENT_ID = 410;

DROP SEQUENCE SEQ_test;
```

III. Definirea indecsilor:

- Un index este un obiect al unei scheme utilizator care este utilizat de *server*-ul *Oracle* pentru a mări performanţele unui anumit tip de cereri asupra unui tabel.

- Indecşii :
  - evită scanarea completă a unui tabel la efectuarea unei cereri;
  - reduc operaţiile de **citire/scriere** de pe disc utilizând o cale mai rapidă de acces la date şi anume pointeri la liniile tabelului care corespund unor anumite valori ale unei chei (coloane);
  - sunt independenţi de tabelele pe care le indexează, în sensul că dacă sunt şterşi nu afectează conţinutul tabelelor sau comportamentul altor indecşi;
  - sunt menţinuţi şi utilizaţi automat de către *server*-ul *Oracle*;
  - la ştergerea unui tabel, sunt şterşi şi indecşii asociaţi acestuia.

  Un index este un obiect al schemei unei baze de date care:

  - Creşte viteza de execuţie a cererilor;

Indecşii pot fi creaţi în două moduri:

- automat, de server-ul Oracle (PRIMARY KEY, UNIQUE KEY);
- manual, de către utilizator (CREATE INDEX, CREATE TABLE).

Server-ul Oracle creează automat un index unic atunci când se defineşte o constrângere **PRIMARY KEY** sau **UNIQUE** asupra unei coloane sau unui grup de coloane. Numele indexului va fi acelaşi cu numele constrângerii.

➢ **Se creează** un index atunci când:
- O coloană conţine un domeniu larg de valori;
- O coloană conţine nu număr mare de valori null;
- Una sau mai multe coloane sunt folosite des în clauza ***WHERE*** sau în condiţii de ***JOIN*** în programele de aplicaţii;
- Tabelul este mare şi de obicei cererile obţin mai puţin de 2%-4% din liniile tabelului;

➢ **Nu se creează** un index atunci când:
- Tabelul este mic;
- Coloanele nu sunt folosite des în clauza ***WHERE*** sau în condiţiile de ***JOIN*** ale cererilor;
- Majoritatea cererilor obţin peste 2%-4% din conţinutul tabelului;
- Tabelul este **modificat** frecvent;
- Coloanele indexate sunt referite des în expresii;

➢ **Crearea unui index** se face prin comanda:
CREATE *{****UNIQUE*** | ***BITMAP****}* ***INDEX*** *nume_index*
ON *tabel (coloana1 [, coloana2…]);*

➢ **Modificarea unui index** se face prin comada ***ALTER INDEX***;

➢ **Eliminarea unui index** se face prin comanda: ***DROP INDEX*** *nume_index;*

CREATE INDEX upper_nume_idx ON EMPLOYEES (UPPER(nume));

Indexul creat prin instrucţiunea precedentă facilitează prelucrarea unor interogări precum:

```
SELECT * FROM  EMPLOYEES
WHERE  UPPER(last_name) = 'KING';
```

Pentru a asigura că server-ul Oracle utilizează indexul şi nu efectuează o căutare asupra întregului tabel, valoarea funcţiei corespunzătoare expresiei indexate trebuie să nu fie null în interogările ulterioare creării indexului.

Următoarea instrucţiune garantează utilizarea indexului dar, în absenţa clauzei WHERE, serverul Oracle ar putea cerceta întreg tabelul.

```
SELECT * FROM EMPLOYEES
WHERE UPPER(LAST_NAME) IS NOT NULL
ORDER BY UPPER(LAST_NAME);
```