Baze de date - Anul 2
Laborator 9

Limbajul de definire a datelor (LDD) - II :
Definirea vizualizărilor, secvenţelor, indecşilor, sinonimelor.
Definirea tabelelor temporare şi a vizualizărilor materializate (opţional).

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;
- asigurarea independenţei datelor de programele de aplicaţii;
- prezentarea de diferite 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. Daca lipseşte numele constrângerii atunci sistemul asociază un nume implicit de tip SYS_C*n* acestei constrangeri (*n* este un număr 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. Pe baza tabelului *EMP_PNU*, să se creeze o vizualizare *VIZ_EMP30_PNU*, care conţine codul, numele, email-ul şi 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.
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 300 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*).

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=300;
```

Comentaţi efectul următoarelor instrucţiuni, analizând şi efectul asupra tabelului de bază:

```
UPDATE emp_pnu
SET department_id=30
WHERE employee_id=300;

UPDATE viz_emp30_pnu
SET hire_date=hire_date-15
WHERE employee_id=300;
```

Ştergeţi angajatul având codul 300 prin intermediul vizualizării. Analizaţi efectul asupra tabelului de bază.

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.

4. a) Inseraţi o linie prin intermediul vizualizării precedente. Comentaţi.
b) Care sunt coloanele actualizabile ale acestei vizualizări? Verificaţi răspunsul în dicţionarul datelor (*USER_UPDATABLE_COLUMNS*).

c) Inseraţi o linie specificând valori doar pentru coloanele actualizabile.
d) Analizaţi conţinutul vizualizării *viz_empsal50_pnu* şi al tabelului *emp_pnu*.

5. 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?

6. a) Definiţi o vizualizare, *VIZ_EMP_S_PNU*, care să conţină detalii despre angajaţii corespunzători departamentelor care încep cu litera S. Se pot insera/actualiza linii prin intermediul acestei vizualizări? În care dintre tabele? Ce se întâmplă la ştergerea prin intermediul vizualizării?
b) Recreaţi vizualizarea astfel încât să nu se permită nici o operaţie asupra tabelelor de bază prin intermediul ei. Încercaţi să introduceţi sau să actualizaţi înregistrări prin intermediul acestei vizualizări.

7. 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';
```

8. 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?

9. Să se creeze vizualizarea *V_EMP_PNU* asupra tabelului *EMP_PNU* care conţine codul, numele, prenumele, email-ul şi numărul de telefon ale angajaţilor companiei. Se va impune unicitatea valorilor coloanei email şi constrângerea de cheie primară pentru coloana corespunzătoare codului angajatului.

***Obs**:* Constrângerile asupra vizualizărilor pot fi definite numai în modul *DISABLE NOVALIDATE*. Aceste cuvinte cheie trebuie specificate la declararea constrângerii, nefiind permisă precizarea altor stări.

```
CREATE VIEW viz_emp_pnu (employee_id, first_name, last_name,
                         email UNIQUE DISABLE NOVALIDATE, phone_number,
  CONSTRAINT pk_viz_emp_pnu PRIMARY KEY (employee_id) DISABLE NOVALIDATE)
 AS SELECT employee_id, first_name, last_name, email, phone_number
 FROM emp_pnu;
```

10. Să se implementeze în două moduri constrângerea ca numele angajaţilor nu pot începe cu şirul de caractere „Wx”.

Metoda 1:

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

Metoda 2:

```
CREATE OR REPLACE VIEW viz_emp_wx_pnu
AS SELECT   *
  FROM     emp_pnu
  WHERE    UPPER(last_name) NOT LIKE 'WX%'
WITH CHECK OPTION CONSTRAINT ck_name_emp_pnu2;
UPDATE   viz_emp_wx_pnu
SET      nume = 'Wxyz'
WHERE    employee_id = 150;
```

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;*

Exerciţii [II]

11. 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.
12. Să se selecteze informaţii despre secvenţele utilizatorului curent (nume, valoare minimă, maximă, de incrementare, ultimul număr generat).
13. Creaţi o secvenţă pentru generarea codurilor de angajaţi, *SEQ_EMP_PNU*.
14. Să se modifice toate liniile din *EMP_PNU* (dacă nu mai există, îl recreeaţi), regenerând codul angajaţilor astfel încât să utilizeze secvenţa *SEQ_EMP_PNU* şi să avem continuitate în codurile angajaţilor.
15. Ștergeți secvența *SEQ_DEPT_PNU*.

III. Definirea tabelelor temporare

Pentru crearea tabelelor temporare, se utilizează următoarea formă a comenzii *CREATE TABLE*:

CREATE GLOBAL TEMPORARY TABLE [*schema.*]*nume_tabel*
({*nume_coloană_1 tip_date* [***DEFAULT*** *expresie*]
[*constr_coloană* [*constr_coloană*] …]
| *constr_tabel_sau_view* }
[, {*nume_coloană_2 …* }]) [***ON COMMIT*** {***DELETE*** | ***PRESERVE***} ***ROWS***;

- Un tabel temporar stochează date numai pe durata unei tranzacţii sau a întregii sesiuni.

- Definiţia unui tabel temporar este accesibilă tuturor sesiunilor, dar informaţiile dintr-un astfel de tabel sunt vizibile numai sesiunii care inserează linii în acesta.

- Tabelelor temporare nu li se alocă spaţiu la creare decât dacă s-a folosit clauza "*AS subcerere*"; altfel, spaţiul este alocat la prima instrucţiune "*INSERT*" care a introdus linii în el. De aceea, dacă o instrucţiune *DML*, inclusiv "*SELECT*", este executată asupra tabelului înaintea primului "*INSERT*", ea vede tabelul ca fiind vid.

- Precizarea opţiunii *ON COMMIT* determină dacă datele din tabelul temporar persistă pe durata unei tranzacţii sau a unei sesiuni :
  - Clauza *DELETE ROWS* se utilizează pentru definirea unui tabel temporar specific unei tranzacţii, caz în care sistemul trunchiază tabelul, ştergând toate liniile acestuia după fiecare operaţie de permanentizare (*COMMIT*).
  - Clauza *PRESERVE ROWS* se specifică pentru a defini un tabel temporar specific unei sesiuni, caz în care sistemul trunchiază tabelul la terminarea sesiunii.

- O sesiune este ataşată unui tabel temporar dacă efectuează o operaţie *INSERT* asupra acestuia. Detaşarea sesiunii de un tabel temporar are loc:
  - în urma execuţiei unei comenzi *TRUNCATE*,
  - la terminarea sesiunii sau
  - prin efectuarea unei operaţii *COMMIT*, respectiv *ROLLBACK* asupra unui tabel temporar specific tranzacţiei.

- Comenzile *LDD* pot fi efectuate asupra unui tabel temporar doar dacă nu există nici o sesiune ataşată acestuia.

Exerciţii [V]

16. Creaţi un tabel temporar *TEMP_TRANZ_PNU*, cu datele persistente doar pe durata unei tranzacţii. Acest tabel va conţine o singură coloană *x*, de tip *NUMBER*. Introduceţi o înregistrare în tabel. Listaţi conţinutul tabelului. Permanentizaţi tranzacţia şi listaţi din nou conţinutul tabelului.
17. Creaţi un tabel temporar *TEMP_SESIUNE_PNU*, cu datele persistente pe durata sesiunii. Cerinţele sunt cele de la punctul 1.
18. Iniţiaţi încă o sesiune SQL*Plus. Listaţi structura şi conţinutul tabelelor create anterior. Introduceţi încă o linie în fiecare din cele două tabele.
19. Să se creeze un tabel temporar *angajati_azi_pnu*. Sesiunea fiecărui utilizator care se ocupă de angajări va permite stocarea în acest tabel a angajaţilor pe care i-a recrutat la data curentă. La sfârşitul sesiunii, aceste date vor fi şterse. Se alocă spaţiu acestui tabel la creare ?

```
CREATE GLOBAL TEMPORARY TABLE angajati_azi_pnu
  ON COMMIT PRESERVE ROWS
  AS SELECT  *
     FROM    emp_pnu
     WHERE   hire_date = SYSDATE;
```

20. Inseraţi o nouă înregistrare în tabelul *angajati_azi_pnu*. Incercaţi actualizarea tipului de date al coloanei *last_name* a tabelului *angajati_azi_pnu*.