Baze de date - Anul 1

Laborator 10

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

I. [Obiective]

- Operații de definire a altor tipuri de obiecte ale bazei de date: vizualizări, secvenţe

II. [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. Dacă lipseşte numele constrângerii atunci sistemul asociază un nume implicit de tip *SYS_Cn* 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*, a devenit 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 clasifica 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 întodeauna! Există restricţii care trebuie respectate, însă atunci când reactualizarea poate avea loc, datele modificate sunt cele din tabelele de bază!

III. [Exerciţii - vizualizări]

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.

Observaţie: 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. a) Să se creeze vizualizarea VIZ_EMP_DEP30_PNU, astfel încât aceasta să includă coloanele vizualizării VIZ_EMP_30_PNU, precum şi numele şi codul departamentului. Să se introducă aliasuri pentru coloanele vizualizării.

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

b) Inseraţi o linie prin intermediul acestei vizualizări.

c) Care sunt coloanele actualizabile ale acestei vizualizări? Ce fel de tabel este cel ale cărui coloane sunt actualizabile? Inseraţi o linie, completând doar valorile corespunzătoare.

d) Ce efect are o operaţie de ştergere prin intermediul vizualizării VIZ_EMP_DEP30_PNU? Comentaţi.

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 (complexă sau simplă)? Se poate actualiza vreo coloană prin intermediul acestei vizualizări?

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.

8. 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.

9. 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';
```

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

11. 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?

12. a) 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.

Observaţie: 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;
```

b) Să se adauge o constrângere de cheie primară asupra vizualizării VIZ_EMP_S_PNU.

13. 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;
```

IV. [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 iniţial, 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*, 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.

Observație: 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*.

Observație: 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;*

V. [Exerciţii – secvențe]

14. 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.

15. Să se selecteze informaţii despre secvenţele utilizatorului curent (nume, valoare minimă, maximă, de incrementare, ultimul număr generat).

16. Creaţi o secvenţă pentru generarea codurilor de angajaţi, SEQ_EMP_PNU.

17. 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.

18. Să se insereze câte o inregistrare nouă în EMP_PNU şi DEPT_PNU utilizând cele 2 secvenţe create.

19. Să se selecteze valorile curente ale celor 2 secvenţe.

```
SELECT seq_emp_pnu.currval
FROM dual ;
```

20. Ştergeţi secvenţa SEQ_DEPT_PNU.