

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

I. [Obiective]

- Operații de definire a altor tipuri de obiecte ale bazei de date: vizualizări, indecși și sininime

II. [Definirea vizualizărilor (*view*)]

- Vizualizările sunt tabele virtuale construite pe baza unor tabele sau a altor vizualizări, denumite tabele de bază.
- Vizualizările nu conțin 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:
 - restricționarea accesului la date;
 - simplificarea unor cereri complexe;
 - asigurarea independenței datelor de programele de aplicații;
 - 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 definiția unei vizualizări fără a mai reacorda eventualele privilegii.
- Opțiunea *FORCE* permite crearea vizualizării înainte de definirea tabelelor, ignorând erorile la crearea vizualizării.
- Subcererea poate fi oricât de complexă dar nu poate conține clauza *ORDER BY*. Dacă se dorește ordonare se utilizează *ORDER BY* la interogarea vizualizării.
- *WITH CHECK OPTION* permite inserarea și modificarea prin intermediul vizualizării numai a liniilor ce sunt accesibile vizualizării. Dacă lipsește numele

constrângerii atunci sistemul asociază un nume implicit de tip *SYS_Cn* acestei constrângeri (*n* este un număr generat astfel încât numele constrângerii să fie unic).

- *WITH READ ONLY* asigură că prin intermediul vizualizării nu se pot executa operații LMD.
- Modificarea vizualizărilor se realizează prin recrearea acestora cu ajutorul opțiunii *OR REPLACE*. Totuși, 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*;

- Informații despre vizualizări se pot găsi în dicționarul datelor interogând vizualizările: *USER_VIEWS*, *ALL_VIEWS* . Pentru aflarea informațiilor despre coloanele actualizabile, este utilă vizualizarea *USER_UPDATABLE_COLUMNS*.
- Subcererile însoțite 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 entități temporare (valabile doar pe perioada execuției instrucțiunii LMD respective).

Operații LMD asupra vizualizărilor

Vizualizările se pot clasifica în simple și complexe. Această clasificare este importantă pentru că asupra vizualizărilor simple se pot realiza operații LMD, dar în cazul celor complexe acest lucru nu este posibil întotdeauna (decât prin definirea de triggeri de tip *INSTEAD OF*).

- Vizualizările simple sunt definite pe baza unui singur tabel și nu conțin funcții sau grupări de date.
- Vizualizările compuse sunt definite pe baza mai multor tabele sau conțin funcții sau grupări de date.
- Nu se pot realiza operații LMD în vizualizări ce conțin:
 - funcții grup,
 - clauzele *GROUP BY*, *HAVING*, *START WITH*, *CONNECT BY*,
 - cuvântul cheie *DISTINCT*,
 - pseudocoloana *ROWNUM*,
 - operatori pe mulțimi.
- Nu se pot actualiza:

- coloane ale căror valori rezultă prin calcul sau definite cu ajutorul funcției *DECODE*,
- coloane care nu respectă constrângerile din tabelele de bază.
- Pentru vizualizările bazate pe mai multe tabele, orice operație *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 numește *key-preserved* dacă are proprietatea că fiecare valoare a cheii sale primare sau a unei coloane având constrângerea de unicitate, este unică și în vizualizare.
- Prima condiție ca o vizualizare a cărei cerere conține un join să fie modificabilă este ca instrucțiunea LMD să afecteze un singur tabel din operația de join.
- Reactualizarea tabelelor implică reactualizarea corespunzătoare a vizualizărilor!!!

Reactualizarea vizualizărilor implică reactualizarea tabelelor de bază? Nu întodeauna! Există restricții care trebuie respectate, însă atunci când reactualizarea poate avea loc, datele modificate sunt cele din tabelele de bază!

III. [Exerciții - vizualizări]

1. Pe baza tabelului EMP_PNU, să se creeze o vizualizare VIZ_EMP30_PNU, care conține codul, numele, email-ul și salariul angajaților din departamentul 30. Să se analizeze structura și conținutul vizualizării. Ce se observă referitor la constrângeri? Ce se obține de fapt la interogarea conținutului vizualizării? Intrați o linie prin intermediul acestei vizualizări; comentați.
2. Modificați VIZ_EMP30_PNU astfel încât să fie posibilă inserarea/modificarea conținutului tabelului de bază prin intermediul ei. Intrați și actualizați o linie (cu valoarea 300 pentru codul angajatului) prin intermediul acestei vizualizări.

Observație: Trebuie introduse neapărat în vizualizare coloanele care au constrângerea *NOT NULL* în tabelul de bază (altfel, chiar dacă tipul vizualizării permite operații 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 operație de actualizare?

```
UPDATE viz_emp30_pnu
```

```
SET hire_date=hire_date-15
```

```
WHERE employee_id=300;
```

Comentați efectul următoarelor instrucțiuni, analizând și 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;
```

Ștergeți angajatul având codul 300 prin intermediul vizualizării. Analizați efectul asupra tabelului de bază.

3. Să se creeze o vizualizare, VIZ_EMPSAL50_PNU, care conține coloanele cod_angajat, nume, email, functie, data_angajare si sal_anual corespunzătoare angajaților din departamentul 50. Analizați structura și conținutul vizualizării.
4. a) Inșerați o linie prin intermediul vizualizării precedente. Comentați.
b) Care sunt coloanele actualizabile ale acestei vizualizări? Verificați răspunsul în dicționarul datelor (USER_UPDATABLE_COLUMNS).
c) Inșerați o linie specificând valori doar pentru coloanele actualizabile.
d) Analizați conținutul vizualizării VIZ_EMPSAL50_PNU și 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 și numele și codul departamentului. Să se introducă aliasuri pentru coloanele vizualizării.

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

- b) Inșerați 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? Inșerați o linie, completând doar valorile corespunzătoare.
 - d) Ce efect are o operație de ștergere prin intermediul vizualizării VIZ_EMP_DEP30_PNU? Comentați.
6. Să se creeze vizualizarea VIZ_DEPT_SUM_PNU, care conține codul departamentului și pentru fiecare departament salariul minim, maxim si media salariilor. Ce fel de vizualizare se obține (complexă sau simplă)? Se poate actualiza vreo coloană prin intermediul acestei vizualizări?
 7. Modificați vizualizarea VIZ_EMP30_PNU astfel încât să nu permită modificarea sau inserarea de linii ce nu sunt accesibile ei. Vizualizarea va selecta și coloana

department_id. Dați un nume constrângerii și regăsiți-o în vizualizarea *USER_CONSTRAINTS* din dicționarul datelor. Încercați să modificați și să inserați linii ce nu îndeplinesc condiția department_id = 30.

8. a) Definiți o vizualizare, VIZ_EMP_S_PNU, care să conțină detalii despre angajații corespunzători departamentelor care încep cu litera S. Se pot insera/actualiza linii prin intermediul acestei vizualizări? În care dintre tabele? Ce se întâmplă la ștergerea prin intermediul vizualizării?
- b) Recreați vizualizarea astfel încât să nu se permită nici o operație asupra tabelului de bază prin intermediul ei. Încercați să introduceți sau să actualizați înregistrări prin intermediul acestei vizualizări.
9. Să se consulte informații despre vizualizările utilizatorului curent. Folosiți vizualizarea dicționarului datelor *USER_VIEWS* (coloanele *VIEW_NAME* și *TEXT*).

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

10. Să se selecteze numele, salariul, codul departamentului și 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 conține numele angajaților, numele departamentelor, salariile și locațiile (orașele) pentru toți angajații. Etichetați sugestiv coloanele. Considerați 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 conține codul, numele, prenumele, email-ul și numărul de telefon ale angajaților companiei. Se va impune unicitatea valorilor coloanei email și constrângerea de cheie primară pentru coloana corespunzătoare codului angajatului.

Observație: 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 angajaților nu pot începe cu șirul 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 indecșilor]

- Un index este un obiect al unei scheme utilizator care este utilizat de server-ul Oracle pentru a mări performanțele unui anumit tip de cereri asupra unui tabel.
- Indecșii :
 - evită scanarea completă a unui tabel la efectuarea unei cereri;
 - reduc operațiile de citire/scriere de pe disc utilizând o cale mai rapidă de acces la date și anume pointeri la liniile tabelului care corespund unor anumite valori ale unei chei (coloane);
 - sunt independenți de tabelele pe care le indexează, în sensul că dacă sunt șterși nu afectează conținutul tabelelor sau comportamentul altor indecși;
 - sunt menținuți și utilizați automat de către server-ul Oracle;
 - la ștergerea unui tabel, sunt șterși și indecșii asociați acestuia.
- Tipuri de indecși:
 - indecși normali (indecși ce folosesc B-arbori);
 - indecși *bitmap*, care stochează identificatorii de linie (ROWID) asociați cu o valoare cheie sub forma unui bitmap – sunt de obicei folosiți pentru coloane care nu au un domeniu mare de valori în contextul unei concurențe limitate, de exemplu în data warehouse;
 - indecși partiționați, care constau din partiții corespunzătoare valorilor ce apar în coloanele indexate ale tabelului;

- indecși bazați pe funcții (pe expresii). Aceștia permit construcția cererilor care evaluează valoarea returnată de o expresie, expresie ce poate conține funcții predefinite sau definite de utilizator.
- Indecșii pot fi creați :
 - automat: odată cu definirea unei constrangeri PRIMARY KEY sau UNIQUE;
 - manual: cu ajutorul comenzii CREATE INDEX;
- Se creează un index atunci când:
 - O coloană conține un domeniu larg de valori;
 - O coloană conține nu număr mare de valori null;
 - Una sau mai multe coloane sunt folosite des în clauza WHERE sau în condiții de join în programele de aplicații
 - Tabelul este mare și de obicei cererile obțin mai puțin 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 condițiile de join ale cererilor;
 - Majoritatea cererilor obțin peste 2%-4% din conținutul tabelului;
 - Tabelul este modificat frecvent;
 - Coloanele indexate sunt referite des în expresii;
- Informații despre indecși și despre coloanele implicate în indecși se pot găsi în vizualizările dicționarului datelor *USER_INDEXES*, *USER_IND_COLUMNS*, *ALL_INDEXES*, *ALL_IND_COLUMNS*.
- **Crearea unui index** se realizează prin comanda:
CREATE {UNIQUE | BITMAP} INDEX nume_index
ON tabel (coloana1 [, coloana2...]);
- **Modificarea unui index** se realizează prin comada **ALTER INDEX**.
- **Eliminarea unui index** se realizează prin comanda: **DROP INDEX nume_index;**

V. [Exerciții - indecși]

14. Să se creeze un index (normal, neunic) denumit *IDX_EMP_LAST_NAME_PNU*, asupra coloanei *last_name* din tabelul *emp_pnu*.

15. Să se creeze indecși unici asupra codului angajatului (*employee_id*) și asupra combinației *last_name*, *first_name*, *hire_date* prin două metode (automat și manual).

Observație: Pentru metoda automată impuneți constrângeri de cheie primară asupra codului angajatului și constrângere de unicitate asupra celor 3 coloane. Este recomandabilă această metodă.

16. Creați un index neunic asupra coloanei *department_id* din *EMP_PNU* pentru a eficientiza *join*-urile dintre acest tabel și *DEPT_PNU*.
17. Presupunând că se fac foarte des căutări *case insensitive* asupra numelui departamentului și asupra numelui angajatului, definiți doi indecși bazați pe expresiile *UPPER(department_name)*, respectiv *LOWER(last_name)*.
18. Să se selecteze din dicționarul datelor numele indexului, numele coloanei, poziția din lista de coloane a indexului și proprietatea de unicitate a tuturor indecșilor definiți pe tabelele *EMP_PNU* și *DEPT_PNU*.
19. Eliminați indexul de la exercițiul 14.

VI. [Definirea sinonimelor]

- Pentru a simplifica accesul la obiecte, acestora li se pot asocia sinonime. Crearea unui sinonim este utilă pentru a evita referirea unui obiect ce aparține altui utilizator prefixându-l cu numele utilizatorului și pentru a scurta numele unor obiecte cu numele prea lung.
- Informații despre sinonime se găsesc în vizualizarea din dicționarul datelor *USER_SYNONYMS*.
- **Crearea unui sinonim** se realizează prin comanda:
CREATE [PUBLIC] SYNONYM *nume_sinonim*
FOR *obiect*;
- **Eliminarea unui sinonim** se realizează prin comanda:
DROP SYNONYM *nume_sinonim*;

VII. [Exerciții - sinonime]

20. Creați un sinonim public *EMP_PUBLIC_PNU* pentru tabelul *EMP_PNU*.
21. Creați un sinonim *V30_PNU* pentru vizualizarea *VIZ_EMP30_PNU*.

-
22. Creați un sinonim pentru *DEPT_PNU*. Utilizați sinonimul pentru accesarea datelor din tabel. Redenumiți tabelul (*RENAME ... TO ..*). Încercați din nou să utilizați sinonimul pentru a accesa datele din tabel. Ce se obține?
 23. Eliminați sinonimele create anterior prin intermediul unui script care să selecteze numele sinonimelor din *USER_SYNONYMS* care au terminația "*pnu*" și să genereze un fișier cu comenzile de ștergere corespunzătoare.