

Limbajul de definire a datelor (LDD) – partea II :
Definirea vizualizărilor, secvențelor

I. 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;
 - 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 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*, este 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 împărți 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! Există restricții care trebuie respectate!!!

Exerciții [I]

1. 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? Inserați o linie prin intermediul acestei vizualizări; comentați.

```
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','email',10000);
```

```
DROP VIEW VIZ_EMP30_PNU;
```

2. Modificați *VIZ_EMP30_PNU* astfel încât să fie posibilă inserarea/modificarea conținutului tabelului de bază prin intermediul ei. Inserați și actualizați 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 operații *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 operație de actualizare?

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

Ștergeți angajatul având codul 601 prin intermediul vizualizării. Analizați efectul asupra tabelului de bază.

```
DELETE FROM viz_emp30_pnu
WHERE employee_id = 601;
```

```
COMMIT;
```

3. Să se creeze o vizualizare, *VIZ_EMP30_PNU*, care conține coloanele **cod_angajat**, **nume**, **email**, **functie**, **data_angajare** și **sal_anual** corespunzătoare angajaților din departamentul 50. Analizați structura și conținutul 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) Inserați o linie prin intermediul vizualizării precedente. Comentați.

*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? Verificați răspunsul în dicționarul datelor (**USER_UPDATABLE_COLUMNS**).

c) 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_EMP30_PNU*, precum și numele și codul departamentului. Să se introducă aliasuri pentru coloanele vizualizării.

! Asigurați-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) Inserați 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 operație de ștergere prin intermediul vizualizării *viz_emp_dep30_pnu*? Comentați.

DELETE FROM VIZ_EMP_DEP30_PNU WHERE employee_id = 358;

6. Să se creeze vizualizarea *VIZ_DEPT_SUM_PNU*, care conține codul departamentului și pentru fiecare departament salariul minim, maxim și media salariilor. Ce fel de vizualizare se obține (complexă sau simplă)? 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. 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*.

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 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';*

9. Să se selecteze numele, salariul, codul departamentului și 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 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?

```
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 angajaților să nu înceapă cu șirul de caractere „Wx”.

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

II. Definirea secvențelor

- Secvența este un obiect al bazei de date ce permite generarea de întregi unici pentru a fi folosiți ca valori pentru cheia primară sau coloane numerice unice. Secvențele sunt independente de tabele, așa că aceeași secvență poate fi folosită pentru mai multe tabele.
- **Crearea secvențelor** 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 secvențe se pot specifica:

- numele secvenței
- diferența 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 secvență ascendentă și -1 pentru una descendentă;
- valoarea minimă, implicit fiind 1 pentru o secvență ascendentă și -10^{27} pentru o secvență descendentă;
- dacă secvența ciclează după ce atinge limita; (*CYCLE*)
- câte numere să încarce în *cache server*, implicit fiind încărcate 20 de numere (*CACHE*).

- Informații despre secvențe găsim în dicționarul datelor. Pentru secvențele utilizatorului curent, interogăm *USER_SEQUENCES*. Alte vizualizări utile sunt *ALL_SEQUENCES* și *DBA_SEQUENCES*.
 - **Pseudocoloanele *NEXTVAL* și *CURRVAL*** permit lucrul efectiv cu secvențele.
 - *Nume_secv.NEXTVAL* - returnează următoarea valoare a secvenței, o valoare unică la fiecare referire. Trebuie aplicată cel puțin o dată înainte de a folosi *CURRVAL*;
 - *Nume_secv.CURRVAL* – obține valoarea curentă a secvenței.
- 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 conține *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*.
- **Ștergerea secvențelor** se face cu ajutorul comenzii *DROP SEQUENCE*.
***DROP SEQUENCE* nume_secventa;**

12. Creați o secvență pentru generarea codurilor de departamente, *SEQ_DEPT_PNU*. Secvența va începe de la 400, va crește cu 10 de fiecare dată și va avea valoarea maximă 10000, nu va cicla și nu va încărca nici un număr înainte de cerere.

13. Ștergeți secvența *SEQ_DEPT_PNU*.

```
CREATE SEQUENCE SEQ_test
INCREMENT BY 10
START WITH 400
MAXVALUES 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;
```