1. TEORIE

Limbajul de definire a datelor (LDD) - II

1. Definirea vizualizărilor

➔ "cereri stocate" / tabele virtuale - nu conţin date, dar reflectă datele din tabelele de bază
➔ Avantaje:
- restricționarea accesului la date;
- simplificarea unor cereri complexe;

a. Crearea vizualizărilor

```
CREATE [OR REPLACE] [FORCE | NOFORCE] VIEW
      nume_vizualizare AS subcerere
[WITH CHECK OPTION [CONSTRAINT nume_constrangere]]
[WITH READ ONLY [CONSTRAINT nume_constrangere]];
```

- *OR REPLACE* - suprascrie vizualizarea dacă aceasta există deja
- *FORCE* - permite crearea vizualizării inainte de definirea tabelelor
- *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 operații *LMD*

Observaţii: Subcererea nu poate conține clauza *ORDER BY*. Dacă se doreşte ordonare se utilizează *ORDER BY* la interogarea vizualizării.

b. Modificarea vizualizărilor

➔ se realizează prin recrearea acestora cu ajutorul opțiunii *OR REPLACE*

➔ începând cu *Oracle9i*, este posibilă utilizarea comenzii *ALTER VIEW* pentru adăugare de constrângeri

c. Suprimarea vizualizărilor

➔ se realizează cu comanda DROP VIEW

```
DROP VIEW nume_vizualizare;
```

Observaţii:

➔ 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 şi 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 şi **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,
    - *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 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 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 NU implică reactualizarea tabelelor

de bază.

2. Definirea tabelelor temporare

➔ Un tabel temporar stochează date numai pe durata unei tranzacții sau a întregii sesiuni.

```
CREATE GLOBAL TEMPORARY TABLE nume_tabel
(coloane...)
[ON COMMIT DELETE | PRESERVE ROWS];
```

- *ON COMMIT* - determină dacă datele din tabelul temporar persistă pe durata unei tranzacții sau a unei sesiuni, astfel:
  - *DELETE ROWS* - definirea unui tabel temporar specific unei tranzacții, caz în care sistemul trunchiază tabelul, ştergând toate liniile acestuia după fiecare *COMMIT*
  - *PRESERVE ROWS* - definirea unui tabel temporar specific unei sesiuni, caz în care sistemul trunchiază tabelul la terminarea sesiunii

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

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

➔ O sesiune este ataşată unui tabel temporar dacă efectuează o operație *INSERT* asupra acestuia. Detaşarea sesiunii de un tabel temporar are loc:

- ◆ în urma execuției unei comenzi *TRUNCATE*,
- ◆ la terminarea sesiunii
- ◆ prin efectuarea unei operații *COMMIT*, respectiv *ROLLBACK* asupra unui tabel temporar specific tranzacției.

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

2. EXERCIŢII

[Vizualizări]

1. Pe baza tabelului *EMP_PNU*, să se creeze o vizualizare *VIZ_EMP30_PNU*, care conține codul, numele, email-ul şi salariul angajaților din departamentul 30. Să se analizeze structura şi 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 as
select employee_id, last_name, email, salary
from employees
where department_id = 30;

desc viz_emp30;

select * from viz_emp30;

select * from ( select employee_id, last_name, email, salary
                from employees
                where department_id = 30);

insert into viz_emp30
values(1000, 'View', 'view@view.ro', 5000);
```

2. Modificați *VIZ_EMP30_PNU* astfel încât să fie posibilă inserarea/modificarea conținutului tabelului de bază prin intermediul ei. Inserați şi actualizați 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 operații *LMD*, acestea nu vor fi posibile din cauza nerespectării constrângerilor *NOT NULL*).

```
create or replace view viz_emp30 as
select employee_id, last_name, email, salary, job_id, hire_date
from employees
where department_id = 30;

insert into viz_emp30
values (300, 'nume', 'email421', null, 'SA_REP', sysdate);

select * from viz_emp30;
select * from employees where employee_id = 300;
```

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

   Ce efect are următoarea operație de actualizare?

```
update viz_emp30
set hire_date = hire_date - 15
where employee_id = 300;
```

Comentați efectul următoarelor instrucțiuni, analizând şi efectul asupra tabelului de bază:

```
update employees
set department_id = 30
where employee_id = 300;

select * from viz_emp30;

update viz_emp30
set hire_date = hire_date - 15
WHERE employee_id = 300;
```

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

```
delete
from viz_emp30
where employee_id = 300;
```

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

```
create or replace view viz_empsal50 as
select employee_id, last_name, email, job_id, hire_date,
salary * 12 salanual
from employees
where department_id = 50;

desc viz_empsal50;

select *
from viz_empsal50;
```

4.

a. Inserați o linie prin intermediul vizualizării precedente. Comentați.

```
insert into viz_empsal50 (employee_id, last_name, email,
      job_id, hire_date, salanual)
values (421, 'nume', 'email421', 'SA_REP', sysdate, 500);
```

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

```
select *
from USER_UPDATABLE_COLUMNS
where lower(table_name) = 'viz_empsal50';
```

c. Inserați o linie specificând valori doar pentru coloanele actualizabile.

```
insert into viz_empsal50 (employee_id, last_name, email,
      job_id, hire_date)
values (421, 'nume', 'email421', 'SA_REP', sysdate);
```

d. Analizați conținutul vizualizării *viz_empsal50_pnu* şi al tabelului *emp_pnu*.

```
select * from viz_empsal50;

select * from employees;
```

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

```
create or replace view viz_dep_sum as
select department_id, min(salary) minim, max(salary) maxim,
      avg(salary) medie
from employees
group by department_id;

select *
from USER_UPDATABLE_COLUMNS
where lower(table_name) = 'viz_dep_sum';
```

6. Modificați 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*. Încercați să modificați şi să inserați linii ce nu îndeplinesc condiția *department_id* = 30.

```
create or replace view viz_emp30 AS
    select employee_id, last_name, email, salary, job_id,
hire_date, department_id
from employees
where department_id=30
with check option constraint valid_viz_emp30;

insert into viz_emp30
values (478, 'nume', 'email', 2500, 'SA_MAN', sysdate, 30);

rollback;
```

7. Definiți o vizualizare, *VIZ_EMP_S_PNU*, care să conțină detalii despre angajații corespunzători departamentelor care încep cu litera S. Creați vizualizarea astfel încât să nu se permită nici o operație asupra tabelelor de bază prin intermediul ei. Încercați să introduceți sau să actualizați înregistrări prin intermediul acestei vizualizări.

```
create or replace view viz_emp_s as
select    last_name,    job_id,    email,    e.department_id,
department_name
from emp e join dept d on (e.department_id = d.department_id)
where lower(department_name) like '%s'
with read only;

update viz_emp_s
set department_id = 80
where department_id = 50;

rollback;
```

8. Să se consulte informații despre vizualizările utilizatorului curent. Folosiți vizualizarea dicționarului 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) maxim
from employees e;
```

10. Să se creeze o vizualizare *VIZ_SAL_PNU*, ce conține numele angajaților, numele departamentelor, salariile şi locațiile (oraşele) 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 as
select last_name, department_name, salary, city
from employees join departments using(department_id)
  join locations using (location_id);

select *
from USER_UPDATABLE_COLUMNS
where lower(table_name) = 'viz_sal';
```

11. Să se creeze vizualizarea *V_EMP_PNU* asupra tabelului *EMP_PNU* care conține codul, numele, prenumele, email-ul şi numărul de telefon ale angajaților 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 or replace 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 employees;
```

12. Să se implementeze în două moduri constrângerea ca numele angajaților 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      last_name = 'Wxyz'
WHERE    employee_id = 150;
```

[Tabele temporare]

1. Creați un tabel temporar *TEMP_TRANZ_PNU*, cu datele persistente doar pe durata unei tranzacții. Acest tabel va conține o singură coloană *x*, de tip *NUMBER*. Introduceți o înregistrare în tabel. Listați conținutul tabelului. Permanentizați tranzacția şi listați din nou conținutul tabelului.

```
create global temporary table temp_tranz
(x number(4))
on commit delete rows;

insert into temp_tranz
values(5);

select * from temp_tranz;

commit;
```

2. Creați un tabel temporar *TEMP_SESIUNE_PNU*, cu datele persistente pe durata sesiunii. Cerințele sunt cele de la punctul 1.

```
create global temporary table temp_sesiune
(x NUMBER(4))
on commit preserve rows;

insert into temp_sesiune
values(5);

select * from temp_sesiune;

commit;
```

3. Ştergeți tabelele create anterior.

```
drop table temp_tranz;


truncate table temp_sesiune;

drop table temp_sesiune;
```

4. Să se creeze un tabel temporar *angajati*. Acesta va conține o singură coloană, *nume*.

La sfârşitul tranzacţiei, aceste date vor fi şterse. Se alocă spațiu acestui tabel la creare ?

```
create global temporary table angajati
(nume VARCHAR2(30))
on commit delete rows;
```

5. Inserați o nouă înregistrare în tabelul *angajati*. Incercați actualizarea tipului de date al coloanei *nume.*

```
insert into angajati
values('Ion');

alter table angajati
modify nume VARCHAR2(50);

select * from angajati;
```