

## Baze de date-Anul 2

### Laborator 9 SQL

---

#### Limbajul de definire a datelor (LDD) - II :

Definirea tabelelor temporare, a vizualizărilor, secvențelor, indecșilor, clusterelor, sinonimelor, vizualizărilor materializate

#### I. 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 tranzacții sau a întregii sesiuni.
- 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.
- Precizarea opțiunii *ON COMMIT* determină dacă datele din tabelul temporar persistă pe durata unei tranzacții sau a unei sesiuni :
  - Clauza *DELETE ROWS* se utilizează pentru definirea unui tabel temporar specific unei tranzacții, caz în care sistemul trunchiază tabelul, ștergând toate liniile acestuia după fiecare operație 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 atașată unui tabel temporar dacă efectuează o operație *INSERT* asupra acestuia. Detașarea sesiunii de un tabel temporar are loc :
  - în urma execuției unei comenzi *TRUNCATE*,
  - la terminarea sesiunii sau
  - 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 atașată acestuia.

#### Exerciții [I]

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 și listați din nou conținutul tabelului.
2. Creați un tabel temporar *TEMP\_SESIUNE\_PNU*, cu datele persistente pe durata sesiunii. Cerințele sunt cele de la punctul 1.
3. Inițiați încă o sesiune. Listați structura și conținutul tabelelor create anterior. Introduceți încă o linie în fiecare din cele două tabele.
4. Ștergeți tabelele create anterior. Cum se poate realiza acest lucru?

5. Să se creeze un tabel temporar *angajati\_azi\_pnu*. Sesiunea fiecărui utilizator care se ocupă de angajări va permite stocarea în acest tabel a angajaților pe care i-a recrutat la data curentă. La sfârșitul sesiunii, aceste date vor fi șterse. Se alocă spațiu acestui tabel la creare ?
6. Inserați o nouă înregistrare în tabelul *angajati\_azi\_pnu*. Incercați actualizarea tipului de date al coloanei *last\_name* a tabelului *angajati\_azi\_pnu*.

## 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 reaccorda 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 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, din 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 niste 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 unuia sau a 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*).
- Reactualizarea tabelelor implică reactualizarea corespunzătoare a vizualizărilor!!!  
Reactualizarea vizualizărilor implică reactualizarea tabelelor de bază? NU! Există restricții!!!

## Exerciții [II]

7. 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? Inseși o linie prin intermediul acestei vizualizări; comentați.
8. Modificați *VIZ\_EMP30\_PNU* astfel încât să fie posibilă inserarea/modificarea conținutului tabelului de bază prin intermediul ei. Inseși și actualizați o linie 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*).
9. Să se creeze o vizualizare, *VIZ\_EMP50\_PNU*, care conține coloanele *cod\_angajat*, *nume*, *email*, *functie*, *data\_angajare* și *sal\_anual* corespunzătoare angajaților din departamentul 50. Analizați structura și conținutul vizualizării.
10. a) Inseși o linie prin intermediul vizualizării precedente. Comentați.  
b) Care sunt coloanele actualizabile ale acestei vizualizări?  
c) Inseși o linie specificând valori doar pentru coloanele actualizabile.  
d) Analizați conținutul vizualizării *viz\_emp50\_pnu* și al tabelului *emp\_pnu*.

11. 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. Asigurați-vă că există constrângerea de cheie externă între tabelele de bază ale acestei vizualizări.
- b) Inserați 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? Inserați o linie, completând doar valorile corespunzătoare.
- d) Ce efect are o operație de ștergere prin intermediul vizualizării *viz\_emp\_dep30\_pnu*? Comentați.
12. 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?
13. 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*.
14. 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 tabelelor de bază prin intermediul ei. Încercați să introduceți sau să actualizați înregistrări prin intermediul acestei vizualizări.
15. Să se consulte informații despre vizualizările utilizatorului curent. Folosiți vizualizarea dicționarului datelor *USER\_VIEWS* (coloanele *VIEW\_NAME* și *TEXT*).
- Obs:** Coloana *TEXT* este de tip *LONG* așa că trebuie utilizată comanda *SET LONG n* (din *SQL\*Plus*) pentru a seta numărul de caractere afișate în cazul selectării unei coloane de tip *LONG*.
16. Să se selecteze numele, salariul, codul departamentului și salariul maxim din departamentul din care face parte, pentru fiecare angajat. Este necesară o vizualizare *inline*?
17. 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?
18. 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.
- 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;
```

19. Să se adauge o constrângere de cheie primară asupra vizualizării *viz\_emp\_s\_pnu* (*alter view*).

### III. 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;
  - 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 comandă **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*;

### Exerciții [III]

20. Creați o secvență pentru generarea codurilor de departamente, *SEQ\_DEPT\_PNU*. Secvența va începe de la 200, 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.
21. Să se selecteze informații despre secvențele utilizatorului curent (nume, valoare minimă, maximă, de incrementare, ultimul număr generat).
22. Creați o secvență pentru generarea codurilor de angajați, *SEQ\_EMP\_PNU*.
23. Să se modifice toate liniile din *EMP\_PNU* (dacă nu mai există, îl re-creeați), regenerând codul angajaților astfel încât să utilizeze secvența *SEQ\_EMP\_PNU* și să avem continuitate în codurile angajaților.
24. Să se insereze câte o înregistrare nouă în *EMP\_PNU* și *DEPT\_PNU* utilizând cele 2 secvențe create.
25. Să se selecteze valorile curente ale celor 2 secvențe.
26. Ștergeți secvența *SEQ\_DEPT\_PNU*.

## 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 (indecsi 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 face prin comanda:
 

```
CREATE {UNIQUE | BITMAP} INDEX nume_index  
ON tabel (coloana1 [, coloana2...]) (source1, source2) ;
```
- **Modificarea unui index** se face prin comada **ALTER INDEX** ([source1](#), [source2](#)).
- **Eliminarea unui index** se face prin comanda: **DROP INDEX** nume\_index;

#### Exerciții [IV]

27. Să se creeze un index (normal, neunic) *IDX\_EMP\_LAST\_NAME\_PNU*, asupra coloanei *last\_name* din tabelul *emp\_pnu*.
28. 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).  
*Obs* : Pentru metoda automată impuneți constrângeri de cheie primară asupra codului angajatului și constrângere de unicitate asupra celor 3 coloane. Este recomandabilă această metodă.
29. Creați un index neunic asupra coloanei *department\_id* din *EMP\_PNU* pentru a eficientiza *join*-urile dintre acest tabel și *DEPT\_PNU*.
30. 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)*.
31. 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*.
32. Eliminați indexul de la exercițiul 27.

Alte exemple la aceasta [adresa](#).

## V. Clustere

- Un *cluster* reprezintă o grupare fizică a două sau mai multe tabele, relativ la una sau mai multe coloane, cu scopul măririi performanțelor.
- Coloanele comune definesc cheia *cluster*-ului.
- Crearea unui *cluster* presupune:
  - crearea structurii *cluster*-ului;
  - crearea indexului *cluster*-ului;
  - crearea tabelor care vor compune *cluster*-ul.
- Crearea unui cluster se realizează prin comanda:  
CREATE CLUSTER nume\_cluster  
(nume\_coloana tip\_data [,nume\_coloana tip\_data] ...) [SIZE n]
- Ștergerea unui cluster se realizează prin comanda:  
DROP CLUSTER nume\_cluster;
- Ștergerea unui cluster, a tuturor tabelor definite relativ la acel cluster și constrângerile lor de integritate se realizează prin comanda:  
DROP CLUSTER nume\_cluster  
INCLUDING TABLES  
CASCADE CONSTRAINTS;
- Modificarea unui cluster permite redefinirea condițiilor, modificarea parametrilor de stocare și a caracteristicilor de stare (ALTER CLUSTER).

### Exerciții [V]

33. Creați un cluster denumit *angajati\_pnu* având cheia denumită *angajat* și dimensiunea 512 bytes. Extensia inițială alocată *cluster*-ului va avea dimensiunea 100 bytes, iar următoarele extensii alocate vor avea dimensiunea de 50 bytes. Pentru a specifica dimensiunile în kilobytes sau megabytes se folosește K, respectiv M (de exemplu, 100K sau 100M).

```
CREATE CLUSTER angajati_pnu  
(angajat NUMBER(6))  
SIZE 512  
STORAGE (initial 100 next 50);
```

34. Definiți un index pe cheia *cluster*-ului.  
CREATE INDEX idx\_angajati\_pnu ON CLUSTER angajati\_pnu;

35. Adăugați *cluster*-ului următoarele trei tabele:
- tabelul ang\_1\_pnu care va conține angajații având salariul mai mic decât 5000;
  - tabelul ang\_2\_pnu care va conține angajații având salariul între 5000 și 10000;
  - tabelul ang\_3\_pnu care va conține angajații având salariul mai mare decât 10000.



```
CREATE TABLE ang_1_pnu  
CLUSTER angajati_pnu(employee_id)  
AS SELECT * FROM employees WHERE salary < 5000;
```

36. Afișați informații despre cluster-ele create de utilizatorul current (USER\_CLUSTERS).
37. Afișați numele cluster-ului din care face parte tabelul ang\_3\_pnu (USER\_TABLES).
38. Eliminați tabelul ang\_3\_pnu din cluster.
39. Consultând vizualizarea USER\_TABLES, verificați dacă tabelul ang\_3\_pnu mai face parte dintr-un cluster.
40. Stergeți tabelul ang\_2\_pnu. Consultați vizualizarea USER\_TABLES afișați numele tabelelor care fac parte din cluster-ul definit.
41. Stergeți cluster-ul eliminând și tabelele asociate.

```
DROP CLUSTER angajati_pnu  
INCLUDING TABLES CASCADE CONSTRAINTS;
```

Alte informații la aceste adrese: [link1](#) și [link2](#).

## 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 sinonimelor** se realizează prin comanda:  
**CREATE [PUBLIC] SYNONYM** nume\_sinonim  
**FOR** obiect;
- **Eliminarea sinonimelor** se face prin comanda :  
**DROP SYNONYM** nume\_sinonim;

### Exerciții [VI]

42. Creați un sinonim public *EMP\_PUBLIC\_PNU* pentru tabelul *EMP\_PNU*.
43. Creați un sinonim *V30\_PNU* pentru vizualizarea *VIZ\_EMP30\_PNU*.
44. 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?

45. 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.

```
SET FEEDBACK OFF
SET HEADING OFF
SET TERMOUT OFF
SPOOL "H:\...\delSynonym.sql"
SELECT 'DROP SYNONYM ' || synonym_name ||'; '
FROM user_synonyms
WHERE lower(synonym_name) LIKE '%pnu'
/
SPOOL OFF
SET FEEDBACK ON
SET HEADING ON
SET TERMOUT ON
```

Obs:

*SET TERMOUT OFF* – suprimă afișarea rezultatelor pe ecran (acestea vor fi regăsite numai în fișierul specificat în comanda *SPOOL*).

*SPOOL nume\_fisier* determină înregistrarea tuturor comenzilor care urmează și a rezultatelor acestora în fișierul specificat.

*SPOOL OFF* determină oprirea înregistrării.

## VII. Definirea vizualizărilor materializate

- O vizualizare materializată, cunoscută în versiunile anterioare sub numele de clișeu (*snapshot*), este un obiect al schemei ce stochează rezultatele unei cereri și care este folosit pentru a rezuma, calcula, replica și distribui date. Vizualizările materializate sunt utile în domenii precum *data warehouse*, suportul deciziilor și calcul distribuit sau mobil.
- Clauza *FROM* a cererii poate referi tabele, vizualizări sau alte vizualizări materializate. Luate în ansamblu, aceste obiecte sunt referite prin tabele *master* (în temeni de replicare) sau prin tabele detaliu (în termeni de *data warehouse*).
- Optimizorul pe bază de costuri (cel folosit de *Oracle9i*) poate utiliza vizualizările materializate pentru a îmbunătăți execuția cererilor. Acesta recunoaște automat situațiile în care o astfel de vizualizare poate și trebuie să fie utilizată pentru rezolvarea unei cereri. În urma unui asemenea demers, optimizorul rescrie cererea utilizând vizualizarea materializată.
- Din câteva puncte de vedere, vizualizările materializate sunt similare indecșilor:
  - consumă spațiu de stocare;
  - trebuie reactualizate când datele din tabelele de bază sunt modificate;
  - îmbunătățesc performanța execuției instrucțiunilor *SQL* dacă sunt folosite pentru rescrierea cererilor;
  - sunt transparente aplicațiilor *SQL* și utilizatorilor.
- Spre deosebire de indecși, vizualizările materializate pot fi accesate utilizând instrucțiuni *SELECT* și pot fi actualizate prin instrucțiunile *INSERT*, *UPDATE*, *DELETE*.
- Asupra unei vizualizări materializate se pot defini unul sau mai mulți indecși.
- Similar vizualizărilor obișnuite, asupra celor materializate se pot defini constrângerile *PRIMARY KEY*, *UNIQUE* și *FOREIGN KEY*. Singura stare validă a unei constrângeri este *DISEABLE NOVALIDATE*.

Pentru compatibilitate cu versiunile anterioare, cuvintele cheie *SNAPSHOT* și *MATERIALIZED VIEW* sunt echivalente.

- **Crearea vizualizărilor materializate** se realizează prin comanda *CREATE MATERIALIZED VIEW*, a cărei sintaxă simplificată este:

```
CREATE MATERIALIZED VIEW [schema.]nume_viz_materializată  
[ {proprietăți_vm | ON PREBUILT TABLE  
[{WITH | WITHOUT} REDUCED PRECISION] } ]  
[refresh_vm] [FOR UPDATE]  
[ {DISABLE | ENABLE} QUERY REWRITE] AS subcerere;
```

Opțiunea *ON PREBUILT TABLE* permite considerarea unui tabel existent ca fiind o vizualizare materializată predefinită.

Clauza *WITH REDUCED PRECISION* permite ca precizia coloanelor tabelului sau vizualizării materializate să nu coincidă cu precizia coloanelor returnate de *subcerere*.

Printre *proprietăți\_vm* poate fi menționată opțiunea *BUILD IMMEDIATE | DEFERRED* care determină introducerea de linii în vizualizarea materializată imediat, respectiv la prima operație de reactualizare (*refresh*). În acest ultim caz, până la prima operație de actualizare, vizualizarea nu va putea fi utilizată în rescrierea cererilor. Opțiunea *IMMEDIATE* este implicită.

Prin *refresh\_vm* se specifică metodele, modurile și momentele la care sistemul va reactualiza vizualizarea materializată. Sintaxa simplificată a clauzei este următoarea:

```
{REFRESH  
[ {FAST | COMPLETE | FORCE} ] [ON {DEMAND | COMMIT}] ]  
[START WITH data] [NEXT data]  
[ WITH {PRIMARY KEY | ROWID} ]  
| NEVER REFRESH}
```

Opțiunea *FAST* indică metoda de reactualizare incrementală, care se efectuează corespunzător modificărilor survenite în tabelele *master*. Modificările sunt stocate într-un fișier *log* asociat tabelului *master*. Clauza *COMPLETE* implică reactualizarea completă, care se realizează prin reexecutarea completă a cererii din definiția vizualizării materializate. Clauza *FORCE* este implicită și presupune reactualizarea de tip *FAST*, dacă este posibil. În caz contrar, reactualizarea va fi de tip *COMPLETE*.

Clauza *ON COMMIT* indică declanșarea unei operații de reactualizare de tip *FAST* ori de câte ori sistemul permanentizează o tranzacție care operează asupra unui tabel *master* al vizualizării materializate. Aceasta ar putea duce la creșterea timpului pentru completarea operației *COMMIT*, întrucât reactualizarea va face parte din acest proces. Clauza nu este permisă pentru vizualizările materializate ce conțin tipuri obiect.

Clauza *ON DEMAND* este implicită și indică efectuarea reactualizării vizualizării materializate la cererea utilizatorului, prin intermediul procedurilor specifice din pachetul *DBMS\_MVIEW* (*REFRESH*, *REFRESH\_ALL\_MVIEWS*, *REFRESH\_DEPENDENT*).

Opțiunile *START WITH* și *NEXT* nu pot fi specificate dacă s-au precizat *ON COMMIT* sau *ON DEMAND*. Expresiile de tip dată calendaristică indicate în cadrul acestor opțiuni specifică momentul primei reactualizări automate și determină intervalul dintre două reactualizări automate consecutive.

Clauza *WITH PRIMARY KEY* este implicită și permite ca tabelele *master* să fie reorganizate fără a afecta eligibilitatea vizualizării materializate pentru reactualizarea de tip *FAST*. Tabelul *master* trebuie să conțină o constrângere *PRIMARY KEY*. Clauza nu poate fi specificată pentru vizualizări materializate obiect.

Opțiunea *WITH ROWID* asigură compatibilitatea cu tabelele *master* din versiunile precedente lui Oracle8.

Clauza *NEVER REFRESH* previne reactualizarea vizualizării materializate prin mecanisme *Oracle* sau prin proceduri. Pentru a permite reactualizarea, trebuie efectuată o operație *ALTER MATERIALIZED VIEW...REFRESH*.

Clauza *FOR UPDATE* permite actualizarea unei vizualizări materializate. *QUERY REWRITE* permite specificarea faptului că vizualizarea materializată este eligibilă pentru operația de rescriere a cererilor.

Opțiunea *AS* specifică cererea care definește vizualizarea materializată. Vizualizările materializate nu pot conține coloane de tip *LONG*. Dacă în clauza *FROM* a cererii din definiția vizualizării materializate se face referință la o altă vizualizare materializată, atunci aceasta va trebui reactualizată întotdeauna înaintea celei create în instrucțiunea curentă.

#### ➤ **Modificarea vizualizărilor materializate**

O sintaxă simplificată a comenzii *ALTER MATERIALIZED VIEW* este următoarea:

```
ALTER MATERIALIZED VIEW nume_viz_materializată  
[alter_vm_refresh]  
[ {ENABLE | DISABLE} QUERY REWRITE  
| COMPILE | CONSIDER FRESH];
```

Clauza *alter\_vm\_refresh* permite modificarea metodelor, modurilor și timpului implicit de reactualizare automată. Clauza *QUERY REWRITE*, prin opțiunile *ENABLE* și *DISABLE*, determină ca vizualizarea materializată să fie, sau nu, eligibilă pentru rescrierea cererilor.

Clauza *COMPILE* permite revalidarea explicită a vizualizării materializate.

Opțiunea *CONSIDER FRESH* indică sistemului să considere vizualizarea materializată ca fiind reactualizată și deci eligibilă pentru rescrierea cererilor.

#### ➤ **Suprimarea vizualizărilor materializate**

Pentru ștergerea unei vizualizări materializate existente în baza de date se utilizează instrucțiunea:

```
DROP MATERIALIZED VIEW nume_viz_materializată;
```

**Obs:** La ștergerea unui tabel *master*, sistemul nu va suprima vizualizările materializate bazate pe acesta. Atunci când se încearcă reactualizarea unei astfel de vizualizări materializate, va fi generată o eroare.

#### **Exerciții [V]**

46. Să se creeze și să se completeze cu înregistrări o vizualizare materializată care va conține numele joburilor, numele departamentelor și suma salariilor pentru un job, în cadrul unui departament. Reactualizările ulterioare ale acestei vizualizări se vor realiza prin reexecutarea cererii din definiție. Vizualizarea creată va putea fi aleasă pentru rescrierea cererilor.

```
CREATE MATERIALIZED VIEW job_dep_sal_pnu  
BUILD IMMEDIATE  
REFRESH COMPLETE  
ENABLE QUERY REWRITE  
AS SELECT d.department_name, j.job_title, SUM(salary) suma_salarii  
FROM employees e, departments d, jobs j  
WHERE e.department_id = d.department_id  
AND e.job_id = j.job_id  
GROUP BY d.department_name, j.job_title;
```

47. Să se creeze tabelul *job\_dep\_pnu*. Acesta va fi utilizat ca tabel sumar preexistent în crearea unei vizualizări materializate ce va permite diferențe de precizie și rescrierea cererilor.

```
CREATE TABLE job_dep_pnu (  
  job VARCHAR2(10),  
  dep NUMBER(4),  
  suma_salarii NUMBER(9,2));
```

```
CREATE MATERIALIZED VIEW vm_job_dep_pnu  
ON PREBUILT TABLE WITH REDUCED PRECISION  
ENABLE QUERY REWRITE  
AS SELECT d.department_name, j.job_title, SUM(salary) suma_salarii  
FROM employees e, departments d, jobs j  
WHERE e.department_id = d.department_id  
AND e.job_id = j.job_id  
GROUP BY d.department_name, j.job_title;
```

Să se adauge o linie nouă în această vizualizare.

48. Să se creeze o vizualizare materializată care conține informațiile din tabelul *dep\_pnu*, permite reorganizarea acestuia și este reactualizată la momentul creării, iar apoi la fiecare 5 minute.

```
CREATE MATERIALIZED VIEW dep_vm_pnu  
REFRESH FAST START WITH SYSDATE NEXT SYSDATE + 1/288  
WITH PRIMARY KEY  
AS SELECT * FROM dep_pnu;
```

Pentru reactualizarea de tip *FAST*, este necesar un fișier *log* în care să fie stocate modificările. Instrucțiunea precedentă generează eroarea „ORA-23413: table ... does not have a materialized view log”. Pentru remedierea acestei situații, înainte de crearea vizualizării, se va lansa următoarea comandă:

```
CREATE MATERIALIZED VIEW LOG ON dep_pnu;
```

49. Să se modifice vizualizarea materializată *job\_dep\_sal\_pnu* creată anterior, astfel încât metoda de reactualizare implicită să fie de tip *FAST*, iar intervalul de timp la care se realizează reactualizarea să fie de 7 zile. Nu va fi permisă utilizarea acestei vizualizări pentru rescrierea cererilor.

```
ALTER MATERIALIZED VIEW job_dep_sal_pnu  
REFRESH FAST NEXT SYSDATE + 7 DISABLE QUERY REWRITE;
```

Pentru că nu se specifică valoarea corespunzătoare opțiunii *START WITH* în clauza *REFRESH*, următoarea reactualizare va avea loc la momentul stabilit prin comanda de creare a vizualizării materializate sau prin ultima comandă de modificare a acesteia. Sistemul va reactualiza vizualizarea evaluând expresia din clauza *NEXT*, iar apoi va executa această operație o dată pe săptămână.

41. Să se șteargă vizualizările materializate create anterior.