Baze de date - Anul 2
Laborator 9

Limbajul de definire a datelor (LDD) - II :
Definirea vizualizărilor, secvențelor, indecşilor, sinonimelor.
Definirea tabelelor temporare şi a vizualizărilor materializate (opțional).

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;
- 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 inainte de definirea tabelelor, ignorând erorile la crearea vizualizării.
- Subcererea poate fi oricât de complexă dar nu poate conține 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. Daca lipseşte numele constrângerii atunci sistemul asociază un nume implicit de tip SYS_C*n* acestei constrangeri (*n* este un număr astfel încât numele constrângerii să fie unic).
- *WITH READ ONLY* asigură că prin intermediul vizualizarii nu se pot executa operații *LMD*.

➢ **Modificarea vizualizărilor** se realizează prin recrearea acestora cu ajutorul opțiunii *OR REPLACE*. Totuşi, 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 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 intotdeauna (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,
    - 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 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 implică reactualizarea tabelelor de bază? NU! Există restricții care trebuie respectate!!!

Exerciții [I]

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.
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*).

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 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;
```

Ştergeți angajatul având codul 300 prin intermediul vizualizării. Analizați 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 angajaților din departamentul 50. Analizați structura şi conținutul vizualizării.

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

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

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

d) Analizați conținutul 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.

! 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 ştergere prin intermediul vizualizării *viz_emp_dep30_pnu*? Comentați.

6. 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?

7. 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*. Dați un nume constrângerii şi regăsiți-o în vizualizarea *USER_CONSTRAINTS* din dicționarul datelor. Încercați să modificați şi 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 ştergerea 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.

9. 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';
```

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 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?

12. 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 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;
```

13. 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      nume = 'Wxyz'
WHERE    employee_id = 150;
```

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, aşa că aceeaşi 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 şi –1 pentru una descendentă;
- valoarea minimă, implicit fiind 1 pentru o secvență ascendentă şi -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* şi *DBA_SEQUENCES*.

- **Pseudocoloanele *NEXTVAL* şi *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*.

- **Ştergerea secvențelor** se face cu ajutorul comenzii *DROP SEQUENCE*.
  ***DROP SEQUENCE** nume_secventa;*

Exerciții [II]

14. Creați o secvență pentru generarea codurilor de departamente, *SEQ_DEPT_PNU*. Secvența 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 informații despre secvențele utilizatorului curent (nume, valoare minimă, maximă, de incrementare, ultimul număr generat).
16. Creați o secvență pentru generarea codurilor de angajați, *SEQ_EMP_PNU*.
17. Să se modifice toate liniile din *EMP_PNU* (dacă nu mai există, îl recreeați), regenerând codul angajaților astfel încât să utilizeze secvența *SEQ_EMP_PNU* şi să avem continuitate în codurile angajaților.
18. Să se insereze câte o inregistrare nouă în *EMP_PNU* şi *DEPT_PNU* utilizând cele 2 secvențe create.
19. Să se selecteze valorile curente ale celor 2 secvențe.

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

20. Ştergeți secvența *SEQ_DEPT_PNU*.

III. Definirea indecşilor

- 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.

- Indecşii :
  - 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 şi 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 şterşi nu afectează conținutul tabelelor sau comportamentul altor indecşi;
  - sunt menținuți şi utilizați automat de către *server*-ul *Oracle*;
  - la ştergerea unui tabel, sunt şterşi şi indecşii asociați acestuia.

- Tipuri de indecşi:
  - indecşi normali (indecsi ce folosesc B-arbori);
  - indecşi *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*;
  - indecşi partiționați, care constau din partiții corespunzătoare valorilor ce apar în coloanele indexate ale tabelului;
  - indecşi bazați pe funcții (pe expresii). Aceştia permit construcția cererilor care evaluează valoarea returnată de o expresie, expresie ce poate conține funcții predefinite sau definite de utilizator.

- Indecşii 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 şi 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 indecşi şi despre coloanele implicate în indecşi 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…]);*

- **Modificarea unui index** se face prin comada ***ALTER INDEX***.

- **Eliminarea unui index** se face prin comanda: ***DROP INDEX** nume_index;*

Exerciții [III]

21. Să se creeze un index (normal, neunic) *IDX_EMP_LAST_NAME_PNU*, asupra coloanei *last_name* din tabelul *emp_pnu*.
22. Să se creeze indecşi unici asupra codului angajatului (*employee_id*) şi asupra combinației *last_name*, *first_name*, *hire_date* prin două metode (automat şi manual).

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

23. Creați un index neunic asupra coloanei *department_id* din *EMP_PNU* pentru a eficientiza *join*-urile dintre acest tabel si *DEPT_PNU*.
24. Prespupunând că se fac foarte des căutari *case insensitive* asupra numelui departamentului şi asupra numelui angajatului, definiți doi indecşi bazați pe expresiile *UPPER(department_name)*, respectiv *LOWER(last_name)*.
25. Să se selecteze din dicționarul datelor numele indexului, numele coloanei, poziția din lista de coloane a indexului şi proprietatea de unicitate a tuturor indecşilor definiți pe tabelele *EMP_PNU* şi *DEPT_PNU*.
26. Eliminați indexul de la exercițiul 23.

IV. 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 şi 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 [IV]

27. Creați un sinonim public *EMP_PUBLIC_PNU* pentru tabelul *EMP_PNU*.
28. Creați un sinonim *V30_PNU* pentru vizualizarea *VIZ_EMP30_PNU*.
29. 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?
30. Eliminați sinonimele create anterior prin intermediul unui script care să selecteze numele sinonimelor din *USER_SYNONYMS* care au terminația "*pnu*" şi să genereze un fişier cu comenzile de ştergere corespunzătoare.

V. 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, ştergând 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 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 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 ataşată acestuia.

Exerciții [V]

31. 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.
32. Creați un tabel temporar *TEMP_SESIUNE_PNU*, cu datele persistente pe durata sesiunii. Cerințele sunt cele de la punctul 1.
33. Inițiați încă o sesiune SQL*Plus. Listați structura şi conținutul tabelelor create anterior. Introduceți încă o linie în fiecare din cele două tabele.
34. Ştergeți tabelele create anterior. Cum se poate realiza acest lucru?

Obs: Pentru deconectarea şi reconectarea la SQL*Plus, fără închiderea acestuia, se utilizează comenzile:

DISCONNECT
CONNECT g233/baze@o9i

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

```
CREATE GLOBAL TEMPORARY TABLE angajati_azi_pnu
  ON COMMIT PRESERVE ROWS
  AS SELECT  *
    FROM    emp_pnu
    WHERE   hire_date = SYSDATE;
```

36. Inserați o nouă înregistrare în tabelul *angajati_azi_pnu*. Incercați actualizarea tipului de date al coloanei *last_name* a tabelului *angajati_azi_pnu*.

VI. Definirea vizualizărilor materializate [opțional]

- O vizualizare materializată, cunoscută în versiunile anterioare sub numele de clişeu (*snapshot*), este un obiect al schemei ce stochează rezultatele unei cereri şi care este folosit pentru a rezuma, calcula, replica şi distribui date. Vizualizările materializate sunt utile în domenii precum *data warehouse*, suportul deciziilor şi 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 recunoaşte automat situațiile în care o astfel de vizualizare poate şi 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 indecşilor:
  - consumă spațiu de stocare;
  - trebuie reactualizate când datele din tabelele de bază sunt modificate;
  - îmbunătățesc performanța execuției instrucțiunilor *SQL* dacă folosite pentru rescrierea cererilor;
  - sunt transparente aplicațiilor *SQL* şi utilizatorilor.
- Spre deosebire de indecşi, vizualizările materializate pot fi accesate utilizând instrucțiuni *SELECT* şi pot fi actualizate prin instrucțiunile *INSERT*, *UPDATE*, *DELETE*.
- Asupra unei vizualizări materializate se pot defini unul sau mai mulți indecşi.
- Similar vizualizărilor obişnuite, asupra celor materializate se pot defini constrângerile *PRIMARY KEY*, *UNIQUE* şi *FOREIGN KEY*. Singura stare validă a unei constrângeri este *DISABLE NOVALIDATE*.

Pentru compatibilitate cu versiunile anterioare, cuvintele cheie *SNAPSHOT* şi *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 şi 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 fişier *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 şi presupune reactualizarea de tip *FAST*, dacă este posibil. În caz contrar, reactualizarea va fi de tip *COMPLETE*.

Clauza *ON COMMIT* indică declanşarea 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 creşterea 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 şi 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* şi *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 şi determină intervalul dintre două reactualizări automate consecutive.

Clauza *WITH PRIMARY KEY* este implicită şi 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 defineşte 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 şi timpului implicit de reactualizare automată. Clauza *QUERY REWRITE*, prin opțiunile *ENABLE* şi *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 şi deci eligibilă pentru rescrierea cererilor.

- **Suprimarea** vizualizărilor materializate

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

  DROP MATERIALIZED VIEW *nume_viz_materializată*;

Obs: La ştergerea 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 [VI]

37. Să se creeze şi să se completeze cu înregistrări o vizualizare materializată care va conține numele joburilor, numele departamentelor şi 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;
```

38. 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 şi 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.

39. Să se creeze o vizualizare materializată care conține informațiile din tabelul *dep_pnu*, permite reorganizarea acestuia şi 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 fişier *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;
```

40. 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 şteargă vizualizările materializate create anterior.