

LABORATOR 8 SQL

Vizualizări. Vizualizări materializate. Secvențe. Indecși. Sinonime. Tabele temporare. Cluster-e.

I. Vizualizări

- Sunt tabele virtuale construite pe baza unor tabele sau vizualizări, denumite tabele de bază.
- Nu conțin date; sunt niște imagini logice asupra datelor din tabelele de bază.
- Sunt definite de o cerere *SQL*, de aceea mai sunt denumite și cereri stocate.
- Permit:
 - restricționarea accesului la date;
 - simplificarea unor cereri complexe;
 - prezentarea unor imagini diferite asupra datelor.
- Pot fi simple sau complexe.
 - Asupra vizualizărilor simple se pot realiza operații *LMD*.
 - Asupra vizualizărilor complexe nu sunt posibile operații *LMD* decât dacă au definiți declanșatori de tip *INSTEAD OF*.

Caracteristici	Simple	Complexe
Număr tabele de bază	Un singur tabel	Unul sau mai multe tabele
Conține funcții	Nu	Da
Conține grupări de date	Nu	Da

- Sintaxa simplificată a comenzii *CREATE VIEW*

```
CREATE [OR REPLACE] [FORCE | NOFORCE]
VIEW nume_view [(alias, alias, ..)]
AS subcerere
[WITH CHECK OPTION [CONSTRAINT nume_constr]]
[WITH READ ONLY [CONSTRAINT nume_constr]];
```

- *FORCE* permite crearea vizualizării înainte de a defini tabelele de bază;
 - subcererea poate fi oricât de complexă, dar nu poate conține clauza *ORDER BY*;
 - *WITH CHECK OPTION* permite inserarea și modificarea prin intermediul vizualizării numai a înregistrărilor ce sunt accesibile vizualizării; dacă numele constrângerii lipsește atunci sistemul asociază acesteia un nume implicit de tip *SYS_Cn*;
 - *WITH READ ONLY* asigură că prin intermediul vizualizării nu se pot executa operații *LMD*; dacă numele constrângerii lipsește atunci sistemul asociază acesteia un nume implicit de tip *SYS_Cn*.
- Nu se pot realiza operații *LMD* asupra unor vizualizări ce conțin:
 - funcții grup,

- clauza *GROUP BY* sau *HAVING*,
 - cuvântul cheie *DISTINCT*,
 - pseudocoloana *ROWNUM* etc.
- Nu se pot actualiza prin intermediul unei vizualizări:
 - coloane ale căror valori rezultă prin calcule sau pe care sunt aplicate funcții;
 - coloane ale căror valori nu respectă constrângerile definite asupra tabelelor de bază.
 - Pentru vizualizările bazate pe mai multe tabele, orice operație *LMD* poate modifica datele doar dintr-un singur tabel de bază. Acest tabel este cel protejat prin cheie (*key preserved*).
 - Eliminarea unei vizualizări se realizează prin comanda

```
DROP VIEW nume_vizualizare;
```

- Informații despre vizualizările definite de utilizatorul curent se găsesc în vizualizarea

```
USER_VIEWS
```

Observații:

- Reactualizarea tabelelor de bază implică reactualizarea corespunzătoare a vizualizărilor.
 - Reactualizarea vizualizărilor nu implică întotdeauna reactualizarea tabelelor de bază.
1.
 - a. Definiți vizualizarea *v_emp_**** care va conține codul și numele salariaților din tabelul *emp_****. Coloanele vizualizării vor fi denumite *cod*, respectiv *nume*.
 - b. Afișați conținutul acesteia.
 - c. Inserați o nouă înregistrare în această vizualizare. Ce observați?
 - d. Ștergeți vizualizarea definită anterior.

```
CREATE OR REPLACE VIEW v_emp_*** (cod, nume)
AS SELECT employee_id, last_name
FROM emp_***;
```

sau

```
CREATE OR REPLACE VIEW v_emp_***
AS SELECT employee_id cod, last_name nume
FROM emp_***;
```

Observație: Vizualizarea trebuie să conțină coloanele care în tabelul de bază au definită constrângerea *NOT NULL* și nu au specificată o valoare *DEFAULT*. Altfel, chiar dacă tipul vizualizării permite operații *LMD*, acestea nu vor fi posibile din cauza nerespectării constrângerilor *NOT NULL* definite asupra tabelului de bază.

2. a. Definiți vizualizarea *v_emp_**** care va conține codul, numele, emailul, data angajării, salariul și codul jobului salariaților din tabelul *emp_****.
 - b. Analizați structura vizualizării.
 - c. Afișați conținutul acesteia.
 - d. Inserați o nouă înregistrare în această vizualizare folosind pentru codul angajatului valoarea 400.
 - e. Verificați că noua înregistrare a fost inserată în tabelul de bază.
 - f. Măriți cu 1000 salariul angajatului având codul 400 folosind vizualizarea. Ce efect va avea această acțiune asupra tabelului de bază?
 - g. Ștergeți angajatul având codul 400 din vizualizare. Ce efect va avea această acțiune asupra tabelului de bază?

3. a) Definiți vizualizarea *v_emp_dept_**** care va conține coloanele *employee_id*, *last_name*, *hire_date*, *job_id*, *department_id* din tabelul *emp_**** și coloana *department_name* din tabelul *dept_****.

```
CREATE VIEW v_emp_dept_***
AS SELECT employee_id, last_name, email, hire_date, job_id,
        e.department_id, department_name
FROM emp_*** e, dept_*** d
WHERE e.department_id = d.department_id;
```

- b) Încercați să inserați următoarea înregistrare (500, 'N2', 'E2',SYSDATE,'SA_REP',30, 'Administrativ') în vizualizarea creată anterior.

- c) Care dintre coloanele vizualizării *v_emp_dept_**** sunt actualizabile?

```
SELECT *
FROM USER_UPDATABLE_COLUMNS
WHERE table_name = UPPER('v_emp_dept_***');
```

- d) Adăugați tabelului *emp_**** constrângerea de cheie externă care referă tabelul *dept_****, apoi verificați ce coloane din vizualizarea *v_emp_dept_**** sunt actualizabile.
- e) Recreați vizualizarea *v_emp_dept_****, apoi verificați ce coloane sunt actualizabile.
- f) Inserați o înregistrare în vizualizare.

Observație: Tabelul ale cărui coloane sunt actualizabile este tabelul protejat prin cheie.

- g) Ce efect are o operație de ștergere realizată asupra vizualizării *v_emp_dept_****? Comentați. Pentru exemplificare ștergeți prin vizualizare angajatul cu codul 500.

4. Definiți o vizualizare, *v_emp_s_**** care va conține detalii despre angajații corespunzători departamentelor care încep cu litera S. Se pot insera/actualiza înregistrări prin intermediul acestei vizualizări? În care dintre tabelele de bază? Ce se întâmplă dacă este ștersă o înregistrare din vizualizare?

```
CREATE OR REPLACE VIEW v_emp_s_***
AS SELECT employee_id, last_name, email, hire_date, job_id,
        e.department_id, department_name
FROM emp_*** e, dept_*** d
WHERE e.department_id = d.department_id
AND UPPER(department_name) LIKE 'S%';
```

- prin intermediul vizualizării se pot insera înregistrări în tabelul protejat prin cheie

```
INSERT INTO v_emp_s_*** (employee_id, last_name, email,
        hire_date, job_id)
VALUES (222, 'n222', 'e222', SYSDATE, 'SA_REP');
```

- următoarea comandă nu este permisă, deoarece se încearcă actualizarea unei coloane care nu provine din tabelul protejat prin cheie

```
UPDATE v_emp_s_***
SET department_name = 'Vanzari'
WHERE department_id=80;
```

- următoarea comandă nu afectează nicio înregistrare, deoarece pentru angajatul 222 coloana *department_id* are valoarea *null*, deci nu se regăsește în vizualizare

```
DELETE FROM v_emp_s_***
WHERE employee_id=222;
```

```
SELECT *
FROM emp_***
WHERE employee_id=222;
```

- Definiți vizualizarea *v_dept_**** care va conține codul și numele departamentului, numărul de angajați din departament și suma alocată pentru plata salariilor. Această vizualizare permite actualizări?
- a) Creați vizualizarea *v_emp30_**** care să conțină numele, emailul, data angajării, salariul, codul jobului și codul departamentului celor care lucrează în departamentul 30. În această vizualizare nu se va permite modificarea sau inserarea înregistrărilor ce nu sunt accesibile acestuia. Dați un nume constrângerii.

```
CREATE VIEW v_emp30_***
AS SELECT employee_id, last_name, email, hire_date, salary,
        job_id, department_id
FROM emp_***
WHERE department_id=30
WITH CHECK OPTION CONSTRAINT ck_option1_***;
```

- b) Listați structura și conținutul vizualizării `v_emp30_***`.
 - c) Încercați prin intermediul vizualizării inserarea unui angajat în departamentul 10, respectiv a unui angajat în departamentul 30. Ce observați?
 - d) Încercați prin intermediul vizualizării modificarea codului departamentului unui angajat.
7. Definiți vizualizarea `v_dept_***` asupra tabelului `dept_***` care să nu permită efectuarea operațiilor *LMD* asupra sa. Testați operațiile de inserare, modificare și ștergere asupra acestei vizualizări.

```
CREATE VIEW v_dept_***  
AS SELECT *  
FROM dept_***  
WITH READ ONLY;
```

8. Definiți vizualizarea `v_emp_info_***` asupra tabelului `emp_***` care va conține codul, numele, prenumele, emailul și numărul de telefon ale angajaților companiei. Se va impune unicitatea valorilor coloanei email și constrângerea de cheie primară pentru coloana corespunzătoare codului angajatului. Prin intermediul acestei vizualizări se pot insera valori care să încalce constrângerile impuse?

Observație: Constrângerile asupra vizualizărilor pot fi definite numai în modul *DISABLE NOVALIDATE*. Aceste opțiuni trebuie specificate la declararea constrângerii, nefiind permisă precizarea altor stări.

```
CREATE VIEW v_emp_info_*** (  
    employee_id,  
    first_name,  
    last_name,  
    email UNIQUE DISABLE NOVALIDATE,  
    phone_number,  
    CONSTRAINT ccp_*** PRIMARY KEY (employee_id) DISABLE NOVALIDATE)  
AS  
SELECT employee_id, first_name, last_name, email, phone_number  
FROM emp_***;
```

9. Definiți vizualizarea `v_manager_***` care va conține informații despre toți managerii. Apoi, adăugați o constrângere de cheie primară asupra acestei vizualizări.

```
ALTER VIEW v_manager_***  
ADD CONSTRAINT ccp_v_manager PRIMARY KEY (employee_id)  
DISABLE NOVALIDATE;
```

10. a. Afișați structura vizualizării *USER_VIEWS*.
b. Afișați informații despre vizualizările create.

Observație: Coloana *TEXT* a vizualizării *USER_VIEWS* este de tip *LONG*.

În cazul selectării unei coloane de tip *LONG* poate fi utilizată comanda *SQL*Plus SET LONG n*, unde *n* reprezintă numărul de caractere afișate.

II. Vizualizări materializate - FACULTATIV

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.

```
CREATE MATERIALIZED VIEW nume
[refresh] [FOR UPDATE]
[ {DISABLE | ENABLE} QUERY REWRITE] AS subcerere;
```

```
refresh = {REFRESH
  [ {FAST | COMPLETE | FORCE} ] [ON {DEMAND | COMMIT} ]
  [START WITH data] [NEXT data]
  | NEVER REFRESH}
```

- *FAST* indică metoda de reactualizare incrementală, care se efectuează corespunzător modificărilor survenite în tabelele de bază.
- *COMPLETE* implică reactualizarea completă, care se realizează prin reexecuția completă a cererii din definiția vizualizării materializate.
- *FORCE* este implicită și presupune reactualizarea de tip *FAST*, dacă este posibil. În caz contrar, reactualizarea va fi de tip *COMPLETE*.
- *ON COMMIT* indică declanșarea unei operații de reactualizare ori de câte ori sistemul permanentizează o tranzacție care operează asupra unui tabel de bază al vizualizării materializate.
- *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*).
- *START WITH* și *NEXT* nu pot fi specificate dacă s-au precizat clauzele *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.
- *NEVER REFRESH* previne reactualizarea vizualizării materializate prin mecanisme *Oracle* sau prin proceduri. Pentru a permite reactualizarea, trebuie efectuată o operație *ALTER MATERIALIZED VIEW...REFRESH*.
- *FOR UPDATE* permite actualizarea unei vizualizări materializate.
- *QUERY REWRITE* permite specificarea faptului că vizualizarea materializată poate fi utilizată pentru rescrierea cererilor.

- 11. a.** Creați vizualizarea materializată *vm_dept_**** care va conține codul și numele departamentului, numărul de angajați din departament și suma alocată pentru plata salariilor. Reactualizările ulterioare ale acestei vizualizări se vor realiza prin reexecuția cererii din definiție, după permanentizarea modificărilor asupra tabelelor de bază.
- b.** Inserați un nou angajat în departamentul 110 (tabelul *emp_****) și observați modificările apărute în vizualizare.

```
CREATE MATERIALIZED VIEW vm_dept_***
  REFRESH ON COMMIT
AS SELECT e.department_id, department_name,
          COUNT(*) nr_angajati, SUM(salary) val_salarii
FROM   emp_*** e, dept_*** d
WHERE  e.department_id = d.department_id
GROUP BY e.department_id, department_name;

SELECT * FROM vm_dept_***;

INSERT INTO emp_*** (employee_id, last_name, email, hire_date,
                    job_id, salary, department_id)
VALUES (500, 'x', 'x', sysdate, 'x', 1000, 110);

SELECT * FROM vm_dept_***;
COMMIT;
SELECT * FROM vm_dept_***;
```

- 12.** Definiți o vizualizare materializată care conține informațiile din tabelul *dept_**** și este reactualizată la momentul creării, iar apoi la fiecare 5 minute.

```
CREATE MATERIALIZED VIEW vm_dept_***
  REFRESH START WITH SYSDATE NEXT SYSDATE + 1/288
AS SELECT *
FROM dept_***;
```

- 13.** Creați și populați cu înregistrări o vizualizare materializată care va conține numele joburilor, numele departamentelor și suma salariilor la nivel de job, în cadrul departamentelor. Reactualizările ulterioare ale acestei vizualizări se vor realiza prin reexecuția cererii din definiție.

```
CREATE MATERIALIZED VIEW job_dep_sal_pnu
  BUILD IMMEDIATE
  REFRESH COMPLETE
AS SELECT d.department_name, j.job_title,
```

```

SUM(salary) suma_salariei
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;
```

III. Secvențe

O secvență este un obiect al bazei de date ce permite generarea de numere întregi unice cu scopul de a fi folosiți ca valori pentru cheia primară sau coloane numerice unice. Secvențele sunt independente de tabele.

Sintaxa comenzii CREATE SEQUENCE este:

```

CREATE SEQUENCE nume_secvență
[INCREMENT BY n]
[START WITH valoare_start]
[ {MAXVALUE valoare_maximă | NOMAXVALUE} ]
[ {MINVALUE valoare_minimă | NOMINVALUE} ]
[ {CYCLE | NOCYCLE} ]
[ {CACHE n | NOCACHE} ];
```

- *INCREMENT BY* specifică diferența dintre valorile succesive ale secvenței (valoare implicită 1).
- *START WITH* specifică primul număr care va fi generat de secvență (valoare implicită 1).
- *MAXVALUE*, *MINVALUE* precizează valoarea maximă, respectiv minimă pe care o poate genera secvența. Opțiunile *NOMAXVALUE*, *NOMINVALUE* sunt implicite. *NOMAXVALUE* specifică valoarea maximă de 10^{27} pentru o secvență crescătoare și -1 pentru o secvență descrescătoare. *NOMINVALUE* specifică valoarea minimă 1 pentru o secvență crescătoare și -10^{26} pentru o secvență descrescătoare.

- *CYCLE* și *NOCYCLE* specifică dacă secvența continuă să genereze numere după obținerea valorii maxime sau minime. *NOCYCLE* este opțiunea implicită.

- *CACHE n* precizează numărul de valori pe care *server-ul Oracle* le prealocă și le păstrează în memorie. În mod implicit, acest număr de valori este 20. Opțiunea *CACHE* permite accesul mai rapid la valorile secvenței care sunt păstrate în memorie. Aceste valori sunt generate la prima referință asupra secvenței. Fiecare valoare din secvență se furnizează din secvența memorată. După utilizarea ultimei valori prealocate secvenței, următoarea solicitare a unei valori determină încărcarea unui alt set de numere în memorie. Pentru a nu fi prealocate și reținute în memorie astfel de valori, se utilizează opțiunea *NOCACHE*.

Pseudocoloanele NEXTVAL și CURRVAL permit utilizarea secvențelor.

- 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 returnează valoarea curentă a secvenței.

Pseudocoloanele NEXTVAL și CURRVAL 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.

Pseudocoloanele NEXTVAL și CURRVAL 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 realizează cu ajutorul comenzii DROP SEQUENCE.

DROP SEQUENCE nume_secv;

15. Să se creeze o secvență care are pasul de incrementare 10 și începe de la 10, are ca valoare maximă 10000 și nu ciclează.

```
CREATE SEQUENCE sec_***  
INCREMENT BY 10  
START WITH 10  
MAXVALUE 10000  
NOCYCLE;
```

16. Să se modifice toate liniile din tabelul emp_***, regenerând codul angajaților astfel încât să utilizeze secvența sec_emp***.

```
UPDATE emp_***  
SET employee_id = sec_emp***.NEXTVAL;  
ROLLBACK;
```

17. Să se introducă un nou salariat în tabelul emp_*** folosindu-se pentru codul salariatului secvența creată.

```
INSERT INTO emp_*** (employee_id,last_name,email,hire_date,job_id)  
VALUES (sec_emp***.NEXTVAL, 'x', 'x', sysdate, 'x');  
ROLLBACK;
```

18. Să se afișeze valoarea curentă a secvenței.

```
SELECT sec_***.CURRVAL valoare  
FROM DUAL;
```

Exercițiu

a) Creați o secvență pentru generarea codurilor de departamente, seq_dept_***. Secvența va începe de la 200, va crește cu 10 la fiecare pas și va avea valoarea maximă 20000, nu va cicla.

b) Să se selecteze informații despre secvențele utilizatorului curent (nume, valoare minimă, maximă, de incrementare, ultimul număr generat). Se va utiliza vizualizarea *user_sequences*.

c) Să se insereze o înregistrare nouă în *dept_**** utilizând secvența creată. Anulați modificările efectuate.

d) Să se selecteze valoarea curentă a secvenței.

e) Să se ștergă secvența.

IV. Indecși

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;
- sunt șterși odată cu eliminarea tabelului asociat.

Indecșii pot fi creați :

- automat: la definirea unei constrângeri PRIMARY KEY sau UNIQUE;
- manual: cu ajutorul comenzii CREATE INDEX.

Se creează un index atunci când:

- o coloană conține un domeniu mare de valori;
- o coloană conține un 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;
- tabelul nu este modificat frecvent.

Sintaxa comenzii CREATE INDEX:

CREATE [UNIQUE] INDEX nume_index

ON tabel (coloana1 [, coloana2...]);

Modificarea unui index se face prin comanda ALTER INDEX.

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

19. Să se creeze un index neunic, *emp_last_name_idx_****, asupra coloanei *last_name* din tabelul *emp_****.

20. Să se creeze indecși unici asupra codului angajatului (*employee_id*) și asupra combinației *last_name*, *first_name*, *hire_date*.
21. Creați un index neunic asupra coloanei *department_id* din *emp_**** pentru a eficientiza joinurile dintre acest tabel și *dept_****.
22. a) Presupunând că se fac foarte des căutări asupra numelui departamentului definiți un index bazat pe expresia `UPPER(department_name)`.
- b) Ștergeți indexul creat anterior.

V. Sinonime

Pentru a simplifica accesul la obiecte se pot asocia sinonime acestora.

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.

Comanda pentru crearea sinonimelor este:

```
CREATE [PUBLIC] SYNONYM nume_sinonim  
FOR obiect;
```

Eliminarea sinonimelor se face prin comanda `DROP SYNONYM nume_sinonim;`

23. Creați un sinonim public *se_**** pentru tabelul *emp_****.
24. Creați un sinonim pentru vizualizarea *vm_dept_****.
25. Creați un tabel *test_**** care să conțină 3 coloane, a de tip numeric, b de tip șir de caractere și c de tip data calendaristică. Introduceți câteva înregistrări. Creați un sinonim pentru acest tabel. Utilizați sinonimul pentru accesarea datelor din tabel. Redenumiți tabelul (`RENAME .. TO ..`). Încercați din nou să utilizați sinonimul pentru a accesa datele din tabel. Ștergeți sinonimul creat anterior.

VI. Tabele temporare - FACULTATIV

```
CREATE [GLOBAL TEMPORARY] TABLE [schema.]nume_tabel  
( {nume_coloană_1 tip_date [DEFAULT expresie]  
  [constr_coloană [constr_coloană] ...]  
  | {constr_tabel }  
  [, {nume_coloană_2...} ] )  
[ON COMMIT {DELETE | PRESERVE} ROWS];
```

- un tabel temporar stochează date specifice unei sesiuni;
- datele sunt stocate în tabel numai pe durata unei tranzații 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;
- `ON COMMIT` determină dacă datele din tabelul temporar persistă pe durata unei tranzații sau a unei sesiuni;

- 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);
- PRESERVE ROWS se specifică pentru a defini un tabel temporar specific unei sesiuni, caz în care sistemul trunchiază tabelul la terminarea sesiunii.

26. a) Să se creeze un tabel temporar *angajati_**** care să conțină lista salariaților angajați în anul 1990. Fiecare sesiune va permite stocarea în acest tabel a salariaților angajați în anul 1990. La sfârșitul sesiunii, aceste date vor fi șterse.

b) Inserați o nouă înregistrare în tabelul temporar creat. Ștergeți tabelul creat.

Exercițiu

- a)** Creați un tabel temporar TEMP_TRANZ_***, 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.
- b)** Creați un tabel temporar TEMP_SESIUNE_***, cu datele persistente pe durata sesiunii. Cerințele sunt cele de la punctul a.
- c)** Inițiați încă o sesiune SQL*Plus. Listați structura și conținutul tabelelor create anterior. Introduceți încă o linie în fiecare din cele două tabele.
- d)** Ștergeți tabelele create anterior. Cum se poate realiza acest lucru? (Eliminarea unui tabel temporar se poate realiza doar dacă nu există nici o sesiune atașată lui).

VII. Grupări (*cluster-e*) - FACULTATIV

Cluster-ul este o regrupare fizică a două sau mai multe tabele, relativ la una sau mai multe coloane, cu scopul mării 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]

Suprimarea unui *cluster* din baza de date se face prin comanda:

DROP CLUSTER nume_cluster;

Suprimarea unui *cluster* din baza de date, a tuturor tabelelor definite relativ la acel *cluster* și constrângerile lor de integritate se face 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).

- 27. a)** Să se creeze un *cluster* numit *personal_**** având cheia *cluster* departament, dimensiunea 512 bytes.

```
CREATE CLUSTER personal_***  
    (department NUMBER(4))  
SIZE 512  
STORAGE (initial 100K next 50K);
```

- b)** Să se creeze un index pe cheia clusterului *personal_****.

```
CREATE INDEX idx_personal_*** ON CLUSTER personal_***;
```

- c)** Să se adauge *cluster*-ului *personal_**** două tabele: tabelul *dept_10_**** care conține angajații din departamentul 10 și tabelul *dept_20_**** care conține angajații din departamentul 20.

```
CREATE TABLE dept_10_***  
    CLUSTER personal_*** (department_id)  
AS SELECT * FROM employees WHERE department_id = 10;
```

```
CREATE TABLE dept_20_***  
    CLUSTER personal_*** (department_id)  
AS SELECT * FROM employees WHERE department_id = 20;
```

- e)** Scoateți tabelul *dept_20_**** din clusterul *personal_****, apoi ștergeți *cluster*-ul.

Obs. Pentru a scoate un tabel dintr-un *cluster* sunt parcurse următoarele etape:

- se creează un nou tabel, în afara *cluster*-ului, prin duplicarea celui vechi;
- se șterge tabelul din *cluster*;
- se suprimă *cluster*-ul.

```
CREATE TABLE dept_20c_***  
AS SELECT * FROM dept_20_***;  
  
DROP TABLE dept_20_***;  
  
RENAME dept_20c_*** TO dept_20_***;  
  
DROP CLUSTER personal_***  
INCLUDING TABLES CASCADE CONSTRAINTS;  
  
SELECT * FROM dept_10_***;  
SELECT * FROM dept_20_***;
```

VIII. Informații din dicționarul datelor

28. Informații referitoare la secvențe:

```
SELECT  sequence_name, min_value, max_value,  
        increment_by, last_number  
FROM    user_sequences;
```

29. Informații referitoare la indecși:

```
SELECT index_name ,table_name  
FROM   user_indexes;
```

```
SELECT index_name  
FROM   user_ind_columns;
```

```
SELECT a.index_name, a.column_name,a.column_position poz,  
       b.uniqueness  
FROM   user_indexes b, user_ind_columns a  
WHERE  a.index_name = b.index_name  
AND    a.table_name = 'EMP_***';
```

30. Informații referitoare la sinonime:

```
SELECT synonym_name, table_owner, table_name  
FROM   user_synonyms;
```