10. Limbajul de definire a datelor(LDD). Vizualizări, secvenţe, indecşi, sinonime

Vizualizări

Vizualizările sunt tabele virtuale care sunt construite pe baza unor tabele sau vizualizări, denumite tabele de bază. Ele nu conţin date ci sunt ca nişte imagini logice asupra datelor din tabelele de bază. Sunt definite de o cerere SQL, de aceea mai sunt denumite şi cereri stocate.

Avantajele utilizării vizualizărilor:
- restricţionarea accesului la date;
- simplificarea unor cereri complexe;
- prezentarea de diferite imagini asupra datelor.

Vizualizările se pot fi simple sau complexe. Asupra vizualizărilor simple se pot realiza operaţii *LMD*. Asupra vizualizărilor complexe nu sunt posibile operaţii *LMD* în toate cazurile. Pentru vizualizările bazate pe mai multe tabele, orice operaţie *INSERT*, *UPDATE* sau *DELETE* poate modifica datele doar din unul din tabelele de bază. Acest tabel este cel protejat prin cheie (key preserved). Un tabel protejat prin cheie este un tabel în care valorile cheilor primare sau valorile cheilor unice sunt unice şi în tabelul obţinut prin join (în vizualizare). Tabelul următor cuprinde o comparaţie între vizualizările simple şi complexe.

Caracteristici	Simple	Complexe
Număr de tabele de baza	Un singur tabel	Unul sau mai multe tabele
Conţine funcţii	Nu	Da
Conţine grupări de date	Nu	Da

Sintaxa simplificată a comenzii *CREATE VIEW* este:

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 vizualizarea înainte de a defini tabelele de bază;
- subcererea poate fi oricât de complexă dar nu poate conţine clauza *ORDER BY*;
- *WITH CHECK OPTION* permite inserarea şi modificarea prin intermediul vizualizării numai a liniilor ce sunt accesibile vizualizării; dacă lipseşte numele constrângerii atunci sistemul asociază un nume implicit de tip SYS_Cn acestei constrângeri;

- *WITH READ ONLY* asigură că prin intermediul vizualizării nu se pot executa operaţii *LMD*.

Nu se pot realiza operaţii *LMD* în vizualizări ce conţin:
- funcţii grup,
- clauza *GROUP BY* sau *HAVING*, *CONNECT BY*, *START WITH*
- cuvântul cheie *DISTINCT*,
- pseudocoloana *ROWNUM*,
- coloane *NOT NULL* din tabelul de bază, care nu sunt incluse în coloanele vizualizării.
- linii ce nu sunt accesibile vizualizării (în cazul utilizării clauzei *WITH CHECK OPTION*).

Eliminarea unei vizualizări se face prin comanda *DROP VIEW* :

DROP VIEW nume_viz;

Observaţie:

Subcererile temporare caracterizate de un alias ce apar în comenzile *SELECT*, *INSERT*, *UPDATE*, *DELETE*, *MERGE* se numesc vizualizări inline (de exemplu o subcerere utilizată în clauza *FROM* a comenzii *SELECT*). Spre deosebire de vizualizările propriu-zise acestea nu sunt considerate obiecte ale schemei şi sunt entităţi temporare.

Observaţie:

Reactualizarea tabelelor implică şi reactualizarea corespunzătoare a vizualizărilor. Reactualizarea vizualizărilor nu implică întotdeauna reactualizarea tabelelor de bază.

Constrângeri asupra vizualizărilor: pot fi specificate explicit numai constrângerile *UNIQUE*, *PRIMARY KEY*, *FOREIGN KEY*. Constrângerea de tip *CHECK* poate fi realizată prin pecizarea clauzei *WITH CHECK OPTIONS*.

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

Modificarea vizualizărilor se realizează prin recrearea acestora cu ajutorul opţiunii *OR REPLACE*. Totuşi, din începând cu Oracle9i, este posibilă utilizarea comenzii *ALTER VIEW* pentru adăugarea de constrângeri vizualizării.

Pentru a se obţine informaţii referitoare la vizualizările definite, se pot interoga vizualizările *USER_ VIEWS* şi *ALL_ VIEWS*.

Exemplul 1 *Să se creeze vizualizarea v_jobs_*** *care să conţină codul şi numele jobului şi media dintre salariul minim şi cel maxim. Este obligatorie atribuirea unui alias coloanelor provenite din expresii.*

```
CREATE VIEW v_jobs_***
AS SELECT job_id, job_title, (min_salary + max_salary) / 2 as medie
FROM jobs_***;
```

Exemplul 2 *Următoarea instrucţiune este corectă şi are ca rezultat adaugarea unei noi înregistrări în tabelul jobs_ ***.*

*INSERT into v_jobs_ ***(job_id,job_title)*
VALUES (9,'AS Programmer');

Exemplul 3 *Dacă s-ar omite din listă coloana job_title instrucţiunea utilizată în exemplul precedent ar genera eroare. În definirea tabelului jobs s-a utilizat constângerea: job_title NOT NULL. Nu se poate adăuga vizualizării o linie în care să se precizeze valoarea coloanei virtuale "medie". Instrucţiunea de mai jos este eronată deoarece nu se poate insera o valoare într-o caloană bazată pe o expresie.*

*INSERT into v_jobs_ ****
VALUES (9,'AS Programmer',1000);

Exemplul 4 *Să se creeze o vizualizare v_dept_loc_ **** care să conţină codul departamentelor, numele departamentelor şi numele oraşului departamentului.*

*CREATE VIEW v_dept_loc_ ****
AS SELECT department_id, department_name, city
FROM departments d, locations l
WHERE d.location_id = l.location_id;

Exemplul 5 *Tabelul departments este key-preserved. Sunt permise următoarele instrucţiuni:*

*INSERT INTO v_dept_loc_ ***(department_id,department_name)*
VALUES(300,'Project Management');

*UPDATE v_dept_loc_ ****
SET department_name = 'Project Management'
WHERE department_id = 270;

Exemplul 6 *Să se creeze o vizualizare care să conţină numele_departmentului codul managerului (unic). Să se adauge o constângere UNIQUE asupra numelui.*

*CREATE VIEW v_dept_ ****
(nume ,manager UNIQUE DISABLE NOVALIDATE)
AS SELECT department_name, manager_id
FROM departments;

*ALTER VIEW v_dept_ ****
*ADD CONSTRAINT u_v_dept_ **** UNIQUE(nume) DISABLE NOVALIDATE*

Exerciţiul 7 *Să se creeze vizualizarea v_emp_*** care să conţină codul, numele, emailul, data angajării, salariul şi codul jobului salariaţilor din tabelul emp_***. Să se insereze o nouă înregistrare în această vizualizare. Să se verifice că noua înregistrare a fost inserată şi în tabelul de bază.*

```
CREATE VIEW v_emp_***
AS SELECT employee_id, last_name, email, hire_date, salary,job_id
FROM emp_***;

INSERT INTO v_emp_***
VALUES (400,'N1','E1',SYSDATE,5000,'SA_REP');
SELECT employee_id, last_name, email, hire_date, salary, job_id
FROM emp_***;
```

Exerciţiul 8 *Să se mărească cu 1000 salariul angajatului având codul 400 din vizualizarea creată anterior. Ce efect va avea această acţiune asupra tabelului de bază?*

```
UPDATE v_emp_***
SET salary=salary+1000
WHERE      employee_id = 400;

SELECT      employee_id, last_name, salary
FROM      emp_***
WHERE      employee_id = 400;
```

Exerciţiul 9 *Să se şteargă angajatul având codul 400 din vizualizarea creată anterior. Ce efect va avea această acţiune asupra tabelului de bază?*

```
DELETE FROM v_emp_***
WHERE      employee_id = 400;

SELECT      employee_id, last_name, salary
FROM      emp_***
WHERE      employee_id = 400;
```

Exerciţiul 10 *Să se creeze vizualizarea v_emp_dept_*** care să conţină employee_id, last_name, hire_date, job_id, department_id din tabelul emp_*** şi coloana department_name din tabelul dept_***.*

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

Exerciţiul 11 *Să încerce inserarea înregistrării (500, 'N2',SYSDATE,'SA_REP',350, 'Administrativ') în vizualizarea creată anterior.*

```
INSERT INTO v_emp_dept_***
VALUES (500, 'N2',SYSDATE,'SA_REP',350, 'Administrativ');
```

Exerciţiul 12 *Care dintre coloanele vizualizării v_emp_dept_*** sunt actualizabile?*

```
SELECT      column_name, updatable
FROM      user_updatable_columns
WHERE      table_name = 'V_EMP_DEPT_***';
```

Exerciţiul 13 *Să se creeze vizualizarea v_dept_*** care să conţine codul şi numele departamentului, numărul de angajaţi din departamentul respectiv şi suma alocată pentru plata salariilor. Această vizualizare permite actualizări?*

```
CREATE VIEW v_dept_*** (cod, nume, nr_angajati, val_salarii)
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;
```

Exerciţiul 14 *Să se creeze vizualizarea v_manager_*** care să conţină numele, emailul, data angajării, salariul şi codul jobului managerilor din tabelul emp_***. (Managerii sunt angajaţii pentru care job_id-ul conţine şirul 'MAN') În această vizualizare nu se va permite modificarea joburilor. (Să se utilizeze constângerea WITH CHECK OPTION)*

```
CREATE VIEW v_manager_***
AS SELECT employee_id, last_name, email, hire_date, salary,job_id
FROM emp_***
WHERE job_id LIKE '%MAN'
WITH CHECK OPTION;
```

Exerciţiul 15 *Să se listeze structura şi conţinutul vizualizării v_manager_***. Să se încerce modificarea jobului unui angajat.*

Exerciţiul 16 *Să se creeze o vizualizare (v_dept_***) asupra tabelului dept_*** să nu permită efectuarea nici unei operaţii LMD. Testaţi operaţiile de inserare, modificare şi ştergere asupra acestei vizualizări. (Să se utilizeze constângerea WITH READ ONLY)*

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

Exerciţiul 17 *Să se creeze vizualizarea v_emp_info_*** asupra tabelului emp_*** care conţine codul, numele, prenumele, emailul şi numărul de telefon ale angajaţilor companiei. Se va impune unicitatea valorilor coloanei email şi constrângerea de cheie primară pentru coloana corespunzătoare codului.*

*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_***;*

Exerciţiul 18 *Să se adauge o constrângere de cheie primară asupra vizualizării v_manager_***.*

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

Secvenţe

O secvenţă este un obiect al bazei de date ce permite generarea de numere întregi unice cu scopul de a fi folosite ca valori pentru cheia primară sau coloane numerice unice. Secvenţele sunt independente de tabele. Aceeaşi secvenţă poate fi folosită pentru mai multe tabele.

Sintaxa comenzii *CREATE SEQUENCE* este:

$CREATE\ SEQUENCE\ nume\_secvenţă$
$[INCREMENT\ BY\ n]$
$[START\ WITH\ valoare\_start]$
$[\{MAXVALUE\ valoare\_maximă \mid NOMAXVALUE\}\ ]$
$[\ \{MINVALUE\ valoare\_minimă \mid NOMINVALUE\}\ ]$
$[\ \{CYCLE \mid NOCYCLE\}]$
$[\ \{CACHE\ n \mid NOCACHE\}\ ];$

- *INCREMENT BY* specifică diferenţa dintre valorile succesive ale secvenţei (valoare implicită 1).
- *START WITH* specifică primul număr care va fi generat de secvenţă (implicit 1).
- *MAXVALUE, MINVALUE* precizează valoarea maximă, respectiv minimă pe care o poate genera secvenţa. Opţiunile *NOMAXVALUE, NOMINVALUE* sunt implicite. *NOMAXVALUE* specifică valoarea maximă de 1027 pentru o secvenţă crescătoare şi -1 pentru o secvenţă descrescătoare. *NOMINVALUE* specifică valoarea minimă 1 pentru o secvenţă crescătoare şi -1026 pentru o secvenţă descrescătoare.
- *CYCLE* şi *NOCYCLE* specifică dacă secvenţa continuă să genereze numere după obţinerea valorii maxime sau minime. *NOCYCLE* este opţiunea implicită.
- *CACHE n* precizează numărul de valori pe care server-ul Oracle le prealocă şi le păstrează în memorie. În mod implicit, acest număr de valori este 20.

Opţiunea *CACHE* pemite accesul mai rapid la valorile secvenţei care sunt păstrate în memorie. Aceste valori sunt generate la prima referinţă asupra secvenţei. Fiecare valoare din secvenţă se furnizează din secvenţa memorată. După utilizarea ultimei valori prealocate secvenţei, următoarea solicitare a unei valori determină încărcarea unui alt set de numere în memorie. Pentru a nu fi prealocate şi reţinute în memorie astfel de valori, se utilizează opţiunea *NOCACHE*.

Pseudocoloanele *NEXTVAL* şi *CURRVAL* permit utilizarea secvenţelor.

- nume_secv.*NEXTVAL* returnează următoarea valoare a secvenţei, o valoare unică la fiecare referire. Trebuie aplicată cel puţin o dată înainte de a folosi *CURRVAL;*
- nume_secv.*CURRVAL* returnează valoarea curentă a secvenţei.

Pseudocoloanele *NEXTVAL* şi *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* şi *CURRVAL* nu se pot utiliza:

- în lista *SELECT* a unei vizualizări;
- într-o comandă *SELECT* ce conţine *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 secvenţelor se realizează cu ajutorul comenzii *DROP SEQUENCE*.

DROP SEQUENCE nume_secv;

Exemplul 19 *Să se creeze o secvenţă care să aibă valoarea maximă 5000, să înceapă de la 0 şi să se incrementeze cu 5. Utilizând această secvenţă să se creeze un tabel dept_*** *care să conţină numele departamentelor din tabelul departments.*

```
CREATE SEQUENCE sec_dep_***
INCREMENT BY 5
START WITH 0
MINVALUE 0
MAXVALUE 5000

CREATE TABLE dept_***
AS SELECT sec_dep_***.NEXTVAL,department_name
FROM departments;
```

Exemplul 20 *Să se creeze o secvenţă care are pasul de incrementare 10 şi începe de la 10, are ca valoare maximă 10000 şi să nu cicleze.*

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

Exemplul 21 *Să se modifice toate liniile din tabelul emp_ ****, regenerând codul angajaţilor astfel încât să utilizeze secvenţa sec_ ****.*

*UPDATE emp_ ****
*SET employee_id = sec_ ****. *NEXTVAL;*

Indecşi

Un index este un obiect al unei scheme utilizator care e utilizat de server-ul Oracle pentru a mări performanţele unui anumit tip de cereri asupra unui tabel.

Indecşii:

- evită scanarea completă a unui tabel la efectuarea unei cereri;
- reduc operaţiile 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 independenţi de tabelele pe care le indexează, în sensul că dacă sunt şterşi nu afectează conţinutul tabelelor sau comportamentul altor indecşi;
- sunt menţinuţi şi utilizaţi automat de către server-ul Oracle;
- sunt şterşi odată cu eliminarea tabelului asociat.

Indecşii pot fi creaţi :

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

Se creează un index atunci când:

- o coloană conţine un domeniu mare de valori;
- o coloană conţine un număr mare de valori null;
- una sau mai multe coloane sunt folosite des în clauza WHERE sau în condiţii de join în programele de aplicaţii;
- tabelul este mare şi de obicei cererile obţin mai puţin de 2%-4% din liniile tabelului;
- tabelul nu este modificat frecvent.

Sintaxa comenzii *CREATE INDEX*:

CREATE [UNIQUE] INDEX nume_index
ON tabel (*coloana*1 [, *coloana*2 . . .]);

Sinonime

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

Crearea unui sinonim este utilă pentru a evita referirea unui obiect ce aparţine altui utilizator prefixându-l cu numele utilizatorului şi pentru a scurta numele unor obiecte cu numele prea lung.

Comanda pentru crearea sinonimelor este:

CREATE [PUBLIC] SYNONYM nume_sinonim
FOR obiect;