Crearea, modificarea și suprimarea vederilor (VIEW). Utilizarea vederilor.

O *vedere* este un tabel logic bazat pe un alt tabel sau pe o altă vedere şi nu conţine date proprii. Tabelele pe baza cărora sunt create vederile se numesc *tabele de bază*.

La nivelul bazei de date o vedere este stocată sub forma unei instrucţiuni **SELECT** în dicţionarul de date.

Avantajele utilizării vederilor

Vederile prezintă o serie de vederi:

- restricţionează accesul la baza de date deoarece o vedere pune la dispoziţia utilizatorilor doar o parte a datelor din baza de date;

- permit utilizatorilor să apeleze la interogări simple pentru a obţine rezultatele unor interogări complexe. De exemplu, vederile permit utilizatorilor să extragă informaţii din mai multe tabele fără a cunoaşte modul de realizare a joncţiunilor..

- asigură independenţa datelor pentru utilizatori şi aplicaţii. O singură vedere poate fi utilizată pentru a obţine date din mai multe tabele;

- furnizează grupurilor de utilizatori accesul la date în conformitate cu anumite criterii, stabilite de administratorul bazei de date.

Vederile se clasifică în două grupe: ***simple şi complexe***. Diferenţa de bază între cele două grupe este legată de operaţiile LMD – limbajul de manipulare a datelor (inserare, actualizare şi ştergere).

Caracteristici	**Vederi simple**	**Vederi complexe**
Număr de tabele	1	1 sau mai multe
Conţine funcţii	Nu	Da
Conţine grupuri de date	Nu	Da
Operaţii LMD	Da	Nu întotdeauna

O vedere simplă este o vedere care:

- furnizează (oferă, ne dă) date dintr-un singur tabel;

- nu conţine funcţii sau grupuri de date;

- permite execuţia unor operaţii LMD.

O vedere complexă este o vedere care :

- extrage date din mai multe tabele;

- conţine funcţii sau grupuri de date;

- nu permite întotdeauna operaţii LMD.

Crearea şi suprimarea unei vederi

Instrucţiunea ce permite crearea unei vederi este CREATE VIEW, având următoarea sintaxă.

CREATE VIEW <nume_viziunii> [(lista de atribute)]
AS interogare
[WITH [CASCADE | LOCAL] CHECK OPTION]

unde:
- numele viziunii – este numele relaţiei virtuale corespunzătoare viziunii;
- lista de attribute defineşte atributele relaţiei virtuale;
- interogarea permite calcularea tuplurilor necesare pentru popularea relaţiei virtuale;
- ***CASCADE*** - specifică faptul că coloana va fi actualizată atunci când coloana referențiată este actualizată, iar rândurile vor fi șterse atunci când ștergerea rândurilor;
- ***WITH CHECK OPTION*** - specifică faptul că pot fi actualizate doar liniile accesibile vederii şi trebuie să îndeplinească condiţiile interogării.

Exemplu

Pentru a crea o vedere numită **EMPVU10** *care conţine detalii despre Angajații din departamentul 10 se va executa următoarea instrucţiune:*

CREATE VIEW empvu10
AS SELECT empno, ename, job
FROM emp
WHERE deptno=10;

Observaţii:
- SELECT poate conţine toate clauxele unei instrucţiuni SELECT,cu excepţia clauzei ORDER BY.

Viziunile sunt **suprimate** (eliminate) (numai definiţia sa, dar nu şi relaţiile pe care se bazează) folosind instrucţiunea **DROP VIEW.**

DROP VIEW <nume viziune>

Suprimarea viziunii nu are consecinţe asupra tuplurilor

Exemplu

1. *Viziunea următoare reprezintă o restricţie a relaţiei funcţionari pentru funcţionarii departamentului Dept02*

CREATE VIEW functDept02

AS SELECT *
FROM functionari
WHERE DeptId = 'Dept02';

Suprimarea se face astfel

DROP VIEW functDept02

2. *Să cream o vedere **ang20** care va conţine persoanele din departamentul 20.*

CREATE VIEW ang20
AS SELECT angajat_id, nume, functie, salariu
FROM angajati
WHERE id_dept=20

Modificarea unei vederi

Opţiunea OR REPLACE permite crearea unei vederi, chiar dacă există deja o vedere cu acelaşi nume, înlocuindu-se astfel vechea versiune a vederii cu cea nouă. Aceasta înseamnă că o vedere poate fi modificată fără a fi necesară ştergerea şi recrearea ei.

Următorul exemplu modifică vederea EMPVU10(ex de mai sus) utilizând clauza CREATE OR REPLACE VIEW. Suplimentar se adaugă un alias pentru fiecare coloană.

CREATE OR REPLACE VIEW empvu10 (employee_number, employee_name, job_title)
AS SELECT empno, ename, job
FROM emp
WHERE deptno=10;

Notă: Când se furnizează nume alternative (alias-uri) coloanelor în clauza CREATE VIEW, alias-urile trebuie listate în aceeaşi ordine ca şi coloanele din subinterogare.

Crearea unei vederi complexe

CREATE VIEW dept_sum_vu(name, minsal, maxsal, avgsal)
AS SELECT d.dname, MIN(e.sal), MAX(e.sal), AVG(e.sal)
FROM emp e, dept d
WHERE e.deptno=d.deptno
GROUP BY d.dname;

Exemplul anterior creează o vedere complexă care conţine numele, salariul minim, salariul maxim şi salariul mediu pentru fiecare departament. De notat că au fost specificate alias-uri pentru vedere. **Acest lucru este necesar dacă una din coloanele vederii rezultă din evaluarea unei funcţii sau expresii.**

Reguli pentru executarea operaţiilor LMD asupra unei vederi

Pentru a putea executa operaţii LMD asupra datelor din baza de date prin intermediul vederii trebuie avute în vedere o serie de reguli.

1. Nu se poate şterge o linie aparţinând unei vederi dacă definiţia vederii conţine :

- funcţii grup;
- clauză GROUP BY;
- cuvântul cheie DISTINCT.

2. Nu se pot modifica datele dintr-o vedere dacă vederea conţine:

- oricare din elementele de la punctul 1;
- coloane definite prin expresii (de exemplu, SALARY*12); - pseudocoloana ROWNUM.

3. Nu pot fi inserate date într-o vedere dacă:

- vederea conţine oricare din elementele de la punctul 1 şi 2;
- există coloane NOT NULL fără valoare implicită în tabelul de bază şi care nu au fost selectate în vedere.

Utilizarea clauzei WITH CHECK OPTION

Există posibilitatea ca, prin intermediul vederilor, să se efectueze verificări de integritate referenţială asupra datelor. Se pot, de asemenea, forţa constrângeri la nivelul bazei de date. O vedere poate fi utilizată pentru a asigura integritatea datelor, dar într-o manieră destul de limitată.

Clauza **WITH CHECK OPTION** specifică faptul că instrucţiunile **INSERT** şi **UPDATE** executate asupra vederii nu pot crea linii noi pe care vederea nu le poate selecta. Prin urmare, vederile pot forţa verificări asupra datelor ce vor fi inserate sau actualizate. Dacă se încearcă execuţia unei operaţii LMD pe linii care nu au fost selectate de vedere se va afişa un mesaj de eroare, împreună cu numele constrângerii (dacă acesta a fost specificat).

Pentru exemplificare să considerăm vederea

```
CREATE OR REPLACE VIEW empvu20
 AS SELECT *
 FROM emp
 WHERE deptno=20
 WITH CHECK OPTION CONSTRAINT empvu20_ck;
```

Orice încercare de **a modifica** numărul departamentului pentru orice linie din vedere va eşua deoarece nu respectă constrângerea **WITH CHECK OPTION.**

```
UPDATE empvu20
  SET deptno=10
 WHERE empno=7788;
```

Exemplu spre realizare:

Se consideră spre exemplificare tabela **Angajati,** având câmpurile:
- CNP char (13) PRIMARY KEY,
- Nume varchar (30) NOT NULL,
- Salariu numeric NOT NULL,
- Cod_departament varchar (5) NOT NULL.

```
CREATE TABLE angajati(
 CNP char(13)PRIMARY KEY,
 Nume varchar(30) NOT NULL,
 Salariu numeric NOT NULL,
 Cod_departament VARCHAR(5) NOT NULL);
```

Tabela Angajati conține următoarele 3 înregistrări:

CNP	Nume	Salariu	Cod_departament
111	Vali	1500	AIA
222	Ana	2500	AIA
333	Maria	3000	CTI

```
SET AUTOCOMMIT ON;
INSERT INTO angajati VALUES ('111', 'Vali', 1500, 'AIA');
INSERT INTO angajati VALUES ('222', 'Ana', 2500, 'AIA');
INSERT INTO angajati VALUES ('333', 'Maria', 3000, 'CTI');
```

```
SQL> select * from angajati;
CNP         NUME                          SALARIU   COD_D
----------- ------------------------------ --------- ---------------
111         Vali                           1500      AIA
222         Ana                            2500      AIA
333         Maria                          3000      CTI
```

Se crează un **VIEW** care permite accesul (pe toate coloanele) doar la liniile din tabela **Angajat**i având câmpul *Cod_departament = 'AIA':*

```
CREATE VIEW vederel AS
(SELECT * FROM angajati
           WHERE Cod_departament='AIA');
```

şi se interoghează acest **VIEW** (practic se realizează o interogare a tabelei Angajati utilizând filtrarea oferită de *view-ul vedere1*):

```
SQL> select * from vedere1;
CNP        NUME                          SALARIU   COD_D
---------- ------------------------------ --------- ----------
111        Vali                             1500      AIA
222        Ana                              2500      AIA
```

Se execută o operaţie de adăugare a unei noi linii în tabela **Angajati** prin intermediul viewului vedere1:

```
INSERT INTO vedere1 VALUES ('444', 'Anca', 2500, 'CTI');
```

```
SQL> select * from angajati;
CNP        NUME                          SALARIU COD_D
---------- ------------------------------ ---------- -----
111        Vali                             1500      AIA
222        Ana                              2500      AIA
333        Maria                            3000      CTI
444        Anca                             2500      CTI

SQL> select * from vedere1;
CNP        NUME                          SALARIU COD_D
---------- ------------------------------ ---------- -----
111        Vali                             1500      AIA
222        Ana                              2500      AIA
```

Se poate observa că angajatul cu **CNP-ul '444'** a fost adăugat în tabela **Angajati**, însa el nu poate fi accesat prin intermediul view-ului, deoarece nu respectă condiția **Cod_departament='AIA'** din comanda de definire a view-ului.

O comandă de ştegere de tipul:

```
SQL> DELETE FROM vedere1 WHERE Cod_departament='CTI';
0 rows deleted.
```

nu va semnala eroare, dar nici nu va şterge linia referită, existentă în tabela dept, deoarece pentru View această linie nu este vizibilă (deci nici accesibilă). Identic şi în cazul unei modificări de date:

```
SQL> UPDATE vedere1 SET Salariu=5000 WHERE Cod_departament='CTI';
0 rows updated.
```

Astfel, concluzia care se impune este ca: **CEEA CE VIEW-UL NU VEDE, NU POATE MODIFICA SAU ŞTERGE.**

Observaţie: Ştergerea unei tabele referite de un View nu conduce automat şi la ştergerea view-ului asociat. Ştergerea doar a view-ului se poate face cu o comandă de tipul **DROP VIEW** *nume_view*.

In continuare, se crează un **VIEW** care permite accesul (pe coloanele *Nume, Salariu, Cod_departament*) doar la liniile din tabela Angajati având campul *Cod_departament='AIA':*

```
CREATE VIEW vedere2 AS
(SELECT Nume,Salariu,Cod departament FROM angajati
            WHERE Cod departament='AIA');
```

Incercarea de a insera o linie nouă în tabela angajati, prin intermediul view-ului vedere2, va esua, având în vedere ca coloana CNP din tabela Angajati este cheie primară, deci şi de tip **NOT NULL.**

```
INSERT INTO vedere2 VALUES ('Raul', 1400,'AIA');
```

va eşua cu mesajul de eroare:

```
ERROR at line 1:
ORA-01400: cannot insert NULL into ("SYSTEM"."ANGAJATI"."CNP");
```

Acest lucru era de altfel previzibil, datorită constrângerii **PRIMARY KEY** impuse coloanei neaccesibile prin View.

De asemenea, o comandă:

```
DELETE FROM vedere2 WHERE CNP='111';
```

va eşua (ca de altfel orice comandă în care view-ul ar face referire la coloana CNP, care pentru el nu exista). Aceasta însă nu înseamnă că operaţia de ştergere nu este posibilă, comanda următoare executându-se fără probleme:

```
DELETE FROM vedere2 WHERE Nume='Vali';
```

Alt exemplu:

Un View poate fi definit (asociat) şi pe mai multe tabele. Fie tabela **Angajati**, definita mai sus şi tabela **Departamente**, având următoarele câmpuri:

- Cod_departament varchar (5) PRIMARY KEY,
- Nume_departament varchar(40) NOT NULL.

```
CREATE TABLE departamente(
 Cod departament VARCHAR(5)PRIMARY KEY,
 Nume Departament varchar(40) NOT NULL
);
```

Tabela **Departamente** contine urmatoarele 2 inregistrări:

Cod_departament	Nume_departament
AIA	Automatica si Informatica Aplicata
CTI	Calculatoare si tehnologia Informatiei

```
INSERT INTO departamente VALUES ('AIA', 'Automatica si Informatica Aplicata');
INSERT INTO departamente VALUES ('CTI', 'Calculatoare si Tehnologia
Informatiei');
```

Între cele două tabele, pentru exemplificarea care va urma, nu este obligatoriu să existe o constrângere de tip *cheie străină*. Un View asociat unor date din cele două tabele poate fi creat astfel:

```
CREATE VIEW vedere3 AS
      (SELECT A.Cod departament, Nume departament, CNP, Nume, Salariu
      FROM departamente A, angajati B
      WHERE A.Cod_departament=B.Cod_departament);
```

In continuare, se prezintă câteva exemple de interogări, folosind acest view:

```
SQL> select * from vedere3;

COD_D  NUME_DEPARTAMENT                  CNP        NUME                 SALARIU
-----  ----------------------------------- ---------- -------------------- ----------
AIA    Automatica si Informatica Aplicata  222        Ana                  2500
CTI    Calculatoare si Tehnologia Informatiei 333     Maria                3000
CTI    Calculatoare si Tehnologia Informatiei 444     Anca                 2500

SQL> select Nume_departament, MIN(Salariu) from vedere3 GROUP BY Nume_departament;

NUME_DEPARTAMENT                    MIN(SALARIU)
----------------------------------- ------------
Automatica si Informatica Aplicata          2500
Calculatoare si Tehnologia Informatiei      2500
```

Un alt exemplu de View asociat unor date din cele două tabele este prezentat mai jos:

```
CREATE OR REPLACE VIEW vedere4
  (Cod_v, Dep_v, Media_v, Maxim_v, Minim_v, Suma_v, Nr_v)
  AS SELECT D.Cod departament, Nume departament, AVG(Salariu), MAX(Salariu),
  MIN(Salariu), SUM(Salariu), COUNT(*)
  FROM angajati A, departamente D
  WHERE A.Cod departament=D.Cod departament
  GROUP BY D.Cod_departament, D.Nume_departament;
```

Observaţii: Astfel de view-uri, definite printr-o subinterogare pe mai multe tabele nu permit operaţii de modificare a informaţiei din tabele (adăugare, ştergere, modificare), ci doar simple interogări asupra tabelelor referite.