Laborator 5
Vederi in SQL.

Ce este o vedere?

O vedere este o tabela virtuala, organizata, la fel ca si tabelele fizice, în rânduri şi coloane. Ea preia rezultatul unei interogări şi îl tratează ca pe o tabela, de aceea o vedere mai poarta şi denumirea de tabela logica.

Spre deosebire de tabelele fizice, vederea nu stochează date şi nici nu are alocat vreun spaţiu de stocare; vederea doar extrage sau derivă datele din tabelele la care aceasta se referă. Aceste tabele poartă numele de tabele de bază ale vederii. Oracle stochează definiţia vederii în dicţionarul de date sub forma textului interogării care defineşte vederea, de aceea o vedere poate fi gândită ca o "interogare stocată".

Denumirea de tabelă virtuală derivă din faptul că orice operaţie asupra datelor (vizibile) prin intermediul unui VIEW se face folosind comenzi consacrate (SELECT, INSERT, DELETE, UPDATE) specifice manipulării datelor dintr-o tabelă fizică, utilizându-se în cadrul sintaxei comenzii numele VIEW-ului în loc de numele tabelei fizice.

Vederea este un instrument foarte puternic pentru dezvoltatorul. de aplicaţii. Ea poate fi bazată pe mai multe tabele sau vederi care se pot găsi pe maşini diferite sau pot aparţine unor utilizatori diferiţi, acestea fiind prezentate ca şi cum ar fi un singur tabel logic.

Sintaxa generală a comenzii pentru crearea unui VIEW este:

```
CREATE VIEW nume_vedere AS subquery;
```

, unde prin *subquery* se înţelege o subinterogare de tip SELECT (practic o comandă SELECT executată pe una sau mai multe tabele.

Se considera spre exemplificare tabela *angajati*, avand campurile:

- 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* contine urmatoarele 3 inregistrari:

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 *angajati* având campul *Cod_departament*=’AIA’:

```
CREATE VIEW vedere1 AS
(SELECT * FROM angajati
            WHERE Cod_departament=’AIA’);
```

şi se interoghează acest VIEW (practic se realizeaza 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 *view*-ului *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 ca anagajatul cu *CNP-ul* ’444’ a fost adaugat in tabela angajati, insa el nu poate fi accesat prin intermediul view-ului, deoarce nu respecta conditia *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* aceasta linie nu este vizibilă (deci nici accesibilă).

Identic şi în cazul unei modificari 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 noua in 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';
```

Un *view* poate fi definit (asociat) şi pe mai multe tabele. Fie tabela *angajati*, definita mai

sus si tabela *departamente*, avand urmatoarele campuri:

- 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 inregistrari:

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 prezinta cateva 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 doua 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;
```

In continuare, se prezinta cateva exemple de interogări, folosind acest *view*:

```
SQL> SELECT * FROM vedere4;

COD_V DEP_V                                MEDIA_V   MAXIM_V   MINIM_V   SUMA_V      NR_V
----- ---------------------------------- ---------- ---------- ---------- ---------- ----------
CTI   Calculatoare si Tehnologia Informatiei  2750       3000       2500       5500          2
AIA   Automatica si Informatica Aplicata      2500       2500       2500       2500          1

SQL> SELECT * FROM vedere4 WHERE Media_v>2500;

COD_V DEP_V                                MEDIA_V   MAXIM_V   MINIM_V   SUMA_V      NR_V
----- ---------------------------------- ---------- ---------- ---------- ----------
CTI   Calculatoare si Tehnologia Informatiei  2750       3000       2500       5500          2
```

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.

Problema propusa:

Se considera baza de date XE, avand urmatoarea structura:

- Tabela studenti:

- Marca char(6) PRIMARY KEY,
- Nume varchar (30) NOT NULL,
- Stare_civila char(1) NOT NULL,
- Sex char (1) NOT NULL,
- An integer NOT NULL CHECK (An>0 and An<5),
- Grupa integer NOT NULL.

- Tabela discipline

- Cod_disciplina integer PRIMARY KEY,
- Denumire_disciplina varchar(30) NOT NULL.

- Tabela note

- Marca char(6) NOT NULL REFERENCES studenti (Marca) ON DELETE CASCADE,
- Nota integer NOT NULL CHECK (Nota >0 AND Nota <=10),
- Cod_disciplina integer NOT NULL REFERENCES discipline(Cod_disciplina) ON DELETE CASCADE.

```
CREATE TABLE studenti (
Marca char(6) PRIMARY KEY,
Nume varchar(30) NOT NULL,
Stare_civila char(1) NOT NULL,
Sex char(1) NOT NULL,
An integer NOT NULL CHECK (An>0 and An<5),
Grupa integer NOT NULL);

CREATE TABLE discipline (
Cod_disciplina integer PRIMARY KEY,
Denumire_disciplina varchar(30) NOT NULL);

CREATE TABLE note(
Marca char(6) NOT NULL REFERENCES studenti (Marca) ON DELETE CASCADE,
Nota integer NOT NULL CHECK (Nota >0 AND Nota <=10),
Cod_disciplina integer NOT NULL REFERENCES discipline(Cod_disciplina)
ON DELETE CASCADE);
```

Se considera baza de date populata cu urmatoarele inregistrari:

Tabela studenti:

Marca	Nume	Stare_civila	Sex	An	Grupa
111111	Rosu Alina	N	F	1	1
222222	Vlad Simona	N	F	1	2
333333	Lung Ionut	C	M	2	1

Tabela discipline:

Cod_disciplina	Denumire_disciplina
1	Programare WEB
2	Baze de date
3	Retele neuronale
4	Java

Tabela note:

Marca	Nota	Cod_disciplina
111111	8	1
111111	10	2
222222	4	2
333333	7	3
333333	8	4

```
SET AUTOCOMMIT ON;
INSERT INTO studenti VALUES('111111', 'Rosu Alina', 'N', 'F', 1,1);
INSERT INTO studenti VALUES('222222', 'Vlad Simona','N', 'F', 1,2);
INSERT INTO studenti VALUES('333333', 'Lung Ionut', 'C', 'M', 2,1);

INSERT INTO discipline VALUES(1, 'Programare WEB');
INSERT INTO discipline VALUES(2, 'Baze de date');
INSERT INTO discipline VALUES(3, 'Retele neuronale');
INSERT INTO discipline VALUES(4, 'Java');

INSERT INTO note VALUES('111111',8, 1);
INSERT INTO note VALUES('111111',10, 2);
INSERT INTO note VALUES('222222',4, 2);
INSERT INTO note VALUES('333333',7, 3);
INSERT INTO note VALUES('333333',8, 4);
```

a). Sa se afiseze toti studentii din tabela studenti care au starea civila necasatorit (’N’);

b). Sa se creeze o vedere numita Date_studenti_M care sa contina o evidenta a studentilor de sex masculin din tabela Studenti si sa se afiseze dintre acestia, studentii care au starea civila necasatorit (‚N’), iar apoi casatorit (’C’).

c). Sa se creeze o vedere numita *Medii* care sa acceseze urmatoarele date din tabelele *Studenti, Note*: marca, numele studentului, anul, grupa, media notelor si numarul de note obtinute de fiecare student.

-Sa se afiseze continutul vederii numite *Medii*.

-Sa se afiseze studentii care au participat la minim 2 examene (au minim 2 note). Se cere utilizarea vederii *Medii*:

d). Sa se creeze o vedere numita *Note_studenti* care sa acceseze urmatoarele date din tabelele *Studenti, Discipline, Note*: marca, numele studentului, anul, grupa, denumirea disciplinei si nota obtinuta de fiecare student la respectivele discipline.

- Sa se afiseze continutul vederii numite *Note_studenti*, ordonand inregistrarile dupa Marca:

-Sa se afiseze notele studentilor de la disciplina care incepe cu expresia ’Baze de’ (utilizand vederea *Note_studenti*).

-Sa se afiseze toti studentii bursieri, care au media notelor>7.5 (utilizand vederea *Note_studenti*). Se doreste afisarea urmatoarelor date: marca, numele, anul, grupa, precum si media notelor studentilor bursieri.

-Sa se afiseze media studentilor pe fiecare an de studiu (utilizand vederea *Note_studenti*):

e). Se cere crearea unei vederi numita *Medii_restantieri,* care sa permita accesul la urmatoarele date din tabelele *Studenti* si *Note*: marca, nume, an, grupa, precum si media notelor studentilor restantieri (care au media notelor <5).

- Sa se afiseze toti studentii restantieri.

f). Sa se creeze o vedere numita *NoteBD*, care sa acceseze marca si notele studentilor de la disciplina ’Baze de date’.

-Sa se scada 1 punct din toate notele de la disciplina ‘Baze de date’ (utilizand vederea *NoteBD*).

-Sa se stearga notele studentului cu *Marca*=’111111’, de la disciplina ‘Baze de date’ (utilizand vederea *NoteBD*).

-Sa se incerce inserarea urmatoarei inregistrari, (utilizand vederea *NoteBD*) si sa se comenteze reusita sau esecul inserarii.

```
INSERT INTO NoteBD VALUES('111111',8);
```

g). Sa se creeze o vedere numita *NoteAll*, care sa acceseze marca, notele studentilor si codul disciplinei la care s-au obtinut note >4.

-Sa se incerce inserarea urmatoarelor inregistrari, (utilizand vederea *NoteAll*) si sa se comenteze reusita sau esecul inserarii.

```
INSERT INTO NoteAll VALUES('111111',8,2);
```

```
INSERT INTO NoteAll VALUES('123573',8,2);
```