Lucrarea 6.

Utilizare vederi, Outer Join si variabile SQL+

Definire vederi View

Vederile definesc o macheta prin care se vad numai anumite coloane din tabelele bazei de date.
O vedere se defineşte printr-un SELECT care poate conține unul sau mai multe operații de JOIN.
Vederile sunt tratate ca nişte tabele virtuale pentru care nu se alocă spațiu.
SELECT-ul prin care se defineşte o vedere se ține intr-un rând din tabela **All_Views** ca o coloana **Tex**t de tip LONG (echivalent cu tipul memo din dBase).
Vederile - simplifică scrierea interogărilor complexe;
- asigura posibilitatea de a acorda drepturi de acces pe vedere si nu pe tabele;
- permite definirea unor tabele virtuale pentru aplicații.

CREATE VIEW nume_view (lista_coloane) AS SELECT lista_col
FROM tab1,tab2 WHERE cond [WITH READ ONLY];

CREATE VIEW Vemp (Nume,Salar,Departament) AS
SELECT E.Ename Nume,E.Sal Salar , D.Dname Nume_dept
FROM EMP E, DEPT D WHERE E.Deptno=D.Deptno;

SELECT * FROM Vemp WHERE salar>1200; -- Afişează tabela virtuala executând JOIN.

Descr All_Views -- afişează structura tabelei (vederii) cu descrierea vederilor.
Afişarea vederilor existente in BD se face din tabela All_Views unde definirea este un Select memorat in coloana Text de tip LONG.
SET LONG 2000 -- setează lungime afişată pentru coloane de tipul LONG
Afişarea vederii Vemp definită anterior se poate face specificând numele în interogarea:
SELECT Owner, View_Name, Text FROM All_Views WHERE View_Name= 'VEMP';

OWNER	VIEW_NAME	TEXT
SCOTT	VEMP	SELECT E.Ename Nume,E.Sal Salar , D.Dname Nume_dept FROM EMP E, DEPT D WHERE E.Deptno=D.Deptno

Vederile simple permit modificările si inserările (DML):
- Definite pe o tabela
- Fără funcții agregat sau clauza GROUP BY

Vederile complexe conțin operații de Join pe 2 sau mai multe tabele sau operații de grup. Asupra acestor vederi nu se permit decât operații de interogare.

Drepturile de acces pentru grupuri de utilizatori se recomandă să se dea pe vederi si nu pe tabele. Adăugările sau modificări se pot face pe vedere şi fizic se fac în tabela corespunzătoare. Se pot realiza astfel tabele virtuale. Pot exista utilizatori, care au acces la toată tabela şi uni numai la anumite coloane, sau la anumite înregistrări.

La proiectarea BD se va evita fragmentarea tabelelor din motive de 'proprietate' sau locație, deoarece prelucrările globale se complică. Va exista o singură tabelă pentru fiecare entitate a BD. Intr-o întreprindere va fi o singură tabelă de stocuri, chiar dacă sunt 10 magazii şi gestionari. Se va defini câte o vedere Mag_i pentru fiecare magazie şi pentru vederea Mag_i va primi acces nelimitat numai gestionarul $Gest_i$, care va răspunde pentru materialele din magazia lui. Serviciul de contabilitate va avea acces la toată tabela printr-o vedere **numai la citire,** pentru control şi prelucrări financiar contabile globale.

Pentru gestiunea universitară va exista o singură o singură tabelă de studenți şi mai multe vederi pe facultăți sau chiar secții. Fiecare secretară va primi drepturi complete pentru secția de care răspunde. Conducerea facultății va avea drepturi de citire în toate secțiile facultății pentru informare, control şi prelucrări statistice. Conducerea universității va avea drepturi de citire pentru întreaga tabelă de studenți. Anumite câmpuri din tabela studenți pot fi consultate de oricine fiind publice (Nume, adresa, tel, e-mail, data_nasterii,..).

Create View Stud_AC AS Select * From stud Where cods='AC';
GRAT SELECT, INSERT, UPDATE, DELETE ON Stud_AC TO secr_ac;
GRANT SELECT(Nume, Adresa, Tel) ON Stud TO Public;
Secretara de la AC are drepturi complete asupra studenților de la AC.
Câmpurile Nume, Adresa, Tel din fişierul studenți pot fi citite de oricine dar nu modificate.
Select * From stud_ac Where cods Like 'AC4%'; -- Afişează studenții din anul 4 AC

Outer JOIN

Iner Join este Join-ul clasic şi preia din cele două tabele numai înregistrările care au corespondent. Interogarea următoare nu va afişa salariații care nu au specificat DeptNo (persoane din conducere,întretinere):

```
SELECT E.Ename Nume,E.Sal Salar , D.Dname Nume_dept
        FROM EMP E, DEPT D WHERE E.Deptno=D.Deptno;
```

Left Outer Join - ia toate înregistrările din prima tabele si

- Se completează coloanele din a doua tabela daca au corespondent conf. condiției de Join
- Coloanele din a doua tabela rămân goale daca nu există înregistrarea corespondenta

```
SELECT E.Ename Nume,E.Sal Salar , D.Dname Nume_dept
        FROM EMP E, DEPT D WHERE E.Deptno=D.Deptno(+);
```

Right Outer Join – lucrează asemănător luând toate înregistrările din tabela a doua.

In exemplul următor se afişează toți salariații, dar care nu au telefon o sa aibă coloana TEL si Impuls (număr de impulsuri) necomplectate.

```
Select P.Nume, P.Adresa, P.Tel, T.Impuls FROM Pers P, Telefon T
        Where P.Tel=T.Tel(+);
```

Semnul (+) se pune după coloana din condiția de Join corespunzătoare tabelei care este incompleta.

Self JOIN

Este JOIN-ul realizat in cadrul aceleaşi tabele. Un câmp din tabela refera alt câmp din aceiaşi tabela. Se realizează deschizând aceeaşi tabela de 2 ori cu alias-uri diferite.
Se vor afişa date salariaţi, numele si salarul şefului folosind tabela EMP unde fiecare salariat are codul şefului (MGR), care este tot un salariat cu codul Empno.

```
SELECT E.Ename Nume, E.Sal Salar, S.Ename Nume_sef, S.Sal Salar_sef
FROM EMP E, EMP S WHERE E.Mgr=S.Empno;
```

NUME	SALAR	NUME_SEF	SALAR_SEF
FORD	3000	JONES	2975
SCOTT	3000	JONES	2975
JAMES	950	BLAKE	2850
TURNER	1500	BLAKE	2850
MARTIN	1250	BLAKE	2850
WARD	1250	BLAKE	2850
ALLEN	1600	BLAKE	2850
MILLER	1300	CLARK	2450
ADAMS	1100	SCOTT	3000
CLARK	2450	KING	5000
BLAKE	2850	KING	5000
JONES	2975	KING	5000
SMITH	800	FORD	3000

-- Se observa că în lista de salariaţi nu exista salariatul KING fiindcă el este preşedinte şi nu are şef

Dacă se utilizează Left Outer Join vor apare toţi salariaţii:

```
SELECT E.Ename Nume, E.Sal Salar, S.Ename Nume_sef, S.Sal Salar_sef
FROM EMP E, EMP S WHERE E.Mgr=S.Empno(+)
```

NUME	SALAR	NUME_SEF	SALAR_SEF	
FORD	3000	JONES	2975	
SCOTT	3000	JONES	2975	
JAMES	950	BLAKE	2850	
TURNER	1500	BLAKE	2850	
MARTIN	1250	BLAKE	2850	
WARD	1250	BLAKE	2850	
ALLEN	1600	BLAKE	2850	
MILLER	1300	CLARK	2450	
ADAMS	1100	SCOTT	3000	
CLARK	2450	KING	5000	
BLAKE	2850	KING	5000	
JONES	2975	KING	5000	
SMITH	800	FORD	3000	
KING	5000			-- datele despre şef lipsesc

Select-uri imbricate

Consideram o BD universitara având tabelele:
STUD(Cods, Nume, Adresa, Datan,Bursa) - Date personale studenți
NOTE(Cods,Codc,Nota) - tabela note cu cheia Cods,Codc
CU**RS(Codc,Titlu,Codp)** - tabela cursuri cu referința la tabela PROF prin Codp
PR**OF(Codp,Nume,Grad,Salar)** - tabela de profesori
Afişare studenți care urmează cursurile profesorului 'POP':

```
SELECT Nume,Cods,Datan FROM Stud
WHERE Cods=ANY (SELECT Cods FROM Note
WHERE Codc=ANY (SELECT Codc FROM Curs
WHERE Codp=ANY (SELECT Codp WHERE  Nume='POP')));
```

Operatorul ' **=ANY'** poate fi înlocuit cu **IN** si considera ca se iau toate valorile selectate.
Operatorul **ALL** se foloseşte când toate elementele selectate trebuie sa îndeplinească condiția.
Afişare salariați care au salarul mai mare decât toate salariile medii pe departamente:

```
SELECT * FROM Emp WHERE
sal > ALL (SELECT AVG(Sal) FROM EMP GROUP BY Deptno);
```

Selectul imbricat poate avea coloane multiple:

```
SELECT lista_coloane FROM tabela
WHERE (col1,col2) IN (SELECT col1,col2 FROM tabela WHERE conditie);
```

Un Select poate apare si in clauza FROM:
Afisare persoane cu salarul > salarul mediu pe departamentul din care fac parte:

```
SELECT E.Ename Nume ,E.Sal Salr ,E.Deptno B.Salavg Salar_mediu
FROM Emp E , (SELECT Deptno,Avg(sal) Salavg FROM Emp GROUP BY Deptno) B
WHERE E.deptno=B.deptno AND E.sal>B.salavg;
```

Variabile SQL+

In SQL nu exista variabile.
In SQL+ se accepta definirea de variabile mai ales pentru dialog cu utilizatorul.
DEFINE var= valoare - creaza o variabila CHAR ce poate fi initializata
DEFINE var - afiseaza valoarea variabilei
DEFINE - afiseaza toate variabilele
UNDEFINE var - strge variabila
Variabilele se pot introduce si prin dialog in comanda ACCEPT:

```
ACCEPT vdept PROMPT ' Dati numele departamentuli: '
```

Sintaxa pentru commanda ACCEPT este:

```
                NUMBER                  '99.999'
ACCEPT var      CHAR        FORMAT      'An'    PROMPT ' Mesaj' [Hide]
                DATE
```

In comenzile SQL pot apare variabile temporare care daca nu sun definite cer valoarea de la consola. Acest mod simplifica comenzile care pot fi parametrizate.

INSERT INTO Stud (Nume,Adresa,datan,Bursa)
VALUES (&nume,&adresa,&datan,&bursa);
'&nume','&adresa',

Dialogul va fi de forma:
Introduceti nume: 'POPESCU'
Introduceti Datan: ,'15-ian-1981'
Introduceti bursa: 800

Daca s-a dat '&nume' se poate introduce numele ca text fara apostroafe:
SET VERIFY OFF - nu se mai afiseaza vloare veche si noua a variabilei

Variabilele se pot introduce in lista de campuri sau conditii si vor fi cerute prin dialog:
SELECT &camp1,&camp2 FROM stud WHERE &cond;

Utilizare rapoarte

TTITLE 'text' ON|OFF - titlul de sus
BTITLE 'text' ON|OFF - titlul din partea de jos a paginii
SET PAGESIZE 40 - Numar de linii pe pagina
SET LINESIZE 120 - Numar de caractere pe pagina
SET LONG 200 - Numar de cactere afisate la tipul LONG
SE**T HEADING ON|OFF** - afisare sau nu titlul raportului
SET COLSEP ' | ' - separatorul de coloane (implicit spatiu)
SHOW ALL - afisare toate setarile variabilelor sistem

Definirea coloanelor dintr-un raport:

COLUMN lista_optiuni							Right
COLUMN	**Nume HEADING**	**'Nume| Prenume**	**' FORMAT**	**A20**	**JUST**	**Center**
COLUMN	Bursa **HEADING**	' Bursa '		L9.999		Left
COLUMN	Datan **HEADING**	'Data nasterii'				

Afisarea raportului cu Select:
SELECT Nume,Adresa,Datan,Bursa FROM Stud;

Setarile se fac inainte de Select si trebuie ca secventa sa fie memorarta intr-un fisier.SQL
EDIT prog1 - editeaza un program in directorul curent (script SQL)

Lansarea unui script SQL se face cu:
START prog1
GET prog1 - citeste programul specificat in bufferul curent