LIMBAJUL DE MANIPULARE A DATELOR

1. Obiectivul lucrării:

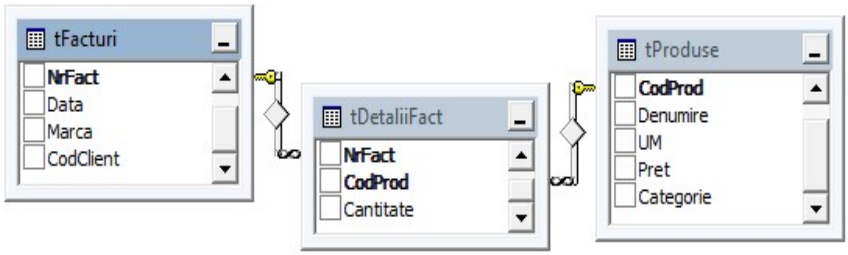
   Formarea și dezvoltarea abilitaților de interogare a bazelor de date.

2. Breviar teoretic cu exerciții și probleme rezolvate

Limbajul de manipulare a datelor (LMD) este limbajul de modificare a informațiilor conținute în baza de date. Există patru comenzi SQL care permit modificarea datelor și anume:

INSERT- pentru adăugarea de rânduri
UPDATE –pentru actualizarea(modificarea) rândurilor
DELETE –pentru ștergerea rândurilor
TRUNCATE-pentru ștergerea întregului conținut al unui tabel.

Pentru exemplificarea lucrului cu aceste comenzi, considerăm următorul model relațional:



Comanda INSERT

Comanda INSERT a mai fost prezentată într-un paragraf anterior.

Pentru a insera noi linii într-un tabel sau într-o vedere se utilizează comanda INSERT. O linie nouă inserată va ocupa în tabel o locaţie arbitrară (independenţa logică a datelor în bazele de date relaţionale). Comanda INSERT are una din următoarele două forme:

```
INSERT [INTO] denumire_tabel [(denumire_coloană[,…])]
VALUES (expresie [,… ])[,… ]
```

```
INSERT [INTO] denumire_tabel[(denumire_coloană[,…])]
  Comanda_Select
```

Prima formă permite inserarea directă a unui rând sau a mai multor rânduri într-un tabel al bazei de date, iar a doua formă permite inserarea mai multor rânduri care sunt rezultatul unei comenzi SELECT.
expresie poate fi o expresie constantă***, null*** (dacă este acceptată) sau ***default*** (dacă este definită o valoare implicită).

Comnda SELECT INTO

Comanda SELECT INTO creează un nou tabel și introduce în el rândurile furnizate de interogare:

```
SELECT lista_de_coloane
INTO tabel_nou [IN baza_de_date_externa]
FROM sursa_datelor
[WHERE conditie]
```

Exemple:

Sa se insereze date in tabelul **tProduse**

```
insert into tProduse(CodProd,Denumire,UM, Pret, Categorie)
 values ('P1','Mere','kg',4,'fructe'),
         ('P2','Pere','kg',7,'fructe'),
('P3','Banane','kg',5,'fructe'),
('P4','Rosii','kg',12,'legume'),
('P5','Castraveti','kg',6,'legume'),
('P6','Cartofi','kg',3,'legume'),
('P7','Faina','Kg',2,'cereale'),
('P8','Malai','Kg',2,'cereale'),
('P9','Taitei','Kg',4,'cereale')
```

Sa se creeze tabelul **tLucru** cu produsele din categoria *fructe* inserate in **tProduse**

```
select CodProd,Denumire,UM, Pret, Categorie
into tLucru
from tProduse
where Categorie='fructe'
```

Sa se adauge in tabelul **tLucru** produsele din tabelul **tProduse** ce apartin categoriei *legume*

```
insert into tLucru
select CodProd,Denumire,UM, Pret, Categorie
from tProduse
where Categorie='legume'
```

Comanda UPDATE

Comanda `UPDATE` este folosită pentru a modifica valorile datelor existente în tabele.
Sintaxa uzuală a comenzii UPDATE prezintă următoarele variante:
1)

```
UPDATE tabel
SET coloana1 = expresie1,
    coloana2 = expresie2, ...
```

```
[WHERE condiţie]
```
2)
```
UPDATE tabel
SET coloana1 = expresie1,
    coloana2 = expresie2, ...
FROM tabele_cuplate
[WHERE condiţie]
```

Clauza *FROM* este o extensie specifică SQL Server, inexistentă în standardul SQL92. Prin această extensie se poate specifica o cuplare de mai multe tabele care se poate folosi în locul unei fraze *Select* imbricată în clauza *WHERE*. Sunt acceptate toate tipurile de asocieri (*Inner, Left, Right, Full*) sunt acceptate şi rezultate obţinute din fraze *Select*.

Comanda *UPDATE* modifică valorile înregistrărilor în funcţie de condiţia clauzei *WHERE*. În lipsa clauzei WHERE, vor fi actualizate toate înregistrările din tabelul dat.

Expresia furnizată ca valoare nouă a unei coloane poate utiliza valorile curente ale câmpurilor din înregistrarea care este actualizată.
Exemple:

```
1)Să se mărească prețul tuturor produselor cu 10%

update tProduse  set pret=pret*1.1

2)Să se micșoreze prețul produselor din categoria 'Lactate' cu 5%

update tProduse  set pret=pret*0.95 where categoria = 'Lactate'
```

Să se exemplifice utilizarea funcției CASE într-o comandă UPDATE

```
update tProduse
set pret=CASE categorie
          when  'lactate'   then  pret*1.1
          when  'fructe'   then  pret*1.08
          when  'panificatie'  then  pret*1.05
          else    pret*1.03
         END
```

3) Presupunând că avem un tabel tModiPret cu structura:

```
create table tModiPret
   ( codProd char(10) primary key,
     procent float
   )
```

ce conţine procente diferențiate de modificare a prețurilor anumitor produse,
Pentru a modifica datele din tabelul **tProduse** pe corespunzător datelor din tabelul **tModiPret** se poate folosi următoarea comandă care conţine tabele corelate

```
update tProduse
set pret=pret * (1+procent/100)
from tProduse A inner join tModiPret B on  A.codProd=B.codProd
```

sau, fără clauza *from*,

```
UPDATE tProduse
```

```
SET pret=pret*(1+ (SELECT procent
                   FROM tModiPret A
                   WHERE A.codProd=tProduse.codProd)/100)
WHERE codProd IN (SELECT codProd FROM tModiPret)
```

Obs.
1) Nu se permite asocierea unui alias tabelului din clauza *update*
2) Dacă nu includem condiţia

```
    WHERE codProd IN (SELECT codProd FROM tModiPret)
```

atunci

```
(SELECT procent
FROM tModiPret A
WHERE A.codP=tProduse.codProd)/100)
```

va returna *null* pentru valorile din coloana *codProd* ce nu se găsesc în *tModiPret* şi prin urmare pentru aceste coduri, *pret* va fi setat cu *null*. Acest lucru poate fi evitat prin utilizarea functiei *isnull* astfel:

```
UPDATE tProduse
SET pret=pret*
       (1+ isnull((SELECT procent
                   FROM tModiPret A
                   WHERE A.codProd=tProduse.codProd)/100,0))
```

Dacă tabelul de actualizat este implicat în diverse legături cu alte tabele ca partea unu a unei legături unu la mulți şi opţiunea *update cascading* este activată, atunci orice modificare a coloanei (coloanelor) de legatură va fi propagată şi în tabelele secundare corelate.

```
ALTER TABLE tDetaliiFact
drop constraint FK_tDetaliiFact_CodProd

ALTER TABLE tDetaliiFact
add Constraint FK_tDetaliiFact_CodProd
    FOREIGN KEY(CodProd)
    REFERENCES tProduse (CodProd) on update cascade

update tProduse set codProd ='1101' where codProd='1001'
```

va genera și modificarea tabelului *tDetaliiFact*: valoarea '1001' din coloana *CodProd* va fi inlocuită cu valoarea '1101' atât în tabelul *tProduse* cât și în *tDetaliiFact*.

<u>Comanda DELETE</u>

Comanda `DELETE` realizează ştergerea din tabelul asociat a înregistrărilor care îndeplinesc anumite condiţii.
Comanda `DELETE` şterge numai înregistrări din tabel nu şi tabelul. Pentru a şterge un tabel se foloseşte comanda `DROP TABLE`.

Sintaxa uzuală a comenzii DELETE prezintă următoarele variante:

1)

```
DELETE [FROM] tabel
[WHERE condiţie];
```

2)

```
DELETE [FROM]  tabel
FROM tabele_cuplate
```

```
[WHERE condiţie];
```

Similar comenzii UPDATE, comanda DELETE şterge anumite înregistrări în funcţie de condiţia din clauza WHERE. În lipsa clauzei WHERE vor fi şterse toate înregistrările din tabelul dat. În această clauză pot fi incluse şi subinterogări.

1)Să se șteargă toate rândurile tabelului tDetaliiFact

```
DELETE tDetaliiFact
```

2)Să se șteargă toate rândurile tabelului tDetaliiFact pentru care NrFact =’f1’

```
DELETE tDetaliiFact where NrFact=’f1’
```

3) Să se șteargă facturile emise înainte de 01/01/2013.
Vom șterge mai întâi rândurile tabelului tDetaliiFact corespunzătoare facturilor (din tabelul tFacturi) emise înainte de 01/01/2013, apoi ștergem facturile (din tabelul tFacturi) emise înainte de 01/01/2013

```
DELETE tDetaliiFact
FROM tFacturi A inner join tDetaliiFact B on  A.NrFact=B.NrFact
where Data <'01/01/2013'

DELETE tFacturi where data <'01/01/2013'
```

Dacă tabelul din care ştergem este implicat în diverse legături cu alte tabele ca partea unu a unei legături unu la mulți şi opţiunea *delete cascading* este activată atunci ştergerea unui rând va genera și ştergerea rândurilor din tabelelor secundare corelate ce corespund la cheie.

```
ALTER TABLE tDetaliiFact
drop constraint  FK_tDetaliiFact_NrFact

ALTER TABLE tDetaliiFact
add Constraint FK_tDetaliiFact_NrFact
    FOREIGN KEY(NrFact)
REFERENCES tFacturi (NrFact) on delete cascade
```

```
delete from tFacturi where NrFact='f1'
```

va produce stergerea automată din tabelul *tFacturi* și *tDetaliiFact* a tuturor rândurilor pentru care *NrFact* =’f1’

Comanda TRUNCATE

Pentru a şterge rapid toate înregistrările dintr-un tabel se foloseşte comanda TRUNCATE cu următoarea sintaxă:

```
TRUNCATE TABLE tabel
```

Exemplu

```
truncate table tModiPret
```

Ştergerea înregistrărilor cu ajutorul comenzii `TRUNCATE` este mult mai avantajoasă decât eliminarea tabelului şi recrearea lui ulterioară deoarece eliminarea tabelului necesită recrearea indecşilor, constrângerilor de integritate, declanşatorilor etc.