TRANZACȚII

I. OBIECTIVE

1. Tranzacții - concepte și mecanisme
2. Elemente de programare a unei tranzacții în limbajul SQL

II. FUNDAMENTARE TEORETICĂ

2.1.CONCEPTE ȘI MECANISME

In mod obișnuit, un sistem/aplicaţie cu baze de date deservește mai mulți utilizatori care accesează concurent datele din tabele. Accesul concurent al utilizatorilor este asigurat prin capacitatea de *multiprogramare a sistemului de operare* al calculatorului,fapt ce permite execuția concurentă a mai multor procese și prin *suportul pentru concurență oferit de către Sistemul de gestiune a bazei de date.* Accesul concurent al mai multor utilizatori la datele memorate în tabelele unei baze de date *necesită tehnici de menținere a consistenței (corectitudinii) și a siguranței datelor* memorate. Menținerea consistenței bazelor de date în situația în care mai mulți utilizatori le accesează concurent și in condițiile în care pot să apară erori de funcționare (defecte) ale sistemului de calcul se bazează pe *conceptul de tranzacție* care va fi prezentat în secțiunea următoare.

O tranzacție este o unitate logică de prelucrare indivizibilă (atomică) a datelor unei baze de date,prin care se asigură consistența acesteia. In principiu, orice execuţie a unui program care accesează o bază de date poate fi considerată o tranzacţie, dacă baza de date este într-o stare consistentă, atât înainte, câ și după execuție. O tranzacție trebuie să asigure consistența bazei de date indiferent dacă a fost *executată individual sau concurent* cu alte tranzacții, precum și în condițiile în care au apărut erori în cursul execuției tranzacției.

Cele mai importante proprietăți ale tranzactiilor sunt identificate prin acronimul ACID ce reprezintă :atomicitate, consistență, izolare și durabilitate.Semnificaţia acestor proprietăţi este următoarea :

- **Atomicitatea** *(*atomicity*)* este proprietatea unei tranzacții de a reprezenta o unitate de execuție indivizibilă, adică de a executa "totul sau nimic". Dacă o tranzacție este întreruptă dintr-o cauză oarecare, atunci sistemul SGBD va asigura, după eliminarea cauzei care a întrerupt executarea tranzacției, *fie completarea și validarea* tranzactiei, *fie abandonarea* tranzacției și anularea tuturor efectelor acțiunilor acesteia până în momentul întreruperii apărute.
- **Consistența** (consistency*)* unei tranzacții referă proprietatea acesteia de a efectua *modificări corecte* ale bazei de date. Cu alte cuvinte, o tranzacție transformă baza de date dintr-o stare consistentă în altă stare consistentă.
- **Izolarea** (isolation) este proprietatea unei tranzacții de a *face vizibile modificările efectuate numai după ce a fost validată (committed)*. Dacă în acest timp sunt executate alte tranzacții concurente, acestea nu "văd" modificările parțiale efectuate de tranzacția respectivă până în momentul încheierii-validarii tranzactiei.
- **Durabilitarea** (durability, sau permanența - permanency) este proprietatea prin care, după validarea unei tranzacții, modificările efectuate de aceasta în baza de date *nu vor mai fi pierdute* datorită unor defectări ulterioare a sistemului. Proprietatea de durabilitate este asigurată prin metode de refacere (recovery) ale sistemului SGBD.

Tehnici de control a concurenței. Controlul concurenței se poate realiza prin protocoale (set de reguli) impuse tranzacțiilor astfel încât, dacă acestea sunt respectate de fiecare tranzacție, orice planificare în care astfel de tranzacții participă, este serializabilă deci corectă.

Cele mai utilizate tehnici de control a concurenței sunt cele *bazate pe blocare* prin intermediul lacătelor*(locking techniques)* și cele bazate pe *marcaje de timp (timestamps)* și vor fi descrise sumar, astfel :

- Un lacăt *(lock*) este o variabilă asociată cu un *articol al unei baze de date* (poate fi câmp, tuplă, grup de tuple, tabelă) care descrie starea acelui articol în raport cu operațiile care se pot aplica acelui articol.

- *marcaj de timp* este un identificator unic al unei tranzacții, creat de Sistemul de gestiune a bazei de date, care se bazează pe *timpul de start* al tranzacței. Controlul concurenței tranzacțiilor bazat pe marcajele de timp se realizează impunând anumite condiții ordinii de accesare a articolelor în funcție de acestea.

Aplicațiile de baze de date au un *control limitat* asupra opțiunilor de gestiune a tranzacțiilor prin intermediul unor comenzi SQL standardizate, mecanismele enunțate fiind integrate în Sistemul de gestiune a bazei de date, într-o componentă specifică, numită *managerul de tranzacții.*

2.2.INSTRUCTIUNI SQL PENTRU CONTROLUL TRANZACȚILOR

În situația în care mai multe instrucțiuni trebuie realizate împreună , acestea vor fi grupate în cadrul unei tranzacții.

Comanda SQL de specificare a unei tranzacţii este :

```
START TRANSACTION
    [transaction_characteristic [, transaction_characteristic] ...]

transaction_characteristic: {
    WITH CONSISTENT SNAPSHOT
  | READ WRITE
  | READ ONLY}

BEGIN [WORK]
COMMIT [WORK] [AND [NO] CHAIN] [[NO] RELEASE]
ROLLBACK [WORK] [AND [NO] CHAIN] [[NO] RELEASE]
        SET autocommit = {0 | 1}
```

O tranzacție este marcată prin intermediul instrucțiunii START TRANSACTION sau BEGIN [WORK]. Instrucțiunile din cadrul acesteia vor fi executate împreună în momentul în care se apelează instrucțiunea COMMIT ,sau se va reveni la starea anterioară acesteia dacă se apelează ROLLBACK. În cadrul unei tranzacții modul autocommit (marcarea modificărilor pe discul local imediat ce este emisă o anumită instrucțiune) - activat în mod implicit - este dezactivat pe toată perioada acesteia.
Modificarea acestui comportament se poate controla prin intermediul variabilei de sesiune *autocommit.*

În MySQL nu se pot realiza tranzacții imbricate deși alte SGBD permit acest lucru (Oracle,MSSQL), reprezentând în acest sens o limitare pentru anumite contexte logice mai complexe. O astfel de tentativă va rezulta în terminarea tranzacției vechi și începerea unei noi tranzacții. Clauza AND [NO] CHAIN specifică faptului că se va realiza sau nu încă o tranzacție cu același nivel de izolare odată ce sunt

executate instrucțiunile COMMIT sau ROLLBACK. Clauza [NO] RELEASE indică faptul că se dorește sau nu deconectarea sesiunii curente odată ce sunt executate instrucțiunile COMMIT sau ROLLBACK.

De asemenea, se pot specifica proprietăți ce controlează caracteristicile tranzacției:

- WITH CONSISTENT SNAPSHOT - realizează o citire consistentă dacă nivelul de izolare al tranzacției permite acest lucru;
- READ WRITE (implicit) - permite tranzacției să modifice sau să obțină drepturi de blocare (pentru operații de citire sau scriere) asupra tabelelor bazei de date
- READ ONLY - restricționează tranzacția de la a modifica sau de a obține drepturi de blocare asupra tabelelor tranzacționale sau non-tranzacționale care sunt vizibile la nivelul altor tranzacții.

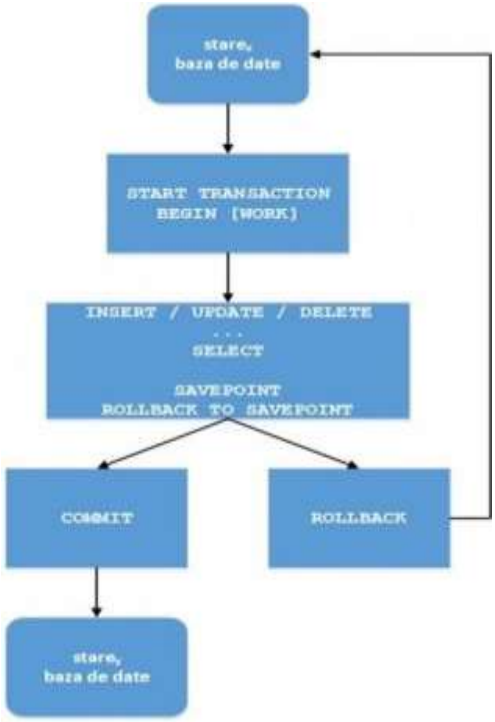

Figura 1

O tranzacție poate fi împărțită în mai multe zone, delimitată de așa-numitele puncte de salvare (*eng.* save-points), prin intermediul comenzii SAVEPOINT identifier. Astfel, se reține starea de la un anumit moment de timp la care se poate reveni prin instrucțiunea ROLLBACK [WORK] TO [SAVEPOINT] identifier. Realizarea acestei operații nu este însă echivalentă cu realizarea tranzacției, ci doar revenirea la o anumită situație, cu pierderea punctelor intermediare ulterioare.
Eliminarea unui astfel de marcaj se realizează prin comanda RELEASE SAVEPOINT identifier.

Comanda SET TRANSACTION stabilește proprietățile tranzacțiilor și admite următoarele opțiuni de setare a modului de gestiune a tranzacțiilor:

```
SET [GLOBAL | SESSION] TRANSACTION
    transaction_characteristic [, transaction_characteristic] ...

transaction_characteristic: {
    ISOLATION LEVEL level
  | access_mode}
 level:
{
     REPEATABLE READ
   | READ COMMITTED
   | READ UNCOMMITTED
   | SERIALIZABLE}
 access_mode: {
READ WRITE
   | READ ONLY}
```

Semnificația parametrilor din comandă de stabilire a proprietăților tranzacției sunt :

- Nivelul de izolare a tranzactiilor (ISOLATION LEVEL) cu valorile posibile, de la cel mai slab către cel mai puternic restrictiv: READ UNCOMMITTED, READ COMMITTED, REPETABLE READ, SERIALIZABLE.
- Modul de acces la articole - cu valorile posibile READ ONLY, READ WRITE.

Controlul comportamentului tranzacțional

Controlul comportamentului tranzacțional se realizeză prin intermediul a două variabile de sesiune:

1.AUTOCOMMIT ce gestioneză momentul la care sunt executate instrucțiunile

1 - (implicit): modificările sunt vizibile imediat;

0 - modificările sunt stocate pe discul local în momentul în care s-a apelat una din instrucțiunile COMMIT sau ROLLBACK;

2.TRANSACTION ISOLATION LEVEL care specifică nivelul de izolare pentru tranzacții (gradul de interacțiune cu alte tranzacții)

- READ UNCOMMITTED - sunt vizibile modificări nesalvate de către alte tranzacții, ceea ce reprezintă o vulnerabilitate față de datele fantomă și citiri irepetabile
- READ COMMITTED - se operează cu valorile stocate pe discul local de către alte tranzacții ceea ce ridică probleme legate de citirile irepetabile
- REPEATABLE READ (implicit) - informațiile actualizate de alte tranzacții sunt disponibile numai în momentul în care s-a realizat și tranzacția din care se solicită datele respective
- SERIALIZABLE - tranzacțiile concurente sunt implementate ca și când acestea s-ar desfășura secvențial, asigurându-se astfel faptul că datele obținute sunt cele stocate în mod real pe discul local

Nivelurile de izolare determină modul în care sistemul de gestiune a bazei de date introduce diferitele mecanisme de control al concurenței (cel mai frecvent lacăte cu stări multiple). De exemplu, pe nivelul READ COMMITTED , sunt prevăzute lacăte partajate pentru toate articolele citite, ceea ce impiedică apariția citirilor improprii, dar aceste lacăte sunt eliberate înainte de terminarea tranzacției si de aceea, pot rezulta citiri nerepetabile și citiri "fantomă" (eronate).

Pe orice nivel de izolare, inclusiv pe cel mai slab (READ UNCOMMITTED), se folosesc mecanisme de control al concurenței tranzacțiilor care previn pierderea actualizărilor. Astfel de anomalii sunt foarte grave, baza de date nu reflectă operațiile care s-au efectuat asupra datelor și nici nu există vreo posibilitate de refacere a acestor pierderi. De aceea nu există nici un nivel de izolare care să permită pierderea actualizării datelor.

Nivelul de izolare	Citire improprie	Citire nerepetabila	Citire fantoma
READ UNCOMMITTED	DA	DA	DA
READ COMMITTED	NU	DA	DA
REPETABLE READS	NU	NU	DA
SERIALIZABLE	NU	NU	NU

Tabelul 7.1 Nivelurile de izolare a tranzacțiilor

Pe toate nivelurile de izolare, cu excepția nivelului SERIALIZABLE, pot să apară diferite anomalii (cele date în tabelul de mai sus), dar aceste anomalii sunt anomalii de citire, care pot fi gestionate de tranzacții, și nu anomalii memorate permanent în baza de date. Cu cât nivelul de izolare a tranzacțiilor este mai scăzut, cu atât pot să apară mai multe anomalii de actualizare, dar crește gradul de concurență

a execuției și scade probabilitatea de apariție a impasului. De aceea, pentru proiectarea unor tranzacții eficiente se recomandă utilizarea unor niveluri de izolare cât mai scăzute, atât cât este posibil pentru ca tranzacțiile respective să se execute totuși corect.

Modul de specificare a tranzacțiilor

Sistemul MySQL admite trei moduri de specificare a tranzacțiilor: tranzactii cu autovalidare *(autocommit),* tranzactii explicite ,tranzactii implicite.

- T*ranzactiile cu autovalidare* oferă modul de lucru **implicit** al sistemului , in care nu este necesara nici o instructiune de control al tranzacțiilor. Toate exemplele de programe studiate anterior ,au presupus acest mod de lucru, astfel ca au putut fi realizate *fara precizări* privind tranzactiile. Modul de lucru cu autovalidare este, de asemenea, modul *implicit* și pentru interfața de programare JDBC.(care se va studia în lucrarea nr. 9)
- *Tranzacțiile explicite* sunt pornite prin instructiunea BEGIN TRANSACTION.
- *Tranzacțiile implicite* presupun setarea modului implicit prin instructiunea SET IMPLICIT_TRANSACTIONS ON. După această setare, următoarea instrucțiune SQL reprezintă începutul unei tranzacții. Instrucțiunea SET IMPLICIT_TRANSACTIONS OFF trece conexiunea respectivă în modul cu autovalidare.

Tranzacțiile sunt administrate la nivelul conexiunii unei aplicații client cu serverul bazei de date (per sesiune). In general, sistemele SGBD implementează protocoalele și funcțiile de control al concurenței și gestionează automat execuția tranzacțiilor și refacerea datelor, pentru a asigura consistența și integritatea datelor memorate

Simularea tranzacțiilor prin operații de blocare

Este posibilă și realizarea de pseudo-tranzacții, prin blocarea / deblocarea anumitor privilegii (scriere, citire) pentru alte tranzacții la nivel de tabelă, un anumit număr de înregistrări, înregistrare.

Instucțiunile corespunzătoare unor astfel de operații sunt LOCK TABLES, respectiv UNLOCK

```
LOCK TABLES
    tbl_name [[AS] alias] lock_type
    [, tbl_name [[AS] alias] lock_type] ...

lock_type: {
    READ [LOCAL]
  | [LOW_PRIORITY] WRITE
}
       UNLOCK TABLES
```

Se observă că este posibilă blocarea concomitentă a mai multor tabele, fiecare pentru operații diferite:

- blocarea în modul citire: atât tranzacția care a realizat blocarea cât și alte tranzacții au doar drepturi de citire asupra tabelei respective;
- blocarea în modul scriere: tranzacția care a realizat blocarea are atât drepturi de citire cât și drepturi de scriere asupra tabelei respective, în timp ce alte tranzacții nu au nici un fel de drepturi asupra acesteia

Detalii suplimentarehttp://www.mysqltutorial.org/mysql-transaction.aspx
Și https://www.tutorialspoint.com/dbms/dbms_transaction.htm

III.PROBLEME REZOLVATE

3.1.Se va studia problema consistentei bazelor de date pe exemplul unui sistem de rezervare a locurilor la curse aeriene. Un număr mare de agenți de vânzări vor accesa relațiile care memorează datele de rezervare și vânzare a biletelor de avion. De exemplu, vor exista relațiile:

CURSE(IdCursa,AeroportPlecare,AeroportSosire,Data,NrLocuriLibere)
PASAGERI(IdPasager,Nume,Prenume,Adresa,NrCreditCard)
REZERVARI(IdRezervare,*IdPasager, IdCursa)*
FACTURI(IdFactura,IdPasager,DataFacturarii,Pret)

Cheile primare și străine au fost marcate ,iar semnificația atributelor acestor relații este destul de clar exprimată chiar prin numele lor. Detalii ca: tipul locului rezervat (turist, business, etc), reduceri de plată a biletului (bilet copil, etc.), mai multe rezervări făcute de același client, intervalul de timp dintre rezervare și cumpărarea biletului, posibilitatea ca o rezervare să fie anulată, etc., au fost ignorate, dat fiind că nu modifică fondul problemei de consistență a bazei de date.

Atunci când un agent de vânzări rezervă un loc la o cursă și vinde biletul corespunzător, se efectuează mai multe operații:

1. Se insereaza o linie noua in tabelul PASAGERI , care contine datele pasagerului.
2. Dacă există locuri libere la cursa dorită, atunci se face propriu-zis rezervarea, prin inserarea unei linii noi în tabelul REZERVARI , linie care conține numărul de identificare al pasagerului, numărul de identificare al cursei și (eventual) numărul locului rezervat; altfel, rezervarea este imposibilă.
3. La achitarea biletului se inserează o linie in tabelul FACTURI. Acestă înregistrare este folosită pentru a tipări o factură, care va fi folosită pentru plata în numerar sau va fi trimisî companiei de cărți de credit.
4. Se emite (tipărește) biletul (pornind de la datele din rezervare și factura corespunzătoare).

Dacă sistemul se defectează dupa ce s-a executat pasul 2, s-a facut o rezervare, dar biletul nu a fost facturat și nici emis. Mai rău, dacă defectiunea are loc dupa ce s-a executat pasul 3, atunci clientului i se trimite factura, dar el nu a primit biletul. Astfel de situatii sunt, bineînțeles, inacceptabile. Chiar dacă nu se defectează sistemul, pot să apara probleme dacă baza de date este accesată concurent de mai mulți utilizatori. De exemplu, daca doi agenti de vânzari atribuie același loc la doi pasageri diferiți, atunci vor fi probleme la îmbarcarea pasagerilor.

Dacă toate acțiunile aferente unei rezervări ar fi grupate ca o operație indivizibilă (atomică), o parte din problemele arătate mai sus ar disparea. O operație indivizibilă de acces la baza de date este numită *tranzacție* și ea fie execută cu succes toate acțiunile și se termină cu o validare a modificărilor efectuate asupra bazei de date *(commit),* fie nu poate efectua (din diferite motive) toate acțiunile și este abandonată și anulata *(abort, rollback*).

In cazul în care o tranzacție a efectuat cu succes toate acțiunile și este validată, în momentul validării toate modificarile efectute asupra bazei de date devin permanente (durabile), vor fi vizibile altor tranzacții și nu mai pot fi anulate. Până în momentul validării, modificările efectuate de tranzacție au un caracter provizoriu, nu sunt vizibile altor tranzacții și pot fi oricând revocate (anulate).

In cazul abandonării unei tranzacții, execuția acesteia este oprită și efectele tuturor acțiunilor executate până în momentul abandonarii sunt anulate, astfel încât baza de date este adusă în starea de dinaintea lansării tranzactiei.

In programul următor este prezentat (foarte simplificat) modul de definire a tranzacției de rezervare a biletelor de avion, descrisă printr-o procedură stocată.

Se definește tranzacția de rezervare a biletelor într-o procedură stocată (sp_rezervare) care are ca argumente de intrare datele necesare rezervării (datele pasagerului, ale cursei dorite, etc.) și returnează un șir de caractere care conține un mesaj privind modul de terminare a tranzacției (validata sau anulata).

Procedura stocata de rezervare a biletelor de avion .

```
IF EXISTS   (SELECT *  FROM dbo.sysobjects
       WHERE name =  'sp_rezervare')   DROP PROCEDURE sp_rezervare
DELIMITER
CREATE  PROCEDURE  sp_rezervare
       @TIdPasager  int,@TNume  varchar(2  0),@TPrenume  varchar(20),  @TAdresa
       varchar(2  0),@TCard  varchar(20),  @TPlecare  varchar(2  0),@TSosire
       varchar(20),
       @TData varchar(2 0),@TPret INT,   @TREZULTAT varchar(20)   OUTPUT
AS
DECLARE  @TIdCursa INT,@TNrLocuri  INT

BEGIN TRANSACTION
INSERT INTO PASAGERI VALUES(@TIdPasager,@TNume,@TPrenume,@TAdresa,@TCard) SELECT  @TNrLocuri
= NrLocuriLibere,   @TIdCursa = IdCursa FROM CURSE
 WHERE AeroportPlecare=@TPlecare AND AeroportSosire=@TSosire AND Data=@TData IF @TNrLocuri>0
BEGIN
       INSERT INTO REZERVARI VALUES(@TIdPasager,@TIdCursa)
       UPDATE CURSE SET NrLocuriLibere=@TNrLocuri-1 WHERE  IdCursa=@TIdCursa INSERT
       INTO FACTURI VALUES(@TIdPasager,GETDATE(),@TPret)
       COMMIT
       SELECT  @TREZULTAT =   'Tranzactie validata'
END
ELSE BEGIN
       ROLLBACK
       SELECT   @TREZULTAT =    'Tranzactie anulata' END   /DELIMITER
```

In procedura stocată *sp_rezervare* se definește o tranzacție explicită (prin instrucțiunea BEGIN TRANSACTION) care grupează toate operațiile aferente unei rezervări: inserarea în tabelul PASAGERI a unei linii care conține date despre pasagerul respectiv (nume, prenume, etc.); după aceasta, din tabelul CURSE se citește numărul de locuri disponibile la cursa dorită (în variabila @TNrLocuri) si se continua operațiile de rezervare numai dacă există locuri disponibile, altfel se face anularea tranzacției (ROLLBACK) și parametrul de iesire @TREZULTAT este setat cu șirul de caractere Tranzacție anulată.

Apelul acestei proceduri se poate face direct ,sau dintr-o aplicație care folosește o interfață de programare.

3.2. Se vor testa implementările propuse pentru baza de date *Bancar* de la adresa http://ftp.utcluj.ro/pub/users/CIvan/IBD/2_Laborator/07_Tranzacții pentru care operațiile curente pe cont au fost implementate folosind tranzacții.

IV.PROBLEME PROPUSE

4.1. Definiți proprietățile unei tranzacții și semnificația acronimului ACID din această perspectivă.

4.2. Ce modalități de control a concurenței implementează SGBD-ul MySQL și explicați imaginativ mecanisme de utilizare a acestora.

Care este modul de lucru tranzacțional implicit pentru INNODB- MySQL ?

4.3. Identificați diferențele între nivelurile posibile de izolare a tranzacțiilor READ UNCOMMITTED, READ COMMITTED, REPETABLE READ, SERIALIZABLE.Care este cel mai restrictive și de ce?

4.4. Ce semnificație are Autocommit, cum poate fi controlat?Dar Rollback? Câte marcaje de tip SAVEPOINT există , unde pot fi urmărite și ce mecanism specific bazelor de date relaționale implementează acestea ?

4.5. Se dă planificarea cu patru tranzacţii care urmeazăv (vertical, este marcat timpul):

	T1	**T2**	**T3**	**T4**
1		Rlock A		
2			Rlock A	
3		Wlock B		
4		Unlock A		
5			Wlock A	
6		Unlock B		
7	Rlock B			
8			Unlock A	
9				Rlock B
10	Rlock A			
11				Unlock B
12	Wlock C			
13	Unlock A			
14				Wlock A
15				Unlock A
16	Unlock B			
17	Unlock C			

Se cere:

a. să fie construit graful de precedenţă al planificării. Pe această bază, se va decide dacă planificarea este sau nu este serializabilă. Descrieţi algoritmul prin care se obţin ramurile grafului, precum şi maniera în care folosiţi algoritmul pentru a construi graful.
b. Dacă planificarea nu este serializabilă, încercaţi modificarea ei pentru a o transforma în o planificare serializabilă. Explicaţi modul în care este făcută modificarea. Prezentaţi atât noua planificare, cât şi noul graf de precedenţă.
c. În cazul în care obţineţi o planificare serializabilă, determinaţi planificarea serială echivalentă folosind procedeul de sortare topologică.

4.6. Fie tranzacţiile T1 şi T2, reprezentate în continuare: **T1** Lock A, Lock B, Unlock A, Unlock B; **T2** Lock B, Unlock B, Lock A, Unlock A.

Analizaţi în câte moduri pot fi planificate tranzacţiile şi câte din planificările găsite vor fi serializabile.

4.7. Creați următoarele tabele în baza de date :

CLIENTI(<u>IdClient</u>,Nume,Prenume,DataNasterii,Adresa,NrCreditCard)
PRODUSE(<u>IdProdus</u>,Categorie,Descriere,PretUnitar,Stoc)
COS(<u>IdCos</u>, *IdClient, IdProdus*,Data,NumarBucati)

Introduceți mai multe linii de date în aceste tabele și scrieti o procedură stocată de realizare a tranzacției de plată a produselor din "coș"de către un client. Verificați funcționarea corectă a programului apelând procedura cu valori care să permită validarea sau anularea tranzacției.