Indrumar Baze de Date – Capitolul 6

GESTIUNEA TRANZACTIILOR

În mod obisnuit, un sistem SGBD deserveste mai multi utilizatori, care acceseaza concurent datele din tabele. Accesul concurent al utilizatorilor este asigurat prin capacitatea de multiprogramare a sistemului de operare al calculatorului gazda, care permite executia concurenta a mai multor procese. Executia concurenta a mai multor procese poate avea loc atât într-un sistem uniprocesor, prin partajarea (împartirea) timpului de executie al procesorului între mai multe procese, cât si într-un sistem multiprocesor în care mai multe procese pot fi executate în mod real simultan, pe mai multe procesoare ale sistemului. Indiferent de numarul de procesoare ale sistemului, accesul concurent al mai multor utilizatori la datele memorate în tabelele unei baze de date necesita tehnici de mentinere a consistentei (corectitudinii) si a sigurantei datelor memorate.

Mentinerea consistentei si a sigurantei bazelor de date în situatia în care mai multi utilizatori le acceseaza concurent si în conditiile în care pot sa apara erori de functionare (defecte) ale sistemului de calcul se bazeaza pe conceptul de tranzactie care va fi prezentat în sectiunea urmatoare.

6.1 TRANZACTII

O tranzactie este o unitate logica de prelucrare indivizibila (*atomica*) *a datelor unei baze de date prin care se asigura consistenta acesteia.*

În principiu, orice executie a unui program care acceseaza o baza de date poate fi considerata o tranzactie, daca baza de date este într-o stare consistenta atât înainte cât si dupa executie. O tranzactie trebuie sa asigure consistenta bazei de date indiferent daca a fost executata individual sau concurent cu alte tranzactii precum si în conditiile în care au aparut defecte ale sistemului hardware în cursul executiei tranzactiei.

Se va studia problema consistentei bazelor de date pe exemplul unui sistem de rezervare a locurilor la curse aeriene. Un numar mare de agenti de vânzari vor accesa relatiile care memoreaza datele de rezervare si vânzare a biletelor de avion. De exemplu, vor exista relatiile:

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

Cheile primare si straine au fost marcate conform conventiilor care au mai fost folosite si în sectiunile precedente, iar semnificatia atributelor acestor relatii este destul de clar exprimata chiar prin numele lor. Detalii ca: tipul locului rezervat (turist, business, etc), reduceri de plata a biletului (bilet copil, etc.), mai multe rezervari facute de acelasi client, intervalul de timp dintre rezervare si cumpararea biletului, posibilitatea ca o rezervare sa fie anulata, etc., au fost ignorate, dat fiind ca nu modifica fondul problemei de consistenta a bazei de date. Atunci când un agent de vânzari rezerva un loc la o cursa si vinde biletul corespunzator, se efectueaza mai multe operatii:

1. Se insereaza o linie noua în tabelul `PASAGERI`, care contine datele pasagerului.
2. Daca exista locuri libere la cursa dorita, atunci se face propriu-zis rezervarea, prin inserarea unei linii noi în tabelul `REZERVARI`, linie care contine numarul de identificare al pasagerului, numarul de identificare al cursei si (eventual) numarul locului rezervat; altfel, rezervarea este imposibila.
3. La achitarea biletului se insereaza o linie în tabelul `FACTURI`. Acesta înregistrare este folosita pentru a tipari o factura, care va fi folosita pentru plata în numerar sau va fi trimisa companiei de carti de credit.
4. Se emite (tipareste) biletul (pornind de la datele din rezervare si factura corespunzatoare).

Daca sistemul se defecteaza dupa ce s-a executat pasul 2, s-a facut o rezervare, dar biletul nu a fost facturat si nici emis. Mai rau, daca 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, bineînteles, inacceptabile. Chiar daca nu se defecteaza sistemul, pot sa apara probleme daca baza de date este accesata concurent de mai multi utilizatori. De exemplu, daca doi agenti de vânzari atribuie acelasi loc la doi pasageri diferiti, atunci vor fi probleme la îmbarcarea pasagerilor.

Daca toate actiunile aferente unei rezervari ar fi grupate ca o operatie indivizibila (atomica), o parte din problemele aratate mai sus ar disparea. O operatie indivizibila de acces la baza de date este numita *tranzactie* si ea fie executa cu succes toate actiunile si se termina cu o validare a modificarilor efectuate asupra bazei de date (*commit*), fie nu poate efectua (din diferite motive) toate actiunile si este abandonata si anulata (*abort*, *rollback*).

În cazul în care o tranzactie a efectuat cu succes toate actiunile si este validata, în momentul validarii toate modificarile efectute asupra bazei de date devin permanente (durabile), vor fi vizibile altor tranzactii si nu mai pot fi anulate. Pâna în momentul validarii, modificarile efectuate de tranzactie au un caracter provizoriu, nu sunt vizibile altor tranzactii si pot fi oricând revocate (anulate).

În cazul abandonarii unei tranzactii, executia acesteia este oprita si efectele tuturor actiunilor executate pâna în momentul abandonarii sunt anulate, astfel încât baza de date este adusa în starea de dinaintea lansarii tranzactiei.

6.1.1 PROPRIETATILE TRANZACTIILOR

Cele mai importante proprietati ale tranzactiilor sunt identificate în literatura prin acronimul ACID: atomicitate, consistenta, izolare, durabilitate.

Atomicitatea (*atomicity*) *este proprietatea unei tranzactii de a reprezenta o unitate de executie indivizibila, adica de a executa "totul sau nimic"*. Daca o tranzactie este întrerupta dintr-o cauza oarecare, atunci sistemul SGBD va asigura, dupa eliminarea cauzei care a întrerupt executarea tranzactiei, fie completarea si validarea tranzactiei, fie abandonarea tranzactiei si anularea tuturor efectelor actiunilor efectuate de tranzactie pâna în momentul întreruperii.

Consistenta (*consistency*) *unei tranzactii înseamna proprietatea acesteia de a efectua modificari corecte ale bazei de date. Cu alte cuvinte, o tranzactie transforma baza de date dintr-o stare consistenta în alta stare consistenta.*

Izolarea (*isolation*) *este proprietatea unei tranzactii de a face vizibile modificarile efectuate numai dupa ce a fost validata* (*committed*). Daca în acest timp sunt executate alte tranzactii concurente, acestea nu "vad" modificarile partiale efectuate de tranzactia respectiva pâna în momentul validarii tranzactiei.

Durabilitarea (*durability*, sau *permanenta* - *permanency*) este proprietatea prin care, dupa validarea unei tranzactii, modificarile efectuate de aceasta în baza de date nu vor mai fi pierdute datorita unor defectari ulterioare a sistemului. Proprietatea de durabilitate este asigurata prin metode de refacere (*recovery*) ale sistemului SGBD.

6.2 TEHNICI DE CONTROL AL CONCURENTEI

Controlul concurentei se poate realiza prin protocoale (set de reguli) impuse tranzactiilor astfel încât, daca acestea sunt respectate de fiecare tranzactie, orice planificare în care astfel de tranzactii participa este serializabila si, deci, corecta.

Cele mai utilizate tehnici de control al concurentei sunt cele bazate pe blocare si cele bazate pe marci de timp (*timestamps*). Controlul concurentei tranzactiilor prin blocare (*locking technique*) se realizeaza folosind zavoare. Un zavor (*lock*) este o variabila asociata cu un articol al unei baze de date care descrie starea acelui articol în raport cu operatiile care se pot aplica acelui articol. O marca de timp este un identificator unic al unei tranzactii, creat de sistemul de gestiune a bazei de date, care se bazeaza pe timpul de start al tranzactiei. Controlul concurentei tranzactiilor bazat pe marci de timp se realizeaza impunând anumite conditii ordinii de accesare a articolelor în functie de marcile lor de timp.

Tehnicile de gestiune a tranzactiilor si de refacere a datelor sunt incluse în componentele sistemelor de gestiune a bazelor de date (administratorul de tranzactii si administratorul de refacere) într-o forma specifica fiecarui SGBD, cu diferite grade de complexitate.

Aplicatiile de baze de date au un control limitat asupra optiunilor de gestiune a tranzactiilor prin intermediul unor comenzi care se bazeaza pe standardul SQL2.

6.2.1 INSTRUCTIUNI SQL PENTRU CONTROLUL TRANZACTIILOR

În standardul SQL2 sunt prevazute urmatoarele comenzi de specificare a tranzactiilor:

```
SET TRANSACTION optiuni
COMMIT [WORK]
ROLLBACK [WORK]
```

Comanda `SET TRANSACTION` stabileste proprietatile tranzactiilor si admite urmatoarele optiuni de setare a modului de gestiune a tranzactiilor:

- Nivelul de izolare a tranzactiilor (`ISOLATION LEVEL`) cu valorile posibile: `READ UNCOMMITTED`, `READ COMMITTED`, `REPETABLE READS`, `SERIALIZABLE`.
- Modul de acces la articole - cu valorile posibile `READ ONLY`, `READ WRITE`.
- Modul de refacere a datelor (`SET CONSTRAINTS`), cu valorile posibile `DEFERRED` (refacere amânata) si `IMMEDIATE` (refacere imediata).

Nivelurile de izolare determina modul în care sistemul de gestiune a bazei de date introduce diferitele mecanisme de control al concurentei (cel mai frecvent zavoare cu stari multiple). De exemplu, pe nivelul `READ COMMITTED`, sunt prevazute zavoare partajate pentru toate articolele citite, ceea ce împiedica aparitia citirilor improprii, dar aceste zavoare sunt eliberate înainte de terminarea tranzactiei si, de aceea, pot rezulta citiri nerepetabile si citiri fantoma.

Pe orice nivel de izolare, inclusiv pe cel mai slab (`READ UNCOMMITTED`), se folosesc mecanisme de control al concurentei tranzactiilor care previn pierderea actualizarilor. Astfel de anomalii sunt foarte grave, baza de date nu reflecta operatiile care s-au efectuat asupra datelor si nici nu exista vreo posibilitate de refacere a acestor pierderi. De aceea nu este prevazut nici un nivel de izolare care sa permita pierderea actualizarii datelor.

Tabelul 6.1 Nivelurile de izolare a tranzactiilor în standardul SQL2.

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

Pe toate nivelurile de izolare, cu exceptia nivelului `SERIALIZABLE`, pot sa apara diferite anomalii (cele date în tabelul de mai sus), dar aceste anomalii sunt anomalii de citire, care pot fi gestionate de tranzactii, si nu anomalii memorate permanent în baza de date. Cu cât nivelul de izolare a tranzactiilor este mai scazut, cu atât pot sa apara mai multe anomalii de actualizare, dar creste gradul de concurenta a executiei si scade probabilitatea de aparitie a impasului. De aceea, pentru proiectarea unor tranzactii eficiente se recomanda utilizarea unor niveluri de izolare cât mai scazute, atât cât este posibil pentru ca tranzactiile respective sa se execute totusi corect.

În general, sistemele SGBD implementeaza protocoalele si functiile de control al concurentei si gestioneaza automat executia tranzactiilor si refacerea datelor, pentru a asigura consistenta si integritatea datelor memorate. Tranzactiile sunt administrate la nivelul conexiunii unei aplicatii client cu serverul bazei de date.

În continuare vor fi prezentate câteva din cele mai importante aspecte ale gestiunii tranzactiilor din perspectiva dezvoltarii aplicatiilor de baze de date: controlul tranzactiilor la nivelul unui limbaj procedural de extensie a limbajului SQL (Transact-SQL) si controlul tranzactiilor în interfetele de programare ODBC si JDBC.

6.2.2 CONTROLUL TRANZACTIILOR ÎN LIMBAJUL TRANSACT-SQL

Sistemul SQL Server admite trei moduri de specificare a tranzactiilor: tranzactii cu autovalidare (*autocommit*), tranzactii explicite si tranzactii implicite.

Modul de lucru cu tranzactii cu autovalidare este modul implicit al sistemului SQL Server, în care nu este necesara nici o instructiune de control al tranzactiilor. Toate exemplele de programe date în capitolele precedente au presupus acest mod de lucru, astfel ca au putut fi realizate fara precizari privind tranzactiile. Modul de lucru cu autovalidare este, de asemenea, modul implicit si pentru interfetele de programare ODBC si JDBC.

Tranzactiile explicite sunt pornite prin instructiunea Transact-SQL `BEGIN TRANSACTION`, care trece conexiunea respectiva în modul cu tranzactii explicite.

Pentru a se initia o tranzactie implicita, se seteaza modul implicit prin instructiunea Transact-SQL `SET IMPLICIT_TRANSACTIONS ON`. Dupa aceasta setare, urmatoarea instructiune SQL reprezinta începutul unei tranzactii. Instructiunea `SET IMPLICIT_TRANSACTIONS OFF` trece conexiunea respectiva în modul cu autovalidare.

Nivelul de izolare a tranzactiilor se stabileste cu instructiunea:

```
SET TRANSACTION ISOLATION LEVEL
   {READ COMMITTED | READ UNCOMMITTED | REPEATABLE READ | SERIALIZABLE}
```

Se observa ca optiunile de stabilire a nivelului de izolare a tranzactiilor sunt aceleasi ca cele din standardul SQL2 (doar denumirea `REPETABLE READ` se deosebeste cu o litera fata de denumirea `REPETABLE READS` din standardul SQL2). Nivelul implicit de izolare al sistemului SQL Server este nivelul `READ COMMITTED`. Instructiunea `BEGIN TRANSACTION`, folosita pentru crearea tranzactiilor explicite, are urmatoarea sintaxa:

```
BEGIN TRAN[SACTION] [nume_trans|@var_nume_trans [WITH MARK['descr']]]
```

În mod optional, o tranzactie poate avea un nume care se poate specifica direct în instructiunea `BEGIN TRANSACTION` (`nume_trans`), sau poate fi continut într-o variabila de program (`@var_nume_trans`). Numele tranzactiei se poate folosi pentru o referire ulterioara (într-o instructiune `COMMIT` sau `ROLLBACK`). Tot optional, se poate desemna unei tranzactii cu nume o marca, continând descrierea tranzactiei (`'descr'`). Numele unei tranzactii marcate poate fi folosit în locul orei sau al datei calendaristice în operatiile de refacere a datelor.

Instructiunile `COMMIT TRANSACTION` si `COMMIT WORK` au acelasi rol ca cel specificat în standardul SQL2, de a valida executia unei tranzactii implicite sau explicite. Diferenta dintre cele doua variante este ca instructiunea `COMMIT TRANSACTION` poate accepta un nume dat de utilizator al unei tranzactii explicite, desi acest nume este ignorat de sistem, si are rol doar de urmarire mai usoara a textului programului. Sintaxa acestor instructiuni este urmatoarea:

```
COMMIT [TRAN[SACTION][nume_trans|@var_nume_trans]]
COMMIT [WORK]
```

Instructiunile `ROLLBACK TRANSACTION` si `ROLLBACK WORK` se folosesc pentru anularea unei tranzactii. Diferenta dintre cele doua variante este ca instructiunea `ROLLBACK TRANSACTION` admite un nume definit de utilizator al tranzactiei anulate. Sintaxa acestor instructiuni este urmatoarea:

```
ROLLBACK [TRAN[SACTION][nume_trans|@var_nume_trans]]
ROLLBACK [WORK]
```

În sistemul SQL Server se pot defini tranzactii explicite imbricate pe mai multe niveluri (*nested*), daca se lanseaza o noua tranzactie explicita cu instructiunea `BEGIN TRANSACTION`, înainte de terminarea tranzactiei curente.

6.2.2.1 Exemplu de tranzactie: Rezervarea biletelor de avion

În programul urmator este prezentat (foarte simplificat) modul de definire a tranzactiei de rezervare a biletelor de avion, descrisa la începutul capitolului, printr-o procedura stocata.

În Programul 6.2 (a) se defineste tranzactia de rezervare a biletelor într-o procedura stocata (`sp_rezervare`) care are ca argumente de intrare datele necesare rezervarii (datele pasagerului, ale

cursei dorite, etc.) si returneaza un sir de caractere care contine un mesaj privind modul de terminare a tranzactiei (validata sau anulata).

Procedura stocata Transact-SQL de rezervare a biletelor de avion (Programul 6.2 -a).

```
IF EXISTS (SELECT * FROM dbo.sysobjects
      WHERE name = 'sp_rezervare') DROP PROCEDURE sp_rezervare
GO
CREATE PROCEDURE sp_rezervare
      @TIdPasager int,@TNume varchar(20),@TPrenume varchar(20),
      @TAdresa varchar(20),@TCard varchar(20),
      @TPlecare varchar(20),@TSosire varchar(20),
      @TData varchar(20),@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
GO
```

În procedura stocata `sp_rezervare` se defineste o tranzactie explicita (prin instructiunea `BEGIN TRANSACTION`) care grupeaza toate operatiile aferente unei rezervari: inserarea în tabelul `PASAGERI` a unei linii care contine date despre pasagerul respectiv (nume, prenume, etc.); dupa aceasta, din tabelul `CURSE` se citeste numarul de locuri disponibile la cursa dorita (în variabila locala `@TNrLocuri`) si se continua operatiile de rezervare numai daca exista locuri disponibile, altfel se face anularea tranzactiei (`ROLLBACK`) si parametrul de iesire `@TREZULTAT` este setat cu sirul de caractere `Tranzactie anulata`.

Apelul acestei proceduri se poate face dintr-un lot de prelucrare (*batch*) Transact-SQL (ca în Programul 6.2(b) care se gaseste în directorul acestui capitol), sau dintr-o aplicatie care foloseste o interfata de programare.

6.2.3 CONTROLUL TRANZACTIILOR ÎN PL/SQL

Instructiunile PL/SQL de control al tranzactiilor sunt foarte asemanatoare cu instructiunile SQL si Transact-SQL. Cele mai importante instructiuni (cu semnificatii evidente) sunt urmatoarele:

```
SET TRANSACTION ISOLATION LEVEL {SERIALIZABLE | READ COMMITED}
COMMIT
ROLLBACK
```

Programul de mai jos exemplifica folosirea acestor instructiuni:

Transactie PL/SQL (Programul 6.5)

```
CREATE OR REPLACE PROCEDURE sp_test (val IN number)
AS
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
```

```
BEGIN
      INSERT INTO ANGAJATI(IdAngajat, Nume, Prenume)
          VALUES(PK_ANGAJATI.NEXTVAL, 'NUME1', 'PRENUME1');
      IF val = 1 THEN COMMIT;
      ELSE ROLLBACK;
      END IF;
END;
```

În procedura stocata `sp_test` se lanseaza o tranzactie (o operatie `INSERT`) care se valideaza (`COMMIT`) sau se anuleaza (`ROLLBACK`) în functie de valoarea parametrului de intrare `val`. Apelul procedurii se face dintr-un bloc PL/SQL astfel:

```
BEGIN
   sp_test(1)
END;
```

Pe baza acestor informatii se poate scrie si testa o procedura stocata PL/SQL pentru realizarea tranzactiei de rezervare a biletelor de avion.

6.2.4 CONTROLUL TRANZACTIILOR PRIN INTERFATA ODBC

Interfata de programare ODBC permite controlul tranzactiilor la nivel de conexiune, care poate fi setata în modul cu autovalidare sau în modul cu validare manuala a tranzactiilor.

Modul cu autovalidare este modul în care fiecare instructiune SQL transmisa SGBD-ului (prin diferitele functii ODBC, cum este functia `SQLExecute()`) este validata automat, daca este executata cu succes. Acesta este modul implicit în ODBC. Pentru comutarea în acest mod se apeleaza functia `SQLSetConnectAttr()` cu numele atributului `SQL_ATTR_AUTOCOMMIT` si valoarea `SQL_AUTOCOMMIT_ON` (prototipul functiei `SQLSetConnectAttr` si explicarea parametrilor acesteia se gaseste în manualul sistemului – *Books Online*).

Modul cu validare manuala este modul în care toate instructiunile SQL transmise pe o conexiune fac parte din aceeasi tranzactie, pâna ce se apeleaza functia `SQLEndTran()`. Aceasta functie poate primi ca argument una din constantele `SQL_COMMIT` sau `SQL_ROLLBACK`, pentru operatia de validare, respectiv de anulare a tranzactiei. Prima comanda transmisa bazei de date dupa apelul functiei `SQLEndTran()` începe o noua tranzactie. Pentru comutarea în modul cu validare manuala se apeleaza functia `SQLSetConnectAttr()` cu numele atributului `SQL_ATTR_AUTOCOMMIT` si valoarea `SQL_AUTOCOMMIT_OFF`.

Se poate observa cu usurinta corespondenta dintre modurile de executie a tranzactiilor din interfata ODBC (care este o interfata independenta de baza de date) si modurile stabilite într-un anumit sistem de gestiune, de exemplu SQL Server: modul cu autovalidare din ODBC corespunde modului cu autovalidare în sistemul SQL Server, iar modul cu validare manuala ODBC corespunde (cu unele mici diferente) modului cu tranzactii explicite din SQL Server.

Nivelurile de izolare a tranzactiilor admise de interfata ODBC sunt similare celor din standardul SQL2 (si din limbajul Transact-SQL) si se pot seta prin apelul functiei `SQLSetConnectAttr()` pentru atributul `SQL_TXN_ ISOLATION_OPTION` care poate lua ca valoare una din constantele definite în fisierul header al bibliotecii (`sql.h`), care stabilesc nivelurile de izolare:

```
SQL_TXN_READ_UNCOMMITTED
SQL_TXN_READ_COMMITTED
SQL_TXN_REPEATABLE_READ
SQL_TXN_SERIALIZABLE
```

Aceste niveluri de izolare sunt corespunzatoare nivelurilor de izolare `READ UNCOMMITTED`, `READ COMMITTED`, `REPEATABLE READ`, `SERIALIZABLE` descrise în sectiunea precedenta pentru limbajul Transact-SQL.

Functia de rezervare a biletelor de avion cu controlul tranzactiilor se poate realiza complet într-un program ODBC. Un astfel de program este Programul 6.3, care se gaseste în directorul acestui capitol si poate fi studiat si executat.

Exercitii - Capitolul 6

6.1 Creati urmatoarele tabele în baza de date proprie din sistemul SQL Server sau Oracle:

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

Introduceti (cel putin) urmatoarele linii în tabelele create:

```
CURSE    (1, 'Otopeni', 'Chicago', '7/23/2003',150)
PASAGERI (1, 'Ionescu', 'Ion', 'Bucuresti', '134265789')
REZERVARI   (1,1,1)
FACTURI  (1,1, '7/03/2003',820)
```

Creati o procedura stocata de rezervare a biletelor de avion. Pentru sistemul SQL server procedura se poate crea prin executia Programului 6.2(a), iar apelul acesteia se face prin executia Programului 6.2(b). Pentru sistemul Oracle scrieti si executati programele de creare si apel al unei proceduri stocate cu functionare similara. Urmariti executia procedurii stocate si modul în care se modifica datele din tabele.

6.2 Modificati continutul tabelelor si programul de apel al procedurii stocate astfel încât tranzactia sa fie abandonata. De exemplu, sa se solicite rezervare la o cursa care nu exista (nu a fost înscrisa în tabelul CURSE), sau sa se solicite bilet la o cursa la care nu mai exista nici un loc. Daca tranzactia a fost abandonata, s-au mai înscris date în tabelele bazei de date?

6.3 Creati urmatoarele tabele în baza de date proprie din sistemul SQL Server sau Oracle:

```
CLIENTI(IdClient,Nume,Prenume,DataNasterii,Adresa)
CONTURI(IdCont,IdClient,Tip,DataCreare,DataExpirare)
TRANSFERURI(IdTransfer,IdContExtragere,IdContDepunere,Data,Suma)
```

Introduceti mai multe linii de date în aceste tabele si scrieti o procedura stocata Transact-SQL sau PL/SQL de realizare a tranzactiei de transfer al unei sume de bani dintr-un cont (identificat prin IdContExtragere) în alt cont (identificat prin IdContDepunere). Verificati functionarea corecta a programului apelând procedura cu valori care sa permita validarea sau anularea tranzactiei.

6.4 Creati urmatoarele tabele în baza de date proprie din sistemul SQL Server sau Oracle:

```
CLIENTI(IdClient,Nume,Prenume,DataNasterii,Adresa,NrCreditCard)
PRODUSE(IdProdus,Categorie,Descriere,PretUnitar,Stoc)
COS(IdCos,IdClient,IdProdus,Data,NumarBucati)
```

Introduceti mai multe linii de date în aceste tabele si scrieti o procedura stocata Transact-SQL sau PL/SQL de realizare a tranzactiei de plata a produselor din "cos"de catre un client. Verificati functionarea corecta a programului apelând procedura cu valori care sa permita validarea sau anularea tranzactiei.