**MINISTERUL EDUCAŢIEI ȘI CERCETĂRII AL REPUBLICII MOLDOVA**

**Universitatea Tehnică a Moldovei**

**Facultatea Calculatoare, Informatică şi Microelectronică**

**Departamentul Ingineria Software și Automatică**

**Programul de studii: Tehnologia informației**

RAPORT

# **LUCRARE DE LABORATOR NR. 4**

# **la Baze de Date**

A efectuat:

st. gr. TI-211          Popa Cătălin

A verificat: Olga Grosu

UTM, Chișinău 2023

**Tranzacții și paralelism**

Prin tranzacții se subînțelege indivizibilitatea, în ceea ce privește impactul bazei de date asupra secvenței de operatori de manipulare a datelor (citire, ștergere, înserare, modificare), fapt ce conduce la una dintre cele două rezultate posibile: fie secvența se realizează, în cazul în care toți operatorii au reușit cu succes, fie întreaga tranzacție este derulată înapoi, în cazul în care cel puțin unul dintre operatori nu a putut fi executat cu succes. Procesarea tranzacțiilor asigură integritatea datelor din baza de date. Astfel, o tranzacție transferă o bază de date dintr-o stare integrată în alta.

**Tranzacția implicită** - stabilește orice instrucțiune individuală INSERT, UPDATE sau DELETE ca unitate de tranzacție.

--tranzactie implicita (stabileste orice instructiune individuala INSERT, UPDATE sau DELETE ca unitate de tranzactie)

--tranzacția de mai jos va insera in baza de date in tabela Furnizori un rand nou cu date

USE SistemGestionalBon

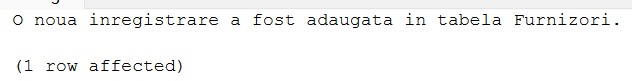
GO

INSERT INTO SistemGestionalBon.dbo.Furnizori

(FurnizorID, Denumire, Adresa)

VALUES(17, 'Import SRL', 'Hristo-Botev 11/1')

**Răspuns**



**Figura 2 - Tranzacție implicită**



**Figura 2 - Tranzacție implicită**

**Tranzacție explicită** - un grup de instrucțiuni Transact-SQL, începutul și sfârșitul cărora sunt indicate de comenzi, precum BEGIN TRANSACTION, COMMIT și ROLLBACK.

--tranzactie explictita

--aceasta tranzactie de asemenea va insera on rand nou cu date, dar deja folosim BEGIN TRANSACTION si COMMIT

BEGIN TRANSACTION

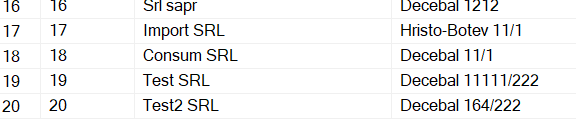
INSERT INTO SistemGestionalBon.dbo.Furnizori (FurnizorID, Denumire, Adresa)

VALUES(20, 'Test2 SRL', 'Decebal 164/222')

COMMIT;



**Figura 3 - Tranzacție explicită**



**Figura 4 – Rezultatul înserării**

--Mai jos este un exemplu de tranzactie cand la fel inseram date intr-o tabela si salvam transactia respectiva, apoi am folosit rollback pentru a anula transactia

BEGIN TRANSACTION

INSERT INTO SistemGestionalBon.dbo.Furnizori (FurnizorID, Denumire, Adresa)

VALUES(22, 'Import SRL', 'Sarmizegetusa 22')

SAVE TRANSACTION a

--ROLLBACK se foloseste pentru a anula tranzactia

ROLLBACK TRANSACTION a



**Figura 5 – Tranzactie cu folosirea SAVE si ROLLBACK**

**Tranzacții imbricate**

Tranzacțiile imbricate (nested transactions) în SQL sunt tranzacții care sunt încapsulate în cadrul altei tranzacții. În general, într-o tranzacție imbricată, o tranzacție internă este începută și încheiată în cadrul unei tranzacții externe. Această abordare poate fi utilă atunci când avem nevoie de niveluri diferite de tranzacționalitate pentru operațiile noastre.

--mai jos am realizat o tranzactie imbricata care se include intr-o tranzactie externa si depinde de ea

--mai intai se adauga un produs in tabela Produse si daca a fost inserat cu succes, se va insera si in tabela cu valabilitate

BEGIN TRANSACTION; -- Tranzactia externa incepe

INSERT INTO Produse(ProdusID, NumeProdus, MagazinID, PretProdus, FurnizorID) VALUES (16, 'Chifir', 3, 15,4);

BEGIN TRANSACTION; -- Tranzactia imbricata incepe

-- Verificam daca inserarea in Orders a avut loc cu succes

IF @@ERROR = 0

BEGIN

-- Daca nu sunt erori, inseram in OrderDetails

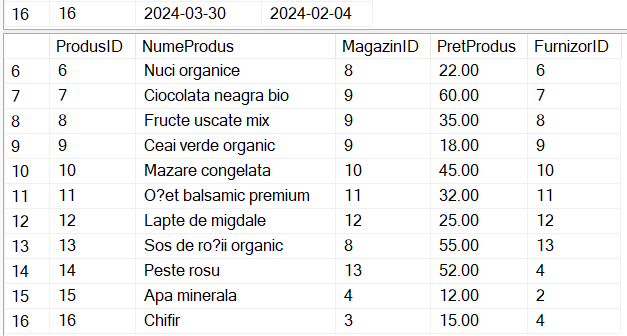
INSERT INTO Valabilitate(ProdusID, DataProducere, DataExpirare) VALUES (16, GETDATE(), '02.04.2024');

END

COMMIT TRANSACTION; -- Tranzactia imbricata se incheie

-- Tranzactia externa se incheie

COMMIT TRANSACTION;



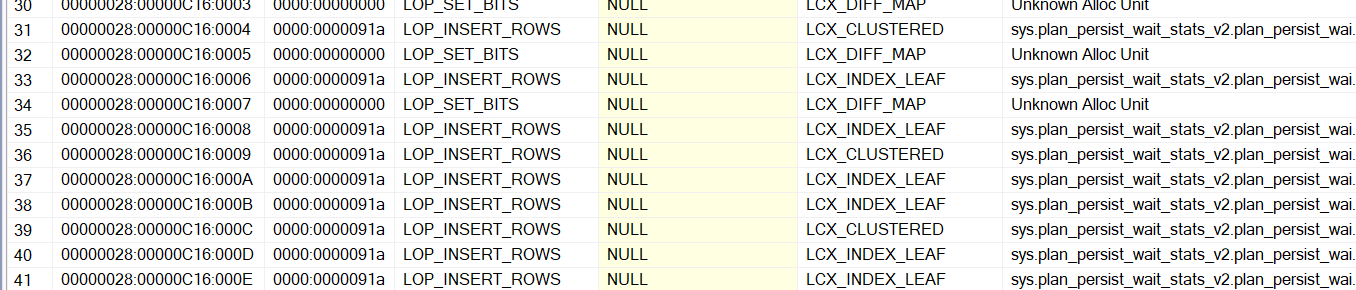
**Figura 6 – Tranzacție imbricată**

**Jurnal de tranzacții**

Jurnalul de tranzacții este gestionat intern și nu este accesibil direct pentru utilizatorii obișnuiți într-un format ușor de citit. Cu toate acestea, există anumite funcții și proceduri stocate pe care le puteți folosi pentru a obține informații din jurnalul de tranzacții. O modalitate obișnuită de a inspecta jurnalul de tranzacții este utilizarea funcției fn\_dblog.

--mai jos este un exmplu cum putem vizualiza jurnalul de tranzactii folosind functia fn\_dblog

SELECT \* FROM fn\_dblog(NULL, NULL);



**Figura 7 – Vizualizare jurnal de tranzacții**

**Blocări**

Blocările sunt mecanisme utilizate pentru a gestiona accesul concurent la resursele comune. Atunci când mai multe tranzacții încearcă să acceseze sau să modifice aceleași resurse simultan, pot apărea conflicte care pot duce la inconsistente în date sau la pierderea integrității tranzacțiilor. Utilizarea blocărilor ajută la prevenirea acestor conflicte.

--Mia jos am folosit o blocare cu parametrul TABLE, care are valoare implicita si stabileste escaladarea blocarii

--la nivelul tabelului (la fel sunt parameetrii auto si disable)

ALTER TABLE [myschema].[MagazinCopy]

SET ( LOCK\_ESCALATION = TABLE )



**Figura 7 – Blocari**

**Nivelul de izolare**

Nivelul de izolare în SQL se referă la modul în care baza de date gestionează accesul concurent la date și modificațiile efectuate de tranzacții simultane. Există mai multe niveluri de izolare, iar fiecare nivel oferă un echilibru între consistență și performanță. Voi oferi exemple pentru câteva niveluri de izolare comune, inclusiv Read Uncommitted, Read Committed, Repeatable Read și Serializable.

**Read Uncommitted**

Acest nivel permite tranzacțiilor să citească date necomise (neconfirmate) și să vadă modificările necomise ale altor tranzacții. Este cel mai puțin restrictiv nivel de izolare și poate duce la fenomene de citire murdară.

--realizarea unui nivel de izolare Read Uncommitted

-- Tranzacția 1

BEGIN TRANSACTION;

UPDATE Bon SET SumaTotala = SumaTotala - 50.00 WHERE BonID = 1;

-- In acest moment, datele modificate de tranzactia 1 nu sunt inca comise

-- Tranzacția 2

BEGIN TRANSACTION;

SELECT SumaTotala FROM Bon WHERE BonId = 1; -- Va citi modificarile necomise din tranzactia 1

COMMIT; -- Comite tranzactia 1 si confirma modificarile

****

**Figura 8 – Nivelul de izolare Read Uncommitted**

**Read Committed**

Acest nivel permite tranzacțiilor să citească numai datele comise și să nu vadă modificările necomise ale altor tranzacții.

--exemplu de utilizare a nivelului de izolare read committed

-- Tranzactia 1

BEGIN TRANSACTION;

UPDATE Bon SET SumaTotala= SumaTotala- 10.00 WHERE BonID= 4;

COMMIT; -- Confirma si comite modificarile

-- Tranzactia 2

BEGIN TRANSACTION;

SELECT SumaTotala FROM Bon WHERE BonID = 1; -- Va citi doar datele comise, nu modificarile necomise

COMMIT;



**Figura 9 – Nivelul de izolare Read Committed**

**Repeatable Read**

Acest nivel garantează că o tranzacție va vedea aceleași date în cadrul tranzacției sale, chiar dacă alte tranzacții fac modificări asupra datelor.

--exemplu de nivel de izolare Repeatable Read unde la fel avem 2 tranzactii date ca exemplu de aplicare

-- Tranzactia 1

BEGIN TRANSACTION;

SELECT SumaTotala FROM Bon WHERE BonID = 1; -- Va citi balanțta curenta

-- In acest moment, alte tranzactii nu pot modifica datele citite in cadrul acestei tranzactii

-- Tranzactia 2

BEGIN TRANSACTION;

UPDATE Bon SET SumaTotala = SumaTotala + 100.00 WHERE BonID = 1;

COMMIT;

-- Continuare tranzactia 1

SELECT SumaTotala FROM Bon WHERE BonID = 1; -- Va citi aceeasi balanta ca la inceputul tranzactiei

COMMIT;



**Figura 10 – Nivelul de izolare Repeatable Read**

**Serializable**

Acest nivel garantează că rezultatele tranzacției sunt aceleași ca și cum tranzacțiile ar fi rulate în serie, una după alta.

--exemplu nivel de izolare Serializable unde tranzactiile sunt rulate ca si cum in serie una dupa alta

-- Tranzactia 1

BEGIN TRANSACTION;

SELECT SumaTotala FROM Bon WHERE BonID = 1; -- Va citi balanța curenta

-- In acest moment, nicio alta tranzactie nu poate modifica datele citite in cadrul acestei tranzactii

-- Tranzactia 2

BEGIN TRANSACTION;

UPDATE Bon SET SumaTotala = SumaTotala + 100.00 WHERE BonID = 1; -- Va aștepta blocarea pana la finalizarea tranzactiei 1

COMMIT; -- Se va comite după ce tranzactia 1 se finalizeaza

-- Continuare tranzactia 1

SELECT SumaTotala FROM Bon WHERE BonID = 1; -- Va citi aceeasi balanta ca la inceputul tranzactiei

COMMIT;



**Figura 11– Nivelul de izolare Serializable**

**Concluzie**

În concluzie, în lucrarea data am explorat conceptele de tranzacții și blocări în SQL, ilustrând diferite scenarii și niveluri de izolare. Exemplele practice au demonstrat tranzacțiile implicite și explicite, gestionarea tranzacțiilor imbricate, precum și utilizarea nivelurilor de izolare Read Uncommitted, Read Committed, Repeatable Read și Serializable. De asemenea, s-a examinat vizualizarea jurnalului de tranzacții folosind fn\_dblog, precum și gestionarea blocării la nivelul tabelelor prin stabilirea escaladării blocării. Aceste concepte sunt fundamentale în asigurarea consistenței, integrității și performanței în operațiile bazelor de date.