# TRANZACȚII

# CONTROLUL CONCURENȚEI ÎN SQL SERVER

Realizator: Emilia Pop

# Tranzacții în SQL Server

- Tranzactie = bloc de instrucțiuni SQL executate de un user (de ex. select, insert, update, delete) in vederea modificarii bazei de date.
- **Serverul de date trebuie să garanteze execuţia** tuturor **intrucţiunilor grupate** in **tranzacţie. Dacă** nu se poate executa cel putin una din operatii, atunci **trebuie anulate toate operaţiile din tranzacţie.**
- **După execuţia** tuturor operatiilor din tranzactie, trebuie precizat daca modificarile produse se pastreaza (COMMIT) sau se anuleaza (ROLLBACK).
- Comenzi pentru tranzactii: BEGIN TRAN, ROLLBACK TRAN, COMMIT TRAN

# Tranzacții în SQL Server

- SQL Server folosește tranzacțiile pentru a grupa mai multe operații într-o singură unitate
  - Bucata fiecărui user este procesată prin folosirea unei tranzacții diferite
  - Pentru a maximiza tranzitarea, tranzacțiile ar trebui să se execute în paralel
  - Proprietățile ACID ale tranzacțiilor cer ca rezultatul efectiv să fie ca și atunci când tranzacțiile s-ar executa serial

# ACID

- A – Atomicitate (Atomicity) – se execută toate operațiile din bloc sau nimic
- C – Consistență (Consistency) – datele trebuie să se găsească într-o stare consistentă după orice tranzacție (păstrarea integrității BD – datele nu sunt alterate)
- I – Izolare (Isolation) – tranzacția în execuție nu este afectată de alte tranzacții (este izolată de altă tranzacție – niveluri de izolare)
- D – Durabilitate (Durability) – după execuția cu succes a unei tranzacții (committed), modificările realizate persistă în BD

# Niveluri de izolare în SQL Server

- **READ UNCOMMITTED** – fără blocări la citire
- **READ COMMITTED** – păstrează blocările pe durata execuției codului (implicit) (*Dirty reads*)
- **REPEATABLE READ** – păstrează blocările pe durata tranzacției *(Unrepeatable reads)*
- **SERIALIZABLE** - păstrează blocările și blocările key range pe durata întregii tranzacții *(Phantom reads)*
- **SNAPSHOT** - funcționează pentru date snapshot
- SQL syntax:
  - **SET TRANSACTION ISOLATION LEVEL …**

# Niveluri de izolare în SQL Server – probleme de concurenta

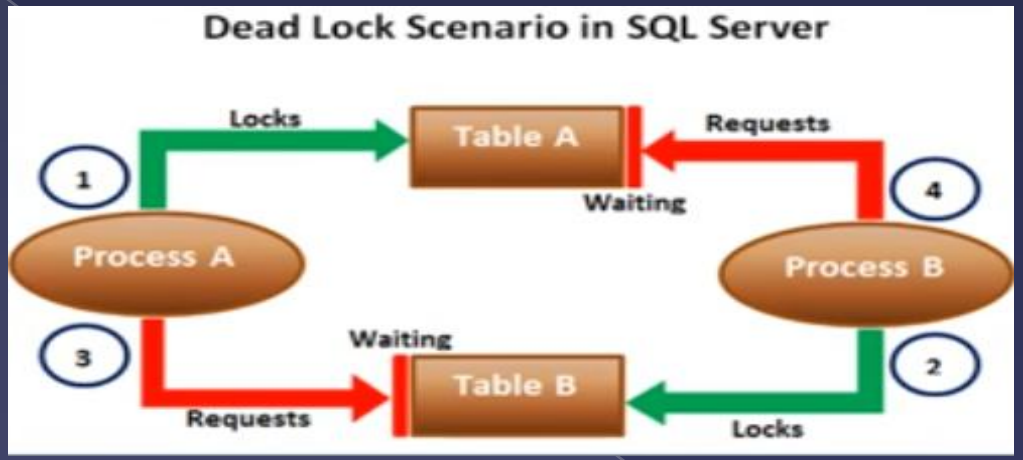
| Nivel de izolare | Problema de concurenta | Nivel de izolare – pentru rezolvare problema de concurenta |
|---|---|---|
| **READ UNCOMMITTED** | *Dirty reads* | **READ COMMITTED** |
| **READ COMMITTED** | *Nonrepeatable reads* | **REPEATABLE READ** |
| **REPEATABLE READ** | *Phantom reads* | **SERIALIZABLE** |
| **SERIALIZABLE** | | |

Rezolvarea unei probleme de concurenta presupune schimbarea nivelului de izolare un nivel de izolare imediat superior.

# Niveluri de izolare in SQL Server

| Common Name | Chaos | Read Uncommitted | Read Committed | Repeatable Read | Serializable |
|---|---|---|---|---|---|
| Lost Updates? | Yes | No | No | No | No |
| Dirty Reads? | Yes | Yes | No | No | No |
| Unrepeatable Reads? | Yes | Yes | Yes | No | No |
| Phantoms? | Yes | Yes | Yes | Yes | No |

# Deadlocks




- MS SQL Server recunoaște un deadlock.
- Una din cele 2 tranzactii se termina cu eroare (1205) si cealalta cu succes.

# Deadlocks

- Error 1205 - ar trebui să fie prinsă și gestionată corespunzător
- Msg 1205, Level 13, State 51, Line 9
  - Transaction (Process ID 63) was deadlocked on lock resources with another process and has been chosen as the deadlock victim. Rerun the transaction.
- SET LOCK TIMEOUT - folosită pentru a specifica cât timp (în milisecunde) o tranzacție așteaptă pentru un obiect blocat ca să fie eliberat (0= terminare imediată)
- SET DEADLOCK PRIORITY
  - Valori: Low, Medium, High, numeric -10 -5 0 5 10

# Deadlocks

- Dacă DEADLOCK_PRIORITY sunt diferite (în cele 2 tranzacții), sesiunea cu cea mai mică prioritate e aleasă victimă (câștigă cealaltă tranzacție și modificările sunt cele din date de tranzacția câștigătoare)
- Dacă ambele sesiuni au aceeași prioritate, tranzacția cea mai puțin scumpă la ROLLBACK  e aleasă victimă
- Dacă ambele sesiuni au aceeași prioritate și același cost, victima se alege aleator