Seminar 3

Tranzacţii

Controlul concurenţei în MS SQL Server

Tranzacții în SQL Server

- SQL Server utilizează tranzacții pentru compunerea mai multor operații într-o singură unitate de lucru
  - Acțiunile fiecărui utilizator sunt procesate utilizând o tranzacție diferită
  - Pentru a maximiza *throughput*-ul, tranzacțiile ar trebui să se execute în paralel
  - Proprietățile ACID ale tranzacțiilor: obiectiv - serializabilitate

Tranzacții în SQL Server

- Mecanisme pentru invocarea tranzacțiilor:
  - **fiecare comandă** este o tranzacție dacă nu se specifică altceva
  - BEGIN TRAN, ROLLBACK TRAN, COMMIT TRAN - cel mai des utilizate
    - BEGIN TRAN marchează punctul de început al unei tranzacții explicite
    - ROLLBACK TRAN întoarce o tranzacție implicită sau explicită până la începutul tranzacției sau până la un *savepoint* din interiorul tranzacției
    - COMMIT TRAN marchează finalul unei tranzacții implicite sau explicite efectuate cu succes

Tranzacții în SQL Server

- SET IMPLICIT_TRANSACTIONS ON
  - Se marchează doar finalul tranzacției (tranzacții înlănțuite)
  - În acest caz, o nouă tranzacție este începută în mod implicit atunci când tranzacția anterioară este finalizată (fiecare tranzacție este finalizată explicit cu COMMIT sau ROLLBACK)
- SET XACT_ABORT ON
  - **orice** erori SQL vor duce la roll back-ul tranzacției

Tranzacții în SQL Server

@@TRANCOUNT returnează numărul de instrucțiuni BEGIN TRANSACTION care au avut loc pe conexiunea curentă

- Instrucțiunile BEGIN TRANSACTION incrementează @@TRANCOUNT cu 1
- Instrucțiunile COMMIT TRANSACTION decrementează @@TRANCOUNT CU 1
- ROLLBACK TRANSACTION setează @@TRANCOUNT pe 0, excepție ROLLBACK TRANSACTION *savepoint_name*, care nu afectează @@TRANCOUNT

Tipuri de tranzacții

- SQL Server poate utiliza tranzacții locale sau distribuite
  - Imbricarea tranzacțiilor (dar nu tranzacții imbricate) este permisă
  - *Savepoints* cu nume - rollback-ul unei porțiuni dintr-o tranzacție

Probleme de concurență

- Izolarea tranzacțiilor în SQL Server rezolvă patru probleme majore de concurență:
  - Lost updates - când doi scriitori modifică aceleași date
  - Dirty read - când un cititor citește date necomise
  - Unrepeatable read - când o înregistrare existentă se schimbă în cadrul unei tranzacții
  - Phantoms - când sunt adăugate noi înregistrări și apar în cadrul unei tranzacții

Probleme de concurență

- SQL Server asigură izolarea prin blocări
  - Blocările pentru scriere sunt blocări exclusive, blocările pentru citire permit existența altor cititori
  - O tranzacție *well-formed* obține tipul corect de blocare înainte de a utiliza date
  - O tranzacție *two-phased* menține toate blocările până când s-au obținut toate acestea
  - Nivelurile de izolare determină durata blocărilor, cât timp sunt menținute acestea

Blocarea în SQL Server

- Blocările sunt gestionate de obicei dinamic de Lock Manager, nu prin intermediul aplicațiilor
- Granularitatea blocărilor:
  - *Row/Key, Page, Table, Extent (grup contiguu de 8 pagini) , Database*
  - Blocările pot fi atribuite la mai mult de un nivel → ierarhie de blocări înrudite
  - *Lock escalation* > 5000 blocări pe obiect (pro&contra)
- Durata blocărilor (specificată prin niveluri de izolare):
  - până la finalizarea operației
  - până la finalul tranzacției

Blocarea în SQL Server

	S	U	X
S	Da	Da	Nu
U	Da	Nu	Nu
X	Nu	Nu	Nu

- Tipuri de blocări
  - ***Shared*** (S) – pentru citirea datelor
  - ***Update*** (U) – blocare S - se anticipează X
  - ***Exclusive*** (X) – pentru modificarea datelor; incompatibile cu alte blocări (excepție hint-ul NOLOCK și nivelul de izolare READ UNCOMITTED)
  - ***Intent*** (IX, IS, SIX) – intenție de blocare (îmbunătățirea performanței)
  - ***Schema*** (Sch-M, Sch-S) – modificare schemă, stabilitate schemă (Sch-M și Sch-S nu sunt compatibile)
  - ***Bulk Update*** (BU) – blochează întregul tabel în timpul *bulk insert*
  - ***Key-range –*** blochează mai multe rânduri pe baza unei conditii

Blocări speciale

- Blocări *schema*
  - nu blochează LMD, doar LDD
- Blocări *Bulk Update*
  - Blocare explicită (TABLOCK)
  - Automat în timpul SELECT INTO
- Blocări *Application*
  - Aplicațiile pot utiliza *lock manager-ul* intern al SQL Server - blocare resurse aplicație

Niveluri de izolare în SQL Server

- **`READ UNCOMMITTED`**: fără blocare la citire
- **`READ COMMITTED`**: menține blocările pe durata execuției instrucțiunii (default) (*Dirty reads*);
- **`REPEATABLE READ`**: menține blocările pe durata tranzacției (*Unrepeatable reads*);
- **`SERIALIZABLE`**: menține blocările și blocările *key range* pe durata întregii tranzacții (*Phantom reads*);
- **`SNAPSHOT`**: lucrează pe un *snapshot* al datelor
- sintaxă SQL:

```
SET TRANSACTION ISOLATION LEVEL …
```

Grade de izolare

Denumire	Haos	Read Uncommitted	Read Committed	Repeatable Read	Serializable
Lost Updates?	Da	Nu	Nu	Nu	Nu
Dirty Reads?	Da	Da	Nu	Nu	Nu
Unrepeatable Reads?	Da	Da	Da	Nu	Nu
Phantoms?	Da	Da	Da	Da	Nu

Blocare *Key Range*

- blochează seturi de rânduri controlate de un predicat …**`WHERE salary between 35000 and 45000`**
- necesară blocarea datelor care nu există!
  - dacă utilizarea "**`salary between 35000 and 45000`**" nu returnează niciun rând prima dată, nu ar trebui să returneze niciun rând nici la scanările ulterioare
- versiunea mai veche - blocare unități mai mari pentru a preveni *phantoms*
  - înainte de SQL Server 7.0, doar paginile și tabelele erau blocate

Blocări *Transaction Workspace*

- Indică o conexiune
  - Fiecare conexiune la o bază de date obține blocare ***Shared_Transaction_Workspace***
  - Excepții - conexiunile la master și tempdb
- Utilizate pentru prevenirea:
  - DROP
  - RESTORE

Deadlocks

- MS SQL Server recunoaște un *deadlock*
- Implicit, tranzacția mai nouă este terminată
  - Error 1205 – ar trebui detectată și gestionată corespunzător
- SET LOCK TIMEOUT: se menționează cât timp (în milisecunde) o tranzacție așteaptă ca un obiect blocat să fie eliberat (0 = terminare imediată)
- SET DEADLOCK PRIORITY
  - Valori: Low, Medium, High, numeric între -10 și 10

Reducerea situațiilor de *deadlock*

- Tranzacții scurte & într-un singur *batch*
- Colectarea și verificarea datelor input de la utilizatori înainte de deschiderea unei tranzacții
- Accesarea resurselor în aceeași ordine în tranzacții
- Utilizarea unui nivel de izolare mai scăzut sau a unui nivel de izolare *row versioning*
- Accesarea celei mai mici cantități posibile de date în tranzacții