

1. Einführung

2. Datenbankentwurf

3. Datenbankimplementierung

4. Physische Datenorganisation

5. Anfrageoptimierung

6. Transaktionsverwaltung

7. Datensicherheit und Wiederherstellung

8. Business Intelligence

D a t e n b a n k e n

Gliederung

1.

Einführung

2.

Datenbankentwurf

3.

Datenbankimplementierung

4.

Physische Datenorganisation

5.

Anfrageoptimierung

6.

Transaktionsverwaltung

7.

Datensicherheit und Wiederherstellung

8.

Business Intelligence

Gliederung

6.

Transaktionsverwaltung



Transaktionen



Phänomene



Sperren



Isolationslevel



Beispiele

Transaktionen

Als Transaktion bezeichnet man in der Informatik eine Folge von Operationen, die als eine logische Einheit betrachtet werden.

Insbesondere wird für Transaktionen gefordert, dass sie entweder vollständig oder überhaupt nicht ausgeführt werden (Atomizität).

Realisieren den Übergang vom konsistenten Zustand A in den konsistenten Zustand B.

Transaktionen

Bei der Ausführung von Transaktionen muss das Transaktionssystem die ACID-Eigenschaften garantieren:

- **Atomarität (Atomicity):** Eine Transaktion wird entweder ganz oder gar nicht ausgeführt. Wenn eine Transaktion abgebrochen wird, ist das System unverändert.
- **Konsistenz. (Consistency):** Nach Ausführung der Transaktion muss der Datenbestand in einer konsistenten Form sein, wenn er es bereits zu Beginn der Transaktion war.
- **Isolation (Isolation):** Bei gleichzeitiger Ausführung mehrerer Transaktionen dürfen sich diese nicht gegenseitig beeinflussen.
- **Dauerhaftigkeit (Durability):** Die Auswirkungen einer Transaktion müssen im Datenbestand dauerhaft bestehen bleiben.

Transaktionen

Bisher haben sie den Autocommit-Modus benutzt.
Dabei wird jeder Befehl als einzelne Transaktion ausgeführt.

- Begin der Transaktion
 - SQL – Befehl
- Ende der Transaktion
- Begin der Transaktion
 - SQL – Befehl
- Ende der Transaktion

Dies ist das Standardverhalten des MS-SQL Servers und vieler anderer Datenbanksysteme.

Transaktionen

Im Expliziten- & Impliziten-Modus können mehrere Befehle zu einer Transaktion gebündelt werden.

➤ Begin der Transaktion

- SQL – Befehl
- SQL – Befehl
- SQL – Befehl
- ...

➤ Ende der Transaktion

Dies ist sinnvoll um die Konsistenz der geschriebenen Daten zu gewährleisten, wenn diese voneinander abhängig sind (atomar).

Transaktionen

Um eine Transaktionen einzuleiten gibt es zwei Möglichkeiten:

- **SET IMPLICIT_TRANSACTION ON** - Impliziter-Modus
- **BEGIN TRANSACTION** - Expliziter-Modus

Das Ende der Transaktion muss in beiden Modi explizit angegeben werden:

- **COMMIT TRANSACTION**
 - Die Transaktion wird geschrieben
- **ROLLBACK TRANSACTION**
 - Die Transaktion wird verworfen

Gliederung

6.

Transaktionsverwaltung



Transaktionen



Phänomene



Sperren



Isolationslevel



Beispiele

Phänomene

Lost Update

- Transaktion 1 liest eine Tabellenzeile
- Transaktion 2 liest dieselbe Zeile
- Transaktion 1 schreibt die Zeile
- Transaktion 2 schreibt ebenfalls
- Änderungen von T1 sind verloren

Wird durch Sperrverfahren verhindert

Phänomene

Dirty Read

- Transaktion 1 ändert Wert in Tabellenzeile
- Transaktion 2 liest diesen geänderten Wert
- Transaktion 1 bricht ab mit ROLLBACK
- Transaktion 2 arbeitet nun mit einem ungültigen Wert da durch den ROLLBACK dieser nie in der Datenbank existiert hat

Phänomene

Non-Repeatable Read

- Transaktion 1 liest eine Anzahl Zeilen
- Transaktion 2 ändert einige dieser Zeilen bzw. fügt Zeilen hinzu.
- Transaktion 1 wiederholt die SQL Abfrage und bekommt nun eine andere Ergebnismenge als beim ersten lesen.

Phänomene

Phantom Rows

- Spezialfall der Non-Repeatable Reads
- Hier wird nur das Hinzufügen von Zeilen durch eine andere Transaktion betrachtet
- Bei zweimaliger Ausführung eines Statements bekommt die Transaktion bei der zweiten Ausführung mehr Zeilen zurück.

Phänomene

Über die Einstellung des Isolationslevel kann kontrolliert werden, welche Phänomene für eine Transaktion auftreten können.

Vor allem Anwendungsentwickler sollten diese Thematik verstehen.

Gliederung

6.

Transaktionsverwaltung



Transaktionen



Phänomene



Sperren



Isolationslevel



Beispiele

Sperren

Jede Transaktion, die Sperren nutzt, muss die folgenden Sperrbedingungen einhalten:

- Für jedes Objekt, welches von der Transaktion benutzt werden soll, muss vor der Nutzung eine Sperre angefordert werden.
- Keine Transaktion fordert eine Sperre an die sie schon besitzt.
- Spätestens am Transaktionsende werden alle Sperren aufgehoben
- Die Sperren anderer Transaktionen müssen beachtet werden → warten auf Freigabe.

Sperren

Das 2-Phasen Sperrprotokoll:







- 2PL gibt vor, dass jede Transaktion zu Beginn alle ihre Sperren anfordert und am Ende alle Sperren wieder abgibt.
- Striktes 2PL verschärft das Protokoll durch die Forderung, dass Sperren erst zurückgegeben werden wenn der commit der Transaktion sichergestellt ist.
- Dadurch wird garantiert, dass Änderungen der Transaktion nicht vor dem Ende der Transaktion sichtbar werden.

Sperren

RX-Sperrverfahren

Bsp.:

- T1 liest Zeile 1 und setzt R Sperre
- Lesesperren sind zueinander kompatibel
T2 liest Zeile 1 und setzt ebenfalls R Sperre → Beide lesen
- T1 will nun schreiben und versucht R Sperre in X Sperre zu verwandeln →
Schlägt fehl da andere Transaktion noch R Sperre hält, T1 wartet nun auf die Freigabe der Sperre
- T2 wird beendet
- T1 kann nun R Sperre in X Sperre verwandeln Schreiben.
- T3 will Zeile 1 lesen, schlägt fehl da Zeile 1 nun X Sperre hält T3 wartet auf Ende von T1

		Aktueller Modus		
		NL	R	X
Angeforderter Modus	R			
	X			

Legende:

NL: Keine Sperre

R: Lesen

X: Schreiben

Sperren

Sperrarten in MS-SQL:

- Shared
- Exclusive
- Update
- Intent
 - Intent-Shared
 - Intent-Exclusive
 - Shared mit Intent-Exclusive
- Schema
- Bulk Update

Sperren

Shared

- Entspricht der Read Sperre im RX-Sperrverfahren

Exclusive

- Entspricht der Write Sperre im RX-Sperrverfahren

Update

- Wird angefordert, weil die Ressource unter Umständen aktualisiert wird. Sie wird in eine Shared oder Exclusive Sperre umgewandelt. Je nachdem ob die Ressource verändert wird oder nicht.

Sperren

Intent Sperren geben an, das man beabsichtigt etwas mit der Ressource zu tun, aber noch auf die Freigabe wartet.

Intent-Shared

- Eine Transaktion möchte eine Shared-Sperre setzen

Intent-Exclusive

- Eine Transaktion möchte eine Exclusive-Sperre setzen

Shared mit Intent-Exclusive

- Eine Transaktion beabsichtigt für einige Ressourcen Shared und für andere Exclusive Sperren zu setzen

Sperren

Schema

- Werden verwendet, wenn das Schema einer Tabelle verwendet wird. Wenn z.B. Spaltentypen geändert werden, eine neue Spalte hinzukommt, ...

Bulk Update

- Wird verwendet, wenn Massenkopien in eine Tabelle geladen werden. Diese Sperre muss mit dem Hinweis TABLOCK angefordert werden.

Deadlock

Bsp.:

- T1 liest Zeile 1 und setzt R Sperre
- Lesesperren sind zueinander kompatibel T2 liest Zeile 1 und setzt ebenfalls R Sperre → Beide lesen
- T1 will nun schreiben und versucht R Sperre in X Sperre zu verwandeln → Schlägt fehl da T2 noch R Sperre hält, T1 wartet nun auf die Freigabe der Sperre
- T2 will ebenfalls schreiben und versucht R Sperre in X Sperre zu verwandeln → Schlägt fehl da T1 noch R Sperre hält, T2 wartet nun auf die Freigabe der Sperre



Deadlock

Im MS-SQL Server gibt es eine interne Funktion die alle Sperren anzeigt

➤ `sp_lock`

Gliederung

6.

Transaktionsverwaltung



Transaktionen



Phänomene



Sperren



Isolationslevel



Beispiele

Isolationslevel

In ANSI SQL 92 sind folgende Isolationsebenen (Isolation Level) definiert:

- READ UNCOMMITTED
- READ COMMITTED
- REPEATABLE READ
- SERIALIZABLE

Isolationslevel

READ UNCOMMITTED

- Lässt das Lesen von allen Daten zu
- Auftretende Phänomene:
 - Dirty Reads
 - Non-repeatable reads
 - Phantom Rows
- Sehr hohe Parallelität
- vorteilhaft bei vielen nur-lese Transaktionen

Isolationslevel

READ COMMITTED

- Auftretende Phänomene:
 - Non-repeatable reads
 - Phantom Rows
- Transaktion liest nur Daten von anderen Transaktionen die mit COMMIT beendet sind.
- Sperren werden so kurz wie möglich Gehalten → hohe Parallelität

Isolationslevel

REPEATABLE READ

- Auftretende Phänomene:
 - Phantom Rows
- Transaktion hält Sperren auf alle Objekte die gelesen wurden bis zum Ende der Transaktion.
- Eingeschränkte Parallelität

Isolationslevel

SERIALIZEABLE

- Auftretende Phänomene:
 - Keine
- Ausführung der Transaktionen nahezu seriell
- Geringe Parallelität

Isolationslevel

In MS-SQL gibt es den zusätzlichen Isolationslevel Snapshot

- Bei Lesevorgängen wird ein Snapshot der zu Beginn der Transaktion konsistenten Daten gelesen.
- Verhält sich sonst wie READ COMMITTED

Isolationslevel

Auswahl des Isolationslevels

auf Verbindungsebene

➤ **SET TRANSACTION ISOLATION LEVEL <isolationlevel>**

auf Statementebene

➤ **SELECT <spalten> FROM <tabelle> WITH
(<sperrhinweis>) WHERE ...**

Auf Statementebene sind alle Isolationslevel auch Sperrhinweise. Aber es gibt noch mehr Sperrhinweise.

Isolationslevel

Zusätzliche Sperrhinweise

- HOLDLOCK - wie SERIALIZABLE
- NOLOCK - lesen ohne Berücksichtigung von Sperren
- PAGLOCK - Seiten Sperren anstatt Tabelle Sperren
- READPAST - lesen nur von nicht gesperrten Zeilen
- ROWLOCK - Zeilen Sperren anstatt Seite oder Tabelle
- TABLOCK - Tabelle Sperren anstatt Zeile oder Seite
- TABLOCKX - EXCLUSIVE Sperre auf die Tabelle
- UPDLOCK - UPDATE Sperren anstatt SHARED Sperren

Gliederung

6.

Transaktionsverwaltung



Transaktionen



Phänomene



Sperren



Isolationslevel



Beispiele

Beispiele

Tabelle: konto

- id
- kontonummer
- kontoart
- guthaben

Beispiel: Überweisung

```
>begin transaction
```

```
>update konto
```

```
  set guthaben = guthaben - 1000
```

```
  where (kontonummer = 123456);
```

```
>update konto
```

```
  set guthaben = guthaben + 1000
```

```
  where (kontonummer = 456789);
```

```
>commit transaction
```

Beispiel: Einzahlung

```
>begin transaction
>update konto
  set guthaben = guthaben + 100
  where (kontonummer = 123456);
```

```
>commit transaction
```

```
>select guthaben from konto
  where (kontonummer = 123456);
```

Beispiel: Zinsen

```
>SET TRANSACTION ISOLATION
  LEVEL READ UNCOMMITTED;
```

```
>select * from konto;
```

```
>select * from konto;
```

```
>begin transaction
```

```
>update konto
  set guthaben=guthaben*1.25
  where (kontoart='Sparbuch')
  and (guthaben > 0);
```

```
>rollback transaction;
```

Beispiel: Kontoliste

```
>select * from  
konto with (readpast);
```

```
>begin transaction  
>update konto with (rowlock)  
set guthaben=guthaben*1.025  
where (guthaben>0);
```

```
>commit transaction;
```

Beispiel: Gesamtsumme

```
>select sum(guthaben)
  from konto with (readpast);
```

```
>begin transaction
>select kontonummer, kontoart
  from konto with (updlock);
```

```
>commit transaction;
```


Beispiel: Neues Konto

```
>begin transaction
>insert into konto values
  (111222, 'Girokonto', 500);
>commit transaction;
```

```
>begin transaction
>select kontonummer
  from konto;
```

```
>select kontonummer
  from konto;
>commit transaction;
```