.NET 2 Praktikum 5

Das Praktikum muss jeder Studierende einzeln durchführen. Dieses Praktikum ist wichtig für den täglichen Einsatz von beliebigen Datenbanksystemen (aus Sicht Entwickler).

Senden Sie Ihre Antwort zur Frage 7 als pdf-Dokument an spij@zhaw.ch. Abgabetermin ist eine Woche nach Praktikumsdurchführung. Vergessen Sie nicht, die Namen aller Studierenden im PDF aufzuführen!

Abgabetermin ist eine Woche nach Praktikumsdurchführung.

1. Ziele und Motivation des Praktikums:

1. Grundverständnis der Isolationsebenen in Transaktionen im SQL-Server
2. Einfache Beherrschung von Backup und Recovery

Datenbanksysteme werden in der Mehrheit aller SW-Applikationen eingesetzt. Es ist daher für SW-Entwickler absolut zentral, zumindest die wichtigsten Mechanismen zu kennen. Ohne diese Kenntnisse ist es «normal», dass die Daten über kurz oder lang «überraschend» fehlerhaft sind! Einen Teilbereich dieser Kenntnisse sind die Locking-Mechanismen/Transaktionslogik des Datenbanksystems. In diesem Praktikum werden hierzu die wichtigsten Aspekte aufgezeigt. Am Ende des Praktikums wird dann noch das Thema Backup angesprochen – die Lebensversicherung falls Sie mit Datenbanksystemen arbeiten - ohne Backupkonzept ist wie Autofahren ohne Versicherung.

Die im Praktikum gestellten Fragen können Sie alle selbst beantworten, respektive direkt im SQL-Server durchspielen. Die entsprechenden SQL-Statements werden Ihnen zur Verfügung gestellt. Viel Spass!

2. Kurze Theorie zu Isolationslevel und Transaktionen:

Für Details zum SQL-Server siehe: https://docs.microsoft.com/en-us/sql/relational-databases/sql-server-transaction-locking-and-row-versioning-guide?view=sql-server-ver15.

Bei Datenbanken können durch mangelnde Transaktionsisolation im Wesentlichen die folgenden Probleme (Anomalien) auftreten:

1. Dirty Read:
   «Vorläufige» Änderungen an Daten einer noch nicht abgeschlossenen Transaktion werden von einer anderen Transaktion gelesen.
2. Lost Updates:
   Zwei Transaktionen modifizieren parallel denselben Datensatz und nach Ablauf dieser beiden Transaktionen wird nur die Änderung von einer von ihnen übernommen (Timing-Problem).
3. Non-Repeatable Read:
   Wiederholte Lesevorgänge derselben Daten liefern unterschiedliche Ergebnisse.
4. Phantom Read:
   Suchkriterien treffen während einer Transaktion auf unterschiedliche Datensätze zu, weil eine (während des Ablaufs dieser Transaktion laufende) andere Transaktion Datensätze hinzugefügt, entfernt oder verändert hat. D.h. beim neuerlichen Lesen erhält man anderer Datensätze.

Isolationsebenen gemäss ANSI-Standard:

Isolationsebene	Dirty Read	Non-Repeatable Read	Phantom Read	Lost Update
READ UNCOMMITTED	möglich	möglich	möglich	möglich
READ COMMITTED	verhindert	möglich	möglich	verhindert
REPEATABLE READ	verhindert	verhindert	möglich	verhindert
SERIALIZABLE	verhindert	verhindert	verhindert	verhindert

Die Implementierung der Isolation-Level unterscheidet sich unter den Datenbankherstellern deutlich. Der SQL-Server kennt folgende Level (siehe https://docs.microsoft.com/de-de/sql/t-sql/statements/set-transaction-isolation-level-transact-sql?view=sql-server-ver16):

```
SET TRANSACTION ISOLATION LEVEL
    { READ UNCOMMITTED -- mit oder ohne Snapshot-Option
    | READ COMMITTED
    | REPEATABLE READ
    | SNAPSHOT
    | SERIALIZABLE
    }
```

Im SQL-Server kann zudem der Isolation Level READ COMMITTED noch mit der Option READ_COMMITTED_SNAPSHOT ON|OFF beeinflusst werden.

Der Unterschied macht sich bei konkurrierenden Lese- und Schreiboperationen bemerkbar (nicht in den "Fehler-Effekten"). Als Hilfestellung ein Auszug der MSDN Dokumentation (https://docs.microsoft.com/de-de/sql/t-sql/statements/set-transaction-isolation-level-transact-sql?redirected-from=MSDN&view=sql-server-ver16):

- Wenn READ_COMMITTED_SNAPSHOT auf OFF gesetzt ist (Default im SQL-Server), verwendet die DB-Engine Lesesperren, um zu verhindern, dass andere Transaktionen Zeilen ändern, während die aktuelle Transaktion einen Lesevorgang ausführt. Die Lesesperren blockieren auch Anweisung, von anderen Transaktionen modifizierte Zeilen zu lesen, bis die andere Transaktion abgeschlossen ist.
- Wenn READ_COMMITTED_SNAPSHOT auf ON gesetzt ist (Default im SQL-Azure-Server), verwendet die Datenbank die "Zeilenversionierung", um jeder Anweisung einen transaktionskonsistenten Snapshot der Daten zu präsentieren, wie er zu Beginn der Anweisung vorhanden war.

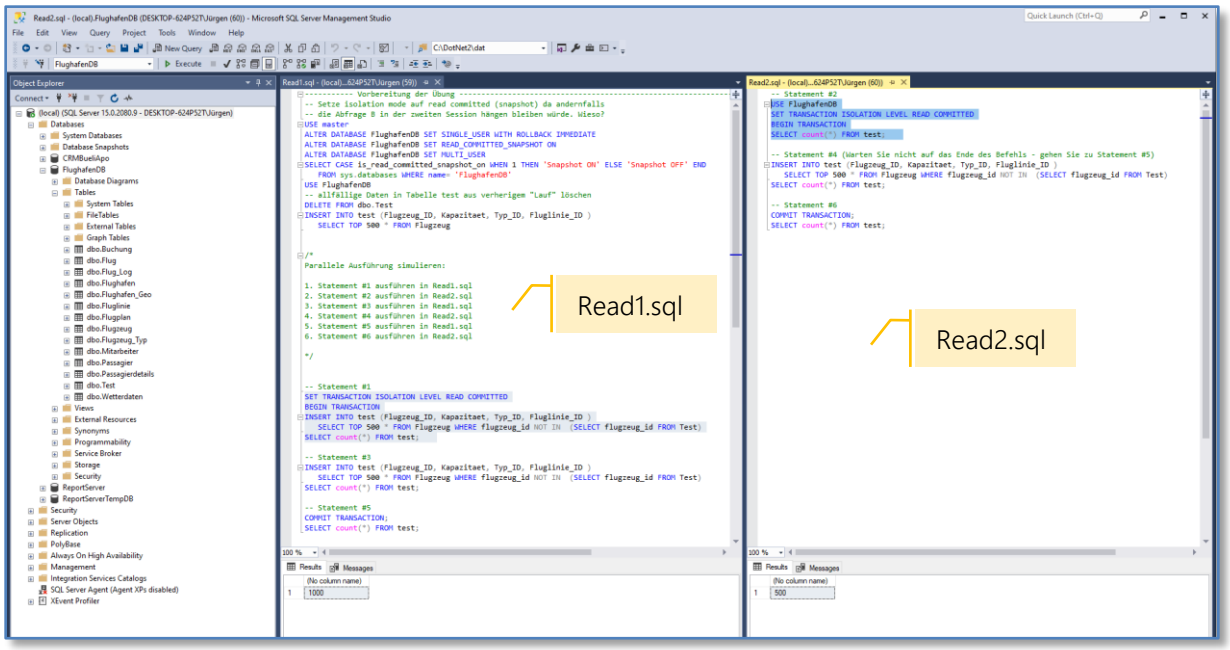
Um die Option zu aktivieren/deaktivieren können Sie die folgenden Anweisungen ausführen (wird in der 1. Aufgabe verwendet): ALTER DATABASE FlughafenDB SET READ_COMMITTED_SNAPSHOT ON. Diese Anweisungen aktivieren die SNAPSHOT-Option auf der ganzen FlughafenDB. Dazu müssen alle bestehenden Verbindungen getrennt werden (Zeilen 2/4).

Zum besseren Verständnis: Diese Isolations-Level werden mittels verschiedenartigen Locks (Sperren) implementiert. Zur Performance-Optimierung verfügt der SQL-Server über 6 verschiedene Grund-Locktypen (Exclusive, Shared, Update, Intent mit 7 Subtypen, Schema mit 2 Subtypen und Bulk-Update) in Kombination mit 11 Lock-Ressourcen (Database, Table, Page, Row, B-Tree, etc.).

3. Verhalten einer einzelnen Transaktion und konkurrenter Lesetransaktionen:

Führen Sie die T-SQL-Anweisungen gemäss `Read1.sql` und `Read2.sql` Schritt für Schritt parallel in zwei Abfragen im SSMS (Sql Server Management Studio) aus. Markieren Sie hierzu die auszuführenden

Befehle und drücken Sie F5 (resp. Execute). Achtung, der erste Block der Anweisungen (Snapshot aktivieren und Daten der Tabelle Test löschen) wird nur einmal ausgeführt.



Falls Sie Fehler machen können Sie mittels Rollback die Transaktionen zurücksetzen und die Übung noch einmal von vorne beginnen. Falls ein Statement einen Fehler erzeugt (Insert nach Statement 4), dann überspringen Sie das Statement (ignorieren).

Wieviele Sätze sehen die Abfragen 1 bis 6 unter der Voraussetzung, dass Flugzeug mehrere tausend Sätze enthält. Was sieht eine konkurrente Session 2 bei Transaktionsisolation READ COMMITTED mit der Option Snapshot jeweils zum selben Zeitpunkt?

Abfrage	Session 1	Session 2
1/2	1000	500
3/4	1500	1500
5/6	1500	1500

4. Konkurrierende Schreiboperationen und Deadlock

Die voreingestellte Isolations-Ebene unserer Übungsdatenbanken ist eigentlich READ COMMITTED (ohne Option Snapshot, wird im SQL-Script Deadlock1.sql wieder zurückgestellt) und die notwendigen Schreibsperren sind auf Satzebene (row level locking). Veranschaulichen Sie sich das Verhalten und die Problematik des Deadlocks, indem Sie die Deadlock Demo mithilfe der Skripts Deadlock1.sql und Deadlock2.sql (parallel) nachvollziehen.

5. Konkurrierende Lese- und Schreiboperationen

In einer Transaktion T_1 läuft ein Massenupdate über eine halbe Stunde. Eine konkurrente Transaktion T_2 beginnt und endet bevor T_1 endet. Angenommen in T_2 wird dieselbe Tabelle wie in T_1 abgefragt.

1. Kann bei dem Transaktionsverhalten READ COMMITTED die Transaktion T_2 stattfinden und falls ja, welchen Zustand der Daten sieht sie? Wie beeinflusst die Option Snapshot ON für READ COMMITTED das Verhalten? Testen Sie das Verhalten anhand der Statements und Anweisungen in Concurrent1.sql und Concurrent2.sql. Ja. Liest die alten Daten.
2. Angenommen in Transaktion T_2 sollen ebenfalls Datenänderungen auf dieser Tabelle stattfinden. Was geschieht in T_2, falls die Datenänderungen vor dem Commit von T_1 durchgeführt werden?

Testen Sie das Verhalten anhand der Statements und Anweisungen in Concurrent3.sql und Concurrent4.sql. `Hat kein Einfluss. Die Daten werden durch den Commit von T2 aktu`

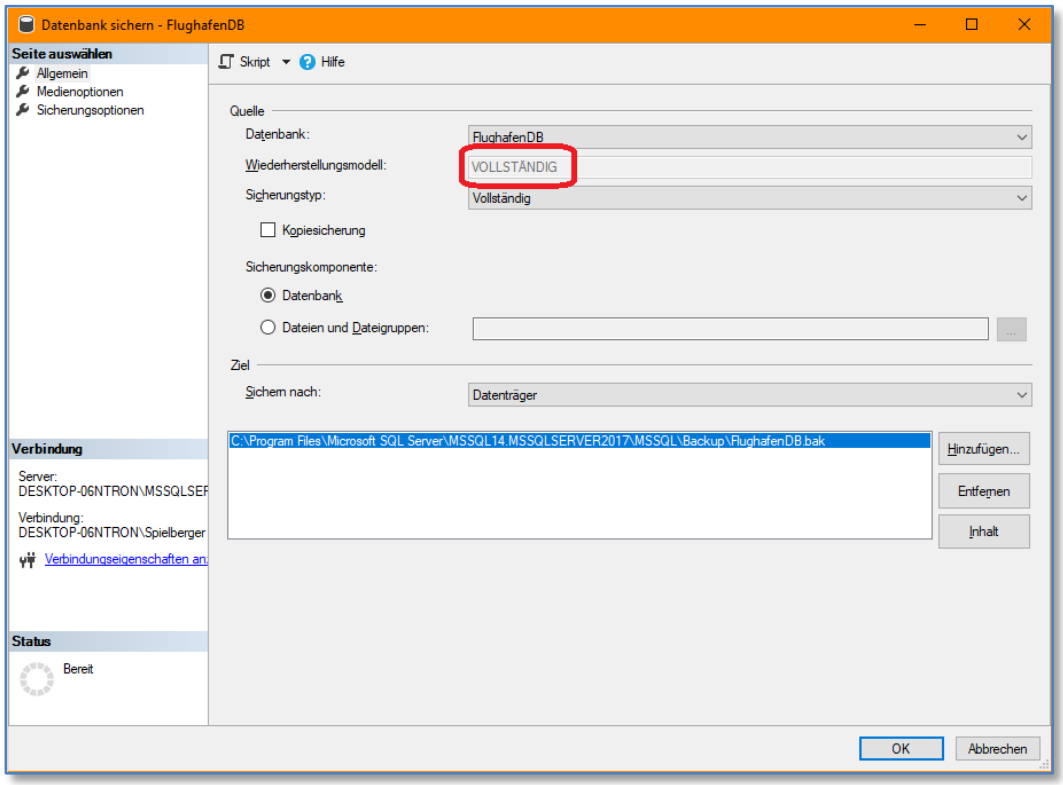
3. Diskutieren Sie die vorhergehenden Fragen falls T_2 im Transaktionsverhalten SERIALIZABLE durchgeführt wird. Testen Sie das Verhalten anhand der Statements und Anweisungen in Concurrent5.sql und Concurrent6.sql. `Abfragen könnten nicht parallel verlaufen.`

6. Backup/Restore

Das SQL Server Management Studio bietet einige nützliche Funktionen für das Erstellen und Wiederherstellen von Backups an (Objekt-Explorer -> Datenbank (FlughafenDB) -> Kontextmenü -> Tasks -> Sichern/Wiederherstellen).

Aufgaben:

a) Für ein Backup stehen verschiedene Wiederherstellungsmodelle zur Verfügung (Datenbankeigenschaft), unter anderem ‚Einfach' und ‚Vollständig'. Was ist der Unterschied zwischen diesen Modi (https://docs.microsoft.com/de-de/sql/relational-databases/backup-restore/recovery-models-sql-server?view=sql-server-ver16)? `Im einfachen Modus werden keine Protokolle gesichert`



b) Erstellen Sie ein vollständiges Backup in einen beliebigen Pfad

c) Suchen Sie die Datei, in welcher die Daten der FlughafenDB gespeichert sind (Objekt-Explorer -> Datenbank (FlughafenDB) -> Kontextmenü -> Eigenschaften -> Dateien). Löschen Sie die Datei (.mdf). Sie müssen hierzu vorgängig den SQL-Server-Dienst stoppen (MSSQLSERVER, Task Manager verwenden).

d) Starten Sie den SQL-Server wieder. Führen Sie jetzt einen Restore der FlughabenDB aus. Aktivieren Sie beim Restore die Option "Vorhandene Datenbank überschreiben (WITH REPLACE)". Der Vorgang dauert wenige Minuten. Überprüfen Sie, ob die Datei wieder vorhanden ist.

e) Erstellen Sie erneut ein vollständiges Backup, wählen Sie unter den Medienoptionen "Alle vorhandenen Sicherungssätze überschreiben" aus. Bei den nächsten Aufgaben f) bis h): lassen Sie sich **mindestens eine Minute Zeit, bis Sie zur jeweils nächsten Aufgabe gehen**.

f) Ändern Sie den Mitarbeiternamen von Mitarbeiter 1 auf Ryo2 und erstellen Sie ein weiteres Backup, diesmal aber nur vom Transaction Log (Transaktionsprotokoll):

```
UPDATE [FlughafenDB].[dbo].[Mitarbeiter]
    SET Nachname = 'Ryo2' WHERE Mitarbeiter_ID = 1
```

Ja genau, jetzt 1 Minute warten 😉.

g) Ändern Sie den Mitarbeiternamen noch einmal und erstellen Sie wiederum ein Transaction Log Backup

```
UPDATE [FlughafenDB].[dbo].[Mitarbeiter]
    SET Nachname = 'Ryo3' WHERE Mitarbeiter_ID = 1
```

h) Ändern Sie den Mitarbeiternamen ein letztes Mal, diesmal ohne ein Backup zu machen:

```
UPDATE [FlughafenDB].[dbo].[Mitarbeiter]
     SET Nachname = 'Ryo4' WHERE Mitarbeiter_ID = 1
```

i) Führen Sie eine Wiederherstellung der Datenbank aus (Objekt-Explorer -> Datenbank (FlughafenDB) -> Kontextmenü -> Tasks -> Wiederherstellen). Wählen Sie *Zeitachse* und *Bestimmtes Datum und bestimmte Uhrzeit* aus, setzen Sie das Zeitintervall auf *Stunde* und studieren Sie das angezeigte Diagramm. Wie wird der Mitarbeiter heissen, wenn Sie

   a. den letzten Zeitpunkt im hellgrünen Bereich (Tail-Log, Protokollfragment) Ryo4
   b. den letzten Zeitpunkt im dunkelgrünen Bereich (Transaction Log Backup) Ryo3

auswählen.

j) Wählen Sie einen Zeitpunkt im Tail-Log Bereich und beobachten Sie die Einträge im Wiederherstellungsplan, was hat sich geändert und was bedeutet dies? Die Änderungen seit dem letzten Backup wurden gesich
Hinweis: Sie können die Wiederherstellung der Datenbank auch als Script anzeigen. Wählen Sie dazu den Button „Skript“ oben links.

k) Erklären Sie die Bedeutung/Unterschiede zwischen:

   a. Vollständiges Backup — Backup aller vorhandenen Daten
   b. Transaction Log Backup — Backup der Transaktionen und deren Uhrzeit
   c. Tail-Log Backup — Backup der Log-Einträge seit dem letzten (Full oder Transaction Log

7. Praktikums-Abschluss

Beschreiben Sie ein x-beliebiges Szenario (z.B. «Erfassung eines Fluges, Hin- und Rückreise») und begründen Sie mit welcher Isolationsebene gemäss ANSI Sie für den von Ihnen gewählten Fall in den DB-Server-Aufrufen (diese bilden eine Transaktion) verwenden würden (Vor- und Nachteile). Antwort (max. 1'000 Buchstaben).