Transaktionen in DBS

Transaktionen: Einführung

- Wann werden Transaktionen benötigt?
  - bei Mehrnutzerbetrieb
  - Aktionen auf großen Datenbanken
- Was bedeutet „Transaktion“?
- Wie erfolgt die Steuerung von Transaktionen mit SQL?
- Übung

Transaktion

- **Definition: Transaktion**: ist eine Folge von Datenbank-Operationen (DML-Befehlen), welche die Datenbank von einem konsistenten Zustand A in einen neuen (möglicherweise geänderten) konsistenten Zustand B überführt.
- Eine Transaktion stellt einen **garantiert ununterbrechbaren** Übergang von A nach B dar, d.h. sie sind unteilbar = atomar. Es wird **alles oder nichts** ausgeführt.
- **Ergebnis einer Transaktion:** Daten sind immer konsistent, Unterbrechungen durch HW- oder SW-Fehler haben keinen Einfluß auf die Datenbankoperationen
- DBMS gewährleistet folgende Eigenschaften (Eselsbrücke = ACID):
  - **Atomicity**: ALLES oder NICHTS (= Fehlerisolierung)
  - **Consistency**: eine erfolgreiche Transaktion führt wieder zu einer konsistenten Datenbank = Gewährleistung der definierten Integritätsbedingungen, Erhaltung der logischen und physischen Datenintegrität, Führung von Änderungsprotokollen → **Logging**, Wiederherstellungsalgorithmen für Fehlerfall → **Recovery**
  - **Isolation**: alle Aktionen einer T. müssen vor parallel ablaufenden T. verborgen werden, Ablaufintegrität, kontrollierter Mehrnutzerbetrieb, Synchronisation im Allgemeinen durch Sperren → **Locking**
  - **Durability**: die Ergebnisse von **erfolgreich** absolvierten Transaktionen werden garantiert dauerhaft (= persistent) auf Datenträgern gespeichert auch beim Auftreten beliebiger anderer Fehler ✓

Probleme bei Mehrnutzerbetrieb

informativ!!!

Bei parallelem Zugriff auf dieselben Daten können **folgende Probleme** auftreten:

- **„dirty reads“**: Nutzer1 liest Daten, die von Nutzer2 noch nicht permanent gemacht wurden
- **„nonrepeatable reads“**: 2 identische Datenabfragen in **einem** Vorgang liefern unterschiedliche Ergebnisse, da ein zweiter Vorgang zwischenzeitlich Änderungen an den Daten vorgenommen hat.
- **„phantom reads“**: Nutzer1 führt Abfrage mit Bedingung C1 aus → Nutzer2 ändert Daten, die die Bedingung C1 erfüllen → nochmalige Abfrage durch Nutzer1 mit C1 führt zu anderen Ergebnissen
- **„verlorene Änderungen“/ „lost changes“**: Änderung von Daten durch Nutzer1 überschreibt Änderungen von Daten durch Nutzer2, bevor Nutzer2 ihre Änderungen permanent machen konnte ✓

Lösungen für den Mehrnutzerbetrieb

informativ!!!

- Datenbankobjekte müssen speziell vor dem Schreiben/ Speichern für den Zugriff durch andere Nutzer/ Prozesse gesperrt werden
- Sperre kann erfolgen auf folgenden Leveln:
  - Datensatz
  - Speicherseite (z.B. bei MS ist das ein 8 kB großer Bereich von Datensätzen)
  - Tabellen-Ebene
  - Datenbank-Ebene
  - siehe auch Lock Escalation: beim Überschreiten bestimmter Schwellwerte wird die jeweils übergeordnete Ebene gesperrt

- 3 Arten von Sperren:
  - **shared lock „S“ (lesender Zugriff):** Sperre zum Lesen von Daten, verbietet das Ändern/ Schreiben von Daten durch andere Transaktionen, Lesezugriff für andere T. wird nicht eingeschränkt
  - **update lock „U“:** Ankündigung eines schreibenden Zugriffs, wird automatisch in einen exclusiv lock umgewandelt, sobald alle anderen Sperren aufgehoben sind
  - **exclusiv lock „X“ (schreibender Zugriff):** Sperre zum Ändern von Daten, keine T. kann auf die Daten zugreifen, weder lesend noch schreibend ✓

Steuerung von Transaktionen

informativ!!!

- Transaktionen verwenden Sperren
- in der Praxis regelt das DBMS über die Aktivierung unterschiedlicher sogenannter „Transaktionsmodis“ den Zugriff auf die Datensätze und Tabellen → in diesen Modis werden nur ganz bestimmte Operationen erlaubt
- Synchronisation: Verwaltung von konkurrierenden Transaktionen, die von unterschiedlichen Programmen aus auf denselben Datenbestand zugreifen.
- Änderungen, die mit DML vorgenommen werden, haben den Status „vorläufig“ → sie werden mit der DCL (data control language)
  (1) endgültig ausgeführt ODER
  (2) vollständig zurückgenommen
- **eine** Transaktion ist **eine Folge** logisch zusammengehörender Datenbankoperationen (z.B. über SQL)
- Transaktionen sind unteilbar = atomar → entweder wird **alles** (commit-Operation) oder **nichts** (rollback) ausgeführt ✓

Steuerung von Transaktionen

informativ!!!

- Transaktionssteuerung u.a. über SQL, Isolationslevel
- Anweisungen für die Steuerung von Transaktionen
  - **begin of transaction (BOT)**: signalisiert den Beginn einer Transaktion → Sperrung
  - **commit transaction (commit work)**: signalisiert den erfolgreichen Abschluß einer Transaktion → Freigabe → neue Transaktionen sind mögliche = normales Ende
  - **rollback transaction (rollback work)**: Beenden einer nicht erfolgreichen Transaktion (Systemausfall, Programmfehler) und Wiederherstellung des Zustands zum Zeitpunkt des BOT
- Transaktionen:
  - **implizit:** jede einzelne SQL-Anweisung wird standardmäßig als Transaktion ausgeführt
  - **explizit:** ausdrückliche Steuerung durch SQL-Befehle (im Skript oder interaktiv)
- Transaktionen können geschachtelt werden: Transaktionen werden in umgekehrter Reihenfolge ihres Beginns abgeschlossen oder zurückgerollt
- über Systemvariable @@TRANCOUNT kann die Anzahl aktuell offener Transaktionen abgefragt werden
  Beispiel: SELECT @@TRANCOUNT
- ✓

Steuerung von Transaktionen

- Problem bei Sperren und Transaktionen: deadlock
- 2 oder mehrere aufeinander angewiesene gleichzeitige Datenbankvorgänge blockieren sich → können selbständig nicht entscheiden, wer „gewinnt“
- Beispiel: MS SQL Server erkennt und löst deadlocks automatisch, indem 1 Vorgang zum Opfer erklärt wird und zurückgerollt wird
  die abgebrochenen Transaktionen müssen dann wieder angestoßen werden
- Gegenmaßnahmen gegen deadlocks:
  - Transaktionen nur so groß wie unbedingt nötig
  - keine Anwender-Interaktionen innerhalb von Transaktionen
  - abgestimmte Aktionen auf Tabellen innerhalb von Skripts