

Transaktionsverwaltung

VL Datenbanksysteme

Ingo Feinerer

Arbeitsbereich Datenbanken und Artificial Intelligence
Institut für Informationssysteme
Technische Universität Wien

Transaktionsverwaltung

- ▶ Transaktionen: Anforderungen und Eigenschaften
- ▶ Architektur eines DBMS

Transaktionen

- ▶ Anforderungen und Eigenschaften

Transaktionsverwaltung

Beispiel

Typische Transaktion in einer Bankanwendung:

1. Lese den Kontostand von A in die Variable a : **read**(A,a);
2. Reduziere den Kontostand um 50 €: $a := a - 50$;
3. Schreibe den neuen Kontostand in die Datenbasis: **write**(A,a);
4. Lese den Kontostand von B in die Variable b : **read**(B,b);
5. Erhöhe den Kontostand um 50 €: $b := b + 50$;
6. Schreibe den neuen Kontostand in die Datenbasis: **write**(B,b);

Anforderungen an Transaktionen

Definition

Eine *Transaktion* ist eine Zusammenfassung von Datenbankoperationen, die

- ▶ ohne Beeinflussung durch andere Benutzer (andere Transaktionen)
- ▶ als Einheit fehlerfrei ausgeführt werden sollen.

Transaktionskonzept ist Grundlage für

Mehrbenutzersynchronisation: Nebenläufigkeit ist notwendig wegen Performance, diese Nebenläufigkeit darf aber nicht unkontrolliert ablaufen.

Fehlertoleranz des DBMS: Für jede Transaktion gilt „alles oder nichts“. Wenn eine Transaktion erfolgreich beendet wurde, dürfen die Änderungen auch bei Systemfehlern nicht verloren gehen.

Operationen auf Transaktionsebene

Auf jeden Fall verfügbare Operationen:

- ▶ **begin of transaction (BOT)** kennzeichnet den Beginn einer Transaktion und wird vor der ersten read/write-Operation einer Transaktion ausgeführt.
- ▶ **commit** leitet die erfolgreiche Beendigung einer Transaktion ein.
- ▶ **abort** leitet den Abbruch einer Transaktion ein. Dabei muss die DB wieder in den Zustand zurückgesetzt werden, der vor Beginn der Transaktionsausführung existierte.

Operationen auf Transaktionsebene

Zusätzliche Operationen:

- ▶ **define savepoint:**

- ▶ Definition eines Rücksetzpunktes, auf den sich die (noch aktive) Transaktion zurücksetzen lässt.
- ▶ Ein vollständiger Abbruch der Transaktion mittels **abort** ist aber immer noch möglich.

- ▶ **rollback to savepoint:**

- ▶ Zurücksetzen der aktiven Transaktion auf einen Rücksetzpunkt.
- ▶ Je nach Funktionalität des DBMS ist nur das Zurücksetzen zum zuletzt angelegten oder auch zu einem früheren Rücksetzpunkt möglich.

Abschluss einer Transaktion

Für den Abschluss einer Transaktion gibt es zwei Möglichkeiten:

1. den erfolgreichen Abschluss durch ein **commit**, und
2. den erfolglosen Abschluss durch ein **abort**:
 - ▶ entweder vom User initiiert mittels rollback Befehl,
 - ▶ oder vom DBMS initiiert aufgrund eines Fehlers.

Transaktionseigenschaften

ACID

Atomicity (Atomarität)

- ▶ Transaktion ist kleinste, nicht weiter zerlegbare Einheit
- ▶ alles oder nichts

Consistency

- ▶ Eine Transaktion führt die DB von einem konsistenten Zustand in einen konsistenten Zustand über.
- ▶ D.h. am Ende einer Transaktion müssen alle Konsistenzbedingungen laut Datenbankschema erfüllt sein.

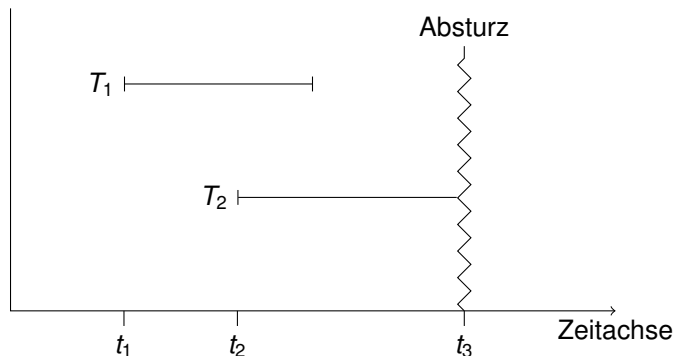
Isolation

- ▶ Nebenläufige Transaktionen dürfen sich nicht beeinflussen

Duration

- ▶ Änderungen erfolgreicher Transaktionen dürfen nicht mehr verloren gehen (auch bei HW/SW-Systemfehlern)

Atomicity and Durability



- ▶ Transaktion T_1 : Sie muss in der DB auch nach dem Wiederanlauf des Systems vorhanden sein.
- ▶ Transaktion T_2 : Sämtliche Datenbankänderungen durch T_2 müssen nach dem Wiederanlauf aus der DB entfernt sein.

Transaktionsverwaltung in SQL

begin of transaction

- ▶ kein expliziter Befehl
- ▶ BOT implizit bei der ersten Anweisung

Befehl **commit work** (oder einfach **commit**):

- ▶ Die in der Transaktion vollzogenen Änderungen werden, sofern keine Konsistenzverletzung oder andere Probleme aufgedeckt werden, festgeschrieben.

Befehl **rollback work** (oder einfach **rollback**):

- ▶ Alle Änderungen dieser Transaktion werden zurückgesetzt.
- ▶ DBMS muss die „erfolgreiche“ Ausführung eines rollback Befehls immer garantieren können (anders als commit).

Transaktionsverwaltung in SQL

savepoint *sp_name*:

- ▶ definiert einen Rücksetzpunkt mit dem Namen *sp_name* innerhalb der laufenden Transaktion.
- ▶ Damit ist ein Rücksetzen zu diesem Punkt möglich (ohne die ganze Transaktion zurückzurollen)
- ▶ Savepoints sollte man „sparsam“ einsetzen. Es ist besser, lange Transaktionen in kleinere Transaktionen zu zerlegen.

rollback to *sp_name*:

- ▶ Macht alle Datenbankänderungen innerhalb dieser Transaktion seit dem Savepoint *sp_name* rückgängig.

Transaktionsverwaltung in SQL

Beispiel

```
INSERT INTO Vorlesungen  
VALUES (5275, 'Kernphysik', 3, 2141)  
INSERT INTO Professoren  
VALUES (2141, 'Meitner', 'C4', 205)  
COMMIT WORK
```

Implizites Transaktionsende

Implizites commit

- ▶ Nach jedem DML Kommando, falls AUTOCOMMIT ON (Standard in PostgreSQL)
- ▶ Bei jedem DDL Kommando (z.B. **CREATE TABLE, ...**) und DCL Kommando (z.B. **GRANT, ...**) (Standard in Oracle)

Implizites rollback:

- ▶ Bei Systemabsturz, bei Verbindungsabbruch, etc.

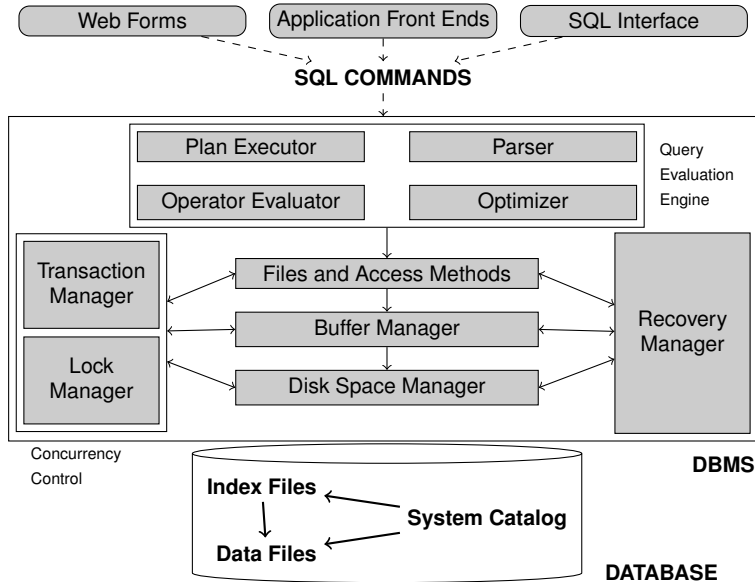
Guter Stil:

- ▶ (Wenn möglich) Transaktionen *explizit* mittels commit bzw. mittels rollback beenden.

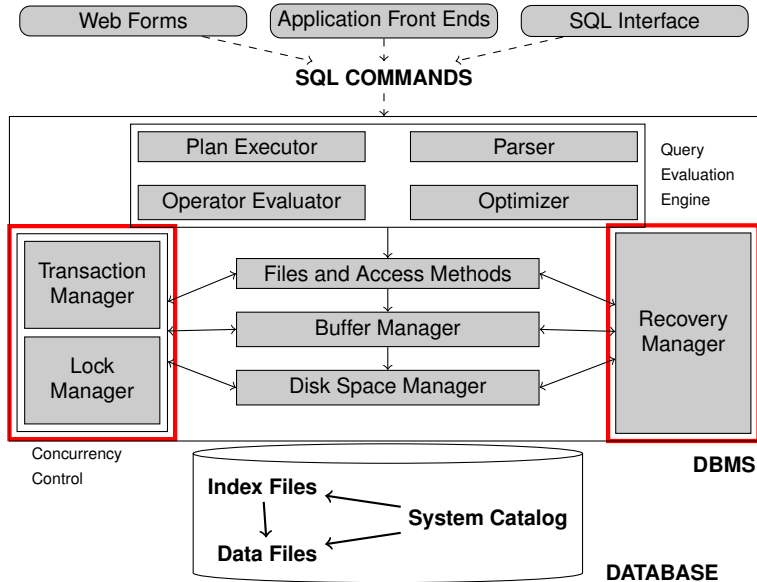
Architektur eines DBMS

- ▶ Softwarekomponenten der Fehlerbehandlung und Mehrbenutzersynchronisation

Architektur eines DBMS



Architektur eines DBMS



Concurrency Control

Lock Manager:

- ▶ Verwaltet die Sperranforderungen auf Datenbankobjekte (Tupel, Seite, ...).
- ▶ Erfüllt Sperranforderungen für Datenbankobjekte, sobald diese verfügbar sind.

Transaction Manager:

- ▶ Steuert die Abarbeitung der Transaktionen.

Recovery Manager

Im laufenden Betrieb:

- ▶ Verwaltung des log files

Beim Wiederanlauf nach einem Systemausfall:

- ▶ Redo aller Operationen von erfolgreich abgeschlossenen Transaktionen
- ▶ Undo aller Operationen von nicht abgeschlossenen Transaktionen