Kapitel 12

Transaktionsverwaltung

12.1 Begriffe

Unter einer $Transaktion$ versteht man die Bündelung mehrerer Datenbankoperationen zu einer Einheit. Verwendet werden Transaktionen im Zusammenhang mit

- **Mehrbenutzersynchronisation** (Koordinierung von mehreren Benutzerprozessen),
- **Recovery** (Behebung von Fehlersituationen).

Die Folge der Operationen (lesen, ändern, einfügen, löschen) soll die Datenbank von einem konsistenten Zustand in einen anderen konsistenten Zustand überführen.

Als Beispiel betrachten wir die Überweisung von 50,-DM von Konto A nach Konto B:

read(A, a);
a := a - 50;
write(A, a);
read(B, b);
b := b + 50;
write(B, b);

Offenbar sollen entweder alle oder keine Befehle der Transaktion ausgeführt werden.

12.2 Operationen auf Transaktionsebene

Zur Steuerung der Transaktionsverwaltung sind folgende Operationen notwendig:

- **begin of transaction (BOT):** Markiert den Anfang einer Transaktion.
- **commit:** Markiert das Ende einer Transaktion. Alle Änderungen seit dem letzten BOT werden festgeschrieben.

- **abort:** Markiert den Abbruch einer Transaktion. Die Datenbasis wird in den Zustand vor Beginn der Transaktion zurückgeführt.
- **define savepoint:** Markiert einen zusätzlichen Sicherungspunkt.
- **backup transaction:** Setzt die Datenbasis auf den jüngsten Sicherungspunkt zurück.

12.3 Abschluß einer Transaktion

Der erfolgreiche Abschluß einer Transaktion erfolgt durch eine Sequenz der Form

$$BOT\ \ op_1;\ op_2;\ \ \ldots\ ; op_n;\ \ commit$$

Der erfolglose Abschluß einer Transaktion erfolgt entweder durch eine Sequenz der Form

$$BOT\ \ op_1;\ op_2;\ \ \ldots\ ; op_j;\ \ abort$$

oder durch das Auftreten eines Fehlers

$$BOT\ \ op_1;\ op_2;\ \ \ldots\ ; op_k;\ \ \ < Fehler >$$

In diesen Fällen muß der Transaktionsverwalter auf den Anfang der Transaktion zurücksetzen.

12.4 Eigenschaften von Transaktionen

Die Eigenschaften des Transaktionskonzepts werden unter der Abkürzung $ACID$ zusammengefaßt:

- **Atomicity:** Eine Transaktion stellt eine nicht weiter zerlegbare Einheit dar mit dem Prinzip *alles-oder-nichts*.
- **Consistency:** Nach Abschluß der Transaktion liegt wieder ein konsistenter Zustand vor, während der Transaktion sind Inkonsistenzen erlaubt.
- **Isolation:** Nebenläufig ausgeführte Transaktionen dürfen sich nicht beeinflussen, d. h. jede Transaktion hat den Effekt, den sie verursacht hätte, als wäre sie allein im System.
- **Durability:** Die Wirkung einer erfolgreich abgeschlossenen Transaktion bleibt dauerhaft in der Datenbank (auch nach einem späteren Systemfehler).

12.5 Transaktionsverwaltung in SQL

In SQL-92 werden Transaktionen implizit begonnen mit Ausführung der ersten Anweisung. Eine Transaktion wird abgeschlossen durch

- **commit work:** Alle Änderungen sollen festgeschrieben werden (ggf. nicht möglich wegen Konsistenzverletzungen).

- **rollback work:** Alle Änderungen sollen zurückgesetzt werden (ist immer möglich).

Innerhalb einer Transaktion sind Inkonsistenzen erlaubt. Im folgenden Beispiel fehlt vorübergehend der Professoreneintrag zur Vorlesung:

```
insert into Vorlesungen
values (5275, 'Kernphysik', 3, 2141);
insert into Professoren
values (2141, 'Meitner', 'C4', 205);
commit work;
```

12.6 Zustandsübergänge einer Transaktion

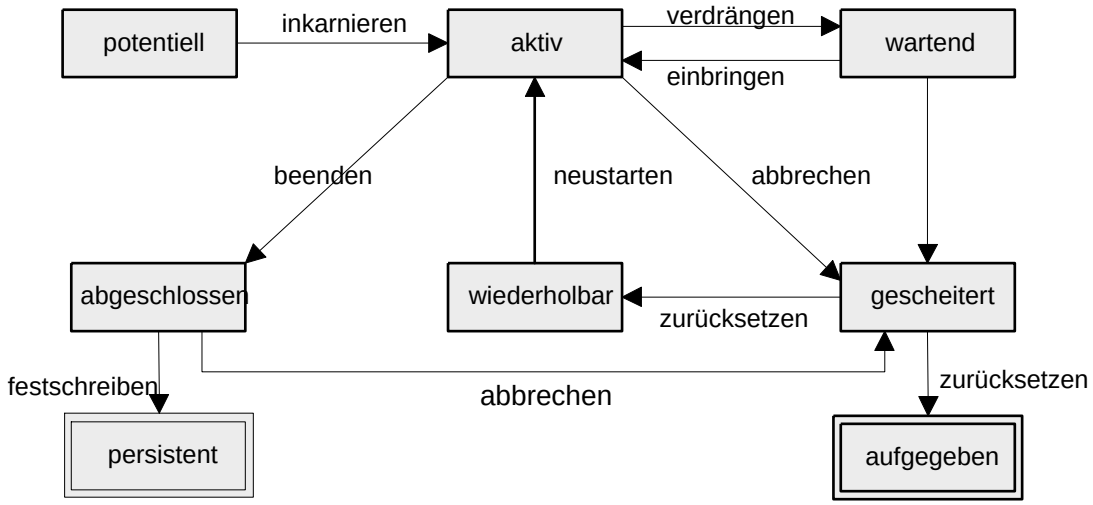

Abbildung 12.1: Zustandsübergangsdiagramm für Transaktionen

Abbildung 12.1 zeigt die möglichen Übergänge zwischen den Zuständen:

- **potentiell:** Die Transaktion ist codiert und wartet auf ihren Einsatz.
- **aktiv:** Die Transaktion arbeitet.
- **wartend:** Die Transaktion wurde vorübergehend angehalten
- **abgeschlossen:** Die Transaktion wurde durch einen commit-Befehl beendet.
- **persistent:** Die Wirkung einer abgeschlossenen Transaktion wird dauerhaft gemacht.
- **gescheitert:** Die Transaktion ist wegen eines Systemfehlers oder durch einen abort-Befehl abgebrochen worden.
- **wiederholbar:** Die Transaktion wird zur erneuten Ausführung vorgesehen.
- **aufgegeben:** Die Transaktion wird als nicht durchführbar eingestuft.

12.7 Transaktionsverwaltung beim SQL-Server 2000

Listing 12.1 zeigt ein Beispiel für den Einsatz einer Transaktion. Durch das explizite `begin transaction` sind nach dem `insert` solange andere Transaktionen blockiert, bis durch das explizite `commit transaction` die Transaktion abgeschlossen wird.

```
begin transaction
insert into professoren
values(55555,'Erika','C4',333,1950-12-24)
select * from professoren where name='Erika'
commit transaction
```

Listing 12.1: Beispiel für Commit

Listing 12.2 zeigt ein Beispiel für den die Möglichkeit, die Auswirkungen einer Transaktion zurückzunehmen. Der zweite `select`-Befehl wird den Studenten mit Namen Fichte nicht auflisten. Andere Transaktionen sind blockiert. Nach dem `Rollback`-Befehl taucht der Student Fichte wieder auf.

```
begin transaction
select * from studenten where name like 'F%'
delete   from studenten where name='Fichte'
select * from studenten where name like 'F%'
rollback transaction
```

Listing 12.2: Beispiel für Rollback