SQL – Transaktionen

Stephan Karrer

Transaktionskonzept bei DBMS

- Eine Transaktion umfasst eine oder mehrere Anweisungen für die gilt:

Atomic	Entweder alle Anweisungen sind erfolgreich oder keine
Consistent	Eine erfolgreiche Transaktion führt den Datenbestand in einen konsistenten Zustand (semantischer Begriff!).
Isolated	Die Zwischenzustände des Datenbestands während einer Transaktion sind für parallel laufende Zugriffe nicht sichtbar.
Durable	Die Ergebnisse einer erfolgreichen Transaktion werden in der Datenbank persistiert.

- ANSI fordert: Alle schreibenden Zugriffe müssen innerhalb einer Transaktion erfolgen.
  - ➢ dies betrifft auf jeden Fall alle DML-Anweisungen.

- Bei allen Systemen gilt:
  Eine einzelne SQL-Anweisung ist auf jeden Fall transaktional.

Umsetzung des Transaktionskonzepts

- Da die Anweisungsfolge innerhalb einer Transaktion anforderungsspezifisch ist:

  COMMIT für die explizite erfolgreiche Beendigung

  ROLLBACK für den Abbruch (und damit rückgängig machen aller Änderungen)

- Einige Systeme, z.B. Oracle, DB2 benutzen implizite Transaktionssteuerung:

  Eine neue Transaktion startet automatisch, wenn die vorherige Transaktion explizit oder implizit durch das System beendet wird und umfasst jetzt alle folgenden Anweisungen.

- Andere Systeme nutzen „AutoCommit“-Modus:

  Standardmäßig ist nur eine einzelne Anweisung eine Transaktion. Sollen mehrere Anweisungen in einer Transaktionsklammer ausgeführt werden, so muss diese explizit gestartet werden:

  - START TRANSACTION (ANSI) , herstellerspezifische Anweisungen sind auch üblich!

Transaktionsteuerung am Bsp. PostgreSQL

Default: AutoCommit-Modus

```
-- Ende der letzten Transaktion
BEGIN;
INSERT INTO departments
       VALUES (280, 'Recreation', 110, 1700);
UPDATE employees SET salary = 10 WHERE employee_id = 100;
SELECT * FROM employees WHERE employee_id =100;
ROLLBACK;
SELECT * FROM employees WHERE employee_id =100;

-- jetzt sind wir wieder im AutoCommit-Modus
-- nächste Transaktion beginnt
BEGIN;
UPDATE employees SET salary = 48000 WHERE employee_id = 100;
SAVEPOINT punkt1;
UPDATE employees SET salary = 200 WHERE employee_id = 100;
SELECT * FROM employees WHERE employee_id =100;
ROLLBACK TO SAVEPOINT punkt1;
SELECT * FROM employees WHERE employee_id =100;
COMMIT; -- comm = 100
SELECT * FROM employees WHERE employee_id =100;
-- jetzt sind wir wieder im AutoCommit-Modus
UPDATE employees SET salary = 24000 WHERE employee_id = 100;
```

Transaktionssteuerung bei SQL Server

```
BEGIN TRANSACTION;
   -- Aktionen (DML oder DDL)
ROLLBACK;
-- Aktionen (AutoCommit-Modus)
BEGIN TRAN t1;
   -- Aktionen (DML oder DDL)
COMMIT t1;
```

- SQLServer verwendet standardmäßig AutoCommit-Modus:
  Jede SQL-Anweisung ist eine Transaktion, COMMIT erfolgt automatisch bei Erfolg
- Sollen mehrere Anweisungen in einer Transaktion zusammengefasst werden,
  kann man explizite Transaktionen verwenden
- Es gibt auch einen einstellbaren impliziten Transaktionsmodus, in dem DML- und DDL-Anweisungen automatisch eine neue Transaktion starten
  (Entspricht dem Standard von Oracle bzw. DB2)

Transaktionsteuerung am Bsp. Oracle

```
-- Ende der letzten Transaktion
-- implizit bei Oracle durch jede DDL- oder DCL-Anweisung
-- egal ob erfolgreich oder nicht!

INSERT INTO departments
       VALUES (280, 'Recreation', DEFAULT, 1700);
UPDATE emp SET sal = 10;
ROLLBACK;
-- nächste Transaktion beginnt
UPDATE emp SET comm = 100;
SAVEPOINT punkt1;
UPDATE emp SET sal = sal * 1.1;
ROLLBACK TO SAVEPOINT punkt1;
COMMIT; -- comm = 100
```

- Innerhalb einer Transaktion können Zwischenpunkte gesetzt werden, auf deren Zustand ein lokaler Rollback erfolgen kann, ohne die gesamte Transaktion zurückzusetzen
- Entscheidend ist allerdings, wie die umfassende Transaktion beendet wird (COMMIT oder ROLLBACK). Ein Commit bzgl. SAVEPOINT hat keine Auswirkungen.

SavePoints bei SQL Server

```
BEGIN TRANSACTION;
   -- Aktionen (DML oder DDL)
   SAVE TRANSACTION P1
         -- Aktionen
   ROLLBACK TRANSACTION P1
   -- Aktionen
COMMIT;
```

- Üblicherweise lassen sich bei allen heutigen Systemen Savepoints setzen, nur die Syntax ist immer ein wenig Hersteller-spezifisch.

Isolations-Stufe setzen (am Beispiel SQL Server)

```
SET TRANSACTION ISOLATION LEVEL
       REPEATABLE READ;

BEGIN TRANSACTION;

   -- Aktionen (DML oder DDL)

COMMIT;

SET TRANSACTION ISOLATION LEVEL
       READ COMMITTED;
```

- Üblicherweise wird die Isolation der Transaktionen untereinander aufgeweicht, um die parallele Nutzung der Datenbank zu erhöhen.
- ANSI definiert hier 4 Isolations-Stufen:
  Üblicherweise (so auch bei SQL Server) ist der Default „READ COMMITED“.
- Das lässt sich allerdings bei den heute üblichen Systemen für die jeweilige Transaktion auch anders einstellen.