Leibniz Universität IT Services

Leibniz Universität Hannover

S(tructured)Q(uery)L(anguage) Transaction Control Language

Transaction Control Language

- Logischer Ablauf von SQL-Anweisungen.
- Zusammenfassung von SQL-Anweisungen zu einer Transaktion.

Transaktion

- Zusammenfassung von aufeinander aufbauenden SQL-Anweisung.
- Kapselung von SQL-Anweisungen.
- Ausführung von `INSERT`-, `DELETE`- und `UPDATE`-Anweisungen. Änderungen an den Datensätzen werden in der Datenbank gespeichert oder zurückgenommen.

… ist atomar

- Eine Transaktion wird komplett durchgeführt oder komplett rückgängig gemacht.
- Hinweis: Mit Hilfe von Wiederherstellungspunkten kann die Atomarität durchbrochen werden.

… werden isoliert

- SQL-Anweisungen in Transaktionen werden nicht durch Anweisungen in anderen Transaktionen beeinflusst.
- Abweichungen von dieser Regel sind möglich.

Dauerhaftigkeit

- Bis zum Abschluss einer Transaktion können alle darin enthaltenen SQL-Anweisungen rückgängig gemacht werden.
- Mit Abschluss der Transaktion durch `COMMIT` werden die ausgelösten Änderungen dauerhaft in der Datenbank gespeichert.

Konsistenz der Datenbank

- Eine Datenbank, die vorher sich in einem konsistenten Zustand befunden hat, besitzt nach Durchführung der Transaktion auch einen korrekt gespeicherten Datenbestand.
- Bei Verstößen gegen die Konsistenz der Datenbank, wird die Transaktion nicht ausgeführt.

Beispiel

```
BEGIN TRANSACTION;

CREATE TABLE IF NOT EXISTS tblCity(
   idCity INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
   postalcode TEXT,
   city TEXT
);

INSERT INTO tblCity(postalcode, city)
   SELECT DISTINCT PostalCode, City FROM customers
   WHERE ((City is not null)
   AND (city <> '' )) ORDER BY City;

COMMIT
```

Beginn der Transaktion

```
BEGIN TRANSACTION;
```

- `BEGIN` oder `BEGIN TRANSACTION` startet die Transaktion.
- Jede nachfolgende SQL-Anweisung kann wieder rückgängig gemacht werden.

Speicherung der Transaktionen

```
COMMIT;
```

- `COMMIT` oder `COMMIT TRANSACTION` beendet die Transaktion.
- Wenn alle Anweisungen fehlerfrei ausgeführt wurden, werden die Änderungen sichtbar in der Datenbank angezeigt.

Transaktion rückgängig machen

```
ROLLBACK TRANSACTION;
```

- `ROLLBACK` oder `ROLLBACK TRANSACTION` beendet die Transaktion.
- Alle Transaktionen, auch wenn sie fehlerfrei durchgeführt wurden, werden rückgängig gemacht.

Wiederherstellungspunkte nutzen

```
BEGIN TRANSACTION;

CREATE TABLE IF NOT EXISTS Quartalszahlen(
      id INTEGER PRIMARY KEY AUTOINCREMENT,

);

SAVEPOINT AfterCreate;

INSERT INTO Quartalszahlen(produktname,
gesamtbestellmenge, quartal, jahr)

ROLLBACK TO AfterCreate;
```

Setzen eines Wiederherstellungspunktes

```
SAVEPOINT AfterCreate;
```

- `SAVEPOINT [name].`
- Innerhalb der Transaktion ist der Name des Wiederherstellungspunktes eindeutig.
- Ein Wiederherstellungspunkt kann nach Abschluss einer SQL-Anweisung gesetzt werden.

… und rückgängig machen

```
ROLLBACK TO AfterCreate;
```

- `ROLLBACK TO [wiederherstellungspunkt].`
- Der Wiederherstellungspunkt ist in den Transaktionen definiert.
- Alle Aktionen bis zu diesem Wiederherstellungspunkt werden rückgängig gemacht.

Verletzung von Constraints

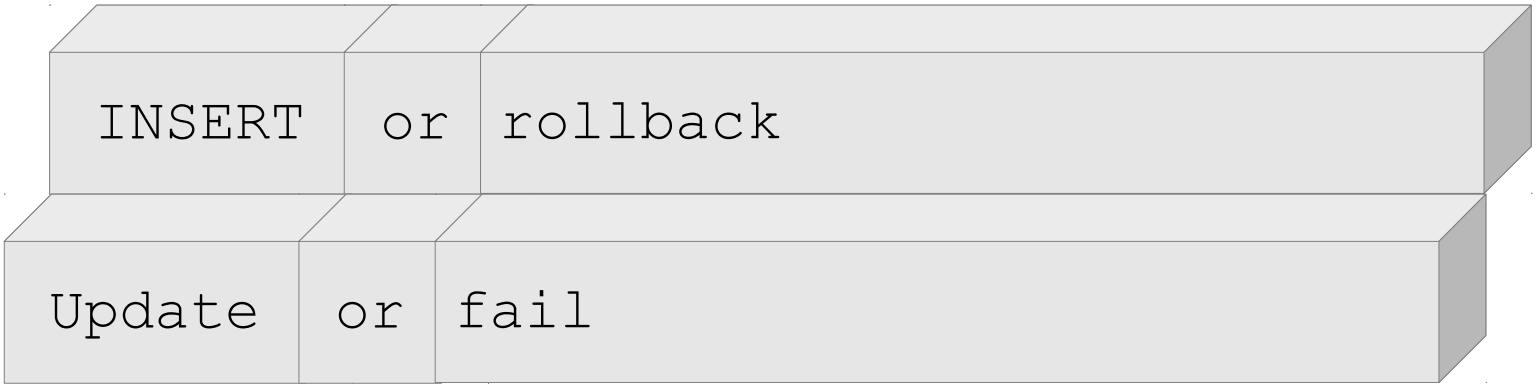
- Falls die Transaktion ein Constraint verletzt, wird die Transaktion abgebrochen.
- In SQLite können Verletzungen von Constraint gesteuert behandelt werden.

Beispiel

```
INSERT  or rollback INTO invoice_items
(InvoiceID, TrackID, UnitPrice, Quantity)
VALUES(
(SELECT MAX(InvoiceId) FROM invoices),3503,
(SELECT UnitPrice FROM tracks
WHERE (TrackID == 3503)),
2);

UPDATE or fail invoices
Set Total = (SELECT (SUM(UnitPrice * Quantity))
    FROM invoice_items
    WHERE (InvoiceID == (SELECT MAX(INVOICEId)
    FROM invoices)))
WHERE (InvoiceId == (SELECT MAX(InvoiceId)
       FROM invoices));
```

Resolutionen

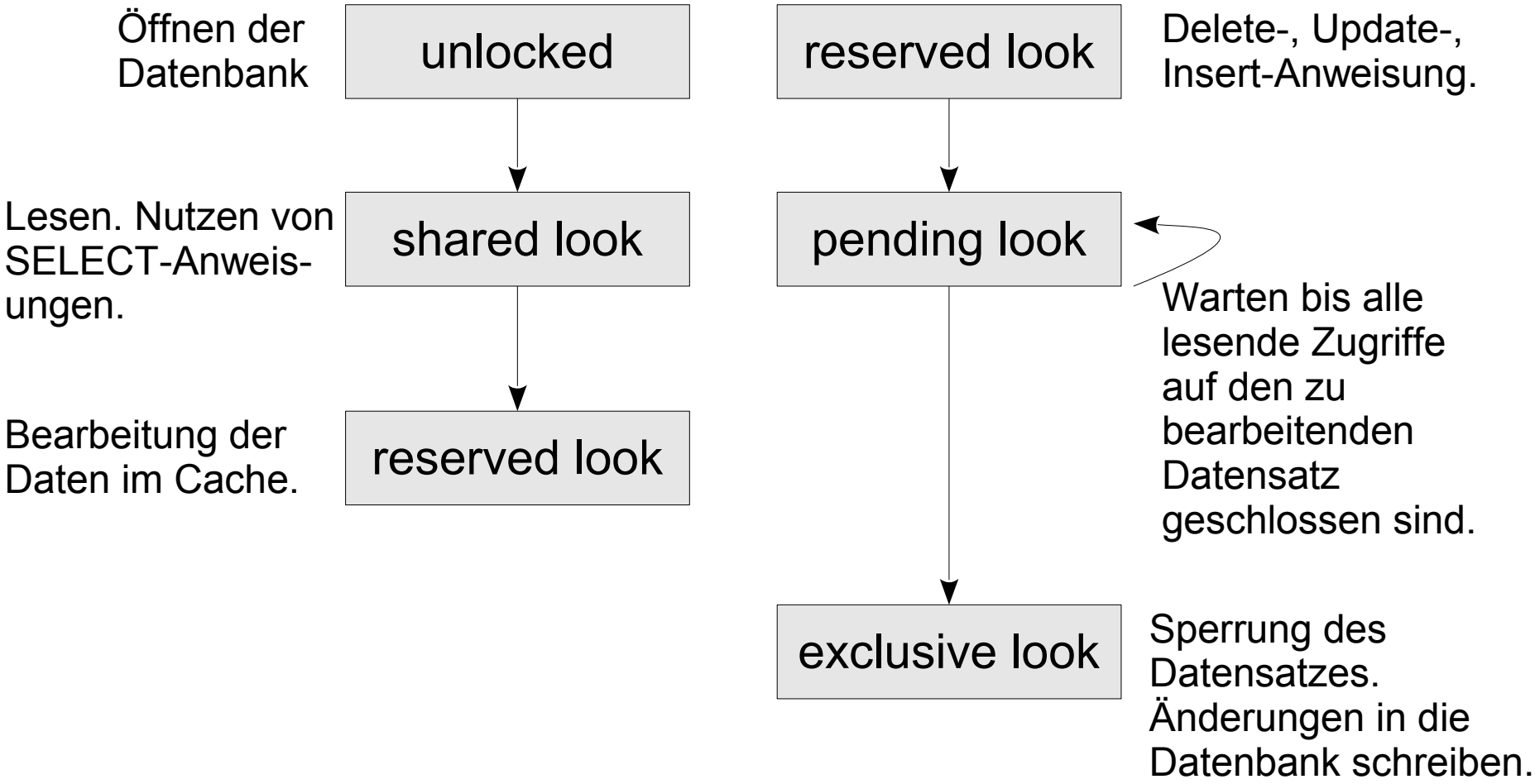


- Die Anweisungen `CREATE TABLE`, `INSERT` oder `UPDATE` können mit einer Resolution durch den Operator `or` verbunden werden.
- Entweder wird die Aktionsabfrage ausgeführt oder wenn ein Fehler vorliegt, wird entsprechend der Resolution verfahren.

Möglichkeiten

- `ignore`. Der Fehler wird ignoriert.
- `fail`. Die aktuelle SQL-Anweisung wird abgebrochen. Alle Änderungen der Anweisung vor Auftreten des Fehlers bleiben erhalten.
- `abort`. Die aktuelle SQL Anweisung wird abgebrochen. Alle Änderungen durch die Anweisung werden rückgängig gemacht Die Resolution implementiert den SQL-Standard.
- `rollback`. Die aktuelle SQL-Anweisung wird abgebrochen. Die dazugehörige Transaktion wird abgebrochen. Alle Änderungen der Transaktion werden rückgängig gemacht.
- `replace` bezieht sich nur auf unique-Constraints.

Zustand einer Datenbank



unlocked

- Keinerlei Sperrung der Datenbank.
- Keine Schreib- und Lesezugriffe.
- Standard-Zustand.

Shared locked

- Beliebig viele Lesezugriff auf die Datenbank.
- Lesen von Datensätzen mit Hilfe von Auswahlabfragen (`SELET … FROM`)
- Keine Schreibzugriffe auf die Datenbank.

Reserved locked

- Die Datenbank plant einen Schreibzugriff.
- Ausführung einer `Insert`-, `Delete`- oder `Update`-Anweisung.
- Datensätze werden im Cache verändert, aber nicht direkt in der Datenbank.
- Vor weiteren schreibenden Zugriffen werden die zu ändernden Datensätze geschützt. Lesende Zugriff sind weiter möglich.

Pending locked

- Die Datenbank will schreibend zugreifen.
- Ein lesender Zugriff ist nicht mehr möglich. Der Prozess wartet bis alle momentan vorhandenen lesenden Zugriffe abgeschlossen sind.

Exclusive locked

- Die geänderten Datensätze werden in die Datenbank geschrieben.
- Ein lesender oder schreibender Zugriff von anderen Prozessen ist nicht möglich.

Zustand zu Beginn einer Transaktion

```
BEGIN IMMEDIATE TRANSACTION;

INSERT INTO invoices(CustomerID, InvoiceDate,
 BillingAddress, BillingCitty, BillingState,
 BillingCountry, BillingPostalCode)
VALUES(
 6,
 datetime('now'),
 SELECT Address FROM customers WHERE (CustomerId == 6),
 SELECT City FROM customers WHERE (CustomerId == 6),
 SELECT State FROM customers WHERE (CustomerId == 6),
 SELECT Country FROM customers WHERE (CustomerId == 6),
 SELECT PostalCode FROM customers WHERE (CustomerId == 6)
)
```

Einstellung von „Unlocked“

```
BEGIN DEFERRED TRANSACTION;
BEGIN TRANSACTION;
```

- Standard-Zustand.
- Keine Sperrung vorhanden.
- Sobald in der Transaktion ein lesender Zugriff erfolgt, wird die Datenbank in den Zustand „shared lock“ versetzt.

Einstellung von „Reserved lock“

```
BEGIN IMMEDIATE TRANSACTION;
```

- Kein andere Prozess kann schreibend auf die Datenbank zugreifen.
- Die Datenbank ist für schreibende Zugriffe gesperrt.
- Lesende Zugriffe sind möglich.
- Der Befehl `COMMIT` kann durchgeführt werden, wenn alle lesenden Zugriffe abgeschlossen sind.

Einstellung von „Exclusive lock“

```
BEGIN EXCLUSIVE TRANSACTION;
```

- Kein andere Prozess kann schreibend oder lesend auf die Datenbank zugreifen.