Datenbanken & Informationssysteme
Übungen Teil 2

Transaktionen und Synchronisationskontrolle

1. Dauer serieller Transaktionen

100 Kunden einer Bank wollen gleichzeitig an 100 verschiedenen Geldautomaten Geld abheben. Nehmen wir an, dass die Transaktion auf dem Rechner der Bank 0,3 Sekunden dauert und dass die Transaktionen *seriell* durchgeführt werden.

Wie lange muss der Kunde warten, dessen Abhebung als letzte bearbeitet wird? Wie lange ist die durchschnittliche Wartezeit des Systems?

2. Serialisierbarkeit

Geben Sie an, welche der folgenden Abläufe serialisierbar sind – mit Begründung

(a) $r_1(x); r_2(y); r_1(z); r_3(z); r_2(x); r_1(y);$

(b) $r_1(x); w_2(y); r_1(z); r_3(z); w_2(x); r_1(y);$

(c) $r_1(x); w_2(y); r_1(z); r_3(z); w_1(x); r_2(y);$

(d) $r_1(x); r_2(y); r_1(z); r_3(z); w_1(x); w_2(y);$

(e) $r_1(x); r_2(y); w_2(x); w_3(x); w_3(y); r_1(y);$

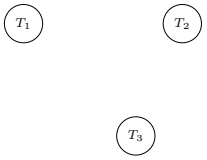
(f) $w_1(x); r_2(y); r_3(z); r_1(x); w_2(y);$

(g) $r_1(z); w_2(x); r_2(y); w_1(x); w_3(z); w_1(y); r_3(x);$

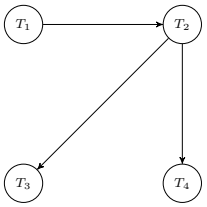
3. Serielle Abläufe zum Präzedenzgraph finden

Aus Abläufen von Aktionen wurde der Präzedenzgraph ermittelt, wie er in den folgenden Aufgaben vorgegeben ist. Die Aufgabe besteht darin, alle *seriellen* Abläufe anzugeben, zu denen der Ablauf konflikt-äquivalent ist, von dem der jeweilige Präzedenzgraph stammt.

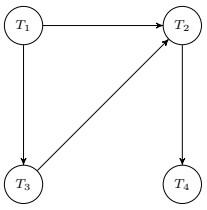
(a)



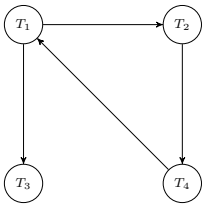
(b)



(c)



(d)



4. 2PL und Serialisierbarkeit

Beweisen Sie folgende Aussage:

Gegeben sei ein Ablauf, der dem 2PL-Protokoll entspricht. Dann ist ein äquivalenter serieller Ablauf derjenige, bei dem die Transaktionen in der Reihenfolge ausgeführt werden, wie sie im gegebenen Ablauf den ersten Lock freigeben.

5. Beispiele für Abläufe – 2PL?

Geben Sie bei den folgenden Abläufen an, ob sie dem 2-Phasen-Lock-Protokoll genügen oder sogar dem strikten 2PL. (b_i steht für den Beginn der Transaktion T_1, c_i für Commit von T_i.)

(a) $b_1; l_1(x); r_1(x); r_1(y); l_1(y); l_1(z); w_1(x); w_1(y); u_1(z); u_1(x); c_1;$

❏ nicht 2PL ❏ 2PL, nicht strikt ❏ striktes 2PL

(b) $b_1; l_1(x); r_1(x); l_1(y); b_2; l_2(z); w_1(x); w_1(y); w_2(z); u_1(y); r_1(x);$
$u_1(x); u_2(z); c_2; c_1;$

❏ nicht 2PL ❏ 2PL, nicht strikt ❏ striktes 2PL

(c) $b_1; b_2; l_1(y); w_1(y); l_2(x); r_1(x); u_1(y); u_2(x); c_1; c_2;$

❏ nicht 2PL ❏ 2PL, nicht strikt ❏ striktes 2PL

(d) $b_1; l_1(x); l_1(y); l_1(z); r_1(x); u_1(x); w_1(y); u_1(y); r_1(z); u_1(z); c_1;$

❑ nicht 2PL ❑ 2PL, nicht strikt ❑ striktes 2PL

(e) $b_2; b_3; b_1; l_1(x); r_1(x); l_2(y); r_2(y); w_1(x); u_1(x); c_1; l_2(z); w_2(y); w_2(z); l_3(x); w_3(x); u_3(x); u_2(z); u_2(y); c_2; c_3$

❑ nicht 2PL ❑ 2PL, nicht strikt ❑ striktes 2PL

6. 2PL spielen

Gegeben sei folgender Ablauf:

$b_1; r_1(x); b_2; r_2(y); b_3; r_3(z); w_1(z); b_4; w_4(y); w_3(x); c_2; c_4; \ldots$

Dabei steht b_i für den Beginn von Transaktion T_i und c_i für Commit der Transaktion T_i.

Sie spielen nun das Datenbankmanagementsystem, das diesen Ablauf im *strikten 2-Phasen-Lock-Protokoll* mit *binären* Sperren abwickelt.

(a) Geben Sie für jeden Schritt an, welche Transaktionen aktiv sind, welche Transaktion welche Sperren hält und welche Transaktion auf welche Sperren wartet.

Die Lösung beginnt also folgendermaßen:

01: T_1 ist aktiv

02: T_1 ist aktiv, T_1 sperrt x

(b) Wie bezeichnet man die Situation im Zustand 12?

7. Protokoll für Modus-Sperren

In der Vorlesung haben wir ein simples Modell eines Datenbanksystems behandelt, das binäre Sperren $l_i(x)$ (Lock) und $u_i(x)$ verwendet. Wir haben gesehen, wie ein Sperrprotokoll definiert wird, in dem man das Verhalten der Transaktionen einerseits und des Systems, genauer des Lockmanagers andererseits definiert.

Ein etwas realistischeres Modell eines Datenbanksystems verwendet sogenannte Modus-Sperren:

> $lr_i(x)$ bezeichnet einen Read-Lock der Transaktion T_i auf dem Datenobjekt x (besser ist die Bezeichnung „shared lock“).
>
> $lw_i(x)$ bezeichnet einen Write-Lock der Transaktion T_i auf dem Datenobjekt x (besser ist die Bezeichnung „exclusive lock“).
>
> $u_i(x)$ bezeichnet die Freigabe einer Sperre auf Datenobjekt x durch Transaktion T_i.

Beschreiben Sie das Sperrprotokoll für solche Modus-Sperren.

8. Verklemmung & Wartegraph

Gegeben sei folgende Situation: Die Transaktionen T_1 - T_7 halten Sperren auf den Datenobjekten x_i und warten auf die Freigabe der Sperren auf Datenobjekten x_i entsprechend folgender Aufstellung.

Transaktion	hat Lock auf	wartet auf
T_1	x_2	x_1, x_3
T_2	x_3, x_{10}	x_7, x_8
T_3	x_8	x_4, x_5
T_4	x_7	x_1
T_5	x_1, x_5	x_3
T_6	x_4, x_9	x_6
T_7	x_6	x_5

Zeichnen Sie einen Wait-for-Graph und ermitteln Sie Deadlocks.

9. Auflösung des Deadlocks

Verwenden Sie den Wait-for-Graph aus der vorherigen Aufgabe und wenden Sie den Algorithmus aus der Vorlesung an, um die Deadlocks zu erkennen und aufzulösen.

10. DBMS installieren

In der folgenden Aufgabe möchten wir die Phänomene erzeugen, die im SQL-Standard der Definition der Isolationslevel zugrunde liegen.

Zur Vorbereitung der Aufgabe soll jeder von Ihnen zwei Datenbankmanagementsystem auf seinem Rechner installieren – jeweils eines aus einer der beiden folgenden Gruppen:

Gruppe 1:

- Microsoft SQL Server Express https://www.microsoft.com/de-de/sql-server/sql-server-editions-express
- Apache Derby http://db.apache.org/derby
- IBM DB2 Express http://db2express.com/de/

Gruppe 2:

- PostgreSQL http://www.postgresql.org/
- MySQL http://www.mysql.de/
- Oracle Database XE https://www.oracle.com/database/technologies/appdev/xe.html

11. Überweisung zwischen zwei Konten

Spielen Sie die Beispiele für eine Überweisung zwischen zwei Konten aus der Vorlesung mit einem DBMS Ihrer Wahl nach. Experimentieren Sie dabei auch mit den Isolationsleveln.

12. Phänomene

Die Isolationslevel in SQL-Datenbanksystemen werden dadurch definiert, dass angegeben wird, welche Phänomene der Beeinflussung von Transaktionen garantiert ausgeschlossen werden. In dieser Aufgabe sollen Sie in den wechselseitigen Einfluss zweier Transaktionen auf den verschiedenen Isolationsleveln erproben.

Legen Sie eine Datenbank an und schreiben Sie mit JDBC oder ADO.NET kleine Programme, mit denen Sie die Phänomene erzeugen können. Spielen Sie alle Kombinationsmöglichkeiten der Isolationslevel durch.

Vergleichen Sie die Ergebnisse und notieren Sie die Unterschiede der DBMS in Bezug auf die Synchronisation konkurrierender Zugriffe.

13. Eigenschaften von *Snapshot Isolation*

Schreiben Sie ein kleines Programm, das die Anomalie *Write Skew* nachbildet und untersuchen Sie das Verhalten von Oracle und PostgreSQL im Isolationslevel `SERIALIZABLE` in Bezug auf dieses Phänomen.

14. Eine endlose Transaktion

Ein etwas abwegiges Beispiel soll die Konzeption der Isolationslevel veranschaulichen: Gegeben sei eine Relation `PC(model, speed, ram, price)` und wir nehmen an, eine Transaktion enthält eine Endlosschleife, die immerzu nachschaut, ob es mittlerweile einen PC mit 2 Gigahertz unter 1000 Euro gibt. Währenddessen finden andere Transaktionen statt, die mit verschiedenen Isolationsleveln laufen, etwa eine solche, die ein gesuchtes PC-Modell in die Relation einfügt.

Pseudocode:

```
begin transaction;
forever
{
  sleep( 1 sec );
  select * from PC;
  untersuche ergebnismenge;
  if ( found ) goto ende;
}
ende:
  commit();
  ausgabe;
```

Erläutern Sie was passiert, wenn die endlose Transaktion in folgendem Isolationslevel läuft:

(a) `SERIALIZABLE`

(b) `REPEATABLE READ`

(c) `READ COMMITTED`

(d) `READ UNCOMMITTED`

15. Wirkung des Isolationslevels

Wir gehen von folgender Relation aus: `M(MId, Gehalt)`. In der Relation sind zwei Tupel gespeichert: `(A, 1000)` und `(B, 2000)`. Nun werden folgende Transaktionen durchgeführt:

```
Transaktion 1:
  begin transaction;
  update M set Gehalt = Gehalt*2 where MId = 'A';
  update M set Gehalt = Gehalt+100 where MId = 'B';
  commit;

Transaktion 2:
  begin transaction;
  select sum(Gehalt) as G1 from M;
  select sum(Gehalt) as G2 from M;
  commit;
```

Die erste Transaktion wird im Isolationslevel `SERIALIZABLE` durchgeführt.

Berechnen Sie alle möglichen Werte für G1 und G2, wenn

(a) die zweite Transaktion mit dem Isolationslevel `SERIALIZABLE` durchgeführt wird.

(b) die zweite Transaktion mit dem Isolationslevel `READ COMMITTED` durchgeführt wird.

(c) die zweite Transaktion mit dem Isolationslevel `READ UNCOMMITTED` durchgeführt wird.

16. Snapshot-Isolation

Erklären Sie, weshalb bei Snapshot-Isolation folgende Phänomene nicht auftreten können:

(a) Dirty Reads

(b) Lost Updates

(c) Nonrepeatable Reads

(d) Phantom Rows

Ist Snapshot-Isolation mit dem Isolationslevel SERIALIZABLE identisch?

17. SERIALIZABLE

Gegeben seien zwei nebenläufige Transaktionen T_1 und T_2. Beweisen Sie folgende Aussage:

Wird T_1 im Isolationslevel `SERIALIZABLE` ausgeführt und T_2 in einem beliebigen Isolationslevel, dann sieht T_1 entweder alle Änderungen, die T_2 gemacht hat oder keine.

18. Statistische Auswertung

Eine Transaktion für eine statistische Auswertung liest Daten aus der Datenbank, die im vergangenen Monat eingegeben wurden und erstellt aus diesen Daten einen Bericht.

Welches Isolationslevel kann man für diese Transaktion verwenden?

19. Kompensatorische Transaktionen

Geben Sie zu folgenden Datenbankänderungen an, ob es eine kompensatorische Transaktion gibt. Wenn ja, was müsste sie tun?

(a) Erhöhung des Gehalts aller Professoren um 10%.

(b) Erhöhung des Gehalts aller Mitarbeiter um 10%, sofern sie weniger als 3000 Euro verdienen.

(c) Setze die Note von Student Max auf 2.

(d) Füge einen Datensatz mit der Matrikelnummer (Primärschlüssel) 65432 und den Angaben 'Max', 'Schneider' für Vorname, Name ein.

(e) Setze einen Wert auf das Quadrat des bisherigen Wertes.

20. Synchronisation in der Java Persistence API

Lesen Sie das Kapitel 3.4 der Spezifikation der Java Persistence API 1.0 („Optimistic Locking and Concurrency“). Quelle: http://jcp.org/en/jsr/detail?id=220

Vergleichen Sie mit dem Kapitel 3.4 der Version 2.1 Quelle: http://jcp.org/en/jsr/detail?id=338

21. Persistenz von Objekten – ein Code-Beispiel

In SQL hat man die Anweisungen `INSERT` und `UPDATE`. In einem Programm möchte man diesen Unterschied oft verbergen. Hat man etwa eine Klasse `Kunde` mit den Attributen `kNr` und `kName`, dann möchte man eine Methode `save` der Klasse `Kunde` zum Speichern von Objekten programmieren, die selbst überprüft, ob in der Datenbank ein `INSERT` oder ein `UPDATE` nötig ist.

Wir nehmen an, dass in der Datenbank eine Tabelle `Kunde` mit den Feldern `Knr` (Primärschlüssel) und `Kname` angelegt ist.

Neulich habe ich zu diesem Thema in einer *Multiuser-Anwendung* folgenden Code gesehen (vereinfachter Pseudocode):

```
Pseudocode der Methode save der Klasse Kunde:

  Connection con = DriverManager.getConnection(...);
  Statement stmt = con.createStatement();
  ResultSet rs = stmt.executeQuery(
      "SELECT Knr FROM Kunde WHERE Knr = " + this.kNr );

  boolean found =
    ... wenn in rs die Knr gefunden wird, wird die Variable found
```

```
   ... auf true gesetzt

if ( found ) {
    stmt.executeUpdate( "UPDATE Kunde set Kname = " + this.kName +
                  "WHERE Knr = " + this.kNr );
} else {
    stmt.executeUpdate( "INSERT into Kunde values( " + this.kNr +
                   ", '" + this.kName + "')" );
}

con.close();
```

Aufgabe:

(a) Worin besteht der konzeptionelle Fehler in diesem Vorgehen?

(b) Wie kann man das Vorgehen verbessern?

22. Aussage zu Isolationsleveln in SQL

In der deutschen Wikipedia fand ich am 20.10.2009 zum Stichwort „Isolation (Datenbank)“ einen Beitrag, der zum Isolationslevel „Repeatable Read“ folgendes ausführt:

> „**Repeatable Read**
> Bei dieser Isolationsebene ist sichergestellt, dass wiederholte Leseoperationen mit den gleichen Parametern auch dieselben Ergebnisse haben. Üblicherweise wird dies sichergestellt, indem eine Transaktion nur Daten sieht, die vor ihrem Startzeitpunkt vorhanden waren. Eine parallele Änderung führt somit auch nach *commit* nicht zu Inkonsistenzen während einer Transaktion. Dennoch ist es möglich, dass zwei Transaktionen parallel denselben Datensatz modifizieren und nach Ablauf dieser beiden Transaktionen nur die Änderungen von einer von ihnen übernommen werden. “

Diskutieren Sie diesen Abschnitt kritisch. Hinweis: Denken Sie an (a) die Definition der Isolationslevel und (b) an die beiden Techniken der Implementierung, über die wir gesprochen haben.

23. Aktuelle Entwicklungen zum Thema Synchronisation konkurrierender Zugriffe

Sehen Sie sich den Vortrag von Martin Kleppmann auf der Strange Loop September 2015 mit dem Titel „Transactions: myths, surprises and opportunities“ auf Youtube https://youtu.be/5ZjhNTM8XU8 an und bereiten Sie folgende Fragen vor:

- Treffen die Aussagen zu den Isolationsleveln alle zu?
- Welche Probleme mit schwachen Isolationsleveln spricht Kleppmann an?
- Wie steht es mit geographisch verteilten Daten – welche Probleme treten auf?

Rev 3.0 – 10. August 2018