Tutorium 8
Serialisierbarkeitstheorie, Isolationsstufen
Big Data Engineering

Prof. Dr. Jens Dittrich

bigdata.uni-saarland.de

04./05. Juli 2022

Verbesserung Übungsblätter - Häufige Fehler

Aufgabe 1:

- (b) 2.: Versucht, die serielle Reihenfolge aus (b) 1. in der gleichen Reihenfolge in den Multithreads umzusetzen.

Aufgabe 2:

- ungenaue Begründungen.

Aufgabe 3:

- Vergessen, dass T_4 auch bei `READ COMMITTED` abbrechen soll.

Aufgabe 4:

- langen Zyklus ($T_1 \to T_4 \to T_6 \to T_3 \to T_5 \to T_2 \to T_1$) übersehen.

Wiederholung - Frage 1

Frage

Was versteht man unter einem Dirty Read?

Wiederholung - Frage 1

Frage

Was versteht man unter einem Dirty Read?

Lösung

Mit Dirty Read bezeichnen wir das Lesen eines Wertes durch eine Transaktion, der von einer anderen nicht committeten oder abgebrochenen Transaktion geschrieben wurde. Es wurde also ein Wert gelesen, der im Sinne der Isolation noch nicht für andere Transaktionen hätte sichtbar sein dürfen.

Wiederholung - Frage 2

Frage

Bei welchen der unten stehenden Operationen handelt es sich im folgenden Ausführungsplan um einen Dirty Read?

$$w_1(A) \rightarrow r_2(A) \rightarrow w_2(B) \rightarrow r_2(B) \rightarrow r_1(B)$$

(A): $r_2(A)$

(B): $w_2(B)$

(C): $r_2(B)$

(D): $r_1(B)$

Wiederholung - Frage 2

Frage

Bei welchen der unten stehenden Operationen handelt es sich im folgenden Ausführungsplan um einen Dirty Read?

$$w_1(A) \rightarrow r_2(A) \rightarrow w_2(B) \rightarrow r_2(B) \rightarrow r_1(B)$$

Lösung

Die richtigen Antworten lauten (A) und (D):

Es handelt sich lediglich bei $r_2(A)$ aufgrund des vorherigen $w_1(A)$ und bei $r_1(B)$ aufgrund des vorherigen $w_2(B)$ um einen Dirty Read.

Wiederholung - Frage 3

Frage

Was versteht man unter Non-Repeatable Reads und Cascading Rollbacks? Wie hängen diese mit Dirty Reads zusammen?

Wiederholung - Frage 3

Frage

Was versteht man unter Non-Repeatable Reads und Cascading Rollbacks? Wie hängen diese mit Dirty Reads zusammen?

Lösung

- Non-Repeatable Read: Hierunter versteht man, dass eine Transaktion wiederholt dasselbe Datenobjekt liest, aber unterschiedliche Werte erhält ohne selbst Änderungen diesbzgl. eingebracht zu haben. Dabei kann es sich bei den wiederholten Zugriffen um Dirty Reads handeln, muss es aber nicht, da auch Änderungen von committeten Transaktionen nicht sichtbar sein sollten.
- Cascading Rollback: Hier führt eine Transaktion Schreiboperationen basierend auf Dirty Reads aus, wodurch es passieren kann, dass andere Transaktionen die Aktionen basierend auf diesen Schreiboperationen durchführen ebenfalls zurückgesetzt werden müssen (was sich dann kaskadierend fortführen kann).

Wiederholung - Frage 4

Frage

Welche Probleme können Sie in unten stehendem Ausführungsplan identifizieren?

	T_1	T_2	T_3
1	w(A)		
2		r(A)	
3		w(B)	
4			r(B)
5			w(C)
6	abort		
7			commit
8		commit	

Wiederholung - Frage 4

Lösung

T_2 liest in Zeile 2 das Datenobjekt A, wobei T_1 ihre Änderungen noch nicht committet hat, führt also einen Dirty Read aus und greift anschließend schreibend auf B zu, wobei dieser Wert potentiell von A abhängt. Das gleiche passiert nun allerdings bei T_3 in Zeile 4, da hier ebenfalls ein nicht-committeter Wert gelesen wird und danach potentiell in einer Schreiboperation verwendet wird. Das Problem ist hier nun, dass T_1 aborted, das heißt die Änderungen durch T_2 potentiell zurückgesetzt werden müssen, was bedeutet, dass auch T_3 potentiell zurückgesetzt werden müsste. Es handelt sich also um einen Cascading Rollback.

Wiederholung - Frage 5

Frage

Welche Isolationsstufen wurden in der Vorlesung behandelt? Wie behandeln diese jeweils Lese- und Schreibsperren?

Wiederholung - Frage 5

Frage

Welche Isolationsstufen wurden in der Vorlesung behandelt? Wie behandeln diese jeweils Lese- und Schreibsperren?

Lösung

	Read	**Write**
`Read Uncommitted`	keine Sperren	S2PL
`Read Committed`	kurzzeitige Sperren	S2PL
`Repeatable Read`	S2PL	
`Serializable`	S2PL mit Prädikatsperren	

Wiederholung - Frage 6

Frage

Beurteilen Sie folgenden Aussage:

Die Isolationsstufe `Serializable` *ist äquivalent mit dem Konzept der Konfliktserialisierbarkeit.*

Wiederholung - Frage 6

Frage

Beurteilen Sie folgenden Aussage:

Die Isolationsstufe `Serializable` *ist äquivalent mit dem Konzept der Konfliktserialisierbarkeit.*

Lösung

Diese Aussage stimmt nicht, da `Serializable` zwar Konfliktserialisierbarkeit impliziert, allerdings noch viel mehr bietet, da sie z.B. ebenfalls abbrechende Transaktionen handhabt.

Wiederholung - Frage 7

Frage

Ab welcher Isolationsstufe werden Cascading Rollbacks verhindert?

(A): `Read Uncommitted`

(B): `Read Committed`

(C): `Repeatable Read`

(D): `Serializable`

Wiederholung - Frage 7

Frage

Ab welcher Isolationsstufe werden Cascading Rollbacks verhindert?

Lösung

Die richtige Antwort lautet (B):

Wenn eine (oder mehrere) Transaktionen auf geschriebene Objekte einer anderen Transaktion zugreifen, die allerdings noch nicht committet wurde, kann es passieren, dass diese jedoch abbricht, was zur Folge hat, dass sämtliche Operation rückgängig gemacht werden müssen. Dieser sogenannte Cascading Rollback kann jedoch bei `Read Committed` nicht passieren, da hier eine S2PL-Schreibsperre existiert und Leseoperationen eine kurze Lesesperre anfordern müssen. Dadurch kann auf geschriebene Objekte bis zum Commit nicht zugegriffen werden.

Wiederholung - Frage 8

Frage

Was ist das Phantomproblem und wodurch entsteht es?

Wiederholung - Frage 8

Frage

Wodurch entsteht das Phantomproblem?

Lösung

Das Phantomproblem entsteht, wenn während einer Operation auf mehrere Tupel gleichzeitig anstatt nur auf eines zugegriffen wird. Dies kann insofern problematisch werden, als dass z.B. zwischen zwei Lesezugriffen ein Tupel in einer anderen Transaktionen hinzugefügt wird (was durch S2PL nicht blockiert wird), sodass die Lesebefehle mitunter unterschiedliche Ergebnisse liefern können.

Aufgabe 1

Frage

Beweisen oder widerlegen Sie folgende Aussage:

Zwei Ausführungspläne sind konfliktäquivalent genau dann, wenn sie den gleichen Konfliktgraphen haben.

Falls diese Aussage stimmt, so geben Sie einen formalen Beweis, falls nicht argumentieren Sie anhand eines Gegenbeispiels.

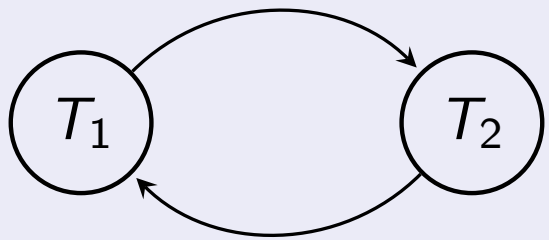
Aufgabe 1

Lösung

Die Aussage stimmt nicht. Betrachten Sie folgende Ausführungspläne:

1. $w_1(A) \rightarrow r_2(A) \rightarrow r_2(B) \rightarrow w_1(B)$
2. $r_2(A) \rightarrow w_1(A) \rightarrow w_1(B) \rightarrow r_2(B)$

Beide Pläne haben den gleichen Konfliktgraphen:

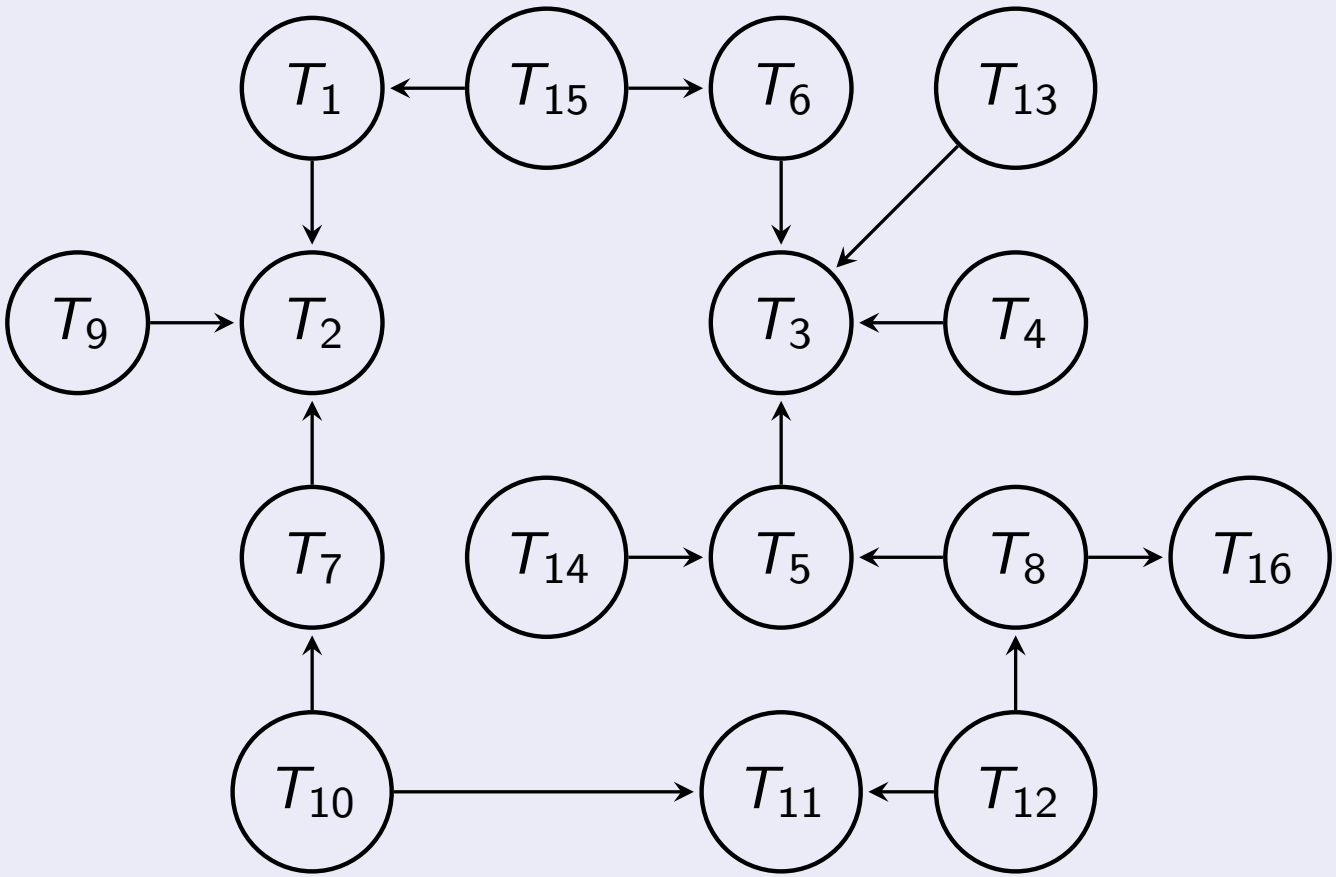


Jedoch sind die Pläne nicht konfliktäquivalent, da die Reihenfolge bei beiden Konfliktoperationen getauscht wurde.

Aufgabe 2

Frage

Betrachten Sie folgenden Konfliktgraphen:



Aufgabe 2.1

Frage

Fertigen Sie einen seriellen Ausführungsplan des obigen Konfliktgraphen an, in welchem alle Konflikte gelöst werden.

Aufgabe 2.1

Lösung

$T_9 \to T_{10} \to T_{12} \to T_{14} \to T_{11} \to T_7 \to T_8 \to T_{16} \to T_5 \to T_{15} \to T_1 \to T_6 \to T_{13} \to T_4 \to T_3 \to T_2$

Aufgabe 2.2

Frage

Angenommen Sie können Transaktionen parallel durchführen, wobei jede Transaktion genau eine Zeiteinheit kostet. Geben Sie für **2 Threads und 4 Threads** einen möglichen Ausführungsplan an, der unter Ausnutzung der parallen Ausführung die Transaktionen möglichst schnell durchführt. Nutzen Sie hierfür die folgende Tabelle als Schablone.

	1	2	3	⋯
Thread 1				
Thread 2				
⋮				

Aufgabe 2.2

Mögliche Lösung für 2 Threads

	1	2	3	4	5	6	7	8
Thread 1	T_9	T_{14}	T_{12}	T_8	T_5	T_4	T_3	T_{11}
Thread 2	T_{10}	T_7	T_{15}	T_6	T_{13}	T_1	T_{16}	T_2

Mögliche Lösung für 4 Threads

	1	2	3	4
Thread 1	T_{15}	T_8	T_1	T_{16}
Thread 2	T_{10}	T_4	T_9	T_3
Thread 3	T_{12}	T_{13}	T_5	T_{11}
Thread 4	T_{14}	T_6	T_7	T_2

Aufgabe 3

Frage

Geben Sie für jede der Isolationsstufen an, welche Transaktionen im unten stehenden Ausführungsplan fehlschlagen. Geben Sie dabei jeweils die Zeile und den Grund des Scheiterns an. Sie können davon ausgehen, dass eine Transaktion fehlschlägt, sofern Sie eine Sperre anfordert, diese aber nicht bekommt.

	T_1	T_2	T_3	T_4
1		r(X.x < 30)		
2	u(Y.y > 20)			
3			u(X.x < 20)	
4				r(Y.y = 125)
5				u(X.x > 40)
6		u(X.x = 130)		
7	i(X.x = 0)			
8				commit
9			commit	
10		commit		
11	commit			

Aufgabe 3

Lösung

- `READ UNCOMMITTED`: T_2 bricht in Zeile 6 ab, da sie eine Schreibsperre für X.x = 130 anfordert, diese allerdings schon durch Zeile 5 (X.x > 40) im Besitz von T_4 ist.
- `READ COMMITTED`: T_4 bricht in Zeile 4 ab, da sie eine Lesesperre für Y.y = 125 anfordert, wobei jedoch T_1 bereits durch Zeile 2 eine Schreibsperre auf Y.y > 20 besitzt.
- `REPEATABLE READ`: T_3 bricht in Zeile 3 ab, da T_2 durch Zeile 1 bereits eine Lesesperre auf X.x < 30 hält. T_4 bricht wegen des gleichen Grundes wie in `Read Committed` ab.
- `SERIALIZABLE`: T_1 bricht in Zeile 7 ab, da T_2 eine Prädikatslesesperre auf X.x < 30 hält. T_3 und T_4 brechen aus dem gleichen Grund wie bei `Repeatable Read` ab.

Aufgabe 4

Frage

Betrachten Sie folgenden unvollständigen Ausführungsplan:

	T_1	T_2	T_3	T_4
1		r(X.x < 30)		
2	?			
3			?	
4			commit	
5				?
6				commit
7	commit			
8		commit		

Aufgabe 4

Frage

Vervollständigen Sie den vorherigen Ausführungsplan, sodass bei allen Isolationsstufen die folgenden Bedingungen erfüllt sind:

- `READ UNCOMMITTED`: T_3 bricht ab, alle anderen Transaktionen committen erfolgreich.
- `READ COMMITTED`: T_3 bricht ab, alle anderen Transaktionen committen erfolgreich.
- `REPEATABLE READ`: T_1 bricht ab, alle anderen Transaktionen committen erfolgreich.
- `SERIALIZABLE`: T_1 und T_4 brechen ab, alle anderen Transaktionen committeten erfolgreich.

Sie können davon ausgehen, dass eine Transaktion fehlschlägt, sofern Sie eine Sperre anfordert, diese aber nicht sofort erhält.

Aufgabe 4

Lösung

- Zeile 2: Hier kann man z.B. u(X.x < 60) einfügen, was ab `Repeatable Read` fehlschlägt, da hier eine Schreibsperre für X.x < 60 angefordert wird, T_2 allerdings schon durch Zeile 1 eine Lesesperre für X.x < 30 hat.
- Zeile 3: Hier kann man z.B. u(X.x = 55) einfügen, was ab `Read Uncommitted` fehlschlägt, da T_1 durch Zeile 2 eine Schreibsperre auf X.x < 60 besitzt. Dies gilt jedoch nur bis `Repeatable Read`, da ab dann T_1 abbricht, sodass T_3 die Sperre erhält.
- Zeile 5: Hier kann man z.B. i(X.x = 25) einfügen, was dazu führt, dass T_4 ab `Serializable` aufgrund der Leseprädikatsperre auf X.x < 30 von T_2 durch Zeile 1 fehlschlägt.