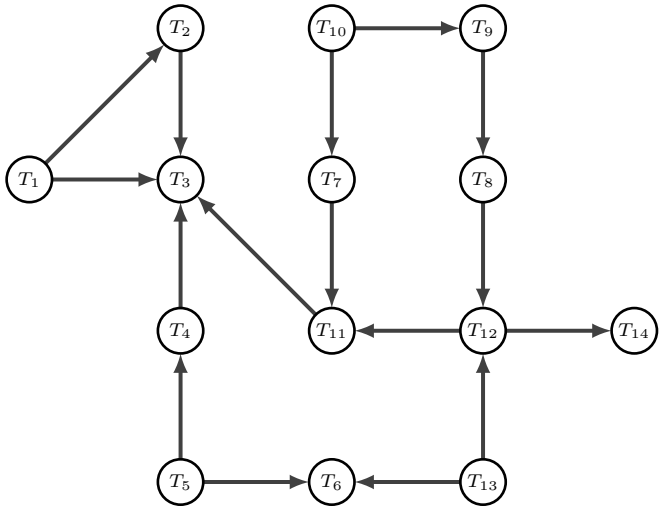
1 Konfliktgraphen (6 Punkte)

(a) Betrachten Sie die folgenden beiden Ausführungspläne:

	Ausführungspläne
AP_1	$r_3(\mathrm{C}) \rightarrow r_2(\mathrm{B}) \rightarrow r_1(\mathrm{A}) \rightarrow w_2(\mathrm{B}) \rightarrow w_3(\mathrm{C}) \rightarrow w_1(\mathrm{A}) \rightarrow r_1(\mathrm{B}) \rightarrow r_3(\mathrm{A}) \rightarrow w_1(\mathrm{B}) \rightarrow w_3(\mathrm{A})$
AP_2	$r_2(\mathrm{B}) \rightarrow r_3(\mathrm{C}) \rightarrow w_3(\mathrm{C}) \rightarrow r_1(\mathrm{A}) \rightarrow r_3(\mathrm{A}) \rightarrow w_3(\mathrm{A}) \rightarrow w_2(\mathrm{B}) \rightarrow w_1(\mathrm{A}) \rightarrow r_1(\mathrm{B}) \rightarrow w_1(\mathrm{B})$

Erstellen Sie für jeden Ausführungsplan den entsprechenden Konfliktgraphen und argumentieren Sie, ob der Ausführungsplan konfliktserialisierbar ist. Sofern der Ausführungsplan konfliktserialisierbar ist, sortieren Sie die Nicht-Konfliktoperationen so um, dass Sie einen seriellen, konfliktäquivalenten Ausführungsplan erhalten und geben Sie diesen an.

(b) Betrachten Sie folgenden Konfliktgraphen.



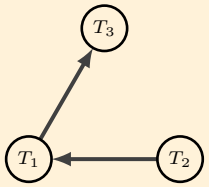
1. Bringen Sie die Transaktionen des Konfliktgraphen in eine serielle Ausführungsreihenfolge, sodass alle Konflikte aufgelöst werden.
2. Gehen Sie nun davon aus, dass Sie Transaktionen parallel ausführen können und alle Transaktionen eine Zeiteinheit zur Ausführung ihrer Operationen benötigen. Geben Sie **für 2 Threads und für 4 Threads** einen Ausführungsplan an, der unter Ausnutzung von paralleler Ausführung die Transaktionen in der kürzest möglichen Zeit ausführt. Nutzen Sie dabei folgende Tabelle zur Darstellung eines Ausführungsplans, wobei eine Zeile einen Thread und eine Spalte eine Zeiteinheit darstellt.

	1	2	3	...
Thread 1				
Thread 2				
⋮				

Lösung:

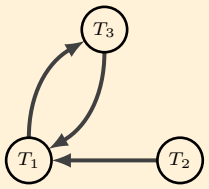
(a) Vorgesehen ist 1 Punkt pro Plan (0,5 Punkte für einen korrekten Konfliktgraphen + 0,5 Punkte für eine korrekte Argumentation mitsamt einem korrekten konfliktäquivalenten, seriellen Ausführungsplan) .

- AP_1:
  Kanten z.B. durch: $r_2(\text{B}) \rightarrow w_1(\text{B})$ und $r_1(\text{A}) \rightarrow w_3(\text{A})$.



  Der Konfliktgraph enthält keinen Zyklus, darum ist der Ausführungsplan konfliktserialisierbar. Der zugehörige serielle Ausfürungspläne wäre hiermit $T_2 \rightarrow T_1 \rightarrow T_3$.

- AP_2:
  Kanten z.B. durch: $r_1(\text{A}) \rightarrow w_3(\text{A})$, $r_3(\text{A}) \rightarrow w_1(\text{A})$ und $r_2(\text{B}) \rightarrow w_1(\text{B})$.



  Der Konfliktgraph enthält einen Zyklus zwischen T_1 und T_3, darum ist der Ausführungsplan nicht konfliktserialisierbar.

(b) 1. Vorgesehen ist 1 Punkt (je 0,5 Punkte Abzug für inkorrekt geordnete Transaktionspaare, usw.). Es gibt mehrere Möglichkeiten für serielle Ausführungspläne. Für einen korrekten Ausführungsplan muss folgendermaßen vorgegangen werden:

   (i) Suche einen Knoten ohne eingehende Kanten und füge ihn zum Plan hinzu.
   (ii) Entferne alle ausgehenden Kanten dieses Knotens aus dem Graphen.
   (iii) Wiederhole Schritte (i) und (ii) bis der Graph keinen Knoten mehr enthält.

   Eine mögliche Lösung:

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

2. Vorgesehen sind 1 Punkt für den Plan mit zwei Threads und 2 Punkte für den Plan mit vier Threads (je 0,5 Punkte Abzug für inkorrekt parallelisierte Transaktionspaare, unnötige Zeiteinheiten, usw.).

   2 Threads: Eine mögliche Lösung:

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

   4 Threads: Eine mögliche Lösung:

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

2 Isolationsstufen I (8 Punkte)

In der Vorlesung sowie im Notebook `Transaction Manager.ipynb` haben wir gesehen, dass die Isolationsstufe bei gleichzeitiger Ausführung mehrerer Transaktionen einen Einfluss darauf haben kann, welche Transaktionen abbrechen und welche festgeschrieben werden.

Gegeben sei die folgende Ausführungsreihenfolge der Transaktionen T_1 bis T_5. Nehmen Sie an, dass die Attribute x, y und z jeweils den Tabellen X, Y bzw. Z entnommen sind.

	T_1	T_2	T_3	T_4	T_5
1		r(Y.y $\neq$ 30)			
2			r(X.x $\leq$ 100)		
3			r(Z)		
4	u(Y.y = 29)				
5	commit				
6			u(X.x $\geq$ 100)		
7		r(Z.z = 20)			
8		u(X.x = 120)			
9		commit			
10				r(X)	
11					u(Y.y = 30)
12					i(Z.z = 20)
13					commit
14				i(Z.z $\geq$ 40)	
15				commit	
16			r(Y.y = 31)		
17			commit		

Der Ausführungsplan besteht aus `read` (r), `update` (u) und `insert` (i) Operationen. Die Prädikate in den einzelnen Operationen zeigen an, welche Objekte von der entsprechenden Operation betroffen sind. So verändert zum Beispiel die Operation u(X.x $<$ 5) alle Tupel aus X, deren Attribut x einen Wert kleiner als fünf hat. Die Operation u(X) verändert alle Tupel aus X, da kein Prädikat spezifiziert ist. Beachten Sie, dass die Prädikate je nach Isolationsstufe auch für Prädikatsperren verwendet werden. Bei einer Einfügeoperation gehen wir davon aus, dass das eingefügte Objekt noch nicht in der Datenbank existiert.

Wir nehmen für diese Aufgabe an, dass eine Transaktion abbricht, falls sie eine Sperre anfordert, diese aber nicht sofort bekommt.

Geben Sie für jede der folgenden Isolationstufen an, welche der Transaktionen festgeschrieben werden und welche abbrechen. Begründen Sie jeweils Ihre Antwort und nennen Sie im Falle eines Abbruchs die genaue Stelle.

1. `Read Uncommitted`
2. `Read Committed`
3. `Repeatable Read`
4. `Serializable`

Lösung:

Vorgesehen sind 2 Punkt pro Isolationsstufe, davon 1 Punkt für die Begründung und 1 Punkt für die genaue Zeile bzw. das korrekte Beispiel, welches im Vergleich zur vorherigen Isolationsstufe neu dazu gekommen ist (je 0,5 Punkte Abzug für falsche/zusätzliche Beispiele, falsche Begründung, usw.).

1.
   - Zeile 8: T_2 bricht ab, da T_3 durch u(X.x $\geq$ 100) in Zeile 6 eine Schreibsperre auf X.x = 120 hält.
2.
   - wie bei 1.
   - Zeile 10: T_4 bricht ab, da T_3 durch u(X.x $\geq$ 100) in Zeile 6 eine Schreibsperre auf X.x $\geq$ 100 hält.
3.
   - wie bei 2.
   - Zeile 4: T_1 bricht ab, da T_2 durch r(Y.y $\neq$ 30) in Zeile 1 eine Lesesperre auf Y.y = 29 hält.
4.
   - wie bei 3.
   - Zeile 12: T_5 bricht ab, da T_3 durch r(Z) in Zeile 3 eine Prädikatslesesperre auf Z.z = 20 hält.

3 Isolationsstufen II (3 Punkte)

Gegeben sei die folgende lückenhafte Ausführungsreihenfolge der Transaktionen T_1 bis T_4. Nehmen Sie an, dass die Attribute x, y und z jeweils den Tabellen X, Y bzw. Z entnommen sind.

	T_1	T_2	T_3	T_4
1	r(Y.y>30)			
2		Lücke T_2		
3			r(X.x = 20)	
4				r(X.x ≥ 20)
5				commit
6			Lücke T_3	
7			commit	
8	Lücke T_1			
9		i(Z.z = 4)		
10		commit		
11	commit			

Analog zur zweiten Aufgabe besteht der Ausführungsplan aus `read` (r), `update` (u) und `insert` (i) Operationen. Die Prädikate in den einzelnen Operationen zeigen an, welche Objekte von der entsprechenden Operation betroffen sind. So verändert zum Beispiel die Operation u(X.x < 5) alle Tupel aus X, deren Attribut x einen Wert kleiner als fünf hat. Die Operation u(X) verändert alle Tupel aus X, da kein Prädikat spezifiziert ist. Beachten Sie, dass die Prädikate je nach Isolationsstufe auch für Prädikatsperren verwendet werden. Bei einer Einfügeoperation gehen wir davon aus, dass das eingefügte Objekt noch nicht in der Datenbank existiert.

Wir nehmen wie in der zweiten Aufgabe an, dass eine Transaktion abbricht, falls sie eine Sperre anfordert, diese aber nicht sofort bekommt.

Fügen Sie in jede Lücke eine Operation ein, sodass die folgenden Bedingungen gelten:

Isolationsstufe	T_1	T_2	T_3	T_4
`Read Uncommitted`	✓	✓	✓	✓
`Read Committed`	✓	✓	✓	✗
`Repeatable Read`	✓	✓	✗	✗
`Serializable`	✓	✗	✗	✗

Bei einem Häkchen (✓) wird die Transaktion festgeschrieben, bei einem Kreuzchen (✗) bricht Sie vorher ab.

Geben Sie zu jeder der drei ausgefüllten Lücken kurz an, warum Sie sich für die jeweilige Operation entschieden haben.

Tipp: Überlegen Sie sich für jede Transaktion welchen Konflikt Sie herbeiführen müssen, damit diese erst ab der entsprechenden Isolationsstufe abbricht und wählen Sie die Operationen für die Lücken dementsprechend.

Lösung:

Vorgesehen ist 1 Punkt pro Lücke, davon 0,5 Punkte für die Begründung und 0,5 Punkte für die korrekte Operation (je 0,5 Punkte Abzug für falsche Operationen, zusätzliche Konflikte, falsche Begründung, usw.)

- Lücke T_2: Schreib- oder Einfügeoperationen auf X, die sich mit der Lesesperre von T_4 durch Zeile 4 überschneiden, jedoch keinen Konflikt mit Zeile 3 hervorrufen.
  Zum Beispiel: u(X.x=21), i(X.x=21), u(X.x>126), ...
- Lücke T_3: Schreiboperationen auf Y, die sich mit der Lesesperre von T_1 durch Zeile 1 überschneiden. Ein Konflikt mit T_4 ist nicht möglich, da diese Transaktion bereits abgebrochen ist.
  Zum Beispiel: u(Y.y=35), u(Y), u(Y.y>30), ...
- Lücke T_1: Lese-, Schreib-, oder Einfügeoperationen in Z, die sich mit der Einfügeoperationen in Z in Zeile 9 überschneiden.
  Zum Beispiel: r(Z), i(Z.z=4), i(X.x<5), ...

Eine mögliche Lösung sieht somit folgendermaßen aus:

	T_1	T_2	T_3	T_4
1	r(Y.y>30)			
2		u(X.x>20)		
3			r(X.x = 20)	
4				r(X.x $\geq$ 20)
5				commit
6			u(Y.y>30)	
7			commit	
8	r(Z)			
9		i(Z.z = 4)		
10		commit		
11	commit			

4 Deadlocks (3 Punkte)

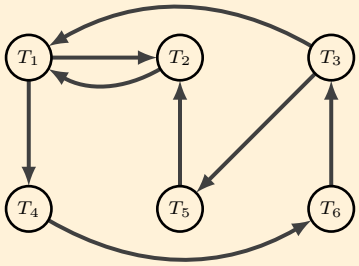
Gegeben seien die Transaktionen $T_1 - T_6$ die laut folgender Tabelle sowohl Sperren auf den Datenobjekten $a - j$ besitzen, als auch auf Freigabe anderer Sperren warten.

Transaktion	besitzt Sperre für	wartet auf Freigabe von
T_1	b, c	a, e
T_2	a	b
T_3	d	f, c
T_4	e, i	g
T_5	f	a
T_6	g, h	d

Zeichnen Sie den Wartegraphen und zeigen Sie eventuelle Deadlocks auf. Im Falle von Deadlocks, geben Sie die kleinste Menge an Transaktionen an, die die Deadlocks durch Zurücksetzen auflösen.

Lösung: 1 Punkt bei korrektem Graphen (-0,5 Punkte für falsche/fehlende Kanten), 1 Punkt für korrekte gefundene Deadlocks (-0,5 Abzug für fehlende/falsche Deadlocks), 1 Punkt bei korrekter Auflösung des Konflikts

Man erhält folgenden Wartegraphen:



Durch Betrachtung des Wartegraphen fallen einem drei verschiedene Zyklen und damit Deadlocks auf:

- $T_1 \to T_2 \to T_1$
- $T_1 \to T_4 \to T_6 \to T_3 \to T_1$
- $T_1 \to T_4 \to T_6 \to T_3 \to T_5 \to T_2 \to T_1$

Alle Zyklen lassen sich durch Zurücksetzten von T_1 auflösen.

Abgabe

Lösungen sind in Teams von 2 bis 3 Studierenden bis zum 30. Juni 2022, 10:15 Uhr über Ihre persönlichen Statusseite im CMS einzureichen. Nutzen Sie hierfür die Team Groupings Funktionalität im CMS.

Reichen Sie die Abgabe als PDF ein, die Ihre Lösungen zu den Aufgaben 1, 2, 3 und 4 enthält.

Abgaben, die nicht den oben angegeben Vorgaben entsprechen, führen zu Punktabzug. Einzelabgaben werden nicht mehr korrigiert und mit 0 Punkten bewertet.