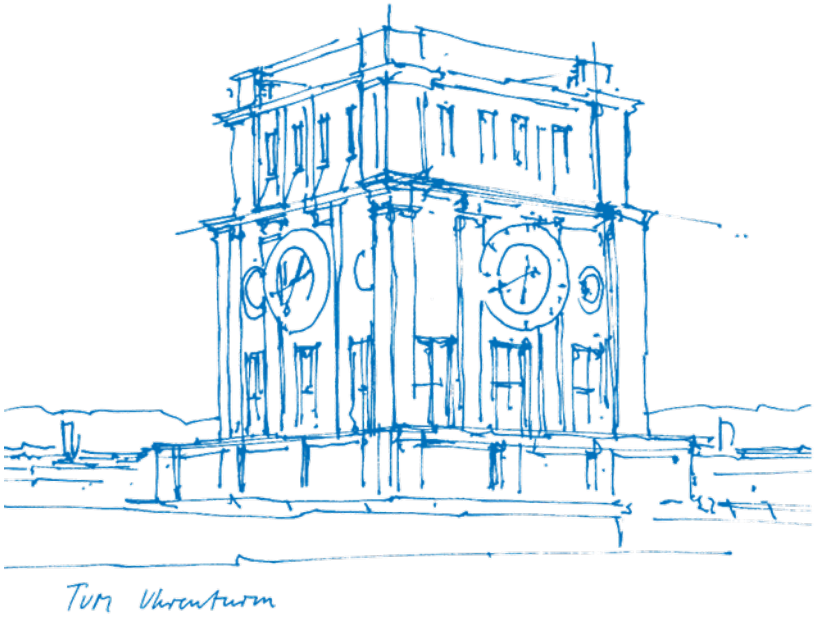
Zentralübung ERDB 2018

Maximilian E. Schüle
Technische Universität München
Institut für Informatik
Lehrstuhl III: Datenbanksysteme
Garching, 12. Juli 2018



Klausur

Hauptklausur

Freitag, 20.07.2018, 16-18 Uhr

Von	Bis	Raum
A	K	MW0001
L	Net	MW2001 Galerie
Neu	Z	MW2001

Wiederholungsklausur

Freitag, 08.10.2018, 16-18 Uhr

Von	Bis	Raum
A	Z	MW0001

Durchführung

90 Minuten Bearbeitungszeit

keine Hilfsmittel (wie Taschenrechner, Spickzettel,...) erlaubt!

Überblick Prüfungsstoff

Disclaimer

Werde heute hauptsächlich besprechen welche Themen besonders wichtig sind.

Falls etwas auf den Folien nicht erwähnt ist, aber in den Übungen / Vorlesung besprochen wurde, kann daraus nicht geschlossen werden, dass es nicht in der Prüfung drankommt.

Auf den Folien dargestellte Übungen können Fehler enthalten.

Recovery

Blatt 01 (besonders wichtig: 01.1,01.2,01.3,01.4)

- **Write Ahead Logging (WAL)**
  - Schreiben der Log-Einträge vor Commit
  - Vor Auslagerung einer Seite: Schreiben aller zugehörigen Log-Einträge

Recovery

Blatt 01 (besonders wichtig: 01.1,01.2,01.3,01.4)

- **Write Ahead Logging (WAL)**
  - Schreiben der Log-Einträge vor Commit
  - Vor Auslagerung einer Seite: Schreiben aller zugehörigen Log-Einträge
- **Wiederanlaufphasen**
  1. Analyse der Winner und Loser
  2. Redo aller Transaktion
  3. Undo der Losertransaktionen

Recovery

Blatt 01 (besonders wichtig: 01.1,01.2,01.3,01.4)

- **Write Ahead Logging (WAL)**
  - Schreiben der Log-Einträge vor Commit
  - Vor Auslagerung einer Seite: Schreiben aller zugehörigen Log-Einträge
- **Wiederanlaufphasen**
  1. Analyse der Winner und Loser
  2. Redo aller Transaktion
  3. Undo der Losertransaktionen
- Log-Einträge und Kompensationslogeinträge (CLR)
- Unterschied Physische/Logische Protokollierung

Recovery

Blatt 01 (besonders wichtig: 01.1,01.2,01.3,01.4)

- **Write Ahead Logging (WAL)**
  - Schreiben der Log-Einträge vor Commit
  - Vor Auslagerung einer Seite: Schreiben aller zugehörigen Log-Einträge
- **Wiederanlaufphasen**
  1. Analyse der Winner und Loser
  2. Redo aller Transaktion
  3. Undo der Losertransaktionen
- Log-Einträge und Kompensationslogeinträge (CLR)
- Unterschied Physische/Logische Protokollierung

- Wiederanlaufphasen bei Seitenersetzungsstrategien

	Force	No Force
Steal	Kein Redo, Undo	Redo, Undo
No Steal	Kein Redo, Kein Undo	Redo, Kein Undo

  - No Force: Redo-Phase notwendig
  - Steal: Undo-Phase notwendig
- MMDBs: No Force, No Steal: Redo-Phase, keine Undo-Phase notwendig

Recovery

Schritt	T_1	T_2	Log [LSN,TA,PageID,Redo,Undo,PrevLSN]
1.	**BOT**		[#1, T_1, **BOT**, 0]
2.	$r(A, a_1)$		
3.		**BOT**	[#2, T_2, **BOT**, 0]
4.		$r(C, c_2)$	
5.	$a_1 := a_1 - 50$		
6.	$w(A, a_1)$		[#3, T_1, P_A, A–=50, A+=50, #1]
7.		$c_2 := c_2 + 100$	
8.		$w(C, c_2)$	[#4, T_2, P_C, C+=100, C–=100, #2]
9.	$r(B, b_1)$		
10.	$b_1 := b_1 + 50$		
11.	$w(B, b_1)$		[#5, T_1, P_B, B+=50, B–=50, #3]
12.	**commit**		[#6, T_1, **commit**, #5]
13.		$r(A, a_2)$	
14.		$a_2 := a_2 - 100$	
15.		$w(A, a_2)$	[#7, T_2, P_A, A–=100, A+=100, #4]
16.		**commit**	[#8, T_2, **commit**, #7]

Recovery

Schritt	T_1	T_2	Log [LSN,TA,PageID,Redo,Undo,PrevLSN]
1.	**BOT**		[#1, T_1, **BOT**, 0]
2.	$r(A, a_1)$		
3.		**BOT**	[#2, T_2, **BOT**, 0]
4.		$r(C, c_2)$	
5.	$a_1 := a_1 - 50$		
6.	$w(A, a_1)$		[#3, T_1, P_A, A–=50, A+=50, #1]
7.		$c_2 := c_2 + 100$	
8.		$w(C, c_2)$	[#4, T_2, P_C, C+=100, C–=100, #2]
9.	$r(B, b_1)$		
10.	$b_1 := b_1 + 50$		
11.	$w(B, b_1)$		[#5, T_1, P_B, B+=50, B–=50, #3]
12.	**commit**		[#6, T_1, **commit**, #5]
13.		$r(A, a_2)$	
14.		$a_2 := a_2 - 100$	
15.		$w(A, a_2)$	[#7, T_2, P_A, A–=100, A+=100, #4]
16.		**commit**	[#8, T_2, **commit**, #7]

A, B und C mit 1000, 2000 und 3000

[#3, T_1, P_A, A=950, A=1000, #1]

[#4, T_2, P_C, C=3100, C=3000, #2]

[#5, T_1, P_B, B=2050, B=2000, #3]

[#7, T_2, P_A, A=850, A=950, #4]

Recovery

Schritt	T_1	T_2	Log [LSN,TA,PageID,Redo,Undo,PrevLSN]
1.	**BOT**		[#1, T_1, **BOT**, 0]
2.	$r(A, a_1)$		
3.		**BOT**	[#2, T_2, **BOT**, 0]
4.		$r(C, c_2)$	
5.	$a_1 := a_1 - 50$		
6.	$w(A, a_1)$		[#3, T_1, P_A, A–=50, A+=50, #1]
7.		$c_2 := c_2 + 100$	
8.		$w(C, c_2)$	[#4, T_2, P_C, C+=100, C–=100, #2]
9.	$r(B, b_1)$		
10.	$b_1 := b_1 + 50$		
11.	$w(B, b_1)$		[#5, T_1, P_B, B+=50, B–=50, #3]
12.	**commit**		[#6, T_1, **commit**, #5]
13.		$r(A, a_2)$	
14.		$a_2 := a_2 - 100$	
15.		$w(A, a_2)$	[#7, T_2, P_A, A–=100, A+=100, #4]
16.		**commit**	[#8, T_2, **commit**, #7]

A, B und C mit 1000, 2000 und 3000

[#3, T_1, P_A, A=950, A=1000, #1]

[#4, T_2, P_C, C=3100, C=3000, #2]

[#5, T_1, P_B, B=2050, B=2000, #3]

[#7, T_2, P_A, A=850, A=950, #4]

Absturz nach Schritt 13 – CLR?
<LSN, TA, Page-ID, Redo, PrevLSN, NextLSN>
<#4′, T_2, P_C, C–=100, #4, #2 >

Mehrbenutzersynchronisation

Blatt 02 (besonders wichtig: 02.1, 02.2, 02.3)

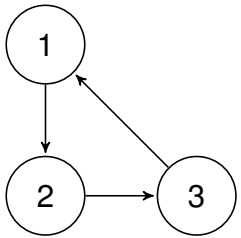
`https://transactions.db.in.tum.de/`

- Aufspannen des **Serialisierbarkeitsgraphen (SG)**
- Historienklassen und Zuordnung
  - *SR*: Serialisierbarkeitsgraph $SG(H)$ azyklisch
  - *RC*: Wenn T_i liest T_j, dann $c_j <_H c_i$
  - *ACA*: Wenn T_i liest T_j bezüglich Datum A, dann $c_j <_H r_i(A)$
  - *ST*: Wenn $w_j(A) <_H o_i(A)$, dann $c_j <_H o_i(A)$ oder $a_j <_H o_i(A)$
- Überlegen, ob Sperren mit 2PL/striktes 2PL (2-Phase-Locking) möglich sind

Mehrbenutzersynchronisation

Eigenschaften von Historien?

$$w_1(x), r_2(y), w_3(y), w_2(x), w_3(z), c_3, w_1(z), c_2, c_1$$



- nicht SR (und somit nicht 2PL)
- RC
- ACA
- nicht ST wegen $w_1(x) < w_2(x) < c_1$

Fügen Sie commits ein: $H = w_1(x), w_1(y), r_2(x), r_2(y)$

- dass die Historie RC aber nicht ACA erfüllt: $w_1(x), w_1(y), r_2(x), r_2(y), c_1, c_2$
- dass die Historie ACA erfüllt. $w_1(x), w_1(y), c_1, r_2(x), r_2(y), c_2$

Sicherheit

Blatt 03.1-03.5 (wichtig: 03.4)

- **RSA** Welche Schlüssel?, Verschlüsseln, Entschlüsseln, Signieren
- **SQL Injection** `http://db.in.tum.de/~schuele/sql_verzeichnis.html`
- k-Anonymität, Angriffsarten

Datalog

Blatt 03.6-04.5 (besonders wichtig: 03.6,04.1,04.2,04.3)

- Datalog Theorie (Wann ist Programm sicher? Stratifizierbar?) im Hintergrund
- **Datalog Programme! Wichtig!**
- Definition neuer Regeln
- $\setminus =$, *not*(...), $+$
- Einfache Regeln zu SQL übersetzen und zurück
- Rekursion
- Domänenkalkül zu Datalog übersetzen
- Keine Aggregationsfunktionen!
- Üben!!!: `http://datalog.db.in.tum.de/`

```
indirekt(A,C,S) :- direkt(A,C,_),S=0.
indirekt(A,C,S) :- indirekt(A,B,R),direkt(B,C,_),S=R+1.
indirekt(garching_forschungszentrum,C,S).
```

Verteilte Datenbanken

Blatt 04.6-05.6 (besonders wichtig: 04.6,04.7,05.1,05.3,05.6)

- horizontale und vertikale **Fragmentierung**
  - Vertikal/Horizontal
  - Korrektheit
  - Rekonstruktion
- **Quorum Consensus**
  - Lesequorum $Q_r(A)$, Schreibquorum $Q_W(A)$
  - $2 * Q_w(A) > W(A)$
  - $Q_r(A) + Q_w(A) > W(A)$
- **Chord**-Netzwerk
  - Finden von Schlüsseln
  - Fingertabellen ausfüllen

	vertikal	horizontal
fragmentieren	π Projektion	σ Selektion
vereinigen	$\bowtie$ Join	$\cup$ Union

Betriebliche Anw./Data Warehouse/Data-Mining

Blatt 06.1-07.2 (besonders wichtig: 06.1,06.2,06.3.06.4,06.5)

- **Apriori**: Frequentitemsets bestimmen, Konfidenz von Assoziationsregeln ableiten
- **Skyline**

```
select MatrNr from Klausur k skyline of k.Vorbereitungszeit min, k.Note min

select MatrNr from Klausur k where not exists (
   select * from klausur dom where
     dom.Vorbereitungszeit <= k.Vorbereitungszeit and dom.Note <= k.Note and
     (dom.Vorbereitungszeit < k.Vorbereitungszeit or dom.Note < k.Note)
)
```

- **Threshold/NRA**
- Fensterfunktionen (Windowfunctions)

Betriebliche Anw./Data Warehouse/Data-Mining: Windowfunctions

- Laufende Summe

```
SELECT *, sum(gehalt) OVER (PARTITION BY rang ORDER BY gehalt DESC)
```

- Wachsende Anzahl

```
SELECT *, count(gehalt) OVER (PARTITION BY rang ORDER BY gehalt DESC)
```

- gleitendes Mittel

```
SELECT *, avg(gehalt) OVER (PARTITION BY rang ORDER BY gehalt DESC
RANGE BETWEEN 500 PRECEDING AND 500 FOLLOWING) FROM Professoren;
```

- Ranken

```
SELECT * FROM (
SELECT *, rank() OVER (ORDER BY gehalt desc) FROM Professoren
) WHERE rank < 4
```

Hauptspeicherdatenbanksysteme

Blatt 07.3,07.4,07.5,07.6,07.7,08.1,08.2,08.3 (besonders wichtig: 08.1,08.2)

- **Adaptive Radix Tree**: Finden von Schlüsseln, Einfügen von Schlüsseln
- **MVCC** mit Precision Locking: welche Historien sind möglich?
  1. Lesende Anfragen sind immer erlaubt: kein Precision Locking. Falls schreibend:
  2. Überlappender Prädikatbereich? Falls ja, dann:
  3. BOT und commit-Reihenfolge beachten.

XML, XPath und XQuery

Blatt 09.1,09.2,10.1,10.2,10.3,10.4 (besonders wichtig: 09.1,09.2,10.1,10.2,10.3,10.4)

- Grundkenntnisse XML-Dokumente lesen
- **XPath**: Achsen
- **XQuery**: FLOWR, Grundlegende Aggregatsfunktionen wie count()
- Syntax XML vs. JSON kennen

Big Data

Blatt 10.5,10.6,11.1,11.2,11.3 (besonders wichtig: 10.6, 11.1, 11.3)

- **TF-IDF**-Werte für Wörter berechnen (log-Werte dürfen stehen bleiben)
- **PageRank** und **HITS** für Graphen berechnen

Hitliste FAQs

Vorjahr:

1. Gilt der Notenbonus für beide Klausuren? Ja.
2. Was ist der Unterschied zwischen Haupt- und Hintergrundspeicher..?

Heuer:

1. Das Datalog-Tool akzeptiert keine Regeln bei ’Query’ – ist es defekt? Nein, Regeln bei ’rules’ angeben.

Leider nie gefragt und dennoch vorgefallen:

1. Macht es Sinn, die erste Klausur durchzustreichen? Nein, nie, die Aufgaben sind gleich schwer, aber unterschiedlich.
2. Dürfen Taschenrechner oder Spickzettel in die Klausur genommen? Nein, dies wird als Unterschleif gewertet.