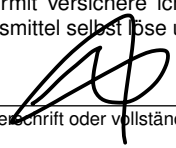


Bestätigung der Verhaltensregeln

Hiermit versichere ich, dass ich diese Klausur ausschließlich unter Verwendung der unten aufgeführten Hilfsmittel selbst löse und unter meinem Namen abgebe.


Unterschrift oder vollständiger Name, falls keine Stifteingabe verfügbar

Einsatz und Realisierung von Datenbanksystemen

Klausur: IN2031 / Finalklausur
Prüfer: Prof. Dr. Alfons Kemper

Datum: Mittwoch, 29. Juli 2020
Uhrzeit: 08:00 – 09:30

Bearbeitungshinweise

- Diese Klausur umfasst **12 Seiten** mit insgesamt **10 Aufgaben**.
Bitte kontrollieren Sie jetzt, dass Sie eine vollständige Angabe erhalten haben.
- Die Gesamtpunktzahl in dieser Prüfung beträgt 90 Punkte.
- Das Heraustrennen von Seiten aus der Prüfung ist untersagt.
- Als Hilfsmittel sind zugelassen:
 - alle Vorlesungsmaterialien („Open-Book“)
 - ein **analoges Wörterbuch** Deutsch ↔ Muttersprache **ohne Anmerkungen**
- Mit * gekennzeichnete Teilaufgaben sind ohne Kenntnis der Ergebnisse vorheriger Teilaufgaben lösbar.
- **Es werden nur solche Ergebnisse gewertet, bei denen der Lösungsweg erkennbar ist.** Auch Textaufgaben sind **grundsätzlich zu begründen**, sofern es in der jeweiligen Teilaufgabe nicht ausdrücklich anders vermerkt ist.
- Schreiben Sie weder mit roter / grüner Farbe noch mit Bleistift.
- Schalten Sie alle mitgeführten elektronischen Geräte vollständig aus, verstauen Sie diese in Ihrer Tasche und verschließen Sie diese.

Hörsaal verlassen von _____ bis _____ / Vorzeitige Abgabe um _____





Aufgabe 1 Recovery (7 Punkte)

In der folgenden Tabelle ist die verzahnte Ausführung der drei Transaktionen T_1 , T_2 und T_3 sowie das zugehörige Log auf der Basis logischer Protokollierung gezeigt.

Initialwerte: $A = 100$; $B = 200$; $C = 300$

Schritt	T_1	T_2	T_3	Log
1.	BOT			[#1, T_1 , BOT , 0]
2.	$r(A, a_1)$			
3.		BOT		[#2, T_2 , BOT , 0]
4.		$r(B, b_2)$		
5.		$b_2 := b_2 + 40$		
6.		$w(B, b_2)$		[#3, T_2 , P_B , $B += 40$, $B -= 40$, #2]
7.		commit		[#4, T_2 , commit , #3]
8.	$r(C, c_1)$			
9.	$c_1 := c_1 - 130$			
10.	$a_1 := a_1 + 130$			
11.			BOT	[#5, T_3 , BOT , 0]
12.	$w(A, a_1)$			[#6, T_1 , P_A , $A += 130$, $A -= 130$, #1]
13.	$w(C, c_1)$			[#7, T_1 , P_C , $C -= 130$, $C += 130$, #6]
14.			$r(C, c_3)$	
15.			$c_3 := c_3 + 90$	
16.			$w(C, c_3)$	[#8, T_3 , P_C , $C += 90$, $C -= 90$, #5]
17.			commit	[#9, T_3 , commit , #8]
18.	commit			[#10, T_1 , commit , #7]

0	
1	
2	
3	
4	

a)* Das Log soll nun in die Form der physischen Protokollierung überführt werden. Geben Sie hierzu chronologisch geordnet alle geänderten Logeinträge an.

[#3, T_2 , P_B , $B = 240$, $B = 200$, #2]

[#6, T_1 , P_A , $A = 230$, $A = 100$, #1]

[#7, T_1 , P_C , $C = 170$, $C = 300$, #6]

[#8, T_3 , P_C , $C = 260$, $C = 170$, #5]

0	
1	
2	
3	

b)* Das Datenbanksystem stürzt nach Ausführen von Schritt 12 ab. Geben Sie die im Rahmen des Wiederanlaufs erzeugten CLRs (*compensation log records*) auf Basis logischer Protokollierung an.

[#6', T_1 , P_A , $A -= 130$, #6, #1]

[#5', T_3 , -1-1, #5, #0]

[#1', T_1 , -1-1, #1, #0]





Aufgabe 2 Historien (8 Punkte)

Gegeben seien die folgenden Konfliktoperationen:

- 1) $w_1(x) < r_3(x)$
- 2) $w_2(x) < w_1(x)$
- 3) $w_2(x) < r_3(x)$
- 4) $w_2(y) < r_3(y)$



Weiterhin ist die Reihenfolge der Commits wie folgt festgelegt:

$$c_2 < c_1 < c_3$$

Außerdem ist bekannt, dass die Commits die letzten Schritte der Historie sind.

Geben Sie für jede Eigenschaft an, ob sie von der Historie erfüllt wird und begründen Sie kurz.

a)* Serialisierbar (SR)



ja, da Serial.graph keine Zyklen hat

b)* Rücksetzbar (RC)



ja, da laut 1) 3) 4) sowohl 1 als 2 von 3 committen müssen

c)* Vermeidet kaskadierendes Zurücksetzen (ACA)



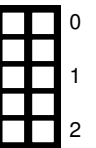
nein, da der Commit von 1, 2 nicht vor dem von 3 liegt

d)* Strikt (ST)



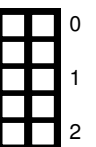
nein, da wäre strikter als ACA
=> nicht erfüllt

e)* Geben Sie eine alternative Reihenfolge der Commits an, die nichts an den Eigenschaften der Historie ändert. Falls dies nicht möglich ist, begründen Sie, wieso.



c_1, c_2, c_3

f)* Geben Sie eine alternative Reihenfolge der Commits an, die bewirkt, dass die Historie keine der 4 Eigenschaften (SR, RC, ACA, ST) erfüllt. Falls dies nicht möglich ist, begründen Sie, wieso.



SR kann zB nicht erfüllt werden da
Commits keinen Einfluss darauf
haben





Aufgabe 3 Sicherheit (9 Punkte)

Gegeben sei die Tabelle PrAbgabe mit einigen Beispielwerten. Die Tabelle speichert für alle Studenten, wann Sie Ihre Prüfung hochgeladen haben.

PrAbgabe

MatrNr	Name	Klausur	Zeit
0365 0815	Anna	ERDB	1596015015
0366 1234	Alex	ERDB	1596015384
0367 0000	Thuy	ERDB	1596016290
:	:	:	:

Es gibt eine Website, in der sich nach dem Hochladen der Abgabe einsehen lässt, ob dies rechtzeitig geschehen ist. Leider hat das Entwicklungsteam vergessen, die Benutzereingabe auf SQL-Injections zu prüfen.

Die im Formular genutzte Anfrage lautet:

`SELECT Zeit FROM PrAbgabe WHERE MatrNr={Benutzereingabe} AND Klausur='ERDB';`

Schreiben Sie folgende SQL-Injections für das Feld {Benutzereingabe}:

0	
1	
2	
3	
4	
5	
6	

a)* Sie haben verpasst, die ERDB-Klausur rechtzeitig hochzuladen, und kennen keinen validen Wert für die Abgabezeit. Schreiben Sie eine SQL-Injection, um bei Ihrer Matrikelnummer (MatrNr) den Wert 60 von der Zeit (Zeit) abzuziehen.

*O; update PrAbgabe
set Zeit=Zeit-60
where MatrNr='x'
(ERDB gegeben von Rest d. Anfrage)*

0	
1	
2	
3	

b)* Die Manipulation ist aufgefallen und Sie verwischen Ihre Spuren. Schreiben Sie eine SQL-Injection, um die gesamte Tabelle zu leeren, ohne die gesamte Tabelle zu löschen.

O; truncate table PrAbgabe





Aufgabe 4 Datalog (13 Punkte)

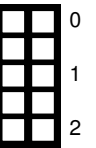
Gegeben seien die Fakten `strecke` und `linie` eines Kursbuches (hier nur in Ausschnitten dargestellt):
Die KBS identifiziert eine Strecke eindeutig.

```
%%strecke(KBS,Start,Ziel,Distanz)
strecke(900,münchen,nürnberg,171).
strecke(981,münchen,ugsburg,62).
...
%%linie(KBS,Gattung,Dauer)
linie(900,re,90).
linie(900,ice,60).
linie(981,re,41).
linie(981,ice,20).
linie(981,ic,30).
...
```

Schreiben Sie Datalog-Ausdrücke für folgende Prädikate:

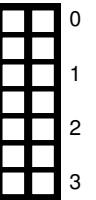
a)* `zugattungen(Start,Ziel,Gattung)`, das für jede Strecke die verfügbaren Zugattungen ausgibt.

— ' — : — `strecke(KBS,S,Z,-),linie(KBS,G,-)`



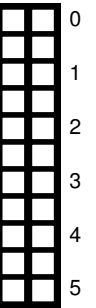
b)* `bunt(KBS)`, das alle Strecken ausgibt, auf denen mehr als eine Zugattung fährt.

`Sum(K):-linie(K,GA,-),linie(K,GB),GA\=GB`



c)* `schnellster(KBS,Gattung)`, das für jede Strecke die schnellste Zugattung ausgibt.

2

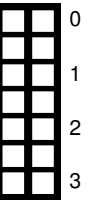


d)* Gegeben seien die folgenden Datalog-Prädikate:

```
erreichbar(V,N) :- strecke(V,N).
erreichbar(V,N) :- erreichbar(V,X),strecke(X,N).
nichtErreichbar(V,NV) :- not(erreichbar(V,NV)).
```

Begründen Sie stichpunktartig für jede erzeugte Relation (`erreichbar`, `nichtErreichbar`), ob sie sicher ist.

e : sicher, alle Variablen im Ruuff
durchen wieder out
ne : — ' —





Aufgabe 5 Fragmentierung (10 Punkte)

Gegeben sei folgende Relation Abitur mit Schlüssel ID:

CREATE TABLE Abitur(ID integer primary key, Schuelername text, Abiturnote float, Schule text);

Für eine verteilte Datenbank soll die Tabelle geeignet fragmentiert werden. Ziel ist, die Schüler-Namen mit der Schule lokal und die Abiturnoten zentral und getrennt abzuspeichern.

0					
1					
2					

a) * Die Relation soll zunächst *vertikal* fragmentiert werden. Geben Sie in SQL-92 die zwei resultierenden Relationen AbiturV1 und AbiturV2 als Hilfstabellen an.

with AbiturV1 as (

select ID, Schuelername, Schule from Abitur,
AbiturV2 as (
select ID, Abiturnote from Abitur)

0					
1					
2					

b) * Die geeignetere der beiden resultierenden Relationen soll *horizontal* mit dem Prädikat Schule='Schiller-Gymnasium' fragmentiert werden. Geben Sie in SQL-92 die zwei resultierenden Relationen AbiturH1 und AbiturH2 als Hilfstabellen an.

, AbiturH1 as (

*select * from AbiturV1 where Schule='Schiller-*
*Gymnasium'), AbiturH2 as (select **
from AbiturV1 where Schule!='Schiller-Gymn.')

0					
1					
2					
3					

c) Schreiben Sie eine SQL-Abfrage, die die Ursprungsrelation aus den Teilrelationen zusammensetzt.

*select **
from (AbiturH1 union AbiturH2) as V1
join AbiturV2 V2 on V1.ID = V2.ID

0					
1					
2					
3					

d) * Befüllen Sie die Ursprungs-Relation geeignet mit vier Tupeln, sodass nach *horizontaler* Fragmentierung jede Teilrelation jeweils zwei Tupel enthält.

insert into Abitur (values

← Klammern falsch?
(1, 'Max', 1.0, 'SG'),
(2, 'Karl', 1.0, 'SG'),
(3, 'Tim', 1.0, 'Goethe'),
(4, 'Johann', 1.0, 'Goethe'),

);





Aufgabe 6 Skyline (7 Punkte)

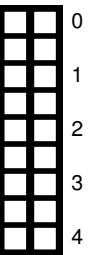
Gegeben sei die Relation Koprozessoren:

id	GFLOPS	TDP
1	2900	75
2	2400	90
3	3200	100
4	6500	180
5	4300	140
6	5200	230
7	6380	220
8	4300	110
9	3900	80

Wir betrachten die Skyline über das Minimum des Attributs TDP und das Maximum über das Attribut GFLOPS.
Offensichtlich ist das Tupel mit dem Primärschlüssel **8** in der Skyline enthalten.

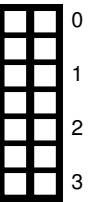
a)* Geben Sie die Primärschlüssel von allen weiteren Tupeln an, die sich abgesehen von Tupel 8 in der Skyline befinden.

4, 1, 9, 13



b)* Fügen Sie nun mittels SQL-92 ein neues Tupel ein, welches Tupel 8 dominiert, alle weiteren Zuordnungen zur Skyline sollen allerdings unverändert bleiben.

DA insert into Koprozessoren
values(10, 4300, 105);





Aufgabe 7 Window-Functions (12 Punkte)

Betrachten Sie den folgenden Ausschnitt aus der Tabelle Infektionen mit Anzahl an Neuinfektionen (Neu) und Genesenen pro Tag und Landkreis (im Juli 2020).

Landkreis	Tag	Neu	Genesen
München Stadt	1	11	10
München Stadt	2	19	8
München Stadt	3	33	23
München Stadt	4	19	4
München Stadt	5	4	4
München Stadt	6	22	5
München Stadt	7	32	39
München Land	7	32	18
⋮	⋮	⋮	⋮

Lösen Sie nachfolgende Teilaufgaben in SQL:2003.

0	
1	
2	
3	
4	
5	

a)* Berechnen Sie mittels Window-Functions die gleitende Summe der Neuinfizierten der letzten sieben Tage pro Landkreis.

```
select *, sum (Neu) over (order by Tag
partition by Landkreis rows
between 7 preceding and current
row)
from Infektionen
```

0	
1	
2	
3	
4	

b)* Berechnen Sie mittels Window-Functions die tagesaktuelle Anzahl aller aktuell Erkrankten pro Landkreis.

```
select sum (Neu) - sum (Genesen) over
(partition by Landkreis)
from Infektionen
```

0	
1	
2	
3	

c)* Aktualisieren Sie nun ein Tupel so, dass die tagesaktuelle Anzahl der aktuell Erkrankten am siebten Tag in München Stadt um eins geringer ist.

```
update Infektionen set neu = neu - 1
where Tag = 7 and Landkreis =
'München Stadt'
```





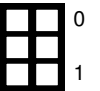
Aufgabe 8 Hauptspeicher-Datenbanken (6 Punkte)

Gegeben sei die Tabelle Professoren mit einigen Beispielwerten. Die Tabelle hat genau einen Index auf den Primärschlüssel PersNr. Ein Prädikat auf Raum ist sehr selektiv. Die Größe des jeweiligen Attributs ist in Byte direkt unter dem Namen angegeben.

Professoren						
PersNr 4B	Name 48B	Rang 2B	Raum 4B	Fakultaet 48B	Gehalt 4B	Steuerklasse 4B
2125	Sokrates	C4	226	Philosophie	85000	1
2126	Russel	C4	232	Philosophie	80000	3
2127	Kopernikus	C3	310	Physik	65000	5

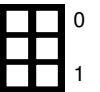
a)* Berechnen Sie die Größe eines Tupels.

114B



b) Berechnen Sie, wie viele Cachelines mindestens geladen werden müssen, um ein vollständiges Tupel der Tabelle Professoren anzuzeigen. Verwenden Sie als Cacheline-Größe 64B. Nehmen Sie dazu an, dass die Tabelle im Row-Store-Format gespeichert ist.

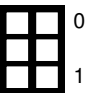
2CC = 128B



Entscheiden Sie für jede der folgenden Anfragen, ob Row- oder Column-Store für die Speicherung der Tabelle Professoren optimal wäre. Wenn der Unterschied unerheblich ist, geben Sie bitte beides an.

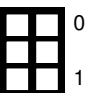
c)* SELECT * FROM Professoren;

beides



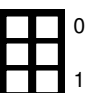
d)* SELECT * FROM Professoren WHERE Raum = '253';

column



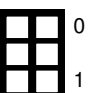
e)* SELECT * FROM Professoren WHERE PersNr = 2148;

beides



f)* SELECT Fakultaet, sum(Gehalt) GROUP BY Fakultaet;

column





Aufgabe 9 XML und JSON (10 Punkte)

0	
1	
2	
3	
4	
5	
6	
7	

a)* Gegeben sei folgende XQuery-Anfrage:

```
<a>
  <pruefungsgroesse> {
    for $v in doc('klausur.xml')/uni/vls
    let $c := count(doc('klausur.xml')/uni/prs/pr[@VNr=$v/VNr/text()])
    return <pruefung fach = "{$v/name/text()}" studenten = "{$c}"/>
  } </pruefungsgroesse>
  <studentenlast> {
    for $s in doc('klausur.xml')/uni/sts
    let $c := count(doc('klausur.xml')/uni/prs/pr[@MNr=$s/MNr/text()])
    return <person name = "{$s/name/text()}" anzahl = "{$c}" />
  } </studentenlast>
</a>
```

uni
vls
VNr
name
prs
pr@VNr
MNr
sts
sname
MNr

Erstellen Sie ein zu folgender Ausgabe passendes XML-Dokument klausur.xml:

```
<a>
  <pruefungsgroesse>
    <pruefung fach="ERDB" studenten="1"/>
    <pruefung fach="GDB" studenten="2"/>
  </pruefungsgroesse>
  <studentenlast>
    <person name="Josef" anzahl="1"/>
    <person name="Max" anzahl="2"/>
  </studentenlast>
</a>
```

```
<uni>
  <vls>
    <v>
      <VNr>1</VNr>
      <name>ERDB</name>
    </v>
    <v>
      <VNr>2</VNr>
      <name>GDB</name>
    </v>
  </vls>
  <sts>
    <s>
      <MNr>1</MNr>
      <name>Josef</name>
    </s>
    <s>
      <MNr>2</MNr>
      <name>Max</name>
    </s>
  </sts>
  <prs>
    <pr VNr=1 MNr=2/>
    <pr VNr=2 MNr=2/>
    <pr VNr=2 MNr=1/>
  </prs>
</uni>
```

0	
1	
2	
3	

b)* Gegeben sei folgende SQL-Anfrage: select doc->'Name' as Name, doc->'kennt'->1 as kennt from uni_json;

Das Ergebnis sei wie folgt:

Name	kennt
Max	Josef

Vervollständigen Sie folgendes insert-Statement durch einen passenden JSON-Ausdruck:

insert into uni_json (name, doc) values ('uni','

```
{
  'Name': 'Max',
  'kennt': ['Josef']
}
```

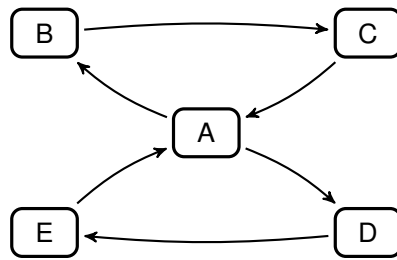
');





Aufgabe 10 PageRank (8 Punkte)

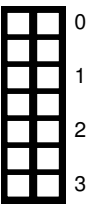
Gegeben Sei der unten stehende Webgraph für ein Netzwerk aus 5 Webseiten.



Der Startwert des Pageranks ist $1/|V|$ für jeden Knoten mit $V = 5$ und $\alpha = 0$.
Geben Sie die Lösung in folgender Form an: $A=1/5; B=1/5; C=1/5; D=1/5; E=1/5$

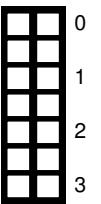
a)* Berechnen Sie den PageRank für das Netzwerk nach der 1. Iteration.

$$M = \begin{pmatrix} 0 & 0 & 1 & 0 & 1 \\ 1/2 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1/2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}; M \times p_0 = \begin{pmatrix} 2/5 \\ 1/10 \\ 1/5 \\ 1/10 \\ 1/5 \end{pmatrix} = p_1$$



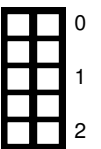
b)* Berechnen Sie den PageRank für das Netzwerk nach der 2. Iteration.

$$M \times p_1 = p_2 = \begin{pmatrix} 2/5 \\ 1/5 \\ 1/10 \\ 1/5 \\ 1/10 \end{pmatrix}$$



c)* Nehmen Sie an, der PageRank nach der ersten Iteration ist:

$$\begin{pmatrix} A = \frac{3}{10} \\ B = \frac{1}{10} \\ C = \frac{1}{5} \\ D = \frac{1}{5} \\ E = \frac{1}{5} \end{pmatrix} \leftarrow \text{v. Code = a}$$



Welche Kante müsste dem Webgraph hinzugefügt werden, damit dieser Wert korrekt wäre?

$C \rightarrow D$



This image shows a full page of blank graph paper. The grid consists of small, equal-sized squares formed by thin gray lines. There are 20 columns and 20 rows of squares, creating a total area of 400 small squares. The grid covers the entire page except for a narrow white border around the edges.