

Gruppe A

Bitte tragen Sie **sofort** und **leserlich** Namen, Studienkennzahl und Matrikelnummer ein und legen Sie Ihren Studentenausweis bereit.

PRÜFUNG AUS DATENBANKSYSTEME VO 181.146			24. 03. 2006
Kennnr.	Matrikelnr.	Familienname	Vorname

Arbeitszeit: 120 Minuten. Aufgaben sind auf den Angabeblättern zu lösen; Zusatzblätter werden nicht gewertet.

Aufgabe 1:

(20)

Die Schiverbände von Österreich, Italien und Deutschland beschließen, gemeinsam eine verteilte Datenbank aller Schiennläufer dieser drei Länder zu erstellen. In dieser Datenbank ist folgende Laeuer-Tabelle enthalten:

Laeuer(id, name, land, geb, pAkt, pMax),

Für diese Tabelle wird folgende (vertikale) Fragmentierung festgelegt:

LaeuerDaten(id, name, land, geb)

LaeuerRangliste(id, pAkt, pMax) (d.h.: aktuelle Punkte und maximal jemals erreichte Punkte)

Die Tabelle LaeuerDaten(id, name, land, geb) wird nach dem Attribut "land" (horizontal) weiter fragmentiert in die drei Tabellen LaeuerDatenAUT, LaeuerDatenITA und LaeuerDatenGER. Es ist die Anfrage

select name, geb, pAkt

from Laeuer

where geb > 1985 and pAkt > 500

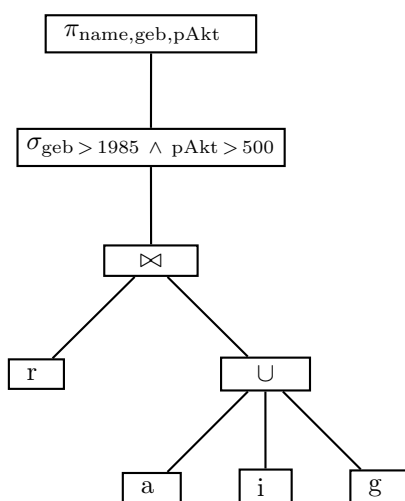
auszuführen (d.h. Informationen über junge Läufer mit vielen Punkten).

(a) Zeichnen Sie ins erste Kästchen den Operator-Baum für die kanonische Übersetzung. Verwenden Sie für die 4 Fragmente der Laeuer-Tabelle folgende Abkürzungen: **r** (LaeuerRangliste), **a** (LaeuerDatenAUT), **i** (LaeuerDatenITA) und **g** (LaeuerDatenGER).

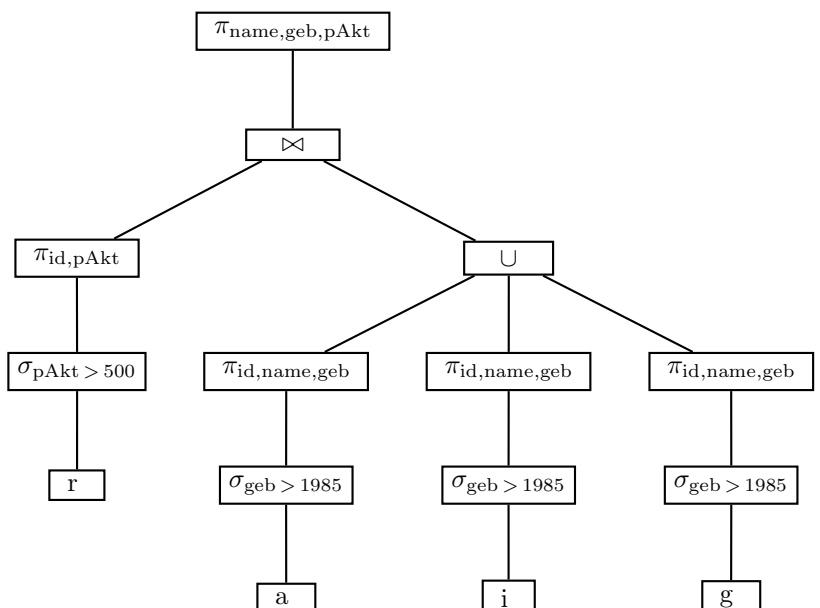
(b) Zeichnen Sie ins zweite Kästchen den Operator-Baum für den optimierten algebraischen Ausdruck. Wenden Sie für die Optimierung folgende Heuristiken an:

- Selektionen so weit wie möglich nach unten verschieben,
- Attribute, die nicht mehr benötigt werden, möglichst früh wegprojizieren.

(a) Kanonische Übersetzung:



(b) Optimierter Ausdruck:



Aufgabe 2:

Gegeben ist die folgende Historie von Transaktionen:

Schritt	T_1	T_2	T_3	Log: [LSN, TA, PageID, Redo, Undo, PrevLSN] or ⟨LSN, TA, PageID, Redo, PrevLSN, UndoNextLSN⟩
1	BOT			[#1, T_1 , BOT, 0]
2		BOT		[#2, T_2 , BOT, 0]
3			BOT	[#3, T_3 , BOT, 0]
4	$r(A, a_1)$			
5		$r(A, a_2)$		
6			$r(A, a_3)$	
7		$r(B, b_2)$		
8		$w(A, a_2 + b_2)$		[#4, T_2 , P_A , A+=150, A-=150, #2]
9	$w(A, a_1 + 100)$			[#5, T_1 , P_A , A-=50, A+=50, #1]
10		$w(B, b_2 - a_2)$		[#6, T_2 , P_B , B-=100, B+=100, #4]
11		$w(C, 3 * a_2)$		[#7, T_2 , P_C , C+=100, C-=100, #6]
12			$w(A, 3 * a_3)$	[#8, T_3 , P_A , A+=100, A-=100, #3]
13			$w(C, 2 * a_3)$	[#9, T_3 , P_C , C-=100, C+=100, #8]
14	commit			[#10, T_1 , commit, #5]
15		commit		[#11, T_2 , commit, #7]
16			rollback ...	[#12, T_3 , rollback, #9]
17				⟨#13, T_3 , P_C , C+=100, #12, #8⟩
18				⟨#14, T_3 , P_A , A-=100, #13, #3⟩
19				⟨#15, T_3 , (BOT), #14, 0⟩

(a) Nehmen Sie an, dass in Zeile 16 auch die Transaktion T_3 mittels commit beendet wird. Ist die resultierende Historie serialisierbar?

☐ ja ☒ nein

Wenn ja, in Reihenfolge $T - \dots$ vor $T - \dots$ vor $T - \dots$.

Wenn nein: die Historie wird durch das Streichen von zumindest 2 Operationen serialisierbar.

(b) Führen Sie nun – unabhängig davon, ob die Historie serialisierbar ist – in Zeile 16 ein *rollback* für T_3 durch.

Zu Beginn ist der relevante Datenbestand in der Datenbank $A = 100$, $B = 150$, und $C = 200$. Tragen Sie nun das Recovery-Log zu dieser Historie (mit rollback von T_3 in Zeile 16) in die rechte Spalte der obigen Tabelle ein. Dabei sind Undo/Redo-Einträge *relativ* zum Datenbestand anzugeben. Geben Sie in den Zeilen 16 ff. die Log-Einträge für die Recovery an.

Die Werte von A , B und C nach dem rollback von T_3 sind $A : 200$; $B : 50$; $C : 300$

Aufgabe 3:

(20)

Kreuzen Sie an, ob die folgenden Aussagen wahr oder falsch sind.

1. Beim Discretionary Access Protocol (DAC) kann man jederzeit einen Sicherheitsverantwortlichen festlegen, der die Zugriffsrechte auf die Daten zentral verwaltet. wahr ☐ falsch ☒
2. Polymorphismus und Polyinstanziierung sind zwei objektrelationale Erweiterungen in SQL:1999. wahr ☐ falsch ☒
3. Die Datalog-Erweiterungen in SQL:1999 ermöglichen unter Umständen eine effizientere Realisierung von n:m Beziehungen. wahr ☐ falsch ☒
4. Angenommen der Abhängigkeitsgraph eines Datalog-Programms ist zyklensfrei; dann lässt sich die intensionale Datenbank (IDB) auch ausschließlich mittels relationaler Algebra ausdrücken. wahr ☒ falsch ☐
5. Multilevel-Datenbanken wurden unter anderem deswegen entwickelt, um feinere Sperrgranulate (also z.B. Satz-weise und nicht bloß Seiten-weise) zu ermöglichen. wahr ☐ falsch ☒
6. Die Anzahl der Zyklen im Wartegraphen entspricht immer der Anzahl der Transaktionen, die zurückgesetzt werden müssen, um ein Deadlock aufzulösen. wahr ☐ falsch ☒
7. Angenommen in einer objektorientierten Datenbank enthält eine Klasse eine Methode, mit der eine mathematische Funktion realisiert wird. Dann lässt sich mit Hilfe des Schlüsselworts "inverse" auf einfache Weise die Umkehrfunktion darstellen. wahr ☐ falsch ☒
8. "Spätes Binden" wird in einer objektorientierten Datenbank unter anderem dafür verwendet, um alle Attribute eines Obertyps an den Untertyp zu vererben. wahr ☐ falsch ☒
9. Nehmen Sie an, dass beim Zweiphasen-Commit-Protokoll einer der Agenten abstürzt. Dann kann der Koordinator trotzdem entscheiden, ob eine Transaktion erfolgreich war oder nicht. wahr ☒ falsch ☐
10. Betrachten Sie zwei Relationen $R(AB)$ und $S(BC)$. Das Ergebnis des Ausdrucks $(\pi_B(R) \cup \pi_B(S)) \bowtie S$ enthält ausschließlich Tupel, die auch in S enthalten sind. wahr ☒ falsch ☐

(Pro korrekter Antwort 2 Punkte, **pro inkorrektter Antwort -2 Punkte**, insgesamt mindestens 0 Punkte)

Aufgabe 4:

(24)

Die Datenbank eines internationalen Konzerns enthalte Informationen über die Weiterbildung der Mitarbeiter. Folgende Relationen sind in dieser Datenbank enthalten:

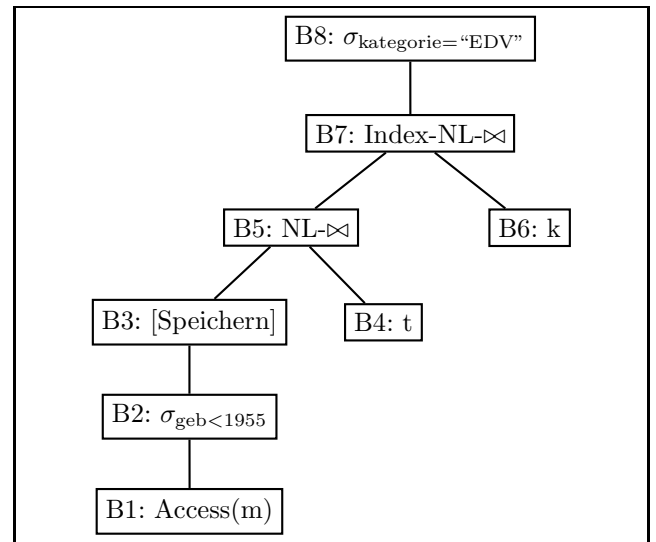
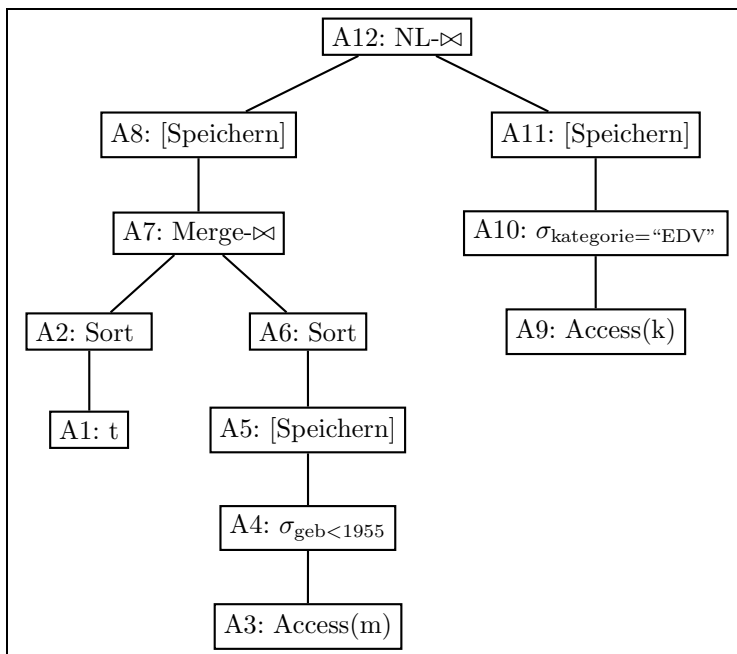
Mitarbeiter(persNr, name, gehalt, geb) (kurz m),
Kurs(kNr, titel, preis, kategorie) (kurz k) und
Teilnahme(persNr, kNr, Datum) (kurz t).

Nehmen Sie an, dass $|m| = 50000$, $|k| = 2000$, und $|t| = 500000$. Für die durchschnittlichen Tupelgrößen von m , k und t sind die Werte 80, 100 und 20 Bytes anzunehmen. Nehmen Sie weiters an, dass pro Seite 1000 Bytes an Nutzinformation gespeichert werden können und dass die Hauptspeicher-Puffergröße 20 Seiten beträgt. Es ist die Anfrage

```
select *  
from Mitarbeiter m, Kurs k, Teilnahme t  
where m.persNr = t.persNr  
and k.kNr = t.kNr  
and m.geb < 1955  
and k.kategorie = 'EDV';
```

auszuführen (d.h. gesucht sind Informationen über ältere Mitarbeiter, die einen EDV-Kurs besucht haben). Es sind die Selektivitäten $Sel_{m/t} = 1/50000 = 0.00002$, $Sel_{k/t} = 1/2000 = 0.0005$, $Sel_{k.kategorie} = 0.20$ und $Sel_{m.geb < 1955} = 0.15$ anzunehmen. Für die Primärschlüssel der Relationen m und k sei jeweils ein Hash-Index vorhanden. Nehmen Sie an, dass das Auslesen eines einzelnen Tupels mit einem Hash-Index durchschnittliche Kosten von 1.2 Page I/O erfordert.

Für diese Anfrage sind die Operator-Bäume für 2 Auswertungspläne gegeben: Plan A im linken Kästchen und Plan B im rechten Kästchen. Nehmen Sie bei den Block Nested Loop Joins an, dass die äußere Relation jeweils links im Baum steht – unabhängig davon, ob dies optimal ist oder nicht.



(a) Berechnen Sie für jeden Knoten im Operatorbaum des Auswertungsplans A eine Abschätzung für die Anzahl der Tupel im Resultat, die Tupelgröße, die Anzahl der Seiten im Resultat, und die Kosten (Page I/O). Für das Sortieren und für Joinoperationen ist auch noch die passende Kostenformel anzugeben. Tragen Sie Ihre Berechnungen in folgende Tabelle ein.

Knoten# i	Anzahl Tupel T_i	Tupel- größe g_i	Anzahl Seiten b_i	Kostenformel	Kosten (Page I/O)
A1	500000	20	10000	-	0
A2	500000	20	10000	$2 * b_1 * (1 + I)$ mit $I = \log_{19}(\lceil b_1/20 \rceil) = 3$	80000
A3	50000	80	4000	-	4000
A4	7500	80	600	-	0
A5	7500	80	600	-	600
A6	7500	80	600	$2 * b_5 * (1 + I)$ mit $I = \log_{19}(\lceil b_5/20 \rceil) = 2$	3600
A7	75000	100	7500	$b_2 + b_6$	10600
A8	75000	100	7500	-	7500
A9	2000	100	200	-	200
A10	400	100	40	-	0
A11	400	100	40	-	40
A12	15000	200	3000	$b_8 + 1 + \lceil b_8/18 \rceil * (b_{11} - 1)$	23764

Kosten insgesamt (Page I/O):

$$80000 + 4000 + 600 + 3600 + 10600 + 7500 + 200 + 40 + 23764 = 130304$$

(b) Berechnen Sie für jeden Knoten im Operatorbaum des Auswertungsplans B eine Abschätzung für die Anzahl der Tupel im Resultat, die Tupelgröße, die Anzahl der Seiten im Resultat, und die Kosten (Page I/O). Für die Joinoperationen ist auch noch die passende Kostenformel anzugeben. Tragen Sie Ihre Berechnungen in folgende Tabelle ein.

Knoten# i	Anzahl Tupel T_i	Tupel- größe g_i	Anzahl Seiten b_i	Kostenformel	Kosten (Page I/O)
B1	50000	80	4000	-	4000
B2	7500	80	600	-	0
B3	7500	80	600	-	600
B4	500000	20	10000	-	0
B5	75000	100	7500	$b_3 + 1 + \lceil b_3/18 \rceil * (b_4 - 1)$	340567
B6	2000	100	200	-	0
B7	75000	200	15000	$b_5 + 1.2 * T_5$	97500
B8	15000	200	3000	-	0

Kosten insgesamt (Page I/O):

$$4000 + 600 + 340567 + 97500 = 442667 \dots\dots\dots$$

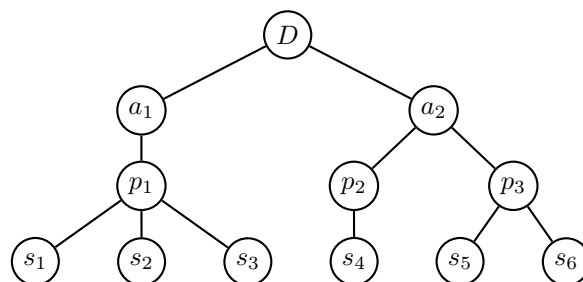
(c) Wie sehen im Auswertungsplan A die Berechnungen für den Knoten A12 aus, wenn der Block Nested Loop Join durch einen Hash Join ersetzt wird?

Knoten#	Anzahl Tupel	Tupelgröße	Anzahl Seiten	Kostenformel	Kosten
A12	15000	200	3000	$3 * (b_8 + b_{11})$	22620

Aufgabe 5:

(12)

Multi-Granularity Locking. Betrachten Sie folgende Datenbasis-Hierarchie.



Beantworten Sie, welche der folgenden Sequenzen von Anforderungen von Schlössern (bei zwei Transaktionen T_1 und T_2) zu Blockierungen bzw. Deadlocks führen. (Hier bedeutet (T_i, x, L) , daß Transaktion T_i versucht, Knoten x in der Hierarchie mit einem Schloß vom Typ L zu belegen.)

- $(T_1, D, IX), (T_2, D, IS), (T_2, a_1, IS), (T_1, a_1, IX), (T_1, p_1, IX), (T_2, p_1, S), (T_1, s_3, X)$:
 Blockierung: ja ☒ nein ☐ Deadlock: ja ☐ nein ☒
- $(T_1, D, IX), (T_2, D, IX), (T_1, a_1, IX), (T_2, a_2, IX), (T_1, p_1, X), (T_2, p_2, X), (T_1, a_2, S), (T_2, a_1, S)$:
 Blockierung: ja ☒ nein ☐ Deadlock: ja ☒ nein ☐
- $(T_1, D, IX), (T_2, D, IX), (T_1, a_1, IX), (T_2, a_2, IX), (T_2, p_3, X), (T_1, a_2, S), (T_2, a_1, IX), (T_2, p_1, IX), (T_2, s_2, X)$:
 Blockierung: ja ☒ nein ☐ Deadlock: ja ☐ nein ☒

Gesamtpunkte: 100