

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			4. 5. 2006
Kennnr.	Matrikelnr.	Familienname	Vorname

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

Aufgabe 1:

(20)

Ein Unternehmen mit Standorten in Wien und Linz verwaltet die Mitarbeiter-Daten in einer verteilten Datenbank. Folgende Tabelle ist in dieser Datenbank enthalten:

Mitarbeiter(id, name, standort, geb, gehalt, projekt, funktion),

Für diese Tabelle wird folgende (vertikale) Fragmentierung festgelegt:

MitarbeiterVerwaltung(id, name, standort, geb, gehalt)

MitarbeiterProjekt(id, standort, projekt, funktion)

Die beiden Tabellen werden nach dem Attribut "standort" (horizontal) weiter fragmentiert in die vier Tabellen MitarbeiterVerwaltungWien, MitarbeiterVerwaltungLinz, MitarbeiterProjektWien, MitarbeiterProjektLinz.

Es ist folgende Anfrage auszuführen (d.h. Informationen über Projektleiter mit relativ niedrigem Gehalt):

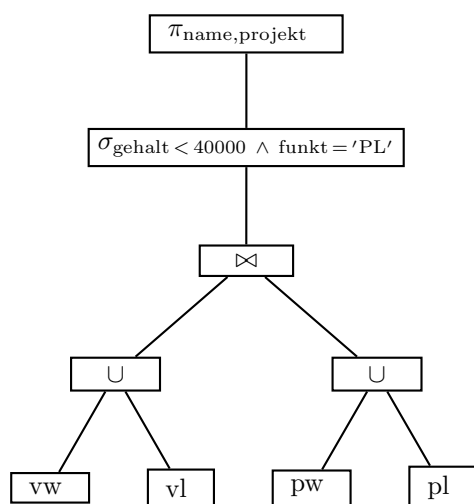
```
select name, projekt
from Mitarbeiter
where gehalt < 40000
and funktion = 'PL';
```

(a) Zeichnen Sie ins erste Kästchen den Operator-Baum für die kanonische Übersetzung. Verwenden Sie für die 4 Fragmente der Mitarbeiter-Tabelle folgende Abkürzungen: **vw** (Mitarbeiter**V**erwaltung**W**ien), **vl** (Mitarbeiter**V**erwaltung**L**inz), **pw** (Mitarbeiter**P**rojekt**W**ien) und **pl** (Mitarbeiter**P**rojekt**L**inz).

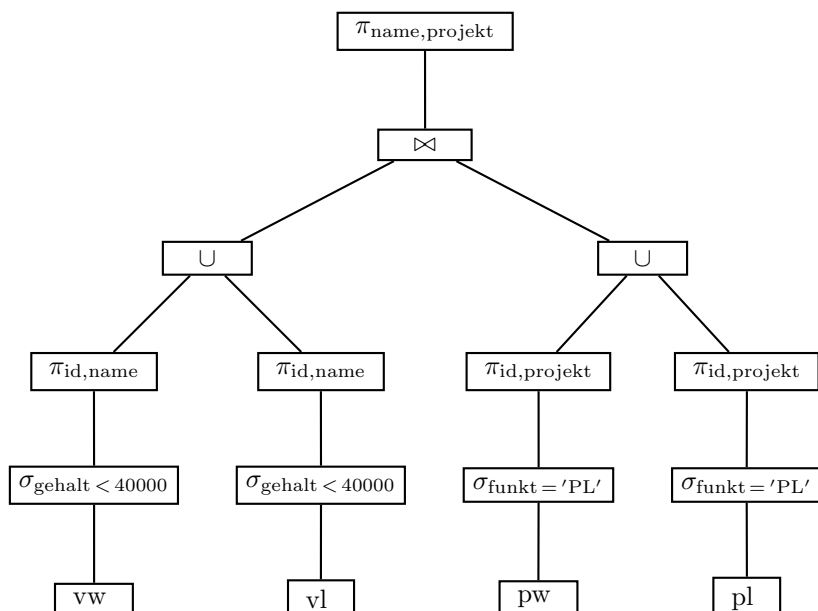
(b) Zeichnen Sie ins zweite Kästchen den Operator-Baum für den optimierten algebraischen Ausdruck. Wenden Sie für die Optimierung folgende Heuristiken an:

- 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:

(24)

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	$w(A, a_1 + 100)$			[#4, T_1 , P_A , A+=100, A-=100, #1]
7			$r(B, b_3)$	
8		$r(B, b_2)$		
9		$w(B, a_2 + 200)$		[#5, T_2 , P_B , B+=100, B-=100, #2]
10	$w(B, 2 * a_1)$			[#6, T_1 , P_B , B-=100, B+=100, #4]
11			$w(C, 2 * b_3)$	[#7, T_3 , P_C , C+=150, C-=150, #3]
12	$w(C, 2 * a_1)$			[#8, T_1 , P_C , C-=200, C+=200, #6]
13		$w(A, 2 * b_2)$		[#9, T_2 , P_A , A+=200, A-=200, #5]
14	commit			[#10, T_1 , commit, #8]
15				⟨#11, T_2 , P_A , A-=200, #9, #5⟩
16				⟨#12, T_3 , P_C , C-=150, #7, #3⟩
17				⟨#13, T_2 , P_B , B-=100, #11, #2⟩
18				⟨#14, T_3 , (BOT), #12, 0⟩
19				⟨#15, T_2 , (BOT), #13, 0⟩

(a) Nehmen Sie an, dass unmittelbar nach Zeile 14 zunächst die Transaktion T_2 und dann 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 1 Operationen serialisierbar.

(b) Nehmen Sie wie in Frage (a) an, dass unmittelbar nach Zeile 14 zunächst die Transaktion T_2 und dann die Transaktion T_3 mittels commit beendet wird. Wie viele Operationen muss man mindestens streichen, damit die resultierende Historie strikt wird? ☐ 0 ☐ 1 ☐ 2 ☒ 3

(c) Zu Beginn ist der relevante Datenbestand in der Datenbank $A = 100$, $B = 200$, und $C = 250$. Nehmen Sie an, dass unmittelbar nach Zeile 14 ein System-Absturz mit anschließendem Wiederanlauf passiert. Tragen Sie die entsprechenden Log-Einträge zu dieser Historie in die rechte Spalte der obigen Tabelle ein. Dabei sind Undo/Redo-Einträge *relativ* zum Datenbestand anzugeben. Geben Sie in den Zeilen 15ff. die Log-Einträge für die Recovery an.

Die Werte von A , B und C nach Abschluss der Recovery sind $A : 200 \dots$; $B : 200 \dots$; $C : 200 \dots$.

Aufgabe 3:

(20)

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

1. Geschachtelte Relationen in objektrelationalen Datenbanken erlauben unter Umständen eine effizientere Auswertung von Anfragen bezüglich 1:n-Relationen. wahr ☒ falsch ☐
2. Bei Erweiterung eines rein relationalen Datenbanksystems um die 2 objektrelationalen Features "Objekt-Identität" und "Referenzen auf Objekte" lassen sich Hilfstabellen für n:m-Relationen vermeiden. wahr ☐ falsch ☒
3. Eine Relation R sei an 4 Netzwerk-Knoten materialisiert mit den Gewichten 8, 5, 3 und 9. Dann sind $Q_r(R) = 12$ und $Q_w(R) = 13$ gültige Lese- bzw. Schreib-Quoten. wahr ☐ falsch ☒
4. Betrachten Sie drei Relationen $R(AB)$, $S(AB)$ und $T(BC)$. Dann gilt auf jeden Fall folgende Gleichheit: $(\pi_B(R) \cap \pi_B(S)) \bowtie T = (\pi_B(R) \bowtie T) \bowtie \pi_B(S) \bowtie T$. wahr ☒ falsch ☐
5. Betrachten Sie wiederum drei Relationen $R(AB)$, $S(AB)$ und $T(BC)$. Dann gilt auf jeden Fall folgende Gleichheit: $(\pi_B(R) \cap \pi_B(S)) \bowtie T = \pi_{BC}(R \bowtie S \bowtie T)$. wahr ☐ falsch ☒
6. Die Fragmentierung der Tabelle Mitarbeiter in die Fragmente MitarbeiterVerwaltung und MitarbeiterProjekt (in Aufgabe 1) ist vollständig und disjunkt. wahr ☐ falsch ☒
7. Die Fragmentierung der Tabelle MitarbeiterVerwaltung in die Fragmente MitarbeiterVerwaltungWien und MitarbeiterVerwaltungLinz (in Aufgabe 1) ist vollständig aber nicht disjunkt. wahr ☐ falsch ☒
8. Bei Verwendung von Role-based Access Control können einem Datenbank-Benutzer Zugriffsrechte erteilt werden, die er/sie bei Verwendung von Discretionary Access Control nicht bekommen könnte. wahr ☐ falsch ☒
9. Mit "Datalog mit Rekursion aber ohne Negation" lassen sich Anfragen formulieren, die man mit relationaler Algebra nicht formulieren könnte. wahr ☒ falsch ☐
10. Mit relationaler Algebra lassen sich Anfragen formulieren, die man mit "Datalog mit Rekursion aber ohne Negation" nicht formulieren könnte. wahr ☒ falsch ☐

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

Aufgabe 4:

(20)

In einer Datenbank über Tennisspieler sind folgende Relationen enthalten:

Spieler(id, sname, geb, rang) (kurz s),

Turnier(tname, land, stadt, kategorie, preis) (kurz t) und

gespielt(id, tname, jahr) (kurz g).

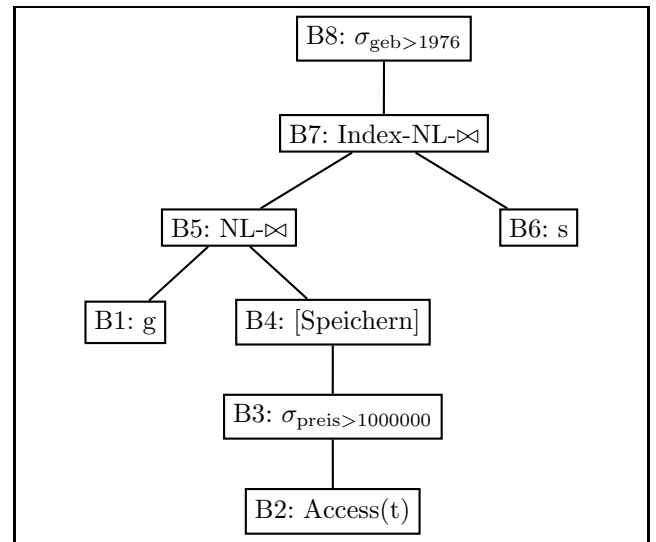
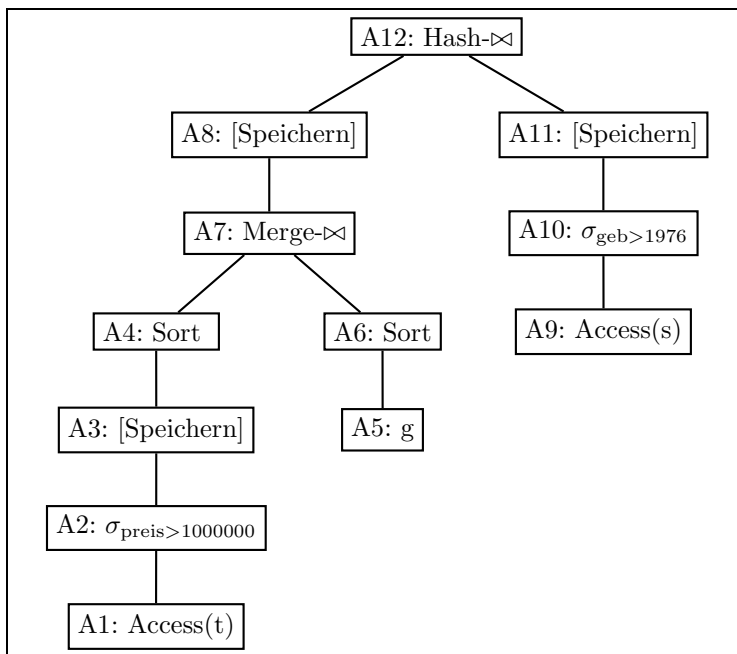
Nehmen Sie an, dass $|s| = 200000$, $|t| = 20000$, und $|g| = 1000000$. Für die durchschnittlichen Tupelgrößen von s , t und g sind die Werte 100, 160 und 40 Bytes anzunehmen. Nehmen Sie weiters an, dass pro Seite 2000 Bytes an Nutzinformation gespeichert werden können und dass die Hauptspeicher-Puffergröße 128 Seiten beträgt. Es ist die Anfrage

```
select *
from Spieler s, Turnier t, gespielt g
where s.id = g.id
and t.tname = g.tname
and s.geb > 1976
and t.preis >= 1000000;
```

auszuführen (d.h. gesucht sind Informationen über junge Spieler, die schon einmal bei einem Turnier mit großem Preisgeld mitgespielt haben).

Nehmen Sie folgende Selektivitäten an: $Sel_{g/s} = 1/200000 = 0.000005$, $Sel_{g/t} = 1/20000 = 0.00005$, $Sel_{t.preis > 1000000} = 0.1$ und $Sel_{s.geb > 1976} = 0.2$. Für die Primärschlüssel der Relationen s und t sei jeweils ein Hash-Index vorhanden. Nehmen Sie an, dass das Auslesen eines einzelnen Tupels mit einem Hash-Index durchschnittliche Kosten von 1.5 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	20000	160	1600	-	1600
A2	2000	160	160	-	0
A3	2000	160	160	-	160
A4	2000	160	160	$2 * b_3 * (1 + I)$ mit $I = \log_{127}(\lceil b_3/128 \rceil) = 1$	640
A5	1000000	40	20000	-	0
A6	1000000 ..	40	20000	$2 * b_5 * (1 + I)$ mit $I = \log_{127}(\lceil b_5/128 \rceil) = 2$	120000
A7	100000	200	10000	$b_4 + b_6$	20160
A8	100000	200	10000	-	10000
A9	200000	100	10000	-	10000
A10	40000	100	2000	-	0
A11	40000	100	2000	-	2000
A12	20000	300	3000	$3 * (b_8 + b_{11})$	36000

Kosten insgesamt (Page I/O):

$$1600 + 160 + 640 + 120000 + 20160 + 10000 + 10000 + 2000 + 36000 = 200560$$

(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	1000000	40	20000	-	0
B2	20000	160	1600	-	1600
B3	2000	160	160	-	0
B4	2000	160	160	-	160
B5	100000	200	10000	$b_1 + 1 + \lceil b_1/126 \rceil * (b_4 - 1)$	45282
B6	200000	100	10000	-	0
B7	100000	300	15000	$b_5 + 1.5 * T_5$	160000
B8	20000	300	3000	-	0

Kosten insgesamt (Page I/O):

1600 + 160 + 45282 + 160000 = 207042

Aufgabe 5: (16)

(a) Betrachten Sie die folgende Folge von Sperranforderungen, wobei “lockS_{*i*}(O)” (bzw. “lockX_{*i*}(O)”) bedeutet, dass die Transaktion *T_i* eine Lesesperre (bzw. eine Schreibsperr) auf das Datenobjekt O anfordert: lockX₁(A) vor lockX₂(B) vor lockS₃(C) vor lockS₄(D) vor lockS₄(B) vor lockS₁(D) vor lockX₁(C) vor lockS₃(B) vor lockS₂(A).

Zeichnen Sie ins erste Kästchen den Wartegraphen unter der Annahme, dass zum momentanen Zeitpunkt keine der erhaltenen Sperren wieder zurückgegeben wurde. Verwenden Sie dabei folgende Konvention: eine Kante *T_i* → *T_j* bedeutet “Transaktion *T_i* wartet auf die Freigabe einer Sperre durch *T_j*” (vgl. VO-Folien bzw. Kemper-Buch).

(b) Eine Historie sei gegeben durch folgende Folge von Elementaroperationen: *r*₁(A) vor *r*₂(B) vor *w*₁(A) vor *w*₂(B) vor *w*₃(C) vor *r*₄(D) vor *r*₄(C) vor *r*₁(C) vor *r*₂(D) vor *r*₂(A) vor *w*₄(D) vor *c*₁ vor *c*₂ vor *c*₃ vor *c*₄.

Zeichnen Sie ins zweite Kästchen den Serialisierbarkeitsgraphen für die Transaktionen *T*₁, *T*₂, *T*₃ und *T*₄. Verwenden Sie dabei folgende Konvention: eine Kante *T_i* → *T_j* bedeutet, dass in einer äquivalenten seriellen Historie die “Transaktion *T_i* vor *T_j*” ausgeführt werden muss (vgl. VO-Folien bzw. Kemper-Buch).

(a) Wartegraph:

(b) Serialisierbarkeitsgraph:

(c) Geben Sie für den Serialisierbarkeitsgraphen aus (b) eine mögliche topologische Sortierung an:

T 3 vor *T* 1 vor *T* 2 vor *T* 4