Gruppe A

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

PRÜFUNG AUS DATENBANKSYSTEME **VU 184.686**			08. 03. 2016
Kennnr.	Matrikelnr.	Familienname	Vorname

Arbeitszeit: 100 Minuten. Aufgaben sind auf den Angabeblättern zu lösen; Zusatzblätter werden nicht gewertet.

Aufgabe 1: (15)

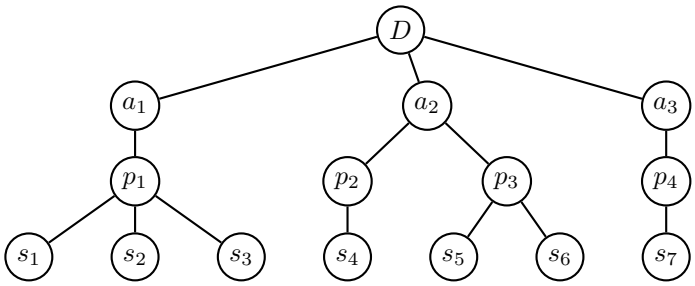
Kreuzen Sie an, ob die folgenden Aussagen wahr oder falsch sind.

1. Nehmen Sie an, dass eine Relation R aus 5000 Seiten besteht und dass die Puffergröße 100 beträgt. Nehmen Sie weiters an, dass eine Hash-Funktion eine gleichmäßige Verteilung auf die gewünschte Anzahl von Buckets liefert. Dann ist bei einem Hash-Join von R mit einer beliebigen Relation S in der Build-Phase möglicherweise ein Re-hashing von S erforderlich. wahr ◯ falsch ⊗
2. Wenn in einem Auswertungsplan alle Block Nested Loop Joins durch Sort Merge Joins ersetzt werden, dann kann dies unter Umständen zu einer Verringerung der Anzahl der Tupel in den Zwischenergebnissen führen. wahr ◯ falsch ⊗
3. Betrachten Sie zwei Relationen $R(ABC)$ und $S(BDE)$. Dann gilt in jedem Fall folgende Teilmengenbeziehung: $((\pi_B(R) \cup \pi_B(S)) \bowtie R) \subseteq R$. wahr ⊗ falsch ◯
4. Betrachten Sie drei Relationen $R(AB)$, $S(AC)$ und $T(BC)$. Dann gilt in jedem Fall folgende Gleichheit: $(R \bowtie S) \bowtie T = (R \bowtie S) \cap (R \bowtie T)$ wahr ⊗ falsch ◯
5. Betrachten Sie eine Datenbank mit der Tabelle Mannschaft (<u>Land</u>, Gruppe, AnzahlTitel, AnzahlTeilnahmen) der 24 Mannschaften, die bei der Euro 2016 dabei sind. Diese Tabelle sei mit Hilfe der 6 möglichen Werte des Attributs "Gruppe" in Fragmente zerlegt. Die resultierende Fragmentierung ist horizontal, vollständig und redundanzfrei. wahr ⊗ falsch ◯
6. Die Historie $r_1(A)$, $r_2(B)$, $w_2(B)$, $w_1(A)$, $w_1(C)$, $w_2(C)$, c_2, c_1 ist zwar beim Zwei-Phasen Sperrprotokoll jedoch nicht beim strengen Zwei-Phasen Sperrprotokoll möglich. wahr ⊗ falsch ◯
7. Die Historie $r_1(C)$, $r_2(D)$, $w_2(D)$, $w_1(D)$, $w_2(C)$, c_1, c_2 vermeidet kaskadierendes Rücksetzen und ist nicht serialisierbar. wahr ⊗ falsch ◯
8. Sowohl beim isolation level "Read committed" als auch beim isolation level "Read uncommitted" kann es zu einem Deadlock kommen. wahr ⊗ falsch ◯
9. Die Verwendung eines RAID-Systems garantiert die Eigenschaften *R*edundancy, *A*tomicity, *I*solation, und *D*urability. wahr ◯ falsch ⊗
10. Nehmen Sie an, dass eine relationale Datenbank um die beiden objektrelationalen Features "Objektidentität" und "Referenzen" erweitert wurde. Dann lassen sich $n{:}\,m$ Beziehungen auf jeden Fall – im Gegensatz zu rein relationalen Datenbanken – ohne Hilfstabelle realisieren. wahr ◯ falsch ⊗

(Pro korrekter Antwort 1.5 Punkte, **pro inkorrekter Antwort -1.5 Punkte**, pro nicht beantworteter Frage 0 Punkte, für die gesamte Aufgabe mindestens 0 Punkte)

Aufgabe 2: (12)

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



Beantworten Sie, welche der folgenden geplanten Sequenzen von Sperr-Anforderungen (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 einer Sperre vom Typ L zu belegen.)

Hinweis: Unter Umständen werden nicht alle Sperren dieser Sequenzen auch tatsächlich angefordert, d.h.: Im Falle einer Blockierung einer Transaktion werden die weiter hinten liegenden Sperr-Anforderungen dieser Transaktion gar nicht mehr durchgeführt.

1. (T_1, D, IX), (T_2, D, IX), (T_2, a_2, IX), (T_1, a_1, X), (T_2, p_2, X), (T_1, a_2, IX), (T_1, p_3, X):
   Blockierung: ja ◯ nein ⊗ Deadlock: ja ◯ nein ⊗
2. (T_1, D, IS), (T_2, D, IX), (T_1, a_3, IS), (T_2, a_1, X), (T_1, p_4, S), (T_2, a_3, X), (T_1, a_1, IS), (T_1, p_1, S):
   Blockierung: ja ⊗ nein ◯ Deadlock: ja ⊗ nein ◯
3. (T_1, D, IS), (T_2, D, IX), (T_1, a_1, IS), (T_1, p_1, IS), (T_2, a_3, X), (T_2, a_1, IX), (T_1, s_2, S), (T_2, p_1, IX), (T_2, s_3, X):
   Blockierung: ja ◯ nein ⊗ Deadlock: ja ◯ nein ⊗
4. (T_1, D, IX), (T_2, D, IS), (T_2, a_2, IS), (T_1, a_2, IX), (T_2, p_3, S), (T_1, p_2, X), (T_1, p_3, IX), (T_1, s_5, X):
   Blockierung: ja ⊗ nein ◯ Deadlock: ja ◯ nein ⊗

Aufgabe 3: (18)

Eine Datenbank von Tennisspielern enthält folgende Relationen:

Spieler(SpielerNr, Name, Land, Geburtsdatum, Platzierung) (kurz s),
Turnier(TurnierNr, Name, Jahr, Stadt, Datum) (kurz t) und
gewinnt(SpielerNr, TurnierNr, Platz, Preisgeld) (kurz g).

Nehmen Sie an, dass $|s| = 40000$, $|t| = 50000$ und $|g| = 500000$. Für die durchschnittlichen Tupelgrößen von s, t und g sind die Werte 100, 120 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

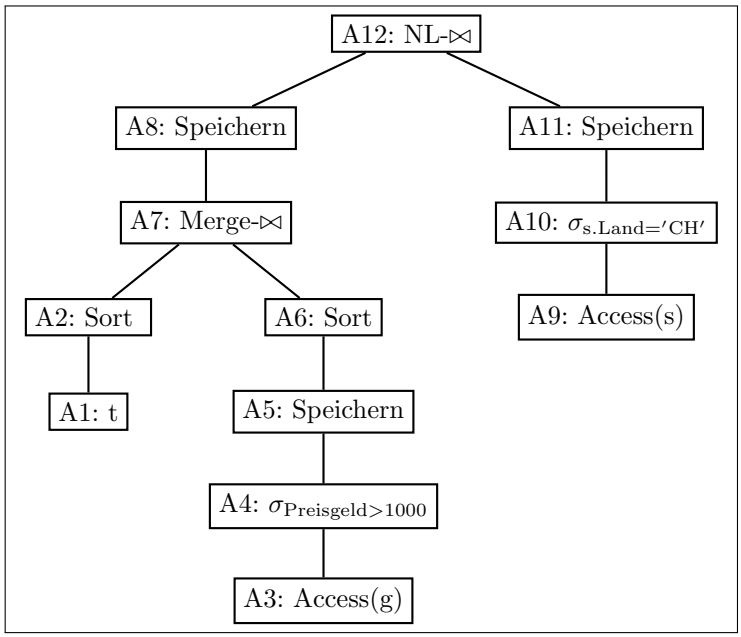
```
select *
from Spieler s, Turnier t, gewinnt g
where s.SpielerNr = g.SpielerNr
and t.TurnierNr = g.TurnierNr
and s.Land = 'CH'
and g.Preisgeld > 1000;
```

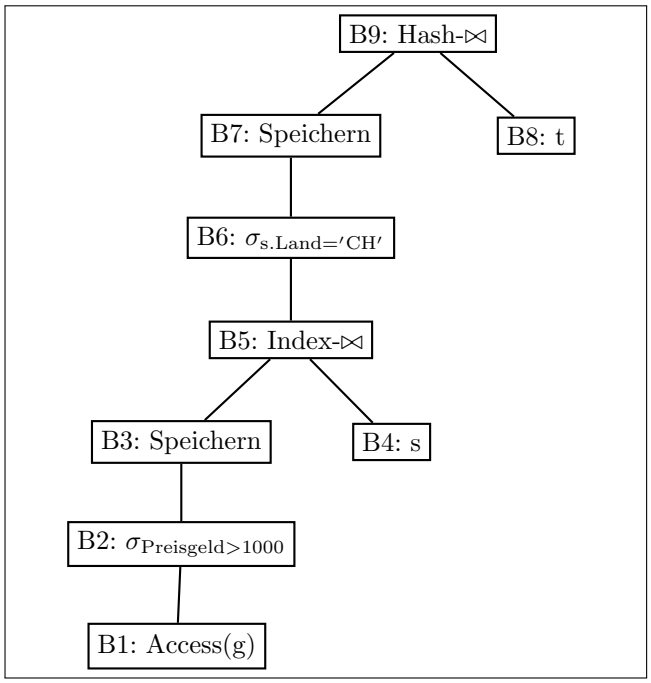
auszuführen (d.h. gesucht sind Informationen über Spieler aus der Schweiz, die ein hoch-dotiertes Turnier gewonnen haben).

Es sind folgende Selektivitäten anzunehmen: $Sel_{s/g} = 1/40000 = 0.000025$, $Sel_{t/g} = 1/50000 = 0.00002$, $Sel_{s.Land='\mathrm{CH}'} = 0.02$ und $Sel_{g.Preisgeld>1000} = 0.1$.

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. Mit "NL-⋈" sind Block Nested Loop Joins gemeint. Nehmen Sie bei diesen Block Nested Loop Joins an, dass $k = 1$ gilt und dass die äußere Relation jeweils links im Baum steht – unabhängig davon, ob dies optimal ist oder nicht. Bei der Berechnung der benötigten Seiten zum Speichern einer Relation dürfen Sie vereinfachend annehmen, dass die Tupel nicht unbedingt vollständig auf einer Seite Platz haben müssen. Außerdem dürfen Sie annehmen, dass die Tupelgröße beim Join von 2 Relationen gleich der Summe der einzelnen Tupelgrößen ist.





(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	50000	120	3000	-	0 .....
A2	50000	120	3000	$2 * b_1 * (1 + I)$ mit ..... $I = \log_{127}(\lceil b_1/128 \rceil) = 1$	12000 .....
A3	500000	40	10000	-	10000 .....
A4	50000	40	1000	-	0 .....
A5	50000	40	1000	-	1000 .....
A6	50000	40	1000	$2 * b_5 * (1 + I)$ mit ..... $I = \log_{127}(\lceil b_5/128 \rceil) = 1$	4000 .....
A7	50000	160	4000	$b_2 + b_6$	4000 .....
A8	50000	160	4000	-	4000 .....
A9	40000	100	2000	-	2000 .....
A10	800	100	40	-	0 .....
A11	800	100	40	-	40 .....
A12	1000	260	130	$b_8 + 1 + \lceil b_8/126 \rceil * (b_{11} - 1)$	5249 .....

Kosten insgesamt (Page I/O):

12000 + 10000 + 1000 + 4000 + 4000 + 4000 + 2000 + 40 + 5249 = 42289 .....

(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	500000	40	10000	-	10000 .....
B2	50000	40	1000	-	0 .....
B3	50000	40	1000	-	1000 .....
B4	40000	100	2000	-	0 .....
B5	50000	140	3500	$(b_3 + 1.5 * T_3)$	76000 .....
B6	1000	140	70	-	0 .....
B7	1000	140	70	-	70 .....
B8	50000	120	3000	-	0 .....
B9	1000	260	130	$3 * (b_7 + b_8)$	9210 .....

Kosten insgesamt (Page I/O):

10000 + 1000 + 76000 + 70 + 9210 = 96280 .....

Die folgende Datenbankbeschreibung gilt für die Aufgaben 4 – 7:

Für eine Filmdatenbank soll folgendes stark vereinfachtes Schema verwendet werden:

`person`(pid, name)
`movie`(mid, name, director: *person.pid*, vor: *movie.mid*, budget, sales)
`actor`(mid: *movie.mid*, pid: *person.pid*)

Auf der letzten Seite dieser Prüfung finden Sie eine Beispielinstanz dieses Schemas!

In der Tabelle `person` werden die Daten von Personen gespeichert. Jede Person wird eindeutig durch die Nummer `pid` identifiziert. Weiters, wird der Name `name` der Person gespeichert.

In der Tabelle `movie` werden die Daten von verschiedenen Filmen gespeichert. Jeder Film wird durch die Nummer `mid` eindeutig identifiziert und hat einen Namen `name`. Zusätzlich wird für jeden Film ein Regisseur festgelegt. Dessen `pid` wird in der Spalte `director` gespeichert. Bei Fortsetzungen wird in der Spalte `vor` die `mid` des Vorgängerfilms eingetragen. In den Spalten `budget` und `sales` werden Budget und Verkaufserlöse als Gleitkommazahlen gespeichert. Budget und Sales sind immer größer gleich 0 und werden standardmäßig auf 0 gesetzt.

Die Tabelle `actor` bildet die Zugehörigkeit von Schauspielern zu Filmen ab. Ein Tupel in der Tabelle `actor` bedeutet, dass die Person mit der Nummer `pid` an dem Film mit der Nummer `mid` mitgewirkt hat. Beide Attribute zusammen bilden den Primärschlüssel.

Treffen Sie plausible Annahmen bezüglich der Datentypen der Attribute, sofern nicht angegeben.

Aufgabe 4: (8)

Geben Sie die Create Table Statements mit allen nötigen Constraints für die drei Tabellen an.

```
CREATE TABLE person (
    pid INTEGER PRIMARY KEY,
    name VARCHAR(50) NOT NULL
);

CREATE TABLE movie (
    mid INTEGER PRIMARY KEY,
    name VARCHAR(50) NOT NULL,
    director INTEGER REFERENCES person(pid) NOT NULL,
    vor INTEGER REFERENCES movie(mid),
    budget NUMERIC(7,2) NOT NULL CHECK (budget >= 0) DEFAULT 0,
    sales NUMERIC(7,2) NOT NULL CHECK (budget >= 0) DEFAULT 0
);

CREATE TABLE actor (
    mid INTEGER REFERENCES movie(mid),
    actor INTEGER REFERENCES person(pid) NOT NULL,
    PRIMARY KEY (mid,actor)
);
```

Aufgabe 5: (6)

Evaluieren Sie das folgendes SQL-Statement bezüglich der Datenbankinstanz **movies** (siehe letzte Seite), und geben Sie die Ausgabe der Abfrage an:

```
WITH RECURSIVE temp(mid1, mid2, cnt) AS (
    SELECT mid as mid1, vor as mid2, 1
    FROM movie
UNION
    SELECT t.mid1 as mid1, m.vor as mid2, t.cnt + 1
    FROM temp t JOIN movie m ON t.mid2 = m.mid
)
SELECT mid1, MAX(cnt) as max
FROM temp t
GROUP BY mid1
ORDER BY max, mid1;
```

mid1	max
10	1
20	1
40	1
30	2
50	2
60	3

Aufgabe 6: (8)

Erstellen Sie einen PL/pgSQL Trigger `displayProfit`, der nach dem Einfügen und Verändern in der `movie`-Tabelle den aktuellen Profit des Films (`sales - budget`) und die Veränderung des Profits (neuer Profit - alter Profit) mittels `RAISE NOTICE` ausgibt.

Wenn folgende (exemplarischen) SQL Befehle über der Instanz `movies` ausgeführt werden, soll folgendes Verhalten gezeigt werden.

- `INSERT INTO movie VALUES (70, 'Rueckkehr der Jedi Ritter', 3, 30, 30, 500);`
  Ausgabe: Profit fuer Rueckkehr der Jedi Ritter: 470.00 (470.00)
- `UPDATE movie SET sales=sales+100 WHERE mid=70;`
  Ausgabe: Profit fuer Rueckkehr der Jedi Ritter: 570.00 (100.00)

```
CREATE OR REPLACE FUNCTION displayProfit() RETURNS TRIGGER AS $$
DECLARE
   old_profit NUMERIC(7,2);
BEGIN
   old_profit = 0;
   IF TG_OP = 'UPDATE' THEN
      old_profit = OLD.sales - OLD.budget;
   END IF;

   RAISE NOTICE 'Profit fuer %: % (%)', NEW.name, NEW.sales - NEW.budget,
                                        NEW.sales - NEW.budget - old_profit;

   RETURN NEW;
END;
$$ LANGUAGE plpgsql;


CREATE TRIGGER t_displayProfit AFTER INSERT OR UPDATE ON movie
   FOR EACH ROW EXECUTE PROCEDURE displayProfit();
```

Aufgabe 7: (8)

Vervollständigen Sie die Java Methode `printMovie`, die den Namen des Films und dessen SchauspielerInnen ausgibt. Die Ausgabe soll wie folgt aussehen:

```
Das Imperium schlaegt zurueck: Harrison Ford, Mark Hamill
Indiana Jones und der letzte Kreuzzug: Harrison Ford
Indiana Jones und der Tempel des Todes: Harrison Ford
Jaeger des verlorenen Schatzes: Harrison Ford, Karen Jane Allen
Krieg der Sterne: Harrison Ford, Mark Hamill
Minority Report: Tom Cruise
```

Erstellen Sie ein PreparedStatement, dass den Namen der SchauspielerInnen in einem bestimmten Film ausgibt. Die Namen sollen aufsteigend sortiert sein. Sie können hier auf eine Connection `c` zugreifen.

```
PreparedStatement pStmt = c.prepareStatement("SELECT name FROM person JOIN actor ON pid = actor
                            WHERE mid = ? ORDER BY name");
```

Vervollständigen Sie nun die Methode `printMovie`. Dazu laufen Sie mit einer Schleife durch alle Filme (sortiert nach `name`), geben den Namen aus und verwenden das **oben angelegte** PreparedStatement um die SchauspielerInnen auch auszugeben. Beachten Sie, dass die Ausgabe exakt wie im obigen Beispiel ist. Schließen Sie alle verwendeten Ressourcen. Um die Fehlerbehandlung brauchen Sie sich nicht zu kümmern.

```
public static void printMovie() throws Exception {
    Statement stmt = c.createStatement();
    ResultSet rs1 = stmt.executeQuery("SELECT mid, name FROM movie ORDER BY name");

    while (rs1.next()) {
        System.out.print(rs1.getString(2)+": ");

        pStmt.setInt(1, rs1.getInt(1));

        ResultSet rs2 = pStmt.executeQuery();
        while (rs2.next()) {

            System.out.print(rs2.getString(1));

            if (rs2.isLast())
                System.out.println();
            else
                System.out.print(", ");
        }

        rs2.close();
    }

    rs1.close();
    stmt.close();
}
```

Gesamtpunkte: 75

Sie können diese Seite abtrennen und brauchen sie nicht abzugeben!

Datenbankinstanz **movies**:

movie					
mid	name	director	vor	budget	sales
10	Minority Report	5		102.00	358.00
20	Krieg der Sterne	3		11.00	503.00
30	Das Imperium schlaegt zurueck	3	20	33.00	538.38
40	Jaeger des verlorenen Schatzes	5		18.00	384.00
50	Indiana Jones und der Tempel des Todes	5	40	28.00	333.00
60	Indiana Jones und der letzte Kreuzzug	5	50	48.00	500.00

person	
pid	name
1	Harrison Ford
2	Mark Hamill
3	George Lucas
4	Karen Jane Allen
5	Steven Spielberg
6	Tom Cruise

actor	
mid	pid
10	6
20	1
20	2
30	1
30	2
40	1
40	4
50	1
60	1