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**			09. 01. 2015
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. Die Anzahl der Zyklen im Wartegraphen entspricht immer der Anzahl der Transaktionen, die zurückgesetzt werden müssen, um ein Deadlock aufzulösen. wahr ◯ falsch ⊗
2. 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 ◯
3. Die Vollständigkeit der Fragmentierung in einem verteilten Datenbankmanagementsystem ist Voraussetzung für die Rekonstruierbarkeit. wahr ⊗ falsch ◯
4. 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 ◯
5. 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 ◯
6. 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-Quoren. wahr ◯ falsch ⊗
7. Geschachtelte Relationen in objektrelationalen Datenbanken erlauben unter Umständen eine effizientere Auswertung von Anfragen bezüglich 1:n-Relationen. wahr ⊗ falsch ◯
8. Nehmen Sie an, dass ein Index mit 1000 Schlüsseln mittels B-Baum vom Grad 10 realisiert wird. Dann gibt es mehrere gültige Möglichkeiten, diese 1000 Schlüssel im B-Baum anzuordnen. Aber in jedem Fall hat der B-Baum die Tiefe 2 (d.h.: Wurzel + 2 zusätzliche Ebenen darunter). wahr ⊗ falsch ◯
9. Betrachten Sie einen Hash Index, der die Daten selbst (und nicht nur die TIDs) enthält. Bei einer Punktanfrage mit diesem Index sind im Idealfall 1.1 Seitenzugriffe notwendig. wahr ◯ falsch ⊗
10. Bei der Historie $r_1(A)$, $r_2(A)$, $w_2(B)$, $r_1(B)$, a_1 führt der Abbruch der Transaktion T_1 zu einem kaskadierenden Rücksetzen von T_2. 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: Anfragebearbeitung (18)

Eine Uni-Datenbank enthalte die Relationen Studenten(MatrNr, Name, Sem) (kurz s), Hoeren(MatrNr, VorlNr) (kurz h) und Vorlesungen(VorlNr, SWS, Titel) (kurz v). Nehmen Sie an, daß $|s| = 40000$, $|v| = 2500$, und $|h| = 60000$. Nehmen Sie auch an, daß die durchschnittlichen Tupelgrößen für s, h und v 60, 16 und 120 Bytes sind. Die Seitengröße beträgt 1024 Bytes, und die Hauptspeicher-Puffergröße beträgt 32 Seiten. Es ist die Anfrage

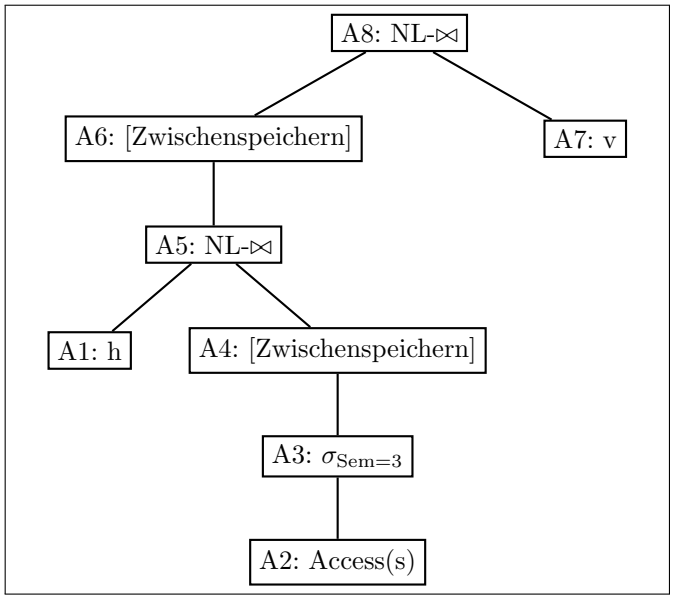
```
select *
from Studenten s, Vorlesungen v, Hoeren h
where s.Sem = 3
and v.VorlNr = h.VorlNr
and s.MatrNr = h.MatrNr
```

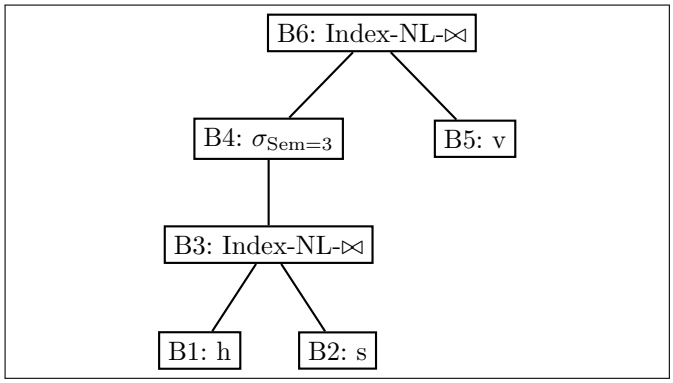
auszuführen (dh. gesucht sind Vorlesungen, die die drittsemestrigen Studenten hören – mit allen dazugehörigen Informationen über Vorlesungen und Studenten). Es sind die Selektivitäten $sel_{v/h} = 1/2500 = 0.0004$, $sel_{s/h} = 1/40000 = 2.5 * 10^{-5}$ und $sel_{s.Sem} = 0.1$ anzunehmen. Für die Primärschlüssel der Relationen s, h, v 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.

Weiters dürfen Sie folgende vereinfachende Annahmen treffen:

1. Die Tupelgröße beim Join von 2 Relationen ist gleich der Summe der einzelnen Tupelgrößen.
2. Pro Seite stehen 1000 Bytes fur das Speichern der Tupel zur Verfügung.

Für diese Anfrage sind die Operator-Bäume für 2 Auswertungspläne gegeben: Plan A realisiert alle Joins mittels Block Nested Loop Joins, während Plan B alle Joins mittels Index Nested Loop Joins realisiert.





(a) Berechnen Sie für jeden Knoten im Operatorbaum des Auswertungsplans A die Anzahl der Tupel im Resultat, die Tupelgröße, die Anzahl der Seiten im Resultat, und die geschätzten Kosten. Für Joinoperationen ist auch noch die passende Kostenformel anzugeben. Tragen Sie Ihre Berechnungen in die Tabelle auf der folgenden Seite ein.

Knoten# i	Anzahl Tupel T_i	Tupel-größe g_i	Anzahl Seiten b_i	Kostenformel	Kosten (Page I/O)
A1	60000	16	960	-	0 .....
A2	40000	60	2400	-	2400 .....
A3	4000	60	240	-	0 .....
A4	4000	60	240	-	240 .....
A5	6000	76	456	$b_1 + 1 + \lceil b_1/30 \rceil * (b_4 - 1)$	8370 .....
A6	6000	76	456	-	456 .....
A7	2500	120	300	-	0 .....
A8	6000	196	1176	$b_6 + 1 + \lceil b_6/30 \rceil * (b_7 - 1)$	2633 .....

Kosten insgesamt (Page I/O):

2400 + 240 + 8370 + 456 + 2633 = 16099 ...

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

Knoten# i	Anzahl Tupel T_i	Tupel-größe g_i	Anzahl Seiten b_i	Kostenformel	Kosten (Page I/O)
B1	60000	16	960	-	0 .....
B2	40000	60	2400	-	0 .....
B3	60000	76	4690	$b_1 + 1.2 * T_1$	74400 .....
B4	6000	76	469	-	0 .....
B5	2500	120	300	-	0 .....
B6	6000	196	1176	$b_4 + 1.2 * T_4$	7496 .....

Kosten insgesamt (Page I/O):

74400 + 7496 = 81896 .....

(c) Wie sehen die Gesamtkosten aus, wenn im Plan A alle Block Nested Loop Joins durch Hash-Joins ersetzt werden? (Page I/O) Geben Sie die Gesamtsumme an, aber auch (einzeln) die Kosten der Joins.

2400 + 240 + 3*$(b_1 + b_4)$ + 456 + 3*$(b_6 + b_7)$ = 8316

Aufgabe 3: (12)

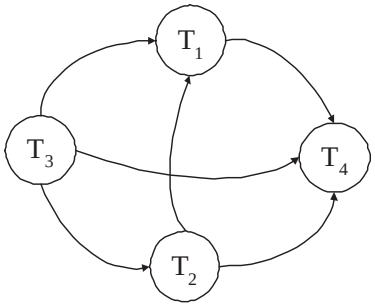
(a) (5 Punkte) Eine Historie sei gegeben durch folgende Folge von Elementaroperationen: $r_1(A)$, $r_2(B)$, $r_3(A)$, $w_2(B)$, $w_3(C)$, $r_4(D)$, $r_1(B)$, $r_1(C)$, $r_2(C)$, $r_3(C)$, $w_4(C)$, c_1, c_2, c_3, c_4.

Zeichnen Sie ins erste Kästchen den Serialisierbarkeitsgraphen für die Transaktionen T_1, T_2, T_3 und T_4. Verwenden Sie dabei folgende Konvention: eine Kante $T_i \rightarrow T_j$ bedeutet, dass in einer äquivalenten seriellen Historie die "Transaktion T_i vor T_j" ausgeführt werden muss (vgl. VO-Folien bzw. Kemper-Buch).

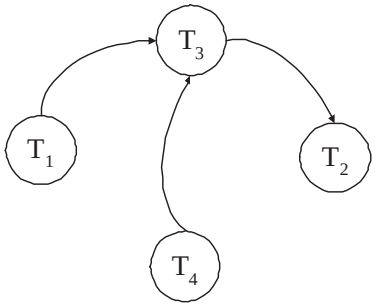
(b) (5 Punkte) Betrachten Sie die folgende Folge von Sperranforderungen, wobei "lockS_i(O)"(bzw. "lockX_i(O)") bedeutet, dass die Transaktion T_i eine Lesesperre (bzw. eine Schreibsperre) auf das Datenobjekt O anfordert: lockS_1(A), lockX_2(B), lockX_3(C), lockS_1(C), lockS_3(B), lockS_4(C), lockS_2(A).

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

(a) Serialisierbarkeitsgraph:



(b) Wartegraph:



(c) (2 Punkte) Geben Sie für den Serialisierbarkeitsgraphen aus (a) eine mögliche Reihenfolge der Serialisierung an:

T 3 ..... vor T 2 ..... vor T 1 ..... vor T 4 .....

Die folgende Datenbankbeschreibung gilt für die Aufgaben 4 – 7:

Eine Lehrerin möchte für ihren Mathematikunterricht ein neues Bewertungssystem einführen. Dazu werden die erledigten Aufgaben eines/-r Schülers/-in mittels Erfahrungspunkten (XPs) bewertet. Eine Note entspricht dem Erreichen einer bestimmten Punkteanzahl. Die Daten sollen in folgendem stark vereinfachten Datenbankschema gespeichert werden.

S(sid, sname)
K(kid, kname, anz)
XP(sid: *S.sid*,kid: *K.kid*,datum,xp)

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

In der Tabelle `S` werden die Schüler/innen gespeichert. Jede/r Schüler/in hat eine eindeutige Nummer `sid`, und einen Namen `sname`.

In der Tabelle `K` werden Kategorien der verschiedenen Aufgaben gespeichert (z.B.: Schularbeiten, Mitarbeit, etc.). Jede Kategorie wird durch eine eindeutige Nummer `kid` identifiziert. Zudem wird der Name `kname` und die maximalen Vorkommnisse `anz` der Kategorie gespeichert. Die Werte im Attribut `anz` müssen größer 0 sein.

In der Tabelle `XP` werden Erfahrungspunkte gespeichert. Für jede Lösung einer Aufgabe wird der/die Schüler/in, die Kategorie der Aufgabe, das Datum und die gesammelten Erfahrungspunkte gespeichert.

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

Aufgabe 4: (5)

Geben Sie CREATE TABLE Statements mit allen nötigen Constraints für die drei Tabellen an.

```
CREATE TABLE S (
        sid INTEGER PRIMARY KEY,
        sname VARCHAR(30)
);

CREATE TABLE K (
        kid INTEGER PRIMARY KEY,
        kname VARCHAR(30),
        anz INTEGER CHECK (anz > 0)
);

CREATE TABLE XP (
        sid INTEGER REFERENCES S(sid),
        kid INTEGER REFERENCES K(kid),
        datum DATE,
        xp INTEGER,
        PRIMARY KEY (sid, kid, datum)
);
```

Aufgabe 5: (4)

Evaluieren Sie die folgenden SQL-Statements bezüglich der Datenbankinstanz **unterricht** (siehe letzte Seite), und geben Sie die Ausgaben der Abfrage an. Falls mehrere Ergebnisse zurückgegeben werden, trennen Sie diese durch ; (Semikolon).

```
SELECT COUNT(*) FROM XP
WHERE kid IN (SELECT kid FROM K WHERE anz >= 5);
```

8

```
SELECT COUNT(*) FROM S, K;
```

12

```
SELECT SUM(xp) FROM K, XP
WHERE K.kid = XP.kid AND kname = 'Mitarbeit';
```

20

```
SELECT SUM(xp) FROM XP
GROUP BY datum HAVING SUM(xp) > 20;
```

155 ; 25

Aufgabe 6: (9)

Erstellen Sie einen PL/pgSQL Trigger `checkAnz`, der verhindert, dass in die `XP`-Tabelle Einträge hinzugefügt werden, obwohl bereits `anz` Einträge dieses/-r Schülers/-in vorhanden sind. In diesem Fall soll eine Exception geworfen werden.

Betrachten Sie beispielsweise die Instanz **unterricht** auf der letzten Seite:

- Wird versucht, den Eintrag `(3,11,2014-12-22,10)` hinzuzufügen, soll eine Exception geworfen werden, denn die Schülerin hat bereits **2** Wiederholungen (maximale Anzahl im Feld `anz` der `K` Tabelle).
- Wird versucht, den Eintrag `(1,11,2014-12-22,10)` hinzuzufügen, soll **keine** Exception geworfen werden, denn der Schüler hat in dieser Beispielinstanz noch keine Erfahrungspunkte durch eine Wiederholung gesammelt.

```
CREATE OR REPLACE FUNCTION fCheckAnz() RETURNS TRIGGER AS $$
DECLARE
   max INTEGER;
BEGIN
   SELECT anz INTO max FROM K WHERE kid = NEW.kid;

   IF EXISTS (SELECT COUNT(*)
              FROM XP
              WHERE XP.kid=NEW.kid AND XP.sid = NEW.sid
              HAVING COUNT(*) >= max)
   THEN
      RAISE EXCEPTION 'Bereits zuviele Erfahrungspunkte in dieser Kategorie!';
   END IF;
   RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER checkAnz BEFORE INSERT ON XP FOR EACH ROW
EXECUTE PROCEDURE fCheckAnz();
```

Aufgabe 7: (12)

Vervollständigen Sie die folgende Java Funktion `XPBericht`, sodass für jede/n Schüler/in die aktuellen Note ausgegeben wird. Dazu verwendet die Funktion `XPBericht` zwei Statements. Das Statement `sS` wählt alle Zeilen der Tabelle `S` sortiert nach `sname` aufsteigend aus. Das ResultSet des Statements `sS` wird durchlaufen, und danach werden mittels dem Prepared Statement `psSumXP` die Erfahrungspunkte des/-r Schüler/-in ermittelt und in der Variable `XP` gespeichert. Das Ergebnis wird mittels der vordefinierten Zeile ausgegeben.

```
private static void XPBericht(Connection c) throws SQLException {
        //PreparedStatement zum Berechnen der XP pro Schueler/-in
        PreparedStatement psSumXP = c.prepareStatement("SELECT SUM(xp) FROM XP WHERE sid=?");
        ResultSet rsXP = null;
        int XP;


        //Statement erstellen und alle Schueler/-innen auswaehlen
        Statement sS = c.createStatement();
        ResultSet rsS = sS.executeQuery("SELECT sid, sname FROM S ORDER BY sname ASC");

        while (rsS.next()) {
                //PreparedStatement ausfuehren und Summe der XP in der Variable XP speichern
                psSumXP.setInt(1, rsS.getInt("sid"));
                rsXP = psSumXP.executeQuery();
                if (rsXP.next())
                        XP = rsXP.getInt(1);
                else
                        XP = 0;

                System.out.println(rsS.getString("sname") + " hat " + XP +
                                   " Erfahrungspunkte. Note: " + Note(XP));

                //Korrektes ResultSet schliessen
                rsXP.close();
        }

        //Ressourcen freigeben (ausser die Connection c)
        rsS.close();
        sS.close();
        psSumXP.close();
}

private static int Note(int xP) {
        if (xP >= 100) return 1;
        if (xP >= 87) return 2;
        if (xP >= 75) return 3;
        if (xP >= 50) return 4;
        return 5;
}
```

Fortsetzung der Aufgabe 8 auf der nächsten Seite!

Geben Sie nun die Ausgabe der Funktion `XPBericht` bezüglich der Datenbankinstanz **unterricht** aus.

```
Anna hat 40 Erfahrungspunkte. Note: 5
Klaus hat 45 Erfahrungspunkte. Note: 5
Maria hat 65 Erfahrungspunkte. Note: 4
Max hat 75 Erfahrungspunkte. Note: 3
```

Gesamtpunkte: 75

Sie können diese Seite abtrennen und brauchen sie nicht abzugeben!

Datenbankinstanz **unterricht**:

S	
sid	sname
1	Max
2	Klaus
3	Maria
4	Anna

K		
kid	kname	anz
10	Mitarbeit	10
11	Wiederholung	2
12	Schularbeit	6

XP			
sid	kid	datum	xp
1	10	2014-09-11	5
1	10	2014-09-12	5
1	12	2014-12-10	45
2	10	2014-09-23	5
2	12	2014-12-10	35
2	11	2014-12-15	5
3	12	2014-12-10	50
3	11	2014-12-15	10
3	11	2014-12-19	5
4	10	2014-09-12	5
4	12	2014-12-10	25
4	11	2014-12-15	10