Gruppe A

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

PRÜFUNG AUS DATENBANKSYSTEME **VL 181.186**			10. 3. 2011
Kennnr.	Matrikelnr.	Familienname	Vorname

Arbeitszeit: 120 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. Eine Tupel-ID kann auch auf eine weitere Tupel-ID verweisen. Die wirklich Werte befinden sich dann in einer anderen Seite. Dies wird auch als "Forward" bezeichnet. wahr ⊗ falsch ◯
2. Für $X(ABC)$ und $Y(BCD)$ ist $X \bowtie Y$ äquivalent zu $\Pi_{A,X.B,X.C,D}(\sigma_{X.B=Y.B,X.C=Y.C}(X \times Y))$. wahr ⊗ falsch ◯
3. Obwohl die Reihenfolge von Joins für das Ergebnis irrelevant ist (da Joins assoziativ sind), kann sie doch starke Auswirkungen auf die Laufzeit der Query haben. wahr ⊗ falsch ◯
4. Bei der horizontalen Fragmentierung ergeben sich für n Zerlegungsprädikate 2^n Fragmente, die durch eine Vereinigung wieder zusammengeführt werden können. wahr ⊗ falsch ◯
5. Ein WITH RECURSIVE Statement in PostgreSQL erzeugt eine Common Table Expression, deren Inhalt in der darauffolgenden referenzierenden SQL-Query verwendet werden kann. wahr ⊗ falsch ◯
6. Betrachten Sie drei Relationen $U(AB)$, $V(BC)$ und $W(CD)$. Dann gilt auf jeden Fall folgende Gleichheit: $(U \bowtie V) \bowtie W) = (U \times W) \ltimes V)$ wahr ⊗ falsch ◯
7. Nehmen Sie an, dass eine Relation S 10000 Seiten umfasst und dass die Puffergröße 32 beträgt. Dann ist bei einem Hash-Join von R mit S in der Build-Phase möglicherweise aber nicht notwendigerweise ein Re-hashing von S erforderlich. wahr ⊗ falsch ◯
8. Betrachten Sie die Kostenformel $2 \cdot b_R \cdot (1 + I)$ mit $I = \lceil \log_{m-1}(\lceil b_R/m \rceil) \rceil$ für das externe Sortieren. Dabei steht I für die Anzahl der "passes". wahr ⊗ falsch ◯
9. In der relationalen Algebra ist die Operation σ distributiv mit $\cup$, $\cap$, und $\setminus$, die Operation π ist distributiv mit $\cup$. wahr ⊗ falsch ◯
10. Die Historie $r_1(A)$, $r_2(B)$, $w_1(A)$, $w_1(B)$, $w_2(B)$, c_2, c_1 hat folgende Eigenschaften: Sie vermeidet kaskadierendes Rücksetzen, und sie ist nicht serialisierbar. 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: (18)

Die statistischen Ämter in Los Angeles und San Francisco wollen gemeinsam eine verteilte Wählerdatenbank erstellen. In dieser Datenbank sind folgende Tabellen enthalten:

Waehler(SSN, name, stadt, partei, geschlecht, alter, adresse, beruf)
Durchschnittsgehalt(beruf, alter, gehalt)

Die Wähler Tabelle wird (vertikal) fragmentiert in folgende 2 Tabellen:

WaehlerBasic(SSN, name, stadt, partei, geschlecht, alter) und
WaehlerExtra(SSN, adresse, beruf).

Außerdem wird die Tabelle WaehlerBasic(SSN, name, stadt, partei, geschlecht, alter) nach dem Attribut "stadt" (horizontal) weiter fragmentiert in die zwei Tabellen WaehlerBasicLA und WaehlerBasicSF. Es ist die Anfrage

select name
from Waehler w, Durchschnittsgehalt d
where w.partei = 'Demokratische' and d.gehalt > 800k and w.beruf = d.beruf and w.alter = d.alter

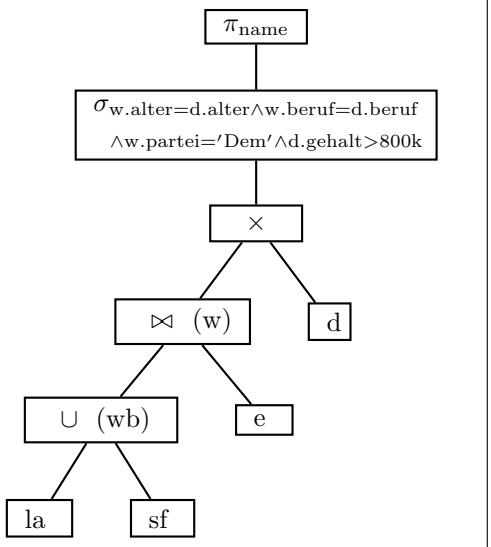
auszuführen (d.h. Informationen über Wähler mit höheren Einkommen, die Demokratische Patei gewählt haben.

(a) Zeichnen Sie ins erste Kästchen den Operator-Baum für die kanonische Übersetzung. Verwenden Sie für die 3 Fragmente der Wähler-Tabelle sowie für die Durchschnittsgehalt-Tabelle folgende Abkürzungen: **e** (WaehlerExtra), **la** (WaehlerBasicLA), **sf** (WaehlerBasicSF) und **d** (Durchschnittsgehalt). Außerdem können Sie den String 'Demokratische' mit 'Dem' abkürzen.

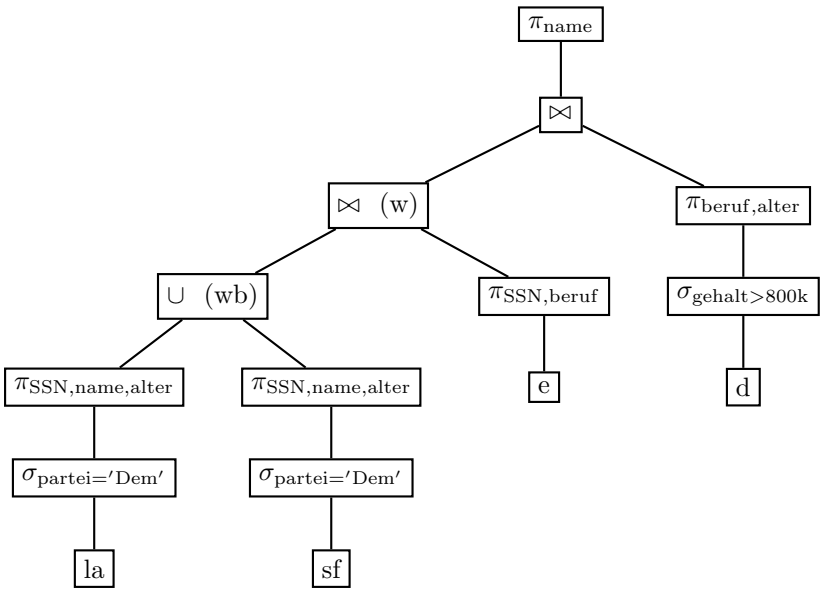
(b) Zeichnen Sie ins zweite Kästchen den Operator-Baum für den optimierten algebraischen Ausdruck. Wenden Sie für die Optimierung folgende Heuristiken an:

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

(a) Kanonische Übersetzung:



(b) Optimierter Ausdruck:



Aufgabe 3: (8)

Erweitern Sie das MGL-Sperrverfahren (multiple-granularity locking) um einen weiteren Sperrmodus SIX. Dieser Sperrmodus sperrt den betreffenden Knoten im S-Modus (und damit implizit alle Unterknoten) und kennzeichnet die beabsichtigte Sperrung von einem (oder mehreren) Unterknoten im X-Modus.

(a) Erweitern Sie die Kompatibilitätsmatrix des MGL-Sperrverfahrens um diesen Sperrmodus. Benutzen Sie ✓(Hakerl) um Kompatibilität bzw. – (Strich) um Nicht-Kompatibilität zu kennzeichnen. Füllen Sie hierzu die folgende Tabelle komplett aus. [5]

	NL	S	X	IS	IX	SIX
S	✓	✓	--	✓	--	--
X	✓	--	--	--	--	--
IS	✓	✓	--	✓	✓	✓
IX	✓	--	--	✓	✓	--
SIX	✓	--	--	✓	--	--

(b) Charakterisieren Sie Transaktionen, für die dieser zusätzliche Sperrmodus vorteilhaft ist. [2]

```
Transaktionen, die innerhalb der Sperrhierarchie große Teile eines Unterbaums
lesen, aber nur wenige Daten darin modifizieren.
```

(c) Transaktionen, die von SIX profitieren, tun dies aufgrund [1]

```
erhöhter Parallelität mit konkurrierenden Transaktionen.
```

Die folgende Datenbankbeschreibung gilt für die Aufgaben 4 – 7:

Gegeben ist folgendes stark vereinfachtes Datenbankschema, das Personen und deren Freundschatfsbeziehung beschreibt.

person(pid, name)
friends(a: *person.pid*, b: *person.pid*)

Jede Person `person` hat eine eindeutige Identifikationsnummer `pid` und einen Namen `name`. Die Tabelle `friends` gibt an, dass die Personen mit ID `a` und ID `b` befreundet sind.

Wählen Sie entsprechende Datentypen (INTEGER, VARCHAR) für die Attribute.

Aufgabe 4: (6)

Geben Sie nun CREATE-Statements mit allen entsprechenden Constraints für die Tabellen `person` und `friends` an.

```
CREATE TABLE person (
    pid INTEGER PRIMARY KEY,
    name VARCHAR (30)
);

CREATE TABLE friends (
    a INTEGER REFERENCES person(pid),
    b INTEGER REFERENCES person(pid),
        PRIMARY KEY(a,b)
);
```

Geben Sie ein CREATE Statement an, das eine Sequence `pidSequence` erstellt. Diese Sequence soll Werte von 1000 bis 9000 in 10er Schritten vergeben, und bei Erreichen von 9000 wieder bei 1000 fortsetzen.

```
CREATE SEQUENCE pid_sequence
     START WITH 1000
     INCREMENT BY 10
     MAXVALUE 9000
     CYCLE;
```

Geben Sie nun ein INSERT-Statement an, das in die person-Tabelle eine Zeile einfügt. Der Wert für `pid` soll dabei mittels der `pidSequence` erzeugt werden.

```
INSERT INTO person VALUES (nextval('pid_sequence'), 'Max');
```

Aufgabe 5: Hierarchische Abfragen (10)

Geben Sie ein SELECT-Statement an, das alle "friends of a friend" der Person mit dem Namen "Max" auswählt. D.h. es sollen die IDs jener Personen ausgegeben werden, die in der `friends`-Tabelle zwei Schritte weit von Max entfernt sind.

Hat beispielsweise Max die ID 1, dann soll für folgende `friends`-Tabelle

a	b
1	2
2	3

ausgegeben werden, dass die Person mit der ID 3 ein "friend of a friend" von Max ist.

Nehmen Sie der Einfachheit halber an, dass wirklich alle Personen-IDs, die durch zwei Schritte von Max erreichbar sind auch ausgegeben werden sollen (also zB auch Max selbst). Nehmen Sie weiters vereinfachend an, dass die `friends`-Tabelle symmetrisch ist, d.h. wenn a Freund von b ist, so ist auch b Freund von a (ist also wie beim obigen Beispiel das Tupel (1,2) vorhanden, so ist auch das Tupel (2,1) vorhanden usw.).

```
SELECT f2.b
    FROM person p, friends f1, friends f2
    WHERE p.name = "Max"
    AND p.pid = f1.a
    AND f1.b = f2.a
```

Geben Sie nun eine hierarchische Anfrage an, die Max, seine Freunde, Freunde seiner Freunde, ... zurückgibt. D.h. es sollen alle Personen-IDs ausgegeben werden, die durch beliebig viele Schritte in der `friends`-Tabelle von Max aus erreichbar sind.

```
WITH RECURSIVE temp(a) AS (
    SELECT p.pid
    FROM person p
    WHERE p.name = "Max"
UNION ALL
   SELECT f.b
   FROM temp t, friends f
   WHERE t.a = f.a
)
SELECT * FROM temp;
```

Aufgabe 6: PL/SQL Trigger (8)

Erstellen Sie einen Trigger, der immer dann aufgerufen wird, wenn eine `person`-Zeile gelöscht werden soll.

Dieser Trigger soll für jede zu löschende Zeile

- die Anzahl der Freunde der zu löschenden Person bestimmen, d.h. die Anzahl der Zeilen in der friends-Tabelle, die den zu löschenden User betreffen.
- wenn diese Anzahl grösser als 0 ist, eine Exception werfen. In der Meldung der Exception soll die Anzahl der Freunde der zu löschenden Person angegeben werden.

Vergessen Sie nicht, Variablen die Sie benötigen auch zu deklarieren.

```
CREATE FUNCTION deleteNotification()  RETURNS trigger AS $$
DECLARE
    friendCount INTEGER;
BEGIN
    SELECT count(*) INTO friendCount FROM friends f WHERE OLD.pid=f.a;
    IF friendCount > 0 THEN
        RAISE EXCEPTION 'Zu löschender User hat ' || friendCount || ' Freunde!'
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER deleteTrigger
    BEFORE DELETE ON person FOR EACH ROW
    EXECUTE PROCEDURE deleteNotification();
```

Aufgabe 7: (10)

Erstellen Sie eine Java Methode, die für jede in der Datenbank vorhandene Person die Anzahl der jeweiligen Freunde ausgibt.

- Verwenden Sie für die Bestimmung der Freunde-Anzahl ein `PreparedStatement`.
- Es reicht aus, die IDs der jeweiligen Personen auszugeben.

Um die Behandlung von Exception brauchen Sie sich bei dieser Aufgabe nicht zu kümmern.

```
public void printFriends() throws Exception {
    Class.forName("org.postgresql.Driver");
    Connection c = DriverManager.getConnection(
        "jdbc:postgresql://localhost/db", "username", "password");

    PreparedStatement friendPS = c.prepareStatement(
        "SELECT count(*) FROM friends WHERE a=?");

    Statement personS = c.createStatement();
    ResultSet personRS = personS.executeQuery(
        "SELECT pid FROM person");

    while (personRS.next()) {
        int pid = personRS.getInt(1);
        friendPS.setInt(pid, 1);
        ResulSet friendRS = friendPS.executeQuery();
        System.out.println("User " + pid + " hat " +
            friendRS.getInt(1) + " Freunde.");
    }

    friendPS.close();
    personS.close();
    c.close();
}
```

Gesamtpunkte: 75