	Lehrveranstaltung	**Grundlagen von Datenbanken**		WS 2016/17
	Aufgabenzettel	**4 (Lösungsvorschläge)**		
	Gesamtpunktzahl	**40**		
	Ausgabe	**Mi. 30.11.2016**	Abgabe	**Fr. 16.12.2016**

1 Relationenalgebra

[8 P.]

Gegeben seien die folgenden Relationenschemata:

Land(LNR, Name)

Job(JNR, Titel, Jahresgehalt)

Person(PNR, Vorname, Nachname, Geburtsdatum, Heimat → Land.LNR)

Bewerbung(Bewerber → Person.PNR, Job → Job.JNR, Sachbearbeiter → Person.PNR)

Benutzen Sie zur Lösung der folgenden Aufgaben ausschließlich die in der Vorlesung vorgestellten Operatoren der Relationenalgebra!

a) Geben Sie einen Relationenalgebra-Ausdruck an, der zu dem unten angegebenen SQL-Ausdruck äquivalent ist. [2 P.]

```
SELECT DISTINCT Jahresgehalt
  FROM Person, Bewerbung, Job
  WHERE PNR = Bewerber
    AND Job = JNR
    AND Geburtsdatum >= '1980-01-01'
```

Lösungsvorschlag:

$$\pi_{Jahresgehalt}(\sigma_{Geburtsdatum \geq \text{"1980-01-01"}}(Person) \underset{PNR=Bewerber}{\bowtie} Bewerbung \underset{Job=JNR}{\bowtie} Job)$$

b) Geben Sie einen Relationenalgebra-Ausdruck an, der Titel und Jahresgehalt der Jobs ausgibt, auf die sich mindestens ein Schweizer beworben hat. [2 P.]

Lösungsvorschlag:

$$\pi_{Titel, Jahresgehalt}(\sigma_{Name=\text{"Schweiz"}}(Land) \underset{LNR=Heimat}{\bowtie} Person \underset{PNR=Bewerber}{\bowtie} Bewerbung \underset{Job=JNR}{\bowtie} Job)$$

c) Geben Sie einen Relationenalgebra-Ausdruck an, der die Vor- und Nachnamen aller Personen ausgibt, die sich noch nie um einen Job beworben haben. [2 P.]

Lösungsvorschlag:

$$\pi_{Vorname,Nachname}(Person \bowtie (\pi_{PNR}(Person) - \rho_{Bewerber \leftarrow PNR}(\pi_{Bewerber}(Bewerbung))))$$

d) Geben Sie eine natürlichsprachliche Beschreibung der Ergebnismenge des folgenden Relationenalgebra-Ausdrucks an. [2 P.]

$$\pi_{Geburtsdatum}(Person \underset{PNR=Bewerber}{\bowtie} Bewerbung \underset{Sachbearbeiter=PNR}{\bowtie} \pi_{PNR}(\sigma_{Geburtsdatum \geq "1994-12-31"}(Person)))$$

Lösungsvorschlag:

Das Geburtsdatum aller Bewerber, deren Bewerbung von einem Sachbearbeiter bearbeitet wird, der nach dem 31.12.1994 (*1994-12-31*) geboren wurde.

	Lehrveranstaltung	**Grundlagen von Datenbanken**		WS 2016/17
	Aufgabenzettel	**4 (Lösungsvorschläge)**		
	Gesamtpunktzahl	**40**		
	Ausgabe	**Mi. 30.11.2016**	Abgabe	**Fr. 16.12.2016**

2 Schemadefinition

[17 P.]

Wir verwenden das gleiche Datenbankschema wie in der dritten Aufgabe von Aufgabenblatt 3:

Buch	Titel	Erscheinungsjahr	Seitenzahl	Verlag
	Schall und Wahn	1929	304	Diogenes
	Als ich im Sterben lag	1930	173	Diogenes
	Hundert Jahre Einsamkeit	1967	480	Fischer
	Der Fremde	1942	160	rororo
	Krieg und Frieden	1869	1536	Anaconda
	Anna Karenina	1878	991	Anaconda
	Schuld und Sühne	1866	752	Deutscher Taschenbuch Verlag
	Requiem für einen Traum	1978	316	Rowohlt
	Der Talisman	1984	714	Heyne

Person	PID	Vorname	Nachname	Lieblingsbuch
	1	Leo	Tolstoi	Schuld und Sühne
	2	Fjodor	Dostojewski	Krieg und Frieden
	3	Hubert	Selby	Der Fremde
	4	Albert	Camus	Schuld und Sühne
	5	William	Faulkner	Schuld und Sühne
	6	Stephen	King	Hundert Jahre Einsamkeit
	7	Peter	Straub	Schall und Wahn
	8	Gabriel	García Márquez	Requiem für einen Traum

Lieblingsbuch → Buch.Titel

Schreibt	Autor	Buch
	1	Krieg und Frieden
	1	Anna Karenina
	2	Schuld und Sühne
	3	Requiem für einen Traum
	4	Der Fremde
	5	Schall und Wahn
	5	Als ich im Sterben lag
	6	Der Talisman
	7	Der Talisman
	8	Hundert Jahre Einsamkeit

Autor → Person.PID, Buch → Buch.Titel

Begutachtet	Lektor	Buch
	2	Anna Karenina
	1	Schuld und Sühne
	8	Requiem für einen Traum
	6	Requiem für einen Traum
	5	Der Fremde
	4	Als ich im Sterben lag
	2	Krieg und Frieden
	7	Hundert Jahre Einsamkeit

Lektor → Person.PID, Buch → Buch.Titel

Um die Konsistenz der Daten sicherzustellen, sollen folgende Integritätsbedingungen gelten:
IB1: Die Seitenzahl eines Buches muss zwischen 0 und 4.000 liegen.
IB2: Der Nachname eines Autors ist eindeutig.
IB3: Alle Felder bis auf *Person.Lieblingsbuch* sind Pflichtfelder.

	Lehrveranstaltung	**Grundlagen von Datenbanken**		WS 2016/17
	Aufgabenzettel	**4 (Lösungsvorschläge)**		
	Gesamtpunktzahl	**40**		
	Ausgabe	**Mi. 30.11.2016**	Abgabe	**Fr. 16.12.2016**

a) Definieren Sie das angegebene Schema mithilfe von Befehlen der SQL DDL (Data Definition Language). [8 P.]
Zur Prüfung Ihrer Lösung führen Sie die DDL-Befehle bitte in MySQL aus.
Hinweis: Legen Sie Fremdschlüssel bitte als benannte Constraints an. Legen Sie die Prüfung der Seitenzahl als eine Check-Klausel an, auch wenn MySQL diese (ohne Fehler) ignoriert.

Lösungsvorschlag:

```
CREATE TABLE Buch(
  Titel             varchar(50) PRIMARY KEY,
  Erscheinungsjahr  int         NOT NULL,
  Seitenzahl        int         NOT NULL CHECK(Seitenzahl > 0 AND Seitenzahl < 4000),
  Verlag            varchar(50) NOT NULL
);

CREATE TABLE Person(
  PID               int         PRIMARY KEY,
  Vorname           varchar(50) NOT NULL,
  Nachname          varchar(50) UNIQUE NOT NULL,
  Lieblingsbuch     varchar(50),
  CONSTRAINT fk_pers_lbuch FOREIGN KEY (Lieblingsbuch) REFERENCES Buch (Titel)
);

CREATE TABLE Schreibt(
  Autor             int,
  Buch              varchar(50),
  CONSTRAINT pk_schreibt PRIMARY KEY (Autor, Buch),
  CONSTRAINT fk_schreibt_autor FOREIGN KEY (Autor) REFERENCES Person (PID),
  CONSTRAINT fk_schreibt_buch FOREIGN KEY (Buch) REFERENCES Buch (Titel)
);

CREATE TABLE Begutachtet(
  Lektor            int,
   Buch             varchar(50),
  CONSTRAINT pk_begutachtet PRIMARY KEY (Lektor, Buch),
  CONSTRAINT fk_begutachtet_lektor FOREIGN KEY (Lektor) REFERENCES Person (PID),
  CONSTRAINT fk_begutachtet_buch FOREIGN KEY (Buch) REFERENCES Buch (Titel)
);
```

b) Erklären Sie knapp, was es für Transaktionen bedeutet, dass in MySQL die referentielle Integrität von Fremdschlüsseln nicht verzögert am Ende der Transaktion (*deferred*) geprüft werden kann, sondern stets direkt. [3 P.]

Erläutern Sie, was passieren würde, wenn *Buch* das Feld *Editor* erhalten würde, welches ein Fremdschlüssel auf *Person.PID* ist. Was müsste man bei der Definition des Schemas in SQL DDL beachten?

Lösungsvorschlag:

Laut Dokumentation unterstützt MySQL das deferred Checking von Fremdschlüsseln nicht:

> Deviation from SQL standards: Like MySQL in general, in an SQL statement that inserts, deletes, or updates many rows, InnoDB checks UNIQUE and FOREIGN KEY constraints row-by-row. When performing foreign key checks, InnoDB sets shared row-level locks on child or parent records it has to look at. InnoDB checks foreign key constraints immediately; the check is not deferred to transaction commit. According to the SQL standard, the default behavior should be deferred checking. That is, constraints are only checked after the entire SQL statement has been processed. Until InnoDB implements deferred constraint checking, some things will be impossible, such as deleting a record that refers to itself using a foreign key.

Dies hat Auswirkungen auf Transaktionen, z.B:

- Einträge, die sich gegenseitig oder selbst referenzieren, können nicht gelöscht werden.
- Beim Löschen von Einträgen muss auf die richtige Reihenfolge geachtet werden, z.B. müssen wegen *Person.Lieblingsbuch* $\rightarrow$ *Buch.Titel* beim Löschen eines Buches zuerst alle referenzierenden Personen gelöscht oder geändert werden.
- Beim Einfügen muss ebenfalls auf die richtige Reihenfolge geachtet werden, damit keine nichtexistrierenden Tupel referenziert werden.
- Das Löschen ganzer Tabellen erfordert ebenfalls die Einhaltung der Reihenfolge, die durch die Fremdschlüssel vorgegeben ist (z.B. *Buch* nach *Begutachtet* löschen).

Wenn in *Buch* das Feld *Editor* in der beschriebenen Weise eingeführt wird, entsteht eine wechselseitige Abhängigkeit. Da Fremdschlüsselintegrität nicht deferred geprüft wird, können neue Einträge dann nur angelegt werden, wenn einer der Fremdschlüssel NULLable ist, d.h. auch NULL-Werte annehmen darf. Bei der Definition des Schemas muss darauf geachtet werden, zunächst nur einen Foreign Key Constraint anzulegen. Sobald beide Tabellen existieren, kann über ALTER TABLE der zweite Constraint hinzugefügt werden. Beim Löschen eines Tupel oder beim Löschen ganzer Tabellen müssen entsprechend alle Ringbezüge zunächst aufgelöst werden.

c) Befüllen Sie die Datenbank mit den in der Tabelle angegebenen Datensätzen. Schreiben Sie die SQL-Befehle auf. [4 P.]

Lösungsvorschlag:

```
INSERT INTO Buch (Titel, Erscheinungsjahr, Seitenzahl, Verlag) VALUES
 ("Schall und Wahn", 1929, 304, "Diogenes"),
 ("Als ich im Sterben lag", 1930, 173, "Diogenes"),
 ("Hundert Jahre Einsamkeit", 1967, 480, "Fischer"),
 ("Der Fremde", 1942, 160, "rororo"),
 ("Krieg und Frieden", 1869, 1536 , "Anaconda"),
 ("Anna Karenina", 1878, 991, "Anaconda"),
 ("Schuld und Sühne", 1866, 752, "Deutscher Taschenbuch Verlag"),
 ("Requiem für einen Traum", 1978, 316, "Rowohlt"),
 ("Der Talisman", 1984, 714 , "Heyne ");
INSERT INTO Person (PID, Vorname, Nachname, Lieblingsbuch) VALUES
 (1, "Leo", "Tolstoi", "Schuld und Sühne"),
 (2, "Fjodor", "Dostojewski", "Krieg und Frieden"),
 (3, "Hubert", "Selby", "Der Fremde"),
 (4, "Albert", "Camus", "Schuld und Sühne"),
 (5, "William", "Faulkner", "Schuld und Sühne"),
 (6, "Stephen", "King ", "Hundert Jahre Einsamkeit"),
 (7, "Peter", "Straub", "Schall und Wahn"),
 (8, "Gabriel", "García Márquez", "Requiem für einen Traum");
INSERT INTO Schreibt (Autor, Buch) VALUES
 (1, "Krieg und Frieden"),
 (1, "Anna Karenina"),
 (2, "Schuld und Sühne"),
 (3, "Requiem für einen Traum"),
 (4, "Der Fremde"),
 (5, "Schall und Wahn"),
 (5, "Als ich im Sterben lag"),
 (6, "Der Talisman"),
 (7, "Der Talisman"),
 (8, "Hundert Jahre Einsamkeit");
INSERT INTO Begutachtet (Lektor, Buch) VALUES
 (2, "Anna Karenina"),
 (1, "Schuld und Sühne"),
 (8, "Requiem für einen Traum"),
 (6, "Requiem für einen Traum"),
 (5, "Der Fremde"),
 (4, "Als ich im Sterben lag"),
 (2, "Krieg und Frieden"),
 (7, "Hundert Jahre Einsamkeit");
```

d) Geben Sie die SQL Befehle an, um: (2 Punkte)

- alle Personen mit dem Vornamen „Peter“ zu löschen

Lösungsvorschlag:

Aufgrund bestehender Fremdschlüsselbeziehungen müssen zunächst die referenzierten Tupel gelöscht werden. Erst dann können die Tupel in der Person-Relation gelöscht werden:

```
DELETE FROM Schreibt
  WHERE Autor IN (SELECT PID
                  FROM Person
                  WHERE Vorname="Peter");
DELETE FROM Begutachtet
  WHERE Lektor IN (SELECT PID
                   FROM Person
                   WHERE Vorname="Peter");
DELETE FROM Person
  WHERE Vorname = "Peter";
```

- alle Tabellen zu löschen (Reihenfolge beachten!)

Lösungsvorschlag:

```
DROP TABLE Begutachtet;
DROP TABLE Schreibt;
DROP TABLE Person;
DROP TABLE Buch;
```

	Lehrveranstaltung	**Grundlagen von Datenbanken**		WS 2016/17
	Aufgabenzettel	**4 (Lösungsvorschläge)**		
	Gesamtpunktzahl	**40**		
	Ausgabe	**Mi. 30.11.2016**	Abgabe	**Fr. 16.12.2016**

3 SQL

[8 P.]

Gegeben seien die aus Aufgabe 1 bekannten Relationenschemata.

Formulieren Sie entsprechende SQL-Anweisungen für die in den nachfolgenden Teilaufgaben angeführten natürlichsprachlich formulierten Mengenbeschreibungen. **Verwenden Sie den in der Vorlesung verwendeten SQL-Standard.** Das SQL-Schlüsselwort JOIN darf dabei nicht verwendet werden.

Hinweis: Die Statements können ausprobiert werden. Hierzu wird auf der Veranstaltungsseite ein SQL-Skript angeboten, welches ein entsprechendes Schema erstellt und Daten einfügt.

a) Die PNR sowie der Nachname aller Sachbearbeiter nebst der Anzahl der von ihnen jeweils bearbeiteten Bewerbungen. [2 P.]

Lösungsvorschlag:

```
SELECT b.Sachbearbeiter, p.Nachname, COUNT(*) AS Bewerbungen
  FROM Bewerbung b, Person p
  WHERE b.Sachbearbeiter = p.PNR
  GROUP BY b.Sachbearbeiter, p.Nachname
```

Es muss mindestens nach `b.Sachbearbeiter` gruppiert werden.

b) Die PNR aller Sachbearbeiter, die mehr als 2 Bewerbungen bearbeitet haben. [2 P.]

Lösungsvorschlag:

```
SELECT b.Sachbearbeiter
  FROM Bewerbung b
  GROUP BY b.Sachbearbeiter
  HAVING COUNT(*) > 2
```

c) Die Vornamen aller Personen, die denselben Nachnamen haben wie der Sachbearbeiter einer ihrer Bewerbungen. [2 P.]

Lösungsvorschlag:

```
SELECT bew.Vorname
  FROM Bewerbung b, Person bew, Person sb
  WHERE b.Bewerber = bew.PNR
    AND b.Sachbearbeiter = sb.PNR
    AND bew.Nachname = sb.Nachname
```

d) Die PNR, Vornamen und Nachnamen aller Personen, die bisher kein Sachbearbeiter einer Bewerbung waren. [2 P.]

Lösungsvorschlag:

```
SELECT p.PNR, p.Vorname, p.Nachname
  FROM Person p
  WHERE NOT EXISTS(
      SELECT *
        FROM Bewerbung b
        WHERE p.PNR = b.Sachbearbeiter
      )
```

Oder:

```
SELECT p.PNR, p.Vorname, p.Nachname
  FROM Person p
  WHERE p.PNR NOT In(
      SELECT b.Sachbearbeiter
        FROM Bewerbung b
      )
```

4 Optimierung

[7 P.]

Gegeben seien die aus Aufgabe 1 bekannten Relationenschemata.

a) Für die nachfolgende Anfrage soll eine algebraische Optimierung durchgeführt werden. Zeichnen Sie dafür als erstes den entsprechenden Operatorbaum für die vorgegebene Anfrage und optimieren Sie diesen anschließend anhand der in der Vorlesung eingeführten Regeln (Projektionen sollen dabei jedoch **nicht** nach unten gezogen werden). [4 P.]

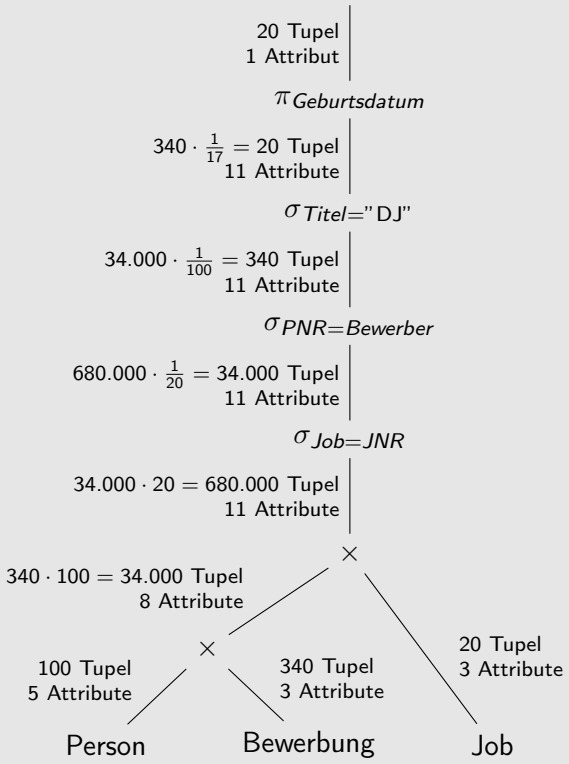
b) Bewerten Sie den Operatorbaum mit den Kardinalitäten der Zwischenergebnisse. [3 P.]

Für die zugehörige Datenbank werden folgende Kardinalitäten angenommen:
Card(*Job*) = 20, Card(*Person*) = 100, Card(*Bewerbung*) = 340.
Es gibt insgesamt 17 verschiedene Job-Titel.

$$\pi_{Geburtsdatum}\left(\sigma_{Titel=\text{"DJ"}}\left(\sigma_{PNR=Bewerber}\left(\sigma_{Job=JNR}\left((Person \times Bewerbung) \times Job\right)\right)\right)\right)$$

Lösungsvorschlag:

nicht optimiert:



optimiert:

$$\pi_{Geburtsdatum}\left(Person \underset{PNR=Bewerber}{\bowtie} \left(Bewerbung \underset{Job=JNR}{\bowtie} \sigma_{Titel=\text{"DJ"}}(Job)\right)\right)$$