	Lehrveranstaltung	Grundlagen von Datenbanken			WS 2013/14
	Aufgabenzettel	4 (Lösungsvorschläge)			
	Gesamtpunktzahl	40			
	Ausgabe	Mi. 27.11.2013	Abgabe	Do. 12.12.2013	

Aufgabe 1: Relationenalgebra

[6 P.]

Gegeben seien die folgenden Relationenschemata:

Person(PNR, Vorname, Nachname, DOB, Lieblingsobst \rightarrow Obst.ONR)

Obst(ONR, Sorte, Entdecker \rightarrow Person.PNR)

Allergie(Person \rightarrow Person.PNR, Obst \rightarrow Obst.ONR, Symptom)

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 o.Sorte
FROM Personen p, Obst o
WHERE o.Entdecker = p.PNR AND
      p.Vorname = 'Horst'
```

Lösungsvorschlag:

$$\pi_{\text{Sorte}}((\sigma_{\text{Vorname}='Horst'}(\text{Person})) \bowtie_{\text{PNR}=\text{Entdecker}} \text{Obst})$$

- b) Geben Sie einen Relationenalgebra-Ausdruck an, der die Vor- und Nachnamen aller Personen ausgibt, die eine Allergie haben, die mit dem Symptom „Halskratzen“ auftritt. [2 P.]


Lösungsvorschlag:

$$\pi_{\text{Vorname}, \text{Nachname}}(\text{Person} \bowtie_{\text{PNR}=\text{Person}} (\sigma_{\text{Symptom}='Halskratzen'}(\text{Allergie})))$$

- c) Geben Sie einen Relationenalgebra-Ausdruck an, der für jede Obstsorte die Sorte und den Nachnamen des jeweiligen Entdeckers listet, wenn die Obstsorte bei ihrem Entdecker einen Würgereiz auslöst. [2 P.]

Lösungsvorschlag:

$$\pi_{\text{Sorte}, \text{Nachname}}(\text{Obst} \bowtie_{\text{Entdecker}=\text{Person} \wedge \text{ONR}=\text{Obst}} (\sigma_{\text{Symptom}='Würgereiz'}(\text{Allergie})) \bowtie_{\text{Person}=\text{PNR}} \text{Person})$$

	Lehrveranstaltung	Grundlagen von Datenbanken			WS 2013/14
	Aufgabenzettel	4 (Lösungsvorschläge)			
	Gesamtpunktzahl	40			
	Ausgabe	Mi. 27.11.2013	Abgabe	Do. 12.12.2013	

Aufgabe 2: SQL – Schemadefinition

[18 P.]

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

Rennfahrer	<u>RID</u>	Vorname	Nachname	Geburt	Wohnort	<u>Rennstall</u>
	4	Sebastian	Vettel	1987-07-03	Kemmental (Schweiz)	2
	6	Fernando	Alonso	1981-07-29	Lugano (Schweiz)	5
	8	Marc	Webber	1976-08-27	Aston Clinton (UK)	2
	9	Lewis	Hamilton	1985-01-07	Genf (Schweiz)	31
	20	Jenson	Button	1980-01-19	Monte Carlo (Monaco)	31
	21	Felipe	Massa	1982-04-25	São Paulo (Brasilien)	5
	44	Kimi	Räikkönen	1979-10-17	Espoo (Finnland)	34


Rennstall → Rennstall.RSID

Rennstall	<u>RSID</u>	Name	Teamchef	Budget
	2	Red Bull	Christian Horner	370
	5	Ferrari	Stefano Domenicali	350
	31	McLaren	Martin Whitmarsh	220
	34	Lotus F1	Eric Boullier	100

Rennort	<u>OID</u>	Name	Strecke
	4	Australien GP	Albert Park Circuit
	15	Malaysia GP	Sepang International Circuit
	21	China GP	Shanghai International Circuit

Platzierung	<u>RID</u>	<u>OID</u>	Platz
	8	4	6
	4	15	1
	20	15	17
	4	4	3
	6	4	2
	8	15	2
	6	21	1
	9	4	5
	21	15	5
	20	4	9
	21	4	4

RID → Rennfahrer.RID, OID → Rennorte.OID

	Lehrveranstaltung	Grundlagen von Datenbanken			WS 2013/14
	Aufgabenzettel	4 (Lösungsvorschläge)			
	Gesamtpunktzahl	40			
	Ausgabe	Mi. 27.11.2013	Abgabe	Do. 12.12.2013	

Um die Konsistenz der Daten sicherzustellen, sollen folgende Integritätsbedingungen gelten:

IB1: Das Rennstallbudget muss zwischen 0 und 500 liegen.

IB2: Alle Felder bis auf Rennfahrer.Wohnort und Rennstall.Teamchef sind Pflichtfelder.

IB3: Rennstallnamen sind eindeutig.

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

[8 P.]

Lösungsvorschlag:

```


CREATE TABLE Rennstall(
  RSID      INT          PRIMARY KEY,
  Name      VARCHAR(50) UNIQUE NOT NULL,
  Teamchef   VARCHAR(50),
  Budget    INT          NOT NULL,
  CONSTRAINT check_budget CHECK (Budget > 0 AND Budget < 500)
);

CREATE TABLE Rennort(
  OID      INT          PRIMARY KEY,
  Name     VARCHAR(50) NOT NULL,
  Strecke  VARCHAR(50) NOT NULL
);

CREATE TABLE Rennfahrer(
  RID      INT          PRIMARY KEY,
  Vorname  VARCHAR(50) NOT NULL,
  Nachname VARCHAR(50) NOT NULL,
  Geburt   DATE NOT NULL,
  Wohnort  VARCHAR(50),
  Rennstall INT          NOT NULL,
  CONSTRAINT fk_rfahrer_rstall FOREIGN KEY (Rennstall) REFERENCES Rennstall (RSID)
);

CREATE TABLE Platzierung(
  RID      INT,
  OID      INT,
  Platz    INT          NOT NULL,
  CONSTRAINT pk_platzierung PRIMARY KEY (RID, OID),
  CONSTRAINT fk_platzierung_rennfahrer FOREIGN KEY (RID) REFERENCES Rennfahrer (RID),
  CONSTRAINT fk_platzierung_rennort FOREIGN KEY (OID) REFERENCES Rennort (OID)
);

```

	Lehrveranstaltung	Grundlagen von Datenbanken			WS 2013/14
	Aufgabenzettel	4 (Lösungsvorschläge)			
	Gesamtpunktzahl	40			
	Ausgabe	Mi. 27.11.2013	Abgabe	Do. 12.12.2013	

- b) In MySQL wird die referentielle Integrität von Fremdschlüsseln nicht verzögert am Ende der Transaktion (*deferred*) geprüft, sondern stets direkt. Welche Einschränkungen bringt dies mit sich?

Was müsste man bei der Definition des Schemas in SQL DDL beachten, wenn die *Rennstall*-Relation ein neues Attribut *Star* erhalten würde, welches einen Rennfahrer referenziert? Welches Problem träte bei der Manipulation der Daten in SQL DML auf?

[4 P.]

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 *Rennfahrer.Rennstall* → *Rennstall.RSID* beim Löschen einer Abteilung zuerst alle referenzierenden Rennfahrer gelöscht oder geändert werden, bevor ein Rennstall gelöscht werden kann.
- Beim Einfügen muss ebenfalls auf die richtige Reihenfolge geachtet werden, da keine Referenz auf ein nichtexistierendes Tupel angelegt werden kann.
- Das Löschen ganzer Tabellen erfordert ebenfalls die Einhaltung der Reihenfolge, die durch die Fremdschlüssel vorgegeben ist; so müsste beispielsweise die *Rennfahrer*-Relation (oder zumindest der entsprechende Fremdschlüssel) vor der *Rennstall*-Relation gelöscht werden.

Die Einführung des Attributs *Star* als Fremdschlüssel auf einen Rennfahrer würde einen Referenzzyklus erzeugen. Da Fremdschlüssel nicht *deferred* geprüft werden, können neue Einträge dann nur angelegt oder gelöscht 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 ein Foreign Key Constraint anzulegen. Sobald beide Tabellen existieren, kann über ALTER TABLE das zweite Constraint hinzugefügt werden.

	Lehrveranstaltung	Grundlagen von Datenbanken			WS 2013/14
	Aufgabenzettel	4 (Lösungsvorschläge)			
	Gesamtpunktzahl	40			
	Ausgabe	Mi. 27.11.2013	Abgabe	Do. 12.12.2013	


c) Geben Sie SQL-Befehle an, die die Datenbank mit den in der Tabelle angegebenen Datensätzen füllen. [4 P.]

Lösungsvorschlag:

```

INSERT INTO Rennort(OID, Name, Strecke) VALUES
  (4, "Australien GP", "Albert Park Circuit"),
  (15, "Malaysia GP", "Sepang International Circuit"),
  (21, "China GP", "Shanghai International Circuit");
INSERT INTO Rennstall (RSID, Name, Teamchef, Budget) VALUES
  (2, "Red Bull", "Christian Horner", 370),
  (5, "Ferrari", "Stefano Domenicali", 350),
  (31, "McLaren", "Martin Whitmarsh", 220),
  (34, "Lotus F1", "Eric Boullier", 100);
INSERT INTO Rennfahrer (RID, Vorname, Nachname, Geburt, Wohnort, Rennstall) VALUES
  (4, "Sebastian", "Vettel", "1987-07-03", "Kemmental (Schweiz)", 2),
  (6, "Fernando", "Alonso", "1981-07-29", "Lugano (Schweiz)", 5),
  (8, "Marc", "Webber", "1976-08-27", "Aston Clinton (UK)", 2),
  (9, "Lewis", "Hamilton", "1985-01-07", "Genf (Schweiz)", 31),
  (20, "Jenson", "Button", "1980-01-19", "Monte Carlo (Monaco)", 31),
  (21, "Felipe", "Massa", "1982-04-25", "São Paulo (Brasilien)", 5),
  (44, "Kimi", "Räikkönen", "1979-10-17", "Espoo (Finnland)", 34);
INSERT INTO Platzierung (RID, OID, Platz) VALUES
  (8, 4, 6),
  (4, 15, 1),
  (20, 15, 17),
  (4, 4, 3),
  (6, 4, 2),
  (8, 15, 2),
  (6, 21, 1),
  (9, 4, 5),
  (21, 15, 5),
  (20, 4, 9),
  (21, 4, 4);

```

	Lehrveranstaltung	Grundlagen von Datenbanken WS 2013/14		
	Aufgabenzettel	4 (Lösungsvorschläge)		
	Gesamtpunktzahl	40		
	Ausgabe	Mi. 27.11.2013	Abgabe	Do. 12.12.2013

d) Geben Sie SQL Befehle an, um:

[2 P.]

- alle Rennfahrer zu löschen, deren Vorname mit einem „F“ beginnt.
- alle Tabellen zu löschen.

Lösungsvorschlag:

Alle Rennfahrer löschen, deren Vorname mit einem „F“ beginnt:


```
DELETE FROM Platzierung
WHERE RID IN
(SELECT RID FROM Rennfahrer
WHERE Vorname LIKE "F%");
DELETE FROM Rennfahrer
WHERE Vorname LIKE "F%";
```

Alle Tabellen löschen:

```
DROP TABLE Platzierung;
DROP TABLE Rennfahrer;
DROP TABLE Rennstall;
DROP TABLE Rennort;
```

oder so:

```
DROP TABLE Platzierung, Rennfahrer, Rennstall, Rennort;
```

	Lehrveranstaltung	Grundlagen von Datenbanken			WS 2013/14
	Aufgabenzettel	4 (Lösungsvorschläge)			
	Gesamtpunktzahl	40			
	Ausgabe	Mi. 27.11.2013	Abgabe	Do. 12.12.2013	

Aufgabe 3: SQL – Anfragen

[10 P.]

Gegeben seien die aus Aufgabe 1 bekannten Relationenschemata.

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

- a) Alle Obstsorten, gegen die ein Peter Meyer allergisch ist, ohne Duplikate in absteigender Sortierung.

[2 P.]

Lösungsvorschlag:


```
SELECT DISTINCT o.Sorten
FROM Person p, Obst o, Allergie a
WHERE a.Person = p.PNR
      AND a.Obst = o.ONR
      AND p.Vorname = 'Peter'
      AND p.Nachname = 'Meyer'
ORDER BY o.Sorten DESC
```

- b) Die PNR, den Nachnamen und die Anzahl der Allergien jedes Allergikers.

[2 P.]

Lösungsvorschlag:

```
SELECT p.PNR, p.Nachname, COUNT(*) as Allergien
FROM Person p, Allergie a
WHERE a.Person = p.PNR
GROUP BY p.PNR, p.Nachname
```

	Lehrveranstaltung	Grundlagen von Datenbanken			WS 2013/14
	Aufgabenzettel	4 (Lösungsvorschläge)			
	Gesamtpunktzahl	40			
	Ausgabe	Mi. 27.11.2013	Abgabe	Do. 12.12.2013	

c) Die PNR aller Personen, die mehr als 6 Obstsorten entdeckt haben.

[2 P.]

Lösungsvorschlag:


```
SELECT o.Entdecker
FROM Obst o
GROUP BY o.Entdecker
HAVING COUNT(*) > 6
```

d) Vorname und Nachname aller Personen, deren Lieblingsobst von einer Person mit dem gleichen Vornamen entdeckt wurde.

[2 P.]

Lösungsvorschlag:

```
SELECT p.Vorname, p.Nachname
FROM Person p, Obst o, Person e
WHERE p.Lieblingsobst = o.ONR
      AND o.Entdecker = e.PNR
      AND e.Vorname = p.Vorname
```


	Lehrveranstaltung	Grundlagen von Datenbanken WS 2013/14		
	Aufgabenzettel	4 (Lösungsvorschläge)		
	Gesamtpunktzahl	40		
	Ausgabe	Mi. 27.11.2013	Abgabe	Do. 12.12.2013

e) Die PNR, Vorname und Nachname aller Personen, die noch keine Obstsorte entdeckt haben.


[2 P.]

Lösungsvorschlag:

```
SELECT p.PNR, p.Vorname, p.Nachname
FROM Person p
WHERE NOT EXISTS(
    SELECT *
    FROM Obst o
    WHERE p.PNR = o.Entdecker
)
```

oder:

```
SELECT p.PNR, p.Vorname, p.Nachname
FROM Person p
WHERE p.PNR NOT IN(
    SELECT o.Entdecker
    FROM Obst o
)
```

	Lehrveranstaltung	Grundlagen von Datenbanken			WS 2013/14
	Aufgabenzettel	4 (Lösungsvorschläge)			
	Gesamtpunktzahl	40			
	Ausgabe	Mi. 27.11.2013	Abgabe	Do. 12.12.2013	

Aufgabe 4: Optimierung

[6 P.]

Gegeben seien die aus Aufgabe 1 bekannten Relationenschemata:

Für die nachfolgende Anfrage soll eine algebraische Optimierung durchgeführt werden. Zeichnen Sie dafür als erstes den Operatorbaum für die vorgegebene Anfrage und optimieren Sie diese anschließend anhand der in der Vorlesung eingeführten Regeln. Bewerten Sie den Operatorbaum mit den Kardinalitäten der Zwischenergebnisse. Benutzen Sie bei SQL-Anweisungen das in der Vorlesung verwendete SQL-Erklärungsmodell, um die Anfrage in einen Operatorbaum zu überführen.

Für die zugehörige Datenbank werden folgende Kardinalitäten angenommen:

$\text{Card}(\text{Person}) = 2.000$, $\text{Card}(\text{Obst}) = 25$, $\text{Card}(\text{Allergie}) = 10.000$. Es gibt 400 verschiedene Nachnamen und jedes Obst besitzt eine eindeutige Sorte, wobei es genau 5 Obstsorten gibt, die mit einem 'K' beginnen.

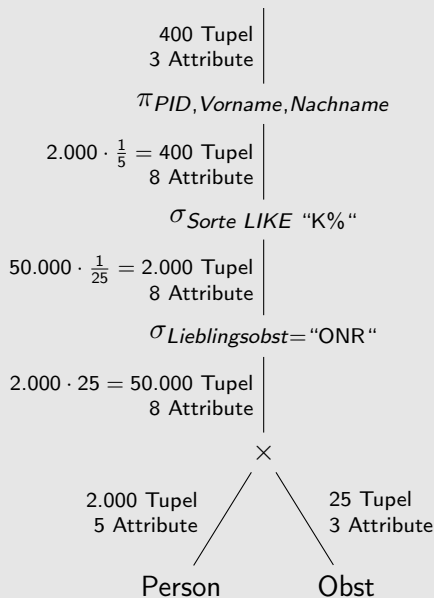
```

SELECT DISTINCT p.PNR, p.Vorname, p.Nachname
FROM Person p, Obst o
WHERE p.Lieblingsobst = o.ONR
      AND o.Sorte LIKE "K%"

```

Lösungsvorschlag:

ursprünglich:



optimiert:

