

	Lehrveranstaltung	Grundlagen von Datenbanken		WS 2019/20
	Aufgabenzettel	4 (Lösungsvorschläge - Präsenzaufgaben)		
	Gesamtpunktzahl	30		
	Ausgabe	Mi. 27.11.2019	Abgabe	Fr. 13.12.2019

1 Präsenzaufgabe: Schemadefinition

Gegeben sei folgendes Relationenschema:

Hafen(HNr, Ozean)

Schiff(Name, Tiefgang, Nutzlast, Besatzungsstaerke)

Tankschiff(Name → *Schiff*.Name, Substanz)

Containerschiff(Name → *Schiff*.Name)

Container(*Schiff* → *Containerschiff*.Name, Container)

Fahrt(FNr, Abfahrt, Ankunft, Von → *Hafen*.HNr, Nach → *Hafen*.HNr, *Schiff* → *Schiff*.Name)

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

IB1: Die Besatzungsstaerke eines Schiffes und die Ankunft einer Fahrt sind optional. Alle anderen Attribute sind verpflichtend anzugeben.

IB2: Das Abfahrtsdatum einer Fahrt muss kleiner als das aktuelle Datum (CURRENT_DATE) sein.

Geben Sie die SQL-DDL-Anweisungen an, die notwendig sind, um das DB-Schema zu erstellen. Wählen Sie dabei geeignete SQL-Standard-Datentypen. Verwenden Sie **vertikale Partitionierung**, um evtl. Vererbungen abzubilden. Beachten Sie, dass die Integritätsbedingungen zusätzlich zum Relationenschema ins SQL-Schema eingebaut werden sollen.

Lösungsvorschlag:

```
CREATE TABLE Hafen(
  HNr          int          PRIMARY KEY NOT NULL,
  Ozean       varchar(50)  NOT NULL
);
```

```
CREATE TABLE Schiff(
  Name        varchar(12)  PRIMARY KEY NOT NULL,
  Tiefgang    int          NOT NULL,
```

	Lehrveranstaltung	Grundlagen von Datenbanken		WS 2019/20
	Aufgabenzettel	4 (Lösungsvorschläge - Präsenzaufgaben)		
	Gesamtpunktzahl	30		
	Ausgabe	Mi. 27.11.2019	Abgabe	Fr. 13.12.2019

```

Nutzlast          int          NOT NULL,
Besatzungsstaerke int
);

CREATE TABLE Tankschiff(
  Name              varchar(12)  PRIMARY KEY NOT NULL,
  Substanz          varchar(50)  NOT NULL,
  CONSTRAINT fk_tankschiff_name FOREIGN KEY (Name) REFERENCES Schiff(Name)
);

CREATE TABLE Containerschiff(
  Name              varchar(12)  PRIMARY KEY NOT NULL,
  CONSTRAINT fk_containerschiff_name FOREIGN KEY (Name) REFERENCES Schiff(Name)
);

CREATE TABLE Container(
  Schiff            varchar(12)  NOT NULL,
  Container         int          NOT NULL,
  CONSTRAINT pk_container PRIMARY KEY (Schiff, Container),
  CONSTRAINT fk_container_schiff FOREIGN KEY (Schiff) REFERENCES Containerschiff(Name)
);

CREATE TABLE Fahrt(
  FNr              int          PRIMARY KEY NOT NULL,
  Abfahrt          date        NOT NULL CHECK(Abfahrt < CURRENT_DATE),
  Ankunft          date,
  Von              int          NOT NULL,
  Nach             int          NOT NULL,
  Schiff           varchar(12)  NOT NULL,
  CONSTRAINT fk_fahrt_schiff FOREIGN KEY (Schiff) REFERENCES Schiff(Name),
  CONSTRAINT fk_fahrt_von FOREIGN KEY (Von) REFERENCES Hafen(HNr),
  CONSTRAINT fk_fahrt_nach FOREIGN KEY (Nach) REFERENCES Hafen(HNr)
);

```

	Lehrveranstaltung	Grundlagen von Datenbanken		WS 2019/20
	Aufgabenzettel	4 (Lösungsvorschläge - Präsenzaufgaben)		
	Gesamtpunktzahl	30		
	Ausgabe	Mi. 27.11.2019	Abgabe	Fr. 13.12.2019

2 Präsenzaufgabe: SQL

Gegeben sei folgendes Relationenschema:

Fahrzeug(SerienNr, Modell, Hersteller → Hersteller.HNr, Fabrik → Fabrik.FNr)

Person(PNr, Vorname, Nachname, Alter, Liebblingsautomarke → Hersteller.HNr)

FZSchein(Kennzeichen, Anmeldedatum, Fahrzeug → Fahrzeug.SerienNr,
Halter → Person.PNr)

Hersteller(HNr, Name, Firmensitz, CEO → Person.PNr, GewinnInEuro)

Fabrik(FNr, Standort, Leiter → Person.PNr, Firma → Hersteller.HNr, AutosProJahr)

Hinweis: *FZSchein* steht für *Fahrzeugschein*.

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.

- a) Die Modelle von Fahrzeugen (ohne Duplikate), in deren Fahrzeugscheinen ein Halter eingetragen ist, dessen Liebblingsautomarke 'Toyota' ist. Das Ergebnis soll in alphabetischer Reihenfolge (aufsteigend) sortiert sein.

Lösungsvorschlag:

```
SELECT DISTINCT f.Modell
FROM Fahrzeug f, Person p, FZSchein fz, Hersteller h
WHERE f.SerienNr = fz.Fahrzeug
      AND p.PNr = fz.Halter
      AND p.Liebblingsautomarke = h.HNr
      AND h.Name = 'Toyota'
ORDER BY f.Modell ASC;
```

- b) Die SerienNr aller Fahrzeuge (ohne Duplikate), die laut ausgestellter Fahrzeugscheine in der Vergangenheit das gleiche Kennzeichen mit unterschiedlichen Haltern hatten.

	Lehrveranstaltung	Grundlagen von Datenbanken		WS 2019/20
	Aufgabenzettel	4 (Lösungsvorschläge - Präsenzaufgaben)		
	Gesamtpunktzahl	30		
	Ausgabe	Mi. 27.11.2019	Abgabe	Fr. 13.12.2019

Lösungsvorschlag:

```
SELECT DISTINCT f.SerienNr,
  FROM Fahrzeug f, FZschein fz1, FZschein fz2
  WHERE f.SerienNr = fz1.Fahrzeug
        AND f.SerienNr = fz2.Fahrzeug
        AND fz1.Kennzeichen = fz2.Kennzeichen
        AND fz1.Halter != fz2.Halter;
```

- c) Die PNr, Vornamen und Nachnamen aller Personen sowie die Anzahl der Fahrzeuge, die sie laut ausgestellter Fahrzeugscheine bisher angemeldet hat.

Lösungsvorschlag:

```
SELECT p.PNr, p.Vorname, p.Nachname, COUNT(fz.Fahrzeug) AS AnzahlFahrzeuge
  FROM Person p, FZSchein fz
  WHERE p.PNr = fz.Halter
  GROUP BY p.PNr, p.Vorname, p.Nachname;
```

Ein GROUP BY nur mit der p.PNr ist auch in Ordnung.

Anmerkung: In der ursprünglichen Präsenzlösung befand sich ein Fehler. In dem unten stehenden SELECT-Statement wird die Anzahl der Fahrzeugscheine gezählt. Da mehrere Fahrzeugscheine aber zu ein und dasselben Fahrzeug gehören können (bspw. bei unterschiedlichen Kennzeichen oder einem anderen Anmeldedatum, muss (wie oben angegeben) nach dem Fahrzeug gruppiert werden.

```
SELECT p.PNr, p.Vorname, p.Nachname, COUNT(*) AS AnzahlFahrzeuge
  FROM Person p, FZSchein fz
  WHERE p.PNr = fz.Halter
  GROUP BY p.PNr, p.Vorname, p.Nachname;
```

- d) **Optional:**

PNrs, Vor- und Nachnamen aller CEOs von Herstellern, die keine Fabrik leiten.

Lösungsvorschlag:

```
SELECT p.PNr, p.Vorname, p.Nachname
  FROM Person p, Hersteller h
  WHERE p.PNr = h.CEO
  AND NOT EXISTS(
```

	Lehrveranstaltung	Grundlagen von Datenbanken		WS 2019/20
	Aufgabenzettel	4 (Lösungsvorschläge - Präsenzaufgaben)		
	Gesamtpunktzahl	30		
	Ausgabe	Mi. 27.11.2019	Abgabe	Fr. 13.12.2019

```

SELECT *
  FROM Fabrik f
 WHERE p.PNR = f.Leiter
)

```

oder

```

SELECT p.PNR, p.Vorname, p.Nachname
  FROM Person p, Hersteller h
 WHERE p.PNr = h.CEO
 AND p.PNR NOT IN(
   SELECT Leiter
   FROM Fabrik
 )

```

e) **Optional:**

Die HNrs und die Namen aller Hersteller, die in ihren Fabriken insgesamt mindestens 2,5 Mio Fahrzeuge pro Jahr herstellen.

Lösungsvorschlag:

```

SELECT h.HNr, h.Name
  FROM Hersteller h, Fabrik f
 WHERE h.HNr = f.Firma
 GROUP BY h.HNr
 HAVING SUM(f.AutosProJahr) >= 2500000;

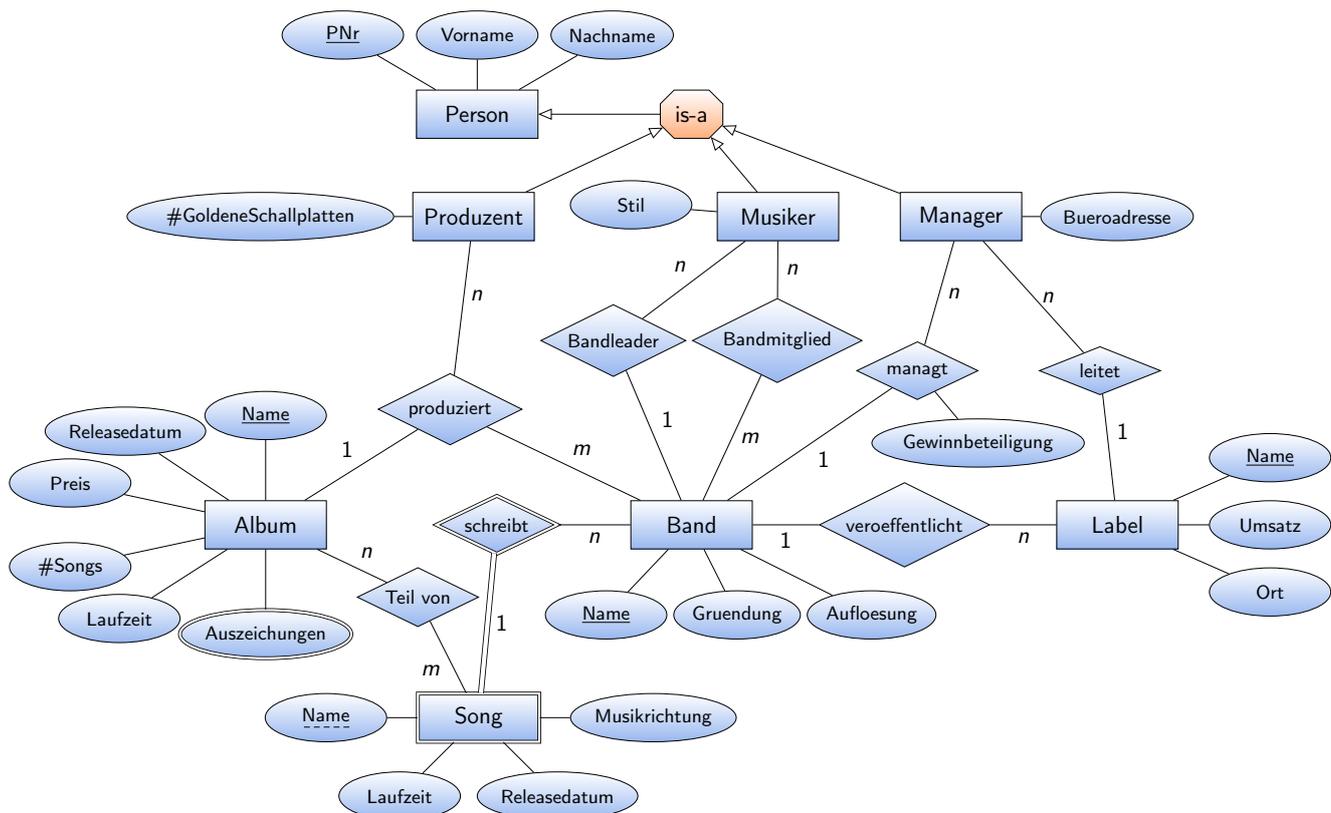
```

	Lehrveranstaltung	Grundlagen von Datenbanken		WS 2019/20
	Aufgabenzettel	4 (Lösungsvorschläge - Präsenzaufgaben)		
	Gesamtpunktzahl	30		
	Ausgabe	Mi. 27.11.2019	Abgabe	Fr. 13.12.2019

3 Übungsaufgabe: Schemadefinition

[20 P.]

Gegeben sei folgendes Entity-Relationship-Diagramm:



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

- IB1:** Der Preis eines Albums muss größer oder gleich 5 (Euro) sein.
- IB2:** Der Umsatz eines Labels und das Auflösungsdatum einer Band sind optional. Alle anderen einwertigen Attribute sind verpflichtend anzugeben.
- IB3:** Das Auflösungsdatum einer Band muss (sofern angegeben) größer sein als das Gründungsdatum.
- IB4:** Die Büroadresse des Managers soll eindeutig sein.

Geben Sie die SQL-DDL-Anweisungen an, die notwendig sind, um das DB-Schema zu erstellen. Wählen Sie dabei geeignete SQL-Standard-Datentypen. Verwenden Sie **vertikale Partitionierung**, um evtl. Vererbungen abzubilden. Beachten Sie, dass die Abbildungstypen durch geeignete Constraints exakt abzubilden sind. Testen Sie die SQL-Ausdrücke auf der Übungsdatenbank.

	Lehrveranstaltung	Grundlagen von Datenbanken		WS 2019/20
	Aufgabenzettel	4 (Lösungsvorschläge - Präsenzaufgaben)		
	Gesamtpunktzahl	30		
	Ausgabe	Mi. 27.11.2019	Abgabe	Fr. 13.12.2019

4 Übungsaufgabe: Optimierung

[10 P.]

Gegeben sei folgendes Relationenschema:

Eigentümer(EID, Vorname, Nachname, Vermoegen, Wohnsitz → *Immobilie*.IID)

Handwerksfirma(HID, Name, Stundensatz, Fachgebiet, Firmensitz → *Immobilie*.IID)

Immobilie(IID, Adresse, Stockwerke, Baujahr, Bauherr → *Eigentuemer*.EID,
Baufirma → *Handwerksfirma*.HID, Eigentuemer → *Eigentuemer*.EID)

Auftrag(Vermieter → *Eigentuemer*.EID, Handwerker → *Handwerksfirma*.HID,
Immobilie → *Immobilie*.IID, Von, Bis, Beschreibung, Kosten)

- 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 indem Sie den optimierten Operatorbaum zeichnen (Projektionen sollen dabei jedoch **nicht** nach unten gezogen werden.) [7 P.]
- b) Bewerten Sie die beiden Operatorbäume mit den Kardinalitäten der Zwischenergebnisse. (Die Anzahl der Attribute soll dabei **nicht** betrachtet werden.) Für die zugehörige Datenbank werden folgende Kardinalitäten angenommen: [3 P.]
 $Card(Eigentuemer) = 50$, $Card(Immobilie) = 1000$ und $Card(Handwekrsfirma) = 20$. Unter den Handwekrsfirmen gibt es nur 2 verschiedene Fachgebiete wobei von denen jede gleich oft auftritt.

$$\pi_{Name}(\sigma_{Fachgebiet="Maurer"}(\sigma_{Wohnsitz=IID}(\sigma_{Baufirma=HID}(((\sigma_{EID=1}(Eigentuemer) \times Immobilie) \cup (\sigma_{EID=2}(Eigentuemer) \times Immobilie)) \times Handwerksfirma))))))$$