Friedrich-Alexander-Universität
Technische Fakultät

Department Informatik
Lehrstuhl für Informatik 6
(Datenmanagement)

Einführung in Datenbanksysteme
Sommersemester 2025

Übungsblatt 06: Structured Query Language (Teil 1)

1 Allgemeine Hinweise zur Bearbeitung

Die nachfolgende Aufgabensammlung bieten Ihnen die Möglichkeit, die in der Vorlesung behandelten Sprachkonstrukte der Structured Query Language praktisch auszuprobieren.

Aufgrund der großen Anzahl der bereitgestellten Aufgaben können **nur ausgewählte Aufgaben in den Präsenzübungen** behandelt werden. Deren Aufgabenstellungen sind entsprechend markiert. Die Aufgaben werden **über zwei Übungswochen** (Teil 1 und Teil 2) verteilt behandelt.

Wir erwarten von Ihnen jedoch, dass Sie **alle der nachfolgenden Aufgaben eigenständig bearbeiten**. Für alle Aufgaben, die nicht in den Präsenzübungen behandelt werden, stellen wir Ihnen wie gewohnt Lösungsvorschläge auf StudOn bereit. Dabei handelt es sich ausdrücklich um Vorschläge, da alle Aufgaben auf verschiedene Arten gelöst werden können.

In der Modulprüfung am Ende des Semesters müssen Sie in der Lage sein, Aufgaben von vergleichbarer Komplexität innerhalb weniger Minuten syntaktisch und semantisch korrekt zu lösen. Dies kann Ihnen nur gelingen, wenn Sie sich vorher umfassend und eigenständig mit SQL beschäftigt haben. Am Ende dieses Übungsblattes finden Sie daher auch noch weitere Quellen zur Vertiefung Ihrer SQL-Kenntnisse.

2 Technische Arbeitsumgebung

Die nachfolgenden Aufgaben beziehen sich auf das Datenbankmanagementsystem MySQL. Auf relevante Abweichungen vom SQL-Standard wird direkt in den Aufgabenstellungen gesondert hingewiesen.

Für die Installation unter Windows empfehlen wir Ihnen den **MySQL Installer**. Dieser enthält neben dem MySQL-Server auch die grafische MySQL-Workbench. In StudOn stellen wir Ihnen einen **Screencast** bereit, der die Installation und grundlegende Einrichtung demonstriert. MySQL ist ebenso für Linux verfügbar. Viele gängige Distributionen stellen dazu entsprechende Pakete bereit. Bitte konsultieren Sie die Dokumentation Ihrer verwendeten Distribution für weitere Details. Neben der Installation des MySQL-Servers empfehlen wir Ihnen, die MySQL Workbench als grafisches Werkzeug für den Umgang mit dem DBMS zu installieren. Eine Installation von MySQL ist auch unter macOS möglich. Auch hierfür haben wir Ihnen in StudOn einen **Screencast** bereitgestellt.

Wichtige Links:

- Downloadseite: `http://dev.mysql.com/downloads/mysql/`
- Installationsanleitung: `https://dev.mysql.com/doc/refman/8.0/en/installing.html`
- Dokumentation: `https://dev.mysql.com/doc/refman/8.0/en/`

3 Support

Nutzen Sie bitte das Forum der FSI Informatik, falls bei der Verwendung von MySQL Fehler auftreten. Geben Sie in Ihrem Posting unbedingt folgende Informationen an: Betriebssystem, verwendetes Client-Tool, ausgeführtes Skript/abgesetzte SQL-Befehle. Nur so kann Ihnen geholfen werden!
Hinweis: Versuchen Sie Ihre Probleme immer zuerst selbst mit Hilfe der Dokumentation von MySQL zu lösen, so lernen Sie am meisten!

4 Erläuterungen zur Beispieldatenbank

Die Beispieldatenbank in diesem Buch versucht, eine Versicherungsgesellschaft für Kfz-Versicherungen abzubilden. Das Beispiel ist eine starke Vereinfachung der Realität; zum Beispiel fehlen alle Teile der laufenden Abrechnung der Prämien und Schadensfälle. Die relevanten Begriffe sind für eine bessere Übersichtlichkeit fett gekennzeichnet. Folgender

Sachverhalt wird in der Datenbank abgebildet: **Die Versicherungsgesellschaft *UnsereFirma* verwaltet mit dieser Datenbank ihre Kundendaten und die Schadensfälle. Wegen der Schadensfälle müssen auch Daten „fremder“ Kunden und Versicherungen gespeichert werden.**

1. Jeder **Versicherungsvertrag** wird mit einem Versicherungsnehmer über genau ein Fahrzeug abgeschlossen. Der Versicherungsvertrag wird durch folgende Eigenschaften gekennzeichnet; zu jeder Eigenschaft gehört eine Spalte:
   1) Vertragsnummer
   2) Datum des Abschlusses, Art des Vertrages; dabei gibt es die Arten Haftpflicht (HP), Haftpflicht mit Teilkasko (TK), Vollkasko (VK).
   3) Verweis auf den Versicherungsnehmer
   4) Verweis auf das Fahrzeug
   5) Verweis auf den Mitarbeiter, der den Vertrag bearbeitet
   6) Basisprämie, Prämiensatz und das Datum der letzten Prämienänderung
2. Der **Versicherungsnehmer** ist gekennzeichnet durch diese Eigenschaften:
   1) Kundennummer
   2) Name und Anschrift: PLZ, Ort, Straße, Hausnummer
   3) bei natürlichen Personen zusätzlich durch Vorname, Geburtsdatum, Geschlecht, Datum des Führerscheins
   4) Kennzeichnung ob es sich um einen eigenen Kunden handelt, oder um den einer fremden Versicherungsgesellschaft (Flag "J/"N")
   5) Verweis auf eine „Fremdversicherung“, wenn ein (fremdes) Fahrzeug an einem Unfall beteiligt ist
3. Das **Fahrzeug** ist gekennzeichnet durch diese Eigenschaften:
   1) polizeiliches Kennzeichen
   2) Farbe
   3) Fahrzeugtyp und damit indirekt auch den Fahrzeughersteller
4. Ein **Mitarbeiter** ist gekennzeichnet durch diese Eigenschaften:
   1) Name, Vorname, Geburtsdatum
   2) Personalnummer
   3) Verweis auf Abteilung, Vermerk, ob es sich um den Leiter der Abteilung handelt (Flag "J/"N")

4) Kontaktdaten wie Telefon, Mobiltelefon, Email, Raum

5) Geschlecht

5. Die **Abteilung** ist gekennzeichnet durch diese Eigenschaften:

   1) Nummer

   2) Kurzbezeichnung, Bezeichnung

   3) Ort

6. Zusätzlich gibt es **Dienstwagen**. Dabei handelt es sich um eine Tabelle mit den gleichen Eigenschaften wie bei Fahrzeug und zusätzlich:

   1) Verweis auf den Mitarbeiter, zu dem ein Dienstwagen gehört

   2) denn es gibt auch Dienstwagen, die keinem Mitarbeiter persönlich zugeordnet sind

7. Ein **Schadensfall** ist gekennzeichnet durch diese Eigenschaften:

   1) Datum, Ort und Umstände des Unfalls

   2) Vermerk, ob es Verletzte gab, sowie Höhe des Gesamtschadens

   3) Verweis auf den Mitarbeiter, der den Schadensfall bearbeitet

8. An einem Schadensfall können mehrere Fahrzeuge unterschiedlicher Versicherungen beteiligt sein. (Unfälle mit Radfahrern und Fußgängern werden nicht betrachtet.) Deshalb gibt es eine weitere Tabelle **Zuordnung_SF_FZ**:

   1) Liste aller Schadensfälle und aller beteiligten Fahrzeuge

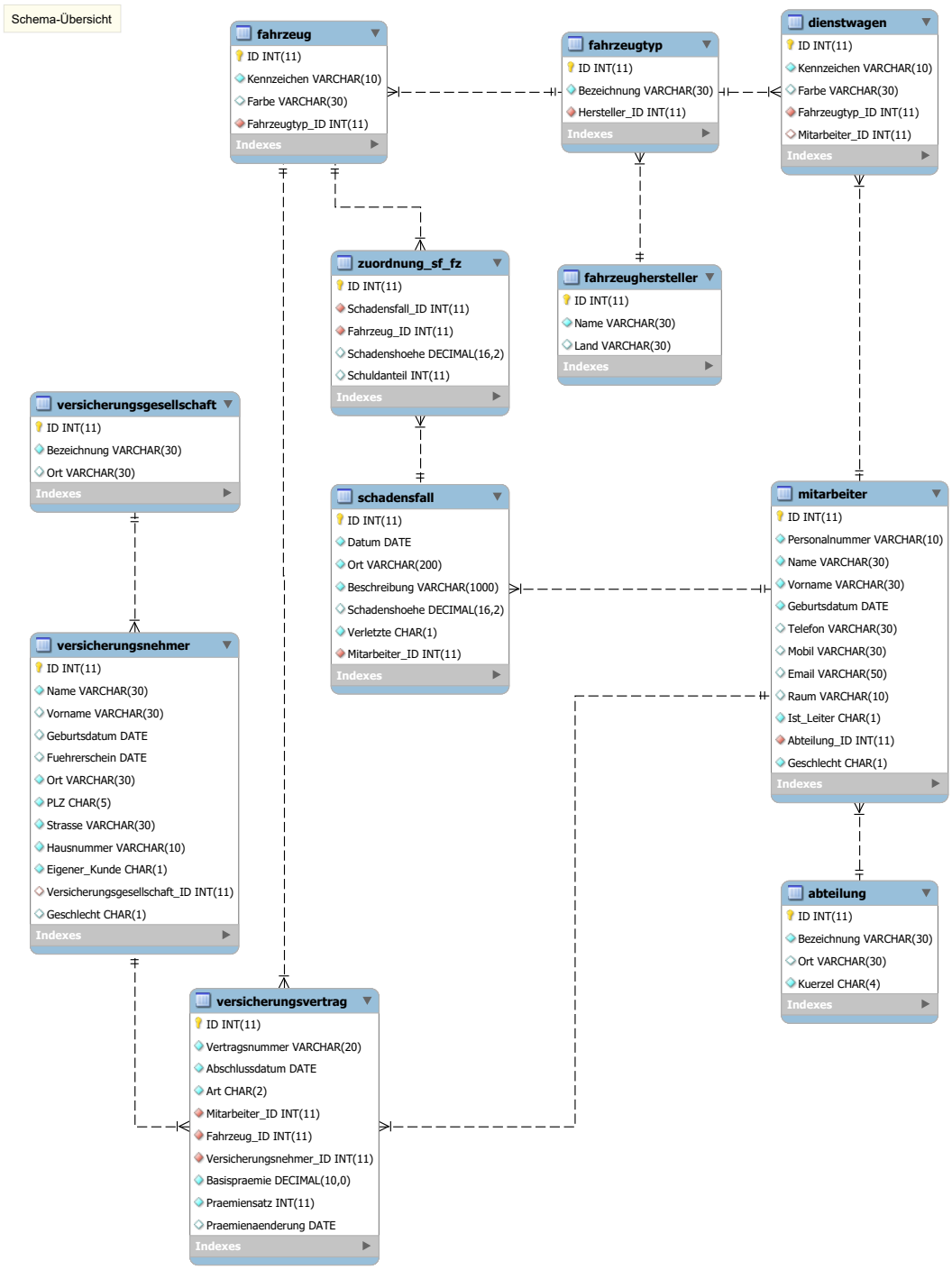
   2) anteilige Schadenshöhe eines Fahrzeugs an dem betreffenden Unfall

   3) Verweis auf den Mitarbeiter, der den Schadensfall bearbeitet

   4) Anteil am Verschulden

   Über die Verbindung Schadensfall → Fahrzeug → Versicherungsvertrag → Versicherungsnehmer → Versicherungsgesellschaft können alle beteiligten Gesellschaften festgestellt und in die Schadensabwicklung eingebunden werden.

Anschließend finden Sie zusätzlich eine grafische Darstellung aller Tabellen und Attribute. Dabei wird die Ihnen aus der Vorlesung bekannte Krähenfuß-Notation verwendet. Diese Darstellung können Sie sich mit Hilfe der MySQL-Workbench auch selbst generieren (`Database` → `Reverse Engineer`, Kürzel `Strg+R`).


Schema-Übersicht
fahrzeug
ID INT(11)
Kennzeichen VARCHAR(10)
Farbe VARCHAR(30)
Fahrzeugtyp_ID INT(11)
Indexes
fahrzeugtyp
ID INT(11)
Bezeichnung VARCHAR(30)
Hersteller_ID INT(11)
Indexes
dienstwagen
ID INT(11)
Kennzeichen VARCHAR(10)
Farbe VARCHAR(30)
Fahrzeugtyp_ID INT(11)
Mitarbeiter_ID INT(11)
Indexes
zuordnung_sf_fz
ID INT(11)
Schadensfall_ID INT(11)
Fahrzeug_ID INT(11)
Schadenshoehe DECIMAL(16,2)
Schuldanteil INT(11)
Indexes
fahrzeughersteller
ID INT(11)
Name VARCHAR(30)
Land VARCHAR(30)
Indexes
versicherungsgesellschaft
ID INT(11)
Bezeichnung VARCHAR(30)
Ort VARCHAR(30)
Indexes
schadensfall
ID INT(11)
Datum DATE
Ort VARCHAR(200)
Beschreibung VARCHAR(1000)
Schadenshoehe DECIMAL(16,2)
Verletzte CHAR(1)
Mitarbeiter_ID INT(11)
Indexes
mitarbeiter
ID INT(11)
Personalnummer VARCHAR(10)
Name VARCHAR(30)
Vorname VARCHAR(30)
Geburtsdatum DATE
Telefon VARCHAR(30)
Mobil VARCHAR(30)
Email VARCHAR(50)
Raum VARCHAR(10)
Ist_Leiter CHAR(1)
Abteilung_ID INT(11)
Geschlecht CHAR(1)
Indexes
versicherungsnehmer
ID INT(11)
Name VARCHAR(30)
Vorname VARCHAR(30)
Geburtsdatum DATE
Fuehrerschein DATE
Ort VARCHAR(30)
PLZ CHAR(5)
Strasse VARCHAR(30)
Hausnummer VARCHAR(10)
Eigener_Kunde CHAR(1)
Versicherungsgesellschaft_ID INT(11)
Geschlecht CHAR(1)
Indexes
abteilung
ID INT(11)
Bezeichnung VARCHAR(30)
Ort VARCHAR(30)
Kuerzel CHAR(4)
Indexes
versicherungsvertrag
ID INT(11)
Vertragsnummer VARCHAR(20)
Abschlussdatum DATE
Art CHAR(2)
Mitarbeiter_ID INT(11)
Fahrzeug_ID INT(11)
Versicherungsnehmer_ID INT(11)
Basispraemie DECIMAL(10,0)
Praemiensatz INT(11)
Praemienaenderung DATE
Indexes

5 Aufgaben für die erste SQL-Übungswoche

Wir erwarten von Ihnen, dass Sie mindestens die nachfolgenden Aufgaben für die erste Woche der SQL-Übungen eigenständig lösen.

5.1 Anlegen der Datenbank

Verwenden Sie die in StudOn bereitgestellte Datei **`Versicherung.sql`**, um Ihre Datenbank mit dem benötigten Schema und einigen Testdaten zu füllen. Sie können diese Datei auch verwenden, um die Datenbank bei unbeabsichtigten Änderungen wieder auf einen definierten Zustand zurückzusetzen.

Folgen Sie dem Screencast aus StudOn, um Ihre Datenbank unter Verwendung der MySQL-Workbench zu initialisieren.

5.2 Allgemeine Fragen

1. Welche der folgenden Spaltenlisten aus der Beispieldatenbank sind richtig, welche nicht?

   1) `SELECT * FROM Mitarbeiter;`
   2) `SELECT ID, Name FROM Mitarbeiter;`
   3) `SELECT ID, Name FROM Mitarbeiter, Abteilung;`
   4) `SELECT ID, Name, Kuerzel FROM Mitarbeiter, Abteilung;`
   5) `SELECT ab.ID, Name FROM Mitarbeiter, Abteilung ab;`
   6) `SELECT ab.ID, Name, Krz Kuerzel FROM Mitarbeiter, Abteilung ab;`
   7) `SELECT ab.ID, Name, Kuerzel Krz FROM Mitarbeiter, Abteilung ab;`
   8) `SELECT ab.ID, mi.Name, ab.Kuerzel FROM Mitarbeiter mi, Abteilung ab;`

2. Welche der folgenden Selektionsbedingungen für die Tabelle `Mitarbeiter` sind richtig, welche nicht? Welche korrekten Bedingungen liefern immer `FALSE` als Ergebnis? Einige der Selektionsbedingungen können Sie aufgrund Ihres Wissens aus der Vorlesung beurteilen, bei anderen können Sie mögliche Fehlermeldungen des DBMS auswerten.

   1) `WHERE Name NOT = 'Meyer';`

2) `WHERE 1 = 2;`

3) `WHERE NOT Name LIKE 'M%';`
(Hinweis: Recherchieren Sie eigenständig die Semantik von `LIKE`)

4) `WHERE ID BETWEEN 20 AND 10;`
(Hinweis: Recherchieren Sie eigenständig die Semantik von `BETWEEN .. AND ..`)

5) `WHERE Name LIKE 'L%' AND LIKE '_a%';`

6) `WHERE ID IN (1, 3, 'A');`

7) `WHERE ID IN (1, 3, '5');`

3. **[Besprechung Ihrer Lösung in der Präsenzübung]** Nennen Sie eine gültige Reihenfolge der `INSERT`-Befehle, wenn ein Schadensfall mit drei beteiligten Fahrzeugen aufzunehmen ist. Dabei soll ein Fahrzeug zu einem „eigenen Kunden“ und zwei Fahrzeuge zu „Fremdkunden“ gehören; die eine „fremde“ Versicherungsgesellschaft soll schon gespeichert sein, die andere nicht.

4. Welche der folgenden Aussagen sind wahr, welche sind falsch?

   1) Um alle Mitarbeiter mit Dienstwagen aufzulisten, benötigt man einen `LEFT OUTER JOIN`.

   2) `LEFT JOIN` ist nur eine Kurzschreibweise für `LEFT OUTER JOIN` und hat keine zusätzliche inhaltliche Bedeutung.

   3) Ein `LEFT JOIN` von zwei Tabellen enthält alle Zeilen, die nach Auswahlbedingung in der linken Tabelle enthalten sind.

   4) Ein `RIGHT JOIN` von zwei Tabellen enthält nur noch diejenigen Zeilen, die nach der Verknüpfungsbedingung in der linken Tabelle enthalten sind.

   5) Wenn wir bei einer `LEFT JOIN`-Abfrage mit zwei Tabellen die beiden Tabellen vertauschen und stattdessen einen `RIGHT JOIN` verwenden, erhalten wir dieselben Zeilen in der Ergebnismenge.

   6) Wir erhalten dabei nicht nur dieselben Zeilen, sondern auch dieselbe Reihenfolge.

5. Welche der folgenden Aussagen sind wahr, welche falsch?

   1) Die SQL-Schlüsselwörter `JOIN` und `INNER JOIN` sind äquivalent.

   2) Eine Tabelle kann mit sich selbst verknüpft werden.

   3) SELF JOIN ist nur ein inhaltlicher Begriff, aber kein SQL-Schlüsselwort.

   4) Bei einem SELF JOIN sind nur `INNER JOIN`s erlaubt.

5) Eine bestimmte Tabelle darf in einem SELF JOIN nur zweimal verwendet werden.

6) Für einen SELF JOIN können Tabellen-Aliase benutzt werden, aber sie sind nicht überall erforderlich.

7) Ein `CROSS JOIN` ist eine Verknüpfung zweier Tabellen ohne Verknüpfungsbedingung.

8) Bei einem `CROSS JOIN` darf sich die `WHERE`-Klausel nicht auf die (rechte) Tabelle des `JOIN`s beziehen.

6. **[Besprechung Ihrer Lösung in der Präsenzübung]** Liefern folgende Queries (bei beliebigem Zustand der Datenbank) unterschiedliche Ergebnisse? Warum bzw. warum nicht?

```
   SELECT *
     FROM Versicherungsvertrag vv
LEFT JOIN Versicherungsnehmer vn
       ON (vv.Versicherungsnehmer_ID = vn.ID
           AND vn.ID > 5)

   SELECT *
     FROM Versicherungsvertrag vv
LEFT JOIN Versicherungsnehmer vn
       ON (vv.Versicherungsnehmer_ID = vn.ID)
    WHERE vn.ID > 5
```

Ja, die liefern unterschiedliche Ergebnisse, da die Folge der Ausführung anders ist.
Bei dem zweiten Query wird erst WHERE ausgeführt.

5.3 Fehlersuche

1. Was ist an dieser Anfrage falsch?

```
SELECT ID, Abschlussdatum, Art,
       vv.Kennzeichen, Farbe
  FROM Versicherungsvertrag vv, Fahrzeug
 WHERE vv.Fahrzeug_ID = Fahrzeug.ID
   AND Kennzeichen LIKE 'BO%';
```

ID ist nicht eindeutig, muss explizit stehen, von welchem ID man redet.

2. **[Besprechung Ihrer Lösung in der Präsenzübung]** Was ist an dieser Anfrage falsch?

```
SELECT ID, Vorname + ' ' + Name AS Kunde, Ort,
       Name AS Sachbearbeiter, Telefon
  FROM Versicherungsvertrag, Versicherungsnehmer, Mitarbeiter
 WHERE ID = 27
   AND Versicherungsvertrag.Versicherungsnehmer_ID
       = Versicherungsnehmer.ID
   AND Versicherungsvertrag.Mitarbeiter_ID = Mitarbeiter.ID;
```

3. Zeigen Sie zu jedem Mitarbeiter die Daten seines Dienstwagens (Kennzeichen, Typ, Hersteller) an. Berichtigen Sie dazu den folgenden SELECT-Befehl.

```
SELECT ID, Name, Vorname,
       Kennzeichen, Bezeichnung, Name
  FROM Mitarbeiter mi, Dienstwagen dw,
       Fahrzeugtyp ft, Fahrzeughersteller fh
 WHERE ID = dw.Mitarbeiter_ID
   AND ID = dw.Fahrzeugtyp_ID
   AND ID = ft.Hersteller_ID;
```

5.4 Anfrageformulierung

1. Wir benötigen zu allen Abteilungen die ID der Abteilung und die Anzahl der zugehörigen Mitarbeiter pro Abteilung. Sortieren Sie das Ergebnis absteigend nach der Anzahl der Mitarbeiter.
2. Erstellen Sie mit Hilfe eines Mengenquantors eine Liste der IDs aller Mitarbeiter, die denselben Vornamen wie mindestens ein Versicherungsnehmer haben.
3. **[Besprechung Ihrer Lösung in der Präsenzübung]** Erstellen Sie unter Verwendung eines Existenzquantors eine aufsteigend sortierte, wiederholungsfreie Liste aller Geburtsdaten von Versicherungsnehmern. Berücksichtigen Sie dabei nur Versicherungsnehmer, die aus Orten stammen, in denen mindestens eine Abteilung ansässig ist.
4. Suchen Sie alle Versicherungsnehmer, die folgenden Bedingungen entsprechen: Der erste Buchstabe des Namens ist nicht bekannt, der zweite ist ein 'r'. Der Vorname enthält ein 'a'. Die Postleitzahl gehört zum Bereich Essen (PLZ 45...).
5. Wir benötigen eine Liste aller Orte, in denen mehr als ein Versicherungsnehmer ansässig ist. Neben dem Städtenamen ist dann die Anzahl dortigen Versicherungsnehmer interessant. Die Sortierung erfolgt nach der Anzahl absteigend.

6. Formulieren Sie eine Abfrage zur Tabelle `Versicherungsvertrag`, die nur die wichtigsten Informationen (Vertragsnummer, Abschlussdatum, Art, sowie die Fremdschlüssel auf andere Tabellen) ausgibt. Wie viele Ergebnistupel erhalten Sie?

7. Erweitern Sie die vorherige Anfrage, sodass anstelle der ID des Versicherungsnehmers dessen Name und Vorname angezeigt werden. Verzichten Sie auf eine `WHERE`-Klausel. Wie viele Ergebnistupel erhalten Sie?

8. Erweitern Sie die Anfrage 7, sodass anstelle der ID des Fahrzeugs das Kennzeichen und anstelle der ID des Mitarbeiters dessen Name und Vorname angezeigt werden. Verzichten Sie auf eine `WHERE`-Klausel. Wie viele Ergebnistupel erhalten Sie?

9. Erweitern Sie die Anfrage 7, sodass Name und Vorname des Versicherungsnehmers genau zum jeweils passenden Vertrag ausgegeben werden. Wie viele Ergebnistupel erhalten Sie?

10. Erweitern Sie die Anfrage 9, sodass Name und Vorname des Mitarbeiters sowie das Fahrzeug-Kennzeichen genau zum jeweils passenden Vertrag ausgegeben werden. Wie viele Einträge zeigt die Ergebnismenge an?

11. [**Besprechung Ihrer Lösung in der Präsenzübung**] Erweitern Sie die vorherige Anfrage, sodass die ausgewählten Zeilen den folgenden Bedingungen entsprechen: Es geht ausschließlich um eigene Kunden (Eigener_kunde = 'J'). Vollkasko-Verträge sollen immer angezeigt werden, ebenso Fahrzeuge aus dem Kreis Recklinghausen 'RE'. Teilkasko-Verträge sollen angezeigt werden, wenn sie nach '1990-12-31' abgeschlossen wurden. Haftpflicht-Verträge sollen angezeigt werden, wenn sie nach '1985-12-31' abgeschlossen wurden. Wie viele Einträge zeigt die Ergebnismenge an?

12. Zeigen Sie alle Mitarbeiter der Abteilungen „Vertrieb“ (Kürzel 'Vert') und „Ausbildung“ (Kürzel 'Ausb') an.
    Bestimmen Sie dazu zunächst die IDs der gesuchten Abteilungen in einer Unteranfrage und benutzen Sie das Ergebnis für die eigentliche Anfrage.

13. Zu jedem Eintrag der Tabelle `Versicherungsvertrag` sollen Vertragsnummer, Basisprämie und Prämiensatz angegeben sowie die aktuelle Prämie (Basisprämie * Prämiensatz / 100) berechnet werden.

14. Geben Sie die Gesamtzahl der Versicherungsverträge sowie den Gesamtbetrag aller aktuellen Prämien an.

15. [**Besprechung Ihrer Lösung in der Präsenzübung**] Suchen Sie zu jedem Mitarbeiter den Namen und Vornamen des Leiters der zugehörigen Abteilung (Ist_Leiter = 'J'). Die Abteilungsleiter in unserer einfachen Firmenhierarchie haben keinen Vorgesetzten; sie sollen in der Liste deshalb nicht aufgeführt werden.

16. **[Besprechung Ihrer Lösung in der Präsenzübung]** Suchen Sie Einträge in der Tabelle `Versicherungsnehmer`, bei denen Name, Vorname, PLZ, Strasse übereinstimmen. Jeweils zwei dieser Adressen sollen mit ihrer ID und den übereinstimmenden Angaben aufgeführt werden.
Hinweis: Benutzen Sie einen JOIN, der sich nicht auf übereinstimmende IDs bezieht.

5.5 Datenmodifikation

1. Fügen Sie sich selbst als Mitarbeiter mit der ID 29 in den vorhandenen Datenbestand ein.
2. Entfernen Sie Ihren soeben angelegten Eintrag (und nur diesen) wieder aus der Datenbank.
3. Überlegen Sie sich, was passiert, wenn Sie die folgende Anfrage ausführen:

```
DELETE FROM Mitarbeiter;
```

4. **[Besprechung Ihrer Lösung in der Präsenzübung]** Von der Deutschen Post AG wird eine Tabelle `plz_aenderung`(ID, PLZalt, Ortalt, PLZneu, Ortneu) geliefert:

```
CREATE TABLE PLZ_Aenderung (
  ID INT NOT NULL AUTO_INCREMENT,
  PLZalt INT NOT NULL,
  Ortalt VARCHAR(64) NOT NULL,
  PLZNeu INT NOT NULL,
  Ortneu VARCHAR(64) NOT NULL,
  PRIMARY KEY (ID));

INSERT INTO PLZ_Aenderung(ID, PLZalt, Ortalt, PLZNeu, Ortneu)
            VALUES ('1', '45658', 'Recklinghausen', '45659'
                    , 'Recklinghausen');
INSERT INTO PLZ_Aenderung(ID, PLZalt, Ortalt, PLZNeu, Ortneu)
            VALUES ('2', '45721', 'Hamm-Bossendorf', '45721'
                    , 'Haltern-Hamm');
INSERT INTO PLZ_Aenderung(ID, PLZalt, Ortalt, PLZNeu, Ortneu)
            VALUES ('3', '45772', 'Marl', '45770', 'Marl');
INSERT INTO PLZ_Aenderung(ID, PLZalt, Ortalt, PLZNeu, Ortneu)
            VALUES ('4', '45701', 'Herten', '45699', 'Herten');
```

Ändern Sie die Tabelle `Versicherungsnehmer` so, dass bei Adressen, bei denen `PLZ/Ort` mit `PLZalt/Ortalt` übereinstimmen, diese Angaben durch `PLZneu/Ortneu` geändert werden. Beschränken Sie sich auf die Änderung mit der `ID` 3.


```
UPDATE Versicherungsnehmer
SET PLZ = (SELECT PLZneu FROM PLZ_Aenderung WHERE ID = 3),
ORT = (SELECT Ortneu FROM PLZ_Aenderung WHERE ID = 3)
WHERE PLZ = (SELECT PLZalt FROM PLZ_Aenderung WHERE ID = 3)
AND ORT = (SELECT Ortneu FROM PLZ_Aenderung WHERE ID = 3);
```