4. Übungsblatt (WS 2019)

3.0 VU Datenmodellierung 2 / 6.0 VU Datenbanksysteme

Informationen zum Übungsblatt

Allgemeines

Inhalt des vierten Übungsblattes sind die (vertiefenden) Funktionen von SQL. Sie werden das Erstellen eines Datenbankschemas üben, das Definieren und Sicherstellen der Datenintegrität, sowie das Schreiben komplexerer SQL Anfragen (inklusive prozeduraler Programme).

In dieser Übung geben Sie eine einzige ZIP Datei ab (max. 5MB). Diese ZIP Datei enthält alle notwendigen SQL Dateien zum Erstellen und Testen Ihrer Datenbank, siehe Aufgabe 6. Zum Testen ihrer Dateien stellen wir Ihnen einen PostgreSQL Server (Version 11.5) zur Verfügung. Sie können sich dazu per SSH auf `bordo.dbai.tuwien.ac.at` verbinden und ihn mittels `psql` nutzen. Die Zugangsdaten erhalten Sie von uns in einem E-Mail. Beachten Sie die Erklärungen bei den einzelnen Aufgaben.

Wichtig! Stellen Sie sicher, dass die von Ihnen abgegebenen SQL Befehle auf dem Server (`bordo.dbai.tuwien.ac.at`) ausgeführt werden können. D.h. dass es sich um (syntaktisch) gütlige SQL Befehle handelt. Sollte dies nicht der Fall sein (sollte es z.B. Syntaxfehler geben beim Versuch die Files auszuführen), dann bekommen Sie auch **keine Punkte** für die entsprechende Datei.

Das Übungsblatt enthält 6 Aufgaben, auf welche Sie insgesamt 15 Punkte erhalten können.

Deadlines

bis 07.01.	**12:00 Uhr**	Upload der Abgabe über TUWEL
bis 07.01.	**12:00 Uhr**	Anmeldung zu einem Kontrollgespräch über TUWEL

Kontrollgespräch

Im Rahmen des Kontrollgespräches wird nicht nur die Korrektheit, sondern vor allem das Verständnis der Konzepte überprüft. Sie müssen daher bei Ihrem Kontrollgespräch in der Lage sein, nicht nur Ihre Beispiele zu erklären, sondern ebenfalls zeigen, dass Sie die in der Vorlesung behandelte Theorie zu diesen Beispielen ausreichend verstanden haben.

Die Bewertung Ihres Übungsblattes basiert zum Überwiegenden Teil auf Ihrer Leistung beim Kontrollgespräch! Es daher ist im Extremfall durchaus möglich, dass eine korrekte Abgabe mit 0 Punkten bewertet wird. Insbesondere werden nicht selbstständig gelöste Abgaben immer mit 0 Punkten bewertet!

Hinweis: Noch einmal der Hinweis, dass Ihre Lösung beim Kontrollgespräch auf dem Server ausführbar sein muss. Testen Sie Ihre Lösung daher bevor Sie sie abgeben auf `bordo.dbai.tuwien.ac.at`. Es spielt keine Rolle, ob Sie beim Kontrollgespräch auftretende Syntaxfehler sofort beheben und erklären können – sollte Ihre Abgabe nicht lauffähig sein wird es **keine Punkte für die entsprechende Datei** geben.

Erscheinen Sie bitte pünktlich zum Kontrollgespräch. Bringen Sie bitte Ihren Studentenausweis zum Kontrollgespräch mit. Ein Kontrollgespräch ohne Ausweis ist nicht möglich.

Tutorensprechstunden (freiwillig)

Rund eine Woche vor der Abgabedeadline bieten die TutorInnen Sprechstunden an. Falls Sie Probleme mit oder Fragen zum Stoff des Übungsblattes haben, es Verständnisprobleme mit

den Beispielen oder technische Fragen gibt, kommen Sie bitte einfach vorbei. Die TutorInnen beantworten Ihnen gerne Ihre Fragen zum Stoff, oder helfen Ihnen bei Problemen weiter.

Ziel der Sprechstunden ist es, Ihnen beim **Verständnis des Stoffs** zu helfen, nicht, das Übungsblatt für Sie zu rechnen, oder die eigenen Lösungen vorab korrigiert zu bekommen.

Die Teilnahme ist vollkommen freiwillig — Termine und Orte der Tutorensprechstunden finden Sie in TUWEL.

Aufgaben: Datenbank mit PostgreSQL

Die folgenden Aufgaben basieren auf der Datenbank die Sie bereits aus Übungsblatt 1 kennen. Wir geben hier nochmals das Relationenschema für Sie an. Sie finden das entsprechende EER-Diagramm in Abbildung 1 auf der letzten Seite der Angabe.

Kaffee	name,aroma,saeure,koerper
Lungo	name: *Kaffee.name*
Ristretto	name: *Kaffee.name*, roetemp
Espresso	name: *Kaffee.name*, druck
Produkt	modelnr, preis, produziertVon: *Hersteller.marke*
Kaffeemaschine	modelnr: *Produkt.modelnr*, name, druck, anzkaps
BasiertAuf	alt: *Kaffeemaschine.modelnr*, neu: *Kaffeemaschine.modelnr*
Unterstuezt	maschine: *Kaffeemaschine.modelnr*, typID: *Kaffeekapseltyp.ktid*, typGroesse: *Kaffeekapseltyp.groesse*
KannZubereiten	maschine: *Kaffeemaschine.modelnr*, kaffee: *Kaffee.name*
Kaffeekapseltyp	ktid, groesse, volumen
KompatibelMit	vonID: *Kaffeekapseltyp.ktid*, vonGroesse: *Kaffeekapseltyp.groesse*, zuID: *Kaffeekapseltyp.ktid*, zuGroesse: *Kaffeekapseltyp.groesse*
Kaffeekapsel	modelnr: *Produkt.modelnr*, material, typID: *Kaffeekapseltyp.ktid*, typGroesse: *Kaffeekapseltyp.groesse*, enthaelt: *Kaffee:name*
Hersteller	marke, stl
Review	hersteller: *Hersteller.marke*, id, txt
Lizenz	maschine:Kaffeemaschine.modelnr, typID: *Kaffeekapseltyp.ktid*, typGroesse: *Kaffeekapseltyp.groesse*, hersteller: *Hersteller.marke*, gebuehr
Haendler	name, webseite
Verkauft	haendler: *Haendler.name*, produkt: *Produkt.modelnr*
Mengenrabatt	angebotenVon: *Haendler.name*, label, rabatt
BestehtAus	rabattVon: *Mengenrabatt.angebotenVon*, rabattLabel: *Mengenrabatt.Label*, rabattRabatt: *Mengenrabatt.rabatt*, produkt: *Produkt.modelnr*, menge
GutscheinCode	rabattVon: *Mengenrabatt.angebotenVon*, rabattLabel: *Mengenrabatt.Label*, rabattRabatt: *Mengenrabatt.rabatt*, code

Aufgabe 1 (Erstellen von Sequenzen & Tabellen) [2 Punkte]

Erstellen Sie eine Datei `create.sql`, in welcher die nötigen CREATE-Befehle gespeichert werden, um das gegebene Relationenschema mittels SQL zu realisieren.

Beachten Sie dabei folgendes:

(a) Ändern Sie das Relationenschema, um auch folgende Sachverhalte korrekt darzustellen:

- Jeder Hersteller hat ein Hauptprodukt.
- Jeder Kaffee verweist auf eine Kaffeekapsel, die als "Referenzkapsel" für diesen dient.

Diese Änderungen können Sie direkt in den CREATE Statements abbilden.

(b) Realisieren Sie die fortlaufende Nummerierung des Attributs `ModelNr` der Relation `Produkt` mit Hilfe einer Sequence. Die Sequence soll bei 1 beginnen und in Einerschritten erhöht werden.

(c) Realisieren Sie die fortlaufende Nummerierung des Primärschlüssel-Attributs `KTId` in der Tabelle `Kaffeekapseltyp` mit Hilfe einer Sequence. Die Sequence soll bei 7000 beginnen und in Dreierschritten erhöht werden.

(d) Das Attribut `Aroma` in der Tabelle `Kaffee` kann nur die Werte `'Mild'`, `'Normal'` und `'Stark'` annehmen. Erstellen Sie dazu einen ENUM Typ.

(e) Repräsentieren Sie den Druck von Kaffeemaschinen in Bar als NUMERIC mit zwei Nachkommastellen.

(f) Sollten zwischen zwei Tabellen zyklische FOREIGN KEY Beziehungen existieren, so achten Sie darauf, dass eine Überprüfung dieser FOREIGN KEYs erst zum Zeitpunkt eines COMMITs stattfindet.

(g) Verwenden Sie keine Umlaute für Bezeichnungen von Relationen, Attributen, etc.

(h) Stellen Sie die folgenden Sachverhalte durch geeignete Bedingungen sicher:

- Ein Espresso muss einen Druck von mindestens 7 Bar haben.
- Ein GutscheinCode muss einen Code mit einer Länge von mindestens 10 Zeichen haben, und zumindest einen Großbuchstaben enthalten.
- AnzKaps jeder Kaffeemasschine muss zwischen (inklusive) 1 und 20 liegen.
- Ein Kaffeekapseltyp kann von einem Hersteller nicht mehrmals lizenziert werden.

(i) Treffen Sie für alle fehlenden Angaben (z.B.: Typen von Attributen) plausible Annahmen. Vermeiden Sie NULL Werte in den Tabellen, d.h. alle Attribute müssen angegeben werden.

(j) Sie müssen sich nicht um die min/max Notationen aus dem EER-Diagramm in Abbildung 1 sorgen.

Aufgabe 2 (Einfügen von Testdaten) [1 Punkt]

Erstellen Sie eine weitere Datei `insert.sql`, welche die INSERT-Befehle für die Testdaten der in Punkt 2 erstellten Tabellen enthält. Jede Tabelle soll zumindest drei Zeilen enthalten. Sie dürfen die Wahl der Namen, Bezeichnungen etc. so einfach wie möglich gestalten, d.h. Sie müssen nicht "real existierende" Kaffeemaschinen, Hersteller, Händler, etc. wählen. Stattdessen können Sie auch einfach "Kaffeemaschine 1", "Kaffeemaschine 2", "Hersteller I", "Hersteller II" etc. verwenden. Sie können zum Anlegen geeigneter Relationen die in Aufgaben 4 angelegten Trigger und Procedures verwenden.

Aufgabe 3 (SQL Abfragen) [4 Punkte]

Erstellen Sie eine Datei `queries.sql`, welche den Code für folgende Views enthält.

(a) Erstellen Sie eine View `MaxVolumen` welche das maximale unterstützte Volumen einer Kaffeemaschine ermittelt. Das maximale unterstützte Volumen einer Kaffeemaschine ist das Maximum über alle Kaffeekapseltypen, die diese Kaffeemaschine unterstützt.

(b) Erstellen sie eine View `AllBasiertAuf` welche die BasiertAuf Relation so erweitert, dass wenn A auf B basiert und B auf C basiert, dann basiert auch A auf C. Beachten Sie, dass diese Definition beliebig lange Verkettungen von Kaffeemaschinen miteinbezieht. (Sei etwa im obigen Beispiel zusätzlich X basierend auf A, dann basiert X auch auch auf B und C.)

(c) Erstellen Sie eine View `KompatibelDistanz`, welche auf der Basis des Kapseltyps mit `ktid` 7000 und `groesse` 2 alle Kaffeekapseln ausgibt die kompatibel zu dieser sind, und über wie viele Schritte in der Relation sie kompatibel sind. Als Beispiel: Wenn Kapsel A kompatibel mit Kapsel B ist, und Kapsel B kompatibel mit Kapsel C, dann ist A über 2 Schritte kompatibel mit Kapsel C.

Hinweis: *Sie benötigen für die letzten beiden Teilaufgaben jeweils eine rekursive Abfrage. Achten Sie insbesondere darauf, dass ihre Abfragen mit zyklischen Beziehungen umgehen können, und trotzdem terminieren. Darüber hinaus ist auch zu empfehlen, dass Sie entsprechende Einträge in Ihrer* `insert.sql` *Datei anlegen um diese Abfragen sinnvoll auswerten zu können.*

Aufgabe 4 (Erstellen und Testen von Trigger) [6 Punkte]

Erstellen Sie eine Datei `plpgsql.sql`, welche den Code für die folgenden Trigger und Funktionen enthält.

(a) Erstellen Sie einen Trigger, der beim Anlegen einer `KannZubereiten` Beziehung sicherstellt, dass falls es sich beim Kaffee um einen Espresso handelt, der `Druck` der Kaffeemaschine nicht niedriger als der `Druck` des Espresso ist. Falls der `Druck` des Espresso diese Bedingung verletzt, soll die Beziehung nicht angelegt werden.

(b) Erstellen Sie einen Trigger, der folgendes Verhalten bei einer Änderung in der Relation `Lizenz` implementiert:

- Wenn `gebuehr` für ein Tupel erhöht wurde, dann soll das Attribut `Preis` der entsprechenden Kaffeemaschine um die Differenz des neuen Wert vom bisherigen Wert von `gebuehr` erhöht werden. Weiters soll in diesem Fall per `RAISE NOTICE` der neue Preis als Teil einer Nachricht ausgegeben werden.
- Wenn `gebuehr` per `UPDATE` auf den selben Wert gesetzt wurde der bereits gespeichert war, dann soll dazu mittels `RAISE` eine Warnung ausgegeben werden.

  Hinweis: Mehr Informationen zur Ausgabe von Meldungen mittels `RAISE` finden Sie in der Online Dokumentation[1].

(c) Erstellen Sie einen Trigger, der beim hinzufügen eines Kaffees K zu der Relation `KannZubereiten` für Kaffeemaschine M folgendes Verhalten implementiert:

- Wenn andere Kaffeemaschinen auf M basieren, dann sollen auch entsprechende Tupel für jene Kaffeemaschinen und K in die Relation `KannZubereiten` hinzugefügt werden.
- Wenn K bereits in der `KannZubereiten` Relation für diese Nachfolge-Kaffeemaschinen vorkommt, dann soll der Kaffee **nicht** nochmal hinzugefügt werden.

Bedenken Sie, dass Sie sich nicht um die rekursiven Abhängigkeiten zwischen den Kaffeemaschinen kümmern müssen, sondern dass Trigger wieder Trigger ausführen.

(d) Erstellen Sie eine Procedure `CreateManufacturer` die automatisch einen neuen Hersteller anlegt. Die Procedure hat drei Parameter: Eine Zahl welche die Anzahl an `Produkten` für den zu erstellenden Hersteller angibt, ein Land und die `Marke` des neuen Herstellers. Es wird davon ausgegangen, dass für diesen noch keine Lizenz-Vereinbarungen bestehen oder Reviews geschrieben wurden.

Beachten Sie folgendes:

- Der neue Hersteller muss mindestens 3 Produkte haben. Falls der Parameter für die Anzahl an Produkten kleiner als 3 ist, dann geben Sie eine entsprechende Fehlermeldung aus.
- Für das Land in dem der Hersteller steuerpflichtig ist `StL`, soll der entsprechende Parameter verwendet werden
- Das erste Produkt muss eine Kaffeemaschine sein. Als Name der Kaffeemaschine geben Sie "Coffee Machine of $Manufacturer" an, wobei hier der entsprechende Name des Herstellers aus dem Parameter zu wählen ist. Wählen Sie alle weiteren Attribute nach belieben. Alle weiteren Produkte sollen Kaffeekapseln sein.
- Legen Sie drei Kaffeesorten an: "Ristretto of $Manufacturer"', "Espresso of $Manufacturer" und "Lungo of $Manufacturer" (wählen sie alle weiteren Attribute der Kaffees nach belieben).
- Legen Sie nun so viele Kaffeekapseln an, wie noch weitere Produkte zu erstellen sind (abzüglich der Kaffeemaschine). Als Material für diese können Sie ein beliebigen String wählen. Es soll in ihrer Prozedur ein Zähler vorhanden sein, der angibt wie viele Kapseln bisher erstellt wurden.

  Je nach der Teilbarkeit des gegenwärtigen Stand des Zählers soll diese Kapsel den oben hinzugefügten Ristretto, Lungo oder Espresso enthalten. (Als Beispiel: 0 für eine Kapsel mit Ristretto, 1 für eine mit Lungo, 2 für eine mit Espresso, 3 wieder

[1] `https://www.postgresql.org/docs/11/static/plpgsql-errors-and-messages.html`

für eine mit Ristretto, und so weiter). Wir empfehlen, dass Sie dafür den Modulo Operator verwenden.

Nota bene: Für alle von Ihnen angelegten Entitäten (Produkte und Kaffees) gilt, dass ihre Funktion mit einer Fehlermeldung abbrechen soll, falls bereits Einträge in der Datenbank mit demselben Namen existieren. Im Fall eines Abbruchs soll die Procedure alle bisherigen Änderungen rückgängig machen.

Aufgabe 5 (Löschen der angelegten Objekte) [1 Punkt]

Erstellen Sie eine Datei `drop.sql`, welche die nötigen DROP-Befehle enthält, um alle erzeugten Datenbankobjekte wieder zu löschen. Das Schlüsselwort CASCADE darf dabei NICHT verwendet werden.

Aufgabe 6 (Testen der Datenbank und erstellen des Abgabearchivs) [1 Punkt]

(a) Erstellen Sie eine Datei `test.sql`. Dazu überlegen Sie sich eine sinnvolle Testabdeckung für die in Aufgabe 1 gegebenen Bedingungen, für die in Aufgabe 3 erstellen Views und für die PL/pgSQL-Programmteile in Aufgabe 4. Es sollen möglichst alle Fälle, auch positive, abgedeckt werden, z.B. Hinzufügen eines Espresso zur `KannZubereiten` Relation einer Kaffeemaschine mit erlaubtem und zu hohem Druck.

(b) Stellen Sie eine Listing-Datei mit dem Namen `listing.txt` bereit, die Sie bei der Ausführung der SQL-Dateien erzeugt haben. Diese Erstellen Sie am Besten auf unserem Übungsserver `bordo.dbai.tuwien.ac.at`. Dort starten Sie `psql`. Mittels "`\o listing.txt`" lässt sich die Ausgabe in die Datei `listing.txt` umleiten. Dann führen Sie die Dateien (sofern vorhanden) in dieser Reihenfolge mittels "`\i xxx.sql`" aus:

1. `create.sql`
2. `insert.sql`
3. `queries.sql`
4. `plpgsql.sql`
5. `test.sql`
6. `drop.sql`

Erstellen Sie aus diesen und der `listing.txt` Datei ein ZIP-Archiv `blatt4.zip` und laden dieses in TUWEL hoch.

EER-Diagramm

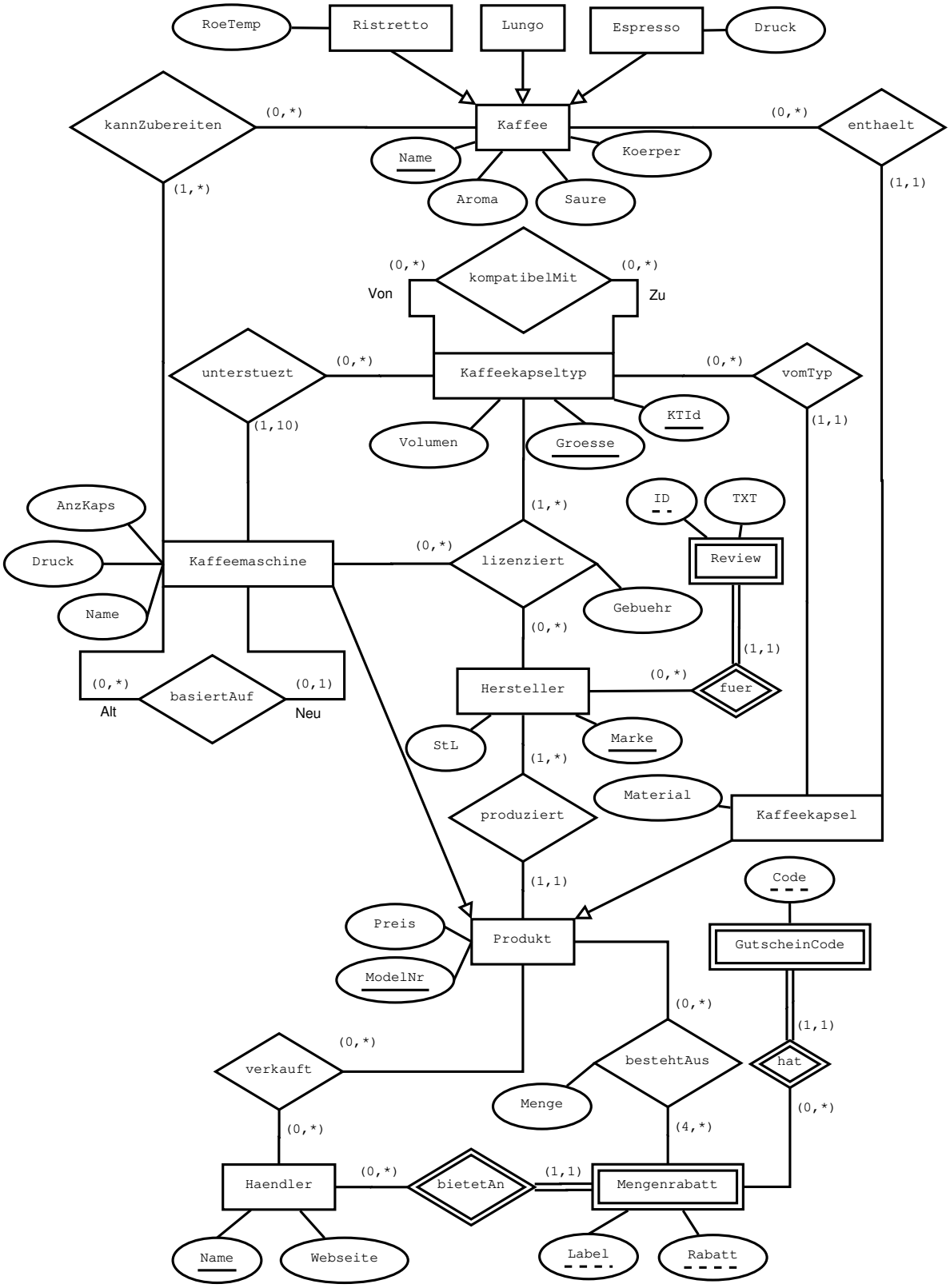

Abbildung 1: EER-Diagramm zu diesem Übungszettel