4. Übungsblatt (WS 2022)

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 finden Sie auf TUWEL unter `https://tuwel.tuwien.ac.at/mod/assign/view.php?id=1654195`. 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 10.01.	**12:00 Uhr**	Upload der Abgabe über TUWEL
bis 10.01.	**12:00 Uhr**	Anmeldung zu einem Kontrollgespräch über TUWEL

Kontrollgespräch

Dieses Semester werden die Kontrollgespräche über Zoom durchgeführt. Zu Anfang Ihres Gesprächs müssen Sie Ihre Kamera einschalten und Ihren Studierendenausweis dem Prüfer zeigen, um Ihre Identität zu bestätigen. Anschließend findet das eigentliche Kontrollgespräch statt, in welchem nicht nur die Korrektheit Ihrer Lösung, sondern vor allem das Verständnis der Konzepte überprüft wird. 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! Daher ist es 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 zu Ihrem Zoom Kontrollgespräch und halten Sie bitte Ihren Studierendenausweis zum Zoom Kontrollgespräch bereit. Ein Kontrollgespräch ohne Ausweis ist nicht möglich.

Aufgaben: Datenbank mit PostgreSQL

Die folgenden Aufgaben basieren auf der Datenbank die Sie bereits aus Übungsblatt 1 (aus Aufgabe 3) 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.

```
Grundstueck (Adresse, Groesse)
grenzt_an   (Grundstuecks_Adresse: Grundstueck.Adresse,
             Nachbar_Adresse: Grundstueck.Adresse)
enthaelt    (Grundstuecks_Adresse: Grundstueck.Adresse,
             Adresse_Des_Teilgrundstuecks: Grundstueck.Adresse)
Heizanlage  (Adresse: Grundstueck.Adresse, HId, kW)
Waermepumpe (Adresse: Heizanlage.Adresse, HId: Heizanlage.Hid)
Gasheizung  (Adresse: Heizanlage.Adresse, HId: Heizanlage.Hid)
Gewaechshaus(Adresse: Grundstueck.Adresse, GId, Größe)
beheizt     (Heizanlage_Adresse: Heizanlage.Adresse, HId: Heizanlage.HId,
             Gewaechshaus_Adresse: Gewaechshaus.Adresse, GId: Gewaechshaus.GId)
Kontrolle   (Adresse:Gewaechshaus.Adresse, GId: Gewaechshaus.GId,
             SVNR: Angestellter.SVNR, Datum, Beobachtung)
Angestellter(SVNR, Name, Adresse)
Pflanzen    (Adresse: Gewaechshaus.Adresse, GId: Gewaechshaus.GId, Name, Anzahl)
Palmen      (Adresse: Pflanzen.Adresse, GId: Pflanzen.GId,
             Name: Pflanzen.Name, Winterhart)
Orchideen   (Adresse: Pflanzen.Adresse, GId: Pflanzen.GId,
             Name: Pflanzen.Name, Farbe)
Sukkulenten (Adresse: Pflanzen.Adresse, GId: Pflanzen.GId,
             Name: Pflanzen.Name)
Duenger     (Art, Lagerbestand)
duengt      (SVNR: Angestellter.SVNR, Art: Duenger.Art, Adresse: Pflanzen.Adresse,
             GId: Pflanzen.GId, Name: Pflanzen.Name)
```

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:

- Jedes Grundstück hat eine Primärheizanlage.
- Jede Kontrolle erfasst einen hauptverantwortlichen Angestellten.

Diese Änderungen können Sie direkt in den CREATE Statements abbilden.

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

(c) Realisieren Sie die fortlaufende Nummerierung des Attributs `HId` in der Tabelle `Heizung` mit Hilfe einer Sequence. Die Sequence soll bei 1.000 beginnen und in Zehnerschritten erhöht werden.

(d) Das Attribut `Farbe` in der Tabelle `Orchideen` kann nur die Werte `'Weiß'`, `'Rosa'` und `'Gelb'` annehmen. Erstellen Sie dazu einen ENUM Typ.

(e) Repräsentieren Sie die Größe eines Gewächshauses und eines Grundstücks als NUMERIC mit zwei Nachkommastellen (dabei werden Größen in Quadratmeter m^2 gemessen).

(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:

- Die Größe eines Gewächshauses und eines Grundstücks muss größer sein als 0.
- Die Bezeichnung der Art eines Düngers besteht aus genau 10 Symbolen, die mit drei Buchstaben beginnt, gefolgt von einem Bindestrich (“-”) und 6 alphanumerischen Zeichen.
- Das Datum einer Kontrolle ist größer als der 1. Februar 1990.
- Eine Pflanze kann nicht mit verschiedenen Düngerarten gedüngt werden (jede Pflanze wird mit maximal einer Düngerart gedüngt).

(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" Pflanzennamen, Adressen, etc. wählen. Stattdessen können Sie auch einfach "Pflanze 1", "Pflanze 2", "Adresse 1", "Adresse 2" 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 folgenden Views enthält.

(a) Erstellen Sie eine View `NumberOfBigGreenhouses`, welche die Anzahl von großen Gewächshäusern pro Grundstück ausgibt. Die Anzahl der großen Gewächshäuser eines Grundstücks ist definiert als die Anzahl aller Gewächshäuser mit einer Größe über $30m^2$ die auf dem Grundstück liegen.

(b) Erstellen Sie eine View `AllBlockNeighbours`, welches die Relation `grenzt_an` in folgender Art erweitert: Wenn laut `AllBlockNeighbours` ein Grundstück A mit einem Grundstück B benachbart ist und B mit einem weiteren Grundstück C benachbart ist, dann soll im View auch dargestellt werden dass A mit C benachbart ist. Beachten Sie, dass diese Definition beliebig lange Verkettungen von benachbarten Grundstücken miteinbezieht. Sei etwa im obigen Beispiel zusätzlich X ein Nachbar von A, dann ist X auch ein Nachbar von B und C.

(c) Erstellen Sie eine View `NeighbourhoodSize`, welche all jene Tupel (A, B, G_{AB}, S) enthält, sodass ein Grundstück A über S "Schritte" mit dem Grundstück B benachbart ist und sodass G_{AB} die Summe der Größen der Grundstücke zwischen A und B darstellt.

Als Beispiel: Angenommen X, Y und Z sind Grundstücke und X ist benachbart mit Y und Y ist benachbart mit Z und die Größen von X, Y und Z sind jeweils $G_X = 100m^2$, $G_Y = 150m^2$ und $G_Z = 120m^2$, dann ist beispielsweise X mit Y über einen "Schritt" benachbart mit der Nachbarschaftsgröße $G_{XY} = G_X + G_Y = 100m^2 + 150m^2 = 250m^2$ und dann ist X auch mit Z über zwei "Schritte" benachbart mit der Nachbarschaftsgröße $G_{XZ} = G_X + G_Y + G_Z = 370m^2$. Beachten Sie, dass in diesem Beispiel Y auch mit Z über einen "Schritt" benachbart ist mit der Nachbarschaftsgröße $G_{YZ} = G_Y + G_Z = 270m^2$.

Beachten Sie weiters, dass Sie die Summe der Distanzen im SELECT Ausdruck in den Typen den Sie für das `Größe` Attribut verwendet haben umwandeln müssen. Ein kurzes Tutorial zum Thema Typumwandlung in PostgreSQL finden Sie unter `https://www.postgresqltutorial.com/postgresql-cast/`.

Hinweis: *Sie benötigen für die letzten beiden Teilaufgaben jeweils eine rekursive Abfrage. Wir 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 eines `Gewaechshauses` sicherstellt, dass die `Groesse` des `Gewaechshauses` kleiner gleich der `Groesse` des zugehörigen `Grundstuecks` ist. Das heißt:

- Falls die `Groesse` des hinzuzufügenden `Gewaechshauses` gleich 0 ist, so soll das neue `Gewaechshaus` mit der `Groesse` des zugehörigen `Grundstuecks` der `Gewaechshaus` Tabelle hinzugefügt werden.
- Falls die `Groesse` des hinzuzufügenden `Gewaechshauses` größer der `Groesse` des zugehörigen `Grundstuecks` ist, so soll das `Gewaechshaus` nicht angelegt werden.

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

- Wenn die `Anzahl` für ein Tupel vergößert/verkleinert wurde, dann soll das Attribut `Lagerbestand` der Dünger, welche zum Düngen dieser Pflanze verwendet werden, um die Differenz des neuen Werts vom bisherigen Wert der `Anzahl` vergößert/verkleinert werden. Weiters, soll in diesem Fall per `RAISE NOTICE` der neue `Lagerbestand` als Teil einer Nachricht ausgegeben werden.
- Wenn die `Anzahl` einer Pflanze 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 bei einer Änderung der `Groesse` eines Grundstücks folgendes Verhalten implementiert:

- Wenn die `Groesse` eines Grundstücks G_c um einen bestimmten Betrag vergößert/verkleinert wird, dann sollen auch die `Groessen` von allen Grundstücken G_p die G_c enthalten, um den selben Betrag vergößert/verkleinert werden.

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

(d) Erstellen Sie eine Procedure `CreateHeatingSystems` die automatisch neue Heizanlagen anlegt. Die Procedure hat drei Parameter: Eine Zahl welche die `kW` der zu erstellenden `Heizanlagen` angibt, die Anzahl der zu erstellenden `Heizanlagen` und die `Groesse` des Grundstücks auf welchem die `Heizanlagen` liegen sollen.

Beachten Sie folgendes:

- Es muss mindestens eine Heizanlage angelegt werden. Falls der Parameter für die Anzahl an Heizanlagen kleiner als 1 ist, dann geben Sie eine entsprechende Fehlermeldung aus.

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

- Das durch die Parameter festgelegte Grundstück soll angelegt werden. Die `Groesse` des neuen Grundstücks soll dem entsprechenden Eingabeparameter entnommen werden.
- Weiters, soll die angegebene Anzahl an Heizanlagen wie folgt erzeugt werden:
  - Es soll in Ihrer Prozedur ein Zähler vorhanden sein, der angibt wie viele Heizanlagen bisher erstellt wurden. Je nach der Teilbarkeit des gegenwärtigen Stands des Zählers durch 2 soll eine Wärmepumpe oder eine Gasheizung erstellt werden.

    Als Beispiel soll bei Zählerstand 0 ein Wärmepumpe, bei 1 eine Gasheizung, bei 2 wieder eine Wärmepumpe und so weiter erstellt werden. Wir empfehlen, dass Sie dafür den Modulo Operator verwenden. Vergessen Sie nicht die entsprechenden Einträge in der `Waermepumpe` und `Gasheizung` Tabelle zu generieren. Sie können die kW der Heizanlagen beliebig wählen.
  - Sie können die `Adresse` der zu erstellenden Grundstücke frei wählen. Stellen Sie aber sicher, dass in jedem Schritt eine verfügbare `Adresse` verwendet wird (damit keine Primärschlüsselbedingung verletzt wird).
  - Wählen Sie eine der erstellten Heizanlagen (beispielweise die erste) als the Primärheizanlage des angelegten Grundstücks.
  - Verwenden Sie die in Aufgabe 1c erstellte Sequenz um für die zu erstellenden Heizanlagen eine `HId` zu generieren.
  - Zum Abschluss sollen Einträge erzeugt werden, sodass die erstellten Heizanlagen auf dem initial angelegten Grundstück liegen.

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 erstellten Views und für die PL/pgSQL-Programmteile in Aufgabe 4. Es sollen möglichst alle Fälle, positive als auch negative, abgedeckt werden, z.B. Hinzufügen eines Eintrags in die Tabelle `Kontrolle` mit einem Datum vor und nach dem 1. Februar 1990.

(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) `plpgsql.sql`

(3) `insert.sql`

(4) `queries.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

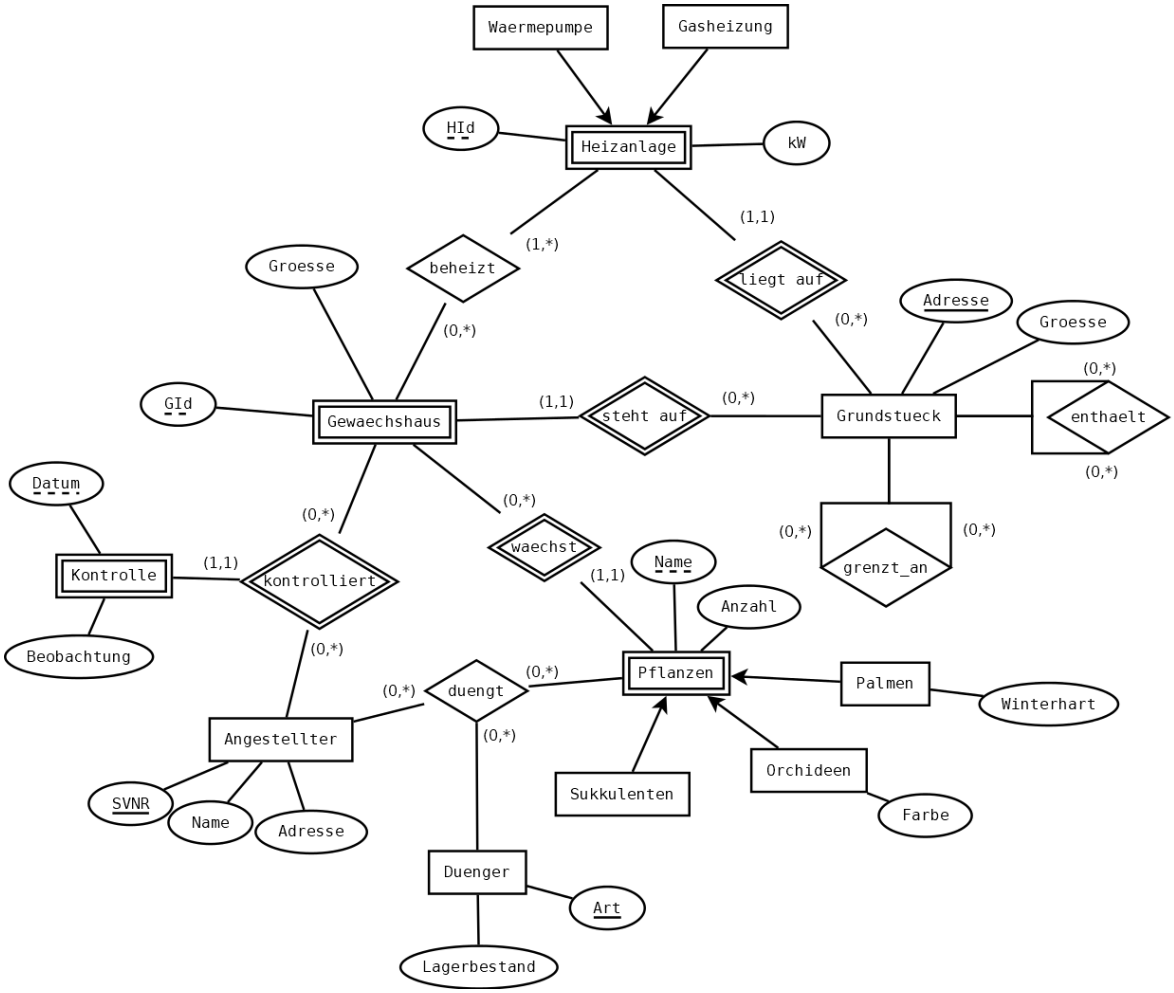

Abbildung 1: EER-Diagramm zu diesem Übungszettel