4. Übungsblatt (WS 2020)

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=994171`. 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 11.01.	**12:00 Uhr**	Upload der Abgabe über TUWEL
bis 11.01.	**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 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.

Schriftstück	SID, Titel, Seiten
Sachbuch	SID: *Schriftstück.SID*
Roman	SID: *Schriftstück.SID*, Genre
Lyrik	SID: *Schriftstück.SID*
über	Sachbuch: *Sachbuch.SID*, Roman: *Roman.SID*
schrieb	Autorin: *Autorin.AID*, Schriftstück: *Schriftstück.SID*, StartDatum
Autorin	AID, Name, GebDat
Edition	Schriftstück: *Schriftstück.SID*, EDNR, Jahr, Verlagsname: *Verlag.Name*
Auflage	Schriftstück: *Edition.Schriftstück*, EDNR: *Edition.EDNR*, ANR, Druckerei
Sonderedition	Schriftstück: *Edition.Schriftstück*, EDNR: *Edition.EDNR*, Anlass
Verlag	Name, Budget
Abteilung	Verlagsname: *Verlag.Name*, Bereich, Stadt
wirbt	Verlagsname: *Verlag.Name*, Kanal: *MarketingKanal.KName*, Zielgruppe: *Zielgruppe.Bez*, Datum
MarketingKanal	KName, Kosten
Social	Kanal: *MarketingKanal.KName*, Platform
Zeitung	Kanal: *MarketingKanal.KName*, Schriftstück: *Schriftstück.SID*, Auflage
Zielgruppe	Bez, Agruppe
nutzt	Zielgruppe: *Zielgruppe.Bez*, SocialKanal: *Social.KName*
interessiert	Zielgruppe: *Zielgruppe.Bez*, Schriftstück: *Schriftstück.SID*
liebt	Zielgruppe: *Zielgruppe.Bez*, Autorin: *Autorin.AID*
hasst	Zielgruppe: *Zielgruppe.Bez*, Autorin: *Autorin.AID*
aka	alias: *Autorin.AID*, aliasVon: *Autorin.AID*
basiertAuf	neu: *Roman.SID*, alt: *Roman.SID*

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 Verlag hat eine Hauptabteilung.
- Jede Edition hat eine Erstauflage.

Diese Änderungen können Sie direkt in den CREATE Statements abbilden.

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

(c) Realisieren Sie die fortlaufende Nummerierung des Primärschlüssel-Attributs `AID` in der Tabelle `Autorin` mit Hilfe einer Sequence. Die Sequence soll bei 10.000 beginnen und in Siebenerschritten erhöht werden.

(d) Das Attribut `Agruppe` in der Tabelle `Zielgruppe` kann nur die Werte `'young'`, `'middle-aged'` und `'old'` annehmen. Erstellen Sie dazu einen ENUM Typ.

(e) Repräsentieren Sie das Budget von Verlägern in Euro 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:

- Das Jahr einer Edition muss zumindest 1400 sein (da der Buchdruck erst in der Mitte des 15. Jahrhunderts entwickelt wurde).
- Ein Bereichscode einer Abteilung (repräsentiert durch das "Bereich" Attribut) besteht aus genau 5 Symbolen, die mit zwei alphanumerischen Zeichen beginnt, gefolgt von drei Zahlen.
- Das Budget jedes Verlages muss zwischen 10€ und 1.000.000€ liegen (inklusive 10€ und 1.000.000€).
- Ein Verlag kann eine Zielgruppe nicht mehrmals am selben Datum bewerben.

(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" Autorinnen, Schriftstücke, Verläge, etc. wählen. Stattdessen können Sie auch einfach "Autor 1", "Autorin 2", "Buch 1", "Buch 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 folgende Views enthält.

(a) Erstellen Sie eine View `NumberOfLargeBooks`, welche die Anzahl der dicken Bücher pro Autorin ausgibt. Die Anzahl der dicken Bücher einer Autorin ist definiert als die Anzahl aller Bücher, die von der Autorin geschrieben wurden und mehr als 200 Seiten lang sind.

(b) Erstellen sie eine View `AllAliases`, welche die `aka` Relation so erweitert, dass wenn Autorin A ein Alias von Autorin B ist und B ein Alias von Autorin C ist, dann auch im View enthalten ist, dass A ein Alias von C ist. Beachten Sie, dass diese Definition beliebig lange Verkettungen von Autorinnen Aliassen miteinbezieht. (Sei etwa im obigen Beispiel zusätzlich X ein Alias von A, dann ist X auch ein Alias von B und C.)

(c) Erstellen Sie eine View `NovelDistance`, welche all jene Tupel (X, Z, G, S) enthält, sodass Roman X über S "Schritte" mit Roman Z verbunden ist über Romane des gleichen Genres G wie X und Z. Als Beispiel: Wenn Romane X, Y und Z das selbe Genre G haben und weiters X auf Y basiert und Y auf Z basiert, so basiert X über 2 Schritte des selben Genres auf Z. Jedoch, wenn im vorherigen Beispiel Y nicht das selbe Genre G wie X und Z hätte (sondern Y ein anderes Genre G_Y hätte mit $G_Y \neq G$) dann basiert X nicht auf Z über Romane des selben Genres.

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 `schrieb` Beziehung sicherstellt, dass das Startdatum (`Start`) der *schrieb* Beziehung nicht vor dem `Geburtsdatum` der Autorin liegt. Falls das `Startdatum` der schrieb Beziehung diese Bedingung verletzt, soll die Beziehung nicht angelegt werden.

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

- Wenn die `Kosten` für ein Tupel verändert wurden, dann soll das Attribut `Budget` der entsprechenden Verläge um die Differenz des neuen Werts vom bisherigen Wert der `Kosten` angepasst werden. Beispielsweise, wenn die `Kosten` für einen Marketingkanal steigen, so muss das Budget aller Verläge, die auf diesem Kanal Zielgruppen bewerben, entsprechend erhöht werden. Weiters soll in diesem Fall per `RAISE NOTICE` das neue Budget als Teil einer Nachricht ausgegeben werden.
- Wenn die `Kosten` eines Marketingkanals per `UPDATE` auf den selben Wert gesetzt wurden 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 einer neuen Zielgruppe Z zu der Relation `liebt` für Autorin A folgendes Verhalten implementiert:

- Wenn A ein Alias von anderen Autorinnen A' ist, dann sollen auch entsprechende Tupel in die Relation `liebt` hinzugefügt werden, sodass diese ausdrücken, dass A' von den selben Zielgruppen Z geliebt wird wie Alias A.
- Wenn Z bereits in der `liebt` Relation für diese Nachfolge-Alias-Autorin vorkommt, dann soll die Zielgruppe **nicht** nochmal hinzugefügt werden.

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

(d) Erstellen Sie eine Procedure `CreateAuthor` die automatisch eine neue Autorin anlegt. Die Procedure hat drei Parameter: Eine Zahl welche die Anzahl an `Schriftstücken` für die zu erstellende Autorin angibt, ihren `Namen` und ihr `Geburtsdatum`. Es wird davon ausgegangen, dass die neue Autorin noch von niemanden geliebt oder gehasst wird.

Beachten Sie folgendes:

- Die neue Autorin muss mindestens 3 Schriftstücke geschrieben haben. Falls der Parameter für die Anzahl an Schriftstücken kleiner als 3 ist, dann geben Sie eine entsprechende Fehlermeldung aus.
- Für den Namen und das Geburtsdatum der Autorin, sollen die entsprechenden Parameter verwendet werden
- Weiters muss ein Alias für die Autorin erstellt werden. Als Alias geben Sie die Anfangsbuchstaben der Namen der Autorin an (z.B. "Cleo Virginia" wird zu "C.V."). Sie dürfen davon ausgehen, dass Autoren genau einen Vor- und einen Nachnamen (keine Mittelnamen) haben. Alle weiteren Attribute des Aliases können beliebig gewählt werden. Vergessen Sie nicht die Alias Autorin mit der echten Autorin über die aka Beziehung in Verbindung zu setzen.
- Nun müssen so viele Schriftstücke angelegt werden, wie in den Parametern der Prozedur bestimmt wurde. Es soll in ihrer Prozedur ein Zähler vorhanden sein, der angibt wie viele Schriftstücke bisher erstellt wurden. Je nach der Teilbarkeit des gegenwärtigen Stands des Zählers durch 3 soll ein Roman, eine Lyrik oder ein Sachbuch erstellt

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

werden. Als Beispiel soll bei Zählerstand 0 ein Roman, bei 1 eine Lyrik, bei 2 ein Sachbuch, bei 3 wieder ein Roman und so weiter erstellt werden. Wir empfehlen, dass Sie dafür den Modulo Operator verwenden.

Wählen Sie für die Titel der Schriftstücke jeweils "Novel of $aka", "Poetry of $aka" und "Non-fiction of $aka", wobei $aka für das im vorigen Schritt erstellte Alias der Autorin steht (z.B. "C.V."). Vergessen Sie nicht zugehörig auch jeweils einen Eintrag in den Roman, Lyrik und Sachbuch Tabellen zu erstellen. Wählen Sie alle weiteren Attribute der Schriftstücke, Romane, et cetera nach belieben.

- Legen Sie zum Abschluss Tupel in der "schrieb" Relation an, die ausdrücken, dass die erstellte Autorin unter ihrem Alias alle in den vorherigen Schritten angelegten Schriftstücke geschrieben hat.

Nota bene: Für alle angelegten Autorinnen (alle Aliasse inkludierend) 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 Prozedur 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 Eintrags in die Relation `schrieb` mit einem Startdatum vor und nach dem Geburtsdatum der entsprechenden Autorin.

(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

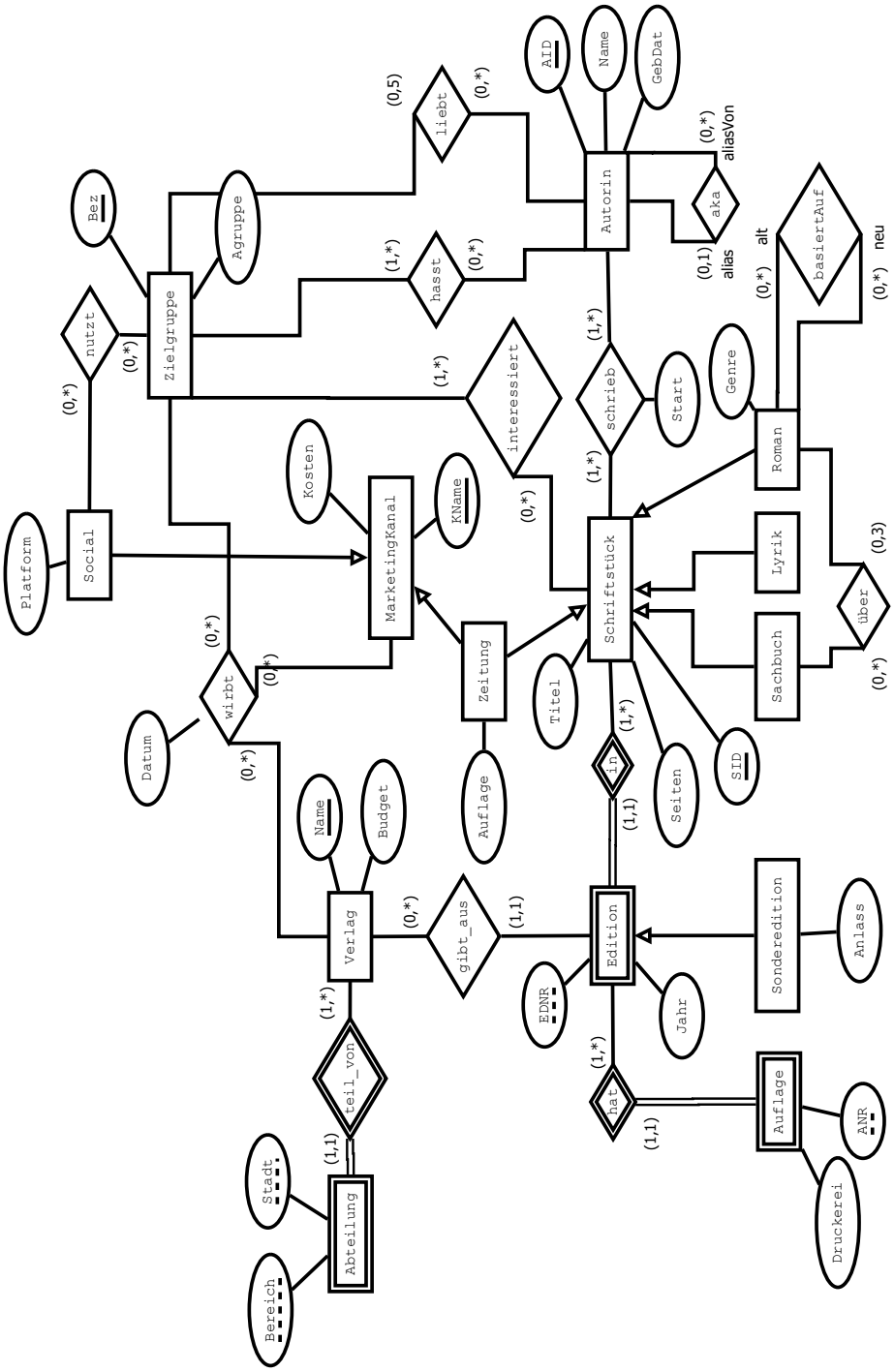

Abbildung 1: EER-Diagramm zu diesem Übungszettel