4. Übungsblatt (WS 2017)

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 9.4) 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.

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

Deadlines

bis 8.1.	**12:00 Uhr**	Upload der Abgabe über TUWEL
bis 8.1.	**12:00 Uhr**	Anmeldung zu einem Kontrollgespräch

Kontrollgespräch

Im Rahmen des Kontrollgespräches wird nicht nur die Korrektheit, sondern vor allem das Verständnis der Konzepte überprüft. Durch die Übung sollen sowohl Ihre praktische Problemlösungskompetenz als auch das theoretische Wissen über Datenbanksysteme gefördert werden. 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. Dies soll Ihnen die Vorbereitung für die Prüfung erleichtern und so können Sie Ihr Wissen während der Kontrollgespräche selbst testen und gegebenenfalls vertiefen.

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. Insbesonders werden nicht selbstständig gelöste Abgaben immer mit 0 Punkten bewertet!

Erscheinen Sie in Ihrem eigenen Interesse bitte pünktlich zum Kontrollgespräch, da andernfalls nicht garantiert werden kann, dass Ihre gesamte Lösung in der verbleibenden Zeit beurteilt werden kann.

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.

Weitere Fragen – TUWEL Forum

Sie können darüber hinaus das TUWEL Forum verwenden, sollten Sie inhaltliche oder organisatorische Fragen haben.

Aufgaben: Datenbank mit PostgreSQL

Die folgenden Aufgaben basieren auf der Datenbank die Sie bereits aus Aufgabenblatt 1 & 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.

```
Segler          (id, name, gebdatum, notfallkontakt)
Koch            (id:Segler.id, seehauben)
Ersthelfer      (id:Segler.id, ausbildung)
Skipper         (id:Segler.id, schein, seemeilen, lieblingstour:Tour.tid)
Crew            (leiter:Skipper.id, cid, crewname)
CrewMitglieder  (segler:Segler.id, cleiter:Crew.leiter, cid:Crew.cid)
CrewErsthelfer  (ersthelfer:Ersthelfer.id, cleiter:Crew.leiter,
                 cid:Crew.cid)
CrewKoch        (koch:Koch.id, cleiter:Crew.leiter, cid:Crew.cid)

Ankerplatz      (krz, koordinaten, name)
Tankstelle      (krz:Ankerplatz.krz,selfservice)
Fjord           (krz:Ankerplatz.krz,nacht,beschreibung)
Hafen           (krz:Ankerplatz.krz,gebuehr,duschen,einkauf)
Segelschiff     (hafen:Hafen.krz,sid,name,baujahr,kojen,kapazitaet)

Tour            (tid, bez, dauer, geplant_von:Skipper.id)
Teiltour        (tid:Tour.tid,teil_von:Tour.tid)
TourAnkerplatz  (tid:Tour.tid,krz:Ankerplatz.krz, tag)
Gechartert      (cleiter:Crew.leiter,cid:Crew.cid,hafen:Segelschiff.hafen,
                 sid:Segelschiff.sid,tid:Tour.tid,startdatum)
```

Aufgabe 1 (Erstellen von Sequenzen & Tabellen) *[6 Punkte]*

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

Beachten Sie dabei folgendes:

(a) Nehmen Sie Änderungen am Relationenschema vor, damit auch folgende Sachverhalte korrekt dargestellt werden:

- Im Rahmen einer Tour kann ein Ankerplatz auch mehrmals (an verschiedenen Tagen) angefahren werden.
- Ein Schiff kann von derselben Crew zur selben Tour mehrmals (mit verschiedenem Startdatum) gechartert werden.

Diese Änderungen können Sie direkt in den CREATE Statements abbilden.

(b) Realisieren Sie die fortlaufende Nummerierung der künstlichen Primärschlüssel-Attribute in der Tabelle `Segler` mit Hilfe einer Sequence. Die Sequence soll bei 1 beginnen und in Einerschritten erhöht werden.

(c) Realisieren Sie die fortlaufende Nummerierung der künstlichen Primärschlüssel-Attribute in der Tabelle `Tour` mit Hilfe einer Sequence. Die Sequence soll bei 1000 beginnen und in Zehnerschritten erhöht werden.

(d) Das Attribut `schein` in der Tabelle `Skipper` kann nur die Werte `'A'`, `'B'` und `'C'` annehmen. Erstellen Sie dazu einen ENUM Typ.

(e) Wählen Sie für Geldbeträge keine Gleitkommatypen sondern z. B. NUMERIC mit zwei Nachkommastellen.

(f) Der Primärschlüssel der Tabelle `Ankerplatz` ist ein Kürzel aus 3 Zeichen mit fixer Länge.

(g) 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.

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

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

- Die Anzahl der Kojen und die Kapazität von Segelschiffen sind immer positive ganze Zahlen (inkl. 0).
- Der Tag in der Tabelle `TourAnkerplatz` muss eine positive ganze Zahl sein (ohne 0).
- Eine Tour kann pro Tag nur einen Ankerplatz anfahren.
- Der/Die LeiterIn, ErsthelferInnen und KöchInnen einer Crew müssen auch MitgliederInnen einer Crew sein.

(j) 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.

Aufgabe 2 (Einfügen von Testdaten) *[1 Punkte]*

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" SeglerInnen, Ankerplätze, Schiffe, etc. wählen. Stattdessen können Sie auch einfach "SeglerIn 1", "SeglerIn 2", "Schiff 1", "Schiff 2" etc. verwenden. Sie können zum Anlegen der Crew und der Schiffe die in Aufgaben 4 und 5 angelegten Trigger und Procedures verwenden.

Aufgabe 3 (SQL Abfragen) *[2 Punkte]*

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

(a) Erstellen Sie eine View `TourG` welche die Gebühr einer Tour ermittelt. Die Gebühr einer Tour ist die Summe der Gebühren der Häfen, die im Rahmen dieser Tour angefahren werden. Die Teiltouren werden dabei nicht berücksichtigt.

(b) Erstellen Sie nun eine zweite View `TourGebuehr` welche die Teiltouren mitberücksichtigt. Die View `TourGebuehr` ermittelt somit die Summe aus der Gebühr der Tour selbst (mit Hilfe der View `TourG`) und allen Gebühren die in den Touren anfallen, die Teil dieser Tour sind (rekursiv). Als Beispiel: "Fjorde" und "Nordkap" sind Teil der Tour "Skandinavien", die noch zusätzlich den Hafen "Oslo" anfährt. Die Gebühr im Hafen "Oslo" beträgt 50, die Gebühr der Touren "Fjorde" und "Nordkap" je 30. Somit beträgt die Gesamtgebühr der Tour "Skandinavien" 110.

Aufgabe 4 (Erstellen und Testen von Trigger) *[4 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 Segelschiffs überprüft, ob die `sid` gesetzt wurde. Falls nicht, erhöhen Sie die höchste `sid` des Hafens (d.h. die höchste `sid` aller in diesem Hafen beheimateten Schiffe) um 1 und setzen Sie damit die `sid` des Segelschiffs. Dadurch erhält das 1. Segelschiff im Hafen "ABC" die `sid` 1, das 2. die `sid` 2, etc.

(b) Erstellen Sie einen Trigger, der beim Chartern eines Schiffs folgendes überprüft:

- Das Schiff hat genügend Kapazität für die Crew.
- Das Schiff wurde noch nicht für diesen Zeitraum gechartert. Den Zeitraum ermitteln Sie aus dem Startdatum und der Dauer der Tour. Verwenden Sie dabei den SQL Operator `OVERLAPS`. *Hinweis:* Sie können in PostgreSQL zu einem `DATE` Datentyp jeden `INTEGER` X addieren. Dadurch erhalten Sie ein neues Datum, welches X Tage nachdem ursprünglichem Datum ist. (Falls nötig finden Sie weitere Informationen unter `https://www.postgresql.org/docs/9.4/static/datatype-datetime.html` bzw. `https://www.postgresql.org/docs/9.4/static/functions-datetime.html`.)

(c) Erstellen Sie Trigger, welche den gespeicherten Wert "dauer" einer aktualisiert sobald sich die Dauer (d.h. der Wert "dauer" dieser Touren) der Teiltouren verändert. Dazu benötigen Sie zumindest zwei Trigger:

- Ein Trigger gibt die Veränderung zwischen alter und neuer Dauer an die übergeordneten Touren weiter. *Hinweis*: `AFTER UPDATE ON TABLE Tour`.
- Ein Trigger addiert/subtrahiert die Dauer einer Tour zu/von der Dauer der übergeordneten Tour sobald eine Tour Teil/nicht mehr Teil einer anderen Tour ist.

  Hinweis: `AFTER INSERT OR UPDATE OR DELETE ON TABLE TeilTour`.

Bedenken Sie, dass Sie sich nicht um die rekursiven Abhängigkeiten zwischen den Tourteilen kümmern müssen, sondern dass Trigger sehr wohl wieder Trigger ausführen, usw.

(d) Erstellen Sie eine Procedure `CreateCrew` die automatisch eine Crew zusammenstellt. Die Procedure hat folgende Parameter: Eine `id` eines Skippers, der die Crew leiten wird, einen Namen `name` für die Crew, die Anzahl der notwendigen Ersthelfer, die Anzahl der notwendigen Köche und die Anzahl der sonstigen Mitglieder. Beachten Sie folgendes:

- Die Anzahl der notwendigen Ersthelfer und Köche muss mit den Bedingungen im ER-Diagramm übereinstimmen (mind. 2 Ersthelfer, mind. 1 und max. 4 Köche).
- Finden Sie eine geeignete Crew ID `cid`, z.B. die erste Crew des Skippers mit der `id` X erhält die `cid` 1, die zweite Crew die `cid` 2, etc.

- Finden Sie Köche die noch nicht Crewmitglieder sind und fügen Sie sie zur Crew hinzu. Sie können annehmen, dass ausreichend Köche angelegt wurden.
- Finden Sie Ersthelfer die noch nicht Crewmitglieder sind und fügen Sie sie zur Crew hinzu. Sie können annehmen, dass ausreichend Ersthelfer angelegt wurden.
- Geben Sie am Ende mittels `RAISE NOTICE` die neue Crewid `cid` aus.

  Hinweis: Mehr Informationen zur Ausgabe von Meldungen mittels `RAISE` finden Sie in der Online Dokumentation: `https://www.postgresql.org/docs/9.4/static/plpgsql-errors-and-messages.html`.

Aufgabe 5 (Löschen der angelegten Objekte) *[1 Punkte]*

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 Punkte]*

(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, auche positive, abgedeckt werden, z.B. Chartern eines Segelschiffs mit ausreichender und zu geringer Kapazitäet, oder Anlegen eines Schiffes ohne `sid`, etc.

(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) `queries.sql`
  (3) `plpgsql.sql`
  (4) `insert.sql`
  (5) `test.sql`
  (6) `drop.sql`

  Fügen Sie alle Dateien zu einer ZIP-Datei `blatt4.zip` hinzu und laden Sie diese in TUWEL hoch.

EER-Diagramm

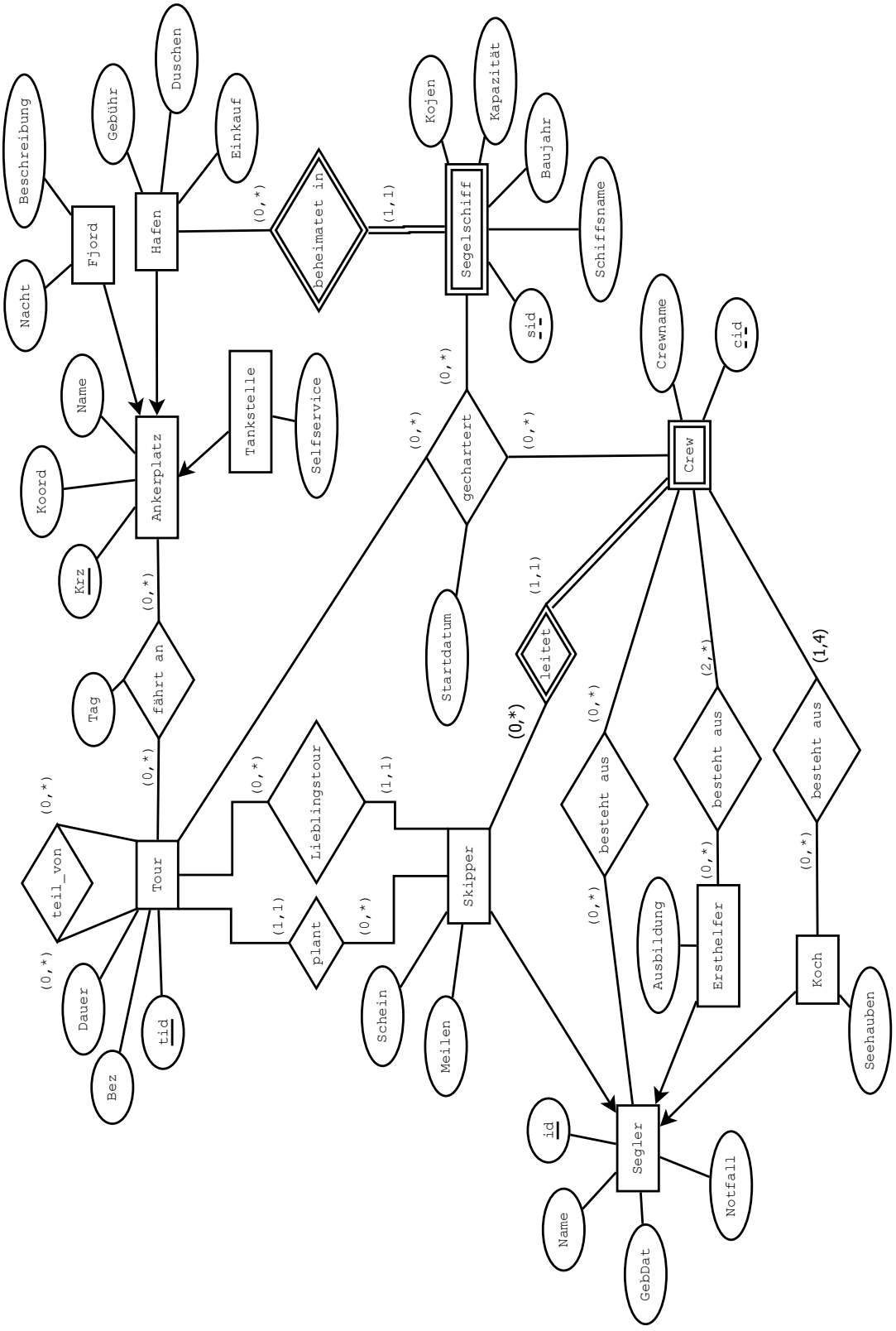

Abbildung 1: EER-Diagramm zu diesem Übungszettel