Datenbanksysteme (VU 4.0, 184.686)
Übungsteil, WS 2013/2014

Beispiel 2

Praktikums-Austauschprogramme bieten Studierenden die Möglichkeit international Erfahrung im Rahmen eines Praktikums bei einem Unternehmen zu sammeln. Erstellen Sie für eine Organisation zur Verwaltung von Austauschprogrammen eine Datenbank, die folgende Informationen speichert:

- Ein Austauschprogramm wird über das Land, in welchem es stattfindet, das Jahr und den Namen eindeutig identifiziert. Zusätzlich wird die Bewerbungsfrist zum Austauschprogramm gespeichert. Jedes Austauschprogramm wird von einer Kommission verwaltet, wobei eine Kommission ein oder mehrere Austauschprogramme verwalten kann.
- Eine Kommission wird über eine künstliche ID eindeutig identifiziert. Zusätzlich werden noch Standort und Name der Kommission gespeichert. Jede Kommission kann eine übergeordnete und beliebig viele untergeordnete Kommissionen haben.
  Kommissionen werden von einem Mitarbeiter koordiniert und es können beliebig viele Mitarbeiter in einer Kommission arbeiten. Jeder Mitarbeiter kann eine Kommission leiten und in einer Kommission arbeiten.
  Ein Mitarbeiter wird über seine Sozialversicherungsnummer (SVNR) eindeutig identifiziert. Weiters werden noch der Name, Anschrift, Gehaltsklasse und das Datum der Einstellung gespeichert. Neben normalen Mitarbeitern gibt es auch Betreuer. Jeder Betreuer ist automatisch auch ein Mitarbeiter. Bei Betreuern wird zusätzlich noch ein Bonus in der Datenbank vermerkt.
- Außerdem sollen vom System auch noch die Firmen erfasst werden. Eine Firma wird über eine Nummer eindeutig identifiziert. Weiters besitzt sie noch die Attribute Standort und Name. Eine Firma kann von beliebig vielen Betreuern betreut werden. Jeder Betreuer kann auch beliebig viele Firmen betreuen.
  Eine Firma besteht aus einer oder mehreren Abteilungen. Diese werden ebenfalls über eine Nummer eindeutig pro Firma identifiziert. Außerdem wird noch der Name der Abteilung vermerkt.
  Jede Abteilung kann beliebig viele Praktika anbieten. Ein Praktikum wird wiederum über eine Nummer eindeutig pro Firma und Abteilung identifiziert. Außerdem werden noch der Aufgabenbereich, eine Beschreibung sowie ein von- und ein bis-Datum gespeichert.
  Praktika können zu beliebig vielen Austauschprogrammen zugeordnet sein.

Für ein Austauschprogramm können auch beliebig viele Praktika ausgeschrieben sein.

- Studenten werden auch in der Datenbank gespeichert. Ein Student wird über eine SVNR eindeutig identifiziert. Dazu werden noch Anschrift, Name, Studienkennzahl sowie dessen Matrikelnummer im System erfasst. Optional kann ein Student Bewerbungsunterlagen besitzen. Studenten können sich zu beliebig vielen Austauschprogrammen bewerben. Zu jedem Austauschprogramm können beliebig viele Bewerber vorhanden sein. Wird ein Student von der zuständigen Kommission zu einem Austauschprogramm akzeptiert, so kann er sich zu Praktika des Austauschprogramms bewerben. Studenten, welche akzeptiert wurden, sind "shortlisted". Jeder "shortlisted" Student ist automatisch auch ein Student. Sie können sich zu bis zu zwei Praktika bewerben, wobei sie für jede Bewerbung auch eine Präferenznummer angeben müssen. Zu jedem Praktikum können sich beliebig viele "shortlisted" Studenten bewerben. Sollte eine Firma jetzt einen "shortlisted" Studenten akzeptieren, so ist er "matched". Jeder "matched" Student ist automatisch auch ein "shortlisted" Student und wird von einem Betreuer zu dem Praktikum betreut.

Sie finden nun im folgenden Bild ein ER-Diagramm, das die grobe Struktur der Datenbank abbildet, wobei Sie davon ausgehen können, dass die Abbildung den Sachverhalt als auch alle Kardinalitäten bereits korrekt abdeckt. Ihre Aufgabe ist es nur mehr, die korrekten CREATE-, INSERT- und DROP-Befehle zu erstellen, um alle vorhin genannten Informationen verwalten zu können.

Wichtig
Treffen Sie geeignete Annahmen für die Schlüssel und Fremdschlüssel der einzelnen Tabellen!

ER-Diagramm

Bei der Überführung in die CREATE-Anweisungen werden Sie feststellen, dass NULL-Werte durch das ERD erlaubt sind. Beim Abgabegespräch sollten Sie in der Lage sein, die betroffenen Tabellen zu nennen und Vorschläge zu machen, wie man diese NULL-Werte vermeiden kann. Es ist aber nicht notwendig, NULL-Werte bei den CREATE-Befehlen zu vermeiden und die mündliche Erklärung während des Abgabegesprächs ist ausreichend.

PL/pgSQL

Trigger:

1. Über die Tabelle ausgeschrieben wird ein Praktikum einem Austauschprogramm zugeordnet. Beim Einfügen in diese Tabelle, soll sichergestellt werden, dass das von- und bis-Datum des Praktikums nicht vor dem Jahr liegen, für welches das Austauschprogramm ausgeschrieben ist. Schreiben Sie einen Trigger für die Tabelle ausgeschrieben, der diese

beiden Bedingungen sicherstellt.
2. Über die Tabelle zuständig werden Betreuer den Firmen zugeordnet, welche Sie betreuen. Dabei ist zu beachten, dass Betreuer, welche gleichzeitig auch Koordinatoren einer Kommission sind, diese Position nicht einnehmen dürfen.
   Weiters ist sicherzustellen, dass der Betreuer in einer Kommission arbeitet, welche für die Verwaltung eines Austauschprogrammes zuständig ist, für welches die entsprechende Firma ein oder mehrere Praktika ausgeschrieben hat.
   Schreiben Sie einen Trigger für die Tabelle zuständig der dieses Verhalten garantiert.

Function:

1. Schreiben Sie eine Funktion `f_calc_salary`, welche als Parameter die SVNR eines Mitarbeiters und ein Datum erhält.
   Die Funktion `f_calc_salary` soll nun für den übergebenen Mitarbeiter zu dem übergebenem Datum (berücksichtigen sie den kompletten Monat, unabhängig vom übergebenen Tag), das auszuzahlende oder bereits ausgezahlte Monatsgehalt berechnen und als NUMERIC(7,2) zurückliefern.
   Das Grundgehalt, wird über die Gehaltsklasse identifiziert ("C1" = 3000, "C2" = 2500, "M1" = 2000, "M2" = 1500, "M3" = 1000).
   Sollte der Mitarbeiter ein Betreuer sein, kann es möglich sein, dass etwaige Bonuszahlungen anfallen. Der gespeicherte "Bonus" wird dem Betreuer dabei pro Student bezahlt, welchen er zu dem übergebenen Datum betreut hat. Beachten Sie dabei das von- und bis-Datum der Tabelle Praktikum.
   Stellen Sie sicher, dass die übergebene SVNR auch tatsächlich existiert, und, dass der Mitarbeiter zu dem übergebenen Datum überhaupt bereits angestellt war. Andernfalls beenden Sie die Verarbeitung mit einer Exception samt passendem Hinweis.

Procedure:

1. Schreiben Sie eine Prozedur `p_update_students`.
   Die Prozedur soll alle Studenten durchlaufen, und sie in der Hierarchie eine Stufe hinaufsetzen. Ein "normaler" Student wird zu einem "shortlisted" Student, "shortlisted" Studenten werden zu "matched" Studenten und ein "matched" Student bleibt weiterhin "matched".
   Geben Sie während der Abarbeitung der Prozedur aus, welche Studenten (SVNR reicht) zu welchem Typen in der Hierarchie geändert wurden.

Aufgabenstellung

1. Leiten Sie aus dem ER-Diagramm die Relationen der Datenbank in 3. Normalform so ab, dass sie verbundtreu und abhängigkeitstreu sind. Sie

können die Ableitung der Relationen gleich unmittelbar in der geforderten Datei `create.sql` vornehmen und müssen KEIN explizites Dokument für das Relationenmodell (wie in den vorangegangenen Semestern) erstellen.

2. Erstellen Sie eine Datei `create.sql`, in welcher die nötigen CREATE-Befehle gespeichert werden, um die Relationen mittels SQL zu realisieren. Dabei sind folgende Punkte zu beachten:
    - Wählen Sie für Geldbeträge keine Gleitkommatypen sondern z. B. NUMERIC mit zwei Nachkommastellen.
    - Realisieren Sie die fortlaufende Nummerierung der "id"/"nummer"-Attribute von Kommission, Abteilung und Praktikum mit Hilfe von Sequences. Die Sequence für den Schlüssel der Tabelle Firma soll bei 10 beginnen und in Zehnerschritten erhöht werden (d.h. 10, 20, 30, ...).
    - 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.
    - Verwenden Sie keine Umlaute für Bezeichnungen von Relationen, Attributen, etc.
    - Stellen Sie die folgenden Sachverhalte durch geeignete CHECK-Bedingungen sicher:
        - Der Bonus, der für einen Betreuer ausgezahlt wird, muss größer 0 sein.
        - Der Typ einer Gehaltsklasse muss aus der folgenden Liste an Möglichkeiten stammen: 'M1', 'M2', 'M3', 'C1', 'C2'
        - Die Bewerbungsfrist des Austauschprogramms darf spätestens am 31.12. des Jahres, für welches das Programm ausgeschrieben ist liegen.
        - Bei einem Praktikum muss das von-Datum kleiner oder gleich dem bis-Datum sein.

3. 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" Austauschprogramme, Kommissionen, etc. wählen. Stattdessen können Sie auch einfach "Austauschprogramm 1", "Austauschprogramm 2", "Kommission 1", "Kommission 2" etc. verwenden.

4. Erstellen Sie eine Datei `plpgsql-teil.sql`, welche den Code für die beiden Trigger, die Funktion `f_calc_salary` und die Prozedur `p_update_students` enthält.

5. Erstellen Sie eine Datei `drop.sql`, welche die nötigen DROP-Befehle enthält, um alle in den Punkten 2 und 4 erzeugten Datenbankobjekte wieder zu löschen. Das Schlüsselwort CASCADE darf dabei **NICHT** verwendet werden.

6. Überlegen Sie sich eine sinnvolle Testabdeckung für die PL/pgSQL-Programmteile laut Punkt 5, z.B.: Erweiterung der Testdaten, Aufruf der zu testenden PL/pgSQL-Programmteile mit entsprechenden Ausgaben, so dass sich die erfolgreiche Durchführung der Tests überprüfen lässt. Denken Sie auch an negative Tests, welche mit einer Exception enden sollen, z.B. Aufruf der Funktion `f_calc_salary` mit einer falschen SVNR. Stellen Sie in Ihrem ZIP-Archiv die SQL-Dateien mit den zusätzlichen INSERT-Befehlen und den "Testtreibern" in der Datei `test.sql` bereit. Sie müssen in der Lage sein, diese SQL-Dateien und PL/pgSQL-Dateien im Rahmen des Abgabegesprächs ablaufen zu lassen.

7. Stellen Sie in Ihrem Abgabearchiv eine Listing-Datei mit dem Namen `listing.txt` bereit, die Sie bei der Ausführung der SQL-Dateien erzeugt haben. Diese Datei soll alle Informationen beinhalten, die beim Ablauf der Dateien `create.sql, insert.sql, plpgsql-teil.sql, test.sql` und `drop.sql` erzeugt werden. Beachten Sie dazu bitte die Hinweise zur Benützung von psql am Ende dieses Dokuments.

8. Im Rahmen des Abgabegesprächs müssen Sie in der Lage sein, alle von Ihnen in der Abgabe bereitgestellten Dateien erklären zu können. Weiters wird erwartet, dass Sie auch das ER-Diagramm einwandfrei analysieren können, auch wenn es in diesem Semester bereits vorgegeben wurde. Bitte achten Sie auch darauf, dass sich alle von Ihnen abgegebenen Dateien auf dem bordo-Server ausführen lassen, um Probleme (und möglicherweise daraus resultierenden Punktabzug) zu vermeiden

Abgabe

Zusammenfassend sind nun folgende Dateien abzugeben:

- `create.sql`
- `insert.sql`
- `drop.sql`
- `plpgsql-teil.sql`
- `test.sql`
- `listing.txt`

Die aufgezählten Dateien sind in einer ZIP-Datei `beispiel2.zip` bis spätestens **03.11.2013** um **23:59** im CourseManager abzugeben. Es wird stets die zuletzt hochgeladene Version Ihrer Lösung gewertet.
Achten Sie darauf, dass der Name jeder dieser abzugebenden Dateien so lautet wie oben beschrieben. Erstellen Sie keine Ordner innerhalb der ZIP-Datei.

Abgabegespräche

Die Verteilung der Punkte erfolgt nach folgendem Schlüssel:

- SQL Teil (`create.sql`, `insert.sql`, `drop.sql`): max. 5 Punkte
- PL/pgSQL Teil (`plpgsql-teil.sql`, `test.sql`): max. 5 Punkte
- `listing.txt`: Vorhandensein für das Erreichen der vollen Punkteanzahl notwendig

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 den Abgabegesprächen in der Lage sein, nicht nur Ihre Beispiele zu erklären, sondern ebenfalls zeigen, dass Sie die bisher 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 Abgabegespräche selbst testen und gegebenenfalls vertiefen.

Die volle Punktezahl gibt es nur, wenn die Beispiele korrekt gelöst wurden und die Lösung einwandfrei erklärt werden kann. Nicht selbstständig gelöste Abgaben werden mit 0 Punkten bewertet!

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

Bringen Sie bitte Ihren Studentenausweis zur Abgabe mit. Eine Abgabe ohne Ausweis ist nicht möglich.

Hinweise zur Verwendung von psql

Folgende Befehle können für Ihre Arbeit mit der interaktiven SQL-Shell psql von PostgreSQL hilfreich sein:

- `\?`: Listet alle psql-internen Befehle samt Erklärung auf.
- `\i <dateiname>`: Führt das Skript `<dateiname>` aus. Beispiel: `\i create.sql`
- `\o <dateiname>`: Lenkt die Ausgabe in eine Datei mit dem Namen `<dateiname>` um. Lässt man den Parameter `<dateiname>` weg, so wird dieses Verhalten wieder abgestellt. `Beispiel: \o listing.txt`