Datenbanksysteme (VU 4.0, 184.686)
Übungsteil, WS 2014/15

Beispiel 2

Ein Verbund von Krankenanstalten entschließt sich, ihre Patientendaten zentral zu sammeln. Erstellen Sie für diesen Krankenanstaltsverbund eine Datenbank, die folgende Informationen speichert:

- Personen werden über die Sozialversicherungsnummer eindeutig identifiziert. Zusätzlich wird der Name und die Anschrift jeder Person gespeichert.
  Jede Person kann ein Krankenhausmitarbeiter und/oder ein Patient sein. Für jeden Patienten wird das Datum, seit dem er Patient ist, und ob er bereits geheilt ist, gespeichert. Beim Anlegen eines Patienten wird das Datum, seit er Patient ist, falls nicht angegeben, auf das aktuelle Datum gesetzt; ebenso wird geheilt auf Falsch gesetzt. Für jeden Mitarbeiter wird das Datum, seit dem er Mitarbeiter ist, und dessen Gehalt gespeichert. Das Gehalt wird als Stundenlohn angegeben. Zu jedem Mitarbeiter werden die Monatslohnzettel gespeichert. Ein Monatslohnzettel weist ein Monat, ein Jahr und ein Honorar auf.
  Jeder Mitarbeiter kann auch ein Arzt sein. Alle Ärzte sind auf Werksvertragsbasis im Krankenhaus angestellt.
- Ein Krankenhaus hat einen Namen und eine Anschrift. Jedes Krankenhaus wird durch einen Mitarbeiter geleitet und besteht aus ein oder mehreren Abteilungen. Diese hat wiederum eine Anschrift und Namen. In jeder Abteilung arbeiten mehrere Mitarbeiter, jedoch wird jede Abteilung nur von einem Mitarbeiter koordiniert. Jede Abteilung spezialisiert sich auf eine oder mehrere Krankheiten.
- Eine Krankheit hat einen Namen und einen Bonusmultiplikator. Der Multiplikator gibt an, um wie viel das Grundgehalt des Arztes erhöht wird. Zusätzlich ist jede Krankheit einer Klasse zugeordnet. Jede Klasse kann einer oder mehreren anderen Klasse übergeordnet sein.
- Patienten werden für eine bestimmte Dauer von einem Arzt zu einer bestimmten Krankheit behandelt. Ärzte dürfen nur Krankheiten behandeln, auf die sich die Abteilung, in der sie arbeiten, spezialisiert hat. Ein Patient wird immer nur zu einer Krankheit behandelt. Bei der Behandlung wird zusätzlich gespeichert, ob sie bereits abgerechnet wurde.
- Zu jeder Person werden historische Krankendaten gespeichert. Sobald ein Patient geheilt ist, wird diese Information in einem Akteneintrag abgelegt. Eine Person hat mehrere Akteneinträge. Ein Akteneintrag enthält ein von

und ein bis Datum. Zusätzlich werden das behandelnde Krankenhaus und die Krankheit gespeichert.

Sie finden nun im folgenden Bild ein ER-Diagramm, das die grobe Struktur der Datenbank abbildet, wobei Sie davon ausgehen können, dass die Abbildung sowohl den Sachverhalt als auch alle Kardinalitäten bereits korrekt abdeckt.

ER-Diagramm

Ihre Aufgabe ist es nun:

- Die Schlüsselattribute für alle Entitäten einzeichnen. Treffen Sie geeignete Annahmen. Sie können dazu Dia verwenden und das vorhande Diagramm (ERD_krankenhaus.dia) erweitern.
- Die korrekten CREATE-, INSERT- und DROP-Befehle erstellen, um alle vorhin genannten Informationen verwalten zu können.
- Vergessen Sie nicht, die Schlüssel und Fremdschlüssel der einzelnen Tabellen korrekt abzubilden!

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. Schreiben Sie einen Trigger, der über die Relation "behandelt" folgende Bedingungen überprüft:
   a. Ein Patient kann immer nur zu einer Krankheit behandelt werden.
   b. Ein Arzt kann nur Krankheiten behandeln, auf die die Abteilung, in der er arbeitet, spezialisiert ist.
2. Der Mitarbeiter, der eine Abteilung koordiniert, muss auch in dieser arbeiten.
3. Der Mitarbeiter, der das Krankenhaus leitet, muss auch in einer der Abteilungen des Krankenhauses arbeiten.

Function:

1. Schreiben Sie eine Funktion `f_calc_salary`, welche als Parameter die SVNR eines Mitarbeiters, ein Monat und ein Jahr erhält.
   Die Funktion `f_calc_salary` soll nun für den übergebenen Mitarbeiter sein ausstehendes Honorar für das übergebene Monat und Jahr berechnen. Dieses berechnet sich wie folgt:
   - Falls der Mitarbeiter auch ein Arzt ist: Für jeden geheilten Patienten mit nicht abgerechneten Behandlungen wird die Dauer der Behandlung mit dem Gehalt und dem Bonusmultiplikator der Krankheit multipliziert. Die Summe über alle Patienten wird dann ausgegeben.
   - Falls der Mitarbeiter kein Arzt ist: Wenn im übergebenen Monat noch kein Gehaltszettel angelegt wurde (bzw. das Honorar gleich 0 ist), wird das eingetragene Gehalt mit 167 (durschnittliche Anzahl der Arbeitsstunden pro Monat) multipliziert.

Procedure:

1. Schreiben Sie eine Prozedur `p_move_healed`.
   Die Prozedur soll folgende Tätigkeiten durchführen:
   a. Für alle geheilten Patienten, bei denen alle Behandlungen abgerechnet wurden, wird ein Akteneintrag in der Krankenakte angelegt. Das bis-Datum entspricht dem aktuellen Datum. Bitte beachten Sie das alle Informationen richtig in die Krankenakte übertragen werden.

    b. Alle geheilten und komplett abgerechneten Patienten werden zusammen mit deren Behandlung gelöscht.
2. Schreiben Sie eine Prozedur `p_calc_salary`.
   Berechnet fuer jeden Mitarbeiter das aktuelle Monatsgehalt (`f_calc_salary`) und legt die Monatslohnzettel an. Vergessen Sie nicht die Behandlungen auf abgerechnet zu setzen.

Aufgabenstellung

1. Erweitern Sie das ER-Diagramm um Schlüsselattribute.

2. 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.

3. 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 künstlichen Primärschlüssel-Attribute mit Hilfe von Sequences. Die Sequence für den Schlüssel der Tabelle Krankenhaus 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 für eine Krankheit muss größer gleich 1 sein.
     - Das Gehalt muss größer 0 sein.
     - Bei einem Akteneintrag muss das von-Datum kleiner oder gleich dem bis-Datum sein.

4. 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" Krankenhäuser, Abteilungen, Krankheiten, etc. wählen. Stattdessen können Sie auch einfach "Krankenhaus 1", "Krankenhaus 2", "Krankheit 1", "Krankheit 2" etc. verwenden.

5. Erstellen Sie eine Datei `plpgsql-teil.sql`, welche den Code für die Trigger, die Funktion `f_calc_salary` und die Prozeduren `p_move_healed` und `p_calc_salary` enthält.

6. 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.

7. 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.

8. 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.

9. 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:

- `ERD_krankenhaus.png`
- `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 **02.11.2014** 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:

- ER-Diagramm Teil (`ERD_krankenhaus.png`): max. 1 Punkt
- SQL Teil (`create.sql`, `insert.sql`, `drop.sql`): max. 4 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`