Hochschule Anhalt
Fachbereich Informatik und Sprachen
Anika Groß

Datenbanken und Anwendungsentwicklung
Praktisches Projekt – Leistungsnachweis (LNW)

In unserem praktischen Projekt soll zunächst der erste Schritt der Erstellung einer vollständigen Datenbankanwendung durchgeführt, der Datenbank-Entwurf (Teil 1). In Teil 2 sollen Anfragen auf der erstellten Datenbank formuliert werden und ein fortgeschrittener Mechanismus zu Änderungsoperationen implementiert werden. [Info: Im später folgenden 3. Teil werden Sie eine Java-Applikation implementieren, die die hier implementierte Datenbank nutzt.]

Die folgenden Aufgaben in Teil 1 und Teil 2 sind für das Bestehen des LNW zu lösen. Beachten Sie die im Anschluss erläuterten Bedingungen zur Abgabe und Bewertung der Aufgaben.

TEIL 1 (20 Punkte)

a) **Relationenmodell (5 Punkte)**
Überführen Sie das bereitgestellte UML-Diagramm in ein valides Relationenmodell. Benennen Sie alle Relationen, Attribute und Datentypen, die Sie für den Entwurf verwenden. Achten Sie darauf, dass Kindklassen in Vererbungshierarchien (z.B. City-[erbt von]->Place, Post-[erbt von]->Message, Company-[erbt von]->Organization) als eigene Relationen modelliert werden. Entscheiden Sie je nach Kardinalität einer Beziehung, ob es sich um eine 1:1, 1:n oder n:m Beziehung handelt und erzeugen Sie die beteiligten Relationen entsprechend.
Wichtig: das dargestellte Modell im UML-Diagramm ist kein valides Relationenmodell! Verwenden Sie unbedingt eine korrekte Notation für das Relationenmodell.

b) **Integritätsbedingungen (5 Punkte)**
Formulieren Sie relevante Bedingungen, die für die *Integrität* Ihrer Datenbank wichtig sind.
- Primärschlüsselbedingung und referentielle Integrität (Definieren Sie auch entsprechende Lösch- und Updateregeln für Fremdschlüsselbeziehungen)
- Formulieren Sie mindestens folgende Attributbedingungen:
  i. Datentypen entsprechend der Quelldaten (z.B. timestamp fuer 'creationDate')
  ii. Der Geburtstag einer Person darf nicht in der Zukunft liegen
  iii. Die Email-Adresse einer Person muss valide sein

c) **SQL-DDL-Skript (5 Punkte)**
Entwickeln Sie anhand Ihres Relationenmodells (aus Aufgabe 1) ein SQL-Skript, welches die notwendigen Tabellen, Schlüssel-Constraints und die weiteren Integritätsbedingungen (aus Aufgabe 2) in *PostgreSQL* erzeugt.

d) **Datenimport (5 Punkte)**
Importieren Sie die bereitgestellten CSV-Daten in Ihre Datenbank. Beachten Sie, dass die bereitgestellten CSV-Daten nicht dem Zielschema Ihrer Datenbank entsprechen. Das Laden der Daten kann auf zwei Wegen erfolgen:
- Variante 1 (Programm): Schreiben Sie ein Ladeprogramm, welches die CSV-Daten einliest, entsprechend Ihrem Zielschema konvertiert und in die Datenbank schreibt.
- Variante 2 (SQL): Nutzen Sie für die Transformation die von der Datenbank bereitgestellten Mittel. Laden Sie die CSV-Daten in temporäre Tabellen via `COPY` und transformieren Sie die Daten in Ihr Zielschema via SQL.

TEIL 2 (15 Punkte)

a) Sichtenerstellung (2 Punkte)

Die Freundschaftsbeziehung ist als gerichtete Beziehung gespeichert, um Anfragen bezüglich der Freundschaftsbeziehung komfortabel zu lösen sollen die Beziehungen ungerichtet gespeichert werden. Diesbezüglich sollen Sie eine Sicht "pkp_symmetric" erstellen, die beide Richtungen enthält.

b) Anfragen auf der Datenbank (10 Punkte)

Formulieren Sie SQL-Anfragen, um folgende Fragen zu beantworten.

1) In wie vielen verschiedenen afrikanischen Städten gibt es eine Universität?
2) Wie viele Forenbeiträge (Posts) hat die jüngste Person verfasst (Ausgabe: Name, #Forenbeiträge)?
3) Wie viele Kommentare zu *Posts* gibt es aus jedem Land (Ausgabe aufsteigend sortiert nach Kommentaranzahl)? Die Liste soll auch Länder enthalten, für die keine Post-Kommentare existieren, d.h. die Kommentaranzahl = 0 ist! (Funktion Coalesce)
4) Aus welchen Städten stammen die meisten Nutzer (Ausgabe Name + Einwohnerzahl)?
5) Mit wem ist 'Hans Johansson' befreundet?
6) Wer sind die "echten" Freundesfreunde von 'Hans Johansson'? "Echte" Freundesfreunde dürfen nicht gleichzeitig direkte Freunde von 'Hans Johansson' sein. Sortieren Sie die Ausgabe alphabetisch nach dem Nachnamen.
7) Welche Nutzer sind Mitglied in allen Foren, in denen auch 'Mehmet Koksal' Mitglied ist (Angabe Name)?
8) Geben Sie die prozentuale Verteilung der Nutzer bezüglich ihrer Herkunft aus verschiedenen Kontinenten an!
9) Welche Foren enthalten mehr Posts als die durchschnittliche Anzahl von Posts in Foren (Ausgabe alphabetisch sortiert nach Forumtitel)?
10) Welche Personen sind mit der Person befreundet, die die meisten Likes auf einen Post bekommen hat? Sortieren Sie die Ausgabe alphabetisch nach dem Nachnamen.

Zusatz (4 Zusatzpunkte)**:**

11) Welche Personen sind direkt oder indirekt mit 'Jun Hu' (id 94) verbunden (befreundet)? Geben Sie für jede Person die minimale Distanz zu Jun an.
12) Erweitern Sie die Anfrage zu Aufgabe 11 indem Sie zusätzlich zur Distanz den minimalen Pfad zwischen den Nutzern ausgeben.
    Hinweis: Die Kardinalität der Ergebnismenge ist größer, da zwischen zwei Personen mehrere minimale Pfade existieren können.

c) Änderungen in der erzeugten Datenbank (3 Punkte)

- Es soll ein Mechanismus umgesetzt werden, um die Beendigung eines Arbeitsverhältnisses zu dokumentieren. Der entsprechende Eintrag in `person_workAt_company` soll mittels SQL-Anweisung gelöscht werden. Um die Datenmanipulation nachvollziehen zu können, soll der Löschvorgang in einer separaten Tabelle protokolliert werden. Dabei soll zusätzlich hinterlegt werden, wann das Arbeitsverhältnis beendet wurde (orientieren Sie sich am Löschzeitpunkt). Die Protokollierung soll **automatisch** erfolgen, wenn ein Mitarbeiter sein Arbeitsverhältnis bei einem Unternehmen beendet (Löschung in `person_workAt_company`). Formulieren Sie eine Löschanweisung, die zeigt, dass Ihr Mechanismus funktioniert.

Darstellung der Lösung

Abzugeben sind:

- Relationenmodell, inklusive aller in der Aufgabenstellung geforderten Angaben (Aufgabe 1a+b)
- SQL-DDL Skript zum Erzeugen der Datenbank (Aufgabe 1c)
- SQL-Skipt / Programm zum Importieren der CSV-Daten in die Datenbank (Aufgabe 1d)
- SQL-Skript mit den SQL-Statements aus Aufgabe 2a, 2b und 2d

Alle Dateien Ihres Projekts **müssen** in einem **GitLab-Repository** der Hochschule Anhalt abgelegt werden. Sie nutzen ein Repository gemeinsam als Team. Den Link zu Ihrem Repository und speziell den Unterordner mit den abzugebenden Dateien, teilen Sie mir bei Ihrer Abgabe mit.

Die Abgabe **muss bis zum 04.01.2021** per Moodle über ein Mitglied jeder Gruppe erfolgen. Bitte laden Sie eine **Textdatei in Moodle** hoch, die folgende Informationen enthält:

- Name und Matrikelnummer aller Teammitglieder
- Link zum Ordner mit den abzugebenden Dateien in Ihrem GitLab-Repository.

Zusätzlich ist die Lösung in einem Testat zu präsentieren. Jedes Gruppenmitglied muss am Testat teilnehmen und Fragen zur eingereichten Lösung beantworten können, um den LNW zu bestehen. Die Testate werden gruppenweise voraussichtlich am 08.01.2021 (bei Bedarf auch am 13.01.2021 zum Vorlesungs- bzw. Praktikumstermin stattfinden und pro Gruppe ca. 20 Minuten umfassen. Die genauen Termine vereinbaren wir rechtzeitig nach Absprache in der Vorlesung/im Praktikum.

*Es ist auch **möglich** die **Lösungen bereits am 14.12.2020** abzugeben und das **Testat am 16.12.2020** zu machen, wenn Sie gern die LNW-Aufgabe vor den Feiertagen abschließen möchten.*

Bedingungen zum Bestehen des LNW

1) Ihre Lösungen zu Teil 1 müssen je Teilaufgabe, d.h. 1a, 1b, 1c und 1d mit jeweils mindestens 50% der Punkte bewertet werden.
2) Ihre Lösungen zu Teil 2 müssen mit mindestens 50% der Punkte bewertet werden.
3) Die Teilnahme am Gruppentestat ist verpflichtend.
4) Sie diskutieren Ihre Lösung und beantworten Fragen zur Lösung im Gruppentestat je Gruppenmitglied.
5) Erfüllen von Auflagen: Sie erhalten gegebenenfalls Auflagen, was Sie an Ihrer Lösung korrigieren müssen. Diese Auflagen müssen Sie umsetzen und mir in nachvollziehbarer Weise zusenden. Das hat den Hintergrund, dass Sie mit Ihrer Datenbank im Projekt weiterarbeiten und darauf aufbauen für die später prüfungsrelevanten Aufgaben (Teil 3).

Details zum Schema und zu den Daten

Daten

- UML-Diagramm der Miniwelt
- Testdatensatz für den Datenimport

Miniweltbeschreibung

Das bereitgestellte UML-Diagramm definiert die Entitäten eines online sozialen Netzwerkes und ihre Beziehungen zueinander. Die bereitgestellten Daten repräsentieren die Aktivität innerhalb des sozialen Netzwerkes in einem festgelegten Zeitraum. Das Schema modelliert Entitäten, wie zum Beispiel Personen, Organisationen und Orte. Außerdem werden Beziehungen, wie zum Beispiel Freundschaften zwischen Personen und die Interaktion zwischen Personen und z.B. Nachrichten / Kommentaren abgebildet.

Entitäten: Entitäten besitzen in den bereitgestellten Daten eine eindeutige Identität.

- **`Person`:** Eine Person repräsentiert eine real existierende Person innerhalb des sozialen Netzwerkes. Zur Person werden verschiedene Daten und Beziehungsinformationen abgelegt.
- **`Tag` (Thema/Konzept):** Ein Tag definiert ein gewisses Konzept, welches genutzt wird um zum Beispiel die Interessen von Personen zu definieren oder die Inhalte von Foren und Beiträgen auszuzeichnen.
- **`TagClass`:** Tags sind (evtl. mehreren) Klassen (TagClass) zugeordnet. Die Klassen selbst bilden eine Hierarchie.
- **`City` (Stadt):** Eine Stadt repräsentiert eine real existierende Stadt. Eine Stadt ist genau einem Land zugeordnet.
- **`Country` (Land):** Ein Land repräsentiert ein real existierendes Land. Länder enthalten mehrere Städte und sind Teil eines Kontinents.
- **`Continent`** (**Kontinent):** Ein Kontinent repräsentiert einen real existierenden Kontinent und ist somit nicht in einem anderen Ort enthalten.
- **`Company` (Firma):** Eine Firma ist ein Ort, an dem eine Person arbeitet. Eine Firma hat einen Hauptsitz in einem Land.
- **`University` (Universität):** Eine Universität ist ein Ort, an dem Personen studieren.
- **`Forum`:** Ein Forum ist ein Treffpunkt, an dem Personen Beiträge und Kommentare verfassen. Foren werden anhand von Tags charakterisiert. Es existieren drei Arten von Foren: Die persönliche "Wall" einer Person, Bilderalben und Gruppen. Die Unterscheidung erfolgt anhand des Forentitels.
- **`Post` (Beitrag):** Eine Nachricht, welche in einem Forum geschrieben wird. Beiträge sind genau einem Forum zugeordnet und können entweder textuellen Inhalt oder ein Bild beinhalten, niemals beides.
- **`Comment` (Kommentar):** Ein Kommentar wird von einer Person zu einer existierenden Nachricht (Message) abgegeben. Diese Nachricht kann entweder ein Post oder ein anderer Kommentar sein.

Beziehungen: Beziehungen verbinden Entitäten unter Verwendung ihrer jeweiligen Identität.

- **`Forum_containerOf_Post`:** Beschreibt das Enthaltensein eines Beitrages in einem Forum.
- **`Message_hasCreator_Person`:** Definiert die Person, welche eine Nachricht (Post oder Comment) erzeugt hat.
- **`Forum_hasMember_Person`:** Definiert, dass eine Person Mitglied eines Forums ist.
- **`Forum_hasModerator_Person`:** Definiert, dass eine Person der Moderator eines Forums ist.
- **`Forum_hasTag_Tag`:** Definiert das Thema/Konzept eines Forums.
- **`Message_hasTag_Tag`:** Definiert das Thema einer Nachricht (Post oder Comment).
- **`Tag_hasType_TagClass`:** Definiert die Klasse eines Tags.
- **`TagClass_isSubclassOf_TagClass`:** Definiert eine Vererbungshierarchie zwischen Tag-Klassen.
- **`Company_isLocatedIn_Country`:** Legt fest, in welchem Land sich eine Firma befindet.
- **`University_isLocatedIn_City`:** Legt fest, in welcher Stadt sich eine Universität befindet.
- **`Person_isLocatedIn_City`:** Legt fest, in welcher Stadt sich eine Person befindet.
- **`Message_isLocatedIn_Country`:** Das Land, in dem die Nachricht verfasst wurde.
- **`Person_knows_Person`:** Legt fest, dass sich zwei Personen kennen.
- **`Person_likes_Message`:** Definiert, dass eine Person eine Nachricht (Post oder Kommentar) mag.
- **`Comment_replyOf_Message`:** Legt fest, dass ein Kommentar die Antwort auf einen anderen Kommentar oder auf einen Post ist.
- **`Person_studyAt_University`:** Legt fest, an welcher Universität eine Person studiert.
- **`Person_workAt_Company`:** Legt fest, in welcher Firma eine Person arbeitet.