Database and Artificial !ntelligence Group

Datenbanksysteme (VU 4.0, 184.686) Übungsteil, WS 2011/2012

Beispiel 2

Erstellen Sie für einen App Store eine Datenbank, die folgende Informationen speichert:

- Um Artikel herunterladen zu können, ist eine Registrierung als Benutzer notwendig. Dabei werden der eindeutige Benutzername ("name"), ein Passwort ("passwort") und eine E-Mail-Adresse ("mail") gespeichert. Für Entwickler besteht darüber hinaus die Möglichkeit, ein Entwicklerkonto zu erstellen, für das ebenfalls diese Informationen und zusätzlich eine Beschreibung ("beschreibung") gespeichert werden.
- Eine Anwendung wird durch einen Namen ("name") identifiziert, und besitzt eine Beschreibung ("beschreibung") und einen Preis ("preis"). Es soll vermerkt werden, welcher Benutzer welche Anwendungen an welchem Datum ("datum") gekauft hat. Jede Anwendung hat einen oder mehreren Entwickler als Autoren. Von jeder Anwendung kann es verschiedene Versionen geben (aber mindestens eine), wobei eine Version einer Anwendung durch den Anwendungsnamen und die Versionsnummer ("vnr") eindeutig bestimmbar ist. Außerdem kann jede Version einer Anwendung bestimmte Plattformen unterstützen, die durch einen Namen ("name") und eine Versionsnummer ("vnr") identifiziert werden. Von allen Versionen, die es zu einer Anwendung gibt, gilt immer genau eine Version als empfohlen (etwa um standardmäßig eine möglichst aktuelle, aber stabile Version anzubieten). Zu jeder Anwendung wird die dazugehörige empfohlene Version gespeichert. (Hinweis: Modellieren Sie dies als Beziehung zwischen Anwendung und Version.)
- Es soll außerdem gespeichert werden, welcher Benutzer welche Version einer Anwendung für welche Plattform an welchem Datum ("datum") heruntergeladen hat, wobei ein Download durch eine fortlaufende Nummer ("id") identifiziert wird.
- Die angebotenen Anwendungen werden in Kategorien organisiert. Für eine Kategorie wird der eindeutige Name ("name") gespeichert, sowie beliebig viele Anwendungen, die unter diese Kategorie fallen. Eine Anwendung kann in beliebig vielen Kategorien untergebracht werden. Außerdem sollen Unterkategorien wie folgt modelliert werden: Unterkategorien sind Kategorien, die zusätzlich zum Namen eine genauere Beschreibung ("beschreibung") besitzen und mindestens einer Kategorie untergeordnet sind. Ein Beispiel (das Sie auch gerne für Punkt 4 verwenden können): Es gibt auf oberster Stufe die Kategorien "Spiele" und "Bildung". "Geschicklichkeitsspiele" ist eine Unterkategorie von "Spiele" und hat die Beschreibung "Spiele, die präzise Bedienung erfordern". "Lernspiele" ist eine Unterkategorie sowohl von "Spiele" als auch von "Bildung" und hat die Beschreibung: "Spiele als Ergänzung des Unterrichts". "Quizspiele" ist wiederum eine Unterkategorie von "Lernspiele" und hat als Beschreibung "Spiele, die Allgemeinwissen vermitteln".
- Benutzer können Anwendungen bewerten. Eine Bewertung wird durch eine fortlaufende Nummer ("id") identifiziert, und besteht aus einer Punkteanzahl ("punkte", enthält ganze Zahlen zwischen 0 und 5) und einem Kommentar ("kommentar"). Für jede Bewertung soll der Rezensent und die bewertete Version der jeweiligen Anwendung gespeichert werden.
- Entwickler sollen die Möglichkeit haben, Inserate aufzugeben, die durch eine fortlaufende Nummer ("id") identifiziert werden, und aus einem Titel ("titel") und einem Text ("text") bestehen. Für jedes Inserat soll gespeichert werden, welcher Entwickler es aufgegeben hat, welche Entwickler daran interessiert sind, und welche Plattformen dafür relevant

sind. Inserate können sich auf eine Anwendung beziehen (die dann dazu gespeichert werden soll), müssen es aber nicht.

Aufgabenstellung

1. Erstellen Sie für die oben beschriebene Datenbank ein Entity-Relationship-Diagramm. Benützen Sie dazu die **(min, max)-Notation** wie im Lehrbuch von Kemper/Eickler bzw. in der VU Datenmodellierung beschrieben. Andere Notationen werden nicht akzeptiert. Erstellen Sie das ER-Diagramm mit einem Grafikprogramm (z. B. Dia, Visio).
2. Leiten Sie aus dem ER-Diagramm die Relationen der Datenbank in 3. Normalform so ab, dass sie verbundtreu und abhängigkeitstreu sind. Halten Sie diese Relationen schriftlich fest und machen Sie dabei PRIMARY und FOREIGN KEYS eindeutig kenntlich. Folgende Notation wird empfohlen:

   RelX (attr1, attr2, attr3, attr4: *RelY.attrZ*)

   bedeutet, dass die Datenbank eine Relation RelX enthält. Diese Relation hat 4 Attribute attr1, attr2, attr3 und attr4. Der Primary Key dieser Relation besteht aus den Attributen attr1 und attr2. Das Attribut attr4 ist ein Foreign Key auf das Attribut attrZ in der Relation RelY.

   Denken Sie beim Übersetzen der "empfohlenen Version einer Anwendung" ins Relationenmodell daran, dass für eine gegebene Anwendung nur die Versionsnummer notwendig ist, um die empfohlene Version zu identifizieren, nicht aber noch ein zusätzlicher Anwendungsname, da sich die empfohlene Versionsnummer immer auf den gleichen Anwendungsnamen bezieht. Ein ER-Diagramm kann solche Sachverhalte nicht ausdrücken – das bedeutet also nicht, dass Ihre Modellierung schlecht ist. Übersetzen Sie hier daher nicht "mechanisch" vom ER-Diagramm ins Relationenmodell, sondern behalten Sie die Spezifikation im Hinterkopf.

   Das Entity-Relationship-Diagramm sowie das Relationenmodell sollen gemeinsam in der Datei `entwurf.pdf` gespeichert werden. Für die Erstellung des PDFs kann beispielsweise das Tool doc2pdf verwendet werden.
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:
   - Die Datenbank soll keine NULL-Werte enthalten.
   - Wählen Sie für Geldbeträge keine Gleitkommatypen sondern z. B. NUMERIC mit zwei Nachkommastellen. Für Versionsnummern können Sie der Einfachheit halber ganze Zahlen verwenden.
   - Realisieren Sie die fortlaufende Nummerierung der "id"-Attribute von Bewertung, Inserat und Download mit Hilfe von Sequences. Die Sequence für den Schlüssel der Inserate soll bei 100 beginnen und in Zehnerschritten erhöht werden (d. h. 100, 110, 120, ...).
   - 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.
4. Erstellen Sie eine weitere Datei `insert.sql`, welche die INSERT-Befehle für die Testdaten der in Punkt 3 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" Anwendungen, Plattformen, etc. wählen. Stattdessen können Sie ruhig "Anwendung 1", "Anwendung 2", "Plattform 1", "Plattform 2" etc. verwenden.
5. Erstellen Sie eine Datei `drop.sql`, welche die nötigen DROP-Befehle enthält, um alle in Punkt 3 erzeugten Datenbankobjekte wieder zu löschen. Das Schlüsselwort `CASCADE` darf dabei nicht verwendet werden.

Für die Abgabe bereitzustellen

- Bringen Sie bitte Ihren Studentenausweis zur Abgabe mit. Eine Abgabe ohne Ausweis ist nicht möglich.
- Stellen Sie sicher, dass die SQL-Befehle (laut Punkten 3-5) unter PostgreSQL 8.4 auf dem Server **bordo.dbai.tuwien.ac.at** fehlerlos laufen. Sie müssen in der Lage sein, die von Ihnen bereitgestellten SQL-Dateien im Rahmen des Abgabegesprächs ablaufen zu lassen.
- Stellen Sie in Ihrem Abgabeverzeichnis 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` und `drop.sql` erzeugt werden. Beachten Sie dazu bitte die Hinweise zur Benützung von **psql** am Ende dieses Dokuments.
- Zusammenfassung: In Summe sind also folgende **5 Dateien** zu erstellen:
  - `entwurf.pdf`
  - `create.sql`
  - `insert.sql`
  - `drop.sql`
  - `listing.txt`
- Die Beispiele müssen bis zum Abgabetermin am 6.11.2011 um Mitternacht auf unserem Server (bordo.dbai.tuwien.ac.at) im Unterverzeichnis `beispiel2` verfügbar sein (die Dateien werden automatisch abgesammelt und den Tutorinnen und Tutoren zur Verfügung gestellt). Es sind keine Ausdrucke erforderlich.
- Wir erwarten von Ihnen eigenständige Lösungen. Plagiate werden **nicht** akzeptiert und mit 0 Punkten bewertet.

Bewertung
Für das Übungsbeispiel 2 werden **maximal 10 Punkte** vergeben. Im Rahmen des Abgabegesprächs wird nicht nur die Korrektheit der Lösungen, sondern auch (und vor allem) das **Verständnis** überprüft. Für die einzelnen Aufgaben erhalten Sie die maximal möglichen Punkte nur dann, wenn die Lösung richtig ist **und** wenn Sie in der Lage sind, diese Lösung auch entsprechend zu erklären.

Die Verteilung der Punkte erfolgt nach folgendem Schlüssel:

- Datei `entwurf.pdf`: max. 5 Punkte (max. 3 Punkte für das ER-Diagramm; max. 2 Punkte für das Relationenmodell)
- Datei `create.sql`: max. 2 Punkte
- Datei `insert.sql`: max. 2 Punkte
- Datei `drop.sql`: max. 1 Punkt
- Datei `listing.txt`: Vorhandensein für das Erreichen der vollen Punktezahl notwendig.

Hinweise zur Verwendung von psql
Folgende Befehle können für Ihre Arbeit mit der interaktiven SQL-Shell psql von PostgreSQL 8.4 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`