Datenbanken 1: Praktikum 4

Thorsten Peter, Michael Roth, Johann Schaible

Lernziele

- Implementieren von Stored Procedures mit PL/pgSQL
- Definieren von Triggeraktionen, Aufruf von Stored Procedures

Vorbemerkungen

Einreichen der Lösung Hiermit ist das Hochladen auf git gemeint, was bis zum Montag **vor** Ihrem eigentlichen Praktikumstermin zu erledigen ist.

Finale Abgabe Dies meint die **im** Praktikumstermin stattfindende Abgabe Ihres Praktikums beim jeweiligen Dozenten.

1 Stored Procedures

Implementieren Sie folgende Funktionen als Stored Procedures auf dem Datenbankserver und testen Sie diese.

Verwenden Sie, wenn nicht anders angegeben, die Sprache `PL/pgSQL`.

1. Eine Funktion `getFreeSeats`, welche eine Flugnummer und ein Datum übergeben bekommt, und die freien Plätze auf diesem Flug berechnet und zurück gibt.
2. Eine Funktion `maintenance`, welche als Parameter ein Flugzeugkennzeichen entgegen nimmt, und für dieses Flugzeug einen Wartungsvorgang mit Flugfreigabe für den heutigen Tag erstellt. „Heute" bedeutet das aktuelle Datum, mit dem die Funktion aufgerufen wurde.

   Beachten Sie hierbei, dass ihre Funktion mit den folgenden beiden Fällen umgehen können muss:

   - Es gibt keinen Eintrag in der Wartungstabelle, daher muss ein neuer Eintrag erstellt werden
   - Es existiert bereits ein Eintrag, unter Umständen ohne Flugfreigabe
3. Eine Funktion (oder View[1]), welche die gleiche Aufgabe erfüllt wie Aufgabe 1.12 aus dem letzten Praktikum (Ausgabe aller Passagiere incl. aller Flugdaten). Sie dürfen hierfür auch eine Stored Procedure mit SQL schreiben[2].

[1] Siehe `https://www.postgresql.org/docs/current/static/sql-createview.html`
[2] Siehe `https://www.postgresql.org/docs/current/static/xfunc-sql.html`

2 Entfernungsberechnung für Bonusmeilen

Es soll ein Bonusmeilen System eingeführt werden, bei denen Passagiere bei Buchung 10% der zu erwartenden Reisedistanz auf ein Konto gut geschrieben bekommen. Diese Bonusmeilen können für Flüge verwendet werden.

Beispiel:

- Ein Passagier fliegt den Flug FRA nach LAX (Entfernung 9321km) und bekommt dafür 932 Bonusmeilen gut geschrieben
- Der gleiche Passagier (mit 932 Bonusmeilen) bucht erneut einen Flug, diesmal von Frankfurt nach Paris (Entfernung 449km). Die Bonusmeilen reichen also aus, um diesen Flug zu bezahlen. Dem Passagier werden demnach 449km von seinen 932 Bonusmeilen abgezogen, der Flug allerdings ist kostenlos.
- Ob Sie Bonusmeilen mit Nachkommastellen erfassen bzw. ob Sie für einen kostenlosen Flug erneut Bonusmeilen vergeben, bleibt Ihnen überlassen.

Um dies umzusetzen, implementieren Sie folgendes:

1. Eine Funktion `getDistance`, welche zwei IATA-Codes als Parameter erhält und die Entfernung zwischen diesen beiden Flughäfen berechnet.
2. Das „automatische" Verarbeiten der Bonusmeilen wird in Aufgabe 3 als Trigger implementiert

Für die Entfernungsberechnung auf einer Kugeloberfläche **muss** die Haversine Formel[3] verwendet werden. Das ist notwendig, da Sie nicht die „direkte" Route von Frankfurt nach Peking nehmen können, da diese zu nah am Erdkern verläuft.
Diese Formel finden Sie in Moodle in der Datei `dist.c`, welche Sie als Inspiration für Ihren PL/pgSQL Code verwenden können.

3 Trigger

Sie haben bereits ein Attribut für Bonusmeilen in Passagier bzw. Kunde aus einem früheren Praktikum. Dies können und sollen Sie verwenden, um für jeden Kunden Bonusmeilen zu speichern.
Implementieren Sie die folgenden drei Triggerfunktionen und Trigger auf die Buchungstabelle. Verwenden Sie hierfür, wenn angebracht, die Stored Procedures aus Aufgabe 1 bzw. 2:

1. Einen `before insert` Trigger. Dieser soll überprüfen, ob auf dem gewünschten Flug überhaupt noch Platz ist, und gegebenenfalls den `insert` abweisen.
2. Einen `before insert` Trigger welcher überprüft, ob der Flug mit Bonusmeilen anstatt mit Geld bezahlt werden kann. Falls dies der Fall ist, werden dem Passagier die Bonusmeilen entsprechend abgezogen und der Preis der Buchung wird auf 0 € gesetzt.
3. Einen `after insert` Trigger der die Bonusmeilen für den gerade gebuchten Flug berechnet und dem Passagier gut schreibt.

Hinweis: Die Aufgabe verlangt zwei Trigger als `before insert` auf die gleiche Tabelle zu legen. In diesem Fall werden die Trigger in alphabetischer Reihenfolge ausgeführt.[4] Die Reihenfolge ist wichtig, da der Trigger aus 1 auf jeden Fall **vor** dem Trigger aus 2 ausgeführt werden soll.

Sie können allerdings auch Aufgabe 1 und 2 in einem einzigen Trigger realisieren, wenn Sie wollen.

[3] Siehe `https://en.wikipedia.org/wiki/Haversine_formula`
[4] SQL definiert hier keinen Standard, die alphabetische Reihenfolge bezieht sich nur auf postgresql, siehe `https://www.postgresql.org/docs/current/static/trigger-definition.html`

Einreichen der Lösungen

Laden Sie rechtzeitig, also am Montag **vor** den Praktikumstermin, folgende Dateien auf git **in einen Unterordner** 🗀 **P4** hoch:

- Kopieren Sie die Datei 🗀 `Readme.md` in das Verzeichnis 🗀 P4. Die Datei finden Sie auf Moodle.
- Schreiben Sie bzw. kopieren Sie Ihre Definitionen der Stored Procedures und Trigger in die entsprechenden Felder in der `Readme.md`.

Laden Sie zusätzlich die folgenden, ihrem aktuellen Arbeitsstand entsprechenden Dateien auf git **in den Unterordner** 🗀 **P4** hoch, sodass ihre Gesamtlösung nachvollziehbar wird:

- Ihr aktuelles, mit PowerDesigner erstellte ER-Diagramm als `.jpeg` oder `.png` Datei
- Ihr aktuelles, von PowerDesigner erstellte SQL Create Script
- Ihr aktuelles SQL Insert Script (mit allen bisher einzufügenden Daten exkl. Testdaten)

Abgabe der finalen Lösung im Praktikum

- Sie sind in der Lage, jede der oben genannten Stored Procedures auszuführen und im Detail zu erklären.
- Sie haben Beispiele vorbereitet und die Korrektheit der Ergebnisse überprüft.
- Beachten Sie ebenfalls, im Praktikum eine Verbindung zur Datenbank zu haben. Bauen Sie diese **nicht** erst auf, wenn der Betreuer zu Ihnen in den Breakout Room kommt!