Databases and Information Systems

7. Übungsblatt

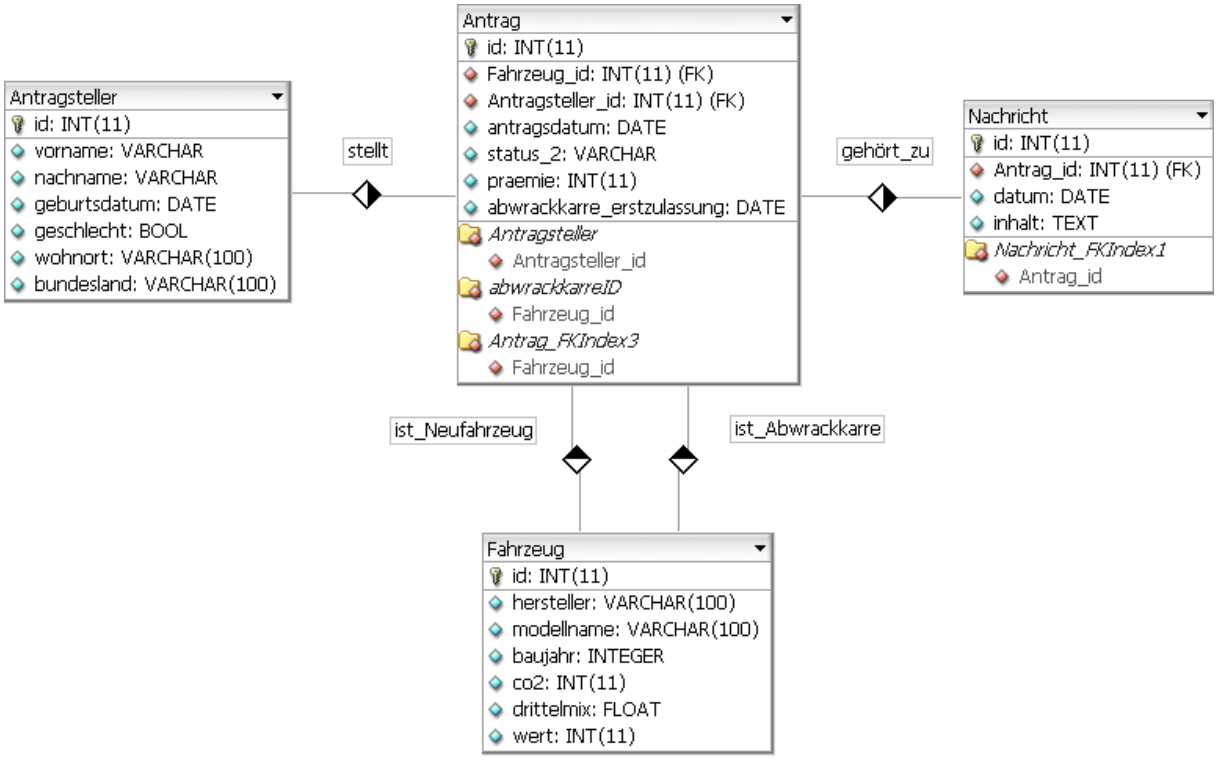
(Testatwoche: 1. - 3. Juni 2010)

Einführung in Datenbanksysteme
Datenbanken für die Bioinformatik

Heinz Schweppe, Katharina Hahn

Aufgabe 1 (Funktionen, Trigger) *Punkte*

Im Jahr 2009 wurde die Umweltprämie („Abwrackprämie“) im Rahmen des Konjunkturpakets II in Deutschland eingeführt. Käufer konnten bei Kauf eines Neuwagens und gleichzeitiger Verschrottung ihres alten PKWs die Prämie beantragen. Im Jahr nach der „Abwrackprämie“ soll nun Bilanz gezogen werden. Betrachten Sie dazu das folgende Schema für die „Abwrackdatenbank“.



SQL-Anweisungen zum Erzeugen des Schemas stehen Ihnen sowohl für Oracle als auch Postgres auf der Übungsseite zur Verfügung. Schreiben Sie folgende Funktionen in PL/SQL bzw. PL/pgSQL:

a) Die Funktion *CO2BILANZ(NUMBER konstante)* → *NUMBER* berechnet die CO_2-Bilanz der Umweltprämie. Dazu wird die Differenz des CO_2-Ausstoß zwischen allen durch die Umweltprämie geförderten Neufahrzeugen und allen „Abwrackkarren“ bestimmt. Für die frühzeitige Herstellung eines Neufahrzeuges wird die Konstante *konstante* als zusätzlicher CO_2-Ausstoß eingerechnet. Die Differenz unter Berücksichtigung der Konstante wird zurückgegeben.

```
--oracle
```

```
CREATE OR REPLACE
FUNCTION CO2BILANZ2 ( konstante IN NUMBER DEFAULT 30) RETURN NUMBER AS
co2 INTEGER := 0;
BEGIN
 SELECT SUM(neu.co2 - alt.co2 + konstante) INTO co2
  FROM antrag a JOIN fahrzeug neu ON (a.neufahrzeugid = neu.id)
  JOIN fahrzeug alt ON (a.abwrackkarreid = alt.id);

 RETURN co2;
END CO2BILANZ2;

-- pl/pgsql
CREATE OR REPLACE
FUNCTION CO2BILANZ2 (integer) RETURNS INTEGER AS '
DECLARE
 co2 INTEGER := 0;
BEGIN
 SELECT SUM(neu.co2 - alt.co2 + $1) INTO co2
  FROM antrag a JOIN fahrzeug neu ON (a.neufahrzeugid = neu.id)
  JOIN fahrzeug alt ON (a.abwrackkarreid = alt.id);

 RETURN co2;
END;
' LANGUAGE plpgsql;
```

b) Die Kosten (ggf. der Gewinn), die der Regierung durch die Gewährung der Umweltprämie entstanden sind, sollen durch die Funktion *KOSTEN(INTEGER steuer_vorteil_pauschal, FLOAT verbrauch_schwellwert)* → *VARCHAR* berechnet werden. Die Kosten für den Staat sind bei jedem gewährten Antrag die Prämie. Gewinne ergeben sich aus der erhaltenen Mehrwertsteuer beim Verkauf eines Neuwagens (19% des Wertes des Neuwagens). Liegt der Verbrauch (*drittelmix*) des Neuwagens unter dem angegebenen Schwellwert, müssen zusätzlich als Kosten noch KFZ-Steuerausfälle von *steuer_vorteil_pauschal* addiert werden. Sind dem Staat Kosten entstanden, so gibt die Funktion aus: „Durch die Gewährung der Prämie sind Kosten von ??? € entstanden." Ansonsten gibt die Funktion aus: „Der Staat hat durch die Abwrackprämie einen Gewinn von sage und schreibe ??? € gemacht."

```
CREATE OR REPLACE
FUNCTION KOSTEN ( kfz_steuer_pauschale IN NUMBER  verbrauchs_schwellwert IN NUMBER)
 RETURN VARCHAR2 AS
kosten NUMBER := 0;
steuergewinn NUMBER := 0;
BEGIN
  FOR neuwagen IN (SELECT praemie, wert, drittelmix FROM antrag a JOIN fahrzeug f ON
(a.neufahrzeugid = f.id))
 LOOP
  IF(neuwagen.drittelmix <= verbrauchs_schwellwert)
   THEN kosten := kosten + (neuwagen.praemie + kfz_steuer_pauschale);
  ELSE
   kosten := kosten + neuwagen.praemie;
  END IF;
  steuergewinn := steuergewinn + (neuwagen.wert * 0.19);
 END LOOP;

 IF(kosten - steuergewinn > 0) THEN RETURN
     'Durch die Gewaehrung der Praemie sind Kosten von ' || (kosten - steuergewinn) || ' â,¬
entstanden.';
```

```
  ELSE RETURN 'Der Staat hat durch die Abwrackpraemie einen Gewinn von sage und schreibe
' || (steuergewinn - kosten) || ' â,¬ gemacht.';
  END IF;
END KOSTEN;
```

Damit Antragsteller sich über den Status ihres Antrags informieren können, werden Nachrichten in die Tabelle Nachricht geschrieben. Schreiben Sie einen Trigger, der die folgende Aufgabe erfüllt:

c) Wenn ein Sachbearbeiter einen Antrag bearbeitet, ändert er aus Gründen der Nachvollziehbarkeit den *status* des Antrags. Ein Antrag kann folgende Status haben: „eingegangen“, „in Bearbeitung“, „abgelehnt“, „wird gewährt“, „überwiesen“. Schreiben Sie einen Trigger, der bei der Änderung des Status Antrags eine Nachricht in die Nachrichten Tabelle schreibt. Die Nachricht soll einen sinnvollen, natürlichsprachlichen Text ihrer Wahl enthalten, der den Status erläutert (z.B. „Ihr Antrag ist in Bearbeitung und wird in Kürze erledigt.“) .

```
--oracle
CREATE OR REPLACE TRIGGER NACHRICHTENTRIGGER
AFTER UPDATE ON ANTRAG
REFERENCING OLD AS alt NEW AS neu
FOR EACH ROW
BEGIN
 IF :alt.status != :neu.status THEN
   IF :neu.status = 'eingeganen' THEN
     INSERT INTO nachricht (id, datum, inhalt, antragid)
      VALUES (nachricht_sequence.nextval, current_date, 'Ihr Antrag ist eingegangen und wird
baldmoeglichst bearbeitet', :alt.id);
   ELSIF :neu.status = 'in Bearbeitung' THEN
     INSERT INTO nachricht (id, datum, inhalt, antragid)
     VALUES (nachricht_sequence.nextval, current_date, 'Ihr Antrag ist nun in Bearbeitung und
wird in Kuerze erledigt', :alt.id);
   ELSIF :neu.status = 'gewaehrt' THEN
     INSERT INTO nachricht (id, datum, inhalt, antragid)
       VALUES (nachricht_sequence.nextval, current_date, 'Ihr Antrag wurde gewaehrt. Die
Praemie wird in Kueze ueberwiesen.', :alt.id);
   ELSIF :neu.status = 'abgelehnt' THEN
     INSERT INTO nachricht (id, datum, inhalt, antragid)
      VALUES (nachricht_sequence.nextval, current_date, 'Ihr Antrag wurde leider abgelehnt.
Der Grund ist Willkuer.', :alt.id);
   ELSIF :neu.status = 'ueberwiesen' THEN
     INSERT INTO nachricht (id, datum, inhalt, antragid)
       VALUES (nachricht_sequence.nextval, current_date, 'Die Praemie wurde ueberwiesen,
bitte pruefen Sie ihren Kontostand.', :alt.id);
   END IF;
 END IF;
END;

-- pg/plsql
CREATE OR REPLACE FUNCTION FK_NACHRICHTENTRIGGER RETURNS TRIGGER AS $$
BEGIN
 IF OLD.status != NEW.status THEN
   IF NEW.status = 'eingeganen' THEN
     INSERT INTO nachricht (datum, inhalt, antragid)
      VALUES (current_date, 'Ihr Antrag ist eingegangen und wird baldmoeglichst bearbeitet',
OLD.id);
   ELSIF NEW.status = 'in Bearbeitung' THEN
```

```
    INSERT INTO nachricht (datum, inhalt, antragid)
      VALUES (current_date, 'Ihr Antrag ist nun in Bearbeitung und wird in Kuerze erledigt',
OLD.id);
  ELSIF NEW.status = 'gewaehrt' THEN
    INSERT INTO nachricht (datum, inhalt, antragid)
        VALUES (current_date, 'Ihr Antrag wurde gewaehrt. Die Praemie wird in Kueze
ueberwiesen.', OLD.id);
  ELSIF NEW.status = 'abgelehnt' THEN
    INSERT INTO nachricht (datum, inhalt, antragid)
      VALUES (current_date, 'Ihr Antrag wurde leider abgelehnt. Der Grund ist Willkuer.',
OLD.id);
  ELSIF NEW.status = 'ueberwiesen' THEN
    INSERT INTO nachricht (datum, inhalt, antragid)
        VALUES (current_date, 'Die Praemie wurde ueberwiesen, bitte pruefen Sie ihren
Kontostand.', OLD.id);
  END IF;
 END IF;

 RETURN NEW;
END;
$$
LANGUAGE 'plpgsql';

DROP TRIGGER NACHRICHTENTRIGGER ON ANTRAG;
CREATE TRIGGER NACHRICHTENTRIGGER
AFTER UPDATE ON ANTRAG
FOR EACH ROW
EXECUTE PROCEDURE FK_NACHRICHTENTRIGGER;
```

Aufgabe 2 (JDBC) *Punkte*

a) Was versteht man unter dem *Impedance Mismatch*?

> Als Impedance Mismatch bezeichnet man die Diskrepanz zwischen relationalen Datenanfragesprachen (SQL) und Applikationssprachen (z.B., Java). SQL ist relational, d.h., es operiert auf Mengen, Java hingegen ist Objekt-orientiert. Das Cursor-Konzept dient als Adapter zwischen diesen beiden.

Schreiben Sie ein Anwendungsprogramm, das über JDBC auf die *Mondial* Datenbank auf *esel* zugreift. Laden Sie sich dazu das Archiv *ue7.jar* von der Übungsseite runter. Vergessen Sie nicht, den JDBC-Treiber des von Ihnen gewählten Datenbanksystems dem Klassenpfad hinzuzufügen. Die Klasse *BaseDAO* enthält bereits Methoden, um eine Verbindung zu einer Datenbank herzustellen bzw. zu schließen. Wählen Sie die URL und den Klassennamen entsprechend des gewählten Datenbanksystems aus. Studieren Sie, wie diese zusammengesetzt sind. Schreiben Sie Ihre eigene(n) Klasse(n), die folgende Methoden enthält (enthalten). Verwenden Sie dabei *getConnection()* und *closeConnection()* der *BaseDAO*.

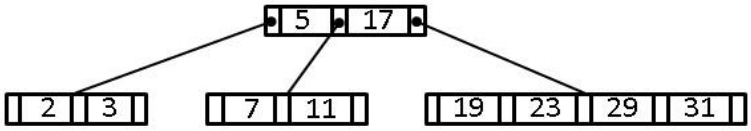
b) Implementieren Sie die Methode *void printAllNeighbors(String country)*, die den Namen eines Landes als Eingabe erhält und die Namen aller Nachbarn alphabetisch sortiert ausgibt.

c) Implementieren Sie die Methode *void listBigCities(String country, int threshold)*, die die Funktionen BIG_CITIES der Aufgabe 3 des Übungsblatts 6 aufruft. Geben Sie die von der Funktion zurückgegebenen Werte zeilenweise aus.

Informationen zur Verwendung der JDBC Schnittstelle zum Ausführen von SQL-Anfragen und Handhabung der Rückgabe finden Sie in den Dokumentationen der Datenbanksysteme, z.B. http://www.postgresql.org/files/documentation/books/pghandbuch/html/jdbc.html.

Aufgabe 3 (Indexstrukturen, B-Bäume) *Punkte*

a) Erstellen Sie einen B-Baum mit maximaler Füllgröße N=4. Fügen Sie nacheinander in den anfänglich leeren Baum Elemente mit den folgenden Schlüsseln in der angegebenen Reihenfolge ein: 2, 3, 5, 7, 11, 17, 19, 23, 29, 31



b) Ist der von Ihnen erstellte B-Baum der einzig mögliche für die enthaltenen Schlüsselwerte? Begründen Sie Ihre Antwort bzw. geben Sie ggf. ein Gegenbeispiel an.

Nein. Wenn die Elemente in unterschiedlicher Reihenfolge eingefügt werden, dann resultiert evtl. ein anderer Baum daraus.

c) Das Einfügen wie vieler Elemente in den Baum führt dazu, dass sich die Höhe um eins erhöht? Begründen Sie ihre Antwort, in dem Sie die Schlüssel der Elemente und den resultierenden Baum angeben. Geben Sie ein untere Schranke für die Anzahl der Elemente an.

B-Bäume sind vollständig balanciert. D.h., damit die Höhe des Baumes um eins wächst, muss der Baum „voll“ sein. Werden weitere 6 Elemente eingefügt, ist der Baum voll, beim Einfügen des 7. Elements wächst der Baum an der Wurzel.

d) Löschen Sie das Element mit dem Schlüsselwert 3 aus dem Baum, den sie in Teilaufgabe a) erstellt haben. Geben Sie den resultierenden Baum an.