5. Übungsblatt

Aufgabe 1 (20 Punkte) – Relationale Algebra und SQL

Verwendet das Datenbank-Schema des letzten Übungsblattes.

a.) Relationale Algebra
   1.) [1 Punkt] Liste alle Details zu den Publishern im State „WA“ auf.
   2.) [1 Punkt] Gebe die Städte aus, in denen Publisher sind.
   3.) [2 Punkte] Ermittele den Titel der Bücher, die von einem Publisher aus Berkeley verlegt und von einem Autor aus Berkeley geschrieben wurden
   4.) [1 Punkt] Ermittele die Städte in denen es Autoren aber keine Verleger gibt.
   5.) [1 Punkt] Ermittele Nachnamen, die sowohl bei Verlegern als auch Autoren vorkommen.
   6.) [2 Punkte] Ermittele die Namen der Autoren, die wenigstens ein Buch geschrieben haben, das von einem Publisher aus einer anderen Stadt als ihrer Heimatstadt verlegt wurde.
   7.) [2 Punkte] Ermittele den hypothetischen Verleger, der alle Bücher verlegt.

b.) SQL
   1.) [2 Punkte] An wievielen Büchern hat der jeweilige Autor mitgeschrieben? Tipp: ein unbekannter Autor hat an einem Buch mitgeschrieben.
   2.) [2 Punkte] Erweitere obige Anfrage dahingehend, dass die Ausgabe nach Nachnamen sortiert wird und nur diejenigen Autoren erscheinen, die mindestens an zwei Büchern mitgearbeitet haben.
   3.) [2 Punkte] Wieviele Bücher hat der durchschnittliche Publisher im Angebot und wie groß ist die Summe aller verlegten Bücher?
   4.) [2 Punkte] Gebe diejenigen Publisher aus, die mehr Bücher im Angebot haben als der durchschnittliche Publisher.
   5.) [2 Punkte] Ermittele diejenigen Autoren, die in Städten wohnen, in denen es Publisher gibt.

Aufgabe 2 (20 Punkte) – XML, XPath & XQuery

Es wird das folgende Dokument verwendet.

```
<?xml version="1.0"?>
<root>
  <flights>
    <flight from="Frankfurt" to="Rome" start="2008-09-07 09:30" arrival="2008-09-07
      10:30" price="70" airlineId="0815"/>
    <flight from="Frankfurt" to="Rome" start="2008-09-07 09:00" arrival="2008-09-07
      10:05" price="70" airlineId="4711"/>
    <flight from="Frankfurt" to="Helsinki" start="2008-09-07 09:50" arrival="2008-09-
      07 11:45" price="90" airlineId="4711"/>
    <flight from="Basel" to="Prag" start="2008-09-07 10:15" arrival="2008-09-07
      11:30" price="60" airlineId="0815" />
    <flight from="Prague" to="Rome" start="2008-09-07 20:15" arrival="2008-09-07
```

```
      21:30" price="160" airlineId="0815" />
    <flight from="Baden-Baden" to="Frankfurt" start="2008-09-07 08:30"
      arrival="2008-09-07 09:05" price="55" airlineId="0007"/>
  </flights>
  <airlines foo="bar">
    <airline id="0007">
      <name>BrianAir</name>
      <homeAirport>Dublin</homeAirport>
    </airline>
    <airline id="4711" >
      <name>Lufthansel</name>
      <homeAirport>Frankfurt</homeAirport>
    </airline>
    <airline id="0815" >
      <name>Einfach<!-- comment -->Jet</name>
      <homeAirport>London</homeAirport>
    </airline>
  </airlines>
</root>
```

a. [1 Punkt] Gibt es einen Unterschied zwischen „/“ und „//*“? Wenn ja, welchen?
b. [1 Punkt] Liste jeden Knoten des Dokuments auf.
c. [2 Punkte] Was liefert die folgende Anfrage zurück: „`root//airline/*[1]`“? Beschreibe das Ergebnis zusätzlich mit eigenen Worten.
d. [2 Punkte] Liste alle Flüge auf, die nach Rom(e) führen. Einmal als reinen XPath-Ausdruck und einmal mittels FLOWR.
e. [2 Punkte] Nun soll es von Frankfurt nach Rom gehen. Welche Direktflüge stehen zur Verfügung? Sortiere nach ansteigender Ankunftszeit.
f. [2 Punkte] Von Baden-Baden nach Rom mit XQuery. Aber Vorsicht: es gibt keine Direktflüge! Liste daher alle Verbindungen auf, die genau einen Zwischenstopp haben. Ignoriere die Flugzeiten.
g. [2 Punkte] Erweitere die letzte Anfrage derart, dass nun auch die Ankunftszeit beim Zwischenstopp vor der Abflugzeit dort liegt.
h. [2 Punkte] Welche ID hat die Airline „BrianAir“?
i. [2 Punkte] Ermittele den Wert des Attributes „foo“ des Elternknotens der Airline mit dem Namen „BrianAir“.
j. [2 Punkte] Gebe alle nachfolgenden Geschwisterknoten der Airline „Lufthansel“ aus. (Dies kann im SQL Server nicht [mit einfachen Mitteln] getestet werden, da die dafür nötige Achse fehlt.)
k. [2 Punkte] Gebe die Ziele aller Flüge aus, ersetze jedoch „Rome“ durch „Rom“

Hinweis1: XPath ist eine Teilmenge von XQuery. Bei XQuery wird die Möglichkeit, XPath-Pfadausdrücke auswerten zu können mit der sog. „FLOWR“-Logik (*for*, *let*, *order by*, *where*, *return*) zu einer vollwertigen Programmiersprache für XML-Anwendungen erweitert.

Hinweis2: Der Microsoft SQL Server verfügt ab der Version 2005 über die Fähigkeit zur Evaluation von XQuery-Ausdrücken. Es ist jedoch nur eine Teilmenge der XQuery-Funktionalität implementiert. Vor

allem fehlen die *doc()*-Funktion und *let*. Experimente mit *let* können jedoch in der aktuellen Vorabversion des SQL Server 2008 oder einer beliebigen anderen XQuery-Runtime durchgeführt werden. Einführungen in XPath und XQuery sind im Moodle verlinkt. Für die XML-Unterstützung im SQL Server siehe z.B. http://www.microsoft.com/sqlserver/2008/en/us/wp-sql-2008-whats-new-xml.aspx.

Hinweis3: wenn die folgende Fehlermeldung auftritt „`Meldung 2395, Ebene 16, Status 1, Zeile 97 XQuery [query()]: Die '{http://www.w3.org/2004/07/xpath-functions}:doc()'-Funktion ist nicht vorhanden.`“, einfach das „doc("….xml")“ löschen.

Hinweis4: Tritt eine Fehlermeldung mit der Bemerkung „`XQuery [query()]: 'xxxxx' erfordert ein Singleton (oder eine leere Sequenz). Ein Operand vom 'xdt:untypedAtomic *'-Typ wurde gefunden.`“ auf, einfach den XPath-Ausdruck mittels „[1]“ erweitern. Dies wählt das jeweils erste Element aus und erzeugt so einen Singleton.

Die Abgabe dieses Arbeitsblattes erfolgt durch Einwurf im Briefkasten #57 bis zum 17.06.2008 10:00 Uhr (bitte gelocht und getackert – für die Herren mit der unleserlichen Handschrift wird eine Abgabe in gedruckter Form empfohlen).

Viel Erfolg!