Database Systems – Practical Exam – September 2013

Problem 1. Database First (10 points)

In **toy store**, toys from different **manufacturers** are sold. For each **toy** certain information is kept in a catalogue: name (mandatory), type (can be any string – "puzzle", "plush", etc.), categories, manufacturer (mandatory), price in dollars (mandatory) and color. Each toy has **age range**. Each **category** has name (mandatory). Each **manufacturer** has name (unique) and country (both mandatory).

Design a database schema "**ToysStore**" in SQL Server to keep the catalog of **manufacturers**, **toys**, **age ranges** and **categories**. Ensure data integrity is fulfilled.

Provide a **SQL script for your database schema** (without the data), backup (.bak file) and a **simple documentation of the relationships** between the tables. Additionally, provide a **picture of the design diagram** describing your created database.

Problem 2. Sample Data (15 points)

Use C# to implement an application for generating random sample data in the toys store. Create at least 50 manufacturers, 100 categories, 100 age ranges and 20 000 toys.

* You are free to use an ORM framework by choice or plain ADO.NET.

Problem 3. SQL Queries (15 points)

Write three queries in native SQL:

1. Get all toys's name and price having type of "puzzle" and price above $10.00 ordering them by price
2. Get all manufacturers' name and how many toys they have in the age range of 5 to 10 years, inclusive
3. Get all toys' name, price and color from category "boys"

Provide three .sql files with the three queries in them.

Problem 4. Performance Tuning (5 points)

Find the bottlenecks in the above queries and improve your database for faster performance on searching.

Problem 5. Code First (15 points)

In a **bookstore**, **books** from different **authors** are sold. For each book certain information is kept in the bookstore's catalogue: author(s), title, ISBN number (unique 13-digits number, without hyphens, e.g. 9780470502259, non-mandatory), price and an official web site (if available). Some books could be **free of charge** and thus have no price. For each book a number of **reviews** could be written. Reviews have date of creation (mandatory) and could have an author (existing author from the database) or could be anonymous. The ISBN number is unique by nature. Author names should be unique in the system, i.e. two authors with the same names are considered the same. Each author could be a book author, review author or even both. The book titles could have duplicates.

Design a database schema "**Bookstore**" using Code First approach with Entity Framework.

Problem 6. Import from XML File (10 points)

Write a **C# program** to **parse an XML file** "**complex-books.xml**" holding a set of **books with authors and reviews** in the format given below and **insert them into the database**:

complex-books.xml

```xml
<?xml version="1.0" ?>
<catalog>
  <book>
    <title>SQL Server 2008 Query Performance Tuning Distilled (Expert's Voice in SQL
       Server)</title>
    <authors>
      <author>Sajal Dam</author>
      <author>Grant Fritchey</author>
    </authors>
    <web-site>http://www.amazon.com/SQL-Server-Performance-Tuning/dp/1430219025</web-site>
    <reviews>
      <review author="Sudheer Kumar" date="17-Jul-2013">
         I will suggest this book for DBAs and SQL Programmers who want to know the
         theories behind Index and Statistics, Blocking and Deadlocking + Fragmentation.
      </review>
      <review date="11-Aug-2003">Nice book about SQL Server.</review>
      <review author="Lefteris Paparakis" date="30-Jan-2013">
         Excellent material, a thorough analysis on MS SQL performance optimization. Book
         chapters are well structured with lots of examples. I enjoyed reading it a couple
         of times!
      </review>
    </reviews>
    <isbn>9781430219026</isbn>
    <price>35.03</price>
  </book>

  <book>
    <authors>
      <author>Ivan Vazov</author>
    </authors>
    <title>Under the Yoke</title>
    <isbn>9789549403015</isbn>
    <price>49.95</price>
  </book>

  <book>
    <web-site>http://www.w3schools.com/sql/</web-site>
    <title>SQL Tutorial</title>
    <reviews>
      <review>Easy to understand tutorial. Thank you W3 Schools!</review>
      <review date="22-Jan-2012">Low-quality, don’t read this</review>
    </reviews>
  </book>

  <book>
    <title>XML: A Deep Understanding</title>
    <authors>
      <author>John Shirrell</author>
    </authors>
    <web-site>http://www.xmlbook.info</web-site>
    <reviews>
      <review date="3-Nov-2011">Free e-book on XML and other XML standards</review>
      <review>Outdated. Written in 1999, useless today…</review>
      <review>Better learn XML at W3 Schools: http://www.w3schools.com/xml/</review>
    </reviews>
  </book>

</catalog>
```

You should **parse the XML** and throw an exception in case of required element is missing or the XML is incorrect. Note that in the **<books>** tag only the **title is mandatory** element, while ISBN, price, web site, authors and reviews are **optional**.

Reviews may have optionally author and date. The **date** always comes in the format "**d-MMM-yyyy**" (day + English month in 3 letters + year), e.g. "22-Jan-2013" and "3-Dec-1976". If the date is missing, assign the current date for the review.

The size of the XML file will be less than **10 MB**. The decimal separator in the price is "**.**" (dot).

* You are free to use an XML parser by choice (or to parse the XML without using a parser).

You should correctly **import the books, authors and reviews into the DB**. Keep in mind that the authors are unique. If some author already exists in the database, it should be reused (no duplicates are allowed).

* You are free to use an ORM framework by choice (in code first or database first approach) or plain ADO.NET.

Inserting a book should be atomic operation (it should be completed entirely and in case of a problem partial results should not be inserted in the database). For example: if we have 7 books in the XML and the first 5 books are correct but the 6th book has an error (e.g. too long web site URL), as a result of the import the first 5 books should be imported, and the last 2 books should be entirely skipped and it is incorrect to have partially imported pieces of the 6th book in the database.

Problem 7. Application Queries (15 points)

Implement a **C# program** for **searching for reviews by given period** or **by given author**. It should be able to process a sequence of queries from the XML file **reviews-queries.xml** in the following format:

reviews-queries.xml

```
<?xml version="1.0" ?>

<review-queries>

  <query type="by-period">
    <start-date>20-Jan-2012</start-date>
    <end-date>31-Dec-2013</end-date>
  </query>

  <query type="by-author">
    <author-name>Sudheer Kumar</author-name>
  </query>

  <query type="by-period">
    <start-date>11-May-2013</start-date>
    <end-date>20-Nov-2010</end-date>
  </query>

</review-queries>
```

The queries are of two types: by period and by author. The "type" attribute is **mandatory**.

When the search type is "**by-period**", find all reviews in the specified period (inclusively) and order them by date, then by content. The "start-date" and "end-date" elements are **mandatory** for this search type. The dates are specified in the format "**d-MMM-yyyy**" (the same like during the import).

When the search type is "**by-author**", find all reviews by the specified author and order them by date, then by content. The "author-name" element is **mandatory** for this search type and can occur exactly once.

Write the results in the XML file "**reviews-search-results.xml**" in the following format:

reviews-search-results.xml

```
<?xml version="1.0" ?>

<search-results>

  <result-set>
    <review>
      <date>22-Jan-2012</date>
      <content>Low-quality, don’t read this</content>
      <book>
        <title>SQL Tutorial</title>
        <url>http://www.w3schools.com/sql/</url>
      </book>
    </review>

    <review>
      <date>30-Jan-2013</date>
      <content>Excellent material, a thorough analysis on MS SQL performance optimization.
        Book chapters are well structured with lots of examples. I enjoyed reading it a
        couple of times!</content>
      <book>
        <title>SQL Server 2008 Query Performance Tuning Distilled (Expert's Voice in SQL
           Server)</title>
        <authors>Grant Fritchey, Sajal Dam</authors>
        <isbn>9781430219026</isbn>
        <url>http://www.amazon.com/SQL-Server-Performance-Tuning/dp/1430219025</url>
      </book>
    </review>

    <review>
      <date>17-Jul-2013</date>
      <content>I will suggest this book for DBAs and SQL Programmers who want to know the
        theories behind Index and Statistics, Blocking and Deadlocking +
        Fragmentation.</content>
      <book>
        <title>SQL Server 2008 Query Performance Tuning Distilled (Expert's Voice in SQL
           Server)</title>
        <authors>Grant Fritchey, Sajal Dam</authors>
        <isbn>9781430219026</isbn>
        <url>http://www.amazon.com/SQL-Server-Performance-Tuning/dp/1430219025</url>
      </book>
    </review>
  </result-set>

  <result-set>
    <review>
      <date>17-Jul-2013</date>
      <content>I will suggest this book for DBAs and SQL Programmers who want to know the
        theories behind Index and Statistics, Blocking and Deadlocking +
        Fragmentation.</content>
      <book>
        <title>SQL Server 2008 Query Performance Tuning Distilled (Expert's Voice in SQL
           Server)</title>
        <authors>Grant Fritchey, Sajal Dam</authors>
        <isbn>9781430219026</isbn>
        <url>http://www.amazon.com/SQL-Server-Performance-Tuning/dp/1430219025</url>
      </book>
    </review>
  </result-set>

  <result-set />

</search-results>
```

The output XML should contain a sequence of all matching **result-sets**. Each result set should contain a sequence of reviews.

Display for each **review** its date, its content and its book. Order the reviews by date, then by content.

Display for each **book** its title, authors, ISBN and URL exactly in this order. If some of these fields is missing, skip it. Display the **authors** as comma separated values, ordered alphabetically.

Implement the search functionality correctly. Ensure it works fast enough for millions of records in the DB, and SQL injection is not possible.

The size of the **input XML** file will be less than **1 MB**.
The size of the **output XML** file will be less than **1 GB**.

* You are free to use an ORM framework by choice or plain ADO.NET.

* You are free to use XML parsers by choice, but take into account the performance.

Problem 8. MongoDB (10 points)

Implement **logging for all search queries** from the previous task. For each search query keep a single record in MongoDB collection "Logs" holding its XML and the date of its execution. The collection should look like the following:

Logs		
Id	**Date**	**QueryXml**

1	25-Jul-2013 16:33:22	`<query type="by-period">`
		`  <start-date>20-Jan-2012</start-date>`
		`  <end-date>31-Dec-2013</end-date>`
		`</query>`

2	25-Jul-2013 16:33:23	`<query type="by-author">`
		`  <author-name>John Shirrell</author-name>`
		`</query>`

3	25-Jul-2013 16:33:23	`<query type="by-period">`
		`  <start-date>11-May-2013</start-date>`
		`  <end-date>20-Nov-2010</end-date>`
		`</query>`

…	…	…

* You are free to use a MongoDB driver by choice (the official is good enough).

Evaluation Criteria

The evaluation criteria are as follows:

- Correct and complete fulfillment of the requirements.
- Good technical design and appropriate use of technologies.
- High-quality programming code – correctness, readability, maintainability.
- Performance – highly-efficient code.

Other Terms

During the exam you are allowed to use any teaching materials, lectures, books, existing source code, and other paper or Internet resources.

Direct or indirect communication with anybody in class or outside is forbidden. This includes, but does not limit to, technical conversations with other students, using mobile phones, chat software (Skype, ICQ, etc.), email communication, posting in forums, folder synchronization software (like Dropbox), etc.

Exam Duration

Students are allowed to work up to 8 hours.