**Database Systems – Practical Exam – July 2013**

1. **Bookstore Database**

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. ISBN 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 both. The book titles could have duplicates.

Design a database schema "**Bookstore**" in SQL Server to keep the catalog of **authors**, **books** and **reviews**.

Provide a **SQL script for your database schema**. Script the schema only without any data in it.

10 score

1. **Database Tuning**

Ensure you have defined appropriate **constraints** to ensure data integrity. Ensure you have defined appropriate database **indexes** for fast "exact-match" searching by book title, author, and ISBN.

Provide a **SQL script for your database schema**. Script the schema only without any data in it. Ensure the **indexes and constraints are scripted**.

5 score (optional)

1. **Simple Books Import from XML File**

Write a **C# program** to **parse an XML file** "**simple-books.xml**" holding a set of books (without reviews) in the format given below and **insert these books into the database**:

|  |
| --- |
| **simple-books.xml** |
| <?xml version="1.0" ?>  <catalog>  <book>  <author>Ivan Vazov</author>  <title>Under the Yoke</title>  <isbn>9789549403015</isbn>  <price>49.95</price>  </book>  <book>  <title>Introduction to Programming with C#</title>  <isbn>9789544005276</isbn>  <author>Svetlin Nakov and Co.</author>  <price>10.00</price>  <web-site>http://www.introprogramming.info/intro-csharp-book/</web-site>  </book>  <book>  <author>Ivan Vazov</author>  <title>Selected Stories</title>  <price>22.75</price>  </book>  <book>  <title>SQL Tutorial</title>  <web-site>http://www.w3schools.com/sql/</web-site>  <author>W3Schools</author>  </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 XML file the ISBN, price and web site are optional while **author and title are obligatory**. In the XML file each book will have exactly one author.

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).

5 score

You should correctly **import the books and authors 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).

The ISBN is also unique but in case of duplicated ISBN, your program should throw an exception.

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

10 score

1. **Complex Books Import from XML File**

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 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).

5 score (optional)

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.

7 score (optional)

**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.

3 score (optional)

1. **Simple Search for Books**

Write a **C# program** for **searching for a book** by title, author or ISBN or any combination of them given as XML document in the following format:

|  |
| --- |
| **simple-query.xml** |
| <?xml version="1.0" ?>  <query>  <title>Under the Yoke</title>  <author>Ivan Vazov</author>  <isbn>9789549403015</isbn>  </query> |

Another example:

|  |
| --- |
| **simple-query.xml** |
| <?xml version="1.0" ?>  <query>  <author>Ivan Vazov</author>  </query> |

Note that **any of the search criteria could be missing** (even all of them). The searching performs **case-insensitive exact-matching**. This means that "IvaN vaZoV" will match "Ivan Vazov" while just "Ivan" or just "Vazov" will not match it. The searching uses **"and" semantics** (all search criteria apply in the same time).

The result should be printed at the console as a list of book titles in **alphabetical order**, along with the review count for each book in the following format:

|  |
| --- |
| 2 books found:  Selected Stories --> 2 reviews  Under the Yoke --> no reviews |

In case of nothing found, print "**Nothing found**" at the console:

|  |
| --- |
| Nothing found |

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

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

10 score

1. **Search for Reviews**

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 (in code first or database first approach) or plain ADO.NET.

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

35 score (optional)

1. **Search Logs (Code First)**

Using **"code first" approach** in your ORM framework implement **logging for all search queries** from the previous task. For each search query keep a single record in the table "**SearchLog**" holding its XML and the date of its execution. The table should look like the following:

|  |  |  |
| --- | --- | --- |
| **SearchLogs** | | |
| **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 an ORM framework by choice.

\* You may use a separate database called "**Logs**" to store the search logs or the same database (by choice).

\* Using the "database-first" approach would give maximum of 50% of the score for this problem.

10 score (optional)

**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.

To pass the exam you need 60 score (of 100 scores total).

**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.