I590 SQL AND NOSQL

Fall Semester 2019

MIDTERM EXAM

1) Multiple choice questions:

i) NoSQL databases are used mainly for handling large volumes of ______________ data.
a) Unstructured
b) Structured
c) Semi-structured
d) All of the mentioned

Answer: (a) Unstructured

ii) The ___________ operation, denoted by −, allows us to find tuples that are in one relation but are not in another.
a) Union
b) Set-difference
c) Difference
d) Intersection

Answer: (b) Set-difference

iii) In which of the following can many entity instances of one type be related to many entity instances of another type?
a) One-to-One Relationship
b) One-to-Many Relationship
c) Many-to-Many Relationship
d) Composite Relationship

Answer: (c) Many-to-Many Relationship

iv) Relational Algebra does not have
a) Selection operator
b) Projection operator
c) Aggregation operators
d) Division operator

Answer: (c) Aggregation operators

v) Normal form which only includes indivisible values or single atomic values is classified as
a) Third normal form
b) First normal form
c) Second normal form
d) Fourth normal form

Answer: (b) First normal form

vi) Which of the SQL statements is correct?
a) SELECT Username AND Password FROM Users
b) SELECT Username, Password FROM Users
c) SELECT Username, Password WHERE Username = 'user1'
d) None of these

Answer: (b) SELECT Username, Password FROM Users

vii) A UNION query is which of the following?
a) Combines the output from no more than two queries and must include the same number of columns.
b) Combines the output from no more than two queries and does not include the same number of columns.
c) Combines the output from multiple queries and must include the same number of columns.
d) Combines the output from multiple queries and does not include the same number of columns.

Answer: (c) Combines the output from multiple queries and must include the same number of columns.

viii) Disadvantages of DTD are

(i) DTDs are not extensible
(ii)DTDs are not in to support for namespaces
(iii)There is no provision for inheritance from one DTDs to another

a) (i) is correct
b) (i),(ii) are correct
c) (ii),(iii) are correct
d) (i),(ii),(iii) are correct

Answer: (d) (i),(ii),(iii) are correct

ix) Which of the following XML documents are well-formed?

a) <firstElement>some text goes here
<secondElement>another text goes here</secondElement>
</firstElement>

b) <firstElement>some text goes here</firstElement>
<secondElement> another text goes here</secondElement>

c) <firstElement>some text goes here
<secondElement> another text goes here</firstElement>
</secondElement>

d) </firstElement>some text goes here
</secondElement>another text goes here
<firstElement>

**Answer: (a) <firstElement>some text goes here
<secondElement>another text goes here</secondElement>
</firstElement>**

x) Why do we use exist method in Xquery?
a) To determine if the XML data contains a certain node
b) To examine the XML and return back a scalar value
c) To Shred the XML nodes of the XML data into relational columns
d) To search inside xml data types

Answer: (a) To determine if the XML data contains a certain node

2) Consider the following two tables:

Table Name: Employee
Attributes: Employee_id, First_name, Last_name, Salary, Joining_date, Department
Table Name: Incentives
Attributes: Employee_id, Incentive_date, Incentive_amount

Write SQLs for the following scenarios:

a) Get First_Name from employee table in upper case
Answer: SELECT TO_UPPER(First_name) FROM Employee;

b) Get unique DEPARTMENT from employee table
Answer: SELECT DISTINCT(Department) FROM Employee;

c) Select first 3 characters of FIRST_NAME from EMPLOYEE
Answer: SELECT SUBSTRING(First_Name,1,3) FROM Employee;

d) Get length of FIRST_NAME from employee table

Answer: SELECT LENGTH(First_Name) AS First_Name FROM Employee;

e) Get FIRST_NAME, Joining year, Joining Month and Joining Date from employee table
Answer: SELECT First_name, YEAR(DATE(Joining_Date)) AS Joining_Year, MONTH(DATE(Joining_Date)) AS Joining_Month, Joining_date FROM Employee;

f) Get all employee details from the employee table order by First_Name Ascending and Salary descending
Answer: SELECT * FROM Employee ORDER BY First_Name ASC, Salary DESC;

g) Get employee details from employee table whose employee name are not "John" and "Roy"
Answer: SELECT * FROM Employee
WHERE First_Name NOT IN ("John","Roy");

h) Get employee details from employee table whose Salary between 500000 and 800000
Answer: SELECT * FROM Employee WHERE Salary BETWEEN 500000 AND 800000;

i) Get employee details from employee table whose joining month is "January"
Answer: SELECT * FROM Employee WHERE MONTH(DATE(Joining_Date)) = "January"

j) Get department, total salary with respect to a department from employee table order by total salary descending
Answer: SELECT Department, SUM(Salary) AS Total_Salary FROM Employee
Order by Salary DESC

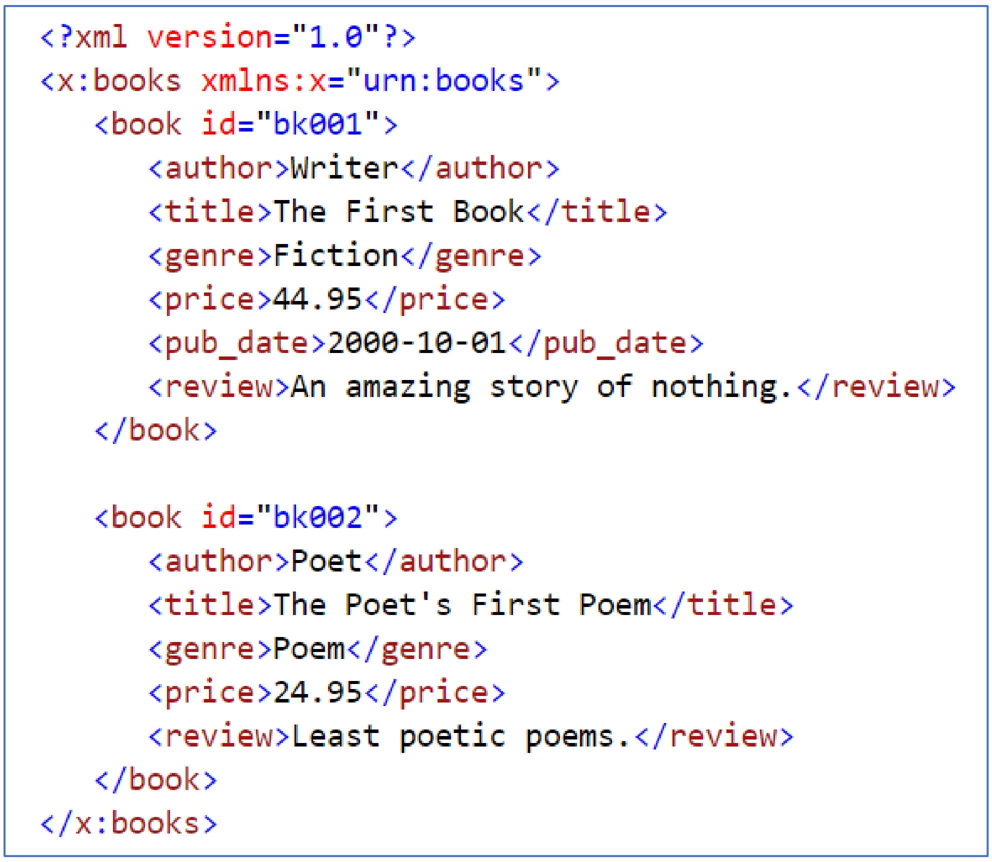
3) Write the DTD for the following xml file:

```xml
<?xml version="1.0"?>
<!DOCTYPE DatabaseInventory SYSTEM "DatabaseInventory.dtd">

<DatabaseInventory>

  <DatabaseName>
    <GlobalDatabaseName>production.iDevelopment.info</GlobalDatabaseName>
    <OracleSID>production</OracleSID>
    <DatabaseDomain>iDevelopment.info</DatabaseDomain>
    <Administrator EmailAlias="jhunter" Extension="6007">Jeffrey Hunter</Administrator>
    <DatabaseAttributes Type="Production" Version="9i"/>
    <Comments>
      The following database should be considered the most stable for
      up-to-date data. The backup strategy includes running the database
      in Archive Log Mode and performing nightly backups. All new accounts
      need to be approved by the DBA Group before being created.
    </Comments>
  </DatabaseName>

  <DatabaseName>
    <GlobalDatabaseName>development.iDevelopment.info</GlobalDatabaseName>
    <OracleSID>development</OracleSID>
    <DatabaseDomain>iDevelopment.info</DatabaseDomain>
    <Administrator EmailAlias="jhunter" Extension="6007">Jeffrey Hunter</Administrator>
    <Administrator EmailAlias="mhunter" Extension="6008">Melody Hunter</Administrator>
    <DatabaseAttributes Type="Development" Version="9i"/>
    <Comments>
      The following database should contain all hosted applications. Production
      data will be exported on a weekly basis to ensure all development environments
      have stable and current data.
    </Comments>
  </DatabaseName>

  <DatabaseName>
    <GlobalDatabaseName>testing.iDevelopment.info</GlobalDatabaseName>
    <OracleSID>testing</OracleSID>
    <DatabaseDomain>iDevelopment.info</DatabaseDomain>
    <Administrator EmailAlias="jhunter" Extension="6007">Jeffrey Hunter</Administrator>
    <Administrator EmailAlias="mhunter" Extension="6008">Melody Hunter</Administrator>
    <Administrator EmailAlias="ahunter">Alex Hunter</Administrator>
    <DatabaseAttributes Type="Testing" Version="9i"/>
    <Comments>
      The following database will host more than half of the testing
      for our hosting environment.
    </Comments>
  </DatabaseName>

</DatabaseInventory>
```

Answer:

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT DatabaseInventory (DatabaseName+)>
<!ELEMENT DatabaseName (   GlobalDatabaseName
                         , OracleSID
                         , DatabaseDomain
                         , Administrator+
                         , DatabaseAttributes
                         , Comments)
>
<!ELEMENT GlobalDatabaseName (#PCDATA)>
<!ELEMENT OracleSID          (#PCDATA)>
<!ELEMENT DatabaseDomain     (#PCDATA)>
<!ELEMENT Administrator      (#PCDATA)>
<!ELEMENT DatabaseAttributes EMPTY>
<!ELEMENT Comments           (#PCDATA)>

<!ATTLIST Administrator       EmailAlias CDATA #REQUIRED>
<!ATTLIST Administrator       Extension  CDATA #IMPLIED>
<!ATTLIST DatabaseAttributes  Type       (Production|Development|Testing)
#REQUIRED>
<!ATTLIST DatabaseAttributes  Version    (7|8|8i|9i) "9i">

<!ENTITY AUTHOR "Jeffrey Hunter">
<!ENTITY WEB    "www.iDevelopment.info">
<!ENTITY EMAIL  "jhunter@iDevelopment.info">
```

4) Write XML schema for the following XML file:

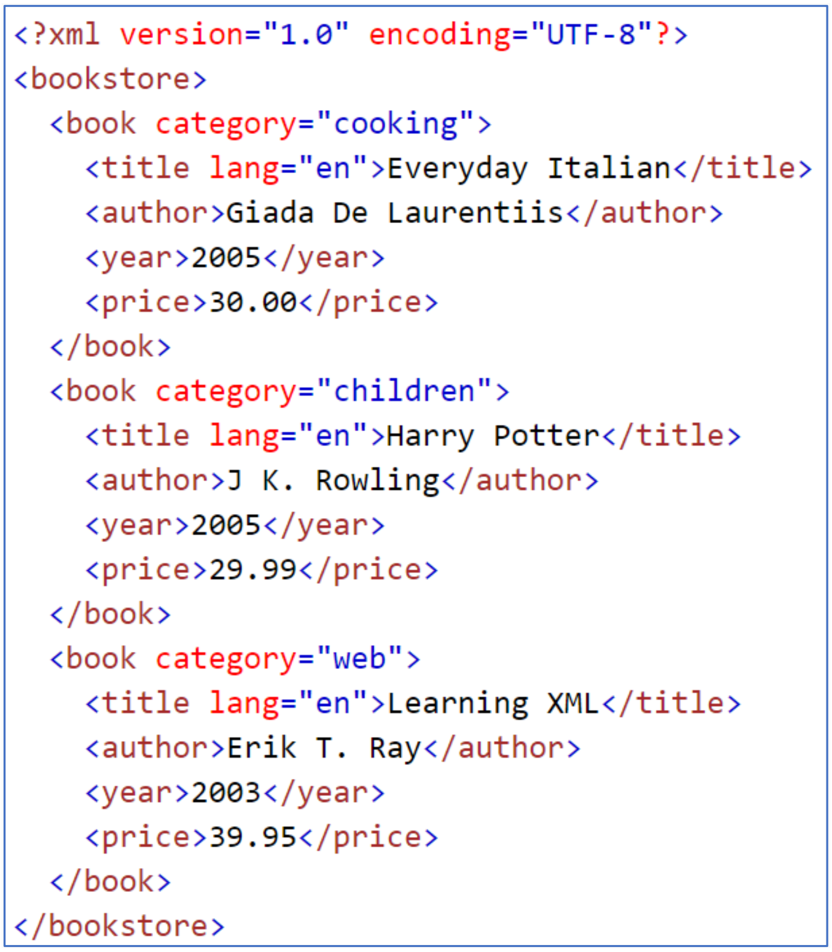


```xml
<?xml version="1.0"?>
<x:books xmlns:x="urn:books">
   <book id="bk001">
      <author>Writer</author>
      <title>The First Book</title>
      <genre>Fiction</genre>
      <price>44.95</price>
      <pub_date>2000-10-01</pub_date>
      <review>An amazing story of nothing.</review>
   </book>

   <book id="bk002">
      <author>Poet</author>
      <title>The Poet's First Poem</title>
      <genre>Poem</genre>
      <price>24.95</price>
      <review>Least poetic poems.</review>
   </book>
</x:books>
```

Answer:

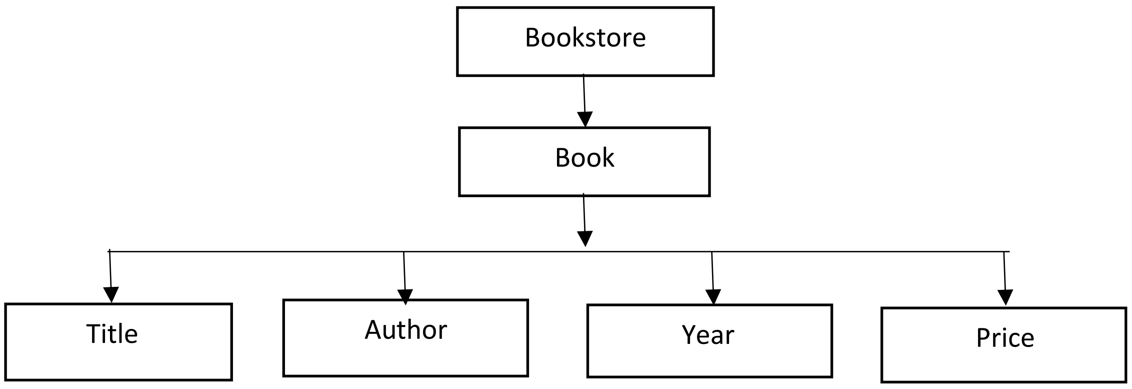
```xml
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
           targetNamespace="urn:books"
           xmlns:bks="urn:books">

 <xsd:element name="books" type="bks:BooksForm"/>

 <xsd:complexType name="BooksForm">
  <xsd:sequence>
   <xsd:element name="book"
                type="bks:BookForm"
                minOccurs="0"
                maxOccurs="unbounded"/>
   </xsd:sequence>
 </xsd:complexType>
 <xsd:complexType name="BookForm">
   <xsd:sequence>
     <xsd:element name="author"   type="xsd:string"/>
     <xsd:element name="title"    type="xsd:string"/>
     <xsd:element name="genre"    type="xsd:string"/>
     <xsd:element name="price"    type="xsd:float" />
     <xsd:element name="pub_date" type="xsd:date" />
     <xsd:element name="review"   type="xsd:string"/>
   </xsd:sequence>
   <xsd:attribute name="id"   type="xsd:string"/>
 </xsd:complexType>
</xsd:schema>
```

5) Write XML tree for the following XML file:



```xml
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
  <book category="cooking">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  <book category="children">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="web">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```

Answer:



6) For the xml below, answer the questions:

```
<?xml version="1.0" encoding="UTF-8"?>
 <bookstore>
 <book category="cooking">
 <title lang="en">Everyday Italian</title>
 <author>Giada De Laurentiis</author>
<year>2005</year>
 <price>30.00</price>
 </book>
 <book category="children">
 <title lang="en">Harry Potter</title>
 <author>J K. Rowling</author>
<year>2005</year>
<price>29.99</price>
 </book>
 <book category="web">
 <title lang="en">XQuery Kick Start</title>
 <author>James McGovern</author>
 <author>Per Bothner</author>
<author>Kurt Cagle</author>
 <author>James Linn</author>
 <author>Vaidyanathan Nagarajan</author>
< year>2003</year>
<price>49.99</price>
</book>
<book category="web">
<title lang="en">Learning XML</title>
<author>Erik T. Ray</author>
<year>2003</year>
<price>39.95</price>
</book>
</bookstore>
```

Write XPaths for the following scenarios:

a) Select the first book element that is the child of the bookstore element

Answer: //book[substring-before(./year/text(), '5')][./price =30.00]/title

b) Selects the last but one book element that is the child of the bookstore element
Answer: //book[contains(./title, 'XQuery')]

c) Select the first two book elements that are children of the bookstore element
Answer: //book[substring-before(./year/text(), '5')]

d) Select all the title elements that have a "lang" attribute with a value of "en"
Answer: contains(/bookstore/book/title[@lang = 'en']/title)

e) Select all the title elements of the book elements of the bookstore element that have a price element with a value greater than 35.00
Answer: **//book[substring-before(./year/text(), '5')][./price >=35.00]/title**

7) General SQL and NoSQL questions:

a) What is the difference between JOIN and UNION?

Answer: The primary difference between JOIN and UNION is that JOIN combines the tuples from two relations and the resultant tuples include attributes from both the relations. On the other hand, the UNION combines the result of two SELECT queries.

The JOIN clause is applicable only when the two relations involved have at least one attribute common in both. On the other hands, the UNION is applicable when the two relations have the same number of attribute and the domains of corresponding attributes are same.

There are four types of JOIN INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL OUTER JOIN. But there are two types of UNION, UNION and UNION ALL.

In JOIN, the resultant tuple has the larger size as it includes attributes from both the relation. On the other hands, in UNION the number of tuples are increased as a result include the tuple from both the relations present in the query.

b) What are aggregate and scalar functions? Give some examples

Answer: A **scalar function**is a function that operates on scalar values -- that is, it takes one (or more) input values as arguments directly and returns a value.

Examples:

- UCASE() - Converts a field to upper case
- LCASE() - Converts a field to lower case
- MID() - Extract characters from a text field
- LEN() - Returns the length of a text field
- ROUND() - Rounds a numeric field to the number of decimals specified
- NOW() - Returns the current system date and time

- FORMAT() - Formats how a field is to be displayed

An **aggregate function i**s a function that operates on aggregate data -- that is, it takes a complete *set* of data as input and returns a value that is computed from all the values in the set.

Examples:

- AVG() - Returns the average value
- COUNT() - Returns the number of rows
- FIRST() - Returns the first value
- LAST() - Returns the last value
- MAX() - Returns the largest value
- MIN() - Returns the smallest value
- SUM() - Returns the sum

c) What is the difference between NoSQL & Mysql DBs'?

The primary Comparison between MySQL vs NoSQL are discussed below:

The Basis Of Comparison	**MySQL**	**NoSQL**
Nature	A relational database in nature	A non-relational database in nature
Design	Modeled based on the concept of "table"	Modeled based on the concept of "document"
Scalable	Being relational in nature can be a tough task to scale big data	Easily scalable big data as compared to relational

Model	Detailed database model needs to be in place before the creation	No need to develop a detailed database model
Community	A vast and expert community is available	A community is growing rapidly and smaller as compared to MySQL
Standardization	SQL is standard language	Lack of a standard query language
Schema	Schema is rigid	Dynamic schema is a key benefit of NoSQL
Flexibility	Not so flexible design-wise, new column or field insertion affects a design	New column or fields can be inserted without existing design

d) When should a NoSQL database be used instead of a relational database?

Answer: NoSQL database be used instead of a relational database in the following scenarios:

Flexible Data Requirements

One of NoSQL's greatest strengths is its ability to handle flexible data storage. Traditional RDMS relies on static data structure—in fact, best practices stipulate the establishment of a database schema before any coding even begins. The reason is obvious: changing the database structure later is often a time-consuming and expensive affair.

In contrast, NoSQL's ability to handle an ever-changing data model make it ideal for molecular modeling, engineering parts, geo-spatial data and similar applications, all of which revolve around data that is constantly changing and evolving.

Similarly, NoSQL is the obvious choice for scenarios where the data's structure may be more static, but not fully known at the outset of the project.

Rapid Data Growth and Scalability

Another area where NoSQL shines is its ability to handle volatile data growth. Scenarios involving the launch of a new app or service, where a company's customer base can suddenly skyrocket into the millions, are perfect examples of the need for NoSQL.

A traditional RDMS is designed to scale vertically. This means that when a database begins to experience performance issues, the common solution is to migrate it to a larger hard drive or faster server. In the modern era of Big Data, however, the rapid growth of the database often exceeds the speed at which such a painstaking migration process can occur.

Because of NoSQL's decentralized nature, however, a NoSQL database is far more adept at scaling horizontally, distributed across multiple hosts rather than a single monolithic server. In addition, because NoSQL databases are designed to be distributed, it lowers the performance load—and therefore the performance requirements—of the individual pieces of hardware that are used.

The decentralized nature of NoSQL also makes it a far better option for organizations who need or want to migrate their database to the cloud.

These factors make it much faster, and often much cheaper, to scale out a NoSQL database to keep up with rapid growth.

Concurrent Performance

A third, closely related, scenario where a NoSQL database may offer significant advantages is when serving a high volume of concurrent users.

Many traditional databases use a locking mechanism to ensure data integrity. Before beginning a transaction, the RDMS marks the data so that no other process can modify it until the transaction either succeeds or fails. Unfortunately, though, in a large database serving tens of thousands of users, such a mechanism quickly impacts the performance of the database.

NoSQL, in contrast, makes a trade-off between consistency and performance, eschewing the use of locked transactions. This makes it a much better choice for situations where a large number of users need simultaneous access.