Free Questions for I10-003 by dumpssheet

Shared by Barber on 24-05-2024

For More Free Questions and Preparation Resources

Check the Links on Last Page

Question 1

Question Type: MultipleChoice

See separate window.

[example.xml]

```
<example>
  <data>
    <userid>id1</userid>
    <password>pass1</password>
    <name>name1</name>
    <address>add1</address>
  </data>
  <data>
    <userid>id2</userid>
    <password>pass2</password>
    <name>name2</name>
    <address>add2</address>
  </data>
</example>
```

[Execution Result]

```
<result>
  <data>
    <userid>id1</userid>
    <password>pass1</password>
    <name>name1</name>
    <address>add1</address>
  </data>
  <data>
    <userid>id2</userid>
    <password>pass2</password>
    <name>name2</name>
    <address>add2</address>
  </data>
</result>
```

A certain Web application displays user information according to user input via Web browser. The XML data managing user information is as shown in [example.xml] (separate window). The following [XQuery] is executed when the Web application retrieves user information from [example xml].

[XQuery]

{

fn:doc("example.xml")//data[userid = "(1)"][password = "(2)"]

}

At this time, the Web application completes the [XQuery] by replacing (1) and (2) with the user input character string, and executes the query.

No character escapes (e.g. convert "

Options:

A- (1) ' or''='
(2) OK

B- (1) ' or''='
(2) ' or''='

C- (1) idorfn:true()
(2) OK

D- (1) idorfn.true()

(2) idorfn:true()

E- (1) ' or fn:true() or any='
(2) OK

F- (1) 'or fn:true() or any='
(2) 'or fn:true() or any='

Answer:

B, F

Question 2

Question Type: **MultipleChoice**

Select the correct result of executing the following [XQuery] on [example xml] referenced in a separate window.

[XQuery]

{fn:doc("example.xml")/example/record/data[@condition="good"][2]}

{fn:doc("example.xml")/example/record/data[2][@condition="good"]}

[example.xml]

```
<example>
  <record date="2007-05-15">
    <data condition="bad">50</data>
    <data condition="bad">80</data>
    <data condition="good">250</data>
  </record>
  <record date="2007-05-16">
    <data condition="bad">60</data>
    <data condition="good">90</data>
    <data condition="good">150</data>
  </record>
</example>
```

Options:

A- <result>
<data condition='good'>90</data>
<data condition='good'>90</data>
</result>

B- <result>
<data condition='good'> 150</data>
<data condition='good'> 150</data>
</result>

C- <result>

<data condition='good'>90</data>
<data condition='good'> 150</data>
</result>

D- <result>
<data condition='good'>150</data>
<data condition='good'>90</data>
</result>

Answer:

D

Question 3

Question Type: **MultipleChoice**

Assume that a certain XMLDB can perform a validation check using DTD when inserting an XML document. Select two of the following that are unsuitable when using the xml:id attribute to manage the uniqueness of XML document elements. Assume that the XMLDB can properly process the xml:id attribute, and that an error is reported when a violation of the xml:id specification occurs.

Options:

A- The xml:id attribute can be defined as the ID type using DTD

B- Duplication errors in the values of the xml:id attribute cannot be detected unless a validation check is performed

C- Duplications may occur in the values of the xml:id attribute when combining two XML documents (neither having errors related to xml:id attributes) into one XML document

D- Duplication errors in the values of the xml:id attribute cannot be detected when the XMLDB does not support XML namespaces

Answer:

B, D

Question 4

Question Type: MultipleChoice

Assume that a certain XMLDB can be configured to [Simple Validation Mode] when inserting an XML document.

[Simple Validation Mode] checks for (1) through (4) below, and will not insert any non-conforming XML document into the XMLDB.

[Simple Validation Mode] does not check for anything other than (1) through (4) below.

[Simple Validation Mode]

(1) The XML document is a well-formed XML document

(2) No elements or attributes not declared in the XML Schema are present in the XML document

(3) Elements within the XML document follow the number of occurrences (root element (document element) must occur once) defined in the XML Schema

(4) Attributes within the XML document follow the allowed occurrences defined in the XML Schema

Select the correct result of inserting the following [XML Document] via [Simple Validation Mode] when using the [XML Schema] referenced in a separate window.

[XML Document]

```
<example>
  <record date="2007-05-15">
    <data condition="*****">100</data>
    <data condition="*****">50</data>
  </record>
  <record date="2007-05-16">
    <data>60</data>
    <data>200</data>
    <data>300</data>
  </record>
</example>
```

[XML Schema]

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="example" type="exampleType"/>
 <xs:complexType name="exampleType">
  <xs:sequence>
   <xs:element ref="record" minOccurs="0"
maxOccurs="unbounded"/>
  </xs:sequence>
 </xs:complexType>

 <xs:element name="record" type="recordType"/>
 <xs:complexType name="recordType">
  <xs:sequence>
   <xs:element ref="data" minOccurs="0"
maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="date" type="xs:date" use="required"/>
 </xs:complexType>

 <xs:element name="data" type="dataType"/>
 <xs:complexType name="dataType">
  <xs:simpleContent>
   <xs:extension base="xs:int">
    <xs:attribute name="condition">
     <xs:simpleType>
      <xs:restriction base="xs:string">
       <xs:enumeration value="good"/>
       <xs:enumeration value="bad"/>
      </xs:restriction>
```

Options:

A- The XML document can be inserted

B- A violation of (2) prevents the XML document from being inserted

C- A violation of (3) prevents the XML document from being inserted

D- A violation of (4) prevents the XML document from being inserted

Answer:

A

Question 5

Question Type: **MultipleChoice**

Select the correct result of executing the following [XQuery] on [example xml] referenced in a separate window.

```
[XQuery]
<result>{
  let $current := fn:doc("example.xml")/example/record/name
  return
    ($current/preceding-sibling::*[fn:position() <= 2])
}</result>

[example.xml]
<example>
 <record>
  <dept>Sales Department</dept>
  <group>Group No1</group>
  <title>Group Leader</title>
  <name>John Smith</name>
 </record>
</example>
```

Options:

A- <result>
<dept>Sales Department</dept>
<group>Group No1</group>
</result>

B- <result>
<group>Group No1</group>
<title>Group Leader</title>

</result>

C- <result>
<title>Group Leader</title>
<group>Group No1</group>
</result>

D- <result>
<group>Group No1</group>
<dept>Sales Department</dept>
</result>

Answer:

B

Question 6

Question Type: **MultipleChoice**

A certain Web application displays user information according to user input via Web browser. The XML data managing user information is as shown in [example xm l] referenced in a separate window.

```
[example.xml]
<example>
 <data>
  <userid>id1</userid>
  <password>pass1</password>
  <name>name1</name>
  <address>add1</address>
 </data>
 <data>
  <userid>id2</userid>
  <password>pass2</password>
  <name>name2</name>
  <address>add2</address>
 </data>
</example>
```

The following [XQuery] is executed when the Web application retrieves user information from [example xml].

```
[XQuery]
<result>{
 fn:doc("example.xml")//data[userid = "(1)"][password = "(2)"]
}</result>
```

At this time, the Web application completes the [XQuery] by replacing (1) and (2) with the user input character string, and executes the query.

No character escapes (e.g. convert "

Select the query execution result when the user input character string is as follows:

(1) "]/fn:root(),()/a[a="

(2) OK

Options:

A- <result/>

B- <result>
<example>
<data>
<userid>id1</userid>
pass1
<name>name1</name>
add1

</data>
<data>
<userid>id2</userid>
pass2
<name>name2</name>
add2

</data>
</example>
</result>

C- <result>
<example>
<data>
<userid>id1</userid>
pass1
<name>name1</name>
add1

</data>
<data>
<userid>id2</userid>
pass2
<name>name2</name>
add2

</data>
</example>

</result>

D- an error occurs

Answer:

A

Question 7

Question Type: MultipleChoice

Assume that a certain XMLDB can store an XML document as a model based on DOM Level2, and can retrieve the stored XML data via XQuery. Also assume that you wish to store the following [XML Document] in the XMLDB, and treat the script element content as a CDATA section when retrieving the XML data from the XMLDB. Select the most appropriate description regarding the following [Procedure].

Assume that the XMLDB follows the DOM Level2 specification, as well as specifications related to XQuery.

```
[XML Document]
<example>
 <script><![CDATA[
  for(i=0; i<5; i++) {
    document.write(i + "<BR>");
  }
 ]]></script>
</example>
```

[Procedure]

(1) Store [XML Document] in XMLDB

(2) Retrieve the stored XML data via XQuery (at this stage the XML data has not been serialized as a character string)

(3) Serialize the XML data as a character string

Options:

A- Under Procedure (1), the script element content will no longer be a CDATA section; accordingly, the script element content must be designated as a CDATA section under Procedure (2)

B- Under Procedure (1), the script element content will no longer be a CDATA section; accordingly, the script element content must be designated as a CDATA section under Procedure (3)

C- Under Procedure (1), the script element content is a CDATA section; however, under Procedure (2), the script element content will no longer be a CDATA section. Accordingly, the script element content must be designated as a CDATA section under Procedure (3)

D- The script element content is a CDATA section through Procedure (1), (2) and (3)

Answer:

C

Question 8

Question Type: MultipleChoice

Select the correct result of executing the [XQuery] on [example xml] referenced in a separate window. Do not consider indents (ignorable white space such as line feeds, tabs, etc.) in [example.xml] or the execution results.

```
[example.xml]
<example>
 <record>
  <dept>Sales Department</dept>
  <group>Group No1</group>
  <title>Group Leader</title>
  <name>John Smith</name>
 </record>
 <record>
  <dept>Engineering Department</dept>
  <name>Harold Jones</name>
 </record>
</example>

[XQuery]
<result>{
 for $rec in doc("example.xml")//record
 let $temp := (for $n in $rec//node() return $n)
  return $temp except $temp[fn:name() = ("group", "title")]//node()
}</result>
```

Options:

A- <result>

<dept>Sales Department</dept>
<name>John Smith</name>
<dept>Engineering Department</dept>
<name>Harold Jones</name>
</result>

B- <result>
<dept>Sales Department</dept>Sales Department
<name>John Smith</name>John Smith
<dept>Engineering Department</dept>Engineering Department
<name>Harold Jones</name>Harold Jones
</result>

C- <result>
<dept>Sales Department</dept>Sales Department
<group/>
<title/>
<name>John Smith</name>John Smith
<dept>Engineering Department</dept>Engineering Department
<name>Harold Jones</name>Harold Jones
</result>

D- <result>
<dept>Sales Department</dept>Sales Department
<group>Group No1</group>
<title>Group Leader</title>
<name>John Smith</name>John Smith
<dept>Engineering Department</dept>Engineering Department

<name>Harold Jones</name>Harold Jones
</result>

Answer:

D

Question 9

Question Type: MultipleChoice

See separate window.

[example.xml]

```
<example>
  <record date="2007-05-15">
    <data condition="bad">50</data>
    <data condition="bad">80</data>
    <data condition="good">150</data>
  </record>
  <record date="2007-05-16">
    <data condition="bad">50</data>
    <data condition="good">90</data>
    <data condition="good">150</data>
  </record>
</example>
```

[Output Result]

```
<result>
  <data value="50">
    <date>2007-05-15</date>
    <date>2007-05-16</date>
  </data>
  <data value="80">
    <date>2007-05-15</date>
  </data>
  <data value="90">
    <date>2007-05-16</date>
  </data>
  <data value="150">
    <date>2007-05-15</date>
    <date>2007-05-16</date>
  </data>
</result>
```

Assume you wish to execute an XQuery on [example.xml] (separate window) to obtain [Output Result] (separate window). Select the correct XQuery to obtain [Output Result].

A.
```
<result>{
 let $example := fn:doc("example.xml")
 for $data in fn:distinct-values($example//data)
 order by fn:number($data)
  return <data value="{ $data }">
   <date>{ $example//record[data = $data]/@date }</date>
  </data>
}</result>
```

B.
```
<result>{
 let $example := fn:doc("example.xml")
 for $data in fn:distinct-values($example//data)
 order by fn:number($data)
  return <data value="{ $data }">{
   for $record in $example//record[data = $data]
    return <date>{ fn:string($record/@date) }</date>
  }</data>
}</result>
```

C.
```
<result>{
 let $example := fn:doc("example.xml")
 for $data in fn:distinct-values($example//data)
 order by fn:number($data)
  return
   for $record in $example//record[data = $data]
    return <data value="{ $data }">
     <date>{ fn:string($record/@date) }</date>
    </data>
}</result>
```

D.
```
<result>{
 let $example := fn:doc("example.xml")
 for $data in fn:distinct-values($example//data),
    $record in $example//record[data = $data]
 order by fn:number($data)
  return <data value="{ $data }">
   <date>{ fn:string($record/@date) }</date>
  </data>
}</result>
```

Options:

A- Option A

B- Option B

C- Option C

D- Option D

Answer:

B

Question 10

Question Type: MultipleChoice

Assume that a certain XMLDB can perform a validation check using DTD.

Further assume that this XMLDB can create an XDM from the post-validation XML Information Set (Infoset), store this XDM in the XMLDB, and retrieve the XDM from the XMLDB.

Consider using the following [example.dtd] to perform a validation check, and inserting the [XML Document] below into the XMLDB. A validation check is not performed when retrieving XML data from the XMLDB.

In this case, select two statements correctly explaining the XML data retrieved from the XMLDB.

```
[example.dtd]
<!ELEMENT example (record)*>
<!ELEMENT record (data)*>
<!ELEMENT data (#PCDATA)>
<!ATTLIST record condition CDATA "average">

[XML Document]
<!DOCTYPE example SYSTEM "example.dtd">
<example>
 <record>
  <data>100</data>
  <data>200</data>
 </record>
</example>
```

Options:

A- The XML data contains a document type declaration

B- The XML data does not contain a document type declaration

C- The record element does not contain an attribute

D- The record element contains a condition attribute

Answer:

B, D

Question 11

Question Type: MultipleChoice

Select which of the following is not a correct description regarding XQuery 1.0 or SOL standards.

Options:

A- XQuery 1.0 defines a method to output data within RDB (relational database) tables in XML format

B- XQuery 1.0 does not define a method for partially updating XML data

C- SQL2003 (ISO/EC 9075) defines a method to output data within RDB (relational database) tables in XML format

D- SQL2003 (ISO/EC 9075) does not define a method for partially updating XML data

Answer:

A

Question 12

Question Type: **MultipleChoice**

See separate windows.

[example.xml]

```
<example>
  <record date="2007-05-15">
    <data condition="bad">50</data>
    <data condition="bad">80</data>
    <data condition="good">150</data>
  </record>
  <record date="2007-05-16">
    <data condition="bad">50</data>
    <data condition="good">90</data>
    <data condition="good">150</data>
  </record>
</example>
```

[Output Result]

```
<result>
  <record date="2007-05-15">
    <data condition="bad">2</data>
    <data condition="good">1</data>
  </record>
  <record date="2007-05-16">
    <data condition="bad">1</data>
    <data condition="good">2</data>
  </record>
</result>
```

Assume you wish to execute an XQuery on [example.xml] (separate window) to obtain [Output Result] (separate window).

Select the correct XQuery to obtain [Output Result],

Each data element of [Output Result] shows the number of data elements for each [example xml] record element having an attribute value equivalent to the condition attribute value.

A.
```
<result>{
 let $example := fn:doc("example.xml")
 let $condition := fn:distinct-values($example//@condition)
 for $record in $example/example/record
  for $each in $condition
   return <record date="{ $record/@date }">
    <data condition="{ $each }">{
     fn:count($record/data[@condition eq $each])
    }</data>
   </record>
}</result>
```

B.
```
<result>{
 let $example := fn:doc("example.xml")
 let $condition := fn:distinct-values($example//@condition)
 for $record in $example/example/record
  return <record date="{ $record/@date }">{
   for $each in $condition
    return <data condition="{ $each }">{
     fn:count($record/data[@condition eq $each])
    }</data>
  }</record>
}</result>
```

C.
```
<result>{
 let $example := fn:doc("example.xml")
 let $condition := fn:distinct-values($example//@condition)
 for $each in $condition
  return <record date="{ $example//record[@date eq $each]/@date }">{
   for $record in $example/example/record
    return <data condition="{ $each }">{
     fn:count($record/data[@condition eq $each])
    }</data>
   }</record>
}</result>
```

D.
```
<result>{
 let $example := fn:doc("example.xml")
 let $condition := fn:distinct-values($example//@condition)
 for $each in $condition
  for $record in $example/example/record
   return <record date="{ $record/@date }">
    <data condition="{ $each }">{
     fn:count($record/data[@condition eq $each])
    }</data>
   </record>
}</result>
```

Options:

A- Option A

B- Option B

C- Option C

D- Option D

Answer:

B