Exam 70-464
Developing Microsoft SQL Server Databases
Version 13.75 (183 Questions)
**Topic 1, Scenario 1**  
**Application Information**

Your company receives invoices in XML format from customers. Currently, the invoices are stored as files and processed by a desktop application. The application has several performance and security issues. The application is being migrated to a SQL Server-based solution. A schema named InvoiceSchema has been created for the invoices xml.

The data in the invoices is sometimes incomplete. The incomplete data must be stored and processed as-is. Users cannot filter the data provided through views.

You are designing a SQL Server database named DB1 that will be used to receive, process, and securely store the invoice data. A third-party Microsoft .NET Framework component will be purchased to perform tax calculations. The third-party tax component will be provided as a DLL file named Treytax.dll and a source code file named Amortize.cs. The component will expose a class named TreyResearch and a method named Amortize(). The files are located in c:\temp\.

The following graphic shows the planned tables:

![Invoices (Accounting) - InvoiceStatus (Accounting)](image)

You have a sequence named Accounting.InvoiceID_Seq.

You plan to create two certificates named CERT1 and CERT2. You will create CERT1 in master. You will create CERT2 in DB1.

You have a legacy application that requires the ability to generate dynamic T-SQL statements against DB1. A sample of the queries generated by the legacy application appears in Legacy.sql.

**Application Requirements**

The planned database has the following requirements:

- All stored procedures must be signed.
- The original XML invoices must be stored in the database.
- An XML schema must be used to validate the invoice data.
Dynamic T-SQL statements must be converted to stored procedures.
Access to the .NET Framework tax components must be available to T-SQL objects.
Columns must be defined by using data types that minimize the amount of space used by each table.
Invoices stored in the InvoiceStatus table must refer to an invoice by the same identifier used by the Invoice table.
To protect against the theft of backup disks, invoice data must be protected by using the highest level of encryption.
The solution must provide a table-valued function that provides users with the ability to filter invoices by customer.
Indexes must be optimized periodically based on their fragmentation by using the minimum amount of administrative effort.

Usp_InsertInvoices.sql

```sql
01 CREATE PROCEDURE InsertInvoice @XML nvarchar(1000)
02 AS
03 DECLARE @XmlDocumentHandle INT;
04 DECLARE @XmlDocument nvarchar(1000);
05 SET @XmlDocument = @XML;
06
07 EXEC sp_xml_preparedocument @XmlDocumentHandle OUTPUT, @XmlDocument;
08
09 INSERT INTO DB1.Accounting.Invoices (;
10 InvoiceID,
11 InvoiceXML,
12 CustomerID,
13 CustomerName,
14 InvoiceDate,
15 Total,
16 PONumber
17 )
18 SELECT (NEXT VALUE FOR Accounting.InvoiceID_Seq),
19 @XML, * FROM OPENXML (@XmlDocumentHandle, '/Invoice', 2)
20 WITH (;
21 CustomerID nvarchar(11) 'Customer/@ID',
22 CustomerName nvarchar(50) 'Customer/@Name',
23 InvoiceDate date 'InvoiceDate',
24 Total decimal(8, 2) 'Total',
25 PONumber bigint 'PONumber'
26 );
27
28 EXEC sp_xml_removedocument @XmlDocumentHandle;
```

Invoices.xml
All customer IDs are 11 digits. The first three digits of a customer ID represent the customer's country. The remaining eight digits are the customer's account number.

The following is a sample of a customer invoice in XML format:

```xml
01 <?xml version="1.0"?>
02 <Invoice InvoiceDate="2012-02-20">
03  <Customer ID="00156590099" Name="Lotware" />
04  <Total>125</Total>
05  <PONumber>1666</PONumber>
06 </Invoice>
```
InvoicesByCustomer.sql

```sql
01 (SELECT CustomerID, 
02     CustomerName, 
03     InvoiceID, 
04     InvoiceDate, 
05     Total, 
06     PONumber 
07     FROM Accounting.Invoices 
08     WHERE CustomerID=@CustID);
```

Legacy.sql

```sql
01 DECLARE @sqlstring AS nvarchar(1000);
02 DECLARE @CustomerID AS varchar(11), @Total AS decimal(8,2);
03
04 SET @sqlstring=N'SELECT CustomerID, InvoiceID, Total 
05     FROM Accounting.Invoices 
06     WHERE CustomerID=@CustomerID AND Total > @Total;';
07
08 EXEC sys.sp_executesql
09     @statement=@sqlstring,
10     @params=N'@CustomerID AS varchar(11), @Total AS decimal(8,2)',
11     @CustomerID=999, @Total=500;
```

CountryFromID.sql

```sql
01 CREATE FUNCTION CountryFromID (@CustomerID varchar(11)) RETURNS varchar(20)
02 AS
03 BEGIN
04 DECLARE @Country varchar(20);
05 SET @CustomerID = LEFT(@CustomerID,3);
06 SELECT @Country = CASE @CustomerID 
07     WHEN '001' 
08         THEN 'United States'
09     WHEN '002' 
10         THEN 'Spain'
11     WHEN '003' 
12         THEN 'Japan'
13     WHEN '004' 
14         THEN 'China'
15     WHEN '005' 
16         THEN 'Brazil'
17     ELSE 'Other'
18 END;
19 RETURN @CustomerID;
20 END;
```

IndexManagement.sql
You are testing disaster recovery procedures.

You attempt to restore DB1 to a different server and you receive the following error message: "Msg 33111.
Level 16, State 3, Line 1
Cannot find server certificate with thumbprint ,0xA694FBEA88C9354E5E2567C30A2A69E8FB4C44A9
Msg 3013, Level 16, State 1, Line 1
RESTORE DATABASE is terminating abnormally."

You need to ensure that you can restore DB1 to a different server.
Which code segment should you execute?

A. RESTORE CERTIFICATE CERT2
   FROM FILE='CERT2.CER'
   WITH PRIVATE KEY (FILE = 'CERT2.KEY',
   DECRYPTION BY PASSWORD='p@ssw0rd1');

B. CREATE CERTIFICATE CERT1
   FROM FILE='CERT1.CER'
   WITH PRIVATE KEY (FILE = 'CERT1.KEY',
   DECRYPTION BY PASSWORD='p@ssw0rd1');

C. CREATE CERTIFICATE CERT2
   ENCRYPTION BY PASSWORD='p@ssw0rd1'
   WITH SUBJECT = 'EncryptionCertificate';

D. CREATE CERTIFICATE CERT1
   ENCRYPTION BY PASSWORD='p@ssw0rd1'
   WITH SUBJECT = 'EncryptionCertificate';

**Answer: B**

---

**Question No : 2 - (Topic 1)**

You need to create the InvoiceStatus table in DB1.

How should you define the InvoiceID column in the CREATE TABLE statement?
Which data type should you use for CustomerID?

A. varchar(11)  
B. bigint  
C. nvarchar(11)  
D. char(11)

**Answer: D**

**Explanation:**

Invoices.xml

All customer IDs are 11 digits. The first three digits of a customer ID represent the customer's country. The remaining eight digits are the customer's account number.

- int: \(-2^{31} \) to \(2^{31}-1\) (just 10 digits max)
- bigint: \(-2^{63} \) to \(2^{63}-1\) (9,223,372,036,854,775,807)

You need to modify InsertInvoice to comply with the application requirements.

Which code segment should you execute?

A. OPEN CERT1;
   ALTER PROCEDURE Accounting.usp_InsertInvoice
   WITH ENCRYPTION;
   CLOSE CERT1;

B. OPEN CERT2;
   ALTER PROCEDURE Accounting.usp_InsertInvoice
   WITH ENCRYPTION;
   CLOSE CERT2;

C. ADD SIGNATURE TO Accounting.usp_InsertInvoice
   BY CERTIFICATE CERT1;

D. ADD SIGNATURE TO Accounting.usp_InsertInvoice
   BY CERTIFICATE CERT2;

Answer: D
You attempt to process an invoice by using usp_InsertInvoice.sql and you receive the following error message: "Msg 515, Level 16, State 2, Procedure usp_InsertInvoice, Line 10

Cannot insert the value NULL into column 'InvoiceDate', table 'DB1.Accounting.Invoices'; column does not allow nulls. INSERT fails."

You need to modify usp_InsertInvoice.sql to resolve the error.

How should you modify the INSERT statement?

A. InvoiceDate varchar(100) 'InvoiceDate',
B. InvoiceDate varchar(100) 'Customer/InvoiceDate',
C. InvoiceDate date '@InvoiceDate',
D. InvoiceDate date 'Customer/@InvoiceDate',

Answer: C

You need to modify the function in CountryFromID.sql to ensure that the country name is returned instead of the country ID.

Which line of code should you modify in CountryFromID.sql?

A. 04
B. 05
C. 06
D. 19

Answer: D

Explanation:
### Question No : 7 - (Topic 1)

You execute IndexManagement.sql and you receive the following error message:

"Msg 512, Level 16, State 1, Line 12
Subquery returned more than 1 value. This is not permitted when the subquery follows =, !=, <, <=, >, >= or when the subquery is used as an expression."

You need to ensure that IndexManagement.sql executes properly.

Which WHILE statement should you use at line 18?

A. WHILE SUM(@RowNumber) < (SELECT @counter FROM @indextable)
B. WHILE @counter < (SELECT COUNT(RowNumber) FROM @indextable)
C. WHILE COUNT(@RowNumber) < (SELECT @counter FROM @indextable)
D. WHILE @counter < (SELECT SUM(RowNumber) FROM @indextable)

**Answer: B**

### Question No : 8 - (Topic 1)

You need to convert the functionality of Legacy.sql to use a stored procedure.

Which code segment should the stored procedure contain?
A. Option A
B. Option B
C. Option C
D. Option D

Answer: D

Explanation:

Question No : 9 - (Topic 1)

You need to create a function that filters invoices by CustomerID. The SELECT statement
for the function is contained in InvoicesByCustomer.sql.

Which code segment should you use to complete the function?

A. CREATE FUNCTION Accounting.fnInvoicesByCustomer (@CustID varchar(11))
   RETURNS @tblInvoices TABLE (CustomerID bigint, CustomerName NVARCHAR(255),
   InvoiceID bigint, InvoiceDate date, Total decimal(8,2), PONumber bigint)
   AS

B. CREATE FUNCTION Accounting.fnInvoicesByCustomer (@CustID varchar(11))
   RETURNS @tblInvoices TABLE (CustomerID bigint, CustomerName NVARCHAR(255),
   InvoiceID bigint, InvoiceDate date, Total decimal(8,2), PONumber bigint)
   AS
   INSERT INTO @tblInvoices

C. CREATE FUNCTION Accounting.fnInvoicesByCustomer (@CustID varchar(11))
   RETURNS xml
   AS
   RETURN

D. CREATE FUNCTION Accounting.fnInvoicesByCustomer (@CustID varchar(11))
   RETURNS @tblInvoices TABLE (CustomerID bigint, CustomerName NVARCHAR(255),
   InvoiceID bigint, InvoiceDate date, Total decimal(8,2), PONumber bigint)
   AS

A. Option A
B. Option B
C. Option C
D. Option D

Answer: A

---

You need to build a stored procedure that amortizes the invoice amount. Which code segment should you use to create the stored procedure? To answer, move the appropriate code segments from the list of code segments to the answer area and arrange them in the correct order.
**Topic 2, Scenario 2**

**Application Information**

You have two servers named SQL1 and SQL2 that have SQL Server 2012 installed.

You have an application that is used to schedule and manage conferences.
Users report that the application has many errors and is very slow.

You are updating the application to resolve the issues.

You plan to create a new database on SQL1 to support the application. A junior database administrator has created all the scripts that will be used to create the database. The script that you plan to use to create the tables for the new database is shown in Tables.sql. The script that you plan to use to create the stored procedures for the new database is shown in StoredProcedures.sql. The script that you plan to use to create the indexes for the new database is shown in Indexes.sql. (Line numbers are included for reference only.)

A database named DB2 resides on SQL2. DB2 has a table named SpeakerAudit that will audit changes to a table named Speakers.

A stored procedure named usp_UpdateSpeakersName will be executed only by other stored procedures. The stored procedures executing usp_UpdateSpeakersName will always handle transactions.

A stored procedure named usp_SelectSpeakersByName will be used to retrieve the names of speakers. Usp_SelectSpeakersByName can read uncommitted data.

A stored procedure named usp_GetFutureSessions will be used to retrieve sessions that will occur in the future.

Procedures.sql
01 CREATE PROCEDURE usp_UpdateSpeakerName
02     @SpeakerID int,
03     @LastName nvarchar(100)
04 AS
05
06 BEGIN TRY
07
08 UPDATE Speakers
09 SET LastName = @LastName
10 WHERE SpeakerID = @SpeakerID;
11
12 INSERT INTO SQL2.DB2.dbo.SpeakerAudit(SpeakerID, LastName)
13 VALUES (@SpeakerID, @LastName);
14
15 END TRY
16 BEGIN CATCH
17
18 END CATCH;
19
20 GO
21
22 CREATE PROCEDURE usp_SelectSpeakersByName
23     @LastName nvarchar(100)
24 AS
25 SELECT SpeakerID,
26     FirstName,
27     LastName
28 FROM Speakers
29 WHERE LastName LIKE @LastName + '%'
30
31 GO
32
33 CREATE PROCEDURE usp_InsertSessions
34     @SessionData SessionDataTable READONLY
35 AS
36 INSERT INTO Sessions
37 (SpeakerID, Title, Abstract, DeliveryTime, TitleAndSpeaker)
38 SELECT SpeakerID, Title, Abstract, DeliveryTime, TitleAndSpeaker
39 FROM @SessionData;
40 GO
41
42 CREATE PROCEDURE usp_UpdateSessionRoom
43     @RoomID int,
44     @SpeakerID int
45 AS
SET TRANSACTION ISOLATION LEVEL SNAPSHOT
BEGIN TRANSACTION;

SELECT SessionID,
    Title
FROM Sessions
WHERE SpeakerID = @SpeakerID;

UPDATE Sessions
SET RoomID = @RoomID
WHERE SpeakerID = @SpeakerID;

COMMIT TRANSACTION;

CREATE PROCEDURE usp_AttendeesReport
    @LastName varchar(100)
AS
SELECT FirstName + ' ' + LastName AS FullName
FROM Attendees
WHERE LastName = @LastName;
GO

CREATE PROCEDURE usp_GetFutureSessions
AS
SELECT SpeakerID,
    RoomID,
    DeliveryTime
FROM Sessions
GO

CREATE PROCEDURE usp_TestSpeakers
AS
EXECUTE usp_SelectSpeakersByName 'a';
EXECUTE usp_SelectSpeakersByName 'an';
EXECUTE usp_SelectSpeakersByName 'and';
EXECUTE usp_SelectSpeakersByName 'ander';
EXECUTE usp_SelectSpeakersByName 'anderson';
EXECUTE usp_SelectSpeakersByName 'b';
EXECUTE usp_SelectSpeakersByName 'bi';
...
EXECUTE usp_SelectSpeakersByName 'zzz';
GO

Indexes.sql
CREATE INDEX IX_Sessions ON Sessions (SessionID, DeliveryTime) INCLUDE (RoomID)
GO
CREATE INDEX IX_Speakers ON Speakers (LastName);
GO
CREATE INDEX IX_Attendees_Name ON Attendees (FirstName, LastName);
GO
CREATE INDEX IX_Attendees_Confirmed ON Attendees (Confirmed);
GO
You need to provide referential integrity between the Sessions table and Speakers table.

Which code segment should you add at line 47 of Tables.sql?
A. Option A
B. Option B
C. Option C
D. Option D

Answer: B

Explanation:

---

**Question No : 12 - (Topic 2)**

You execute usp_TestSpeakers.

You discover that usp_SelectSpeakersByName uses inefficient execution plans.

You need to update usp_SelectSpeakersByName to ensure that the most efficient execution plan is used.
What should you add at line 30 of Procedures.sql?

A. OPTION (FORCESCAN)
B. OPTION (FORCESEEK)
C. OPTION (OPTIMIZE FOR UNKNOWN)
D. OPTION (OPTIMIZE FOR (@LastName= 'Anderson'))

Answer: C
Explanation:

---

You need to recommend a solution to ensure that SQL1 supports the auditing requirements of usp_UpdateSpeakerName.

What should you include in the recommendation?

A. The Distributed Transaction Coordinator (DTC)
B. Transactional replication
C. Change data capture
D. Change tracking

Answer: A

---

You are evaluating the table design.

You need to recommend a change to Tables.sql that reduces the amount of time it takes for usp_AttendeesReport to execute.

What should you add at line 14 of Tables.sql?

A. FullName nvarchar(100) NOT NULL CONSTRAINT DF_FuIlName DEFAULT
(dbo.CreateFullName (FirstName, LastName)),
B. FullName AS (FirstName + ' ' + LastName),
C. FullName nvarchar(100) NOT NULL DEFAULT (dbo.CreateFullName (FirstName, LastName)).
D. FullName AS (FirstName + ' ' + LastName) PERSISTED,

Answer: D
Explanation:

---

You need to modify usp_SelectSpeakersByName to support server-side paging. The solution must minimize the amount of development effort required.

What should you add to usp_SelectSpeakersByName?

A. A table variable
B. An OFFSET-FETCH clause
C. The ROWNUMBER keyword
D. A recursive common table expression

Answer: B
Explanation:
You need to add a new column named Confirmed to the Attendees table.

The solution must meet the following requirements:

- Have a default value of false.
- Minimize the amount of disk space used.

Which code block should you use?

A. ALTER TABLE Attendees
   ADD Confirmed bit DEFAULT 0;
B. ALTER TABLE Attendees
   ADD Confirmed char(1) DEFAULT '1';
C. ALTER TABLE Attendees
   ADD Confirmed bit DEFAULT 1;
D. ALTER TABLE Attendees
   ADD Confirmed char(1) DEFAULT '1';

Answer: A

Explanation:

You need to create the object used by the parameter of usp_InsertSessions.

Which statement should you use?

A. CREATE XML SCHEMA COLLECTION SessionDataTable
B. CREATE TYPE SessionDataTable AS Table
C. CREATE SCHEMA SessionDataTable
D. CREATE TABLE SessionDataTable

Answer: B
Developers report that usp_UpdateSessionRoom periodically returns error 3960.

You need to prevent the error from occurring. The solution must ensure that the stored procedure returns the original values to all of the updated rows.

What should you configure in Procedures.sql?

A. Replace line 46 with the following code:
   SET TRANSACTION ISOLATION LEVEL SERIALIZABLE
B. Replace line 46 with the following code:
   SET TRANSACTION ISOLATION LEVEL REPEATABLE READ
C. Move the SELECT statement at line 49 to line 57.
D. Move the SET statement at line 46 to line 53.

Answer: A

You discover that usp.SelectSpeakersByName executes slowly if usp_UpdateSpeakerName executes simultaneously.

You need to minimize the execution time of usp.SelectSpeakersByName. The solution must not affect the performance of the other stored procedures.

What should you update?

A. Usp_UpdateSpeakerName to use the NOLOCK query hint
B. Usp_UpdateSpeakerName to use snapshot isolation
C. Usp_SelectSpeakersByName to use the NOLOCK query hint
D. Usp_SelectSpeakersByName to use snapshot isolation

Answer: C

Explanation: NOLOCK
Is equivalent to READUNCOMMITTED.
READUNCOMMITTED
Specifies that dirty reads are allowed.
While testing usp.GetFutureSessions, you discover that IX_Sessions is accessed by a scan rather than a seek.

You need to minimize the amount of time it takes to execute usp_GetFutureSessions.

What should you do? (Each correct answer presents part of the solution. Choose all that apply.)

A. Change line 02 of Indexes.sql to:
   (DeliveryTime, SessionID)
B. At line 04 of Indexes.sql, add:
   WHERE GETDATE() < DeliveryTime;
C. Change line 02 of Indexes.sql to:
   (SpeakerID, RoomID, DeliveryTime)
D. Change line 74 of Procedures.sql to:
   WHERE GETDATE() > DeliveryTime;
E. Change line 74 of Procedures.sql to:
   WHERE GETDATE() < DeliveryTime;
F. At line 04 of Indexes.sql, add:
   WHERE GETDATE() > DeliveryTime;

A. Option A
B. Option B
You need to ensure that if any of the statements in usp_UpdateSpeakerName return an error message, all of the changes executed by usp_UpdateSpeakerName are not committed to the database.

What should you do in Procedures.sql? (Each correct answer presents part of the solution. Choose all that apply.)

- **A.** Add the following at line 17:
  
  ROLLBACK TRANSACTION

- **B.** Add the following at line 05:
  
  BEGIN TRANSACTION SpeakerUpdate

- **C.** Add the following at line 05:
  
  SAVE TRANSACTION SpeakerUpdate

- **D.** Add the following at line 17:
  
  ROLLBACK TRANSACTION SpeakerUpdate

- **E.** Add the following at line 07:
  
  BEGIN TRANSACTION

**Answer:** B,E
**Explanation:** Future delivery dates.
You are evaluating the index design.

You need to recommend a change to Indexes.sql that will minimize the amount of time it takes for usp_AttendeesReport to execute. The solution must minimize the amount of database fragmentation.

Which line of code should you use to replace line 12 of Indexes.sql?

A. (LastName);
B. (FirstName) INCLUDE (LastName);
C. (LastName, FirstName);
D. (LastName) INCLUDE (FirstName);

Answer: C

You need to create the object used by the parameter of usp_InsertSessions.

Which statement should you use?

A. CREATE SCHEMA SessionDataTable
B. CREATE TYPE SessionDataTable AS Table
C. CREATE TABLE SessionDataTable
D. CREATE XML SCHEMA COLLECTION SessionDataTable

Answer: A
Topic 3, Scenario 3

Application Information
You have two servers named SQL1 and SQL2. SQL1 has SQL Server 2012 Enterprise installed. SQL2 has SQL Server 2008 Standard installed.

You have an application that is used to manage employees and office space. Users report that the application has many errors and is very slow.

You are updating the application to resolve the issues. You plan to create a new database on SQL1 to support the application. The script that you plan to use to create the tables for the new database is shown in Tables.sql. The script that you plan to use to create the stored procedures for the new database is shown in StoredProcedures.sql. The script that you plan to use to create the indexes for the new database is shown in Indexes.sql.

A database named DB2 resides on SQL2. DB2 has a table named EmployeeAudit that will audit changes to a table named Employees.

A stored procedure named usp_UpdateEmployeeName will be executed only by other stored procedures. The stored procedures executing usp_UpdateEmployeeName will always handle transactions.

A stored procedure named usp_SelectEmployeesByName will be used to retrieve the names of employees. Usp_SelectEmployeesByName can read uncommitted data.

A stored procedure named usp_GetFutureOfficeAssignments will be used to retrieve office assignments that will occur in the future.

StoredProcedures.sql
CREATE PROCEDURE usp_UpdateEmployeeName
    @EmployeesInfo EmployeesInfo READONLY
AS
BEGIN TRY
    UPDATE Employees
    SET LastName = ei.LastName
    FROM Employees e
    INNER JOIN @EmployeesInfo ei ON e.EmployeeID = ei.EmployeeID;
    INSERT INTO SQL2.DB2.dbo.EmployeeAudit(EmployeeID, LastName)
    SELECT EmployeeID, LastName
    FROM @EmployeesInfo;
END TRY
BEGIN CATCH
END CATCH;
GO

CREATE PROCEDURE usp_SelectEmployeesByName
    @LastName nvarchar(100)
AS
    SELECT EmployeeID,
           FirstName,
           LastName
    FROM Employees
    WHERE LastName LIKE @LastName + '%';
GO

CREATE PROCEDURE usp_UpdateOffice
    @OfficeID int,
    @EmployeeID int
AS
    SET TRANSACTION ISOLATION LEVEL SNAPSHOT
    BEGIN TRANSACTION;
    SELECT OfficeID,
           OfficeName
    FROM Offices
    WHERE EmployeeID = @EmployeeID;
    UPDATE Offices
    SET EmployeeID = @EmployeeID,
        StartDate = GETDATE()
    WHERE OfficeID = @OfficeID;
    COMMIT TRANSACTION;
GO

CREATE PROCEDURE usp_GetFutureOfficeAssignments
AS
    SELECT EmployeeID,
           OfficeID,
           StartDate
    FROM Offices
    WHERE StartDate > GETDATE();
GO
01 CREATE INDEX IX_Offices ON Offices
02 (EmployeeID, StartDate)
03 INCLUDE (OfficeID)
04
05 GO
06
07 CREATE INDEX IX_Employees ON Employees
08 (LastName);
09 GO
10

Tables.sql

01 CREATE DATABASE HumanResources;
02 GO
03
04 ALTER DATABASE HumanResources
05 SET ALLOW_SNAPSHOT_ISOLATION ON;
06 GO
07
08 USE HumanResources
09 GO
10
11 CREATE TABLE Employees
12 (  
13    EmployeeID int IDENTITY(1,1) NOT NULL,
14    FirstName nvarchar(100) NOT NULL,
15    LastName nvarchar(100) NOT NULL,
16    );
17 GO
18
19 CREATE TABLE Offices
20 (  
21    OfficeID int IDENTITY(1,1) NOT NULL,
22    EmployeeID int NOT NULL,
23    OfficeName nvarchar(100) NOT NULL,
24    StartDate datetime NOT NULL
25    );
26 GO
You execute usp_SelectEmployeesByName multiple times, passing strings of varying lengths to @LastName. You discover that usp_SelectEmployeesByName uses inefficient execution plans.

You need to update usp_SelectEmployeesByName to ensure that the most efficient execution plan is used.

What should you add at line 31 of StoredProcedures.sql?

A. OPTION (ROBUST PLAN)
B. OPTION (OPTIMIZE FOR UNKNOWN)
C. OPTION (KEEP PLAN)
D. OPTION (KEEPFIXED PLAN)

Answer: B
Explanation:

You need to recommend a solution to ensure that SQL1 supports the auditing requirements of usp_UpdateEmployeeName.

What should you include in the recommendation?

A. Change data capture
B. Change tracking
C. Transactional replication
D. The Distributed Transaction Coordinator (DTC)

Answer: D