Vendor: Microsoft

Exam Code:70-762

Exam Name: Developing SQL Databases

Version: Demo

QUESTION 1

You run the following Transact-SQL following statement:

```
CREATE TABLE Customer(
      CustomerId INT IDENITY (1, 1) PRIMARY KEY,
      Code CHAR(S) NOT NULL,
      FirstName VARCHAR (50) NOT NULL,
      LastName VARCHAR (50) NOT NULL
)
```

Customer records may be inserted individually or in bulk from an application.

You observe that the application attempts to insert duplicate records.

You must ensure that duplicate records are not inserted and bulk insert operations continue without notifications.

Which Transact-SQL statement should you run?

A. CREATE UNIQUE NONCLUSTERED INDEX IX_Customer_Code ON Customer (Code) WITH (ONLINE = OFF)

B. CREATE UNIQUE INDEX IX_CUSTOMER_Code O Customer (Code) WITH (IGNORE_DUP_KEY = ON)

C. CREATE UNIQUE INDEX IX Customer Code ON Customer (Code) WITH (IGNORE DUP KEY =OFF)

D. CREATE UNIQUE NONCLUSTERED INDEX IX_Customer_Code ON Customer (Code)

E. CREATE UNIQUE NONCLUSTERED INDEX IX_Customer_Code ON Customer (Code) WITH (ONLINE = ON)

Correct Answer: B

IGNORE_DUP_KEY = { ON | OFF } specifies the error response when an insert operation attempts to insert duplicate key values into a unique index. The IGNORE_DUP_KEY option applies only to insert operations after the index is created

or rebuilt. The option has no effect when executing CREATE INDEX, ALTER INDEX, or UPDATE. The default is OFF.

Incorrect Answers:

ONLINE = { ON | OFF } specifies whether underlying tables and associated indexes are available for queries and data modification during the index operation. The default is OFF.

References: https://docs.microsoft.com/en-us/sql/t-sql/statements/create-index-transact-sql?view=sql-server-2017

QUESTION 2

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution. Determine whether the solution meets the stated goals.

You have a database that contains a table named Employees. The table stores information about the employees of your

company.

You need to implement and enforce the following business rules:

Limit the values that are accepted by the Salary column.

Prevent salaries less than $15,000 and greater than $300,000 from being entered.

Determine valid values by using logical expressions.

Do not validate data integrity when running DELETE statements.

Solution: You implement a FOR UPDATE trigger on the table.

Does the solution meet the goal?

A. Yes

B. No

Correct Answer: B

References: http://stackoverflow.com/questions/16081582/difference-between-for-update-of-and-for-update

---

QUESTION 3

You have several real-time applications that constantly update data in a database. The applications run more than 400 transactions per second that insert and update new metrics from sensors.

A new web dashboard is released to present the data from the sensors. Engineers report that the applications take longer than expected to commit updates.

You need to change the dashboard queries to improve concurrency and to support reading uncommitted data.

What should you do?

A. Use the NOLOCK option.

B. Execute the DBCC UPDATEUSAGE statement.

C. Use the max worker threads option.

D. Use a table-valued parameter.

E. Set SET ALLOW_SNAPSHOT_ISOLATION to ON.

F. Set SET XACT_ABORT to ON.

G. Execute the ALTER TABLE T1 SET (LOCK_ESCALATION = AUTO); statement.

H. Use the OUTPUT parameters.

Correct Answer: A

The NOLOCK hint allows SQL to read data from tables by ignoring any locks and therefore not being blocked by other

processes. This can improve query performance, but also introduces the possibility of dirty reads.

Incorrect Answers:

F: When SET XACT_ABORT is ON, if a Transact-SQL statement raises a run-time error, the entire transaction is terminated and rolled back.

G: DISABLE, not AUTO, would be better.

There are two more lock escalation modes: AUTO and DISABLE.

The AUTO mode enables lock escalation for partitioned tables only for the locked partition. For non-partitioned tables it works like TABLE. The DISABLE mode removes the lock escalation capability for the table and that is important when

concurrency issues are more important than memory needs for specific tables.

Note: SQL Server\\'s locking mechanism uses memory resources to maintain locks. In situations where the number of row or page locks increases to a level that decreases the server\\'s memory resources to a minimal level, SQL Server\\'s locking strategy converts these locks to entire table locks, thus freeing memory held by the many single row or page locks to one table lock. This process is called lock escalation, which frees memory, but reduces table concurrency.

References: https://www.mssqltips.com/sqlservertip/2470/understanding-the-sql-server-nolock-hint/

QUESTION 4

Note: This question is part of a series of questions that present the same scenario. Each question in this series contains a unique solution. Determine whether the solution meets the stated goals. The Account table was created using the following Transact-SQL statement:

```
CREATE TABLE Account
(
    AccountNumber int NOT NULL,
    ProductCode char(2) NOT NULL,
    Status tinyint NOT NULL,
    OpenDate date NOT NULL,
    CloseDate date,
    Balance decimal(15,2),
    AvailableBalance decimal(15,2)
);
```

There are more than 1 billion records in the Account table. The Account Number column uniquely identifies each account. The ProductCode column has 100 different values. The values are evenly distributed in the table. Table statistics are refreshed and up to date.

You frequently run the following Transact-SQL SELECT statements:

```
SELECT ProductCode, SUM(Balance) AS TotalSUM FROM Account WHERE ProductCode
<> 'CD' GROUP BY ProductCode;
SELECT AccountNumber, Balance FROM Account WHERE ProductCode = 'CD'
```

You must avoid table scans when you run the queries. You need to create one or more indexes for the table. Solution: You run the following Transact-SQL statement:

```
CREATE CLUSTERED INDEX PK_Account On Account(AccountNumber);
CREATE NONCLUSTERED INDEX IX_Account_ProductCode On Account(ProductCode)
INCLLUDE (Balance);
```

Does the solution meet the goal?

A. Yes

B. No

Correct Answer: A

Create a clustered index on theAccountNumber column as it is unique. Create a nonclustered index that includes the ProductCode column.

References:https://msdn.microsoft.com/en-us/library/ms190457.aspx

---

QUESTION 5

You are experiencing performance issues with the database server.

You need to evaluate schema locking issues, plan cache memory pressure points, and backup I/O problems.

What should you create?

A. a System Monitor report

B. a sys.dm_exec_query_stats dynamic management view query

C. a sys.dm_exec_session_wait_stats dynamicmanagement view query

D. an Activity Monitor session in Microsoft SQL Management Studio.

Correct Answer: C

sys.dm_exec_session_wait_stats returns information about all the waits encountered by threads that executed for each session. You can use this view to diagnose performance issues with the SQL Server session and also with specific queries and batches.

Note: SQL Server wait stats are, at their highest conceptual level, grouped into two broad categories: signal waits and resource waits. A signal wait is accumulated by processes running on SQL Server which are waiting for a CPU to become available (so called because the process has "signaled" that it is ready for processing). A resource wait is accumulated by processes running on SQL Server which are waiting fora specific resource to become available, such as waiting for the release of a lock on a specific record.

QUESTION 6

NOTE: This question is part of a series of questions that present the same scenario. Each question in the series

contains a unique solution. Determine whether the solution meets the stated goals.

Your company has employees in different regions around the world.

You need to create a database table that stores the following employee attendance information:

1.

Employee ID

2.

Date and time employee checked in to work

3.

Date and time employee checked out of work

Date and time information must be time zone aware and must not store fractional seconds.

Solution: You run the following Transact-SQL statement:

```
CREATE TABLE [dbo].[EmployeeAttendance](
      EmployeeID int NOT NULL,
      DateCheckedIn datetime2(0) NOT NULL,
      DateCheckedOut datetime2(0) NOT NULL )
```

Does the solution meet the goal?

A. Yes

B. No

Correct Answer: B

Datetime2 stores fractional seconds.

Datetime2 defines a date that is combined with a time of day that is based on 24-hour clock. datetime2 can be considered as an extension of the existing datetime type that has a larger date range, a larger default fractional precision, and

optional user-specified precision.

Reference:

https://docs.microsoft.com/en-us/sql/t-sql/data-types/datetime2-transact-sql?view=sql-server-2017

https://msdn.microsoft.com/en-us/library/bb677335.aspx

QUESTION 7

Note: The question is part of a series of questions that use the same or similar answer choices. An answer choice may be correct for more than one question in the series. Each question is independent of the other question in the series.

Information and details provided in a question apply only to that question.

You have a reporting database that includes a non-partitioned fact table named Fact_Sales. The table is persisted on disk.

Users report that their queries take a long time to complete. The system administrator reports that the table takes too much space in the database. You observe that there are no indexes defined on the table, and many columns have

repeating values.

You need to create the most efficient index on the table, minimize disk storage and improve reporting query performance.

What should you do?

A. Create a clustered index on the table.

B. Create a nonclustered index on the table.

C. Create a nonclustered filtered index on the table.

D. Create a clustered columnstore index on the table.

E. Create a nonclustered columnstore index on the table.

F. Create a hash index on the table.

Correct Answer: D

The columnstore index is the standard for storing and querying largedata warehousing fact tables. It uses column-based data storage and query processing to achieve up to 10x query performance gains in your data warehouse over

traditional row-oriented storage, and up to 10x data compression over the uncompressed data size.

A clustered columnstore index is the physical storage for the entire table.

QUESTION 8

Note: this question is part of a series of questions that use the same or similar answer choices. An answer choice may be correct for more than one question in the series. Each question is independent of the other questions in the series.

Information and details provided in a question apply only to that question.

You are developing and application to track customer sales.

You need to create an object that meet the following requirements:

- Run managed code packaged in an assembly that was created in the Microsoft.NET Framework and uploaded in Microsoft SQL Server.

-

Run written a transaction and roll back if a future occurs.

-

Run when a table is created or modified.

What should you create?

A. extended procedure

B. CLR procedure

C. user-defined procedure

D. DML trigger

E. scalar-valued function

F. table-valued function

Correct Answer: B

The common language runtime (CLR) is the heart of the Microsoft .NET Framework andprovides the execution environment for all .NET Framework code. Code that runs within the CLR is referred to as managed code. With the CLR hosted in Microsoft SQL Server (called CLR integration), you can author stored procedures, triggers, user-defined functions, user-defined types, and user-defined aggregates in managed code. Because managed code compiles to native code prior to execution, you can achieve significant performance increases in some scenarios.

QUESTION 9

Note: This question is part of a series of questions that present the same scenario. Each question in this series contains a unique solution. Determine whether the solution meets the stated goals. The Account table was created using the following Transact-SQL statement:

```
CREATE TABLE Account
(
    AccountNumber int NOT NULL,
    ProductCode char(2) NOT NULL,
    Status tinyint NOT NULL,
    OpenDate date NOT NULL,
    CloseDate date,
    Balance decimal(15,2),
    AvailableBalance decimal(15,2)
);
```

There are more than 1 billion records in the Account table. The Account Number column uniquely identifies each account. The ProductCode column has 100 different values. The values are evenly distributed in the table. Table statistics are refreshed and up to date.

You frequently run the following Transact-SQL SELECT statements:

```
SELECT ProductCode, SUM(Balance) AS TotalSUM FROM Account WHERE ProductCode
<> 'CD' GROUP BY ProductCode;
SELECT AccountNumber, Balance FROM Account WHERE ProductCode = 'CD'
```

You must avoid table scans when you run the queries. You need to create one or more indexes for the table. Solution: You run the following Transact-SQL statement:

```
CREATE NONCLUSTERES INDEX PK_Account(AccountNumber);
CREATE NONCLUSTERED INDEX IX_Account_productCode On Account(ProductCode)
INCLUDE (Balance);
```

Does the solution meet the goal?

A. Yes

B. No

Correct Answer: B

Create a clustered index on the AccountNumber column as it is unique, not a non nonclustered one.
References:https://msdn.microsoft.com/en-us/library/ms190457.aspx

---

QUESTION 10

You run the following Transact-SQL statements:

```
CREATE TABLE Customers (
 CustomerID INT NOT NULL IDENTITY PRIMARY KEY CLUSTERED,
 CustomerName NVARCHAR (100) UNIQUE NOT NULL
)

CREATE TABLE Orders (
 OrderID INT NOT NULL IDENTITY PRIMARY KEY CLUSTERED,
 CustomerID INT NOT NULL REFERENCES Customers (CustomerID),
 OrderDate DATE NOT NULL
)
CREATE VIEW v_CustomerOrder
AS SELECT
 b.CustomerName, a.OrderID, a.OrderDate,
 (SELECT COUNT(*) FROM Orders c WHERE c.CustomerID = a.CustomerID) AS CustomerOrderCount
FROM Orders a
INNER JOIN Customers b ON a.CustomerID = b.CustomerID
```

Records must only be added to the Orders table by using the view. If a customer name does not exist, then a new customer name must be created. You need to ensure that you can insert rows into the Orders table by using the view.

A. Add the CustomerID column from the Orders table and the WITH CHECK OPTION statement to the view.

B. Create an INSTEAD OF trigger on the view.

C. Add the WITH SCHEMABINDING statement to the view statement and create a clustered index on the view.

D. Remove the subquery from the view, add the WITH SCHEMABINDING statement, and add a trigger to the Orders

table to perform the required logic.

Correct Answer: A

The WITH CHECK OPTION clause forces all data-modification statements executed against the view to adhere to the criteria set within the WHERE clause of the SELECT statement defining the view. Rows cannot be modified in a way that causes them to vanish from the view.

References: http://www.informit.com/articles/article.aspx?p=130855andseqNum=4

QUESTION 11

You are optimizing the performance of a batch update process. You have tables and indexes that were created by running the following Transact-SQL statements:

```
CREATE TABLE Invoices (
    InvoiceID INT NOT NULL IDENTITY PRIMARY KEY CLUSTERED,
    CustomerID INT NOT NULL,
    OrderID INT NULL,
    IsCreditNote BIT NOT NULL,
    IsCreditValidated BIT NOT NULL DEFAULT Ø
)

CREATE INDEX IX_invoices_CustomerID_Filter_IsCreditValidated ON Invoices
(CustomerID) WHERE IsCreditValidated = 1

CREATE TABLE CreditValidation (
   CreditValidationID INT NOT NULL IDENTITY PRIMARY KEY CLUSTERED,
   CustomerID INT NOT NULL,
   ValidationDate DATETIME NOT NULL
)
```

The following query runs nightly to update the isCreditValidated field:

```
UPDATE I
SET IsCreditValidated = 1
FROM Invoices I
WHERE EXISTS (SELECT Ø FROM CreditValidation CV WHERE CV.CustomerID =
I.CustomerID AND CV.ValidationDate >= I.InvoiceDate)
AND I.IsCreditNote = 1
AND I.IsCreditValidated = Ø
AND I.InvocieDate >= DATEADD (DD, -7, GETDATE () )
```

You review the database and make the following observations:

Most of the IsCreditValidated values in the Invoices table are set to a value of 1.

There are many unique InvoiceDate values.

The CreditValidation table does not have an index.

Statistics for the index IX_invoices_CustomerID_Filter_IsCreditValidated indicate there are no individual seeks but multiple individual updates.

You need to ensure that any indexes added can be used by the update query. If the IX_invoices_CustomerId_Filter_IsCreditValidated index cannot be used by the query, it must be removed. Otherwise, the query must be modified to use with

the index.

Which three actions should you perform? Each correct answer presents part of the solution.

NOTE: Each correct selection is worth one point.

A. Add a filtered nonclustered index to Invoices on InvoiceDate that selects where IsCreditNote= 1 and IsCreditValidated = 0.

B. Rewrite the update query so that the condition for IsCreditValidated = 0 precedes the condition for IsCreditNote = 1.

C. Create a nonclustered index for invoices in IsCreditValidated, InvoiceDate with an include statement using IsCreditNote and CustomerID.

D. Add a nonclustered index for CreditValidation on CustomerID.

E. Drop the IX_invoices_CustomerId_Filter_IsCreditValidatedIndex.

Correct Answer: ABE

A filtered index is an optimized nonclustered index especially suited to cover queries that select from a well-defined subset of data. It uses a filter predicate to index a portion of rows in the table. A well-designed filtered index can improve query performance as well as reduce index maintenance and storage costs compared with full-table indexes.

References: https://docs.microsoft.com/en-us/sql/relational-databases/indexes/create-filtered-indexes

---

QUESTION 12

You suspect deadlocks on a database.

Which two trace flags in the Microsoft SQL Server error log should you locate? Each correct answer presents part of the solution.

NOTE: Each correct selection is worth one point.

A. 1204

B. 1211

C. 1222

D. 2528

E. 3205

Correct Answer: AC

Trace flag 1204 returns the resources and types of locks participating in a deadlock and also the current command affected. Trace flag 1222 returns the resources and types of locks that are participating in a deadlock and also the current command affected, in an XML format that does not comply with any XSD schema.

References: https://docs.microsoft.com/en-us/sql/t-sql/database-console-commands/dbcc-traceon-trace-flags-transact-sql?view=sql-server-2017