Microsoft 70-433

TS: Microsoft SQL Server 2008, Database Development

Version: 6.1

Topic 1, Volume A

QUESTION NO: 1

You have a user named John. He has SELECT access to the Sales schema. You need to eliminate John's SELECT access rights from the Sales.SalesOrder table without affecting his other permissions.

Which Transact-SQL statement should you use?

A. DROP USER John;
B. DENY SELECT ON Sales.SalesOrder TO John;
C. GRANT DELETE ON Sales.SalesOrder TO John;
D. REVOKE SELECT ON Sales.SalesOrder FROM John;

Answer: B
Explanation: Explanation/Reference:
REVOKE – permanently removes both granted an denied permissions on an object, resulting in no permissions.Main thing you have to remember is, this does not restrict user accessing the object completely. If user is in a role that has permission on the object for the operation, user will be able to perform the operation.

DENY – Denies permission to the object for an operation. Once it set, it takes precedence over all other GRANT permissions, user will not be able to perform the operation against the object.

To sum up, because John does not have GRANT permissions on the Sales.SalesOrder table (instead it has GRANT permission on Sales schema), then REVOKE SELECT ON Sales.SalesOrder from John will not remove any permissions.

Here is a code that shows it clearly.

```
-- create a login and user
CREATE LOGIN [John] WITH PASSWORD = '1', CHECK_POLICY = OFF
GO
USE [AdventureWorks2008]
GO
CREATE USER [John] FOR LOGIN [John]
WITH DEFAULT_SCHEMA = [dbo]
GO
-- grant permission on Sales schema
GRANT SELECT ON SCHEMA :: Sales TO [John]
-- Run SELECT with John's credentials and see
-- He sees records
EXECUTE AS USER = 'John'
```

```
SELECT * FROM Sales.SalesOrderHeader
REVERT
-- Revoke permisson for the table from him
REVOKE SELECT ON Sales.SalesOrderHeader FROM [John]
-- He still sees data
EXECUTE AS USER = 'John'
SELECT * FROM Sales.SalesOrderHeader
REVERT
-- This explicitly denies permission on SalesOrderHeader to John
-- Once this is executed, he will not be able to see data
-- even we grant him again.
DENY SELECT ON Sales.SalesOrderHeader TO [John]
-- He sees error message: The SELECT permission was denied on the object
'SalesOrderHeader', database 'AdventureWorks2008', schema 'Sales'.
EXECUTE AS USER = 'John'
SELECT * FROM Sales.SalesOrderHeader
REVERT
```

QUESTION NO: 2

You need to create a column that allows you to create a unique constraint.

Which two column definitions should you choose? (Each correct answer presents a complete solution. Choose two.)

A. nvarchar(100) NULL
B. nvarchar(max) NOT NULL
C. nvarchar(100) NOT NULL
D. nvarchar(100) SPARSE NULL

Answer: A,C
Explanation:

QUESTION NO: 3

You manage a SQL Server 2008 database that is located at your company's corporate headquarters. The database contains a table named dbo.Sales. You need to create different views of the dbo.Sales table that will be used by each region to insert, update, and delete rows. Each

regional office must only be able to insert, update, and delete rows for their respective region. Which view should you create for Region1?

A. CREATE VIEW dbo.Region1Sales
AS
SELECT SalesID,OrderQty,SalespersonID,RegionID FROM dbo.Sales
WHERE RegionID = 1;
B. CREATE VIEW dbo.Region1Sales
AS
SELECT SalesID,OrderQty,SalespersonID,RegionID FROM dbo.Sales
WHERE RegionID = 1 WITH CHECK OPTION;
C. CREATE VIEW dbo.Region1Sales
WITH SCHEMABINDING
AS SELECT SalesID,OrderQty,SalespersonID,RegionID FROM dbo.Sales WHERE RegionID = 1;
D. CREATE VIEW dbo.Region1Sales
WITH VIEW_METADATA
AS
SELECT SalesID,OrderQty,SalespersonID,RegionID FROM dbo.Sales
WHERE RegionID = 1;

Answer: B

Explanation: Explanation/Reference:

CHECK OPTION

Forces all data modification statements executed against the view to follow the criteria set within select_statement. When a row is modified through a view, the WITH CHECK OPTION makes sure the data remains visible through the view after the modification is committed.

QUESTION NO: 4

You administer a SQL Server 2008 database that contains a table name dbo.Sales, which contains the following table definition:

CREATE TABLE [dbo].[Sales](

[SalesID] [int] IDENTITY(1,1) NOT NULL PRIMARY KEY CLUSTERED,

[OrderDate] [datetime] NOT NULL,

[CustomerID] [int] NOT NULL,

[SalesPersonID] [int] NULL,

[CommentDate] [date] NULL);

This table contains millions of orders. You run the following query to determine when sales persons comment in the dbo.Sales table:

```
SELECT SalesID,CustomerID,SalesPersonID,CommentDate
FROM dbo.Sales
WHERE CommentDate IS NOT NULL
AND SalesPersonID IS NOT NULL;
```

You discover that this query runs slow. After examining the data, you find only 1% of rows have comment dates and the SalesPersonID is null on 10% of the rows. You need to create an index to optimize the query. The index must conserve disk space while optimizing your query.

Which index should you create?

A.
```
CREATE NONCLUSTERED INDEX idx1
ON dbo.Sales (CustomerID)
INCLUDE (CommentDate,SalesPersonID);
```
B.
```
CREATE NONCLUSTERED INDEX idx1
ON dbo.Sales (SalesPersonID)
INCLUDE (CommentDate,CustomerID);
```
C.
```
CREATE NONCLUSTERED INDEX idx1
ON dbo.Sales (CustomerID)
INCLUDE(CommentDate)
WHERE SalesPersonID IS NOT NULL;
```
D.
```
CREATE NONCLUSTERED INDEX idx1
ON dbo.Sales (CommentDate, SalesPersonID)
INCLUDE(CustomerID)
WHERE CommentDate IS NOT NULL;
```

Answer: D

Explanation:

QUESTION NO: 5

Your database is 5GB and contains a table named SalesHistory. Sales information is frequently inserted and updated.

You discover that excessive page splitting is occurring.

You need to reduce the occurrence of page splitting in the SalesHistory table.

Which code segment should you use?.

A. ALTER DATABASE Sales
MODIFY FILE
(NAME = Salesdat3,
SIZE = 10GB);
B. ALTER INDEX ALL ON Sales.SalesHistory
REBUILD WITH (FILLFACTOR = 60);
C. EXEC sys.sp_configure 'fill factor (%)', '60';
D. UPDATE STATISTICS Sales.SalesHistory(Products)
WITH FULLSCAN, NORECOMPUTE;

Answer: B
Explanation: Explanation/Reference:

Fill Factor
The fill-factor option is provided for fine-tuning index data storage and performance. When an index is created or rebuilt, the fill-factor value determines the percentage of space on each leaf-level page to be filled with data, reserving the remainder on each page as free space for future growth. For example, specifying a fill-factor value of 60 means that 40 percent of each leaf-level page will be left empty, providing space for index expansion as data is added to the underlying table.

Page Splits
A correctly chosen fill-factor value can reduce potential page splits by providing enough space for index expansion as data is added to the underlying table.When a new row is added to a full index page, the Database Engine moves approximately half the rows to a new page to make room for the new row. This reorganization is known as a page split. A page split makes room for new records, but can take time to perform and is a resource intensive operation. Also, it can cause fragmentation that causes increased I/O operations.

Fragmentation
Fragmentation breaks down to physical and logical fragmentation (or, internal and external fragmentation).
Physical/Internal fragmentation is caused when there is wasted space in the index caused by page splits, deletes, FILLFACTOR and PAD_INDEX. Logical/External fragmentation is when the pages that make up the leaf levels of the index are not in good order. This is usually caused by page splits. Both cause performance problems. Internal fragmentation causes problems because the indexes get bigger and bigger requiring more and more pages to store the data, which means that reads from the index slow things down. External fragmentation causes problems because when the data needs to be read from the index it has to skip all over the place following the links between pages, again, slowing things down.

The SQL Server Database Engine automatically maintains indexes whenever insert, update, or delete operations are made to the underlying data. Over time these modifications can cause the

information in the index to become scattered in the database (fragmented). Fragmentation exists when indexes have pages in which the logical ordering, based on the key value, does not match the physical ordering inside the data file.
You can remedy index fragmentation by either reorganizing an index or by rebuilding an index.
To reorganize one or more indexes, use the ALTER INDEX statement with the REORGANIZE clause.

Reorganizing an index defragments the leaf level of clustered and nonclustered indexes on tables and views by physically reordering the leaf-level pages to match the logical order (left to right) of the leaf nodes. Having the pages in order improves index-scanning performance. The index is reorganized within the existing pages allocated to it; no new pages are allocated.

Rebuilding an index drops the index and creates a new one. In doing this, fragmentation is removed, disk space is reclaimed by compacting the pages using the specified or existing fill factor setting, and the index rows are reordered in contiguous pages (allocating new pages as needed). This can improve disk performance by reducing the number of page reads required to obtain the requested data.
You can use the CREATE INDEX with the DROP_EXISTING clause or ALTER INDEX with the REBUILD clause statements to set the fill-factor value for individual indexes.
EXEC sys.sp_configure 'fill factor (%)', '60'; will modify the server default fill factor value.

QUESTION NO: 6

You have a table named dbo.Customers. The table was created by using the following Transact-SQL statement:

```
CREATE TABLE dbo.Customers
(
CustomerID int IDENTITY(1,1) PRIMARY KEY CLUSTERED,
AccountNumber nvarchar(25) NOT NULL,
FirstName nvarchar(50) NOT NULL,
LastName nvarchar(50) NOT NULL,
AddressLine1 nvarchar(255) NOT NULL,
AddressLine2 nvarchar(255) NOT NULL,
City nvarchar(50) NOT NULL,
```

StateProvince nvarchar(50) NOT NULL,

Country nvarchar(50) NOT NULL,

PostalCode nvarchar(50) NOT NULL,

CreateDate datetime NOT NULL DEFAULT(GETDATE()),

ModifiedDate datetime NOT NULL DEFAULT(GETDATE())

)

You create a stored procedure that includes the AccountNumber, Country, and StateProvince columns from the dbo.Customers table. The stored procedure accepts a parameter to filter the output on the AccountNumber column.

You need to optimize the performance of the stored procedure. You must not change the existing structure of the table.

Which Transact-SQL statement should you use?

A. CREATE STATISTICS ST_Customer_AccountNumber ON dbo.Customer (AccountNumber) WITH FULLSCAN;
B. CREATE CLUSTERED INDEX IX_Customer_AccountNumber ON dbo.Customer (AccountNumber);
C. CREATE NONCLUSTERED INDEX IX_Customer_AccountNumber ON dbo.Customer (AccountNumber) WHERE AccountNumber = '';
D. CREATE NONCLUSTERED INDEX IX_Customer_AccountNumber ON dbo.Customer (AccountNumber) INCLUDE (Country, StateProvince);

Answer: D
Explanation:

QUESTION NO: 7

You have a table named Customer.

You need to ensure that customer data in the table meets the following requirements:

credit limit must be zero unless customer identification has been verified.

credit limit must be less than 10,000.

Which constraint should you use?

A. CHECK (CreditLimt BETWEEN 1 AND 10000)

B. CHECK (Verified = 1 AND CreditLimt BETWEEN 1 AND 10000)
C. CHECK ((CreditLimt = 0 AND Verified = 0) OR (CreditLimt BETWEEN 1 AND 10000 AND Verified = 1))
D. CHECK ((CreditLimt = 0 AND Verified = 0) AND (CreditLimt BETWEEN 1 AND 10000 AND Verified = 1))

Answer: C
Explanation:

QUESTION NO: 8

You have a table named AccountsReceivable. The table has no indexes. There are 75,000 rows in the table. You have a partition function named FG_AccountData. The AccountsReceivable table is defined in the following Transact-SQL statement:

CREATE TABLE AccountsReceivable (

column_a INT NOT NULL,

column_b VARCHAR(20) NULL)

ON [PRIMARY];

You need to move the AccountsReceivable table from the PRIMARY file group to FG_AccountData.

Which Transact-SQL statement should you use?

A. CREATE CLUSTERED INDEX idx_AccountsReceivable ON AccountsReceivable(column_a) ON [FG_AccountData];
B. CREATE NONCLUSTERED INDEX idx_AccountsReceivable ON AccountsReceivable(column_a) ON [FG_AccountData];
C. CREATE CLUSTERED INDEX idx_AccountsReceivable ON AccountsReceivable(column_a) ON FG_AccountData(column_a);
D. CREATE NONCLUSTERED INDEX idx_AccountsReceivable ON AccountsReceivable(column_a) ON FG_AccountData(column_a);

Answer: C
Explanation:

QUESTION NO: 9

You have a SQL Server 2008 database named Contoso with a table named Invoice. The primary key of the table is InvoiceId, and it is populated by using the identity property. The Invoice table is related to the InvoiceLineItem table. You remove all constraints from the Invoice table during a data load to increase load speed. You notice that while the constraints were removed, a row with InvoiceId = 10 was removed from the database. You need to re-insert the row into the Invoice table with the same InvoiceId value.

Which Transact-SQL statement should you use?

A. INSERT INTO Invoice (InvoiceId, ...VALUES (10, ...
B. SET IDENTITY_INSERT Invoice ON;
INSERT INTO Invoice (InvoiceId, ...
VALUES (10, ...
SET IDENTITY_INSERT Invoice OFF;
C. ALTER TABLE Invoice;
ALTER COLUMN InvoiceId int;
INSERT INTO Invoice (InvoiceId, ...
VALUES (10, ...
D. ALTER DATABASE Contoso SET SINGLE_USER;
INSERT INTO Invoice (InvoiceId, ...
VALUES (10, ...
ALTER DATABASE Contoso SET MULTI_USER;

Answer: B

Explanation:

QUESTION NO: 10

You are developing a new database. The database contains two tables named SalesOrderDetail and Product.

You need to ensure that all products referenced in the SalesOrderDetail table have a corresponding record in the Product table.

Which method should you use?

A. JOIN
B. DDL trigger
C. Foreign key constraint
D. Primary key constraint

Answer: C

Explanation: