70-433 Q&As

TS: Microsoft SQL Server 2008, Database Development

Pass Microsoft 70-433 Exam with 100% Guarantee

Free Download Real Questions & Answers **PDF** and **VCE** file from:

https://www.geekcert.com/70-433.html

100% Passing Guarantee
100% Money Back Assurance

Following Questions and Answers are all new published by Microsoft Official Exam Center

- **Instant Download** After Purchase
- **100% Money Back** Guarantee
- **365 Days** Free Update
- **800,000+** Satisfied Customers

QUESTION 1

You administer a Microsoft SQL Server 2008 database that contains a stored procedure named dbo.SalesOrderDetails. The stored procedure has following definition:

```
CREATE PROCEDURE dbo.SalesOrderDetails
  @CustomerID int,
  @OrderDate datetime,
  @SalesOrderID int
AS
SELECT
  h.SalesOrderID,
  h.OrderDate,
  d.OrderQty,
  d.ProductID
FROM
  Sales.SalesOrderHeader h
    INNER JOIN
  Sales.SalesOrderDetail d
    ON d.SalesOrderID = h.SalesOrderID
WHERE
  h.CustomerID = @CustomerID
  or h.OrderDate > @OrderDate
  or h.SalesOrderID > @SalesOrderID
GO
```


Parameter values passed to the stored procedure largely vary.

You discover that the stored procedure executes quickly for some parameters but slowly for other parameters.

You need to ensure that the query plan generated is optimized to provide the most consistent execution times for any set of parameters passed to the stored procedure.

A. OPTION (NOLCCK)

B. OPTION (KEEP PLAN)

C. OPTION (ROBUST PLAN)

D. OPTION (RECOMPILE)

Correct Answer: C

QUESTION 2

You have tables named Products and OrderDetails. The Products table has a foreign key relationship with the OrderDetails table on the ProductID column. You have the following Transact-SQL batch:

BEGIN TRY BEGIN TRANSACTION DELETE FROM Products WHERE ProductID = 5; BEGIN TRANSACTION INSERT INTO OrderDetails (OrderID, ProductID, Quantity) VALUES (1234, 5, 12); COMMIT TRANSACTION COMMIT TRANSACTION END TRY BEGIN CATCH ROLLBACK TRANSACTION PRINT ERROR_MESSAGE(); END CATCH

You need to analyze the result of executing this batch.

What should be the expected outcome?

A. 1. The product will be deleted from the Products table.

2. The order details will be inserted into the OrderDetails table.

B. 1. The product will be deleted from the Products table.

2. The order details will not be inserted into the OrderDetails table.

C. 1. The product will not be deleted from the Products table.

2. The order details will be inserted into the OrderDetails table.

D. 1. The product will not be deleted from the Products table.

2. The order details will not be inserted into the OrderDetails table.

Correct Answer: D

ROLLBACK { TRAN | TRANSACTION }

[transaction_name | @tran_name_variable

| savepoint_name | @savepoint_variable]

[;]

transaction_name

Is the name assigned to the transaction on BEGIN TRANSACTION. When nesting transactions, transaction_name must be the name from the outermost BEGIN TRANSACTION statement.

savepoint_name

Is savepoint_name from a SAVE TRANSACTION statement. Use savepoint_name when a conditional rollback should affect only part of the transaction. ROLLBACK TRANSACTION without a savepoint_name or transaction_name rolls back

to the beginning of the transaction. When nesting transactions, this same statement rolls back all inner transactions to the outermost BEGIN TRANSACTION statement. In both cases, ROLLBACK TRANSACTION decrements the

@@TRANCOUNT system function to 0. ROLLBACK TRANSACTION savepoint_name does not decrement @@TRANCOUNT.

A transaction cannot be rolled back after a COMMIT TRANSACTION statement is executed, except when the COMMIT TRANSACTION is associated with a nested transaction that is contained within the transaction being rolled back. In this

instance, the nested transaction will also be rolled back, even if you have issued a COMMIT TRANSACTION for it.

SQL Server 2008 error handling best practice

CREATE PROCEDURE SaveTranExample

@InputCandidateID INT

AS

-- Detect whether the procedure was called from an active transaction and save that for later use. -- In the procedure, @hasOuterTransaction = 0 means there was no active transaction -- and the procedure started one.

-- @hasOuterTransaction > 0 means an active transaction was started before the -- procedure was called.

DECLARE @hasOuterTransaction BIT = CASE WHEN @@TRANCOUNT > 0 THEN 1 ELSE END;

-- Save points need unique names if modules can nest otherwise you can rollback -- to the wrong save point. The solution is to use a GUID to name the save points. DECLARE @rollbackPoint nchar(32) = REPLACE(CONVERT(NCHAR(36),

NEWID()), N\\'-\\', N\\'\\'); IF @hasOuterTransaction > 0

BEGIN

-- Procedure called when there is an active transaction. -- Create a savepoint to be able to roll back only the work done in the procedure if there is an error.

SAVE TRANSACTION @rollbackPoint;

END

ELSE

-- Procedure must start its own transaction.

BEGIN TRANSACTION @rollbackPoint;

-- Modify database.

BEGIN TRY

-- Do work;

DELETE HumanResources.JobCandidate

WHERE JobCandidateID = @InputCandidateID;

-- Get here if no errors; must commit

-- any transaction started in the

-- procedure, but not commit a transaction

-- started before the transaction was called.

IF @hasOuterTransaction = 0

BEGIN

-- @hasOuterTransaction = 0 means no transaction was started before the procedure was called. -- The procedure must commit the transaction it started.

COMMIT TRANSACTION;

END

END TRY

BEGIN CATCH

-- An error occurred;

-- If the transaction is still valid

IF XACT_STATE() = 1

-- The XACT_STATE function can return the following values:

-- 1 An open transaction exists that can be either committed or rolled back.

-- 0 There is no open transaction.

-- 1 An open transaction exists, but it is in a doomed state. Due to the type of error that was raised, the transaction can only be rolled back.

BEGIN

-- Because the syntax for ROLLBACK TRANSACTION is the same for the transaction and for a savepoint -- (ROLLBACK TRANSACTION [transaction_name | @tran_name_variable | savepoint_name | @savepoint_variable]) -- we can write

the following:

ROLLBACK TRANSACTION @rollbackPoint;

-- In case @rollbackPoint has the name of a transaction, roll back to the beginning of the transaction.

-- In case @rollbackPoint has the name of a savepoint, roll back to the savepoint.

END;

ELSE IF XACT_STATE() = -1

BEGIN

IF @hasOuterTransaction = 0

BEGIN

-- Transaction started in procedure.

-- Roll back complete transaction.

ROLLBACK TRANSACTION;

END

-- If the transaction is uncommitable, a rollback to the savepoint is not allowed -- because the savepoint rollback writes to the log. Just return to the caller, which -- should roll back the outer transaction.

END

-- Execute Standard module error handler;

-- After the appropriate rollback, echo error information to the caller.

DECLARE @ErrorMessage NVARCHAR(4000);

DECLARE @ErrorSeverity INT;

DECLARE @ErrorState INT;

SELECT @ErrorMessage = ERROR_MESSAGE();

SELECT @ErrorSeverity = ERROR_SEVERITY();

SELECT @ErrorState = ERROR_STATE();

RAISERROR (@ErrorMessage, -- Message text.

@ErrorSeverity, -- Severity.

@ErrorState -- State.

);

END CATCH

GO

QUESTION 3

You have a database named Contoso. The Contoso database has a Service Broker queue named VacationRequestQueue. The Contoso database has been restored to a new server. Since restoring the database, Service Broker is no longer

able to send new messages.

You need to configure Service Broker in order to resolve the issue.

Which Transact-SQL statement should you use?

A. ALTER DATABASE Contoso SET NEW_BROKER;

B. ALTER DATABASE Contoso SET ENABLE_BROKER;

C. ALTER QUEUE VacationRequestQueue WITH STATUS = ON;

D. ALTER QUEUE VacationRequestQueue WITH ACTIVATION (STATUS = ON);

Correct Answer: A

QUESTION 4

You are the database developer for an order-processing application.

After a customer places an order, a confirmation message must be sent to the customer.

The following Transact-SQL batch has been run in the database: You need to place the message in the EmailSendQueue for the email system to process. Which Transact-SQL batch should you use?

```
ALTER DATABASE NORTHWIND SET ENABLE_BROKER;

CREATE MESSAGE TYPE EmailMessage
VALIDATION = NONE;

CREATE CONTRACT EmailContract
(EmailMessage SENT BY INITIATOR);

CREATE QUEUE EmailSendQueue;

CREATE QUEUE EMailReceiveQueue;

CREATE SERVICE EmailSendService
ON QUEUE EmailSendQueue (EmailContract);

CREATE SERVICE EmailReceiveService
ON QUEUE EmailReceiveQueue (EmailContrac
```

A.
```
DECLARE
  @EmailDialog UNIQUEIDENTIFIER,
  @Message NVARCHAR(128);
BEGIN DIALOG @EmailDialog
FROM SERVICE EmailSendService
TO SERVICE 'EmailReceiveService'
ON CONTRACT EmailContract
WITH LIFETIME = 1000;
SET @Message = N'Dear Sir/Madam. Your order has been received.';
SEND ON CONVERSATION @EmailDialog
MESSAGE TYPE EmailMessage (@Message);
GO
```

B.
```
DECLARE
  @EmailDialog BIGINT,
  @Message NVARCHAR(128);
BEGIN DIALOG @EmailDialog
TO SERVICE EmailReceiveService
FROM SERVICE 'EmailSendService'
ON CONTRACT EmailContract;
SET @Message = N'Dear Sir/Madam. Your order has been received.';
SEND ON CONVERSATION @EmailDialog
MESSAGE TYPE EmailMessage (@Message);
GO
```

C.
```
DECLARE
  @EmailDialog BIGINT,
  @Message NVARCHAR(128);
BEGIN DIALOG @EmailDialog
FROM SERVICE EmailSendService
TO SERVICE 'EmailReceiveService'
ON CONTRACT EmailContract;
SET @Message = N'Dear Sir/Madam. Your order has been received.';
SEND ON CONVERSATION @EmailDialog
MESSAGE TYPE EmailMessage (@Message);
GO
```

D.
```
DECLARE
  @EmailDialog BIGINT,
  @Message NVARCHAR(128);
BEGIN DIALOG @EmailDialog
TO SERVICE 'EmailReceiveService'
FROM SERVICE EmailSendService
ON CONTRACT EmailContract;
SET @Message = N'Dear Sir/Madam. Your order has been received.';
SEND ON CONVERSATION @EmailDialog
MESSAGE TYPE EmailMessage (@Message);
GO
```

Correct Answer: A

A. B. C. D.

Correct Answer: A

QUESTION 5

You have a SQL Server database. The database contains two schemas named Marketing and Sales. The Marketing schema is owned by a user named MarketingManager. The Sales schema is owned by a user named SalesManager.

A user named John must be able to access the Sales.Orders table by using a stored procedure named Marketing.GetSalesSummary. John is not granted a SELECT permission on the Sales.Orders table. A user named SalesUser does have

SELECT permission on the Sales.Orders table. You need to implement appropriate permissions for John and the stored procedure Marketing.GetSalesSummary.

What should you do?

A. Marketing.GetSalesSummary should be created by using the EXECUTE AS \\'SalesUser\\' clause. John should be granted EXECUTE permission on Marketing.GetSalesSummary.

B. Marketing.GetSalesSummary should be created by using the EXECUTE AS OWNER clause. John should be granted EXECUTE WITH GRANT OPTION on Marketing.GetSalesSummary.

C. Marketing.GetSalesSummary should be created by using the EXECUTE AS CALLER clause. John should be granted IMPERSONATE permission for the user named SalesUser.

D. Marketing.GetSalesSummary should be created without an EXECUTE AS clause. John should be granted SELECT permission on the Sales.Orders table.

Correct Answer: A

1.

When the module is executed, the Database Engine first verifies that the user executing the module has EXECUTE permission on the module. So John should be granted EXECUTE permission on Marketing. GetSalesSummary stored procedure.

2.

Additional permissions checks on objects that are accessed by the module are performed against the user account specified in the EXECUTE AS clause. The user executing the module is, in effect, impersonating the specified user. Because John is not granted a SELECT permission on the Sales.Orders table which is referenced by the stored procedure, EXECUTE AS CALLER is not suitable. (CALLER specifies the statements inside the module are executed in the context of the caller of the module. The user executing the module must have appropriate permissions not only on the module itself, but also on any database objects that are referenced by the module.) Because the user named SalesUser DOES have SELECT permission on the Sales.Orders table, he can be specified in EXECUTE AS clause. It means that Marketing. GetSalesSummary stored procedure should be created by using the EXECUTE AS \\'SalesUser\\' clause.