KillTest

Higher Quality, Better Service!

Q&A

http://www.killtest.com

We offer free update service for one year.

Exam : **70-761**

Title : Querying Data with Transact-SQL

Version : DEMO

1.Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
You create a table named Products by running the following Transact-SQL statement:

```
CREATE TABLE Products (
     ProductID int IDENTITY(1,1) NOT NULL PRIMARY KEY,
     ProductName nvarchar(100) NULL,
     UnitPrice decimal(18, 2) NOT NULL,
     UnitsInStock int NOT NULL,
     UnitsOnOrder int NULL
)
```

You have the following stored procedure:

```
CREATE PROCEDURE InsertProduct
     @ProductName nvarchar(100),
     @UnitPrice decimal(18,2),
     @UnitsInStock int,
     @UnitsOnOrder int
AS
BEGIN
     INSERT INTO Products(ProductName,ProductPrice,ProductsInStock,ProductsOnOrder)
     VALUES (@ProductName,@UnitPrice,@UnitsInStock,@UnitsOnOrder)
END
```

You need to modify the stored procedure to meet the following new requirements:
- Insert product records as a single unit of work.
- Return error number 51000 when a product fails to insert into the database.
- If a product record insert operation fails, the product information must not be permanently written to the database.

Solution: You run the following Transact-SQL statement:

```
ALTER PROCEDURE InsertProduct
@ProductName nvarchar(100),
@UnitPrice decimal(18,2),
@UnitsInStock int,
@UnitsOnOrder int
AS
BEGIN
     SET XACT_ABORT ON
     BEGIN TRY
          BEGIN TRANSACTION
             INSERT INTO Products(ProductName,ProductPrice,ProductsInStock,ProductsOnOrder)
             VALUES (@ProductName,@UnitPrice,@UnitsInStock,@UnitsOnOrder)
          COMMIT TRANSACTION
     END TRY
     BEGIN CATCH
          IF XACT_STATE() <> 0 ROLLBACK TRANSACTION
          THROW 51000, 'The product could not be created.', 1
     END CATCH
END
```

Does the solution meet the goal?

A. Yes

B. No

Answer: B

Explanation:

With X_ABORT ON the INSERT INTO statement and the transaction will be rolled back when an error is

raised, it would then not be possible to ROLLBACK it again in the IF XACT_STATE() <> O ROLLBACK TRANSACTION statement.
Note: A transaction is correctly defined for the INSERT INTO ..VALUES statement, and if there is an error in the transaction it will be caughtant he transaction will be rolled back, finally an error 51000 will be raised.
Note: When SET XACT_ABORT is ON, if a Transact-SQL statement raises a run-time error, the entire transaction is terminated and rolled back. XACT_STATE is a scalar function thatreports the user transaction state of a current running request. XACT_STATE indicates whether the request has an active user transaction, and whether the transaction is capable of being committed.
The states of XACT_STATE are:
- 0 There is no active user transaction for the current request.
- 1 The current request has an active user transaction. The request can perform any actions, including writing data and committing the transaction.
- 2 The current request has an active user transaction, but an error hasoccurred that has caused the transaction to be classified as an uncommittable transaction.
References:
https://msdn.microsoft.com/en-us/library/ms188792.aspx
https://msdn.microsoft.com/en-us/library/ms189797.aspx

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 that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
You create a table named Products by running the following Transact-SQL statement:

```
CREATE TABLE Products (
     ProductID int IDENTITY(1,1) NOT NULL PRIMARY KEY,
     ProductName nvarchar(100) NULL,
     UnitPrice decimal(18, 2) NOT NULL,
     UnitsInStock int NOT NULL,
     UnitsOnOrder int NULL
)
```

You have the following stored procedure:

```
CREATE PROCEDURE InsertProduct
     @ProductName nvarchar(100),
     @UnitPrice decimal(18,2),
     @UnitsInStock int,
     @UnitsOnOrder int
AS
BEGIN
     INSERT INTO Products(ProductName,ProductPrice,ProductsInStock,ProductsOnOrder)
     VALUES (@ProductName,@UnitPrice,@UnitsInStock,@UnitsOnOrder)
END
```

You need to modify the stored procedure to meet the following new requirements:
- Insert product records as a single unit of work.
- Return error number 51000 when a product fails to insert into the database.
- If a product record insert operation fails, the product information must not be permanently written to the database.
Solution: You run the following Transact-SQL statement:

```
ALTER PROCEDURE InsertProduct
@ProductName nvarchar(100),
@UnitPrice decimal(18,2),
@UnitsInStock int,
@UnitsOnOrder int
AS
BEGIN
     SET XACT_ABORT ON
     BEGIN TRY
          BEGIN TRANSACTION
             INSERT INTO Products(ProductName,ProductPrice,ProductsInStock,ProductsOnOrder)
             VALUES (@ProductName,@UnitPrice,@UnitsInStock,@UnitsOnOrder)
          COMMIT TRANSACTION
     END TRY
     BEGIN CATCH
          IF XACT_STATE() <> 0 ROLLBACK TRANSACTION
          THROW 51000, 'The product could not be created.', 1
     END CATCH
END
```

Does the solution meet the goal?

A. Yes

B. No

Answer: B

Explanation:

A transaction is correctly defined for the INSERT INTO .VALUES statement, and if there is an error in the transaction it will be caught ant he transaction will be rolled back.

How ever, error number 51000 will not be returned, as it is only used in an IF @ERROR = 51000 statement.

Note: @@TRANCOUNT returns the number of BEGIN TRANSACTION statements that have occurred on the current connection.

References: https://msdn.microsoft.com/en-us/library/ms187967.aspx

3.Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.

After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.

You create a table named Products by running the following Transact-SQL statement:

```
CREATE TABLE Products (
     ProductID int IDENTITY(1,1) NOT NULL PRIMARY KEY,
     ProductName nvarchar(100) NULL,
     UnitPrice decimal(18, 2) NOT NULL,
     UnitsInStock int NOT NULL,
     UnitsOnOrder int NULL
)
```

You have the following stored procedure:

```
CREATE PROCEDURE InsertProduct
     @ProductName nvarchar(100),
     @UnitPrice decimal(18,2),
     @UnitsInStock int,
     @UnitsOnOrder int
AS
BEGIN
     INSERT INTO Products(ProductName,ProductPrice,ProductsInStock,ProductsOnOrder)
     VALUES (@ProductName,@UnitPrice,@UnitsInStock,@UnitsOnOrder)
END
```

You need to modify the stored procedure to meet the following new requirements:

- Insert product records as a single unit of work.
- Return error number 51000 when a product fails to insert into the database.
- If a product record insert operation fails, the product information must not be permanently written to the database.

Solution: You run the following Transact-SQL statement:

```
ALTER PROCEDURE InsertProduct
@ProductName nvarchar(100),
@UnitPrice decimal(18,2),
@UnitsInStock int,
@UnitsOnOrder int
AS
BEGIN
     SET XACT_ABORT ON
     BEGIN TRY
          BEGIN TRANSACTION
             INSERT INTO Products(ProductName,ProductPrice,ProductsInStock,ProductsOnOrder)
             VALUES (@ProductName,@UnitPrice,@UnitsInStock,@UnitsOnOrder)
          COMMIT TRANSACTION
     END TRY
     BEGIN CATCH
          IF XACT_STATE() <> 0 ROLLBACK TRANSACTION
          THROW 51000, 'The product could not be created.', 1
     END CATCH
END
```

Does the solution meet the goal?

A. Yes

B. No

Answer: A

Explanation:

If the INSERT INTO statement raises an error, the statement will be caught and an error 51000 will be thrown. In this case no records will have been inserted.

Note: You can implement error handling for the INSERT statement by specifying the statement in a TRY… CATCH construct. If an INSERT statement violates a constraint or rule, or if it has a value incompatible with the data type of the column, the statement fails and an error message is returned.

References: https://msdn.microsoft.com/en-us/library/ms174335.aspx

4.Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.

After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.

You create a table named Customer by running the following Transact-SQL statement:

```
CREATE TABLE Customer (
     CustomerID int IDENTITY(1,1) PRIMARY KEY,
     FirstName varchar(50) NULL,
     LastName varchar(50) NOT NULL,
     DateOfBirth date NOT NULL,
     CreditLimit money CHECK (CreditLimit < 10000),
     TownID int NULL REFERENCES dbo.Town(TownID),
     CreatedDate datetime DEFAULT(Getdate())
)
```

You must insert the following data into the Customer table:

Record	First name	Last name	Date of Birth	Credit limit	Town ID	Created date
Record 1	Yvonne	McKay	1984-05-25	9,000	no town details	current date and time
Record 2	Jossef	Goldberg	1995-06-03	5,500	no town details	current date and time

You need to ensure that both records are inserted or neither record is inserted.
Solution: You run the following Transact-SQL statement:

```
INSERT INTO Customer (FirstName, LastName, DateOfBirth, CreditLimit, CreatedDate)
VALUES ('Yvonne', 'McKay', '1984-05-25', 9000, GETDATE())
INSERT INTO Customer (FirstName, LastName, DateOfBirth, CreditLimit, CreatedDate)
VALUES ('Jossef', 'Goldberg', '1995-06-03', 5500, GETDATE())
GO
```

Does the solution meet the goal?
A. Yes
B. No
Answer: B
Explanation:
As there are two separate INSERT INTO statements we cannot ensure that both or neither records are inserted.

5.Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.
After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.
You create a table named Customer by running the following Transact-SQL statement:

```
CREATE TABLE Customer (
     CustomerID int IDENTITY(1,1) PRIMARY KEY,
     FirstName varchar(50) NULL,
     LastName varchar(50) NOT NULL,
     DateOfBirth date NOT NULL,
     CreditLimit money CHECK (CreditLimit < 10000),
     TownID int NULL REFERENCES dbo.Town(TownID),
     CreatedDate datetime DEFAULT(Getdate())
)
```

You must insert the following data into the Customer table:

Record	First name	Last name	Date of Birth	Credit limit	Town ID	Created date
Record 1	Yvonne	McKay	1984-05-25	9,000	no town details	current date and time
Record 2	Jossef	Goldberg	1995-06-03	5,500	no town details	current date and time

You need to ensure that both records are inserted or neither record is inserted.

Solution: You run the following Transact-SQL statement:

```
INSERT INTO Customer (FirstName, LastName, DateOfBirth, CreditLimit, CreatedDate)
VALUES ('Yvonne', 'McKay', '1984-05-25', 9000, GETDATE())
INSERT INTO Customer (FirstName, LastName, DateOfBirth, CreditLimit, CreatedDate)
VALUES ('Jossef', 'Goldberg', '1995-06-03', 5500, GETDATE())
GO
```

Does the solution meet the goal?

A. Yes

B. No

Answer: B

Explanation:

As there are two separate INSERT INTO statements we cannot ensure that both or neither records are inserted.