

Unit 5 Research Project

Eddie S. Jackson

Kaplan University

IT526: SQL Query Design

Jon Walter McKeeby, DSc, MBA

07/22/2014

Unit 5 Research Project

Part 1. Questions (50 points)

Each answer should be about 100 to 300 words.

- a. Compare and contrast autocommit mode with explicit transaction mode. (10)

In SQL server, there are transactions. Transactions represent types of changes to a database. Specifically, there are autocommit, explicit, and implicit transactions; autocommit and explicit are briefly discussed here.

In an autocommit transaction, which also happens to be the default transaction, SQL Server commits changes right after the execution of a given statement (Microsoft, n.d.). These changes are either committed if the SQL statement is correct; or, in the case that an error is encountered, the transaction will be rolled back.

In an explicit transaction, there are also transactions that represent changes. The primary difference between an autocommit and explicitly transaction, is that in an explicit transaction you define the beginning and the end of the transaction. When defining explicit transactions, SQL uses `BEGIN TRANSACTION`, `COMMIT TRANSACTION` or `COMMIT WORK`, `ROLLBACK TRANSACTION` or `ROLLBACK WORK` (Microsoft, n.d.). Each term performs a role in defining explicit transactions. For instance, `BEGIN` initializes the beginning of an explicit transaction.

- b. Compare and contrast a DML trigger with a stored procedure. (15)

A stored procedure is a code module that is normally used for performing specific user tasks. A DML trigger, while similar to a stored procedure, is a special kind of stored procedure that can be used for auditing work, and tracking database changes and activities. Both a stored procedure and a trigger cannot span batches, meaning they must be created in one batch and then compiled into one execution plan. When contrasting DML triggers and stored procedures, there are several differences that must be considered. For example, stored procedures can be scheduled to run via jobs; however, triggers cannot be scheduled. Another difference between the two is stored procedures allow input parameters, whereas triggers cannot accept parameters as input. Finally, the use of transactions (such as `begin`, `commit`, and `rollback` transactions) can be compiled inside a stored procedure; transactions cannot be embedded inside a trigger (Sahoo, 2013).

- c. Compare and contrast a DML trigger with either a foreign key or a check constraint. Explain why use of a constraint (if possible) is preferred to use of a trigger. (15)

A constraint is a specific object that is used to maintain database integrity in tables. Examples of constraints would include foreign keys, check keys, unique, and even primary keys. Another type of database object that is connected to a table is a trigger. A trigger initiates a SQL action when

events such as INSERT, DELETE, and UPDATE statements have been executed. When considering constraints, the foreign key has been selected for further review. The foreign key constraint establishes a link to another table, and thus enforces referential integrity. Likewise, a trigger can also be used to enforce referential integrity. It is usually more practical to use a foreign key constraint over a trigger, due to the complexity of logic necessary to code triggers, and the fact that triggers can cause loops; loops are coded routines that get executed over and over again (Chapman, 2007). Foreign keys are great at linking one table to the next, whereas triggers are better at comparing past and current states in rows, and performing backup and auditing logic.

- d. Explain why READ COMMITTED isolation allows more concurrency than SERIALIZABLE. (10)

When considering the isolation database property, it is important to understand that isolation is what determines the visibility of transaction integrity to users and other parts of a database. This isolation is accomplished by “locking” data, which affects the concurrency of transactions. Concurrency refers to multiple processes that have the ability to access and change data (Momjian, n.d.). Two common isolation levels are READ COMMITTED locks and SERIALIZABLE locks. READ COMMITTED locks rows and releases when the statement is complete. SERIALIZABLE locks does not allow data to be read that has been modified but has yet to be committed by transactions. The held range lock of SERIALIZABLE prevents transactions from performing changes to rows, such as INSERTS or UPDATES, until the transaction is complete (Microsoft, n.d.). Because of the longer, locked period of time in SERIALIZABLE locks, this contributes to the lower concurrency.

Part 2. Create Routines (50 points)

- a. You want to make sure that rows in the Sales.Orders table are archived when deleted. You have created the table Sales.OrdersArchive that has the same columns / data types as Sales.Orders, plus one additional column, Archived, of type datetime, to store the date and time the row is written to the archive table. Archived has a default value of CURRENT_TIMESTAMP.

The SQL Code and executed successfully

```
21 CREATE TABLE Sales.OrderArchive
22 (
23  orderid int not null,
24   custid int null,
25   empid int not null,
26   orderdate datetime not null,
27   requireddate datetime not null,
28   shippeddate datetime null,
29   shipperid int not null,
30   freight money not null,
31   shipname nvarchar(40) not null,
32   shipaddress nvarchar(60) not null,
33   shipcity nvarchar(15) not null,
34   shipregion nvarchar(15) null,
35   shippostalcode nvarchar(10) null,
36   shipcountry nvarchar(15) not null,
37   CurrentTime datetime default CURRENT_TIMESTAMP,
38   CONSTRAINT pk_orderid2 PRIMARY KEY (orderid),
39   CONSTRAINT fk_custid2 FOREIGN KEY(custid) REFERENCES Sales.Customers(custid),
40   CONSTRAINT fk_empid2 FOREIGN KEY(empid) REFERENCES HR.Employees(empid),
41   CONSTRAINT fk_shipperid2 FOREIGN KEY(shipperid) REFERENCES Sales.Shippers(shipperid)
42 );
43
```

100 %

Messages

Command(s) completed successfully.

Because of the foreign key constraint in the Sales.OrderDetails table on the orderid column, you know you cannot delete a row from Orders without first deleting all rows with the same orderid value from OrderDetails. You create a table Sales.OrderDetailsArchive that has the same columns / data types as Sales.OrderDetails, plus the Archived column of type datetime with default value CURRENT_TIMESTAMP.

The SQL Code and executed successfully

```
7
8 CREATE TABLE Sales.OrderDetailsArchive
9 (
10  orderid int not null,
11  productid int not null,
12  unitprice money not null,
13  qty smallint not null,
14  discount numeric(4,3) not null,
15  CurrentTime datetime default CURRENT_TIMESTAMP,
16  CONSTRAINT pk_orderID PRIMARY KEY (orderid,productid)
17 );
18 GO
```

100 %

Messages

Command(s) completed successfully.

To solve the deletion problem, create an INSTEAD OF trigger, Sales.tr_ArchiveOrders, that watches the Sales.Orders table for a DELETE and instead does the following:

1. Copies all relevant rows from Sales.Orders to Sales.OrdersArchive
2. Copies all relevant rows from Sales.OrderDetails to Sales.OrderDetailsArchive
3. Deletes those rows from Sales.OrderDetails
4. Finally, deletes the relevant rows from Sales.Orders

The SQL Code and executed successfully

```
51 --The create trigger:
52 CREATE TRIGGER Sales.tr_ArchiveOrders
53 ON Sales.Orders
54 INSTEAD OF DELETE
55 AS
56 -- Copy Orders rows to OrderArchive
57 INSERT Sales.OrderArchive
58     SELECT *, CURRENT_TIMESTAMP
59     FROM Deleted
60 -- Copy [OrderDetails] rows to OrderDetailsArchive
61 INSERT Sales.OrderDetailsArchive
62     SELECT *, CURRENT_TIMESTAMP
63     FROM Sales.OrderDetails
64     WHERE OrderID IN (SELECT OrderID FROM Deleted)
65 -- Delete rows from [OrderDetails]
66 DELETE Sales.OrderDetails
67     WHERE OrderID IN (SELECT OrderID FROM Deleted)
68 -- Delete rows from Orders
69 DELETE Sales.Orders
70     WHERE OrderID IN (SELECT OrderID FROM Deleted)
71 GO
```

100 %

Messages

Command(s) completed successfully.

Trigger testing. I deleted 2 rows from the table and they ended up in Sales.OrderArchive:

	orderid	custid	empid	orderdate	requireddate	shippeddate	shipperid	freight	shipname	
▶	10248	85	5	2006-07-04 00:0...	2006-08-01 00:0...	2006-07-16 00:0...	3	32.3800	Ship to 85-B	6
	10249	79	6	2006-07-05 00:0...	2006-08-16 00:0...	2006-07-10 00:0...	1	11.6100	Ship to 79-C	L

b. In the *pubs* database, create a stored procedure that will INSERT an employee. You can develop the procedure in the same way as the procedure in Exercise 2 starting on page 486 is developed. Your finished procedure should have all relevant parameter testing. Only the finished CREATE PROC code should be given in your assignment document. Also show a call to the procedure that would insert an employee. (25)

SQL Code

```
CREATE PROCEDURE dbo.InsertEmployee
```

```

    @emp_id    AS CHAR(9)
    ,@fname    AS VARCHAR(20)
    ,@minit    AS CHAR(1)
    ,@lname    AS VARCHAR(30)
    ,@job_id   AS SMALLINT = 1
    ,@job_lvl  AS TINYINT = 10
    ,@pub_id   AS CHAR(4) = '0877'
    ,@hire_date AS datetime
AS
BEGIN -- start body of procedure
    DECLARE @ClientMessage VARCHAR(100)
    BEGIN TRY
        --Test the emp_id check constraint
        IF NOT (@emp_id LIKE
            '[A-Z][A-Z][A-Z][1-9][0-9][0-9][0-9][0-9][FM]'
            OR @emp_id LIKE
            '[A-Z]-[A-Z][1-9][0-9][0-9][0-9][0-9][FM]'
        )
        BEGIN
            SET @ClientMessage = 'empid ' + @emp_id + 'is not valid;'
            + 'See the check constraint definition.';
            THROW 50000, @ClientMessage, 0;
        END;
        --Test foreign key @job_id
        IF NOT EXISTS (SELECT 1 FROM dbo.jobs
            WHERE job_id = @job_id)

```

```
BEGIN
  SET @ClientMessage = 'job_id ' + CAST(@job_id AS VARCHAR)
    + ' is invalid; it must equal a job_id value in jobs';
  THROW 50000, @ClientMessage, 0;
END;
--Test foreign key pub_id
IF NOT EXISTS (SELECT 1 FROM dbo.publishers
  WHERE pub_id = @pub_id)
  BEGIN
    SET @ClientMessage = 'pub_id ' + @pub_id
      + ' is invalid; it must equal a pub_id value in publishers';
    THROW 50000, @ClientMessage, 0;
  END;
--Perform the insert
INSERT dbo.employee (emp_id, fname, minit, lname
  , job_id, job_lvl, pub_id, hire_date)
VALUES (@emp_id, @fname, @minit, @lname
  , @job_id, @job_lvl, @pub_id, @hire_date)
END TRY
BEGIN CATCH
  THROW;
END CATCH;
END; -- end procedure
```

The Call – with the entry in the table

EmpID	Name	Sex	JobID	Job	Level	PubID	HireDate
EJS12345M	Eddie	S	Jackson	1	10	0877	2014-07-22 00:00:00
KFJ64308F	Karin	F	Josephs	14	100	0736	1992-10-17 00:00:00
MGK44605M	Matti	G	Karttunen	6	220	0736	1994-05-01 00:00:00
POK93028M	Pirkko	O	Koskitalo	10	80	9999	1993-11-29 00:00:00
JYL26161F	Janine	Y	Labrune	5	172	9901	1991-05-26 00:00:00
M-L67958F	Maria		Larsson	7	135	1389	1992-03-27 00:00:00


```
SQLQuery1.sql - M...C\Mothership (54))* X
145 EXEC dbo.InsertEmployee
146     @emp_id = 'EJS12345M'
147     ,@fname = 'Eddie'
148     ,@minit = 'S'
149     ,@lname = 'Jackson'
150     ,@job_id = '1'
151     ,@job_lvl = '10'
152     ,@pub_id = '0877'
153     ,@hire_date = '20140722'
154 GO
```

100 %

Messages

(1 row(s) affected)

100 %

Query executed successfully. | MOTHERSHIP-PC\SQLEXPRESS (1... | Mothership-PC\Mothersh...

References

- Ben-Gan, I., Sarka, D., & Talmage, R. (2013). *Training Kit (Exam 70-461): Querying Microsoft® SQL Server® 2012*. Sebastopol, CA: O'Reilly Media, Inc.
- Chapman, Tim. (06/04/2007). Comparing SQL Server constraints and DML triggers. Retrieved from <http://www.techrepublic.com/blog/software-engineer/comparing-sql-server-constraints-and-dml-triggers>
- Microsoft. (n.d.). Autocommit Transactions. Retrieved from [http://technet.microsoft.com/en-us/library/ms187878\(v=SQL.105\).aspx](http://technet.microsoft.com/en-us/library/ms187878(v=SQL.105).aspx)
- Microsoft. (n.d.). Explicit Transactions. Retrieved from [http://technet.microsoft.com/en-us/library/ms175127\(v=SQL.105\).aspx](http://technet.microsoft.com/en-us/library/ms175127(v=SQL.105).aspx)
- Microsoft. (n.d.). Implicit Transactions. Retrieved from [http://technet.microsoft.com/en-us/library/ms188317\(v=sql.105\).aspx](http://technet.microsoft.com/en-us/library/ms188317(v=sql.105).aspx)
- Microsoft. (n.d.). Set Transaction Isolation Level (Transact-SQL). Retrieved from <http://msdn.microsoft.com/en-us/library/ms173763.aspx>
- Microsoft. (n.d.). SQL Stored Procedures. Retrieved from [http://technet.microsoft.com/en-us/library/aa174792\(v=sql.80\).aspx](http://technet.microsoft.com/en-us/library/aa174792(v=sql.80).aspx)
- Momjian, Bruce. (n.d.). Read Committed and Serializable Isolation Levels. Retrieved from http://momjian.us/main/writings/pgsql/aw_pgsql_book/node101.html
- Sahoo, Kulamani. (07/22/2013). Differences between a stored procedure and a trigger. Retrieved from <http://www.codeproject.com/Tips/624566/Differences-between-a-Stored-Procedure-and-a-Trigg>