Luis Valderrama

April 26, 2021

IT FDN 130 A

Module 04

Processing Data in SQL

# Intro

During the course of the fourth module, I learned the basic SQL Statements and Transactions available to define the data structure and manipulate the data while protecting its integrity. This document addresses concepts and practical uses of SQL Statements and Transactions compounded with Error Handling codes as demonstrated through the database created for this assignment, Assignment04DB\_LuisValderrama.

# Basic SQL Statements

*“A SQL statement is an atomic* ***unit of work and either completely succeeds or completely fails. A SQL statement is a set of instruction that consists of identifiers, parameters, variables, names, data types, and SQL reserved words that compiles successfully.*** *Perhaps the most important is the SELECT that retrieves rows from the database and enables the selection of one or many rows or columns from one or many tables in SQL Server”.* [*https://docs.microsoft.com/en-us/sql/t-sql/statements/statements?view=sql-server-ver15*](https://docs.microsoft.com/en-us/sql/t-sql/statements/statements?view=sql-server-ver15)*,* (2020) (external site)

The **SELECT** statement is considered a Data Manipulation Language (DML), and is used to retrieve rows from the database and enables the selection of one or many rows or columns from one or many tables in SQL server and **displays the data in a readable format.** The SELECT statement follows a consistent and specific format. If an asterisk (\*) is placed after SELECT, the command will return all attributes, and rows in the table. The example presented below shows two ways to use the SELECT statement. One uses (\*) in order to display all records in the Inventories table, while the second SELECT statement is specific to the ProductName attribute of the Products table. (figure 1.1)

SELECT \* FROM Inventories;

SELECT ProductName FROM Products;

GO

***Figure 1.1: Example of the SELECT statement to return readable data.***

The **CREATE** statement is considered a Data Definition Language (DDL), and is **used to establish a new database, table, index, or stored procedure**. The CREATE statement in SQL establishes a component in a RDBMS. The example presented below shows CREATE statement plus TABLE. (figure 1.2)

CREATE TABLE Categories

([CategoryID] [int] IDENTITY(1,1) NOT NULL

,[CategoryName] [nvarchar](100) NOT NULL

);

go

***Figure 1.2: Example of Creating a table using the CREATE statement.***

The **UPDATE** statement is considered a DML, and is **used to change existing data in a table or view** in SQL Server. The UPDATE statement is followed by SET, column name and new value, followed by WHERE clause, column reference and value. The example presented below shows a basic UPDATE statement used to rename the existing CategoryName to ‘Drinks’ for all items with CategoryID of ‘1’.

UPDATE Categories

SET CategoryName = 'Drinks'

WHERE CategoryID = 1

GO

***Figure 1.3: Example of using the UPDATE statement to revise existing information.***

The **DELETE** statement is also considered a DML, and is **used to remove existing records from a table**. It is recommended to use a key word such as FROM between DELETE and the target table. You may also use the conditional clause, WHERE, in order to filter the data on the targeted rows. The example presented below shows the DELETE statement used with the conditional WHERE clause in order to target all items with ProductID of ‘3’ listed in the Inventories table.

DELETE FROM Inventories

WHERE ProductID = 3;

GO

***Figure 1.4: Example of using the DELETE statement to remove data.***

# SQL Transactions

*“A transaction is a unit of work that is performed against a database.* ***Transactions are units or sequences of work accomplished in a logical order, whether in a manual fashion by a user or automatically by some sort of a database program****.* ***A transaction is the propagation of one or more changes to the database****. For example, if you are creating a record or updating a record or deleting a record from the table, then you are performing a transaction on that table. It is important to control these transactions to ensure the data integrity and to handle database errors”.* [*https://www.tutorialspoint.com/sql/sql-transactions.htm*](https://www.tutorialspoint.com/sql/sql-transactions.htm)*,* (2021), (external link)

The purposes of a database transactions are:

1. Provide reliable units of work that allow correct recovery from failures and keep a database consistent even in system failure.
2. Provide isolation between programs accessing the database concurrently.

Each transaction is started with BEGIN TRANSACTION statement and ended with a COMMIT or ROLLBACK. A practical example of an Advanced SQL Transaction best practices is found in the database created for this assignment, Assignment04DB\_LuisValderrama, where the BEGIN TRANSACTION is placed before the transaction code which in this case is to add the value ‘Beverage’ to the Categories table as CategoryName. The transaction is then followed by COMMIT TRANSACTION. Also, this same example shows the TRY-CATCH block code to handle errors. Note, the PRINT ERROR MESSAGE is customizable to display a specific message whenever an error occurs. The example below is set up to display the standard RDMS error message by leaving the () blank. Additionally, the example below uses @@TRANCOUNT > 0 which returns the number of BEGIN TRANSACTION statements that have occurred on the current connection.

BEGIN TRY

BEGIN TRAN

--TRANSACTION CODE--

INSERT INTO Categories (CategoryName)

VALUES ('Beverages');

COMMIT TRAN

END TRY

BEGIN CATCH

IF @@TRANCOUNT > 0

BEGIN

ROLLBACK TRAN

END

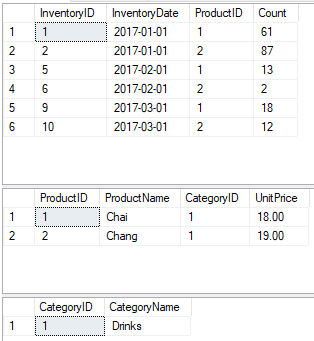
PRINT ERROR\_MESSAGE()

END CATCH

GO

***Figure 2.1: Example using advance transaction with error handling codes.***

The image presented below is the conclusion of several entries made to tables in the database created for this assignment, followed by updates, and deletion of data, all to which were written with SQL Statements such as INSERT INTO, UPDATE, DELETE Statements as well as BEGIN TRANSACTION, COMMIT TRANSACTION Transactions and TRY-CATCH error handling codes. (figure 2.2). See <Assignment04DB_LuisValderrama.sql>, (2021) (SQL Script) for greater details of the SQL script.



***Figure 2.2: End result of assignment04 after applying the SQL statements, transactions and error handling.***

## Identity, @@Identity, and Ident\_Current () in SQL

**INDENTITY**(Property)(Transact-SQL) *“****Creates an identity column in a table.*** *This property is used with the CREATE TABLE and ALTER TABLE Transact-SQL statements. Identity columns can be used for generating key values. The identity property on a column guarantees: 1) Each new value is generated based on the current seed & increment, and 2) Each new value for a particular transaction is different from other concurrent transactions on the table.”,* [*https://docs.microsoft.com/en-us/sql/t-sql/statements/create-table-transact-sql-identity-property?view=sql-server-ver15*](https://docs.microsoft.com/en-us/sql/t-sql/statements/create-table-transact-sql-identity-property?view=sql-server-ver15)*,* (2017) (external site). The example presented below shows the IDENTITY property used in the creating a table transaction in order to establish identity for the CategoryID attribute. (figure 3.1)

Create Table Categories

([CategoryID] [int] IDENTITY(1,1) NOT NULL

,[CategoryName] [nvarchar](100) NOT NULL

);

go

***Figure 3.1: Example using IDENTITY (property) in a basic transaction***

**@@IDENTITY** (Transact-SQL) **“*Is a system function that returns the last-inserted identity value****. After an INSERT, SELECT INTO, or bulk copy statement is completed, @@IDENTITY contains the last identity value that is generated by the statement.* [*https://docs.microsoft.com/en-us/sql/t-sql/functions/identity-transact-sql?view=sql-server-ver15*](https://docs.microsoft.com/en-us/sql/t-sql/functions/identity-transact-sql?view=sql-server-ver15)*,* (2017) (external).Note: The @@IDENTITY track the last-inserted identity value **on a given connection.** The example presented below (contained in the link above) inserts a row into a table with an identity column (LocationID) and uses @@IDENTITY to display the identity value used in the new row*.* (figure 3.2)

USE AdventureWorks2012;

GO

--Display the value of LocationID in the last row in the table.

SELECT MAX(LocationID) FROM Production.Location;

GO

INSERT INTO Production.Location (Name, CostRate, Availability, ModifiedDate)

VALUES ('Damaged Goods', 5, 2.5, GETDATE());

GO

SELECT @@IDENTITY AS 'Identity';

GO

--Display the value of LocationID of the newly inserted row.

SELECT MAX(LocationID) FROM Production.Location;

GO

***Figure 3.2:*** ***Example using @@IDENTITY in a basic transaction.***

**IDENT\_CURRENT** (Transact-SQL) *“****Returns the last identity value generated for a specified table or view****. The last identity value generated can be for any session and any scope.* [*https://docs.microsoft.com/en-us/sql/t-sql/functions/ident-current-transact-sql?view=sql-server-ver15*](https://docs.microsoft.com/en-us/sql/t-sql/functions/ident-current-transact-sql?view=sql-server-ver15)*,* (2017) (external site). The example presented below (contained in the link above) returns the last identity value generated for the Person.Address table in the AdventureWorks2012 database. (figure 3.3)

USE AdventureWorks2012;

GO

SELECT IDENT\_CURRENT ('Person.Address') AS Current\_Identity;

GO

***Figure 3.3:*** ***Example using IDENT\_CURRENT in a basic transaction.***

# Summary

To recap, the fourth module taught me the concepts of the basic Statements and Transactions available in the MS SQL RDMS to define the data structure and manipulate the data while protecting its integrity. The database created for this assignment, Assignment04DB\_LuisValderrama, applies functionalities learned throughout modules one through three which includes creating databases and tables, adding constraints, plus updating and deleting data using SQL Statements and Transactions compounded with error handling codes.