Jennifer DeVore

23 March 2020

CIS 435L

**Unit 4 Lab Assignment: Working with views, functions, and stored procedures**

**Terms:**

1. **View**

According to Tutorials Point,” A view is nothing more than a SQL statement that is stored in the database with an associated name. A view is actually a composition of a table in the form of a predefined SQL query (Tutorials Point, 2020).”

1. **CREATE VIEW**

This is how you start a statement to actually create a new view – it can be from one table, multiple tables, and even other views.



1. **Updatable View**

A view can only be updated under specific conditions. According to Tutorials Point, a view can only be updated under the following, “the SELECT clause may not contain the keyword DISTINCT, the SELECT clause may not contain summary functions, the SELECT clause may not contain set functions, the SELECT clause may not contain set operators, the SELECT clause may not contain an ORDER BY clause, the FROM clause may not contain multiple tables, the WHERE clause may not contain subqueries, the query may not contain GROUP BY or HAVING, calculated columns may not be updated, all NOT NULL columns from the base table must be included in the view in order for the INSERT query to function (Tutorials Point, 2020).”

1. **Catalog View**

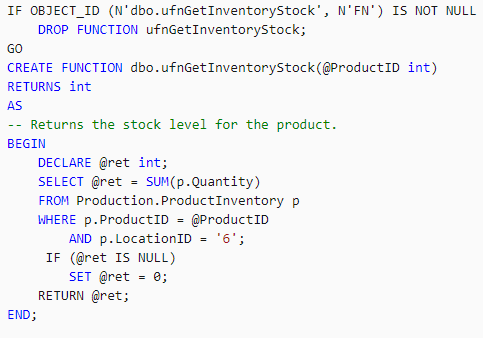
According to Microsoft, “Catalog views return information that is used by the SQL Server Database Engine. We recommend that you use catalog views because they are the most general interface to the catalog metadata and provide the most efficient way to obtain, transform, and present customized forms of this information. All user-available catalog metadata is exposed through catalog views (Microsoft, 2016).” Catalog views can return rows from things other than tables too – such as views and stored procedures. Catalog views will not have information about backups, db maintenance, or replication data.

1. **User-defined Function**

A user-defined function behaves similarly to those functions and methods we created in programming languages. They’re modular and allow for faster functionality because you don’t need to continuously rewrite the code to run them. According to Microsoft, “Like functions in programming languages, SQL Server user-defined functions are routines that accept parameters, perform an action, such as a complex calculation, and return the result of that action as a value. The return value can either be a single scalar value or a result set (Microsoft, 2016).”

1. **Scalar-value Function**

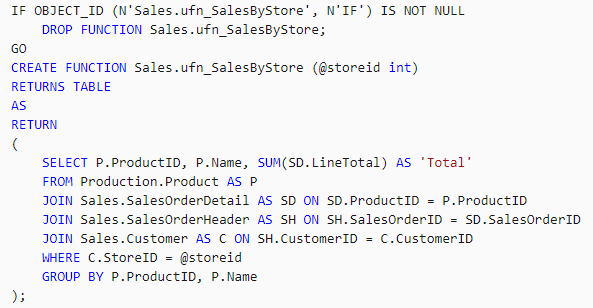
These are also similar to programming language methods and functions that return only a single data value of the type that is defined in the RETURNS clause. According to Microsoft, “For a multistatement scalar function, the function body can contain a series of Transact-SQL statements that return the single value. The return type can be any data type except text, ntext, image, cursor, and timestamp (Microsoft, 2016).”



(Microsoft, 2016).

1. **Table-valued Function**

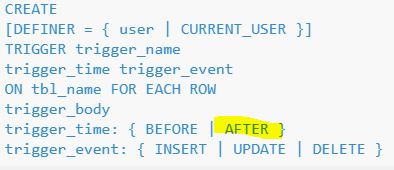
Same as the other types of functions except these return a table data type. When coding inline table-valued functions there is no function body, just the ,” able is the result set of a single SELECT statement (Microsoft, 2016).”



(Microsoft, 2016).

1. **AFTER Trigger**

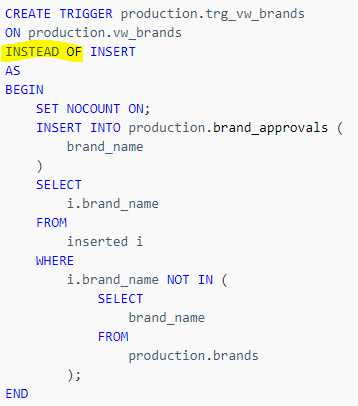
This trigger is executed AFTER a specific event (whether UPDATE, DELETE, or INSERT).



(W3Resources, 2020).

1. **INSTEAD OF Trigger**

This trigger allows the user to skip INSERT, UPDATE, or DELETE statements and execute other parts of the statements that are defined within the trigger. According to SQL Server Tutorial, “The actual insert, delete, or update operation does not occur at all. In other words, an INSTEAD OF trigger skips a DML statement and execute other statements (SQL Server Tutorial, 2020).”



(SQL Server Tutorial, 2020).

1. **Stored Procedure**

According to Murach and Syverson, “Stored procedures give the SQL programmer control over who accesses the data- base and how. Since some application programmers don’t have the expertise to write certain types of complex SQL queries, stored procedures can simplify their use of the database (Murach & Syverson, 2012).”

|  |  |
| --- | --- |
| **Exercise 1** | **VIEWS**  **Create a VIEW using at least four columns from a table in the AdventureWorks2014 database.**  **SCREENSHOT BELOW AFTER CREATION SHOWS VIEW IN OBJECT EXPLORER**    **List the name of two VIEWs already defined in the AdventureWorks2014 database.**  **Dbo.EmployeePersons**  **HumanResources.vEmployee** |
| **Exercise 2** | **User-Defined Functions  The AdventureWorks2014 database includes several user-defined functions.  Run the code below and explain each function's performance.**  **Code Section 1**  **USE AdventureWorks2014; GO SELECT CustomerID,('AW' + dbo.ufnLeadingZeros(CustomerID))     AS GenerateAccountNumber FROM Sales.Customer ORDER BY CustomerID; GO**  **This code was executed and ran in less that one second. It returned 19,820 rows in less than a second.**    **Code Section 2**  **USE AdventureWorks2014; GO SELECT ProductID, ListPrice, dbo.ufnGetProductDealerPrice(ProductID, StartDate) AS DealerPrice,    StartDate, EndDate FROM Production.ProductListPriceHistory WHERE ListPrice > 3000.00 ORDER BY ProductID, StartDate;**  **This code returned 13 rows and was executed in less than a single second as well.** |
| **Exercise 3** | **Stored Procedures**  **The AdventureWorks2014 database includes several stored procedures.  Run the code below and explain what each stored procedure returns.**  **Code Section 1**  **USE AdventureWorks2014; GO EXEC dbo.uspGetManagerEmployees 100;**  **It returns the RecursionLevel, OrganizationNode, ManagerFirstName, ManagerLastName, BusinessEntityID, FirstName, and LastName where the BusinessEntityID is 100. It returns the manager and employee name.**    **Code Section 2**  **USE AdventureWorks2014; GO DECLARE @CheckDate datetime; SET @CheckDate = GETDATE(); EXEC dbo.uspGetWhereUsedProductID 819, @CheckDate;**  **This code passes the function value as a variable. This code returns the ProductAssemblyID, ComponentID, ComponentDesc, TotalQuantity, StandardCost, ListPrice, BOMLevel, and RecursionLevel where the ComponentID equals 819.** |

# References

Microsoft. (2016, May 2). *System Catalog Views (Transact-SQL).* Retrieved from Microsoft: https://docs.microsoft.com/en-us/sql/relational-databases/system-catalog-views/catalog-views-transact-sql?view=sql-server-ver15

Microsoft. (2016, August 5). *User-Defined Functions.* Retrieved from Microsoft: https://docs.microsoft.com/en-us/sql/relational-databases/user-defined-functions/user-defined-functions?view=sql-server-ver15

Murach, J., & Syverson, B. (2012). *Murach's SQL Server 2012 for Developers.* Boston: Mike Murach & Associates, Inc.

SQL Server Tutorial. (2020, March 23). *SQL Server INSTEAD OF Trigger.* Retrieved from SQL Server Tutorial.Net: https://www.sqlservertutorial.net/sql-server-triggers/sql-server-instead-of-trigger/

Tutorials Point. (2020, March 23). *SQL - Using Views.* Retrieved from Tutorials Point: https://www.tutorialspoint.com/sql/sql-using-views.htm

W3Resources. (2020, February 26). *MySQL Triggers.* Retrieved from W3Resources: https://www.w3resource.com/mysql/mysql-triggers.php