Luis Valderrama

May 7, 2021

IT FDN 130 A

Module 05

SQL Joins

# Intro

During the course of the fifth module, I learned the basic concepts and practical uses of Joins available in the MS SQL RDMS to facilitate the combination of data presented in multiple tables. This document addresses the uses, types, similarities and differences of Joins. This document also presents examples used in the database created for this assignment, Assignment05DB\_LuisValderrama.

# SQL Join

*“By using joins, you can* ***retrieve data from two or more tables based on logical relationships between the tables****. Joins indicate how SQL Server should use data from one table to select the rows in another table.”* [*https://docs.microsoft.com/en-us/sql/relational-databases/performance/joins?view=sql-server-ver15*](https://docs.microsoft.com/en-us/sql/relational-databases/performance/joins?view=sql-server-ver15)*, (2019) (external).* **The data analyst would utilize a join(s) to extract information from multiple tables by “combining” or Joining data applicable tables, based on attributes and rows contained in the RDMS.** Joins are used with SELECT statement and the Transact-SQL Syntax:

* Inner Join
* Left (Outer) Join
* Right (Outer) Join
* Full (Outer) Join
* Cross Join.

The example presented below on (figure 1.1) created for [Assignment05DB\_LuisValderrama](Assignment05DB_LuisValderrama.sql), (2021) (SQL Script) shows two tables, Categories and Products, combined using Inner Join. This is accomplished by using the Primary Key (PK) that uniquely identifies each record in the Categories table which in this case it is CategoryID, and linking it with the Foreign Key (FK), also CategoryID in the Products table, that acts as a cross-reference between the two tables thus referencing the Primary Key.

SELECT CategoryName, ProductName, UnitPrice

FROM Categories

INNER JOIN Products

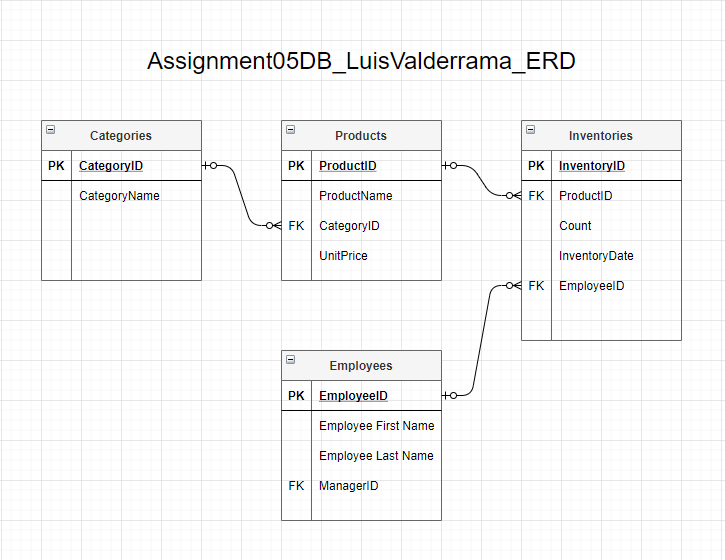
ON Categories.CategoryID = Products.CategoryID

ORDER BY CategoryName, ProductName ASC;

GO

***Figure 1.1: Example of a simple Inner Join of two tables.***

Note that it is important to have a good understanding of the database structure and all the tables in the database and know which of the attributes are Primary Keys and which are Foreign Keys. In the example created for my assignment I constructed an Entity Relational Diagram to help me visualize the tables, Primary Key, and Foreign Key. (figure 1.2)



***Figure 1.2: Entity Relationship Diagrams created for Assignment05DB\_LuisValderrama.***

## Differences and similarities between an Inner, Outer and Cross Join

The most **common similarity between an Inner Join, Outer and Cross join is that they are all designed to combine multiple tables.** Whereas, the primary difference is **how the tables are combined**. In other words:

**Inner Joins**. Select all records from Table 1 and Table 2, where the join condition is met. (figure 2.1)

**Outer Joins**. Select all records from one table along with records from the other table for which the join condition is met. For example, in (figure 2.1) below a **Left Join** would return all records from Table 1 along with records with met conditions from Table 2. Vice versa, a **Right Join** would return all records from Table 2 along with records with met conditions from Table 1. By design, the Outer Join will display more records than an Inner Join would.

**Full Joins.** Select all records from Tables 1 and 2 regardless of whether the join condition is met or not. (figure 2.1)

**Cross Join.** Returns products of rows from the row sets in the join. Simply said, it will combine each row from the first set of rows with each row from the second set of rows. **Caution** should be exercised as this operation may lead to a large display of data.

## 

***Figure 2.1: Visualization of Inner and Outer Joins.***

Another difference is that “*Inner joins* ***can be specified in either the FROM or WHERE clauses****. Outer joins and cross joins* ***can be specified in the FROM clause only****. The join conditions combine with the WHERE and HAVING search conditions to control the rows that are selected from the base tables referenced in the FROM clause.”* [*https://docs.microsoft.com/en-us/sql/relational-databases/performance/joins?view=sql-server-ver15*](https://docs.microsoft.com/en-us/sql/relational-databases/performance/joins?view=sql-server-ver15)*, (2019) (external)*

## Self-Join

*“A self-join is a Structured Query Language (SQL)* ***statement in which a table is joined with itself,*** *an inner join is performed on a single table with itself, particularly in cases when comparisons have to be made between the records of the same table to determine a relationship or in the cases when the table has a FOREIGN KEY which references its own PRIMARY”.* [*https://www.educba.com/sql-self-join/*](%20https:/www.educba.com/sql-self-join/), (2021)(external site)

The example presented below on (figure 2.2) created for [Assignment05DB\_LuisValderrama](Assignment05DB_LuisValderrama.sql), (2021) (SQL Script) shows the Employees table joined with itself with an INNER JOIN. This is accomplished by connecting the table using its own Primary Key and Foreign Key. In this case, ManagerID (FK) and EmployeeID (PK). It is recommended to use aliases for the attributes in order to make the results easier to interpret. For example, “Employee” and “Manager” are the aliases chosen for the combination of the employee’s first and last name by way of a function that will ultimately display both, the managers and Employees’ full name in one column as shown on (figure 2.3). As a point of comparison also shown on the same figure you can see how the data is presented in the table prior to the application of the aliases and functions mentioned above. Other aliases used in this example are the “Mgr” and “Emp” used primarily for the Employees table in order to differentiate one table from the other by way of creating a parallel version of the same table.

SELECT

[Manager] = IIF (ISNULL(Mgr.EmployeeID, 0) = 0, 'Manager', Mgr.EmployeeFirstName + ' ' + Mgr.EmployeeLastName),

[Employee] = Emp.EmployeeFirstName + ' ' + Emp.EmployeeLastName

FROM Employees AS Emp

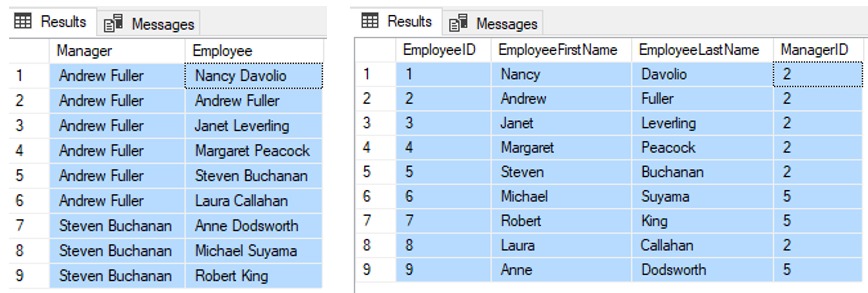
INNER JOIN Employees AS Mgr

ON Emp.ManagerID = Mgr.EmployeeID

ORDER BY 'Manager';

GO

***Figure 2.2: Example of a Self-Join created for Assignment05DB\_LuisValderrama.***



***Figure 2.3: Results on the left image uses aliases and functions to make the data easier to read.***

# Summary

To recap, the fifth module, taught me the basic concepts and practical uses of Joins available in the MS SQL RDMS to facilitate the combination of data presented in multiple tables within a database. The database created for this assignment, Assignment05DB\_LuisValderrama, applies the functionalities learned throughout prior modules and the practical application of SQL Joins.