Luis Valderrama

April 17, 2021

IT FDN 130 A

Module 02

Common Database Options

# Intro

During the course of the second module, I learned the common options and tools available to design and define the structure of a database, as well as protect its integrity. This document addresses concepts of Constraints, Abstraction Layers, Entity Relationship Diagrams (ERD) and Meta-Data, and their practical use as demonstrated through the database created for this assignment. The DB created for this document is Assignment02DB\_LuisValderrama and consists of three basic tables.

# Database Constraints

# *“Constraints define rules regarding the values allowed in columns and are the standard mechanism for enforcing integrity”* [*https://docs.microsoft.com/en-us/previous-versions/sql/sql-server-2008-r2/ms189862(v=sql.105)?redirectedfrom=MSDN*](https://docs.microsoft.com/en-us/previous-versions/sql/sql-server-2008-r2/ms189862(v=sql.105)?redirectedfrom=MSDN)*, (2012) (external site).* The practical use of Constraints on an SQL Script is that it allows the user to set acceptable values for an attribute and restrict undefined values. The types of Constraints used in my database example are: Not Null, Check, Unique, Primary Key and Foreign Key. Another integrity management tool used in my database example is Default which specifies the value used in the column if no value is previously specified for the column when the record is inserted. In the example shown below, (figure 1.1), the Inventory table had already been created, then altered to incorporate the constraints by using SQL Commands such as ALTER, ADD CONSTRAINTS, followed by the constraint name, type and the column to which the constraint is applied to. In cases such as FOREIGN KEY a reference must be made to the table where the matching attribute is located. It is important to note that the table being referenced can be the same table. However, more often than not the references are made to separate tables. Similar process was followed for the Categories and Products tables.

ALTER TABLE Inventories

ADD CONSTRAINT PK\_InventoryID PRIMARY KEY CLUSTERED (InventoryID);

GO

ALTER TABLE Inventories

ADD CONSTRAINT FK\_ProductID FOREIGN KEY (ProductID) REFERENCES Products (ProductID);

GO

ALTER TABLE Inventories

ADD CONSTRAINT DF\_InventoryDate DEFAULT GETDATE () FOR InventoryDate;

GO

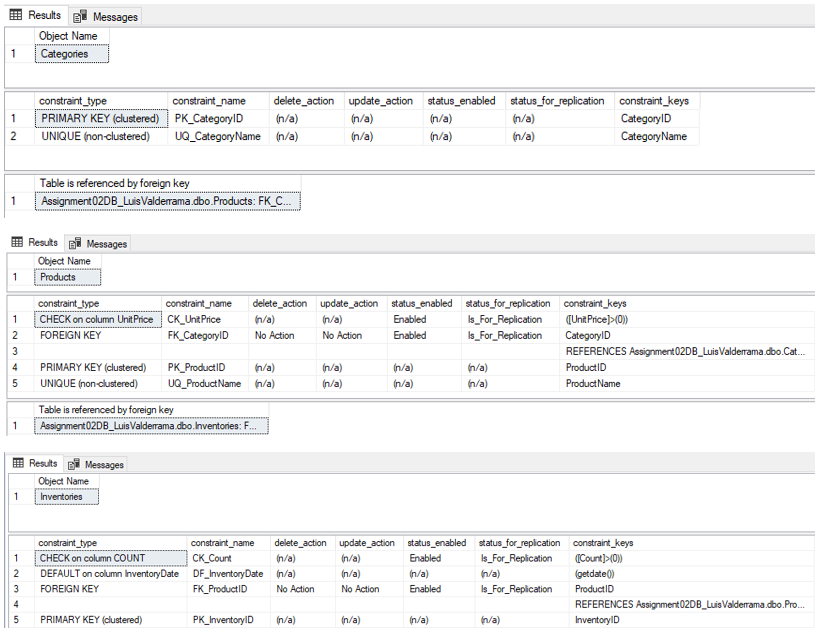
ALTER TABLE Inventories

ADD CONSTRAINT CK\_Count CHECK (COUNT > 0);

GO

***Figure 1.1: Example of altering, adding constraints to the existing Inventories table.***

Below is a screen capture of all constraints added to the tables created for these assignment. (figure 1.2)



***Figure 1.2: Assignment02DB\_LuisValderrama\_Table Column Constraints.***

# Abstraction Layers (DBAL or DAL)

The practical use of the abstraction layers in simple terms is a set of functions developed with the purpose of providing clear and user-friendly interface which unifies the communication between the application and databases. The design of the database should allow for the structure to evolve with minimal disruption to the users. The RDMS, SM SQL offers DB abstraction layers such as View, Functions and Stored Procedures to which are all conceptually similar in enabling the user to save SELECT statements, yet, Functions allows for filtering parameters to be included in the definition while Stored Procedures also allows for INSERT, UPDATE, and DELETE statements. For the purpose of my DB example created for this assignment I used the View option of Abstraction Layer. The example displayed below shows the SQL script to CREATE VIEW of one of the tables created for this assignment, (figure 2.1). Similar process was followed for the Categories and Products tables.

CREATE VIEW vInventories

AS SELECT

InventoryID,

ProductID,

InventoryDate,

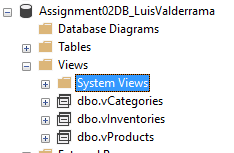
Count

FROM Inventories

GO

SELECT \* FROM vInventories

***Figure 2.1: CREATE VIEW script example for the Inventory table.***

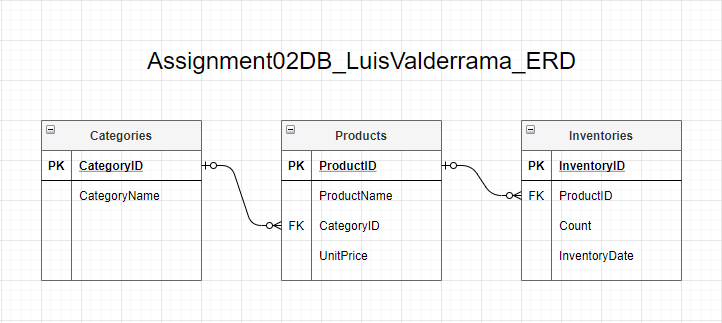


***Figure 2.2: View of Categories, Inventories and Products tables.***

# ERD and Meta-Data

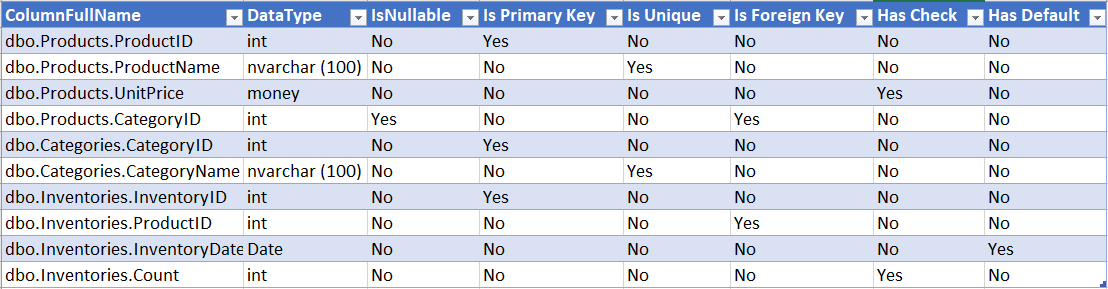
Two other important DB designing tools are ERD and Meta-Data. The application of these tools become more critical as the complexity level of the database increases. However, it is best practices to always develop an ERD and a Meta-Data for all database projects regardless of the complexity levels as it serves as documentation and visualization of the database.

“*An entity relationship diagram (ERD) shows the relationships of entity sets stored in a database. An entity in this context is an object, a component of data. An entity set is a collection of similar entities. These entities can have attributes that define its properties. By defining the entities, their attributes, and showing the relationships between them, an ER diagram illustrates the logical structure of databases”* <https://www.smartdraw.com/entity-relationship-diagram/>, (2021) (external site).I designed a basic ERD not only to document the structure, and relationships of the database created for this assignment but it also helps me visualize the database before creating it MS SQL. (figure3.1)



***Figure 3.1: Visualization of Assignment02DB\_LuisValderrama\_ERD***

*“A metabase (sometimes called a metadatabase or metadata repository) is a database for storing metadata (data that describes data) for a specific purpose”.* [*https://searchoracle.techtarget.com/definition/metabase*](https://searchoracle.techtarget.com/definition/metabase)*,* (2021) (external site). I designed a Meta-Data for the database created for this assignment primarily to document and describe the tables’ attributes, data types, and their corresponding constraints. (figure 3.2)



***Figure 3.2: Visualization of Assignment02DB\_LuisValderrama\_Metadata***

# Designing a Simple Database

Based on what I have learned throughout the course of Modules 1 and 2, I would outline the steps to design a simple database as follows:

1. Identify the objects needed in the database
   1. What data are we trying to store?
2. Identify tables and attributes
   1. How many tables are needed to store data? Consider normalization rules by asking:
      1. Identify candidate keys
      2. Identify multi-part values
      3. Identify multi-value fields
3. Define tables and database structure.
   1. Define naming conventions
      1. Are the table names comprehensive and predictable?
      2. Are the columns’ names comprehensive, consistent and predictable?
      3. Are the data types comprehensive, consistent and predictable?
      4. Is there a Primary Key candidate?
   2. Identify Constraints (Primary and Foreign Key, Unique, Check, Nullability)
   3. Visualize the database structure by designing an ERD
   4. Develop Meta-data
4. Identify extraction layers (View, Functions, Store Procedures)
5. The relational database should include a view, or store procedure for managing the data

# Summary

To recap, the second module, I learned the common options and tools available to design, and define the structure of a database, and protect its integrity. The DB created for this assignment gave me the experience to update tables with Constraints, design the database using tools such as ERD and Meta-Data, and their practical use. In addition to learning the basics to design a database.