Jonathan Owen
April 26, 2022
IT FDN 130 A Sp 22
Assignment 02

Database Design and Constraints

Introduction

The second module in the FDN introduced database design and the tools to ensure all desired data and its type is captured in the appropriate tables and that the relations between tables is understood before a single line of code is written. This paper will describe an Entity Relationship Diagram (ERD) or a meta-data worksheet which are tools that may be used to design a new or understand an existing database. In addition to the design process, this paper will cover constraints and abstractions layers that can be used to maintain database data integrity.

Database Constraints

Constraints are rules that may be placed on a table column in order to restrict the possible values which may be entered to a range, specific data set, or so that it is unique within the column. By applying constraints to columns the integrity of the data held within a database may be maintained. This is because the database will display an error and prevent data entry if a user attempts to enter a value that does not comply with the defined constraints.

Types of Constraints:

> `NULL` and `NOT NULL`
> By default, all columns have the NULL constraint applied. This means that a value does not need to be specified when inserting a value into a table. Similarly the NOT NULL constraint forces a value to be entered in the column when inserting data into the table. The NOT NULL constraint essentially makes a value required.
>
> `CHECK`
> The CHECK constraint is used when a value must meet a defined condition or conditions. Only values that meet the defined conditions are allowed to be inserted into a table. An example of a CHECK condition would be if an integer must be greater than 10.
>
> `UNIQUE`
> The UNIQUE constraint is used when no two values in a column may be the same. An example of this might be if a table was used to store user ids for a website. If more than two users shared the same id, problems and confusion would occur.
>
> `PRIMARY KEY`
> The PRIMARY KEY constraint is assigned to a single column or a set of columns and is used to uniquely identify a row of data within a table. Primary keys are most important when creating relationships between tables. A column with the PRIMARY KEY constraint inherently adds the UNIQUE and NOT NULL constraints to the column.

> `FOREIGN KEY`
> The FOREIGN KEY constraint is used in the creation of relations between tables. A column with this constraint is restricted to being a PRIMARY KEY of the table to which the association has been made.

Abstraction Layer

The use of an abstraction layer is a method of hiding the structure and implementation of a database from an end-user. This allows development work to continue on the database while minimizing disruptions to an end-user by keeping the interface with which they interact consistent. In a SQL database, the most common method of creating an abstraction layer is to create views. Views allow queries to be updated to maintain consistent return values following database changes.

An example of where an abstraction layer would be of benefit is a web app. In this example, the web app could continue to request data from a database while expecting the results to always be returned in the same format. The abstraction layer would allow database to grow and change without requiring the app developer to make changes to their code in conjunction with each database change.

Entity Relationship Diagrams (ERDs)

ERDs serve as a visual representation of a database and allow a viewer to see the relations between the tables of a database easily. An ERD displays the tables in a database and the columns within tables. Additional information about table columns may be given, such as the data type and constraint information. Crow's foot notation is used to display relations between tables. Depending upon the style of crow's foot used, the relationship type can be expressed (i.e., 1 to 1, 1 to many).

Meta-Data worksheets

A meta-data worksheet is a method of describing a database and has more of an emphasis on table columns. A meta-data worksheet can easily be created using a spreadsheet where each row represents a column within a table. Details about a given column, such as the column's name, constraint requirements, default values, and the data type, are recorded in the worksheet.

ERD vs. Meta-Data Worksheets

ERDs and meta-data worksheets are useful tools on their own but shine when used together. The ERD is a great tool for understanding the overall structure of a database and its table's relations, while the meta-data worksheet breaks down the nitty-gritty details of a table column. When either tool is used, errors may be caught or design improvements made before writing any code. Additionally, ERDs and meta-data worksheets can easily be shared amongst colleagues, allowing for more input on the structure and potential errors.

Simple Database design

Designing a database requires a basic understanding of what data you intend to capture and how it is going to be used. If possible, having a small sample set of data available will be helpful when testing the design.

The first step is to start at a high level and create an ERD and determine what tables the database will require. Then using the ERD, relationships between tables may be established, and locations where a bridge table is needed identified.

During the previous steps, it may be helpful to populate the tables with column names; if not, this should be done now to ensure the data you want to collect is accounted for. Once the initial ERD design is complete, normalization may be performed so that data is not stored in duplicate and all table rows can be uniquely identified.

Once the ERD is complete, a meta-data worksheet can be produced, which will describe each column in every table. The meta-data worksheet will be an important document when writing code as it will contain the information on column data types and any constraints that may need to be applied.

Following the completion of the ERD and meta-data worksheet, it is advisable that they be peer-reviewed. This will help in catching errors, missing information, or possible design improvements. At this point, coding can begin, and the database can be populated.

Additional considerations to consider when designing a database are how the end-user will interact with it. For example, these considerations may determine if an abstraction layer is needed and what views or stored procedures will need development.

Summary

A well-designed database with well-thought-out constraints will pay off in the long run by making queries easier to write and future development less complex. In addition, by using constraints the integrity of the data stored in the database can be maintained. Skipping the design phase and jumping directly into code might be easy, but a great deal of effort and frustration will be saved if errors are caught and resolved during the design phase.