**Relational Database and SQL (CIS 335)**

**Final Project: Developer Reflection**

Martin Dwyer

December 7, 2017

**Introduction**

This paper offers personal reflection and insight related to a recently completed assignment as database developer for a book store database. Completion of the project can be broken down into four phases; (1) researching the business and understanding business rules, (2) breaking down the various business components into database objects, (3) organizing these objects with diagrams and flow charts, and (4) rendering the code to build the database, provide report viewing and audit functions for the client. The reflection offered herein provides an overview of the database development project, the tools used in its development, as well as a conclusion related to the important business needs that will be served with the database.

**Reflection**

The first phase of the development process was to spend time researching the industry and business organization, in this case a book store. To this end, research was obtained through competitors such as *Barnes & Noble* and *Amazon*, as well as recognized industry resources including the *International ISBN Agency* and the *American Booksellers Association.*  To proceed with the database project, certain business rules would have to be adopted. Through discussions with the book store client, it was agreed upon that book inventory would be entered manually by store personnel. The point of sale terminals would be able to provide individual book sale data to the database, regardless of the number of books purchased at each transaction. The client was eager to discuss these important issues and agree upon business rules and procedures that would integrate the database into their daily operations.

It was agreed upon that there are five primary objects for measurement and monitoring within the database: (1) books on hand at the book store, (2) various publishers for the books, (3) customers who frequent the store, (4) members of the store’s book club, and (5) sales of books. The smallest useful components for each of these objects were identified in the development of the database. For example, for every book has a title, author, genre, ISBN, publication date, publisher, number of pages and list price. Each publisher record should include publisher name, address, phone, as well as contact information for the person representing the publisher. Similar structures were built for customers, members, and book sales.

Once the smallest useful individual components of the objects were identified, graphical tools were used to begin creation of a database where redundancy is minimized, maintenance is simplified, and indices and searches can be most efficiently conducted. This process, referred to as *normalization*,[[1]](#footnote-1) was completed using analysis of functional dependencies between the various data components. A chart of functional dependencies has been provided with the final project. This charting process made it possible to reduce functional dependencies within individual objects. For example, when analyzing the various components of the publisher object, it became apparent that the zip code alone could determine two other component parts related to publisher address: city and state. As a result, a separate zip code table was added to eliminate the redundancy within the publisher object components.

Once the foundation of objects and their individual components is laid, with functional dependencies identified, it is possible to build tables using the graphical tool known as an Entity-Relationship (ER) Diagram. The ER Diagram allows the developer work in a graphical environment to simultaneously build tables while identifying primary keys for each table and graphically connecting tables by their appropriate keys. A completed ER diagram has been submitted for the book store development project. In this diagram, field names are identified for each of the eleven tables that were deemed necessary components of the book store database. Note that a primary key was identified in each of the tables, each having an exclamation point (“!”) appearing next to the field name. For example, the field name ‘people\_id’ was added to the table named ‘people’ where all relevant data pertaining to people in this project would be contained. The ER diagram allows one to illustrate the connection between database components. While ‘people\_id’ is a primary key, uniquely identifying every person connected to the database, ‘people\_id’ also serves the role of telling the ‘customers’ table where to find detail on each of the store’s customers.[[2]](#footnote-2)

With the object identified, along with components and tables normalized, the code was written to build the ‘book\_store’ database using *MySQL Workbench*. To illustrate, the code for building the ‘book\_genre’ table has been placed in Figure 1. Each of the eleven tables were created with similar structure, including; (a) dropping the table if it exists, (b) creating the table in the ‘book\_store’ database, and (c) defining the fields in the table with the appropriate data types and characteristics. At this point the script expressly identifies any primary and foreign keys, along with appropriate restrictions for deleting and updating data where field values are affected in multiple tables.[[3]](#footnote-3)

**Figure 1**

**Creating Tables**

DROP TABLE IF EXISTS people;

CREATE TABLE IF NOT EXISTS people

(

people\_id INT NOT NULL AUTO\_INCREMENT COMMENT 'Unique identifier for people',

people\_first\_name VARCHAR(50) NOT NULL COMMENT 'Person first name',

people\_last\_name VARCHAR(50) NOT NULL COMMENT 'Person last name',

people\_address1 VARCHAR(75) NOT NULL COMMENT 'Street address for person',

people\_address2 VARCHAR(75) NULL COMMENT 'Street address for person',

people\_email\_address VARCHAR(75) NULL COMMENT 'Email address for person',

people\_postal\_code INT NOT NULL COMMENT 'references postal.postal\_code',

PRIMARY KEY (people\_id),

CONSTRAINT fk\_postal\_code

FOREIGN KEY (people\_postal\_code)

REFERENCES postal (postal\_code)

ON DELETE RESTRICT

ON UPDATE CASCADE

)

COMMENT = 'Table to hold people data';

With tables created, data can be inserted into each table with the form of syntax identified in Figure 2.[[4]](#footnote-4) Where primary keys exist that are auto incremented, table values are not inserted. Instead, the field name is merely omitted from the table.

**Figure 2**

**Inserting Data**

INSERT INTO people

(people\_first\_name,people\_last\_name,people\_address1,people\_address2,

people\_email\_address,people\_postal\_code)

VALUES

("Travis","French","5042 Est. Road","818-9181 Ipsum Street", "ante.blandit@dolorelitpellentesque.edu","01841"),

("Amanda","Lloyd","P.O. Box 403, 860 Est. Avenue","Ap #413-5401 Malesuada. Rd.","Etiam.laoreet.libero@sedtortor.org","10001"),

("Lee","Henry","844-4799 Scelerisque Avenue","588-531 Eu, St.","lectus@pretiumetrutrum.ca","30301"),

("Fitzgerald","Bean","Ap #334-6848 Senectus Av.","7060 Tristique Avenue","Lorem.ipsum.dolor@nequeMorbiquis.net","52206")

;

Once the database is built and data is input useful queries, views, triggers and stored procedures should be developed to meet important business objectives. One example would be the generation of sales data by publisher, the code for which appears in Figure 3.

**Figure 3**

**Creating Tables**

**DROP VIEW IF EXISTS sales\_by\_publisher\_90\_day;**

**CREATE VIEW sales\_by\_publisher\_90\_day AS**

**SELECT pu.publisher\_id AS 'Publisher\_ID',**

**pu.publisher\_name AS 'Publisher Name',**

**COUNT(bs.sale\_id) AS 'Number\_of\_Books',**

**SUM(bo.list\_price) AS 'Total\_$\_Sales',**

**CONCAT(po.people\_first\_name," ",po.people\_last\_name) AS 'Publisher\_Contact',**

**pu.publisher\_phone AS 'Publisher\_Phone',**

**pu.publisher\_email\_address AS 'Publisher\_Email'**

**FROM book\_sales bs**

**JOIN books bo**

**ON bs.book\_id = bo.book\_id**

**JOIN publishers pu**

**ON pu.publisher\_id = bo.publisher\_id**

**JOIN people po**

**ON pu.publisher\_contact = po.people\_id**

**WHERE sale\_date >= '2017-06-30' AND sale\_date <= '2017-09-30'**

**GROUP BY pu.publisher\_id**

**ORDER BY Total\_$\_Sales DESC**

**;**

**SELECT \* FROM sales\_by\_publisher\_90\_day;**

Tracking sales by publisher, as well as being able to contact the publisher were important business objectives in the database design. The view in Figure 3 can be pulled at any time to get both the sales ranking and the contact information for the publisher. Additional business oriented views, triggers, and stored procedures were developed, including three custom views:

* Sales by customer,
* Sales by genre, and
* Sales by publisher.

Two functions were designed as triggers which produce audit records when attempts to make changes to the book sales table are made. Finally, four stored procedures with error handling were created as follows;

* CREATE records in book\_sales
* READ periodic sales data from book\_sales
* UPDATE discount applied to sales in book\_sales
* DELETE sales from the book\_sales

**Summary**

This paper has provided a reflective narrative of the process of building a book store database designed to meet specific business objectives. In the process, the importance of understanding the business, breaking down database objects, normalization and design have all been emphasized. Database design tools such as functional dependency charts and ER Diagrams were utilized and explained. SQL code for creation and insertion of data into the database were offered and briefly discussed. Moreover, the paper demonstrated that custom queries, views, triggers, and stored functions could all be developed to help the book store achieve business goals.

# Bibliography

Murach, Joel. *MySQL: Training & Reference.* Fresno: Mike Murach & Associates, Inc., 2015.

1. . The type of normalization completed for this project is more properly referred to as the “Third Normal Form” and is discussed at length in Joel Murach, *MySQL: Training and Reference*, Mike Murach & Associates, 2015, pp. 294-301. [↑](#footnote-ref-1)
2. 2. In other words, ‘people\_id’ is the primary key in the ‘people’ table while serving as a foreign key in the ‘customers’ table. [↑](#footnote-ref-2)
3. . The complete SQL code, including field definition and descriptive comments, has been attached to the final report. In the case presented, for example, the foreign key postal code is restricted on delete and cascaded on update to allow for changes that would be reflected across tables. [↑](#footnote-ref-3)
4. . Data for the testing database, sometimes referred to as ‘dummy data’, was obtained through the services of GenerateData.com. [↑](#footnote-ref-4)