SQL Management of Product Catalog

Damodar Panigrahi
Indiana University Bloomington
dpanigra@iu.edu

Yee Sern Tan
Indiana University Bloomington
yeestan@indiana.edu

ABSTRACT

This report describes a prototype developed for the management of product catalogs encompassing a relational database consisting of the tables of products, product lines, and stores. The schema of the database and the interrelationship between tables are designed and mapped with ER Modeling.

The end-product of this exercise is a website interface that enables administrative CRUD operations. Sample values are inserted for a demonstration of the operations. Sound web development practices are for realizing this endeavor are documented.

Keywords

Relational databases; SQL; Product line management

1. INTRODUCTION

In business marketing, products are often sold in mixes of products, called product lines. These product lines form a natural grouping that can be more attractive to the customer, by turning many purchase decisions into a single purchase decision, thus helping the customer to decide. This can be helpful when the marketers know which products go with which better than the customer, although in doing so the flexibility of purchase decision is inevitably reduced [2]. In this paper we introduce a product catalog database where product lines are sold by selected stores. It would be desirable, while doing this management, to have handy tools that navigate the relationship between products, product lines, and stores.

2. ER MODEL

The design of our prototype is kept simple, to illustrate how databases can be accessed from a website. We limit ourselves to 3 entities — Store, Product, and Product Line — having 2 many-to-many relationships between them. Each of the many-to-many relationship can be implemented with a bridge table comprising solely of foreign keys, and mapping two one-to-many relationship with the related entities.

The Store entity has columns Id, Name, and Address; the Product entity has columns Id, Name, and Price; the Product Line entity has columns Id and Name. The primary keys of each of these entities are Id. The bridge table Store-ProductLine has columns Store-Id and ProductLine-Id, while the bridge table ProductLine-Product has columns ProductLine-Id and Product-Id. All the columns in the bridge tables are foreign keys. The entities are designed

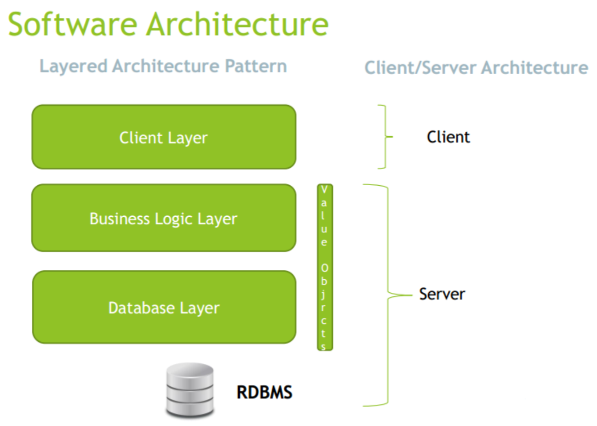


Figure 1: UI for Store Tab

with such simple schema that it is trivial to assume they are in 3NF.

The setup of our ER Model is designed with MySQL Workbench [9]. The model is exported with its DDL. The example database is then constructed by inserting entries. The final database is exported with MySQL Dump Command Line, and is submitted with this project as iuworkd-export.sql. The DDL file submitted iuwork.mwb can be opened with MySQL Workbench.

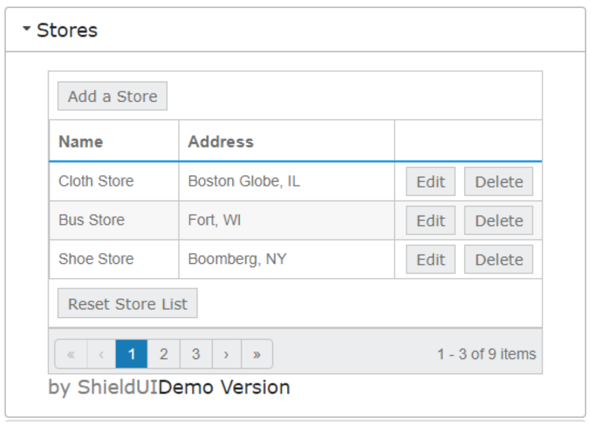
2.1 Enforcement of Relational Integrity

Removal of entity entries are not permitted if the foreign key in a bridge table corresponding to their primary key is present, as implemented with foreign-key constraints specifying ON DELETE NO ACTION. Similarly, updates of such primary keys are also prohibited under foreign key constraints specifying ON UPDATE NO ACTION.

3. SOFTWARE ARCHITECTURE AND DESIGN

The overall web application employs a client-server application, where the client, a browser, invokes the server through RESTful calls. The server is a backend comprising of Apache Tomcat Server [11] and MySQL Database [8].

For clarity of logic and ease of development and maintainability, the backend code is split into layers (see Figure 1), organized as packages. The package com.pc.vo stores, as Value Objects, comprises the foundational object structures for database operations. The package com.pc.restjersey imple-



Name	Address		
Cloth Store	Boston Globe, IL	Edit	Delete
Bus Store	Fort, WI	Edit	Delete
Shoe Store	Boomberg, NY	Edit	Delete

Figure 2: UI for Store Tab

ments the interface with the frontend. The package com.pc.bo implements as Business Objects the transaction processing requirement of each operation. The package com.pc.db implements the interface with the database.

A Test Driven Development (TDD) approach is employed in the development. Unit tests are written and handled with Java's JUnit framework [3], and the atomicity of SQL operations obviated the need for any integration tests, greatly simplifying the workflow and conflict handling.

3.1 Atomic Implementation of Operations

Updates to bridge tables are handled by first grouping according to one of the foreign keys. For the table Store-ProductLine, each update is separated according to its Store-Id, while for the table ProductLine-Product, the separation is according to ProductLine-Id. The permissible operation for updates are CREATE and DELETE. The operations are further separated for CREATE and DELETE. The option of auto-commit is set to false to prevent partial updates. Only when all the updates are executed, the commit function is called to commit all the included operations.

3.2 Secure Implementation

Security is crucial as SQL-injection is a frequent vulnerability in web databases. Whenever there is user input in the form of strings, caution needs to be taken. Java has provided the class java.sql.PreparedStatement for parameterized query that prevents this vulnerability from being exploited.

4. USER INTERFACE DESIGN

The design of our UI is aided with HTML5, Javascript and CSS. Bootstrap [4] is used for a responsive front-end. Javascript embeds the jQuery [13] plugin which in turn has add-ons of ShieldUI [10] and Chosen [1]. These available tools greatly contributed to the ease of development, aesthetics and ease of usage.

There are five tabs each corresponding to a table. The tabs corresponding to Store, Product, and Product Line are similar in interface (see Figure 2). The number of items viewable on a page is 3. The buttons are available for creating, updating, and deleting an entry. For the first two, the required information can be keyed into the text boxes, and there are the save and cancel buttons. There is another

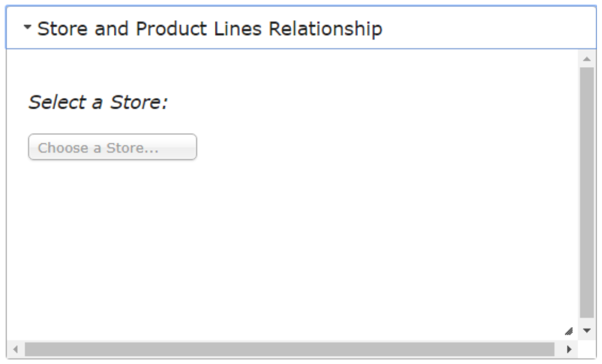


Figure 3: UI for Store-ProductLine Tab (Step 1)

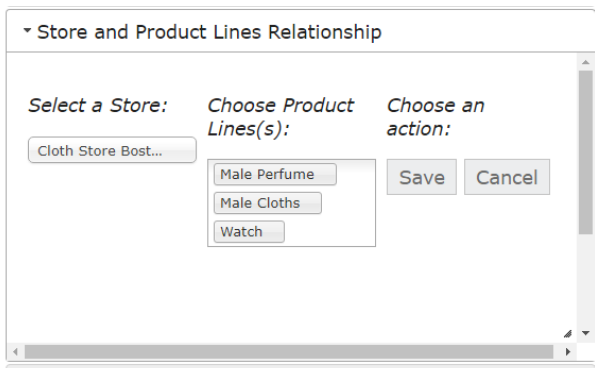


Figure 4: UI for Store-ProductLine Tab (Step 2)

button for refreshing the list.

The tabs corresponding to the bridge tables are similar in interface. To use, first select an item in the first foreign key (see Figure 3). The page will then display all the existing items in other foreign key (see Figure 4). The user can then insert or delete entries from this interface.

Being a simple prototype, the number of items viewable on our website is only suitable for small applications. On each page, there will only be 3 items of stores, product lines, or products viewable. Whereas for the bridge tables, the scrollbar is used for accessing items. While the website is navigable and user-friendly, for better efficiency of large number of items, it would be desirable that items can be searched within fields and that more items can be viewed on one page. To accomplish such, the design of layout will need reconsidering.

5. DEPLOYMENT STEPS AND POTENTIAL CONSIDERATIONS

The current development is only for testing on a standalone workstation, it can be possible to extend to a distributed client-server deployment with Java Naming and Directory Interface (JNDI).

For installation on a machine in this current setting, JRE of JavaSE 1.8 [7] needs to be installed and configured on the server. In our work, we used Apache Maven [12] on Eclipse to import the necessary libraries. These can be found in the pom.xml file submitted with this project under path /project/dependencies/dependency. The downloaded jar files should be copied onto the lib subfolder under Tomcat folder (%CATALINA_HOME%).

MySQL 5.7 needs to be installed and running on the server. For simplicity, we have hard-coded the password to ”root”, and the database needs its password to be set to the same. (The correct way to implement this is to create another webpage for authentication, together with a table for CRUD operations on user credentials.) Having logged in to MySQL, switch to database ”iuwork”, and simply copy the contents of the file iuworkd-export.sql, submitted with this project into MySQL Command Line Client. This will initialize the database.

As evident from earlier in this section, Tomcat needs to be installed and configured on the server. Version 8.0 onwards is recommended.The pc-pc.war file submitted with this is to be copied to Tomcat under webapps folder. Tomcat can then be started to deploy the war file.

We have tested this web application on the Chrome and Firefox browsers. As mentioned, the current setting only works for standalone workstations. In the browser, the URL to access the website is http://localhost:8080/pc-pc/ .

Since this website is intended only for administrator level privileges, authentication can be implemented to allow for single level of access control. MySQL being a relational database, there could be issues on responsiveness when multiple clients are accessing a server. Access to related parts of the database can get locked when it is used. In order to prevent hogging of resources by uncooperative clients, it may be reasonable to let the system administrator use MySQL Admin Command Line Client to kill such threads.

6. SOFTWARE AND TECHNOLOGY STACK

For work on database, the schema is fixed, and so DDL is not a concern after setting up. Data Manipulation Language is hard-coded into the web application, and the parameters can be passed into parameterized queries.

With a MySQL database running and connected, Tomcat connects to it via JDBC. Since we do not foresee the database operating under a heavy load, JDBC is a more natural choice for object-oriented connectivity. Thus, for development of the backend we rely on JavaSE 1.8. The Java EE Edition 7 [5] is also installed to incorporate Dynamic Web Application functionalities and JAX-RS (Java API for RESTful Web Services). Jersey Web Application [6] is used to support JAX-RS. We have also employed JSON for encapsulating database entries as it is passed around in the code.

The frontend communicates with the server with RESTful calls using the asynchronous AJAX functions. All the functionalities are found on the welcome page.

Among the stack, the technologies we use are common. Particularly, REST and JDBC are advantageous in terms of interoperability with many different systems.

6.1 Additional Tools

- Eclipse IDE
- Chrome Developer Tools
- Chrome POSTMAN
- Firefox Firebug
- Firefox HTTPLiveReader

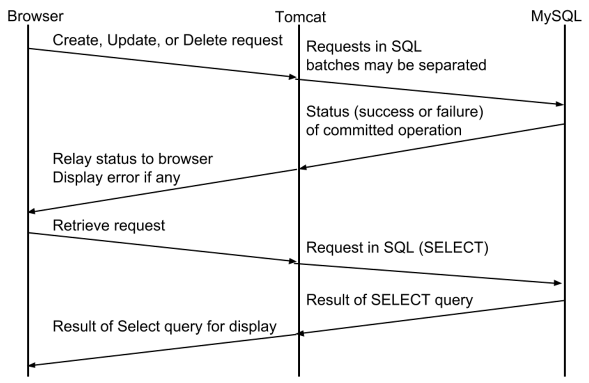

Browser
Tomcat
MySQL
Create, Update, or Delete request
Requests in SQL batches may be separated
Status (success or failure) of committed operation
Relay status to browser Display error if any
Retrieve request
Request in SQL (SELECT)
Result of SELECT query
Result of Select query for display

Figure 5: Sequence Diagram of Operations

- MySQL Command Line Tool
- MySQL Admin Command Line Tool
- MySQL Dump Command Line Tool
- MySQL Workbench
- Apache Maven

7. SUMMARY

This report shows how a prototypical web application can be modeled, built and deployed for using MySQL database. In the sequence diagram (Figure 5) the functionalities in our web application is illustrated. Many potential extensions to its functionalities are possible, e.g. databases with multiple-level access control, data access layer code design patterns, inventory tracking, business transaction processing, automatic report generation, non-repudiation measures etc. Where needed, additional features usually add to the application's complexity, thus calling for a sound architecture to base upon. Our design has merits in terms of usability, maintainability (at this level of complexity), interoperability, security, and testability.

8. AUTHOR CONTRIBUTION BREAKDOWN

The first author did the design, the proof-of-concept, the choice of technologies, and some preliminary testing. The second author did a review of the topic of product line marketing, refined the coding and deployment, recorded the presentation and wrote this report.

9. REFERENCES

[1] P. Filler and M. Lettini. Chosen 1.7.0. software: Javascript library.
[2] K. E. Homa. website. http://faculty.msb.edu/homak/homahelpsite/webhelp/Content/Product_Line_Management_Overview.htm.
[3] JUnit. Junit 4. software: unit testing framework.
[4] MIT. Bootstrap 3.3.7. software: web development framework.
[5] Oracle Corporation. Jdk javaee 7. software: development kit.
[6] Oracle Corporation. Jersey web application. software: framework.

[7] Oracle Corporation. Jre javase 1.8. software: runtime.

[8] Oracle Corporation and/or its affiliates. Mysql 5.7. software: relational database.

[9] Oracle Corporation and/or its affiliates. Mysql workbench 6.3. software.

[10] Shield UI Ltd. Shield ui. software: Javascript framework.

[11] The Apache Software Foundation . Apache tomcat 8.0. software: servlet.

[12] The Apache Software Foundation. Apache maven 3.3. software: project management and comprehension tool.

[13] The jQuery Foundation. jquery 3.2.1. software: Javascript library.