Phase 1 Deliverable

CIS4301

Group #15:

Christine King

Michael Bentz

Issa Malke

Joshua Trinkler

Our project topic is a book rating system that utilizes the Book-Crossing dataset found at <http://www2.informatik.uni-freiburg.de/~cziegler/BX/>. The dataset is organized into three tables, one table containing user information, another containing book information, and the third containing the individual ratings for each book. Our database will consist of roughly 100,000 users with over 1,000,000 ratings on 200,000 books. By integrating the data, our application program will provide users the ability search for books based on title, author, ratings, and other various attributes. Users will also be able to create new entries for books, rate existing books, and update existing books’ attributes. We hope that this application will be useful for determining book popularity and recommending books to users based on ratings and popularity with various demographics.

For our project, the database system we will use is Oracle 11g provided by the CISE department (the orcl instance running on CISE servers). Our application will be hosted on the internet by the CISE department as well. We will use JDBC (Java Database Connectivity) to make the connection between our database and our web-based user interface application. Also, Javascript, HTML5, and CSS will be used for the web design portion. In addition, we will use an IDE to facilitate coding, such as Eclipse or Netbeans. To test our queries on our database before we integrate them into our webpages, we will use Oracle SQL Developer.

The main functions that the web-based interface of our program will provide are the following: Account, which consists of create a new account or sign in exiting account. Also, the function Rate Book will allow the user to rate any book from 1 to 10 scale without restriction. In addition, Find a book, which will allow any visitor regardless if he/she has an account or not to search for books and get their rating. Adding a new book, is another function that will allow the user who has an account to add new book and rate it in the same time. Also, a user can update his last rating of a book.

These functions work together depending on the other respective functions. For example, non-account user functionality will be limited to account creation and searching the database. Other functionality such as adding, reviewing or editing a review is exclusive to users with an account. When a user signs in, he or she will also be able to add a book with certain criteria such as ISBN, title, one author, year published, and rating (ranging from 1-10). However, the ISBN must be unique because it will serve as our primary key for the books throughout the database. Other attributes such as the title, year published, and author can have multiple occurrences since these will not be treated as a primary key. While adding a book is available to users with an account, book removal is restricted to all users, with the exception of updating a books’ various attributes.

The functions above require us to store the following data:

* Users
  + This table contains information about the users. The User ID is the primary key, and each one is mapped to an integer. Also, the user’s demographic data is provided (*`Location`*, *`Age`*) if it is available. Otherwise, these fields contain *NULL*-values.
* Books
  + This table contains information about the books that have been rated by the users. Books are identified by their respective ISBN, which are the primary keys. Also, ISBNs cannot be the same, so if the user tries to add a new book to the database, the ISBN number will be compared to existing ISBNs to see if it already exists. Moreover, some content-based information is given such as*`Book-Title`*, *`Book-Author`*, *`Year-Of-Publication`*, and *`Publisher`*. In case of several authors, only the first is provided, which will not be a problem since we are depending on the ISBN number as the primary key.
* Book-Ratings
  + This table contains the ratings for each book.. Ratings are expressed on a scale from 1-10 (higher values denoting higher appreciation). Each rating is identified by a superkey of a User ID and ISBN. A book can have multiple ratings by different users. There will be only one or zero ratings made by a user per book. A user can rate many books.

We will be able to derive new information from the stored data in our database when we query. Some of these interesting queries include:

* What are the top 10 highest rated books?
* Which user has rated the most books?
* How many users have rated a certain book?
* How many users have rated at least X books?
* Which user(s) have rated the least amount of books?
* What books have been rated by people in location X and what are those ratings?
* What books have been rated by people of age Y (or age range X to Y)?
* What are the top 10 highest-rated books amongst a certain age group?
* What are the top 10 highest-rated books rated by users from location X?
* What is the average rating of a certain book?
* What is the average rating of all books written by a certain author?
* Which books have been rated by the most users?
* What is the average rating of books published in year X?

The answers to these queries and many other interesting queries can be derived from the data even though they are not directly stored in the data. Thus, our application would greatly benefit from database support.