Database Programming Project Proposals

Zhizhang Shen *
Dept. of Computer Science and Technology
Plymouth State University

November 30, 2018

Abstract

This is Part III of the lab notes prepared for the students of *CS3600 Introduction to the Database Systems* for Fall 2018.

We present a few ideas, with consistent level of challenge, which you can use to develop your project proposal for this course.

This project worths fifteen points. We also give some specific criteria and guideline, in §4, as how the students will be graded for the outcome in completing this project.

Contents

Address correspondence to Dr. Zhizhang Shen, Dept. of Computer Science and Technology, Plymouth State University, Plymouth, NH 03264, USA. *Email address:* zshen@plymouth.edu.

1 The rule of the game

You will learn lots of database related concepts and skills throughout this course. But, *Application is the key.* To apply some of the stuff as we have been learning in this course, and to develop some habits of team work, the key ingredients of carrying out database projects, the class will be cut into a few project teams, with 3 to 4 students. Each team will elect a project leader, come up with a project proposal, a rough schedule, and complete the project by the end of this semester. The project group will then make a presentation to the whole class at the end of the semester.

In principle, all the members of the same team will be assigned the same grade. All the members of a team are expected to make a reasonable contribution, and the team leader will make a report about what each and every team member has done, which will be taken into consideration when a grade is assigned.

2 Project proposals

We list seven proposals in the following. Some of them are adapted based on some ideas as contained in [2].

A project team can certainly come up with their own non-trivial proposal, consistent with these sample proposals.

2.1 Video Chain Database

This is essentially a management system for a chain of video stores. Thus, those stores might share the same inventory, but each store has its own rental records, and employee records. Thus, such a database should contain information about *Stores, Employees, Movies, Members, Rentals,* and *Vendors.*

The group should go through the following steps in completing this project:

1. Design the database, following an E/R approach; then go through the normalization process to come up with a collection of tables that are in Boyce-Codd normal forms.

   The team has to include the original design in terms of E/R diagrams and paper work to show the normalization process in the final documentation.

2. Describe some realistic constraints such as primary keys, foreign keys, check constraints, and `not null` constraints, for the tables and attributes, etc..

   The team has to include a requirement document *in the form of the one as discussed in [1, 14.2], a preliminary analysis and a list of final constraints.*

3. Use *MariaDB* to create the normalized tables.

   The team has to come up with a list of print-outs of the table structure and the content, similar to those as contained in the lab notes.

4. Do a research on how the most recent version of *MariaDB* implements the concept of *triggers* and then create appropriate triggers for your database. to maintain integrity of the database and to perform appropriate actions on database updates.

   For example, one possible trigger could be placed and then acted upon *after* the last copy of a movie is being rented out.

   The team has to come up with a list of issues, and a discussion as whether an issue can be resolved with a trigger, *before, after* or *instead;* and why it is not done.

5. Populate the database by using *MariaDB* statements, or with some GUI interface such as `PhpMySQLAdmin.`

6. Write an interface in *HTML,* with embedded *PhP* script(s). The main interface should contain a *Main* menu, including *Member functions, Administrative functions,* and *Reporting functions.*

   - Implement the *password security* mechanism as described in Section 5.1.1 in the *PhP* Notes.
   - The *Member* page should include such functions as *Check out, New member sign-up, list of videos for a member that have yet to be returned,* and *Membership cancellation.*
   - The *administrative* page should include such functions as *add/delete employees, buy new tapes, ,* as well as *open new stores,* etc..
   - The *report* page should include such features as *print the whole catalog, print a due list of tapes, print the employee list, print rental summary (ordered by rental frequency,* etc..

   The group should include in the final documentation a list of questions a user of this system might want to ask, together with a list of *MariaDB* queries that help to answer them.

7. Document the project, containing at least the pieces as pointed out in the above.

2.2 Library Management System

This is supposed to develop a database system for the local library. Your finished product should contain information about *books, book authors, publishers, employees, ILLs,* and *borrowers.*

The group should go through the following steps in completing this project:

1. Design the database, following an E/R approach; then go through the normalization process to come up with a collection of tables that are in Boyce-Codd normal forms.

   The team has to include the original design in terms of E/R diagrams and paper work to show the normalization process in the final documentation.

2. Describe some realistic constraints such as primary keys, foreign keys, check constraints, and `not null` constraints, for the tables and attributes, etc..

   The team has to include a requirement document *in the form of the one as discussed in [1, 14.2], a preliminary analysis and a list of final constraints.*

3. Use *MariaDB* to create the normalized tables.

   The team has to come up with a list of print-outs of the table structure and the content, similar to those as contained in the lab notes.

4. Do a research on how the most recent version of *MariaDB* implements the concept of *triggers* and then create appropriate triggers for your database. to maintain integrity of the database and to perform appropriate actions on database updates.

   For example, One of such *before* triggers could be that someone wants to check out a book on how to build bombs

   The team has to come up with a list of issues, and a discussion as whether an issue can be resolved with a trigger, *before, after* or *instead;* and why it is not done.

5. Populate the database by using *MariaDB* statements, or with some GUI interface such as `PhpMySQLAdmin.`

6. Write an interface in *HTML,* with embedded *PhP* script(s).

   - The top page should direct the users to either *Patron functions, Administrative functions,* or *Reporting functions.*
   - Implement the *password security* mechanism as described in Section 5.1.1 in the *PhP* Notes.
   - The *Patron* page should include such functions as *check out, sign-up, return, pay fine,* and etc..
   - The *administrative* page should include such functions as *buy a new book, sell a book, search for book (by title, subject, author)*, and *patron cancellation.*
   - The *report* page should include such features as *print the patron list, print a due list, print the ILL list, print monthly Ten Hot Book list,* etc..

   The group should include in the final documentation a list of questions a user of this system might want to ask, together with a list of *MariaDB* queries that help to answer them.

7. Document the project, containing at least the pieces as pointed out in the above.

2.3 A Music Store Database

This is to develop a database for a Music store. You product should contain information about *Employees, Inventory, Customers, Sales,* and *Returns.*

The group should go through the following steps in completing this project:

1. Design the database, following an E/R approach; then go through the normalization process to come up with a collection of tables that are in Boyce-Codd normal forms.

   The team has to include the original design in terms of E/R diagrams and paper work to show the normalization process in the final documentation.

2. Describe some realistic constraints such as primary keys, foreign keys, check constraints, and `not null` constraints, for the tables and attributes, etc..

   The team has to include a requirement document *in the form of the one as discussed in [1, 14.2], a preliminary analysis and a list of final constraints.*

3. Use *MariaDB* to create the normalized tables.

   The team has to come up with a list of print-outs of the table structure and the content, similar to those as contained in the lab notes.

4. Do a research on how the most recent version of *MariaDB* implements the concept of *triggers* and then create appropriate triggers for your database. to maintain integrity of the database and to perform appropriate actions on database updates.

   For example, one such *before* triggers could be that some one returns a piece s/he bought on the same day. As an action, you might want to ask him/her questions as why it is returned.

   The team has to come up with a list of issues, and a discussion as whether an issue can be resolved with a trigger, *before, after* or *instead;* and why it is not done.

5. Populate the database by using *MariaDB* statements, or with some GUI interface such as `PhpMySQLAdmin.`

6. Write an interface in *HTML,* with embedded *PhP* script(s).

   - Implement the *password security* mechanism as described in Section 5.1.1 in the *PhP* Notes.
   - For the Sales/Return *submenu,* you should include such functions as *Process a sale, process a return, view a sale,* and *view a return.*
   - For the administrative *submenu,* you should include such functions as *add/delete employees, add/drop "stuff", add/drop customer,* etc..

- For the report submenu, you should include such features as *print the whole catalog, print a list of stuff (by format such as CD, tape, DVD, or by category), print the employee list, print monthly Top Ten lists, print out the Most Returned List,* etc..

The group should include in the final documentation a list of questions a user of this system might want to ask, together with a list of *MariaDB* queries that help to answer them.

7. Document the project, containing at least the pieces as pointed out in the above.

2.4 University Student Database

This is supposed to develop a database for the College. Your finished product should contain information about *courses and their sections, departments, instructors, students,* as well as *Enrollments.*

The group should go through the following steps in completing this project:

1. Design the database, following an E/R approach; then go through the normalization process to come up with a collection of tables that are in Boyce-Codd normal forms.

   The team has to include the original design in terms of E/R diagrams and paper work to show the normalization process in the final documentation.

2. Describe some realistic constraints such as primary keys, foreign keys, check constraints, and `not null` constraints, for the tables and attributes, etc..

   The team has to include a requirement document *in the form of the one as discussed in [1, 14.2], a preliminary analysis and a list of final constraints.*

3. Use *MariaDB* to create the normalized tables.

   The team has to come up with a list of print-outs of the table structure and the content, similar to those as contained in the lab notes.

4. Do a research on how the most recent version of *MariaDB* implements the concept of *triggers* and then create appropriate triggers for your database. to maintain integrity of the database and to perform appropriate actions on database updates.

   For example, one such *before* triggers could be that when a student fails to maintain full-time status after dropping another course. An action could be to remind him that his/her parents can no longer cover his/her medical insurance.

   The team has to come up with a list of issues, and a discussion as whether an issue can be resolved with a trigger, *before, after* or *instead;* and why it is not done.

5. Populate the database by using *MariaDB* statements, or with some GUI interface such as `PhPMySQLAdmin.`

6. Write an interface in *HTML,* with embedded *PhP* script(s).

   - Implement the *password security* mechanism as described in Section 5.1.1 in the *PhP* Notes.
   - The *student* page should include such functions as *register for a course, add/drop a course, asks for a copy of transcript, and pay fees.*
   - The *administrative* page should include such functions as *create a new course and drop an existing one, add sections of an existing courses and drop one,* as well as *add/drop a student.*
   - The *report* page should include such features as *print the whole schedule, print an individual schedule, print the student list of a course, or a department,* etc..

   The group should include in the final documentation a list of questions a user of this system might want to ask, together with a list of *MariaDB* queries that help to answer them.

7. Document the project, containing at least the pieces as pointed out in the above.

2.5 Department Management Database

This is supposed to develop a database for an academic department. You product should contain information about *administration, faculty, programs, assessments, committees* and various *documents.*

The group should go through the following steps in completing this project:

1. Design the database, following an E/R approach; then go through the normalization process to come up with a collection of tables that are in Boyce-Codd normal forms.

   The team has to include the original design in terms of E/R diagrams and paper work to show the normalization process in the final documentation.

2. Describe some realistic constraints such as primary keys, foreign keys, check constraints, and `not null` constraints, for the tables and attributes, etc..

   The team has to include a requirement document *in the form of the one as discussed in [1, 14.2], a preliminary analysis and a list of final constraints.*

3. Use *MariaDB* to create the normalized tables.

   The team has to come up with a list of print-outs of the table structure and the content, similar to those as contained in the lab notes.

4. Do a research on how the most recent version of *MariaDB* implements the concept of *triggers* and then create appropriate triggers for your database. to maintain integrity of the database and to perform appropriate actions on database updates.

For example, one of such triggers could be to check when a certain date comes up, e.g., May 1, to see if a faculty has filed an annual report. If not, whenever that faculty member signs up, show a warning message.

The team has to come up with a list of issues, and a discussion as whether an issue can be resolved with a trigger, *before, after* or *instead;* and why it is not done.

5. Populate the database by using *MariaDB* statements, or with some GUI interface such as `PhPMySQLAdmin.`

6. Write an interface in *HTML,* with embedded *PhP* script(s).

   - Implement the *password security* mechanism as described in Section 5.1.1 in the *PhP* Notes.
   - The *Information* page might contain something like general announcements, news information, etc..
   - The *administrative* page should include such functions as *add/delete/update faculty, form/dismiss/update committees,* as well as *add/delete/update programs,* etc..
   - The *report* page should include such features as *print various documents, print a list of faculty members/committee members,* and *check out the assessment results (maybe some statistics should be provided),* etc..

   The group should include in the final documentation a list of questions a user of this system might want to ask, together with a list of *MariaDB* queries that help to answer them.

7. Document the project, containing at least the pieces as pointed out in the above.

2.6 Grade Book Database

This is supposed to develop a database to manage a grade book. You product should contain information about *courses, students, grades for homework, quizzes, midterm, final,* and various *letter grades.*

The group should go through the following steps in completing this project:

1. Design the database, following an E/R approach; then go through the normalization process to come up with a collection of tables that are in Boyce-Codd normal forms.

   The team has to include the original design in terms of E/R diagrams and paper work to show the normalization process in the final documentation.

2. Describe some realistic constraints such as primary keys, foreign keys, check constraints, and `not null` constraints, for the tables and attributes, etc..

   The team has to include a requirement document *in the form of the one as discussed in [1, 14.2], a preliminary analysis and a list of final constraints.*

3. Use *MariaDB* to create the normalized tables.

   The team has to come up with a list of print-outs of the table structure and the content, similar to those as contained in the lab notes.

4. Do a research on how the most recent version of *MariaDB* implements the concept of *triggers* and then create appropriate triggers for your database. to maintain integrity of the database and to perform appropriate actions on database updates.

   For example, one such *after* triggers could be that when an 'F' is generated as the grade for a student, generate a sympathetic email message to the student. How should this be done exactly?

   The team has to come up with a list of issues, and a discussion as whether an issue can be resolved with a trigger, *before, after* or *instead;* and why it is not done.

5. Populate the database by using *MariaDB* statements, or with some GUI interface such as `PhpMySQLAdmin.`

6. Write an interface in *HTML,* with embedded *PhP* script(s).

   - Implement the *password security* mechanism as described in Section 5.1.1 in the *PhP* Notes.
   - The *administrative* page should include such functions as *add/delete /update courses/students/grades.* Certain security measure, such as password, should be included. Certain feature on curving should also be included.
   - The *report* page should include such features as *printing out grades for courses, with certain restrictions such as printing out only those who fail the course,* and *given a name, or a student number, looking for his/her grade,* and *randomly generating unique pseudo name for students for a course, and printing out labels for the whole class,* etc..

   The group should include in the final documentation a list of questions a user of this system might want to ask, together with a list of *MariaDB* queries that help to answer them.

7. Document the project, containing at least the pieces as pointed out in the above.

3 What to hand in...

Hand in a general description of the project, the database design, interface design, the source code (a hyperlink when applicable), as well as the database tables, and any other supporting material, by 10 p.m., Wednesday, December 5, 2018.

Check out the course page for a sampler writing, which is one way to summarize all the work that you will have done for this project.

A presentation will be held at the end of this semester to the class, on Friday, December 7, 2018.

4 Grading criteria

This project will add up to 15 points towards the total grade of the course, and is graded by the following criteria and guidelines:

1. *Database design:* A well developed database, as described in a well written and readable document, a set of appropriate ER charts, a collection of normalized tables obtained through a normalization process, appropriate keys and foreign keys consistent with a set of well chosen integrity constraints, and their FD equivalents, appropriate population and password encryption. (7)

2. *Queries:* Develop a collection of non-trivial and application oriented queries and/or triggers, addressing users's needs, and a correct implementation in *MariaDB*. (4)

3. *Presentation:* Clear, cohesive and complete understanding of the project and be able to answer questions related to the project (4)

References

[1] Kifer, M., Bernstein, A., and Lewis, P., `Database Systems(Introduction Version),(Second Ed.)` Addison-Wesly, Boston, MA, 2005.

[2] Sunderraman, R., *Oracle 9i Programming–A Primer,* Addison-Wesley, Reading, MA, 2004.