CS 410: Database Engineering (WI)

High-stakes Writing Assignment: Data-intensive Application Development

Contents

1 Introduction

This is a semester-long, two-member team project. The overarching goal of this project is to develop a data-intensive application from inception to completion. As part of this process, you will develop several *project artifacts* such as: data and process requirements specification; conceptual, logical, and physical schema design; transaction and application design. You will also: develop SQL scripts for creating, loading, and querying the database; and perform transaction and application implementation.

Your team is responsible for selecting an appropriate *data-intensive application*. The problem should have sufficient *breadth and depth* to demonstrate various *work products* discussed later in this document. Your professor will assist you in identifying an appropriate problem. Often students start off with a problem of sufficient breadth and depth, and reduce it to a trivial problem towards the end of the semester due to time constraints. You need to guard against this trend to earn a good grade.

This is a *high-stakes writing assignment* and accounts for 20% of the course grade. Writing activities span the entire semester. Furthermore, each document needs to be *revised at least once* based on the instructor and peer feedback.

The intended audience for the project artifacts are technical people with background in relational database systems, and software applications development. The artifacts are crucial for developing data-intensive applications. Therefore, they need to be accurate, concise, precise, and yet comprehensive.

You need to perform *self-assessment* before submission. Use the rubric specified in section **??** to perform self-assessment. The instructor will also use the same rubric to grade this assignment.

Both the instructor and peers will provide feedback on artifacts as they are produced. However, score/grade is not assigned to each artifact. Score is assigned at the end of the semester for all the artifacts as a whole. This approach presents ample opportunities for students to revise their writing throughput the semester. Final submission in the form of a single PDF document. You need to use the instructor provide LaTeX templates (no exceptions) to generate the PDF file.

Here is a suggested roadmap for project activities. Each activity requires developing and revising one or more technical documents.

1. Select a data-intensive application to develop
2. Develop use-cases (aka user stories) for the application
3. Develop use-case diagram
4. Identify and document tasks
5. Identify and document database transactions
6. Identify and document database queries
7. Develop conceptual data model
8. Develop logical data model

9. Perform initial physical database design
10. Implement the database
11. Implement transactions and queries
12. Implement database applications
13. Perform self-assessment
14. Perform metacognitive reflection
15. Create PDF portfolio

In the following, some of the above activities are described briefly (as they are discussed in prerequisite courses). For additional details about them, consult the documents that you have developed in algorithms and software engineering courses.

2 Data-intensive Application Selection

A data-intensive application is characterized by: lots of data; relatively little processing; large number of insertions, deletions, updates, and queries. Data is viewed as a corporate resource and is often used for tasks varying from marketing campaigns, new product strategies, inventory management and distribution logistics, to improving customer loyalty.

Ideally, the application should have: well-understood or well-designed business processes; sufficient documentation about the business processes; tasks are known; application boundary is well-demarcated; allows developing one organization-wide database schema from multiple department-wide database views.

You should either possess sufficient domain knowledge of the application area, or have access to a domain expert. Otherwise, it is extremely difficult to develop a successful data-intensive application.

3 Identification and Documentation of Use-cases

Identify various *classes of users* (aka *actors*) for the data-intensive application. Note that users can be human as well other systems. In some applications, *time* is a user. For example, end-of-day, end-of-week, end-of-month, end-of-quarter, and end-of-fiscal-year are all time-triggered events. Your application needs to respond to these time-triggered events.

Use-cases describe interactions between the users and the system. Some interactions can be normal (no error conditions), other interactions may entail additional processing (e.g., preferred customers receive additional services), and yet other interactions require error recovery due to various conditions such as erroneous input or device malfunctioning. Each path through a use-case is called a *scenario*. In other words, a use-case is a set of related scenarios.

Conceptually, a use-case represents a *unit of work* from an end-user perspective. A use-case involves executing a set of tasks in certain sequence.

4 Use-case Diagram

Use-case diagram is a pictorial representation of interactions between the application users and use-cases. It also shows relationships between use-cases such as one use-case being embedded in another use-case, or one use-case extending the functionality of another use-case.

5 Identification and Documentation of Data Tasks in the Application

Each use-case scenario requires executing a set of tasks. For each task identify and document inputs needed, and outputs generated. Also, specify possible error conditions that might occur as inputs are transformed into outputs.

6 Identification and Documentation of Transactions

A *transaction* is a unit of work both from a database end-user perspective as well as from the database system perspective. A transaction requires executing all the tasks that comprise a unit of work in entirety âĂŤ all or nothing proposition. For each transaction specify its frequency of execution.

7 Identification and Documentation of Database Queries

Unlike transactions, database *queries* do not change the data in the database. Queries require only read access to the database. Some queries may take quite a bit of time to complete. Therefore, performance is often an issue for database queries.

Specify queries in plain English. For each query specify what data is to be retrieved (not how) as well as its frequency of execution.

8 Conceptual Data Model

Start with Entity-Relationship (ER) and Enhanced Entity-Relationship (EER) diagrams for department-wise transactions and queries. The number of departments you will have (e.g., registrar, library, financial aid, campus housing) depends on the scope of the data-intensive application. In the second step, integrate these department-wise diagrams into one corporate-wide ER/EER diagram. Follow established diagrammatic conventions. Use SQL Power Architect tool for developing ER/EER diagram.

9 Logical Data Model

Identify functional and multivalued dependencies. Transform ER/EER diagrams into a relational schema. Determine functional dependencies and perform normalization. Transform

each table into 3NF or BCNF using the functional dependencies and normalization rules. You may use Database Design (DBD) tool for this task. For each table in the final schema, specify primary and foreign keys. Also specify data integrity constraints.

10 Physical Data Model

For each database file, specify *initial* storage structures and access paths. Typically, these storage structures and access paths need modifications based on *observed performance* once the database is in operation (aka database tuning).

11 Database Creation and Data Loading

Now that your logical database schema and physical database design is in place, write SQL scripts to create the database using PostgreSQL. Load existing data into the tables using either SQL statements or *bulk loading*. Resolve any data integrity constraint violations.

12 Implementing Database Transactions and Queries

Write SQL code for transactions and queries. Verify and validate all transactions and queries. Comment SQL code sensibly.

13 Developing Database Applications

This step involves writing database applications using Java or scripting languages such as JSP, PHP, and ASP.NET. Include rationale for choosing a specific language for developing the database applications. Students should not choose a scripting language unless they are already familiar with it. Simply there is no time to learn a new scripting language. Demonstrate a simple Web application based on the database that you have developed.

14 Summary of Revisions

Briefly describe who critiqued your document (e.g., instructor, peer, friend) and provided suggestions for improvement, and how you have incorporated the suggestions and revised the document.

15 Trunitin.com

Compile the LaTeX document and produce a PDF file. Submit the PDF file to turnitin.com. What does the results from turnitin.com say? What percentage of your document is similar to other documents? How do you defend if more than 15% of your document is similar to other documents?

16 Metacognitive Reflection

We learn how to learn through metacognitive reflection by actively planning, monitoring, and evaluating our own thinking and learning.

Learning how to learn involves going beyond the cognitive and into the realm of the metacognitive. In the context of this assignment, cognitive part is the development of the data-intensive application. Metacognitive part refers to the strategies, techniques, and tools you have used to accomplish these tasks.

Perform metacognitive reflection on this assignment by answering the following questions:

1. Did I solve the right problem?
2. Did I solve the problem right?
3. How did I approach solutions to the problems?
4. What strategies and techniques did I draw upon?
5. Did I learn a new strategy in completing this assignment? If so, how is it different from and similar to the repertoire of techniques that I have already acquired?
6. Any other information you may wish to add · · ·

17 Preparing Written Report

Prepare a document that describes the artifacts of this assignment using the (provided) LaTeX template (no exceptions). Consult the rubric listed in Section 18) to ensure that you have addressed all aspects of this assignment.

18 Self-assessment

You need to assign a grade for this assignment yourself. Use the rubric listed below to come up with a score. The instructor will also assign a score. Without this section, assignment will be returned with a score of 0.

The first two traits correspond to writing and the remaining ones relate to domain aspects of the project.

Perf Level Trait	Poor	Fair	Good	Outstanding
Diction	Chooses non-technical vocabulary that inadequately conveys the intended meaning of the communication.	Chooses technical vocabulary that conveys the intended meaning of the communication.	Chooses appropriate, technical, and varied vocabulary that conveys the intended meaning of the communication.	Chooses lively, precise, technical, and compelling vocabulary and skillfully communicates the message.
Communication Style	Has only a few (but noticeable) errors in style, mechanics, or other issues that might distract from the message.	Is virtually free of mechanical, stylistic or other issues.	Uses complex and varied sentence styles, concepts, or visual representations.	Creates a distinctive communication style by combining a variety of materials, ideas, or visual representations.
Application Selection	Not a data-intensive application.	Application is somewhat data-intensive	Application is data-intensive but limited access to domain expertise.	Application is data-intensive with adequate access to domain expertise.
Use-cases	Less than 50% of the use-cases are identified, and documented poorly.	Over 75% of the uses-cases are identified and documented using a standard template.	All the use-cases are identified, but detail is missing for some use-cases.	All the use-cases are identified, well-documented using a standard template, and verified against application requirements.
Data Tasks	Inputs, outputs, and possible error conditions are documented for less than 50% of data tasks.	Inputs, outputs, and possible error conditions are documented for less than 75% of data tasks.	Inputs, outputs, and possible error conditions are documented for all data tasks.	Inputs, outputs, and possible error conditions are documented for all data tasks. Processing logic (or high-level algorithms) for transforming inputs into outputs is also described.
Transactions and Queries	Less than 50% of the transactions and queries are identified and described.	Less than 75% of the transactions and queries are identified and described.	All the transactions and queries are identified and described.	All the transactions and queries are identified and described including their frequency of execution.

Perf Level Trait	Poor	Fair	Good	Outstanding
Data Models	Only conceptual data model is described in detail. Cursory treat of logical data model. Physical data model design is missing.	Conceptual and logical data models are described in detail. Physical data model design is missing.	Conceptual, logical, and physical data models are described completely and precisely.	Conceptual, logical, and physical data models are described completely and precisely. Database normalization based on functional dependencies is discussed in detail.
Creation and Loading	SQL scripts are written and executed to create the database and load the data. Data in the database is trivial in size.	SQL scripts are written and executed to create the database and load the data. Data in the database is moderate in size.	Conceptual, logical, and physical data models are described completely and precisely. Data in the database is huge in size – in the order of millions of rows.	Conceptual, logical, and physical data models are described completely and precisely. Data in the database is huge in size – in the order of millions of rows. Detail evidence is provided on how referential integrity constraints are resolved.
Implementing Transactions and Queries	Less than 50% of the transactions and queries are implemented.	Less than 75% of the transactions and queries are implemented.	All the transactions and queries are implemented; run and execute correctly.	All the transactions and queries are implemented; run and execute correctly. There is also written evidence that transactions and queries are tested.
Revisions	Only peer or instructor feedback is solicited, but not incorporated.	Both peer and instructor feedback is solicited but not incorporated.	Both peer and instructor feedback is solicited and incorporated.	Both peer and instructor feedback solicited and incorporated. Evidence is presented to show how the feedback improved the document.
Turnitin.com	No submission is made to turnitin.com	Made to turnitin.com but results are not analyzed.	Made to turnitin.com and results are cursorily analyzed.	Made to turnitin.com and results are analyzed thoroughly.
Meta-cognitive Reflection	Not performed.	Is shallow and incomplete.	Is complete but not thorough.	Is complete and thorough.

19 Submission

Upload the PDF file to the muOnline system.