CSCI 3700 - Database Management Systems
Data-intensive Application Development
Term Project

Nicholas Ellis

11/19/18

Contents

1 Data-intensive Application Selection

What are its characteristics?
A record company needs a way to manage its music inventory.
What makes it data-intensive?
It is data intensive because it handles huge amounts of data - namely songs. With songs constantly being added/removed from the database.
Who is the sponsor of this project?
A record company would be a possible sponsor for this project
Who are the end-users?
The end users would be the record company since they will be the ones adding/removing songs from the database.
How will they benefit from this application?
They will benefit from this because they can easily keep track of information on their songs.

2 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.

How may user classes do you have?

- 2

How many actors (including human, application, and abstract ones like time)?

- Human
- Execution time

Name use-cases. Document them using the LaTeX template.

- View Songs
- Delete Song
- Add Song
- Sort Songs

3 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.

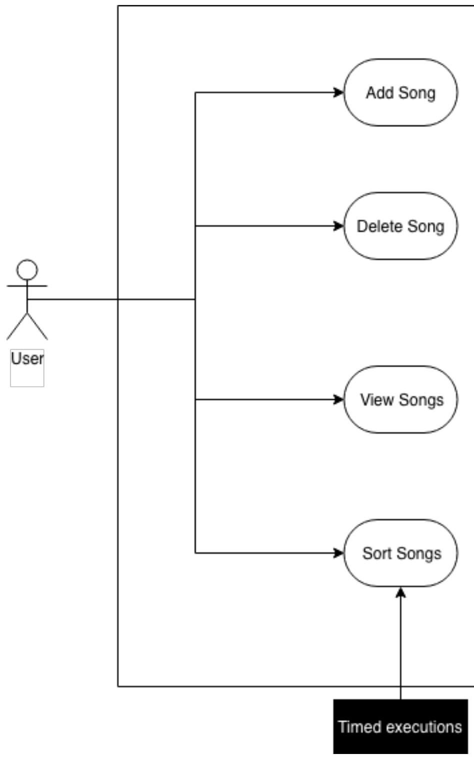

Figure 1: Use-case diagram

4 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.

Transactions:

- Add Song
  - Input: New data
  - Error: Wrong type of data
- Delete Song

  - Input: Data that needs to be deleted
  - Error: Data is already deleted

- View Songs
  - Input: Data that needs to be pulled to viewed
  - Error: Pulls the wrong information

- Sort Songs
  - Input: Which data to sort by
  - Error: Not a valid data type to sort by

5 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 all or nothing proposition. For each transaction specify its frequency of execution.

- Add Song
  - This will need to be done every time a person wants to delete data
- Delete Song
  - This will need to be done every time a person wants to delete data
- View Songs
  - This will need to be done every time a person wants to view the data
- Sort Songs
  - This will need to be done every time a person wants to delete data

6 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.

- A person sorts the songs
- A person adds a song
- A person deletes a song

- A person gets information on a song
- A person gets release year on a song
- A person gets the rank of a song
- A person gets the artist of a song
- A person gets the name of a song
- A person gets the lyrics of a song

7 Conceptual Data Model

Start with Entity-Relationship (E-R) diagram 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 diagram. Follow established diagrammatic conventions. Use SQL Power Architect tool (or similar) for developing E-R diagrams.

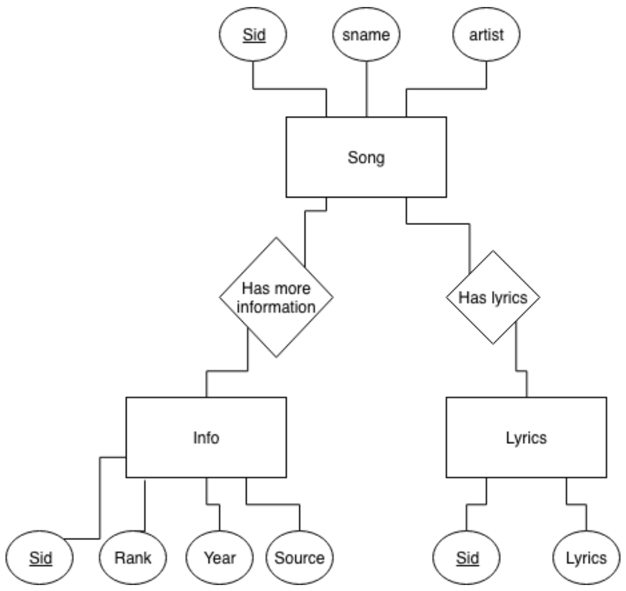

Figure 2: Conceptual Schema

8 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.

Song {sid, sname, artist}
Info {sid, rank, year, source}
Lyrics {sid, lyrics}

9 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).

Table	**Column**	**Data Type**
Song	sid	Integer
	sname	String
	artist	String
Info	sid	Integer
	rank	Integer
	year	Integer
	source	Integer
Lyrics	sid	Integer
	lyrics	String

Table 1: Physical Data Model

10 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.

See Github link: Database creation and loading scripts

11 Implementing Database Transactions and Queries

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

Here is an LaTeX markup for typesetting SQL code.

```
SELECT name AS "Country␣Name", population AS "Population",
   lifeexpectancy AS "Life␣Expectancy"
FROM   country
WHERE  lifeexpectancy IS NOT NULL
ORDER  BY lifeexpectancy DESC;
```

For the uninitiated, you may use the verbatim command for an uninspiring typeset.

See Github link: Implementing Database Transactions and Queries

12 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.

Discuss the design and implementation details of the database application here. Please do not include actual code. You may include code in Appendix.

I chose Java for my application because I have had previous experience writing in Java. A web application for this project would be a simple webpage. This webpage has a textbox where you can interact with the database via queries. There is also a output from the database above the input textbox.

13 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.

Include information on: who critiqued your document, what suggestions were made, and how you incorporated the suggestions and revised the document.

My friend, Kathleen, critiqued my document. She said that the file format of the document did not seem to suite this type of project very well. She mentioned perhaps using a Markdown file instead. This would make it easier to incorporate code blocks and charts without the learning curve of LaTeX .

14 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?

   Yes, I believe that I solved the right problem

2. Did I solve the problem right?

   Yes, I solved the problem by creating a database of songs make it easier to get information of songs for a record company.

3. How did I approach solutions to the problems?

   I could not get pgAdmin4 to work on my system (macOS Mojave). I resorted to using a different GUI application. This application allowed me to interacted with my database on my local machine. I was able to load data and query the database via this application.

4. What strategies and techniques did I draw upon?

   I made schemas and diagrams to aid in my development process.

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?

   I learned how to create a database, populate it with data, and perform queries on it. These are the main skills that I learned from this assignment.

6. Any other information you may wish to add ···

15 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/grader feedback is solicited, but not incorporated.	Both peer and instructor/grader feedback is solicited but not incorporated.	Both peer and instructor/grader feedback is solicited and incorporated.	Both peer and instructor/grader feedback solicited and incorporated. Evidence is presented to show how the feedback improved the document.
Meta-cognitive Reflection	Not performed.	Is shallow and incomplete.	Is complete but not thorough.	Is complete and thorough.

Use the following table to score your solution. Circle the appropriate number in each row. For example, to circle 4, use the LaTeX markup code `\circled{4}`, which produces $\textcircled{4}$.

Perf Level	*Poor*	*Fair*	*Good*	*Outstanding*
Trait				
Diction	2	3	4	[✓ 5]
Communication Style	2	3	4	[✓ 5]
Application Selection	4	6	10	[✓ 15]
Use-cases	4	6	8	[✓ 10]
Data Tasks	4	6	[✓ 8]	10
Transactions and Queries	4	6	[✓ 8]	10
Data Models	4	6	[✓ 8]	10
Creation and Loading	4	6	[✓ 8]	10
Implementing Transactions and Queries	4	6	[✓ 8]	10
Revisions	4	6	8	[✓ 10]
Meta-cognitive Reflection	2	3	4	[✓ 5]

Total score: 90 / 100.

A Code Listings

Github link: https://github.com/ellisn15/CSCI-3700-Term-Project

B Test Cases

C Other