CS 387 Project Guidelines and Expectations

Umesh Bellur

Lab Grade and Project!

- Project carries 50% of grade for CS 387
  - Assignments carry the other 50%
- 4 person team
  - Grading will be uniform for all team members

Project Goals (Any of….)

- *To construct a <u>real system</u> that uses relational DBs.*
  - *Schema + data*
  - *Front end for people to use*
  - *Transactions that bind the two.*
- *To make some <u>significant</u> modification to a DB engine such as postgres or mysql.*
- *To develop some theoretical underpinning for a concept related to DBs.*

<u>Software Engineering Exercise</u>

- Conceptualization & Feasibility
  - Teams – roles within a team
  - Choose project domain
- Specification
  - Detail the functionality
- Design
  - ER, Relational, Schema, Front end
  - 3 Tier systems are acceptable.
- Implementation
- Testing
  - Test plans, testing
- Documentation and Demo

Activities

- Identify an application area for which database systems may prove beneficial,
- Determine the functionalities of the database application,
- Model the data stored in the database (Identify the entities, roles, relationships, constraints, etc.),
- Design, normalise, and perfect the relational database schema,
- Write the SQL commands to create the database, find appropriate data, and populate the database, and
- Finally and most importantly, write the software needed to embed the database system in the application.
- The end result should be a functioning application that runs native and/or on the web and that uses your database to allow useful functionality.

6 Stages

Stage #	**Desc**	**Due on**
1	Functional Spec	Sep 22
2	E/R Model + Test plan	Sep 29
3	Relational Model + Screen Design	Oct 10
4	Normalization & Final Schemas	Oct 20
5	SQL Scripts for Schema + Data pop	Oct 27
6	Final Demo	Nov 1st + 2nd week.

Specification

- Name of the project. Name of the students in the group.
- The domain of your database application.
- What are the application specifications (i.e., what functionality will your completed system provide)?
  - These must be a numbered list from different perspectives of different types of users who will “use” the system.
- What aspects of the application will your system model? What will your system not model?
- What is the role of each project member in the project?
- What other "value-added" facilities could your system support (but that you will not build explicitly)? The goal is for us to mutually agree on a project that is feasible over the course of one semester.

Specification - FAQ

- *What do you mean by the "domain" of an application?*
- *Can you explain what you mean by "value-added facilities"?*
  - Eg: A recommender system that will recommend what to rent or buy. This is enabled by having data in a structured form in the DB.
- *In the part, "what will be modeled (and what will not)", what do you expect us to write?*
  - Eg: We will model books, authors, publishers, and printers" but not "reviews, bookstores and sales figures"

Some ideas

- **Bibliography database**
  - www.connotea.org
  - www.citeulike.org
  - DBLP etc.
- **Nobel Awards Database**
  - www.nobel.se/index.html
- **Movies Database**
  - regal.hollywood.com
  - www.imdb.com
- **A Service Bazaar**
  - An eBay for services (not for goods).

More ideas

- **Personal Photos database**

- **Home Finder**: This domain would require modeling apartments and their attributes, areas of town and their various characteristics (e.g., BEST bus lines, crime rates, distance from various landmarks). You would provide an interface for offering apartments for rent, finding apartments based on various requirements ("air conditioning + pets allowed + rent less than Rs 5000 + close to campus + modem facility").

- **Web Sites:** How do you think web search engines such as Google model their domain? You could think of them as a glorified database system where the basic entities modeled are web sites. You could then model the various properties of a web site: Topic, URL, domain name, other sites it links to, the background colour, etc. Retrieval could be for sites that have similar characteristics and properties.

- **Others**: Of course, there are a whole host of other ideas such as bank accounts, student records, election results, car rentals, auto insurance, consumer products, courses at IIT, Indian Cricket statistics, "match-making services" and so on. Let your imagination run riot!

Stage 2 - E/R Modeling

- Starting from the FS that you turned in for Step 1 of the project, design an E/R diagram for your application. Your model should provide

  1. 8-10 entity sets,
  2. a similar number of relationships,
  3. one example of a multi-way relationship, and
  4. one example of inheritance, and
  5. (preferably) one example of weak entity sets.

Your design must satisfy the first two criteria. Satisfying the last three criteria is *optional*. If your design does not satisfy any of the last three criteria, you must state in your response, why you think it is "unnatural" for your application to involve multi-way relationships, inheritance, and/or weak entity sets.

Stage 2 - Test plan

- For every numbered use case, create a set of tests that will be used to verify whether the feature works correctly.
  - You need to test *correctness* <u>**AND**</u> *exception* handling.
  - Test *parameter verification* <u>**AND**</u> *backend functionality*
  - Include the kinds of test data you intend to use.
- Output is a document of test cases
  - Spreadsheets are a easy way of organizing this.
- Submit a document

Stage 3 – RM and Screen Design

- Convert the ER to a relational model. This will NOT be normalized.
  - Deviations from the standard conversion procedure as described next.
  - Submit a document.
- Screen designs
  - You can do them in HTML, Powerpoint, Visio or any other tool. This is to get an idea of the **layouts** and the **flow** of screens.
  - Throwaways may be easier to get an idea of.

Relational Model

- In your conversion, deviate from the procedures discussed in class as described below. These deviations deliberately create a bad relational design so that you can use normalisation techniques in later project assignments to improve the design.
  - Pick one many-one relationship (let us call it R) in your E/R diagram. Do not create a relation for R. Suppose R is many-one from an entity set E to an entity set F. In class, we learnt how to merge the attributes for R (and for F) into the relation for E. In this step, you should merge the attributes of the relation you would normally have created for R into the relation for F.
  - Pick one many-many relation (let us call it S) in your E/R diagram between entity sets G and H. Do not create a relation for S. Merge the attributes you would normally have created in the relation for S into the relation for G or for H.
- Describe in words the types of problems you have introduced by deviating as instructed. Explain in terms of the attributes of the relations you have created.
- Include the relations so created in the set of relations that result from the previous step. Be sure to mark which relations you applied the deviations to with text such as "Relation for many-one relationship R resulting from deviation."

Relational Model (2)

- If you do not have a many-one relationships, pick two many-many relationships to apply the deviation to. Similarly, if you do not have a many-many relationship, pick two many-one relationship to apply the deviation to. If all but one relationship are one-one, then your E/R diagram has problems we should have addressed already.
- If you have inheritance in your E/R diagram, provide two sets of relations for the entity sets in the IS-A hierarchy and the relationships involving these entity sets. For one set of relations, use the E/R technique described in class for conversion. For the other set of relations, use the OO method described in class. Label each set of relations clearly with the method used for conversion.

Stage 4 - Normalization

- Are all the relations in your chosen schema in 3NF? Are they are in BCNF? Explain both answers.
  1. If any of your relations are not in BCNF, normalize them to BCNF. If you choose to normalize your relations only till they are in 3NF, explain your reasons (e.g., the amount of redundancy introduced is limited or some other valid reason).
  2. Notice that if you decide to go ahead with normalization, you will have to list the violating FDs for each of your relations and explain why you think they violate 3NF and/or BCNF.
- At this stage, if your relations are not in 4NF, normalize them into 4NF.

Stage 5 - Schemas

- SQL Scripts to create the DB schema you need for your project.
  - Constraints
- Data population scripts to populate the DB with test data.
  - Test data creation
  - We are looking at data in the order of 40-50 tuples for each relation in your application. Get it from some web source (making sure that it is public first, of course) or write a program in any scripting/programming language (like Perl, TCL or C) that creates large files of records in a format acceptable to the bulk loader. Either way, realize that you may need to transform data from one form to another to be acceptable for use in PostgreSQL.

Stage 6 – SQL + Front end

- Write the SQL that will allow you to build the functionality of the application.
- Create the front end (JSPs, PHP, HTML5, Ruby) and hook it up to the SQL using DBC.
- This is the “implementation” stage of your project.

More on complexity

- Your queries should preferably illustrate several different aspects of database querying, such as:
  - Queries over more than one relation (by listing more than one relation in the FROM clause)
  - Queries involving aggregate functions, such as SUM, COUNT, and AVG
  - Queries involving complicated selects and joins, and/ or sub-queries
  - Queries involving GROUP BY, HAVING or other similar functions.
  - Queries that require the use of the DISTINCT or ALL keyword.

A word of caution

Do not cook up query problems to cover each and everyone of the above aspects! You do not have to illustrate all the above aspects, just the ones that occur *naturally* in your application. Try to infuse some reality into your project and think of some reasonable queries that people would want to make with your application. For example, in a movie application, writing a query such as "Find all actors whose ages are three times more than their street number" sounds ridiculous!

Stage 7

- Run through your test plan and fix issues that you can.
- Submit a final copy of the test status against your plan and summarize against requirements.
- Create a presentation about your project (20 slides at most).
- Demo it in the class.

Grading Guidelines

Stage #	**Desc**	**Due on**	**Weight (%)**
1	Functional Spec	Sep 22	10
2	E/R Model + Test plan	Sep 29	15
3	Relational Model + Screen Design	Oct 10	15
4	Normalization	Oct 20	10
5	SQL + JSPs	Oct 27	20
6	Final Demo	Nov 1-14	25
7	Project Report	Nov 1-14	5

Grading is subjective (particularly in 6) – I will be the final arbiter of project grades and this will be based on my feeling on how much you accomplished in the project.