CSE 3330/5330 – Database Systems
Project description, Phases, Deliverables, and the demonstration

Sharma Chakravarthy
UT Arlington

This is a semester long single project. The project is accomplished, submitted, and graded in phases. The final phase corresponds to the demonstration of the project that brings all the parts together.

IMPORTANT – Whatever files you need to submit in each of the following phases should be put in a single zipped folder. The zipped folder should be named as "lastname_phaseX_versionY. zip" where X indicates the phase of the project. When you send the folder for the first time, Y will be 1. If you send the same phase files again due to any changes you make, increment Y by 1 each time. This will also help you keep track of your versions.

Also, only ONE submission per TEAM. NO NEED for submission by each member of the team.

Suggestion: Start a new directory when you start each phase and copy the previous phase files to avoid any confusion of submitting the wrong phase or wrong files. This is easy to do and avoids headaches down the road. We will ONLY evaluate what has been submitted!

1. Phase 1 [0%]: Problem statement

Choose an interesting problem in your domain/area of interest/expertise that requires data modeling and support for a database management system (DBMS). You should have some understanding of the problem domain or be able to get that info or research that. DO NOT choose a problem that you do not know anything about! Do not make up the problem as you go! Write a ***detailed*** description of the application under the following 3 categories as if the owner of the business is describing what the problem is and what s/he wants and NOT from a DBMS person's viewpoint:

i) General problem description

ii) Business goals or functional requirements in terms what kind of reports or information the biz owner would like to get from the system. This should not be operational aspects, such as users login, users' provided reviews, administration can do this etc. It should be more of: generate a report that contains the following information. This should be more complicated than what you already know about your business. These will be needed later in Phase 4 and 5 for writing complex SQL queries.

iii) Indicate in detail what data is of interest to the business and what aspects of that need to be captured. Again this should not contain operational information. Different components of the business in terms their data requirements need to be captured. This information will be used in Phase 2 for EER modeling

Please make sure the description is as detailed as possible so that the rest of the phases will be easier. Otherwise, you have to revise this phase until you have all the details, and cannot proceed to the next phase.

It would be useful to think about the problem from 2 perspectives: as the owner of the enterprise (for phases 1, 2, and 3) who is providing the requirements analysis and as a DBMS consultant who is going to solve the problem (for phases 4 to 5.) Phase 1 is problem description and requirements analysis where you, as the owner of the enterprise, are specifying the needs, what information you have that needs to be captured, what you want to get out of this system (reports, customer service, profit/loss analysis, info for direct marketing and incentives etc.) once it is operational. DO NOT worry about the end product at this stage or the tables or the queries.

In phases 2 to 5, you will act as the consultant who is going to solve the problem posed in phase 1. DO NOT try to predict the outcome of any phase.

See the sample project descriptions given and a complete example. More examples can be found on the url given.

DUE: See course schedule (No late submission.) This phase will be revised as many times as needed until you have the right problem statement and description to proceed to the next phase. Interaction with the instructor and TA is very important during this phase.

2. Phase 2 [20%]: EER Modeling

The TA/instructor may recommend modifications or additions to the document submitted in phase 1. After revising phase 1 as recommended, in this phase, develop an Extended Entity Relationship (EER) diagram for the problem described in phase 1. This EER modeling should adhere to the principles taught in the course. It should also use the notation taught in the course. Entities should have keys clearly identified and relationships should have the cardinality information using the (min, max) notation. The resulting EER diagram should contain:

- at least 5 entities
- at least 5 relationships (1:1, 1:N, and N:M), one of them preferably on more than 2 entities
- some attributes on relationships
- some composite/multi-valued attributes
- weak entities and weak relationships
- set/subset relationships

DUE: See course schedule (No late submission)

Submission should be in .doc or .pdf. If you draw by hand (which is fine as long it is readable), please scan and upload it. Also include the initial and approved problem statement of phase 1 and the revision used so that we have the final problem description.

3. Phase 3 [10%]: Mapping to Relations

After revising phase 2 as recommended, proceed to this phase. In phase 3, you will convert the EER diagram of phase 2 into relations using the mapping steps explained in the class/book. Make sure the conversion identifies the primary keys and foreign keys in all the relations.

Identify other keys (candidate keys) in your relations.
Identify functional dependencies in you application and list them

DUE: See course schedule (No late submission)

4. Phase 4 [30%]: Creation of the database and Execution of complex SQL queries

In this phase, you will finally transform your relations into tables in a specific Relational database management system (RDBMS) by using DDL. We will use either Oracle RDBMS on Omega for this project.

Before you write your DDL, please make sure that you normalize your relations at least up to BCNF (Byce-Codd normal form) using the techniques discussed in the class. The functional dependencies (FDs) you listed in phase 2 come in handy here. Please make sure all FDs applicable to your problem (based on the data and application semantics) are captured by your design.

Make sure you prefix each table as <semYear_teamNum_>. For example, for a table named Employee, if your team number is 8 and you are taking this course in Fall 2018, the prefix will be F18_8_Employee. This should be followed for all tables.

Make sure you transform all your constraints into appropriate constraints on the corresponding tables. (using check and assertion statements) Please make sure that you define triggers where appropriate for managing referential integrity constraints and other application specific situation monitoring.

Step 1: DDL Script [myDBcreate.sql]

Make a file containing the SQL statements that create your entire database schema, name it as myDBcreate.sql. This includes the tables with their constraints, and views, indexes, and other objects if you have them.

Make sure you create the tables that do not reference any other tables first. Otherwise, if you give a CREATE statement and the referenced object/table does not exist, the CREATE will fail. If you need to grant privileges on your tables, that should be part of the DDL script.

Step 2: DML Script [myDBinsert.sql]

Make a file containing INSERT statements that populate the tables created in step 1, name it as myDBInsert.sql. This script will contain SQL commands to fill data in your tables. You should insert about 10 to 20 rows in each table. The more the better.

Step 3: DML Update Script [myDBupdate.sql]

Create a script that will update the database through a series of insertions, deletes, and updates. This will allow you to verify the correctness of your queries when the database is updated/changed. Make sure your updates have some effect on the results of the queries! It should give different results for the same queries!

Step 4: Drop script [myDBdrop.sql]

Create a script that will drop all the tables you have created for your project. This will be useful to start from a clean slate after some inserts and deletes have been added to your application to check the correctness of ad hoc queries. You should be able to clean everything through this script and re-load the database instance using the above steps. Remember that this will NEVER be done in an operational environment. Here you are learning and hence it is ok.

Again, make sure you drop tables that are not referenced by other tables first. If you try to drop a table that is being referenced by some other table, the DROP will fail.

If you see some unwanted temporary tables in your database, they can be removed by using the following SQL statements

Drop table <table_name> purge;   or

Purge recyclebin;

You can include these statements in your .sql files to manage your database properly

Step 5: Ad-hoc SQL queries [myDBqueries.sql]

Create a script with ad-hoc queries on your database, say myDBqueries.sql. This script should contain at least 5 queries on your database. Use the comment facility in sqlplus (starting a line with --) to write the English version of your query, followed by the SQL version of the query. Also show the expected output in the file (as was shown in the homework example). These queries need to satisfy the following. We will go over these classes of queries in the class.

- No select * from table name queries!
- Starting Fall 2017, you will be required to write at least 6 queries in 3 categories as follows:
  - Group by and aggregate queries
    - At least 2 queries on multiple relations containing GROUP BY, HAVING, and aggregate computations
- At least 2 queries that corresponds to data analysis (as in a data warehouse)
    - Should have CUBE and/or ROLLUP clauses
- At least 2 queries using an OVER clause for windowing operation
- All queries should be at least join queries (some involving more than 2 relations)
- Some should have logical/physical windowing
- Some of them should have nested subqueries (note the plural)
- Some of them should have a subquery in the FROM clause
- Some of them should have a subquery in the SELECT clause
- Some of them should have ORDER BY to make the output interesting and meaningful

The purpose of having you write these is to make you think about slightly complex scenarios on your database schema and have you write queries involving join, aggregation, nesting, data analytics, and stream processing that you have learned in the class.

These .sql files can be executed from the sqlplus prompt by giving the command

```
SQL> @<filename>  or
SQL>start<filename>
```

(The file needs to be in the directory from which you launch sqlplus)

Please bring all the graded material from earlier phases to demonstrate this phase of the project. Please submit/bring all the script files to demonstrate this phase of the project. If the scripts do not work as expected or give errors, we will not be able to let you debug during the demo. Please make sure they work as expected, and come prepared to execute any other queries that the TA/instructor may ask you to demonstrate.

Please be prepared to answer questions on the design and implementation of your project.

DUE: See course schedule (No late submission) + Demo during that week

5. Phase 5 [35%]: Application Development and Demo

In this phase, you will develop a front-end application that will interface with your DBMS at the backend. The user will interact with the DBMS only through this interface. The GUI will be used by the user to retrieve and update the database without having to write SQL queries. The interface will have a menudriven input through which all interaction with the DBMS is accomplished. The results will be displayed to the user as well.

As indicated in the class, you can use Java for this. If you want to do it with as a web-based interface which can be accessed from anywhere, that is fine. This may be a good alternative for distance education students who are geographically away. The demonstration of this will be easier. However, the database has to be in the Oracle or MySQL database on omega. You may choose to run a web server on any other machine you want, but the application must connect to the Oracle database on omega. The interface you develop should have the following features –

- Should be able to show the contents of relations, and update relations
- Should be able to generate reports using parameters from the GUI (As an example, if it is a library database, one should be able to search for books on different parameter values or list all books that are overdue by a week.)
- Should demonstrate the use of dynamic SQL with *prepare and execute* methods.
- Should demonstrate the use of various dynamic SQL queries including aggregates, HAVING clause, GROUP BY, and ORDER BY, CUBE, ROLLUP clauses

The emphasis in grading this phase of the project will be on the functionality and not on the GUI design. First get the functionality working and then improve the GUI if you have time. However, the GUI should be easy to understand and easy to use.

Note: Each functionality needs be implemented only once. For example, implementing insert/delete/update into one relation is sufficient. For reports and others, implement **separate** functionality and not do the same again and again. Should exercise all aspects of dynamic SQL indicated above.

- **How to demo and what to bring for the demo**

Sign up for a demo (a sign-up sheet will be made available at the appropriate time). Be prepared for the demo to last about 10 to 15 minutes. Bring along the graded E-R diagram and relational mapping. (If you have made changes to these bring the latest copy as well). If your application needs setting up on our machine, make sure it can be done in as less time as possible by automating it using compilation scripts etc. If you are using some specific web servers or tools not generally provided by omega or a standard university computer, you need to demo on your own laptop.

Distance education students need to sign up for a demo slot too. You can either demo in person during the time slot, or be on the phone as the application is being set up and evaluated. Details of how the application can be executed need to be sent earlier by email.

A general guideline for evaluating the application is as follows:

Functionality: How much functionality does the application provide with respect to the domain selected?

Robustness of the application (graceful handling of errors, improper inputs etc.)

How clearly the work is presented.

Submit all your code on bb even if we end up seeing your demo on your laptop.

Again, the DDL, DML, drop scripts and ad-hoc queries need to be delivered in addition to your application code. You can choose to demo these along with the application demo, but it is recommended that you demo them earlier. That way, the application demo will not take as long.

DUE: last day of classes (No late submission) + Demo during that week
6. Complex SQL queries [5%]

Starting this semester (Fall 2018), we plan on creating a large Database that you can write queries to understand the response times. This cannot be done on your project database as it is very small.

This is assuming OIT cooperates with us in allowing us to create a data base with ½ a Million tuples in a relation. You will be asked to write 5 to 10 complex queries after we cover SQL and see how response time varies depending upon the type of the query and the index available. Will discuss more about this as we reach that stage.

How to connect to Oracle on Omega using your PC/laptop

1. open a cmd window from your laptop/pc and do the following

   a. to open the cmd window, click on windows button on the task bar and type cmd in the search field

   b. once you have the cmd window, type
   ssh2 -L 1521:localhost:1521 netid@omega.uta.edu (e.g., sharmac@omega.uta.edu)

   c. it will ask for your netid password and you type it in.
   minimize (but do not close) the window.

2. compile the OracleJDBC.java file using the command (will be given for phase 5)
   make sure the ojdbc14.jar file is in the same directory! 9will be given to you)

```
javac -cp ojdbc14.jar; OracleJDBC.java
```

3. execute the OracleJDBC using the command

```
java -cp ojdbc14.jar; OracleJDBC
```

4. should get the following output

```
-------- Oracle JDBC Connection Testing ------
Oracle JDBC Driver Registered!
You made it, take control your database now!
Masahiro Ishida
James Borg
Franklin Wong
John Smith
Jennifer Wallace
Alicia Zelaya
Ramesh Narayan
Joyce English
Ahmad Jabbar
Amanda Rose
James Cate
Snail Eleph
Dogy Cat
```

```
Nancy Plant
Joe Nathern
Lin Kwon
Ellie Iseki
Arnold Ksdfa
Mashu Sent
Koichi Tateyama
Merry Adams
Dory Borg
Franklin Kent
Nissan Smith
Ford Mustang
Cellica Toyota
Joge Narayan
Jhon Mathews
Goro Honda
Eclipse Mitsui
Ronald Mac
```

5. you have successfully connected to Oracle in Omega and ran a Java program using the JDBC driver!

6. Follow the same for your phase 5! Good luck!

7. if you want to run phase 5 demos, please create the tables using
phase 4 files and then do phase 5