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 submit 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. All project submissions will be on the blackboard!

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**: domain and scope of the problem or business

ii) **Business goals or functional (NOT operational) 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 user login, users' providing reviews online, administration in terms who can see what, and who can what etc. It should be reports or aggragate information that you want to generate and contains useful information for understanding and growing the business. These reports or aggregate information or analysis of your business should provide insights into your business than what you already know. These will be needed later in Phases 4 and 5 for writing complex SQL queries and a graphic user interface.

iii) **Data to be captured for your business:** 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. See sample examples posted to understand this clearly.

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,) who is providing the requirements analysis and as a DBMS consultant who is going to

solve the problem (for phases 3, 4, and 5.) Phase 1 is problem description and requirements analysis where you, as the owner of the enterprise, are specifying the needs, what information/data 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.) Phase 1 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, as part of the mapping, you identify:

identify the primary keys and foreign keys in all the relations.
Draw the arrows as discussed in the class.

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 in your project, 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.

Tables have to be created in a specific order. 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 20 to 50 rows in each table. The more the better. Make sure the data inserted make sense with respect to primary and foreign keys. This is important. Otherwise, you may not get any answer when you run your queries.

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 (in the reverse order of create tables). 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 6 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 (final ones) 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.
For this phase, you need to run the ad hoc queries initially on the populated database. After that you need to update the database as specified above. After that, you re-execute the SAME queries to see what difference has been there in the output. This depends on your updates.

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 on your laptop (or some machine) that will interface with your DBMS on Omega 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. This is very similar to how you interact with the web using a fillable template. The interface will have a menu-driven 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 implement a web-based interface which can be accessed from anywhere instead of a GUI, you are welcome to do so. We will be happy to include some bonus points for this. 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 the database on omega that you created in phase 4 of this project. 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 relations in the database.
- The contents of chosen relations, and updates to them
- 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 on one relation is sufficient. For reports and others, implement **separate/different** 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.

DUE: last day of classes (No late submission) + Demo during that week

6. Home work [5%] Complex SQL queries on a Large IMDb Database

Starting this semester (Fall 2018), we have created 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.

You will write and execute at most 5 queries on this database for this homework. This home work will be given when you are working on Phase 4 to augment you ability to write complex queries and see how you can extract a small portion of a very large database using these queries.

The IMDb database has all movie and TV episode information from its beginnings. It has information on movie title, director, writers, actors, year released, genre, and more detailed information on TV episodes including seasons, details of episodes in each season.

It has Millions of movie and TV episode titles and related information. Please make sure you do not write queries that produces large amounts of output. You need to think in terms of aggregate queries so you can extract the sliver of information that you are interested in.

Note that you have to use the prefix sharmac. To access these tables. For example, to access the table title_basics, use sharmac.title_basics.

Also remember these tables are LARGE. So, be aware of the queries you test. They can produce very large output. Also, use spool to get the output in a file so it will be easier to study the results.

```
SQL> select count(*) from title_basics;

  COUNT(*)
----------
   4809386            //4.8M tuples!
```

Similarly, title_mapping has 14 Million tuples. You can access these tables by prefixing each table with sharmac. For example, to access/query the table title_basics, you have to reference it as sharmac.title_basics

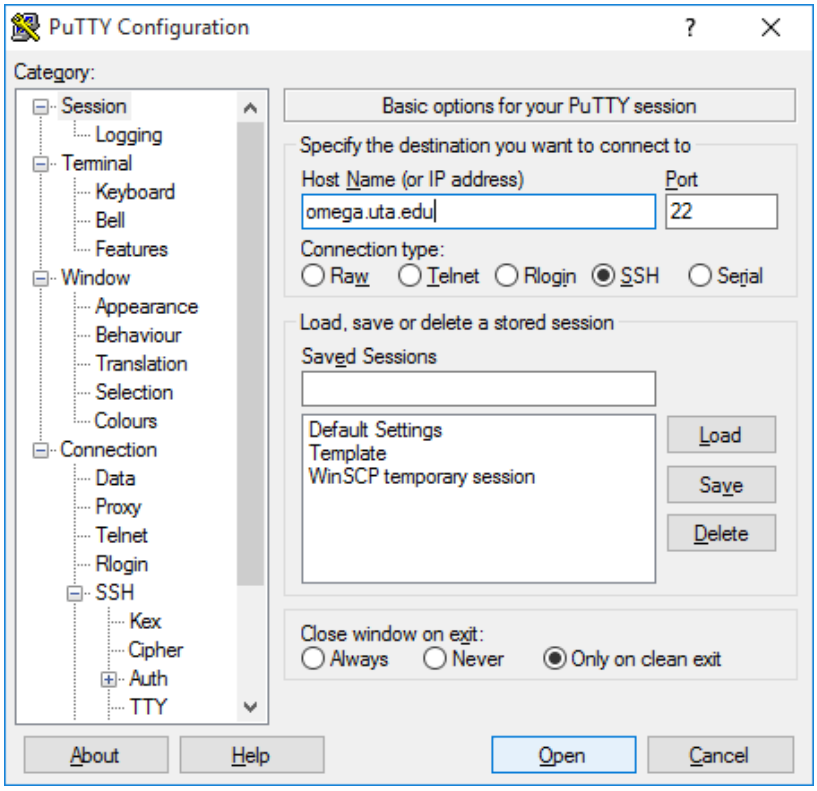
How to connect to Oracle on Omega using your PC/laptop

1. You need to have the app PuTTY on your laptop or desktop!
   a. If you do not have it, you can download it from the UTA website.

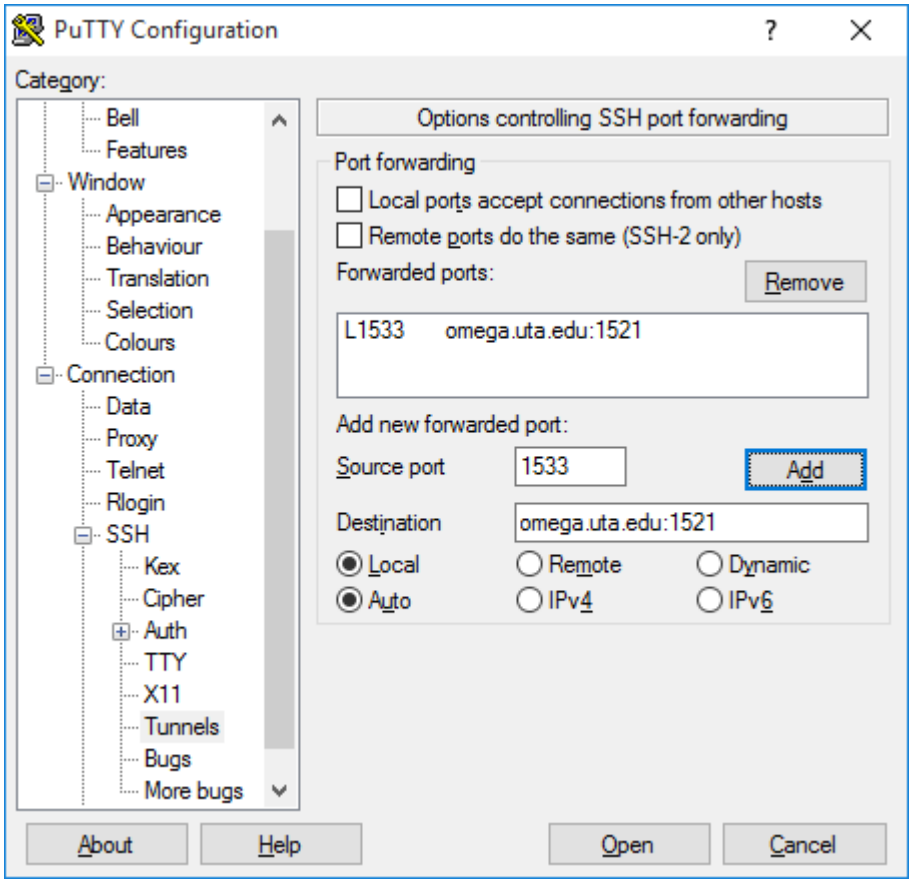
2. If you are logged in on the UTA campus network (UTA auto login), then proceed to step 4. Else do step 3.

3. Connect using UTA vpn. You will have to use two factor identification for this. May have to download Duo Mobile on to your phone for this. See UTA website for this.

4. Double click (or open) PuTTY. You will see a prompt like the one shown below.



5. Expand SSH, click on Tunnels. You should see the following:



In the above, YOU NEED TO FILL IN omega.uta.edu:1521 in the destination box and add 1533 in the source port. Click add and you should see the above.

6. Click open button. It will open a PuTTY command prompt window and will ask for your netid login and password. Make sure you do this AFTER you connect to UTA vpn. And if the vpn connection breaks, you have to do this again after the new UTA vpn connection is established.

7. Open a command prompt or ide that you use for developing your application in Java. Make sure the jdbc library ojdbc14.jar is in the same directory. Ojdbc14.jar is available on bb

Download and Compile the test java program (available on bb) Oracle_JDBC.java in the directory in which the jar file is located. Using the following commands:

8. Compile the OracleJDBC.java file using the command below. Make sure the ojdbc14.jar file is in the same directory! **You need to edit this file and add your netid and Oracle pwd (not netid pwd)**

javac -cp ojdbc14.jar; OracleJDBC_test.java //note the semi colon!

9. execute the OracleJDBC using the command

java -cp ojdbc14.jar; OracleJDBC_test //note the semi colon!

should get the following output. If not make sure you have followed the steps CORRECTLY. If it still does not work, see one of us.

```
-------- Oracle JDBC Connection Testing ------
Oracle JDBC Driver Registered!
John Smith
Franklin Wong
Alicia Zelaya
Jennifer Wallace
Ramesh Narayan
Joyce English
Ahmad Jabbar
James Borg
```

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

11. Use the code from this file for connecting to the Oracle database. The Driver.Manager.getConnection call is important and should be used with the correct argument as given.

12. Good Luck!!!!!