Project

CSE 241 and 341
Database Systems
Spring 2012

Due date for project: Thursday April 19, 2012

Goal:

The goal of this project is to provide a realistic experience in the conceptual design, logical design, implementation, operation, and maintenance of a relational database and associated applications. First, I shall describe the application, then the categories of requirements, and then some suggestions on how deeply you need to go in each category. A real project of this sort would require a substantial development team working for several months (or more). You will do this alone over several weeks. I have chosen to go with individual rather than group projects because the goal of this project is for you to gain a personal appreciation of the depth and breadth of issues that go into the design of a database application, rather than to have you specialize in just one aspect (and rely on others for the rest).

The project can go well beyond the minimal requirements I outline at the end. I encourage such extensions. Those of you who are BS in CS majors in your junior year might consider a senior project next fall based on an extension of this project.

The description given here of the enterprise you are modeling is necessarily somewhat vague and incomplete. This is by design — in real life, your "customers" are managers in the enterprise whose degree of computer literacy is, to put it kindly, variable. You will need to fill in the holes in this document to create a precise design and concrete implementation of the interfaces and applications using the database. The checkpoints specified for the project are designed to help you get some feedback along the way and keep you on schedule.

Enterprise description:

The enterprise is a package delivery enterprise (similar to FedEx, UPS, DHL, the USPS, etc.). The enterprise needs to keep track of packages shipped and keep information about customers. To find out more about this application, think about any experiences you may have had shipping packages and receiving packages, and browse shippers' web sites.

Below, we list some of the items you need to model in your database design. As you develop your own understanding of the enterprise's' needs, you may add, subtract, or alter items.

- **Customers:** Some customers have a contract with the shipper and bill their shipments to an account number. Other customers pay with a credit card. Certain shipments are prepaid by the recipient, as might be the case of someone returning something that was purchased, e.g. returning clothes that don't fit, or returning malfunctioning electronics.
- **Packages:** For the most part, the shipping enterprise does not care what is being shipped. However, there are cases where it matters. Some examples include:
  - hazardous materials
  - international shipments, for which a customs declaration stating the contents and their value is needed

- **Tracking:** The enterprise needs to track each package from the time the customer drops it off (or it is picked up) until the time it is delivered and signed for. Take a look at the online tracking offered by various shippers to get an idea of how this service works. If you are having something shipped to you, you'll find you can get every little detail of where the package is, where it has been, and to where it is currently headed. Some shippers even send email or text notifications to recipients about "significant" events. Beyond what the customer sees, the enterprise itself needs to know on which truck or plane or warehouse the package is at any point in time.

  Tracking is not just an "in the present" issue. The enterprise may want to look back in time and find out where the package was yesterday, for example. It may also want to look at data from the standpoint of a truck or warehouse (e.g., if a particular truck breaks down, it may be important to know which packages are on that truck, and if a warehouse burns down, it is important to know which packages are possibly destroyed).

- **Pricing of services:** There are a variety of shipping services offered. These include overnight shipping within the 48-state region, express shipping, regular shipping etc. There are extra charges for shipping internationally, and for shipping oversize or hazardous items.

  The exact set of services and charges may change over time.

  Furthermore, some customers may not pay the the normal default rate. Some large customers may enter into a contract with custom pricing. Regardless of the arrangements, when a package is shipped, the system must be able to compute the cost for shipping based on the type of service, the source/destination of the package, and the identity of the customer.

- **Accounting:** The system needs keep track of customer accounts.

  For credit-card customers, this is just a list of charges.

  For customers with an account with the shipper, this is not just the list of charges but also a list of payments made. Bills may be sent periodically and interest or penalties may be charges if payments are not made by the due date.

- **Other features that we shall ignore:** For this assignment, we'll consider only the package handling and billing aspects of the database. There are other aspects to the operation of the enterprise besides package tracking, some examples of which are discussed below:

  - Instead of just assigning packages to trucks and/or planes, the company would like to make the assignment in a way that permits minimal-cost routing of trucks, flying planes that are mostly full, and so on. The data in the database can be used to enable these sorts of optimizations. Making this problem harder is the fact that we cannot simply optimize over a given set of shipments. Instead, as packages arrive, the optimal strategy may change. This is an important business problem that a real system would address. I just need to keep things within bounds for a one-course project.
  - Taking the optimization problem a step further, the enterprise may recognize the value of advance planning in such matters as leasing additional trucks for peak time periods. This involves looking at past customer demand and using that to predict demand far enough in the future to enable such strategies.
  - The marketing department may want to look at customer demographics in detail and use that information to improve advertising campaigns. For example, it might target key cities for advertising or target those people most likely to use a shipping service. This involves at minimum OLAP queries, plus possibly other sorts of analysis.
  - There are other aspects of corporate operation that would be considered in a real implementation such as employee data, payroll, stockholder data, taxation, and the various reporting of financial information required by law.

What you need to do: There are several steps to this project. Although it is inevitable that you will need to go back and change things as you move along, it is desirable to do a very good job at each step so as to reduce the amount of work that winds up being redone. Here is a set of stages to follow:

1. **ER design** Construct a good, complete ER design for the enterprise. There will be a formal checkpoint in which I will review your design. It is worthwhile to refine this design in considerable detail. That makes the next steps much easier.

   To start, it is best to sketch the diagram with pencil and paper. Not only are there a lot of changes initially, but often you discover that the the placement of entity sets on the page can influence how many lines cross. Relocating entity sets physically on the page can clean up a digram considerably.

   Note that a good ER design includes careful choice of what things are entity sets and which are relationship sets, proper placement of attributes, use of generalization/specialization where appropriate, etc. A common error is to think relationally and then reverse-engineer the ER design. That approach often leads to hidden relationship sets encoded in common attributes between entity sets. Foreign keys are a relational concept; they are not a feature of ER designs.

   As you make decisions, include notes explaining the assumptions you made about the enterprise leading to those decisions. That will help you when you go back and reconsider your design. You may want to turn in some of those notes with your diagram to help me understand how you view this enterprise.

   Once your design is well along the way, you will need to create an online version. There are many ER notations in use. I ask that you use the version from the text (6th edition – we used a different style in the 5th and earlier editions, but the latest style is much easier to create online). Probably the easiest approach is simply to use Powerpoint (or Keynote or Open Office). If you use some other tool, make sure that it does not enforce some tool-specific form of diagram and make sure it allows you to export pdf. I will want all ER diagrams in pdf format when you submit them.

2. **Relational schema** The text gives a set of rules for generating a relational schema, including primary-key and foreign-key constraints, directly from the ER design. If your ER design is good, you will nearly done at this point. There may be some data dependencies that were not captured in the ER design that may lead to some further normalization. Check for this, but for a good ER design there won't be many, if any. You may decide to add some additional indices for performance. You may decide to add sone triggers or stored procedures later on. Those don't all have to be done at this point, you can always add them.

   Once you have a conceptual version of your relational database schema, you need to generate a SQL DDL version of it. That means deciding on reasonable datatypes and getting the syntactic details of SQL right. Enter this in Oracle under your account. By default only you (and me, since I am a DBA) have access to these tables.

   Note that if at any point you find flaws in your database design, you need not only to fix the design in Oracle but also make any changes that may be required in your ER design. When you submit the final project, your ER diagram must be consistent with the database you have created in Oracle.

   Hint: Don't type your DDL directly into SQL Developer. Instead create a plain text file with your DDL and copy/paste it into SQL Developer. If there are errors, edit the file and then copy/paste again. This allows you to retain a full version of your DDL for future editing without having to extract it from Oracle. Note that you have to drop tables before you can re-create them. I just put the drop statements at the start of my DDL text file. Also note that foreign keys can't reference a relation that does not exist. So be sure to list the create table commands in an order that ensures that referenced tables are created before referencing tables.

3. **Data generation and population of relations:** You need to put data into your tables. Include enough data to make answers to your queries interesting and nontrivial for test purposes, but there is no need to create huge databases.

   To avoid a typing marathon for data generation, write a program to generate test data, or use data that you can find on the web. You can get some data fairly easily without much typing. For example, you can get a bunch of names for people by doing a **select** *name* **from** *student* in the university database we are using for SQL homework assignments. Then you can write a program to pick names from that list randomly. Similarly, you can generate random numeric data. In any case, you will want to automate data loading so that if you need to redesign part of your database or your interface code trashes the data due to a bug, you can reload your data without too much effort. A trick in the automated process is working around referential integrity constraints. Inserts have to be done in the right order to avoid a foreign-key violation.

   Another possible concern is the generation of unique id values. Clever Java programming can deal with this. Alternatively, you might consider *sequence counters* created in SQL using the **create sequence** statement (see page 1043 of the text). Here is an example:

```
create table someTable (id numeric(5,0), name varchar(20));
create sequence getID;
create trigger nextID before insert on someTable
for each row
begin
select getID.nextval into :new.id from dual;
end;
insert into someTable values (null,'aardvark');
insert into someTable values (null,'bobcat');
select * from someTable;

ID                   NAME
-------------------- --------------------
1                    aardvark
2                    bobcat
```

   Oracle has a proprietary bulk loader whose use we do not cover in this course, but simply generating insert statements and running them is probably easiest.

   You are allowed to share raw data files and data generation software (see below for a full description of the collaboration rules.

   It is likely for many of you that your ER design will be sufficiently extensive that you may choose to populate only some of your relations, at least initially. This is a good strategy as it helps ensure that you are able to get some interfaces working before devoting too much time to data generation.

4. **Queries:** You should run a number of test queries in SQL to see that you have loaded your database in the way you intended. The queries listed below are those that your clients (managers from the enterprise) may find of interest. They may provide further hints about database design, so think about them at the outset of your work on this project. Note that there is no need to submit the answers to these queries; I'll run queries of my own choosing and also run your interfaces.

   - Final all packages shipped to Green Bay, WI in 2011. (Note that the default in Oracle has you list these dates as '01-JAN-11' and '31-DEC-11'. Somebody forgot to tell Larry Ellison about the Y2K problem.)

- Assume truck 1721 is destroyed in a crash. Find all customers who had a package on that truck at the time of the crash. Find all recipients who had a package on that truck at the time of the crash. Find the last successful delivery by that truck prior to the crash.
- Find the customer who has shipped the most packages in the past year.
- Find the customer who has spent the most money on shipping in the past year.
- Find the city with the most customers.
- Find those packages that were not delivered within the promised time.
- Generate the bill for each customer for the past month. Consider creating several types of bills.
  - A simple bill: customer, address, and amount owed.
  - A bill listing charges by type of service.
  - An itemize billing listing each individual shipment and the charges for it.

5. **Interfaces:** There are several types of users who access the database, and several applications that run on their own.

   - The database administrator (you) may use SQL either via the command line or SQL Developer. Nothing for you to implement in this regard.
   - Various managers in the enterprise need to access some, but perhaps not all of the information in the database. There may be several customized interfaces for various types of manager.
   - Customers may wish in inquire about the status of a shipment: either one that was sent or one whose arrival is anticipated. A customer could be an individual or perhaps a staff member at a company contracting with our package-shipping enterprise. You may want to design the interfaces differently for different types of customer.

   The latter two categories of users are people who don't like to write SQL. These interfaces can be built as

   - Web applications using Java applets or a scripting language.
   - A standalone Java application using Swing to create a GUI
   - Other GUI development tools you may know (but be sure they are platform independent, see note below)
   - Since this course is not a Java course, nor a GUI course, I will accept a simple command-line interface. In fact, even if you are expert in GUI development, you may want to start with a simple command-line and then upgrade later.

   Take care in designing an interface. Even if it is just a command-line interface, it does not have to be tedious to use. Imagine a real user using the system and try to envision where the obstacles might be. You might have a roommate or friend try out your interface to get an outsider's view of whether your concept makes sense. Don't design the interface assuming the user has memorized the database, including various ID numbers. Imagine how much fun an airline site would be if you had to type in airport codes (like MCO) instead of city names (like Orlando). If I can't use your interface without opening SQL Developer and looking up values in your tables, that is not good!

   Don't forget the basics of good programming. Check user input for being valid. If you are inputting an **int** using *nextInt* first check that the user did not enter a nonnumeric character. Produce good-looking output. And so on... Then when the user does something wrong, don't just quit. Provide a chance to try again.

6. Concurrency: In real life, lots of updates and queries would be run on this database concurrently. Most likely you will not face any problems in this regard, but a poorly designed JDBC application might suffer from concurrency anomalies.

Checkpoints: There are several checkpoints scheduled. These are set in order to keep you on target for completing the project on time.

- **checkpoint 1: Feb 21.** I find it very important that I discuss your ER design with you. During the week of Feb 20, I will allocate a substantial block of time for 15 minute meetings. There are currently 52 students total in 241 and 341, so it will be a challenge to get everyone scheduled. I will have sign-ups in advance for specific time slots and will try to stay on schedule to avoid queuing. Of course, we can schedule additional meetings as needed.

  By the start of class on Feb 21 or the time of our scheduled 15 minute meeting, whichever comes first, you must have a pdf file with your ER diagram submitted on coursesite. Bring a hard copy of your most recent ER design with you to our meeting.

  It is to be expected that I will suggest changes. The 1 point out of 33 for the project allocated to this checkpoint will be awarded for a good ER diagram. It need not be perfect to get the full point. The ER grading will be more stringent in the final version submitted at the end of the project. Thus, a perfect score on this checkpoint is not a guarantee of a perfect score for the ER component on the final version of the project.

- **checkpoint 2: Mar 15.** At this point you should have your relation schemas created in Oracle. I will looking online to see that they are there and include reasonable key declarations (including foreign keys). I don't expect to see "fancy" features like triggers or stored procedures, but it is certainly fine if they are there. My review will be cursory (I have quizlets to grade at the same time), so if you have questions, you should be sure to ask.

  The 1 point out of 33 for the project allocated to this checkpoint will be awarded for a good set of relation schemas. As for checkpoint 1, it need not be perfect. I will take a more careful look at your schemas when I evaluate the final project. So, as is the case for the first checkpoint, a perfect score on the checkpoint is not a guarantee of a perfect score on the relational design component of the final version of the project.

  At this point you should also have in place a plan for user-interface development. I shall not be reviewing that plan at this time.

- **checkpoint 3: Mar 22.** I will check online to see that your relations have been populated with data. I will award the one point allocated for this checkpoint if all your relations have a reasonable amount of data. Note that if you have a very large database schema (because you are doing more than the minimum), it is okay that only some relations are populated; in such cases send me a note by email before the checkpoint deadline describing your plans.

  You should have one user-interface working, but I will not check that.

- **checkpoint 4: Mar 29.** Submit on coursesite a plain text file stating your current status on the project and a schedule for project completion by the due date. No grade will be assigned to this, but failure to complete this checkpoint (or completing it with bad status) should be a warning to you that you need to step up the pace significantly.

- **checkpoint 5: Apr 12.** Submit on coursesite an updated status. No grade will be assigned. Ideally you should have at least 2 working interfaces and the others close to working. Ideally the upcoming week should be devoted to assembling the materials you are going to submit and not to a lot of last-minute debugging.

- **project due: Apr 19**. This is one week before the end of classes. The deadline is set here to allow you time to prepare for the final and to allow me time to get a good start on project evaluation prior to the final.

What to turn in:

1. Do NOT turn in a listing of all your data. I can see them online.
2. A set of sample queries you know work and that I can run as my first test suite. (just the queries, not the answers).
3. Any code you may have used for data generation. Remember to cite in your README file the source of any data generation code you did not write yourself (see collaboration rules below).
4. The code to implement the various interfaces. I will accept command-line interfaces, but encourage more elegant interfaces.
5. Please avoid platform-specific solutions. It is hard for me to debug custom installations in the time-frame I have to grade the projects. If you really want to do something different (for example, host this as a web application off of your personal web server), please talk to me well in advance so we can test/debug details well in advance.
6. In a normal business environment, you'd have your customers run your executables. Here, I am not a normal customer, since I am evaluating not only what your executable does but also the source code that was compiled to generate the executables.

   For this reason it is important that you submit your java code as .java files that I can compile and run. While I can run a .jar file, I can't easily verify that the code that I am running matches the code that I am reading; that's why I need to be able to do my own compilation.

   Another pitfall is the placement of ojdbc6.jar. If you use Dr. Java (or just use the Unix command-line to compile and run), the easiest solution is to put all your files , including ojdbc6.jar at the top level in one folder. Then I just open the folder, compile, and run. If you put ojdbc6.jar elsewhere and set your classpath, then you have to tell me where you expect it, and I have to set my classpath manually. Make it easy for me! If you are using NetBeans, you add ojdbc6.jar to your libraries. Do this using a relative, not an absolute, pathname and be sure to include ojdbc6.jar in your zip file. Then, when I open your NetBeans project, everything should be in the right place. If you use an absolute path, then I have to put ojdbc6.jar in exactly that place on my machine (something I probably can't do since you don't have an account on my machine!). Stated more generally, your goal should be that I can simply compile and run without having to get any configurations right.

   Please do not submit projects from any IDE other than NetBeans or Dr. Java. If you use any other IDE, you will need to extract your source code from the IDE and make sure it compiles and runs outside the IDE. Don't overlook this test phase. Non-running interfaces contribute zero to your grade!

   Another difference between this project and a business situation is that I want to be able to go directly into the database and see what data you have there without having to go through a filter provided by your project. That is, I will use my DBA rights on Oracle to look at your tables. This means that your data must be on the Oracle system we are using for our course and cannot be on a personal installation of a database system. (Yes, you could give me access to such a system, but unless there is some compelling reason for this, I prefer the simplicity of having your data on our course instance of Oracle.)
7. A README file at the top-level in the folder hierarchy that explains what is where, etc. Include usage instructions for the interfaces. Also include sources of all data and code obtained from others (note the collaboration rules below).
8. Create a directory with all your materials and name it using your last name (and first name if needed for disambiguation) so that it will have a unique name when I gather all the projects for review. Zip the directory with all your materials, but don't do it recursively. Everything should be in a single zip file so that when I unzip it, I can read the README file, follow the directions, and run your project without further unzipping. Submit the zip file on course site.

I apologize for being so picky regarding submission format, but I anticipate having 50+ projects to review in a tight timeframe, and I have learned the hard way that flexibility on my part can lead to problems with the grading deadline. That said, don't fret if you get some details wrong. I can deal with problems — I'm just trying to minimize them!

Grading: I shall use the following approximate template for grading:

1. Checkpoints:

   3 points, 1 each for checkpoints 1, 2, and 3.

2. ER design:

   8 points

3. Relational design, including constraints and indices:

   8 points

4. Data creation: sufficient quantity, reasonable realism, sufficiently "interesting":

   4 points

5. User interfaces, including proper features, proper updating of the database, etc.:

   10 points

6. I reserve the right to give extra points for exceptional solutions to parts of the project. I also reserve the right to deduct points in the unlikely event that I identify problems not covered by the items above.

Collaboration:

- Your project database design and interface implementation is to be your own work with no outside help except from the course TA or from me.
- You may share data to load into your database. You may also share code that generates those data. Include a note in the README file as to the source of your data and/or data-generation code; also include a note if you have given code or data to someone else.