Project

CSE 241 and 341
Database Systems
Spring 2009

- Due Thursday April 16, 2009
- Extension available for the asking to: Thursday April 23
- Checkpoint 3: Tuesday April 7, 2009
  (updated schedule, executables for at least 2 interfaces), meet with me or TA
- Checkpoint 2: Tuesday Mar 17, 2009
  (relational design, user interfaces and their features, development schedule), meet with me or TA
- Checkpoint 1: Thursay Feb 26, 2009 (E-R diagram), meet with me

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. They could turn into a senior design project or other independent work.

Application description: The application is an automobile company, such as General Motors, Ford, Toyota, or Volkswagen (or maybe a company from yesteryear like Studebaker, Hudson, Nash, or Packard).

In our hypothetical company, it has been decided to redesign a major part of the database that underlies company operations. Unfortunately, the manager assigned to solicit database design proposals is not very computer literate and is unable to provide a very detailed specification at the technical level. Fortunately, you are able to do that.

The company needs to keep quite a bit of data, but we shall focus on the following aspects of corporate operations.

- vehicles. Each vehicle as a *vehicle identification number* (VIN). Lots of stuff is encoded in real VINs (they are well described on Wikipedia), but you can just make them up if you want.
- brands: Each company may have several brands (for example, GM has Chevrolet, Pontiac, Buick, Cadillac, GMC, Saturn, Hummer, Saab, Daewoo, Holden, Vauxhall, and Opel and Volkswagen has Volkswagen, Audi, Lamborghini, Bentley, Bugatti, Skoda, and SEAT)
- models: Each brand offers several models (for example, Buick's models are the Enclave, LaCrosse, and Lucerne, and Mercury's models are the Mariner, Milan, Sable, and Grand Marquis). Each model may come in a variety of body styles (4-door, wagon, etc.)
- options: We'll stick to color, and maybe engine and transmission.
- dealers and customers: dealers buy vehicles from the manufacturer and sell them to customers. We'll keep track of sales by date, brand, model, and color; and also by dealer. This will allow us to use SQL's OLAP tools. Note that a dealer may not sell some of the car company's brands. Dealer's keep some cars in inventory. Some of course, are already sold, but the dealer still keeps track of that fact.
- suppliers: suppliers supply certain parts for certain models
- company-owned manufacturing plants: Some plants supply certain parts for certain models; others do final assembly of actual cars.
- customers: In reality, lots of demographic data are gathered. We'll stick to name, address, phone, gender, and annual income for individual buyers. The customer may also be a company (e.g. Hertz, Avis, or other companies that maintain corporate fleets, but we'll skip that).
- We'll skip data on corporate finance, pending bailouts, bankruptcy status etc. Not that these data are unimportant, but we need to keep the project within bounds.

Data Generation: For simplicity, I will not require realistic data. You can just create some names or get real ones from the car company web site.[1]

There are many different vehicles, grouped into a variety of (possibly overlapping) categories. If you get realistic here, things get to be interesting. Thanks to "badge engineering" many vehicles are the same except for name.[2] That means they can be build in the same plants from the same parts from the same suppliers. The ISA relationship will get heavy use here.

[1] But if you want real, but funny data, there are some really strange car names out there.

[2] For example, the Volkswagen Routan is really a Chrysler minivan, and the Buick LaCrosse is sold in Canada as the Buick Allure.

Client Requests:

1. E-R Model
   - Construct an E-R diagram representing the conceptual design of the database.
   - Be sure to identify primary keys, relationship cardinalities, etc.
2. Relational Model
   - After creating an initial relational design from your E-R design, refine it based on the principles of relational design (Chapter 7).
   - Create the relations in Oracle database you used for earlier projects.
   - Create indices and constraints as appropriate.
   - If as you refine your design, you discover flaws in the E-R design, go back and change it (even if the earlier design passed the checkpoint.) Your final E-R design must be consistent with your relational design.
3. Populate Relations
   - Include enough data to make answers to your queries interesting and nontrivial for test purposes.
   - You may find it helpful to write a program to generate test data.
4. Queries: You should run a number of test queries to see that you have loaded your database in the way you intended. The queries listed below are those that your client (the manager from the package delivery company) wants turned in. They may provide further hints about database design, so think about them at the outset of the project.
   - Show sales trends for various brands over the past 3 years, by year, month, week. Then break these data out by gender of the buyer and then by income range.
   - Suppose that it is found that transmissions made by supplier Getrag between two given dates are defective. Find the VIN of each car containing such a transmission and the customer to which it was sold. If your design allows, suppose the defective transmissions all come from only one of Getrag's plants.
   - Find the top 2 brands by dollar-amount sold in the past year.
   - Find the top 2 brands by unit sales in the past year.
   - In what month(s) do convertibles sell best?
   - Find those dealers who keep a vehicle in inventory for the longest average time.
5. Interfaces: There are several types of users who access the database. Each may need a special application
   - The database administrator (you) may use SQL either via the command line or SQL Developer.

- The vehicle locator service needs a lookup application to check inventory both locally and at nearby dealers. This service allows a dealer to find a vehicle match the desires of a potential customer. Marketing may want to review these inquiries to do future product planning.
- Online customers need an elegant Web interface to find dealers and check products, inventories, and prices. However, for this project, a command-line interface will suffice if your Web and/or GUI skills are not up to the challenge. (After all, this is a database course, not the Web Apps course, nor the User Interface course.)
- The marketing department needs sales reports and may want to do special data mining and analysis. Provide them with a simple interface to generate some OLAP results.

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.

6. Concurrency: The company stock will go down rapidly if the database cannot support more than one customer at a time making a purchase or if it cannot deal with more than one newly manufactured vehicle being entered at a time. Be sure the Oracle transaction mechanism is providing the needed guarantees.

What to turn in:

The checkpoints are meetings only; nothing needs to be turned in. Usually, I find the first checkpoint requires some discussion, while checkpoint 2 can often be handled quickly (even via email). Checkpoint 3 is mainly in place to ensure that the projects are on schedule (again, email will do). Please talk to me about questions at any point; don't wait for a checkpoint.

The final version of the project is to be turned in as a single zip file on blackboard. I will accept paper for the ER diagram since we are not covering drawing tools for these diagrams, but Powerpoint does work fairly well for this purpose.

1. E-R diagram, plus any explanatory notes. At minimum you must include all the entity and relationship sets implied by this handout. You may go beyond the minimum. Remember that the manager who defined the specifications is not computer literate so the specifications should not be viewed as necessarily being precise and complete.

2. Relational schema. It is likely for many of you that your ER design will be sufficiently extensive that we agree that only a part of the resulting relational design will actually

be implemented under Oracle on Edgar3. This is something on which we'll agree before Checkpoint 2. This is the point where we cut implementation effort and data-entry time to something realistic for the course time frame.

3. Do NOT turn in a listing of all your data. I can see them online if I find it necessary. *I, as DBA, will have access to your database online and will use that if your submission leaves me with some unresolved questions.*

4. A set of sample queries.

5. The code to implement the various interfaces. (By restricting your development to Java, your code should be platform independent. I will accept quite basic interfaces (command line with a modest command set), but encourage more elegant interfaces. Depending on the degree of sophistication you plan for each interface, we can agree to fewer than the 6 interfaces requested by the client.

6. Please avoid platform-specific solutions. It is a bit hard for me to debug custom installations in the time-frame I have to grade the projects. Please check with me before making any design decisions that bind your application to a specific hardware or software platform.

7. A README file in the top-level folder that explains what is where, etc. Include usage instructions for the interfaces

8. 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.

Collaboration: Your project design and interface implementation is to be your own work. *You may share data to load into your database. You may also share code that generates those data.* Please credit your source in each such case.