COMS W4111 - Introduction to Databases - Fall 2011
Project 2: Object-Relational Schema Design
Due Date : Friday December 2 at 5:00pm

Project Description

In this project, you will learn about some of the **object-relational features** of the Oracle 10g database management system and you will also implement some of these features in the system you developed for Project 1.

Oracle 10g provides several object-relational features that we ask you to explore in this project. You can find supporting documentation on these features in the Oracle Database Application Developer's Guide--Object-Relational Features. Start by reading the "Introduction to Oracle Objects" section of the Oracle manual to get a general view of the key features supported and get familiar with the terminology. Then proceed to the "Basic Components of Oracle Objects" section.

In particular, you will find the following subsections useful for this assignment:

- Object Types
- Objects
- Object Methods
- Inheritance in SQL Object Types and IS OF type
- References, REF, and DEREF

For Project 2, we ask you to modify your Project 1 schema (and associated functionality) to make use of the following four object-relational features:

1. **Object types** (also known as **row types**, see Section 23.2 of the Ramakrishnan and Gehrke textbook).
2. **Methods** (Section 23.4.1 of the textbook) on your object types.
3. **Inheritance** (Section 23.5 of the textbook).
4. **Object references (OIDs, REFs**; see Section 23.6 of the textbook).

Your grade will depend on whether you use all of these features, and whether you use these features in a natural and meaningful way. To be eligible for full credit, use each of the above four features at least once in your design.

We suggest that you do **not** dramatically change your original relational design. Simple (but meaningful) changes will do the job. As an example, several of you use an address field (of a person or some other entity) in your relational schema. The address field is usually defined as some kind of string type. The following shows how you can use the OR features above in a simple and meaningful way.

You can change the representation of the address field by defining an address **object type** which includes attributes like: street, number, state, city and zip code. Then you may define various **object methods** for address objects: for example, a method that returns true if two addresses are in the same city or in the same state, another one that returns true if an address is in some geographic regions (e.g., New England), etc. Now, you may write SQL queries that group persons according to their state and compute some kinds of statistics. To expand further on this seemingly simple example, if your design deals with foreign countries/addresses, you have an opportunity to use **inheritance** too. Not all countries have states or zip codes and even if they do the types may be different. For example, a UK address includes "postcode" and its type is variable length string, not a 5-character string as in the US' zip code. So, you may define an address object as a base type (with attributes street, number, city) and then define US address as a subtype of address with additional attributes state (a variable length string) and zip code (a 5-character string), UK address as a subtype of address with additional attribute postcode (a variable length string), etc. To expand even further on this (as we said, seemingly simple) example, why should we define the state attribute to be of type string ("Alex" is also a string but not a US state). We can model states, countries, geographical regions, and continents as objects which provides an opportunity to (a) use references (**REF**) to indicate address membership in a state, country, etc, and (b) use multiple names for the same state (like New York/NY), country (like Unites States of America/United States/USA/US), etc, to avoid the tyranny of dealing with multiple names and having to go through tedious computations to figure out if two names correspond to the same thing.

Notes:

You are not required to know PL/SQL, Oracle's server-based procedural extension to SQL, in order to complete this assignment. The object types and methods that you implement for this project will generally fit the following template, but you will most likely have to return more complex SQL expressions, of course; in particular, note that `attr_1 + attr_2` in the return statement may be replaced by any valid expression that may appear in a SELECT clause (e.g., the result of an arithmetic operation, a call to an aggregate function, a SQL case statement, etc.).

```
create type my_typ as object (
  attr_1 number,
  attr_2 number,
  member function My_Function return number
```

```
);
/

create or replace type body my_typ as
  member function My_Function return number is
  begin
    return attr_1 + attr_2;
  end;
end;
/
```

Also note:

- While you are designing an object-relational schema for this project, the design guidelines that you learned earlier in the course still apply: for example, you should define appropriate primary keys and enforce referential integrity as much as possible.
- To populate your object-relational database, you may either load your data directly into to your tables or, it may be more convenient to read your data from your existing tables from Project 1 and insert them into your object-relational schema using SQL insert statements. You may use either option to populate part or all of your schema. You will not be graded on the method you used to populate your schema, and you do not need to submit your load scripts with the project.
- You do **not** need to delete your Project 1 schema from your Oracle account in order to do Project 2. In fact, we encourage you **not** to delete your Project 1 schema, in case we need to revisit it for whatever reason later in the semester. For your Project 2 submission, however, please make sure that you only export the schema tables, etc. corresponding to Project 2 and **not** those for Project 1. Specifically, when you run the **exp** export utility, you should only list the Project 2 tables in the command line (see item 1 in the "What to Submit" section, below).
- A word about Oracle errors. You can show precise schema error information by typing "sho err" after getting error message like "Warning: Type created with compilation errors."

Accessing Oracle

As for Project 1, you will be using the Oracle servers that we have at **w4111b.cs.columbia.edu** and **w4111c.cs.columbia.edu.** Please refer to Part 2 of Project 1 for instructions on how to connect to and access the Oracle servers. Just as for Project 1, and to avoid losing any data, we strongly suggest that you periodically use the Oracle export utility, **exp**, as explained in Part 2 of Project 1, to generate a file with all your schema definitions and with the data that you uploaded to your database. You should then save this file in a safe place. For a variety of technical reasons that are not under our control, **we cannot provide backups of the contents of your Oracle account, so do back up your data frequently**.

Teams

Your mentor for Project 2 will be the one you had for Project 1.

You will carry out this project with the same team you worked for Project 1. If there is a problem with your existing teammate that makes it impossible to continue working as a team (e.g., you teammate dropped the course), please send email to your project mentor right away to explain the problem and work out an alternative solution. To make any change related to your team composition, you need to get explicit permission from your mentor - if you do not, we will reduce your Project 2 grade by 25%.

Submission Instructions

What to Submit

1. **A transcript of your schema**. For this, generate the dump file **expdat.dmp** using Oracle's export utility **exp**. The export utility lets you store your database objects in an external file. Note that the **exp** command must be executed from your shell if you are using Unix, or from the command prompt if you are using Windows, **not** from within SQL*Plus. The **exp** command creates a file called **expdat.dmp** in the directory where it is executed. Here is how you should run **exp**:
    - For the Oracle server on w4111b.cs.columbia.edu:
      exp userid=<userid>/<password>@ADB2 TABLES=table-1,table-2,...,table-n ROWS=N
      where **table-1**, **table-2**, ..., **table-n** are the names of the **Project 2** tables that you defined in your database; note that you do **not** need to include any auxiliary/temporary tables that you **only** created to populate the object tables
    - For the Oracle server on w4111c.cs.columbia.edu:
      exp userid=<userid>/<password>@ADB TABLES=table-1,table-2,....,table-n ROWS=N
      where **table-1**, **table-2**, ..., **table-n** are the names of the **Project 2** tables that you defined in your database; note that you do **not** need to include any auxiliary/temporary tables that you **only** created to populate the object tables
2. The following **queuries.txt plain-text file:**

- The **queries.txt** file is a plain-text file that contains "interesting" SQL queries over your database, with a sentence or two per query explaining what the query is supposed to do and what features of your object-relational database is demonstrating. **You will be graded on these queries.** They have to work and they have to show the use of object-relational features in your application. There must be execution examples for all object methods you implemented.

3. A **README** file containing the following information:
   - Your name and your partner's name
   - A list of all the files that you are submitting
   - Your Oracle account name (**not** the password), so that we can check your schema implementation
   - A clear description of your schema design decisions, listing explicitly which object-relational features of Oracle you used, where, and why
   - Any additional information that you consider significant

How to Submit

1. Create a directory named **<your-UNI>-proj2**, where you should replace **<your-UNI>** with the Columbia UNI of one teammate (for example, if the teammate's UNI is **abc123**, then the directory should be named **abc123-proj2**).
2. Copy all the files described in the "What to Submit" section above into the **<your-UNI>-proj2** directory.
3. *(Unix option only)* **Tar** and **gzip** the **<your-UNI>-proj2** directory, to generate **a single file <your-UNI>-proj2.tar.gz** (containing the files in the "What to Submit" section above), which is the file that you will submit.
4. *(Windows option only)* Zip the **<your-UNI>-proj2** directory, to generate **a single file <your-UNI>-proj2.zip** (containing the files in the "What to Submit" section above), which is the file that you will submit.
5. Login to Courseworks at https://courseworks.columbia.edu/ and select the site for our class.
6. Select "Class Files" and then "Post File."
7. Enter **"Project 2"** in the "Title" field and select your **<your-UNI>-proj2.tar.gz** file (from Unix) or your **<your-UNI>-proj2.zip** file (from Windows) in the "File" field using the "Browse" button.
8. Select **folder "Project 2" (under "Shared Files")** in the "Post File To" field.
9. Enter the names of the members of the team and their UNI in the "Comments" field.
10. Hit the "Submit" button.

Grading

Your grade will be based on the following factors:

1. Quality of your schema design and implementation, including your use of object-relational features of Oracle. In particular, we are going to look at if and how you have incorporated into your design each the four OR featured: object types, object methods, inheritance, and object references.
2. Quality of the submitted documentation and the README file.
3. Execution of the queries in the queries.txt file.
