CST2355 – Database Systems Group Lab Assignment 2

# Assignment Overview:

Group Lab Assignment 2 will give you a chance to work in a team of 2 or 3 students to create a database application consisting of a Microsoft Access application that uses a separate **Oracle 19c** database to store application data in linked tables rather than in the Microsoft Access file itself.

You will add to three of your original tables from Group Lab Assignment 1 by creating a set of ‘views’ in Oracle that when queried have all the fields from your original tables. You will be doing this step for at least two of your original tables. This means that your previous Access application should work if the linked tables are updated to point to the new Oracle Views.

You will also be creating a set of triggers (and stored procedures if desired) to allow INSERTS, UPDATES, and DELETES to the views so that if a value of a field gets changed, a history of those changes is kept in a new set of tables to store the multi-valued fields. We will be putting timestamps on the updates, so that we have a history of when each value was valid.

# Details of the Required Historical Data Framework Requirements:

Let’s start with an example. If you have an original table with two multi-valued fields (e,g, person – containing an ID (primary key), and a FIRSTNAME and LASTNAME fields). That table could be replaced with a view (e.g., person\_view) that is defined as a SELECT statement from a join across a set of new tables. The table would get replaced with a table of the same name that has only the primary key (and any other single-valued fields that might have originally been in the table). We end up with: the original table (with just the key and the single-valued fields); a new view; a new table for each multi-valued field; and an association table that links the new table to each of the multi-valued fields. We could use a view defined as shown directly below. (using PERSON, FNAMES, LNAMES as the underlying tables, and the two association tables as PERSON\_FNAME and PERSON\_LNAME)

Example view (you need to REALLY understand this code….):

CREATE VIEW PERSON\_VIEW AS

SELECT PERSON.ID, FNAMES.FIRSTNAME, LNAMES.LASTNAME

FROM PERSON

LEFT JOIN PERSON\_FNAME

ON PERSON.ID = PERSON\_FNAME.PID

LEFT JOIN FNAMES

ON PERSON\_FNAME.FID = FNAMES.FID

LEFT JOIN PERSON\_LNAME

ON PERSON.ID = PERSON\_LNAME.PID

LEFT JOIN LNAMES

ON PERSON\_LNAME.LID = LNAMES.LID

WHERE

(person\_fname.ENDTIME is NULL)

AND

(person\_lname.ENDTIME is NULL);

Each multi-valued field was replaced with a new association table to store the one-to-many mappings, and a new table to hold the values as they are updated. In addition, the association table should have an additional field containing the “STARTTIME” (the timestamp from when the value was inserted or given the linked value) and an additional field containing the “ENDTIME” (the timestamp when the value was changed to a new value or DELETED.

OPTIONAL: You can add another field to the association tables hold a text field to store ‘Notes’ about the change – allow this field to be default NULL.

# New Requirements to Extend Group Assignment 1 to Group Assignment 2:

1. Introduce at least one multi-valued historical data field in a table that is part of each of the following 3 types of relationships: Is-a, contains, is-related-to. This will mean that you have to update the forms that use those 3 tables to use your new view instead of each of the original tables.
2. For each of the multi-valued field examples get your original group assignment 1 application still able to work using the new VIEWs. You will need to build a set of INSTEAD OF triggers for each these new views to handle the SELECT, INSERT, UPDATE, and DELETE commands sent from your application to the database.
3. Update your forms for editing each of the three original tables to have buttons or some other forms so that you are able to look at the historical data for each field in a new pop-up form specific to that field (with start and end timestamps). You could do this by having a button such as “View Name History” which shows a form containing the name history data.

# Steps (Similar to Group Assignment 1):

1. The first step is to confirm the membership of your group. An update to your one page proposal describing your assignment topic **is required**. It should clearly state your group membership details and who is responsible for which components of the solution, and describe the data in some detail. It will be edited to become the application description and usage instruction documents in the final submission of the assignment.
   1. Group Membership: **YES** you can change group members, but this will need to be discussed with your lab professor in advance.
      1. All group members MUST be in the same lab section.
      2. Discuss topic choice in the lab and then after agreeing to work together, all members enrol in the same group # in Brightspace.
   2. List of topic suggestions: (as before)
      1. Vinyl recording collection management
      2. Spotify streaming data (XML download for individuals, import data, …)
      3. Knitting supplies management
      4. Tool rental management
      5. Doll collection management
      6. Guitar collection management
      7. Appointment scheduling for piano tuners
      8. Appointment scheduling for appliance repair
      9. Safety inspection scheduling in apartment buildings
      10. Any interesting example that has the required types of relationships….
2. Proposal Template: (1 or more pages)

Section 1: Introduction:

* Describe the topic; for example, an application to manage a personal vinyl recording collection. Some details of the information to be stored and how ‘hierarchies’, ‘is-a’, ‘contains’, and ‘related-to’ relationships are dealt with in the data. Try to limit the number of tables but cover all the required relationship types.

e.g. For a vinyl recording management application: a list of artists, each artist is-a ‘type’ of artist (e.g., song-writer, musician, singer, …), artists are related-to vinyl albums, each vinyl album contains recordings, individual musicians, singers, etc are related-to each recording. A song can be recorded by several musicians on different vinyl recordings. Etc. etc, You will need to build a precise model later, but at least give the professor an idea of what you are planning to store and manage,

Section 2: Group Membership and Tasks:

* List of members and email addresses;
* Table showing who will do which pieces of the assignment. You need to have only one individual per piece.

1. The demo will be during the last lab session of the course. The lab professor will spend 5 or 10 minutes to have a quick look at the application and ask you any questions they might have about the application and how you implemented it. PLEASE plan your demo carefully so that you are able to show the deliverables quickly and demonstrate that you should receive full marks. After the demo, one of your group members should submit the 6 files to Brightspace for grading.
2. You will need to plan your time carefully, and not leave too much for the last couple of weeks. In the end, there will be 6 files to hand in (see below). The files should be submitted using Brightspace by one of the group members, in a single submission .zip format. Normally, all team members should have a chance to review the files before they are submitted, and all team members will receive the same grade,

Marking Scheme:

The assignment will be graded out of 15 points, using the following breakdown:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| TASK | | | | | | MAX |
| Front End | |  |  |  |  |  |
|  | • 2 points for set of forms to update ALL underlying tables – including new multi-valued | | | | | 2 |
|  | • 2 points for set of reports to navigate and display relevant data (esp. hierarchies) | | | | | 2 |
|  | • 1 point for set of reports &/or forms for convenient updates/inserts/imports | | | | | 1 |
|  |  |  |  | Task Total |  | 5 |
| Physical model | |  |  |  |  |  |
|  | o 2 points – Covered all the minimum required entities; hierarchies, is-a, contains, related-to | | | | | 2 |
|  | o 1 point – appropriate use of surrogate keys | | | | | 1 |
|  | o 1 point - Relationships match business rules; with correct cardinality | | | | | 1 |
|  |  |  |  | Task Total |  | 4 |
| Data logistics | |  |  |  |  |  |
|  | • 1 point - test data complete, useful | | | | | 1 |
|  | • 1 point – Test Data and database creation script supplied | | | | | 1 |
|  | • 1 point – Database creation steps identified, concise, clear (Users, Instance names…) | | | | | 1 |
|  |  |  |  | Task Total |  | 3 |
| Demo/Documentation | | |  |  |  |  |
|  | • 1 point - planning - 6 Files: | | | | | 1 |
|  |  | MS-Access, Oracle database creation script, Data Model Diagram | | | |  |
|  |  | 1 pager database creation instructions, 1 pager application description, 1 pager usage instructions; | | | |  |
|  | • 2 points - execution | | | | | 2 |
|  |  |  |  | Task Total |  | 3 |