

**AUSTRALIAN NATIONAL UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE**

COMP2400

Assignment 3 – External Schema

Deadline : 5.00pm Friday 25 October 2002

The database for this assignment is part of one which might be maintained for managing student marks in a course. Suppose the relational database schema for the base tables are as described below. The base tables ‘belong’ to the DBA who manages the database and the default situation is that the base tables may be accessed directly only by the DBA user.

Tutorial (TutId, Day, Time, Room, TutorName)

A3Student (StudentId, Surname, FirstName, Username, TutId, Withdrawn)

AssessmentItem (AId, ADate, TotalMark, FinalPercent)

Mark (StudentId, AId, AMark, DateSubmitted)

Withdrawn gives the date of withdrawal for a student who has withdrawn and is NULL for a currently enrolled student. **ADate** is the due date of an assessment item. The **TutId** (Tutorial Id) is a character string identifying the tutorial. The **AId** (Assessment Id) is a string with values ‘a1’, ‘a2’, *etc.* for assignments, ‘wk2’, ‘wk4’, *etc.* for tutorial attendance and ‘exam’ for the final exam.

TotalMark gives the total number of points received for a ‘perfect’ paper (for example 60 for a1, 100 for exam), while **FinalPercent** gives the contribution each assessment item makes toward the final grade (for example, if the exam counts for 50% of the final grade then FinalPercent for exam will be 50). The **AMark** field of a row in the Mark table will have a value less than or equal to the corresponding TotalMark. There will be one row for each submitted assessment item, *i.e.* if a student has not submitted an assignment, there will be no row in the table for that student with that assignment.

The file *createa3basetables.sql* in the COMP2400 public area will define the database. You can use this file to create your own copy of the base tables, to experiment with and to test your assignment on. In order to allow your *views* to have obvious names, all the base tables will have names prefixed with ‘**B_**’ (for base table).

The activities of Assignment 3 would normally be carried out by a DBA, who must define appropriate **views**, create appropriate **roles** and grant appropriate **permissions** for each of the categories of users of the database described in Part A. While COMP2400 students do not have DBA authority, you do have authority to create views on tables you own and you will also be given authority to create roles and to grant permissions to roles and to users on objects (tables, views, roles) that you create. Since roles are global objects, in order to make sure that you don’t get in each other’s way, **every role you create must be prefixed with your system username**, *e.g.* U3012345_PUBLISHER (as for the lab in Week 10).

Part A – VIEWS, ROLES and GRANTS

[40]

There are various legitimate users of this database with differing requirements.

The database and the categories of users and their activities (below), are deliberately limited mainly to keep the amount of work in this assignment under control. You may wish to write a paragraph suggesting improvements. Do not, however, add functions far afield from what is suggested below.

- 1. Public** – Anyone (who can connect to the database) can see (**select** from) the following ‘table’:

Tutorial with columns **TutId, TutorName, Room, Day, Time**

Besides read access of public views, the following categories of users have additional privileges. All of the users below have read access to all of the AssessmentItem information. They also have access to summary information about assignment marks (not other kinds of marks), that is, the minimum, maximum, and average marks for the whole class for each assignment as well as the total number of submissions for each assignment.

- 2. Student** – Student users have read-only access to the information about themselves in the A3Student table and the Mark table, excluding the exam mark. Student users have usernames like **student<StudentId>**.
- 3. Tutor** – Tutors need read access on the A3Student table for information about the students in their tutorials. They also need write access (insert, update, delete) on the Mark table, but only for students in their tutorials. Besides the class summary information available to other users, tutors would also find it useful to see the mark summary data by tutorial group; each tutor may see the averages for all tutorial groups. Tutor users have usernames like **tutor<TutorName>**
- 4. CourseManagers** – Only course managers make changes to the data in the Tutorial, A3Student, and AssessmentItem tables. (Of course, they can also update the Mark table.) They would like to have a view showing the final mark for each student (using the information in Amark, TotalMark and FinalPercent). Course managers have user names like **manager<StaffId>**.

Since you do not have the authority to create Oracle users, I shall set up some example Oracle users Student2010123, Student2010999, TutorSmith, TutorJones, Manager8100234, Manager9100888. They will have passwords the same as the usernames. Note that usernames are stored in Oracle system tables in uppercase, and that the value returned by the Oracle function USER is in uppercase. To test a role granted to a test user, you will need to connect to Oracle as the test user and enable the role with the SET ROLE command. There is a system limit (20) on the maximum number of roles that can be enabled for a single user in a given session. Remember that the test user will need to prefix the names of your views with your user name (e.g. OPS\$U3012345.Tutorial).

Your answers to Part A should be submitted as the two files ***createa3.sql*** and ***dropa3.sql***.

The command **start createa3.sql** should successfully create all the required roles, define all the required views, and grant all the required permissions. Please organise your answers in the following order :

- Create all the ROLES for the application, using comments to clearly indicate the purpose of each, and the users to whom the ROLES are appropriate.
- Define all the VIEWS for the application, using comments to clearly indicate the purpose of each and the users to whom the VIEWS are appropriate.
- Grant all the appropriate permissions for the application, again using comments.

The command **start dropa3.sql** should successfully **DROP ALL VIEWS AND ROLES** . (There is a 5 point penalty if your submission causes another student’s marker job to fail.)

Part B – Transactions

[10]

In practice common transactions involving database access would be implemented by ‘simple-to-use’ application programs. For example a simple transaction such as changing a student’s mark when you have the student’s name, but not the StudentId, may use a program which needs both read access and write access to various parts of the database. In the general case, a transaction may involve several database access operations and an application program would be written in a language such as C or Java with embedded database access commands.

For each of the following types of transactions, identify which base tables will be read–accessed and which base tables will be write–accessed (**insert, update, delete**). Identify which categories of users will make the transaction and what input values the user will need to produce for the transaction, noting which values are required and which are optional. State what SQL statements are required for the database access. (You may write down the interactive form instead of the embedded form. VIEWS accessible to the relevant database user should be made use of.)

1. Report the final marks for the class giving StudentId, student name, and final mark ordered by decreasing final mark.
2. Get a list of the students in a particular tutorial group.
3. Enter the marks for a particular assignment for a particular tutorial group.
4. Change a student’s tutorial, but only to a tutorial group which has fewer than 19 students.
5. Report which students received less than 50% (half the maximum possible) on a particular assignment.

The answers to Part B should be submitted as the text file called ***a3.PartB*** .