EE562
Project 2

Library Information system

Due date: 9:00am, October 24, 2017

In this project you will implement a **Library Information Management System.** The database schema for this system consists of the following five relations. The primary key of each relation is underlined.

Books(book_id: number, book_title: varchar2(50), author_id: number, year_of_publication: number, edition: number, status:varchar2(20))

Author(author_id: number, Name: varchar2(30))

Borrower (borrower_id: number, name: varchar2(30), status: varchar2(20))

Issue(book_id: number, borrower_id: number, issue_date: date, return_date: date)

Pending_request(book_id: number, requester_id: number, request_date: date, Issue_date: date). Note: Issue_date represents the date when the book is issued to the pended request. (Note: No tupple is ever deleted from this table)

Rules/Constraints:

1. Status in the **Books** relation can have only two values: **charged /not charged.**
2. Status in the **Borrower** relation can be either **student** or **faculty.**
3. Only a maximum of two books can be issued to a student and a maximum of three books to a faculty member at a time.
4. Books are to be returned within five days of their date of issue. Otherwise, a fine of $5 per day is charged for late return.
5. There is only one copy of every book. If a book is already issued it cannot be issued to another person.
6. If a person (x) requests a book which is already issued to someone, his/her request goes to the **Pending_request** table under the following conditions: The book is not currently borrowed by x and the sum of the number of unserved pending requests by x and the number of books currently issued to x is less than 7 AND no other unserved request by x for the same book is already pending. The limit of 7 is applicable to both

students and faculty. Requests **for a given book** in this table are served on a first come first serve basis.

7. If there is a pending request for an already issued book, the current borrower (x) cannot renew it, rather person (x) must return the book and his/her request is then put in the **Pending_request** table according to Rule 6.

8. In case, a borrower gets his/her book renewed, a new record is inserted in the Issue table.

9. Whenever a book is issued, a new record is added to the **Issue** table and a NULL value is assigned to **return_date**. When the borrower returns the book, this field is updated to the date of return.

Triggers:

1. Implement a trigger that enforces rule 3 in the database. Name this trigger as **trg_maxbooks**.

2. Implement a trigger that changes the status in the **Books** table to '*charged*' whenever a book is issued, i.e., when a new tuple is added to the **Issue** table. Name this trigger as **trg_charge**.

3. Implement a trigger that changes the status in the **Books** table to *'not charged'* whenever a borrower returns the book. Name this trigger as **trg_notcharge**.

4. Implement a trigger that enforces rule 7 in the database. Name this trigger as **trg_renew**.

Functions:

1. Write a function (name it **fun_issue_book**) that takes the following arguments: **borrower_id**, **book_id**, and **current_date**. This function issues a book to the requester if it is not charged, otherwise it adds the requester's record in the **Pending_request** table. The **current_date** corresponds to **issue_date** if the book is issued immediately, or **request_date** if the requester waits for the requested book in the **pending_request** table. The function will return ' 1' the book is issued to the requester, otherwise it will return '0'. (This function is called by **fun_issue_anyedition)**

2. Write a function (name it **fun_issue_anyedition**) that takes the following input arguments: **borrower_id, book_title, author_name** and **current_date**. This function will issue the latest edition of the requested book. In case, the latest edition is already issued, the next older edition that is currently available in the library will be issued. If there is no edition of the requested book currently available, the request will be put in the **Pending_request** table (through calling of **fun_issue_book**. The requester will

wait for the edition that will become available at the earliest possible time. The function returns '1' if the request is satisfied, otherwise it returns '0'.

3. Write a function (name it **fun_most_popular**) which for a given month (input as three characters, eg 'JAN" etc.), returns the book_id of the book that has been borrowed by the maximum number of borrowers (including the number of renewals). Note that multiple books can be most popular in a given month and your function should return all of them. Also, in case, entries for multiple years exist, then month-year pair should be printed out along with the book_id.

4. Write a function (name it **fun_return_book**) which takes **book_id** and **return_date** as inputs and returns the book to the library by updating appropriate tables. The function returns '1' if the operation is successful; otherwise it returns '0'. In addition to updating the **return_date** field of the issue table, this function also browses through the **Pending_request** table and checks the pending requests against the returned book. If there is any pending request, the function issues the book to the requester. If there are multiple requesters then the one on the head of the queue gets the book.

Procedures:

1. Write a procedure (name it **pro_print_borrower**) to print out current borrowers' list in the following format. The number of days equals to the difference between the **issue_date** and today's date.

Borrower Name	Book Title	<= 5 days	<= 10 days	<= 15 days	>15 days
Adah Talbot	Fundamentals of Democracy				100
Adah Talbot	Programming in Unix	1			

2. Write a procedure (name it **pro_print_fine**) which will take the current_date as an argument (the user will specify the current date explicitly). This procedure will print out the **borrowers_name, book_id, issue_date** and the **fine** paid or to be paid (if the book is not returned till to-date).

3. Write a procedure (name it **pro_listborr_mon**) which will take the following arguments as input: **borrower_id** and a given month (JAN through DEC), search the **Issue** table, and print the **borrower_id, borrower_name, book_id, book_title, issue_date** and **return_date.**

4. Write a procedure (name it **pro_listborr**) to print out the names of the borrower who have not returned the books yet (including both overdue and not overdue). Also print the **book_id** and **issue_date.**

5. Write a procedure (name it **pro_list_popular)** to display the **month, year, author_name** and the **number of editions** maintained by the library for the most popular book for every month of each year.

Execution phase:

1. Populate the **Books, Author** and **Borrower** tables.

2. Execute all the triggers.

3. Use the function **fun_issue_book()** to populate the **Issue** and **Pending_request** tables.

4. Use the function **fun_issue_anyedition()** to insert the following records in your sample database for testing. This function must take all the four parameters.

Borrower_id	Book_title	Author	Date
2	DATA MANAGEMENT	C.J. DATES	3/3/05
4	CALCULUS	H. ANTON	3/4/05
5	ORACLE	ORACLE PRESS	3/4/05
10	IEEE MULTIMEDIA	IEEE	2/27/05
2	MIS MANAGEMENT	C.J. CATES	5/3/05
4	CALCULUS II	H. ANTON	3/4/05
10	ORACLE	ORACLE PRESS	3/4/05
5	IEEE MULTIMEDIA	IEEE	2/26/05
2	DATA SRUCTURE	W. GATES	3/3/05
4	CALCULUS III	H. ANTON	4/4/05
11	ORACLE	ORACLE PRESS	3/8/05
6	IEEE MULTIMEDIA	IEEE	2/17/05

5. Execute **pro_print_borrower.**

6. Execute **pro_print_fine.**

7. Use the function **fun_return_book()** to return books with book_id 1,2, 4, 10. Also, specify the returns date as the second parameter.

8. Print the **Pending_request** table and the **Issue** table.

9. Execute **pro_listborr_mon** for the month of February and March, for a given **borrow_id.**

10. Execute **pro_list_borr.**

11. Execute **pro_list_popular.**

12. Print the average time a requester waits in the **Pending_request** table.

13. Print the **name** and the **borrower_id** of the person who has waited the longest amount of time for any book.

NOTE: Your project should NOT use any temporary table otherwise our testing script will fail and you will get no grades. (Internally you can create temporary tables, but not explicitly declared in create.sql file)

What and How to Submit:

- All the SQL commands for creating the tables and defining integrity constraints should be in one file, createtable.sql.
- The code for your triggers should be in one sql file trg.sql.
- The code for functions should be in one sql file fun.sql.
- The code for procedure should be in one sql file pro.sql
- The code for populating the **Books, Authors** and **Borrower** tables should be in one sql file populate.sql.
- The code for the execution phase should be in one sql file myexecution.sql
- You are also required to submit the sample data which you will create to test your procedures, functions and triggers in a separate file mydata.sql (primarily via loading the Issue table).
- Make sure at the end of your execution phase, you include statements to drop all the tables, triggers, functions and procedures in a separate file dropall.sql, to ensure proper testing and grading of your project.

Note:

When you are ready to submit your project, you should have a directory PRJ2 where your files are. Go to the directory which contains the directory PRJ2, run the following UNIX command

```
turnin -c ee562 -p proj2_f17 PRJ2
```

Your whole directory will be submitted for grading. You can check the submitting with

```
turnin -c ee562 -p proj2_f17 -v
```

Note:

1. Your project will be tested on an arbitrary data set, so make sure that all the data types of your tables, procedures and functions conform to the given schema.
2. Your project testing script will proceed as follows:

   - createtable.sql (provided by grader)
   - fun.sql (submitted by student)
   - pro.sql (submitted by student)
   - trg.sql (submitted by student)

- populate.sql (provided by grader)
- myexecution.sql (provided by grader)

3. You must use PL/SQL (Oracle procedural extension to SQL) to write your triggers, procedures and functions. Use the ORACLE Reference book mentioned in the handouts given in the first week of the class.

4. Please use the same version of Oracle that is installed on ECN. No grade will be awarded on the queries or procedures that can not run on Oracle version installed on ECN machines.