CS 340
Programming Assignment 2

Library Information system

Due date: 11:59 pm, November 26, 2015

Following is the schema for a **Library Information System** which consists of 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: int number, requester_id: number, request_date: date, Issue_date: date)

Rules/Constraints:

1. Status in the **Books** relation can have only two values: **issued /not_issued**.

2. Status in the **Borrower** relation can be either **student** or **faculty.**

3. A faculty cannot have more than 3 books issued against his/her name. A student cannot have more than 2 books issued against his/her name.

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 borrower requests a book which is already issued, his/her request goes to the **Pending_request** table. 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 cannot renew it.

8. In case, a borrower gets his/her book renewed, the **issue_date** is updated to the date when renewal is requested.

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 '*issued*' whenever a book is issued, i.e., when a new tuple is added to the **Issue** table. Name this trigger as **trg_issue**.

3. Implement a trigger that changes the status in the **Books** table to '*not_issued*' whenever a borrower returns the book. Name this trigger as **trg_notissue**.

Functions:

1. Write a function (call 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 issued, 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'.

2. Write a function (call it **fun_issue_anyedition**) that will take 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. 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 (call it **fun_most_popular**) which for a given month and year (take month and year as input arguments), returns the book_id of the book that has been borrowed by the maximum number of borrowers. Note that multiple books can be most popular in a given month and your function should return all of them.

4. Write a function (call it **fun_return_book**) which takes **book_id** and **current_date** as input 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.

5. Write a function (call it **fun_renew_book**) that takes the following arguments: **borrower_id**, **book_id**, and **current_date**. This function renews a book for the borrower if: 1) the book is already issued to the borrower, AND ii) there is no pending request against the given book to be renewed. If the book is renewed, the issue date of the book is changed to the current date in the ISSUE table. The function returns ' 1' if the book is renewed, otherwise, it returns '0'.

Procedures:

1. Write a procedure (call 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 (call it **pro_print_fine**) which will take the current_date as an input 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 (call it **pro_listborr_mon**) which will take the following arguments as input: **borrower_id,** a given month (JAN through DEC), and year (yyyy), search the **Issue** table, and print the **borrower_id, borrower_name, book_id**, **book_title, issue_date** and **return_date** for the given month and year.

4. Write a procedure (call 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**.

6. Write a procedure (call 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 the year.

Execution phase:

1. Populate the **Books**, **Autho**r and **Borrower** tables.

2. Execute all the triggers.

3. Use the function **fun_issue_book()** to populate the **Issue** and **Pending_request** tables with your own data.

4. Use the function **fun_issue_anyedition()** to insert the following records in your sample database for testing.

Borrower_id	Book_title	Author	Date (mm/day/yy)
2	DATA MANAGEMENT	C.J. DATES	3/3/15
4	CALCULUS	H. ANTON	3/4/15
5	ORACLE	ORACLE PRESS	3/4/15
10	IEEE MULTIMEDIA	IEEE	2/27/15
2	MIS MANAGEMENT	C.J. CATES	5/3/15
4	CALCULUS II	H. ANTON	3/4/15
10	ORACLE	ORACLE PRESS	3/4/15
5	IEEE MULTIMEDIA	IEEE	2/26/15
2	DATA SRUCTURE	W. GATES	3/3/15
4	CALCULUS III	H. ANTON	4/4/15
11	ORACLE	ORACLE PRESS	3/8/15
6	IEEE MULTIMEDIA	IEEE	2/17/15

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.

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

9. Execute **pro_listborr_mon** for the month of February and March 2015

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.

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 tgr.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.
- 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.
- **When you are ready to submit your programming Assignment, put all the files to be submitted in a folder/directory "PA2_[your login]." For example, if your login is "1610002" put the file in the folder/directory "PA2_1610002". Compress the folder into a zip file, and submit the zip file page under Programming Assignment 2 on LMS site.**

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. You must use PL/SQL (oracle procedural extension to SQL) to write your triggers, procedures and functions. Use the ORACLE PL/SQL guide and ORACLE application development guide uploaded on the LMS course site under /Resources / ProgrammingAssignment/Oracle PL/SQL and Triggers/ directory.