



CYPRUS INTERNATIONAL UNIVERSITY
ULUSLARARASI KIBRIS ÜNİVERSİTESİ

CPE-344
DATABASE MANAGEMENT SYSTEMS AND
PROGRAMMING II
Term Project

Prepared By:

Yasser El Kabbout - 20131196

Advisor:

Lecturer Salahi Halil Altıncı

June 2016,
Nicosia

TABLE OF CONTENTS

1. TABLES CREATION.....	3
2. CONSTRAINTS ON TABLES	3
2.1 PRIMARY KEYS & FOREIGN KEYS	3
2.2 NULL CONSTRATINTS	3
3. FUNCTIONS	3
3.1 IS_BOOK_AVAILABLE.....	3
3.2 MEMBER_HAS_OVERDUE_BOOK	4
3.3 MAXIMUM_BOOK_LIMIT	4
3.4 GET_TOTAL_BORROW_COUNT.....	5
4.TRIGGERS	5
5. PROCEDURES	6
5.1 BORROW_A_BOOK.....	6
5.2 RETURN_A_BOOK	6
6. JAVA APPLICATION	7

1. TABLES CREATION

The DDL regarding the tables creation was available on the Moodle's course page. To access them, the exported database is available within the project submission.

2. CONSTRAINTS ON TABLES

2.1 PRIMARY KEYS & FOREIGN KEYS

All the primary keys and the foreign keys were implemented on the tables. They can be checked within the exported database.

2.2 NULL CONSTRAINTS

All the required NOT NULL constraints were implemented on the tables. They can be checked within the exported database.

3. FUNCTIONS

3.1 IS_BOOK_AVAILABLE

The required function was implemented and it is checking whether a book is available to be borrowed or not.

Books should be checked if they are borrowed by another members, or available.

```
DECLARE
  x number;
BEGIN
  x := is_book_available(3);
  if x=1 THEN
    dbms_output.put_line('the book is available');
  else
    dbms_output.put_line('the book is not available');
  END IF;
END;
```

Snip 1 | is_book_available function in action

The function's body is available within the exported database.

3.2 MEMBER_HAS_OVERDUE_BOOK

The required function was implemented and it is checking whether the member is having a due or not.

Dues should be checked before borrowing a new book.

```
DECLARE
  x number;
BEGIN
  x := member_has_overdue_book(3);
  if x=1 THEN
    dbms_output.put_line('the user has overdue');
  else
    dbms_output.put_line('the user has no overdue');
  END IF;
END;
```

Snip 2 | member_has_overdue function in action

The function body is available within the exported database.

3.3 MAXIMUM_BOOK_LIMIT

The required function was implemented and it is getting the maximum number of books a member can borrow.

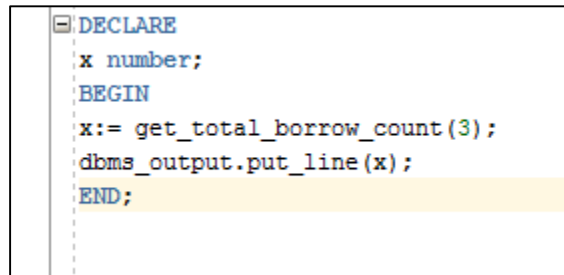
```
DECLARE
  max_number number;
BEGIN
  max_number:= maximum_book_limit(3);
  dbms_output.put_line(max_number);
END;
```

Snip 3 | maximum_book_limit function in action

The function's body is available within the exported database.

3.4 GET_TOTAL_BORROW_COUNT

The required function was implemented and it is getting the total number of books a user is currently borrowing.

A screenshot of a code editor showing the implementation of the get_total_borrow_count function. The code is as follows:

```
DECLARE
x number;
BEGIN
x:= get_total_borrow_count(3);
dbms_output.put_line(x);
END;
```

The code is color-coded: DECLARE is blue, x number; is black, BEGIN is blue, x:= get_total_borrow_count(3); is black, dbms_output.put_line(x); is black, and END; is blue. The last line, END;, is highlighted with a yellow background.

Snip 4 | get_total_borrow_count function in action

The function's body is available within the exported database.

4. TRIGGERS

A trigger is a special kind of stored procedure that automatically executes when an event occurs in the database server. In this example, a trigger was implemented to fulfill the following requirements:

1. Check if a **book is available** on selected date (Use the stored function is_book_available). If it is not available, raise an error and prevent borrowing the book.
2. Check if the member has an **overdue book** (Use the stored function member_has_overdue). In case of a overdue or non-availability raise an error and prevent borrowing the book.
3. Check if the member reached the **maximum number of books** that he/she can borrow. If so, raise an error and prevent borrowing the book.

This trigger will be triggered before inserting a new row to the table transactions.

The Trigger's body is available within the exported database.

5. PROCEDURES

5.1 BORROW_A_BOOK

The required procedure was implemented and it is inserting a new row to the transactions' table. By adding a new row, the implemented trigger will be triggered as well.

```
BEGIN  
borrow_a_book(1,2,'01-JUN-16','01-JUN-17');  
END;
```

Snip 5 | borrow_a_book function in action

The procedure's body is available within the exported database.

5.2 RETURN_A_BOOK

The required procedure was implemented and it is performing an update on the transactions table. It is updating the return_date column when a member is returning the book he withdrew.

```
BEGIN  
return_a_book(3,1);  
END;
```

Snip 6 | return_a_book in action

The procedure's body is available within the exported database.

6. JAVA APPLICATION

A simple application was implemented by using the JAVA programming language. The NetBeans IDE was used as the development environment.

The graphical user interface was implemented by using JFrames.

The main interface is as follows:

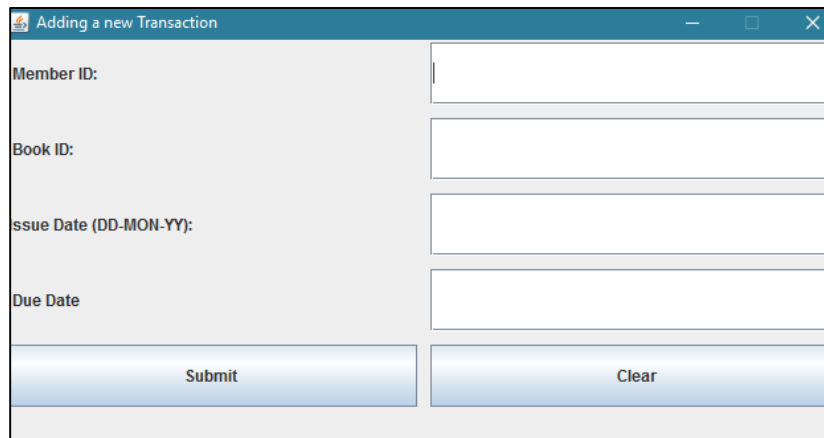
Member_id	Name	Address	Phone_no	Member_type_id
1	Yasser	Haspolat	+905428658745	1
2	Zubeyde	Istanbul	+5298498548654	2
3	James	England	+56849845485	3
4	Jafar	mohraseb	+644886464586	2
5	funda	haspolat	+65484654898787	1

Transactio...	Member_id	Book_id	Issue_date	Due_date	Return_date	Book_id	Name	Author	Edition	book_typ...	rack_id	date_of_...
1	1	2	2016-05-2...	2016-06-0...		1	the_crad...	Mary	1	1	1	2013-06-...
22	2	1	2016-06-0...	2017-02-0...		2	The_7_h...	Stephen	2	2	2	2012-06-...
						3	Peculiar...	Riggs	3	1	3	2010-06-...

Add Transaction

Snip 7 | Main user Interface

By clicking on the *Add Transaction* button, the following interface will be available:



The screenshot shows a Java Swing window titled "Adding a new Transaction". The window has a standard title bar with minimize, maximize, and close buttons. The main content area is divided into two columns. The left column contains four labels: "Member ID:", "Book ID:", "Issue Date (DD-MON-YY):", and "Due Date:". The right column contains four corresponding text input fields. At the bottom of the window, there are two buttons: "Submit" and "Clear".

Snip 8 | Adding a transaction

For this program, three classes were used and they are as follows:

1. *mainProgram* Class used as the main user interface.
2. *javaConnectDb* Class used to establish connections with the Oracle database.
3. *addTransaction* Class used to add a new transaction.

To connect to the database, the JDBC driver was used.

The full source code of this application is attached as well.