Department of Computer Science and Engineering

University of Dhaka

Project Report

CSE2211 - Database Management Systems-I

2nd Year 2nd Semester - 2018

Project Title

[Book Buy-Sell & Rent System]

Submitted By

Name: Sakib Hasan

Registration No: 2015-618-719

Roll No: 149

Index:

Introduction	3
Descriptions	4
Design Diagrams(ERD)	5
Design Diagrams(Schema)	6
Environment of Implementation	7
Application of the Database	7
Conclusions and Discussions	17

Introduction

This project includes managing a database for a book rent and buy-sell system where clients can see available book list along with availability, book details which shows book name, publisher name, price, genre, author name, total number of available books.

They can rent books for a certain time if they become a paid member of the system by paying a certain amount.

They can place order to buy books from the available ones. Admins will only be able to add/update new book info. Customers can add reviews of any books and check ratings of any specific book.

Added/available books from different publishers and authors will be shown / advertised along with proper information.

Customers will get different level of membership status according to their money spent on this system or by their donation amount. They will be gifted with discount or other facilities from the admins corresponding to other factors.

Admins can manage the list / information of the books and change or modify as needed. They can also decide managing all the customers and other features of the database.

Moreover, all the information taken from the customers will be managed in this database for the features but will not be visible to other customers.

So, in other words, these are the main information to be included into the database system.

Here, admins will be able to add books in the list, their prices, total available count of books, availability and other related information of books. They will handle the membership status and accordingly decide the discounts for clients.

Moreover, the customers can search precise books according to name, author name, publications and price as well. They can check the availability and order to buy or borrow the books. Customer information will be stored too to update and monitor them for issuing further benefits like discounts or other facility.

Descriptions

Features:

- Rent books
- Buy & sell books
- Showcasing /advertisement of books from different publishers
- Membership status levels
- Payment & billing details
- Discount on different membership status
- Customer info management.
- Book information management
- Book Reviews and Ratings management

Constraints:

*. Total payment/ donation for a customer cannot be negative.

*. All the IDs of different customers and admins will be Unique / Primary Key.

*. Total number of available books, price of a book cannot be negative value.

*. Number of books ordered must be non-zero positive number.

*. There are four level of membership status – initial is '0' with less than 500 taka revenue, level 2 (membership fee/revenue paid 500 taka), level 3 (by paying at least 1000 taka or total money paid for buy/borrow – 3000 taka), level 4 (revenue at least 2000 taka).

*. User can't input blank/ null book name/ book count. Can't order Null number of books.

*. *NOT NULL*: Customer Name, Admin Name, Customer ID, Admin ID, Contact no.

Design Diagrams

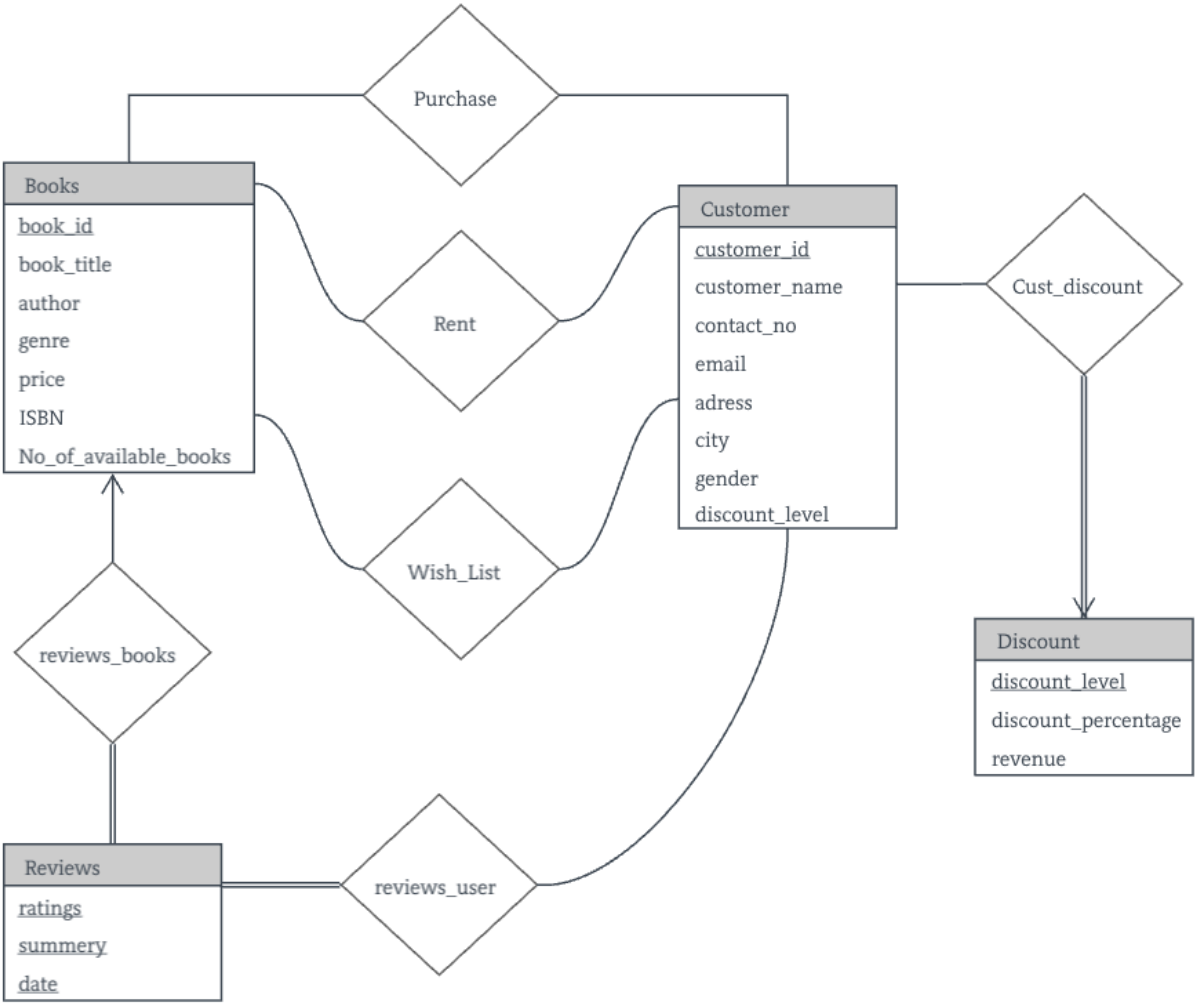


Figure: ERD Diagram of the Database

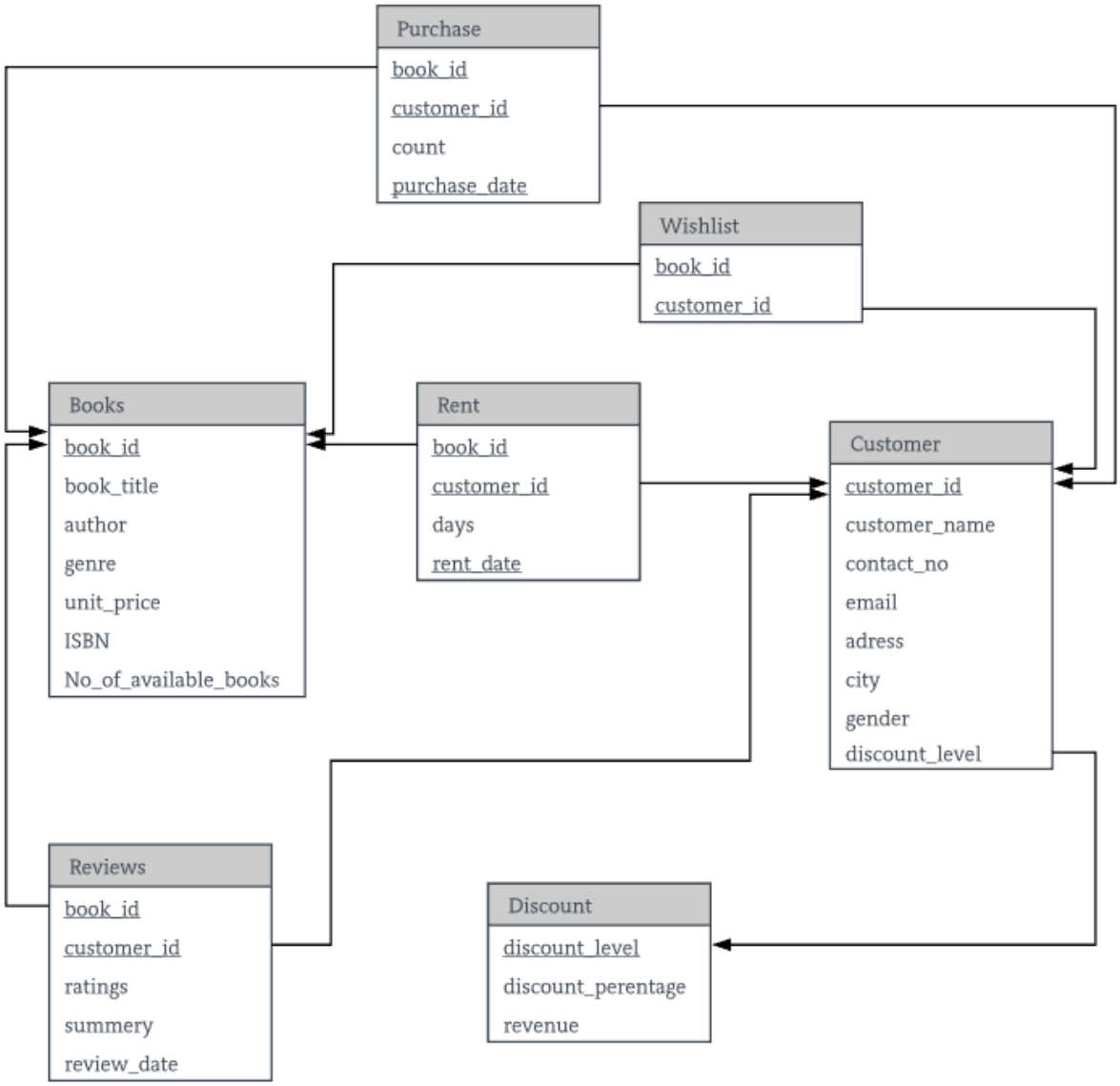


Figure: Schema (minimized) Diagram of the Database

Environment of Implementation

I have implemented this project using Oracle 11g express edition. To write the codes, test it was the best option to me. Here are the reasons why.

Firstly, though this is not totally a free product but the services it provides for free was enough for me to pull this project off. I choose this environment because it has better UI & UX than other options and I can easily have an overview on my tables, data, constraints, functions. Procedures and triggers as well as other queries and scripts. Other environments do not give me that much of user experience as it provides.

It has the most advanced analytic functions built in that leverage powerful operations. Moreover all the data, function, scripts can be found in the object browser.

It has the most powerful and organized way of managing code inside the data base. PL/SQL, packages, functions, procedures etc. Oracle empowers clean coding and the re-usability of code that interacts with data objects. MySQL empowers writing code outside the database which is farther away from the data.

DDL of the Database:

```
create table discount (
      discount_level int,
      discount_percentage int,
      revenue int,
      primary key (discount_level)
);


create table customer(
      customer_id int NOT NULL,
      customer_name varchar(40) NOT NULL,
      contat_no int NOT NULL,
      email varchar(45),
      adress varchar(128),
      city varchar(16),
      gender varchar(8),
      discount_level int,
      primary key (customer_id),
      foreign key (discount_level) references
      discount(discount_level)
);

create table books(
      book_id int NOT NULL,
      book_title varchar(128) NOT NULL,
      author varchar(64),
```

```
      genre varchar(35),
      unit_price int NOT NULL check(unit_price > 0),
      isbn int,
      no_of_available_books int NOT NULL,
      primary key (book_id)
);

create table purchase(
      c_id int NOT NULL,
      b_id int NOT NULL,
      order_count int check(order_count > 0),
      purchase_date date,
      primary key(c_id, b_id, purchase_date),
      foreign key (c_id) references customer(customer_id),
      foreign key (b_id) references books(book_id)
);

create table rent(
      c_id int NOT NULL,
      b_id int NOT NULL,
      days int check(days > 0),
      rent_date date,
      primary key(c_id, b_id, rent_date),
      foreign key (c_id) references customer(customer_id),
      foreign key (b_id) references books(book_id)
);

create table wishlist(
      c_id int NOT NULL,
      b_id int NOT NULL,
      primary key(c_id, b_id),
      foreign key (c_id) references    customer(customer_id),
      foreign key (b_id) references books(book_id)
);
create table reviews(
      c_id int NOT NULL,
      b_id int NOT NULL,
      ratings number NOT NULL check(ratings<11),
      summery varchar(255),
      review_date date,
      primary key(c_id, b_id),
      foreign key (c_id) references customer(customer_id),
      foreign key (b_id) references books(book_id)
);
```

Application of the Database

Triggers:

```
--------------------------------------------------------------------------------------------------------------------------

create or replace trigger new_Cust_trig_af
after insert
on customer
declare
n number;
begin
  dbms_output.put_line('New Customer(s) Added in Database.');
  select count(*) into n  from customer;
  dbms_output.put_line('Total after insertion: '+ n);
end ;


create or replace trigger new_Cust_trig_bf
before insert
on customer
declare
n number;
begin
  dbms_output.put_line('New Customer(s) Adding in Database.');
  select count(*) into n  from customer;
  dbms_output.put_line('Total before insertion: '+ n);
end ;
-------------------------------------------------------------------------

create or replace trigger update_Cust_trig
after update
on customer
begin
  dbms_output.put_line('A Customer data is updated in Database.');
end ;

-------------------------------------------------------------------------

create or replace trigger new_book_trig_bf
before insert
on books
declare
m number;
begin
  dbms_output.put_line('New Book(s) Adding in Database.');
```

```
  select count(*) into m  from books;
  dbms_output.put_line('Total before insertion: '+ m);
end ;
-----------------------------------------------------------------------

create or replace trigger new_book_trig_af
after insert
on books
declare
m number;
begin
  dbms_output.put_line('New Book(s) Added in Database.');
  select count(*) into m  from books;
  dbms_output.put_line('Total after insertion: '+ m);
end ;

-----------------------------------------------------------------------

create or replace trigger update_book_trig
after update
on customer
begin
  dbms_output.put_line('A Book info is updated in Database.');
end ;
-----------------------------------------------------------------------

create or replace trigger new_purchase_trig
after insert
on purchase
begin
  dbms_output.put_line('New Purchase occured.');
end ;

-----------------------------------------------------------------------

create or replace trigger new_rent_trig
after insert
on purchase
begin
  dbms_output.put_line('New Rent occured.');
end ;

-----------------------------------------------------------------------
```

/*Functions:*/

1. Write a function to get average rating for any book, given book_id.

```
create or replace FUNCTION totalCustomer
RETURN number IS
   total number(2) := 0;
BEGIN
   SELECT count(*) into total
   FROM customer;
   RETURN total;
END;
```

2. Write a function to find Total number of customers

```
create or replace FUNCTION totalCustomer
RETURN number IS
   total number:= 0;
BEGIN
   SELECT count(distinct customer_id ) into total
   FROM customer;
   RETURN total;
END;
```

//Basic Queries//

Application scopes:

1. Find Total number of customers in the database using the Function totalCustomer to make easier statistical view.

SQL query:

```
select distinct totalCustomer from customer;
```

Output:

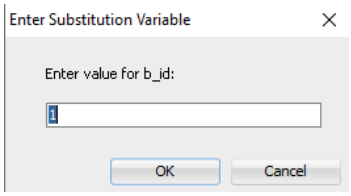
	TOTALCUSTOMER
1	20

2. Find average rating of a specific book where book id is given. Use avgRating(b_id) function to make smaller query. Take Id from user input

SQL query:

```
select DISTINCT b_id,book_title,avgRating(b_id)
from reviews join books on(b_id=book_id)
where b_id = '&b_id';
```

Output:



B_ID	BOOK_TITLE	AVGRATING(B_ID)
1	Data Communications and Networking	9.5

3. Show all purchase list/bills of ALL customers with purchase date, who have bought something.

SQL query:

```
select distinct purchase.c_id, order_count*unit_price,
purchase.PURCHASE_DATE
from purchase join books on(b_id= book_id)
ORDER BY c_id;
```

Output:

	C_ID	ORDER_COUNT*UNIT_PRICE	PURCHASE_DATE
1	1	1120	25-JUL-18
2	1	1500	17-JUL-18
3	3	750	17-JUL-18
4	6	250	09-OCT-18
5	8	1300	27-OCT-18
6	9	250	19-JUL-18
7	11	460	21-OCT-18
8	11	1100	27-SEP-18
9	14	1500	23-OCT-18
10	19	130	19-OCT-18
11	19	600	27-OCT-18
12	20	500	01-NOV-18

4. Show Total revenue of any specific customer as it is needed to update their discount level. Take user input for that.

SQL query:

```
select distinct customer.customer_name, purchase.c_id, sum(order_count*unit_price)
from purchase join books on(b_id= book_id) join customer on(c_id = customer_id)
where customer_name = '&customer_name'
group by customer.customer_name,purchase.c_id;
```

Output:

	CUSTOMER_NAME	C_ID	SUM(ORDER_COUNT*UNIT_PRICE)
1	Sakib Hasan	1	2620

5. Show Total revenue of every customer as it is needed to update their discount level or have an overview on regular/most important customer. Sort them according to their revenue- high to low.

SQL query:

```
With temp as (select c_id,sum(order_count*unit_price) as total
from purchase join books on(b_id= book_id)
group by c_id)
select distinct customer.customer_id, customer.CUSTOMER_NAME, total
from temp join customer on(c_id = customer_id) order by total desc;
```

Output:

	CUSTOMER_ID	CUSTOMER_NAME	TOTAL
1	1	Sakib Hasan	2620
2	11	Fairuz Nawer Meem	1560
3	14	Syeed Abrar Zaoad	1500
4	8	Fatema Tuz Zohra	1300
5	3	Sadia Afrin Mim	750
6	19	Marzan Binte Abid	730
7	20	Esha Ferdous	500
8	9	Asif Zaman	250
9	6	Mahsin Bin Akram	250

6. Find the customer who has the most revenue , who has spent most to purchase books.

SQL query:

```
With temp as (select c_id,sum(order_count*unit_price) as total
from purchase join books on(b_id= book_id)
group by c_id)
select distinct customer.customer_id, customer.CUSTOMER_NAME, total
from temp join customer on(c_id = customer_id)
where total = (select max(total) from temp);
```

Output:

	CUSTOMER_ID	CUSTOMER_NAME	TOTAL
1	1	Sakib Hasan	2620

7. Show all the reviews along with author,book title and review date of a specific book by all customers who have given a feedback. take user input.

SQL query:

```
select c_id,book_title,summery,review_date
from reviews join books on(b_id = book_id)
where b_id = '&b_id';
```

Output:

	C_ID	BOOK_TITLE	SUMMERY	REVIEW_DATE
1	3	Data Communications and Networking	Well Organized.	19-JUL-18
2	9	Data Communications and Networking	Easy to Understand.	19-JUL-18

8. Show all the Rent orders from all customers along with customer name and sort them by date of apply.

SQL query:

```
select customer_name,c_id, days, rent_date
from rent join customer on(c_id = customer_id)
order by rent_date;
```

Output:

	CUSTOMER_NAME	C_ID	DAYS	RENT_DATE
1	Asif Zaman	9	7	13-JUL-18
2	Sakib Hasan	1	14	17-JUL-18
3	Sadia Afrin Mim	3	7	23-JUL-18
4	Sakib Hasan	1	5	15-AUG-18
5	Fairuz Nawer Meem	11	5	17-SEP-18
6	Mahsin Bin Akram	6	14	09-OCT-18
7	Fairuz Nawer Meem	11	7	19-OCT-18
8	Marzan Binte Abid	19	14	19-OCT-18
9	Syeed Abrar Zaoad	14	7	23-OCT-18
10	Fatema Tuz Zohra	8	14	23-OCT-18
11	Marzan Binte Abid	19	3	25-OCT-18
12	Esha Ferdous	20	14	01-NOV-18
13	Esha Ferdous	20	7	21-NOV-18

9. Show the wishlist for specific customer alongwith the info of the book.And customer should be able to put his/her customer id via hand input.

SQL query:

```
select book_title, b_id,c_id
from wishlist join books on(b_id=book_id)
where c_id = '&c_id';
```

Output:

	BOOK_TITLE	B_ID	C_ID
1	The Adventures of Sherlock Holmes	7	2

10. Show start and end date of rental issue for every customer who have issued a book according to the returning date of the book.

SQL query:

```
select c_id,b_id, rent_date, (rent_date + days ) as returning_date
from rent
order by returning_date;
```

Output:

	C_ID	B_ID	RENT_DATE	RETURNING_DATE
1	9	3	13-JUL-18	20-JUL-18
2	3	1	23-JUL-18	30-JUL-18
3	1	1	17-JUL-18	31-JUL-18
4	1	1	15-AUG-18	20-AUG-18
5	11	2	17-SEP-18	22-SEP-18
6	6	1	09-OCT-18	23-OCT-18
7	11	3	19-OCT-18	26-OCT-18
8	19	1	25-OCT-18	28-OCT-18
9	14	2	23-OCT-18	30-OCT-18
10	19	1	19-OCT-18	02-NOV-18
11	8	3	23-OCT-18	06-NOV-18
12	20	1	01-NOV-18	15-NOV-18
13	20	3	21-NOV-18	28-NOV-18

11. Show the customers who has due date to return the book (those who have to return to the day instant). This query will find out which customers have to or haven't return their books in the designated date. Insert new sample data → " (INSERT INTO rent VALUES ('000020','0000003', 7, TO_DATE('21/11/2018', 'DD/MM/YYYY')); "to get desired output.

SQL query:

```
select c_id,b_id,(rent_date + days ) as returning_date, sysdate as
current_date
from rent
where rent_date >= sysdate
order by returning_date;
```

Output :

C_ID	B_ID	RETURNING_DATE	CURRENT_DATE
20	3	28-NOV-18	15-NOV-18

12. Create a view of customers total revenue for later use. Keep their id and revenue along for better overview.

SQL Query:

```
/* to create a view for customer's total revenue*/
create view table_revenue AS
select c_id,sum(order_count*unit_price) as total
from purchase join books on(b_id= book_id)
group by c_id;
select * from TABLE_REVENUE;
```

Output: Same as problem number 5 (without customer name).

Conclusions and Discussions

This project was a great experience as I had to complete all the features by myself. I was able to learn lot of new things about oracle SQL, how to write queries, functions and other PL/SQL features. It helped me to get the practical overview of what I have learned in the theory classes. I got deeper understanding or normalizations, applied query, joins and keys by doing everything practically by myself. I got to know inside and out of many theories as I could manipulate or play with my data an database. This might help be in future life as well to design better database and

Limitations:

- Discount/membership level can't be automatic updated via this database design. No trigger/function written for this feature.
- User can't search specific book by writing a fraction of the book title. No SQL query to implement this feature to find book by part of its name.
- There is not enough data to get better idea on the performance of the database.
- No function added to automatically insert discount into purchase table.

Future Scope:

This database can be re-designed more efficiently and more functions can be further added. This can be used a proper database any real life book buy-sell and rental system.

References:

- https://www.quora.com/Why-would-companies-use-Oracle-database-when-there-is-free-MySQL
- https://www.quora.com/When-I-can-use-MySQL-for-free-why-should-I-go-for-other-databases-like-Oracle-or-SQL-Server
- http://www.vertabelo.com/blog/technical-articles/the-most-useful-date-and-time-functions-in-oracle-database
- https://stackoverflow.com/questions/27451226/add-days-oracle-sql