One University. One World. Yours.

Masters in Computing and Data Analytics

MANAGING AND PROGRAMMING DATABASE

MCDA5540

PROJECT – 1

REPORT

SUBMITTED BY:

SONAM VADSARIA(A00431604)
DIVEN KUMAR SAMBHWANI(A00425905)
JASLEEN KOUR(A00425843)
RAVNEET SINGH OBEROI (A00426623)

SUBMITTED TO: TRISHLA SHAH

1. Introduction

1.1 Purpose and Scope

Halifax Science Library (HSL) maintains its own SQL database that contains various magazine publications. However, HSL has three immediate plans to implement . These three plans are as following:

1. To record sales transactions history.
2. To record data about all articles for each magazine.
3. To record monthly expenses data for HSL.

HSL will improve its existing database design and the efficiency of its system by implementing these plans.

1.2 Project Executive Summary

To implement these plans for HSL, entity and their relations need to be understood. Relational database schema needs to be designed based on this. After designing the schema, tables should be created using SQL scripts. Two bash scripts to be executed to create mongo collection from existing JSON file and use CSV format of it and import it in MySQL. After implementing everything successfully, we have developed PHP web Application which includes all the five operations.

1.3 Tools Used

Various tools are used in this project to develop the web application for HSL and create the database. Following are the tools :

EER Diagram	: Online Diagram Software, draw.io
Relational Database Schema	: Online Database Designer tool, sqldbm.com
Query Optimization	: MySQL Workbench 8.0
Data2mongo	**: Sublime Text,JavaScript, Terminal**
Mongo2sql	**: Sublime Text, Shell Script, Terminal**
PHP Web Application	**: Sublime Text**
Comparing Execution Time	: MySQL Workbench 8.0 and IBM Db2 Warehouse

2. Database Design :

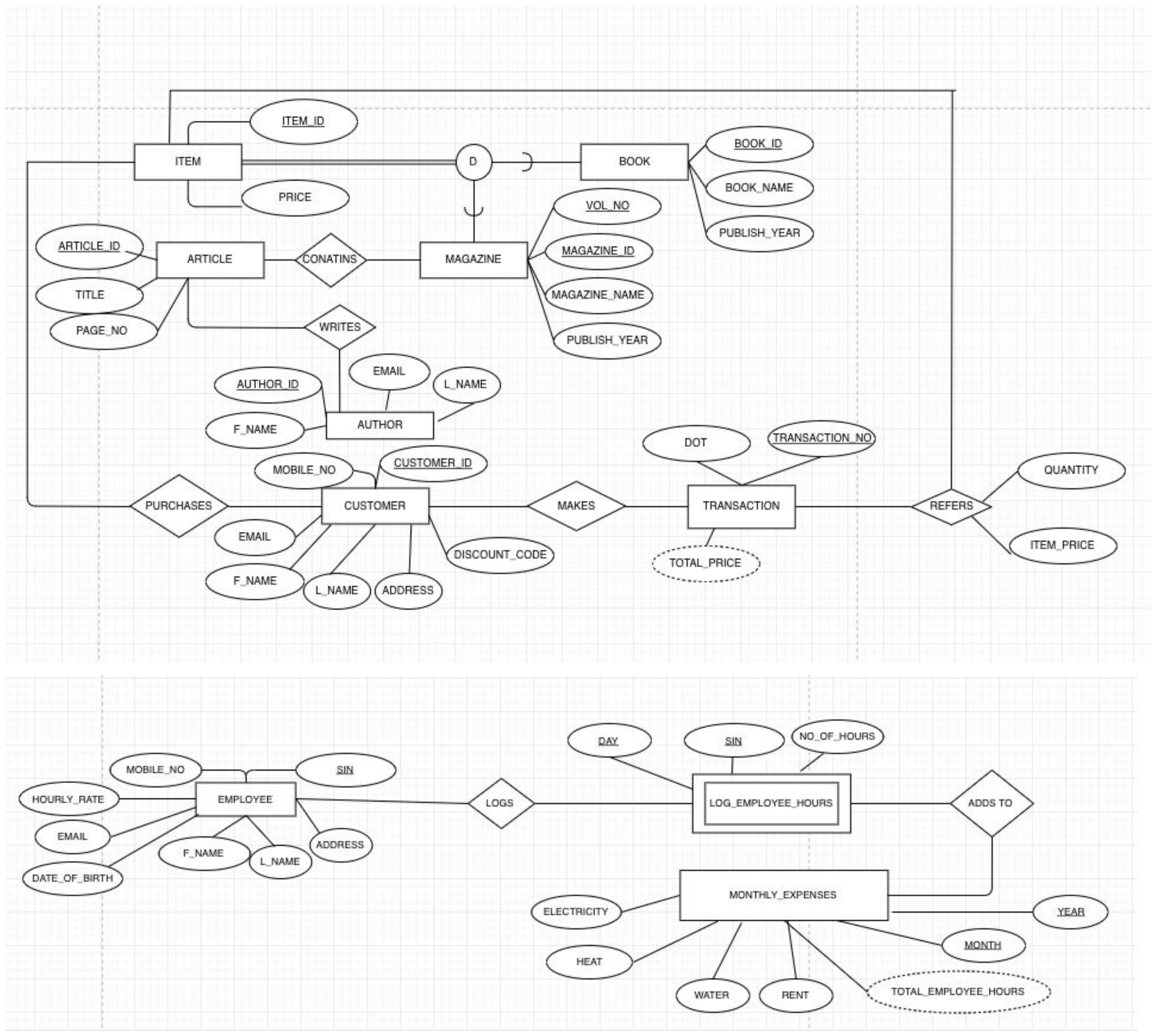


Fig 2.1 **ERR representation of the project is given below.**

1. **Normalization :** Normalization technique organizes tables in a manner that minimizes or even eliminates redundancy and dependency of data.

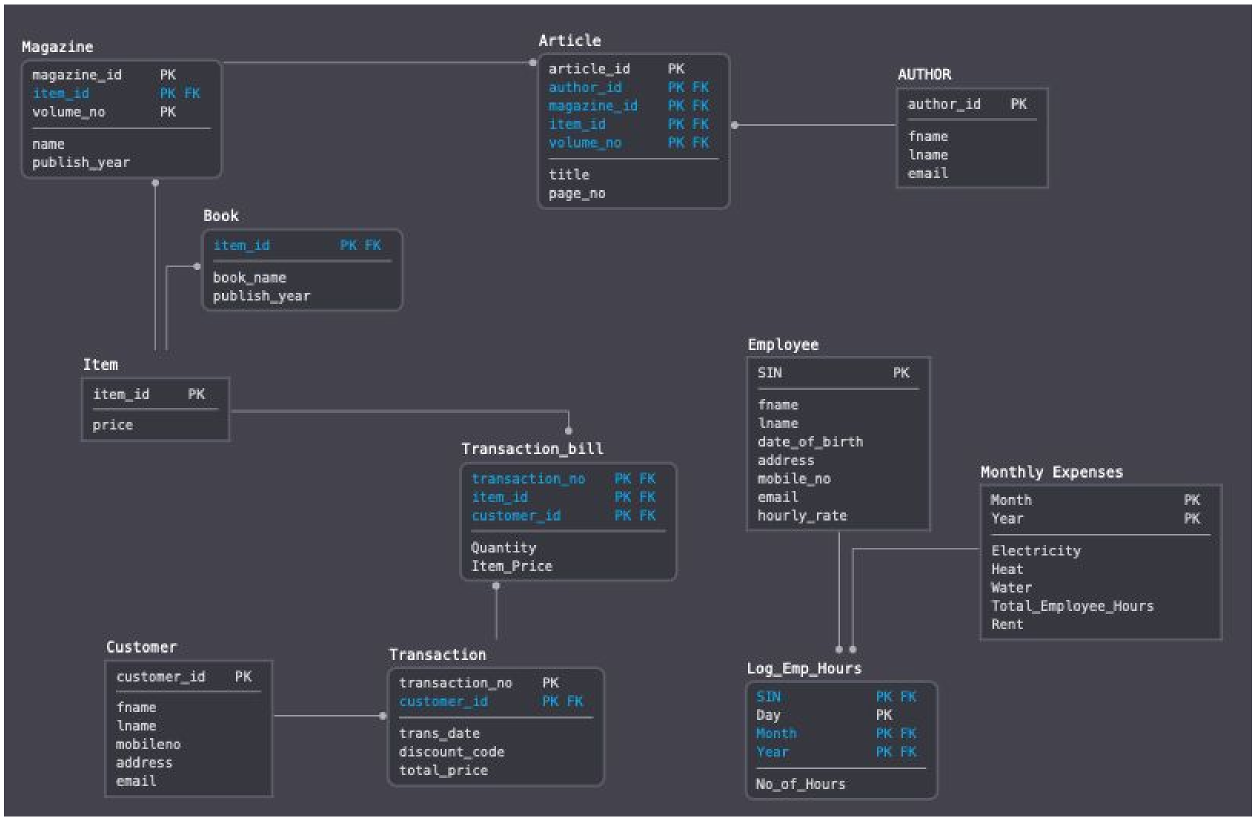
2.1 Normalization Rules:

1NF -First Normal Form:

•**Single Valued Attribute**. Each column of your table should be single valued.
•**Attribute domain should not change**. In each column the values stored must be of the same type.

• **Unique name for Attributes/Columns**. This rule expects that each column in a table should have a unique name.

• **Order doesn't matter.**



2NF -Second Normal Form :

For a table to be in the Second Normal Form, it must satisfy two conditions:

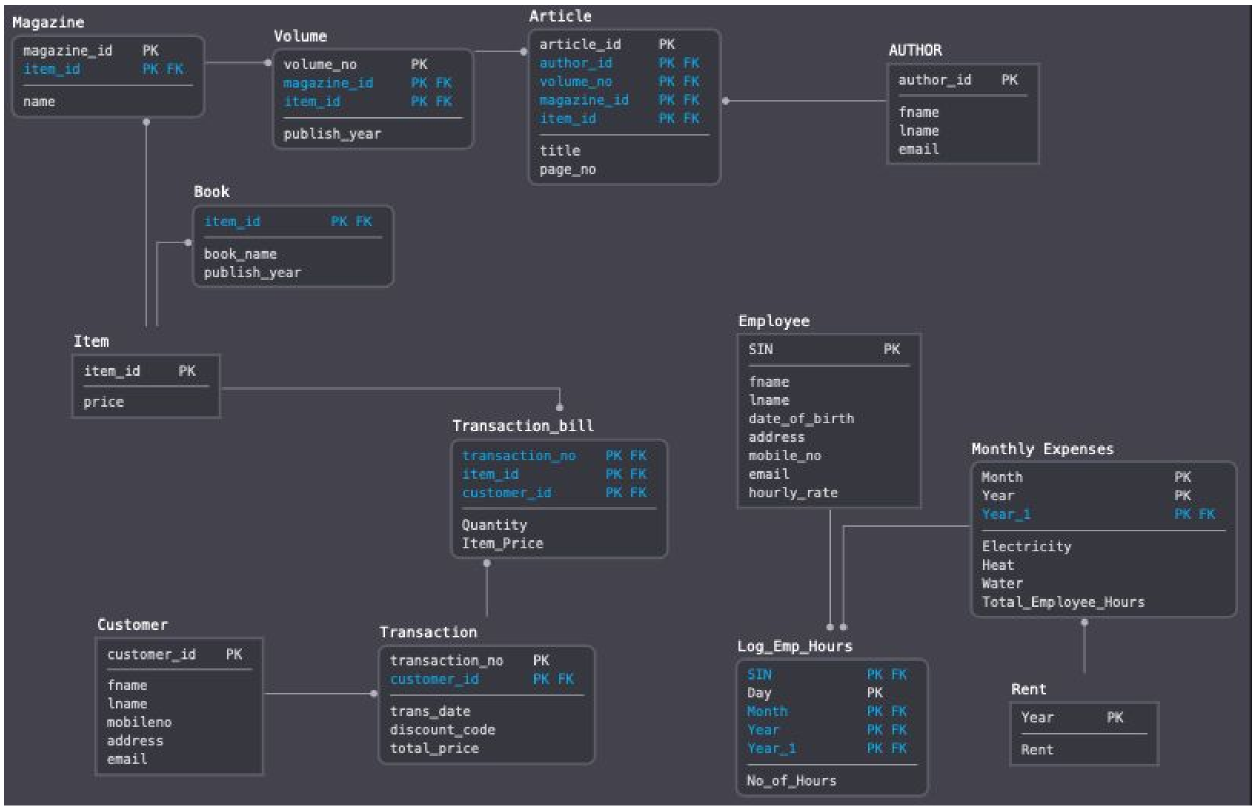
1) First Normal Form required.

2) No Partial Dependency should be there.

3NF -Third Normal Form :

•Remove columns with no dependency on the key.

•Values in a record that are not part of that record's key do not belong in the table.



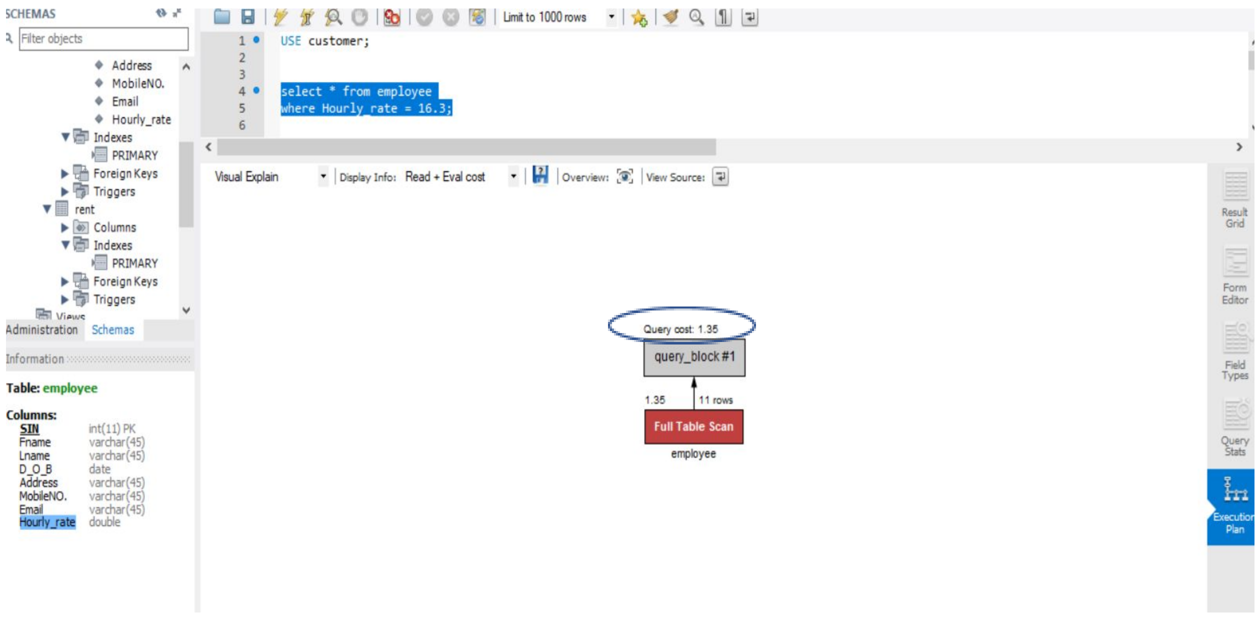
- **Item**: No Partial or transitive dependency as price is unique for each id
- **Magazine**: All the fields are dependent on id and volume no and there's no partial dependency nor transitive dependency as it depends on id + volume no both and not the other way
- **Book**: No Partial or transitive dependency as all the rows are unique for each id
- **Article** : All the fields are dependent on article_id ,volume no and Magazine id. There's no partial dependency nor transitive dependency as each row depends on article id + volume no + magazine_id and not the other way
- **Author**: No Partial or transitive dependency as each row is unique for each author_ id
- **Customer**: No Partial or transitive dependency as each row is unique for each customer_id
- **Transaction:** All fields are dependent on Transaction_no, and there's no partial or transitive dependency
- **Transaction_bill:** All fields are dependent on Transaction_no+Item_id, . There's no partial dependency nor transitive dependency as each row depends on Transaction_no+Item_id and not the other way
- **Employee**: No Partial or transitive dependency as each row is unique for each SIN

- **Monthly Expenses:** All fields are dependent on Month+Year . There's no partial dependency nor transitive dependency as each row depends on Month+Year and not the other way
- **Log_Emp_Hours**: No Partial or transitive dependency as No_of_Hours is unique for SIN+DAY+MONTH+YEAR

2.2 Query Optimization :

Query optimization reduces the system resources needed to fulfill a **query**, and ultimately provide the user with the correct result set faster.

Here we have taken table 'employee' to show the query execution plan.



```
USE customer;


select * from employee
where Hourly_rate = 16.3;
```

Table: employee

Columns:

SIN	int(11) PK
Fname	varchar(45)
Lname	varchar(45)
D_O_B	date
Address	varchar(45)
MobileNO.	varchar(45)
Email	varchar(45)
Hourly_rate	double

Fig 2.2.1 : Execution Plan without Indexing.

According to the data in the table 'employee' there is only one employee with Hourly_rate = 16.3 which is unique therefore result comes out to be only one row. But we can see in the execution plan that shows that Full Table scan has been done. Indexing the 'employee' table on 'Hourly_rate'. After indexing we can clearly see in the figure below that there is reduction in Query cost.

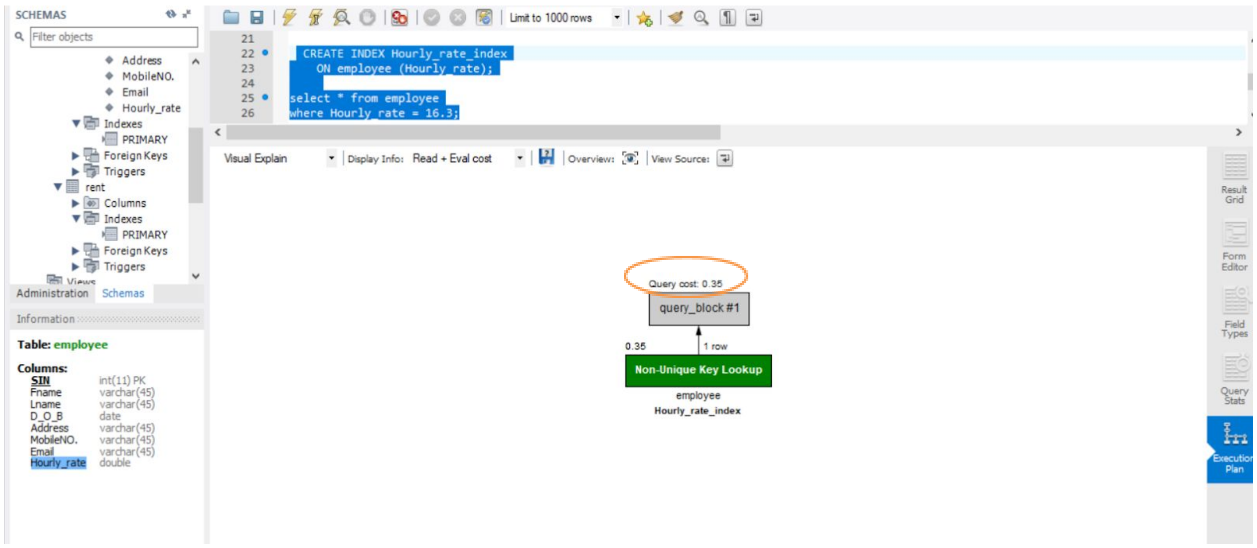
Query :

CREATE INDEX Hourly_rate_index

ON employee (Hourly_rate);

select * from employee

where Hourly_rate = 16.3;



```
 CREATE INDEX Hourly_rate_index
   ON employee (Hourly_rate);

select * from employee
where Hourly_rate = 16.3;
```

Table: employee

Columns:

```
SIN          int(11) PK
Fname        varchar(45)
Lname        varchar(45)
D_O_B        date
Address      varchar(45)
MobileNO.    varchar(45)
Email        varchar(45)
Hourly_rate  double
```

Fig 2.2.2 : Query Optimization

3.Project Flow:

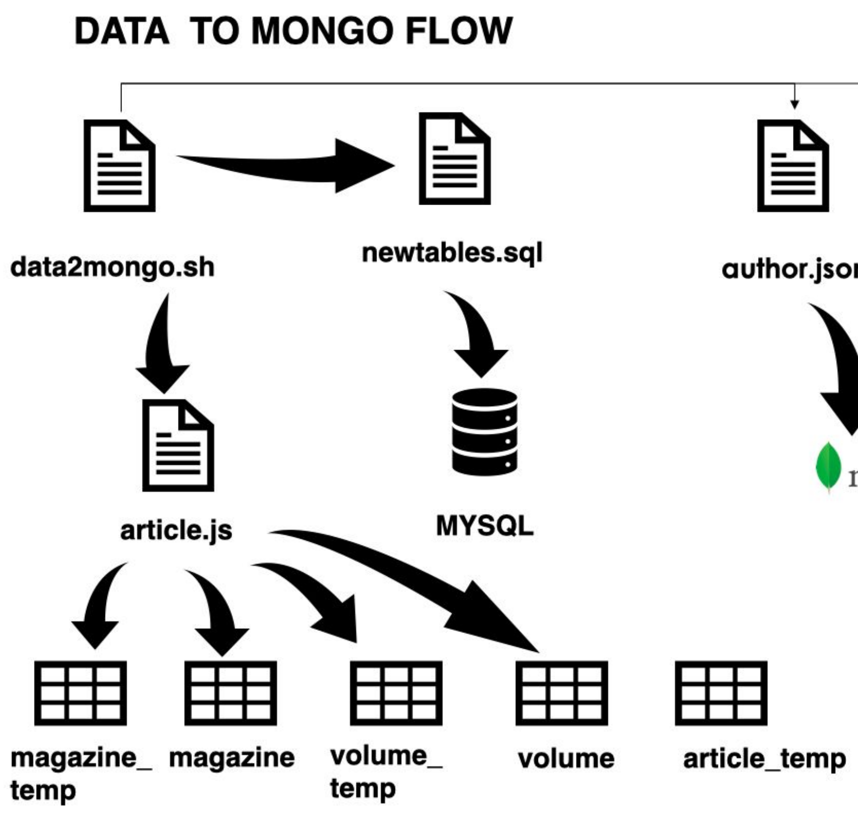


Fig 3.1 Data to Mongo Flow

1. User has to input his SQL credentials to login into the SQL server .
2. sql_file is read which creates the new tables in the sql server.
3. we then ask the user to input his mongodb credentials so that we are able to login into mongodb server and we also ask him to input the name of the database
4. We now ask the user to input the json files and get the author_file to make the Author collection .similarly we get the article_file to make the Article collection .
5. we now drop the following collections :
    a. author
    b. ARTICLE_temp
    c. articles_all

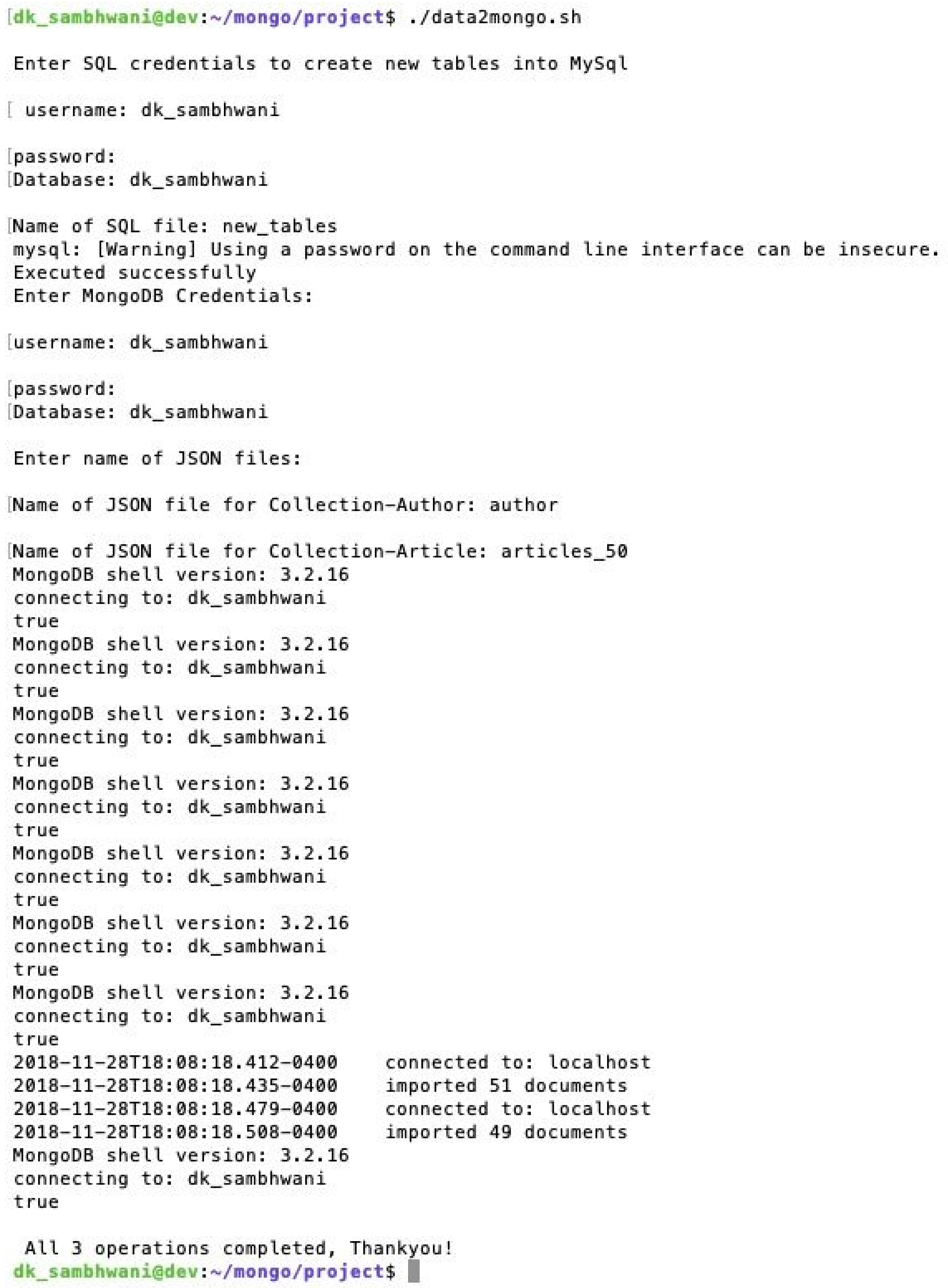
    d. magazine
    e. magazine_temp
    f. volume
    g. volume_temp

6. we now ask the user to input json files for author and article and we get the two files author_file.json and article_file.json from the user both the files are exported into mongodb collection

7. We now load the file article.js which takes only those fields from article which are needed .

Exploring Article .js

1. All the records are fetched from articles_all database .
2. Then we store magazine_id and name in magazine_temp collection . but in this the records are not unique and there can be repetitions also
3. Hence using the aggregate function we store the unique magazine_id and name in the the collection called magazine
4. Same thing was done for volume and author also .
5. Only the desired values were taken from Articles_all and stored in the ARTICLE_temp collection . the values that were taken are :
    a. article_id
    b. author_name
    c. volume_no
    d. magazine_name
    e. title
    f. page_no



```
[dk_sambhwani@dev:~/mongo/project$ ./data2mongo.sh

 Enter SQL credentials to create new tables into MySql

[ username: dk_sambhwani

[password:
[Database: dk_sambhwani

[Name of SQL file: new_tables
 mysql: [Warning] Using a password on the command line interface can be insecure.
 Executed successfully
 Enter MongoDB Credentials:

[username: dk_sambhwani

[password:
[Database: dk_sambhwani

 Enter name of JSON files:

[Name of JSON file for Collection-Author: author

[Name of JSON file for Collection-Article: articles_50
 MongoDB shell version: 3.2.16
 connecting to: dk_sambhwani
 true
 MongoDB shell version: 3.2.16
 connecting to: dk_sambhwani
 true
 MongoDB shell version: 3.2.16
 connecting to: dk_sambhwani
 true
 MongoDB shell version: 3.2.16
 connecting to: dk_sambhwani
 true
 MongoDB shell version: 3.2.16
 connecting to: dk_sambhwani
 true
 MongoDB shell version: 3.2.16
 connecting to: dk_sambhwani
 true
 MongoDB shell version: 3.2.16
 connecting to: dk_sambhwani
 true
 2018-11-28T18:08:18.412-0400    connected to: localhost
 2018-11-28T18:08:18.435-0400    imported 51 documents
 2018-11-28T18:08:18.479-0400    connected to: localhost
 2018-11-28T18:08:18.508-0400    imported 49 documents
 MongoDB shell version: 3.2.16
 connecting to: dk_sambhwani
 true

  All 3 operations completed, Thankyou!
 dk_sambhwani@dev:~/mongo/project$
```

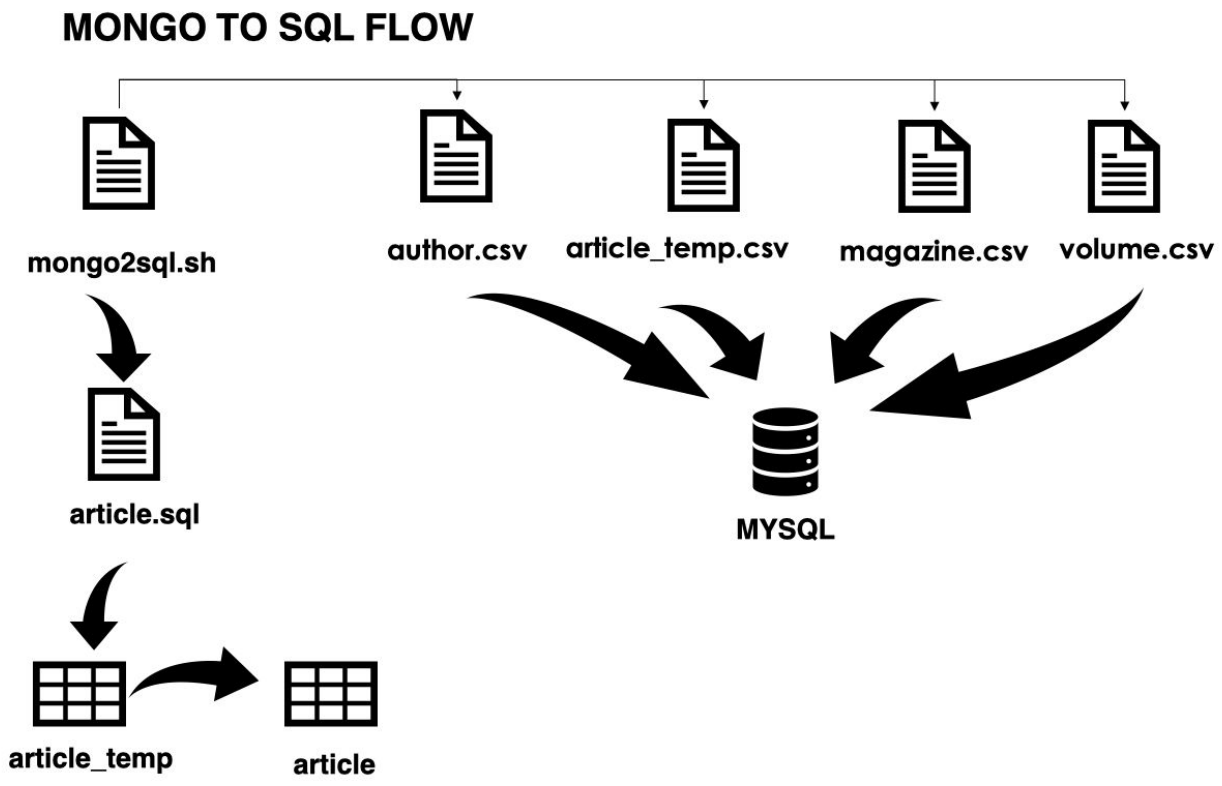
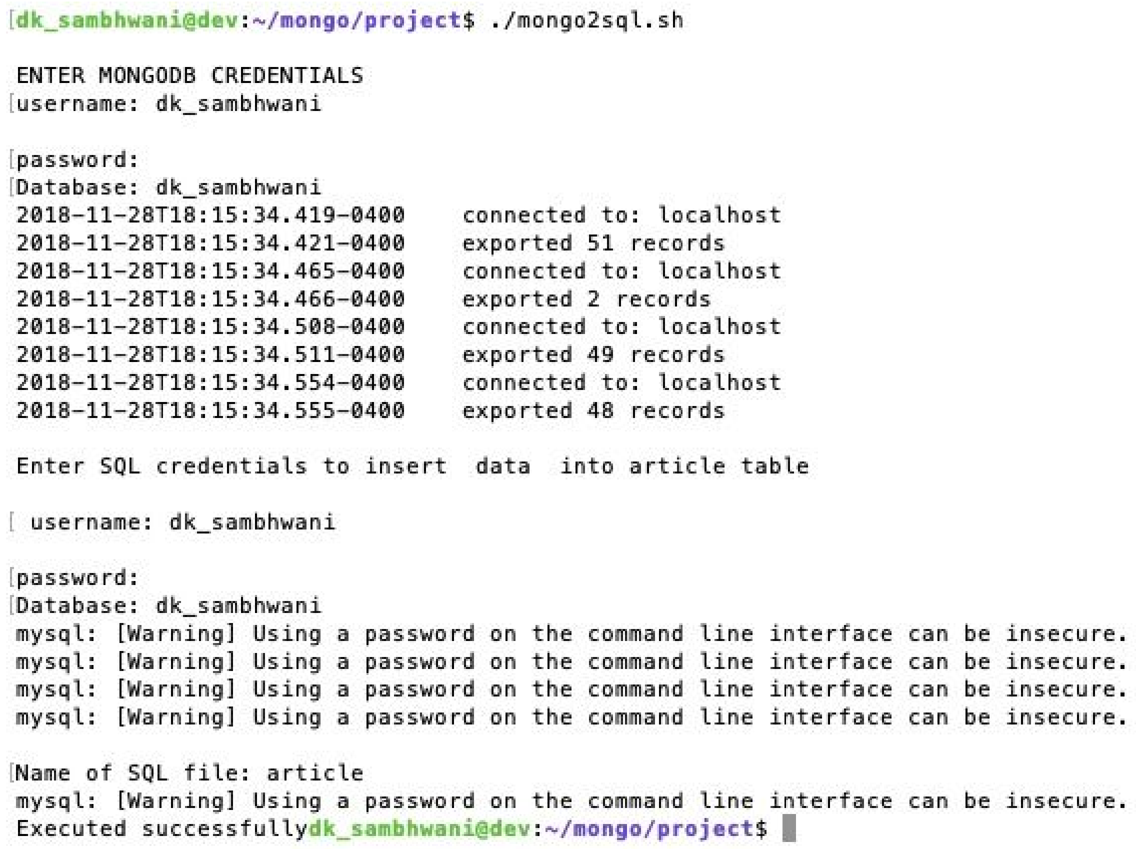


Fig 3.2 Mongo to SQL flow

Mongo to SQL

1. We start logging in with the Mongo DB credentials (username , password and name of DB)
2. We now export the following collections author , magazine, ARTICLE_temp and volume into the csv format
3. We ask the user to enter the sql credentials to enter the SQL serve . r
4. we now import the CSV files into the Sql server . the following files are imported :
    a. AUTHOR.csv
    b. magazine.csv
    c. ARTICLE_temp.csv
    d. volume.csv
5. we call the article.sql which does the following :

author name and magazine name are mapped through author table and magazine table to get the id and insert that in the article table.



```
[dk_sambhwani@dev:~/mongo/project$ ./mongo2sql.sh

 ENTER MONGODB CREDENTIALS
[username: dk_sambhwani

[password:
[Database: dk_sambhwani
 2018-11-28T18:15:34.419-0400    connected to: localhost
 2018-11-28T18:15:34.421-0400    exported 51 records
 2018-11-28T18:15:34.465-0400    connected to: localhost
 2018-11-28T18:15:34.466-0400    exported 2 records
 2018-11-28T18:15:34.508-0400    connected to: localhost
 2018-11-28T18:15:34.511-0400    exported 49 records
 2018-11-28T18:15:34.554-0400    connected to: localhost
 2018-11-28T18:15:34.555-0400    exported 48 records

 Enter SQL credentials to insert  data  into article table

[ username: dk_sambhwani

[password:
[Database: dk_sambhwani
 mysql: [Warning] Using a password on the command line interface can be insecure.
 mysql: [Warning] Using a password on the command line interface can be insecure.
 mysql: [Warning] Using a password on the command line interface can be insecure.
 mysql: [Warning] Using a password on the command line interface can be insecure.

[Name of SQL file: article
 mysql: [Warning] Using a password on the command line interface can be insecure.
 Executed successfullydk_sambhwani@dev:~/mongo/project$ 
```

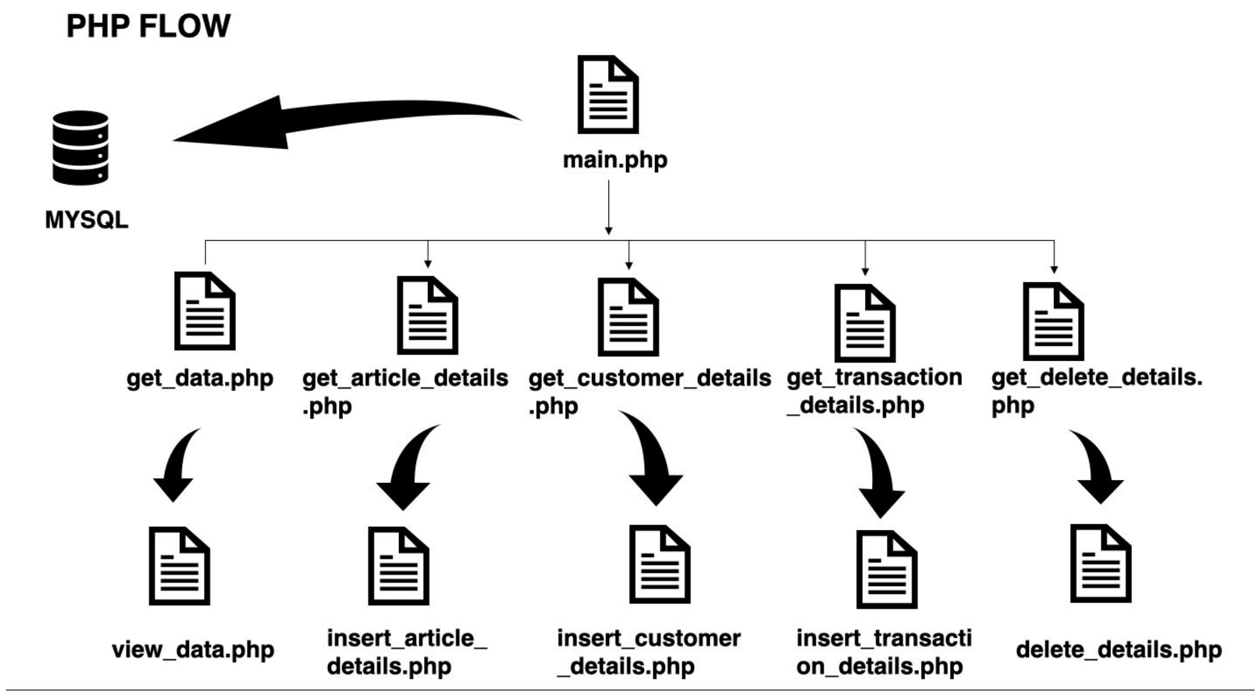


Fig 3.3 PHP Flow

4. PHP APPLICATION :

In the main page of Halifax Science Library we have five main operations to be performed and user can choose any one of them according to the task he has to perform.

1. Show table
2. Add Article
3. Add New Customer
4. Add New Transaction
5. Cancel Transaction.

MAIN PAGE :

URL : http://dev.cs.smu.ca/~j_kour/php/main.php

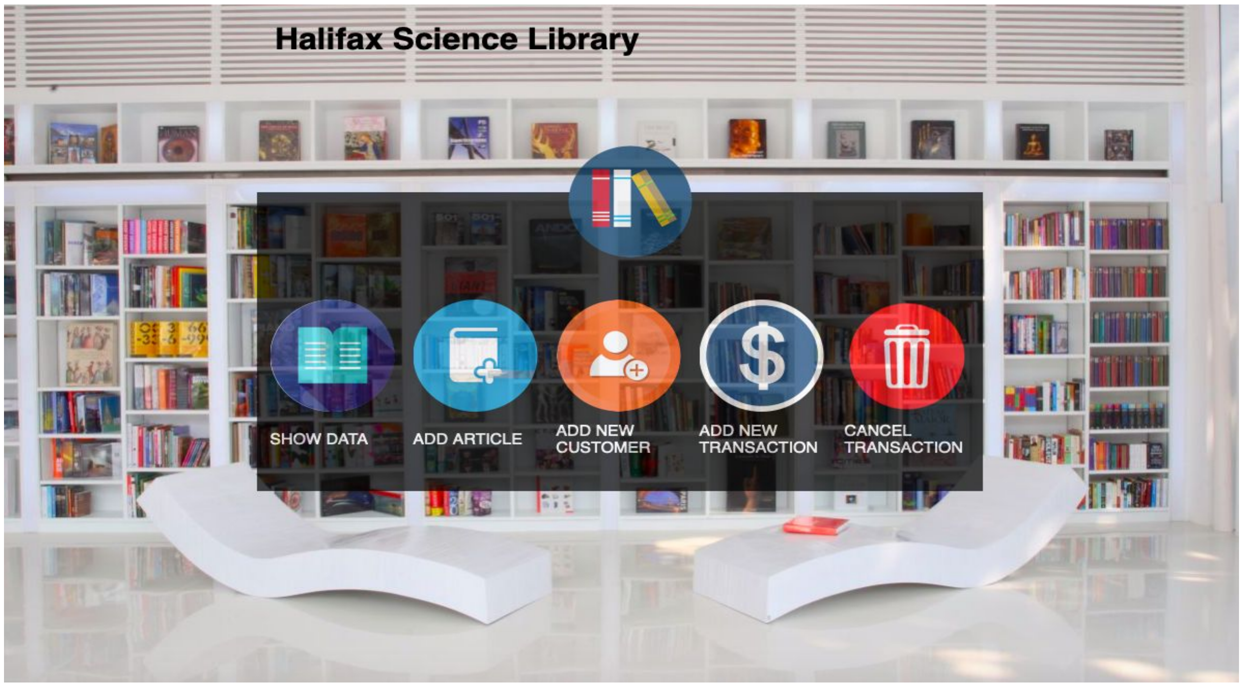
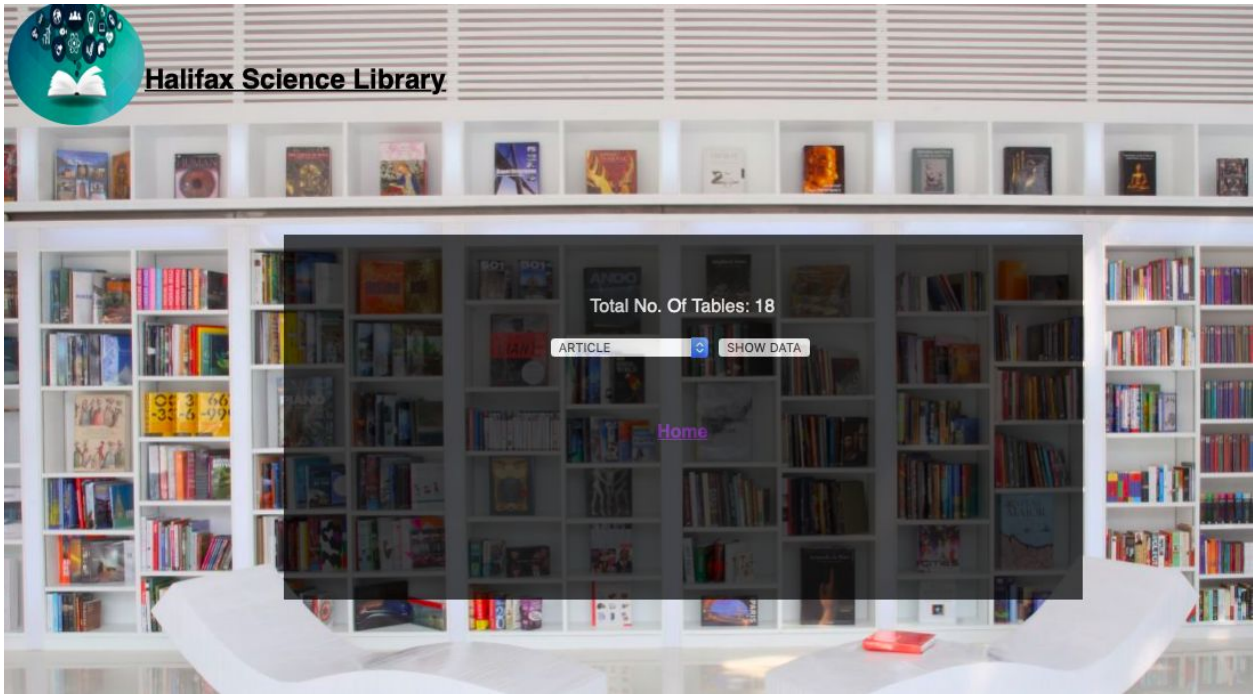


Fig 4.1 Halifax Science Library Main Screen

1. **<u>Show Table :</u>**

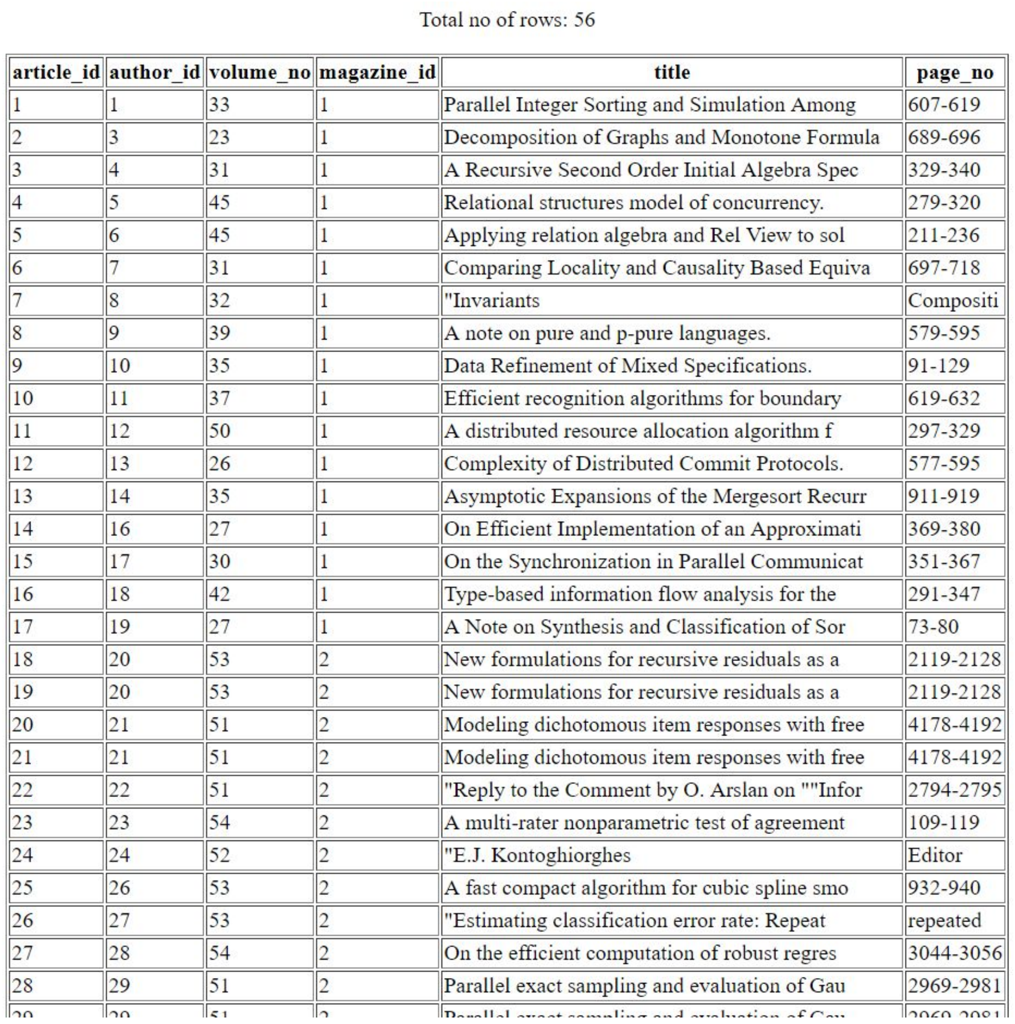
When we click on the show table option we land up to the below screen. Here we can see that there is drop down list with the names of all the tables(i.e. Total no. of tables : 18) in the database. The user can choose any table to view the data contained in that table with the help of *"SHOW DATA"* button. The user if wishes can move back to the main page with the help of the *"HOME"* option on the screen.



<u>Fig 4.2 Show Data Screen</u>

When user chooses table name and clicks on show data button he can easily view the content of the table as shown in the figure 3.3 below.

URL : http://dev.cs.smu.ca/~j_kour/php/view_data.php



Total no of rows: 56

article_id	author_id	volume_no	magazine_id	title	page_no
1	1	33	1	Parallel Integer Sorting and Simulation Among	607-619
2	3	23	1	Decomposition of Graphs and Monotone Formula	689-696
3	4	31	1	A Recursive Second Order Initial Algebra Spec	329-340
4	5	45	1	Relational structures model of concurrency.	279-320
5	6	45	1	Applying relation algebra and Rel View to sol	211-236
6	7	31	1	Comparing Locality and Causality Based Equiva	697-718
7	8	32	1	"Invariants	Compositi
8	9	39	1	A note on pure and p-pure languages.	579-595
9	10	35	1	Data Refinement of Mixed Specifications.	91-129
10	11	37	1	Efficient recognition algorithms for boundary	619-632
11	12	50	1	A distributed resource allocation algorithm f	297-329
12	13	26	1	Complexity of Distributed Commit Protocols.	577-595
13	14	35	1	Asymptotic Expansions of the Mergesort Recurr	911-919
14	16	27	1	On Efficient Implementation of an Approximati	369-380
15	17	30	1	On the Synchronization in Parallel Communicat	351-367
16	18	42	1	Type-based information flow analysis for the	291-347
17	19	27	1	A Note on Synthesis and Classification of Sor	73-80
18	20	53	2	New formulations for recursive residuals as a	2119-2128
19	20	53	2	New formulations for recursive residuals as a	2119-2128
20	21	51	2	Modeling dichotomous item responses with free	4178-4192
21	21	51	2	Modeling dichotomous item responses with free	4178-4192
22	22	51	2	"Reply to the Comment by O. Arslan on ""Infor	2794-2795
23	23	54	2	A multi-rater nonparametric test of agreement	109-119
24	24	52	2	"E.J. Kontoghiorghes	Editor
25	26	53	2	A fast compact algorithm for cubic spline smo	932-940
26	27	53	2	"Estimating classification error rate: Repeat	repeated
27	28	54	2	On the efficient computation of robust regres	3044-3056
28	29	51	2	Parallel exact sampling and evaluation of Gau	2969-2981
29	29	51	2	Parallel exact sampling and evaluation of Gau	2969-2981

Fig 4.3 View Data Screen

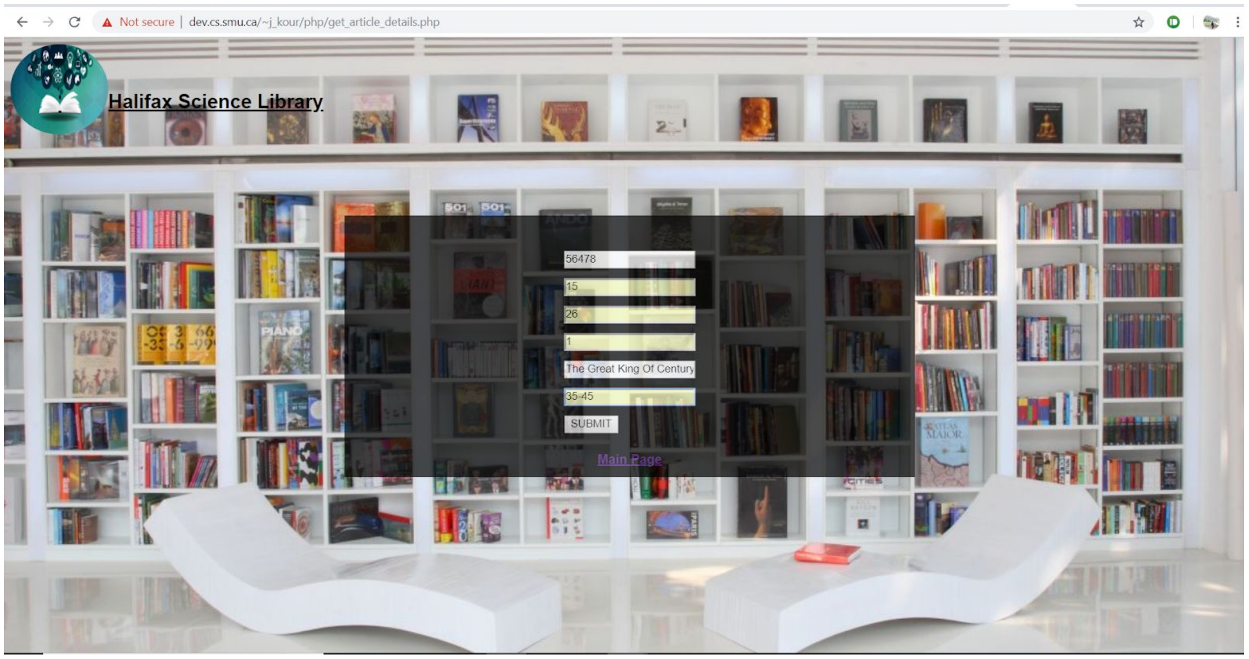
2. Add Article:

This option on main page of the web application helps user to add new article into the database. After clicking on "Add Article" icon on main page the user is taken to another page on which he/ she can add details(i.e. Article ID, Author ID, Volume No., Magazine ID, Title, Page Number) of article they want to add or insert.

Note : Validations have been applied to field in the way that if user tries to enter the volume no, magazine id or author id that are not present in the database he will get an alert *"No volume found for this magazine !" or "No Author found with this Author ID !" or "No magazine found with such ID !".*

Validation has also been applied on the combination of Magazine ID and Article ID to control the duplication and data redundancy.

URL : http://dev.cs.smu.ca/~j_kour/php/get_article_details.php


Not secure | dev.cs.smu.ca/~j_kour/php/get_article_details.php

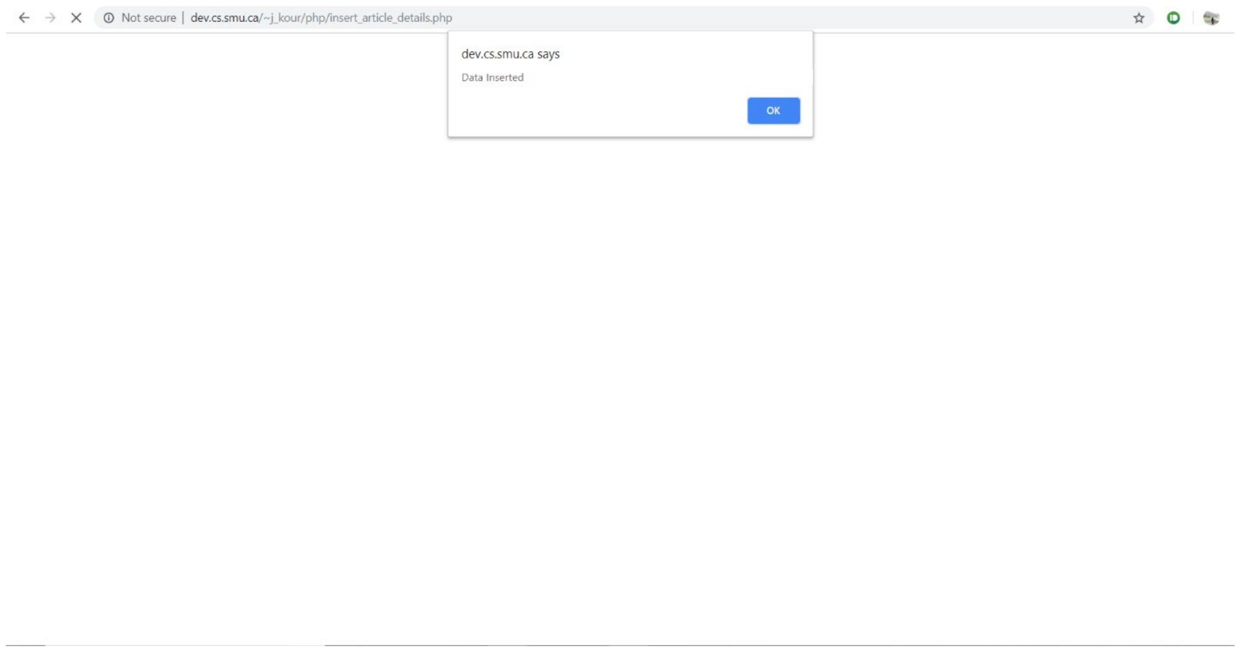
Halifax Science Library

56478
15
26
1
The Great King Of Century
35-45

Fig 4.4 Add Article Screen

When user correctly fills in details required the data gets inserted successfully and is taken to the following page.

URL : http://dev.cs.smu.ca/~j_kour/php/insert_article_details.php


Not secure | dev.cs.smu.ca/~j_kour/php/insert_article_details.php

dev.cs.smu.ca says

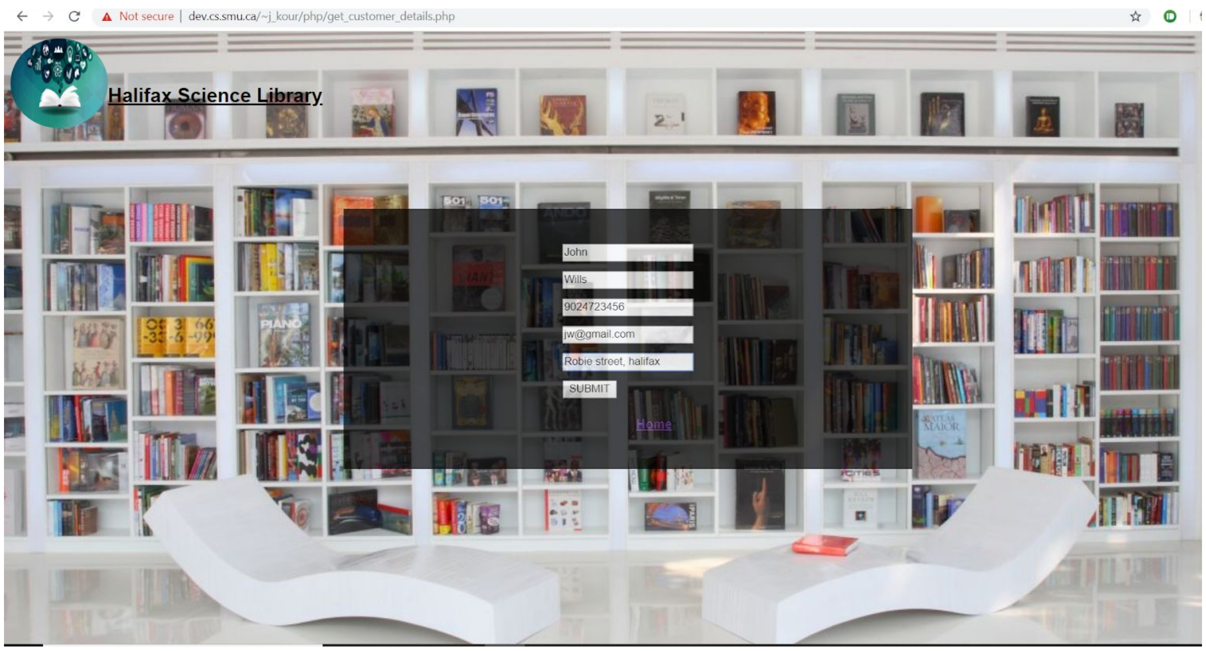
Data Inserted

Fig 4.4 Article Added

3. Add New Customer :

The third icon is of inserting the new customer. When user clicks on the icon he is taken to the below page. On this page he can add details of the new customer.

URL : http://dev.cs.smu.ca/~j_kour/php/get_customer_details.php


dev.cs.smu.ca/~j_kour/php/get_customer_details.php

Halifax Science Library

John
Wills
9024723456
jw@gmail.com
Robie street, halifax

Fig 4.5 Add New Customer

URL : http://dev.cs.smu.ca/~j_kour/php/insert_customer_details.php

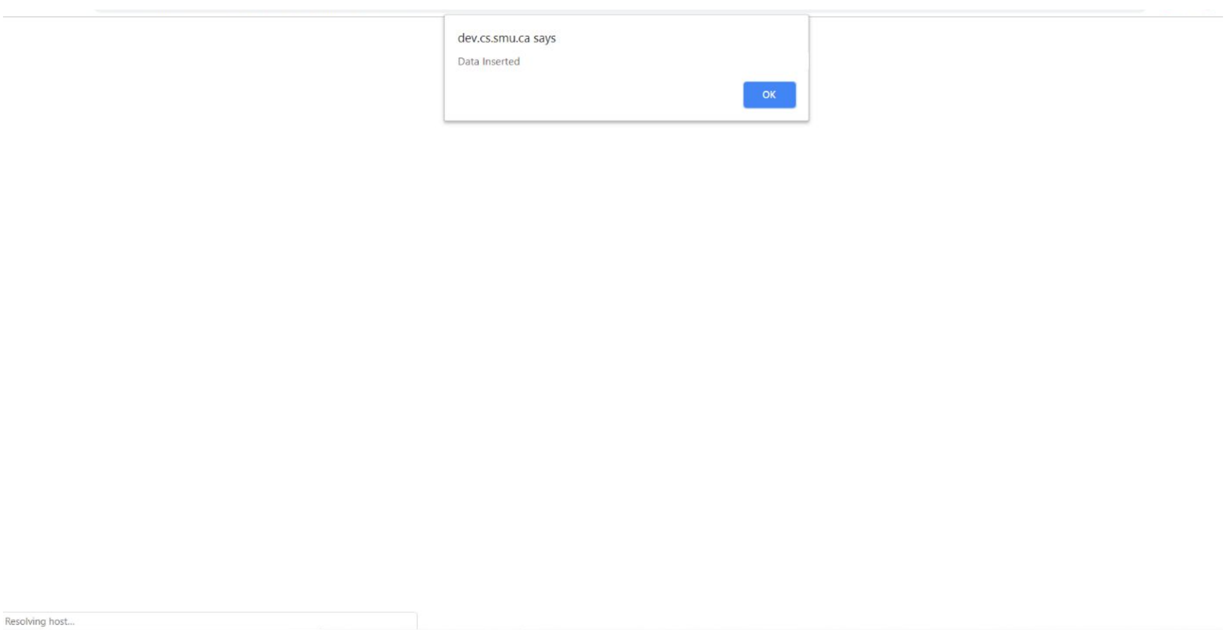
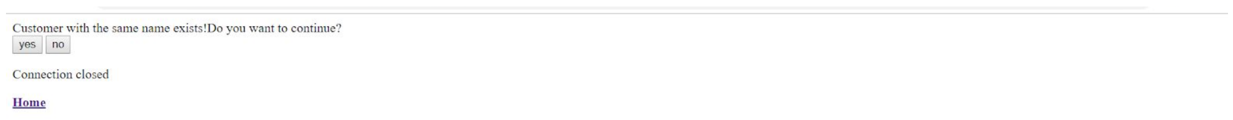

dev.cs.smu.ca says
Data Inserted

Fig 4.6 Customer Added

<u>Note : Validations</u> have been applied to field in the way that if user tries to make entry with same first and last name as done earlier it will prompt user whether he wishes to continue or not. If user clicks yes then data is successfully inserted else data is not inserted. It can been seen in figure below.

URL : http://dev.cs.smu.ca/~j_kour/php/insert_customer_details.php


Customer with the same name exists!Do you want to continue?

Connection closed

Home

Fig 4.7 Validation Check

4. Add New Transaction :

Clicking on add new transaction icon user reaches screen shown below in figure 3.8. Here user can add transaction detail(i.e. Customer ID, Transaction ID, Item ID and Quantity). After filling up details the data gets inserted successfully and by going on to transaction table user can easily view the transaction details and the discount code is applied in the transaction bill table. Below figures show the complete execution of add new transaction option.

Note : Validation has been added so that user can not add same Item ID again with different quantity. Customer ID chosen should be present in the database else it will prompt error message.

URL : http://dev.cs.smu.ca/~j_kour/php/get_transaction_details.php

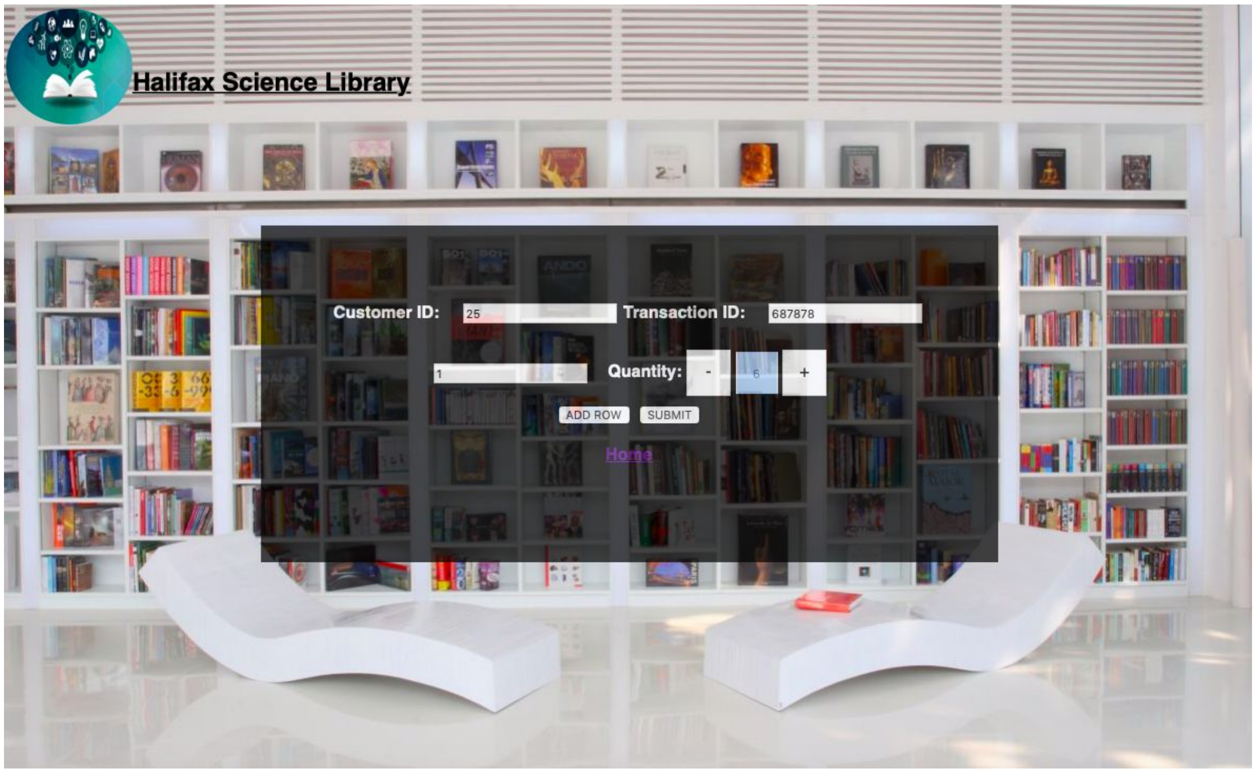
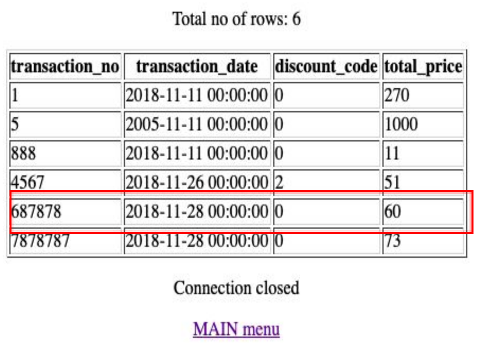


Fig 4.8 Add New Transaction

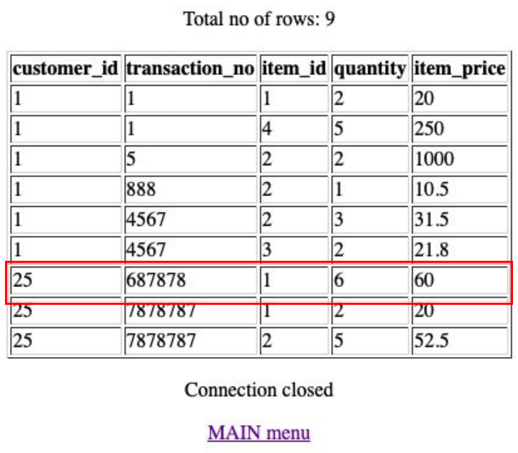

Total no of rows: 6

transaction_no	transaction_date	discount_code	total_price
1	2018-11-11 00:00:00	0	270
5	2005-11-11 00:00:00	0	1000
888	2018-11-11 00:00:00	0	11
4567	2018-11-26 00:00:00	2	51
687878	2018-11-28 00:00:00	0	60
7878787	2018-11-28 00:00:00	0	73

Connection closed

MAIN menu

Fig 4.9 Transaction table


Total no of rows: 9

customer_id	transaction_no	item_id	quantity	item_price
1	1	1	2	20
1	1	4	5	250
1	5	2	2	1000
1	888	2	1	10.5
1	4567	2	3	31.5
1	4567	3	2	21.8
25	687878	1	6	60
25	7878787	1	2	20
25	7878787	2	5	52.5

Connection closed

MAIN menu

Fig 4.10 Transaction bill table

5. Cancel Transaction:

The last option is to cancel transaction or delete transaction made earlier from the database. Below figure shows the fields required to delete transaction.

Note : Validation is applied first checks whether the ID submitted by user is there in DB or not and if the ID is present it checks the transaction date makes it impossible to delete the transaction that has been done 30 days before the current date.

URL : http://dev.cs.smu.ca/~j_kour/php/get_delete_details.php

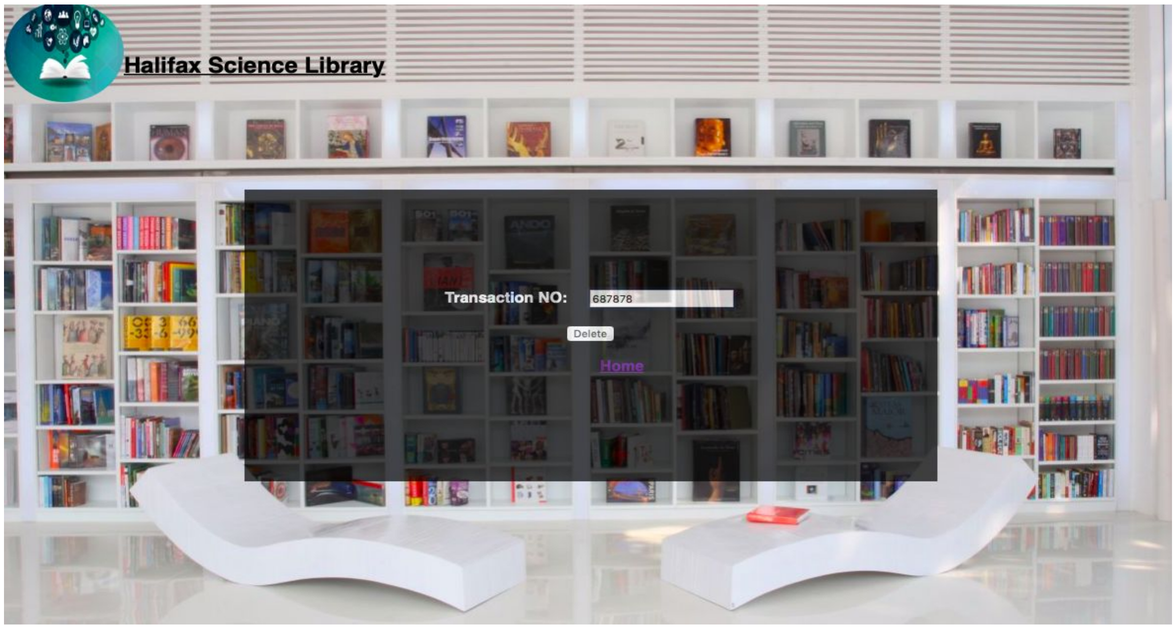


Fig 4.11 Cancel Transaction Screen

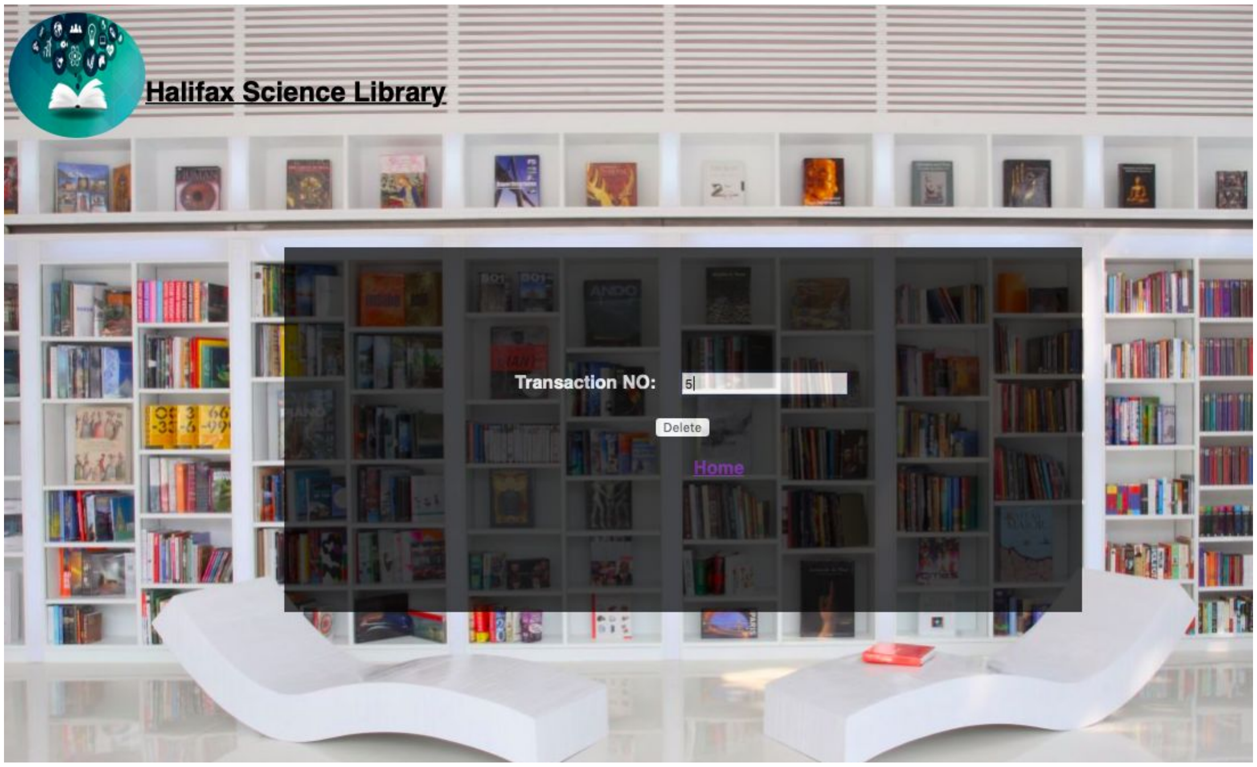


Fig 4.12 Cancel transaction

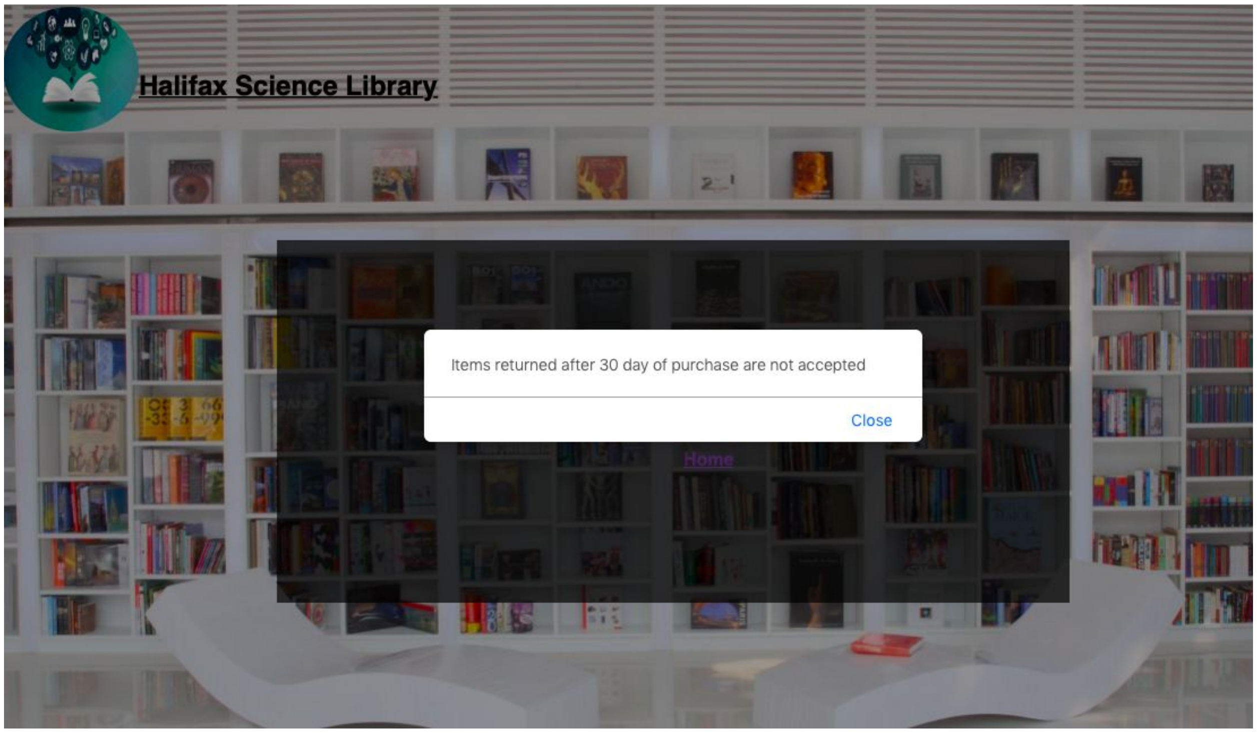

Items returned after 30 day of purchase are not accepted

Fig 4.13 Validation added to check purchase window of item

5.Execution in IBM DB2

We executed the following command in both IBM DB2 and MySql Workbench and compared the execution time in both the platforms .

Query Executed :

INSERT INTO ARTICLE (AUTHOR_ID,VOLUME_NO,MAGAZINE_ID,TITLE,PAGE_NO)

SELECT B.AUTHOR_ID,A.VOLUME_NO,C.MAGAZINE_ID,A.TITLE,A.PAGE_NO

FROM ARTICLE_temp A

JOIN AUTHOR B

ON A.AUTHOR_NAME=B.author_name

INNER JOIN MAGAZINE C

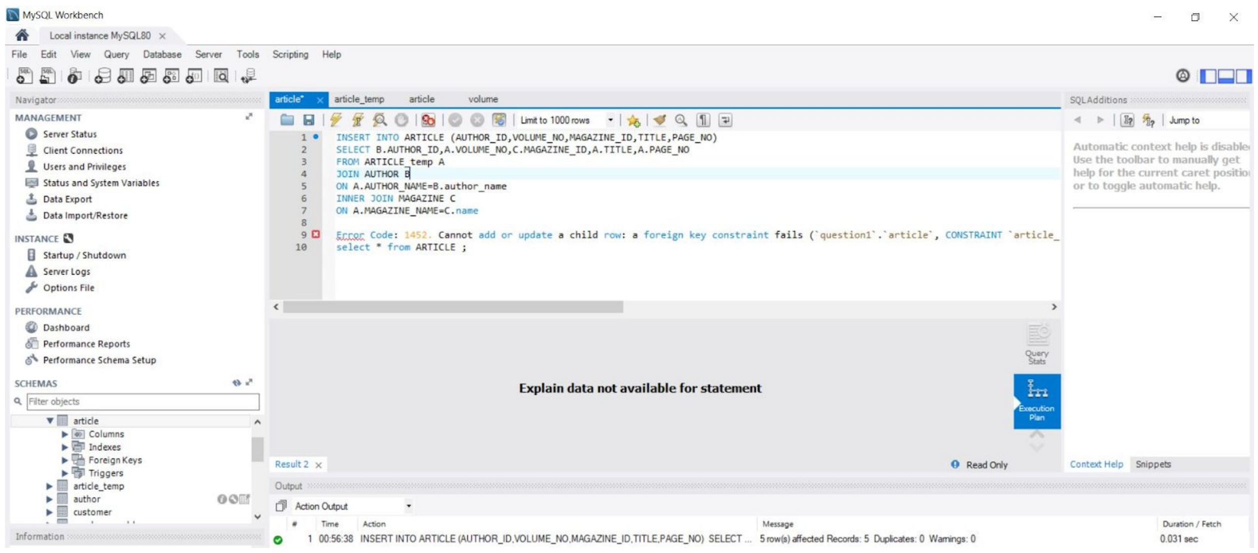
ON A.MAGAZINE_NAME=C.name

Execution time in MySql Workbench: 0.031 seconds

Execution time in IBM DB2: 0.139 seconds

Execution time for IBM DB2 and MySQL varies and is dependent on many other factors. In case of few queries, MySQL turns out to be faster than IBM DB2.

We executed the same query in both the databases IBM DB2 and MYSQL to compare the time. In our case, IBM DB2 took longer time than MYSQL.



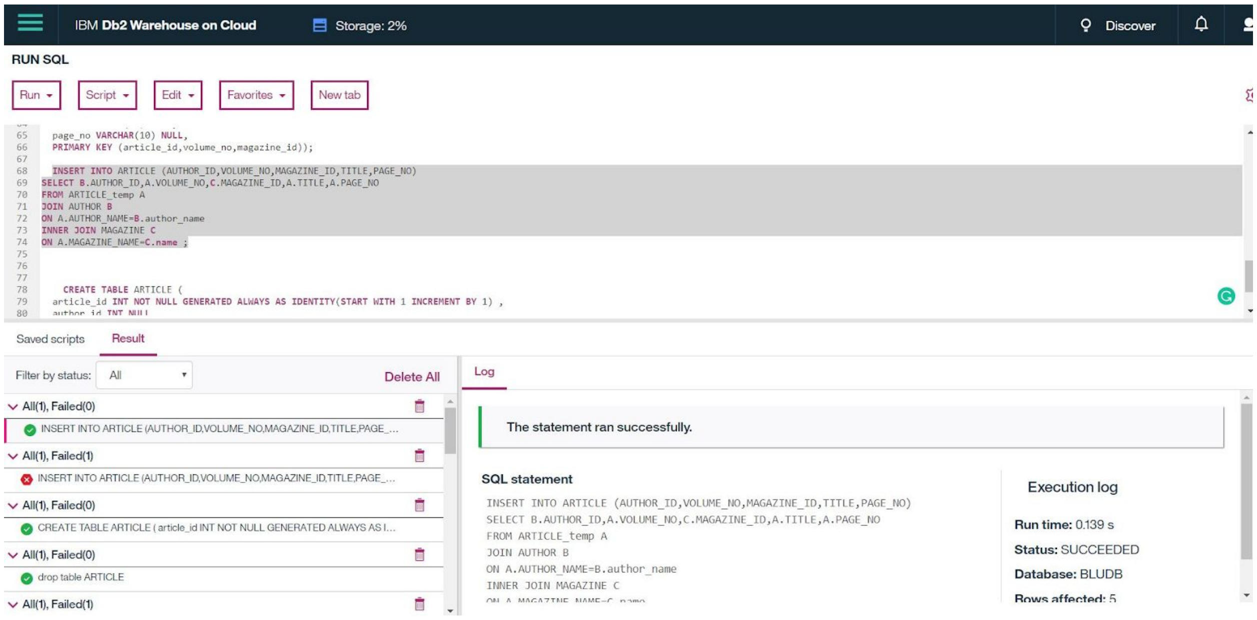
```
INSERT INTO ARTICLE (AUTHOR_ID,VOLUME_NO,MAGAZINE_ID,TITLE,PAGE_NO)
SELECT B.AUTHOR_ID,A.VOLUME_NO,C.MAGAZINE_ID,A.TITLE,A.PAGE_NO
FROM ARTICLE_temp A
JOIN AUTHOR B
ON A.AUTHOR_NAME=B.author_name
INNER JOIN MAGAZINE C
ON A.MAGAZINE_NAME=C.name

Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails (`question1`.`article`, CONSTRAINT `article_
select * from ARTICLE ;
```

Explain data not available for statement

```
#  Time      Action                                                                  Message                                                    Duration / Fetch
1  00:56:38  INSERT INTO ARTICLE (AUTHOR_ID,VOLUME_NO,MAGAZINE_ID,TITLE,PAGE_NO) SELECT ...  5 row(s) affected Records: 5  Duplicates: 0  Warnings: 0  0.031 sec
```

Fig 5.1 Execution in MySQL



```
  page_no VARCHAR(10) NULL,
  PRIMARY KEY (article_id,volume_no,magazine_id));

  INSERT INTO ARTICLE (AUTHOR_ID,VOLUME_NO,MAGAZINE_ID,TITLE,PAGE_NO)
SELECT B.AUTHOR_ID,A.VOLUME_NO,C.MAGAZINE_ID,A.TITLE,A.PAGE_NO
FROM ARTICLE_temp A
JOIN AUTHOR B
ON A.AUTHOR_NAME=B.author_name
INNER JOIN MAGAZINE C
ON A.MAGAZINE_NAME=C.name ;



   CREATE TABLE ARTICLE (
  article_id INT NOT NULL GENERATED ALWAYS AS IDENTITY(START WITH 1 INCREMENT BY 1) ,
  author_id INT NULL
```

All(1), Failed(0)
✓ INSERT INTO ARTICLE (AUTHOR_ID,VOLUME_NO,MAGAZINE_ID,TITLE,PAGE_...

All(1), Failed(1)
✗ INSERT INTO ARTICLE (AUTHOR_ID,VOLUME_NO,MAGAZINE_ID,TITLE,PAGE_...

All(1), Failed(0)
✓ CREATE TABLE ARTICLE (article_id INT NOT NULL GENERATED ALWAYS AS I...

All(1), Failed(0)
✓ drop table ARTICLE

All(1), Failed(1)

The statement ran successfully.

SQL statement

```
INSERT INTO ARTICLE (AUTHOR_ID,VOLUME_NO,MAGAZINE_ID,TITLE,PAGE_NO)
SELECT B.AUTHOR_ID,A.VOLUME_NO,C.MAGAZINE_ID,A.TITLE,A.PAGE_NO
FROM ARTICLE_temp A
JOIN AUTHOR B
ON A.AUTHOR_NAME=B.author_name
INNER JOIN MAGAZINE C
ON A.MAGAZINE_NAME=C.name
```

Execution log

Run time: 0.139 s
Status: SUCCEEDED
Database: BLUDB
Rows affected: 5

Fig 5.2 Execution in IBM DB2

6. **References:**

- https://support.office.com/en-us/article/database-design-basics-eb2159cf-1e30-401a-8084-bd4f9c9ca1f5
- https://medium.com/@kimtnguyen/relational-database-schema-design-overview-70e447ff66f9
- https://smu.brightspace.com/d2l/le/content/43109/viewContent/434128/View
- https://smu.brightspace.com/d2l/le/content/43109/viewContent/438528/View
- https://www.geeksforgeeks.org/query-optimization/
- https://www.w3schools.com/
- https://codepen.io/mtbroomell/pen/yNwwdv
- https://www.w3schools.com/