# DATABASE PROJECT REPORT

MCDA5510

Trishla Shah

Prepared By:

Bhagya Shree (A00431152)

Madeleine Leong (A00430926)

Mohammad Nawaz (A00428036)

Rishab Gupta (A00429019)

# 1. Introduction

## 1.1 Purpose and Scope

Halifax Science Library (HSL) has its own SQL database that containing various magazine publications. However, HSL has come out with three immediate plans to maintain its database for not only publications but also other activities. These three plans are as following:

1. To record all articles for each magazine
2. To record transactions' history
3. To record monthly expenses of HSL

These plans will help HSL to improve its existing database design as well as provide good and efficient operating performance in its organization.

## 1.2 Project Executive Summary

To achieve the plans HSL has, a thorough understanding on the data requirements is crucial, and then design the relational schema according to the data requirements. After designing the relational database, tables are created through executing the MySQL script. Two bash scripts are written and run to create Mongo collection from the existing JSON file, transfer it to CSV files and then import it to MySQL database schema. Finally, a PHP web application with five main operations is developed for achieving HSL personnel's initial plans.

## 1.3 Tools and Technologies Used

Various tools and technologies are used in this project to help to create the database and develop the web application for HSL personnel. Below is the list of tools used in the project:

| | |
|---|---|
| EER Diagram | : Online Diagramming tool, creately.com |
| Relational Database Schema | : Online Database Modelling tool, ERDPlus.com |
| Query Optimization | : MySQL Workbench 8.0 CE |
| Data2mongo | : JavaScript, Terminal, Sublime Text |
| Mongo2sql | : Python, Terminal, Sublime Text |
| PHP Web Application | : Sublime Text |
| Comparing Execution Time | : MySQL Workbench 8.0 CE and IBM Db2 Warehouse |

# 2. Design of the Relational Database Schema

## 2.1 Enhanced Entity Relational (EER) Diagram

An enhanced entity relational (EER) diagram should be first designed before creating relational database schema for HSL because it helps to understand all the data requirements needed for HSL and identify the entities in the EER diagram. After identifying the entities, relationships between entities are created and attributes for all entities are also added. Lastly, cardinalities between entities are given. In the EER design, there is a superclass that is ‘items’ and the subclass for it is ‘magazines’ and ‘books’, meaning that an item is a magazine or a book. Moreover, there are derived attributes in the design, for instance, the total price, that is calculated based on the total of quantity price. Figure 1 and 2 are the overview of the EER diagram for HSL.

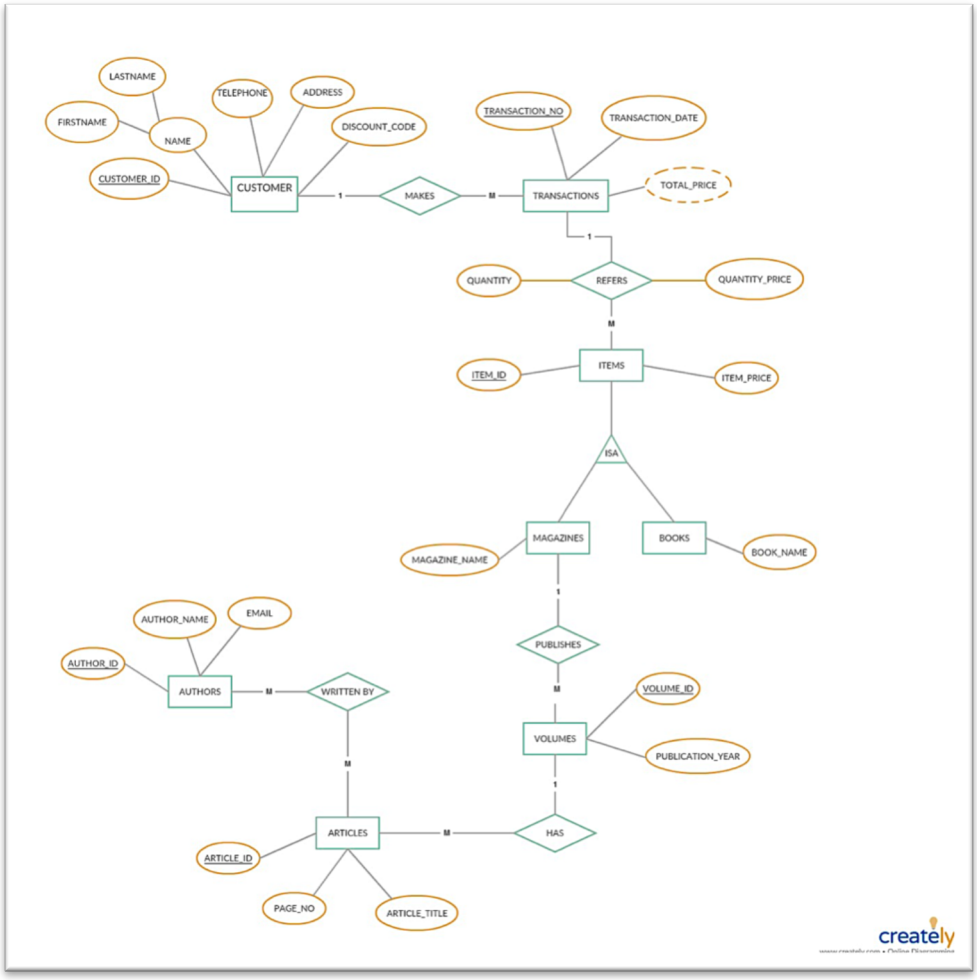



Figure 2.1.1: EER diagram for new library items, customers and transactions

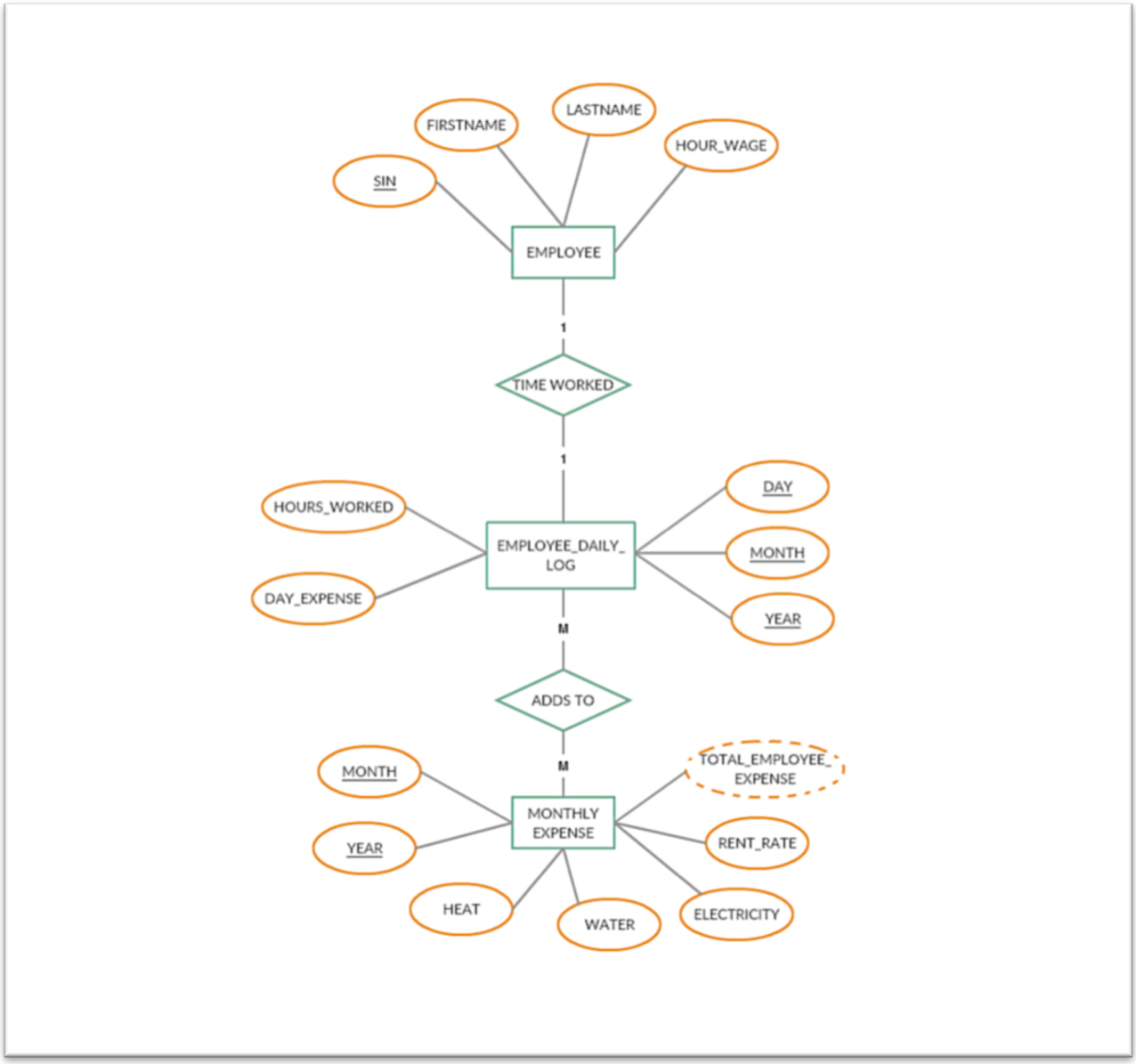



Figure 1.1.2: EER diagram for HSL monthly expenses

## 2.2 Mapping of EER Diagram to Relational Database

After the completion of the EER design, a mapping process from the EER diagram to relational database is done through creating table for each entity, making attributes as the fields of the tables and declaring primary keys and foreign keys. Not to lose any information and make sure all information is one-to-one, two extra tables are created during the mapping process: ARTICLE_AUTHOR and TRANSACTION_ITEM.

### 2.2.1 Normalization

- All relations are in third normal form (3NF) because they are in second normal form and no non-candidate-key attributes are transitively dependent on any candidate keys (i.e., also on primary key).
- All relations are in Boyce-Codd normal form (BCNF) because every determinant is a candidate key.
  For example, the primary key (Article_Id) in ARTICLE table can determine all the attributes directly, hence, it is in 3NF.
  None of the non-prime attributes (Article_title, Page_No and Volume_Id) are determining primary key, therefore it is in BCNF.

| Article_Id | Article_title | Page_No | Volume_Id |
|---|---|---|---|
| | | | |

Table 2.2.1: ARTICLE table

Below is the list of tables with its own attributes:

1. ITEM (Item_Id, Item_price)
2. MAGAZINE (Magazine, Magazine Name)
   *Foreign Key* MAGAZINE (Magazine_Id) *references* ITEM (Item_Id)
3. MAGAZINE_VOLUME (Volume_Id, Magazine_Id, Publication_Year)
   *Foreign Key* MAGAZINE_VOLUME (Magazine_Id) r*eferences* ITEM (Item_Id)
4. ARTICLE (Article_Id, Article_title, Page_No, Volume_Id)
   *Foreign Key* ARTICLE (Volume_Id) *references* VOLUME (Volume_Id)
5. AUTHOR (Author_Id, Author_name, email)
6. ARTICLE_AUTHOR (Art_Auth_Id, Article_Id, Author_Id)
   *Foreign Key* ARTICLE_AUTHOR (Article_Id) *references* ARTICLE (Article_Id)
   *Foreign Key* ARTICLE_AUTHOR (Author_Id) *references* AUTHOR (Author_Id)
7. CUSTOMER (Customer_Id, Firstname, Lastname, Telephone, Address, Discount_Code)
8. TRANSACTION (Transaction_Number, Total_Price, Customer_Id)
   *Foreign Key* TRANSACTION (Customer_Id) *references* CUSTOMER (Customer_Id)
9. TRANSACTION_ITEM (Tran_Item_Id, Quantity, Quantity_price, Item_Id, Transaction_Number)
   Foreign Key TRANSACTION_ITEM (Item_Id) *references* ITEM (Item_Id)
   *Foreign Key* TRANSACTION_ITEM (Transaction_Number) *references* TRANSACTION (Transaction_Number)
10. EMPLOYEE (SIN, Firstname, Lastname, Hour_wage)

11. MONTHLY_EXPENSE (Month, Year, Hear, Water, Electricity, Total_employee_expense)
12. EMPLOYEE_DAILY_LOG (Day, Month, Year, SIN, Hours_Worked, Day_expense)

    *Foreign Key* EMPLOYEE_DAILY_LOG (Month)
    *references* MONTHLY_EXPENSE (Month)
    *Foreign Key* EMPLOYEE_DAILY_LOG (Year)
    *references* MONTHLY_EXPENSE (Year)
    *Foreign Key* EMPLOYEE_DAILY_LOG (SIN)
    *references* EMPLOYEE (SIN)

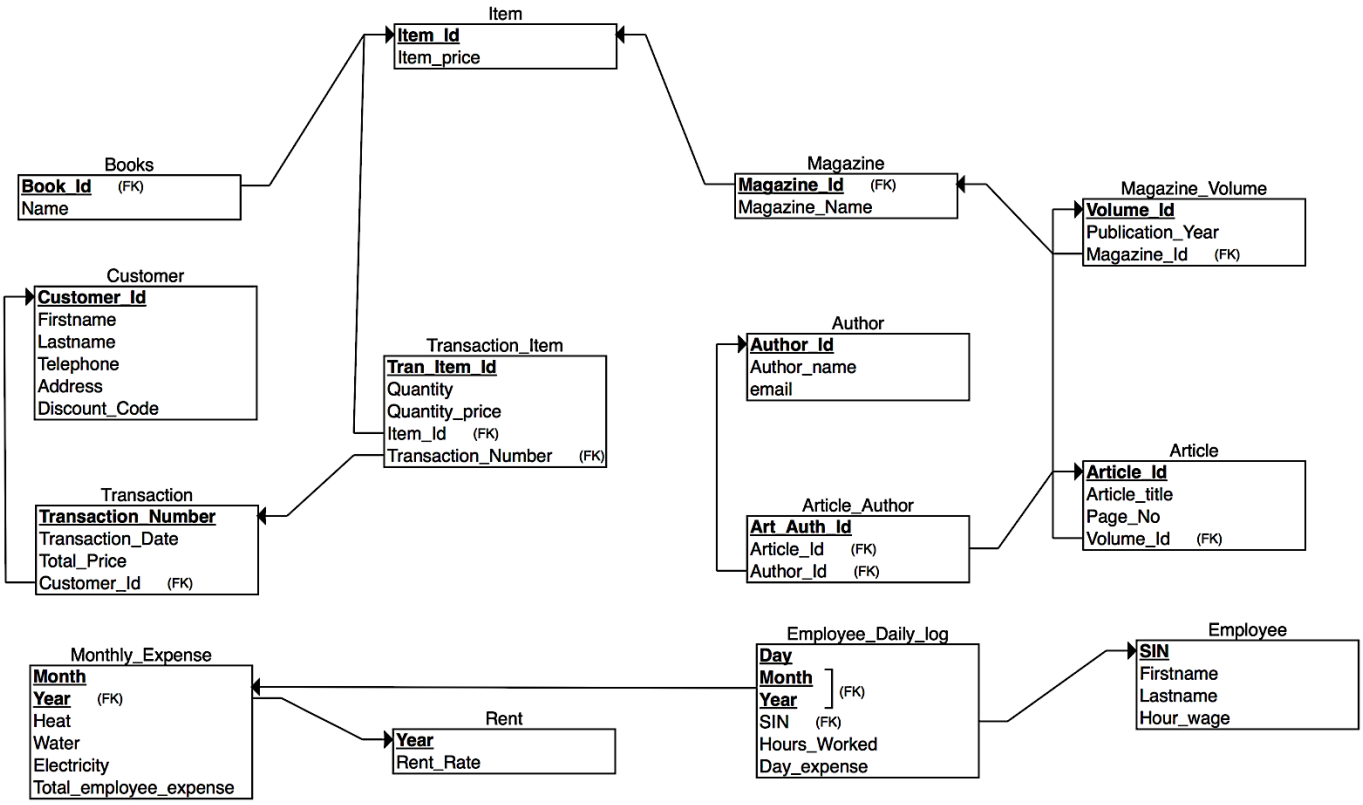



Figure 2.2.1: Relational database schema for Halifax Science Library

## 2.3 Design Guidelines Followed for HSL Database Design

HSL database schema design has followed four guidelines as below:

1. Semantics of the Attributes
   The relational schema is formed with containing some meaning among the attributes for each table. For example, table 1 solely contains customer information details. It is easy for library employees to record or track specific customer from this table.

| Customer_Id | Firstname | Lastname | Telephone | Address | Discount_Code |
|---|---|---|---|---|---|
| | | | | | |

Table 2.3.1: CUSTOMER Table

2. Reducing the Redundant Value in Tuples
   Redundancy of Information is reduced by separating multiple entities in our relational model. Additionally, removing the redundancy in the tuples will prevent anomalies of update, insertion, deletion and modification in our model. For instance, instead of having one table for author and article, and having same author for multiple articles, article and author entities are separated to different tables, and a new table named ARTICLE_AUTHOR is created so that the values will be mapped one-to-one and without losing any information.
3. Reducing Null Values in Tuples
   Grouping too many attributes may bring us many nulls in the tuples and the end user, HSL staffs will have problem understanding the meaning of the attributes. To avoid such problem, some attributes should be ungrouped. For example, article and author attributes are related as author writes article, but these two entities are separated in order to avoid having one record of containing article's values but not author's value when author can't be identified.
4. Disallowing Spurious Tuples
   Our schema is designed as such we have prevented the generation of spurious tuples and erroneous results after joining the tuples of our database, e.g. by joining the customer and transaction table with the equality condition of customer_id, there is no additional tuples created from new table as the customer_id attribute is used to join in which it is a primary key in customer table as well as a foreign key in transaction table.

## 3. Optimization

Below are three steps to optimize our relational database in order to build a cost-effective relational model:

1. Use Joins to connect Foreign Key constraints rather than using WHERE:
   For example, in the Article and Article_Author Table, rather than using WHERE Article.Article_ID = Article_Author.Article_ID we should use an inner join on these fields to ensure better performance. This is because a WHERE clause will use a cross join, matching every possible outcome then filtering out those that we require.
2. Try to reduce usage of Wildcards:
   Wildcard usage for finding records is very inefficient as it can include any types of data containing the characters mentioned between '%'. However, in our database design for our queries we will try our best to limit this to only the end of the phrase, to ensure that we reduce the cost of matching from both sides of the phrase.
3. Indexing
   Indexing allows the sorting of data on a column, so a query knows where exactly to search, thereby reducing the time and improving efficiency. The query will not need to run a FULL-TABLE-Scan, that is scanning each record until it finds the right one to output the results.
   Since we propose to use joins instead of where, we need to take into consideration the SQL constraint of leaning on ONLY ONE index per joined table per query. Even though we could technically have more indexes it will not be beneficial in the joins. Therefore, we have to index only column against a table in our join to improve the efficiency of our database
   We will target our indexing to be placed in tables which have a higher cardinality. This is to maximize our efficiency gain. An index for instance on a field which has 5 values like Discount code will not gain us much efficiency as opposed to placing an index on the Customer ID. This is because the B-Tree will be able to eliminate more records that do not match our criteria
   We need to be mindful however that since indexing will speed up our SELECT and WHERE clauses, it will slow down our UPDATE and INSERT statements.
   Below is an example to show using index is better than using full table scan to find author email.

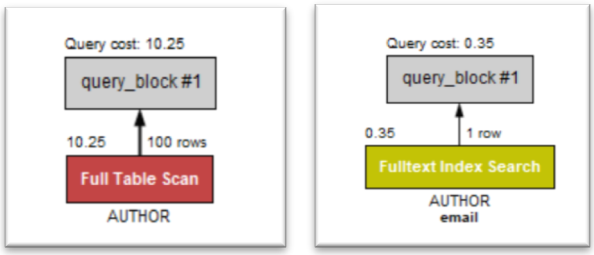



Figure 3.1: FULL-TABLE-Scan and Index Search from AUTHOR table

## 4. Create New Tables in MySQL

A *new_tables.sql* MySQL script is written to create new tables according to the relational database schema. In the script, all existing tables will be dropped, and new tables will be created by declaring the primary keys and foreign keys as well as adding the constraints for certain tables. For examples, there is a constraint in magazine table where 'magazine_id' is a foreign key of 'item_id' because magazine is a subclass of item.

```
DROP TABLE if exists ITEM;

DROP TABLE if exists AUTHOR;
Drop TABLE if exists TRANSACTION;
Drop TABLE if exists CUSTOMERS;

Drop TABLE if exists Employee_Daily_log ;
Drop TABLE if exists MONTHLYEXPENSES;
Drop TABLE if exists Rent;
Drop TABLE if exists Employee;



create table if not exists ITEM (
  _id INT not null,
  price FLOAT(8,2) not null,
  primary key(_id)
) engine = innodb;

CREATE TABLE IF NOT EXISTS BOOKS (
    book_id INT NOT NULL ,
    book_name VARCHAR(15) NOT NULL,
    PRIMARY KEY (book_id),
    CONSTRAINT fk_books FOREIGN KEY (book_id)
        REFERENCES ITEM(_id)
)  ENGINE=INNODB;

create table if not exists MAGAZINE (
  _id INT not null ,
  name varchar(50) not null,
  primary key(_id),
  CONSTRAINT fk_magazines FOREIGN KEY (_id)
        REFERENCES ITEM(_id)
) engine = innodb;
```

Figure 4.1: Snippet of *new_tables.sql*

## 5. Create Mongo Collections from Given Data

A data2mongo Bash script is written and run to perform the following tasks:

1. Connects to MySQL server and execute *existing_tables.sql* and *new_tables.sql*:
   The two sql files first drop the tables if they exist and then create the tables according to our relational database schema design. The *new_tables.sql* creates 14 tables to reflect our database design
2. Drops and Creates a mongo collection AUTHOR
3. The AUTHOR data is then imported from the *author_json.json file*
4. Creates ARTICLE and TEMPARTICLE collection:
   The two mongo collections ARTICLE and TEMPARTICLE are dropped before creating them.
5. *Article_temp.json* file is imported into the TEMPARTICLE collection:
   Since the given article.json file contains enormous records about the articles' publications. An article_temp.json containing first 100 records of the data is imported to a Mongo collection called TEMPARTICLE.
6. Articleparse.js is executed to fetch the clean data from TEMPARTICLE and import to ARTICLE collection:
   Since, there is useless information regarding the articles in the collection, to avoid this circumstance, a JavaScript, articleparse.js is written and run to read the initial collection (TEMPARTICLE) and store the required information to a collection called ARTICLE. The required fields parsed from the initial collection are:
   i. Article ID – which is an auto generated ID field
   ii. Article Title
   iii. Magazine Name
   iv. Volume ID
   v. Publication Year
   vi. Page No
   vii. Authors

7.Articleparse.js explained:
   The result of the mongo collection TEMPARTILCE is stored in a variable ptr. A while loop is used to loop through the collection. An array of AUTHORS is created, and a check is performed if the AUTHORS in the TEMPARTICLE has a single author or has an array of multiple authors. If an article has multiple authors, it is appended to the article array and finally the aforementioned fields are fetched and stored in new variable row. Finally, the row is inserted into the ARTICLE collection which now contains the clean data.

```
#!/bin/bash
#
user=b_shree
pass=A00431152
db=b_shree

#


#step 1 - load sql data from existing_tables.sql
mysql -u $user --password=$pass $db -e "source existing_tables.sql"

#step 2 - create mysql table new_tables.sql
mysql -u $user --password=$pass $db -e "source new_tables.sql"

#step 3 - Drop  mongo collection AUTHOR
mongo $user -u $db -p $pass --eval "db.AUTHOR.drop()"

#step 4 - create mongo collection AUTHOR
mongo $user -u $db -p $pass --eval "db.createCollection('AUTHOR')"

#step 5 - import author json data into mongo collection
mongoimport -d $db -p $pass -u $user -c AUTHOR --file author_json.json --jsonArray

#step 6 - Drop  mongo collection Article and temparticle
mongo $user -u $db -p $pass --eval "db.ARTICLE.drop()"
mongo $user -u $db -p $pass --eval "db.TEMPARTICLE.drop()"

#step 7 - create temporary mongo collection
mongo $user -u $db -p $pass --eval "db.createCollection('TEMPARTICLE')"

#step 8 - import tempjson  into mongo collection
mongoimport -d $db -p $pass -u $user -c TEMPARTICLE --file article_temp.json --jsonArray

#step 9 - create mongo collection ARTICLE
mongo $user -u $db -p $pass --eval "db.createCollection('ARTICLE')"

#step 10 - run javascript to insert element
mongo $user -u $db -p $pass --eval "load('articleparse.js')"
```

Figure 5.1: Snippet of data2mongo.sh

```
var auto_id = 0;
var ptr = db.TEMPARTICLE.find();

while (ptr.hasNext()){
    var doc = ptr.next();
     var authors= new Array();

        // for(i in doc.author)
        // authors.push(doc.author[i].ftext)

         if(Array.isArray(doc.author))
        {
         for (var i=0;i<doc.author.length;i++)
               authors[i]=doc.author[i].ftext;
         }
         else
            authors[0]=doc.author.ftext;

        var row = {"article_id":auto_id,"article_title":doc.title.ftext,"name":doc.journal.ftext, "vol_id":doc.volume.ftext,
          "year":doc.year.ftext, "page_num":doc.pages.ftext,"authors":authors};
        db.ARTICLE.insert(row);
        auto_id++;

}
```

Figure 5.2: Snippet of articleparse.js

8. Data2mongo.sh is executed

```
r_gupta@dev:~/DBProject/FINAL$ ./data2mongo.sh
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.
MongoDB shell version: 3.2.16
connecting to: r_gupta
true
MongoDB shell version: 3.2.16
connecting to: r_gupta
{ "ok" : 1 }
2018-11-28T19:07:26.609-0400    connected to: localhost
2018-11-28T19:07:26.615-0400    imported 100 documents
MongoDB shell version: 3.2.16
connecting to: r_gupta
true
MongoDB shell version: 3.2.16
connecting to: r_gupta
true
MongoDB shell version: 3.2.16
connecting to: r_gupta
{ "ok" : 1 }
2018-11-28T19:07:27.090-0400    connected to: localhost
2018-11-28T19:07:27.102-0400    imported 100 documents
MongoDB shell version: 3.2.16
connecting to: r_gupta
{ "ok" : 1 }
MongoDB shell version: 3.2.16
connecting to: r_gupta
true
```

Figure 5.3: Data2mongo executed

## 6. Update MySQL Tables from the Mongo Collections

```
user=b_shree
pass=A00431152
db=b_shree
sleeptime=3
#
#PLEASE RUN THIS COMMAND BEFORE EXECUTING THIS FILE: pip install pandas
#step 1 - export mongo article collection to a csv file

mongoexport -d $db -p $pass -u $user -c ARTICLE --type=csv --fields article_id,article_title,authors,name,vol_id,page_num,year

#step 2 - export mongo author collection to a csv file

mongoexport -d $db -p $pass -u $user -c AUTHOR --type=csv --fields _id,lname,fname,email --out authors.csv


#step 3 - python scripts
python Item.py
python add_to_Author.py
python New_Article.py
python Magazine_Table.py
python Magazine_Volume.py
python Article_Author.py


#step 3 - run  script to convert mongo(csv) to mysql
mysql -u "$user" --password=$pass "$db" -e "load data local infile 'Item.csv' into table ITEM fields terminated by ',' lines te
mysql -u "$user" --password=$pass "$db" -e "load data local infile 'authors.csv' into table AUTHOR fields terminated by ',' lir
mysql -u "$user" --password=$pass "$db" -e "load data local infile 'Magazines.csv' into table MAGAZINE fields terminated by ','
mysql -u "$user" --password=$pass "$db" -e "load data local infile 'Magazine_Volume.csv' into table MAGAZINE_VOLUME fields term
mysql -u "$user" --password=$pass "$db" -e "load data local infile 'New_Article.csv' into table ARTICLE fields terminated by ',
mysql -u "$user" --password=$pass "$db" -e "load data local infile 'article_author.csv' into table ARTICLE_AUTHOR fields termir


# Delete all csv files
rm Item.csv authors.csv article_author.csv Magazines.csv Magazine_Volume.csv New_Article.csv articles.csv
```

Figure 6.1: Snippet of data2mongo.sh

A mongo2sql Bash script is written and run to perform the following tasks:
Please run this command in the server before executing mongo2sql bash script:

```
Pip install pandas
```

1. Export two existing mongo collections AUTHOR and ARTICLE to CSV file:
   Two existing mongo collections ARTICLE and AUTHOR are exported to *articles.csv* and *authors.csv* respectively to further create CSV files according to our relational database schema
2. Run several python scripts to segregate the CSV file:
   Six python scripts are written and run in the bash script to segregate the two existing CSV files into multiple CSV files to match the relational database schema. Following are the six python scripts used:
   i. *Item.py*
   ii. *add_to_Author.py*
   iii. *New_Article.py*
   iv. *Magazine_Table.py*
   v. *Magazine_Volume.py*
   vi. *Article_Author.py*

   One of the above scripts (*Magazine_Volume.py*) is explained below.

2.1. Magazine_Volume.py explained:
   Each row from the *articles.csv* is traversed and the Magazine name in the articles file is checked for the name in the *Magazines.csv* file. If the name matches, the Magazine_ID is fetched and added into the *magazine_volume.csv* file.

```
import csv

Volume_id = ['vol_id']
Pub_Year = ['publication_year']
Magazine_ID = ['magazine_id']

with open(r"articles.csv") as csvfile:
    readCSV = csv.DictReader(csvfile, delimiter=',')
    for row in readCSV:
        if row['vol_id'] in Volume_id:
            continue
        Volume_id.append(row['vol_id'])
        Pub_Year.append(row['year'])
        name = row['name']

        with open(r"Magazines.csv") as csvfile2:
            readCSV2 = csv.DictReader(csvfile2, delimiter=',')
            for rowm in readCSV2:
                if name == rowm['name']:
                    Magazine_ID.append(rowm['_id'])


X = [list(a) for a in zip(Volume_id, Pub_Year, Magazine_ID)]

with open (r"Magazine_Volume.csv", "w") as csvfile:
    csvwriter = csv.writer(csvfile)
    csvwriter.writerows(X)
```

Figure 6.2: Snippet of magazine_volume.py

3. Import CSV files to MySQL database schema:
   The resulting CSV files are then imported into the newly created tables in MySQL database schema.
4. Mongo2sql.sh executed

```
r_gupta@dev:~/DBProject/FINAL$ ./mongo2sql.script
2018-11-28T19:08:55.739-0400    connected to: localhost
2018-11-28T19:08:55.745-0400    exported 100 records
2018-11-28T19:08:55.787-0400    connected to: localhost
2018-11-28T19:08:55.790-0400    exported 100 records
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.
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.
```

Figure 6.3: A successful execution on mongo2sql Bash Script

## 7. PHP Web Application

A PHP web application is implemented for HSL personnel to maintain their relational schema and record useful information to their database. Below are the main operations which have multiple PHP scripts to perform the corresponding tasks.

| **Main Page** | |
|---|---|
| PHP Script Name | : main.php |
| Link | : http://dev.cs.smu.ca/~b_shree/DBProject/main.php |
| Description | : Allows user to choose one of the main operations from a list. The main operations are included *Show Table, Add New Article, Add New Customer, Add New Transaction, Add New Volume, Add New Magazine, Cancel Transaction*. |

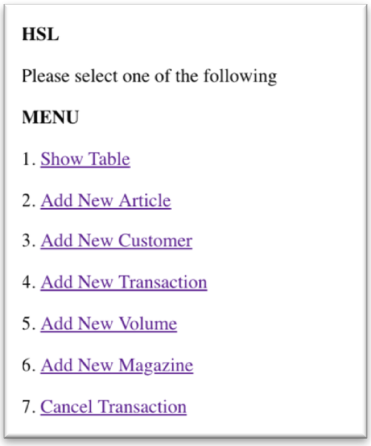



Figure 7.1: Main page of Halifax Science Library PHP web application

| Show Table | |
|---|---|
| PHP script name | : get_table_name.php |
| Link | : http://dev.cs.smu.ca/~b_shree/DBProject/get_table_name.php |
| Description | : Displays all tables on a list and allows user to enter table's name on the textbox provided and be able to hit submit button to get the table shown on the next page |

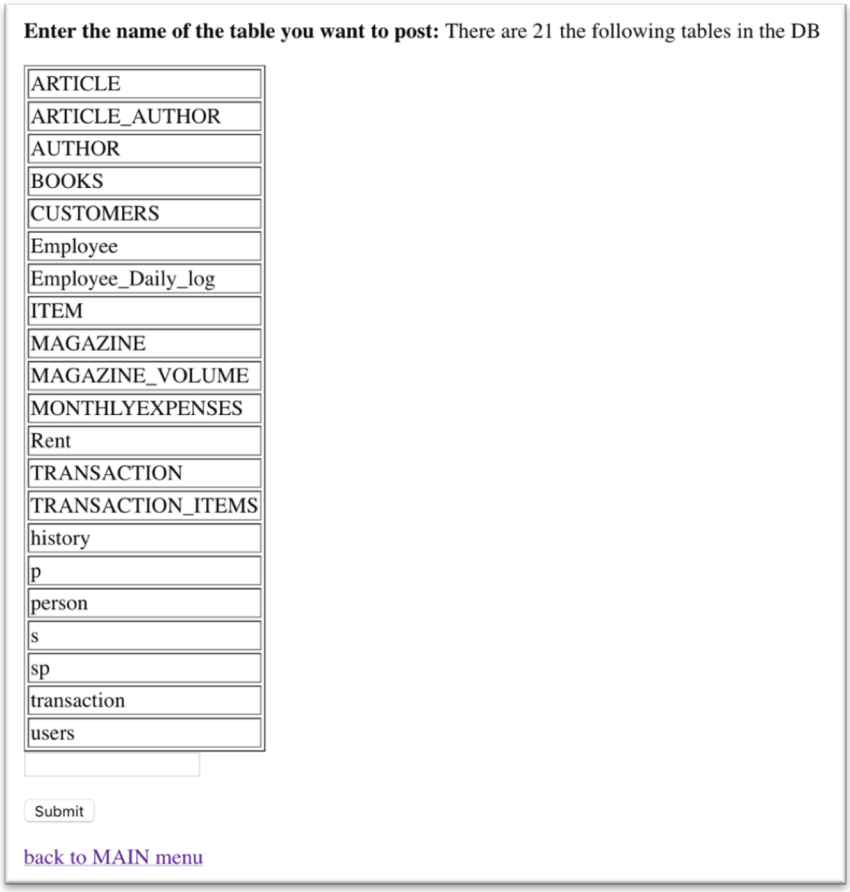



Figure 7.2: Show Table webpage

| Show Table (continued) | |
|---|---|
| PHP script name | : print_table.php |
| Link | : http://dev.cs.smu.ca/~b_shree/DBProject/print_table.php |
| Description | : Print out the table that user has entered above. If the table name is correct, the contents of that table are displayed or else, an error message is shown |

Successfully connected to server

Successfully selected database "b_shree"

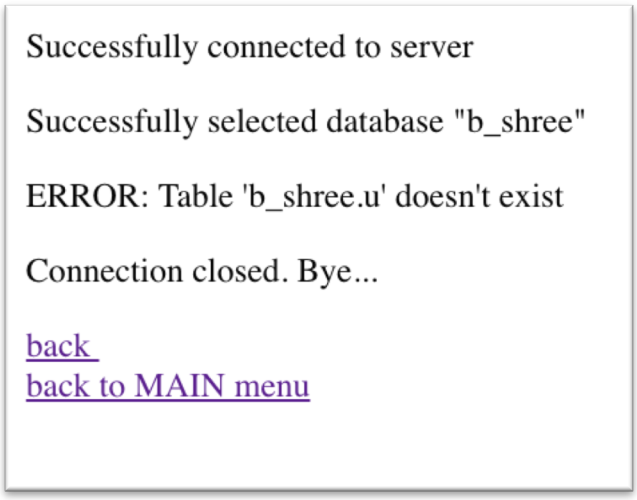
There are 8 rows in the table

| | | | |
|---|---|---|---|
| 1 | 2 | hello | 12 |
| 8 | 2 | sends | 23 |
| 9 | 2 | fdgssfdgfs | 45 |
| 10 | 2 | fads | 20 |
| 11 | 2 | hello again | 67 |
| 16 | 2 | fads | 78 |
| 18 | 2 | fdsa | 56 |
| 22 | 1 | hello | 78 |

Connection closed. Bye...

back
back to MAIN menu

Figure 7.3: ARTICLE table displayed



Successfully connected to server

Successfully selected database "b_shree"

ERROR: Table 'b_shree.u' doesn't exist

Connection closed. Bye...

back
back to MAIN menu

Figure 7.4: Error message shown when enter invalid table name

| Add New Article | |
|---|---|
| PHP script name | : addarticle.php |
| Link | : http://dev.cs.smu.ca/~b_shree/DBProject/addarticle.php |
| Description | : Allows user enter article details and its author(s) details. Author one is mandatory, while the next two are optional. Pre-requisites are magazine ID and volume ID should be present in respective tables in order for the Article to be inserted. |

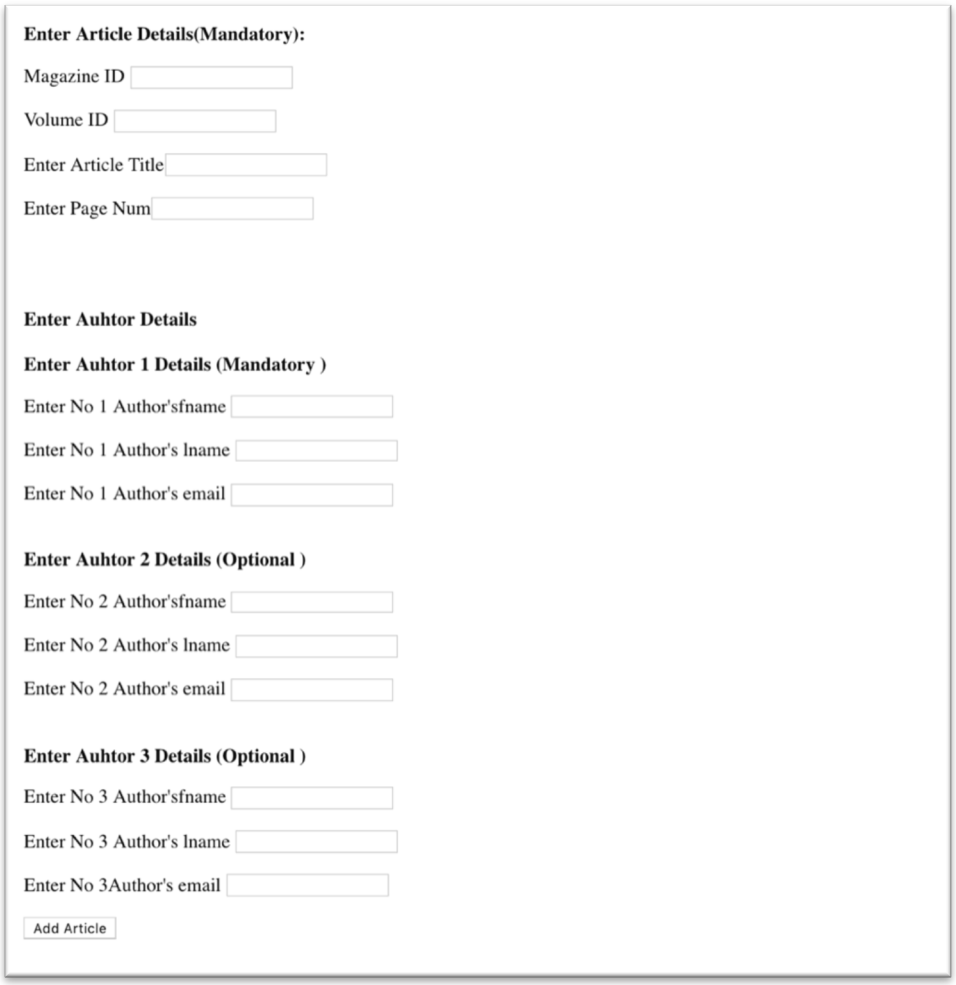



Figure 7.5: Add Article webpage

| Add New Article (continued) | |
|---|---|
| PHP script name | : insertarticle.php |
| Link | : http://dev.cs.smu.ca/~b_shree/DBProject/insertarticle.php |
| Description | : The application will store the information to the relational database. Values are successfully added if magazine ID and volume ID are existed, if not, the application will prompt user to add magazine information or volume information |

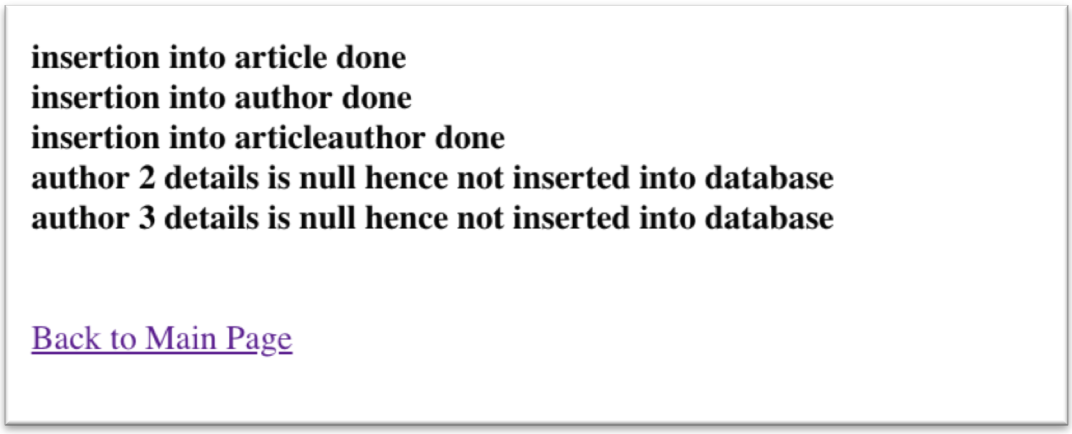



Figure 7.6: Succesful insertion of article information to database

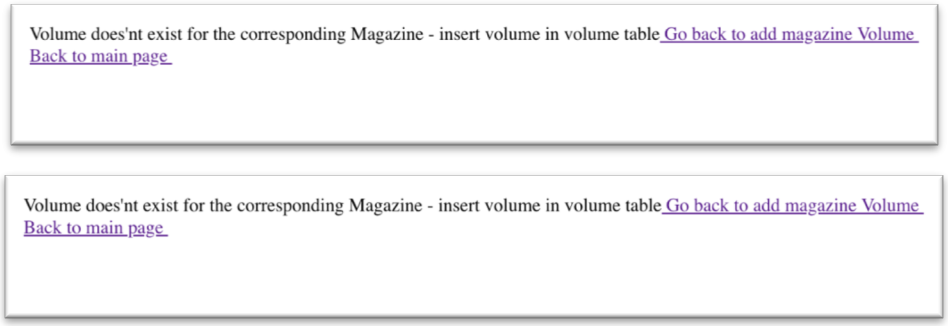



Figure 7.7: Failed to insert article information to database because magazine or volume doesn't exist.

| Add New Magazine | |
|---|---|
| PHP script name | : addMagazine.php |
| Link | : http://dev.cs.smu.ca/~b_shree/DBProject/addMagazine.php |
| Description | : Allows user to create a magazine object in the database by entering the Magazine ID, Magazine Price and Magazine Name. |


Figure 7.8: addMagazine webpage

| | |
|---|---|
| PHP script name | : insertMagazine.php |
| Link | : http://dev.cs.smu.ca/~b_shree/DBProject/insertMagazine.php |
| Description | : After user submits magazine details, the application is able to insert the values to the appropriate tables in the MySQL database, if failed to insert, an error message will be shown. |

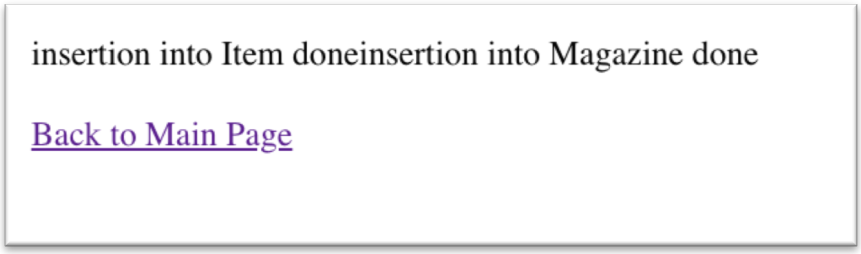



Figure 7.9: Successful insertion of magazine details

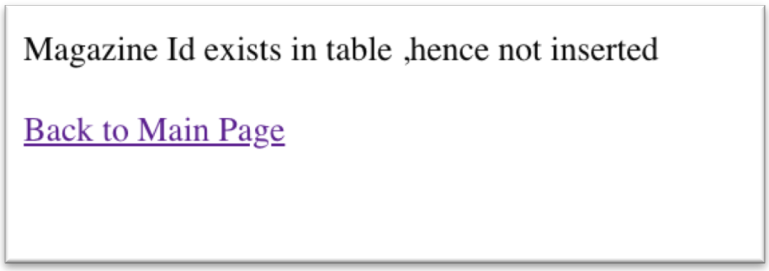



Figure 7.10: Error message if failed to insert magazine details

| Add New Volume | |
|---|---|
| PHP script name | : addVolume.php |
| Link | : http://dev.cs.smu.ca/~b_shree/DBProject/addVolume.php |
| Description | : Allows user to enter magazine volume details |

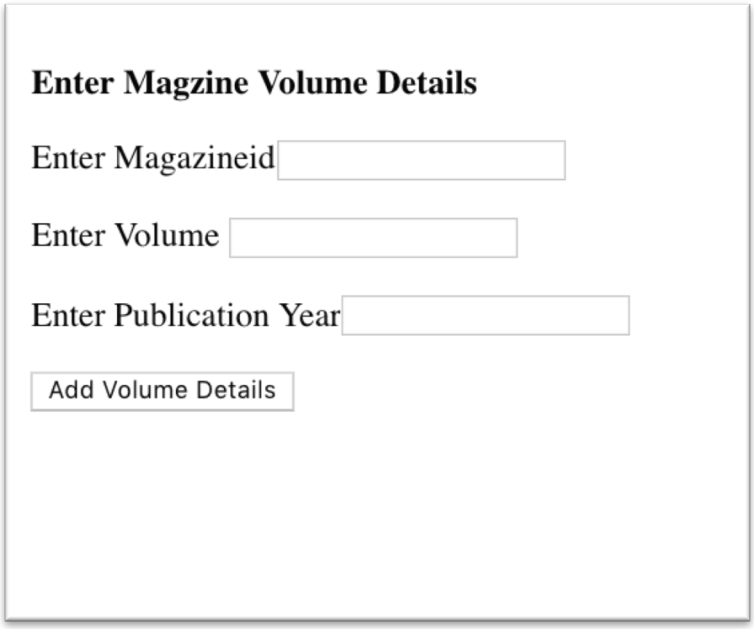



Figure 7.11: addVolume webpage

| | |
|---|---|
| PHP script name | : insertVolume.php |
| Link | : http://dev.cs.smu.ca/~b_shree/DBProject/insertVolume.php |
| Description | : After user submits magazine volume details, a message on the insertion status will be shown. |

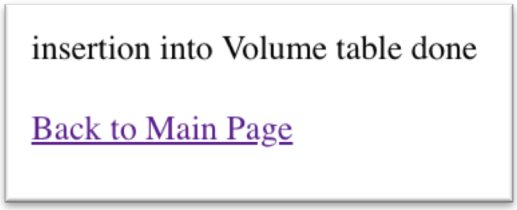



Figure 7.12: Successful insertion of magazine volume details

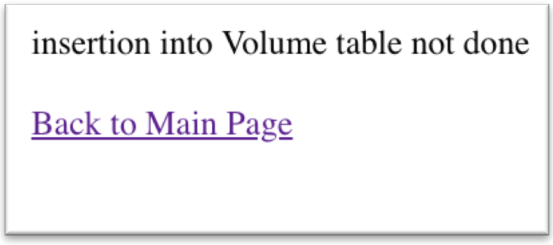



Figure 7.13: Error message on failure to insert magazine volume details

| Add New Customer | |
|---|---|
| PHP script name | : AddCustomer.php |
| Link | : http://dev.cs.smu.ca/~b_shree/DBProject/AddCustomer.php |
| Description | : Allows user to enter customer details |

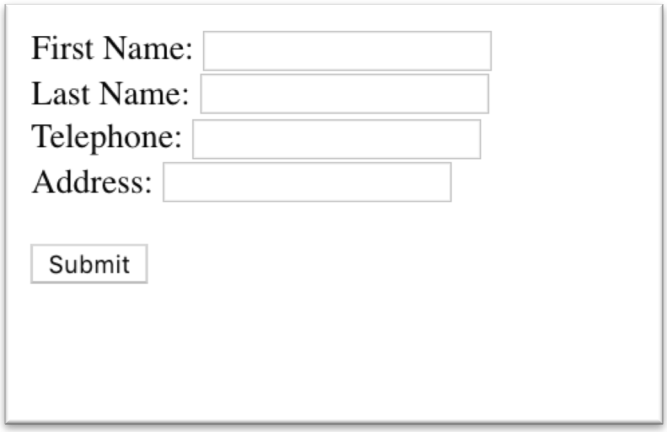



Figure 7.14: AddCustomer webpage

| PHP script name | : insertCustomer.php |
|---|---|
| Link | : http://dev.cs.smu.ca/~b_shree/DBProject/insertCustomer.php |
| Description | : The application will store the customer information to related tables in MySQL database. If the first name and last name of the customer exist in the database, a prompt will ask if the user wants to proceed. |


Figure 7.15: A new customer is successfully added

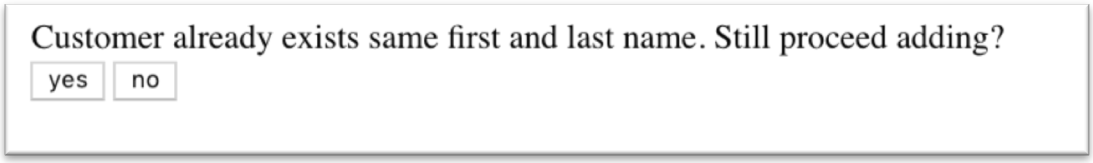



Figure 7.16: Prompting user to proceed adding same last name and first name of customer to the database

| Add New Transaction | |
|---|---|
| PHP script name | : addTransaction.php |
| Link | : http://dev.cs.smu.ca/~b_shree/DBProject/addTransaction.php |
| Description | : Allows user to enter transaction details where user can enter multiple item ID and corresponding prices with ‘,’ separation, for instance, in the item ID textbox, user can enter “1,2,3” and in the item quantity textbox, user can enter “2,3,4”. Additionally, user can choose customer from his/her customer ID on a dropdown list. |

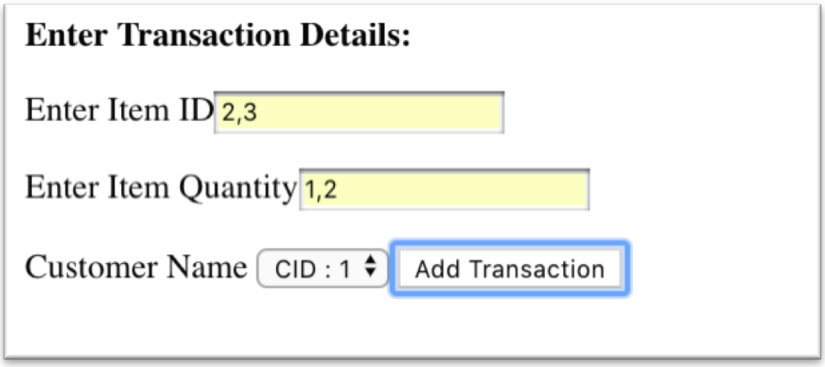



Figure 7.17: addTransaction webpage

| | |
|---|---|
| PHP script name | : insertTransaction.php |
| Link | : http://dev.cs.smu.ca/~b_shree/DBProject/insertTransaction.php |
| Description | : The application will store the transaction information to related tables in MySQL database. The discount code of the customer will be updated in the database if the total amount spent over the past 5 years meet the discount requirement. Also, pre-existing discount code will be taken into account of the current transaction. |

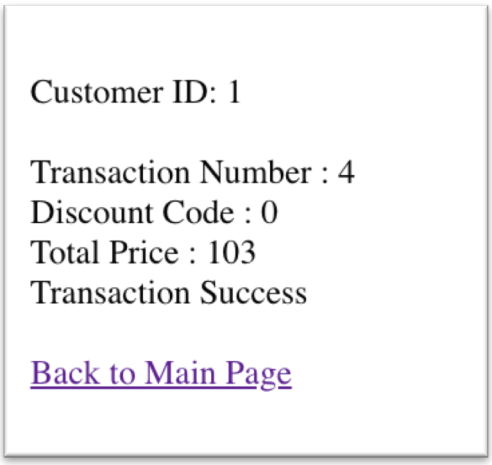



Figure 7.18: Successfully adding a new transaction

| Add New Transaction (continued) |
| --- |
| PHP script name : insertTransaction.php |


Figure 7.19: Failed to add a new transaction

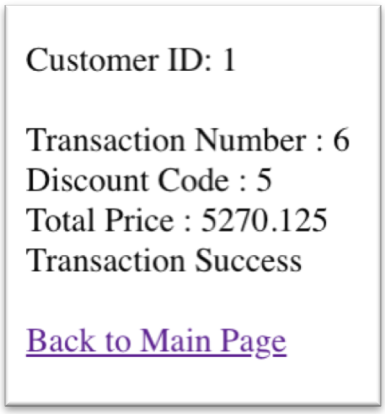



Figure 7.20: The discount code for customer with cid of 1 changed to 5 after spending more than $500

| Cancel Transaction | |
|---|---|
| PHP script name | : cancelTransaction.php |
| Link | : http://dev.cs.smu.ca/~b_shree/DBProject/cancelTransaction.php |
| Description | : Allows user to enter the transaction number that is intended to be removed from the database |

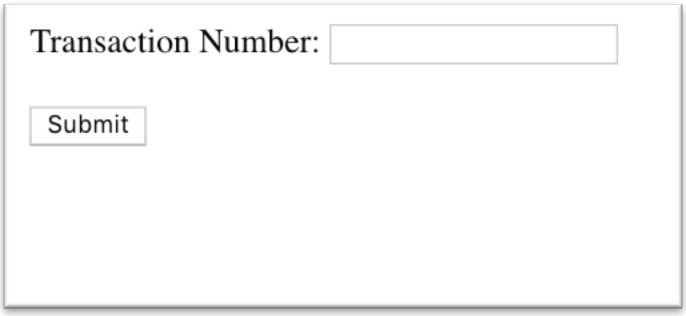



Figure 7.21: cancelTransaction webpage

| PHP script name | : deleteTransaction.php |
|---|---|
| Link | : http://dev.cs.smu.ca/~b_shree/DBProject/deleteTransaction.php |
| Description | : The application will remove the transaction record from the database. An error message will be shown if the transaction number is not existed or is more than 30 days from current day. |

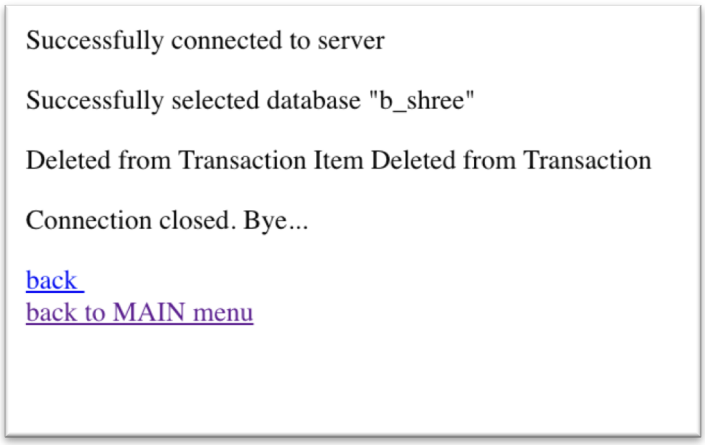



Figure 7.22: Successfully cancel a transaction

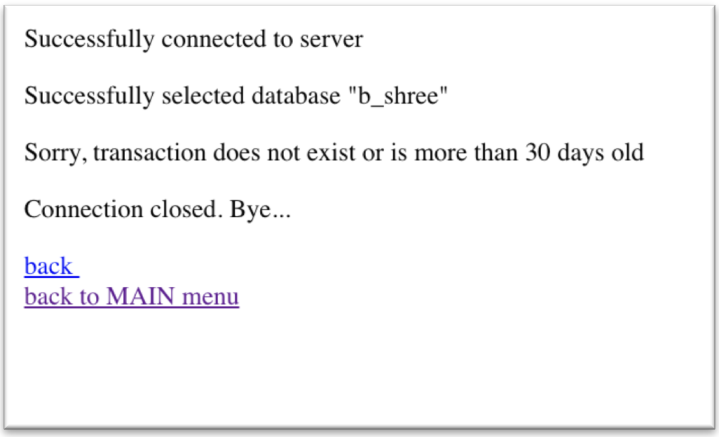



Figure 7.23: Failed to remove a transaction record

## 8. Comparison on Execution Time for MySQL Workbench and DB2

Executing the same queries on different environments can give different execution time. A comparison on the queries' execution time for MySQL Workbench and DB2 is carried out. CUSTOMER and TRANSACTION tables are taken for this comparison. There are 5 records in CUSTOMER table and 2 records in TRANSACTION table. There are 4 main queries to compare: SELECT, INSERT, DELETE, and INNER JOIN. Below are the queries executed on both environment:

1. SELECT firstname, lastname
   FROM CUSTOMER
   WHERE discount_code = 3;
2. INSERT INTO CUSTOMER (firstname, lastname, telephone, address, discount_code) VALUES ("Creadia", "Wang", "9998887777", "88 Youth Street, Dartmouth, NS B7Y 8F4", 5)
3. DELETE FROM TRANSACTION WHERE trans_id = 2;
4. SELECT TRANSACTION.trans_id, CUSTOMERS.cid,
   CUSTOMERS.firstname, CUSTOMERS.lastname
   FROM TRANSACTION INNER JOIN CUSTOMERS
   ON TRANSACTION.cid = CUSTOMERS.cid

| Query | MySQL Workbench | Db2 Warehouse |
|---|---|---|
| SELECT | 0.00156672s | 0.04s |
| INSERT | 0.031s | 0.04s |
| DELETE | 0.015s | 0.027s |
| INNER JOIN | 0.015s | 0.056s |

Table 2: Time taken to execute query on different environment

From the table, the execution time on DB2 Warehouse on Cloud is longer than MySQL Workbench. This may be caused by the network speed and the loading on buffer pool.

**References**


- *Informal Design Guidelines for Relational Schema*. [online] Idc-online. Available at: http://www.idc-online.com/technical_references/pdfs/information_technology/Informal_design_guidelines_for_relational_schema.pdf
- Ginsberg, J. (2018). *3 Things You Should Know About SQL Indexes*. [online] The Celerity Blog. Available at: http://blog.celerity.com/how-to-design-sql-indexes [Accessed 8 Nov. 2018].
- Winnard, M. (2018). *Anatomy of an SQL Index: What is an SQL Index*. [online] Use the Index. Available at: https://use-the-index-luke.com/sql/anatomy
- TP (2018). *SQL - Indexes*. [online] TP. Available at: https://www.tutorialspoint.com/sql/sql-indexes.htm
- Docs.mongodb.com. (2018). mongoexport — MongoDB Manual. [online] Available at: https://docs.mongodb.com/manual/reference/program/mongoexport/#bin.mongoexport
- Stack Overflow. (2018). append new row to old csv file python. [online] Available at: https://stackoverflow.com/questions/2363731/append-new-row-to-old-csv-file-python
- JavaScript Tutorial. Available at: https://www.w3schools.com/js/
- MySQL Select Statment in PHP with Variables - WebDeveloper.com Forums. 2018. MySQL Select Statment in PHP with Variables - WebDeveloper.com Forums. [ONLINE] Available at: https://www.webdeveloper.com/forum/d/176981-mysql-select-statment-in-php-with-variables
- PHP mysqli_num_rows() Function. 2018. PHP mysqli_num_rows() Function. [ONLINE] Available at: https://www.w3schools.com/php/func_mysqli_num_rows.asp
- IBM DB2 Warehouse Tutorial Videos. Available at: https://www.youtube.com/playlist?list=PLo4y_OQEqhGgwwhIJLxBrwFt8b3jYeg4v