

A Database Systems Project Report On

E-Commerce Website with Recommendation System

Submitted by

Divija Nagaraju - 14IT112

Mukta Kulkarni – 14IT220

Pooja Soundalgekar – 14IT230

V SEM B.Tech (IT)

Under the Guidance of

Ms Shruti J. R.

Assistant Lecturer

in partial fulfillment for the award of the degree

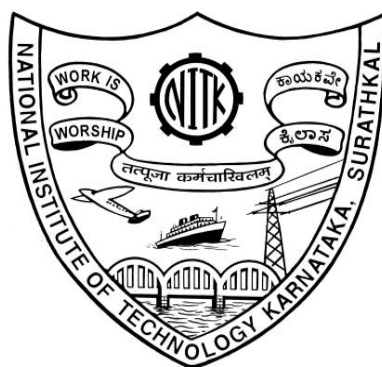
of

Bachelor of Technology

In

Information Technology

At



Department of Information Technology

National Institute of Technology Karnataka, Surathkal.

November 2016

Abstract

This project deals with developing an E-commerce website for an online merchandise store. It provides the user with a variety of available choices sectioned according to the clothing types and accessories. There would be filters to facilitate users in the searching process based on price, popularity, availability and so on. In order to keep track of the online purchase a shopping cart is provided to the user. A shipping portal has been made to manage the contact and delivery details of each user. A user interface of the payment portal is provided.

The system would be implemented using a three tier approach: The front-end and back-end will be developed using the Python-Django framework and linked to MySQL Database. An additional feature to this E-commerce website would be a Recommendation System, based on item based collaborative filtering. The basic concept behind this is that the system recommends based on a user's previous choices. This is an improvement over collaborative filtering since this is independent of the preferences made by other users in the system. The prediction will be done based on likelihood and recommendation based on weighted factors or regression.

ACKNOWLEDGEMENT

We would like to take this opportunity to thank and express our gratitude to all those without whom the completion of this project would not have been possible. This project has helped expand our horizons and improved our analytical skills. We would like to extend our thanks to our project advisor and project co-ordinator, Ms Shruti J. R., Assistant Lecturer, Department of Information Technology for her constant support and guidance. We express my heartfelt gratitude to our HOD, Dr. Ram Mohan Reddy Department of Information Technology who has extended support, guidance and assistance for the successful completion of the project. Finally, we would like to thank all our classmates, friends and seniors for their suggestions and support.

Divija Nagaraju

Mukta Kulkarni

Pooja Soundalgekar

TABLE OF CONTENTS

ABSTRACT.....	
ACKNOWLEDGEMENT	
LIST OF FIGURES.....	
CHAPTER 1: INTRODUCTION.....	
1.1 PROBLEM STATEMENT.....	
1.1.1 ACTORS.....	
1.1.2 SOME SAMPLE QUERIES.....	
1.2 PROJECT OBJECTIVES	
1.3 PROJECT SCOPE	
CHAPTER 2: SYSTEM DESIGN.....	
2.1 LOGICAL DESIGN (CONCEPTUAL MODEL)	
2.2 SCHEMA DIAGRAM.....	
2.3 ADVANCED LOGICAL DESIGN.....	
2.3.1 NORMALIZATION TECHNIQUES.....	
2.3.2 GLOBAL SCHEMA.....	
2.4 QUERIES.....	
2.4.1 CREATION OF TABLES.....	
2.4.2 VIEWS.....	
2.4.3 STORED PROCEDURES.....	
2.4.4 TRIGGERS.....	
CHAPTER 3: PHYSICAL DESIGN.....	
3.1 ASSUMPTIONS.....	
3.2 STORAGE REQUIREMENTS.....	
3.2.1 SPANNED/UNSPANNED RECORDS.....	
3.3 ACCESS METHODS.....	
3.4 TIMING.....	
3.5 SYSTEM SPECIFICATIONS.....	
3.6 QUERY COSTS.....	
CHAPTER 4: IMPLEMENTATION AND RESULTS.....	
CONCLUSION.....	
REFERENCES	
APPENDIX	

CHAPTER 1

INTRODUCTION

1.1 Problem Statement

This project deals with developing an E-commerce website for an online merchandise store. It provides the user with a variety of available choices sectioned according to the clothing types and accessories. There would be filters to facilitate users in the searching process based on price, brand, and availability and so on. In order to keep track of the online purchase a shopping cart is provided to the user. A shipping portal to manage the contact and delivery details of each user. The system would be implemented using a three tier approach: The front-end and back-end will be developed using HTML/CSS server side scripting using PHP and linked to MySQL Database which is the back-end. An additional feature to this E-commerce website would be a Recommendation System, based on item based collaborative filtering. The basic concept behind this is that the system recommends based on a user's previous choices. This is an improvement over collaborative filtering since this is independent of the preferences made by other users in the system. The prediction will be done based on likelihood and recommendation based on weighted factors.

Consider the following set of requirements for a E-commerce database that is used to keep track of the merchandise inventory and customer specific choices

The administrator keeps track of each item's specifics. These include item code, item name, brand images, cost.

Each item is available in a variety of sizes

Each item is grouped according to its category. Each category from the item's parameters has a category name.

Each brand has a discount rate that is applicable for all items of that brand

Each customer has a name (first name and last name), email id, phone number and customer id and login credentials like username and password.

A customer can be a buyer or a seller. A buyer can place one or more orders wherein each order can include multiple items.

An order has the shipping address, date of order and bill amount

Each customer's transaction is maintained in a cart.

The sellers will have transaction in terms of items they sell. Each seller can sell more than 1 item but item can be brought by only 1 seller.

1.1.1Actors

The actor in the database here is a customer. A customer has been specialised as a buyer and a seller. The other actor is the admin of the e-commerce website.

1.1.2 Some Sample Queries

A customer can:

Check list of items belonging to a specific category.

Choose items and add them to cart

Place an order by entering required details.

View items by brand, price.

Sell items to the e-commerce admin

The admin can:

View list of buyers and sellers

Verify login details of buyers and sellers

Display all items selected by the buyer.

1.2 Project Objectives

The project aims at developing a MySQL database for an E-commerce website- Plethora. The website has a large amount of features involving the use of databases. Firstly, for the purpose of successful login and signup a database of customer details has been created. The Graphical user interface for the website includes a grid of products arranged category wise. This grid changes according to the user's specifications such as specificity of brand and price. For this very purpose, different MySQL queries have been used. Once a user selects items they are placed into the particular user's cart, furthermore an order is generated and inserted into the database. The website also has an additional feature of a recommendation system. The system queries items from the database based on the item based collaborative filter algorithm.

1.3 Project Scope

The database has been built keeping in mind the theoretical concepts of Database Management Systems such as Conceptual, Physical, Relational Schema, Normalisation and Global Schema. The benefits of creating triggers, stored procedures and views have been utilised. Several access methods such as indexing have been used for faster computation. The querying costs have been analysed and minimised to a large extent.

CHAPTER 2

SYSTEM DESIGN

2.1 Logical Design

The EER diagram for the database system is as follows:

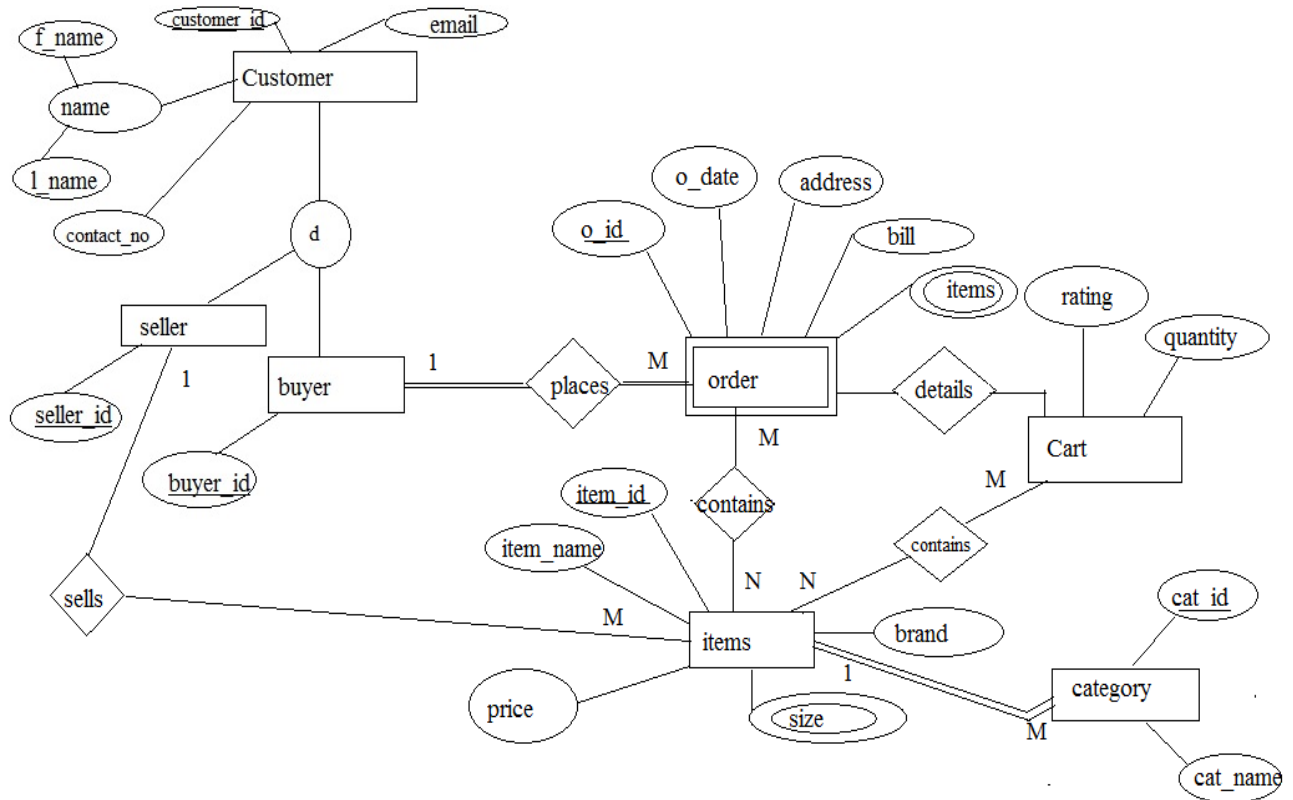


Figure2.1: Logical Design: EER Diagram for the E-commerce database.

2.2 Schema Diagram

The transformation from the entity-relationship model to the relational model is very straightforward. A feasible set of relational schema is as follows.

Customer

<u>customer_id</u>	Name	Contact	Email
--------------------	------	---------	-------

Login :

login_id	Password	<u>Uname</u>
----------	----------	--------------

Buyer:

<u>buyer_id</u>	<u>o_id</u>	o_date	address
-----------------	-------------	--------	---------

Seller :

<u>seller_id</u>	item_id
------------------	---------

Cart:

<u>order_id</u>	item_id
-----------------	---------

Item:

<u>item_id</u>	item_name	Brand	Picture	cat_id	Price	Discount	size
----------------	-----------	-------	---------	--------	-------	----------	------

Category:

<u>category_id</u>	category_name
--------------------	---------------

2.3 Advanced Logical Design

2.3.1 Normalization Techniques

In simple words Normalization is a systematic way of ensuring that a database structure is suitable for general-purpose querying and free of certain undesirable characteristics—insertion, update, and deletion anomalies—that could lead to a loss of data integrity.

Normal forms in a database or the concept of Normalization makes a Relation or Table free from insert/update/delete anomalies and saves space by removing duplicate data.

According to E. F. Codd the objectives of normalization were stated as follows:

1. To free the collection of relations from undesirable insertion, update and deletion dependencies.
2. To reduce the need for restructuring the collection of relations as new types of data are introduced, and thus increase the life span of application programs.
3. To make the relational model more informative to users.
4. To make the collection of relations neutral to the query statistics, where these statistics are liable to change as time goes by.

Example: A single table that stores Employee and Department details, thus:

1. On inserting a detail of an Employee then his department detail will also be entered for every employee record, thus departments details will be repeated with multiple records, thus storing duplicate data or Departments.
2. While updating a department detail you have to update the same department for various employees, which may lead to inconsistent state if any record is left while updating or on any error.
3. If a department is closed, then deleting department record will also delete the Employee records, thus missing records.

The process of normalization makes this Employee Department table to decompose or split into 2 or more tables and linked them by Foreign Keys, thus eliminating duplicate records, data redundancy and making data/records consistent across all relations/tables.

1st NF talks about atomic values and non-repeating groups.

2nd NF enforces that a non-Key attribute should belong to entire Key attribute.

3rd NF makes sure that there should be no transitive dependency between a non-Key and a Key attribute.

A functional dependency is a kind of Integrity Constraint that generalizes the concept of a key.

- Let R be the Relation Schema
- Let X and Y be the non-empty sets of attributes in R
- We say an instance r of R [r(R)] satisfies the FD $X \rightarrow Y$ if the following holds for every pair of Tuples t1, t2 in r
IF t1[X] = t2[X] THEN
t1[Y] = t2[Y]
END IF

1. Trivial Functional Dependency (FD): What is got on RHS is already LHS

Example: A determines itself.

$A \rightarrow A$

$A \rightarrow AB$

$AB \rightarrow A$

In general form to represent this $X \supseteq Y$. However, there are no useful operations using this kind of dependency.

2. Non-trivial FD: given a value of an attribute get a unique value.

Example:

$A \rightarrow B$

$A \rightarrow BC$

$AB \rightarrow CD$

In general, the rule is $X \cap Y = \varnothing$.

3. Semi non-trivial FD: it is not deriving completely new information. It partially brings a record, that is, not the value of entire set of attributes.

Example:

$AB \rightarrow BC$

B is common on both sides

$ACD \rightarrow EFCD$

CD is common both sides

It is generally represented as $X \cap Y \neq \varnothing$.

If an attribute functionally determines all the other attributes fully in a Relation, then the attribute is a Candidate Key.

- FD is generalization – Generalization of keys

- Superkey – Is a Key which contains a Candidate Key

1st Normal Form:

A relation is said to be in first normal form then it should satisfy the following

- ✓ No multi-valued attribute
- ✓ No composite attribute
- ✓ identify primary key

Here, the relationship is converted to either relation or foreign key or merging relations.

Foreign key: giving primary key of one table as a reference to another table.

Customer

<u>customer_id</u>	f_name	l_name	contact_no	email
--------------------	--------	--------	------------	-------

Login

login_id	Password	<u>uname</u>
----------	----------	--------------

Buyer

<u>buyer_id</u>	<u>order_id</u>
-----------------	-----------------

Seller

<u>seller_id</u>	<u>Sitemid</u>
------------------	----------------

Order

<u>o_id</u>	o_date	Address	bill_amt
-------------	--------	---------	----------

Cart

<u>Orderid</u>	Itemid
----------------	--------

Item

<u>item_id</u>	item_name	<u>Brand</u>	price	icat_id	discount	picture
----------------	-----------	--------------	-------	---------	----------	---------

Size

<u>sitem_id</u>	<u>Size</u>
-----------------	-------------

Category

<u>category_id</u>	category_name
--------------------	---------------

Outcome of 1st normalization:

1. Primary key has been identified in each table using closure property (minimal super key)
2. Composite attributes has been resolved
3. Multi-valued attributes has been resolved

2nd Normal Form:

A relation is said to be in second normal form then it should satisfy the following

- ✓ Repeating column values are taken out and maintained in a separate table. So that change can be done only once in the new table rather than all entries in the first table. Rule is foreign key must be on the N side else again multi-value in a column will occur.
- ✓ Identify **prime attribute** (part of candidate key that determines anything else), it is also called **partial dependency**, and eliminate it. Because, 2nd NF is based on **Full Functional dependency** (key should determine all other attributes in a table)
- ✓ Use foreign key on many side.

Customer

<u>customer_id</u>	f_name	l_name	contact_no	Email
--------------------	--------	--------	------------	-------

Login

login_id	Password	<u>Uname</u>
----------	----------	--------------

Buyer

<u>buyer_id</u>	<u>order_id</u>
-----------------	-----------------

Seller

<u>seller_id</u>	<u>Sitemid</u>
------------------	----------------

Order

<u>o_id</u>	o_date	Address	bill_amt
-------------	--------	---------	----------

Cart

<u>Orderid</u>	Itemid
----------------	--------

Item

<u>item_id</u>	item_name	<u>brand_name</u>	Price	picture	cat_id
----------------	-----------	-------------------	-------	---------	--------

Size

<u>sitem_id</u>	<u>Size</u>
-----------------	-------------

Category

<u>category_id</u>	category_name
--------------------	---------------

Brand

<u>brand_name</u>	Discount
-------------------	----------

Outcomes of 2nd Normalization

1. Repeating column values are taken out and maintained in a separate table. So that change can be done only once in the new table rather than all entries in the first table. Rule is foreign key must be on the N side else again multi-value in a column will occur.
2. Fully functional dependency
3. Use foreign key on many side

3rd Normal Form:

- ✓ Only columns with direct dependency of the primary key shall be in the entity.
- ✓ No transitive dependencies: non-prime attributes transitively depending on the key.

Example: $A \rightarrow B \rightarrow C \implies A \rightarrow C$.

$A \rightarrow B$ B is non-key attribute here

$B \rightarrow C$ suddenly becomes key attribute here. because of this, we will get repeated values in a column. Therefore, it should be eliminated.

- ✓ 3rd NF should hold the condition that: if $X \rightarrow Y$ then Either X is a super key

Or Y is a prime attribute

Following this condition will never allow transitive dependency.

Customer

<u>customer_id</u>	f_name	l_name	contact_no	email
--------------------	--------	--------	------------	-------

Login

login_id	Password	<u>uname</u>
----------	----------	--------------

Buyer

<u>buyer_id</u>	<u>order_id</u>
-----------------	-----------------

Seller

<u>seller_id</u>	<u>Sitemid</u>
------------------	----------------

Order

o_id	o_date	address
------	--------	---------

Cart

Orderid	Itemid	Quantity	rating
---------	--------	----------	--------

Item

<u>item_id</u>	item_name	<u>brand_name</u>	Price	picture	cat_id
----------------	-----------	-------------------	-------	---------	--------

Size

<u>sitem_id</u>	<u>Size</u>
-----------------	-------------

Category

<u>category_id</u>	category_name
--------------------	---------------

Brand

<u>brand_name</u>	Discount
-------------------	----------

2.3.2. Global Schema**Customer:**

Attribute Name	Attribute Size
customer_id	Int(10)
f_name	Varchar(50)
l_name	Varchar(50)
contact_no	Int(10)
Email	Varchar(50)

Login

Attribute Name	Attribute Value
Login	int(10)
Password	Varchr(50)
Uname	varchar(50)

Buyer

Attribute Name	Attribute Size
buyer_id	Int (10)
order_id	Int (10)

Seller

Attribute Name	Attribute Size
seller_id	Int(10)
sitem_id	int(10)

Order

Attribute Name	Attribute Size
O_id	Int(10)
O_date	Date
Address	varchar(50)

Cart

Attribute Name	Attribute Size
Ordered	int(10)
Itemid	Int(10)
Quantity	Int(10)
Rating	Int(10)

Item

Attribute Name	Attribute Size
item_id	int(10)
item_name	varchar(50)
Price	int(10)
brand_name	varchar(50)
Picture	varchar(50)
cat_id	int(10)

Size

Attribute Name	Attribute Size
Sitemid	int(10)
Size	varchar(50)

Brand

Attribute Name	Attribute Size
Brandname	varchar(50)
Discount	int(10)

Category

Attribute Name	Attribute Size
category_id	int(10)
category_name	varchar(50)

2.4 Queries

2.4.1 Creation of Tables

```
CREATE TABLE `Brand` (  
  `brand_name` varchar(50) NOT NULL,  
  `discount` float NOT NULL  
)
```

```
CREATE TABLE IF NOT EXISTS `Buyer` (  
  `buyer_id` int(10) NOT NULL,  
  `order_id` int(10) NOT NULL  
)
```

```
CREATE TABLE `Cart` (  
  `orderid` int(10) NOT NULL,  
  `itemid` int(10) NOT NULL,  
  `rating` int(10) NOT NULL,  
  `quantity` int(10) NOT NULL  
)
```

```
CREATE TABLE `Category` (  
  `category_id` int(10) NOT NULL,  
  `category_name` varchar(100) NOT NULL  
)
```

```
CREATE TABLE `Customer` (  
  `customer_id` int(10) NOT NULL,  
  `f_name` varchar(30) NOT NULL,  
  `l_name` varchar(30) NOT NULL,  
  `contact_no` varchar(10) DEFAULT NULL,  
  `email` varchar(30) NOT NULL  
)
```



```
CREATE TABLE `Item` (  
  `item_id` int(10) NOT NULL,  
  `item_name` varchar(100) NOT NULL,  
  `cat_id` int(10) NOT NULL,  
  `price` int(10) NOT NULL,  
  `picture` varchar(1000) DEFAULT NULL,  
  `ibrand_name` varchar(100) NOT NULL  
)
```

```
CREATE TABLE `Itemsize` (  
  `isitem_id` int(10) NOT NULL,  
  `size` varchar(10) NOT NULL  
)
```

```
CREATE TABLE `Login` (  
  `login_id` int(100) NOT NULL,  
  `password` varchar(20) NOT NULL,  
  `uname` varchar(20) NOT NULL  
)
```

```
CREATE TABLE `PlaceOrder` (  
  `o_id` int(10) NOT NULL,  
  `o_date` varchar(20) NOT NULL,  
  `address` varchar(50) NOT NULL  
)
```

```
CREATE TABLE `Seller` (  
  `seller_id` int(10) NOT NULL,  
  `sitem_id` int(10) NOT NULL  
)
```

```
CREATE TABLE `tempCart` (  
  `titem_id` int(10) NOT NULL,  
  `quantity` int(10) DEFAULT NULL,  
  `rating` int(10) NOT NULL)
```

Adding Primary Keys

```
ALTER TABLE `Brand`
```

```
ADD PRIMARY KEY (`brand_name`);
```

```
ALTER TABLE `Buyer`
```

```
ADD PRIMARY KEY (`buyer_id`,`order_id`), ADD KEY `order_id` (`order_id`);
```

```
ALTER TABLE `Cart`
```

```
ADD PRIMARY KEY (`orderid`,`itemid`), ADD KEY `itemid` (`itemid`);
```

```
ALTER TABLE `Category`
```

```
ADD PRIMARY KEY (`category_id`);
```

```
ALTER TABLE `Customer`
```

```
ADD PRIMARY KEY (`customer_id`);
```

```
ALTER TABLE `Item`
```

```
ADD PRIMARY KEY (`item_id`), ADD KEY `cat_id` (`cat_id`), ADD KEY `ibrand_id`  
(`ibrand_name`), ADD KEY `ibrand_name` (`ibrand_name`);
```

```
ALTER TABLE `Itemsize`
```

```
ADD PRIMARY KEY (`isitem_id`,`size`);
```

```
ALTER TABLE `Login`
```

```
ADD PRIMARY KEY (`uname`), ADD KEY `login_id` (`login_id`);
```

```
ALTER TABLE `PlaceOrder`
```

```
ADD PRIMARY KEY (`o_id`);
```

```
ALTER TABLE `Seller`
```

```
ADD PRIMARY KEY (`seller_id`,`sitem_id`), ADD KEY `sitem_id` (`sitem_id`);
```

```
ALTER TABLE `tempCart`
```

```
ADD PRIMARY KEY (`titem_id`);
```

Adding foreign keys

```
ALTER TABLE `Buyer`  
ADD CONSTRAINT FOREIGN KEY (`buyer_id`) REFERENCES `Customer` (`customer_id`),  
ADD CONSTRAINT FOREIGN KEY (`order_id`) REFERENCES `PlaceOrder` (`o_id`);
```

```
ALTER TABLE `Cart`  
ADD CONSTRAINT FOREIGN KEY (`itemid`) REFERENCES `Item` (`item_id`),  
ADD CONSTRAINT FOREIGN KEY (`orderid`) REFERENCES `PlaceOrder` (`o_id`);
```

```
ALTER TABLE `Item`  
ADD CONSTRAINT FOREIGN KEY (`cat_id`) REFERENCES `Category` (`category_id`);
```

```
ALTER TABLE `Itemsize`  
ADD CONSTRAINT FOREIGN KEY (`isitem_id`) REFERENCES `Item` (`item_id`);
```

```
ALTER TABLE `Login`  
ADD CONSTRAINT FOREIGN KEY (`login_id`) REFERENCES `Customer` (`customer_id`);
```

```
ALTER TABLE `Seller`  
ADD CONSTRAINT FOREIGN KEY (`seller_id`) REFERENCES `Customer` (`customer_id`),  
ADD CONSTRAINT FOREIGN KEY (`sitem_id`) REFERENCES `Item` (`item_id`);
```

Inserting into tables

```
INSERT INTO `Seller` (`seller_id`, `sitem_id`) VALUES  
(10, 200),  
(500, 600),  
(102, 780),  
(103, 1200);
```

```
INSERT INTO `PlaceOrder` (`o_id`, `o_date`, `address`) VALUES  
(1, 'abcd', 'pqrs'),  
(2, 'abcd', 'pqrs');
```

```
INSERT INTO `Login` (`login_id`, `password`, `uname`) VALUES
(87, 'abcd', 'uname_abcd'),
(246, 'divija', 'uname_divija'),
(100, 'mukta', 'uname_mukta'),
(300, 'pooja', 'uname_pooja');
```

```
INSERT INTO `Itemsize` (`isitem_id`, `size`) VALUES
(1, 'L'),
(1, 'M'),
(1, 'S'),
(2, 'L'),
(2, 'M'),
(2, 'S'),
(3, 'L');
```

```
INSERT INTO `Item` (`item_id`, `item_name`, `cat_id`, `price`, `picture`, `ibrand_name`)
VALUES (1, 'Pannkh Black Printed Top', 1, 399,
'http://assets.myntassets.com/h_240,q_95,w_180/v1/assets/images/982055/2015/9/11/11441963966
117-Pannkh-Women-Tops-6221441963965791-1_mini.jpg', 'Pannkh'),
(2, 'Miss Chase White & Black Striped Top', 1, 599,
'http://assets.myntassets.com/h_240,q_95,w_180/v1/assets/images/1341217/2016/5/6/11462525103
439-Miss-Chase-Women-Tops-7561462525103141-1_mini.jpg', 'Miss Chase'),
(3, 'La Firangi Green Printed Layered A-Line Kurta', 1, 479,
'http://assets.myntassets.com/h_240,q_95,w_180/v1/assets/images/1220100/2016/2/16/1145560180
5448-La-Firangi-Green-Printed-Layered-A-Line-Kurta-3811455601805122-1_mini.jpg', 'La
Firangi'),
(4, 'Belle Fille Maroon Top', 1, 499,
'http://assets.myntassets.com/h_240,q_95,w_180/v1/assets/images/1297897/2016/4/11/1146036440
2759-Belle-Fille-Maroon-Top-5161460364402437-1_mini.jpg', 'Belle Fille'),
(5, 'AKS Red Printed Anarkali Kurta', 1, 764,
'http://assets.myntassets.com/h_240,q_95,w_180/v1/assets/images/1410489/2016/7/6/11467807016
293-AKS-Red-Printed-Anarkali-Kurta-1501467807016047-1_mini.jpg', 'AKS');
```

```
INSERT INTO `Customer` (`customer_id`, `f_name`, `l_name`, `contact_no`, `email`) VALUES
(49, 'Mukta', 'Kulkarni', '7829584877', 'mukta3396@gmail.com'),
(50, 'Divija', 'Nagaraju', '9964297074', 'divijanagraju@gmail.com'),
(51, 'Mounisha', 'R', '9916613444', 'mounishar121@gmail.com'),
(52, 'Dhiraj', 'Bhakta', '7022209370', 'dhirajbhakta110@gmail.com'),
(53, 'Aditya', 'Kumar', '9902769411', 'adds.96@gmail.com'),
(54, 'Vedant', 'Patel', '7406170034', 'vedant.patel031096@gmail.com'),
(56, 'Sandhya', 's', '', 'sandhyaskarla14@gmail.com'),
(57, 'Neha', 'B', '9964303480', 'neharao.nbr@gmail.com');
```

```
INSERT INTO `Category` (`category_id`, `category_name`) VALUES
(1, 'Women-Indianwear'),
(2, 'Women-Westernwear'),
(3, 'Women-Accessories'),
(4, 'Women-Beauty-Grooming'),
(5, 'Women-FootWear'),
(6, 'Women-Watches');
```

```
INSERT INTO `Cart` (`orderid`, `itemid`, `rating`, `quantity`) VALUES
(18, 1511, 3, 2),
(18, 1512, 4, 3),
(19, 1514, 4, 3),
(19, 1515, 4, 2),
(20, 97, 5, 1),
(21, 193, 5, 2),
(22, 1, 4, 1);
```

```
INSERT INTO `Buyer` (`buyer_id`, `order_id`) VALUES
(971, 2),
(980, 13),
(969, 14),
(968, 15);
```

```
INSERT INTO `Brand` (`brand_name`, `discount`) VALUES
('20Dresses', 50),
('2go ACTIVE GEAR USA', 25),
('612 league', 5),
('Adidas', 20);
```

2.4.2 Creation of Views

```
CREATE view CustomerDetails as Select f_name, l_name, email from Customer;
```

```
CREATE view ItemView as Select * from Item, Cart where Item.item_id=Cart.itemid;
```

2.4.3 Creation of Procedures

Stored Procedures:

```
DELIMITER //
CREATE PROCEDURE pname()
BEGIN
    SELECT * FROM Customer;
END //
DELIMITER ;
```

```
Call pname();
```

2.4.4 Creation of Triggers

```
CREATE TRIGGER `Trigger2` AFTER INSERT ON `Item`
FOR EACH ROW INSERT INTO Seller VALUES((SELECT customer_id FROM Customer order
by customer_id DESC LIMIT 1),(SELECT item_id FROM Item order by item_id DESC LIMIT 1))
```

```
CREATE TRIGGER `Trigger1` AFTER INSERT ON `Cart`
FOR EACH ROW DELETE from tempCart
```

CHAPTER 3

PHYSICAL DESIGN

3.1 Assumptions

Number of tuples in each relation

Customer	943
Login	943
Buyer	500
Seller	443
Order	5
Cart	8000
Brand	100
Item	1700
Category	18
Size	1700
Cart	5

3.2 Storage Requirements: Disk Parameters

Avg Seek Time

Rotational Delay (Latency time)

Block Transfer Time

Block pointer size

Block Size

Following are the assumptions which are considered for storage requirements:

- Fixed length records are considered for all relations.
- The delimiter for each field is length of the field
- Total number of records in respective relations (provided in below table).
- Block size is 1024 bytes.
- Record doesn't span over multiple blocks (this can be achieved by taking floor function during calculating number of records per block to restrict single record doesn't span over blocks).
- Block pointer(Bp) size is 4 bytes
- Average Seek Time(S) is 20 ms irrespective of any site.

- Average Disk rotation time (Latency) Time (L) is 10 ms irrespective of any site.
- Block transfer rate (Tr) is 0.5 ms irrespective of any site.
- **Blocking factor= ceil(Block size / Record size in bytes)**
- **# no of blocks = ceil(# of records/ Blocking factor)**

Relation	# of records	Record size in bytes	Blocking factor	# no. of blocks
Customer	943	100	11	86
Login	943	50	21	45
Buyer	500	20	52	10
Seller	5	20	52	1
Order	5	80	13	1
Cart	5	40	26	1
Brand	100	58	18	6
Item	1652	730	2	850
Category	18	60	18	1
Size	1652	20	52	33

3.2.1 Spanned and Unspanned Records

- 1)customer - Unspanned
- 2)login - Unspanned
- 3)buyer - Unspanned
- 4)seller - Unspanned
- 5)order - Unspanned
- 6)cart - Unspanned
- 7)Brand- Unspanned
- 8)Item- Spanned
- 9)category - Unspanned
- 10)size - Unspanned

Spanned means occupying more than one physical block.

Un-spanned means occupying only one physical block

Space required to store each relation

SELECT Table_NAME "Table_Name",

-> data_length "table data_length in Bytes",

-> index_length "table index_length in Bytes",

-> data_free "Free Space in Bytes"

-> FROM information_schema.TABLES where Table_schema=database_name;

Example:

```
SELECT Table_NAME "Customer",data_length "table data_length in Bytes",index_length "table  
index_length in Bytes",data_free "Free Space in Bytes" FROM information_schema.TABLES  
where Table_schema='plethora';
```

3.3 Access Methods

Considering the assumption we can calculate easily the size of single record (tuple) of every relation with the help of Global Schema. The above table gives the number of records in each relation, size of each record, blocking factor for a particular block of that relation and number of blocks required to store entire relation.

Having records on secondary storage, if you want to access them faster, then you need indexing. If a database is frequently queried and it is too large then it is supposed to have index to increase performance. There are various indexes used in databases. Here, we consider the following indexing scheme: Primary Index, Clustered Index and Secondary index. Based on the query, we decide what type of indexing file.

Relation	Indexing type	Indexing attribute(s)	Is a key?
Customer	primary	customer_id	Yes
Login	primary	login_id	Yes
Buyer	primary	buyer_id	Yes
Seller	NA	NA	No
Order	NA	NA	No
Cart	NA	NA	No
Brand	NA	NA	No
Item	primary	item_id	Yes
Category	NA	NA	No
Size	NA	NA	No

The following table explains what is the disk block access time to extract particular record for all the relations.

Relation	# of records	# no of data blocks	Index size per record	# of index records per block	# no of index blocks	# no of block access without indexing	# no of block access with indexing
customer	943	86	4+4	128	8	86	4
login	943	45	4+4	128	8	45	4
buyer	500	10	4+4	128	4	10	3
seller	443	9	NA	NA	NA	9	1
order	5	1	NA	NA	NA	1	1
cart	5	1	NA	NA	NA	1	1
Brand	100	6	NA	NA	NA	6	1
Item	1653	850	4+4	128	13	850	5
category	18	1	NA	NA	NA	1	1
Size	1653	33	NA	NA	NA	33	1

of index records per block = Block size / Index size per record

no of index blocks = ceil(# of records / # of index records per block)

no of block access without indexing = # no of data blocks

number of block accesses with indexing = ceil [log(# no of index blocks)] + 1

Indexing the data file definitely reduces the number of block accesses needed to find a particular record from the data file. The complete statistics is showed in above table.

3.4 Timings

Disk access time = Avg seek time + latency time + block transfer time
 $= 20 + 10 + 0.5$
 $= 30.5 \text{ ms}$

Therefore, to access one random block and transfer it, the time is 30.5ms.If the blocks are consecutive seek time and latency time are not included.

Also, there can be overhead delay and queuing delay.

3.5 System Specification

Space required to store each relation

SELECT Table_NAME "Table_Name",
 -> data_length "table data_length in Bytes",
 -> index_length "table index_length in Bytes",

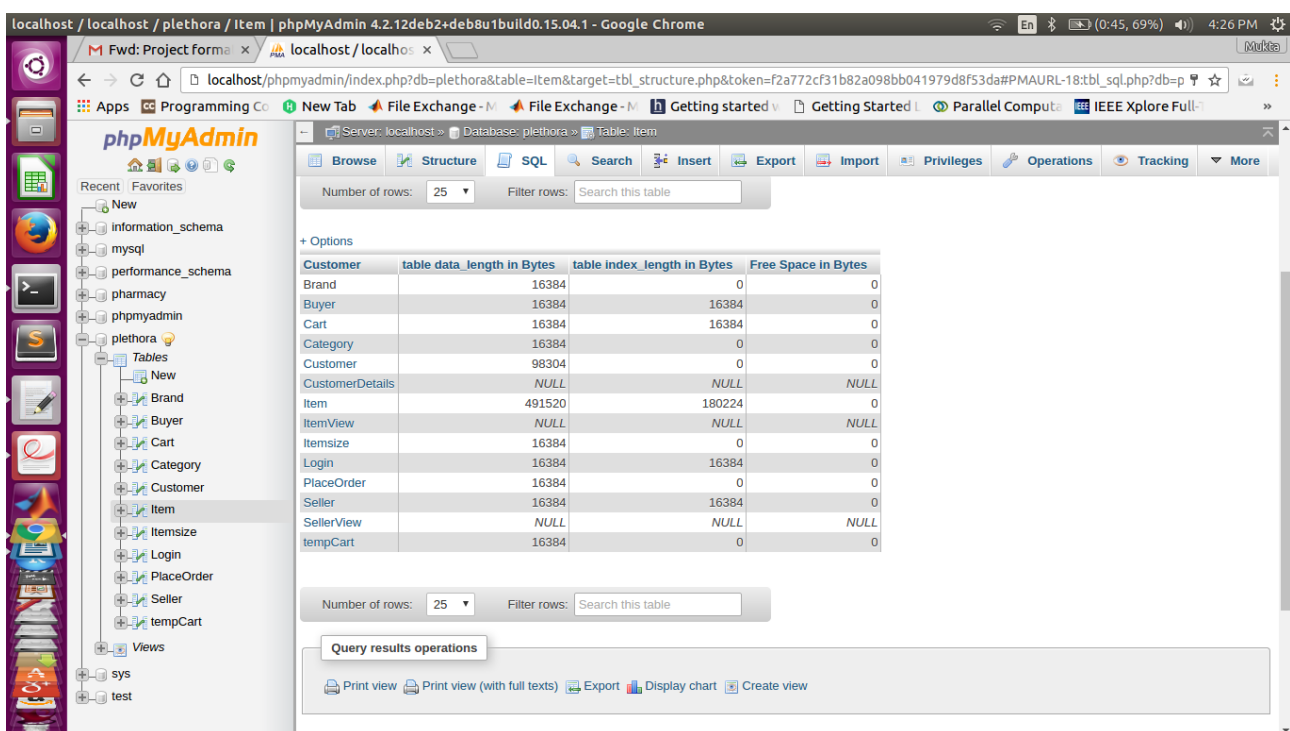
-> data_free "Free Space in Bytes"

-> FROM information_schema.TABLES where Table_schema=database_name

-> ;¹

Example:

SELECT Table_NAME "Customer",data_length "table data_length in Bytes",index_length "table index_length in Bytes",data_free "Free Space in Bytes" FROM information_schema.TABLES where Table_schema='plethora';



Customer	table data_length in Bytes	table index_length in Bytes	Free Space in Bytes
Brand	16384	0	0
Buyer	16384	16384	0
Cart	16384	16384	0
Category	16384	0	0
Customer	98304	0	0
CustomerDetails	NULL	NULL	NULL
Item	491520	180224	0
ItemView	NULL	NULL	NULL
Itemsize	16384	0	0
Login	16384	16384	0
PlaceOrder	16384	0	0
Seller	16384	16384	0
SellerView	NULL	NULL	NULL
tempCart	16384	0	0

Figure 3.1: System Specification for the database

3.6 Query Costs

Example: PLETHORA SYSTEM data

Assumptions: C- number of blocks in customer table = 86

L- number of blocks in login table = 45

B- number of blocks in the buyer table = 10

I- number of blocks in item table = 850

1. Simple Query

SELECT * FROM Item;

1653 total, Query took 0.0003 seconds.

2. SELECT * from Customer c, Buyer b WHERE b.buyer_id=c.customer_id;

27 total, Query took 0.0003 seconds.

Block nested loop join:

for each block C of c do begin

for each block B of b do begin

for each tuple c in C do begin

for each tuple b in B do begin

Check if (c,b) satisfy the condition $b.buyer_id = c.customer_id$;

if they do add c.b to the result

end if

end

end

end

end

Worst case cost: $C + (C*B) = 86 + (86*10) = 946$.

Best case cost: $C + B = 86 + 10 = 96$

Index block nested loop join:

For each C record, cost of disk access for B is 1 for hash index and 3 on an average for BTree.

Cost: $C + (C * \text{Cost of search for B})$

3.SELECT * FROM Customer c, Buyer b, Login l WHERE l.login_id=c.customer_id AND c.customer_id=b.buyer_id;

Time taken: 0.0008 sec.

Block nested loop:

```
for each block L of l do begin
for each block C of c do begin
for each block B of b do begin
    for each tuple l in L do begin
    for each tuple c in C do begin
    for each tuple b in B do begin
        Check if (l,c,b) satisfy the condition
        if they do add l.c.b to the result
        end if
    end
    end
    end
end
end
end
end
```

Cost (Worst Case): $L + (L * C) + (L * B * C) = 45 + (45*86) + (45*10*86)$

I/Os: 42615

Cost (Best Case): $L + C + B = 45 + 86 + 10$

I/Os: 141

4. SELECT * from Customer c, Login l WHERE c.customer_id = l.login_id;

Time taken: 0.0003 sec.

Nested loop join:

```
for each tuple c in C do
    for each tuple l in L do
        if c.customer_id == l.login_id
            then add c.l into result
        end if
    end
end
```

Worst case cost: $nr * bs + br = \text{number of records(tuples) of } C * \text{number of blocks of } L + \text{number of blocks of } C$

Best case cost: $C + L = 86 + 86 = 176$

Index nested loop join:

For each record C, cost of disk access for L is 1 for **hash index** and about 3 for **BTree**.

Cost: $C + (C * \text{blocking factor} * \text{Cost of search in } L)$

Creating Indexes

SQL executed: **SELECT name FROM Item;**

Time taken Before Creating Index: 0.0003 sec.

Index Created: **CREATE index index1 ON Item(item_id);**

Time taken After Creating Index index1: 0.0002 sec

Indexing uses BTree : The index leaf nodes are stored in arbitrary order. Their position on the disk does not correspond to the logical position according to the indexing order. Therefore to find entry among the arbitrarily stored index leaf nodes we use 'Balanced tree'.

CHAPTER 4

IMPLEMENTATION AND RESULTS

4.1 Implementation

The database “plethora” was created using phpMyAdmin. CREATE operation was used to create new tables required for all the queries was developed. Large amount of data was inserted by web scraping using python- BeautifulSoup library. These data sets were imported as CSV files into phpMyAdmin. A connection with the website was made by programming in PHP which linked the database to the Graphical User Interface. From signup/login to adding items to Cart and placing orders, INSERT operation was performed. To delete an item from cart and to delete an order we performed DELETE operation and to use filters we put conditions in the WHERE clause. To get the last user who signed up and the last order made the LIMIT clause was used. The recommendation system algorithm is a python script which is linked with the PHP code and used to give item recommendations to users.

4.2 Results

This project implemented the concepts related to database schemas at the conceptual, relational and physical level. Normalisation concepts and global schema for the database is implemented successfully. The reduction of query cost is performed by indexing due to which faster results are obtained. The system requirements were successfully satisfied.

CONCLUSION

E-commerce is the fastest growing business to consumer online trade. This makes the study of the backend behind an e-commerce website important. The project implemented successfully explores into this field taking into consideration all database design related concept. The project also implements an additional feature of recommendation system which is a widely growing necessity in every e-commerce website.

REFERENCES

- [1] Swapna Kodali, "Design and Implementation of E-commerce Site", MS Thesis, Indiana University, South Bend-2007.
- [2] B.M. Sarwar, G. Karypis, J.A. Konstan, and J. Reidl. Item-based collaborative filtering recommendation algorithms. In Proceedings of the 10th International World Wide Web Conference, pages 285-295, 2001.