DBMS

Mini Project

Online Bookstore Database

Krunal Rank – U18CO081

CONTENT

Problem Statement

Books have been an important part of our lives. Selling books has also been a source of livelihood for many people.

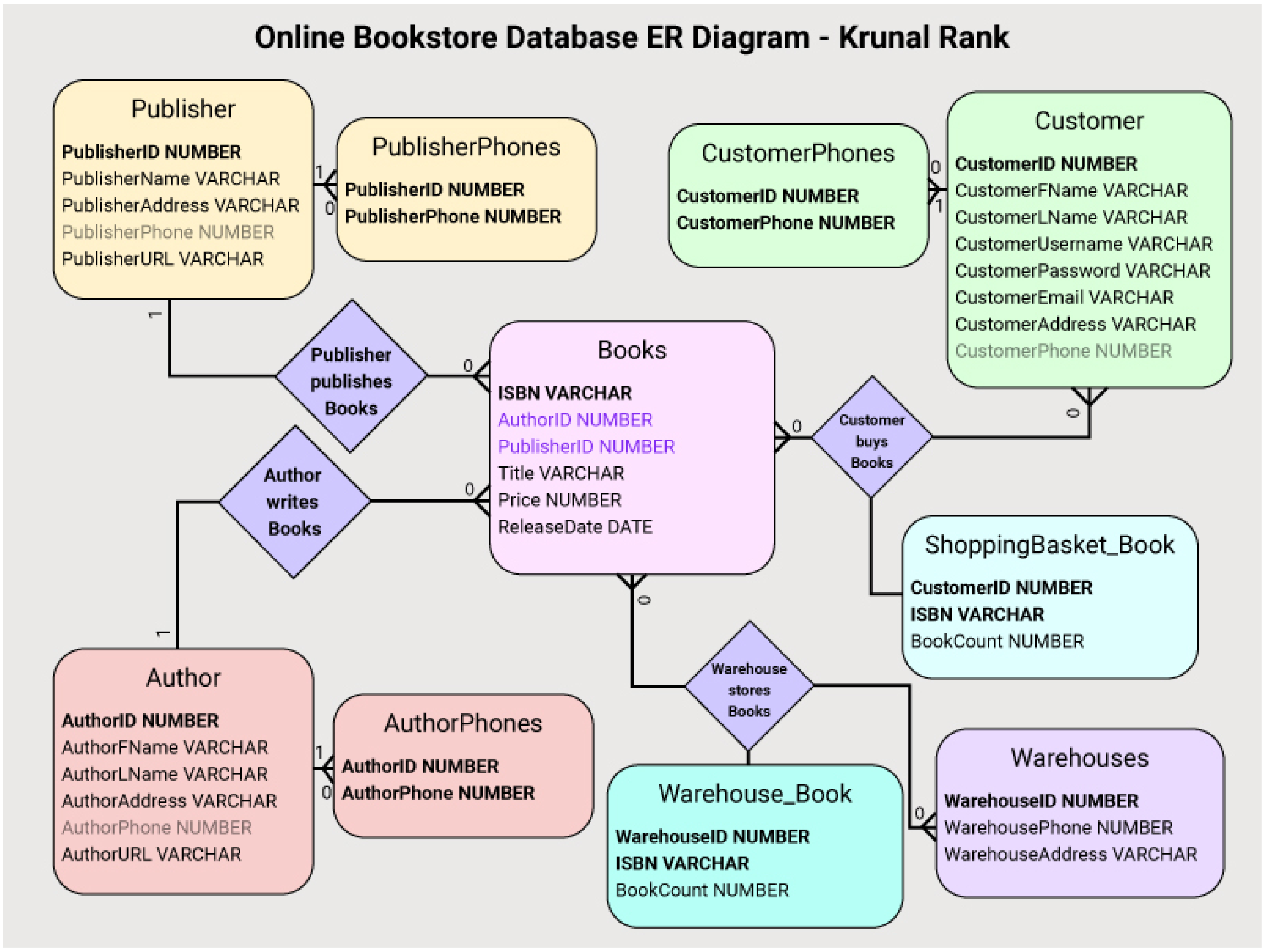
There are times when the business of selling books is set up online either under a big umbrella e-commerce organisation such as Amazon, Flipkart, e-Bay or Snapdeal, or having a self-sustained website such as BooksWagon, BuyBooksIndia, etc. Even Amazon, an e-commerce giant, started as an Online Bookstore.

In any case, a clean, scalable and secure database for managing the data regarding the books, their purchases and their inventory is an absolute necessity.

This mini project provides a simple solution to such a problem of managing information about Books and their purchases.

Entity Relationship (ER) Diagram

- ER Diagram is the first step of any Database Management System.
- It includes diagrammatic representation of several entities and their relationships with each other.
- Here, Entity refers to an object of significance such as, in our case, a Book, the person who purchased it, etc.
- The following diagram describes the Entity Relationships of a simple Online Bookstore Database:



Description of ER Diagram:

- Each Book has ISBN, Publisher Name, Author Name, Author Address, ReleaseDate, Title and Price.
- The Authors have Name, Address, their URL and possibly their Contact details.
- The Publishers have the same attributes as Authors.
- The Books need to be stored in Warehouses which are identified with a unique Code, and has attributes Phone and Address.
- Multiple Warehouses can have multiple books of same or different ISBN number.

- Now, the Books can be purchased by a person identified using an ID and has an Email Address, Name, Phone(s) and Address.
- Each Shopping basket can have multiple books of same or different ISBN number.
- Phones of all the entities are stored in different relations as they are multi valued attributes.
- As you can observe from the ER Diagram, the entities Customer, Author, Publisher and Warehouse are strong entities.
- Other entities such as Books, Warehouse_Book, ShoppingBasket_Book,AuthorPhones, CustomerPhones and PublisherPhones are weak entities.
- Note that Warehouse only has single phone number because warehouses usually have only primary contact.
- On the other hand, authors, publishers and customers can have more than one phone whose relationship is clearly shown in the diagram.

Note that this ER Diagram will be modified when normalised subsequently.

Relation Model

As we can see from the ER Diagram, we have distributed the entities in Strong entities and Weak entities. Also, viable keys have been marked using bold letters and cardinalities have been shown.

The following are the conversions of entities into Relations:-

Books

Attribute	Type	Constraint
ISBN	VARCHAR(255)	PRIMARY KEY
PublisherID	NUMBER(15)	FOREIGN KEY
AuthorID	NUMBER(15)	FOREIGN KEY
ReleaseDate	DATE	NOT NULL,DEFAULT
Title	VARCHAR(255)	NOT NULL
Price	NUMBER(10,2)	NOT NULL

Publishers

Attribute	Type	Constraint
PublisherID	NUMBER(15)	PRIMARY KEY/AUTO INCREMENT
PublisherName	VARCHAR(255)	NOT NULL
PublisherAddress	VARCHAR(255)	NOT NULL
PublisherURL	VARCHAR(255)	NOT NULL

PublisherPhones

Attribute	Type	Constraint
PublisherID	NUMBER(15)	FOREIGN KEY
PublisherPhone	NUMBER(14)	NOT NULL

In this table, the Primary key is formed by (PublisherID,PublisherPhone).

Authors

Attribute	Type	Constraint
AuthorID	NUMBER(15)	PRIMARY KEY/AUTO INCREMENT
AuthorFName	VARCHAR(255)	NOT NULL
AuthorLName	VARCHAR(255)	NOT NULL
AuthorAddress	VARCHAR(255)	NOT NULL
AuthorURL	VARCHAR(255)	NOT NULL

AuthorPhones

Attribute	Type	Constraint
AuthorID	NUMBER(15)	FOREIGN KEY
AuthorPhone	NUMBER(14)	NOT NULL

In this table, the Primary key is formed by (AuthorID,AuthorPhone).

Customers

Attribute	Type	Constraint
CustomerID	NUMBER(15)	PRIMARY KEY/AUTO INCREMENT
CustomerFName	VARCHAR(255)	NOT NULL
CustomerLName	VARCHAR(255)	NOT NULL
CustomerAddress	VARCHAR(255)	NOT NULL
CustomerEmail	VARCHAR(255)	NOT NULL/UNIQUE
CustomerUsername	VARCHAR(255)	NOT NULL/UNIQUE
CustomerPassword	VARCHAR(255)	NOT NULL

CustomerPhones

Attribute	Type	Constraint
CustomerID	NUMBER(15)	FOREIGN KEY
CustomerPhone	NUMBER(14)	NOT NULL

In this table, the Primary key is formed by (CustomerID,CustomerPhone).

Warehouses

Attribute	Type	Constraint
WarehouseID	NUMBER(15)	PRIMARY KEY/AUTO INCREMENT
WarehouseAddress	VARCHAR(255)	NOT NULL
WarehousePhone	NUMBER(14)	NOT NULL/UNIQUE

ShoppingBasket_Book

Attribute	Type	Constraint
CustomerID	NUMBER(15)	FOREIGN KEY
ISBN	VARCHAR(255)	FOREIGN KEY
BookCount	NUMBER(14)	NOT NULL

In this table, the Primary key is formed by (CustomerID,ISBN).

Warehouse_Book

Attribute	Type	Constraint
WarehouseID	NUMBER(15)	FOREIGN KEY
ISBN	VARCHAR(255)	FOREIGN KEY
BookCount	NUMBER(14)	NOT NULL

In this table, the Primary key is formed by (WarehouseID,ISBN).

As seen in the above Relation Models, some new attributes are also added so that Primary keys can be made more independent than data.

After formation of these Relation Models, we will look into the existing functional dependencies in these models and verify that the Relations are indeed formed into BCNF.

Normalisation of Relation Models

- All the attributes in given relation must be atomic (1 NF).
- There should be no partial dependencies (2 NF).
- There should be no transitive dependencies (3 NF).
- All the LHS of Functional dependencies must be Super Key.

Further there is 5th Normal Form which requires following conditions despite satisfying conditions for BCNF:-

- There should be no multivariate dependencies (4 NF).
- There should be no Join dependencies (5 NF).

However, 5th Normal Form is not as important as BCNF in practice.

The following Functional Dependencies have been observed from the above Relation Model:-

For Relation 'Book',

(ISBN) → (ISBN, Title, PublisherID, AuthorID, ReleaseDate, Price)

For Relation 'Authors',

(AuthorID) → (AuthorID, AuthorFName, AuthorLName, AuthorURL, AuthorAddress)

For Relation 'AuthorPhones',

(AuthorID, AuthorPhone) → (AuthorID, AuthorPhone)

For Relation 'Publishers',

(PublisherID) → (PublisherID, PublisherName, PublisherURL, PublisherAddress)

For Relation 'PublisherPhones',

(PublisherID, PublisherPhone) → (PublisherID, PublisherPhone)

For Relation ‘Customers’,

(CustomerID) → (CustomerID, CustomerFName, CustomerLName, CustomerEmail, CustomerAddress, CustomerUsername, CustomerPassword)

For Relation ‘CustomerPhones’,

(CustomerID, CustomerPhone) → (CustomerID, CustomerPhone)

For Relation ‘Warehouses’,

(WarehouseID) → (WarehouseID, WarehousePhone, WarehouseAddress)

For Relation ‘ShoppingBasket_Book’,

(CustomerID, ISBN) → (CustomerID, ISBN, BookCount)

For Relation ‘Warehouse_Book,

(WarehouseID, ISBN) → (WarehouseID, ISBN, BookCount)

Summary regarding Normalisation:

From the above FDs we find that the existing relation is already in BCNF.

Even Phone Numbers, as they are unique, can be used as Candidate Keys but we must avoid it so that there is no discrepancy later.

Since I have not used any multivariate FDs or JDs I did not face any problems regarding 4th Normal Form or 5th Normal Form.

Note that the above FDs are not written in one go by me. Lots of decompositions of tables were done in order to prepare the final model.

The process of decomposition can be described as removing certain attributes and making new relations so that the cardinalities also satisfy and Normal Form conditions are also satisfied.

Implementation of Relation Models in SQL

As seen from the Relation Models section, all the constraints have been properly mentioned.

Most of the Constraints such as PRIMARY KEY, FOREIGN KEY, NOT NULL, UNIQUE and DEFAULT have been included.

Certain constraints such as ENUM and SET have not been used because they are not needed in current implementation due to the Relations being in BCNF.

The Following SQL Code correctly Implements the Relations along with certain Joins and Views:-

TABLE CREATION

```
CREATE TABLE Authors(

    AuthorID NUMBER(15) GENERATED ALWAYS AS IDENTITY (START WITH 10000000001
INCREMENT BY 1) NOT NULL,

    AuthorFName VARCHAR(255) NOT NULL,

    AuthorLName VARCHAR(255) NOT NULL,

    AuthorAddress VARCHAR(255) NOT NULL,

    AuthorURL VARCHAR(255) NOT NULL,

    PRIMARY KEY(AuthorID)

);



CREATE TABLE AuthorPhones(

    AuthorID NUMBER(15) NOT NULL,

    AuthorPhone NUMBER(14) NOT NULL,

    PRIMARY KEY(AuthorID, AuthorPhone),

    FOREIGN KEY(AuthorID) REFERENCES Authors(AuthorID) ON DELETE CASCADE

);


CREATE TABLE Publishers(

    PublisherID NUMBER(15) GENERATED ALWAYS AS IDENTITY (START WITH 20000000001
INCREMENT BY 1),

    PublisherName VARCHAR(255) NOT NULL,

    PublisherAddress VARCHAR(255) NOT NULL,
    PublisherURL VARCHAR(255) NOT NULL,
    PRIMARY KEY(PublisherID)
);

CREATE TABLE PublisherPhones(
    PublisherID NUMBER(15) NOT NULL,
    PublisherPhone NUMBER(14) NOT NULL,
    PRIMARY KEY(PublisherID, PublisherPhone),
    FOREIGN KEY(PublisherID) REFERENCES Publishers(PublisherID)  ON DELETE
CASCADE
);

CREATE TABLE Customers(
    CustomerID NUMBER(15) GENERATED ALWAYS AS IDENTITY (START WITH 30000000001
INCREMENT BY 1),
    CustomerFName VARCHAR(255) NOT NULL,
    CustomerLName VARCHAR(255) NOT NULL,
    CustomerAddress VARCHAR(255) NOT NULL,
    CustomerEmail VARCHAR(255) NOT NULL,
    CustomerUsername VARCHAR(255) NOT NULL UNIQUE,
    CustomerPassword VARCHAR(255) NOT NULL,
    PRIMARY KEY(CustomerID)
);

CREATE TABLE CustomerPhones(
    CustomerID NUMBER(15) NOT NULL,
    CustomerPhone NUMBER(14) NOT NULL,
    PRIMARY KEY(CustomerID, CustomerPhone),
    FOREIGN KEY(CustomerID) REFERENCES Customers(CustomerID) ON DELETE CASCADE
);

CREATE TABLE Warehouses(
    WarehouseID NUMBER(15) GENERATED ALWAYS AS IDENTITY (START WITH 40000000001
INCREMENT BY 1) NOT NULL,

    WarehousePhone NUMBER(14) NOT NULL UNIQUE,

    WarehouseAddress VARCHAR(255) NOT NULL,

    PRIMARY KEY(WarehouseID)

);


CREATE TABLE Books(

    ISBN VARCHAR(255) PRIMARY KEY,

    PublisherID NUMBER(15) NOT NULL,

    AuthorID NUMBER(15) NOT NULL,

    ReleaseDate DATE DEFAULT CURRENT_DATE,

    Title VARCHAR(255) NOT NULL,

    Price NUMBER(10,2) NOT NULL,

    FOREIGN KEY(AuthorID) REFERENCES Authors(AuthorID)  ON DELETE CASCADE,

    FOREIGN KEY(PublisherID) REFERENCES Publishers(PublisherID)  ON DELETE
CASCADE

);



CREATE TABLE ShoppingBasket_Book(

    CustomerID NUMBER(15) NOT NULL,

    ISBN VARCHAR(255) NOT NULL,

    BookCount NUMBER(15) NOT NULL,

    PRIMARY KEY(CustomerID,ISBN),

    FOREIGN KEY(CustomerID) REFERENCES Customers(CustomerID) ON DELETE CASCADE,

    FOREIGN KEY(ISBN) REFERENCES Books(ISBN) ON DELETE CASCADE

);


CREATE TABLE Warehouse_Book(

    WarehouseID NUMBER(15) NOT NULL,

    ISBN VARCHAR(255) NOT NULL,
    BookCount NUMBER(15) NOT NULL,

    PRIMARY KEY(WarehouseID,ISBN),

    FOREIGN KEY(WarehouseID) REFERENCES Warehouses(WarehouseID) ON DELETE
CASCADE,

    FOREIGN KEY(ISBN) REFERENCES Books(ISBN) ON DELETE CASCADE

);
```

POPULATING TABLES WITH CORRECT DATA

```
INSERT INTO Authors(AuthorFName,AuthorLName,AuthorAddress,AuthorURL) VALUES
('Joanne','Rowling','557 Broadway New York','www.jkrowling.com');

INSERT INTO Authors(AuthorFName,AuthorLName,AuthorAddress,AuthorURL) VALUES
('Mark', 'Twain','351 Farmington Avenue','www.marktwainproject.com');

INSERT INTO Authors(AuthorFName,AuthorLName,AuthorAddress,AuthorURL) VALUES
('Rick', 'Riordan','125 West End Avenue','www.rickriordan.com');

INSERT INTO Authors(AuthorFName,AuthorLName,AuthorAddress,AuthorURL) VALUES
('Enid', 'Blyton','83 Shortlands Road, Shortlands','www.enidblyton.net');

INSERT INTO Authors(AuthorFName,AuthorLName,AuthorAddress,AuthorURL) VALUES
('Arundhati', 'Roy','55 LaxmanNagar New Delhi','www.weroy.com');

SELECT * FROM Authors;


INSERT INTO AuthorPhones VALUES(10000000001,2884713658);

INSERT INTO AuthorPhones VALUES(10000000002,7865412019);

INSERT INTO AuthorPhones VALUES(10000000002,2881069845);

INSERT INTO AuthorPhones VALUES(10000000004,1100548762);

INSERT INTO AuthorPhones VALUES(10000000004,7842361220);

INSERT INTO AuthorPhones VALUES(10000000004,1100745288);

INSERT INTO AuthorPhones VALUES(10000000005,7063254112);

SELECT * FROM AuthorPhones;


INSERT INTO Publishers(PublisherName,PublisherAddress,PublisherURL)
VALUES('IndianInk','India','www.indianink.com');

INSERT INTO Publishers(PublisherName,PublisherAddress,PublisherURL)
VALUES('Bloomsburry Publishing','UK','www.bloomsburry.com');

INSERT INTO Publishers(PublisherName,PublisherAddress,PublisherURL)
VALUES('Miramax Books','US','www.miramax.com');

INSERT INTO Publishers(PublisherName,PublisherAddress,PublisherURL)
VALUES('Disney','Hollywood, LA','www.disney.com');

INSERT INTO Publishers(PublisherName,PublisherAddress,PublisherURL) VALUES('S
Chand','Mumbai, India','www.schandpublishing.com');

INSERT INTO Publishers(PublisherName,PublisherAddress,PublisherURL)
VALUES('Hachette','UK','www.hachette.com');

SELECT * FROM Publishers;
```

```
INSERT INTO PublisherPhones VALUES(20000000001,2881076632);

INSERT INTO PublisherPhones VALUES(20000000001,2881036547);

INSERT INTO PublisherPhones VALUES(20000000002,1124533125);

INSERT INTO PublisherPhones VALUES(20000000003,3665544712);

INSERT INTO PublisherPhones VALUES(20000000004,1004569774);

INSERT INTO PublisherPhones VALUES(20000000005,6644722331);

INSERT INTO PublisherPhones VALUES(20000000005,3655110124);

INSERT INTO PublisherPhones VALUES(20000000006,4975233110);

SELECT * FROM PublisherPhones;


INSERT INTO
Customers(CustomerFName,CustomerLName,CustomerAddress,CustomerEmail,CustomerUse
rname,CustomerPassword)

    VALUES('Krunal','Rank','Jamnagar,
India','krunalrnak@gmail.com','krunalrank123','K@runal0609');

INSERT INTO
Customers(CustomerFName,CustomerLName,CustomerAddress,CustomerEmail,CustomerUse
rname,CustomerPassword)

    VALUES('Tanmay','Shah','Jamnagar,
India','tanmayshah29@gmail.com','tanmay2901','ultimate@360');

INSERT INTO
Customers(CustomerFName,CustomerLName,CustomerAddress,CustomerEmail,CustomerUse
rname,CustomerPassword)

    VALUES('Hardev','Khandhar','Jamnagar,
India','hardevkhandhar71@gmail.com','hardevk','Hardev@Kh');

INSERT INTO
Customers(CustomerFName,CustomerLName,CustomerAddress,CustomerEmail,CustomerUse
rname,CustomerPassword)

    VALUES('Neil','Pandya','Ahmedabad,
India','pandyaneil08@gmail.com','neilP','neilPandya@0710');

INSERT INTO
Customers(CustomerFName,CustomerLName,CustomerAddress,CustomerEmail,CustomerUse
rname,CustomerPassword)

    VALUES('Mihir','Khambhati','Ankleshwar,
Gujarat','mihirk1313@gmail.com','mihir1313','KhambhatiM@1313');
```

```
INSERT INTO
Customers(CustomerFName,CustomerLName,CustomerAddress,CustomerEmail,CustomerUse
rname,CustomerPassword)

    VALUES('Miley','Joshi','Texas,
US','mileyj@hotmail.com','MileyJ','MileyJoshi@Texas');

SELECT * FROM Customers;


INSERT INTO CustomerPhones VALUES(30000000001,9979891142);

INSERT INTO CustomerPhones VALUES(30000000001,7016507648);

INSERT INTO CustomerPhones VALUES(30000000002,6355487754);

INSERT INTO CustomerPhones VALUES(30000000003,7444689954);

INSERT INTO CustomerPhones VALUES(30000000004,3667844566);

INSERT INTO CustomerPhones VALUES(30000000005,9999633145);

INSERT INTO CustomerPhones VALUES(30000000006,4477233145);

SELECT * FROM CustomerPhones;


INSERT INTO Warehouses(WarehousePhone,WarehouseAddress)
VALUES(2886423114,'Mumbai, India');

INSERT INTO Warehouses(WarehousePhone,WarehouseAddress)
VALUES(2369984512,'Delhi, India');

INSERT INTO Warehouses(WarehousePhone,WarehouseAddress)
VALUES(1014662133,'Texas, US');

INSERT INTO Warehouses(WarehousePhone,WarehouseAddress) VALUES(1001355645,'New
York, US');

INSERT INTO Warehouses(WarehousePhone,WarehouseAddress)
VALUES(5423300112,'Beijing, China');

INSERT INTO Warehouses(WarehousePhone,WarehouseAddress)
VALUES(4231100148,'London, UK');

SELECT * FROM Warehouses;


INSERT INTO Books VALUES('9780747532743',20000000002,10000000001,'25-01-
2004','Harry Potter and the Philosophers Stone',449.99);

INSERT INTO Books VALUES('9780747532744',20000000002,10000000001,'17-07-
2004','Harry Potter and the Chamber of Secrets',500.50);
```

```
INSERT INTO Books VALUES('9780747532745',20000000002,10000000001,'15-03-
2005','Harry Potter and the Prisoner of Azkaban',510.31);

INSERT INTO Books VALUES('9780747532746',20000000002,10000000001,'07-06-
2005','Harry Potter and the Goblet of Fire',530.27);

INSERT INTO Books VALUES('9780747532747',20000000002,10000000001,'25-12-
2005','Harry Potter and the Order of Pheonix',570.00);

INSERT INTO Books VALUES('9780747532748',20000000002,10000000001,'20-08-
2006','Harry Potter and the Half Blood Prince',630.47);

INSERT INTO Books VALUES('9780747532749',20000000002,10000000001,'14-07-
2007','Harry Potter and the Deathly Hallows',700.00);

INSERT INTO Books VALUES('9780141346816',20000000003,10000000003,'27-03-
2005','Perccy Jackson The Lightning Thief',890.30);

INSERT INTO Books VALUES('9781484707234',20000000003,10000000003,'18-04-
2011','Perccy Jackson The Last Olympian',900.00);

INSERT INTO Books VALUES('9780520266124',20000000005,10000000002,'23-05-
1876','The Adventures of Tom Sawyer',900.50);

INSERT INTO Books VALUES('9780525120014',20000000005,10000000002,'15-06-
1878','Huckleberry Finn',230.99);

INSERT INTO Books VALUES('9780532021509',20000000001,10000000005,'21-07-
1997','The God of Small Things',500.00);

INSERT INTO Books VALUES('9780350016478',20000000001,10000000005,'05-03-
2017','The Ministry of Utmost Happiness',500.45);

INSERT INTO Books VALUES('9780042100364',20000000001,10000000005,'03-04-
1998','The End of Imagination',360.00);

INSERT INTO Books VALUES('9785633100784',20000000006,10000000004,'03-04-
1998','The Magic Faraway Tree',120.00);

INSERT INTO Books VALUES('9780566210017',20000000006,10000000004,'28-02-
1938','The Secret Island',150.00);

INSERT INTO Books VALUES('9780332100140',20000000006,10000000004,'12-05-
1947','The Valley of Adventure',130.97);

SELECT * FROM Books;


INSERT INTO Warehouse_Book VALUES(40000000001,'9780747532743',200);

INSERT INTO Warehouse_Book VALUES(40000000002,'9780747532743',230);

INSERT INTO Warehouse_Book VALUES(40000000003,'9780747532743',300);

INSERT INTO Warehouse_Book VALUES(40000000004,'9780747532743',120);
INSERT INTO Warehouse_Book VALUES(40000000005,'9780747532743',350);
INSERT INTO Warehouse_Book VALUES(40000000006,'9780747532743',200);

INSERT INTO Warehouse_Book VALUES(40000000001,'9780747532744',250);
INSERT INTO Warehouse_Book VALUES(40000000002,'9780747532744',100);
INSERT INTO Warehouse_Book VALUES(40000000003,'9780747532744',310);
INSERT INTO Warehouse_Book VALUES(40000000004,'9780747532744',250);
INSERT INTO Warehouse_Book VALUES(40000000005,'9780747532744',311);
INSERT INTO Warehouse_Book VALUES(40000000006,'9780747532744',250);

INSERT INTO Warehouse_Book VALUES(40000000001,'9780747532745',120);
INSERT INTO Warehouse_Book VALUES(40000000002,'9780747532745',310);
INSERT INTO Warehouse_Book VALUES(40000000003,'9780747532745',120);
INSERT INTO Warehouse_Book VALUES(40000000004,'9780747532745',120);
INSERT INTO Warehouse_Book VALUES(40000000005,'9780747532745',120);
INSERT INTO Warehouse_Book VALUES(40000000006,'9780747532745',200);

INSERT INTO Warehouse_Book VALUES(40000000001,'9780747532746',200);
INSERT INTO Warehouse_Book VALUES(40000000003,'9780747532746',130);
INSERT INTO Warehouse_Book VALUES(40000000005,'9780747532746',150);
INSERT INTO Warehouse_Book VALUES(40000000006,'9780747532746',90);

INSERT INTO Warehouse_Book VALUES(40000000001,'9780747532747',200);

INSERT INTO Warehouse_Book VALUES(40000000001,'9780747532748',100);
INSERT INTO Warehouse_Book VALUES(40000000003,'9780747532748',100);
INSERT INTO Warehouse_Book VALUES(40000000004,'9780747532748',100);
INSERT INTO Warehouse_Book VALUES(40000000005,'9780747532748',100);

INSERT INTO Warehouse_Book VALUES(40000000005,'9780747532749',10);
INSERT INTO Warehouse_Book VALUES(40000000001,'9780141346816',50);
INSERT INTO Warehouse_Book VALUES(40000000002,'9780141346816',45);
INSERT INTO Warehouse_Book VALUES(40000000003,'9780141346816',150);
INSERT INTO Warehouse_Book VALUES(40000000004,'9780141346816',120);
INSERT INTO Warehouse_Book VALUES(40000000005,'9780141346816',200);
INSERT INTO Warehouse_Book VALUES(40000000006,'9780141346816',150);

INSERT INTO Warehouse_Book VALUES(40000000001,'9781484707234',70);
INSERT INTO Warehouse_Book VALUES(40000000002,'9781484707234',120);
INSERT INTO Warehouse_Book VALUES(40000000003,'9781484707234',130);
INSERT INTO Warehouse_Book VALUES(40000000005,'9781484707234',100);


INSERT INTO Warehouse_Book VALUES(40000000001,'9780520266124',70);
INSERT INTO Warehouse_Book VALUES(40000000002,'9780520266124',120);
INSERT INTO Warehouse_Book VALUES(40000000003,'9780520266124',130);
INSERT INTO Warehouse_Book VALUES(40000000005,'9780520266124',100);

INSERT INTO Warehouse_Book VALUES(40000000001,'9780525120014',50);
INSERT INTO Warehouse_Book VALUES(40000000002,'9780525120014',45);
INSERT INTO Warehouse_Book VALUES(40000000003,'9780525120014',150);
INSERT INTO Warehouse_Book VALUES(40000000004,'9780525120014',120);
INSERT INTO Warehouse_Book VALUES(40000000005,'9780525120014',200);
INSERT INTO Warehouse_Book VALUES(40000000006,'9780525120014',150);

INSERT INTO Warehouse_Book VALUES(40000000001,'9780532021509',50);
INSERT INTO Warehouse_Book VALUES(40000000002,'9780532021509',45);
INSERT INTO Warehouse_Book VALUES(40000000003,'9780532021509',150);
INSERT INTO Warehouse_Book VALUES(40000000004,'9780532021509',120);
INSERT INTO Warehouse_Book VALUES(40000000005,'9780532021509',200);
INSERT INTO Warehouse_Book VALUES(40000000006,'9780532021509',150);
INSERT INTO Warehouse_Book VALUES(40000000001,'9780350016478',175);
INSERT INTO Warehouse_Book VALUES(40000000002,'9780350016478',150);
INSERT INTO Warehouse_Book VALUES(40000000003,'9780350016478',100);
INSERT INTO Warehouse_Book VALUES(40000000004,'9780350016478',200);
INSERT INTO Warehouse_Book VALUES(40000000005,'9780350016478',100);
INSERT INTO Warehouse_Book VALUES(40000000006,'9780350016478',150);

INSERT INTO Warehouse_Book VALUES(40000000001,'9780042100364',175);
INSERT INTO Warehouse_Book VALUES(40000000002,'9780042100364',150);
INSERT INTO Warehouse_Book VALUES(40000000006,'9780042100364',150);

INSERT INTO Warehouse_Book VALUES(40000000004,'9785633100784',200);
INSERT INTO Warehouse_Book VALUES(40000000005,'9785633100784',100);
INSERT INTO Warehouse_Book VALUES(40000000006,'9785633100784',150);

INSERT INTO Warehouse_Book VALUES(40000000001,'9780566210017',200);
INSERT INTO Warehouse_Book VALUES(40000000002,'9780566210017',140);
INSERT INTO Warehouse_Book VALUES(40000000003,'9780566210017',300);
INSERT INTO Warehouse_Book VALUES(40000000004,'9780566210017',170);
INSERT INTO Warehouse_Book VALUES(40000000005,'9780566210017',280);
INSERT INTO Warehouse_Book VALUES(40000000006,'9780566210017',190);

INSERT INTO Warehouse_Book VALUES(40000000001,'9780332100140',120);
INSERT INTO Warehouse_Book VALUES(40000000002,'9780332100140',130);
INSERT INTO Warehouse_Book VALUES(40000000003,'9780332100140',400);
INSERT INTO Warehouse_Book VALUES(40000000004,'9780332100140',320);
INSERT INTO Warehouse_Book VALUES(40000000005,'9780332100140',250);
INSERT INTO Warehouse_Book VALUES(40000000006,'9780332100140',180);

SELECT * FROM Warehouse_Book;
```

```
INSERT INTO ShoppingBasket_Book VALUES(30000000001,'9780747532743',1);
INSERT INTO ShoppingBasket_Book VALUES(30000000001,'9780747532744',1);
INSERT INTO ShoppingBasket_Book VALUES(30000000001,'9780747532745',1);
INSERT INTO ShoppingBasket_Book VALUES(30000000001,'9780747532746',1);
INSERT INTO ShoppingBasket_Book VALUES(30000000001,'9780747532747',1);
INSERT INTO ShoppingBasket_Book VALUES(30000000001,'9780747532748',1);
INSERT INTO ShoppingBasket_Book VALUES(30000000001,'9780747532749',1);

INSERT INTO ShoppingBasket_Book VALUES(30000000002,'9780042100364',2);
INSERT INTO ShoppingBasket_Book VALUES(30000000002,'9780520266124',1);

INSERT INTO ShoppingBasket_Book VALUES(30000000003,'9780532021509',1);
INSERT INTO ShoppingBasket_Book VALUES(30000000003,'9780350016478',2);
INSERT INTO ShoppingBasket_Book VALUES(30000000003,'9781484707234',2);

INSERT INTO ShoppingBasket_Book VALUES(30000000005,'9780525120014',1);
INSERT INTO ShoppingBasket_Book VALUES(30000000005,'9780332100140',2);
INSERT INTO ShoppingBasket_Book VALUES(30000000005,'9780747532747',1);

INSERT INTO ShoppingBasket_Book VALUES(30000000006,'9781484707234',3);
SELECT * FROM ShoppingBasket_Book;
```

RECOMMENDED JOINS AND VIEWS

The View BookDetailsView shows detailed view of the Books along with the Author Name and Publisher Name:

```
CREATE OR REPLACE VIEW BookDetailsView AS

    SELECT Books.ISBN ,Books.Title , Authors.AuthorFName, Authors.AuthorLName ,
Publishers.PublisherName, Book.Price

    FROM Books NATURAL JOIN Authors NATURAL JOIN Publishers;

SELECT * FROM BookDetailsView;
```

The View AuthorDetailsView shows detailed view of the Authors along with the Author Phone Numbers and the number of Books written by the Author:

```
CREATE OR REPLACE VIEW AuthorDetailsView AS

    WITH TTABLE(AuthorID,Phone) AS

    (SELECT AuthorID, RTRIM (XMLAGG (XMLELEMENT (e, AuthorPhone || ' ,
')).EXTRACT ('//text()'), ',') Phone

    FROM AuthorPhones GROUP BY AuthorID),

    TTABLE2(AuthorID,BooksWritten) AS

    (SELECT AuthorID,COUNT(*) FROM Books GROUP BY AuthorID)

        SELECT
Authors.AuthorID,AuthorFName,AuthorLName,AuthorURL,TTABLE.Phone,AuthorAddress,B
ooksWritten

        FROM Authors LEFT JOIN TTABLE ON(Authors.AuthorID = TTABLE.AuthorID)
LEFT JOIN TTABLE2 ON (Authors.AuthorID=TTABLE2.AuthorID);

SELECT * FROM AuthorDetailsView;
```

The View PublisherDetailsView shows detailed view of the Publishers along with the Publisher Phone Numbers and the number of Books published by the Publisher:

```
CREATE OR REPLACE VIEW PublisherDetailsView AS

    WITH TTABLE(PublisherID,Phone) AS

    (SELECT PublisherID, RTRIM (XMLAGG (XMLELEMENT (e, PublisherPhone || ' ,
')).EXTRACT ('//text()'), ',') Phone

    FROM PublisherPhones

    GROUP BY PublisherID),
    TTABLE2(PublisherID,BooksPublished) AS

    (SELECT PublisherID, COUNT(*) FROM Books GROUP BY PublisherID)

        SELECT
Publishers.PublisherID,PublisherName,PublisherURL,TTABLE.Phone,PublisherAddress
,BooksPublished

        FROM Publishers LEFT JOIN TTABLE ON(Publishers.PublisherID =
TTABLE.PublisherID) LEFT JOIN TTABLE2 ON(Publishers.PublisherID =
TTABLE2.PublisherID);

SELECT * FROM PublisherDetailsView;
```

The view CustomerDetailsView shows detailed information about Customers that even includes their Phone Numbers:

```
CREATE OR REPLACE VIEW CustomerDetailsView AS

    WITH TTABLE(CustomerID,Phone) AS

    (SELECT CustomerID, RTRIM (XMLAGG (XMLELEMENT (e, CustomerPhone || ' ,
')).EXTRACT ('//text()'), ',') Phone

    FROM CustomerPhones

    GROUP BY CustomerID)

        SELECT
Customers.CustomerID,CustomerFName,CustomerLName,CustomerUsername,CustomerPassw
ord,CustomerEmail,TTABLE.Phone,CustomerAddress

        FROM Customers LEFT JOIN TTABLE ON(Customers.CustomerID =
TTABLE.CustomerID);

SELECT * FROM Customers;
```

Implementation of PLSQL Functions/Triggers

PLSQL Functions and Triggers allow us to easily regulate changes in data and view these changes in a compact manner.

Most of the PLSQL Triggers are used before, during or after the execution of an SQL statement which provides the Database Administrator with numerous functionalities.

I have implemented following PLSQL Functionalities for convenient searching of Data in the Relations formed in the previous section:-

The below Procedure is used for Displaying Shopping Cart of a given User based on Username as Input:

```
CREATE OR REPLACE PROCEDURE show_shopping_basket(customer_username IN VARCHAR)

IS

total NUMBER(10,2);

BEGIN

    DBMS_OUTPUT.PUT_LINE('Shopping Basket: '||customer_username);

    total := 0.00;

    FOR rec IN (SELECT ISBN,Title,Price,BookCount FROM Books NATURAL JOIN
ShoppingBasket_Book NATURAL JOIN Customers WHERE CustomerUsername =
customer_username)

    LOOP

        total := total + (rec.Price * rec.BookCount);

        DBMS_OUTPUT.PUT_LINE(rec.ISBN||' '||rec.Title||' '||rec.Price||'
'||rec.BookCount);

        END LOOP;

    DBMS_OUTPUT.PUT_LINE('Shopping Basket Total: Rs.'||total);

END;

/


DECLARE

    customer_username VARCHAR(255):= '&customer_username';

BEGIN

    show_shopping_basket(customer_username);

END;

/
```

The below Trigger warns the Administrator when the Book Count of a particular warehouse for a particular existing Book falls below 20 and asks for Restocking books:

```
CREATE OR REPLACE TRIGGER show_low_book_levels

AFTER INSERT OR UPDATE ON Warehouse_Book

FOR EACH ROW

WHEN (NEW.BookCount<20)

DECLARE

    CURSOR cur IS

    SELECT ISBN,Title FROM Books WHERE ISBN = :NEW.ISBN;

BEGIN

    FOR rec IN cur

    LOOP

        DBMS_OUTPUT.PUT_LINE(rec.ISBN||' '||rec.Title||' '||'have been reduced
below 20 at Warehouse '||:NEW.WarehouseID||'! Restock required!');

    END LOOP;

END;

/

Update Warehouse_Book set BookCount=15 WHERE ISBN = '9780747532743' AND
WarehouseID=40000000006;
```

The Mini Project has been successfully completed under the guidance of SVNIT Computer Engineering Department faculties for Course BTech 2nd Year DBMS Course

THANK YOU