Project Workbook:

Design & Implementation of "Online Retail Catalog"

By Neelima Potharaj, Dheepak Chidambaram

Project Plan

Description and Functionalities:

The project consists of a retail purchasing application implementation. A customer can register to purchase an item from a catalog. The first time a customer registers, details like username, password, bank name, and bank account number will be stored in a database. A product catalog displays the details present in the product database for the customer. As a customer makes purchases, all the purchase details are also stored into the database. Once they select the products from the products catalog, the final purchase details are presented to the customer along with the product prices and the total price in a receipt format. The customer will also be able to view their purchase history.

Technical Functionalities:

1. New customer instance is created in the Customer relation when a new customer registers.
2. Details from each purchase are stored in the Purchase relation of the database.
3. Customer Login/Logout process through opening and closing of a session in php.
4. Insertion to the purchase tables, when a customer makes a purchase.
5. Updating number of purchases detail in the customer table, when a customer makes a purchase.
6. Display of order history.
7. Ability to make more than one purchase during a logged in session.
8. Registration of new users (Insertion into customer table).
9. Search option to display specific products.
10. My profile page to view the customer's account details and to edit these details.

Note: The unit test case chart includes more specific details about the functionalities.

Data Model & Schema:

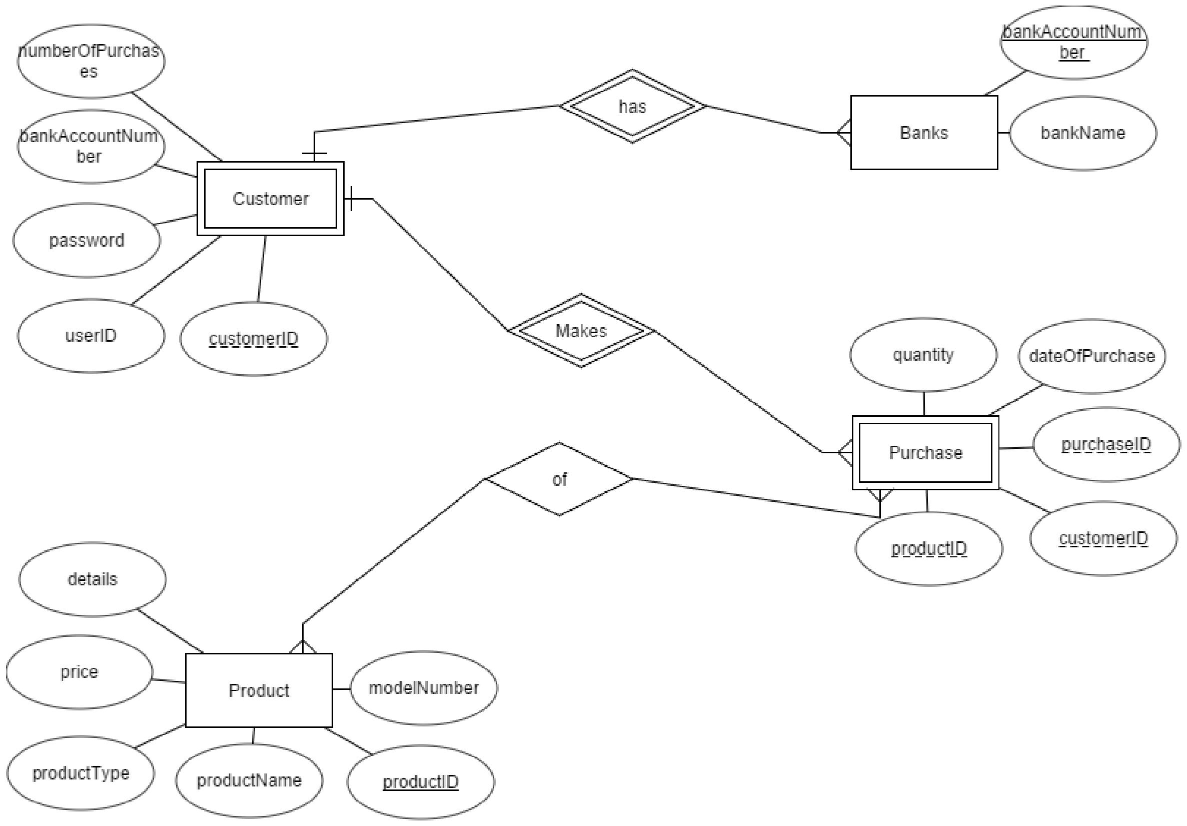
Primary design

1. Customer (customerID, userID, password, bankAccountNumber, numberOfPurchases)
2. Banks (bankAccountNumber, bankName)
3. Product (productID, productName, productType, details, price)
4. Purchase (purchaseID, customerID, productID, quantity, dateOfPurchase)

Intermediate design

1. ER Model
    - A customer can have many bank accounts; however, a bank account is associated with only one customer.
    - A customer can more than one purchase, however a single purchase is only associated with one customer (because of the presences of the attribute 'customerID').
    - A purchase can be of having several products, and a product can be associated with several purchases.

- Customer is a weak entity, thereby making 'has' relationship a weak relationship. As this an online retail system customers need to have bank accounts to make purchases.
- Purchase is also a weak entity, making 'makes' a weak relationship. Purchases do not exist without customers. A customer could have an account on the online retail system without making any purchases.



2. Relational Schema
   Banks (<u>bankAccountNumber</u>, bankName)
   Customers (<u>customerID</u>, userID, password, numberOfPurchases, bankAccountNumber)
   Purchase (<u>purchaseID</u>, <u>customerID</u>, <u>productID</u>, quantity, dateOfPurchase)
   Product (<u>productID</u>, productName, modelNumber, productType, price, details)
   Note: 'of' relationship is not displayed as relation because it would make a poor design choice for this case. The relation 'of' would contain productID, customerID, and purchaseID as its attributes. However, all three attributes are already presented together in the 'Purchase' relation.
3. Normalization
   a. Banks (bankAccountNumber, bankName)
      <u>Assumptions</u>: the first 2 digits in the 'bankAccountNumber' are uniquely associated with the 'bankName'. No special bank id column was created to avoid

unnecessary memory allocation; the name of the bank does not need to be queried as often for our online retail system.

FDs and MVDs:

bankAccountNumber → bankName

- As this relational schema consists does not contain any of the 1NF violations (importantly does not contain duplicate tuples and the tuples or columns do not need to be in any specific order), it is in 1NF. This schema only consists of 2 attributes which makes it to be in all Normal Forms, thereby in 4NF.

b. Customers (customerID, userID, password, numberOfPurchases, bankAccountNumber)

   Assumptions: customerID uniquely determines each tuple, as there will be no customerID duplicates in this relation. 2 different customerID values could have the same userID and password values but would have different bankAccountNumber values.

   FDs and MVDs:

   customerID → bankAccountNumber, userID, password, numberOfPurchases
   customerID, bankAccountNumber→userID, password, numberOfPurchases

   - As this relational schema consists does not contain any of the 1NF violations (importantly does not contain duplicate tuples and the tuples or columns do not need to be in any specific order), it is in 1NF. Does not contain any FDs where partial keys determine non-key attributes, thereby in 2NF. No transitive FDs present, leading to 3NF. As every key is a candidate key, this is in BCNF. In BCNF with no MVDs makes this relation to be in 4NF.

c. Purchase (purchaseID, customerID, productID, quantity, dateOfPurchase)

   Assumptions: A purchaseID could be the same between different customerIDs (several customers could have made 3 purchases to be linked to purchaseID '3'). Similarly, productID is not unique to neither purchaseID nor customerID. Moreover, one customerID could be associated with several purchaseIDs. Once a customer chooses products to buy and clicks the pay button, this dataset of products and quantity would be stored as one purchase. And if this is the customer's first purchase, then this particular instance of purchase would be deemed with purchaseID '1'.

   FDs and MVDs:

   purchaseID, customerID → dateOfPurchase
   purchaseID, customerID, productID → quantity

   1. In 1NF: along with other requirements to be a relational schema, does not contain duplicate tuples and the tuples or columns do not need to be in any specific order.
   2. Not in 2NF: partial key does determine non-key attributes. However not in 2NF, as partial key determines non-key attribute (purchaseID, customerID → dateOfPurchase).

   - New Schema:
     - purchaseDate (purchaseID, customerID, dateOfPurchase)

FDs and MVDs:
purchaseID, customerID → dateOfPurchase

1. In 1NF: along with other requirements to be a relational schema, does not contain duplicate tuples and the tuples or columns do not need to be in any specific order.
2. In 2NF: partial key does not determine non-key attributes.
3. In 3NF: no transitive nature present here.
4. In BCNF: determinants in the FDs are part of the primary key.
5. In 4NF: in BCNF with no non-trivial MVDs.

- purchaseQuantity (purchaseID, customerID, productID, quantity)
  FDs and MVDs:
  purchaseID, customerID, productID → quantity
  1. In 1NF: along with other requirements to be a relational schema, does not contain duplicate tuples and the tuples or columns do not need to be in any specific order.
  2. In 2NF: partial key does not determine non-key attributes.
  3. In 3NF: no transitive nature present here.
  4. In BCNF: determinants in the FDs are part of the primary key.
  5. In 4NF: in BCNF with no non-trivial MVDs.

d. Product (productID, productName, productType, modelNumber, details, price)
Assumptions: productID uniquely determines each tuple in this relation. The attribute details just contain extra description to display on the interface. Two productName values could be associated with same modelNumber values, however a single productName would associated with a modelNumber only once.
FDs and MVDs:
productID → productName, productType, price
productName, modelNumber → productID, productType, price

1. In 1NF: along with other requirements to be a relational schema, does not contain duplicate tuples and the tuples or columns do not need to be in any specific order.
2. In 2NF: partial key does not determine non-key attributes as the primary key is only one attribute.
3. In 3NF: no transitive nature present here.
4. In BCNF: determinants in FDs (other than the primary key FD) is a candidate key.
5. In 4NF: in BCNF with no non-trivial MVDs.

Final design

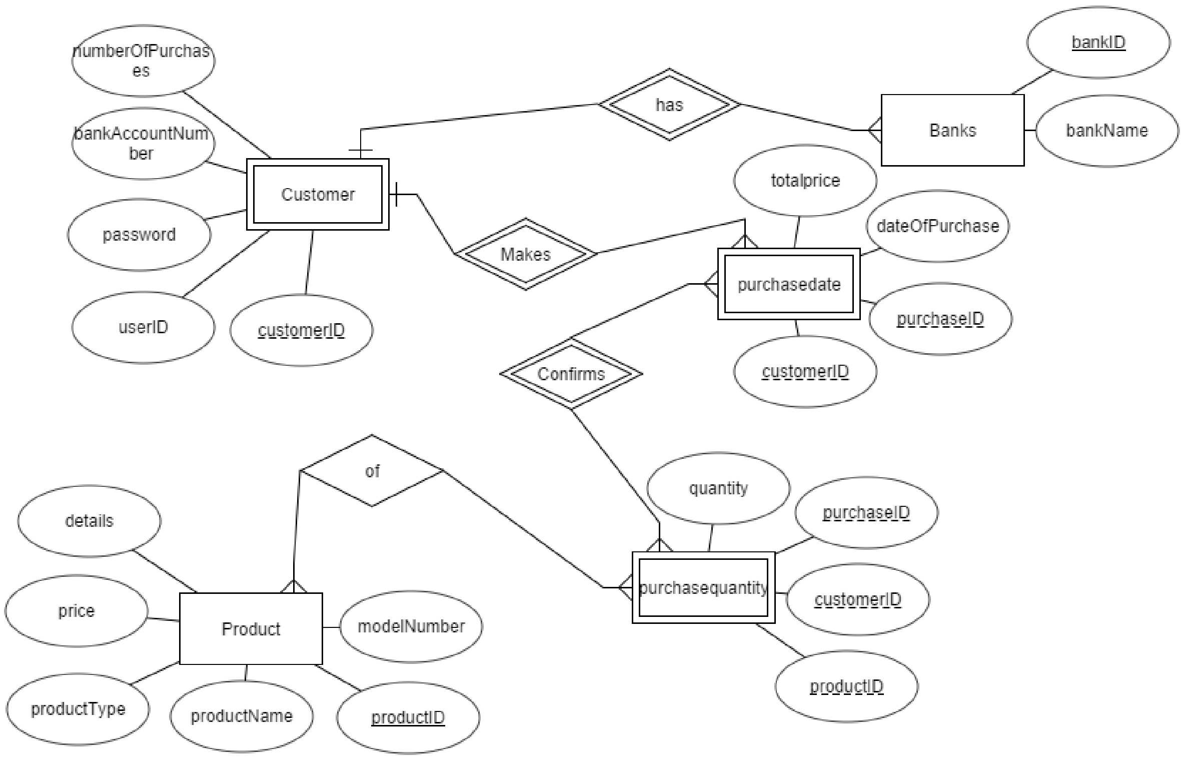
1. Updates to the Design

a. Addition of attribute "totalprice" to purchaseDate table.
- This addition does not skew with the normalization process as it passes all
  Normal forms with no constraints.

b. Modification of banks table.
    - BankAccountNumber attribute was replaced by BankID, as BankAccountNumber attribute stored in banks uses too much of memory when banks table is only used for accessing names of the banks the BankAccountNumber is associated with. BankID shortened this storage.
    - The last two digits of the each BankAccountNumber value from customer table represent the bankID that the number is associated with.
    - So, BankAccountNumber is no longer a foreign key to the customer table.

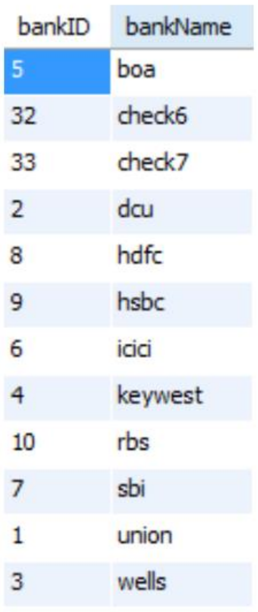
      This did not impact the normalization process of banks as it in 4NF form.

c. Updated ER diagram representing the above changes



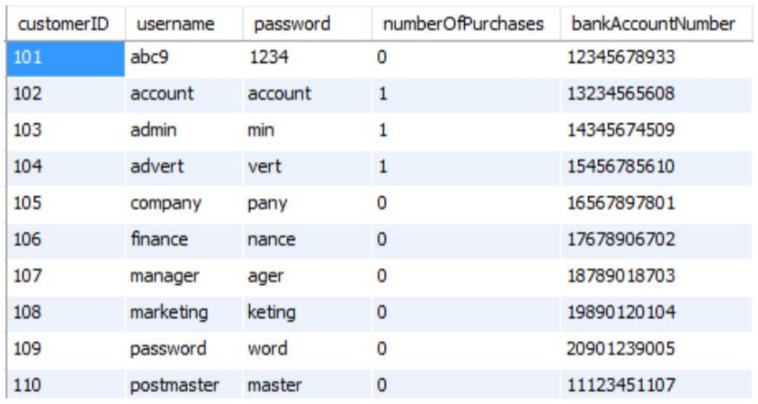
Populate Database:

1. SELECT * FROM retailpurchase.banks;



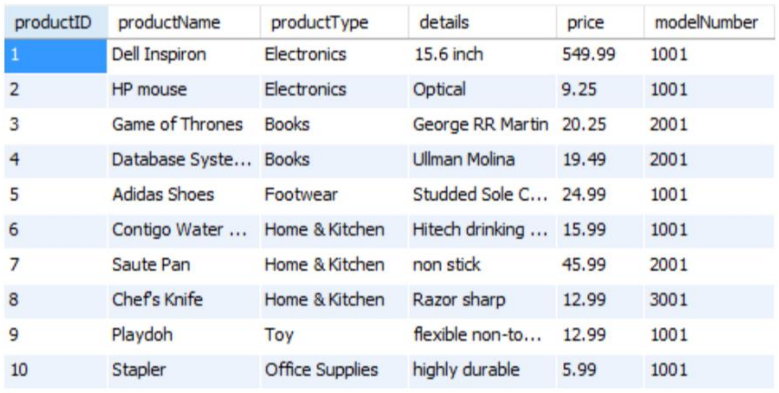
bankID	bankName
5	boa
32	check6
33	check7
2	dcu
8	hdfc
9	hsbc
6	icici
4	keywest
10	rbs
7	sbi
1	union
3	wells

2. SELECT * FROM retailpurchase.customer;



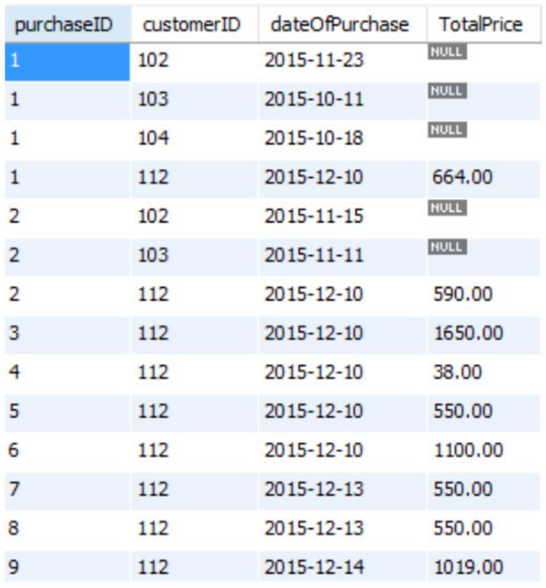
customerID	username	password	numberOfPurchases	bankAccountNumber
101	abc9	1234	0	12345678933
102	account	account	1	13234565608
103	admin	min	1	14345674509
104	advert	vert	1	15456785610
105	company	pany	0	16567897801
106	finance	nance	0	17678906702
107	manager	ager	0	18789018703
108	marketing	keting	0	19890120104
109	password	word	0	20901239005
110	postmaster	master	0	11123451107

3. SELECT * FROM retailpurchase.product;



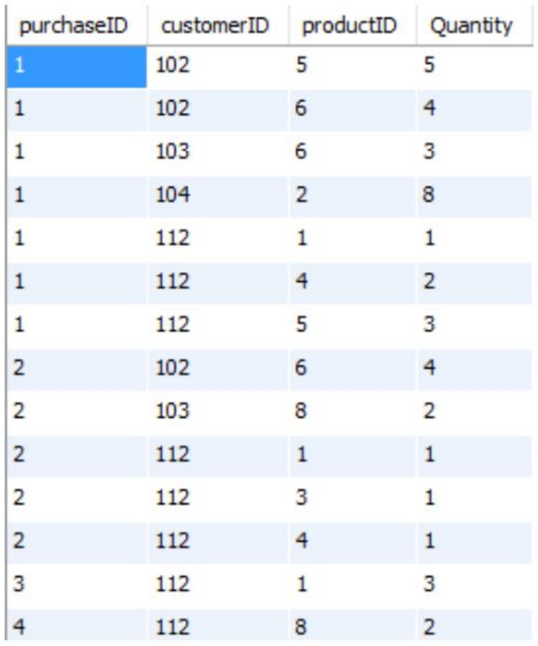
productID	productName	productType	details	price	modelNumber
1	Dell Inspiron	Electronics	15.6 inch	549.99	1001
2	HP mouse	Electronics	Optical	9.25	1001
3	Game of Thrones	Books	George RR Martin	20.25	2001
4	Database Syste...	Books	Ullman Molina	19.49	2001
5	Adidas Shoes	Footwear	Studded Sole C...	24.99	1001
6	Contigo Water ...	Home & Kitchen	Hitech drinking ...	15.99	1001
7	Saute Pan	Home & Kitchen	non stick	45.99	2001
8	Chef's Knife	Home & Kitchen	Razor sharp	12.99	3001
9	Playdoh	Toy	flexible non-to...	12.99	1001
10	Stapler	Office Supplies	highly durable	5.99	1001

4. SELECT * FROM retailpurchase.purchasedate;



purchaseID	customerID	dateOfPurchase	TotalPrice
1	102	2015-11-23	NULL
1	103	2015-10-11	NULL
1	104	2015-10-18	NULL
1	112	2015-12-10	664.00
2	102	2015-11-15	NULL
2	103	2015-11-11	NULL
2	112	2015-12-10	590.00
3	112	2015-12-10	1650.00
4	112	2015-12-10	38.00
5	112	2015-12-10	550.00
6	112	2015-12-10	1100.00
7	112	2015-12-13	550.00
8	112	2015-12-13	550.00
9	112	2015-12-14	1019.00

5. SELECT * FROM retailpurchase.purchasequantity;



purchaseID	customerID	productID	Quantity
1	102	5	5
1	102	6	4
1	103	6	3
1	104	2	8
1	112	1	1
1	112	4	2
1	112	5	3
2	102	6	4
2	103	8	2
2	112	1	1
2	112	3	1
2	112	4	1
3	112	1	3
4	112	8	2

Example Queries:

Queries

1. To group products together for clean display (In-progress). Planning to implement this with
   a click of a button.

   *Select **
   From Product

```
group by productType, productID;
```

2. To display order history that includes details from purchasequantity and product tables.

```
Select purchasequantity.productID, productName, productType, (price*Quantity) as
Price
From purchasequantity join product
on (purchasequantity.productID = product.productID)
where customerID = 112 AND purchaseID = 9;
```

3. To retrieve name of the bank for displaying it in the profile page. This is query gets the last two digits of the bankAccountNumber which is in turn the bankID.

```
Select bankName
from banks
where bankID = (
        Select MOD(bankAccountNumber, 100)
  from customer
  where customerID = 112);
```

4. New user bank details insertion

```
Insert into banks values ('banknum', 'bankname');
```

5. New user personal details insertion

```
Insert into customer (username, password, numberOfPurchases, bankAccountNumber)
values ('username', 'password', 0, 'banknum');
```

Stored Procedures

1. Selecting bank for user profile

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `select_banks`(IN cusID int,
OUT bnknm varchar(25), OUT bnkaccno int)
BEGIN
Select bankName , banks.bankAccountNumber into bnknm,bnkaccno
from banks join customer
on (banks.bankAccountNumber = customer.bankAccountNumber)
where customerID = cusID;
END
```

2. Inserting on purchase confirm

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `insert_purchase_date`(IN
purID int, IN cusID int, IN totprice float)
```

```
BEGIN
insert into purchasedate
values
(purID,cusID,curdate(),totprice);
END

CREATE DEFINER=`root`@`localhost` PROCEDURE `insert_purchase_quantity`(IN
purID int,IN cusID int, IN proID int, IN quantity int)
BEGIN
insert into purchasequantity
values
(purID,cusID,proID,quantity);
END
```

Discussion:

The online retail application deals with displaying product related data for view and purchase by customers. We were able to accomplish many of planned implementations. However, to improve this project two main concerns need to be addressed: implementation of a robust and simple to use program and reduction of the amount of MySQL queries used. The current system contains some setbacks due the lack of functionality present in the php code, such as when the programming becomes skewed in some pages when the refresh button on the browser is clicked. To launch this program for a wider audience, further implementation can be done to involve accounting information of customer to finalize the purchase of products. Moreover, further security constraints need to be setup as this program involves sensitive information such as the bank account details. Finally, customer related information like geography and product preference can be used to recommend products through predictive modelling, to make this product more accessible to general public. Overall this project has been a good learning experience for both designing the database and working with web development.

Bibliography:

Build a Shopping Cart With PHP and MySQL - Envato Tuts+ Code Tutorial. (n.d.). Retrieved December 16, 2015, from http://code.tutsplus.com/tutorials/build-a-shopping-cart-with-php-and-mysql--net-5144

PHP 5 Tutorial. (n.d.). Retrieved December 16, 2015, from http://www.w3schools.com/pHP/

PHP Login Form with Sessions | FormGet. (n.d.). Retrieved December 16, 2015, from http://www.formget.com/login-form-in-php/