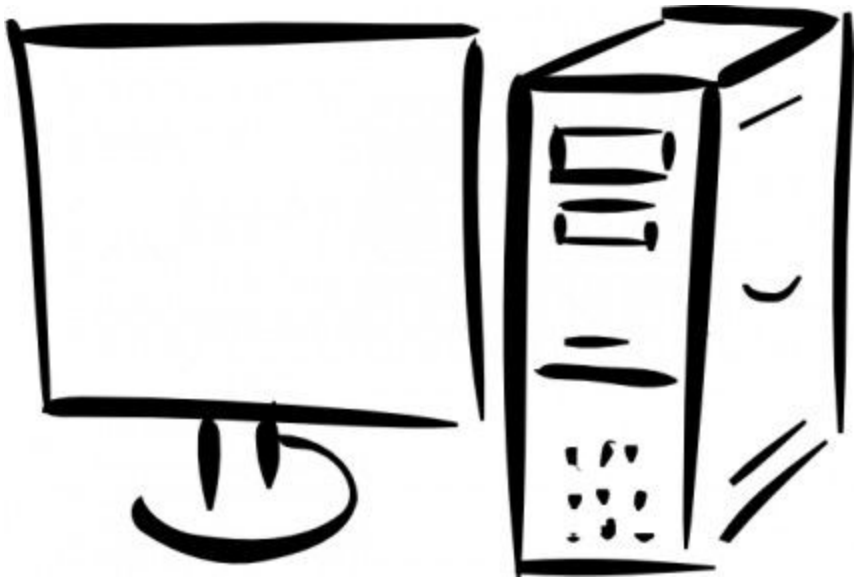
Bachelor of Science in Information Technology

Database Design Concepts Project

ITDA211 - 2017



Matthew Van der Bijl
Student Number: XQ9X3WV31

Table of Contents

Introduction

Please refer to the README handed in alongside this document. The QAC Shop requires a new Information System (IS) to manage its inventory. The QAC Shop is a home furnishing stored in a upscale urban neighbourhood that sells both antiques as well as new items to its customers. The new items sold are design to complement the antiques. The QAC Shop purchases its antiques from wholesalers and individual and the current items are bought from distributors.

The new system should be able to:

1. Keep records of all sales made;
2. Allow customers to view the availability of all products;
3. Allow the shop to buy stock from individuals and wholesalers as well as distributors;
4. Reorder stock; and
5. Generate and display the customer's bill.

All antiques are unique and thus only can ever be in stock while there can be multiple new items in stock and once depleted an order can be placed for more. New items may come in various colours and sizes. The system will make use of a MySQL database to store data and will be interacted with using Structured Query Language (SQL).

As per the project specifications my student number (XQ9X3WV31) has been used as a part of the naming convention. More than four interrelated tables have been used. The document that follows will provide screenshots of my work.

Database Structure and Design

Entity Relationship Diagram (ERD)

The ERD below show all entities and their attributes that exist within the QAC Shop's new inventory system. An JPG image of this ERD will be submitted along side this document.

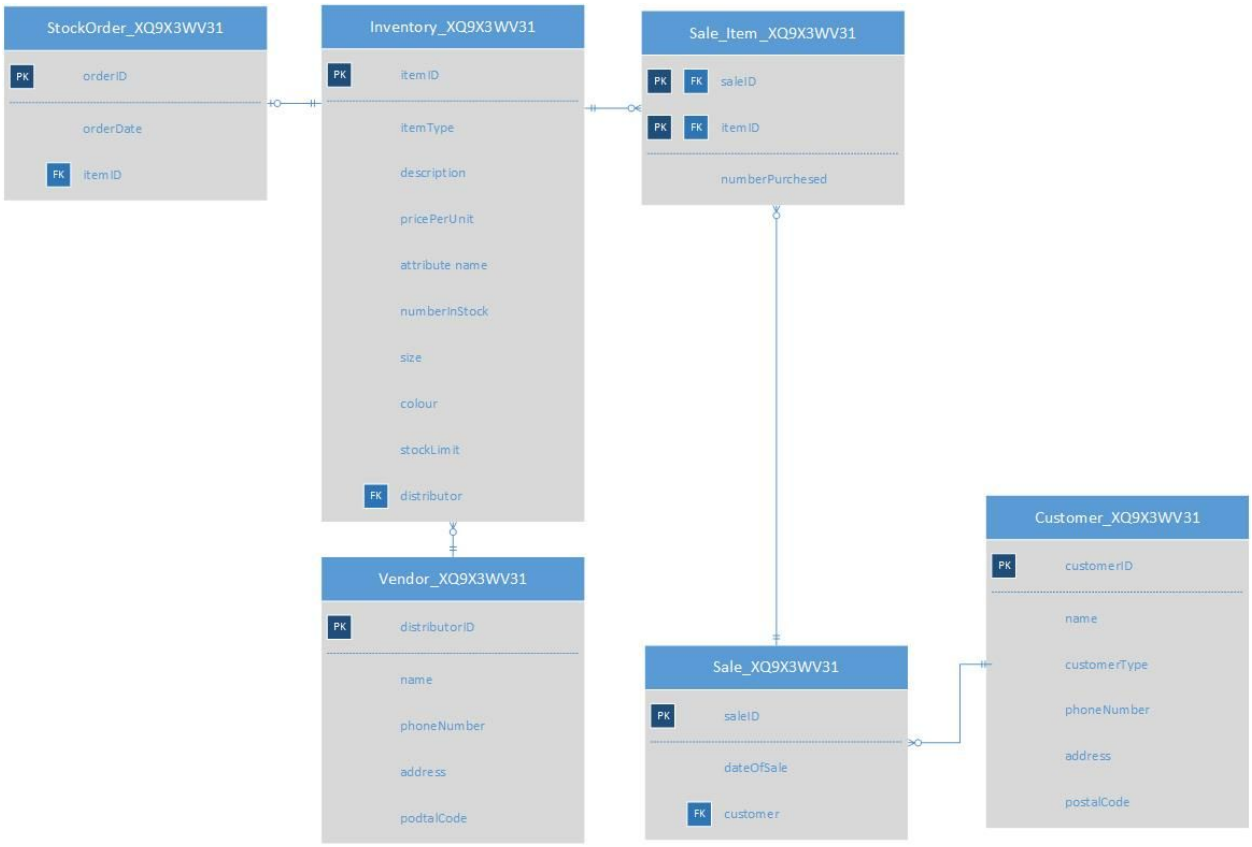


Figure 1 - ERD displaying all entities

Database Table

The tables below specifies the column properties for each table in the database.

Customer Item

Table 1 - Customer table

Customer_XQ9X3WV31					
Column Name	**Data Type (Length)**	**Key**	**Required**	**Default Value**	**Remarks**
customerID	Integer	Primary Key	Yes	None	Surrogate key.
name	Varchar	No	Yes	Null	The first name of the customer.
customerType	Varchar	No	Yes	Null	The type customer it is/they are.
phoneNumber	Varchar	No	Yes	Null	The customer's phone number.
address	Varchar	No	Yes	Null	The customer's address.
postalCode	Varchar	No	Yes	Null	The postal code of the Customer address.

Sale Table

Table 2 - Sale table

Sale_XQ9X3WV31					
Column Name	**Data Type (Length)**	**Key**	**Required**	**Default Value**	**Remarks**
invoiceNumber	Integer	Primary Key	Yes	None	Surrogate key.
dateOfSale	Date	No	Yes	None	The date that the sale was made on.
customer	Integer	Foreign Key	Yes	None	Reference to a record in the Customer table.

Sale_Item Table

Table 3 - Sale_Item table

Sale_Item_XQ9X3WV31					
Column Name	**Data Type (Length)**	**Key**	**Required**	**Default Value**	**Remarks**
saleID	Integer	Primary Key Foreign Key	Yes	None	Reference to a record in the Sales table.
itemID	Integer	Primary Key Foreign Key	Yes	None	Reference to a record in the Inventory table.
numberPurchesed	Integer	No	Yes	None	The number of Items purchased.

Item Table

Table 4 - Item table

Inventory_XQ9X3WV31					
Column Name	**Data Type (Length)**	**Key**	**Required**	**Default Value**	**Remarks**
itemID	Integer	Primary Key	Yes	None	Surrogate key. Similar to an item's barcode.
itemType	Varchar	No	Yes	None	The type of item (Current Product or Antique).
description	Varchar	No	Yes	None	Description of the item.
pricePerUnit	Numeric	No	Yes	None	The price at which the item is bought at (Inc. VAT).
markup	Numeric	No	Yes	None	The markup that the item is being sold at.
numberOfStock	Integer	No	Yes	None	The number of items currently in stock.
itemSize	Integer	No	No	Null	The size of the item if it is new.
colour	Varchar	No	No	Null	The colour of the item if it is new.
stockLimit	Integer	No	Yes	0	Once the stock drops

					below this limit it is repurchased
distributor	Integer	Foreign Key	Yes	None	From whom the item was purchased.

Vendor Table

Table 5 - Vendor table

Vendor_XQ9X3WV31					
Column Name	**Data Type (Length)**	**Key**	**Required**	**Default Value**	**Remarks**
distributorID	Integer	Primary Key	Yes	None	Surrogate key.
name	Varchar	No	Yes	None	The name of the Distributor.
phoneNumber	Varchar	No	Yes	None	Distributor's phone number.
address	Varchar	No	Yes	None	The Distributor's address.
postalCode	Varchar	No	Yes	None	The postal code of the Distributor address.

Stock Order

Table 6 - StockOrder table

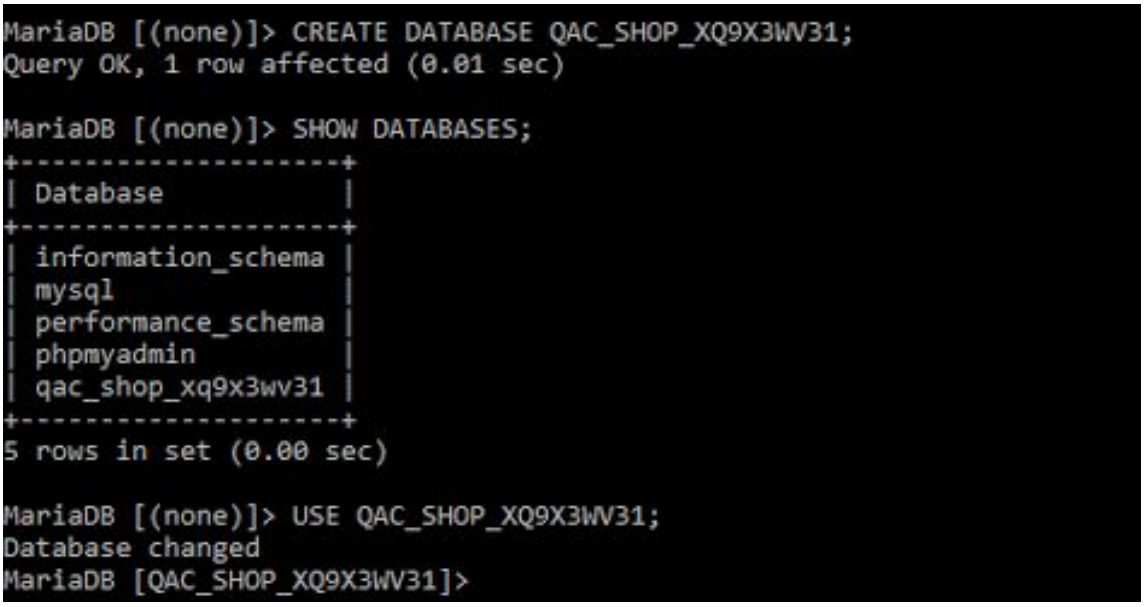
StockOrder_XQ9X3WV31					
Column Name	**Data Type (Length)**	**Key**	**Required**	**Default Value**	**Remarks**
orderID	Integer	Primary Key	Yes	None	Surrogate key.
orderDate	Date	No	Yes	None	The date that the order was made on.
itemID	Integer	Foreign Key	Yes	None	Reference to the Inventory Table.

SQL Statements

Create Database

The statements that follow are used to create the database:

1. CREATE DATABASE QAC_SHOP_XQ9X3WV31;
2. SHOW DATABASES; and
3. USE QAC_SHOP_XQ9X3WV31.



Create Tables

As indicated in Figure 1 the table will consist of four tables. The SQL statements below have been used to create the tables in the QAC_SHOP database. Once all tables have been created it should be reflected in MySQL Workbench as indicated in Figure 3.

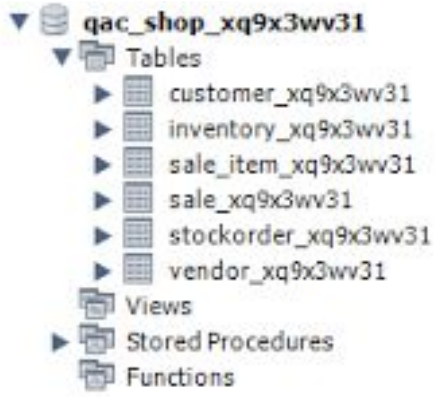


Figure 3 - The created tables

Vendor

The Vendor table will be used to store all data associated with any entity that sells stock to the QAC shop.

```
MariaDB [QAC_SHOP_XQ9X3WV31]> CREATE TABLE Vendor_XQ9X3WV31 (
    -> VendorID INTEGER NOT NULL PRIMARY KEY AUTO_INCREMENT, /* PK */
    -> name VARCHAR(32),
    -> VendorType VARCHAR(32),
    -> phoneNumber VARCHAR(10),
    -> address VARCHAR(32),
    -> postalCode VARCHAR(4)
    -> );
Query OK, 0 rows affected (0.05 sec)
```

Figure 4 - The statements used to create the Vendor table

```
MariaDB [QAC_SHOP_XQ9X3WV31]> INSERT INTO Vendor_XQ9X3WV31 (name, vendorType, phoneNumber, address, postalCode)
    -> VALUES ('Bob MacDavid', 'individual', '0826456601', '45 Berg Street', '6456');
Query OK, 1 row affected (0.01 sec)

MariaDB [QAC_SHOP_XQ9X3WV31]>
MariaDB [QAC_SHOP_XQ9X3WV31]> INSERT INTO Vendor_XQ9X3WV31 (name, vendorType, phoneNumber, address, postalCode)
    -> VALUES ('Shavon Custard', 'individual', '0837991047', '10 East Street', '8069');
Query OK, 1 row affected (0.01 sec)

MariaDB [QAC_SHOP_XQ9X3WV31]>
MariaDB [QAC_SHOP_XQ9X3WV31]> INSERT INTO Vendor_XQ9X3WV31 (name, vendorType, phoneNumber, address, postalCode)
    -> VALUES ('Rebekah Veal', 'individual', '0796471078', '78 Harrison Avenue', '0232');
Query OK, 1 row affected (0.01 sec)

MariaDB [QAC_SHOP_XQ9X3WV31]>
MariaDB [QAC_SHOP_XQ9X3WV31]> INSERT INTO Vendor_XQ9X3WV31 (name, vendorType, phoneNumber, address, postalCode)
    -> VALUES ('Elmer Triplett', 'individual', '0794221043', '62 Devonshire Drive', '6523');
Query OK, 1 row affected (0.00 sec)

MariaDB [QAC_SHOP_XQ9X3WV31]>
MariaDB [QAC_SHOP_XQ9X3WV31]> INSERT INTO Vendor_XQ9X3WV31 (name, vendorType, phoneNumber, address, postalCode)
    -> VALUES ('Elton Beauchesne', 'individual', '0793171012', '28 Cypress Court', '5411');
Query OK, 1 row affected (0.00 sec)

MariaDB [QAC_SHOP_XQ9X3WV31]>
MariaDB [QAC_SHOP_XQ9X3WV31]> INSERT INTO Vendor_XQ9X3WV31 (name, vendorType, phoneNumber, address, postalCode)
    -> VALUES ('Anibal Alsop', 'individual', '0794501014', '32 Pine Street', '4538');
Query OK, 1 row affected (0.00 sec)

MariaDB [QAC_SHOP_XQ9X3WV31]>
MariaDB [QAC_SHOP_XQ9X3WV31]> INSERT INTO Vendor_XQ9X3WV31 (name, vendorType, phoneNumber, address, postalCode)
    -> VALUES ('Isaura Brumfield', 'individual', '0831651082',  '62 Devonshire Drive', '5621');
Query OK, 1 row affected (0.01 sec)
```

Figure 5 - Example insert statements

vendorID	name	vendorType	phoneNumber	address	postalCode
1	Valint & Balk	distributor	0826776601	56 Main Road	7975
2	E-Corp	wholesaler	0826756601	76 Hill Street	2343
3	Sam Jacobs	individual	0218954842	73 Berg Street	6456
4	Bob MacDavid	individual	0826456601	45 Berg Street	6456
5	Shavon Custard	individual	0837991047	10 East Street	8069
6	Rebekah Veal	individual	0796471078	78 Harrison Avenue	0232
7	Elmer Triplett	individual	0794221043	62 Devonshire Drive	6523
8	Elton Beauchesne	individual	0793171012	28 Cypress Court	5411
9	Anibal Alsop	individual	0794501014	32 Pine Street	4538
10	Isaura Brumfield	individual	0831651082	62 Devonshire Drive	5621

Figure 6 - Sample data for Vendor table

Customer

The Customer table will be used to store all data associated with any entity that purchases stock from the QAC shop.

```
MariaDB [QAC_SHOP_XQ9X3WV31]> CREATE TABLE Customer_XQ9X3WV31 (
    -> customerID INTEGER NOT NULL PRIMARY KEY AUTO_INCREMENT, /* PK */
    -> name VARCHAR(32),
    -> customerType VARCHAR(32),
    -> phoneNumber VARCHAR(10),
    -> address VARCHAR(32),
    -> postalCode VARCHAR(4)
    -> );
Query OK, 0 rows affected (0.06 sec)
```

Figure 7 - The statements used to create the Customer table

```
MariaDB [QAC_SHOP_XQ9X3WV31]> INSERT INTO Customer_XQ9X3WV31 (name, customerType, phoneNumber, address, postalCode)
    -> VALUES ('Sherise Fagin', 'Indvidual', '0796611068', '76 Brown Street', '6657');
Query OK, 1 row affected (0.01 sec)

MariaDB [QAC_SHOP_XQ9X3WV31]>
MariaDB [QAC_SHOP_XQ9X3WV31]> INSERT INTO Customer_XQ9X3WV31 (name, customerType, phoneNumber, address, postalCode)
    -> VALUES ('Emmett Hausmann', 'Indvidual', '0794201068', '7 King Street', '2433');
Query OK, 1 row affected (0.00 sec)

MariaDB [QAC_SHOP_XQ9X3WV31]>
MariaDB [QAC_SHOP_XQ9X3WV31]> INSERT INTO Customer_XQ9X3WV31 (name, customerType, phoneNumber, address, postalCode)
    -> VALUES ('Margarita Culpepper', 'Indvidual', '0839691029', '46 Route 100', '4234');
Query OK, 1 row affected (0.00 sec)

MariaDB [QAC_SHOP_XQ9X3WV31]>
MariaDB [QAC_SHOP_XQ9X3WV31]> INSERT INTO Customer_XQ9X3WV31 (name, customerType, phoneNumber, address, postalCode)
    -> VALUES ('Owen Hersh', 'Indvidual', '0837591013', '71 Hillside Avenue', '4645');
Query OK, 1 row affected (0.01 sec)

MariaDB [QAC_SHOP_XQ9X3WV31]>
MariaDB [QAC_SHOP_XQ9X3WV31]> INSERT INTO Customer_XQ9X3WV31 (name, customerType, phoneNumber, address, postalCode)
    -> VALUES ('Jean Villani', 'Indvidual', '0799531014', '97 Deerfield Drive', '8978');
Query OK, 1 row affected (0.00 sec)

MariaDB [QAC_SHOP_XQ9X3WV31]>
MariaDB [QAC_SHOP_XQ9X3WV31]> INSERT INTO Customer_XQ9X3WV31 (name, customerType, phoneNumber, address, postalCode)
    -> VALUES ('Luna Mulkey', 'Indvidual', '0838811032', '86 Grove Street', '2343');
Query OK, 1 row affected (0.01 sec)
```

Figure 8 - Example insert statements

customerID	name	customerType	phoneNumber	address	postalCode
1	Joe Soap	Indvidual	0217824465	53 Main Road	7974
2	Anibal Alsop	Indvidual	0794501014	60 Chestnut Street	3454
3	Andre Brumm	Indvidual	0795091027	65 Cross Street	5434
4	Tomoko Ringgold	Indvidual	0831701054	97 Hartford Road	2343
5	Sherise Fagin	Indvidual	0796611068	76 Brown Street	6657
6	Emmett Hausmann	Indvidual	0794201068	7 King Street	2433
7	Margarita Culpepper	Indvidual	0839691029	46 Route 100	4234
8	Owen Hersh	Indvidual	0837591013	71 Hillside Avenue	4645
9	Jean Villani	Indvidual	0799531014	97 Deerfield Drive	8978
10	Luna Mulkey	Indvidual	0838811032	86 Grove Street	2343
11	Mikes B&B	B&B	0834811032	45 Grove Street	2253
12	Bed with Sally	B&B	0838816732	44 Adam Street	2353
13	Vally Stop	B&B	0728841032	67 Heney Street	7746
14	Darell Um	Interior Desig...	0834819364	24 Street West	5536
15	Matthew Rank	Interior Desig...	0835624732	67 New Street	5252
16	Rodolfo Mose	Interior Desig...	0746241032	78 Harrison Avenue	2413

Figure 9 - Sample data for Customer table

Inventory

The Inventory table will keep track of all the QAC Shop's inventory. Inventory can be bought from vendors and sold to customers. The 'itemType' column is used to distinguish between antiques and new items. This allows for the adding of stock from both individuals, wholesalers and distributors.

```
MariaDB [QAC_SHOP_XQ9X3WV31]> CREATE TABLE Inventory_XQ9X3WV31 (
    -> itemID INTEGER NOT NULL PRIMARY KEY AUTO_INCREMENT, /* PK */
    -> itemType VARCHAR(32),
    -> description VARCHAR(128),
    -> pricePerUnit FLOAT,
    -> markup FLOAT,
    -> numberInStock INTEGER,
    -> itemSize INTEGER NULL,
    -> colour VARCHAR(16) NULL,
    -> stockLimit INTEGER,
    -> vendor  INTEGER,
    ->
    -> FOREIGN KEY (vendor) /* FK */
    ->     REFERENCES Vendor_XQ9X3WV31(vendorID)
    ->     ON UPDATE CASCADE ON DELETE CASCADE
    -> );
Query OK, 0 rows affected (0.04 sec)
```

Figure 10 - The statements used to create the Inventory table

```
MariaDB [QAC_SHOP_XQ9X3WV31]> INSERT INTO Inventory_XQ9X3WV31 (itemType, description, pricePerUnit, markup, numberInStock, itemSize, colour, stockLimit,vendor)
    -> VALUES ('antique', 'Chair and table set', 50, 125, 1, null, null, 0, 4);
Query OK, 1 row affected (0.01 sec)

MariaDB [QAC_SHOP_XQ9X3WV31]>
MariaDB [QAC_SHOP_XQ9X3WV31]> INSERT INTO Inventory_XQ9X3WV31 (itemType, description, pricePerUnit, markup, numberInStock, itemSize, colour, stockLimit,vendor)
    -> VALUES ('antique', 'old lamp', 52, 10, 1, null, null, 0, 4);
Query OK, 1 row affected (0.01 sec)

MariaDB [QAC_SHOP_XQ9X3WV31]>
MariaDB [QAC_SHOP_XQ9X3WV31]> INSERT INTO Inventory_XQ9X3WV31 (itemType, description, pricePerUnit, markup, numberInStock, itemSize, colour, stockLimit,vendor)
    -> VALUES ('antique', 'Spanish  coin collection', 60, 30, 1, null, null, 0, 4);
Query OK, 1 row affected (0.01 sec)

MariaDB [QAC_SHOP_XQ9X3WV31]>
MariaDB [QAC_SHOP_XQ9X3WV31]> INSERT INTO Inventory_XQ9X3WV31 (itemType, description, pricePerUnit, markup, numberInStock, itemSize, colour, stockLimit,vendor)
    -> VALUES ('antique', 'wooden lamp', 52, 10, 1, null, null, 0, 4);
Query OK, 1 row affected (0.01 sec)

MariaDB [QAC_SHOP_XQ9X3WV31]>
MariaDB [QAC_SHOP_XQ9X3WV31]> INSERT INTO Inventory_XQ9X3WV31 (itemType, description, pricePerUnit, markup, numberInStock, itemSize, colour, stockLimit,vendor)
    -> VALUES ('new item', 'table cloth', 78, 65, 20, 2, 'black', 5, 1);
Query OK, 1 row affected (0.00 sec)

MariaDB [QAC_SHOP_XQ9X3WV31]>
MariaDB [QAC_SHOP_XQ9X3WV31]> INSERT INTO Inventory_XQ9X3WV31 (itemType, description, pricePerUnit, markup, numberInStock, itemSize, colour, stockLimit,vendor)
    -> VALUES ('new item', 'table cloth', 8, 125, 20, 3, 'green', 5, 1);
Query OK, 1 row affected (0.00 sec)

MariaDB [QAC_SHOP_XQ9X3WV31]>
MariaDB [QAC_SHOP_XQ9X3WV31]> INSERT INTO Inventory_XQ9X3WV31 (itemType, description, pricePerUnit, markup, numberInStock, itemSize, colour, stockLimit,vendor)
    -> VALUES ('new item', 'table cloth', 57, 30, 20, 2, 'white', 5, 1);
Query OK, 1 row affected (0.00 sec)
```

Figure 11 - Example insert statements

itemID	itemType	description	pricePerUnit	markup	numberInStock	itemSize	colour	stockLimit	vendor
1	antique	Victorian coin collection	95	10	1	NULL	NULL	0	3
2	antique	assorted coin collection	52	25	1	NULL	NULL	0	3
3	antique	wizard of Oz original printing	44	45	1	NULL	NULL	0	3
4	antique	vintage car	500	25	1	NULL	NULL	0	4
5	antique	Chair and table set	50	125	1	NULL	NULL	0	4
6	antique	old lamp	52	10	1	NULL	NULL	0	4
7	antique	Spanish coin collection	60	30	1	NULL	NULL	0	4
8	antique	wooden lamp	52	10	1	NULL	NULL	0	4
9	new item	table cloth	78	65	20	2	black	5	1
10	new item	table cloth	8	125	20	3	green	5	1
11	new item	table cloth	57	30	20	2	white	5	1
12	new item	table cloth	20	30	20	5	brown	5	2
13	new item	table cloth	60	45	20	2	brown	1	2

Figure 12 - Example data for the Inventory table

Sale

The Sale table will be used to keep record of all sales made by the QAC Shop. A customer can make multiple sales and purchase multiple items as a part of each sale. The ID number of the sale needs to be used when a customer bill is generated.

```
MariaDB [QAC_SHOP_XQ9X3WV31]> CREATE TABLE Sale_XQ9X3WV31 (
    -> SaleID INTEGER NOT NULL PRIMARY KEY AUTO_INCREMENT, /* PK */
    -> dateOfSale DATE,
    -> customer INTEGER,
    ->
    -> FOREIGN KEY (customer) /* FK */
    ->     REFERENCES Customer_XQ9X3WV31(customerID)
    ->     ON UPDATE CASCADE ON DELETE CASCADE
    -> );
Query OK, 0 rows affected (0.04 sec)
```

Figure 13 - The statements used to create the Sale table

```
MariaDB [QAC_SHOP_XQ9X3WV31]> INSERT INTO Sale_XQ9X3WV31 (dateOfSale, customer)
    -> VALUES ('2011-12-16', 6);
Query OK, 1 row affected (0.01 sec)

MariaDB [QAC_SHOP_XQ9X3WV31]>
MariaDB [QAC_SHOP_XQ9X3WV31]> INSERT INTO Sale_XQ9X3WV31 (dateOfSale, customer)
    -> VALUES ('2015-05-30', 7);
Query OK, 1 row affected (0.01 sec)

MariaDB [QAC_SHOP_XQ9X3WV31]>
MariaDB [QAC_SHOP_XQ9X3WV31]> INSERT INTO Sale_XQ9X3WV31 (dateOfSale, customer)
    -> VALUES ('2011-11-26', 8);
Query OK, 1 row affected (0.01 sec)

MariaDB [QAC_SHOP_XQ9X3WV31]>
MariaDB [QAC_SHOP_XQ9X3WV31]> INSERT INTO Sale_XQ9X3WV31 (dateOfSale, customer)
    -> VALUES ('2010-05-17', 9);
Query OK, 1 row affected (0.01 sec)

MariaDB [QAC_SHOP_XQ9X3WV31]>
MariaDB [QAC_SHOP_XQ9X3WV31]> INSERT INTO Sale_XQ9X3WV31 (dateOfSale, customer)
    -> VALUES ('2001-11-09', 10);
Query OK, 1 row affected (0.01 sec)
```

Figure 14 - Example insert statements

saleID	dateOfSale	customer
1	2011-05-25	1
2	2017-03-05	2
3	2011-06-30	1
4	2010-01-01	1
5	2015-11-30	2
6	2015-11-30	2
7	2013-03-03	3
8	2013-01-25	4
9	2016-02-23	3
10	2013-05-25	5
11	2011-12-16	6
12	2015-05-30	7
13	2011-11-26	8
14	2010-05-17	9
15	2001-11-09	10
16	2003-07-28	11
17	2011-02-25	12
18	2009-12-26	13
19	2014-02-04	14
20	2011-12-23	15
21	2006-11-20	16

Figure 15 - Example data for Sale table

Sale_Item

The Sale_Item table will link the item sold from the Inventory table and the sale that it was made in. According Beynon-Davies (2003) to a link entity is used to cross-reference between an instance of an entity and multiple instances of another entity.

```
MariaDB [QAC_SHOP_XQ9X3WV31]> CREATE TABLE Sale_Item_XQ9X3WV31 (
    -> SaleID INTEGER NOT NULL, /* PK */
    -> itemID INTEGER NOT NULL, /* PK */
    -> numberPurchesed INTEGER,
    ->
    -> PRIMARY KEY(SaleID, itemID),
    ->
    -> FOREIGN KEY (SaleID) /* FK */
    ->     REFERENCES Sale_XQ9X3WV31(SaleID)
    ->     ON UPDATE CASCADE ON DELETE CASCADE,
    ->
    -> FOREIGN KEY (itemID) /* FK */
    ->     REFERENCES Inventory_XQ9X3WV31(itemID)
    ->     ON UPDATE CASCADE ON DELETE CASCADE
    -> );
Query OK, 0 rows affected (0.05 sec)
```

Figure 16 - The statements used to create the Sale_Item table

```
MariaDB [QAC_SHOP_XQ9X3WV31]> INSERT INTO Sale_Item_XQ9X3WV31 (saleID, itemID, numberPurchesed)
    -> VALUES (11, 11, 1);
Query OK, 1 row affected (0.01 sec)

MariaDB [QAC_SHOP_XQ9X3WV31]>
MariaDB [QAC_SHOP_XQ9X3WV31]> INSERT INTO Sale_Item_XQ9X3WV31 (saleID, itemID, numberPurchesed)
    -> VALUES (12, 13, 2);
Query OK, 1 row affected (0.01 sec)

MariaDB [QAC_SHOP_XQ9X3WV31]>
MariaDB [QAC_SHOP_XQ9X3WV31]> INSERT INTO Sale_Item_XQ9X3WV31 (saleID, itemID, numberPurchesed)
    -> VALUES (12, 12, 3);
Query OK, 1 row affected (0.00 sec)

MariaDB [QAC_SHOP_XQ9X3WV31]>
MariaDB [QAC_SHOP_XQ9X3WV31]> INSERT INTO Sale_Item_XQ9X3WV31 (saleID, itemID, numberPurchesed)
    -> VALUES (13, 11, 4);
Query OK, 1 row affected (0.00 sec)

MariaDB [QAC_SHOP_XQ9X3WV31]>
MariaDB [QAC_SHOP_XQ9X3WV31]> INSERT INTO Sale_Item_XQ9X3WV31 (saleID, itemID, numberPurchesed)
    -> VALUES (13, 9, 2);
Query OK, 1 row affected (0.00 sec)

MariaDB [QAC_SHOP_XQ9X3WV31]>
MariaDB [QAC_SHOP_XQ9X3WV31]> INSERT INTO Sale_Item_XQ9X3WV31 (saleID, itemID, numberPurchesed)
    -> VALUES (14, 10, 1);
Query OK, 1 row affected (0.00 sec)

MariaDB [QAC_SHOP_XQ9X3WV31]>
MariaDB [QAC_SHOP_XQ9X3WV31]> INSERT INTO Sale_Item_XQ9X3WV31 (saleID, itemID, numberPurchesed)
    -> VALUES (14, 1, 1);
Query OK, 1 row affected (0.01 sec)

MariaDB [QAC_SHOP_XQ9X3WV31]>
MariaDB [QAC_SHOP_XQ9X3WV31]> INSERT INTO Sale_Item_XQ9X3WV31 (saleID, itemID, numberPurchesed)
    -> VALUES (15, 10, 8);
Query OK, 1 row affected (0.00 sec)
```

Figure 17 - Example insert statements

saleID	itemID	numberPurchesed
1	9	5
2	13	2
3	11	7
4	10	1
5	12	6
5	13	6
6	9	2
6	11	3
7	10	2
7	12	2
7	13	1
8	10	3
9	9	4
10	13	1
11	11	1
12	12	3
12	13	2
13	9	2
13	11	4
14	1	1
14	10	1
15	10	8
16	12	1

Figure 18 - Example data of the Sale_Item table

StockOrder

The ‘StockOrder’ table is used to keep track of all stock orders that have been made. Records will be added to the ‘StockOrder’ table by a trigger that will run once the quantity in stock of a new item drops below a specified level. Once the stock has been delivered the record needs to be removed from the table.

```
MariaDB [QAC_SHOP_XQ9X3WV31]> CREATE TABLE StockOrder (
    -> orderID INTEGER NOT NULL PRIMARY KEY AUTO_INCREMENT, /* PK */
    -> orderDate DATE,
    -> itemID INTEGER,
    ->
    -> FOREIGN KEY (itemID) /* FK */
    ->     REFERENCES Inventory_XQ9X3WV31(itemID)
    ->     ON UPDATE CASCADE ON DELETE CASCADE
    -> );
Query OK, 0 rows affected (0.04 sec)
```

Figure 19- The statements used to create the ‘StockOrder’ table

```
/* Triggers */
delimiter //

CREATE TRIGGER Table2insert AFTER UPDATE ON Sale_XQ9X3WV31_Item
FOR EACH ROW
BEGIN
    UPDATE Inventory_XQ9X3WV31 SET numberInStock = numberInStock - NEW.numberPurchesed WHERE itemID = NEW.itemID;
END; //

delimiter ;
delimiter //

CREATE TRIGGER Table1insert AFTER UPDATE ON Inventory_XQ9X3WV31
FOR EACH ROW
BEGIN
    IF NEW.numberInStock<NEW.stockLimit THEN
        INSERT INTO StockOrder (orderDate, itemID) VALUES (NOW(), NEW.itemID);
    END IF;
END;//

delimiter ;
```

Figure 20 - The triggers used to populate the ‘StockOrder’ table

Ultimately, the triggers are used to indicate when and what stock needs to be reordered.

Queries

The following queries can be used to select all data stored in their respective tables.

Table 7 - Select queries for all tables

Table	Select Query
Vendor	SELECT * FROM Vendor_XQ9X3WV31;
Customer	SELECT * FROM Customer_XQ9X3WV31;
Inventory	SELECT * FROM Inventory_XQ9X3WV31;
Sale	SELECT * FROM Sale_XQ9X3WV31;
Sale_Item	SELECT * FROM Sale_Item_XQ9X3WV31;
StockOrder	SELECT * FROM StockOrder_XQ9X3WV31;

The statements show in Table 8 are example insert statements for all tables in the database.

Table 7 - Example insert statements for all tables

Table	Select Query
Vendor	INSERT INTO Vendor_XQ9X3WV31 (name, vendorType, phoneNumber, address, postalCode) VALUES ('Valint & Balk', 'distributor', '0826776601', '56 Main Road', '7975');
Customer	INSERT INTO Customer_XQ9X3WV31 (name, customerType, phoneNumber, address, postalCode) VALUES ('Vally Stop', 'B&B', '0728841032', '67 Heney Street', '7746');
Inventory	INSERT INTO Inventory_XQ9X3WV31 (itemType, description, pricePerUnit, markup, numberInStock, itemSize, colour, stockLimit, vendor) VALUES ('antique', 'Victorian coin collection', 95, 10, 1, null, null, 0, 3);
Sale	INSERT INTO Sale_XQ9X3WV31 (dateOfSale, customer) VALUES ('2011-05-25', 1);

Sale_Item	INSERT INTO Sale_Item_XQ9X3WV31 (saleID, itemID, numberPurchesed) VALUES (16, 12, 1);
StockOrder	INSERT INTO StockOrder (orderDate, itemID) VALUES ('2001-07-21', 1);

The records of all sales made can be found by select all data from the Sales table and from there the Sale_Item table can be queried to find the exact items sold. Customers can view the availability of a product/all products by calling the checkAvalibility procedure (checkAvalibility_XQ9X3WV31). As indicated in Figure 20, the checkAvalibility procedure will return all items in the QAC Shop's inventory that have one or more instance in stock.

```
delimiter //

CREATE PROCEDURE checkAvalibility_XQ9X3WV31()
BEGIN
    SELECT description, colour, itemSize AS 'size', (pricePerUnit + (pricePerUnit * (markup / 100)))
    AS 'Cost per Unit', numberInStock FROM Inventory_XQ9X3WV31 WHERE numberInStock > 0;
END; //

delimiter ;
```

Figure 21 - The checkAvalibility stored procedure

```
call checkAvalibility_XQ9X3WV31();
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

description	colour	size	Cost per Unit	numberInStock
Victorian coin collection	NULL	NULL	104.5	1
assorted coin collection	NULL	NULL	65	1
wizard of Oz original printing	NULL	NULL	63.8	1
vintage car	NULL	NULL	625	1
Chair and table set	NULL	NULL	112.5	1
old lamp	NULL	NULL	57.2	1
Spanish coin collection	NULL	NULL	78	1
wooden lamp	NULL	NULL	57.2	1
table cloth	black	2	128.7	20
table cloth	green	3	18	20
table cloth	white	2	74.1	20
table cloth	brown	5	26	20
table cloth	brown	2	87	20

Figure 22 - Check availability sample output

A customer's bill can be generated by calling the 'generateBill' procedure (generateBill_XQ9X3WV31). The 'generateBill' procedure accepts the saleID of the sale in question as parameter and will return the total cost of the transaction. The screenshot below show the statements used to create the 'generateBill' procedure.

```
delimiter //

CREATE PROCEDURE generateBill_XQ9X3WV31(_saleID INTEGER)
BEGIN
    SELECT Inventory_XQ9X3WV31.description, Inventory_XQ9X3WV31.itemType, Inventory_XQ9X3WV31.itemSize, Inventory_XQ9X3WV31.colour
    FROM Inventory_XQ9X3WV31 JOIN Sale_Item_XQ9X3WV31 ON Inventory_XQ9X3WV31.itemID = Sale_Item_XQ9X3WV31.itemID WHERE Sale_Item_XQ9X3WV31.saleID = _saleID;

    SELECT SUM(pricePerUnit + (pricePerUnit * (markup / 100))) AS 'Total Cost'
    FROM Inventory_XQ9X3WV31 JOIN Sale_Item_XQ9X3WV31 ON Inventory_XQ9X3WV31.itemID = Sale_Item_XQ9X3WV31.itemID WHERE Sale_Item_XQ9X3WV31.saleID = _saleID;
END; //

delimiter ;
```

Figure 23 - The 'generateBill' stored procedure

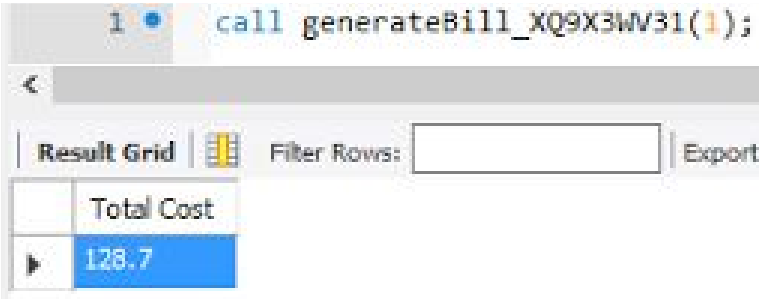


Figure 34 - 'generateBill' example

The 'generateBill' procedure can be called by member of staff of the QAC Shop to get the running bill during a transaction and on completion of the transaction to get the total cost as demonstrated in the screenshot above. Once the bill is generated it is printed out to the client as indicated in the screenshot below.

```
MariaDB [QAC_SHOP_XQ9X3WV31]> call generateBill_xq9x3wv31(5);
+-------------+----------+----------+--------+
| description | itemType | itemSize | colour |
+-------------+----------+----------+--------+
| table cloth | new item |        5 | brown  |
| table cloth | new item |        2 | brown  |
+-------------+----------+----------+--------+
2 rows in set (0.00 sec)

+------------+
| Total Cost |
+------------+
|        113 |
+------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

MariaDB [QAC_SHOP_XQ9X3WV31]>
```

Figure 35 - 'generateBill' procedure called from the command line

Conclusion

An inventory management system has successfully been created for the QAC Shop. The system is able to keep records of sales made, customers can view the availability of stock, inventory can be bought and it can generate a bill. In conclusion, the task of creating an inventory managements system for the QAC Shop was successful.

Bibliography

Beynon-Davies, P. (2004). Database systems. 1st ed. Basingstoke: Palgrave; Macmillan.