Montreal Bookstore Inventory Management System:

Design Documentation

COMP353

By

David Cohen - [redacted]

Rafal Filipiuk - [redacted]

Jeffrey How - [redacted]

Gabriel F. Reyes Baldó - [redacted]

Table of Contents

Background and Assumptions

The purpose of this document is to present the design steps for the MBS Inventory Management System. The following relations will be used:

Book: This entity will hold book-specific information. The original requirements specified that the information should include ISBN, title, author(s), price, subject, quantity-on-hand and year-to-date-qty-sold. Every book published has a unique ISBN, which we have chosen as the primary key.

We have removed quantity-on-hand and year-to-date-qty-sold as attributes for this entity since there is potential for inconsistency and human-error. For example, when a sale of two copies of book X is made, there is no safeguard to ensure that the attribute quantity-on-hand is decreased by exactly two. Instead, an incorrect quantity could be entered by accident, which would result in inconsistency between the number of books held and the number of books documented.

To satisfy this requirement, we use the Sale and Receipt tables (see below) for determining the quantity on hand. Additionally, we use the Sale table to determine year-to-date-qty-sold.

Customer: This entity will hold customer-specific information. The original requirements specified that the information should include name, address, city, province, postal code, phone-number, and cumulative purchases. The primary key we have decided upon includes the name and phone-number attributes, which uniquely identify an individual person. The only exception may be a child using the same phone number as a parent with the same name. But that would imply that they are living together (and would have the same address). Therefore, we believe that name and phone imply address.

We have removed cumulative purchases as an attribute since there is potential for inconsistency and human-error (similar to year-to-date-qty-sold in Book). We use the Purchase table to determine cumulative purchases by a customer.

Employee: This entity set holds employee IDs and names;

MBSBranch: An MBSBranch (Montreal Bookstore Branch) entity has one attribute: name. For example, a name could include “Brossard”, “Downtown”, “Laval”, etc. We assume that when a new branch is created, it will never use the same name as another branch. When an Order or Sale is made, the MBS branch is documented in order to categorize information by branch. We assume that any other information regarding branches (address, phone-number, etc.) is kept external to this database.

Sale: Each sale has a corresponding MBSBranch and Book (strong relationships). For attributes, a sale has a transaction number, a date, a quantity and a clerk. The transaction number helps discriminate a sale (along with its branch and book). A clerk can work at different MBS branches. Hence, he or she can relate to Sales made at different MBSBranches. A sale may or may not have one customer because a customer may or may not want to leave his/her information. When a customer does not leave his or her information, CustomerName and CustomerPhone will be represented with NULL.

Publisher: The attributes for a publisher include its name, id (chosen as primary key), phone and address.

Publishes: This relationship set determines a unique publisher given a particular book.

PBranch: A PBranch represents a Publisher’s branch. Per preliminary requirements, name, representative, address, and phone-number are included as attributes. Its key includes the key of its strong entity (Publisher) and its discriminator, name (of branch).

Order: An Order corresponds to the request of books to a publisher’s branch. An order is made directly to particular branches of the publisher. An “Order” or “Special Order” does not imply a “Sale”.

Each Special Order has a related Customer. Note that the NULL method is used for converting the “isa” relationship of “Special Order”. Therefore, any special orders will be recorded in the order table. Orders

that are not special would have NULL in the attributes identifying a related Customer. This justifies the Customer-related attributes not being in the primary key.

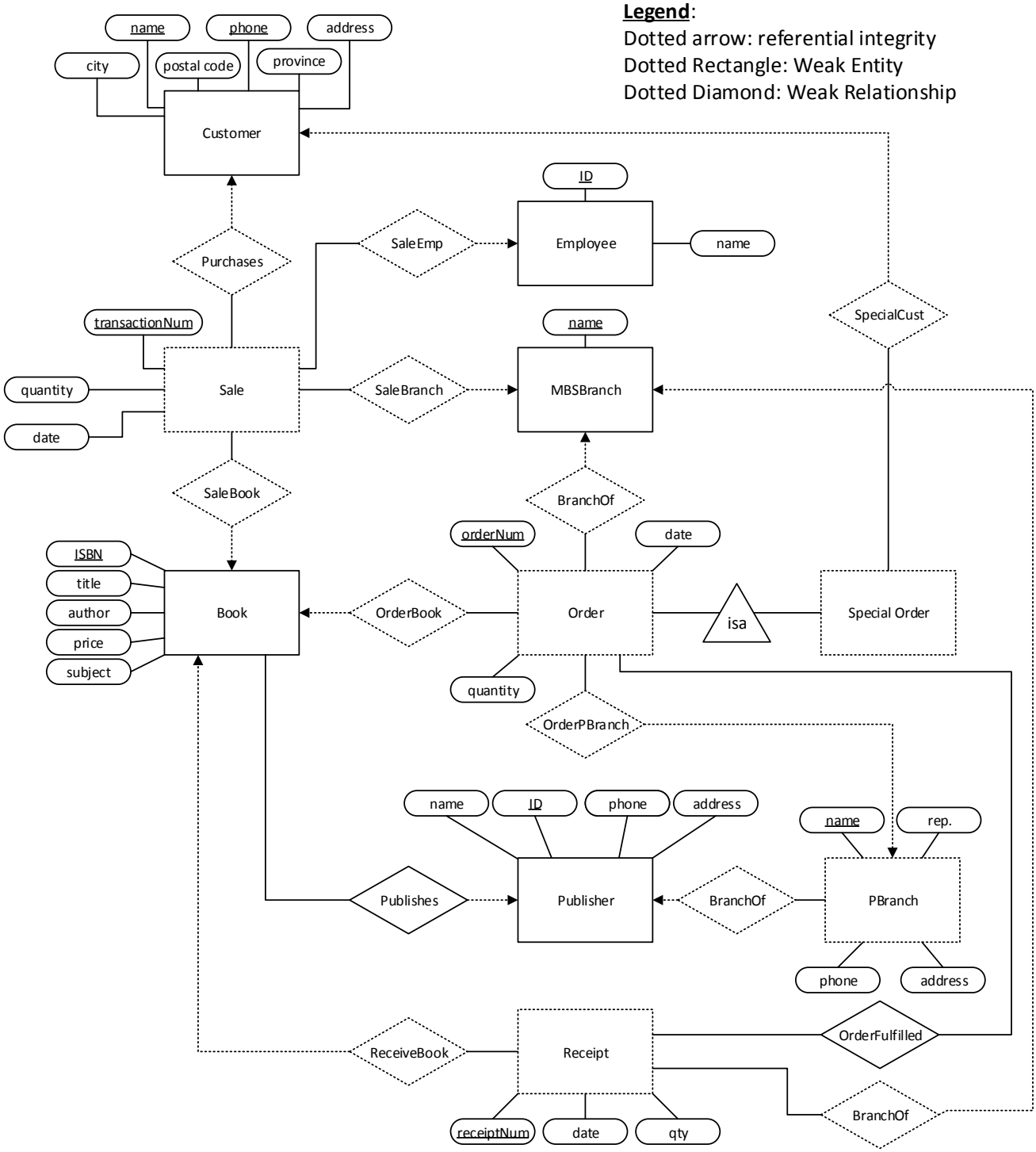
Receipt: A Receipt corresponds to the receipt of goods from a publisher (otherwise known as a shipment). This entity set was made for two reasons. First, it is a comparator to the Orders entity set. Many receipts will correspond to a particular order and their equality in quantities will tell if the order has been fulfilled either partially or fully (See OrderFulfilled, for more information). Second, it helps properly assess the stock (quantity) of a book. When a receipt of books occurs, the quantity inherently increases in information given in receipts. Thus, an assessment of quantity for book X would be Receipts of book X minus Sales of book X.

OrderFulfilled: An order may or may not have a corresponding "Receipt" in the "OrderFulfilled" Relationship Set. The existence of a corresponding receipt implies that the order has been fulfilled (either fully or partially). The lack of a corresponding receipt implies that no shipment has been made from the Publisher's branch.

A receipt may not correspond fully to an "Order" in the "OrderFulfilled" Relationship Set. The existence of a corresponding "Order" implies that the "Receipt" was made in response to a previous order. The lack of a corresponding "Order" could imply exceptions such as donations/goodwill/promotions. Additionally, the existence of a "Receipt" quantity greater than its corresponding "Order" quantity implies that the additional goods were donations or promotional items.

The relation holds information for two different book ISBN's because an "Order" in the "OrderFulfilled" relationship set (which is identified by the book order's ISBN number) might be fulfilled by a different book's ISBN. This might happen in a situation where a different edition (with a different ISBN) is shipped to fulfill a pending order.

E/R Diagram


Legend:
Dotted arrow: referential integrity
Dotted Rectangle: Weak Entity
Dotted Diamond: Weak Relationship
name
phone
address
city
postal code
province
Customer
ID
Employee
name
SaleEmp
Purchases
SpecialCust
transactionNum
name
MBSBranch
Sale
SaleBranch
quantity
date
SaleBook
BranchOf
orderNum
date
ISBN
title
author
price
subject
Book
OrderBook
Order
isa
Special Order
quantity
OrderPBranch
name
ID
phone
address
name
rep.
Publishes
Publisher
BranchOf
PBranch
phone
address
OrderFulfilled
ReceiveBook
Receipt
BranchOf
receiptNum
date
qty

Relational Schema

Schemas: Red → Foreign Keys

Entity sets	Attributes								
Book	ISBN	Title	Authors	Price	Subject				
Publisher	ID	Name	Phone	Address					
Customer	Name	Phone	Address	PostalCode	City	Province			
Employee	ID	Name							
MBS Branch	Name								
Sale	MBSBranchName	ISBN	TransactionNumber	ClerkID	DateOfSale	CustName	CustPhone	Quantity	
Orders	ISBN	MBSBranchName	OrderNumber	PublisherID	PublisherBranchName	CustName	CustPhone	Quantity	DateOfOrder
Receipt	ISBN	MBSBranchName	ReceiptNumber	ReceiptDate	Quantity				
Pbranch	PublisherID	Name	Representative	Phone	Address				

Relationship sets	Attributes				
Orders Fulfilled	ReceiptISBN	OrderISBN	ReceiptNumber	OrderNumber	MBSBranchName
Publishes	ISBN	ID			

Functional Dependencies

Table	Functional Dependencies
Book	{(ISBN) --> (Title, Authors, Price, Subject)}
Publisher	{(ID) --> (Name, Phone, Address)}
Customer	{(name, phone) --> (address, city, postal code, province)}
Employee	{(ID) --> (Name)}
MBSBranch	{}
Sale	{(MBSBranchName, ISBN, TransactionNumber)-->(ClerkID, Date, CustomerName, CustomerPhone, Quantity)}
Order	{(ISBN, MBSBranchName, OrderNumber) --> (PublisherID, PublisherBranchName, DateOfOrder, CustName, CustPhone, Quantity)}
Receipt	{(ISBN, MBSBranchName, ReceiptNumber) --> (ReceiptDate, Quantity)}
Pbranch	{(PublisherID, Name)-->(Representative, Phone, Address)}
Orders Fulfilled	{}
Publishes	{(ISBN)-->(ID)}

Data Types

Table	Attribute	Type	Assumption
Book	ISBN	bigint[10] unsigned	ISBN length between 10 and 13 digits
	Title	varchar[25]	
	Authors	varchar[25]	Can have many authors
	Price	decimal[8,2] unsigned	Maximum price of $99k
	Subject	varchar[25]	
Publisher	ID	bigint[10] unsigned	Billion unique ID's
	Name	varchar[25]	
	Phone	bigint[10] unsigned	No country codes
	Address	varchar[255]	
Customer	Name	varchar[25]	
	Phone	bigint[10] unsigned	No country codes
	Address	varchar[255]	
	Postal Code	varchar[7]	No whitespace
	City	varchar[25]	
	Province	varchar[25]	Province abbreviations used
Employee	ID	int[3] unsigned	
	Name	varchar[25]	
MBS Branch	Name	varchar[25]	
Sale	MBSBranchName	varchar[25]	
	ISBN	bigint[10] unsigned	
	TransactionNumber	bigint[9] unsigned	
	ClerkID	int[3] unsigned	
	DateOfSale	date	
	Quantity	int[4] unsigned	
	CustomerName	varchar[25]	
	CustomerPhone	bigint[10] unsigned	
Orders	ISBN	bigint[10] unsigned	
	PublisherID	bigint[10] unsigned	
	PublisherBranchName	varchar[40]	
	MBSBranchName	varchar[25]	
	CustomerName	varchar[25]	
	CustomerPhone	bigint[10] unsigned	
	OrderNumber	bigint[9] unsigned	
	DateOfOrder	date	
	Quantity	int[4] unsigned	Max order of 10,000
Receipt	ISBN	bigint[10] unsigned	
	MBSBranchName	varchar[25]	
	ReceiptNumber	bigint[9] unsigned	
	ReceiptDate	date	
	Quantity	int[4] unsigned	Max shipment of 10,000
Pbranch	PublisherID	bigint[10] unsigned	
	Name	varchar[40]	
	Representative	varchar[25]	
	Phone	bigint[10] unsigned	No country codes
	Address	varchar[255]	
Orders Fulfilled	ReceiptISBN	bigint[10] unsigned	
	OrderISBN	bigint[10] unsigned	
	MBSBranchName	varchar[255]	
	ReceiptNumber	bigint[9] unsigned	
	OrderNumber	bigint[9] unsigned	
Publishes	ISBN	bigint[10] unsigned	
	ID	bigint[10] unsigned	

Implementation

Steps to implementation were as follows:

1. Create SQL queries for each requirement
2. In PHP, program the interface of each requirement using the SQL queries as a guide.

To provide an understanding of our implementation, the following section will provide example SQL queries and screenshots of the interfaces. Note that due to different parameters that the user may choose, every combination of each query is not presented. Instead, one example gives a general idea behind out thought-process. Also, keep in mind that the SQL queries were *guidelines* for the PHP. Some queries may have been slightly altered to convert to proper PHP code.

Requirement 1: MAIN - Create suitable data entry forms for manipulating the tables (at least for the entity sets: Employees, Books, and Customers), that supports operations such as insertion, deletion, and update of any attributes except the primary keys (for the simplicity).

SQL queries include basic queries displaying the chosen table; as well as basic INSERT, DELETE and UPDATE queries for manipulating tables.

As seen above, any updateable fields have textboxes. The keys cannot be updated (highlighted in red). To insert an entry, the first line of empty textboxes can be used. Afterwards, entry validation is performed. To delete an entry, hover your mouse over the entry and an ‘X’ button will appear to the right.

Requirement 2: BOOKSEARCH - Create an interface to retrieve the information about all or specific books from any branch as well as the entire MBS based on the information: author, title, subject, publisher, and ISBN.

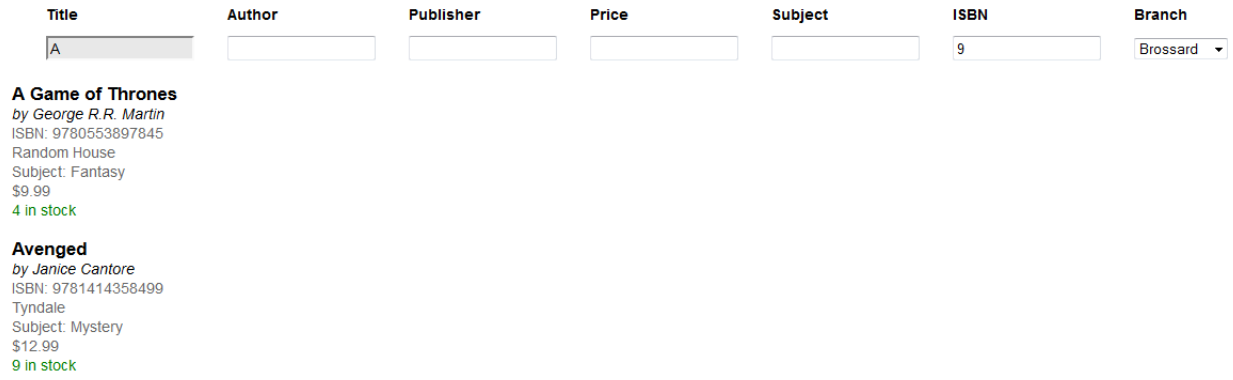
SQL queries relate to Book table (as well as the Publishes and Publisher tables). Mainly the queries involve selectively finding tuples based upon the input of the user. Depending on the input information, conditionals in the WHERE statement will change.

Example query for obtaining quantity of Books:

```
(SELECT Rec.ISBN, (RecQty - SalQty) AS Qty
FROM
(SELECT ISBN, SUM(Quantity) AS RecQty
 FROM Receipt
 WHERE Receipt.MBSBranchName = 'Brossard'
 GROUP BY ISBN)Rec,
(SELECT ISBN, SUM(Quantity) AS SalQty
 FROM Sale
 WHERE Sale.MBSBranchName = 'Brossard'
 GROUP BY ISBN)Sal
WHERE Rec.ISBN = Sal.ISBN)

UNION

(SELECT ISBN, Quantity AS Qty
 FROM Receipt
 WHERE MBSBranchName = 'Brossard' AND
 ISBN NOT IN (SELECT ISBN FROM Sale WHERE MBSBranchName ='Brossard'));
```



The above interface filters books according to user-defined criteria using the textboxes at the top of the screen. In this example, we see the stock in Brossard of books that contain the letter ‘A’ and have an ISBN containing the digit ‘9.’

Requirement 3: CUSTOMER ORDERS - List of all back orders made by Customers group by customer, the book's subject, and branch as well as the information of the customers who made the order.

Example query:

```
SELECT  Name, Subject, MBSBranchName AS Branch, OrderNumber AS 'Order #',
        DateOfOrder AS Date, Title, Quantity AS Qty, Phone, Address,
        PostalCode, City, Province
FROM    Customer, Orders, Book
WHERE   CustomerName = Name AND
        CustomerPhone = Phone AND
        Book.ISBN = Orders.ISBN
        GROUP BY MBSBranchName, Name, Subject;
```

Name	Title	Subject	Quantity	Branch	Order Date	Order
Anna Queen	The End of Diabetes	Health	1	Brossard	2010-01-04	5
Buck Naykitt	The Hobbit	Fantasy	1	Brossard	2010-01-04	2
Dixie Normous	Fifty Shades of Grey	Romance	2	Brossard	2010-03-11	24
Elvis Abdul	Sweet Tea Revenge	Romance	2	Brossard	2010-01-07	9
Jacques Strap	Calculated in Death	Thriller	2	Brossard	2010-01-28	20
Mike Rotch	Shred: The Revolutionary	Health	1	Brossard	2010-02-13	21
Raul Soules	11th Hour	Mystery	1	Brossard	2010-03-18	25
Sam Willis	The End of Diabetes	Health	2	Brossard	2010-02-22	23
	Team of Rivals	Political	1	Brossard	2010-02-22	22
Stephanie Johnson	World War Z	Horror	1	Brossard	2013-04-03	27
Fredrick Sabal	Divergent	Children	2	Downtown	2011-04-01	21

Requirement 4: TOP CUSTOMERS - List of top five Customers of each branch as well as the entire MBS defined as those customers having the highest amount of cumulative purchase in the last month (say if we are in April, do it for March).

Example query: Top 5 customers with most purchases last month. (All branches)

```
SELECT  CustomerName, CustomerPhone, COUNT(Quantity) AS Purchases
FROM    Sale
WHERE   YEAR(DateOfSale) = YEAR(CURDATE() - INTERVAL 1 MONTH) AND
        MONTH(DateOfSale) = MONTH(CURDATE() - INTERVAL 1 MONTH) AND
        CustomerName IS NOT NULL GROUP BY CustomerName, CustomerPhone
ORDER BY COUNT(Quantity) DESC LIMIT 5;
```

Branch

Laval

Name	Phone #	# of Books Purchased
Buck Naykitt	8672345556	2
Jacques Strap	2042621891	1
Jane Doe	5143229856	1
Mariana Malaret	6042395116	1
Mark Plithkin	5143214569	1

Requirement 5: CLERKS Per branch or per all branches: list the total amount of sales for a given sales clerk.

Example Query: Number of sales made by clerks for branch CHOSENBRANCH

```
SELECT  Name, COUNT(DISTINCT TransactionNumber) AS 'Number Of Sales'
FROM    Sale, Employee
WHERE   ClerkID = ID AND
        MBSBranchName LIKE 'CHOSENBRANCH'
GROUP BY Name;
```

Branch **Clerk ID**

Brossard 3 Search

Name	# of Sales
Martin C.	22

Requirement 6: Sales Details - Create an interactive form for all the mangers to list weekly, monthly, or yearly sales details, quantity and amount, group by either ISBN, title, subject, and/or publisher for each branch as well as the entire MBS.

Example Query: Show quantity and amount of books sold by year.

```
SELECT YEAR(DateOfSale), Title, SUM(Quantity), SUM(Amount)
FROM
(SELECT Sale.ISBN AS ISBN, Title, Sale.MBSBranchName AS Branch, Subject,
        Name, DateOfSale, Quantity ,(Quantity*Price) AS Amount
 FROM   Sale, Book, Publisher, Publishes
 WHERE  Sale.ISBN = Publishes.ISBN AND Sale.ISBN = Book.ISBN AND
        ID = PublisherID)SalesData
WHERE SalesData.Branch LIKE 'Downtown'
GROUP BY YEAR(DateOfSale), Title
ORDER BY YEAR(DateOfSale), Title;
```

Branch	Range	Group By
All	Yearly	Title

Year	Title	Quantity	Amount
2010	11th Hour	9	$134.91
2010	A Game of Thrones	1	$9.99
2010	Avenged	1	$12.99
2010	C.S. Lewis: A Life	7	$174.93
2010	Calculated in Death	3	$23.97
2010	Divergent	7	$76.93
2010	Dork Diaries 5	6	$95.94
2010	Fifty Shades of Grey	15	$239.25
2010	Frost Burned	4	$107.80
2010	How to Get Filthy Rich	1	$26.95
2010	Shred: The Revolutionary	6	$149.94
2010	Team of Rivals	1	$20.95
2010	The End of Diabetes	10	$299.90
2010	The Hobbit	6	$95.88
2010	World War Z	8	$208.00
2011	11th Hour	7	$104.93
2011	A Game of Thrones	5	$49.95
2011	Avenged	6	$77.94
2011	C.S. Lewis: A Life	4	$99.96
2011	Calculated in Death	11	$87.89
2011	Divergent	11	$120.89
2011	Dork Diaries 5	3	$47.97

Requirement 7: Pending Orders - Receive orders in the warehouse: A warehouse employee would enter the fact that a given order has been received (identified by a unique order number). In this case, the applications program should make the appropriate changes to the database indicating that additional book(s) are now in stock.

Example Query: Display all pending orders

```
(SELECT Orders.MBSBranchName AS Branch, Orders.DateOfOrder AS Date,
        Orders.OrderNumber AS 'Order #', Orders.ISBN,
        ((Orders.Quantity) - SUM(Receipt.Quantity)) AS Qty
 FROM   Orders, OrdersFulfilled, Receipt
 WHERE  OrdersFulfilled.ReceiptISBN = Receipt.ISBN AND
        OrdersFulfilled.ReceiptNumber = Receipt.ReceiptNumber AND
        OrdersFulfilled.MBSBranchName LIKE Receipt.MBSBranchName AND
        OrdersFulfilled.OrderISBN = Orders.ISBN AND
        OrdersFulfilled.OrderNumber = Orders.OrderNumber AND
        OrdersFulfilled.MBSBranchName LIKE Orders.MBSBranchName
        GROUP BY Orders.MBSBranchName, Orders.OrderNumber, Orders.ISBN, Orders.Quantity
        HAVING Orders.Quantity - SUM(Receipt.Quantity) >0)

UNION

(SELECT MBSBranchName, DateOfOrder,OrderNumber, ISBN, Quantity
 FROM   Orders
 WHERE  (Orders.MBSBranchName, Orders.OrderNumber) NOT IN
        (SELECT MBSBranchName, OrderNumber FROM OrdersFulfilled));
```

Branch	Order #	ISBN	Quantity	Date	
Plateau	Choose an order #				Add

Branch	Date	Order #	ISBN	Quantity
Plateau	2011-10-03	15	9780441020010	2
Plateau	2011-10-13	16	9780441020010	1
Plateau	2011-10-15	17	9780441020010	1
Plateau	2011-10-03	14	9781414339351	2
Plateau	2011-09-03	12	9781414358499	1
Plateau	2011-09-03	13	9781414358499	1
Plateau	2011-09-02	11	9781847242532	1

This bottom half of this interface displays all pending orders for all branches or by branch (selected by the Branch drop-down box).

If a receipt of goods has come in, the warehouse worker will select the corresponding order and book. He will then enter the quantity that has been received as well as the receipt date. Then the table should update according as books are received. Quantities received which are greater than ordered amounts are assumed to include promotional goods or donations from the publisher.

Additional Features

Foreign Key Constraints

Foreign keys references were used to ensure referential integrity. For example:

```
Sale  | CREATE TABLE `Sale` (
`MBSBranchName` varchar(25) NOT NULL DEFAULT '',
`ISBN` bigint(10) unsigned NOT NULL DEFAULT '0',
`TransactionNumber` bigint(9) unsigned NOT NULL DEFAULT '0',
`ClerkID` int(3) unsigned NOT NULL,
`DateOfSale` date NOT NULL,
`Quantity` int(4) unsigned NOT NULL DEFAULT '0',
`CustomerName` varchar(25) DEFAULT NULL,
`CustomerPhone` bigint(10) unsigned DEFAULT NULL,
PRIMARY KEY (`MBSBranchName`,`ISBN`,`TransactionNumber`),
KEY `ISBN` (`ISBN`),
KEY `TransactionNumber` (`TransactionNumber`),
KEY `CustomerName` (`CustomerName`,`CustomerPhone`),
KEY `ClerkID` (`ClerkID`),
CONSTRAINT `Sale_ibfk_4` FOREIGN KEY (`ClerkID`) REFERENCES `Employee` (`ID`),
CONSTRAINT `Sale_ibfk_1` FOREIGN KEY (`ISBN`) REFERENCES `Book` (`ISBN`) ON UPDATE CASCADE,
CONSTRAINT `Sale_ibfk_2` FOREIGN KEY (`MBSBranchName`) REFERENCES `MBSBranch` (`Name`) ON UPDATE CASCADE,
CONSTRAINT `Sale_ibfk_3` FOREIGN KEY (`CustomerName`, `CustomerPhone`) REFERENCES `Customer` (`Name`, `Phone`)
ENGINE=InnoDB DEFAULT CHARSET=latin1 |
```

This example shows the foreign keys used in the Sale entity set. For more information about our foreign keys, please refer to our Relational Design Scheme.

Triggers Constraints: Auto-incrementing transactions numbers in Sale

We have a trigger, which auto-increments transaction numbers of sale should an invalid transaction number be input. For example, a Brossard sale's transaction number cannot be 15 if there exists no previous Brossard sale with transaction number 14. An almost identical trigger is used for order numbers.

Additionally, if there are two Sale entries with the same transaction number but different ISBNS (which represents two books being sold in the same transaction), this trigger ensures that they are given the same date.

```
CREATE TRIGGER trxCnt
BEFORE INSERT ON Sale
FOR EACH ROW
BEGIN
	DECLARE maxTrx BIGINT;
	SET maxTrx = (  SELECT MAX(TransactionNumber)
			FROM Sale
			WHERE MBSBranchName = NEW.MBSBranchName);
	IF maxTrx IS NOT UNKNOWN THEN
		IF NEW.TransactionNumber < maxTrx OR NEW.TransactionNumber > maxTrx +1 THEN
			SET NEW.TransactionNumber = maxTrx +1;
		END IF;
	ELSE SET NEW.TransactionNumber = 1;
	END IF;

	IF (NEW.TransactionNumber,NEW.MBSBranchName) IN (SELECT TransactionNumber, MBSBranchName FROM Sale) THEN
		SET NEW.DateOfSale = (SELECT DateOfSale FROM Sale WHERE TransactionNumber = NEW.TransactionNumber AND MBSBranchName = NEW.MBSBranchName ORDER BY
				      DateOfSale LIMIT 1);
	END IF;
END;
```

Additional Web Features

- Dynamic inserting / updating / deleting of entries on Main page (updating done merely by editing field and changing focus)
- 'Order By' display on Main page changed with a simple click on whichever field
- Real-time searching on Book Search page
- Branch select dropdown box dynamically generated with SQL query on all pages
- Hidden Admin page used for inputting data in tables that are not necessarily updateable via the required interfaces.
- Shortcut keys to different interfaces.

Work Distribution

Work Distribution	
David Cohen	Web Development - Lead
Rafal Filipiuk	Database Design, Data Entry, Constraints
Jeffrey How	Database Design, SQL Queries, Constraints
Gabriel Reyes	Data Entry, Web Development, Constraints

Accessing the Website

URL: [redacted]

Login: [redacted]

Password: [redacted]