**CSE 3241 Final Project – The Final Report**

**Section 1. Database Description**

1. ER diagram

Diagram

Description automatically generated

1. Relational schema (with functional dependencies)

A picture containing table

Description automatically generated

Timeline

Description automatically generated

**Note**: GENRE and WRITES have no functional dependencies

1. Level of normalization

BOOK: BCNF

GENRE: BCNF

AUTHOR: BCNF

WAREHOUSE: BCNF

CUSTOMER: BCNF

PUBLISHER: BCNF

ORDERS: BCNF

CREDIT\_CARD: BCNF

PAYPAL: BCNF

STORES: BCNF

WRITES: BCNF

ADDED\_TO: BCNF

REFUND: BCNF

1. Index

Publisher’s name: This index is created on the BOOK table (indexing by the same publisher)

* Each publisher may require which books are currently sold in our bookstore.

Book’s price: This index is created on the BOOK table (indexing by the price range)

* Customers want to buy books within their budget, so they want to view them in a specific price range.

Warehouse’s location: This index is created on the WAREHOUSE table (indexing by the exact location)

* The bookstore can determine which books are in each location's warehouse for even faster delivery.

Book’s genre: This index is created on the GENRE table (indexing by the same genre)

* Customers want to look up their interesting genre’s books.

Book’s ordered quantity: This index is created on the ADDED\_TO table (indexing by the quantity range)

* The bookstore wants to check which books are sold mostly.

1. View

retrieve the number of sold books by each genre

* Based on this result, we(bookstore) can buy more books for popular genre and promote for less popular genre books.

CREATE VIEW VIEW1 AS

SELECT G.Genre, SUM(AT.Quantity) AS num\_of\_sold\_books

FROM GENRE AS G, ADDED\_TO AS AT

WHERE G.B\_isbn=AT.B\_isbn

GROUP BY G.Genre

ORDER BY num\_of\_sold\_books DESC;

Table

Description automatically generated

retrieve one specific customer (e.g., Edwin5) favorite genres

* Based on this result, we can send e-mails promoting books in genres that consumers are interested in.

CREATE VIEW VIEW2 AS

SELECT G.Genre, COUNT(AT.B\_isbn) AS number\_of\_books

FROM ORDERS AS O, ADDED\_TO AS AT, GENRE AS G

WHERE O.Order\_num=AT.O\_order\_num

AND AT.B\_isbn=G.B\_isbn

AND O.Customer\_username='Edwin5'

GROUP BY G.Genre

ORDER BY number\_of\_books DESC;

Table

Description automatically generated

retrieve book's ISBN with less than 100 in stock in each warehouse in Columbus

* Based on the result, we can decide which books are needed to be restocked.

CREATE VIEW VIEW3 AS

SELECT AT.B\_isbn AS ISBN, W.Id AS warehouse\_id, (SUM(S.Quantity)-SUM(AT.Quantity)) AS remaining\_stock

FROM STORES AS S, ADDED\_TO AS AT, BOOK AS B, WAREHOUSE AS W

WHERE B.Isbn=AT.B\_isbn

AND S.B\_isbn=AT.B\_isbn

AND W.Id=S.W\_id

AND W.Location = 'Columbus'

GROUP BY S.B\_isbn

HAVING remaining\_stock < 100

ORDER BY W.Id;

Table

Description automatically generated

1. Transaction

Adds a specific book to order, updating quantity and capacity in a specific warehouse.

BEGIN TRANSACTION addBookToOrder

INSERT INTO ADDED\_TO VALUES (IsbnNum, orderNum, quantityGiven);

IF error THEN GO TO UNDO; END IF;

UPDATE STORES

SET Quantity = Quantity - quantityGiven

WHERE B\_Isbn = IsbnNum

AND W\_id = specificWarehouseID;

IF error THEN GO TO UNDO; END IF;

UPDATE WAREHOUSE

SET Capacity = Capacity + quantityGiven

WHERE W\_id = specificWarehouseID;

IF error THEN GO TO UNDO; END IF;

COMMIT;

GO TO FINISH;

UNDO:

ROLLBACK;

FINISH:

END TRANSACTION;

Add/writes a card payment into an order.

BEGIN TRANSACTION addCardPayment

INSERT INTO CREDIT\_CARD VALUES (Number1, Company1, Cvv1, Name1, Bill\_Address1);

IF error THEN GO TO UNDO; END IF;

IF COUNT(Number1) > 1 THEN GO TO UNDO; END IF;

UPDATE ORDER

SET Creditcard\_number = Number1

WHERE order\_num = specificOrderNum;

IF error THEN GO TO UNDO; END IF;

COMMIT;

GO TO FINISH;

UNDO:

ROLLBACK;

FINISH:

END TRANSACTION;

Adds a new book into a specific warehouse and updates quantity and capacity.

BEGIN TRANSACTION addBookToWarehouse

INSERT INTO BOOK VALUES (newIsbn, newTitle, newYear, newGenre, newPrice);

IF error THEN GO TO UNDO; END IF;

IF COUNT(Number1) > 1 THEN GO TO UNDO; END IF;

UPDATE STORES

SET Quantity = Quantity + addedQuantity

WHERE B\_isbn = newIsbn AND W\_id = specificWarehouse;

IF error THEN GO TO UNDO; END IF;

UPDATE WAREHOUSE

SET capacity = capacity - addedQuantity

WHERE ID = specificWarehouse;

IF error THEN GO TO UNDO; END IF;

COMMIT;

GO TO FINISH;

UNDO:

ROLLBACK;

FINISH:

END TRANSACTION;