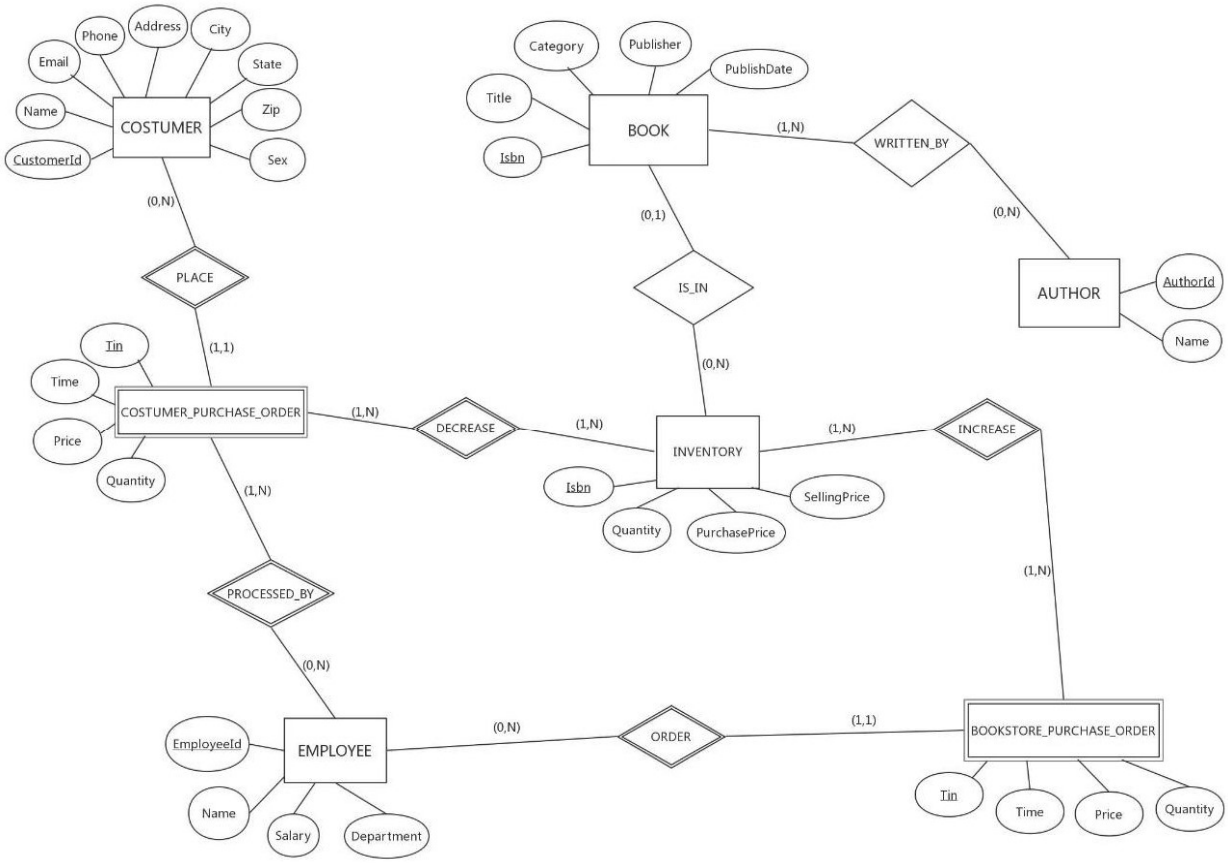
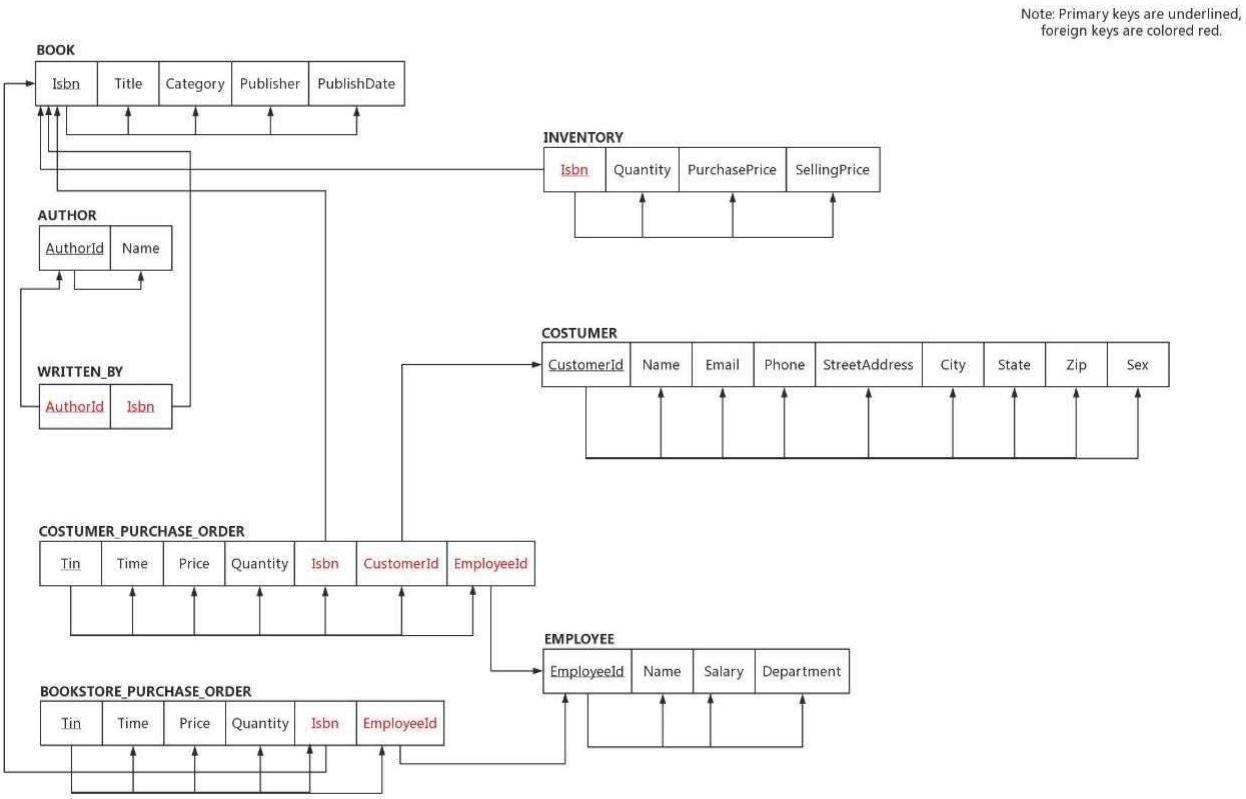
Database Description

ER DIAGRAM



RELATIONAL SCHEMA



DESCRIPTION OF NORMALIZATION

Table Name	Level of Normalization	Reason
BOOK	BCNF	BOOK is not only 3NF, but also ISBN → (Any other attribute) in BOOK, so ISBN is a superkey of BOOK.
AUTHOR	BCNF	AUTHOR is not only 3NF, but also AuthorID → Name in AUTHOR, then AuthorID is a superkey of AUTHOR.
WRITTEN_BY	BCNF	N/A
EMPLOYEE	BCNF	EMPLOYEE is not only 3NF, but also EmployeeID → (Any other attribute) in EMPLOYEE, then EmployeeID is a superkey of EMPLOYEE.
INVENTORY	BCNF	INVENTORY is not only 3NF, but also Isbn → (Any other attribute) in INVENTORY, then ISBN is a superkey of INVENTORY.
CUSTOMER	BCNF	CUSTOMER is not only 3NF, but also if CustomerId → (Any other attribute) in CUSTOMER, then CustomerId is a superkey of CUSTOMER.
COSTUMER_PURCHASE_ORDER	BCNF	COSTUMER_PURCHASE_ORDER is not only 3NF, but also if Tin → (Any other attribute) in COSTUMER_PURCHASE_ORDER, then Tin is a superkey of COSTUMER_PURCHASE_ORDER.
BOOKSTORE_PURCHASE_ORDER	BCNF	BOOKSTORE_PURCHASE_ORDER is not only 3NF, but also if Tin → (Any other attribute) in BOOKSTORE_PURCHASE_ORDER, then Tin is a superkey of BOOKSTORE_PURCHASE_ORDER.

DESCRIPTION OF INDEXING

It is assumed that the database automatically applies indexing on the Primary Key of each relation. Below is an additional index that has been included in the database.

```
CREATE UNIQUE INDEX ID_RETRIEVE ON CUSTOMER_PURCHASE_ORDER (CUSTOMERID, EMPLOYEEID);
```

This index aims to promote the efficiency of retrieving for both CustomerId and EmployeeId, as these two attributions frequently used by the queries to join different tables; It is a Hash-based index because Hash-based is good at equality tests. The hash function would convert the text as the special hash value and then put it into hashtable. This process is reversed when the text needs to be retrieved.

DESCRIPTION OF VIEWS

View 1: Select and show the ISBN, title and total number of sold copies of books that sold more than 5 copies.

$$\pi_{Isbn,\ Title,\ sum(Quantity)}(\sigma_{sum(Quantity)>5}\ (_{Isbn,\ Title}F_{SUM\ Quantity}(BOOK * CUSTOMER\_PURCHASE\_ORDER)))$$

```
CREATE VIEW BOOKS_SOLD_OVER_5
AS      SELECT B.Isbn, B.Title, sum(P.Quantity)
FROM BOOK B, CUSTOMER_PURCHASE_ORDER P
WHERE B.Isbn = P.Isbn
GROUP BY B.Isbn, B.Title
HAVING sum(P.Quantity) > 5;
```

Sample Output

ISBN	Title	sum(P.Quantity)
0060958332	THE LANGUAGE INSTINCT: HOW THE MIND CREATES LANGUAGE	10
0130323519	FINANCIAL REPORTING AND ANALYSIS	6
0375508325	COSMOS	8
0393318486	HOW THE MIND WORKS	7
0596004478	GOOGLE HACKS	10

View 2: Provide the list of Customers who purchased more than 3 books with the CustomerId, name and total number of books purchased by the customer.

$\pi_{CustomerId,\ Name,\ sum(Quantity)}(\sigma_{sum(Quantity)>3}\ (_{CustomerId,\ Name}F_{SUM\ Quantity}(CUSTOMER * CUSTOMER\_PURCHASE\_ORDER)))$

```
CREATE VIEW CUSTOMERS_PURCHASE
AS      SELECT C.CustomerId, C.Name, sum(P.quantity)
FROM CUSTOMER C, CUSTOMER_PURCHASE_ORDER P
WHERE C.CustomerId = P.CustomerId
GROUP BY C.CustomerId, C.Name
HAVING sum(P.Quantity) > 3;
```

Sample Output

CustomerId	Name	sum(P.quantity)
478440894	CENGLIN BAO	6
161408684	JENNIFER GARNER	10
252732007	KANYE WEST	15
307704140	KIM KARDASHIAN	6
694132332	MARILYN MONROE	5

View 3: Provide the list of Employees who sold more than 10 books with the EmployeeId, name, salary and total number of books sold by the employee.

$\pi_{EmployeeId,\ Name,\ Salary,\ sum(Quantity)}(\sigma_{sum(Quantity)>10}\ (_{EmployeeId}F_{SUM\ Quantity}(EMPLOYEE * CUSTOMER\_PURCHASE\_ORDER)))$

```
CREATE VIEW EMPLOYEE_SOLD
AS      SELECT E.EmployeeId, E.Name, E.SALARY, sum(P.Quantity)
FROM EMPLOYEE E,  CUSTOMER_PURCHASE_ORDER P
WHERE E.EmployeeId = P.EmployeeId
GROUP BY E.EmployeeId
HAVING sum(P.Quantity) > 10;
```

Sample Output

EmployeeId	Name	Salary	sum(P.Quantity)
100935428	PEDRO SANCHEZ	40000	13
244618185	CRISTIANO RONALDO	38000	16
282149295	DAVE CHAPELLE	41000	12
339490835	BENJAMIN FRANKLIN	34000	12
838890111	WILLIAM SHAKESPEARE	37500	12

DESCRIPTION OF TRANSACTIONS

Transaction 1: Create new customer.

```
BEGIN TRANSACTION NEW_CUSTOMER

    INSERT INTO CUSTOMER VALUES ('945120271', 'NAPOLEON
    DYNAMITE', 'VOTE4PEDRO@GMAIL.COM', '9141024200', '7839
    MIDDLEOFNOWHERE ST', 'BUFFALO', 'NY', '14201', 'M');
        IF error THEN GO TO UNDO; END IF;
        COMMIT;
        GO TO FINISH;
    UNDO:
        ROLLBACK;
    FINISH:
END TRANSACTION;
```

Transaction 2: Create new bookstore purchase order on books with low inventory and update inventory information.

```
BEGIN TRANSACTION BOOK_STORE_PURCHASE_ORDER_FOR LOW_INVENTORY
    INSERT INTO BOOKSTORE_PURCHASE_ORDER
    VALUES('897991245', '2020-01-19 10:03:34', '6.99', '7',
    '0345452089', '336872945');
        IF error THEN GO TO UNDO; END IF;
    UPDATE INVENTORY
        SET Quantity = Quantity + 7, Purchase_price = 6.99
        WHERE ISBN = '0345452089';
        IF error THEN GO TO UNDO; END IF;
        COMMIT;
        GO TO FINISH;
    UNDO:
        ROLLBACK;
    FINISH:
```

```
END TRANSACTION;
```

Transaction 3: Update customer purchase order information.

```
BEGIN TRANSACTION UPDATE_CUSTOMER_PUCHASE_ORDER
        UPDATE INVENTORY
                SET QUANTITY = QUANTITY + 2
                WHERE Isbn =  '0596004478';
                IF error THEN GO TO UNDO; END IF;
        UPDATE BOOKSTORE_PURCHASE_ORDER
                SET Time = '2020-03-17 12:17:02', Price = '9.99', Quantity
                ='5';Isbn = '0061020656'
                WHERE Tin =  '442188895';
                IF error THEN GO TO UNDO; END IF;
        UPDATE INVENTORY
                SET Quantity = Quantity - 5, Selling_price = '9.99'
                WHERE ISBN = '0061020656';
                IF error THEN GO TO UNDO; END IF;
                COMMIT;
                GO TO FINISH;
        UNDO:
                ROLLBACK;
        FINISH:
END TRANSACTION;
```