

Functional dependency

CUSTOMER

{Email} → {Interests, Birth\_date, LName, FName, Gender, Phone, CA\_id, CusLicense\_no}

{CusLicense\_no} → {Email, Interests, Birth\_date, LName, FName, Gender, Phone, CA\_id, CusLicense\_no}

CUSTOMER\_SERVICE

{Service\_Ssn} → {Typing\_speed}

REQUEST\_SERVICE

{Service\_code} → {SEmail, Request\_Ssn}

PAYMENT

{POrder.no,Card\_number} → {Amount,Billing\_zip,Security\_code}

CHECK\_ORDER

{Check\_Order\_no,Check\_Ssn} → {Check\_Quantity,Chech\_Date}

BOOK\_ORDER

{Order\_no} → {CusEmail, Amount, Date}

SHIPMENT

{SOrder\_no, Cost} → {Tax\_id, Last\_location\_tracking}

WAREHOUSE

{Ware\_Tax\_id} → {Warehouse\_Name, Stock, Capacity, Mgr\_Ssn, Phone, Wid}

{Mgr\_Ssn} → {Ware\_Tax\_id, Warehouse\_Name, Stock, Capacity, Phone, Wid}

{Wid}→ {Ware\_Tax\_id, Warehouse\_Name, Stock, Capacity, Mgr\_Ssn, Phone}

SUPPLY

{Store\_Tax\_id, S\_Tax\_id} → {Amount}

SUPPLIER

{Supplier\_Tax\_id} → {Sup\_Name, Phone, Sup\_id}

{Sup\_id} → {Supplier\_Tax\_id, Sup\_Name, Phone}

SALE

{Promo\_code} → {Discount\_percent}

SALE\_DETAIL

{SPromo\_code, SISBN} → {Start\_date, End\_date,Amount}

EMPLOYEE

{Ssn}→{Salary, Birth\_date, Lname, FName, Gender, Phone, EM\_Lience\_no,EA\_id}

{EM\_Lience\_no}→{Ssn,Salary, Birth\_date, Lname, FName, Gender, Phone, EA\_id}

TECHNICIAN

{Tech\_Ssn} → {Certifcate}

MANAGER

{Manager\_Ssn} → {Skills}

CONTAIN

{ConOrder\_no, CISBN} → {CQuantity}

BOOK

{ISBN} → {W\_Tax\_id, Language, Price, Version, Title, Quantity}

PUBLISH

{Publisher\_Tax\_Id, PubISBN}→{Date, Amount,Date}

PUBLISHER

{Publisher\_Tax\_Id}→{Pub\_Name, Phone, Pub\_id}

{Pub\_id}→{Publisher\_Tax\_Id,Pub\_Name, Phone}

POST\_OFFICE

{Tax\_id} → {Post\_Name, Phone, Po\_id}

{Po\_id} → {Tax\_id, Post\_Name, Phone}

AUTHOR

{Id} → {Birth\_date, LName, FName, Gender, Phone, AU\_License\_no, AA\_id}

{AU\_License\_no} → {Id, Birth\_date, LName, FName, Gender, Phone, AA\_id}

WRITE

{WISBN,Wid} → {Wid,WISBN}

LIVE

{drive\_no,Live\_id} → {Live\_id,drive\_no}

ADDRESS

{Aid} → {Zip, Street, City, State}

{Zip} →{City, State}

PROVIDE

{Supplier\_Tax\_id, Pub\_Tax\_id} → {Amount}

1. All of our tales are in BCNF.

|  |  |  |  |
| --- | --- | --- | --- |
| Table | Column | Index type | Reason |
| PAYMENT | Billing\_zip | Tree-based index | Easy to find the areas where residents buy more book |
| BOOK | ISBN,Quantity | Tree-based index | Easy to keep track of the quantity of books and replenish the amount of books on time |
| CUSTOMER | Interests | Hash-based index | Easy to find what most customers’’ interests are |
| EMPLOYEE | Salary | Tree-based index | Easy to find what most employees’ salary are and easy to increase or decrease their salary according to the market |

e.

1. first view

i. The view produces the titles and ISBNs of the books that are sold more than average amount, through this view, managers know the books that are popular with readers and can decide to increase the stock of those books to increase company’s profits.

ii. Allorder←𝝅ISBN,Title,CQuantity(BOOK⨝ BOOK.lSBN=CONTAIN.CISBN CONTAIN)

T1←ISBN,TitleƑSUM CQuantity(Allorder)

T2← ρ(ISBN,Title,SumQuantity)(T1)

T3←AVG SumQuantity (T2)

T4← ρ(AvgQuantity)(T3)

𝝅ISBN,Title (𝝈SumQuantity> AvgQuantity (T2XT4))

iii.CREATE VIEW Popular\_Book(Popular\_ISBN,Popular\_Title)

AS SELECT ISBN,Title

FROM BOOK

WHERE Title IN

(SELECT Title

FROM (SELECT Title, SUM(CQuantity)

FROM BOOK,CONTAIN

WHERE ISBN=CISBN

GROUP BY ISBN

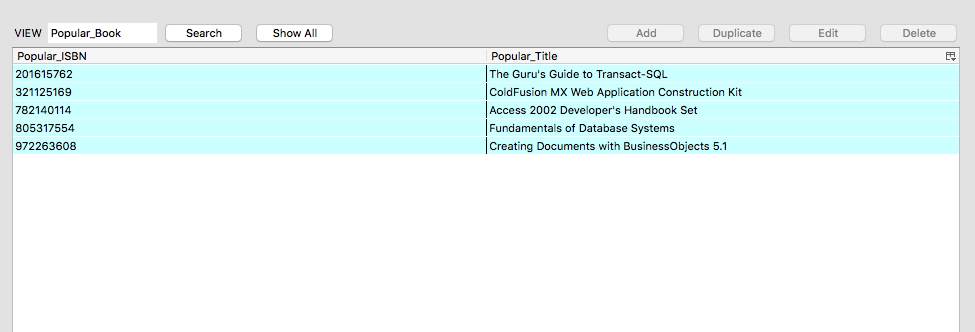
HAVING SUM(CQuantity)>(SELECT AVG(sum) FROM

(SELECT SUM(CQuantity) AS sum

FROM BOOK,CONTAIN

WHERE ISBN=CISBN

GROUP BY ISBN))));

iv

2. second view

i. This view produces the ISBN, Title of the books that are sold more than the average during sales and the discount percent. This view could be useful when managers find the stocks of some books remain large and then managers can check this view to find if discount is a way to increase sales of those books.

ii. T1←𝝅SISBN, Discount\_percent(SALE\_DETAIL ⨝ SPromo\_Code = Promo\_Code SALE)

T2←𝝅ISBN,Title,Discount\_percent(T1 ⨝ SISBN = ISBN BOOK)

T3←ISBN,Title,Discount\_percent ƑSUM Amount(T2)

T4←ρ(ISBN,Title, Discount\_percent ,Am)(T3)

T5←AVG Am (T4)

T6← ρ(AvgAmount)(T5)

𝝅ISBN,Title, Discount\_percent (𝝈Am> AvgAmout (T4XT6))

iii.

CREATE VIEW Salegood\_Book AS

SELECT ISBN,Title,Discount\_percent

FROM SALE, SALE\_DETAIL,BOOK

WHERE ISBN=SISBN AND SPromo\_Code=Promo\_Code AND ISBN IN

(SELECT SISBN

FROM(SELECT SISBN,SUM(Amount)

FROM SALE\_DETAIL,SALE

WHERE SPromo\_Code=Promo\_Code

GROUP BY SISBN, Discount\_percent

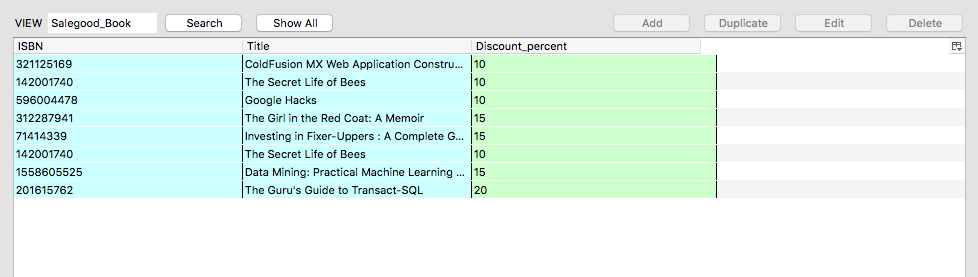
HAVING SUM(Amount)> (SELECT AVG(sum) FROM

(SELECT SUM(Amount) AS sum

FROM SALE\_DETAIL, SALE

WHERE SPromo\_Code=Promo\_Code

GROUP BY SISBN,Discount\_percent))));

iv. 

f.

Transaction 1:

One customer makes change to a book order by decreasing the quantity of books he or she buys, so his or her payment and the quantity attribute in CHECK\_ORDER table which contains the records of each employee checking the orders will also need to be updated.

In the following example, the customer who buys an order number 123456962 containing 5 books with ISBN 201615762 wants to buy 4 of this book rather than 5, so his or her payment should be changed by subtracting the price of book from the old payment.

BEGIN TRANSACTION Change\_Order

UPDATE CONTAIN SET CQuantity= CQuantity-1

WHERE ConOrder\_no=123456962 AND CISBN=201615762

IF error THEN GO TO UNDO; END IF;

UPDATE PAYMENT SET Amount= Amount-38.49

WHERE POder\_no=123456962

IF error THEN GO TO UNDO; END IF;

UPDATE CHECK\_ORDER SET CHECK\_Quantity= CHECK\_Quantity-1

WHERE Check\_Order\_no=123456962

IF error THEN GO TO UNDO; END IF;

COMMIT;

GO TO FINISH;

UNDO:

ROLLBACK;

FINISH:

END TRANSACTION;

Transaction 2:

One customer wants to change his or her email address, so his or her email address in CUSTOMER, BOOK\_ORDER and REQUEST\_SERVICE tables which contain the information of customers, records of book orders he or she bought and customer services he or she required respectively need to be changed.

In the following example, the customer whose original email address was [b@osu.edu](mailto:b@osu.edu) updates his or her email email to 123@osu.edu

BEGIN TRANSACTION Change\_EmailAD

UPDATE CUSTOMER SET Email=123@osu.edu

WHERE [Email=b@osu.edu](mailto:Email=b@osu.edu)

IF error THEN GO TO UNDO; END IF;

UPDATE BOOK\_ORDER SET [CusEmail=123@osu.edu](mailto:CusEmail=123@osu.edu)

WHERE [CusEmail=b@osu.edu](mailto:CusEmail=b@osu.edu)

IF error THEN GO TO UNDO; END IF;

UPDATE REQUEST\_SERVICE SET [SEmail=123@osu.edu](mailto:SEmail=123@osu.edu)

WHERE [SEmail=b@osu.edu](mailto:SEmail=b@osu.edu)

IF error THEN GO TO UNDO; END IF;

COMMIT;

GO TO FINISH;

UNDO:

ROLLBACK;

FINISH:

END TRANSCTION;

TRANSACTION 3:

Because a book is so popular that the publisher publishes more, supplies them to the supplier and supplier provides them to the warehouse, so the Quantity attribute in the BOOK table, the Amount attribute in the PUBLISH table which contains the records of books publishers published, the Amount attribute in the SUPPLY table which contains the records of books that suppliers supplied to warehouse, the stock attribute in the WAREHOUSE table, the Amount in the PROVIDE table which contains records of books that publishers provided to suppliers need to be updated.

In the following example, a book with ISBN 72227885 published by a publisher with tax id 568 is republished 1000 and supplied to the supplier with tax id 374. And then the supplier provides the warehouse with tax id 123 with this book, so the quantity of this book increases by 1000.

BEGIN TRANSACTION Pub\_more

UPDATE PUBLISH SET AMOUNT=AMOUNT+1000

WHERE Publish\_Tax\_id=568 AND PubISBN=72227885

IF error THEN GO TO UNDO; END IF;

UPDATE PROVIDE SET AMOUNT=AMOUNT+1000

WHERE Sup\_Tax\_Id=374 AND Pub\_Tax\_Id=568

IF error THEN GO TO UNDO; END IF;

UPDATE SUPPLY SET AMOUNT=AMOUNT+1000

WHERE Store\_Tax\_Id=123 AND S\_Tax\_Id=374

IF error THEN GO TO UNDO; END IF;

UPDATE WAREHOUSE SET Stock=Stock+1000

WHERE Ware\_Tax\_Id=123

IF error THEN GO TO UNDO; END IF;

UPDATE BOOK SET Quantity=Quantity+1000

WHERE ISBN=72227885

IF error THEN GO TO UNDO; END IF;

COMMIT;

GO TO FINISH;

UNDO:

ROLLBACK;

FINISH:

END TRANSACTION;