Part I

Section 1 - Database Description

Chen Cai

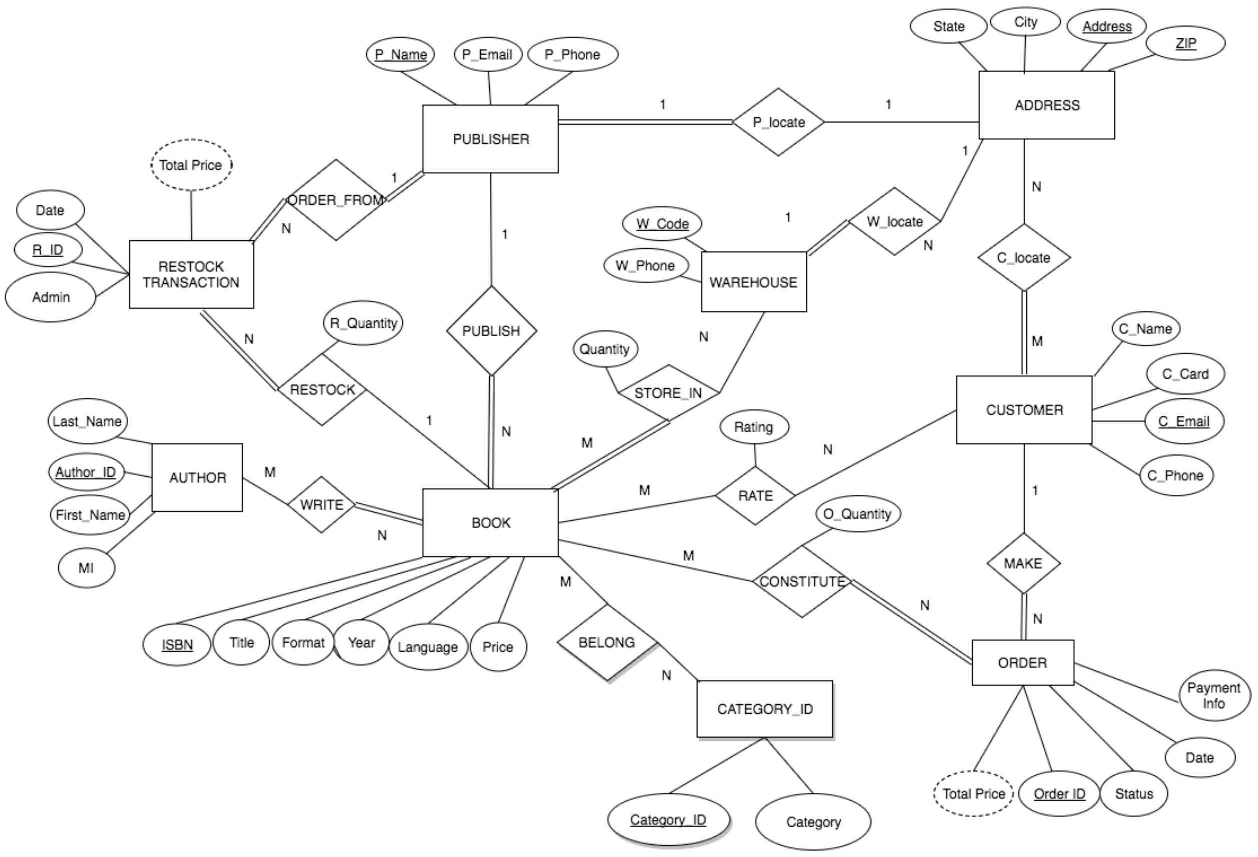
Tingwei Duan

Hui Li

En-Ju Lin

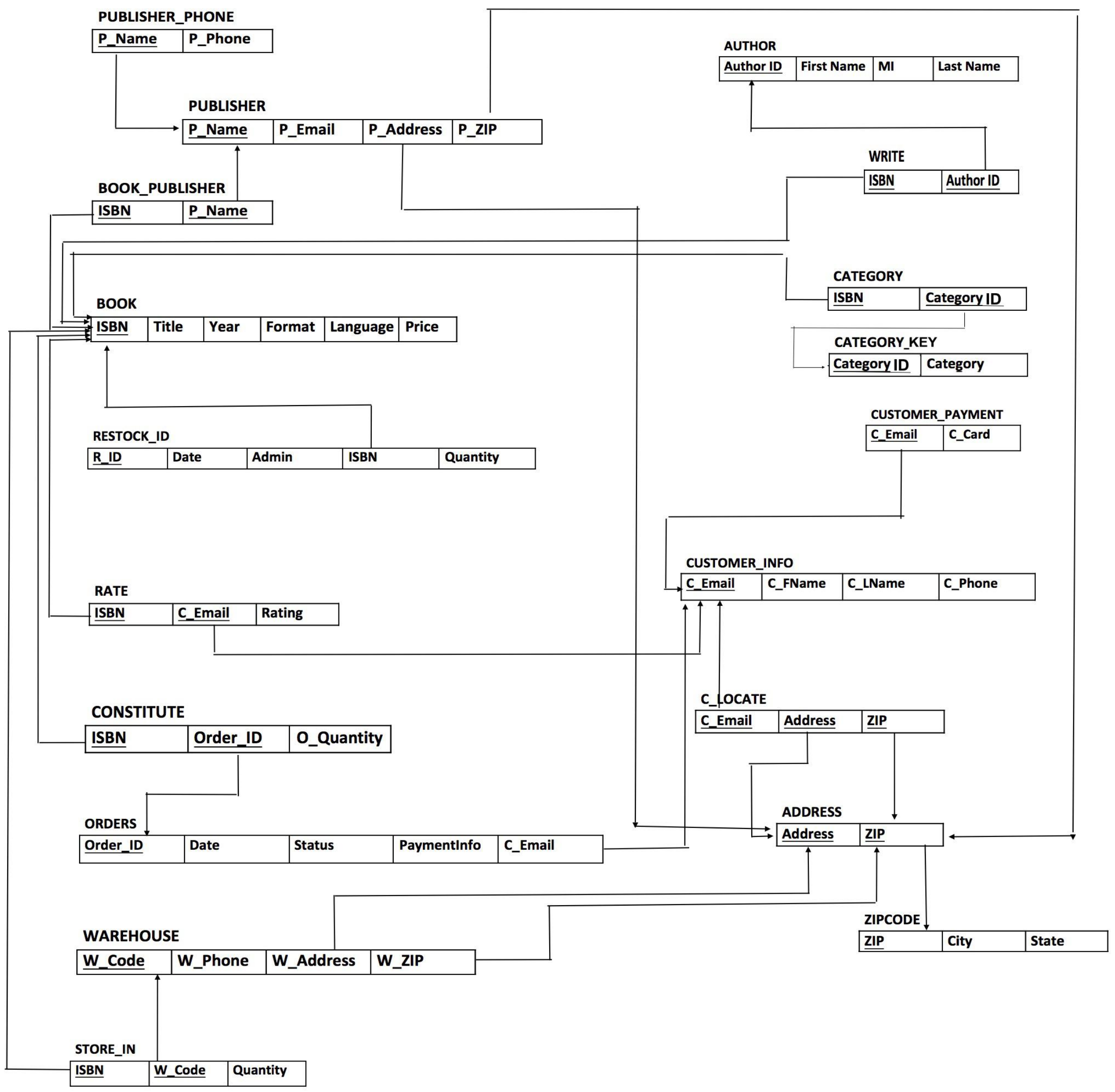
Jun, 2016

Part A. Entity-Relationship model

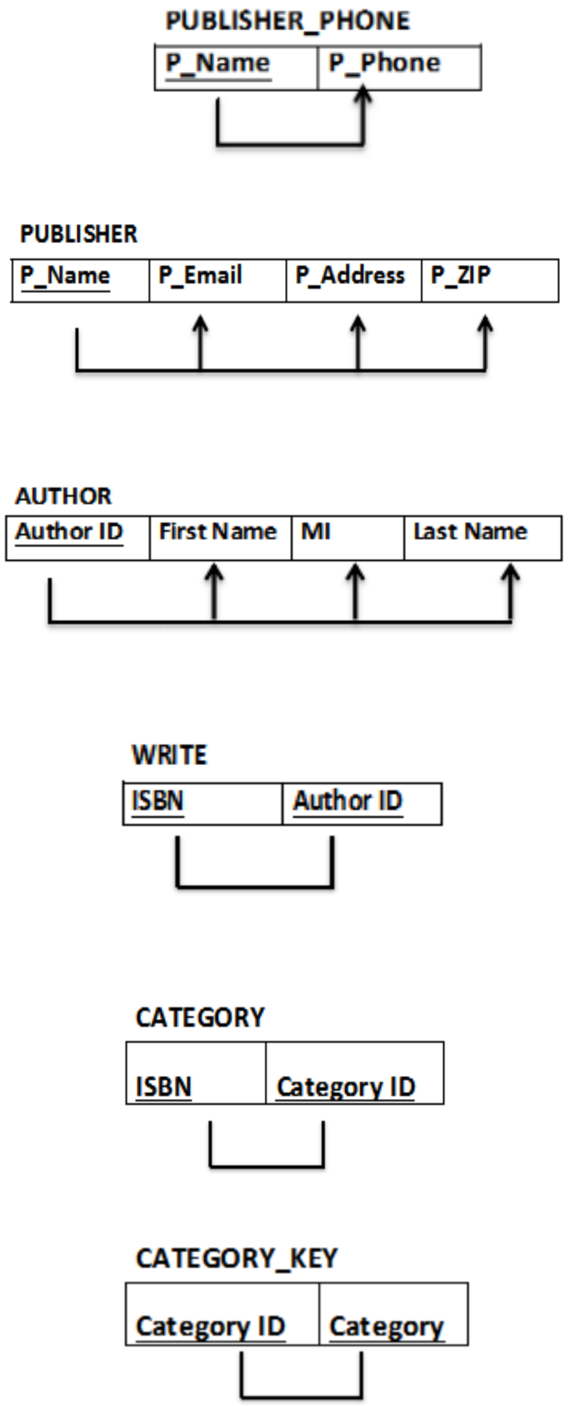


Part B. Relational Schema

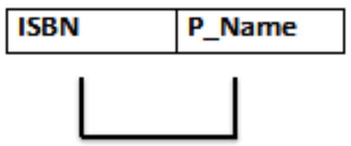
Relational schema with primary keys underlined and foreign keys indicated by arrows pointing to the referenced relation/attributes:



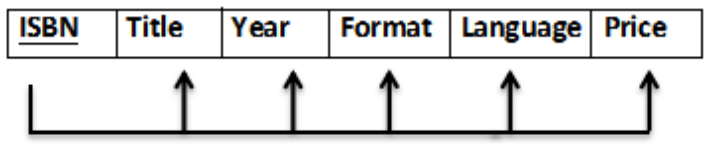
Relational schema with functional dependency:



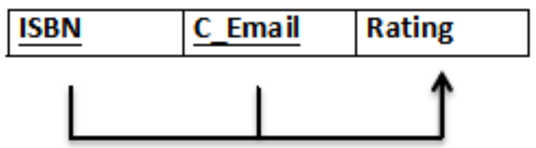
BOOK_PUBLISHER



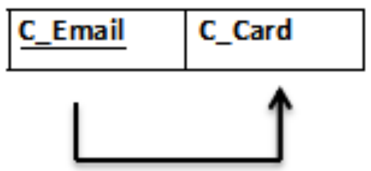
BOOK



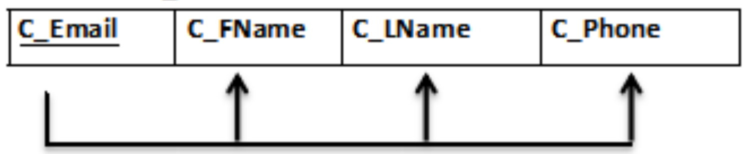
RATE



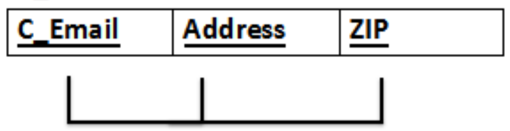
CUSTOMER_PAYMENT

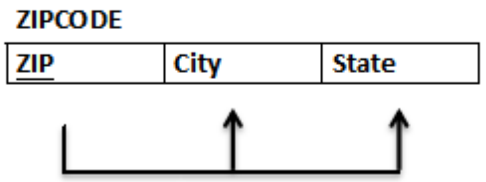


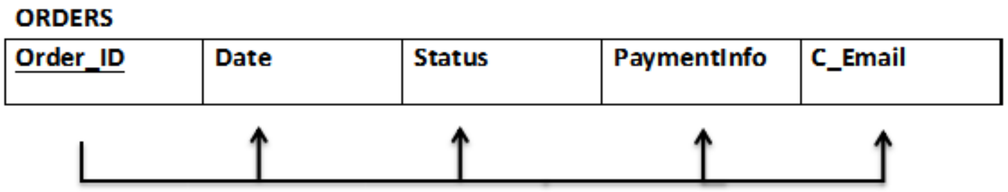
CUSTOMER_INFO

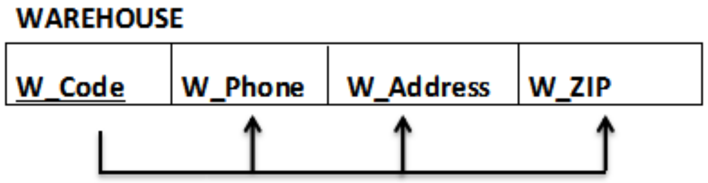


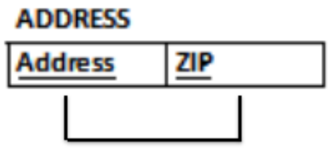
C_LOCATE

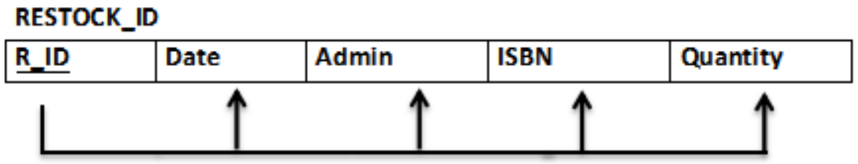


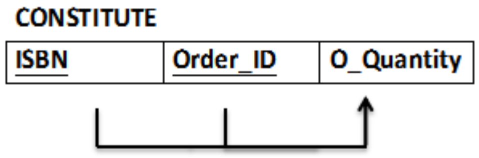

ZIPCODE
ZIP
City
State

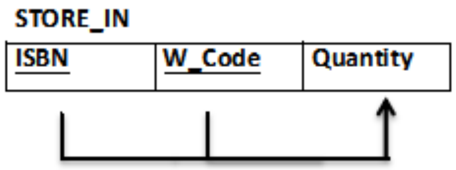

ORDERS
Order_ID
Date
Status
PaymentInfo
C_Email


WAREHOUSE
W_Code
W_Phone
W_Address
W_ZIP


ADDRESS
Address
ZIP


RESTOCK_ID
R_ID
Date
Admin
ISBN
Quantity


CONSTITUTE
ISBN
Order_ID
O_Quantity


STORE_IN
ISBN
W_Code
Quantity

PART C. Normalization and Justification

- PUBLISHER: In BCNF, as every non-key attribute (P_Email, P_Address, P_ZIP) is fully and non-transitively dependent on the key, and every determinant (in this case P_Name) is a candidate key.

  PUBLISHER
  {P_Name, P_Email,P_Address, P_ZIP }

- PUBLISHER_PHONE: In BCNF, as P_Phone is fully and non-transitively dependent on the key, P_Name, which is a candidate key.

  PUBLISHER_PHONE
  {P_Name, P_Phone}

- BOOK: In BCNF, as every non-key attribute (Title, Year, Format, Language, Price) is fully and non-transitively dependent on the primary key ISBN.

  BOOK
  {ISBN, Title, Year, Format, Language, Price}

- BOOK_PUBLISHER: Already in BCNF. Actually dependency does not apply as both ISBN and P_Name are primary keys.

  BOOK_PUBLISHER
  {ISBN, P_Name}

- CUSTOMER_INFO: In BCNF, as every non-key attribute (C_FName, C_LName, C_Phone) is fully and non-transitively dependent on the primary key C_Email.

  CUSTOMER_INFO
  {C_Email, C_FName, C_LName, C_Phone}

- CUSTOMER_PAYMENT: In BCNF, as C_Card is fully and non-transitively dependent on the primary key C_Email.

  CUSTOMER_PAYMENT
  {C_Email, C_Card}

- ADDRESS: Already in BCNF. Actually dependency does not apply, as both Address and ZIP are both primary keys.

ADDRESS
{Address, ZIP}

- ZIPCODE: This is in BCNF, since every non-key attribute (City, State) is non-fully and non-transitively dependent on ZIP, which is the primary key.

  ZIPCODE
  {ZIP, City, State}

- ORDERS: This is in BCNF, since every non-key attribute (Date, Status, PaymentInfo, C_Email) is non-fully and non-transitively dependent on the primary key Order_ID.

  ORDERS
  {Order_ID, Date, Status, PaymentInfo, C_Email}

- WAREHOUSE: This is in BCNF, since every non-key attribute (W_Phone, W_Address, W_ZIP) is non-fully and non-transitively dependent on the primary key W_Code.

  WAREHOUSE
  {W_Code, W_Phone, W_Address, W_ZIP}

- STORE_IN: This is in BCNF, since every non-key attribute (Quantity) is non-fully and non-transitively dependent on *both* primary keys ISBN and W_Code.

  STORE_IN
  {ISBN, W_Code, Quantity}

- RESTOCK_ID: This is in BCNF, since every non-key attribute (Date, Admin, ISBN, Quantity) is non-fully and non-transitively dependent on the primary key R_ID.

  RESTOCK_ID
  {R_ID, Date, Admin, ISBN, Quantity}

- CATEGORY_KEY: Already in BCNF. Actually dependency does not apply as both Category_ID and Category are primary keys.

  CATEGORY_KEY
  {Category_ID, Category}

- CATEGORY: Already in BCNF. Actually dependency does not apply, as both ISBN and Category_ID are primary keys.

  CATEGORY
  {ISBN, Category_ID}

- AUTHOR: This is in BCNF, since every non-key attribute (First Name, MI, Last Name) is non-fully and non-transitively dependent on the primary key Author_ID.

  AUTHOR
  {Author_ID, First_Name, MI, Last_Name}

- WRITE: Already in BCNF. Actually dependency does not apply, as both ISBN and Author_ID are primary keys.

  WRITE
  {ISBN, Author ID}

- RATE: This is in BCNF, since every non-key attribute (Rating) is non-fully and non-transitively dependent on *both* primary keys ISBN and C_Email.

  RATE
  {ISBN, C_Email, Rating}

- CONSTITUTE: This is in BCNF, since every non-key attribute (O_Quantity) is non-fully and non-transitively dependent on *both* primary keys ISBN and Order_ID.

  CONSTITUTE
  {ISBN, ORDER_ID, O_Quantity}

- C_LOCATE: In BCNF. Actually dependency does not apply, as all three attributes are primary keys.

  {C_Email, Address, ZIP}

PART D. Indexes

Table for indexes type in ideal condition

Table/Attribute	Hash-index	Tree-index
PUBLISHER_PHONE	P_NAME	
PUBLISHER	P_NAME	P_Address, P_ZIP
BOOK_PUBLISHER	P_NAME	ISBN
BOOK	Format, Year, Language	ISBN, Price, Title
RESTOCK_ID	Admin	ISBN, Date, R_ID
RATE	Rating	ISBN, C_Email
CONSTITUTE	O_Quantity	ISBN, Order_ID
ORDERS	Status	Order_ID, Date, C_Email
WAREHOUSE	W_Address, W_Code, W_ZIP	-
STORE_IN	W_Code	ISBN, Quantity
AUTHOR	First_Name, Last_Name	Author_ID
WRITE	-	Author_ID, ISBN
CATEGORY	Category_ID	ISBN
CATEGORY_KEY	Category_ID	Category
CUSTOMER_PAYMENT	-	C_Email
CUSTOMER_INFO	-	C_Email
C_LOCATE	-	C_Email, Address, ZIP
ADDRESS	-	Address, ZIP
ZIPCODE	City, State	ZIP

Rationale:

Hash-indexes maybe suitable for attributes such as:

P_Name, Format, Year, Language, Admin, Rating, Status, O_Quantity, W_Address, W_Code, W_ZIP, First_Name, Last_Name, Category, City, State

As they meet some or all of the following characteristics:

1. Contain data that are discrete rather than numbers, which means more likely to be involved in equality search rather than range search.
2. Do not require large size for hash table
3. Do not change frequently
4. Frequently used for join or search functions

So we make them **hash-index** to avoid scanning the whole table.

Tree-indexes are suitable for attributes like:

P_Address, P_ZIP, ISBN, Price, Title, Date, R_ID, C_Email, Order_ID, Quantity, Author_ID, Address, ZIP

Due to the following characteristics:

1. Likely to be searched by range
2. They contain a large scale of data
3. They maybe updated frequently
4. They are frequently used for join or search functions

So we make them **tree-index** to avoid scanning the whole table.

Primary keys/Foreign keys

We should add index for all primary keys and foreign keys since they are frequently used for join operation and so on.

SQL Code for Indexes:

Since SQL support tree indexing only. We will make all the indexes tree-index for temporary use.

Indexes for Primary keys (Will not run on SQL since SQL automatically generate indexes for them):

```
CREATE UNIQUE INDEX PU_name ON PUBLISHER(P_Name);

CREATE UNIQUE INDEX PP_name ON PUBLISHER_PHONE(P_Name);

CREATE UNIQUE INDEX RA_cema ON RATE(C_Email,ISBN);

CREATE UNIQUE INDEX  CO_orid ON CONSTITUTE(Order_ID);

CREATE UNIQUE INDEX  OR_orid ON ORDERS(Order_ID);

CREATE UNIQUE INDEX  WA_code ON WAREHOUSE(W_Code);

CREATE UNIQUE INDEX SI_wcod ON STORE_IN(W_Code);

CREATE UNIQUE INDEX  AU_auth ON AUTHOR(Author_ID);

CREATE UNIQUE INDEX  WR_auth ON WRITE(Author_ID);

CREATE UNIQUE INDEX  CA_cate ON CATEGORY(Category_ID);

CREATE UNIQUE INDEX  CK_cate ON CATEGORY_KEY(Category_ID);

CREATE UNIQUE INDEX  CP_cema ON CUSTOMER_PAYMENT(C_Email);

CREATE UNIQUE INDEX  CI_cema ON CUSTOMER_INFO(C_Email);

CREATE UNIQUE INDEX  CL_cema ON C_LOCATE(C_Email);

CREATE UNIQUE INDEX  ZI_zip ON ZIPCODE(ZIP);

CREATE UNIQUE INDEX  BP_pnam ON BOOK_PUBLISHER(P_Name);
```

Additional indexes which will be added to SQL

```
CREATE UNIQUE INDEX  PU_addr ON PUBLISHER(P_Address);

CREATE INDEX  PU_zip ON PUBLISHER(P_ZIP);

CREATE INDEX  BO_form ON BOOK(Format);

CREATE INDEX  BO_year ON BOOK(Year);

CREATE INDEX  BO_lang ON BOOK(Language);

CREATE INDEX  BO_titl ON BOOK(TITLE);

CREATE INDEX BO_pric ON BOOK(PRICE);

CREATE INDEX  RI_adm ON RESTOCK_ID(Admin);

CREATE UNIQUE INDEX  RI_isbn ON RESTOCK_ID(ISBN);

CREATE INDEX  RI_date ON RESTOCK_ID(Date);

CREATE INDEX  RA_rati ON RATE(Rating);

CREATE INDEX  CO_quan ON CONSTITUTE(O_Quantity);

CREATE INDEX  OR_stat ON ORDERS(Status);

CREATE INDEX  OR_date ON ORDERS(Date);

CREATE UNIQUE INDEX  OR_cema ON ORDERS(C_Email);

CREATE UNIQUE INDEX  WA_addr ON WAREHOUSE(W_Address);

CREATE INDEX  WA_zip ON WAREHOUSE(W_ZIP);

CREATE INDEX  SI_quan ON STORE_IN(Quantity);

CREATE INDEX  AU_fnam ON AUTHOR(First_Name);
CREATE INDEX  AU_lnam ON AUTHOR(Last_Name);

CREATE UNIQUE INDEX  CA_id ON CATEGORY(Category_ID);

CREATE UNIQUE INDEX  CA_isbn ON CATEGORY(ISBN);

CREATE INDEX  CK_cate ON CATEGORY_KEY(Category);

CREATE UNIQUE INDEX  CL_cema ON C_LOCATE(C_Email);

CREATE INDEX  CL_zip ON C_LOCATE(ZIP);

CREATE INDEX  AD_zip ON ADDRESS(ZIP);

CREATE INDEX  ZI_city ON ZIPCODE(City);

CREATE INDEX  ZI_Stat ON ZIPCODE(State);
```

PART E. Useful Views

1.View about the best selling books. This can help customers find the most popular books.

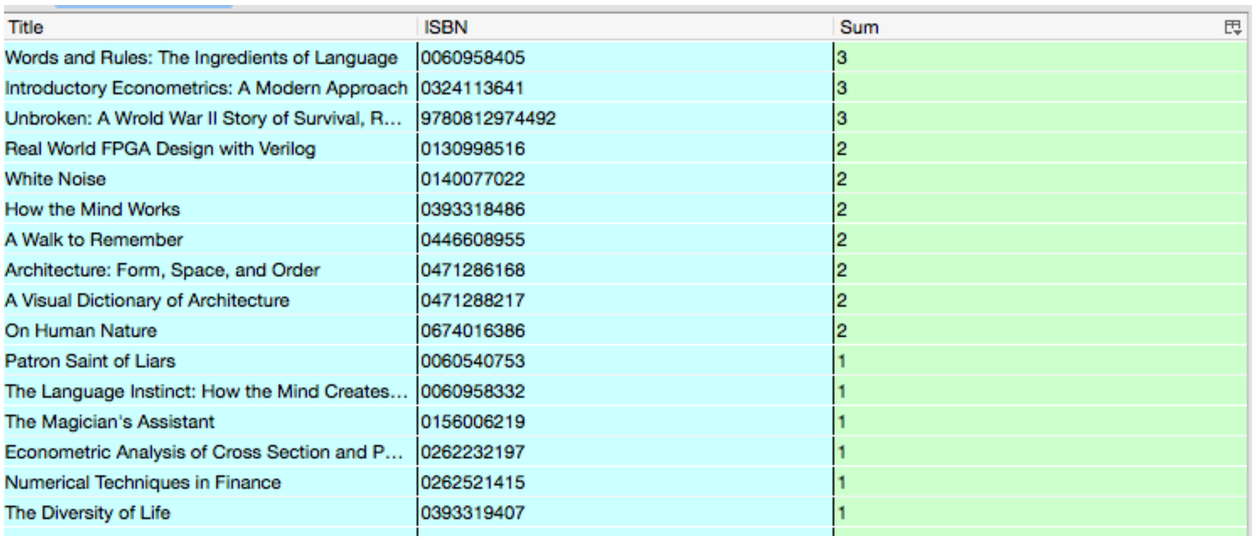
Relational Algebra:

$\text{Temp1} \leftarrow {}_{\text{ISBN}}\mathscr{F}_{\text{SUM O\_Quantity}}\ (\text{CONSTITUTE})$

$\text{Res} \leftarrow \text{T}_{\text{SUM O\_Quantity, Title, ISBN}}\ (\text{Temp1} \bowtie_{\text{Temp1.ISBN = BOOK.ISBN}} \text{BOOK})$

SQL:

```
CREATE VIEW POPULAR
AS      SELECT Title, B.ISBN,SUM(O_Quantity) AS Sum
        FROM BOOK AS B, CONSTITUTE AS C
        WHERE
                B.ISBN = C.ISBN
        GROUP BY B.ISBN
        ORDER BY Sum DESC;
```



Title	ISBN	Sum
Words and Rules: The Ingredients of Language	0060958405	3
Introductory Econometrics: A Modern Approach	0324113641	3
Unbroken: A Wrold War II Story of Survival, R...	9780812974492	3
Real World FPGA Design with Verilog	0130998516	2
White Noise	0140077022	2
How the Mind Works	0393318486	2
A Walk to Remember	0446608955	2
Architecture: Form, Space, and Order	0471286168	2
A Visual Dictionary of Architecture	0471288217	2
On Human Nature	0674016386	2
Patron Saint of Liars	0060540753	1
The Language Instinct: How the Mind Creates...	0060958332	1
The Magician's Assistant	0156006219	1
Econometric Analysis of Cross Section and P...	0262232197	1
Numerical Techniques in Finance	0262521415	1
The Diversity of Life	0393319407	1

2.View about the titles and ISBNs for all books with less than 15 copies in stock. This view can help administrator find which books is running low on inventory so they can decide what to reorder.

Relational Algebra:

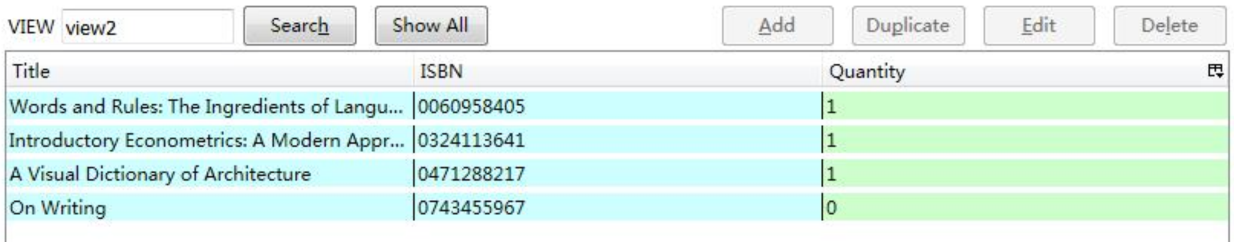
$\text{Temp1} \leftarrow {}_{\text{ISBN}}\mathscr{F}_{\text{SUM Quantity}} (\pi_{\text{STORE\_IN.ISBN}} (\text{STORE\_IN}))$

$\text{Temp2} \leftarrow \delta_{\text{SUM(Quantity)} < 15} (\text{Temp1})$

$\text{Res} \leftarrow \pi_{\text{Title,ISBN, SUM Quantity}} (\text{BOOK} \bowtie_{\text{BOOK.ISBN = Temp2.ISBN}} \text{Temp2})$

SQL:

```
CREATE VIEW INV_SHORT
AS      SELECT B.Title, B.ISBN, S.Quantity
        FROM BOOK AS B, STORE_IN AS S
        WHERE
                B.ISBN = S.ISBN
        GROUP BY S.ISBN
        HAVING sum (S.Quantity) < 15;
```



Title	ISBN	Quantity
Words and Rules: The Ingredients of Langu...	0060958405	1
Introductory Econometrics: A Modern Appr...	0324113641	1
A Visual Dictionary of Architecture	0471288217	1
On Writing	0743455967	0

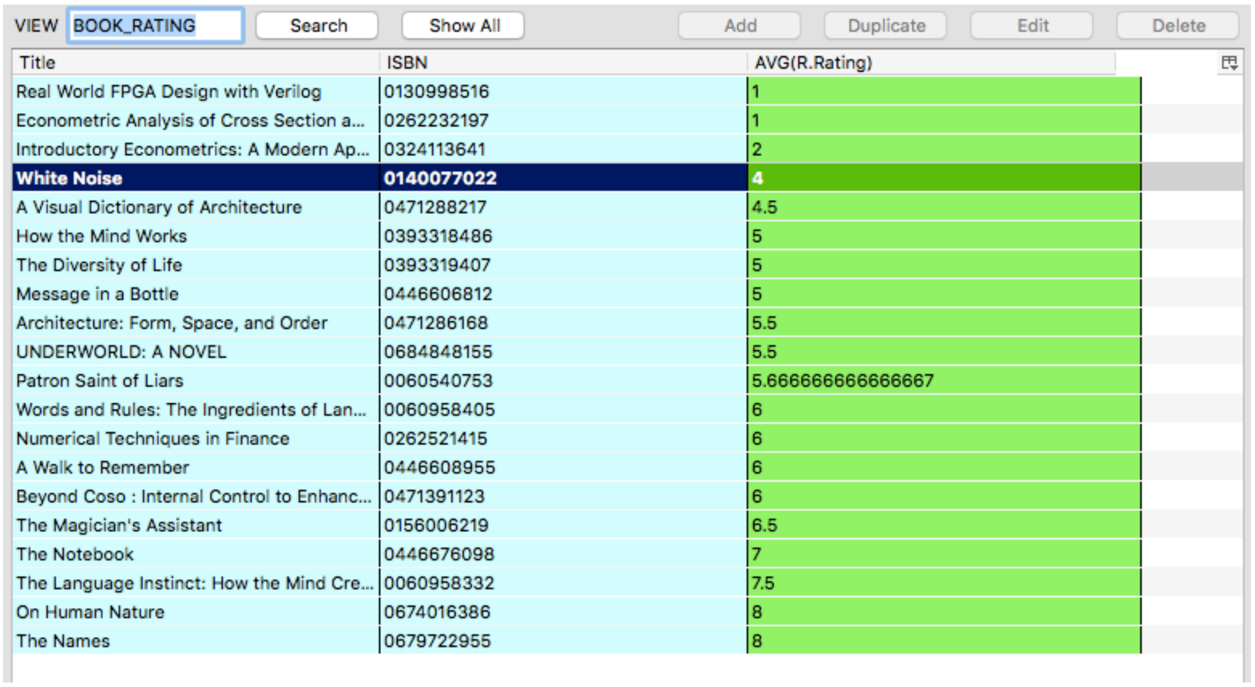
3. View on book rating. This view provides the average book rating (as provided by customers) in descending order and will be useful for other customers as they decide whether to purchase a book and also for administrators to decide how much inventory to keep.

Relational Algebra:

$$\text{Temp1} \leftarrow {}_{\text{ISBN}}\mathscr{F}_{\text{AVG Rating}}(\text{RATE})$$
$$\text{Res} \leftarrow \text{T}_{\text{AVG Rating, Title, ISBN}}(\text{BOOK} \bowtie_{\text{BOOK.ISBN} = \text{Temp1.ISBN}} \text{Temp1})$$

SQL:

```
CREATE VIEW BOOK_RATING
AS      SELECT B.Title, B.ISBN, AVG(R.Rating)
        FROM BOOK AS B, RATE AS R
        WHERE
                B.ISBN = R.ISBN
        GROUP BY R.ISBN
ORDER BY AVG(R.Rating);
```


VIEW BOOK_RATING Search Show All Add Duplicate Edit Delete

Title	ISBN	AVG(R.Rating)
Real World FPGA Design with Verilog	0130998516	1
Econometric Analysis of Cross Section a...	0262232197	1
Introductory Econometrics: A Modern Ap...	0324113641	2
White Noise	**0140077022**	**4**
A Visual Dictionary of Architecture	0471288217	4.5
How the Mind Works	0393318486	5
The Diversity of Life	0393319407	5
Message in a Bottle	0446606812	5
Architecture: Form, Space, and Order	0471286168	5.5
UNDERWORLD: A NOVEL	0684848155	5.5
Patron Saint of Liars	0060540753	5.666666666666667
Words and Rules: The Ingredients of Lan...	0060958405	6
Numerical Techniques in Finance	0262521415	6
A Walk to Remember	0446608955	6
Beyond Coso : Internal Control to Enhanc...	0471391123	6
The Magician's Assistant	0156006219	6.5
The Notebook	0446676098	7
The Language Instinct: How the Mind Cre...	0060958332	7.5
On Human Nature	0674016386	8
The Names	0679722955	8

4. Provide a list of customer names, along with the total dollar amount each customer has spent, in decreasing order. This can help employee find valuable customers.

Relational Algebra:

$\text{Temp1} \leftarrow (\text{BOOK} \bowtie_{\text{BOOK.ISBN = CONSITITUE.ISBN}} (\text{CONSTITUTE}))$

$\text{Temp2} \leftarrow (\text{Temp1} \bowtie_{\text{Temp1.Order\_ID = ORDER.Order\_ID}} (\text{ORDERS})$

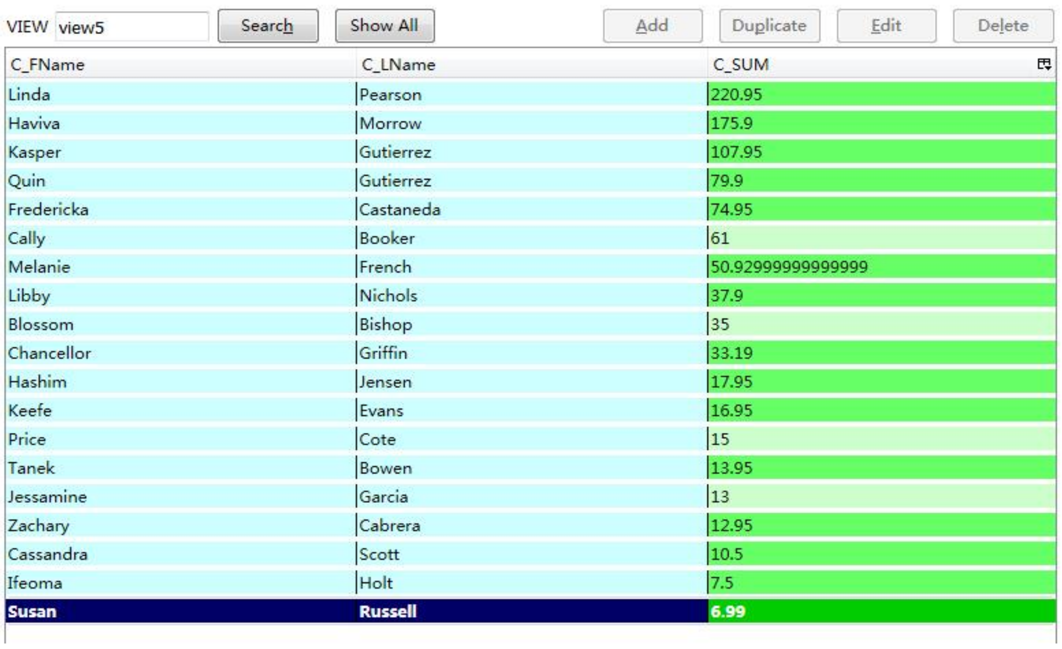
$\text{Temp3} \leftarrow \rho_{(\text{C\_Email, TotalSpending})} (_{\text{C\_Email}}\mathcal{F}_{\text{SUM(B.Price * C.O\_Quantity)}} (\text{Temp2}))$

$\text{Temp4} \leftarrow (\text{Temp3} \bowtie_{\text{Temp2.C\_Email = CUSTOMER\_INFO.C\_Email}} (\text{CUSTOMER\_INFO})$

$\text{Res} \leftarrow \text{T}_{\text{TotalSpending, C\_FName, C\_LNameISBN}} (\text{Temp4})$

SQL:

```
CREATE VIEW CUSTOMER_SALE
AS      SELECT      C_FName, C_LName, SUM(B.Price * C.O_Quantity) AS C_SUM
        FROM        BOOK AS B, CONSTITUTE AS C, CUSTOMER_INFO AS U,
ORDERS AS O
        WHERE       B.ISBN = C.ISBN AND O.C_Email = U.C_Email
                    AND C.Order_ID = O.Order_ID
        GROUP BY    U.C_Email
        ORDER BY    C_SUM DESC;
```



C_FName	C_LName	C_SUM
Linda	Pearson	220.95
Haviva	Morrow	175.9
Kasper	Gutierrez	107.95
Quin	Gutierrez	79.9
Fredericka	Castaneda	74.95
Cally	Booker	61
Melanie	French	50.92999999999999
Libby	Nichols	37.9
Blossom	Bishop	35
Chancellor	Griffin	33.19
Hashim	Jensen	17.95
Keefe	Evans	16.95
Price	Cote	15
Tanek	Bowen	13.95
Jessamine	Garcia	13
Zachary	Cabrera	12.95
Cassandra	Scott	10.5
Ifeoma	Holt	7.5
Susan	Russell	6.99

PART F. Transactions

1. Add a new book rating for the book with ISBN ‘471391123’ from the customer with email ‘lectus.Nullam.suscipit@Maecenas.com’ rating 5.

```
INSERT the new rating into RATE

If INSERT fails, UNDO INSERT
```

SQL Code:

```
BEGIN TRANSACTION NEW_RATE

    INSERT INTO RATE VALUES (‘471391123’,
‘lectus.Nullam.suscipit@Maecenas.com’, 5);

    IF error THEN GO TO UNDO; END IF;

        COMMIT;

    GO TO FINISH;

    UNDO:

    ROLLBACK;

    FINISH:

END TRANSACTION;
```

2. Update certain book admin (change the admin name with book ISBN ‘471391123’)

```
UPDATE the Admin for ISBN ‘471391123’ IN RESTOCK_ID

IF UPDATE fails, UNDO UPDATE
```

SQL Code:

```
BEGIN TRANSACTION Admin_Change
    UPDATE RESTOCK_ID SET admin= ‘Tom Black’
        WHERE ISBN = ‘471391123’;
    IF error THEN GO TO UNDO; END IF;
        COMMIT;
    GO TO FINISH;
    UNDO:
    ROLLBACK;
    FINISH:
END TRANSACTION;
```

3. Update certain Order status (change the order status to ‘Completed’)

```
    UPDATE the Status with Order_ID = ‘5764938434’ AND C_Email =
‘erat.neque.non@tempuseuligula.co.uk’ in ORDERS

    If UPDATE fails, UNDO UPDATE
```

SQL Code:

```
        BEGIN TRANSACTION Status_Change

            UPDATE ORDERS SET Status = ‘Completed’

                WHERE Order_ID = ‘5764938434’ AND C_Email
            =‘erat.neque.non@tempuseuligula.co.uk’;

            IF error THEN GO TO UNDO; END IF;

                COMMIT;

            GO TO FINISH;

            UNDO:

            ROLLBACK;

             FINISH:

        END TRANSACTION;
```