A simple database supporting an online book seller

Changes in blue font

Tables about Books and Authors

```
CREATE TABLE Book (
   Isbn              INTEGER,
   Title             CHAR[120] NOT NULL,
   Synopsis          CHAR[500],
   ListPrice         CURRENCY NOT NULL,
   AmazonPrice       CURRENCY NOT NULL,
   SavingsInPrice    CURRENCY NOT NULL,   /* redundant
   AveShipLag        INTEGER,
   AveCustRating     REAL,
   SalesRank         INTEGER,
   CoverArt          FILE,
   Format            CHAR[4] NOT NULL,
   CopiesInStock     INTEGER,
   PublisherName     CHAR[120] NOT NULL,  //Remove NOT NULL if you want 0 or 1
   PublicationDate   DATE NOT NULL,
   PublisherComment  CHAR[500],
   PublicationCommentDate DATE,
   PRIMARY KEY (Isbn)
   FOREIGN KEY (PublisherName) REFERENCES Publisher,
        ON DELETE NO ACTION,  ON UPDATE CASCADE,
   CHECK (Format = 'hard' OR Format = 'soft' OR Format = 'audi'
                              OR Format = 'cd' OR Format = 'digital')
         // alternatively, CHECK (Format IN ('hard', 'soft', 'audi', 'cd', 'digital'))
   CHECK (AmazonPrice + SavingsInPrice = ListPrice)
)

CREATE TABLE Author (
   AuthorName     CHAR[120],
   AuthorBirthDate DATE,
   AuthorAddress   ADDRESS,
   AuthorBiography  FILE,
   PRIMARY KEY (AuthorName, AuthorBirthDate)
)

CREATE TABLE WrittenBy (/*Books are written by authors
   Isbn INTEGER,
   AuthorName CHAR[120],
   AuthorBirthDate DATE,
   OrderOfAuthorship INTEGER NOT NULL,
   AuthorComment  FILE,
```

```
   AuthorCommentDate DATE,
   PRIMARY KEY (Isbn, AuthorName, AuthorBirthDate)
   FOREIGN KEY (Isbn) REFERENCES Book,
      ON DELETE CASCADE, ON UPDATE CASCADE,
   FOREIGN KEY (AuthorName, AuthorBirthDate) REFERENCES Author,
      ON DELETE CASCADE, ON UPDATE CASCADE
)
// Apropos WrittenBy: consider how to insure that there are no "skipped"
// OrderOfAuthorship beginning at 1 (i.e.,disallow entries that represent  first, third,
// fourth authors (only) for a given Isbn, disallow a second author, but no first author, etc.


CREATE TABLE Publisher (
   PublisherName      CHAR[120],
   PublisherAddress   ADDRESS,
   PRIMARY KEY (PublisherName)
)

// insure participation constraint of Publisher in Book (you were asked to write this)
CREATE ASSERTION PublisherBookConstraint
CHECK (NOT EXISTS (SELECT *
                   FROM Publisher P
                   WHERE P.PublisherName
                                 NOT IN (SELECT B.PublisherName
                                         FROM Book B)))


// insure participation constraint of Books in WrittenBy
CREATE ASSERTION BookWrittenByConstraint
CHECK (NOT EXISTS
                  (SELECT *
                   FROM Book B
                   WHERE B.Isbn NOT IN (SELECT W.Isbn FROM WrittenBy W)))


// insure participation constraint of Authors in WrittenBy
CREATE ASSERTION AuthorWrittenConstraint
CHECK (NOT EXISTS
                  (SELECT *
                   FROM Author A
                   WHERE A.AuthorName, A.AuthorBirthDate
                            NOT IN (SELECT W.AuthorName, W.AuthorBirthDate
                                    FROM WrittenBy W)))
```

Tables about Customers and Customer Service

```
CREATE TABLE Customers (/* Customers identified by email address
   CustEmailAddr    CHAR[120],
   CustName    CHAR[120] NOT NULL,
   CustPassword  CHAR[20] NOT NULL,
   PRIMARY KEY (CustEmailAddr)
)

//Customers can request notification about new books by an author
CREATE TABLE AlertTo (
   CustEmailAddress  CHAR[120],
   DateOfAlertRequest  DATE NOT NULL,
   AuthorName  CHAR[120],
   AuthorBirthDate DATE,
   PRIMARY KEY (UserEmailAddr, AuthorName, AuthorBirthDate)
   FOREIGN KEY (AuthorName, AuthorBirthDate) REFERENCES Author
     ON DELETE CASCADE, ON UPDATE CASCADE,
   FOREIGN KEY (CustEmailAddr) REFERENCES Customer
      ON DELETE NO ACTION, ON UPDATE CASCADE
)

CREATE TABLE Accounts (/* Customers can have zero or more accounts
   CustEmailAddr    CHAR[120],
   CreditCardNumber  INTEGER,
   ShippingAddr   ADDRESS NOT NULL,
   DateOpened    DATE NOT NULL,
   PRIMARY KEY (CustEmailAddr, CreditCardNumber)
   FOREIGN KEY (CustEmailAddr) REFERENCES Customer
        ON DELETE CASCADE, ON UPDATE CASCADE
)


// insure participation constraint of Customer in /Accounts – every customer have at least
// one account
CREATE ASSERTION CustomerAccountConstraint
CHECK (NOT EXISTS (SELECT *
                           FROM Customer C
                           WHERE C.CustEmailAddr
                                NOT IN (SELECT A.CustEmailAddr FROM Account A)))
```

Tables about Purchases and Shipments

```
//Transactions (purchases) are made on a customer account
CREATE TABLE Transactions (
  TransNumber   INTEGER,
  OrderDate   DATE,
  PaymentClearanceDate  DATE,      /* if NULL, then payment has not cleared */
  CustEmailAddr    CHAR[120] NOT NULL,
  CreditCardNo   INTEGER NOT NULL,
  PRIMARY KEY (TransNum),
  FOREIGN KEY (CustEmailAddr, CreditCardNo) REFERENCES Account
    ON DELETE NO ACTION, ON UPDATE CASCADE
 )

CREATE TABLE Shipment (/* A record of purchases awaiting or when shipment
  ShipId INTEGER,
  ShipCost CURRENCY,
  ShipDate DATE,                  /* if this is NULL, then not shipped yet */
  TransNumber INTEGER NOT NULL,
  PRIMARY KEY (ShipId)
  FOREIGN KEY (TransNumber) REFERENCES Transaction
     ON DELETE CASCADE, ON UPDATE CASCADE
)

// insure participation constraint on Transaction in Shipment (at least one shipment
// per transaction
CREATE ASSERTION TransactionsShipmentConstraint
CHECK (NOT EXISTS (SELECT *
                          FROM Transaction T
                          WHERE T.TransNumber NOT IN (SELECT S.TransNumber
                                                                FROM Shipment S)))


CREATE TABLE BookShipment (/* A quantity of book associated with a shipment and
                                    /* therefore transaction
   Quantity        INTEGER,
   ShipId          INTEGER,
   Isbn            INTEGER,
   PRIMARY KEY (ShipId, Isbn),
   FOREIGN KEY (ShipId) REFERENCES Shipment
     ON DELETE CASCADE, ON UPDATE CASCADE,
   FOREIGN KEY (Isbn) REFERENCES Book
     ON DELETE NO ACTION, ON UPDATE CASCADE
)
```

Upload Answers to these to Oak Discussion Board

***Study Group members in attendance*:**

Special Note: I have changed table previously named 'Shipped' to 'BookShipment'

Add the following constraints to the Database above (USE BLUE font for your answers, including your changes to the DB above and your query corrections below)

Does it appear that a Book is associated with more than one publisher? No, it appears that a book is associated with EXACTLY one publisher.

1. Add a FOREIGN KEY constraint to Book that references Publisher so as to make this association unambiguous. See new DB definitions above for answer

Does it appear that a Book can be associated with more than one author? YES

2. Add FOREIGN KEY constraints to WrittenBy so as to make the association between Book and Author unambiguous. See new DB definitions above for answer

Does it appear that a Book can be associated with zero (known) authors? YES, THERE IS NO REQUIREMENT THAT AN ENTRY IN BOOK NEEDS TO BE PRESENT IN WRIITENBY AT ALL

Can two of a customer's accounts be associated with the same credit card ? NO

Does it appear that a shipment can be associated with more than one Transaction? NO, BUT A TRANSACTION CAN BE ASSOCIATED WITH MULTIPLE SHIPMENTS

3. Add other FOREIGN KEY constraints, associating
- **(a) Account with Customer**
- **(b) AlertTo with Author**
- **(c) AltertTo with Customer**
- **(d) BookShipment with Shipment**
- **(e) Shipment with Transaction**
- **(f) Transaction with Account**

See new DB definitions above for answer

IN WRITING THE FOREIGN KEY CONSTRAINTS FOR (a) and (e), INSURE THAT IF A CUSTOMER IS DELETED THEN ALL THE CUSTOMER'S ACCOUNTS ARE DELETED, UNLESS ANY OF THE CUSTOMER"S ACCOUNTS PARTICIPATES IN A TRANSACTION (IN WHICH CASE THE DELETION OF THAT ACCOUNT, AND THEREFORE THE CUSTOMER WITH THAT ACCOUNT, IS BLOCKED)

See new DB definitions above for answer

4. Encode the constraint in Book that AmazonPrice + SavingsInPrice = ListPrice as an in-table CHECK statement. See new DB definitions above for answer

5. Encode the constraint that book Format be 'hardcover', 'softcover', 'audio', 'cd', or 'digital' as an in-table CHECK statement. See new DB definitions above for answer

6. Write an ASSERTION that insures that every Customer has at least one Account (place this assertion here). See new DB definitions above for answer

7. Write an ASSERTION that insures that each Publisher is associated at least one Book its published (place this assertion here). See new DB definitions above for answer

8. Change the definition of Book so that each Book is associated with 0 or 1 publishers. See new DB definitions above for answer

9. DEBUG QUERIES 2 and 3 below

Queries *Each group write the query – suggestion is that each person spend a few minutes working on it, then discuss and reach consensus, or not (Groups can upload more than one answer per query)*

You will refer to the table definitions of the Book retailer DB for this assignment.

1. An online book retailer undoubtedly uses a more sophisticated resource planning policy than this, but certainly if the number of copies of a given book (Isbn) awaiting shipment exceeds the number of copies in stock, then ordering more books of the given Isbn is appropriate. Write a query that lists the Isbns of all books in which the number of copies awaiting shipment (Shipment.ShipDate IS NULL) to all customers exceeds the number of copies in stock (Books.CopiesInStock) . The query result should also list the difference between the total number of copies awaiting shipment and the number of copies in stock for each qualifying Isbn. Thus, your query should produce a table with two fields (Isbn, NumberofCopiesShort), and there will be only one entry per Isbn that is short.

Two alternatives

```
SELECT B.Isbn, Temp.Sold – B.CopiesInStock AS NumberOfCopiesShort
FROM  Books B,
        (SELECT Bs.Isbn, SUM(Bs.Quantity) AS Sold
         FROM BookShipment Bs, Shipment St
         WHERE St.ShipDate IS NULL AND St.ShipId = Bs.ShipId
         GROUP BY Bs.Isbn) AS Temp
```

```
WHERE B.Isbn = Temp.Isbn AND B.CopiesInStock < Temp.Sold

SELECT B.Isbn, SUM(Bs.Quantity) – B.CopiesInStock AS NumberOfCopiesShort
FROM Books B, BookShipment Bs, Shipment St
WHERE St.ShipDate IS NULL AND St.ShipId = Bs.ShipId AND Bs.Isbn = B.Isbn
GROUP BY B.Isbn, B.CopiesInStock
HAVING SUM(Bs.Quantity) > B.CopiesInStock
```

2. Write a query that lists the CustEmailAddresses of all customers that have purchased books (Transactions.PaymentClearenceDate IS NOT NULL) that have not been shipped (Shipment.ShipDate IS NULL). The query should also list the Isbns and quantities of each unshipped book for each customer. Thus, your query should produce a table with three fields (CustEmailAddr, Isbn, #UnshippedBooks), where there will be only one entry for each (CustEmailAddr, Isbn) pair.

DEBUG THIS QUERY (see blue font addition for answer)

```
SELECT T.CustEmailAddr, Bs.Isbn, SUM(Bs.Quantity) AS NumberUnshippedBooks
FROM  BookShipment Bs,  Shipment St, Transaction T
WHERE St.ShipDate IS NULL AND T.PaymentClearanceDate IS NOT NULL AND
          AND St.ShipId = Bs.ShipId AND T.TransNumber = St.TransNumber
GROUP BY T.CustEmailAddr, Bs.Isbn
```

3. An online book retailer would like to know how effective its Alert service is at promoting sales. Each alert is entered on a particular date for a particular author. A customer that buys a book that was published on a date after the alert date by the given author, has arguably responded to the alert in a positive way as far as our retailer is concerned. Write a query that lists the email addresses of all customers (CustEmailAddr) that have purchased books (Books.Isbn, Transactions.PaymentClearenceDate IS NOT NULL) by authors (AuthorName, AuthorBirthDate) specified in the alert that were published after that customer (AlertTo.CustEmailAddr = Customers.CustEmailAddr) initiated a relevant alert request. The table produced will have two fields (CustEmailAddr, TotalNumberofPostAlertPurchases). The query result should order the customers (CustEmailAddresses) from those that have purchased the most total quantity of books (including copies of the same book) to those that have purchased the least quantity.

Since our tables assume a field type DATE, you can assume that if D1 and D2 are DATEs, then D1 > D2, D1 < D2, D1 = D2, etc are legal comparisons with the obvious meaning.

DEBUG THIS QUERY

```
SELECT T.CustEmailaddr, SUM(Bs.Quantity) AS TotalNumberofPostAlertPurchases
FROM WrittenBy W, Books B, AlertTo A, Transactions T, BookShipment Bs,
        Shipment S
WHERE A.DateOfAlertRequest < B.PublicationDate AND
          T.CustEmailAddr = A.CustEmailAddr AND
          W.AuthorName = A.AuthorName AND
          W.AuthorBirthDate = A.AuthorBirthDate AND
          Bs.Isbn = B.Isbn AND W.Isbn = B.Isbn AND
          T.TransNumber  = Bs.TransNumber S.TransNumber AND
          S.ShipId = Bs.ShipId AND
          T.PaymentClearanceDate IS NOT NULL
GROUP BY T.CustEmailAddr
ORDER BY TotalNumberofPostAlertPurchases
```