Databases and Transaction Processing: An Application-Oriented Approach (Complete Version)
Test Bank: Additional Problems and Solutions

Michael Kifer, Arthur Bernstein, Philip M. Lewis
Contents

Problems for Chapter 2: The Big Picture 4
Problems for Chapter 3: The Relational Data Model 7
Problems for Chapter 4: Conceptual Modeling of Databases with E-R and UML 13
Problems for Chapter 5: Relational Algebra and SQL 31
Problems for Chapter 6: Database Design with the Relational Normalization Theory 49
Problems for Chapter 7: Triggers and Active Databases 65
Problems for Chapter 8: Using SQL in an Application 69
Problems for Chapter 9: Physical Data Organization and Indexing 73
Problems for Chapter 10: The Basics of Query Processing 85
Problems for Chapter 11: An Overview of Query Optimization 89
Problems for Chapter 12: Database Tuning 101
Problems for Chapter 13: Relational Calculus, Visual Query Languages, and Deductive Databases 107
Problems for Chapter 14: Object Databases 114
Problems for Chapter 15: XML and Web Data 121
Problems for Chapter 16: Distributed Databases 138
Problems for Chapter 17: OLAP and Data Mining 140
Problems for Chapter 18: ACID Properties of Transactions 142
Problems for Chapter 19: Models of Transactions 147
Problems for Chapter 20: Implementing Isolation 149
Problems for Chapter 21: Isolation in Relational Databases 158
Problems for Chapter 22: Atomicity and Durability 172
Problems for Chapter 23: Architecture of Transaction Processing Systems 178
Problems for Chapter 24: Implementing Distributed Transactions 181
Problems for Chapter 25: Web Services 191
Problems for Chapter 26: Security and Electronic Commerce 199
Problems for Appendix A: An Overview of Transaction Processing 204
Problems for Chapter 2: The Big Picture

1. Consider a table with the columns as shown below:

   Book(ISBN,Title,Publisher,PublicationDate)

Write the following query using SQL: Find the titles of all books that were published on 9/9/1991.

Solution:

```
SELECT Title
FROM Book B
WHERE B.PublicationDate = '9-9-1991'
```
2. Consider a table with the columns as shown below:

\[ \text{CAR} (\text{Make}, \text{ModelNum}, \text{MName}, \text{Kind}) \]

Write the following query using SQL: *Find the kinds of all cars whose make is “Ford”.*

**Solution:**

```sql
SELECT C.Kind
FROM CAR C
WHERE C.Make='Ford'
```
3. Consider a table with the columns as shown below:

\[ \text{Movie(Title, ReleaseDate, Director, StudioName)} \]

Express the following queries using SQL:

(a) Find all movies released on 1/11/2002.
(b) Find all movie titles produced by Sony Pictures.
(c) Count the number of movies that were released on 1/11/2002.

Solution:

```
SELECT * FROM Movies
WHERE ReleaseDate = "1/11/2002"
```

```
SELECT Title FROM Movies
WHERE StudioName = "SonyPictures"
```

```
SELECT COUNT(*) FROM Movies
WHERE ReleaseDate = "1/11/2002"
```
Problems for Chapter 3: The Relational Data Model

1. (a) Design a schema for a library system containing the following data.
   i. the name, unique Id and number of books on loan for each patron
   ii. the unique isbn number, title, author (each book has a single author), current status
       (possible values are on-shelf or on-loan), borrower Id (if book is on-loan), and shelf-
       Id of each book
   iii. the unique shelf-Id and capacity (number of books) of each shelf

Show all primary and foreign keys.

Solution:

CREATE TABLE PATRONS ( 
   Id INTEGER, 
   Name CHAR(20), 
   NumBorrowed INTEGER, 
   PRIMARY KEY (Id) )

CREATE TABLE BOOKS ( 
   Isbn CHAR(20), 
   Title CHAR(40), 
   Author CHAR(20), 
   Status CHAR(5), 
   BorrowerId INTEGER, 
   ShelfId INTEGER, 
   CHECK (Status IN ('on-loan', 'on-shelf') ) 
FOREIGN KEY (ShelfId) 
   REFERENCES SHELVES (ShelfId) 
FOREIGN KEY (BorrowerId) 
   REFERENCES PATRONS (Id) )

CREATE TABLE SHELVES ( 
   ShelfId INTEGER, 
   Capacity INTEGER 
   PRIMARY KEY (ShelfId) )

(b) Add a constraint that enforces the restriction that a patron cannot borrow more than 5
    books at a time.

Solution:

Add the following to PATRONS

CHECK (NumBorrowed < 5)

(c) Add a constraint to the schema that enforces the restriction that the number of books
    assigned to a shelf cannot exceed the shelf capacity.

Solution:
CREATE ASSERTION ShelfCapacity
CHECK ( NOT EXISTS ( SELECT *
    FROM SHELVES S
    WHERE S.Capacity <
        ( SELECT COUNT (*)
            FROM BOOKS B
            WHERE B.ShelfId = S.ShelfId ) ) )
2. (a) Design a schema for a part of a package delivery company, which contains information about packages (PkgId, AddresseeId), addressees (Id, Name, StreetNumber, StreetName, City), and streets (StreetName, City, MinHouseNumber, MaxHouseNumber). Show the primary and foreign keys. Indicate the NOT NULL constraint wherever applicable.

Solution:

CREATE TABLE Package (  
PkgId INTEGER,  
AddresseeId CHAR(20) NOT NULL,  
PRIMARY KEY (PkgId)  
FOREIGN KEY (AddresseeId) REFERENCES Addressee (Id)  
)

CREATE TABLE Addressee (  
Id CHAR(20),  
Name CHAR(20) NOT NULL,  
StreetNumber INTEGER NOT NULL,  
StreetName CHAR(40) NOT NULL,  
City CHAR(20) NOT NULL,  
PRIMARY KEY (Id)  
FOREIGN KEY (StreetName,City) REFERENCES Streets  
)

CREATE TABLE Streets (  
StreetName CHAR(40),  
City CHAR(20),  
MinHouseNumber INTEGER NOT NULL,  
MaxHouseNumber INTEGER NOT NULL,  
PRIMARY KEY (StreetName, City)  
)
(b) Express the constraint that the street number in the addressee’s address must be within the range valid for the corresponding street.

Solution:

```sql
CREATE ASSERTION ValidateRanges
CHECK ( NOT EXISTS
    ( SELECT *
        FROM Address A, Streets S
        WHERE A.StreetName = S.StreetName AND
        A.City = S.City AND
        (A.StreetNumber < S.MinHouseNumber
            OR A.StreetNumber > S.MaxHouseNumber) ) )
```
3. (a) Design a schema for an airline containing the following information.
   
i. the unique Id and name of each passenger
   ii. the unique Id, type (possible value A, B, and C) and capacity of each plane
   iii. the unique Id, date, plane Id, starting location and destination of each flight (a plane
cannot be scheduled for more than one flight on any particular day)
   iv. each reservation made by a passenger on a flight

Show all primary and foreign keys.

Solution:

CREATE TABLE PASSENGERS (
    Id INTEGER,
    Name CHAR(20),
    PRIMARY KEY (Id) )

CREATE TABLE PLANES (
    Id INTEGER,
    Capacity INTEGER,
    Type CHAR(1),
    CHECK (Type IN ('A', 'B', 'C'))
    PRIMARY KEY (Id) )

CREATE TABLE FLIGHTS (
    Id INTEGER,
    PlaneId INTEGER,
    Date DATE,
    Start TIME,
    Destination CHAR(20),
    PRIMARY KEY (Id),
    UNIQUE (PlaneId, Date),
    FOREIGN KEY (PlaneId)
    REFERENCES PLANES (Id) )

CREATE TABLE RESERVATIONS (
    PassengerId INTEGER,
    FlightId INTEGER,
    PRIMARY KEY (PassengerId, FlightId),
    FOREIGN KEY (PassengerId)
    REFERENCES TRAVELLERS (Id),
    FOREIGN KEY (FlightId)
    REFERENCES FLIGHTS (Id) )

(b) Add a constraint that enforces the restriction that the number of passengers on a plane
cannot exceed the plane’s capacity

Solution:
CREATE ASSERTION PlaneCapacity
CHECK ( NOT EXISTS ( SELECT *
  FROM PLANES P, FLIGHTS F
  WHERE P.Id = F.PlaneId AND P.Capacity <
    ( SELECT COUNT (*)
      FROM RESERVATIONS R
      WHERE R.FlightId = F.Id ) ) )

(c) Add a view that shows for each reservation on '6/8/2004' the passenger’s name, the starting location, and the destination.

Solution:

CREATE VIEW Trips (PassName, Start, Destination) AS
SELECT P.Name, F.Start, F.Destination
FROM PASSENGER P, FLIGHTS F, RESERVATIONS R
WHERE P.Id = R.PassengerId AND R.FlightId = F.Id
  AND F.Date = '6/8/2004'
1. Consider the following enterprise, which includes books, authors and publishers. Authors are people with normal attributes, like name, date of birth, etc., but in addition they wrote one or more books. A book has the usual attributes, such as title, ISBN, publication date, etc. Publishers are companies that publish books. They have an address, phone numbers (typically more than one), name, etc.

A book can be written by more than one author, but it can be published by only one publisher. Books do not write themselves and do not publish themselves (hint: these are constraints). An author can write more than one book and to be called an author one, of course, has to write at least one book.

(a) Represent the above as an E-R diagram; include all relevant constraints.

**Solution:**
Assumptions: the author name is unique and the publisher name is unique.
E_R diagram:
(b) Translate the above diagram into the relational model by supplying the appropriate CREATE TABLE statements. Note that ISBN is a 10-digit string (which can have leading zeros), sex can have only two values, 'M' or 'F', and a phone number is a 10 digit number that never starts with a zero. Specify these as domains.

Specify all the key and foreign key constraints. Try to preserve as many participation constraints as possible. List all the participation constraints that are present in the E-R diagram, but not in its translation to SQL.

Solution:

```sql
CREATE DOMAIN ISBN_TYPE CHAR(10)
CHECK (VALUE BETWEEN '0000000000' AND '9999999999').

CREATE DOMAIN SEX_TYPE CHAR(1)
CHECK (VALUE IN ('M', 'F'))

CREATE DOMAIN PHONE_TYPE INTEGER
CHECK (VALUE > 999999999 AND VALUE < 10000000000 )

CREATE TABLE Book (
ISBN ISBN_TYPE,
Title CHAR(60),
PublicationDate DATE,
PName CHAR(60) NOT NULL,
PRIMARY KEY (ISBN),
FOREIGN KEY (PName) REFERENCES Publisher )

CREATE TABLE Author (
AName CHAR(60),
DOB DATE,
Sex SEX_TYPE,
PRIMARY KEY (AName) )

CREATE TABLE Publisher ( 
PName CHAR(60),
Address CHAR(60),
PRIMARY KEY (PName) )

CREATE TABLE Publisher_Phone ( 
PName CHAR(60),
Phone PHONE_TYPE,
PRIMARY KEY (PName, Phone),
FOREIGN KEY (PName) REFERENCES Publisher )

CREATE TABLE Wrote ( 
ISBN ISBN_TYPE,
ISBN ISBN_TYPE,
Title CHAR(60),
PublicationDate DATE,
PName CHAR(60) NOT NULL,
PRIMARY KEY (ISBN),
FOREIGN KEY (PName) REFERENCES Author )

CREATE TABLE Review ( 
Reviewer CHAR(60),
Book ISBN, 
ReviewDate DATE,
Rating INT,
PRIMARY KEY (Reviewer, Book ISBN),
FOREIGN KEY (Reviewer) REFERENCES Author, 
FOREIGN KEY (Book ISBN) REFERENCES Book )
```

AName  CHAR(60),
    PRIMARY KEY  (ISBN, AName),
    FOREIGN KEY  (ISBN) REFERENCES  Books,
    FOREIGN KEY  (AName) REFERENCES  Author
    )

Participation constraints not present in SQL:
  i. Author has to write at least one book.
  ii. Book has to be written by at least one author.

Note that we could have combined Publisher and Publisher.Phone in one table (by just adding the attribute phone). But then the referential integrity constraint that Book.pname references Publisher.pname will no longer be a foreign key constraint in Book, because pname will no longer be a key of Publisher (the key will be (pname, phone)). So, we would have to use an assertion to express this more general inclusion dependency.