COMPLETE VERSION

Database Systems

An Application-Oriented Approach

SECOND EDITION

Michael Kifer, Arthur Bernstein, Philip M. Lewis

Solutions Manual

Contents

PART ONE

Introduction

1

Overview of Databases and Transactions

EXERCISES

This chapter has no exercises.

2

The Big Picture

EXERCISES

2.1 Design the following two tables (in addition to that in Figure 2.1) that might be used in the Student Registration System. Note that the same student Id might appear in many rows of each of these tables.

a. A table implementing the relation CoursesRegisteredFor, relating a student's Id and the identifying numbers of the courses for which she is registered

Solution:

Id	CrsCode
111111111	CSE515
111111111	CSE505
111111111	CSE532
666666666	CSE532
666666666	CSE541
111223344	CSE504
987654321	CSE504
023456789	CSE515
123454321	CSE505

b. A table implementing the relation CoursesTaken, relating a student's Id, the identifying numbers of the courses he has taken, and the grade received in each course

Solution:

Id	CrsCode	Grade
111111111	CSE501	A

111111111	CSE533	B+
666666666	CSE505	A-
666666666	CSE541	C
111223344	CSE533	B-
987654321	CSE515	B+
023456789	CSE505	A
123454321	CSE532	B+

Specify the predicate corresponding to each of these tables.

Solution:
For the first table: Student X is registered for Course Y
For the second table: Student X has taken Course Y and gotten Grade Z

2.2 Write an SQL statement that

a. Returns the Ids of all seniors in the table STUDENT

Solution:

```
SELECT S.Id
FROM Student
WHERE S.Status = 'senior'
```

b. Deletes all seniors from STUDENT

Solution:

```
DELETE
FROM Student S
WHERE S.Status = 'senior'
```

c. Promotes all juniors in the table STUDENT to seniors

Solution:

```
UPDATE Student S
SET S.Status = 'senior'
WHERE S.Status = 'junior'
```

2.3 Write an SQL statement that creates the TRANSCRIPT table.

Solution:

```
CREATE TABLE Transcript  (
StudId INTEGER,
CrsCode CHAR(6),
Semester CHAR(6),
Grade CHAR(1),
PRIMARY KEY (StudId, CrsCode, Semester) )
```

2.4 Using the TRANSCRIPT table, write an SQL statement that

a. Deregisters the student with `Id = 123456789` from the course `CS305` for the fall of 2001

Solution:

```
DELETE
FROM Transcript
WHERE StudId = '123456789'
    AND CrsCode = 'CS305' AND Semester = 'F2001'
```

b. Changes to an A the grade assigned to the student with `Id = 123456789` for the course `CS305` taken in the fall of 2000

Solution:

```
UPDATE Transcript
SET Grade = 'A'
WHERE StudId = '123456789'
    AND CrsCode = 'CS305' AND Semester ='F2000'
```

c. Returns the Id of all students who took `CS305` in the fall of 2000

Solution:

```
SELECT StudId
FROM Transcript
WHERE CrsCode = 'CS305' AND Semester = 'F2000'
```

⋆**2.5** Given the relation MARRIED that consists of tuples of the form $\langle a, b\rangle$, where a is the husband and b is the wife, the relation BROTHER that has tuples of the form $\langle c, d\rangle$, where c is the brother of d, and the relation SIBLING, which has tuples of the form $\langle e, f\rangle$, where e and f are siblings, use SQL to define the relation BROTHER-IN-LAW, where tuples have the form $\langle x, y\rangle$ with x being the brother-in-law of y.
(*Hint*: This query can be represented as a union of three separate SQL queries. SQL provides the operator `UNION` to achieve this effect.)

Solution:
The first SQL query, below, describes the situation where someone is the brother of the wife and hence the brother-in-law of the husband. The second disjunct describes the situation where someone is the brother of the husband and hence the brother-in-law of the wife. The third disjunct describes the situation where, someone is the husband and hence the brother-in-law of all the wife's brothers and sisters.

```
(SELECT Brother.col1, Married.col1
 FROM MARRIED, BROTHER
 WHERE  Brother.col2 = Married.col2)
UNION
(SELECT Brother.col1, Married.col2
 FROM MARRIED, BROTHER
 WHERE  Brother.col2 = Married.col1)
UNION
(SELECT Married.col1, Sibling.col2
 FROM MARRIED, SIBLING
 WHERE  Sibling.col1 = Married.col2)
```

2.6 Write an SQL statement that returns the names (not the Ids) of all students who received an A in `CS305` in the fall of 2000.

Solution:

```
SELECT Name
FROM Student, Transcript
WHERE StudId = Id AND Grade = 'A'
     AND CrsCode = 'CS305' AND Semester = 'F2000'
```

2.7 State whether or not each of the following statements could be an integrity constraint of a checking account database for a banking application. Give reasons for your answers.

a. The value stored in the `balance` column of an account is greater than or equal to `$0`.

Solution:
Yes. It describes a constraint on a snapshot of the database.

b. The value stored in the `balance` column of an account is greater than it was last week at this time.

Solution:
No. It does not describe a snapshot.

c. The value stored in the `balance` column of an account is `$128.32`.

Solution:
No. The balance will change.

d. The value stored in the `balance` column of an account is a decimal number with two digits following the decimal point.

Solution:
Yes. It is a domain constraint.

e. The `social_security_number` column of an account is defined and contains a nine-digit number.

Solution:
Yes. It is a domain constraint.

f. The value stored in the `check_credit_in_use` column of an account is less than or equal to the value stored in the `total_approved_check_credit` column. (These columns have their obvious meanings.)

Solution:
Yes. It describes a constraint on a snapshot

2.8 State five integrity constraints, other than those given in the text, for the database in the Student Registration System.

Solution:

1. The courses for which the student enrolled (registered) must be offered this semester(next semester).
2. An instructor cannot be assigned to two courses taught at the same time in the same semester.
3. Two courses are not taught in the same room at the same time in a given semester.
4. No student must be registered (enrolled) in two courses taught at the same hour.
5. No student must be allowed to register for more than 20 credits in a given semester.
6. The room assigned to a course must have at least as many seats as the maximum allowed enrollment for the course.

2.9 Give an example in the Student Registration System where the database satisfies the integrity constraints IC0–IC3 but its state does not reflect the state of the real world.

Solution:
We register a student, but do not change the database.

2.10 State five (possible) integrity constraints for the database in an airline reservation system.

Solution:

1. The flight for which a person makes a reservation must be on the schedule.
2. The number of reservations on each flight must not exceed the number of seats on the plane.
3. A passenger cannot order two meals
4. The number of meals ordered must equal the number of passengers who ordered meals
5. The same seat on the plane must not be reserved for two passengers.

2.11 A reservation transaction in an airline reservation system makes a reservation on a flight, reserves a seat on the plane, issues a ticket, and debits the appropriate credit card account. Assume that one of the integrity constraints of the reservation database is that the number of reservations on each flight does not exceed the number of seats on the plane. (Of course, many airlines purposely over-book and so do not use this integrity constraint.) Explain how transactions running on this system might violate

a. Atomicity

Solution:
A passenger makes a reservation and reserves a seat. The transaction records the reservation, but the system crashes before it records the seat reservation.

b. Consistency

Solution:
Flight is over-booked (More reservations are made than there are seats on the airplane.)

c. Isolation

Solution:
The same seat is given to two people because of a particular interleaving of the reservation transactions.

d. Durability

Solution:
A reservation is made; the system crashes; the system forgets the reservation

2.12 Describe informally in what ways the following events differ from or are similar to transactions with respect to atomicity and durability.

a. A telephone call from a pay phone (Consider line busy, no answer, and wrong number situations. When does this transaction "commit?")

Solution:
Commit occurs when caller hangs up. Billing information is durable. For line busy or no answer, the transaction aborts. For a wrong number the transaction commits, but later is compensated for by returning the callers money (Read about compensation later in the book,)

b. A wedding ceremony (Suppose that the groom refuses to say "I do." When does this transaction "commit?")

Solution:
Commit occurs when license is signed. Marriage is durable (hopefully).

c. The purchase of a house (Suppose that, after a purchase agreement is signed, the buyer is unable to obtain a mortgage. Suppose that the buyer backs out during the closing. Suppose that two years later the buyer does not make the mortgage payments and the bank forecloses.)

Solution:
Various levels of commit; Every time someone signs something. For example, when purchaser makes an offer to purchase and includes a deposit, he is committed to

either purchase the house at that price (assuming he is approved for the mortgage) or forfeit the deposit. If he is not approved for the mortgage, he is no longer committed to purchase the house and gets his deposit back. If he does not pay his mortgage payment the transaction is compensated for when the bank forecloses.

d. A baseball game (Suppose that it rains.)

Solution:
Commit occurs after game is official. If it rains before the game is official, the game is aborted.

2.13 Assume that, in addition to storing the grade a student has received in every course he has completed, the system stores the student's cumulative GPA. Describe an integrity constraint that relates this information. Describe how the constraint would be violated if the transaction that records a new grade were not atomic.

Solution:
The integrity constraint is that the GPA stored in the database is the GPA of the course grades stored. That constraint could be violated if a transaction updated a course grade and aborted before it could update the GPA.

2.14 Explain how a lost update could occur if, under the circumstances of the previous problem, two transactions that were recording grades for a particular student (in different courses) were run concurrently.

Solution:
The first transaction reads the course grades before the second updated its grade and then updates the GPA. The second transaction reads the course grades before the first updated its grade and then updates the GPA. The first update of the GPA is lost and the final one is incorrect.

PART TWO

Database Management

3

The Relational Data Model

EXERCISES

3.1 Define data atomicity as it relates to the definition of relational databases. Contrast data atomicity with transaction atomicity as used in a transaction processing system.

Solution:
These concepts are not related. Data atomicity means that the relational model does not specify any means for looking into the internal structure of the values, so they appear as indivisible to the relational operators. Transaction atomicity means that the system must ensure that either the transaction runs to completion or, if it does not complete, it has no effect at all. It does not mean that the transaction must be indivisible, but it is a type of all-or-none execution.

3.2 Prove that every relation has a key.

Solution:
Since relations are sets and, thus, cannot have identical elements, the set of all attributes in a relation must be a superkey. If this is not a minimal superkey, some strict subset of it must also be a superkey. Since the number of the attributes in every relation is finite, we will eventually get a minimal superkey, i.e., a key of the relation.

3.3 Define the following concepts:

a. Key

Solution:
A *key*, $\texttt{key}(\bar{K})$, associated with a relation schema, $\mathbf{S}$, is a minimal (by inclusion) subset $\bar{K}$ of attributes of $\mathbf{S}$ with the following property: An instance $\mathbf{s}$ of $\mathbf{S}$ *satisfies* $\texttt{key}(\bar{K})$ if it does not contain a pair of distinct tuples whose values agree on *all* of the attributes in $\bar{K}$.

b. Candidate key

Solution:
Every key of a relation is also called a candidate key for that relation

c. Primary key

Solution:
One of the keys of a relation is designated as primary key.

d. Superkey

Solution:
A superkey is a set of attributes in a relation that contains a key of that relation

3.4 Define

a. Integrity constraint

Solution:
An integrity constraint is an application-specific restriction on the tuples in one or several relations.

b. Static, as compared with dynamic, integrity constraint

Solution:
Static integrity constraints restrict the legal instances of a database. Examples of static ICs are domain constraints, key constraints, foreign-key constraints, etc. Dynamic integrity constraints restrict the evolution (over time) of legal instances of a database, for instance, a salary increase should not exceed 5%.

c. Referential integrity

Solution:
A referential integrity constraint is a requirement that the referenced tuple must exist.

d. Reactive constraint

Solution:
A reactive constraint is a static constraint with a trigger attached. The trigger specifies what to do if the constraint is violated by an update.

e. Inclusion dependency

Solution:
An inclusion dependency is a statement "$\mathbf{S}(\bar{F})$ references $\mathbf{T}(\bar{K})$", which states that for every tuple $s \in \mathbf{s}$, there is a tuple $t \in \mathbf{t}$ that has the same values over the attributes in $\bar{K}$ as does s over the corresponding attributes in $\bar{F}$.

f. Foreign-key constraint

Solution:
A foreign-key constraint is an inclusion dependency in which the set of attributes referred to is a candidate key in the referenced relation.

3.5 Looking at the data that happens to be stored in the tables for a particular application at some particular time, explain whether or not you can tell

a. What the key constraints for the tables are

Solution:
No, in one particular instance of the table, a particular attribute might uniquely identify the rows (and thus appear to be a key), but in other instances it might not.

b. Whether or not a particular attribute forms a key for a particular table

Solution:
You cannot tell whether a particular attribute is a key (for the reasons mentioned in (a)), but you can sometimes tell that a particular attribute cannot form a key by itself (if two distinct rows have the same value over that attribute).

c. What the integrity constraints for the application are

Solution:
No. Again, one cannot determine integrity constraints just by looking at database instances.

d. Whether or not a particular set of integrity constraints is satisfied

Solution:
Yes. Simply check if every constraint in the set is satisfied in the given instance.

3.6 We state in the book that once constraints have been specified in the schema, it is the responsibility of the DBMS to make sure that they are not violated by the execution of any transactions. SQL allows the application to control when each constraint is checked. If a constraint is in *immediate mode*, it is checked immediately after the execution of any SQL statement in a transaction that might make it false. If it is in *deferred mode*, it is not checked until the transaction requests to commit. Give an example where it is necessary for a constraint to be in deferred mode.

Solution:
Suppose the constraint states that the value of one attribute, A, is the sum of the values of two other attributes, B and C. If we want to increment the value of B, we must also increment the value of A in the same transaction. But no matter in what order we do the incrementing, the constraint will be false between the two statements that do the incrementing, and if we used immediate mode checking, the transaction would abort when the first increment statement was attempted.

3.7 Suppose we do not require that all attributes in the primary key are non-null and instead request that, in every tuple, at least one key (primary or candidate) does not have nulls in it. (Tuples can have nulls in other places and the non-null key can be different for different tuples.) Give an example of a relational instance that has two distinct tuples that *might* become one once the values for all nulls become known (that is, are replaced with real values). Explain why this is not possible when one key (such as the primary key) is designated to be non-null for all tuples in the relation.

Solution:
Let the relation have attributes `A` and `B`, each of which is a key. The tuples can be $\langle a, \texttt{NULL}\rangle$ and $\langle \texttt{NULL}, b\rangle$. If the first `NULL` becomes b and the second a then these tuples become the same.

If all tuples are non-`NULL` over the same key, then they must differ over that key somewhere and thus they cannot become the same regardless of what is substituted for nulls in other places.

3.8 Use SQL DDL to specify the schema of the Student Registration System fragment shown in Figure 3.4, including the constraints in Figure 3.6 and Example 3.2.2. Specify SQL domains for attributes with small numbers of values, such as `DeptId` and `Grade`.

Solution:

```
CREATE TABLE  STUDENT (
     Id            INTEGER,
     Name          CHAR(20),
     Address       CHAR(50),
     Status        CHAR(10)
     PRIMARY KEY (Id) )
```

```
CREATE TABLE  PROFESSOR (
     ProfId        INTEGER,
     Name          CHAR(20),
     DeptId        DEPARTMENTS,
     PRIMARY KEY (ProfId) )
```

```
CREATE TABLE  COURSE (
     CrsCode       CHAR(6),
     DeptId        DEPARTMENTS,
     CrsName       CHAR(20),
     Descr         CHAR(100),
     PRIMARY KEY (CrsCode),
     UNIQUE (DeptId, CrsName) )
```

```
CREATE TABLE  TRANSCRIPT (
     StudId        INTEGER,
     CrsCode       CHAR(6),
     Semester      SEMESTERS,
     Grade         GRADES,
     PRIMARY KEY (StudId, CrsCode, Semester),
     FOREIGN KEY (StudId) REFERENCES STUDENT (Id)
          ON DELETE NO ACTION
          ON UPDATE CASCADE,
     FOREIGN KEY (CrsCode) REFERENCES COURSE (CrsCode)
          ON DELETE NO ACTION
          ON UPDATE CASCADE
     FOREIGN KEY (CrsCode, Semester) REFERENCES
                              TEACHING (CrsCode, Semester)
          ON DELETE NO ACTION
          ON UPDATE CASCADE )
```

```
CREATE TABLE  Teaching (
    ProfId        INTEGER,
    CrsCode       CHAR(6),
    Semester      Semesters,
    PRIMARY KEY (CrsCode, Semester),
    FOREIGN KEY (ProfId) REFERENCES Professor(Id)
        ON DELETE NO ACTION
        ON UPDATE CASCADE,
    FOREIGN KEY (CrsCode) REFERENCES Course (CrsCode)
        ON DELETE SET NULL
        ON UPDATE CASCADE )
```

The Grades domain is defined in Section 3.3.6. The domain of departments is defined below.

```
CREATE DOMAIN  Departments CHAR(3)
    CHECK ( VALUE IN ('CS','MAT','EE','MUS','PHY','CHE') )
```

```
CREATE DOMAIN  Semesters CHAR(6)
    CHECK ( VALUE IN ('fall', 'spring', 'summer') )
```

3.9 Consider a database schema with four relations: Supplier, Product, Customer, and Contracts. Both the Supplier and the Customer relations have the attributes `Id`, `Name`, and `Address`. An `Id` is a nine-digit number. Product has `PartNumber` (an integer between 1 and 999999) and `Name`. Each tuple in the Contracts relation corresponds to a contract between a supplier and a customer for a specific product in a certain quantity for a given price.

a. Use SQL DDL to specify the schema of these relations, including the appropriate integrity constraints (primary, candidate, and foreign key) and SQL domains.

Solution:

```
CREATE TABLE  Supplier (
    Id            Suppliers,
    Name          CHAR(20),
    Address       CHAR(50),
    PRIMARY KEY (Id) )
```

```
CREATE TABLE  Customer (
    Id            Customers,
    Name          CHAR(20),
```

```
        Address       CHAR(50),
        PRIMARY KEY (Id) )
```

```
    CREATE TABLE  Product (
        PartNumber    Products,
        Name          CHAR(50),
        PRIMARY KEY (PartNumber) )
```

```
    CREATE TABLE  Contract (
        Customer      Customers,
        Supplier      Suppliers,
        Product       Products,
        Quantity      INTEGER,
        Price         INTEGER,
        PRIMARY KEY (Customer, Supplier, Product),
        FOREIGN KEY (Customer) REFERENCES Customer(Id)
            ON DELETE NO ACTION
            ON UPDATE CASCADE,
        FOREIGN KEY (Supplier) REFERENCES Supplier(Id)
            ON DELETE NO ACTION
            ON UPDATE CASCADE ),
        FOREIGN KEY (Product) REFERENCES Product(PartNumber)
            ON DELETE NO ACTION
            ON UPDATE CASCADE )
```

```
    CREATE DOMAIN  Suppliers INTEGER
        CHECK ( VALUE =< 999999999 AND VALUE > 0 )
```

The domain Customers is defined identically. The domain Products is similar, except that 999999 is used instead of 999999999.

b. Specify the following constraint as an SQL assertion: *there must be more contracts than suppliers.*

Solution:

```
    CREATE ASSERTION  ContractsShaltExceedSuppliers
        CHECK ( (SELECT COUNT(*) FROM Supplier)
                    < (SELECT COUNT(*) FROM Contract) ) )
```

3.10 You have been hired by a video store to create a database for tracking DVDs and videocassettes, customers, and who rented what. The database includes these relations: RentalItem, Customer, and Rentals. Use SQL DDL to specify the schema for this database, including all the applicable constraints. You are free to choose reasonable attributes for the first two relations. The relation Rentals is intended to describe who rented what and should have these attributes: `CustomerId`, `ItemId`, `RentedFrom`, `RentedUntil`, and `DateReturned`.

Solution:

3.11 You are in a real estate business renting apartments to customers. Your job is to define an appropriate schema using SQL DDL. The relations are Property(`Id`, `Address`, `NumberOfUnits`), Unit(`ApartmentNumber`, `PropertyId`, `RentalPrice`, `Size`), Customer (choose appropriate attributes), Rentals (choose attributes; this relation should describe who rents what, since when, and until when), and Payments (should describe who paid for which unit, how much, and when). Assume that a customer can rent more than one unit (in the same or different properties) and that the same unit can be co-rented by several customers.

Solution:
The students must recognize that the key of Unit is ⟨`ApartmentNumber`, `PropertyId`⟩ and not just `ApartmentNumber`. The students should also recognize that both Rentals and Payments should include attributes that reference ⟨`ApartmentNumber`, `PropertyId`⟩. In addition, neither ⟨`ApartmentNumber`, `PropertyId`⟩ nor customer should be a key in Payments.

3.12 You love movies and decided to create a personal database to help you with trivia questions. You chose to have the following relations: Actor, Studio, Movie, and PlayedIn (which actor played in which movie). The attributes of Movie are `Name`, `Year`, `Studio`, and `Budget`. The attributes of PlayedIn are `Movie` and `Actor`. You are free to choose the attributes for the other relations as appropriate. Use SQL DDL to design the schema and all the applicable constraints.

Solution:

3.13 You want to get rich by operating an auction Web site, similar to eBay, at which students can register used textbooks that they want to sell and other students can bid on purchasing those books. The site is to use the same proxy bidding system used by eBay (*http://www.ebay.com*).

Design a schema for the database required for the site. In the initial version of the system, the database must contain the following information:

1. For each book being auctioned: name, authors, edition, ISBN number, bookId (unique), condition, initial offering price, current bid, current maximum bid, auction start date and time, auction end date and time, userId of the seller, userId of the current high bidder, and an indication that the auction is either currently active or complete
2. For each registered user: name, userId (unique), password, and e-mail address

Solution:

3.14 You want to design a room-scheduling system that can be used by the faculty and staff of your department to schedule rooms for events, meetings, classes, etc. Design a schema for the database required for the system. The database must contain the following information:

1. For each registered user: name, userId (unique), password, and e-mail address
2. For each room: room number, start date of the event, start time of the event, duration of the event, repetition of the event (once, daily, weekly, monthly, mon-wed-fri, or tues-thurs), and end date of repetitive event

Solution:

3.15 Design the schema for a library system. The following data should either be contained directly in the system or it should be possible to calculate it from stored information:

1. About each patron: name, password, address, Id, unpaid fines, identity of each book the patron has currently withdrawn, and each book's due date
2. About each book: ISBN number, title, author(s), year of publication, shelfId, publisher, and status (on-shelf, on-loan, on-hold, or on-loan-and-on-hold). For books on-loan the database shall contain the Id of the patron involved and the due date. For books on hold the database shall contain a list of Ids of patrons who have requested the book.
3. About each shelf: shelfId and capacity (in number of books)
4. About each author: year of birth

The system should enforce the following integrity constraints. You should decide whether a particular constraint will be embedded in the schema, and, if so, show how this is done or will be enforced in the code of a transaction.

1. The number of books on a shelf cannot exceed its capacity.
2. A patron cannot withdraw more than two books at a time.
3. A patron cannot withdraw a book if his/her unpaid fines exceed $5. Assume that a book becomes overdue after two weeks and that it accumulates a fine at the rate of $.10 a day.

Solution:

3.16 Suppose that the fragment of the Student Registration System shown in Figure 3.4 has two user accounts: `Student` and `Administrator`. Specify the permissions appropriate for these user categories using the SQL GRANT statement.

Solution:
An administrator should be able to read and modify all information in STUDENT, PROFESSOR, COURSE, TRANSCRIPT, and TEACHING — if your university is like ours, only administrators can change information. Here is an example:

```
GRANT SELECT, INSERT, DELETE, UPDATE ON TRANSCRIPT
    TO Administrator WITH GRANT OPTION
```

Assuming that Ids and addresses are considered private, a student should be able to see names and student status in the STUDENT relation, professors and departments in the PROFESSOR relation, everything in the COURSE relation, nothing in TRANSCRIPT, and, perhaps, `CrsCode` and `Semester` in TEACHING. Here is an example:

```
GRANT SELECT (CrsCode, Semester) ON TEACHING
    TO Student
```

3.17 Suppose that the video store of Exercise 3.10 has the following accounts: `Owner`, `Employee`, and `User`. Specify GRANT statements appropriate for each account.

Solution:
This question is formulated with a pitfall: the student is supposed to recognize that users should not be granted any privileges, because a user should be allowed to see only the data pertaining to that particular user.

`Employee` and `Owner` can have the same permissions. In any case, `Employee` should not be allowed to delete records. `Owner` might be allowed all privileges.

3.18 Explain why the REFERENCES privilege is necessary. Give an example of how it is possible to obtain partial information about the contents of a relation by creating foreign-key constraints referencing that relation.

Solution:
Suppose `Phone` number is a candidate key in the LOAN relation. One can then find out who has loans by creating a relation, PROBE, with the attribute `Phone`, which REFERENCES the `Phone` attribute in LOAN. Then, by exhaustively inserting all possible phone numbers and recording which ones produce errors, such a user would find out the phone numbers of all people who have loans. Reverse telephone lookup using a number of services on the Web will make it possible to also find names and addresses of those people.