Database System Concepts 7th edition Silberschatz Solutions Manual

SKU: 9780078022159-SOLUTIONS

Solution 2.10

A relation schema is a type definition, and a relation is an instance of that schema. For example, *student* (*ss#, name*) is a relation schema and

123-456-222	John
234-567-999	Mary

is a relation based on that schema.

Solution 2.11

No, *s_id* would not be a primary key, since there may be two (or more) tuples for a single student, corresponding to two (or more) advisors. The primary key should then consist of the two attributes *s_id, i_id*.

Solution 2.12

a. The primary keys of the various schemas are underlined. We allow customers to have more than one account, and more than one loan.

branch(*branch_name*, *branch_city*, *assets*)

customer (*ID*, *customer_name*, *customer_street*, *customer_city*)

loan (*loan_number*, *branch_name*, *amount*)

borrower (*ID*, *loan_number*)

account (*account_number*, *branch_name*, *balance*)

depositor (*ID*, *account_number*)

b. The foreign keys are as follows:

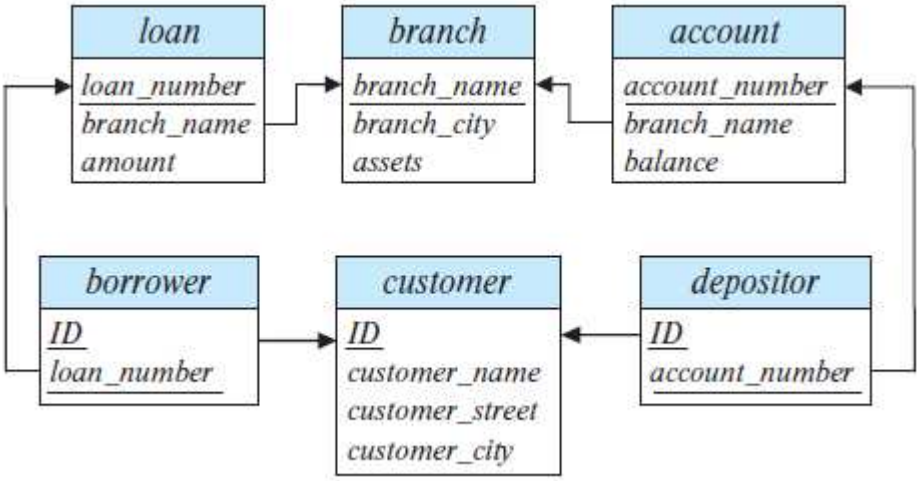
i. For *loan*: *branch_name* referencing *branch*.

ii. For *borrower*: Attribute *ID* referencing *customer* and *loan_number* referencing *loan*

iii. For *account*: *branch_name* referencing *branch*.

iv. For *depositor*: Attribute *ID* referencing customer and *account_number* referencing *account*

Solution 2.13



Solution 2.14

a. $\prod_{ID,\ person\_name} (\sigma_{company\_name = \text{“BigBank”}} (works))$

b. $\prod_{ID,\ person\_name,\ city} (employee \bowtie_{employee.id=works.id} (\sigma_{company\_name = \text{“BigBank”}} (works)))$

c. $\prod_{ID,\ person\_name,\ street,\ city} (\sigma_{(company\_name = \text{“BigBank”} \wedge salary > 10000)} (Works \bowtie_{employee.id=works.id} employee))$

d. $\prod_{ID,\ person\_name} (\sigma_{employee.city=company.city}(employee \bowtie_{employee.ID=works.ID} works \bowtie_{works.company\_name=company.company\_name} \text{company}))$

Solution 2.15

a. $\prod_{loan\_number} (\sigma_{amount > 10000}(loan))$

b. $\prod_{ID} (\sigma_{\text{balance} > 6000}$
$(depositor \bowtie_{depositor.account\_number = account.account\_number} account))$

c. $\prod_{ID} (\sigma_{\text{balance} > 6000 \wedge \text{branch\_name} = \text{“Uptown”}}$
$(depositor \bowtie_{depositor.account\_number = account.account\_number} account))$

Solution 2.16

Nulls may be introduced into the database because the actual value is either unknown or does not exist. For example, an employee whose address has changed and whose new address is not yet known should be retained with a null address.

Solution 2.17

Declarative languages greatly simplify the specification of queries (at least, the types of queries they are designed to handle). They free the user from having to worry about how the query is to be evaluated; not only does this reduce programming effort, but in fact in most situations the query optimizer can do a much better job of choosing the best way to evaluate a query than a programmer working by trial and error.

Both functional and imperative languages require the programmer to specify a specific set of actions. Functional languages have no side-effects, that is, there is no program state to update. This avoids bugs that result from unantipated side effects. Functional languages permit the use of algebraic equivalences in query optimization. The step-by-step execution plan for actually running a database query is usually expressed in an imperative style. Imperative programming is the style most familiar to most programmers.

Solution 23.18

a. $\prod_{ID,name}(\sigma_{dept\_name=}\}Physics"\ (instructor))$

b. $\prod_{ID,name}(instructor \bowtie_{instructor.dept\_name=department.dept\_name}$
$(\sigma_{building=}\}\ Watson"\ (department)))$

c. $\prod_{ID,name}(student \bowtie_{student.ID=takes.ID} takes \bowtie_{takes.course\_id=course.course\_id}$
$\sigma_{dept\_name=}\}\ Comp.\ Sci."\ (course))$

d. $\prod_{ID,name}(student \bowtie_{student.ID=takes.ID} \sigma_{year=2018}\ (takes))$

e. $\prod_{ID,name}(student)-$
$\prod_{ID,name}(student \bowtie_{student.ID=takes.ID} (\sigma_{year=2018}\ (takes)))$

INSTRUCTOR'S MANUAL
TO ACCOMPANY

Database System Concepts

Seventh Edition

Abraham Silberschatz
Yale University

Henry F. Korth
Lehigh University

S. Sudarshan
Indian Institute of Technology, Bombay

April 1, 2019

Preface

This volume is an instructor's manual for the Seventh Edition of *Database System Concepts* by Abraham Silberschatz, Hank Korth, and S. Sudarshan. It consists of answers to the Exercises in the parent text.

Although we have tried to produce an instructor's manual that will aid all of the users of our book as much as possible, there can always be improvements (improved answers, additional questions, sample test questions, programming projects, alternative orders of presentation of the material, additional references, and so on). We invite you to help us in improving this manual. If you have better solutions to the exercises or other items that would be of use with *Database System Concepts*, we invite you to send them to us for consideration in later editions of this manual. All contributions will, of course, be properly credited to their contributor. Email should be addressed to db-book-authors@cs.yale.edu.

A. S.
H. F. K
S. S.

Contents

CHAPTER 1

Introduction

Exercises

1.6 List four applications you have used that most likely employed a database system to store persistent data.

Answer:

- Banking: For account information, transfer of funds, banking transactions.
- Universities: For student information, online assignment submissions, course registrations, and grades.
- Airlines: For reservation of tickets and schedule information.
- Online news sites: For updating news and maintaining archives.
- Online-trade: For product data, availability and pricing information, order-tracking facilities, and generating recommendation lists.

1.7 List four significant differences between a file-processing system and a DBMS.

Answer:
Some main differences between a database management system and a file-processing system are:

- Both systems contain a collection of data and a set of programs which access the data. A database management system coordinates both the physical and the logical access to the data, whereas a file-processing system coordinates only the physical access.
- A database management system reduces the amount of data duplication by ensuring that a physical piece of data is available to all programs authorized to have access to it, whereas data written by one program in a file-processing system may not be readable by another program.

- A database management system is designed to allow flexible access to data (i.e., queries), whereas a file-processing system is designed to allow predetermined access to data (i.e., compiled programs).
- A database management system is designed to coordinate multiple users accessing the same data at the same time. A file-processing system is usually designed to allow one or more programs to access different data files at the same time. In a file-processing system, a file can be accessed by two programs concurrently only if both programs have read-only access to the file.

1.8 Explain the concept of physical data independence and its importance in database systems.

Answer:
Physical data independence is the ability to modify the physical scheme without making it necessary to rewrite application programs. Such modifications include changing from unblocked to blocked record storage, or from sequential to random access files. Such a modification might be adding a field to a record; an application program's view hides this change from the program.

1.9 List five responsibilities of a database-management system. For each responsibility, explain the problems that would arise if the responsibility were not discharged.

Answer:
A general-purpose database-management system (DBMS) has five responsibilities:

a. interaction with the file manager
b. integrity enforcement
c. security enforcement
d. backup and recovery
e. concurrency control

If these responsibilities were not met by a given DBMS (and the text points out that sometimes a responsibility is omitted by design, such as concurrency control on a single-user DBMS for a microcomputer) the following problems can occur, respectively:

a. No DBMS can do without this. If there is no file manager interaction then nothing stored in the files can be retrieved.
b. Consistency constraints may not be satisfied. For example, an instructor may belong to a nonexistent department, two students may have the same ID, account balances could go below the minimum allowed, and so on.

c. Unauthorized users may access the database, or users authorized to access part of the database may be able to access parts of the database for which they lack authority. For example, a low-level user could get access to national defense secret codes, or employees could find out what their supervisors earn (which is presumably a secret).

d. Data could be lost permanently, rather than at least being available in a consistent state that existed prior to a failure.

e. Consistency constraints may be violated despite proper integrity enforcement in each transaction. For example, incorrect bank balances might be reflected due to simultaneous withdrawals and deposits on the same account, and so on.

1.10 List at least two reasons why database systems support data manipulation using a declarative query language such as SQL, instead of just providing a library of C or C++ functions to carry out data manipulation.

Answer:

a. Declarative languages are easier for programmers to learn and use (and even more so for nonprogrammers).

b. The programmer does not have to worry about how to write queries to ensure that they will execute efficiently; the choice of an efficient execution technique is left to the database system. The declarative specification makes it easier for the database system to make a proper choice of execution technique.

1.11 Assume that two students are trying to register for a course in which there is only one open seat. What component of a database system prevents both students from being given that last seat?

Answer:
The concurrency-control manager, which is part of the transaction manager, ensures that at most one student will register successfully.

1.12 Explain the difference between two-tier and three-tier application architectures. Which is better suited for web applications? Why?

Answer:
In a two-tier application architecture, the application runs on the client machine and directly communicates with the database system running on the server. In contrast, in a three-tier architecture, application code running on the client's machine communicates with an application server at the server, and it never directly communicates with the database. The three-tier archicture is better suited for web applications.

1.13 List two features developed in the 2000s and that help database systems handle data-analytics workloads.

Answer:
Traditional database systems store data row-by-row. Because data analytics often focus on only a few columns of a table, column-stores were introduced to allow faster retrieval of those columns actually being used.

Because of the high processing demands of data analytics combined with the broader availability of parallel processing, the map-reduce framework was introduced to facilitate coding parallel data-analytics applications.

1.14 Explain why NoSQL systems emerged in the 2000s, and briefly contrast their features with traditional database systems.

Answer:
NoSQL systems relax the rigidity of storing data in tables by allowing a diverse set of data types. They allow for faster initial application development. However, NoSQL systems lack traditional systems' support for strong data consistency, instead relying on a weaker concept of eventual consistency.

1.15 Describe at least three tables that might be used to store information in a social-networking system such as Facebook.

Answer:
Some possible tables are:

a. A *users* table containing users, with attributes such as account name, real name, age, gender, location, and other profile information.

b. A *content* table containing user-provided content, such as text and images, associated with the user who uploaded the content.

c. A *friends* table recording for each user which other users are connected to that user. The kind of connection may also be recorded in this table.

d. A *permissions* table, recording which categories of friends are allowed to view which content uploaded by a user. For example, a user may share some photos with family but not with all friends.

CHAPTER 2

Introduction to the Relational Model

The relational model remains the primary data model for commercial data-processing applications. It attained its primary position because of its simplicity, which eases the job of the programmer, compared to earlier data models such as the network model or the hierarchical model. It has retained this position by incorporating various new features and capabilities over its half-century of existence. Among those additions are object-relational features such as complex data types and stored procedures, support for XML data, and various tools to support semi-structured data. The relational model's independence from any specific underlying low-level data structures has allowed it to persist despite the advent of new approaches to data storage, including modern column-stores that are designed for large-scale data mining.

In this chapter, we first study the fundamentals of the relational model. A substantial theory exists for relational databases. In Chapter 6 and Chapter 7, we shall examine aspects of database theory that help in the design of relational database schemas, while in Chapter 15 and Chapter 16 we discuss aspects of the theory dealing with efficient processing of queries. In Chapter 27, we study aspects of formal relational languages beyond our basic coverage in this chapter (Section 2.6).

Exercises

2.10 Describe the differences in meaning between the terms *relation* and *relation schema*.

Answer:

A relation schema is a type definition, and a relation is an instance of that schema. For example, *student* (*ss#, name*) is a relation schema and

123-456-222	John
234-567-999	Mary

is a relation based on that schema.

2.11 Consider the *advisor* relation shown in the schema diagram in Figure 2.9, with *s_id* as the primary key of *advisor*. Suppose a student can have more than one advisor. Then, would *s_id* still be a primary key of the *advisor* relation? If not, what should the primary key of *advisor* be?

Answer:

No, *s_id* would not be a primary key, since there may be two (or more) tuples for a single student, corresponding to two (or more) advisors. The primary key should then consist of the two attributes *s_id, i_id*.

2.12 Consider the bank database of Figure 2.18. Assume that branch names and customer names uniquely identify branches and customers, but loans and accounts can be associated with more than one customer.

a. What are the appropriate primary keys?

b. Given your choice of primary keys, identify appropriate foreign keys.

Answer:

a. The primary keys of the various schemas are underlined. We allow customers to have more than one account, and more than one loan.

branch(*branch_name*, *branch_city, assets*)
customer (*ID*,*customer_name*, *customer_street, customer _city*)
loan (*loan_number*, *branch_name, amount*)
borrower (*ID*, *loan_number*)
account (*account_number*, *branch_name, balance*)
depositor (*ID*, *account_number*)

b. The foreign keys are as follows:

i. For *loan*: *branch_name* referencing *branch*.

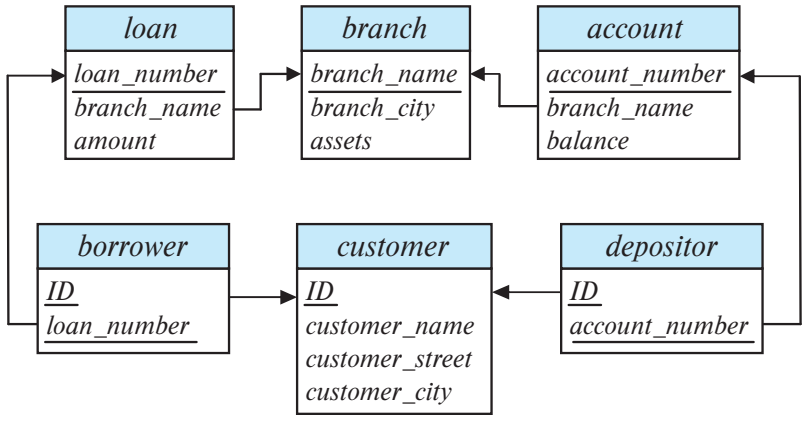
ii. For *borrower*: Attribute *ID* referencing *customer* and *loan_number* referencing *loan*

iii. For *account*: *branch_name* referencing *branch*.

iv. For *depositor*: Attribute *ID* referencing *customer* and *account_number* referencing *account*

2.13 Construct a schema diagram for the bank database of Figure 2.18.

Answer:



2.14 Consider the employee database of Figure 2.17. Give an expression in the relational algebra to express each of the following queries:

a. Find the ID and name of each employee who works for "BigBank".

b. Find the ID, name, and city of residence of each employee who works for "BigBank".

c. Find the ID, name, street address, and city of residence of each employee who works for "BigBank" and earns more than $10000.

d. Find the ID and name of each employee in this database who lives in the same city as the company for which she or he works.

Answer:

a. $\Pi_{ID,\ person\_name}\ (\sigma_{company\_name\ =\ \text{"BigBank"}}\ (works))$

b. $\Pi_{ID,\ person\_name,\ city}\ (employee\ \bowtie_{employee.id=works.id}\ (\sigma_{company\_name\ =\ \text{"BigBank"}}\ (works)))$

c. $\Pi_{ID,\ person\_name,\ street,\ city}\ (\sigma_{(company\_name\ =\ \text{"BigBank"}\ \wedge\ salary\ >\ 10000)}\ (works \bowtie_{employee.id=works.id}\ employee))$

d. $\Pi_{ID,person\_name}\ (\sigma_{employee.city=company.city}(employee \bowtie_{employee.ID=works.ID} works \bowtie_{works.company\_name=company.company\_name} company))$

2.15 Consider the bank database of Figure 2.18. Give an expression in the relational algebra for each of the following queries:

a. Find each loan number with a loan amount greater than $10000.

b. Find the ID of each depositor who has an account with a balance greater than $6000.

c. Find the ID of each depositor who has an account with a balance greater than $6000 at the "Uptown" branch.

Answer:

a. $\Pi_{loan\_number}\ (\sigma_{amount > 10000}(loan))$

b. $\Pi_{ID}\ (\sigma_{balance > 6000}\ (depositor \bowtie_{depositor.account\_number = account.account\_number} account))$

c. $\Pi_{ID}\ (\sigma_{balance > 6000 \wedge branch\_name = \text{"Uptown"}}\ (depositor \bowtie_{depositor.account\_number = account.account\_number} account))$

2.16 List two reasons why null values might be introduced into a database.

Answer:
Nulls may be introduced into the database because the actual value is either unknown or does not exist. For example, an employee whose address has changed and whose new address is not yet known should be retained with a null address.

2.17 Discuss the relative merits of imperative, functional, and declarative languages.

Answer:
Declarative languages greatly simplify the specification of queries (at least, the types of queries they are designed to handle). They free the user from having to worry about how the query is to be evaluated; not only does this reduce programming effort, but in fact in most situations the query optimizer can do a much better job of choosing the best way to evaluate a query than a programmer working by trial and error.
Both functional and imperative languages require the programmer to specify a specific set of actions. Functional languages have no side-effects, that is, there is no program state to update. This avoids bugs that result from unantipated side effects. Functional languages permit the use of algebraic equivalences in query optimization. The step-by-step execution plan for actually running a database query is usually expressed in an imperative style. Imperative programming is the style most familiar to most programmers.

2.18 Write the following queries in relational algebra, using the university schema.

a. Find the ID and name of each instructor in the Physics department.

b. Find the ID and name of each instructor in a department located in the building "Watson".

c. Find the ID and name of each student who has taken at least one course in the "Comp. Sci." department.

d. Find the ID and name of each student who has taken at least one course section in the year 2018.

e. Find the ID and name of each student who has not taken any course section in the year 2018.

Answer:

a. $\Pi_{ID,name}(\sigma_{dept\_name=\}\}Physics''}(instructor))$

b. $\Pi_{ID,name}(instructor \bowtie_{instructor.dept\_name=department.dept\_name} (\sigma_{building=\}\}Watson''}(department)))$

c. $\Pi_{ID,name}(student \bowtie_{student.ID=takes.ID} takes \bowtie_{takes.course\_id=course.course\_id} \sigma_{dept\_name=\}\}Comp.\ Sci.''}(course))$

d. $\Pi_{ID,name}(student \bowtie_{student.ID=takes.ID} \sigma_{year=2018}(takes))$

e. $\Pi_{ID,name}(student) - \Pi_{ID,name}(student \bowtie_{student.ID=takes.ID} (\sigma_{year=2018}(takes)))$