INSTRUCTOR'S MANUAL
TO ACCOMPANY

Database System Concepts

Sixth Edition

Abraham Silberschatz
Yale University

Henry F. Korth
Lehigh University

S. Sudarshan
Indian Institute of Technology, Bombay

Contents

Preface

This volume is an instructor's manual for the 6^{th} edition of *Database System Concepts* by Abraham Silberschatz, Henry F. Korth and S. Sudarshan. It contains answers to the exercises at the end of each chapter of the book. (Beginning with the 5^{th} edition, solutions for Practice Exercises have been made available on the Web; to avoid duplication, these are not included in the instructors manual.)

Before providing answers to the exercises for each chapter, we include a few remarks about the chapter. The nature of these remarks vary. They include explanations of the inclusion or omission of certain material, and remarks on how we teach the chapter in our own courses. The remarks also include suggestions on material to skip if time is at a premium, and tips on software and supplementary material that can be used for programming exercises.

The Web home page of the book, at http://www.db-book.com, contains a variety of useful information, including laboratory relation information such as sample data, lab exercises, and links to database software, online appendices describing the network data model, the hierarchical data model, and advanced relational database design, model course syllabi, and last but not least, up-to-date errata. We will periodically update the page with supplementary material that may be of use to teachers and students.

We would appreciate it if you would notify us of any errors or omissions in the book, as well as in the instructor's manual. Internet electronic mail should be addressed to db-book-authors@cs.yale.edu. Physical mail may be sent to Avi Silberschatz, Department of Computer Science, Yale University, 51 Prospect Street, P.O. Box 208285, New Haven, CT, 06520, USA.

Although we have tried to produce an instructor's manual which will aid all of the users of our book as much as possible, there can always be improvements. These could include improved answers, additional questions, sample test questions, programming projects, suggestions on alternative orders of presentation of the material, additional references, and so on. If you would like to suggest any such improvements to the book or the instructor's manual, we would be glad to hear from you. All contributions that we make use of will, of course, be properly credited to their contributor.

Several students at IIT Bombay contributed to the instructor manual for the 6^{th} edition, including Mahendra Chavan, Karthik Ramachandra, Bikmal Harikrishna, Ankush Jain, Manas Joglekar, Parakram Majumdar, Prashant Sachdeva, and Nisarg Shah. This manual is derived from the manuals for the earlier editions. John Corwin and Swathi Yadlapalli did the bulk of the work in preparing the instructors manual for the 5^{th} edition. The manual for the 4^{th} edition was prepared by Nilesh Dalvi, Sumit Sanghai, Gaurav Bhalotia and Arvind Hulgeri. The manual for the 3^{th} edition was prepared by K. V. Raghavan with help from Prateek R. Kapadia. Sara Strandtman helped with the instructor manual for the 2^{nd} and 3^{rd} editions, while Greg Speegle and Dawn Bezviner helped us to prepare the instructor's manual for the 1^{th} edition.

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

CHAPTER 1

Introduction

Chapter 1 provides a general overview of the nature and purpose of database systems. The most important concept in this chapter is that database systems allow data to be treated at a high level of abstraction. Thus, database systems differ significantly from the file systems and general purpose programming environments with which students are already familiar. Another important aspect of the chapter is to provide motivation for the use of database systems as opposed to application programs built on top of file systems. Thus, the chapter motivates what the student will be studying in the rest of the course.

The idea of abstraction in database systems deserves emphasis throughout, not just in discussion of Section 1.3. The overview of the structure of databases is, of necessity, rather brief, and is meant only to give the student a rough idea of some of the concepts. The student may not initially be able to fully appreciate the concepts described here, but should be able to do so by the end of the course.

The specifics of the E-R, relational, and object-oriented models are covered in later chapters. These models can be used in Chapter 1 to reinforce the concept of abstraction, with syntactic details deferred to later in the course.

If students have already had a course in operating systems, it is worthwhile to point out how the OS and DBMS are related. It is useful also to differentiate between concurrency as it is taught in operating systems courses (with an orientation towards files, processes, and physical resources) and database concurrency control (with an orientation towards granularity finer than the file level, recoverable transactions, and resources accessed associatively rather than physically). If students are familiar with a particular operating system, that OS's approach to concurrent file access may be used for illustration.

Exercises

1.7 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 new, maintainence of archives.
- Online-trade: For product data, availability and pricing informations, order-tracking facilities, and generating recommendation lists.

1.8 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 that 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 pre-determined 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.

1.9 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.

1.10 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 micro computer) 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 non-existent 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.

1.11 List at least two reasons why database systems support data manipulation using a declarative query language such as SQL, instead of just providing a 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 non-programmers).

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.

1.12 Explain what problems are caused by the design of the table in Figure 1.4.
Answer:

- If a department has more than one instructor, the building name and budget get repeated multiple times. Updates to the building name and budget may get performed on some of the copies but not others, resulting in an inconsistent state where it is not clear what is the actual building name and budget of a department.
- A department needs to have at least one instructor in order for building and budget information to be included in the table. Nulls can be used when there is no instructor, but null values are rather difficult to handle.
- If all instructors in a department are deleted, the building and budget information are also lost. Ideally, we would like to have the department information in the database irrespective of whether the department has an associated instructor or not, without resorting to null values.

1.13 What are five main functions of a database administrator?
Answer:

- To backup data
- In some cases, to create the schema definition
- To define the storage structure and access methods
- To modify the schema and/or physical organization when necessary
- To grant authorization for data access
- To specify integrity constraints

1.14 Explain the difference between two-tier and three-tier 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 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 never directly communicates with the database. The three-tier archicture is better suited for Web applications.

1.15 Describe at least 3 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 category 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

This chapter presents the relational model and a brief introduction to the relational-algebra query language. The short introduction to relational algebra is sufficient for courses that focus on application development, without going into database internals. In particular, the chapters on SQL do not require any further knowledge of relational algebra. However, courses that cover internals, in particular query processing, require a more detailed coverage of relational algebra, which is provided in Chapter 6.

Exercises

2.9 Consider the bank database of Figure 2.15.

a. What are the appropriate primary keys?

employee (*person_name*, *street*, *city*)
works (*person_name*, *company_name*, *salary*)
company (*company_name*, *city*)

Figure 2.14 Relational database for Exercises 2.1, 2.7, and 2.12.

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

Figure 2.15 Banking database for Exercises 2.8, 2.9, and 2.13.

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

Answer:

a. The primary keys of the various schema are underlined. Although in a real bank the customer name is unlikely to be a primary key, since two customers could have the same name, we use a simplified schema where we assume that names are unique. We allow customers to have more than one account, and more than one loan.

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

b. The foreign keys are as follows
   i. For *loan*: *branch_name* referencing *branch*.
   ii. For *borrower*: Attribute *customer_name* referencing *customer* and *loan_number* referencing *loan*
   iii. For *account*: *branch_name* referencing *branch*.
   iv. For *depositor*: Attribute *customer_name* referencing *customer* and *account_number* referencing *account*

2.10 Consider the *advisor* relation shown in Figure 2.8, 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 be *s_id, i_id*.

2.11 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.

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

a. Find the names of all employees who work for "First Bank Corporation".

b. Find the names and cities of residence of all employees who work for "First Bank Corporation".

c. Find the names, street address, and cities of residence of all employees who work for "First Bank Corporation" and earn more than $10,000.

Answer:

a. $\Pi_{person\_name}(\sigma_{company\_name = \text{"First Bank Corporation"}}(works))$

b. $\Pi_{person\_name, city}(employee \bowtie (\sigma_{company\_name = \text{"First Bank Corporation"}}(works)))$

c. $\Pi_{person\_name, street, city}(\sigma_{(company\_name = \text{"First Bank Corporation"} \wedge salary > 10000)}(works \bowtie employee))$

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

a. Find all loan numbers with a loan value greater than $10,000.

b. Find the names of all depositors who have an account with a value greater than $6,000.

c. Find the names of all depositors who have an account with a value greater than $6,000 at the "Uptown" branch.

Answer:

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

b. $\Pi_{customer\_name}(\sigma_{balance > 6000}(depositor \bowtie account))$

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

2.14 List two reasons why null values might be introduced into the 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. If employee tuples have a composite attribute *dependents*, and a particular employee has no dependents, then that tuple's *dependents* attribute should be given a null value.

2.15 Discuss the relative merits of procedural and nonprocedural languages.

Answer: Nonprocedural languages greatly simplify the specification of queries (at least, the types of queries they are designed to handle). The 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 task of choosing the best way to evaluate a query than a programmer working by trial and error.

On the other hand, procedural languages are far more powerful in terms of what computations they can perform. Some tasks can either not be

done using nonprocedural languages, or are very hard to express using nonprocedural languages, or execute very inefficiently if specified in a nonprocedural manner.