CENGAGE

16e

Problem-Solving Cases in Microsoft® Access® & Excel®

MONK | BRADY | MENDELSOHN

PROBLEM-SOLVING CASES IN MICROSOFT® ACCESS™ AND EXCEL®

PROBLEM-SOLVING CASES IN MICROSOFT® ACCESS™ AND EXCEL®

Sixteenth Annual Edition

Ellen F. Monk
Joseph A. Brady
Emilio I. Mendelsohn

CENGAGE

Australia • Brazil • Mexico • Singapore • United Kingdom • United States

CENGAGE

***Problem-Solving Cases in Microsoft® Access™ and Excel®*, 16th Annual Edition**
Ellen F. Monk, Joseph A. Brady, Emilio I. Mendelsohn

SVP, Higher Education Product Management: Erin Joyner

VP, Product Management: Mike Schenk

Product Director: Lauren Murphy

Product Manager: Jaymie Falconi

Product Assistant: Anna Goulart

Vice President, Marketing – Science, Technology, & Math: Jason Sakos

Senior Marketing Director: Michele McTighe

Executive Marketing Manager: Jill Staut

Marketing Development Manager: Samantha Best

Product Specialist: Danielle Klahr

Director, Content Creation: Juliet Steiner

Senior Manager, Content Creation: Patty Stephan

Content Manager: Michele Stulga

Developmental Editor: Dan Seiter

Technical Editor: John Freitas

Designer: Lizz Anderson

Production Service/Composition: Lumina Datamatics Ltd.

Cover image: Peach Mango/Shutterstock.com

The authors acknowledge **www.fakenamegenerator.com**, which can be used to generate data for case scenarios.

Library of Congress Control Number: 2019938234

ISBN: 978-0-357-13863-2

Cengage
20 Channel Street
Boston, MA 02210
USA

Printed in the United States of America
Print Number: 01 Print Year: 2019

To Karen: Thanks for your support!

—JAB

To my problem-solving students.

—EFM

To our little bean: Although we never met you, we loved you all the same. Mom and Dad

—EIM

BRIEF CONTENTS

PREFACE

For more than two decades, we have taught MIS courses at the university level. From the start, we wanted to use good computer-based case studies for the database and decision-support portions of our courses.

At first, we could not find a casebook that met our needs! This surprised us because we thought our requirements were not unreasonable. First, we wanted cases that asked students to think about real-world business situations. Second, we wanted cases that provided students with hands-on experience, using the kind of software that they had learned to use in their computer literacy courses—and that they would later use in business. Third, we wanted cases that would strengthen students' ability to analyze a problem, examine alternative solutions, and implement a solution using software. Undeterred by the lack of casebooks, we wrote our own for Cengage.

This is the sixteenth casebook we have written for Cengage. The cases are all new, and the tutorials have been updated using Microsoft Office 2019.

As with our prior casebooks, we include tutorials that prepare students for the cases, which are challenging but doable. The cases are organized to help students think about the logic of each case's business problem and then about how to use the software to solve the business problem. The cases fit well in an undergraduate MIS course, an MBA information systems course, or a computer science course devoted to business-oriented programming.

To date we have written nearly 200 cases. We are proud to say that all of them have tried to teach students how to use software to help them make good business decisions. The business computing environment continues to evolve, and our cases should follow that evolution. To stay current, students need to learn a broader range of data analysis activities. Therefore, we have increased the range of topics covered in our casebook. This edition contains instruction and a case on the differences between handling operational data and analytical data. We have placed greater emphasis throughout the Excel portion of the text on the various uses of pivot tables. In the future, we intend to further increase the range of data analysis topics we cover. We welcome feedback from instructors on this process. If there are additional topics you want to see covered in future editions or topics that should be covered differently, let us know! Please contact Jaymie Falconi, our product manager at Cengage.

BOOK ORGANIZATION

The book is organized into five parts:

- Database cases using Access
- Decision support cases using the Excel Scenario Manager
- Decision support cases using Excel Solver
- Integration cases and data analysis using Access and Excel
- Advanced Excel skills

Part 1 begins with two tutorials that prepare students for the Access case studies. Parts 2 and 3 each begin with a tutorial that prepares students for the Excel case studies. All four tutorials provide students with practice in using the software's features—the kind of support that other books about Access and Excel do not provide. Part 4 challenges students to use both Access and Excel to find a solution to a business problem. Part 5 is a tutorial about advanced skills students might need to complete some of the Excel cases. The next sections explore these parts of the book in more depth.

Part 1: Database Cases Using Microsoft Access

This section begins with two tutorials and then presents five case studies.

Tutorial A: Database Design

This tutorial helps students understand how to set up tables to create a database without requiring students to learn formal analysis and design methods, such as data normalization.

Tutorial B: Microsoft Access

The second tutorial teaches students the more advanced features of Access queries and reports—features that students will need to know to complete the cases.

Cases 1–5

Five database cases follow Tutorials A and B. The students must use the Access database in each case to create forms, queries, and reports that help management. The first case is an easier "warm-up" case. The next four cases require more effort to design the database and implement the results.

Part 2: Decision Support Cases Using Microsoft Excel Scenario Manager

This section has one tutorial and two decision support cases that require the use of the Excel Scenario Manager.

Tutorial C: Building a Decision Support System in Excel

This section begins with a tutorial that uses Excel to explain decision support and fundamental concepts of spreadsheet design. The case emphasizes the use of Scenario Manager to organize the output of multiple "what-if" scenarios.

Cases 6–7

Students can complete these two cases with Scenario Manager. In each case, students must use Excel to model two or more solutions to a problem. Students then use the model outputs to identify and document the preferred solution in a memo.

Part 3: Decision Support Cases Using Microsoft Excel Solver

This section has one tutorial and two decision support cases that require the use of Excel Solver.

Tutorial D: Building a Decision Support System Using Microsoft Excel Solver

This section begins with a tutorial for using Excel Solver, a powerful decision support tool for solving optimization problems.

Cases 8–9

Students use the Excel Solver tool in each case to analyze alternatives and identify and document the preferred solution.

Part 4: Integration Cases Using Microsoft Access and Excel

Cases 10–12

These cases integrate Access and Excel. In Case 10, students will analyze data using Access queries, an Excel table, and an Excel pivot table. Cases 11 and 12 show students how to share data between Access and Excel to solve problems and to use the fundamentals of data analysis. For example, Case 12 illustrates the basic differences between operational and analytical data and how to minimize operational impacts during data analysis.

Part 5: Advanced Skills Using Microsoft Excel

This part contains one tutorial that focuses on using advanced techniques in Excel.

Tutorial E: Guidance for Excel Cases

Some cases may require the use of Excel techniques that are not discussed in other tutorials or cases in this casebook. For example, techniques for using tables and pivot tables are explained in Tutorial E rather than in the cases themselves.

Note that previous editions of the book included a Tutorial F for guidance on giving an oral presentation. This tutorial is now available at the Cengage Web site. To access the tutorial, see "Using the Cases and Tutorials" below.

INDIVIDUAL CASE DESIGN

The format of the cases uses the following template:

- Each case begins with a *Preview* and an overview of the tasks.
- The next section, *Preparation*, tells students what they need to do or know to complete the case successfully. Again, the tutorials also prepare students for the cases.
- The third section, *Background*, provides the business context that frames the case. The background of each case models situations that require the kinds of thinking and analysis that students will need in the business world.
- The *Assignment* sections are generally organized to help students develop their analyses.
- The last section, *Deliverables*, lists the finished materials that students must hand in: printouts, a memorandum, and files. The list is similar to the deliverables that a business manager might demand.

USING THE CASES AND TUTORIALS

We have successfully used cases like these in our undergraduate MIS courses. We usually begin the semester with Access database instruction. We assign the Access database tutorials and then a case to each student. Then, to teach students how to use the Excel decision support system, we do the same thing: We assign a tutorial and then a case.

Some instructors have asked for access to extra cases, especially in the second semester of a school year. For example, they assigned the integration case in the fall, and they need another one for the spring. To meet this need, we have set up an online "Hall of Fame" that features some of our favorite cases from prior editions. These password-protected cases are available to instructors on the Instructor Companion Site for this text. Go to *www.cengage.com* to log in to your instructor account and search for this textbook by title, author, or ISBN. Note that the cases are in Microsoft Office 2016 format.

TECHNICAL INFORMATION

The cases in this textbook were written using Microsoft Office 2019, and the textbook was tested for quality assurance using the Windows 10 operating system, Microsoft Access 2019, and Microsoft Excel 2019.

DATA FILES AND SOLUTION FILES

We have created "starter" data files as needed for some of the Excel cases, so students need not spend time typing in the spreadsheet skeleton. Other cases ask students to load Access starter files. All of these files are on the Instructor and Student Companion Sites for this text. To access the Instructor or Student Companion Site, go to *www.cengage.com*, log in to your account, and search for this textbook by title, author, or ISBN. You are granted a license to copy the data files to any computer or computer network used by people who have purchased this textbook.

Solutions to the material in the text are available to instructors on the Instructor Companion Site for this text at *www.cengage.com*. Search for this textbook by title, author, or ISBN. The solutions are password protected.

ACKNOWLEDGMENTS

We would like to give many thanks to the team at Cengage, including our product manager, Jaymie Falconi; our developmental editor, Dan Seiter; our technical editor, John Freitas; and our content manager, Michele Stulga. As always, we acknowledge our students' diligent work.

PART 1

DATABASE CASES USING MICROSOFT ACCESS

TUTORIAL **A**

DATABASE DESIGN

This tutorial has three sections. The first section briefly reviews basic database terminology. The second section teaches database design. The third section features a database design problem for practice.

REVIEW OF TERMINOLOGY

You will begin by reviewing some basic terms that will be used throughout this textbook. In Access, a **database** is a group of related objects that are saved in one file. An Access **object** can be a table, form, query, or report. You can identify an Access database file by its suffix, .accdb.

A **table** consists of data that is arrayed in rows and columns. A **row** of data is called a **record**. A **column** of data is called a **field**. Thus, a record is a set of related fields. The fields in a table should be related to one another in some way. For example, a company might want to keep its employee data together by creating a database table called Employee. That table would contain data fields about employees, such as their names and addresses. It would not have data fields about the company's customers; that data would go in a Customer table.

A field's values have a **data type** that is declared when the table is defined. Thus, when data is entered into the database, the software knows how to interpret each entry. Data types in Access include the following:

- *Text* for words
- *Integer* for whole numbers
- *Double* for numbers that have a decimal value
- *Currency* for numbers that represent dollars and cents
- *Yes/No* for variables that have only two values (such as 1/0, on/off, yes/no, and true/false)
- *Date/Time* for variables that are dates or times

Each database table should have a **primary key** field—a field in which each record has a *unique* value. For example, in an Employee table, a field called Employee Identification Number (EIN) could serve as a primary key. (This assumes that each employee is given a number when hired and that these numbers are not reused later.) Sometimes, a table does not have a single field whose values are all different. In that case, two or more fields are combined into a **compound primary key**. The combination of the fields' values is unique.

Database tables should be logically related to one another. For example, suppose a company has an Employee table with fields for EIN, Name, Address, and Telephone Number. For payroll purposes, the company has an Hours Worked table with a field that summarizes Labor Hours for individual employees. The relationship between the Employee table and Hours Worked table needs to be established in the database, so you can determine the number of hours worked by any employee. To create this relationship, you include the primary key field from the Employee table (EIN) as a field in the Hours Worked table. In the Hours Worked table, the EIN field is then called a **foreign key** because it's from a "foreign" table.

In Access, data can be entered directly into a table, or it can be entered into a form, which then inserts the data into a table. A **form** is a database object that is created from an existing table to make the process of entering data more user-friendly.

A **query** is the database equivalent of a question that is posed about data in a table (or tables). For example, suppose a manager wants to know the names of employees who have worked for the company for more than five years. A query could be designed to search the Employee table for the information. The query would be run, and its output would answer the question.

Queries can be designed to search multiple tables at a time. For this to work, the tables must be connected by a **join** operation, which links tables on the values in a field that they have in common. The common field acts as a "hinge" for the joined tables; when the query is run, the query generator treats the joined tables as one large table.

In Access, queries that answer a question are called *select queries* because they select relevant data from the database records. Queries can also be designed to change data in records, add a record to the end of a table, or delete entire records from a table. These queries are called **update**, **append**, and **delete** queries, respectively.

Access has a **report** generator that can be used to format a table's data or a query's output.

DATABASE DESIGN

Designing a database involves determining which tables belong in the database and then creating the fields that belong in each table. This section begins with an introduction to key database design concepts, then discusses design rules you should use when building a database. First, the following key concepts are defined:

- Entities
- Relationships
- Attributes

Database Design Concepts

Computer scientists have highly formalized ways of documenting a database's logic. Learning their notations and mechanics can be time-consuming and difficult. In fact, doing so usually takes a good portion of a systems analysis and design course. This tutorial will teach you database design by emphasizing practical business knowledge; the approach should enable you to design serviceable databases quickly. Your instructor may add more formal techniques.

A database models the logic of an organization's operation, so your first task is to understand the operation. You can talk to managers and workers, make your own observations, and look at business documents, such as sales records. Your goal is to identify the business's "entities" (sometimes called *objects*). An **entity** is a thing or event that the database will contain. Every entity has characteristics, called **attributes**, and one or more **relationships** to other entities. Let's take a closer look.

Entities

As previously mentioned, an entity is a tangible thing or an event. The reason for identifying entities is that *an entity eventually becomes a table in the database*. Entities that are things are easy to identify. For example, consider a car rental agency. The database for the agency would probably need to contain the names of vehicles and the names of customers who rent them, so you would have one entity named Vehicles and another named Customer.

In contrast, entities that are events can be more difficult to identify, probably because they are more conceptual. However, events are real, and they are important. In the rental agency example, one event would be Vehicle Rental and another event would be Hours Worked by employees.

In general, your analysis of an organization's operations is made easier when you realize that organizations usually have physical entities such as these:

- Employees
- Customers
- Inventory (products or services)
- Suppliers

Thus, the database for most organizations would have a table for each of these entities. Your analysis also can be made easier by knowing that organizations engage in transactions internally (within the company) and externally (with the outside world). Such transactions are explained in an introductory accounting course, but most people understand them from events that occur in daily life. Consider the following examples:

- Organizations generate revenue from sales or interest earned. Revenue-generating transactions include event entities called Sales and Interest Earned.
- Organizations incur expenses from paying hourly employees and purchasing materials from suppliers. Hours Worked and Purchases are event entities in the databases of most organizations.

Thus, identifying entities is a matter of observing what happens in an organization. Your powers of observation are aided by knowing what entities exist in the databases of most organizations.

Relationships

As an analyst building a database, you should consider the relationship of each entity to the other entities you have identified. For example, a college database might contain entities for Student, Course, and Section to contain data about each. A relationship between Student and Section could be expressed as "Students enroll in sections."

An analyst also must consider the **cardinality** of any relationship. Cardinality can be one-to-one, one-to-many, or many-to-many:

- In a one-to-one relationship, one instance of the first entity is related to just one instance of the second entity.
- In a one-to-many relationship, one instance of the first entity is related to many instances of the second entity, but each instance of the second entity is related to only one instance of the first.
- In a many-to-many relationship, one instance of the first entity is related to many instances of the second entity, and one instance of the second entity is related to many instances of the first.

For a more concrete understanding of cardinality, consider again the college database with the Student, Course, and Section entities. The university catalog shows that a course such as Accounting 101 can have more than one section: 01, 02, 03, 04, and so on. Thus, you can observe the following relationships:

- The relationship between the entities Course and Section is one-to-many. Each course has many sections, but each section is associated with just one course.
- The relationship between Student and Section is many-to-many. Each student can be in more than one section, because each student can take more than one course. Also, each section has more than one student.

Thinking about relationships and their cardinalities may seem tedious to you. However, as you work through the cases in this text, you will see that this type of analysis can be valuable in designing databases. In the case of many-to-many relationships, you should determine the tables a given database needs; in the case of one-to-many relationships, you should decide which fields the tables need to share.

Attributes

An attribute is a characteristic of an entity. You identify attributes of an entity because *attributes become a table's fields*. If an entity can be thought of as a noun, an attribute can be considered an adjective that describes the noun. Continuing with the college database example, consider the Student entity. Students have names, so Last Name would be an attribute of the Student entity and therefore a field in the Student table. First Name would be another attribute, as well as Address, Phone Number, and other descriptive fields.

Sometimes, it can be difficult to tell the difference between an attribute and an entity, but one good way is to ask whether more than one attribute is possible for each entity. If more than one instance is possible, but you do not know the number in advance, you are working with an entity. For example, assume that a student could have a maximum of two addresses—one for home and one for college. You could specify attributes Address 1 and Address 2. Next, consider that you might not know the number of student addresses in advance, meaning that all addresses have to be recorded. In that case, you would not know how many fields to set aside in the Student table for addresses. Therefore, you would need a separate Student Addresses table (entity) that would show any number of addresses for a given student.

Database Design Rules

As described previously, your first task in database design is to understand the logic of the business situation. Once you understand this logic, you are ready to build the database. To create a context for learning about database design, look at a hypothetical business operation and its database needs.

Example: The Talent Agency

Suppose you have been asked to build a database for a talent agency that books musical bands into nightclubs. The agent needs a database to keep track of the agency's transactions and to answer day-to-day questions. For example, a club manager often wants to know which bands are available on a certain date at a certain time, or wants to know the agent's fee for a certain band. The agent may want to see a list of all band members and the instrument each person plays, or a list of all bands that have three members.

Suppose that you have talked to the agent and have observed the agency's business operation. You conclude that your database needs to reflect the following facts:

1. A booking is an event in which a certain band plays in a particular club on a particular date, starting and ending at certain times and performing for a specific fee. A band can play more than once a day. The Heartbreakers, for example, could play at the East End Cafe in the afternoon and then at the West End Cafe on the same night. For each booking, the club pays the talent agent. The agent keeps a 5 percent fee and then gives the remainder of the payment to the band.
2. Each band has at least two members and an unlimited maximum number of members. The agent notes a telephone number of just one band member, which is used as the band's contact number. No two bands have the same name or telephone number.
3. Band member names are not unique. For example, two bands could each have a member named Sally Smith.
4. The agent keeps track of just one instrument that each band member plays. For the purpose of this database, "vocals" are considered an instrument.
5. Each band has a desired fee. For example, the Lightmetal band might want $700 per booking and would expect the agent to try to get at least that amount.
6. Each nightclub has a name, an address, and a contact person. The contact person has a telephone number that the agent uses to call the club. No two clubs have the same name, contact person, or telephone number. Each club has a target fee. The contact person will try to get the agent to accept that fee for a band's appearance.
7. Some clubs feed the band members for free; others do not.

Before continuing with this tutorial, you might try to design the agency's database on your own. Ask yourself: What are the entities? Recall that business databases usually have Customer, Employee, and Inventory entities, as well as an entity for the event that generates revenue transactions. Each entity becomes a table in the database. What are the relationships among the entities? For each entity, what are its attributes? For each table, what is the primary key?

Six Database Design Rules

Assume that you have gathered information about the business situation in the talent agency example. Now you want to identify the tables required for the database and the fields needed in each table. Observe the following six rules:

Rule 1: You do not need a table for the business. The database represents the entire business. Thus, in the example, Agent and Agency are not entities.

Rule 2: Identify the entities in the business description. Look for typical things and events that will become tables in the database. In the talent agency example, you should be able to observe the following entities:

- *Things*: The product (inventory for sale) is Band. The customer is Club.
- *Events*: The revenue-generating transaction is Bookings.

You might ask yourself: Is there an Employee entity? Isn't Instrument an entity? Those issues will be discussed as the rules are explained.

Rule 3: Look for relationships among the entities. Look for one-to-many relationships between entities. The relationship between those entities must be established in the tables, using a foreign key. For details, see the following discussion in Rule 4 about the relationship between Band and Band Member.

Look for many-to-many relationships between entities. Each of these relationships requires a third entity that associates the two entities in the relationship. Recall the many-to-many relationship from the college database scenario that involved Student and Section entities. To display the enrollment of specific students in specific sections, a third table would be required. The mechanics of creating such a table are described in Rule 4 during the discussion of the relationship between Band and Club.

Rule 4: Look for attributes of each entity and designate a primary key. As previously mentioned, you should think of the entities in your database as nouns. You should then create a list of adjectives that describe those nouns. These adjectives are the attributes that will become the table's fields. After you have identified fields for each table, you should check to see whether a field has unique values. If such a field exists, designate it as the primary key field; otherwise, designate a compound primary key.

In the talent agency example, the attributes, or fields, of the Band entity are Band Name, Band Phone Number, and Desired Fee, as shown in Figure A-1. Assume that no two bands have the same name, so the primary key field can be Band Name. The data type of each field is shown.

BAND	
Field Name	**Data Type**
Band Name (primary key)	Text
Band Phone Number	Text
Desired Fee	Currency

FIGURE A-1 The Band table and its fields

Two Band records are shown in Figure A-2.

Band Name (primary key)	Band Phone Number	Desired Fee
Heartbreakers	981 831 1765	$800
Lightmetal	981 831 2000	$700

FIGURE A-2 Records in the Band table

If two bands might have the same name, Band Name would not be a good primary key, so a different unique identifier would be needed. Such situations are common. Most businesses have many types of inventory, and duplicate names are possible. The typical solution is to assign a number to each product to use as the primary key field. A college could have more than one faculty member with the same name, so each faculty member would be assigned an EIN. Similarly, banks assign a personal identification number (PIN) for each depositor. Each automobile produced by a car manufacturer gets a unique vehicle identification number (VIN). Most businesses assign a number to each sale, called an invoice number. (The next time you go to a grocery store, note the number on your receipt. It will be different from the number on the next customer's receipt.)

At this point, you might be wondering why Band Member would not be an attribute of Band. The answer is that, although you must record each band member, you do not know in advance how many members are in each band. Therefore, you do not know how many fields to allocate to the Band table for members. (Another way to think about band members is that they are the agency's employees, in effect. Databases for organizations usually have an Employee entity.) You should create a Band Member table with the attributes Member ID Number, Member Name, Band Name, Instrument, and Phone. A Member ID Number field is needed because member names may not be unique. The table and its fields are shown in Figure A-3.

BAND MEMBER	
Field Name	**Data Type**
Member ID Number (primary key)	Text
Member Name	Text
Band Name (foreign key)	Text
Instrument	Text
Phone	Text

FIGURE A-3 The Band Member table and its fields

Note in Figure A-3 that the phone number is classified as a Text data type because the field values will not be used in an arithmetic computation. The benefit is that Text data type values take up fewer bytes than Numerical or Currency data type values; therefore, the file uses less storage space. You should also use the Text data type for number values such as zip codes.

Five records in the Band Member table are shown in Figure A-4.

Member ID Number (primary key)	Member Name	Band Name	Instrument	Phone
0001	Jamal Marshall	Heartbreakers	Guitar	981 444 1111
0002	Chantal Marshall	Heartbreakers	Vocals	981 444 1234
0003	Tirsa Abboud	Heartbreakers	Keyboard	981 555 1199
0004	Diego Flores	Lightmetal	Sax	981 888 1654
0005	Sue Hoopes	Lightmetal	Piano	981 888 1765

FIGURE A-4 Records in the Band Member table

You can include Instrument as a field in the Band Member table because the agent records only one instrument for each band member. Thus, you can use the instrument as a way to describe a band member, much like the phone number is part of the description. Phone could not be the primary key because two members might share a telephone and because members might change their numbers, making database administration more difficult.

You might ask why Band Name is included in the Band Member table. The common-sense reason is that you did not include the Member Name in the Band table. You must relate bands and members somewhere, and the Band Member table is the place to do it.

To think about this relationship in another way, consider the cardinality of the relationship between Band and Band Member. It is a one-to-many relationship: one band has many members, but each member in the database plays in just one band. You establish such a relationship in the database by using the primary key field of one table as a foreign key in the other table. In Band Member, the foreign key Band Name is used to establish the relationship between the member and his or her band.

The attributes of the Club entity are Club Name, Address, Contact Name, Club Phone Number, Preferred Fee, and Feed Band?. The Club table can define the Club entity, as shown in Figure A-5.

CLUB	
Field Name	**Data Type**
Club Name (primary key)	Text
Address	Text
Contact Name	Text
Club Phone Number	Text
Preferred Fee	Currency
Feed Band?	Yes/No

FIGURE A-5 The Club table and its fields

Two records in the Club table are shown in Figure A-6.

Club Name (primary key)	Address	Contact Name	Club Phone Number	Preferred Fee	Feed Band?
East End	1 Duce St.	Neev Kalb	981 444 8877	$600	Yes
West End	99 Duce St.	Vera Kaan	981 555 0011	$650	No

FIGURE A-6 Records in the Club table

You might wonder why Bands Booked into Club (or a similar name) is not an attribute of the Club table. There are two reasons. First, you do not know in advance how many bookings a club will have, so the value cannot be an attribute. Second, Bookings is the agency's revenue-generating transaction, an event entity, and you need a table for that business transaction. Consider the booking transaction next.

You know that the talent agent books a certain band into a certain club for a specific fee on a certain date, starting and ending at a specific time. From that information, you can see that the attributes of the Bookings entity are Band Name, Club Name, Booking Date, Start Time, End Time, and Fee. The Bookings table and its fields are shown in Figure A-7.

BOOKINGS	
Field Name	**Data Type**
Band Name (foreign key)	Text
Club Name (foreign key)	Text
Booking Date	Date/Time
Start Time	Date/Time
End Time	Date/Time
Fee	Currency

FIGURE A-7 The Bookings table and its fields—and no designation of a primary key

Some records in the Bookings table are shown in Figure A-8.

Band Name	Club Name	Booking Date	Start Time	End Time	Fee
Heartbreakers	East End	11/21/20	21:30	23:30	$800
Heartbreakers	East End	11/22/20	21:00	23:30	$750
Heartbreakers	West End	11/28/20	19:00	21:00	$500
Lightmetal	East End	11/21/20	18:00	20:00	$700
Lightmetal	West End	11/22/20	19:00	21:00	$750

FIGURE A-8 Records in the Bookings table

Note that no single field is guaranteed to have unique values, because each band is likely to be booked many times and each club might be used many times. Furthermore, each date and time can appear more than once. Thus, no one field can be the primary key.

If a table does not have a single primary key field, you can make a compound primary key whose field values will be unique when taken together. Because a band can be in only one place at a time, one possible solution is to create a compound key from the Band Name, Booking Date, and Start Time fields. An alternative solution is to create a compound primary key from the Club Name, Booking Date, and Start Time fields.