

Relational Databases

LEARNING OBJECTIVES

After studying this chapter, you should be able to:

1. Explain the importance and advantages of databases, as well as the difference between database systems and file-based legacy systems.
2. Explain database systems, including logical and physical views, schemas, the data dictionary, and DBMS languages.
3. Describe what a relational database is, how it organizes data, and how to create a set of well-structured relational database tables.

INTEGRATIVE CASE

S&S

S&S is very successful and operates five stores and a popular website. Ashton Fleming believes that it is time to upgrade S&S's accounting information system (AIS) so that Susan and Scott can easily access the information they need to run their business. Most new AISs are based on a relational database. Since Ashton knows that Scott and Susan are likely to have questions, he prepared a brief report that explains why S&S's new AIS should be a relational database system. His report addresses the following questions:

1. What is a database system, and how does it differ from file-oriented systems?
2. What is a *relational* database system?
3. How do you design a well-structured set of tables in a relational database?
4. How do you query a relational database system?

Introduction

Relational databases underlie most modern integrated AISs. This chapter and Chapters 17 through 19 explain how to participate in the design and implementation of a database. This chapter defines a database, with the emphasis on understanding the relational database structure. Chapter 17 introduces two tools used to design databases—entity-relationship diagramming and REA data modeling—and demonstrates how to use them to build a data model.



Chapter 18 explains how to implement an REA data model and how to produce the information needed to manage an organization. Chapter 19 discusses advanced data modeling and database design issues.

Databases and Files

To appreciate the power of databases, it is important to understand how data are stored in computer systems. Figure 4-1 shows a data hierarchy. Information about the attributes of a customer, such as name and address, are stored in fields. All the fields containing data about one entity (e.g., one customer) form a record. A set of related records, such as all customer records, forms a file (e.g., the customer file). A set of interrelated, centrally coordinated data files that are stored with as little data redundancy as possible forms a **database**. A database consolidates records previously stored in separate files into a common pool and serves a variety of users and data processing applications.

Databases were developed to address the proliferation of master files. For many years, companies created new files and programs each time a need for information arose. Bank of America once had 36 million customer accounts in 23 separate systems. This proliferation created problems such as storing the same data in two or more master files, as shown in Figure 4-2. This made it difficult to integrate and update data and to obtain an organization-wide view of data. It also created problems because the data in the different files were inconsistent. For example, a customer's address may have been correctly updated in the shipping master file but not the billing master file.

database - A set of interrelated, centrally coordinated data files that are stored with as little data redundancy as possible.

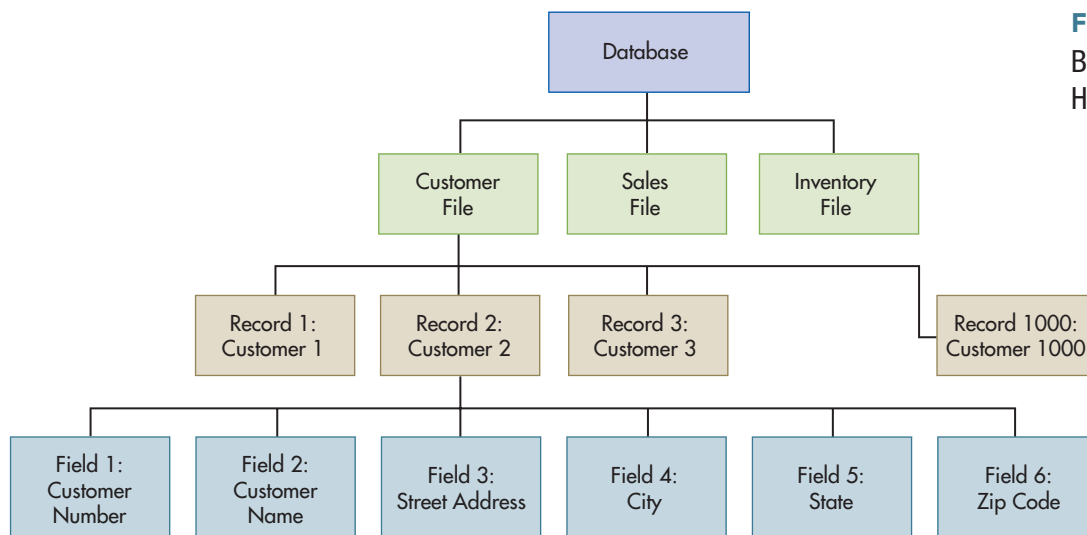


FIGURE 4-1
Basic Elements of Data Hierarchy

database management system (DBMS) - The program that manages and controls the data and the interfaces between the data and the application programs that use the data stored in the database.

database system - The database, the DBMS, and the application programs that access the database through the DBMS.

database administrator (DBA) - The person responsible for coordinating, controlling, and managing the database.

data warehouse - Very large databases containing detailed and summarized data for a number of years that are used for analysis rather than transaction processing.

business intelligence - Analyzing large amounts of data for strategic decision making.

online analytical processing (OLAP) - Using queries to investigate hypothesized relationships among data.

data mining - Using sophisticated statistical analysis to “discover” unhypothesized relationships in the data.

Figure 4-2 illustrates the differences between file-oriented systems and database systems. In the database approach, data is an organizational resource that is used by and managed for the entire organization, not just the originating department. A **database management system (DBMS)** is the program that manages and controls the data and the interfaces between the data and the application programs that use the data stored in the database. The database, the DBMS, and the application programs that access the database through the DBMS are referred to as the **database system**. The **database administrator (DBA)** is responsible for coordinating, controlling, and managing the database.

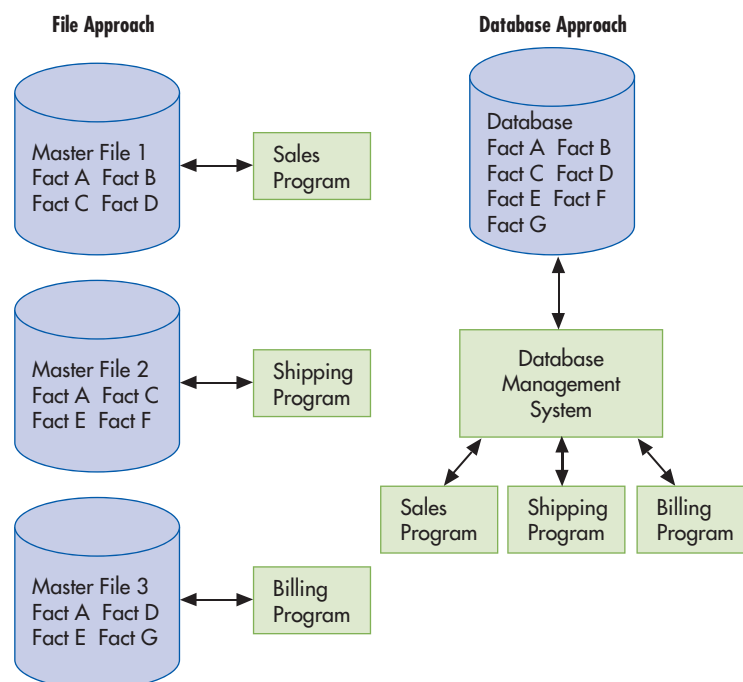
USING DATA WAREHOUSES FOR BUSINESS INTELLIGENCE

In today’s fast-paced global economy, management must constantly reevaluate financial and operating performance in light of strategic goals and quickly alter plans as needed. Since strategic decision making requires access to large amounts of historical data, organizations are building separate databases called data warehouses. A **data warehouse** is one or more very large databases containing both detailed and summarized data for a number of years that is used for analysis rather than transaction processing. It is not unusual for data warehouses to contain hundreds or thousands of terabytes of data. Some data warehouses are measured in petabytes (1,000 terabytes or 1 million gigabytes).

Data warehouses do not replace transaction processing databases; they complement them by providing support for strategic decision making. Since data warehouses are not used for transaction processing, they are usually updated periodically rather than in real time. Whereas transaction processing databases minimize redundancy and maximize the efficiency of updating them to reflect the results of current transactions, data warehouses are purposely redundant to maximize query efficiency.

Analyzing large amounts of data for strategic decision making is often referred to as **business intelligence**. There are two main techniques used in business intelligence: online analytical processing (OLAP) and data mining. **Online analytical processing (OLAP)** is using queries to investigate hypothesized relationships among data. For example, a manager may analyze supplier purchases for the last 3 years, followed by additional queries that “drill down” to lower levels by grouping purchases by item number and by fiscal period. **Data mining** is using sophisticated statistical analysis, including artificial intelligence techniques such as neural networks, to “discover” unhypothesized relationships in the data. For example,

FIGURE 4-2
File-Oriented Systems
versus Database
Systems



credit card companies use data mining to identify usage patterns indicative of fraud. Similarly, data mining techniques can identify previously unknown relationships in sales data that can be used in future promotions.

Proper controls are needed to reap significant benefits from data warehousing. Data validation controls are needed to ensure that data warehouse input is accurate. Verifying the accuracy, called scrubbing the data, is often one of the most time-consuming and expensive steps in creating a data warehouse. It is also important to control access to the data warehouse as well as to encrypt the stored data. Finally, it is important to regularly backup the data warehouse and store the backups securely.

Bank of America created a customer information database to provide customer service, marketing analysis, and managerial information. It was the largest in the banking industry, with over 600 billion characters of data. It contained all bank data on checking and savings accounts; real estate, consumer, and commercial loans; ATMs; and bankcards. Although the bank spends \$14 million a year to maintain the data warehouse, it is worth the cost. Queries that formerly averaged two hours took only five minutes. Minutes after Los Angeles suffered an earthquake, the bank sorted its \$28 billion mortgage loan portfolio by zip code, identified loans in the earthquake area, and calculated its potential loan loss.

THE ADVANTAGES OF DATABASE SYSTEMS

Virtually all mainframes and servers use database technology, and database use in personal computers is growing rapidly. Most accountants are involved with databases through data entry, data processing, querying, or auditing. They also develop, manage, or evaluate the controls needed to ensure database integrity. Databases provide organizations with the following benefits:

- **Data integration.** Master files are combined into large “pools” of data that many application programs access. An example is an employee database that consolidates payroll, personnel, and job skills master files.
- **Data sharing.** Integrated data are more easily shared with authorized users. Databases are easily browsed to research a problem or obtain detailed information underlying a report. The FBI, which does a good job of collecting data but a poor job of sharing it, is spending eight years and \$400 million to integrate data from their different systems.
- **Minimal data redundancy and data inconsistencies.** Because data items are usually stored only once, data redundancy and data inconsistencies are minimized.
- **Data independence.** Because data and the programs that use them are independent of each other, each can be changed without changing the other. This facilitates programming and simplifies data management.
- **Cross-functional analysis.** In a database system, relationships, such as the association between selling costs and promotional campaigns, can be explicitly defined and used in the preparation of management reports.

THE IMPORTANCE OF GOOD DATA

Incorrect database data can lead to bad decisions, embarrassment, and angry users. For example:

- A company sent half its mail-order catalogs to incorrect addresses. A manager finally investigated the large volume of returns and customer complaints. Correcting customer addresses in the database saved the company \$12 million a year.
- Valparaiso, Indiana, used the county database to develop its tax rates. After the tax notices were mailed, a huge error was discovered: A \$121,900 home was valued at \$400 million and caused a \$3.1 million property tax revenue shortfall. As a result, the city, the school district, and governmental agencies had to make severe budget cuts.

The Data Warehousing Institute estimates that bad data cost businesses over \$600 billion a year in unnecessary postage, marketing costs, and lost customer credibility. It is estimated that over 25% of business data is inaccurate or incomplete. In a recent survey, 53% of 750 information technology (IT) professionals said their companies experienced problems due to poor-quality data.

Managing data gets harder every year: The quantity of data generated and stored doubles every 18 months. To avoid outdated, incomplete, or erroneous data, management needs policies and procedures that ensure clean, or scrubbed, data. The Sarbanes-Oxley Act (SOX) states that top executives face prosecution and jail time if a company’s financial data are not in order. Preventing and detecting bad data are discussed in more detail in Chapters 5 through 11.

Database Systems

LOGICAL AND PHYSICAL VIEWS OF DATA

In file-oriented systems, programmers must know the physical location and layout of records. Suppose a programmer wants a report showing customer number, credit limit, and current balance. To write the program, she must understand the location and length of the fields needed (i.e., record positions 1 through 10 for customer number) and the format of each field (alphanumeric or numeric). The process becomes more complex if data from several files are used.

A **record layout** is a document that shows the items stored in a file, including the order and length of the data fields and the type of data stored in an accounts receivable file. Figure 4-3 shows a record layout of an accounts receivable file.

Database systems overcome this problem by separating the storage of the data from the use of data elements. The database approach provides two separate views of the data: the physical view and the logical view. The **logical view** is how people conceptually organize and understand the relationships among data items. For example, a sales manager views all customer information as being stored in a table. The **physical view** refers to the way data are physically arranged and stored in the computer system.

As shown in Figure 4-4, database management (DBMS) software links the way data are physically stored with each user’s logical view of the data. The DBMS allows users to access, query, or update the database without reference to how or where data are physically stored. Separating the logical and physical views of data also means that users can change their logical view of data without changing the way data are physically stored. Likewise, the DBA can change physical storage to improve system performance without affecting users or application programs.

SCHEMAS

A **schema** is a description of the data elements in a database, the relationships among them, and the logical model used to organize and describe the data. There are three levels of schemas: the conceptual, the external, and the internal. Figure 4-5 shows the relationships among these three levels. The **conceptual-level schema**, the organization-wide view of the *entire* database, lists all data elements and the relationships among them. The **external-level schema** is an individual user’s view of portions of a database, each of which is referred to as a **subschema**. The **internal-level schema**, a low-level view of the database, describes how the data are stored and accessed, including record layouts, definitions, addresses, and indexes. Figure 4-5 connects each of the levels with bidirectional arrows to represent schema mappings. The DBMS uses the mappings to translate a user’s or a program’s request for data (expressed in terms of logical names and relationships) into the indexes and addresses needed to physically access the data.

record layout - Document that shows the items stored in a file, including the order and length of the data fields and the type of data stored.

logical view - How people conceptually organize, view, and understand the relationships among data items.

physical view - The way data are physically arranged and stored in the computer system.

schema - A description of the data elements in a database, the relationships among them, and the logical model used to organize and describe the data.

conceptual-level schema - The organization-wide view of the entire database that lists all data elements and the relationships between them.

external-level schema - An individual user’s view of portions of a database; also called a subschema.

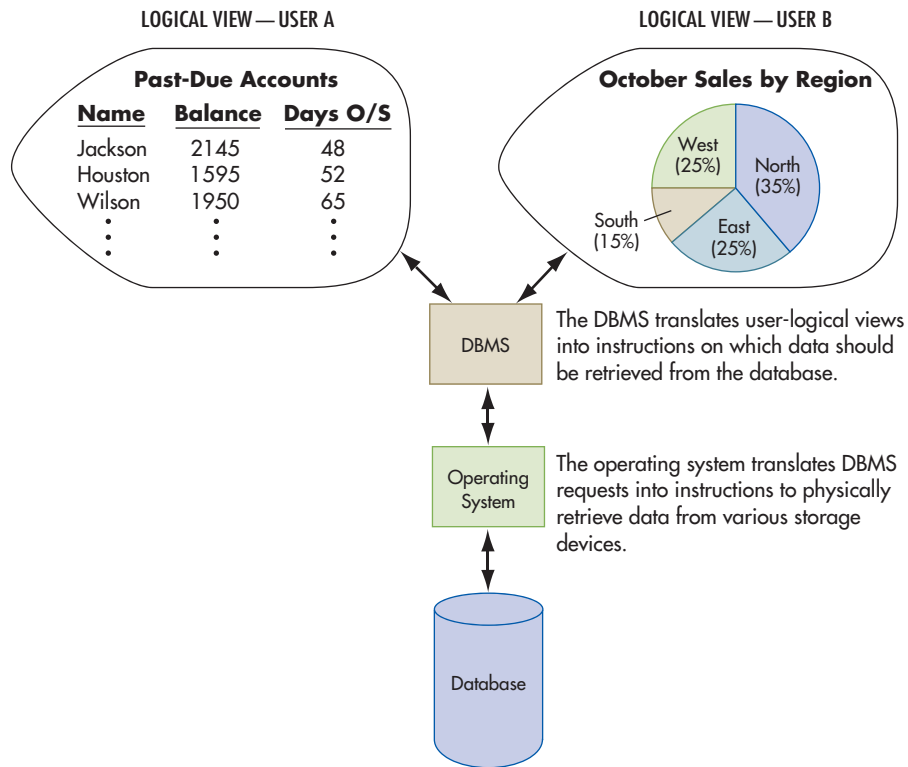
subschema - A subset of the schema; the way the user defines the data and the data relationships.

internal-level schema - A low-level view of the entire database describing how the data are actually stored and accessed.

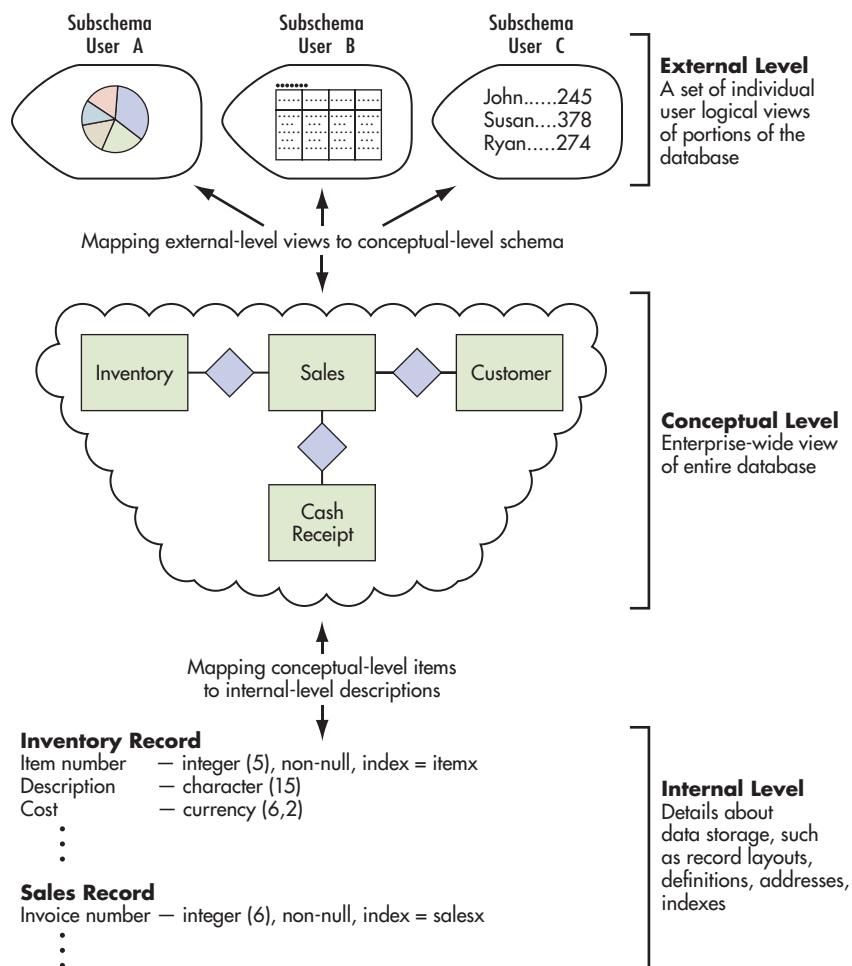
FIGURE 4-3
Accounts Receivable File
Record Layout

Customer Number N		Customer Name A		Address A		Credit Limit N		Balance N	
1	10	11	30	31	60	61	68	69	76

A = alphanumeric field
N = numeric field

**FIGURE 4-4**

Function of the DBMS:
To Support Multiple
Logical Views of Data

**FIGURE 4-5**

Three Levels of Schemas

At S&S, the conceptual schema for the revenue cycle database contains data about customers, sales, cash receipts, sales personnel, cash, and inventory. External subschemas are derived from this schema, each tailored to the needs of different users or programs. Each subschema can prevent access to those portions of the database that do not apply to it. For example, the sales order entry subschema includes data about customer credit limits, current balances, and inventory quantities and prices. It would not include the cost of inventory or bank account balances.

THE DATA DICTIONARY

data dictionary - Information about the structure of the database, including a description of each data element.

A **data dictionary** contains information about the structure of the database. As shown in Table 4-1, for each data element stored in the database, there is a record in the dictionary describing it. The DBMS maintains the data dictionary, whose inputs include new or deleted data elements and changes in data element names, descriptions, or uses. Outputs include reports for programmers, designers, and users, such as (1) programs or reports using a data item, (2) synonyms for the data elements in a file, and (3) data elements used by a user. These reports are used for system documentation, for database design and implementation, and as part of the audit trail.

DBMS LANGUAGES

data definition language (DDL) - DBMS language that builds the data dictionary, creates the database, describes logical views, and specifies record or field security constraints.

data manipulation language (DML) - DBMS language that changes database content, including data element creations, updates, insertions, and deletions.

data query language (DQL) - High-level, English-like, DBMS language that contains powerful, easy-to-use commands that enable users to retrieve, sort, order, and display data.

report writer - DBMS language that simplifies report creation.

data model - An abstract representation of database contents.

relational data model - A two-dimensional table representation of data; each row represents a unique entity (record) and each column is a field where record attributes are stored.

tuple - A row in a table that contains data about a specific item in a database table.

primary key - Database attribute, or combination of attributes, that uniquely identifies each row in a table.

foreign key - An attribute in a table that is also a primary key in another table; used to link the two tables.

A DBMS has several languages. The **data definition language (DDL)** builds the data dictionary, creates the database, describes logical views for each user, and specifies record or field security constraints. The **data manipulation language (DML)** changes database content, including data element creations, updates, insertions, and deletions. The **data query language (DQL)** is a high-level, English-like language that contains powerful, easy-to-use commands that enable users to retrieve, sort, order, and display data. A **report writer** simplifies report creation. Users specify the data elements they want printed, and the report writer searches the database, extracts the data elements, and prints them in the user-specified format. The DQL and report writer are available to users. The DDL and DML should be restricted to authorized administrators and programmers.

Relational Databases

A DBMS is characterized by the logical **data model**, or abstract representation of database contents. As most new DBMSs are relational databases, this chapter focuses primarily on them. The **relational data model** represents conceptual- and external-level schemas as if data are stored in two-dimensional tables like the one shown in Table 4-2. The data are actually stored not in tables, but in the manner described in the internal-level schema.

Each row in a table, called a **tuple** (rhymes with *couple*), contains data about a specific item in a database table. Each column contains data about an attribute of that entity. For example, each row in Table 4-2 contains data about a particular inventory item that S&S carries, and each column contains data about specific inventory attributes, such as description, color, and price. Similarly, each row in a Customer table contains data about a specific customer, and each column contains data about customer attributes, such as name and address.

TYPES OF ATTRIBUTES

A **primary key** is the database attribute, or combination of attributes, that uniquely identifies a specific row in a table. The primary key in Table 4-2 is Item Number, as it uniquely identifies each merchandise item that S&S sells. Usually, the primary key is a single attribute. In some tables, two or more attributes are needed to identify uniquely a specific row in a table. The primary key of the Sales-Inventory table in Table 4-5 is the combination of Sales Invoice # and Item #.

A **foreign key** is an attribute in a table that is also a primary key in another table and is used to link the two tables. Customer # in Table 4-5 is the primary key in the Customer table and a foreign key in the Sales table. In the Sales table, Customer # links a sale to data about the customer who made the purchase, as contained in the Customer table (see arrows connecting tables).

TABLE 4-1 Example of a Data Dictionary

DATA ELEMENT NAME	DESCRIPTION	RECORDS IN WHICH CONTAINED	SOURCE	FIELD LENGTH	FIELD TYPE	PROGRAMS IN WHICH USED	OUTPUTS IN WHICH CONTAINED	AUTHORIZED USERS	OTHER DATA NAMES
Customer number	Unique identifier of each customer	A/R record, customer record, sales analysis record	Customer number listing	10	Numeric	A/R update, customer file update, sales analysis update, credit analysis	A/R aging report, customer status report, sales analysis report, credit report	No restrictions	None
Customer name	Complete name of customer	Customer record	Initial customer order	20	Alphanumeric	Customer file update, statement processing	Customer status report, monthly statement	No restrictions	None
Address	Street, city, state, and zip code	Customer record	Credit application	30	Alphanumeric	Customer file update, statement processing	Customer status report, monthly statement	No restrictions	None
Credit limit	Maximum credit that can be extended to customer	Customer record, A/R record	Credit application	8	Numeric	Customer file update, A/R update, credit analysis	Customer status report, A/R aging report, credit report	D. Dean G. Allen H. Heaton	CR_limit
Balance	Balance due from customer on credit purchases	A/R record, sales analysis record	Various sales and payment transactions	8	Numeric	A/R update, sales analysis update, statement processing, credit analysis	A/R aging report, sales analysis report, monthly statement, credit report	G. Burton B. Heninger S. Summers	Cust_bal

TABLE 4-2 Sample Inventory Table for S&S

Inventory					
Item Number	Description	Color	Vendor Number	Quantity on Hand	Price
1036	Refrigerator	White	10023	12	1199
1038	Refrigerator	Almond	10023	7	1299
1039	Refrigerator	Hunter Green	10023	5	1499
2061	Range	White	10011	6	799
2063	Range	Black	10011	5	999
3541	Washer	White	10008	15	499
3544	Washer	Black	10008	10	699
3785	Dryer	White	10019	12	399
3787	Dryer	Almond	10019	8	499
*	0		0	0	0
Record: 10 of 10 No Filter Search					

Other nonkey attributes in a table store important information about that entity. The inventory table in Table 4-2 contains information about the description, color, vendor number, quantity on hand, and price of each item S&S carries.

DESIGNING A RELATIONAL DATABASE FOR S&S, INC.

In a manual accounting system, S&S would capture sales information on a preprinted sales invoice that provides both a logical and physical view of the data collected. Physical storage of sales invoice data is simple; a copy of the invoice is stored in a file cabinet.

Storing the same data in a computer is more complex. Suppose S&S wanted to store five sales invoices (numbered 101 to 105) electronically. On several invoices, a customer buys more than one item. Let us look at the effects of several ways of storing this information.

1: Store All Data in One Uniform Table. S&S could store sales data in one table, as illustrated in Table 4-3. This approach has two disadvantages. First, it stores lots of redundant data. Examine invoice 102 in Table 4-3. Because three inventory items are sold, invoice and customer data (columns 1 to 9) are recorded three times. Likewise, inventory descriptions and unit prices are repeated each time an item is sold. Because sales volumes are high in a retail store (remember, Table 4-3 represents only five invoices), such redundancy makes file maintenance unnecessarily time-consuming and error-prone.

Second, problems occur when invoice data are stored in these types of tables. The first is called an **update anomaly**, because data values are not correctly updated. Changing a customer's address involves searching the entire table and changing every occurrence of that customer's address. Overlooking even one row creates an inconsistency, because multiple addresses would exist for the same customer. This could result in unnecessary duplicate mailings and other errors.

An **insert anomaly** occurs in our example because there is no way to store information about prospective customers until they make a purchase. If prospective customer data is entered before a purchase is made, the Sales Invoice # column would be blank. However, the Sales Invoice # is the primary key for Table 4-3 and cannot be blank, as it uniquely identifies the record.

A **delete anomaly** occurs when deleting a row has unintended consequences. For example, if customer addresses are stored in the sales table, then deleting the row where the only sale to a customer is stored results in the loss of all information for that customer.

update anomaly - Improper database organization where a non-primary key item is stored multiple times; updating the item in one location and not the others causes data inconsistencies.

insert anomaly - Improper database organization that results in the inability to add records to a database.

delete anomaly - Improper organization of a database that results in the loss of all information about an entity when a row is deleted.

TABLE 4-3 Example of Storing All Sales Data for S&S in One Table

Sales													
Sales Invoice #	Date	Salesperson	Customer #	Invoice Total	Customer Name	Street	City	State	Item #	Quantity	Description	Unit Price	Extended Amount
101	10/15/2018	J. Buck	151	1447	D. Ainge	123 Lotus Lane	Phoenix	AZ	10	2	Television	499	998
101	10/15/2018	J. Buck	151	1447	D. Ainge	123 Lotus Lane	Phoenix	AZ	50	1	Microwave	449	449
102	10/15/2018	S. Knight	152	4394	G. Kite	40 Quatro Road	Mesa	AZ	10	1	Television	499	499
102	10/15/2018	S. Knight	152	4394	G. Kite	40 Quatro Road	Mesa	AZ	20	3	Freezer	699	2097
102	10/15/2018	S. Knight	152	4394	G. Kite	40 Quatro Road	Mesa	AZ	30	2	Refrigerator	899	1798
103	10/15/2018	S. Knight	151	898	D. Ainge	123 Lotus Lane	Phoenix	AZ	50	2	Microwave	449	898
104	10/15/2018	J. Buck	152	789	G. Kite	40 Quatro Road	Mesa	AZ	40	1	Range	789	789
105	11/14/2018	J. Buck	153	3994	F. Roberts	401 Excel Way	Chandler	AZ	10	3	Television	499	1497
105	11/14/2018	J. Buck	153	3994	F. Roberts	401 Excel Way	Chandler	AZ	20	1	Freezer	699	699
105	11/14/2018	J. Buck	153	3994	F. Roberts	401 Excel Way	Chandler	AZ	30	2	Refrigerator	899	1796
0				0					0	0		0	0
Record: 11 of 11 Search No Filter													

TABLE 4-4 Example of Storing S&S Sales Data by Adding Columns for Each Additional Item Sold

Sales Invoice #	Columns 2-9	Item #	Quantity	Description	Unit Price	Extended Amount	Item #2	Quantity2
101	Same 8	10	2	Television	499	998	50	1
102	columns	10	1	Television	499	499	20	3
103	as in	50	2	Microwave	449	898		
104	Table 4-3	40	1	Range	789	789		
105	above	10	3	Television	499	1497	20	1
*	0	0	0		0	0	0	0

Record: 14 6 of 6 No Filter Search

2: Vary the Number of Columns. An alternative to Table 4-3 is to record sales invoice and customer data once and add additional columns to record each item sold. Table 4-4 illustrates this approach. Although this reduces data redundancy and eliminates some anomalies associated with Table 4-3, it has drawbacks. S&S would have to decide in advance how many item numbers to leave room for in each row (i.e., how many columns to put in the table; note in Table 4-4 that to store each additional item requires five additional columns—Item, Quantity, Description, Unit Price, and Extended Amount). If room is left for four items (20 columns), how would data about a sale involving eight items (40 columns) be stored? If room is left for eight items, however, there will be a great deal of wasted space, as is the case for sales invoices 103 and 104.

3: The Solution: A Set of Tables. The storage problems in Tables 4-3 and 4-4 are solved using a **relational database**. The set of tables in Table 4-5 represent a well-structured relational database.

relational database - A database built using the relational data model.

BASIC REQUIREMENTS OF A RELATIONAL DATABASE

We now turn to the guidelines used to develop a properly structured relational database.

- 1. Every column in a row must be single valued.** In a relational database, there can only be one value per cell. At S&S, each sale can involve more than one item. On invoice 102, the customer bought a television, a freezer, and a refrigerator. If Item # were an attribute in the Sales table, it would have to take on three values (item numbers 10, 20, and 30). To solve this problem, a Sales-Inventory table was created that lists each item sold on an invoice. The third line in the Sales-Inventory table in Table 4-5 shows invoice 102 and item number 10 (television). The fourth line shows invoice 102 and item 20 (freezer). The fifth line shows invoice 102 and item 30 (refrigerator). This table repeats the invoice number as often as needed to show all the items purchased on a sales invoice.
- 2. Primary keys cannot be null.** A primary key cannot uniquely identify a row in a table if it is null (blank). A nonnull primary key ensures that every row in a table represents something and that it can be identified. This is referred to as the **entity integrity rule**. In the Sales-Inventory table in Table 4-5, no single field uniquely identifies each row. However, the first two columns, taken together, do uniquely identify each row and constitute the primary key.
- 3. Foreign keys, if not null, must have values that correspond to the value of a primary key in another table.** Foreign keys link rows in one table to rows in another table. In Table 4-5, Customer # can link each sales transaction with the customer who participated in that event only if the Sales table Customer # value corresponds to an actual customer number in the Customer table. This constraint, called the **referential integrity rule**, ensures database consistency. Foreign keys can contain null values. For example, when customers pay cash, Customer # in the sales table can be blank.
- 4. All nonkey attributes in a table must describe a characteristic of the object identified by the primary key.** Most tables contain other attributes in addition to the primary and foreign keys. In the Customer table in Table 4-5, Customer # is the primary key, and customer name, street, city, and state are important facts that describe the customer.

entity integrity rule - A non-null primary key ensures that every row in a table represents something and that it can be identified.

referential integrity rule - Foreign keys which link rows in one table to rows in another table must have values that correspond to the value of a primary key in another table.

TABLE 4-4 Continued

Description2 ▾	Unit Price2 ▾	Extended Amount2 ▾	Item #3 ▾	Quantity3 ▾	Description3 ▾	Unit Price3 ▾	Extended Amount3 ▾
Microwave	449	449					
Freezer	699	2097	30		2 Refrigerator	889	1798
Freezer	699	699	30		2 Refrigerator	899	1798
	0	0	0	0		0	0
No Filter <input type="text" value="Search"/> <input type="button" value="⏮"/> <input type="button" value="⏭"/>							

These four constraints produce a well-structured (normalized) database in which data are consistent and data redundancy is minimized and controlled. In Table 4-5, having a table for each entity of interest avoids the anomaly problems discussed previously and minimizes redundancy. Redundancy is not eliminated, as certain items, such as Sales Invoice #, appear in more than one table when they are foreign keys. The referential integrity rule ensures that there are no update anomaly problems with the foreign keys.

When data about objects of interest are stored in separate database tables, it is easy to add new data by adding another row to the table. For example, adding a new customer is as simple as adding a new row to the Customer table. Thus, the tables depicted in Table 4-5 are free from insert anomalies.

Relational databases also simplify data deletion. Deleting sales invoice 105, the only sale to customer 153, does not erase all data about that customer, because it is stored in the Customer table. This avoids delete anomalies.

Another benefit of the schema shown in Table 4-5 is that space is used efficiently. The Sales-Inventory table contains a row for each item sold on each invoice. There are no blank rows, yet all sales data are recorded. In contrast, the schema in Table 4-4 results in much wasted space.

TWO APPROACHES TO DATABASE DESIGN

One way to design a relational database, called **normalization**, begins by assuming that everything is initially stored in one large table. Rules are then followed to decompose that initial table into a set of tables in what is called *third normal form (3NF)*, because they are free of update, insert, and delete anomalies. The details of the normalization process are found in the Appendix to this chapter.

In an alternative design approach, called **semantic data modeling**, the designer uses knowledge of business processes and information needs to create a diagram that shows what to include in the database. This diagram is used to create a set of relational tables that are already in 3NF.

Semantic data modeling has significant advantages. First, using a system designer's knowledge of business processes facilitates the efficient design of transaction processing databases. Second, the graphical model explicitly represents the organization's business processes and policies and, by facilitating communication with system users, helps ensure that the new system meets users' actual needs. Semantic data modeling is discussed in Chapters 17 through 19. Chapter 17 introduces two semantic data modeling tools, entity-relationship diagramming and REA modeling, used to design transaction processing databases. Chapter 18 discusses how to implement an REA data model in a relational database. Chapter 19 discusses special topics in REA modeling.

CREATING RELATIONAL DATABASE QUERIES

To retrieve stored data, users query databases. This section of the chapter shows you how to query databases using Microsoft Access. If you want to follow along by creating the queries illustrated in this section, download the S&S In-Chapter Database from the text's website. When you open the database and select the Create ribbon, the ribbon in the top half of Table 4-6 appears. There are two ways to query the database: create a query in Design view

normalization - Following relational database creation rules to design a relational database that is free from delete, insert, and update anomalies.

semantic data modeling - Using knowledge of business processes and information needs to create a diagram that shows what to include in a fully normalized database (in 3NF).

TABLE 4-5 Set of Relational Tables for Storing S&S Sales Data

Sales				
Sales Invoice #	Date	Salesperson	Customer #	
101	10/15/2018	J. Buck	151	
102	10/15/2018	S. Knight	152	
103	10/28/2018	S. Knight	151	
104	10/31/2018	J. Buck	152	
105	11/14/2018	J. Buck	153	
0			0	
Record: 6 of 6 No Filter Search				

Sales-Inventory			
Sales Invoice #	Item #	Quantity	
101	10	2	
101	50	1	
102	10	1	
102	20	3	
102	30	2	
103	50	2	
104	40	1	
105	10	3	
105	20	1	
105	30	2	
0	0	0	
Record: 11 of 11 No Filter Search			

Inventory			
Item #	Unit Price	Description	
10	499	Television	
20	699	Freezer	
30	899	Refrigerator	
40	789	Range	
50	449	Microwave	
0	0		
Record: 6 of 6 No Filter Search			

Customer				
Customer #	Customer Name	Street	City	State
151	D. Ainge	124 Lotus Lane	Phoenix	AZ
152	G. Kite	40 Quatro Roac	Mesa	AZ
153	F. Roberts	401 Excel Way	Chandler	AZ
0				
Record: 4 of 4 No Filter Search				

(the “Query Design” button) or use the wizard (the “Query Wizard” button). These options are outlined in blue in the top half of Table 4-6. The Design view is used in all of the query examples shown. Clicking on the “Query Design” button produces the Show Table window shown in Table 4-6. The user selects the tables needed to produce the desired information; if more tables than necessary are selected, the query may not run properly.

We will use the tables in Table 4-5 to walk through the steps needed to create and run five queries. This will not make you an expert in querying an Access database, but it will show you how to produce useful information.

QUERY 1

Query 1 answers two questions: What are the invoice numbers of all sales made to D. Ainge, and who was the salesperson for each sale?

The Sales and Customer tables contain the three items needed to answer this query: Sales Invoice #, Salesperson, and Customer Name. Click the “Query Design” button (see Table 4-6), and select the Sales and Customer tables by double-clicking on their names or by single-clicking on the name and clicking the “Add” button. The selected tables appear as shown in Table 4-7. A line between the two tables connects the Customer # fields (the Customer table primary key and the Sales table foreign key). Click on Close to make the Show Table window disappear.

To populate the bottom half of the screen shown in Table 4-7, double-click on Sales Invoice #, Salesperson, and Customer Name or drag and drop them into the Field row. Access automatically checks the box in the Show line, so the item will be shown when the query is run.

Since we only want sales to D. Ainge, enter that in the criteria line of the Customer Name column. Access will automatically put question marks around the criteria. Run the query by clicking on the red ! (exclamation) mark on the Query Tools Design ribbon. Table 4-8 shows the tables used, the relationship of the primary and foreign keys between tables, and the query

TABLE 4-6 Creating Queries in the Microsoft Access Database

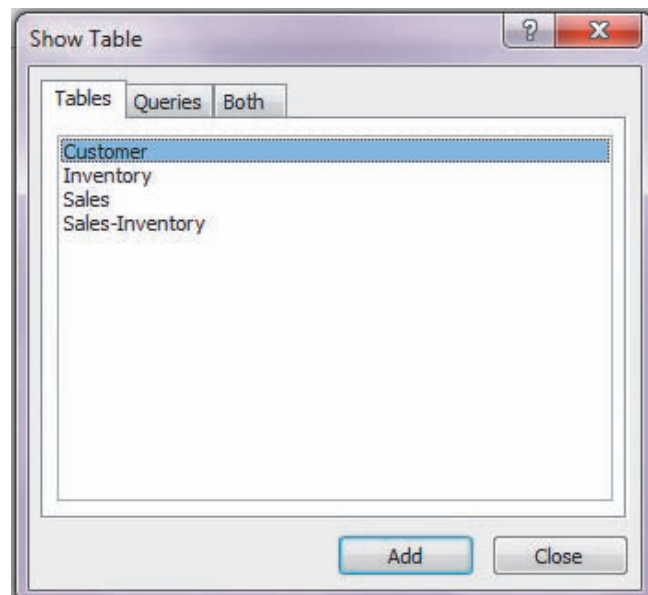
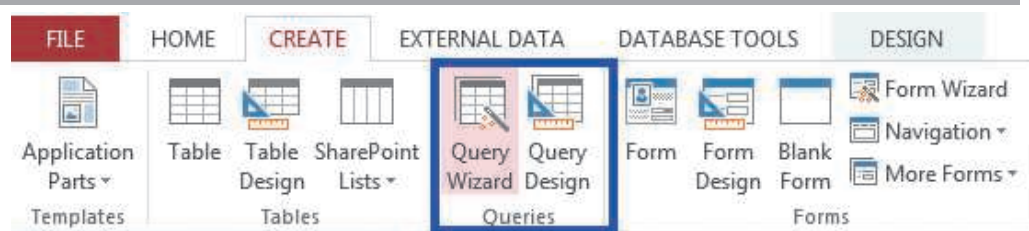


TABLE 4-7 Completed Query 1

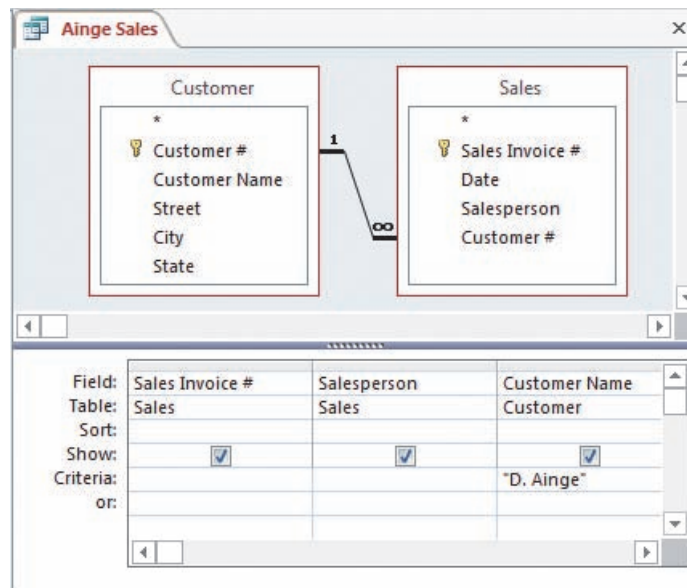


TABLE 4-8 Query 1 Relationships and Query Answer

The screenshot shows the 'Sales' table with the following data:

Sales Invoice #	Date	Salesperson	Customer #
101	10/15/2018	J. Buck	151
102	10/15/2018	S. Knight	152
103	10/28/2018	S. Knight	151
104	10/31/2018	J. Buck	152
105	11/14/2018	J. Buck	153
0			0

Record: 6 of 6

No Filter

Search

The screenshot shows the 'Customer' table with the following data:

Customer #	Customer Name	Street	City	State
151	D. Ainge	124 Lotus Lane	Phoenix	AZ
152	G. Kite	40 Quatro Roac	Mesa	AZ
153	F. Roberts	401 Excel Way	Chandler	AZ
0				

Record: 4 of 4

No Filter

Search

The screenshot shows the 'Ainge Sales' query result window. It displays the following data:

Sales Invoice #	Salesperson	Customer Name
101	J. Buck	D. Ainge
103	S. Knight	D. Ainge
*		

Record: 1 of 2

No Filter

Search

answer. The query answer does not automatically have the title “Ainge Sales.” To assign the query a name, save it by selecting File from the Access menu, then Save Object As, and then enter “Ainge Sales” in the first line of the Save As window, making sure the Object select box is set to “Query,” and then clicking OK. When the query is rerun, the title shown in Table 4-8 will appear.

QUERY 2

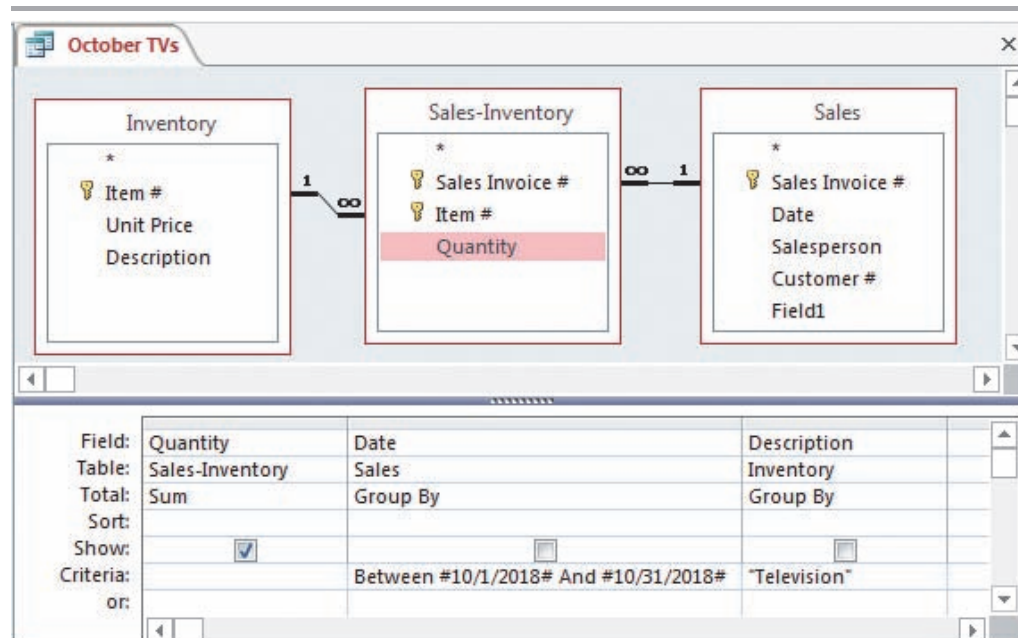
Query 2 answers this question: How many televisions were sold in October?

The Sales, Inventory, and Sales-Inventory tables contain the three items needed to answer this query: Date, Inventory Description, and Quantity. Click on the “Query Design” button in the Create ribbon and select the three tables and the three fields, as shown in Table 4-9. Since we want the quantity of televisions sold in October, we add the criteria “Between #10/1/2018# And #10/31/2018#” to the Date field and “Television” to the Description field.

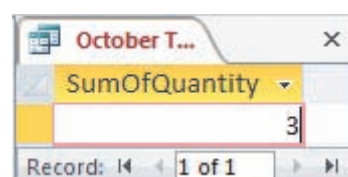
To specify criteria, Access uses operators such as “And,” “Or,” and “Between.” An “And” operator returns the data that meets *all* the criteria linked by “And” operators. The “Between” operator selects all the data in October of 2018; that is, between and including the first and last days of the month. The “Or” operator returns data that meets at least one of the criteria linked by the “Or” operators. The “#” symbol tells Access to look for a date rather than some other type of text.

Since we are only looking for total television sales in October, we don’t need to show the Date or Description. Uncheck the “Show” box in the Date and Description columns. To generate total sales, click the “Totals” button in the Show/Hide portion of the Query Tools Design ribbon. A new row, labeled Total, appears (compare Tables 4-7 and 4-9). Click on the Totals line in the Quantity column, click on the down-arrow symbol, and select Sum from the drop-down menu that appears. The remaining two fields in the Total line will stay as Group By. Running the query in Table 4-9 produces the answer shown.

TABLE 4-9 Completed Query 2 and Answer



Field	Table	Total	Sort	Show	Criteria
Quantity	Sales-Inventory	Sum		<input checked="" type="checkbox"/>	
Date	Sales	Group By		<input type="checkbox"/>	Between #10/1/2018# And #10/31/2018#
Description	Inventory	Group By		<input type="checkbox"/>	"Television"



SumOfQuantity
3

Record: 1 of 1

QUERY 3

Query 3 answers this question: What are the names and addresses of customers buying televisions in October?

This query needs these fields: Date (to select October sales), Description (to select televisions), and Customer Name, Street, City, and State (the information requested). All four tables are used because the Sales-Inventory table is used to move between the Sales and Inventory tables. The query uses the same criteria as Query 2. The Date and Description data do not need to be displayed, so the boxes in the Show line are unchecked. Running the query produces the answer shown in Table 4-10.

QUERY 4

Query 4 answers this question: What are the sales invoice numbers, dates, and invoice totals for October sales, arranged in descending order by sale amount?

Since the database does not contain an Invoice Total column, it is calculated by multiplying the unit price by the quantity for each sale. For example, we would calculate the total sales price of each item sold by multiplying the Quantity field in the Sales-Inventory table by the Unit Price field in the Inventory table. The Sales-Inventory table in Table 4-5 shows that three items were sold on Sales Invoice 102. For item 20, we multiply the quantity (3) by the Unit Price (699), producing 2,097. The same calculation is made for items 10 and 30. Finally, we sum the three item totals to get an invoice total.

Query 4 requires the Sales table (Date, Sales Invoice #), Sales-Inventory table (Quantity), and the Inventory table (Unit Price). However, some fields will not appear in columns on the Select Query window. As shown in Table 4-11, three columns are displayed: Sales Invoice #, Date, and Invoice Total, which we will calculate. The other fields, Quantity and Unit Price, are used in the Invoice Total calculations.

TABLE 4-10 Completed Query 3 and Answer

The screenshot shows the Microsoft Access interface. At the top, the 'Customers TV October' database is open. Below it, the Relationships window displays four tables: Inventory, Sales-Inventory, Sales, and Customer. The relationships are as follows: Inventory (1) to Sales-Inventory (∞), Sales-Inventory (∞) to Sales (1), and Sales (∞) to Customer (1). Below the Relationships window, the Select Query window for 'Customers TV October' is shown. It has a table grid with the following fields and criteria:

Field:	Customer Name	Street	City	State	Date	Description
Table:	Customer	Customer	Customer	Customer	Sales	Inventory
Sort:						
Show:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Criteria:					Between #10/1/2018# And #10/31/2018#	"Television"
or:						

The screenshot shows the query results for 'Customers TV October'. The results are displayed in a table with the following data:

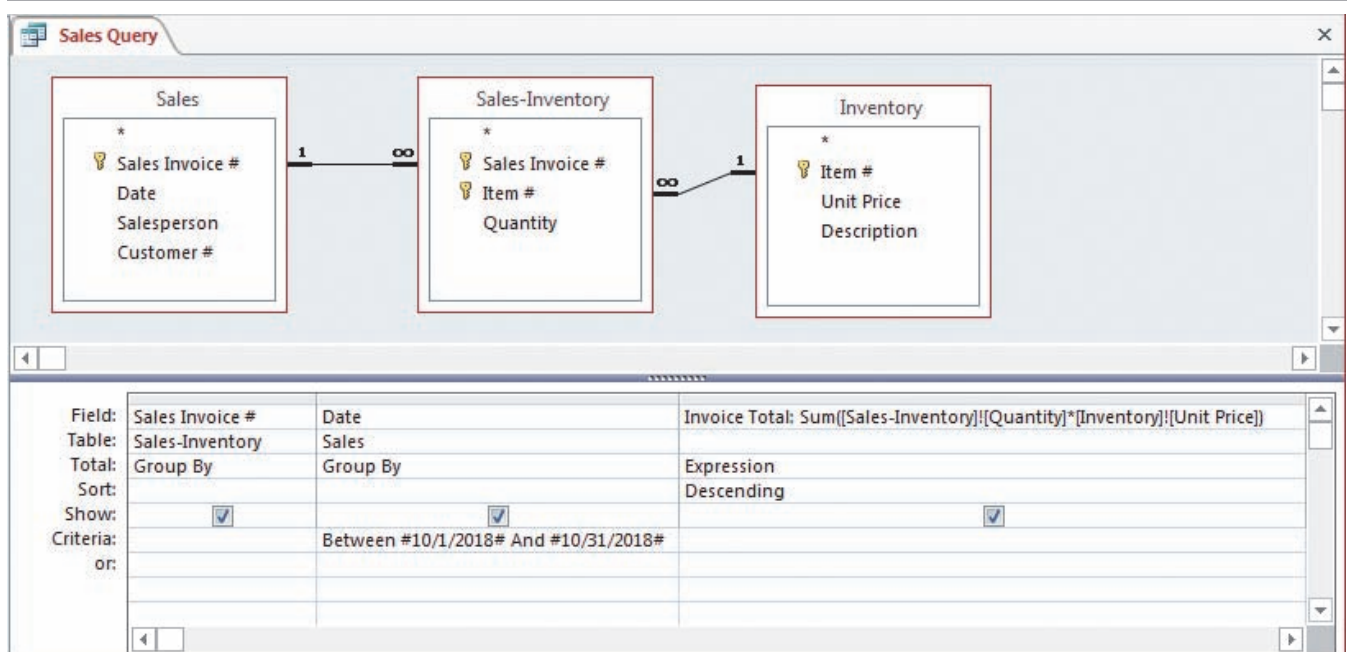
Customer Name	Street	City	State
D. Ainge	124 Lotus Lane	Phoenix	AZ
G. Kite	40 Quatro Road	Mesa	AZ

At the bottom of the screenshot, there is a status bar showing 'Record: 1 of 2' and a search bar.

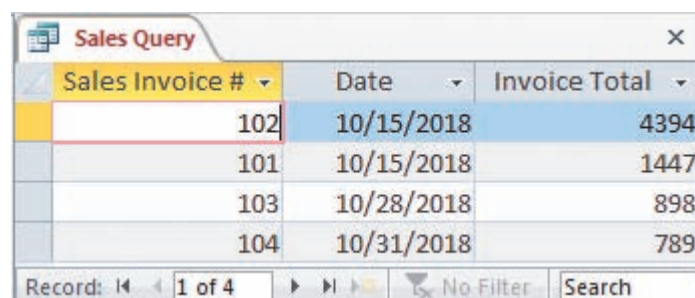
To calculate Invoice Total, type “Invoice Total:” in the first blank Field cell, right-click in the cell, and select Build from the pop-up menu that appears. An Expression Builder window (see Table 4-12) appears, where the formula to calculate the Invoice Total is entered by typing “Sum()”. Between the parentheses, click on the + sign in front of the S&S In-Chapter Database folder in the Expressions Elements box. Then clicking on the + sign in the Tables folder causes the four database tables to appear. Click on the Sales-Inventory table, and the fields in the Sales-Inventory table appear. Double-click on Quantity to put this field in the expression. Note in Table 4-12 that the expression shows the table name and the field name, separated by an exclamation point. To multiply Quantity by Unit Price, type * (the multiplication symbol) and select the Inventory table and the Unit Price field. The formula is now complete, and the screen will appear as shown. To enter the expression into the Select Query window, click on OK.

To complete Query 4, click the “Totals” button in the Query Tools Design ribbon. Click on the down arrow in the Total row of the Invoice Totals column, and select Expression from the pop-up menu. This tells Access to calculate the indicated expression for all items with the same sales invoice number and date. In the same column, click on the down arrow in the Sort row, and select Descending so that the answer is shown in descending Invoice Total order. In the criteria section of the Date column, use the “Between” operator to specify the month of October. Running Query 4 produces the answer shown in Table 4-11.

TABLE 4-11 Completed Query 4 and Answer

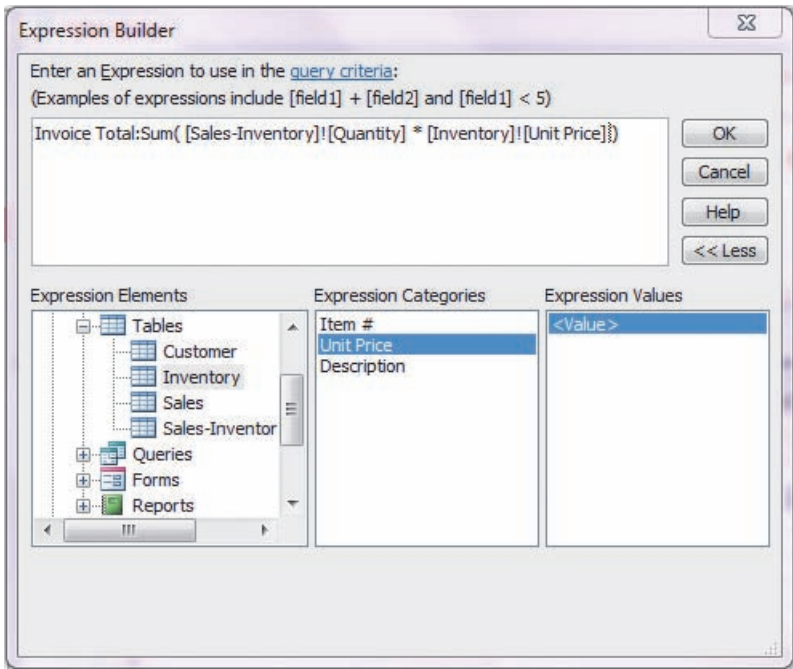


Field:	Sales Invoice #	Date	Invoice Total: Sum([Sales-Inventory]![Quantity]*[Inventory]![Unit Price])
Table:	Sales-Inventory	Sales	Inventory
Total:	Group By	Group By	Expression
Sort:			Descending
Show:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Criteria:		Between #10/1/2018# And #10/31/2018#	
or:			



Sales Invoice #	Date	Invoice Total
102	10/15/2018	4394
101	10/15/2018	1447
103	10/28/2018	898
104	10/31/2018	789

TABLE 4-12 Expression Builder for Query 4



QUERY 5

Query 5 will answer this question: What are total sales by salesperson?

This query is similar to Query 4, except that we total invoices by salesperson rather than by invoice number. We are also not confining our query to the month of October. Try coming up with the query by yourself. The completed query and the answer are shown in Table 4-13.

DATABASE SYSTEMS AND THE FUTURE OF ACCOUNTING

Database systems have the potential to alter external reporting significantly. Considerable time and effort are currently invested in defining how companies should summarize and report accounting information to external users. In the future, companies may make a copy of the company’s financial database available to external users in lieu of the traditional financial statements. Users would be free to analyze the raw data however they see fit.

A significant advantage of database systems is the ability to create ad hoc queries to provide the information needed for decision making. No longer is financial information available only in predefined formats and at specified times. Instead, powerful and easy-to-use relational database query languages can find and prepare the information management needs whenever they want it.

Relational DBMSs can also accommodate multiple views of the same underlying phenomenon. For example, tables storing information about assets can include historical costs as well as current replacement costs and market values. Thus, managers will no longer be forced to look at data in ways predefined by accountants.

Finally, relational DBMSs are capable of integrating financial and operational data. For example, customer satisfaction data can be stored in the database, giving managers a richer set of data for decision making.

Relational DBMSs have the potential to increase the use and value of accounting information. Accountants must understand database systems so they can help design and use the

TABLE 4-13 Completed Query 5 and Answer

Sales by Salesperson

Diagram showing relationships between tables:

- Sales** (Primary Key: Sales Invoice #) is connected to **Sales-Inventory** (Primary Key: Sales Invoice #, Foreign Key: Item #) with a 1-to-many relationship.
- Sales-Inventory** (Foreign Key: Item #) is connected to **Inventory** (Primary Key: Item #) with a 1-to-many relationship.

Query Design Grid:

Field:	Salesperson	Total Salesperson Sales: Sum([Inventory]![Unit Price]*[Sales-Inventory]![Quantity])
Table:	Sales	
Total:	Group By	Expression
Sort:		
Show:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Criteria:		
or:		

Query Results:

Salesperson	Total Salesperson Sales
J. Buck	6230
S. Knight	5292

Record: 1 of 2

AISs of the future. Such participation is important for ensuring that adequate controls are included in those systems to safeguard the data and ensure the reliability of the information produced.

Summary and Case Conclusion

Ashton prepared a report for Scott and Susan summarizing what he knew about databases. He explained that a database management system (DBMS), the software that makes a database system work, is based on a logical data model that shows how users perceive the way the data is stored. Many DBMSs are based on the relational data model that represents data as being stored in tables. Every row in a relational table has only one data value in each column. Neither row nor column position is significant. These properties support the use of simple, yet powerful, query languages for interacting with the database. Users only need to specify the data they want and do not need to be concerned with how the data are retrieved. The DBMS functions as an intermediary between the user and the database, thereby hiding the complex addressing schemes actually used to retrieve and update the information stored in the database.

After reading Ashton's report, Scott and Susan agreed that it was time to upgrade S&S's AIS and to hire a consulting firm to help select and install the new system. They asked Ashton to oversee the design process to ensure that the new system meets their needs.