CHAPTER 2

Database Concepts and Applications in Human Resource Information Systems

Janet H. Marler and Barry D. Floyd

EDITORS' NOTE

As mentioned in the book overview in Chapter 1, this chapter is focused on understanding databases and the applications of IT to the development and use of an HRIS. Although this chapter may be a review for some students, the material in it is critical to understanding the remaining chapters of the book. As such, students may want to refer to this chapter as they are studying subsequent chapters. This introductory chapter is also an excellent example of the contribution of IT to the field of HRM in building an HRIS.

CHAPTER Objectives

After completing this chapter, you should be able to

- Discuss the difference between data, information, and knowledge
- Identify problems with early database structures
- Understand what a relational database is and why it is better than older database structures
- Discuss three types of data sharing and why they are important
- Know where data in a database are stored
- Know the different ways in which data can be delivered to the end user
- Know what a query is and discuss three different types of queries
- Discuss how queries are used to support decision making
- Discuss the key steps involved in designing a simple database in Microsoft (MS) Access
- Know what the top HR databases are
- Identify key data fields in an HR database
- Understand the difference between operational databases and a data warehouse
- Discuss how business intelligence software can support HR decision making

Introduction

Whether an organization purchases, leases, or develops its HRIS, the data and the information it produces are stored and retrieved through a database. Today's HRIS have as their foundation electronic databases that work in conjunction with **business applications** to transform data into information that is essential for business operations and for decision making. Many believe that managing electronic databases and turning data into accessible and actionable information is a competency necessary to succeed in today's marketplace. Indeed, data are produced, stored, updated, and shared by HR employees and managers on a daily basis. This process is so pervasive that it often goes unnoticed. Yet, the effective collection, storage, integration, and use of data are essential for any business, and the most successful organizations are masters of this process!

In this chapter, we provide an insight into how commercially delivered HRIS databases work. We define key **relational database** terminology, describe how a database is structured, and show how to develop a basic database using **MS Access**, a basic **database management system (DBMS)**, as an example. We discuss how DBMSs provide the capability to integrate HR data and to link this data with other data essential to the operations of a business. We close by providing examples of HRIS built on MS Access to provide a basic understanding of larger, more complex commercially developed HRIS databases.

Data, Information, and Knowledge

Data are the lifeblood of an organization. The production and maintenance of data are critical to the smooth operation of every part of the organization. Data represent the "facts" of transactions that occur on a daily basis. A transaction can be thought of as an event of consequence, such as hiring a new employee for a particular position for a specified salary. The organization attempts to capture the data (facts) associated with each of these transactions, such as the date hired, the name of the person hired, the title of the position, the location where the new hire will work, and so on, and then store these data for future use.

Information, on the other hand, is the interpretation of these data. An interpretation of data always has some goal and context such as making a hiring decision for a particular department or understanding the performance of an employee to make a promotion decision. Note that sometimes the data themselves can be informative without any additional transformation (e.g., the salary range of the job). But other times, we must do additional work (e.g., calculating totals or presenting the data in some order) to turn the data into information to answer important questions such as "What is our full-time employee headcount in Corporate Sales?" or "Which employee should be promoted?"

Knowledge is information that has been given meaning (Whitehill, 1997). Knowledge is different from data and information. More than what and why, knowledge is about *how*. Knowledge, therefore, consists of the procedures one follows to use data and information to make decisions and conduct business. In many instances, such procedural knowledge is mostly hidden, residing in the minds of individuals and groups in the organization. For example, in HRIS, facts about age, gender, and education are the data. Information created from these data includes average age, gender ratio, and number and types of graduates at the business unit level. Such data and information help HR managers plan recruitment, schedule training programs to bridge skill gaps, and identify whether there may be employee discrimination. Knowledge represents how HR managers can execute the recruitment plan, decide which training programs are best to bridge skill gaps, or determine what to do if employee discrimination exists. In the HR function, *data* about employees and jobs are the foundation of most of the *information* that is critical to analyzing and making HR decisions. *Knowledge,* on the other hand, constitutes knowing what information is needed from a database and how to use it to achieve HR objectives.

Database Management Systems

A DBMS is a set of software applications (i.e., computer programs) combined with a database. A DBMS electronically allows organizations to effectively manage data. Managing data means

- identifying the data needed to create information that is necessary to make HR decisions,
- defining the characteristics of that data (e.g., number data vs. character data),
- organizing those data in a manner that promotes integration, data quality and accessibility, and finally
- restricting access to the data to the right personnel.

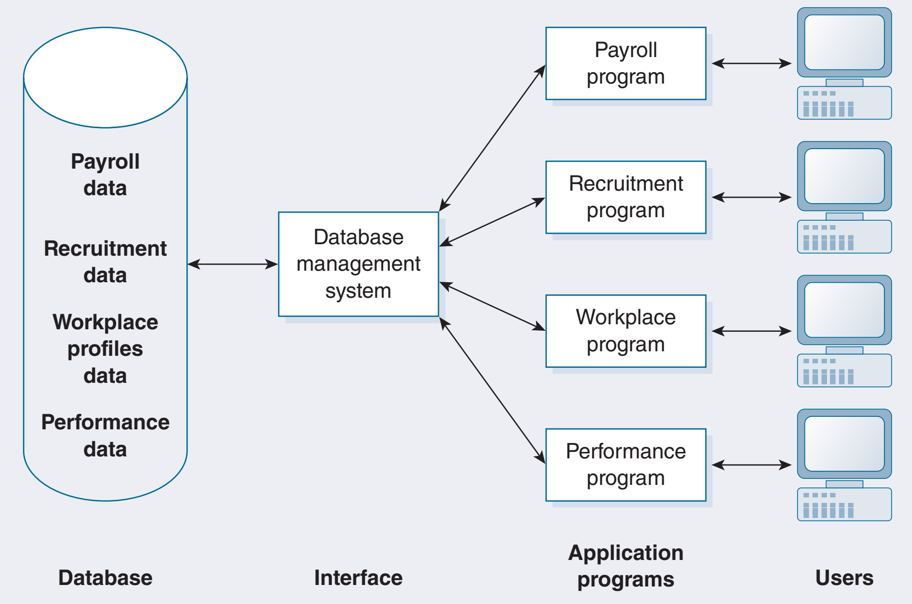
By performing these functions effectively, a DBMS turns data into an organizational resource.

A database is a set of organized data. Importantly, it is a permanent, self-descriptive store of interrelated data items that can be processed by one or more business applications. *Self-descriptive* means that the database knows about the characteristics of the data (e.g., the length of an employee's last name can be no greater than 30 characters) or that a paycheck can only be associated with one employee. *Interrelated* means that there are "links" between different sets of data in the database. For example, there can be a link between the data about employees and the jobs that they have. There can also be links between HR data and other data in the organization, such as linking a managerial position to specific company facility resources such as office space or a production facility. As a central repository of data, many different business applications and users can access the data, making an organization's database a very valuable organizational asset that, therefore, needs to be managed appropriately.

The main functions of a DBMS are to create the database; insert, read, update, and delete database data; maintain data integrity (i.e., making sure that the data are correct) and security (i.e., making sure that only the right people have access to the data); and prevent data from being lost by providing backup and recovery capabilities. Database management systems are also designed to have high performance, allowing data to be retrieved quickly by the many users in the organization.

DBMSs and databases work in conjunction with business applications, such as **transaction processing systems (TPS)**, to make organizations run smoothly. As shown in Figure 2.1, these business applications consist of a set of one or more computer programs that serve as an intermediary between the user and the DBMS while providing the "functions" or "tasks" that the user wants performed (e.g., store data about a new hire) (Kroenke, 2003). The business application must talk both to the user sitting at a computer terminal in an easy-to-use manner and to the database in a way that is very efficient. For example, a payroll business application involves collecting data from an employee's time card, storing these data in a database, and then retrieving and manipulating these data to produce a paycheck. Data from this transaction processing system can also be used to generate reports on monthly personnel expenses. These reports are the basis of **management reporting systems (MRS)**. We'll talk more about these later in the chapter.

Figure 2.1 **Database, Database Management System, and Business Applications**



There are thousands of commercially available business applications that work in conjunction with a DBMS to process business transactions. In a 2000 census of comprehensive HR software for the HR function, Richard Frantzreb catalogued more than 150 HR applications (Meade, 2003). In another census of specialized HR products under headings such as employment management, Equal Employment Opportunity (EEO), training management, career development, HR planning, performance management, personnel policy, survey processing, employee scheduling, attendance/timekeeping, payroll, and so on, Frantzreb counted 2,500 HR software products from about 1,700 vendors (Meade, 2003).

Early DBMSs[1]

Early DBMSs were simply data-processing systems that performed record-keeping functions that mimicked existing manual procedures. Thus, electronic data were stored in computers in much the same way that they were stored in paper filing systems. Paper filing systems typically consisted of a filing cabinet and a drawer for each type of business document (e.g., an employee personnel form). These documents were also called "records." Inside would be paper documents with each document being a "record" of a transaction (e.g., promoting Susan to senior manager). Computer systems mimicked this, creating individual computer files, typically one for each type of document. For example, there would be an Employee File with employee records, a Time Card File with time card records, multiple Employee Benefit Files with their associated documents, and so on. The main objective of these file-processing systems was to process transactions such as update payroll records and produce payroll checks as efficiently as possible. The goal was not on data sharing among different business applications and users.

These traditional **file-oriented data structures** had a number of shortcomings. These shortcomings included: (a) data redundancy—an employee's name and address could be stored in many different files; (b) poor data control—if you had access to the file you had access to *all* the data in the file, which may not be desirable because you may want to restrict the data viewed by a particular user; (c) inadequate data manipulation capabilities—it was very difficult to combine the data across files and to easily update and to add new data; and (d) excessive programming effort—any change in the structure of the data (e.g., adding a new field such as a mobile phone number or a screen name to an employee record) required extensive changes in the software program that accessed the data.

In general, early file systems were good at specialized transaction processing. They were not designed to easily and quickly provide information to answer questions such as "What was the average hourly wage for female programmers last year compared with this year?" because the data to answer the more complicated questions came from different files; for example, employee gender and salary would be in the **master file on employees**, and hours worked would be in the time-card transaction file. Difficulties also arose when managers in the organization wanted to share data across applications: Fundamentally, there was no easy way to "link" information. For example, managers could not connect information about employee salaries and sales projections.

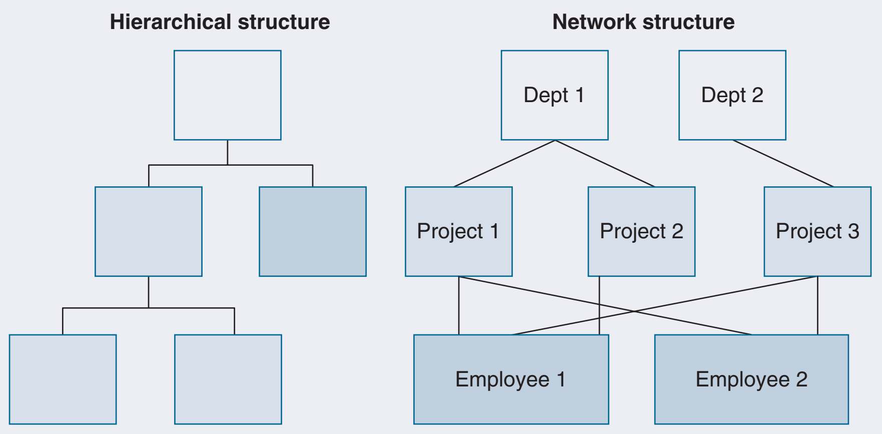
To overcome the shortcomings of file-oriented structures, *hierarchical and network database systems* evolved in the mid-1960s and early 1970s. The key to these systems was that relationships between different records were explicitly maintained. Although relationships among the data were created between sets of data, as illustrated in Figure 2.2, the relationships were created based on where the data were stored (e.g., the job records for Employee X are located in Sector 3 of Disk 4). Thus, only the very knowledgeable technical staff was able to effectively interact with the database. These database systems also required an excessive programming effort and suffered from inadequate data manipulation capabilities if the program was poorly designed.

The advent of *relational database management systems* addressed the many problems associated with these older DBMSs and database structures.

Relational DBMSs

In 1970, E. F. Codd introduced the notion that rather than programming relationships between data based on physical location, the information needed to integrate data should

Figure 2.2 Hierarchical and Network Database Structures



reside within the data (Hansen & Hansen, 1996). Included in Codd's proposal was that data be stored in tables where each table represented one "entity" in the real world and the information associated with that "entity"' be stored only in that table. For example, a company could have an employee table (i.e., employee is an "entity"), and so information about the employee, such as name, address, and date of hire, would only be stored in that table and nowhere else. Such an idea removed problems with redundancies such as storing the employee's address in many locations and then not knowing which one is the correct one, if the employee's address is changed in one location and not in the other location. These tables were called relations, and from this model came the name *relational database*.

In relational database systems, retrieval of data from different tables was based on logical relationships built into the table structures, which made feasible the creation of a query capability that was much more accessible to end users who generally had limited programming experience. This technique also allowed for relationships to be easily built among all the entities in the organization. We'll talk more about this a bit later in the chapter.

Perhaps, the most significant difference between a file-based system and a relational database system is that data are easily shared. There are three types of *data sharing*: (1) data sharing between functional units, (2) data sharing between management levels, and (3) data sharing across geographically dispersed locations. Data sharing requires a major change in end-user thinking, particularly in those employees who are accustomed to owning their own data on their PCs. Fundamentally, sharing data means sharing power because both data and information are power. Sharing data also means being a good citizen and making certain that the data you enter is correct.

Data Sharing Between Different Functions

Relational DBMSs facilitate data integration across different functions such that each function might have access not only to its own data but also to other data as well. Thus, the HR department is able to maintain its employee database but also access cost information from the accounting department's database. As a result, relational database technology increased the feasibility and popularity of integrated business applications. These integrated applications used in large organizations are referred to as **enterprise resource planning (ERP)** business applications.

ERP software applications are a set of integrated database applications, or modules, that carry out the most common business functions, including HR, general ledger, accounts payable, accounts receivable, order management, inventory control, and customer relationship management. ERP modules are integrated, primarily through a common set of definitions and a common database (Martin, Brown, DeHayes, Hoffer, & Perkins, 1999).

Data Sharing Between Different Levels

Operational employees, managers, and executives also share data but have different objectives and, thus, different information needs. Operational employees focus on data-processing transactions to ensure smooth operation of critical business transactions. A common business transaction is processing the information from an employee's timecard. At this level, transaction-processing information systems help conduct business on a day-to-day basis to provide timely and accurate information to managers and executives. For example, transaction-processing systems update employee work history, attendance, and work hours. Operational employees are concerned with the accuracy and efficiency with which these data are processed.

Managers, on the other hand, are more interested in summary data, such as reports generated from daily operational data that can be summarized into daily, weekly, or monthly reports on hours worked by employee or absences by employee.

Executives rely on information produced at an even more aggregated level to evaluate trends and develop business strategies. For example, executives might ask for reports that compare turnover statistics across business groups and over time.

These three different levels of use correspond to three different types of software systems that have evolved over the past three decades: transaction processing systems (TPS), management reporting systems (MRS), and **decision support systems (DSS)** (Hansen & Hansen, 1996). TPS were first applied to lower operational levels of the organization to automate manual processes such as payroll. Their basic characteristics include (a) a focus on data storage, processing, and flows at the daily operational level; (b) efficient transaction processing; and (c) summary reports for management (Sprague & Watson, 1989). Early ERP applications were used primarily for their transaction processing functionality.

Note the similarity between the categorization of information systems into electronic data processing (EDP), management information systems (MIS), and decision support systems (DSS) discussed in Chapter 1 (Sprague & Carlson, 1982). These terms correspond to TPS, MRS, and DSS in this chapter. As you may recall from Chapter 1, an additional information system was identified—the human resources management decision system (HRMDS). The HRMDS was described as consisting of the reports managers and HR professionals receive on a regular basis but that are actually used in their daily work, *particularly in their decision-making capacity.* The HRMDS could be classified as a special instance of an MRS or MIS system but focused specifically on information used in decision making—a central theme of this book.

In addition to TPS capabilities, relational databases can also provide MRS capability. Characteristics of an MRS include (1) information aimed at middle managers; (2) integration of TPS data by business functions such as manufacturing, marketing, and HR; and

(3) inquiry and report generation from the database (Sprague & Watson, 1989). Management reporting systems can be designed to provide daily, monthly, quarterly, or annual summary reports of key transactions such as employee headcounts by department or distribution of employee performance reports to meet budgets and manage performance.

Decision support systems assist senior managers and business professionals in making business decisions. Data mining, data analytics, and **business intelligence (BI)** are examples of information derived from a DSS, which relies on data warehouses. Data warehouses represent aggregated data (e.g., the total salary information by department by month) collected from various databases available to a business.

Data Sharing Across Locations

In today's global environment, access to data from any physical location in the world is increasingly important. Teams of employees may be stationed in Thailand, India, and the United States. Two issues arise when data are shared across wide geographic locations. These are (1) managing the day/time of a transaction and (2) determining where to store the various components of the business application, DBMS, and database.

To deal with day/time, developers of DBMSs such as Oracle, MS SQL Server, and IBM DB2 are building the capability to deal with recording dates and times according to the time zone in which the data originated. So, for example, if a database is stored in London and an employee records a transaction while sitting at a terminal in Los Angeles, in addition to the time (say 1 P.M. in Los Angeles), the time zone (-08:00 from Greenwich Mean Time) is also stored with the transaction.

As part of a global information system design, organizations have chosen to break their business application and DBMS into components, often called "tiers." More detail on tiers will also be covered in Chapter 3. Traditional client-server architectures broke an application into two tiers, typically with the user interface and some business logic on the user's computer, such as a PC (the client) and the database and mainstream parts of the application stored on a server. In today's global environment with high-speed data networks, **N-tier architectures** exist with databases and applications being distributed among many different computers around the world. So if, for example, you are in an Internet café in Bangkok trying to get information about your benefit election, the hosting computer may be in London and the data may be located on a computer in Chicago. In sum, computer networks are created that provide instant access to these operational data, allowing real-time managerial decision capability regardless of physical location.

A centralized database allows a company to confine its data to a single location and, therefore, more easily control data integrity, updating, backup, query, and control access to the database. A company with many locations and telecommuters, however, must develop a communications infrastructure to facilitate data sharing over a wide geographical area. The advent of the Internet and a standardized communication protocol made the centralized database structures and geographically dispersed data sharing feasible.

Key Relational Database Terminology

As discussed earlier, relational DBMSs are used to store data important to the organization. Key terms in relational database management include entities, attributes, tables, primary keys, foreign keys and relationships, queries, forms, and reports. Below we define each term and describe its function in a database.

Entities and Attributes

Entities are things such as employees, jobs, promotion transactions, positions in company, and so on. They include both physical things such as desks and conceptual things such as bank accounts. A company must analyze its business operations and identify all the entities that it believes are important.

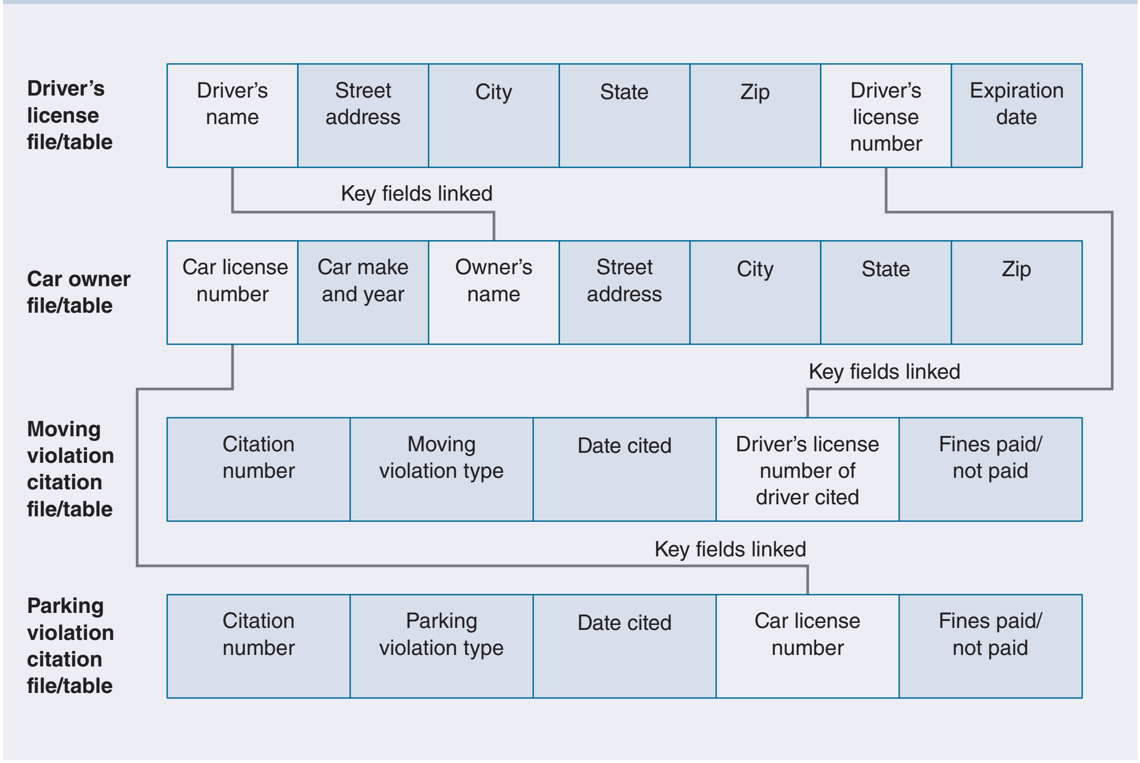
Each of these entities is made up of "attributes." An attribute is a characteristic of the entity. For example, an employee has a name, address, phone number, education, and so on. Attributes also have characteristics such as the type of data (e.g., date, number, or character) and size (e.g., number of characters or the largest number that can be stored).

In addition to identifying the entities and attributes, the relationships among the entities must be defined. For example, a company may have an employee entity and a department entity. Then the company must define the relationship between the employee entity and the department entity (e.g., Does an employee have to be assigned to a department? Can an employee be assigned to more than one department?).

Tables

How does this information fit into a relational DBMS? Tables are used to store information about entities. As illustrated in Figure 2.3, one table is created for each entity—in this example, driver table, car table, moving violation table, and parking violation table. Attributes are stored as the columns (also called fields) in the table. As noted earlier, attributes represent a single data element or characteristic of the data table. For example, a table of driver data would have the following columns or characteristics: first name, last

Figure 2.3 **Relational Database Structure**



name, street address, city, state, driver license number, expiration, and so on. Each of these characteristics represents an attribute or field of the table.

Each table in a database contains rows. Rows are also referred to as records and represent an "instance" of the entity. For example, in the driver table, each row contains data about a particular driver, and each column contains data that represent an attribute of that driver, such as name, phone number, and license number.

Relationships, Primary Keys, and Foreign Keys

To represent the relationships among the tables, we have to do a bit more work. In a relational DBMS, relationships are created by having the same attribute in each table with the value of the attribute being the same in each table. Most often this is done by taking the "primary key" of one table and including it in the related table. What is a primary key?

Typically, each entity has an attribute that has unique values for each instance of the entity. For example, each employee has a Social Security number that is unique (i.e., only one person has a particular number). Other entities, such as jobs, locations, and positions can be assigned a unique number if one doesn't exist. These unique attributes can be used as a table's primary key. Given that we have a unique attribute, to create a relationship, we simply store that attribute in the related table. So if an employee is associated with a position, we have two tables, an employee table and a positions table. We then take the primary key of the employee table and store it in the position table. In the example in Figure 2.3, the driver's license number is the primary key in the driver table and it is also stored in the moving violation table. When a primary key from one table is stored as an attribute of another table, that attribute is called a **foreign key**. Thus, in Figure 2.3, driver's license is the primary key in the driver table and is the foreign key in the moving violations table.

Storing data in related tables allows users to utilize the database application to create queries, forms, and reports that retrieve, update, or analyze data from multiple tables together. The relationships between the tables allow users to accurately combine information that "go together" from two (or more) tables. For example, if a manager wished to provide bonuses to his or her top salespeople, he or she would likely use data from an employee file, a sales file, and some type of compensation criterion table.

Queries[2]

A query is a question that you ask about the data stored in a database. For example, you may want to know which employees live within a specific city. You could generate these results by scrolling through the relevant table or by sorting the table by city and then looking at the result, but this is both time-consuming and you would have to do this task each time you wished to find the answer to your question. A better approach is to create a query. A query is a structured way of posing your question to the DBMS in a language the DBMSs understand. This definition (e.g., show all employees with city Albany) can be saved in the database and used again and again. Importantly, each time the query is executed it searches through the *current* table records and lists the results. The results of a query on a table(s) are always displayed in something that looks just like another table. However, this result table is only temporary and is not stored in the database. It is important to note that queries do not store data! All data are stored in tables. Queries only report on data currently in the table.

There are three different kinds of queries: select queries, action queries, and cross-tab queries. A **select query** allows you to ask a question based on one or more tables in a database. This is the most commonly used query. These queries can be quite general or quite specific. For example, a general query might extract all employees from the database who have reached retirement age. A more specific query might retrieve employees who have reached retirement age and who live in New York and are engineers.

An **action query** performs an action on the table on which it is based. Actions include updating data in the table (e.g., increasing the base salary of all employees who were rated above average in the latest performance rating), deleting records from the table (e.g., removing employees from the employees table if they no longer work at the company), or inserting records (e.g., the query may add a new set of benefits to the benefits table). You can also use this type of query to create new tables.

A **cross-tab query** performs calculations on the values in a field and displays the results in a datasheet. The reason it is called "cross-tab" is that it tabulates the data for a set of descriptor attributes, contrasting them or crossing them in a table format. For example, we might want to see the total personnel count by gender by region. So we would see the gender on the left-hand side and the different regions listed across the top of a table. A cross-tab query could display different aspects of the data, including sums or averages or minimum or maximum values. As another example, a cross-tab query could determine headcount by department or determine pay range maximums and minimums in pay grades by department.

Select queries and cross-tab queries provide the information that managers and executives expect from IT. These queries can serve as the foundation for MRS and DSS information and decision making. Action queries, on the other hand, improve the operational efficiency of managing and maintaining a database and are most closely associated with TPS. These tasks are important to the operational staff but of less interest to HR managers and executives.

Queries are also used as the basis for forms and reports. In addition to retrieving data, they can add, update, and delete records in tables. You can define fields in a query that perform calculations, such as sums and averages. The following list summarizes the typical capabilities of queries (Bast, Cygman, Flynn, & Tidwell, 2006):

- Display selected fields and records from a table
- Sort records on one or multiple fields
- Perform calculations
- Generate data for forms, reports, and other queries
- Update data in the tables of a database
- Find and display data from two or more tables
- Create new tables
- Delete records in a table based on one or more criteria

Forms[3]

A form is an object in a database that you can use to maintain, view, and print records in a database in a more "structured" manner. Although you can perform these same functions with tables and queries, forms can present data in many customized and useful ways. For example, you can design a form to look like the time sheet submitted by an employee. Well-designed forms can improve data input efficiency and accuracy. Consequently, forms represent the main mechanism for creating end-user interfaces.

A form can be based on a table, multiple tables, or queries. A form can display one record at a time or many records. Often, we select only one record and then create a nice-looking, easy-to-use layout to work with the data in that one record. To view and maintain or add data using a form, you must know how to move from field to field and from record to record. Forms provide navigation buttons that facilitate moving from field to field and from record to record. Data that are entered or changed in a form automatically change the values in the underlying table once you save the changes.

Reports

A report is a formatted presentation of data from a table, multiple tables, or queries that is created as a printout or to be viewed on screen. Data displayed in a report are dynamic, reflecting the latest data from the tables on which the report is based. Unlike forms, however, you cannot change the data or add a new record in a report. You can only view the data in a report.

Although you can print data appearing in tables, queries, and forms, reports provide you with the greatest flexibility for formatting printed output. As with forms, you can design your own reports or use a report wizard to create reports automatically.[4]

Introduction to MS Access

MS Access is a relational DBMS in which data are organized as a collection of tables. Like any relational database, the data in tables can be queried. MS Access also makes it easy to create forms and reports through the use of form or report wizards. A form or report wizard is a computer program or tool that guides you through the creation of a form by asking you a series of questions. For example, which table is the form to be created from, and which attributes do you want to be displayed on the form? The form or report is created based on your answers.

MS Access is designed for relatively small databases and assumes limited knowledge of database programming. MS Access provides the following functions (Adamski & Finnegan, 2005):

- It allows you to create databases containing tables and table relationships.
- It lets you easily add new records, change table values in existing records, and delete records.

- It contains a built-in query language, which lets you obtain immediate answers to questions you ask about your data.
- It contains a built-in report generator and report wizard, which lets you produce professional-looking, formatted reports from your data.
- It provides protection of databases through security, control, and recovery facilities.

Data in an MS Access table or query can be exported to other database applications or to spreadsheet programs such as Excel or Lotus 123. Once these records are in a spreadsheet program, then further analyses may be conducted and graphs and charts constructed to enhance analytical HR metric reports. Data can be exported by simply opening the database that has the object—for example, table or query—that you want to export. Then select File, Export from the database menu. Select the type of file—for example, .xls—you want the object to be saved to and specify a name. Click Save. Now you can open the file in Excel. You may also "link" the data in the database to the spreadsheet. When the spreadsheet is opened, the most recent data from the database are retrieved and presented in the spreadsheet.

Unlike spreadsheet software programs, MS Access handles substantially more data and contains the ability to model relationships. Each MS Access database, for example, can be up to 2 GB in size and can contain up to 32,768 objects, including tables, queries, forms, reports, and so on.

Designing an MS Access Database

The design process begins with an analysis of the data and information that the users of the database will need to have stored and retrieved in order to accomplish their work. Typically, we think of work as consisting of tasks within a business process, and so we can think of the data that will be required to be stored in a database and of the information that will need to be extracted. We find out the data to be stored by interviewing the intended end users of the database. We ask about entities that they need to keep information on, the attributes of those entities, and also how the entities are related. In addition, we may watch users at work and look at the forms, reports, and other business documents that they use to be successful. Gathering copies of all existing forms and reports currently used may also act as guidelines for creating forms and reports, though sometimes our intention is to change how they are doing business, and so some of these documents may be significantly changed or even discarded.

In general, the database design process can be broken down into several steps that are somewhat sequential but oftentimes have to be repeated until the database meets the users' needs:

- Determine what the users want from the database: What questions need to be answered? What information needs to be tracked? What reports are produced? What data are needed to provide the basis for those results?

- Identify the data fields needed to produce the required information; in doing so, we also identify rules that define the integrity of the data, including data type (number, character) and data limits (e.g., if we are storing days, we might only allow the numbers 1 to 31).
- Group related fields into tables (entities).
- Determine each table's primary key.
- Normalize the data: Make sure the data for an entity are really associated with only that entity.
- Determine how the tables are related to one another and include common keys.
- Create the relationships among the different entities and insure referential integrity.
- Create queries to define data needs that are not handled by only looking at individual tables.
- Create reports to provide a structured view of the data.
- Create forms, and in doing so, identify a common design for the forms: Typically, we create a form for each table along with a "main menu" form that allows the user to navigate to each form associated with a table and to view queries and reports.
- Enter test data to verify the quality/accuracy of the system design.
- Test the system: Do all the queries work correctly? Are the forms easy to use? Are the end users happy?
- Enter or populate the database.

HR Database Application Using MS Access

For small companies, generally with fewer than 1,000 employees, there are commercially available HR database applications based on MS Access. One such system, popular in the United States, is HRSource from Auxillium West (www.auxillium.com). Both software products offer wide breadth of functionality and flexibility to import and export data from and to Excel and to integrate with other database applications, particularly payroll. Both provide a centralized relational database with basic transaction processing and management reporting systems.

Both HRSource and HRVantage have familiar MS Access forms as user interfaces. They both allow users to create custom queries and reports. However, the database applications also come with preconfigured reports and queries. For example, HRVantage provides more than 150 standard reports, which include Absence reports, EEO reports and graphs, termination analyses, employee skill searches, employee profiles, OSHA reports, employee performance reports, and many others. HRSource offers users 70 built-in reports. Customers also claim that with a little expertise in MS Access, they are able to mine their HR information in a way that they never could before they centralized on one HRIS database (Meade, 2003).

Other HR Databases

A few decades ago, database application programs were often written by companies for their particular use; in today's business environment, customized application programs termed *legacy systems* are being replaced by commercially developed HR systems supported by enterprise database application programs (e.g., PeopleSoft Enterprise HCM, MySAP ERP HCM, Lawson HCM, Epicor HCM, SuccessFactors Employee Central, UltiPro HR, Workday). The most well-known HR database applications have the capability to operate on various DBMS platforms (e.g., Oracle, SQL Server, DBS2). These commercial database application programs can either be licensed and installed onto computer hardware a company buys themselves, or they can be accessed remotely through an approach called software as a service or "SaaS" The SaaS approach to HRIS is discussed further in Chapters 3 and 17.

Regardless of how complex your HRIS DBMS is, you must ensure that you know what information can be derived from any database. To know this, one must have an idea of what tables and attributes (fields) are in the database. Software vendors should be able to provide this information to end users; however, for the large, complex HR applications, this may run into thousands of tables and fields! Auxillium West, makers of a low-priced HRIS, offer a document to prospective customers that lists the data items commonly tracked. Their most commonly tracked HR fields are listed in Table 2.1 (Meade, 2003).

Although the list in Table 2.1 appears to be comprehensive, in fact, it is quite sparse when compared with more complex database applications. More complex database applications will also have fields that relate to business processes other than HR, such as accounting and finance. Integrated databases allow sophisticated queries and analytical reports, such as hours spent on recruiting, recruiters' hourly pay, job board posting costs, number of positions filled, number of declined offers, number of open positions, number of voluntary terminations, and number of involuntary terminations.

Data Integration: Database Warehouses, Business Intelligence, and Data Mining

An organization's ability to generate meaningful information to make good decisions is only as good as its underlying database. As Dr. John Sullivan notes, "I have found that the

Table 2.1 Examples of Common Fields in an HR Database

Employee ID	Job Code/Title
First Name	Pay Rate Type
Last Name	Rate Effective Date
Address	Salary
City	Bonuses
State	Status
ZIP Code	Category (full-time/part-time)
Home Phone Number	Contract Employee Status
Gender	Department
Ethnic Code	Office Information
Birth Date	Manager
Veteran Status	Division/Location
Visa Expiration Date	Company Property
Education	Emergency Contact
Past Employment	Time-Off Accruals
Skill Code	Benefits
Training/Certification	Work-Related Injuries
Performance Rating	Disability
Next Review Date	
Hire Date	
Termination Date	
Termination Reason	
Rehire Date as Applicable	

largest single difference between a great HR department and an average one is the use of metrics" (Gur, 2006). Metrics are measures of organizational performance outcomes that are derived from measures of important individual and organizational outcomes (e.g., individual job performance and absentee rate). As was discussed in Chapter 1, the current emphasis in HRM is functioning as a strategic business partner. A prerequisite for meeting this goal is the use of metrics to assess and monitor quantitative data from HRM programs like recruiting and training effectiveness. The primary objective of measuring HR metrics is to improve individual and organizational effectiveness. The use of metrics in HRM programs will be discussed in greater detail in Chapter 6. Chapter 7 will describe how to use these metrics to compute cost-benefit analyses (CBAs) and return on investment (ROI) calculations.

Much of the measurement data to create HR metrics will come from an organization's data warehouse. A data warehouse is a special type of database that is optimized for reporting and analysis and is the raw material for management's decision support system. Business intelligence, BI, is a broad category of business applications and technologies for creating data warehouses to analyze and provide easy access to these data in order to help organizational users make better business decisions. BI applications include the activities of decision support systems, query and reporting, statistical analysis, forecasting, and data mining.

BI systems allow organizations to improve business performance by leveraging information about customers, suppliers, and internal business operations from databases across functions and organizational boundaries. Essentially, BI systems retrieve specified data from multiple databases, including old legacy file database systems, and store these data into a new database, which becomes that data warehouse. The data in the data warehouse can then be accessed via queries and used to uncover patterns and diagnose problems.

Patterns in large data sets are identified through data mining, which involves statistically analyzing large data sets to identify recurring relationships. For example, data mining an employee database might reveal that most employees reside within a group of particular ZIP codes. This may help if the organization wants to supply transportation or encourage car pooling. Data mining is relatively new to business analytics and has not yet been widely used for HRM decisions.

BI systems also provide reporting tools and interfaces (e.g., forms) that distribute the information to Excel spreadsheets, Internet-based portals, PDF files, or hard copies. These results can also be distributed to key executives in specialized formats known as executive dashboards, which are becoming a popular executive decision support tool.

A major reason for a DBMS is to provide information from various parts of the organization in an "ad hoc" manner. Ad hoc means that a user can ask a question of the data that no one has thought about yet. The user can sign into the data and pose his or her question in the form of a query. This is a very powerful concept that enables all levels of the organization. Data warehouses and BI software enable managers to create information from an even greater store of data.

SUMMARY

In this chapter, we have described the key aspects of current DBMS technologies and how they work to create, store, and manage critical data about an organization. Data are transformed into information by relational DBMSs and business applications that work together. The underlying data in a database are collected from business transactions and stored in tables that are related to each other through shared fields called primary and foreign keys. Queries represent questions asked of the data and are used to access specific data stored in tables. The results of queries can be viewed in forms or reports that are customized so that the end user can better interpret the data that are retrieved from the database. More sophisticated data analyses and reports such as executive dashboards are produced from specialized databases called data warehouses and business application software called BI software.

Most HRIS rely on an underlying database. Understanding how database systems work, therefore, is relevant to HR decision makers because knowledge about how to create, store, and access data can be a key differentiator in a competitive environment. Small HR databases can be created using MS Access, or more sophisticated ones can be purchased from software vendors. There are literally hundreds of HR database business applications that create, process, and analyze HR data. The challenge is to find one that can most cost-effectively collect and share data from which meaningful information can be extracted to support making good decisions.

KEY TERMS

action query 47

business applications 35

business intelligence (BI) 43

cross-tab query 47

database management system (DBMS) 36

decision support systems (DSS) 42

enterprise resource planning (ERP) 41

file-oriented data structures 39

foreign key 46

management reporting systems (MRS) 38

master file on employees 39

MS Access 36

N-tier architectures 43

relational database 36

select query 46

transaction processing systems (TPS) 38

DISCUSSION QUESTIONS

1. Explain the differences between data, information, and knowledge.
2. What are the main functions of a database management system,

and how is it different from a database?

3. What were the shortcomings of early file-oriented database structures?
4. What are the three types of data sharing?
5. Define the key terms in a relational database.
6. What is the difference between a primary key and a foreign key?
7. What are the three types of queries?
8. How are forms and reports similar, and how are they different?
9. Take the list of HR database common fields and group them into tables.
10. What are the differences between data warehouses, BI, and data mining?
11. Can knowledge be turned into a database?

CASE STUDY

You have been asked to create an applicant database for a small recruiting firm that specializes in recruiting HR professionals for small to medium-sized firms. Describe the process that you would use to design this database. Use MS Access to develop a prototype of the database that you could show your manager.

STUDENT STUDY SITE

Visit the Student Study Site at **http://www.sagepub.com/kavanagh3e** for additional learning tools such as access to SAGE journal articles and related web resources.

NOTES

1. For a more detailed discussion, see Hansen and Hansen (1996, pp. 52–56).
2. For a more detailed discussion, see Bast et al. (2006, chap. 3).
3. For a more detailed treatment, see Tutorials 4 and 5 in Adamski and Finnegan (2005).
4. For a more detailed treatment, see Tutorials 4 and 6 in Adamski and Finnegan (2005).

REFERENCES

Adamski, J., & Finnegan, K. (2005). *New perspectives on Microsoft Access 2003.* Boston: Course Technology Thomson Learning.

Bast, K., Cygman, L., Flynn, G., & Tidwell, R. (2006). *Succeeding in business with Microsoft Office Access 2003.* Boston: Course Technology Thomson Learning.

Gur, Z. (2006, June/July). Up.link. IHRIM. *link,* 5.

Hansen, G. W., & Hansen, J. V. (1996). *Database management and design.* Upper Saddle River, NJ: Prentice Hall.

Kroenke, D. M. (2003). *Database concepts.* Upper Saddle River, NJ: Prentice Hall.

Martin, E., Brown, C., DeHayes, D., Hoffer, J., & Perkins, W. (1999). *Managing information technology.* Upper Saddle River, NJ: Prentice Hall.

Meade, J. (2003). *The human resources software handbook.* San Francisco: Jossey-Bass.

Sprague, R. H., & Carlson, E. D. (1982). *Building effective decision support systems.* Englewood Cliffs, NJ: Prentice Hall.

Sprague, R., & Watson, H. (1989). *Decision support systems* (2nd ed.). Englewood Cliffs, NJ: Prentice Hall.

Whitehill, M. (1997). Knowledge-based strategy to deliver sustained competitive advantage. *Long Range Planning, 30*(4), 621–627.