Chapter 1

The Database Approach

Discussion Focus

How often have your students heard that "you have only one chance to make a good first impression?" That's why it's so important to sell the importance of databases and the desirability of good database design during the first class session.

Start by showing your students that they interact with databases on a daily basis. For example, how many of them have bought anything using a credit card during the past day, week, month, or year? None of those transactions would be possible without a database. How many have shipped a document or a package via an overnight service or via certified or registered mail? How many have checked course catalogs and class schedules online? And surely all of your students registered for your class? Did anybody use a web search engine to look for – and find – information about almost anything? This point is easy to make: Databases are important because we depend on their existence to perform countless transactions and to provide information.

If you are teaching in a classroom equipped with computers, give some "live" performances. For example, you can use the web to look up a few insurance quotes or compare car prices and models. Incidentally, this is a good place to make the very important distinction between data and information. In short, spend some time discussing the points made in Section 1.1, "Why Databases?" and Section 1.2 "Data vs. Information."

After demonstrating that modern daily life is almost inconceivable without the ever-present databases, discuss how important it is that the (database) transactions are made successfully, accurately, and quickly. That part of the discussion points to the importance of database design, which is at the heart of this book. If you want to have the keys to the information kingdom, you'll want to know about database design and implementation. And, of course, databases don't manage themselves … and that point leads to the importance of the database administration (DBA) function. There is a world of exciting database employment opportunities out there.

After discussing why databases, database design, and database administration are important, you can move through the remainder of the chapter to develop the necessary vocabulary and concepts. The review questions help you do that … and the problems provide the chance to test the newfound knowledge.

Answers to Review Questions

1. Discuss each of the following terms:

a. data

Raw facts from which the required information is derived. Data have little meaning unless they are grouped in a logical manner.

b. field

A character or a group of characters (numeric or alphanumeric) that describes a specific characteristic. A field may define a telephone number, a date, or other specific characteristics that the end user wants to keep track of.

c. record

A logically connected set of one or more fields that describes a person, place, event, or thing. For example, a CUSTOMER record may be composed of the fields CUST_NUMBER, CUST_LNAME, CUST_FNAME, CUST_INITIAL, CUST_ADDRESS, CUST_CITY, CUST_STATE, CUST_ZIPCODE, CUST_AREACODE, and CUST_PHONE.

d. file

Historically, a collection of file folders, properly tagged and kept in a filing cabinet. Although such manual files still exist, we more commonly think of a (computer) file as a *collection of related records* that contain information of interest to the end user. For example, a sales organization is likely to keep a file containing customer data. Keep in mind that the phrase *related records* reflects a relationship based on function. For example, customer data are kept in a file named CUSTOMER. The records in this customer file are related by the fact that they all pertain to customers. Similarly, a file named PRODUCT would contain records that describe products – the records in this file are all related by the fact that they all pertain to products. You would not expect to find customer data in a product file, or vice versa.

NOTE

Note: Field, record, and file are computer terms, created to help describe how data are stored in secondary memory. Emphasize that computer file data storage does not match the human perception of such data storage.

2. What is data redundancy, and which characteristics of the file system can lead to it?

Data redundancy exists when unnecessarily duplicated data are found in the database. For example, a customer's telephone number may be found in the customer file, in the sales agent file, and in the invoice file. Data redundancy is symptomatic of a (computer) file system, given its inability to represent and manage data relationships. Data redundancy may also be the result of poorly-designed databases that allow the same data to be kept in different locations. (Here's another opportunity to emphasize the need for good database design!)

3. What is data independence, and why is it lacking in file systems?

File systems exhibit data dependence because file access is dependent on a file's data characteristics. Therefore, any time the file data characteristics are changed, the programs that access the data within those files must be modified.

Data ***independence*** exists when changes in the data characteristics don't require changes in the programs that access those data. File systems lack data independence because all data access programs are subject to change when any of the file system’s data storage characteristics – such as changing a data type -- change.

4. What is a DBMS, and what are its functions?

A DBMS is best described as a collection of programs that manage the database structure and that control shared access to the data in the database. Current DBMSes also store the relationships between the database components; they also take care of defining the required access paths to those components. The functions of a current-generation DBMS may be summarized as follows:

- The DBMS stores the definitions of data and their relationships (metadata) in a data dictionary; any changes made are automatically recorded in the data dictionary.
- The DBMS creates the complex structures required for data storage.
- The DBMS transforms entered data to conform to the data structures in item 2.
- The DBMS creates a security system and enforces security within that system.
- The DBMS creates complex structures that allow multiple-user access to the data.
- The DBMS performs backup and data recovery procedures to ensure data safety.
- The DBMS promotes and enforces integrity rules to eliminate data integrity problems.
- The DBMS provides access to the data via utility programs and from programming languages interfaces.
- The DBMS provides end-user access to data within a computer network environment.

5. What is structual independence, and why is it important?

Structural independence exists when data access programs are not subject to change when the file's structural characteristics, such as the number or order of the columns in a table, change. Structural independence is important because it substantially decreases programming effort and program maintenance costs.

6. **Explain the difference between data and information.**

Data are raw facts. Information is processed data to reveal the meaning behind the facts. Let's summarize some key points:

- Data constitute the building bocks of information.
- Information is produced by processing data.
- Information is used to reveal the meaning of data.
- Good, relevant, and timely information is the key to good decision making.
- Good decision making is the key to organizational survival in a global environment.

7. **What is the role of a DBMS, and what are its advantages? What are its disadvantages?**

A **database management system** (**DBMS**) is a collection of programs that manages the database structure and controls access to the data stored in the database. Figure 1.2 (shown in the text) illustrates that the DBMS serves as the intermediary between the user and the database. The DBMS receives all application requests and translates them into the complex operations required to fulfill those requests. The DBMS hides much of the database's internal complexity from the application programs and users. The application program might be written by a programmer using a programming language such as COBOL, Visual Basic, or C++, or it might be created through a DBMS utility program.

Having a DBMS between the end user's applications and the database offers some important advantages. First, the DBMS enables the data in the database *to be shared* among multiple applications or users. Second, the DBMS *integrates* the many different users' views of the data into a single all-encompassing data repository.

Because data are the crucial raw material from which information is derived, you must have a good way of managing such data. As you will discover in this book, the DBMS helps make data management more efficient and effective. In particular, a DBMS provides advantages such as:

- *Improved data sharing*. The DBMS helps create an environment in which end users have better access to more and better-managed data. Such access makes it possible for end users to respond quickly to changes in their environment.
- *Better data integration*. Wider access to well-managed data promotes an integrated view of the organization's operations and a clearer view of the big picture. It becomes much easier to see how actions in one segment of the company affect other segments.
- *Minimized data inconsistency*. **Data inconsistency** exists when different versions of the same data appear in different places. For example, data inconsistency exists when a company's sales department stores a sales representative's name as "Bill Brown" and the company's personnel department stores that same person's name as "William G. Brown" or when the company's regional sales office shows the price of product "X" as $45.95 and its national sales office shows the same product's price as $43.95. The probability of data inconsistency is greatly reduced in a properly designed database.
- *Improved data access*. The DBMS makes it possible to produce quick answers to ad hoc queries. From a database perspective, a **query** is a specific request for data manipulation (for

example, to read or update the data) issued to the DBMS. Simply put, a query is a question and an **ad hoc query** is a spur-of-the-moment question. The DBMS sends back an answer (called the **query result set**) to the application. For example, end users, when dealing with large amounts of sales data, might want quick answers to questions (ad hoc queries) such as:

  - What was the dollar volume of sales by product during the past six months?
  - What is the sales bonus figure for each of our salespeople during the past three months?
  - How many of our customers have credit balances of $3,000 or more?

- *Improved decision making*. Better-managed data and improved data access make it possible to generate better quality information, on which better decisions are based.
- *Increased end-user productivity*. The availability of data, combined with the tools that transform data into usable information, empowers end users to make quick, informed decisions that can make the difference between success and failure in the global economy.

The advantages of using a DBMS are not limited to the few just listed. In fact, you will discover many more advantages as you learn more about the technical details of databases and their proper design.

Although the database system yields considerable advantages over previous data management approaches, database systems do carry significant disadvantages. For example:

- *Increased costs*. Database systems require sophisticated hardware and software and highly skilled personnel. The cost of maintaining the hardware, software, and personnel required to operate and manage a database system can be substantial. Training, licensing, and regulation compliance costs are often overlooked when database systems are implemented.
- *Management complexity*. Database systems interface with many different technologies and have a significant impact on a company's resources and culture. The changes introduced by the adoption of a database system must be properly managed to ensure that they help advance the company's objectives. Given the fact that databases systems hold crucial company data that are accessed from multiple sources, security issues must be assessed constantly.
- *Maintaining currency*. To maximize the efficiency of the database system, you must keep your system current. Therefore, you must perform frequent updates and apply the latest patches and security measures to all components. Because database technology advances rapidly, personnel training costs tend to be significant.
- *Vendor dependence*. Given the heavy investment in technology and personnel training, companies might be reluctant to change database vendors. As a consequence, vendors are less likely to offer pricing point advantages to existing customers, and those customers might be limited in their choice of database system components.
- *Frequent upgrade/replacement cycles*. DBMS vendors frequently upgrade their products by adding new functionality. Such new features often come bundled in new upgrade versions of the software. Some of these versions require hardware upgrades. Not only do the upgrades themselves cost money, but it also costs money to train database users and administrators to properly use and manage the new features.

8. List and describe the different types of databases.

The focus is on Section 1.3.2, TYPES OF DATABASES. Organize the discussion around the number of users, database site location, and data use:

- Number of users
    - Single-user
    - Multiuser
    - Workgroup
    - Enterprise
- Database site location
    - Centralized
    - Distributed
- Type of data
    - General-purpose
    - Discipline-specific
- Database use
    - Transactional (production) database (OLTP)
    - Data warehouse database (OLAP)
- Degree of data structure
    - Unstructured data
    - Structured data

9. What are the main components of a database system?

The basis of this discussion is Section 1.7.1, THE DATABASE SYSTEM ENVIRONMENT. Figure 1.9 provides a good bird's eye view of the components. Note that the system's components are hardware, software, people, procedures, and data.

10. What are metadata?

Metadata is data about data. That is, metadata define the data characteristics such as the data type (such as character or numeric) *and the relationships that link the data*. Relationships are an important component of database design. What makes relationships especially interesting is that they are often defined by their environment. For instance, the relationship between EMPLOYEE and JOB is likely to depend on the organization's definition of the work environment. For example, in some organizations an employee can have multiple job assignments, while in other organizations – or even in other divisions within the same organization – an employee can have only one job assignment.

The details of relationship types and the roles played by those relationships in data models are defined and described in Chapter 3, "Data Models." Relationships will play a key role in subsequent chapters. You cannot effectively deal with database design issues unless you address relationships.

11. Explain why database design is important.

The focus is on Section 1.4, WHY DATABASE DESIGN IS IMPORTANT. Explain that modern database and applications development software is so easy to use that many people can quickly learn to implement a simple database and develop simple applications within a week or so, without giving design much thought. As data and reporting requirements become more complex, those same people will simply (and quickly!) produce the required add-ons. That's how data redundancies and all their attendant anomalies develop, thus reducing the "database" and its applications to a status worse than useless. Stress these points:

- Good applications can't overcome bad database designs.
- The existence of a DBMS does not guarantee good data management, nor does it ensure that the database will be able to generate correct and timely information.
- Ultimately, the end user and the designer decide what data will be stored in the database.

A database created without the benefit of a detailed blueprint is unlikely to be satisfactory. Pose this question: would you think it smart to build a house without the benefit of a blueprint? So why would you want to create a database without a blueprint? (Perhaps it would be OK to build a chicken coop without a blueprint, but would you want your house to be built the same way?)

12. What are the potential costs of implementing a database system?

Although the database system yields considerable advantages over previous data management approaches, database systems do impose significant costs. For example:

- *Increased acquisition and operating costs*. Database systems require sophisticated hardware and software and highly skilled personnel. The cost of maintaining the hardware, software, and personnel required to operate and manage a database system can be substantial.
- *Management complexity*. Database systems interface with many different technologies and have a significant impact on a company's resources and culture. The changes introduced by the adoption of a database system must be properly managed to ensure that they help advance the company's objectives. Given the fact that databases systems hold crucial company data that are accessed from multiple sources, security issues must be assessed constantly.
- *Maintaining currency*. To maximize the efficiency of the database system, you must keep your system current. Therefore, you must perform frequent updates and apply the latest patches and security measures to all components. Because database technology advances rapidly, personnel training costs tend to be significant.
- *Vendor dependence*. Given the heavy investment in technology and personnel training, companies may be reluctant to change database vendors. As a consequence, vendors are less likely to offer pricing point advantages to existing customers and those customers may be limited in their choice of database system components.

13. Use examples to compare and contrast unstructured and structured data. Which type is more prevalent in a typical business environment?

Unstructured data are data that exist in their original (raw) state, that is, in the format in which they were collected. Therefore, unstructured data exist in a format that does not lend itself to the processing that yields information. **Structured data** are the result of taking unstructured data and formatting (structuring) such data to facilitate storage, use, and the generation of information. You apply structure (format) based on the type of processing that you intend to perform on the data. Some data might be not ready (unstructured) for some types of processing, but they might be ready (structured) for other types of processing. For example, the data value 37890 might refer to a zip code, a sales value, or a product code. If this value represents a zip code or a product code and is stored as text, you cannot perform mathematical computations with it. On the other hand, if this value represents a sales transaction, it is necessary to format it as numeric.

Structured data are more prevalent than unstructured data in a business environment. For example, if invoices are stored as images for future retrieval and display, you can scan them and save them in a graphic format. On the other hand, if you want to derive information such as monthly totals and average sales, such graphic storage would not be useful. Instead, you could store the invoice data in a (structured) spreadsheet format so that you can perform the requisite computations.

14. What are some basic database functions that a spreadsheet cannot perform.
Spreadsheets do not support self-documentation through metadata, enforcement of data types or domains to ensure consistency of data within a column, defined relationships among tables, or constraints to ensure consistency of data across related tables.

15. What common problems do a collection of spreadsheets created by end users share with the typical file system?
A collection of spreadsheets shares several problems with the typical file system. First problem is that end users create their own, private, copies of the data, which creates issues of data ownership. This situation also creates islands of information where changes to one set of data are not reflected in all of the copies of the data. This leads to the second problem – lack of data consistency. Because the data in various spreadsheets may be intended to represent a view of the business environment, a lack of consistency in the data may lead to faulty decision making based on inaccurate data.

16. Explain the significance of the loss of direct, hands-on access to business data that users experienced with the advent of computerized data repositories.
Users lost direct, hands-on access to the business data when computerized data repositories were developed because the IT skills necessary to directly access and manipulate the data were beyond the average user's abilities, and because security precautions restricted access to the shared data. This was significant because it removed users from the direct manipulation of data and introduced significant time delays for data access. When users need answers to business questions from the data, necessity often does not give them the luxury of time to wait days, weeks, or even months for the required reports. The desire to return hands-on access to the data to the users, among other drivers, helped to propel the development of database systems. While database systems have greatly improved the ability of users to

directly access data, the need to quickly manipulate data for themselves has lead to the problems of spreadsheets being used when databases are needed.

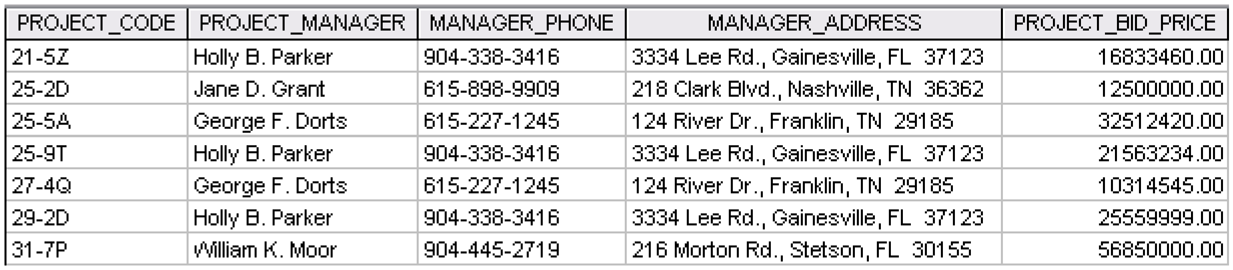
Problem Solutions

ONLINE CONTENT

The file structures you see in this problem set are simulated in a Microsoft Access database named Ch01_Problems, available in the Premium Website for this book. The Premium Website also includes SQL script files (Oracle and SQLServer) for all of the data sets used throughout the book.

Given the file structure shown in Figure P1.1, answer Problems 1 - 4.

FIGURE P1.1 The File Structure for Problems 1-4



PROJECT_CODE	PROJECT_MANAGER	MANAGER_PHONE	MANAGER_ADDRESS	PROJECT_BID_PRICE
21-5Z	Holly B. Parker	904-338-3416	3334 Lee Rd., Gainesville, FL 37123	16833460.00
25-2D	Jane D. Grant	615-898-9909	218 Clark Blvd., Nashville, TN 36362	12500000.00
25-5A	George F. Dorts	615-227-1245	124 River Dr., Franklin, TN 29185	32512420.00
25-9T	Holly B. Parker	904-338-3416	3334 Lee Rd., Gainesville, FL 37123	21563234.00
27-4Q	George F. Dorts	615-227-1245	124 River Dr., Franklin, TN 29185	10314545.00
29-2D	Holly B. Parker	904-338-3416	3334 Lee Rd., Gainesville, FL 37123	25559999.00
31-7P	William K. Moor	904-445-2719	216 Morton Rd., Stetson, FL 30155	56850000.00

1. **How many records does the file contain? How many fields are there per record?**

   The file contains seven records (21-5Z through 31-7P) and each of the records is composed of five fields (PROJECT_CODE through PROJECT_BID_PRICE.)

2. **What problem would you encounter if you wanted to produce a listing by city? How would you solve this problem by altering the file structure?**

   The city names are contained within the MANAGER_ADDRESS attribute and decomposing this character (string) field at the application level is cumbersome at best. (Queries become much more difficult to write and take longer to execute when internal string searches must be conducted.) If the ability to produce city listings is important, it is best to store the city name as a separate attribute.

3. **If you wanted to produce a listing of the file contents by last name, area code, city, state, or zip code, how would you alter the file structure?**

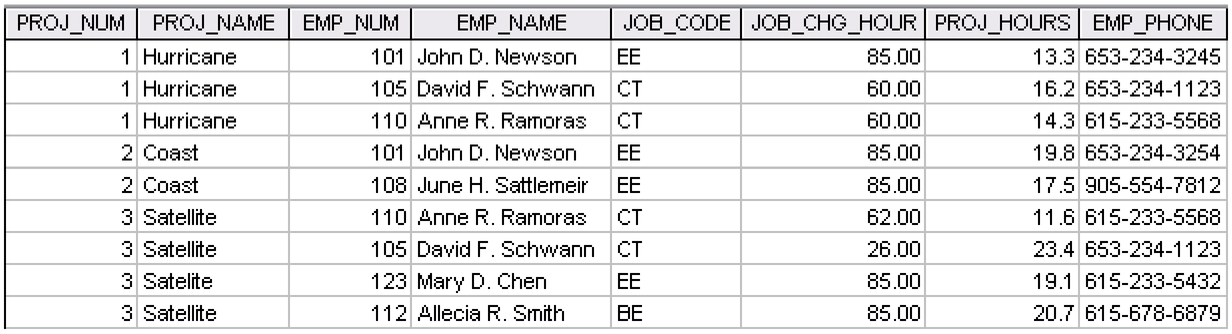
   The more we divide the address into its component parts, the greater its information capabilities. For example, by dividing MANAGER_ADDRESS into its component parts (MGR_STREET, MGR_CITY, MGR_STATE, and MGR_ZIP), we gain the ability to easily select records on the basis of zip codes, city names, and states. Similarly, by subdividing the MANAGER name into its components MGR_LASTNAME, MGR_FIRSTNAME, and MGR_INITIAL, we gain the ability to produce more efficient searches and listings. For example, creating a phone directory is easy when you can sort by last name, first name, and initial. Finally, separating the area code and the phone number will yield the ability to efficiently group data by area codes. Thus MGR_PHONE might be decomposed into MGR_AREA_CODE and MGR_PHONE. The more you decompose the data into their component parts, the greater the search flexibility. Data that are decomposed into their most basic components are said to be ***atomic***.

4. **What data redundancies do you detect? How could those redundancies lead to anomalies?**

   Note that the manager named Holly B. Parker occurs three times, indicating that she manages three projects coded 21-5Z, 25-9T, and 29-2D, respectively. (The occurrences indicate that there is a 1:M relationship between PROJECT and MANAGER: each project is managed by only one manager but, apparently, a manager may manage more than one project.) Ms. Parker's phone number and address also occur three times. If Ms. Parker moves and/or changes her phone number, these changes must be made more than once *and they must all be made correctly... without missing a single occurrence*. If any occurrence is missed during the change, the data are "different" for the same person. After some time, it may become difficult to determine what the correct data are. In addition, multiple occurrences invite misspellings and digit transpositions, thus producing the same anomalies. The same problems exist for the multiple occurrences of George F. Dorts.

5. **Identify and discuss the serious data redundancy problems exhibited by the file structure shown in Figure P1.5.**

FIGURE P1.5 The File Structure for Problems 5-8



PROJ_NUM	PROJ_NAME	EMP_NUM	EMP_NAME	JOB_CODE	JOB_CHG_HOUR	PROJ_HOURS	EMP_PHONE
1	Hurricane	101	John D. Newson	EE	85.00	13.3	653-234-3245
1	Hurricane	105	David F. Schwann	CT	60.00	16.2	653-234-1123
1	Hurricane	110	Anne R. Ramoras	CT	60.00	14.3	615-233-5568
2	Coast	101	John D. Newson	EE	85.00	19.8	653-234-3254
2	Coast	108	June H. Sattlemeir	EE	85.00	17.5	905-554-7812
3	Satellite	110	Anne R. Ramoras	CT	62.00	11.6	615-233-5568
3	Satellite	105	David F. Schwann	CT	26.00	23.4	653-234-1123
3	Satelite	123	Mary D. Chen	EE	85.00	19.1	615-233-5432
3	Satellite	112	Allecia R. Smith	BE	85.00	20.7	615-678-6879

NOTE

It is not too early to begin discussing proper structure. For example, you may focus student attention on the fact that, ideally, each row should represent a single entity. Therefore, each row's fields should define the characteristics of one entity, rather than include characteristics of several entities. The file structure shown here includes characteristics of multiple entities. For example, the JOB_CODE is likely to be a characteristic of a JOB entity. PROJ_NUM and PROJ_NAME are clearly characteristics of a PROJECT entity. Also, since (apparently) each project has more than one employee assigned to it, the file structure shown here shows multiple occurrences for each of the projects. (Hurricane occurs three times, Coast occurs twice, and Satellite occurs four times.)

Given the file's poor structure, the stage is set for multiple anomalies. For example, if the charge for JOB_CODE = EE changes from $85.00 to $90.00, that change must be made twice. Also, if employee June H. Sattlemeier is deleted from the file, you also lose information about the existence of her JOB_CODE = EE, its hourly charge of $85.00, and the PROJ_HOURS = 17.5. The loss of the PROJ_HOURS value will ultimately mean that the Coast project costs are not being charged properly, thus causing a loss of PROJ_HOURS*JOB_CHG_HOUR = 17.5 x $85.00 = $1,487.50 to the company.

Incidentally, note that the file contains different JOB_CHG_HOUR values for the same CT job code, thus illustrating the effect of changes in the hourly charge rate over time. The file structure appears to represent transactions that charge project hours to each project. However, the structure of this file makes it difficult to avoid update anomalies and it is not possible to determine whether a charge change is *accurately* reflected in each record. Ideally, a change in the hourly charge rate would be made in only one place and this change would then be passed on to the transaction based on the hourly charge. Such a structural change would ensure the historical accuracy of the transactions.

You might want to emphasize that the recommended changes require a lot of work in a file system.

6. Looking at the EMP_NAME and EMP_PHONE contents in Figure P1.5, what change(s) would you recommend?

A good recommendation would be to make the data more *atomic*. That is, break up the data componnts whenever possible. For example, separate the EMP_NAME into its componenst EMP_FNAME, EMP_INITIAL, and EMP_LNAME. This change will make it much easier to organize employee data through the employee name component. Similarly, the EMP_PHONE data should be decomposed into EMP_AREACODE and EMP_PHONE. For example, breaking up the phone number 653-234-3245 into the area code 653 and the phone number 234-3245 will make it much easier to organize the phone numbers by area code. (If you want to print an employee phone directory, the more atomic employee name data will make the job much easier.)

7. **Identify the various data sources in the file you examined in Problem 5.**

Given their answers to problem 5 and some additional scrutiny of Figure 1.5, your students should be able to identify these data sources:

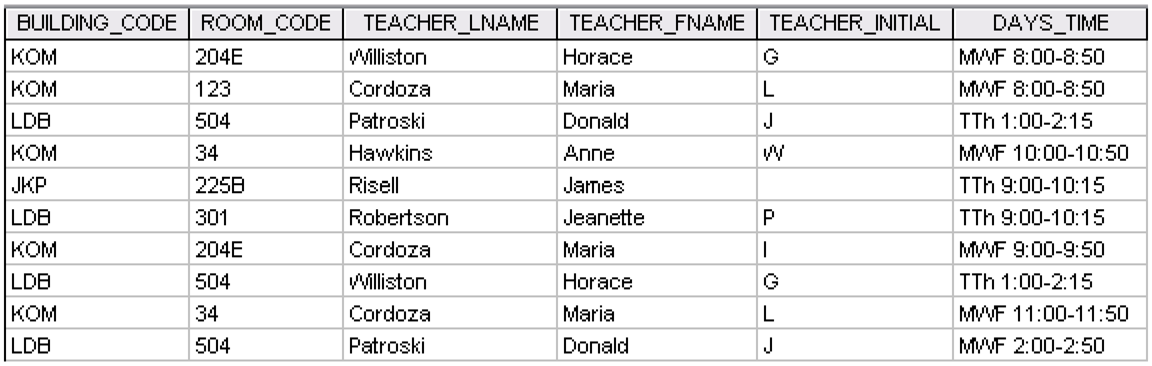
- Employee data such as names and phone numbers.
- Project data such as project names. If you start with an EMPLOYEE file, the project names clearly do not belong in that file. (Project names are clearly *not* employee characteristics.)
- Job data such as the job charge per hour. If you start with an EMPLOYEE file, the job charge per hour clearly does not belong in that file. (Hourly charges are clearly *not* employee characteristics.)
- The project hours, which are most likely the hours worked by the employee for that project. (Such hours are associated with a work product, not the employee per se.)

8. **Given your answer to Problem 7, what new files should you create to help eliminate the data redundancies found in the file shown in Figure P1.5?**

The data sources are probably the PROJECT, EMPLOYEE, JOB, and CHARGE. The PROJECT file should contain project characteristics such as the project name, the project manager/coordinator, the project budget, and so on. The EMPLOYEE file might contain the employee names, phone number, address, and so on. The JOB file would contain the billing charge per hour for each of the job types – a database designer, an applications developer, and an accountant would generate different billing charges per hour. The CHARGE file would be used to keep track of the number of hours by job type that will be billed for each employee who worked on the project.

9. **Identify and discuss the serious data redundancy problems exhibited by the file structure shown in Figure P1.9. (The file is meant to be used as a teacher class assignment schedule. One of the many problems with data redundancy is the likely occurrence of data inconsistencies – that two different initials have been entered for the teacher named Maria Cordoza.)**

FIGURE P1.9 The File Structure for Problems 9-10



BUILDING_CODE	ROOM_CODE	TEACHER_LNAME	TEACHER_FNAME	TEACHER_INITIAL	DAYS_TIME
KOM	204E	Williston	Horace	G	MWF 8:00-8:50
KOM	123	Cordoza	Maria	L	MWF 8:00-8:50
LDB	504	Patroski	Donald	J	TTh 1:00-2:15
KOM	34	Hawkins	Anne	W	MWF 10:00-10:50
JKP	225B	Risell	James		TTh 9:00-10:15
LDB	301	Robertson	Jeanette	P	TTh 9:00-10:15
KOM	204E	Cordoza	Maria	I	MWF 9:00-9:50
LDB	504	Williston	Horace	G	TTh 1:00-2:15
KOM	34	Cordoza	Maria	L	MWF 11:00-11:50
LDB	504	Patroski	Donald	J	MWF 2:00-2:50

Note that the teacher characteristics occur multiple times in this file. For example, the teacher named Maria Cordoza's first name, last name, and initial occur three times. If changes must be made for any given teacher, those changes must be made multiple times. All it takes is one incorrect entry or one forgotten change to create data inconsistencies. Redundant data are not a luxury you can afford in a data environment.

10. Given the file structure shown in Figure P1.9, what problem(s) might you encounter if building KOM were deleted?

You would lose all the time assignment data about teachers Williston, Cordoza, and Hawkins, as well as the KOM rooms 204E, 123, and 34. Here is yet another good reason for keeping data about specific entities in their own tables! This kind of an anomaly is known as a *deletion anomaly*.