**1.** **Define 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.

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

Data independence is a condition in which the programs that access data are not dependent on the data storage characteristics of the data. Systems that lack data independence are said to exhibit data dependence. 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.

Dataindependence 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.

**10. What is metadata?**

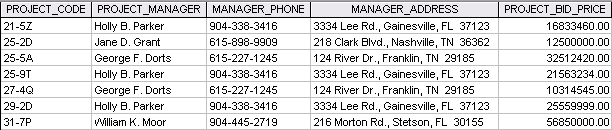
Metadata is data about data. That is, metadata define the data characteristics such as the data type (such as character or numeric) *and the relationships that link the data*. Relationships are an important component of database design. What makes relationships especially interesting is that they are 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 2, Data Models.”. Relationships will play a key role in subsequent chapters. You cannot effectively deal with database design issues unless you address relationships.

**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.

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



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.)

1. **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.

1. **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?**

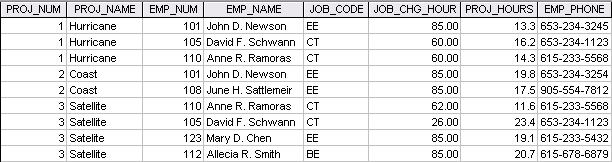
The more we divide the address into its component parts, the greater its information capabilities. For example, by dividing MANAGER\_ADDRESS into its component parts (MGR\_STREET, MGR\_CITY, MGR\_STATE, and MGR\_ZIP), we gain the ability to easily select records on the basis of zip codes, city names, 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***.

1. **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.

1. **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**



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.

1. **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:

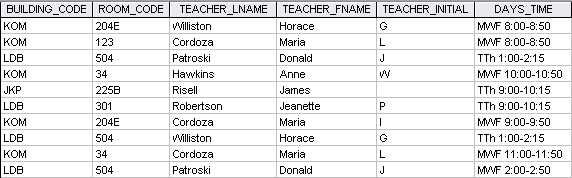
* Employee data such as names and phone numbers.
* Project data such as project names. If you start with an EMPLOYEE file, the project names clearly do not belong in that file. (Project names are clearly *not* employee characteristics.)
* Job data such as the job charge per hour. If you start with an 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**



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.

1. **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*.

**Problem #1 – Retrieving all rows and all columns from a table**

For each of our tables, retrieve all rows and all columns.

Tables are Student, Faculty, Offering, Course, and Enrollment

(no need to sort at this point)

SELECT s.\*

FROM Student s

SELECT f.\*

FROM Faculty f

SELECT o.\*

FROM Offering o

SELECT c.\*

FROM Course c

SELECT e.\*

FROM Enrollment e

**Problem #2 – Retrieving a subset of columns from a table and sorting them**

**both with and without the ASC keyword**

Retrieve the student number, student first name, and student last name for all students

Sort the results by student last name then by student first name

Use the ASC keyword on the query

Repeat the query omitting ASC

SELECT s.StdNo, s.StdFirstName, s.StdLastName

FROM Student s

ORDER BY s.StdLastName ASC, s.StdFirstName ASC

SELECT s.StdNo, s.StdFirstName, s.StdLastName

FROM Student s

ORDER BY s.StdLastName, s.StdFirstName

**Problem #3 – Retrieving a subset of columns from a table and sorting them on multiple columns mixing ascending and descending order, using both named and positional notation**

Retrieve the student last name, student first name, and GPA for all students

Sort the results by GPA highest first, then by student last name, then by student first name

Use column names to sort (omit ASC)

Repeat the query using positional notation

SELECT s.StdLastName, s.StdFirstName, s.StdGPA

FROM Student s

ORDER BY s.StdGPA DESC, s.StdLastName, s.StdFirstName

SELECT s.StdLastName, s.StdFirstName, s.StdGPA

FROM Student s

ORDER BY 3 DESC, 1, 2

**Problem #4 – Retrieving columns from a table both with and without duplicates**

Retrieve the student city and class for all students with duplicates

Repeat query without duplicates

SELECT s.StdCity, s.StdClass

FROM Student s

ORDER BY 1,2

SELECT DISTINCT s.StdCity, s.StdClass

FROM Student s

ORDER BY 1,2

**Problem #5 – Retrieving a subset of rows with a single Boolean expression**

Retrieve the student last name, student first name, and GPA for all students with a GPA greater than 3.2

SELECT s.StdLastName, s.StdFirstName, s.StdGPA

FROM Student s

WHERE s.StdGPA > 3.2

ORDER BY 3DESC, 1, 2