(c) aggregation
(d) specialization
(e) generalization

2.2 Choose from the following list an organization you are most familiar with: college or university, public library, hospital, fast-food restaurant, department store. Determine, as in Exercise 1.9, the entities of interest and the relationships that exist between these entities. Draw the E-R diagram for the organization. Construct a tabular representation of the entities and relationships.

2.3 Are weak entities necessary? What is the distinction between a weak entity and a strong one? Can a weak entity be converted to a strong entity?

2.4 Using the EMPLOYEE entity of Figure 2.6, convert each of the one-to-many associations into a weak entity and a relationship. Identify the discriminator of each weak entity and the attributes of each relationship.

2.5 Convert the E-R diagram that you prepared for Exercise 2.2 into a network database model. List the record types and the set types in your model. Indicate for each set type the owner and member record types.

2.6 Convert the E-R diagram that you prepared for Exercise 2.2 into a hierarchical database model. List the record types and the hierarchy. Indicate how you can handle the situation where a record type occurs in more than one hierarchy or occurs more than once in the same hierarchy.

2.7 Explain the distinction between the representation of association and relationship in the network and hierarchical models.

2.8 The People's Bank offer five types of accounts: loan, checking, premium savings, daily interest saving, and money market. It operates a number of branches and a client of the bank can have any number of accounts. Accounts can be joint, i.e., more than one client may be able to operate a given account. Identify the entities of interest and show their attributes. What relationships exist among these entities? Draw the corresponding E-R diagram.

2.9 Give a sample of each of the tables that would be required for the E-R diagram of Exercise 2.8.

2.10 Complete the network sets and the hierarchical trees for the portion of the data for the Universal Hockey League given in the tables of Figure 2.35. Comment on the relative merits of the three models from the point of view of data duplication and ease of retrieval.

2.11 Suppose that in the database design for the UHL of Section 2.5, we wished to maintain the career statistics for each player. (The total goals and assists over the lifetime—career—of a player are to be maintained in addition to the season statistics.) Draw the modified E-R diagram and give the corresponding database design using the relational, network, or hierarchical model.

2.12 In each of the database designs given in Section 2.8, how would you find out if a certain player played as a forward or as a goalie? Introduce two *IS_A* relationships between players and entities FORWARD_POSITION and GOAL_POSITION and draw an E-R diagram for a database application that requires keeping the player's career statistics as well as the statistics indicated in the text.

2.13 Explain why navigation is simpler in the relational data model than in the hierarchical data model.

Bibliographic Notes

Senko (Senk 77), in a survey article, gave details of some of the models discussed in this chapter. The entity-relationship data model (Chen 76) grew out of the exercise of using commercially available DBMS to model application databases. Recently the E-R model has been enriched and used in conceptual view design (Buss 83).

The DBTG proposal was the first data model to be formalized. The first report (which has been revised a number of times), issued by the Database Task Group of the Conference on Data System Languages (DBTG/CODASYL) (CODA 71) contained detailed specifications for the network data model (a model conforming to these specifications is also known as the DBTG data model). The specifications contained in the first report and subsequent revisions have been subjected to much debate and criticism (Tayl 76). Many of the current database applications have been built on commercial DBMSs using this approach.

Bachman (Bach 69) introduced a graphical means called the data structure diagram to denote the logical relationship implied by the DBTG set. The data structure diagrams have been extended to include field names in the record type rectangle and the arrow is used to clearly identify the data fields involved in the set association (Brad 78).

In the network model, the relationships are predefined at database creation time; however, dynamic relationships as in the relational model have been proposed (Brad 78).

The hierarchical model has been widely used in many existing database systems since the late 1960s and early 1970s, due to the promotion of the IMS system by IBM (IBM 75) and SYSTEM 2000 by MRI Systems Corporation (MRI 74).

The relational data model (Codd 70) is a model for representing the association between the attributes of an entity and the association between different entities using the relation as a construct. One of the main reasons for the introduction of this model was to increase the productivity of the application programmer (Codd 82).

The relational model had its roots in the binary relations for data storage, namely the relational data file of Levien and Maron (Levi 67) and the TRAMP system of Ash and Sibley (Ash 68). The generalization of the binary relation to an n-ary relation was proposed by Codd (Codd 70). He gave a definition of the n-ary relation for use in large shared data banks and outlined the advantages of this approach.

Codd's paper was instrumental in setting the direction of research in relational database systems. After more than a decade of development and trials, relational data management systems are on the market. Examples of these are SYSTEM R, DB2, SQL/DS, ORACLE, INGRES, RAPPORT, QBE, and Knowledgeman.

The universal relational model aims at relieving the user of providing even the logical navigation through the database (Maie 84). Another concept missing from the relational model was that of specifying constraints between some relations; this problem has been addressed in (Codd 79).

Textbook-level discussions of data models can be found in (Tsic 82), (Kort 86), (Maie 83), (Ullm 82), (Date 86) and (Brod 84a).

Bibliography

(ANSI 75) ANSI/X3/SPARC Study Group on Database Management Systems, Interim Report, FDT (ACM SIGMOD bulletin), vol. 7(2), 1975.

(Ash 68) W. L. Ash & E. H. Sibley, "TRAMP: A Relational Memory with Deductive Capabilities," Proc. ACM 23rd National Conf. August 1968. Princeton, N.K.: Brandon Systems Press, 1968, pp. 143–156.

(Bach 69) C. W. Bachman, "Data Structure Diagrams," *Data Base* (ACM) 1(2), 1969, pp. 4–10.

(Brad 78) J. Bradley, "An Extended Owner-Coupled Set Data Model in Predicate Calculus for Database Management," *ACM TODS* 3(4), 1978, pp. 385–416.

(Brod 84a) M. J. Brodie, J. Mylopoulos, & J. W. Schmidt, *On Conceptual Modelling*, New York: Springer-Verlag, 1984.

(Brod 84b) M. J. Brodie, "On the Development of Data Models," in (Brod 84a), pp. 19–47.

(Buss 83) U. Bussolati, S. Ceri, V. De Antonellis, & B. Zonta, "Views: Conceptual Design," in *Methodology and Tools for Data Base Design* ed. S. Ceri, Amsterdam: North Holland, 1983, pp. 25–55.

(Cahe 83) R. G. G. Cahell, "Design and Implementation of a Relational-Entity-Datum Data Model," Technical Report CSL-83-4, XEROX PARC, Palo Alto, CA: May 1983.

(Chen 76) P. P. Chen, "The ER Model Toward a Unified View of Data,' *ACM TODS* 1(1), 1976, pp. 9–36.

(Chen 80) P. P. Chen, ed., "Entity-Relationship Approach to System Analysis and Design," North Holland, Mass., 1980.

(CODA 71) CODASYL Database Task Group Report, April 1971, ACM, New York, 1971.

(Codd 70) E. F. Codd, "A Relational Mode of Data for Large Shared Data Banks,' *CACM*, 13(2), 1970, pp. 377–387.

(Codd 79) E. F. Codd, "Extending the Database Relation Model to Capture More Meaning," *ACM TODS* 4, 1979, pp. 392–434.

(Codd 82) E. F. Codd, "Relational Database: A Practical Foundation For Productivity," The 1981 ACM Turing Award Lecture, in CACM, 25(2), 1982, pp. 109–117.

(Date 86) C. J. Date, *An Introduction to Database Systems*, vol. 1, 4th ed. Reading, MA: Addison-Wesley, 1986.

(Feld 69) J. A. Feldman & P. D. Rovner, "An Algol-Based Associative Language," *CACM* 12(8), August 1969, pp. 439–447.

(Find 79) N. V. Findler, ed., *Associative Networks: Representation and Use of Knowledge by Computer,"* New York: Academic Press, 1979.

(Grif 82) R. L. Griffith, "Three Principles of Representation for Semantic Networks," *ACM TODS* 7(3), 1982, pp. 417–442.

(Hamm 81) M. Hammer & D. McLeod, "Database Description with SDM: A Semantic Database Model," *ACM TODS* 6(3), 1981, pp. 351–386.

(IBM 75) Information Management System Publications, GH70-1260, White Plains, NY: IBM, 1975.

(Jard 77) D. A. Jardine, ed, "The ANSI/SPARC DBMS Model," Proceedings of the Second SHARE Working Conference on Database Manage ›nt Systems, Montreal, Canada, 1976. Amsterdam: North-Holland, 1977.

(Kers 67) L. Kerschberg, A. Klug, & D. C. Tsichritzis, "A Taxonomy of Data Models," in *Systems for Large Databases*, ed. (P. C. Lockermann & E. J. Neuhold.) Amsterdam: North-Holland, 1967, pp. 43–64.

(Knut 68) D. E. Knuth, "The Art of Computer Programming," vol. 1. Reading, MA: Addison-Wesley, 1968.

(Kort 86) H. F. Korth & A. Silberschatz, *Database System Concepts*, New York: McGraw-Hill, 1986.

(Levi 67) R. E. Levien M. E. Maron, "A Computer System for Inference Execution and Data Retrieval," *CACM* 10(11), Nov. 1967, pp. 715–721.

(Maie 83) D. Maier, *The Theory of Relational Databases*, Rockville, MD: Computer Science Press, 1983.

(Maie 84) D. Maier, J. D. Ullman, & M. Y. Vardi, "On the Foundation of the Universal Relation Model," *ACM TODS* 9(2), 1984, pp. 283–308.

(MR 174) Systems 2000 Publications, A-1, C-1, F-1, G-1, I-1, P-1, R-1. Austin, TX: MRI Systems Corp., 1974.

(Mylo 80) J. Mylopoulos, P. A. Bernstein, & H. K. T. Eong, "A Language Facility for Database Intensive Applications," *ACM TODS* 5(2), 1980, pp. 185–207.

(Senk 77) M. E. Senko, "Data Structures and Data Accessing in Database Systems, Past, Present and Future," *IBM Systems Journal*, vol. 16, 1977, pp. 16, 208–257.

(Smit 77) J. M. Smith, D. C. P. Smith, "Database Abstractions: Aggregation and Generalization," *ACM TODS* 2(2), 1977, pp. 105–133.

(Su 79) S. Y. W. Su & D. H. Lo, "A Semantic Association Model for Conceptual Database Design," Proc. of Int. Conf. on Entity-Relationship Approach to System Analysis and Design, Los Angeles, CA, December 1979, pp. 147–171.

(Tayl 76), R. W. Taylor, R. L. Frank, "CODASYL Database Management System," *ACM Computing Surveys* 8(1), 1976, pp. 67–104.

(Teic 77) D. Teichroew & E. A. Hershey III, "PSL/PSA: A Computer-Aided Technique for Structured Documentation and Analysis of Information Processing System," *IEEE Trans. on Software Eng.* 3(1), 1977, pp. 41–48.

(Teor 86) T. J. Teory, D. Yang, & J. P. Fry, "A Logical Design Methodology for Relational Databases Using the Extended Entity-Relationship Model," *ACM Computing Survey* 18(2), June 1986, pp. 197–222.

(Tsic 78) D. C. Tsichritzis & A. Klug, eds, "The ANSI/X3/SPARC DBMS Framework Report of the Study Group on Database Management Systems," *Information Systems* 3, 1978, pp. 173–191.

(Tsic 82) D. C. Tsichritzis & F. H. Lochovsky, *Data Models*, Englewood Cliffs, NJ: Prentice-Hall, 1982.

(Ullm 82) J. D. Ullman, *Principles of Database Systems*, Rockville, MD: Computer Science Press, 1982.

Chapter

3

File Organization

Contents

In this chapter we focus on a number of methods used to organize files and the issues involved in the choice of a method. File organization deals with the structure of data in secondary storage devices. In designing the structure the designer is concerned with the access time involved in the retrieval of records based on primary or secondary keys, as well as the techniques involved in updating data. We discuss the following file organization schemes: sequential, index sequential, multilist, direct, extendable hashing, and tree structured. The general principles involved in these schemes are presented, although we do not discuss the implementation issues under a specific operating system.

3.1 Introduction

Just as lists, arrays, trees, and other data structures are used to implement data organization in main memory, a number of strategies are used to support the organization of data in secondary memory. We can expect, as in main memory data organization, that there is no universal secondary data organization strategy suitable under all usage conditions. As discussed earlier, certain attribute (or field) values can uniquely identify a record, i.e., these attributes makeup the primary key of the record. Other attribute values identify not one but a set of records. These attributes are called secondary or nonprimary keys. In this chapter we consider both primary key and nonprimary key retrieval and updates, bearing in mind that there are space/time trade-offs for all structures.

Traditionally the term **file** has been used to refer to the folder that holds related material in ordered storage for quick reference. We use the same word, file, to describe the object as well as its contents. The order of the file is an arrangement of its contents according to one's expected needs for future reference. For example, if we have a file of birth dates of persons we know, we may wish to arrange them by date. We could also arrange them alphabetically by family or first name. The choice of arrangement depends on the reason for the file. If we wish to consult the file periodically to discover upcoming birthdays, chronological order would be chosen. If, however, we wish to know the date of Bill's birthday; we would opt for the alphabetical ordering on first names. What are we to do when we have both types of requirements? We could, for example, maintain a copy of the file in chronological order and another in lexical order. In this case, the contents would be the same but the order would be different. We would rarely remove (or delete) a person's birth date from the file; rather, we would add new names and dates to the file. We may need to change someone's name. In all of these cases both copies of the file would be changed. It is impossible to change both files at the same instance, i.e., we first alter one copy and then the other. Can we, while the changes are being made, make use of either file? Imagine what would happen when a number of copies and a large number of users exist. The method of creating a copy for each application is replete with problems. A possible solution is to maintain the file in some physical order and allow access in some other order, i.e., the logical access order is different from the physical access order. This concept is very important because the same file could then be used to support different access orders.

To further classify the contents, a file should be labeled. We can label the file described above as a file of Birth_Dates. Similarly, we can create suitably named files for other things such as Recipes, Bills, and so on. We could keep all these files in a box. The box, by definition, is also a file—it is a file of files. We could treat the secondary storage medium as this box (a file of files). In this chapter we look at techniques for managing files. The same techniques are applicable to the file of files.

3.1.1 Storage Device Characteristics

Presently, the common secondary storage media used to store data are disk and tape. Tape is generally used for archival data. The storage medium used in a disk is a **disk pack.** A disk pack, shown in Figure 3.1, is made up of a number of surfaces. Data is read and written from the disk pack by means of transducers called **read/write heads.** The number of read/write heads depends on the type of disk drive. If we trace the projection of one head on the surface associated with it as the disk rotates, we would create a circular figure called a **track.** The tracks at the same position on every surface of the disk form the surface of an imaginary cylinder. In disk terminology, therefore, a **cylinder** consists of the tracks under the heads on each of its surfaces.

In one type of disk drive each track on each surface has a dedicated stationary head, which does not move. Such a disk drive is called a **fixed head drive.** The other type of disk drive is a **moving head drive,** wherein a single head is used for each surface. When data from a given track is to be read, the head is moved to the track. Figure 3.2 shows the cross section of a fixed head drive and Figure 3.3 shows that of a moving head drive.

The disk stores data along concentric tracks. It takes some time for the read/write head of a moving head disk drive to move from track (or cylinder) i to track (or cylinder) j. This is called the **seek time.** (For a fixed head disk, the seek time is 0.) In the case of a moving head drive, the seek time depends on the distance between the current head and the target head positions. Typical values are from 10 to 50 msec (msec = 1/1000 sec). If a file consists of c consecutive cylinders and we assume uniform and random distribution of requests for the different cylinders, we can show that the average distance (in cylinders) the head moves is c/3 (proof for this is given in Appendix 3.2 at the end of the text). Before data can be read or written the disk has to rotate so that the head is positioned at some point relative to

Figure 3.1 Structure of a disk pack with read/write heads.

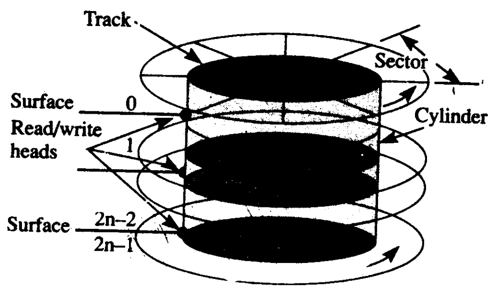
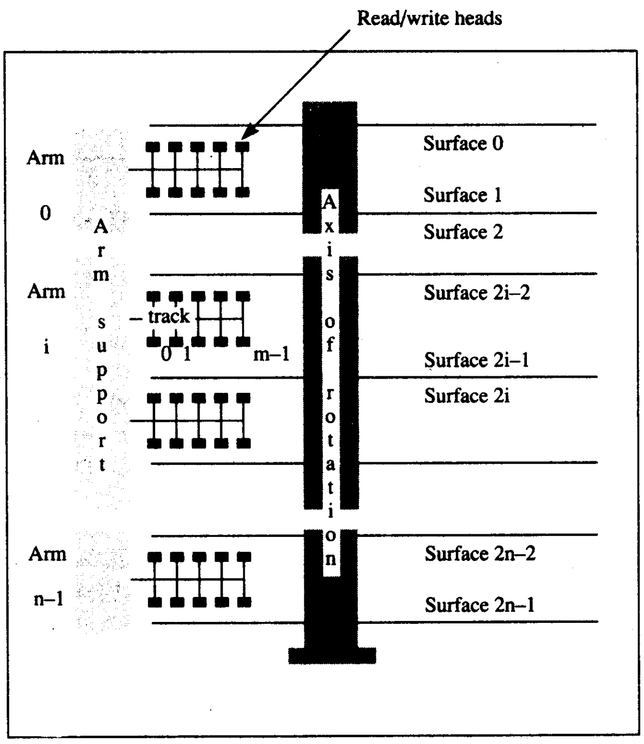


Figure 3.2 Fixed head disk with read/write head per track.



a marked start point. The time needed for the reading or writing to start depends on the rotational delay. On the average, the rotational delay is half the rotation time, that is, the time for the disk to rotate once. The rotational delay is called **latency time.** For a drive that rotates at 3600 revolutions per minute, the average latency time is 8.33 msec. The **access time,** therefore, depends on the seek time and the latency time.

On magnetic tapes, data blocks are separated by **interblock gaps (IBG).** The IBG can be attributed to the deceleration/ acceleration (stop/start) that takes place between successive block reads. This only happens when, after a single access, time is needed to process the data before a second read. When continuously scanning over the data, there is no need to stop/start after reading each block. The IBG is also scanned at the faster rate. The typical value for the IBG is 0.6 inch. The access time, i.e., the time required to locate the target block on a magnetic tape, depends on the distance between the current and target blocks.

As we see from the above, the access time depends on the distance between the current and target positions for both types of storage devices. This time can be optimized by suitably placing records. It can also be affected by the file organization employed.

We can abstract the disk storage medium in terms of a two-dimensional array and the tape as a one-dimensional array of data blocks (see Figure 3.4). Note that in both cases we can specify a unique address for a block (or physical record). We will