Chapter 1

The Worlds of Database Systems

Databases today are essential to every business. Whenever you visit a major Web site — Google, Yahoo!, Amazon.com, or thousands of smaller sites that provide information — there is a database behind the scenes serving up the information you request. Corporations maintain all their important records in databases. Databases are likewise found at the core of many scientific investigations. They represent the data gathered by astronomers, by investigators of the human genome, and by biochemists exploring properties of proteins, among many other scientific activities.

The power of databases comes from a body of knowledge and technology that has developed over several decades and is embodied in specialized software called a ***database management system***, or *DBMS*, or more colloquially a "database system." A DBMS is a powerful tool for creating and managing large amounts of data efficiently and allowing it to persist over long periods of time, safely. These systems are among the most complex types of software available. In this book, we shall learn how to design databases, how to write programs in the various languages associated with a DBMS, and how to implement the DBMS itself.

1.1 The Evolution of Database Systems

What is a database? In essence a database is nothing more than a collection of information that exists over a long period of time, often many years. In common parlance, the term ***database*** refers to a collection of data that is managed by a DBMS. The DBMS is expected to:

1. Allow users to create new databases and specify their *schemas* (logical structure of the data), using a specialized *data-definition language.*

2. Give users the ability to *query* the data (a "query" is database lingo for a question about the data) and modify the data, using an appropriate language, often called a *query language* or *data-manipulation language.*

3. Support the storage of very large amounts of data — many terabytes or more — over a long period of time, allowing efficient access to the data for queries and database modifications.

4. Enable *durability,* the recovery of the database in the face of failures, errors of many kinds, or intentional misuse.

5. Control access to data from many users at once, without allowing unexpected interactions among users (called *isolation*) and without actions on the data to be performed partially but not completely (called *atomicity*).

1.1.1 Early Database Management Systems

The first commercial database management systems appeared in the late 1960's. These systems evolved from file systems, which provide some of item (3) above; file systems store data over a long period of time, and they allow the storage of large amounts of data. However, file systems do not generally guarantee that data cannot be lost if it is not backed up, and they don't support efficient access to data items whose location in a particular file is not known.

Further, file systems do not directly support item (2), a query language for the data in files. Their support for (1) — a schema for the data — is limited to the creation of directory structures for files. Item (4) is not always supported by file systems; you can lose data that has not been backed up. Finally, file systems do not satisfy (5). While they allow concurrent access to files by several users or processes, a file system generally will not prevent situations such as two users modifying the same file at about the same time, so the changes made by one user fail to appear in the file.

The first important applications of DBMS's were ones where data was composed of many small items, and many queries or modifications were made. Examples of these applications are:

1. Banking systems: maintaining accounts and making sure that system failures do not cause money to disappear.

2. Airline reservation systems: these, like banking systems, require assurance that data will not be lost, and they must accept very large volumes of small actions by customers.

3. Corporate record keeping: employment and tax records, inventories, sales records, and a great variety of other types of information, much of it critical.

The early DBMS's required the programmer to visualize data much as it was stored. These database systems used several different data models for describing the structure of the information in a database, chief among them the "hierarchical" or tree-based model and the graph-based "network" model. The latter was standardized in the late 1960's through a report of CODASYL (Committee on Data Systems and Languages).[1]

A problem with these early models and systems was that they did not support high-level query languages. For example, the CODASYL query language had statements that allowed the user to jump from data element to data element, through a graph of pointers among these elements. There was considerable effort needed to write such programs, even for very simple queries.

1.1.2 Relational Database Systems

Following a famous paper written by Ted Codd in 1970,[2] database systems changed significantly. Codd proposed that database systems should present the user with a view of data organized as tables called *relations*. Behind the scenes, there might be a complex data structure that allowed rapid response to a variety of queries. But, unlike the programmers for earlier database systems, the programmer of a relational system would not be concerned with the storage structure. Queries could be expressed in a very high-level language, which greatly increased the efficiency of database programmers. We shall cover the relational model of database systems throughout most of this book. SQL ("Structured Query Language"), the most important query language based on the relational model, is covered extensively.

By 1990, relational database systems were the norm. Yet the database field continues to evolve, and new issues and approaches to the management of data surface regularly. Object-oriented features have infilrated the relational model. Some of the largest databases are organized rather differently from those using relational methodology. In the balance of this section, we shall consider some of the modern trends in database systems.

1.1.3 Smaller and Smaller Systems

Originally, DBMS's were large, expensive software systems running on large computers. The size was necessary, because to store a gigabyte of data required a large computer system. Today, hundreds of gigabytes fit on a single disk, and it is quite feasible to run a DBMS on a personal computer. Thus, database systems based on the relational model have become available for even very small machines, and they are beginning to appear as a common tool for computer applications, much as spreadsheets and word processors did before them.

Another important trend is the use of documents, often tagged using XML (eXtensible Modeling Language). Large collections of small documents can

[1] *CODASYL Data Base Task Group April 1971 Report*, ACM, New York.

[2] Codd, E. F., "A relational model for large shared data banks," *Comm. ACM*, **13**:6, pp. 377–387, 1970.

serve as a database, and the methods of querying and manipulating them are different from those used in relational systems.

1.1.4 Bigger and Bigger Systems

On the other hand, a gigabyte is not that much data any more. Corporate databases routinely store terabytes (10^{12} bytes). Yet there are many databases that store petabytes (10^{15} bytes) of data and serve it all to users. Some important examples:

1. Google holds petabytes of data gleaned from its crawl of the Web. This data is not held in a traditional DBMS, but in specialized structures optimized for search-engine queries.

2. Satellites send down petabytes of information for storage in specialized systems.

3. A picture is actually worth way more than a thousand words. You can store 1000 words in five or six thousand bytes. Storing a picture typically takes much more space. Repositories such as Flickr store millions of pictures and support search of those pictures. Even a database like Amazon's has millions of pictures of products to serve.

4. And if still pictures consume space, movies consume much more. An hour of video requires at least a gigabyte. Sites such as YouTube hold hundreds of thousands, or millions, of movies and make them available easily.

5. Peer-to-peer file-sharing systems use large networks of conventional computers to store and distribute data of various kinds. Although each node in the network may only store a few hundred gigabytes, together the database they embody is enormous.

1.1.5 Information Integration

To a great extent, the old problem of building and maintaining databases has become one of *information integration*: joining the information contained in many related databases into a whole. For example, a large company has many divisions. Each division may have built its own database of products or employee records independently of other divisions. Perhaps some of these divisions used to be independent companies, which naturally had their own way of doing things. These divisions may use different DBMS's and different structures for information. They may use different terms to mean the same thing or the same term to mean different things. To make matters worse, the existence of legacy applications using each of these databases makes it almost impossible to scrap them, ever.

As a result, it has become necessary with increasing frequency to build structures on top of existing databases, with the goal of integrating the information distributed among them. One popular approach is the creation of *data warehouses*, where information from many legacy databases is copied periodically, with the appropriate translation, to a central database. Another approach is the implementation of a mediator, or "middleware," whose function is to support an integrated model of the data of the various databases, while translating between this model and the actual models used by each database.

1.2 Overview of a Database Management System

In Fig. 1.1 we see an outline of a complete DBMS. Single boxes represent system components, while double boxes represent in-memory data structures. The solid lines indicate control and data flow, while dashed lines indicate data flow only. Since the diagram is complicated, we shall consider the details in several stages. First, at the top, we suggest that there are two distinct sources of commands to the DBMS:

1. Conventional users and application programs that ask for data or modify data.

2. A *database administrator*: a person or persons responsible for the structure or *schema* of the database.

1.2.1 Data-Definition Language Commands

The second kind of command is the simpler to process, and we show its trail beginning at the upper right side of Fig. 1.1. For example, the database administrator, or *DBA*, for a university registrar's database might decide that there should be a table or relation with columns for a student, a course the student has taken, and a grade for that student in that course. The DBA might also decide that the only allowable grades are A, B, C, D, and F. This structure and constraint information is all part of the schema of the database. It is shown in Fig. 1.1 as entered by the DBA, who needs special authority to execute schema-altering commands, since these can have profound effects on the database. These schema-altering data-definition language (DDL) commands are parsed by a DDL processor and passed to the execution engine, which then goes through the index/file/record manager to alter the *metadata*, that is, the schema information for the database.

1.2.2 Overview of Query Processing

The great majority of interactions with the DBMS follow the path on the left side of Fig. 1.1. A user or an application program initiates some action, using the data-manipulation language (DML). This command does not affect the schema of the database, but may affect the content of the database (if the