1 Database Systems

Alice: *I thought this was a theory book.*
Vittorio: *Yes, but good theory needs the big picture.*
Sergio: *Besides, what will you tell your grandfather when he asks what you study?*
Riccardo: *You can't tell him that you're studying the fundamental implications of genericity in database queries.*

Computers are now used in almost all aspects of human activity. One of their main uses is to manage information, which in some cases involves simply holding data for future retrieval and in other cases serving as the backbone for managing the life cycle of complex financial or engineering processes. A large amount of data stored in a computer is called a *database*. The basic software that supports the management of this data is called a *database management system* (dbms). The dbms is typically accompanied by a large and evergrowing body of application software that accesses and modifies the stored information. The primary focus in this book is to present part of the theory underlying the design and use of these systems. This preliminary chapter briefly reviews the field of database systems to indicate the larger context that has led to this theory.

1.1 The Main Principles

Database systems can be viewed as mediators between human beings who want to use data and physical devices that hold it (see Fig. 1.1). Early database management was based on explicit usage of *file systems* and customized application software. Gradually, principles and mechanisms were developed that insulated database users from the details of the physical implementation. In the late 1960s, the first major step in this direction was the development of *three-level architecture*. This architecture separated database functionalities into physical, logical, and external levels. (See Fig. 1.2. The three views represent various ways of looking at the database: multirelations, universal relation interface, and graphical interface.)

The separation of the logical definition of data from its physical implementation is central to the field of databases. One of the major research directions in the field has been the development and study of abstract, human-oriented models and interfaces for specifying the structure of stored data and for manipulating it. These models permit the user to concentrate on a logical representation of data that resembles his or her vision of the reality modeled by the data much more closely than the physical representation.

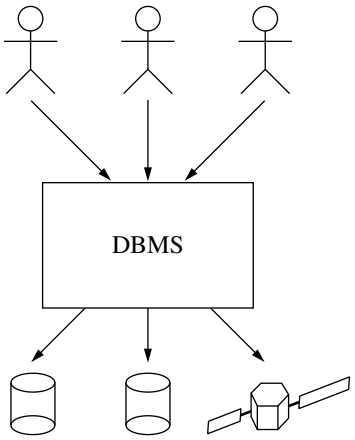

Figure 1.1: Database as mediator between humans and data

Several logical *data models* have been developed, including the hierarchical, network, relational, and object oriented. These include primarily a *data definition language* (DDL) for specifying the structural aspects of the data and a *data manipulation language* (DML) for accessing and updating it. The separation of the logical from the physical has resulted in an extraordinary increase in database usability and programmer productivity.

Another benefit of this separation is that many aspects of the physical implementation may be changed without having to modify the abstract vision of the database. This substantially reduces the need to change existing application programs or retrain users.

The separation of the logical and physical levels of a database system is usually called the *data independence principle*. This is arguably the most important distinction between file systems and database systems.

The second separation in the architecture, between external and logical levels, is also important. It permits different perspectives, or *views*, on the database that are tailored to specific needs. Views hide irrelevant information and restructure data that is retained. Such views may be simple, as in the case of automatic teller machines, or highly intricate, as in the case of computer-aided design systems.

A major issue connected with both separations in the architecture is the trade-off between human convenience and reasonable performance. For example, the separation between logical and physical means that the system must compile queries and updates directed to the logical representation into "real" programs. Indeed, the use of the relational model became widespread only when query optimization techniques made it feasible. More generally, as the field of physical database optimization has matured, logical models have become increasingly remote from physical storage. Developments in hardware (e.g., large and fast memories) are also influencing the field a great deal by continually changing the limits of feasibility.

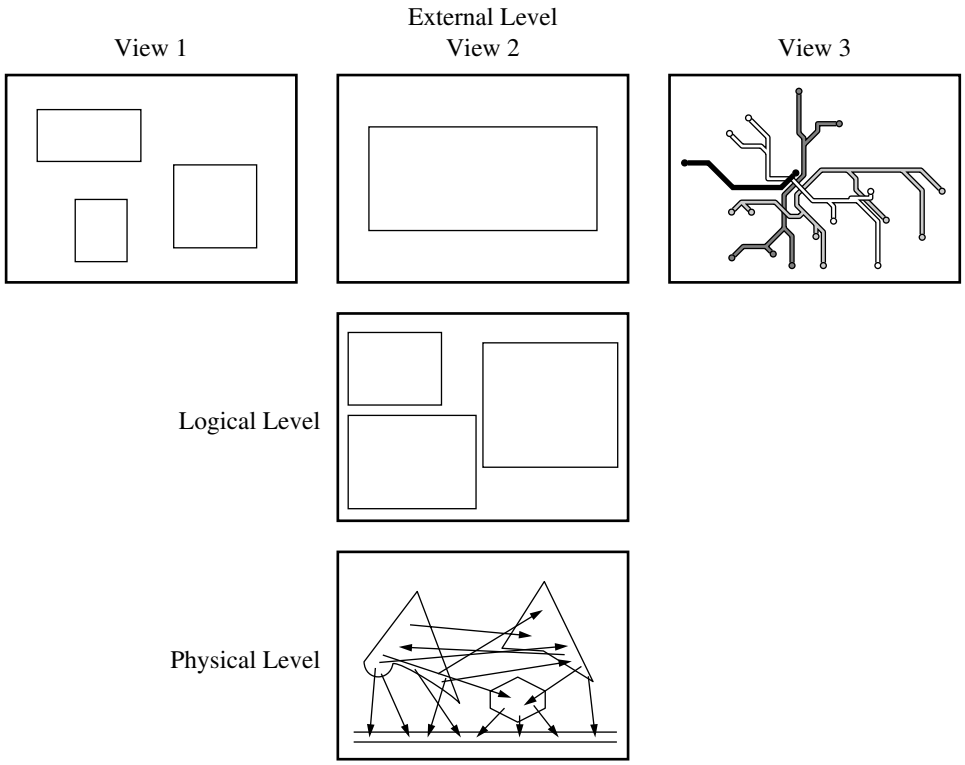

Figure 1.2: Three-level architecture of database systems

1.2 Functionalities

Modern dbms's include a broad array of functionalities, ranging from the very physical to the relatively abstract. Some functionalities, such as database recovery, can largely be ignored by almost all users. Others (even among the most physical ones, such as indexing) are presented to application programmers in abstracted ways.

The primary functionalities of dbms's are as follows:

Secondary storage management: The goal of dbms's is the management of large amounts of shared data. By *large* we mean that the data is too big to fit in main memory. Thus an essential task of these systems is the management of secondary storage, which involves an array of techniques such as indexing, clustering, and resource allocation.

Persistence: Data should be persistent (i.e., it should survive the termination of a particular database application so that it may be reused later). This is a clear divergence from standard programming, in which a data structure must be coded in a file to live beyond the execution of an application. Persistent programming languages (e.g., persistent C^{++}) are now emerging to overcome this limitation of programming languages.

Concurrency control: Data is shared. The system must support simultaneous access to shared information in a harmonious environment that controls access conflicts and presents a coherent database state to each user. This has led to important notions such as *transaction* and *serializability* and to techniques such as two-phase locking that ensure serializability.

Data protection: The database is an invaluable source of information that must be protected against human and application program errors, computer failures, and human misuse. *Integrity checking mechanisms* focus on preventing inconsistencies in the stored data resulting, for example, from faulty update requests. Database *recovery* and *back-up* protocols guard against hardware failures, primarily by maintaining snapshots of previous database states and logs of transactions in progress. Finally, *security control* mechanisms prevent classes of users from accessing and/or changing sensitive information.

Human-machine interface: This involves a wide variety of features, generally revolving around the logical representation of data. Most concretely, this encompasses DDLs and DMLs, including both those having a traditional linear format and the emerging visual interfaces incorporated in so-called fourth-generation languages. Graphically based tools for database installation and design are popular.

Distribution: In many applications, information resides in distinct locations. Even within a local enterprise, it is common to find interrelated information spread across several databases, either for historical reasons or to keep each database within manageable size. These databases may be supported by different systems (interoperability) and based on distinct models (heterogeneity). The task of providing transparent access to multiple systems is a major research topic of the 1990s.

Compilation and optimization: A major task of database systems is the translation of the requests against the external and logical levels into executable programs. This usually involves one or more compilation steps and intensive optimization so that performance is not degraded by the convenience of using more friendly interfaces.

Some of these features concern primarily the physical data level: concurrency control, recovery, and secondary storage management. Others, such as optimization, are spread across the three levels.

Database theory and more generally, database models have focused primarily on the description of data and on querying facilities. The support for designing application software, which often constitutes a large component of databases in the field, has generally been overlooked by the database research community. In relational systems applications can be written in C and extended with embedded SQL (the standard relational query language) commands for accessing the database. Unfortunately there is a significant distance between the paradigms of C and SQL. The same can be said to a certain extent about fourth-generation languages. Modern approaches to improving application programmer productivity, such as object-oriented or active databases, are being investigated.

1.3 Complexity and Diversity

In addition to supporting diverse functionalities, the field of databases must address a broad variety of uses, styles, and physical platforms. Examples of this variety include the following:

Applications: Financial, personnel, inventory, sales, engineering design, manufacturing control, personal information, etc.

Users: Application programmers and software, customer service representatives, secretaries, database administrators (dba's), computer gurus, other databases, expert systems, etc.

Access modes: Linear and graphical data manipulation languages, special purpose graphical interfaces, data entry, report generation, etc.

Logical models: The most prominent of these are the network, hierarchical, relational, and object-oriented models; and there are variations in each model as implemented by various vendors.

Platforms: Variations in host programming languages, computing hardware and operating systems, secondary storage devices (including conventional disks, optical disks, tape), networks, etc.

Both the quality and quantity of variety compounds the complexity of modern dbms's, which attempt to support as much diversity as possible.

Another factor contributing to the complexity of database systems is their longevity. Although some databases are used by a single person or a handful of users for a year or less, many organizations are using databases implemented over a decade ago. Over the years, layers of application software with intricate interdependencies have been developed for these "legacy" systems. It is difficult to modernize or replace these databases because of the tremendous volume of application software that uses them on a routine basis.

1.4 Past and Future

After the advent of the three-level architecture, the field of databases has become increasingly abstract, moving away from physical storage devices toward human models of information organization. Early dbms's were based on the network and hierarchical models. Both provide some logical organization of data (in graphs and trees), but these representations closely mirror the physical storage of the data. Furthermore, the DMLs for these are primitive because they focus primarily on navigation through the physically stored data.

In the 1970s, Codd's relational model revolutionized the field. In this model, humans view the data as organized in relations (tables), and more "declarative" languages are provided for data access. Indexes and other mechanisms for maintaining the interconnection between data are largely hidden from users. The approach became increasingly accepted as implementation and optimization techniques could provide reasonable response times in spite of the distance between logical and physical data organization. The relational model also provided the initial basis for the development of a mathematical investigation of databases, largely because it bridges the gap between data modeling and mathematical logic.

Historically dbms's were biased toward business applications, and the relational model best fitted the needs. However, the requirements for the management of large, shared amounts of data were also felt in a variety of fields, such as computer-aided design and expert systems. These new applications require more in terms of structures (more complex than relations), control (more dynamic environments), and intelligence (incorporation of knowledge). They have generated research and developments at the border of other fields. Perhaps the most important developments are the following:

Object-oriented databases: These have come from the merging of database technology, object-oriented languages (e.g., C++), and artificial intelligence (via semantic models). In addition to providing richer logical data structures, they permit the incorporation of behavioral information into the database schema. This leads to better interfaces and a more modular perspective on application software; and, in particular, it improves the programmer's productivity.

Deductive and active databases: These originated from the fusion of database technology and, respectively, logic programming (e.g., Prolog) and production-rule systems (e.g., OPS5). The hope is to provide mechanisms that support an abstract view of some aspects of information processing analogous to the abstract view of data provided by logical data models. This processing is generally represented in the form of rules and separated from the control mechanism used for applying the rules.

These two directions are catalysts for significant new developments in the database field.

1.5 Ties with This Book

Over the past two decades, database theory has pursued primarily two directions. The principal one, which is the focus of this book, concerns those topics that can meaningfully be discussed within the logical and external layers. The other, which has a different flavor and is not discussed in this book, is the elegant theory of concurrency control.

The majority of this book is devoted to the study of the relational model. In particular, relational query languages and language primitives such as recursion are studied in depth. The theory of dependencies, which provides the formal foundation of integrity constraints, is also covered. In the last part of the book, we consider more recent topics whose theory is generally less well developed, including object-oriented databases and behavioral aspects of databases.

By its nature, theoretical investigation requires the careful articulation of all assumptions. This leads to a focus on abstract, simplified models of much more complex practical situations. For example, one focus in the early part of this book is on conjunctive queries. These form the core of the *select-from-where* clause of the standard language in database systems, SQL, and are perhaps the most important class of queries from a practical standpoint. However, the conjunctive queries ignore important practical components of SQL, such as arithmetic operations.

Speaking more generally, database theory has focused rather narrowly on specific areas that are amenable to theoretical investigation. Considerable effort has been directed toward the expressive power and complexity of both query languages and dependencies, in which close ties with mathematical logic and complexity theory could be exploited. On the

other hand, little theory has emerged in connection with physical query optimization, in which it is much more difficult to isolate a small handful of crucial features upon which a meaningful theoretical investigation can be based. Other fundamental topics are only now receiving attention in database theory (e.g., the behavioral aspects of databases).

Theoretical research in computer science is driven both by the practical phenomena that it is modeling and by aesthetic and mathematical rigor. Although practical motivations are touched on, this text dwells primarily on the mathematical view of databases and presents many concepts and techniques that have not yet found their place in practical systems. For instance, in connection with query optimization, little is said about the heuristics that play such an important role in current database systems. However, the homomorphism theorem for conjunctive queries is presented in detail; this elegant result highlights the essential nature of conjunctive queries. The text also provides a framework for analyzing a broad range of abstract query languages, many of which are either motivated by, or have influenced, the development of practical languages.

As we shall see, the data independence principle has fundamental consequences for database theory. Indeed, much of the specificity of database theory, and particularly of the theory of query languages, is due to this principle.

With respect to the larger field of database systems, we hope this book will serve a dual purpose: (1) to explain to database system practitioners some of the underlying principles and characteristics of the systems they use or build, and (2) to arouse the curiosity of theoreticians reading this book to learn how database systems are actually created.

Bibliographic Notes

There are many books on database systems, including [Dat86, EN89, KS91, Sto88, Ull88, Ull89b, DA83, Vos91]. A (now old) bibliography on databases is given in [Kam81]. A good introduction to the field may be found in [KS91], whereas [Ull88, Ull89b] provides a more in-depth presentation.

The relational model is introduced in [Cod70]. The first text on the logical level of database theory is [Mai83]. More recent texts on the subject include [PBGG89], which focuses on aspects of relational database theory; [Tha91], which covers portions of dependency theory; and [Ull88, Ull89b], which covers both practical and theoretical aspects of the field. The reader is also referred to the excellent survey of relational database theory in [Kan88], which forms a chapter of the *Handbook of Theoretical Computer Science* [Lee91].

Database concurrency control is presented in [Pap86, BHG87]. Deductive databases are covered in [Bid91a, CGT90]. Collections of papers on this topic can be found in [Min88a]. Collections of papers on object-oriented databases are in [BDK92, KL89, ZM90]. Surveys on database topics include query optimization [JK84a, Gra93], deductive databases [GMN84, Min88b, BR88a], semantic database models [HK87, PM88], database programming languages [AB87a], aspects of heterogeneous databases [BLN86, SL90], and active databases [HW92, Sto92]. A forthcoming book on active database systems is [DW94].