

## Database architecture

Laboratory of Information and Communication Systems Security  
Department of Information and Communication Systems Engineering

(2012)

### Abstract

We are now in a position to provide a single picture (Figure 1.5) of the various components of a database system and the connections among them.

The architecture of a database system is greatly influenced by the underlying computer system on which the database system runs. Database systems can be centralized, or client-server, where one server machine executes work on behalf of multiple client machines. Database systems can also be designed to exploit parallel computer architectures. Distributed databases span multiple geographically separated machines.

In Chapter 17 we cover the general structure of modern computer systems. Chapter 18 describes how various actions of a database, in particular query processing, can be implemented to exploit parallel processing. Chapter 19 presents a number of issues that arise in a distributed database, and describes how to deal with each issue. The issues include how to store data, how to ensure atomicity of transactions that execute at multiple sites, how to perform concurrency control, and how to provide high availability in the presence of failures. Distributed query processing and directory systems are also described in this chapter.

Most users of a database system today are not present at the site of the database system, but connect to it through a network. We can therefore differentiate between **client** machines, on which remote database users work, and **server** machines, on which the database system runs.

## Introduction

Database applications are usually partitioned into two or three parts, as in Figure 1.6. In a **two-tier architecture**, the application resides at the client machine, where it invokes database system functionality at the server machine through

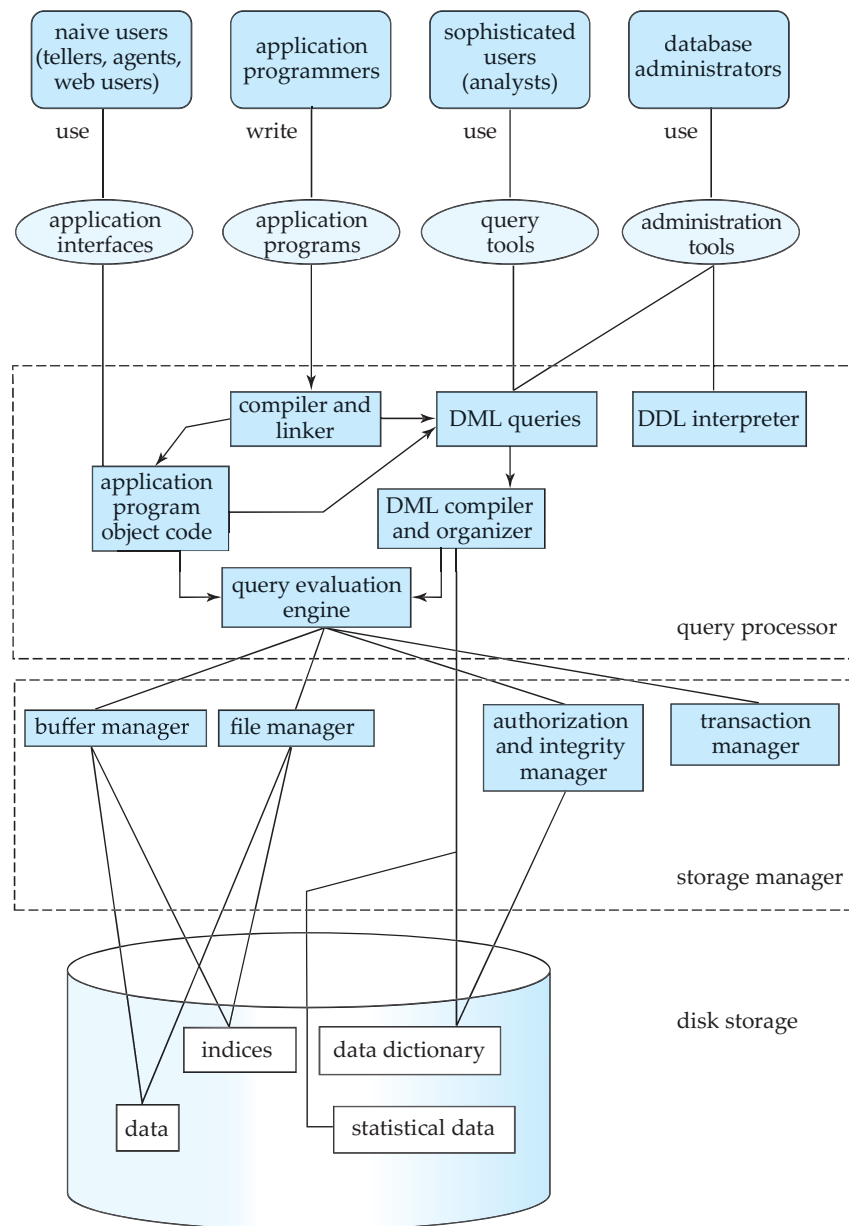
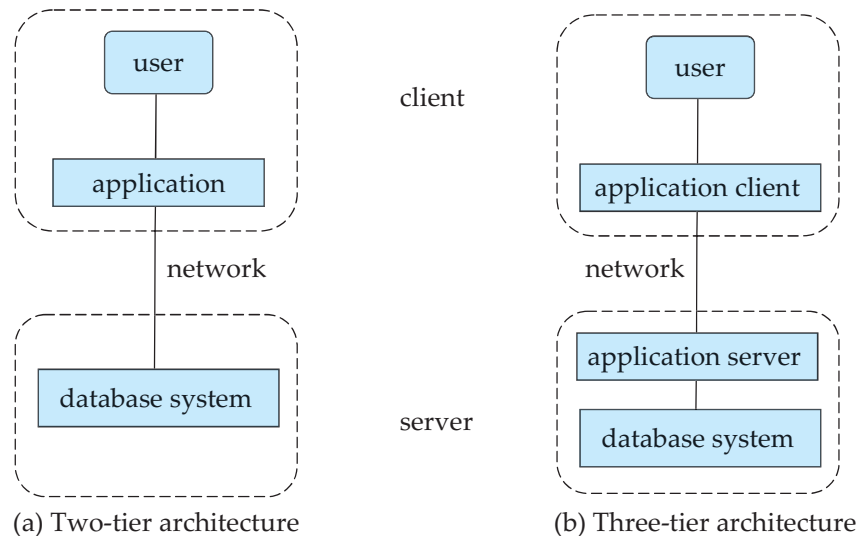


Figure 1.5 System structure.



**Figure 1.6** Two-tier and three-tier architectures.

query language statements. Application program interface standards like ODBC and JDBC are used for interaction between the client and the server.

In contrast, in a **three-tier architecture**, the client machine acts as merely a front end and does not contain any direct database calls. Instead, the client end communicates with an **application server**, usually through a forms interface. The application server in turn communicates with a database system to access data. The **business logic** of the application, which says what actions to carry out under what conditions, is embedded in the application server, instead of being distributed across multiple clients. Three-tier applications are more appropriate for large applications, and for applications that run on the World Wide Web.

## 1.10 Data Mining and Information Retrieval

The term **data mining** refers loosely to the process of semiautomatically analyzing large databases to find useful patterns. Like knowledge discovery in artificial intelligence (also called **machine learning**) or statistical analysis, data mining attempts to discover rules and patterns from data. However, data mining differs from machine learning and statistics in that it deals with large volumes of data, stored primarily on disk. That is, data mining deals with “knowledge discovery in databases.”

Some types of knowledge discovered from a database can be represented by a set of **rules**. The following is an example of a rule, stated informally: “Young women with annual incomes greater than \$50,000 are the most likely people to buy small sports cars.” Of course such rules are not universally true, but rather have

degrees of “support” and “confidence.” Other types of knowledge are represented by equations relating different variables to each other, or by other mechanisms for predicting outcomes when the values of some variables are known.

There are a variety of possible types of patterns that may be useful, and different techniques are used to find different types of patterns. In Chapter 20 we study a few examples of patterns and see how they may be automatically derived from a database.

Usually there is a manual component to data mining, consisting of preprocessing data to a form acceptable to the algorithms, and postprocessing of discovered patterns to find novel ones that could be useful. There may also be more than one type of pattern that can be discovered from a given database, and manual interaction may be needed to pick useful types of patterns. For this reason, data mining is really a semiautomatic process in real life. However, in our description we concentrate on the automatic aspect of mining.

Businesses have begun to exploit the burgeoning data online to make better decisions about their activities, such as what items to stock and how best to target customers to increase sales. Many of their queries are rather complicated, however, and certain types of information cannot be extracted even by using SQL.

Several techniques and tools are available to help with decision support. Several tools for data analysis allow analysts to view data in different ways. Other analysis tools precompute summaries of very large amounts of data, in order to give fast responses to queries. The SQL standard contains additional constructs to support data analysis.

Large companies have diverse sources of data that they need to use for making business decisions. To execute queries efficiently on such diverse data, companies have built *data warehouses*. Data warehouses gather data from multiple sources under a unified schema, at a single site. Thus, they provide the user a single uniform interface to data.

Textual data, too, has grown explosively. Textual data is unstructured, unlike the rigidly structured data in relational databases. Querying of unstructured textual data is referred to as *information retrieval*. Information retrieval systems have much in common with database systems—in particular, the storage and retrieval of data on secondary storage. However, the emphasis in the field of information systems is different from that in database systems, concentrating on issues such as querying based on keywords; the relevance of documents to the query; and the analysis, classification, and indexing of documents. In Chapters 20 and 21, we cover decision support, including online analytical processing, data mining, data warehousing, and information retrieval.

## 1.11 Specialty Databases

Several application areas for database systems are limited by the restrictions of the relational data model. As a result, researchers have developed several data models to deal with these application domains, including object-based data models and semistructured data models.

### 1.11.1 Object-Based Data Models

Object-oriented programming has become the dominant software-development methodology. This led to the development of an **object-oriented data model** that can be seen as extending the E-R model with notions of encapsulation, methods (functions), and object identity. Inheritance, object identity, and encapsulation (information hiding), with methods to provide an interface to objects, are among the key concepts of object-oriented programming that have found applications in data modeling. The object-oriented data model also supports a rich type system, including structured and collection types. In the 1980s, several database systems based on the object-oriented data model were developed.

The major database vendors presently support the **object-relational data model**, a data model that combines features of the object-oriented data model and relational data model. It extends the traditional relational model with a variety of features such as structured and collection types, as well as object orientation. Chapter 22 examines the object-relational data model.

### 1.11.2 Semistructured Data Models

Semistructured data models permit the specification of data where individual data items of the same type may have different sets of attributes. This is in contrast with the data models mentioned earlier, where every data item of a particular type must have the same set of attributes.

The XML language was initially designed as a way of adding markup information to text documents, but has become important because of its applications in data exchange. XML provides a way to represent data that have nested structure, and furthermore allows a great deal of flexibility in structuring of data, which is important for certain kinds of nontraditional data. Chapter 23 describes the XML language, different ways of expressing queries on data represented in XML, and transforming XML data from one form to another.

## 1.12 Database Users and Administrators

A primary goal of a database system is to retrieve information from and store new information into the database. People who work with a database can be categorized as database users or database administrators.

### 1.12.1 Database Users and User Interfaces

There are four different types of database-system users, differentiated by the way they expect to interact with the system. Different types of user interfaces have been designed for the different types of users.

- **Naïve users** are unsophisticated users who interact with the system by invoking one of the application programs that have been written previously. For example, a clerk in the university who needs to add a new instructor to

department *A* invokes a program called *new\_hire*. This program asks the clerk for the name of the new instructor, her new *ID*, the name of the department (that is, *A*), and the salary.

The typical user interface for naïve users is a forms interface, where the user can fill in appropriate fields of the form. Naïve users may also simply read *reports* generated from the database.

As another example, consider a student, who during class registration period, wishes to register for a class by using a Web interface. Such a user connects to a Web application program that runs at a Web server. The application first verifies the identity of the user, and allows her to access a form where she enters the desired information. The form information is sent back to the Web application at the server, which then determines if there is room in the class (by retrieving information from the database) and if so adds the student information to the class roster in the database.

- **Application programmers** are computer professionals who write application programs. Application programmers can choose from many tools to develop user interfaces. **Rapid application development (RAD)** tools are tools that enable an application programmer to construct forms and reports with minimal programming effort.
- **Sophisticated users** interact with the system without writing programs. Instead, they form their requests either using a database query language or by using tools such as data analysis software. Analysts who submit queries to explore data in the database fall in this category.
- **Specialized users** are sophisticated users who write specialized database applications that do not fit into the traditional data-processing framework. Among these applications are computer-aided design systems, knowledge-base and expert systems, systems that store data with complex data types (for example, graphics data and audio data), and environment-modeling systems. Chapter 22 covers several of these applications.

### 1.12.2 Database Administrator

One of the main reasons for using DBMSs is to have central control of both the data and the programs that access those data. A person who has such central control over the system is called a **database administrator (DBA)**. The functions of a DBA include:

- **Schema definition.** The DBA creates the original database schema by executing a set of data definition statements in the DDL.
- **Storage structure and access-method definition.**
- **Schema and physical-organization modification.** The DBA carries out changes to the schema and physical organization to reflect the changing needs of the organization, or to alter the physical organization to improve performance.

- **Granting of authorization for data access.** By granting different types of authorization, the database administrator can regulate which parts of the database various users can access. The authorization information is kept in a special system structure that the database system consults whenever someone attempts to access the data in the system.
- **Routine maintenance.** Examples of the database administrator's routine maintenance activities are:
  - Periodically backing up the database, either onto tapes or onto remote servers, to prevent loss of data in case of disasters such as flooding.
  - Ensuring that enough free disk space is available for normal operations, and upgrading disk space as required.
  - Monitoring jobs running on the database and ensuring that performance is not degraded by very expensive tasks submitted by some users.

## Conclusion

Information processing drives the growth of computers, as it has from the earliest days of commercial computers. In fact, automation of data processing tasks predates computers. Punched cards, invented by Herman Hollerith, were used at the very beginning of the twentieth century to record U.S. census data, and mechanical systems were used to process the cards and tabulate results. Punched cards were later widely used as a means of entering data into computers.

Techniques for data storage and processing have evolved over the years:

- **1950s and early 1960s:** Magnetic tapes were developed for data storage. Data processing tasks such as payroll were automated, with data stored on tapes. Processing of data consisted of reading data from one or more tapes and writing data to a new tape. Data could also be input from punched card decks, and output to printers. For example, salary raises were processed by entering the raises on punched cards and reading the punched card deck in synchronization with a tape containing the master salary details. The records had to be in the same sorted order. The salary raises would be added to the salary read from the master tape, and written to a new tape; the new tape would become the new master tape.

Tapes (and card decks) could be read only sequentially, and data sizes were much larger than main memory; thus, data processing programs were forced to process data in a particular order, by reading and merging data from tapes and card decks.
- **Late 1960s and 1970s:** Widespread use of hard disks in the late 1960s changed the scenario for data processing greatly, since hard disks allowed direct access to data. The position of data on disk was immaterial, since any location on disk could be accessed in just tens of milliseconds. Data were thus freed from

the tyranny of sequentiality. With disks, network and hierarchical databases could be created that allowed data structures such as lists and trees to be stored on disk. Programmers could construct and manipulate these data structures.

A landmark paper by Codd [1970] defined the relational model and nonprocedural ways of querying data in the relational model, and relational databases were born. The simplicity of the relational model and the possibility of hiding implementation details completely from the programmer were enticing indeed. Codd later won the prestigious Association of Computing Machinery Turing Award for his work.

- **1980s:** Although academically interesting, the relational model was not used in practice initially, because of its perceived performance disadvantages; relational databases could not match the performance of existing network and hierarchical databases. That changed with System R, a groundbreaking project at IBM Research that developed techniques for the construction of an efficient relational database system. Excellent overviews of System R are provided by Astrahan et al. [1976] and Chamberlin et al. [1981]. The fully functional System R prototype led to IBM's first relational database product, SQL/DS. At the same time, the Ingres system was being developed at the University of California at Berkeley. It led to a commercial product of the same name. Initial commercial relational database systems, such as IBM DB2, Oracle, Ingres, and DEC played a major role in advancing technique for efficient processing of declarative queries. By the early 1980s, relational databases had become competitive with network and hierarchical database systems even in the area of performance. Relational databases were so easy to use that they eventually replaced network and hierarchical databases; programmers using such databases were forced to deal with many low-level implementation details, and had to code their queries in a procedural fashion. Most importantly, they had to keep efficiency in mind when designing their programs, which involved a lot of effort. In contrast, in a relational database, almost all these low-level tasks are carried out automatically by the database, leaving the programmer free to work at a logical level. Since attaining dominance in the 1980s, the relational model has reigned supreme among data models.

The 1980s also saw much research on parallel and distributed databases, as well as initial work on object-oriented databases.

- **Early 1990s:** The SQL language was designed primarily for decision support applications, which are query-intensive, yet the mainstay of databases in the 1980s was transaction-processing applications, which are update-intensive. Decision support and querying re-emerged as a major application area for databases. Tools for analyzing large amounts of data saw large growths in usage.

Many database vendors introduced parallel database products in this period. Database vendors also began to add object-relational support to their databases.



## REFERENCES

- [1] J. Campbell, "Speaker recognition: a tutorial," *Proc. IEEE*, vol. 85, pp. 1437–1462, Sept. 1997.
- [2] D. A. Reynolds, T. Quatieri, and R. Dunn, "Speaker verification using adapted Gaussian mixture models," *Digital Signal Processing*, vol. 10, no. 1–3, pp. 19–41, 2000.
- [3] D. A. Reynolds, "Comparison of background normalization methods for text-independent speaker verification," in *Proc. Eurospeech*, 1997.
- [4] D. A. Reynolds and R. C. Rose, "Robust text-independent speaker identification using Gaussian mixture speaker models," *IEEE Trans. Speech Audio Processing*, vol. 3, no. 1, pp. 72–83, 1995.
- [5] J. L. Gauvain and C.-H. Lee, "Maximum a posteriori estimation for multivariate Gaussian mixture observations of Markov chains," *IEEE Trans. Speech Audio Processing*, vol. 2, pp. 291–298, Apr. 1994.
- [6] E. Bocchieri, "Vector quantization for the efficient computation of continuous density likelihoods," in *Proc. Int. Conf. Acoustics, Speech, Signal Processing*, 1993, pp. 692–695.
- [7] K. M. Knill, M. J. F. Gales, and S. J. Young, "Use of Gaussian selection in large vocabulary continuous speech recognition using HMMs," in *Proc. Int. Conf. Spoken Language Processing*, 1996.
- [8] D. B. Paul, "An investigation of Gaussian shortlists," in *Proc. Automatic Speech Recognition and Understanding Workshop*, 1999.
- [9] T. Watanabe, K. Shinoda, K. Takagi, and K.-I. Iso, "High speed speech recognition using tree-structured probability density function," in *Proc. Int. Conf. Acoustics, Speech, Signal Processing*, 1995.
- [10] J. Simonin, L. Delphin-Poulat, and G. Damnati, "Gaussian density tree structure in a multi-Gaussian HMM-based speech recognition system," in *Proc. Int. Conf. Acoustics, Speech, Signal Processing*, 1998.
- [11] T. J. Hanzen and A. K. Halberstadt, "Using aggregation to improve the performance of mixture Gaussian acoustic models," in *Proc. Int. Conf. Acoustics, Speech, Signal Processing*, 1998.
- [12] M. Padmanabhan, L. R. ahl, and D. Nahamoo, "Partitioning the feature space of a classifier with linear hyperplanes," *IEEE Trans. Speech Audio Processing*, vol. 7, no. 3, pp. 282–288, 1999.
- [13] R. Auckenthaler and J. Mason, "Gaussian selection applied to text-independent speaker verification," in *Proc. A Speaker Odyssey—Speaker Recognition Workshop*, 2001.
- [14] J. McLaughlin, D. Reynolds, and T. Gleason, "A study of computation speed-ups of the GMM-UBM speaker recognition system," in *Proc. Eurospeech*, 1999.
- [15] S. van Vuuren and H. Hermansky, "On the importance of components of the modulation spectrum of speaker verification," in *Proc. Int. Conf. Spoken Language Processing*, 1998.
- [16] B. L. Pellom and J. H. L. Hansen, "An efficient scoring algorithm for Gaussian mixture model based speaker identification," *IEEE Signal Processing Lett.*, vol. 5, no. 11, pp. 281–284, 1998.
- [17] J. Oglesby and J. S. Mason, "Optimization of neural models for speaker identification," in *Proc. Int. Conf. Acoustics, Speech, Signal Processing*, 1990, pp. 261–264.
- [18] Y. Bengio, R. De Mori, G. Flammia, and R. Kompe, "Global optimization of a neural network—hidden Markov model hybrid," *IEEE Trans. Neural Networks*, vol. 3, no. 2, pp. 252–259, 1992.
- [19] H. Bourlard and C. J. Wellekins, "Links between Markov models and multilayer perceptrons," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 12, pp. 1167–1178, Dec. 1990.
- [20] J. Navrátil, U. V. Chaudhari, and G. N. Ramaswamy, "Speaker verification using target and background dependent linear transforms and multi-system fusion," in *Proc. Eurospeech*, 2001.
- [21] L. P. Heck, Y. Konig, M. K. Sonmez, and M. Weintraub, "Robustness to telephone handset distortion in speaker recognition by discriminative feature design," *Speech Commun.*, vol. 31, pp. 181–192, 2000.
- [22] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. R. Statist. Soc.*, vol. 39, pp. 1–38, 1977.
- [23] K. Shinoda and C. H. Lee, "A structural Bayes approach to speaker adaptation," *IEEE Trans. Speech Audio Processing*, vol. 9, no. 3, pp. 276–287, 2001.
- [24] K. Fukunaga, *Introduction to Statistical Pattern Recognition*. New York: Academic, 1990.
- [25] J. C. Junqua, *Robust Speech Recognition in Embedded Systems and PC*
- [26] U. V. Chaudhari, J. Navrátil, S. H. Maes, and R. A. Gopinath, "Transformation enhanced multi-grained modeling for text-independent speaker recognition," in *Proc. Int. Conf. Spoken Language Processing*, 2000.
- [27] Q. Lin, E.-E. Jan, C. W. Che, D.-S. Yuk, and J. Flanagan, "Selective use of the speech spectrum and a VQGMM method for speaker identification," in *Proc. Int. Conf. Spoken Language Processing*, 1996.
- [28] S. Raudys, *Statistical and Neural Classifiers: An Integrated Approach to Design*. New York: Springer, 2001.
- [29] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing*. Cambridge, MA: MIT Press, 1986, pp. 318–364.
- [30] [Online] Available: <http://www.nist.gov/speech/tests/spk/index.htm>.
- [31] J. Pelecanos and S. Sridharan, "Feature warping for robust speaker verification," in *Proc. A Speaker Odyssey—Speaker Recognition Workshop*, 2001.
- [32] B. Xiang, U. V. Chaudhari, J. Navrátil, N. Ramaswamy, and R. A. Gopinath, "Short-time Gaussianization for robust speaker verification," in *Proc. Int. Conf. Acoustics, Speech, Signal Processing*, 2002.
- [33] G. R. Doddington, M. A. Przybocki, A. F. Martin, and D. A. Reynolds, "The NIST speaker recognition evaluation—overview, methodology, systems, results, perspective," *Speech Communication*, vol. 31, pp. 225–254, 2000.



**Bing Xiang** (M'03) was born in 1973 in China. He received the B.S. degree in radio and electronics and M.E. degree in signal and information processing from Peking University in 1995 and 1998, respectively. In January, 2003, he received the Ph.D. degree in electrical engineering from Cornell University, Ithaca, NY.

From 1995 to 1998, he worked on speaker recognition and auditory modeling in National Laboratory on Machine Perception, Peking University. Then he entered Cornell University and worked on speaker recognition and speech recognition in DISCOVER Lab as a Research Assistant. He also worked in the Human Language Technology Department of IBM Thomas J. Watson Research Center as a summer intern in both 2000 and 2001. He was a selected remote member of the SuperSID Group in the 2002 Johns Hopkins CLSP summer workshop in which he worked on speaker verification with high-level information. In January, 2003, he joined the Speech and Language Processing Department of BBN Technologies where he is presently a Senior Staff Consultant-Technology. His research interests include large vocabulary speech recognition, speaker recognition, speech synthesis, keyword spotting, neural networks and statistical pattern recognition.



**Toby Berger** (S'60–M'66–SM'74–F'78) was born in New York, NY, on September 4, 1940. He received the B.E. degree in electrical engineering from Yale University, New Haven, CT in 1962, and the M.S. and Ph.D. degrees in applied mathematics from Harvard University, Cambridge, MA in 1964 and 1966, respectively.

From 1962 to 1968 he was a Senior Scientist at Raytheon Company, Wayland, MA, specializing in communication theory, information theory, and coherent signal processing. In 1968 he joined the faculty of Cornell University, Ithaca, NY where he is presently the Irwin and Joan Jacobs Professor of Engineering. His research interests include information theory, random fields, communication networks, wireless communications, video compression, voice and signature compression and verification, neuroinformation theory, quantum information theory, and coherent signal processing. He is the author/co-author of *Rate Distortion Theory: A Mathematical Basis for Data Compression*, *Digital Compression for Multimedia: Principles and Standards*, and *Information Measures for Discrete Random Fields*.

Dr. Berger has served as editor-in-chief of the *IEEE TRANSACTIONS ON INFORMATION THEORY* and as president of the *IEEE Information Theory*