CHAPTER 2

# The Database System Environment

This chapter discusses the environment of a database system. The sections in this chapter are as follows:

- Levels of Architecture
- Inter-Level Mappings
- Database Personnel
- Database Management System
- Components of the DBMS Suite
- Front-End and Back-End Perspectives
- Database System Architecture
- Database System Classifications
- Summary and Concluding Remarks

## 2.1 Levels of Architecture

In [Date, 2004], Christopher Date describes three levels of architecture of a database system, namely, the *external level*, the *conceptual level*, and the *internal level*. These levels are illustrated in Figure 2-1; we will briefly discuss each.


E. C. Foster and S. Godbole, *Database Systems,* DOI 10.1007/978-1-4842-1191-5_2

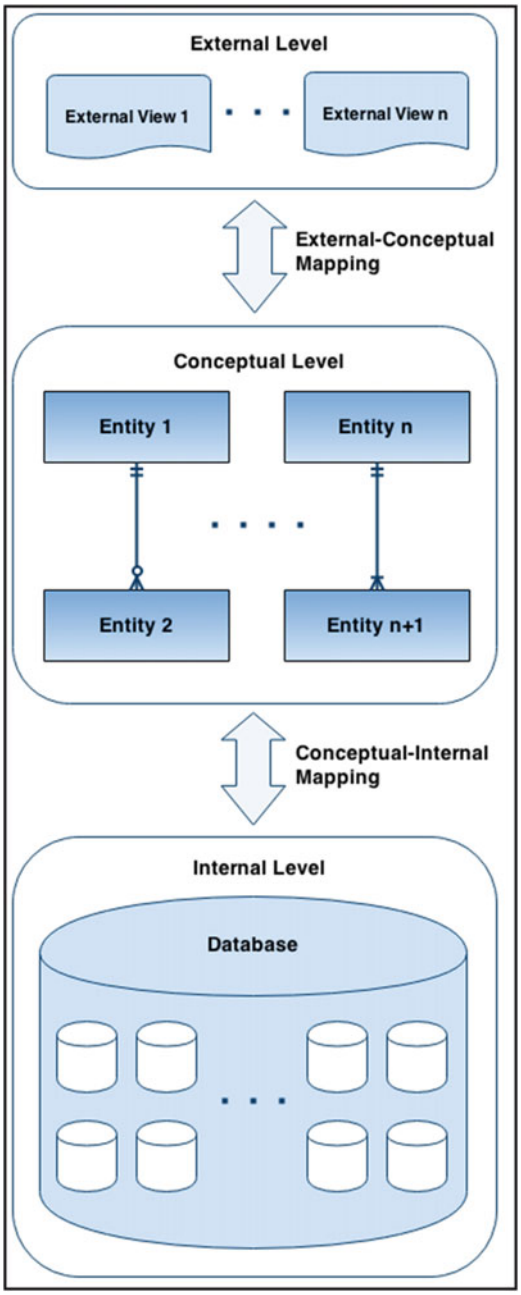


***Figure 2-1.*** *Levels of DBS architecture*

### 2.1.1 External Level

The *external level* is concerned with individual user views. It therefore varies according to users' perspectives. The external level is defined by the *external schema*.

Typically, the database is accessed through its external schema. The application programmer uses both the *host language* and the *data sublanguage* (DSL) to create a user interface that end users use to access the system:

- The DSL is the language that is concerned specifically with database objects and operations. To illustrate, SQL (*structured query language*) is the industry's standard DSL. Another example of a DSL is the knowledge query and management language (KQML). An older example which is no longer prevalent is QUEL. These languages (SQL and QUEL) will be further discussed later in the course.
- The host language is that which supports the DSL in addition to other non-database facilities such as manipulation of variables, computations, and Boolean logic. Host languages are typically high level languages (HLL); examples include C++, Java, C#, Python, PHP, PL/SQL, etc.

Typically, the DSL consists of a *data definition language* (DDL), a *data manipulation language* (DML), and a *data control language* (DCL). These components are not necessarily distinct entities, but are typically part of a single coherent product. These facilities allow users (both technical users and end users) to use the DSL to define assorted logical views of data in the database. In summary, the external schema is the user interpretation of the database, but facilitated by the DSL.

### 2.1.2 Conceptual Level

The *conceptual level* is an abstract representation of the entire information content of the database. It is defined by means of the *conceptual schema* (also called the *logical schema*), which includes the definition of each of the various *persistent objects* comprising the database. By persistent objects, we mean objects that are permanently stored in the database until explicitly deleted. You will learn more about these as the course proceeds.

The conceptual schema includes defining the structure of the database, security constraints, operational constraints, and integrity checks. It represents a closer picture of how data will be actually stored and managed, and is the level to which most technical user will relate.

The conceptual schema must adhere to the data independence requirement. Also, it must be comprehensive since it represents the realization of the entire database design.

### 2.1.3 Internal Level

Also called the *storage view*, the *internal level* is the low level representation of the database. It is one level above the physical level, which deals with pages, cylinders, and tracks on the storage device.

The internal level is defined by the *internal schema*, which addresses issues such as record types, indexes, field representation, physical storage sequence of records, data access, etc., and written in the internal DDL. Ultimately, this physical layer has a significant impact on the performance of the database.

# 2.2 Inter-Level Mappings

Continuing from the previous section, the literature also describes two levels of mappings that connect the three schemas (again, see [Date, 2004]). Figure 2-1 illustrates the different schemas and their interrelationships with respect to the DBMS. From the figure, observe that there are two levels of mapping: the *external-conceptual* mapping and the *conceptual-internal* mapping.

- The conceptual-internal mapping specifies how conceptual records are represented at the internal level. If changes are made at the internal level, this mapping must be updated. Traditionally, the *database administrator* (DBA) maintains this mapping in order to preserve data independence (the DBA is discussed in the next section). In contemporary systems, the DBMS automatically updates and maintains this mapping, transparent to the user.
- The external-conceptual mapping specifies how external views are linked to the conceptual level. In effect, this is achieved by application programs and logical views via the host language and the DSL.

It must be borne in mind that these levels are abstractions that facilitate understanding of the DBS environment. As an end user, you will most likely not visibly observe these levels of architecture. However, if as a software engineer, you find yourself on a software engineering team that is constructing or maintaining a DBMS (a huge undertaking), knowledge of these abstractions becomes critical.

# 2.3 Database Personnel

A database resource team varies on a continuum of a simple one-person team (for small business environments) to a multi-membered team (for more complex business environments). A database systems team may consist of the following role-players: database administrator(s), data architect(s), tools expert(s), application programmer(s), user liaison specialist(s), and network and infrastructure specialist(s). Typically, each team member is tasked with specific responsibilities to ensure the overall success of the database environment. Let's briefly examine each role.

## 2.3.1 Database Administrator and Data Architect

The database administrator (DBA) has overall responsibility for the control of the system at the technical level. The specific responsibilities will vary across different organizations. For instance, special tools may be required in the areas of report generation, data analytics, data preparation, etc. Some of the functions of the DBA may include the following:

- Defining the conceptual schema (i.e. logical database design)
- Defining the internal schema (i.e. physical database design)
- Liaising with users and identifying/defining views to facilitate the external schema
- Defining security and integrity checks
- Defining backup and recovery procedures
- Monitoring performance and responding to changing requirements

In smaller organizations, the tendency is to include these functions in the job description of the software engineer. This is quite rational and prudent, since good software engineering includes good database design. However, large corporations that rely on complex company database(s) on an on-going basis usually employ the services of one or more DBAs. Where there are multiple DBAs, one of them is designated the chief DBA.

Database administration is a very coveted specialization that requires advanced training (for example, see [Thomas 2014]). Because of the importance of having reliable databases, DBAs are among the highest paid information technology (IT) professionals. Chapter 16 provides a more detailed summary of the profession. However, please note that comprehensive coverage is beyond the scope of this course.

In large and complex database environments, a contemporary trend is to also define a role for a data architect. This makes sense, since in such circumstances, the role of the DBA can be overwhelming. A data architect focuses primarily on database design and integration with assorted data sources. This latter matter of database integration is of particular importance in environments where *data warehousing* and *information extraction* are applicable; these matters are discussed in Chapter 19.

### 2.3.2 Tools Expert

In very complex database environments, a tools expert is useful in providing expert advice on various technologies and/methodologies required to ensure an efficient and effective database environment. The required number of tools experts depends on the size and complexity of the DBS environment. For instance, special tools may be required in the areas of report generation, data analytics, data preparation, etc. Among the critical functions of the tools expert are the following:

- Developing mastery of the various technologies and methodologies required to enhance the DBS environment
- Providing training and support to the DBS resource team on the relevant technologies and methodologies
- Keeping abreast of current developments in related areas to ensure that the relevance of the DBS is extended for as long as possible
- Developing resource materials that will be useful to the DBS resource team

### 2.3.3 Application Programmer

Application programmers may be part of the DBS team or the related software engineering team. Whatever the scenario, the functions remain the same. The required number of application programmers varies across different organizations depending on the size and complexity of the DBS environment. Some of the critical functions are as follows:

- Preparation of operation specifications that represent the user needs for the software system(s) that access the database
- Developing application programs corresponding to the operation specifications for the software system(s) that access the database
- Analysis of user requests to determine the best alternative for meeting these requests
- Revision and/or enhancement of existing operation specifications and application programs in response to changing system dynamics and/or user needs
- Identification, diagnosis, and resolution of existing programming problems
- Provision of technical information relating to the various application programs, so that adequate user documentation can be prepared
- Installation of programming upgrades as required
- Documentation of any changes made to an existing related software system
- Recommendations for related software enhancements as required

### 2.3.4 User Liaison Specialist

In large, complex DBS environments with very large and diverse end user populations, it may be necessary to employ the services of one or more user liaison specialist(s) or business analyst(s). In less complex environments where this position is not filled, the responsibilities are spread across the other functional areas of the DBS resource team.

As the title suggests, the user liaison specialist acts as a facilitator who enables effective communication between the end users of the database and the more technical database professionals. The user liaison specialist role includes responsibilities such as

- Investigation of the user needs from various end user departments and for the related software system(s)
- Development and management of system change request forms for the end user departments
- Documentation of the user needs for the related software system(s)
- Reporting the required user needs to the DBS resource team in a timely and organized manner
- Conducting follow-up and/or feedback checks to ensure that user needs are satisfactorily addressed
- Preparation of appropriate system resources for end users
- Planning and execution of training sessions for end users of the related software system(s) as required

### 2.3.5 Network and Infrastructure Specialist

Network and infrastructure specialist(s) may be part of the DBS team or a complimentary networking team that is already in place. Whatever the scenario, the functions are similar. The required number of network/infrastructure specialists varies across different organizations depending on the size and complexity of the DBS environment. Network/infrastructure design and management is a specialization that is best covered in a course on computer networks. From a DBS perspective, some of the critical functions of the network specialist are as follows:

- Configuration, installation, and maintenance of all related hardware and network servers
- Configuration, installation, and management of related operating system(s) and other related software systems
- Definition and management of the network topology, spanning all domains, subnetworks, and reinforcement points
- Definition, implementation, and management of the network security constraints, spanning issues such as user permissions, group policies, access control on file servers, intrusion blocking, and other related issues
- Managing software installations on all network servers and client nodes
- Documentation of the network infrastructure
- Ensuring that any network-related problems are resolved in a timely manner

- Overseeing software integration across the network infrastructure to ensure that defined business requirements are met
- Defining and managing network security mechanisms for the environment
- Development and implementation of a preventative maintenance plan for all computers in the organization's network
- Troubleshooting hardware/software problems in the network, and taking appropriate actions to resolve them
- Management of all related network resources to ensure acceptable operation
- Conducting research into innovative and/or improved ways to enhance network performance
- Keeping abreast of contemporary developments in computer networks to ensure that the organization's network maintains state-of-the-art status

A full treatment of computer networks is beyond the scope of this course; you would typically cover such information in your computer networking course. Fortunately, there is no shortage of excellent resources on this topic (for instance, see [Comer 2015]).

## 2.4 The Database Management System

The database management system (DBMS) is the software that facilitates creation and management of the database. When a user issues a request via some DSL (typically SQL), it is the DBMS that interprets such a request, executes the appropriate instructions, and responds to the request. Depending on the nature of the initial request, the response may be relayed (by the DBMS) directly to the end user, or indirectly to the end user via an executing application program.

Through the DBMS, the objectives of the DBS that were mentioned in Chapter 1 are achieved. Primary functions of this very important software system include the following:

- Data definition (relations, dependencies, integrity constraints, views, etc.)
- Data manipulation (adding, updating, deleting, retrieving, reorganizing, and aggregating data)
- Data security and integrity checks
- Management of data access (including query optimization), archiving, and concurrency
- Maintenance of a user-accessible system catalog (data dictionary)
- Support of miscellaneous non-database functions (e.g. utilities such as copy)
- Programming language support
- Transaction management (either all changes are made or none is made)
- Backup and recovery services
- Communication support (allow the DBMS to integrate with underlying communications software)
- Support for interoperability including open database connectivity (ODBC), Java database connectivity (JDBC), and other related issues

Optimum efficiency and performance are the hallmarks of a good DBMS. To illustrate the critical role of the DBMS, consider the steps involved when an application program accesses the database:

1. **Program-A** issues a request to the DBMS (expressed in terms of subschema language).
2. DBMS looks at **Program-A** subschema, schema, and physical description (this information is stored in tables).
3. DBMS determines the optimal way to access the data, determining which files must be accessed, which records in the files are needed, and the best method to access them.
4. DBMS issues instruction(s) (reads or writes) to the operating system.
5. The operating system causes data transfer between disk storage and main memory.
6. DBMS issues moves to transfer required fields.
7. DBMS returns control to **Program-A** (possibly with a completion code).

Figure 2-2 provides a graphic representation, but bear in mind that these steps are carried out automatically, in a manner that is transparent to the user.

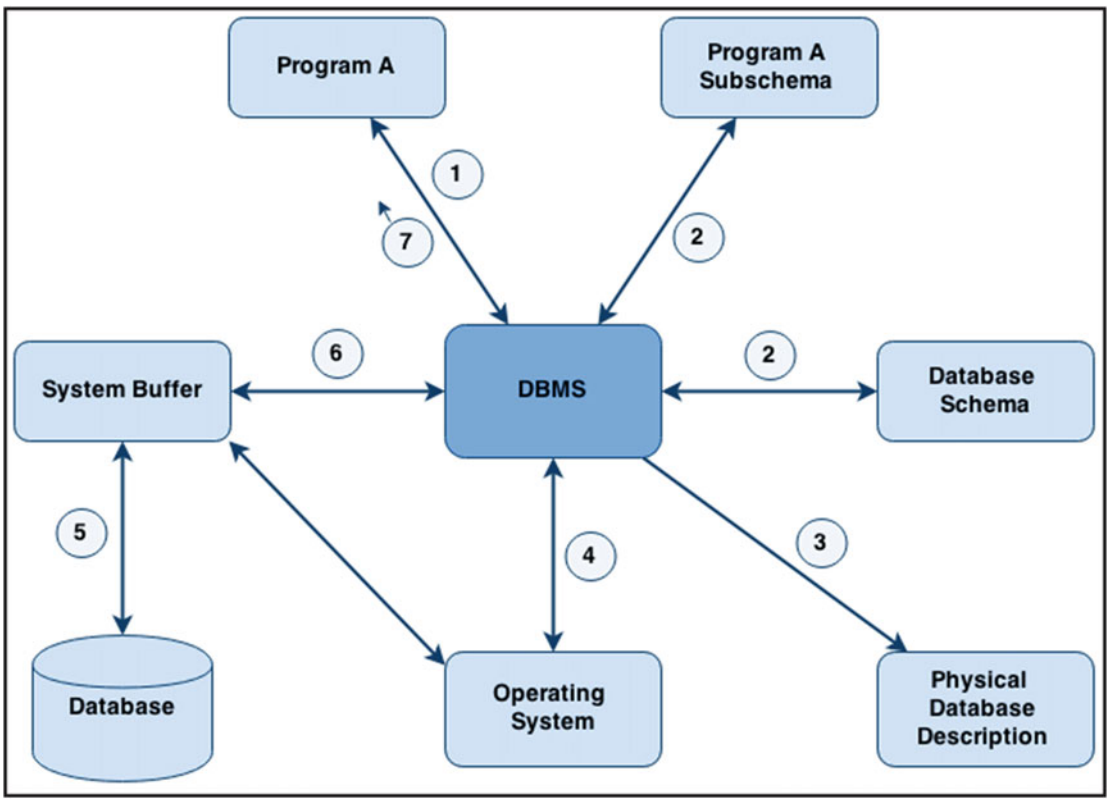


***Figure 2-2.*** *Steps involved when application programs access a database*

## 2.5 Components of DBMS Suite

The DBMS is actually a complex conglomeration of software components working together for a set of common objectives. For the purpose of illustration, we may represent the essential components of the DBMS as the following:

- DBMS Engine
- Data Definition Subsystem
- User Interface Subsystem
- Application Development Subsystem
- Data Administration Subsystem
- Data Dictionary Subsystem
- Data Communications Manager
- Utilities Subsystem

These functional components (illustrated in Figure 2-3) are not necessarily tangibly identifiable, but they exist to ensure the acceptable performance of the DBMS. These functional components are clarified in the upcoming subsections.

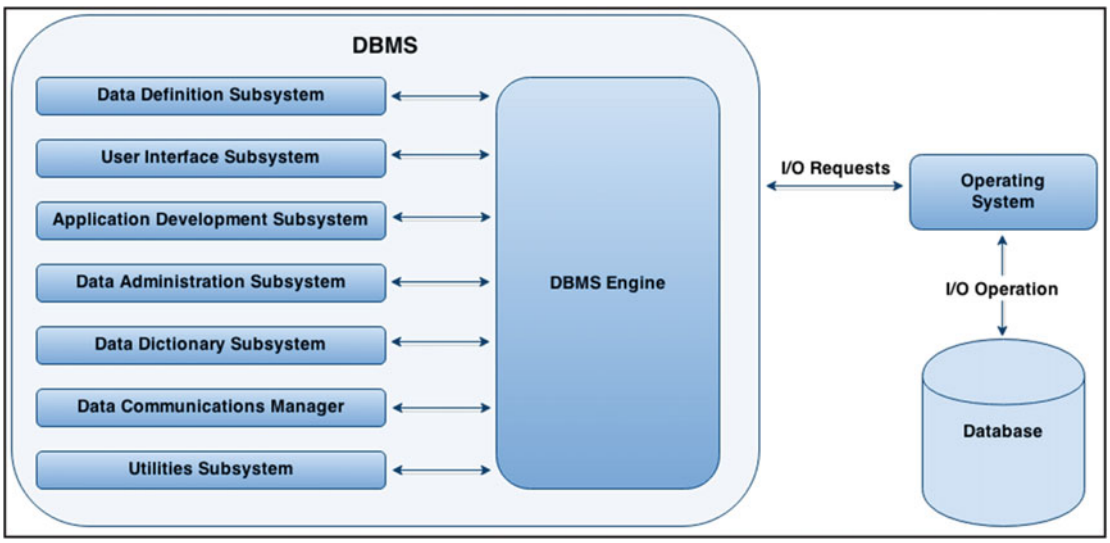


***Figure 2-3.*** *Functional components of a DBMS*

## 2.5.1 The DBMS Engine

The *DBMS engine* is the link between all other subsystems and the physical device (the computer) via the operating system. Some important functions are as follows:

- Provision of direct access to operating system utilities and programs (e.g. input/output requests, data compaction requests, communication requests, etc.)
- Management of file access (and data management) via the operating system
- Management of data transfer between memory and the system buffer(s) in order to effect user requests
- Maintenance of overhead data and metadata stored in the data dictionary (system catalog)

## 2.5.2 Data Definition Subsystem

The *data definition subsystem* (DDS or its equivalent) consists of tools and utilities for defining and changing the structure of the database. The structure includes relational tables, relationships, constraints, user profiles, overhead data structures, etc.

The DDL (data definition language) is used to define all database objects that make up the conceptual schema (relations, relationships, constraints, etc.). The DML (data manipulation language) is used to facilitate manipulation and usually includes a query language to insert, remove, update, and find data in the database. The DCL (data control language) is used to set up control environments for data management by the end user. As mentioned earlier, the DDL, DML, and DCL comprise the DSL.

## 2.5.3 The User Interface Subsystem

The *user interface subsystem* (UIS or its equivalent) allows users and programs to access the database via an interactive query language such as SQL and/or the host language. Suppose for example, that a file named **Student** has fields {ID#, SName, FName, Status, DOB,…} for each record. Two possible SQL queries on this file are shown in Example 2-1 (a more detailed study of SQL will be covered later in the course).

**Example 2-1. Sample SQL Queries**

```
/* Produce a list of students, starting with last name Bell and continuing to end of file */
SELECT ID#, SNAME, FNAME FROM STUDENT WHERE SNAME >= 'BELL';
/* Produce a list of students, starting with date of birth 19960101 and continuing to end of file */
SELECT ID#, SNAME, DOB FROM STUDENT WHERE DoB >= 19660101;
```

The traditional interface is command based; however in recent times menus and graphical user interfaces (GUI) have become more prevalent. Of course, it is not uncommon for a product to provide the user with all three interfaces (for example, Oracle). Other more sophisticated DBMS suites may use a *natural language* interface.

The user interface may also include a DBMS-specific programming language extensions (e.g. Oracle's PL/SQL). These language extensions pertain only to the DBMS in which they are used. Additionally, the DBMS may support multiple high level languages such as C++, Java, etc., thus making it more flexible and marketable.

### 2.5.4 Application Development Subsystem

The *application development subsystem* (ADS or its equivalent) contains tools for developing application components such as forms, reports, and menus. In some cases, it may be merged with the user interface subsystem. Typically, this subsystem provides a graphical user interface (GUI), which is superimposed on an underlying host language. The suite may include an automatic code generator (as in Delphi and Team Developer) or seamless access of the compiler of the host language (as in Oracle).

Much of the application development that takes place in a database environment will involve use of the services of the ADS. Additional facilities that may be provided through the ADS are as follows:

- Quick GUI-based access to data stored in the database
- Report writer for generating formatted reports
- Project manager component for facilitating the management of database-related projects
- Menu builder for assisting in the quick development of menus for database applications
- Graphic data interpreter

### 2.5.5 Data Administration Subsystem

The *data administration subsystem* (DAS) consists of a collection of utilities that facilitate effective management of the database. Included in this subsystem are facilities for backup and recovery, database tuning, and storage management. It is typically used by DBAs as well as software engineers.

### 2.5.6 Data Dictionary Subsystem

The *data dictionary* (DD) is a traditional term to refer to the *system catalog* (which is the preferred term) in many systems. The system catalog contains information on the database structure, relationships among database objects, system and object privileges, users, integrity constraints, etc. It is automatically created and maintained by the DBMS.

The system catalog contains all metadata for the database. It can be queried using the same commands used to manipulate source data; it is therefore of inestimable value to the DBAs and software engineers. More will be said about the system catalog later in the course.

### 2.5.7 Data Communications Manager

Traditionally a separate system that is linked to the DBMS, the *data communications manager* (DCM) caries out functions such as

- Handling communication to remote users in a distributed environment
- Handling messages to and from the DBMS
- Communication with other DBMS suites

Modern systems tend to have this subsystem as an integral part of the DBMS suite. In short, the data communications manager ensures that the database communicates effectively with all client requests in a client-server-based environment. Typically, the server-based portions of the DBMS will be running on machines designated as servers in the network. All other nodes are then deemed as client nodes that can request database services from a server. There may be several database servers in the network; also, a node may act as both a server and a client (provided the essential software components are in place).

### 2.5.8 Utilities Subsystem

Utilities are programs that perform various administrative tasks. The *utilities subsystem* consists of various utility programs that are applicable to the database environment. Examples of utilities are as follows:

- Load routines to create an initial version of a database from non-database files
- Copy routines for duplicating information
- Reorganization routines to reorganize data in the database
- File deletion routine(s)
- Statistics routines to compute and store file statistics
- Backup and recovery utilities
- Tools for importing data of disparate format(s) to the database and exporting data from the database to different formats
- Other utilities (that might have been) developed by application programmers

## 2.6 Front-End and Back-End Perspectives

A DBS can be perceived as a simple two-part structure—a *front end* and a *back end*. The front end consists of end users, applications, and a programming interface; the back end consists of the actual DBMS and the database, and it relates to the actual creation and administration of the database, the inner workings of the system, processing requests, memory management, and I/O management.

The front-end system may be on a different machine from the back-end system and the two are connected by a communication network. Typically, an IDE (integrated development environment) fulfills the role of the front-end system. Common examples include the following: Delphi (supporting languages Object Pascal, and C++); NetBeans (supporting Java, C++, JavaScript, PHP, XML, HTML, etc.); Visual Studio (supporting C#, C++, JavaScript, etc.); and Qt (supporting C++, C#, Java, Python, Ruby, etc.). The back-end system may be any of the leading DBMS suites in the marketplace — Oracle, DB2, MySQL, SQL Server, or PostGreSQL. Figure 2-4 illustrates this dual concept of front-end and back-end perspectives.

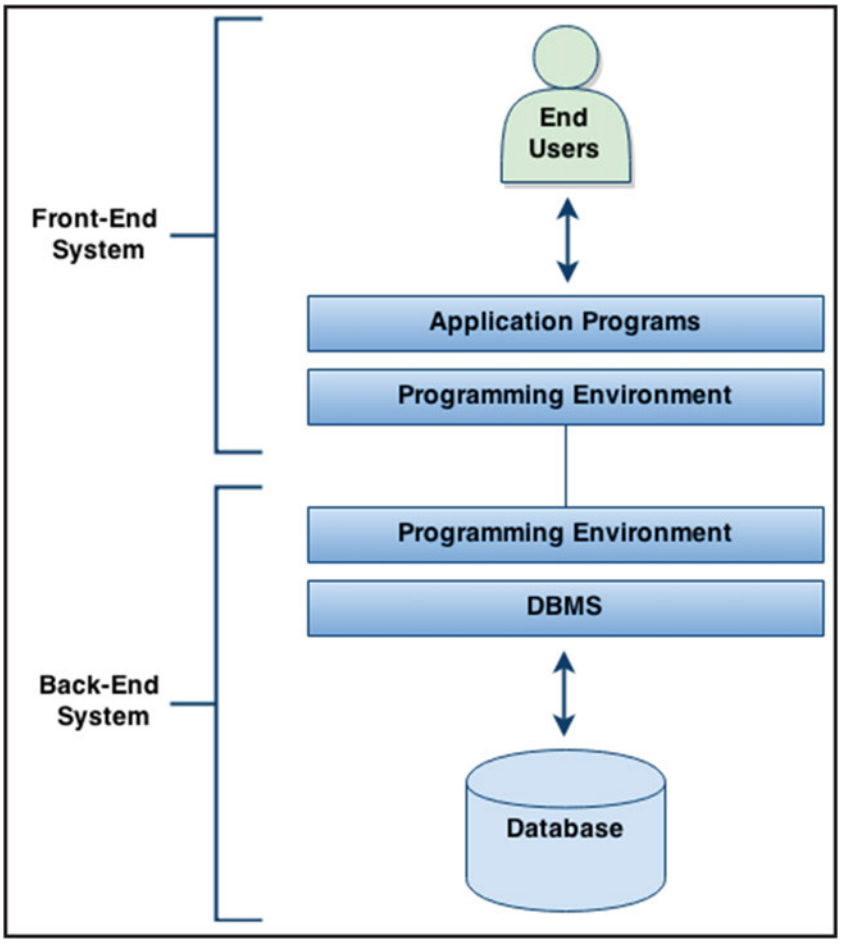


***Figure 2-4.*** *Front-end and back-end perspectives*

## 2.7 Database System Architecture

There may be added benefits of using different machines for the back-end and front-end systems. Figures 2-5 through 2-7 show three possible configurations. Please note also that various network topologies are applicable to any computer network (network topology is outside of the scope of this course; however, it is assumed that you are familiar with such information).

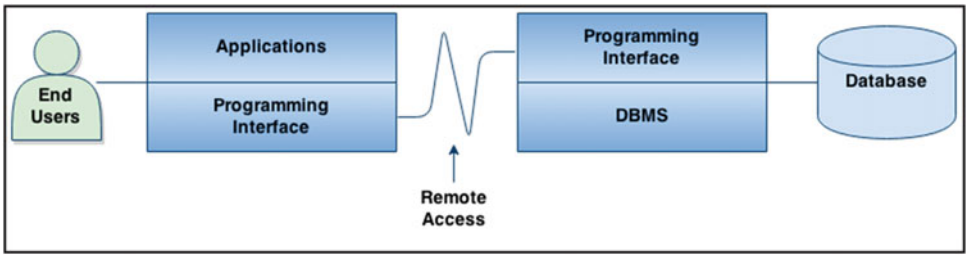


***Figure 2-5.*** *Back end and front end running on different machines*

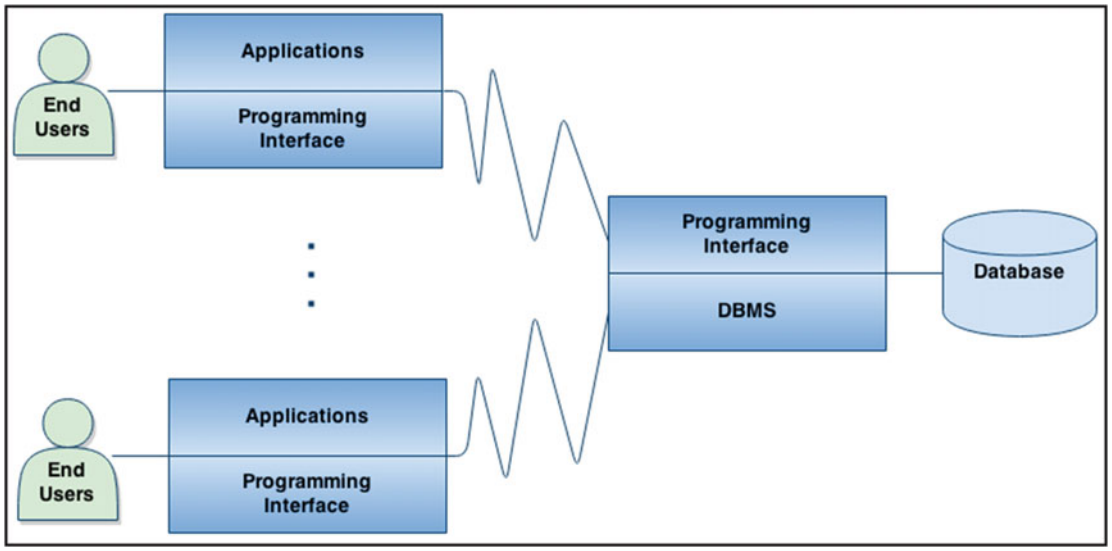


***Figure 2-6.*** *One back end and multiple front ends*

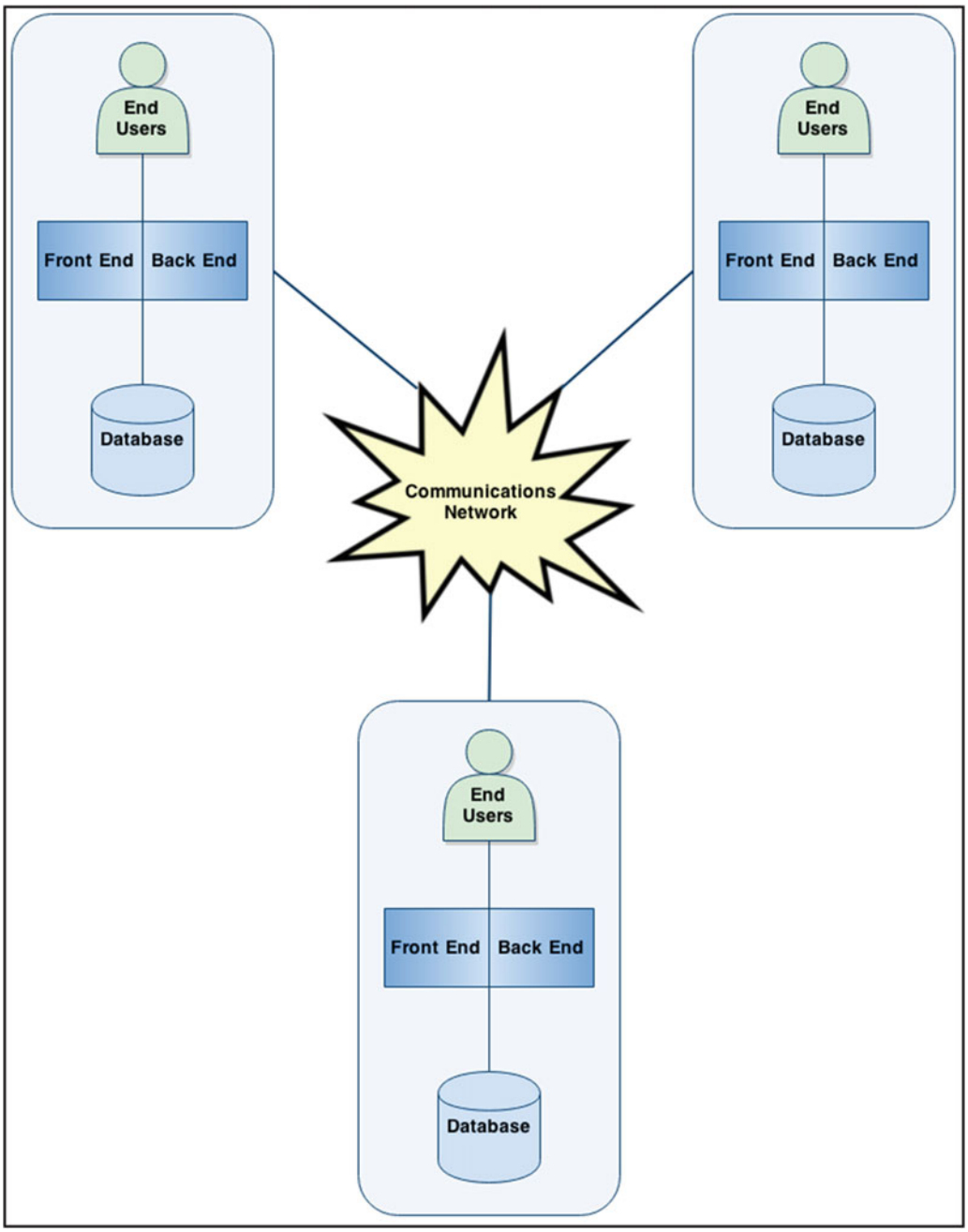


***Figure 2-7.*** *Distributed system where each machine has both a front end and a back end*

Of the three configurations shown, perhaps the one most commonly used is one requiring a single back end and multiple front ends (Figure 2-6). This is so because it provides more flexibility and sophistication than the simpler configuration of Figure 2-5, and less cost and overhead than the more complex configuration of Figure 2-7. The database resides on a single database server (the source) and is accessible from multiple clients.

■ **Discuss** Study the Figures 2-5 through 2-7 and answer the following question: What are some of the advantages of using distributive processing in a DBS environment?

# 2.8 Database Management System Classifications

Database systems may be classified based on the data model supported, number of users facilitated, site configuration, or purpose. Let's clarify each approach.

## 2.8.1 Classification Based on Data Models

Classification based on data models can take a number of traditional approaches, and some that are more contemporary in nature. The traditional approaches are generally looked at as being obsolete, whereas the contemporary approaches represent those in active growth and development in the software market.

The traditional approaches are the hierarchical model, the network model, and the inverted-list model; these were mentioned in Chapter 1. The hierarchical model was developed by IBM in the 1960s, and was epitomized by a product called RAMIS (to be further clarified in Chapter 3). The network model was also developed in the 1960s, inspired by a project that was initiated at General Electric. One of the crowning products of this initiative was the CODASYL system (also clarified in Chapter 3). The inverted-list model prescribed the construction and management of indexes that would map data to their storage locations in database files. This approach has had a significant impact on the modern configuration of DBMSs.

Two dominant contemporary approaches are the relational model and the object-oriented model, with the former occupying most of the market share. Most of the remainder of the course is based on the relational model, except for Chapter 18, which discusses object databases. Prominent relational DBMS (RDBMS) suites include Oracle, DB2, MySQL, SQL Server, Informix, PostGreSQL, and Sybase (recently incorporated into SAP). Object databases seldom appear on their own; they are typically bundled with RDBMS suites and marketed as universal DBMS suites. The leading products such as Oracle, DB2, and Informix fall in this category.

In addition to the relational model and the object-oriented model, other emerging approaches such as the Hadoop framework, the EAV model, and NoSQL are gaining increased attention (review section 1.4).

## 2.8.2 Classification Based on Number of Users

Software engineering firms that market DBMS suites typically provide a single/limited user version for evaluation purposes or limited work. For example, both Oracle and IBM market an Express Edition of their respective DBMS products. In contrast to the limited user edition, the related software engineering firms typically market their complete DBMS suite (allowing multiple users) as an Enterprise Edition. Additionally, there may be different alternatives between these two extremes. Division E of the text provides examples from four of the leading DBMS suites.

## 2.8.3 Classification Based on Site Configuration

A software engineering firm may market its DBMS suite either as a centralized DBS or a distributed DBS. In a centralized system, the DBMS software and the database reside on a single machine, and all other nodes in the network connect to it. Due to the increase in telecommunication bandwidth, centralized databases have become widespread, and will suffice for many small- and medium-sized DBS environments.

In a distributed DBS, the DBMS software and the database are distributed over multiple database servers at different sites in a computer network (review the previous section). The network may be a local area network (LAN), metropolitan area network (MAN), or a wide area network (WAN). This allows for more flexibility and fault tolerance in the system, and is applicable for large, complex organizations with multiple branches across state and/or national borders. If each participating site runs the same DBMS, the system is said to be homogeneous; otherwise, it is heterogeneous. This topic is discussed more fully in Chapter 17.

An emergent trend that has taken off in the marketplace is for the software engineering companies to provide a *cloud implementation* of the DBMS—a facility to configure and store a database in a storage space on the World-Wide Web (WWW) in a manner that is seamless and less tedious than previously possible. This alternative provides additional improved benefits such as inexpensive offshore backup, flexibility, and reliability. Indeed, the latest version of the Oracle DBMS, Oracle 12C, is so named to emphasize the "C" for cloud. Other competitors have pursued alternate strategies geared at incorporating this cloud technology into their DBMS product offerings.

### 2.8.4 Classification Based on Database Purpose

The final classification of databases that we will discuss relates to the purpose of the database. A database may be described as general purpose if it stores various forms of data as defined by its constituent data files. Alternately, the database may be classified as specific purpose if it is configured to store data of a specific type. For instance, databases used in computer-aided design (CAD) and/or computer-aided manufacturing (CAM) are said to be specific purpose databases.

A database may also be classified based on the nature of processing for which it has been configured. From this perspective, two broad categories are *online transaction processing* (OLTP) databases and *data warehouses*.

- An OLTP database (also called an *operational* database) is a database that facilitates a large volume of concurrent transactions. Such databases are widely used in general business operations (for example, banking, manufacturing, insurance, marketing, etc.).
- A data warehouse is a read-only database (except when it is being updated) that has been configured to store summarized information drawn from various operational databases. Data warehouses are also referred to as *online analytic processing* (OLAP) databases. They are updated by a special process called the *extract-transform-load* (ETL) process; this is clarified in Chapter 19.

The rest of this course will focus on operational databases, except for Chapter 19, which provides an introduction to data warehousing. However, please note that a full treatment of data warehousing is beyond the scope of this course.

## 2.9 Summary and Concluding Remarks

Here is a summary of what has been covered in this chapter:

- A database system can be construed as having three levels of architecture: the external, the conceptual, and the internal. These levels are seamlessly interlinked by the DBMS.
- The external level constitutes all the external views that end users have of the database.
- The conceptual level relates to the logical structure of the database.
- The internal level relates to the physical structure of the files making up the database.

- In a DBS environment, the personnel may consist of any appropriate combination of the following roles: database administrator, data architect, tools expert, application programmer, user liaison specialist, and network and infrastructure specialist. The DBA is the official responsible for the planning, construction, implementation, and administration of the database. The other roles provide critical supportive responsibilities.
- The DBMS is the software that facilitates creation and administration of the database.
- A database system can be construed as being comprised of a front-end system and a back-end system. The back-end system relates to the actual creation and administration of the database, the inner workings of the system, processing requests, memory management, and I/O management. The front-end system relates to the creation and administration of the user interface through which end users access the system.
- By applying the principle of separating front end from back end, we can conceive of various database architectures.
- Database systems may be classified based on data model supported (traditional, relational, or object-oriented), number of users facilitated (single user or multi-user), site configuration (centralized or distributed), or purpose (operational or data warehouse).

With this background, you are now ready to move ahead and learn more about the relational database model. You will learn the foundations of the model, and why it is so important.

## 2.10 Review Questions

For review, answer the following questions:

1. With the use of a diagram, explain the different levels of architecture of a database system.
2. Explain the acronyms DSL, DML, DCL, and DDL. How are they related?
3. Identify the main personnel in a DBS environment. What are the primary functions of each role?
4. What are the main functions of the DBMS?
5. With the aid of an appropriate diagram, explain how the DBMS ensures that requests from end users are satisfactorily addressed.
6. Discuss the functional components of a DBMS. Use an appropriate diagram to illustrate.
7. Explain the concept of front-end and back-end systems, and show how they add flexibility to the implementation of distributed database systems.

## 2.11 References and/or Recommended Readings

[Comer 2015] Comer, Douglas. 2015. *Computer and Communication Networks* 2nd ed. Boston: Pearson.

[Connolly 2015] Connolly, Thomas and Carolyn Begg. 2015. *Database Systems: A Practical Approach to Design, Implementation and Management* 6th ed. Boston: Pearson. See Chapters 2 and 3.

[Coronel 2015] Coronel, Carlos and Steven Morris. 2015. *Database Systems: Design, Implementation & Management* 11th ed. Boston: Cengage Learning. See Chapters 1 and 2.

[Date 2004] Date, Christopher J. 2004. *Introduction to Database Systems* 8h ed. Menlo Park, CA: Addison-Wesley. See Chapter 2.

[Elmasri, 2011] Elmasri, Ramez and Shamkant B. Navathe. 2011. *Fundamentals of Database Systems* 6th ed. Boston: Pearson. See Chapter 2.

[Garcia-Molina 2009] Garcia-Molina, Hector, Jeffrey Ullman and Jennifer Widom. 2009. *Database Systems: The Complete Book* 2nd ed. Boston: Pearson. See Chapter 1.

[Hoffer 2013] Hoffer, Jeffrey A., Ramesh Venkataraman, and Heikki Topi. 2013. *Modern Database Management* 11th ed. Boston: Pearson. See Chapters 1 and 2.

[Thomas 2014] Thomas, Biju, Gavin Powell, Robert Freeman, and Charles Pack. 2014. *Oracle Certified Professional on 12C Certification Kit*. Indianapolis, In: John Wiley.

[Ullman 2008] Ullman, Jeffrey D., and Jennifer Widom. 2008. *A First Course in Database Systems* 3rd ed. Boston: Pearson. See Chapter 1.