Навчальна дисципліна

БАЗИ ДАНИХ

Лектор - к.т.н., доцент

Баклан Ігор Всеволодович

Site: baklaniv.at.ua

E-mail: iaa@ukr.net

2016-2017

Лекція №2.

Середовище систем баз даних

- Levels of Architecture
- Inter-level Mappings
- Database Administrator
- Database Management System
- Components of the DBMS Suite
- Front-end and Back-end Perspectives
- Database System Architecture
- 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.

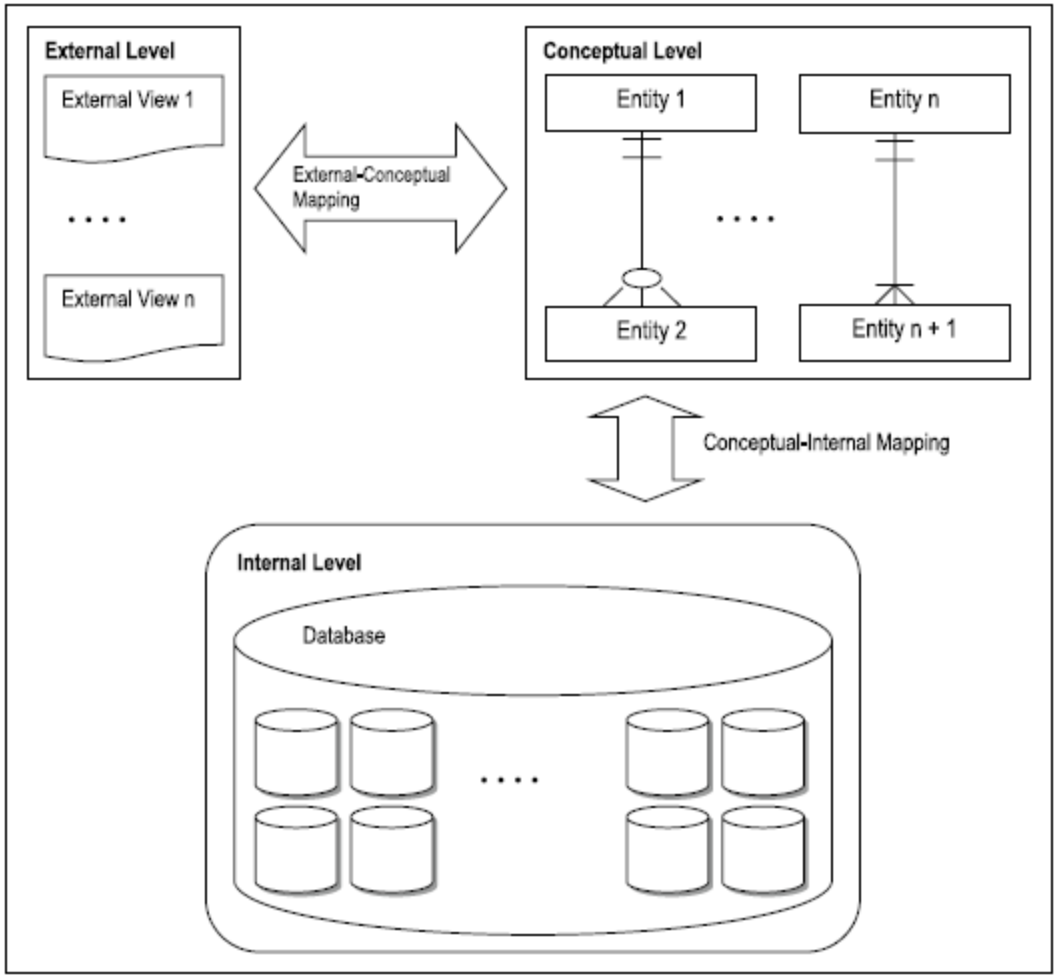

Figure 2-1. *Levels of DBS Architecture*

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. Other examples of data sub-languages are QUEL and KQL (*knowledge query language*). These languages 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 COBOL, C, C++, Java, Pascal, RPG-400, etc.

Typically, the sublanguage 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.

The above-mentioned facilities allow users 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 also referred to as the logical or community user view. It is defined by means of the *conceptual schema*, which includes definition of each of the various types of conceptual records.

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 that 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.

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 The Database Administrator

The database administrator (DBA) has overall responsibility for the control of the system at the technical level. Some of the functions of the DBA include:

- 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 many 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 company database(s) on an on-going basis, usually employ the services of one or more DBAs. Because of the importance of having reliable databases, DBAs are among the highest paid information technology (IT) professionals.

2.4 The Database Management System

The database management system (DBMS) is the software that facilitates management of the database. When a user issues a request via some DSL (for example SQL), it is the DBMS that interprets the request, executes the appropriate instructions and responds to the user.

Functions of the DBMS include the following:

- Data definition (relation, 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), data recovery 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 updates 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 sub-schema language);
2. DBMS looks at **Program-A** sub-schema, schema and physical description (these information are stored in tables);
3. DBMS determines which files must be accessed, which records are needed and how access is done;
4. DBMS issues instruction(s) (reads or writes) to the operating system;
5. 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.

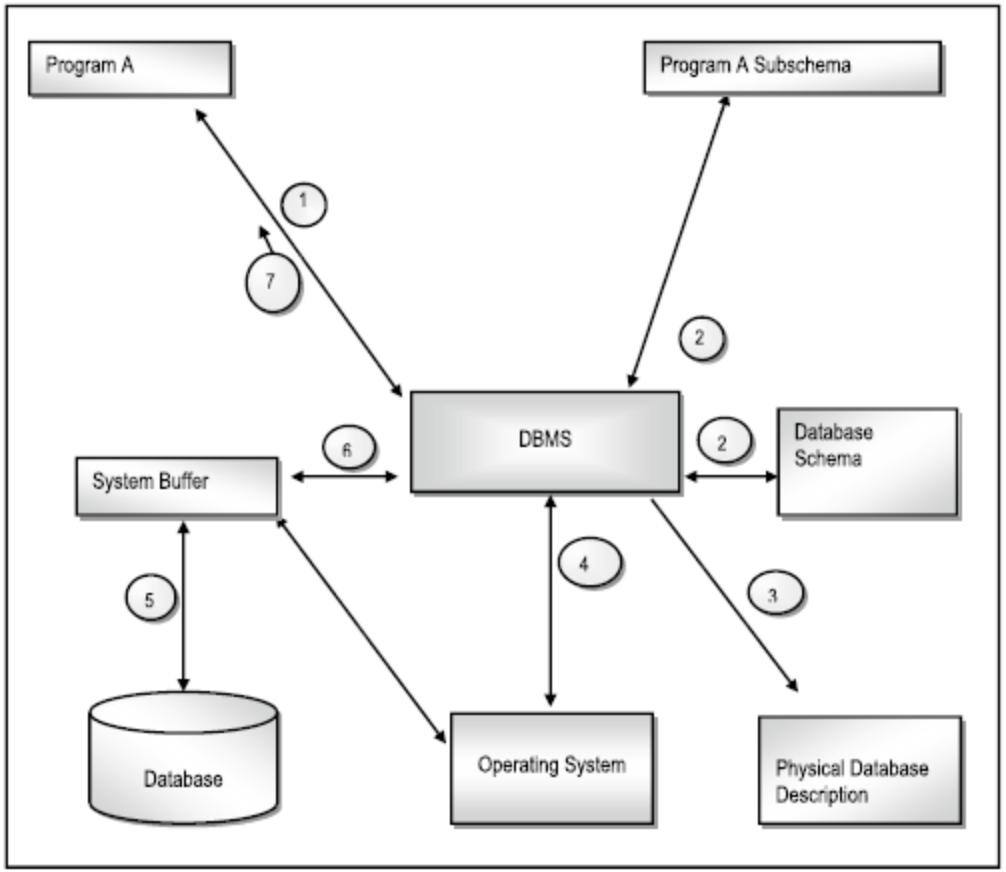

Figure 2-2. *Steps Involved When Application Programs Access a Database*

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.

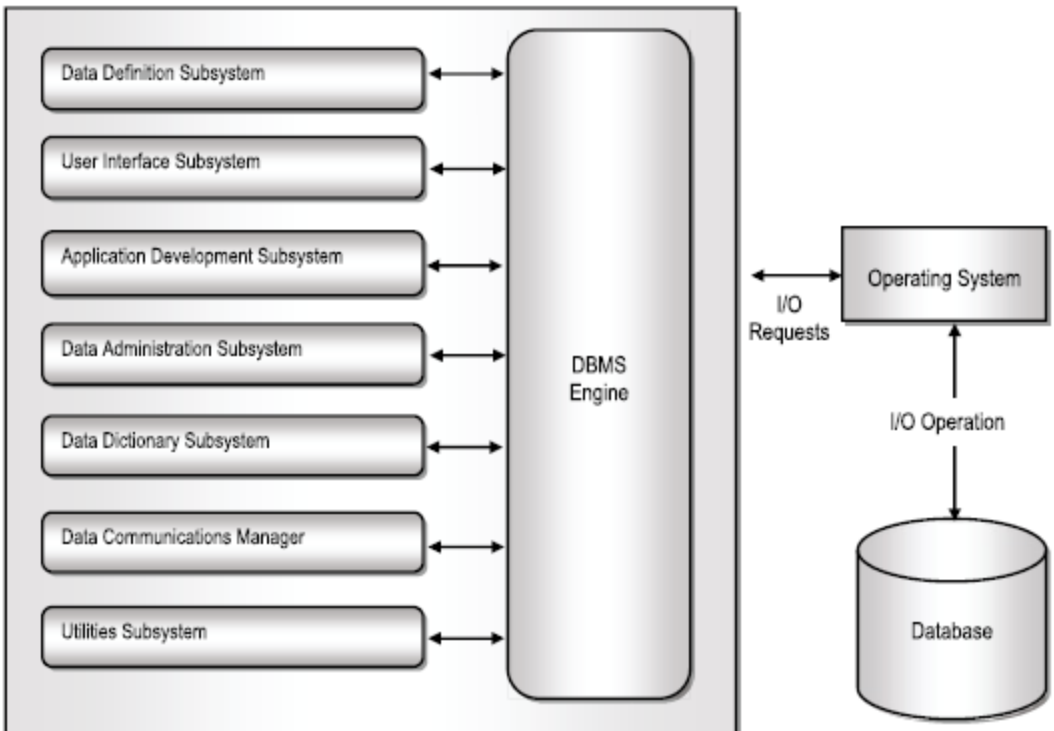

Figure 2-3. *Functional Components of a DBMS*

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. I/O 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 Definition Tools 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 (insert, remove, update, find, query etc.) of data. The DML usually includes a query language. 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. 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 *natural language* interface.

The user interface may also include a DBMS-specific programming language (e.g. FoxPro, Scalable Application Language, and Oracle's PL/SQL). These languages 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.

As an example, suppose that a file, **Student** has fields {ID#, SName, FName, Status, DOB,...} for each record. Two possible SQL queries could be:

```
SELECT ID#, SNAME, FNAME FROM STUDENT WHERE SNAME = "BELL";

SELECT ID#, SNAME, DOB FROM STUDENT WHERE DOB >= 19660101;
```

A more detailed study of SQL will be covered later in the course.

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).

Other facilities that may be included such as:

- Report writer
- Project manager
- Menu builder
- 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

Also called the *system catalog* in many systems, the *data dictionary* (DD) contains information on the database structure as well relationships among database objects. 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 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
- Other utilities (that might have been) developed by application programmers

2.6 The Front-end and Back-end Perspectives

A DBS can be perceived as a simple two-part structure:

- The Front-end consists of end users, applications and a programming interface.
- The Back-end consists of the actual DBMS and the database.

The front-end system may be on a different machine from the back-end system and the two connected by a communication network. For example, we may have a front-end system in Delphi or Java NetBeans, and a back-end system in Oracle, Sybase, MySQL, or DB2. Figure 2-4 illustrates.

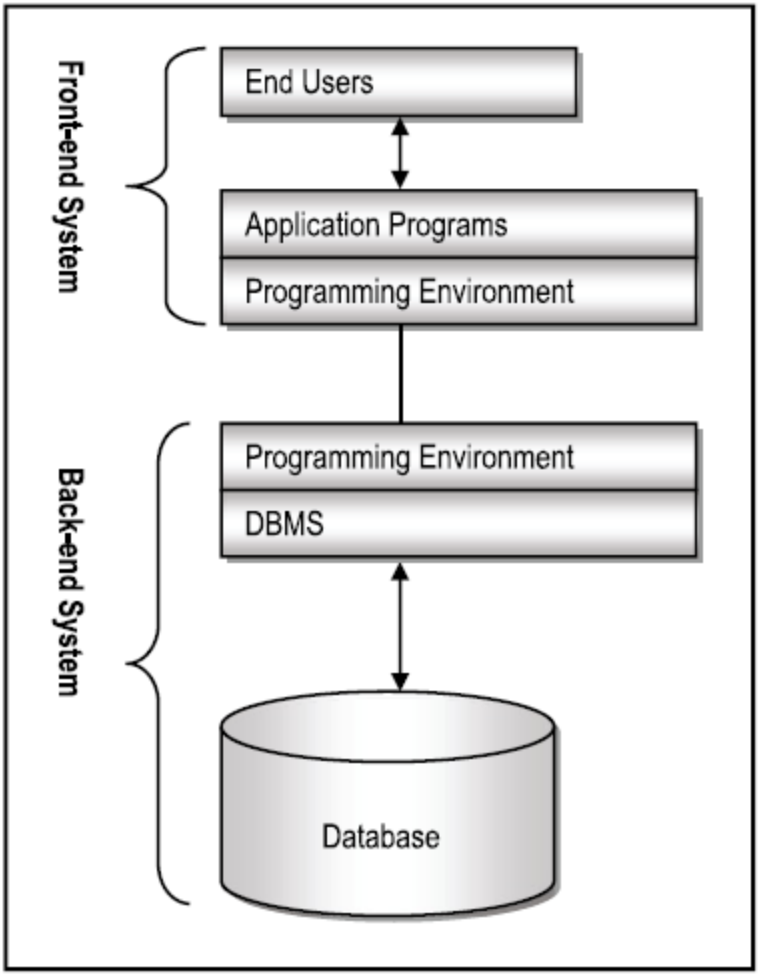

Figure 2-4. Front-end / Back-end Perspective

2.7 Database System Architecture

There may be added benefits of using different machines for the back-end and front-end system. Figures 2-5 – 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).

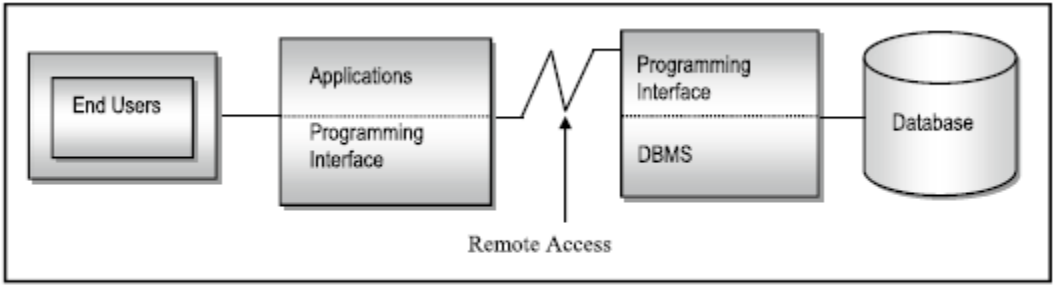

Figure 2-5. Back-end and Front-end Running on Different Machines

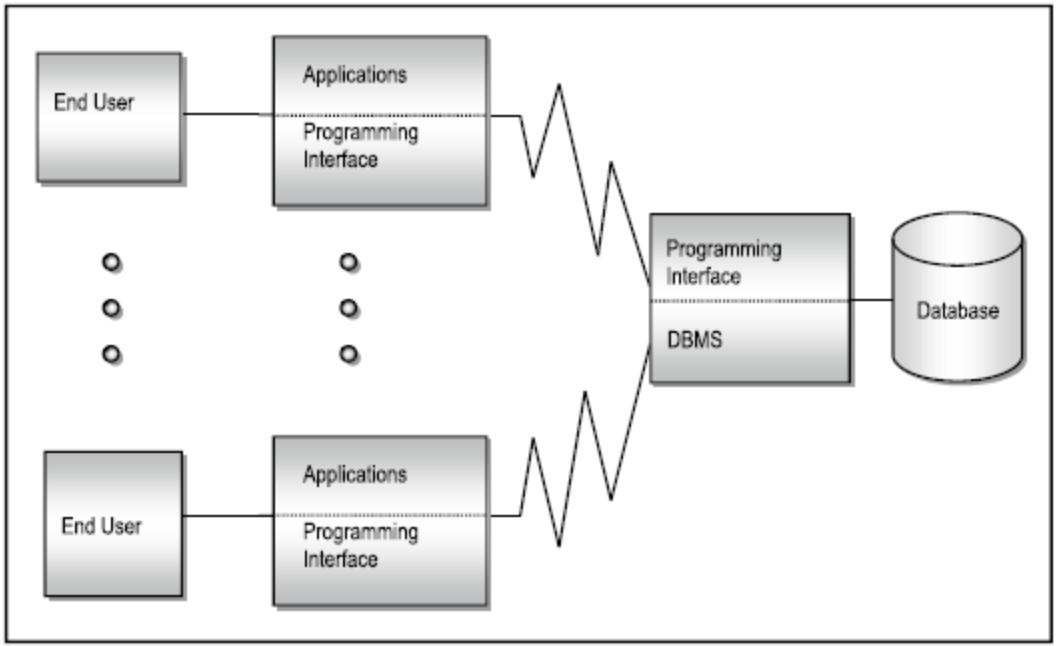

Figure 2-6. *One Back-end, Multiple Front-ends*

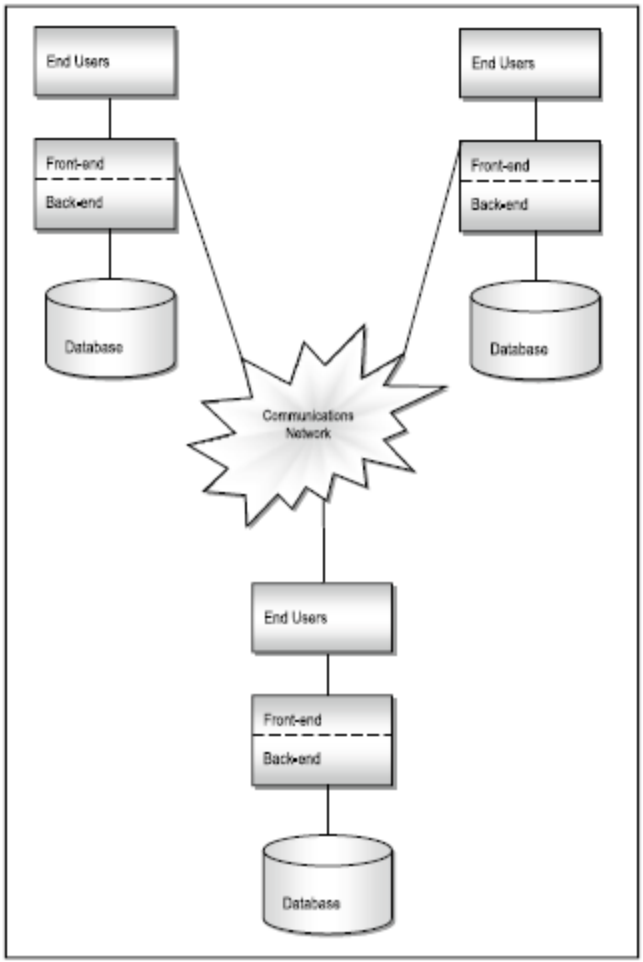

Figure 2-7. *Distributed System Where Each Machine has both Front-end and Back-end*

2.8 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.
- The DBA is the official responsible for the planning, construction, Implementation and administration of the database.
- 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 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.

With this background, we 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.