**UNIT-1 – Chapter 1**

**Introduction:**

As the name suggests, the database management system consists of two parts. They are:

1. Database and
2. Management System

**What is a Database?**

To find out what database is, we have to start from data, which is the basic building block of any DBMS.

**Data**: Facts, figures, statistics etc. having no particular meaning (e.g. 1, ABC, 19 etc).

In a database, data is organized strictly in row and column format. The rows are called **Tuple** or **Record**. The data items within one row may belong to different data types. On the other hand, the columns are often called **Domain** or **Attribute**. All the data items within a single attribute are of the same data type.

**What is Management System?**

A **database-management system** (DBMS) is a collection of interrelated data and a set of programs to access those data. The collection of data, usually referred to as the **database**, contains information relevant to an enterprise. The primary goal of a DBMS is to provide a way to store and retrieve database information that is both *convenient* and *efficient*.

The management system is important because without the existence of some kind of rules and regulations it is not possible to maintain the database. Database systems are designed to manage large bodies of information. Management of data involves both defining structures for storage of information and providing mechanisms for the manipulation of information.

## DATABASE APPLICATIONS – DBMS:

Applications where we use Database Management Systems are:

* Telecom: There is a database to keeps track of the information regarding calls made, network usage, customer details etc.
* Industry: In a manufacturing unit, warehouse or distribution centre, each one needs a database to keep the records of ins and outs.
* Banking System: For storing customer info, tracking day to day credit and debit transactions, generating bank statements etc.
* Sales: To store customer information, production information and invoice details.
* Airlines: To travel though airlines, we make early reservations; this reservation information along with flight schedule is stored in database.
* Education sector: Database systems are frequently used in schools and colleges to store and retrieve the data regarding student details, staff details, course details, exam details, payroll data, attendance details, fees details etc.

## Purpose of Database Systems

This typical **file-processing system** is supported by a conventional operating system. The system stores permanent records in various files, and it needs different application programs to extract records from, and add records to, the appropriate files. Before database management systems (DBMSs) were introduced, organizations usually stored information in such systems.

Keeping organizational information in a file processing system has a number of major disadvantages:

**Data redundancy and inconsistency**. Since different programmers create the files and application programs over a long period, the various files are likely to have different structures and the programs may be written in several programming languages. Moreover, the same information may be duplicated in several places (files). This redundancy leads to higher storage and access cost. In addition, it may lead to **data inconsistency**.

**Difficulty in accessing data:** conventional file-processing environments do not allow needed data to be retrieved in a convenient and efficient manner. More responsive data-retrieval systems are required for general use.

**Data isolation**. Because data are scattered in various files, and files may be in different formats, writing new application programs to retrieve the appropriate data is difficult.

**Integrity problems**. The data values stored in the database must satisfy certain types of consistency constraints**.** Developers enforce these constraints in the system by adding appropriate code in the various application programs. However, when new constraints are added, it is difficult to change the programs to enforce them.

**Atomicity problems**. A computer system may subject to failure. In many applications, it is crucial that, if a failure occurs, the data be restored to the consistent state that existed prior to the failure. Ex: The funds transfer must be *atomic*—it must happen in its entirety or not at all. It is difficult to ensure atomicity in a conventional file-processing system.

**Concurrent-access anomalies**. For the sake of overall performance of the system and faster response, many systems allow multiple users to update the data simultaneously. In such an environment, interaction of concurrent updates is possible and may result in inconsistent data.

**Security problems**. Not every user of the database system should be able to access all the data. But, since application programs are added to the file-processing system in an ad hoc manner, enforcing such security constraints is difficult.

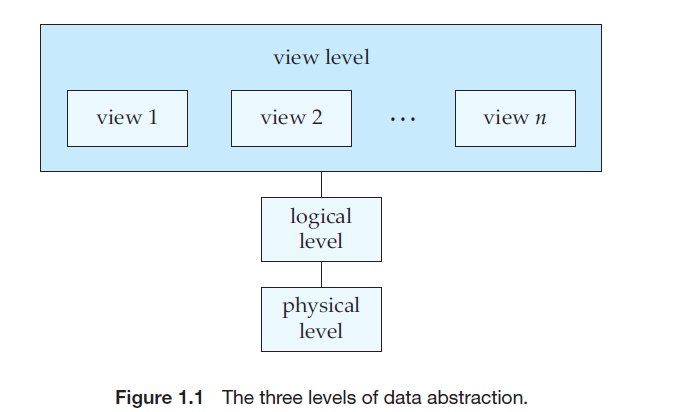
**View of Data**

## 

A database system is a collection of interrelated data and a set of programs that allow users to access and modify these data. A major purpose of a database system is to provide users with an *abstract* view of the data. That is, the system hides certain details of how the data are stored and maintained.

**Data Abstraction:**

For the system to be usable, it must retrieve data efficiently. Since many database-system users are not computer trained, developers hide the complexity from users through several levels of abstraction, to simplify users’ interactions with the system.



**Physical level (or Internal View:** The lowest level of abstraction describes *how* the data are actually stored. The physical level describes complex low-level data structures in detail.

**Logical level (or Conceptual View)**: The next-higher level of abstraction describes *what* data are stored in the database, and what relationships exist among those data. It describes the entire database in terms of a small number of relatively simple structures.

Although implementation of the simple structures at the logical level may involve complex physical-level structures, the user of the logical level does not need to be aware of this complexity. This is referred to as **physical data independence**.

**View level (or External View):**  The highest level of abstraction describes only part of the entire database. Even though the logical level uses simpler structures, complexity remains because of the variety of information stored in a large database. Many users of the database system do not need all this information; instead, they need to access only a part of the database. The view level of abstraction exists to simplify their interaction with the system.

**Instances and Schemas**

Databases change over time as information is inserted and deleted. The collection of information stored in the database at a particular moment is called an **instance** of the database. The overall design of the database is called the database **schema**. Schemas are changed infrequently, if at all.

Database systems have several schemas, partitioned according to the levels of abstraction. The **physical schema** describes the database design at the physical level, while the **logical schema** describes the database design at the logical level. A database may also have several schemas at the view level, sometimes called **subschemas**, which describe different views of the database. Of these, the logical schema is by far the most important, in terms of its effect on application programs, since programmers construct applications by using the logical schema. The physical schema is hidden beneath the logical schema, and can usually be changed easily without affecting application programs.

**Data Models**

Underlying the structure of a database is the **data model**: a collection of conceptual tools for describing data, data relationships, data semantics, and consistency constraints. The data models can be classified into four different categories:

• **Relational Model**. The relational model uses a collection of tables to represent both data and the relationships among those data. Each table has multiple columns, and each column has a unique name. Tables are also known as **relations**.

Each relation contains records of a particular type. Each record type defines a fixed number of attributes. The columns of the table correspond to the attributes of the record type. The relational data model is the most widely used data model, and a vast majority of current database systems are based on the relational model.

**Entity-Relationship Model**. The entity-relationship (E-R) data model uses a collection of basic objects, called *entities*, and *relationships* among these objects.

An entity is a “thing” or “object” in the real world that is distinguishable from other objects. The entity relationship model is widely used in database design.

**Object-Based Data Model**. Object-oriented programming (especially in Java, C++, or C#) 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, and object identity. The object-relational data model combines features of the object-oriented data model and relational data model.

**Semi-structured Data Model**. The semi-structured data model permits the specification of data where individual data items of the same type may have different sets of attributes. The **Extensible Markup Language (XML)** is widely used to represent semi-structured data.

**Database Languages**

A database system provides a **data-definition language** to specify the database schema and a **data-manipulation language** to express database queries and updates. They simply form parts of a database language SQL.

**Data-Manipulation Language**

A **data-manipulation language (DML)** is a language that enables users to access or manipulate data as organized by the appropriate data model. The types of access are:

* Retrieval of information stored in the database
* Insertion of new information into the database
* Deletion of information from the database
* Modification of information stored in the database

There are basically two types:

* **Procedural DMLs** require a user to specify *what* data are needed and *how* to get those data.
* **Declarative DMLs** (also referred to as **nonprocedural DMLs**) require a user to specify *what* data are needed *without* specifying how to get those data.

A **query** is a statement requesting the retrieval of information. The portion of a DML that involves information retrieval is called a **query language**.

**Data-Definition Language (DDL):**

We specify a database schema by a set of definitions expressed by a special language called a **data-definition language** (**DDL**).

We specify the storage structure and access methods used by the database system by a set of statements in a special type of DDL called a **data storage and definition** language. These statements define the implementation details of the database schemas, which are usually hidden from the users.

The data values stored in the database must satisfy certain **consistency constraints**.

. The DDL provides facilities to specify such constraints. The database system checks these constraints every time the database is updated.

* **Domain Constraints**. A domain of possible values must be associated with every attribute. Declaring an attribute to be of a particular domain acts as a constraint on the values that it can take. Domain constraints are the most elementary form of integrity constraint. They are tested easily by the system whenever a new data item is entered into the database.

* **Referential Integrity**. There are cases where we wish to ensure that a value that appears in one relation for a given set of attributes also appears in a certain set of attributes in another relation (referential integrity). Database modifications can cause violations of referential integrity. When a referential-integrity constraint is violated, the normal procedure is to reject the action that caused the violation.

* **Assertions**. An assertion is any condition that the database must always satisfy. Domain constraints and referential-integrity constraints are special forms of assertions. However, there are many constraints that we cannot express by using only these special forms. For example, “Every department must have at least five courses offered every semester” must be expressed as an assertion. When an assertion is created, the system tests it for validity. If the assertion is valid, then any future modification to the database is allowed only if it does not cause that assertion to be violated.

* **Authorization**. We may want to differentiate among the users as far as the type of access they are permitted on various data values in the database. These differentiations are expressed in terms of **authorization**, the most common being: **read authorization**, which allows reading, but not modification, of data; **insert authorization**, which allows insertion of new data, but not modification of existing data; **update authorization**, which allows modification, but not deletion, of data; and **delete authorization**, which allows deletion of data. We may assign the user all, none, or a combination of these types of authorization.

**Relation Databases:**

A relational database is based on the relational model and uses a collection of tables to represent both data and the relationships among those data.

**Tables**

* Each table has multiple columns and each column has a unique name.
* The relational model is an example of a record-based model.
* Record-based models are so named because the database is structured in fixed-format records of several types. Each table contains records of a particular type.
* Each record type defines a fixed number of fields, or attributes.
* The columns of the table correspond to the attributes of the record type.
* It is possible to create schemas in the relational model that have problems such as unnecessarily duplicated information.

**Data-Manipulation Language:**

The SQL query language is nonprocedural. A query takes as input several tables and always returns a single table. Here is an example of an SQL query that finds the names of all instructors in the History department:

**select** *instructor*.*name*

**from** *instructor*

**where** *instructor*.*dept name* = ’History’;

The query specifies that those rows from the table *instructor* where the *dep\_name* is History must be retrieved, and the *name* attribute of these rows must be displayed.

Queries may involve information from more than one table. For instance, the following query finds the instructor ID and department name of all instructors associated with a department with budget of greater than $95,000.

**select** *instructor*.*ID*, *department*.*dept\_name*

**from** *instructor*, *department*

**where** *instructor*.*dept name*= *department*.*dept name* **and** *department*.*budget >* 95000;

**Data-Definition Language:**

SQL provides a rich DDL that allows one to define tables, integrity constraints, assertions, etc.

For instance, the following SQL DDL statement defines the *department* table:

**create table** *department*

(*dept name* **char** (20),

*building* **char** (15),

*budget* **numeric** (12,2));

Execution of the above DDL statement creates the *department* table with three columns: *dept name*, *building*, and *budget*, each of which has a specific data type associated with it.

**Database Access from Application Programs:**

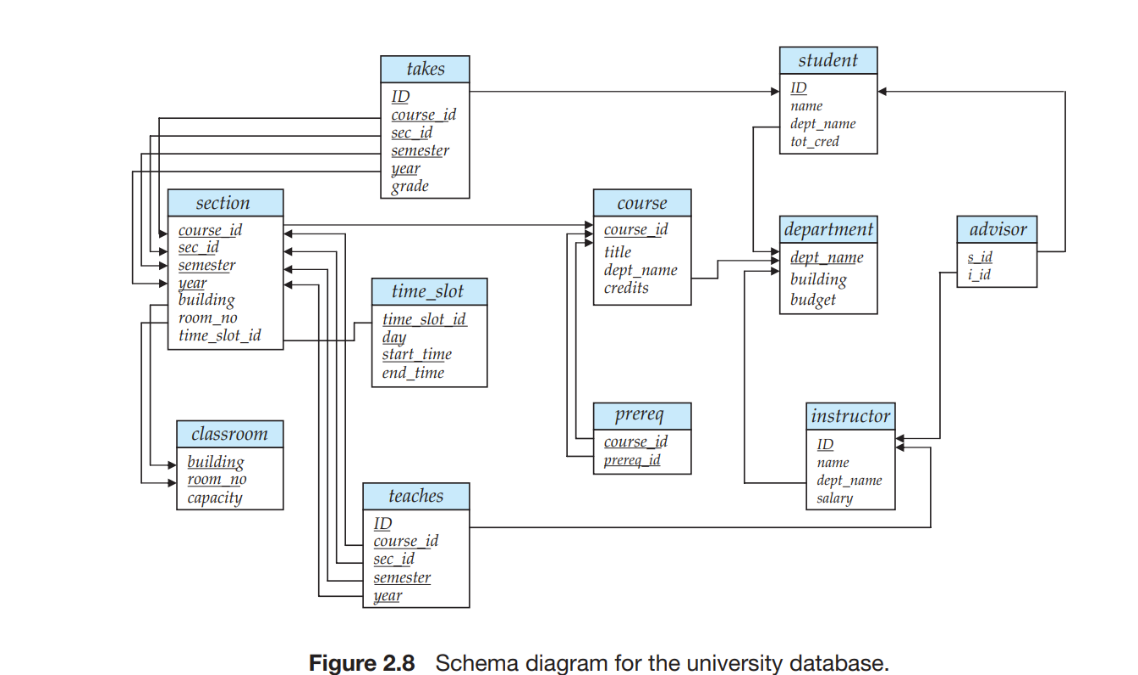
There are some computations that are possible using a general-purpose programming language but are not possible using SQL. SQL also does not support actions such as input from users, output to displays, or communication over the network. Such computations and actions must be written in a *host* language, such as C, C++, or Java, with embedded SQL queries that access the data in the database. Application programs are programs that are used to interact with the database in this fashion. To access the database, DML statements need to be executed from the host language. There are two ways to do this:

* By providing an application program interface that can be used to send DML and DDL statements to the database and retrieve the results.
* The Open Database Connectivity (ODBC) standard for use with the C language is a commonly used application program interface standard. The Java Database Connectivity (JDBC) standard provides corresponding features to the Java language.

**Data Base Design**:

* Database systems are designed to manage large bodies of information.
* They are part of the operation of some enterprise whose end product may be information from the database.
* The initial phase of database design, then, is to characterize fully the data needs of the prospective database users.
* The database designer needs to interact extensively with domain experts and users to carry out this task.
* The outcome of this phase is a specification of user requirements.
* Next, the designer chooses a data model, and by applying the concepts of the chosen data model, translates these requirements into a conceptual schema of the database.
* The schema developed at this conceptual-design phase provides a detailed overview of the enterprise.
* The designer reviews the schema to confirm that all data requirements are indeed satisfied and are not in conflict with one another.

**Database Designw for a University Organization:**



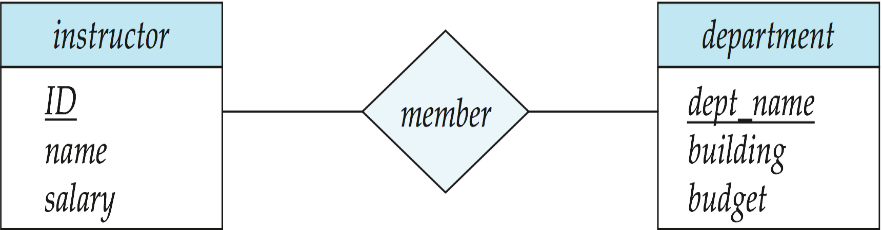
In the relational model, the conceptual-design process involves decisions on what attributes we want to capture in the database and how to group these attributes to form the various tables

* + Business decision – What attributes should we record in the database?
  + Computer Science decision – What relation schemas should we have and how should the attributes be distributed among the various relation schemas?

**The Entity-Relationship Model:**

* Models an enterprise as a collection of *entities* and *relationships*
  + Entity: a “thing” or “object” in the enterprise that is distinguishable from other objects
    - Described by a set of *attributes*
  + Relationship: an association among several entities

Represented diagrammatically by an *entity-relationship diagram*

****

**Database Administrators and Database Users**

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

**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.

**Naive users** are unsophisticated users who interact with the system by invoking one of the application programs that have been written previously.

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

**Application programmers** They are interacting with the database by writing application programs using PHP, .Net or Java.

**Sophisticated users** interact with the system without writing programs. Instead, they form their requests in a database query language. They submit each such query to a **query processor**. 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.

**Database Administrator** Person who has such central control over the system

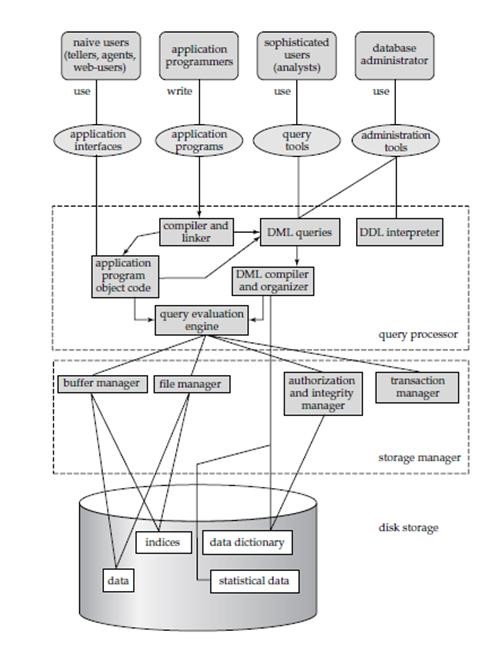
The functions of a DBA include:

• Schema definition.

• Storage structure and access-method definition.

* Schema and physical-organization modification
* Granting of authorization for data access.
* Routine maintenance:
* Periodically backing up the database, either onto tapes or onto remote servers, to prevent loss of data in case of disasters.
* 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.

**Database Architecture:**



**Figure 1.3: System Structure**

A database system is partitioned into modules that deal with each of the responsibilities of the overall system. The functional components of a database system are: Storage manager, Query processor

* The storage manager is important because databases typically require a large amount of storage space.
* The query processor is important because it helps the database system to simplify and facilitate access to data.

**Storage manager: It** is the component of that provides the interface between the low-level data stored in the database and the application programs and queries submitted to the system. The storage manager is responsible to the following tasks:

* Interaction with the file manager
* Efficient storing, retrieving and updating of data.
* The storage manager translates the various DML statements into low-level file-system commands.

**Data storage and querying:**

The storage manager components include:

* **Authorization and integrity manager**, which tests for the satisfaction of integrity constraints and checks the authority of users to access data.
* **Transaction manager**, which ensures that the database remains in a consistent (correct) state despite system failures, and that concurrent transaction executions proceed without conflicting.
* **File manager**, which manages the allocation of space on disk storage and the data structures used to represent information stored on disk.
* **Buffer manager**, which is responsible for fetching data from disk storage into main memory, and deciding what data to cache in main memory.
* The buffer manager is a critical part of the database system, since it enables the database to handle data sizes that are much larger than the size of main memory.
* The storage manager implements several data structures as part of the physical system implementation:
* **Data files**, which store the database itself.
* **Data dictionary**, which stores metadata about the structure of the database, in particular the schema of the database.
* **Indices**, which can provide fast access to data items. Like the index in this textbook, a database index provides pointers to those data items that hold a particular value.

**The Query Processor**

The query processor components include:

• **DDL interpreter**, which interprets DDL statements and records the definitions in the data dictionary.

• **DML compiler**, which translates DML statements in a query language into an evaluation plan consisting of low-level instructions that the query evaluation engine understands.

The DML compiler also performs **query optimization**;

* **Query evaluation engine**, which executes low-level instructions generated by the DML compiler.

**Transaction Management**

A **transaction** is a collection of operations that performs a single logical function in a database application.

* Atomicity : This all-or-none requirement
* Consistency : the value of the sum of the balances of *A* and *B* must be preserved.
* Isolation: all the transactions will be carried out and executed as if it is the only transaction in the system.
* Durability: after the successful execution of a funds transfer, the new values of the balances of accounts *A* and *B* must persist, despite the possibility of system failure.

**Transaction-management component** ensures that the database remains in

a consistent (correct) state despite system failures (e.g., power failures and operating system crashes) and transaction failures.

The **transaction manager** consists of the

* The recovery manager.
* Concurrency-control manager

**Recovery Manager:** Ensuring the atomicity and durability properties is the responsibility of the database system itself—specifically, of the **recovery manager.**

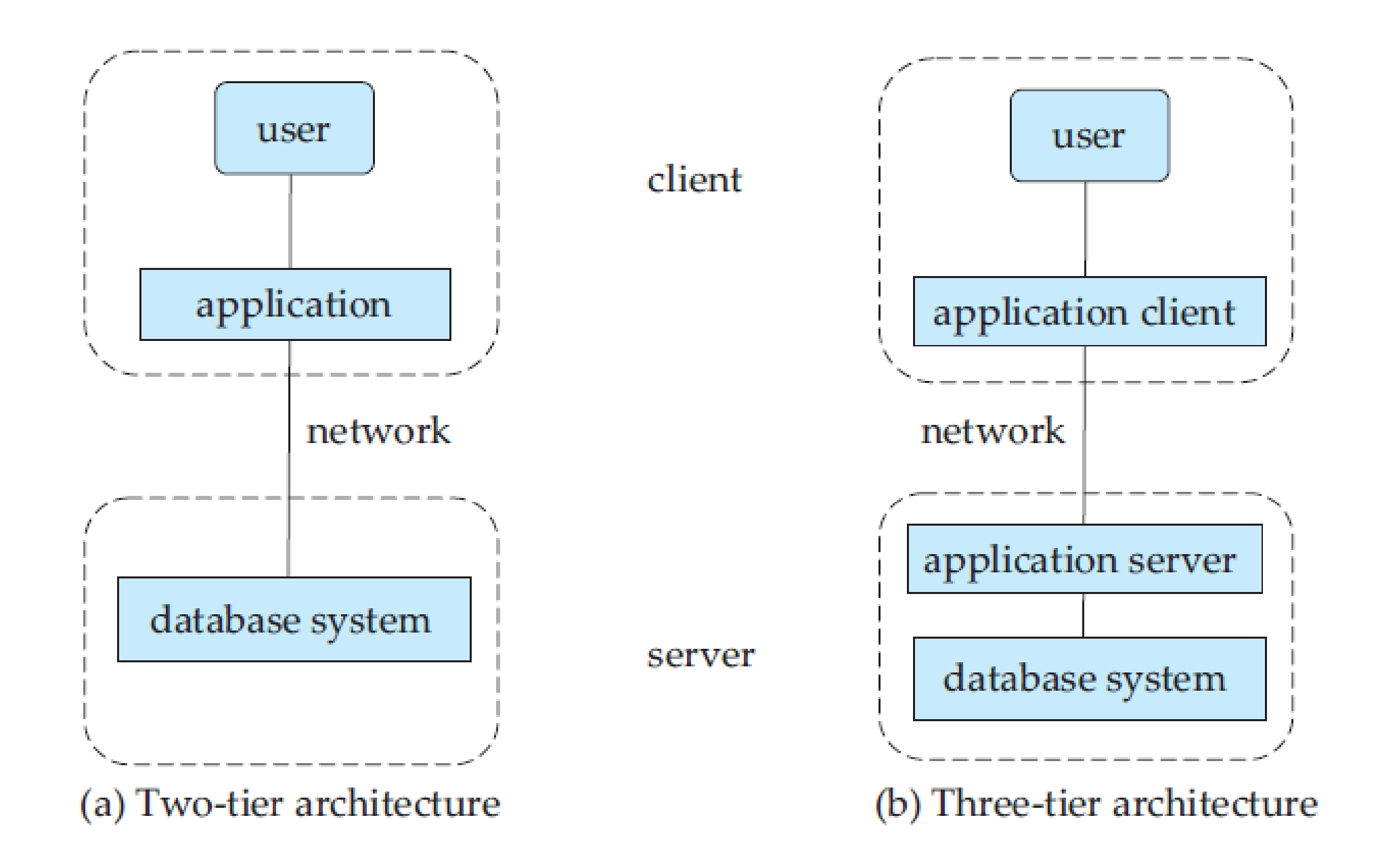
**Concurrency-control manager** controls the interaction among the concurrent transactions, to ensure the consistency of the database.

**Database Architecture:**

Database applications are usually partitioned into two or three parts, as in Figure 1.4. In a two-tier architecture, the application resides at the client machine, where it invokes database system functionality at the server machine through 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.



**Figure 1.4: Two-tier and three-tier architectures.**