DATABASE SYSTEM DEVELOPMENT LIFECYCLE

The stages of the database system development lifecycle are shown in Figure 10.1.

In the following sections we describe the main activities associated with each stage of the database system development lifecycle in more detail.

1. Database planning

Database planning must be integrated with the overall IS strategy of the organization. There are three main issues involved in formulating an IS strategy, which are:

- identification of enterprise plans and goals with subsequent determination of information system's needs;
- evaluation of current information systems to determine existing strengths and weaknesses;
- appraisal of IT opportunities that might yield competitive advantage.

An important first step in database planning is to clearly define the **mission statement** for the database system.

The mission statement defines the major aims of the database system. Those driving the database project within the organization (such as the Director and/or owner) normally define the mission statement.

A mission statement helps to clarify the purpose of the database system and provide a clearer path towards the efficient and effective creation of the required database system. Once the mission statement is defined, the next activity involves identifying the **mission objectives**. Each mission objective should identify a particular task that the database system must support. The assumption is that if the database system sup-ports the mission objectives, then the mission statement should be met.

The mission statement and objectives may be accompanied by some additional information that specifies in general terms the work to be done, the resources with which to do it, and the money to pay for it all.

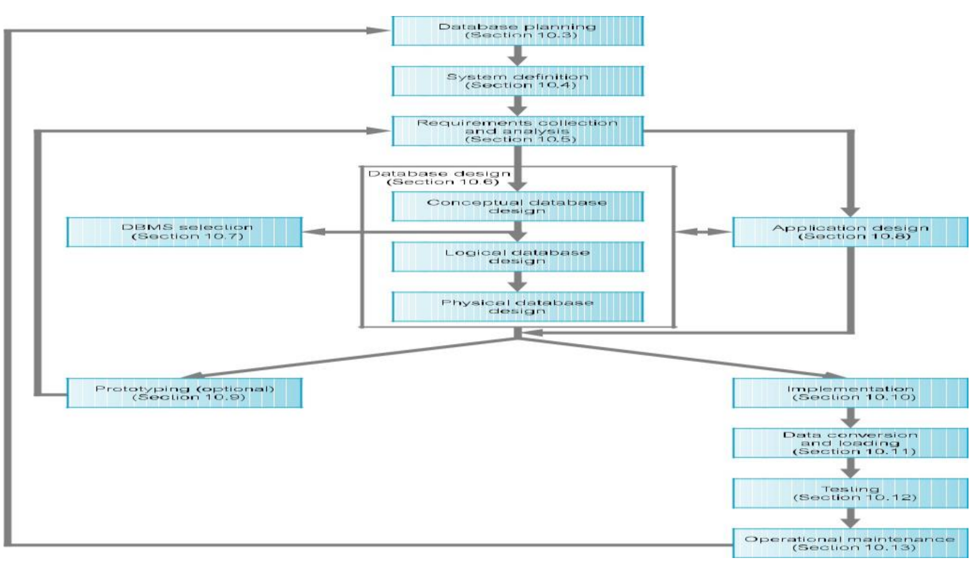

Fig 10.1 The stages of the database system development lifecycle

2.System Definition

Describes the scope and boundaries of the database system and the major user views.

Before attempting to design a database system, it is essential that we first identify the boundaries of the system that we are investigating and how it interfaces with other parts of the organization's information system. It is important that we include within our system boundaries not only the current users and application areas, but also future users and applications.

User Views

Defines what is required of a database system from the perspective of a particular job role (such as Manager or Supervisor) or enterprise application area (such as marketing, personnel, or stock control).

A database system may have one or more user views. Identifying user views is an important aspect of developing a database system because it helps to ensure that no major users of the database are forgotten when developing the requirements for the new database system. User views are also particularly helpful in the development of a relatively complex database system by allowing the requirements to be broken down into manageable pieces.

A user view defines what is required of a database system in terms of the data to be held and the transactions to be performed on the data (in other words, what the users will do with the data). The requirements of a user view may be distinct to that view or overlap with other views. Figure 10.2 is a diagrammatic representation of a data-base system with multiple user views (denoted user view 1 to 6). Note that whereas user views (1, 2, and 3) and (5 and 6) have overlapping requirements (shown as hatched areas), user view 4 has distinct requirements.

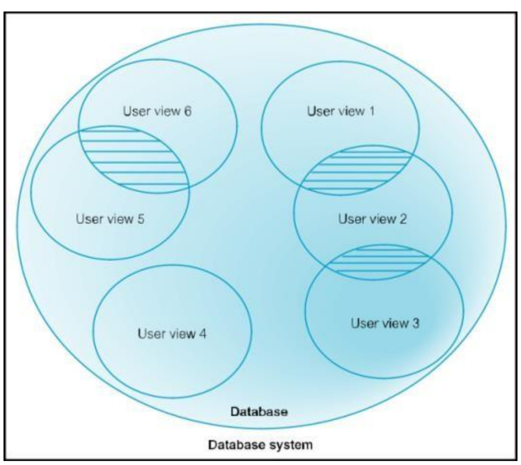

Figure 10.2 Representation of a database system with multiple user views: user views (1, 2, and 3) and (5 and 6) have overlapping requirements (shown as hatched areas), whereas user view

4 has distinct requirements.

Requirements Collection and Analysis

The process of collecting and analyzing information about the part of the organization that is to be supported by the database system, and using this information to identify the requirements for the new system.

This stage involves the collection and analysis of information about the part of the enterprise to be served by the database. There are many techniques for gather-ing this information, called **fact-finding techniques**. Information is gathered for each major user view (that is, job role or enterprise application area), including:

- a description of the data used or generated;
- the details of how data is to be used or generated;
- any additional requirements for the new database system.

This information is then analyzed to identify the requirements (or features) to be included in the new database system. These requirements are described in documents collectively referred to as **requirements specifications** for the new database system.

Requirements collection and analysis is a preliminary stage to database design. The amount of data gathered depends on the nature of the problem and the policies of the enterprise. Too much study too soon leads to *paralysis by analysis.* Too little thought can result in an unnecessary waste of both time and money due to working on the wrong solution to the wrong problem.

The information collected at this stage may be poorly structured and include some informal requests, which must be converted into a more structured statement of requirements. This is achieved using **requirements specification techniques**, which include, for example, Structured Analysis and Design (SAD) techniques, Data Flow Diagrams (DFD), and Hierarchical Input Process Output (HIPO) charts supported by documentation. As you will see shortly, Computer-Aided Software Engineering (CASE) tools may provide automated assistance to ensure that the requirements are complete and consistent

There are three main approaches to managing the requirements of a database sys-tem with
multiple user views:

- the **centralized** approach;
- the **view integration** approach;
- a combination of both approaches.

Centralized Approach

Requirements for each user view are merged into a single set of requirements for the new database system. A data model representing all user views is created during the database design stage.

The centralized (or one-shot) approach involves collating the requirements for different user views into a single list of requirements. The collection of user views is given a name that provides some indication of the application area covered by all the merged user views. In the database design stage (see Section 10.6), a global data model is created, which represents all user views. The global data model is composed of diagrams and documentation that formally describe the data requirements of the users. A diagram representing the management of user views 1 to 3 using the centralized approach is shown in Figure 10.3.

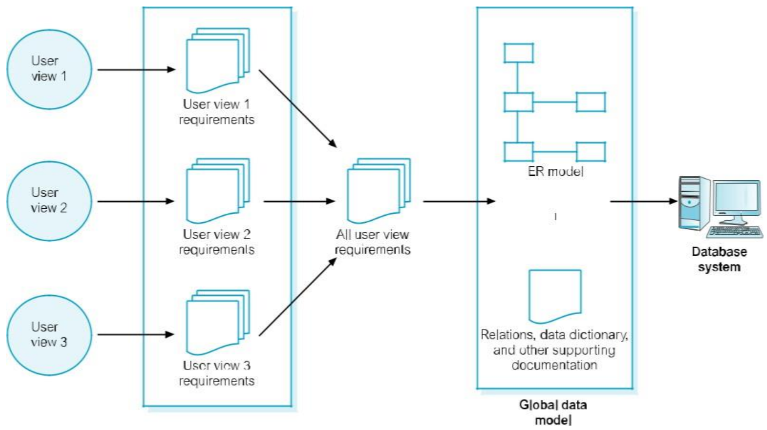

Figure 10.3 The centralized approach to managing multiple user views 1 to 3.

View Integration Approach

Requirements for each user view remain as separate lists. Data models representing each user view are created and then merged later during the database design stage.

The view integration approach involves leaving the requirements for each user view as separate lists of requirements. In the database design stage (see Section 10.6), we first create a data model for each user view. A data model that represents a single user view (or a subset of all user views) is called a **local data model**.

Each model is composed of diagrams and documentation that formally describes the requirements of one or more—but not all—user views of the database.

The local data models are then merged at a later stage of database design to produce a

global data model, which represents *all* user requirements for the database.

A diagram representing the management of user views 1 to 3 using the view integration approach is shown in Figure 10.4.

Generally, this approach is preferred when there are significant differences between user views and the database system is sufficiently complex to justify dividing the work into more manageable parts.

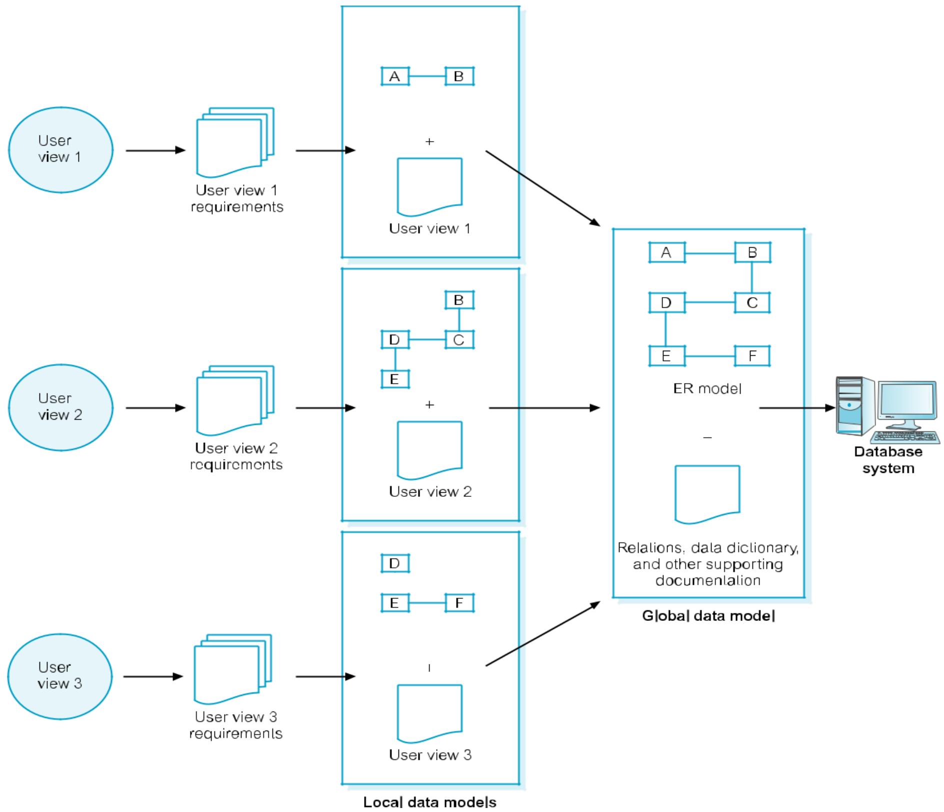

Figure 10.4 The view integration approach to managing multiple user views 1 to 3.

Database Design

The process of creating a design that will support the enterprise's mission statement and mission objectives for the required database system.
In this section we present an overview of the main approaches to database design. We also discuss the purpose and use of data modeling in database design. We then describe the three phases of database design: conceptual, logical, and physical design.

Approaches to Database Design

The two main approaches to the design of a database are referred to as "bottom-up" and "top-down." The **bottom-up** approach begins at the fundamental level of attributes (that is, properties of entities and relationships), which through analysis of the associations between attributes are grouped into relations that represent types of entities and relationships between entities. In Chapters 14 and 15 we dis-cuss the process of normalization, which represents a bottom-up approach to data-base design. Normalization involves the identification of the required attributes and their subsequent aggregation into normalized relations based on functional dependencies between the attributes.

The bottom-up approach is appropriate for the design of simple databases with a relatively small number of attributes. However, this approach becomes difficult when applied to the design of more complex databases with a larger number of attributes, where it is difficult to establish all the functional dependencies between the attributes. As the conceptual and logical data models for complex databases may contain hundreds to thousands of attributes, it is essential to establish an approach that will simplify the design process. Also, in the initial stages of establishing the data requirements for a complex database, it may be difficult to establish all the attributes to be included in the data models.

A more appropriate strategy for the design of complex databases is to use the **top-down** approach. This approach starts with the development of data models that contain a few high- level entities and relationships and then applies successive top-down refinements to identify lower-level entities, relationships, and the associated attributes. The top-down approach is illustrated using the concepts of the Entity-Relationship (ER) model, beginning with the identification of entities and relationships between the entities, which are of interest to the organization.

Data Modeling

The two main purposes of data modeling are to assist in the understanding of the meaning (semantics) of the data and to facilitate communication about the information requirements. Building a data model requires answering questions about entities, relationships, and attributes. In doing so, the designers discover the semantics of the enterprise's data, which exist whether or not they happen to be recorded in a formal data all enterprises. However, their meaning may remain poorly understood until they have been correctly documented. A data model makes it easier to understand the meaning of the data, and thus we model data to ensure that we understand:

- each user's perspective of the data;
- the nature of the data itself, independent of its physical representations;
- the use of data across user views.

Data models can be used to convey the designer's understanding of the information requirements of the enterprise.

Criteria for data models

An *optimal* data model should satisfy the criteria listed in Table 10.2 (Fleming and Von Halle, 1989). However, sometimes these criteria are incompatible with each other and trade-offs are sometimes necessary. For example, in attempting to achieve greater *expressibility* in a data model, we may lose *simplicity*.

Phases of Database Design

Database design is made up of three main phases: conceptual, logical, and physical design.

Conceptual database design

The process of constructing a model of the data used in an enterprise, independent of *all* physical considerations.

The first phase of database design is called **conceptual database design** and involves the creation of a conceptual data model of the part of the enterprise that we are interested in modeling. The data model is built using the information documented in the users' requirements specification.

Table 10.2 The criteria to produce an optimal data model.

Structural validity	Consistency with the way the enterprise defines and organizes information.
Simplicity	Ease of understanding by IS professionals and nontechnical users.
Expressibility	Ability to distinguish between different data, relationships between data, and constraints.
Nonredundancy	Exclusion of extraneous information; in particular, the representation of any one piece of information exactly once.
Shareability	Not specific to any particular application or technology and thereby usable by many.
Extensibility	Ability to evolve to support new requirements with minimal effect on existing users.
Integrity	Consistency with the way the enterprise uses and manages information.
Diagrammatic representation	Ability to represent a model using an easily understood diagrammatic notation.

Logical database design

The process of constructing a model of the data used in an enterprise based on a specific data model, but independent of a particular DBMS and other physical considerations.
The second phase of database design is called **logical database design**, which results in the creation of a logical data model of the part of the enterprise that we interested in modeling. The conceptual data model created in the previous phase is refined and mapped onto a logical data model. The logical data model is based on the target data model for the database (for example, the relational data model).

Whereas a conceptual data model is independent of all physical considerations, a logical model is derived knowing the underlying data model of the target DBMS. In other words, we know that the DBMS is, for example, relational, network, hierarchical, or object-oriented. However, we ignore any other aspects of the chosen DBMS and, in particular, any physical details, such as storage structures or indexes.

Throughout the process of developing a logical data model, the model is tested and validated against the users' requirements. The technique of **normalization** is used to test the correctness of a logical data model.

Physical database design

The process of producing a description of the implementation of the database on secondary storage; it describes the base relations, file organizations, and indexes used to achieve efficient access to the data, and any associated integrity constraints and security measures.
Physical database design is the third and final phase of the database design process, during which the designer decides how the database is to be implemented. The previous phase of database design involved the development of a logical structure for the database, which describes relations and enterprise constraints. Although this structure is DBMS-independent, it is developed in accordance with a particular data model, such as the relational, network, or hierarchic. However, in developing the physical database design, we must first identify the target DBMS. Therefore, physical design is tailored to a specific DBMS system. There is feedback between physical and logical design, because decisions are taken during physical design for improving performance that may affect the structure of the logical data model.

In general, the main aim of physical database design is to describe how we intend to physically implement the logical database design. For the relational model, this involves:

- Creating a set of relational tables and the constraints on these tables from the information presented in the logical data model;
- Identifying the specific storage structures and access methods for the data to achieve an optimum performance for the database system;
- Designing security protection for the system.

Ideally, conceptual and logical database design for larger systems should be separated from physical design for three main reasons:

- It deals with a different subject matter—the *what*, not the *how;*
- It is performed at a different time—the *what* must be understood before the *how* can be determined;
- It requires different skills, which are often found in different people.

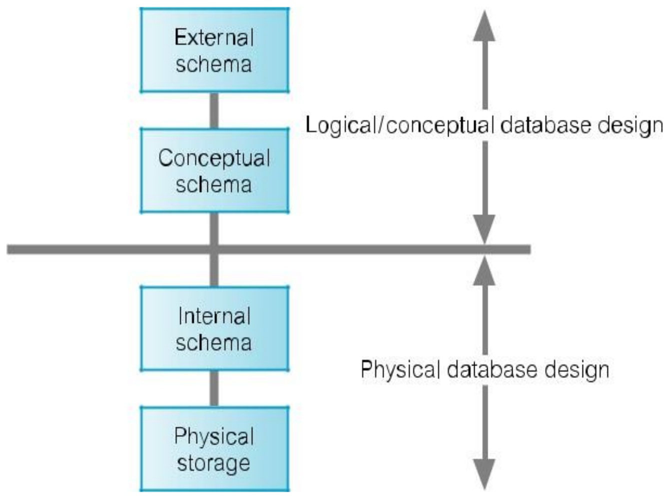

Fig: Data modeling and the ANSI-SPARC architecture.