A typical bottom-up database development project usually focuses on the creation of one database. Some database projects concentrate only on defining, designing, and implementing a database as a foundation for subsequent information systems development. In most cases, however, a database and the associated information processing functions are developed together as part of a comprehensive information systems development project.

Systems Development Life Cycle

Systems development life cycle (SDLC)

The traditional methodology used to develop, maintain, and replace information systems.

As you may know from other information systems courses you've taken, a traditional process for conducting an information systems development project is called the **systems development life cycle (SDLC)**. The SDLC is a complete set of steps that a team of information systems professionals, including database designers and programmers, follow in an organization to specify, develop, maintain, and replace information systems. Textbooks and organizations use many variations on the life cycle and may identify anywhere from 3 to 20 different phases.

The various steps in the SDLC and their associated purpose are depicted in Figure 1-7 (Hoffer et al., 2014). The process appears to be circular and is intended to convey the iterative nature of systems development projects. The steps may overlap in time, they may be conducted in parallel, and it is possible to backtrack to previous steps when prior decisions need to be reconsidered. Some believe that the most common path through the development process is to cycle through the steps depicted in Figure 1-7, but at more detailed levels on each pass, as the requirements of the system become more concrete.

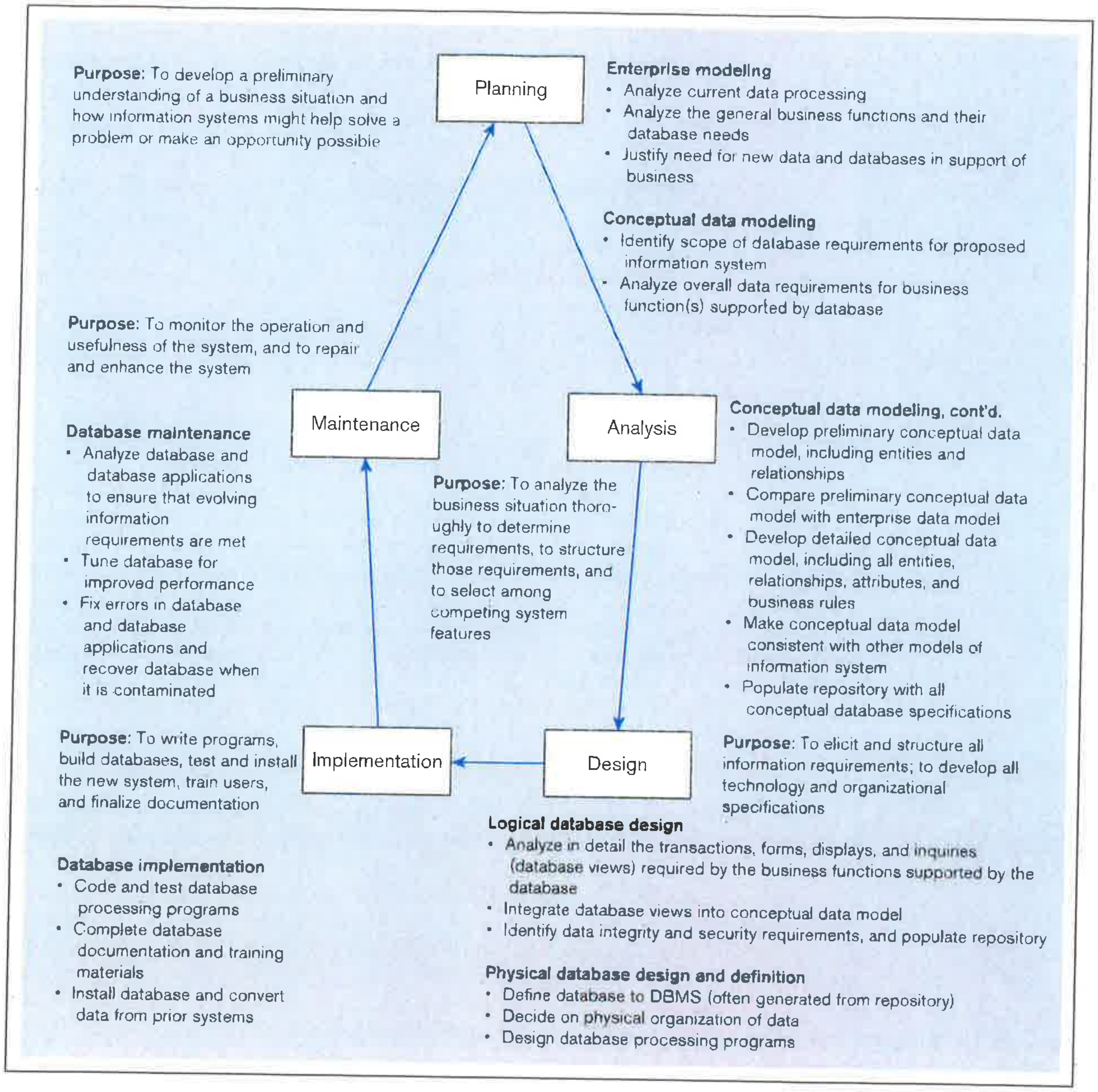
Figure 1-7 also provides an outline of the database development activities typically included in each phase of the SDLC. Note that there is not always a one-to-one correspondence between SDLC phases and database development steps. For example, conceptual data modeling occurs in both the Planning and the Analysis phases. We will briefly illustrate each of these database development steps for Pine Valley Furniture Company later in this chapter.

PLANNING—ENTERPRISE MODELING The database development process begins with a review of the enterprise modeling components that were developed during the information systems planning process. During this step, analysts review current databases and information systems; analyze the nature of the business area that is the subject of the development project; and describe, in general terms, the data needed for each information system under consideration for development. They determine what data are already available in existing databases and what new data will need to be added to support the proposed new project. Only selected projects move into the next phase based on the projected value of each project to the organization.

PLANNING—CONCEPTUAL DATA MODELING For an information systems project that is initiated, the overall data requirements of the proposed information system must be analyzed. This is done in two stages. First, during the Planning phase, the analyst develops a diagram similar to Figure 1-3a, as well as other documentation, to outline the scope of data involved in this particular development project without consideration of what databases already exist. Only high-level categories of data (entities) and major relationships are included at this point. This step in the SDLC is critical for improving the chances of a successful development process. The better the definition of the specific needs of the organization, the closer the conceptual model should come to meeting the needs of the organization, and the less recycling back through the SDLC should be needed.

ANALYSIS—CONCEPTUAL DATA MODELING During the Analysis phase of the SDLC, the analyst produces a detailed data model that identifies all the organizational data that must be managed for this information system. Every data attribute is defined, all categories of data are listed, every business relationship between data entities is represented, and every rule that dictates the integrity of the data is specified. It is also during the Analysis phase that the conceptual data model is checked for consistency

FIGURE 1-7 **Database development activities during the systems development life cycle (SDLC)**


Purpose: To develop a preliminary understanding of a business situation and how information systems might help solve a problem or make an opportunity possible
Planning
Enterprise modeling
• Analyze current data processing
• Analyze the general business functions and their database needs
• Justify need for new data and databases in support of business
Conceptual data modeling
• Identify scope of database requirements for proposed information system
• Analyze overall data requirements for business function(s) supported by database
Purpose: To monitor the operation and usefulness of the system, and to repair and enhance the system
Maintenance
Analysis
Conceptual data modeling, cont'd.
• Develop preliminary conceptual data model, including entities and relationships
• Compare preliminary conceptual data model with enterprise data model
• Develop detailed conceptual data model, including all entities, relationships, attributes, and business rules
• Make conceptual data model consistent with other models of information system
• Populate repository with all conceptual database specifications
Database maintenance
• Analyze database and database applications to ensure that evolving information requirements are met
• Tune database for improved performance
• Fix errors in database and database applications and recover database when it is contaminated
Purpose: To analyze the business situation thoroughly to determine requirements, to structure those requirements, and to select among competing system features
Purpose: To write programs, build databases, test and install the new system, train users, and finalize documentation
Implementation
Design
Purpose: To elicit and structure all information requirements; to develop all technology and organizational specifications
Logical database design
• Analyze in detail the transactions, forms, displays, and inquiries (database views) required by the business functions supported by the database
• Integrate database views into conceptual data model
• Identify data integrity and security requirements, and populate repository
Database implementation
• Code and test database processing programs
• Complete database documentation and training materials
• Install database and convert data from prior systems
Physical database design and definition
• Define database to DBMS (often generated from repository)
• Decide on physical organization of data
• Design database processing programs

with other types of models developed to explain other dimensions of the target information system, such as processing steps, rules for handling data, and the timing of events. However, even this detailed conceptual data model is preliminary, because subsequent SDLC activities may find missing elements or errors when designing specific transactions, reports, displays, and inquiries. With experience, the database developer gains mental models of common business functions, such as sales or financial record keeping, but must always remain alert for the exceptions to common practices followed by an organization. The output of the conceptual modeling phase is a **conceptual schema**.

Conceptual schema

A detailed, technology-independent specification of the overall structure of organizational data.

DESIGN—LOGICAL DATABASE DESIGN Logical database design approaches database development from two perspectives. First, the conceptual schema must be transformed into a logical schema, which describes the data in terms of the data management technology that will be used to implement the database. For example, if relational technology will be used, the conceptual data model is transformed and represented using

elements of the relational model, which include tables, columns, rows, primary keys, foreign keys, and constraints. (You will learn how to conduct this important process in Chapter 4.) This representation is referred to as the **logical schema**.

Logical schema
The representation of a database for a particular data management technology.

Then, as each application in the information system is designed, including the program's input and output formats, the analyst performs a detailed review of the transactions, reports, displays, and inquiries supported by the database. During this so-called bottom-up analysis, the analyst verifies exactly what data are to be maintained in the database and the nature of those data as needed for each transaction, report, and so forth. It may be necessary to refine the conceptual data model as each report, business transaction, and other user view is analyzed. In this case, one must combine, or integrate, the original conceptual data model along with these individual user views into a comprehensive design during logical database design. It is also possible that additional information processing requirements will be identified during logical information systems design, in which case these new requirements must be integrated into the previously identified logical database design.

The final step in logical database design is to transform the combined and reconciled data specifications into basic, or atomic, elements following well-established rules for well-structured data specifications. For most databases today, these rules come from relational database theory and a process called *normalization,* which we will describe in detail in Chapter 4. The result is a complete picture of the database without any reference to a particular database management system for managing these data. With a final logical database design in place, the analyst begins to specify the logic of the particular computer programs and queries needed to maintain and report the database contents.

Physical schema
Specifications for how data from a logical schema are stored in a computer's secondary memory by a database management system.

DESIGN—PHYSICAL DATABASE DESIGN AND DEFINITION A **physical schema** is a set of specifications that describe how data from a logical schema are stored in a computer's secondary memory by a specific database management system. There is one physical schema for each logical schema. Physical database design requires knowledge of the specific DBMS that will be used to implement the database. In physical database design and definition, an analyst decides on the organization of physical records, the choice of file organizations, the use of indexes, and so on. To do this, a database designer needs to outline the programs to process transactions and to generate anticipated management information and decision-support reports. The goal is to design a database that will efficiently and securely handle all data processing against it. Thus, physical database design is done in close coordination with the design of all other aspects of the physical information system: programs, computer hardware, operating systems, and data communications networks.

IMPLEMENTATION—DATABASE IMPLEMENTATION In database implementation, a designer writes, tests, and installs the programs/scripts that access, create, or modify the database. The designer might do this using standard programming languages (e.g., Java, C#, or Visual Basic.NET) or in special database processing languages (e.g., SQL) or use special-purpose nonprocedural languages to produce stylized reports and displays, possibly including graphs. Also, during implementation, the designer will finalize all database documentation, train users, and put procedures into place for the ongoing support of the information system (and database) users. The last step is to load data from existing information sources (files and databases from legacy applications plus new data now needed). Loading is often done by first unloading data from existing files and databases into a neutral format (such as binary or text files) and then loading these data into the new database. Finally, the database and its associated applications are put into production for data maintenance and retrieval by the actual users. During production, the database should be periodically backed up and recovered in case of contamination or destruction.

MAINTENANCE—DATABASE MAINTENANCE The database evolves during database maintenance. In this step, the designer adds, deletes, or changes characteristics of the structure of a database in order to meet changing business conditions, to correct errors

in database design, or to improve the processing speed of database applications. The designer might also need to rebuild a database if it becomes contaminated or destroyed due to a program or computer system malfunction. This is typically the longest step of database development, because it lasts throughout the life of the database and its associated applications. Each time the database evolves, view it as an abbreviated database development process in which conceptual data modeling, logical and physical database design, and database implementation occur to deal with proposed changes.

Alternative Information Systems (IS) Development Approaches

The systems development life cycle or slight variations on it are often used to guide the development of information systems and databases. The SDLC is a methodical, highly structured approach, which includes many checks and balances to ensure that each step produces accurate results and the new or replacement information system is consistent with existing systems with which it must communicate or for which there needs to be consistent data definitions. Whew! That's a lot of work! Consequently, the SDLC is often criticized for the length of time needed until a working system is produced, which occurs only at the end of the process. Instead, organizations increasingly use rapid application development (RAD) methods, which follow an iterative process of rapidly repeating analysis, design, and implementation steps until they converge on the system the user wants. These RAD methods work best when most of the necessary database structures already exist, and hence for systems that primarily retrieve data, rather than for those that populate and revise databases.

One of the most popular RAD methods is **prototyping**, which is an iterative process of systems development in which requirements are converted to a working system that is continually revised through close work between analysts and users. Figure 1-8 shows the prototyping process. This figure includes annotations to indicate roughly which database development activities occur in each prototyping phase. Typically, you make only a very cursory attempt at conceptual data modeling when the information system problem is identified. During the development of the initial prototype, you simultaneously design the displays and reports the user wants while

Prototyping
An iterative process of systems development in which requirements are converted to a working system that is continually revised through close work between analysts and users.

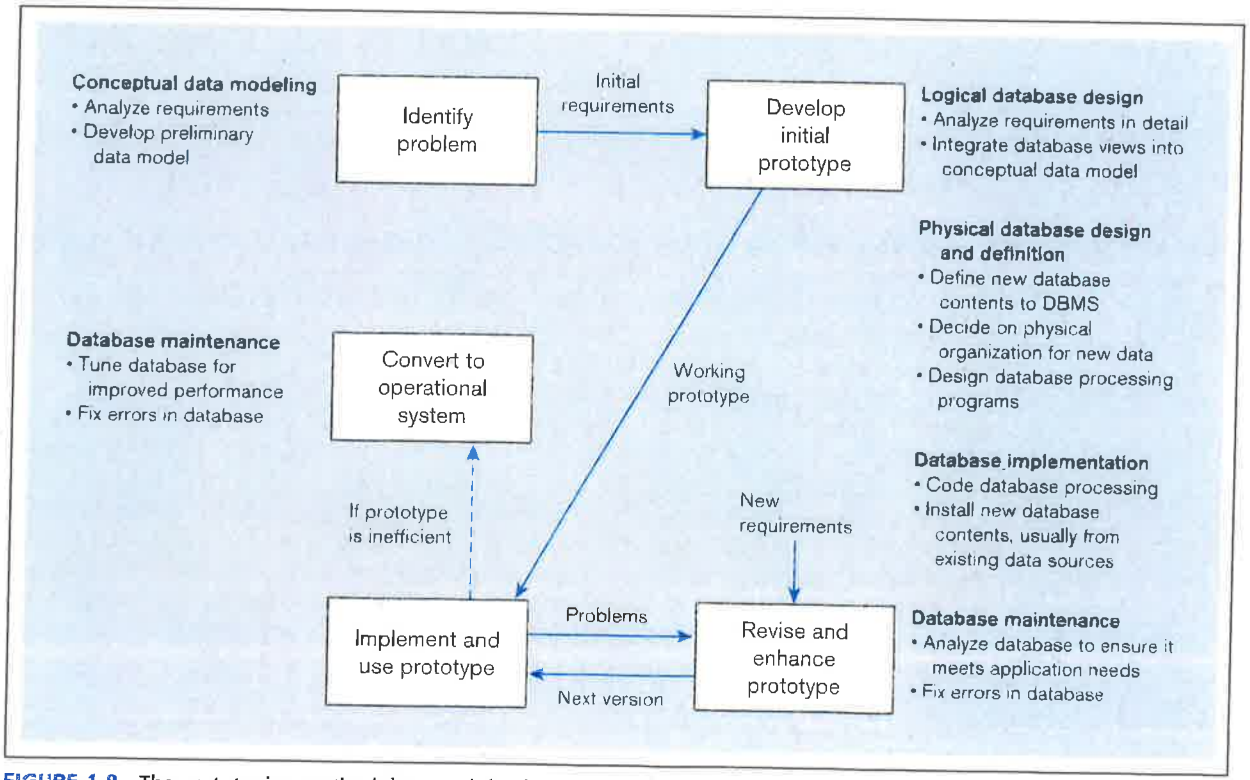

Conceptual data modeling
• Analyze requirements
• Develop preliminary data model
Identify problem
Initial requirements
Develop initial prototype
Logical database design
• Analyze requirements in detail
• Integrate database views into conceptual data model
Physical database design and definition
• Define new database contents to DBMS
• Decide on physical organization for new data
• Design database processing programs
Database maintenance
• Tune database for improved performance
• Fix errors in database
Convert to operational system
Working prototype
Database implementation
• Code database processing
• Install new database contents, usually from existing data sources
If prototype is inefficient
New requirements
Problems
Implement and use prototype
Revise and enhance prototype
Next version
Database maintenance
• Analyze database to ensure it meets application needs
• Fix errors in database

FIGURE 1-8 The prototyping methodology and database development process

understanding any new database requirements and defining a database to be used by the prototype. This is typically a new database, which is a copy of portions of existing databases, possibly with new content. If new content is required, it will usually come from external data sources, such as market research data, general economic indicators, or industry standards.

Database implementation and maintenance activities are repeated as new versions of the prototype are produced. Often security and integrity controls are minimal because the emphasis is on getting working prototype versions ready as quickly as possible. Also, documentation tends to be delayed until the end of the project, and user training occurs from hands-on use. Finally, after an accepted prototype is created, the developer and the user decide whether the final prototype, and its database, can be put into production as is. If the system, including the database, is too inefficient, the system and database might need to be reprogrammed and reorganized to meet performance expectations. Inefficiencies, however, have to be weighed against violating the core principles behind sound database design.

With the increasing popularity of visual programming tools (such as Visual Basic, Java, or C#) that make it easy to modify the interface between user and system, prototyping is becoming the systems development methodology of choice to develop new applications internally. With prototyping, it is relatively easy to change the content and layout of user reports and displays.

The benefits from iterative approaches to systems development demonstrated by RAD and prototyping approaches have resulted in further efforts to create ever more responsive development approaches. In February 2001, a group of 17 individuals interested in supporting these approaches created "The Manifesto for Agile Software Development." For them, **agile software development** practices include valuing **(www.agilemanifesto.org)**:

Agile software development

An approach to database and software development that emphasizes "**individuals and interactions** over processes and tools, **working software** over comprehensive documentation, **customer collaboration** over contract negotiation, and **response to change** over following a plan."

Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation, and

Responding to change over following a plan

Emphasis on the importance of people, both software developers and customers, is evident in their phrasing. This is in response to the turbulent environment within which software development occurs, as compared to the more staid environment of most engineering development projects from which the earlier software development methodologies came. The importance of the practices established in the SDLC continues to be recognized and accepted by software developers including the creators of The Manifesto for Agile Software Development. However, it is impractical to allow these practices to stifle quick reactions to changes in the environment that change project requirements.

The use of agile or adaptive processes should be considered when a project involves unpredictable and/or changing requirements, responsible and collaborative developers, and involved customers who understand and can contribute to the process (Fowler, 2005). If you are interested in learning more about agile software development, investigate agile methodologies such as eXtreme Programming, Scrum, the DSDM Consortium, and feature-driven development.

Three-Schema Architecture for Database Development

The explanation earlier in this chapter of the database development process referred to several different, but related, models of databases developed on a systems development project. These data models and the primary phase of the SDLC in which they are developed are summarized here:

- Enterprise data model (during the Information Systems Planning phase)
- External schema or user view (during the Analysis and Logical Design phases)
- Conceptual schema (during the Analysis phase)
- Logical schema (during the Logical Design phase)
- Physical schema (during the Physical Design phase)

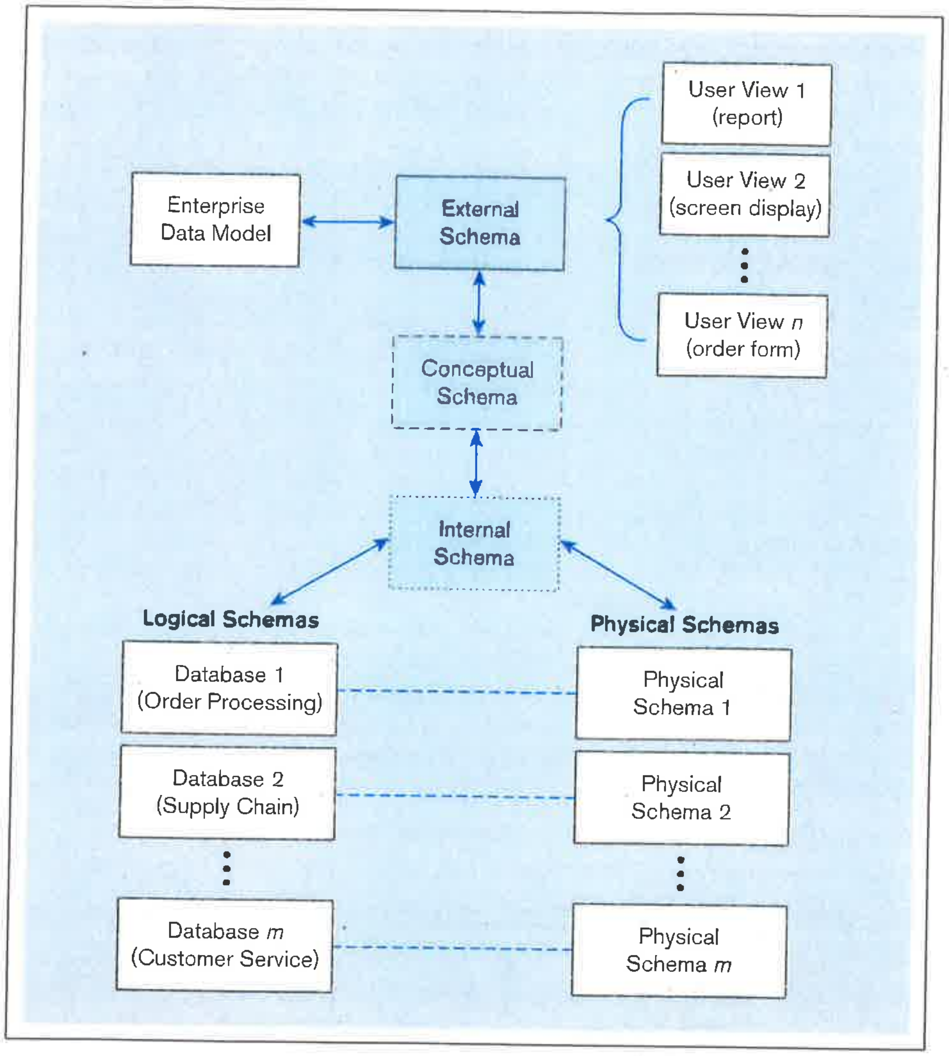


FIGURE 1-9 Three-schema architecture

In 1978, an industry committee commonly known as ANSI/SPARC published an important document that described three-schema architecture—external, conceptual, and internal schemas—for describing the structure of data. Figure 1-9 shows the relationship between the various schemas developed during the SDLC and the ANSI three-schema architecture. It is important to keep in mind that all these schemas are just different ways of visualizing the structure of the same database by different stakeholders.

The three schemas as defined by ANSI (depicted down the center of Figure 1-9) are as follows:

1. ***External schema*** This is the view (or views) of managers and other employees who are the database users. As shown in Figure 1-9, the external schema can be represented as a combination of the enterprise data model (a top-down view) and a collection of detailed (or bottom-up) user views.
2. ***Conceptual schema*** This schema combines the different external views into a single, coherent, and comprehensive definition of the enterprise's data. The conceptual schema represents the view of the data architect or data administrator.
3. ***Internal schema*** As shown in Figure 1-9, an internal schema today really consists of two separate schemas: a logical schema and a physical schema. The logical schema is the representation of data for a type of data management technology (e.g., relational). The physical schema describes how data are to be represented and stored in secondary storage using a particular DBMS (e.g., Oracle).