Chapter 9

Database Design

Discussion Focus

What is the relationship between a database and an information system, and how does this relationship have a bearing on database design?

An information system performs three sets of services:

- It provides for data collection, storage, and retrieval.
- It facilitates the transformation of data into information.
- It provides the tools and conditions to manage both data and information.

Basically, a database is a fact (data) repository that serves an information system. If the database is designed poorly, one can hardly expect that the data/information transformation will be successful, nor is it reasonable to expect efficient and capable management of data and information.

The transformation of data into information is accomplished through application programs. It is impossible to produce good information from poor data; and, no matter how sophisticated the application programs are, *it is impossible to use good application programs to overcome the effects of bad database design.* In short: **Good database design is the foundation of a successful information system**.

Database design must yield a database that:

- Does not fall prey to uncontrolled data duplication, thus preventing data anomalies and the attendant lack of data integrity.
- Is efficient in its provision of data access.
- Serves the needs of the information system.

The last point deserves emphasis: even the best-designed database lacks value if it fails to meet information system objectives. In short, good database designers must pay close attention to the information system requirements.

Systems design and database design are usually tightly intertwined and are often performed in parallel. Therefore, database and systems designers must cooperate and coordinate to yield the best possible information system.

What is the relationship between the SDLC and the DBLC?

The SDLC traces the history (life cycle) of an information system. The DBLC traces the history (life cycle) of a database system. Since we know that the database serves the information system, it is not surprising that the two life cycles conform to the same basic phases.

Suggestion: Use Figure 9.8 as the basis for a discussion of the parallel activities.

What basic database design strategies exist, and how are such strategies executed?

Suggestion: Use Figure 9.14 as the basis for this discussion.

There are two basic approaches to database design: top-down and bottom-up.

Top-down design begins by identifying the different entity types and the definition of each entity's attributes. In other words, top-down design:

- starts by defining the required data sets and then
- defines the data elements for each of those data sets.

Bottom-up design:

- first defines the required attributes and then
- groups the attributes to form entities.

Although the two methodologies tend to be complementary, database designers who deal with small databases with relatively few entities, attributes, and transactions tend to emphasize the bottom-up approach. Database designers who deal with large, complex databases usually find that a primarily top-down design approach is more appropriate.

In spite of the frequent arguments concerning the best design approach, perhaps the top-down *vs*. bottom-up distinction is quite artificial. The text's note is worth repeating:

NOTE

Even if a *generally* top-down approach is selected, the normalization process that *revises* existing table structures is (inevitably) a bottom-up technique. E-R models constitute a top-down process even if the selection of attributes and entities may be described as bottom-up. Since both the E-R model and normalization techniques form the basis for most designs, the top-down *vs*. bottom-up debate may be based on a distinction without a difference.

Answers to Review Questions

1. **What is an information system? What is its purpose?**

   An information system is a system that
   - provides the conditions for data collection, storage, and retrieval
   - facilitates the transformation of data into information
   - provides management of both data and information.

   An information system is composed of hardware, software (DBMS and applications), the database(s), procedures, and people.

   Good decisions are generally based on good information. Ultimately, the purpose of an information system is to facilitate good decision making by making relevant and timely information available to the decision makers.

2. **How do systems analysis and systems development fit into a discussion about information systems?**

   Both systems analysis and systems development constitute part of the Systems Development Life Cycle, or SDLC. Systems analysis, phase II of the SDLC, establishes the need for and the extent of an information system by
   - Establishing end-user requirements.
   - Evaluating the existing system.
   - Developing a logical systems design.

   Systems development, based on the detailed systems design found in phase III of the SDLC, yields the information system. The detailed system specifications are established during the systems design phase, in which the designer completes the design of all required system processes.

3. **What does the acronym SDLC mean, and what does an SDLC portray?**

   SDLC is the acronym that is used to label the System Development Life Cycle. The SDLC traces the history of a information system from its inception to its obsolescence. The SDLC is composed of six phases: planning, analysis, detailed system, design, implementation and maintenance.

4. **What does the acronym DBLC mean, and what does a DBLC portray?**

   DBLC is the acronym that is used to label the Database Life Cycle. The DBLC traces the history of a database system from its inception to its obsolescence. Since the database constitutes the core of an information system, the DBLC is concurrent to the SDLC. The DBLC is composed of six phases: initial study, design, implementation and loading, testing and evaluation, operation, and maintenance and evolution.

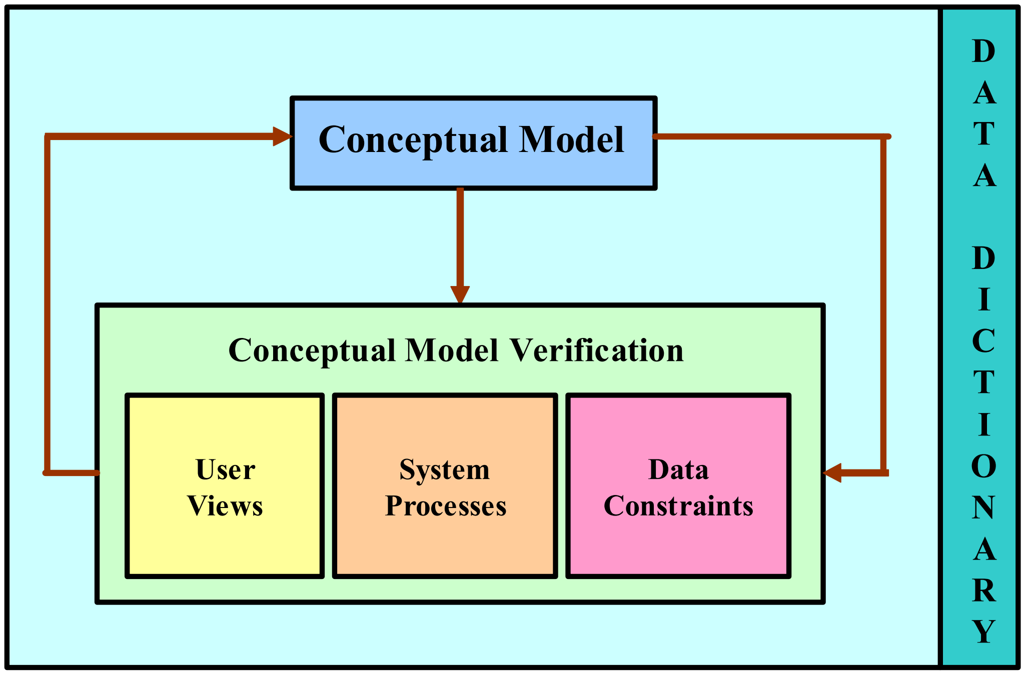
5. Discuss the distinction between centralized and decentralized conceptual database design.

Centralized and decentralized design constitute variations on the bottom-up and top-down approaches we discussed in the third question presented in the discussion focus. Basically, the centralized approach is best suited to relatively small and simple databases that lend themselves well to a bird's-eye view of the entire database. Such databases may be designed by a single person or by a small and informally constituted design team. The company operations and the scope of its problems are sufficiently limited to enable the designer(s) to perform all of the necessary database design tasks:

1. Define the problem(s).
2. Create the conceptual design.
3. Verify the conceptual design with all user views.
4. Define all system processes and data constraints.
5. Assure that the database design will comply with all achievable end user requirements.

The centralized design procedure thus yields the design summary shown in Figure Q9.5A.

Figure Q9.5A The Centralized Design Procedure



Note that the centralized design approach requires the completion and validation of a *single* conceptual design.

NOTE

Use the text's Figures 9.15 and 9.16 to contrast the two design approaches, then use Figure 9.6 to show the procedure flows; demonstrate that such procedure flows are independent of the degree of centralization.

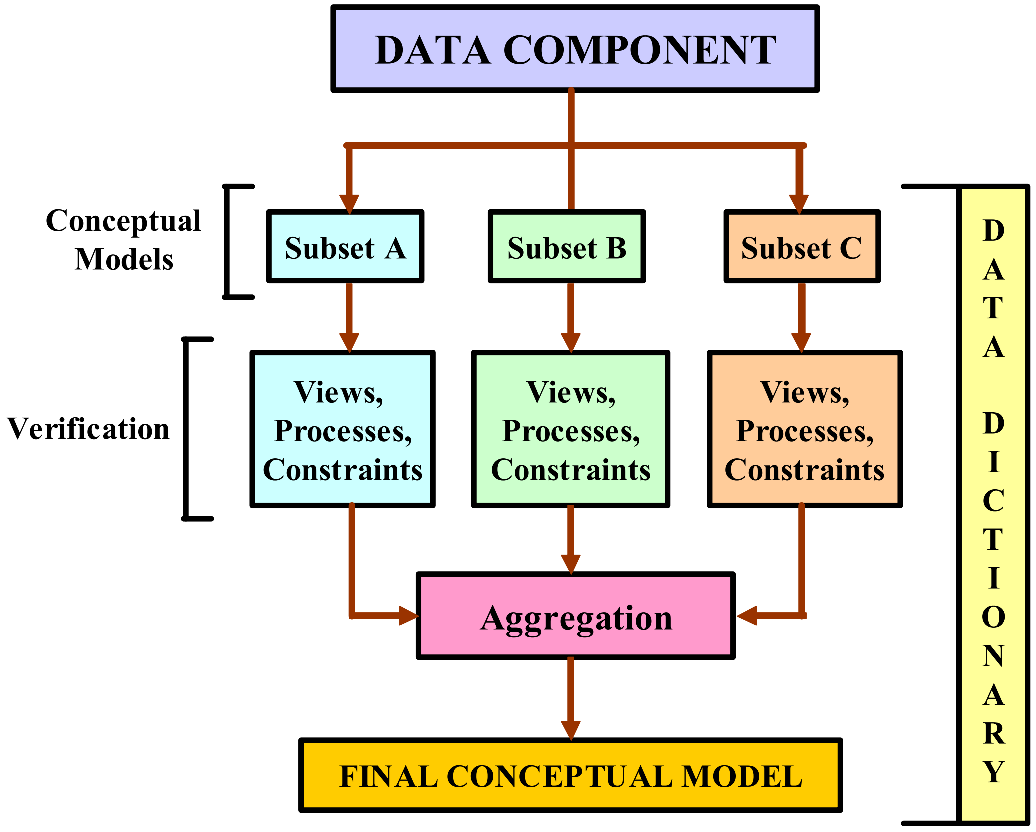
In contrast, when company operations are spread across multiple operational sites or when the database has multiple entities that are subject to complex relations, the best approach is often based on the decentralized design.

Typically, a decentralized design requires that the design task be divided into multiple modules, each one of which is assigned to a design team. The design team activities are coordinated by the lead designer, who must aggregate the design teams' efforts.

Since each team focuses on modeling a subset of the system, the definition of boundaries and the interrelation between data subsets must be very precise. Each team creates a conceptual data model corresponding to the subset being modeled. Each conceptual model is then verified individually against the user views, processes, and constraints for each of the modules. After the verification process has been completed, all modules are integrated in one conceptual model.

Since the data dictionary describes the characteristics of all the objects within the conceptual data model, it plays a vital role in the integration process. Naturally, after the subsets have been aggregated into a larger conceptual model, the lead designer must verify that the *combined* conceptual model is still able to support all the required transactions. Thus the decentralized design activities may be summarized as shown in Figure Q8.6B.

Figure Q9.6B The Decentralized Design Procedure



Keep in mind that the aggregation process requires the lead designer to assemble a single model in which various aggregation problems must be addressed:

- **synonyms and homonyms**. Different departments may know the *same object by different names* (**synonyms**), or they may use the *same name to address different objects* (**homonyms**.) The object may be an entity, an attribute, or a relationship.
- **entity and entity subclasses**. An entity subset may be viewed as a separate entity by one or more departments. The designer must integrate such subclasses into a higher-level entity.
- **Conflicting object definitions**. Attributes may be recorded as different types (character, numeric), or different domains may be defined for the same attribute. Constraint definitions, too, may vary. The designer must remove such conflicts from the model.

6. What is the minimal data rule in conceptual design? Why is it important?

The minimal data rule specifies that all the data defined in the data model are actually required to fit present and expected future data requirements. This rule may be phrased as ***All that is needed is there, and all that is there is needed***.

7. Discuss the distinction between top-down and bottom-up approaches to database design.

There are two basic approaches to database design: top-down and bottom-up.

Top-down design begins by identifying the different entity types and the definition of each entity's attributes. In other words, top-down design:

- starts by defining the required data sets and then
- defines the data elements for each of those data sets.

Bottom-up design:

- first defines the required attributes and then
- groups the attributes to form entities.

Although the two methodologies tend to be complementary, database designers who deal with small databases with relatively few entities, attributes, and transactions tend to emphasize the bottom-up approach. Database designers who deal with large, complex databases usually find that a primarily top-down design approach is more appropriate.

8. What are business rules? Why are they important to a database designer?

Business rules are narrative descriptions of the business policies, procedures, or principles that are derived from a detailed description of operations. Business rules are particularly valuable to database designers, because they help define:

- Entities
- Attributes
- Relationships (1:1, 1:M, M:N, expressed through connectivities and cardinalities)
- Constraints

To develop an accurate data model, the database designer must have a thorough and complete understanding of the organization's data requirements. The business rules are very important to the designer because they enable the designer to fully understand how the business works and what role is played by data within company operations.

NOTE

Do keep in mind that an ERD cannot always include all the applicable business rules. For example, although constraints are often crucial, it is often not possible to model them. For instance, there is no way to model a constraint such as "no pilot may be assigned to flight duties more than ten hours during any 24-hour period."

It is also worth emphasizing that the description of (company) operations must be done in almost excruciating detail and it must be verified and re-verified. An inaccurate description of operations yields inaccurate business

rules that lead to database designs that are destined to fail.

9. **What is the data dictionary's function in database design?**

A good data dictionary provides a precise description of the characteristics of all the entities and attributes found within the database. The data dictionary thus makes it easier to check for the existence of synonyms and homonyms, to check whether all attributes exist to support required reports, to verify appropriate relationship representations, and so on. The data dictionary's contents are both developed and used during the six DBLC phases:

DATABASE INITIAL STUDY
The basic data dictionary components are developed as the entities and attributes are defined during this phase.

DATABASE DESIGN
The data dictionary contents are used to verify the database design components: entities, attributes, and their relationships. The designer also uses the data dictionary to check the database design for homonyms and synonyms and verifies that the entities and attributes will support all required query and report requirements.

IMPLEMENTATION AND LOADING
The DBMS's data dictionary helps to resolve any remaining attribute definition inconsistencies.

TESTING AND EVALUATION
If problems develop during this phase, the data dictionary contents may be used to help restructure the basic design components to make sure that they support all required operations.

OPERATION
If the database design still yields (the almost inevitable) operational glitches, the data dictionary may be used as a quality control device to ensure that operational modifications to the database do not conflict with existing components.

MAINTENANCE AND EVOLUTION
As users face inevitable changes in information needs, the database may be modified to support those needs. Perhaps entities, attributes, and relationships must be added, or relationships must be changed. If new database components are fit into the design, their introduction may produce conflict with existing components. The data dictionary turns out to be a very useful tool to check whether a suggested change invites conflicts within the database design and, if so, how such conflicts may be resolved.

10. **What steps are required in the development of an ER diagram? (*Hint:* See Table 9.3.)**

Table 9.3 is reproduced for your convenience.

TABLE 9.3 Developing the Conceptual Model, Using ER Diagrams

STEP	ACTIVITY
1	Identify, analyze, and refine the business rules.
2	Identify the main entities, using the results of Step 1.
3	Define the relationships among the entities, using the results of Steps 1 and 2.
4	Define the attributes, primary keys, and foreign keys for each of the entities.
5	Normalize the entities. (Remember that entities are implemented as tables in an RDBMS.)
6	Complete the initial ER diagram.
7	Validate the ER model against the user's information and processing requirements.
8	Modify the ER diagram, using the results of Step 7.

Point out that some of the steps listed in Table 9.3 take place concurrently. And some, such as the normalization process, can generate a demand for additional entities and/or attributes, thereby causing the designer to revise the ER model. For example, while identifying two main entities, the designer might also identify the composite bridge entity that represents the many-to-many relationship between those two main entities.

11. List and briefly explain the activities involved in the verification of an ER model.

Section 9-4c, "Data Model Verification," includes a discussion on verification. In addition, Appendix C, "The University Lab: Conceptual Design Verification, Logical Design, and Implementation," covers the verification process in detail. The verification process is detailed in the text's Table 9.5, reproduced here for your convenience.

TABLE 9.5 The ER Model Verification Process

STEP	ACTIVITY
1	Identify the ER model's central entity.
2	Identify each module and its components.
3	Identify each module's transaction requirements: Internal: Updates/Inserts/Deletes/Queries/Reports External: Module interfaces
4	Verify all processes against the ER model.
5	Make all necessary changes suggested in Step 4.
6	Repeat Steps 2−5 for all modules.

Keep in mind that the verification process requires the continuous verification of business transactions as well as system and user requirements. The verification sequence must be repeated for each of the system's modules.

12. What factors are important in a DBMS software selection?

The selection of DBMS software is critical to the information system's smooth operation. Consequently, the advantages and disadvantages of the proposed DBMS software should be carefully studied. To avoid false expectations, the end user must be made aware of the limitations of both the DBMS and the database.

Although the factors affecting the purchasing decision vary from company to company, some of the most common are:

- *Cost*. Purchase, maintenance, operational, license, installation, training, and conversion costs.
- *DBMS features and tools*. Some database software includes a variety of tools that facilitate the application development task. For example, the availability of query by example (QBE), screen painters, report generators, application generators, data dictionaries, and so on, helps to create a more pleasant work environment for both the end user and the application programmer. Database administrator facilities, query facilities, ease of use, performance, security, concurrency control, transaction processing, and third-party support also influence DBMS software selection.
- *Underlying model*. Hierarchical, network, relational, object/relational, or object.
- *Portability*. Across platforms, systems, and languages.
- *DBMS hardware requirements*. Processor(s), RAM, disk space, and so on.

13. List and briefly explain the four steps performed during the logical design stage.

1) Map conceptual model to logical model components.
   In this step, the conceptual model is converted into a set of table definitions including table names, column names, primary keys, and foreign keys to implement the entities and relationships specified in the conceptual design.
2) Validate the logical model using normalization.
   It is possible for normalization issues to be discovered during the process of mapping the conceptual model to logical model components. Therefore, it is appropriate at this stage to validate that all of the table definitions from the previous step conform to the appropriate normalization rules.
3) Validate logical model integrity constraints.
   This step involves the conversion of attribute domains and constraints into constraint definitions that can be implemented within the DBMS to enforce those domains. Also, entity and referential integrity constraints are validated. Views may be defined to enforce security constraints.
4) Validate the logical model against the user requirements.
   The final step of this stage is to ensure that all definitions created throughout the logical model are validated against the users' data, transaction, and security requirements. Every component (table, view, constraint, etc.) of the logical model must be associated with satisfying the user requirements, and every user requirement should be addressed by the model components.

14. List and briefly explain the three steps performed during the physical design stage.

1) Define data storage organization.
   Based on estimates of the data volume and growth, this step involves the determination of the physical location and physical organization for each table. Also, which columns will be indexed and the type of indexes to be used are determined. Finally, the type of implementation to be used for each view is decided.
2) Define integrity and security measures.
   This step involves creating users and security groups, and then assigning privileges and controls to those users and group.
3) Determine performance measurements.
   The actual performance of the physical database implementation must be measured and assessed for compliance with user performance requirements.

15. What three levels of backup may be used in database recovery management? Briefly describe what each of those three backup levels does.

A **full backup** of the database creates a backup copy of all database objects in their entirety.
A **differential backup** of the database creates a backup of only those database objects that have changed since the last full backup.
A **transaction log backup** does not create a backup of database objects, but makes a backup of the log of changes that have been applied to the database objects since the last backup.