WelTec
Te Whare Wānanga o te Awakairangi

西安科技大学高新学院
XI'AN KEDAGAOXIN UNIVERSITY

IT5507 Fundamentals of Data Science

Chapter 9
Database Design

CENGAGE

Learning Objectives

- After completing this chapter, you will be able to:
  - Describe the role of database design as the foundation of a successful information system
  - Describe the five phases in the Systems Development Life Cycle (SDLC)
  - Design databases using the six phases in the Database Life Cycle (DBLC) framework
  - Conduct evaluation and revision within the SDLC and DBLC frameworks
  - Distinguish between top-down and bottom-up approaches in database design
  - Distinguish between centralized and decentralized conceptual database design

The Information System (1 of 2)

- The database is part of a larger whole known as an information system (IS)
- **Information System (IS):** A network of components working together to collect, store, and retrieve data.

Example: A university's information system manages student records, course schedules, and faculty data.

- **People, Hardware, and Software:** Essential elements of an IS.

Example: In a retail business, people use point-of-sale terminals (hardware) running inventory management software to process sales transactions.

- **Database(s), Application Programs, and Procedures:** Key components of an IS.

Example: A hospital's IS includes a patient database, medical billing software (application program), and procedures for admitting patients.

- **Systems Analysis:** Evaluates the need and scope of an IS.

Example: Before implementing a new CRM system, a company conducts systems analysis to assess its customer relationship management requirements.

- **Systems Development:** The process of creating an IS.

Example: Developing a new e-commerce platform involves designing, coding, and testing software to enable online transactions.

The Information System (2 of 2)

Performance Factors of an Information System: Various aspects influencing the efficiency and effectiveness of an IS.

- **Database Design and Implementation:** Crafting the structure and organization of databases to meet specific requirements.
- **Application Design and Implementation:** Developing software applications to interact with databases and fulfill user needs.
- **Administrative Procedures:** Protocols and practices governing the management and maintenance of databases and IS components.

Database Development: The iterative process of creating, refining, and optimizing databases to support business objectives.

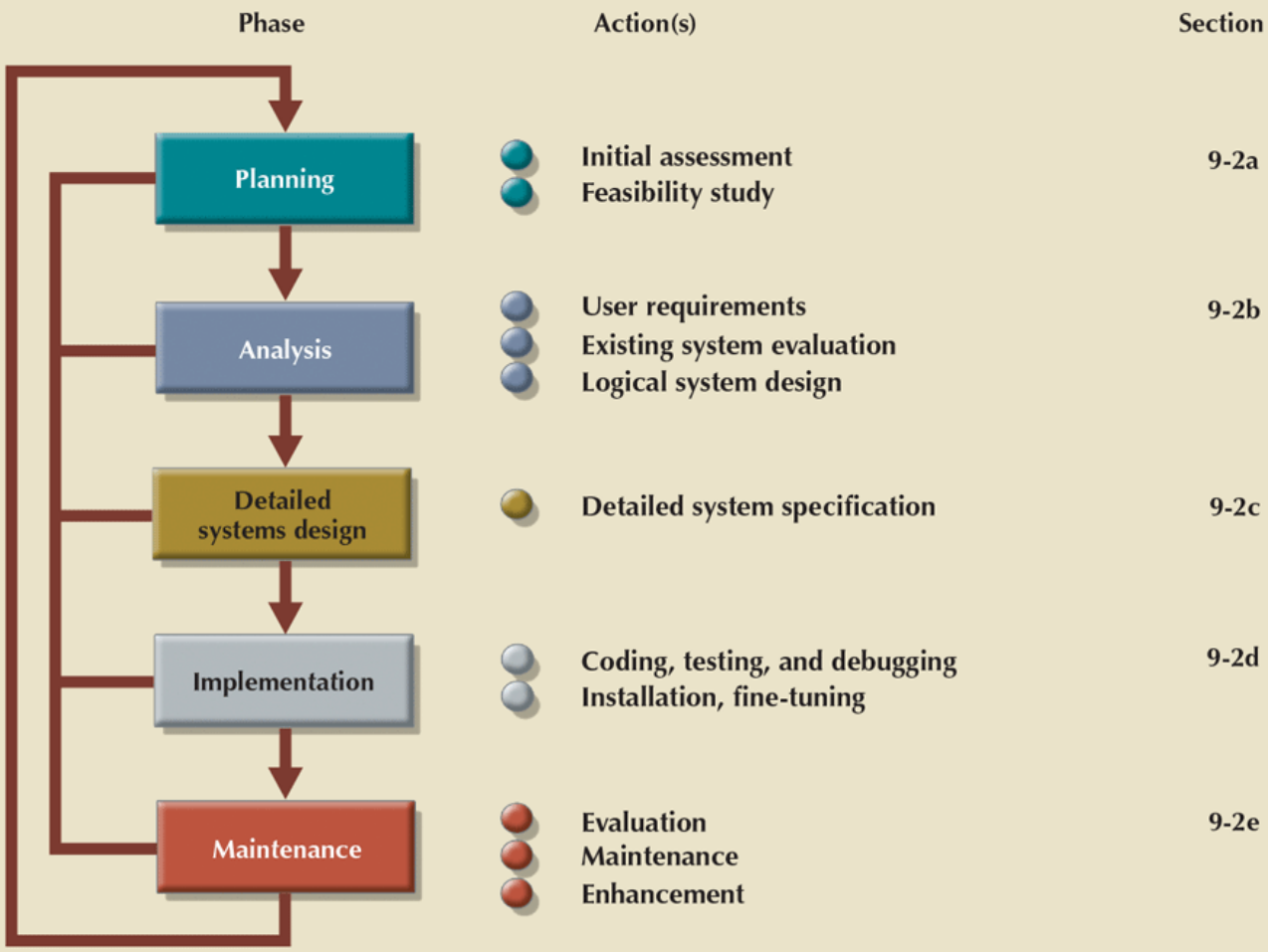
Process of Database Design and its Implementation: The systematic steps involved in designing, building, and deploying databases within an organization.

Systems Development Life Cycle (SDLC) (1 of 2)

- Traces history of an information system
  - Provides a picture within which database design and application development are mapped out and evaluated
- Traditional SDLC is divided into five phases
  1. **Planning:** yields a general overview of the company and its objectives
  2. **Analysis:** problems defined during planning phase are examined in greater detail
  3. **Detailed systems design:** designer completes the design of the system's processes
  4. **Implementation:** hardware, DBMS software, and application programs are installed, and the database design is implemented
  5. **Maintenance:** corrective, adaptive, and perfective
- Iterative rather than sequential process

Systems Development Life Cycle (SDLC) (2 of 2)

FIGURE 9.2 THE SYSTEMS DEVELOPMENT LIFE CYCLE (SDLC)

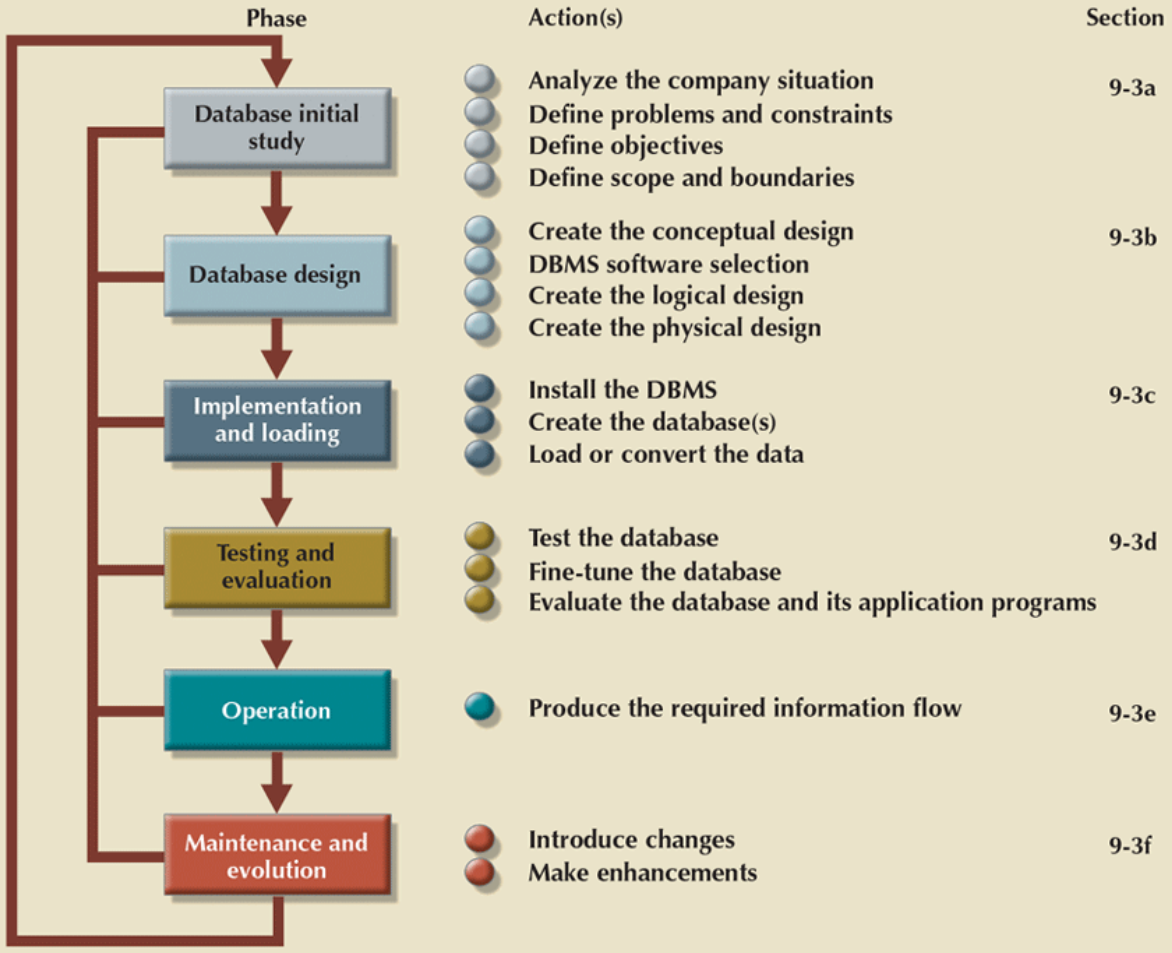


The Database Life Cycle (1 of 5)

- The Database Life Cycle (DBLC) contains six phases
  1. **Database initial study:** define problems, constraints, objectives, scope, and boundaries
  2. **Database design:** making sure that the final product meets user and system requirements
  3. **Implementation and loading:** DBMS is installed, database is created, and data is loaded or converted
  4. **Testing and evaluation:** database is tested, fine-tuned, and evaluated
     - Full backup/dump: all database objects are backed up in their entirety
     - Differential backup: only modified/updated objects since last full backup are backed up
     - Transaction log backup: only the transaction log operations that are not reflected in a previous backup are backed up
  5. **Operation:** problems are identified and solutions implemented
  6. **Maintenance and evolution:** preventative, corrective, adaptive, etc.

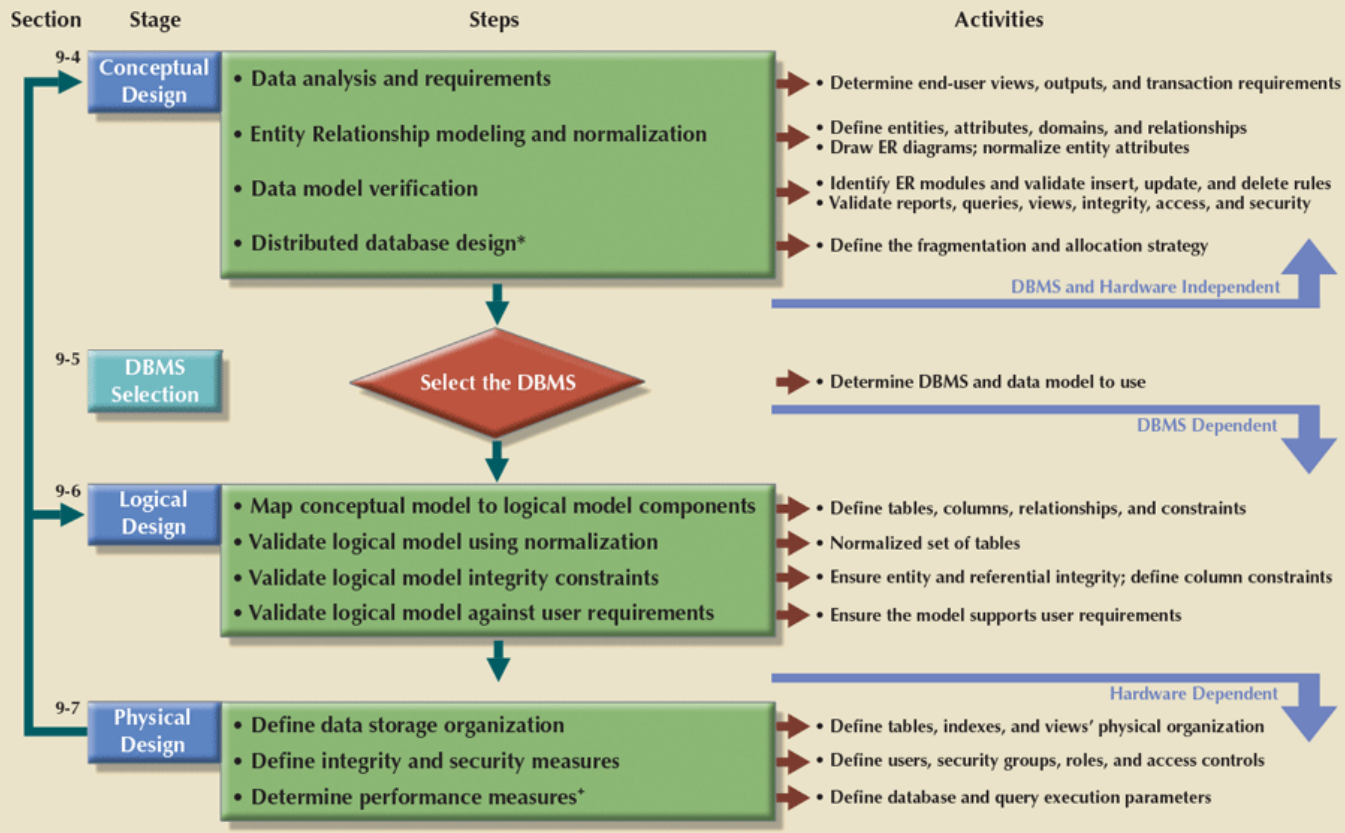
The Database Life Cycle (2 of 5)

FIGURE 9.3 THE DATABASE LIFE CYCLE (DBLC)



The Database Life Cycle (3 of 5)

FIGURE 9.6 DATABASE DESIGN PROCESS

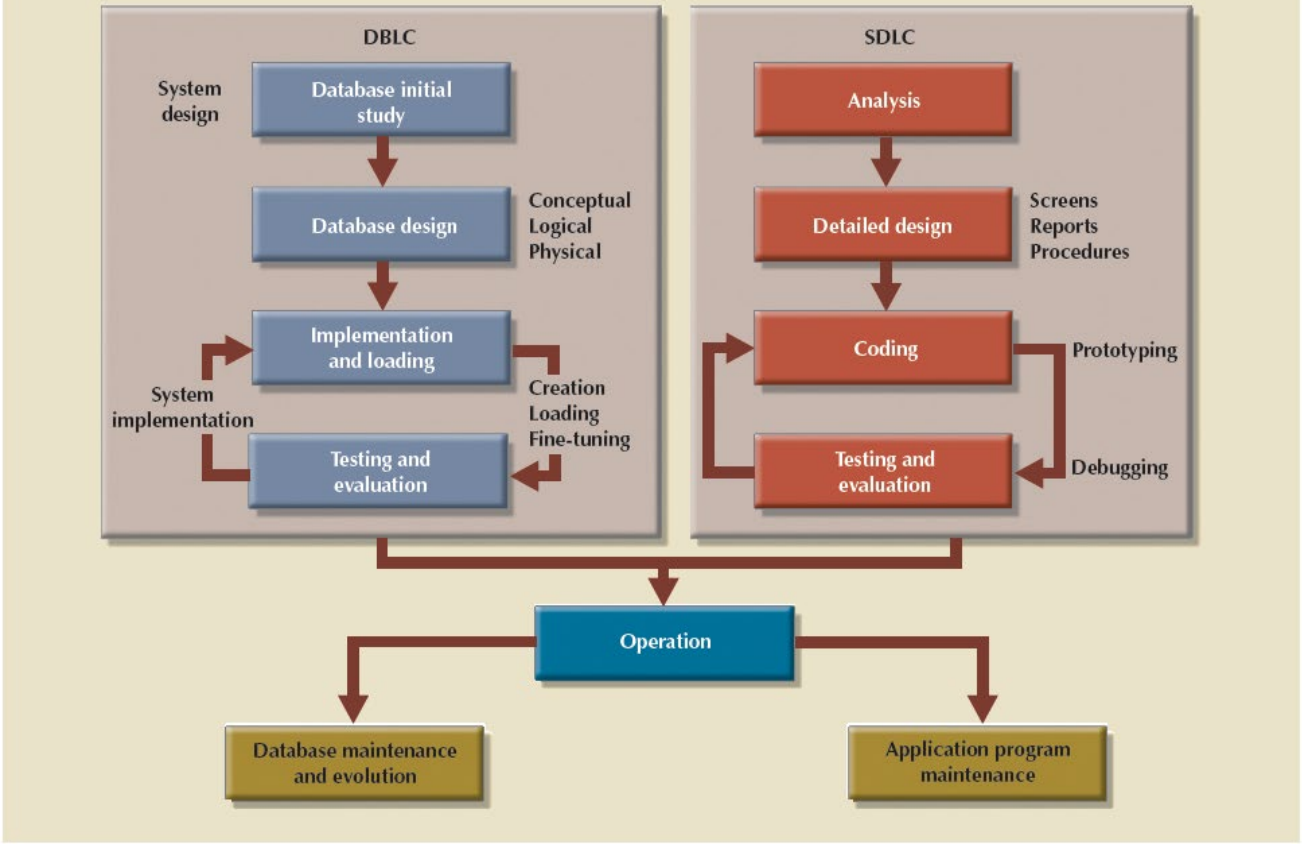


The Database Life Cycle (4 of 5)

Table 9.1: Common Sources of Database Failure		
Source	**Description**	**Example**
Software	Software-induced failures may be traceable to the operating system, the DBMS software, application programs, or viruses and other malware.	In April 2017, a new vulnerability was found in the Oracle E -Business Suite, that allows an unauthenticated attacker to create, modify, or delete critical data.
Hardware	Hardware-induced failures may include memory chip errors, disk crashes, bad disk sectors, and disk-full errors.	A bad memory module or a multiple hard disk failure in a database system can bring it to an abrupt stop.
Programming exemptions	Application programs or end users may roll back transactions when certain conditions are defined. Programming exemptions can also be caused by malicious or improperly tested code that can be exploited by hackers.	In February 2016 a group of unidentified hackers fraudulently instructed the New York Federal Reserve Bank to transfer $81 million from the central bank of Bangladesh to accounts in the Philippines. The hackers used fraudulent messages injected by malware disguised as a PDF reader.
Transactions	The system detects deadlocks and aborts one of the transactions. (See Chapter 10.)	Deadlock occurs when executing multiple simultaneous transactions.
External factors	Backups are especially important when a system suffers complete destruction from fire, earthquake, flood, or other natural disaster.	In August 2015, lightning struck a local utility provider's grid near Google's data centers in Belgium. Although power backup kicked in automatically, the interruption was long enough to cause permanent data loss in affected systems.

The Database Life Cycle (5 of 5)

FIGURE 9.8 PARALLEL ACTIVITIES IN THE DBLC AND THE SDLC



Conceptual Design (1 of 10)

Goal: Designing a database without consideration for specific database software or physical implementation details.

Conceptual Data Model: Depicts primary data entities, attributes, relationships, and constraints in a software and hardware agnostic manner.

Minimum Data Rule: Ensuring that the database contains all necessary information and eliminates redundant or unnecessary data.

In a conceptual data model for a university, the main entities could include "Student," "Course," and "Professor," with attributes such as "StudentID," "CourseID," and "ProfessorID." Relationships would indicate that a student enrolls in courses taught by professors. The minimum data rule would ensure that only essential information, like student details, course information, and professor credentials, is included, avoiding unnecessary data duplication.

Conceptual Design (2 of 10)

	Table 9.2: Conceptual Design Steps
Step	**Activity**
1	Data analysis and requirements
2	Entity relationship modeling and normalization
3	Data model verification
4	Distributed database design

Conceptual Design (3 of 10)

- Data analysis and requirements
  - Designers efforts are focused
    - Information needs, users, sources and constitution
  - Answers obtained from a variety of sources
    - Developing and gathering end-user data views
    - Directly observing current system: existing and desired output
    - Interfacing with the systems design group
- Entity relationship modeling and normalization
  - All objects (entities, attributes, relations, views, and so on) are defined in a data dictionary, which is used in alongside with the normalization process

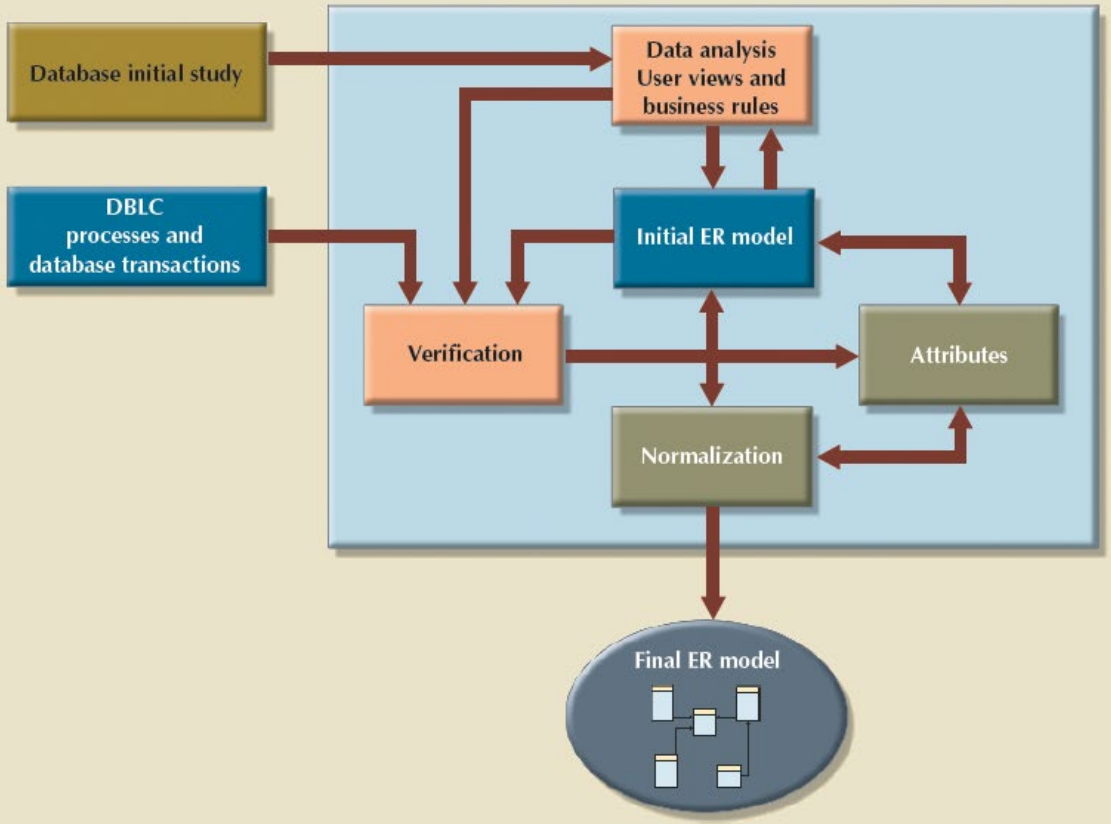
Data analysis focuses on understanding information needs, sources, and user requirements. Designers gather data views, observe current systems, and interface with design groups to create entity-relationship models and normalize data, all documented in a data dictionary.

Conceptual Design (4 of 10)

	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 end users' information and processing requirements
8	Modify the ER model, using the results of Step 7

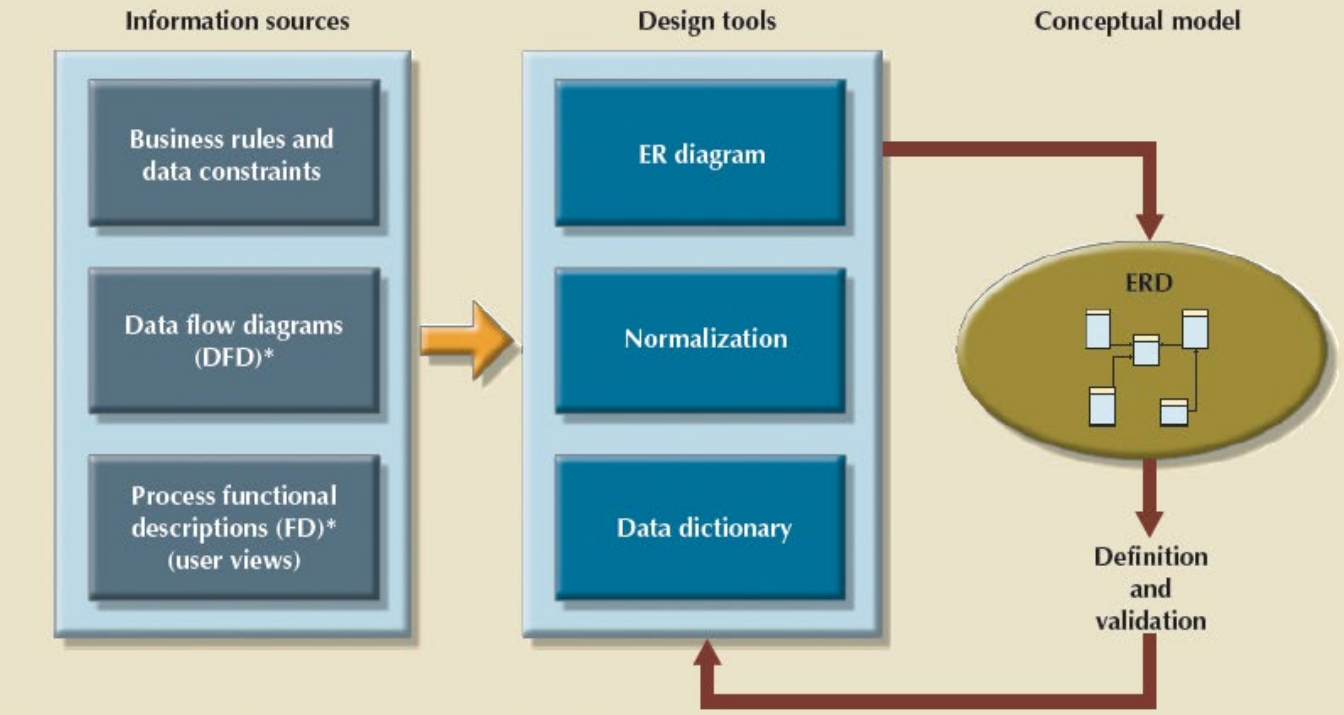
Conceptual Design (5 of 10)

FIGURE 9.10 ER MODELING IS AN ITERATIVE PROCESS BASED ON MANY ACTIVITIES



Conceptual Design (6 of 10)

FIGURE 9.11 CONCEPTUAL DESIGN TOOLS AND INFORMATION SOURCES



Conceptual Design (7 of 10)

Data model verification involves confirming that the proposed data model aligns with the system processes outlined for the intended system. This verification process typically includes running various tests to ensure the model's accuracy and effectiveness.

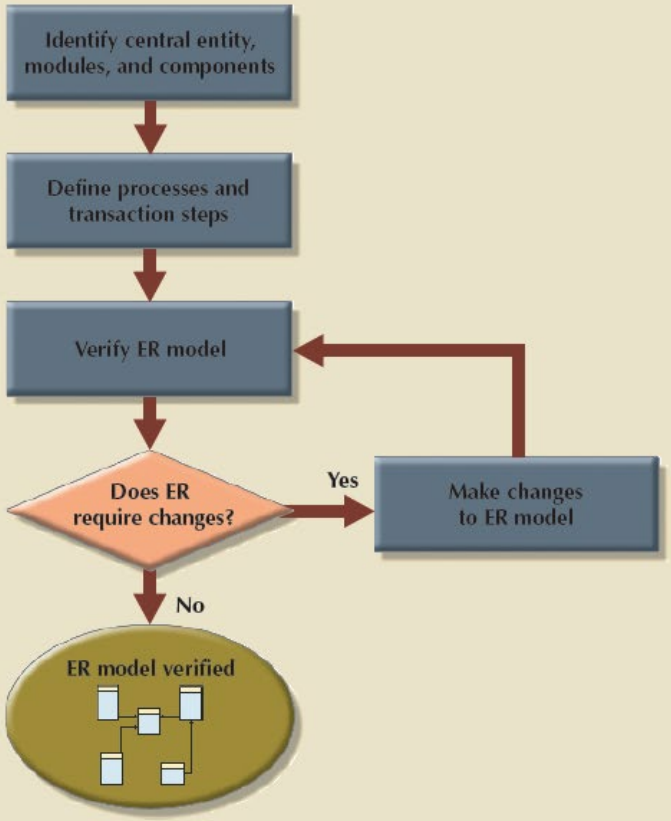
Key concepts in data model verification include modules, which are components of the information system that handle specific business functions. Cohesivity measures the strength of relationships among entities within a module, while module coupling refers to the extent of interdependence between modules. Low coupling is desirable as it reduces unnecessary dependencies between modules.

Conceptual Design (8 of 10)

	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 module's processing and reporting requirements
5	Make all necessary changes suggested in Step 4
6	Repeat Steps 2–5 for all modules

Conceptual Design (9 of 10)

FIGURE 9.12 ITERATIVE ER MODEL VERIFICATION PROCESS



Conceptual Design (10 of 10)

- Distributed database design
  - Portions of database may reside in different physical locations
  - Database fragment: subset of a database stored at a given location
  - Ensures database integrity, security, and performance

- Distributed database design involves organizing portions of a database across different physical locations. Each portion, known as a database fragment, is stored at a specific location. This design approach aims to maintain database integrity, enhance security, and optimize performance across distributed environments.

- Imagine a multinational corporation with branches in different countries. Each branch maintains its own database fragment containing local sales data. However, headquarters has access to all fragments, allowing them to analyze global sales performance while ensuring data consistency and security across locations.

DBMS Software Selection

Factors that needs to be studied for purchasing:
Cost: The monetary investment required to procure and maintain the DBMS software. Example: Purchasing licenses for a commercial DBMS like Oracle or Microsoft SQL Server.

DBMS Features and Tools: The functionalities and utilities provided by the DBMS software for managing and manipulating data. Example: Query optimization, backup and recovery tools, and security features offered by PostgreSQL.

Underlying Model: The theoretical framework or data model on which the DBMS is based, such as relational, hierarchical, or NoSQL. Example: MySQL is based on the relational model, while MongoDB is based on the document-oriented NoSQL model.

Portability: The ease with which the DBMS can be transferred or adapted to different hardware or software environments. Example: SQLite is known for its high portability, as it can run on various operating systems without modification.

DBMS Hardware Requirements: The specifications and resources (e.g., CPU, memory, storage) needed to support the operation of the DBMS. Example: Oracle Database may require high-performance servers with ample memory and storage capacity to handle large datasets efficiently.

Logical Design (1 of 3)

- Goal: design an enterprise-wide database that is based on a specific data model but independent of physical-level details
- Designing an enterprise-wide database involves creating a comprehensive database structure, such as organizing employee data, customer information, and product details, based on a chosen data model like the relational model or the object-oriented model. This design ensures efficient storage, retrieval, and management of data across the organization's various departments and functions.
- Designing an enterprise-wide database may involve organizing customer information, such as names, addresses, and contact details, into a customer table, while employee data, including employee IDs, names, and departments, is organized into an employee table. This design allows for easy access and management of crucial information for business operations.

Logical Design (2 of 3)

	Table 9.6: Logical Design Steps
Step	**Activity**
1	Map the conceptual model to logical model components
2	Validate the logical model using normalization
3	Validate the logical model integrity constraints
4	Validate the logical model against user requirements

Logical Design (3 of 3)

	Table 9.7: Mapping the Conceptual Model to the Relational Model
Step	**Activity**
1	Map strong entities
2	Map supertype/subtype relationships
3	Map weak entities
4	Map binary relationships
5	Map higher-degree relationships

Physical Design

- Process of data storage organization and data access characteristics of the database; ensures integrity, security, and performance
  - **Define data storage organization:** Structuring data within the database system for efficient storage and retrieval such as Storing customer information in separate tables based on demographics, purchases, and interactions.
  - **Define integrity and security measures:** Protocols and mechanisms to maintain data accuracy, consistency, and confidentiality, and prevent unauthorized access such as Implementing user authentication and encryption to safeguard sensitive financial data.
  - **Determine performance measures:** Assessment and optimization of database performance to ensure timely data access and processing, minimizing latency, and maximizing system responsiveness such as Monitoring query execution times and optimizing indexing to improve search speeds in an e-commerce database.

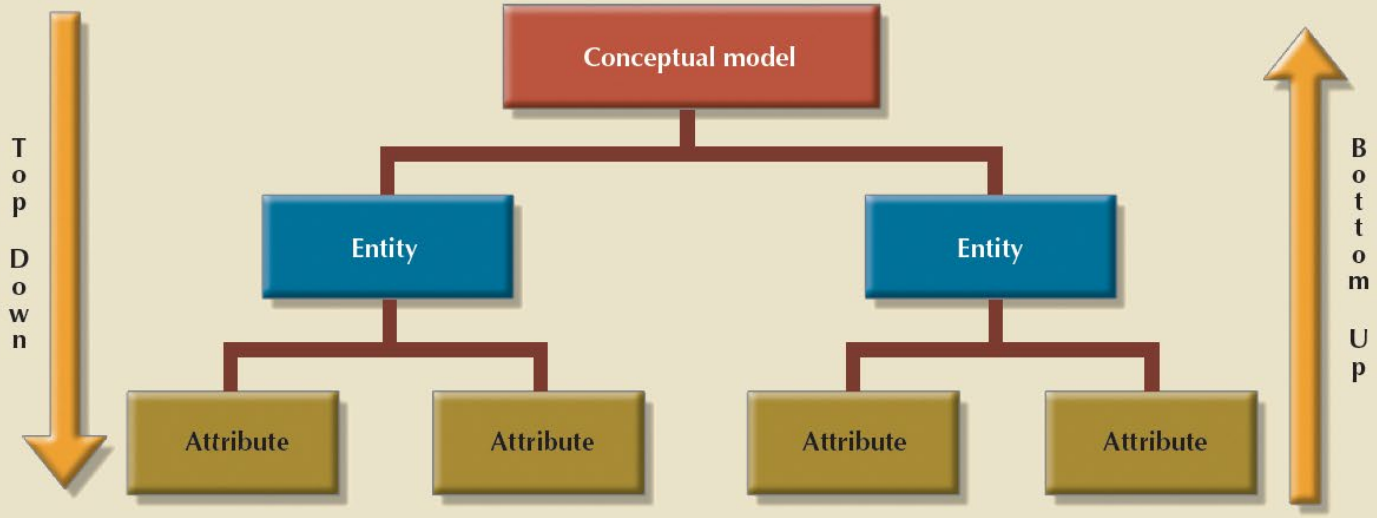
Database Design Strategies (1 of 2)

- **Top-down design:** Begins by identifying the main data sets or entities and then defining the specific data elements or attributes associated with each of those entities. For example, in designing a student information system, the design process may start by identifying entities such as "Students," "Courses," and "Instructors," and then defining the attributes for each entity, such as "Student ID," "Course Name," and "Instructor Name."

- **Bottom-up design:** Begins by identifying individual data elements or items and then grouping them together to form larger data sets or entities. For instance, in designing a database for inventory management, the design process may start by defining attributes like "Product ID," "Product Name," and "Unit Price," and then grouping these attributes together to form the "Products" entity.

Database Design Strategies (2 of 2)

FIGURE 9.14 TOP-DOWN VS. BOTTOM-UP DESIGN SEQUENCING



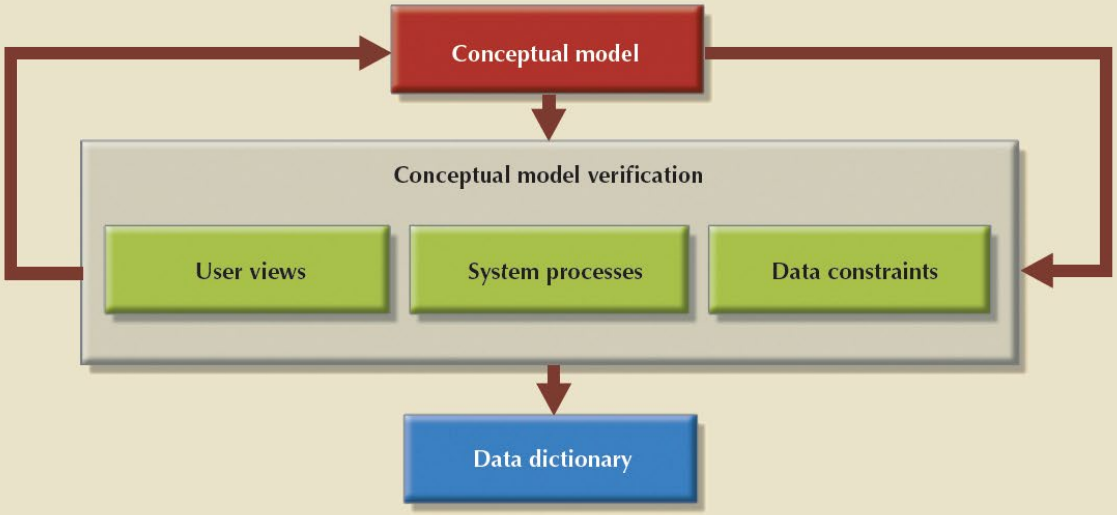
Centralized versus Decentralized Design (1 of 4)

- ***Centralized design:*** Involves making all database design decisions centrally by a small group, often suitable for smaller problem domains like a single department. For example, a centralized approach may be used to design a simple inventory management system for a small retail store.

- ***Decentralized design:*** Involves creating conceptual design models for subsets of an organization's database needs, which are later combined into a complete design. This approach is common in complex systems with many components, such as designing a distributed database system for a large multinational corporation.

Centralized versus Decentralized Design (2 of 4)

FIGURE 9.15 CENTRALIZED DESIGN

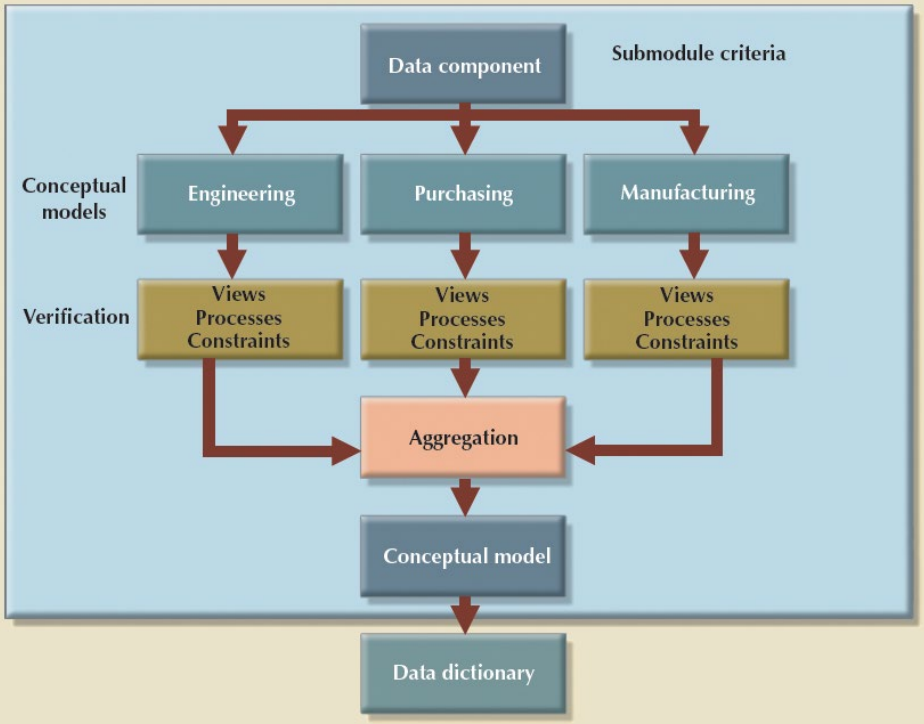


Centralized design refers to a database design approach where all decisions regarding the design are made centrally by a small group of individuals or a single team. This approach is typically used when the problem domain is relatively small, such as within a single unit or department of an organization. In centralized design, the entire database design process, including conceptualization, modeling, and implementation, is carried out by this central group, ensuring consistency and coherence across the database. It allows for efficient communication and coordination but may lack the flexibility needed for larger or more diverse organizational requirements.

Centralized versus Decentralized Design (3 of 4)

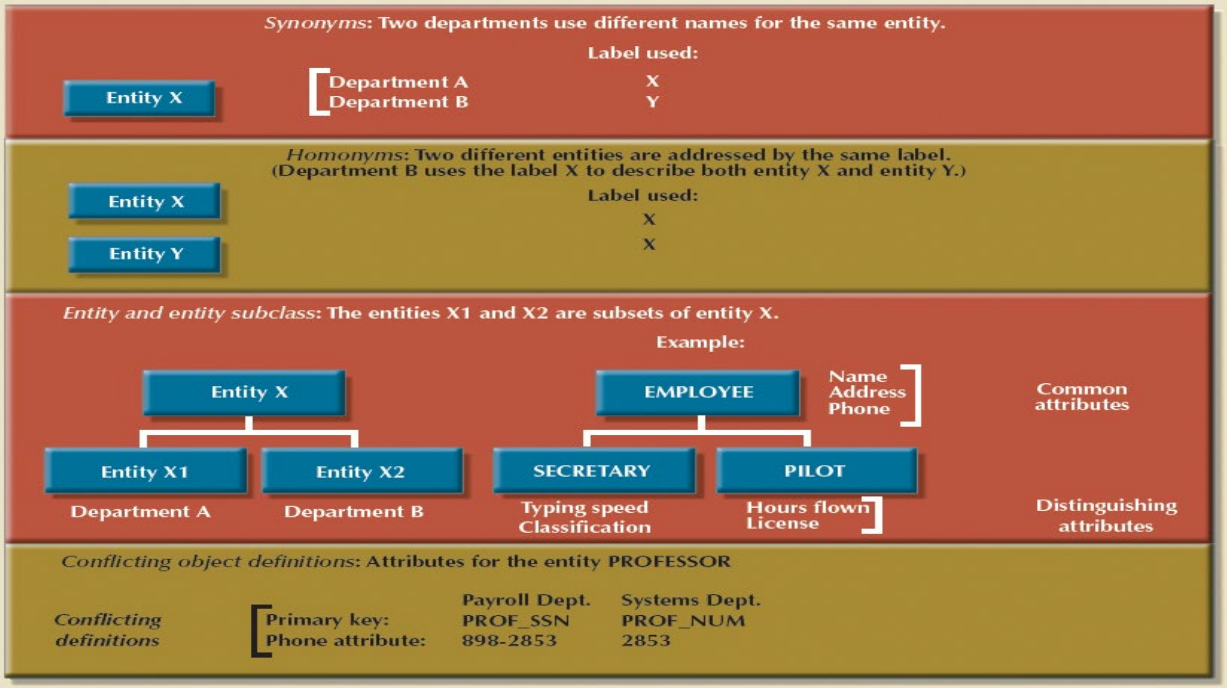
Decentralized design involves breaking down the database design process into smaller, more manageable parts, with different teams or individuals responsible for designing specific subsets of the database. Each subset, or module, focuses on a particular aspect of the overall database requirements. These individual designs are then integrated or aggregated to form the complete database design. This approach allows for greater specialization and parallel development efforts, making it suitable for complex systems with diverse requirements and a large number of stakeholders.

FIGURE 9.16 DECENTRALIZED DESIGN



Centralized versus Decentralized Design (4 of 4)

FIGURE 9.17 SUMMARY OF AGGREGATION PROBLEMS



In decentralized design, the aggregation problem arises when individual units or departments within an organization independently design subsets of the database without considering the broader organizational requirements. This can lead to inconsistencies, redundancies, and difficulties in integrating these subsets into a cohesive and comprehensive database.

For example, suppose a large university adopts a decentralized approach to database design. Each department within the university designs its own database to manage student records, course information, and faculty data. However, without coordination or communication between departments, inconsistencies may arise in data representation, naming conventions, or data formats. For instance, the Department of Computer Science may use different identifiers for courses compared to the Department of Mathematics, making it challenging to integrate these datasets into a unified university-wide database.

Summary

- An information system is designed to help transform data into information and to manage both data and information
- The Systems Development Life Cycle (SDLC) traces the history of an application within the information system
- The Database Life Cycle (DBLC) describes the history of the database within the information system
- The conceptual portion of the design may be subject to several variations based on two basic design philosophies: bottom-up versus top-down and centralized versus decentralized