Database Principles: Fundamentals of Design, Implementation, and Management

Tenth Edition

Database Development Process

Objectives

- In this chapter, you will learn:
  - That successful database design must reflect the information system of which the database is a part
  - That successful information systems are developed within a framework known as the Systems Development Life Cycle (SDLC)
  - That within the information system, the most successful databases are subject to frequent evaluation and revision within a framework known as the Database Life Cycle (DBLC)
  - How to conduct evaluation and revision within the SDLC and DBLC frameworks
  - About database design strategies: top-down vs. bottom-up design

The Information System

- Provides for data collection, storage, and retrieval
- Composed of:
  - People, hardware, software
  - Database(s), application programs, procedures
- Systems analysis
  - Process that establishes need for and extent of information system
- Systems development
  - Process of creating information system

The Information System (cont'd.)

- Applications
  - Transform data into information that forms basis for decision making
  - Usually produce the following:
    - Formal report
    - Tabulations
    - Graphic displays
  - Composed of the following two parts:
    - Data
    - Code: program instructions

FIGURE 2.1 Generating information for decision making

SOURCE: Course Technology/Cengage Learning

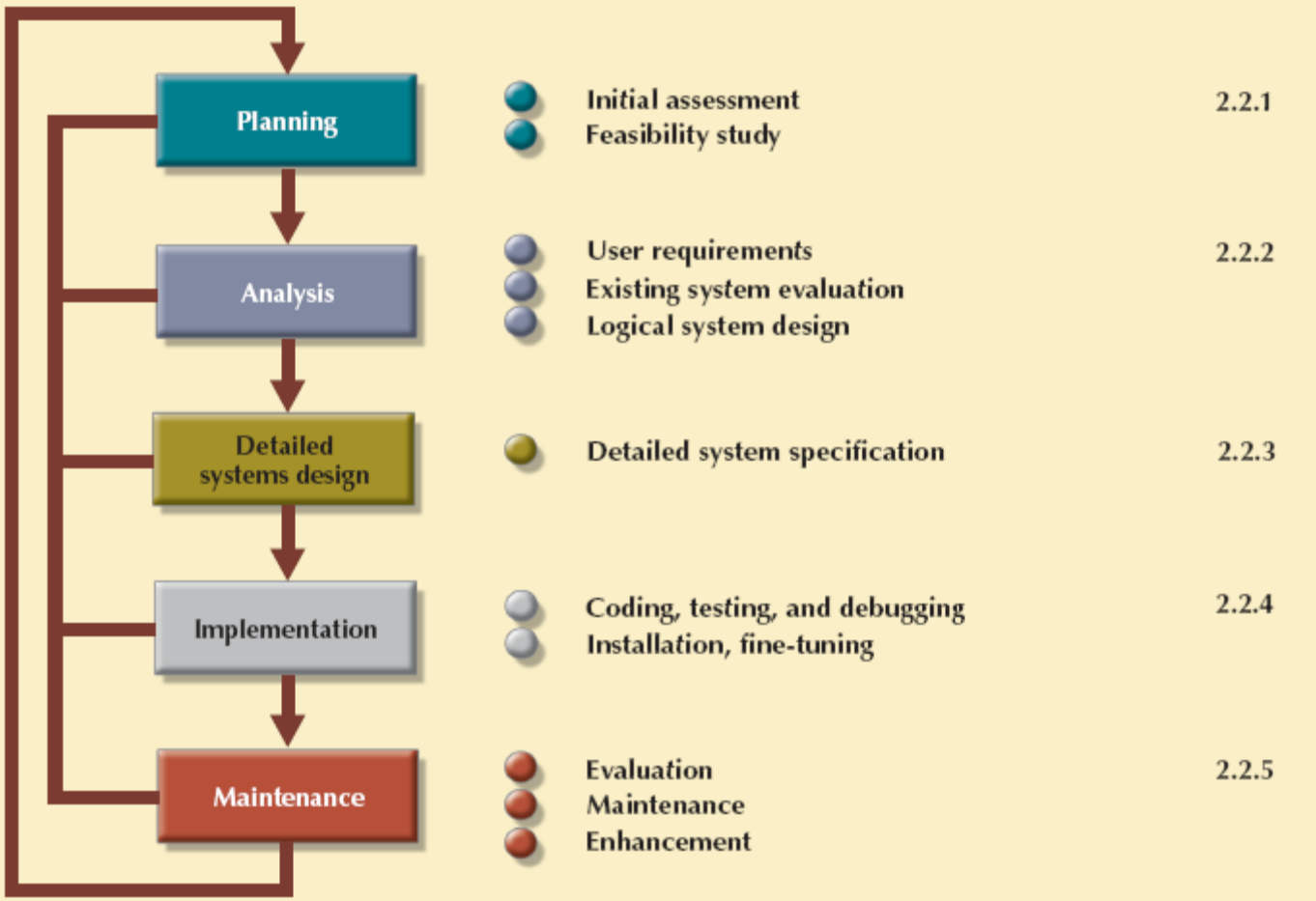
The Information System (cont'd.)

- Performance depends on three factors:
  - Database design and implementation
  - Application design and implementation
  - Administrative procedures
- Database development
  - Process of database design and implementation
  - Implementation phase includes:
    - Creating database storage structure
    - Loading data into the database
    - Providing for data management

The Systems Development Life Cycle (SDLC)

- Traces history (life cycle) of information system
- Database design and application development mapped out and evaluated
- Divided into following five phases:
  - Planning
  - Analysis
  - Detailed systems design
  - Implementation
  - Maintenance
- Iterative rather than sequential process

FIGURE 2.2 The Systems Development Life Cycle



SOURCE: Course Technology/Cengage Learning

Planning

- General overview of company and objectives
- Assessment of flow-and-extent requirements
  - Should the existing system be continued?
  - Should the existing system be modified?
  - Should the existing system be replaced?
- Study and evaluate alternate solutions
  - Technical aspects of hardware and software requirements
  - System cost
  - Operational cost

Analysis

- Problems defined during planning phase are examined in greater detail during analysis
- Thorough audit of user requirements
- Existing hardware and software systems are studied
- Goal:
  - Better understanding of:
    - System's functional areas
    - Actual and potential problems
    - Opportunities

Detailed Systems Design

- Designer completes design of system's processes
- Includes all necessary technical specifications
- Steps laid out for conversion from old to new system
- Training principles and methodologies are also planned
  - Submitted for management approval

Implementation

- Hardware, DBMS software, and application programs are installed
  - Database design is implemented
- Cycle of coding, testing, and debugging continues until database is ready for delivery
- Database is created and system is customized
  - Creation of tables and views
  - User authorizations

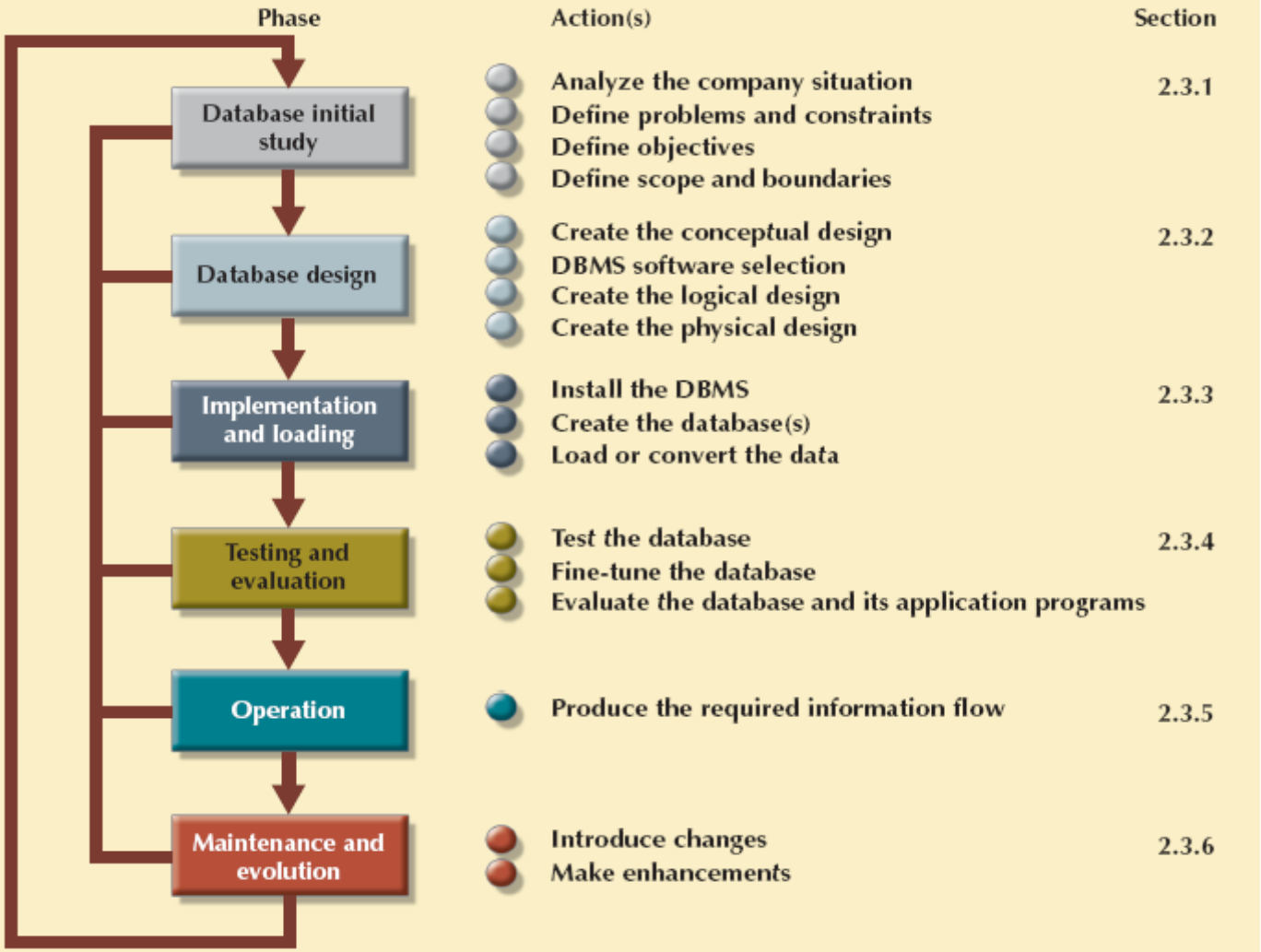
Maintenance

- Three types of maintenance activity:
  - Corrective maintenance
  - Adaptive maintenance
  - Perfective maintenance
- Computer-aided systems engineering (CASE)
  - Produce better systems within reasonable amount of time and at reasonable cost
  - CASE-produced applications are structured, documented, and standardized

The Database Life Cycle (DBLC)

- Six phases:
  - Database initial study
  - Database design
  - Implementation and loading
  - Testing and evaluation
  - Operation
  - Maintenance and evolution

FIGURE 2.3 The Database Life Cycle

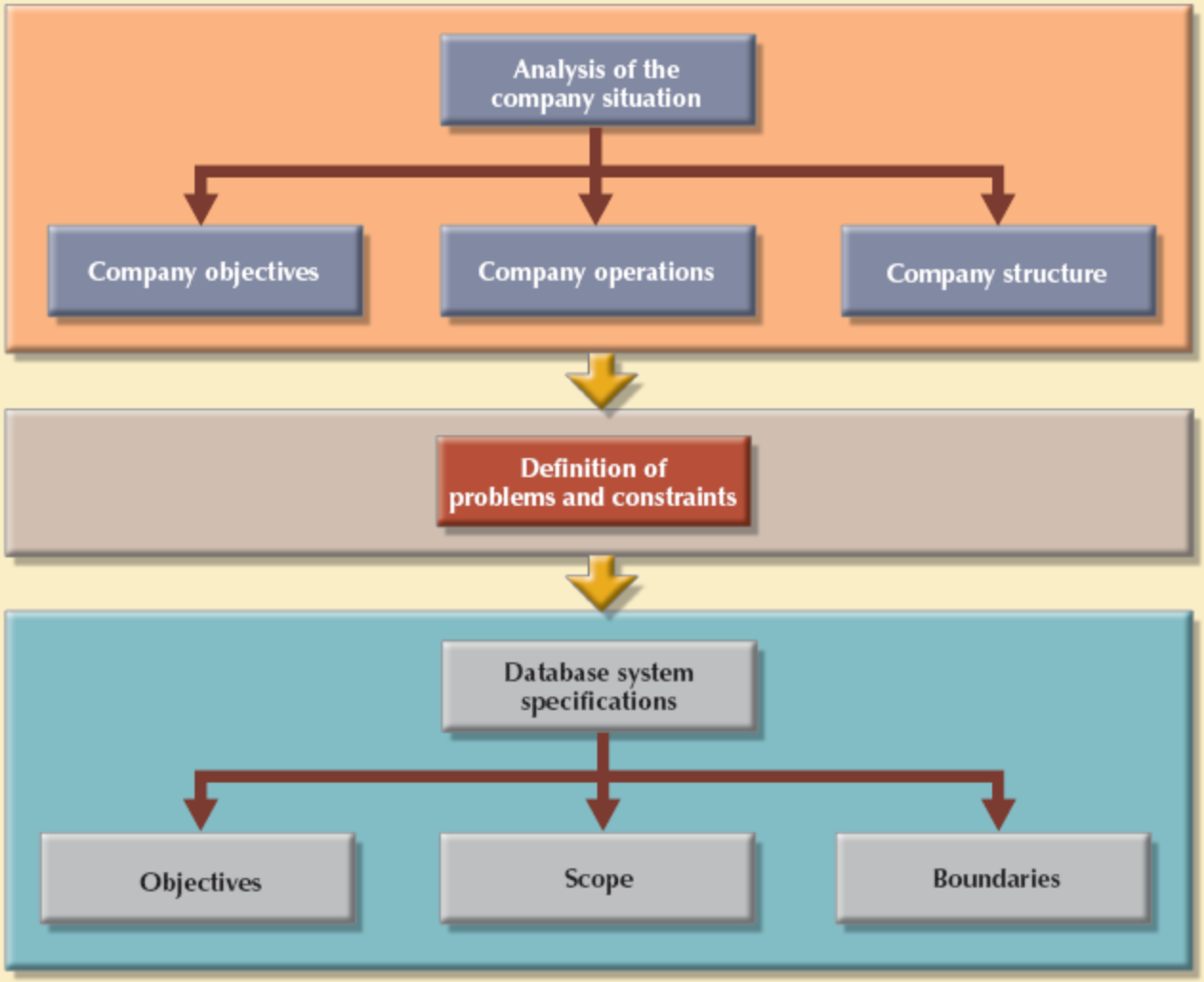


SOURCE: Course Technology/Cengage Learning

The Database Initial Study

- Overall purpose:
  - Analyze company situation
  - Define problems and constraints
  - Define objectives
  - Define scope and boundaries
- Interactive and iterative processes required to complete first phase of DBLC successfully

FIGURE 2.4 A summary of activities in the database initial study



SOURCE: Course Technology/Cengage Learning

The Database Initial Study (cont'd.)

- Analyze the company situation
  - General conditions in which company operates, its organizational structure, and its mission
  - Discover what company's operational components are, how they function, and how they interact

The Database Initial Study (cont'd.)

- Define problems and constraints
  - Formal and informal information sources
  - Finding precise answers is important
  - Accurate problem definition does not always yield a solution

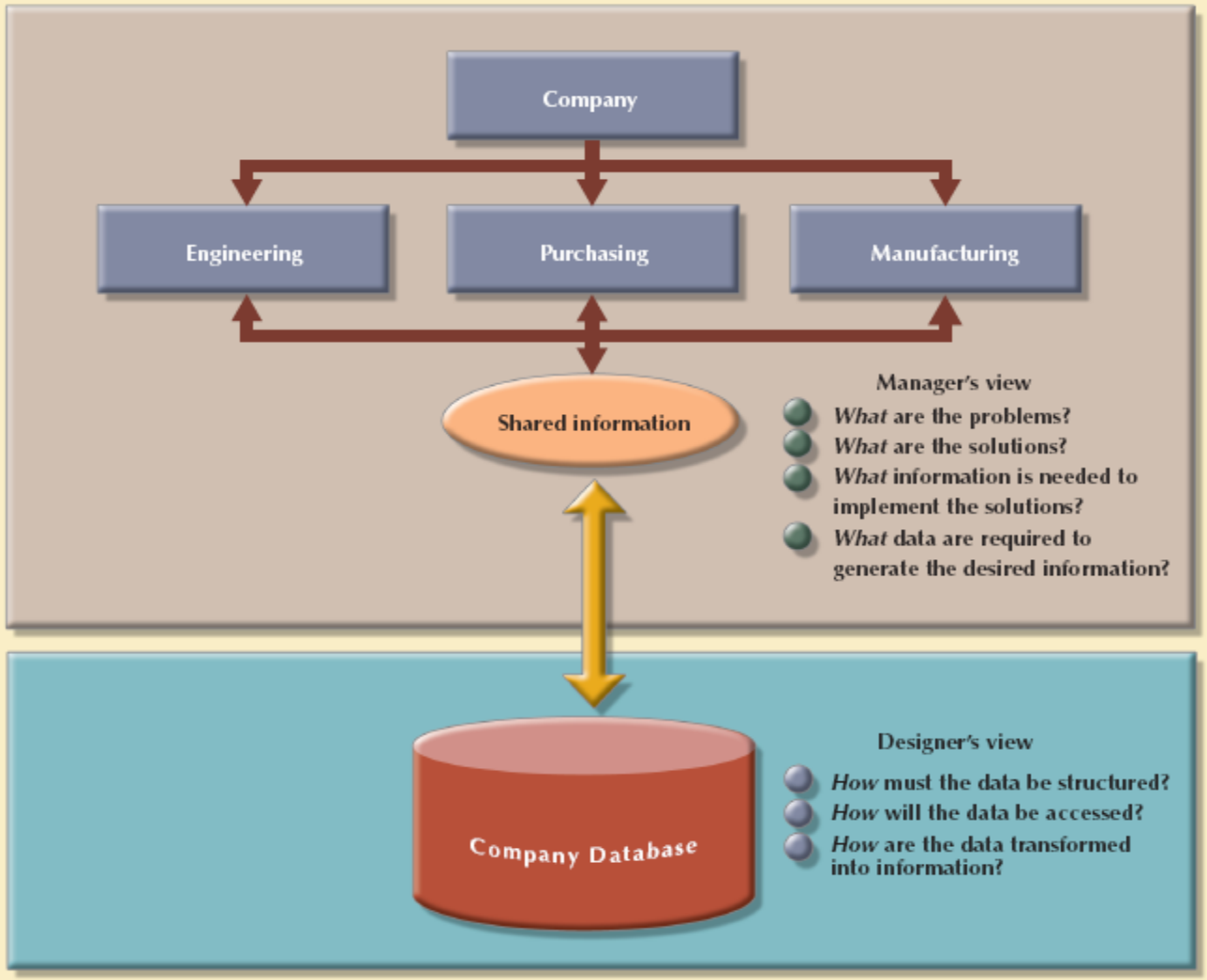
The Database Initial Study (cont'd.)

- Database system objectives must correspond to those envisioned by end users
  - What is proposed system's initial objective?
  - Will system interface with other systems in the company?
  - Will system share data with other systems or users?
- Scope: extent of design according to operational requirements
- Boundaries: limits external to system

Database Design

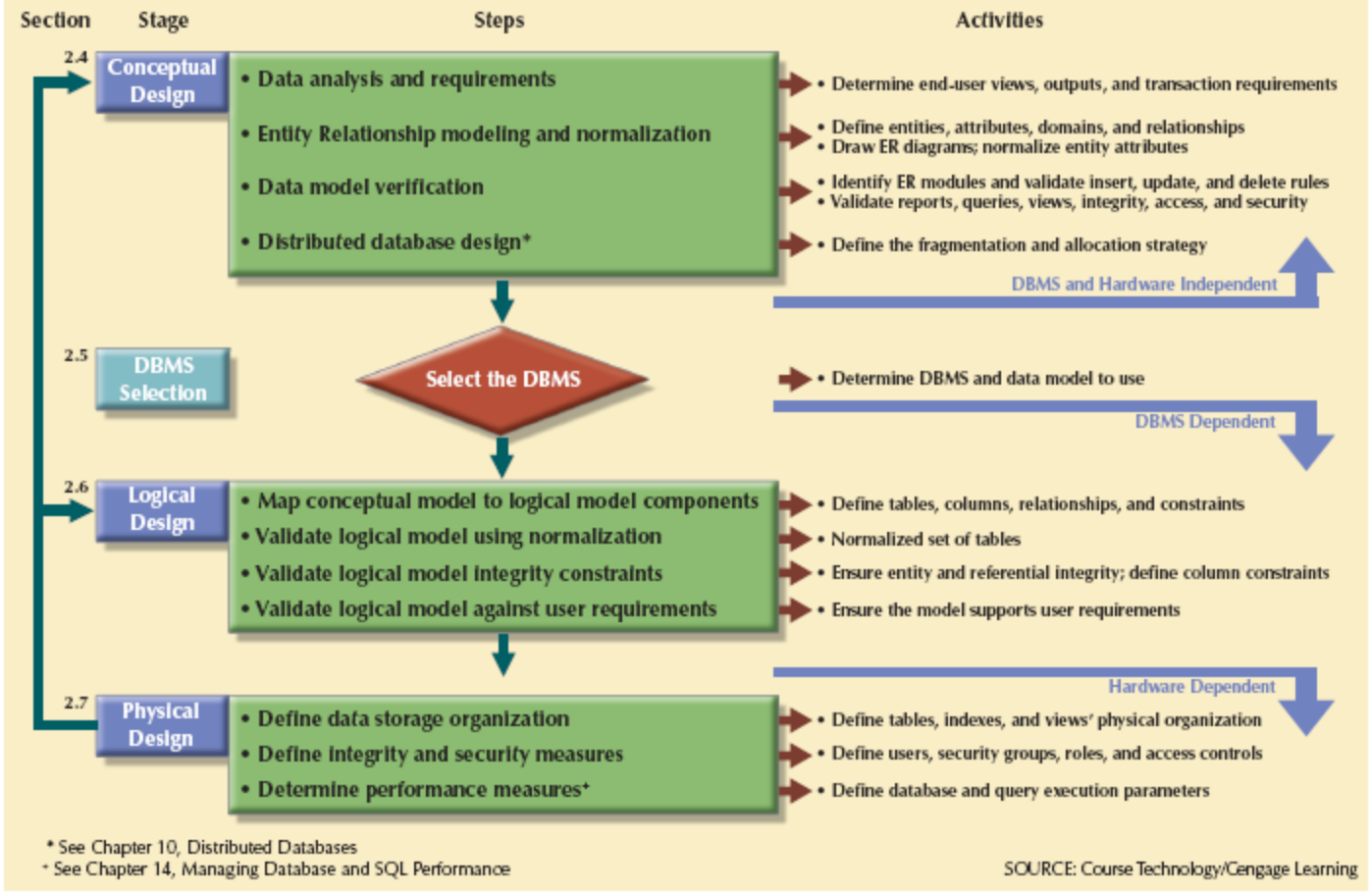
- Necessary to concentrate on data characteristics required to build database model
- Two views of data within system:
  - Business view
    - Data as information source
  - Designer's view
    - Data structure, access, and activities required to transform data into information

FIGURE 2.5 Two views of data: business manager and database designer



SOURCE: Course Technology/Cengage Learning

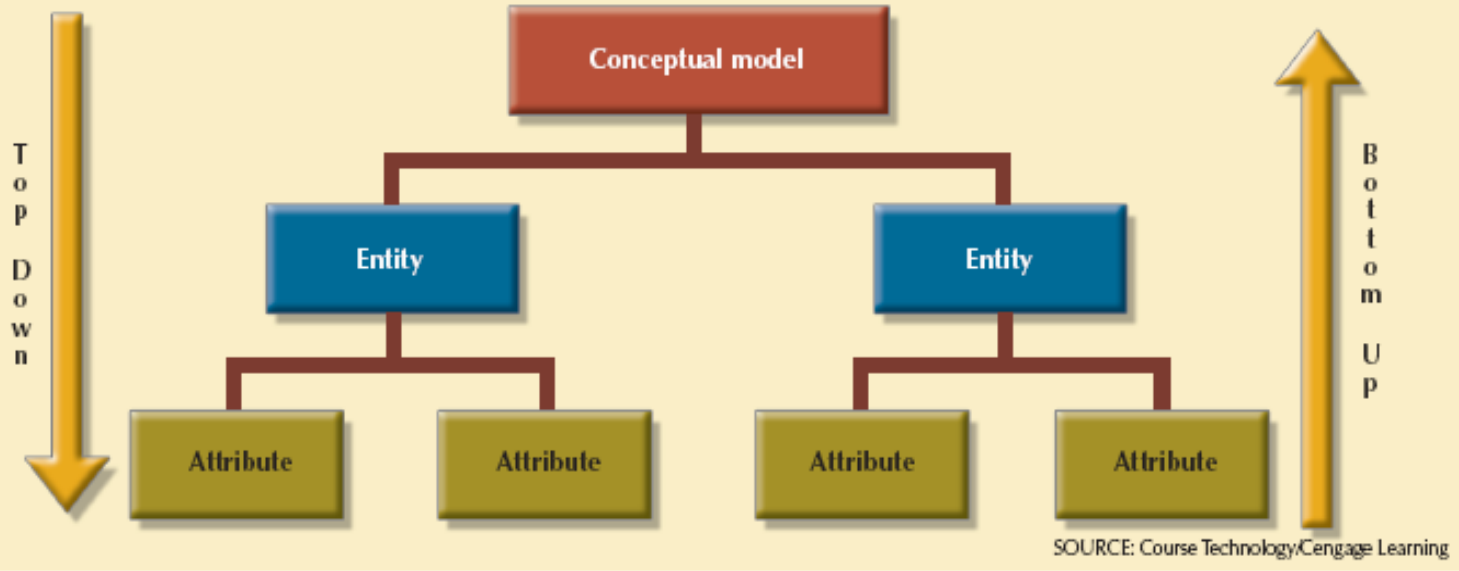
FIGURE 2.6 Database design process



Database Design Strategies

- Top-down design
  - Identifies data sets
  - Defines data elements for each of those sets
    - Definition of different entity types
    - Definition of each entity's attributes
- Bottom-up design
  - Identifies data elements (items)
  - Groups them together in data sets

FIGURE 2.14 Top-down vs. bottom-up design sequencing



SOURCE: Course Technology/Cengage Learning

Implementation and Loading

- Actually implement all design specifications from previous phase:
  - Install the DBMS
    - Virtualization: creates logical representations of computing resources independent of physical resources
  - Create the Database
  - Load or Convert the Data

Testing and Evaluation

- Occurs in parallel with applications programming
- Database tools used to prototype applications
- If implementation fails to meet some of system's evaluation criteria:
  - Fine-tune specific system and DBMS configuration parameters
  - Modify physical or logical design
  - Upgrade software and/or hardware platform

Testing and Evaluation (cont'd.)

- Integrity
  - Enforced via proper use of primary, foreign key rules
- Backup and Recovery
  - Full backup
  - Differential backup
  - Transaction log backup

Operation

- Once database has passed evaluation stage, it is considered operational
- Beginning of operational phase starts process of system evolution
- Problems not foreseen during testing surface
- Solutions may include:
  - Load-balancing software to distribute transactions among multiple computers
  - Increasing available cache

Maintenance and Evolution

- Required periodic maintenance:
  - Preventive maintenance (backup)
  - Corrective maintenance (recovery)
  - Adaptive maintenance
  - Assignment of access permissions and their maintenance for new and old users
  - Generation of database access statistics
  - Periodic security audits
  - Periodic system-usage summaries

Maintenance and Evolution (more info)

- **Nearly all systems will need maintenance over their lifetime. This is because:**
  - ✓ Something in the system needs fixing (*corrective maintenance*)
  - ✓ External changes forces a change to the system (*adaptive maintenance*)
  - ✓ Something can be improved (*perfective maintenance*)

- **There are many reasons for maintaining a system that fall into the categories given above:**
  - ✓ An error / bug is serious enough to need fixing
  - ✓ A new business process needs to be incorporated.
  - ✓ A security weakness in the system has been found and needs fixing.
  - ✓ An user has identified how the system could be improved
  - ✓ The hardware or network is being improved and so the system should take advantage of that.

Maintenance and Evolution (more info)

Adaptive maintenance

This type of maintenance often occurs as a result of external influences or strategic changes within the company. The system is being *adapted* to remain up to date.

Example 1

- The Government recently changed the VAT rate from 17.5% to 20%. This change meant that many organizations had to make alterations to their systems.

Example 2

- A bank decides to offer a new mortgage product. This will have to be included in the system so that mortgage interest and payments can be calculated.

Example 3

- An company has introduced a online system for customers to place orders. The online system needs to be integrated into their normal ordering system.

Maintenance and Evolution (more info)

Perfective maintenance

- The system has been in place and running fine for a while.
- However, over time, the end user will often find tweaks or minor improvements which could be made to improve the way the system works.

Examples of making the system 'more perfect' include

- A better data input screen or form
- A more advanced help system
- Handling error conditions in a more useful way e.g. instead of a cryptic "Error 3045 occurred" warning, the screen actually explains what that means and what the user can do about it.
- Re-organising data sets within a database so they can be searched faster or use less storage
- Providing shortcuts commands that experts can use instead of the slower standard menu system

Maintenance and Evolution (more info)

Corrective maintenance

- Most systems have imperfections. The initial testing of the system should find many of them but more obscure errors may only be encountered as users interact with the system day after day.
- Part of the system should include a way for customers / users to report these problems. Also error logs should be included in the system so maintenance staff can spot problems even if they are not reported.

Software / programming errors

- Almost all commercial software systems seem to need fixing even after they have been released.
- Software code can be very complex so even the most diligent test regime fails to find all of the errors.
- The way these are fixed is often in the form of a *software patch* that the user downloads or it is sent to their computer by the IT staff over the company network..
- Another common corrective maintenance action is to fix security vulnerabilities in the code. Again, the usual way of doing this is by software patch.

Logical or business process errors

- This is where the system is not behaving in the way it was intended, not because of a programming error but because the coder misunderstood what was needed to support the business in the first place.
- For example, a database may be storing data in such a way that it is causing problems for other systems within the company. These types of errors may be much harder to fix compared to straight forward programming errors because they may be a fundamental part of the system.
- This is why a robust requirements document is needed to avoid logical and business process errors.

Summary

- Information system facilitates transformation of data into information
  - Manages both data and information
- SDLC traces history (life cycle) of an application within the information system
- DBLC describes history of database within the information system
- Database design and implementation process moves through series of well-defined stages
- Conceptual design subject to several variations:
  - Top-down vs. bottom-up