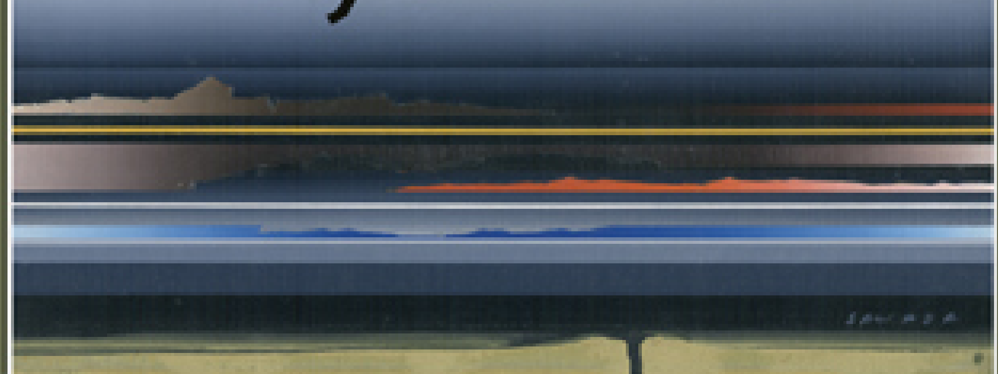
Fundamentals of

Database Systems

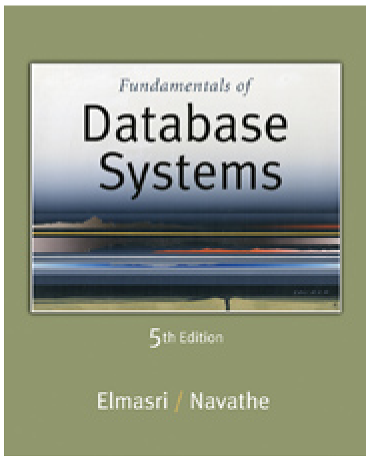
5th Edition



Elmasri / Navathe

Chapter 12

Practical Database Design Methodology and Use of UML Diagrams



Chapter Outline

- Information System Life Cycle
- Phases of Database Design
- UML Diagrams
  - Rational Rose
  - Other tools
- Design Tools

Organizational Context for using Database Systems

- Consolidation and integration of data across organization
- Maintenance of complex data
- Simplicity of developing new applications
- Data independence
  - Protecting application programs from changes in the underlying logical organization and in the physical access paths and storage structures
- External Schemas
  - Allow the same data to be used for multiple applications with each application having its own view of the data

Information System

- Information System includes all resources involved in the collection, management, use and dissemination of the information resources of the organization
- We consider two systems life cycles:
  - Macro Life Cycle
    - Information System Life Cycle
  - Micro Life Cycle
    - Database System Life Cycle

Phases of Information System Life Cycle

- Feasibility Analysis
  - Analyzing potential application areas
  - Identifying the economics of information gathering and dissemination
  - Performing cost benefit studies
  - Setting up priorities among applications
- Requirement Collection and Analysis
  - Detailed Requirements Collection
  - Interaction with Users
- Design
  - Design of Database System
  - Design of programs that use and process the database

Phases of Information System Life Cycle (contd.)

- Implementation
  - Information system is implemented
  - Database is loaded & its transactions are implemented and tested
- Validation and Acceptance Testing
  - Testing against user's requirements
  - Testing against performance criteria
- Deployment, Operation and Maintenance
  - Data conversion
  - Training
  - System maintenance
  - Performance monitoring
  - Database tuning

Database System Life Cycle

- System definition
  - Defining scope of database system, its users and applications
- Database Design
  - Logical and physical design of the database system on the chosen DBMS
- Database implementation
  - Specifying conceptual, external and internal database definitions
  - Creating empty database files
  - Implementing software applications

Database System Life Cycle (contd.)

- Loading or data conversion
  - Populating the database
- Application conversion
  - Converting applications to the new system
- Testing and validation
- Operation
  - Running the new system
- Monitoring and maintenance
  - System maintenance
  - Performance monitoring

Database Design Process

- Problem
  - Design the logical and physical structure of one or more databases to accommodate the information needs of the users in an organization for a defined set of applications.
- Goals
  - Satisfy the content requirements
  - Provide easy structuring of information
  - Support processing requirements and performance objectives

Phases of Database Design and Implementation Process

- Requirements Collections and Analysis
- Conceptual Database Design
- Choice of a DBMS
- Data Model Mapping (Logical Database Design)
- Physical Database Design
- Database System Implementation and Tuning

Phases of Database Design and Implementation Process (contd.)

- Requirements Collections and Analysis
  - Identifying Users
  - Interacting with users to gather requirements
  - Time consuming BUT very important
    - Very expensive to fix requirements error

- Conceptual Database Design
  - Produce a conceptual schema for the database that is independent of a specific DBMS
  - Involves two parallel activities
    - Conceptual Schema Design
    - Transaction and Application Design

Conceptual Schema Design

- Goal
  - Complete understanding of the database structure, semantics, interrelationships and constraints

- Serves as a stable description of the database contents
- Good understanding crucial for the users and designers
- Diagrammatic description serves as an excellent communication tool

Desired Characteristics of Conceptual Data Model

- Expressiveness
  - Able to distinguish different types of data, relationships and constraints
- Simplicity and Understandability
  - Easy to understand
- Minimality
  - Small number of distinct basic concepts
- Diagrammatic Representation
  - Diagrammatic notation for representing conceptual schema
- Formality
  - Formal unambiguous specification of data

Approaches to Conceptual Schema Design

- Centralized Schema Design Approach
  - Also known as one-shot approach
  - Requirements of different applications and user groups are merged into a single set of requirements and a single schema is designed
  - Time consuming, places the burden on DBA to reconcile conflicts
- View Integration Approach
  - Schema is designed for each user group or application
  - These schemas are then merged into a global conceptual schema during the view integration phase
  - More practical

Strategies for Schema Design

- Top Down Strategy
  - Start with a schema containing high-level abstractions and then apply successive top-down refinements

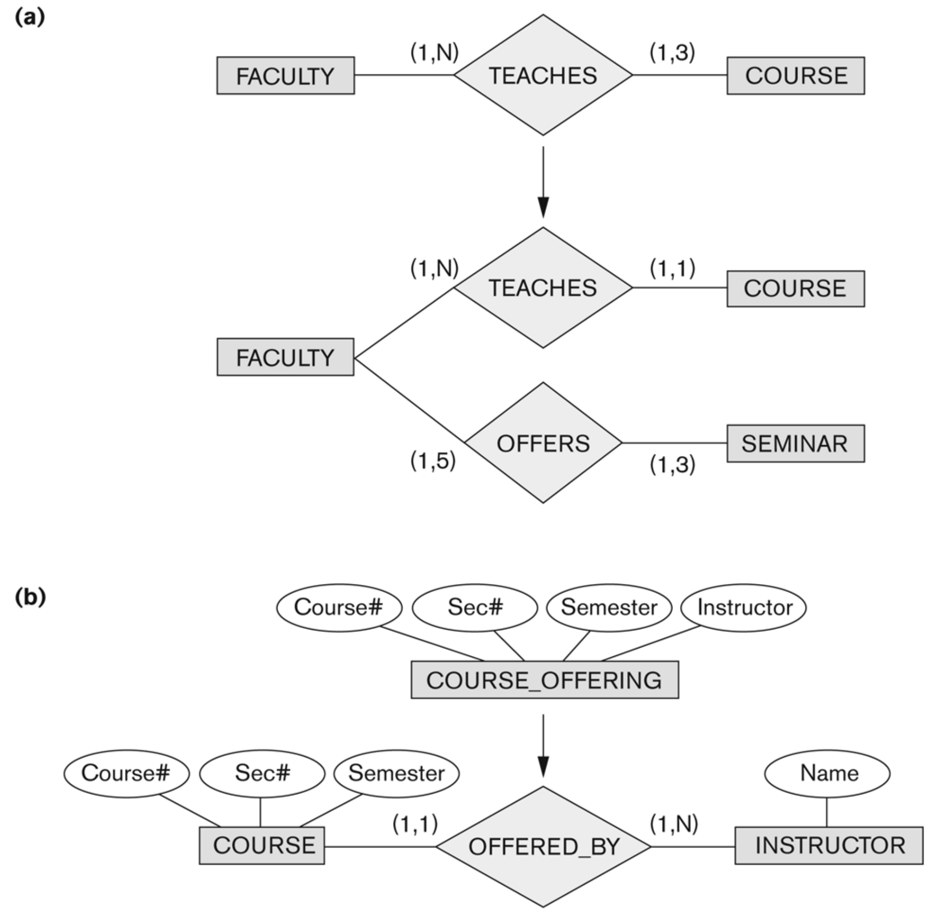


Figure 12.2
Examples of top-down refinement. (a) Generating a new entity type.
(b) Decomposing an entity type into two entity types and a relationship type.

Strategies for Schema Design (contd.)

- Bottom-Up Strategy
  - Start with a schema containing basics abstractions and then combine or add to these abstractions

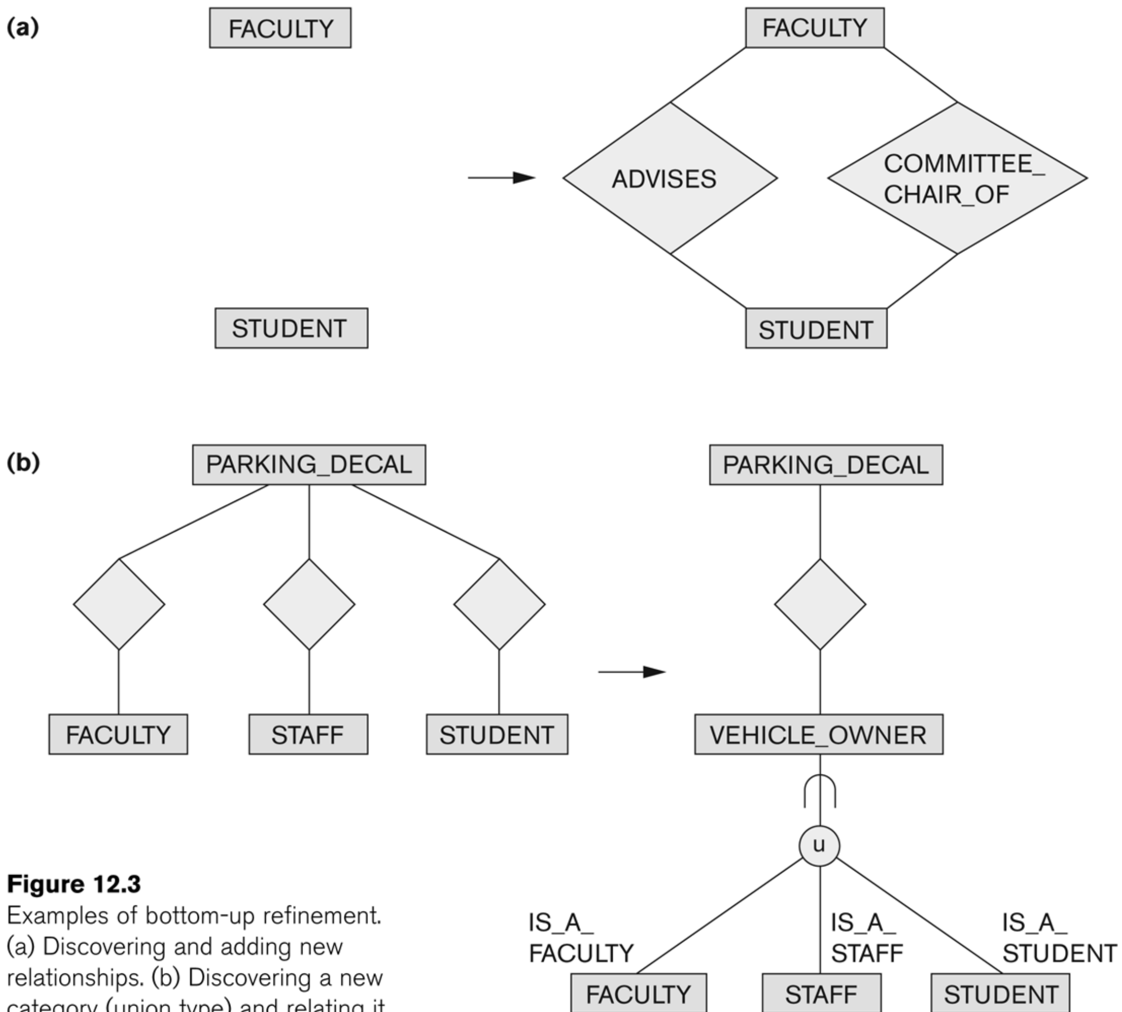


Figure 12.3
Examples of bottom-up refinement. (a) Discovering and adding new relationships. (b) Discovering a new category (union type) and relating it.

Strategies for Schema Design (contd.)

- Inside-out Strategy
  - Start with central set of concepts and then spread outward by considering new concepts in the vicinity of existing ones
- Mixed Strategy
  - Use a combination of top-down and bottom-up strategies

Schema Integration

- Identifying correspondence and conflict among different schemas
  - Naming conflicts
    - Synonyms: The same concept but different names
      - e.g. entity types CUSTOMER and CLIENT
    - Homonyms: Different concepts but same name
      - e.g. entity type PART as computer parts and furniture parts
  - Type Conflicts: Representing the same concept by different modeling constructs
    - e.g. DEPARTMENT may be an entity type and an attribute
  - Domain Conflicts: Attribute has different domains
    - Also known as value set conflicts
    - e.g. SSN as an integer and as a character string
  - Conflict among constraints: Two schemas impose different constraints
    - e.g. different key of an entity type in different schemas

Schema Integration (contd.)

- Modifying views to conform to one another
  - Modifying schemas to conform to one another
- Merging of views
  - Merging Schemas to create a global schema
  - Specifying mappings between views and global schema
    - Time consuming and difficult
- Restructuring
  - Simplifying and restructuring to remove any redundancies

View Integration Strategies

- Binary Ladder Integration
  - Two similar schemas are integrated first and the resulting schema is then integrated with another schema
  - The process is repeated until all schemas are integrated
- N-ary Integration
  - All views are integrated in one procedure after analysis and specification of their correspondences
    - Requires computerized tools for large designs

View Integration Strategies (contd.)

- Binary Balanced Strategy
  - Pairs of schemas are integrated first and the resulting schemas are then paired for further integration.
  - This process is repeated until a final global schema

- Mixed Strategy
  - Schemas are partitioned into groups based on their similarity and each group is integrated separately.
  - This process is repeated until a final global schema

View Integration Strategies (contd.)

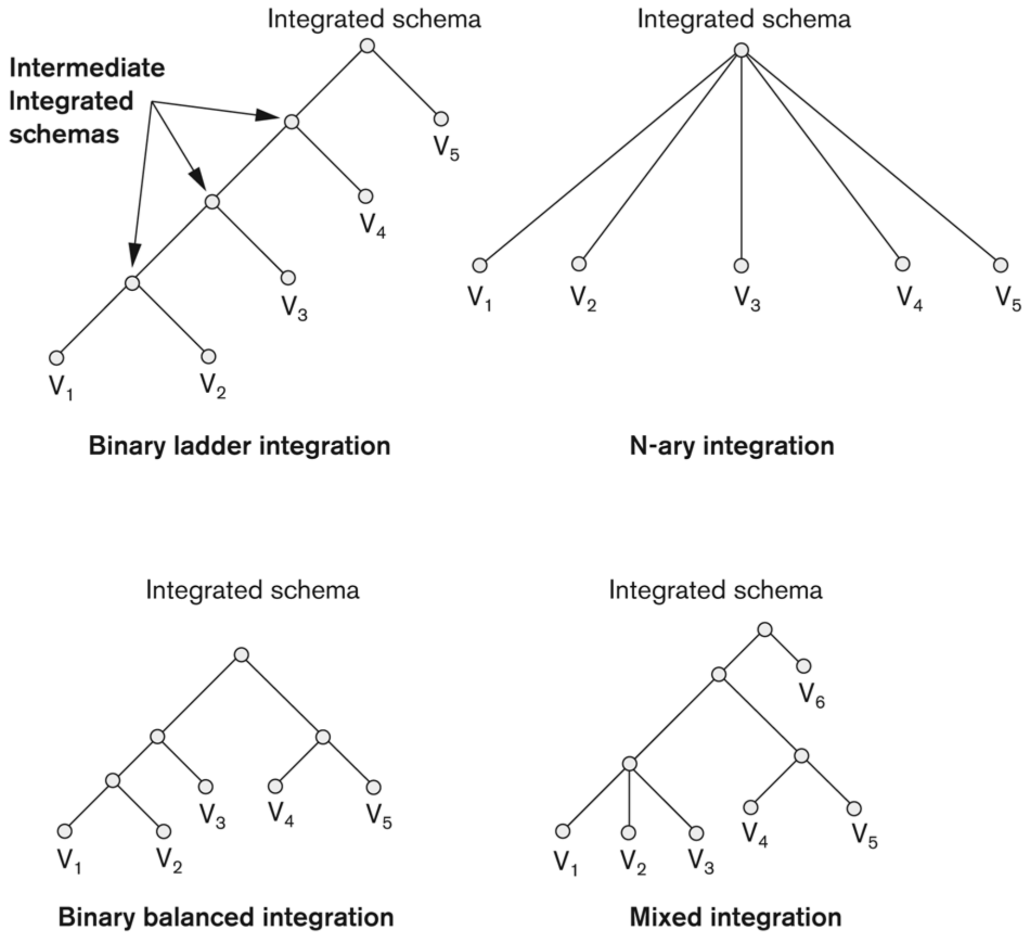


Figure 12.6
Different strategies for the view integration process.

Transaction Design

- Design characteristics of known database transactions in a DBMS
- Types of Transactions
  - Retrieval Transactions
    - Used to retrieve data
  - Update Transactions
    - Update data
  - Mixed Transactions
    - Combination of update and retrieval
- Techniques for Specifying Transactions
  - Input/output
  - Functional Behavior

Choice of DBMS

- Many factors to consider
  - Technical Factors
    - Type of DBMS: Relational, object-relational, object etc.
    - Storage Structures
    - Architectural options
  - Economic Factors
    - Acquisition, maintenance, training and operating costs
    - Database creation and conversion cost
  - Organizational Factors
    - Organizational philosophy
      - Relational or Object Oriented
      - Vendor Preference
    - Familiarity of staff with the system
    - Availability of vendor services

Logical Database Design

- Transform the Schema from high-level data model into the data model of the selected DBMS.
- Design of external schemas for specific applications
- Two stages
  1. System-independent mapping
     - DBMS independent mapping
  2. Tailoring the schemas to a specific DBMS
     - Adjusting the schemas obtained in step 1 to conform to the specific implementation features of the data model used in the selected DBMS
- Result
  - DDL statements in the language of the chosen DBMS

Physical Database Design

- Design the specifications for the stored database in terms of physical storage structures, record placements and indexes.
- Design Criteria
  - Response Time
    - Elapsed Time between submitting a database transaction for execution and receiving a response
  - Space Utilization
    - Storage space used by database files and their access path structures
  - Transaction throughput
    - Average number of transactions/minute
    - Must be measured under peak conditions
- Result
  - Initial determination of storage structures and access paths for database files

Database System Implementation and Tuning

- During this phase database and application programs are implemented, tested and deployed
- Database Tuning
  - System and Performance Monitoring
  - Data indexing
  - Reorganization
- Tuning is a continuous process

UML Diagrams

- Class Diagrams
  - Capture the static structure of the system
  - Represent classes, Interfaces, dependencies, generalizations and other relationships
- Object Diagrams
  - Show a set of objects and their relationships
- Component Diagrams
  - Show the organizations and dependencies among software components
- Deployment Diagrams
  - Represent the distribution of components across the hardware topology

UML Diagrams (contd.)

- Use Case Diagrams
  - Model the functional interactions between users and system
  - Describe scenarios of use
  - Serve as a communication tool between users and developers

UML Diagrams (contd.)

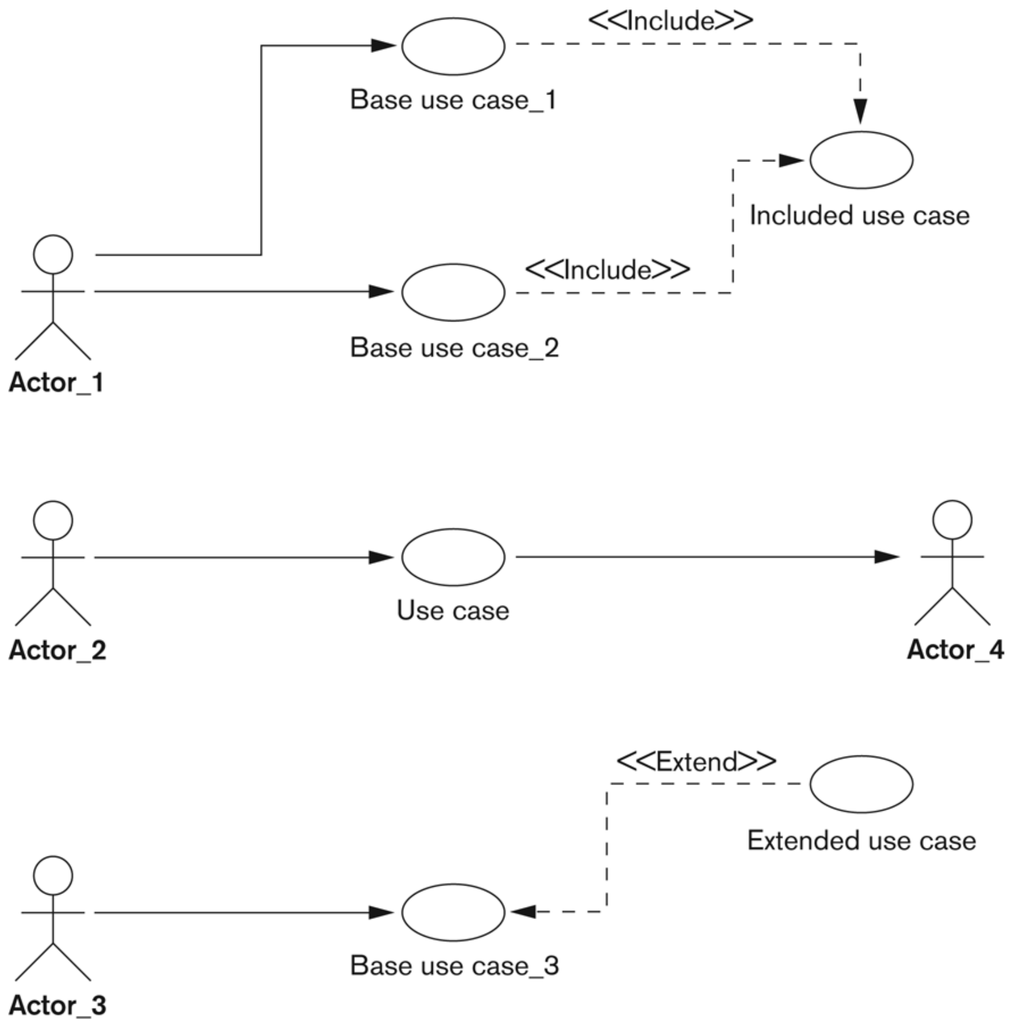


Figure 12.7
The use case diagram notation.

UML Diagrams (contd.)

- Sequence Diagrams
  - Represent interactions between various objects over time
  - Relate uses cases and class diagrams

UML Diagrams (contd.)

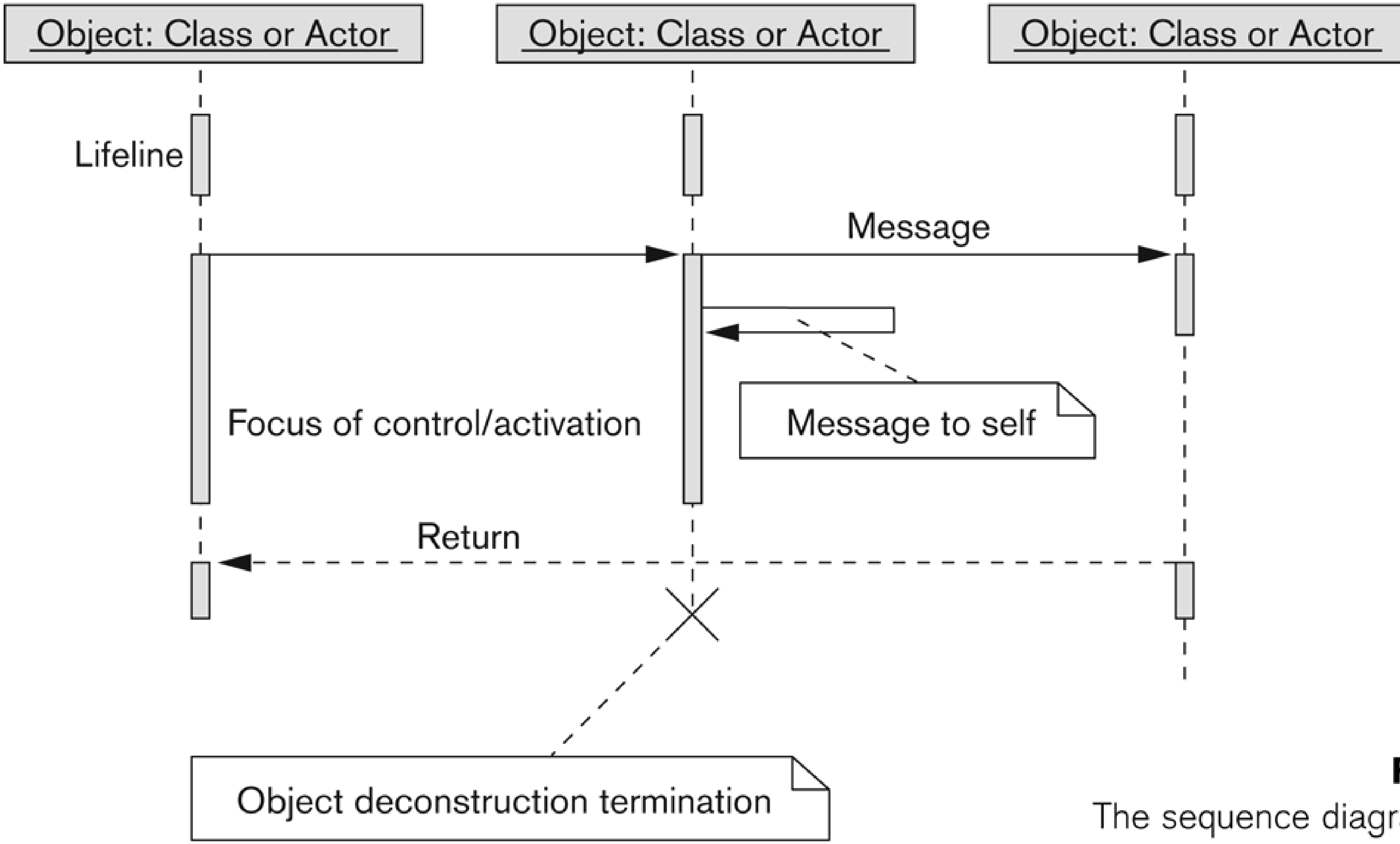


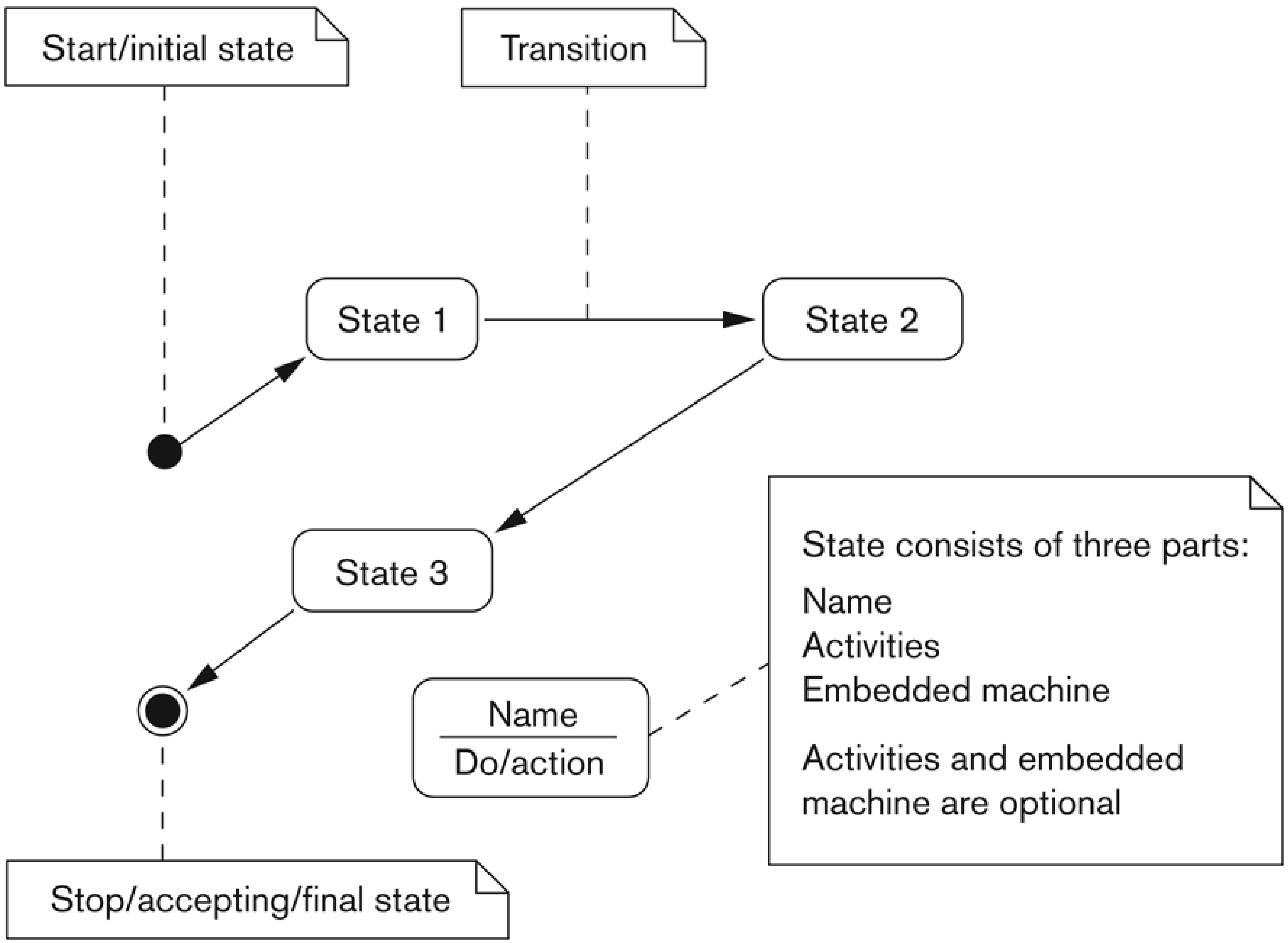
Figure 12.9
The sequence diagram notation.

UML Diagrams (contd.)

- Collaboration Diagrams
  - Represent interactions between objects as a series of sequenced messages
- Statechart Diagram
  - Describe how an object’s state changes in response to external events
  - Consist of states, transitions, actions, activities and events

UML Diagrams (contd.)

Figure 12.10
The statechart diagram notation.



UML Diagrams (contd.)

- Activity Diagrams
  - Model the flow of control from activity to activity
  - Flowcharts with states

Salient Features of Rational Rose Data Modeler

- UML based modeling tool for designing databases
- Reverse Engineering
  - Generate a conceptual data model from an existing DBMS database or DDL
- Forward Engineering
  - Create application/data model
  - Generate DDL from data model

Salient Features of Rational Rose Data Modeler (contd.)

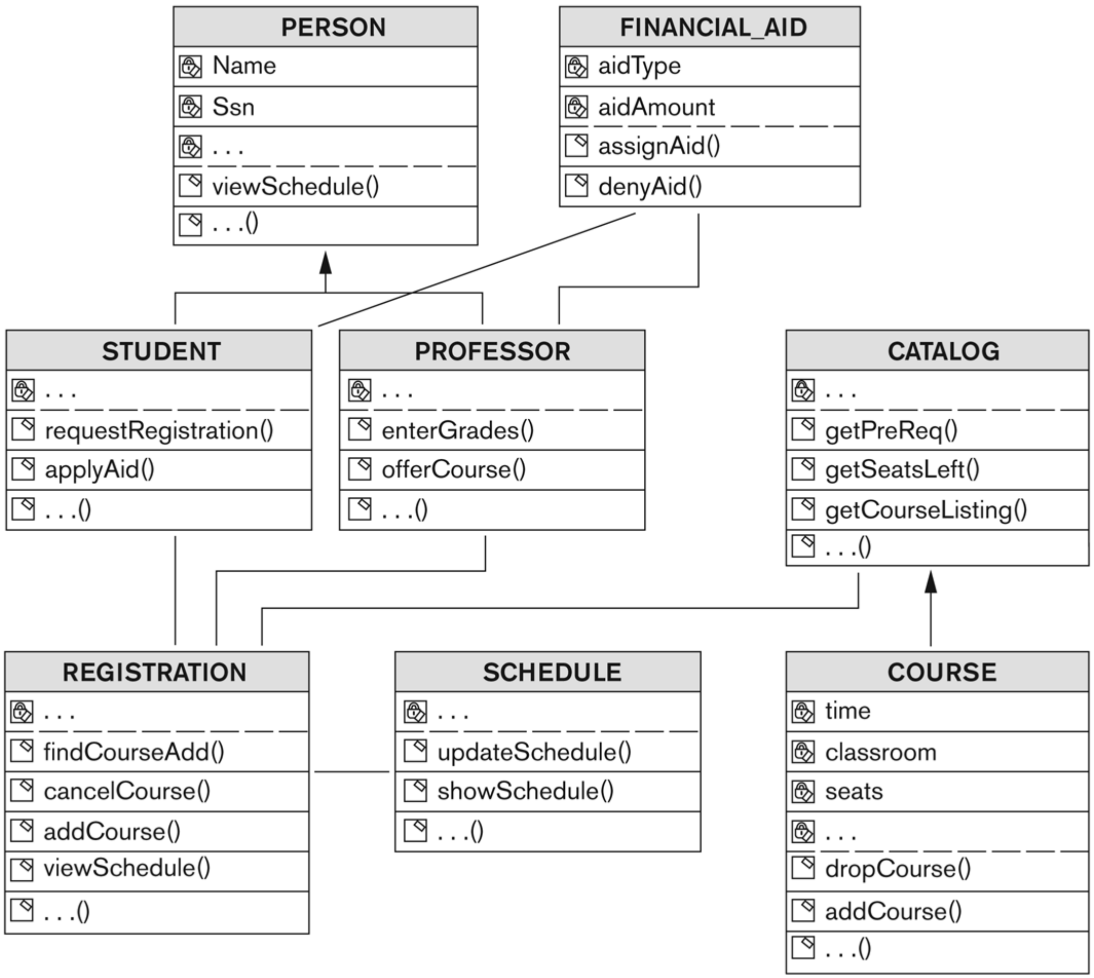


Figure 12.13
The design of the UNIVERSITY database as a class diagram.

Salient Features of Rational Rose Data Modeler (contd.)

- Modeling ER diagrams in UML
  - ER schema from the company example in chapter 3 can be drawn in Rational Rose using UML notation

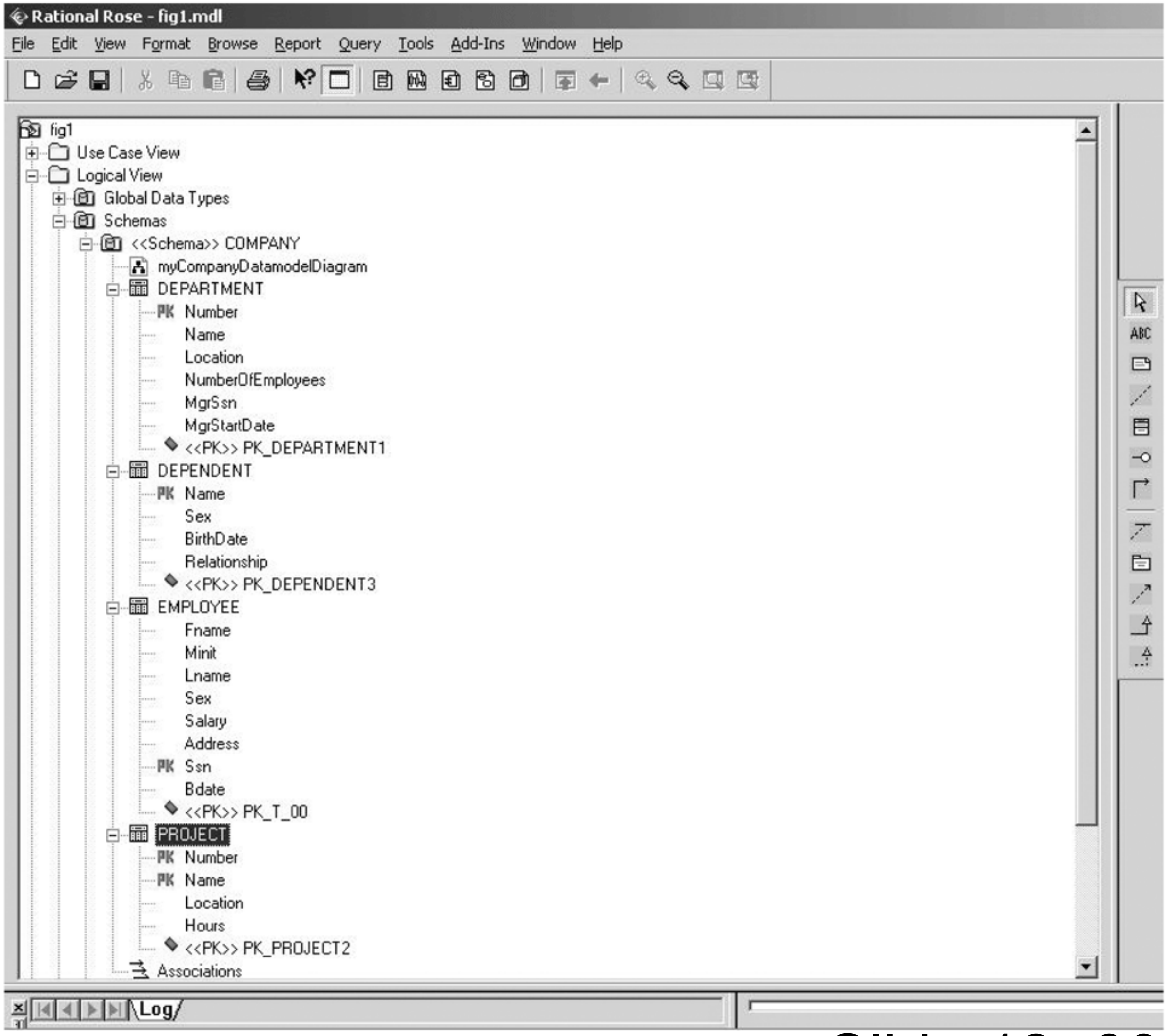


Figure 12.14
A logical data model diagram definition in Rational Rose.

Salient Features of Rational Rose Data Modeler (contd.)

- Keeps the data model and database synchronized
  - Gives the option of updating the model or changing the database
- Provides Extensive Domain Support
  - Allows database designers to create a standard set of user-defined types
- Allows Converting between logical and object model design
- Allows easy communication between various developing and design teams
  - Provides a common tool and platform to designers and developers
  - Rational Rose Web Publisher

Database Design Tools

- Common Features
  - Allow the designer to draw conceptual schema diagram in some tool-specific notation
  - Allow model mapping
  - Allow some level of design normalization
- Problems
  - Most tools do nothing more than representing relationships among tables
  - Most tools lack built-in methodology support
  - Most tools have poor design verification system

Characteristics of a Good Design Tool

- Easy-to-use interface
  - Easy to use
  - Customizable
- Analytical components
  - For difficult tasks
    - such as evaluating physical design alternatives or detecting conflicting constraints among views
- Heuristic components
  - Automating design process using heuristic rules

Characteristics of a Good Design Tool (contd.)

- Trade-off analysis
  - Comparative analysis in case of multiple alternatives
  - At least at the conceptual design level
- Display of design results
  - Displaying results in simple and easy to understand form
- Design Verification
  - Verifying that the resulting design satisfies the initial requirements