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 2
Data Models

Learning Objectives

- After completing this chapter, you will be able to:
  - Discuss data modeling and why data models are important
  - Describe the basic data-modeling building blocks
  - Define what business rules are and how they influence database design
  - Understand how the major data models evolved
  - List emerging alternative data models and the needs they fulfill
  - Explain how data models can be classified by their level of abstraction

Data Modeling and Data Models

- Data modeling: creating a specific data model for a determined problem domain
  - Data model: simple representation of complex real-world data structures
    - Useful for supporting a specific problem domain
  - Model: abstraction of a more complex real-world object or event
- Data Modeling: Involves the creation of a tailored data model designed to address the specific requirements of a defined problem domain.

- Data Model: Serves as a straightforward representation of complex/intricate real-world data structures, offering a simplified and organized framework essential for supporting and understanding the complexities within a designated problem domain.

The Importance of Data Models

- The importance of data modeling cannot be overstated
  - Facilitates communication
  - Gives various views of the database
  - Organizes data for various users
  - Provides an abstraction for the creation of good a database

The importance of data modeling cannot be overstated as it serves multiple critical functions. It facilitates effective communication among stakeholders, offers diverse perspectives of the database, organizes data to cater to different user needs, and provides a crucial abstraction layer for the creation of a well-structured database.

Data Model Basic Building Blocks

- **Entity:** person, place, thing, or event about which data will be collected and stored
  - Attribute: characteristic of an entity
  - Relationship: association among entities
    - One-to-many (1:M OR 1..*)
    - Many-to-many (M:N or *..*)
    - One-to-one (1:1 OR 1..1)
  - Constraint: restriction placed on data
    - Ensures data integrity

    Constraint in Data Modeling:

    A constraint is a limitation or rule imposed on data to ensure its integrity and adherence to specific criteria within a database.

    - Example: In a university database, there might be a constraint that ensures each student (entity) has a unique student ID (attribute) to maintain data integrity. This constraint prevents the occurrence of duplicate student IDs, ensuring a one-to-one relationship between each student and their assigned identification number.

Business Rules

- Brief, precise, and unambiguous description of a policy, procedure, or principle
  - Create and enforce actions within that organization's environment
  - Establish entities, relationships, and constraints

Concise and unambiguous descriptions of policies, procedures, or principles within an organization. Business rules play a crucial role in creating and enforcing actions in the organizational environment, shaping the definition of entities, relationships, and constraints within a database system.

- Example Business Rule:

Policy: Each customer (entity) must have a unique email address (attribute).

Enforcement: In the organizational environment, a business rule is established to ensure that no two customers share the same email address, maintaining data integrity and supporting a one-to-one relationship between customers and their email addresses.

Discovering Business Rules (1 of 2)

- Sources of business rules
  - Company managers
  - Policy makers
  - Department managers
  - Written documentation
  - Direct interviews with end users

Business rules can originate from various sources within an organization, including insights from company managers, policies defined by decision-makers, guidelines set by department managers, written documentation detailing procedures, and direct interviews with end users. These diverse sources contribute to the formulation and implementation of effective business rules in a comprehensive manner.

Discovering Business Rules (2 of 2)

Reasons for Identifying and Documenting Business Rules: Identifying and documenting business rules is essential to:

1. **Standardize Company's View of Data:** Establish a uniform understanding of data across the organization.
2. **Facilitate Communication:** Serve as a communication tool between users and designers, ensuring clarity and alignment in data-related discussions.
3. **Assist Designers:** Provide guidelines and insights for designers in developing effective data structures and processes.
4. **Understand Nature, Role, and Scope:** Gain a comprehensive understanding of the nature, role, and scope of data and business processes within the organization.
5. **Develop Relationship Rules:** Formulate relationship participation rules and constraints for accurate data representation.
6. **Create an Accurate Data Model:** Contribute to the creation of an accurate and efficient data model that aligns with the organization's business requirements

Translating Business Rules into Data Model Components

- **Role of Business Rules in Database Design:** Business rules play a foundational role in guiding the identification of entities, attributes, relationships, and constraints in a database.
- **Entity Identification:** Nouns in business rules often translate into entities, helping define key elements in the database structure.
- **Relationship Translation:** Verbs in business rules often translate into relationships among entities, providing insights into the connections within the data model.
- **Bidirectional Relationships:** Recognizing that relationships are bidirectional ensures a comprehensive understanding of the associations between entities.
- **Questions for Relationship Identification:**

1. "How many instances of B are related to one instance of A?"
2. "How many instances of A are related to one instance of B?"

- These questions help elucidate the nature and cardinality of relationships, contributing to a well-defined and accurate database design.

Naming Conventions

Entity Name Requirements:

- **Descriptive of Business Objects:** Entity names should vividly describe the objects present in the business environment.
- **User-Friendly Terminology:** Utilize terminology familiar to the users for ease of understanding and relevance.

Attribute Name Requirements:

- **Descriptive of Data:** Attribute names must clearly describe the data represented by the attribute.
- **Proper Naming:** Properly named attributes facilitate effective communication between different parties involved in the database design and usage.
- **Promotes Self-Documentation:** Well-chosen attribute names contribute to the self-documentation of the database, making it more understandable and maintainable

Hierarchical and Network Models (1 of 3)

Hierarchical Models: Developed specifically for handling substantial volumes of data within intricate manufacturing projects.

- **Representation:** Typically depicted as an upside-down tree structure comprising segments.
- **Segment Equivalence:** Segments within the hierarchical model serve a role similar to record types in a file system, organizing and categorizing data.
- **Relationship Structure:** The model illustrates a set of one-to-many (1:M) relationships, outlining the hierarchical connections and dependencies within the data.

Hierarchical and Network Models (2 of 3)

- **Network Models:** Developed to efficiently represent intricate data relationships, leading to enhanced database performance and the establishment of a standardized approach.
- **Key Features:**
- **Handling Complex Relationships:** Particularly effective in managing complex data relationships within the database.
- **Performance Improvement:** Contributed to improved database performance and set a standard for organizing and accessing data.

Hierarchical and Network Models (3 of 3)

Innovations and Concepts:

- **Multiple Parent Records:** Allows a record to have more than one parent, providing flexibility in representing complex associations.
- **Enduring Database Concepts:** Standard database concepts introduced with the network model remain integral to modern data models.

Noteworthy Terminology:

- **Schema and Subschema:** Introduced the concepts of schema (overall structure) and subschema (a subset tailored for specific users or applications).
- **Data Manipulation Language (DML):** Defined the language for manipulating and querying data within the database.
- **Data Definition Language (DDL):** Established the language for defining the structure and organization of the database.

The Relational Model (1 of 4)

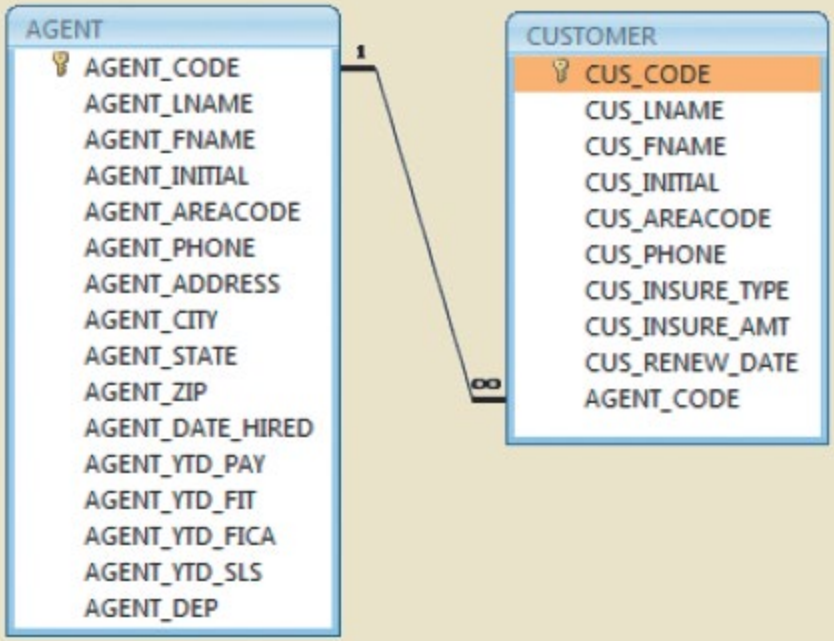
- Produced an automatic transmission database that replaced standard transmission databases
  - Based on a relation (i.e., table): matrix composed of intersecting tuples (rows) and attributes (columns)
- Describes a precise set of data manipulation constructs
- A database model that organizes data into tables (relations) where each row represents a record and each column represents an attribute. The relationships between tables are established based on common attributes, fostering a structured and flexible approach to data organization.

The Relational Model (2 of 4)

- RDBMS stands for Relational Database Management System. It is a type of database management system that is based on the relational model. In an RDBMS, data is organized into tables (relations), and the relationships between these tables are defined by common attributes. RDBMS ensures data integrity, supports data manipulation through SQL (Structured Query Language), and provides a structured framework for efficient data management. Examples include MySQL, Oracle, and Microsoft SQL Server.
- Relational database management system (RDBMS)
  - Performs basic functions provided by the hierarchical and network DBMS systems
  - Makes the relational data model easier to understand and implement
  - Hides the complexities of the relational model from the user

The Relational Model (3 of 4)

FIGURE 2.2 A RELATIONAL DIAGRAM



A visual representation of the structure and relationships within a relational database. It typically illustrates tables (relations) as boxes, with attributes represented as columns within these boxes. Lines connecting the boxes indicate the relationships between tables, often denoting the common attributes used to establish connections. Relational diagrams provide a clear and concise overview of the database schema and its interconnected entities.

The Relational Model (4 of 4)

SQL-Based Relational Database Application: An application that utilizes Structured Query Language (SQL) to interact with a relational database. It provides an end-user interface, enabling users to interact with the data seamlessly.

Key Features:

- **Table Structure:** Comprises a collection of tables stored in the database.
- **Table Independence:** Each table functions independently, holding specific sets of data.
- **Relationships:** Relationships between rows in different tables are established based on common values in shared attributes.

SQL Engine:

- **Query Execution:** The SQL engine executes all queries submitted through the application, retrieving, updating, or manipulating data in accordance with user requests.

The Entity Relationship Model (1 of 2)

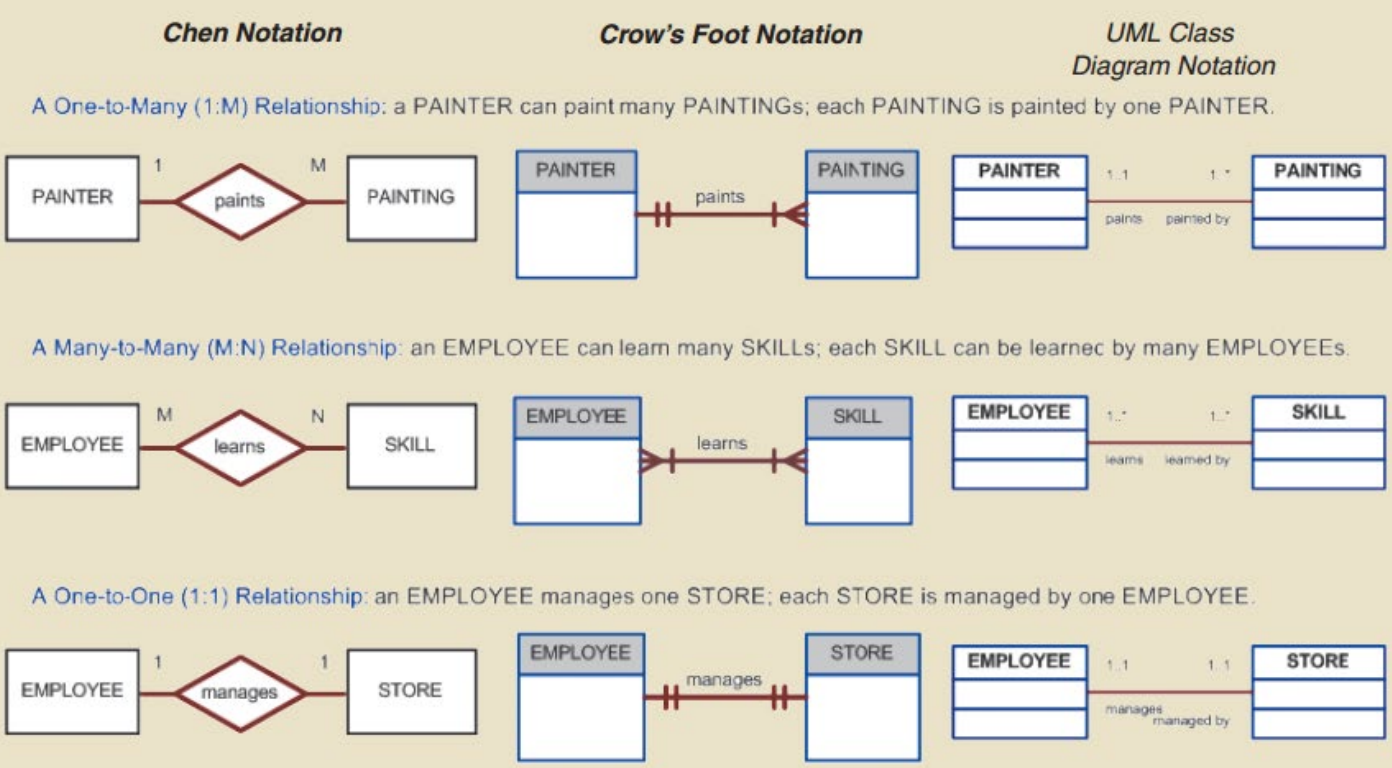
Entity Relationship Diagram (ERD): A visual representation that graphically illustrates entities and their relationships within a database structure. ERD uses graphical symbols and lines to model the components of a database.

Key Components:

- **Entity Instance or Entity Occurrence:** Represented by rows in a relational table, signifying specific occurrences of entities.
- **Attributes:** Describe particular characteristics or properties of entities.
- **Connectivity:** Term used to label and describe the types of relationships between entities, indicating how they are connected and interact within the database structure.

The Entity Relationship Model (2 of 2)

FIGURE 2.3 THE ER MODEL NOTATIONS



The Object-Oriented Data Model (1 of 3)

Object-Oriented Database Management System (OODBMS): An OODBMS is a type of database management system based on the principles of Object-Oriented Data Modeling (OODM). In this system, both data and their relationships are encapsulated within a single structure known as an object.

Key Concepts:

- **Object:** Represents a unit that contains both data and their relationships, along with operations that can be performed on it. Objects are the basic building blocks for autonomous structures in an OODBMS.
- **Abstraction:** Objects in an OODBMS serve as an abstraction of real-world entities, capturing their properties and behaviors.
- **Attribute:** Describes the properties of an object, providing details about the characteristics of the encapsulated data.

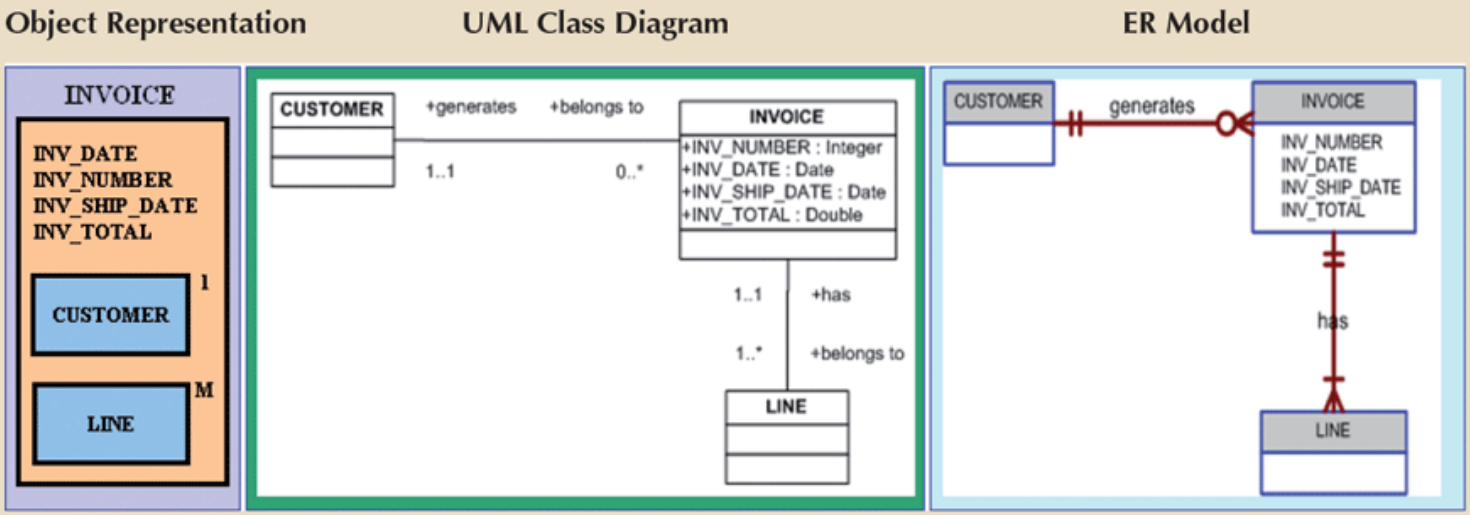
Therefore, Object-oriented databases offer a more intuitive and flexible way to model and manage complex data structures by aligning with the principles of Object-Oriented Programming (OOP).

The Object-Oriented Data Model (2 of 3)

- **Class:** A class is a grouping or collection of similar objects that share a common structure and behavior. Classes are organized in a class hierarchy, forming the basis for object-oriented programming.
- **Class Hierarchy:** Resembles an upside-down tree structure where each class has only one parent, depicting the relationships and inheritances between classes.
- **Inheritance:** Inheritance is a fundamental concept in object-oriented programming where an object inherits methods and attributes from the classes above it in the hierarchy. This promotes code reuse and establishes a clear structure.
- **Unified Modeling Language (UML):** UML is a standardized modeling language that utilizes sets of diagrams and symbols to graphically represent and model a system, including its classes, relationships, and interactions. UML provides a visual and standardized way to document and communicate the design of object-oriented systems.
- *In an object-oriented system for automobiles, the "Vehicle" class could be the parent, with subclasses like "Car" and "Truck." A specific car, say a "Toyota Camry," inherits attributes from both "Vehicle" and "Car," such as make, model, and the number of doors. Unified Modeling Language (UML) diagrams help visually represent this hierarchy, providing a standardized and concise way to understand the relationships within the automotive system.*

The Object-Oriented Data Model (3 of 3)

FIGURE 2.4 A COMPARISON OF THE OO, UML, AND ER MODELS



Object/Relational and XML

Extended Relational Data Model (ERDM): ERDM extends the traditional relational data model to support Object-Oriented (OO) features, including extensible data types based on classes and inheritance. This evolution enriches the relational model to handle more complex data structures.

Object/Relational Database Management System (O/R DBMS): An O/R DBMS is built on the principles of ERDM, integrating both object-oriented and relational database capabilities. It allows for the representation of complex data relationships and structures, combining the strengths of both models.

Extensible Markup Language (XML): XML is a versatile markup language used for managing unstructured data. It enables the efficient and effective exchange of structured, semistructured, and unstructured data across different systems and platforms, providing a standardized format for data representation

Emerging Data Models: Big Data and NoSQL (1 of 3)

Goals of Big Data: The goals of Big Data revolve around finding new and improved methods to handle vast volumes of data generated from sources like the web and sensors. It aims to provide high performance at a reasonable cost, addressing the challenges posed by the scale and complexity of contemporary data sources.

Characteristics of Big Data: Big Data is characterized by the three Vs:

- **Volume:** Refers to the sheer size of the data, often ranging from terabytes to petabytes.
- **Velocity:** Describes the speed at which data is generated, processed, and analyzed in real-time or near real-time.
- **Variety:** Encompasses the diverse types of data, including structured, unstructured, and semi-structured data, requiring flexible and adaptable processing methods.

Emerging Data Models: Big Data and NoSQL (2 of 3)

Challenges of Big Data: The challenges of Big Data include the overwhelming volume, rendering conventional structures inadequate. The sheer scale makes storage and processing expensive. Traditional OLAP tools struggle with the inconsistency of dealing with unstructured data.

New Technologies of Big Data: To address these challenges, new technologies have emerged:

- **Hadoop:** An open-source framework designed for distributed storage and processing of large datasets.
- **Hadoop Distributed File System (HDFS):** The storage component of Hadoop, optimized for handling vast amounts of data across distributed clusters.
- **MapReduce:** A programming model for processing and generating large-scale datasets in parallel.
- **NoSQL:** A category of database systems that provide flexible and scalable solutions for handling unstructured and semi-structured data.

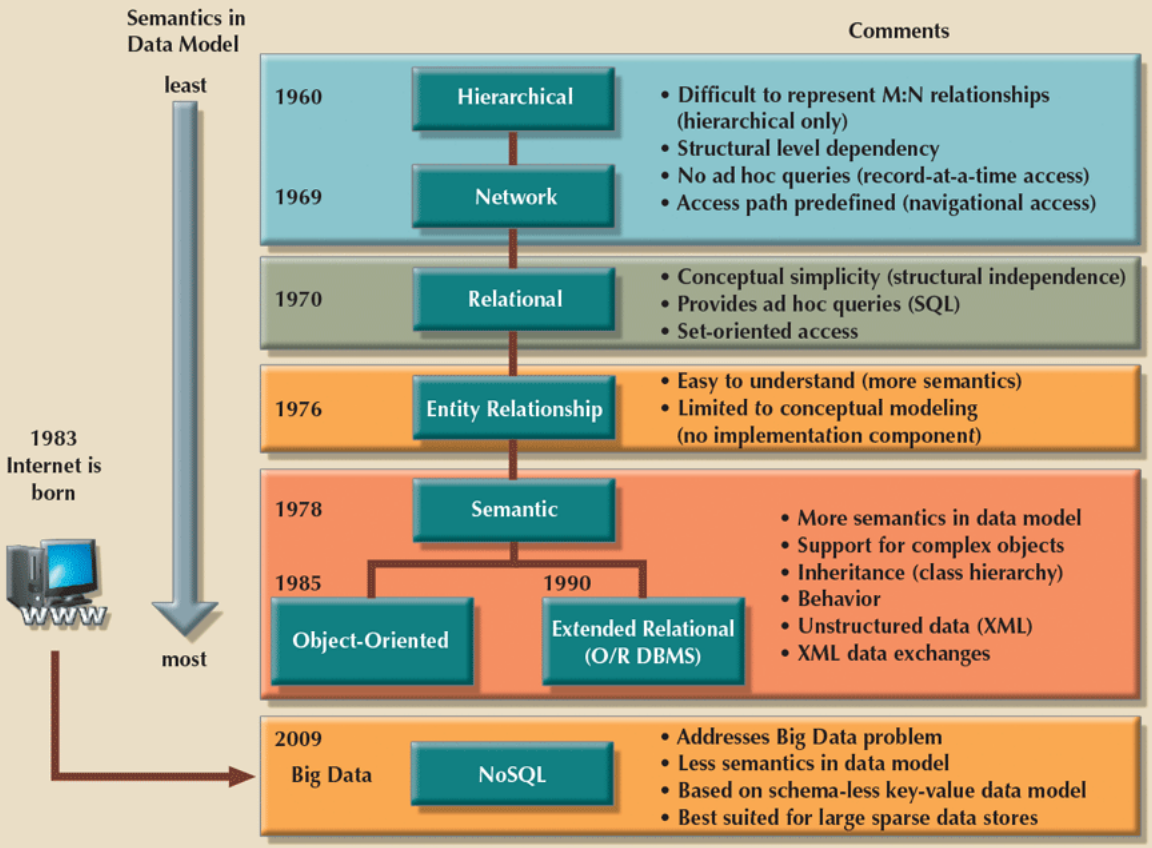
Emerging Data Models: Big Data and NoSQL (3 of 3)

- NoSQL databases
  - Not based on the relational model
  - Support distributed database architectures
  - Provide high scalability, high availability, and fault tolerance
  - Support large amounts of sparse data
  - Geared toward performance rather than transaction consistency
  - Provides a broad umbrella for data storage and manipulation

- NoSQL databases are not based on the traditional relational model. Instead, they support distributed database architectures, offering high scalability, availability, and fault tolerance. These databases excel in handling large amounts of sparse data and prioritize performance over transaction consistency. NoSQL provides a broad umbrella for diverse approaches to data storage and manipulation, catering to the specific needs of various applications and use cases.

Data Models: A Summary

FIGURE 2.5 THE EVOLUTION OF DATA MODELS



Hierarchical Model

- Advantages
  - Promotes data sharing
  - Parent/child relationship promotes conceptual simplicity and data integrity
  - Database security is provided and enforced by DBMS
  - Efficient with 1:M relationships
- Disadvantages
  - Requires knowledge of physical data storage characteristics
  - Navigational system requires knowledge of hierarchical path
  - Changes in structure require changes in all application programs
  - Implementation limitations
  - No data definition
  - Lack of standards

Network Model

- Advantages
  - Conceptual simplicity
  - Handles more relationship types
  - Data access is flexible
  - Data owner/member relationship promotes data integrity
  - Conformance to standards
  - Includes data definition language (DDL) and data manipulation language (DML)
- Disadvantages
  - System complexity limits efficiency
  - Navigational system yields complex implementation, application development, and management
  - Structural changes require changes in all application programs

Relational Model

- Advantages
  - Structural independence is promoted using independent tables
  - Tabular view improves conceptual simplicity
  - Ad hoc query capability is based on SQL
  - Isolates the end user from physical-level details
  - Improves implementation and management simplicity
- Disadvantages
  - Requires substantial hardware and system software overhead
  - Conceptual simplicity gives untrained people the tools to use a good system poorly
  - May promote information problems

Entity Relationship Model

- Advantages
  - Visual modeling yields conceptual simplicity
  - Visual representation makes it an effective communication tool
  - Is integrated with the dominant relational model
- Disadvantages
  - Limited constraint representation
  - Limited relationship representation
  - No data manipulation language
  - Loss of information content occurs when attributes are removed from entities to avoid crowded displays

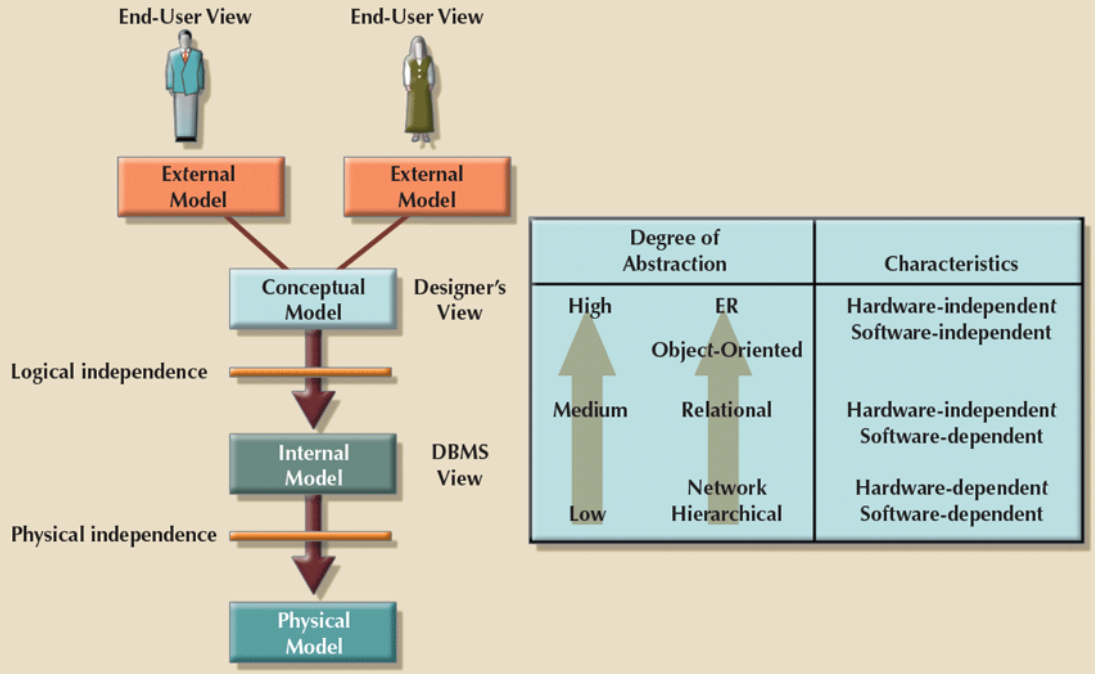
Object-Oriented Model

- Advantages
  - Semantic content is added
  - Visual representation includes semantic content
  - Inheritance promotes data integrity
- Disadvantages
  - Slow development of standards caused vendors to supply their own enhancements
  - Complex navigational system
  - Learning curve is steep
  - High system overhead slows transactions

NoSQL

- Advantages
  - High scalability, availability, and fault tolerance are provided
  - Uses low-cost commodity hardware
  - Supports Big Data
  - Key-value model improves storage efficiency
- Disadvantages
  - Complex programming is required
  - There is no relationship support
  - There is no transaction integrity support
  - In terms of data consistency, it provides an eventually consistent model

FIGURE 2.6 DATA ABSTRACTION LEVELS



1.Physical Level (Internal Schema): Deals with the actual storage and organization of data, detailing data structures, storage methods, and access mechanisms.
2.Logical Level (Conceptual Schema): Represents how users and applications perceive data, abstracting away physical storage details. Focuses on defining relationships and providing a conceptual framework.
3.View Level (External Schema): The highest level involves creating specific views of data for user groups or applications, enabling interaction with a subset of data without considering the entire database structure.

The External Model (1 of 2)

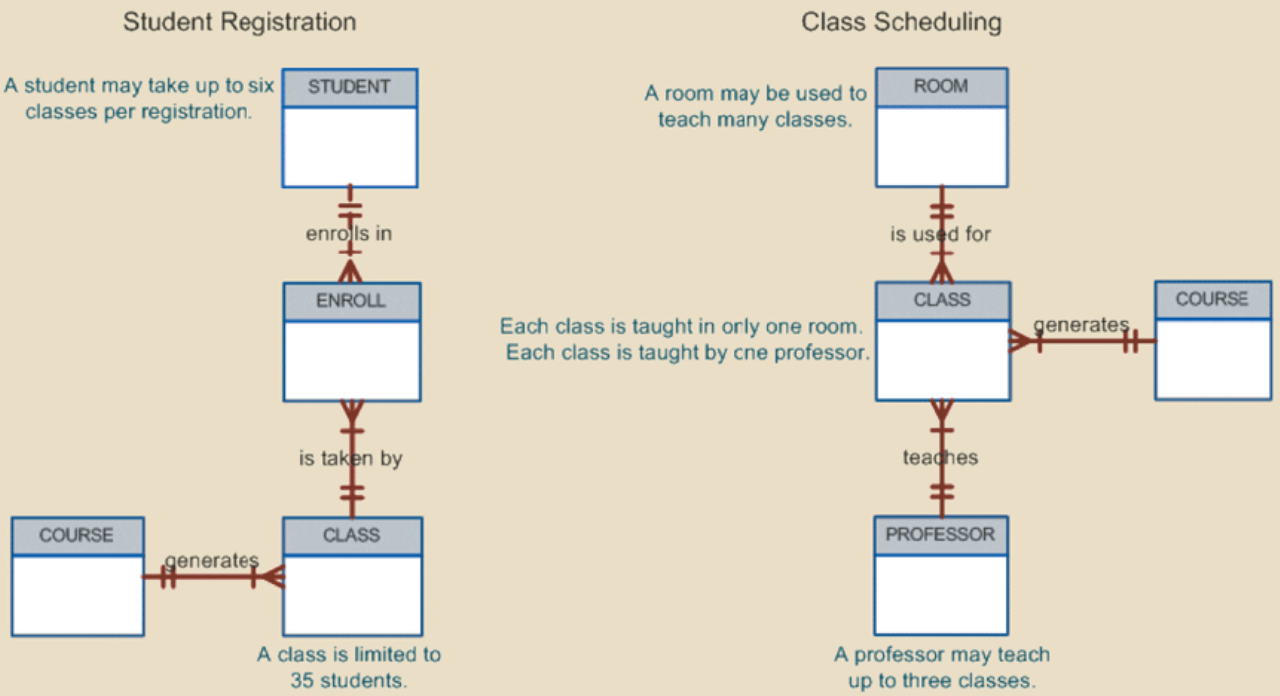
External Schema in Database: The external schema represents the end users' view of the data environment, focusing on those who utilize application programs to interact with and manipulate data, ultimately generating information. External views are often represented using Entity-Relationship (ER) diagrams.

Key Points:

1. **End Users' Perspective:** Tailored for the individuals using application programs to work with data and derive information from the database.
2. **ER Diagrams:** Entity-Relationship diagrams serve as visual tools to depict the external views, showcasing the relationships between different entities from the end users' standpoint.
3. **External Schema Definition:** It is a specific representation of an external view, providing a detailed blueprint of how data is organized and accessed for a particular set of users or applications.

The External Model (2 of 2)

FIGURE 2.7 EXTERNAL MODELS FOR TINY COLLEGE



The Conceptual Model (1 of 2)

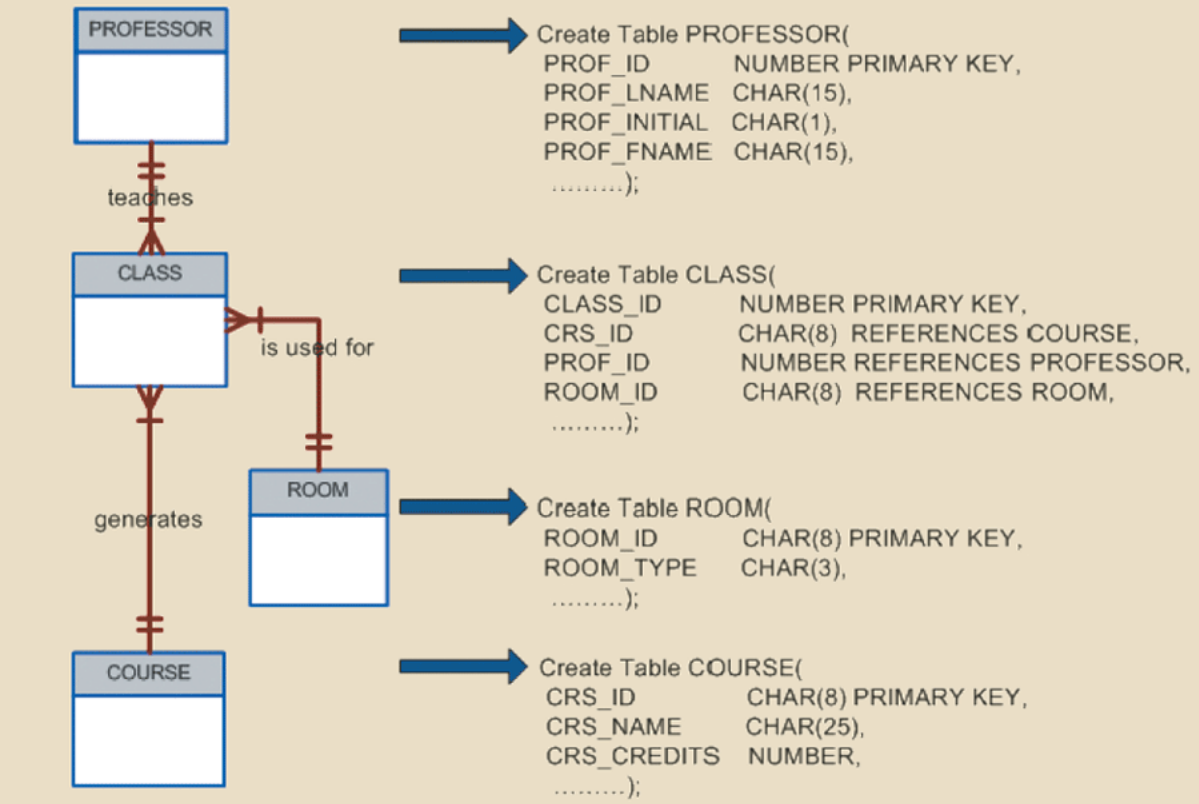
Conceptual Schema in Database: The conceptual schema represents a comprehensive view of the entire database, serving as the foundation for identifying and providing high-level descriptions of the main data objects. The process of creating a conceptual data model, known as logical design, forms the basis for the conceptual schema.

Advantages of Conceptual Model:

1. **Macro-level View:** Offers a macro-level view of the data environment, allowing organizations to understand the overarching structure and relationships within the database.
2. **Software and Hardware Independence:** The conceptual model is independent of specific software or hardware, providing flexibility and adaptability to changes in technology without impacting the core conceptual schema.

The Conceptual Model (2 of 2)

FIGURE 2.9 INTERNAL MODEL FOR TINY COLLEGE



The Internal Model

Internal Schema in Database: The internal schema represents the database as perceived by the Database Management System (DBMS), mapping the conceptual model to the DBMS's specific constructs. It provides a detailed and specific representation of the internal model, utilizing the database constructs supported by the chosen database system.

Key Points:

1. **Mapping to DBMS:** Involves translating the conceptual model into a representation that aligns with the capabilities and structures supported by the chosen Database Management System.
2. **Logical Independence:** Allows for changes to the internal model without affecting the conceptual model, ensuring that modifications at the storage and retrieval level don't impact the overarching understanding of the data.
3. **Hardware Independence:** The internal schema is designed to be independent of the underlying hardware, meaning it remains unaffected by the type of computer on which the database software is installed, providing flexibility in deployment.

The Physical Model (1 of 2)

Physical Schema in Database: The physical schema operates at the lowest level of abstraction, detailing how data is stored on various storage media like magnetic, solid-state, or optical media. It involves defining physical storage structures and data access methods, making it software and hardware dependent.

Key Characteristics:

1.Lowest Abstraction Level: Operates at the lowest level of abstraction, dealing with the nitty-gritty details of data storage on physical media.

2.Software and Hardware Dependency: Being at the physical level, it is closely tied to the specific software and hardware configurations of the system.

3.Relational Model and Physical Independence: In contrast, the relational model is aimed at the logical level, abstracting away physical details. Physical independence ensures that changes in the physical model, such as storage modifications, do not affect the internal model, providing a layer of insulation.

The Physical Model (2 of 2)

Table 2.4 Levels of Data Abstraction			
Model	Degree of Abstraction	Focus	Independent of
External	High	End-user views	Hardware and software
Conceptual	Medium-High	Global view of data (database model independent)	Hardware and software
Internal	Medium-Low	Specific database model	Hardware
Physical	Low	Storage and access methods	Neither hardware nor software

Summary

- A data model is an abstraction of a complex real-world data environment
- There are many types of data models (e.g., hierarchical, network, relational, object-oriented, extended relational data model, etc.)
- Data-modeling requirements are a function of different data views (global versus local) and the level of data abstraction