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 4 Entity Relationship (ER) Modelling

Learning Objectives

- After completing this chapter, you will be able to:
  - Identify the main characteristics of entity relationship components
  - Describe how relationships between entities are defined, refined, and incorporated into the database design process
  - See how ERD components affect database design and implementation
  - Understand that real-world database design often requires the reconciliation of conflicting goals

The Entity Relationship Model (ERM)

- Forms the basis of an entity relationship diagram (ERD)
  - Conceptual database as viewed by end user, ERD serves as a visual representation of a conceptual database, highlighting its core components:
- Database’s main components
  - **Entities:** Entities represent real-world objects or concepts within the database. For example, in a university database, "Student" and "Course" could be entities.
  - **Attributes:** Attributes describe the properties or characteristics of entities. For instance, in the "Student" entity, attributes could include "StudentID," "Name," and "Major."
  - **Relationships:** Relationships illustrate how entities are connected or related to each other. In the university database, a relationship could exist between "Student" and "Course," indicating that a student enrolls in courses.

Entities

- **Entity:** An object of interest to the end user, representing a group of similar items or concepts within the database. For example, in a library database, "Book" could be an entity.
- **Entity Set:** Refers to the collection of all instances of a particular entity within the database. For instance, the entity set "Book" encompasses all books in the library database.
- **Entity Relationship Model (ERM):** A graphical representation that illustrates how entities are related to each other within a database system. It corresponds to a table in the relational environment.
- **Entity Instance (or Entity Occurrence):** A specific occurrence of an entity, representing a single row in a database table. In a library database, each individual book record would be an entity instance.
- **Entity Representation:** In various diagramming notations like Chen, Crow's Foot, and UML, entities are typically represented by rectangles containing the entity's name, usually written in all capital letters.

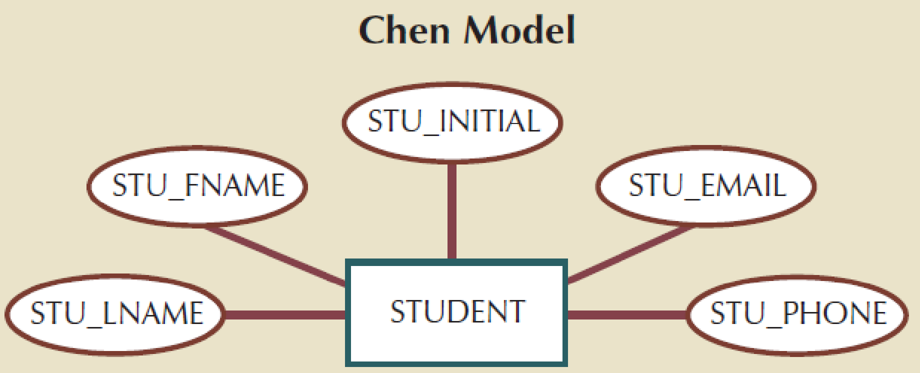
Attributes (1 of 7)

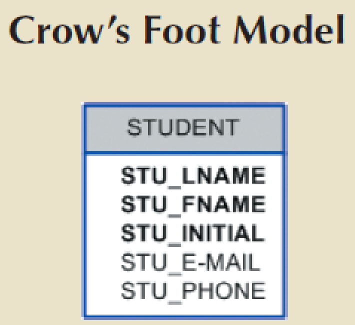
- **Characteristics of Entities:**
- **Required Attribute:** Attributes that must have a value and cannot be left empty. For example, in a "Customer" entity, "CustomerID" might be a required attribute.
- **Optional Attribute:** Attributes that do not require a value and can be left empty. An example could be the "Middle Name" attribute in a "Person" entity.
- **Domain:** The set of possible values for a given attribute. For instance, the domain of the "Gender" attribute could be {Male, Female, Other}.
- **Identifier:** One or more attributes that uniquely identify each entity instance. For example, in an "Employee" entity, the "EmployeeID" might serve as the identifier.
- **Composite Identifier:** A primary key composed of more than one attribute. For instance, in a "Sales" entity, the composite identifier might consist of both "OrderID" and "ProductID".
- **Composite Attribute:** An attribute that can be subdivided to yield additional attributes. An example could be the "Address" attribute, which may comprise "Street", "City", "State", and "Zip Code".
- **Simple Attribute:** An attribute that cannot be subdivided further. For instance, "Age" could be considered a simple attribute.
- **Single-valued Attribute:** An attribute that has only a single value. For example, "Date of Birth" in a "Person" entity.
- **Multivalued Attributes:** Attributes that can have multiple values. For instance, a "Skills" attribute in an "Employee" entity could include multiple skills such as "Programming", "Management", etc.

CENGAGE

Attributes (2 of 7)

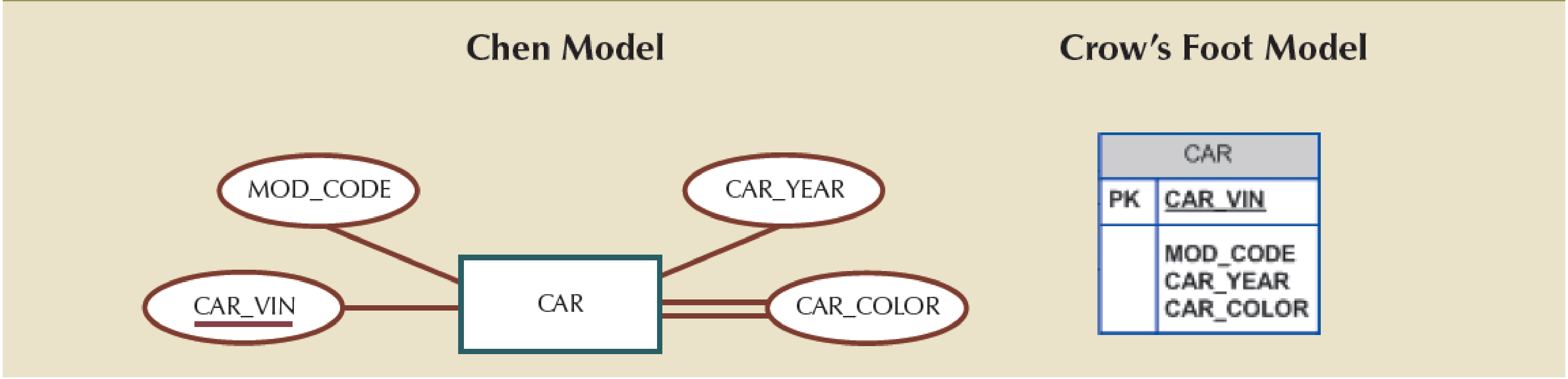
FIGURE 4.1 THE ATTRIBUTES OF THE STUDENT ENTITY: CHEN AND CROW' FOOT





Attributes (3 of 7)

FIGURE 4.3 A MULTIVALUED ATTRIBUTE IN AN ENTITY

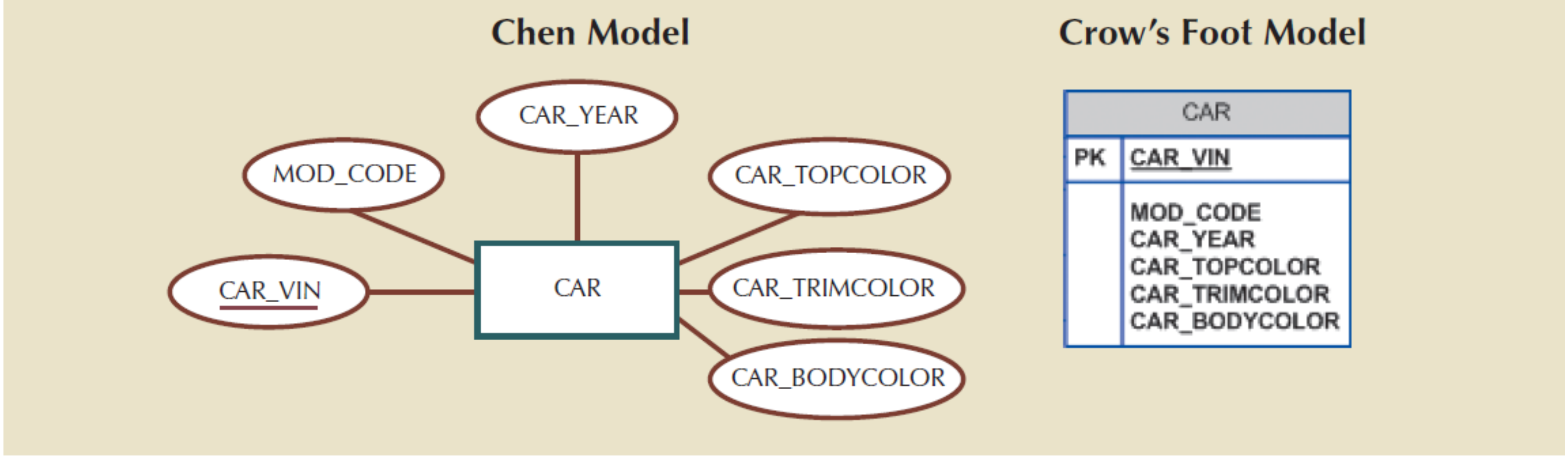


Attributes (4 of 7)

- Requirements of multivalued attributes
  - **Create Several New Attributes:** One approach to handling multivalued attributes is to create several new attributes, one for each component of the original multivalued attribute. For example, if the original multivalued attribute is "Skills", you might create separate attributes such as "Programming Skill", "Management Skill", etc.
  - **Develop a New Entity:** Another approach is to develop a new entity composed of the original multivalued attribute’s components. Using the same example of "Skills", you could create a new entity called "EmployeeSkills" with attributes like "EmployeeID" and "Skill".
  - **Derived Attribute:** A derived attribute is an attribute whose value is calculated from other attributes. These attributes are derived using an algorithm or formula based on other attribute values. For instance, if you have attributes for "Height" and "Weight", you could derive a "Body Mass Index (BMI)" attribute using a formula.

Attributes (5 of 7)

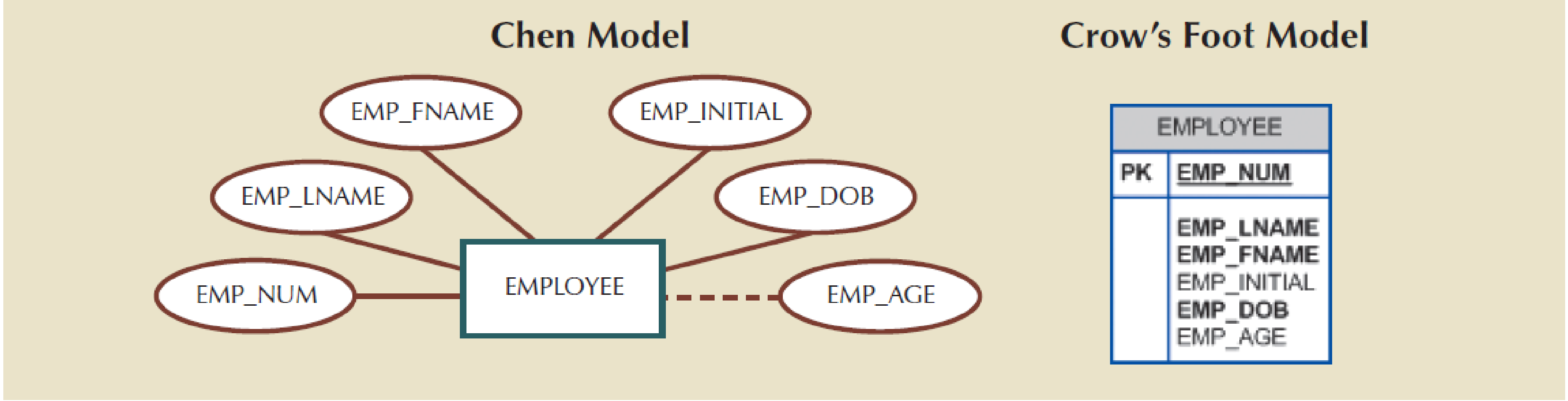
FIGURE 4.4 SPLITTING THE MULTIVALUED ATTRIBUTE INTO NEW ATTRIBUTES



Splitting the multivalued attribute into new attributes involves creating separate attributes for each component of the original multivalued attribute. For example, if the original multivalued attribute is "Skills", it can be split into individual attributes such as "Programming Skill", "Management Skill", "Communication Skill", and so on. Each of these new attributes represents a specific aspect of the original multivalued attribute, allowing for more granular data representation and easier querying and analysis.

Attributes (6 of 7)

FIGURE 4.6 DEPICTION OF A DERIVED ATTRIBUTE



A derived attribute is an attribute whose value is calculated or derived from other attributes within the database. It does not store data directly but rather computes its value based on predefined rules or algorithms applied to other attributes. For example, in a database tracking employee information, a derived attribute could be "Age", which is calculated based on the employee's date of birth and the current date. This ensures that the age is always up-to-date and consistent without needing to be stored separately in the database.

Attributes (7 of 7)

Table 4.2	Advantages and Disadvantages of Storing Derived Attributes	
	Derived Attribute: Stored	**Derived Attribute: Not Stored**
Advantage	Saves CPU processing cycles Saves data access time Data value is readily available Can be used to keep track of historical data	Saves storage space Computation always yields current value
Disadvantage	Requires constant maintenance to ensure derived value is current, especially if any values used in the calculation change	Uses CPU processing cycles Increases data access time Adds coding complexity to queries

CENGAGE

Relationships, Connectivity, and Cardinality

- Association between entities that always operate in both directions
  - **Participants:** entities that participate in a relationship
- **Connectivity:** describes the relationship classification
  - Include 1:1, 1:M, and M:N
- **Cardinality:** expresses the minimum and maximum number of entity occurrences associated with one occurrence of related entity
  - In the ERD, cardinality is indicated by placing the appropriate numbers beside the entities, using the format (x, y)

An association between entities in an ERD is bidirectional, involving participants and expressing connectivity and cardinality. For example, in a relationship between "Student" and "Course," the cardinality might be (1, M), indicating each student (1) can enroll in multiple courses (M).

Existence Dependence

- Existence dependence
    - Entity exists in the database only when it is associated with another related entity occurrence
- Existence independence
    - Entity exists apart from all of its related entities
    - Referred to as a strong entity or regular entity

Existence dependence occurs when an entity exists only when associated with another related entity occurrence. For example, in a university database, a course offering entity exists only when associated with a course entity. Existence independence, on the other hand, means an entity exists independently, such as a student entity in the university database.

Relationship Strength

- Weak (non-identifying) relationship
  - Primary key of the related entity does not contain a primary key component of the parent entity
- Strong (identifying) relationships
  - Primary key of the related entity contains a primary key component of the parent entity

A weak (non-identifying) relationship occurs when the primary key of the related entity does not include any primary key component of the parent entity. For example, consider a relationship between a department and a course offering in a university database. The department's primary key does not become part of the course offering's primary key. Conversely, in strong (identifying) relationships, the primary key of the related entity includes a primary key component of the parent entity.

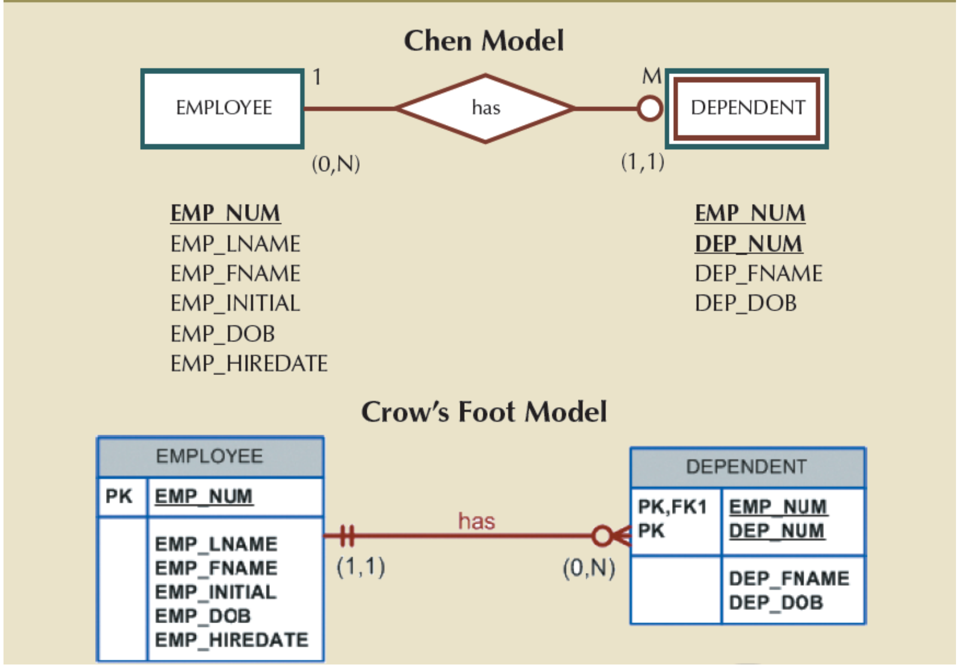
Weak Entities (1 of 3)

- Conditions of a weak entity
  - Existence-dependent
  - Has a primary key that is partially or totally derived from parent entity in the relationship
- Database designer determines whether an entity is weak
  - Based on business rules

A weak entity is existence-dependent, meaning it relies on its relationship with another entity for its existence. Its primary key is partially or entirely derived from the parent entity. For instance, consider an "Order" entity in a database system. It might be weak because its primary key, such as "OrderID," is generated by combining the customer's ID and the date of the order, both of which come from the "Customer" entity.

Weak Entities (2 of 3)

FIGURE 4.10 A WEAK ENTITY IN AN ERD



A weak entity is an entity in a database that does not have a primary key attribute on its own and is dependent on another entity, known as the "owning" or "parent" entity. A weak entity typically relies on the existence of a related entity for identification.

Weak Entities (3 of 3)

FIGURE 4.11 A WEAK ENTITY IN A STRONG RELATIONSHIP

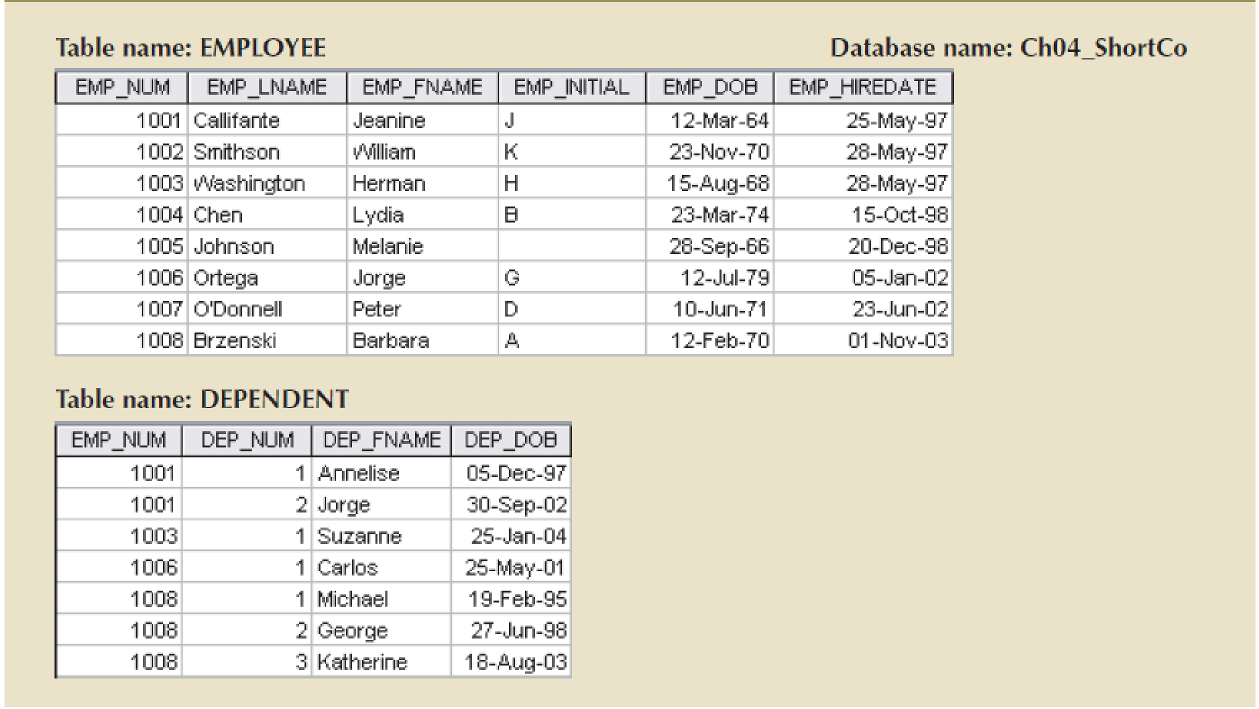


Table name: EMPLOYEE

Database name: Ch04_ShortCo

EMP_NUM	EMP_LNAME	EMP_FNAME	EMP_INITIAL	EMP_DOB	EMP_HIREDATE
1001	Callifante	Jeanine	J	12-Mar-64	25-May-97
1002	Smithson	William	K	23-Nov-70	28-May-97
1003	Washington	Herman	H	15-Aug-68	28-May-97
1004	Chen	Lydia	B	23-Mar-74	15-Oct-98
1005	Johnson	Melanie		28-Sep-66	20-Dec-98
1006	Ortega	Jorge	G	12-Jul-79	05-Jan-02
1007	O'Donnell	Peter	D	10-Jun-71	23-Jun-02
1008	Brzenski	Barbara	A	12-Feb-70	01-Nov-03

Table name: DEPENDENT

EMP_NUM	DEP_NUM	DEP_FNAME	DEP_DOB
1001	1	Annelise	05-Dec-97
1001	2	Jorge	30-Sep-02
1003	1	Suzanne	25-Jan-04
1006	1	Carlos	25-May-01
1008	1	Michael	19-Feb-95
1008	2	George	27-Jun-98
1008	3	Katherine	18-Aug-03

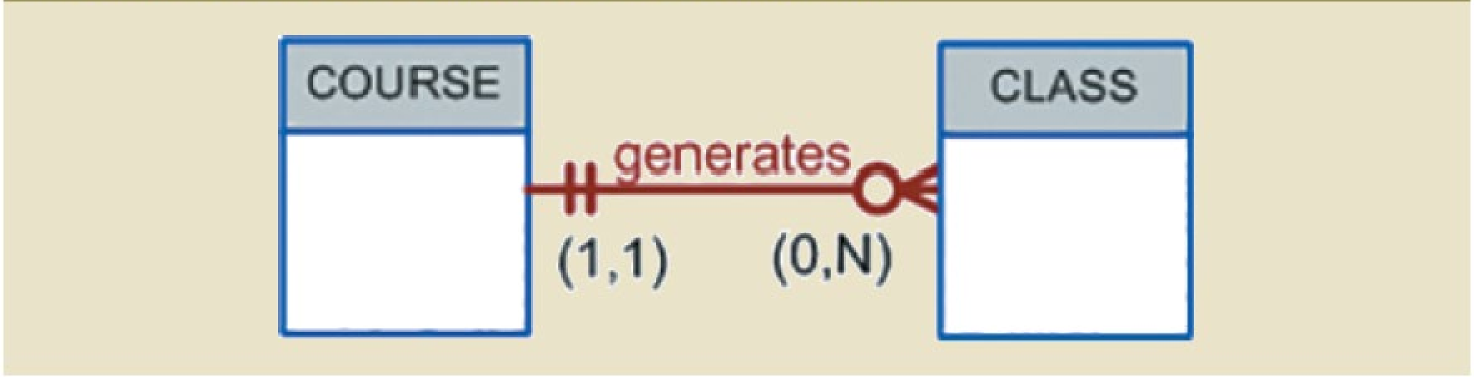
Relationship Participation (1 of 3)

- Optional participation in a relationship means that one entity occurrence does not necessarily require a corresponding entity occurrence in that relationship. For example, in a library database, a "Book" entity can be associated with a "Publisher" entity through an optional participation relationship, as not all books may have a known publisher.

- Mandatory participation, on the other hand, means that one entity occurrence must have a corresponding entity occurrence in the relationship. For instance, in a university database, the relationship between a "Student" entity and an "Advisor" entity might involve mandatory participation, as each student must be assigned an advisor.

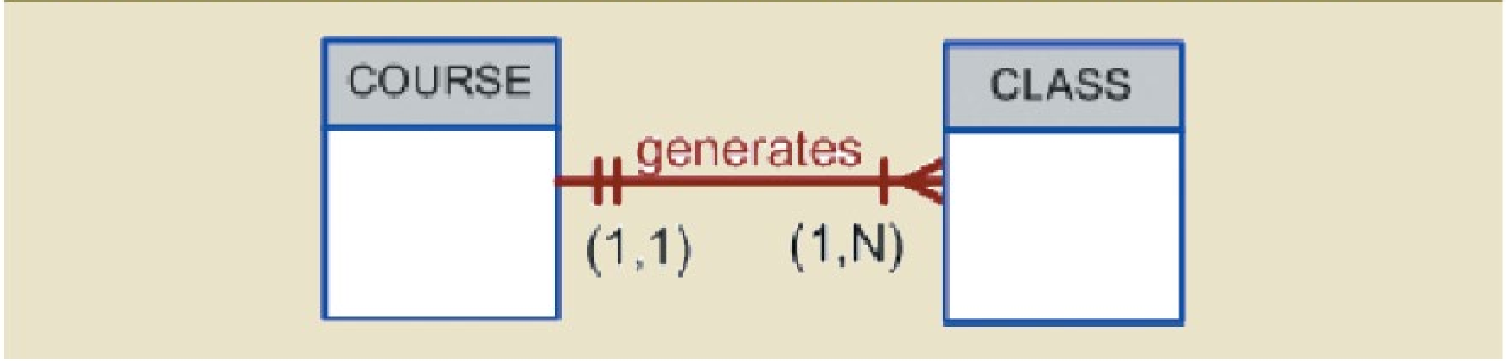
Relationship Participation (2 of 3)

FIGURE 4.13 CLASS IS OPTIONAL TO COURSE



Relationship Participation (3 of 3)

FIGURE 4.14 COURSE AND CLASS IN A MANDATORY RELATIONSHIP



Relationship Degree (1 of 2)

- Relationship degree indicates the number of entities or participants associated with a relationship

  - **Unary Relationship**: An association maintained within a single entity.

  Example: In a "Manager" entity, where each manager reports to another manager within the same entity.

  - **Binary Relationship**: Association between two entities.

  Example: "Employee" and "Department" entities, where each employee belongs to one department, and each department employs multiple employees.

  - **Ternary Relationship**: Association involving three entities.

  Example: In a "Project" management system, where "Employee," "Task," and "Project" entities are associated, indicating which employees are assigned to which tasks within specific projects.

  - **Recursive Relationship**: A relationship existing within a single entity type.

  Example: In an "Employee" entity, where each employee has a "Supervisor" attribute referring to another employee in the same entity, representing a hierarchical structure.

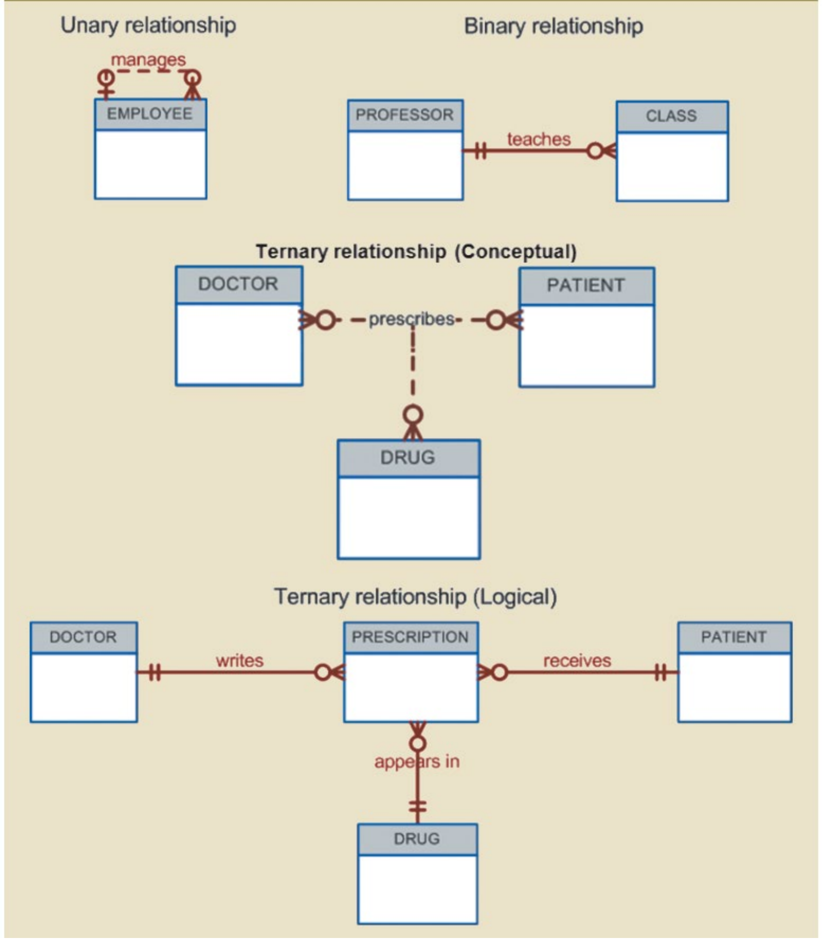
Relationship Degree (2 of 2)

In ER modeling:

- Conceptual Model: Establishes the basis for identifying and describing main data objects, providing a high-level description.

- Logical Model: Represents data as perceived by users, abstracting away physical storage details.

FIGURE 4.15 THREE TYPES OF RELATIONSHIP DEGREE



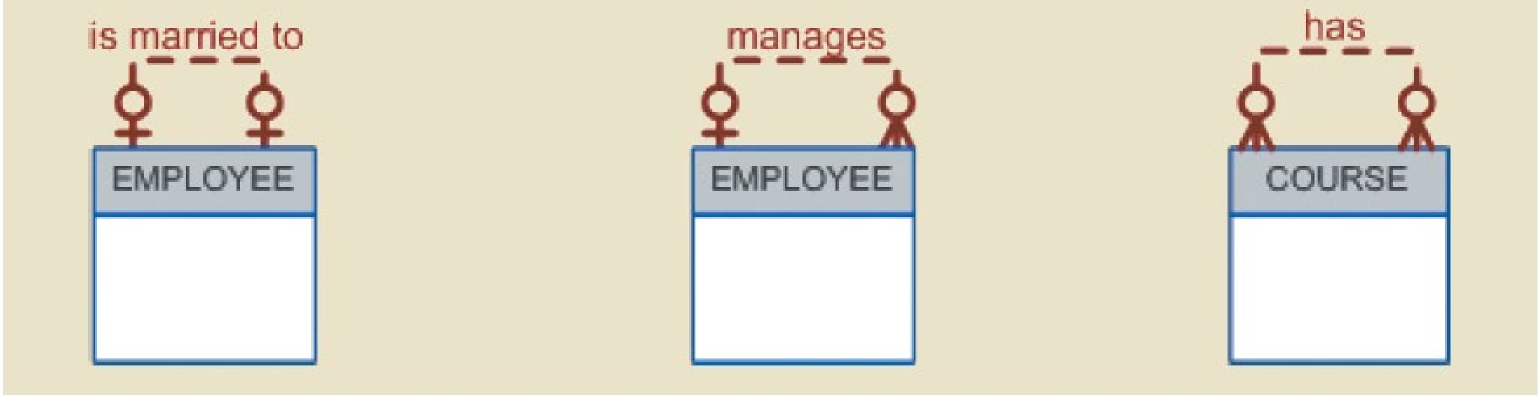
Recursive Relationships (1 of 2)

- Relationship can exist between occurrences of the same entity set
  - Naturally, such a condition is found within a unary relationship
    - Common in manufacturing industries
- One common pitfall when working with unary relationships is to confuse participation with referential integrity
  - Similar because they are both implemented through constraints on the same set of attributes

In a manufacturing context, consider the "Supervises" relationship within an "Employee" entity set. An example could be an employee supervising another employee for training purposes. This demonstrates a unary relationship where an employee participates in the supervision of another employee, distinct from the typical hierarchical reporting structure.

Recursive Relationships (2 of 2)

FIGURE 4.17 AN ER REPRESENTATION OF RECURSIVE RELATIONSHIPS



In an organization, an employee may marry another employee. This recursive relationship signifies that an employee can be married to another employee within the same organization. For example, John is married to Sarah, and both John and Sarah are employees of the company.

Within a company hierarchy, an employee can manage other employees. This recursive relationship illustrates that an employee can hold a managerial position over another employee. For instance, Sarah manages John, who is also an employee under her supervision.

In an academic institution, a course can serve as a prerequisite for another course. This recursive relationship indicates that one course can be a prerequisite for another course. For example, Security 1 is a prerequisite for Security 2, forming a recursive relationship between courses.

Associative (Composite) Entities (1 of 2)

- Used to represent an M:N relationship between two or more entities
- Has a 1:M relationship with the parent entities
  - Composed of the primary key attributes of each parent entity
- May also contain additional attributes that play no role in connective process

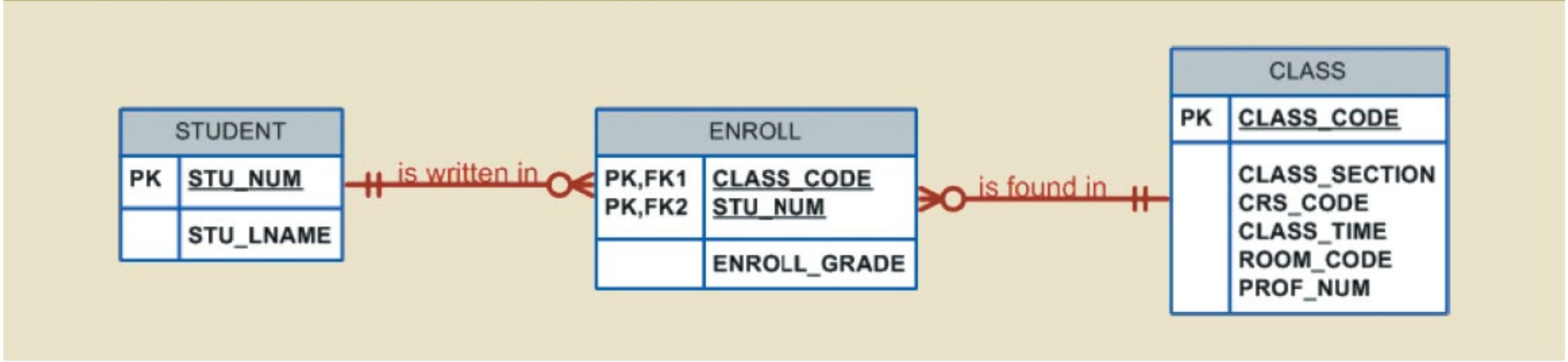
The associative entity, often called a bridge entity, is employed to represent a many-to-many (M:N) relationship between two or more entities. It maintains a one-to-many (1:M) relationship with the parent entities and is composed of the primary key attributes of each parent entity. Additionally, it may include supplementary attributes that do not participate in the connective process.

A classic example of an associative entity is a "Student-Course" relationship in a university database. Here, students can enroll in multiple courses, and each course can have multiple students. The associative entity "Enrollment" would link students and courses, containing attributes such as enrollment date or grade.

CENGAGE

Associative (Composite) Entities (2 of 2)

FIGURE 4.25 A COMPOSITE ENTITY IN AN ERD



Here, students can enroll in multiple classes/ courses, and each course can have multiple students. The associative entity "Enroll" would link students and courses, containing attributes such as enrollment date or grade.

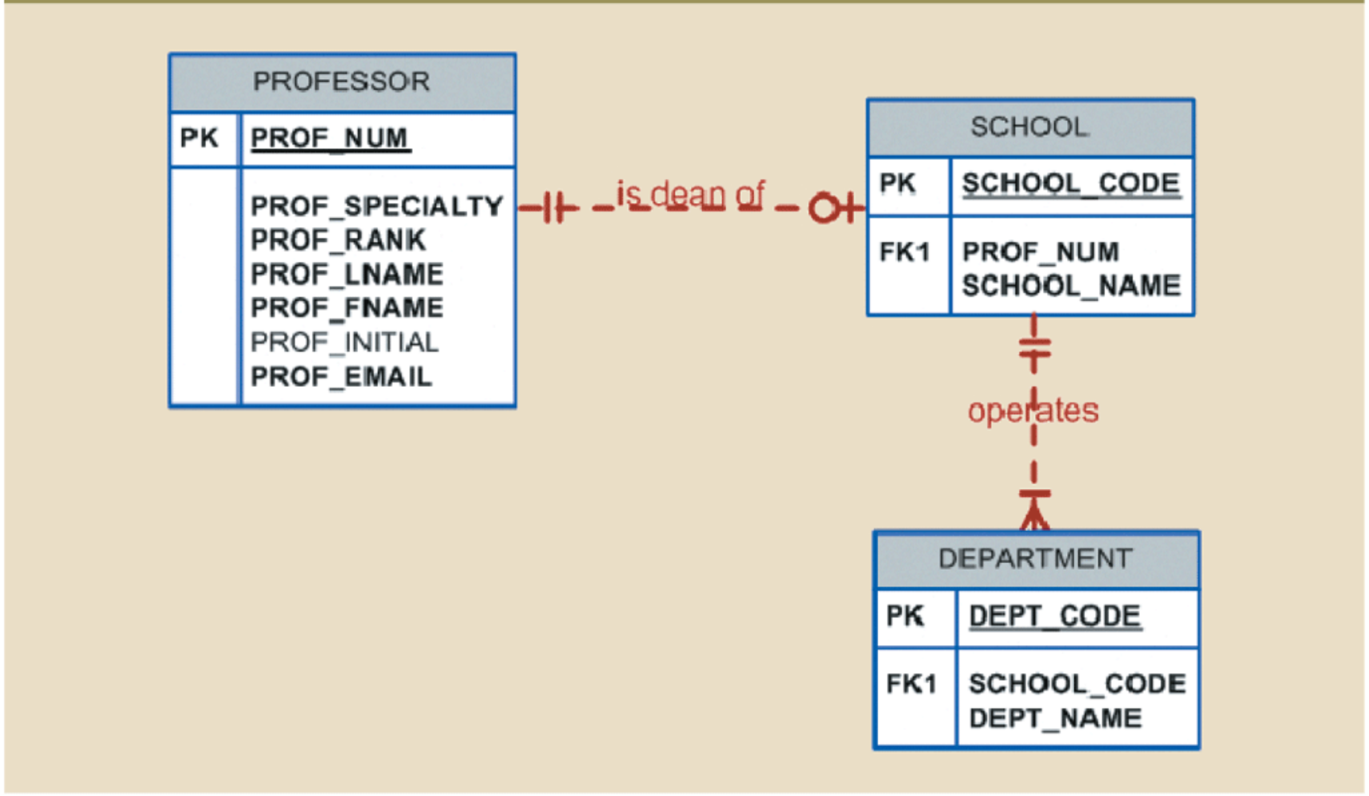
Developing an ER Diagram (1 of 11)

Building an Entity-Relationship Diagram (ERD) involves several key activities:

1. **Gathering Organization Description**: Begin by collecting a comprehensive description of the organization's operations, including its functions, processes, and data requirements.
2. **Identifying Business Rules**: Based on the organization's description, extract business rules that dictate how data should be stored and managed within the system. These rules govern the relationships between different entities and the constraints on data attributes.
3. **Entity and Relationship Identification**: Use the gathered business rules to identify the main entities (such as customers, products, or orders) and the relationships between them. This step involves determining the cardinality and participation constraints for each relationship.
4. **Developing the Initial ERD**: Construct the initial Entity-Relationship Diagram using standard notation, representing entities as rectangles, relationships as diamonds, and attributes as ovals connected to entities.
5. **Attribute and Primary Key Definition**: For each entity, identify and define its attributes, including primary keys that uniquely identify each entity instance. Ensure that attributes capture all necessary data elements required by the organization.
6. **Revision and Review**: Review the initial ERD for accuracy, completeness, and consistency with the organization's requirements. Revise the diagram as needed based on feedback from stakeholders and domain experts.

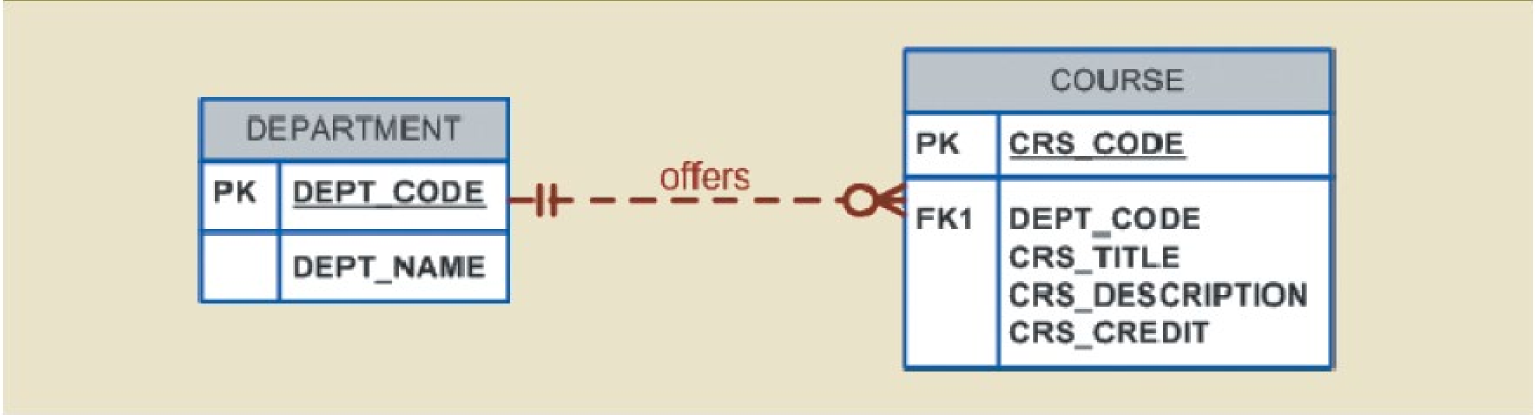
Developing an ER Diagram (2 of 11)

FIGURE 4.26 THE FIRST TINY COLLEGE ERD SEGMENT



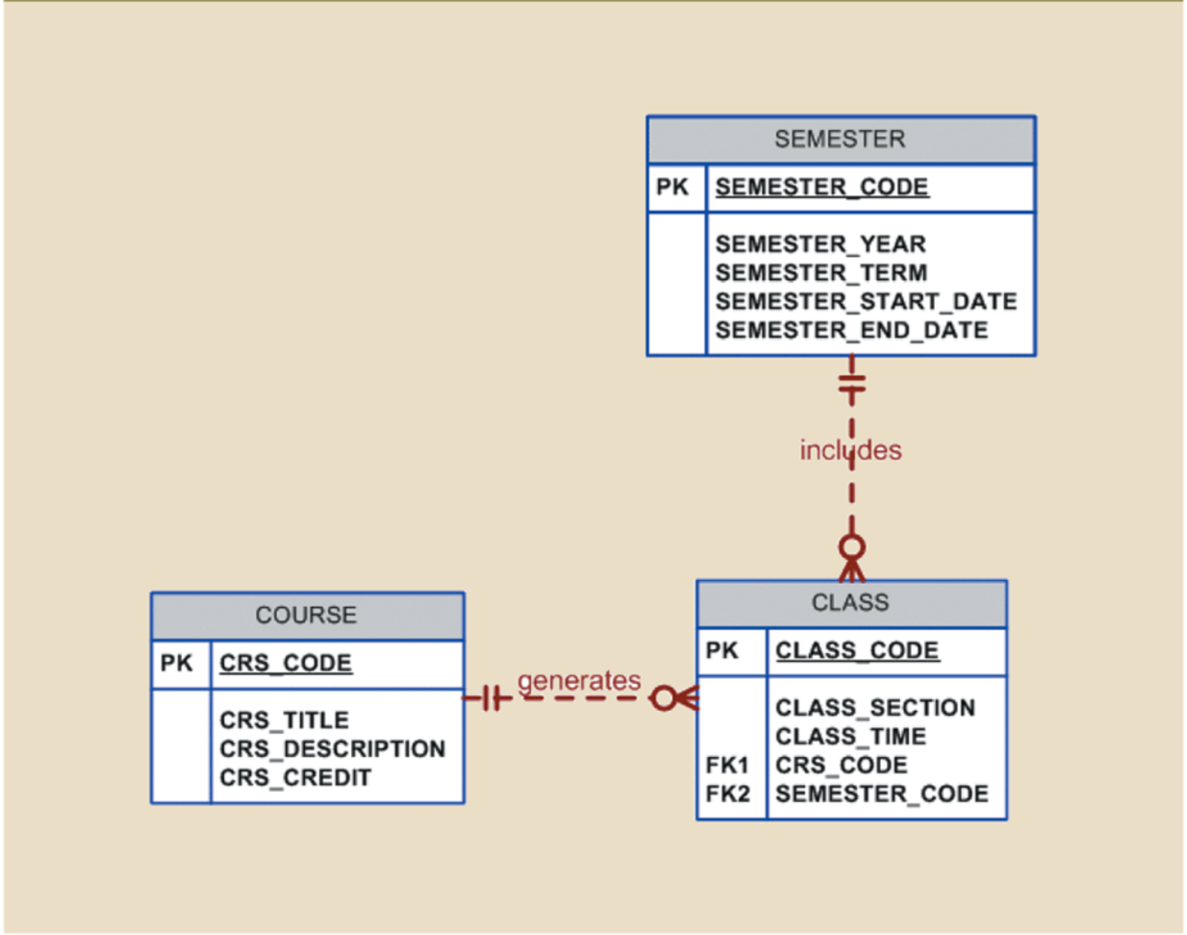
Developing an ER Diagram (3 of 11)

FIGURE 4.27 THE SECOND TINY COLLEGE ERD SEGMENT



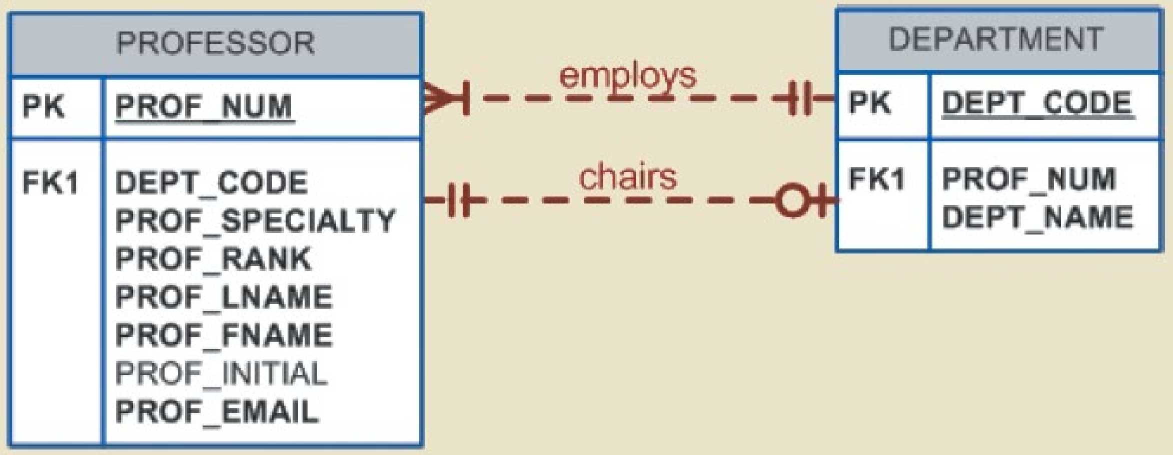
Developing an ER Diagram (4 of 11)

FIGURE 4.28 THE THIRD TINY COLLEGE ERD SEGMENT



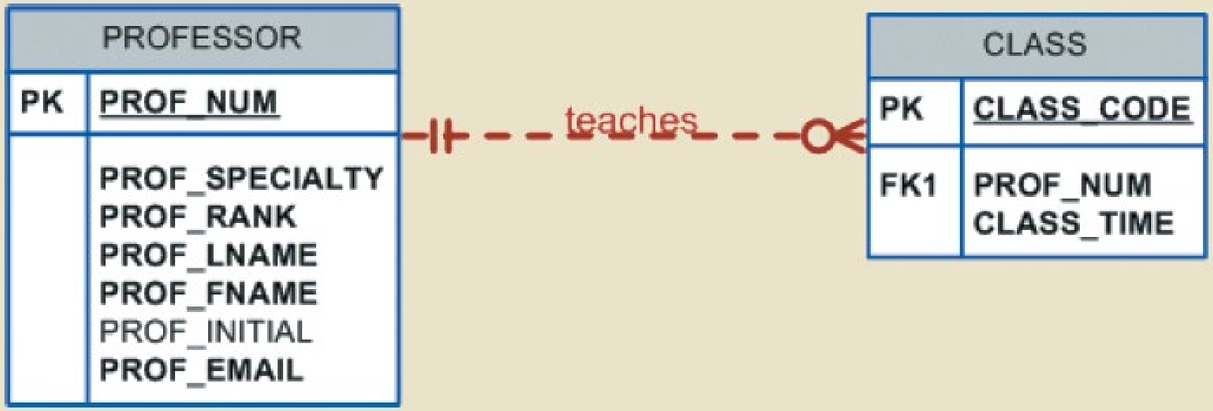
Developing an ER Diagram (5 of 11)

FIGURE 4.29 THE FOURTH TINY COLLEGE ERD SEGMENT



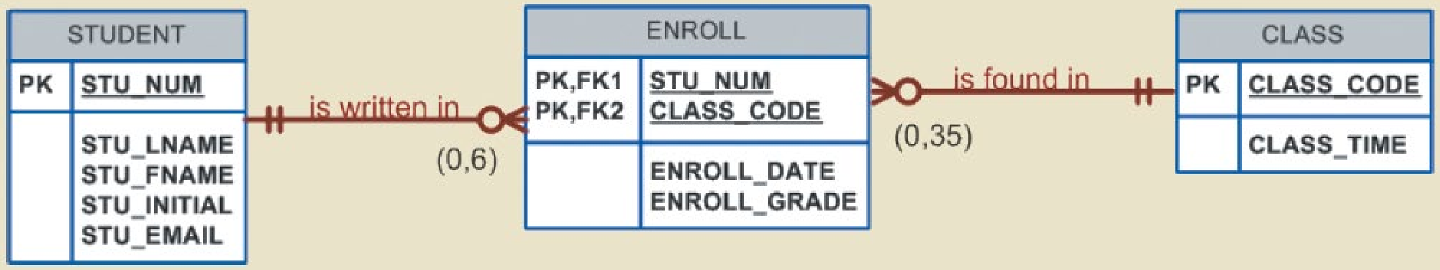
Developing an ER Diagram (6 of 11)

FIGURE 4.30 THE FIFTH TINY COLLEGE ERD SEGMENT



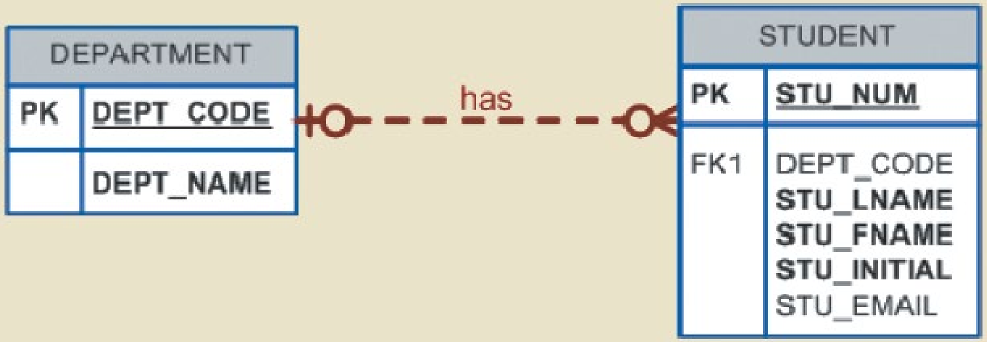
Developing an ER Diagram (7 of 11)

FIGURE 4.31 THE SIXTH TINY COLLEGE ERD SEGMENT



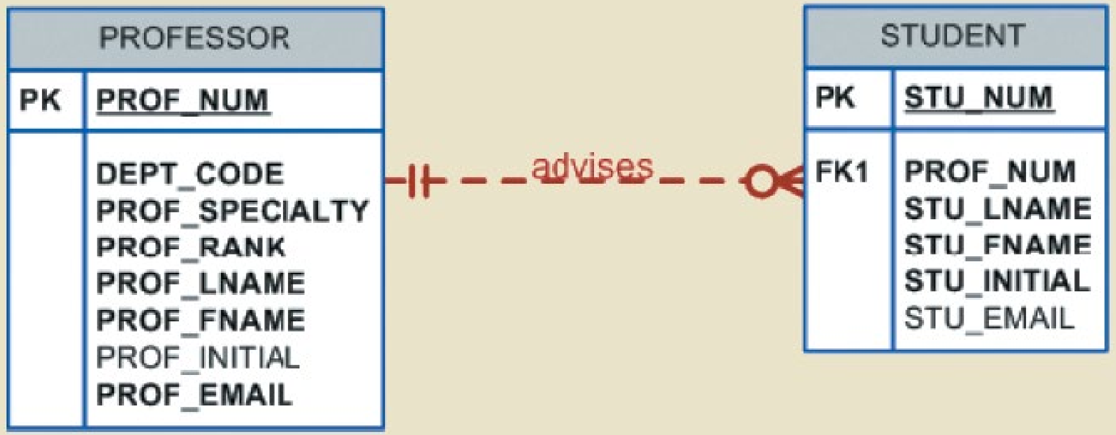
Developing an ER Diagram (8 of 11)

FIGURE 4.32 THE SEVENTH TINY COLLEGE ERD SEGMENT



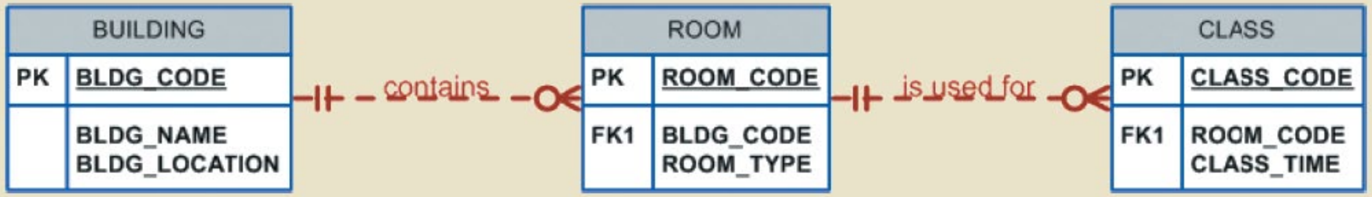
Developing an ER Diagram (9 of 11)

FIGURE 4.33 THE EIGHT TINY COLLEGE ERD SEGMENT



Developing an ER Diagram (10 of 11)

FIGURE 4.34 THE NINTH TINY COLLEGE ERD SEGMENT



Developing an ER Diagram (11 of 11)

Table 4.4	Components of the ERM		
Entity	**Relationship**	**Connectivity**	**Entity**
SCHOOL	operates	1:M	DEPARTMENT
DEPARTMENT	has	1:M	STUDENT
DEPARTMENT	employs	1:M	PROFESSOR
DEPARTMENT	offers	1:M	COURSE
COURSE	generates	1:M	CLASS
SEMESTER	includes	1:M	CLASS
PROFESSOR	is dean of	1:1	SCHOOL
PROFESSOR	chairs	1:1	DEPARTMENT
PROFESSOR	teaches	1:M	CLASS
PROFESSOR	advises	1:M	STUDENT
STUDENT	enrolls in	M:N	CLASS
BUILDING	contains	1:M	ROOM
ROOM	is used for	1:M	CLASS
	Note: ENROLL is the composite entity that implements the M:N relationship “STUDENT enrolls in CLASS.”		

CENGAGE

Database Design Challenges: Conflicting Goals (1 of 3)

- Database designers must often make design compromises that are triggered by conflicting goals
  - Database design must conform to design standards
  - High processing speed may limit the number and complexity of logically desirable relationships
  - Maximum information generation may lead to loss of clean design structures and high transaction speed

Database Design Challenges: Conflicting Goals (2 of 3)

In simplified terms, designing a database poses challenges because **different goals might clash**, requiring compromises. Designers must follow established standards, but there's a trade-off: focusing on high-speed processing might limit the complexity of relationships. On the other hand, striving for maximum information generation could sacrifice a clean and efficient design, potentially impacting transaction speed. Hence, balancing these conflicting objectives is a key challenge in effective database design.

Imagine you're designing a database for an online store. You want it to be fast, handle a lot of transactions, and provide detailed information about customers and products. However, there are conflicts:

1.Speed vs. Complexity:

1. **Goal:** You want the website to load quickly for users.
2. **Challenge:** Including too many intricate/complex details about each product might slow down the website.
3. **Compromise:** You decide to display essential product information on the main page, reserving the detailed specifics for a separate page.

2.Information Generation vs. Clean Design:

1. **Goal:** You aim to gather as much data as possible for market analysis.
2. **Challenge:** Gathering extensive information on every customer and transaction might make the database messy.
3. **Compromise:** You choose to store only essential customer and transaction data in the main database, with the option to extract detailed reports for analysis separately.

In these examples, conflicting goals involve trade-offs between website speed and complexity, as well as between information generation and maintaining a clean database structure. Balancing these compromises is crucial for a successful and efficient database design.

CENGAGE

Database Design Challenges: Conflicting Goals (3 of 3)

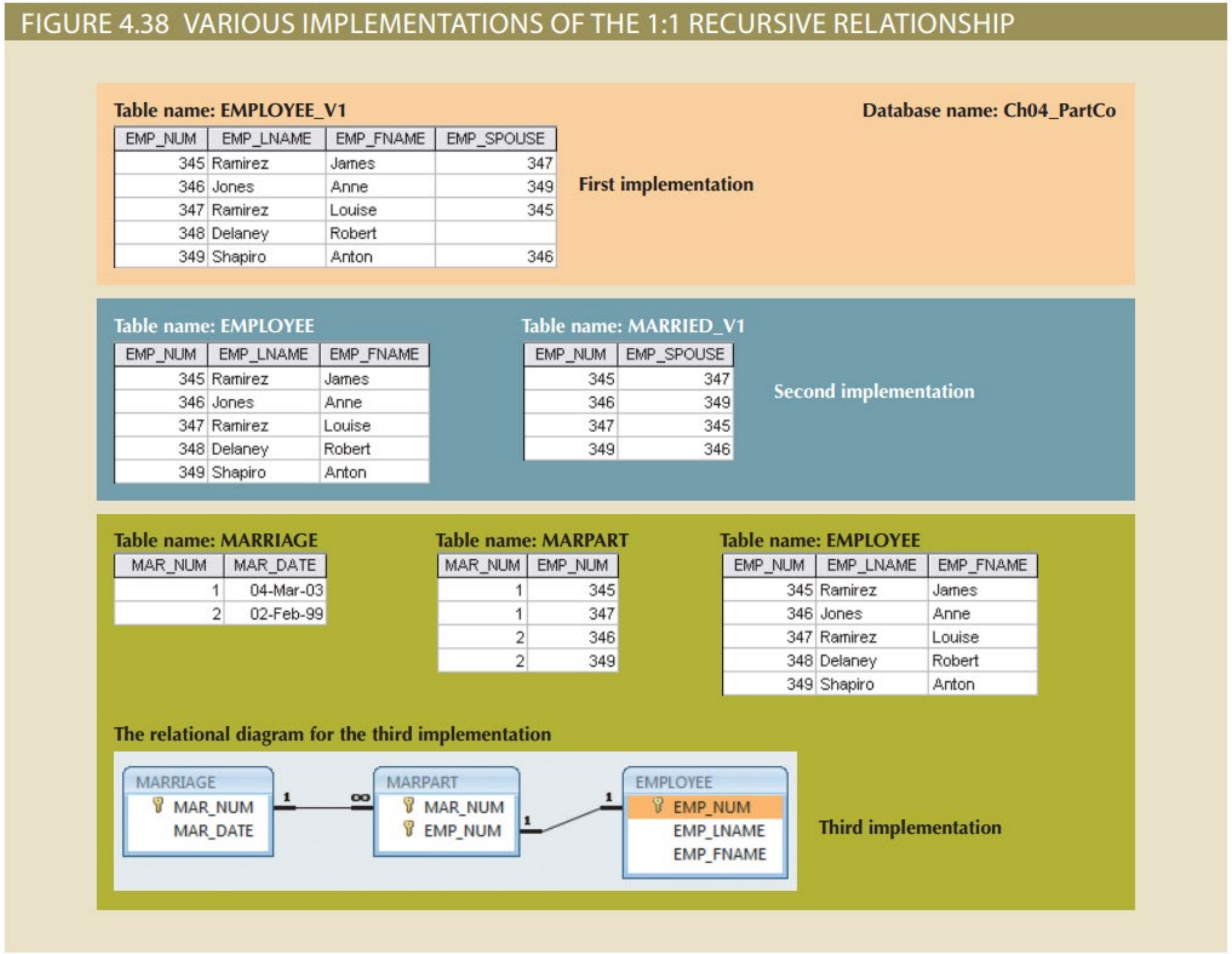


FIGURE 4.38 VARIOUS IMPLEMENTATIONS OF THE 1:1 RECURSIVE RELATIONSHIP

Database name: Ch04_PartCo

First implementation

Table name: EMPLOYEE_V1

EMP_NUM	EMP_LNAME	EMP_FNAME	EMP_SPOUSE
345	Ramirez	James	347
346	Jones	Anne	349
347	Ramirez	Louise	345
348	Delaney	Robert	
349	Shapiro	Anton	346

Second implementation

Table name: EMPLOYEE

EMP_NUM	EMP_LNAME	EMP_FNAME
345	Ramirez	James
346	Jones	Anne
347	Ramirez	Louise
348	Delaney	Robert
349	Shapiro	Anton

Table name: MARRIED_V1

EMP_NUM	EMP_SPOUSE
345	347
346	349
347	345
349	346

Third implementation

Table name: MARRIAGE

MAR_NUM	MAR_DATE
1	04-Mar-03
2	02-Feb-99

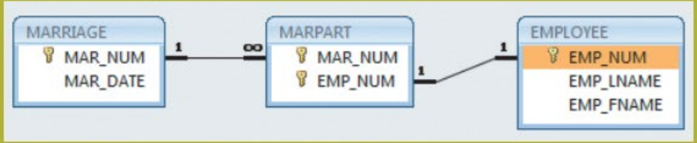
Table name: MARPART

MAR_NUM	EMP_NUM
1	345
1	347
2	346
2	349

Table name: EMPLOYEE

EMP_NUM	EMP_LNAME	EMP_FNAME
345	Ramirez	James
346	Jones	Anne
347	Ramirez	Louise
348	Delaney	Robert
349	Shapiro	Anton

The relational diagram for the third implementation



Summary

- The ERM uses ERDs to represent the conceptual database as viewed by the end user
- Connectivity describes the relationship classification (1:1, 1:M, or M:N)
- In the ERM, an M:N relationship is valid at the conceptual level
- ERDs may be based on many different ERMs
- Unified Modeling Language (UML) class diagrams are used to represent the static data structures in a data model
- Database designers, no matter how well they can produce designs that conform to all applicable modeling conventions, are often forced to make design compromises