Unit 5: Advanced Data Modeling

Learning Objectives

- In this chapter, you will learn:
  - About the extended entity relationship (EER) model
  - How entity clusters are used to represent multiple entities and relationships
  - The characteristics of good primary keys and how to select them
  - How to use flexible solutions for special data-modeling cases

Extended Entity Relationship Model (EERM)

- Result of adding more semantic constructs to the original entity relationship (ER) model

- Question:
  - What is the meaning of semantics?

- **EER diagram (EERD)**: Uses the EER model

Use Case: An Aviation Company

- Employee (common characteristics – firstname, hiredate etc.)
  - Pilots
    - License, ratings, flight_hours (unique to only pilots)
  - Accountants
  - Clerks
  - Managers

- Saving the details of all employees in a table will lead to lots of nulls or the need for dummy entries

Entity Supertypes and Subtypes

- **Entity supertype:** Generic entity type related to one or more entity subtypes
  - Contains common characteristics
- **Entity subtype:** Contains unique characteristics of each entity subtype
- Criteria to determine the usage
  - There must be different, identifiable kinds of the entity in the user's environment
  - The different kinds of instances should each have one or more attributes that are unique to that kind of instance

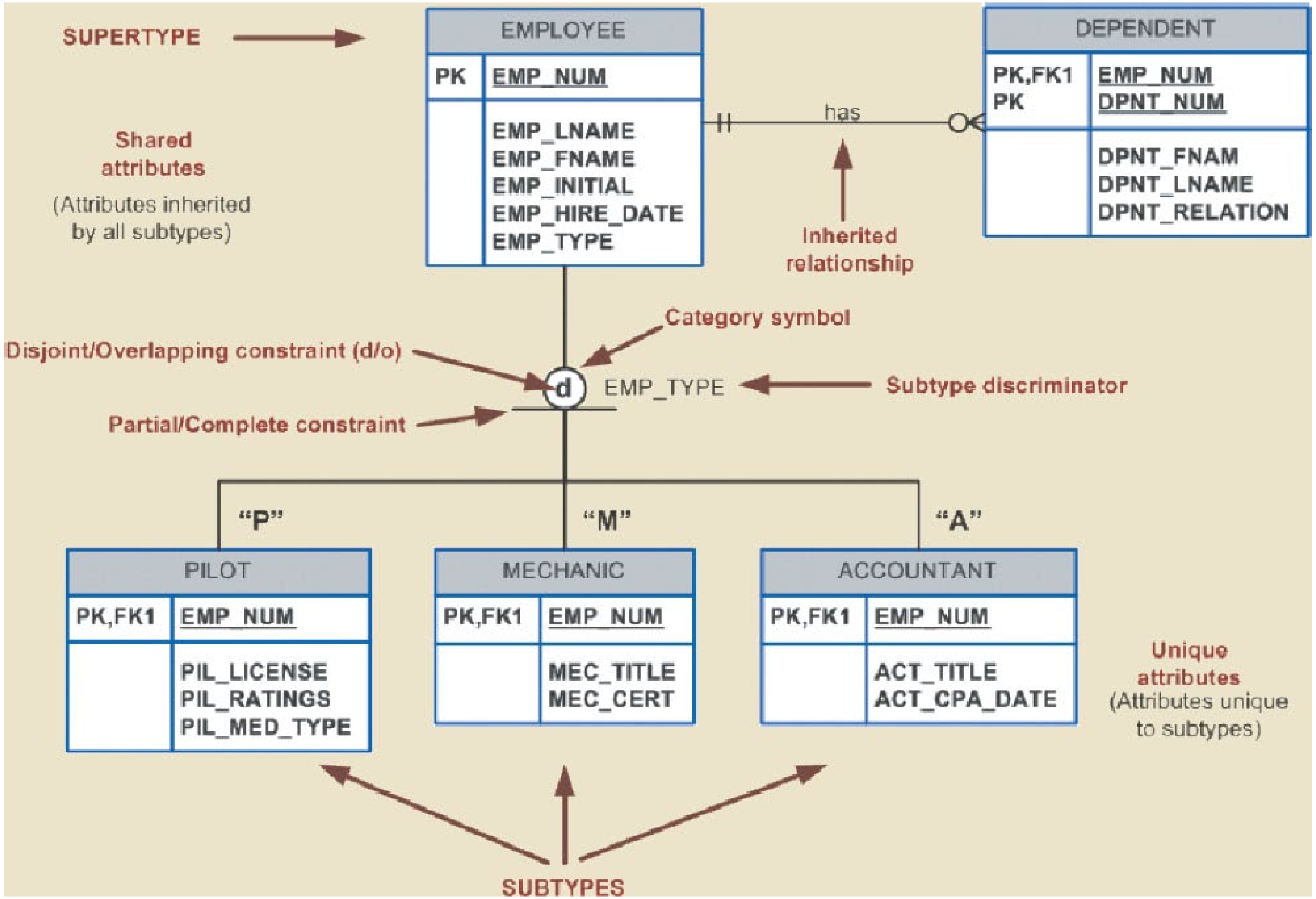
Specialization Hierarchy (1 of 2)

- Depicts arrangement of higher-level entity supertypes and lower-level entity subtypes
- Relationships are described in terms of “is-a” relationships
- Subtype exists within the context of a supertype
- Every subtype has only one supertype to which it is directly related
- Supertype can have many subtypes

Specialization Hierarchy (2 of 2)

- Provides the means to:
  - Support attribute inheritance
  - Define a special supertype attribute known as the subtype discriminator
  - Define disjoint/overlapping constraints and complete/partial constraints

Figure 5.2 - Specialization Hierarchy



CENGAGE

Inheritance

- Enables an entity subtype to inherit attributes and relationships of the supertype
- All entity subtypes inherit their primary key attribute from their supertype
- At the implementation level, supertype and its subtype(s) maintain a 1:1 relationship
- Entity subtypes inherit all relationships in which supertype entity participates
- Lower-level subtypes inherit all attributes and relationships from its upper-level supertypes

Subtype Discriminator

- Attribute in the supertype entity that determines to which entity subtype the supertype occurrence is related

Disjoint and Overlapping Constraints

- **Disjoint subtypes:** Contain a unique subset of the supertype entity set
  - Known as **nonoverlapping subtypes**
  - Implementation is based on the value of the subtype discriminator attribute in the supertype
  - Employee can either be a pilot or mechanic; not both
- **Overlapping subtypes:** Contain nonunique subsets of the supertype entity set
  - Implementation requires the use of one discriminator attribute for each subtype
  - Staff can be both a professor and an administrator

Figure 5.4 - Specialization Hierarchy with Overlapping Subtypes

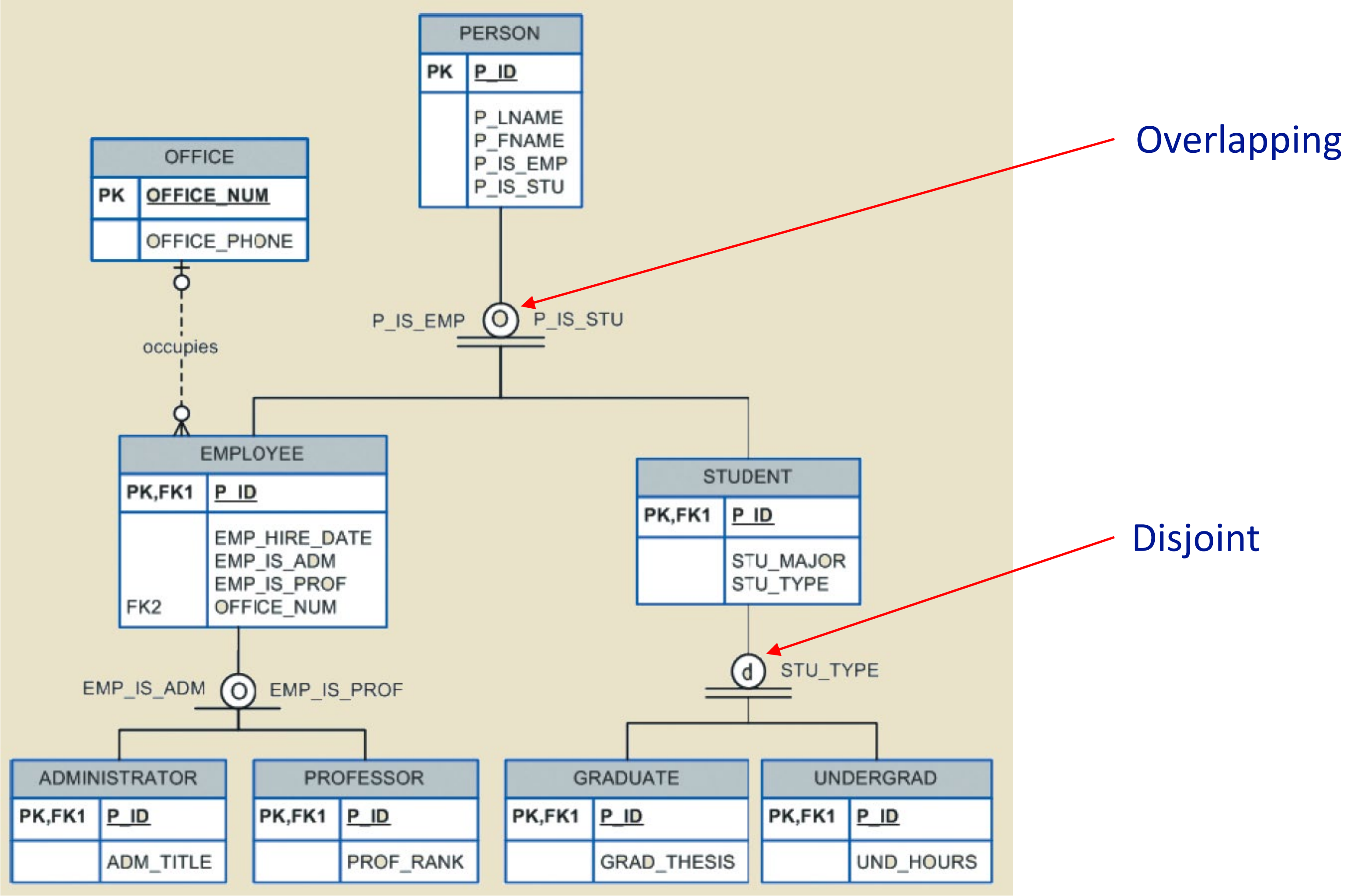


Table 5.1 - Discriminator Attributes with Overlapping Subtypes

DISCRIMINATOR ATTRIBUTES OF PROFESSOR	DISCRIMINATOR ATTRIBUTES OF ADMINISTRATOR	COMMENT
Y	N	The Employee is a member of the Professor subtype.
N	Y	The Employee is a member of the Administrator subtype.
Y	Y	The Employee is both a Professor and an Administrator.

Completeness Constraint

- Specifies whether each supertype occurrence must also be a member of at least one subtype
- Types
  - **Partial completeness:** Not every supertype occurrence is a member of a subtype
  - **Total completeness:** Every supertype occurrence must be a member of a subtype

Table 5.2 - Specialization Hierarchy Constraint Scenarios

TYPE	DISJOINT CONSTRAINT	OVERLAPPING CONSTRAINT
Partial 	Supertype has optional subtypes. Subtype discriminator can be null. Subtype sets are unique.	Supertype has optional subtypes. Subtype discriminators can be null. Subtype sets are not unique.
Total 	Every supertype occurrence is a member of only one subtype. Subtype discriminator cannot be null. Subtype sets are unique.	Every supertype occurrence is a member of at least one subtype. Subtype discriminators cannot be null. Subtype sets are not unique.

Specialization and Generalization

Specialization

- Top-down process
- Identifies lower-level, more specific entity subtypes from a higher-level entity supertype
- Based on grouping **unique** characteristics and relationships of the subtypes

Generalization

- Bottom-up process
- Identifies a higher-level, more generic entity supertype from lower-level entity subtypes
- Based on grouping **common** characteristics and relationships of the subtypes

Entity Cluster

- Virtual entity type used to represent multiple entities and relationships in ERD
- Combines multiple interrelated entities into a single, abstract entity object.
- It is not actually an entity in the final ERD.
- Helps to simplify the ERD; enhances readability.
- Avoid the display of attributes to eliminate complications that result when the inheritance rules change

FIGURE 4.35 THE COMPLETED TINY COLLEGE ERD

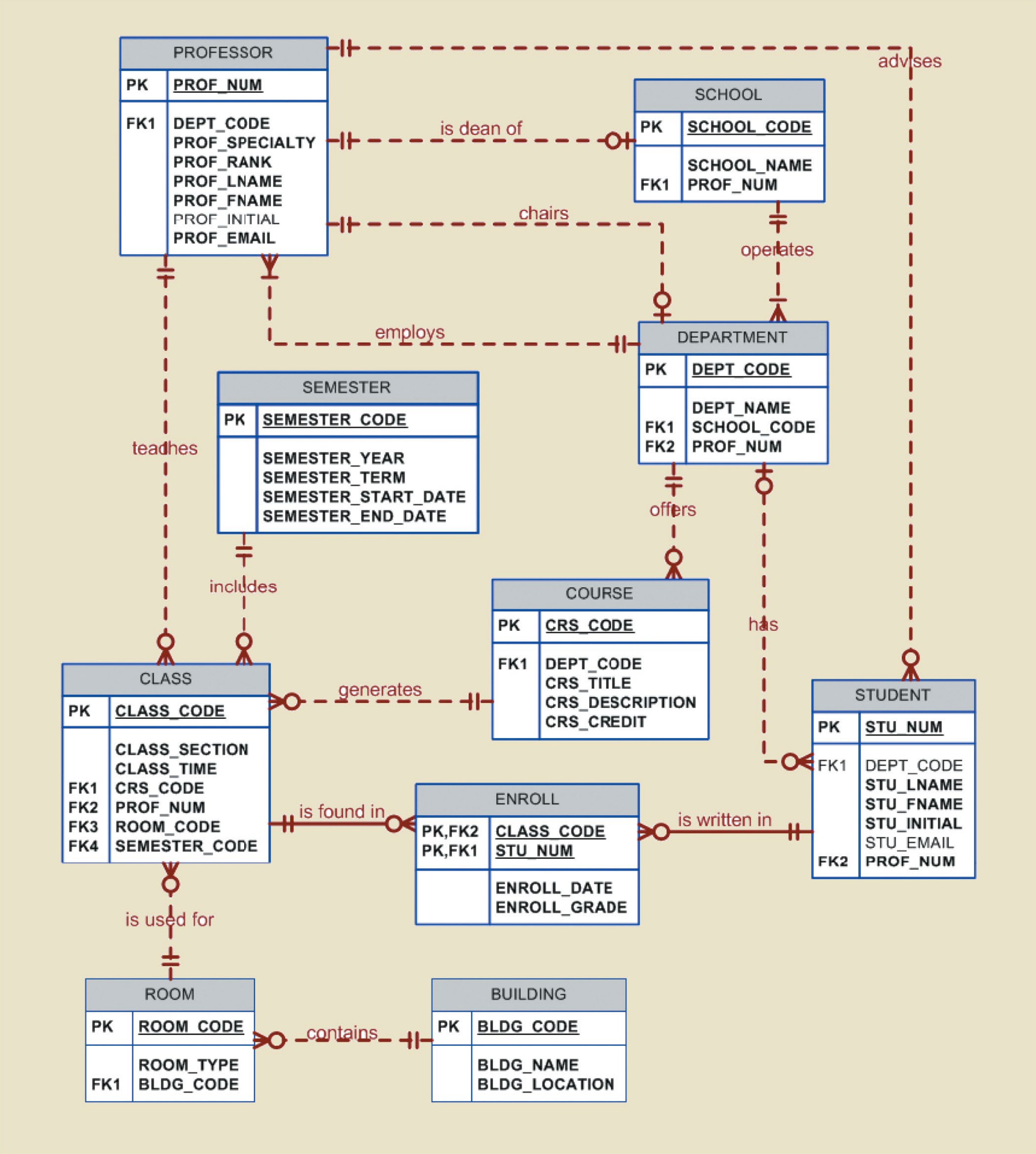
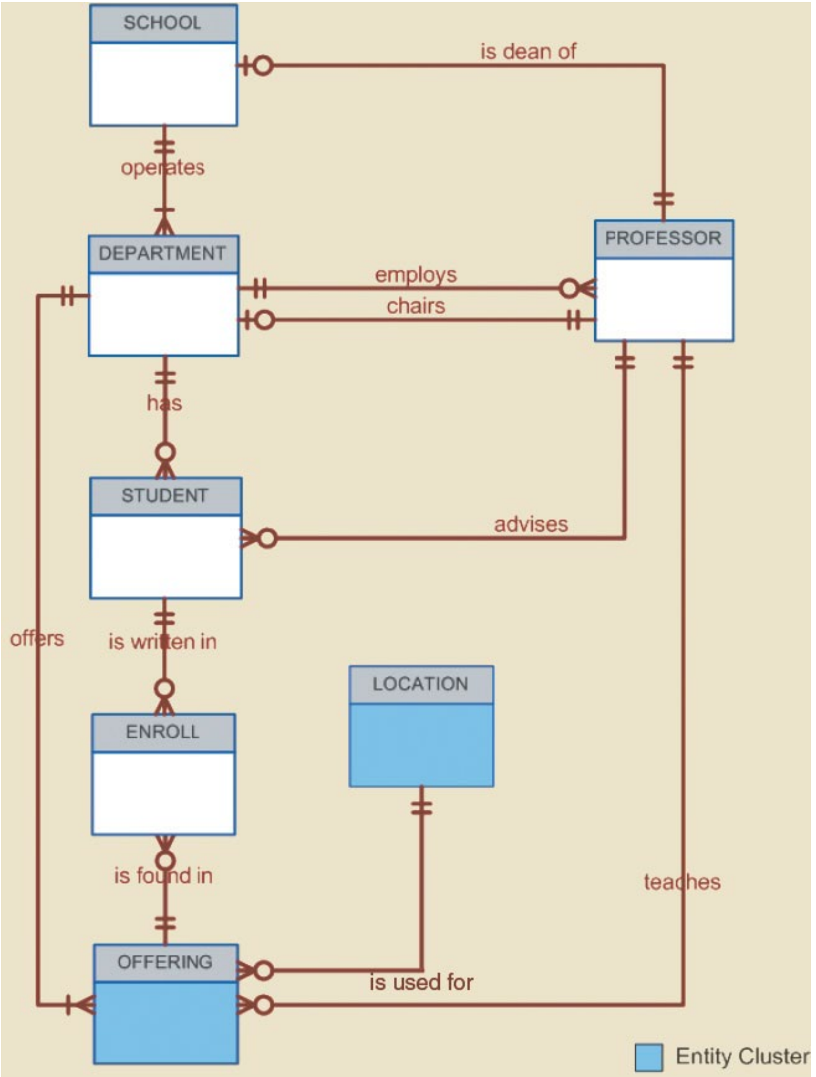


Figure 5.5 - Tiny College ERD Using Entity Clusters



NEXT: Selecting a Primary Key

What is a Primary Key?

- Question:
- Which of the following is TRUE:
  - A Primary key MUST be made up of a combination of attributes
  - A Primary key MUST be a single attribute
  - A Primary key can be a single attribute or be made up of a combination of attributes

Primary Keys

- Single attribute or a combination of attributes, which uniquely identifies each entity instance
  - Guarantees entity integrity
  - Works with foreign keys to implement relationships

Natural Keys or Natural Identifier

- Real-world identifier used to uniquely identify real-world objects
  - Familiar to end users and forms part of their day-to-day business vocabulary
  - Also known as natural identifier
  - Used as the primary key of the entity being modeled
  - E.g Student_number, South African ID number

Desirable Primary Key Characteristics - 1

- Unique values
- Non intelligent
  - should not have embedded semantic meaning
  - eg. STD_NO preferred to STD_NAME, use STD_NAME as a descriptive characteristic of an entity
- No change over time
  - descriptive attributes tend to change, choose attributes that are permanent and unchangeable
- Preferably single-attribute
  - simplifies implementation of foreign keys

Desirable Primary Key Characteristics - 2

- Preferably numeric
  - better managed, easily implemented using auto-increment functionality
- Security-compliant
  - void of any attribute(s) that might be considered a security risk. eg: South African ID as PK

Use of Composite Primary Keys (1 of 2)

- Athough, they are not preferred, they are sometimes useful/needed:
  - Identifiers of composite (bridge) entities
    - Each primary key combination is allowed once in M:N relationship
  - Identifiers of weak entities
    - Weak entity has a strong identifying relationship with the parent entity

Use of Composite Primary Keys (2 of 2)

- When used as identifiers of weak entities, represent a real-world object that is:
  - Existence-dependent on another real-world object
  - Represented in the data model as two separate entities in a strong identifying relationship

Figure 5.6 - The M:N Relationship between STUDENT and CLASS

Database name: Ch05_Tinycollege

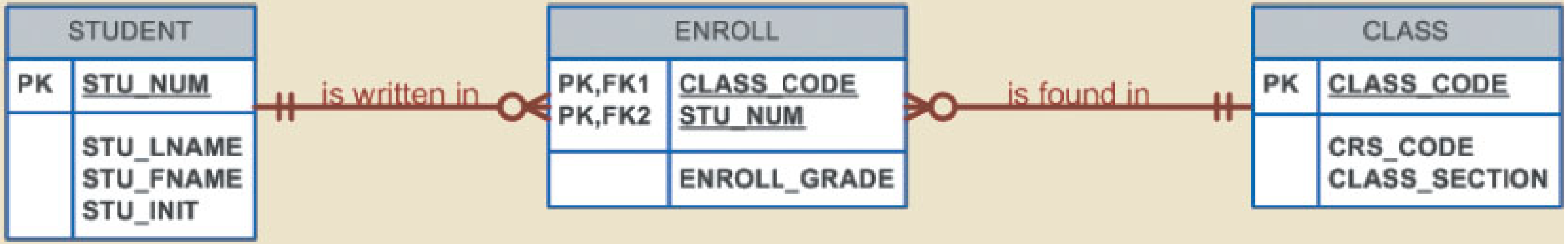


Table name: STUDENT
(first four fields)

STU_NUM	STU_LNAME	STU_FNAME	STU_INIT
321452	Bowser	William	C
324257	Smithson	Anne	K
324258	Brewer	Juliette	
324269	Oblonski	Walter	H
324273	Smith	John	D
324274	Katinga	Raphael	P
324291	Robertson	Gerald	T
324299	Smith	John	B

Table name: ENROLL

CLASS_CODE	STU_NUM	ENROLL_GRADE
10014	321452	C
10014	324257	B
10018	321452	A
10018	324257	B
10021	321452	C
10021	324257	C

Table name: CLASS
(first three fields)

CLASS_CODE	CRS_CODE	CLASS_SECTION
10012	ACCT-211	1
10013	ACCT-211	2
10014	ACCT-211	3
10015	ACCT-212	1
10016	ACCT-212	2
10017	CIS-220	1
10018	CIS-220	2
10019	CIS-220	3
10020	CIS-420	1
10021	QM-261	1
10022	QM-261	2
10023	QM-362	1
10024	QM-362	2
10025	MATH-243	1

15 Minutes break

Surrogate Primary Keys

- A surrogate key is a primary key created by the database designer to simplify the identification of entity instances.
- It has no meaning in the user's environment—it exists only to distinguish one entity instance from another (just
- like any other primary key).
- Can be generated by the DBMS to ensure that unique values are always provided.

Surrogate Primary Keys

- Primary key used to simplify the identification of entity instances are useful when:
  - There is no natural key
  - Selected candidate key has embedded semantic contents or is too long
- Require ensuring that the candidate key of entity in question performs properly
  - Use unique index and not null constraints

Case Study: Data Used to Keep Track of Events

What would the primary key be?

DATE	TIME_START	TIME_END	ROOM	EVENT_NAME	PARTY_OF
6/17/2016	11.00a.m.	2.00p.m.	Allure	Burton Wedding	60
6/17/2016	11.00a.m.	2.00p.m.	Bonanza	Adams Office	12
6/17/2016	3.00p.m.	5.30p.m.	Allure	Smith Family	15
6/17/2016	3.30p.m.	5.30p.m.	Bonanza	Adams Office	12
6/17/2016	1.00p.m.	3.00p.m.	Bonanza	Boy Scouts	33
6/17/2016	11.00a.m.	2.00p.m.	Allure	March of Dimes	25
6/17/2016	11.00a.m.	12.30p.m.	Bonanza	Smith Family	12

Options for the primary key

- EVENT (DATE, TIME_START, TIME_END, ROOM, EVENT_NAME, PARTY_OF)
  - (DATE, TIME_START, ROOM) or
  - (DATE, TIME_END, ROOM)
- Let’s select: (DATE, TIME_START, ROOM)

The CASE STUDY continues…

- There's also another entity – RESOURCE; to capture resources such as projectors, speakers that an event might use

- The RESOURCE entity would be represented by the following attributes:
    - RESOURCE (RSC_ID, RSC_DESCRIPTION, RSC_TYPE, RSC_QTY, RSC_PRICE)

M:N Relationship?

- Now, there's a M:N relationship between RESOURCE and EVENT
- Let's resolve this via the EVNTRSC composite entity with a composite primary key
  - EVNTRSC (DATE, TIME_START, ROOM, RSC_ID, QTY_USED)
- You now have a lengthy, four-attribute composite primary key.
- What would happen if the EVNTRSC entity's primary key were inherited by another existence-dependent entity?

CENGAGE

Problems with the Selected Primary Key

- The EVENT entity's selected primary key does not fare well
  - EVENT entity's selected primary key contains embedded semantic information and is
  - formed by a combination of date, time, and text data columns.
  - the selected primary key would cause lengthy primary keys for existence-dependent entities.
- The preferred alternative is to use a numeric, single-attribute surrogate primary key.

NEXT: Design Cases

- This section presents four special design cases that highlight the importance of flexible designs, proper identification of primary keys, and placement of foreign keys.

RECAP: Example of a 1:1 Relationship

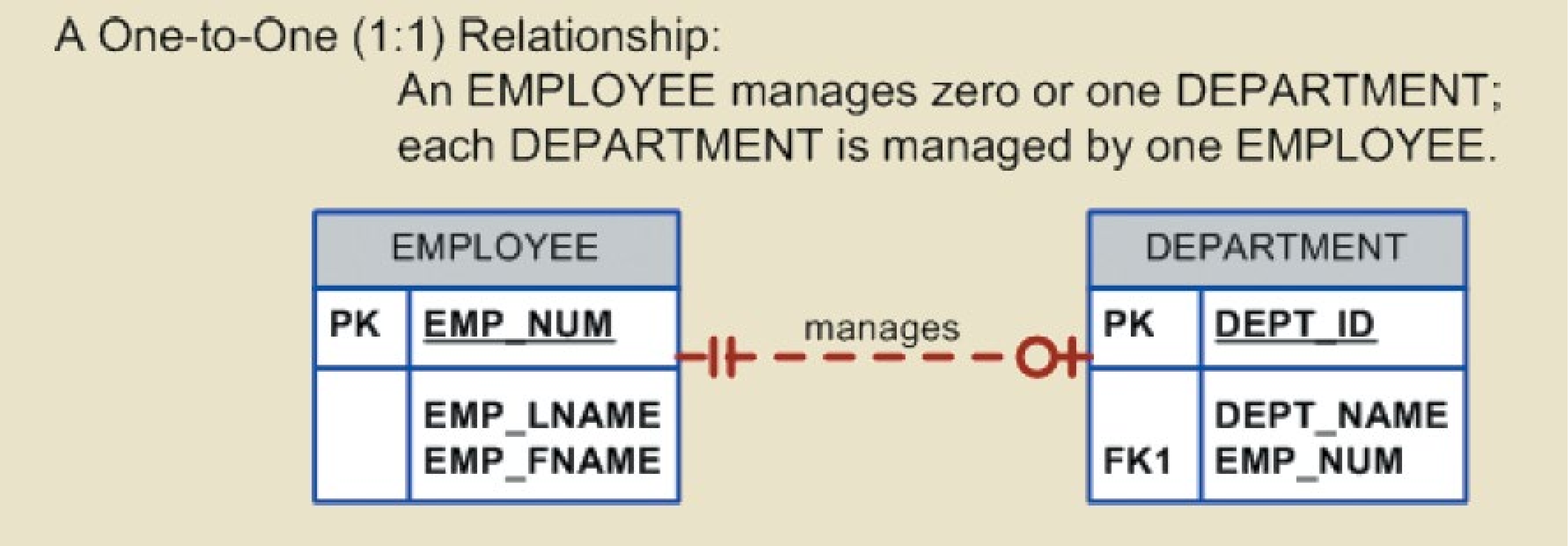
- A 1:1 relationship between EMPLOYEE and DEPARTMENT based on the business rule
  - "one EMPLOYEE is the manager of one DEPARTMENT, and one
  - DEPARTMENT is managed by one EMPLOYEE."

RECAP: Example of a 1:1 Relationship

- The 1:1 relationship is a preferred relationship in the relational data model?

- TRUE or FALSE

Figure 5.7 - The 1:1 Relationship between Department and Employee



Design Case 1: Implementing 1:1 Relationships

- Foreign keys work with primary keys to properly implement relationships in relational model
- Rule (1:M)
  - Put the primary key of the “one” side (the parent entity) on the “many” side (the dependent entity) as a foreign key.
- Options for selecting and placing the foreign key (1:1):
  - Place a foreign key in both entities
  - Place a foreign key in one of the entities

Options for selecting and placing the foreign key (1:1)

- Place a foreign key in both entities
  - Place EMP_NUM as a foreign key in DEPARTMENT, and
  - Place DEPT_ID as a foreign key in EMPLOYEE.
  - Not recommended
    - Duplicates,
    - Can cause conflict with other existing relationships.
    - E.g. DEPARTMENT and EMPLOYEE also participate in a 1:M relationship—one department employs many employees.

Options for selecting and placing the foreign key (1:1)

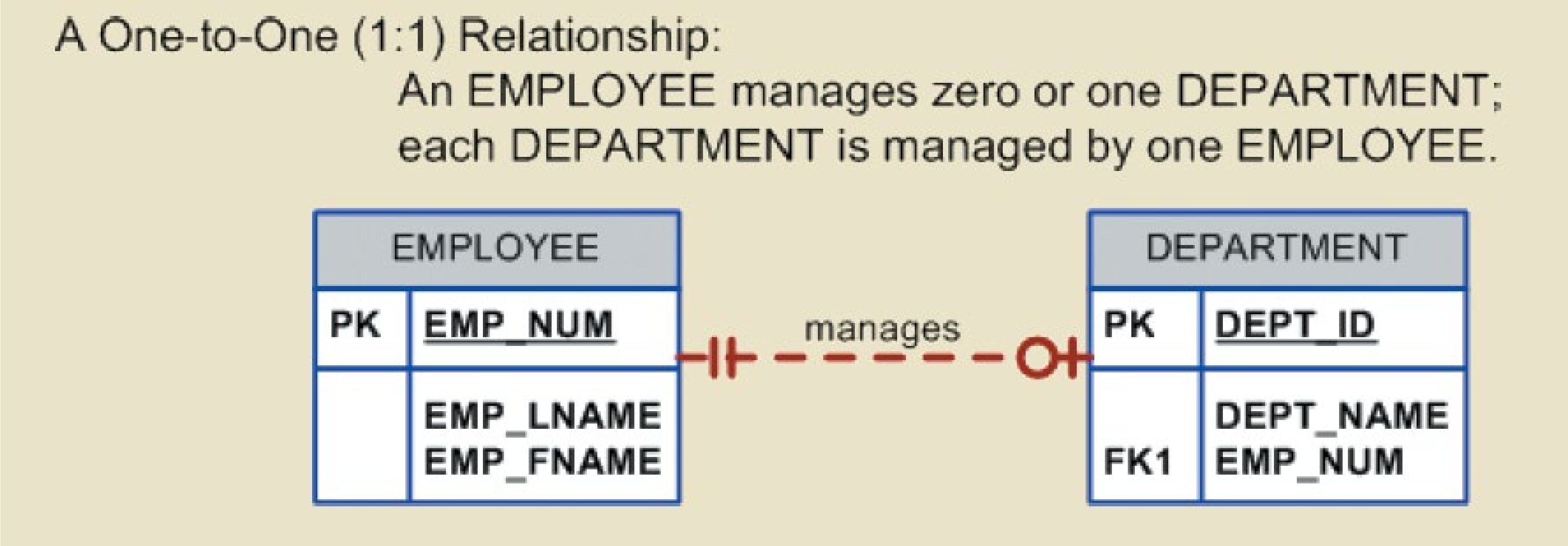
- Place a foreign key in one of the entities
  - the primary key of one of the two entities appears as a foreign key in the other entity.
  - The preferred solution

- Which primary key should be used as a foreign key?

Table 5.5 - Selection of Foreign Key in a 1:1 Relationship

CASE	ER RELATIONSHIP CONSTRAINTS	ACTION
I	One side is mandatory and the other side is optional.	Place the PK of the entity on the mandatory side in the entity on the optional side as a FK, and make the FK mandatory.
II	Both sides are optional.	Select the FK that causes the fewest nulls, or place the FK in the entity in which the (relationship) role is played.
III	Both sides are mandatory.	See Case II, or consider revising your model to ensure that the two entities do not belong together in a single entity.

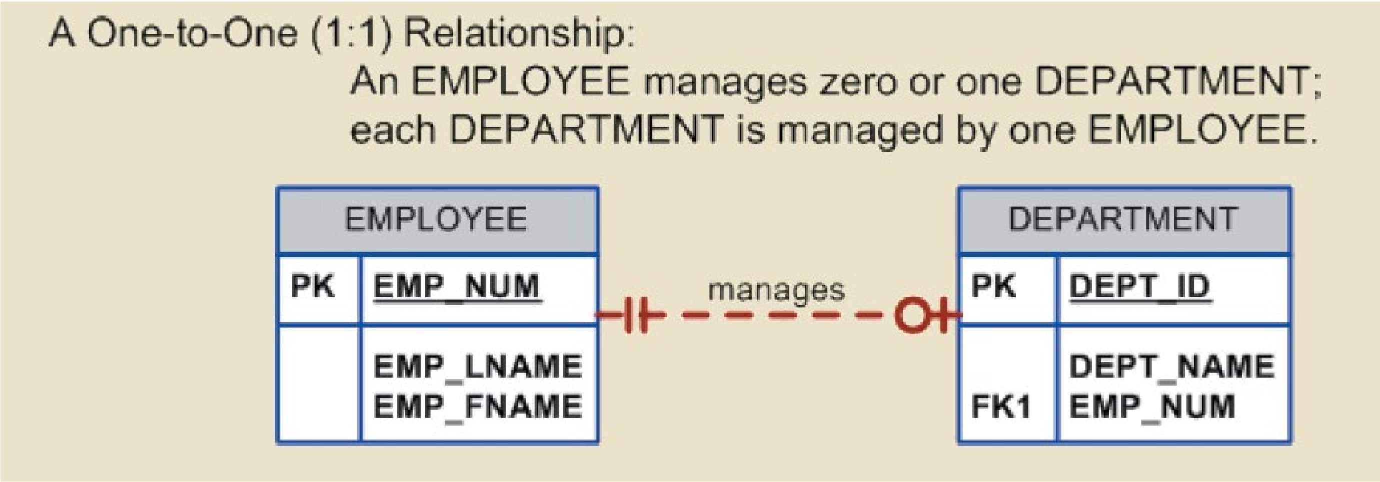
Figure 5.7 - The 1:1 Relationship between Department and Employee



EMP_NUM is placed as the foreign key in DEPARTMENT.

Why?

Figure 5.7 - The 1:1 Relationship between Department and Employee



Two Reasons:

- **EMPLOYEE is mandatory to DEPARTMENT.**
- **“manager” role is played by the EMPLOYEE in the DEPARTMENT.**

Design Case 2: Maintaining History of Time-Variant Data

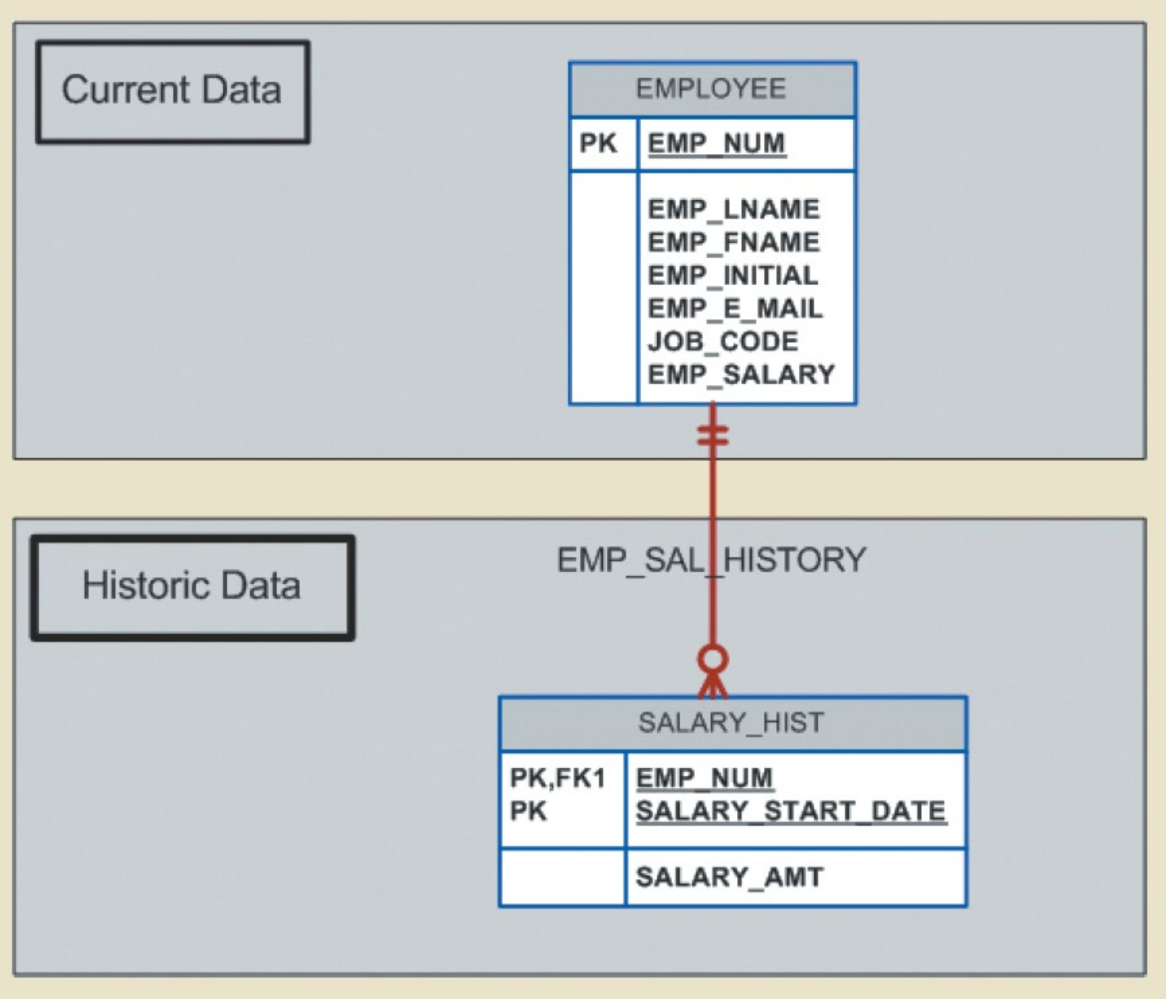
- **Time-variant data**: Data whose values change over time and for which a history of the data changes must be retained

- The storage of time-variant data requires changes in the data model;
  - the type of change depends on the nature of the data
- Ex:
  - an employee's salary over the years
  - a department's manager over the years

Design Case 2: Maintaining History of Time-Variant Data

- Some time-variant data is equivalent to having a multivalued attribute in your entity.
  - An employee's salary over the years (multivalued-attribute);
  - Requires creating a new entity in a 1:M relationship with the original entity
  - New entity contains the new value, date of the change, and other pertinent attribute

Figure 5.8 - Maintaining Salary History



Design Case 2: Maintaining History of Time-Variant Data

- Other time-variant data can turn a 1:M relationship into an M:N relationship.
  - Each department is managed by only one employee
  - Each employee can manage one department at most
    - 1:1 relationship exist between EMPLOYEE and DEPARTMENT.
  - If you want to keep track of the history of all department managers as well as the current manager?

Figure 5.9 - Maintaining Manager History

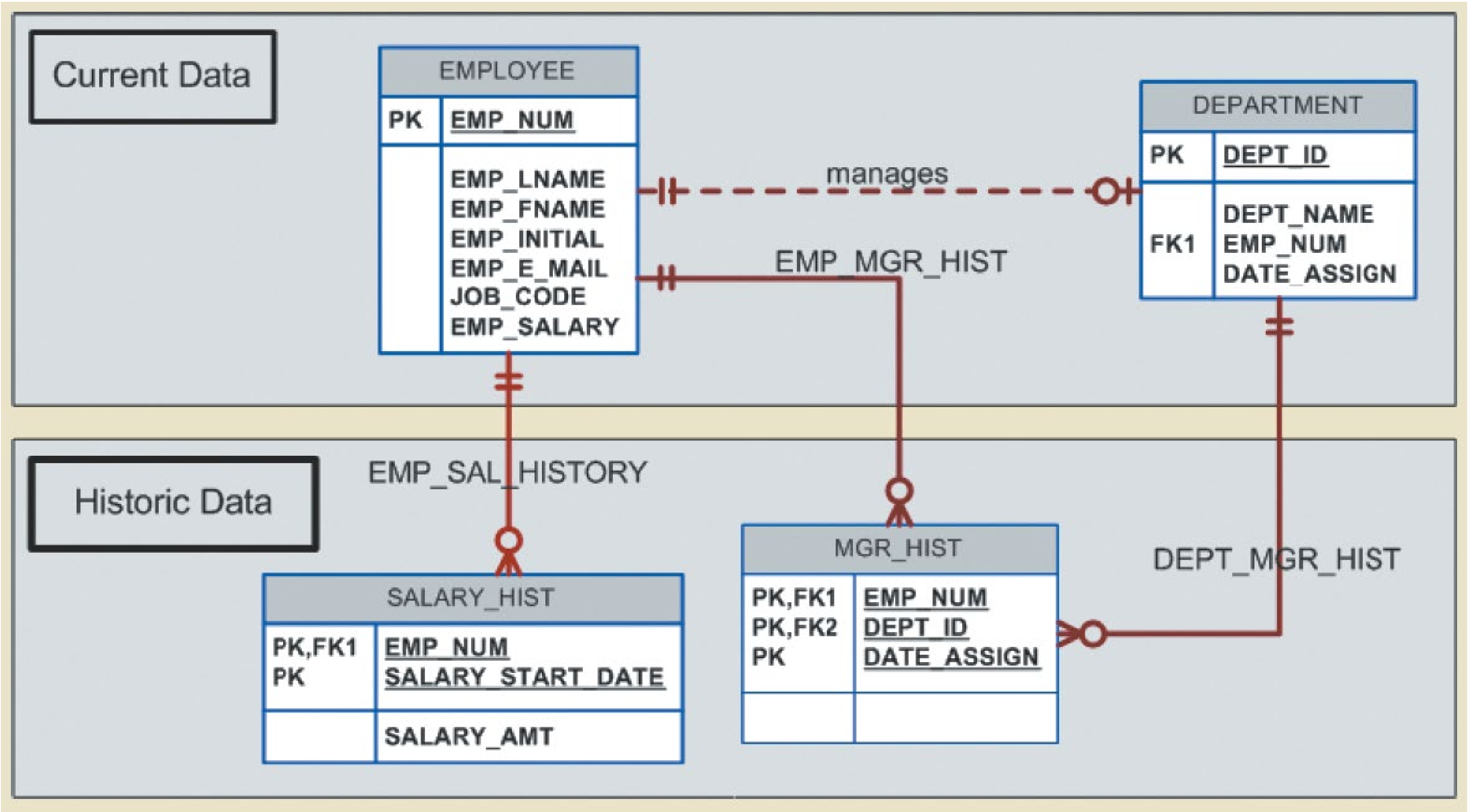
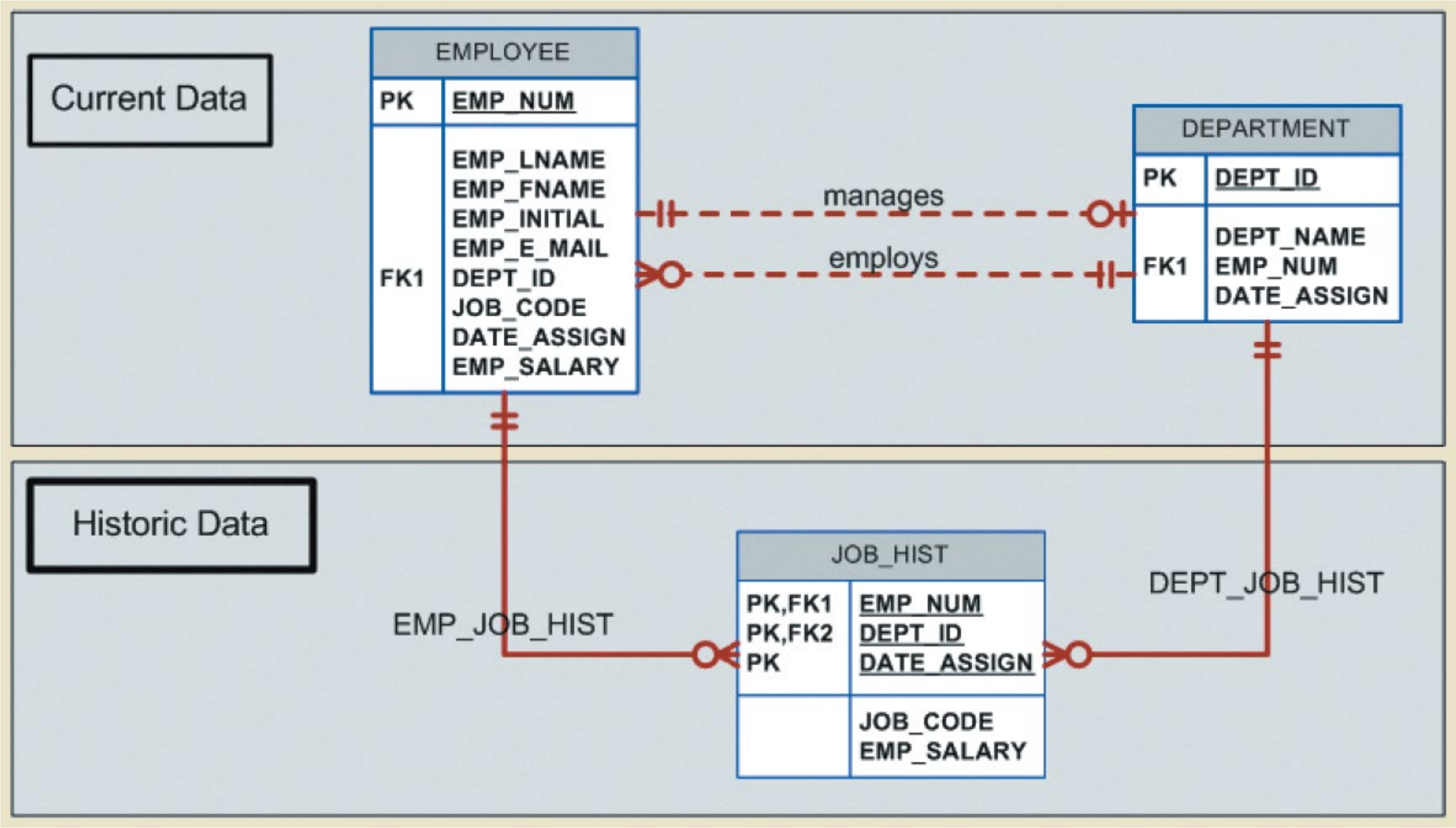


Figure 5.10 - Maintaining Job History



Design Case 3: Fan Traps

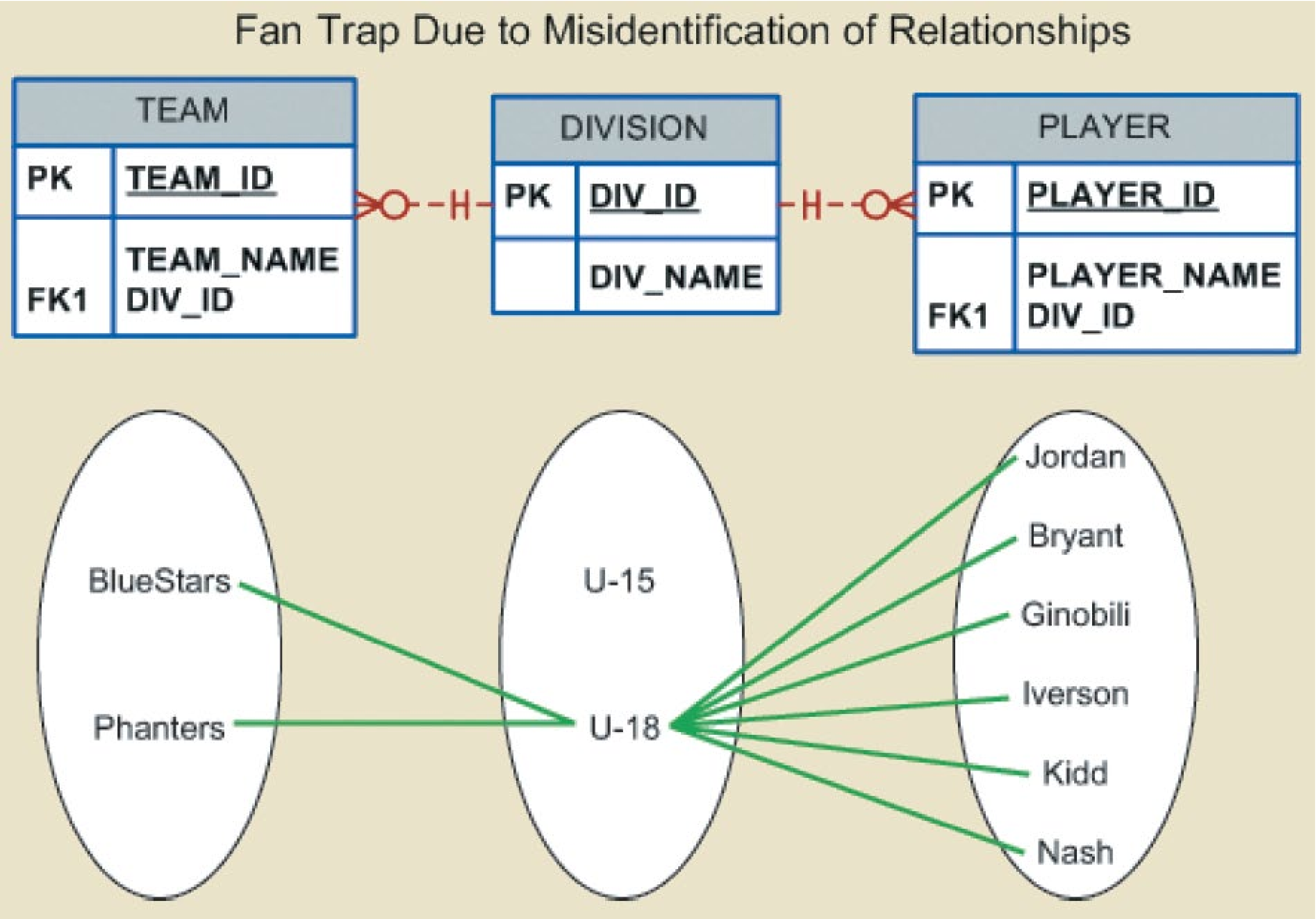
- **Design trap**: Occurs when a relationship is improperly or incompletely identified
  - Represented in a way not consistent with the real world

- **Fan trap**:
  - The most common type of design trap
  - Occurs when one entity is in two 1:M relationships to other entities
  - Produces an association among other entities not expressed in the model

Example

- A basketball league has many divisions.
- Each division has many players,
- Each division has many teams

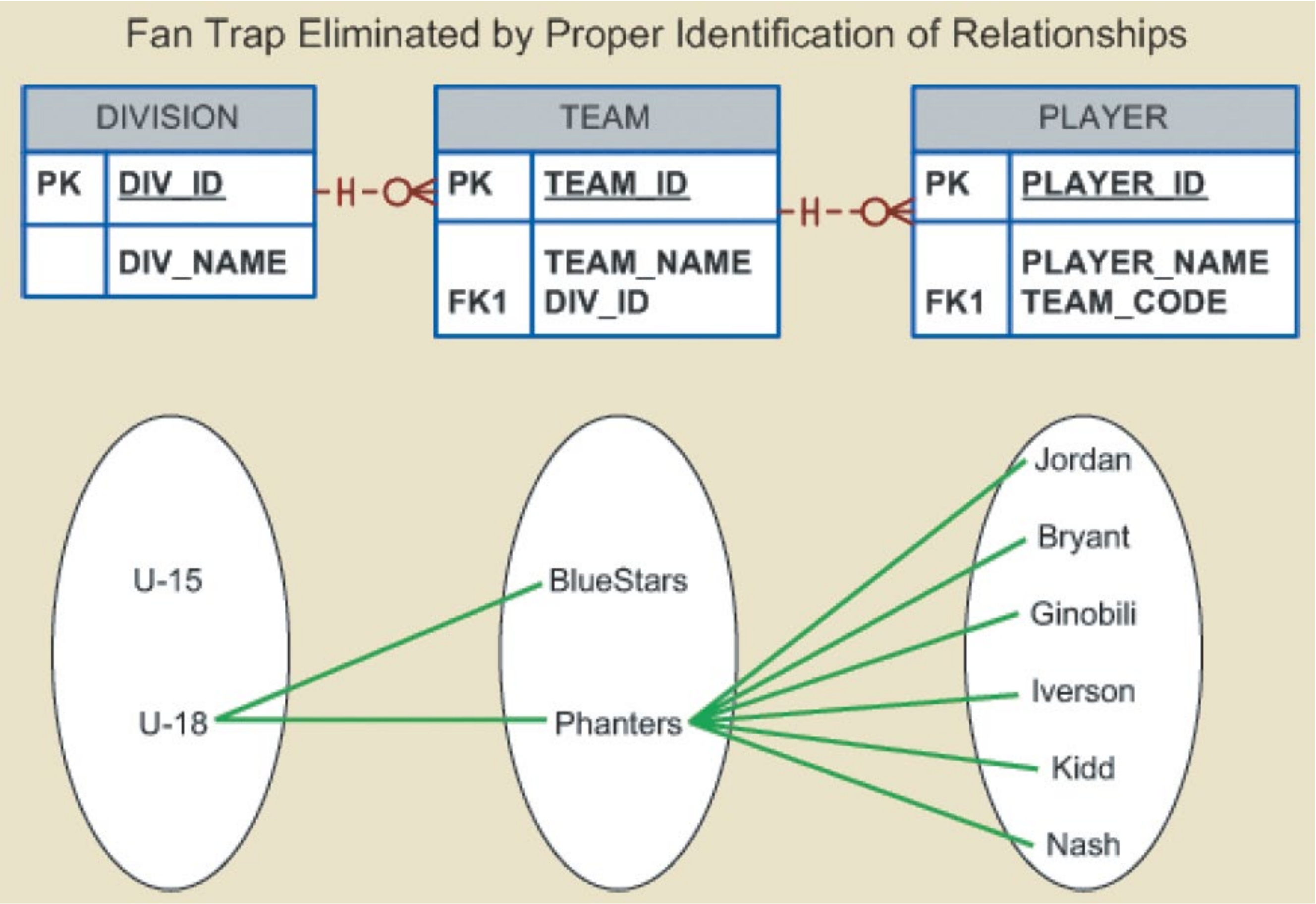
Figure 5.11 - Incorrect ERD with Fan Trap Problem



Although that representation is semantically correct, the relationships are not properly identified.

How do you identify which players belong to which team?

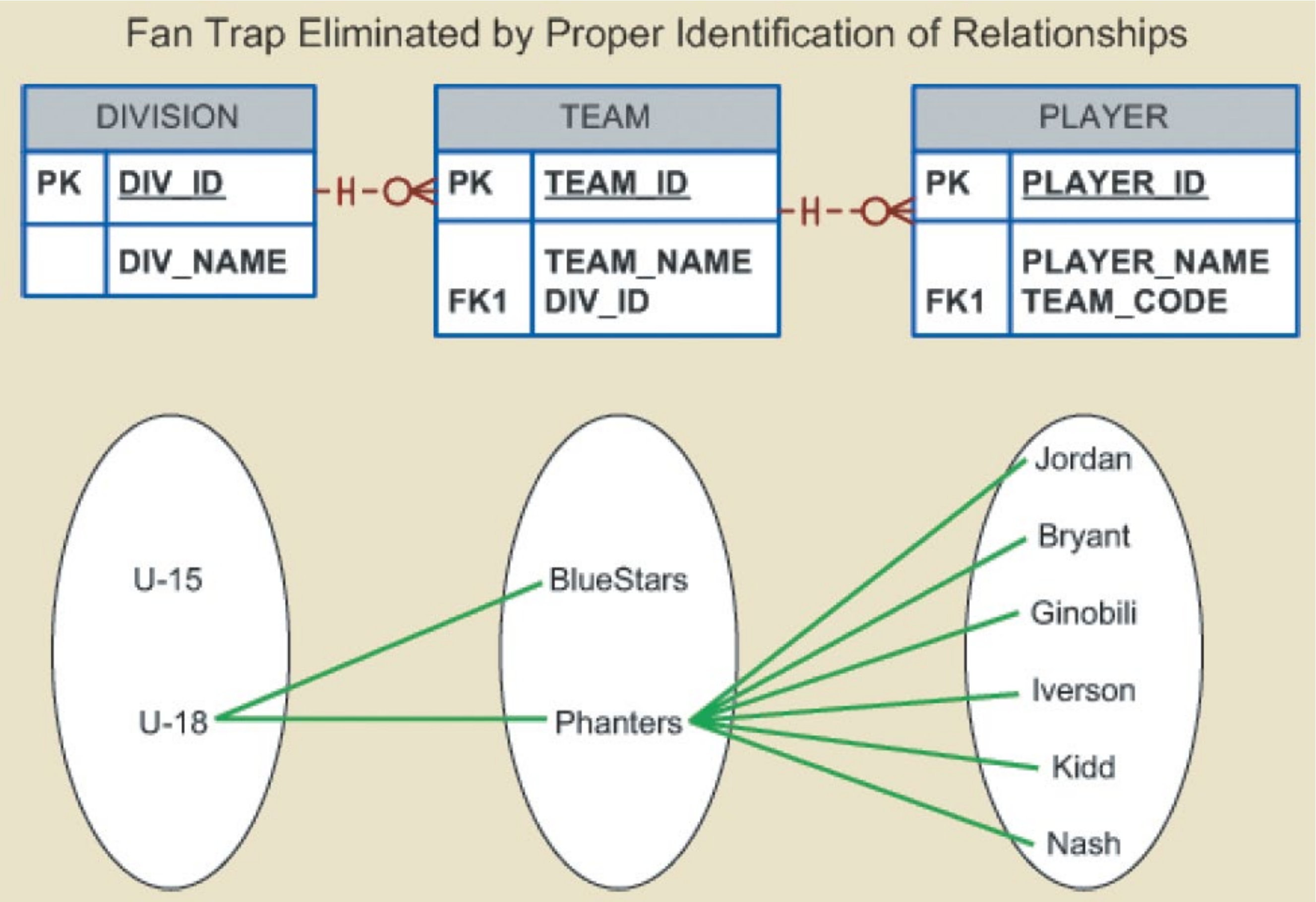
Figure 5.12 - Corrected ERD After Removal of the Fan Trap



Design Case 4: Redundant Relationships

- Occur when there are multiple relationship paths between related entities
- Need to remain consistent across the model
- Help simplify the design

Figure 5.12 - Corrected ERD After Removal of the Fan Trap



Which division does a player belong to ?

Figure 5.13 - A Redundant Relationship