

Chapter 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
- Diagram using this model is called **EER diagram (EERD)**

Entity Supertypes and Subtypes

- **Entity supertype:**
 - Generic entity type related to one or more entity subtypes
 - Contains common characteristics (e.g. Employee ID, Name, DOB)
- **Entity subtype:**
 - Contains unique characteristics of each entity subtype
- (Insert picture to show high-level concept)

**FIGURE
5.1**

Nulls created by unique attributes

EMP_NUM	EMP_LNAME	EMP_FNAME	EMP_INITIAL	EMP_LICENSE	EMP_RATINGS	EMP_MED_TYPE	EMP_HIRE_DATE
100	Kolmycz	Xavier	T				15-Mar-88
101	Lewis	Marcos		ATP	SEL/MEL/Instr/CFII	1	25-Apr-89
102	Vandam	Jean					20-Dec-93
103	Jones	Victoria	R				28-Aug-03
104	Lange	Edith		ATP	SEL/MEL/Instr	1	20-Oct-97
105	Williams	Gabriel	U	COM	SEL/MEL/Instr/CFI	2	08-Nov-97
106	Duzak	Mario		COM	SEL/MEL/Instr	2	05-Jan-04
107	Diante	Venite	L				02-Jul-97
108	Wiesenbach	Joni					18-Nov-95
109	Travis	Brett	T	COM	SEL/MEL/SES/Instr/CFII	1	14-Apr-01
110	Genkazi	Stan					01-Dec-03

Trying to put “too much” into a single table results in many NULL values
Not every employee will be “licensed” (EMP_LICENSE)

Entity Supertypes and Subtypes

- Criteria to decide when to use this:
 - 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
- For example, consider the different types of employees for an Airline

Employee Type	Unique Data
MECHANIC	Annual maintenance training
ACCOUNTANT	CPA-specific information
PILOT	Pilot's license, Date Acquired, Rating

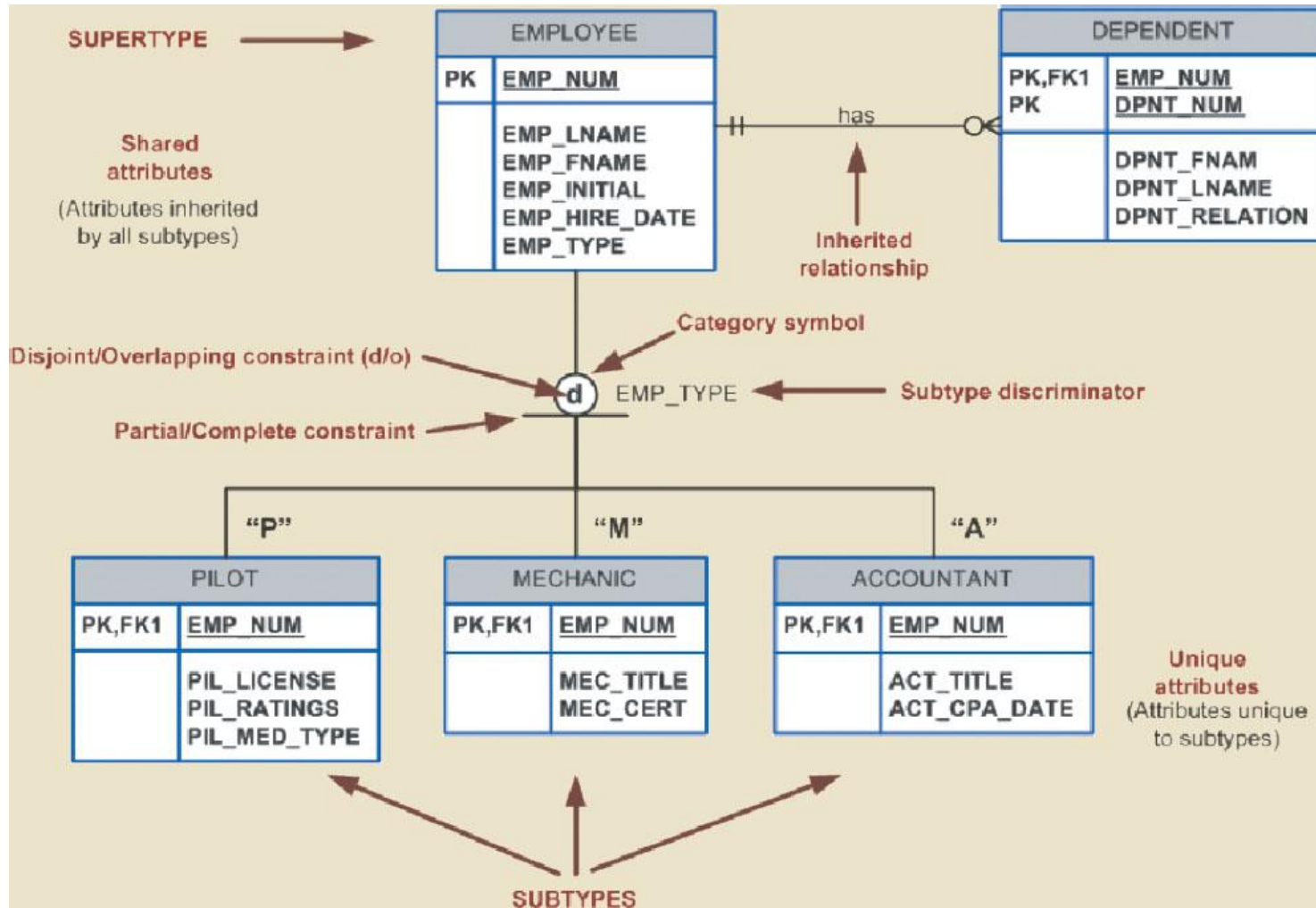
Specialization Hierarchy (1 of 2)

- Depicts arrangement of:
higher-level entity supertypes + lower-level entity subtypes
- Relationships are described in terms of “is-a” relationships
- Subtype exists ONLY within the context of a supertype
- Every subtype has ONLY one supertype to which it is directly related
- Can have many levels of supertype/subtype relationships

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



- What's the Supertype?
- What are the Subtypes?
- What attributes are shared?
- Which attributes are specific to the subtype?

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

Airline example

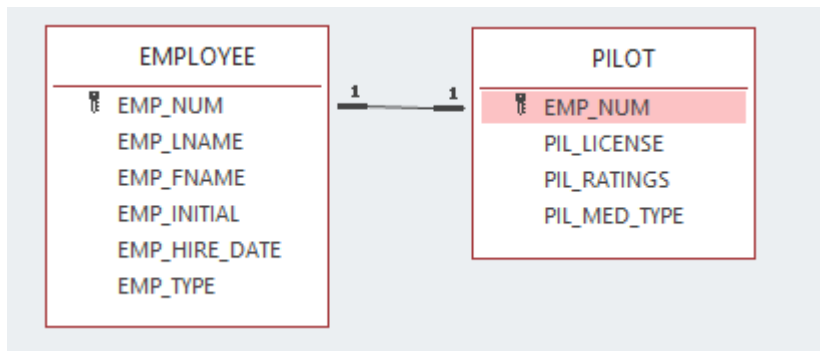
FIGURE 5.3 The EMPLOYEE-PILOT supertype-subtype relationship

Table Name: EMPLOYEE

EMP_NUM	EMP_LNAME	EMP_FNAME	EMP_INITIAL	EMP_HIRE_DATE	EMP_TYPE
100	Kolmycz	Xavier	T	15-Mar-88	
101	Lewis	Marcos		25-Apr-89	P
102	Vandam	Jean		20-Dec-93	A
103	Jones	Victoria	R	28-Aug-03	
104	Lange	Edith		20-Oct-97	P
105	Williams	Gabriel	U	08-Nov-97	P
106	Duzak	Mario		05-Jan-04	P
107	Diante	Verite	L	02-Jul-97	M
108	Miesenbach	Joni		18-Nov-95	M
109	Travis	Brett	T	14-Apr-01	P
110	Genkazi	Stan		01-Dec-03	A

Table Name: PILOT

EMP_NUM	PIL_LICENSE	PIL_RATINGS	PIL_MED_TYPE
101	ATP	SEL/MEL/Instr/CFII	1
104	ATP	SEL/MEL/Instr	1
105	COM	SEL/MEL/Instr/CFI	2
106	COM	SEL/MEL/Instr	2
109	COM	SEL/MEL/SES/Instr/CFII	1



Subtype Discriminator

- Attribute in the supertype entity
 - determines to which entity subtype the supertype occurrence is related
 - e.g. EMP_TYPE: “P” for PILOT, “M” for MECHANIC, “A” for ACCOUNTANT
- Default comparison condition is the equality comparison
 - e.g. if EMP_TYPE = “P” then subtype = PILOT
- Other comparison conditions possible

Disjoint and Overlapping Constraints

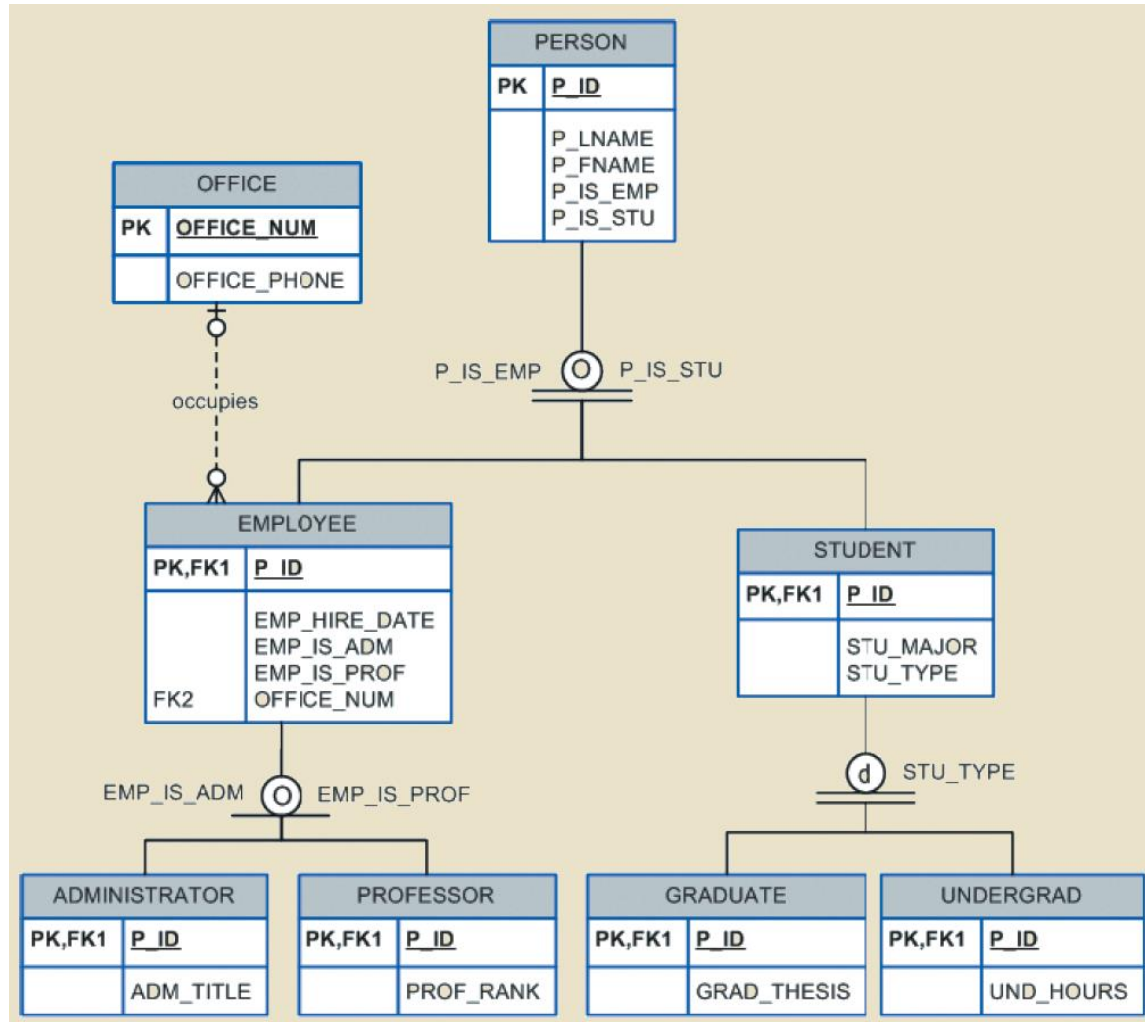
- **Disjoint subtypes:**

- Also known as **nonoverlapping subtypes**
- Implementation is based on the value of the subtype discriminator attribute in the supertype
- E.g. a PILOT cannot also be a MECHANIC. **No Overlap!**

- **Overlapping subtypes:**

- Implementation requires the use of one discriminator attribute for each subtype
 - PROFESSOR (Y/N) + ADMINISTRATOR (Y/N)

Figure 5.4 - Specialization Hierarchy with Overlapping Subtypes



Notice how the PK (P_ID) is consistent
Notice the DISCRIMINATOR type
(d = Disjoint, O = Overlapping)

For example:

- A Person can be an employee **AND/OR** a Student
- But a Student can either be a Graduate **OR** Undergraduate (NOT BOTH)
- An Employee can be a Administrator **AND/OR** a Professor



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
- **Partial completeness:**
 - Symbolized by a circle over a single line
 - Not every supertype occurrence is a member of a subtype
- **Total completeness:**
 - Symbolized by a circle over a double line
 - Every supertype occurrence must be a member of any

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. (Example 1)	Supertype has optional subtypes. Subtype discriminators can be null. Subtype sets are not unique. (Example 3)
Total 	Every supertype occurrence is a member of only one subtype. Subtype discriminator cannot be null. Subtype sets are unique. (Example 2)	Every supertype occurrence is a member of at least one subtype. Subtype discriminators cannot be null. Subtype sets are not unique. (Example 4)

Example 1: An airline employee might just work in IT, and not be a PILOT, ACCOUNTANT, or MECHANIC

Example 2: An airline employee MUST be one of the above (or the DBA would need to create subtype Tables for every possible employee type)

Example 3: A person COULD be neither a STUDENT nor a PROFESSOR (contractor?)

Example 4: A person MUST be either a STUDENT or a PROFESSOR

Specialization and Generalization

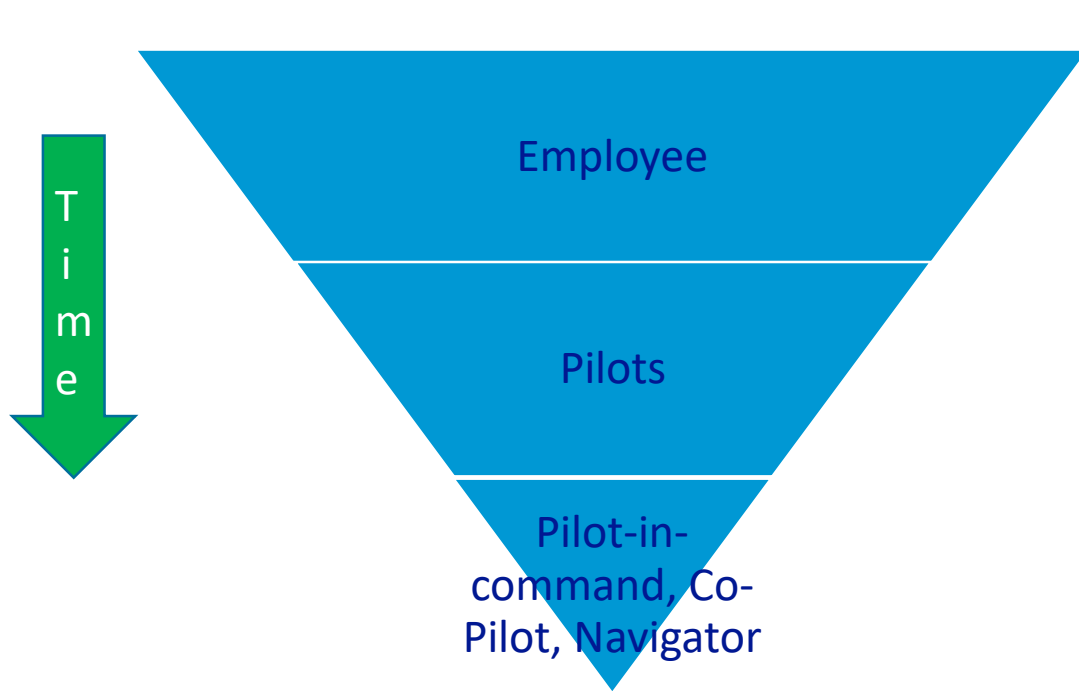
Specialization

- 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

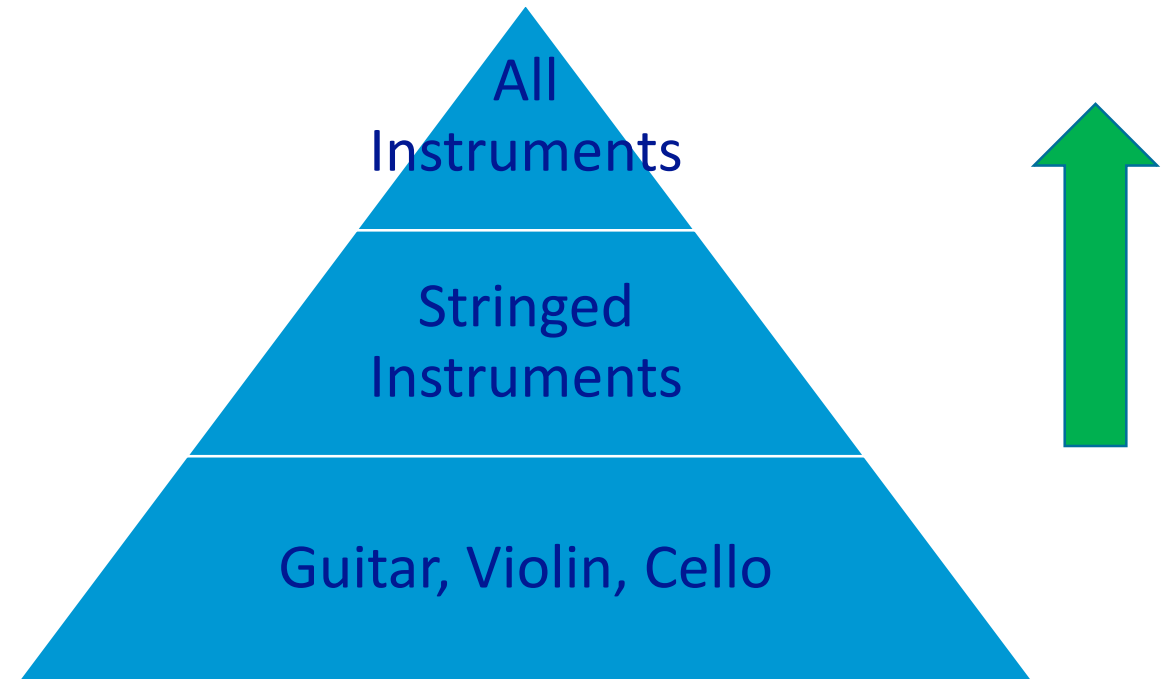
- Bottom-up process
- Identifies a higher-level, more generic entity supertype from lower-level entity subtypes
- Based on grouping common characteristics and relationships of the subtypes

Specialization and Generalization



Specialization:

- Start big, then identify 1st level subtypes, then (if any) 2nd level subtypes



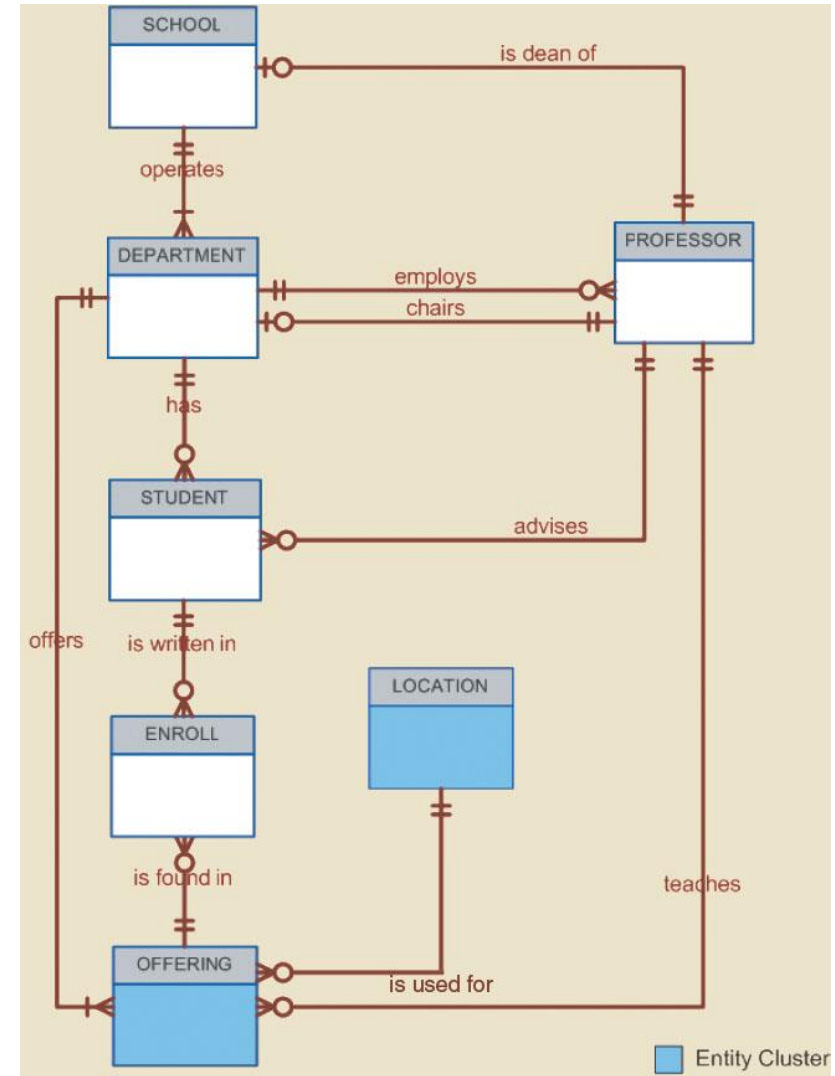
Generalization

- Start small, then figure out the biggest parts

Entity Cluster

- Virtual entity type used to represent multiple entities and relationships in ERD
- Avoid the display of attributes to eliminate complications that result when the inheritance rules change
- Helps when you're trying to look at or explain the "big picture"

Figure 5.5 - Tiny College ERD Using Entity Clusters



Primary Keys

- Single attribute or a combination of attributes, which uniquely identifies each entity instance
 - Guarantees entity integrity
 - Primary keys and foreign keys work together to implement relationships
 - Properly selecting primary key has direct bearing on efficiency and effectiveness

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

Desirable Primary Key Characteristics

- Non intelligent
- No change over time
- Preferably single-attribute
- Preferably numeric (e.g. AutoNumber)
- Security-compliant
 - e.g. no PII/PHI like Social Security Number

TABLE
5.3

Desirable Primary Key Characteristics

PK CHARACTERISTIC	RATIONALE
Unique values	The PK must uniquely identify each entity instance. A primary key must be able to guarantee unique values. It cannot contain nulls.
Nonintelligent	The PK should not have embedded semantic meaning other than to uniquely identify each entity instance. An attribute with embedded semantic meaning is probably better used as a descriptive characteristic of the entity than as an identifier. For example, a student ID of 650973 would be preferred over Smith, Martha L. as a primary key identifier.
No change over time	If an attribute has semantic meaning, it might be subject to updates. This is why names do not make good primary keys. If you have Vickie Smith as the primary key, what happens if she changes her name when she gets married? If a primary key is subject to change, the foreign key values must be updated, thus adding to the database work load. Furthermore, changing a primary key value means that you are basically changing the identity of an entity. In short, the PK should be permanent and unchangeable.
Preferably single-attribute	A primary key should have the minimum number of attributes possible (irreducible). Single-attribute primary keys are desirable but not required. Single-attribute primary keys simplify the implementation of foreign keys. Having multiple-attribute primary keys can cause primary keys of related entities to grow through the possible addition of many attributes, thus adding to the database work load and making (application) coding more cumbersome.
Preferably numeric	Unique values can be better managed when they are numeric, because the database can use internal routines to implement a counter-style attribute that automatically increments values with the addition of each new row. In fact, most database systems include the ability to use special constructs, such as Autonumber in Microsoft Access, to support self-incrementing primary key attributes.
Security-compliant	The selected primary key must not be composed of any attribute(s) that might be considered a security risk or violation. For example, using a Social Security number as a PK in an EMPLOYEE table is not a good idea.

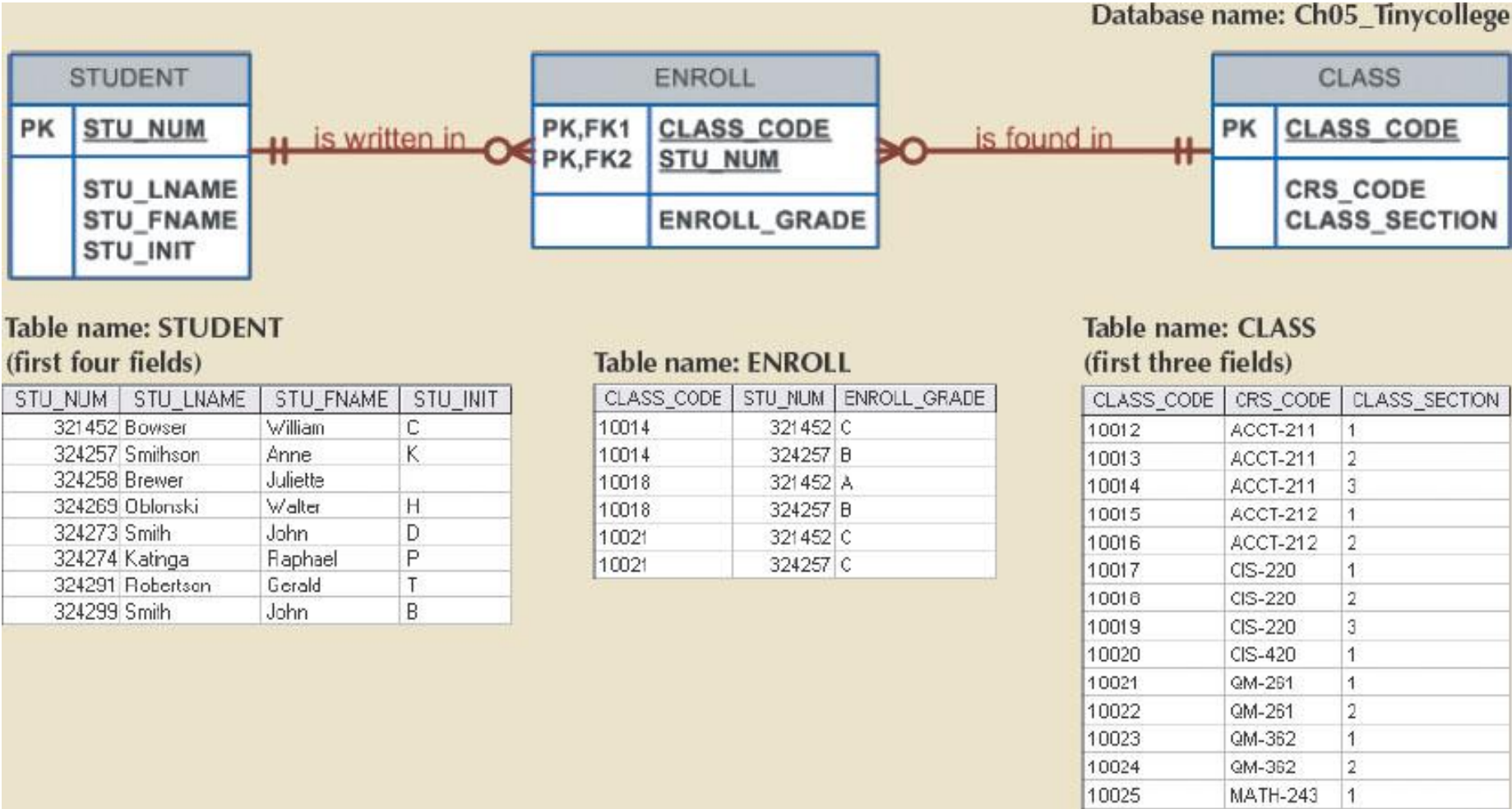
Use of Composite Primary Keys (1 of 2)

- Composite primary keys are useful in two cases:
 - As identifiers of composite entities
 - In which each primary key combination is allowed once in M:N relationship
 - As identifiers of weak entities
 - In which weak entity has a strong identifying relationship with the parent entity
 - Automatically provides benefit of ensuring that there cannot be duplicate values

Use of Composite Primary Keys (2 of 2)

- When used as identifiers of weak entities normally used to represent
 - Real-world object that is existence-dependent on another real-world object
 - Real-world object that is represented in the data model as two separate entities in a strong identifying relationship
- Dependent entity only exists when it is related to the parent entity

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



Surrogate Primary Keys

- Primary key used to simplify the identification of entity instances
- Design decision by Database Modeler and team
- Useful when:
 - There is no natural key
 - Selected candidate key has embedded semantic contents
 - Selected candidate key is too long or cumbersome
- If you use a Surrogate Key:
 - Be aware of the “trade-off”
 - You must enforce unique index and not null constraints
 - You must ensure the Candidate key can be Unique and Not Null

Table 5.4 - Data Used to Keep Track of Events

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

What Primary Key would you create?

A Compound Key?

A unique Identifier?

Design Case 1: Implementing 1:1 Relationships

- Foreign keys work with primary keys to properly implement relationships in relational model
- Rule
 - Put primary key of the parent entity on the dependent entity as a foreign key
- Options for selecting and placing the foreign key:
 - Place a foreign key in both entities
 - Place a foreign key in one of the entities

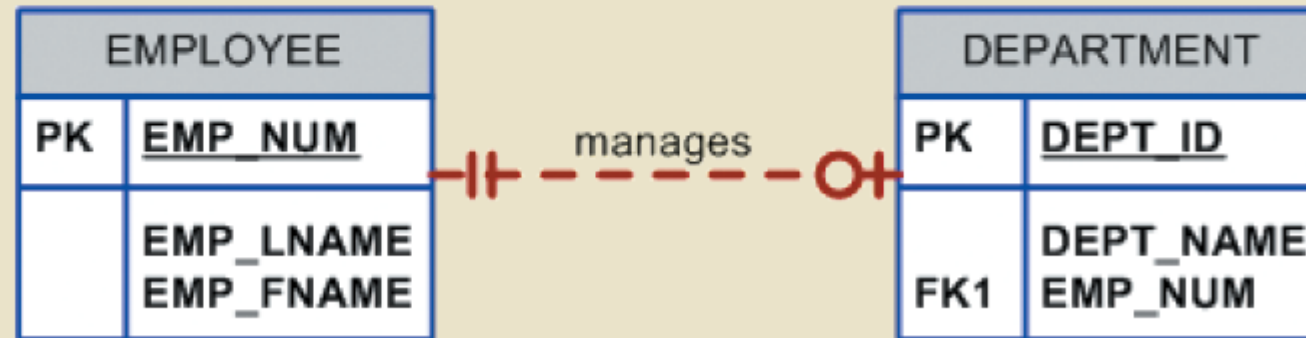
Table 5.5 - Selection of Foreign Key in a 1:1 Relationship

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

A One-to-One (1:1) Relationship:

An EMPLOYEE manages zero or one DEPARTMENT;
each DEPARTMENT is managed by one EMPLOYEE.



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
 - 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

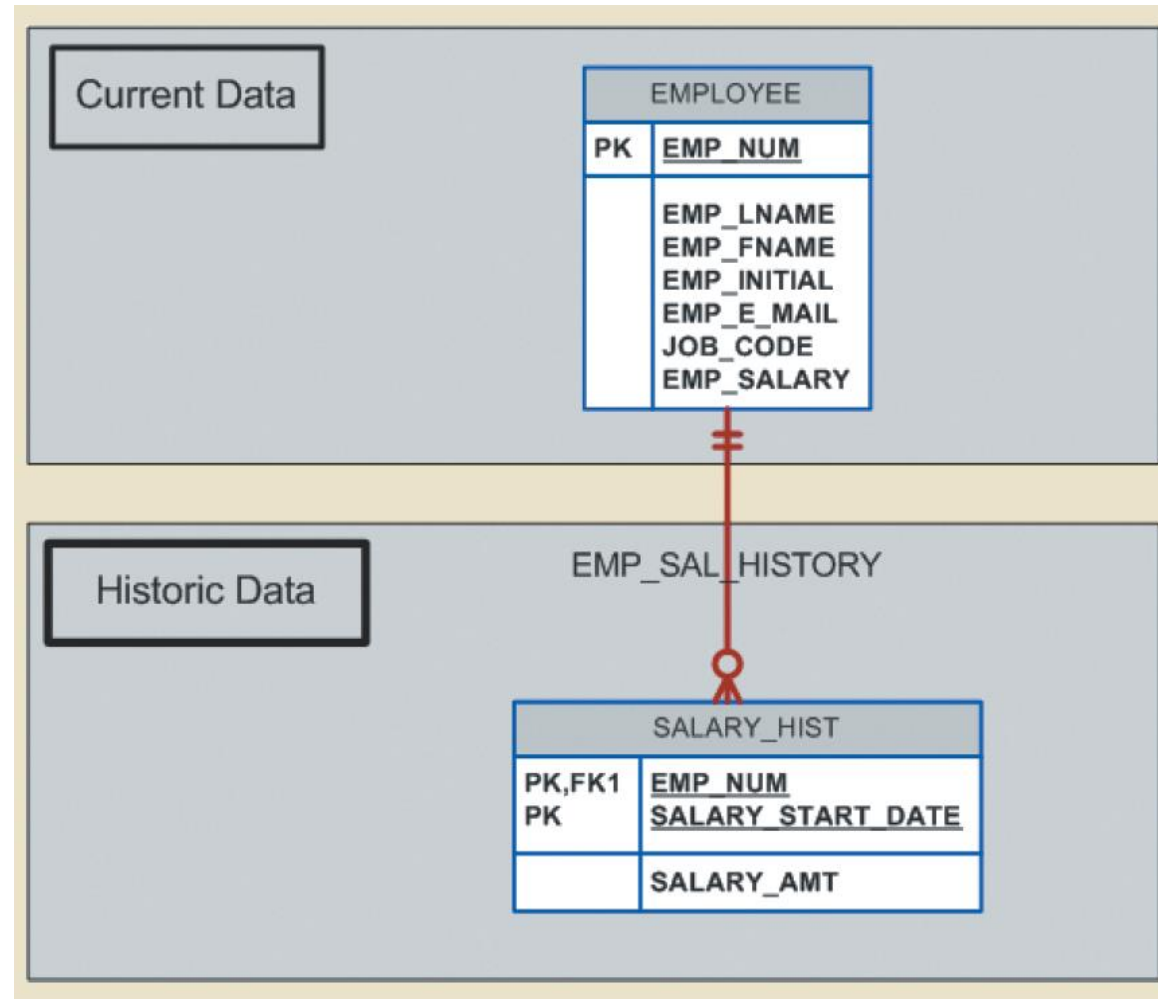


Figure 5.9 - Maintaining Manager History

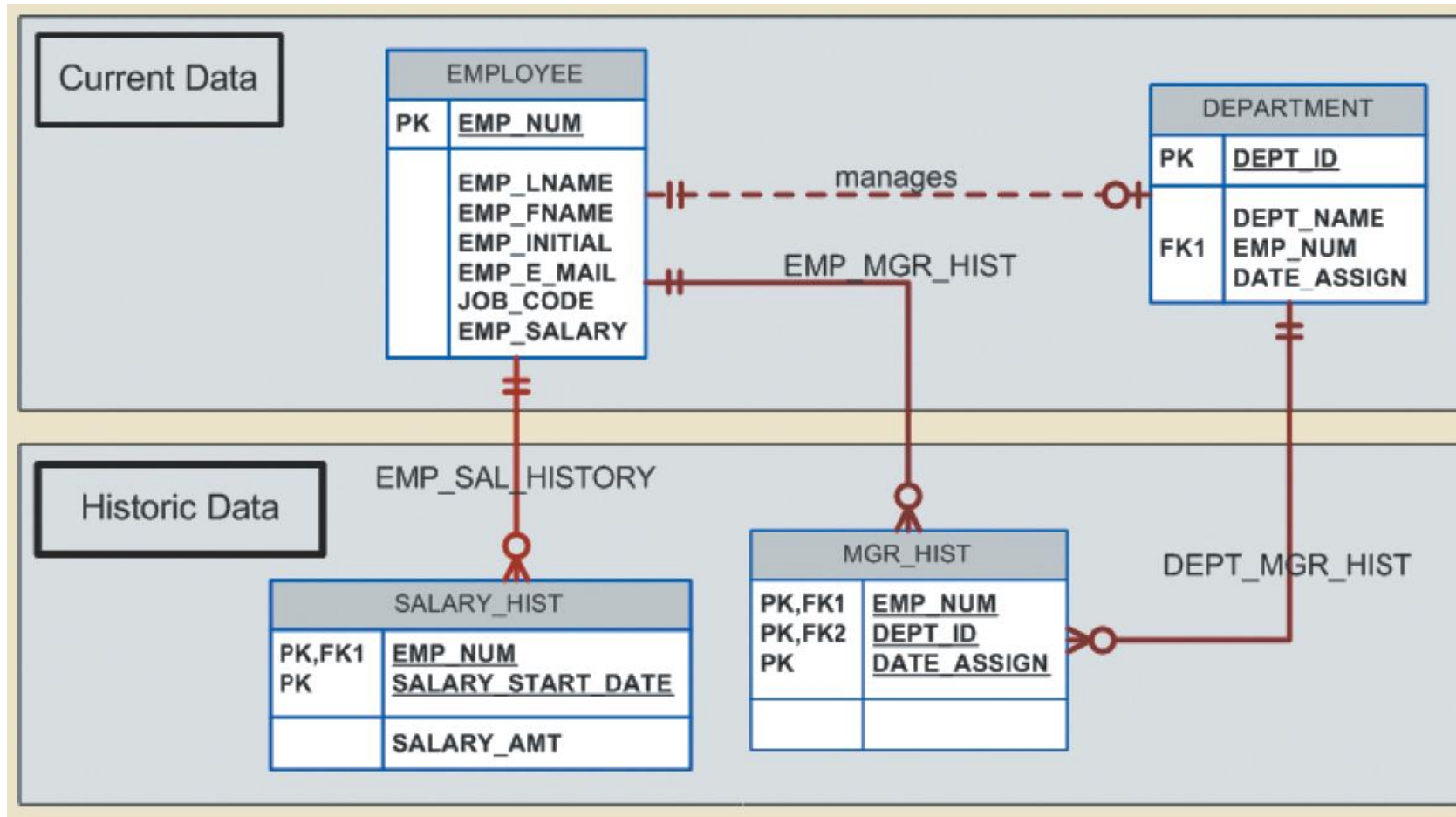
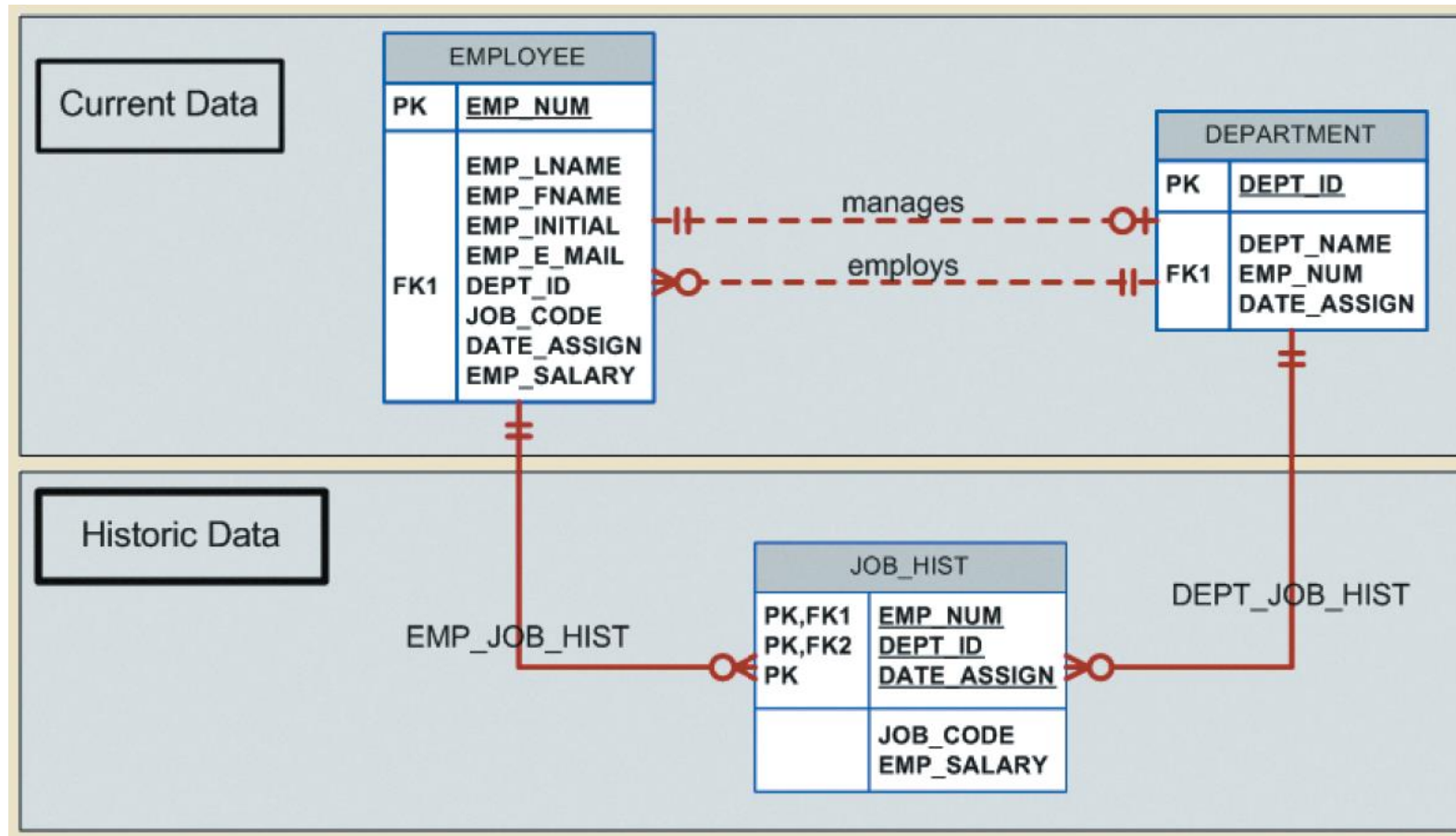


Figure 5.10 - Maintaining Job History



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:** Occurs when one entity is in two 1:M relationships to other entities
 - Produces an association among other entities not expressed in the model

Figure 5.11 - Incorrect ERD with Fan Trap Problem

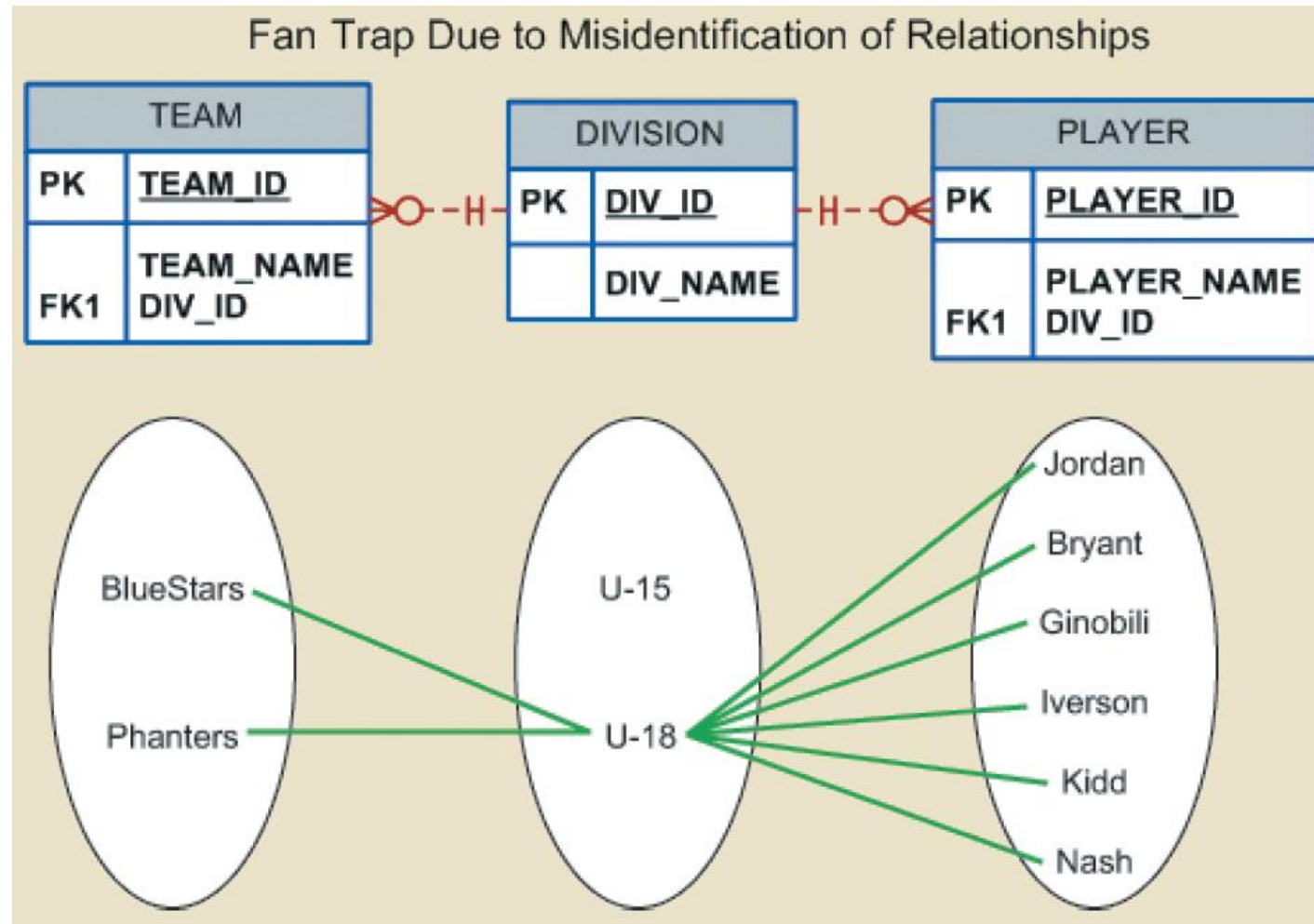
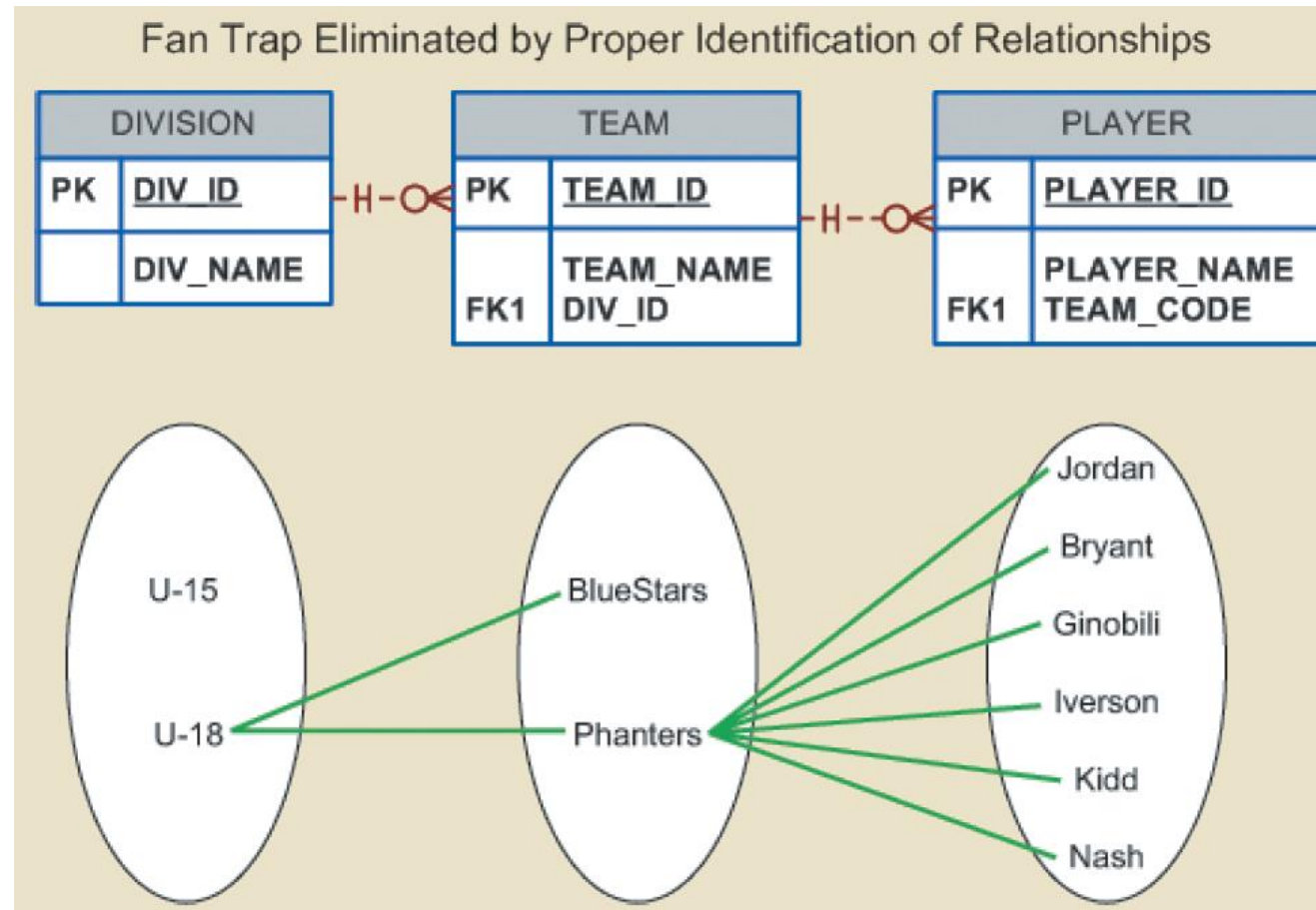


Figure 5.12 - Corrected ERD After Removal of the Fan Trap



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 5.13 - A Redundant Relationship

