DBMD UNIT - 1 and 2 Notes

UNIT-I: CONCEPTUAL DATA MODELLING

1. Introduction

- **Overview of Database Systems Architecture and Components [PYQ Q1a, Q2a]**
  - **Data, Information, Metadata:**
    - **Data:** Raw, unorganized facts (e.g., "1713445232"). (p.5)
    - **Information:** Data processed to reveal meaning in a specific context (e.g., "1713445232" is a phone number). (p.5)
    - **Metadata:** Data that describes the properties of data (data about data). Defines structure, data types, constraints. (p.6) *(Table 1.1, p.6)*

Table 1.1 Some metadata for a manufacturing plant

Record Type	Data Element	Data Type	Size	Source	Role	Domain
PLANT	Pl_name	Alphabetic	30	Stored	Non-key	
PLANT	Pl_number	Numeric	2	Stored	Key	Integer values from 10 to 20
PLANT	Budget	Numeric	7	Stored	Non-key	
PLANT	Building	Alphabetic	20	Stored	Non-key	
PLANT	No_of_employees	Numeric	4	Derived	Non-key	

  - **Data Management:** Involves creation, retrieval, modification, and deletion of data. Requires data access and organization. (p.6-7)
    - Access Methods: Sequential, Direct.
    - Organization: Sequential, Random (hashing), Indexed.
  - **Limitations of File-Processing Systems [PYQ - Implied in understanding DB advantages]** (p.7-9)
    - Lack of data integrity: Inconsistent, incorrect, incomplete data due to duplication.
    - Lack of standards: Difficulty in enforcing naming, access, and protection standards.
    - Lack of flexibility/maintainability: Changes require significant programming effort.
    - **Root Causes:**
      - Lack of data integration: Data is separated and isolated.
      - Lack of program-data independence: File structure is embedded in application programs.

- **ANSI/SPARC Three-Schema Architecture [PYQ - Important for data independence]** (p.9-11)

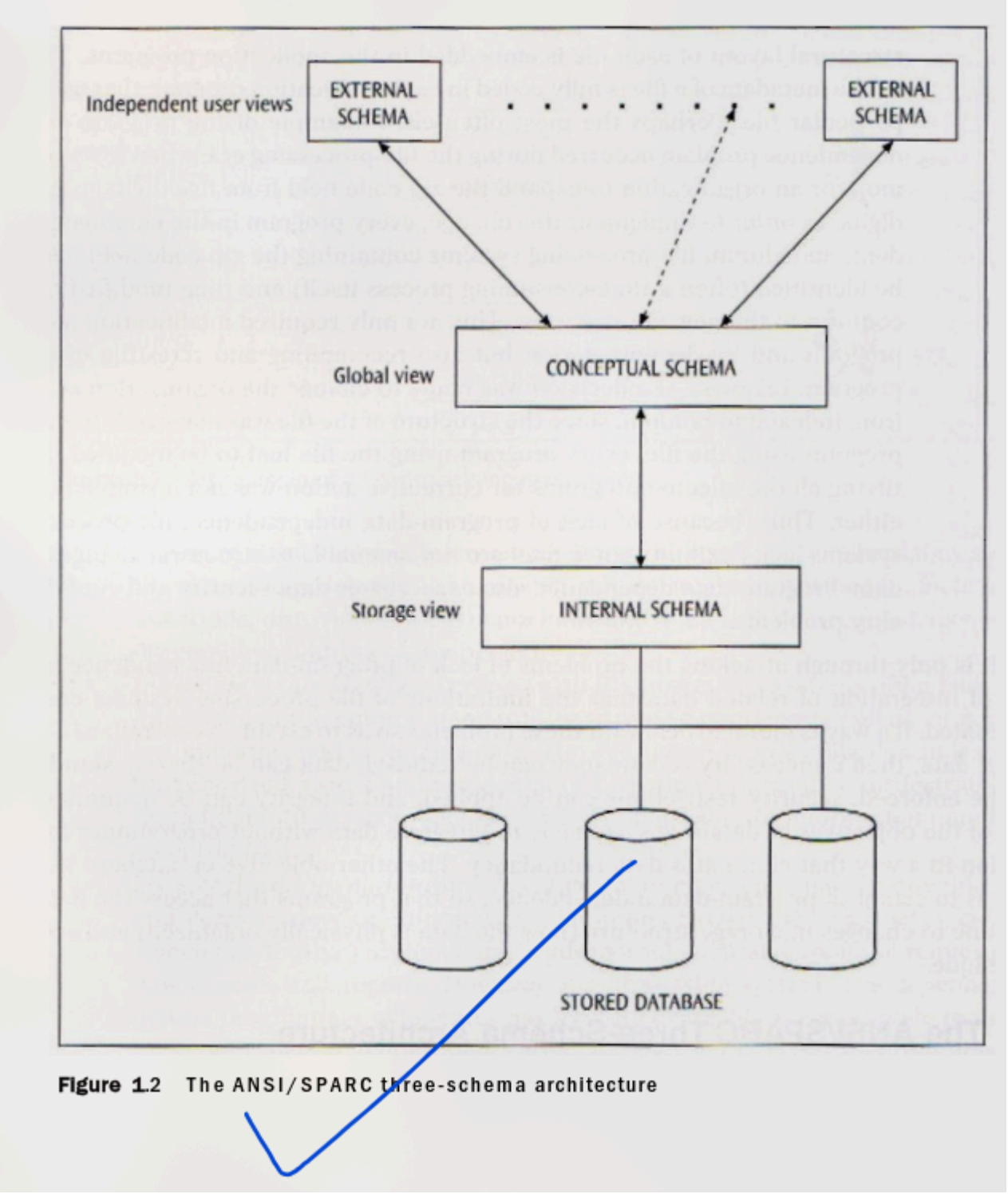


Figure 1.2 The ANSI/SPARC three-schema architecture

  - **Purpose:** To achieve program-data independence. *(Fig 1.2, p.10)*
  - **Levels:**
    - **External Schema (User Views/Subschemas):** Individual user or application views of relevant database portions. Multiple external schemas possible.
    - **Conceptual Schema (Global View):** Community view of the entire database. Describes entities, relationships, constraints. Technology-independent. Hides physical storage details.
    - **Internal Schema (Physical View):** Describes physical storage structures, access paths (indexes, hashing). Technology-dependent.

- **Data Independence:**
  - **Logical Data Independence:** Immunity of external schemas to changes in the conceptual schema.
  - **Physical Data Independence:** Immunity of conceptual schema (and external schemas) to changes in the internal schema.

- **Database System vs. DBMS [PYQ Q1a]**
  - **Database:** A self-describing collection of interrelated data (includes data and metadata). (p.12-14)
    - Types: Single-user (desktop), Multi-user (workgroup, enterprise), Distributed Database (DDB), Data Warehouse.
  - **Database Management System (DBMS):** General-purpose software to define, construct, and manipulate a database. (p.15-16)

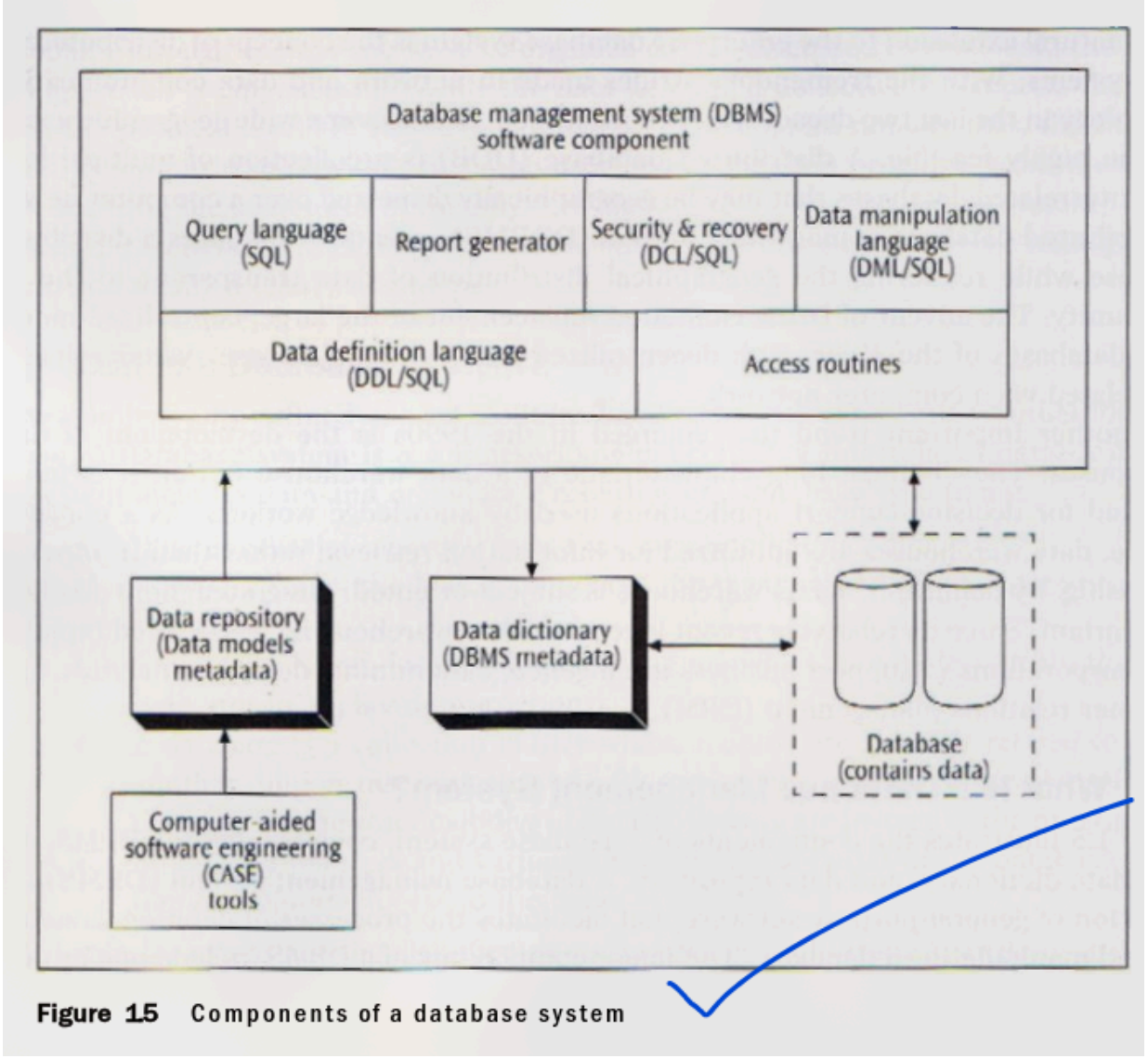


Figure 1.5 Components of a database system

  - **Components [PYQ Q1a]:** *(Fig 1.5, p.16)*
    - Query Languages (e.g., SQL)
    - Report Generators
    - Security, Integrity, Backup & Recovery facilities

        - Data Definition Language (DDL): Defines database structure (e.g., `CREATE TABLE`).
        - Data Manipulation Language (DML): Accesses and manipulates data (e.g., `INSERT`, `SELECT`, `UPDATE`, `DELETE`).
        - Data Control Language (DCL): Manages permissions (e.g., `GRANT`, `REVOKE`).
        - Data Dictionary/Repository: Stores metadata.
- **Advantages of Database Systems** (p.17-18)
    - Controlled redundancy.
    - Improved data integrity and consistency.
    - Improved data sharing and accessibility.
    - Enforcement of standards.
    - Improved data security.
    - Program-data independence.
    - Increased productivity and reduced maintenance.

- **Database Design Life Cycle [PYQ Q2b]** (p.20-22) *(Fig 1.7, p.19 & p.25)*

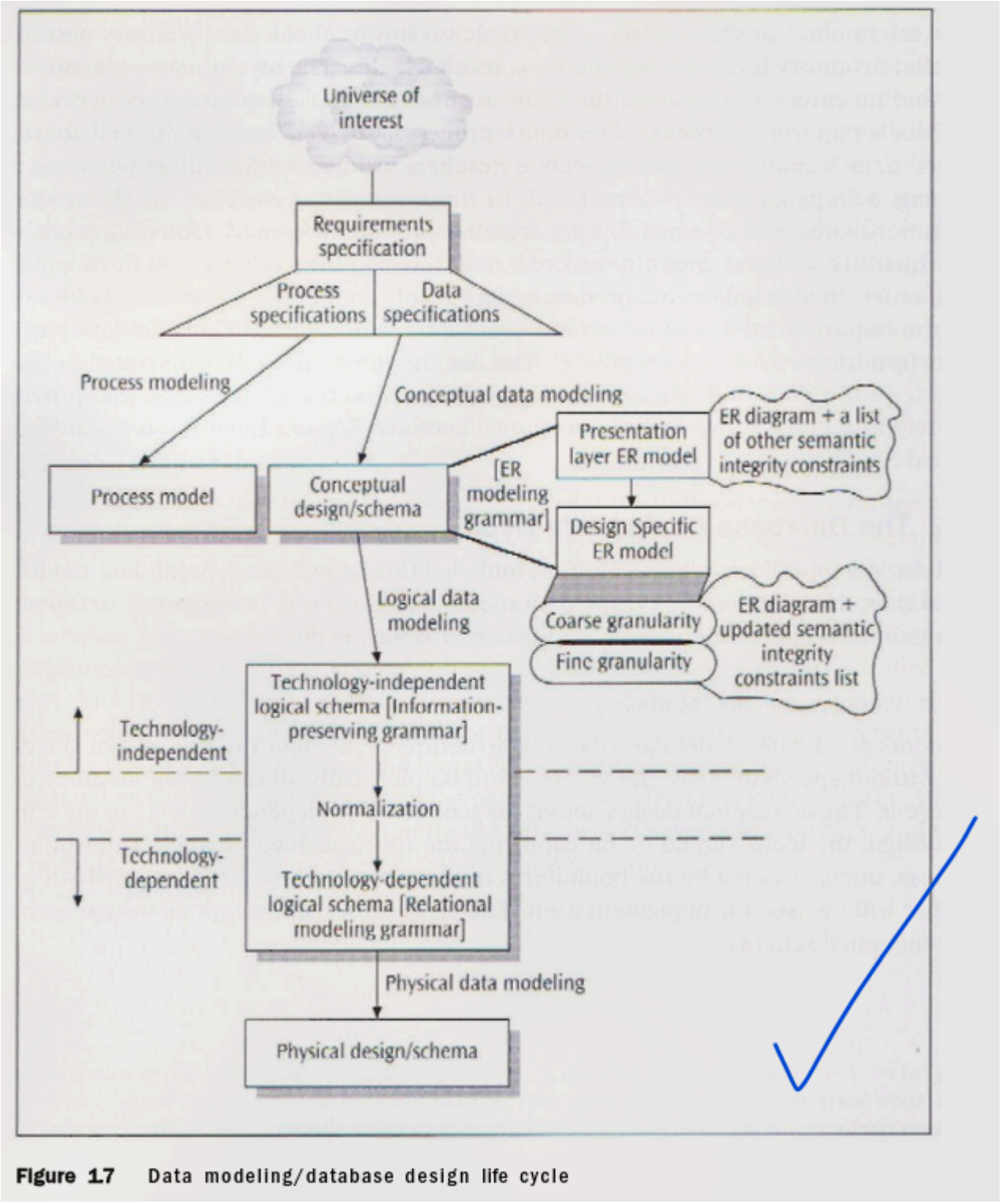


Figure 1.7 Data modeling/database design life cycle

  - **1. Requirements Specification:** Analysts review documents, interview users to identify objectives, data, and process specifications (business rules).
  - **2. Conceptual Data Modeling (Technology-Independent):** Describes data structure without physical storage details. Focus on capturing user-specified business rules. Product: Conceptual Schema (e.g., ER/EER model).
    - Includes Presentation Layer (for user communication) and Design-Specific Layer (for database design).
    - Validation is crucial.

- **3. Logical Data Modeling (Technology-Dependent):** Transforms conceptual schema to be compatible with a chosen data model (e.g., relational, network, hierarchical). Product: Logical Schema. Normalization is part of this.
- **4. Physical Data Modeling (DBMS-Specific):** Specifies internal storage structures, access strategies using tools of a particular DBMS.

2. Conceptual Data Modelling

- **ER Modeling [PYQ Q3b, Q4a]**
  - **Framework:** (p.26) Model represents real-world phenomena using a grammar (constructs & rules) and a method.
  - **ER Modeling Primitives:** (p.26-27) *(Table 2.1, p.26-27)*

Table 2.1 Equivalence between real world primitives and conceptual primitives

Real World Primitive	Conceptual Primitive
Object {type}	Entity(ies) {type}
Object (occurrence)	Entity(ies) {instance}
Property	Attribute
Fact	Value

Table 2.1 Equivalence between real world primitives and conceptual primitives (continued)

Real World Primitive	Conceptual Primitive
Property value set	Domain
Association	Relationship
Object class	Entity class

    - **Entity Type:** A collection of similar objects (e.g., STUDENT, COURSE). Represented by a rectangle.
    - **Entity Instance:** A specific occurrence of an entity type (e.g., student John Doe).
    - **Attribute:** A property or characteristic of an entity type (e.g., StudentName, CourseID). Represented by an oval.

- **Types of Attributes:** (p.28-30) *(Table 2.2, p.28; Fig 2.1, p.29; Fig 2.2, p.30)*

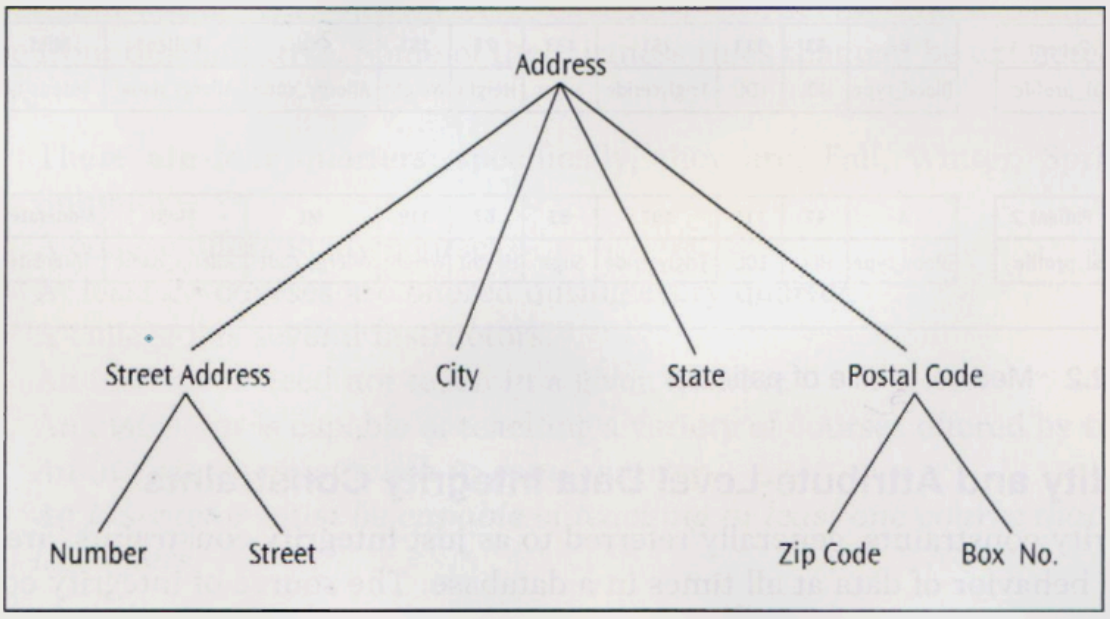


Figure 2.1 An example of a composite attribute hierarchy

								SUL	Sulfa	Severe
								PCN	Penicillin	Mild
Patient 1	B+	53	111	151	133	71	151	POL	Pollen	Mild
Medical_profile	Blood_type	HDL	LDL	Triglyceride	Sugar	Height	Weight	Allergy_code	Allergy_name	Intensity
Patient 2	A-	41	111	197	93	67	119	ML	Mold	Moderate
Medical_profile	Blood_type	HDL	LDL	Triglyceride	Sugar	Height	Weight	Allergy_code	Allergy_name	Intensity

Figure 2.2 Medical profile of patients

  - **Simple (Atomic):** Cannot be subdivided (e.g., Age).
  - **Composite:** Can be subdivided into smaller parts (e.g., Address into Street, City, Zip).
  - **Single-valued:** Has only one value per entity instance (e.g., DateOfBirth).
  - **Multi-valued:** Can have multiple values (e.g., Skills of an employee). Represented by a double oval.
  - **Stored:** Value is directly stored (e.g., BirthDate).
  - **Derived:** Value is calculated from other attributes (e.g., Age from BirthDate). Represented by a dotted oval.
  - **Mandatory:** Must have a value (dark circle).
  - **Optional:** May not have a value (empty circle).
  - **Complex:** Nested composite and/or multi-valued attributes.
- **Domain:** Set of possible values for an attribute.

- **Unique Identifiers (Keys):** [Related to PYQ Q1b] (p.32-33) Attribute(s) whose values uniquely identify each entity instance. Underlined in ERD.
  - **Candidate Key:** A minimal set of attributes that uniquely identifies an entity instance. An entity type can have multiple candidate keys.
  - **Primary Key:** The candidate key chosen to be the main identifier.
  - **Superkey:** Any set of attributes that uniquely identifies an entity, may contain redundant attributes.
  - A key attribute is part of a candidate key.
- **Relationship Type:** Meaningful association among two or more entity types (e.g., ENROLLS between STUDENT and COURSE). Represented by a diamond.

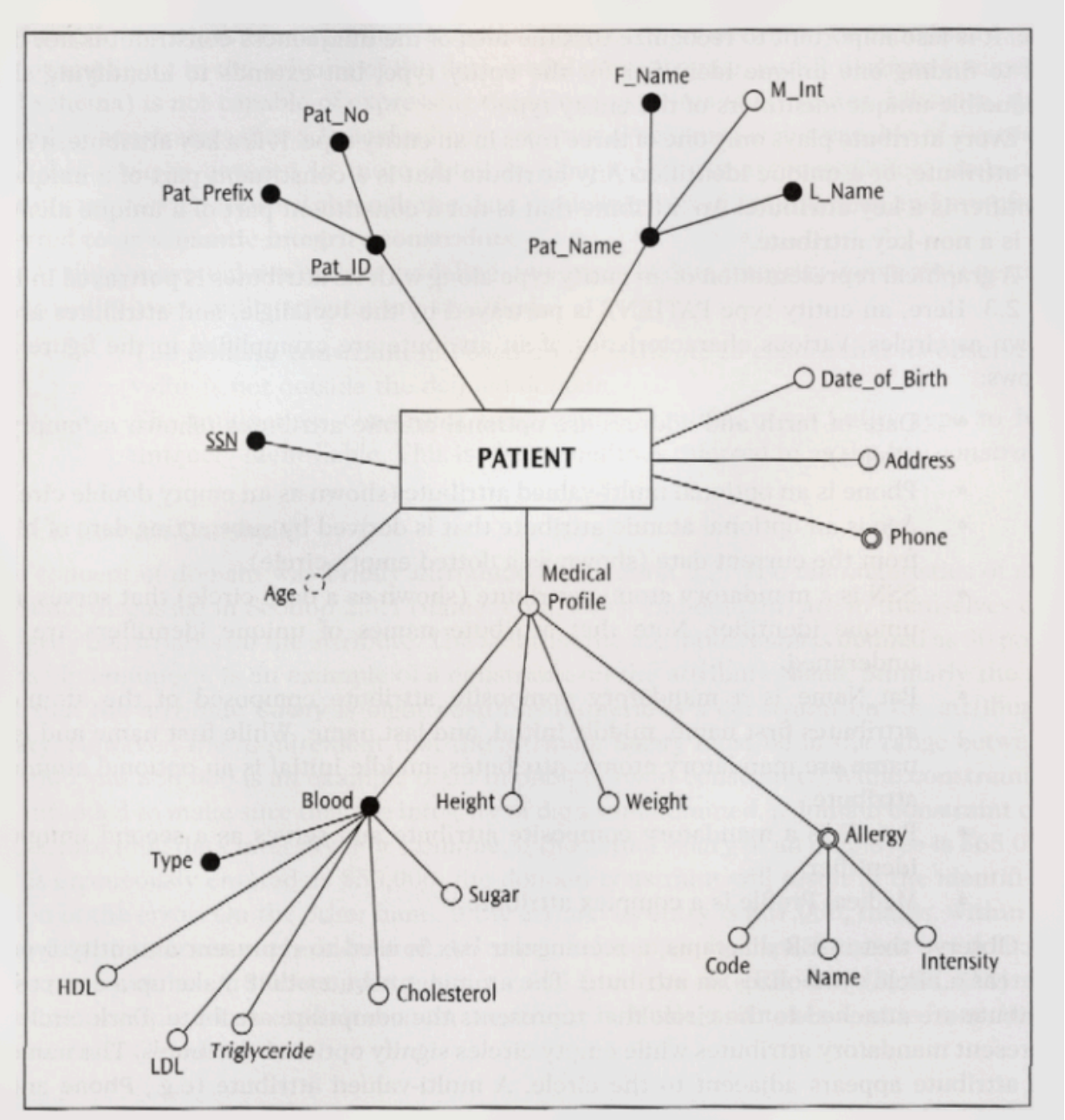


Figure 2.3 A graphical representation of an entity type PATIENT

  - **Relationship Instance:** An association between specific entity instances.
  - **Degree of a Relationship:** Number of participating entity types. (p.33)

- **Binary (degree 2):** Involves two entity types. *(Fig 2.4, p.35)*

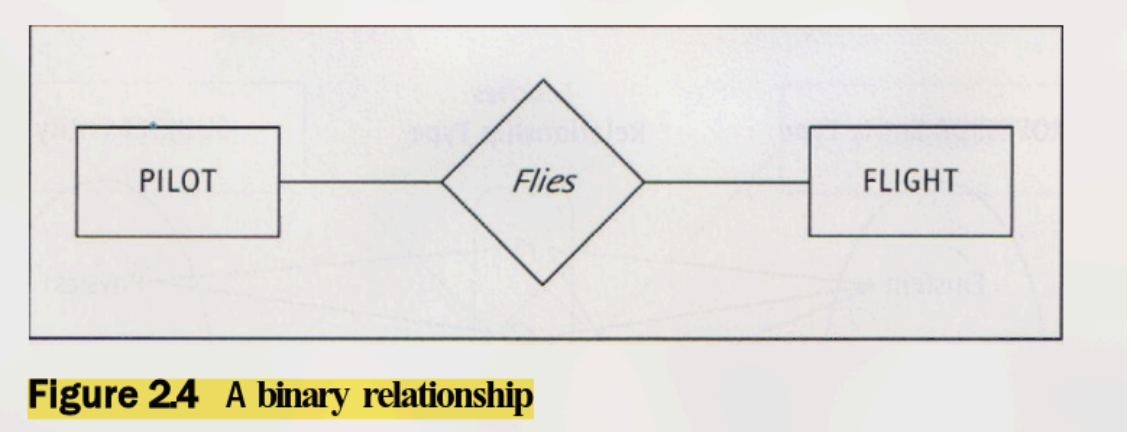


Figure 2.4 A binary relationship

- **Ternary (degree 3):** Involves three entity types. *(Fig 2.5, p.35)*

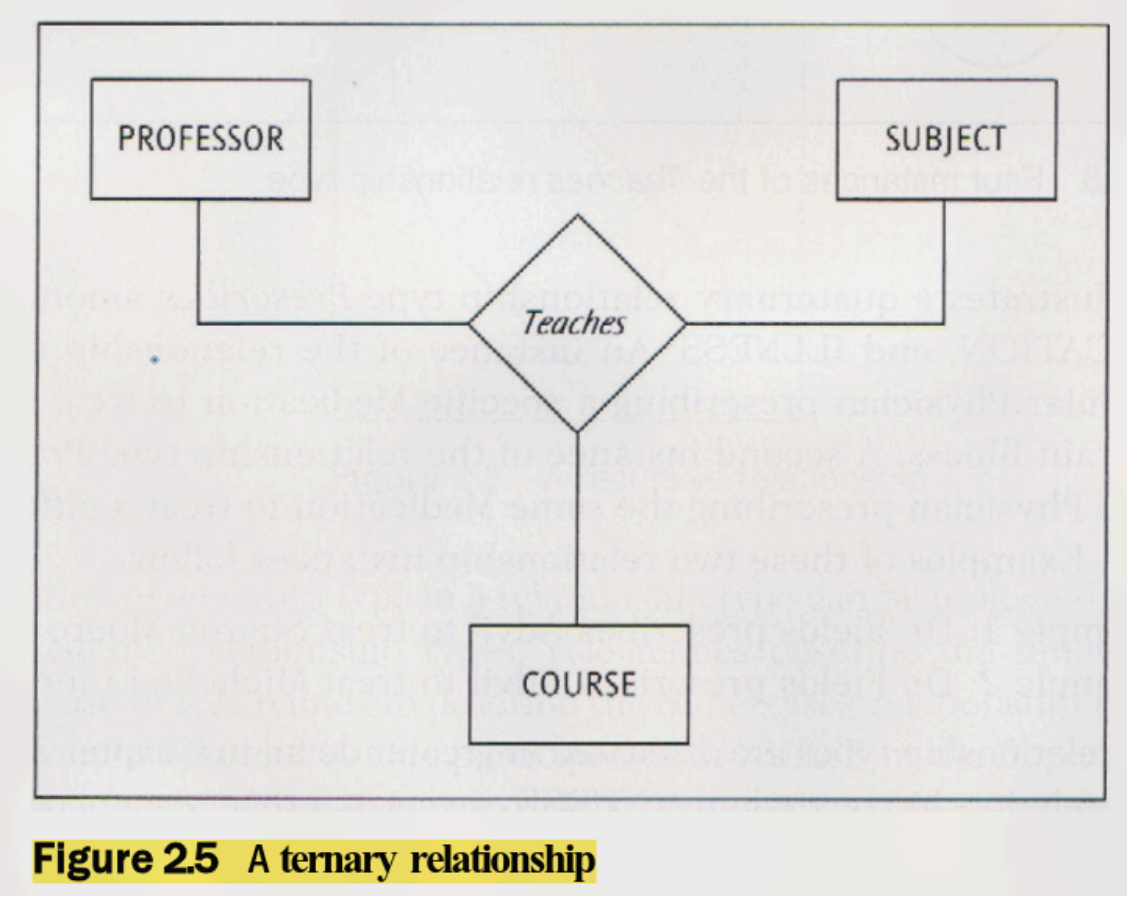


Figure 2.5 A ternary relationship

- **N-ary (degree n):** Involves 'n' entity types.

- **Recursive (Unary):** Relationship between instances of the same entity type. *(Fig 2.8, p.37)*

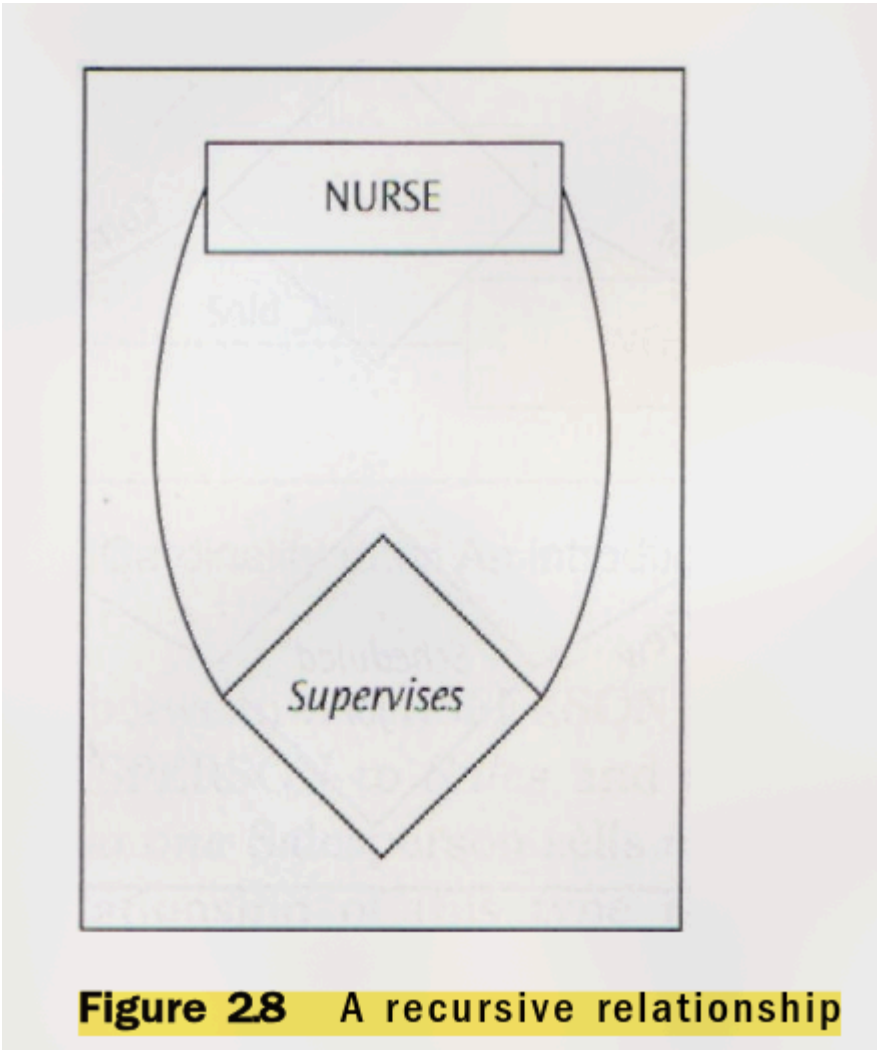


Figure 2.8 A recursive relationship

- **Role Names:** Clarify the role an entity type plays in a relationship, especially in recursive relationships or when multiple relationships exist between the same entity types. *(Fig 2.9, p.38; Fig 2.10, p.38)*

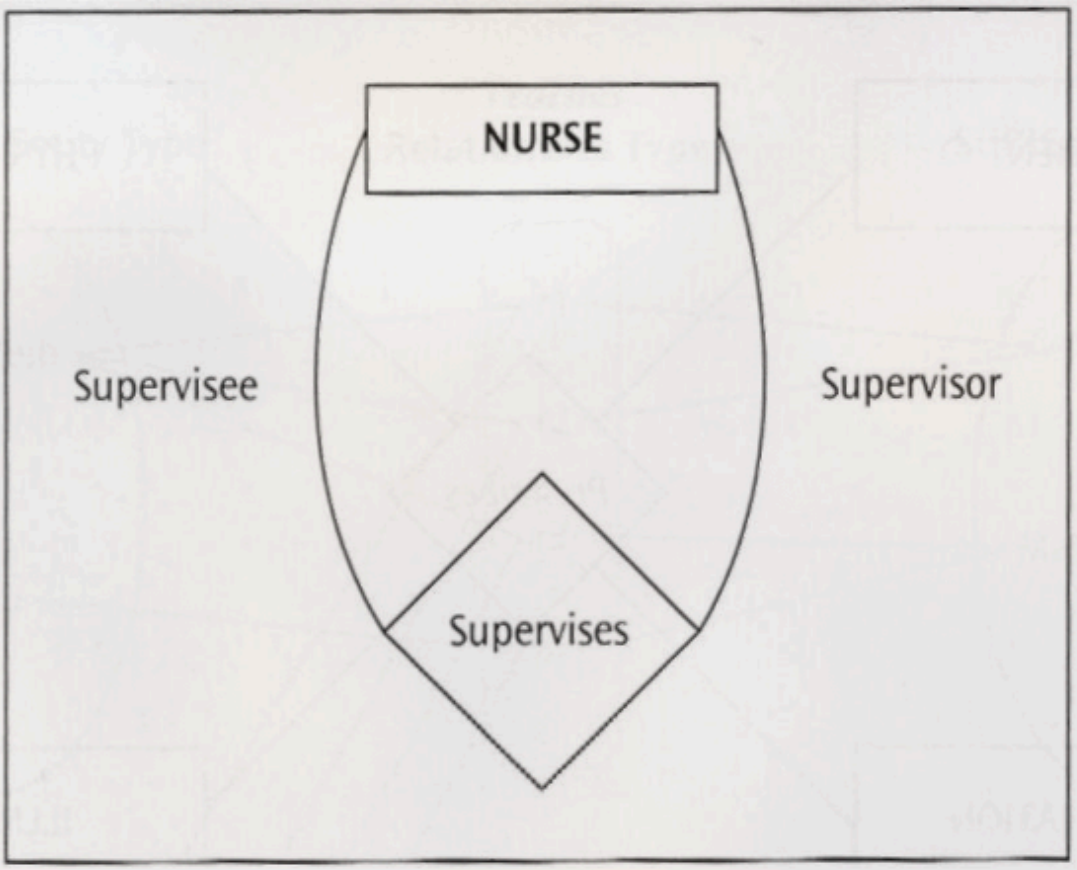


Figure 2.9 Role names in a recursive relationship

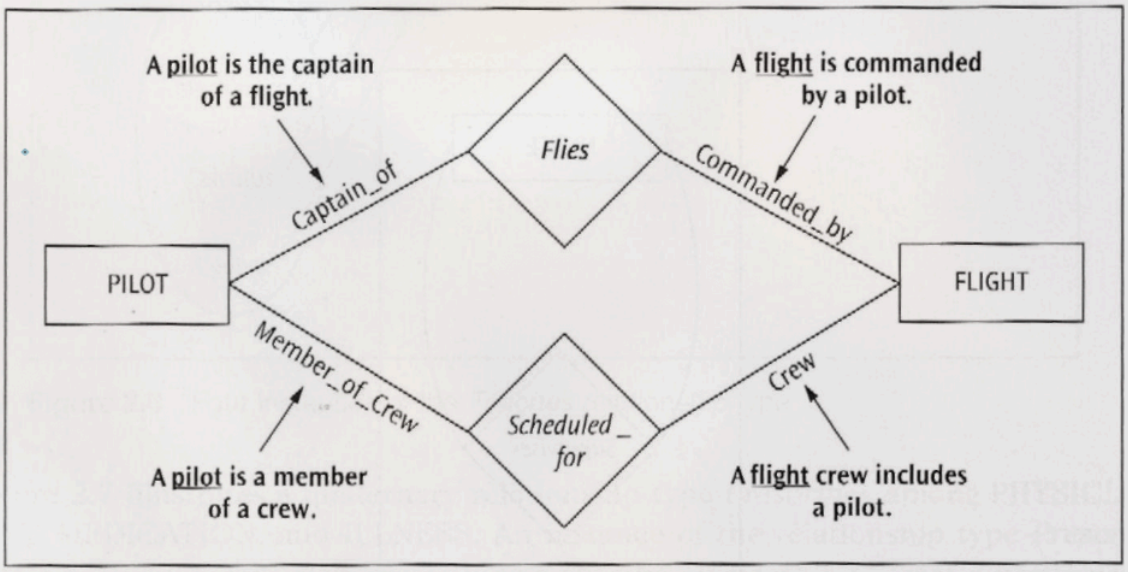
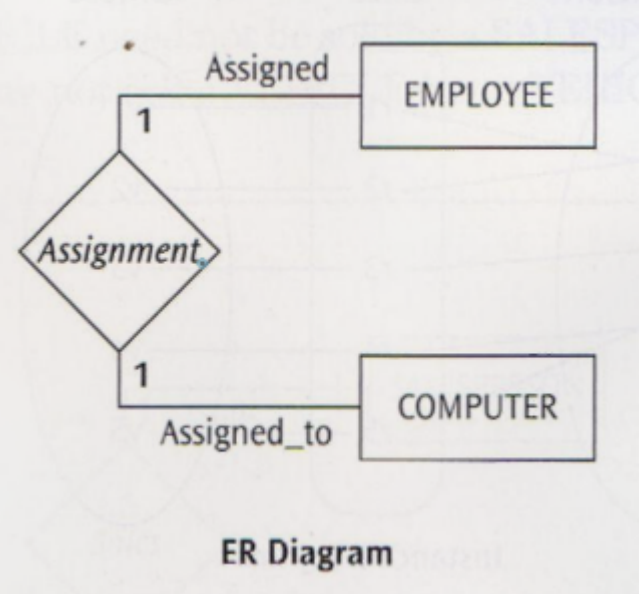


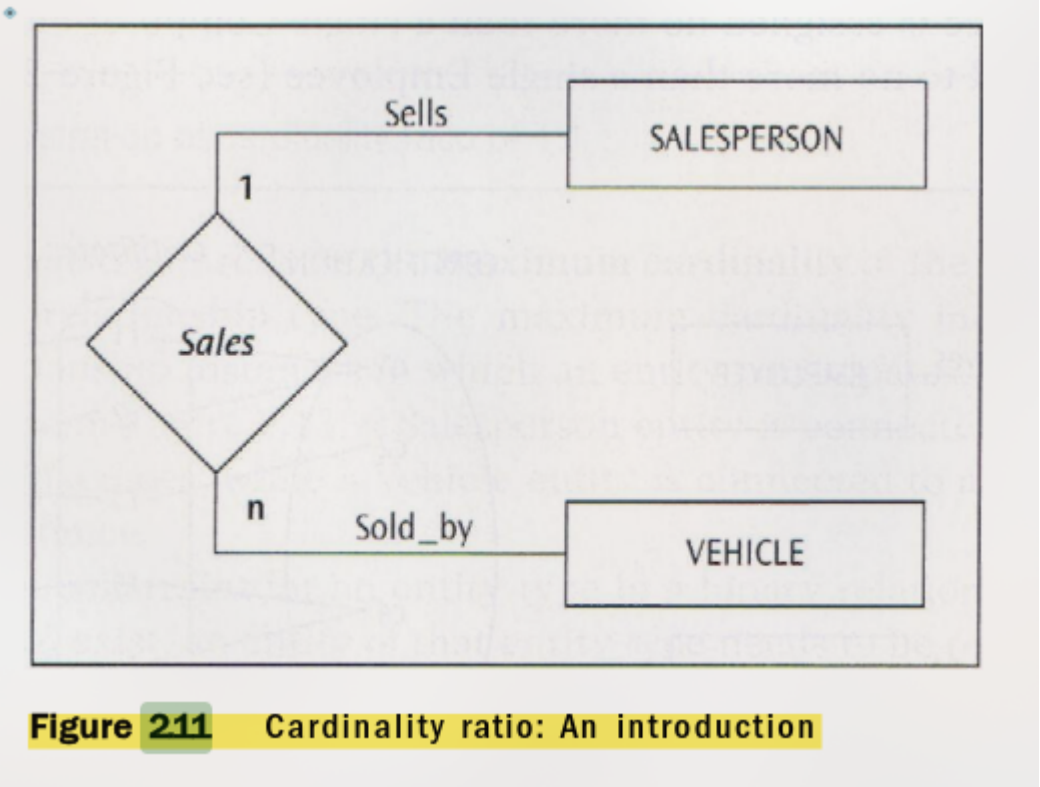
Figure 2.10 Role names in binary relationships

- **Structural Constraints (Cardinality & Participation):** (p.38-46)
  - **Cardinality Ratio (Connectivity/Max):** Maximum number of relationship instances an entity can participate in.

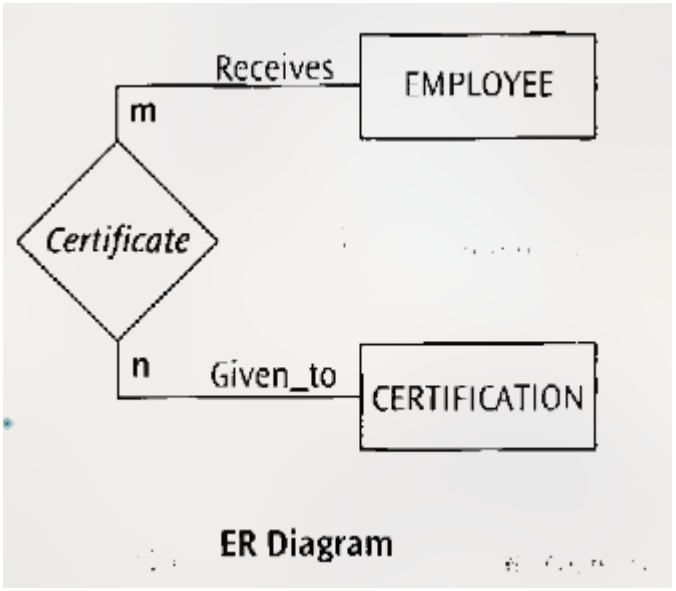
- 1:1 (One-to-One) *(Fig 2.14, p.41)*



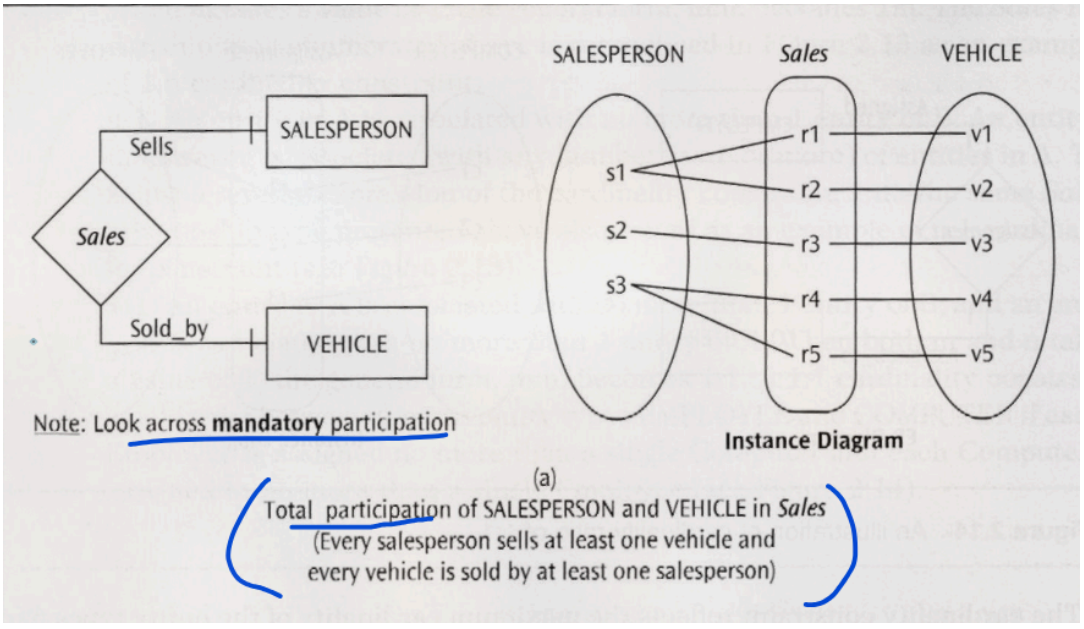
- 1:N (One-to-Many) *(Fig 2.11 & 2.13, p.39-40)*



- M:N (Many-to-Many) *(Fig 2.12, p.40)*



- **Participation Constraint (Min):** Specifies whether an entity's existence depends on its being related to another entity via the relationship.
  - **Total (Mandatory):** Every entity instance must participate (shown by a bar or (1,N)). *(Fig 2.15a, p.42; Fig 2.16a,c,d p.43-44)*
  - **Partial (Optional):** Entity instance may or may not participate (shown by an oval or (0,N)). *(Fig 2.15b, p.42; Fig 2.16b,c,d p.43-44)*



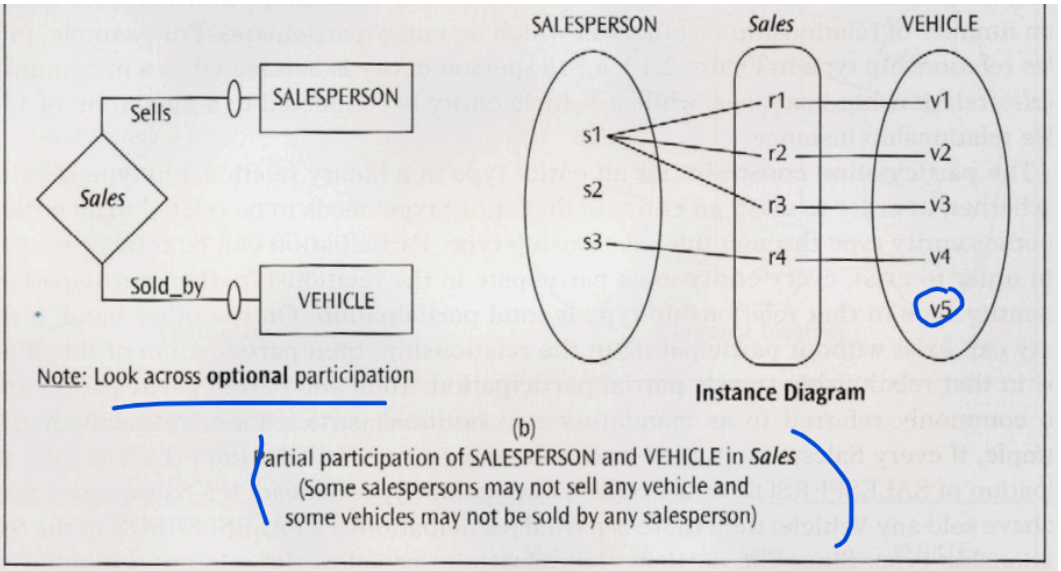


Figure 2.15 Examples of the participation constraint

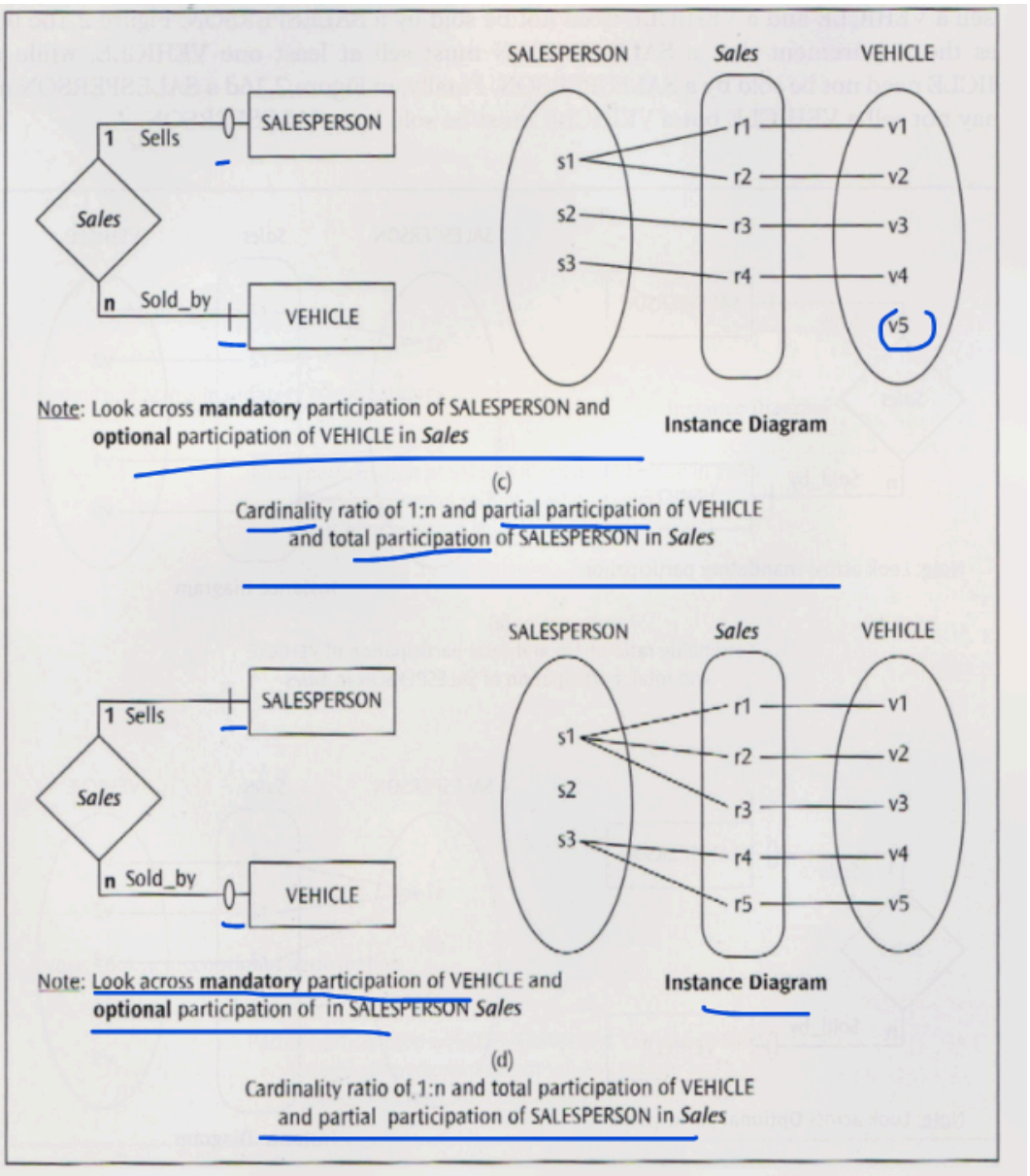


Figure 2.16 Cardinality ratio and participation constraints for a relationship (continued)

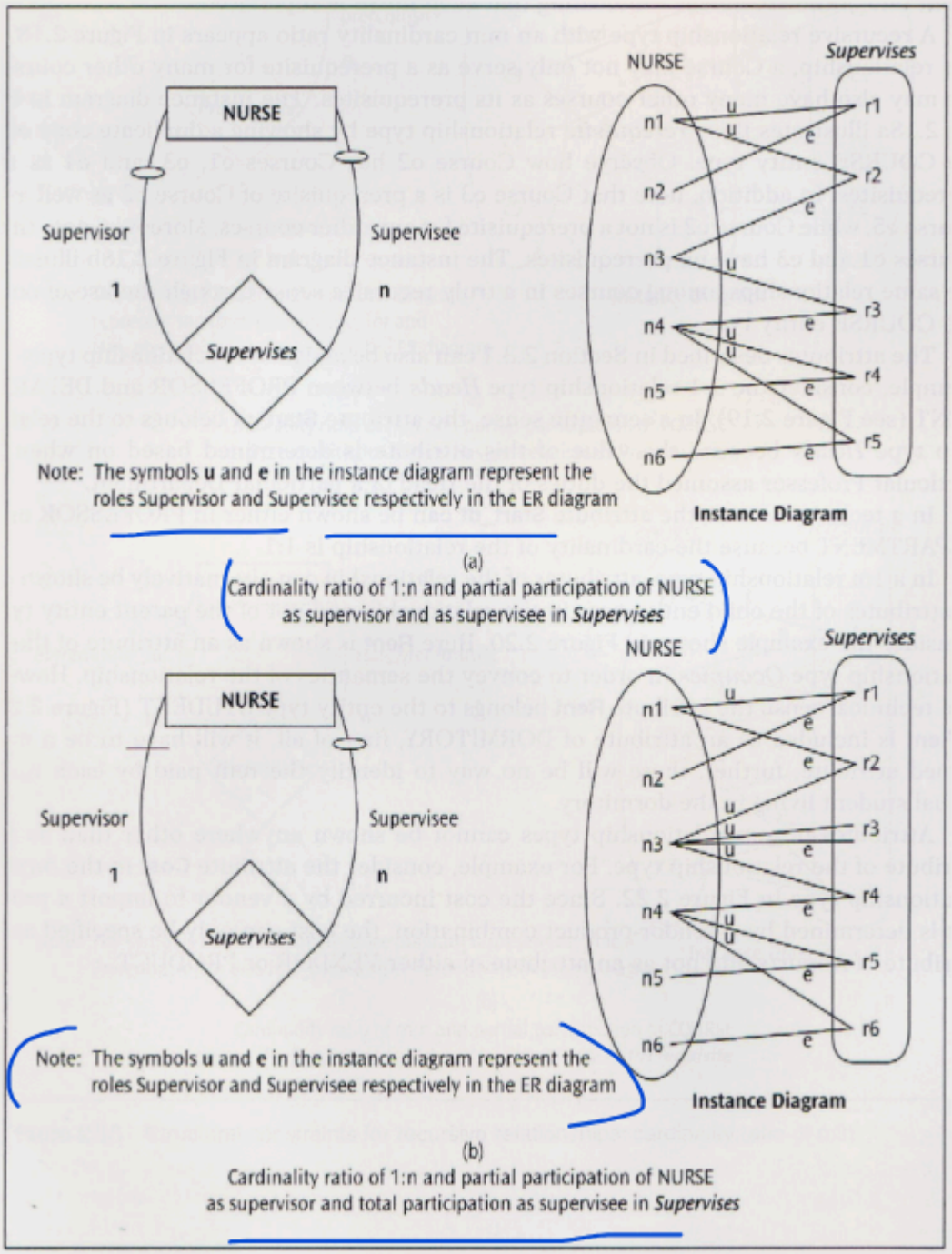


Figure 2.17 Structural constraints for recursive relationships: cardinality ratio of 1:n

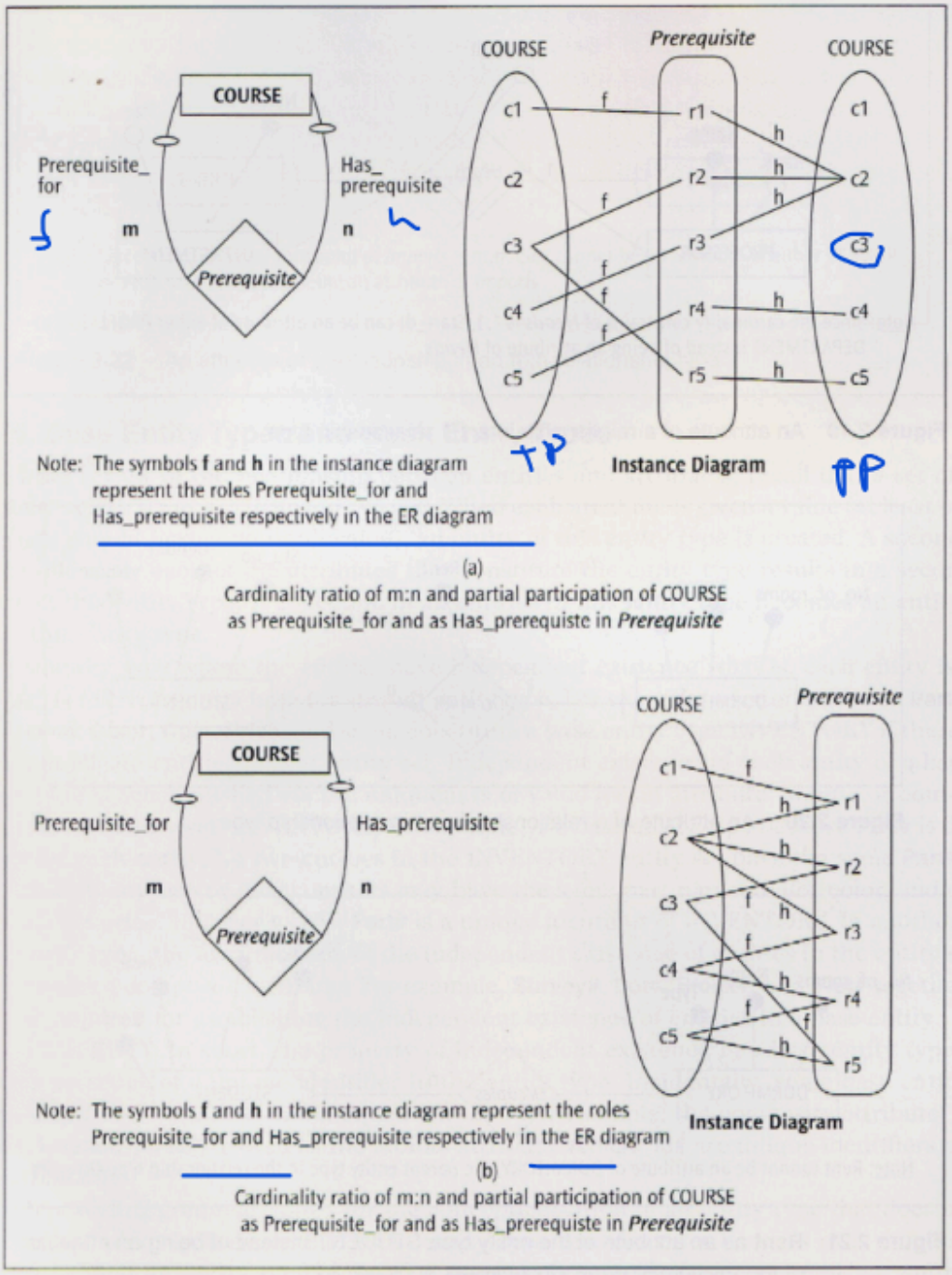


Figure 2.18 Structural constraints for recursive relationships: cardinality ratio of m:n

- **Attributes on Relationships:** Can exist, especially for M:N relationships or sometimes 1:1 or 1:N if the attribute describes the interaction. (p.46-48) *(Fig 2.19, 2.20, 2.22, p.48-49)*

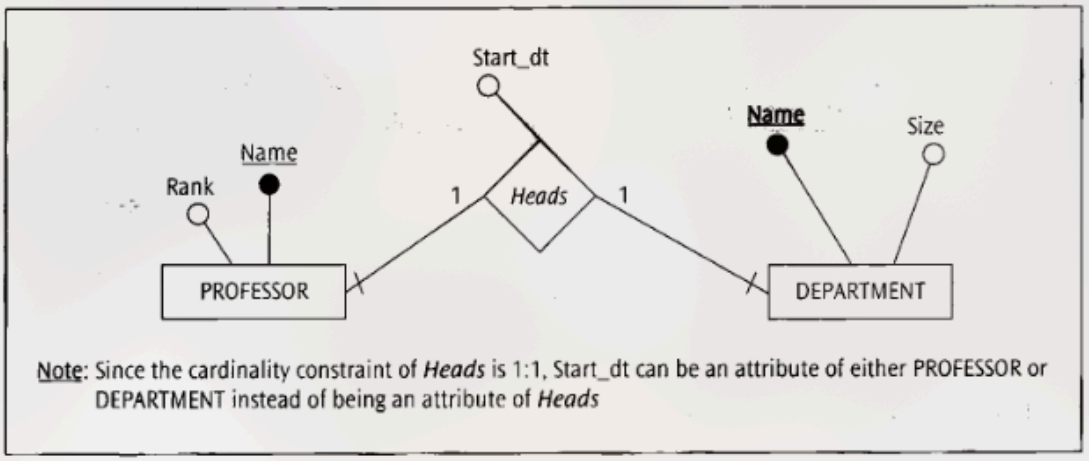


Figure 2.19 An attribute of a relationship in a **1:1** relationship type

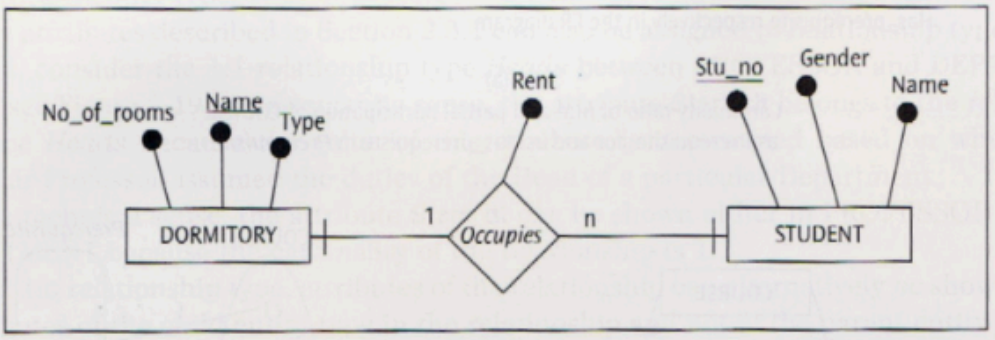


Figure 2.20 An attribute of a relationship in a **1**:n relationship type

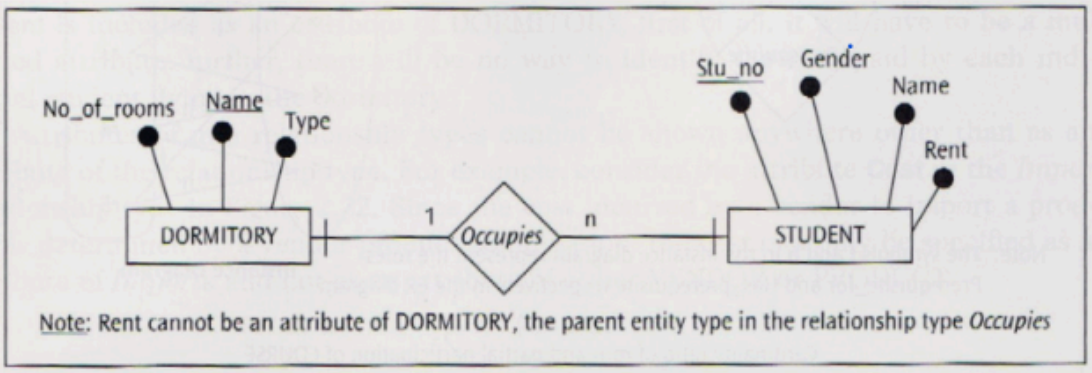


Figure 2.21 **Rent** as an attribute of the entity type **STUDENT** instead of being an attribute of the relationship type *Occupies* as in **Figure 2.20**

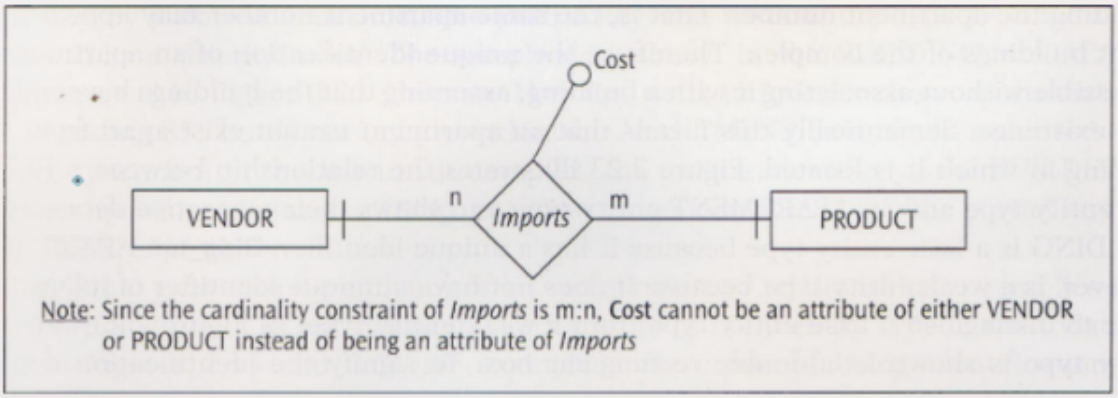
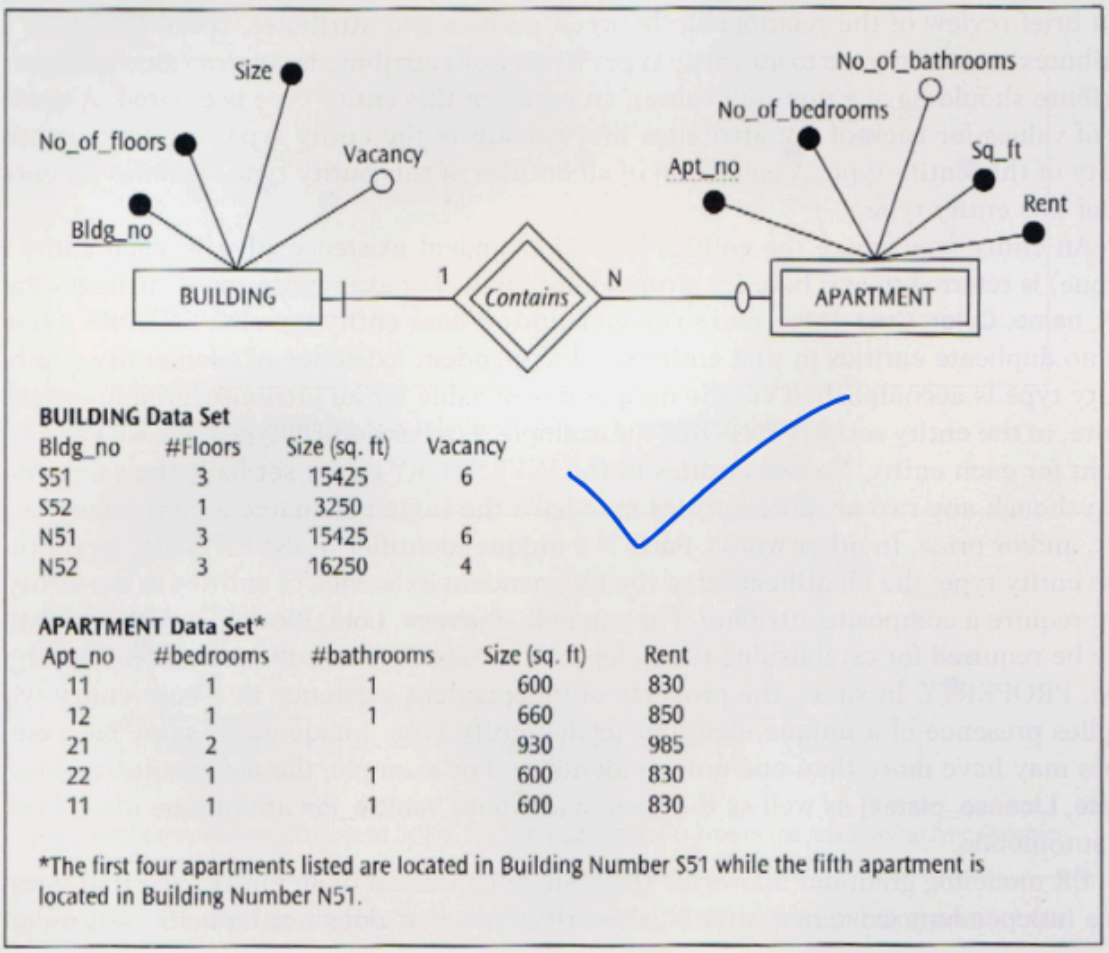


Figure 2.22 An attribute of a relationship in an m:n relationship type

- **Base (Strong) Entity Type:** Has its own unique identifier and exists independently. (p.49)
- **Weak Entity Type:** Does not have a unique identifier of its own; its existence depends on a strong (owner) entity type. Identified by combining its partial key with the owner's key. (p.49-53)
  - **Identifying Relationship:** Links a weak entity to its owner. Represented by a double diamond.
  - **Partial Key (Discriminator):** Attribute(s) that distinguish instances of a weak entity type related to the same owner entity. Dotted underline. *(Fig 2.23, p.50; Fig 2.24, p.51; Fig 2.26, p.53)*



BUILDING Data Set

Bldg_no	#Floors	Size (sq. ft)	Vacancy
S51	3	15425	6
S52	1	3250	
N51	3	15425	6
N52	3	16250	4

APARTMENT Data Set*

Apt_no	#bedrooms	#bathrooms	Size (sq. ft)	Rent
11	1	1	600	830
12	1	1	660	850
21	2		930	985
22	1	1	600	830
11	1	1	600	830

*The first four apartments listed are located in Building Number S51 while the fifth apartment is located in Building Number N51.

Figure 2.23 Weak entity type: an example

- **Data Modeling Errors:** (p.54)
  - **Semantic Errors:** Misinterpretation of requirements.
  - **Syntactic Errors:** Violation of modeling grammar rules.

- **ER Modeling Process (using Bearcat Inc. example in Ch 3):** (p.55-93)

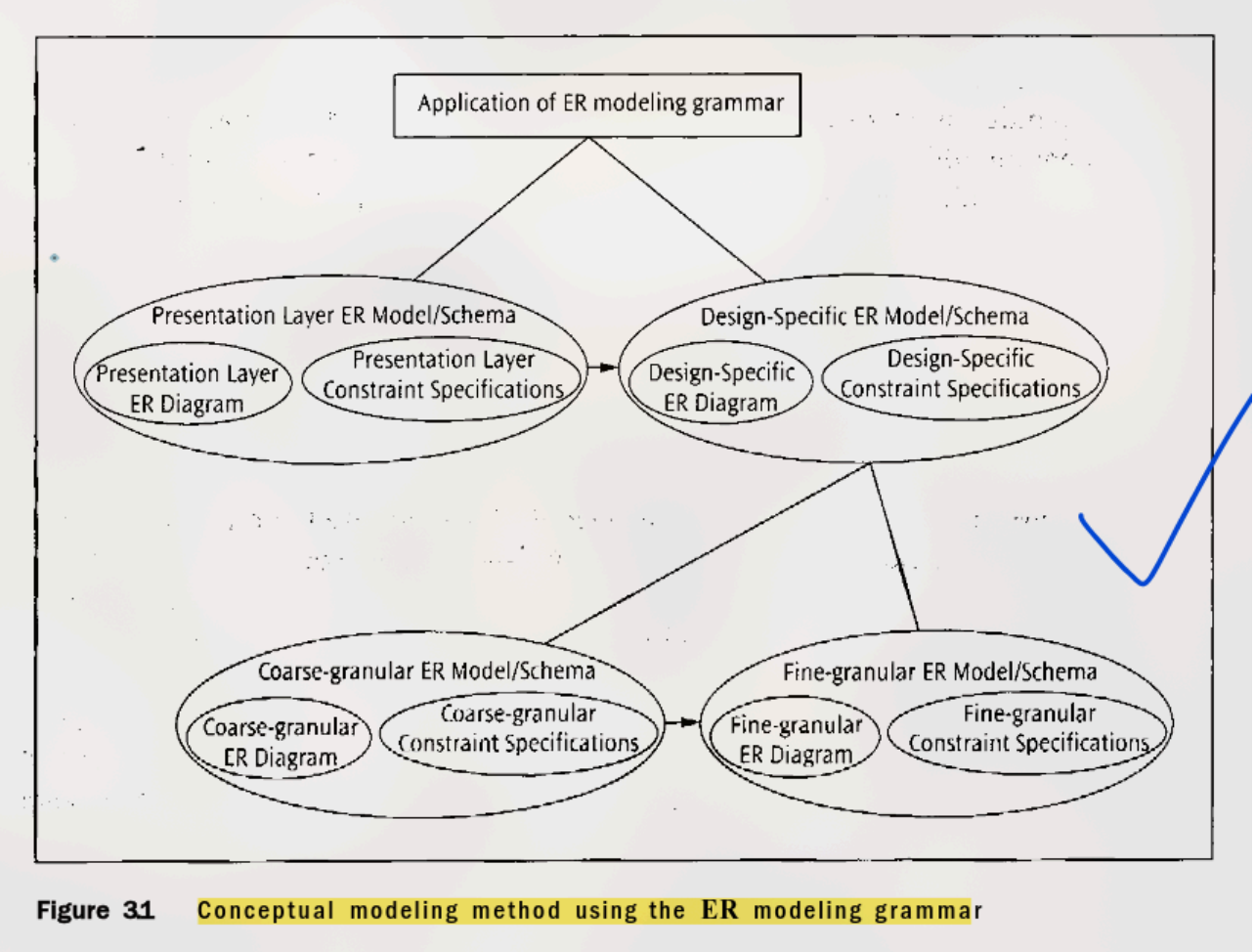


Figure 3.1 Conceptual modeling method using the ER modeling grammar

  - **Presentation Layer ER Model:** For user communication; surface-level expression. *(Fig 3.2 notation summary, p.60; Fig 3.3 Bearcat ERD, p.72)*

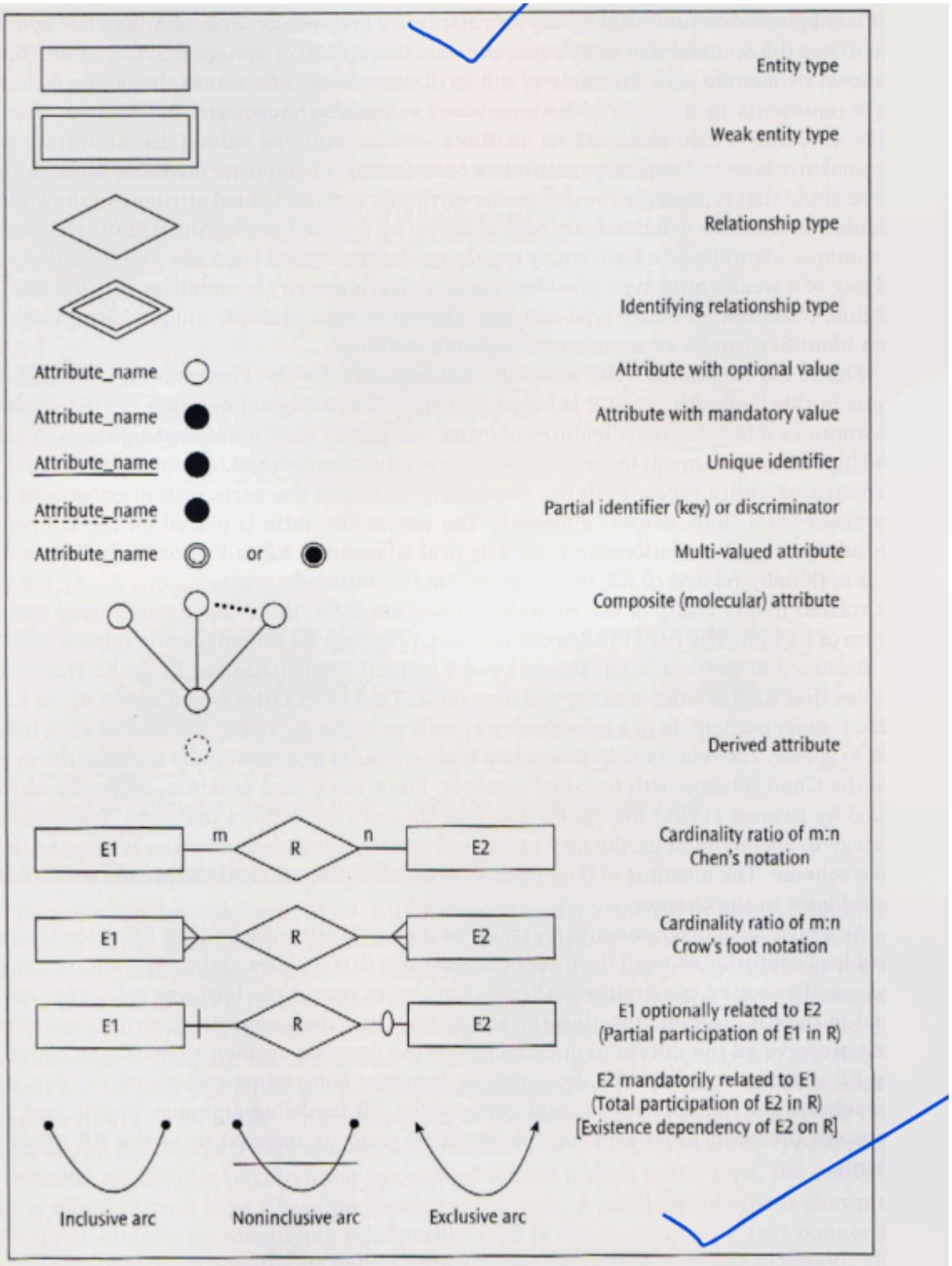


Figure 3.2 Summary of Presentation Layer ER diagram notation

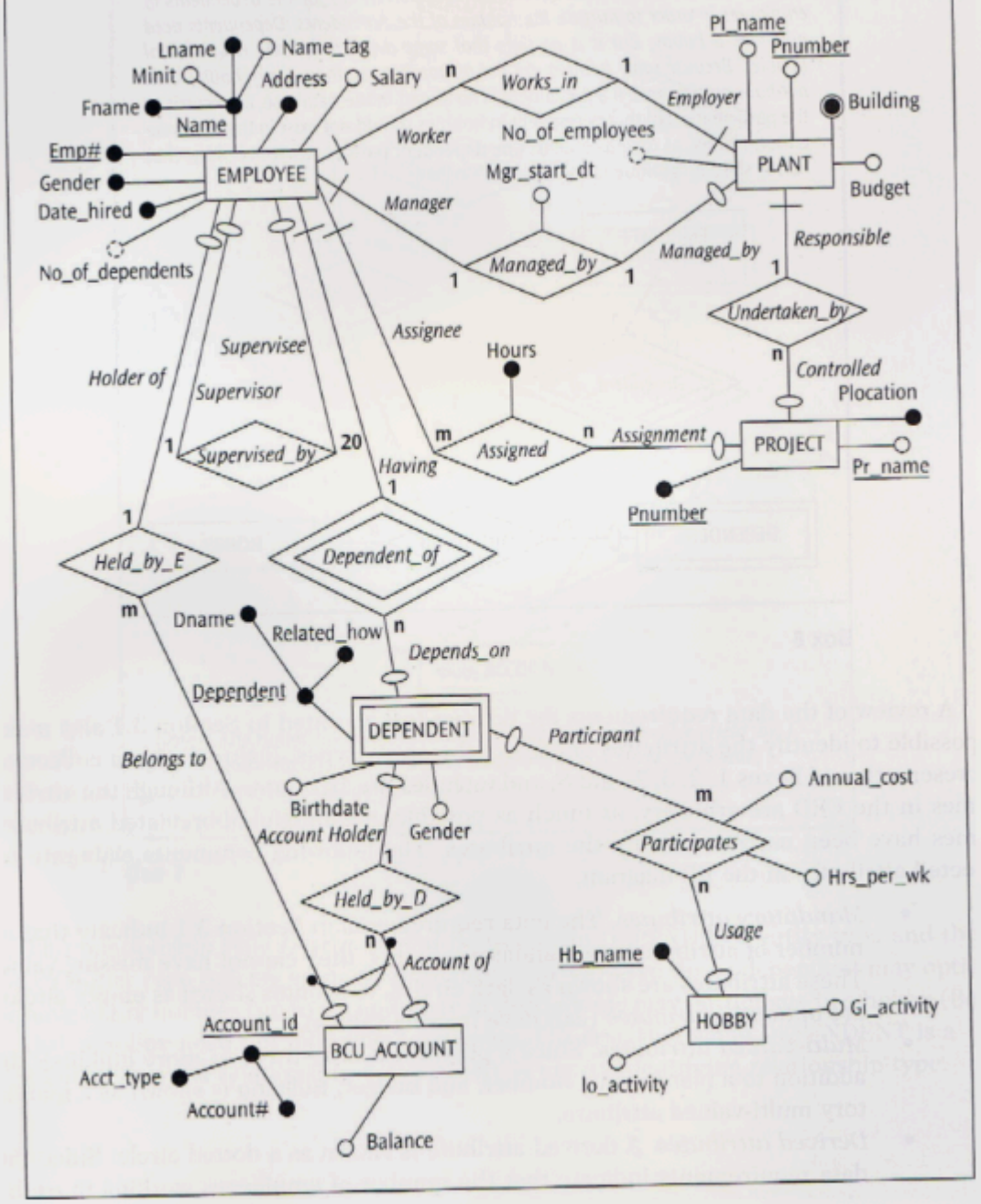


Figure 3.3 Presentation Layer ER diagram for Bearcat Incorporated

- **Design-Specific ER Model:** For database design; incorporates more technical detail.
  - **Coarse Granularity:** Adds details like (min, max) notation for relationships, deletion rules (Restrict, Cascade, Set Null, Set Default). *(Fig 3.4 min/max intro, p.78; Fig 3.5, p.79; Fig 3.6 deletion rules, p.82; Fig 3.8 Bearcat Coarse ERD, p.85)*

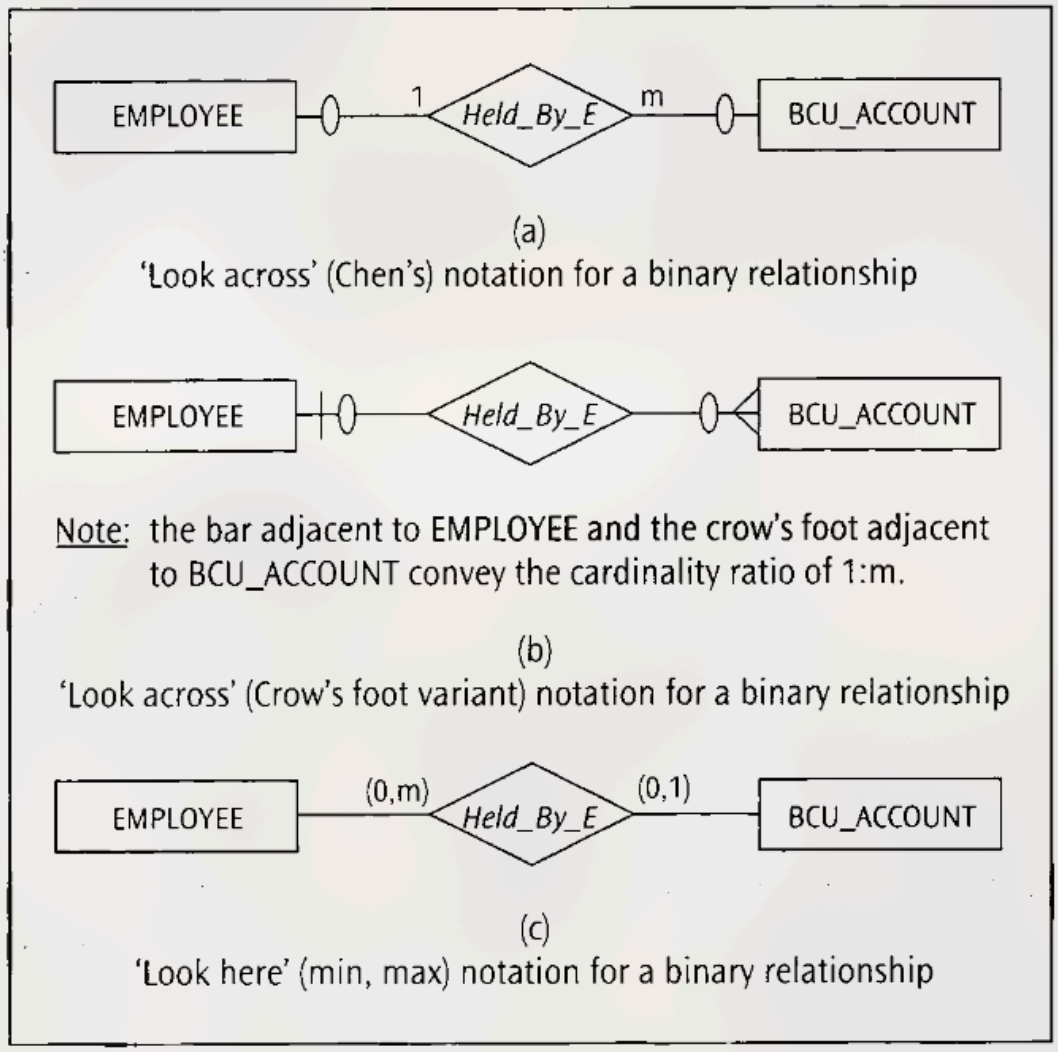


Figure 3.4 Introduction of (min, max) notation for a binary relationship

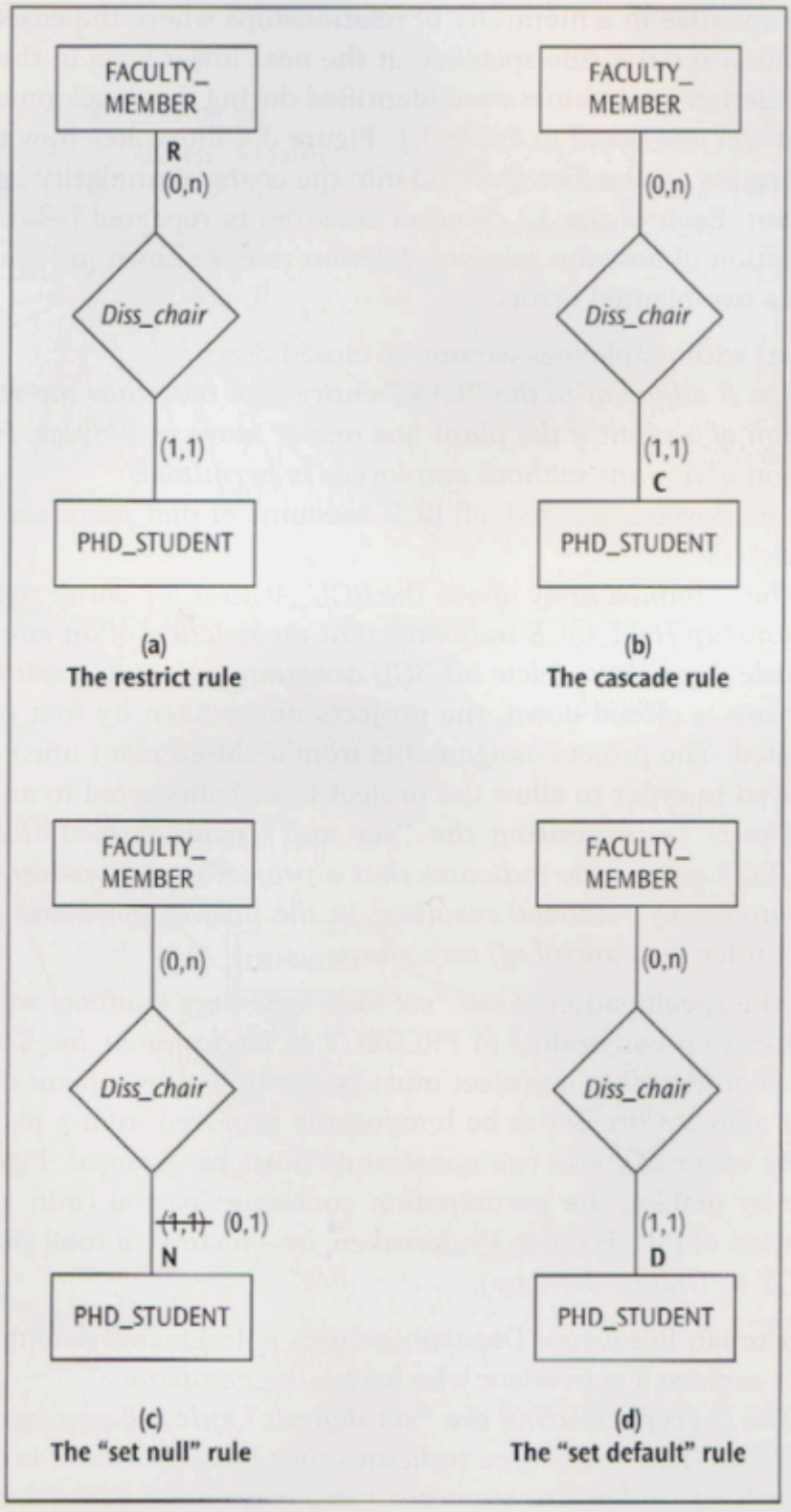


Figure 3.6 Examples of deletion rules

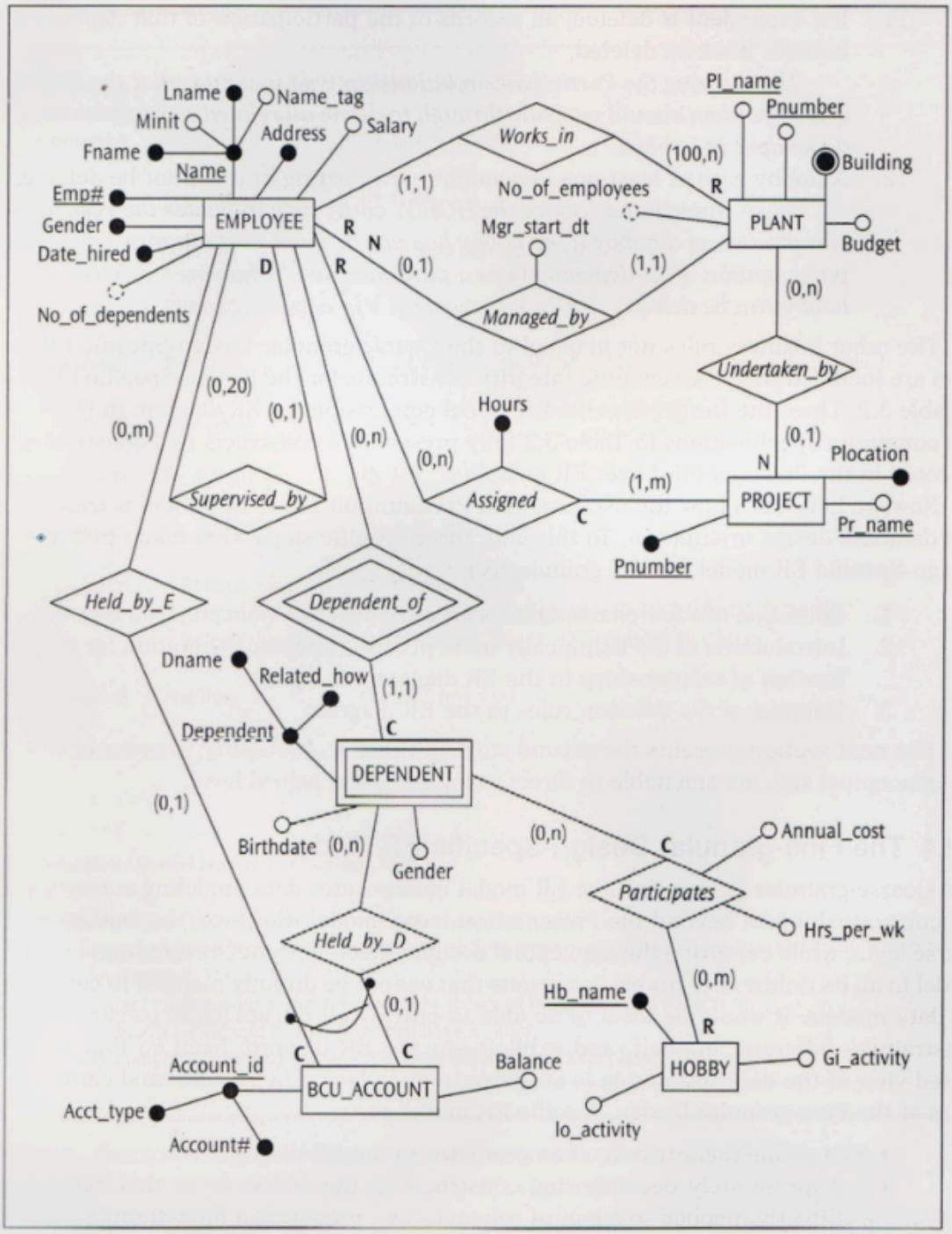


Figure 3.8 Coarse-granular Design-Specific ER diagram for Bearcat Incorporated - Final

- **Fine Granularity:** Decomposes constructs not directly mappable (M:N, multi-valued attributes), adds attribute types/sizes. Ready for logical mapping. *(Fig 3.9 multi-valued resolution, p.88; Fig 3.10 M:N resolution, p.91; Fig 3.11 Bearcat Fine ERD, p.92)*

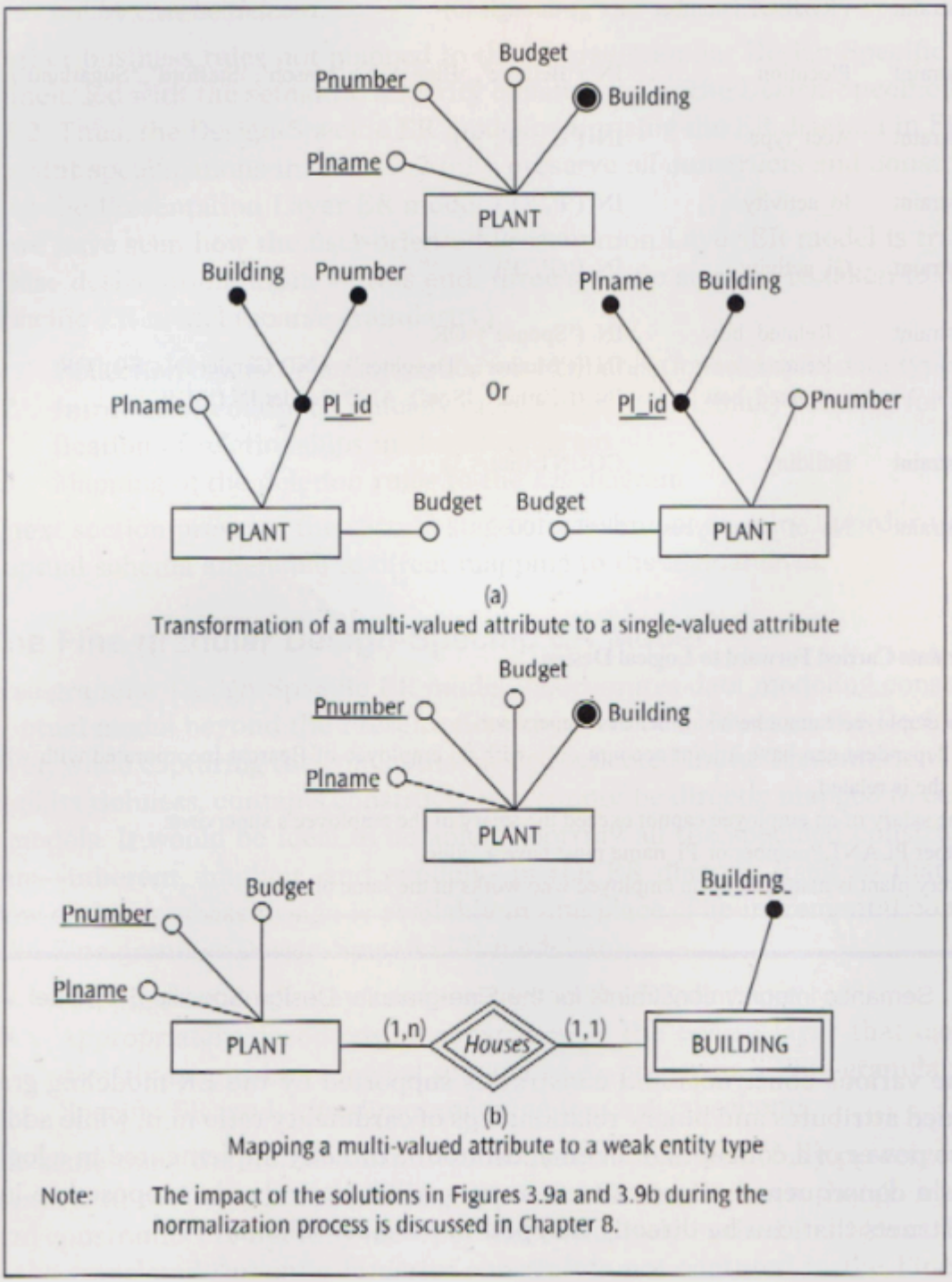


Figure 3.9 Two methods for the resolution of a multi-valued attribute

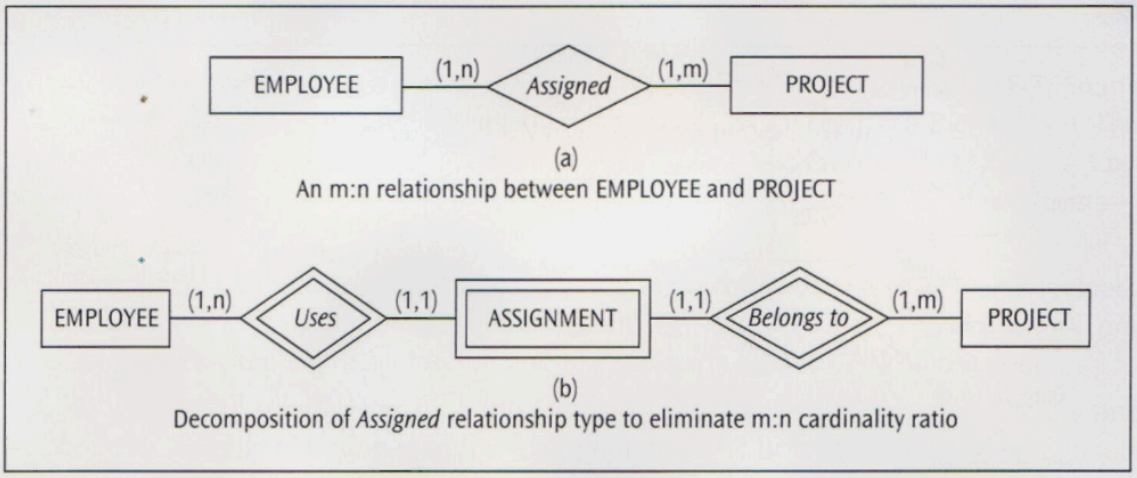


Figure 3.10 Resolution of the m:n cardinality ratio

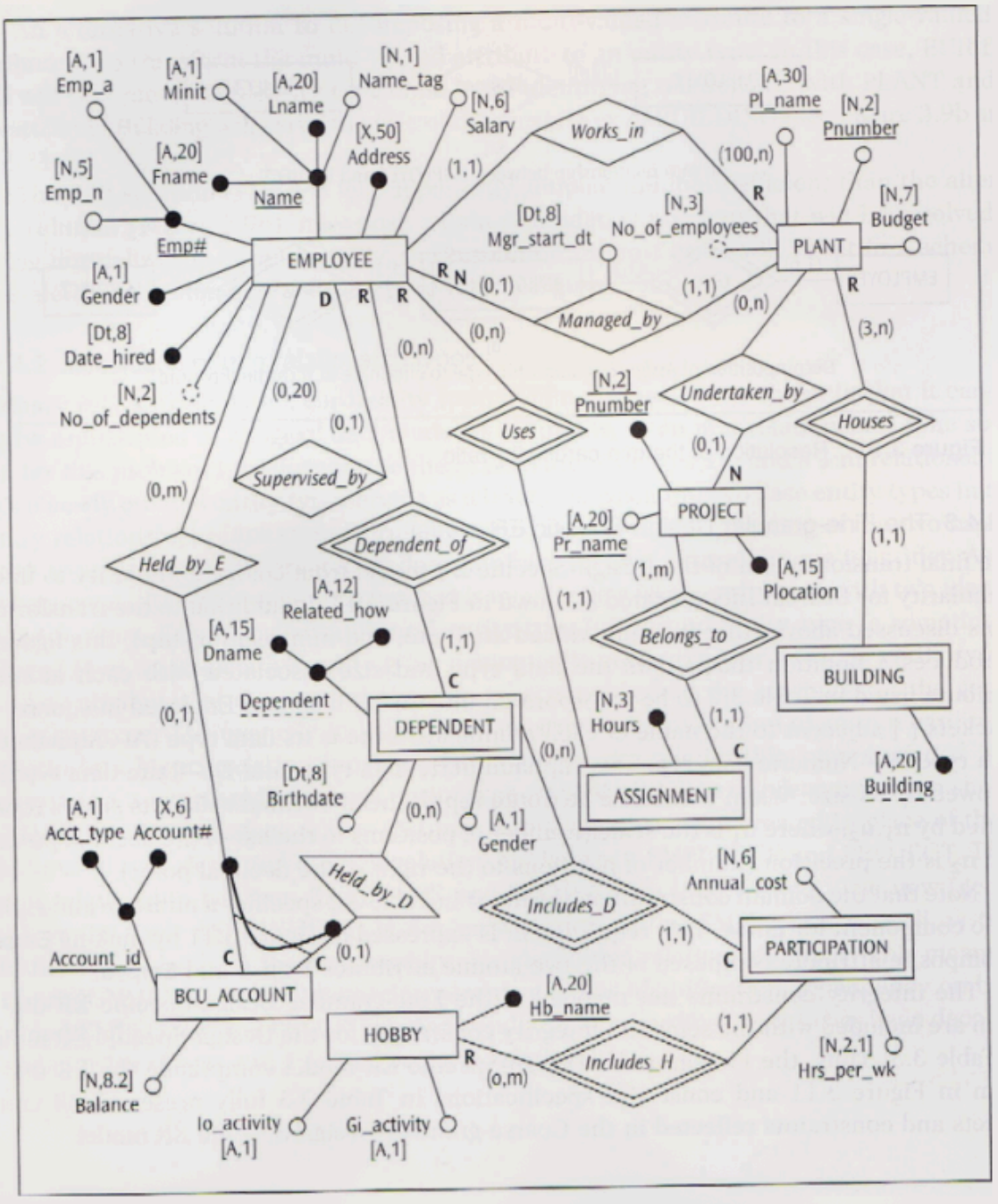


Figure 3.11 Fine-granular Design-Specific ER diagram for Bearcat Incorporated

- **EER Modeling (Enhanced Entity-Relationship) [PYQ Q3b, Q4a]**

- **Purpose:** Extends ER modeling with constructs for more complex applications (e.g., object modeling, AI). (p.119)
- **Superclass/Subclass (SC/sc) Relationship (Is-A Relationship):** (p.119-125) *(Fig 4.4, p.124; Fig 4.5, p.125)*

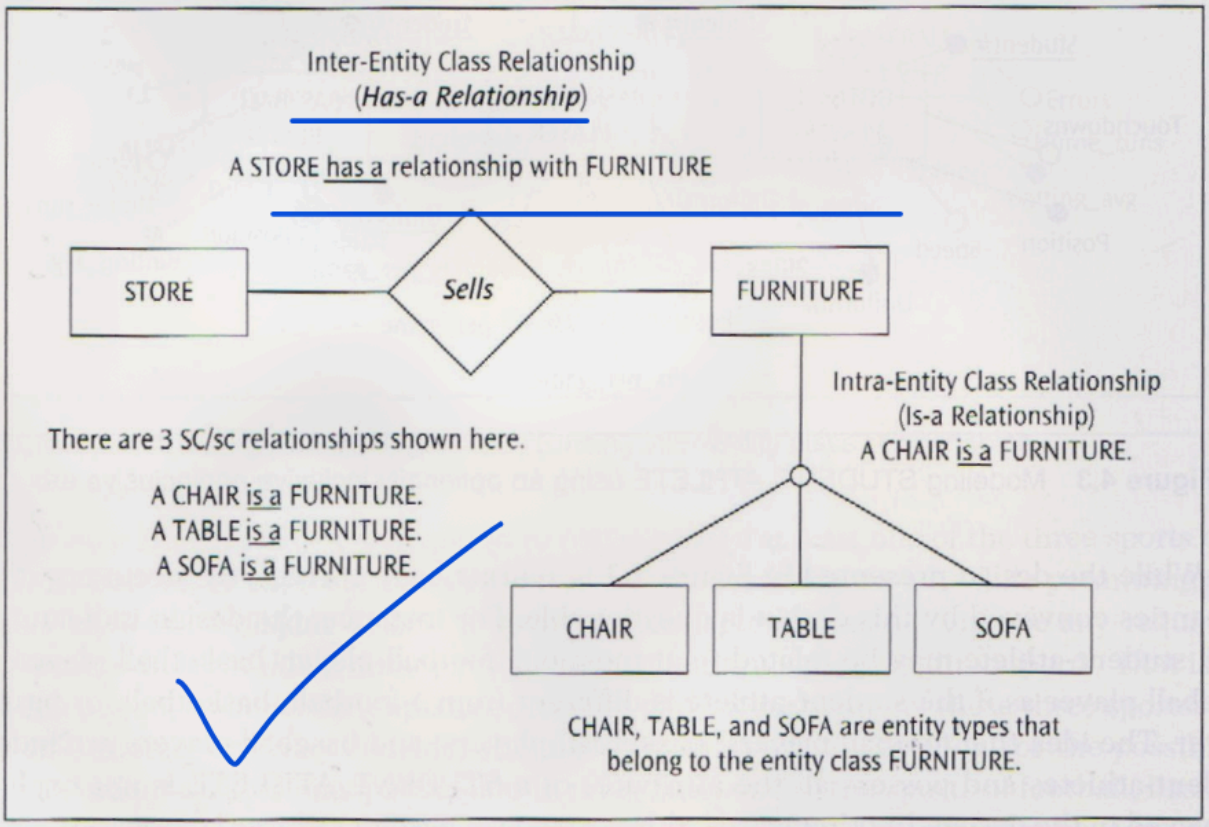


Figure 4.4 Inter-entity and intra-entity class relationships

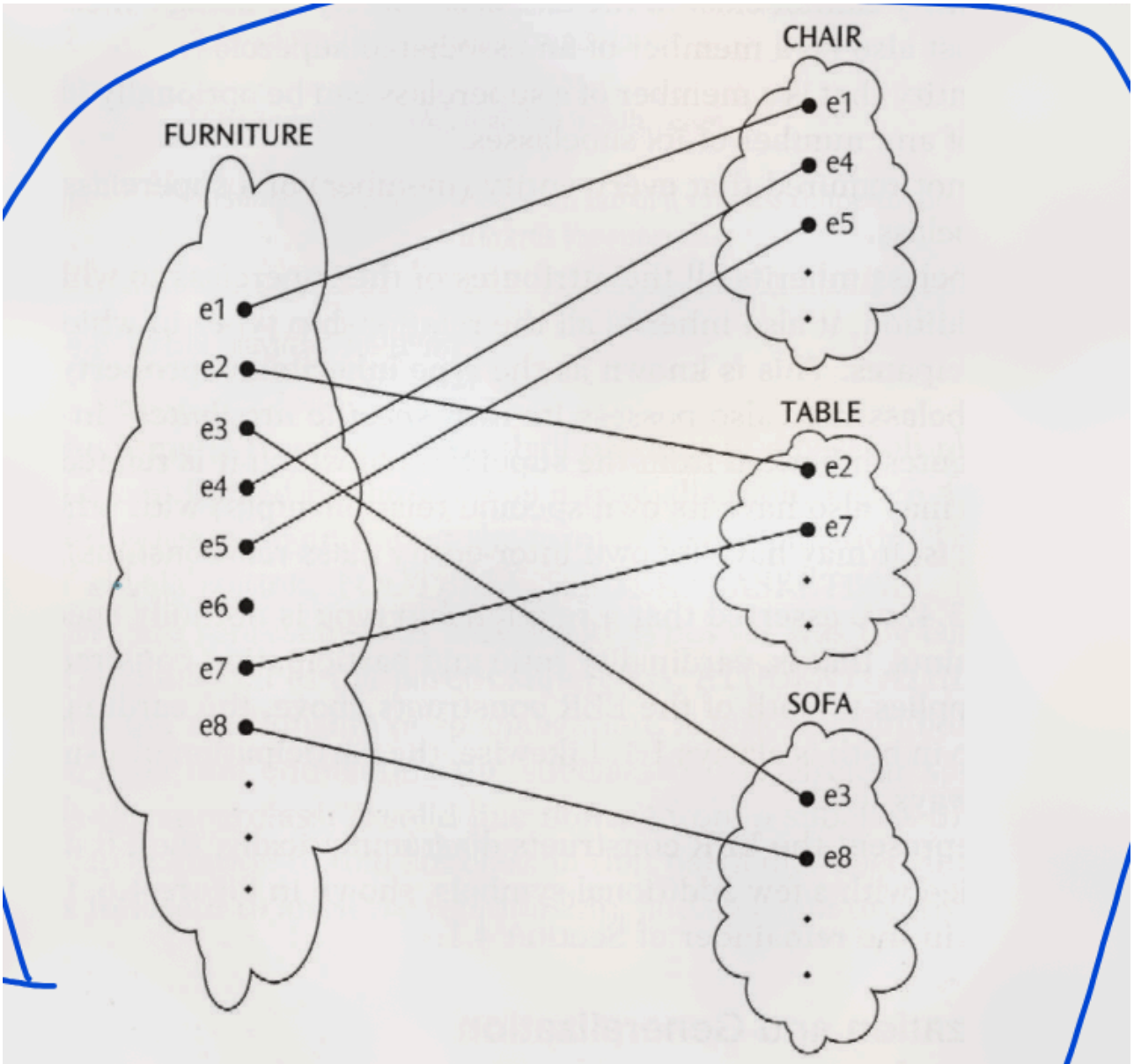


Figure 4.5 Superclass/subclass entity instances

- **Superclass (SC):** A generic entity type.
- **Subclass (sc):** A specialized entity type that inherits attributes and relationships from the superclass (Type Inheritance). Can have its own specific attributes/relationships.
- Cardinality is always 1:1 between an SC instance and its corresponding sc instance. Participation of sc in SC/sc is total.

- **Specialization and Generalization [PYQ Q3b]:** (p.125-133) Two perspectives of the same SC/sc relationship. *(Fig 4.6 notation, p.127; Fig 4.7 example, p.128)*

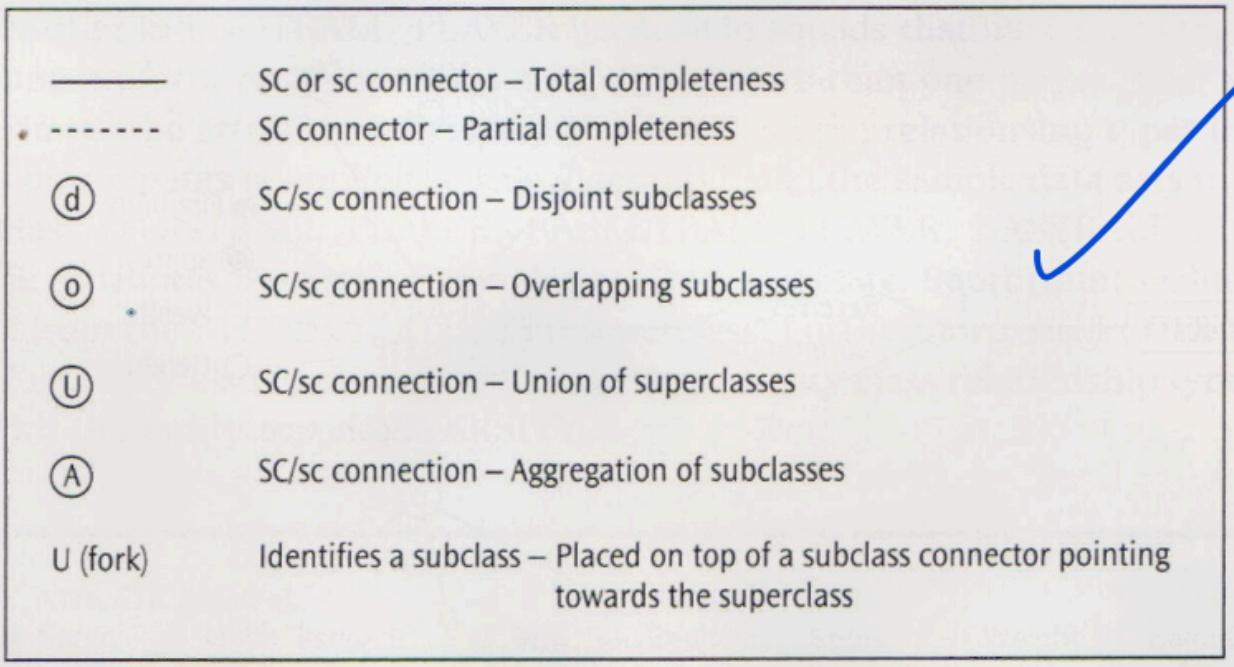


Figure 4.6 EER diagram notation

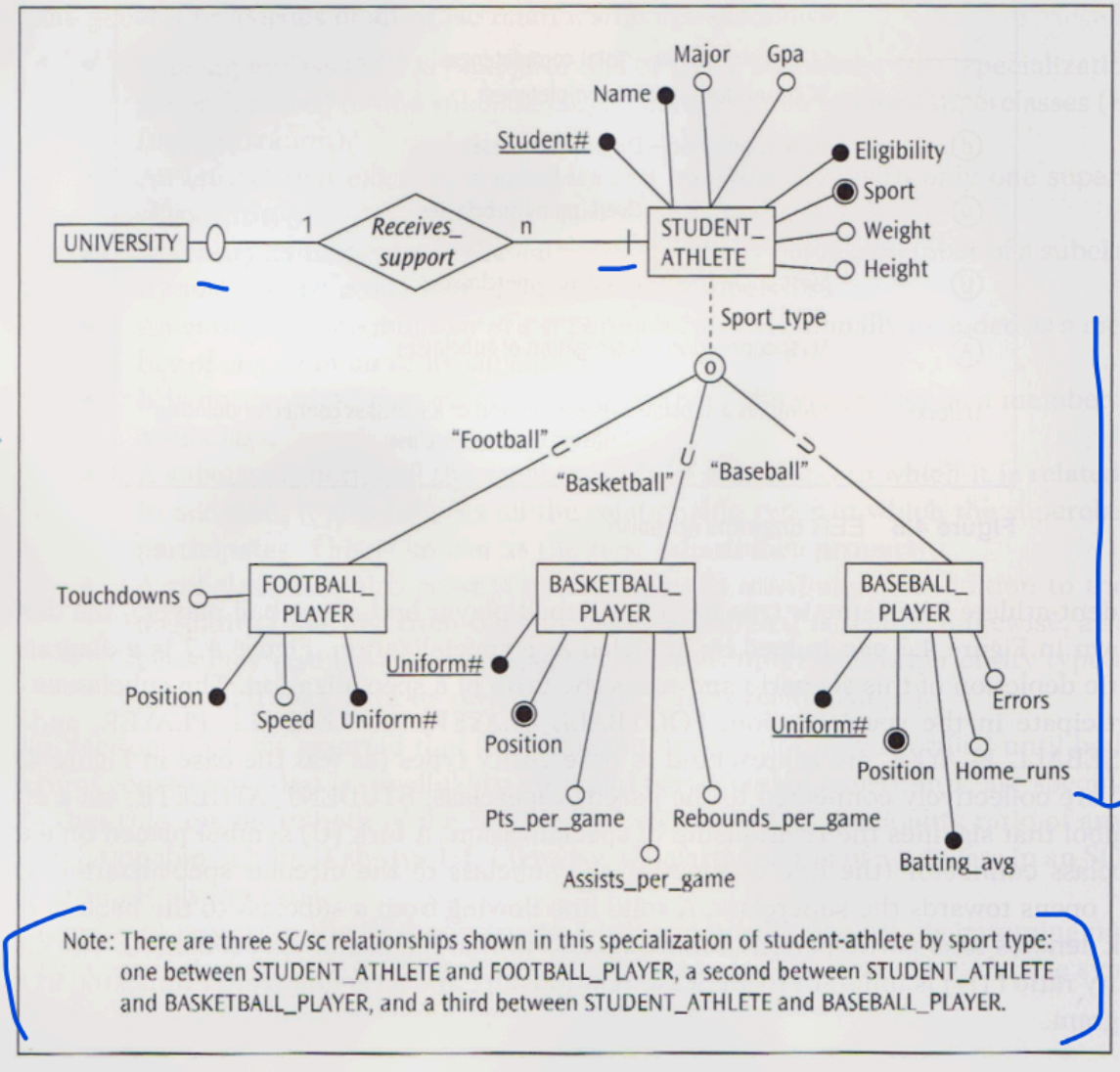


Figure 4.7 Modeling STUDENT_ATHLETE using intra-entity class relationships

- **Specialization:** Top-down; defining subgroups of an SC.

- **Generalization:** Bottom-up; identifying common features of entity types to form an SC.
- **Constraints:**
  - **Disjointness:**
    - **Disjoint (d):** An SC instance can be a member of at most one sc.
    - **Overlapping (o):** An SC instance can be a member of more than one sc.
  - **Completeness (Totalness):**
    - **Total (double line from SC to circle):** Every SC instance must be a member of some sc.
    - **Partial (single line from SC to circle):** An SC instance may not belong to any sc.
- **Predicate-defined (condition-defined) subclass:** Membership based on a condition on an SC attribute.
- **User-defined subclass:** Membership explicitly specified by user.

- **Hierarchy and Lattice [PYQ Q3b]:** (p.133-136)
  - **Specialization Hierarchy:** A subclass participates in only one SC/sc relationship as a subclass (tree structure). Inherits from its direct parent and ancestors. *(Fig 4.9, p.135)*

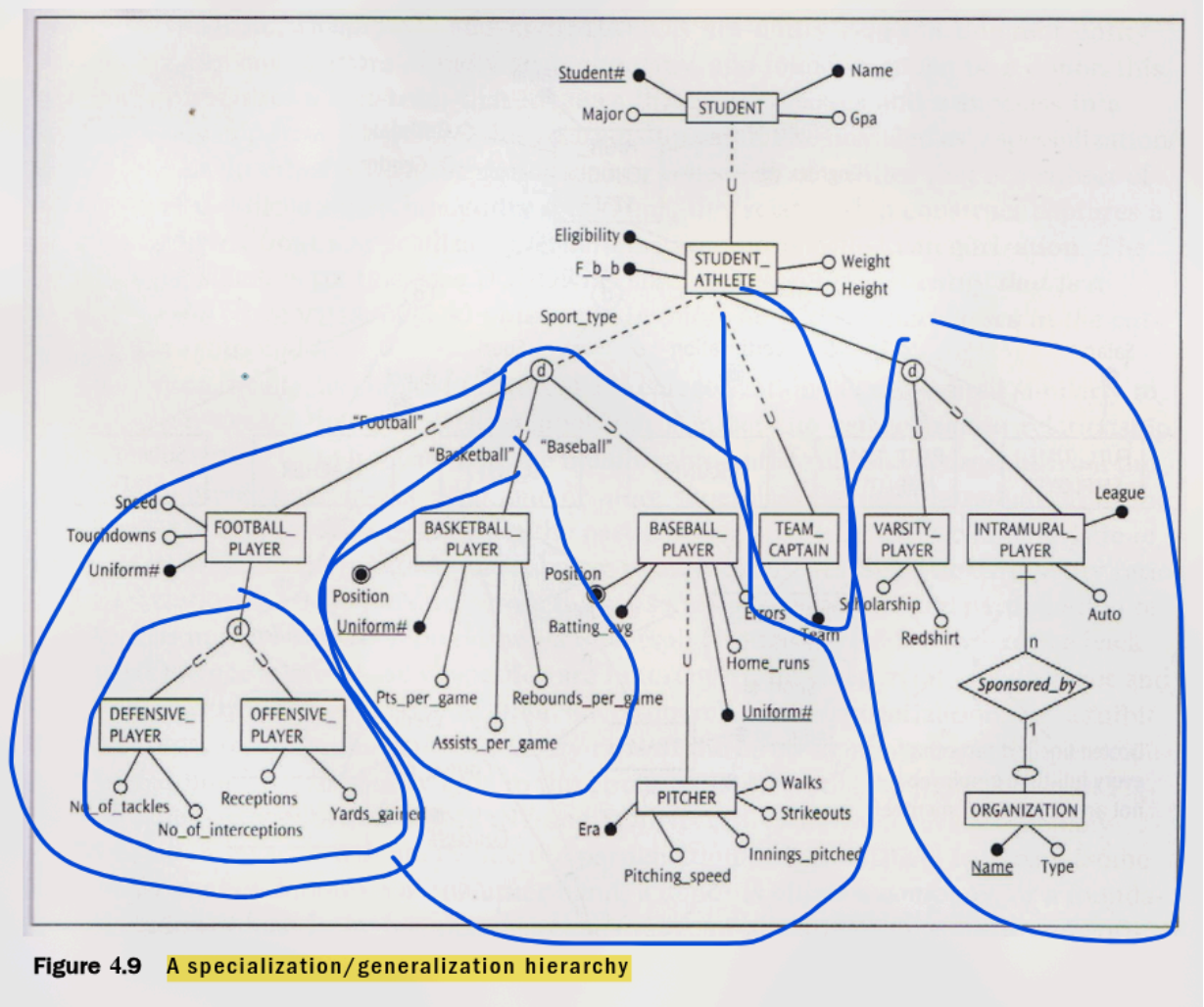


Figure 4.9 A specialization/generalization hierarchy

  - **Specialization Lattice (Multiple Inheritance):** A subclass (shared subclass) can be a subclass in more than one SC/sc relationship. Inherits from all its superclasses. *(Fig 4.10,*

p.136)

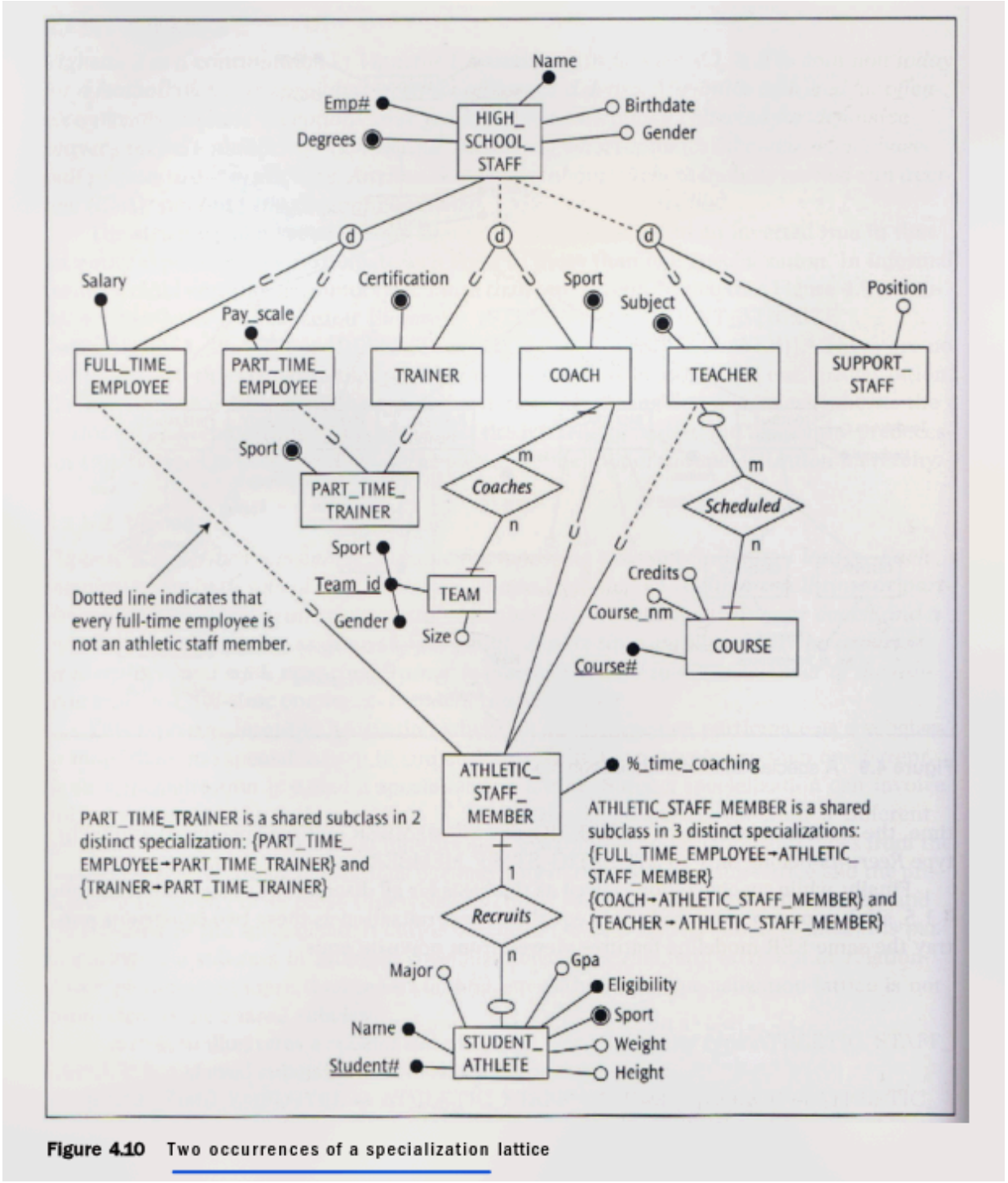


Figure 4.10 Two occurrences of a specialization lattice

- **Categorization [PYQ Q3b]:** (p.136-139) A subclass (category) is a subset of the union of two or more superclasses of different entity types. Represents a single SC/sc relationship type with

multiple SCs. Selective inheritance (inherits from only one of its SCs). *(Fig 4.11, p.138)*

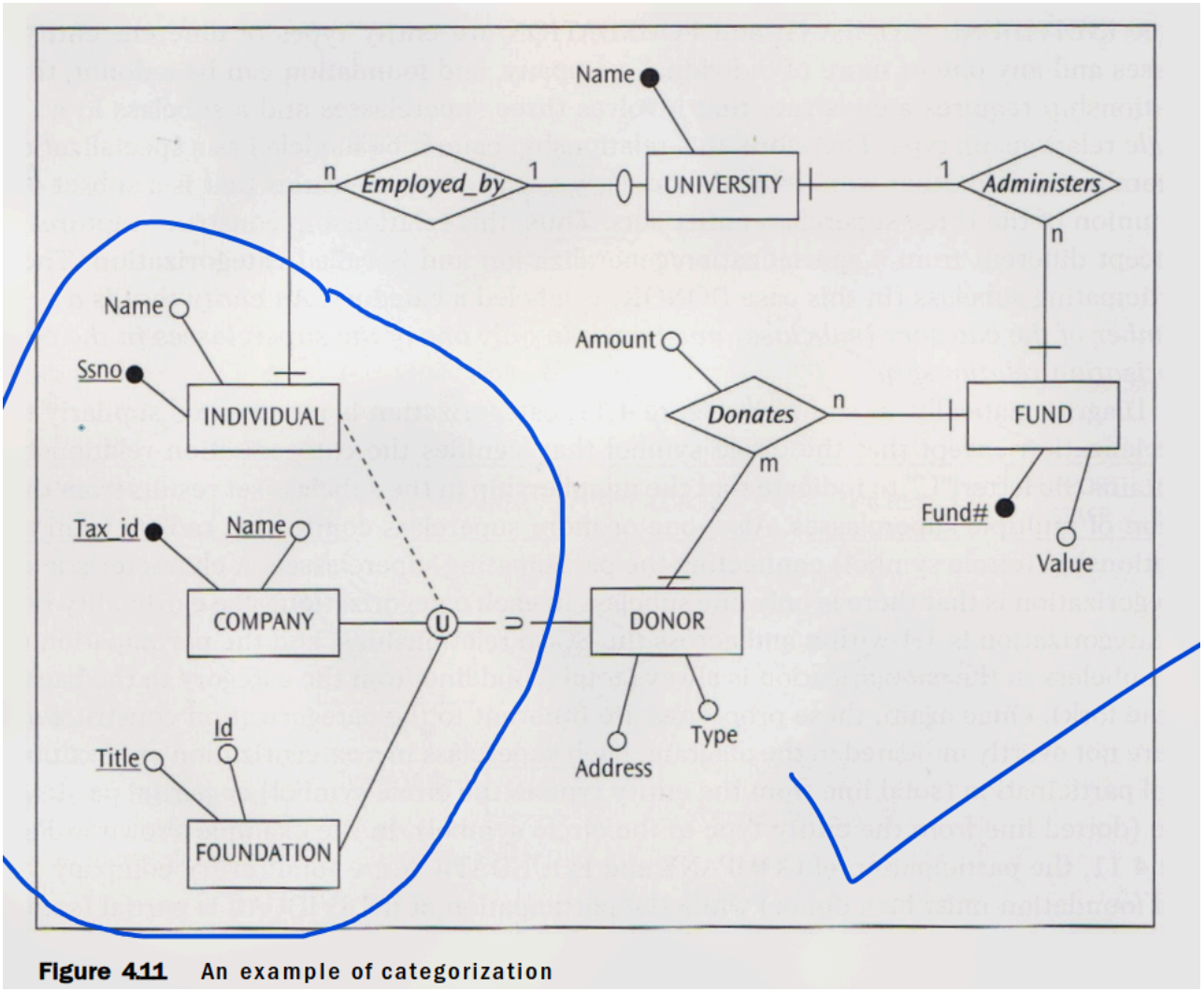
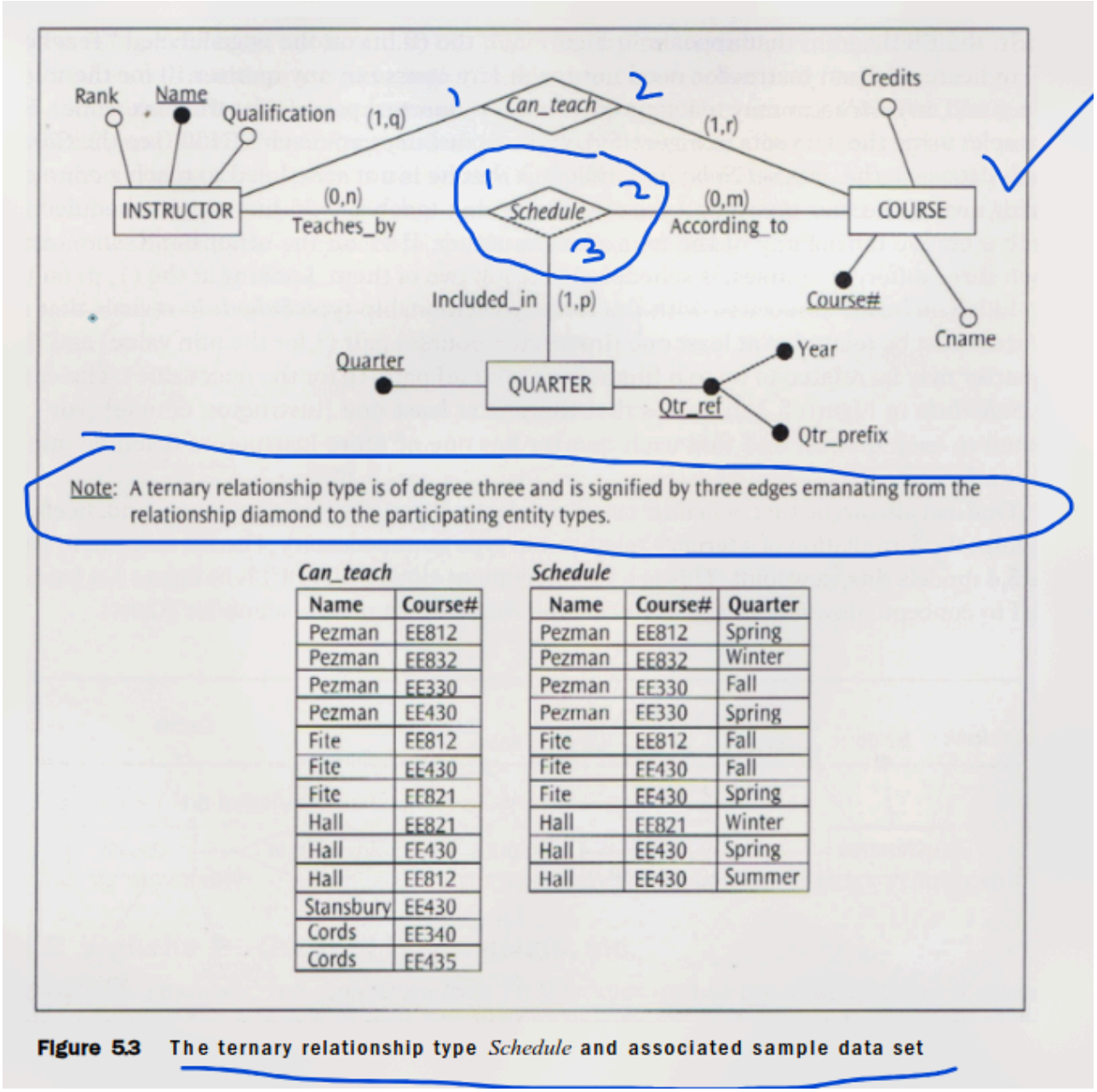


Figure 4.11 An example of categorization

- Indicated by 'U' in the circle. Total/Partial participation for SCs. Category participation is total.

- **Modeling Complex Relationships (Ch 5) [PYQ Q3b]**
  - **Ternary Relationship Type:** (p.164-169) Involves three entity types. (min, max) notation is crucial for precision. Can sometimes be conceptualized as a base entity. *(Fig 5.1-5.7, pp.164-*

169)



Can_teach

Name	Course#
Pezman	EE812
Pezman	EE832
Pezman	EE330
Pezman	EE430
Fite	EE812
Fite	EE430
Fite	EE821
Hall	EE821
Hall	EE430
Hall	EE812
Stansbury	EE430
Cords	EE340
Cords	EE435

Schedule

Name	Course#	Quarter
Pezman	EE812	Spring
Pezman	EE832	Winter
Pezman	EE330	Fall
Pezman	EE330	Spring
Fite	EE812	Fall
Fite	EE430	Fall
Fite	EE430	Spring
Hall	EE821	Winter
Hall	EE430	Spring
Hall	EE430	Summer

Figure 5.3 The ternary relationship type *Schedule* and associated sample data set

- **Beyond Ternary (Cluster Entity Type):** (p.171-179) Grouping of entity types and their relationships into a higher-level abstract entity (cluster). Useful for layered ERDs. *(Fig 5.10-5.14,*

pp.176-179; Fig 5.17, p.182)

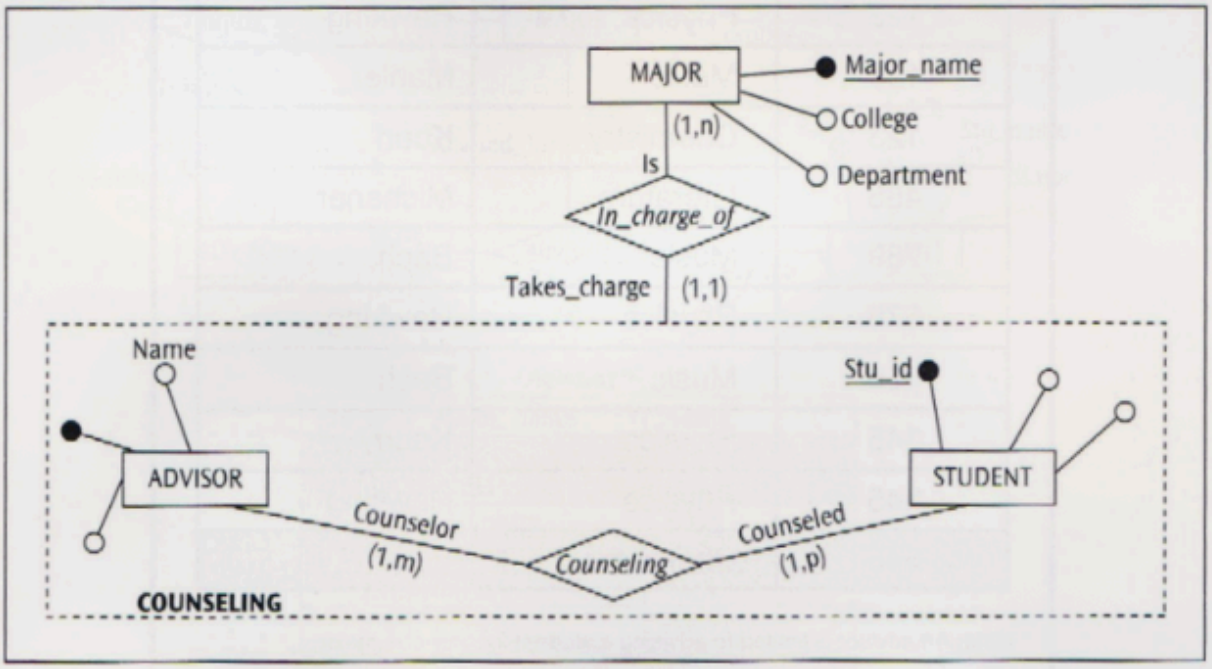


Figure 5.10 The relationship type *In_charge_of* in the cluster entity type COUNSELING

- **Weak Relationship Type (Inter-relationship Integrity Constraint):** (p.190-197) Existence of an instance in one relationship set depends on an instance in another. *(Fig 5.28-5.33, pp.191-195)*

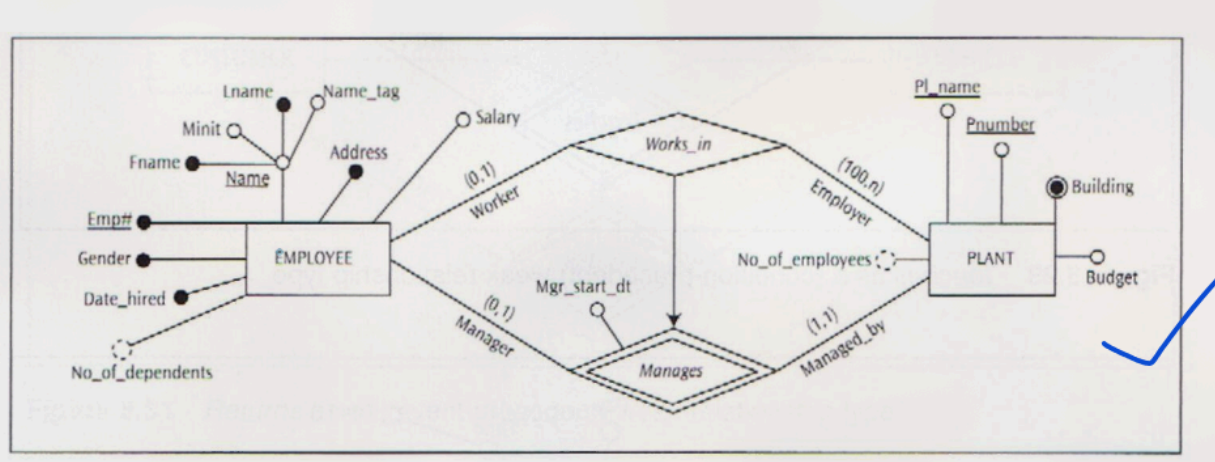


Figure 5.28 *Manages* as a (condition-precedent) weak relationship type

  - **Inclusion Dependency (Subset):** e.g., `Manages ⊆ Works_in`. (Solid arrow from dependent to precedent).
  - **Exclusion Dependency (Mutual Exclusion):** Equivalent to exclusive arc. (Dotted line connecting two weak relationships).

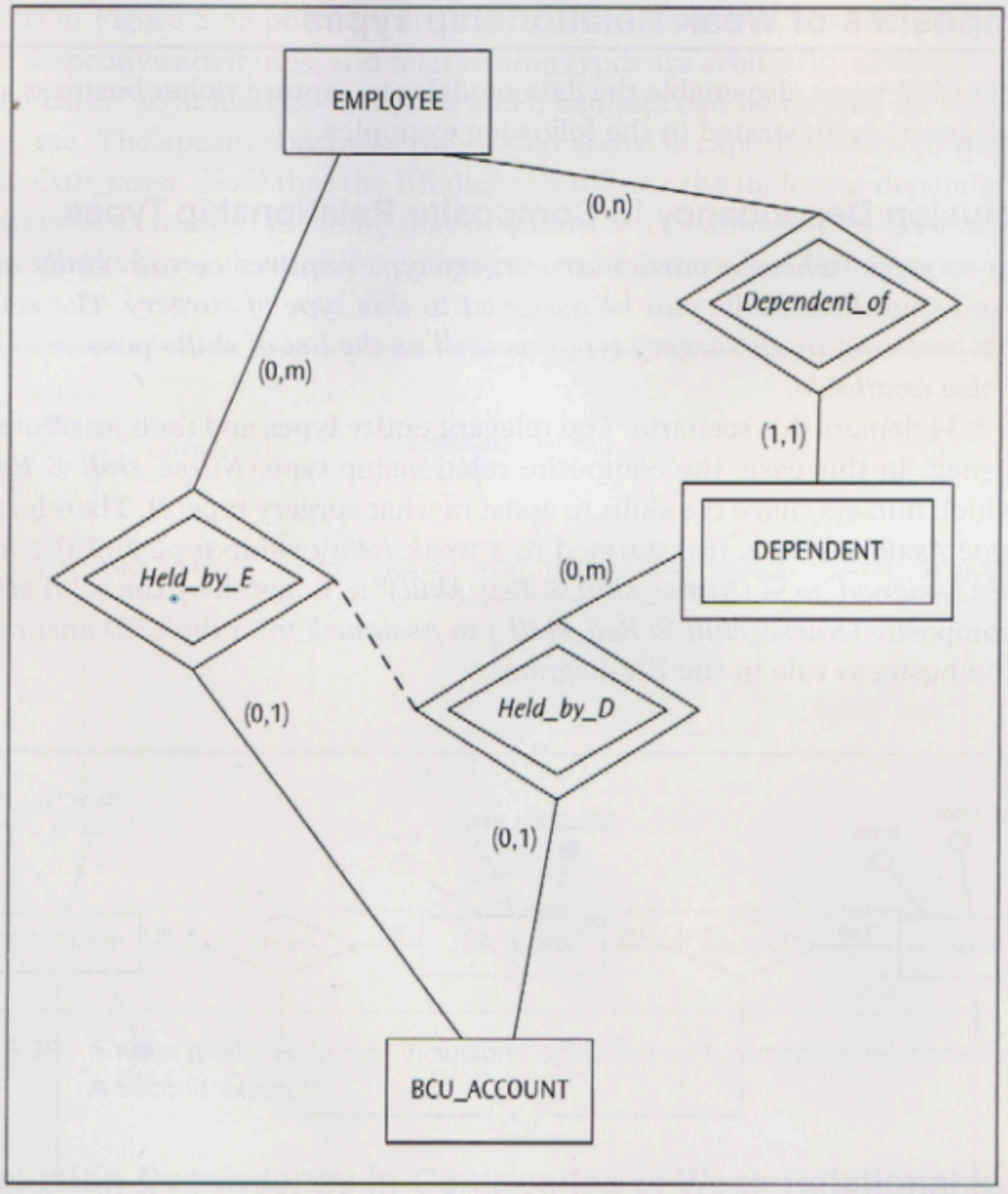


Figure 5.33 The exclusive arc of Figure 5.32 portrayed using exclusion dependency between weak relationship types

- **Composites of Weak Relationship Types:** (p.197-198) Combining weak relationships for richer semantics. *(Fig 5.34-5.37, pp.197-198)*

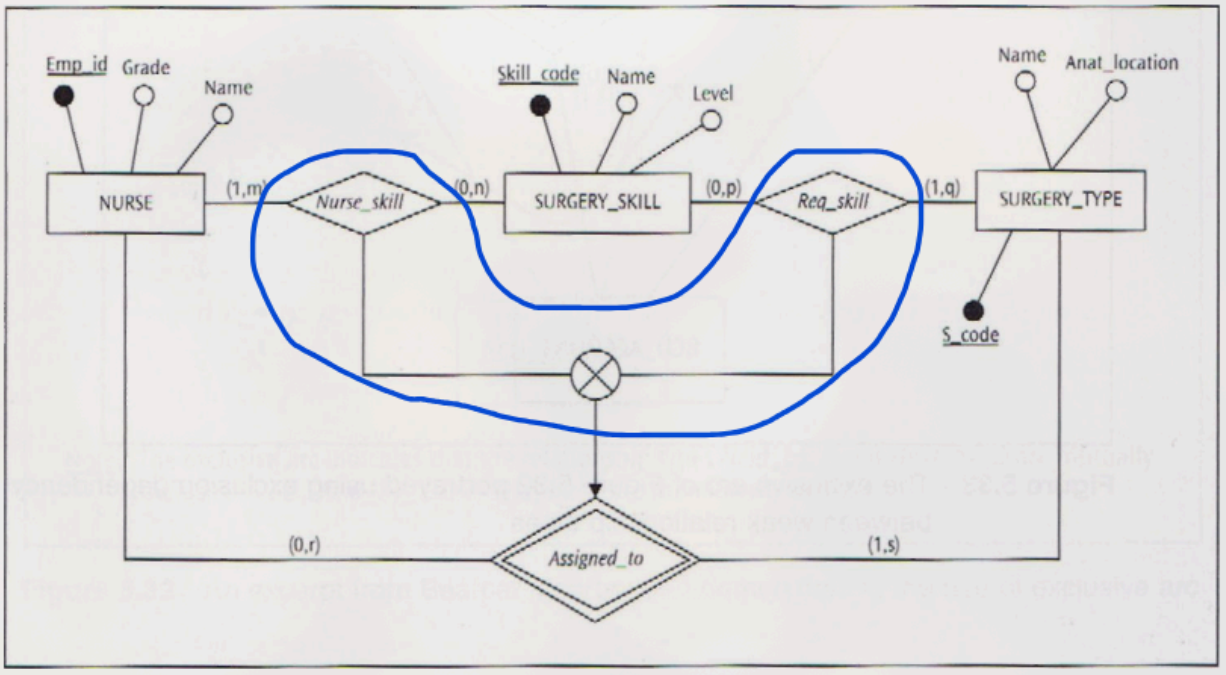


Figure 5.34 A weak relationship type inclusion-dependent on a composite relationship type

- **Design Issues in ER & EER Modeling [PYQ Q3b]**
  - Choosing entity vs. attribute.
  - Binary vs. higher-degree relationships.
  - Use of weak entities.
  - When to use EER constructs (Specialization/Generalization vs. Categorization).
  - Handling M:N relationships and multi-valued attributes (decomposition in Design-Specific ER).
  - **Validation of Conceptual Design (Connection Traps):** (p.209-216)
    - **Fan Trap:** Ambiguity when two or more 1:N relationships fan out from the same entity. *(Fig 5.48, p.211)*

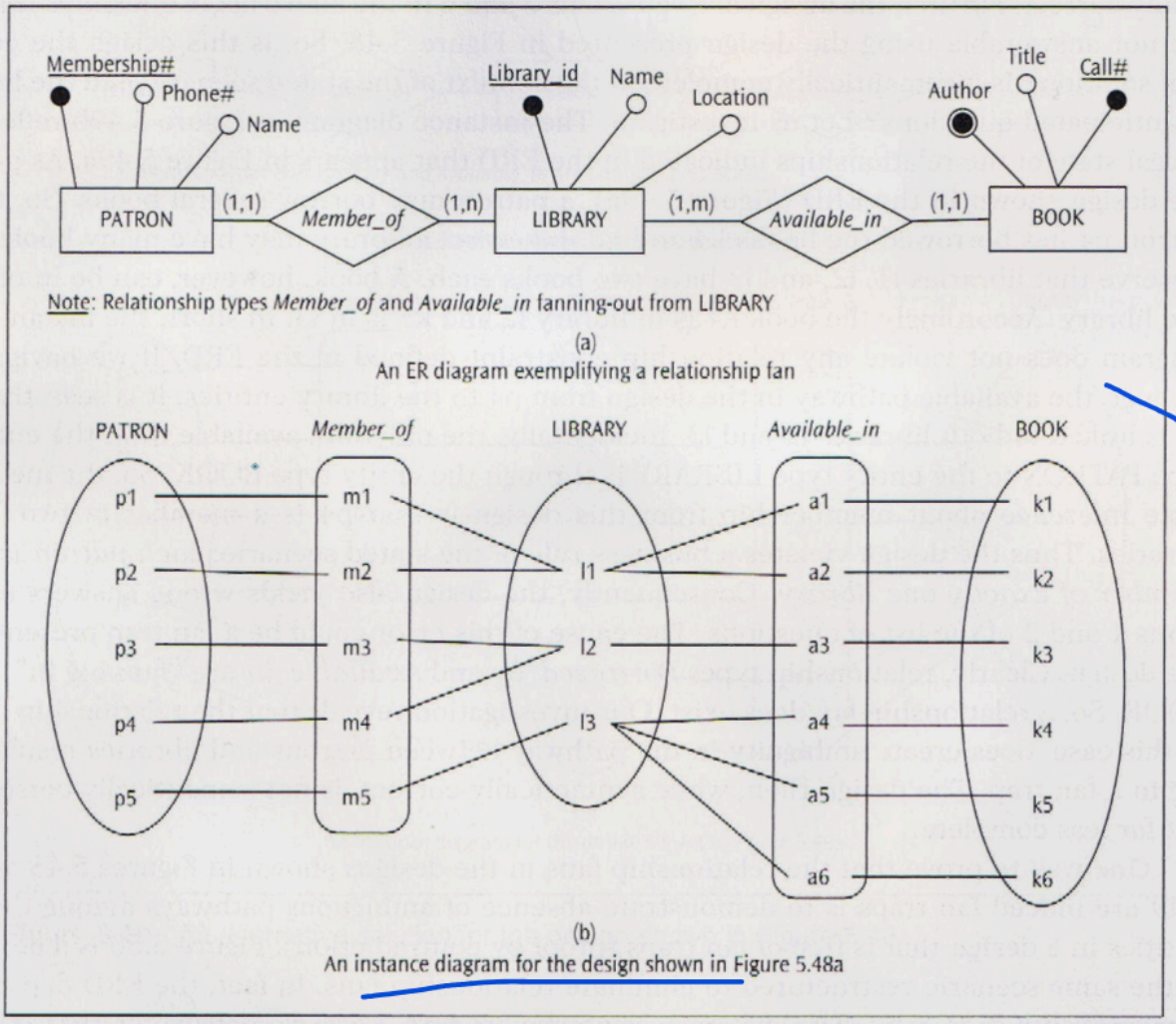


Figure 5.48 An example of a fan trap

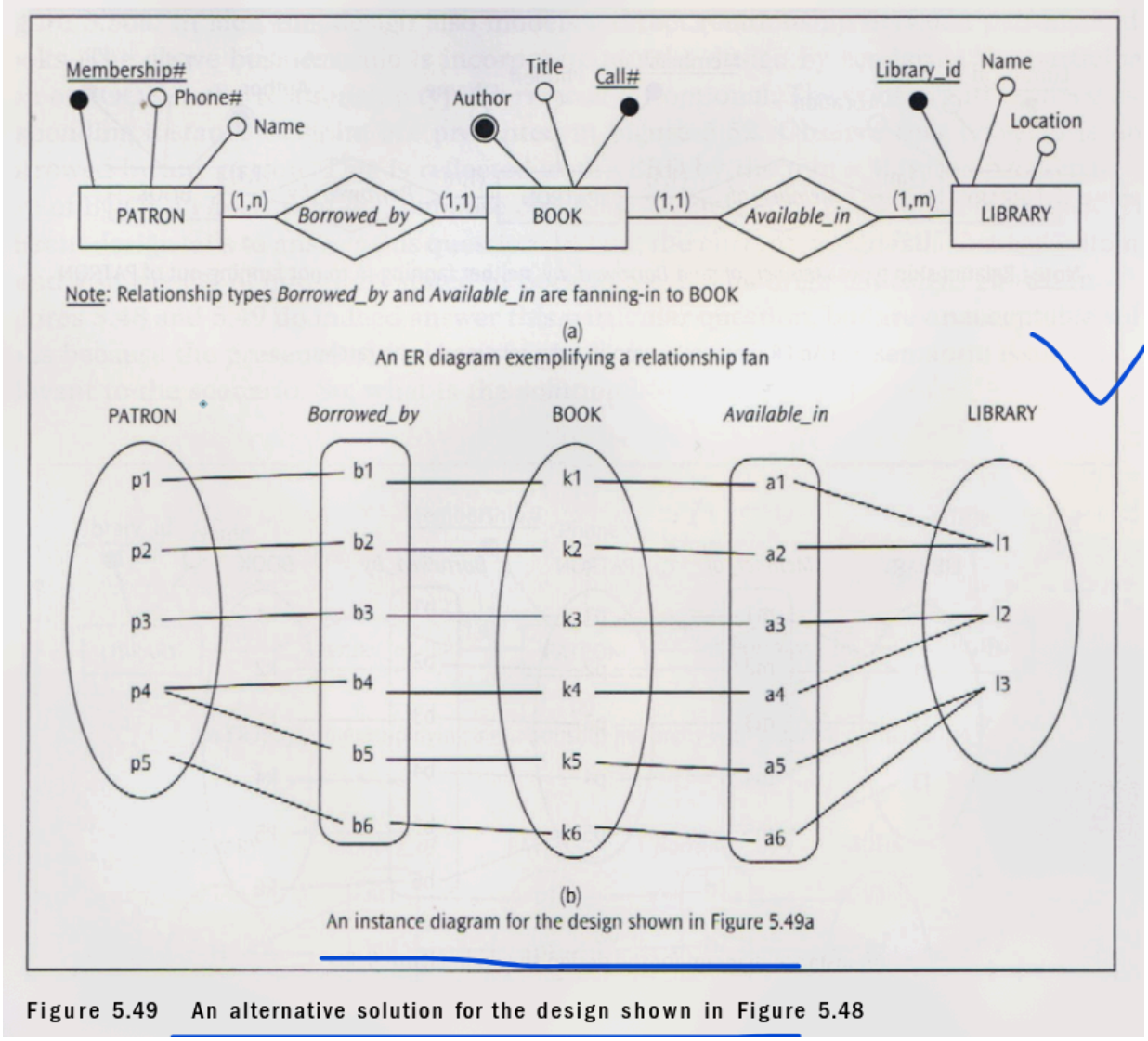


Figure 5.49 An alternative solution for the design shown in Figure 5.48

- **Chasm Trap:** Pathway seems to exist, but missing information (often due to optional participation) prevents joining related entities. *(Fig 5.52, p.215)*

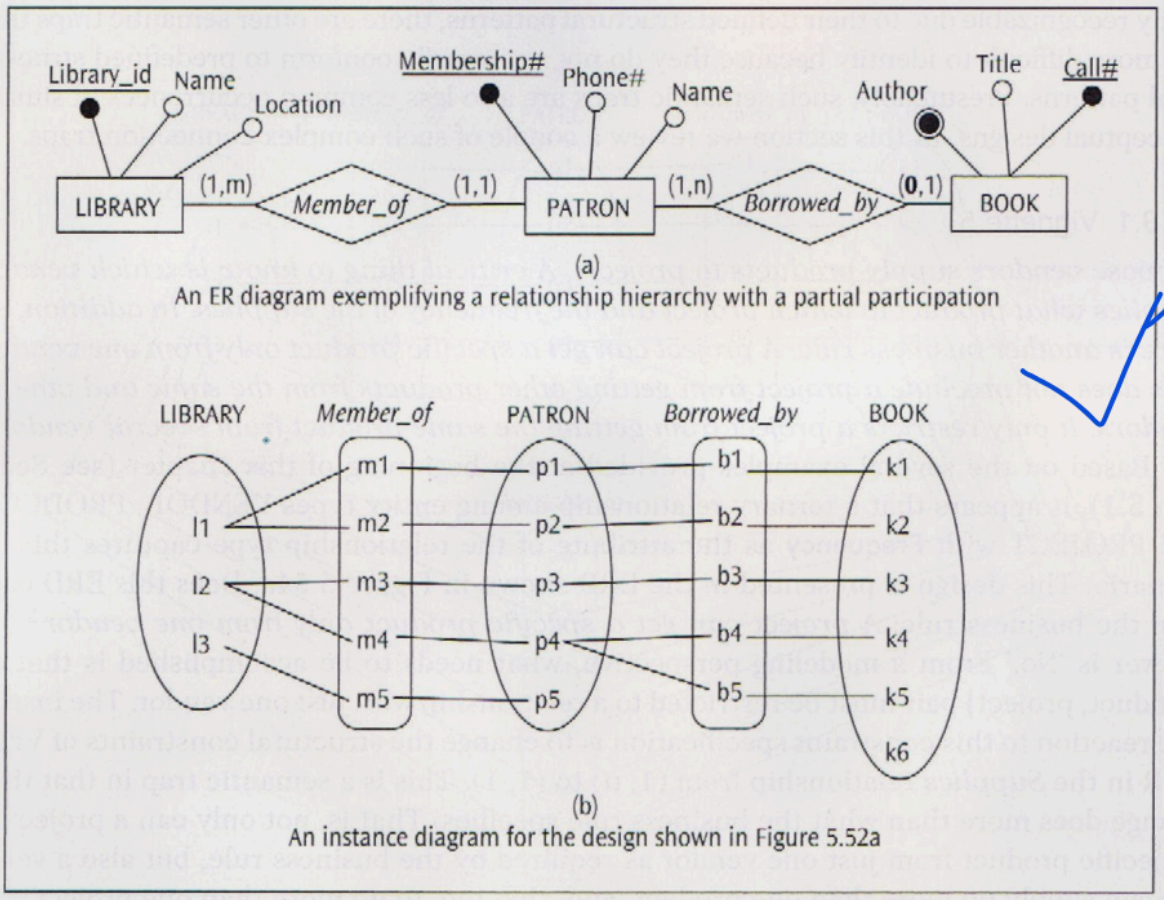


Figure 5.52 An example of a chasm trap

- Resolving traps may involve restructuring the ERD.

UNIT-II: LOGICAL DATA MODELLING

1. Overview of Relational Data Model

- **Definition and Terminology:** (p.244) *(Fig 6.1, p.246)*

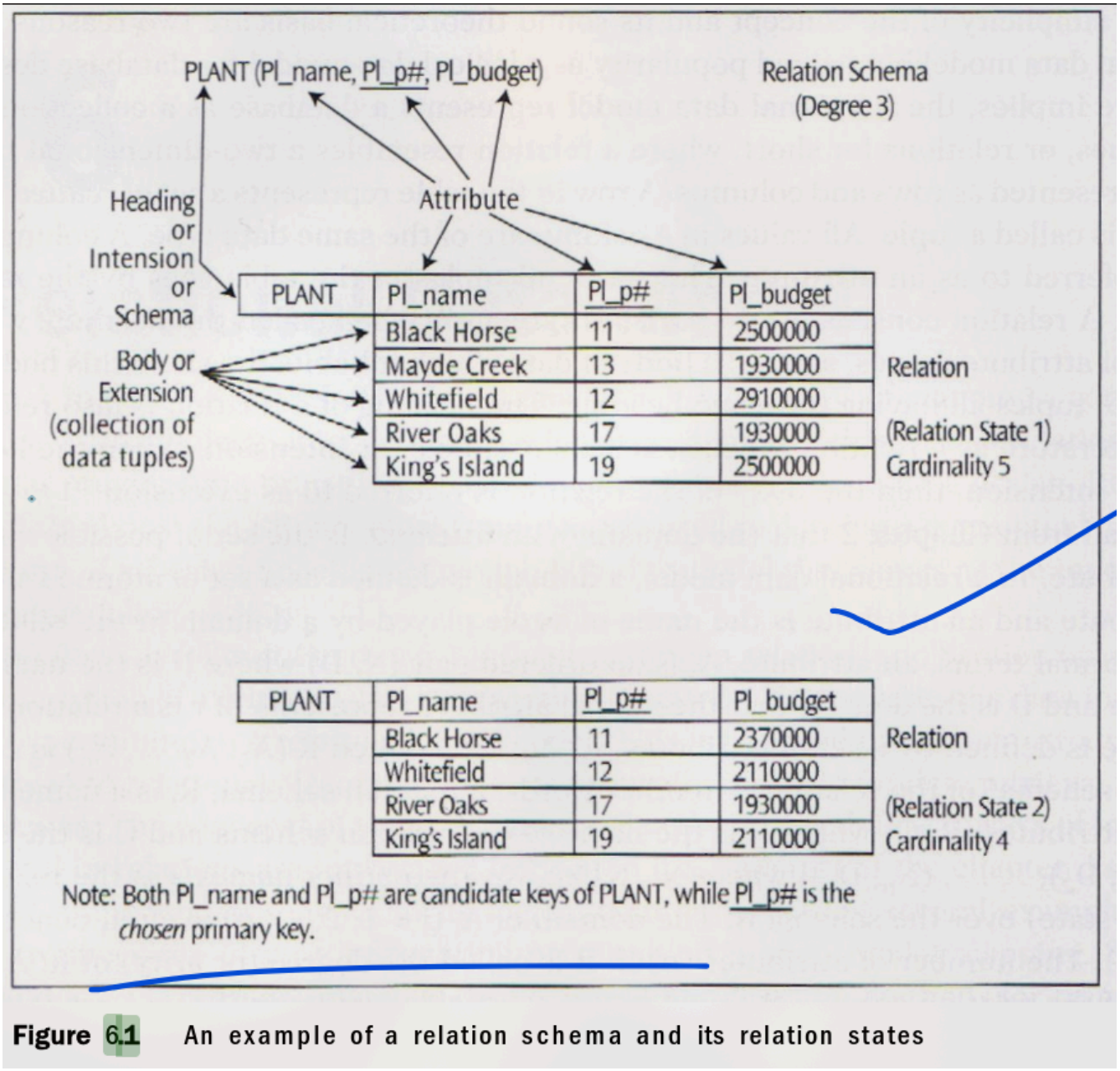
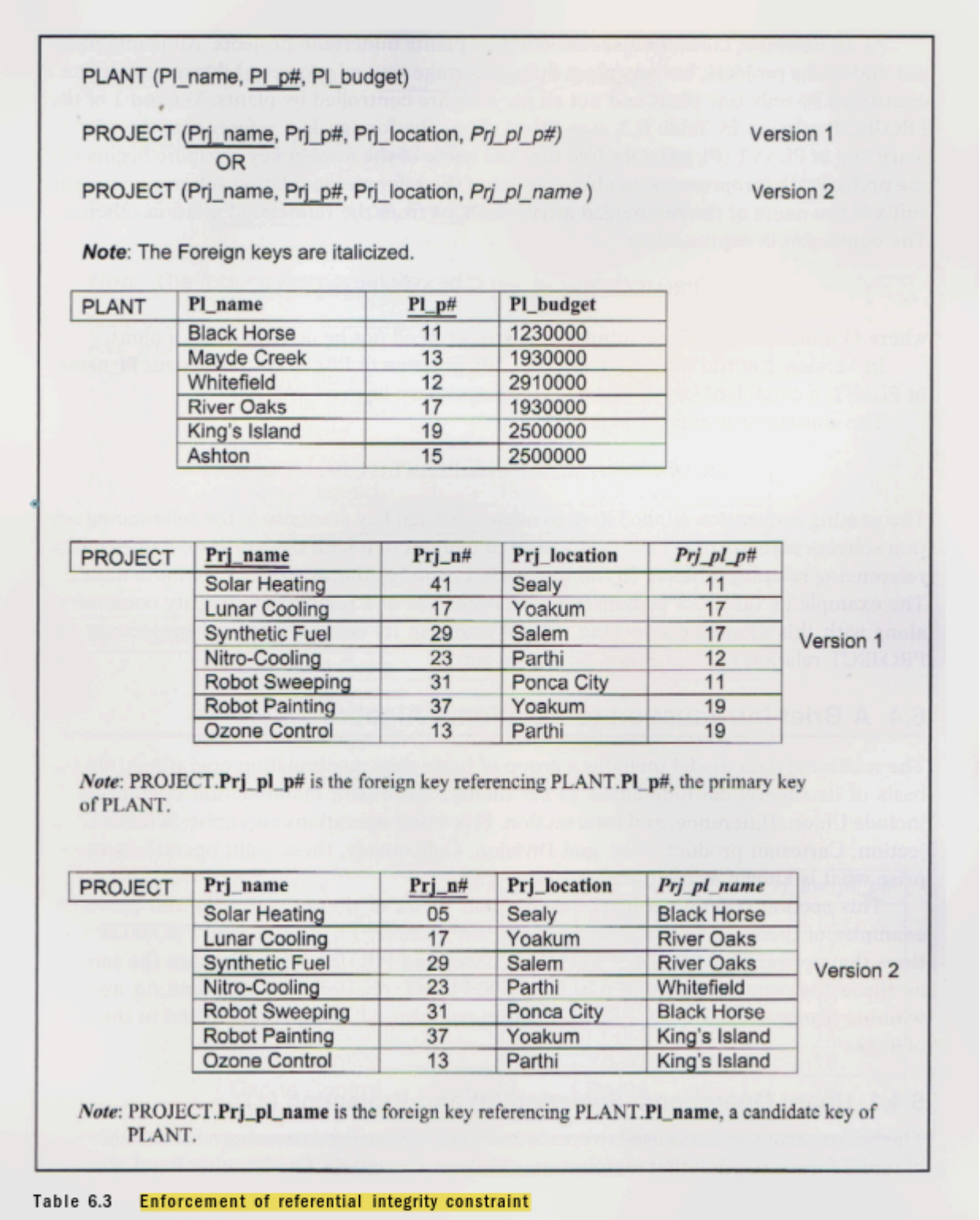


Figure 6.1 An example of a relation schema and its relation states

  - **Relation:** A two-dimensional table of data.
  - **Attribute (Column/Field):** A named column of a relation.
  - **Domain:** The set of allowable values for one or more attributes. Values are atomic.
  - **Tuple (Row/Record):** A row of a relation, representing a set of related data values.
  - **Relation Schema:** Name of the relation and its attributes (e.g., `PLANT(Pl_name, Pl_p#, Pl_budget)`). Heading or intension.
  - **Relation State (Instance):** A set of tuples for a given relation schema at a specific time. Body or extension.
  - **Degree:** Number of attributes in a relation.
  - **Cardinality:** Number of tuples in a relation.
- **Characteristics of a Relation:** (p.245-247)
  - Order of tuples is immaterial.
  - Order of attributes is immaterial (by name).

    - All attribute values are atomic (no composite or multi-valued attributes within a single cell - basis of 1NF).
    - Each tuple is distinct (no duplicate tuples).
- **Integrity Constraints [PYQ - Key concepts are important]** (p.247-254)
    - **Key Constraints [PYQ Q1b]:**
        - **Superkey:** An attribute or set of attributes that uniquely identifies a tuple within a relation.
        - **Candidate Key:** A minimal superkey (no proper subset is a superkey). A relation can have multiple.
        - **Primary Key:** The candidate key chosen to uniquely identify tuples. Underlined in schema.
        - **Alternate Key:** Candidate keys not chosen as primary key.
    - **Entity Integrity Constraint:** No primary key value can be null. Ensures each tuple is uniquely identifiable. (p.251)

- **Referential Integrity Constraint [PYQ Q1b]:** (p.252-254) *(Fig 6.3, p.253)*



PLANT (Pl_name, Pl_p#, Pl_budget)

PROJECT (Prj_name, Prj_p#, Prj_location, *Prj_pl_p#*) Version 1

OR

PROJECT (Prj_name, Prj_p#, Prj_location, *Prj_pl_name*) Version 2

Note: The Foreign keys are italicized.

PLANT	Pl_name	Pl_p#	Pl_budget
	Black Horse	11	1230000
	Mayde Creek	13	1930000
	Whitefield	12	2910000
	River Oaks	17	1930000
	King's Island	19	2500000
	Ashton	15	2500000

PROJECT	Prj_name	Prj_n#	Prj_location	*Prj_pl_p#*
	Solar Heating	41	Sealy	11
	Lunar Cooling	17	Yoakum	17
	Synthetic Fuel	29	Salem	17
	Nitro-Cooling	23	Parthi	12
	Robot Sweeping	31	Ponca City	11
	Robot Painting	37	Yoakum	19
	Ozone Control	13	Parthi	19

Version 1

Note: PROJECT.**Prj_pl_p#** is the foreign key referencing PLANT.**Pl_p#**, the primary key of PLANT.

PROJECT	Prj_name	Prj_n#	Prj_location	*Prj_pl_name*
	Solar Heating	05	Sealy	Black Horse
	Lunar Cooling	17	Yoakum	River Oaks
	Synthetic Fuel	29	Salem	River Oaks
	Nitro-Cooling	23	Parthi	Whitefield
	Robot Sweeping	31	Ponca City	Black Horse
	Robot Painting	37	Yoakum	King's Island
	Ozone Control	13	Parthi	King's Island

Version 2

Note: PROJECT.**Prj_pl_name** is the foreign key referencing PLANT.**Pl_name**, a candidate key of PLANT.

Table 6.3 Enforcement of referential integrity constraint

  - Ensures consistency among tuples in two relations.
  - **Foreign Key (FK):** An attribute or set of attributes in one relation (referencing relation) whose values must either match values of a candidate key (usually primary key) in another relation (referenced relation) or be wholly null.
  - Maintains relationships between tables.
  - Actions on violation: Restrict, Cascade, Set Null, Set Default.

2. Mapping ER Model to a Logical Schema [PYQ Q1d, Q4a] (p.261-277)

(Information-reducing mapping is the traditional approach)

* **Mapping Entity Types (Base and Weak):** (p.261) *(Fig 6.2, 6.3, p.262)*

* **Strong Entity:** Create a relation; choose a primary key. Include simple, atomic components of composite, and stored attributes.
* **Weak Entity:** Create a relation. Include attributes of the weak entity. The primary key is formed by the primary key of the owner entity (as FK) plus the partial key of the weak entity.
* **Mapping Relationship Types:** (p.262-265)
* **1:N (One-to-Many):**
* **Foreign Key Approach (Standard):** Include the primary key of the '1-side' (parent) entity as a foreign key in the relation of the 'N-side' (child) entity. Attributes of the relationship are also placed on the 'N-side'.
* **Cross-Referencing Design (Relationship Relation):** Create a new relation for the relationship. Its PK is the PK of the 'N-side' entity (or '1-side' if participation is optional on N-side and total on 1-side). Include PK of '1-side' as FK. Used if participation is optional on child side to avoid nulls. *(Fig 6.6, p.265)*

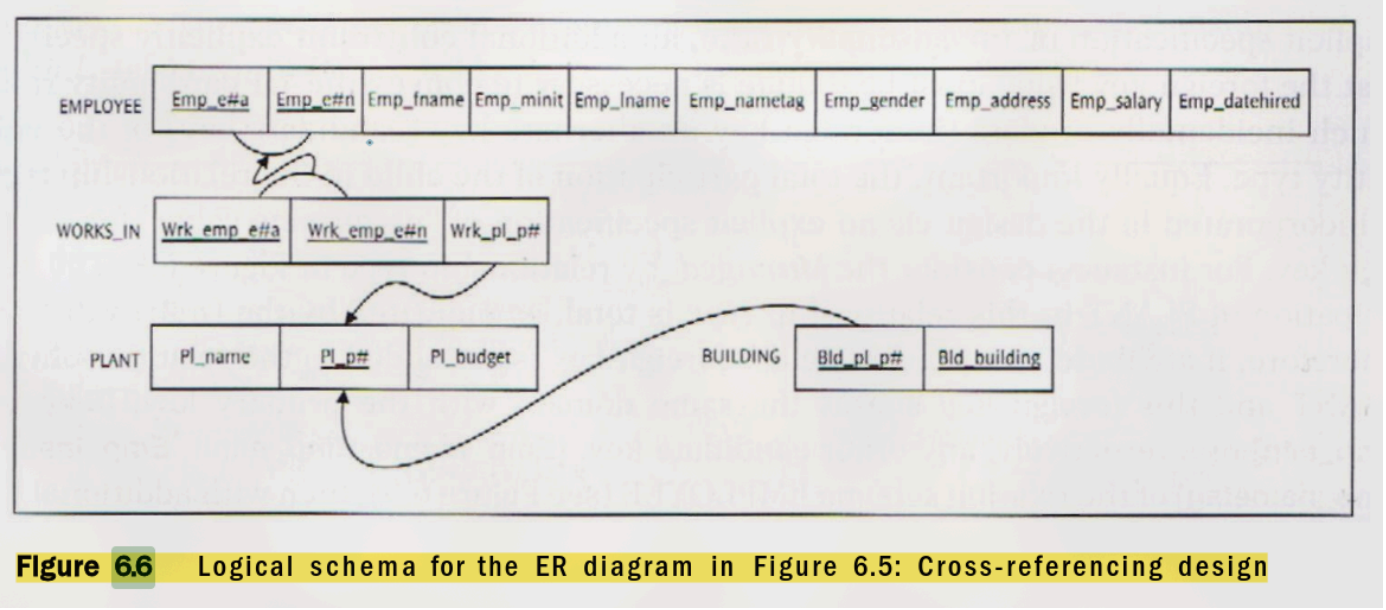


Figure 6.6 Logical schema for the ER diagram in Figure 6.5: Cross-referencing design

* **M:N (Many-to-Many):** Create a new relation (junction/associative table) for the relationship. The primary key of this new relation is the combination of the primary keys of the participating entities (both as FKs). Attributes of the relationship become attributes of this new relation.
* **1:1 (One-to-One):**
* **Option 1 (Foreign Key):** Choose one relation (e.g., if one side has total participation, make it the child), add PK of other as FK. FK must be unique.
* **Option 2 (Merged Relation):** If both have total participation, consider merging into one relation.
* **Option 3 (Relationship Relation/Cross-referencing):** If both partial, can create a separate relation. *(Fig 6.8-6.10 for 1:1 with total participation on one side, p.266-267; Fig 6.11-6.16 for 1:1 with partial participation, p.267-269)*
* **Recursive Relationship:**
* **1:N Recursive:** Add the PK of the entity as an FK in the same relation (with a different role name).
* **M:N Recursive:** Create a new relation with two FKs, both referencing the PK of the original entity (with different role names).
* **Mapping Multi-valued Attributes:** Create a new relation. PK of new relation includes PK of original entity (as FK) and the multi-valued attribute itself.
* **Mapping N-ary Relationships (Higher Degree) [PYQ Q5b]:** Create a new relation. PK is combination of PKs of all participating entities (as FKs).
* **Information-Preserving Mapping:** (p.277-281) A more detailed logical schema grammar that

attempts to retain more metadata (alternate keys, participation using (min,max), deletion rules) directly in the schema definition. *(Fig 6.23, p.280; Fig 6.24, p.282)*

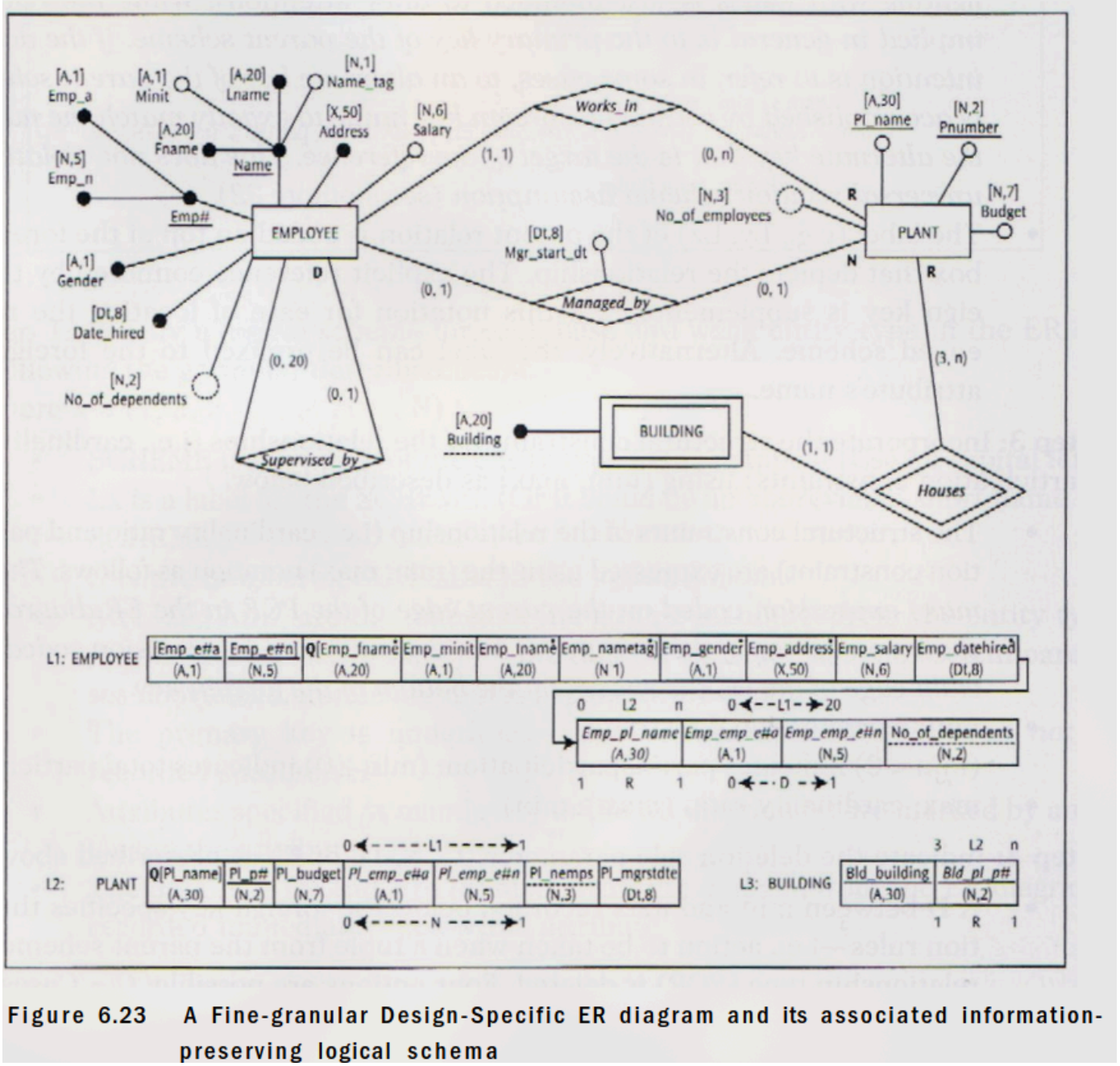


Figure 6.23 A Fine-granular Design-Specific ER diagram and its associated information-preserving logical schema

3. Mapping EER Model to a Logical Schema [PYQ Q4a] (p.281-295)

* **Mapping Specialization/Generalization:** *(Fig 6.25, 6.26, p.283-284)*

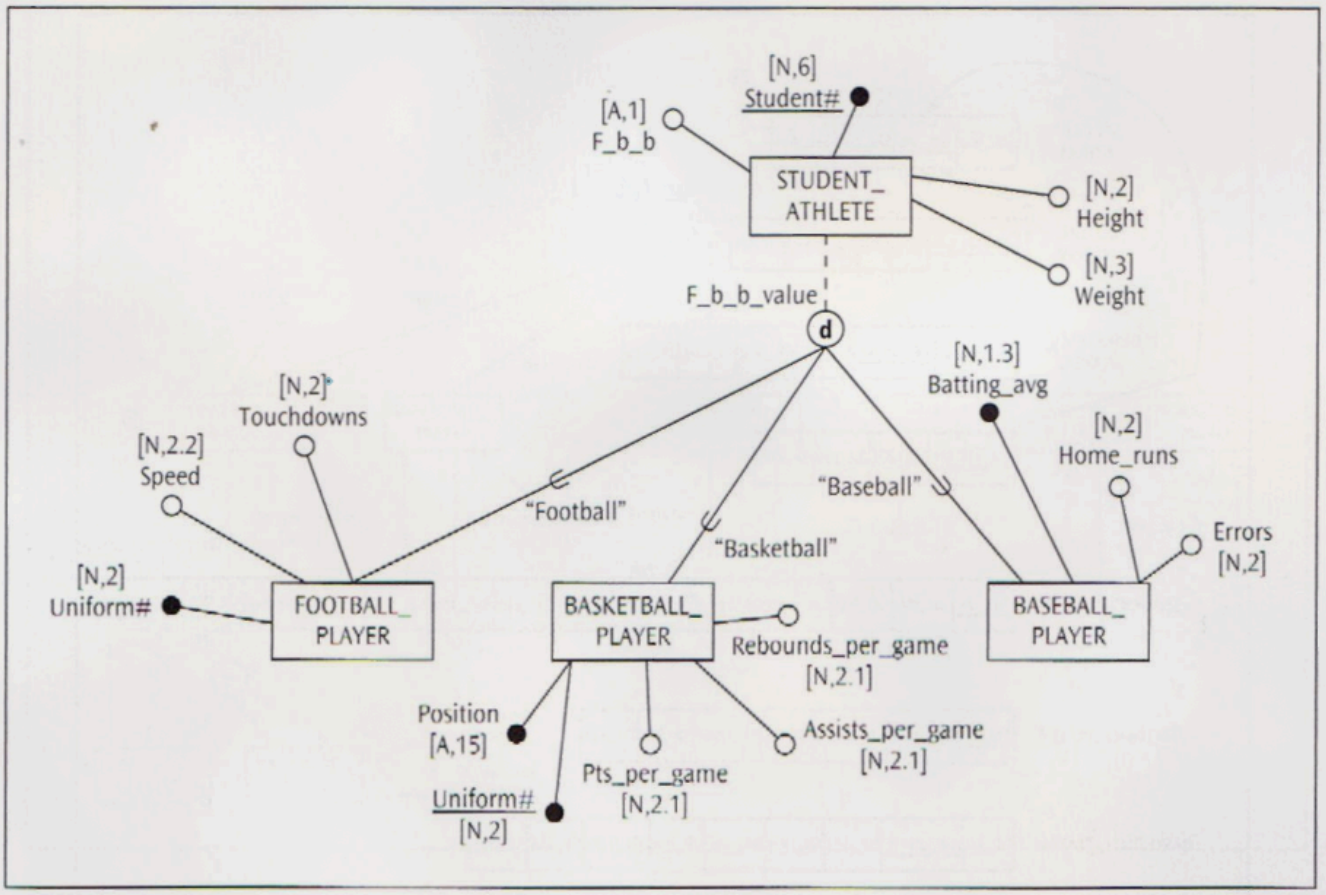


Figure 6.25 An example of a specialization

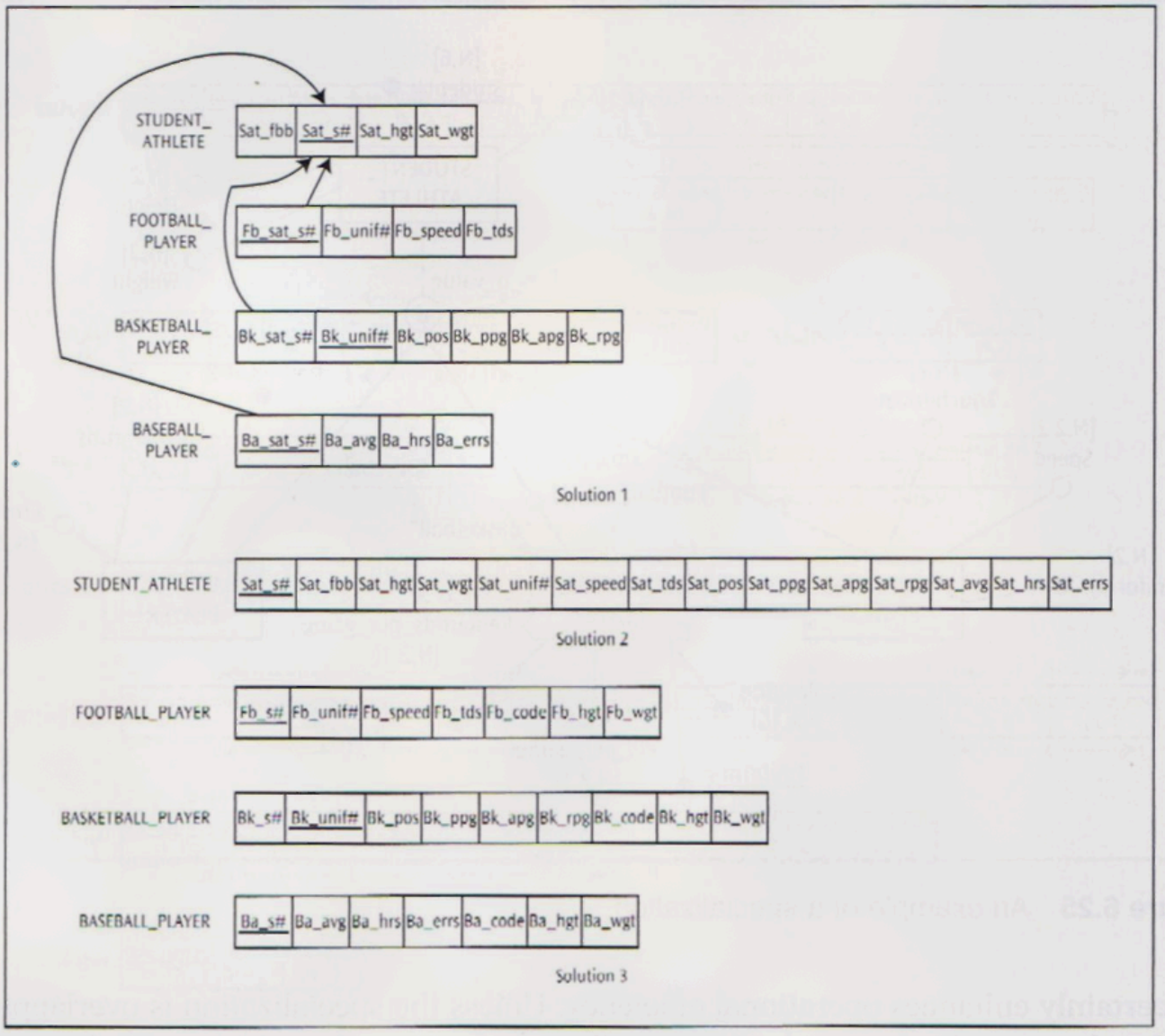


Figure 6.26 Three possible logical schemas for the specialization in Figure 6.25

* **Option 1 (Multiple Relations - SC and sc's):** Create a relation for SC and for each sc. PK of sc relations is same as SC's PK and acts as FK to SC. Good for disjoint subclasses, specific attributes/relationships for sc's.
* **Option 2 (Single Relation - SC only):** Create one relation for SC. Include attributes of all sc's. Use nulls for attributes not applicable to an instance. Add a type attribute to distinguish sc's. Good for overlapping sc's or when sc's have few specific attributes.
* **Option 3 (Multiple Relations - sc's only):** Create relation for each sc. Include inherited attributes from SC. Only if SC is total and disjoint. May lead to redundancy of SC attributes.
* **Mapping Specialization Hierarchy:** Apply chosen SC/sc mapping option recursively down the hierarchy. *(Fig 6.27, 6.28, p.285)*

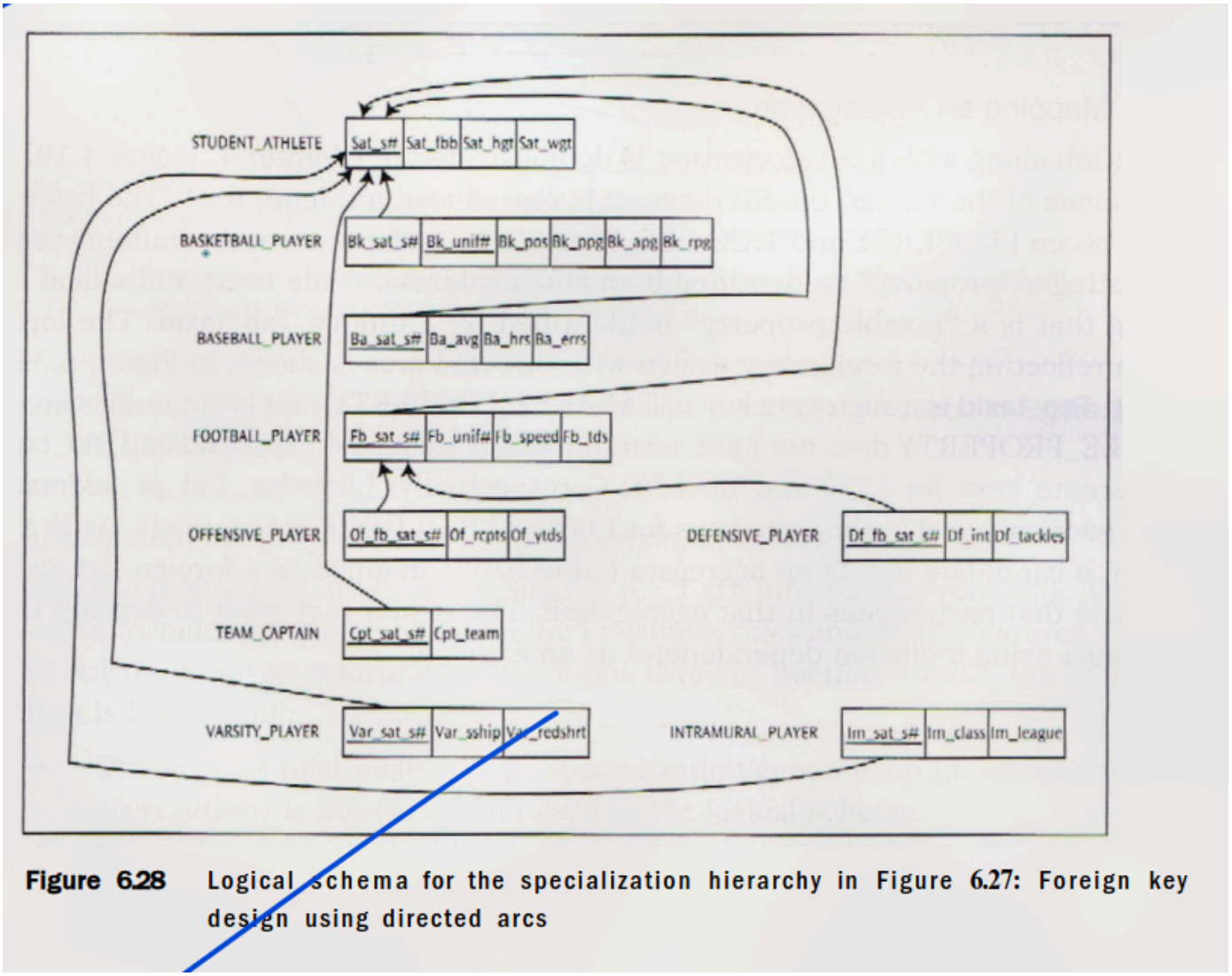


Figure 6.28 Logical schema for the specialization hierarchy in Figure 6.27: Foreign key design using directed arcs

* **Mapping Specialization Lattice (Shared Subclass):** The shared subclass relation will have multiple FKs, one for each SC/sc path it participates in, referencing its respective superclasses. *(Fig 6.29, 6.30, p.286-287)*

* **Mapping Categorization:** Create relation for the category (subclass). Create relations for superclasses. PK of category is usually a surrogate key. This PK is then included as FK in each superclass relation. (Note: This is different from specialization where sc inherits PK from SC). *(Fig 6.29, 6.30, p.286-287)*

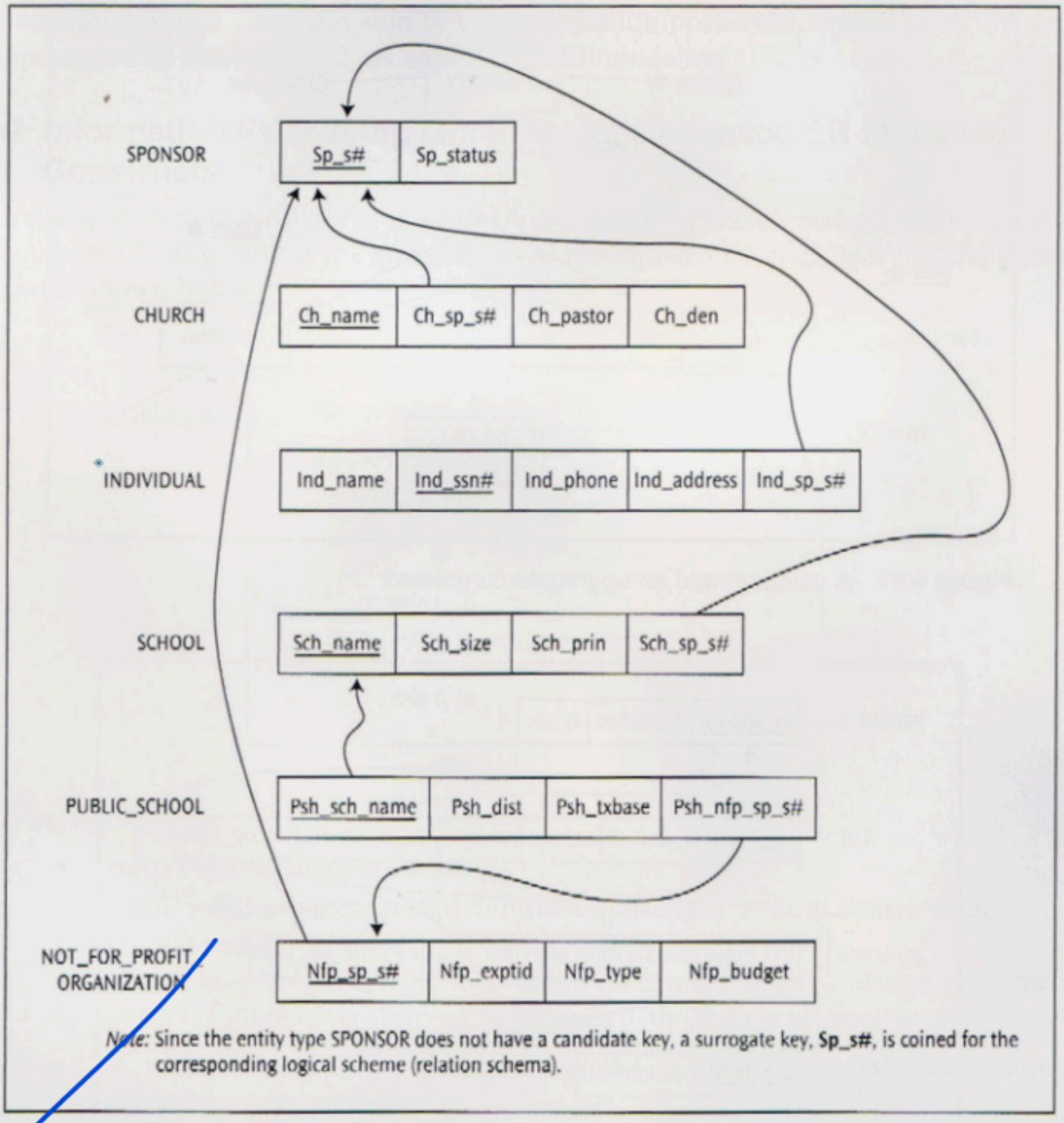


Figure 6.30 Logical schema for the specialization lattice and categorization in Figure 6.29: Foreign key design using directed arcs

* **Mapping Aggregation [PYQ Q5b]:** The "whole" (aggregate) relation includes PK of "part" superclasses as FKs. *(Fig 6.31, 6.32, p.288)*

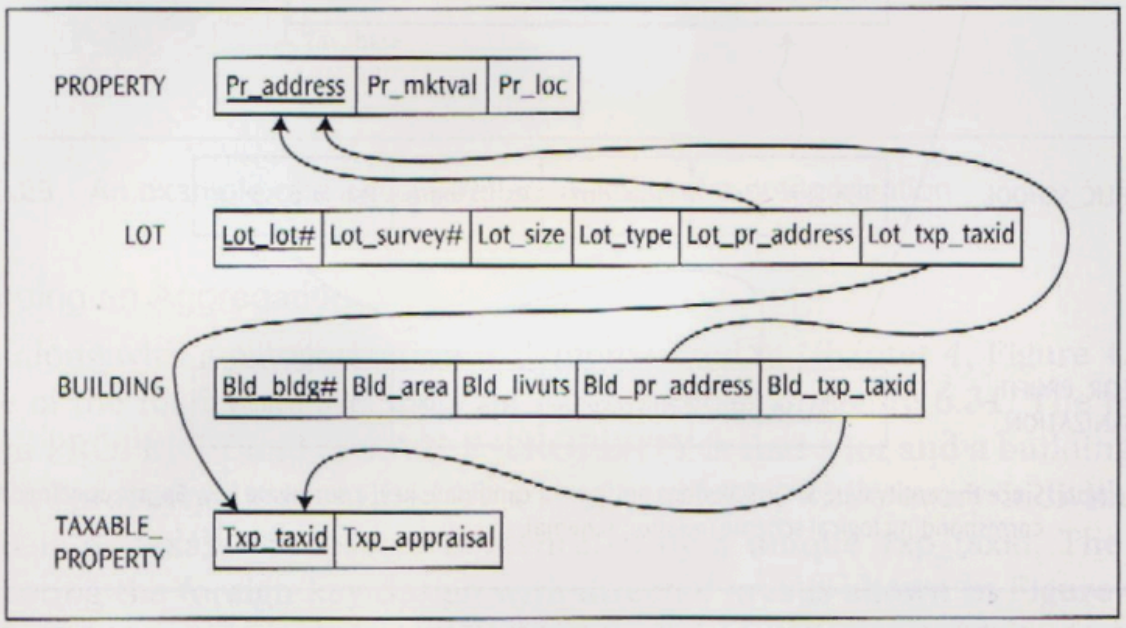


Figure 6.32 Logical schema for the category and aggregate in Figure 6.31: Foreign key design using directed arcs

* **Information Loss in traditional EER mapping:** Type of relationship (SC/sc vs. inter-entity), disjointness, multiple specializations might be lost. (p.287-288)
* **Information-Preserving Grammar for EER:** Extends the logical schema grammar to capture EER specific metadata (disjointness, # of sc's, SC/sc label). *(p.289-295, Fig 6.33-6.38, p.291-295)*

4. Mapping of Higher Degree Relationships [PYQ Q5b]

* (Covered in ER Mapping) Decompose into a gerund (associative) entity type in the conceptual model first (Ch 5.5.1, p.198-199, *Fig 5.38, 5.39, p.199-200*). Then map this gerund entity and its binary relationships to relational schema. The gerund entity becomes a relation whose PK is the combination of PKs of all originally participating entities.

5. Mapping of Aggregation [PYQ Q5b]

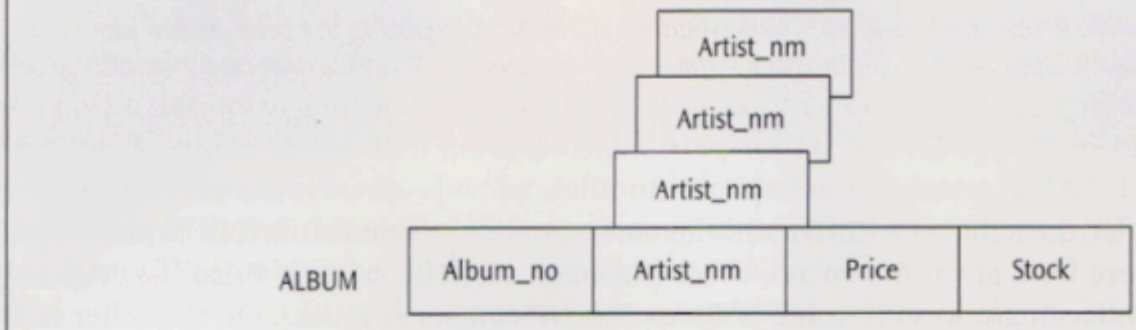
* (Covered in EER Mapping) The aggregate (subclass representing the "whole") is mapped to a relation. The PKs of the superclasses (representing the "parts") are included as foreign keys in the aggregate relation.

6. Mapping Complex ER Model Constructs to a Logical Schema

* **Cluster Entities:** (Ch 5.5.3, p.204-205) Decompose the cluster entity conceptually into a weak or gerund entity related to entities outside the cluster before mapping to relational.
* **Weak Relationships:** (Ch 5.5.4, p.206-208)
* **Inclusion Dependency:** If it doesn't introduce M:N, can be handled by careful FK placement and application-level checks or triggers. If it forms a structure like TEACHING $\subseteq$ CAN_TEACH, it can be mapped using specialization (TEACHING as sc of CAN_TEACH gerund). *(Fig 5.45-5.47, p.208)*
* **Exclusion Dependency:** Typically enforced by application logic or triggers, as direct relational constraints are limited.

7. Normalization [PYQ Q5a]

* **Introduction & Anomalies:** (Ch 8.1, p.345)
* **Normalization:** Process of organizing data in a database to reduce redundancy and improve data integrity. Involves decomposing relations into smaller, well-structured relations.
* **Anomalies (without normalization):**
* **Insertion Anomaly:** Difficulty in inserting data for one entity without data for another related entity.
* **Deletion Anomaly:** Unintended loss of data about one entity when data about another entity is deleted.
* **Modification/Update Anomaly:** Need to change redundant data in multiple places, leading to inconsistencies if not all are updated.
* **Normal Forms:**
* **1NF (First Normal Form):** (Ch 8.1.1, p.346) All attribute values must be atomic. No repeating groups or multi-valued attributes in a single cell. (A relation by definition is in 1NF). *(Fig 8.1, p.349)*



ALBUM

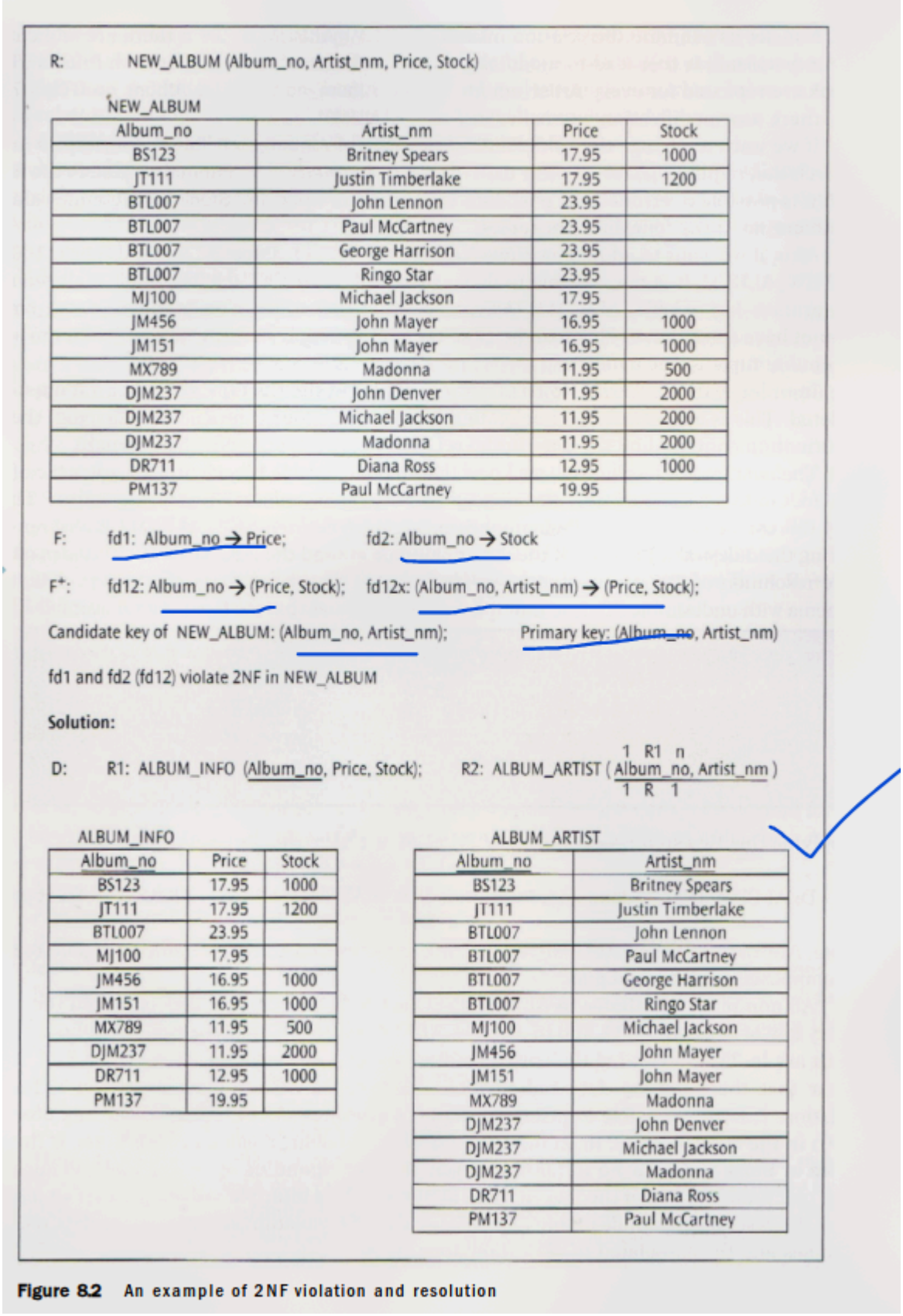
Album_no	Artist_nm	Price	Stock
BS123	Britney Spears	17.95	1000
JT111	Justin Timberlake	17.95	1200
BTL007	{John Lennon, Paul McCartney, George Harrison, Ringo Star}	23.95	
MJ100	Michael Jackson	17.95	
JM456	John Mayer	16.95	1000
JM151	John Mayer	16.95	1000
MX789	Madonna	11.95	500
DJM237	{John Denver, Michael Jackson, Madonna}	11.95	2000
DR711	Diana Ross	12.95	1000
PM137	Paul McCartney	19.95	

NEW_ALBUM

Album_no	Artist_nm	Price	Stock
BS123	Britney Spears	17.95	1000
JT111	Justin Timberlake	17.95	1200
BTL007	John Lennon	23.95	
BTL007	Paul McCartney	23.95	
BTL007	George Harrison	23.95	
BTL007	Ringo Star	23.95	
MJ100	Michael Jackson	17.95	
JM456	John Mayer	16.95	1000
JM151	John Mayer	16.95	1000
MX789	Madonna	11.95	500
DJM237	John Denver	11.95	2000
DJM237	Michael Jackson	11.95	2000
DJM237	Madonna	11.95	2000
DR711	Diana Ross	12.95	1000
PM137	Paul McCartney	19.95	

Figure 8.1 An example of 1NF violation and resolution

* **2NF (Second Normal Form):** (Ch 8.1.2, p.347-350) Must be in 1NF. All non-key attributes must be fully functionally dependent on the entire primary key. No partial dependencies (where a non-key attribute depends on only a part of a composite primary key). *(Fig 8.2, p.350)*



R: NEW_ALBUM (Album_no, Artist_nm, Price, Stock)

NEW_ALBUM

Album_no	Artist_nm	Price	Stock
BS123	Britney Spears	17.95	1000
JT111	Justin Timberlake	17.95	1200
BTL007	John Lennon	23.95	
BTL007	Paul McCartney	23.95	
BTL007	George Harrison	23.95	
BTL007	Ringo Star	23.95	
MJ100	Michael Jackson	17.95	
JM456	John Mayer	16.95	1000
JM151	John Mayer	16.95	1000
MX789	Madonna	11.95	500
DJM237	John Denver	11.95	2000
DJM237	Michael Jackson	11.95	2000
DJM237	Madonna	11.95	2000
DR711	Diana Ross	12.95	1000
PM137	Paul McCartney	19.95	

F: fd1: Album_no → Price; fd2: Album_no → Stock

F+: fd12: Album_no → (Price, Stock); fd12x: (Album_no, Artist_nm) → (Price, Stock);

Candidate key of NEW_ALBUM: (Album_no, Artist_nm); Primary key: (Album_no, Artist_nm)

fd1 and fd2 (fd12) violate 2NF in NEW_ALBUM

Solution:

D: R1: ALBUM_INFO (Album_no, Price, Stock); R2: ALBUM_ARTIST (Album_no, Artist_nm)

1 R1 n

1 R 1

ALBUM_INFO

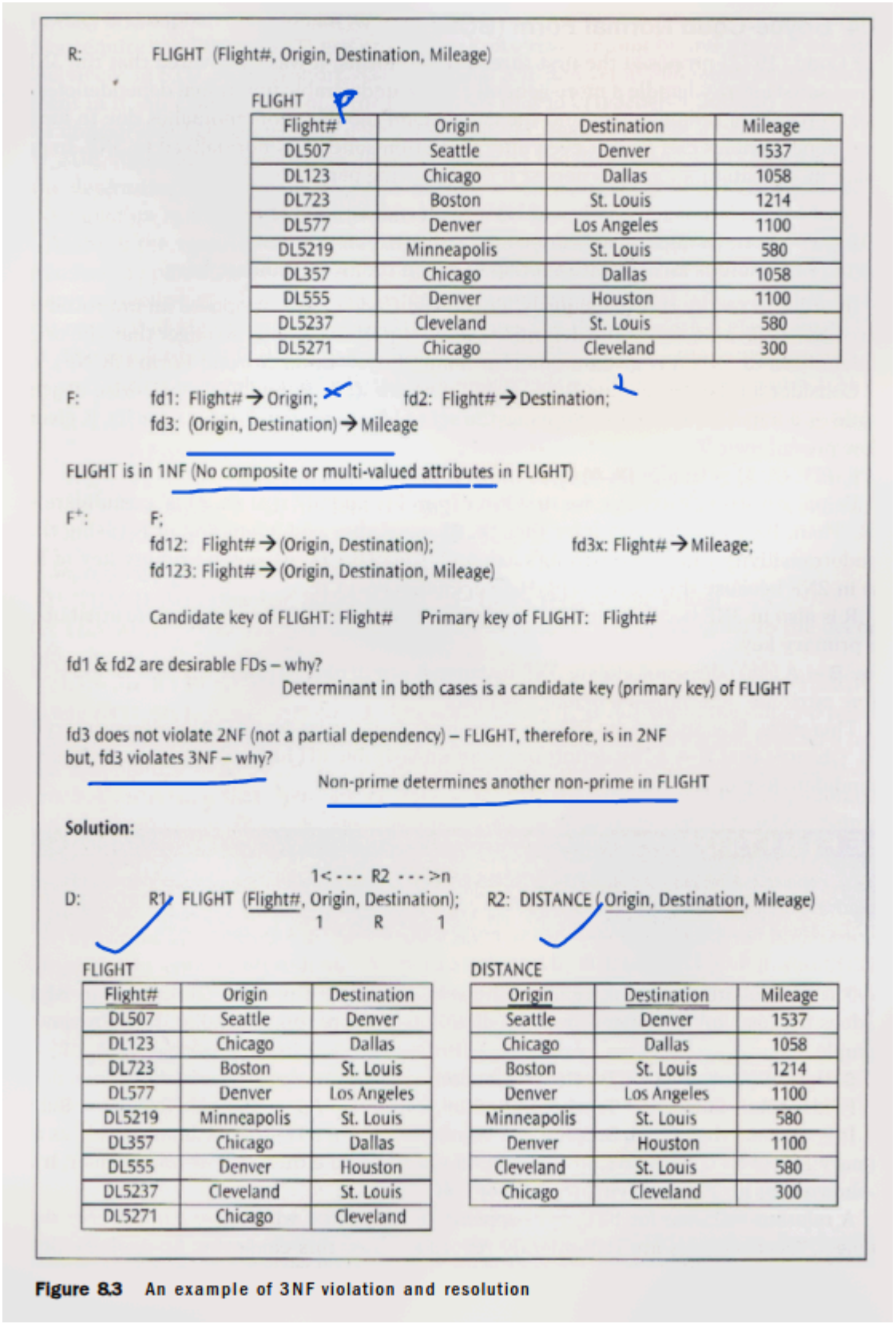
Album_no	Price	Stock
BS123	17.95	1000
JT111	17.95	1200
BTL007	23.95	
MJ100	17.95	
JM456	16.95	1000
JM151	16.95	1000
MX789	11.95	500
DJM237	11.95	2000
DR711	12.95	1000
PM137	19.95	

ALBUM_ARTIST

Album_no	Artist_nm
BS123	Britney Spears
JT111	Justin Timberlake
BTL007	John Lennon
BTL007	Paul McCartney
BTL007	George Harrison
BTL007	Ringo Star
MJ100	Michael Jackson
JM456	John Mayer
JM151	John Mayer
MX789	Madonna
DJM237	John Denver
DJM237	Michael Jackson
DJM237	Madonna
DR711	Diana Ross
PM137	Paul McCartney

Figure 8.2 An example of 2NF violation and resolution

* **3NF (Third Normal Form):** (Ch 8.1.3, p.351-353) Must be in 2NF. No transitive dependencies (where a non-key attribute depends on another non-key attribute, which in turn depends on the PK). *(Fig 8.3,*



R: FLIGHT (Flight#, Origin, Destination, Mileage)

FLIGHT

Flight#	Origin	Destination	Mileage
DL507	Seattle	Denver	1537
DL123	Chicago	Dallas	1058
DL723	Boston	St. Louis	1214
DL577	Denver	Los Angeles	1100
DL5219	Minneapolis	St. Louis	580
DL357	Chicago	Dallas	1058
DL555	Denver	Houston	1100
DL5237	Cleveland	St. Louis	580
DL5271	Chicago	Cleveland	300

F: fd1: Flight# → Origin; fd2: Flight# → Destination;
fd3: (Origin, Destination) → Mileage

FLIGHT is in 1NF (No composite or multi-valued attributes in FLIGHT)

F+: F;
fd12: Flight# → (Origin, Destination); fd3x: Flight# → Mileage;
fd123: Flight# → (Origin, Destination, Mileage)

Candidate key of FLIGHT: Flight# Primary key of FLIGHT: Flight#

fd1 & fd2 are desirable FDs – why?
Determinant in both cases is a candidate key (primary key) of FLIGHT

fd3 does not violate 2NF (not a partial dependency) – FLIGHT, therefore, is in 2NF
but, fd3 violates 3NF – why?
Non-prime determines another non-prime in FLIGHT

Solution:

D: R1: FLIGHT (Flight#, Origin, Destination); R2: DISTANCE (Origin, Destination, Mileage)
1 R 1 1<- - - R2 - - ->n

FLIGHT

Flight#	Origin	Destination
DL507	Seattle	Denver
DL123	Chicago	Dallas
DL723	Boston	St. Louis
DL577	Denver	Los Angeles
DL5219	Minneapolis	St. Louis
DL357	Chicago	Dallas
DL555	Denver	Houston
DL5237	Cleveland	St. Louis
DL5271	Chicago	Cleveland

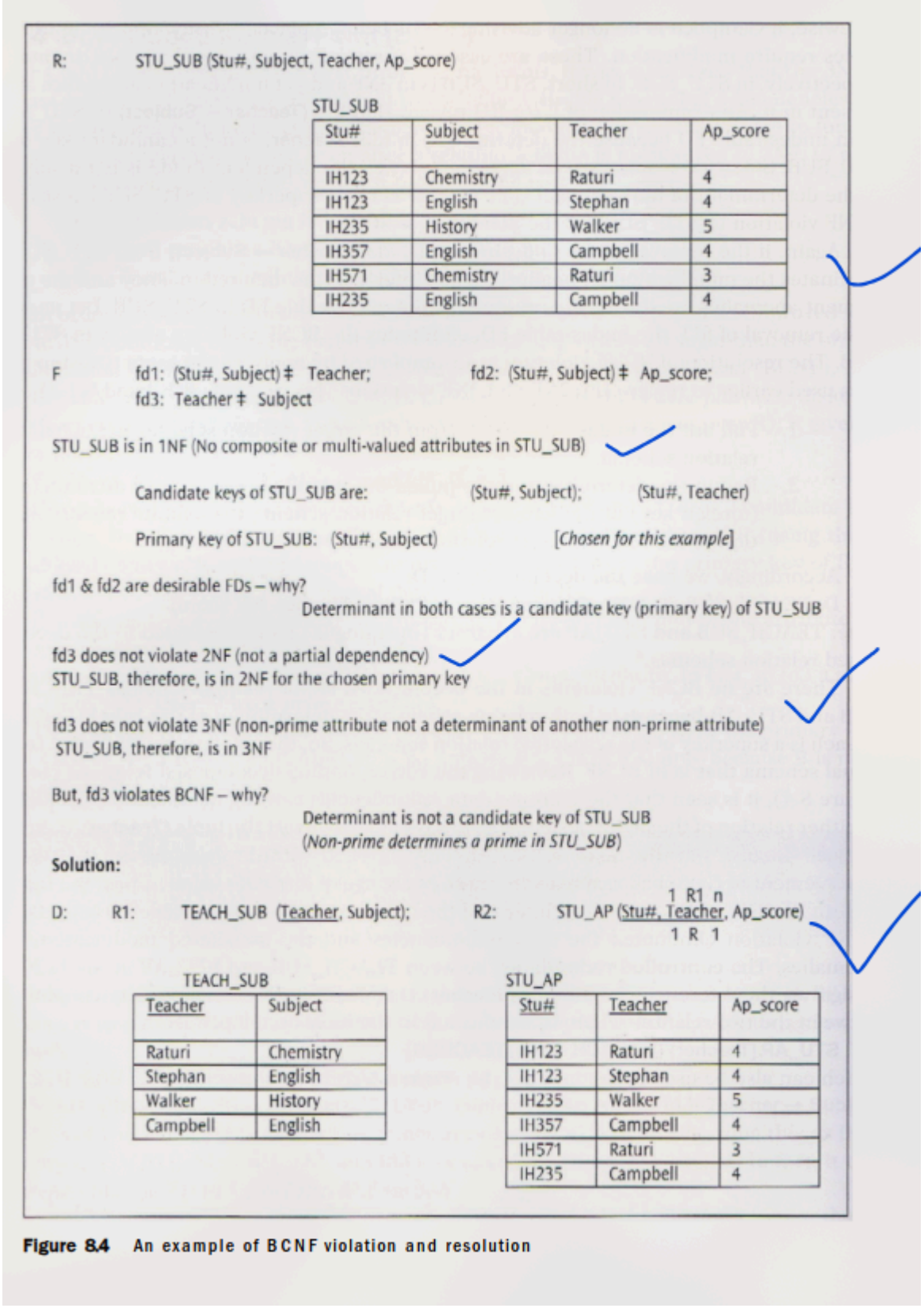
DISTANCE

Origin	Destination	Mileage
Seattle	Denver	1537
Chicago	Dallas	1058
Boston	St. Louis	1214
Denver	Los Angeles	1100
Minneapolis	St. Louis	580
Denver	Houston	1100
Cleveland	St. Louis	580
Chicago	Cleveland	300

Figure 8.3 An example of 3NF violation and resolution

p.353)

* **BCNF (Boyce-Codd Normal Form):** (Ch 8.1.4, p.354-356) A stricter version of 3NF. Must be in 3NF. Every determinant (attribute or set of attributes on which some other attribute is fully functionally dependent) must be a superkey (or candidate key). *(Fig 8.4, p.357)*



R: STU_SUB (Stu#, Subject, Teacher, Ap_score)

STU_SUB

Stu#	Subject	Teacher	Ap_score
IH123	Chemistry	Raturi	4
IH123	English	Stephan	4
IH235	History	Walker	5
IH357	English	Campbell	4
IH571	Chemistry	Raturi	3
IH235	English	Campbell	4

F fd1: (Stu#, Subject) ‡ Teacher; fd2: (Stu#, Subject) ‡ Ap_score;
fd3: Teacher ‡ Subject

STU_SUB is in 1NF (No composite or multi-valued attributes in STU_SUB)

Candidate keys of STU_SUB are: (Stu#, Subject); (Stu#, Teacher)

Primary key of STU_SUB: (Stu#, Subject) [*Chosen for this example*]

fd1 & fd2 are desirable FDs – why?
Determinant in both cases is a candidate key (primary key) of STU_SUB

fd3 does not violate 2NF (not a partial dependency)
STU_SUB, therefore, is in 2NF for the chosen primary key

fd3 does not violate 3NF (non-prime attribute not a determinant of another non-prime attribute)
STU_SUB, therefore, is in 3NF

But, fd3 violates BCNF – why?
Determinant is not a candidate key of STU_SUB
(*Non-prime determines a prime in STU_SUB*)

Solution:

D: R1: TEACH_SUB (Teacher, Subject); R2: STU_AP (Stu#, Teacher, Ap_score)
1 R1 n
1 R 1

TEACH_SUB

Teacher	Subject
Raturi	Chemistry
Stephan	English
Walker	History
Campbell	English

STU_AP

Stu#	Teacher	Ap_score
IH123	Raturi	4
IH123	Stephan	4
IH235	Walker	5
IH357	Campbell	4
IH571	Raturi	3
IH235	Campbell	4

Figure 8.4 An example of BCNF violation and resolution

* **4NF (Fourth Normal Form):** (Ch 9.2, p.422-428) Must be in BCNF. No non-trivial multi-valued dependencies (MVDs) unless the determinant is a superkey. (An MVD X ->> Y means Y values are determined by X independently of other attributes Z). *(Fig 9.1, p.424; Fig 9.2, p.427)*

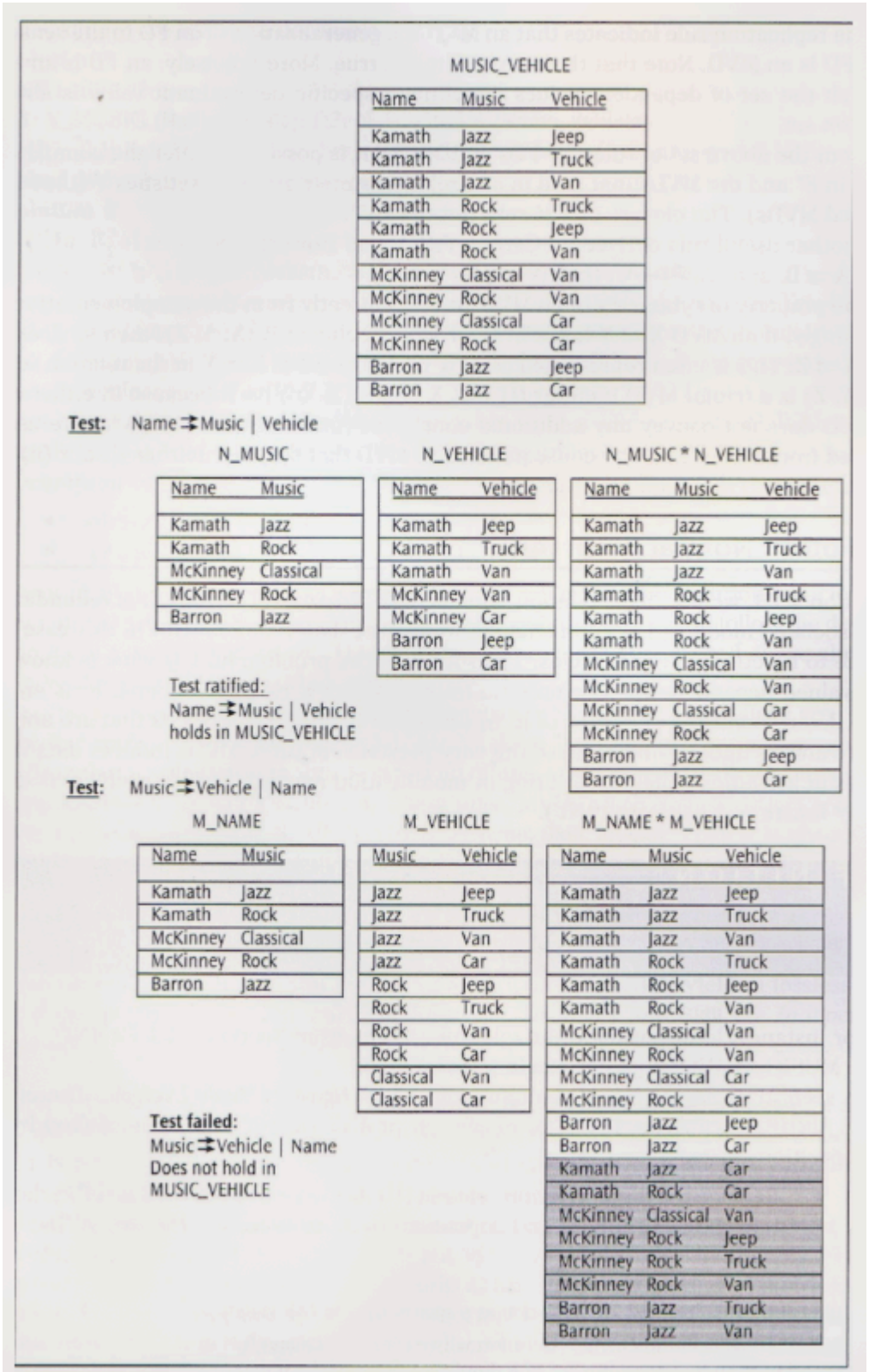
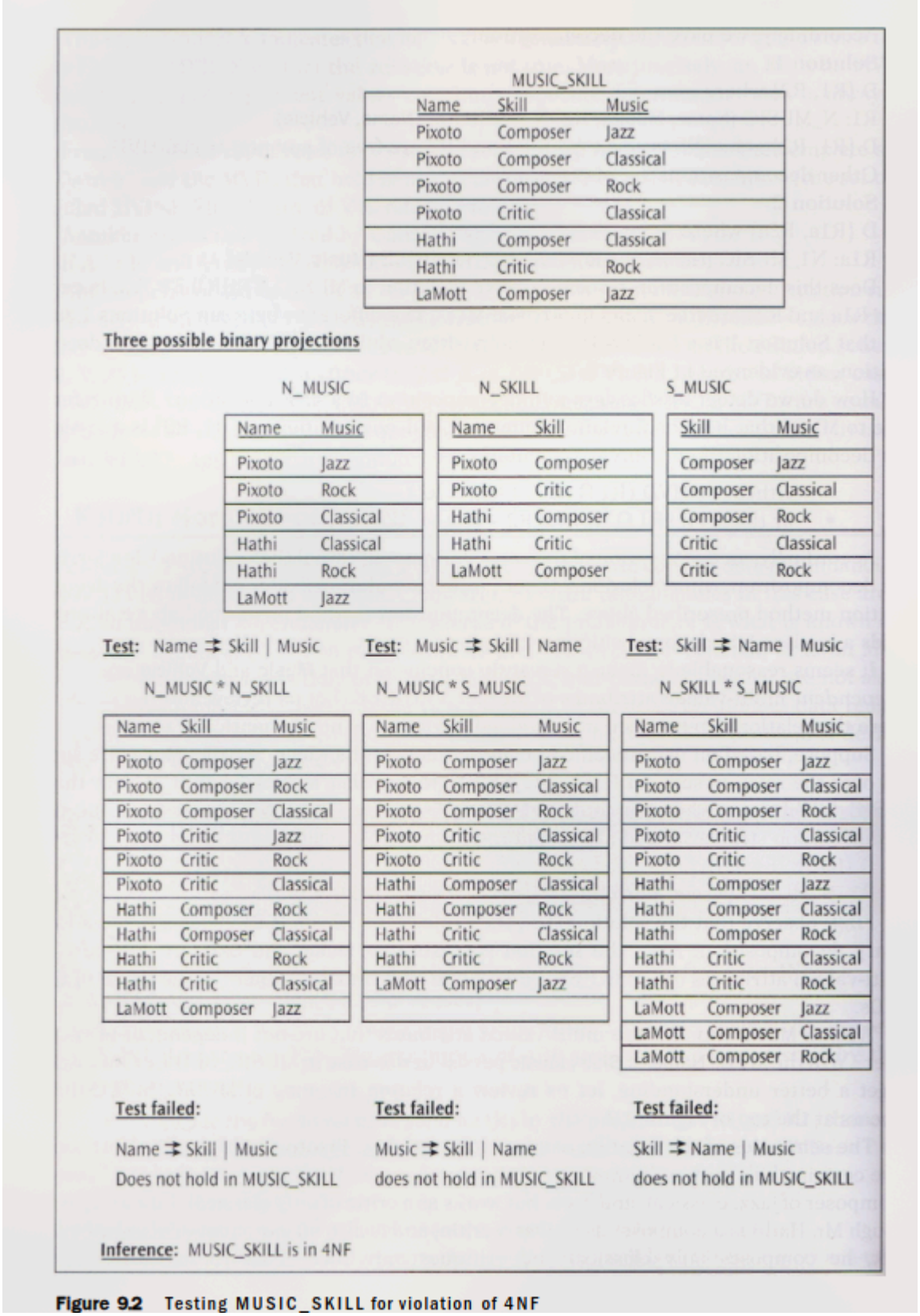


Figure 9.1 Test to check for the presence of multi-valued dependencies



MUSIC_SKILL

Name	Skill	Music
Pixoto	Composer	Jazz
Pixoto	Composer	Classical
Pixoto	Composer	Rock
Pixoto	Critic	Classical
Hathi	Composer	Classical
Hathi	Critic	Rock
LaMott	Composer	Jazz

Three possible binary projections

N_MUSIC

Name	Music
Pixoto	Jazz
Pixoto	Rock
Pixoto	Classical
Hathi	Classical
Hathi	Rock
LaMott	Jazz

N_SKILL

Name	Skill
Pixoto	Composer
Pixoto	Critic
Hathi	Composer
Hathi	Critic
LaMott	Composer

S_MUSIC

Skill	Music
Composer	Jazz
Composer	Classical
Composer	Rock
Critic	Classical
Critic	Rock

Test: Name ⇉ Skill | Music

N_MUSIC * N_SKILL

Name	Skill	Music
Pixoto	Composer	Jazz
Pixoto	Composer	Rock
Pixoto	Composer	Classical
Pixoto	Critic	Jazz
Pixoto	Critic	Rock
Pixoto	Critic	Classical
Hathi	Composer	Rock
Hathi	Composer	Classical
Hathi	Critic	Rock
Hathi	Critic	Classical
LaMott	Composer	Jazz

Test failed:

Name ⇉ Skill | Music

Does not hold in MUSIC_SKILL

Test: Music ⇉ Skill | Name

N_MUSIC * S_MUSIC

Name	Skill	Music
Pixoto	Composer	Jazz
Pixoto	Composer	Classical
Pixoto	Composer	Rock
Pixoto	Critic	Classical
Pixoto	Critic	Rock
Hathi	Composer	Classical
Hathi	Composer	Rock
Hathi	Critic	Classical
Hathi	Critic	Rock
LaMott	Composer	Jazz

Test failed:

Music ⇉ Skill | Name

does not hold in MUSIC_SKILL

Test: Skill ⇉ Name | Music

N_SKILL * S_MUSIC

Name	Skill	Music
Pixoto	Composer	Jazz
Pixoto	Composer	Classical
Pixoto	Composer	Rock
Pixoto	Critic	Classical
Pixoto	Critic	Rock
Hathi	Composer	Jazz
Hathi	Composer	Classical
Hathi	Composer	Rock
Hathi	Critic	Classical
Hathi	Critic	Rock
LaMott	Composer	Jazz
LaMott	Composer	Classical
LaMott	Composer	Rock

Test failed:

Skill ⇉ Name | Music

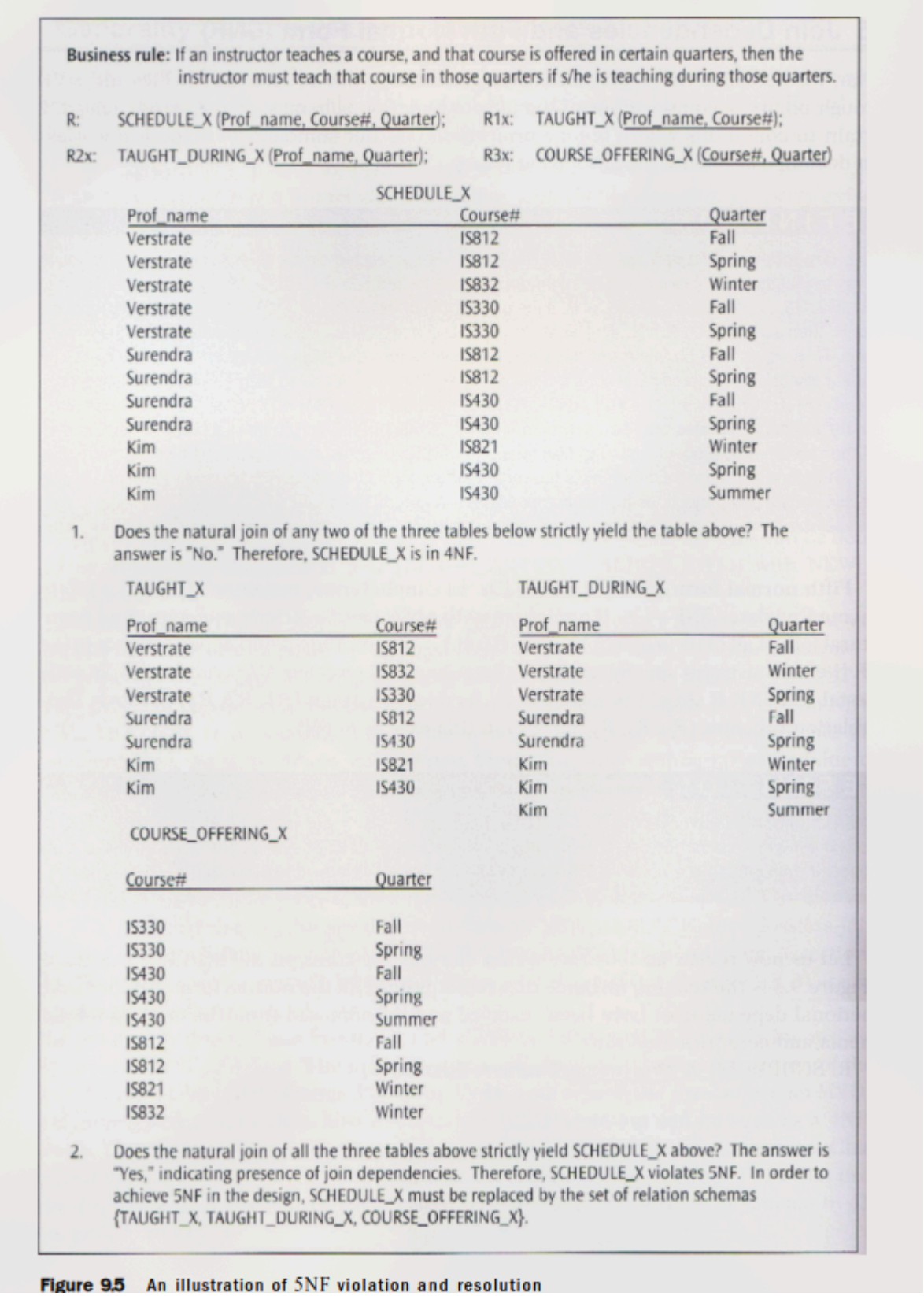
does not hold in MUSIC_SKILL

Inference: MUSIC_SKILL is in 4NF

Figure 9.2 Testing MUSIC_SKILL for violation of 4NF

* **5NF (Fifth Normal Form / Project-Join Normal Form - PJ/NF):** (Ch 9.5, p.429-433) Must be in 4NF. No join dependencies (JDs) that are not implied by candidate keys. (A JD means a relation can be

losslessly decomposed into projections and reconstructed by joining them). *(Fig 9.5, p.430; Fig 9.6, p.432)*



Business rule: If an instructor teaches a course, and that course is offered in certain quarters, then the instructor must teach that course in those quarters if s/he is teaching during those quarters.

R: SCHEDULE_X (Prof_name, Course#, Quarter); R1x: TAUGHT_X (Prof_name, Course#);

R2x: TAUGHT_DURING_X (Prof_name, Quarter); R3x: COURSE_OFFERING_X (Course#, Quarter)

SCHEDULE_X

Prof_name	Course#	Quarter
Verstrate	IS812	Fall
Verstrate	IS812	Spring
Verstrate	IS832	Winter
Verstrate	IS330	Fall
Verstrate	IS330	Spring
Surendra	IS812	Fall
Surendra	IS812	Spring
Surendra	IS430	Fall
Surendra	IS430	Spring
Kim	IS821	Winter
Kim	IS430	Spring
Kim	IS430	Summer

1. Does the natural join of any two of the three tables below strictly yield the table above? The answer is "No." Therefore, SCHEDULE_X is in 4NF.

TAUGHT_X

Prof_name	Course#
Verstrate	IS812
Verstrate	IS832
Verstrate	IS330
Surendra	IS812
Surendra	IS430
Kim	IS821
Kim	IS430

TAUGHT_DURING_X

Prof_name	Quarter
Verstrate	Fall
Verstrate	Winter
Verstrate	Spring
Surendra	Fall
Surendra	Spring
Kim	Winter
Kim	Spring
Kim	Summer

COURSE_OFFERING_X

Course#	Quarter
IS330	Fall
IS330	Spring
IS430	Fall
IS430	Spring
IS430	Summer
IS812	Fall
IS812	Spring
IS821	Winter
IS832	Winter

2. Does the natural join of all the three tables above strictly yield SCHEDULE_X above? The answer is "Yes," indicating presence of join dependencies. Therefore, SCHEDULE_X violates 5NF. In order to achieve 5NF in the design, SCHEDULE_X must be replaced by the set of relation schemas {TAUGHT_X, TAUGHT_DURING_X, COURSE_OFFERING_X}.

Figure 9.5 An illustration of 5NF violation and resolution

This should provide a solid foundation. Remember to actively draw the diagrams and work through the examples in your textbook to solidify your understanding. Good luck!

Most of the concepts in this notes (still they are in syllabus) will not be asked as paper is made very very easy for this subject. Normal basic ER and EER diagram should be more than enough for end term.

Practice those...