ITM 500: Data & Information Management

TOPICS TO REVIEW:

1- **Chapter 3: Relational Database Model**
2- **Chapter 4: ERD**
3- **Chapter 5: Complex relationship diagrams and Physical Design of Relational model**
4- **Chapter 7: Subqueries**

Reference:

(1) Definitions, examples and demonstrations:
Database Systems: Design, Implementation, and Management, 13th Edition.

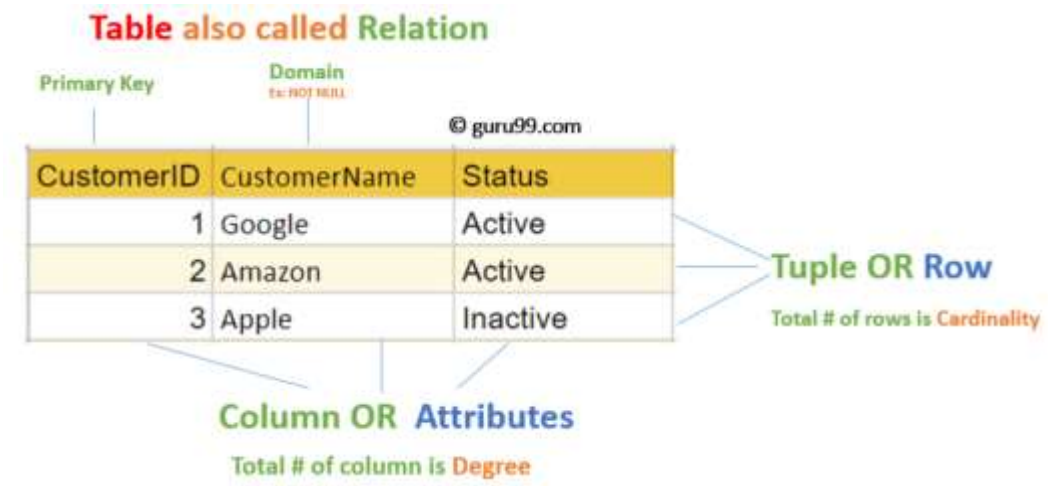
(2) Demonstrations and examples for Chapter 3:
https://www.guru99.com/relational-data-model-dbms.html

ITM 500: Data & Information Management

Chapter 3: Relational Database Model

CONCEPTS:

1. **Attribute:** Each column in a Table. Attributes are the properties which define a relation. e.g., Student_Rollno, NAME,etc.
2. **Tables** – In the Relational model the, relations are saved in the table format. It is stored along with its entities. A table has two properties rows and columns. Rows represent records and columns represent attributes.
3. **Tuple** – It is nothing but a single row of a table, which contains a single record.
4. **Relation Schema:** A relation schema represents the name of the relation with its attributes.
5. **Degree:** The total number of attributes which in the relation is called the degree of the relation.
6. **Cardinality:** Total number of rows present in the Table.
7. **Column:** The column represents the set of values for a specific attribute.
8. **Relation instance** – Relation instance is a finite set of tuples in the RDBMS system. Relation instances never have duplicate tuples.
9. **Relation key** - Every row has one, two or multiple attributes, which is called relation key.
10. **Attribute domain** – Every attribute has some pre-defined value and scope which is known as attribute domain



Source: https://www.guru99.com/relational-data-model-dbms.html

INTEGRITY CONSTRAINTS

Constraints on the Relational database management system is mostly divided into three main categories are:

- Domain constraints
- Key constraints
- Referential integrity constraints

(1) Database Systems: Design, Implementation, and Management, 13th Edition.

ITM 500: Data & Information Management

(1) Database Systems: Design, Implementation, and Management, 13th Edition.
(2) https://www.guru99.com/relational-data-model-dbms.html

1. **Domain Constraints**

Include standard data types integers, real numbers, characters, Booleans, variable length strings, etc.

Example:

```
Create DOMAIN CustomerName
CHECK (value not NULL)
```

The example shown demonstrates creating a domain constraint such that CustomerName is not NULL

2. **Key constraints**

To identify unique attribute.

Example:

In the given table, CustomerID is a key attribute of Customer Table. It is most likely to have a single key for one customer, CustomerID =1 is only for the CustomerName =" Google".

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Inactive

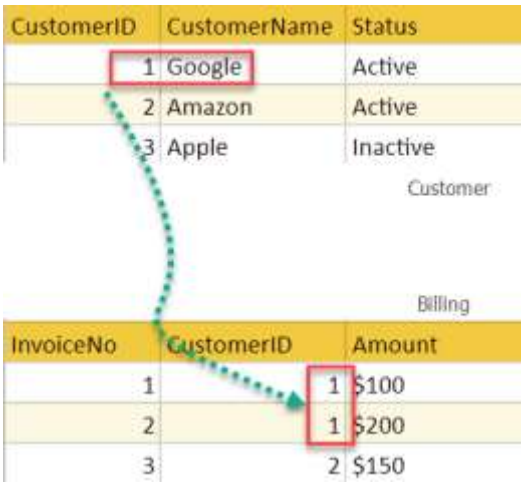
(1) Database Systems: Design, Implementation, and Management, 13th Edition.

ITM 500: Data & Information Management

3. **Referential integrity constraints**

Such as Foreign Keys.

A foreign key is a Primary key in another table, but become a "Foreign" in your current table.



Example:

In the above example, there're 2 relations, Customer and Billing.

Tuple for CustomerID =1 is referenced twice in the relation Billing.
→ CustomerName=Google has billing amount $300

COMMANDS/OPERATIONS IN RELATIONAL DB

- Insert → used to insert data into the relation
- Delete → used to delete tuples from the table.
- Modify → change the values of some attributes in existing tuples.
- Select → choose a specific range of data.

(1) Database Systems: Design, Implementation, and Management, 13th Edition.

ITM 500: Data & Information Management

Chapter 4: Entity Relationship Diagrams

*** The notations used in this prep work is only Crow's Foot/Martin/Information Engineering style*

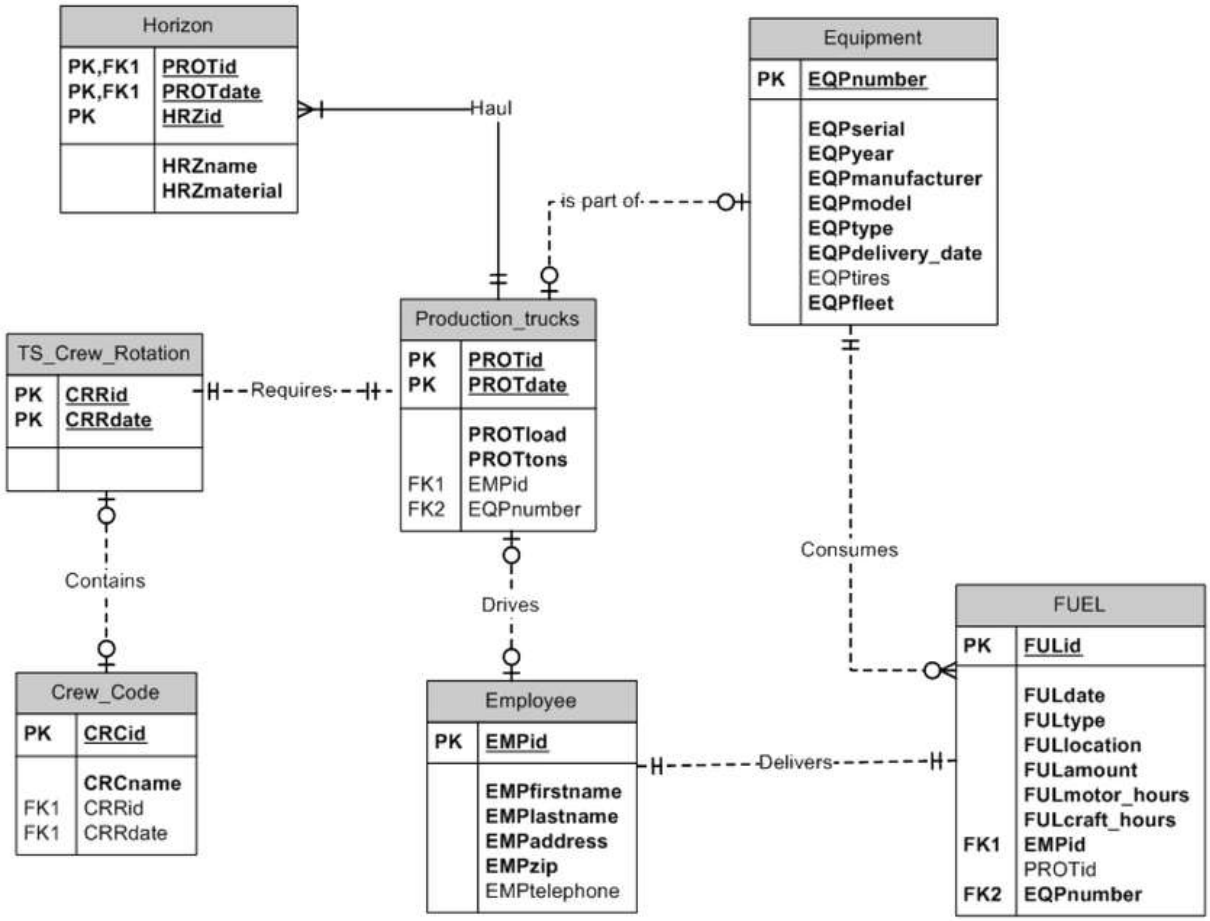
*** Students might be required to use other alternative notations: Chen notation style, Barker style, IDEF1X style*

Definition of an ER diagram

- Is a type of flowchart that illustrates how "entities" such as people, objects or concepts relate to each other within a system.
- ER Diagrams are most often used to design or debug relational databases in the fields of software engineering, business information systems, education and research.
- Also known as ERDs or ER Models, they use a defined set of symbols such as rectangles, diamonds, ovals and connecting lines to depict the interconnectedness of entities, relationships and their attributes.
- ER diagrams mirror grammatical structure, with entities as nouns and relationships as verbs.
- Example:
- ER diagrams are related to data structure diagrams (DSDs)
  - ➔ Focus on the relationships of elements within entities instead of relationships between entities themselves.
- ER diagrams also are often used in conjunction with data flow diagrams (DFDs)
  - ➔ Map out the flow of information for processes or systems.

(1) Database Systems: Design, Implementation, and Management, 13th Edition.

ITM 500: Data & Information Management



Source: https://www.researchgate.net/figure/The-entity-relationship-ER-diagram-developed-with-the-Microsoft-VISIO-CASE-tool_fig2_261061910

Components and features of an ER diagram

ER Diagrams are composed of entities, relationships and attributes.

1- Entity

A definable thing—such as a person, object, concept or event—that can have data stored about it.

(1) Database Systems: Design, Implementation, and Management, 13th Edition.

ITM 500: Data & Information Management

Think of entities as nouns.

Examples: a customer, student, car or product

a- Entity keys:

Primary key: A candidate key chosen by the database designer to uniquely identify the entity set

Foreign key: Identifies the relationship between entities.

2- Attribute

A property or characteristic of an entity

3- Relationship

How entities act upon each other or are associated with each other.

Think of relationships as verbs.

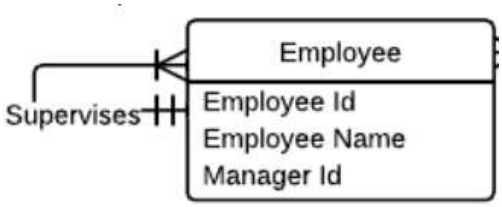
For example, the named student might register for a course. The two entities would be the student and the course, and the relationship depicted is the act of enrolling, connecting the two entities in that way

Types of relationships in ERD

a- Recursive

The same entity participates more than once in the relationship.

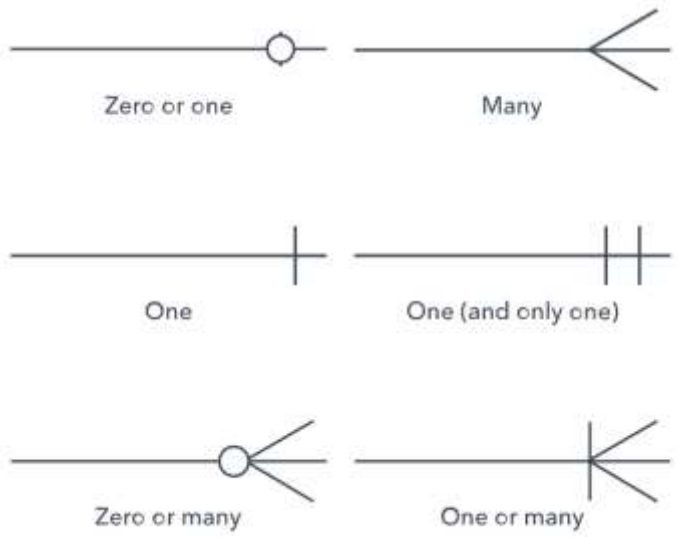
Ex:



b- Cardinal (Defining numbers in the relationship between 2 entities)

(1) Database Systems: Design, Implementation, and Management, 13th Edition.

ITM 500: Data & Information Management



Conceptual, logical and physical data models

ER models and data models are typically drawn at up to three levels of detail:

Conceptual data model:

- The highest-level view → contains the least detail.
- Showing overall scope of the model and portraying the system architecture.
- For a system of smaller scope, it may not be necessary to draw. Instead, start with the logical model.

Logical data model:

- Contains more detail than a conceptual model.
- More detailed operational and transactional entities are now defined.
- The logical model is independent of the technology in which it will be implemented.

Physical data model:

- One or more physical model may be developed from each logical model.
- Must show enough technology detail to produce and implement the actual database.

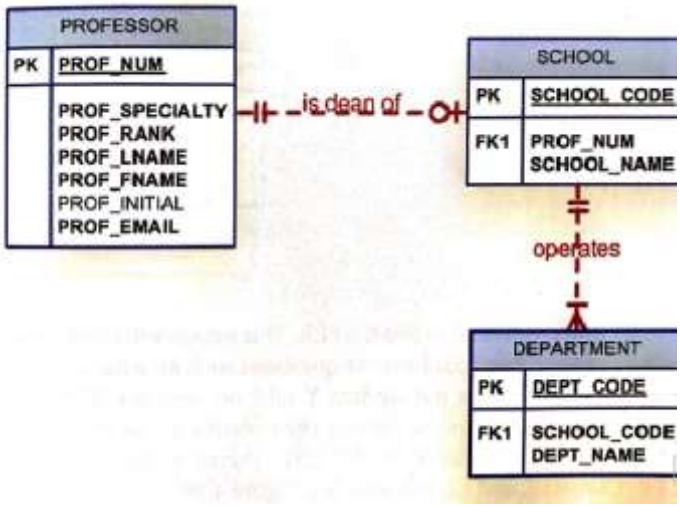
(1) Database Systems: Design, Implementation, and Management, 13th Edition.

ITM 500: Data & Information Management

Chapter 5: Complex relationship diagrams and Physical Design of Relational model

ERD	EERD
- Entity Relationship Diagram	- Extended Entity Relationship Diagram - Base on ERD concepts but have adds on: 1- Supertypes and Subtypes (Relationships) 2- Clustering (Virtual presentation of interrelated entities)

Example of ERD:

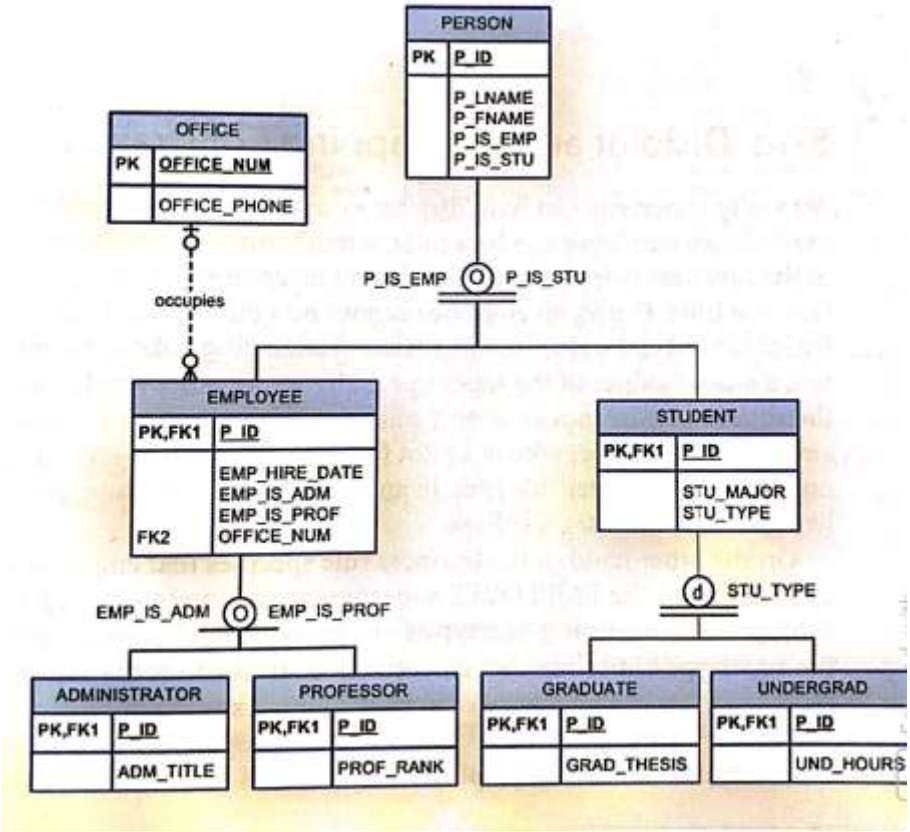


Source: Database Systems: Design, Implementation, and Management, 13th Edition. Chapter 4

(1) Database Systems: Design, Implementation, and Management, 13th Edition.

ITM 500: Data & Information Management

Example of EERD:



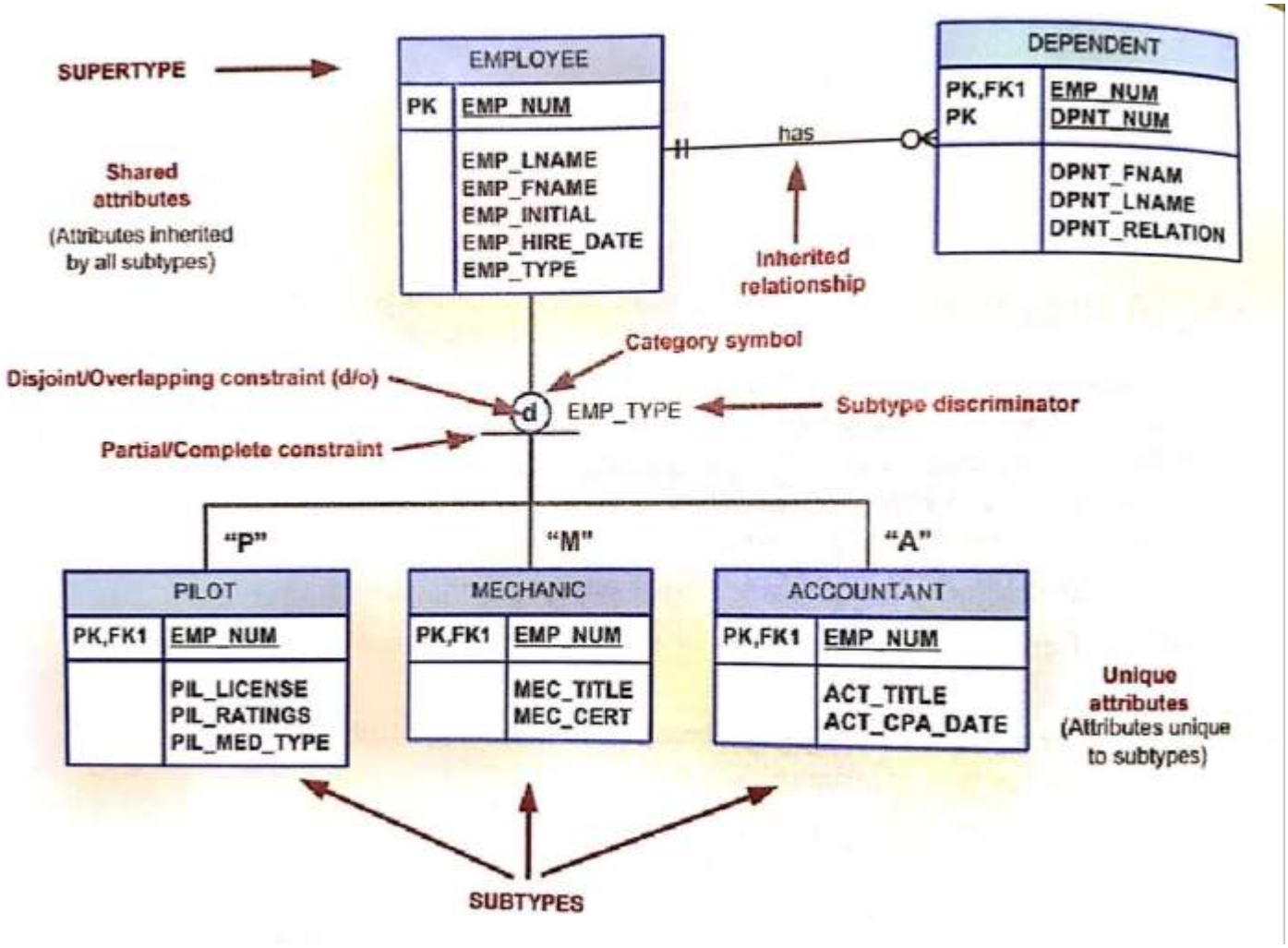
Source: Database Systems: Design, Implementation, and Management, 13th Edition. Chapter 4

ADD ON FEATURES OF EERD:

1- **Example of EERD with add on features**

(1) Database Systems: Design, Implementation, and Management, 13th Edition.

ITM 500: Data & Information Management



Source: Database Systems: Design, Implementation, and Management, 13th Edition. Chapter 4

1- <u>Supertypes and Subtypes</u>

- For presenting relationships between entities

Specialization hierarchy

- Top-down process of identifying lower-level.
- Entity subtypes → More specific
- Specialization is based on grouping unique characteristics of the subtypes

Entity supertype

- Generic entity type

Entity subtype

(1) Database Systems: Design, Implementation, and Management, 13th Edition.

ITM 500: Data & Information Management

- A subset of an Entity supertype
- Contain the unique characteristics of each entity

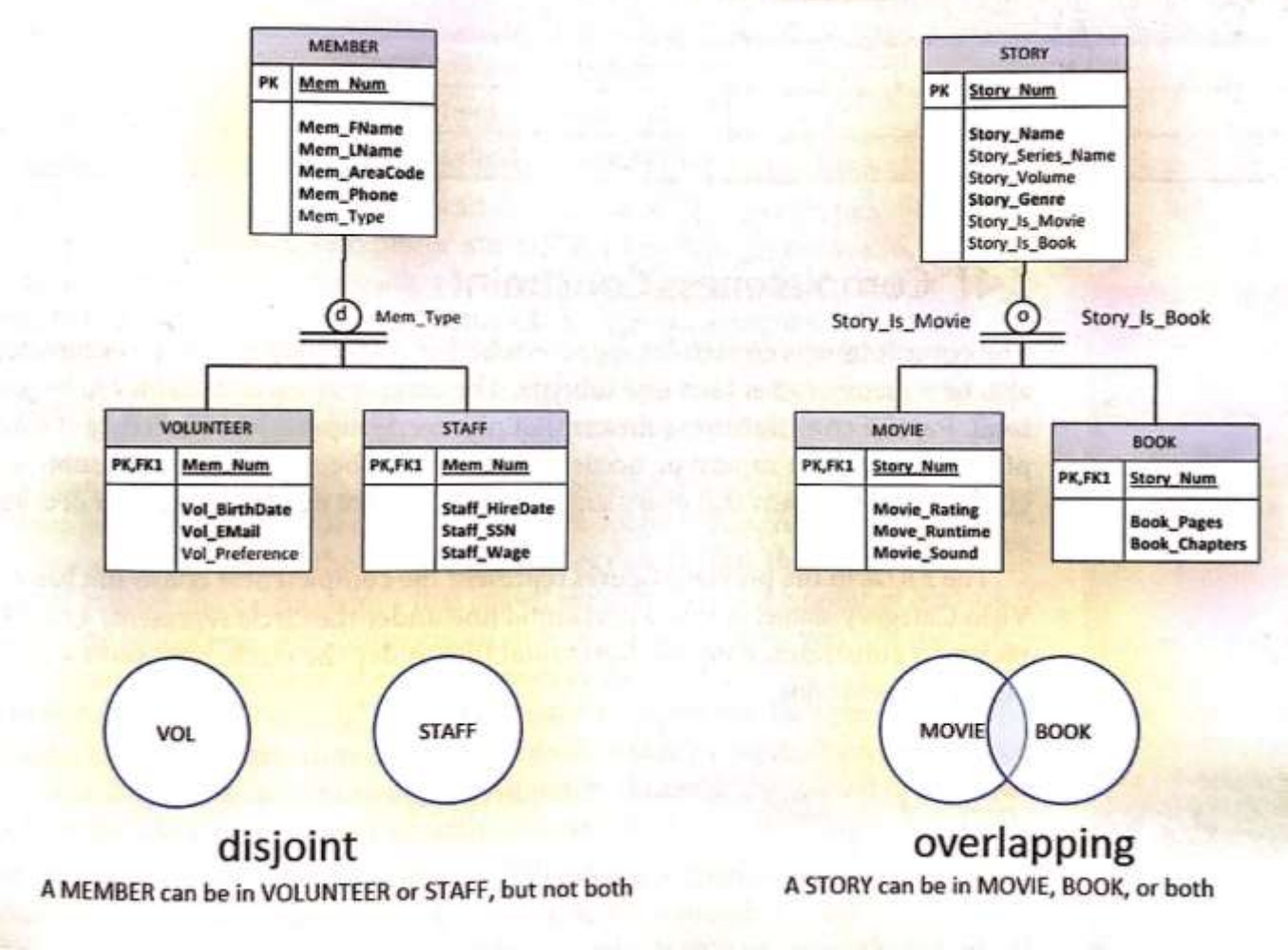
Inheritance

- Relationship between Subtype and Supertype
- Subtype inherits Supertype

Subtype Discriminator

- An attribute between Subtype and Supertype
- Determines which Entity Subtype relates/belongs to an Entity Supertype

Disjoint and Overlap → For Entity Subtypes



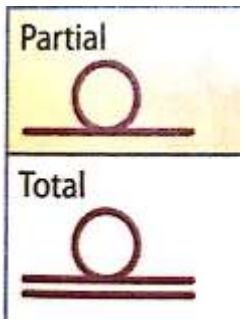
Source: Database Systems: Design, Implementation, and Management, 13th Edition. Chapter 4

(1) Database Systems: Design, Implementation, and Management, 13th Edition.

ITM 500: Data & Information Management

Completeness Constraint

- Partial
- Total



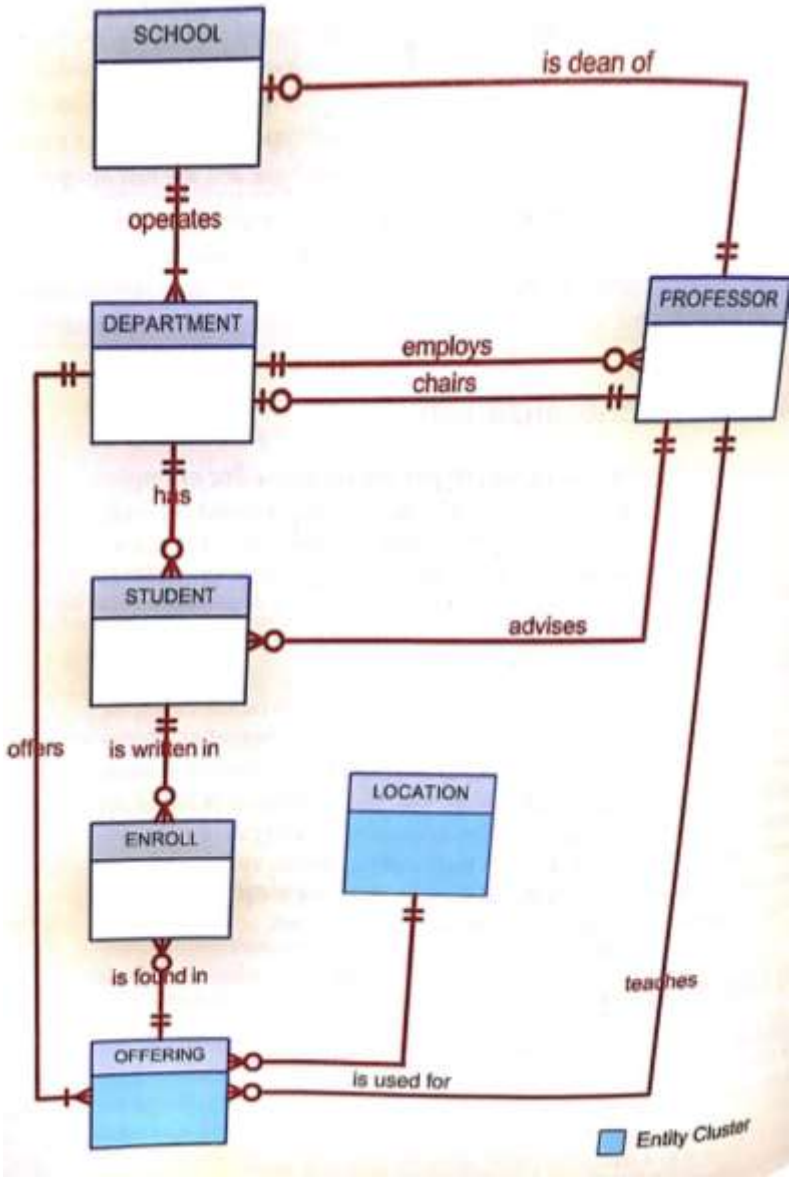
Source: Database Systems: Design, Implementation, and Management, 13th Edition. Chapter 4

2- <u>Entity Clustering</u>

- Used to represent multiple entities and relationships in ERD.
- Is formed by combining multiple interrelated entities into a single entity object
- Not an actual entity in the final ERD → **virtual** or **abstract**

(1) Database Systems: Design, Implementation, and Management, 13th Edition.

ITM 500: Data & Information Management



Source: Database Systems: Design, Implementation, and Management, 13th Edition. Chapter 4

DEFINING KEYS FOR AN ENTITY

Natural key (Natural identifier)

- Is familiar to end users and forms part of their day to day business vocabulary

Guidelines for identifying Primary Key

1- Uniquely identify an entity instance or row within a table
2- Choosing primary keys → Identify foreign keys from the relationships between entities

Surrogate key

- A system-assigned primary key, generally numeric and auto-incremented

(1) Database Systems: Design, Implementation, and Management, 13th Edition.

ITM 500: Data & Information Management

Chapter 7: Subqueries

- ➔ Queries inside queries
- ➔ Nested queries
- ➔ To select attributes from other tables (Which we can do by using JOIN)

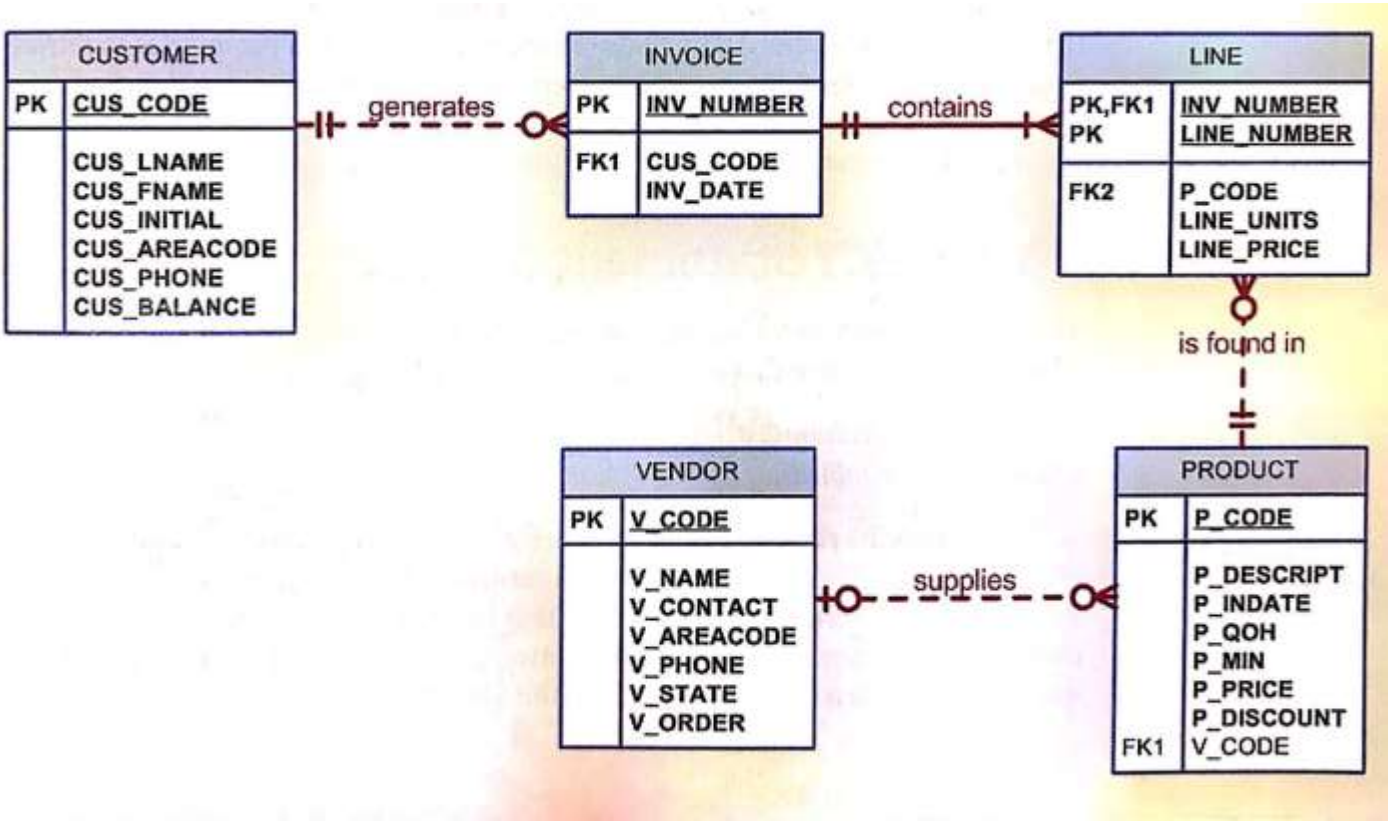
Example

SELECT column-names

FROM table-name1

WHERE value IN (SELECT column-name FROM table-name2 WHERE condition)

SAMPLE DBM:



Source: Database Systems: Design, Implementation, and Management, 13th Edition. Chapter 7

(1) Database Systems: Design, Implementation, and Management, 13th Edition.

ITM 500: Data & Information Management

USING SUBQUERY INSTEAD OF JOIN

JOIN	SUBQUERY
SELECT v_code, v_name FROM product RIGHT JOIN vendor ON product.v_code = vendor.v_code WHERE p_code IS NULL;	SELECT v_code, v_name FROM vendor WHERE v_code NOT IN (SELECT v_code FROM product WHERE v_code IS NOT NULL);

SUBQUERY → SELECT

```
SELECT p_code, p_price,
        (SELECT AVG(p_price) FROM product AS avg_price,
        p_price – (SELECT AVG(p_price) FROM product) AS diff
FROM product;
```

Result:

p_code	p_price	avg_price	diff
….	….	….	….

SUBQUERY → FROM

```
SELECT DISTINCT customer.cus_code, customer.cus_lname
FROM customer JOIN
        (SELECT invoice.cus_code FROM invoice
        JOIN line ON invoice.inv_number = line.inv_number
        WHERE p_code LIKE '%3%');
```

(1) Database Systems: Design, Implementation, and Management, 13th Edition.

ITM 500: Data & Information Management

SUBQUERY → WHERE

```
SELECT p_code, p_price
FROM product
WHERE p_price >= (SELECT AVG (p_price) FROM product);
```

SUBQUERY → WHERE > IN

```
SELECT DISTINCT customer.cus_code, cus_lname, cus_fname
FROM customer JOIN invoice ON customer.cus_code = invoice.cus_code
                JOIN line ON invoice.inv_number = line.inv_number
                JOIN product ON line.p_code = product.p_code
WHERE p_code IN (SELECT p_code FROM product
                    WHERE p_description LIKE '%hammer%' OR p_description LIKE '%saw%');
```

SUBQUERY → HAVING

```
SELECT p_code, sum(line_units) AS total_units
FROM line
GROUP BY p_code
HAVING sum(line_units) > (SELECT AVG (ine_units) FROM line);
```

(1) Database Systems: Design, Implementation, and Management, 13th Edition.