Relational Designs and ER Diagrams

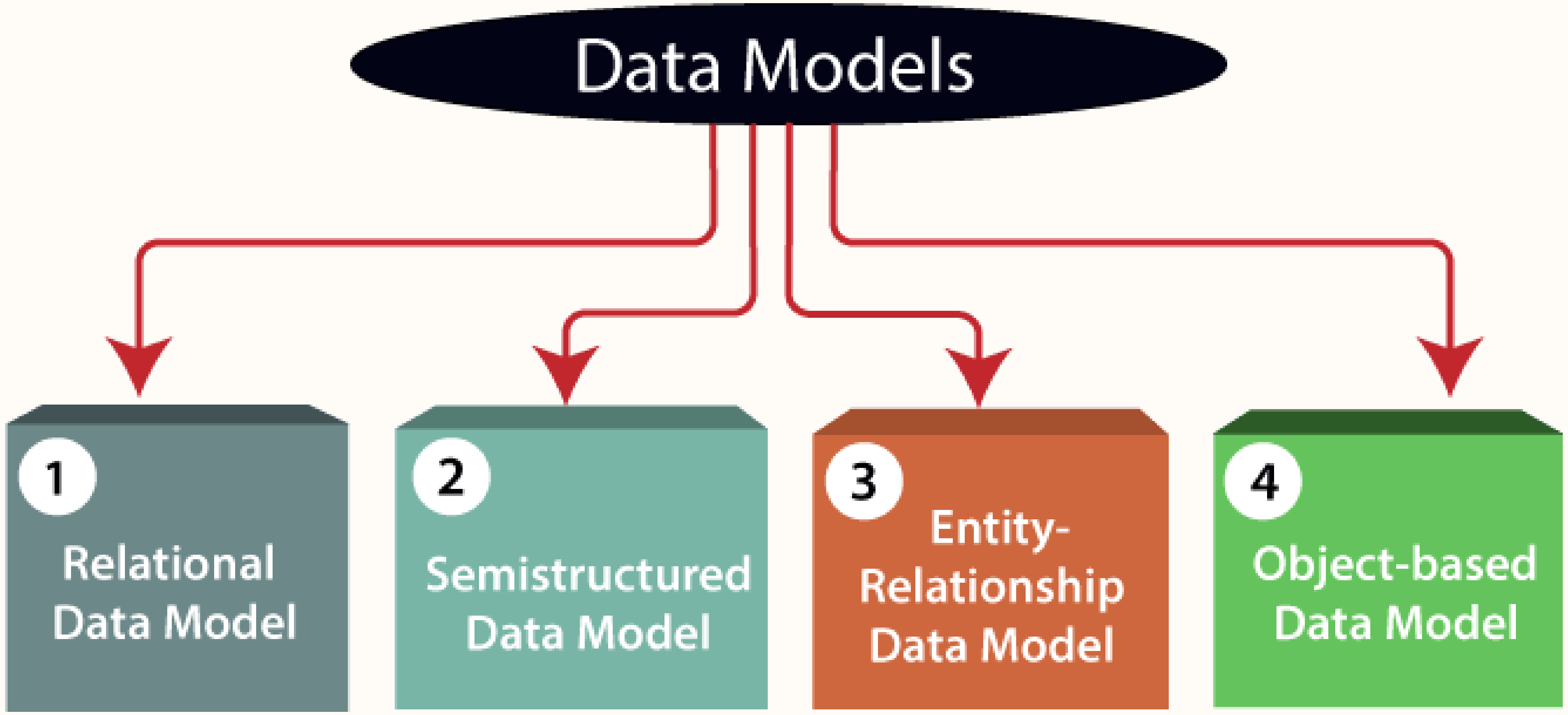
Objectives

By the end of this module, you will be able to:

- Understand data modeling
- Explain the of need relational design
- Become familiar with ER diagram
- Describe normalization

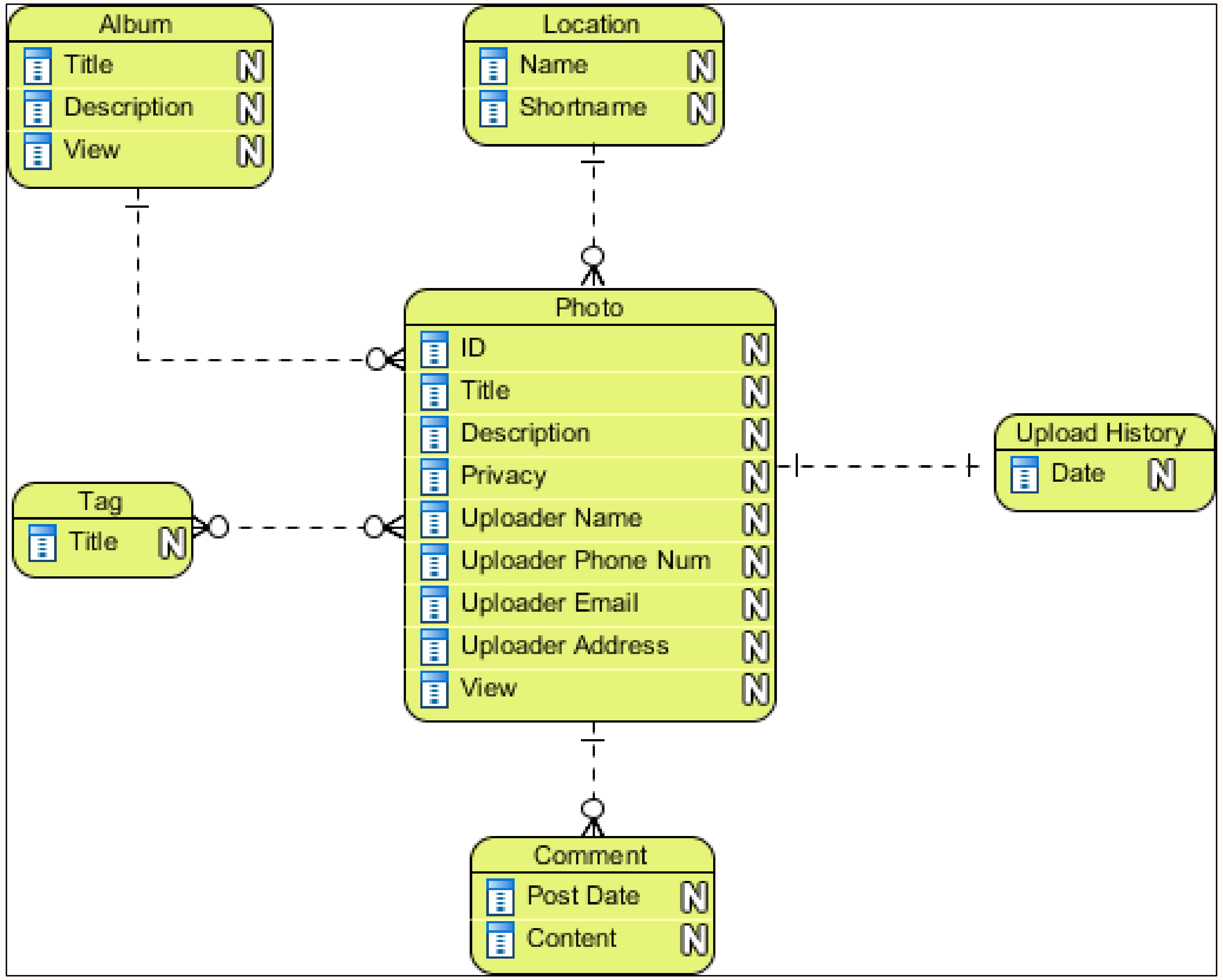
Data Modeling

Data modeling is the **process of creating a data model** for the data to be stored in a database.

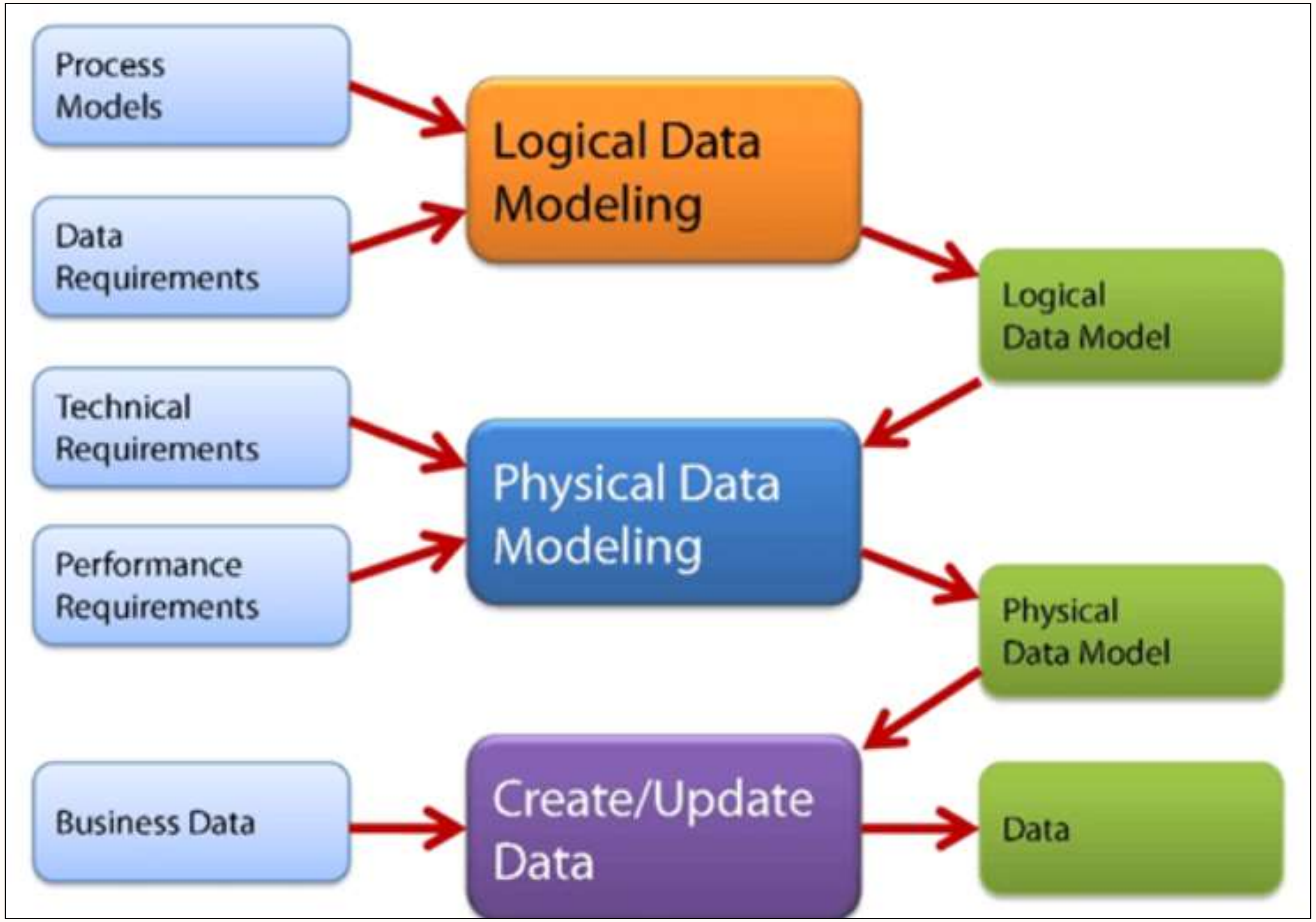


Data Model Examples

Data Model of a Photo Library:



Types of Data Models



Types of Data Models

There are mainly three different types of data models:

- Conceptual data models
- Logical data models
- Physical data models

Types of Data Models

Conceptual Data Model:

- Defines what the system contains.
- Created by Business stakeholders and Data Architects to organize, scope, and define business concepts and rules.

Logical Data Model:

- Defines how the system should be implemented regardless of the DBMS.
- This model is typically created by Data Architects and Business Analysts to a developed technical map of rules and data structures.

Types of Data Models

Physical Data Model:

- Describes how the system will be implemented using a specific DBMS system.
- Created by Database Administrators (DBA) and developers to implement the database.

Conceptual Data Model

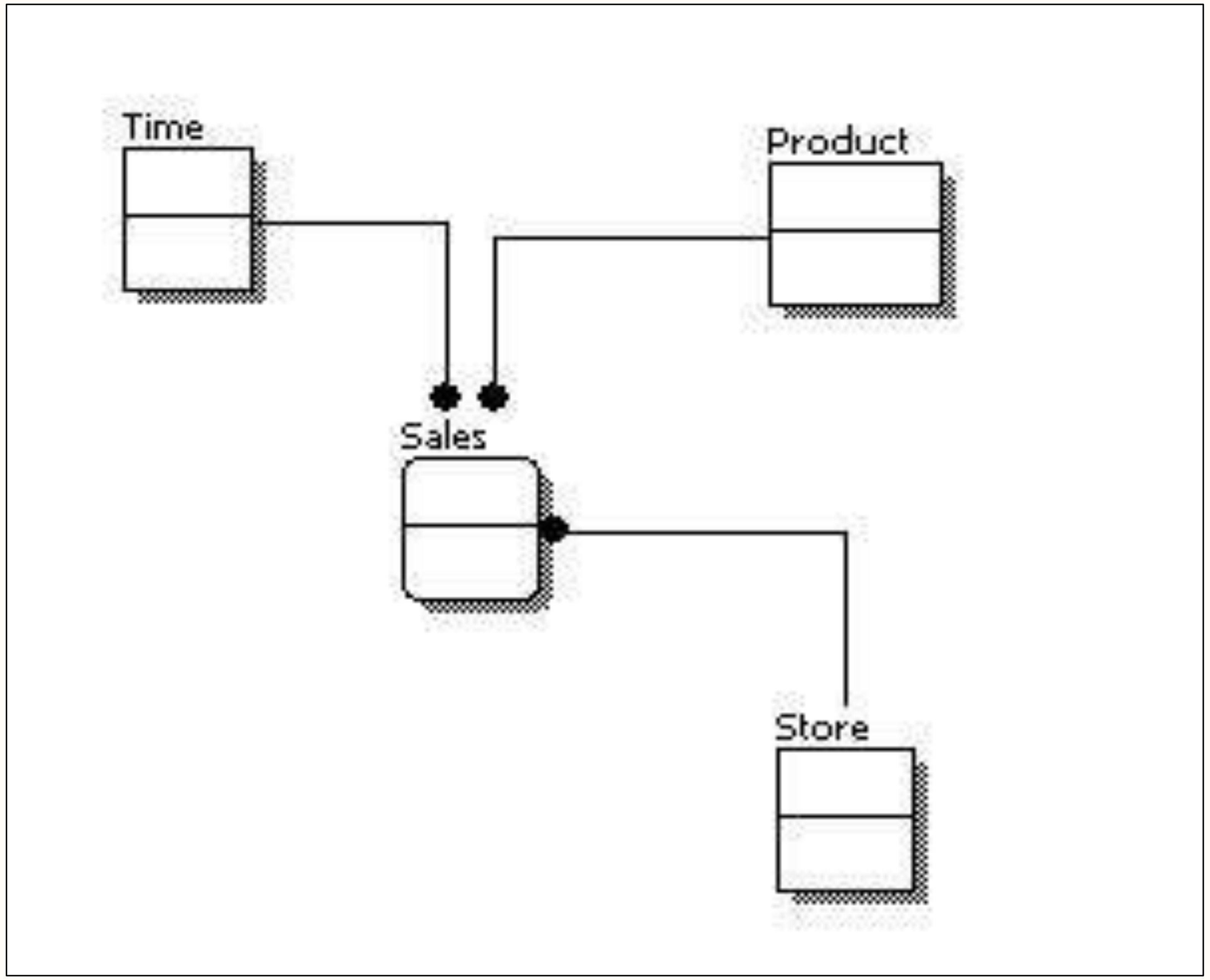
The two basic tenants of the Conceptual Data Model are:

1. **Entity**: A real-world thing
2. **Relationship**: Dependency or association between two entities

Conceptual Data Model

Data model example:

- Every Store has some sales
- Every sales comprises of some products
- Every sale happens at a particular time



Characteristics of a Conceptual Data Model

It offers organization-wide coverage of the **business concepts**.

This type of Data Models are designed and developed **for the business audience**.

The conceptual model is developed **independently of hardware specifications** like data storage capacity, location, or software specifications like DBMS vendor and technology.

The focus is to **represent data** as a user will see it in the **real world**.

Logical Data Model

A logical data model establishes the **structure of data elements** and the **relationships** among them.

It is **independent of the physical database** that details how the data will be implemented.

The logical data model serves as a **blueprint for used data**.

The logical data model takes the elements of **conceptual data modeling a step further by adding more information** to them.

Logical Data Model

Features of a logical data model include:

- Includes all **entities and relationships** among them.
- All **attributes** for each entity are specified.
- The **primary key** for each entity is specified.
- **Foreign keys** (keys identifying the relationship between different entities) are specified.
- **Normalization** occurs at this level.

Logical Data Model

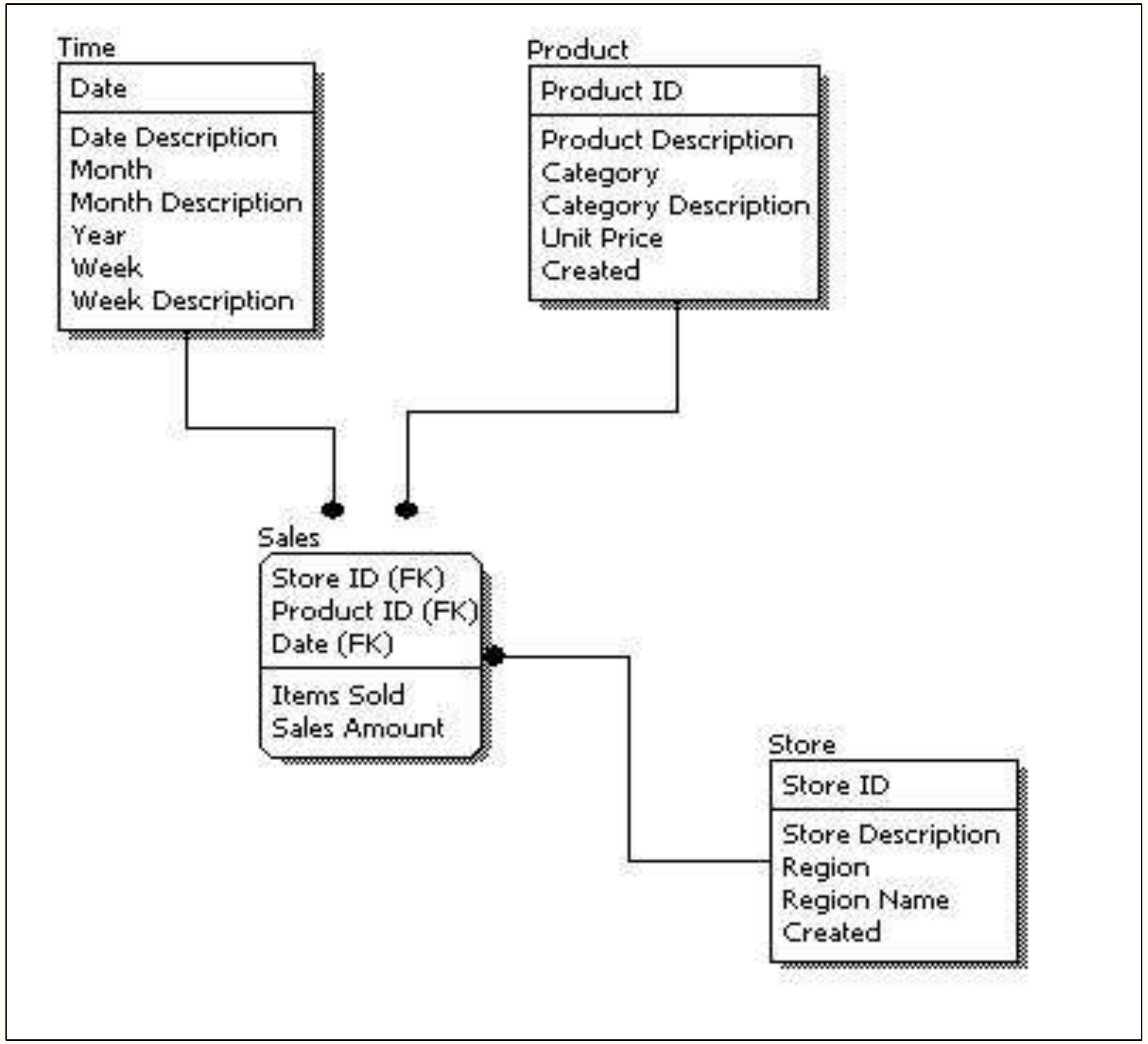
The steps for designing the logical data model are as follows:

1. Specify primary keys for all entities
2. Find the relationships between different entities
3. Find all attributes for each entity
4. Resolve many-to-many relationships
5. Normalization

Logical Data Model

Data model example:

- Identify the **attributes** for Store, Sales, Product, Time
- Identify **key attributes** for all
- Identify the **type of relationship** between all
- **Normalize** the entities if required



Physical Data Model

- The physical data model pertains to **how the system will be implemented**, and factors in the specific databases management system.

- This model is typically **created by developers.** The idea is more to define how the actual database will be used or implemented for business purposes.

- Conceptual data modeling and logical data modeling are “requirements analysis” types of activities, while physical data modeling is a **design activity**.

Physical Data Model

Features of a physical data model are:

- It contains **Specifications** of all tables and columns
- It contains **Foreign Keys** that are used to identify relationships between tables
- Denormalization may occur based on user requirements
- Physical considerations may cause the physical data model to be quite different from the logical data model
- Physical data model will be different for different RDBMS. For example, the data type for a column may be different between Oracle, DB2, etc.

Physical Data Model

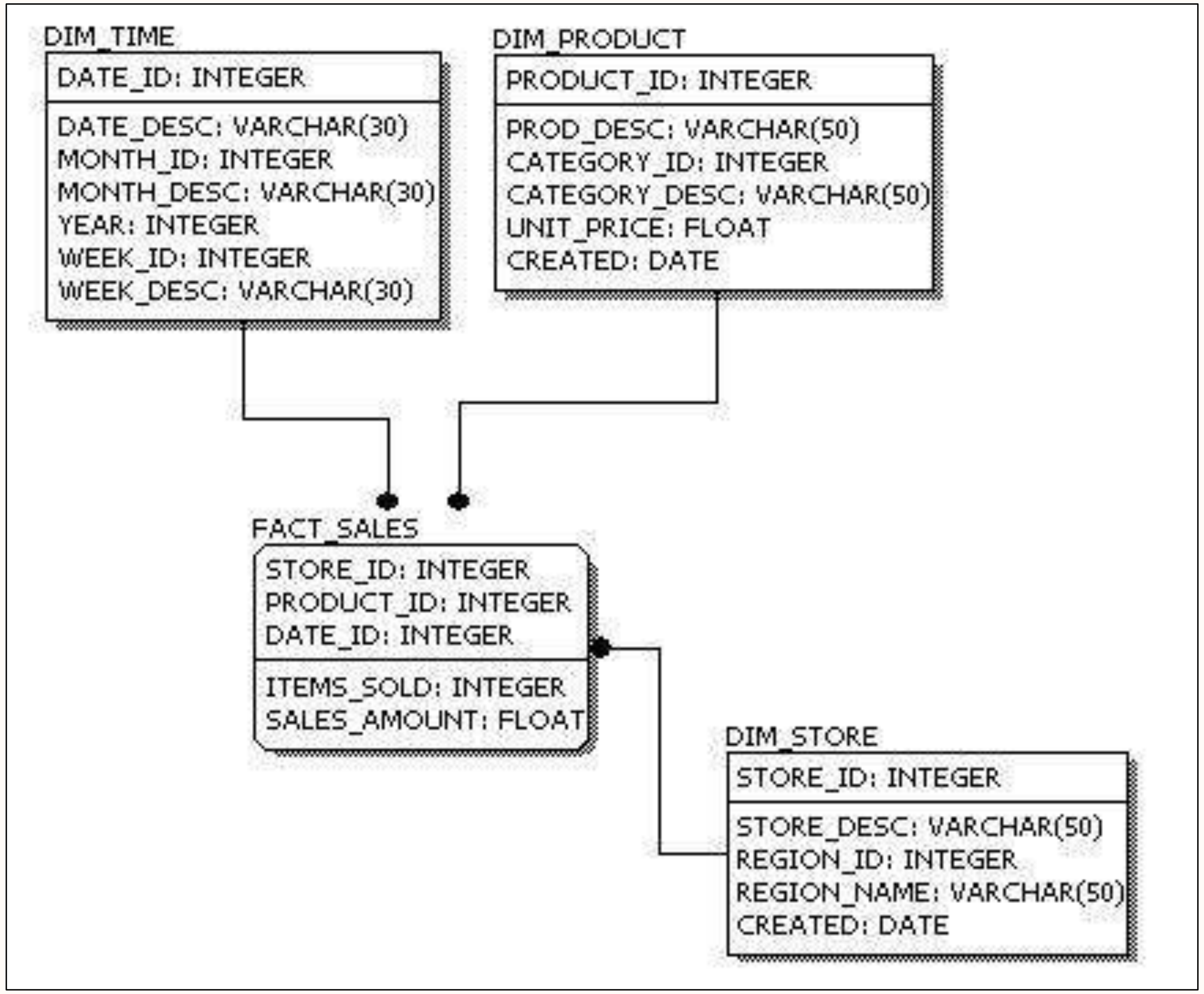
The steps for physical data model design are as follows:

- Convert **entities into tables**.
- Convert **relationships into Foreign Keys**.
- Convert **attributes into columns**.
- Modify the physical data model based on physical constraints/requirements.

Physical Data Model

Data model example:

- Create the required tables
- Connect the tables using Foreign Keys
- Add the respective attributes as columns



Data Model

Advantages of the data model:

- The main goal of designing a data model is to **make certain that data objects offered by the functional team are represented accurately**.
- The data model **should be detailed enough** to be used for building the physical database.
- The information in the data model can be **used for defining the relationship** between tables, primary and foreign keys, and stored procedures.

Data Model

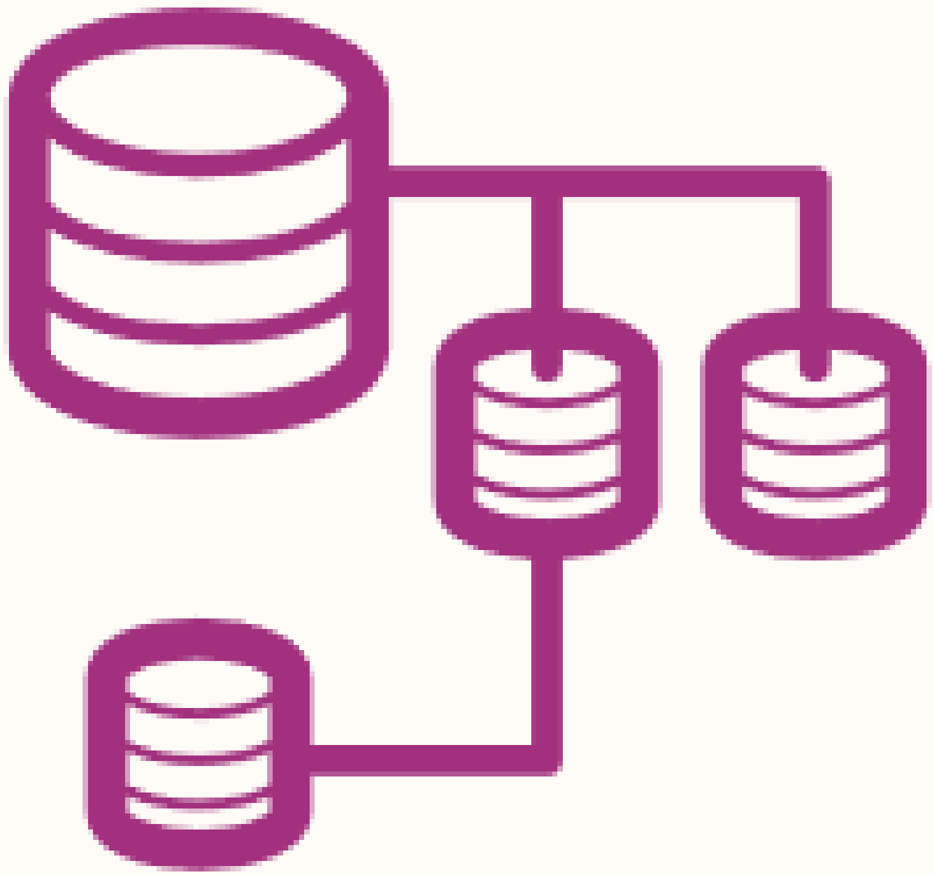
Advantages of the data model:

- Data Model helps businesses to communicate **within and across** organizations
- Data model helps to documents data mappings in the **ETL process**
- Help to **recognize correct sources** of data to populate the model

Data Modeling Technique

Data modeling techniques fall into one of two categories:

- Entity Relationship (E-R) Model
- UML (Unified Modelling Language)



Entity Relationship (ER) Diagram

- An entity relationship diagram (ERD) **shows the relationships of entity sets** stored in a database.
- An entity in this context is an **object**, a component of **data**.
- An **entity set** is a **collection** of similar **entities.**
- These entities can have **attributes** that define their properties.
- By defining the entities, their attributes, and showing the relationships between them, an **ER diagram illustrates the logical structure** of databases.
- ER diagrams are used to sketch out the **design of a database**.

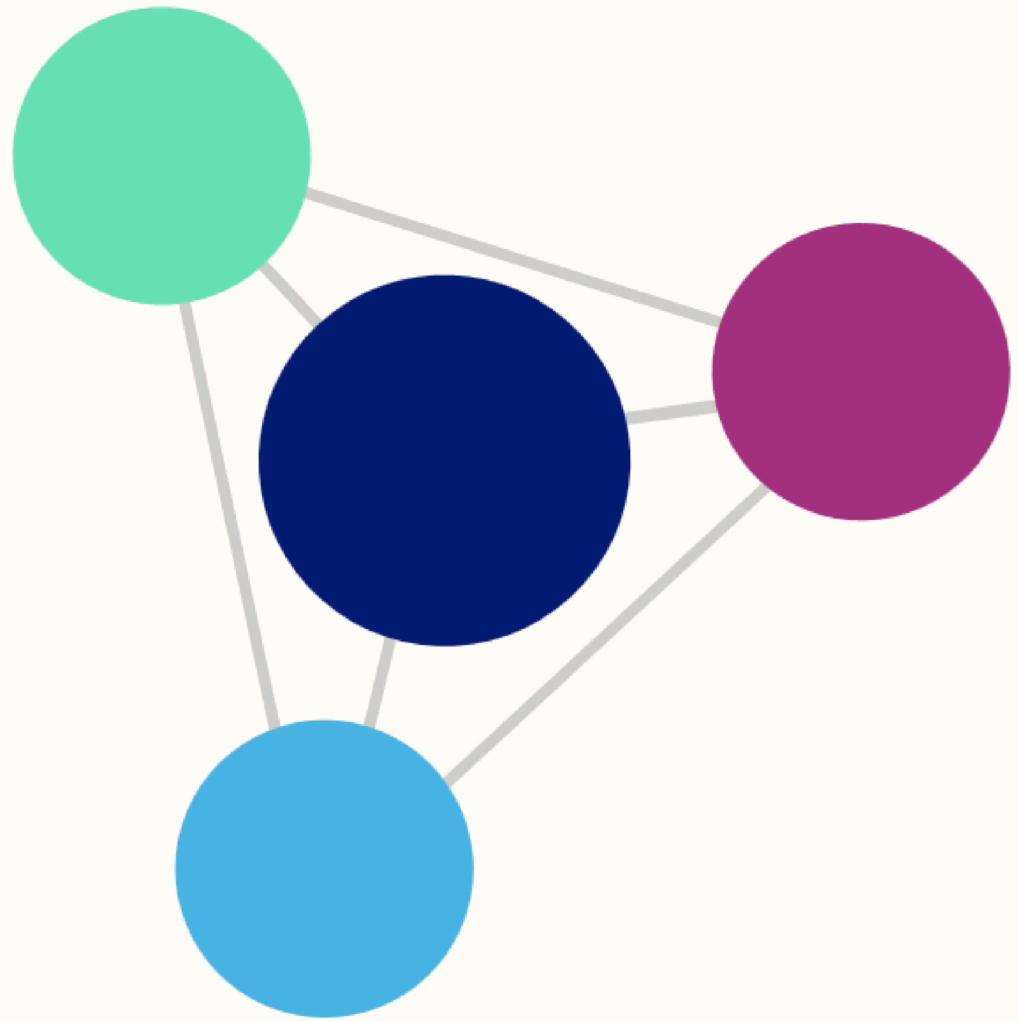
Entity Relationship Diagram

ER Diagram:

- Helps you to **define terms related to entity relationship** modeling
- Provide a preview of **how all your tables should connect**, what fields are going to be on each table
- Helps to **describe entities, attributes, relationships**
- ER diagrams are **translatable into relational tables** which allows you to build databases quickly

Entity Relationship Diagram

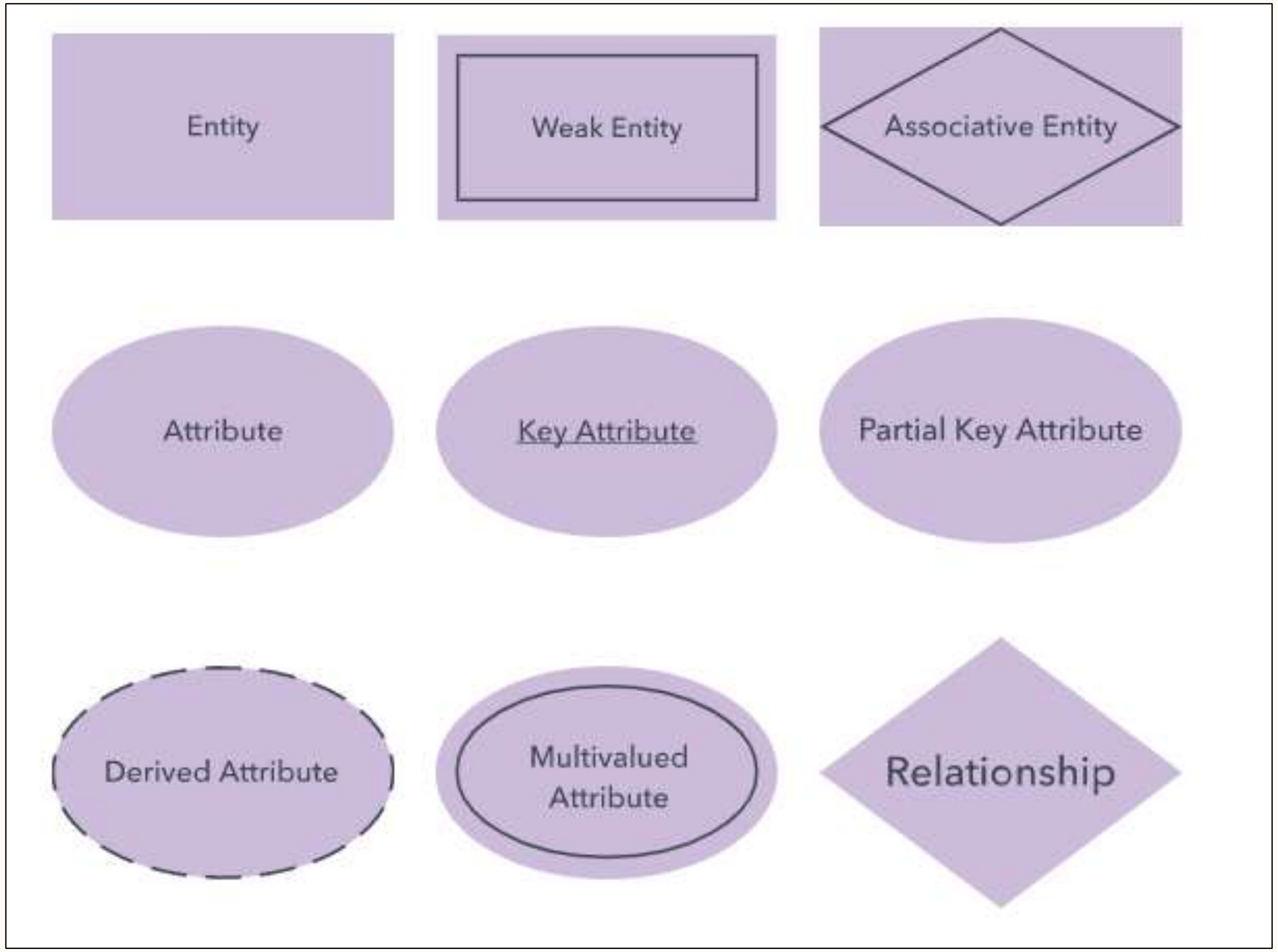
- ER Diagrams (ERD) can be used by database designers **as a blueprint for implementing data** in specific software applications.
- The database designer gains a better understanding of the information to be contained in the database with the help of ERD.
- ERD allows you to communicate with the logical structure of the database to users.

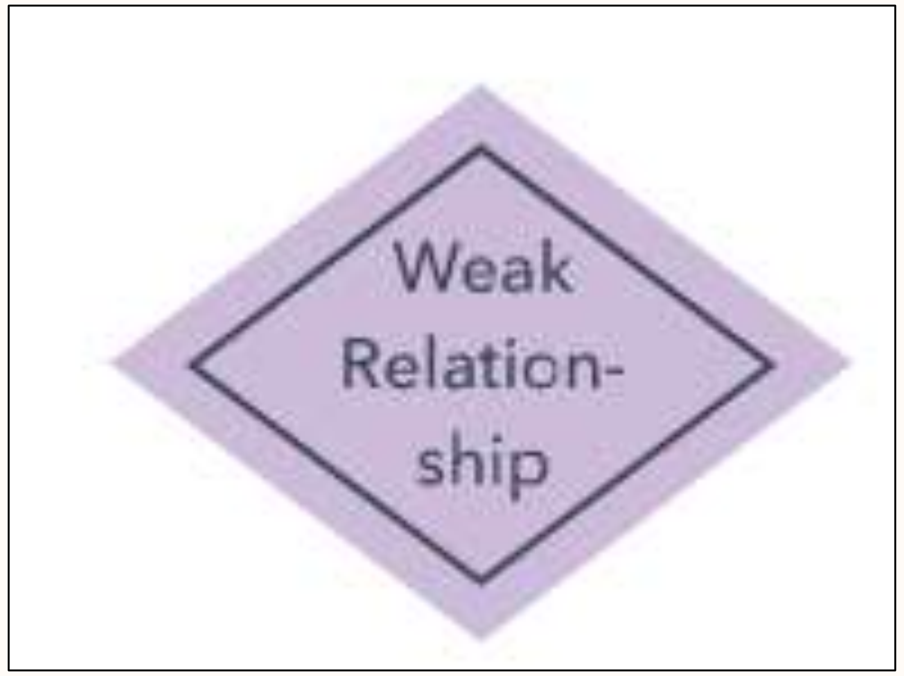


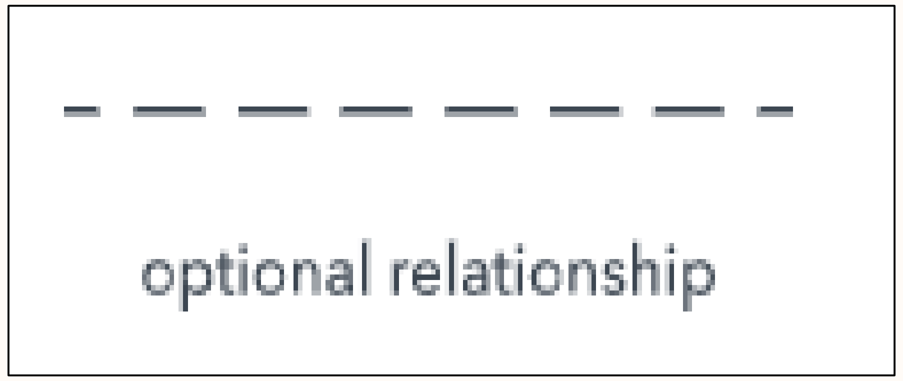
Entity Relationship Diagram - Notations

- **Rectangles:** This Entity Relationship Diagram symbol represents **entity types**
- **Ellipses** : Symbol represent **attributes**
- **Diamonds:** This symbol represents **relationship types**
- **Lines:** It **links attributes** to entity types and entity types with other relationship types
- **Primary Key:** attributes are **underlined**
- **Double Ellipses:** Represent **multi-valued attributes**

Entity Relationship Diagram- Notations

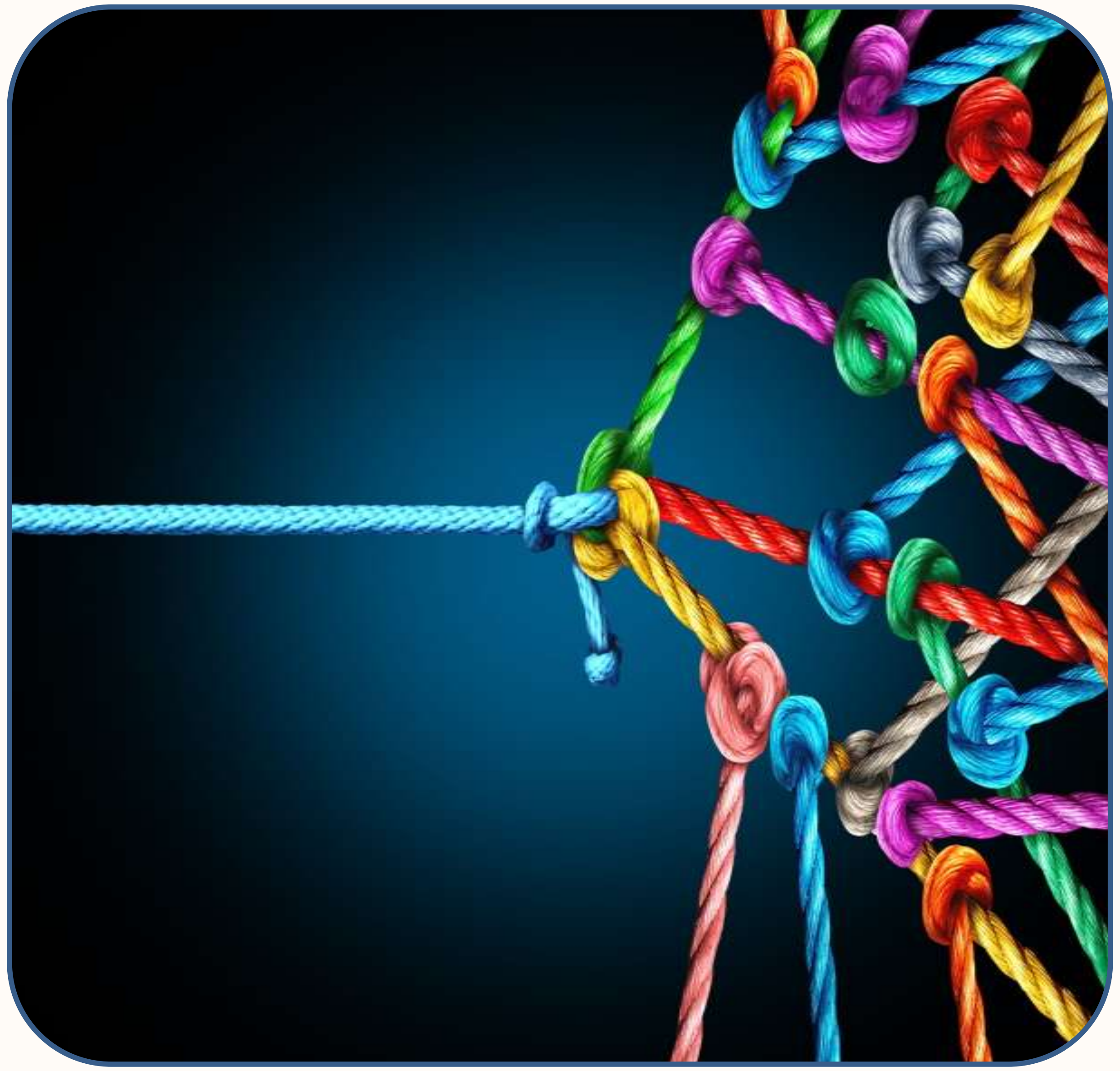




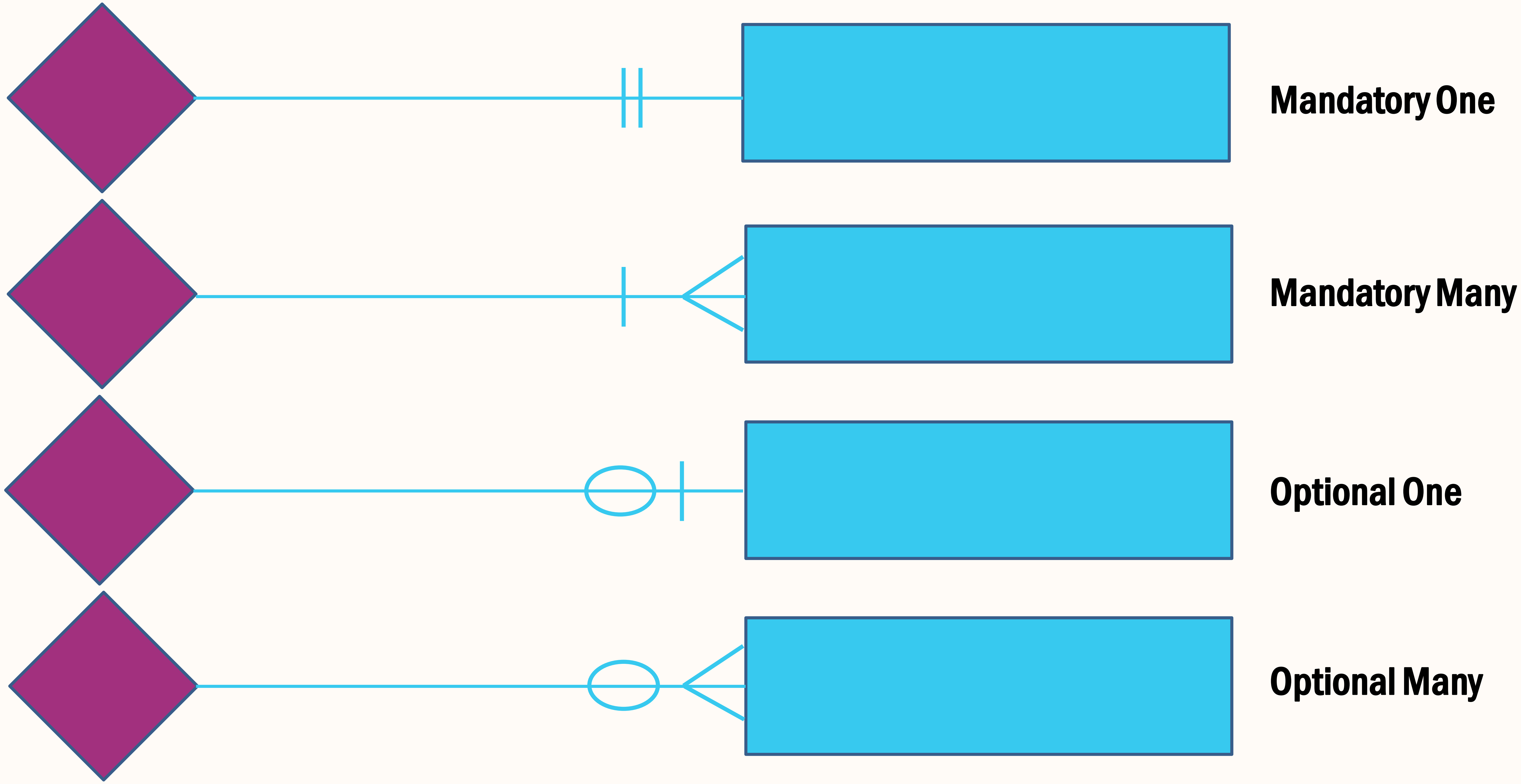


Relationship

- The degree of a relationship is the number of entity sets that participate in the relationship.
- Mostly binary relationships, sometimes more.
- Mapping cardinality of a relationship are:
  - 1 – 1
  - 1 – many
  - many – 1
  - Many-many



Relationship



Relationship Cardinality

Entity Relationship Diagram Steps

Entity Relationship Diagram Steps

1. **Identify the entities:**
   A. Identify all the entities you will use. This could be a customer, a manager, an invoice, a schedule, etc. Draw a rectangle for each entity you can think of on your page.
2. **Identify relationships:**
   A. Look at two entities, are they related?
   B. If so, draw a solid line connecting the two entities.
3. **Describe the relationship:**
   A. Find out how are the entities related?
   B. Draw an action diamond between the two entities on the line you just added.
   C. In the diamond write a brief description of how they are related

Entity Relationship Diagram Steps

4. **Add attributes:**
   A. Add the key attributes of entities using oval-shaped symbols.

5. **Complete the diagram:**
   A. Continue to connect the entities with lines and add diamonds to describe each relationship until all relationships have been described.

Entity Relationship Diagram Steps

Entity Name

Entity

Person, Place, Object, Event or Concept about which data is to be maintained.
Example: Car, Student

Relation

Verb Phrase

Association between the instances of one or more entity types
Example: Blue Car Belongs to Student Jack

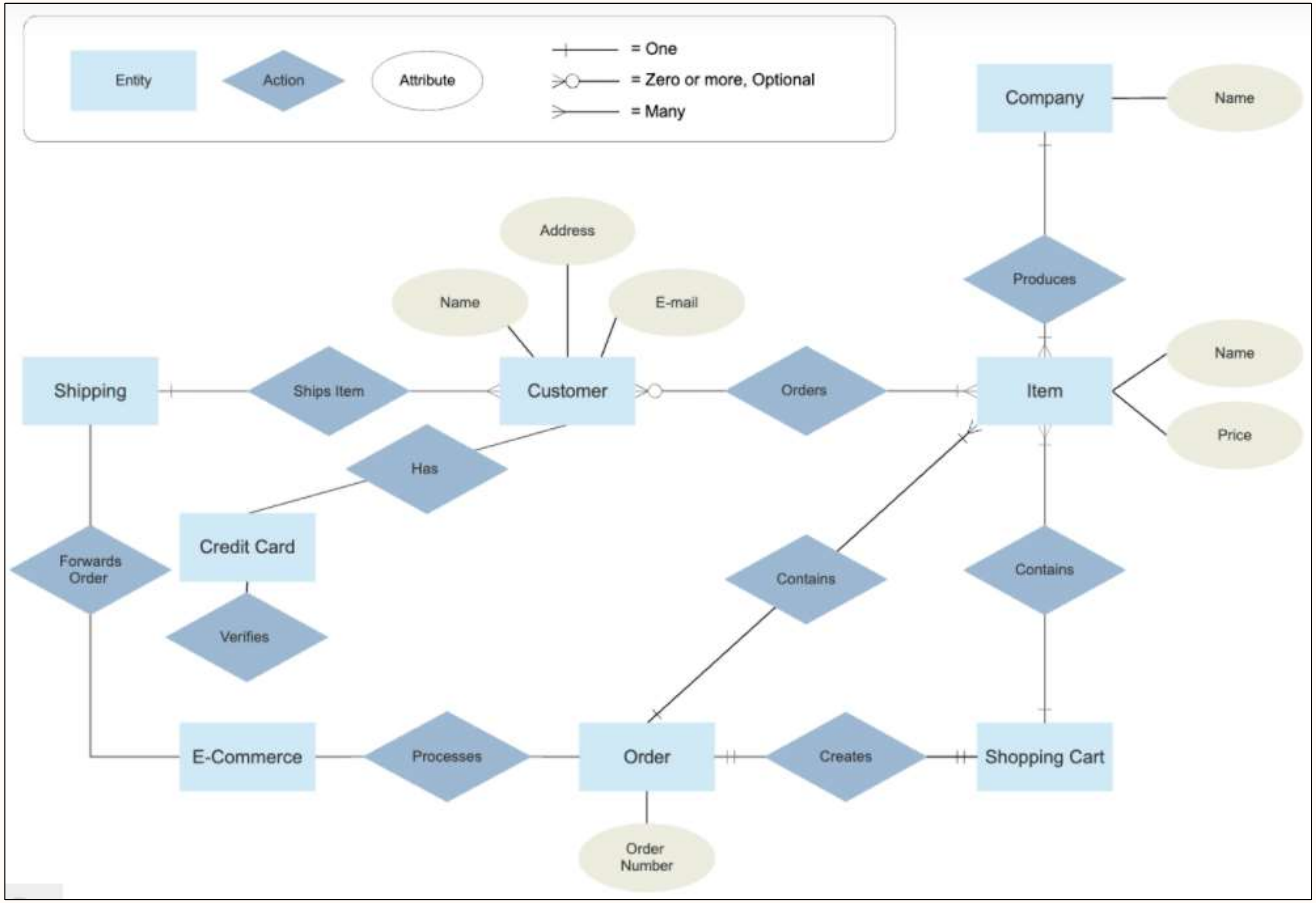
Jack

Attribute Name

Attribute

Property or Characteristic of an entity
Example: Colour of car Entity Name of Student Entity

Sample ER Diagram



ERD – Internal Sales Model

https://www.smartdraw.com/entity-relationship-diagram/

Database Normalization

- Database Normalization is a technique of **organizing the data in the database**.
- Normalization is a systematic approach of decomposing tables to eliminate data redundancy and undesirable characteristics like Insertion, Update, and Deletion Anomalies.
- It is a multi-step process that puts data into tabular form, removing duplicated data from the relation tables.

Normalization is used for mainly two purposes:

- **Eliminating redundant**(useless) data.
- **Ensuring data dependencies** make sense i.e., data is logically stored.

Levels of Normalization

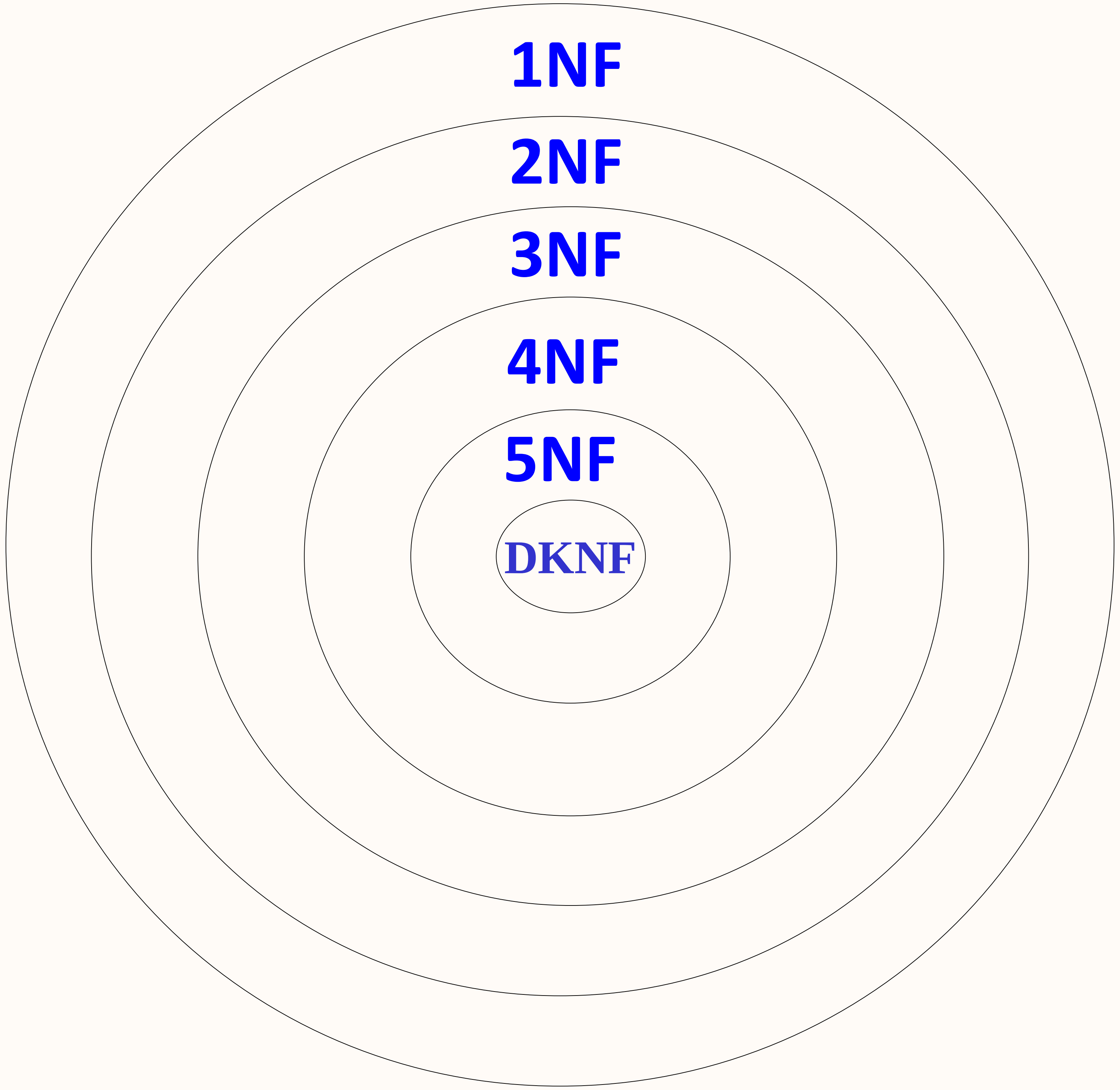
Levels of normalization based on the amount of redundancy in the database.

Various levels of normalization are:

- First Normal Form (1NF)
- Second Normal Form (2NF)
- Third Normal Form (3NF)
- Boyce-Codd Normal Form (BCNF)
- Fourth Normal Form (4NF)
- Fifth Normal Form (5NF)
- Domain Key Normal Form (DKNF)

Most databases should be 3NF or BCNF in order to avoid database anomalies.

Levels of Normalization



Each higher level is a subset of the lower level

First Normal Form (1NF)

A table is considered to be in 1NF if all the fields contain only scalar values (as opposed to list of values).

Example (Not 1NF)

ISBN	Title	AuName	AuPhone	PubName	PubPhone	Price
0-321-32132-1	Balloon	Sleepy, Snoopy, Grumpy	321-321-1111, 232-234-1234, 665-235-6532	Small House	714-000-0000	$34.00
0-55-123456-9	Main Street	Jones, Smith	123-333-3333, 654-223-3455	Small House	714-000-0000	$22.95
0-123-45678-0	Ulysses	Joyce	666-666-6666	Alpha Press	999-999-9999	$34.00
1-22-233700-0	Visual Basic	Roman	444-444-4444	Big House	123-456-7890	$25.00

Author and AuPhone columns are not scalar

First Normal Form (1NF)

1. **Place all items** that appear in the **repeating** group in a **new table**
2. **Designate a primary key** for each new table produced.
3. Duplicate in the new table the primary key of the table from which the repeating group was extracted or vice versa.

Example (1NF)

ISBN	Title	PubName	PubPhone	Price
0-321-32132-1	Balloon	Small House	714-000-0000	$34.00
0-55-123456-9	Main Street	Small House	714-000-0000	$22.95
0-123-45678-0	Ulysses	Alpha Press	999-999-9999	$34.00
1-22-233700-0	Visual Basic	Big House	123-456-7890	$25.00

ISBN	AuName	AuPhone
0-321-32132-1	Sleepy	321-321-1111
0-321-32132-1	Snoopy	232-234-1234
0-321-32132-1	Grumpy	665-235-6532
0-55-123456-9	Jones	123-333-3333
0-55-123456-9	Smith	654-223-3455
0-123-45678-0	Joyce	666-666-6666
1-22-233700-0	Roman	444-444-4444

Functional Dependencies

If one set of attributes in a table determines another set of attributes in the table, then the second set of attributes is said to be functionally dependent on the first set of attributes.

Example 1

ISBN	Title	Price
0-321-32132-1	Balloon	$34.00
0-55-123456-9	Main Street	$22.95
0-123-45678-0	Ulysses	$34.00
1-22-233700-0	Visual Basic	$25.00

Table Scheme: {ISBN, Title, Price}

Functional Dependencies: {ISBN} → {Title}

{ISBN} → {Price}

Functional Dependencies

Example 2

PubID	PubName	PubPhone
1	Big House	999-999-9999
2	Small House	123-456-7890
3	Alpha Press	111-111-1111

Table Scheme: {PubID, PubName, PubPhone}

Functional Dependencies:

$\{PubId\} \rightarrow \{PubPhone\}$

$\{PubId\} \rightarrow \{PubName\}$

$\{PubName, PubPhone\} \rightarrow \{PubID\}$

Functional Dependencies

Example 3

AuID	AuName	AuPhone
1	Sleepy	321-321-1111
2	Snoopy	232-234-1234
3	Grumpy	665-235-6532
4	Jones	123-333-3333
5	Smith	654-223-3455
6	Joyce	666-666-6666
7	Roman	444-444-4444

Table Scheme: {AuID, AuName, AuPhone}

Functional Dependencies: {AuId} → {AuPhone}

{AuId} → {AuName}

{AuName, AuPhone} → {AuID}

Activity

For an academic conference, prospective authors submit papers for review and possible acceptance in the published conference proceedings.

You are tasked to create a database to track reviews of papers submitted.

Identify the possible entities and their functional dependencies.

Functional Dependencies - Example

A database is required to track reviews of papers submitted to an academic conference. Prospective authors submit papers for review and possible acceptance in the published conference proceedings.

Details of the entities:

- Author information includes a unique author number, a name, a mailing address, and a unique (optional) email address.
- Paper information includes the primary author, the paper number, the title, the abstract, and review status (pending, accepted, rejected)
- Reviewer information includes the reviewer number, the name, the mailing address, and a unique (optional) email address
- A completed review includes the reviewer number, the date, the paper number, comments to the authors, comments to the program chairperson, and ratings (overall, originality, correctness, style, clarity)

Functional Dependencies - Example

Functional Dependencies

- AuthNo → AuthName, AuthEmail, AuthAddress
- AuthEmail → AuthNo
- PaperNo → Primary-AuthNo, Title, Abstract, Status
- RevNo → RevName, RevEmail, RevAddress
- RevEmail → RevNo
- RevNo, PaperNo → AuthComm, Prog-Comm, Date, Rating1, Rating2, Rating3, Rating4, Rating5

Second Normal Form (2NF)

For a table to be in 2NF, there are two requirements:

- The database should be in first normal form
- All non-key attributes in the table must be functionally dependent on the entire primary key

Example 1 (Not 2NF)

Scheme → {Title, PubId, AuId, Price, AuAddress}

1. Key → {Title, PubId, AuId}
2. {Title, PubId, AuID} → {Price}
3. {AuID} → {AuAddress}
4. AuAddress does not belong to a key
5. AuAddress functionally depends on AuId which is a subset of a key

Second Normal Form (2NF)

Example 2 (Not 2NF)

Scheme → {City, Street, HouseNumber, HouseColor, CityPopulation}

1. key → {City, Street, HouseNumber}
2. {City, Street, HouseNumber} → {HouseColor}
3. {City} → {CityPopulation}

CityPopulation does not belong to any key.

CityPopulation is functionally dependent on the City which is a proper subset of the key

Second Normal Form (2NF)

Example 3 (Not 2NF)

Scheme → {studio, movie, budget, studio_city}

1. Key → {studio, movie}
2. {studio, movie} → {budget}
3. {studio} → {studio_city}

studio_city is not a part of a key

studio_city functionally depends on studio which is a proper subset of the key

2NF - Decomposition

- If a data item is fully functionally dependent on only a part of the primary key, move that data item and that part of the primary key to a new table.
- If other data items are functionally dependent on the same part of the key, place them in the new table also
- Make the partial primary key copied from the original table the primary key for the new table. Place all items that appear in the repeating group in a new table

Example 1 (Convert to 2NF)

Old Scheme → {Title, PubId, AuId, Price, AuAddress}

New Scheme → {Title, PubId, AuId, Price}

New Scheme → {AuId, AuAddress}

2NF - Decomposition

Example 2 (Convert to 2NF)

- Old Scheme → {Studio, Movie, Budget, StudioCity}
- New Scheme → {Movie, Studio, Budget}
- New Scheme → {Studio, City}

2NF - Decomposition

Example 3 (Convert to 2NF)

- Old Scheme → {City, Street, HouseNumber, HouseColor, CityPopulation}
- New Scheme → {City, Street, HouseNumber, HouseColor}
- New Scheme → {City, CityPopulation}

Third Normal Form (3NF)

This form dictates that all non-key attributes of a table must be functionally dependent on a candidate key i.e., there can be no interdependencies among non-key attributes.

For a table to be in 3NF, there are two requirements:

- The table should be second normal form
- No attribute is transitively dependent on the primary key

Third Normal Form (3NF)

Example (Not in 3NF)

Scheme → {Title, PubID, PageCount, Price }

Key → {Title, PubId}

{Title, PubId} → {PageCount}

{PageCount} → {Price}

Both Price and PageCount depend on a key hence 2NF

Transitively {Title, PubID} → {Price} hence not in 3NF

Third Normal Form (3NF)

Example 2 (Not in 3NF)

Scheme → {Studio, StudioCity, CityTemp}

Primary Key → {Studio}

{Studio} → {StudioCity}

{StudioCity} → {CityTemp}

{Studio} → {CityTemp}

Both StudioCity and CityTemp depend on the entire key hence 2NF

CityTemp transitively depends on Studio hence violates 3NF

Third Normal Form (3NF)

Example 3 (Not in 3NF)

Scheme → {BuildingID, Contractor, Fee}

Primary Key → {BuildingID}

{BuildingID} → {Contractor}

{Contractor} → {Fee}

{BuildingID} → {Fee}

BuildingID	Contractor	Fee
100	Randolph	1200
150	Ingersoll	1100
200	Randolph	1200
250	Pitkin	1100
300	Randolph	1200

Fee transitively depends on the BuildingID

Both Contractor and Fee depend on the entire key hence 2NF

3NF - Decomposition

- Move all items involved in transitive dependencies to a new entity.
- Identify a primary key for the new entity.
- Place the primary key for the new entity as a foreign key on the original entity.

Example 3 (Not in 3NF)

Old Scheme → {Title, PubID, PageCount, Price }

New Scheme → {PubID, PageCount, Price}

New Scheme → {Title, PubID, PageCount}

3NF - Decomposition

Example 2 (Convert to 3NF)

- Old Scheme → {Studio, StudioCity, CityTemp}
- New Scheme → {Studio, StudioCity}
- New Scheme → {StudioCity, CityTemp}

Example 3 (Convert to 3NF)

- Old Scheme → {BuildingID, Contractor, Fee}
- New Scheme → {BuildingID, Contractor}
- New Scheme → {Contractor, Fee}

BuildingID	**Contractor**
100	Randolph
150	Ingersoll
200	Randolph
250	Pitkin
300	Randolph

Contractor	Fee
Randolph	1200
Ingersoll	1100
Pitkin	1100

Boyce-Codd Normal Form (BCNF)

BCNF does not allow dependencies between attributes that belong to candidate keys.

BCNF is a refinement of the third normal form in which it drops the restriction of a non-key attribute from the 3rd normal form.

Third normal form and BCNF are not same if the following conditions are true:

- The table has two or more candidate keys
- At least two of the candidate keys are composed of more than one attribute
- The keys are not disjoint i.e. The composite candidate keys share some attributes

Boyce-Codd Normal Form (BCNF)

Example 1 - Address (Not in BCNF)

Scheme → {City, Street, ZipCode }

Key1 → {City, Street }

Key2 → {ZipCode, Street}

No non-key attribute hence 3NF

{City, Street} → {ZipCode}

{ZipCode} → {City}

Dependency between attributes belonging to a key.

Boyce-Codd Normal Form (BCNF)

Example 2 - Movie (Not in BCNF)

Scheme → {MovieTitle, MovieID, PersonName, Role, Payment }

1. Key1 → {MovieTitle, PersonName}
2. Key2 → {MovieID, PersonName}
3. Both role and payment functionally depend on both candidate keys thus 3NF
4. {MovieID} → {MovieTitle}
5. Dependency between MovieID & MovieTitle Violates BCNF

Boyce-Codd Normal Form (BCNF)

Example 3 - Consulting (Not in BCNF)

Scheme → {Client, Problem, Consultant}

1. Key1 → {Client, Problem}
2. Key2 → {Client, Consultant}
3. No non-key attribute hence 3NF
4. {Client, Problem} → {Consultant}
5. {Client, Consultant} → {Problem}
6. Dependency between attributess belonging to keys violates BCNF

BCNF - Decomposition

1. Place the two candidate primary keys in separate entities
2. Place each of the remaining data items in one of the resulting entities according to its dependency on the primary key.

Example 3 - Consulting (Not in BCNF)

- Old Scheme → {City, Street, ZipCode }
- New Scheme1 → {ZipCode, Street}
- New Scheme2 → {City, Street}
- Loss of relation {ZipCode} → {City}
- Alternate New Scheme1 → {ZipCode, Street }
- Alternate New Scheme2 → {ZipCode, City}

Decomposition – Loss of Information

1. If decomposition does not cause any loss of information, it is called a lossless decomposition.
2. If a decomposition does not cause any dependencies to be lost it is called a dependency-preserving decomposition.
3. Any table scheme can be decomposed in a lossless way into a collection of smaller schemas that are in BCNF form. However, dependency preservation is not guaranteed.
4. Any table can be decomposed in a lossless way into 3rd normal form that also preserves the dependencies.
   A. 3NF may be better than BCNF in some cases

Use your own judgment when decomposing schemas

BCNF - Decomposition

Example 2 (Convert to BCNF)

- Old Scheme → {MovieTitle, MovieID, PersonName, Role, Payment }
- New Scheme → {MovieID, PersonName, Role, Payment}
- New Scheme → {MovieTitle, PersonName}

- Loss of relation {MovieID} → {MovieTitle}
  - New Scheme → {MovieID, PersonName, Role, Payment}
  - New Scheme → {MovieID, MovieTitle}
- We got the {MovieID} → {MovieTitle} relationship back

BCNF - Decomposition

Example 3 (Convert to BCNF)

- Old Scheme → {Client, Problem, Consultant}
- New Scheme → {Client, Consultant}
- New Scheme → {Client, Problem}

Fourth Normal Form (4NF)

Fourth normal form eliminates independent many-to-one relationships between columns.
To be in Fourth Normal Form, following conditions should exist:

- a relation must first be in Boyce-Codd Normal Form.
- a given relation may not contain more than one multi-valued attribute.

Example (Not in 4NF)

Scheme → {MovieName, ScreeningCity, Genre)
Primary Key: {MovieName, ScreeningCity, Genre)
All columns are a part of the only candidate key, hence BCNF
Many Movies can have the same Genre Many Cities can have the same movie
Violates 4NF

Movie	ScreeningCity	Genre
Hard Code	Los Angles	Comedy
Hard Code	New York	Comedy
Bill Durham	Santa Cruz	Drama
Bill Durham	Durham	Drama
The Code Warrier	New York	Horror

Fourth Normal Form (4NF)

Example 2 (Not in 4NF)

1. Scheme → {Manager, Child, Employee}
2. Primary Key → {Manager, Child, Employee}
3. Each manager can have more than one child
4. Each manager can supervise more than one employee
5. 4NF Violated

Manager	Child	Employee
Jim	Beth	Alice
Mary	Bob	Jane
Mary	NULL	Adam

Example 3 (Not in 4NF)

1. Scheme → {Employee, Skill, ForeignLanguage}
2. Primary Key → {Employee, Skill, Language }
3. Each employee can speak multiple languages
4. Each employee can have multiple skills
5. Thus, violates 4NF

Employee	Skill	Language
1234	Cooking	French
1234	Cooking	German
1453	Carpentry	Spanish
1453	Cooking	Spanish
2345	Cooking	Spanish

4NF - Decomposition

1. Move the two multi-valued relations to separate tables
2. Identify a primary key for each of the new entities.

Example 1 (Convert to 3NF)

- Old Scheme → {MovieName, ScreeningCity, Genre}
- New Scheme → {MovieName, ScreeningCity}
- New Scheme → {MovieName, Genre}

Movie	Genre
Hard Code	Comedy
Bill Durham	Drama
The Code Warrier	Horror

Movie	ScreeningCity
Hard Code	Los Angles
Hard Code	New York
Bill Durham	Santa Cruz
Bill Durham	Durham
The Code Warrier	New York

4NF - Decomposition

Example 2 (Convert to 4NF)

- Old Scheme → {Manager, Child, Employee}
- New Scheme → {Manager, Child}
- New Scheme → {Manager, Employee}

Manager	Child
Jim	Beth
Mary	Bob

Manager	Employee
Jim	Alice
Mary	Jane
Mary	Adam

Example 3 (Convert to 4NF)

- Old Scheme → {Employee, Skill, ForeignLanguage}
- New Scheme → {Employee, Skill}
- New Scheme → {Employee, ForeignLanguage}

Employee	Skill
1234	Cooking
1453	Carpentry
1453	Cooking
2345	Cooking

Employee	Language
1234	French
1234	German
1453	Spanish
2345	Spanish

Fifth Normal Form (5NF)

Fifth normal form is satisfied when all tables are broken into as many tables as possible in order to avoid redundancy. Once it is in fifth normal form it cannot be broken into smaller relations without changing the facts or the meaning.

Activity 1 A: Drawing ER Diagram

Construct an E-R diagram for a car-insurance company whose customers own one or more cars each.

Each car has associated with it zero to any number of recorded accidents.

Activity 1 B: Drawing ER Diagram

Suppose you are given the following requirements for a simple database for the

National Hockey League (NHL):

- The NHL has many teams
- Each team has a name, a city, a coach, a captain, and a set of players
- Each player belongs to only one team
- Each player has a name, a position (such as left wing or goalie), a skill level
- A set of injury records
- A team captain is also a player
- A game is played between two teams (referred to as host team and guest team) and has a date (such as May 11th, 1999) and a score (such as 4 to 2)
- Construct a clean and concise ER diagram for the NHL database

Activity 2: Normalization

For the example, below we have one big table.

Put the table in normalized form.

OID = Order ID, O_Date= Order Date, CID = Customer ID,

C_Name = Customer Name, C_State = Customer's State, PID = project id,

P_Desc =Project Name, P_Price = Product Price, Qty = Quantity Purchased

Note: 7, 5, 4 means three Product IDs. Similarly, 1, 1, 5 means three Quantities.

Functional Dependencies are:

OID -> O_Date CID -> C_Name PID -> P_Desc PID -> P_Price

OID -> CID CID -> C_State PID and OID -> Qty

Activity 2: Normalization

OID	O_Date	CID	C_Name	C_State	PID	P_Desc	P_Price	Qty
1006	10/24/09	2	Apex	NC	7, 5, 4	Table, Desk, Chair	800, 325, 200	1, 1, 5
1007	10/25/09	6	Acme	GA	11, 4	Dresser, Chair	500, 200	4, 6

Questions?

Summary

In this module, you have learned:

- To work with real-world data, it is supposed to be organized and stored in a systematic manner
- Data Modelling helps us achieve the same
- The different ways in which data is modeled have a different purpose
- Understanding the way to model data using ER diagrams and refining the storage schema by normalization can help in many ways