DATA MODELLING AND DATABASE DESIGN

BUSINESS INTELLIGENCE & DATA WAREHOUSING — TOPIC 2

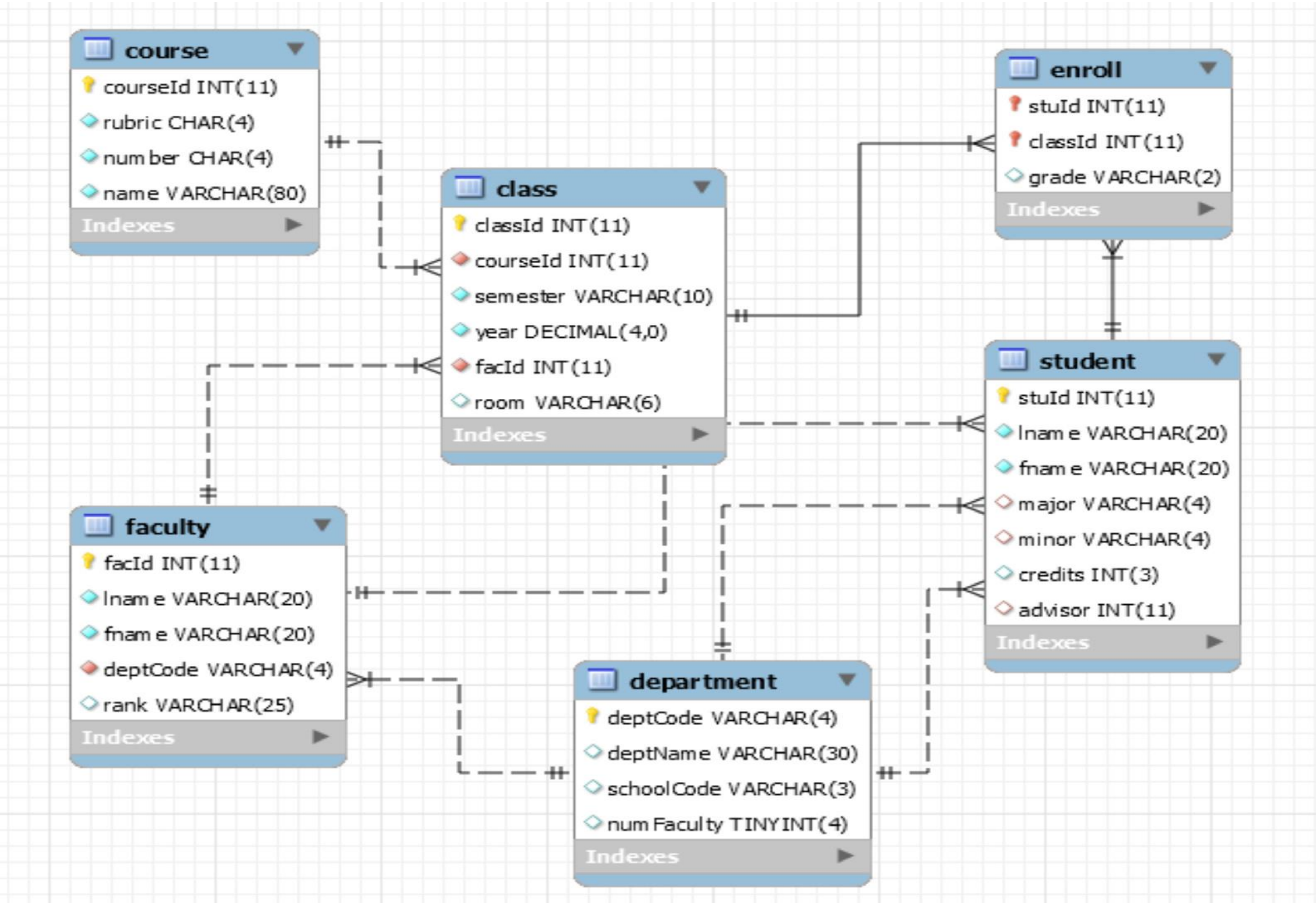
Objectives

- Understand the importance of data modeling in database design.
- Create Entity Relationship Diagrams (ERD) and describe their role in visualizing data.
- Describe fundamental concepts of relational databases and normalization.
- Apply dimensional modelling principles to design a data warehouse structure.
- Create effective fact and dimension tables for a specific business scenario

Data Modelling & ERDs

- Data modelling refers to the process of developing visual representations of the data that will be kept in the database.
- The visual representations of data modeling are diagrams that show data elements, their relationships and their constraints i.e. ERDs.
- Data modeling and entity relationship diagrams (ERDs) are like the blueprints for organizing information in a structured way.
- Data modelling can be done at three levels; ***conceptual model, logical model*** and ***physical model.***

Data Modelling & ERDs ..



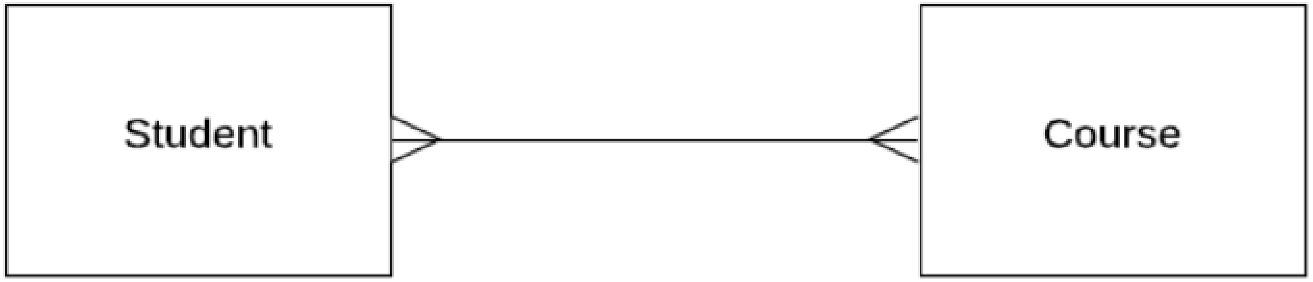
Why Data Modelling?

- Organizing Information: It helps to organize complex information into a structured format, making it easier to understand and manage.
- Improves communication because it t provides a common language for stakeholders to discuss and understand the data requirements of a system.
- Data models serve as blueprints for database design and application development thus guiding the development process.
- Enhances data quality by defining clear structures and relationships, data modeling improves the quality and consistency of data.
- Improves system performance due to properly modeled data structures that optimize data retrieval and manipulation processes.

Levels of Data Modelling - Conceptual

- The **conceptual data model** provides a high-level overview of the system's contents.
- It focuses on what the system contains rather than how it will be implemented and thus it answers the “what” of the system.
- For example, consider designing a system for a university. At the conceptual level, the model would identify entities such as “Student” and “Course”
- Relationships between these entities, like "Enroll" would also be defined.
- This level of abstraction allows stakeholders to understand the system's requirements without getting into technical details.

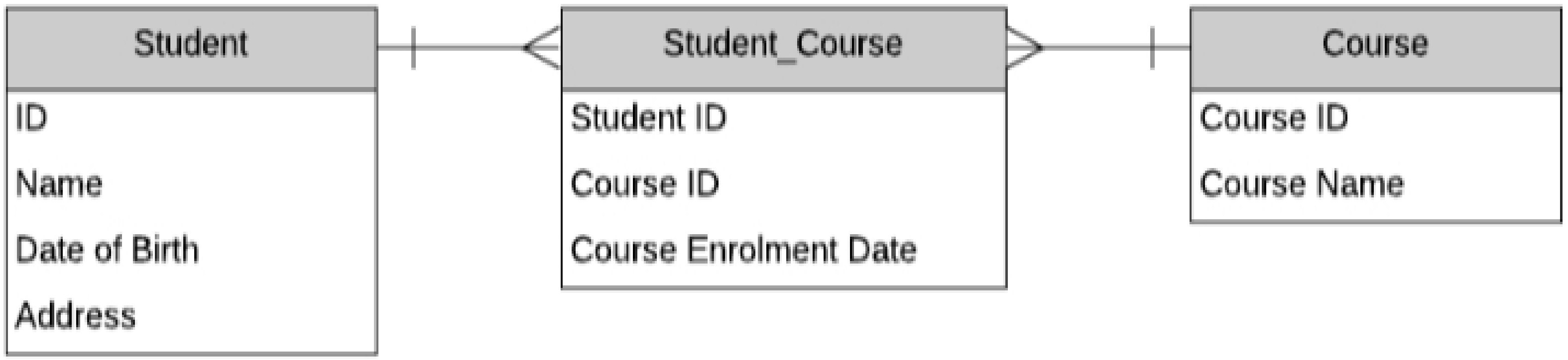
Levels of Data Modelling - Conceptual



Levels of Data Modelling - Logical

- The logical data model provides a more detailed view of the data structures and relationships.
- Data architects and analysts develop this model, focusing on creating a technical map of data structures and rules.
- For instance, in the university system, the logical model would specify attributes for entities such as “Student ID”, “Course Code”
- Primary keys, foreign keys, and relationships between entities would also be defined.
- This level ensures that the database design is robust and comprehensive, independent of the specific technology or DBMS used for implementation.

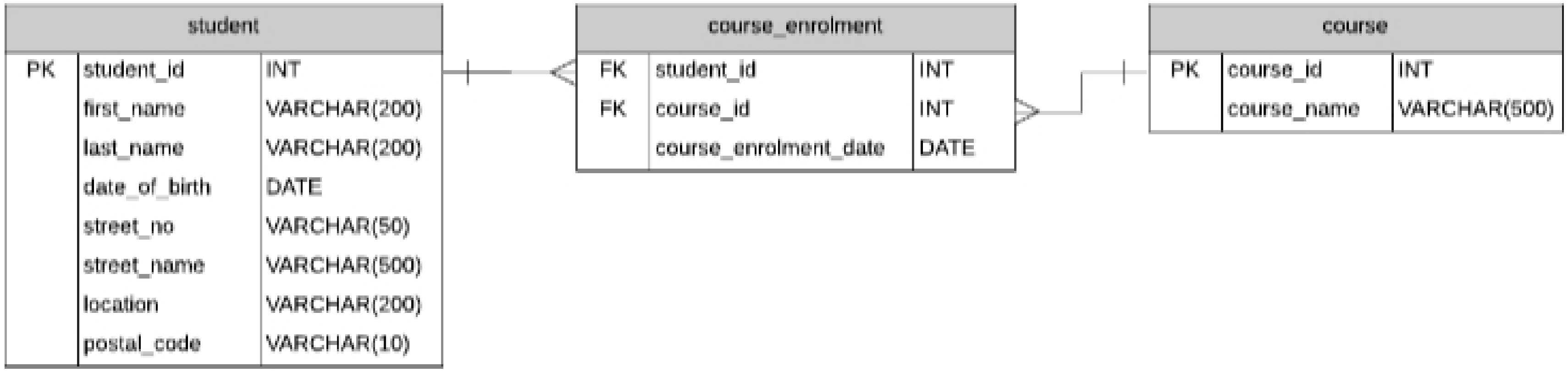
Levels of Data Modelling - Logical



Levels of Data Modelling - Physical

- At the physical data model level, the focus shifts to how the system will be implemented using a particular DBMS or technology.
- Database administrators and developers construct this model, specifying details such as table names, column names, and data types.
- Continuing with the university example, the physical model would include specifics like "Student" table with columns such as "Student_ID," "Name," and "Address."
- It also accounts for optimizations and constraints required for efficient database operations, considering the chosen DBMS's capabilities and requirements.

Levels of Data Modelling - Physical



student

PK	student_id	INT
	first_name	VARCHAR(200)
	last_name	VARCHAR(200)
	date_of_birth	DATE
	street_no	VARCHAR(50)
	street_name	VARCHAR(500)
	location	VARCHAR(200)
	postal_code	VARCHAR(10)

course_enrolment

FK	student_id	INT
FK	course_id	INT
	course_enrolment_date	DATE

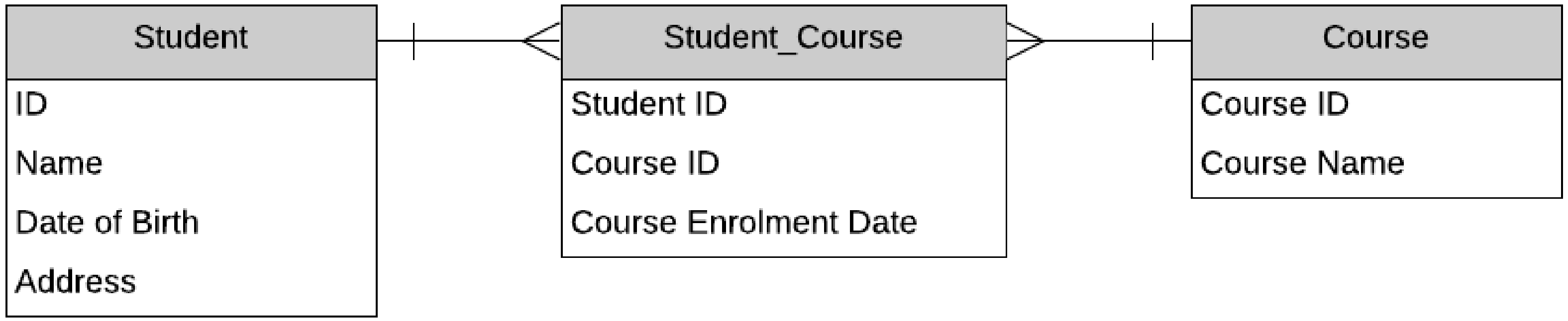
course

PK	course_id	INT
	course_name	VARCHAR(500)

Entity Relationship Diagrams (ERDs)

- ERDs are graphical representations used to visualize relationships between or amongst entities (e.g. people, places, objects, etc.) in a database or a system.
- They help to visualize and communicate the structure and dependencies of data
- ERDs help to collect and understand the data requirements of a system
- ERDs provide a blueprint for the database schema, enabling the effective planning of entities, attributes, and relationships
- ERDs can be created at three levels: conceptual, logical, and physical
- ERDs use symbols and notations to represent entities, attributes, and relationships

Entity Relationship Diagrams (ERDs)



Source: *A Guide to the Entity Relationship Diagram (ERD)*

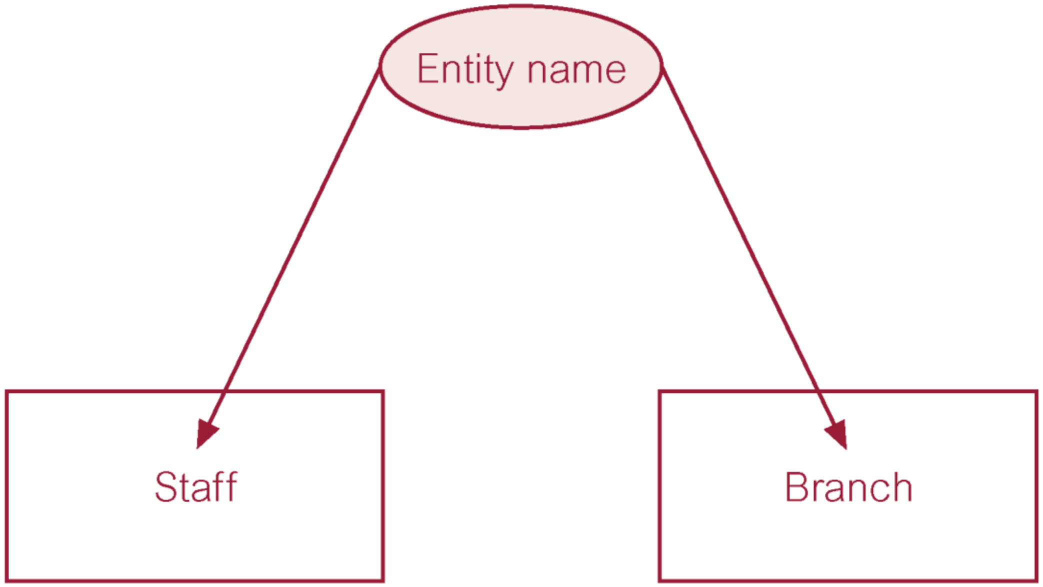
Component of ERDs

- **Entities**
- **Attributes**
- **Keys**
- **Relationship**

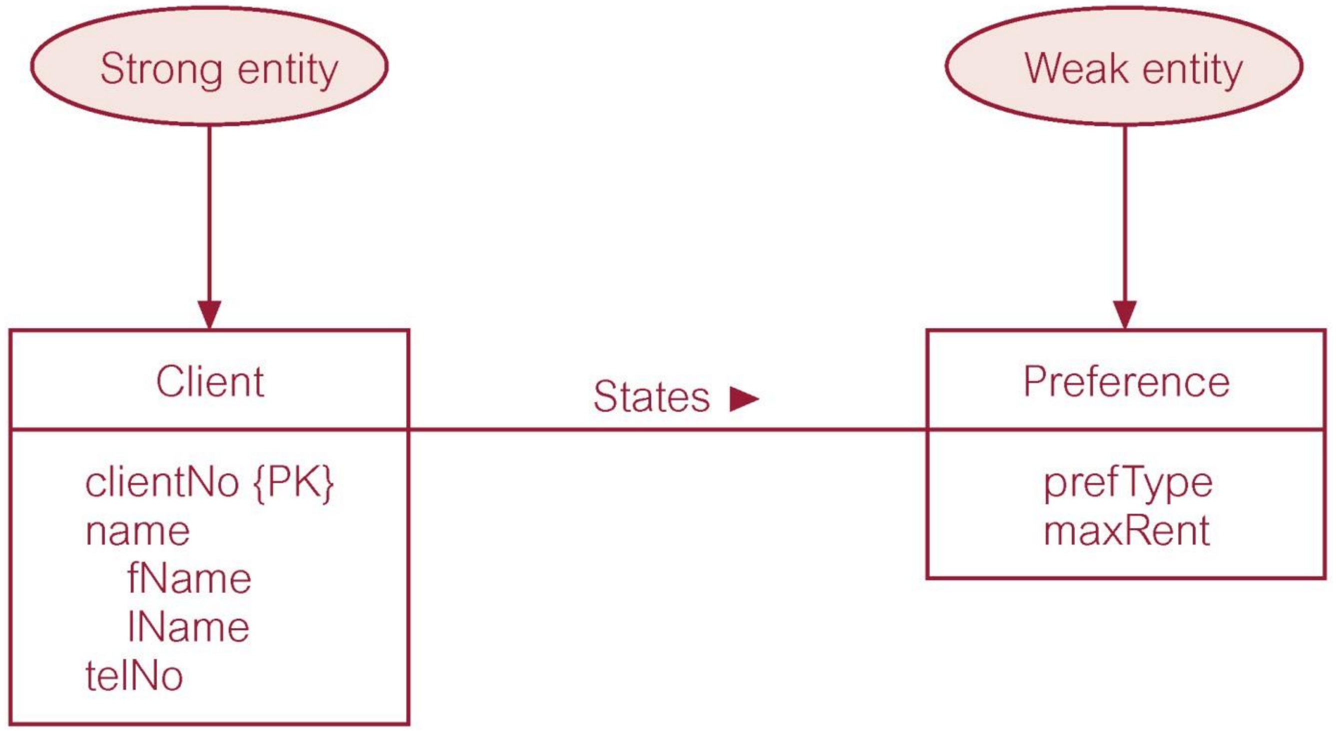
Components of ERDs - Entity

- An entity is anything that can have data stored about it in a database. It can be physical objects, concepts or events (e.g. order, payment, etc.)
- Entities represent a noun in natural language and they are usually represented by rectangles in ER diagrams with entity name inside.
- An entity can be of two types i.e. *strong entity or a weak entity.*
- A strong entity has its own unique ID (primary key) and doesn't rely on other entities. For instance, a student with a student ID is a strong entity as it exists independently.
- Weak entities do not have their own primary keys. Instead, they rely on another entity, known as the identifying or parent entity, for their existence.

Components of ERDs – Entity..



Components of ERDs – Entity..



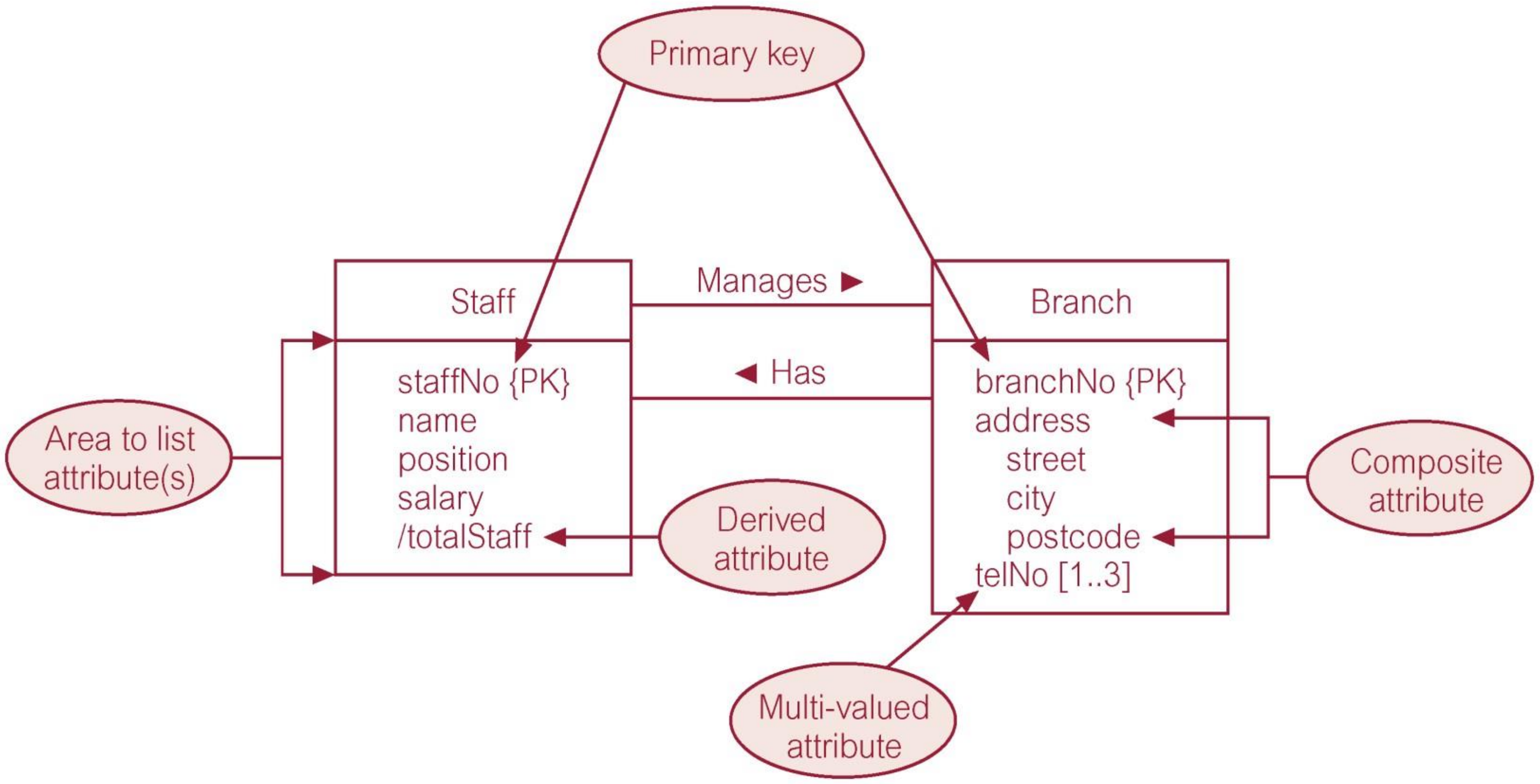
Components Of ERDs – Attributes

- Attribute is a characteristic or property of an entity, for example for a student entity, the possible attributes are registration number, student number, names, date of birth, etc.
- Attributes are represented by ovals or text in ERDs.
- Attribute Domain. Set of allowable values for one or more attributes. For example the types of values, whether numeric or alphabet, the number of digits, etc
- Attributes types include: simple attributes, compound attributes, derived attributes, single-valued attributes and multi-valued attributes.
- Simple attributes are those that can’t be split into other attributes e.g. first name.

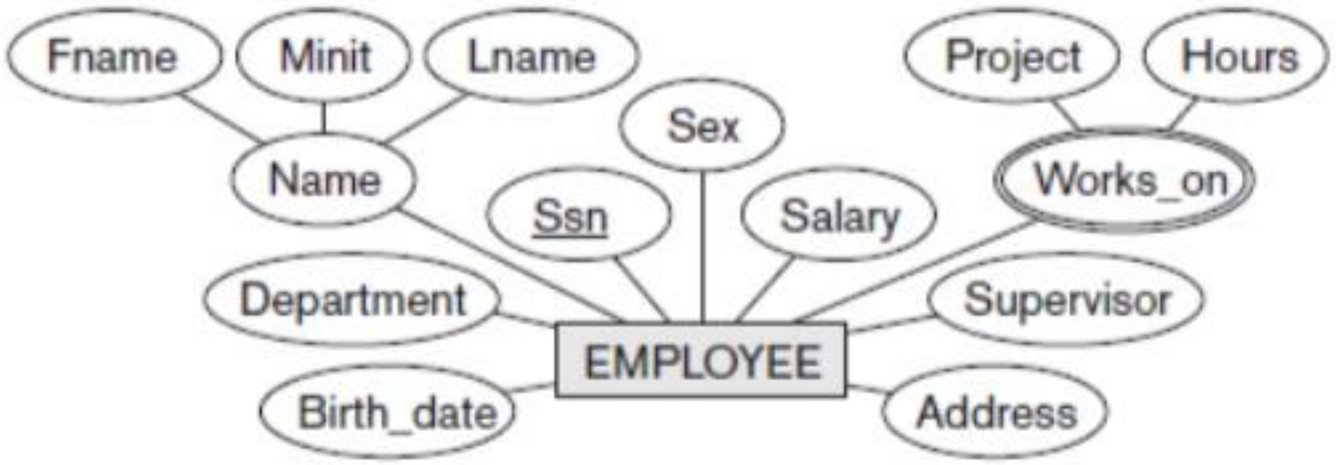
Components Of ERDs – Attributes

- Compound attributes can be split into other attributes e.g. location can be split into street address, city, country, etc.
- Single-valued attributes are those that take a single value e.g age.
- Multi-valued attributes are those that can take multi values for the same occurrence e.g. academic qualifications.
- Derived attributes are those whose value can be derived from other attributes.

Components Of ERDs – Attributes



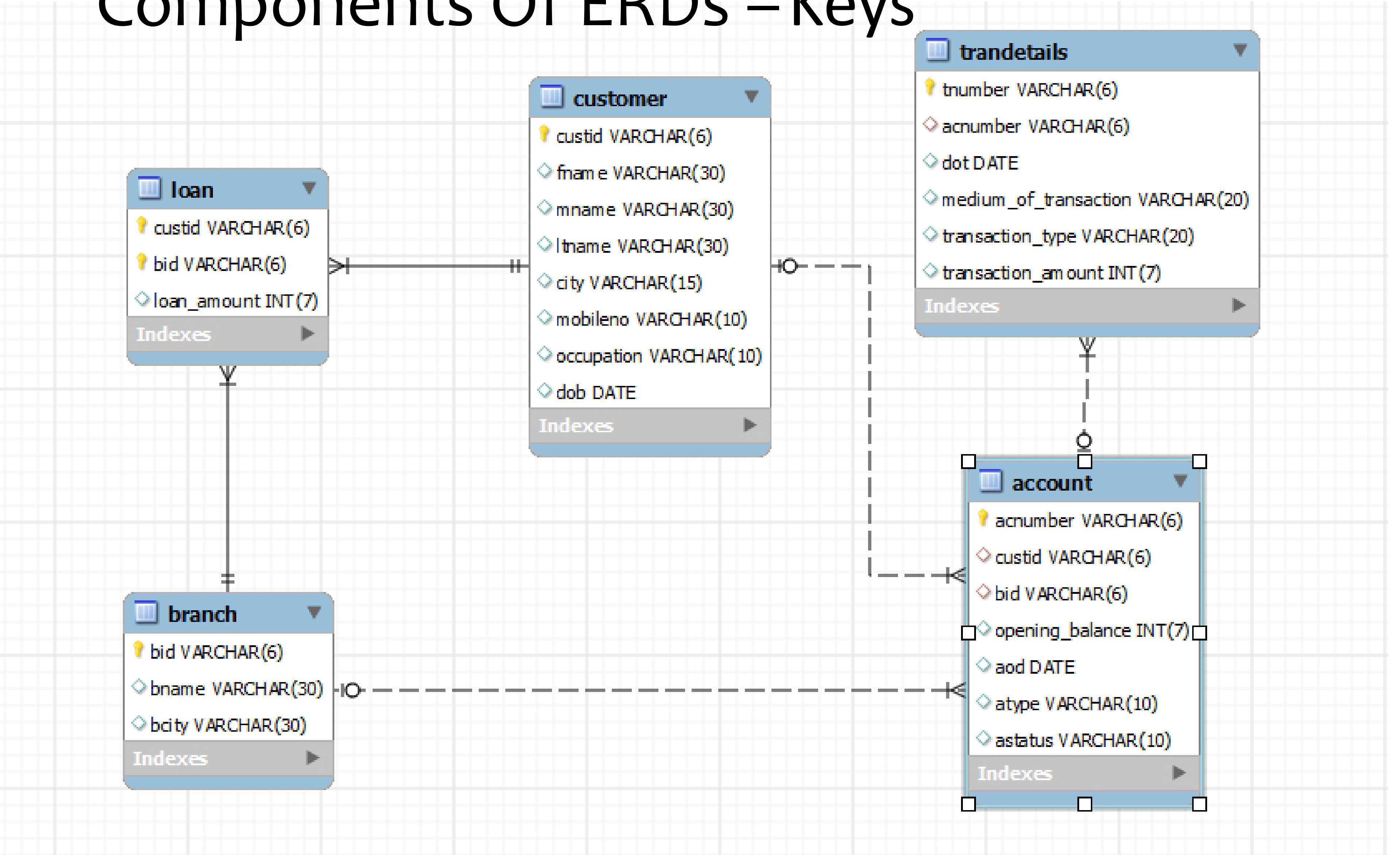
Components Of ERDs – Attributes



Components Of ERDs – Keys

- It is a special attribute or a set of attributes that uniquely identifies an entity or a relationship, such as a student ID, a course ID, or a combination of both.
- Keys are represented by underlining the attribute name in ER diagrams.
- Candidate key is a set of attributes that uniquely identify each record in the a table.
- Primary key is candidate key selected to uniquely identify each record in the table.
- Alternate key is a candidate key that is not selected to be primary key.
- Foreign key is an attribute in one table that refers to the primary key of another table. It creates a relationship between two tables, allowing us to link records in different tables.
- Composite Key is A candidate key that consists of two or more attributes.

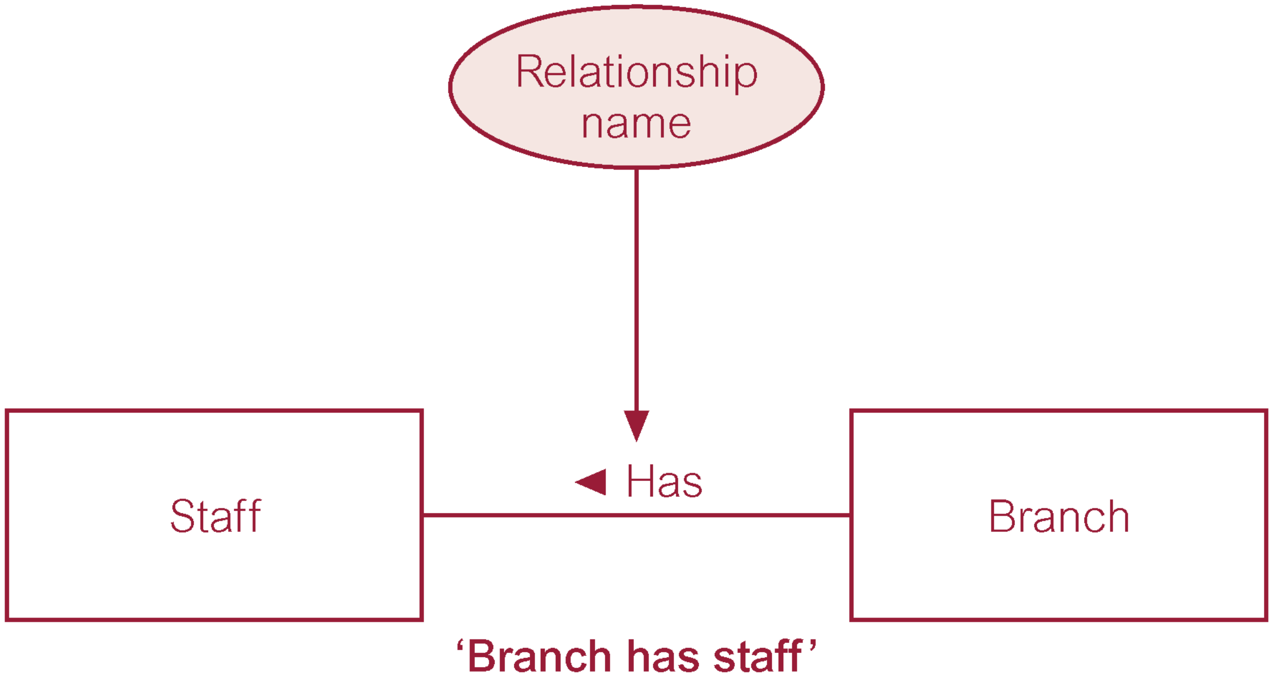
Components Of ERDs – Keys



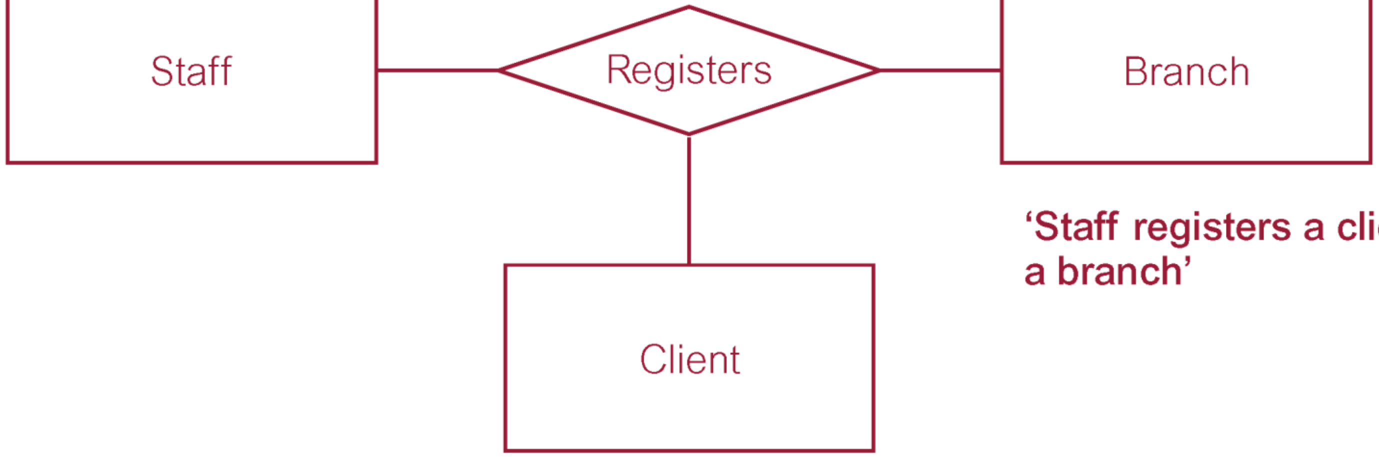
Components of ERDs - Relationship

- Derived from the fact that a database consists of related data meaning that the entities must be related according to the application being developed.
- A relationship in an ERD defines how two entities are related to each other. They can be derived from verbs when speaking about a database or a set of entities.
- Relationships in ERDs are represented as lines between two entities, and often have a label on the line to further describe the relationship or diamonds in ERDs.
- Relationship types include recursive relationship, identifying relationship, non-identifying relationship, mandatory relationship and optional relationship.

Components of ERDs - Relationship



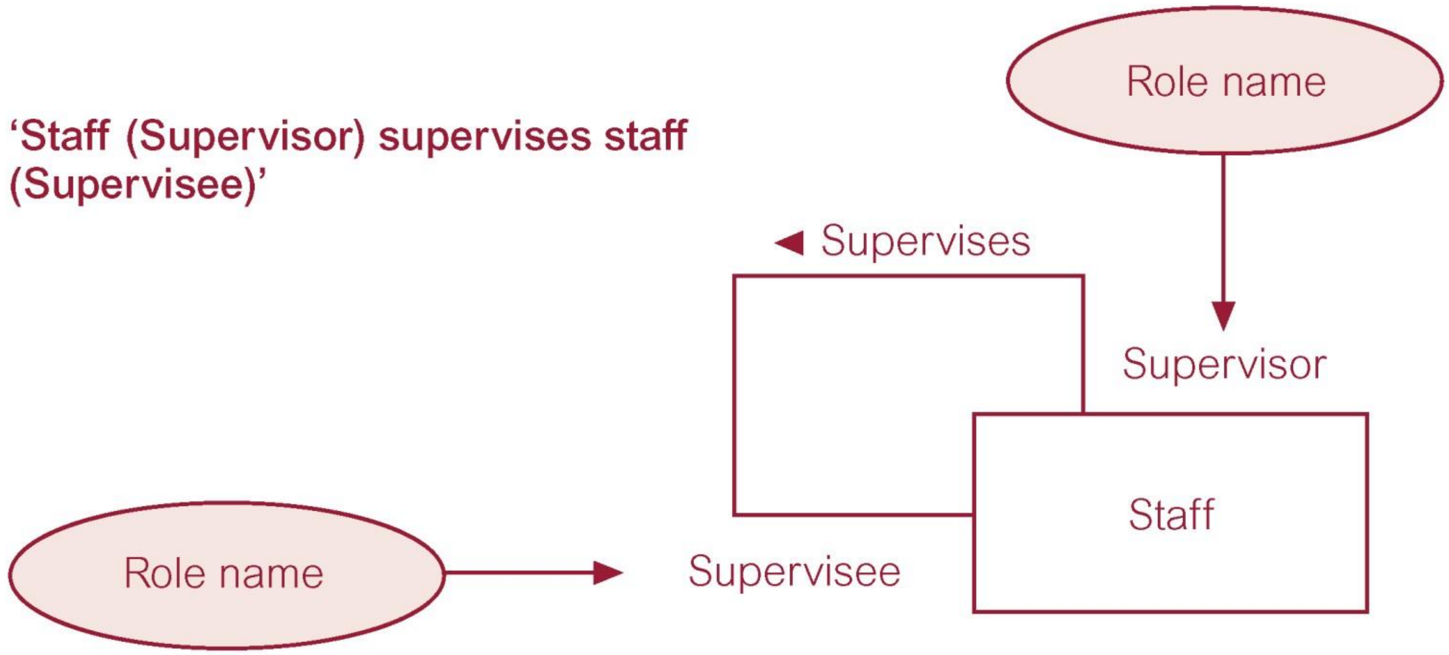
Components of ERDs - Relationship



‘Staff registers a client at a branch’

Components of ERDs - Relationship Types

- ***Recursive relationship:*** A relationship where an entity is related to itself in some way. For example, an employee can be a supervisor of another employee.



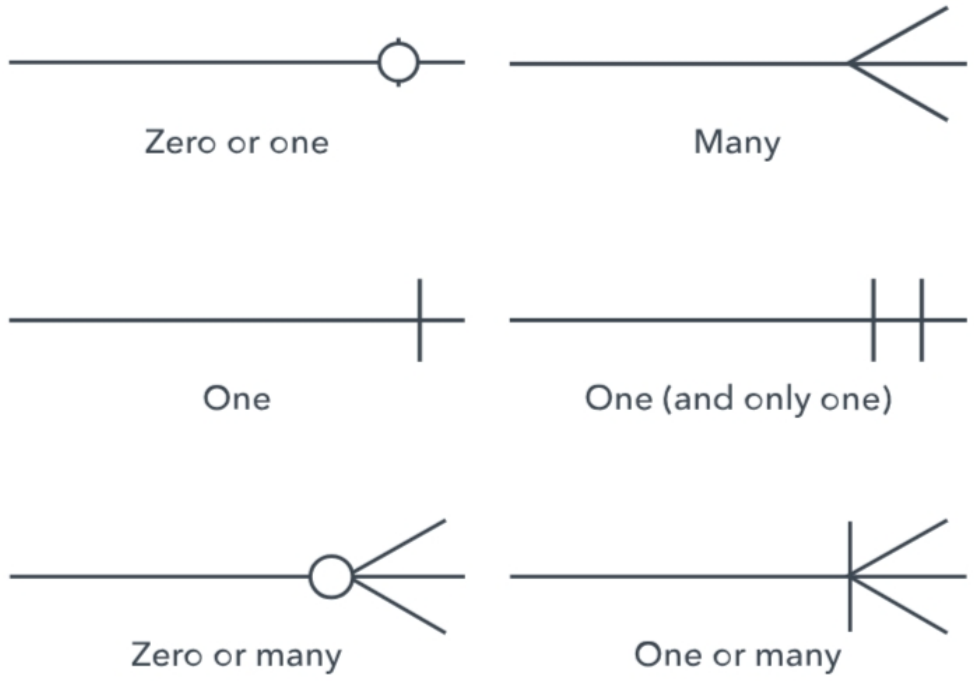
Components of ERDs - Relationship Types

- ***Identifying relationship:*** A relationship where a weak entity depends on a strong entity for its existence and identity. For example, an order line item depends on an order for its existence and identity.
- ***Non-identifying relationship:*** A relationship where an entity does not depend on another entity for its existence and identity. For example, an order does not depend on a customer for its existence and identity, even though it has a relationship with a customer.
- ***Mandatory relationship:*** A relationship where an entity must participate in the relationship with another entity. For example, a student must enrol in at least one course.
- ***Optional relationship:*** A relationship where an entity may or may not participate in the relationship with another entity. For example, a student may or may not have an address.

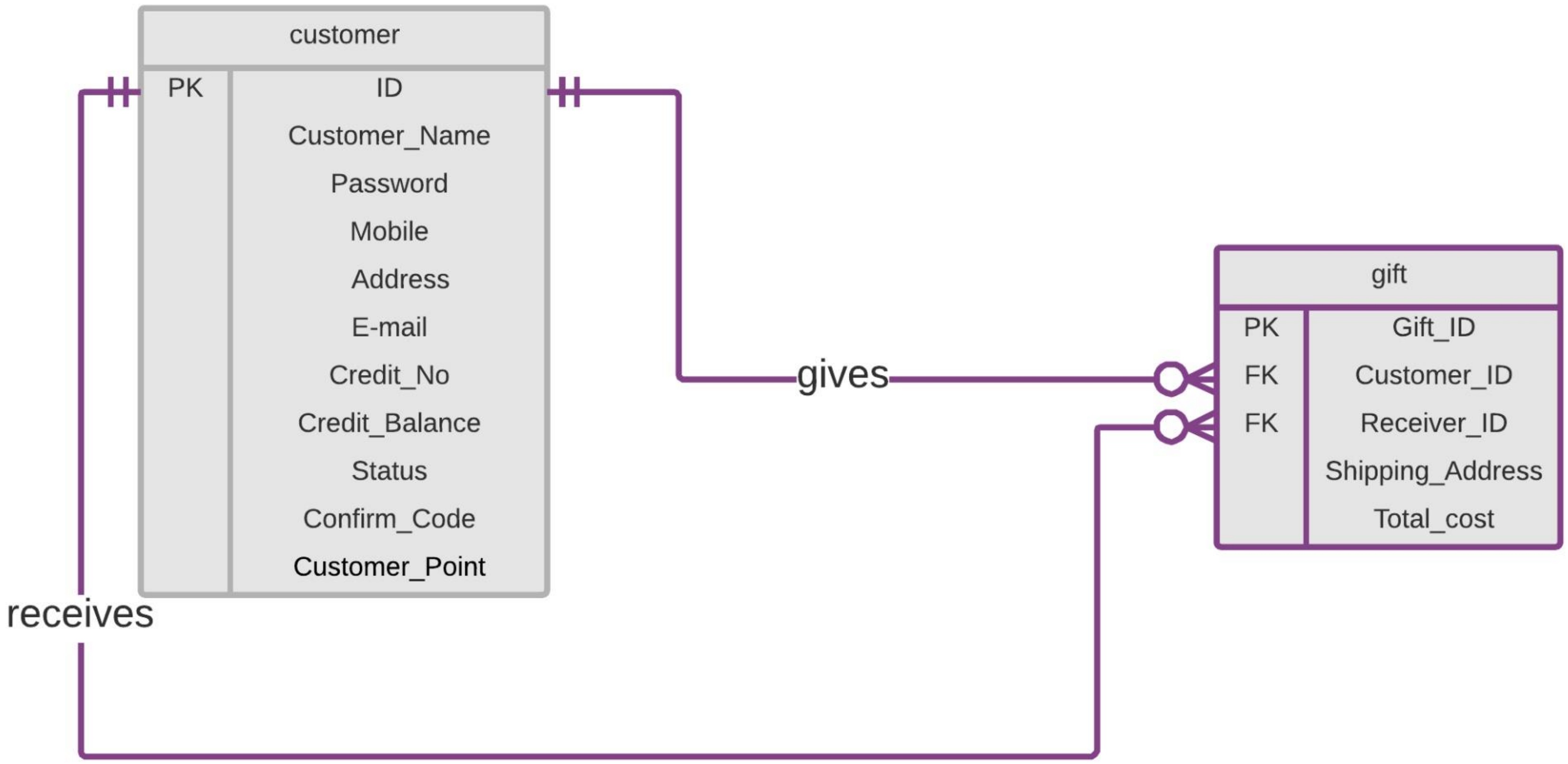
Relationships - Cardinality

- Cardinality represents the number of instances of an entity that exist in a relationship between two entities.
- ***One to one:*** One record of an entity is directly related to another record of an entity. A student has one address, and an address belongs to one student
- ***One to many:*** One record of an entity is related to one or more records of another entity. A professor teaches many courses, and a course is taught by one professor
- ***Many to many:*** Many records of one entity can be related to many records of another entity. A student enrolls in many courses, and a course has many students enrolled
- We often crow's foot notation or the Chen notation to represent the cardinality of a relationship in an ER diagram.

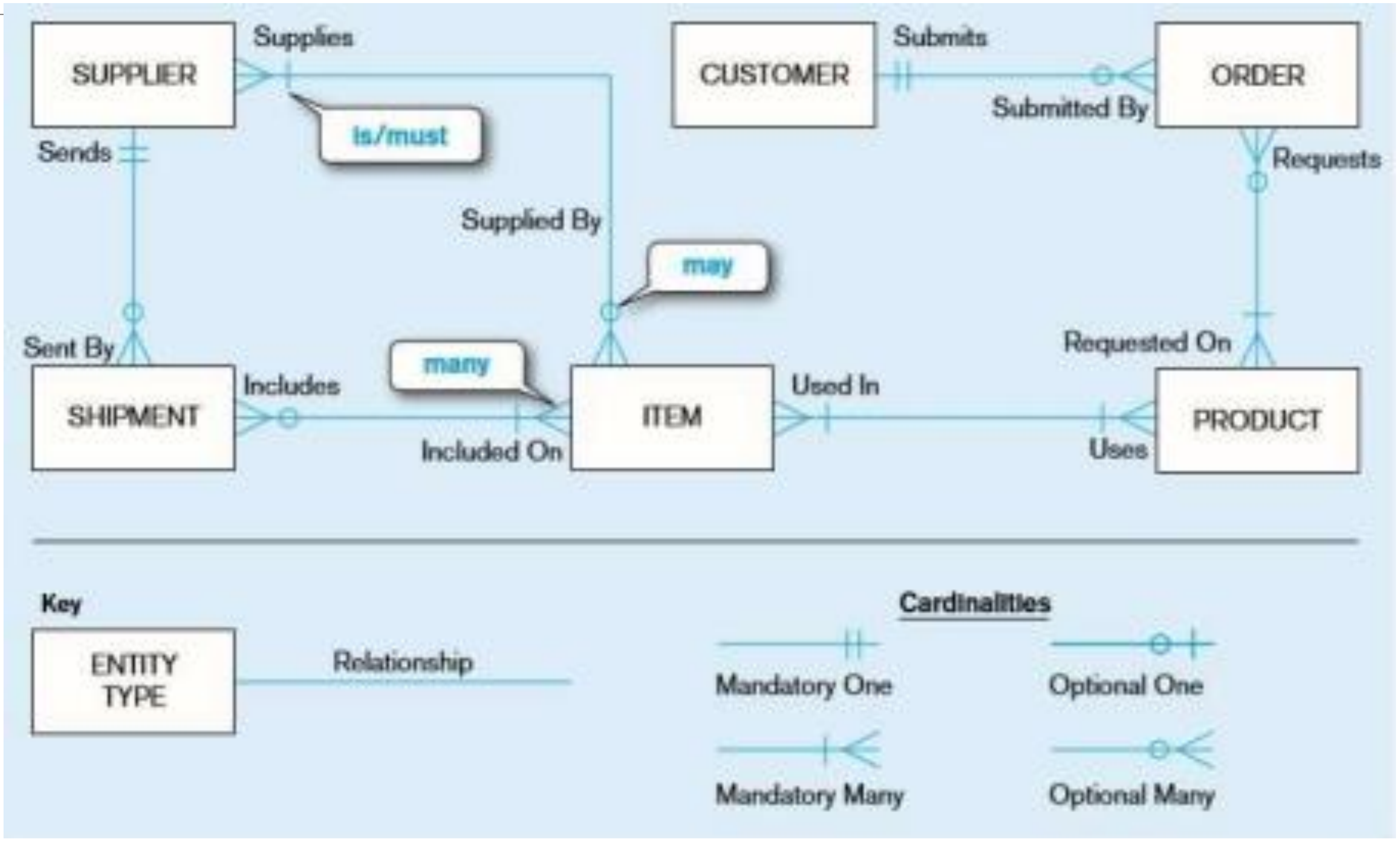
Relationship – Cardinality (Crows Foot)



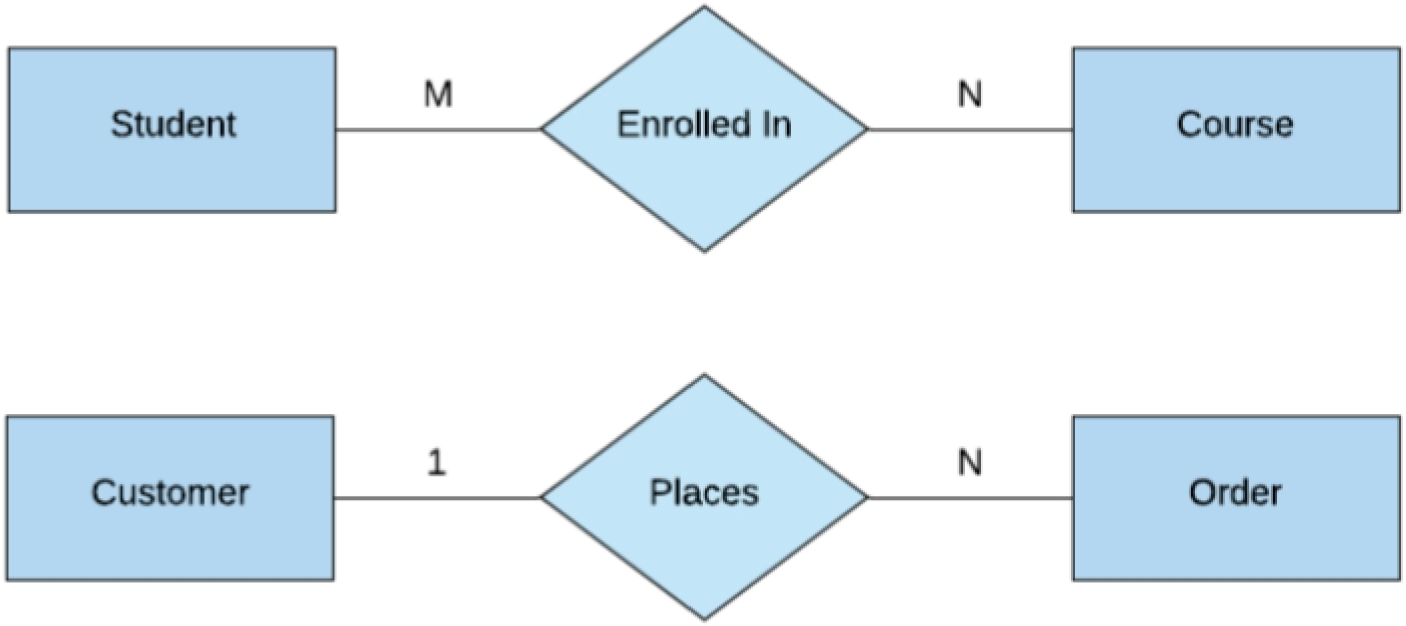
Relationship – Cardinality (Crows Foot)



Relationship – Cardinality (Crows Foot)

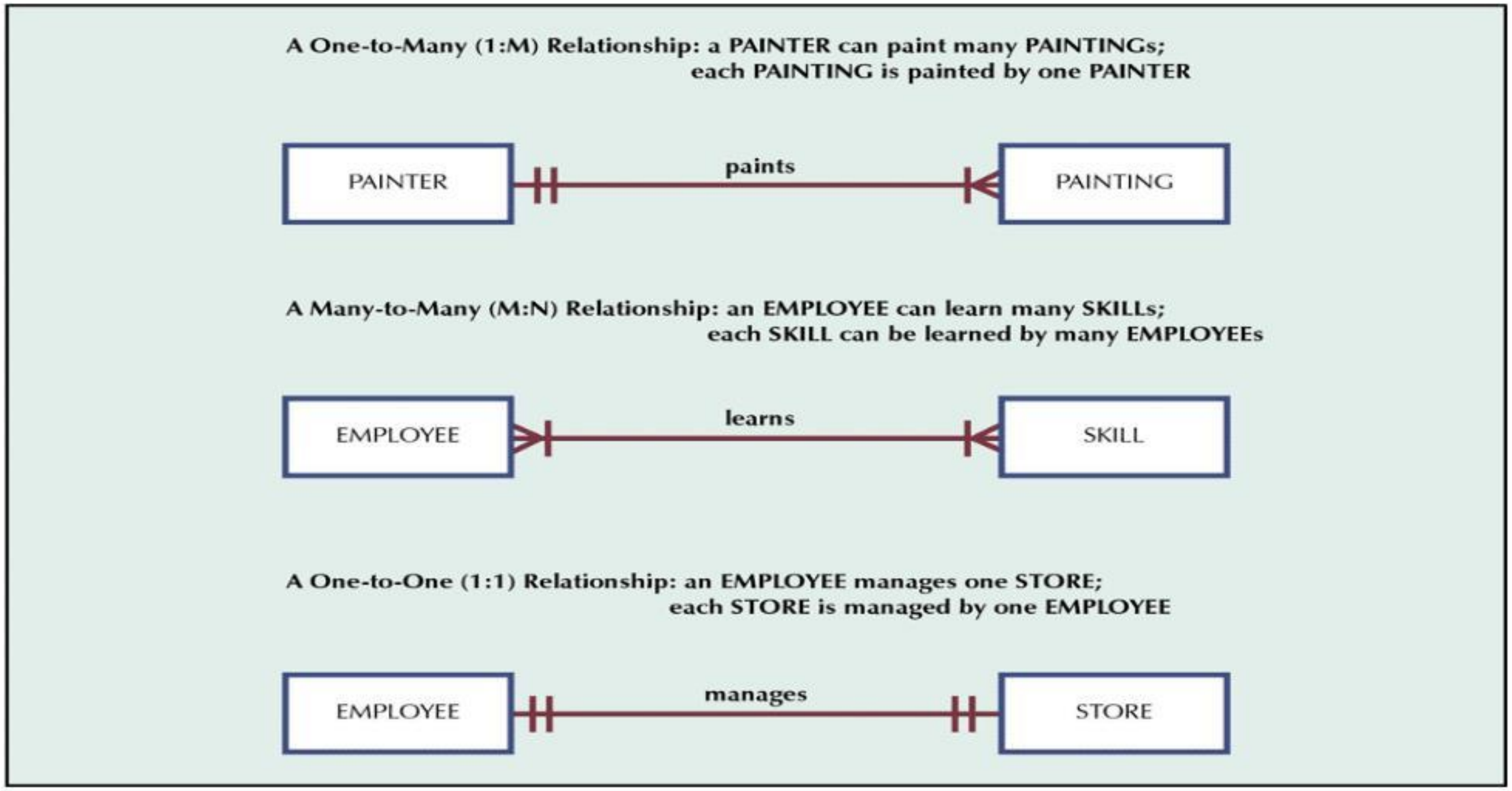


Relationship – Cardinality (Chen Notation)



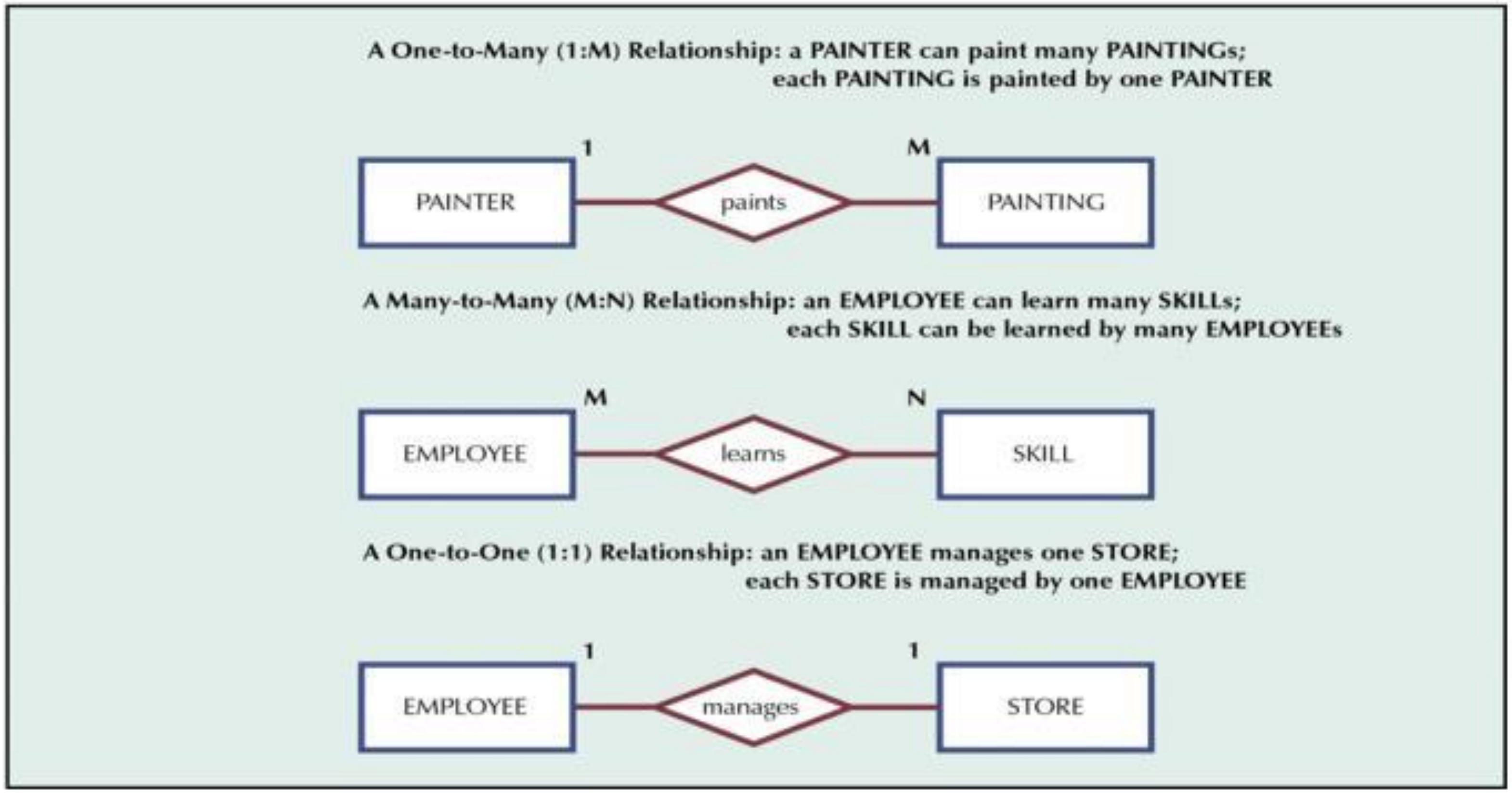
Relationships: The Basic Crow's Foot ERD

FIGURE 2.7 RELATIONSHIPS: THE BASIC CROW'S FOOT ERD


A One-to-Many (1:M) Relationship: a PAINTER can paint many PAINTINGs; each PAINTING is painted by one PAINTER
PAINTER
paints
PAINTING
A Many-to-Many (M:N) Relationship: an EMPLOYEE can learn many SKILLs; each SKILL can be learned by many EMPLOYEEs
EMPLOYEE
learns
SKILL
A One-to-One (1:1) Relationship: an EMPLOYEE manages one STORE; each STORE is managed by one EMPLOYEE
EMPLOYEE
manages
STORE

Relationships: The Basic Chen ERD

FIGURE 2.6 RELATIONSHIPS: THE BASIC CHEN ERD


A One-to-Many (1:M) Relationship: a PAINTER can paint many PAINTINGs; each PAINTING is painted by one PAINTER
1
M
PAINTER
paints
PAINTING
A Many-to-Many (M:N) Relationship: an EMPLOYEE can learn many SKILLs; each SKILL can be learned by many EMPLOYEEs
M
N
EMPLOYEE
learns
SKILL
A One-to-One (1:1) Relationship: an EMPLOYEE manages one STORE; each STORE is managed by one EMPLOYEE
1
1
EMPLOYEE
manages
STORE

Creating ERDs - Steps

- Identify Entities: Determine the main entities in the system being modeled. These are the key objects or concepts that need to be represented.
- Define Relationships: Establish the relationship between and their cardinality entities (e.g., one-to-one, one-to-many, many-to-many)
- Add Attributes: Define the characteristics or properties of each entity. These attributes provide additional details about the entities.
- Draw the Diagram: Use standard notation (such as crow's foot notation) to create the visual representation of the ERD. Draw entities as rectangles, relationships as lines connecting them, and attributes within the rectangles.
- You can use various tools to come up with an ERD.

Creating ERDs – Tools that can be Used

- Microsoft Visio
- Microsoft MySQL Workbench
- Oracle SQL Developer
- LucidChart
- Visual Paradigm
- Get Mind
- Gliffy
- Among others

Creating ERDs – Question

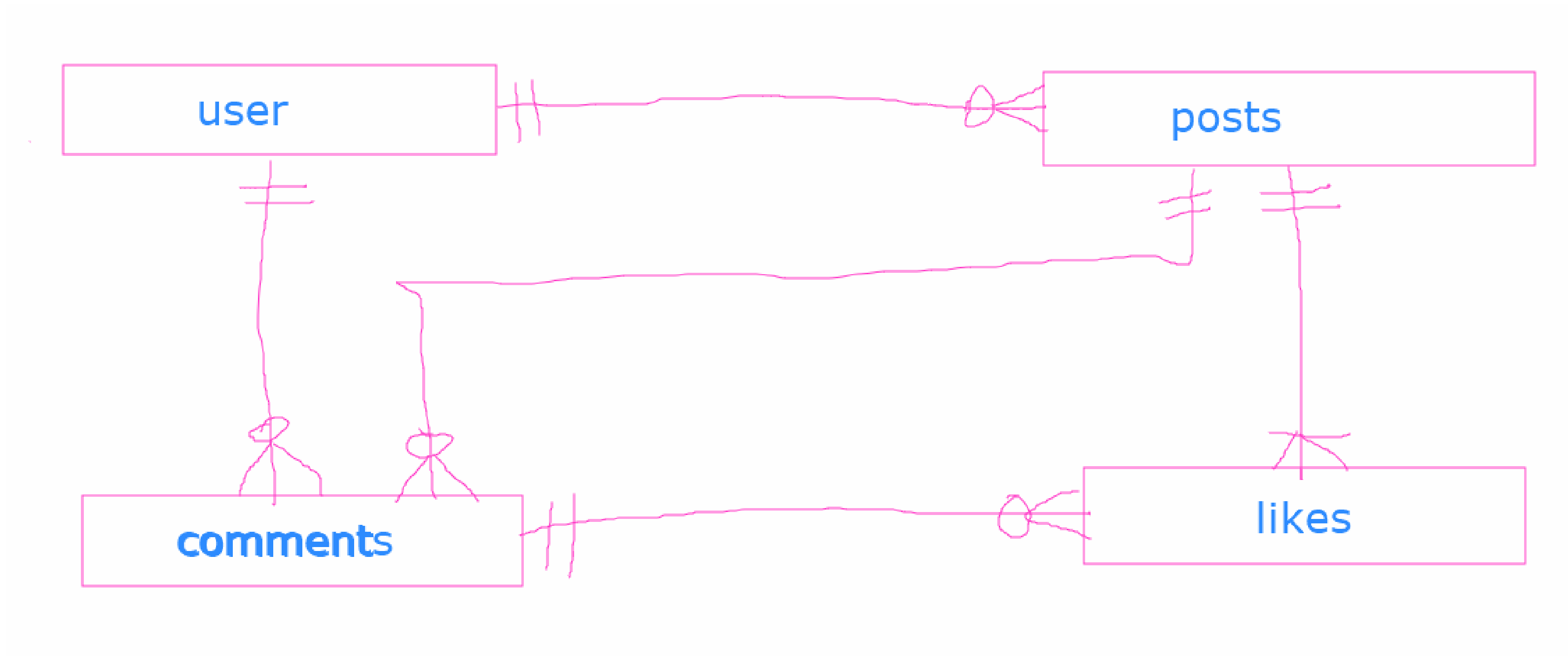
Imagine you're tasked with creating a database schema for a new social media platform. This platform allows users to interact with each other by creating posts, commenting on posts, and liking posts. Each user has a unique user ID, username, and email address. Posts are identified by a post ID and contain content and date information. Comments are associated with posts and have their own comment ID, content, and date. Likes are linked to posts and have a unique like ID along with the date.

Design the Entity-Relationship Diagram (ERD) for this social media platform, including all necessary entities, attributes, and relationships. Ensure that your ERD clearly represents how users interact with posts through comments and likes, capturing the essence of the platform's functionality.

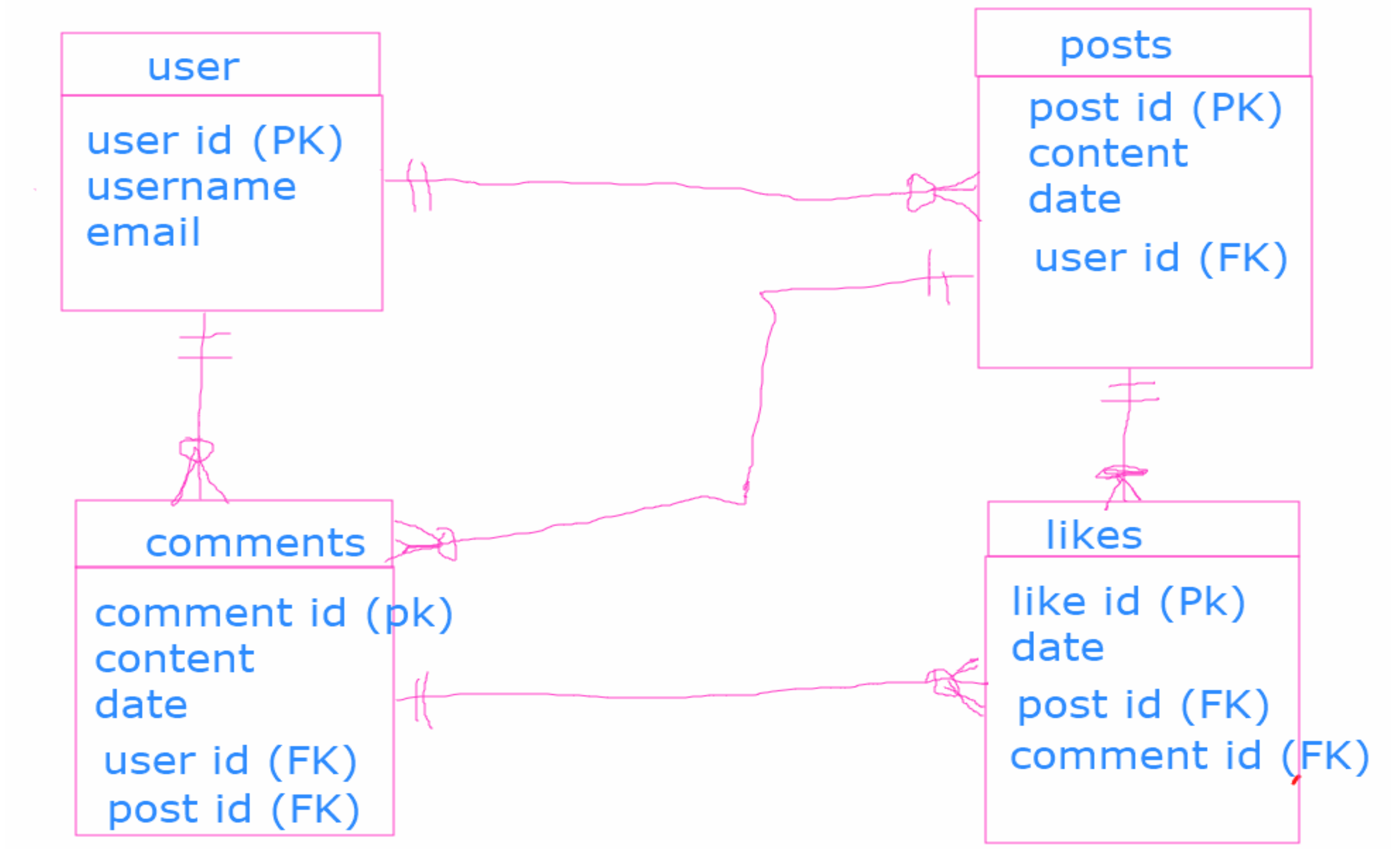
Creating ERDs – Question

- **Entities:** user, posts, comments, likes
- **Attributes:**
  - user: id, username, email
  - posts: post_id, date, content
  - comments: comment_id, date, content
  - likes: like_id, date)
- **Relationship:**
  - user creates posts - 1:M
  - post has comments - 1:M
  - post receives likes - 1:M
  - user comments on posts - 1:M
  - user likes posts - 1:M

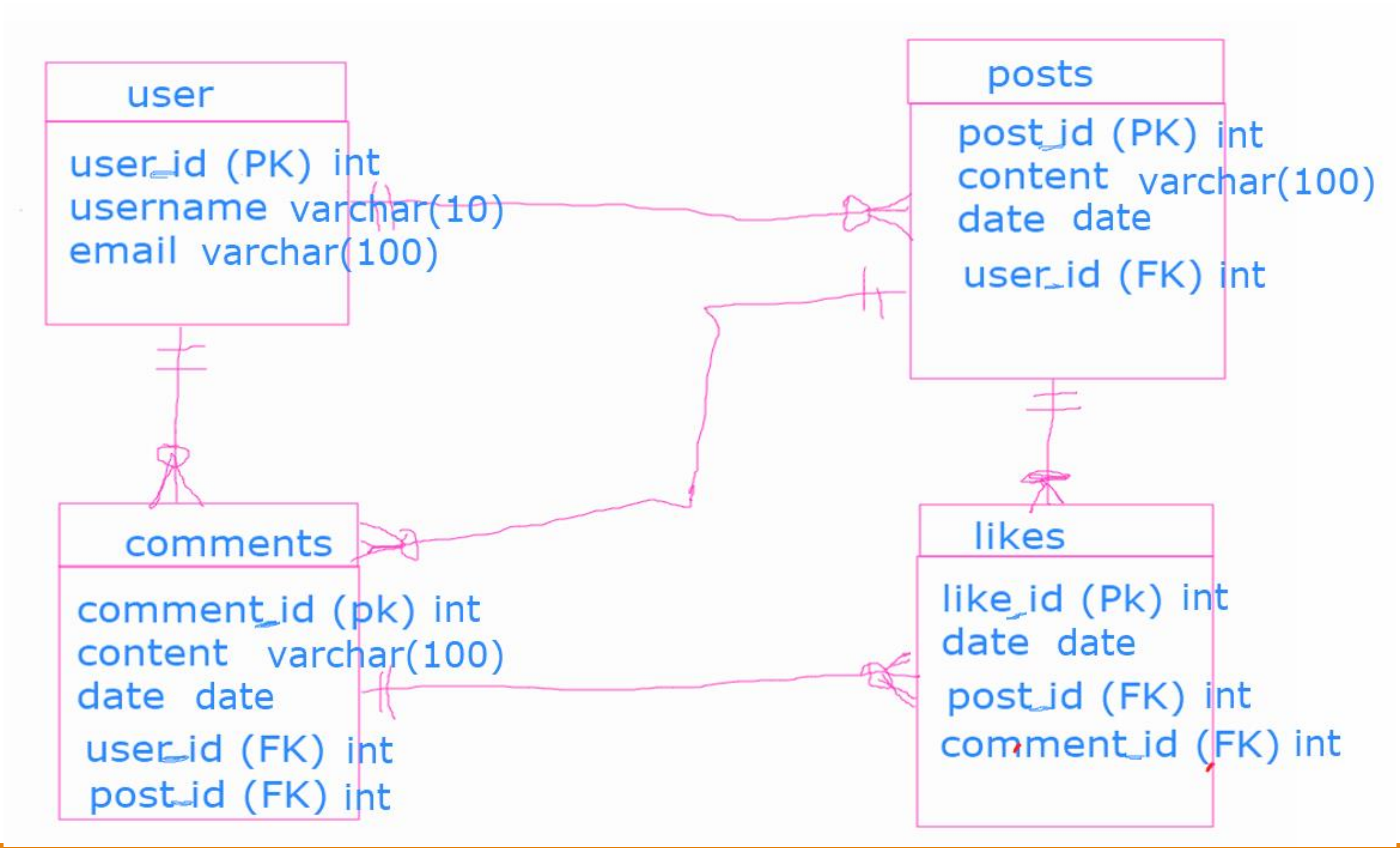
Creating ERDs – Question – Conceptual



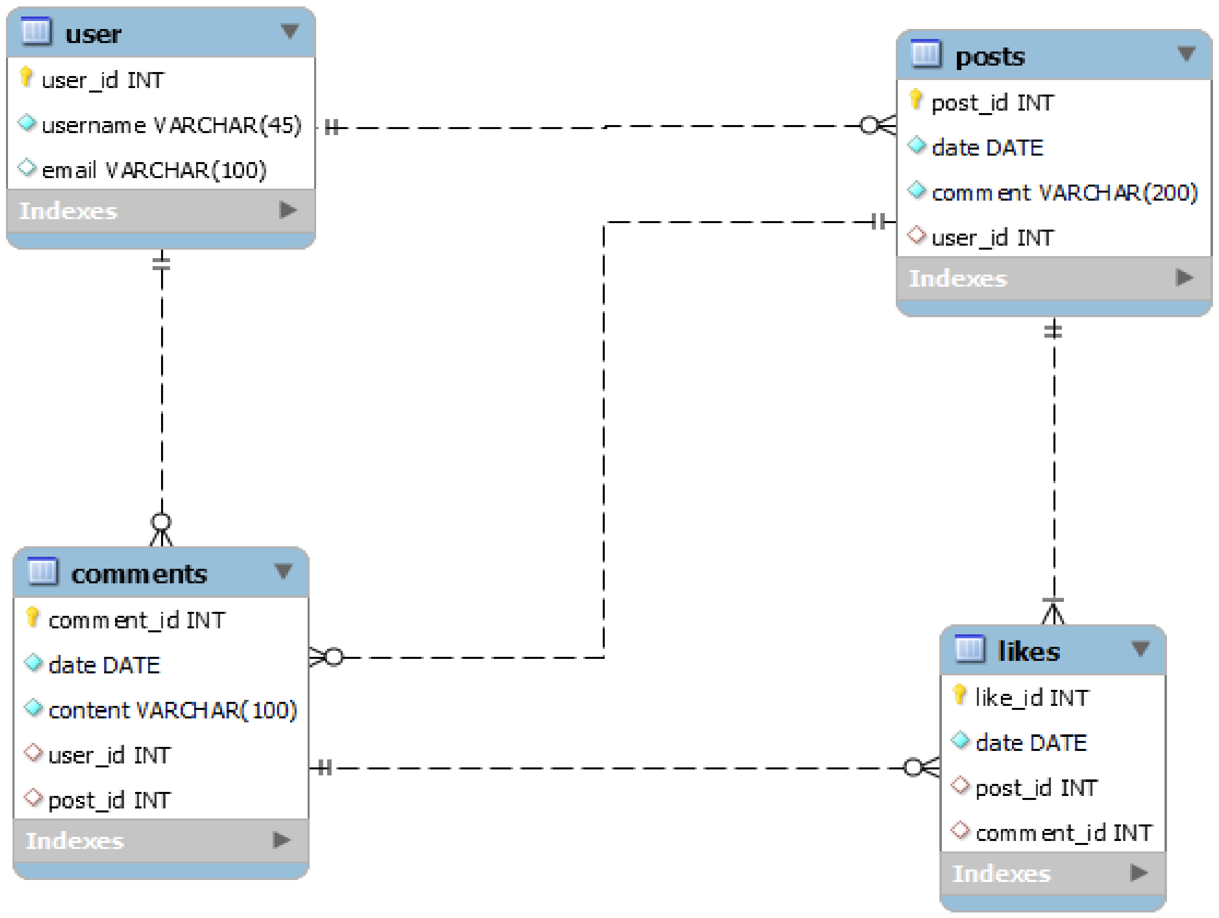
Creating ERDs – Question – Logical Model



Creating ERDs – Question – Physical Model



Creating ERDs – Question – Using a tool



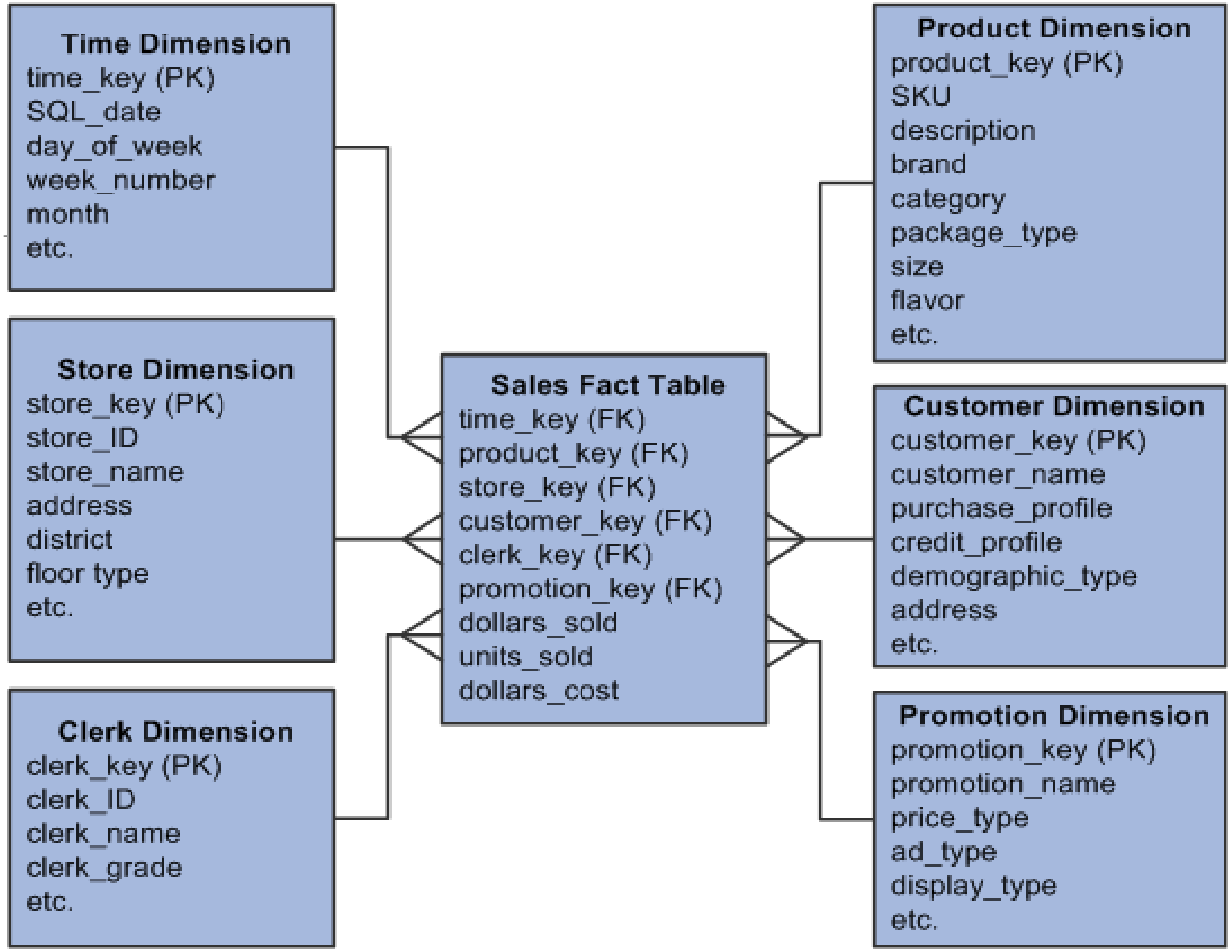
DIMENSION MODELLING

Dimension Modelling

- Dimension modeling is a data modeling technique used in data warehousing to structure and organize data for analysis and reporting purposes.
- It involves designing **fact tables** and **dimension tables** to provide a framework for storing and querying data efficiently.
- It focuses on simplicity, ease of use, and query performance, making it well-suited for decision support systems.
- Dimensional modelling employs simplified schema designs, primarily **star schema** and **snowflake schema.**
- It is a dimension model that facilitates data analysis, reporting and decision making.

Dimension Modelling

- Relational modeling organizes data into normalized tables with emphasis on reducing redundancy and dependency.
- It is suited for transactional or operational databases.
- Dimensional modeling organizes data into fact tables and dimension tables with emphasis on simplifying data relationships for analysis and reporting.
- Dimension modelling is suited for analytical databases, particularly data warehouses and data marts.


Time Dimension
time_key (PK)
SQL_date
day_of_week
week_number
month
etc.
Store Dimension
store_key (PK)
store_ID
store_name
address
district
floor type
etc.
Clerk Dimension
clerk_key (PK)
clerk_ID
clerk_name
clerk_grade
etc.
Sales Fact Table
time_key (FK)
product_key (FK)
store_key (FK)
customer_key (FK)
clerk_key (FK)
promotion_key (FK)
dollars_sold
units_sold
dollars_cost
Product Dimension
product_key (PK)
SKU
description
brand
category
package_type
size
flavor
etc.
Customer Dimension
customer_key (PK)
customer_name
purchase_profile
credit_profile
demographic_type
address
etc.
Promotion Dimension
promotion_key (PK)
promotion_name
price_type
ad_type
display_type
etc.

Why Dimension Modelling?

- Dimension modeling simplifies complex data relationships, making it easier for users to analyze and interpret data.
- By organizing data into dimensions and facts, users can quickly navigate and understand the data structure, leading to more effective analysis.
- Relational models may require complex joins and queries to retrieve data for reporting and analysis, resulting in inefficient query performance.
- Relational modeling is primarily designed for transactional databases and may not be well-suited for data warehousing environments.
- Dimensional models are optimized for query performance, with denormalized structures and pre-aggregated data.

Components of a dimension model

- **Fact tables** contain quantitative data (facts) related to business processes

  e.g. revenue, quantity sold, order amount.
- **Dimension tables** contain descriptive attributes that provide context to facts stored in fact tables e.g. product, customer, time.
- **Attributes** are the properties or characteristics of entities within dimension tables e.g. product name, customer ID, date.
- **Hierarchies** represent ordered levels of attributes within dimensions e.g. Date hierarchy (year > quarter > month), product hierarchy (category > subcategory > product).
- **Keys** are unique identifiers used to establish relationships between fact and dimension tables e.g. primary keys, foreign keys.

Dimension Model – Fact tables

- Fact tables contain quantitative data (facts) related to business processes e.g. revenue, quantity sold, order amount.
- A fact table is the primary table in a dimensional model where the numerical performance measurements of the business are stored.
- Fact tables serve as the central repository for measurable data points in dimension modeling.
- They store transactional or event-based data that can be aggregated and analyzed.
- Fact tables contain foreign keys that establish relationships with dimension tables.

Dimension Model – Fact tables

- Fact tables may contain different types of facts, including additive, semi-additive, and non-additive facts.
- Additive facts can be summed up across all dimensions, while semi-additive and non-additive facts have limitations on aggregation.
- Aggregations include sums, averages, counts, and other statistical measures calculated at different levels of granularity.
- A detail to be added to the fact table normally answers to the "how much/many" question.

Dimension Model – Fact tables

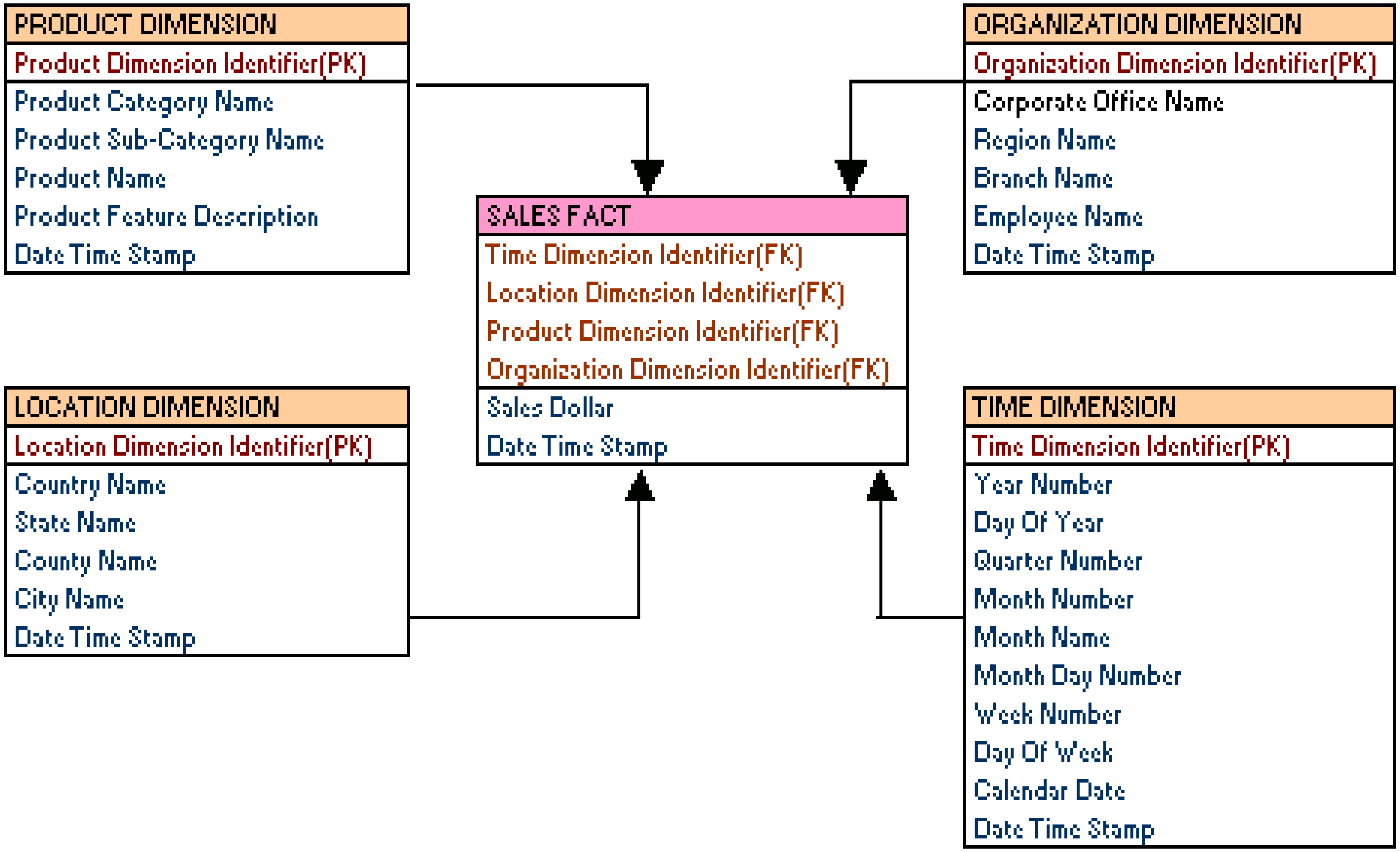
- Fact tables may contain different types of facts, including additive, semi-additive, and non-additive facts.
- Additive facts can be summed up across all dimensions, while semi-additive and non-additive facts have limitations on aggregation.
- Aggregations include sums, averages, counts, and other statistical measures calculated at different levels of granularity.
- A detail to be added to the fact table normally answers to the “how much/many” question.

Dimension Model – Dimension tables

- Dimension tables contain descriptive attributes that provide context to the facts stored in the fact table.
- For example, in a sales data warehouse, dimension tables might include product, time, and location dimensions.
- Descriptive attributes represent the characteristics or properties of entities within the dimension, such as product name, customer ID, or date.
- Dimension tables typically include unique keys or identifiers that uniquely identify each record within the dimension.
- Dimension tables often have a hierarchical structure with ordered levels of attributes.
- Dimension tables store categorical or qualitative data that categorizes and organizes the data in the fact table.

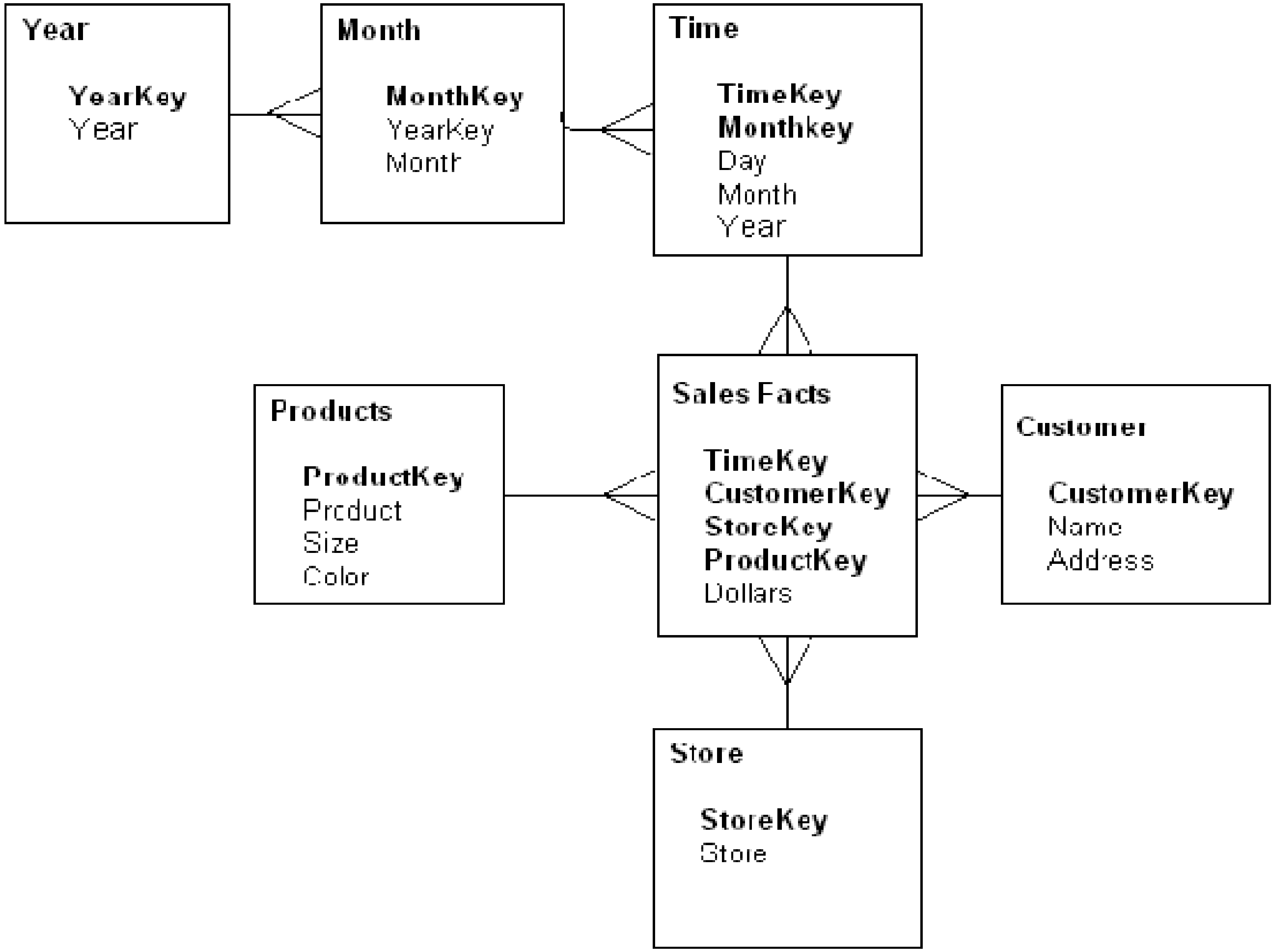
Types of Dimension Models – Star Schema

- In a star schema, data is organized into a central fact table surrounded by dimension tables.
- The fact table contains quantitative data (facts) related to business processes, while dimension tables contain descriptive attributes that provide context to the facts.
- Star schemas are denormalized, with each dimension table directly connected to the fact table.
- Star schemas are optimized for query performance and are well-suited for OLAP (Online Analytical Processing) environments.
- Look for advantages and disadvantages of this schema


PRODUCT DIMENSION
Product Dimension Identifier(PK)
Product Category Name
Product Sub-Category Name
Product Name
Product Feature Description
Date Time Stamp
ORGANIZATION DIMENSION
Organization Dimension Identifier(PK)
Corporate Office Name
Region Name
Branch Name
Employee Name
Date Time Stamp
SALES FACT
Time Dimension Identifier(FK)
Location Dimension Identifier(FK)
Product Dimension Identifier(FK)
Organization Dimension Identifier(FK)
Sales Dollar
Date Time Stamp
LOCATION DIMENSION
Location Dimension Identifier(PK)
Country Name
State Name
County Name
City Name
Date Time Stamp
TIME DIMENSION
Time Dimension Identifier(PK)
Year Number
Day Of Year
Quarter Number
Month Number
Month Name
Month Day Number
Week Number
Day Of Week
Calendar Date
Date Time Stamp

Types of Dimension Models – Snowflake Schema

- A snowflake schema is an extension of the star schema, where dimension tables are normalized into multiple related tables.
- Dimension tables in a snowflake schema are organized into a hierarchical structure, with each level of attributes stored in separate tables.
- Snowflake schemas reduce data redundancy and improve data integrity by normalizing dimension tables.
- While snowflake schemas offer benefits in terms of data normalization and storage efficiency, they may result in more complex queries and potentially slower query performance compared to star schemas.
- Look for advantages and disadvantages of this schema

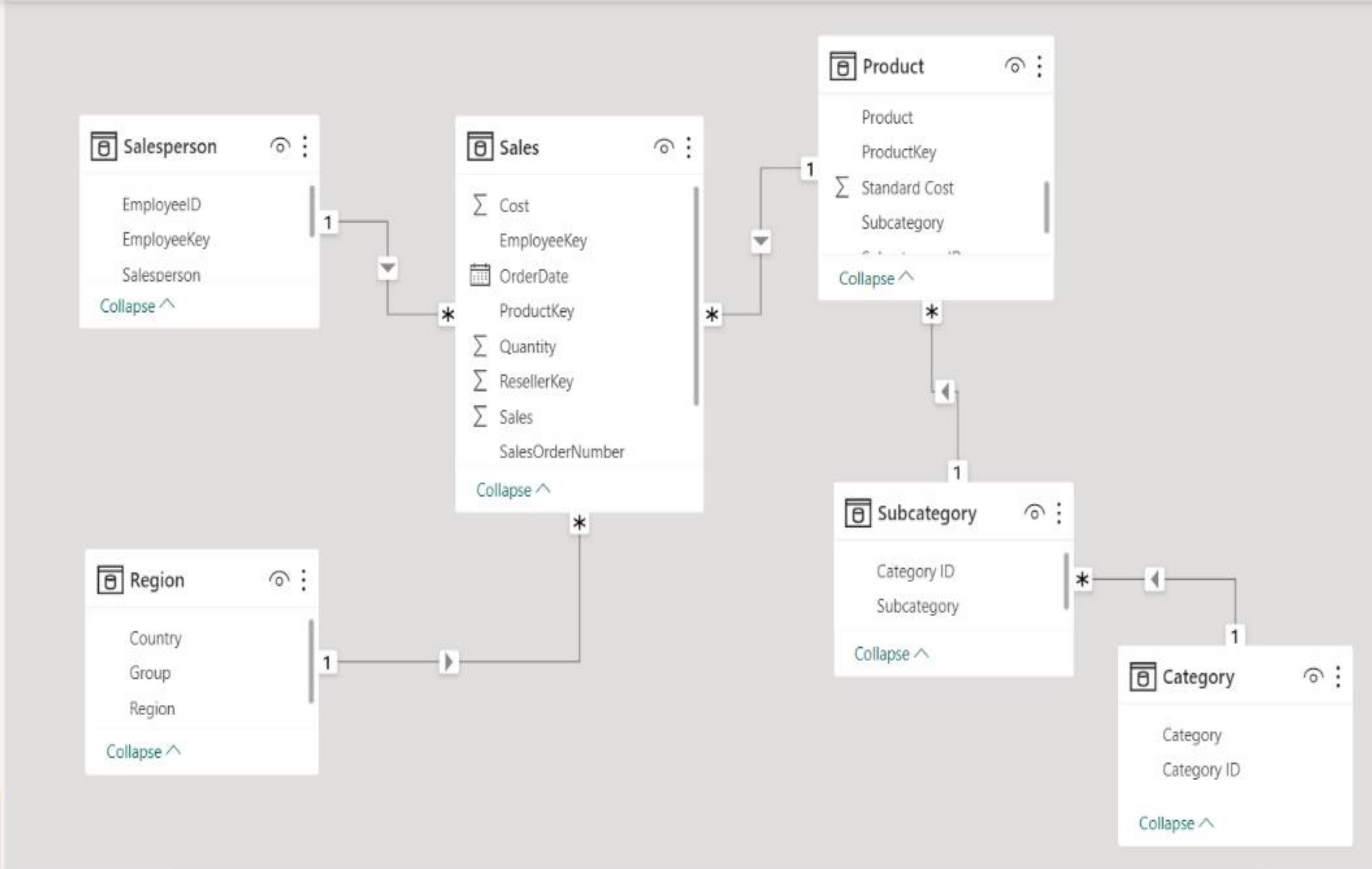


Year
YearKey
Year
Month
MonthKey
YearKey
Month
Time
TimeKey
Monthkey
Day
Month
Year
Products
ProductKey
Product
Size
Color
Sales Facts
TimeKey
CustomerKey
StoreKey
ProductKey
Dollars
Customer
CustomerKey
Name
Address
Store
StoreKey
Store

Types of Dimension Models– Flat Schema

- A flat schema is a schema where all attributes and fields related to the entity are stored in a single table.
- All data is stored in a single table, with each row representing a single record or entity, and each column representing an attribute or field.
- flat schemas do not define relationships between tables. All data is stored together without any inherent relationships or dependencies.
- Flat schemas are straightforward and easy to understand, making them suitable for simple data storage and retrieval tasks.
- Flat schemas are less commonly used in modern industry settings compared to relational or dimensional schemas.

Which type of model is this?



Types of Dimension Models–Conclusion

- Understanding schemas and the available types is crucial for effective database design and management.
- The Flat schema is simple and easy to work with but may not be suitable for complex data relationships or large datasets.
- The Star schema is a popular choice for dimensional modeling, offering reduced data redundancy and intuitive design. Although it may be less flexible when handling complex relationships between dimensions.
- Lastly, the Snowflake schema provides greater flexibility and improved data integrity through normalizing dimension tables. However, its complexity can make it more challenging to understand and query.

Dimension Modelling Process

- Define business process/scenario.
- Identify dimensions (descriptive attributes) and facts (quantifiable measures) that will be relevant for analysis.
- Design fact tables to store the quantitative data (facts) related to the business scenario.
- Design dimension tables to store descriptive attributes related to the dimensions identified in the business scenario.
- Define relationships between fact and dimension tables using foreign keys.
- Populate the fact and dimension tables with relevant data extracted from operational systems or other data sources.
- Create indexes on foreign keys and frequently queried columns to optimize query performance.

Assignment

You are working as a data analyst for a retail company that sells various products through both physical stores and online channels. The company is interested in improving its sales analysis and decision-making process by implementing a dimensional data model.

Your task is to design a dimensional data model that captures the sales transactions and related information for the retail company. Consider the following requirements:

Assignment

- The company sells products across different categories such as electronics, clothing, accessories, etc.
- Each product has a unique identifier, a name, a category, and a price.
- Sales transactions occur both in physical stores and online. Each transaction is recorded with a unique identifier, a timestamp, the customer's information (e.g., ID, name), and the products purchased along with their quantities.
- Customers can be regular or guest. Regular customers have accounts with the company, while guest customers do not.
- Each transaction is associated with a store or an online platform.
- The company wants to analyze sales performance by product category, store/online platform, customer type, and time period.
- Design a dimensional data model to meet the company's requirements. Include appropriate dimension tables, fact tables, and their relationships.

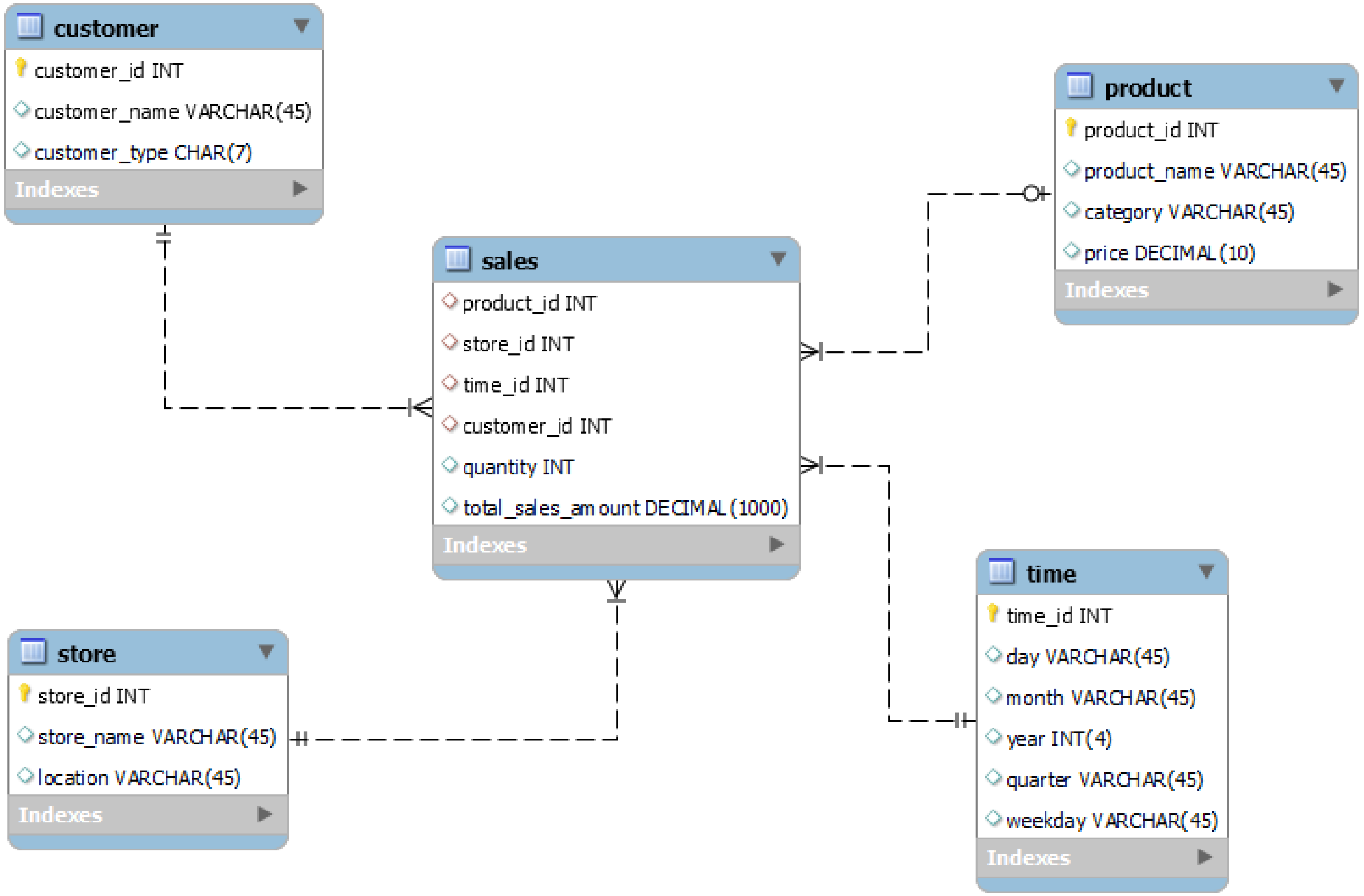
Assignment

Dimension Tables:

- **Product Dimension:** Product_ID (Primary Key), Product_Name, Category, Price
- **Customer Dimension:** Customer_ID (Primary Key), Customer_Name, Customer_Type (Regular/Guest)
- **Store Dimension:** Store_ID (Primary Key), Store_Name, Location
- **Time Dimension:** Date_ID (Primary Key), Date, Day, Month, Year, Quarter, Weekday

Fact Table:

- **Sales Fact Table:** Date_ID (Foreign Key referencing Time Dimension), Product_ID (Foreign Key referencing Product Dimension), Customer_ID (Foreign Key referencing Customer Dimension), Store_ID (Foreign Key referencing Store Dimension), Quantity, Total_Sales_Amount


customer
customer_id INT
customer_name VARCHAR(45)
customer_type CHAR(7)
Indexes
product
product_id INT
product_name VARCHAR(45)
category VARCHAR(45)
price DECIMAL(10)
Indexes
sales
product_id INT
store_id INT
time_id INT
customer_id INT
quantity INT
total_sales_amount DECIMAL(1000)
Indexes
store
store_id INT
store_name VARCHAR(45)
location VARCHAR(45)
Indexes
time
time_id INT
day VARCHAR(45)
month VARCHAR(45)
year INT(4)
quarter VARCHAR(45)
weekday VARCHAR(45)
Indexes

Thank you!

This image was generated by Copilot AI to depict Data Modelling and ERDs