Lab Session 01

1. ER Modelling & Mapping-1

Aim:

The aim of this experiment is to understand and apply the concepts of Entity Relationship (ER) Modelling to design a database schema that accurately represents the relationships between entities in a given domain.

Description:

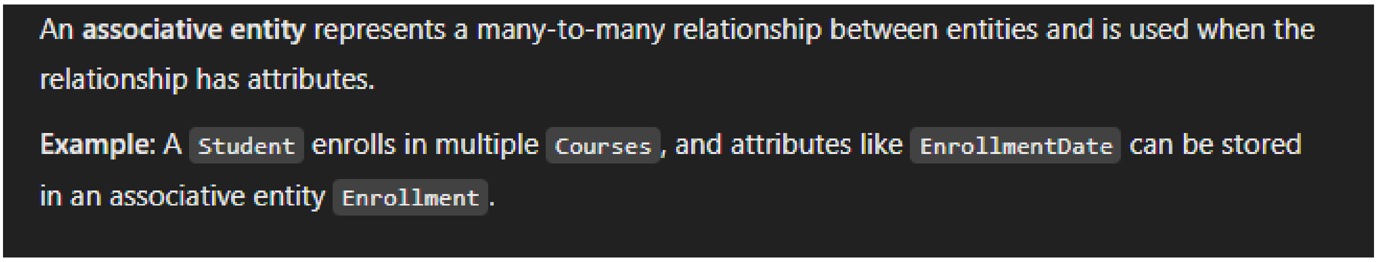
The lab experiment on Entity Relationship (ER) Modelling involves studying and applying the fundamental principles of ER modelling to design a database schema. Students will learn how to identify and define entities, relationships, and attributes within a given domain. They will practice creating an ER diagram that accurately represents the relationships between entities and their attributes. The experiment will also cover the use of cardinality and participation constraints to establish the nature and degree of relationships. By the end of the lab, students will gain hands-on experience in translating realworld scenarios into an ER model, providing a solid foundation for database design and development. Pre-Requisites: PostgreSQL, TerraER Tool, Windows/ Ubuntu/CentOS/Debian, DBMS Concepts.

Pre-Requisites:

PostgreSQL, TerraER Tool, Windows/ Ubuntu/CentOS/Debian, DBMS Concepts.

Pre Lab-Task:

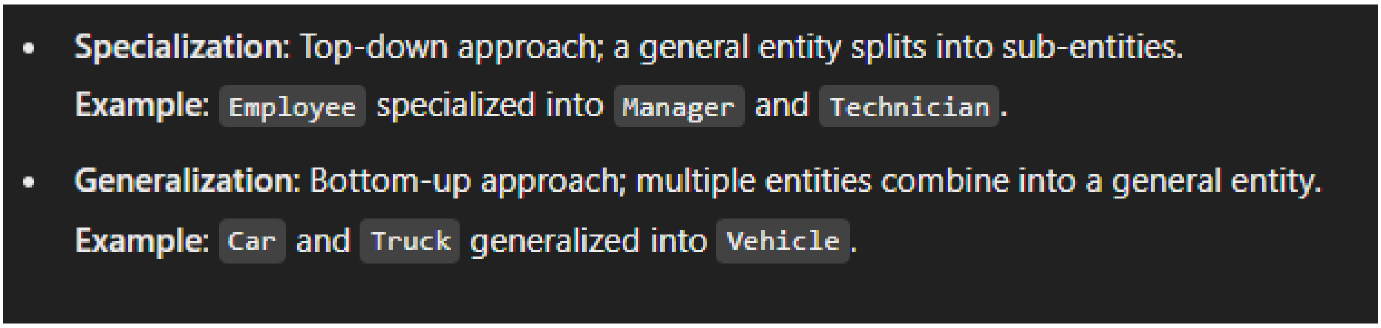
1) What is an associative entity in ER modelling, and when is it used?



An **associative entity** represents a many-to-many relationship between entities and is used when the relationship has attributes.

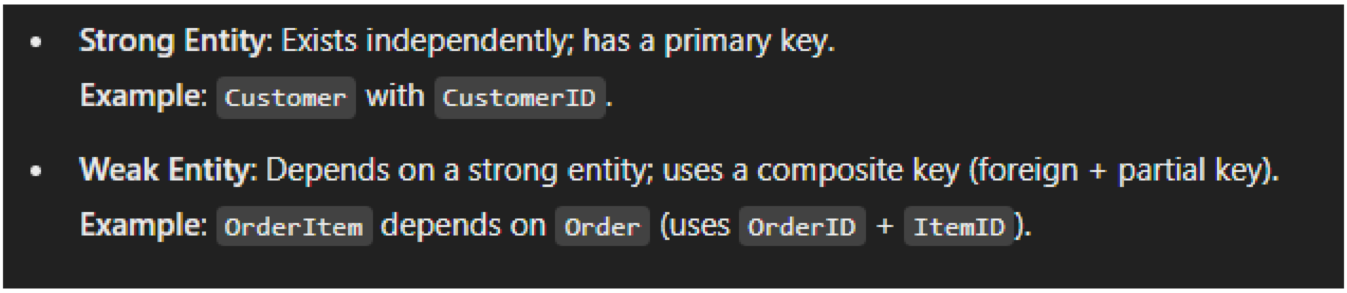
Example: A `Student` enrolls in multiple `Courses`, and attributes like `EnrollmentDate` can be stored in an associative entity `Enrollment`.

2) Discuss the concept of specialization and generalization in ER modelling.



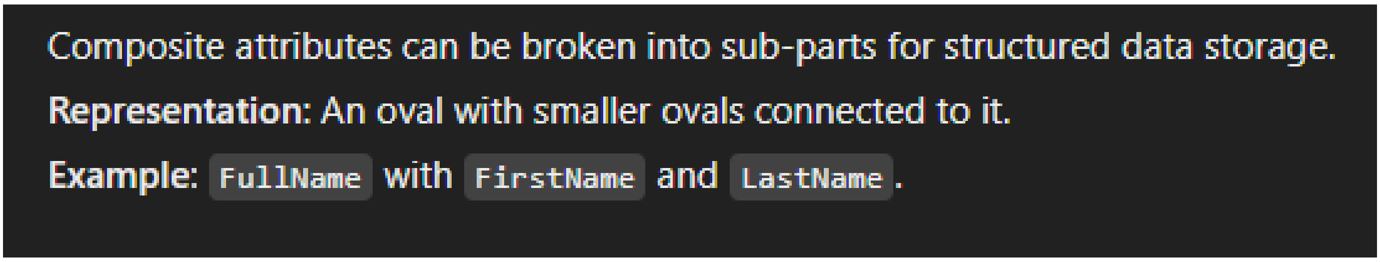
- **Specialization**: Top-down approach; a general entity splits into sub-entities.
  Example: `Employee` specialized into `Manager` and `Technician`.
- **Generalization**: Bottom-up approach; multiple entities combine into a general entity.
  Example: `Car` and `Truck` generalized into `Vehicle`.

3) What is the difference between a weak entity and a strong entity in ER modelling? Provide examples of each.



- **Strong Entity:** Exists independently; has a primary key.
  Example: `Customer` with `CustomerID`.
- **Weak Entity:** Depends on a strong entity; uses a composite key (foreign + partial key).
  Example: `OrderItem` depends on `Order` (uses `OrderID` + `ItemID`).

4) What is the purpose of a composite attribute, and how is it represented in an ER diagram?

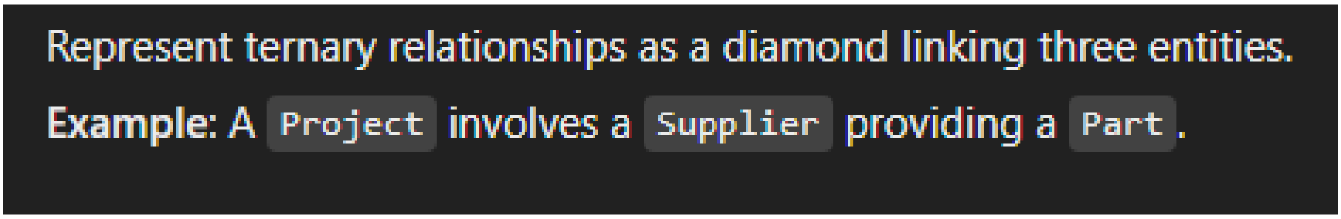


Composite attributes can be broken into sub-parts for structured data storage.

Representation: An oval with smaller ovals connected to it.

Example: `FullName` with `FirstName` and `LastName`.

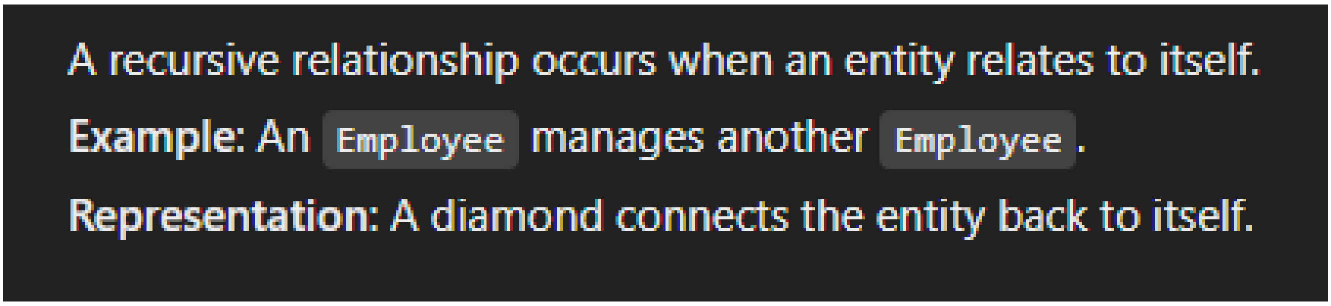
5) How do you handle complex relationships, such as ternary relationships, in an ER diagram?



Represent ternary relationships as a diamond linking three entities.

Example: A `Project` involves a `Supplier` providing a `Part`.

6) Explain the concept of recursive relationships in ER modelling and provide an example.



A recursive relationship occurs when an entity relates to itself.

Example: An `Employee` manages another `Employee`.

Representation: A diamond connects the entity back to itself.

In Lab Task:

Airline Reservation System

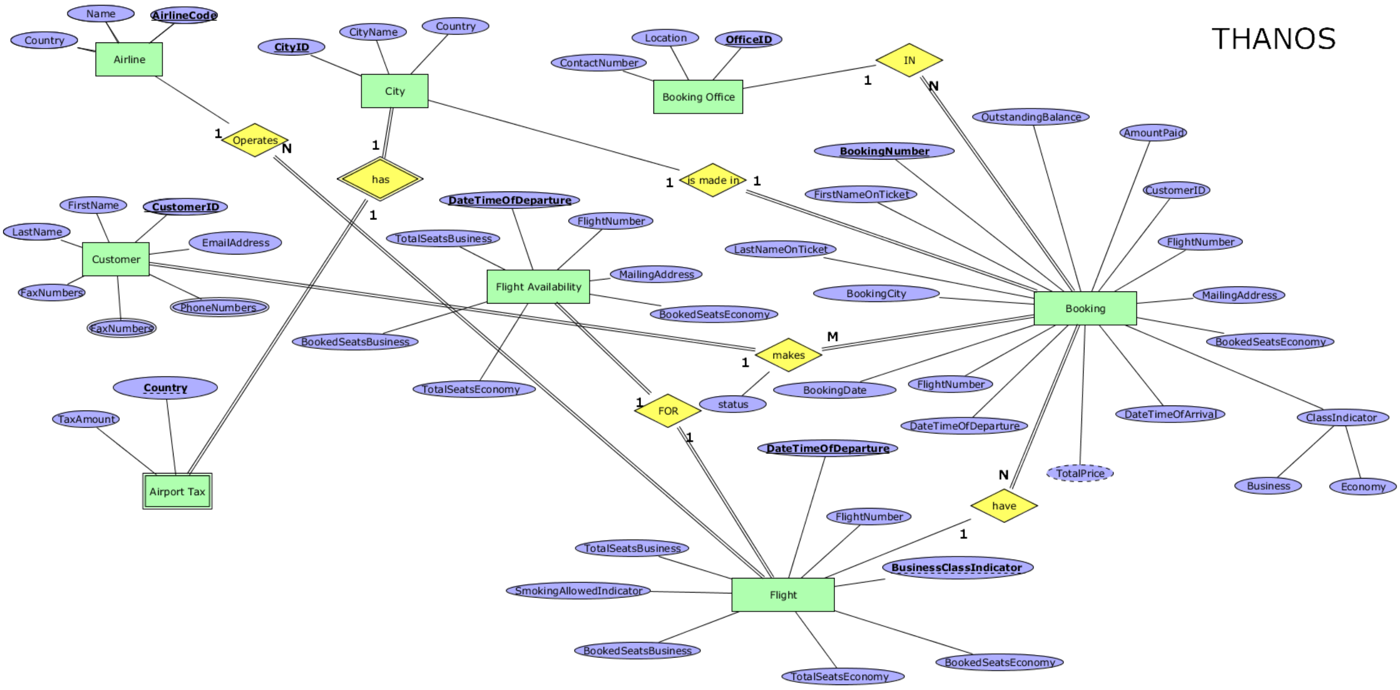
Problem Description: There are 6 different airlines in 6 different countries: Canada – Air Can, USA - USAir, UK – Brit Air, France – Air France, Germany – Luft Air, Italy – Ital Air. Their flights involve the following 12 cities: Toronto and Montreal in Canada, New York and Chicago in US, London and Edinburgh in UK, Paris and Nice in France, Bonn and Berlin in Germany, Rome and Naples in Italy. In each of the 12 cities, there is a (single) booking office. You are going to design a central air- reservation database to be used by all booking offices.

The flight has a unique flight number, airline code, business class indicator, smoking allowed indicator. Flight availability has flight number, date + time of departure, number of total seats available in business class, number of booked seats in business class, number of total seats available in economy class, and number of booked seats in economy class.

The customers may come from any country, not just the 6 above, and from any province/state, and from any city. The customer has first & last name, mailing address, zero or more phone numbers, zero or more fax numbers, and zero or more email addresses. Mailing address has street, city, province or state, postal code and country. Phone/fax number has country code, area code and local number. Email address has only one string, and no structure is assumed. A customer can book one or more flights. Two or more customers may have same mailing address and/or same phone number(s) and/or same fax number(s). But the email address is unique for each customer. First and last names do not have to be unique.

Booking has an unique booking number, booking city, booking date, flight number, date + time of departure (in local time, and time is always in hours and minutes), date + time of arrival (in local time), class indicator, total price (airport tax in origin + airport tax in destination + flight price – in local currency. The flight price for business class is 1.5 times of the listed flight price), status indicator (three types: booked. Canceled – the customer canceled the booking, scratched – the customer had not paid in full 30 days prior to the departure), customer who is responsible for payment, amount-paid-so far (in local currency), outstanding balance (in local currency), the first & last names to be printed on the ticket. The airport taxes must be stored in local currencies (i.e. Canadian 19 dollars, US dollars, British Pounds, French francs, German marks, and Italian Liras). Since the exchange rates change daily, they also must be stored for calculations of all prices involved. Though France, Germany, and Italy have had a common currency for a while, we used the names of their original currencies to involve in this exercise currency exchange rates and their changes.

1) Design an ER diagram that represents the entities, attributes, relationships, and cardinalities for the Airline Reservation System. Consider any necessary assumptions and justify them if required



2) Write the Relational schema that represents the entities, attributes, relationships, and cardinalities for the Airline Reservation System. Consider any necessary assumptions and justify them if required.

Entities and Attributes

1. **Airline**
   - **AirlineCode (Primary Key)**
   - **Name**
   - **Country**
2. **City**
   - **CityID (Primary Key)**
   - **CityName**
   - **Country**

3. **Customer**
    - **CustomerID (Primary Key)**
    - **FirstName**
    - **LastName**
    - **EmailAddress**
    - **PhoneNumbers**
    - **FaxNumbers**
4. **BookingOffice**
    - **OfficeID (Primary Key)**
    - **Location**
    - **ContactNumber**
5. **FlightAvailability**
    - **FlightNumber**
    - **DateTimeOfDeparture (Primary Key)**
    - **TotalSeatsBusiness**
    - **TotalSeatsEconomy**
    - **BookedSeatsBusiness**
    - **BookedSeatsEconomy**
6. **Flight**
    - **FlightNumber (Primary Key)**
    - **DateTimeOfDeparture**
    - **DateTimeOfArrival**
    - **TotalSeatsBusiness**
    - **TotalSeatsEconomy**
    - **SmokingAllowedIndicator**

7. **Booking**
    - **BookingNumber (Primary Key)**
    - **BookingDate**
    - **FlightNumber**
    - **DateTimeOfDeparture**
    - **CustomerID**
    - **ClassIndicator (Business/Economy)**
    - **BookingCity**
    - **FirstNameOnTicket**
    - **LastNameOnTicket**
    - **TotalPrice**
    - **OutstandingBalance**
    - **AmountPaid**
    - **MailingAddress**
8. **AirportTax**
    - **Country (Primary Key)**
    - **TaxAmount**

Relationships

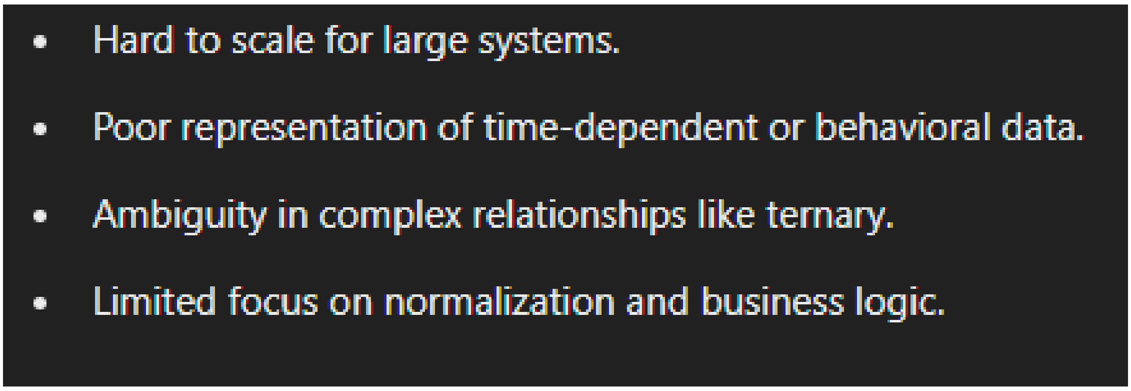
1. **Operates (Airline → City)**
    - **Cardinality: 1:N**
    - **Foreign Key: AirlineCode in City table links to Airline.**
2. **Has (City → BookingOffice)**
    - **Cardinality: 1:N**
    - **Foreign Key: CityID in BookingOffice.**
3. **Is_Made_In (BookingOffice → Booking)**
    - **Cardinality: 1:N**
    - **Foreign Key: OfficeID in Booking.**
4. **Makes (Customer → Booking)**
    - **Cardinality: 1:M**
    - **Foreign Key: CustomerID in Booking.**

5. **For (FlightAvailability → Flight)**
    - **Cardinality: 1:1**
    - **Foreign Key: FlightNumber and DateTimeOfDeparture in FlightAvailability.**

6. **Have (Flight → Booking)**
    - **Cardinality: 1:N**
    - **Foreign Key: FlightNumber and DateTimeOfDeparture in Booking.**

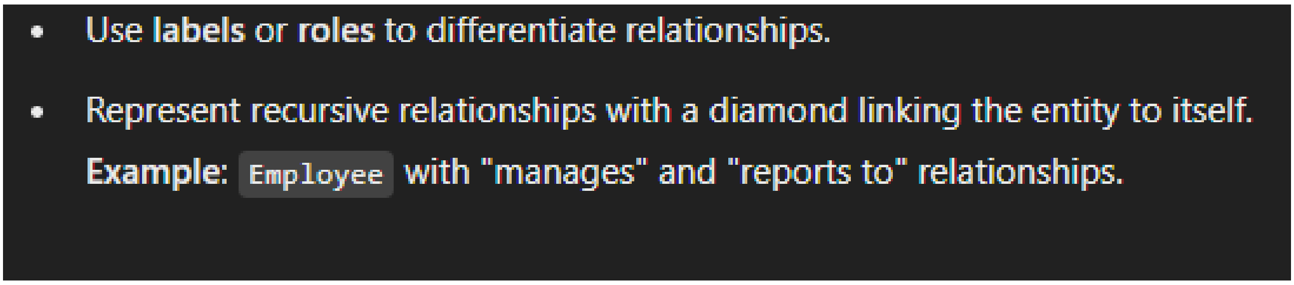
Viva-Voce Questions (In-Lab):

1) Can you explain the limitations or drawbacks of using an ER diagram for complex database systems?



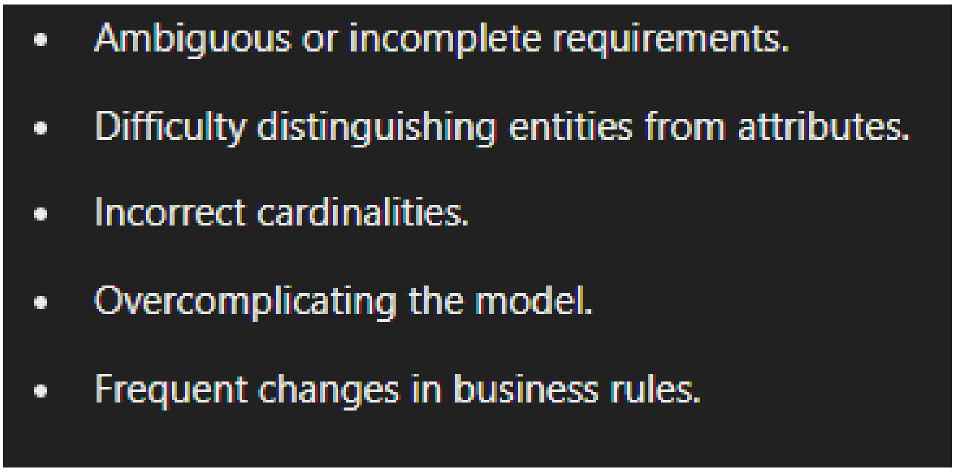
- Hard to scale for large systems.
- Poor representation of time-dependent or behavioral data.
- Ambiguity in complex relationships like ternary.
- Limited focus on normalization and business logic.

2) How would you handle a scenario where an entity has multiple relationships with the same entity type in an ER diagram?



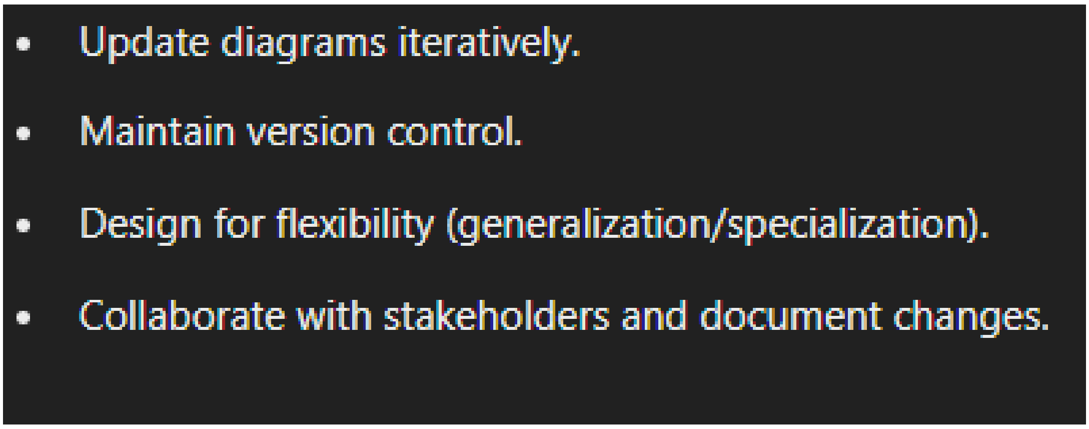
- Use **labels** or **roles** to differentiate relationships.
- Represent recursive relationships with a diamond linking the entity to itself.
  Example: `Employee` with "manages" and "reports to" relationships.

3) What are some common challenges or pitfalls that can arise when identifying entities and relationships in a real-world domain for ER modelling?



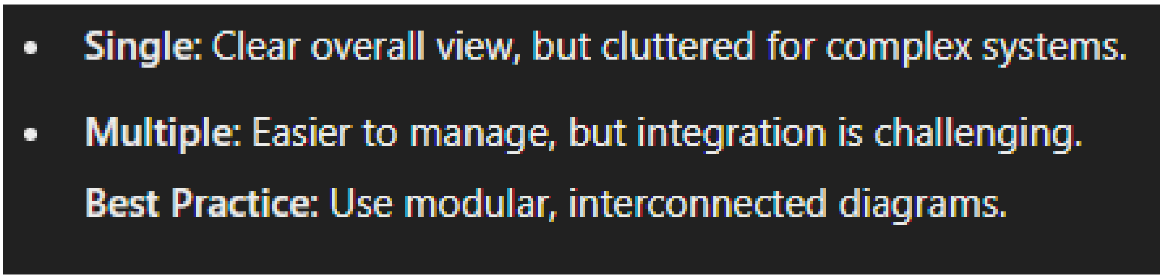
- Ambiguous or incomplete requirements.
- Difficulty distinguishing entities from attributes.
- Incorrect cardinalities.
- Overcomplicating the model.
- Frequent changes in business rules.

4) How do you handle evolving or changing requirements in ER modelling, especially when the relationships between entities need to be modified or updated?



- Update diagrams iteratively.
- Maintain version control.
- Design for flexibility (generalization/specialization).
- Collaborate with stakeholders and document changes.

5) Can you discuss the trade-offs between using a single ER diagram versus multiple interconnected ER diagrams for a complex database system?



- **Single:** Clear overall view, but cluttered for complex systems.
- **Multiple:** Easier to manage, but integration is challenging.

Best Practice: Use modular, interconnected diagrams.

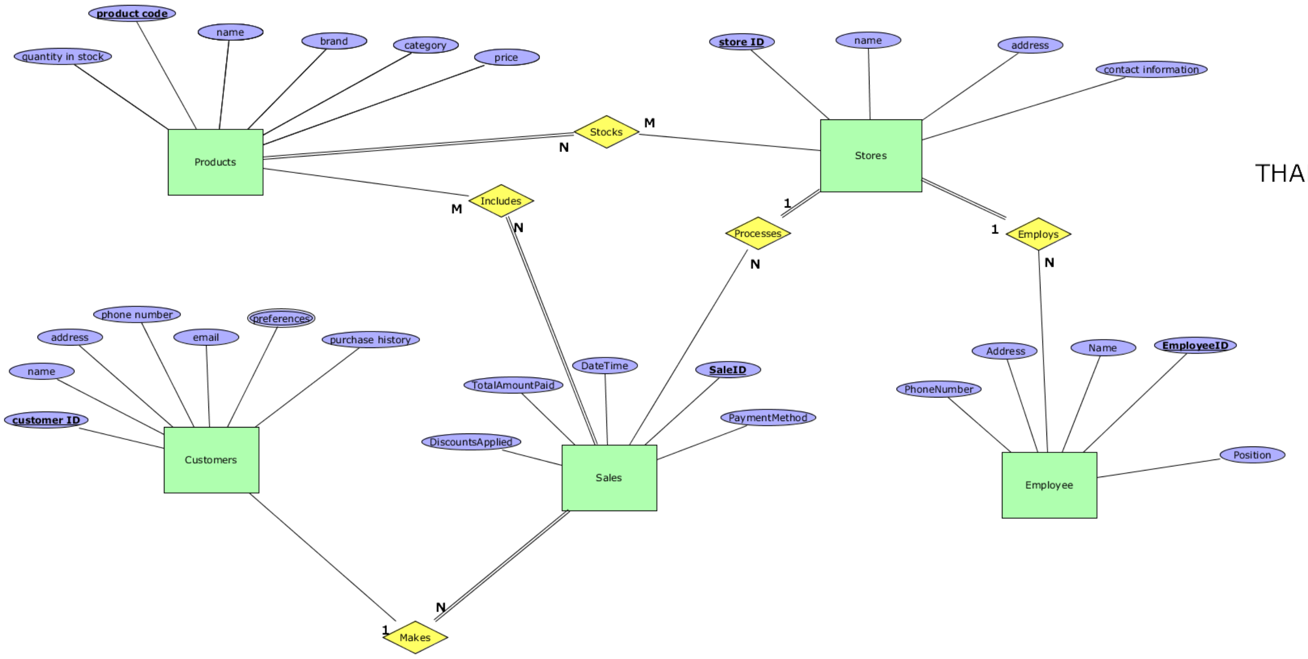
Post Lab:

1) Fashion Hub Retail Company Database:

Problem Description: A retail company named "Fashion Hub" wants to design a database to manage their product inventory and sales. The company sells various fashion items such as clothing, accessories, and footwear. They have multiple physical stores located in different cities. The following information is provided to design the database:

- Products: Fashion Hub sells a wide range of products, including clothing, accessories, and footwear. Each product has a unique product code, name, brand, price, and quantity in stock. Additionally, each product belongs to a specific category (e.g., shirts, dresses, bags, shoes).
- Stores: The company operates multiple physical stores in different cities. Each store has a unique store ID, name, address, and contact information.
- Customers: Customers can create accounts with Fashion Hub to make purchases. Each customer has a unique customer ID, name, address, phone number, and email. The company also stores information about customer preferences and purchase history.
- Sales: Fashion Hub keeps track of sales transactions made by customers. Each sale has a unique sale ID, date and time, customer ID, and store ID. It also includes the total amount paid, any discounts applied, and the payment method used (e.g., cash, credit card).
- Employees: The company employs staff members at each store. Each employee has a unique employee ID, name, address, phone number, and position (e.g., store manager, sales associate).

Design an ER diagram that represents the entities, attributes, relationships, and cardinalities for the Fashion Hub database. Consider any necessary assumptions and justify them if required.



Students Signature

(For Evaluator's use only)

Comment of the Evaluator (if Any)

Evaluator's Observation

Marks Secured:_______ out of ________

Full Name of the Evaluator:

Signature of the Evaluator Date of Evaluation: