Entity-Relationship (ER) Model Study Guide

1. Introduction to ER Modeling

What is an ER Model?

- An Entity-Relationship (ER) Model is a high-level representation of real-world objects (entities) and their relationships.
- It helps in designing databases that are structured, efficient, and to avoid redundancy.
- Example for a retail database:
    - Entities: Customer, Store, Product, Purchase
    - Relationships:
        - A Customer **makes** a Purchase at a Store.
        - A Store **stocks** multiple Products.

Why Use ER Diagrams?

- Visualize database structure before implementation to ensure data integrity and efficiency.
- Guides schema design for databases.

2. Entities and Attributes

What is an Entity?

- An **entity** is an object in the real world that we store information about.
- Examples in retail database:

Entity	Description	Example Attributes
`Customer`	Represents a shopper	`customer_id`, `full_name`, age, `gender`
`Product`	An item available for purchase	`product_id`, `product_category`
`Store`	A retail location	`store_id`, `store_location`, `year_opened`

What is an **Attribute**?

- Each entity has attributes, which describe the entity's characteristics.
- Types of attributes
    - **Simple Attributes**: Cannot be broken down further.
        - *Example:* full_name, age, store_location
    - **Composite Attributes**: Can be divided into subparts.
        - *Example:* full_name → {first_name, last_name}.
    - **Derived Attributes:** Computed from other attributes.

    - *Example:* age can be derived from birth_date, or profit can be derived from product_cost_creation - product_purchased_cost
- **Multivalued Attributes**: An entity can have multiple values.
    - *Example:* phone_numbers for a customer.
- **Key Attributes**: Uniquely identify an entity.
    - *Example:* customer_id (Primary Key for Customer).

3. Relationships and Relationship-Sets

- A relationship defines an association between two or more entities.
- Example:

Relationship	Entities Involved	Example
Makes a Purchase	`Customer` → `Store` → `Product`	A customer buys a product at a store.
Stocks	`Store` → `Product`	A store keeps an inventory of products.
Visits	`Customer` → `Store`	A customer visits a store.

4. Mapping Cardinalities

Cardinality defines how many instances of an entity relate to another, and determines the type of relations.

Type	Meaning	Retail Example
1:1	One entity maps to one other	Each Store has one Admin.
1:M	One entity maps to many others	A Customer makes multiple Purchases.
M:M	Many entities relate to many others	A Store stocks multiple Products, and a Product is stocked in multiple Stores.

If a specific entity *must* have at least one mapping from the entity it maps to, use a double line in the ER diagram.

Types of relations, determined by cardinality:

- One-to-One (1:1)
    - Line with arrow pointing to the “one” (on both sides in this case)
    - Each entity is related to only **one** other entity.
    - *Example:* Each **Admin** manages exactly **one** Store.
- One-to-Many (1:M)
    - Line with arrow pointing to the “one” (on one side in this case)
    - One entity relates to **multiple** other entities.
    - *Example:* A Customer can make **many** Purchases.
- Many-to-Many (M:M)

- Line with no arrow (indicating “many” on both sides)
- Many entities are related to many others.
- *Example:* A Product can be sold in **many** Stores, and a Store sells **many** Products.

5. Weak Entities

A **weak entity** depends on another strong entity and cannot be uniquely identified on its own without the strong entity.

Example: `popularity` table on its own only has the data visited and the foot traffic, which is not enough to uniquely identify the foot traffic for a specific store at a specific location on a specific day. popularity would be a weak entity, as it needs `store_id` and `store_location` from the store table to be “complete.”

6. Specialization and Generalization

- Specialization: Splitting an entity into sub-entities.
  - Ex: User can be specialized into Client and Admin
- Generalization: Merging multiple entities into a superclass.
  - Ex: Admin and Client could be merged into User.
- Constraints in Specialization
  - Disjoint Specialization: An entity can belong to only one subclass.
  - Overlapping Specialization: An entity can belong to multiple subclasses.

7. N-ary Relationships (Relationships Involving More than Two Entities)

A binary relationship involves two entities, but an **N-ary relationship** involves three or more.

Ex: Product Sold at Multiple Stores Across Multiple Competitors

- Entities: Product, Store, Customer
- Relationship: Purchase
- Interpretation: A purchase can be associated with multiple products at a specific store for a specific customer.

8. Translating ER Models to Relational Schemas

Steps to Convert ER Model to SQL Tables

- **Identify Strong Entity-Sets and Create Tables**
  - Strong entity-sets become relations (tables).

    - Each table contains attributes that match the entity-set's attributes.
    - Choose a primary key that uniquely identifies each entity.
    - Ex: customer entity-set → Table with attributes like customer_id, full_name, age, etc.
- **Identify Relationship-Sets and Convert to Tables**
    - Binary relationships *may* require a separate table.
    - The relationship table includes:
        - Primary keys of participating entities as foreign keys.
        - Descriptive attributes related to the relationship.
        - A primary key chosen based on mapping cardinality.
    - Ex: purchase relationship involves customer, product, store. Since it's many-to-many (M:M), the primary key is a composite of entity keys.
- **Determine Primary Keys for Relationship-Sets**
    - Many-to-Many (M:M) → Use composite primary keys from all participating entities.
    - One-to-Many (1:M) → The foreign key of the "many" side is the primary key.
    - One-to-One (1:1) → Either entity's key can be used as the primary key.
- **Handle Weak Entity-Sets**
    - Weak entity-sets depend on strong entity-sets for identification.
    - Use foreign key references from weak entity-sets to strong ones.
    - The primary key of the weak entity-set includes:
        - The primary key of the strong entity.
        - A discriminator (partial key) that differentiates weak entities.
- **Handle N-ary Relationships**
    - If a relationship involves more than two entities, use a junction table.
    - The primary key consists of all foreign keys from the participating entities.
    - Ensure no redundant data when designing N-ary relationships.
- **Convert Specialization & Generalization to Tables**
    - Superclass-subclass relationships can be handled in multiple ways:
        - One table per subclass (each subclass has a foreign key to superclass).
        - Single table with a type discriminator column.
        - One table per entity with shared keys.
    - Ex: Client is specialized into Admin and Client.
        - Can either have separate tables for each or a single table with a role indicator like is_admin.
- **Handle Multivalued Attributes**
- **Create Views for Efficient Data Access**

9. Indexing for Performance

Indexes help speed up queries.

Ex: `CREATE INDEX idx_product_price ON inventory(product_price_usd);`

- Helps **quick lookups** for pricing queries.

10. Views to Simplify complex queries

Ex:

```
CREATE VIEW sales_summary AS
SELECT product_category, SUM(purchased_product_price_usd) AS
total_sales
FROM purchase
JOIN product ON purchase.product_id = product.product_id
GROUP BY product_category;
```