**Chapter 4: Enhanced Entity-Relationship (EER) Model**

1. **Introduction to EER Model**

* The Enhanced Entity-Relationship (EER) Model extends the traditional Entity-Relationship (ER) model by including additional constructs to represent more complex relationships and constraints in databases.
* It is particularly useful for modeling scenarios where entities have attributes or relationships that are not easily represented in the ER model.
* The EER model introduces concepts like specialization, generalization, aggregation, and attribute inheritance to enhance the modeling capabilities.

2. **Subtypes and Supertypes**

* Subtypes and supertypes are used to categorize entities into specialized groupings based on common attributes or relationships.
* A **supertype** is a generalized entity that contains common characteristics shared by one or more specialized entities called **subtypes**.
* Subtypes inherit attributes and relationships from their supertypes, but they can also have their own specific attributes and relationships.
* For example, in a "Vehicle" supertype, "Car" and "Truck" could be subtypes, inheriting common attributes like "Manufacturer" and "Model" from the supertype.

3. **Specialization and Generalization**

* **Specialization** involves defining subtypes based on specific attributes or relationships that distinguish them from other entities.
* **Generalization** involves defining supertypes based on common characteristics shared by subtypes.
* For instance, if we have a "Payment" supertype, it could be specialized into "Credit Card Payment" and "Cash Payment" subtypes based on the method of payment.

4. **Categories of Specialization**

* **Total vs. Partial Specialization:** Total specialization means every entity in the supertype must belong to at least one subtype. Partial specialization allows entities to not belong to any subtype.
* **Overlap vs. Disjoint:** Overlapping subtypes can have entities in common, while disjoint subtypes cannot share entities.
* **Constraint Disjoint vs. Non-Disjoint:** Constraint disjoint means an entity can belong to only one subtype, while non-disjoint allows an entity to belong to multiple subtypes.
* These categories help define the rules for how entities can be classified into subtypes.

5. **Aggregation**

* Aggregation is used to model relationships where a part of an entity is itself an entity.
* For example, an "Order" entity might aggregate "Line Item" entities, where each line item represents a product in the order.
* Aggregation allows for modeling complex relationships without duplicating data, as the aggregated entity is treated as a whole.

6. **Attributes Inheritance**

* Attribute inheritance allows subtypes to inherit attributes from their supertypes.
* This reduces redundancy and ensures consistency in the database design.
* For instance, if "Employee" is a supertype with "Name" and "ID" attributes, "Manager" and "Regular Employee" subtypes can inherit these attributes.

7. **EER Diagrams**

* EER diagrams visually represent the EER model.
* Notations include symbols for entities, attributes, relationships, subtypes, supertypes, specialization, generalization, and aggregation.
* EER diagrams help database designers and stakeholders understand the structure of the database and its complex relationships.

8. **Comparison to ER Model**

* The EER model enhances the traditional ER model by providing more expressive power.
* While the ER model is sufficient for simple database designs, the EER model is necessary for handling more complex data structures and relationships.
* EER model's features like specialization, generalization, aggregation, and attribute inheritance make it a preferred choice for designing databases with intricate requirements.

**Chapter 5: The Relational Data Model and Relational Database Constraints**

5.1 Relational Model Concepts

* **Introduction to Relational Model**:
  + The relational model is a way to organize and structure data in a database. It uses relations (tables) to represent data.
* **Relation (Table)**:
  + A relation is a two-dimensional table with rows and columns.
  + Each row represents a tuple, which is a single record in the table.
  + Each column represents an attribute, which describes a characteristic of the entity being modeled.
* **Tuples and Attributes**:
  + A tuple is a single row in a table, representing a unique instance of the entity being modeled.
  + An attribute is a named column of a relation, describing a property or characteristic of the entity.
* **Primary Key**:
  + The primary key is a unique identifier for each tuple in a relation.
  + It ensures that each row in the table is uniquely identifiable.
* **Domain**:
  + Domain refers to the set of allowable values for an attribute.
  + For example, a "Gender" attribute might have a domain of {Male, Female, Other}.
* **Schema vs. Instance**:
  + The schema of a relation defines its structure, including the name of the relation, attributes, and their types.
  + An instance of a relation is the actual data contained in the table.

5.2 Relational Model Constraints and Relational Database Schemas

* **Key Constraints**:
  + Key constraints ensure that the primary key uniquely identifies each tuple in a relation.
  + It prevents duplicate or null values in the primary key column.
* **Entity Integrity Constraint**:
  + Entity integrity ensures that the primary key attribute cannot have a null value.
  + It guarantees that each tuple in a relation is uniquely identifiable.
* **Referential Integrity Constraint**:
  + Referential integrity ensures that foreign key values in one table match the primary key values they reference in another table.
  + It maintains consistency between related tables.
* **Domain Constraints**:
  + Domain constraints define the allowable values for attributes.
  + For example, an age attribute might have a domain of integers between 0 and 150.
* **Assertions**:
  + Assertions are conditions that must always be true for the database to be in a valid state.
  + They can define more complex rules beyond simple attribute constraints.
* **Relational Database Schema**:
  + The relational database schema is a collection of relation schemas.
  + Each relation schema includes the relation name, attribute names, and their corresponding domains.
  + It also specifies the constraints that apply to each attribute, such as primary key, foreign key, and domain constraints.

5.3 Update Operations, Transactions, and Dealing with Constraint Violations

* **Insertion, Deletion, and Modification**:
  + Insertion is adding new tuples (rows) into a relation.
  + Deletion is removing existing tuples from a relation.
  + Modification is changing the values of attributes in existing tuples.
* **Transactions**:
  + A transaction is a sequence of operations that form a single logical unit of work.
  + It must follow the ACID properties to ensure reliability:
    - **Atomicity**: All operations in a transaction must succeed for the transaction to be committed. If any operation fails, the entire transaction is rolled back.
    - **Consistency**: The database must be in a valid state before and after the transaction.
    - **Isolation**: Transactions should be isolated from each other to prevent interference.
    - **Durability**: Once a transaction is committed, its effects are permanent.
* **Dealing with Constraint Violations**:
  + **Immediate vs. Deferred Constraints**:
    - Immediate constraint checking checks each modification as it occurs.
    - Deferred constraint checking checks constraints at the end of the transaction.
  + **Exception Handling**:
    - When a constraint violation occurs, an exception is raised.
    - The database system can either roll back the entire transaction or reject the operation that caused the violation.
    - This ensures that the database remains in a consistent state.

**9.1 Relational Database Design Using ER-to-Relational Mapping**

9.1.1 ER-to-Relational Mapping Algorithm

**Step 1: Mapping of Regular Entity Types**

* For each regular (strong) entity type E:
  + Create a relation R with all simple attributes of E.
  + Choose one key attribute of E as the primary key for R.
  + If the key of E is composite, use the components as the primary key.
  + Include additional keys as unique keys if identified during conceptual design.

**Step 2: Mapping of Weak Entity Types**

* For each weak entity type W with owner entity type E:
  + Create a relation R with all simple attributes of W.
  + Include primary key(s) of owner(s) as foreign key(s) in R.
  + Primary key of R is a combination of owner's primary key(s) and weak entity's partial key.

**Step 3: Mapping of Binary 1:1 Relationship Types**

* Choose one side, say S, as a relation.
* Include primary key of the other side, say T, as foreign key in S.
* Include attributes of the 1:1 relationship as attributes of S.

**Step 4: Mapping of Binary 1:N Relationship Types**

* For each regular binary 1:N relationship type R:
  + Include primary key of "one" side as foreign key in "many" side.
  + Include attributes of the 1:N relationship as attributes of "many" side.

**Step 5: Mapping of Binary M:N Relationship Types**

* Create a new relation S for each M:N relationship type R.
* Include primary keys of participating entities as foreign keys in S.
* Include attributes of the M:N relationship as attributes of S.

**Step 6: Mapping of Multivalued Attributes**

* Create a new relation R for each multivalued attribute A.
* Include A and primary key of entity/relationship with A as foreign key in R.
* Primary key of R is combination of A and entity/relationship primary key.

**Step 7: Mapping of N-ary Relationship Types (n > 2)**

* Use the relationship relation approach.
* Create a new relation S with primary keys of participating entities as foreign keys.
* Include attributes of the N-ary relationship as attributes of S.

9.1.1 ER-to-Relational Mapping Algorithm Example

**Step 1: Mapping of Regular Entity Types**

* **EMPLOYEE** (SSN, Fname, Minit, Lname, Bdate, Address, Sex, Salary, Super\_ssn, Dno)
  + Primary Key: SSN
  + Partial Key: None
  + Foreign Key: Dno (references DEPARTMENT)
* **DEPARTMENT** (Dname, Dnumber, Mgr\_ssn, Mgr\_start\_date)
  + Primary Key: Dnumber
  + Partial Key: None
  + Foreign Key: Mgr\_ssn (references EMPLOYEE)
* **PROJECT** (Pname, Pnumber, Plocation, Dnum)
  + Primary Key: Pnumber
  + Partial Key: None
  + Foreign Key: Dnum (references DEPARTMENT)

**Step 2: Mapping of Weak Entity Types**

* **DEPENDENT** (Essn, Dependent\_name, Sex, Bdate, Relationship)
  + Primary Key: {Essn, Dependent\_name}
  + Partial Key: None
  + Foreign Key: Essn (references EMPLOYEE)

**Step 3: Mapping of Binary 1:1 Relationship Types**

* **MANAGES** (Dnumber, Mgr\_start\_date)
  + Primary Key: Dnumber (from DEPARTMENT)
  + Foreign Key: Mgr\_ssn (references EMPLOYEE)

**Step 4: Mapping of Binary 1:N Relationship Types**

* **WORKS\_FOR** (Essn, Dno)
  + Primary Key: None (Composite of Essn, Dno)
  + Partial Key: None
  + Foreign Keys:
    - Essn (references EMPLOYEE)
    - Dno (references DEPARTMENT)
* **CONTROLS** (Dnum, Pno)
  + Primary Key: None (Composite of Dnum, Pno)
  + Partial Key: None
  + Foreign Keys:
    - Dnum (references DEPARTMENT)
    - Pno (references PROJECT)
* **SUPERVISION** (Essn, Super\_ssn)
  + Primary Key: None (Composite of Essn, Super\_ssn)
  + Partial Key: None
  + Foreign Keys:
    - Essn (references EMPLOYEE)
    - Super\_ssn (references EMPLOYEE)

**Step 5: Mapping of Binary M:N Relationship Types**

* **WORKS\_ON** (Essn, Pno, Hours)
  + Primary Key: None (Composite of Essn, Pno)
  + Partial Key: None
  + Foreign Keys:
    - Essn (references EMPLOYEE)
    - Pno (references PROJECT)

**Step 6: Mapping of Multivalued Attributes**

* **DEPT\_LOCATIONS** (Dnumber, Dlocation)
  + Primary Key: None (Composite of Dnumber, Dlocation)
  + Partial Key: None
  + Foreign Key: Dnumber (references DEPARTMENT)

**9.2 Mapping EER Model Constructs to Relations**

In this section, we extend the ER-to-relational mapping algorithm to handle EER model constructs, specifically focusing on the mapping of specialization or generalization.

9.2.1 Mapping of Specialization or Generalization

There are several options for mapping a specialization with subclasses or a generalization with a superclass. The main options are:

Step 8: Options for Mapping Specialization or Generalization

1. **Option 8A: Multiple Relations - Superclass and Subclasses**
   * Create a relation for the superclass C with attributes and primary key.
   * Create a relation for each subclass Si with attributes and primary key, including the primary key of the superclass as a foreign key.
   * This option works for any specialization (total or partial, disjoint or overlapping).
2. **Option 8B: Multiple Relations - Subclass Relations Only**
   * Create a relation for each subclass Si with attributes and primary key, including the primary key of the superclass as a foreign key.
   * This option only works for total specialization (every entity in the superclass belongs to at least one subclass) with disjointedness constraint.
   * Not recommended for overlapping subclasses.
3. **Option 8C: Single Relation with One Type Attribute**
   * Create a single relation with attributes of the superclass and all subclasses, plus a type (or discriminating) attribute.
   * The type attribute indicates the subclass each tuple belongs to.
   * Works for disjoint subclasses, potential for many NULL values.
4. **Option 8D: Single Relation with Multiple Type Attributes**
   * Create a single relation with attributes of the superclass and all subclasses, plus multiple Boolean type attributes.
   * Each Boolean attribute indicates whether a tuple belongs to a specific subclass.
   * Works for overlapping subclasses and can be used for disjoint subclasses.

Examples of Mapping Options:

**Option 8A Example:**

* **EMPLOYEE** (Ssn, Fname, Lname, Birth\_date, Address, Job\_type, Typing\_speed, Tgrade, Eng\_type)
  + Primary Key: Ssn
* **SECRETARY** (Ssn, Typing\_speed)
  + Primary Key: Ssn (foreign key to EMPLOYEE)
* **TECHNICIAN** (Ssn, Tgrade)
  + Primary Key: Ssn (foreign key to EMPLOYEE)
* **ENGINEER** (Ssn, Eng\_type)
  + Primary Key: Ssn (foreign key to EMPLOYEE)

**Option 8B Example:**

* **SECRETARY** (Ssn, Typing\_speed, Job\_type)
  + Primary Key: Ssn
* **TECHNICIAN** (Ssn, Tgrade, Job\_type)
  + Primary Key: Ssn
* **ENGINEER** (Ssn, Eng\_type, Job\_type)
  + Primary Key: Ssn

**Option 8C Example:**

* **EMPLOYEE** (Ssn, Fname, Lname, Birth\_date, Address, Job\_type, Typing\_speed, Tgrade, Eng\_type, Employee\_type)
  + Primary Key: Ssn
  + Type Attribute: Employee\_type (values: 'SECRETARY', 'TECHNICIAN', 'ENGINEER')

**Option 8D Example:**

* **EMPLOYEE** (Ssn, Fname, Lname, Birth\_date, Address, Job\_type, Typing\_speed, Tgrade, Eng\_type, Is\_secretary, Is\_technician, Is\_engineer)
  + Primary Key: Ssn
  + Boolean Type Attributes: Is\_secretary, Is\_technician, Is\_engineer (values: true/false)

Conclusion:

* **Option 8A** creates separate relations for the superclass and each subclass.
* **Option 8B** creates separate relations only for the subclasses.
* **Option 8C** creates a single relation with a type attribute for disjoint subclasses.
* **Option 8D** creates a single relation with multiple Boolean type attributes for overlapping subclasses.