Oral Exam Questions and Answers for Database Systems (Lecture 4)

Question 1: How do you map a weak entity type to a relational table?

Answer: To map a weak entity type:

- Create a table for the weak entity.
- Include all attributes of the weak entity.
- Add a foreign key that corresponds to the primary key of the owner (strong) entity type.
- The primary key of the weak entity table is typically a composite key that includes the foreign key and a partial key from the weak entity.

Example: For a DEPENDENT weak entity related to EMPLOYEE, create a table DEPENDENT(Essn, Dependent_name, Sex, Bdate, Relationship), where Essn is a foreign key referencing EMPLOYEE's primary key (Ssn).

Question 2: How is a binary 1:1 relationship with both sides mandatory mapped to relational tables?

Answer: For a binary 1:1 relationship where both sides are mandatory:

- Merge the two entities into a single table.
- The table includes all attributes from both entities.
- Choose either entity's primary key as the primary key of the merged table.

Example: For entities X and Y, create a single table tbl_xy(PK, ...), where PK is either PKx or PKy. For example, END(EID, Ename, Cname, CID).

Question 3: How do you map a binary 1:1 relationship where one side is optional and the other is mandatory?

Answer: For a binary 1:1 relationship where one side (X) is optional and the other (Y) is mandatory:

- Create two separate tables: one for each entity (tbl_x and tbl_y).
- Add the primary key of the optional side (PKx) as a foreign key in the mandatory side's table (tbl_y).

Example: For Chapter (mandatory) and another entity (optional), create Chapter(CID, Cname, EID_FK), where EID_FK references the optional entity's primary key.

Question 4: How do you map a binary 1:1 relationship where both sides are optional?

Answer: For a binary 1:1 relationship where both sides are optional:

- Create three tables: one for each entity (tbl_x, tbl_y) and a third table (tbl_xy) for the relationship.
- The relationship table (tbl_xy) includes foreign keys referencing the primary keys of both entities (PKx, PKy) and any relationship attributes.
- The primary key of tbl_xy can be either PKx or PKy.

Example: For entities X and Y, create tbl_x(PKx, ...), tbl_y(PKy, ...), and tbl_xy(PKxy, FKxy, ...), where PKxy is PKx or PKy.

Question 5: How is a binary 1:N relationship mapped when the N-side is mandatory?

Answer: For a binary 1:N relationship where the N-side is mandatory:

- Create two tables: one for the 1-side (tbl_x) and one for the N-side (tbl_y).
- Add the primary key of the 1-side (PKx) as a foreign key in the N-side table (tbl_y).
- Include any simple attributes of the relationship in the N-side table.

Example: For Department (1-side) and Employee (N-side), create Department(DID, Dname) and Employee(EID, Ename, DID_FK), where DID_FK references Department's DID.

Question 6: How do you map a binary 1:N relationship when the N-side is optional?

Answer: For a binary 1:N relationship where the N-side is optional:

- Create three tables: one for the 1-side (tbl_x), one for the N-side (tbl_y), and a third table (tbl_xy) for the relationship.
- The relationship table includes the primary key of the N-side (PKy) as its primary key and a foreign key referencing the 1-side (PKx).

Example: For a 1:N relationship, create tbl_x(PKx, ...), tbl_y(PKy, ...), and tbl_xy(PKxy, ...), where PKxy = PKy and includes a foreign key to PKx.

Question 7: How do you map a binary M:N relationship to relational tables?

Answer: For a binary M:N relationship:

- Create three tables: one for each entity (tbl_x, tbl_y) and a third table (tbl_xy) for the relationship.

- The relationship table (tbl_xy) includes foreign keys referencing the primary keys of both entities (PKx, PKy) and any simple attributes of the relationship.
- The primary key of tbl_xy is the combination of PKx and PKy.

Example: For Student and Course, create Student(SID, Sname), Course(CID, Cname), and Stud_Course(SID, CID).

Question 8: How do you map an M:N relationship with attributes, such as the Supplies relationship?

Answer: For an M:N relationship with attributes:

- Create three tables: one for each entity (e.g., RAW_MATERIALS, VENDOR) and a third table for the relationship (e.g., QUOTE).
- The relationship table includes foreign keys for both parent tables (Material_ID, Vendor_ID) and the relationship attribute (e.g., Unit_Price).
- The primary key of the relationship table is the combination of the foreign keys.

Example: QUOTE(Material_ID, Vendor_ID, Unit_Price), where Material_ID and Vendor_ID are foreign keys.

Question 9: How do you map an N-ary relationship (where n > 2) to relational tables?

Answer: For an N-ary relationship (n > 2):

- Create a new table for the relationship.
- Add foreign keys referencing the primary keys of all participating entities.
- Include any simple attributes of the relationship in the new table.

Example: For a ternary relationship involving entities A, B, and C, create a table with foreign keys for A, B, and C's primary keys, plus any relationship attributes.

Question 10: How is a unary (recursive) relationship mapped to a relational table?

Answer: For a unary relationship:

- Create a single table for the entity.
- Add a recursive foreign key that references the same table's primary key to represent the relationship.

Example: For EMPLOYEE with a recursive relationship (e.g., an employee manages another employee), create EMPLOYEE(Employee_ID, Name, Birthdate, Manager_ID), where Manager_ID is a foreign key referencing Employee_ID.

Question 11: What is the Enhanced Entity-Relationship (EER) model, and how does it differ from the ER model?

Answer: The EER model extends the ER model to design more accurate database schemas that reflect complex data properties and constraints. It includes all ER model concepts (entities, attributes, relationships) and adds:

- Subclasses and superclasses
- Specialization and generalization
- Category or union types
- Attribute and relationship inheritance

Difference: The EER model supports more complex requirements, such as hierarchical relationships and inheritance, which the ER model does not address.

Question 12: What are subclasses, superclasses, and inheritance in the EER model?

Answer:

- **Superclass**: An entity type with one or more distinct subgroupings (subclasses) that need to be represented in the data model.
- **Subclass**: A distinct subgroup of a superclass's entities, with specific attributes or relationships.
- **Inheritance**: A subclass inherits all attributes and relationships of its superclass, plus any additional attributes or relationships defined for the subclass.

Example: EMPLOYEE (superclass) with subclasses Secretary and Engineer, where both inherit EMPLOYEE's attributes (e.g., Name, Ssn) and have specific attributes (e.g., Job_type).

Question 13: What is the difference between specialization and generalization in the EER model?

Answer:

- **Specialization**: The process of defining subclasses of an entity type (superclass) by identifying distinguishing characteristics, adding specific attributes or relationships to each subclass.
- **Generalization**: The process of minimizing differences between entities by identifying common characteristics to create a superclass.

Example: Specialization creates Secretary and Engineer from EMPLOYEE based on Job_type. Generalization combines similar entities into a common superclass like EMPLOYEE.

Question 14: What are participation and disjoint constraints in specialization/generalization?

Answer:

- **Participation Constraint**: Determines if every superclass member must belong to a subclass.
    - **Total Specialization**: Every superclass member must be in a subclass (shown with a double line).
    - **Partial Specialization**: Superclass members may not belong to any subclass (shown with a single line).
- **Disjoint Constraint**: Indicates whether a superclass member can belong to multiple subclasses.
    - **Disjoint (d)**: A member belongs to only one subclass.
    - **Overlapping (o)**: A member can belong to multiple subclasses.

Question 15: What is the difference between a hierarchy and a lattice in the EER model?

Answer:

- **Hierarchy**: A subclass has only one superclass, resulting in single inheritance.
- **Lattice**: A subclass (shared subclass) has multiple superclasses, leading to multiple inheritance.

Example: ENGINEERING_MANAGER as a shared subclass of ENGINEER, MANAGER, and SALARIED_EMPLOYEE forms a lattice with multiple inheritance. A single superclass like EMPLOYEE with subclasses Secretary and Engineer forms a hierarchy.

Question 16: What is a union type or category in the EER model, and how does it differ from a shared subclass?

Answer:

- **Union Type/Category**: A subclass representing a subset of the union of distinct entity types (superclasses). An entity in the category belongs to at least one superclass.
- **Shared Subclass**: A subclass that is a subset of the intersection of multiple superclasses, requiring membership in all superclasses.

Example: OWNER (category) is a subset of the union of COMPANY, BANK, or PERSON (an entity belongs to one). ENGINEERING_MANAGER (shared subclass) must belong to ENGINEER, MANAGER, and SALARIED_EMPLOYEE (intersection).

Question 17: In the company example, how would you map the WORKS_ON relationship to a relational table?

Answer: The WORKS_ON relationship is an M:N relationship between EMPLOYEE and PROJECT with an attribute (Hours):

- Create three tables: EMPLOYEE(Essn, Fname, Lname, ...), PROJECT(Pnumber, Pname, ...), and WORKS_ON(Essn, Pno, Hours).
- WORKS_ON has foreign keys Essn (referencing EMPLOYEE's Ssn) and Pno (referencing PROJECT's Pnumber), with Hours as a relationship attribute.
- The primary key of WORKS_ON is the combination of Essn and Pno.

Question 18: How would you represent the DEPENDENT entity in the company example as a relational table?

Answer: DEPENDENT is a weak entity related to EMPLOYEE:

- Create a table DEPENDENT(Essn, Dependent_name, Sex, Bdate, Relationship).
- Essn is a foreign key referencing EMPLOYEE's Ssn.
- The primary key is a composite key: (Essn, Dependent_name), as Dependent_name is the partial key.

Question 19: How would you model an EER specialization for EMPLOYEE with subclasses Secretary and Engineer?

Answer:

- **Superclass**: EMPLOYEE(Fname, Lname, Ssn, Birth_date, Address, Job_type).
- **Subclasses**: Secretary and Engineer, each inheriting EMPLOYEE's attributes.
- Add specific attributes to each subclass (e.g., Typing_speed for Secretary, Engineering_field for Engineer).
- **Diagram**: EMPLOYEE is a rectangle connected to a circle (specialization) with "d" (disjoint) or "o" (overlapping). Subclasses Secretary and Engineer are connected to the circle, with a double line for total specialization or a single line for partial specialization.

Question 20: How would you map a category like OWNER in the EER model to relational tables?

Answer:

- For a category like OWNER (a subset of the union of COMPANY, BANK, PERSON):
- Create separate tables for each superclass: COMPANY(CID, ...), BANK(BID, ...), PERSON(PID, ...).
- Create a table for OWNER, e.g., OWNER(Owner_ID, Owner_Type, CID_FK, BID_FK, PID_FK), where:
    - Owner_ID is the primary key.
    - Owner_Type indicates the superclass (e.g., "C" for COMPANY, "B" for BANK, "P" for PERSON).
    - CID_FK, BID_FK, PID_FK are foreign keys referencing the respective superclasses, with only one populated per row.