Homework 2

3.1. Discuss the role of a high-level data model in the database design process

i] High-level data model do not include implementation details; they are easy to understand and are useful in communicating with non-technical users.
ii] it can also be used as reference to ensure that all user requirements are met and that requirements do not conflict with each other.
iii] this also enables database designers to concentrate on specifying the properties of data without being concerned with storage details

3.2. List the various cases where use of a NULL value would be appropriate.

In some cases, a particular entity may not have an applicable value for an attribute.
ex:
i] The Apartment number attribute of an address applies only to addresses that are in apartment buildings and not to other types of residences, such as single-family homes

ii] A College degree attribute applies only to people with college degrees.

3.3. Define the following terms: entity, attribute, attribute value, relationship instance, composite attribute, multivalued attribute, derived attribute, complex attribute, key attribute, and value set (domain).

Entity: A thing or object in the real world with an independent existence; can be an object with a physical existence or an object with a conceptual existence.

Attribute: The particular properties that describe an entity

Attribute value: The value of an attribute for a particular entity.

Relationship Instance: Each relationship instance in a relationship type is an association of entities, where the association includes exactly one entity from each participating entity type. Each relationship instance represents the fact that the entities participating in it are related in some way in the corresponding situation.

Composite Attribute: A composite attribute is an attribute that's composed of multiple simpler attributes. Useful in cases where you sometimes refer to the entire attribute and sometimes refer to its simpler attributes on their own.

Multivalued Attribute: An attribute that can be composed of multiple values; shown on an ERD as a double-outlined oval ex: The Color attribute of CAR instance like for a car dealership who has a bunch of cars in stock

Derived Attribute: An attribute whose value is derived from one or more other attributes; shown on an ERD as a dotted-outlined oval. ex: Age attributed derived from DOB attribute.

Complex Attribute: A composite and/or multivalued attribute can be nested arbitrarily. Can represent arbitrary nesting by grouping components of a composite attribute between parentheses () and separating the values with commas.

Key Attribute: One or more attributes whose values are distinct for each individual entity in the entity set. Its values can be used to identify each entity uniquely. ex: the SSN attribute of a PERSON entity

Value set: Specifies the set of values that may be assigned to that attribute for each individual entity. Similar to the basic data types available in most programming languages. Not specified in ERDs but can be specified in UML class diagrams. The value set provides all possible values.

3.4. What is an entity type? What is an entity set? Explain the differences among an entity, an entity type, and an entity set.

Entity: A thing or object in the real world with an independent existence; can be an object with a physical existence or an object with a conceptual existence.

Entity type: Defines a collection of entities that have the same attributes. Each entity type in the DB is described by its name and attributes. Ex: entity types: Department, Employee, Project, Dependent.

Entity set: Entity is also known as entity collection. The collection of all entities of a particular entity type in the DB at any point in time. Ex: Employee entity set = The set of all employee instances in the DB at a given point in time.

3.5. Explain the difference between an attribute and a value set.

Value set: Each simple attribute of an entity type is associated with a value set (or domain of values), which specifies the set of values that may be assigned to that attribute for each individual entity.

Attributes: An entity, which is a thing or object in the real world with an independent existence. An entity may be an object with a physical existence (for example, a particular per- son, car, house, or employee) or it may be an object with a conceptual existence (for instance, a company, a job, or a university course). Each entity has attributes—the particular properties that describe it.

3.6. What is a relationship type? Explain the differences among a relationship instance, a relationship type, and a relationship set.

Relationship type: A relationship type, R, among n entity types, E1, E2, ..., En defines a set of associations among entities from these entity types.

Relationship instance: Each relationship instance in a relationship type is an association of entities, where the association includes exactly one entity from each participating entity type. Each relationship instance represents the fact that the entities participating in it are related in some way in the corresponding situation.

Relationship set: Relationship set R is a set of relationship instances r_i, where each r_i associates n individual entities ($e_1, e_2, \ldots, e_n$), and each entity e_j in r_i is a member of entity set E_j, $1 \leq j \leq n$. Hence, a relationship set is a mathematical relation on $E_1, E_2, \ldots, E_n$. It can be defined as a subset of the Cartesian product of the entity sets $E_1 \times E_2 \times \ldots \times E_n$.

3.7. What is a participation role? When is it necessary to use role names in the description of relationship types?

Participation role: Each entity type that participates in a relationship type plays a particular role in the relationship. The role name signifies the role that a participating entity from the entity type plays in each relation- ship instance, and it helps to explain what the relationship means. Ex: In the WORKS-FOR relationship type, EMPLOYEE plays the role of employee or worker and DEPARTMENT plays the role of department or employer.

Role Name: Role names are not technically necessary in relationship types where all the participating entity types are distinct, since each participating entity type name can be used as the role name. However, in some cases the same entity type participates more than once in a relationship type in different roles. In such cases the role name becomes essential for distinguishing the meaning of the role that each participating entity plays. Such relationship types are called recursive relationships or self-referencing relationships. ex: The SUPERVISION relationship type relates an employee to a supervisor, where both employee and supervisor entities are members of the same EMPLOYEE entity set. Hence, the EMPLOYEE entity type participates twice in SUPERVISION: once in the role of supervisor (or boss), and once in the role of supervisee (or subordinate).

3.8. Describe the two alternatives for specifying structural constraints on relationship types. What are the advantages and disadvantages of each?

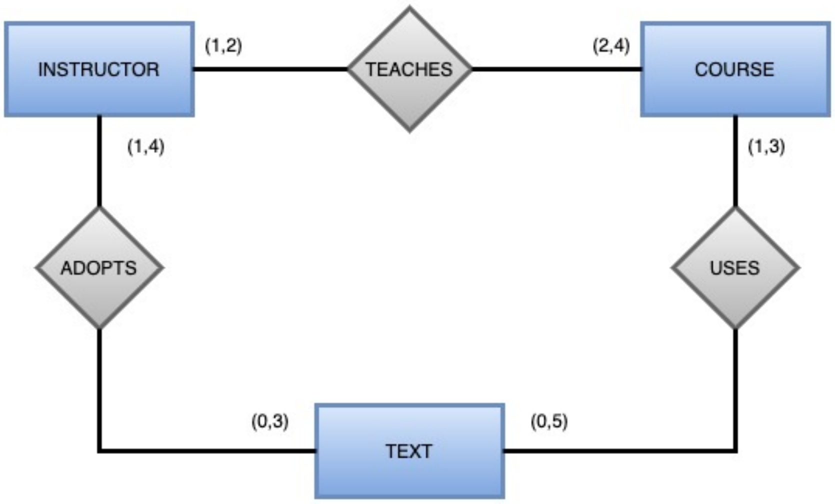
<u>Cardinality ratios</u>: The cardinality ratio for a binary relationship specifies the maximum number of relationship instances that an entity can participate in.
Ex: In the WORKS_FOR binary relationship type, DEPARTMENT: EMPLOYEE is of cardinality ratio 1:N, meaning that each department can be related to any number of employees (N), but an employee can be related to (work for) at most one department (1). This means that for this particular relationship type WORKS_FOR, a particular department entity can be related to any number of employees (N indicates there is no maximum number). On the other hand, an employee can be related to a maximum of one department. The possible cardinality ratios for binary relationship types are 1:1, 1:N, N:1, and M:N.

<u>Participation constraints</u>: The participation constraint specifies whether the existence of an entity depends on its being related to another entity via the relationship type. This constraint specifies the *minimum* number of relationship instances that each entity can participate in and is some- times called the <u>minimum cardinality constraint.</u>
There are two types of participation constraints total and partial.

Ex: If a company policy states that *every* employee must work for a department, then an employee entity can exist only if it participates in at least one WORKS_FOR relationship instance (Figure 3.9). Thus, the participation of EMPLOYEE in WORKS_FOR is called <u>total participation</u>, meaning that every entity in *the total set* of employee entities must be related to a department entity via WORKS_FOR. Total participation is also called <u>existence dependency</u>.

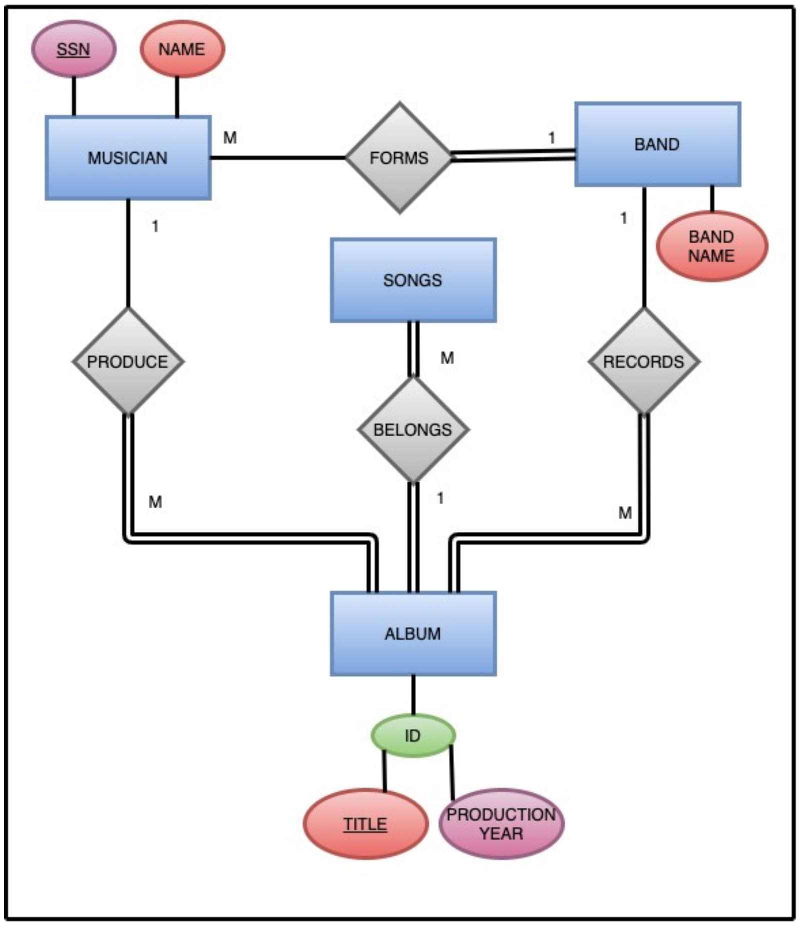
Ex: We do not expect every employee to manage a department, so the participation of EMPLOYEE in the MANAGES relationship type is <u>partial</u>, meaning that *some* or *part of the set of* employee entities are related to some department entity via MANAGES, but not necessarily all.

Exercise 3.25



Assumptions:
Textbook can be adopted by one or maximum 4 instructors.
Course can be taught by one to two instructors.
Instructor may or may not use upto 3 textbooks.
Textbook is used in one to maximum of three courses.

Recording Studio Exercise:



4.1 What is a subclass? When is a subclass needed in data modeling?

It is same as entity in the superclass, but in a distinct specific role. It can be represented as a distinct database object which can be related via key attributes. A subclass is a class whose entities must always be a subset of the entities in another class, called the superclass of the superclass/subclass (or IS-A) relationship.
Reason:
i] Certain attributes may apply to some but not all entities of the superclass entity type. A subclass is defined in order to group the entity to which these attributes apply. The member of subclass may still share the majority of their attribute with other members of the superclass.
ii] Some relationship type may be participated in only by the entities that are member of the subclass.

4.2 Define the following terms: superclass of a subclass, superclass/subclass relationship, IS-A relationship, specialization, generalization, category, specific (local) attributes, and specific relationships.

Subclass:
A relationship between a superclass and subclasses and any one of its subclasses. The subclasses inherit the characteristics of a superclass.

Superclass/Subclass relationship:
Superclass/subclass relationship is one-to-one (1:1). Each member of a subclass is also a member of the superclass but has a distinct role.

IS-A relationship:
The relationship between a superclass and a subclass. Entities of the subclass inherit the attribute(s) and relationships of the superclass.

Specialization:
It is the process of defining a set of subclasses of an entity type; this entity type is called the superclass of the specialization.

Generalization:
Generalization is a process in which two or more entities of lower level combine to form a higher-level entity if they have some attributes in common. Like two or more subclasses entity are merged to form one generalized superclass.

Category:
A category has two or more super-classes that may represent collections of entities from distinct entity types.
Specific(local) attributes: Attributes that apply only to entities of a particular subclass.
Specific relationship: Relationships that apply only to entities of a particular subclass.

Specific Attributes:
Attributes that apply only to entities of a particular subclass.

Specific relationships:
Relationships that apply only to entities of a particular subclass.

4.3 Discuss the mechanism of attribute/relationship inheritance. Why is it useful?

The type of an entity is defined by the attributes it possesses and the relationship types in which it participates. Because an entity in the subclass represents the same real-world entity from the superclass, it should possess values for its specific attributes as well as values as a member of the superclass. We say that an entity that is a member of a subclass inherits all the attributes of the entity member of the superclass. The entity also inherits all the relationships in which the superclass participates.

4.4 Discuss user-defined and predicate-defined subclasses and identify the differences between the two.

Predicate-defined: We can determine exactly the entities that will become members of each subclass by placing a condition on the value of some attribute of the superclass. Such subclasses are called predicate defined.

Predicate: the attribute of the superclass that the constraint is placed on.

User-defined: When we do not have a condition for determining membership in a subclass, the subclass is called user-defined.

4.5 Discuss user-defined and attribute-defined specializations and identify the differences between the two.

Attribute-defined: If all subclasses in a specialization have their membership condition on the same attribute of the superclass, the specialization itself is called an attribute-defined specialization, and the attribute is called the defining attribute of the specialization.

User-defined: When we do not have a condition for determining membership in a subclass, the subclass is called user-defined

4.6 Discuss the two main types of constraints on specializations and generalizations.

The first is the disjointness constraint, which specifies that the subclasses of the specialization must be disjoint sets. This means that an entity can be a member of at most one of the subclasses of the specialization. A specialization that is attribute-defined implies the disjointness constraint (if the attribute used to define the membership predicate is single- valued). If the subclasses are not constrained to be disjoint, their sets of entities may be overlapping; that is, the same (real-world) entity may be a member of more than one subclass of the specialization.

The second constraint on specialization is called the completeness (or totalness) constraint, which may be total or partial. A total specialization constraint specifies that every entity in the superclass must be a member of at least one subclass in the specialization.

Four possible constraints on a specialization:

- Disjoint, total
- Disjoint, partial
- Overlapping, total
- Overlapping, partial

4.7 What is the difference between a specialization hierarchy and a specialization lattice?

Specialization hierarchy:
It has the constraint that every subclass participates as a subclass in only one class/subclass relationship And that is each subclass has only one parent results in a tree structure or strict hierarchy.

Specialization lattice:
A subclass can be a subclass in more than one class/subclass relationship that is each subclass can have more than one parent.

4.8 What is the difference between specialization and generalization? Why do we not display this difference in schema diagrams?

Specialization and generalization are inverses of each other. A superclass that was defined through the generalization process is usually total because the superclass is derived from the subclasses and hence contains only the entities that are in the subclasses. In the specialization process, the DB designers typically start with an entity type and then define subclasses of the entity type by successive specialization; that is, they repeatedly define more specific groupings of entity types top-down conceptual refinement. In the generalization process, the DB designers start with a bunch of entity types and then generalize them bottom-up conceptual synthesis.

Both can end up with the same hierarchy or lattice in the end. A combo of the 2 processes is employed.
We do not distinguish between specialization and generalization on diagrams because the decision as to which process was followed in a particular situation is often subjective.

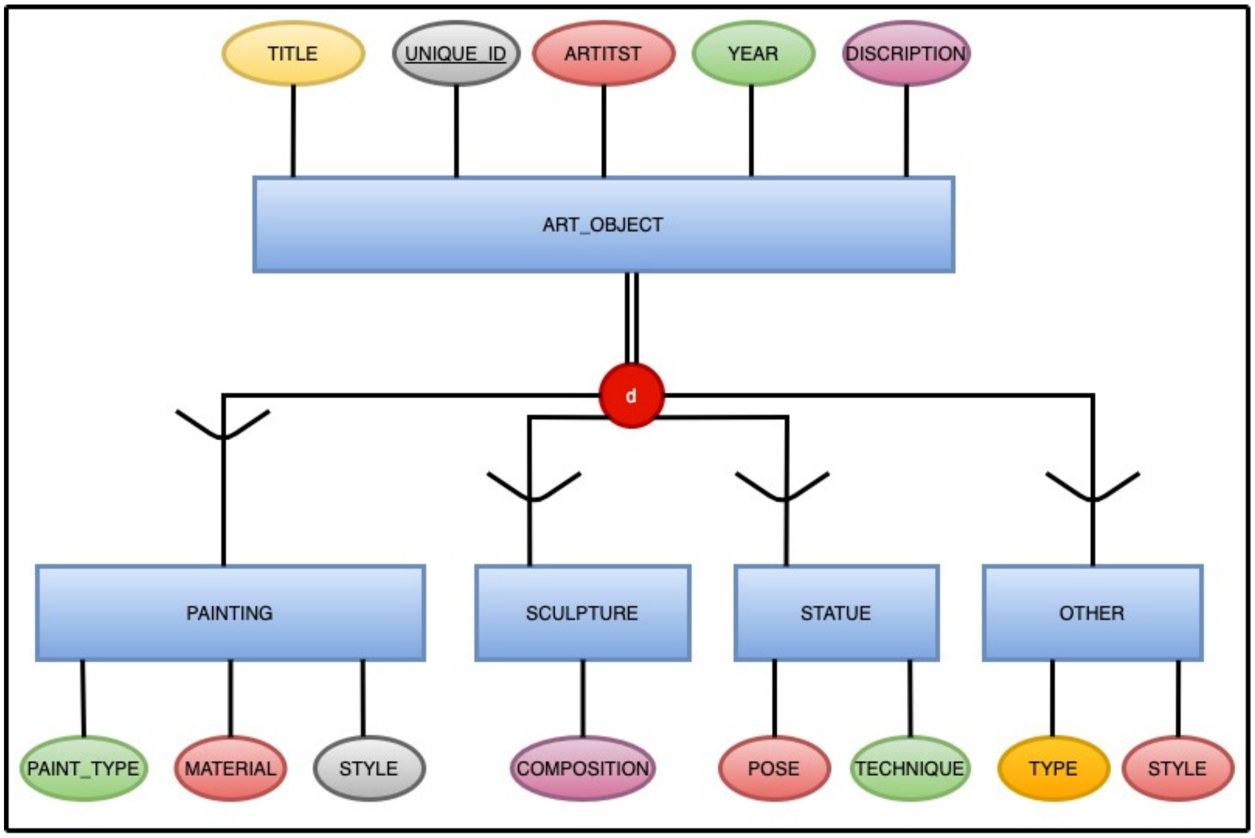
4.9 How does a category differ from a regular shared subclass? What is a category used for? Illustrate your answer with examples.

A category has two or more super-classes that may represent collections of entities from distinct entity types, whereas other super/subclass relationships always have a single superclass with a shared subclass, the subclass entity set is a subset of the intersection of the super-classes that represents with a category, the category is a subset of the union of the super-classes; represents or attribute/relationship inheritance is also more selective with categories because a member of the category only inherits the attributes and relationships of the one superclass of which it belongs whereas with shared subclasses, a member of the shared subclass inherits the attributes of all of its super-classes.

Ex: Categories:
super-classes = bank, person, company; category=owner (like of a vehicle); the owner is either the bank, a person, a company.

Ex: Shared subclasses:
super-classes = engineer, manager, salaried employee, shared subclass = engineering manager; the engineering manager must be an engineer, a manager, a salaried employee.

Exercise 4.20


TITLE
UNIQUE_ID
ARTITST
YEAR
DISCRIPTION
ART_OBJECT
d
PAINTING
SCULPTURE
STATUE
OTHER
PAINT_TYPE
MATERIAL
STYLE
COMPOSITION
POSE
TECHNIQUE
TYPE
STYLE