**Practical No. 5**

**Aim:** Modeling UML ER Diagrams and Sequence diagrams

**Theory:**

**Entity Relationship Model**

Entity-Relationship model is used to represent a logical design of a database to be created. In ER model, real world objects (or concepts) are abstracted as entities, and different possible associations among them are modeled as relationships.

**Entity Set and Relationship Set**

An entity set is a collection of all similar entities. For example, "Student" is an entity set that abstracts all students. Ram, John are specific entities belonging to this set. Similarly, a "Relationship" set is a set of similar relationships.

**Attributes of Entity**

Attributes are the characteristics describing any entity belonging to an entity set. Any entity in a set can be described by zero or more attributes.

For example, any student has got a name, age, an address. At any given time a student can study only at one school. In the school he would have a roll number, and of course a grade in which he studies. These data are the attributes of the entity set Student.

**Keys**

**One or more attribute(s) of an entity set can be used to define the following keys:**

* **Super key:** One or more attributes, which when taken together, helps to uniquely identify an entity in an entity set. For example, a school can have any number of students. However, if we know grade and roll number, then we can uniquely identify a student in that school.
* **Candidate key:** It is a minimal subset of a super key. In other words, a super key might contain extraneous attributes, which do not help in identifying an object uniquely. When such attributes are removed, the key formed so is called a candidate key.
* **Primary key:** A database might have more than one candidate key. Any candidate key chosen for a particular implementation of the database is called a primary key.
* **Prime attribute:** Any attribute taking part in a super key

**Weak Entity**

An entity set is said to be weak if it is dependent upon another entity set. A weak entity can't be uniquely identified only by it's attributes. In other words, it doesn't have a super key.

For example, consider a company that allows employees to have travel allowance for their immediate family. So, here we have two entity sets: employee and family, related by "Can claim for". However, family doesn't have a super key. Existence of a family is entirely dependent on the concerned employee. So, it is meaningful only with reference to employee.

**Entity Generalization and Specialization**

Once we have identified the entity sets, we might find some similarities among them. For example, multiple person interacts with a banking system. Most of them are customers, and rest employees or other service providers. Here, customers, employees are persons, but with certain specializations. Or in other way, person is the generalized form of customer and employee entity sets.

ER model uses the "ISA" hierarchy to depict specialization (and thus, generalization).

**Mapping Cardinalities**

One of the main tasks of ER modeling is to associate different entity sets. Let's consider two entity sets E1 and E2 associated by a relationship set R. Based on the number of entities in E1 and E2 are associated with, we can have the following four type of mappings:

* One to one: An entity in E1 is related to at most a single entity in E2, and vice versa
* One to many: An entity in E1 could be related to zero or more entities in E2. Any entity in E2 could be related to at most a single entity in E1.
* Many to one: Zero or more number of entities in E1 could be associated to a single entity in E2. However, an entity in E2 could be related to at most one entity in E1.
* Many to many: Any number of entities could be related to any number of entities in E2, including zero, and vice versa

**ER Diagram**

From a given problem statement we identify the possible entity sets, their attributes, and relationships among different entity sets. Once we have these information, we represent them pictorially, called an entity-relationship (ER) diagram.

**Graphical Notations for ER Diagram:**

| **Term** | **Notation** | **Remarks** |
| --- | --- | --- |
| **Entity set** | **IMG_256** | **Name of the set is written inside the rectangle** |
| **Attribute** | **IMG_257** | **Name of the attribute is written inside the ellipse** |
| **Entity with attributes** | **IMG_258** | **Roll is the primary key; denoted with an underline** |
| **Weak entity set** | **IMG_259** |  |
| **Relationship set** | **IMG_260** | **Name of the relationship is written inside the diamond** |
| **Related enity sets** | **IMG_261** |  |
| **Relationship cardinality** | **IMG_262** | **A person can own zero or more cars but no two persons can own the same car** |
| **Relationship with weak entity set** | **IMG_263** |  |

**Importance of ER modeling**

Figure - 01 shows the different steps involved in implementation of a (relational) database.

IMG_264

Figure - 01: Steps to implement a RDBMS

Given a problem statement, the first step is to identify the entities, attributes and relationships. We represent them using an ER diagram. Using this ER diagram, table structures are created, along with required constraints. Finally, these tables are normalized in order to remove redundancy and maintain data integrity. Thus, to have data stored efficiently, the ER diagram is to be drawn as much detailed and accurate as possible.

**Case Study:**

Draw ER diagram for the Library Management System.

**Sequence diagram**

It represents the behavioral aspects of a system. Sequence diagram shows the interactions between the objects by means of passing messages from one object to another with respect to time in a system.

**Elements in sequence diagram**

Sequence diagram contains the objects of a system and their life-line bar and the messages passing between them.

**Object**

Objects appear at the top portion of sequence diagram. Object is shown in a rectangle box. Name of object precedes a colon ‘:’ and the class name, from which the object is instantiated. The whole string is underlined and appears in a rectangle box. Also, we may use only class name or only instance name.

Objects which are created at the time of execution of use case and are involved in message passing , are appear in diagram, at the point of their creation.

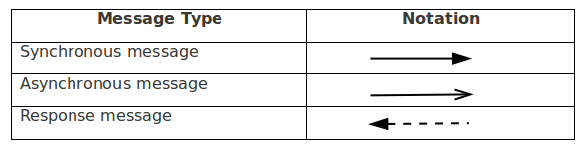
**Life-line bar**

A down-ward vertical line from object-box is shown as the life-line of the object. A rectangle bar on life-line indicates that it is active at that point of time.

**Messages**

Messages are shown as an arrow from the life-line of sender object to the life-line of receiver object and labeled with the message name. Chronological order of the messages passing throughout the objects’ life-line show the sequence in which they occur. There may exist some different types of messages :

* **Synchronous messages**:Receiver start processing the message after receiving it and sender needs to wait until it is made. A straight arrow with close and fill arrow-head from sender life-line bar to receiver end, represent a synchronous message.
* **Asynchronous messages**:For asynchronous message sender needs not to wait for the receiver to process the message. A function call that creates thread can be represented as an asynchronous message in sequence diagram. A straight arrow with open arrow-head from sender life-line bar to receiver end, represent an asynchronous message.
* **Return message:**For a function call when we need to return a value to the object, from which it was called, then we use return message. But, it is optional, and we are using it when we are going to model our system in much detail. A dashed arrow with open arrow-head from sender life-line bar to receiver end, represent that message.
* **Response message:**One object can send a message to self. We use this message when we need to show the interaction between the same object.



**Conclusion**: Thus we have studied, Modeling UML Class Diagrams and Sequence diagrams