ER Diagram

The ER model defines the conceptual view of a database. It works around real-world entities and the associations among them. At view level, the ER model is considered a good option for designing databases.

**Entity**

An entity can be a real-world object, either animate or inanimate, that can be easily identifiable. For example, in a school database, students, teachers, classes, and courses offered can be considered as entities. All these entities have some attributes or properties that give them their identity.

An entity set is a collection of similar types of entities. An entity set may contain entities with attribute sharing similar values. For example, a Students set may contain all the students of a school; likewise a Teachers set may contain all the teachers of a school from all faculties. Entity sets need not be disjoint.

**Attributes**

Entities are represented by means of their properties, called **attributes**. All attributes have values. For example, a student entity may have name, class, and age as attributes.

There exists a domain or range of values that can be assigned to attributes. For example, a student's name cannot be a numeric value. It has to be alphabetic. A student's age cannot be negative, etc.

**Types of Attributes**

* **Simple attribute** − Simple attributes are atomic values, which cannot be divided further. For example, a student's phone number is an atomic value of 10 digits.
* **Composite attribute** − Composite attributes are made of more than one simple attribute. For example, a student's complete name may have first\_name and last\_name.
* **Derived attribute** − Derived attributes are the attributes that do not exist in the physical database, but their values are derived from other attributes present in the database. For example, average\_salary in a department should not be saved directly in the database, instead it can be derived. For another example, age can be derived from data\_of\_birth.
* **Single-value attribute** − Single-value attributes contain single value. For example − Social\_Security\_Number.
* **Multi-value attribute** − Multi-value attributes may contain more than one values. For example, a person can have more than one phone number, email\_address, etc.

A **data flow diagram** (**DFD**) is a graphical representation of the "flow" of data through an [information system](https://en.wikipedia.org/wiki/Information_system), modelling its *process* aspects. A DFD is often used as a preliminary step to create an overview of the system, which can later be elaborated.[[2]](https://en.wikipedia.org/wiki/Data_flow_diagram#cite_note-2) DFDs can also be used for the [visualization](https://en.wikipedia.org/wiki/Data_visualization) of [data processing](https://en.wikipedia.org/wiki/Data_processing) (structured design).

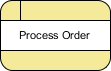
A DFD shows what kind of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of process or information about whether processes will operate in sequence or in parallel (which is shown on a [flowchart](https://en.wikipedia.org/wiki/Flowchart)).The notations used in DFD are:

**External Entity**

An external entity can represent a human, system or subsystem. It is where certain data comes from or goes to. It is external to the system we study, in terms of the business process. For this reason, people used to draw external entities on the edge of a diagram.

notation (enternal entity)

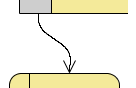
**Process**

A process is a business activity or function where the manipulation and transformation of data takes place. A process can be decomposed to finer level of details, for representing how data is being processed within the process.   


**Data Store**

A data store represents the storage of persistent data required and/or produced by the process. Here are some examples of data stores: membership forms, database table, etc.   
notation (data store)

**Data Flow**

A data flow represents the flow of information, with its direction represented by an arrow head that shows at the end(s) of flow connector.   


Activity diagram is basically a flow chart to represent the flow form one activity to another activity. The activity can be described as an operation of the system.

So the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent. Activity diagrams deals with all type of flow control by using different elements like fork, join etc.

## Purpose:

The basic purposes of activity diagrams are similar to other four diagrams. It captures the dynamic behaviour of the system. Other four diagrams are used to show the message flow from one object to another but activity diagram is used to show message flow from one activity to another.

Activity is a particular operation of the system. Activity diagrams are not only used for visualizing dynamic nature of a system but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in activity diagram is the message part.

Following are the main usages of activity diagram:

* Modeling work flow by using activities.
* Modeling business requirements.
* High level understanding of the system's functionalities.
* Investigate business requirements at a later stage.

In UML there are five diagrams available to model dynamic nature and use case diagram is one of them. Now as we have to discuss that the use case diagram is dynamic in nature there should be some internal or external factors for making the interaction.

These internal and external agents are known as actors. So use case diagrams are consists of actors, use cases and their relationships. The diagram is used to model the system/subsystem of an application. A single use case diagram captures a particular functionality of a system.

So to model the entire system numbers of use case diagrams are used.

## Purpose

The purpose of use case diagram is to capture the dynamic aspect of a system. But this definition is too generic to describe the purpose.

Because other four diagrams (activity, sequence, collaboration and Statechart) are also having the same purpose. So we will look into some specific purpose which will distinguish it from other four diagrams.

Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. So when a system is analyzed to gather its functionalities use cases are prepared and actors are identified.

Now when the initial task is complete use case diagrams are modelled to present the outside view.

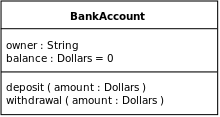
So in brief, the purposes of use case diagrams can be as follows:

* Used to gather requirements of a system.
* Used to get an outside view of a system.
* Identify external and internal factors influencing the system.
* Show the interacting among the requirements are actors.

**Class Diagram**

In [software engineering](https://en.wikipedia.org/wiki/Software_engineering), a **class diagram** in the [Unified Modeling Language](https://en.wikipedia.org/wiki/Unified_Modeling_Language) (UML) is a type of static structure diagram that describes the structure of a system by showing the system's [classes](https://en.wikipedia.org/wiki/Class_%28computer_science%29), their attributes, operations (or methods), and the relationships among objects.

The class diagram is the main building block of [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming) modelling. It is used both for general [conceptual modelling](https://en.wikipedia.org/wiki/Conceptual_model) of the systematics of the application, and for detailed modelling translating the models into [programming code](https://en.wikipedia.org/wiki/Programming_code). Class diagrams can also be used for [data modeling](https://en.wikipedia.org/wiki/Data_modeling).[[1]](https://en.wikipedia.org/wiki/Class_diagram#cite_note-1) The classes in a class diagram represent both the main elements, interactions in the application, and the classes to be programmed.

[](https://en.wikipedia.org/wiki/File:BankAccount1.svg)

A class with three compartments.

In the diagram, classes are represented with boxes that contain three compartments:

* The top compartment contains the name of the class. It is printed in bold and centered, and the first letter is capitalized.
* The middle compartment contains the attributes of the class. They are left-aligned and the first letter is lowercase.
* The bottom compartment contains the operations the class can execute. They are also left-aligned and the first letter is lowercase.

In the design of a system, a number of classes are identified and grouped together in a class diagram that helps to determine the static relations between them. With detailed modelling, the classes of the conceptual design are often split into a number of subclasses.

In order to further describe the behaviour of systems, these class diagrams can be complemented by a [state diagram](https://en.wikipedia.org/wiki/State_diagram) or [UML state machine](https://en.wikipedia.org/wiki/UML_state_machine)]

**2. Data Flow Diagram:**

Level 0:

sent to leads to

Person violating the rule

Person paying the fine.

Person violating the rule

Person paying the fine.

Level 1:

Sent to extract no extract details leads to

Level 2:

Person violating

the rule

Person paying fine

Sent to

Image processing

Vehicle number

Vehicle vehicle details Personal info

Personal

Record

Fine

Fine amount

Leads to challan

generation

**4. Use-case diagram**:

Vehicle Owner

System

System manager

**5.Class diagram**:

**VEHICLE**

Number

Enigine\_number

Type

Model\_number

Register ();

**OWNER**

Name

Contact\_number

Permanent\_Address

Local\_address

UID

Pay\_fine ();

Violate\_rule ();

**SYSTEM**

Ward\_number

Authentication ();

Update\_information ();

Collect\_fine ();

Register\_complainces ();

Generate\_reports ();

**SYSTEM\_MANAGER**

Manager\_id

Assigned\_wards

Analyze\_reports ();

Analyze\_complaints ();

DRIVING

RULE VIOLATED?

NO

YES

COUNT++

CAPTUREIMAGE

EXTRACT NO.

SEARCH

VEHICLE

OWNER

FINE

ACTIONS

MESSAGE & EMAIL ALERT

CHALLAN GENERATION

CHALLAN DELIVERY

FINE PAID

DAYS++

DAYS=30?

COUNT=3?

CANCEL LICENCE

YES

NO

NO

YES

YES

NO

**3**. **Activity diagram:**

**Diagrams for Follow or Pay System.**

1. **ER Diagram**

ER Diagram:

OWNER NO.

VEHICLE NO.

ENGINE NO.

WARD NO.

OFFICE NO.

**OFFICES**

**OFFICERS**

HAVE

OFFICE NO.

OFFICER NO.

OFF\_NAME

PAYS AT

UID NO.

PAN NO.

NAME

ADDRESS

VEHICLE NO.

TYPE

WARD NO.

**OWNER**

**FINE**

OWNS

RULE

VEHICLE NO.

POLICE STATION

LAST SEEN AT

DATE

**THEFT RECORD**

**VEHICLE**

OWNER NO.

TYPE

RULE NO.

FOLLOW OR VIOLATE

HAS