Requirement Engineering

What is Software Requirement?

- **Definition:**
- According IEEE Standard, Software Requirement is a condition needed by a user to solve a problem or achieve an objective.
- Requirement should be clear, correct and well-defined.
- **Requirements engineering (RE)** refers to the process of defining, documenting, and maintaining requirements in the engineering design process. Requirement engineering provides the appropriate mechanism to understand what the customer desires, analyzing the need, and assessing feasibility, negotiating a reasonable solution, specifying the solution clearly, validating the specifications and managing the requirements as they are transformed into a working system.
- Thus, requirement engineering is the disciplined application of proven principles, methods, tools, and notation to describe a proposed system's intended behavior and its associated constraints.

Types of Requirements

There are two types of requirements:

1. Functional Requirements:

2. Non-Functional Requirements

Functional Requirements

- Functional requirements define a function that a system or system element must be qualified to perform and must be documented in different forms. The functional requirements describe the behavior of the system as it correlates to the system's functionality.
- Functional requirements should be written in a simple language, so that it is easily understandable. The examples of functional requirements are authentication, business rules, audit tracking, certification requirements, transaction corrections, etc.
- These requirements allow us to verify whether the application provides all functionalities mentioned in the application's functional requirements. They support tasks, activities, user goals for easier project management.
- There are a number of ways to prepare functional requirements. The most common way is that they are documented in the text form. Other formats of preparing the functional requirements are use cases, models, prototypes, user stories, and diagrams.

Non-functional requirements

- Non-functional requirements are not related to the software's functional aspect. They can be the necessities that specify the criteria that can be used to decide the operation instead of specific behaviors of the system. Basic non-functional requirements are - usability, reliability, security, storage, cost, flexibility, configuration, performance, legal or regulatory requirements, etc.
- They are divided into two main categories:
- **Execution qualities** like security and usability, which are observable at run time.
- **Evolution qualities** like testability, maintainability, extensibility, and scalability that embodied in the static structure of the software system.
- Non-functional requirements specify the software's quality attribute. These requirements define the general characteristics, behavior of the system, and features that affect the experience of the user. They ensure a better user experience, minimizes the cost factor. Non-functional requirements ensure that the software system must follow the legal and adherence rules. The impact of the non-functional requirements is not on the functionality of the system, but they impact how it will perform. For a well-performing product, atleast some of the non-functional requirements should be met.

Functional requirements v/s Non-functional requirements

Functional Requirements	Non-functional requirements
Functional requirements help to understand the functions of the system.	They help to understand the system's performance.
Functional requirements are mandatory.	While non-functional requirements are not mandatory.
They are easy to define.	They are hard to define.
They describe what the product does.	They describe the working of product.
It concentrates on the user's requirement.	It concentrates on the expectation and experience of the user.
It helps us to verify the software's functionality.	It helps us to verify the software's performance.
These requirements are specified by the user.	These requirements are specified by the software developers, architects, and technical persons.
There is functional testing such as API testing, system, integration, etc.	There is non-functional testing such as usability, performance, stress, security, etc.
Examples of the functional requirements are - Authentication of a user on trying to log in to the system.	Examples of the non-functional requirements are - The background color of the screens should be light blue.
These requirements are important to system operation.	These are not always the important requirements, they may be desirable.
Completion of Functional requirements allows the system to perform, irrespective of meeting the non-functional requirements.	While system will not work only with non-functional requirements.

Examples

➤Examples of Functional Requirements:

1. In hospital management system, a doctor should be able to retrieve the information of his patients.
2. Search option given to user to search from various invoices.
3. User should be able to mail any report to management.
4. Only Managerial level employees have the right to view banking revenue data.

➤Examples of Non-Functional Requirements:

1. Security, Logging, Storage, Configuration, Performance, Cost, Interoperability, Flexibility, Disaster recovery, Accessibility
2. Employees never allowed to update their salary information. Reported to the administrator.
3. A website should be capable enough to handle 20 million users with affecting its performance.
4. The software should be portable. Moving from one OS to other OS does not create any issu

What is Requirement Engineering?

The process to gather the software requirements from customers, analyze and document them is known as requirement engineering.

Requirement engineering provides the appropriate mechanism to understand:

- Customer desires / wishes.
- Analyzing Customer need
- Assessing feasibility
- Negotiating a reasonable solution
- Specifying the solution clearly
- Validating the specifications and managing the requirements

as they are transformed into a working system

Users VS System Requirements

No.	Users Requirements	System Requirements
1.	Written for Customers.	Written for Implementation Team.
2.	In Natural language.	In Technical Language.
3.	Describe the services & features provide by system.	Describe the detail description of services, features & complete operations of system.
4.	May include diagrams & tables.	May include system models.
5.	Understandable by system users who don't have technical knowledge.	Understandable by implementation team who have technical knowledge.
6.	For Client Managers, System Users, Contractor Managers, System Architects	For System Architects, Software Developers. Client Engineers, System Users

Tools in Requirement Engineering

1. Observation report
2. Questionnaire (survey, poll)
3. Use cases
4. User stories
5. Requirement workshop
6. Mind mapping
7. Role playing 8. Prototyping

Available Softwares:

1. Xebrio Requirements Management Software
2. Jira for requirements engineering
3. Doc Sheets
4. RequirementsHub
5. IBM Engineering Requirements Management DOORS
6. Modern Requirements
7. SpiraTeam

Establishing Ground Work

1. Identifying the stakeholders:

- Any person who benefits directly or indirectly from the system being developed is a stakeholder.
- **Examples**: Business operations managers, product managers, marketing people, internal and external customers, end-users, consultants, product engineers, software engineers, developers, testers and support/maintenance engineers are the usual stakeholders.
- Each stakeholder sees the system differently, gains different benefits when the system is successfully developed.
- They also faces different risks if the development effort fails.

Establishing Ground Work

2. Recognizing Multiple Viewpoints:

- Because there are so many different stakeholders, the system's requirements will be examined from various perspectives.
- Each of these stakeholders will contribute data to the requirements engineering process.
- As information is gathered from multiple viewpoints, collecting requirements may be inconsistent or contradictory.
- You should categories all stakeholder information so that decision-makers can select an consistent set of system requirements for the system.

Establishing Ground Work

3. Working toward Collaboration:

- If there are five stakeholders involved in a software project, there may be five different opinions on the set of requirements.
- Customers must work together with software development team to create a successful system.
- A requirements engineer's job is to identify areas of commonality requirement as well as areas of conflict requirements.
- At the end project head & requirement engineer may decide on which requirements are accepted.

Establishing Ground Work

4. Asking the first questions: (For identifying stakeholders)

The first set of questions are focuses on the customers, stakeholders as well as the overall project goals and benefits.

1. Who is the person or organization behind the request for this work?
2. Who will make use of the solution?
3. What is the economic value of a successful solution?
4. Is there a different source for the solution you require?

Establishing Ground Work

4. Asking the first questions: (Understanding Problems & Solution)

• Following set of questions helps in gaining a better understanding of the problem and allows the customer to express his or her thoughts on a solution:

1. How would you characterize "good" output produced by a successful solution?

2. To what problem(s) will this solution be applied?

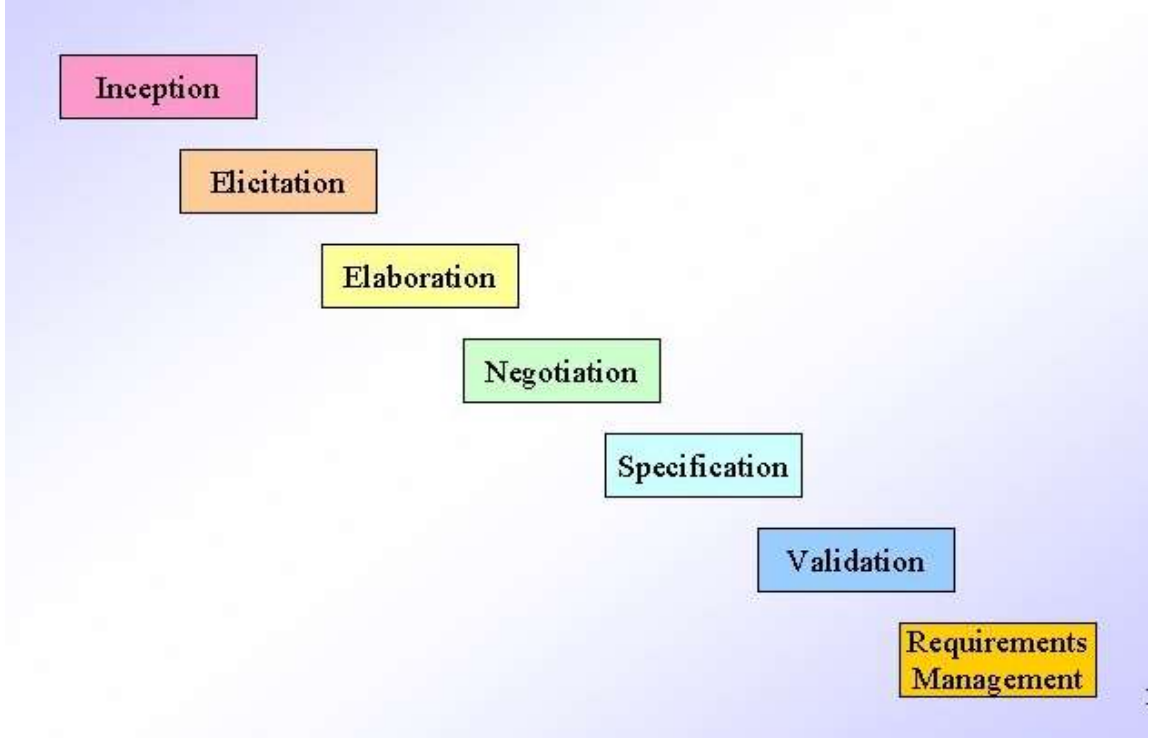
3. Can you describe (or show me) the business environment in which the solution will be used?

4. Will there be any special performance issues or constraints that will influence how the solution is approached?

Requirement Engineering Tasks

- The process of collecting the software requirements from the client and then understanding. analysis, and documenting it is called requirement engineering.
- Requirement engineering constructs a bridge for design and construction.
- Requirement Engineering consists of seven different tasks:

1. Inception
2. Elicitation
3. Elaboration
4. Negotiation
5. Specification
6. Validation
7. Management



1. Inception

- This is the first phase of the requirements analysis process or starting of project.
- Requirement engineering asks a set of questions to establish a software process.
    1. They understand basic details, aim, goal of the project & find out the solution.
    2. They identify stakeholders & who want the solution.
    3. They understand nature of solution.
    4. Enhance collaboration between customer & developer.

2. Elicitation

- This phase focuses on gathering the requirements from the stakeholders.
- The right people must be involved in this phase, no space for mistake.

➤The following problems can occur in the elicitation phase:

1. Problem of scope: The customer give the unnecessary technical details rather than clarity of the overall system objective. Not possible to implement details have given.

2. **Problem of Understanding:** Not having a clear understanding between the developer and customer. Sometimes the customer might not know what they want or the developer might misunderstand one requirement for another.

3. **Problem of Volatility:** Requirements changing over time can cause difficulty in developing project. It can lead to loss and wastage of resources and time.

3. Elaboration & 4. Negotiation

3. Elaboration:

- It takes the requirements that have been gathered in the first two Inception & Elicitation phases and expand them.
- Its main task is developing model & prototype of software using functions, feature and constraints of a software using different tools.

4. Negotiation:

- Negotiation is between the developer and the customer about limited availability of recourses. delivery time, project cost & overall estimation of project.
- Customers are asked to prioritize the requirements for development & also analyze any conflict has occur.

5. Specification & 6. Validation

5. Specification:

- In this task, the requirement engineer constructs a final work product.
- The work product is in the form of software requirement specification document.
- It collect all user, system, functional & non-functional requirements.
- ER. DFD, Data dictionary model used in this phase.
- A software specification document is submitted to the customer in a language that he/she will understand, to give a glimpse of the working model.

6. Validation:

- It checking for errors and debugging in final work product or SRS document.
- It check all the requirements have been stated and met correctly as per stakeholders.
- It check any missing information or want to add any additional information.

7. Requirement Management

- Requirements management is a set of activities where the entire team takes part in identifying, controlling, tracking, and establishing the requirements for the successful and smooth implementation of the project.
- It track on new additional requirement implementation which does not affect on overall system.
- Based on this phase, the working model will be analyzed carefully and ready to be delivered to the customer.

Summary

- Requirement Engineering consists of seven different tasks:

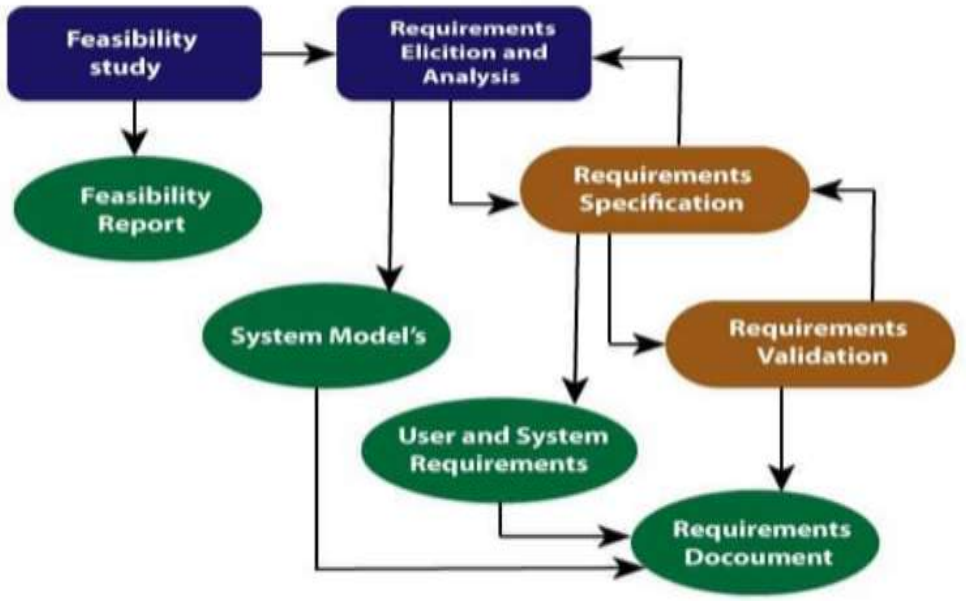
1. **Inception** (Better Understanding)
2. **Elicitation** (Clear Understanding of Requirements)
3. **Elaboration** (Refinement & Modelling)
4. **Negotiation** (Settlement & Conflicts)
5. **Specification** (SRS Document)
6. **Validation** (Validates Requirements)
7. **Management** (Manage changing requirement)

Requirement Engineering Process

- The process to gather the software requirements from customers, analyze and document them is known as requirement engineering.
- ➤Following four steps involve in Requirement Engineering Process:

Feasibility Study

- 1. Requirement Elicitation and Analysis
- 2. Software Requirement Specification
- 3. Software Requirement Validation
- 4. Software Requirement Management



Requirement Engineering Process

Feasibility Study

- To create & analyze the reasons for developing the software that is acceptable to users.
- To check aim & goal of the customer & organization behind implementation of software.

Types of Feasibility Study:

1. Technical Feasibility: Evaluates the current technologies, which are needed to achieve customer requirements within the time and budget.

2. Operational Feasibility: Required software solve business problems and customer requirements.

3. Economic Feasibility: It decides whether the necessary software can generate financial profits for an organization.

The output is to decide whether or not the project should be undertaken

Step 1: Requirement Elicitation & Analysis

- Requirements elicitation is the process of gathering actual requirements about the needs and expectations of stakeholders for a software system.
- To understand customers, business manuals, the existing software of same type, standards and other stakeholders of the project.

Techniques of Requirements Elicitation:

1. Interviews: One-on-one conversations with stakeholders to gather information about their needs and expectations.

2. Surveys: These are questionnaires that are distributed to stakeholders to gather information.

3. Focus Groups: Small groups of stakeholders who are brought together to discuss their needs.

4. Observation: Observing the stakeholders in their work environment to gather information.

5. Prototyping: Creating a working model of the software system, which can be used to gather feedback from stakeholders and to validate requirements.

Step 2: Software Requirement Specification

- Requirements received from client are written in natural language.
- After collecting, System analyst to document the requirements in technical language.
- SRS include Data Flow Diagram. Data Dictionary, Entity Relationship diagram.

➤**SRS defines**

- ❑How the intended software will interact with hardware.
- ❑External interface / GUI
- ❑Speed of operation, Response time of system
- ❑Portability of software across various platforms
- ❑Maintainability
- ❑Speed of recovery after crashing
- ❑Security, Quality. Limitations etc.

Step 3: Software Requirement Validation

- After SRS developed, the requirements discussed in this document are validated or tested
- Requirement validation done through Requirement reviews taken by customers. After developing prototyping check with customers & Test case generations.
- **Requirements can be the check against the following conditions -**
  - If they can practically implement.
  - If they are correct and as per the functionality and specially of software.
  - If there are any ambiguities.
  - If they can describe
  - Any changing or additional requirement

Step 4: Software Requirement Management

- Requirement management is the process of managing changing requirements during the requirements engineering process.

➤**Activities involved in Requirement Management:**

1. **Tracking & Controlling Changes:** Handling changing requirements from customers.
2. **Version Control:** Keeping track of different versions of system.
3. **Traceability:** Trace to software is design, develop or test as per requirements or not.
4. **Communication:** If changing requirements has any issues contact with clients within time.
5. **Monitoring & Reporting:** Monitoring development process & report the status.

What is Software Requirement Specification?

- A software requirements specification (SRS) is a document that mentioned complete description about how the system is expected to perform, Behavior of system, Functional & Non Functional requirement of system.
- SRS is a formal report, that enables the customers to review whether it (SRS) is according to their requirements.
- It is usually signed off at the end of requirements engineering phase.
- **Users of SRS:**
- 1. Client
- 2. Development Team
- 3. Maintenance Team
- 4. Technical Writers

Need of SRS Document

1. It structures and formalizes all project requirements.
2. It helps the development team build a product that exactly meets customer and target audience needs.
3. It provides all team members with the necessary information while working on the project.
4. It minimizes the possible misunderstanding between the members of the development team and the customer.
5. It clearly defines the scope of work, and that allows developers to plan particular iterations and release dates.
6. It allows estimating the required development budget.

Structure of SRS

1. Introduction

- 1.1 Purpose
- 1.2 Intended Audience
- 1.3 Scope
- 1.4 Definition
- 1.5 References

2. Overall Description

- 2.1 User Interface
- 2.2 System Interface Structure of SRS
- 2.3 Software & Hardware Requirements
- 2.4 Constraints
- 2.5 User Characteristics

3. System Features & Requirements

- 3.1 Functional Requirements
- 3.2 Use Cases / Sequence Diagrams
- 3.3 External Interface Requirement
- 3.4 Database Requirement
- 3.5 Non-Functional Requirement

4. Deliver for Approval

Characteristics of good SRS

1. **Correctness:** Provide accurate functional & non functional requirements as discussed with client.

2. **Completeness:** Should complete all essential features like functionality, performance, features, design, constraints & external interfaces.

3. **Consistency:** Same abbreviation, tabular format, diagram format, naming format follow.

4. **Unambiguousness:** Should not be any confusion regarding understanding of the requirements. Everything mention uniquely & clearly.

5. **Ranking for importance and stability:** Every requirement is important. But some are urgent and must be fulfilled before other requirements and some could be delayed. It's better to classify each requirement according to its importance and stability.

6. **Modifiability:** It is capable of quickly obtain changes to the system without affecting other.

Characteristics of good SRS

7. Verifiability: The requirements are verified with the help of reviewers & stakeholders.

8. Traceability: Every requirement having unique number that is easy to use in future development.

9. Design Independence: There should be an option to select from multiple design alternatives for the final system. SRS should not contain any implementation details.

10. Testability: An SRS should be written in such a method that it is simple to generate test cases and test plans from the report.

11. Understandable by the customer: The language should be kept simple and clear.

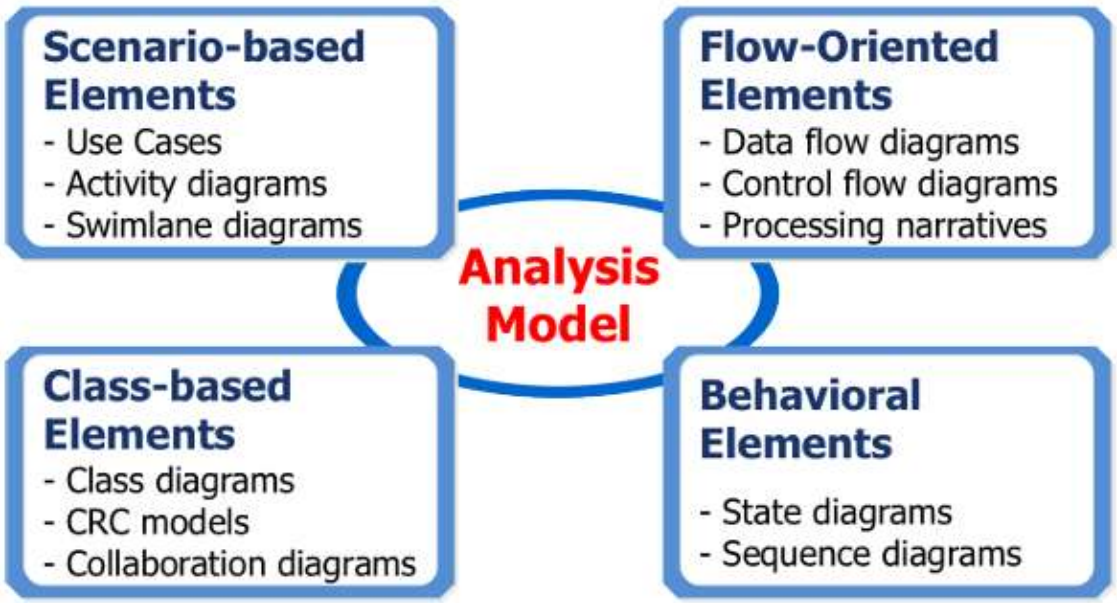
12. The right level of abstraction: If the SRS is written for the requirements stage, the details should be explained explicitly. Whereas, for a feasibility study, fewer analysis can be used. Hence, the level of abstraction modifies according to the objective of the SRS.

Requirement Analysis

- Requirement analysis is significant and essential activity after elicitation.
- This activity reviews all requirements and may provide a graphical view of the entire system.
- After the completion of the analysis, the understandability of the project may improve significantly.

➤ **Requirement Analysis Model / Elements of Requirement Model**

- 1. Scenario based Modelling
- 2. Class based Modelling
- 3. Flow oriented Modelling
- 4. Behavior Modelling



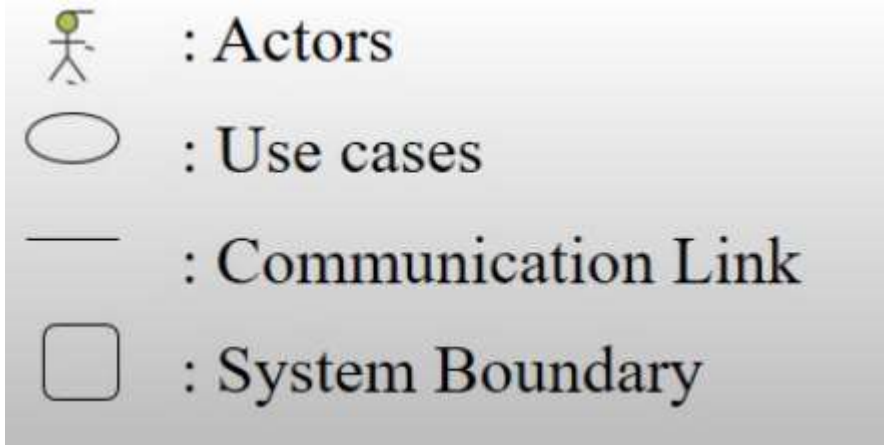
Scenario Based Modelling / Element

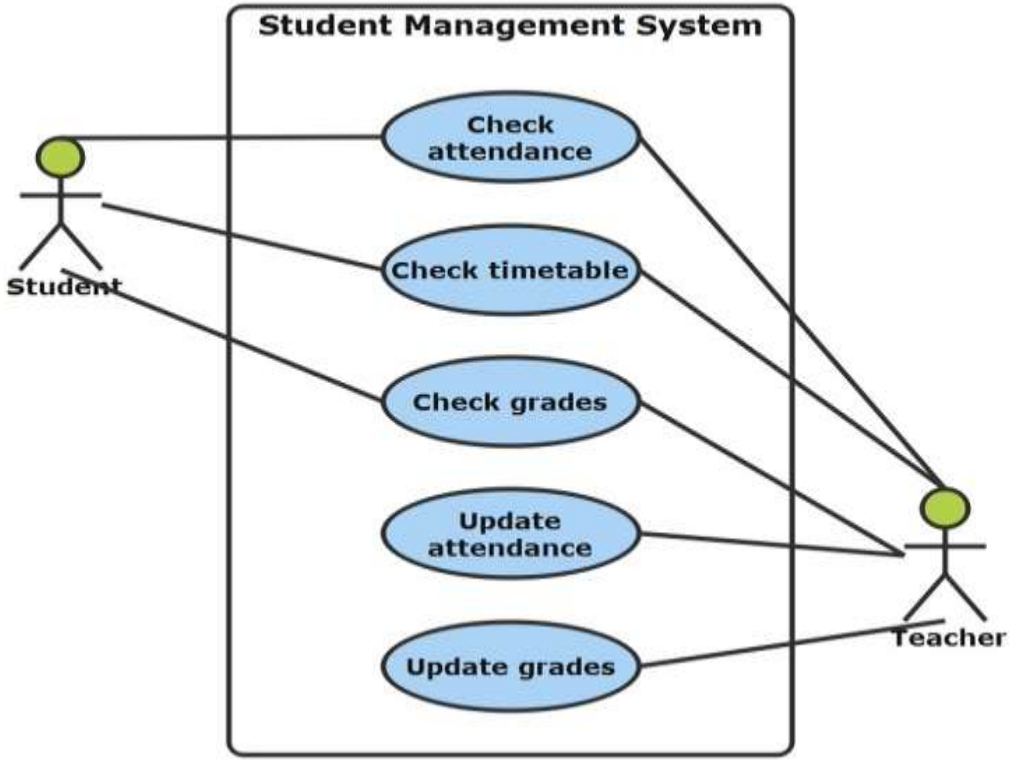
➤For building the design & analysis model it is important for software engineer to understand how end users & other actors want to interact with the system.

➤Here, creation of scenario by using Use case Diagram. Activity Diagram & User Stories.

1. Use case Diagram:

A use case diagram is used to represent the dynamic behavior of a system.

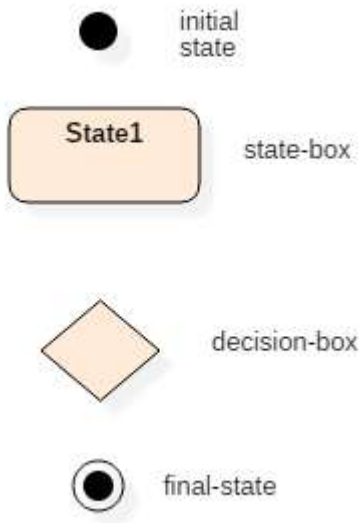


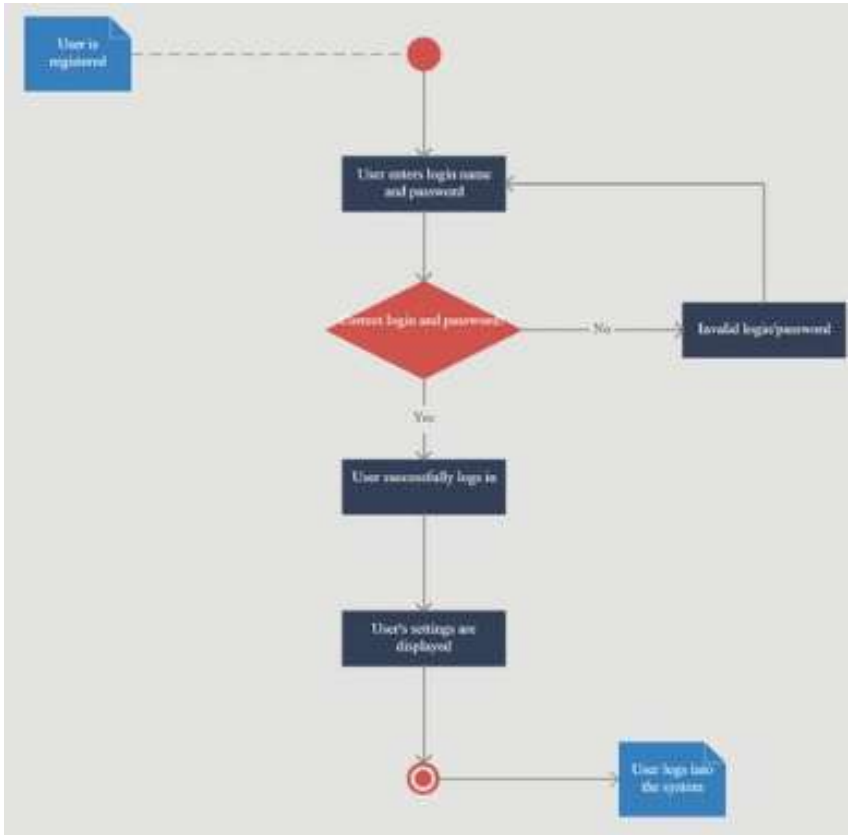


Scenario Based Modelling / Element

2. Activity Diagram:

- The activity diagram helps in predicting the workflow from one activity to another.
- It show condition of flow and the order in which it occurs.



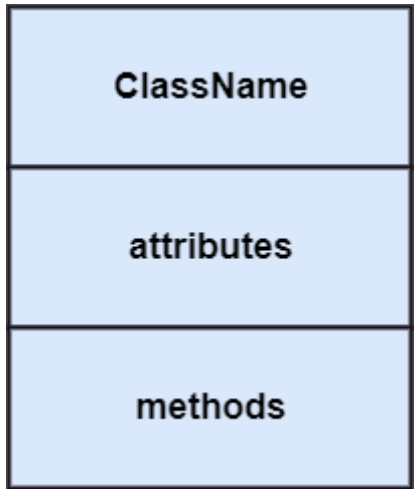


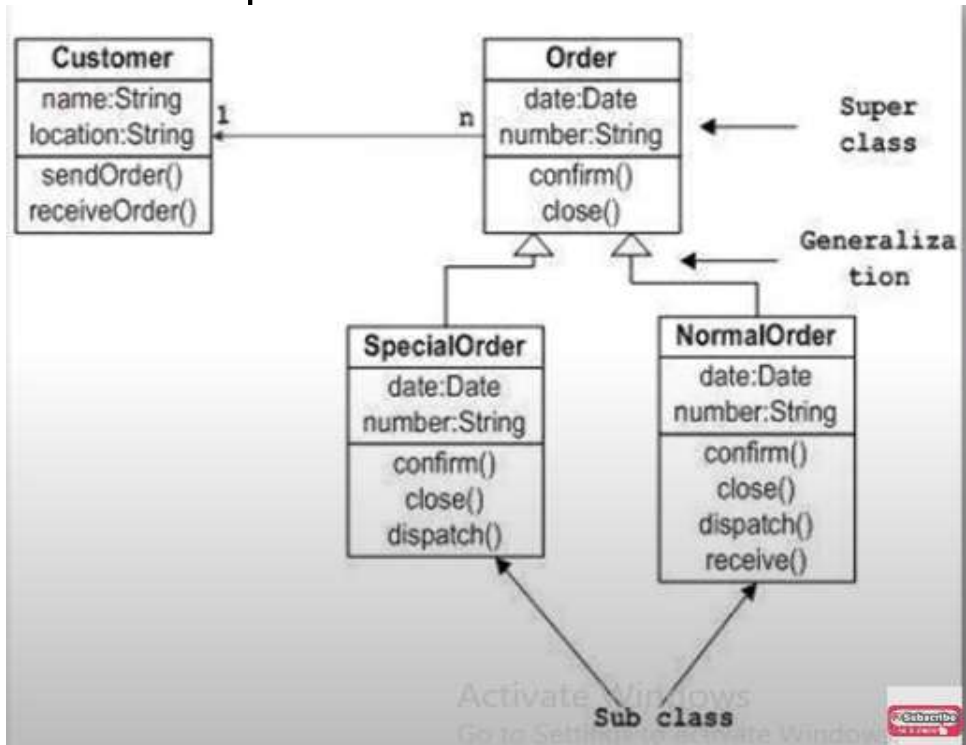
Class Based Modelling / Element

- Class based modeling represent classes, object, attributes & operations of system.

1. Class Diagram:

- The class diagram shows a static view of an application.
- It represents the types of objects residing in the system and the relationships between them.
- It describes the major responsibilities of a system.



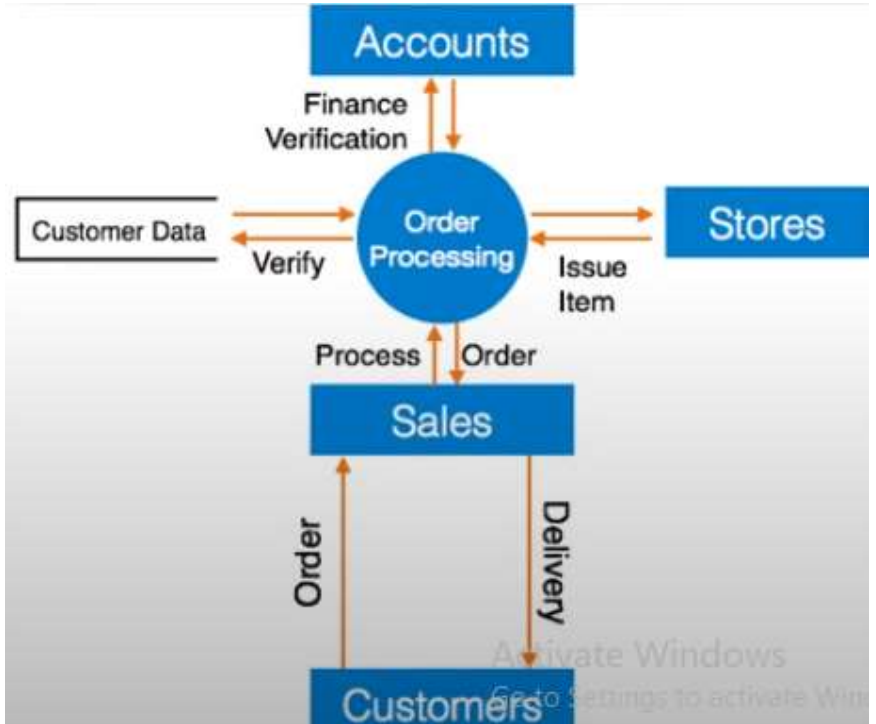


Flow Oriented Modelling / Element

- It shows how data objects are transformed by processing the function.

1. Data Flow Diagram:

- Data flow diagram is graphical representation of flow of data in an information system.
- It is capable of depicting incoming data flow, outgoing data flow and stored data.
- Components of DFD:



Flow Oriented Modelling / Element

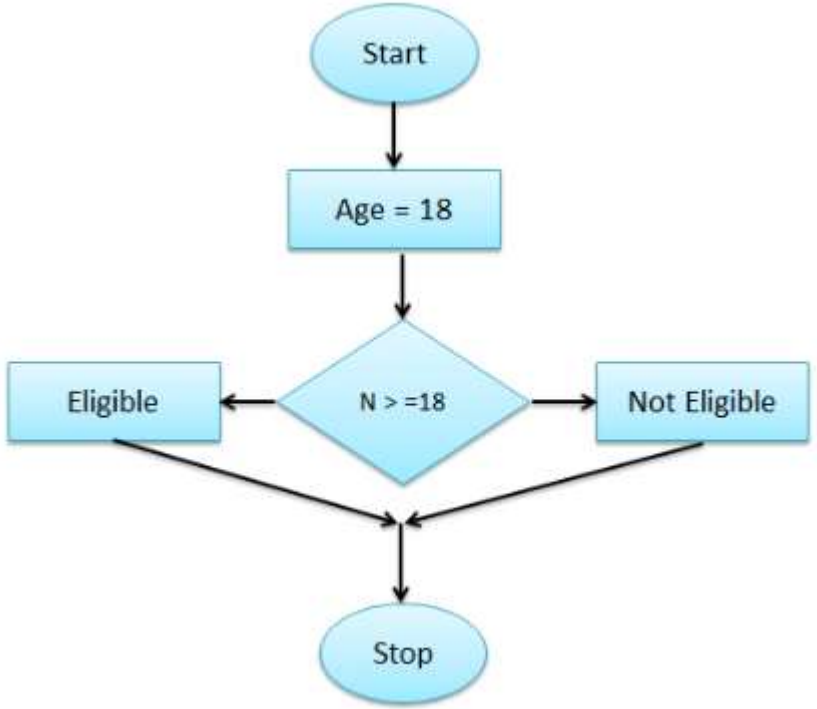
2. Control Flow Diagram:

- It is a graphical representation of control flow or computation during the execution of programs or applications.
- Control flow graphs are mostly used in static analysis as well as compiler applications.
- Accurately represent the flow inside of a program unit.
- Used boolean values are true or false, on or off, I or 0.
- **Entry Block:**

Entry block allows the control to enter into the control flow.

- **Exit Block:**

Control flow leaves through the exit block.



Behavioral Modelling / Element

- Behavioral Model is specially designed to make us understand overall behavior and factors that influence behavior of a System.

1. State Transition Diagram:

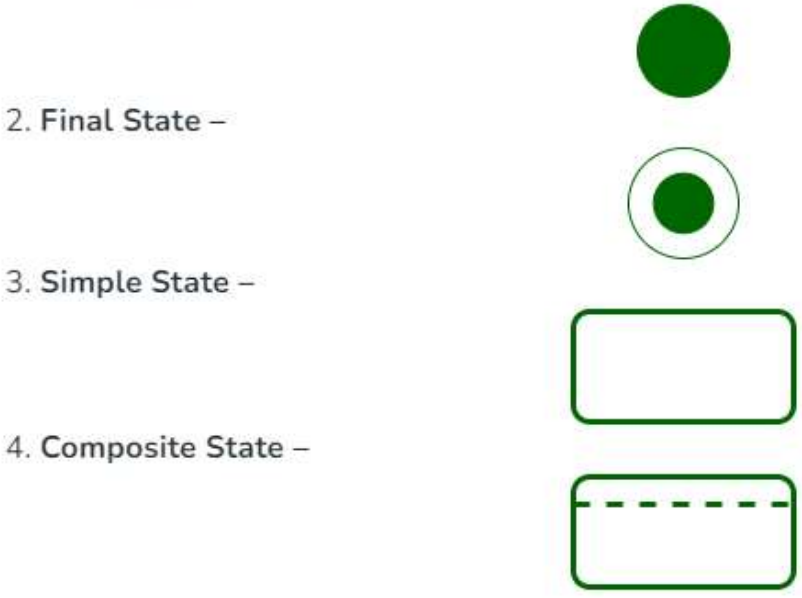
- It usually describes overall states of system.
- And events which are responsible for a change in state of a system.

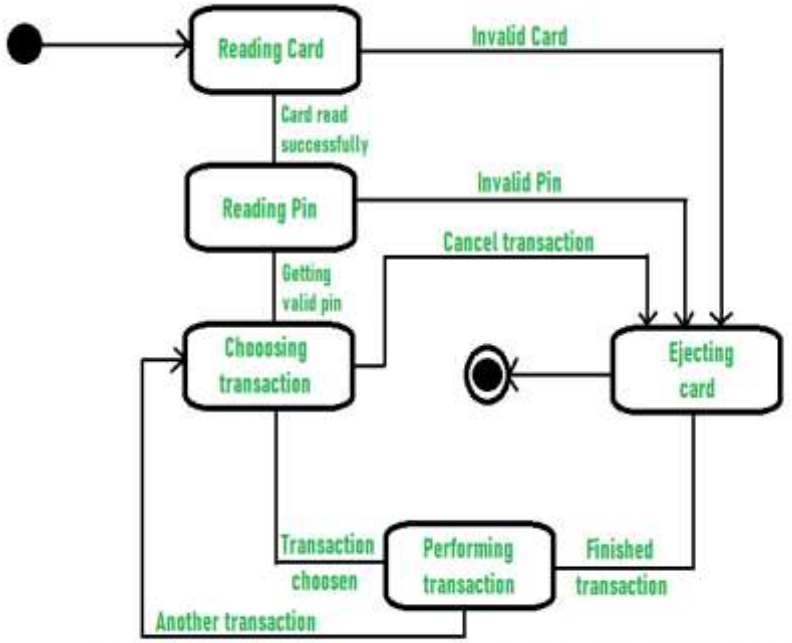
1. Initial State –

2. Final State –

3. Simple State –

4. Composite State –

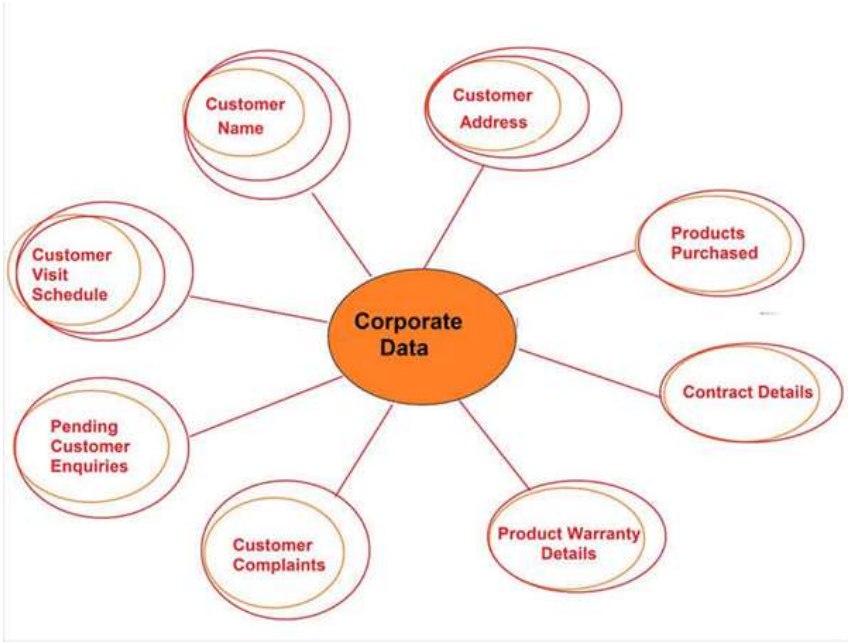


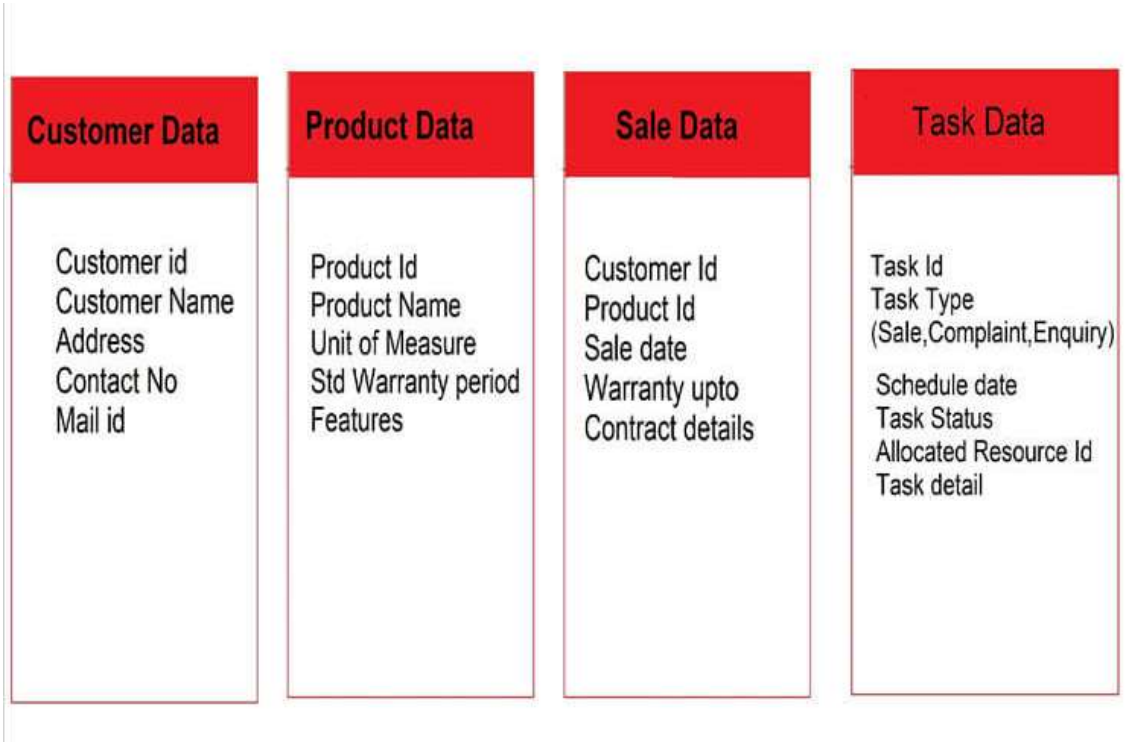


State Transition Diagram for ATM System

What is Data Modeling?

- A data model is an abstract view of the data referred to in the product being developed.
- Data Modeling explaining & visualizing how the data be used by the software and defining data objects that will be stored in a database. Example:

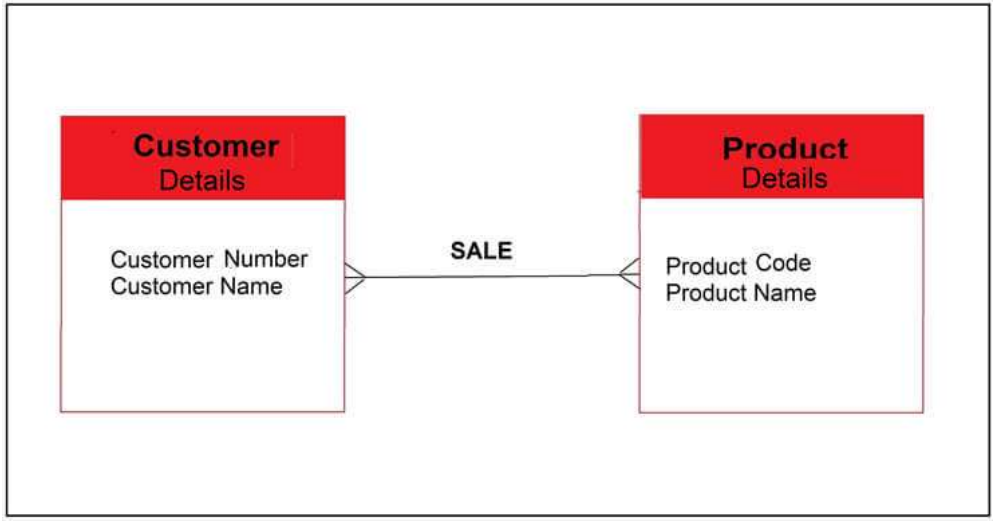




Types of Data Models

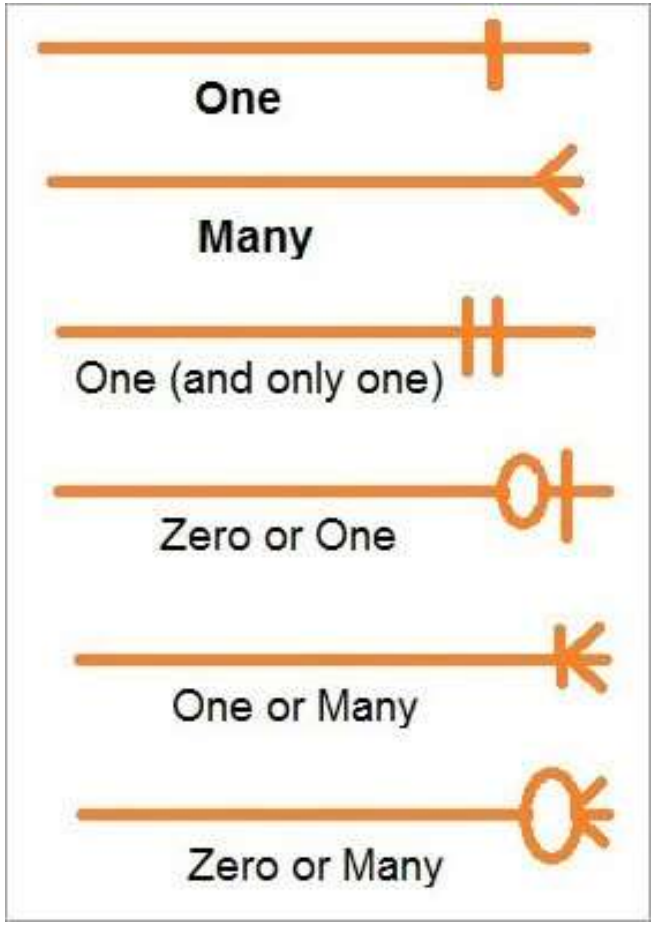
Type 1: Conceptual Data Model

- It gives a front view of the of each data entity and not a technical detail of data.
- It presents all the data entities referred to by the business and their characteristics called attributes and the dependency or the connection between entities called Relationships.
- It makes communication easy and clear as all stakeholders & reducing communication gaps.



Some commonly used Relationship notations are:

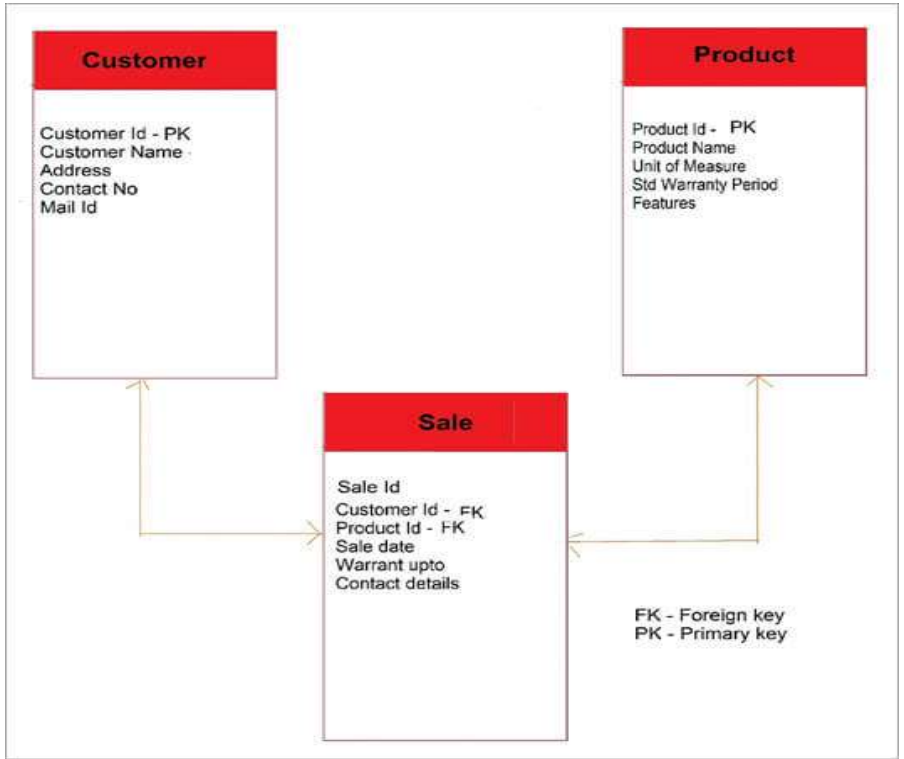
- **One to one:** Employee and his joining details
- **One to many:** Employee and his workplace
- **One and only one:** Employee and his date of birth
- **Zero or one:** Employee and his place of demise
- **Zero or many:** Employee and his subordinates
- **Examples of cardinality:** Relationship Notation



Types of Data Models

Type 2: Logical Data Model

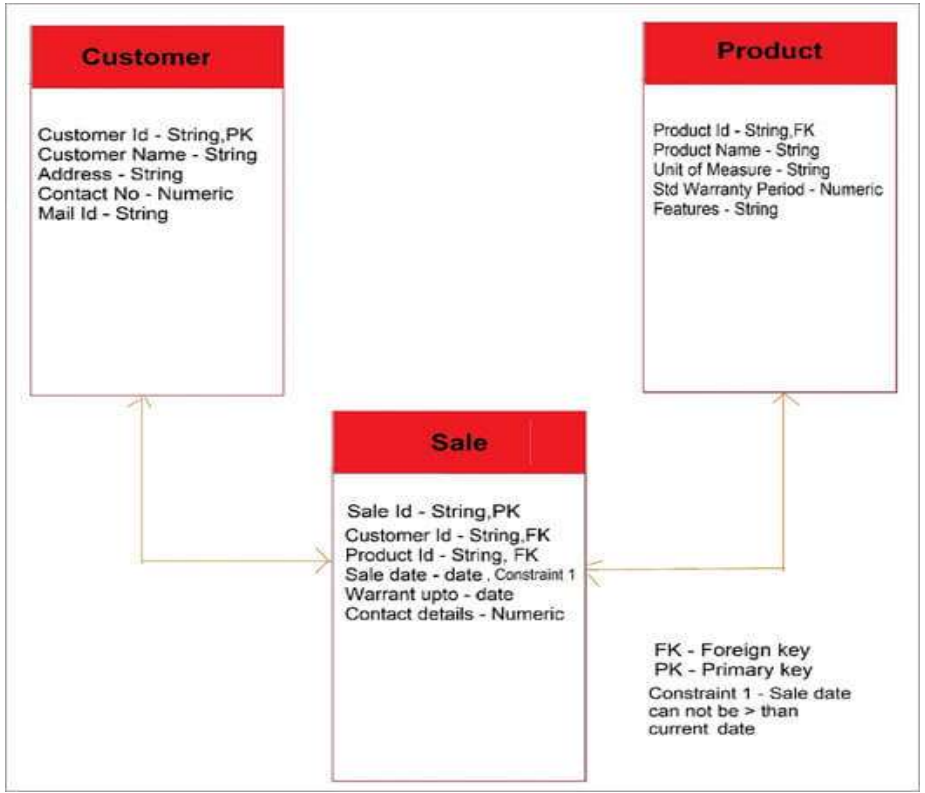
- It gives a detailed description of each data entity their attributes & relationship between two entities giving business purpose to each data.
- The relationship is presented more explicitly with details like Primary key, Foreign key & Parent-child dependent entity type.



Types of Data Models

Type 3: Physical Data Model

- A physical data model is the layout of the actual database with all its components.
- It gives a technical view of the data i.e., the table name, column name, data type, constraints, indexes, primary key, triggers, stored procedures, etc.
- It is developed by using databases SQL, ORACLE etc.



Why Data Modeling?

1. That helps in building the information system & database.

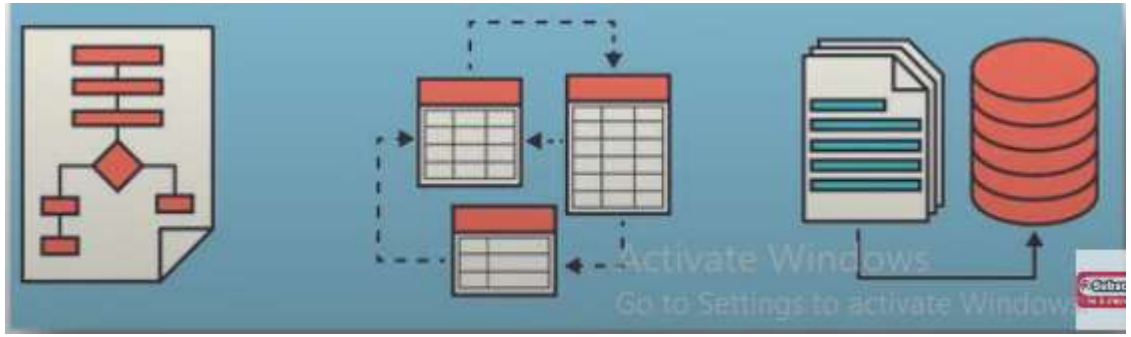
2. Facilitates easy access to the business data.

3. Helps management to take decisions.

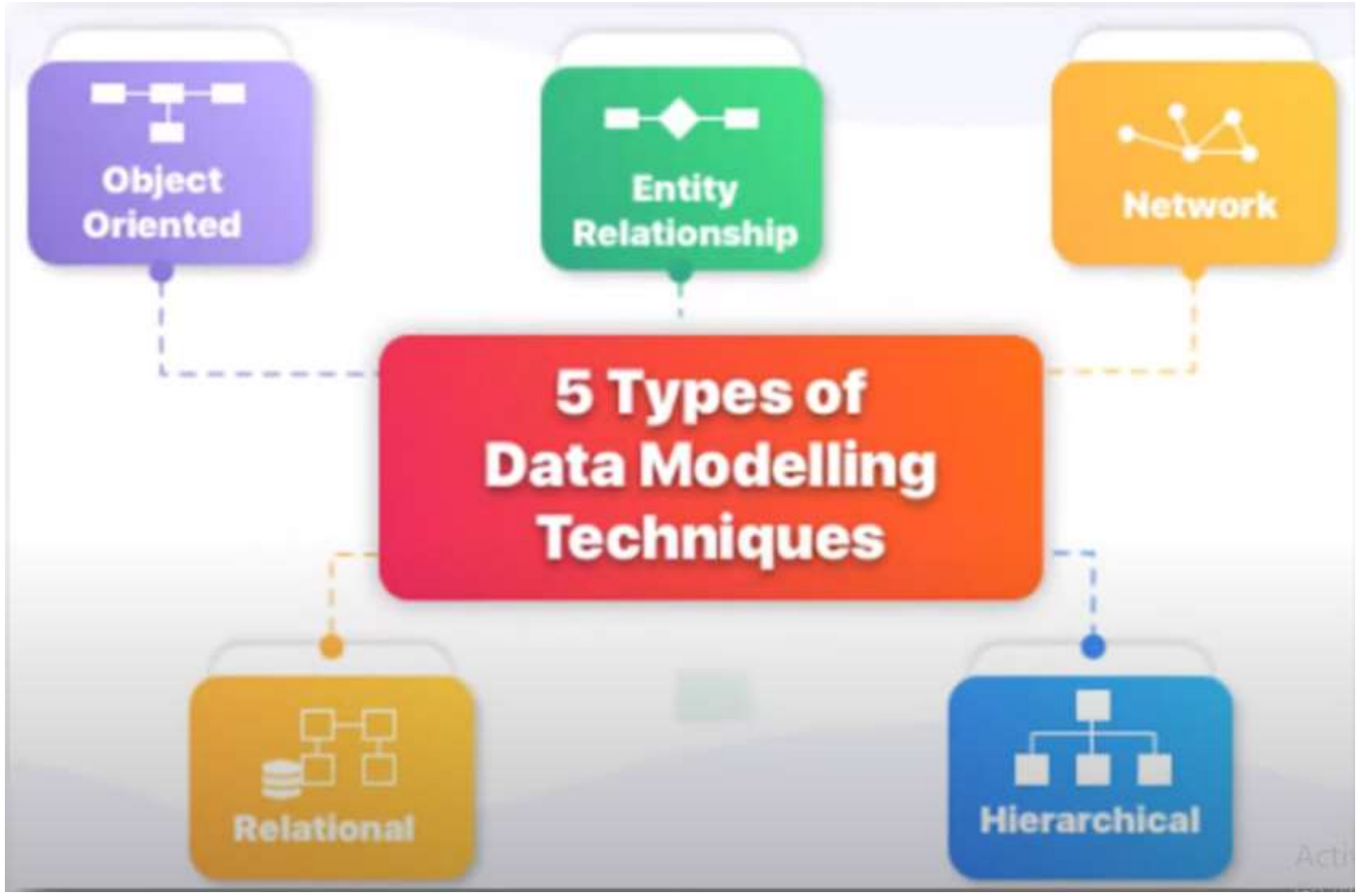
4. Facilitate business to function efficiently.

5. All the stakeholders to get a clear and easy understanding of what information is going to be stored

6. The core rules and policies that will govern each entity

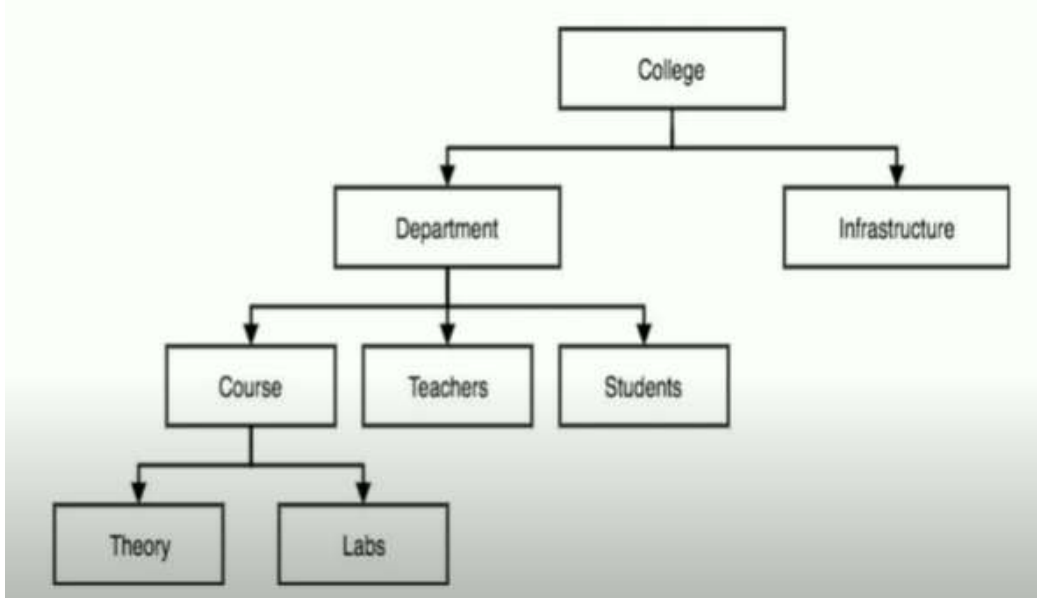


Data Modeling Techniques



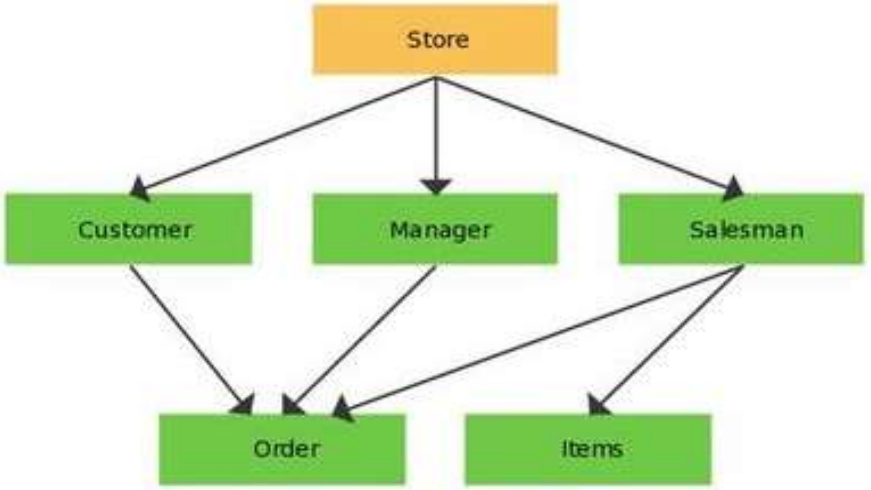
1. Hierarchical Model

- It has a root or a parent node and then follows a tree-like structure with other child nodes.
- Only one parent is allowed for a child node, and a parent node can have multiple child nodes.
- It is very easy and simple to understand.
- Simplex data models can be represented using this technique.



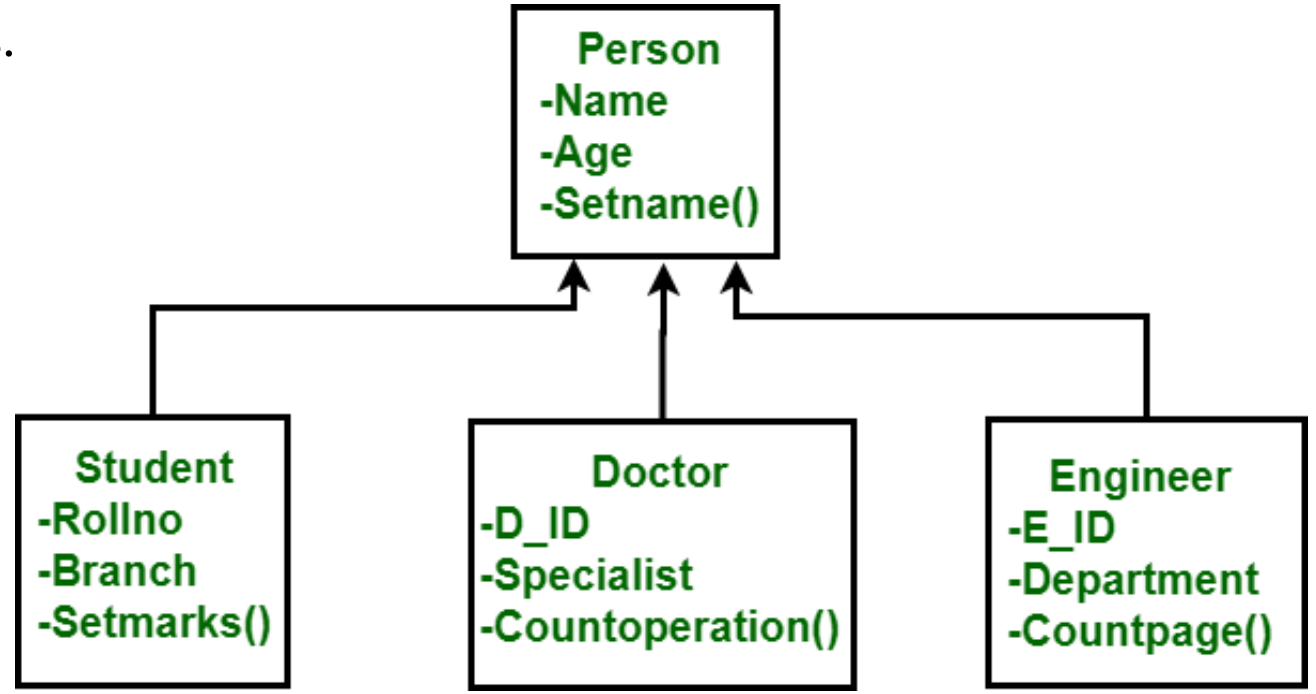
2. Network Model

- This model advance of hierarchical model. Show the Graph of system.
- Here, a child node could have multiple parent nodes.
- Navigation is faster in this model, as there are multiple paths to reach a child entity.
- This model also gets very complex when data is large.



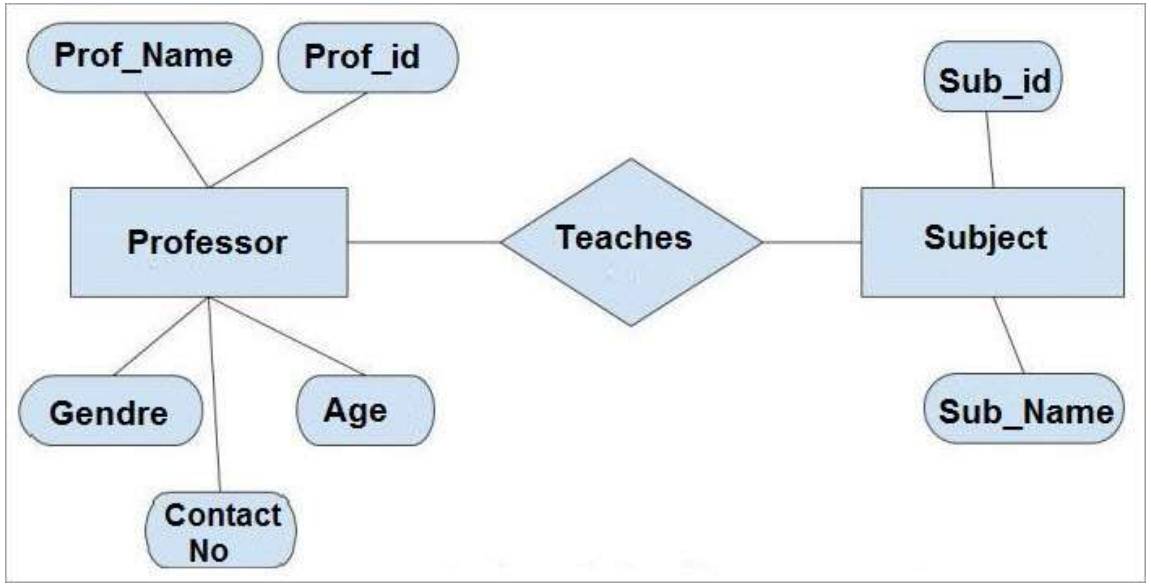
3. Object Oriented Model

- Object-oriented databases gained popularity with object-oriented programming.
- A real-world entity is represented as an object with attributes.
- The model presents a view as a collection of objects.
- Each object has its methods and features.



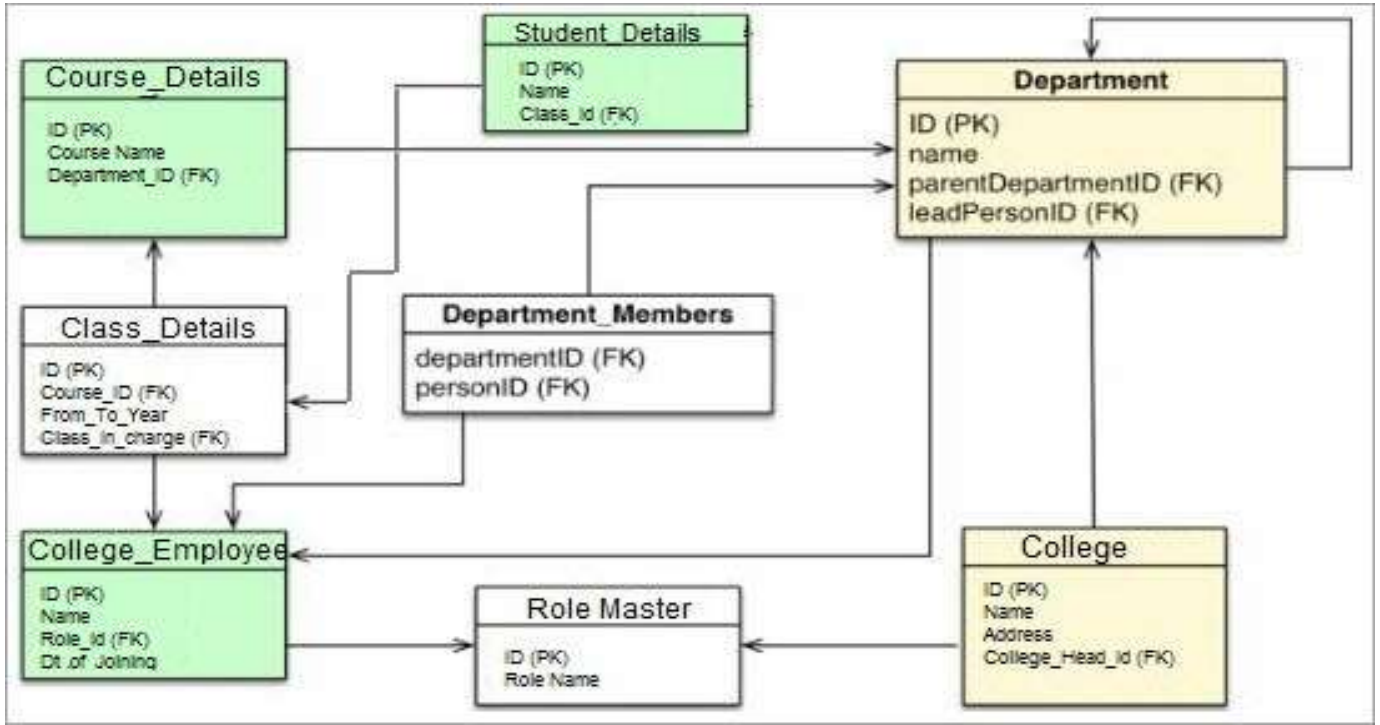
4. Entity Relationship Model

- This model represents the system at a very high level.
- Real-world problems can be represented through this model.
- It is very easy for all technical and non-technical stakeholders to understand.
- It is referred by database architects and developers to build the physical database.



5. Relational Model

- The relational Model is the most popular model.
- In this model, the Entity information is represented as a table.
- All the attributes are represented as columns.
- The primary key and foreign kev of each entity are also mentioned.



Data Modeling Tools

1. MySQL Workbench
2. ER/Studio
3. erwin Data Modeler
4. SQuirreL SOL Client
5. Draw.io
6. Lucidchart
7. Amundsen
8. Postico
9. Navicat
10. Datagrip