UNIT-3

Analysis

Analysis is the first systems development life cycle (SDLC) phase where we begin to understand, in depth, the need for system changes. Analysis is large process, we divide it into two main activities to make overall process easier to understand:

→ Requirements determination
→ Requirements structuring.

⊛. Determining system requirements:

Collection of information is the most important thing for system analysis. System analysts must collect the information about the current system and how users would like to improve their performance with new information system. Accurately understanding the user's requirements will help the system developing team deliver a proper system to end users in limited time and limited budget.

Characteristics of good system analyst:

i) Impertinence: Ask questions about everything that exist and also what may exist in the future.

ii) Impartiality: Find the best solution to a business problem or opportunity.

iii) Relax constraints: Eliminate unfeasibility, assume everything is possible.

iv) Attention to details: Everything must fit together so that the system works properly.

v) Reframing: Every system is different and needs a new creative approach.

Deliverables: The primary deliverables from requirements determination are various forms of information gathered during the determination process: transcripts of interviews, notes from observations, reports, job descriptions, analyzed responses from questionaries, and sets of forms.

⊛. Traditional methods for determining system requirements:

1) **Interviewing and listening:** Interviewing and listening involves talking with users individually or as a group to discover their views about the current and target systems. It also involves careful preparing an interview outline and guide before conducting the interview. Interviews are best done when only a few people are involved.

Types of interview questions:

i) **Open-ended questions:** These are usually used to examine for which we cannot expect all possible responses.

ii) **Closed-ended questions:** Provide a range of possible answers from which the interviewee may choose (True/False, Multiple choice, ranking items etc.).

2) **Group Interviewing:** Group interviewing consists of taking interview of several key people together. The advantages of group interviewing are: more effective use of time, can hear aggreements and disagreements at once etc. The disadvantage is that it is more difficult to schedule than individual interviews.

3) **Directly observing users:** Interviewing involves getting people to recall and convey information they have about organizational processes and the information systems that support them. People are not always reliable and say what they think is the truth. People often do not have a completely accurate appreciation of what they do or how they do it. Because people cannot always be trusted to interpret and report their own actions reliably, we can supplement what people tell us by watching what they do in work situations.

4) Analyzing Procedures and Other Documents:

Both interviewing and observing have some sort of limitations. Methods for determining system requirements can be enhanced by analyzing procedures and other documents of an organization. By examining existing system and organizational documentation, system analyst can find out details about current system and the organization. In documents analyst can find information such as problems with existing system, reasons why current systems are designed and many more.

⊛. Contemporary Methods for determining system requirements:

1) Joint Application Design (JAD): JAD is a team based approach for defining the requirements for new or modified systems. The main idea behind JAD is to bring together the key users, managers, and system analysts involved in the analysis of a current system. The primary purpose of JAD is to collect system requirements simultaneously from the key people involved in the system. The result is an intense and structured, but highly effective process. Having all the key people together in one place at one time allows analysts to see where there are areas of agreement and where there are conflicts.

2) Prototyping: Prototyping is a means of exploring ideas before investing in them. Prototyping allow system analysts quickly show users the basic requirement into a working version of the desired system. Prototyping is most useful when user requirements are not clear or well understood. Most system developers believe that the benefits from early usability data are at least ten times greater than those from late usability data.

⊛ Radical methods for determining system requirements:

1) **Business Process Reengineering:** The overall process by which current methods are replaced with radically new methods is referred to as business process reengineering (BPR). Reorganize complete flow of data in major sections of organization, Eliminate unnecessary steps, Combine steps, and Become more responsive to future change are its main goals. This method search for implementation of radical change in business processes to achieve breakthrough improvements in products and services.

2) **Identification of process to reengineer:** The first step in any BPR effort is to understand what processes need to change, what are the key business processes for the organization. Key business processes are the structured set of measurable activities designed to produce a specific output for a particular customer or market. Key business processes are customer focused. After identifying key business processes, the next step is to identify specific activities that can be radically improved through reengineering.

3) **Disruptive Technologies:** Technologies that enable breaking long-held business rules that discourage organizations from making radical business changes, are called disruptive technologies. In this method problems are first identified and then solutions are formulated.

#Structuring System Process Requirements:

⊛. Process Modeling: Process modeling involves graphically representing the processes, or actions, that capture, manipulate, store and distribute data between the system and its environment and among components within the system. A common form of a process model is data-flow diagram (DFD).

Modeling of system's process: Analysts use both process and data models to establish the specification of an information system. With a supporting tool, such as CASE tool, process and data models can also provide the basis for the automatic generation of an information system.

Deliverables and Outcomes: In structured analysis, the primary deliverables from process modeling are set of coherent, inter-related data-flow diagrams. Following are the deliverables for process modeling:

- → Context DFD
- → DFDs of current physical system.
- → DFDs of new logical system.
- → Through description of each DFD component.

⊛. Data Flow Diagrams: Structure of DFD allows starting from a broad overview and expands it to a hierarchy of detailed diagrams. DFD has often been used due to the following reasons:

- → Logical information flow of the system.
- → Determination of physical system construction requirements.
- → Simplicity of notation.
- → Establishment of manual and automated systems requirements.

Defining DFD: Graphical representation of a system's data and how the processes transform the data is known as DFD. Unlike, flowcharts, DFD's do not give detailed descriptions of modules but graphically describe a system's data and how the data interact with the system.

OR components of DFD

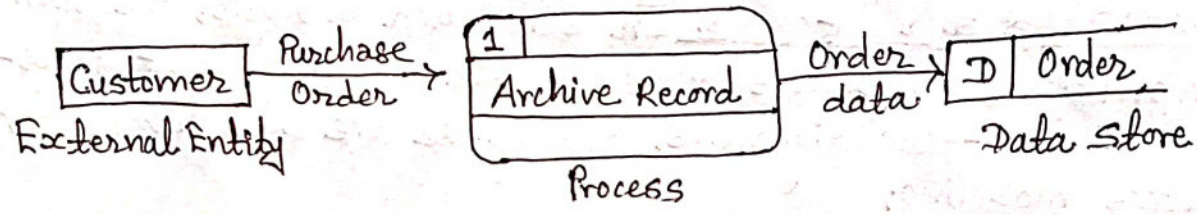
⊛ Symbols used to represent data-flow diagram:

1) **Process:** A process receives input data and produces output. Every process has a name that identifies the function it performs. The name consists of a verb, followed by a singular noun.

Example: Apply payment, Verify order etc.

Notation: A rounded rectangle represents a process. Process are given IDs for easy refrencing.

Process Example:



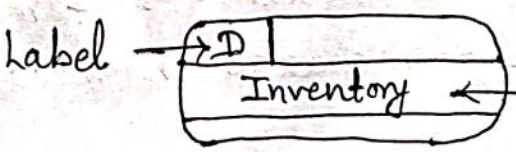
2) **Data Flow:** A data-flow is a path for data to move from one part of the information system to another. A data-flow may represent a single data element such as customer ID or it can represent a set of data element (or a data structure).

Notation;

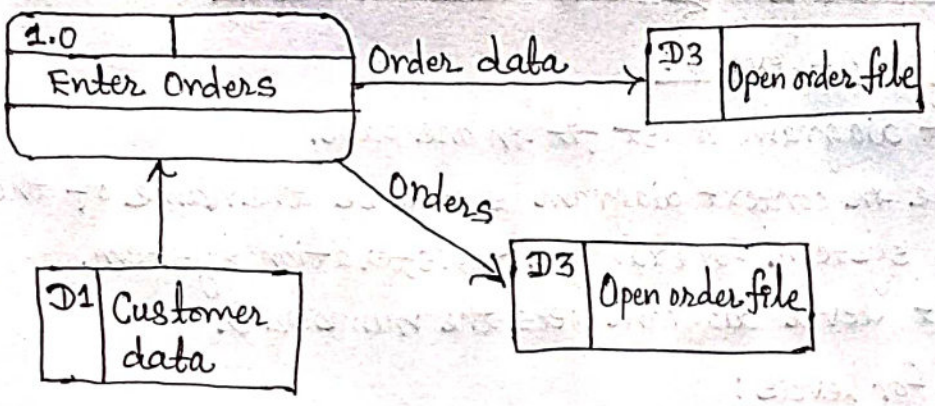
→ Straight lines with incomming arrows are input data flows.
→ Straight lines with outgoing arrows are output data flows.

3) **Data Store:** A data-store is used in DFD to represent a situation when the system must retain data because one or more processes need to use the stored data in a later time. A data-store must be connected to a process with a data-flow. Each data store must have at least one input data-flow and at least one output data-flow.

Notation: → Data can be written into data store, representing by outgoing arrow.
→ Data can be read from data store, representing by an incomming arrow.



Example:

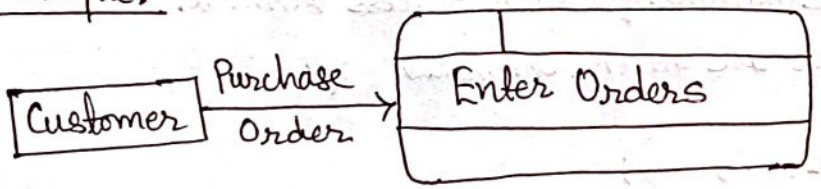


4) External Entity:

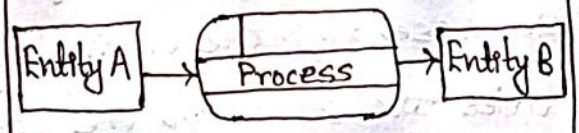
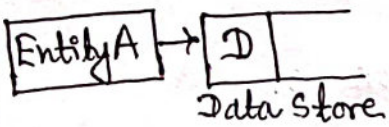
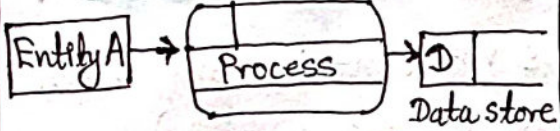
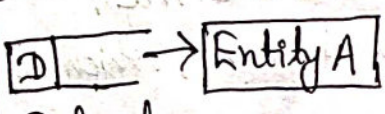
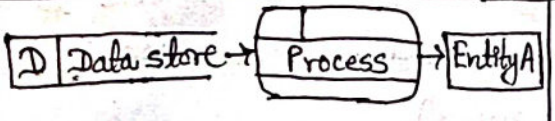
An external entity is a person, department, outside organization, or other information system that provides data to the system or receives output from the system. External entities are also called terminators because they are data origins or final destinations. An external entity must be connected to a process through a data flow.

Notation: A rectangle represents an external entity. They either supply data or receive data but not process data.

Example:



⊛. Rule of Data Flow:

Wrong	Right	Description.
		An entity cannot provide data to another entity without some processing occured.
		Data cannot move directly from an entity to data store without being processed.
		Data cannot move directly from a data store to any entity or another data store without being processed

⊛ Guidelines for Developing Data-Flow Diagram:

1. Context Diagram - Level 0:

→ The context diagram must fit in one page.

→ Process name in context diagram should be the name of the information system. For example: Registeration system.

→ The context level diagram gets the number 0.

2. Unique Name For Levels:

→ Unique Names should be used for each set of symbols.

→ For example, there can be only one entity in all levels of the data-flow diagrams.

3. No Cross Line in DFD:

→ One way to achieve this is to restrict number of process in DFD.

→ Another way is to duplicate an external entity or data store using a special notation such as asterisk.

4. Numbering Convention:

→ Use a unique refrence number for each process symbol.

→ Other process numbers are in the hierarchy of:

- → (1, 2, 3, ...);
- → (1.1, 1.2, ..., 2.1, 2.2, ...);
- → (1.1.1, 1.1.2, ...);

⊛. **Context-Level Diagram:** A context diagram gives an overview and it is the highest level in a data flow diagram, containing only one process representing the entire system.

→ All external entities are shown on the context diagram as well as major data flow to and from them.

→ The diagram does not contain any data storage.

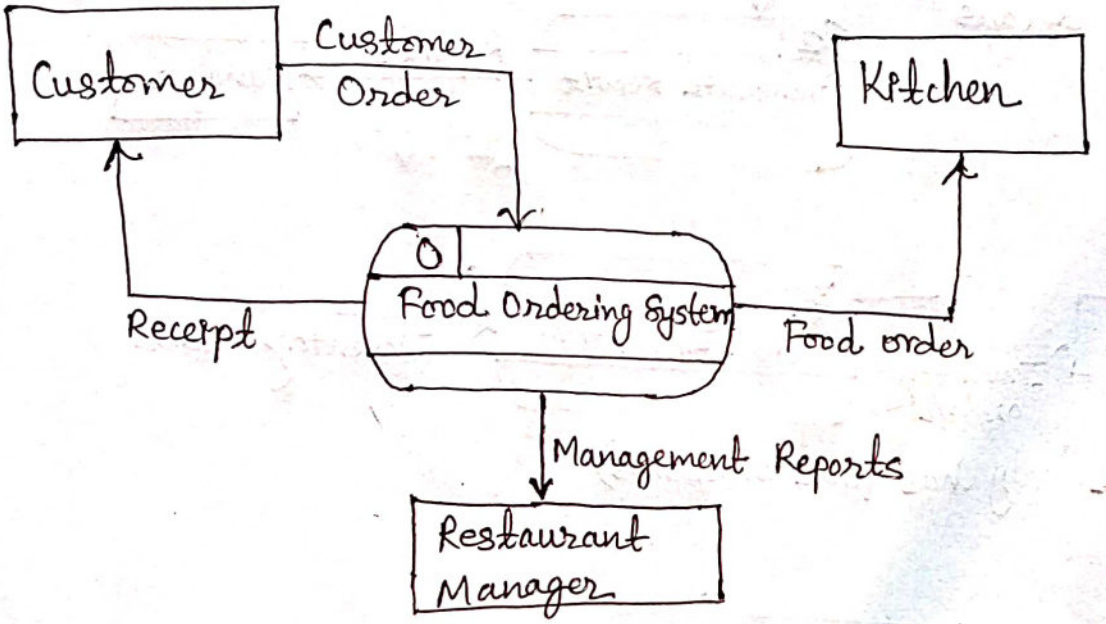
→ The single process in the context-level diagram, representing the entire system, can be exploded to include the major processes of the system in the next level diagram, which is termed as diagram 0.

Model Set [Imp]

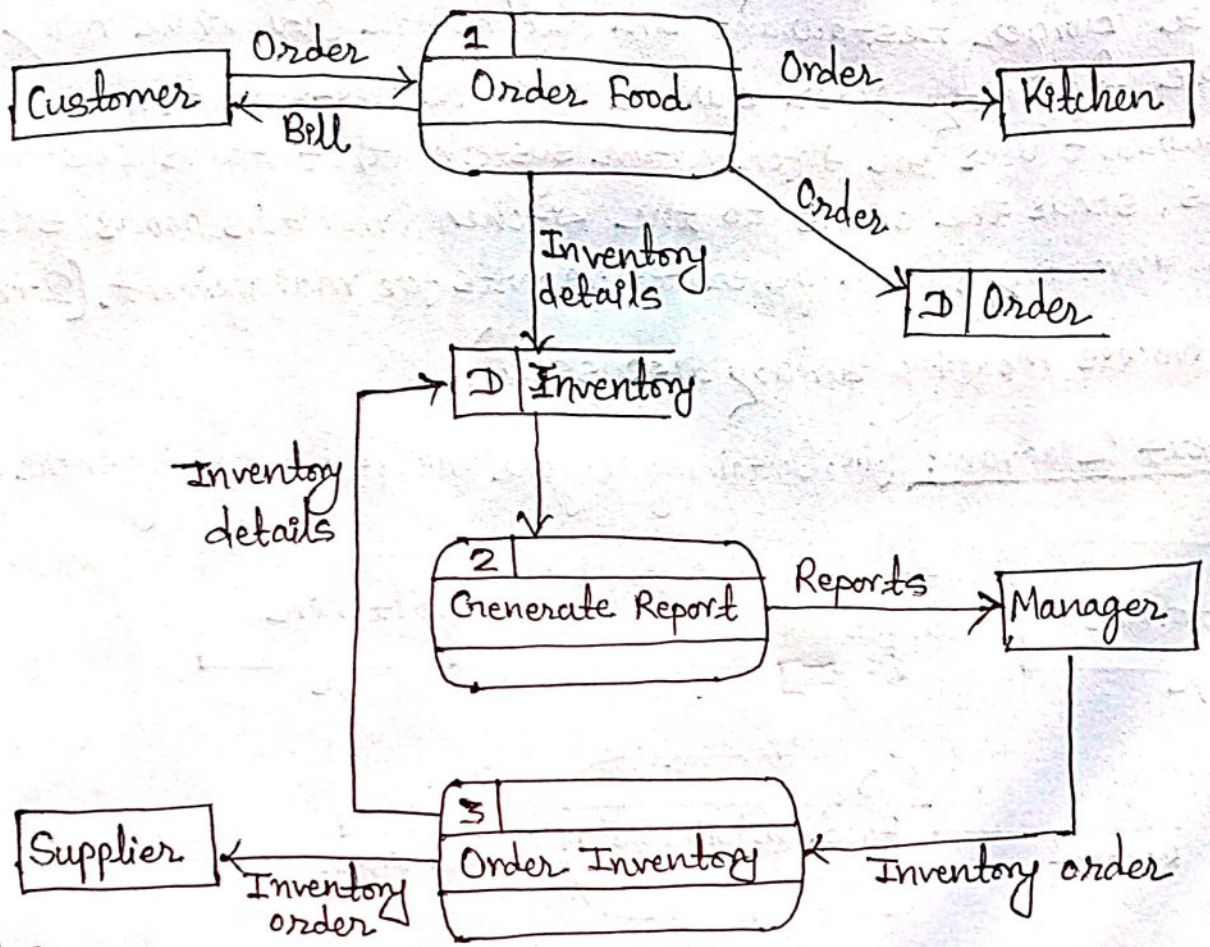
Q.3 What is process modeling? Draw context diagram and DFD for a burger restaurant in Kathmandu City where many people frequently order burger at the restaurant. The restaurant uses an information system that takes customer orders, sends the orders to the kitchen, monitors goods sold and inventory, and generates reports for management. (2+8)

Ans: (Process Modeling already discussed).

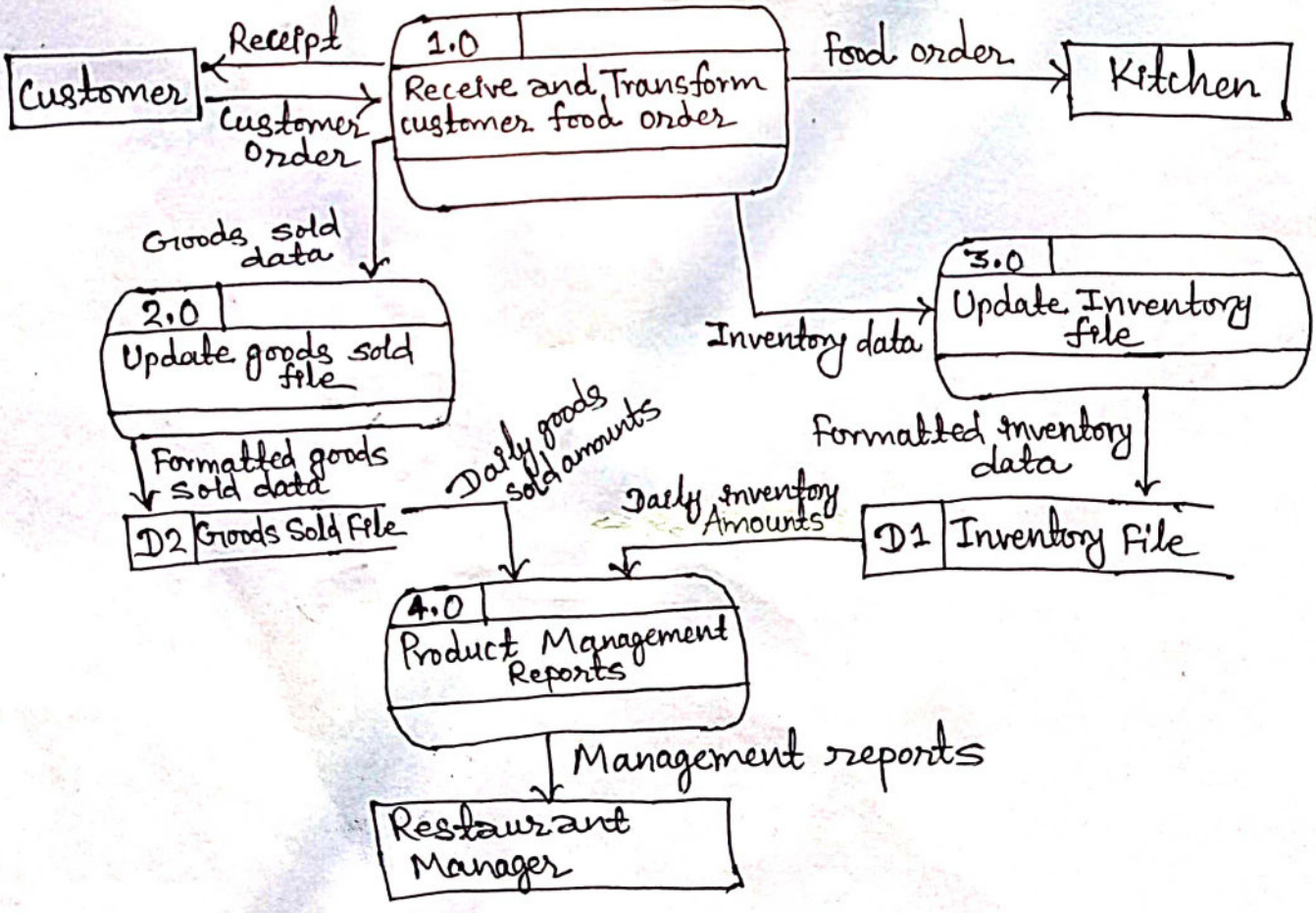
Context Diagram: (i.e, Containing only one process representing entire system).



Level 1 DFD: (only has three processes).



Level 2 DFD: (only has four processes)



⊛. Logical vs. Physical Data Flow Diagrams:

A logical data flow diagram focuses on the business and how the business operates. It is not concerned with how the system will be constructed. We can ignore implementation specifies such as, computer configuration, data storage technology etc.

A physical data flow diagram shows how the system will be implemented, including the hardware, software, files, and people in the system. It is developed such that the processes described in the logical data flow diagrams are implemented correctly to achieve the goal of the business.

Benefits of logical data flow diagram:

→ It is based on business events and independent of particular technology.
→ It allows analyst to understand the business being started.
→ Systems Implemented based on logical DFD will be easier to maintain
→ Physical DFD can be easily formed by modifying a logical DFD.

Benefits of physical data flow diagram:

→ Describes processes in more detail than logical DFDs.
→ Identifies temporary data storage.
→ Clarifies which processes are manual and which are automated.
→ Adds controls to ensure the processes are done properly.

⊛ Advantages of JAD: [Imp]

→ JAD reduces costs and time for project development.
→ JAD encourages the team to push eachother to work faster and deliver on time.
→ JAD allows us to produce better, error-free software.
→ JAD lowers the risks.
→ JAD improves system quality.

⊛. Logic Modeling:

Logic modeling is a graphic representation for the shared relationships among the resources, activities, outputs, outcomes, and impact for our program. Logic model has four components:

i) Needs → Needs are about the problem and why it's important to address it.

ii) Inputs → These are the things that contribute to addressing the need.

iii) Activities → These describe the things that the inputs allow to happen.

iv) Outcomes → Outcomes are usually expressed in terms of measures of success.

⊛ Modeling Logic with Decision Tables: [Imp]

Decision tables are brief visual representation for specifying which actions to perform depending on given conditions. They are algorithms whose output is a set of actions. It is an excellent tool to use in both testing and requirement management.

Example: Decision table for ATM would be as follows:

Conditions	R1	R2	R3
Withdrawl amount<=balance	T	F	F
Credit Granted	-	T	F
Actions			
Withdrawl granted	T	T	F

In a decision table, condition are usually expressed as True(T) or False(F). The above table contains three different columns which are business rules, like one of the business rule for ATM is that machine pays out if the customer has sufficient funds in their account.

Steps to create decision tables:

Step 1: Analyze the requirement and create the first column. Express conditions and resulting actions in a list so that they are either TRUE or FALSE.

Step 2: Add Columns by calculating how many columns are needed in the table. The number of columns depends on number of conditions. Mathematically, no. of columns is $2^{conditions}$. E.g. If 2 conditions then, $2^2 = 4$ columns.

Now, T (True) and F (False) for conditions are filled.

Step3: Reduce the table by deleting the columns that have become identical and mark insignificant values with "—".

Step4: Determine and enter actions for each column in the table.

Step5: Write test cases based on the table.

Advantages of decision table:

→ They are easer to draw.
→ Decision table can be easly changed according to the situation.
→ A small table can replace several pages of flow chart.
→ The requirements become much clearer.
→ Decision tables make it possible to detect combinations of conditions.

Disadvantages of decision table:

→ When there are two many alternatives, decision table cannot list them all.
→ It can not express complete sequence of operations to solve a problem.
→ They only present partial solution.

⑧. Modeling logic with Decision Trees: [Imp]

A decision tree is a flowchart like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of a test, and each terminal node holds a class label. Decision trees are method for defining complex relationships by describing decisions and avoiding the ~~prbo~~ problems in communication.

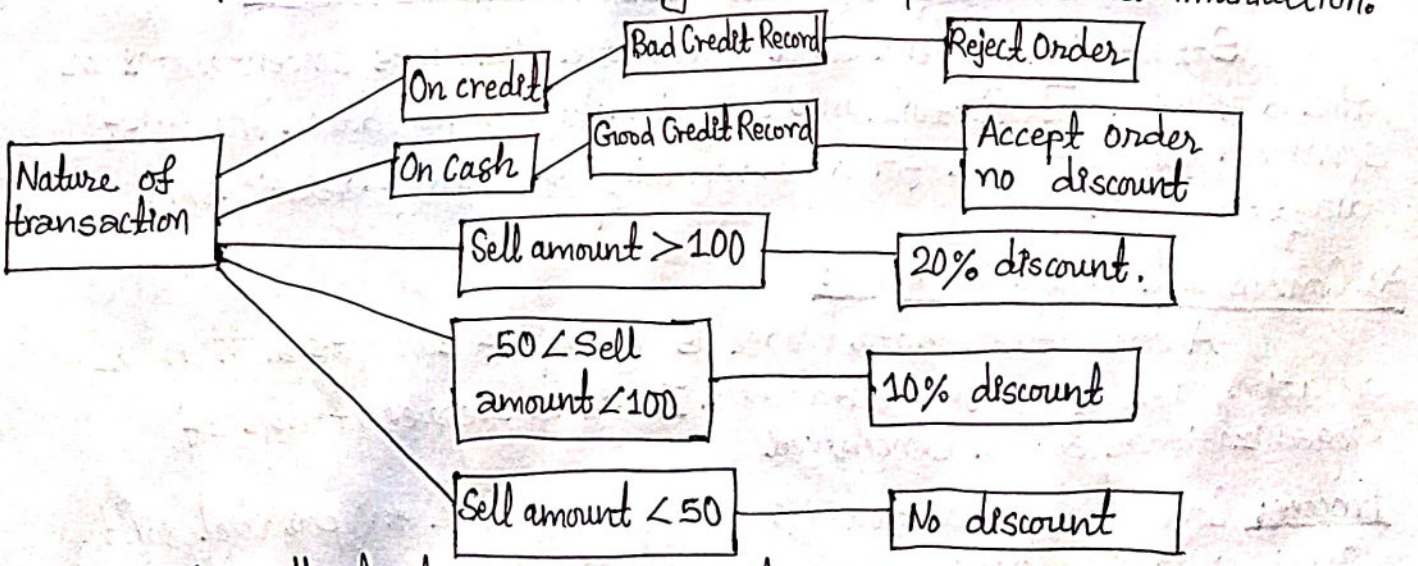


Fig: Illustration of decision tree.

Advantages of decision tree:

→ It expresses the logic of if then else in pictorial form.
→ It helps the analyst to identify the actual decision to be made.
→ It is used to verify logic and problems that involve a few complex decision and limited number of actions.

Disadvantages of decision tree:

→ There is absence of information in its format to take what other combinations of conditions to test.
→ A large number of branches with many paths will confuse rather than help in analysis.

⊛. Modeling logic with Pseudo-Codes:

Pseudo-code is an informal way of programming description that does not require any strict programming language syntax. It is used for creating an outline or a rough draft of a program. It has no syntax like any of the programming language and thus can't be compiled by the computer.

Advantages of Pseudo-code:

→ Improves readability of any approach.
→ It is one of the best approach to start implementation of an algorithm.
→ Acts as a bridge between the program and the algorithm or flowchart.
→ Explains what exactly each line of a program should do.

Example: We can give pseudo-code of any algorithm we know:

#Structuring System Data Requirements:

Structuring system and data requirements concentrates on the definition, structure and relationships within data. The most common format used for data modeling is entity-relationship diagramming.

⊛. Conceptual data modeling:

A conceptual data model is a map of concepts and their relationships used for databases. It is a representation of organizational data.

Conceptual data modeling is typically done in parallel with other requirements analysis and structuring steps during system

analysis. On larger systems a subset of project team concentrates on data modeling while other team members focus attention on process or logic modeling. Analysts develop a conceptual data model for current system.

Process: This process usually begins with developing a conceptual data model for the existing system. Then a new CDM, which includes all of the data requirements for the new system is built. E-R diagrams can be translated into wide variety of technical architectures for data, such as relational, network and hierarchical.

Deliverables and Outcomes: The primary deliverable from the conceptual data modeling is the entity-relationship diagram. Another deliverable from CDM is a full set of entries about data objects to be stored in the project repository.

⊛. Gathering Information for Conceptual Data Modeling:

The first view is generally called the top-down approach. This view derives the business rules for data model. The bottom-up approach is process of gaining an understanding of data by reviewing specific business documents handled within the system. Requirements determination methods must include questions and investigations that take a data.

⊛. Introduction to E-R Modeling: [Imp]

Entity-Relationship (ER) Modeling is a graphical approach to database design. It is a high-level data model that defines data elements and their relationship for a specified software system. An ER model is used to represent real-world objects.

⊛ Elements of ER Diagram / ER Design Issues:

Entity: Entity is a thing or object in the real world with an independent existence. An entity may be an object with a physical existence or it may be an object with conceptual existence. For example: "Aarav" is a particular member of entity type student.

Entity type: It is a collection of entities having common attribute. For eg, student is an entity type.

Entity set: It is same as an entity type, but defined at a particular point in time, such as students enrolled in class on the first day.

Attributes: The particular properties that describe entity are known as attributes. For example, an EMPLOYEE entity may be described by the employee's name, age, address, salery, job etc.

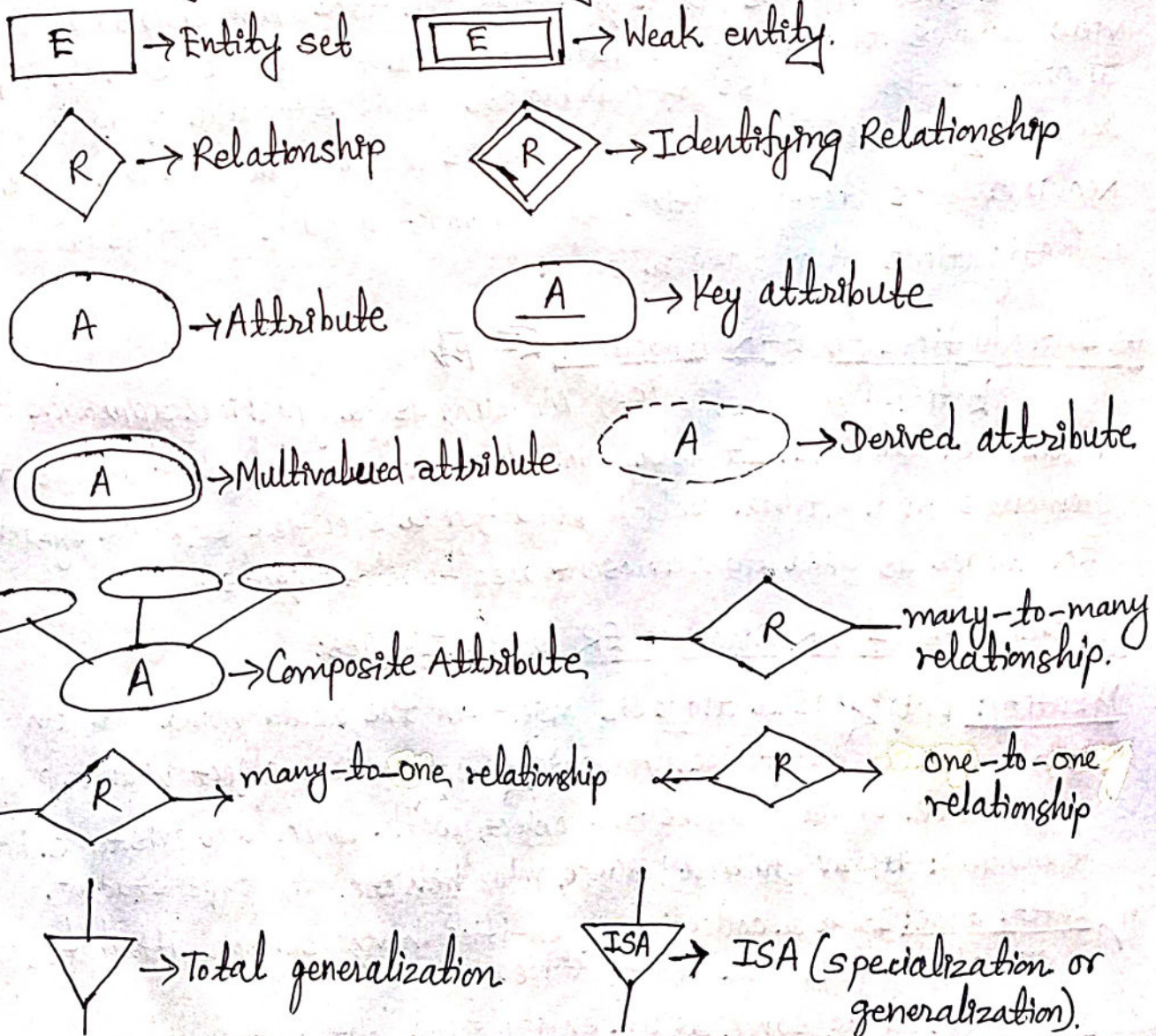
Types of attributes:

Atomic vs. composite → An attribute that cannot be divided into smaller independent attribute is known as atomic attribute. and attribute that can be divided are composite.

Single vs. Multi valued → An attribute having only single value for an entity is single valued attribute and that have multiple value for entity is multi valued.

Key attribute → An attribute that has unique value of each entity is known as key attribute. For e.g, every student has unique roll number in the class.

Different symbols used in ER-diagram:



Relationships: Relationships are the associations among entities.

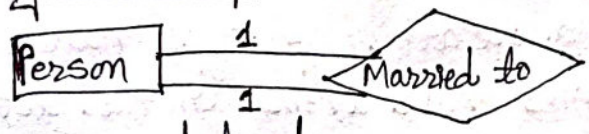
A relation type R among n entity types $E_1, E_2, \ldots, E_n$ defines a set of associations among entities from entity types.

Degree of Relationship: Number of entity sets that participate in a relationship set is called degree of the relationship set. On the basis of degree, relationships can be divided as below:

i) **Unary relationship:** Only one entity set participates in a relation.

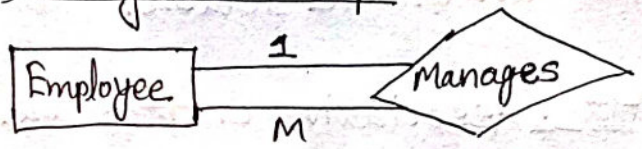
↳ **One to one (1:1) unary relationship:**

e.g.,



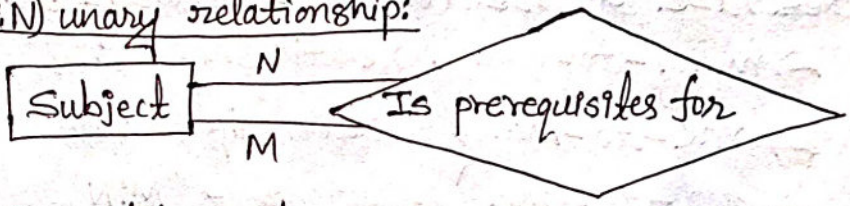
↳ **One to many (1:M) unary relationship:**

e.g.,



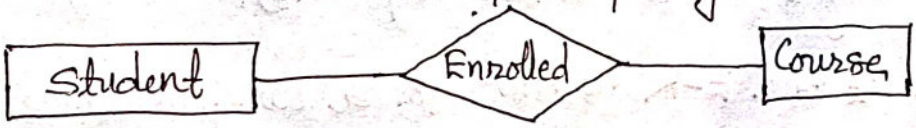
↳ **Many to many (M:N) unary relationship:**

e.g.,



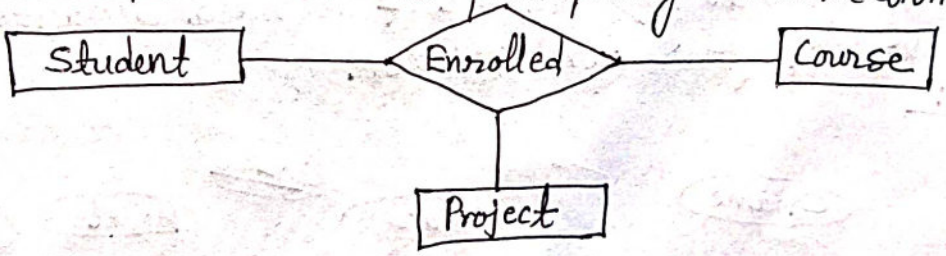
ii) **Binary relationship:** Two entities set participating in a relation.

e.g.,



iii) **N-ary relationship:** n entities set participating in a relation. (i.e, $n > 2$)

e.g.,



Cardinality: Cardinality defines how many instances of one entity are related to instances of another entity. It describes fundamental relationship between two entities or objects. There are three cardinalities: one-to-one, one-to-many, and many-to-many. E-R diagrams are used to describe the cardinality in databases.

⊛ Naming Conventions:

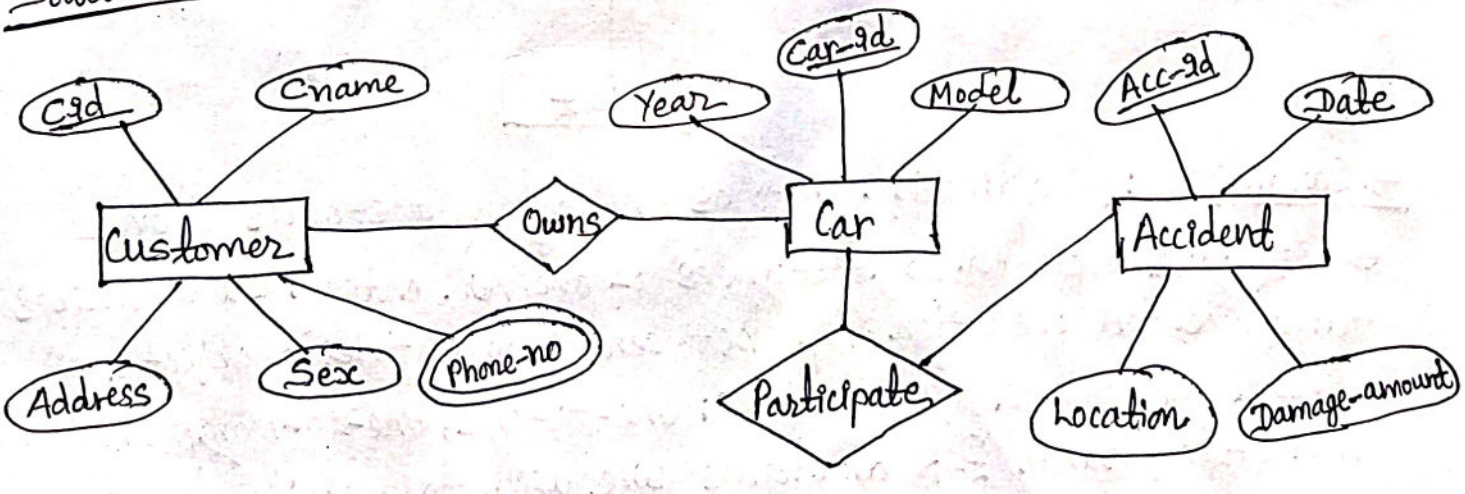
When designing a database schema, the choice of names for entity types, attributes, relationship types, and roles is not always straightforward. One should choose names that convey, as much as possible, the meanings attached to the different constructs in the schema. We choose to use singular names for entity types, rather than plural ones, because the entity type name applies to each individual entity belonging to that entity type. In ER diagrams we use the convention that entity type and relationship type names are uppercase letters, attribute names have their initial letter capitalized and role names are lowercase letters.

⊛ **Associate Entity:** Associative entities are connections that describe a relationship between two different entities.

- ↳ many-to-many relationship (1:1)
- ↳ one-to-many relationship (1:M)
- ↳ one-to-one relationship (M:N)
- ↳ many-to-one relationship (M:1).

Q. Construct an E-R diagram for a car-insurance company whose customers own one or more cars each. Each car has associated with it zero to any number of recorded accidents. Assume attributes of your own interest:

Solution:



[Imp]

Q What is data modeling? How is it different from process modeling? How do you use entity relationship model for data modeling? (1+2+2).

Ans: Data modeling is the process of creating a data model for the data to be stored in a database. This data model is a conceptual representation of data objects, the association between different data objects and the rules. Entity Relationship (E-R) modeling is most popular data modeling techique.

Data Modeling	Process Modeling
i) It is a procedure of documenting a complex software design as an easily understood diagram to explain the way data need to flow.	i) It is a procedure of classifying the processes of the same nature together into a model.
ii) It involves creating data models using formal techniques.	ii) It categorizes similar processes as a single model.
iii) The most common format used is Entity-Relationship diagramming (ERD).	iii) The most common format used is data-flow diagram (DFD).

For data modeling entity relationship model uses following three main constructs:

i) Entities: Entity is a thing or an object in the real world with an independent existence. An entity may be an object with physical existence or it may be an object with conceptual existence. Entities are denoted by rectangles in E-R diagram.

ii) Attributes: The particular properties that describe entity are known as attributes. For Eg. An EMPLOYEE entity may be described by attributes as name, age, salery, job etc. Attributes are denoted by oval shapes in E-R diagram.

Relationships: Relationships are the associations among entities. Relationships are denoted by diamond shapes in E-R diagram.