Unit-3 : Anaysis.

- Ankit Pangeni

DATE

Anaysis is one of the phases of SDLC. It is the phase where we deeply understand the need for system change. It is a large process and it can be divided into two main activities.

→ Requirement determination
→ Requirement structuring

3.1 Determining System Requirements

* Introduction

- It is the process of gathering information on what the new system should do.
- The requirements is gathered from various sources possible such as users of current system, reports, forms and so on.
- Accurately understanding the user's requirements will help the developing team deliver a proper system to the end users in limited time & limited budget.

- A system anaysis should have following characteristics during requirements determination:
  → Relax constraints (Assume anything is possible & eliminate infeasibility)
  → Impertinence (ask question about everything)
  → Impartiality (find best solution to a problem)
  → Attention to details (everything must fit together in the system)
  → Reframing (every system is different & needs a new creative approach)

- The primary derivables from the requirements determination are various forms of information gathered during the determination process. These information can be divided into 3 groups:

PAGE

DATE

→ Informations collected from conservations with or the observation of users (interview, notes from observation, etc)
→ Existing written information (forms, reports, flowchart, documentation of existing system, flowcharts)
→ Computer based information. (JAD session result, CASE repo)

- Determining System requirements
  - Traditional method for D.S.R
    - → Interviewing & Listening
    - → Group Interviewing
    - → Directly observing users
    - → Analyzing procedures & other docum
  - Contemporary methods for D.S.R.
    - Joint Application Design (JAD)
    - Prototyping
  - Radical methods for D.S.R. → Business Process Reingeneer

* Traditional Method for Determining System Requirement

This is the most basic method for determining the system requirements. It includes interview.....

Note: Guidelines for effective interview.
→ Plan the interview
→ don't ask repetitive question
→ Listen carefully & take notes
→ seek diverse views
→ Be neutral.

Interviewing & Listening

→ It is one of the primary ways the analyst gathers information about the system requirements.
→ It involves sitting and talking with each users individually to discover their views about the current system and need for the new system
→ Interviews are best done when few people are involved
→ It is a time and resource consuming method.
→ Generally, interview questions are of two types:

Open-ended questions → allows to give answers by constructing their own response
Close-ended questions → answers are limited (ex: True/False, Multiple choice, rating or ranking items etc)

Group Interviewing

→ It involves interviewing several key people together.
→ Advantages: - More effective use of time
- can hear agreements & disagreements at once
→ Disadvantages: Difficulty in scheduling

Directly observing Users

→ It involves watching users do their job, interact with the current system in their natural environment.
→ This method can provide valuable insights on how users actually use the system, if they actually need the ~~New~~ system or not.
→ In interview, people often don't have a accurate views or they can't accurately interpret the requirements for new system But if you observe them do their work, it becomes easier to understand their need
→ It can be time consuming & resource-intensive method & there is limited time to observe.

DATE

o Analyzing Procedures & other documents

→ It involves reviewing existing documentation to identify the needs and goals of the new system
→ This method can be useful for understanding how a current system is being used & identifying areas of improvement.
→ While observing the documents, analyst can find information about: problems with existing systems, opportunities to meet new needs, reason why current systems are designed as they are, and many more. (data rules for processing data)
→ Problems: Sometimes the instructions in the documents are not up to date, documentation may not match with real life operations.....

* Contemporary Methods for determining systems req.

→ These are the modern methods for determining the system requirements

o Joint Application Design (JAD)

→ It is a structured approach in which key users, managers and analysts work together for several days in a series of intense meetings to specify or determine system requirements
→ It's better than traditional methods because you have all key personnel at one place at one time.
→ It is a team based approach & one of the most efficient method to resolve conflicts.
→ It helps in shorter development life cycle & greater client satisfaction

classmate

PAGE

→ The end result of JAD is a set of documents that detail the working of current system & the features of proposed systems.

→ The list of JAD session participants are:

- JAD session leader: A neutral person that organizes & runs JAD
- key users
- Sponsors.
- Managers
- System analyst:-
- Scribe: Person who takes notes of JAD sessions usually with computer
- IS staff:- programmers, database analyst, ...

○ Prototyping.

→ In prototyping, we quickly understand the requirements of the users & convert those requirements to a working system.

→ Once the user sees the requirements converted to system, will ask for modifications or will generate additional requirements

→ This method is useful when requirements are not clear., few users are involved in the system & tools are already available to build prototype.

→ Drawbacks:- tendency to avoid formal documentation, difficult to adapt to more general audience, SDLC checks are often bypassed.

* Radical methods for determining System Requirements.

It is the most innovative and unconventional method for identifying & determining the requirements for the new system. The goal of these methods is often to generate a deeper understanding of user needs & to ensure that the resulting system is more user-centered & meets the needs of all stakeholders.

DATE

Business Process Reengineering

→ It is a method for determining the system requirements that involves a complete redesign of existing business processes to achieve dramatic improvements in critical aspects like quality, output, cost, service & speed & customer satisfaction.

→ Here, we completely reorganize the flow of data in the major sections of organization, eliminate unnecessary steps, eliminate bottle-necks & inefficiencies and many more.

→ It is a powerful approach for determining system requirements as it allows organizations to fundamentally rethink how they operate to identify new opportunities for growth & innovation.

Identification of process to reengineer.

→ It is an important step in BPR. The selection of process to be reengineered is typically based on the area of the improvement of that process. Some common criteria to identify process for reengineering are:

- ↳ Processes that are time consuming & costly
- ↳ Processes that are performed manually & could be automated
- ↳ Processes that are not aligned with business objectives
- ↳ Processes that have a high impact in the organizations overall performance.

Disruptive Technology

Disruptive technologies are those that disrupt the existing market by introducing new way of doing things that is significantly different from what is currently available. Ex. AI, machine learning, IOT, etc.

classmate

PAGE

DATE

3.2 Structuring System Process Requirements

Once the system requirements have been generated, the next step is to structure those requirements. There are various methods for structing requirements, they are described below:

* Process Modeling

→ It involves graphically representing the processes, or actions that capture, store, manipulate and distribute the data between a system and its environment and among various components within the system Context diagram, DFD are some forms of process modeling.

→ It involves modeling of system's process for structured analysis where we utilize the informations gathered during the requirements determination and organize it into meaningful representation

→ The deliverables and outcomes of process modeling is a set of coherent, interrelated data flow diagrams and context diagrams of current as well as new logical system and the description of each DFD component.

DATE

* Data flow Diagram

◦ Context diagram and DFD

→ A context diagram shows the scope of the system indicating which elements are inside and outside the system. It gives the overview of the system and it is a highest level of abstraction in a DFD. It contains only one process representing the entire system. It contains all external entities, data flow and a process but not data store.

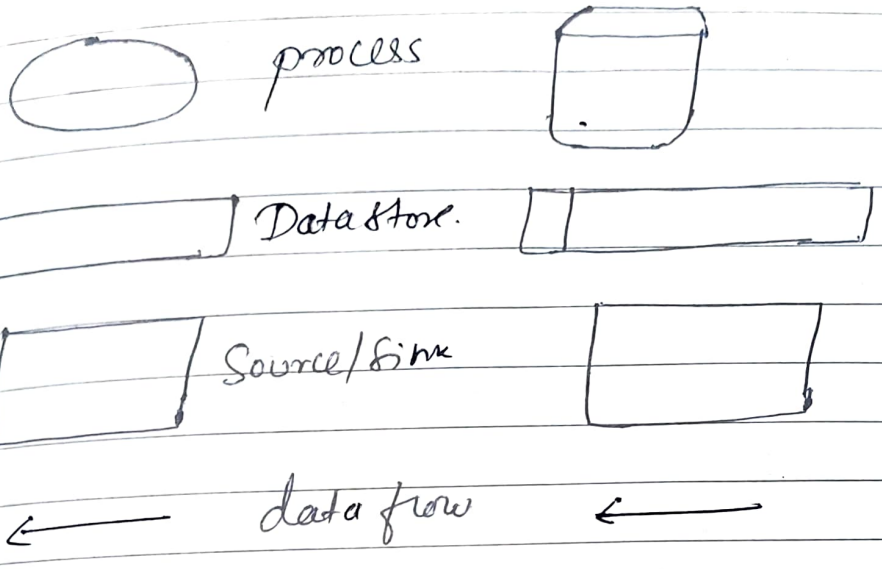
→ DFD are used to graphically illustrate the movement of data between external entities & the processes & the data stored within the system. DFD contain additional information about the system that the context diagram doesn't.

→ DFD has often been used due to the following reasons
- Logical information flow of the system
- Determination of physical system requirements
- Simplicity of notation
- Establishment of manual & automated system requirements

→ Unlike flowcharts, DFD's do not give detailed descriptions of modules but graphically describe a systems data & how the data interact with the system

DATE

- Symbols used in DFD / components of D.F.D



Eg: DeMarco and Yourdon symbols

Gane and Sarson symbols.

→ Process: Receives input and produces output of different content or form. It can be as simple as collecting input data & saving it in database or can be as complex as producing a report containing monthly sales of a retail shop. Every process has a name that identifies the function it performs. Ex: verify order, Calculate bill...

→ Data flows: It is a path for data to move from one part of the information system to another. It may represent a single data element such as a customer ID or a set of data elements. At least one data flow must enter and one data flow must exit from each process symbol because every process needs data to work.

classmate

PAGE

Depicts data at rest

DATE

→ Data Store: It is also known as data repository. It is used in DFD to represent a situation when the system must retain data because one or more process need to use the stored data. A data store must be connected to a process with a data flow.

→ Source/sink. (external Entity)
It represents the origin and/or destination of data. It is also referred to as External Entity. It is drawn as a square symbol.
An external entity can be a person, department outside the organization, or other information system that provides or receives data from the system
ex: a customer submitting order and receiving bill from the system.

o Levels in DFD

→ In DFD, there are different levels that represent the system at different levels of abstraction
→ Levels in DFD are numbered 0, 1, 2, or beyond
→ Here we mainly see 3 levels in DFD. i.e.
0-level DFD, 1-level DFD, 2-level DFD
→ Some authors say context diagrams are level 0 DFD

DATE

Rules of DFD.

→ Data can flow from
- External entity to process & viceversa.
- Process to data store & viceversa.
- Process to process

→ Data cannot flow from
- External entity to External entity
- External entity to data store. & viceversa.
- Data store to another data store

→ No process can have only inputs or only outputs, it needs to have both

→ "Data flow" naming don't start with a verb.
request for membership (X)
Membership request (✓)

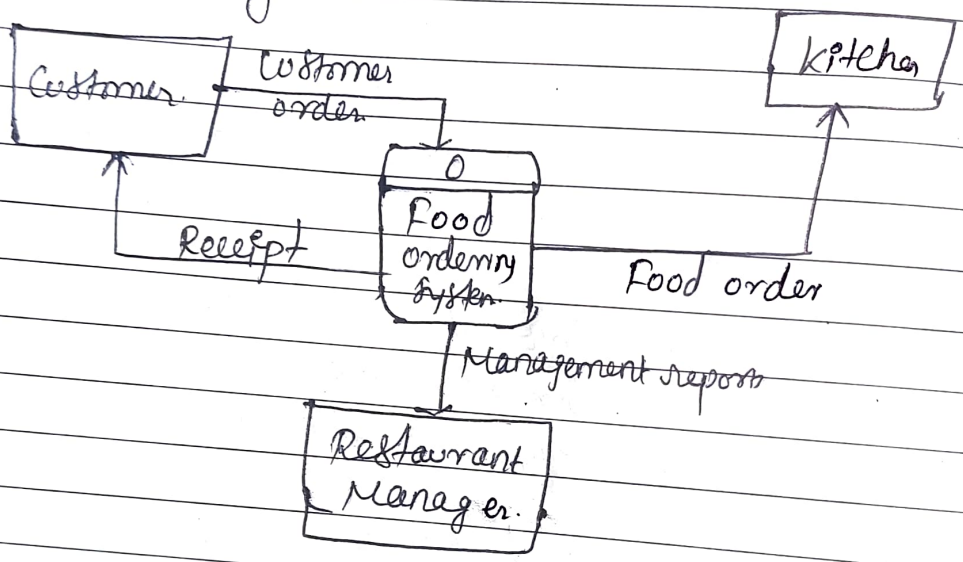
→ "Process" naming should be verb phrase.
membership registration (X)
Register member (✓)

DATE

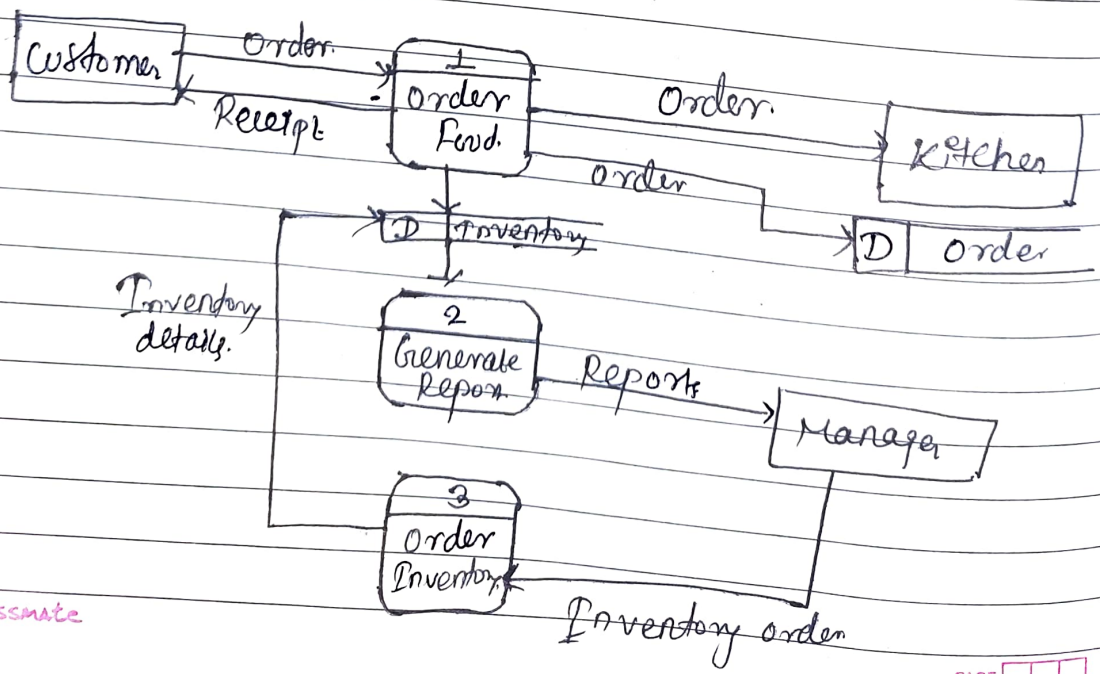
Some Examples of DFD & Context Diagram

1. A burger restaurant in KTM 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

→ Context diagram



Level-1 DFD

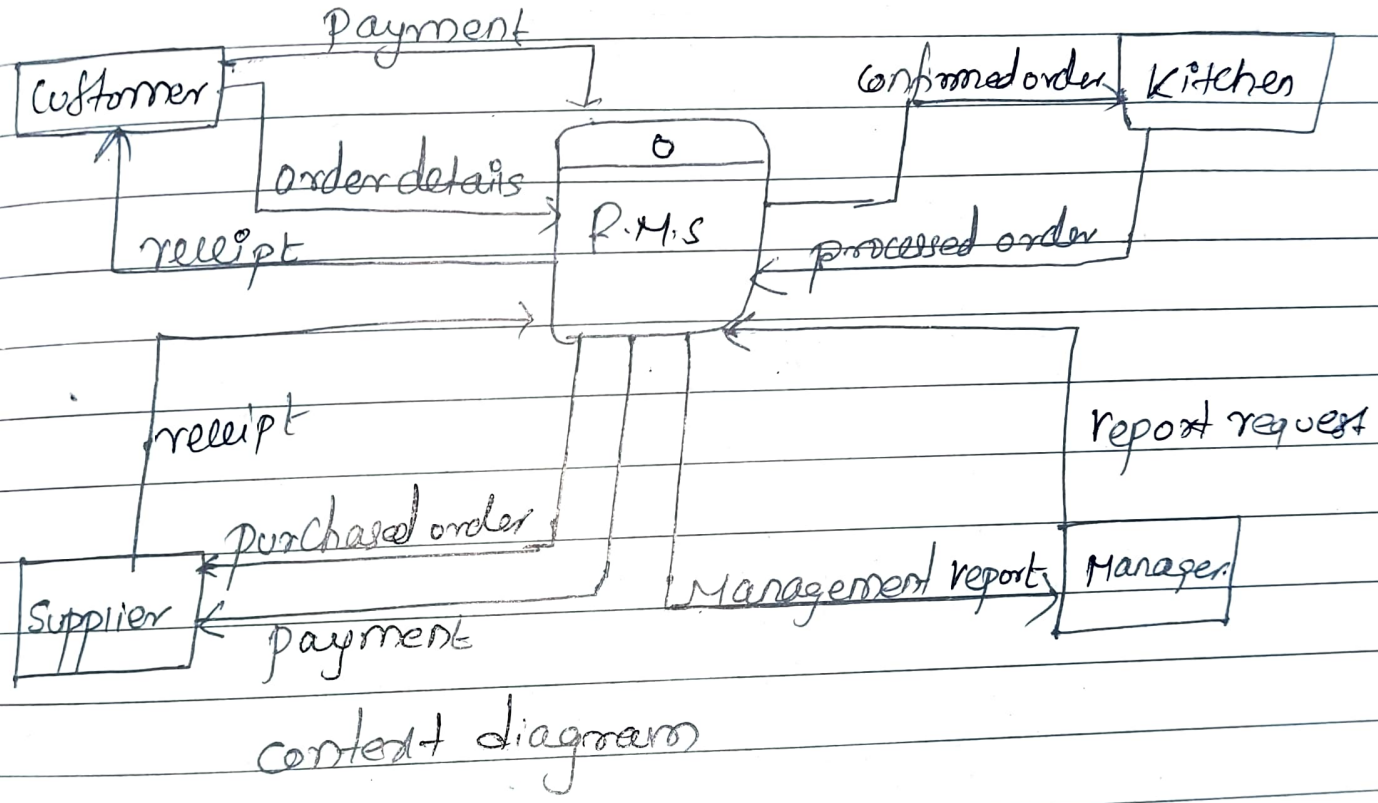


classmate

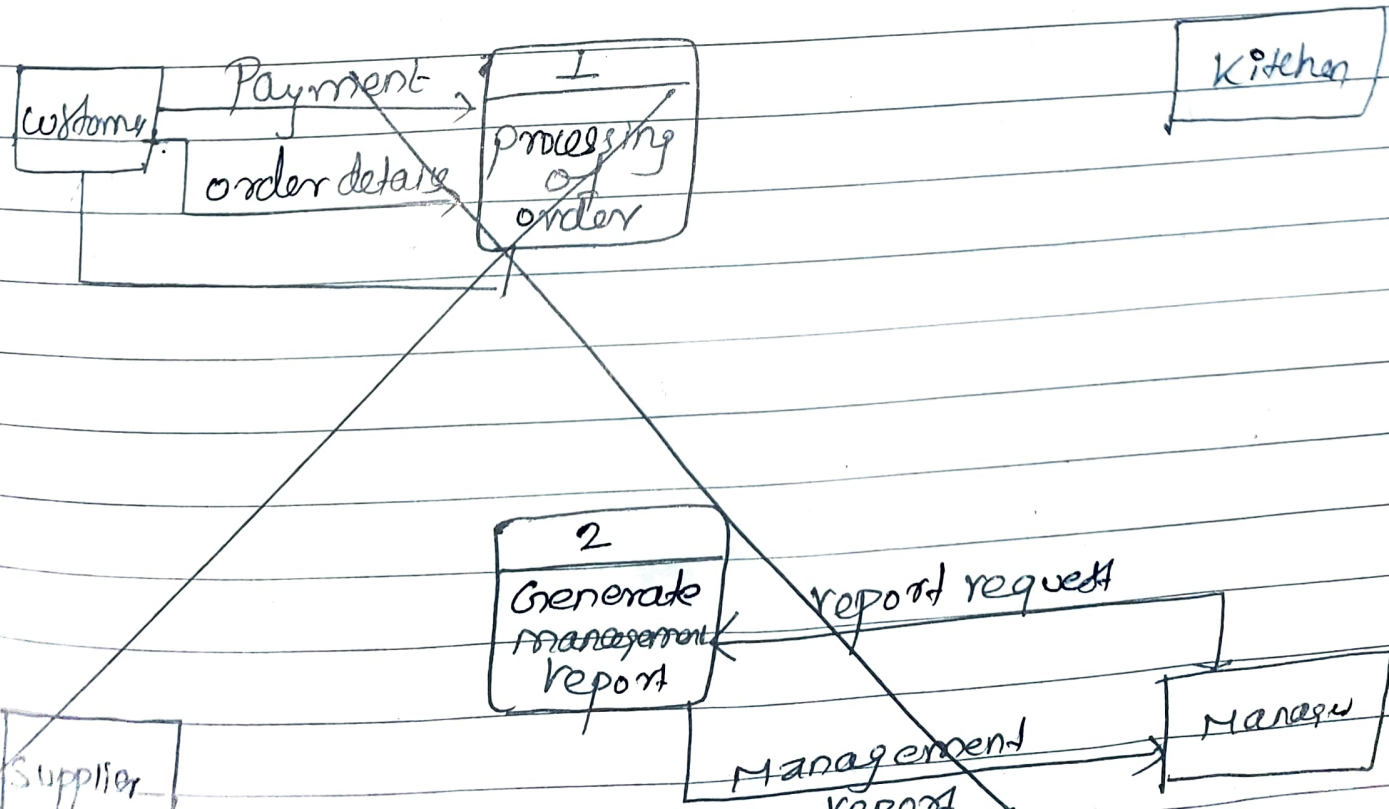
PAGE

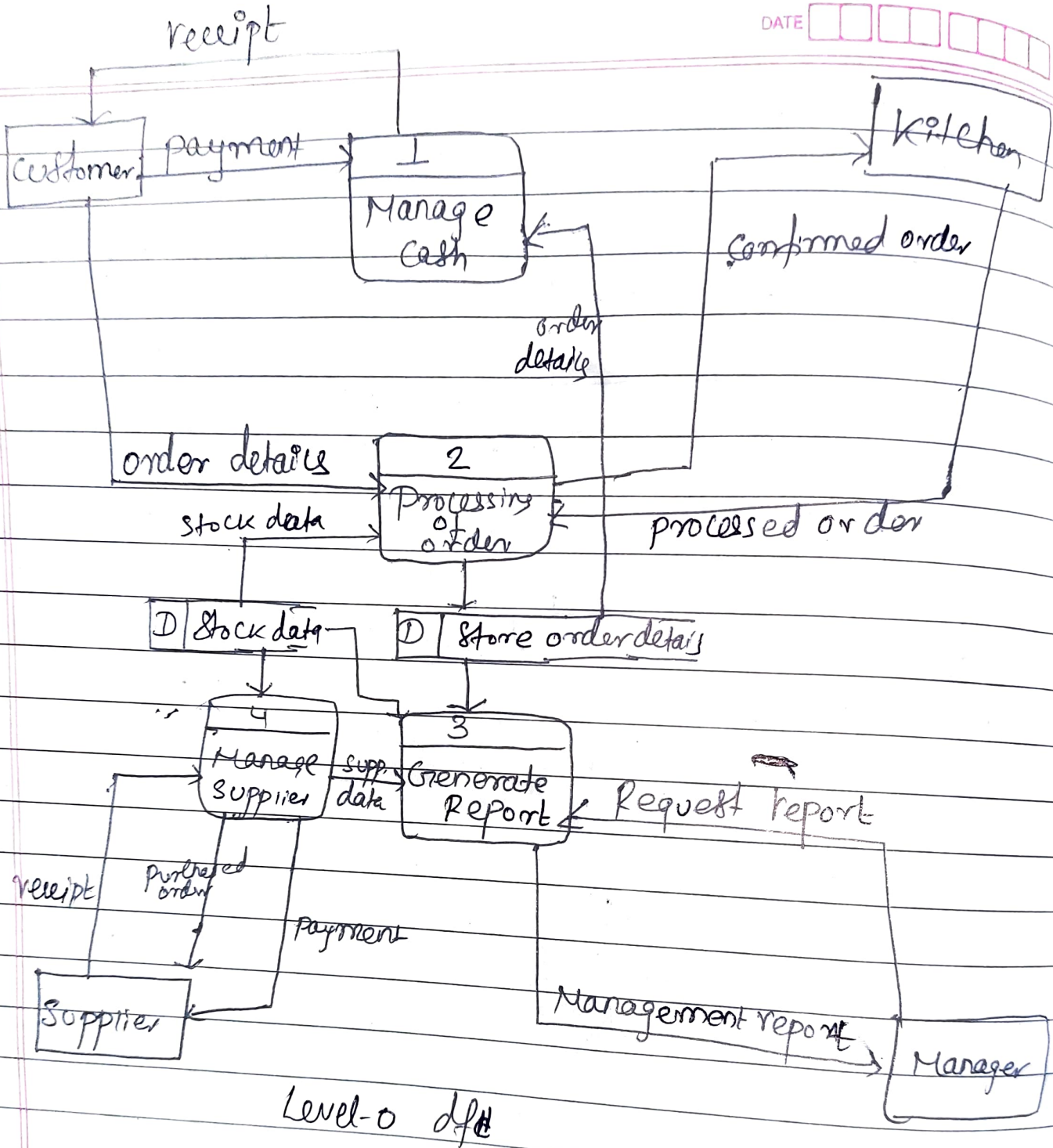
DATE

3. Restaurant Management System



context diagram




receipt
customer
payment
1
Manage cash
kitchen
confirmed order
order details
order details
2
Processing of order
stock data
processed order
D
Stock data
D
Store order details
4
Manage supplier
supp. data
3
Generate Report
Request report
receipt
Purchased order
payment
Supplier
Management report
Manager

Level-0 DFD

DATE

* Modeling logic with decision tables, decision trees and pseudocode

○ Logic Modeling

→ The main disadvantage of DFD is that it does not show the logic inside the process ie. how input data is converted into output transformation.

→ A logic modeling involves representing internal structure and functionality of a process depicted in DFD.

→ Logic model represents the relationship between program activities and intended effects.

→ It has 4 components.
- Needs, Inputs, Activities, Outcomes.

○ Modeling logic with Decision Tables

→ Decision table is a visual (tabular) representation that shows a logical structure with all possible combinations of conditions and resulting actions.

→ They are the algorithms whose output is a set of actions.

→ A decision tables consists of 3 parts: Condition, Actions and rules.

→ Condition are the decisions, actions are the result for the given set of conditions & rules specifies which action are to be followed for the given set of conditions.

classmate

PAGE

DATE

Steps to create decision table.

i) Name the conditions & the values that each condition can assume
ii) Name all possible action that can occur.
iii) List all possible rules.
iv) Define the actions for each rule.
v) Simplify the decision table

→ Advantages: i) Easy to draw
ii) Can be easily changed according to the situation
iii) Requirements become much clearer.

→ Disadvantages:
i) When there are too many alternatives, decision table cannot list them all
ii) Cannot express complete sequence of operations to solve a problem.
iii) They only present partial solution

Modeling logic with decision tree

→ A decision tree is the graphical representation of the conditions, rules & actions of Condns & their alternative actions
→ It shows the logical structure in horizontal form
→ Decision trees & decision tables provide the same results but in different ways.
→ It depicts which conditions to consider 1st, 2nd & so on. ie the relationship of each condition & their actions.
→ In decision tree, conditions are represented in the arc and the actions are represented at the end (ie. leaf node)
Tree is read from left to right.

Classmate

PAGE

DATE

Ex: SALES Promotion policy

Rules

1. Customers who order more than Rs. 1000 will get 5 % discount and additional 5 % discount, if they use our Charge card
2. Customers who don't order more than 1000 will receive Rs. 25 cupon
3. All other customers will receive Rs. 5 cupon

Decision table:

Condition	Rules				
	1	2	3	4	5
1. Preferred customer	T	T	T	T	F
2. Ordered more than 1000	T	T	F	F	f
3 Used our card	T	f	T	F	f
Actions					
1. 5 % discount	X	X			
2. Additional 5% discount	X				
3 Rs. 25 cupon			X		
4. Rs. 5 cupon				X	X

Decision tree

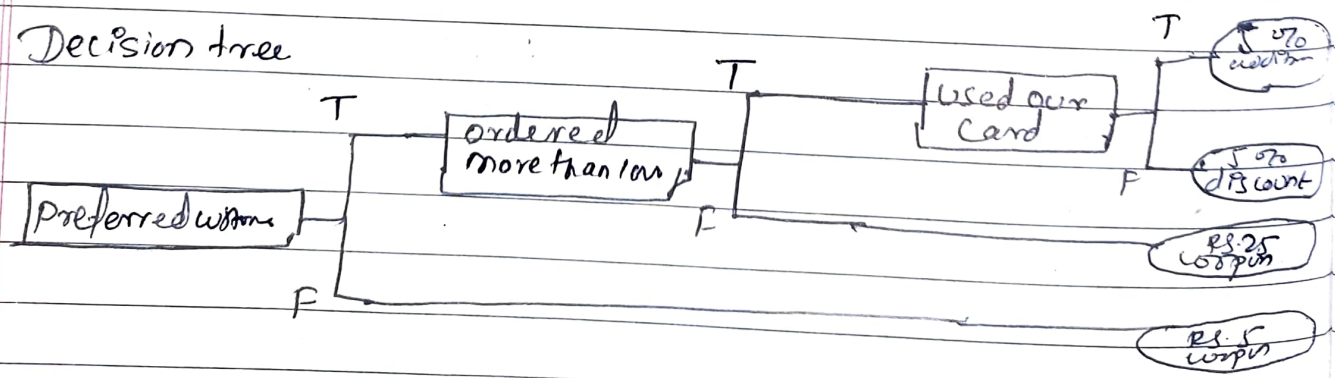


fig: Decision tree.

classmate

PAGE

Note: Action मा T, F, ✓, X जो लगाइन्छ ।

DATE

Eg: A museum has some entry fees
- If age < 5, free entry
- Age > 5 & < 18, Rs. 10 ticket fee
- Age > 18 & < 55, If membership card → fees low
  If not membership card → fees high
- Age > 55, fees low

Decision table

Condition (age)	Rules				
1. < 5	T				
2. ≥ 5 & < 18		T			
3. ≥ 18 & < 55 with card			T	F	
4. > 55					T
Action (charge)					
1. Free	X				
2. Rs. 10		X			
3 Low			X		X
4 High				X	

or

Conditions	Rules				
~~Age~~	< 5	> 5 & < 18	≥ 18 & < 55 with card		> 55
Action: Age	T	T	T	F	T
1. Free	X				
2. Rs. 10		X			
3. Low			X		X
4. High				X	

classmate

PAGE

Decision tree

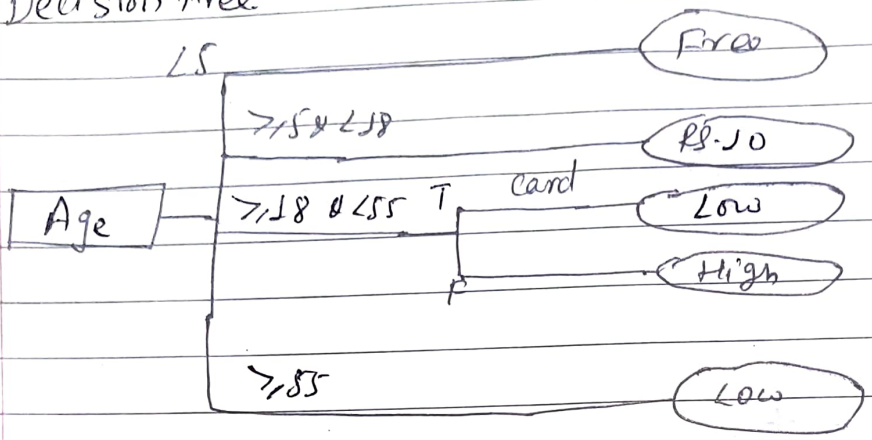


fig: Decision tree.

Ex: ATM.

	Rules		
Conditions	1	2	3
withdrawl ≤ balance	T	F	F
Has credit	-	T	F
Actions			
withdrawl granted	T	T	F

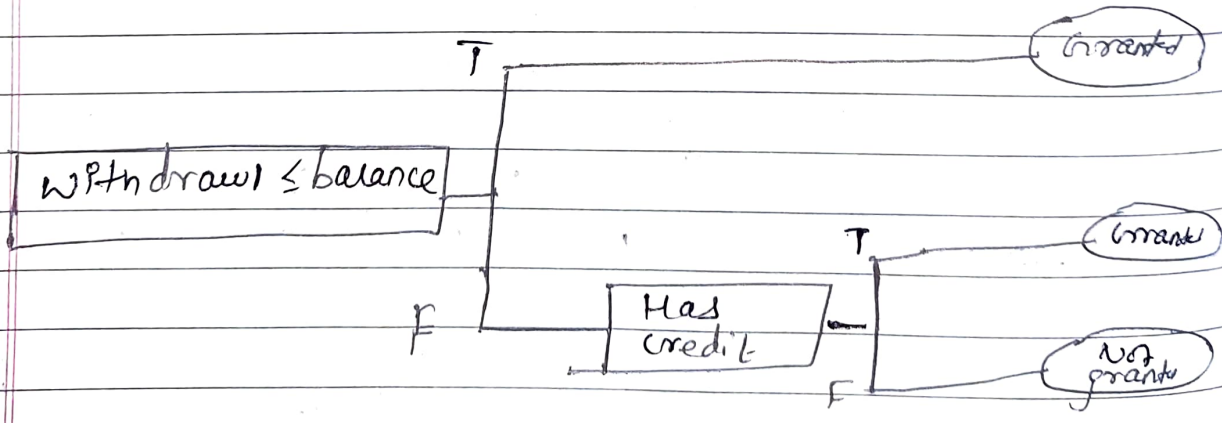


fig: Decision tree for ATM

Modeling Logic with pseudocode

→ Pseudocode is an informal way of programming description that doesnot require any strict programming language syntax.

→ It is used for creating an outline for a program & summarizes a programs flow.

→ It is written in semiformal English language, has no syntax, and can't be compiled by the compiles.

Ex: Factorial.

1. Set Factorial = 1
2. Read N from Keyboard
3. If (N=0) go to step 6
4. Factorial = Factorial x N
5. N=N-1 and then go to step3
6. Print Factorial
7. End.

→ Advantages of pseudo code

i) Acts as a bridge between program & the algo. or flowchart
ii) Simple & easy to understand. for non programmers.
iii) Can be used as blueprint of the program
iv) Can be easily converted into any programming language, making it a versatile tool for software development.

DATE

3.3 Structuring System Data Requirements.

Structuring system data requirements concentrates on the definition, structure and relationship within the organizations data. The most common format used is ER diagram

* 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.

→ During the conceptual data modeling, a modeler works with stakeholders to identify the key entities (such as customers, orders, and products) that are relevant to the system, and the relationships between them.

(NOR: CDM is made for both existing & new systems).

→ Then the modeler creates a conceptual data model, typically using a tool such as ER diagram, that captures these entities & relationships in a graphical form

→ Conceptual data model ensures that the data is organized in a way that is efficient & easy to maintain.

→ Then the conceptual data model is used as the basis for more detailed logical & physical data models that will be used to implement the system

→ The primary deliverables from the conceptual data modeling is the ER diagram. Another deliverable from CDM is a full set of entries about the data objects to be stored in the project repository.

CLASSMATE

PAGE

→ There are two ways of gathering information for DB

- Top-down approach → begins with the high-level views of the system. It derives the business rules for data models and intimate understanding of business

- Bottom-up approach :- It is derived by reviewing specifications and business documents.

* Introduction to E-R modeling

→ ER modeling is the most widely used method ~~for~~ in Conceptual data modeling for structuring the system data requirements.

→ ER model is a high level graphical data model describes the design of database in terms of entities and relationship among them. It represents the real-world objects.

→ Elements of ER diagram

1) Entity : Entity is a thing in real world that has a physical existence. It can be a person, place, object or even a concept which stores data in database. Each entity is made up some attributes which represents that entity. Ex: Let 'Ankit' be a particular member of entity type 'student'. Then, the tuple of Ankit has details id, name, age, which exists in real life, so it is an entity.

NOR: Keys → used to uniquely identify any record or row of data from the table. Also used to establish & identify relationship between tables

Entity type → The collection of entities having common attributes. Ex: student is a entity type, where all the entities in the student have attributes S_id, age, Address, etc & so on.

Entity set → The collection of entities of a particular entity type in the database at any particular point of time is called entity set. It is the subset of entity type. In an entity type student, the set of students whose age>30 is an entity set

ii) Attributes

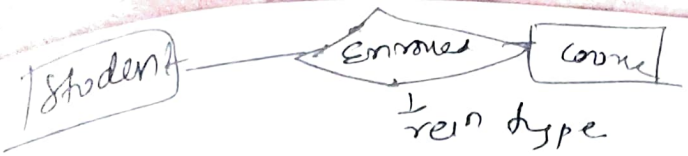
Attributes are the components of entity and are mostly table. They are the descriptive property possessed by each member of an entity set or entity type. ex: In an entity type student, the attributes can be Std_id, Std_name, Std_age, In ER diagram, attributes are represented by ovals.

Types of attribute:

→ Atomic vs composite attribute
→ Single valued vs multivalued attribute
→ Stored vs derived attribute
→ Null valued attribute
→ key attribute.

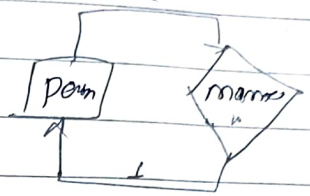
iii) Relationships

Relationships are the association among entities. A relation type R among n entity types E1, E2, ---, En defines a set of associations among entities from entity types.

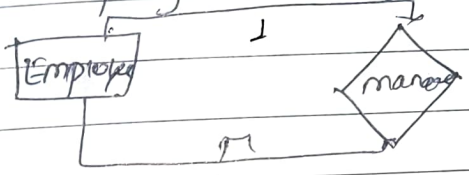


Degree of relationship :- No. of entity sets that participate in a relationship set is called the degree of relationship set. On the basis of degree, relationship can be divided as:

i) Unary relationship : only one entity set participates in rel.
→ One to one relationship: (unary)
Ex: a person married to a person.

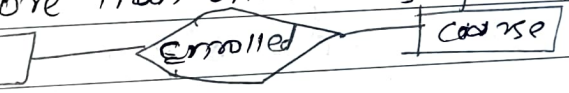


→ One to many (1:M) unary relationship
Ex: an employee manages many employees.

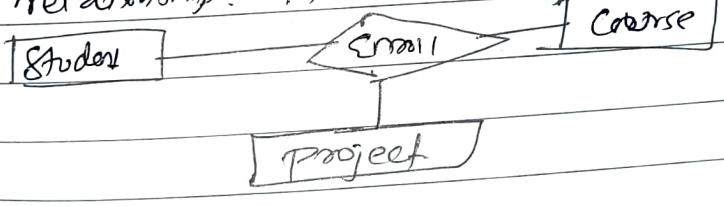


→ Many to Many (M:N) unary relationship

ii) Binary relationship: More than one entities (mostly two) sets participating in a relation. Ex:



iii) N-ary relationship: n entities sets participating in a relation (n>2)



iv) 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 & many-to-many. ER diagrams are used to describe the cardinality in databases.

DATE

V) Naming Conventions.

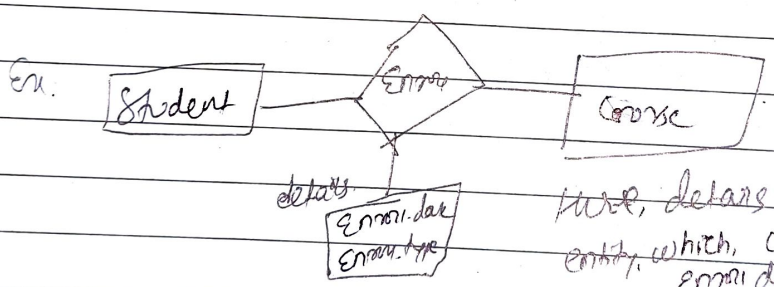
When designing a database schema, the choice of names for entity types, attributes, relationship types and roles is not straight forward.

↳ one should choose the 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

↳ Entity type & relationship type names are upper case letters, attribute names have their initial letter capitalized & role names are lower case letters.

VI) Associative Entity :- Associative Entities are connections that describe a relationship between two different entities They are used mostly in many to many relationships



Here, details is an associative entity, which, contains attribute Enrol date, --

VII) Constraints on ER model / structural constraint in ER

Relationship sets in ER model usually have certain constraints that limit the possible combination of entities that may involve in corresponding relationship set. The most important constraints in ER are

→ Mapping constraints (cardinalities)
→ Participant Constraints

- Mapping constraints (cardinalities)
  => The number of times an entity of an entity set participates in a relationship set is called mapping cardinalities Types

i) One-to-one: Each entity in an Entity set can participate only once in the relationship, then the cardinality is one to one. Ex:- one man marries one woman

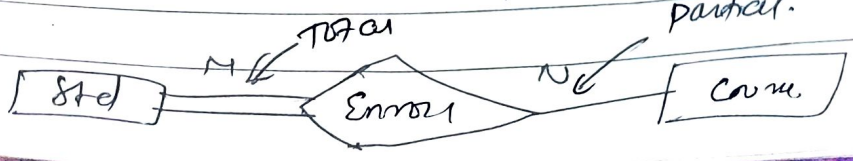
ii) one to many or many to one:
  when entities in one entity set can participate only once in the relationship set & entities in other entity set can take part more than once. Ex: one student can take many course, & one course can be taken by many students

iii) Many to Many: Entities in all entity sets can take part more than once in the relationship. Ex:- many course can be taken by many students.

- Participant constraints
  It determines whether all or only some entity occurance participates in a relationship. Two types

i) Total participation constraint
  => Here each (all) entity in the entity set must participate in the relationship. Ex: Each student must enroll in a course. It is shown by double line

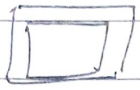
ii) Partial participation constraint
→ The entity in the entity set may or may not participate in the relationship



DATE

~~ER Construct~~ Symbols used.

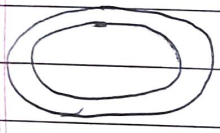
 → Entity type

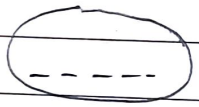
 → weak entity type

 → relationship type

 → attribute.

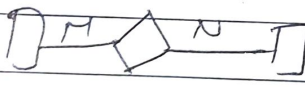
 → key attribute.

 → Multivalued attribut.

 → partial key attribut.

 → derived attribute.

 → Cardinality ratio one to many

 → Many to Many

 : Total participation

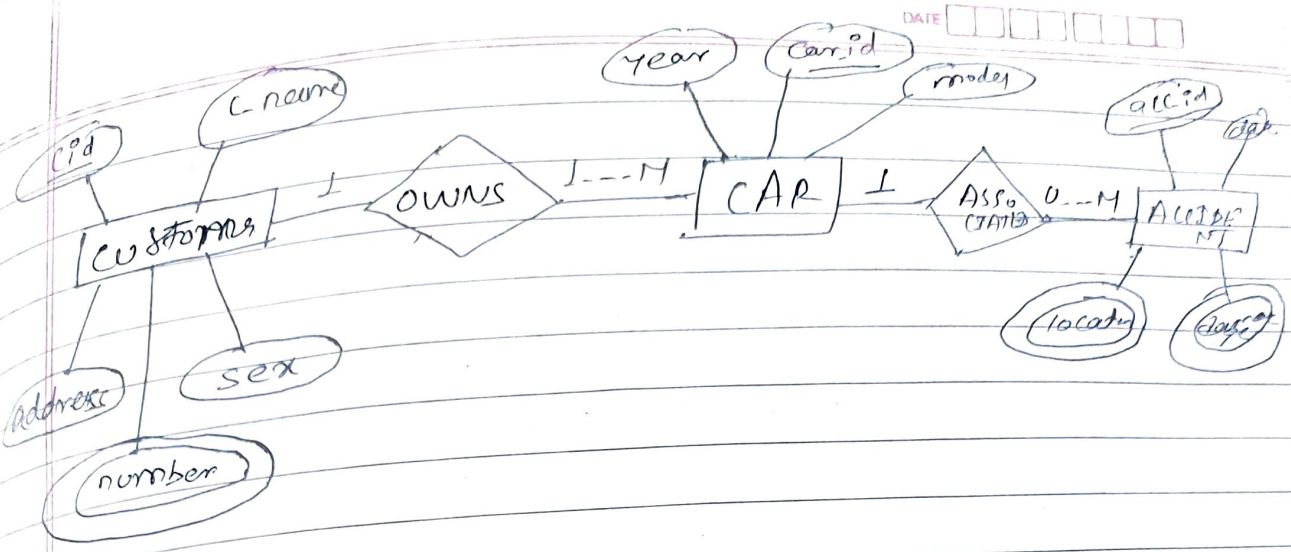
 → relationship with weak entity type

Weak Entity type has total participation

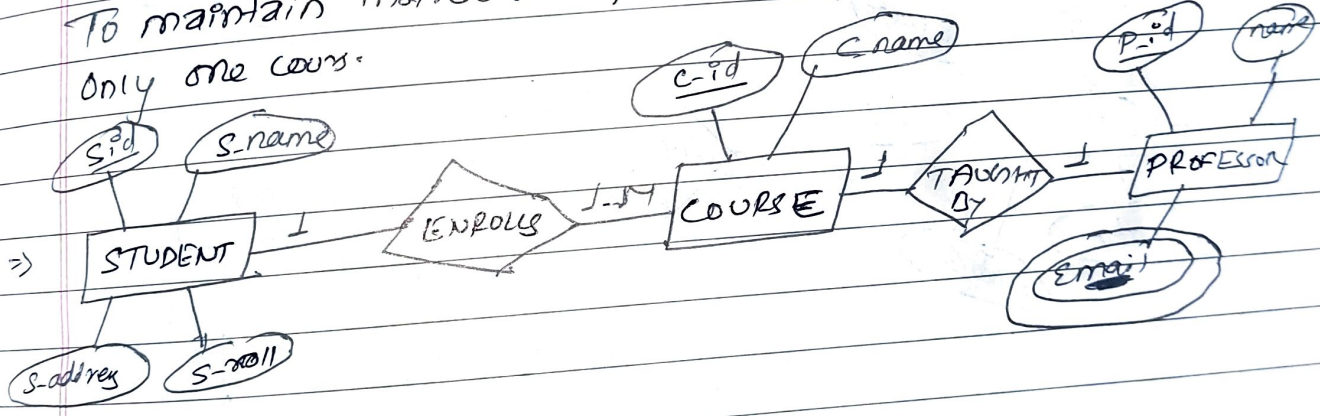
Ex: Construct an ER diagram for a Car-Insurance Company whose customers own one or more cars each. Each car has associated with it zero to any number of recorded accidents.

classmate

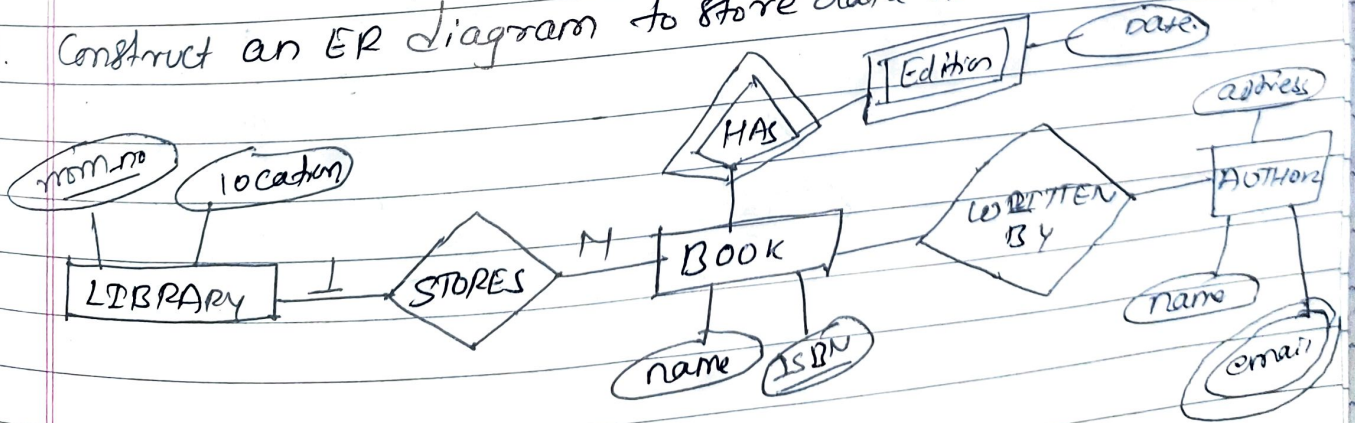
PAGE



Ex. In a university, a student enrolls in a course. A student must be assigned to atleast one or more courses, each course is taught by a single professor. To maintain instruction quality, a professor can deliver only one cours.

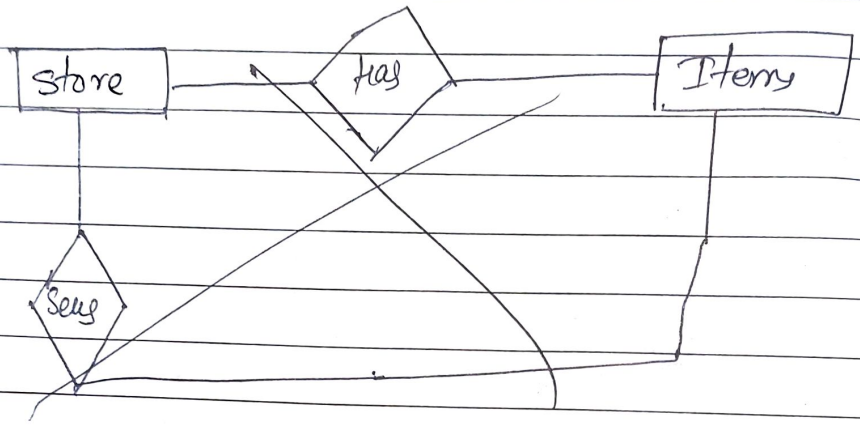


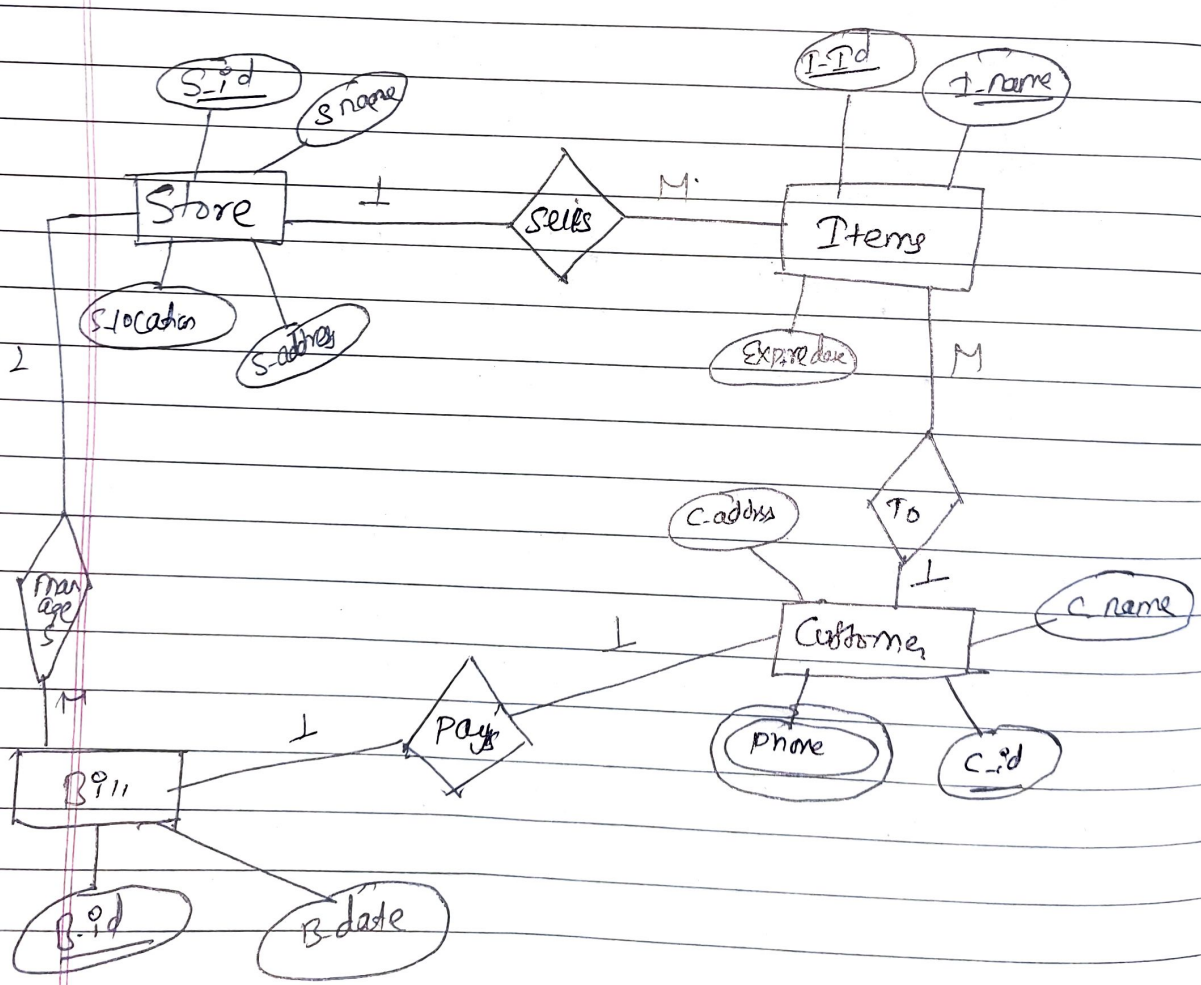
Q. Construct an ER diagram to store data in library of college



DATE

Q.2078 Draw an ER diagram for retail store in a mall which sells different items to its customers





classmate

PAGE