mood-book

DATA BASE MANAGEMENT SYSTEMS.

UNIT - I.

Introduction :-

→ A database - Management System (DBMS) is a collection of interrelated data and a set of programs to access that data.

→ The collection of data, is usually referred to as the database, contains information relevant to an organization.

→ The primary goal of a DBMS is to provide a way to store and retrieve database information that is both convenient and efficient.

→ DB systems are designed to manage large bodies of information.

→ Management of data involves both defining structures for storage of information and providing mechanisms for manipulation of information.

→ DB system must ensure the safety of the information stored, despite system crashes or attempts at unauthorized access.

Database - System Applications.

→ Databases are widely used.
Some of its applications are:-

1) Banking: For Customer information, accounts, loans and banking transactions.

2) Airlines: Airlines were among the first to use databases in a geographically distributed manner. They are used for Reservations and Schedule information.

3) Universities: For maintaining student information, Course registrations and grades.

4) Credit Card-Transactions: For purchases on credit cards and generation of monthly statements.

5) Tele Communications: For keeping records of calls made, generating monthly bills, maintaining balances on prepaid calling cards, storing information about the communication networks.

6) Finance :- For storing info. about holdings, sales, and purchases of financial instruments such as stocks and bonds, also for storing real-time market data to enable on-line trading by customers and automated trading by the Organization.

7) Sales :- For customer, product, and purchase information.

8) On-line retailers : For sales data plus on-line order tracking, generation of order lists and maintenance of on-line product evaluations.

9) Manufacturing : For management of the supply chain and for tracking production of items in factories, inventories of items in warehouse and stores and orders for items.

10) Human Resources : For info. about employees Salaries, payroll taxes, benefits and for generation of paychecks.

→ ATM let users interact directly with database

→ The Internet Revolution of the late 1990s increased direct user access to databases.

→ Although, User interfaces hide details of access to a database, and most people are not even aware they are dealing with a database, accessing databases form an essential part of almost everyone's life today.

A HISTORICAL PERSPECTIVE.

→ The first general-purpose DBMS, designed by Charles Bachman, working at General Electric (GE) in 1960's.

→ It formed the basis for the Network model.

→ Bachman received ACM's Turing Award for work in database area in the year 1973.

→ In late 1960's, IBM developed the Information management System (IMS) DBMS.

→ IMS formed the basis for Hierarchical data model.

→ It allowed several people to access the same data through a computer network.

→ In 1970, Edgar Codd, at IBM's San Jose Research lab, proposed a new data model called Relational data model, which proved to be successful in the development of Database systems.

→ In 1981, Codd won the Turing award for his work.

→ In 1980's, the relational model became dominant in DBMS paradigm, and database systems gained widespread use.

→ The SQL query language for relational databases, developed as part of IBM's System 'R' project, is now the standard Query language.

→ In the late 1980's and the 1990's, advances were made in many areas of database systems.

→ In Early 1990's, the SQL language was designed primarily for decision support applications.

→ In Late 1990's, the major event was explosive growth of world wide web (WWW).

→ Databases were much more extensively used than before.

→ Database systems now had to support very high transaction processing rates, high reliability and 24×7 availability.

→ Early 2000's, saw the emerging of XML and the associated query language XQuery as a new database technology.

→ This period also witnessed the growth in "autonomic Computing / auto admin" techniques for minimizing system administration effort.

FILE SYSTEM VERSUS A DBMS.

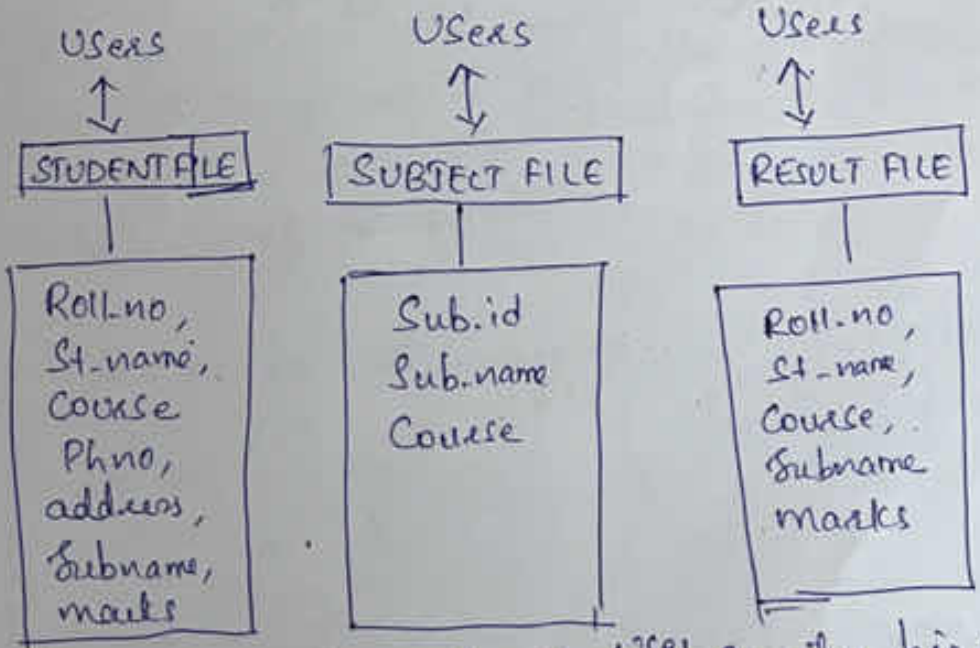
→ A file system is a technique of arranging the files on a storage medium like a harddisk, Pendrive, DVD etc.

→ It is a traditional approach, which and is supported by an operating system like Windows, Linux etc.

→ The users create various files and store permenantly in their systems, and write different application programs to extract records from and add records to the appropriate files.

→ Files can be collectively present in a directory. Directories can again be present inside another directory giving it a hierarchical structure.

Ex:-

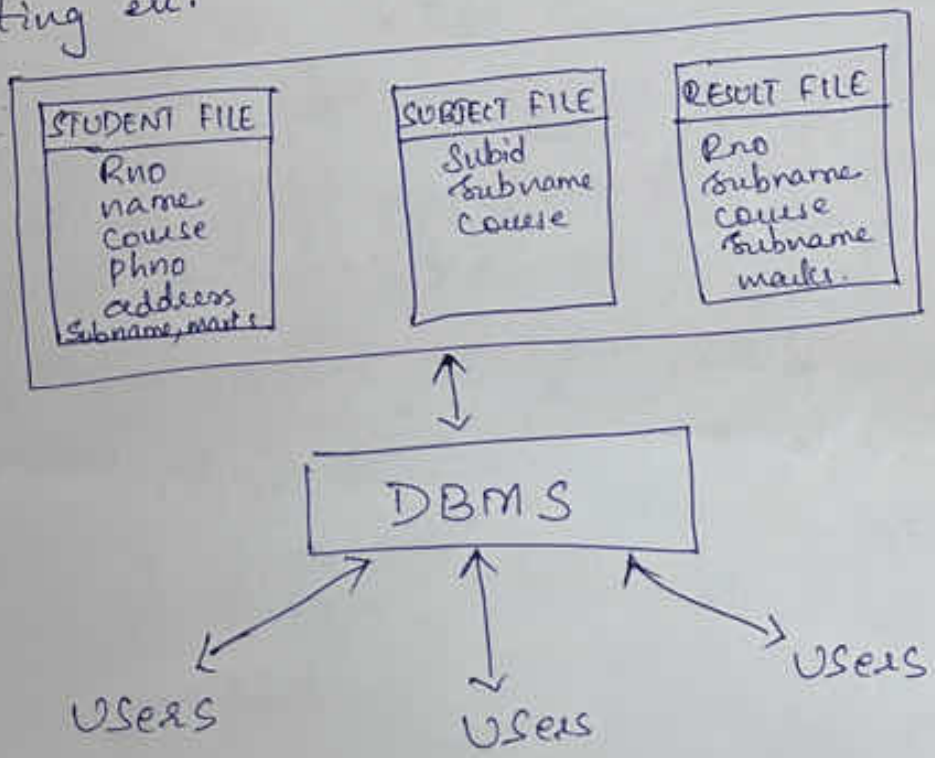


→ In the above example, each ~~student~~ user creates his own file and stores data. Some fields are duplicated in more than one file, leading to data redundancy.

→ A Database approach is a Organised collection of data, that are related in a meaning ful way and which can be accessed by different users but stored only once in a system.

→ A DBMS allows users to perform various operations on the database such as insertion, deletion, Searching, Sorting etc.

Ex:-



The following are the differences between DBMS and File System.

Basis	File System approach	DBMS approach
1. Meaning	The file system is a collection of data. A user has to write the procedures for managing the database.	DBMS is a collection of data. In DBMS the user is not required to write the procedures.
2. Sharing of data.	Data is distributed in many files and files may be different formats. So, it isn't easy to share data.	Due to the centralized approach, data sharing is easy.
3. Data Abstraction	The file system provides the detail of the data representation.	DBMS gives an abstract view of data and hides the details.
4. Data Redundancy and Inconsistency	The files and application programs are created by different users or programmers So that there exists a lot of data duplication which leads to inconsistency	Due to centralization of the database, the problems of data redundancy and inconsistency are controlled.
5. Data Independence	There is no Data Independence	The seperation of data descriptions from the application programs gives data independence. It is of 2 types. 1) Logical data Independence 2) Physical data Independence

Basic	File System approach	DBMS approach.
6. Security and Protection	It is not easy to protect a file. Password given to be a file can be easily broken(or) hacked.	DBMS provides a Role based access to the data in the dbms and hence is a good protection mechanism
7. Recovery Mechanism	If the system crashes before the user saves the file, then the content of file cannot be recovered.	DBMS protects the user from system failure by providing crash recovery mechanism
8. Concurrency Problems.	These system doesn't offer Concurrency.	DBMS takes care of concurrent access of data by using the locking mechanism.
9. Integrity Constraints	Difficult to implement in file system.	Integrity Constraints are easy to apply.
10. Structure	The file system approach has a simple structure	The database structure is complex.
11. Query Processing	Users have to write application programs	Users can easily query the data in the database using SQL.
12. Flexibility	Not flexible.	Changes are necessary to the data, stored in any system. Changes can be done easily with a DB approach.
13. Examples	C++, COBOL etc	Oracle, mySQL, Sybase etc.

THE DATA MODEL.

→ The basic structure of a database is the datamodel.

→ A data model is a collection of high-level data description constructs for describing data, data relationships, data semantics and consistency constraints.

→ A DBMS allows a user to define the data to be stored in terms of data model.

→ Most DBMS today are based on Relational data model.

→ A data model hides many low-level storage details.

→ A Semantic data model is a more abstract, high-level data model that makes it easier for a user to come up with a good initial description of data in an enterprise / organisation.

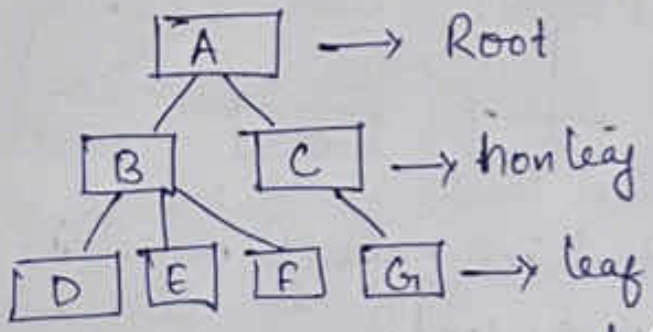
→ These models help describe a real application Scenario.

→ A database design in terms of a semantic model serves as a useful starting point and is subsequently translated in to DB design.

→ A widely used semantic data model called the Entity-Relationship model (ER model).

2) Hierarchial Database Model:-

→ It stores data in the form of hierarchy.



→ Relationship between one record to another record exists.

→ Relationship is formed using pointers (i.e., the address of the next node).

→ It achieves a 1 to many relationship.

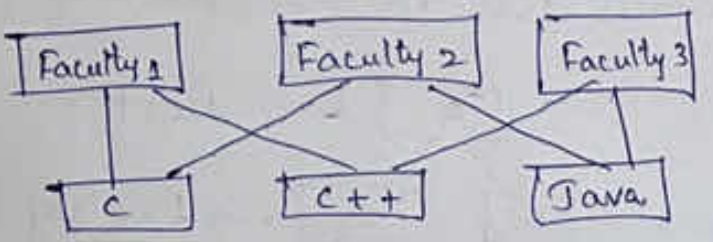
Advantages:-

1) We can form Relationships.
2) No Redundancy of data.

Disadvantages:-

1) Data cannot be entered in the middle if the structure is formed i.e., insertions & deletions are difficult.
2) To Enter data (if) total structure is disturbed.

2) Network Database Model :-



→ The data stored in network model is complex and difficult to understand.

→ It achieves Many - Many relationship.

→ One node can be related to any other node.

→ Pointer are used for this purpose.

→ Disadvantages.

1) Memory allocation problem.
2) Relationships are through the address only.
3) It is not user friendly.
4) If any structure is to be changed all the related ones are to be changed.

3). The Relational Model :-

→ The Central data description Construct in this model is a Relation (table), which can be thought of as a Set of Records.

→ A description of data in terms of a data model is called a Schema.

→ In the relational model, the Schema for a relation Specifies its name, the name of each field (or attribute or Column), and the type of each field.

Ex:- Student (Sid: integer, name: String, gpa: real, login: String, age: integer)

① The ~~following~~ above schema Says that each record in the Student relation has five fields, with field names and types as indicated.

② An example instance of the Student's relation is as follows:

Sid	name	gpa	login	age
53666	Jones	3.4	jones@cs	18
53688	Smith	3.2	Smith@ee	18
53650	Smith	3.8	Smith@math	19.
53831	Madayan	1.8	madd@music	18
53832	Guldu	2.0	guldu@music	18.

→ Each row in the Student's relation is a record (or) tuple that describes a Student.

→ Each columns in the relation is a field (or) attribute that describes the properties of student.

→ We can Specify integrity constraints, which are conditions that the records in a relation must satisfy.

Ex:- Every Student has a unique sid value.

i.e., Primary key Constraint.

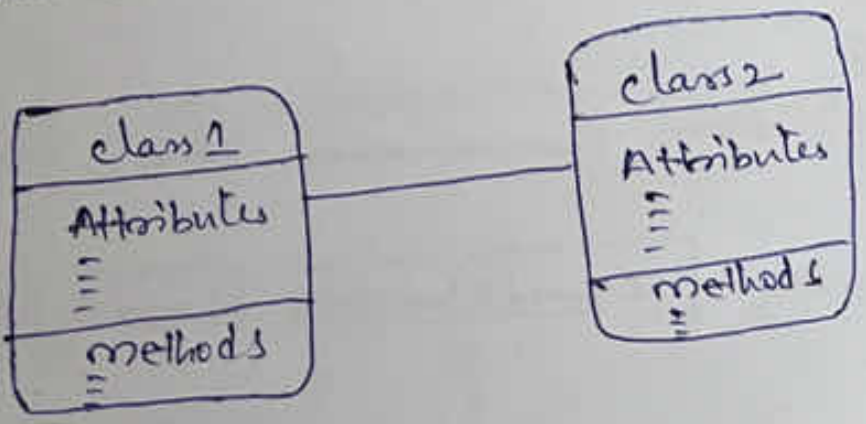
8xxx5.

4) Object - Based Data Model:-

→ An object oriented data model, attributes and methods that operate on those attributes are encapsulated in structures called classes.

→ The major advantage of this data model is that Complex datatypes like graphics, videos and sound are supported as easily as simple datatypes.

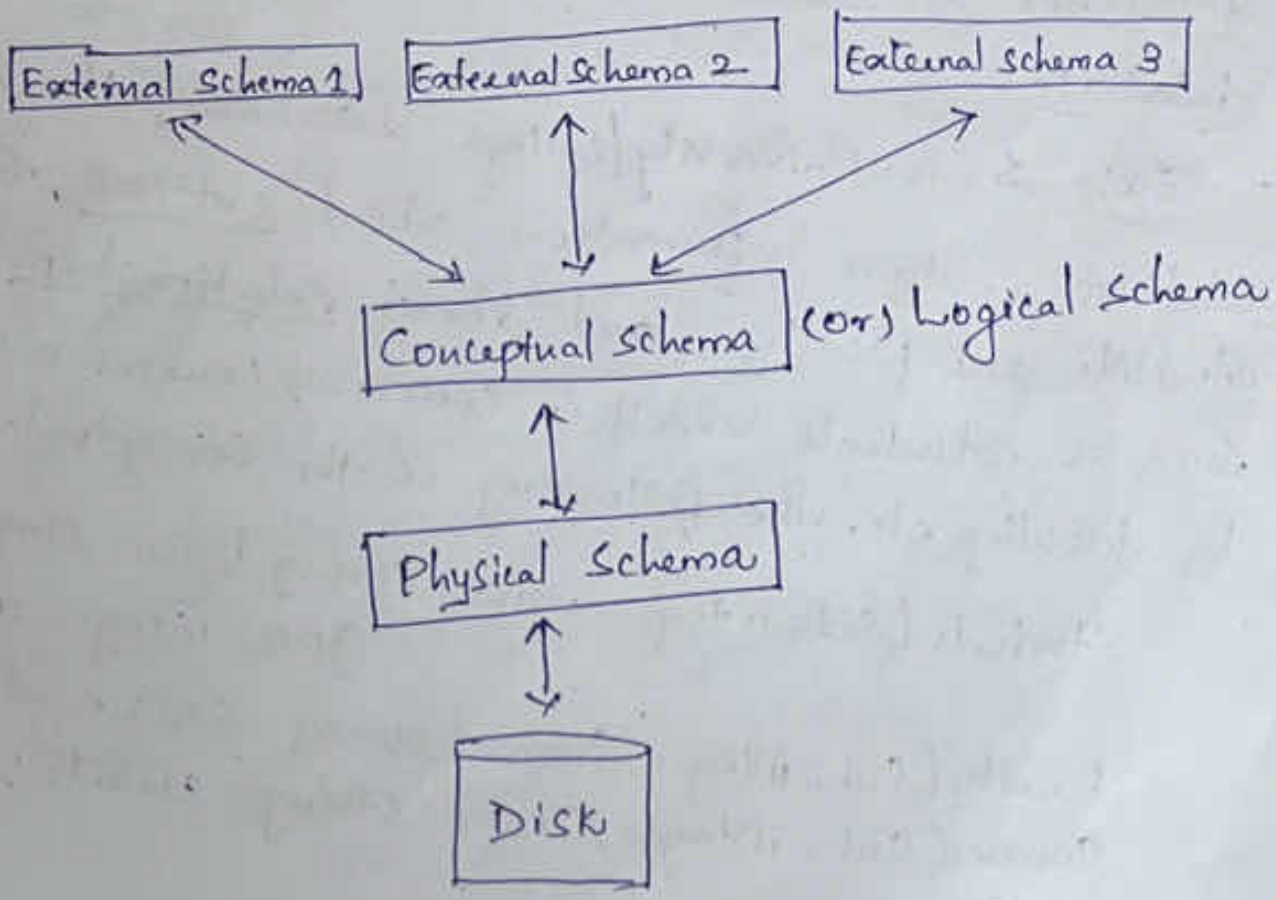
→ The object - relational data model combines features of object oriented data model and relational data model.



i.e., Primary key constraint.

8***5.

2) Levels of Abstraction in a DBMS



Levels of Abstraction in a DBMS

→ Data Abstraction is a process of hiding unwanted or irrelevant details from the end user.

→ The data in a DBMS is described at three levels of abstraction as in the above figure.

→ The database description consists of a schema at each of these three levels of abstraction:
the Conceptual, Physical and external.

① Conceptual Schema :- It is also called the logical schema, describes the stored data in terms of the data model of the DBMS.

② In a relational DBMS, the conceptual schema describes all relations that are stored in the database.

Ex:- In a University/College database, these relations contain information about entities, such as students and faculty, and about relationships, such as students enroll in courses, courses are taught by faculty etc. The following is the conceptual schema:

Students (Sid: integer, name: string, login: string, gpa: integer, age: integer)

Faculty (fid: integer, fname: string, sal: real)

Courses (Cid: integer, Cname: string, credits: integer)

etc.

→ Identifying the relation, and the fields for each relation is important and the process of arriving at a good conceptual schema is called Conceptual database design.

② Physical Schema :- The physical schema specifies the storage details.

→ The physical schema specifies how the relations described in the conceptual schema are actually stored on secondary storage devices such as disks and tapes.

→ We must decide what file organizations (sequential, heap, Indexed, Tree etc) to use to store the relations and create data structures called indexes, to speed up data retrieval operations.

Ex:- → Store all relations as unsorted file of records.
→ Create indexes on the first column of student, faculty etc.

→ The processing of arriving at a good physical schema is called physical database design.

2** Data Independence

→ An important advantage of using a DBMS is that it offers data Independence.

→ That is, application programs are insulated from changes in the way the data is structured and stored.

→ Data Independence is achieved through use of the three levels of data abstraction.

→ Relations in the external schema (views) are generated on demand from the relations in the conceptual schema.

→ Thus, users can be shielded from changes in the logical structure of the data, or changes in the choice of relations to be stored. This property is called logical data independence.

→ The conceptual schema hides users from changes in physical storage details. This property is referred to as physical data independence.

→ The conceptual schema hides details such as how the data is actually stored on disk, the file structure and the choice of indexes.

→ The conceptual schema remains the same, we can change these storage details without altering applications.

(3) External Schema:- External Schema, allow data access to be customized (and authorized) at the level of individual users or groups of users.

→ Any given database has exactly one Conceptual Schema and one physical Schema, but it may have several external schemas, each to a particular group of users.

→ Each external Schema Consists of a collection of one or more Views and relations from the Conceptual Schema.

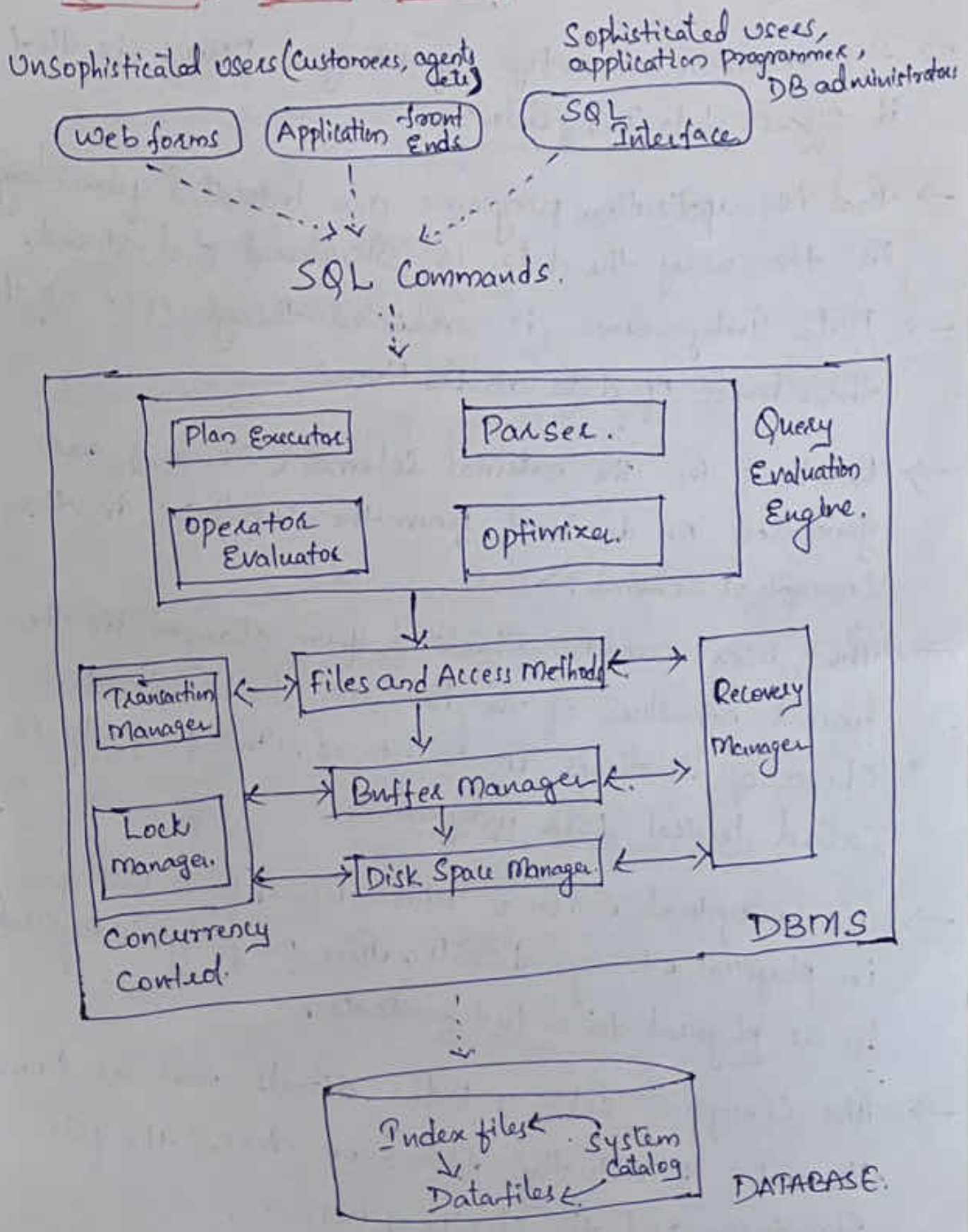
→ The external Schema design is guided by end User requirements.

Ex:- We might want to allow students to find out the names of faculty members teaches courses and the course credit. This can be done by defining the following View:

CourseInfo (Cid: String, Fname: String, credit: integer).

→

Structure of a DBMS.



→ The DBMS accepts SQL commands generated from different user interfaces, produces query evaluation plans, executes these plans on the database, and returns the answers.

→ When a user issues a query, the parsed query is presented to a query optimizer, which uses information about how the data is stored to produce an efficient execution plan for evaluating the query.

→ Query parsing is the first step in query processing, in this step, a query is checked for syntax errors, then it is converted in to parse tree, which is easy for DBMS to understand.

→ An execution plan is a blueprint for evaluating a query, usually represented as a tree of relational operators. Relational operators serve as a building blocks for evaluating queries posed against the data.

→ The files and access methods layer code sits on top of the buffer manager, which brings pages in from disk to main memory.

→ Disk space manager, deals with management of space on disk, where data is stored. Allocate, deallocate, read and write pages is provided through this layer.

→ The DBMS supports concurrency and crash recovery by carefully scheduling user requests and maintaining a log of all changes to the database.

→ Transaction Manager, ensures that transactions request and release locks according to a suitable locking protocol.

→ Lock Manager, keeps track of requests for locks and grants locks on database objects when they become available.

→ Recovery Manager is responsible for maintaining a log and restoring the system to a consistent state after a crash.

→ The disk space manager, buffer manager and file and access method layers must interact with these components.

DATABASE DESIGN AND ER DIAGRAMS

→ Database design is a central part in data-intensive applications, of a larger Software System design.

→ The database design process can be divided into Six steps. The ER model is most relevant to the first three steps.

1) Requirements Analysis:-

(*) The first step in designing a database application is to understand - what data is to be stored in the database, what applications must be built, what operations are most frequently performed and subject to performance requirements.

(*) This is usually an informal process that involves discussions with user groups, study of current operating environment, how it is expected to change etc.

(*) Several methodologies have been proposed for organizing and presenting the information gathered in this step, and some automated tools have been developed to support this process.

2) Conceptual Database Design:-

(*) The information gathered in requirements analysis step is used to develop a high-level description of the data stored in the D.B.

(*) This step is often carried out using the E-R model.

(*) The E-R model is a high-level data model used in database design.

(*) The goal is to create a simple description of the data that shows how users and developers think of the data.

3) Logical Database Design :-

(*) We must choose a DBMS to implement our database design, and convert the conceptual database design into a database schema (structure) in the datamodel of the chosen DBMS, [i.e., Relational datamodel, network, hierarchical, etc]

(*) We consider only relational DBMS and therefore, the task in the logical design step is to convert an ER schema into a relational database schema.

Beyond ER Design

→ The ER diagram is an approximate description of the data, constructed through evaluation of information collected during requirement analysis.

→ Once we have a good logical schema, we must consider performance criteria and design the physical schema.

→ Finally, we must address security issues and ensure that users are able to access the data they need.

4) Schema Refinement :-

(*) The fourth step in database design is to analyze the collections of relations in our relational database schema to identify potential problems and to refine it.

5) Physical Database Design :-

(*) This step involves building indexes on some tables and clustering some tables or it may involve redesign of parts of the database schema obtained from earlier design steps.

(*) In this step, we consider workloads that database must support and further refine the database design to ensure that it meets desired performance Criteria.

6) Application and Security Design :-

(*) Any software project that involves a DBMS must consider aspects of the application.

(*) we must identify the entities (e.g. users, user groups, departments etc) and processes involved in the application.

→ we must describe the role of each entity in every process.

→ For each role, we must identify the parts of the database that must be accessible and the parts that must not be accessible.

→ We must ensure that these access rules are enforced, for security of data in database.

ENTITIES, ATTRIBUTES AND ENTITY SETS

Entities :- An entity is a person, place, object, event or concept in the User Environment about which the Organization wishes to maintain data.

Ex:-

Person: EMPLOYEE, STUDENT, PATIENT
Place: STORE, WAREHOUSE, STATE
Object: MACHINE, BUILDING, AUTOMOBILE
Event: SALE, REGISTRATION, RENEWAL.
Concept: ACCOUNT, COURSE, WORK CENTER.

Entity type (or) Entity Sets

An entity type is a collection of similar entities that share common properties or characteristics

Attributes :- Attributes define the properties or characteristics of an Entity.

→ All entities in a given entity set have the same attributes.

→ For each attribute associated with an entity set, we must identify a domain of possible values.

Ex:- The domain associated with the attribute name of Employees might be the set of 20-character strings.

→ For each entity set we choose a Key.

→ A Key is a minimal set of attributes whose values uniquely identify an entity in the set.

Some of the important keys are:-

1) Primary Key:- A primary key is an attribute (or combination of attributes) that uniquely identifies each row in a table (or) relation.

The primary key is designated by underlining the attribute name.

Ex:- EMPLOYEE (Eid, Ename, Dept_name, Salary)

Eid is the primary key for the table (or) relation EMPLOYEE.

The EMPLOYEE entity set is represented by a rectangle, and an attribute is represented in a oval. The Primary key attribute is underlined.

RELATIONSHIPS AND RELATIONSHIP SETS

→ Relationship :- A relationship is an association among two or more entities.

The Relationship is represented by using a rhombus symbol which is enclosed between two entities with the type of relation among them.

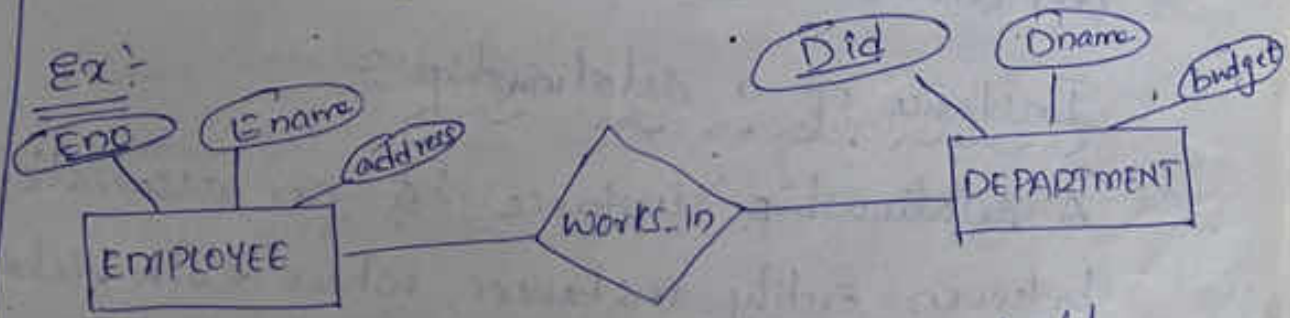
→ Relationship Set :- Collection of a set of similar relationships is called as relationship set.

A relationship set can be considered as a set of n-tuples.

$$\{(e_1, \ldots, e_n) \mid e_1 \in E_1, \ldots, e_n \in E_n\}$$

Each n-tuples denotes a relationship involving n entities e_1 through e_n, where entity e_i is in entity set E_i.

Ex :-



In the above example works_in is the relationship set in which each relationship

indicates a department in which an employee works.

→ Descriptive Attributes (or) Attributes on Relationships :-

(*) Sometimes, a relationship can also have attributes. These attributes are called descriptive attributes.

(*) Descriptive attributes are used to record information about the relationship, rather than about any one of the participating entities.

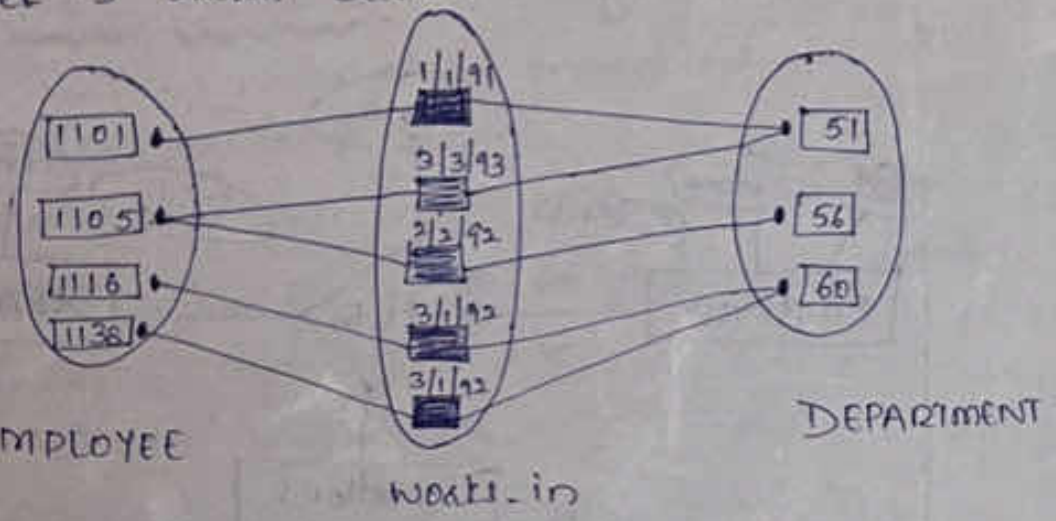
Ex:-

Suppose, we wish to record from when an Employee is working in a department, then this information can be captured by adding an attribute "Since" to "works-in".

Instance of a relationship :-

A relationship instance is an association between entity instances, where each relationship instance includes exactly one entity from each participating entity type.

Ex:- An instance of the works_In relationship set is shown below.



Employee entity is denoted by its Eid, and each departments entity is denoted by its Did. The Since value is shown beside each relationship. Each line represents a relationship instance between one employee and one Department.

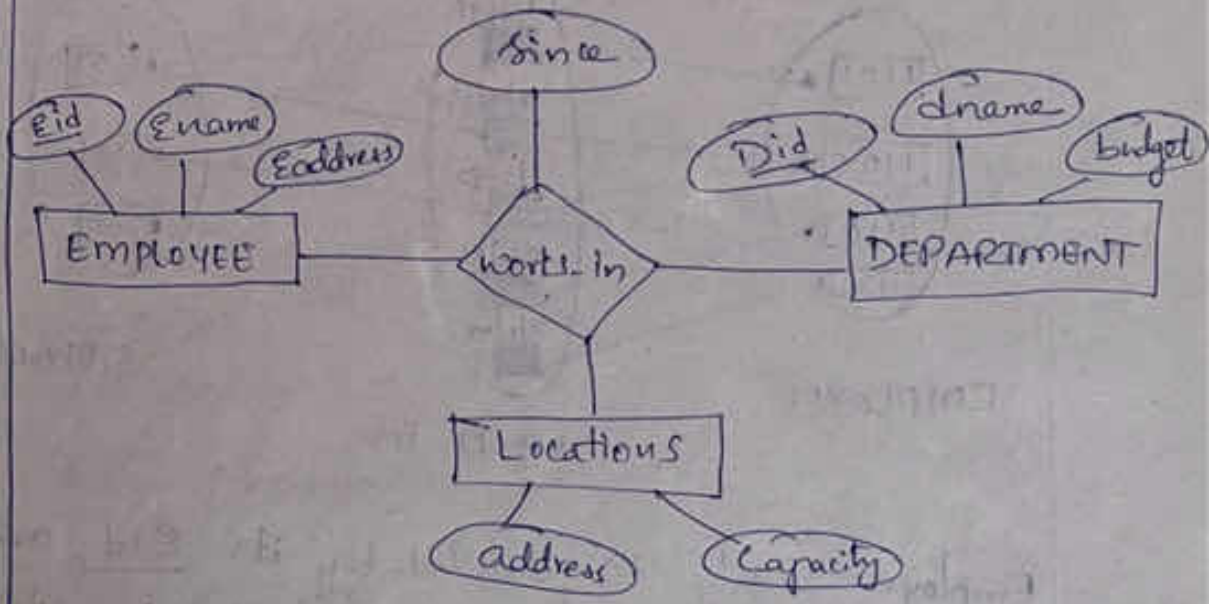
Ternary Relationship:-

"A ternary relationship is a simultaneous relationship among the instances of three entity types".

Ex:- Suppose each department has offices in several locations and we want to record the locations at which each employee works.

This relationship is ternary because we must record an association between an employee, a department and a location.

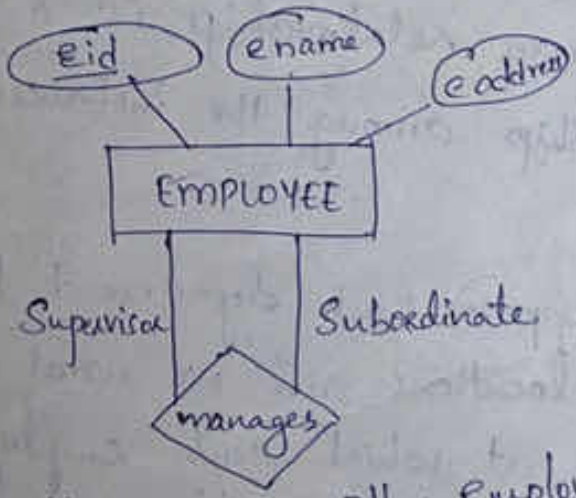
The ER diagram for this is



Unary Relationship :-

"A unary relationship is a relationship between the instance of a single entity type", i.e., Sometimes a relationship might involve two entities in the same entity set.

Ex :-



Since, Employee manages other employees, every relationship in manages is of the form (emp_1, emp_2) where both emp_1 and emp_2 are entities in EMPLOYEE. However, they play different roles which are indicated by role indicators. Supervisor & Subordinate

ADDITIONAL FEATURES OF ER MODEL

→ Key Constraints

(*) The restrictions placed on entities is called as constraints.

Ex:- The works_in relationship in the previous example - An employee can work in several departments, and a department can have several employees.

Ex:- Consider an Example.

In the above example the relationship set called manages between the Employee and Department entity set such that each ~~Employee~~ department has at most one manager, although a single employee is allowed to manage more than one department.

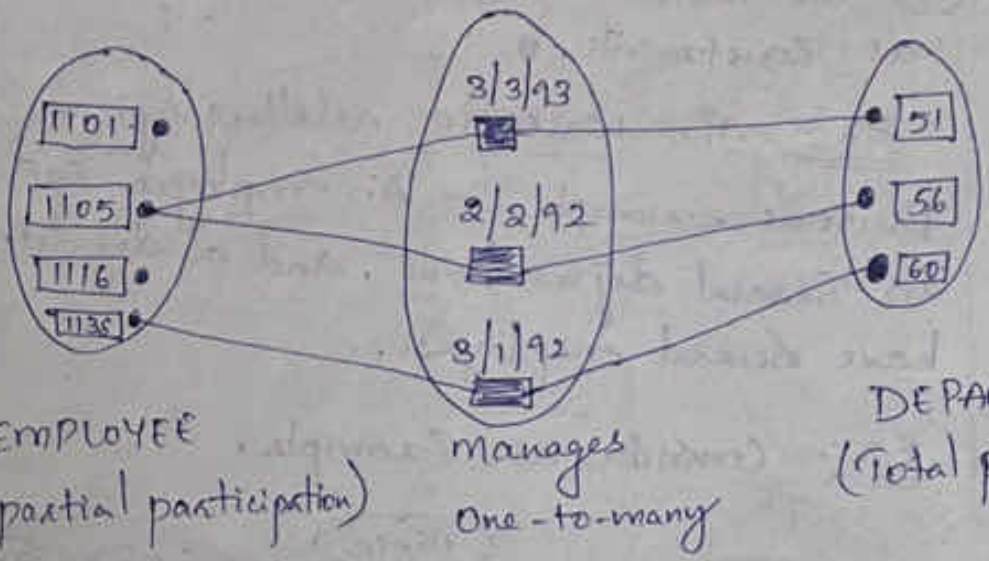
The restriction that each department has at most one manager is an example of a Key Constraint.

This restriction is indicated in the above ER diagram by using an arrow from Departments to Manages.

The arrow states that given a Departments entity, we can uniquely determine the Manages relationship in which it appears.

3 NOV 201

→ An instance of the Manages relationship is shown below:-



→ A relationship set like Manages is said to be One-to-many, to indicate that one employee can be associated with many departments, whereas each department can be associated with atmost one employee as its manager.

→ The works-in relationship set, in which an employee is allowed to work in several departments and a department is allowed to have several employees, is said to be many-to-many.

$$\begin{matrix} A & + & m \\ m & - & 1 \end{matrix}$$

Key Constraints for Ternary Relationships

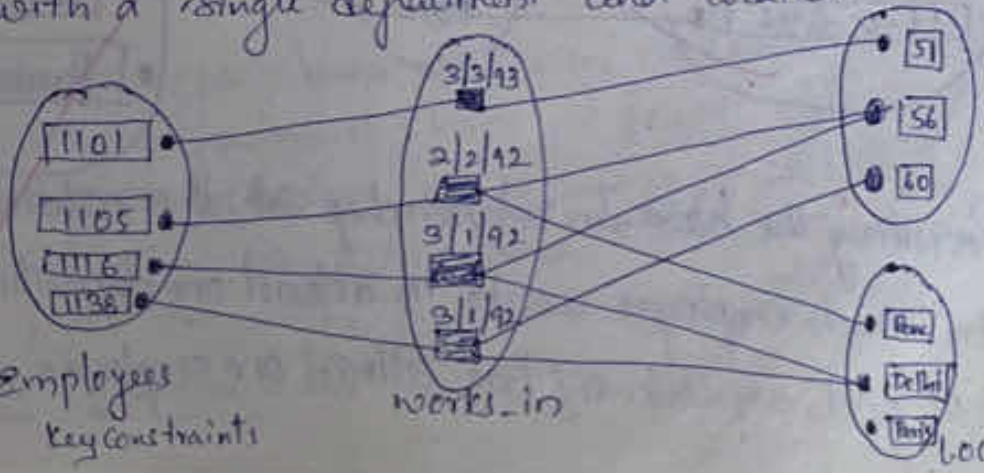
→ The Key Constraint to relationship sets involving three or more entity sets:

"If an entity set E has a key constraint in a relationship set R, each entity in an instance of E appears in at most one relationship in R".

→ To indicate the key constraint on entity set E in relationship set R, we draw an arrow from E to R.

Ex:- A ternary relationship set with key constraints.

→ Each Employee works in at most one department and at a single location. Each department can be associated with several employees and locations and each location can be associated with several departments and employees; however each employee is associated with a single department and location.

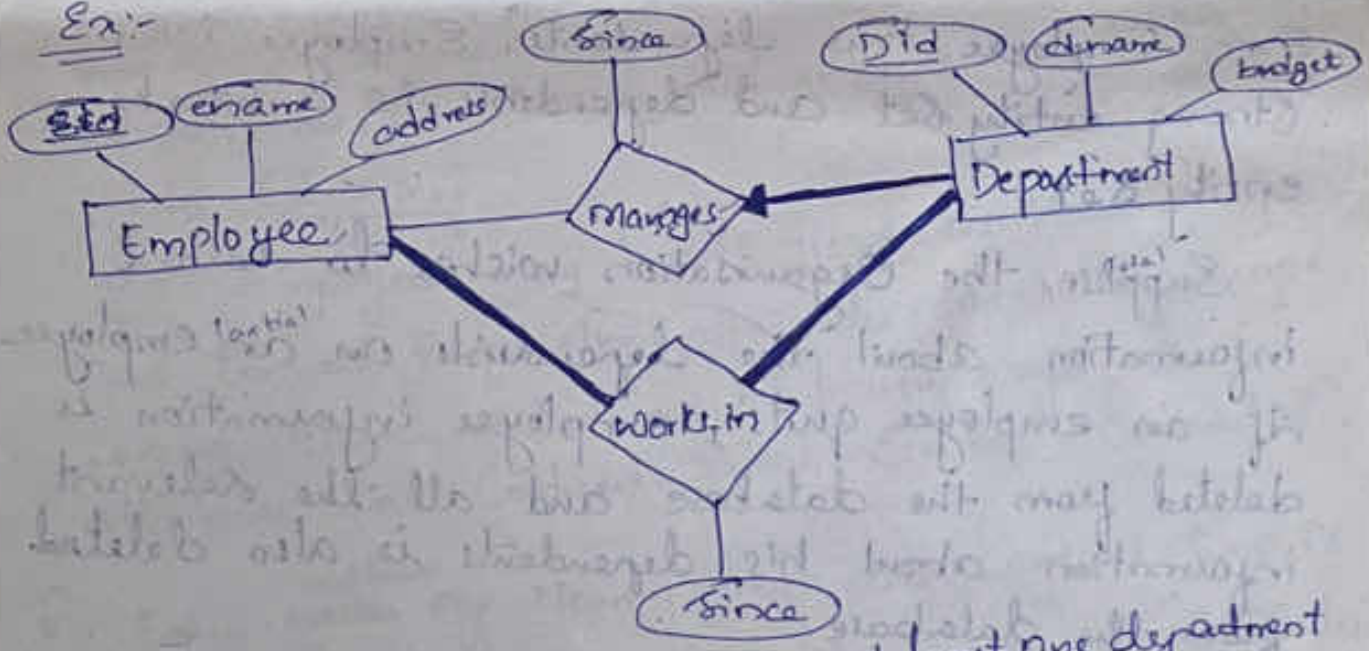


Participation Constraints :-

→ The key constraint on Manages tells that 'a department has at most one manager' and 'every department is required to have a manager'. This requirement is an example of participation Constraint; the participation of the entity set Department in the relationship set Manages is said to be total participation. The participation that is not total is said to be partial participation.

Ex:- The participation of the entity set Employees in Manages is partial, since every employee cannot manage a department.

Ex:-



(*) Each employee ~~has~~ works in at least one department and each department has at least one employee, i.e., the participation of both employees and Departments in Works_in is total. and the two are connected by a thick line, the presence of an arrow indicates a key Constraint.

Weak Entities.

29 OCT 2017

(*) A weak entity type is an entity type whose existence depends on some strong entity type.

(*) A weak entity type has no meaning in the E-R diagram without the entity on which it depends.

(*) The entity type on which the weak entity type depends is called the identifying owner.

(*) The set of ~~identi~~ attributes of a weak entity set that uniquely identify a weak entity for a given owner entity is called a partial key of the weak entity set.

Ex:- "Employee has dependents". Employee is a strong entity set and dependents is a weak entity set.

Suppose, the Organisation wishes to record information about the dependents on an employee. If an employee quits, employee information is deleted from the database and all the relevant information about his dependents is also deleted from the database.

(*) The following conditions must hold :-

1) The owner entity set and the weak entity set must participate in a one-to-many relationship set. This relationship set is called identifying relationship set of the weak entity set.

2) The weak entity set must have total participation in the identifying relationship set "has" is the identifying relationship set

→ To show that Dependents is a weak entity and 'has' is its identifying relationship we draw both with doublelines.

→ To indicate that Pname is a partial key for dependents, we underline it using a brokenline.

CLASS HIERARCHIES

(*) Sometimes, we classify the entities in an entity set into Subclasses.

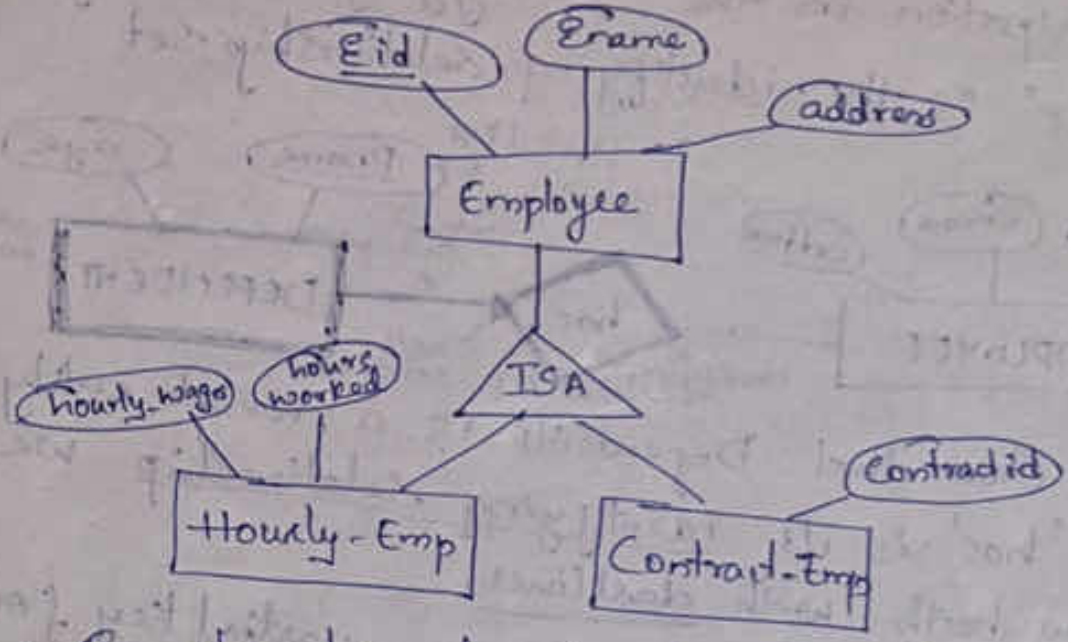
Ex:- Employees may be classified as Hourly-Emps and Contract-Emps. The attributes for Hourly-Emps can be "hours_worked" and "hourly-wage" and attributes for Contract-Emps can be "Contractid".

(*) Every ~~entity~~ instance of Hourly-Emp and Contract-Emp is also an instance of Employee entity and inherits all attributes of Employee entity

Ex:- "The attributes for the entity set Employees are inherited by the entity set Hourly-Emp and Hourly-Emp ISA Employee.

Class hierarchy

29 OCT 2012



→ The class hierarchy can be viewed in one of two ways:

1) Employees is Specialized into subclasses. Specialization is the process of identifying subclasses for a superclass i.e., entity set that show some distinguishing characteristic.

The superclass is defined first, the subclasses are defined next and subclass-specific attributes and relationship sets are then added.

2) Hourly_Emps and Contract_Emps are generalized by Employees.

Ex:- Two entity sets motorboats and cars may be generalized into an single entity set Motor_Vehicles.

"Generalization consists of identifying some common characteristics of a collection of entity sets and creating a new entity set that contains entities possessing these common characteristics."

The subclasses are defined first, the superclass is defined next and any relationship sets that involve the superclass are then defined.

→ We can specify two kinds of constraints with respect to ISA hierarchies.

1. Overlap Constraints
2. Covering Constraints.

Overlap Constraints :- It determines whether two subclasses are allowed to contain the same entity.

Def :- It specifies that an entity instance of the superclass can simultaneously be a member of two (or more) subclasses.

Ex :- A Employee can both be a Contract_Emp and a Senior_Emp.

We can denote this by writing 'Contract_Emp OVERLAPS Senior_Emp.

But, an Employee cannot be both Contract_Emp and Hourly_Emp.

Covering Constraints

→ Covering Constraints determine whether the entities in the subclasses collectively include all entities in the superclass.

Ex:- Every Motor_Vehicle entity have to be either a Motorboat entity or a car Entity.

Def:- Every instance of a superclass is an instance of a subclass.

We denote it by writing 'MOTORBoats AND Cars COVER Motor_Vehicles'.

AGGREGATION — whole/part — has-a relationship

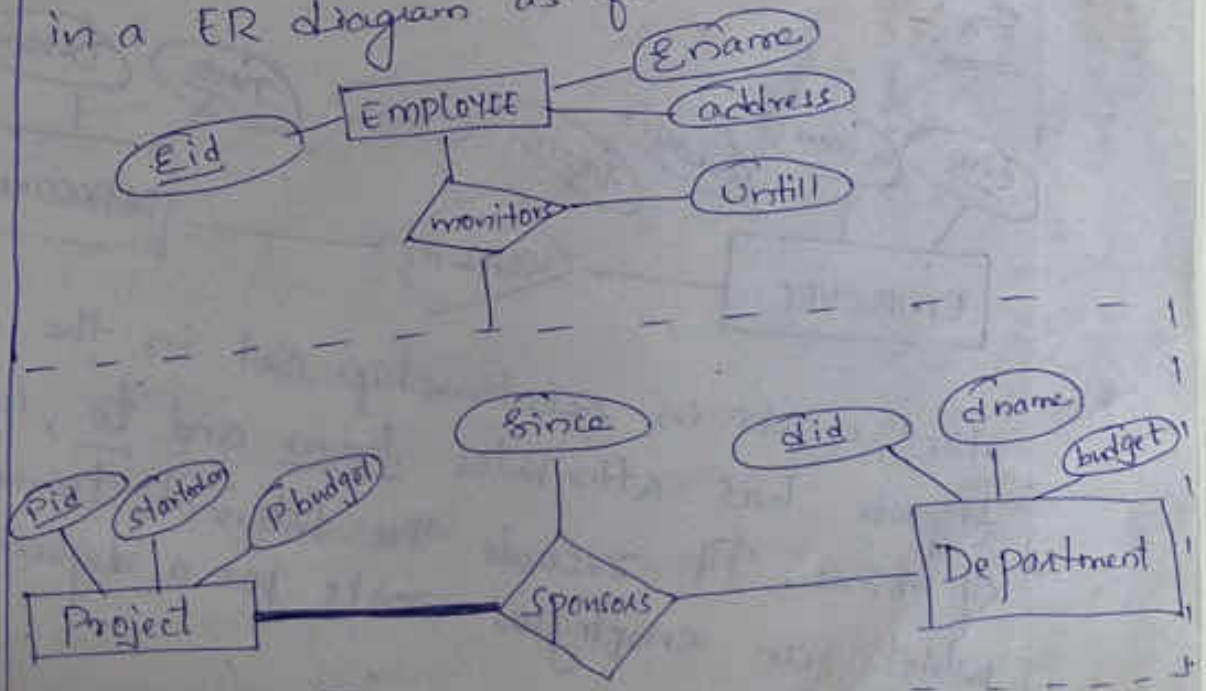
→ A Relationship is an association between entity sets.

→ Sometimes, we have to model a relationship between a collection of entities and relationships.

Aggregation allows us to indicate that a relationship set (identified through a dashed box) participates in another relationship set.

Ex:- Suppose we have an entity set called Projects and that each project entity is sponsored by one or more departments. A department that sponsors a project might assign employees to monitor the sponsorship.

The above concept can be represented in a ER diagram as follows:-



CONCEPTUAL DESIGN WITH THE ER MODEL.

29 OCT

→ Developing ER diagram presents several choices. like :-

(*) Should a Concept be modeled as a Entity or an attribute?

(*) Should a Concept be modeled as an entity or relationship?

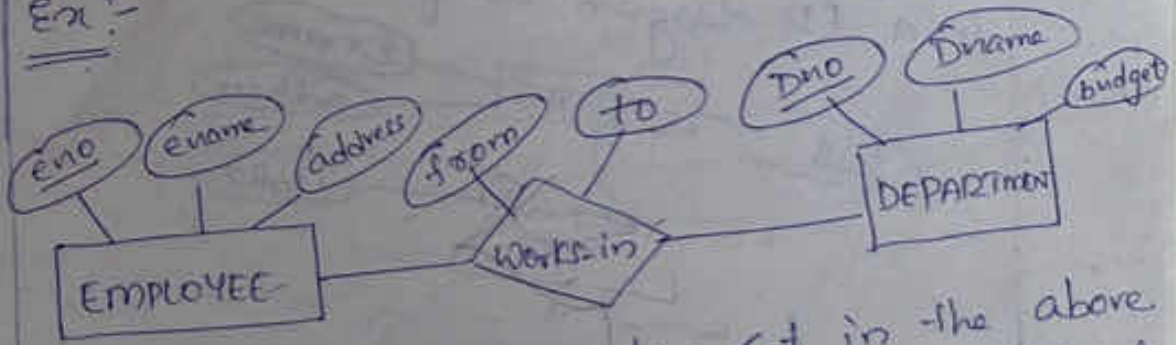
(*) Should we use binary or Tenerary relationship?

(*) Should we use aggregation?

Entity Versus attribute

→ While identifying the attributes of an entity set, it is sometimes not clear whether a property should be modeled as an attribute or as an entity set.

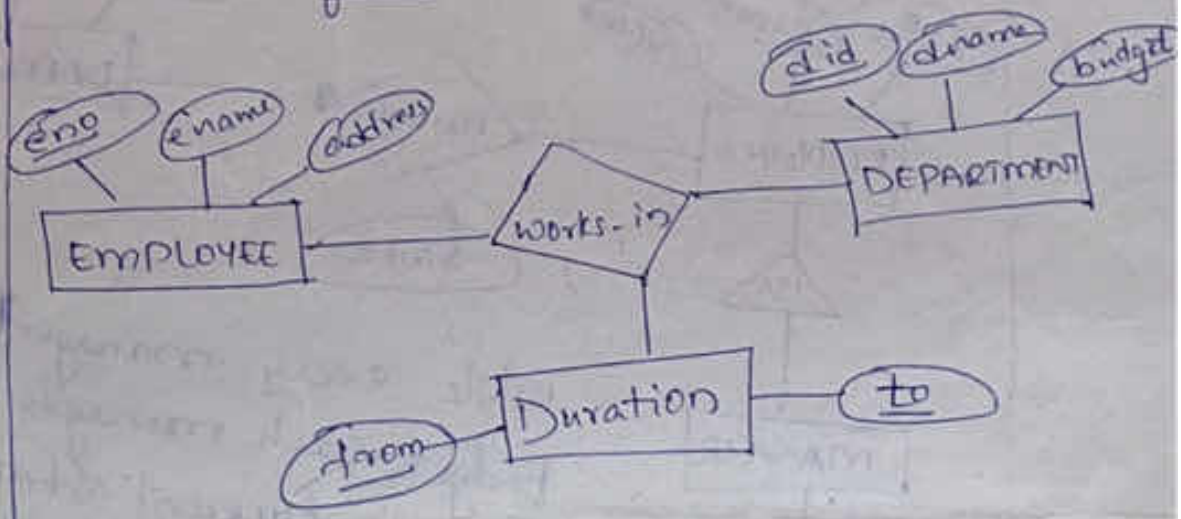
Ex :-



The works-in relationship set in the above figure has attributes from and to, instead of since. It records the interval during which an employee works for a department.

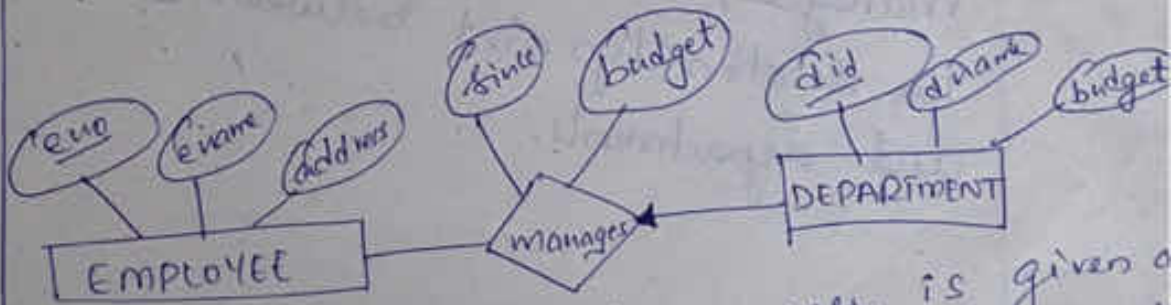
The problem is that we want to record values for the descriptive attributes for each instance of the works-in relationship.

This problem can be solved by introducing an entity set called Duration, which has attributes from and to.



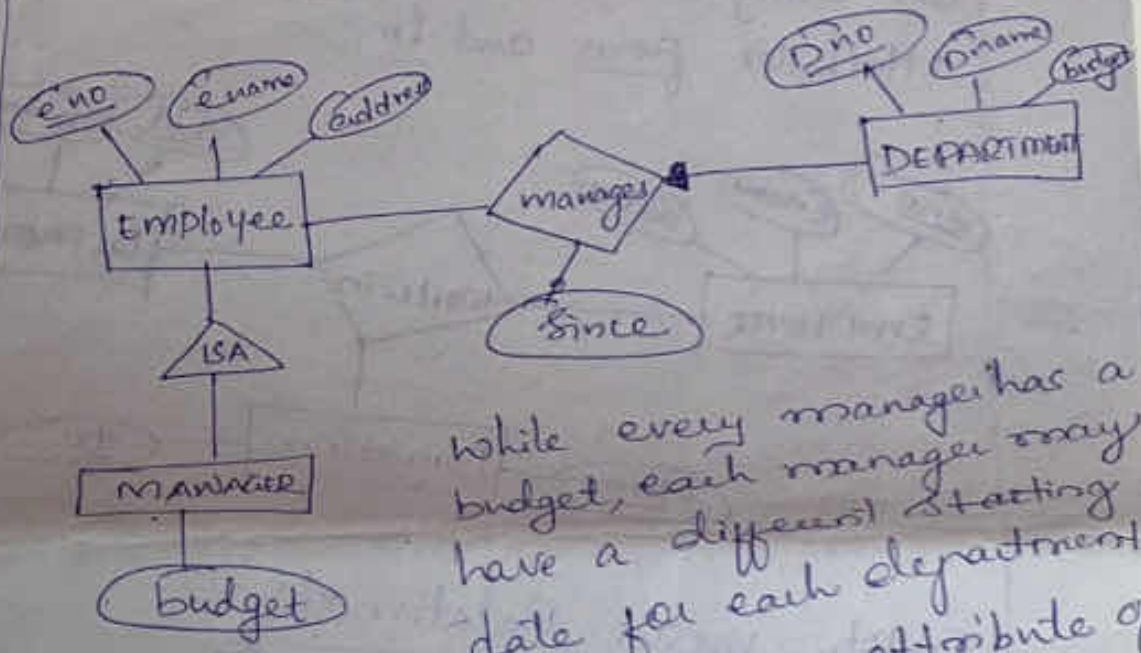
Entity versus Relationship

Ex:- Consider the relationship set called manages



Suppose, the department manager is given a budget. The design misleads, as it suggests that the budget is associated with the relationship, but it is actually associated with the manager.

This problem can be solved by introducing a new entity set called Managers, which can be placed below Employees in an ISA hierarchy, to show that every manager is an employee.

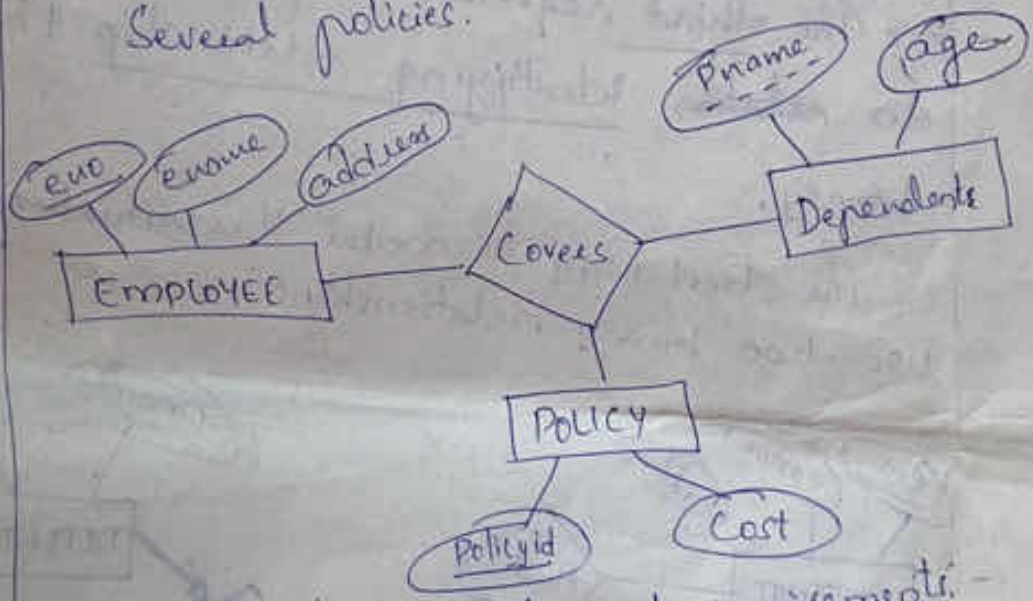


While every manager has a budget, each manager may have a different starting date for each department. In this case, budget is an attribute of managers, but since is an attribute of the relationship set between managers and departments.

Binary Versus Ternary Relationships

Consider the ER diagram shown below. It models a situation in which

* An employee can own several policies,
* Each policy can be owned by several employees
* Each dependent can be covered by several policies.



Suppose, we have additional requirements:

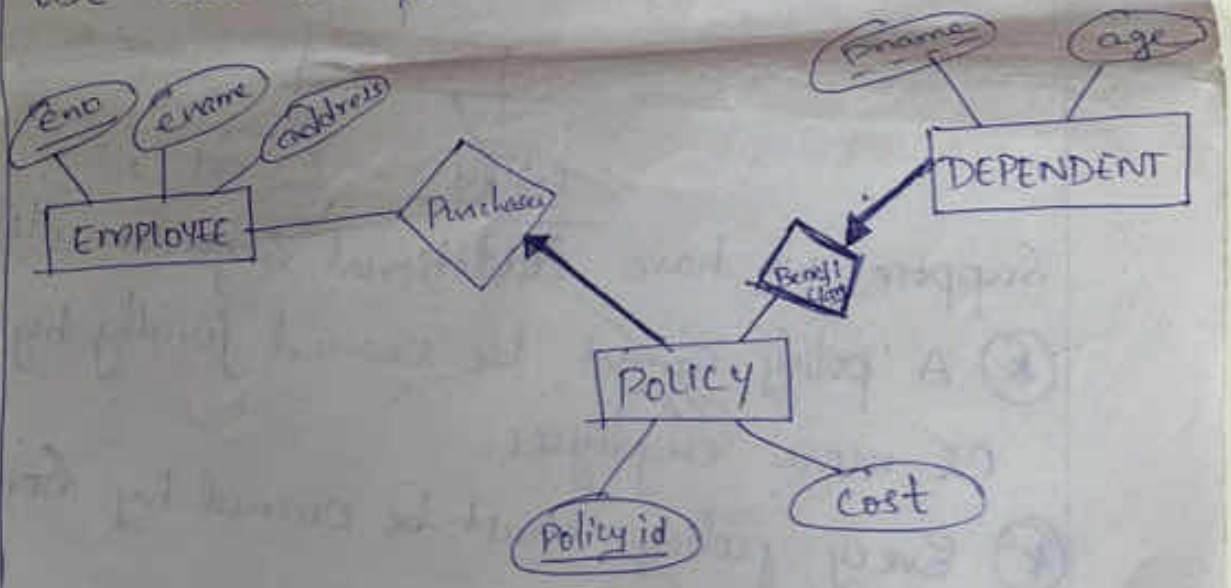
* A policy cannot be owned jointly by two or more employees.
* Every policy must be owned by some employee.
* Dependents is a weak entity set and each dependent entity is uniquely identified by taking Pname in conjunction with the policyid of a policy entity, which covers the given dependent.

→ The first requirement suggests that we impose a key constraint on Policies with respect to Covers.

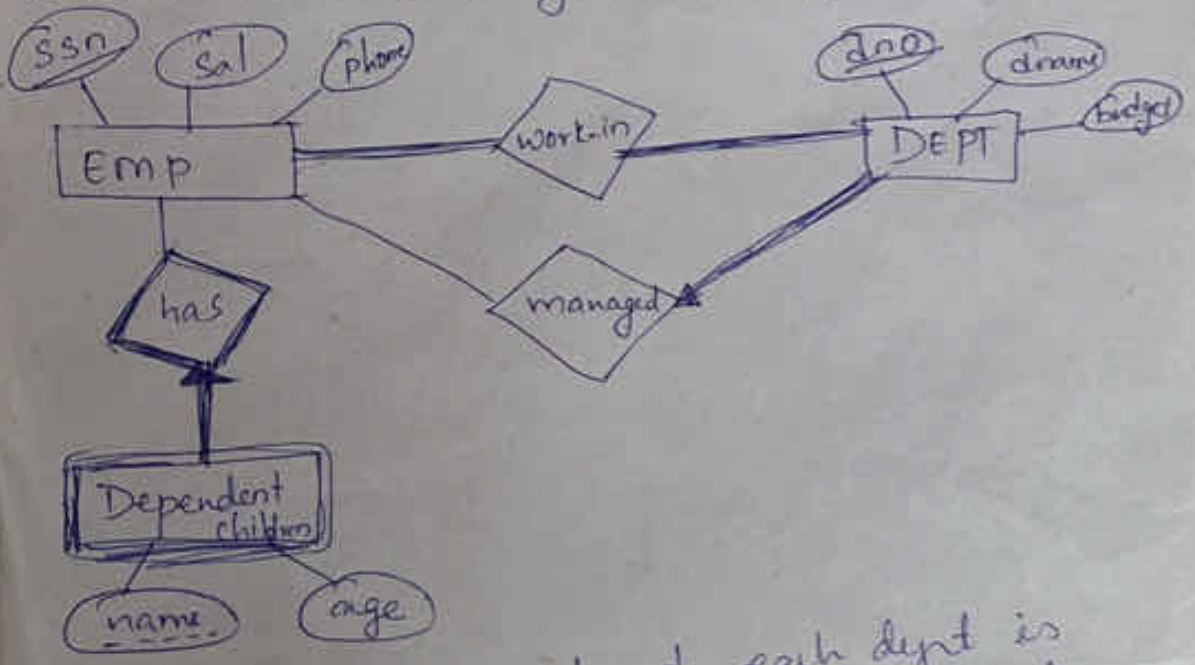
→ The second requirement suggests that we impose a total participation constraint on Policies. i.e., each policy covers atleast one dependent.

→ The third requirement forces us to introduce an identifying relationship that is binary.

The Best way to model this situation is to use two binary relationships.



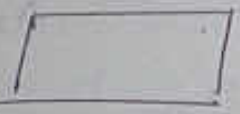
A Company db needs to store info. about employees (identified by SSN, Sal and phone as attributes, departments (identified by dno, dname and budget as attributes) and children of employees (with name and age as attributes). (13)



Emp. work in departments, each dept is managed by an emp, a child must be identified uniquely by name when the parent (who is an employee) is known. We are not interested about a child once the parent leaves the company.

Extra

ER diagram Symbols & Notations.

Entity → 

Attribute → 

Relationship → 

Weak Entity → 

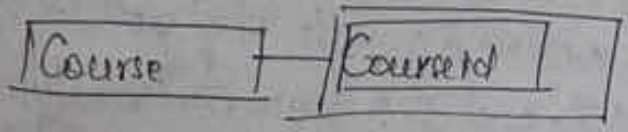
Course

Multivalued Attribute → 

Weak Relationship → 

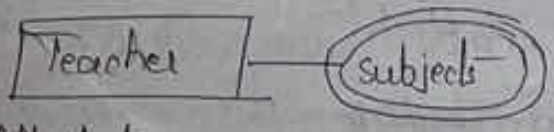
Eg:-

Weak Entity



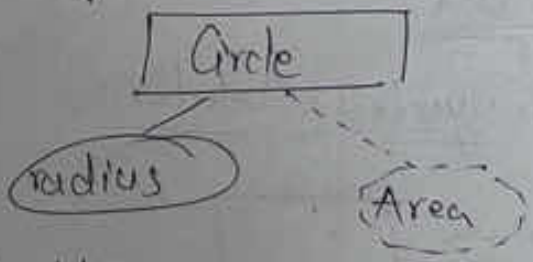
Multivalued Attribute

If an attribute can have more than one value it is called multivalued Attribute.

Eg:- 

Derived Attribute:-

An Attribute based on another attribute this is found rarely in ER diagrams. Circle area can be derived from radius



Cardinality

It specifies how many instances of an entity relate to one instance of another entity

1 : 1

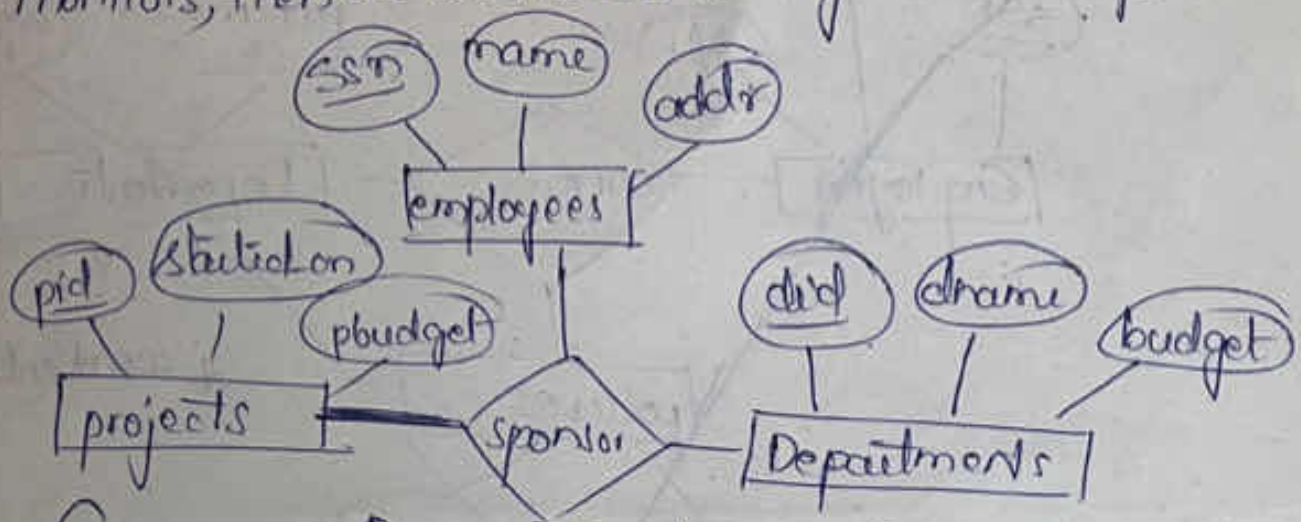
1 : N

accepted if each party

AGGREGATION VERSUS TERNARY RELATIONSHIP

The choice between using aggregation or ternary relationship is mainly determined by the existence of a relationship that relates a relationship set to an entity set. For example, Consider the ER diag below. According to it, a project can be sponsored by any no. of department, a department can sponsor one or more projects and each sponsor is monetored by one or more

employees. If we don't want to record the until attribute of monitors, then we can use a ternary relationship



CONCEPTUAL DESIGN FOR LARGE DATABASES