## CONTENTS:

**UNIT 1 – DATA BASE SYSTEM APPLICATIONS**

## Database System Applications:

* 1. A Historical Perspective
  2. File Systems versus a DBMS
  3. The Data Model
  4. Levels of Abstraction in a DBMS
  5. Data Independence
  6. Structure of a DBMS

## Introduction to Database Design :

* 1. Database Design and ER Diagrams
  2. Entities, Attributes, and Entity Sets,
  3. Relationships and Relationship Sets,
  4. Additional Features of the ER Model,
  5. Conceptual Design With the ER Model

## DATA BASE SYSTEM APPLICATIONS

**DATA**

* Data is a fact or figures or entity.
* When activities in the organization takes place, the effect of these activities need to be recorded which is known as Data.

## INFORMATION

* Processed data to carrying out useful the business activities is called information

## DATABASE

A **database** is a collection of data, typically describing the activities of one or more related organizations.

For example, a university database might contain information about the following:

* **Entities** such as students, faculty, courses, and classrooms.
* **Relationships** between entities, such as students' enrollment in courses, faculty teaching courses, and the use of rooms for courses.

## DATABASE MANAGEMENT SYSTEM

* + A database management system is software designed to assist in maintaining and utilizing large collections of data.
  + It is used for storing data and retrieving the data effectively when it is needed.
  + It also provides proper security measures for protecting the data from unauthorized access.
  + In Database Management System, the data can be fetched by SQL queries and relational algebra query language.
  + It also provides mechanisms for data recovery and data backup.

## ADVANTAGES OF A DBMS:

Managing data with DBMS has many advantages:

* + - Data independence
    - Efficient data access
    - Data integrity and security
    - Data administration
    - Concurrent access and crash recovery
    - Reduced application development time

## Data independence:

Data Independence is the ability to modify a schema definition in one level of dbms structure without affecting a schema definition in the next higher level. DBMS achieves this data independence by providing an abstract view of the data to the higher level users

## Efficient data access:

A DBMS utilizes a variety of sophisticated techniques to store and retrieve data efficiently.

## Data integrity and security:

The DBMS can enforce integrity constraints on the data before being accessed. Also, the DBMS can enforce access controls that govern what data is visible to different classes of users.

## Data administration:

When several users share the data, centralized data administration can be done effectively with DBMS.

## Concurrent access and crash recovery:

Concurrent access means the ability to access the data by multiple users at the same time. This concurrent accessing of data can be achieved in DBMS. Dbms also can protect or recover data even in case of system failures or crash.

## Reduced application development time:

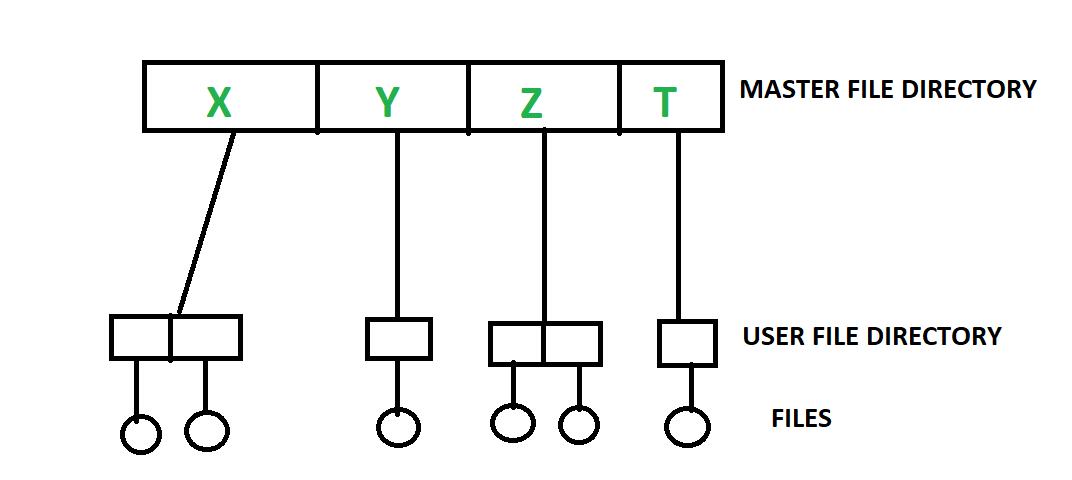
Clearly, the DBMS supports many important functions that are common to many applications accessing data stored in the DBMS. Hence DBMS facilitates quick development of applications.

## A HISTORICAL PERSPECTIVE

* From the earliest days of computers, storing and manipulating data have been a major application focus. The first general-purpose DBMS, called the Integrated Data Store, was designed by Charles Bachman in the early 1960s. It formed the basis for the **network data model.**
* In the late 1960s, IBM developed the **Information Management System** (IMS) DBMS which formed the basis for the **hierarchical data model**.
* In 1970s, **Edgar Codd**, at IBM, proposed **the relational data model**. This sparked rapid development rapid development of several relational DBMS and its usage started increasing in managing data in corporate.
* Soon, IBM, in early 1980s, has developed a SQL (Structured Query Language) for relational databases.
* During this period, the concept of **concurrent execution of database programs, called transactions**, was introduced in DBMS system which allowed the user to run their programs concurrently.
* In the late 1980s and the 1990s, advances have been made in many areas of database systems. More emphasis is given on **new data types** such as images and text, and the ability to support **complex analysis** of data within of an enterprise.
* Over a period of time, **data warehouse systems** was proposed, which is used to consolidate data from several databases, and for carrying out specialized analysis.
* By introduction of **enterprise resource planning (ERP) and management resource planning (MRP)** packages exciting new features were added to existing database system.
* Many other **packages** like Baan, Oracle, PeopleSoft, SAP, and Siebel which were user friendly and allowed the user to carry out the task easily.
* Most significant change in DBMS is through integration of DBMS with internet which allowed DBMS to store data that can be accessed through web browser.
* Today this field is being driven by exciting visions such as multimedia databases, interactive video, digital libraries and so on.
* With humongous data being available online, DBMS is gaining more significance day by day

## FILE SYSTEM vs DBMS

* A File System is a collection of raw data files stored in a storage medium like hard disk. Files are created and manipulated by writing programs, so the permanent records are stored in various files.



* File system **organizes the files and helps in retrieval of files** when they are required. File systems consists of different **files which are grouped into directories**. directories can contain other folders and files.
* Without file management, all files would have no organization and it would be impossible for a file with the same name to exist. Files are most often managed in a [hierarchy,](https://www.computerhope.com/jargon/h/hierfile.htm) which allows you to view files in the [current directory](https://www.computerhope.com/jargon/c/currentd.htm) and then navigate into any [subdirectories](https://www.computerhope.com/jargon/s/subdirec.htm).
* File system performs basic operations like file management, file naming, giving access rules etc.

Fig. 1.1 : Sample File System Architecture

The problems in file processing system are

## Data redundancy and inconsistency

* + **Difficulty in accessing data**

## Data isolation

* + **Integrity problems**

## Atomicity problems

* + **Security problems**

|  |  |
| --- | --- |
| **DBMS** | **File System** |
| DBMS is a collection of data. In DBMS, the user is **not required to write the procedures**. | File system is the part of the OS and organizes a collection of raw data files into hard disk. In this system, the user has to **write the procedures** for  managing the database. |
| It gives an **abstract view of data** that hides  the details. | It provides the **detail of the data representation**  **and storage of data.** |
| It provides a **crash recovery mechanism**  that protects the data from the system  failure. | File system **doesn't have a crash mechanism,** so, the content of the file will be lost in case of the  system failure. |
| It provides a **good protection** mechanism. | It is very **difficult to protect a file** under the file  system. |
| It provides **higher data consistency** using  normalization | It has data inconsistency |
| It contains a wide variety of sophisticated  techniques to easily store and retrieve the data. | It can't efficiently store and retrieve the data. |
| I can easily implement **complicated**  **transactions**. | It does **not** provide support for **complicated**  **transactions.** |
| It takes care of **Concurrent access** of data using some form of locking. | It does not offer **concurrent access** of data. |
| **Ex:** MySQL, MSSQL, Oracle | **Ex:** NFTS and Ext |

## DATA MODEL

A Data model(or Data base model) defines the logical design and structure of a database. It defines how data will be stored, accessed and updated in a database management system.

Data Models are fundamental entities to introduce abstraction in a DBMS. The following are the different types of data models:



**DATA MODELS**

**Hierarchical**

**Model**

**Network**

**Model**

**E-R**

**Models**

**Relational**

**Model**

Object **Oriented Data**

**Model**

## Fig 1.2: Types of Data Models

Among the above data models, relational model is the most widely used database model.

## Hierarchical Model

* This database model organizes data into a tree-like-structure, starts from the root data, and then expands like a tree, adding child nodes to the parent nodes.
* This model efficiently describes many real-world relationships.
* Pointers are used to link the parent node with the child node and also to navigate between the stored data.
* This model represents one-to-many relationship between the two different types of data, for example, one department can have many courses, many professors and many students. So, if we want to access a student record, there is only one path from college node to student node.



Students

Teachers

Department

College

Infrastructure

Course

**Fig. 1.3:** Sample Hierarchical Model

## Advantages:

* **Simplicity:** It is very simple and fast to traverse through a tree-like structure.
* **Data Integrity:** Any change in the parent node is automatically reflected in the child node, so the integrity of data is maintained.

## Disadvantages:

* Searching for data requires the DBMS to run through the entire model from top to bottom until the required information is found, making queries very slow.
* This model does not support complex relationships such as, many-to-one relationships and many-to-many relationships.
* If a parent node is deleted then the child node is automatically deleted.

## Network Model

* + - * Network Database Model organizes the data using **Graph like structures**. It allows a record to have more than one parent.
      * It was the most popular model before the relational model.
      * This model has the ability to manage one-to-one relationships as well as one –to-many and many-to-many relationships. As there are more relationships so there can be more than one path to the same record. This makes data access easier and faster.

Students

Infrastructure

Fig. 1.4 Sample Network Model

Department

College

## Advantages

* The network model is conceptually simple and easy to design.
* The network model can handle the **one to many and many to many relationships**.
* The data access is easier and faster as there can be more than one path to reach a particular node.
* As there is a parent-child relationship so data integrity is present. Any change in parent record is reflected in the child record.

## Disadvantage

* All the records are maintained using pointers and hence the whole database structure becomes very complex.
* The insertion, deletion and updating operations of any record require the large number of pointers adjustments.
* The structural changes to the database are very difficult.

## Entity-relationship Model

* E-R Models(semantic data model) is useful in representing a conceptual design for the database.
* It is used to pictorially denote entities, attributes and the relationships among them.

**Entity –** These are real world objects represented in rectangle.

**Attributes –** These are the properties of entities represented in ellipse.

**Relationship –** This model represents different types of logical relationship such as,

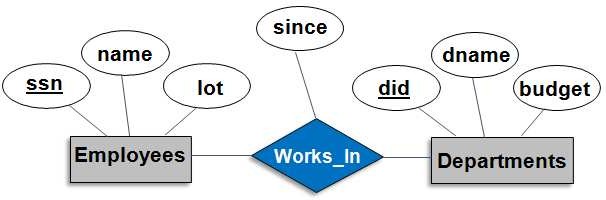


Fig1.5 Sample E-R Model

* One to one
* One to many
* Many to one
* Many to many

## Advantages

* + It is a simple and effective

communication tool, so this model is used widely by the database designers.

* + This model can be easily converted to relational model and also to any other data models like network model, hierarchical model etc.

## Disadvantages

* + There is no industry standard for developing an ER model. So one developer might use notations which are not understood by other developers.
  + As this model represents a high-level design of the database, it is possible that some details of database might be hidden.

## Relational Model

* In this model, data is organised in two-dimensional **tables** and the relationship is maintained by storing a common field. Here tables are also known as **relations** in relational model.
* Each table consists of a **collection rows and columns** where each row represents records and column represents attribute of an entity.
* Here in this relation, **employee** is the table name, **eid, ename, salary and age** are the attributes of the table employee. Every row with values are the records of employee which is usually called as tuples.

## Advantages

* **Simple**: This model is more simple as compared to the network and hierarchical model.
* **Scalable**: This model is easily scalable as any number of rows and columns can be added whenever required.
* The relational database supports both **data**

## Employee

|  |  |  |  |
| --- | --- | --- | --- |
| **eid** | **ename** | **salary** | **age** |
| 22 | Abi | 10000 | 25.0 |
| 31 | Bob | 25000 | 35.5 |
| 58 | Akhil | 45000 | 35.0 |

Fig1.6 **Sample Relational Model**

## independence and structure independence of DBMS.

* It is easier to **maintain security** as compared to other models.

## Disadvantages

* + **Hardware Overheads:** For hiding the complexities and making things easier for the user this model requires more powerful hardware computers and data storage devices.
  + **Bad Design:** As the relational model is very easy to design and use. So the users don't need to know how the data is stored in order to access it. This ease of design can lead to the development of a poor database which would slow down if the database grows.

## But all these disadvantages are minor as compared to the advantages of the relational model.

* + 1. **Object Oriented Data Model**
* In object oriented data model represents **data and the relationships** of real world entities in a single structure, which is known as object. In this model, two are more related objects are connected through links as shown in the below given example.

|  |
| --- |
| **SHAPE** |
|  |
| **Methods**  **GetArea()** |
| **GetPerimeter()** |

**Attributes Length Breadth**

**RECTANGLE**

**TRIANGLE**

**Attributes Base**

**Height**

Fig 1.7 Object Oriented Data Model

**Attributes Center Radius**

**CIRCLE**

* The real-world problems are more closely represented through the object-oriented data model.
* In this model, both the data and relationship are present in a single structure known as an object. In this model, two are more objects are connected through links. We use this link to relate one object to other objects.
* This can be understood by the example given below.
* In this model, the key concepts of object oriented programming languages such as Inheritance, polymorphism, overloading, encapsulation and information hiding can easily be represented.

## Advantages

* Object-oriented databases can handle the **different types of data**, such as example, pictures, voice video, text, numbers and so on.
* This model provide us code reusability, enriched real world modelling, improved reliability and flexibility.
* This model incurs low maintenance costs because most of the tasks are encapsulated in the system.

## Disadvantages

* There is no universally defined data model for an OODBMS.
* It does not provide adequate security mechanisms.
* It makes the system more complex than that of RDBMSs.

## LEVELS OF ABSTRACTION

**Data abstraction** is a process of hiding irrelevant information(such as how the data are stored and maintained) at each level of a database, and representing only the essential features to simplify user's interaction with the system.

Data abstraction is achieved by describing the data that is required for the users at each level with its corresponding schemas such as Physical schema, Conceptual schema and External schema as shown in this figure:

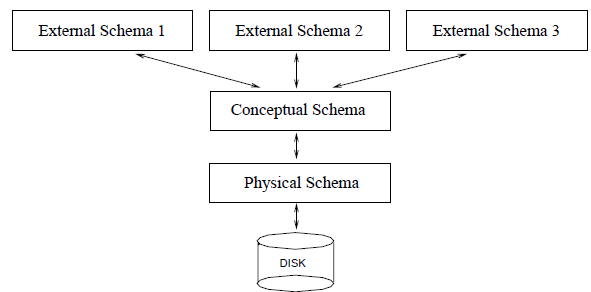


Fig.1.8 Levels of Abstraction in a DBMS

## Physical schema

It describes how data are actually stored on storage devices. That is, it gives a complete description of the physical storage of the database, such as data representation methods, file organization and access methods, etc. this schema provides the lowest level of data abstraction. And The system programmers work at this level.

The process of arriving at a good physical schema is called **physical database design**.

## Conceptual schema

**It** describes the data stored in the database and relationship among them.

* In RDBMS, conceptual schema describes the entities, attributes, and relationships together with integrity constraints for all the relations that are stored in the database.
* The physical storage details are hidden at this level.
* ‘In this level, the programmers as well as the database administrator (DBA) work. The process of arriving at a good conceptual schema is called **conceptual database design**

## View schema

* View schema describes a collection of views that are derived from the conceptual schema.
* Views contain the parts of the entire database to serve the specific needs of particular group of users. As these views are not actually stored into the database, the views are also called as virtual relations.
* View level provides the highest level of abstraction.

**Example: University Database** – Schema at the different levels of DBMS is shown below.

|  |  |
| --- | --- |
| Type of Schema | Implementation |
| External Schema | **View 1**: Course info(cid:int,cname:string)  **View 2**: studeninfo(id:int. name:string) |
| Conceptual Shema | Students(id: int, name: string, login: string, age: integer)  Courses(id: int, cname.string, credits:integer) Enrolled(id: int, grade:string) |
| Physical Schema | * Relations stored as unordered files. * Index on the first column of Students. |

This levels of data abstraction in the three tier architecture of in DBMS reduces the complexity of the database as well as provides data independence at each level.

## DATA INDEPENDENCE

Data independence refers to the ability to modify the schema at one level of the database system without altering the schema at the next higher level.

There are two types of data independence:

## Logical Data Independence

* + Logical data independence can be defined as the capacity to change the conceptual schema without having to change the external schema.
  + If we do any changes in the conceptual view of the data, then the users view in the external level would not be affected.
  + Logical data independence is used to separate the external level from the conceptual view.
  + Logical data independence occurs at the user interface level. The meaning of this is that the external user can access the data while physical schema is getting modified by the DBA.

## Physical Data Independence

* + Physical data independence can be defined as the capacity to change the internal schema without having to change the conceptual schema.
  + If we do any changes in the storage size of the database system server, then the Conceptual structure of the database will not be affected.
  + Physical data independence is used to separate conceptual levels from the internal levels.
  + Physical data independence occurs at the logical interface level.

## STRUCTURE OF A DBMS

The DBMS accepts SQL commands generated from a variety of user interfaces, produces query evaluation plans, executes these plans against the database, and returns the answers.

Figure 1.8 shows the structure of a typical DBMS based on the relational data model.

**[**

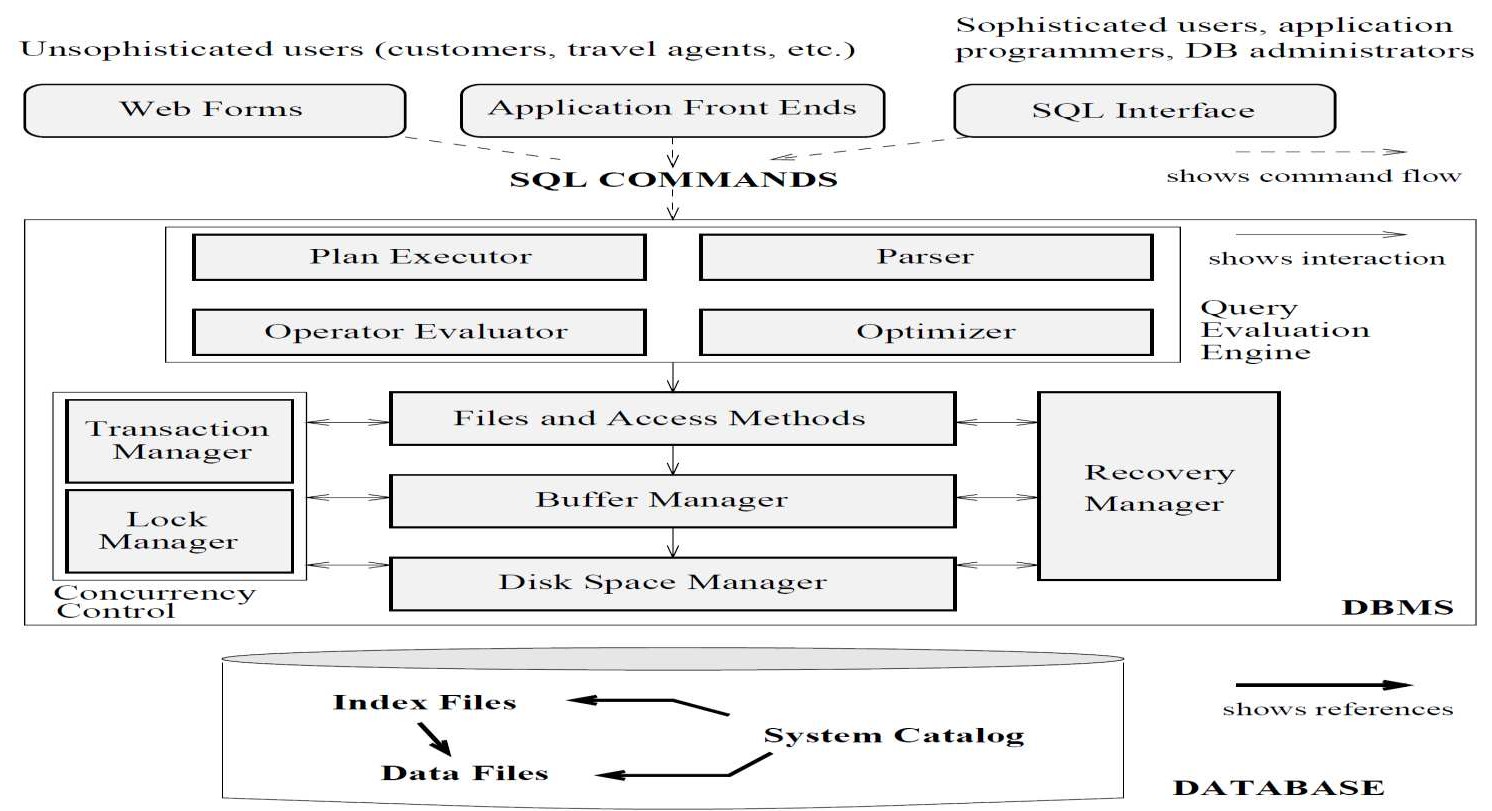
## Query Evaluation Engine consists of the following components:

* **Parser:** The query parser breaks the query into tokens and then it ensures that the query is syntactically and semantically correct. If the query does not have any errors, then it is converted into an algebraic expression and passed into the Operator evaluator.
* **Operator Evaluation:** Operator evaluator evaluates the operators in the query using several alternative techniques such as Iteration techniques, Indexing techniques and partitioning techniques. And then it generates a set of possible query evaluation plans for executing the SQL query,
* **Optimizer:** Optimizer chooses the optimized the query evaluation plan among the set of possible query evaluation plans to execute the users query the most effectively.
* **Execution Engine:** Finally Execution Engine executes the query as per query evaluation plan and display the required result.

During the execution of the query evaluation plan, the responsibility of the modules, such as Files and access methods, Buffer Manager, Disk space manager, Transaction manager, Lock manager, and Recovery manager are given below.

**Files and access methods**: It is responsible for the abstraction of file structures stored and for providing fast access to the desired file by creating indexes on the files.

**Buffer Manager:** It is responsible for fetching data from disk storage into main memory, and deciding what data to cache in main memory.

**Disk space manager:** It manages the space on the disk by providing empty space for new requests, deleting space allocated for existing files which are deleted by users.

## Fig 1.9 Architecture of a DBMS

**Concurrency control is achieved by:**

**Transaction manager: It** ensures that the database remains in a consistent state despite system failures (e.g., power failures and operating system crashes) and transaction failures.

**Lock manager:** It keeps track of requests for locks and grants locks on database objects when they become available.

Transaction manager and Lock manager both ensures that the data remains consistent during the concurrent access of the data for all users.

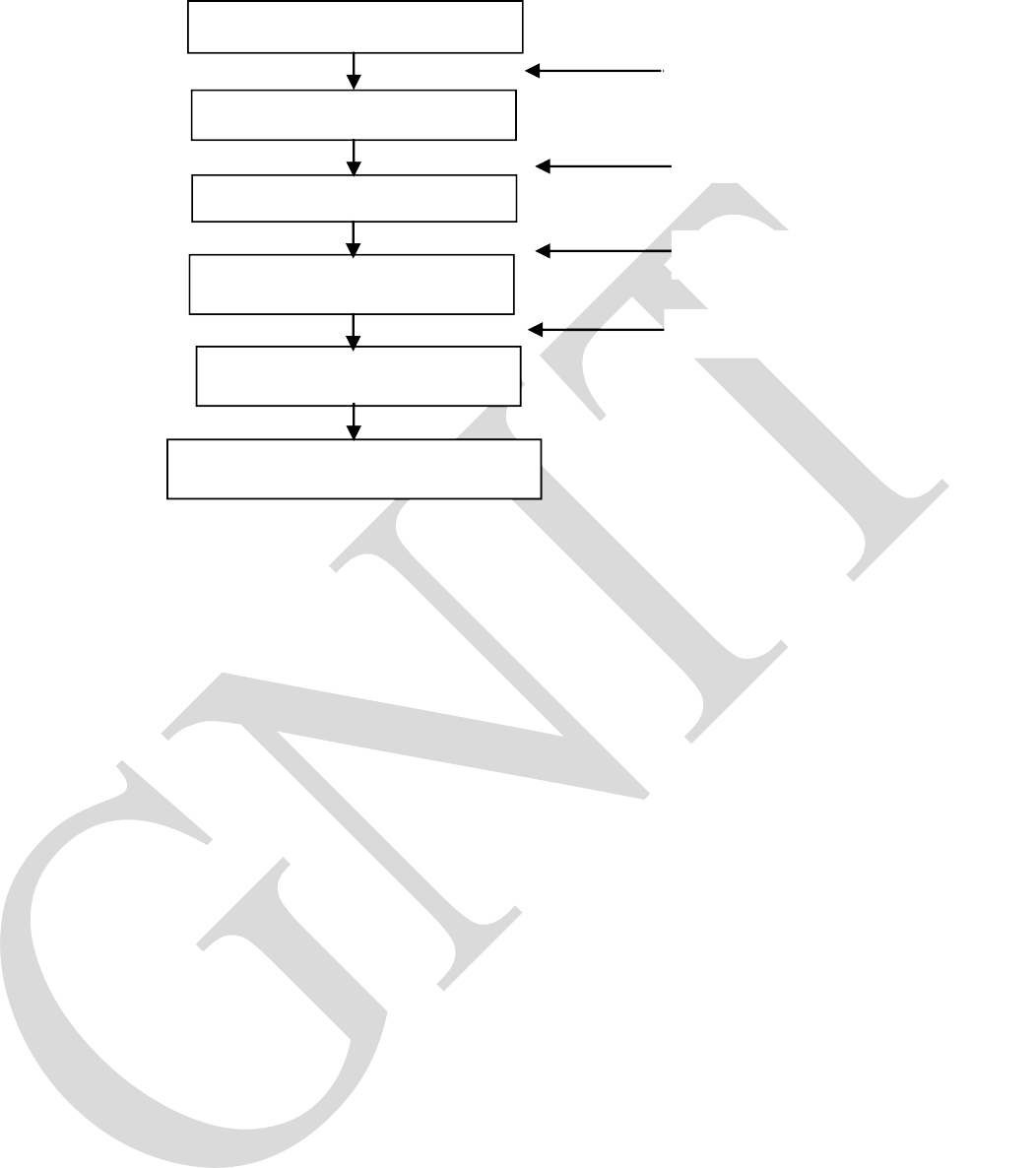
**Recovery manager:** It is responsible for maintaining a log, and restoring the system to a consistent state after a crash.

**System Catalog (Data Dictionary):** It contains all the information about the database. As the name suggests, it is the dictionary of all the data items. It contains description of all the tables, view, data files, indexes, triggers etc.

## 2. INTRODUCTION TO DATABASE DESIGN

* 1. **DATABASE DESIGN AND ER DIAGRAMS**

Database design process can be divided into six steps

Requirements Analysis Conceptual Design Logical Design Schema Refinement

Physical Design

Requirement Specification ER Model

Relational Database Schema

Normalized Data

Application/Security Design

## Requirements Analysis

* Requirements Analysis is the process of determining what the database is to be used for.
* It involves interviews with user groups and other stakeholders to identify what functionality they require from the database, what kinds of data they wish to process and the most frequently performed operations.
* This discussion is at a non-technical level and enables the database designers to understand the business logic behind the desired database.
* Then, the gathered information is organized and presented using suitable tools.

## Conceptual database design:

* Once the information is gathered in the requirements analysis step, a conceptual database design is developed. This step is often carried out using the ER model, or a similar high-level data model.
* The ER Model is used to create a simple description of the data that matches both how users and developers think of the data by identifying entities, relationship between them, attributes and integrity constraints.

## Logical Database Design:

* In this step, the conceptual database design of a database schema is converted into logical data base design.
* That is logical data base design involves translating the ER diagrams into actual relational database schema

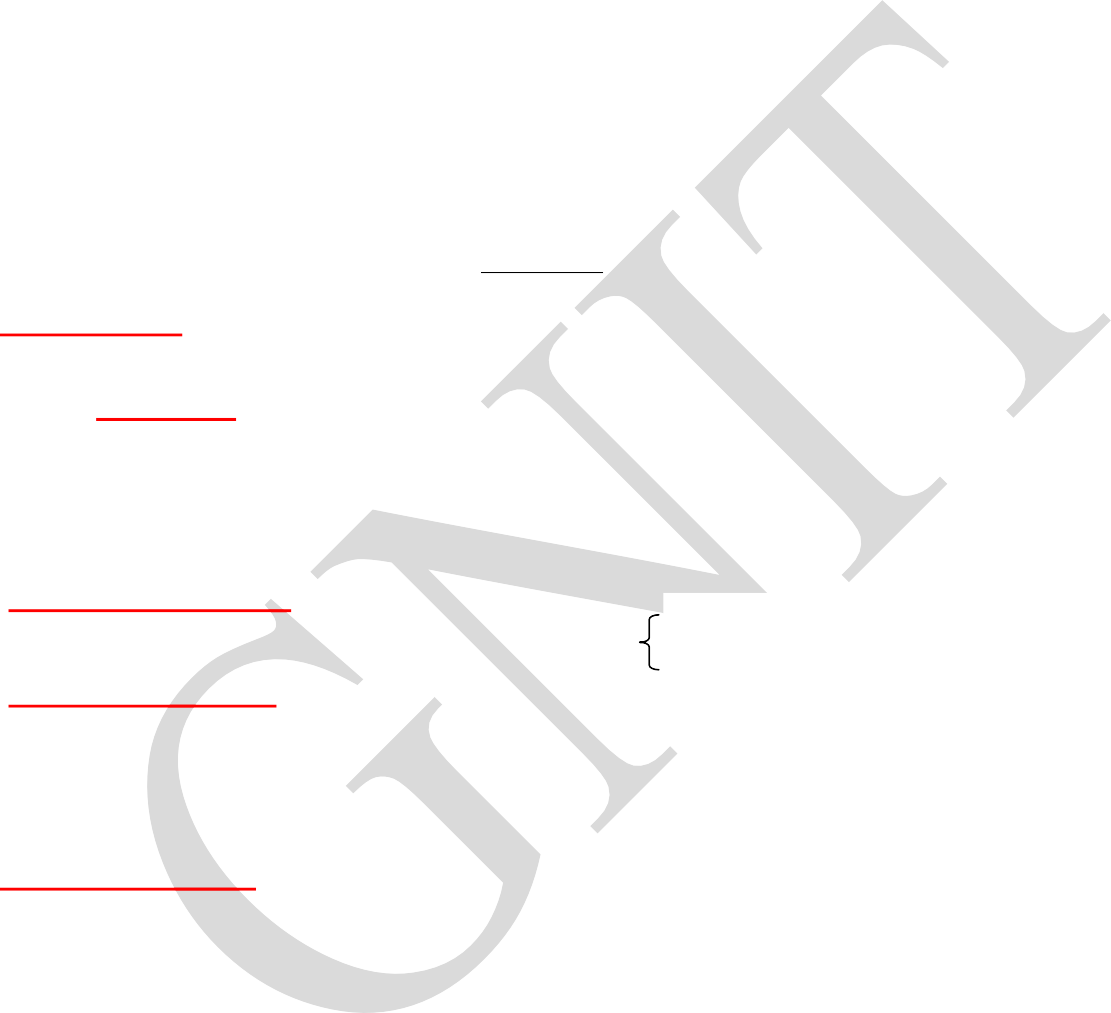
## Schema Refinement:

* The fourth step in data base design is to analyze the collection of relations in our relational database schema to identify the feature problems such as redundancies, anomalies, etc., , and to refine it.

## Physical Database Design:

* In this step, the physical features of the database which includes form of file organization and the internal storage structure are specified.
* This step may simply involve building indexes on some tables and clustering some tables, or it may involve redesign of parts of the database schema obtained from the earlier design steps.

## Application and Security Design :

* + Any soft ware project that involves a DBMS must consider applications that involve processes and identify the entities. **Example:** Users, User groups, departments, etc,.
  + We must describe the role of each entity in every process. As a Security design, for each role, we must identify the parts of the database that must be accessible and the parts of the database that must not be accessible by enforcing access rules.

## ENTITIES, ATTRIBUTES AND ENTITY SETS ENTITY

An entity refers to an object in the real world that is distinguishable from other objects. An entity is

described (in database) using a set of *attributes*.

***ENTITY SET*:** A collection of similar entities, e.g., all employees.

All entities in an entity set have the same set of attributes In ER diagram,

* + - **Attributes** describe the properties of entities in the entity set. (Eg. Attributes of Employees entity are SSN, Name, Lot, etc.,
    - Each attribute has a *domain*.
    - Based on the values of certain attributes, an entity can be identified uniquely.

## Types of Entity Sets-

An entity set may be of the following two types-

## 1. Strong Entity Set

1. Strong entity set
2. Weak entity set

A strong entity set is an entity set that contains sufficient attributes to uniquely identify all its entities. i.e) a primary key exists for a strong entity set.

* + Primary key of a strong entity set is represented by underlining it.

## Weak Entity Set

* + A weak entity set is an entity set that does not contain sufficient attributes to uniquely identify its entities. i.e) a primary key does not exist for a weak entity set.
  + However, it contains a partial key called as a **discriminator.**

## RELATIONSHIPS AND RELATIONSHIP SETS

A **relationship** is defined as an association among several entities.

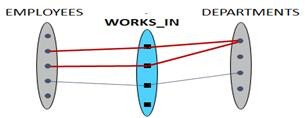
Example**-** „Works in‟ is a relationship that exists between entities Employee and Department.

# Employee

Works\_in

# Department

## Relationship Set-



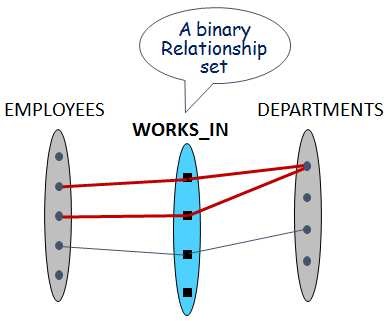
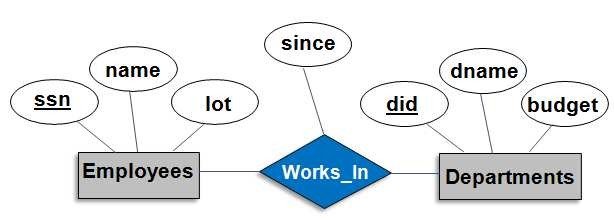
A relationship set is a set of relationships of same type.

## Example-

Set representation of above ER diagram is-

## Degree of a Relationship Set-

The number of entity sets that participate in a relationship set is termed as the degree of that relationship set. Thus, On the basis of degree of a relationship set, a relationship set can be classified into the following types-

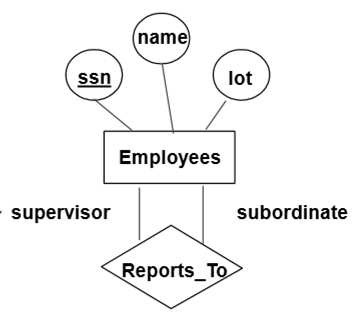


* Unary Relationship Set / Self Relationship set
* Binary Relationship Set
* Ternary Relationship Set
* N-ary Relationship Set

## Types of Relationship Sets-

1. **Unary Relationship Set-**

An entity set can participate in a relationship set with itself. Entities in same set play different roles in the relationship. Role indicators express the role.



## Example-

Role Indicator

## Binary Relationship Set-

Binary relationship set is a relationship set where two entity sets participate in a relationship set.

**Example-** Employees works in Departments

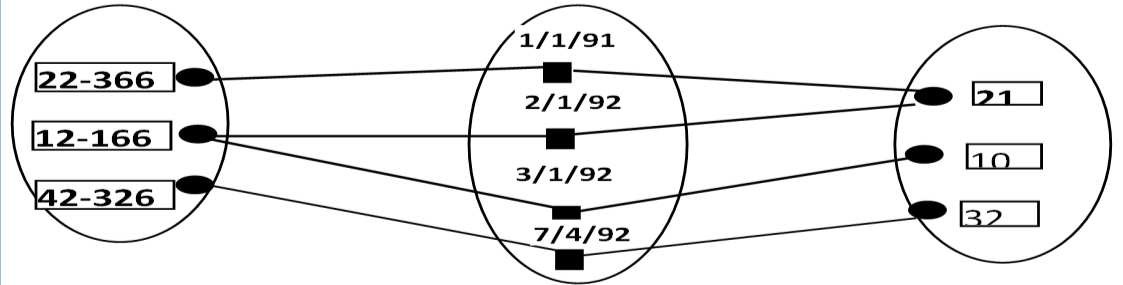
## Ternary Relationship Set-

Ternary relationship set is a relationship set where three entity sets participate in a relationship set.

## Example-

* + If Each department has offices in several locations
  + We want to record the locations at which each employee works

## Descriptive Attributes:



A relationship can also have some

attributes, which are called as „descriptive attributes‟. These are used to record information about

the relationship.

## Example:

James of „Employees‟ entity set works in a department entity set since 1991.

## N-ary Relationship Set-

N-ary relationship set is a relationship set where „n‟ entity sets participate in a relationship set.

## Instance:

An instance of a relationship set is a set of relationships. It is a snapshot of the relationship at some instant of time.

## ADDITIONAL FEATURES OF ER MODEL

**2.4.1.a.) Cardinality Constraint**

Cardinality is the number of entity instances to which another entity set can map under the relationship

## Types of Cardinality Ratios-

1. Many-to-Many cardinality (m:n)

A ***key constraint*** between an entity set S and a relationship set restricts instances of the relationship set by requiring that each entity of S participate in at most one relationship.

1. Many-to-One cardinality (m:1)
2. One-to-Many cardinality (1:n)
3. One-to-One cardinality (1:1 )

## Many-to-Many Cardinality-

By this cardinality constraint,

* + An entity in set A can be associated with any number (zero or more) of entities in set B.
  + An entity in set B can be associated with any number (zero or more) of entities in set A.

## Example-

Name

Street

City DNo

Dname

Budget

Works\_in

Employees

Departments

An employee is allowed to work in different departments and a department is allowed to have several employees.

## Many-to-One Cardinality-

By this cardinality constraint,

* + An entity in set A can be associated with at most one entity in set B.
  + An entity in set B can be associated with any number (zero or more) of entities in set A.

## Example-

Departments

Employees

Street

Dname

Name City DNo

Budget

Works\_in

Each employee works in at most one department. Many employees can work in same department.

## One-to-Many Cardinality-

By this cardinality constraint,

* + An entity in set A can be associated with any number (zero or more) of entities in set B.
  + An entity in set B can be associated with at most one entity in set A.

## Example-

One employee can be associated with many departments, where as each department can be associated with at most one employee as its manager.

Street

Name

City

Dname

~~DNo~~

Budget

Employees

Manages

Departments

## One-to-One Cardinality-

By this cardinality constraint,

* + An entity in set A can be associated with at most one entity in set B.
  + An entity in set B can be associated with at most one entity in set A.

## Example-

Each employee can manage at most 1 department.

Manages

Dname

DNo

Budget

Departments

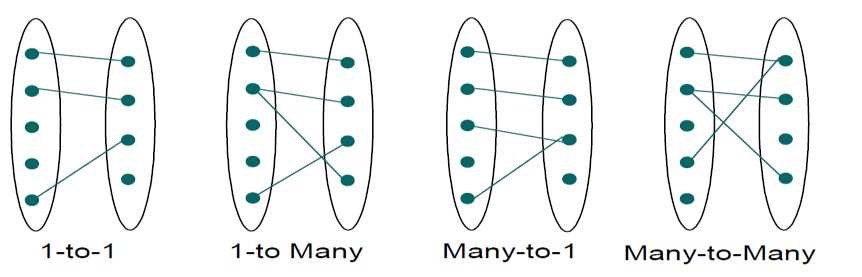
Street

Name

City

Employees

## 2. 4. 1.b ) Key Constraints



A key constraint between an entity set S and a relationship set restricts instances of the relationship set by requiring that each entity of S participate in at most one relationship.

Many-to- one (and) One-to-many cardinality constraints are also known as Key constraints.

## Example : 1(Many to One Cardinality constraint)

Employees

Street

Name

City

Dname

DNo

Budget

Works\_in

Departments

Each employee works in at most one department. Many employees can work in same department.

## Example : 2(One to Many Cardinality constraint)

Employees

Departments

Street

Name

City

Manages

Dname

DNo

Budget

One employee can be associated with many departments, where as each department can be associated with at most one employee as its manager.

## Participation Constraints:

* + - * A participation constraint between an entity set S and a relationship set specifies either each entity of an entity set must participate in at least one relationship or not.

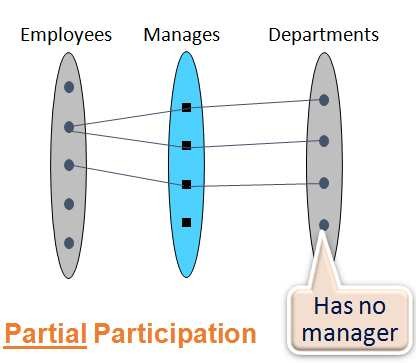
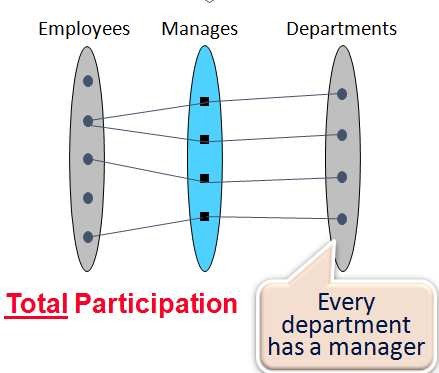
There are 2 types of participation constraints, which are as below.

1. Total Participation
2. Partial Participation

## Total Participation

* + In total participation every entity in the entity set participates in at least one relationship in the relationship set.

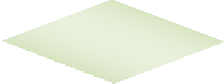
## Example:



* + Every department must have a manager. So “Departments” entity set has total participation

in Manages relationship

* + Represented as thickened line (there is a key constraint as well)



Employees

Departments

Street

Dname

Name City DNo Budget

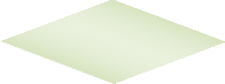
Manages

## Partial Participation

* + In partial participation some entities may not participate in any relationship in the relationship set.
  + It is indicated by single line.

## Example:

* + Not every department has a manager, So “Departments” entity set has partial participation



Departments

Employees

Street

Dname

Name City DNo Budget

Manages

## Weak Entity Set

Entity types that do not have key attributes of their own are called as **weak entity types.**

A weak entity set can be identified uniquely only by considering the primary key of another entity set, known as identifying entity set or owner entity set.

For any weak entity set, following restrictions must hold.

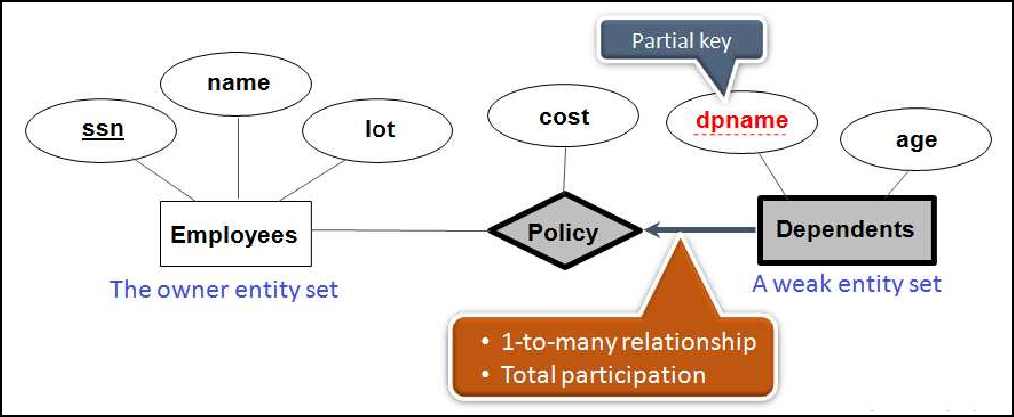
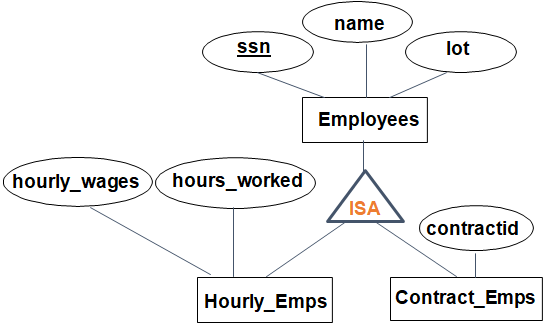
* + - 1. The owner entity set and the weak entity set must participate in a **One-to-many relationship set**, which is called as the „Identifying Relationship Set‟ of the weak entity set.
      2. The weak entity set must have **total participation** in the identifying relationship set.

## Partial key(or discriminator) of the weak entity set:

The set of attributes of a weak entity set that uniquely identify a weak entity together with the primary key of the owner entity set is called as „partial key of the weak entity set‟.

**Primary Key of Weak Entity Set** = Its own Partial Key + Primary Key of Owner entity

## Representation



**Dark lines:** to draw weak entity set & its identifying relationship set (with one-many relationship constraint)

**Broken line:** to underline a partial key

## Example:

„Dependents‟ is an example of a weak entity set which is uniquely identified by (ssn, pname)

**‘dpname’** is a partial key for dependents. **Policy**: identifying relationship set

## Class Hierarchies

Class hierarchies organize structurally similar entities through inheritance into sub- and superclasses using ISA symbol.

* If we declare A **ISA** B, every **A** entity is also considered to be a **B** entity.

## Reasons for using ISA:

* To add attributes specific to a sub-entities only, instead of adding all the Attributes common to different sub-entity sets .
* To identify entities participating in a relationship so as to create a more concise and readable E-R diagram.
* It best maps to object oriented approaches either to databases or related applications.

## Class Hierarchy Based On Sub-super Set Generalization

* + Generalization is like a bottom-up approach in which two or more entities of lower level combine to form a new entity at higher level if they have some attributes in common.
  + i.e., In generalization, entities are combined to form a more generalized entity, i.e., subclasses are combined to make a superclass.

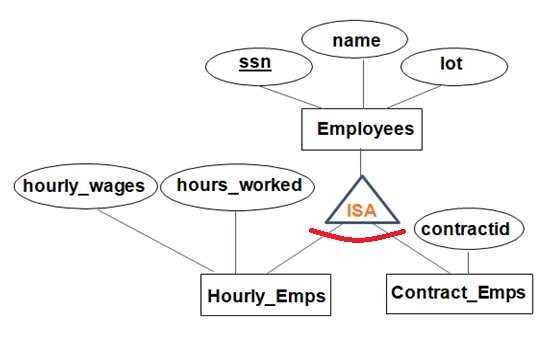
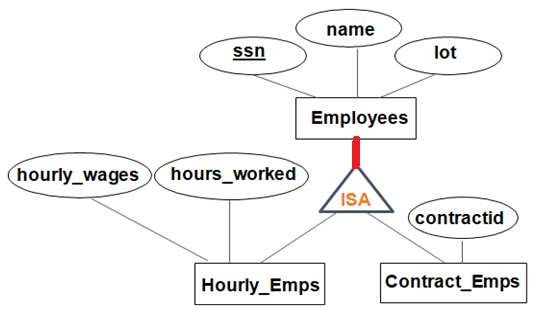
**Example:** Hourly-Emps and Contract-Emps can be generalized and create a higher level entity EMPLOYEE.

## Specialization :

* + Specialization is a top-down approach in which one higher level entity can be broken down into a set of lower level entities.
  + It is the process of identifying the subsets of a superclass that share some special attributes.
  + Normally, the superclass is defined first, the subclass and its related attributes are defined next, and relationship set are then added.

**Example:** EMPLOYEE entity can be specialized as Hourly\_Emps and Contract\_Emps.

## Class Hierarchy Based On Constraints



1. **Overlap constraints(allowed/disallowed)** :Overlap constraints determine whether two subclasses are allowed to contain the same entity commonly.

## Example :

* Can John be both an Hourly\_Emps entity and a Contract\_Emps entity?...Disallowed
* Can he be both a Contract\_Emps entity and a Senior\_Emps entity? ...Allowed



Overlap Constraint

1. **Covering Constraints(Yes/No)** : Covering constraints determine whether all entities in the superclass is collectively included in at least one of the subclasses.





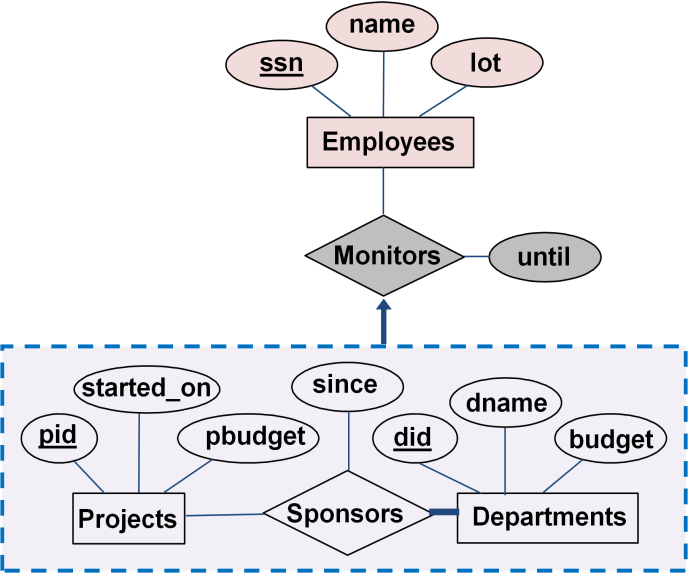
Covering Constraint

## Example:

* + Should every Employee be a Hourly-Emps or Contract-Emps? ….. **No**.
  + Does every hourly employee and contract employee are employees of this organization? … **Yes**

## 2.4.5. Aggregation:

Aggregration is the process of transforming a relationship set into an entity set for the purpose of relating the **resulting construct to other entity sets**.



It is used to model a relationship involving a

*relationship set*

**Representation:** dashed box

## Example:

* + - Consider the constraint that each sponsorship is monitored by at most one employee
    - Sponsors and Monitors are two distinct Relationships
    - Here the **relationship set Sponsors** can be treated as an entity set for the purpose of relating it with **Monitors relationship set**

## CONCEPTUAL DESIGN USING THE ER MODEL Design choices:

Developing an ER diagram presents several design issues, including the following:

* Should a concept be modeled as an entity or an attribute?
* Should a concept be modeled as an entity or a relationship?
* Identifying relationships: Binary or ternary? Aggregation?

## Constraints in the ER Model:

* A lot of data semantics can (and should) be captured.

## Entity vs 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

## Example:

* **Should *address* be:**

– attribute of Employees or an entity (related to Employees)?

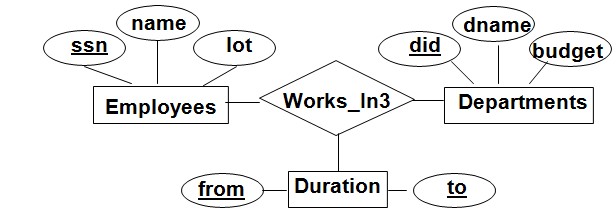
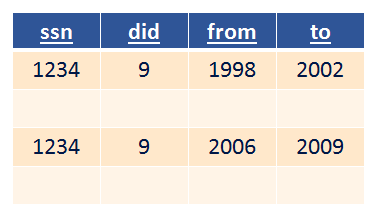
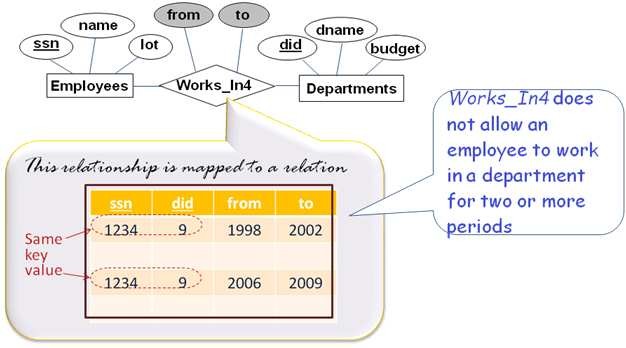


## Depends upon use of address information, and the semantics of the data:

* If several addresses per employee, *address* must be an entity (since attributes cannot be set- valued).
* If structure (city, street, etc.) is important, *address* must be modeled as an entity (since attribute values are atomic).

## Entity vs Relationship

* + **Works\_In4 does not allow an employee to work in a department for two or more periods.**



* If wanting to record several working periods for an employee in Work\_In4 Introduce new entity set, Duration

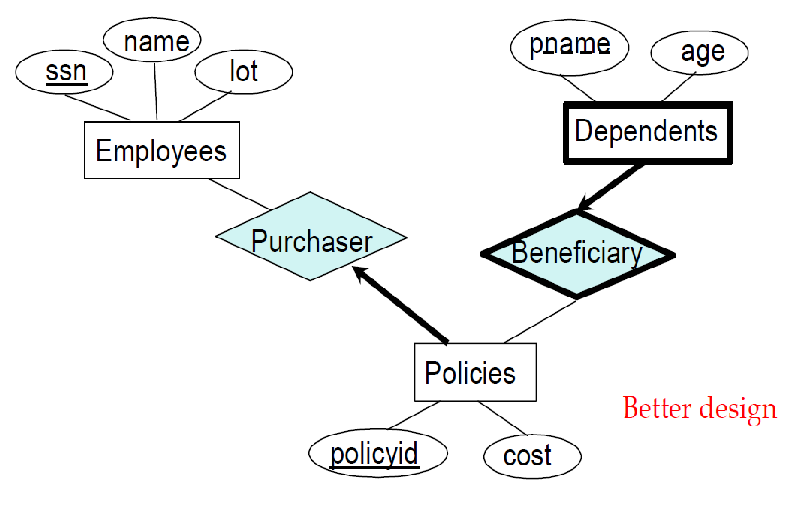
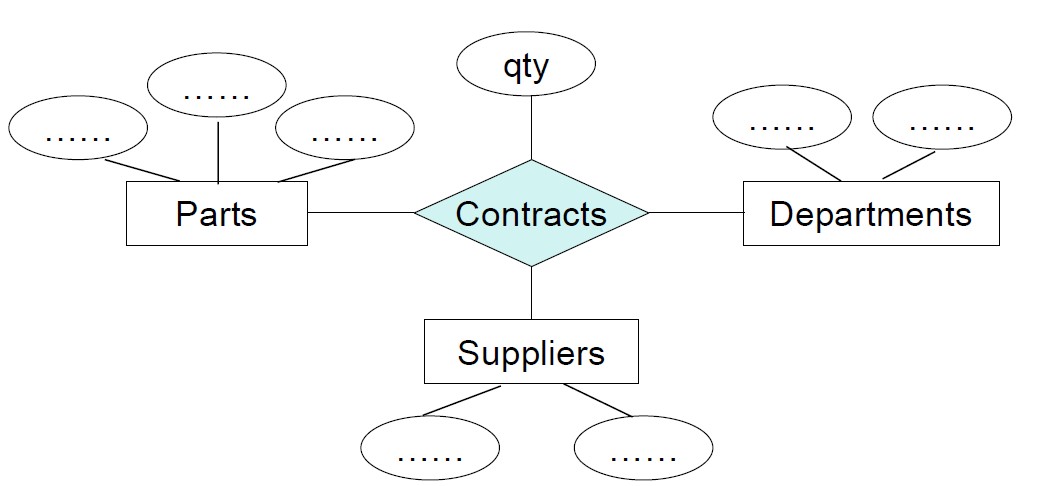
## Binary vs. Ternary Relationships Constraints:

* + An employee can own several policies. Each policy can

be owned by several employee.

* + Each dependent can be covered by several policies
* To specify the above given constraints, ternary relationship is suitable.
* But to specify the below given constraints, this design is **bad design.**

## Constraints:



* Each policy is owned by just 1 employee, and,
* Dependents is a weak entity set, and each dependent is tied to the covering policy

## Better design

In this example: two binary relationships are better than one ternary relationship to specify the above given constraint

## Another example

The contract specifies that a supplier will supply some quantity of a part to a department.

To specify this constraint, a ternary relationship is better than three binary relationships