DATABASE MANAGEMENT SYSTEM UNIT-2 NOTES-2021

UNIT – II: Data Modeling Using Entity-Relationship Model **11 Hours**

Overview of Data Design Entities, Attributes and Entity Sets, Using High-Level Conceptual Data Models for Database Design, An Example Database Application, Entity Types, Entity Sets, Attributes and Keys, Relationship Types, Relationship Sets, Roles and Structural Constraints, Weak Entity Types, Refining the ER Design Company Database Diagrams, Naming Conventions and Design. Issues, File organization and indexing, type of single level ordered index, multi-level indexes, indexes on multiple keys.

Overview of Database Design, Entities, Attributes and Relationships

Database Design is a collection of processes that facilitate the designing, development, implementation and maintenance of enterprise data management systems

It helps produce database systems

1. That meet the requirements of the users

2. Have high performance.

The main objectives of database designing are to produce logical and physical designs models of the proposed database system.

- The logical model concentrates on the data requirements and the data to be stored independent of physical considerations. It does not concern itself with how the data will be stored or where it will be stored physically.
- The physical data design model involves translating the logical design of the database onto physical media using hardware resources and software systems such as database management systems (DBMS).

Entity

An entity can be a real-world object, either animate or inanimate, that can be easily identifiable. For example, in a school database, students, teachers, classes, and courses offered can be considered as entities. All these entities have some attributes or properties that give them their identity.

An entity set is a collection of similar types of entities. An entity set may contain entities with attribute sharing similar values. For example, a Students set may contain all the students of a school; likewise a Teachers set may contain all the teachers of a school from all faculties. Entity sets need not be disjoint.

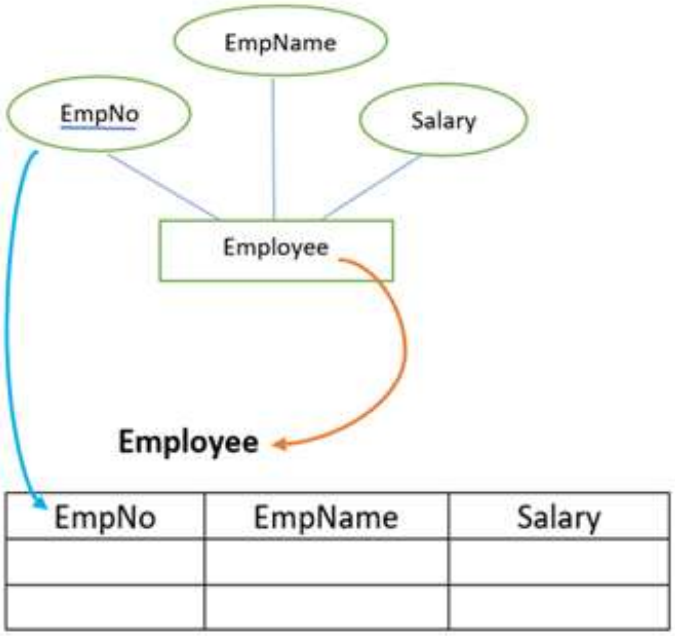
Attributes

Entities are represented by means of their properties, called attributes. All attributes have values. For example, a student entity may have name, class, and age as attributes. There exists a domain or range of values that can be assigned to attributes. For example, a student's name cannot be a numeric value. It has to be alphabetic. A student's age cannot be negative, etc.

Q. What is an entity?

Entity: The E-R model describes data as entities, relationships and attributes:
An **entity** is a thing in the real world with an independent existence that is distinguishable from all other objects. Example: Each employee in an organization is an entity. A company, a job, a book, etc all are entities. Each entity has particular properties called attributes that describes it.

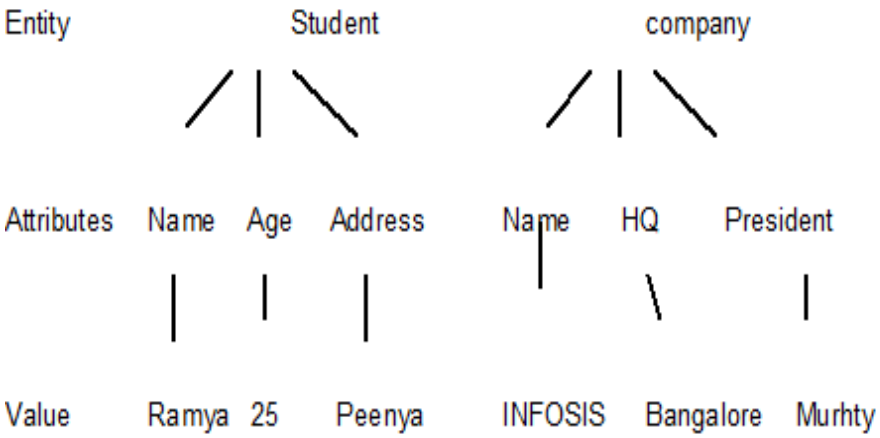
DATABASE MANAGEMENT SYSTEM UNIT-2 NOTES-2021



Q. What is an attribute?

An **attribute** is a property that describes an entity.
Example:
1. The attributes of a person's entity are his name, address, job, salary etc. for an entity each attributes will have a value.
2. **student (regno, name, age, address)**
3. **Employee (ID, name, dept, salary)**
4. **Company (name, HQ, president)**

Attributes:



Q. Explain the various types of attributes?

Types of Attributes

- Simple Attribute
- Composite
- Single valued Attribute
- Multivalued Attribute
- Stored attribute
- Derived attribute
- Complex
- Null value Attribute
- Key attribute

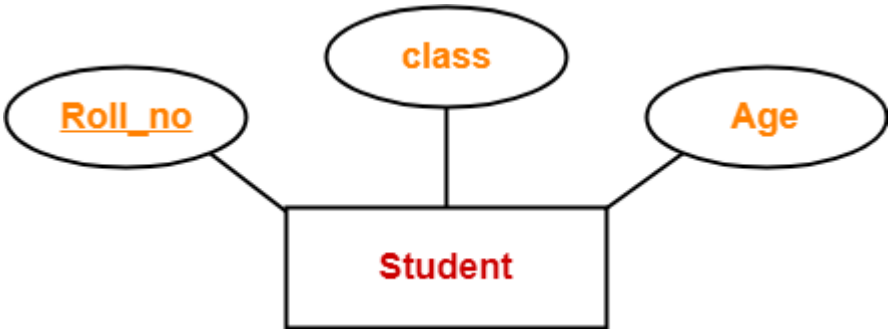
Each attribute is associated set of values called domain.

TYPES OF ATTRIBUTES:

Simple or atomic attributes: These attributes can't be subdivided into further. In other words, the atomic attributes hold **single value and not composed of any attributes**.

Example: Regno, ID, DOB, age, Deptno, sex.

For example, a student's phone number is an atomic value of 10 digits.

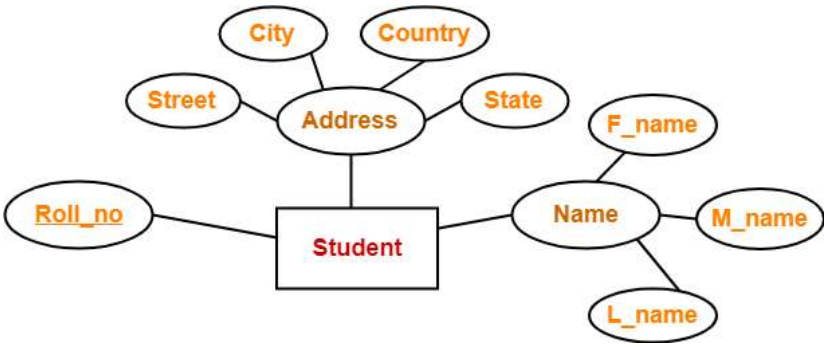


Composite attributes: Composite attributes can be divided into smaller subparts, which represent more basic attributes with independent meanings.

Example:

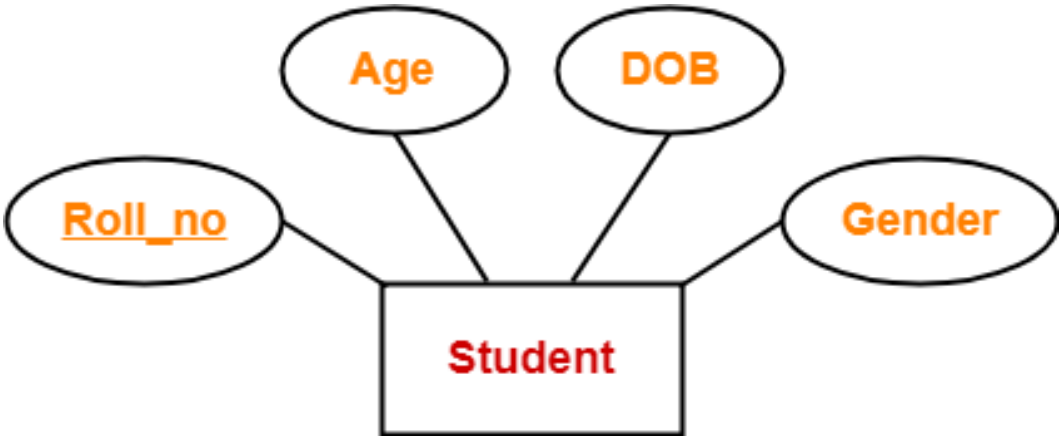
1. The address attribute can be subdivided into **house no, street_name, area, city and country.**

2. The name attributes, which can be composed of **first name, middle name, and last name.**

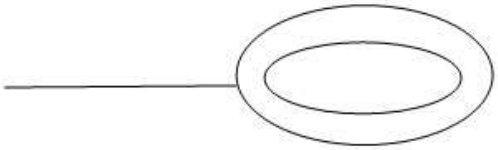


Single valued attributes: An attribute that can take only one value at a time is called single valued attributes. Usually, for a particular entity, each attributes will have single value.

Example: The age attribute will have a single value.

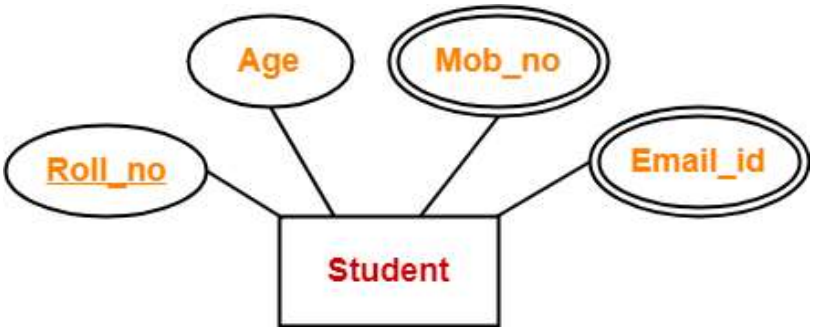


Multi valued attributes: Some attribute have more than one value for the same attribute and are called multi valued attributes.



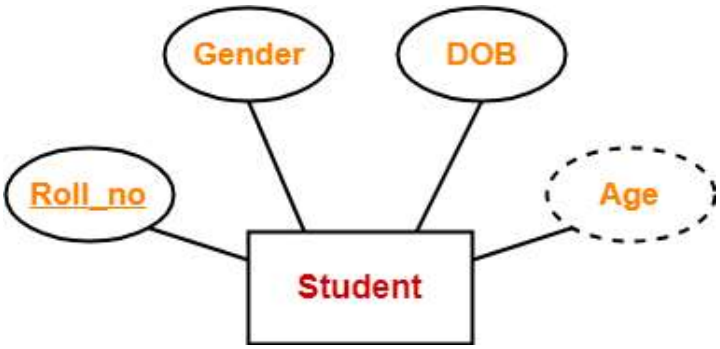
Example: A **college degree** attribute can have multiple values.

Degree [B,Sc, MCA, PhD]
Admin [Bangalore, Mumbai]
Carcolor [Red, Black].



Derived attributes: If the value of an attribute can be derived from some other attributes, then such attributes are called **derived attributes**. Since age can be determined by knowing birthday and current date. **The age attribute is said to be derived attribute.**

Another example**, gross pay of an employee this can be derived by knowing basic pay, allowances, and deduction for employee**.



Stored attributes: The value of certain attributes cannot be obtained or derived from some other attributes, that is, they are not derived from any other attributes, these types of attributes is called as stored attributes.

Example: Birth date, Book_ID, SSN, Part#.

The attribute SSN is direct attribute and is not dependent on any other attribute or derivable from one or more other attributes.

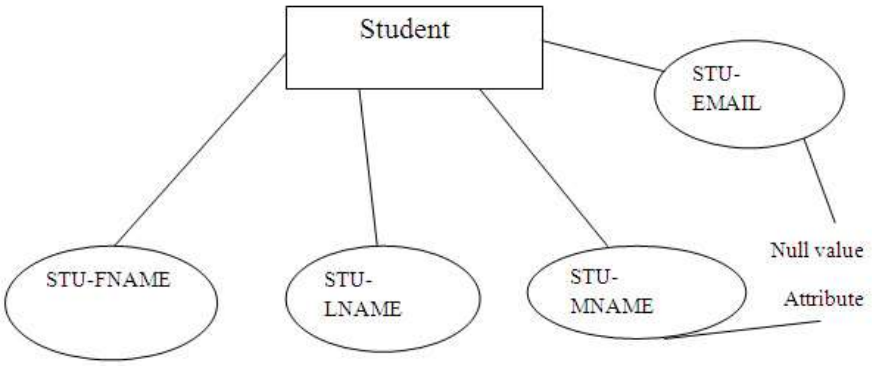
Complex attribute: A attribute which is both composite and multivalued. complex Attribute is a type of attribute in database.

For example, **A person can have many phone numbers, many e-mail addresses, home addresses etc**. So if there is a attribute in the name of 'Contact detail, It can be a complex attribute.

Example: A person can have more than one residence; each residence can have more than one phone.

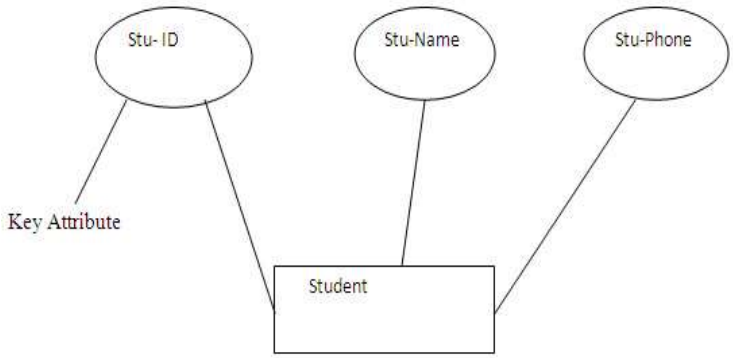
Null attributes: A null attributes used when an attributes does not have any value. A null value does not mean that the value is equal to zero but indicate no value is stored for that attribute.

Example: Abasement number attribute of an address applies only to address that are in apartment buildings, and not to other types of residence such as single family home. Email: All employees in an employee database may not have e-mail address.

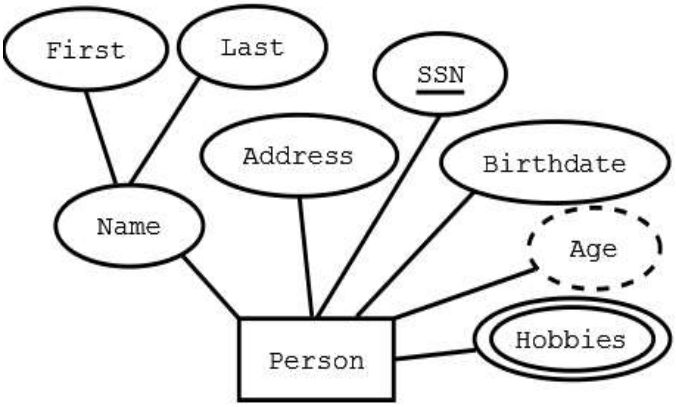


Key Attributes: This attribute represents the main characteristic of an entity i.e. primary key. Key attribute has clearly different value for each element in an entity set.

Example: The entity student ID is a key attribute because no other student will have the same ID.



Graphical Representation in E-R diagram



Rectangle -- Entity

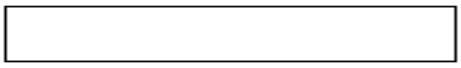
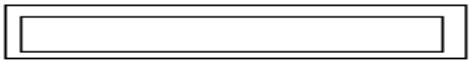
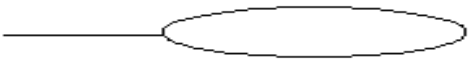
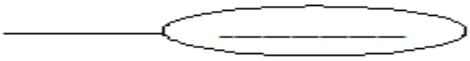
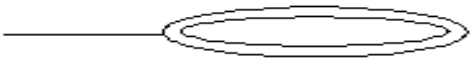
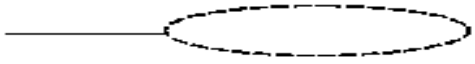
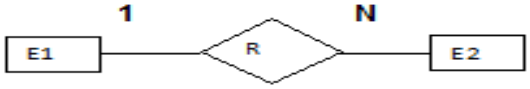
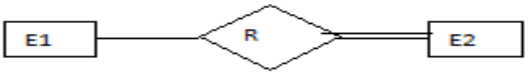
Ellipses -- Attribute (underlined attributes are [part of] the primary key)

Double ellipses -- multi-valued attribute

Dashed ellipses-- derived attribute, e.g. age is derivable from birthdate and current date.

ER MODEL NOTATION

DATABASE MANAGEMENT SYSTEM UNIT-2 NOTES-2021

ER Symbols :	Meaning
	Entity
	Weak Entity
	Relation ships
	Attributes
	Key attribute
	Multi valued attribute
	Composite attributes
	Derived attribute
	Cardinality ratio 1:N E1, E2 in R
	Total participation E2 in R

Q. Define the following?

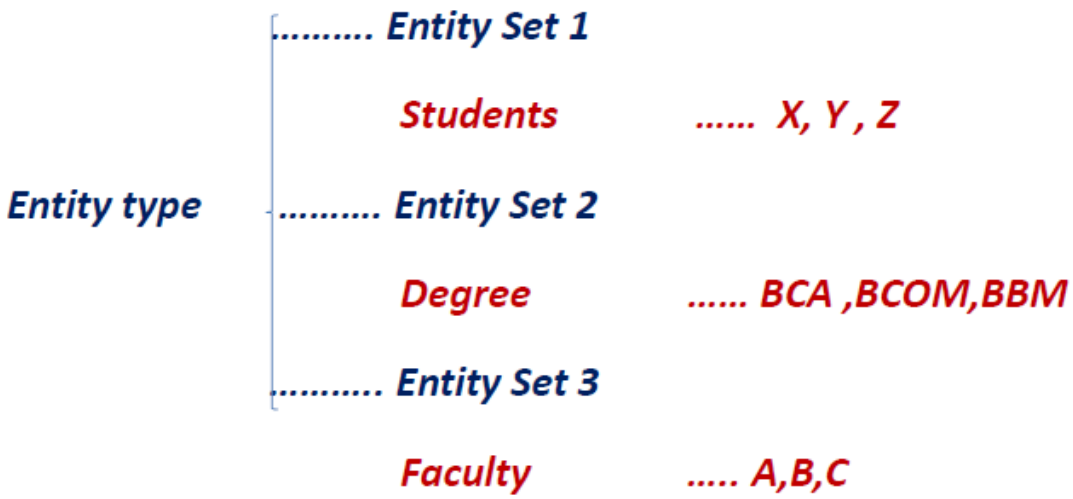
DATABASE MANAGEMENT SYSTEM UNIT-2 NOTES-2021

Entity types, Entity sets, Key attributes, value sets.

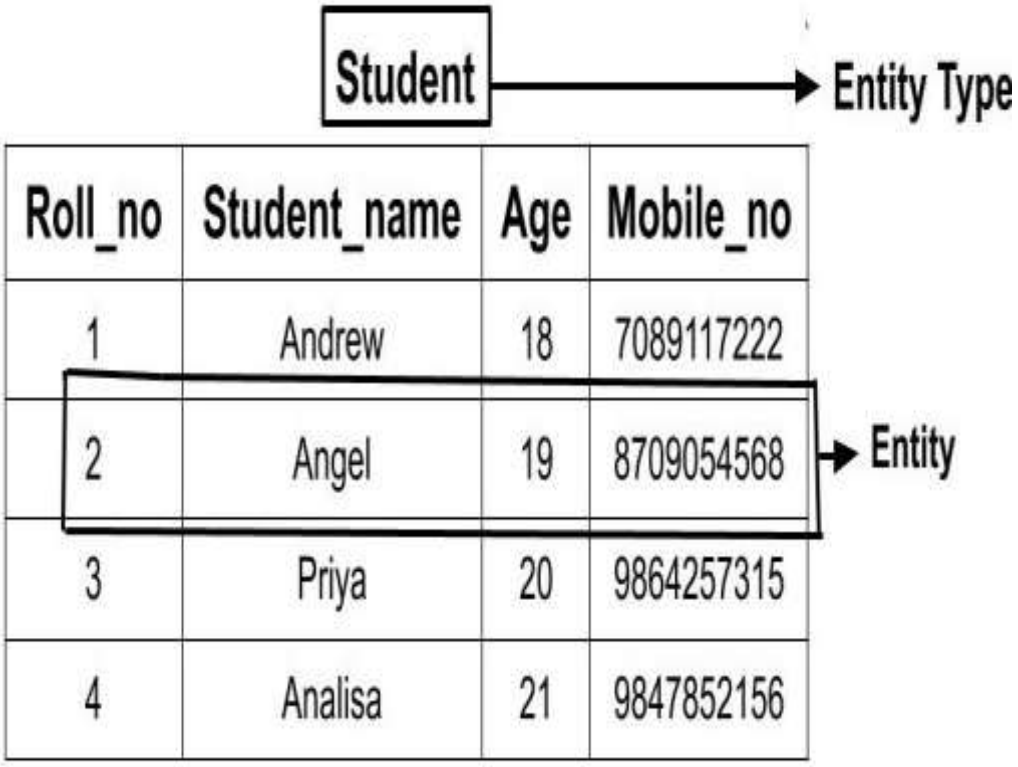
Entity set: It is a set of entities of the same type that share the same properties or attributes. The set of all employees working for the same department can be defined as entity set employee but each entity has own values for each attributes.

Entity Type: An entity type defines a set of entities that have the same attributes. Each entity type in the database is described by its name and a list of attributes.

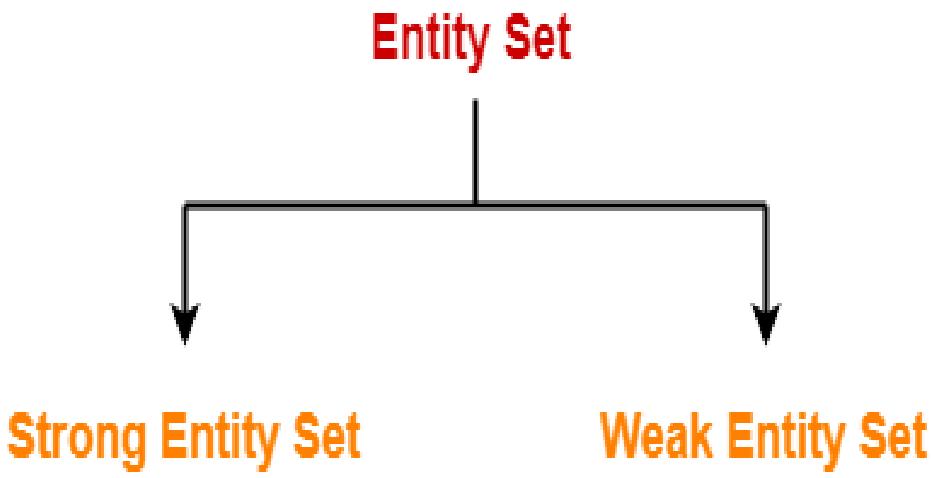
Example :



An entity type describes the schema or intention for a set of entities that share the same structure. The individual entities of a particular entity type are grouped into a collection or entity set. Which is also called the extension of the entity type.



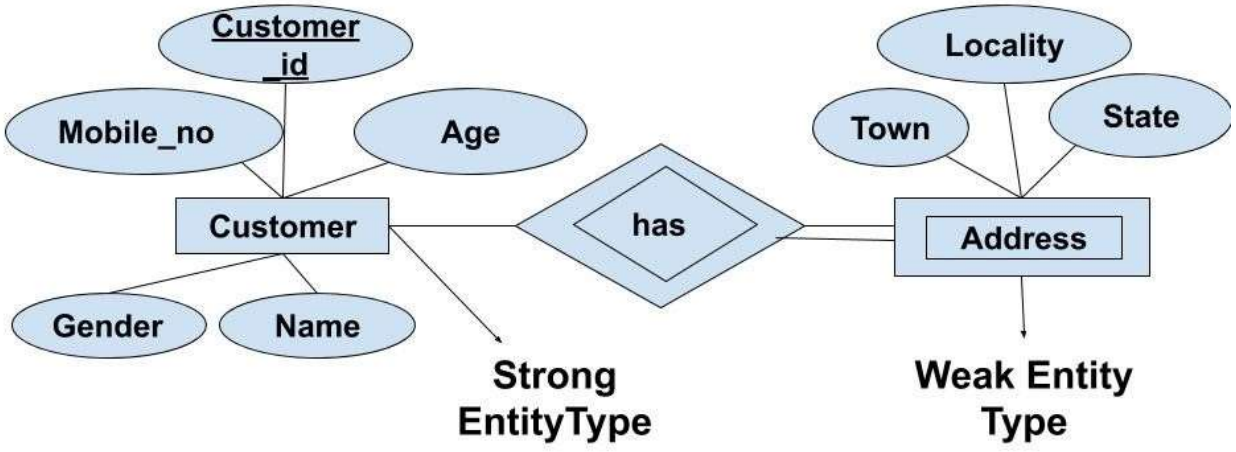
ENTITY TYPES



The **entity** can be classified into two basic categories, strong **entity set** and weak **entity set**.

An **entity set** that has a primary key is the strong **entity set**.

An **entity** that doesn't have a primary key is a weak **entity set**. A weak **entity set** is dependent on one strong **entity set** to distinguish its **entities**.



There are two types of weak entities: **associative entities and subtype entities**.

An associative entity is uniquely identified by concatenating the primary keys of the two entities it connects.

For example, the number of hours the Employee worked on a particular Project is an attribute of the relationship between Employee and Project, not of either Employee or Project.

Subtype: where one entity (the subtype) inherits the attributes of another entity (the supertype).

Relationships between an entity supertype and its subtypes are always described as "is a." For example, Employee is a Permanent Employee, Employee is a Part-time Employee.

Key attributes of an entity type:

- A key attribute is a minimal set of attributes of an entity set, which uniquely identifies an entity in an entity set. A key may be a single attributes or may be more than one attribute.

An entity type usually an attribute whose values are distinct for each individual entity such an attribute is called a key attribute. Whose values can be used to identify each entity.

Example:

For the company entity, name can be the key attribute because no two companies can have the same name. For the student entity, regno can be the key attribute.

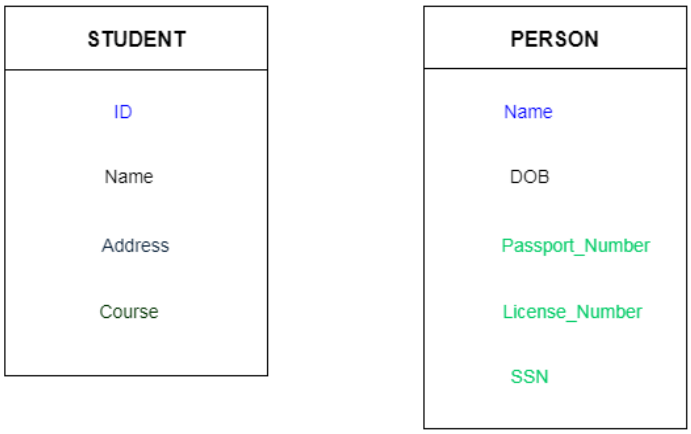
Value sets (Domain) of attributes:

A set of values that may be assigned to the attributes of each individual entity, in an entity set is called the value set or domain.

Keys play an important role in the relational database.

- It is used to **uniquely identify any record or row of data from the table.** It is also used to establish and identify relationships between tables.

- **For example:** In Student table, ID is used as a key because it is unique for each student.
- In PERSON table, passport_number, license_number, SSN are keys since they are unique for each person.



Q. Key attributes (Primary key) of an entity type:

A key attribute is a minimal set of attributes of an entity set, which uniquely identifies an entity in an entity set. A key may be a single attributes or may be more than one attribute.
An entity type usually an attribute whose values are distinct for each individual entity such an attribute is called a key attribute. Whose values can be used to identify each entity?

Example:
1. For the company entity, name can be the key attribute because no two companies can have the same name.
2. For the student entity, regno can be the key attribute.

Q. What is weak entity?

Weak Entity

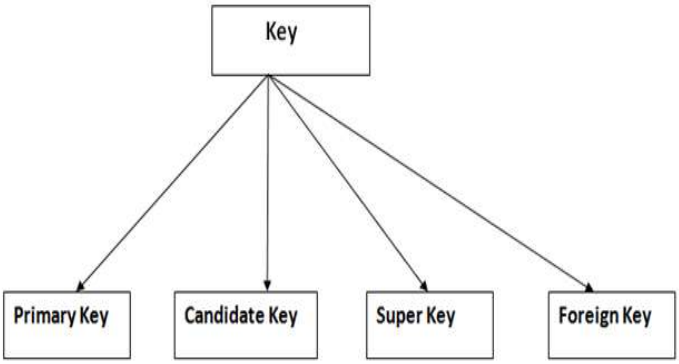
Entities which do not have key attribute (primary key) is known as weak entity weak entity is also known as Child entity or subordinate entity.

Q. What is strong Entity?

Strong entity

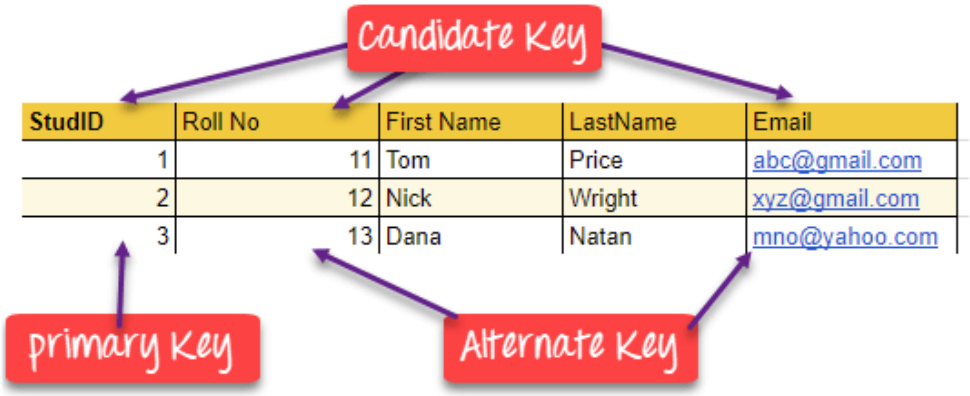
Entities which have key attribute is known as Strong entity

Types of key:



- Super Key: A set of attributes (one or more) that collectively identifies an entity in an entity set.
- Candidate Key: A minimal super key is called a candidate key. An entity set may have more than one candidate key.
- Primary Key: A primary key is one of the candidate keys chosen by the database designer to uniquely identify the entity set.

DATABASE MANAGEMENT SYSTEM UNIT-2 NOTES-2021



1. **Super key:** It is a set of one or more attributes that taken collectively, allows us to identify uniquely an entity in the entity set.

   - The combination of customer_name and social_security_numbers is a super key for the entity set customer.
   - The customer_name attribute is not a super key because several people might have the same name.
   - The social_security_number attribute could also be super key since it is sufficient to distinguish one customer entity from another.

2. Candidate key: A super key may contain extraneous attributes. A super key for which no proper subset is a super key. i.e., minimal super key is called candidate key.

- The combination (**customer_name, SSN**) forms super key, the attribute **SSN alone is a candidate key.**

3. Primary key: It is a candidate key chosen by the database designers as the principal means of identifying entities within an entity set.

4. Prime attribute: An attribute of relation schema **R** is called a prime attribute of **R** if it is a member of some candidate key of **R.**

5. Non prime attribute: An attribute is called non prime if it is not a prime attribute i.e., it is not a member of candidate key.

6. Weak entity set: An entity set not having sufficient attributes to form a primary key is called weak entity set.

7. Strong entity set: An entity set that has a primary key is termed a strong entity set.

8. Foreign key: A set of attributes in a relation is a foreign key if it satisfies the following conditions.

- It should have the some domain as the primary key attributes of another relation schema and is said to refer to this relation.
- A value of foreign key in tuple t1 either occurs as a value of primary key for some tuple t2 in another relation.

9. Composite key: If a key has more than one attribute, it is called composite key.

- **Example:** Consider the relation schema BRANCH with attributes **branch_name, branch_city,** indentify super key, candidate key and primary key.
- **{Branch_name, Branch_city}** à **Super key.**
- **{Branch_name}** à **Candidate key.**
- **{Branch_name}** à **Primary key.**

SUMMARY OF KEYS:

- Super Key - A super key is a group of single or multiple keys which identifies rows in a table.
- Primary Key - is a column or group of columns in a table that uniquely identify every row in that table.
- Candidate Key - is a set of attributes that uniquely identify tuples in a table. Candidate Key is a super key with no repeated attributes.
- Alternate Key - is a column or group of columns in a table that uniquely identify every row in that table.
- Foreign Key - is a column that creates a relationship between two tables. The purpose of Foreign keys is to maintain data integrity and allow navigation between two different instances of an entity.
- Compound Key - has two or more attributes that allow you to uniquely recognize a specific record. It is possible that each column may not be unique by itself within the database.
- Composite Key - An artificial key which aims to uniquely identify each record is called a surrogate key. These kind of key are unique because they are created when you don't have any natural primary key.

- Surrogate Key - An artificial key which aims to uniquely identify each record is called a surrogate key. These kind of key are unique because they are created when you don't have any natural primary key.

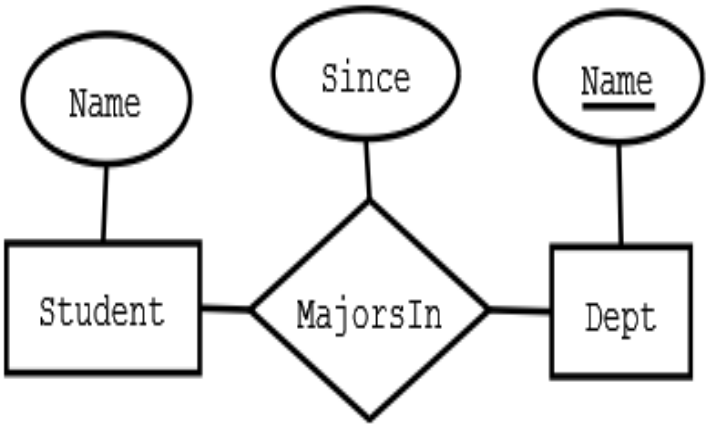
Relationships

Relationship: connects two or more entities into an association/relationship

- "John" majors in "Computer Science"

Relationship Type: set of similar relationships

- *Student* (entity type) is related to *Department* (entity type) by *MajorsIn* (relationship type).



Relationships types, sets and instances:

RELATIONSHIP Definition: A relationship is an association among two or more entities. A relationship captures how two or more entities are related to one another.

Example:

An **'owns'** relationships between a company and computer

A **'Supervises'** relationships between an employee and department.

A **'Performs'** relationships between an artist and song.

A **'Proved'** relationship between a mathematical and a theorem.

Relationships are represented as diamonds symbol, connected by lines to each of the entities in the relationship in an ER diagram.

- ***Relationship:***
  - association between multiple entities

Example-

'Enrolled in' is a relationship that exists between entities **Student** and **Course**.

Relationship types: A relationships type R among 'n' entity' type E1, E2….. Defines a set of associations among entities from these types.

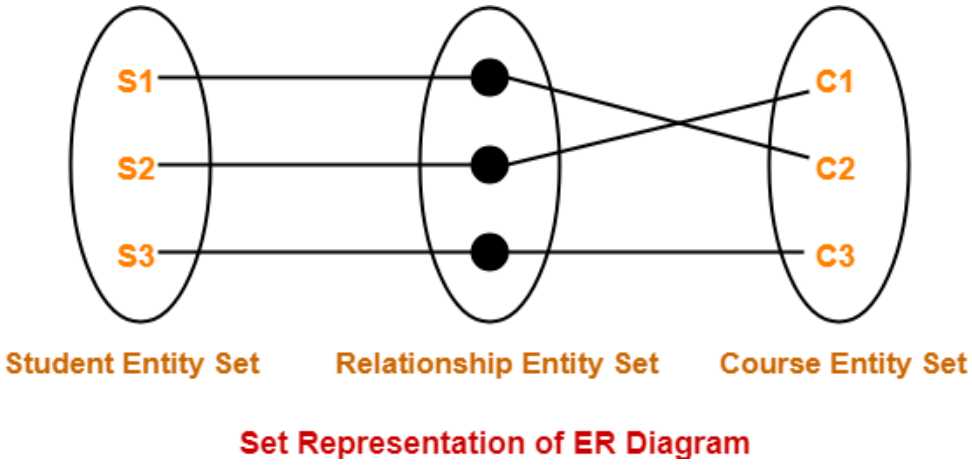
Example: Relationship type **works_for** between two entities of company database, namely employee and department.

Relationship Instance: Each relationship instance **ri** is an association of entities where the association includes exactly one entity from each participating entity type.

Example: Each relationship instance in **works_for** relationship type associates and employee entity and one department entity.

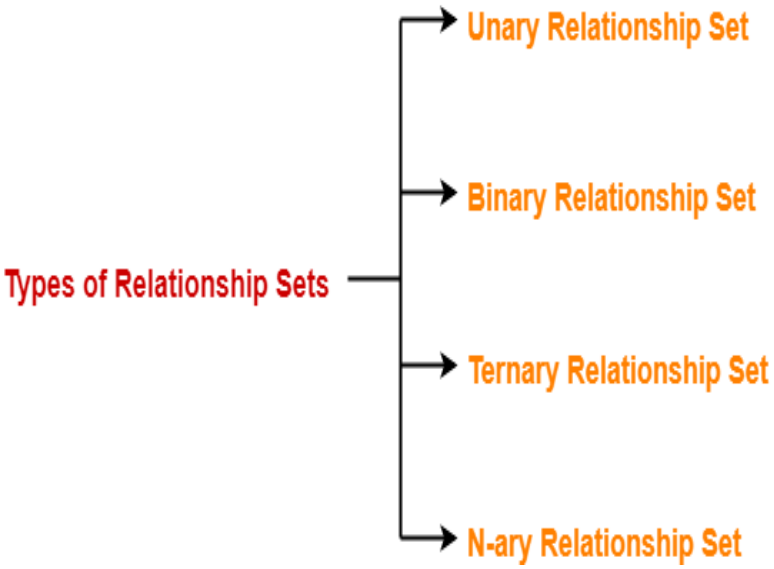
Relationship Set: A collection of relationships of same type is called the relationship set. OR

***Relationship Set*:** A relationship set is a set of relationships of same type



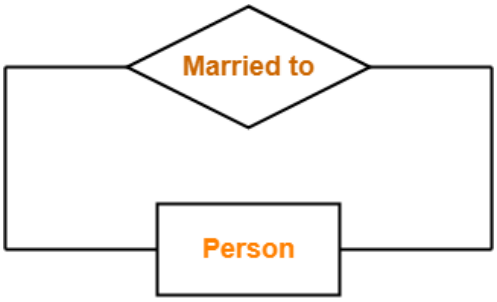
Set Representation of ER Diagram

Types of Relationship Sets-



Degree of a relationship: Degree of a relationship type is the number of entities participating in the relation.

Unary relationship set is a relationship set where only one entity set participates in a relationship set. A unary relationship is **when both participants in the relationship are the same entity**.



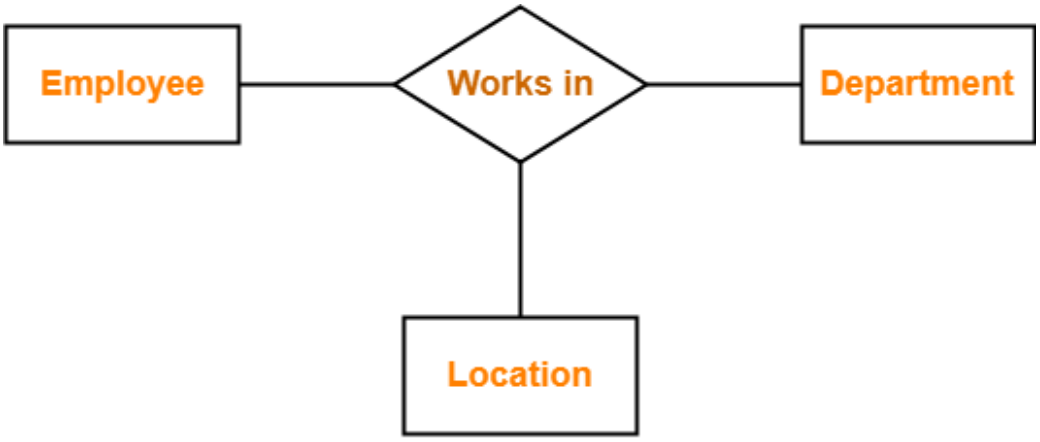
Unary Relationship Set

- A Binary Relationship is the **relationship between two different Entities** i.e. it is a relationship of role group of one entity with the role group of another entity. Binary relationship set is a relationship set where two entity sets participate in a relationship set.

Binary Relationship Set

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

  Ternary relationship exists when there are three types of entity and we call them a degree of relationship is 3. Since the number of entities increases due to this, it becomes very complex to turn E-R into a relational table.



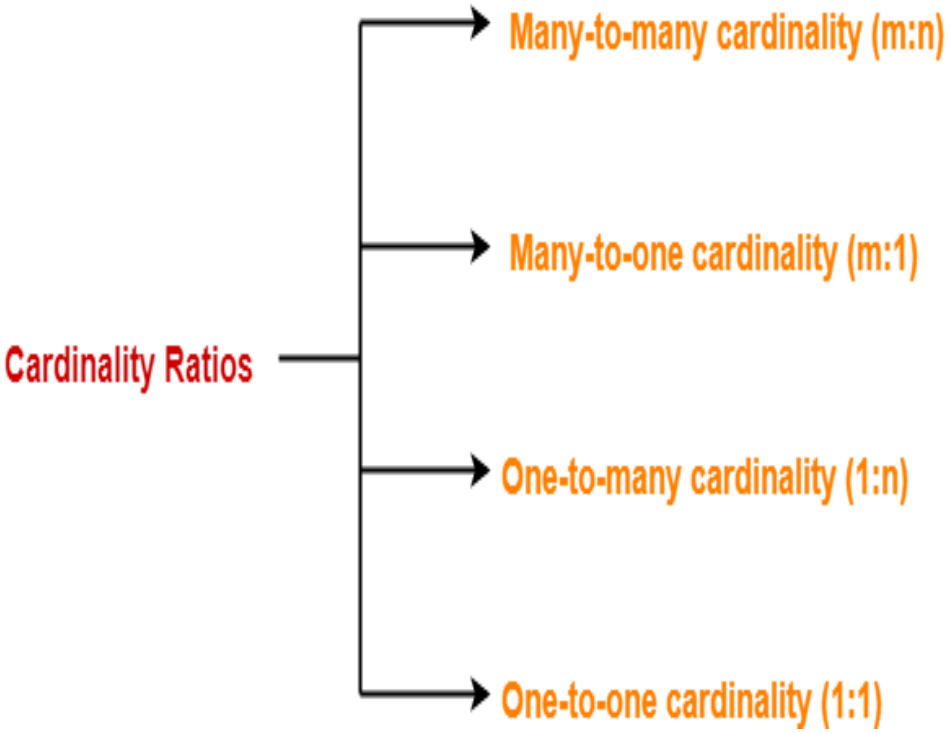
Ternary Relationship Set

4. N-ary Relationship Set-

Nary relationship set is a relationship set where 'n' entity sets participate in a relationship set.

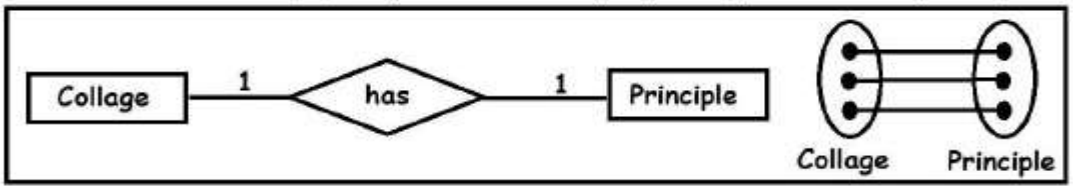
Cardinality: Cardinality constraint defines the maximum number of relationship instances in which an entity can participate. **Cardinality** is the number of instances of one entity that can or must be associated with each instance of another entity

TYPES OF CARDINALITY/ MAPPING

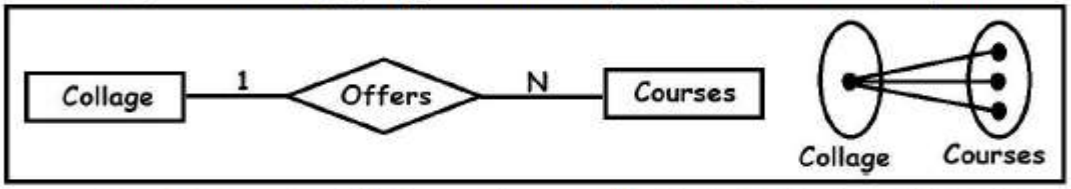


DATABASE MANAGEMENT SYSTEM UNIT-2 NOTES-2021

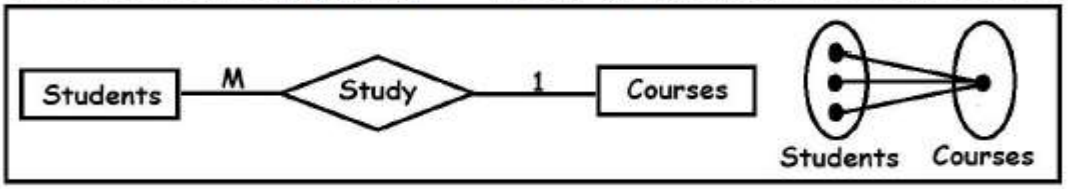
1. One-to-One Relationship (1:1): For ex: (1:1) college can have only one principle



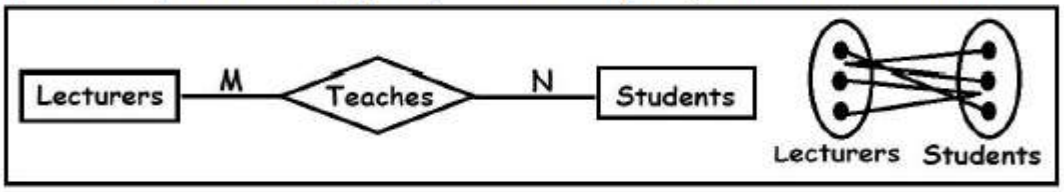
2. One-to-Many Relationship (1:N): For ex: (1:N) Collage offers many courses



3. Many-to-One Relationship (M:1): For ex: (M-1) STUDENTS STUDY COURSE



4. Many-to-Many Relationship (M:N): For ex: (M:N) students enrolled in a classes



DATABASE MANAGEMENT SYSTEM UNIT-2 NOTES-2021

Cardinality Relationship

C Computer
S Social
E Education

Cardinality – The number of entities to which another entity can be associated through a relationship

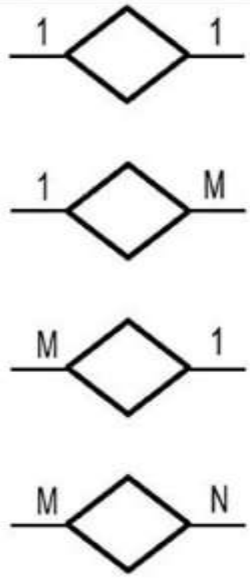
The diagrams on the right show, in order:

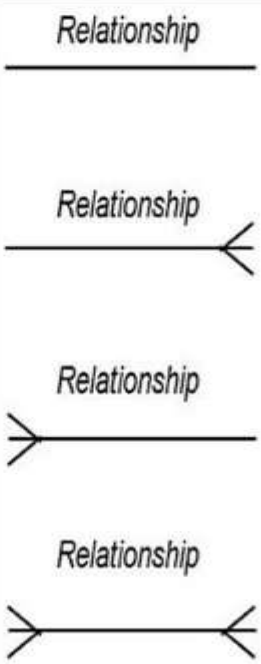
one-to-one

one-to-many

many-to-one

many-to-many



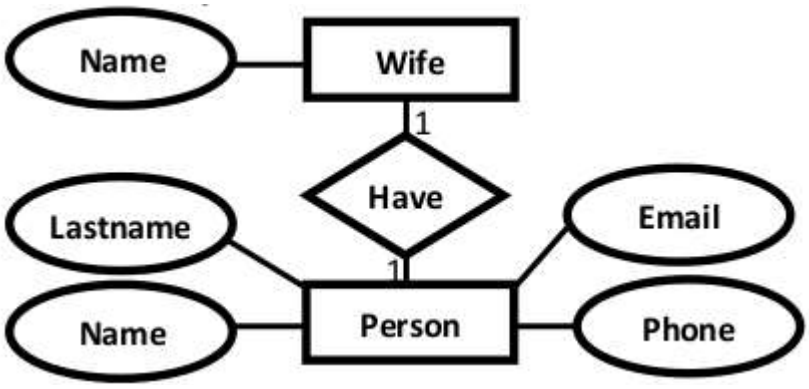


DATABASE MANAGEMENT SYSTEM UNIT-2 NOTES-2021

Cardinality types/ Ratios:

1:1Relationships :

One-to-one: one entity from entity set A can be associated with at most one entity of entity set B and vice versa.

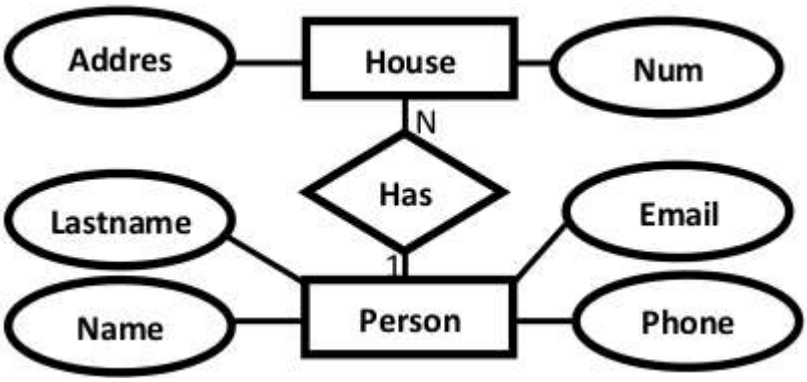


Persons(personid , name, lastname, email , ***wifeid***)
Wife (wifeid , name)

1:N Relationships

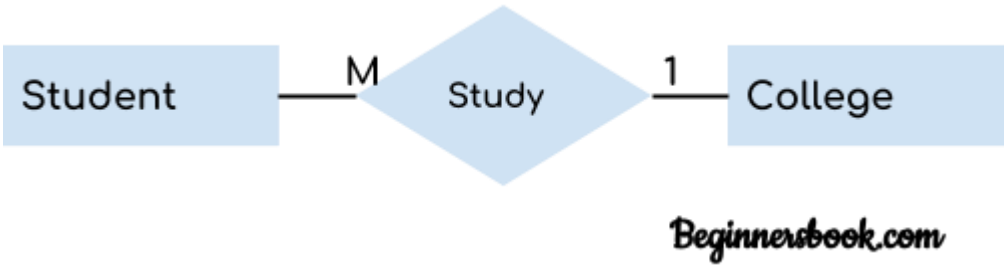
One-to-many: One entity from entity set A can be associated with more than one entities of entity set B but from entity set B one entity can be associated with at most one entity.

For instance, the Person can have a **House** from zero to many , but a **House** can have only one **Person**. To represent such relationship the **personid**as the Parent node must be placed within the Child table as a foreign key but **not the other way around as shown next:**

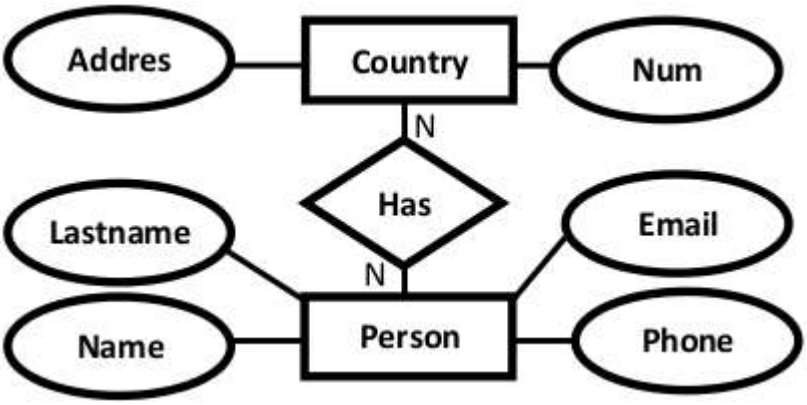


It should convert to :
Persons(personid , name, lastname, email)
House (houseid , num , address, ***personid***)

Many-to-one(M:1): More than one entities from entity set A can be associated with at most one entity of entity set B but one entity from entity set B can be associated with more than one entity from entity set A.



N: N Relationships (M:M)
Many-to-many: one entity from A can be associated with more than one entity from B and vice versa.



Persons(personid , name, lastname, email)

Countries (countryid , name, code)

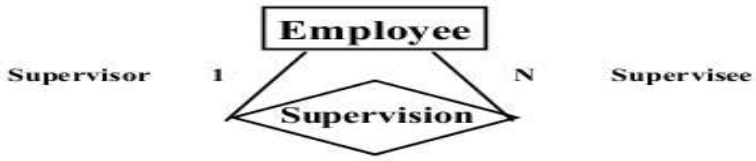
HasRelat (hasrelatid , **personid , countryid**)

ROLE NAMES AND RECURSIVE RELATIONSHIP

Role Names: specifies the role of each entity type participating in the relationship. For example in EMPLOYEE WORKS FOR DEPARTMENT, Employee plays the role of a worker and Department plays the role of Employer.

Entity Roles

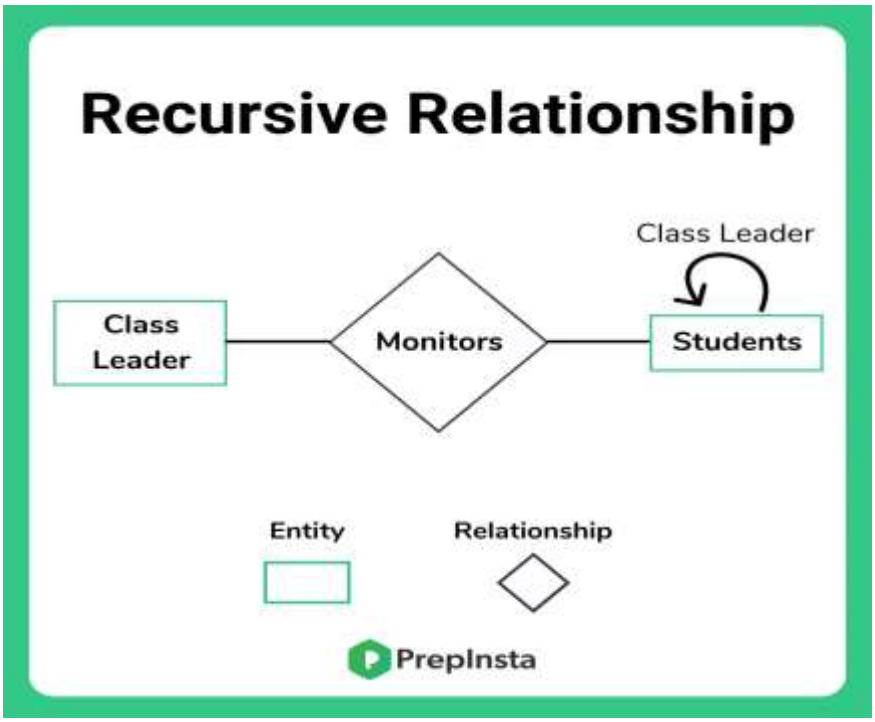
- Each entity type in a relationship type plays a particular role that is described by a role name. Role names are especially important in recursive relationship types where the same entity participates in more than one role:



14

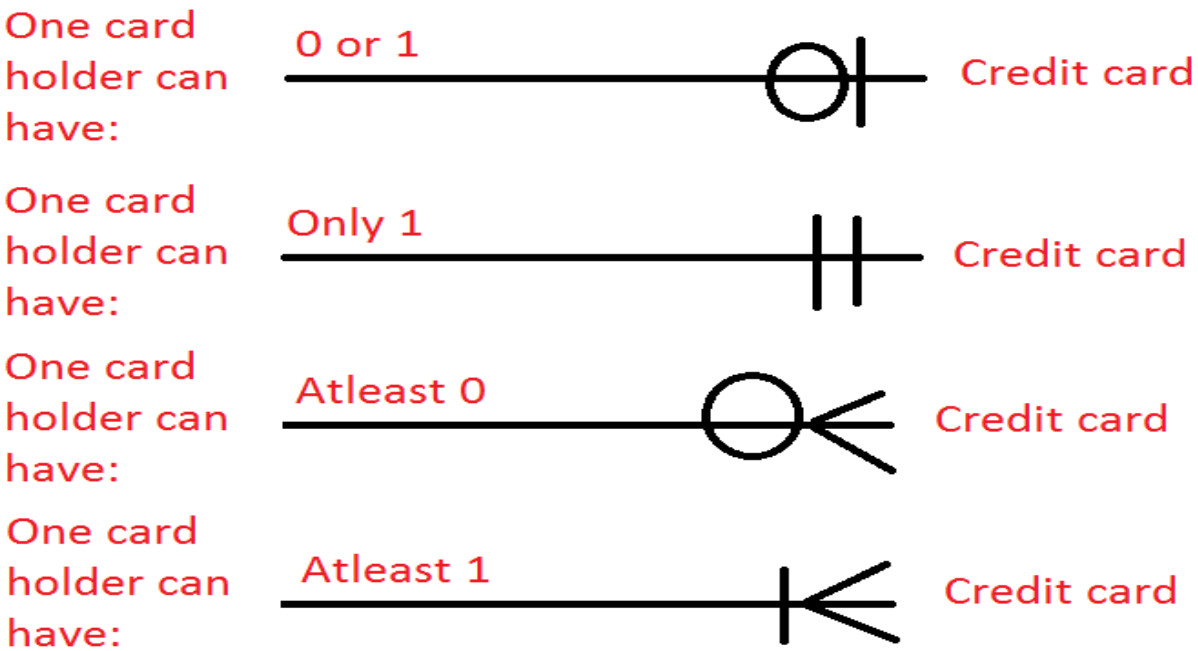
Recursive Relationship: is a relationship in which same entity type participates more than once in a relationship type in different roles.

Ex: SUPERVISION relationship in which entity EMPLOYEE plays the role of SUPERVISOR & SUPERVISION.



Optionality Express whether the relationship is optional or mandatory.

Cardinality expresses the maximum number of Relationships.



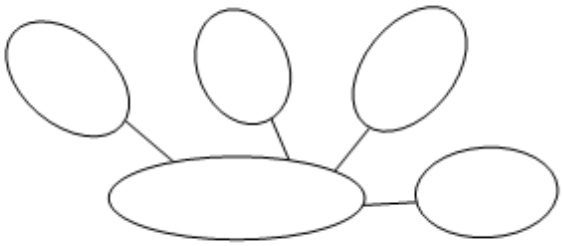
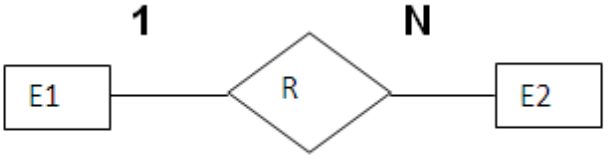
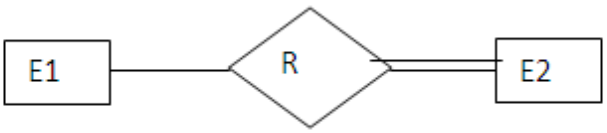
Proper naming of Schema constructs

- When designing a data base schema, the choice of names for entity types, attributes, relationship types & role is not always straight forward
- One should choose names that convey as much as possible, the meaning attached to the different constructs in the schema
- It is preferable to use Singular names for entity types, rather than plural ones, because the entity type name applies to each individual entity belonging to that entity type
- Each object in a schema is known as schema construct

DATABASE MANAGEMENT SYSTEM UNIT-2 NOTES-2021

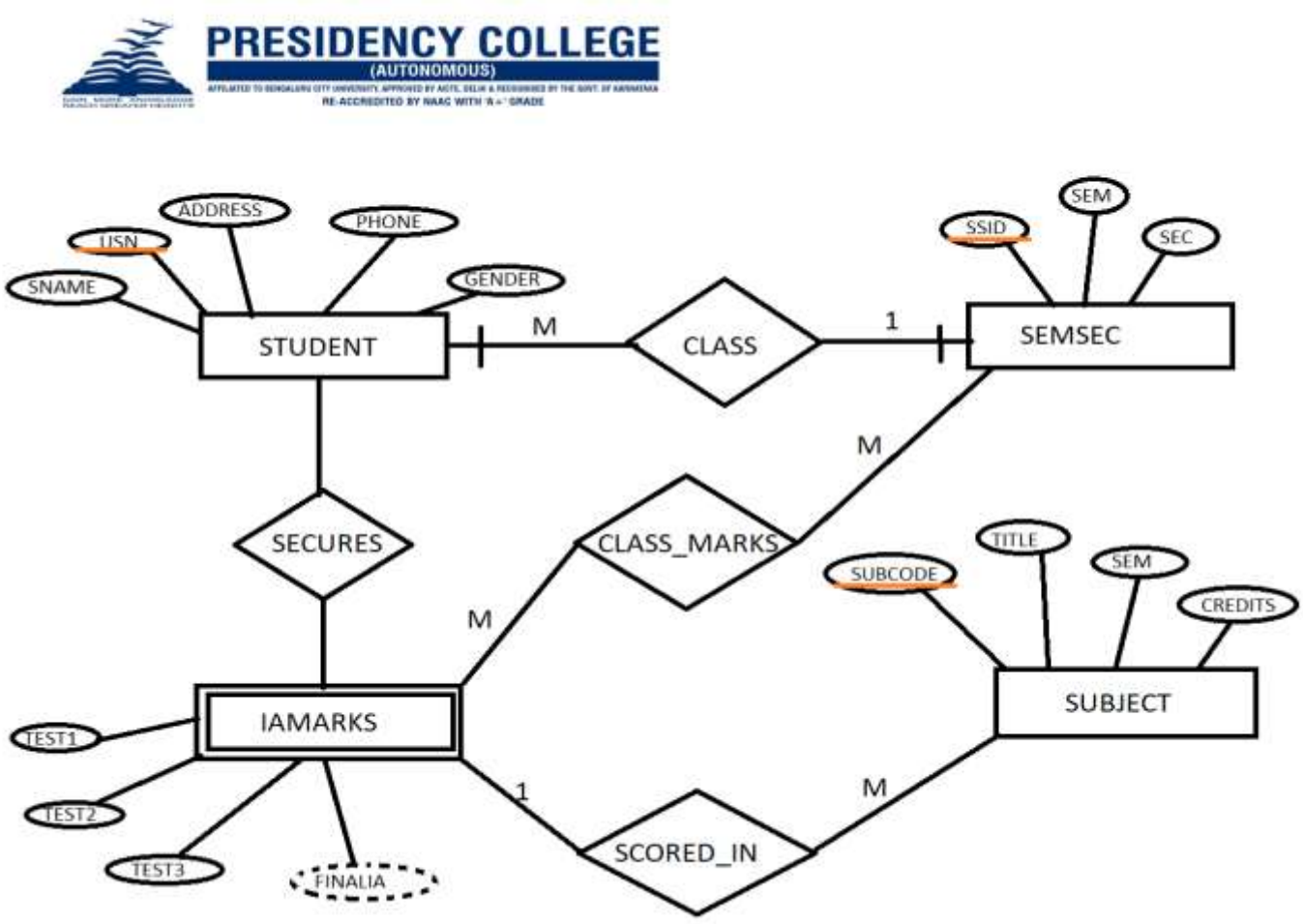
ER DIAGRAM.

An Entity Relationship [ER] diagram is graphical depiction of organizational system elements & the association among the elements where elements are entities & the association is relationship
Thus, the diagrammatic notation associated with ER model is known as ER diagram

	Multi valued attribute
	Composite attributes
	Derived attribute
	Cardinality ratio 1:N E1, E2 in R
	Total participation E2 in R

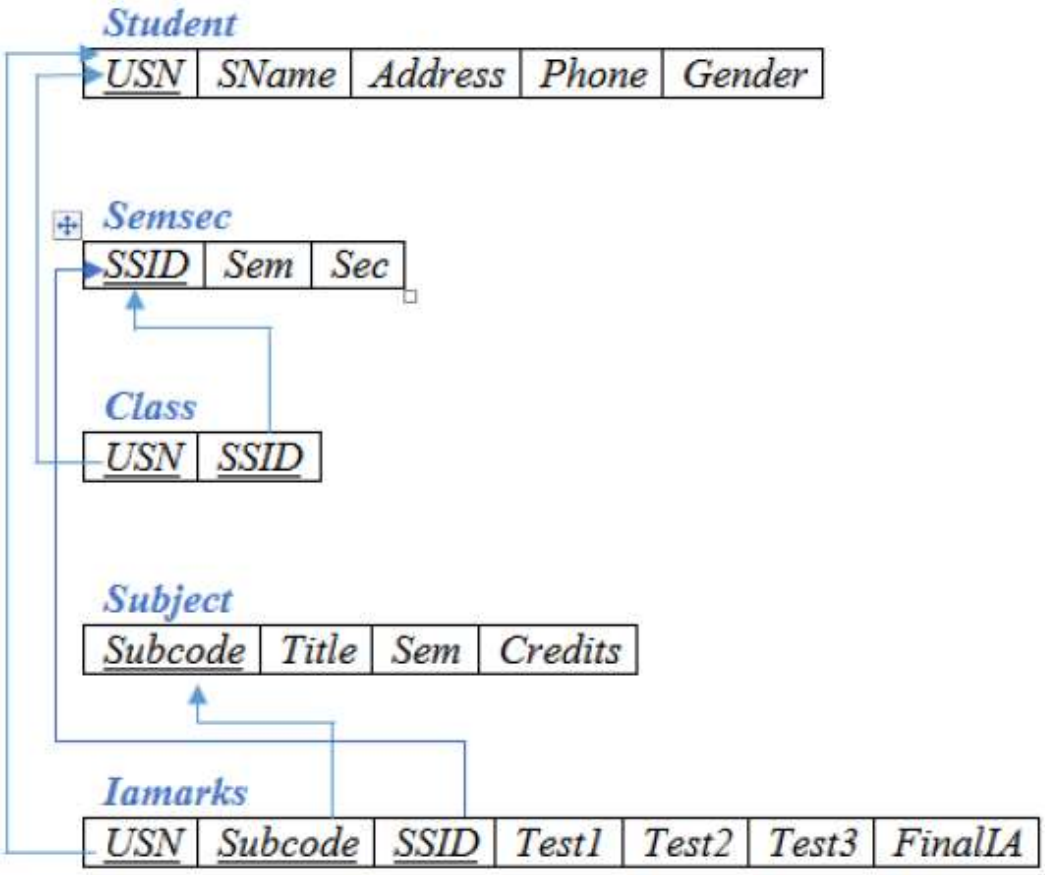
DATABASE MANAGEMENT SYSTEM UNIT-2 NOTES-2021

1. **Draw E-R diagram and convert entities and relationships to relation table for a given scenario. (eg. College)**



DATABASE MANAGEMENT SYSTEM UNIT-2 NOTES-2021

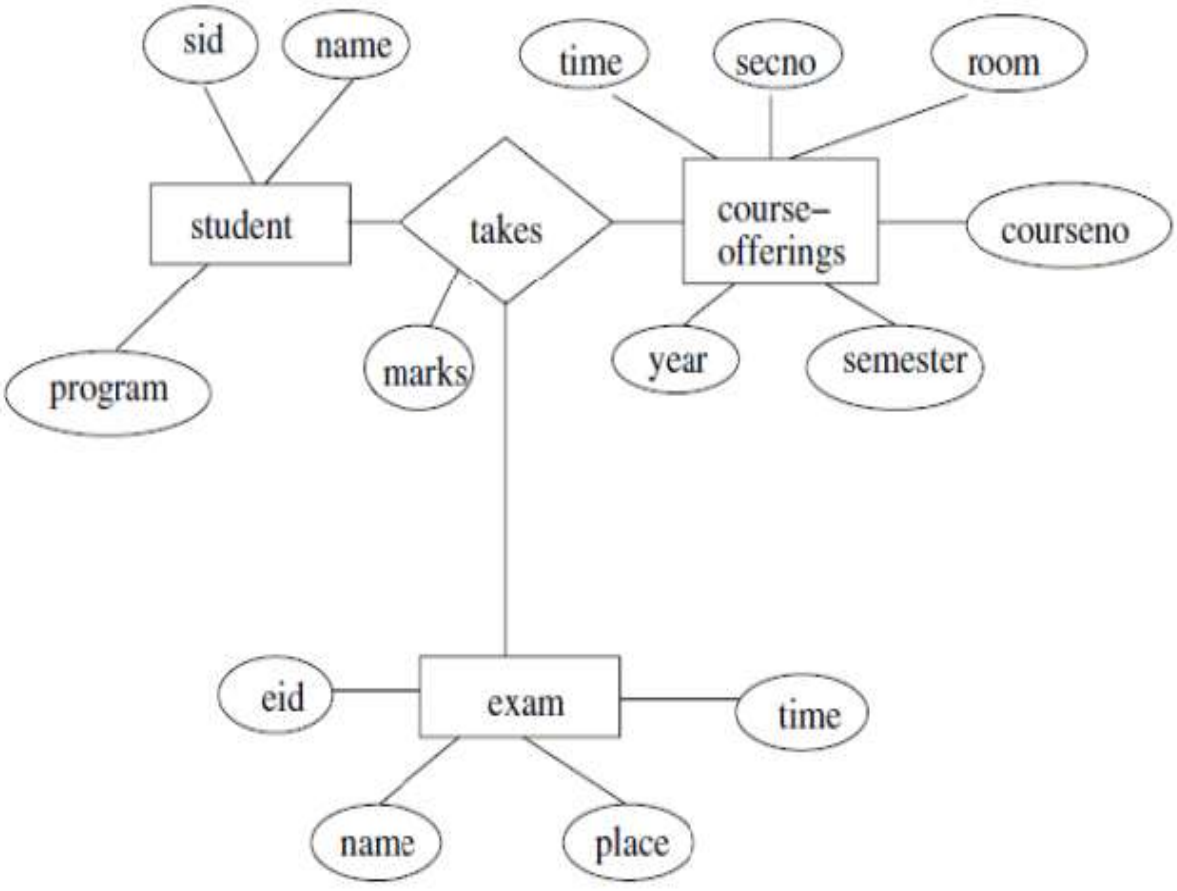
MAPPING ENTITY RELATIONSHIP MODEL TO RELATIONAL TABLE



Consider a database used to record the marks that students get in different exams of different course offerings.

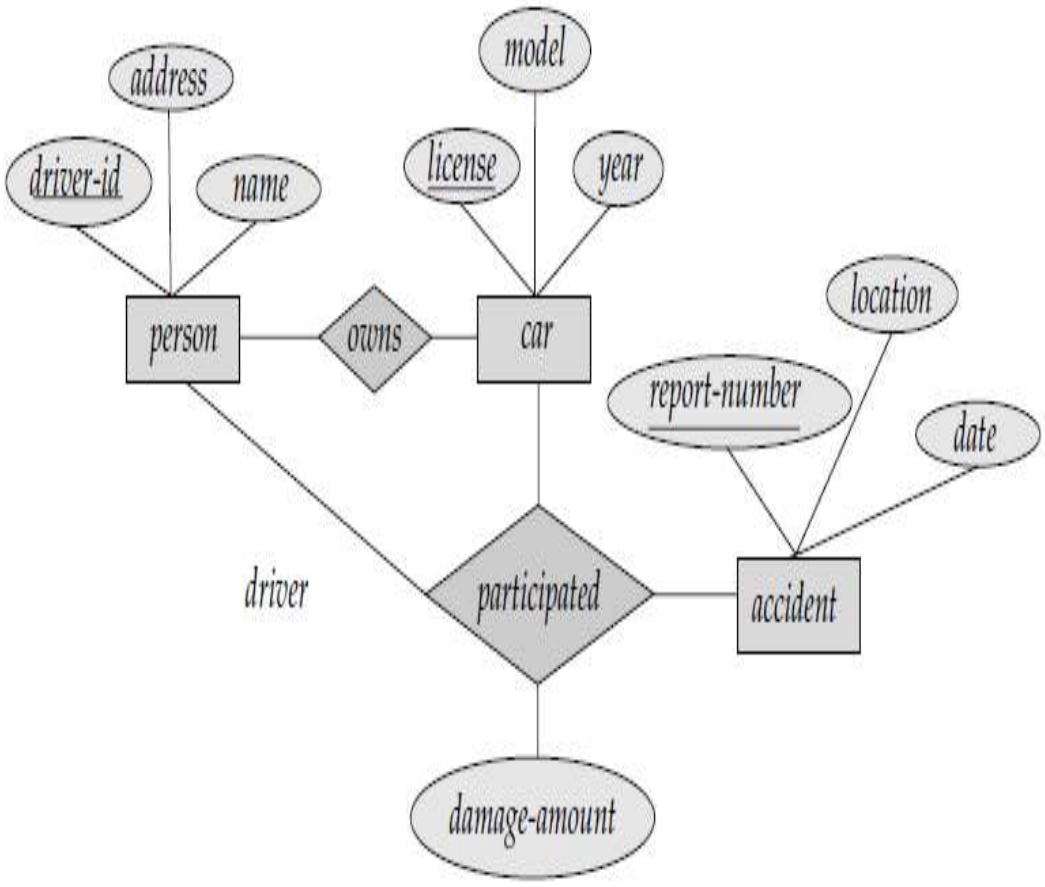
a) Construct an E-R diagram that models exams as entities, and uses a ternary relationship, for the above database.

Answer a :



DATABASE MANAGEMENT SYSTEM UNIT-2 NOTES-2021

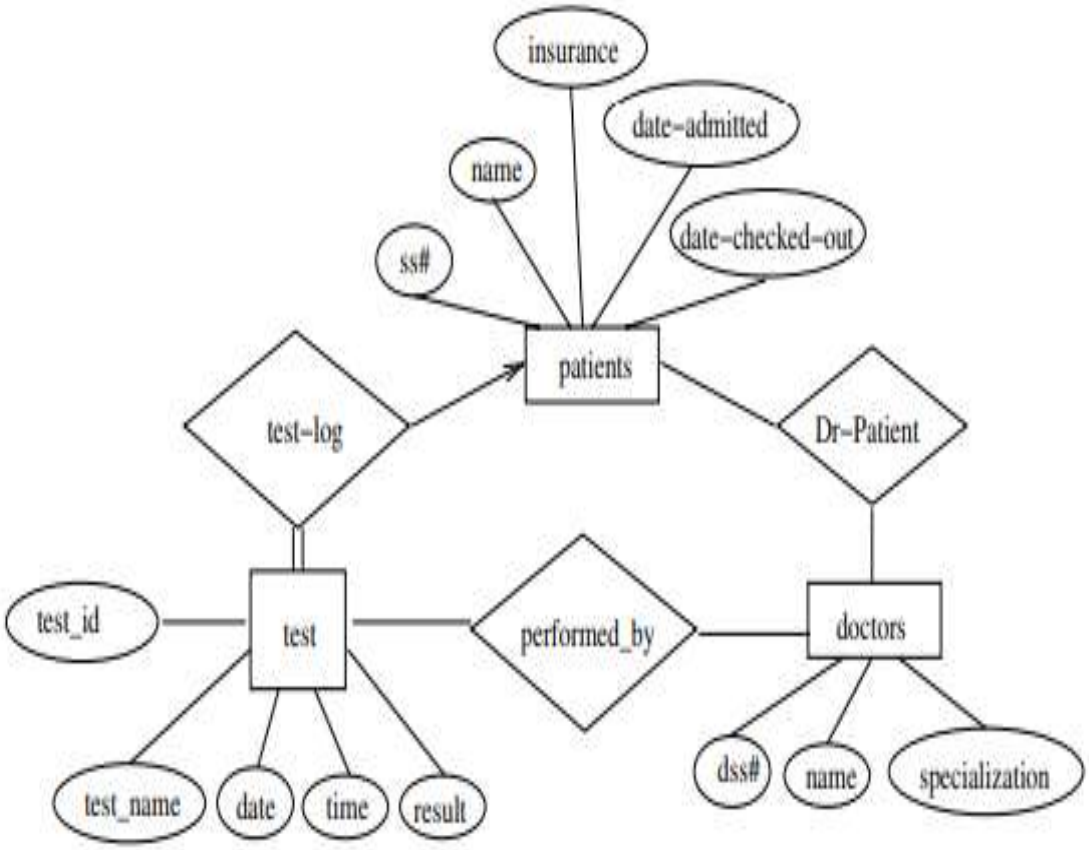
- Construct an E-R diagram for a car-insurance company
- whose customers own one or more cars each.
- Each car has associated with it zero to any number of recorded **accidents.**



DATABASE MANAGEMENT SYSTEM UNIT-2 NOTES-2021

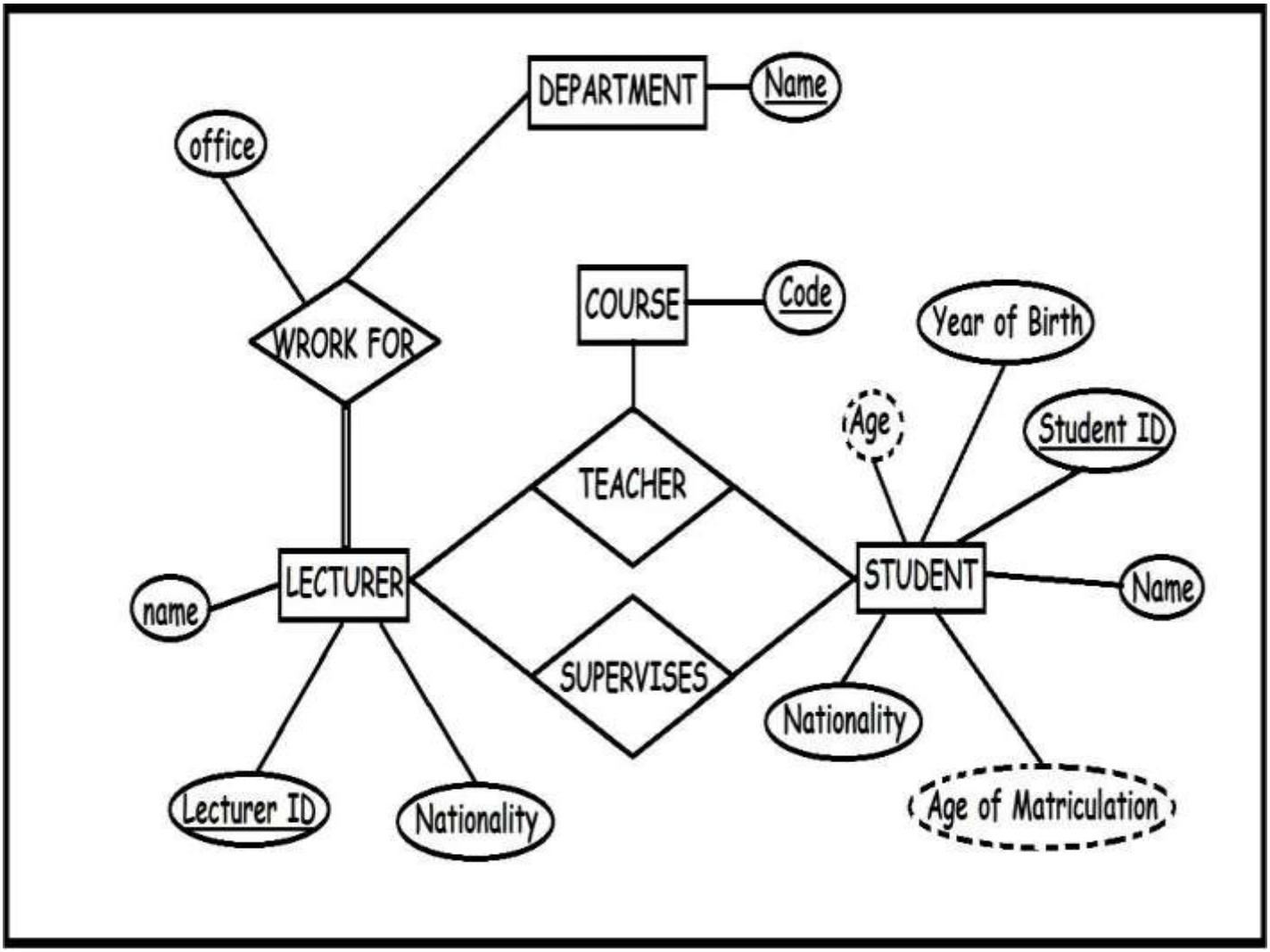
Construct an E-R diagram for a hospital

- with a set of patients and a set of medical doctors.
- Associate with each patient a log of the various tests and examinations conducted.



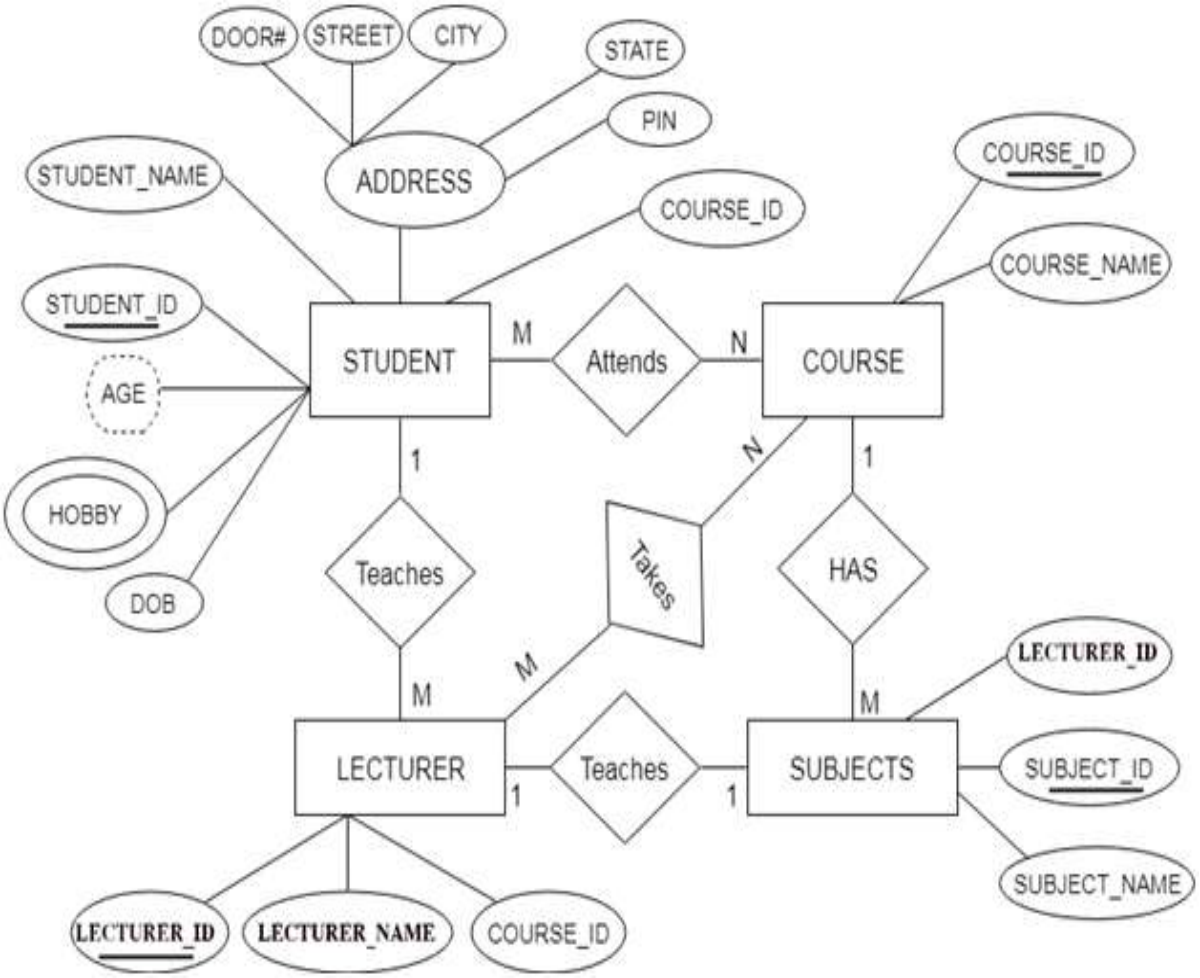
DATABASE MANAGEMENT SYSTEM UNIT-2 NOTES-2021

ER DIAGRAM FOR UNIVERSITY



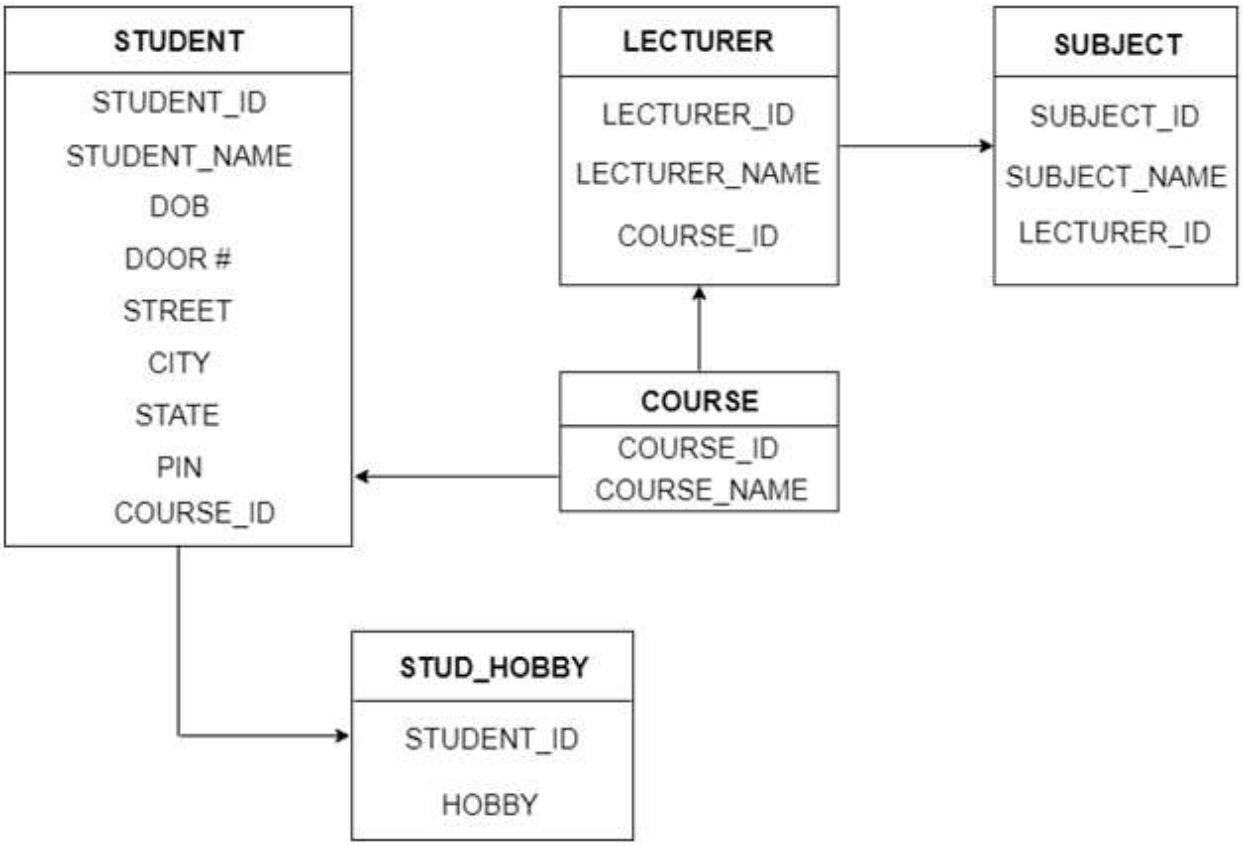
DATABASE MANAGEMENT SYSTEM UNIT-2 NOTES-2021

- Mapping Process (Algorithm)
  - Create table for each entity.
  - Entity's attributes should become fields of tables with their respective data types.
  - Declare primary key.



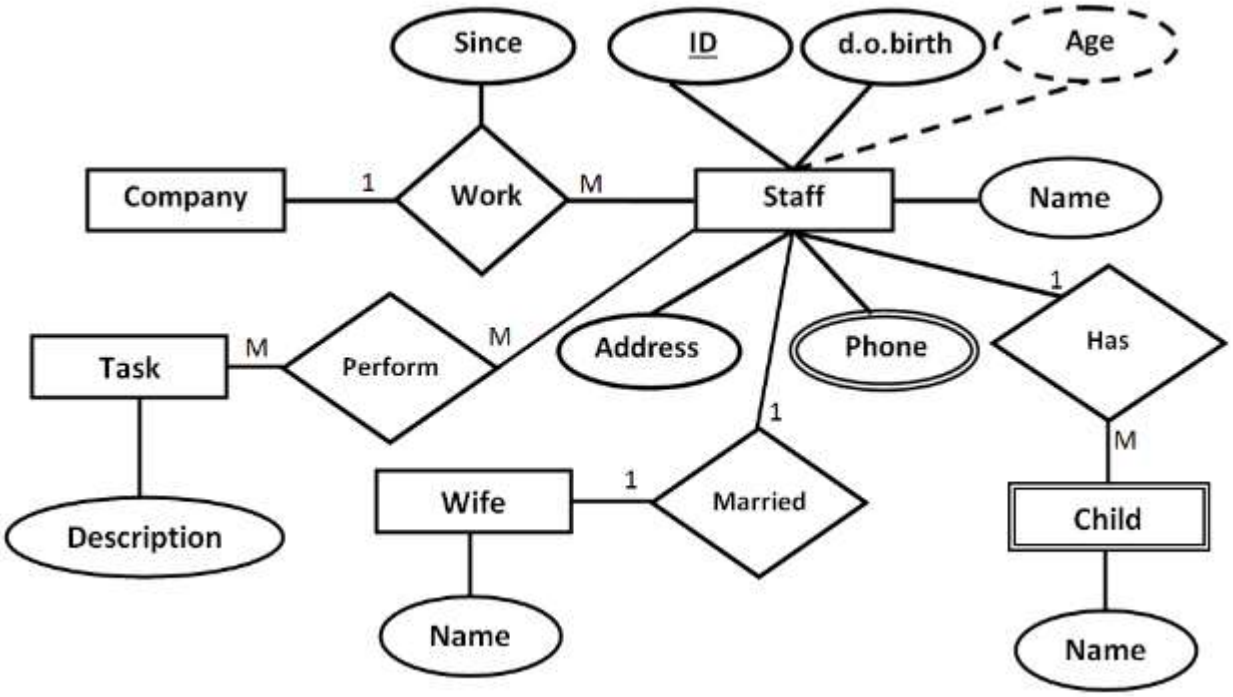
DATABASE MANAGEMENT SYSTEM UNIT-2 NOTES-2021

ER MAPPING TO RELATIONAL DATABASE



DATABASE MANAGEMENT SYSTEM UNIT-2 NOTES-2021

COMPANY ER



The relational schema for the ER Diagram is given below as:
Company(CompanyID , name , address)

Staff(StaffID , dob , address , ***WifeID***)

Child(ChildID , name , ***StaffID***)

Wife (WifeID , name)

Phone(PhoneID , phoneNumber , ***StaffID***)

Task (TaskID , description)

Work(WorkID , ***CompanyID , StaffID*** , since)

Perform(PerformID , ***StaffID , TaskID***)

File organisation

- A database consist of a huge amount of data. The data is grouped within a table in RDBMS, and each table have related records.
- A user can see that the data is stored in form of tables, but in actual this huge amount of data is stored in physical memory in form of files.
- **File –** A file is named collection of related information that is recorded on secondary storage such as magnetic disks, magnetic tapes and optical disks.

Types of File Organizations –

- Various methods have been introduced to Organize files. These particular methods have advantages and disadvantages on the basis of access or selection .
- Sequential File Organization(SORTED)
  - Pile File Method
  - Sorted File Method
- Heap File Organization (UNSORTED)
- Hash File Organization
  - External
  - Internal

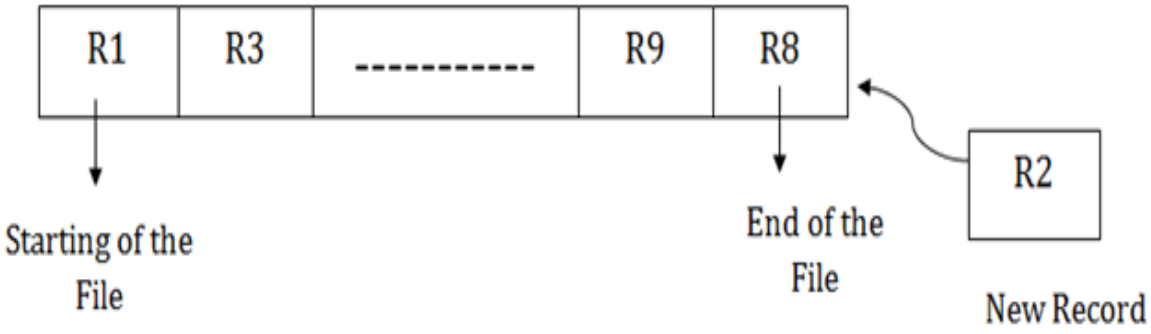
- Sequential File Organization: (SORTED) This method is the easiest method for file organization. In this method, files are stored sequentially.
- This method can be implemented in two ways:

Pile File Method:

- It is a quite simple method. In this method, we store the record in a sequence, i.e**., one after another. Here, the record will be inserted in the order in which they are inserted into tables**.
- In case of updating or deleting of any record, the record will be searched in the memory blocks. When it is found, then it will be marked for deleting, and the new record is inserted.

Insertion of the new record:

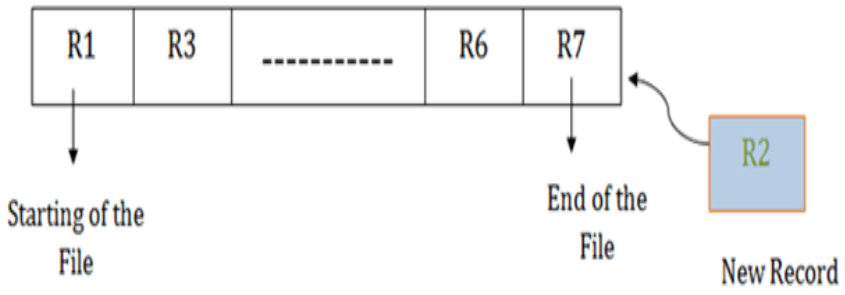
Suppose we have four records R1, R3 and so on upto R9 and R8 in a sequence. Hence, records are nothing but a row in the table. Suppose we want to insert a new record R2 in the sequence, then it will be placed at the end of the file. Here, records are nothing but a row in any table.

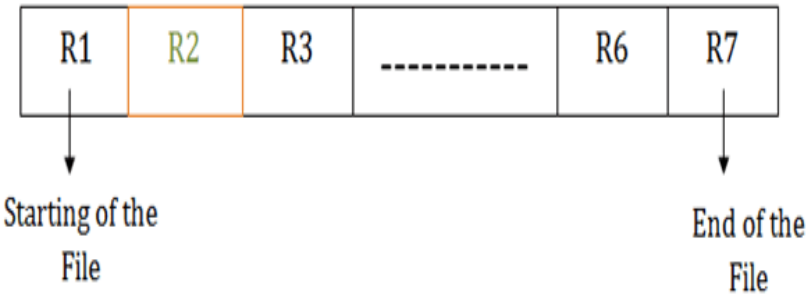


- **Sorted File Method:**

- In this method, the new record is always inserted at the file's end, **and then it will sort the sequence in ascending or descending order**. Sorting of records is based on any primary key or any other key.
- In the case of modification of any record, it will update the record and then sort the file, and lastly, the updated record is placed in the right place.

Suppose there is a preexisting sorted sequence of four records R1, R3 and so on upto R6 and R7. Suppose a new record R2 has to be inserted in the sequence, then it will be inserted at the end of the file, and then it will sort the sequence.

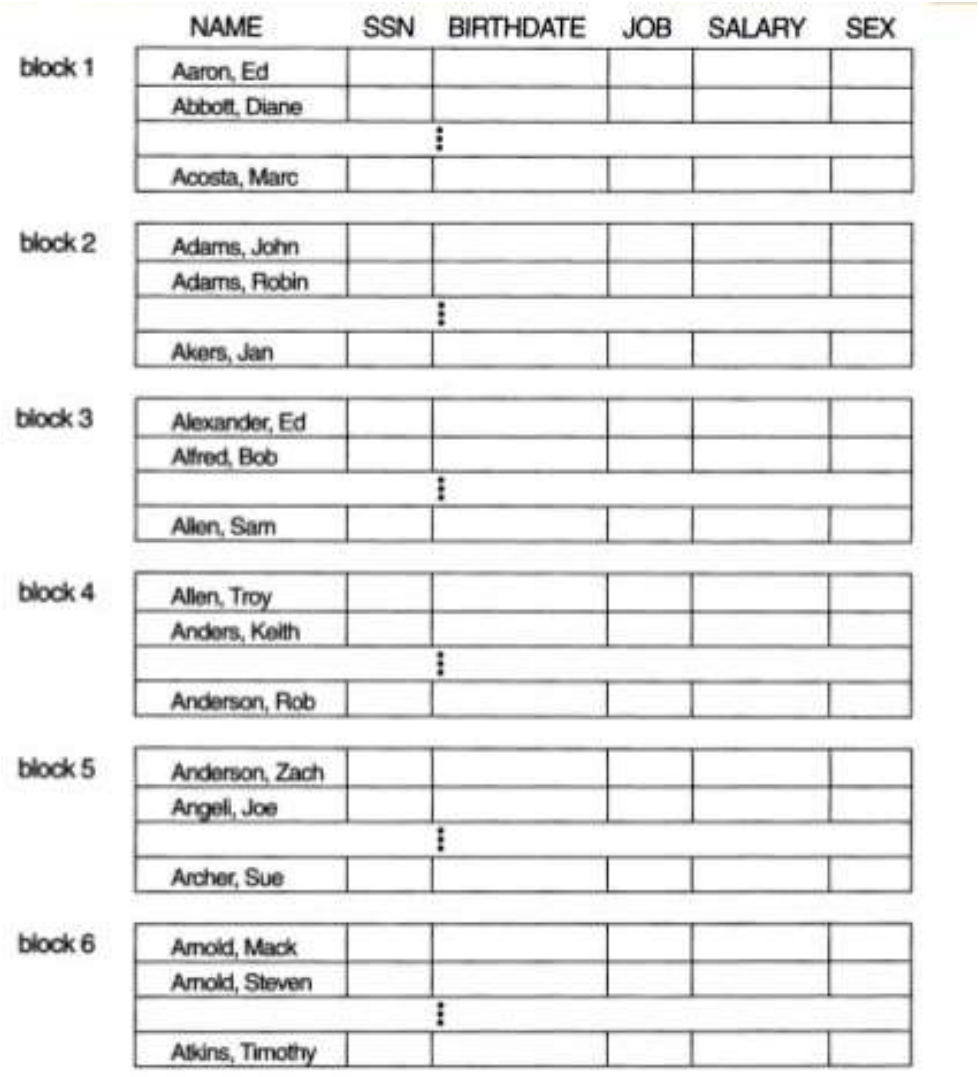




- **Pros of sequential file organization**
  - It contains a fast and efficient method for the huge amount of data.
  - In this method, files can be easily stored in cheaper storage mechanism like magnetic tapes.
  - It is simple in design. It requires no much effort to store the data.
  - This method is used when most of the records have to be accessed like grade calculation of a student, generating the salary slip, etc.
  - This method is used for report generation or statistical calculations.
  - **Cons of sequential file organization**
  - It will waste time as we cannot jump on a particular record that is required but we have to move sequentially which takes our time.
  - Sorted file method takes more time and space for sorting the records.

DATABASE MANAGEMENT SYSTEM UNIT-2 NOTES-2021

ORDERED FILES



Hashing technique:

- Hashing is a method, which provides very fast access to records on certain search conditions.
- The search condition must be an equality condition on a single field called the hash field of the file.
- The hash field is also a key field of the file, in which case it is called hash key.
- **The basic terms associated with the hashing technique are:**
- **1. Hash table:** It is simply an array that is having address of records.
- **2. Hash function:** It is the transformation of a key into the corresponding location or address in the hash table. (it can be defined as function that takes key as input and transforms it into a hash table index)
- **3. Hash key:** Let R be a record and its key hashes into a key called hash key.

Assume a table with 8 slots:

Hash key = key % table size

4 = 36 % 8

2 = 18 % 8

0 = 72 % 8

3 = 43 % 8

6 = 6 % 8

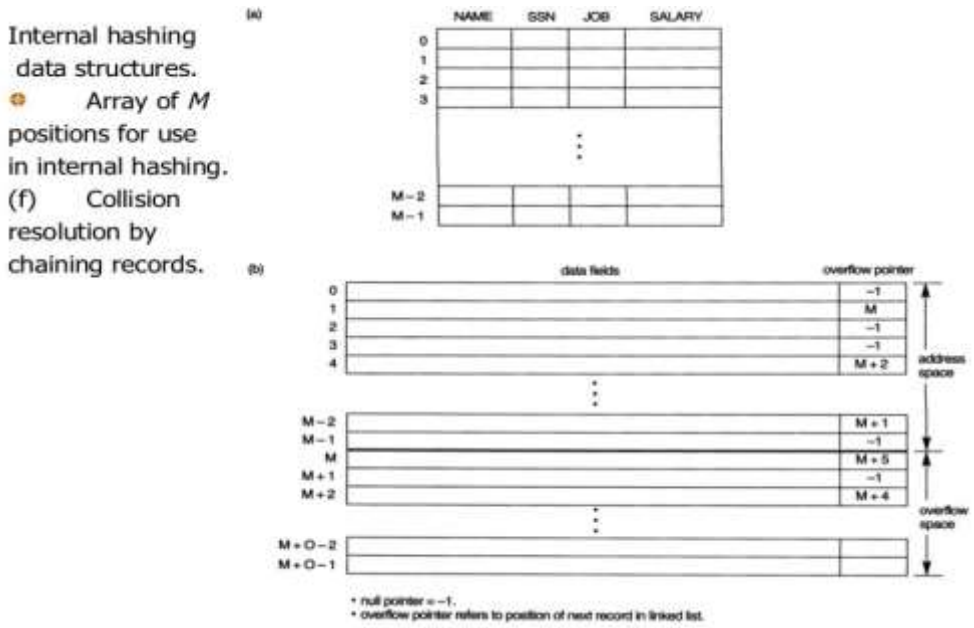
[0]	72
[1]	
[2]	18
[3]	43
[4]	36
[5]	
[6]	6
[7]	

Types of Hashing

1. Internal Hashing: The hashing technique employed for internal files is known as Internal Hashing. Internal files are the files stored in primary memory
2. External Hashing: The hashing technique employed for disk files is known as External hashing

Internal hashing: For internal files, hash table is an array of records, having array index ranges from 0 to M-1, let us consider a hash function H(K) such that **H(K) = key MOD M** which produce a remainder between 0 and M-1 depending on the value of key, this value then used for the record By

Hashing is typically implemented as a **hash table** through the use of an array of records. Let the array index range is from 0 to M–1, as shown in Figure(a); then we have *M* **slots** whose addresses correspond to the array indexes. We choose a hash function that transforms the hash field value into an integer between 0 and M-1. One common hash function is the $h(K) = K$ **mod** M function, which returns the remainder of an integer hash field value K after division by M; this value is then used for the record address.

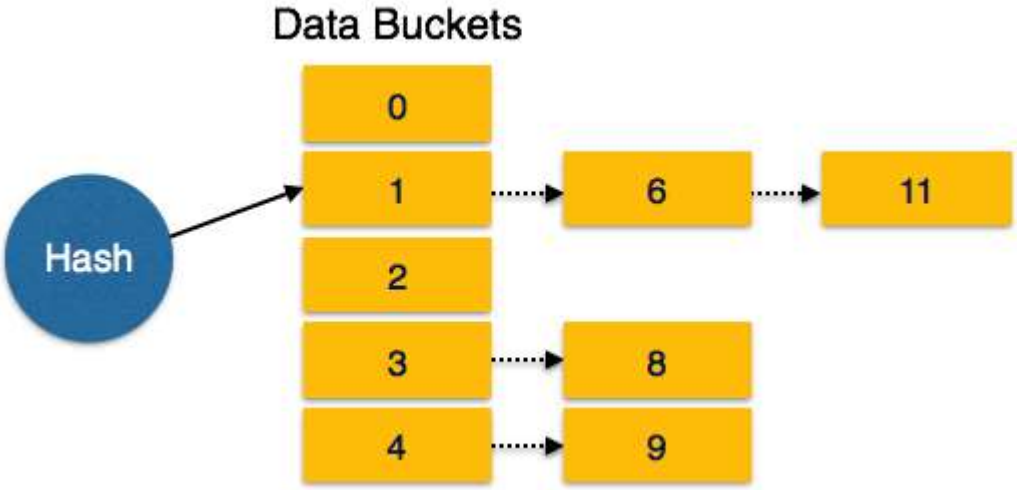


A **collision** occurs when the hash field value of a record that is being inserted hashes to an address that already contains a different record. In this situation, we must insert the new record in some other position. The process of finding another position is called **collision resolution**.

There are numerous methods for collision resolution, including the following:

Open addressing. Ponce a position specified by the hash address is found to be occupied, the program checks the subsequent positions in order until an unused (empty) position is found. Algorithm (b) may be used for this purpose.

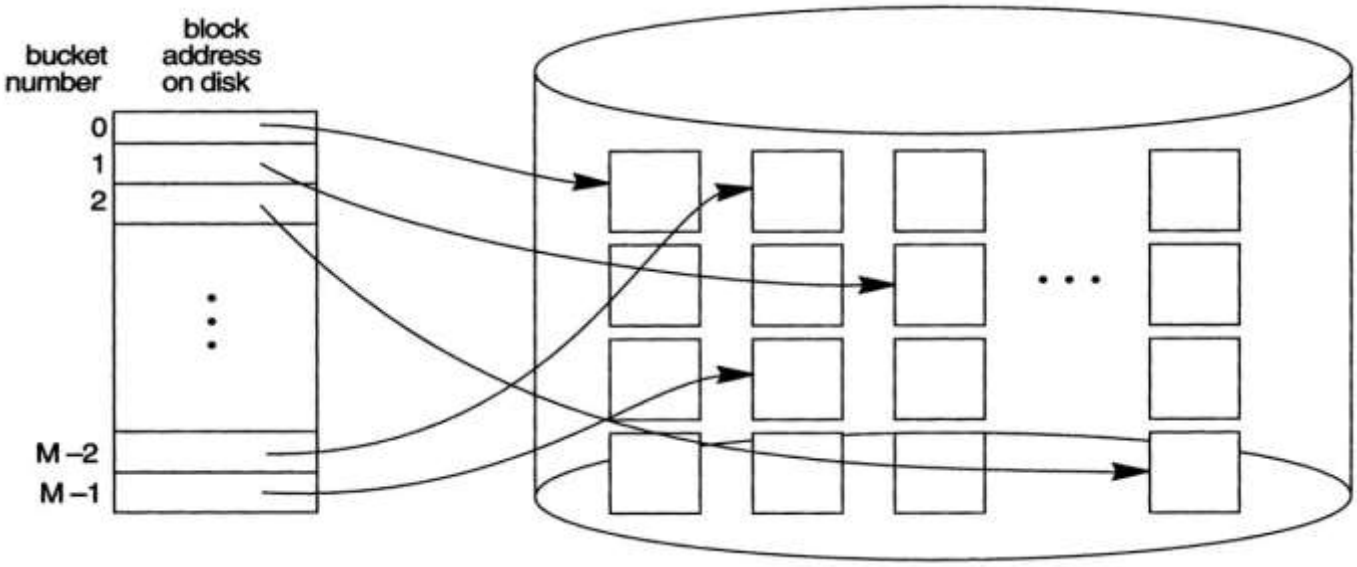
Chaining. For this method, various overflow locations are kept, usually by extending the array with a number of overflow positions. Additionally, a pointer field is added to each record location. A collision is resolved by placing the new record in an unused overflow location and setting the pointer of the occupied hash address location to the address of that overflow location. A linked list of overflow records for each hash address is thus maintained, as shown in Figure (b).



Multiple hashing. The program applies a second hash function if the first results in a collision. If another collision results, the program uses open addressing or applies a third hash function and then uses open addressing if necessary.

DATABASE MANAGEMENT SYSTEM UNIT-2 NOTES-2021

External Hashing for Disk Files Hashing for disk files is called **external hashing**. the target address space is in external hashing is made of **buckets**, A bucket is either one disk block or a cluster of contiguous blocks. The hashing function maps the indexing field's value into a relative bucket number. A table maintained in the file header converts the bucket number into the corresponding disk block address



HEAP (UNSORTED)

Files of unordered records [Heap files]:

- It is the simplest and most basic type of organization.
- It works with data blocks. In heap file organization, the records are inserted at the file's end. When the records are inserted, it doesn't require the sorting and ordering of records.
- When the data block is full, the new record is stored in some other block. This new data block need not to be the very next data block, but it can select any data block in the memory to store new records. The heap file is also known as an unordered file.

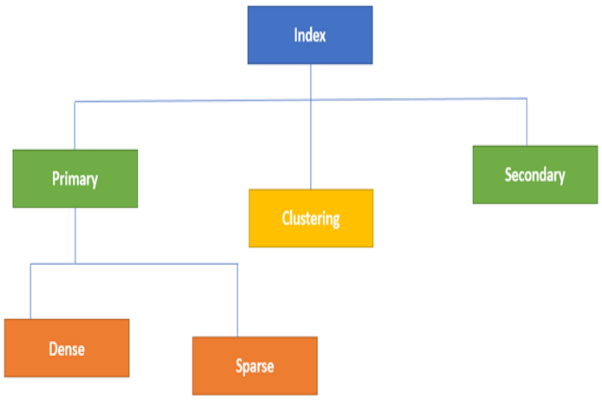
- In the file, every record has a unique id, and every page in a file is of the same size. It is the DBMS responsibility to store and manage the new records.

RAID

- RAID or **R**edundant **A**rray of **I**ndependent **D**isks, is a technology to connect multiple secondary storage devices and use them as a single storage media.
- RAID consists of an array of disks in which multiple disks are connected together to achieve different goals. RAID levels define the use of disk arrays.
-

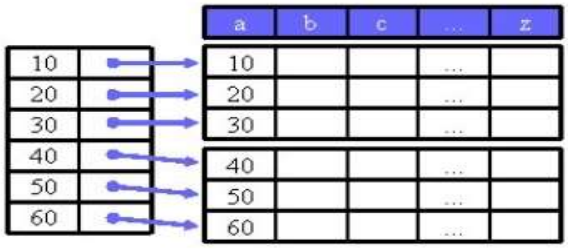
DATABASE MANAGEMENT SYSTEM UNIT-2 NOTES-2021

- **Indexing** is a data structure technique which allows you to quickly retrieve records from a database file.
- An Index is a small table having only two columns.
- The first column comprises a copy of the primary or candidate key of a table.
- Its second column contains a set of pointers for holding the address of the disk block where that specific key value stored.
- Types of Single-level Ordered Indexes
- Primary Indexes
- Clustering Indexes
- Secondary Indexes
- Multilevel Indexes
- Indexes on Multiple Keys



- **Primary Index** − Primary index is defined on an ordered data file. The data file is ordered on a **key field**. The key field is generally the primary key of the relation.
- **Secondary Index** − Secondary index may be generated from a field which is a candidate key and has a unique value in every record, or a non-key with duplicate values.
- **Clustering Index** − Clustering index is defined on an ordered data file. The data file is ordered on a non-key field.
- **Primary Index**
- Primary Index is an ordered file which is fixed length size with two fields.
    - The first field is the same a primary key and second,
    - filed is pointed to that specific data block.
    - In the primary Index, there is always one to one relationship between the entries in the index table.
- The primary Indexing in DBMS is also further divided into two types.
    - Dense Index
    - Sparse Index
- In a dense index, a record is created for every search key valued in the database. This helps you to search faster but needs more space to store index records.
- In this Indexing, method records contain search key value and points to the real record on the disk.

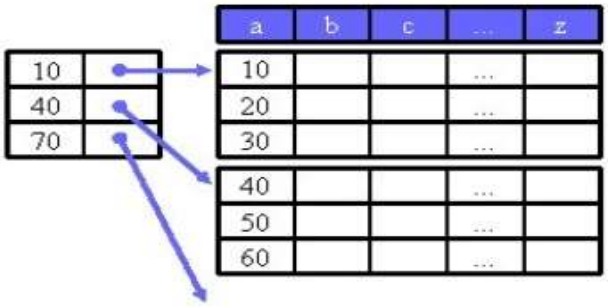
✓ A dense index has one index entry for every search key value



- every data record is represented in the index
- an existence search of a record may be done via index only
- the record is directly found in a block using the pointer, i.e., no search within the block

- It is an index record that appears for only some of the values in the file. Sparse Index helps you to resolve the issues of dense Indexing in DBMS.
- In this method of indexing technique, a range of index columns stores the same data block address, and when data needs to be retrieved, the block address will be fetched.

A sparse index has one index entry for every data block, i.e., the key of the first record

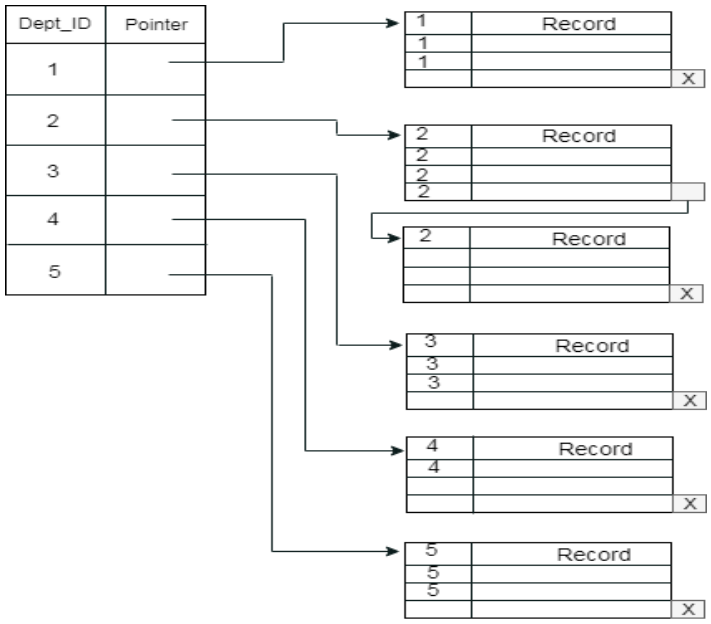


- only one index field per block, i.e., less index data
- cannot find out if a record exists only using the index

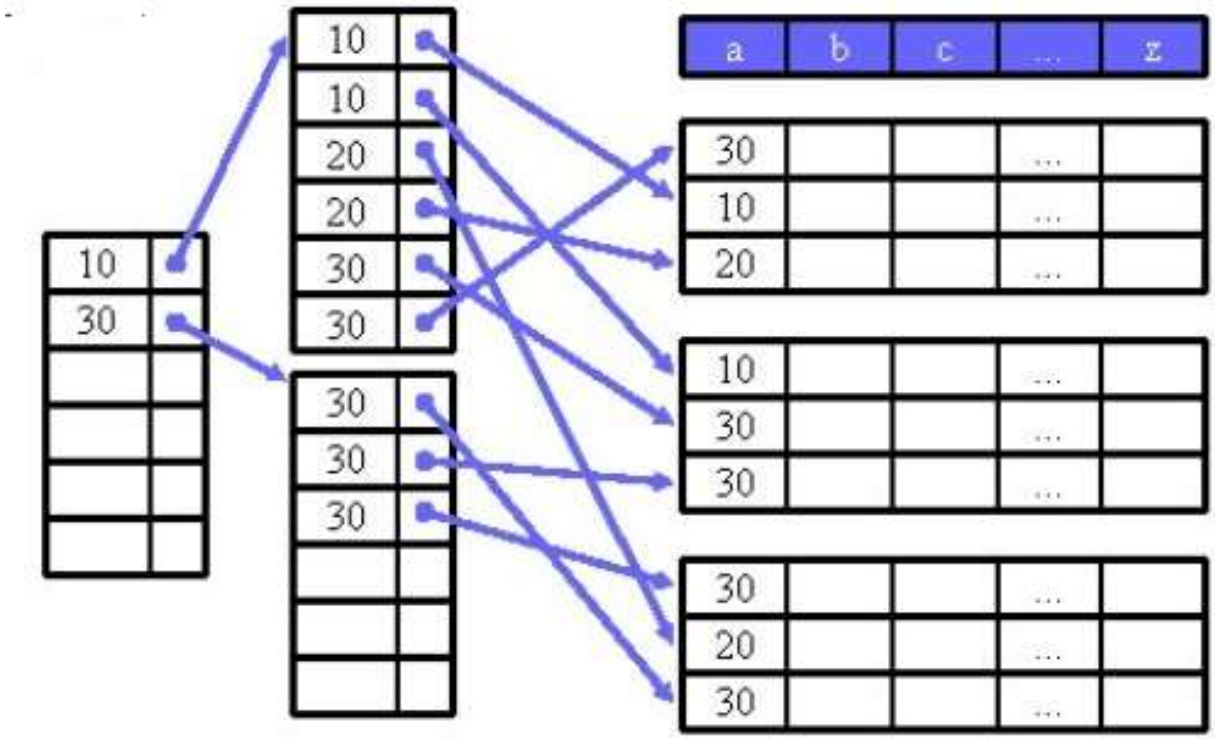
- **In a clustered index,** records themselves are stored in the Index and not pointers.
- Sometimes the Index is created on non-primary key columns which might not be unique for each record. In such a situation, you can group two or more columns to get the unique values and create an index which is called clustered Index. This also helps you to identify the record faster.

Example:

- Let's assume that a company recruited many employees in various departments.
- In this case, clustering indexing in DBMS should be created for all employees who belong to the same dept.
- It is considered in a single cluster, and index points point to the cluster as a whole.
- Here, Department _no is a non-unique key.

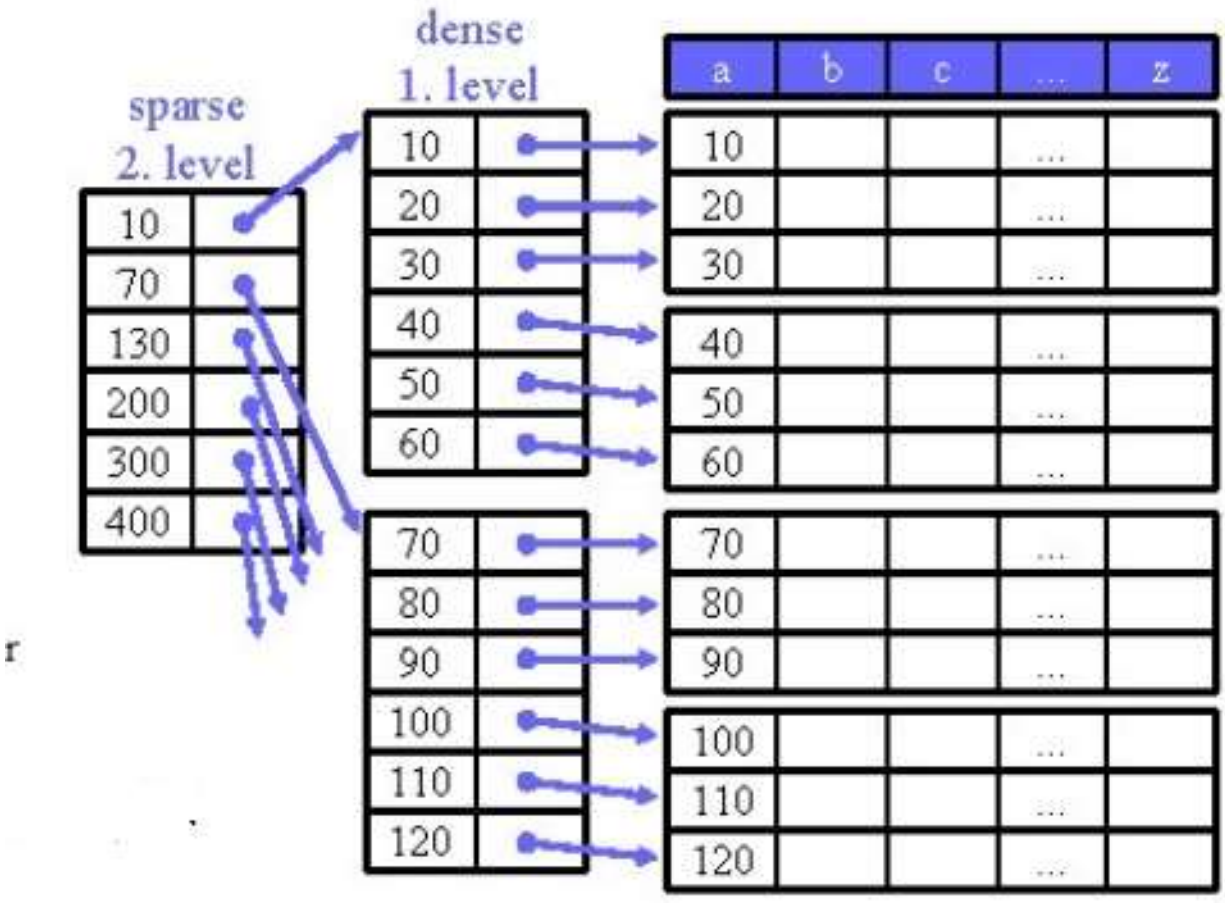


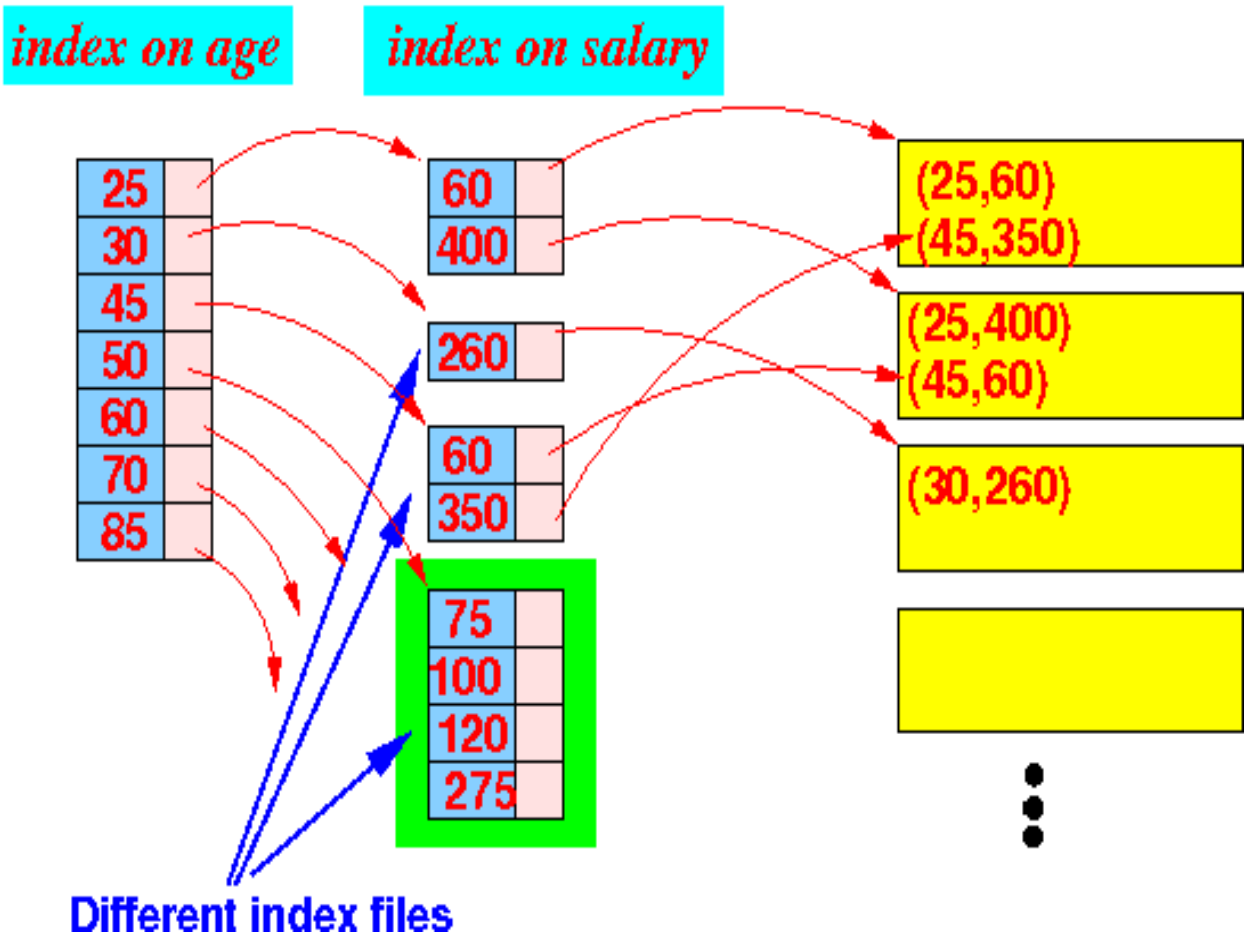
- **The secondary Index in DBMS** can be generated by a field which has a unique value for each record, and it should be a candidate key. It is also known as a non-clustering index.
- This two-level database indexing technique is used to reduce the mapping size of the first level. For the first level, a large range of numbers is selected because of this; the mapping size always remains small.



DATABASE MANAGEMENT SYSTEM UNIT-2 NOTES-2021

- **Multilevel Indexing in Database** is created when a primary index does not fit in memory.
- In this type of indexing method, you can reduce the number of disk accesses to short any record and kept on a disk as a sequential file and create a sparse base on that file.





Multi-key indexes

Multi-key indexes are **used to index all the elements of an array, or all the elements and/or all the keys of a map**. As a result, for each table row, the index contains as many entries as the number of elements/entries in the array/map that is being indexed (modulo duplicate elimination).

Multiple-key indexes

- Index on one attribute provides pointer to an index on the other.
- Let *V* be a value of the first attribute.
- Then the index we reach by following the pointer for *V* is an index into the set of points that *have V for their first value in the first attribute* and *any value for the second attribute*.