17/12/19

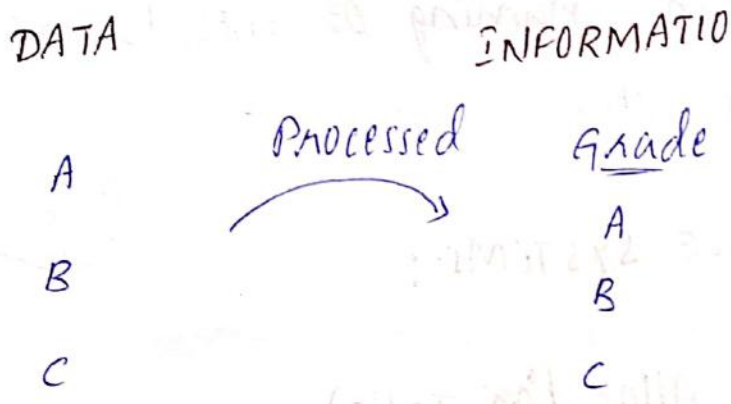
DATA :

It is the collection of raw facts, figures, numbers, symbols which is unorganised and does not carry any meaning.

Data does not have any meaning on its own.

INFORMATION :

It is the processed data which is organised and has useful meaning. We generally perform some operations on data to obtain information.



KNOWLEDGE :

It is the collection of information and with the help of experience inside advanced techniques (Data Mining / ML / AI techniques) we obtain knowledge or interesting patterns which will

benefit the individual or the organisation when taking business decisions.

CONVENTIONAL FILE SYSTEM :

→ After the development of computers people/ organisations tried to digitize their organisations informations using conventional file system that are supported by respective OS.

→ Each OS has it's own file system.

→ A File System is a process/software package that manages and organises the files, data on a storage media.

→ It supports various operations such as storage management, naming of files, folders, access previliges etc.

TYPES OF FILE SYSTEMS :

1. FAT (File Allocation Table)

It is supported by windows OS and was used for floppy disks and adopted for hard disks FAT 12, FAT 16, FAT 32 etc.

2. GFS (Global File System)

Supported by Linux OS.

3. HFS (Hierarchical File System)

Developed for Mac OS

4. NTFS (New Technology File System)

NTFS has become the default file system for Windows NT OS

18/12/19

DISADVANTAGES OF CONVENTIONAL FILE SYSTEMS :-

Conventional File System Vs DBMS

→ Data Redundancy and Inconsistency
→ Data Integrity
→ Atomicity of data
→ Data isolation
→ Concurrent execution anamalies
→ Data isolation
→ Security problems

Data Redundancy :-

Repetation of data in multiple files / Duplication

19/12/19

DATABASE :

Database is a collection of related data or information.

DBMS :-

(Software package consisting of set of programs for efficient storage of data and for managing the data.)

The data in DBMS is related to each other.

(Managing of data refers to various operations such as retrieving, modifying and creation of data)

(The primary goal of DBMS is to provide an efficient way inorder to store and retrieve the database information in a convenient & efficient manner.)

(DBMSs are designed for storing large amounts of information)

FILE SYSTEM Vs DBMS (Disadvantages of File Systems)

1. Data Redundancy and Inconsistency :-

It refers to existence of same data as multiple copies which we call it is repetation of data or duplication of data. It basically leads to wastage of storage space as well as Inconsistency of data.

Inconsistency refers to incorrect data.

2. Difficulty in accessing data :-

As data may be stored in different formats we are not able to retrieve the required data in an efficient manner.

3. Integrity Problems :-

The data stored in the file system has to satisfy certain constraints. As new constraints emerges it was difficult to implement them.

4. Atomicity Problems :-

Whenever a failure occurs for a system it is important that the data has to be restored to the consistent state that was existing before the failure has occured.

5. Concurrent Access Anamalies :-

In order to have good performance and faster response from the systems we design the systems to be supporting multiple users to access the data simultaneously.

In these cases concurrent updation of data may lead to inconsistent state of data.

6. Data Isolation :-

It refers to when and how the result of an operation is made visible to all other users of the system.

7. Security Problems :-

Every user of the database is not allowed to access all the parts of database. There should be certain security mechanisms implemented for the data.

With all the above problems the people have developed DBMSs which are able to solve the above problems.

DATA MODELS :

→ Data model is a collection of conceptual tools for describing the data, relationships among the data and constraints on the data.

→ Data models provide logical structure for databases.

i) Flat File Model :-

→ In this data model a single table is used to store the entire information.

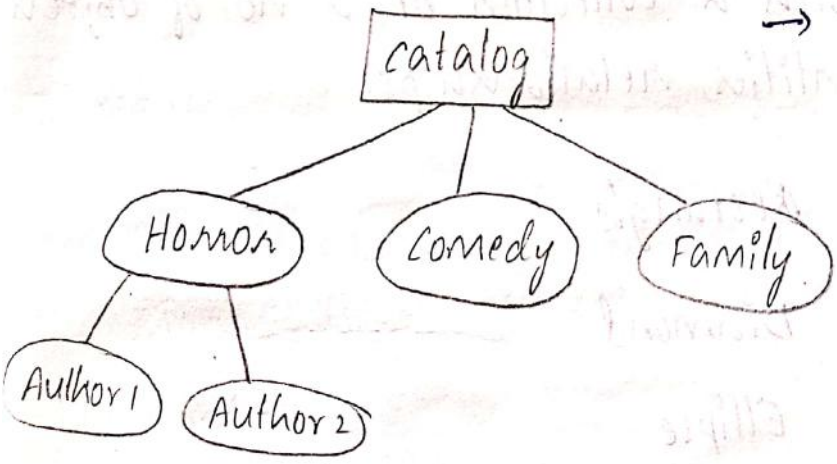
ii) Hierarchical Data model :-

→ In this, one data object will be choosen as the parent data object and other data objects are represented as sub-ordinate data objects.

→ It organises the data in a tree like structure and parent-child relationship exists among the data objects.

→ A parent data object can have multiple child data objects but a child data object can have only one parent object.

→ The order in which the sibling records are stored, the same order is used for storing data in physical media.



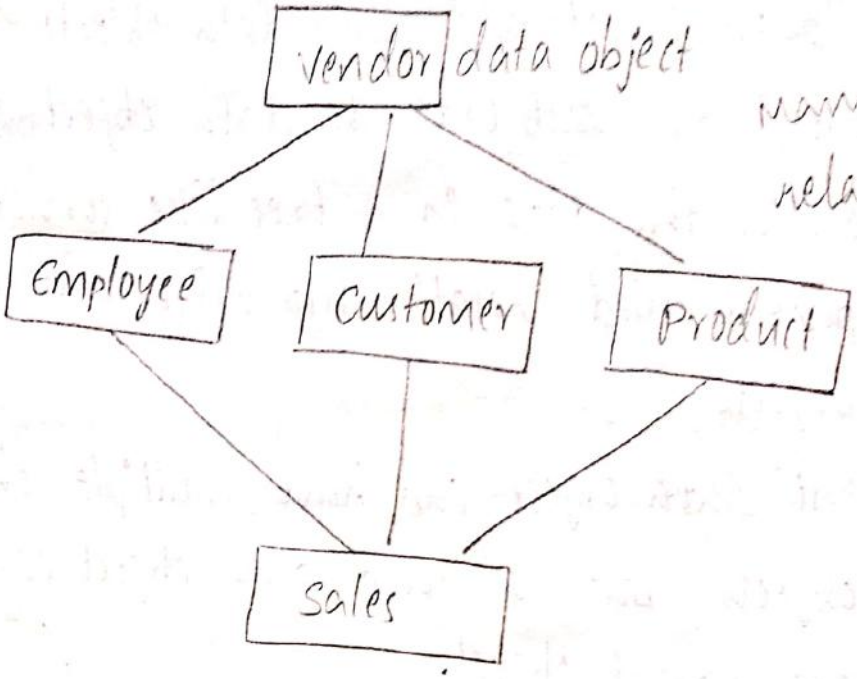
→ This model allows one-to-many relationship among data objects.

iii) Network Model :-

→ It allows data to be stored in the form of graph like structure.

→ It supports many-to-many relationship among data objects.

→ A child data object can have multiple parent objects.



many-to-Many relationship exists

iv) Entity-Relationship Model :-

→ In this the database is designed in a graphical manner with a collection of a no. of objects such as entities, relationships.

Entity - Rectangle

Relationship - Diamond

Attribute - Ellipse

v) Relational model :-

→ In this model, tables are used to represent the data/entities and the relationships among the data.

→ Each table/relation has a no of columns which are the attributes of the entity.

→ Relational model is an example of record based model.

→ Generally, first the E-R model is created for a database and it is converted into relational model.

vi) Object - Oriented Data model :-

→ This is an extension to E-R model which has notations for representing objects, methods, encapsulation, generalization & aggregation.

→ Multimedia DB, Hypertext DB use this kind of data models.

vii) Object relational data model :-

→ This data model will combine the features from relational model and object oriented models.

Speciality Data models :-

These data models can be used for modelling and design of unstructured data such as web data.

31/12

LEVELS OF DATA ABSTRACTION :

→ In order to hide the complexity of database and to provide easy accessing for the user, the developers of the DBMS use different levels of data abstraction.

→ The complexity of the databases is hidden from the users with the help of 3 different levels of data abstraction.

* Physical Level :-

→ This is the lowest level of data abstraction which describes how data is actually stored on the physical media.

→ At this level the DBA will decide the appropriate file organisation and the secondary storage media to be used, any indexes to be defined any compression methods required before storage etc.

* Logical level or conceptual level :-

→ At this level the main focus is on what data has to be stored and what relationship exists among the

data and the various constraints that are to be specified on the data.

→ At this level the data is described in terms of database models.

* View Level :-

→ This is the highest level of data abstraction which describes a part of entire database.

→ The database system will provide different views for different users from the same database.

→ These views are known as sub-schemas.

→ At each level of data abstraction we have different schemas.

→ The physical schema at physical level will specify the storage details.

→ The conceptual schema at the logical level will describe the data to be stored.

→ The sub schema at the view level will specify one or more relations or views.

* INSTANCE :-

The collection of information stored in the database at a particular point of time is called instance of the database.

* SCHEMA :-

→ The structure (or) the overall the design of the database is called schema.

→ These are defined by DBA.

→ Once schemas are defined they are very rarely changed.

→ with the help of data abstraction DBMS achieves data independence at the physical level and at the logical level.

* PHYSICAL DATA INDEPENDENCE :-

Modifying the physical storage structure of the database without affecting the application programs at the view levels.

* LOGICAL DATA INDEPENDENCE :-

It is the ability the modifying the logical schema without affecting the external views and application programs.

DATABASE USERS :-

→ The way in which the users interact with a database we can categorize the users of the database as normal users and DB administrators.

→ Under the normal users category we have,

- Native Users :- They interact with the system with the help of application programs. Their main interface is different kind of forms and reports. These are also known as Unsophisticated users.
- Application Programmers :- They develop different application programs for DBs. They mainly focus on UI development with the help of RAD tools.
- Sophisticated users :- They interact with DBs by writing queries. OLAP tools can be used inorder to analysis and explore the data in DBs, datawave houses.
- Specialized users :-

They will write specialized applications for knowledge base DBs, Expert Systems, CAD Systems.

DBA :-

→ The person who has central control on the data.

→ The various roles of DBA are,

* Schema definition.
* Storage structure and defining accessing methods.
* Schema and physical storage modification
* Granting of authorization for database access.
* Routine maintenance of DBMS.

21/1/2020

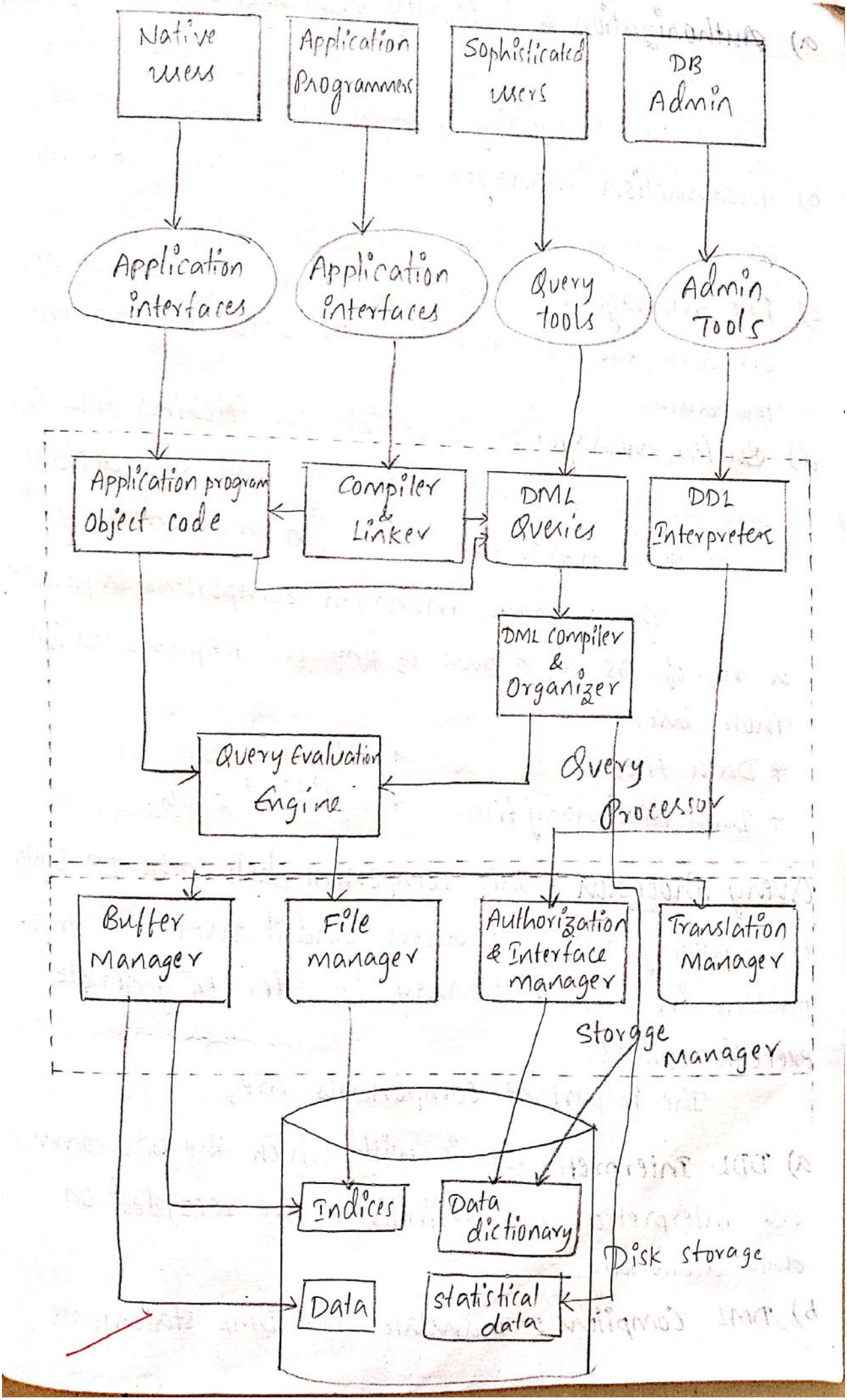
STRUCTURE OF DBMS :-

→ The DBMS uses various software components to execute a query. The major components are,

* Storage manager
* Query processor

Storage manager : It provides the interface between the low level data stored in the databases and application programs, queries submitted to DB.

o The important components of storage manager are,



a) Authorization & Integrity manager :- It will test the satisfaction of various integrity constraints & checks the authority of users who access the DB.

b) Transaction manager :- It ensures that DB remains in a consistent state despite of system failures.

c) File manager :- It manages the allocation of space on disk storage and the DS used to represent the data.

d) Buffer manager :- Responsible for fetching data from disk storage into main memory and also decides what data has to be in main memory

The storage manager component implements a no. of DS as a part of physical implementation Those are,

* Data files
* Indices
* Data dictionary files
* statistical data.

Query Processor : This component helps the DB System to simplify the data access and it uses the info. present in data dictionary in order to generate execution plans.

The important components are,

a) DDL Interpreter :- It is with which the DDL commands are interpreted and definitions are recorded on data dictionary.

b) DML Compilers :- Translates the DML statements

into execution plans consisting of low level instructions that every evaluation component understand. Any query is translated into a no. of actions and for each of the action a no. of execution plans are generated.

The DML compiler component will perform query optimization to choose the low cost execution plans.

Query evaluation engine will execute the low level instructions present in the execution plan.

* APPLICATIONS OF DBMS :

→ Telecommunications
→ Finance
→ Marketing
→ Online Reservation Systems
→ E-commerce application
→ Banking
→ Defence
→ Results
→ Railways
→ Universities

3/1/2020

DATABASE DESIGN AND ER DIAGRAMS :

Major steps in design of Database :-

* Requirement analysis
* Conceptual DB design
* Logical DB design
* Schema refinement
* Physical implementation
* Application and security design.

Requirement analysis :-

In this step the DB designers will gather the info. regarding what data has to be stored, who want to use the DB and the performance requirements of DB. This info. can be gathered by conducting survey, meetings, interviews with end users.

Conceptual DB design :-

Based on the requirements a high level semantic description of the DB is done. We use E-R model for this purpose.

Logical DB design :-

A suitable DBMS software is used to implement the structure of the DB based on conceptual design.

Schema Refinement :-

An analysis will be done one the initial schema so that any modifications to the structure can be done.

Physical implementation :-

It will focus on the file organisation selection, index design in order to meet the specified performance requirements.

Application & security design :-

Identify the role of each entity or user and based on this applications (or) interfaces for the DB are designed by choosing appropriate programming language.

11/1/20

ENTITIES, ATTRIBUTES AND ENTITY SETS :

Entity :-

→ An Entity is an object in the real world that is distinguishable from other objects
→ It can be either a logical or physical object.

Entity set :-

→ Collection of entities of similar type is called as entity set.
→ The entity sets need not be disjoint.

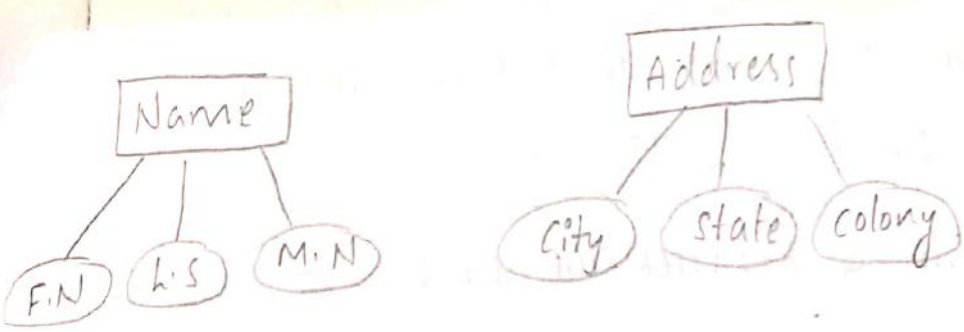
Attributes :-

→ An entity is described using numbers of properties called attributes.
→ All the entities of set will have same attributes
→ The values of attributes are used to uniquely identify an entity in the set.
→ For each attribute there are set of permitted values called as "Domain of the attribute" or 'value set of attribute"
→ A DB generally includes a collection of entity sets.
→ An entity is represented by rectangle and attributes are represented by oval.
→ Each entity set will choose a key which consists of minimal set of attributes whose value will uniquely identify an entity in the set.

Types of Attributes :-

* Simple & Composite :- The attributes which are not divided into sub parts are called simple attributes
Ex: Roll.No, Ph.No, Acc.No etc.

The attributes which can be divided into sub parts are called composite attributes
Ex: Name, Address



* Single valued & Multi valued : Attributes having single value for an entity are called single valued attributes.

Ex: Roll.No, IDs

Attributes having set of values for a specific entity are multivalued attributes.

Ex : mail ID, Ph.no.

* Derived : The value of this type of attributes can be derived from the values of other related attributes (or) entities.

The value of derived attribute is not stored but it can be calculated whenever required.

The attributes which are used for finding the value of derived attribute are called as "Base attribute" (or) "Stored attributes"

NULL Values :-

The attributes of an entity sometimes does not have a value for it. This can be indicated in DBs as NULL value.

A NULL value means that either the value is unknown or it is not available.

RELATIONSHIP & RELATIONSHIP SET :

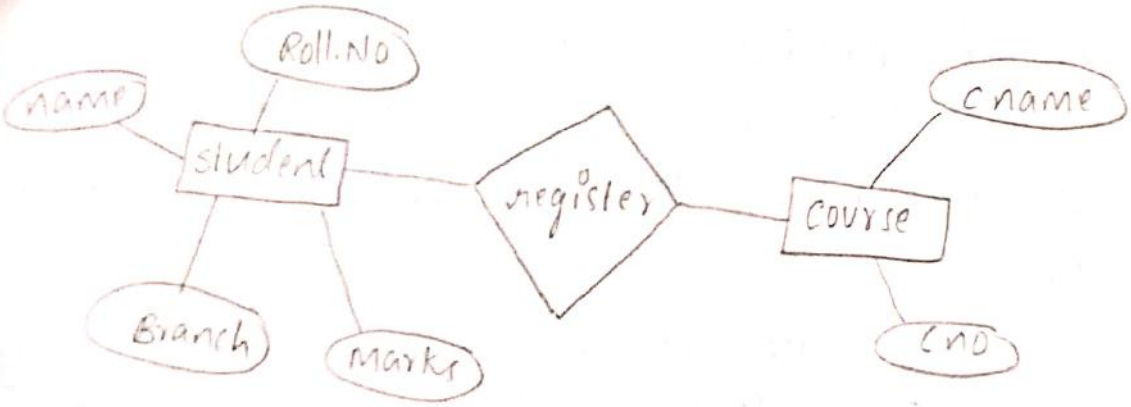
→ A relationship is an association among several entities

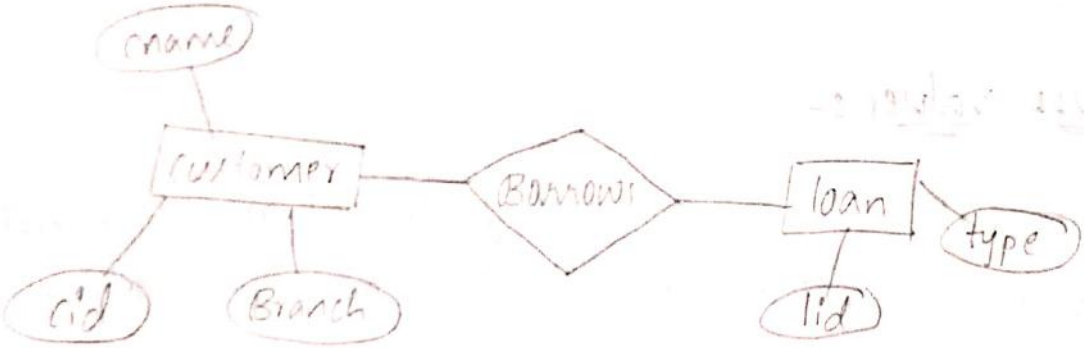
→ A relationship set is a set of relationship of same type.

→ The association between entity set is referred as participation which means that the entity sets E_1, E_2, E_3 participate in a relationship R.

→ Represented by diamond symbol.

Ex :

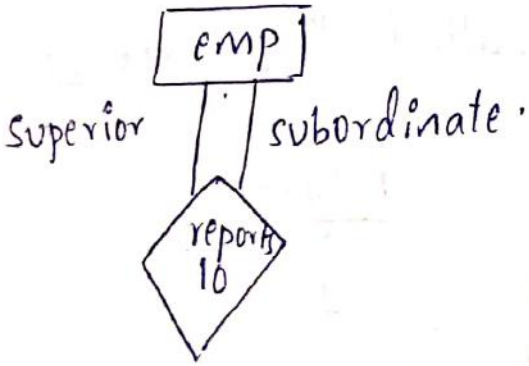




→ The role and entity plays/takes with a relationship will be mentioned in the E-R model.

→ The role of an entity can be more than one with relationship.
Such kind of relationship is called as "recursive relationship"

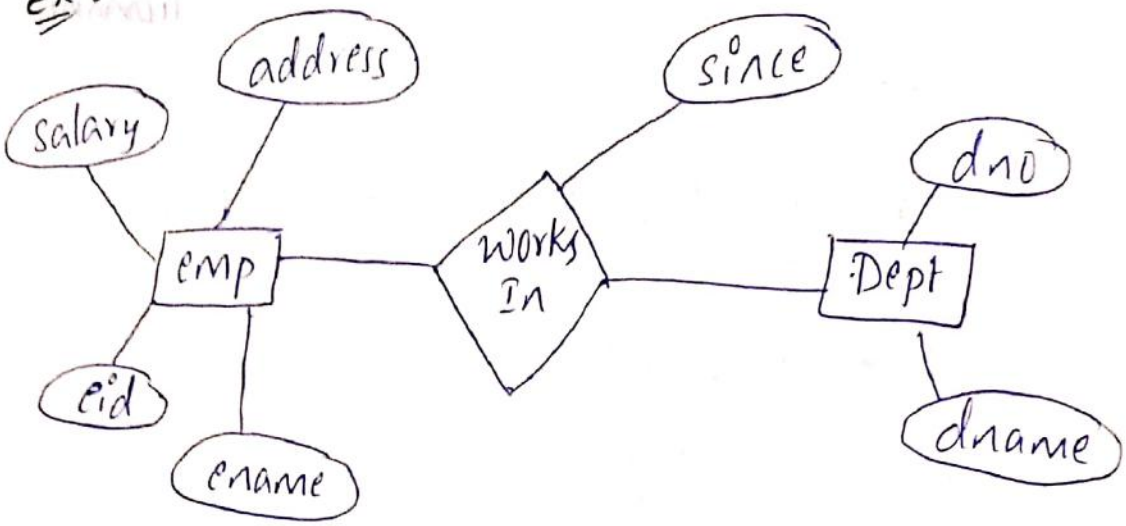
→ In recursive relationship the role names have to be specified explicitely.



→ A relationship may have attributes known as the discriptive attributes which are basically used to record the information about the relationship.

→ A relationship must be uniquely identified by participating entities without reference to descriptive attributes.

Ex:



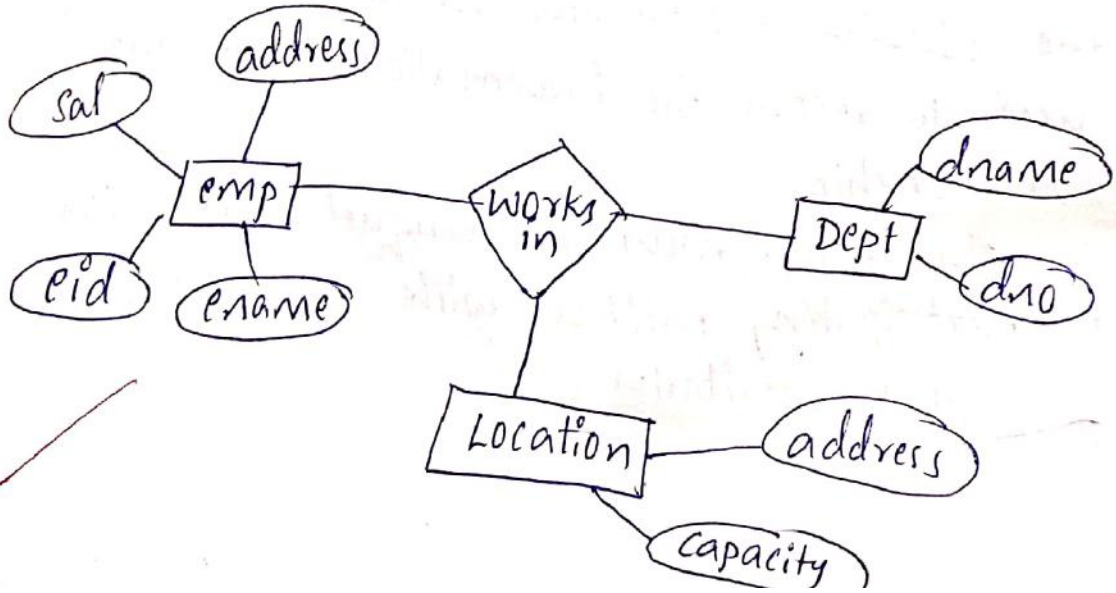
→ If two entities are involved in a relationship it is a "Binary Relationship"

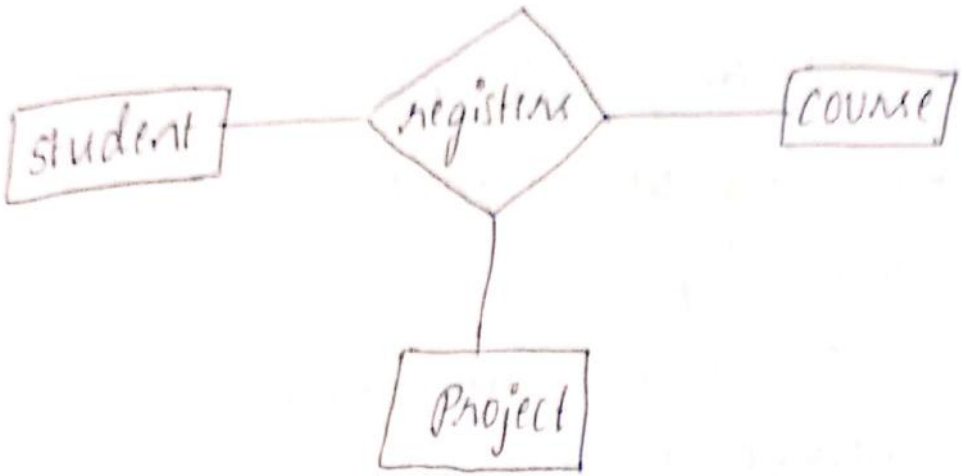
→ If three entities are involved it becomes "Ternary relationship"

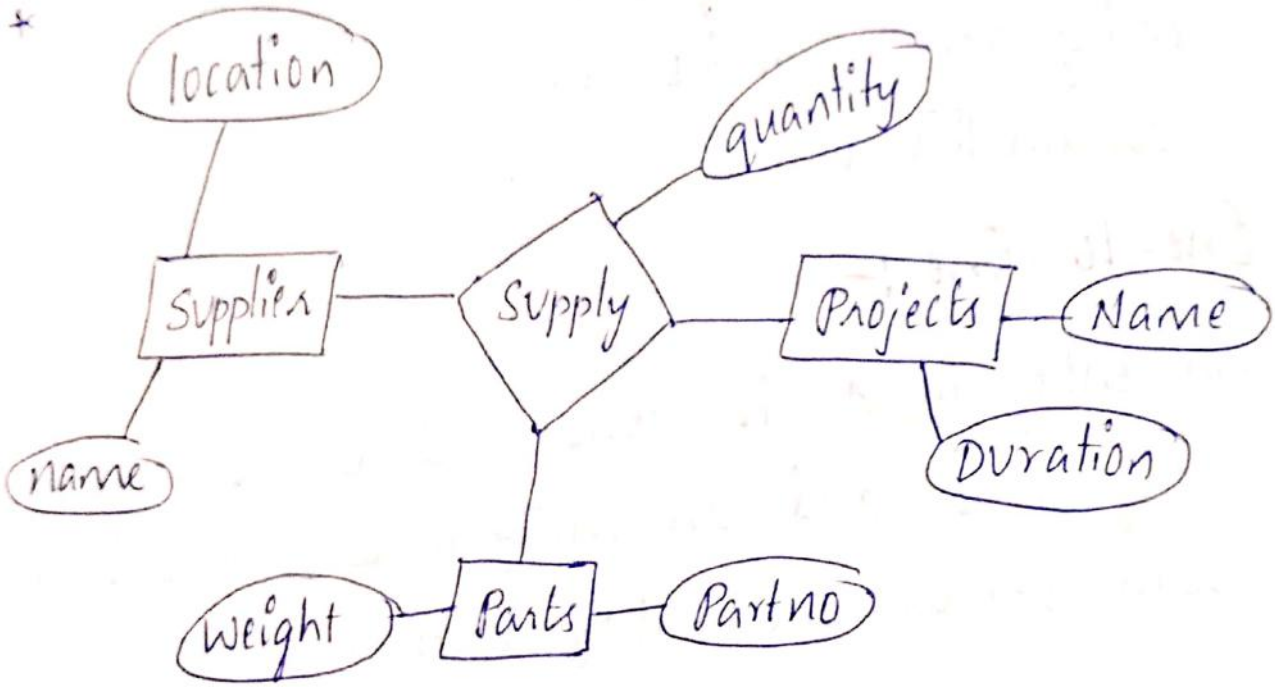
→ The no. of entity sets that participate in a relationship is called "Degree of relationship"

Relation	Degree
Unary relation	1
n-array relation	n

Ex:







CONSTRAINTS IN E-R MODEL :

→ In E-R diagrams certain constraints can be specified to which the contents of DB must confirm.

→ The two types of constraints during E-R design of DB are.

Mapping Cardinality :

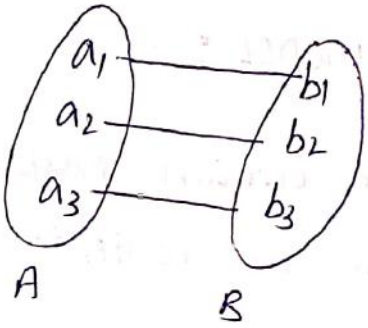
→ Mapping cardinality will indicate the no. of times an entity of an entity set participate in a relationship with another entity.

→ With this constraints we can mention how many instances of an entity relate to one instance of other entity.

→ Consider a binary relationship (or) between entity sets A and B and the various mapping cardinalities will be,

One-to-One :-

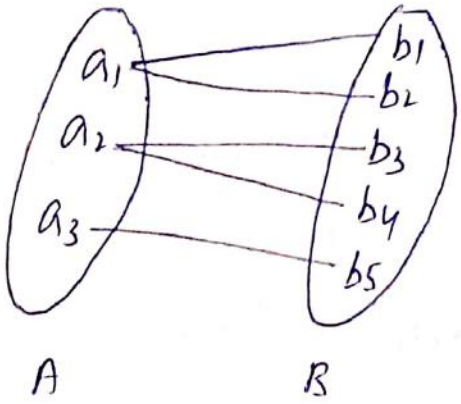
An entity in A is associated with atmost one entity in B and an entity in B is associated with atmost one entity in A.



Each manager occupies atmost one office and an office has one manager.

One-to-many :-

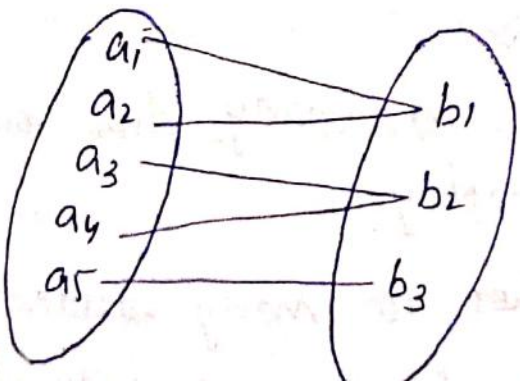
An entity in A is associated with any number of entities in B. However, an entity in B can be associated with atmost one entity in A.

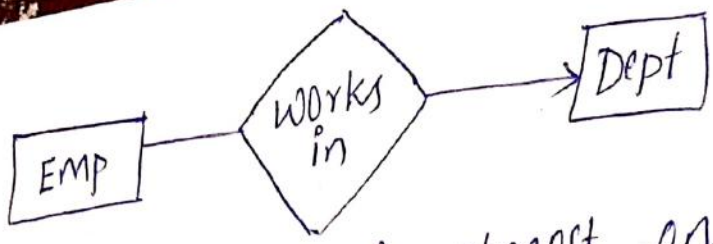


Each dept. has many employees and one employee belongs to one dept. only.

Many-to-one :-

An entity in A is associated with atmost one entity in B and an entity in B is associated with any no. of entities of A

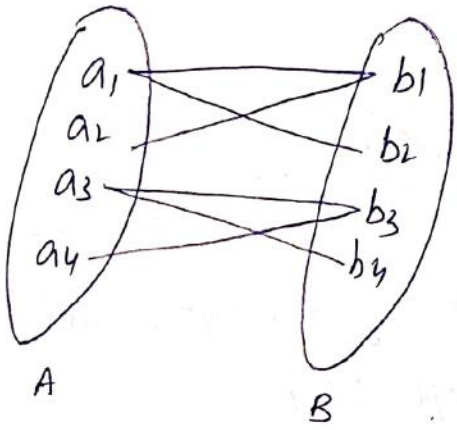


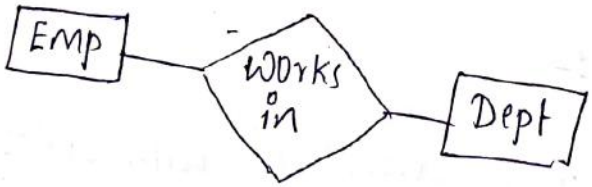


An employee works in atmost one dept and each dept. has many employees working in it.

Many-to-many :-

An entity in A is associated with any no. of entities in B and an entity in B can be associated with any no. of entities in A.





* An employee works in many dept. and a dept can have many employees.
* A Lecturer teaches to many students and a student is taught by many teachers.

9/1/2020

DIFFERENT TYPES OF KEYS FOR RELATION :

→ The entities in an entity set can be uniquely identified with the help of the values of attributes

→ No two entities in an entity set are allowed to have the same values for attributes.

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

→ The keys also help to uniquely identify the relationships from each other.

→ The different types of keys that can be identified for an entity or relation are,

Super Key :

→ A super key is a set of one or more attributes taken collectively to identify uniquely an entity in an entity set.

→ For schema student (R-no, name, branch, marks). The super keys can be,

(R-no)

(R-no, name)

(R-no, branch)

(R-no, marks)

20/1/2020

Candidate Key :

A candidate key is a super key with minimum no. of attributes required to uniquely identify each tuple in a relation.

Candidate key will not contain extra attributes. A relation can have more than one candidate key.

Ex:

Books (Bname, Bauthor, Bpub, yrofpub, Bid)

↓

candidate keys = (B_id)

(Bname, author)

Primary Key :

A Primary key is a candidate key which consists of one or more attributes, which can uniquely identify each tuple of the relation.

If a relation has only one candidate key it becomes the primary key.

If there are more than one candidate keys, one of them is chosen as primary key and other keys are called as 'Alternate keys'

Foreign Key :

It is used to define relationship between the tables (or) relations.

Foreign key is an attribute or set of attributes in a relation whose values match with the primary key of another relation.

A relation can have more than one foreign keys.

Ex: Employee (Foreign key) Department (Primary key)

e_id	e_name	d_id

d_id	d_name

Composite Key :

A primary key consisting of more than 1 attributes is a composite key.

* Primary keys cannot take NULL values, duplicate values.

  We can have only one primary key for a relation.

* Foreign key can take NULL values and duplicate values.

21/1/2020

Weak entity set :- An entity set may not have the sufficient no. of attributes to form a primary key. Such entity is called as weak entity.

Strong entity :- An entity which has sufficient attributes to form a primary key is known as strong entity set.

→ For a weak entity to be meaningful it must be associated with another entity set with the help of relationship

→ The other entity set that is used by the weak entity is known as identifying entity or owner entity set.

→ The relationship b/w the weak & the identifying entities is known as identifying relationships.

→ The weak entity is said to be existence dependent on the identifying entityset.

→ The identifying relationship is one to many relationship from identifying entity to weak entity.

→ One owner entity may be associated with one or more weak entities but a weak entity has a single owner entity. (one-to-many)

→ The set of attributes of weak entity that are chosen to uniquely identify a weak entity for a given owner entity is called "Partial Key"

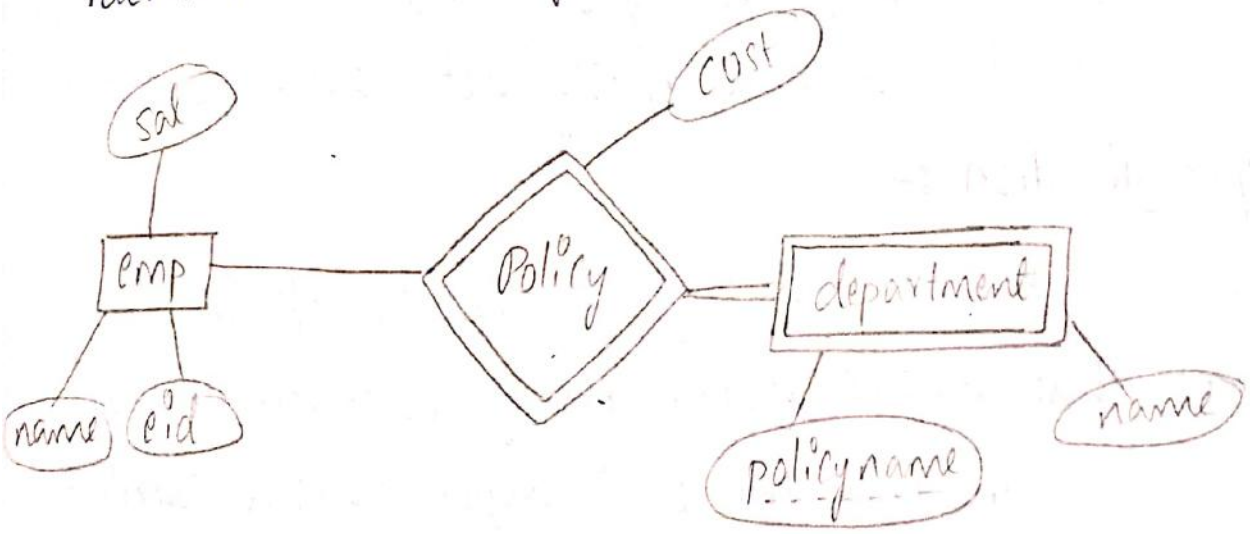
of the weak entity.

→ The primary key for the weak entity is the union of partial key, primary key of identifying entity.

→ A weak entity can have more than one identifying entities.

→ A double lined rectangle is used for representing weak entity.

→ A double lined diamond is used to represent identified relationship.



Partial key of department : Policy name.

Primary key of the dependence is : (eid, policyname)

Extended E-R features and class Hierarchy :-

→ Some entity sets in the real world can be sub grouped

→ We can classify the entities of an entity set into a no. of such sub entities.

→ In such a case sub entities will be inher
the attributes of the super entity.
→ Such kind of relationship is known as Is a
relationship
→ The Is a relationship is a kind of class
hierarchy.
→ The class hierarchy can be created with
generalisation or specialisation process.

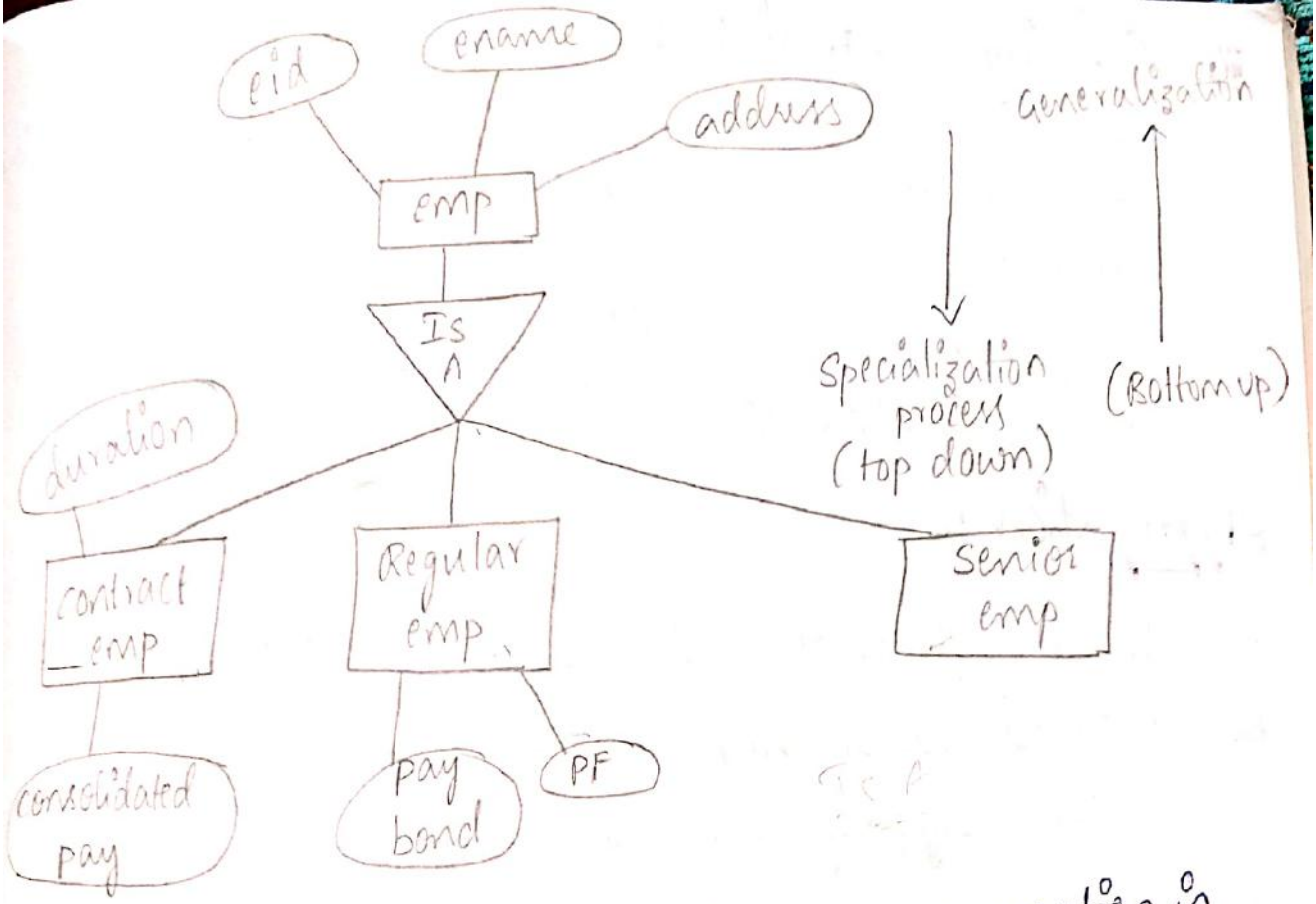
Generalisation :-

- In this we will try to identify the common
- characteristics of an entity set. With these comm
characteristics a new entity set can be created

- Specialisation :-

The process of identifying subsets of an entity sets that share super entity characteristics.

In this the (pro) super class is identified first and sub class are defined afterwards.



The various constraints that we can mention in IsA Relationship are,

i) Overlapping constraints :- If a subentity belongs to more than one sub entities we say that the sub entity satisfy overlapping constraints.

In the above diagram, the senior emp & regular emp entities satisfy overlapping constraints.

ii) Disjoint constraints :- If the sub entity does not belong to any other sub entity we say that the sub entity satisfy disjoint constraints.

In the above diagram, contract emp & regular emp satisfy disjoint constraints.

iii) Covering constraints :- If every instance of the entity belong to one of the subentities we say that super entity is covered by subentities.

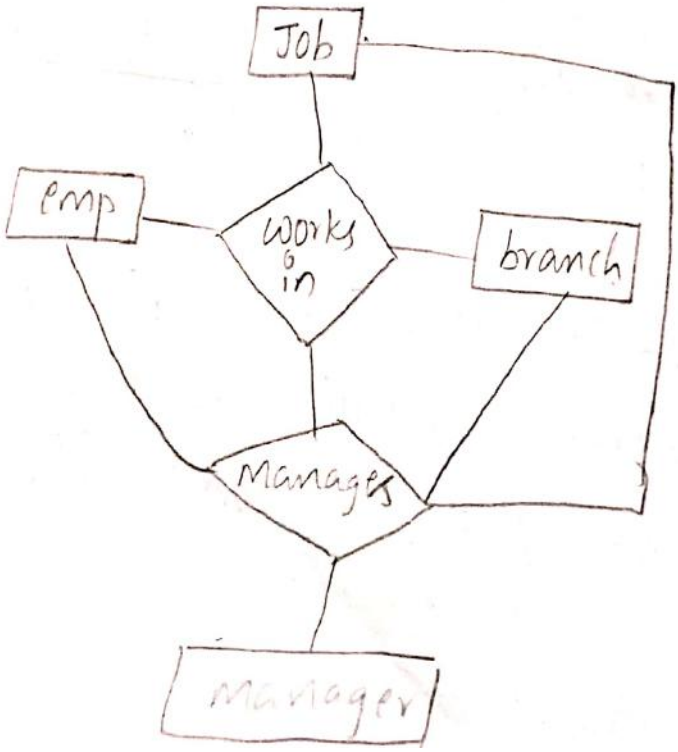
In the above diagram, emp is the covering constraint.

Aggregation :-

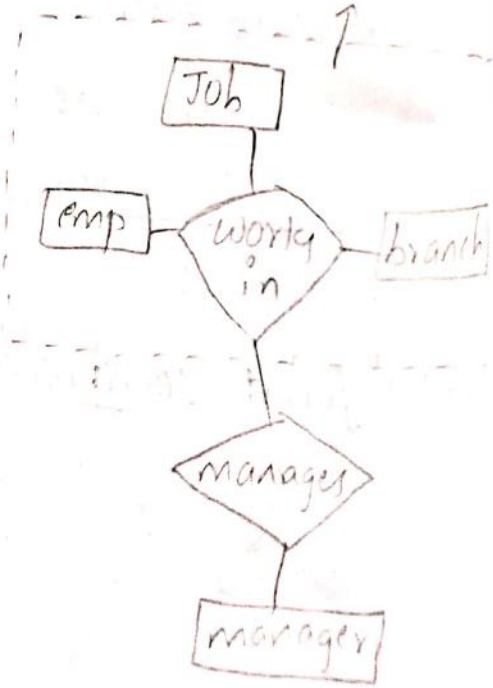
It is an abstraction in which a relationship will be treated as a higher level entity.

The aggregation relationship allows to indicate that a relationship set participate in another relationship set.

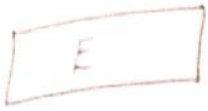
Represented with the help of ..... (dashed) rectangle box.



Aggregation



E-R Notations :-



- Entity set

- Derived attribute

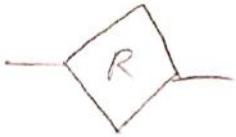
- Relationship set

- Total participation

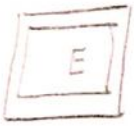

- Attribute

Primary key

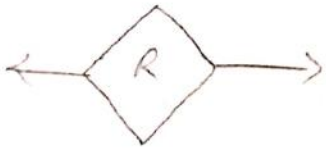

- Multivalued attribute



many to many relationship



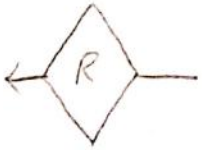
- weak entity



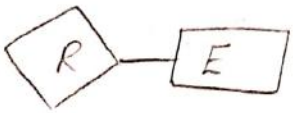
one - to - one


- Identifying relationship

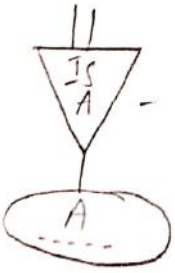
Many to one



One to many relationship

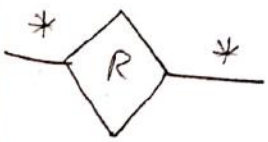


- Partial participation



- Total participation of Is A

- Partial key

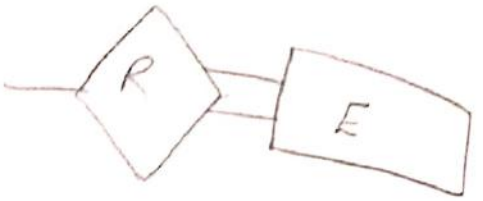


- Many to many

one to one

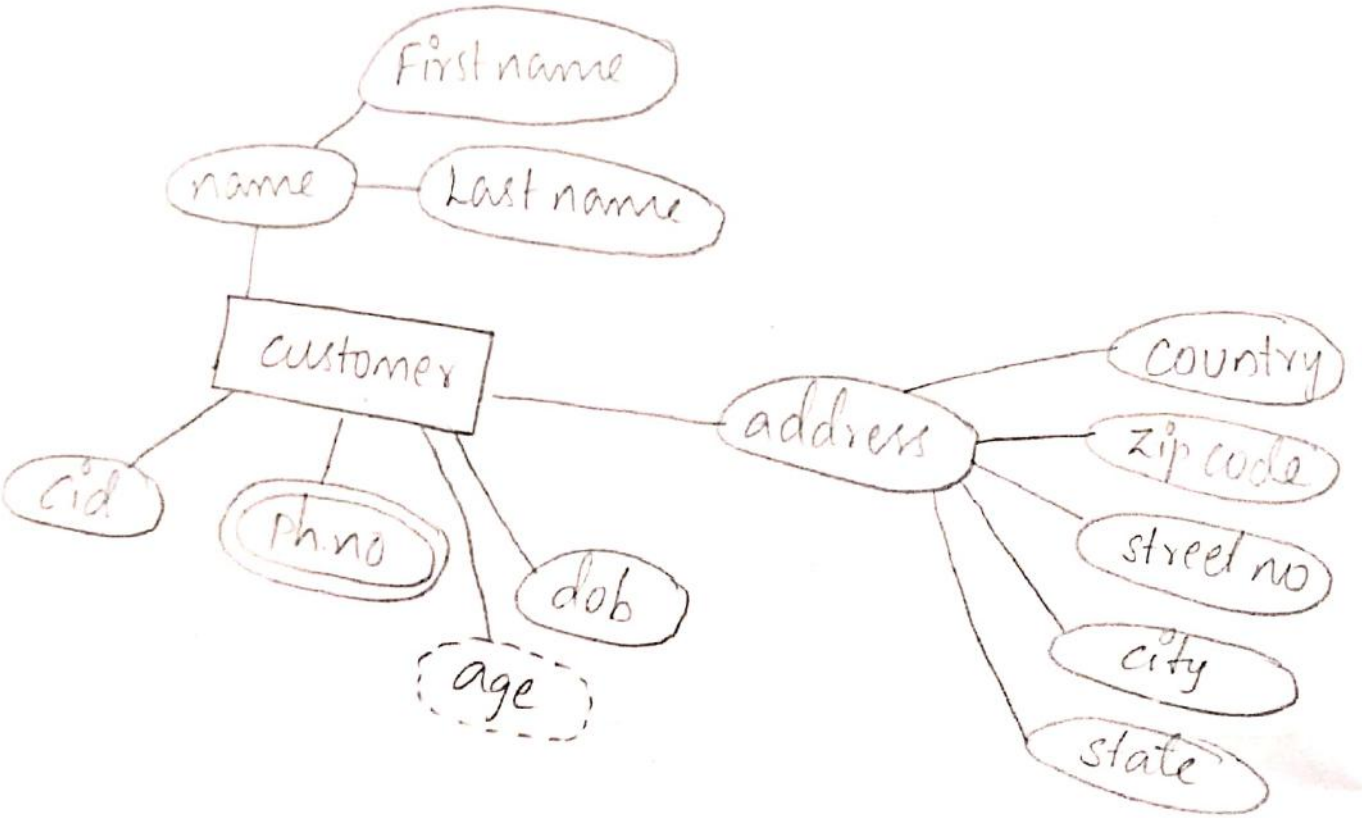
Many-1
one

one - to - many



Recursive relation

- Partial participation



23/1/2020

UNIT-2

RELATIONAL MODEL

→ The RDBMS uses relational model in order to store the data.

→ Relation is the main structure in RDBMS.

Relational Schema :-

It indicates the name of the relation, names of attributes, domain of each attribute.

Relational Instance :-

The set of tuples/rows present in a relation at a particular point of time is called instance of a relation.

Degree of a relation :-

The no. of attributes present in a relation.

Cardinality of a relation :-

The no. of tuples present in a relation is known as the cardinality of a relation.

emp

Degree = 3
Cardinality = 4

name	e-id	sal
xxx	1	2000
yyy	2	4000
zzz	3	6000
ppp	4	5000

} - schema

} - Instance

CONSTRAINTS ON RELATION :

1. Unique constraint
2. Not null constraint
3. check default constraint
4. Primary key constraint
5. Foreign key constraint
6. Domain and domain extended constraint.

→ A constraint is a relation which is basically used in DBMS to prevent entry of incorrect information.

→ The constraints are mentioned in the schema and it is checked by DBMS during DML command execution.

* Domain , Domain Extended constraints

The Domain constraint is mentioned during the schema creation with the help of create statement.

It mentions data type of each variable.

Ex:

```
Create table student (name char(10), Rollno number(10),
                      marks number(10), address char(15));
```

Once the schema is created during execution of DML the constraint is verified. If it satisfies, the stmt is executed and DB is modified.

If DML statement violates data is not modified in DBMS.

ex:

```
insert into student values ('abc', 123, 200, 'hyd');
insert into student values (223, 123, 200, 'hyd');
                                    ↓
                                violated.
```

Domain extended constraint :-

This is used when we want an attribute to take some set of values. If the constraint is mentioned in create command with check clause.

Ex: Roll-no has to take 501 to 570 values

```
create table student (name char(10), Roll-no number(10),
check (Roll-no between 501 and 570), marks number(10),
address char(20));
```

(OR)

```
create table student (name char(10), Roll-no number(10),
                      marks number(10), address char(10),
                      check (Roll-no between 501 and 570));
```

(OR)

```
create table student (name char(10), Roll-no number(10),
                      marks number(10), address
                      char(10), constraint c1 check
                      (Roll-no between 501 and 570));
create table student (name char(10), Roll-no number(10),
address char(10), marks number(10), constraint c1
```

check (Roll-no between 501 and 570), constraint c2 check (address in ('Hyd', 'Vizag')));

Once the schema is created during the execution of insert and update these constraints are verified.

Ex:

insert into student values ('abc', 503, 250, 'hyd');
insert into student values ('xyz', 592, 350, 'hyd');

Not executed.

update student set Roll-no = 564 where address='hyd'
update student set Roll-no = 504 where marks=253.

update student set address = 'Bengalore" where name = 'abc';
insert into student values ('abc', 501, 250, 'Pune');

* Not null constraint

Not null constraints allows you to restrict a column from having null values.

Once a column is declared as not null we cannot pass a null value in that column.

Ex:

create table customer (c-id number(10) not null, c-name char(10), age number(2) not null);
insert into customer (c-id, c-name, age) values (111, 'abc', 20) ✓

insert into customer (cname, age) values ('xyz', 31) ✓
insert into customer (c-id, c-name) values (222, 'ABC'); ✓

* Primary key constraint

A single attribute or group of attributes can be declared as primary key.

The primary key attribute once declared cannot take null values, duplicate values, during insert and update statement.

Ex:

create table emp (e_id number(5) primary key, e_name char(10), location char(10) check (location in ('delhi', 'pune')));

create table books (b-id number(5), bname char(10), author char(10), yop number(5), primary key (bname, author));

update set e_id = null where location = 'delhi'); ✗

insert into emp values (207, 'abc', 'delhi');

This will not take duplicate, null values.

* Unique key constraint

If we want some attributes of a relation to have unique values along with primary key attributes we can declare the attribute as unique.

Unique key attributes will take null values,

they cannot take duplicate values.

We can have any no. of attributes declared as unique. There should not be comm attributes in primary key, unique key.

An attribute declared with not null, uniq constraints is similar to primary key.

Ex:

create table books (bid number(3) unique, bname ..... primary key (bname, author));

insert into books (123, 'xyz', ...);

insert into books (bname, author, yop) values ('xyz', 'abc', 1992);

* Default constraint

It is used to provide a default value for a column. During the execution of DML statements if no value is specified for that attribute, the default value will be taken.

Ex:

* create table customer (cid number(3), c_name char(10), location char(10) default 'HYD');
* insert into customer (cid, c_name, location) values (111, 'xyz', chennai);

insert into customers(cid, cname) values(111, 'xyz');

when this row is inserted for the location attribute 'HYD' is inserted.

* Referential Integrity Constraint (or) Foreign key:

The referential integrity constraint when specified on a table will allow to enter or update consistent data into the dependent tables.

Foreign key is basically the primary key of some other table known as parent or referred table.

The table in which we have the foreign key is known as child table or referencing table.

The foreign key values of child table must match with the primary key values of the parent table.

We can use same name for the foreign key to (or) different name with same datatype.

student

P.key

sid	name	location
1		
2		
3		
4		
5		

Task

F.key

Name of course	Duration	Fee	course id	sid

* create table student (sid number(5) primary key, name char(10), location char(10));

* create table task (nameofcourse char(10), duration number(5), fee number(6), courseid number(5) primary key, foreign key s-id references student

When we perform insert operation on parent no violation of foreign key constraint

If we perform on child there should be a row existing with foreign key attribute value in parent table.

When we perform update, delete operations DBMS verifies and performs these operations in 4 ways.

i) NO action :-

If a row from parent or child is deleted or updated, if we mention no action in the schema of child table no row is updated or deleted from parent, child tables. The query is not executed on the database. When we delete a row from child table, it will be executed on the child table.

ii) cascade :-

when a row is deleted from parent, updated from parent (or) child, the deletion, will be done on the parent table updation is done on both tables.

iii) set null :-

when deleted from parent, updation is done on parent table the row will be updated, deleted, from parent table, the corresponding row of the child table which has same primary value will be replaced with null.

iv) default :-

when a row is deleted from parent, updated from parent table the corresponding row will be replaced with the default value in child table.

create table task (nameofcourse char(10), duration number(5), fee number(5), ... sid number(3) foreign key references student on update no action);

(or)

on delete cascade();

on update set null();

on delete set default();

* We can have more than one foreign key for a table.
* A foreign key attribute can take null values.
* A foreign key attributes can take duplicate values.

27/1/2020

VIEWS :-

→ A view is a virtual table created from one or more than one tables.
→ The tables which are used for view creation are called as Base tables.
→ The content of a view is stored temporarily and the result of the view will be genesated when a user gives a query on it.
→ View is similar to a table structure consisting of set of columns and rows of data.

Syntax :-

Create [or replace] view viewname as select query [with [check option] [read only]];

Ex :-

* Create view v1 as select name, roll_no, branch from student;

* The above statement creates a view for a base table student.
* A view structure with above attributes will be created and it will be an empty structure.
* when we query the view, the data will be retrieved into the view from the corresponding base tables.

SQL> select * from v1;

DML operations on view :- (without check and read only option)

when we perform insert, update and delete operations on a view they will get reflected base table

SQL> create or replace view v1 as select address from student; (To change schema)

SQL> insert into v1 values ('xxx', 201, 'SE', 'HYD');

SQL> update v1 set address='Delhi' where name = 'xxx';

SQL> delete from v1 where name = 'yyy';

* Creation of a view with a condition:

SQL> Create view v2 as select name, roll-no, marks from student where marks > 500;

→ when we apply select statement on v2 it retrieves the rows from base table which satisfy the condition i.e marks > 500

→ when we perform any DML statement on v2 they will be executed on view only if they satisfy the condition otherwise they will be rejected by the database

SQL> update v2 set marks = marks - 400;

→ If the marks values goes below 500 this update is not executed on v2

whether the condition is satisfied or not DML stmts will be reflected on base table content

* With check option :

SQL>
Create or replace view v2 as select name, rollno, marks from student where marks > 500 with check option;

→ when DML stmt satisfy the condition on the view they will be reflected on view as well as base tables

→ when condition is not satisfied and with the check option in view creation the DML stmts are not reflected on views, base tables.

Ex: Update v2 set marks = marks - 400;
Row is not updated in view as well as base table if marks goes below 500.

* View with read only:

→ A view created with read only option cannot be modified with DML statement

SQL> create view v3 as select name, branch, marks from student where marks < 300 with read only;

→ In real time, the read only views will be created from more than one base table. Such kind of views are also known as Complex views.

Types of views :-

i) Simple View

→ Created from one base table.
→ Group by, Aggregate are not allowed.
→ also called as Updatable views.

ii) Complex view

→ Created from more than one base tables.
→ Group by, Aggregate are allowed.

Advantages :-

→ Modification of data is avoided
→ Simplifies query applications
→ Provides security.

Dropping View :-

Syntax :-

drop view <viewname>

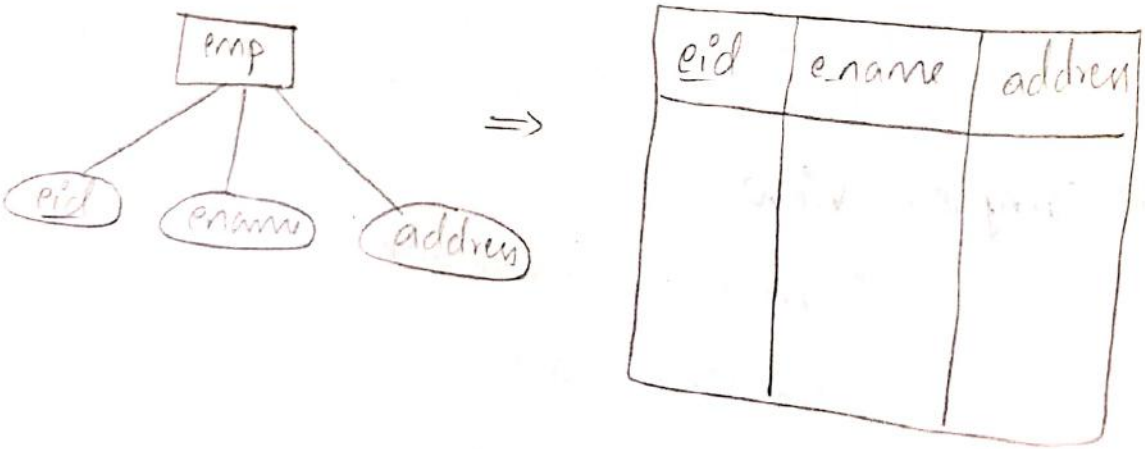
Ex :- drop view v1;

Slip 20

LOGICAL DATABASE DESIGN - ER MODEL TO RELATIONAL MODEL :

The ER model which represents a high level description of the DB will be used inorder to generate a relational DB schema in which most of the constraints can be captured.

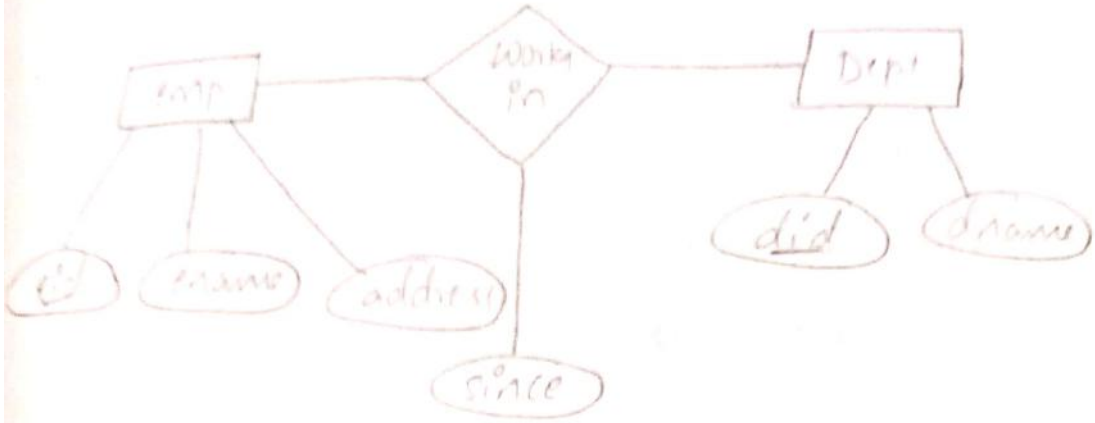
1. Entities into tables :-



The attributes of an entity will become the columns of table.

SQL > create table customer( )

2. Relationship set to tables (No constraints)



since	eid	did

→ In a relationship with no constraints we should identify each of the participating entity to form columns for the relationship table.

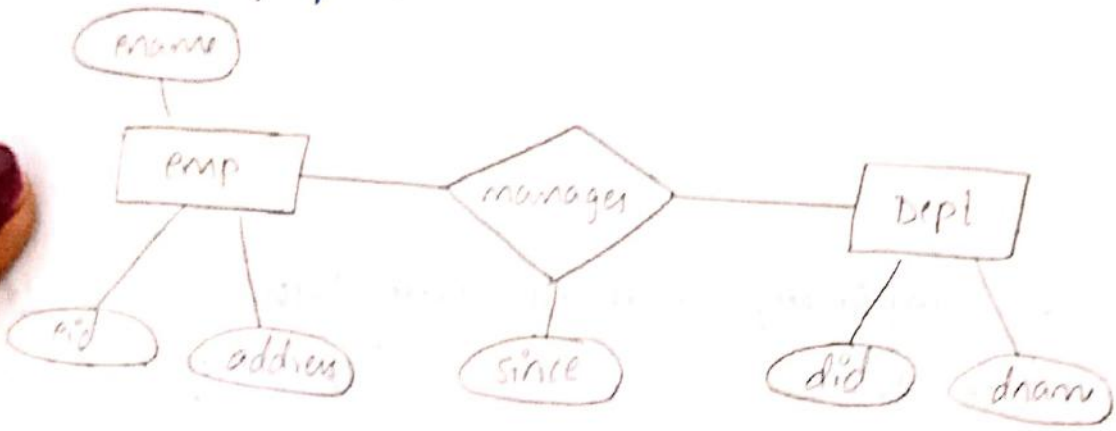
→ The attributes of the relationship will be primary key attribute of each participating entity as foreign keys and any descriptive attributes that are present.

SQL > Create table workIn(cid num(3), did number(3), since number(3) /date, primary key(cid,did), foreign key(cid) references emp, foreign key (did) references dept);

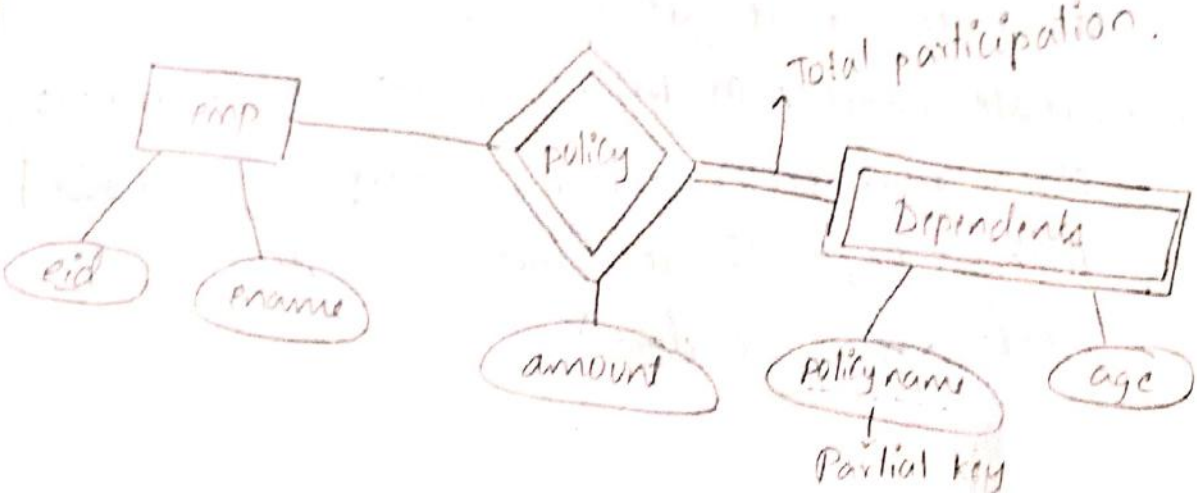
3. Relationship set to tables with key constraints and participation constraints

When we have a ER model in which we have 'n' no. of entities but only 'm' no. of entities participating in the relation, then all these m entities will become candidate keys, then one of them can be chosen as the foreign key for the relationship.

SQL> create table manages (since number(3), primary key (eid), foreign key (eid) references employee);



4. Translating weak entity to tables



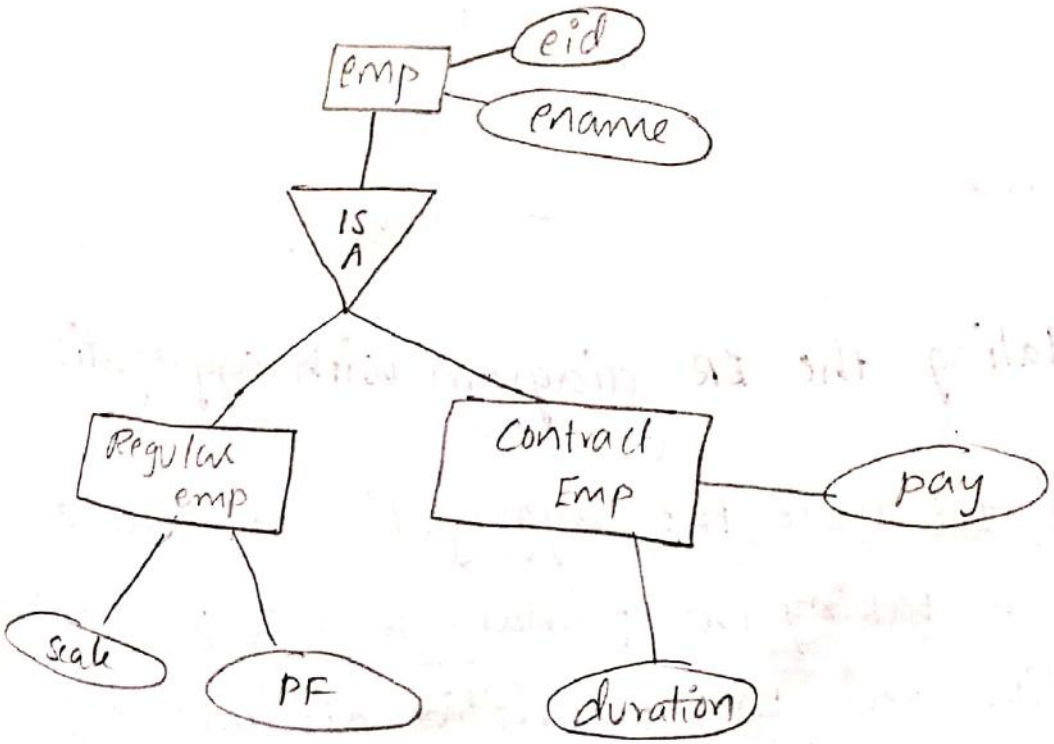
policy Name	age	e-id

→ The weak entity will be taking the help of a strong entities for its existence. The primary key of the strong entity will be taken as an attribute for the weak entity.

→ Entities are declared as foreign keys in this.

SQL> create table depends (policyname char(10), age number(3), eid number(5), primary key (eid, policyname), foreign key(eid) references employee on delete cascade);

5. Translating class Hierarchies:



→ when we have hierarchy in the E-R model, we have two approaches to convert into relational model.

→ In the first approach each of the entity set is translated into a table. The sub entities will have its own attributes and the key attribute of super entity which is declared as a foreign key.

→ In the second approach we can create the tables only for the sub entities. This will include attributes of super entity as well as their own entities.

Contract Emp table will have

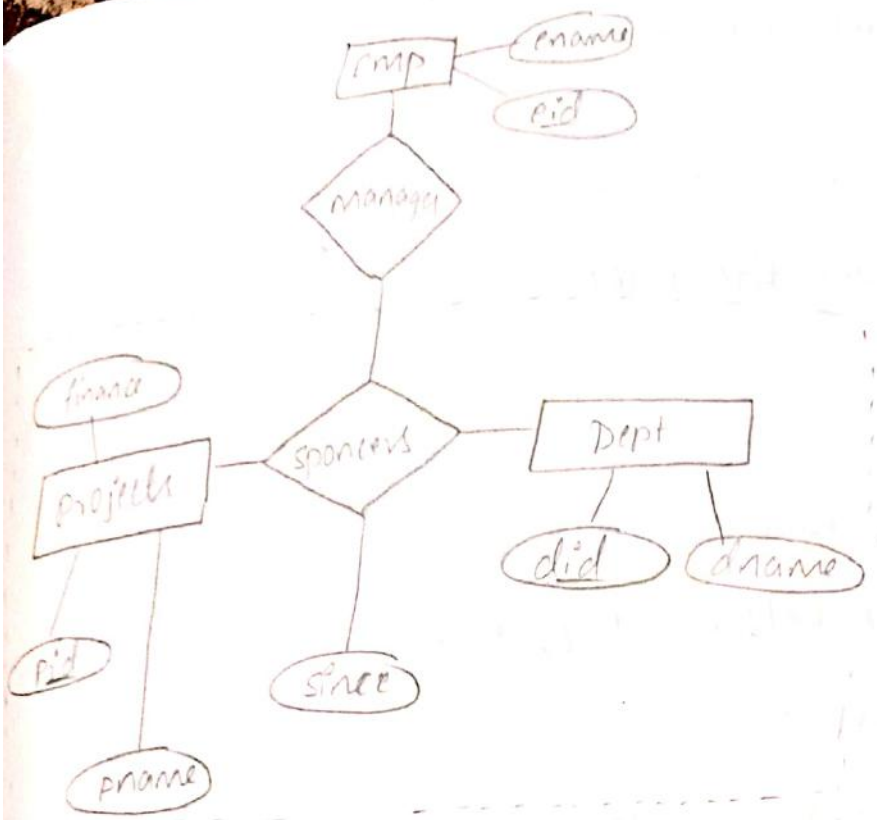
- Pay
- Duration
- cid
- ename

Regular Emp table will have

- Scale
- PF
- eid
- ename

6. Translating the ER diagram with aggregation.

→ when we have the aggregation relations we will be taking the primary keys of participating entities and the descriptive attributes any



→ For the manages relation, one entity is emp, the other entity is sponsers table, so the attributes of manages table will be (e-id)(pid, did) declared as foreign keys.

31/2/20

RELATIONAL ALGEBRA :

→ The database language can be either procedural or non-procedural language.

→ The procedural language will specify to follow a procedure for retrieving data.

→ Relational algebra is an example for procedural language.

→ PL/SQL is another example of procedural language.

→ In non procedural database language we don't mention the procedure to be followed

Ex:

* SQL
* TRC (Tuple Relational Calculus)
* DRC (Domain Relational calculus).

Relational Algebra :-

The fundamental operations under this are,

* selection - σ
* Projection - π
* union - $\cup$
* Intersect - $\cap$
* set difference - -
* Cross product - X
* Rename - ρ
* Joins - $\bowtie$

Tables :

sailors instance S1

sid	sname	rating	age
22			
31			
58			

S2

sid	sname	rating	age
28			
31			
44			

Reserves Instance R1

sid	Bid	day
22	101	
58	103	

a) selection operation : used to select set of tuples from a relation.

Syntax : σ condition (Tablename);

Ex : σ sid = 31 (S1);

Output :

sid	sname	rating	age
31		8	35

b) Projection operation : This operator will allow to select set of columns from a relation.

Syntax : π columnname (Tablename)

Ex : π sid (S2)

Output :

sid
28
31
44
58

when we want to check more than one condition we can use logical operators to combine the condition

We can write relational algebra expressions by combining a selection with projection operation.

Ex: $\pi_{sid, rating}(\sigma_{rating \geq 8}(S2))$;

Output:

sid	rating
28	9
58	10

✱ SET OPERATIONS:

a) UNION (U): This operator will return all the tuples present in either the relation r or s or both.

When we perform union operation both the relations should be union compatible.

It is represented as R U S

Ex: S1 U S2

Output:

sid	sname	rating	age
22		7	45
31		8	55
58		10	35
28		9	35
44		5	35

Ex2: S1 U R1

We can't perform S1 U R1 as the two relations are not union compatible.

b) Intersection ($\cap$) : This operation will return all the tuples that occur in the relation R & S.

R, S must be union compatible.

It is represented as $R \cap S$.

Ex : $S1 \cap S2$

output :

sid	sname	rating	age
31		8	55
58		10	35

c) set difference (-) : This operator will return all the tuples that occur in Relation R but not in relation S.

It is represented as R-S.

Ex : S1 - S2

output :

sid	sname	rating	age
22		7	45

d) Cross Product :- The cross product operation will return a relation whose schema will contain all the fields of R followed by all the fields of S.

The ~~Cestaux~~ resultant relation will contain maximum number of rows where

m is no. of rows in R and n is number of rows in S

when relations contain some column in the output relation they are unnamed and they will be referred by the position in which they appear.

It is represented as R X S. The cross product is also called as cartesian product.

Ex: S1 X R1

sid	sname	rating	age	sid	bid	day
22						
22						
31						
31						
58						
58						

e) Rename Operation:

used to avoid naming conflicts during the relational algebra operations on relations.

Syntax

$\rho(R(F), E)$

↳ O/P

The E represents the relational algebra expression and R is the name that is given to the relation after renaming and F represents the list of renaming columns.

Ex: $\rho(R(1 \rightarrow \text{sailor sid}, 5 \rightarrow \text{reserves sid}), S1 \times K1)$

Relational Algebra :-

1) Selection } unary
2) Projection }
3) Union
4) Intersect } Binary
5) Difference }
6) Rename - unary
7) Cartesian product - Binary
8) Joins

JOINS :

Join operation allows us to combine two relations based on common attributes present in a relation.

The join operation is used to retrieve the info. from 2 or more tables.

The join is basically a cross product followed by selection and projection operations.

Types :

1) Conditional join

This operation accepts a condition on a pair of relations. - $R \bowtie_C S$ where,

C Represents a condition.

Ex : $S1 \bowtie_{S1.sid < R1.sid} R1$

Output :

sid	sname	rating	age	sid	bid	day
22						
31						

b) Equi join

This is similar to the conditional join in which the condition consists of equalities. $R \underset{c}{[X]} S$ where c represents equalities

Ex: $S1 \ [X] \cdot R_1$

$S1.sid = R1.sid.$

Output

sid	sname	rating	age	sid	bid	day
22						
58						

c) Natural Join

If we specify the equality operation on all the common fields present in R, S it becomes a natural join operation. In this no need of specifying the condition explicitely. $R \ [X] S$

In this only one copy of the common field will be appearing in the output schema. If no common attributes present

it becomes cartesian product between the relations.

Ex: S1 ⋈ R1

output:

sid	sname	rating	age	bid	day
22					
58					

DIVISION OPERATION:

→ The division operation on two relations A, B can be performed if and only if A has exactly two columns x, y and B has exactly one column which is y and the output of A/B operation will be the set of all x values such that for a value of x, for every value of y in B there exists a tuple (x, y) in relation A.

→ The idea of division operation is to find all the x attribute values in A that are qualified.

→ An x value is qualified if by attaching a y value from B, we obtain a tuple (x, y) present in A.

A

SNO	PNO
S1	P1
S1	P2
S1	P3
S1	P4
S2	P1
S2	P2
S3	P2
S4	P2
S4	P4

B1

PNO
P2

B2

P.NO
P2
P4

B3

P.NO
P1
P2
P4.

* A/B1

x : S1 S2 S3 S4

y in B1 : P2

(S1, P2) = S1

(S2, P2) = S2

(S3, P2) = S3

(S4, P2) = S4

A/B1 = S1, S2, S3, S4

* A/B2

x : S1 S2 S3 S4

y in B2 : P2 P4

(S1, P2✓) (S1, P4✓) = S1

(S2, P2✓) (S2, P4✗) = NO O/P

(S3, P2✓) (S3, P4) = NO O/P

(S4, P2) (S4, P4) = S4

$A/B_2 = S1, S4$

* A/B_3

x : S1 S2 S3 S4

y in B3 : P1 P2 P4

(S1, P1) (S1, P2) (S1, P4) = S1

(S2, P1) (S2, P2) (S2, P4) = NO O/P

(S3, P1) (S3, P2) (S3, P4) = NO O/P

(S4, P1) (S4, P2) (S4, P4) = NO O/P.

$A/B_3 = S1$

4/2/20

QUERIES USING RELATIONAL ALGEBRA :

1) Find the names of sailors who have reserved boat no. 103

$$\pi_{sname}\left(\left(\sigma_{bid=103}\ Reserves\right) \bowtie sailors\right)$$

First we have to perform a selection operation with bid 103 on reserves table. It gives a temporary table consisting of 3 rows This

temporary table has to be now joined with Sailors table on sid.
After the join apply projection on the name attribute

$\sigma_{bid=103}$ Reserves

sid	bid	day
	103	
	103	
	103	

$\sigma_{bid=103}$ Reserves ⋈ Sailors

sid	sname	rating	age	bid	day
22					
31					
74					

2. Find names of sailors who have reserved a red boat

$\pi_{sname}(((\sigma_{colour=red}\ Boats) \bowtie Reserves) \bowtie Sailors)$

Bid	Bname	colour
102		Red
104		Red

Bid	Bname	colour	sid	day
102		Red	22	
104		Red	22	
102		Red	31	
104		Red	31	
102		Red	64	

Bid	Bname	colour	sid	day	sname	rating	age

RELATIONAL CALCULUS :-

→ The relational calculus is a non procedural database language based on mathematical predicate calculus.

→ The two variations of relational calculus are TRC, DRC.

a) Tuple Relational Calculus (TRC) :-

→ In this a variable represents the tuple of a relation.

→ It takes the values of the tuples of a relation.

→ A query in TRC is represented as.

$\{T \mid P(T)\}$ where,

o/p variable, P → predicate

T - Represents a tuple

P - conditions on which T is true
can be a condition or formula.

→ The TRC will be build on a no. of formulae and each formula may be an atomic expression.

* R(t) is an atomic formula where R is the relation name and t is tuple of R.

* An atom can be in the form of

$t_i.a \; op \; t_i.b$

operator.

The operator can be $<, >, =, !=$

→ A formula in TRC is made up of one or more atomic expressions connected with logical operations $\wedge, \vee, \neg$

→ If F1 and F2 are formulae then, $F1 \vee F2$, $F1 \wedge F2$, $\neg F1$, $\neg F2$ are also formulae.

→ $F1 \rightarrow F2$ is also a formula which means F2 is true whenever F1 is true.

Existential & Universal Quantifiers :-

→ Two special symbols $\exists$, $\forall$ are used in TRC queries.

→ A tuple variable T is a binded variable if it is quantified by $\exists$ and $\forall$. Otherwise T is called as free variable.

$$\exists T \in R(P(T))$$
$$\forall T \in R(P(T))$$

Queries using TRC :-

1, Find the names and ages of sailors who have rating above 7.

$\{P \mid \exists s \in sailors(s.rating > 7 \wedge P.name = s.name \wedge P.age = s.age)\}$.

- The P is the output tuple variable consisting of two columns name, age.
- S is the binded tuple variable belonging to sailors.
- After this, the conditions to be verified are mentioned.

2. Find the sailors name, boat id and reservation day for each of the reservation.

$\{P \mid \exists S \in sailors, \exists R \in Reserves (R.sid = S.sid \wedge P.name = S.name \wedge P.bid = R.bid \wedge P.date = R.day)\}$

3. Find the names of sailors who have reserved boat number 103.

$\{P \mid \exists S \in sailors, \exists R \in Reserves (R.bid = 103 \wedge R.sid = S.sid \wedge P.name = S.sname)\}$

4. Find the names of sailors who have reserved red colour boat.

$\{P \mid \exists S \in sailors, \exists R \in reserves, \exists B \in boats (B.colour = 'red' \wedge B.bid = R.bid \wedge R.sid = S.sid \wedge P.name = S.sname)\}$

5) Find the names of sailors who have reserved all the boats.

$\{P / \exists S \in Sailors, \forall B \in Boats (\exists R \in Reserves (R.bid = B.bid \wedge R.sid = S.sid \wedge P.name = S.name))\}$

6) Find the sailors name who have reserved all the red colour boats.

$\{P / \exists S \in Sailors, \forall B \in Boats (B.colour = 'red') (\exists R \in Reserves (R.bid = B.bid \wedge R.sid = S.sid \wedge P.name = S.name))\}$

b) Domain Relational calculus (DRC) :-

→ In DRC a variable will take the values from the domain of the attributes.

→ A query in DRC will be

$$\{ \langle x_1, x_2, x_3 \ldots x_n \rangle / P(\langle x_1, x_2, x_3 \ldots x_n \rangle) \}$$

where,

→ x represents the domain of an attribute or a constant value. The P indicates the predicate over domain variables.

→ An atomic formula in DRC will be one of the following

* $x_1, x_2, x_3 \ldots x_n \in R$ - will indicate n no. of

attributes of relation R.

* x operator y where x & y are domain variables and operator is <, >, =, !=, >=,

* x operator constant

→ A formula in DRC is written with a no. of atomic expressions.

* If P1 is a formula then ¬P1 is also a formula

* P1, P2 are formulae P1 ∧ P2, P1 ∨ P2 are also formulae.

→ A domain variable can be quantified with ∃, ∀

Queries:

1, Find all the sailors with a rating of 7.

$\{ \langle A,B,C,D \rangle \mid \exists \langle A,B,C,D \rangle \in \text{sailors} \wedge C=7 \}$

2, Find the names of sailors who have reserved boat no. 103.

$\{ \langle B \rangle \mid \exists \langle A,B,C,D \rangle \in \text{sailors}, \exists \langle P,Q,R \rangle \in \text{Reserves} (Q=103 \wedge A=P) \}$

3, Find the names of sailors who have reserved red boat.

$\{\langle B\rangle \mid \exists \langle A,B,C,D\rangle \in sailors, \exists \langle P,Q,R\rangle \in Reserves, \exists \langle X,Y,Z\rangle \in Boats (Z = 'red' \wedge X = Q \wedge R = A)\}$

5/12/20

TRIGGERS :

→ a trigger is a stored procedure which will be executed automatically by the database when some event occurs.

→ The triggers are defined by DBA over databas

→ The three imp parts of trigger definition are,

EVENT PART :

a event can be either DML or DDL or startup operation, shutdown, generating error messages, login and logout operations etc. which will activate the trigger on the database/relation.

CONDITIONAL PART :

It consists either a condition to be verified or a SQL statement which will be verified by the activated trigger.

ACTION PART :

It consists of a no. of SQL statements that will be executed when the trigger is activated and the condition is satisfied.

Syntax:

Create [or replace] trigger triggername
(Create the trigger before or after an event has occurred)
{ before | AFTER | INSTEAD OF }
(view)
{ INSERT (or) UPDATE (or) DELETE } - Event point
[OF COLUMNS] on tablename
[FOR EACH ROW] ← Row level trigger (each row effected by DML)
when (condition) → conditional part
begin
  SQL statements; → Action part
end;

TYPES OF TRIGGERS:

a) ROW level trigger

In this the trigger will be executed for each of the row that is effected by the DML statements

If no row is effected, no trigger is executed. It can be mentioned with FOR EACH ROW option.

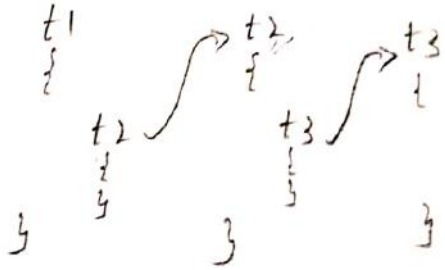
b) Statement level triggers

These triggers will be executed only once for each of the activating statement.

→ A database consisting of triggers is known as active triggers.

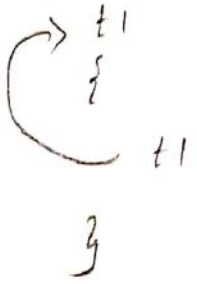
Cascading of triggers / chaining of triggers :-

when one trigger calls another trigger there occurs a chain of triggers. This is called as cascading of triggers.



Recursive triggers :-

when one trigger calls itself again and again it is referred to recursive triggers



17/2/20

UNIT-3

SCHEMA REFINEMENT & NORMALIZATION

SNO	sname	address	CNO	cname
1	x	HYD	001	C
2	v	HYD	001	C
3	y	Pune	002	C++
4	z	Chennai	003	Java
5	xx	Delhi	003	Java
1	x	HYD	003	Java
2	y	HYD	003	Java

→ The conceptual DB design and logical DB design will give us a set of relations with schemas which may satisfy a no. of integrity constraints.

→ The attributes of relations are generallygrouped with the common sense of DB designers.

→ After creating initial design of schemas we have to perform schema refinement in order to avoid redundant storage of data.

→ Schema refinement is the process in which we analyse the existing relations & identify the potential problems and refine the relations.

→ If we don't perform schema refinement it leads to the important problem known as redundant storage of data.

Problems caused by Redundancy :-

1, Redundant storage of data : It means the same info. is stored repeatedly at different places of DB

Anamalies :

These are the problems that occur in a poorly designed database. These occur generally when we store entire DB in one or ~~more~~ two tables.

2, Insertion Anamalies :-

It may not be possible sometimes to store certain information unless some other unrelated information is stored as well.

3, Deletion Anamalies :-

It may not be possible to delete certain information without losing some other information as well.

4, Updation Anamalies :-

If one copy of the repeated data is updated an inconsistency will rise unless all other

copies are updated simultaneously.

5, Wastage of storage space:

It happens due to the entry of null values for the attributes.

6, Generation of spurious tuples (Invalid) during join operations:

There is a chance that invalid tuples may be generated when we join the relations which are poorly designed.

In the above instance, the insertion anamaly occurs when a new student has joined and has not registered for a course. The deletion anamaly occurs when a leaves the college and it leads to deletion of course details also. [Ex: sid with 3]. The updation anamaly occurs when the address of a student is changed in one row only, and the updation is not done automatically in other rows.

To avoid all these problems we will refine the schemas by doing decomposition of relations.

Decomposition :

→ Decomposition of Relation R is the process of replacing single relation with two or more small relations, such that each individual relation contains subset of attributes of R. and together will have all the attributes of R.

→ The decomposition process is based on the normalization concept which works on functional dependencies.

Functional Dependency :-

→ A functional dependency on a relation R is basically a constraint denoted as $X \rightarrow Y$ between the two sets of attributes X, Y such that, for any two tuples t_1, t_2 in relation R if we have $t_1[X] = t_2[X]$ then we must have $t_1[Y] = t_2[Y]$

→ The L.H.S of FD is the determinant and R.H.S of FD is the dependent.

→ We read the FD $X \rightarrow Y$ as, given a value for x, we are able to search the value of attribute Y.

→ For a given schema R we are able to write a no. of FDs.

Ex 1 :-

R1

A	B	C	D
a_1	b_1	c_1	d_1
a_1	b_1	c_1	d_2
a_1	b_2	c_2	d_1
a_2	b_2	c_3	d_1

$AB \rightarrow C$ Holds on R_1

$B \rightarrow C$ Does not hold

$A \rightarrow C$ "

$AC \rightarrow D$ "

Ex 2 :

R_2

A	B	C	D	E
a	2	3	4	5
2	a	3	4	5
a	2	3	6	5
a	2	3	6	6

$A \rightarrow BC$ Holds on R_2

$DE \rightarrow C$ "

$C \rightarrow DE$ Does not hold

$BC \rightarrow A$ Holds on R_2

Ex 3 : R (Acc-no, branch, balance)

The possible FDs can be,

Acc-no → branch

Acc-no → balance

Ex 4 :

R3

A	B	C	D
a_1	b_1	c_1	d_1
a_1	b_2	c_1	d_2
a_2	b_2	c_2	d_2
a_2	b_2	c_2	d_3
a_3	b_3	c_2	d_4

$A \rightarrow C$ Holds on R_3

$AB \rightarrow D$ Does not

$C \rightarrow A$ Does not

$A \rightarrow B$ Does not

$C \rightarrow D$ "

Amstrong's Axioms for finding F.Ds from given FDs (Inference rules)

→ The Amstrong's axioms can be used to derive a no. of additional F.Ds from the given set of F.Ds.

→ If F is the given set of FDs the closure of F is denoted by F^+ which consists the set of all FDs logically implied from F.

→ we can apply the inference rules repeatedly to derive F^+.

→ We use the notation $F \models X \rightarrow Y$ to represent that a new FD $X \rightarrow Y$ is derived with set of FDs F.

The various inference rules are,

1, Reflexive rule :-

if $x \supseteq y$ then $x \rightarrow y$

2, Augumentation rule :-

$\{x \rightarrow y\} \models xz \rightarrow yz$

3, Transitive rule :-

$\{x \rightarrow y, y \rightarrow z\} \models x \rightarrow z$

4. Decomposition Rule :-

$\{x \to yz\} \vdash x \to y$
$x \to z$

5. Union rule :-

$\{x \to y, x \to z\} \vdash x \to yz$

6. Pseudo transitive rule :-

$\{x \to y, wy \to z\} \vdash \{xw \to z\}$

TYPES OF FDs :-

1. Trival FD

A FD $X \to Y$ holds on a relation R where Y is a subset of X then such FD is called trival FD.

$X \to Y, \; y \subseteq x$

Ex : eid, projectno → projectno
eid, ename → ename

2. Non trival FD

A FD $X \to Y$ holds on a relation R where Y is not a subset of X then such FD is called non trival FD.

$X \to Y, \; y \not\subseteq x$

Ex : eid → project-no
eid → ename
eid, projectno → duration
roll-no → branch
name

3. Fully Functional FD

A given FD $X \to Y$ is fully functional if the removal of any attribute from x means that dependency does not hold.

Ex: eid, proj-no → no. of hours persons

This will be FFD if eid → hrs | projectno → hrs will not hold on relation

4. Partial FD

An FD $X \to Y$ in which the attribute Y can be searched with the help of the subset of X. Such FD is partial FD.

Ex: eid, ename → ename

But e-name can be identified only with eid.

Closure of Attributes :-

Ex: R (A, B, C, D)

FDs $A \to BC$

$B \to CD$

closure of all attributes ?

$A^+ = A$

$= ABC$ [$\because A \to BC$]

$= ABCD$ [$\because B \to CD$]

$B^+ = B$

$= BCD \quad [\because B \to CD]$

$C^+ = C$

$D^+ = D$

Ex 2 $R(A, B, C, D, E)$

FDs: $A \to BC$

$CD \to E$

$B \to D$

$E \to A$

$A^+ = A$

$= ABC$

$= ABCD$

$= ABCDE$

$= ABCDE$

$B^+ = B$

$= BD$

$C^+ = C$

$D^+ = D$

$E^+ = E$

$= EA$

$= EABC$

$= EABCD$

$= ABCDE$

Ex 3: $R(A, B, C, D, E)$

$A \to D$

$D \to B$

$B \to C$

$E \to B$

$A^+ = ADBC$

$= ABCD$

$B^+ = BC$

$C^+ = C$

$D^+ = DBC$

$= BCD$

$E^+ = EBC$

$= BCE$

$BD^+ = BD$

$= BCD$

$AE^+ = AE = ABCDE$

$D \; B$

Gate

Ex 4 : $R(A, B, C, D, E, F, G, H)$

$A \rightarrow BC$

$CD \rightarrow E$

$E \rightarrow C$

$D \rightarrow AEH$

$ABH \rightarrow BD$

$DH \rightarrow BC$

Check $BCD \rightarrow H$ Holds or not.

Find L.H.S closure if R.H.S is subset of $[L.H.S]^+$ then it holds on R.

$BCD^+ = BCD$

$= BCDE$

$= ABCDEH$

$= ABCDEH$.

H is subset of BCD^+ Hence, $BCD \rightarrow H$ holds on a relation.

Gate

Ex 5 : $R(A, B, C, D, E)$

$A \rightarrow B$

$A \rightarrow C$

$CD \rightarrow E$

$B \rightarrow D$

$E \rightarrow A$

Find out which of the following FDs are not implied by the above set.

a) $CD \rightarrow AC$

b) $BD \rightarrow CD$

c) $BC \rightarrow CD$

d) $AC \rightarrow BC$

a) $CD^+ = CDEAB$

$= ABCDE$

b) $BD^+ = \underline{BD}$ CD is not subset of BD^+

c) $BC^+ = BCDEA$

$= ABCDE$

d) $AC^+ = ACBDE$

$= ABCDE$.

→ The closure of an attribute is used to determine the set of other attributes which can be identified from this attribute.

→ The main purpose of closure of attributes is to check given FDs hold on relation or not and to determine candidate keys of relation.

Steps to find Attribute closure :

step-1 : Let s be the set of FDs of Relation R.

step-2 : Let x be the set of attributes that appear on L.H.S of any FD. Inorder to determine the set of all attributes that are dependent on x for each such set of attributes determine x^+ based on s.

step3 : For each FD $x \rightarrow y$ if x is subset of result set then result set is equal to result set ∪ y where the initial value of the result set is x.

8th

To determine whether given FD holds or not :-

→ Find the closure of given FD and if R.H.S of FD is subset of closure then the FD holds on relation R.

19/2/20

Finding the candidate keys from given FDs:

In order to determine candidate keys,

→ Take L.H.S of each FDs and find their closures

→ If the closure determines the complete relation schema 'R' then L.H.S of FD becomes one of the candidate keys.

* R(A, B, C, D, E)

$A \rightarrow BC$

$CD \rightarrow E$

$B \rightarrow D$

$E \rightarrow A$

→ $A^+ = ABC$

$ABDC$

$= ABCDE = R$

A is one of the candidate keys.

→ $CD^+ = CDEAB$

$= ABCDE = R$

CD is one of the candidate keys.

→ $B^+ = BD$

$\neq R$

B is not a candidate key.

→ $E^+ = EA$

$= ABCE$

$= ABCDE$

$= R$

E is one of the candidate keys.

Candidate keys = A, CD, E

Prime attributes = {A, C, D, E}

→ The prime attributes are A, C, D and E

→ The key CD may be a super key or candidate key. Inorder to verify that we will take subsets of CD and determine closures of each subset

→ If any of the subset determines complete relation R then that subset becomes candidate key. Otherwise previously identified key will be candidate key.

CD

Subsets :- C and D

$C^+ = C$

$\neq R$

$D = D$

$\neq R$

As any of the subsets of CD is not able to determine the complete relation R, CD becomes

candidate key

→ Along with above 3 candidate keys we may have some other keys for relation. Inorder to determine that

* List out prime attributes.
* check any of the prime attribute is appearing on R.H.S of FDs
* If any FD exists replace prime attribute with L.H.S part of FD in the candidate key.

→ In the above example we can replace prime attribute D with (B → D) with which we will get other candidate key CB.

→ Finally we will obtain 4 candidate keys for the above relation.

R(A, B, C, D, E)

A → C
B → D, A
AB → E

Find the candidate keys.

→ $A^+ = AC$ - Not candidate key.

→ $B^+ = BDA$
$= ABCDE = R$ - Candidate key.

→ $AB^+ = ABE$
$= ABCDE = R$ - (Candidate key).

$A^+ = AC$

$B^+ = BAD$

$= ABCDE$

B subset is able to determine complete relation R.

Hence B is a candidate key & AB is super key.

$\therefore$ Candidate key = B.

* R(A, B, C, D, E, F, G, H)

$AB \rightarrow C$

$BD \rightarrow EF$

$AD \rightarrow G$

$A \rightarrow H$

$\rightarrow AB^+ = ABC$

$= ABCH \neq R$

$\rightarrow BD^+ = BDEF \neq R$

$\rightarrow AD^+ = ADG$

$= ADGH \neq R$

$\rightarrow A^+ = AH \neq R$

As the closure of L.H.S did not determine R Find Out the list of attributes which are not present on R.H.S side of FD.

Here we have A, B, D which are not on R.H.S side of FDs.

→ It means that these 3 attributes will be present in candidate keys that we are going to identify.

→ Find the closure for this set $(ABD)^+$. In most of the cases the closure of attributes not in R.H.S will determine relation R.

$(ABD)^+ = ABD$

$= ABCD$

$= ABCDEF$

$= ABCDEFG$

$= ABCDEFGH$

$= R.$

For the above relation we have only one candidate key.

* $R(A, B, C, D, E, F, H)$

$A \rightarrow B$

$BC \rightarrow D$

$E \rightarrow C$

$D \rightarrow A$

Find the candidate keys.

* $R(A, B, C, D, E, F, G, H)$

$CH \rightarrow G$

$A \rightarrow BC$

$B \rightarrow CFH$

$E \rightarrow A$

Find the candidate keys.

(c)

* $A^+ = AB \neq R$
* $BC^+ = BCD \neq R$
  $= ABCD$
* $E^+ = EC \neq R$
* $D^+ = ABD \neq R$.

Attributes not present on R.H.S = EH

$(EH)^+ = EH$
$= CEH \neq R$.

* $(AEH)^+ = AEHBCD$
  $= ABCDEH = R$
* $(BEH)^+ = BEHCDA$
  $= ABCDEH = R$
* $(CEH)^+ = CEH$
* $(DEH)^+ = DEHAB$
  $= ABCDEH = R$.

∴ candidate keys = AEH, BEH, DEH.

(d)

* $CH^+ = CHG \neq R$
* $A^+ = ABCFHEG \neq R$
* $B^+ = BCFHEG \neq R$
* $E^+ = EABCFHEG \neq R$
* $F^+ = EGABCFH \neq R$.

Attributes not present on R.H.S = D

→ AD^+ = ADBCCFHEG
= ABCDEFGH = R

→ BD^+ = BD
= BCDFH
= BCDEFGH
= ABCDEFGH = R

→ CD^+ = CD ≠ R

→ ED^+ = ED
= AED
= ABCDE
= ABCDEFH
= ABCDEFGH = R

→ FD^+ = FD
= DEFG
= ADEFG
= ABCDEFG.
= ABCDEFGH = R

→ GD^+ = GD ≠ R

→ HD^+ = HD ≠ R

Candidate Keys = AD, BD, ED, FD

20/2/20

Closure Of FDs :-

→ The closure of FDs is the set of all FDs that include F as well as all the FDs that are inferred by applying amstrongs axioms.

→ For a relation R with n attributes there are 2^{2n} no. of possible FDs exists.

→ For a relation with 2 attributes the no. of possible FDs will be $2^{2(2)} = 2^4 = 16$

Ex: $R(A, B, C, G, H, I)$

$F = \{A \to B,\ A \to C,\ CG \to H,\ CG \to I,\ B \to H\}$

Find out F^+.

$\{A \to B, B \to H\} \models \{A \to H\}$ (∵ Transitive rule)

$\{A \to B, A \to C\} \models \{A \to BC\}$ (∵ union rule)

$\{A \to C, CG \to H\} \models \{AG \to H\}$ (∵ Pseduo transitive)

$\{CG \to H, CG \to I\} \models \{CG \to HI\}$ (∵ union rule)

$\{A \to C, CG \to I\} \models \{AG \to I\}$ (∵ Pseduo transitive)

Hir $F^+ = \{A \to B, A \to C, CG \to H, CG \to I, B \to H,$
$A \to H, A \to BC, AG \to H, CG \to HI,$
$AG \to I\}$

$2^{(6)} = 2^{12}$ FDs

*Finding the closure of R without axioms :-

In order to find the closure on to the L.H.S of FD identify the no. of combinations of attributes that we are able to derive and for each of the attribute find the closure. Based on the closure we obtain find the possible closures that we can have on R.H.S As the relation has 2 attributes the possible combinations on L.H.S is

Minimal Cover (or) Canonical Cover of FDs :-

$F = \{A \to B, AB \to C, D \to AC, D \to E\}$

Step-1 : Have all R.H.S part FD with single attribute.

a) $A \to B$

b) $AB \to C$

c) $D \to A$

d) $D \to C$

e) $D \to E$

Step-2 : Find the redundant FDs.

a) $A \to B$

$A^+ = ABC$ Now, consider $A \to B$ is redundant FD.

A^+ from (b) to (e)

$A^+ = A$

Here, $ABC \neq A$. Hence $A \to B$ is not a redundant FD.

b) $AB \to C$

$AB^+ = ABC$

Consider $AB \to C$ is redundant FD

AB^+ from (a), (c), (d), (e)

$AB^+ = AB$

Here, $ABC \neq AB$. Hence $AB \to C$ is not a redundant FD.

c) $D \rightarrow A$

$D^+ = DABCE$

Consider $D \rightarrow A$ is redundant FD

D^+ from (a), (b), (d), (e)

$D^+ = DCE$

Here, $ABCDE \neq CDE$

$\therefore D \rightarrow A$ is not redundant FD

d) $D \rightarrow C$

$D^+ = DABCE$

Consider $D \rightarrow C$ is redundant FD

D^+ from (a), (b), (c), (e)

$D^+ = DABCE$

Here, $ABCDE = ABCDE$

$\therefore D \rightarrow C$ is redundant FD. So it has to be removed from the list of FDs.

Now, will have

a) $A \rightarrow B$

b) $AB \rightarrow C$

c) $D \rightarrow A$

d) $D \rightarrow E$

d) $D \rightarrow E$

$D^+ = DABCE$

Consider $D \to E$ is redundant FD

D^+ from (a), (b), (c)

$D^+ = ABCD$

Here, $ABCDE \neq ABCD$

$\therefore D \to E$ is not redundant FD.

Steps: Find the redundant attributes on L.H.S of FDs

$AB \to C$

$AB^+ = ABC$

* Consider A is redundant

$B^+ = B$ which means that A is required to determine B.

Hence, A is not redundant.

* Consider B is redundant

$A^+ = ABC$ which means B is not required to determine A and C.

$\therefore$ B is redundant.

The final list of FDs,

$A \to B$
$A \to C$ } $A \to BC$

$D \to A$
$D \to E$ } $D \to AE$

* Find the minimal cover for following FDs

$\{BC \rightarrow ADEF, F \rightarrow DE\}$

Step 1: R.H.S part with single attribute.

a) $BC \rightarrow A$
b) $BC \rightarrow D$
c) $BC \rightarrow E$
d) $BC \rightarrow F$
e) $F \rightarrow D$
f) $F \rightarrow E$

Step 2: Finding redundant FDs.

a) $BC \rightarrow A$

$BC^+ = ABCDEF$

consider $BC \rightarrow A$ is redundant FD

BC^+ from (b) - (f)

$BC^+ = DBCEF$

$ABCDEF \neq BCDEF$

$\therefore BC \rightarrow A$ is not redundant FD.

b) $BC \rightarrow D$

$BC^+ = ABCDEF$

consider $BC \rightarrow D$ is redundant FD

BC^+ from (a), (c) to (f)

$BC^+ = BCAEF$

$= BCAEFD$

$ABCDEF = ABCDEF$

$\therefore BC \to D$ is redundant FD.

Hence, it is removed from the list.

$\Rightarrow$ a) $BC \to A$

b) $BC \to E$

c) $BC \to F$

d) $F \to D$

e) $F \to E$.

b) $BC \to E$

$BC^+ = ABCDEF$

Consider $BC \to E$ is redundant FD

BC^+ from @, © to ⓔ

$BC^+ = AFBCDE$

$= ABCDEF$

$ABCDEF = ABCDEF$

$\therefore BC \to E$ is redundant FD.

Hence, it is removed from the list.

$\Rightarrow$ a) $BC \to A$

b) $BC \to F$

c) $F \to D$

e) $F \to E$

b) $BC \to F$

$BC^+ = ABCDEF$

Consider $BC \to F$ is redundant FD

$BC^+ = BCA$

$ABCDEF \neq ABC$

$\therefore BC \rightarrow F$ is not redundant FD.

c) $F \rightarrow D$

$F^+ = FDE$

consider $F \rightarrow D$ is redundant FD

F^+ from ⓐ, ⓑ, ⓔ

$F^+ = FE$

$FDE \neq FE$

$\therefore F \rightarrow D$ is not redundant FD.

d) $F \rightarrow E$

$F^+ = FDE$

consider $F \rightarrow E$ is redundant FD

F^+ from ⓐ, ⓑ, ⓒ

$F^+ = FD$

$FDE \neq FD$

$\therefore F \rightarrow D$ is not redundant FD.

Step 3: Redundant attributes of L.H.S.

* $BC \rightarrow A$

$BC^+ = ABCDEF$

consider B is redundant.

$C^+ = C$

consider C is redundant

$B^+ = B$

* $BC \rightarrow F$

$BC^+ = BCFDEA$

consider B is redundant

$B\ C^+ = C$

consider C is redundant

$B^+ = B$.

The final list of FDs

$BC \rightarrow A$
$BC \rightarrow F$ } $BC \rightarrow AF$

$F \rightarrow D$
$F \rightarrow E$ } $F \rightarrow DE$

* Find the minimal cover of FDs.

$(ABCD \rightarrow E,\ E \rightarrow D,\ AC \rightarrow B,\ A \rightarrow B)$

step1: a) $ABCD \rightarrow E$

b) $E \rightarrow D$

c) $AC \rightarrow D$

d) $A \rightarrow B$

step2: Finding redundant FDs

a) $ABCD \rightarrow E$

$(ABCD)^+ = ABCDE$

consider $(ABCD)^+$ is redundant FD

$(ABCD)^+$ from (b), (c), (d)

$(ABCD)^+ = ABCD$

$ABCDE \neq ABCD$

$\therefore$ $ABCD \rightarrow E$ is not redundant FD

b) $E \rightarrow D$

$E^+ = ED$

Consider $E \rightarrow D$ is redundant FD

E^+ from (a), (c), (d)

$E^+ = E$

$ED \neq E$

$\therefore$ $E \rightarrow D$ is not redundant FD.

c) $AC \rightarrow D$

$AC^+ = ACBDE$

$= ABCDE$

Consider $AC \rightarrow D$ is redundant FD

Ac^+ from (a), (b), (d)

$AC^+ = ACB$

$ABCDE \neq ABC$

$\therefore$ $AC \rightarrow D$ is not redundant FD.

d) $A \rightarrow B$

$A^+ = AB$

Consider $A \rightarrow B$ is redundant FD.

A^+ from (a), (b), (c)

$A^+ = A$

$AB \neq A$

$\therefore$ $A \rightarrow B$ is not redundant FD.

step 3 : Redundant attributes of L.H.S

$AC \rightarrow D$

$AC^+ = ACDB = ABCD$

consider A is redundant

$C^+ = C$

$\therefore$ A is not redundant

consider C is redundant

$A^+ = AB$

$\therefore$ C is not redundant

PROPERTIES OF DECOMPOSITION :-

(1, Depende) The properties of decomposition are used inorder to check whether the decomposition will actually eliminate the redundancy or will it introduce any new problems.

→ when we perform decomposition over relation K we have to check whether the sub relations will cover the original FDs or not.

→ The two properties of decomposition are,

i) Dependency preserving property

A decomposition operation $D = \{R_1, R_2, R_3\}$ is a dependency preserving with respect to F if the union of projections of F on each of sub relation R_i is equivalent to F. i.e

If $(F1 \cup F2 \cup F3 \ldots)^+ = F^+$

Ex : R(A, B, C, D, E)

The above relation decomposed into $R_1(A, B, C)$ $R_2(C, D, E)$ check whether it is satisfying the above property or not.

$AB \rightarrow C$ (obtained from R_1)
$C \rightarrow D$ (obtained from R_2)
$D \rightarrow E$ (obtained from R_2)

∴ It is dependency preserving.

Suppose,

$R_1(A\ B\ C\ D)$

$R_2(C\ E)$

then, $AB \rightarrow C$ (R_1)

$C \rightarrow D$ (R_1)

$D \rightarrow E$ (cannot be obtained either for R_1 or R_2)

∴ It is not dependency preserving.

ii) Loss less Join property (Non additive join property)

→ This property ensures that no extra tuples are generated when a natural join is applied on sub relations.

→ we should recover original relation tuples from the decomposed tuples. If any extra tuples are generated we call it as a Lossy decomposition.

Ex :-

S	P	D
S1	P1	d_1
S2	P2	d_2
S3	P1	d_3

Decomposed into ⇒

S	P
S1	P1
S2	P2
S3	P1

$\Pi_{SP}(A)$

P	D
P1	d_1
P2	d_2
P1	d_3

$\Pi_{Pd}(A)$

S d_1 P d S d P

S1 d_1 P1 d_1 S1 d_1 P1

S2 d_2 P2 d_2 S2 d_2 P2

S3 d_3 P1 d_3 S3 d_3 P3

$\Pi_{SP}(r) \bowtie \Pi_{Pd}(r)$

S	P	d
S1	P1	d_1
S1	P1	d_3
S2	P2	d_2
S3	P1	d_1
S3	P1	d_3

→ Extra tuples

the above decomposition is Lossy decomposition as the join of two relations leads to 2 extra tuples.

S	D
S1	d_1
S2	d_2
S3	d_3

$\Pi_{SD}(r)$

P	D
P1	d_1
P2	d_2
P1	d_3

$\Pi_{Pd}(r)$

$\Pi_{SD}(r) \bowtie \Pi_{PD}(r)$

S	D	P
S1	d_1	P1
S2	d_2	P2
S3	d_3	P1

The above decomposition is not Lossy decomposition.

When the instance is not given we can check whether the decomposition is loss less or not by using 3 conditions If 3 conditions are satisfied then it is loss less decomposition.

i) Union of sub relation must contain all attributes present in original relation

$$R_1 \cup R_2 = R$$

ii) Intersection of sub relations must not be null

$$R_1 \cap R_2 \neq \phi$$

iii) The intersection attribute of relation must be super key of either of the sub relations

* R(A B C D)

$A \rightarrow B$
$B \rightarrow C$
$C \rightarrow D$
$D \rightarrow B$

$R_1(A\ B)$, $R_2(B\ C)$, $R_3(B, D)$

$R_1(A, B, C)$, $R_2(B, D)$

i) $R_1 \cup R_2 = (A, B, C, D)$

ii) $R_1 \cap R_2 = B$

iii) $B^+ = BCD$

B is a super key of R_2.

$R_1(AB)$ $R_2(BC)$

* i) $R_1 \cup R_2 = ABC$

ii) $R_1 \cap R_2 = B$

iii) $B^+ = BCD$

B is a super key of R2.

24/2/20

** NORMALIZATION :-

→ It is the process which will remove all the anamalies and will bring the database into a consistent state.

→ The normalization basically ensures that the information is not stored repeatedly in the DB.

→ It basically decomposes the given relation by a no. of sub relations with the help of functional dependencies.

→ It also try to satisfy the properties of decomposition.

→ It is proposed by "Codd" and a no. of NF are available which will specify certain conditions to be satisfied by the DB relations.

Normal Forms :-

* First Normal Form (1 NF)
* Second Normal Form (2 NF)
* Third Normal Form (3 NF)
* Boyce Codd Normal Form (BCNF)
* Fourth Normal Form (4 NF)
* Fifth Normal Form (5 NF)

FIRST NORMAL FORM (1 NF) :-

→ Every attribute of the relation should contain either a single value or null value.

Ex:

sid	sname	Phone.no
1	x	7760, 1287, 45910
2	y	4030, 9539

The above instance is not in 1NF as the multivalued attribute phone.no consists of more than one value.

The relation can be now converted into 1NF in the following manner.

Method-1 :

Write a seperate tuple for each value of the multivalued attribute.

sid	sname	phone.no
1	x	7760
1	x	1287
1	x	45970
2	y	4030
2	y	9539

method-2 :

Remove the attribute from the relation which violates INF and place it in a new table along with primary key of the original table.

The primary key of the new table will be combination of original primary key and multi valued attribute.

R1

sid	phone.NO
1	7760
1	1287
1	45970
2	4030
2	9539

R2

sid	sname
1	x
2	y

Ex1 :

DNO	DName	Magr.NO	DLocation
01			
02	CSE	10	HYD, DSNR
03	ECE	20	HYD, PUNE
	MECH	30	MEDAK

Converted into INF :-

DNO	DNAME	Mgr. NO	Dlocation
D1	CSE	10	HYD
D1	CSE	10	BANK
D2	ECE	20	HYD
D2	ECE	20	PUNE
D3	MECH	30	METAL

R1 (DNO, DName)
R2 (DNO, Dlocation)
R3 (DNO, Mgr. NO)

SECOND NORMAL FORM (2NF) :-

For a relation 'R' to be in the 2NF the condition

i) It should be in INF

ii) NO partial dependency exists in relation 'R'

Partial Dependency :- (Part of CK → non prime)

Partial dependency in the FD is the one in which a portion of the candidate key determines the non-prime attributes

→ If A → B is a partial FD if 'A' is subset of candidate key 'B' is non-prime attribute of relation R [OR]

→ A relation R is in 2NF if every non prime attribute is fully functionally dependent

or the complete primary key attributes

Steps for converting into 2NF :-

→ Take each of the non-prime attribute check that it is dependent on a part of the candidate key.

→ If it is dependent move the non-prime attribute along with part of candidate key attribute into a new relation and make the key as part of primary key attribute.

Ex) Schema :-

sid	proj-id	sname	proj-name	sponcered amount

sid, proj-id → sponcered amount - satisfies 2NF

sid → sname
proj-id → proj-name } Not satisfying 2NF

sid.	sname

R1

proj-id	proj-name

R2

25/12/2020

THIRD NORMAL FORM (3NF) :-

Non - Trival
Transitive

→ A relation 'R' is said to be in 3NF if,

i) It is in 2NF
ii) No transitive dependency exists for non-prime attributes.

Transitive dependency : (Non-prime → Non prime

A FD $A \rightarrow B$ is a transitive dependency if A is not the super key and B is a non-prime attribute.

Definition :- (or)

→ A relation 'R' is in 3NF for a non-~~prime~~ Trivial FD $X \rightarrow Y$ holds on relation R either X is a super key or Y is a prime attribute of R.

→ A relation in 3NF will not have partial and transitive dependencies on non-prime attributes

Steps for converting into 3NF :-

→ Place the dependent attribute together with the copy of non-prime attribute into a new relation.
→ Make the L.H.S side of the attribute as the key in new table.
→ Put a copy of this attribute in original relation and make it as foreign key.

Ex1: Student (sid, sname, zipcode, city)

sid → city
sid → zipcode
zipcode → city

The FDs we have are,

sid → city
sid → zipcode
zipcode → city

→ The above relation 'R' is in 2NF as there exists no partial dependencies for non-prime attributes.

→ It is not in 3NF because of transitive dependency of city non-prime attribute on the super key sid with the help of a non-prime attribute zipcode.

zipcode → city.
Non-prime attribute — Non-prime

X → Y
* if Y is prime attribute then R is in 3NF as there exists no partial & transitive dependency

→ This can be converted into 3NF by placing 3rd FD attributes in new relation.

* X → Y if X is a C.key then FD is in 3NF

R1

zipcode	city

R2

F.Key

sid	zipcode	sname

[1] : Books (Bookid, categoryid, categorytype, price)

Bookid → price

Bookid → categoryid

categoryid → categorytype.

categoryid → categorytype

Non-prime Non prime

R1

categoryid	categorytype

R2

P.K	F.K	
Bookid	categoryid	Price

BOYCE CODD NORMAL FORM (BCNF) :-

→ For a relation to be in BCNF

i) It should be in 3NF

ii) There exists an FD X → A which is non trivial then X should be super key of relation 'R'.

(OR)

→ A relation R is in BCNF with a FD X → A if B.

if X → A is a trivial FD (or) X is a super key

Steps for converting into BCNF :-

→ For the FD X → A which violates BCNF, decompose relation into two relations in which one relation consists the attributes X, A and other relation consists attributes R - A

→ In final relation make x as super key

→ If the second relation R-A is still not in BCNF, repeat the same process for decomposition.

Ex1 : R(A,B,C)

$AB \rightarrow C$ ✓ BCNF

$C \rightarrow B$ ✗ BCNF

R1

C	B

R2

A	C

→ Upto 3NF all the decomposed relations will be satisfying both the properties. But, BCNF is not always dependency preserving.

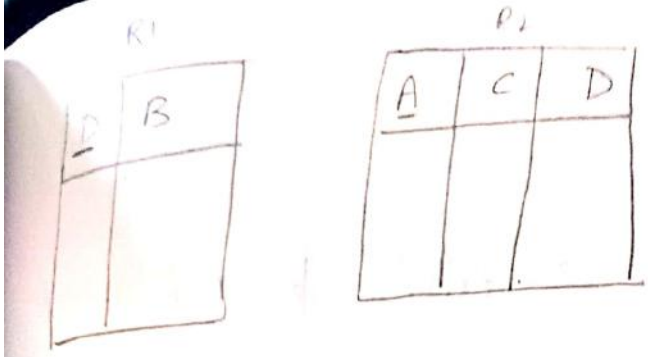
We can achieve loss less join with BCNF but we cannot achieve dependency preserving always in BCNF.

Ex 2 : R(A B C D)

$A \rightarrow BCD$ ✗ BCNF

$BC \rightarrow AD$ ✓ BCNF

$D \rightarrow B$. ✗ BCNF



MULTIVALUED DEPENDENCIES & FOURTH NORMAL FORM :-

→ As the BCNF removes the redundant storage of data in DBs there exists a kind of dependency known as multivalued dependency. which will cause redundant storage of data.

→ The dependency b/w the attributes A, B, C on relation 'R' such that for each value of A there exists set of values of B and a set of values of C. Then the dependency between A, B is MVD represented as $A \rightarrow\rightarrow B$ and an another MVD between A, C i.e $A \rightarrow\rightarrow C$.

Student

name	Hobbies	phno
X	Dance	135
X	Singing	245
Y	Dance	372
Y	singing	487
Z	Art	521

MVDs:

name →→ hobbies

name →→ phno

→ A MVD $X \rightarrow\rightarrow Y$ is trivial if Y is subset of X.

4 NF :-

→ For a relation 'R' to be in 4 NF,

i) It should be in BCNF

ii) There exists no non-trivial MVDs.

→ A relation 'R' is in 4 NF if for every non-trivial MVD $X \rightarrow\rightarrow Y$, X should be the super key of R.

Steps to convert into 4NF :-

Create a new relation for each of the MVD and identify keys for new relation

Ex :- Student instance

R1 (name, hobbies)

R2 (name, phno)

Name	Hobbies

Name	Phno

name, hobbies → name
name, hobbies → hobbies
name, phno → phno.

JOIN DEPENDENCIES & FIFTH NORMAL FORM :

↳ PJNF (Project Join Normal Form)

→ when we decompose a relation R into no. of relations it should satisfy the join dependency.

→ A Join dependency on a relation R which is decomposed into $(R_1, R_2, R_3 \ldots, R_n)$ is said to hold if $(R_1, R_2 \ldots R_n)$ is a non additive decomposition (or) Loss less join decomposition.

5NF :

→ 5NF for a relation R is satisfied when the table is broken into as many tables as possible in order to avoid redundancy.

→ A relation R is in 5NF with respect to closure of FDs, MVDs, JD if for every trivial join dependency every relation R_i is a super key of R.

Problems on Normal Forms :-

* R(A, B, C, D, E)

A → B
B → E
C → D

which normal form 'R' is ?

Key :
→ A, C

Prime Attributes = A, C
NON-Prime = B, D, E.

BCNF

A → B
↓ ↓
NO C.K N.prime

R - Not in BCNF

3NF

A → B ✓ 3NF
↓ ↓
Prime N.prime
Attribute

B → E ✗ 3NF
↓ ↓
N.P N.P

2NF

A → B
↓ ↓
Part of NP
C.key

R - Not in 2NF

∴ R is in 1NF

→ In order to check for 3NF for partial and transitive together we can verify for any of the below conditions for FD.

→ For a FD to be in 3NF ($X \to Y$)

* X should be candidate key

(or)

* Y should be prime attribute.

* R(A B C)

$AB \to C$ check Normal form of 'R'.

$C \to A$

Keys = AB

BCNF

$AB \to C$

↓ ↓

C.K N.P

$C \to A$. P

Not S.K Not P

Not BCNF.

3NF

$AB \to C$.

↓ ↓

C.K. P.P

$C \to A$