DATA BASE MANAGEMENT SYSTEM

Software : Oracle 11g express edition

What is Data Base Management System ?

⇒ collection of inter related data and a set of programs to access these data. It stores data in such a way that it becomes easier to retrieve. It is a general purpose software system that facilitate the process of defining, constructing, manipulating and sharing data bases among various users and applications.

Properties of data base :

1) A single repository of data is maintained i.e. define once and can be accessed by multiple users.
2) Self describing nature of data base system:
Data base system contain not only data but also a complete definition or description of data base.
3) Insulation between programs and data
In traditional file it require all changes while it is program data independent.
4) Sharing of data and multiuser transaction file
It allows sharing of data and multiuser transaction processing.

Advantages :

1) Controlling data Redundancy
2) Elimination of inconsistency
3) Better services to users (Reliability)
4) Flexibility of system is improved.
5) Integrity can be improved.

6) It can provide backup and recovery
7) Standards can be enforced.
8) Security can be Improved.
9) Overall cost of developing and maintaining is low.
10) Concurrency control

Disadvantages:

1) Increased complexity
2) confidentiality, Privacy and security risk
3) Enterprise Vulnerability, Failure of any component can stop working
4) cost of using DBMS is high
   a) because of additional hardware cost
   b) cost of training staff and employing new staff.

Data Models:

Data Model define how the logical structure of data base is created.

Data Model define how data is connected to each other and how they are processed and stored inside the system.

Types of Data Models:

1) ER Model
2) Hierarchal Model
3) Network Model
4) Relational Model

1) ER Model (Entity Relationship Model)

It is based on real world entities and relationship among them. While formulating real world scenario in this model, ER Model creati entity set, relationship set, attributes and constrains.

It is based on :

a) entity and their attributes
b) Relationship among entities

Entity: It is an object in real world with an independent existence.

for eg: Person, student, Teacher

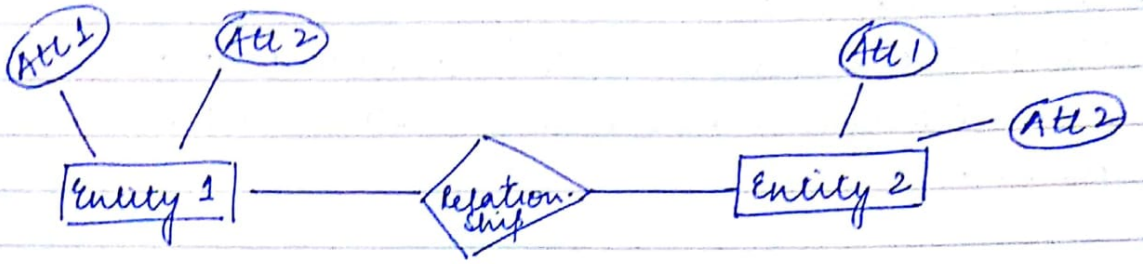
Relationship: Logical association among entities is called Relationship. They are mapped with entities in various ways.

Mapping:

Mapping is no. of association between two entities.

Eg: one to one, one to many, Many to one, Many to Many

Eg: ER Model



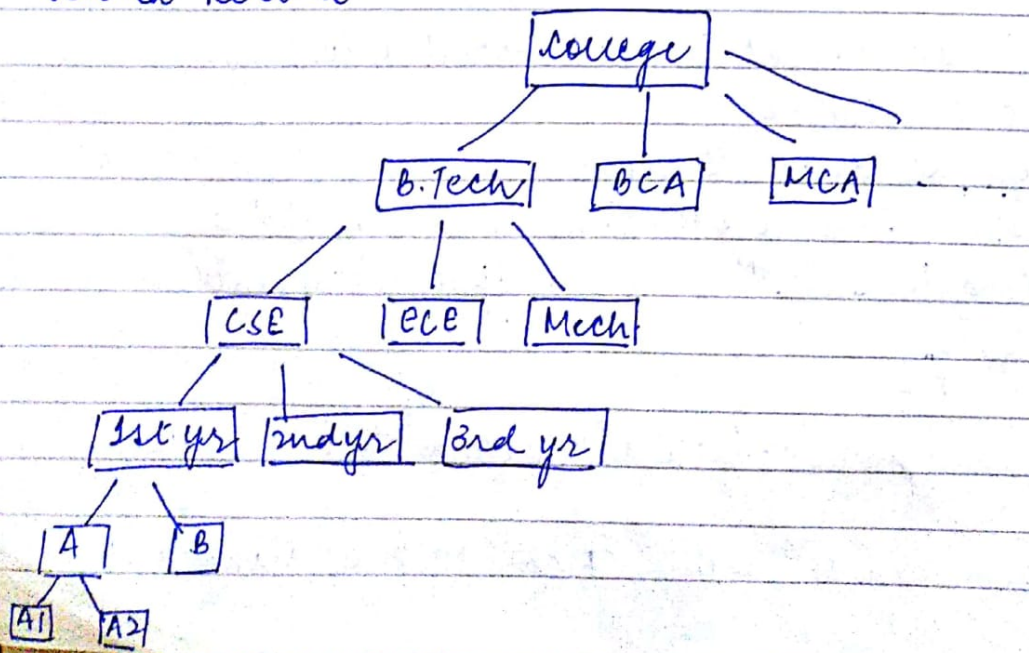
2) Hierarchical Model

This model organize records in a tree structure i.e. Hierarchy of parent and child record relationship. This model employ 2 main concepts:

1) Record
2) Parent child Relationship

Record: It is collection of values that provide information of an entity.

Parent child Relationship: It is one to many relationship between two record. The record is one side ie Parent Record and on other side is called child Record.



3) Network Model:
The ability of this model is to handle many to many relationship between its records. This model permits the child record to have more than one parent. In this model, directed graphs are used instead of tree structure. This model is designed to handle non hierarchical relationships.
Eg: A book data base where an author can have many books and a book could have more than one author.

4) Relational Model:
It consist of components:
a) A set of relations and a set of domains.
b) A set of rules for maintaining the integrity (uniqueness) of data base.

Relation (table)

cardinality is no. of rows in relation.
Degree is no. of columns in relation.

Schema:
Description of data base

Sub Schema:
A subset of schema is called Sub Schema.
It inherits the same properties as that of schema

Instances:
Data in the data base called Instances or database state or snapshot or current set of occurences.

Person: → Schema

	Name	Age	Email	Phone No.
Instances	A	1	-	-
	B	2	-	-
	C	3	-	-

Select * from table name
desc emp
description

exp 2

DDL commands
data definition language

1) create

Syntax:
create table table name(column1, data type(size),
column n data type(size))

create table student(roll no; name varchar(20),
email id varchar(30))

2) Alter

a) Add new column

Alter table tablename add(new column datatype(size))

b) delete a column

Alter table tablename drop column columnname

c) Modifying existing column

Alter table table name modify columnname newdatatype/new size)

3) Rename

a) Rename table :

Rename oldname to newtablename

b) Rename column

Alter table tablename Rename column old columnname to newcolumnname

Table output

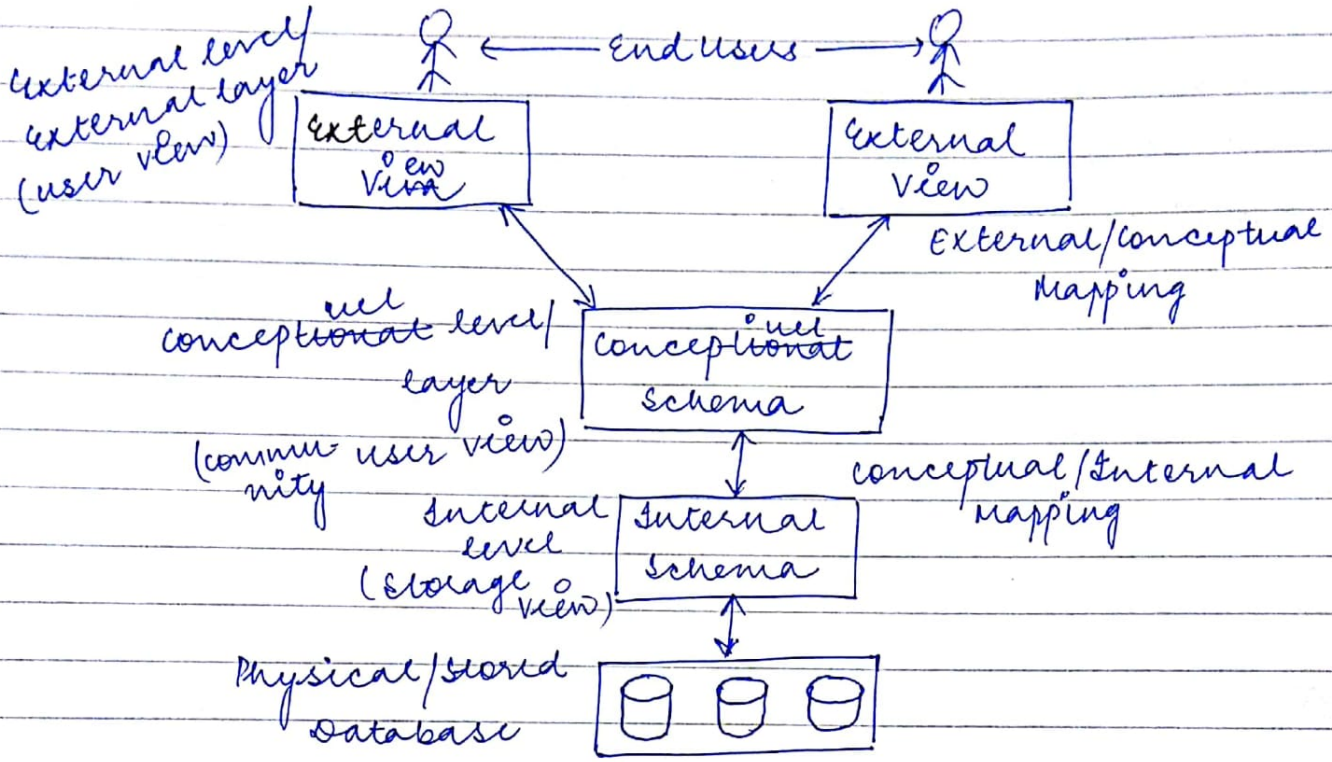
4) Drop

drop table tablename

5) Truncate

Truncate table tablename

Three Schema/level Architecture



The goal of this architecture is to separate the users application and data base. The main aim of designing this architecture is to provide users with an abstract view of data by hiding certain details of how data is stored and maintained.

There are 3 layers:

1) external level
2) conceptual level
3) Internal level

1) external level

It includes no. of external users. Each external schema/view describe the part of database that a particular user is interested in and hide the rest of the data.

For eg: employs in clg library will usually view few details of data such as book issued or book return.

2) Conceptual level / logical level / Middle level
It describe the structure of data base for a community of users. It hide the details of physical structure (data base) and concentrate on describing entities and their relationship and constraints.

3) Internal level
It describe complete details of data storage and access path for database. It includes how attributes are represented, how are records are sequenced.

Mapping between Views:
1) External conceptual Mapping:
Its role is to define communication between end users and community users.
2) Conceptual Internal Mapping:
Its role is to define communication between community user view and storage view.

Data Independence:
It can be defined as capacity to change the schema at one level of database without having changing the schema at next higher level.
There are of two types:
1) Logical Data Independence
2) Physical data Independence

1) Logical Data Independence:
capacity to change the conceptual schema without having change in external schema.

2) Physical Data Independence:
capacity to change the external schema without having change in conceptual schema.

DBMS LAB...

Draw the table:

Emp_id	emp_name	emp_age	emp_location	Department	netsalary	allowance	Gross salary	Contact no.

DML commands

1) Select
2) Insert
3) update
4) Delete
5) Logical operators
6) Relational operators
7) Arithmetic operators
8) LIKE
9) IN
10) Between

1) a) Select all rows and all columns

Syntax:

```
select * from tablename;
```

b) Select specific rows and all columns
Select * from tablename where condition;
c) Select all rows and specific columns
Select column1, column2 from tablename
d) Select specific row and specific column
Select column from tablename where condition
e) Distinct rows
Select distinct * from tablename;
f) Distinct column
Select distinct columnname from tablename;
g) Sorting data
Select * from tablename order by columnname asc/desc.

2) Insert
a) Insert all data.
Insert into tablename values( );
b) Insert into specific ~~(columnname) values( );~~
Insert into tablename(columnname) values( );
c) Insert multiple rows
Insert all
into tablename values( )
into tablename values( )
Select 1 from dual
d) Select specific row and specific column
Select column from tablename where condi

continued

Client / Server Architecture:

There types of client/server architecture:

1) One tier:

In this, the DBMS is only client or server where the user directly sit on the DBMS and uses it.

eg: PC without internet

2) Two tier:

In this, the user interface and application programs can run on client site. The direct communication takes place between client and server. There is no intermediate between client and server. It is divided into two types:

a) Client Tier

b) Data base Tier

On client ~~app~~ site, code is written for saving the data in database. The main problem of two tier architecture is that server cannot respond multiple request at the same time. As a result, it can cause a data integrity issue.

Advantages:

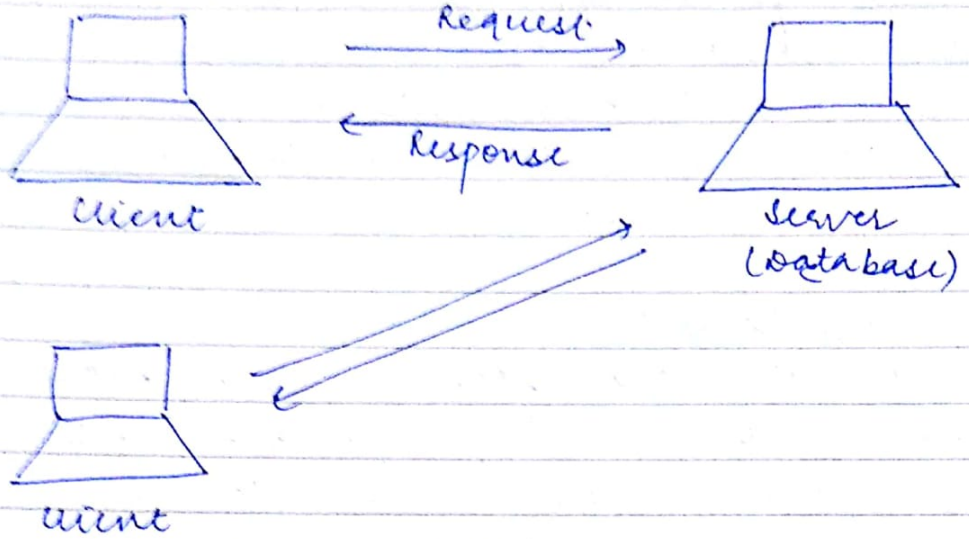
1) Easy to maintain

2) communication is faster

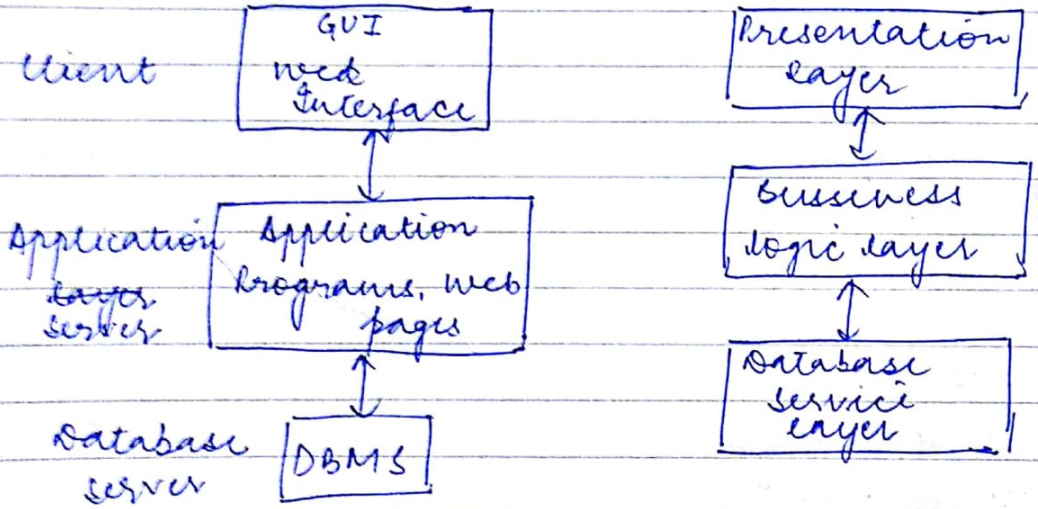
Disadvantages:

1) Performance is poor

2) cost ineffective



3) Three tier :



Presentation layer:
This layer contain GUI part of our application. This layer is used for design purpose where data is presented to the user or input is taken from user.

Business logic layer or Application layer:
In this layer, all bussiness logic written like validation of data, calculation, data insertion, connectivity, etc. This layer act as interface between client and data base server.

Database Service layer:
This layer stores the actual database. The data access layer contains method connect with database and how to perform operations like insertion, updation, deletion to get data from data base based on your input.

Advantages:
1) High Performance
2) Better reuse.
3) Easy to maintain
4) Improve Security
5) Improve data integrity

disadvantages:
1) Does Increase complexity
2) Increase effort

Types of attributes:
1) Singe Value attribute: An attribute that has singe value for a particular entity is known as Singe Value attribute.
Eg: Roll No., Age, Date of Birth

2) Multivalued Attribute : An attribute that have multiple values for an entity is known as multivalued Attribute
Eg. Address, email address, phone no.

3) Compound or composite Attribute : Attribute can be sub-divided into two or more other attributes :
Eg : Name, Address

4) Simple or Atomic Attribute : This attribute which cannot be further sub-divided into other attributes.
Eg : House No., Roll No., Age

5) Stored Attribute : An attribute which cannot be derived from other attribute.
Eg : Date of birth

6) Derived Attribute : which can be derived from other attributes.

7) Key Attribute : This attribute has unique values for each entity. It represents primary key.
Eg : Roll No.

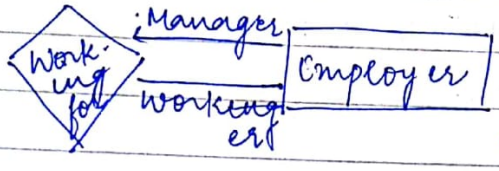
8) Non key Attribute : It is the attribute that doesn't represent ~~nouns~~ main characteristics of the attribute.
Eg : Phone No., Name

9) Required Attribute : It is an attribute that must have a data value. That means this cannot be null.
Eg. Roll No., Name, Address

10) Optional / Null value Attribute: These Attributes may not have a value in it and can be left blank!
Eg: middle name etc.

Types of Relationship:

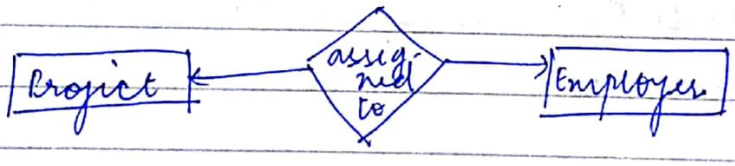
1) Unary Relationship:
When association exists within a single entity type is known as Unary Relationship.



Degree = 1

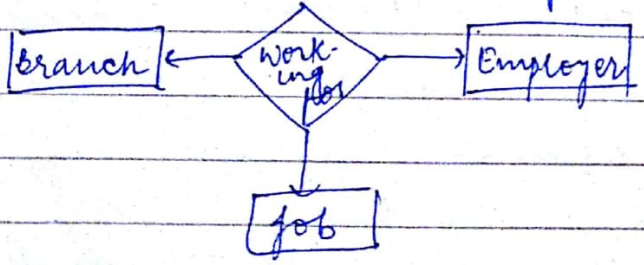
Degree of Relationship: No. of entities associated within the relationship.

2) Binary Relationship:
When association exists between two entities.



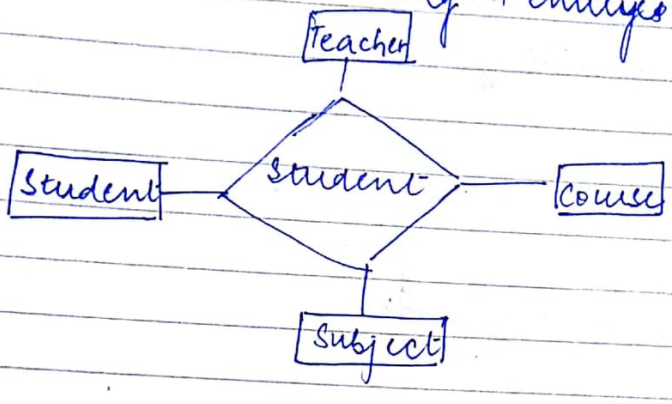
degree: 2

3) Ternary Relationship
When association exists among 3 entity types.



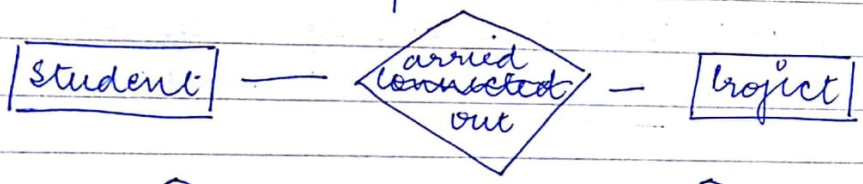
degree = 3

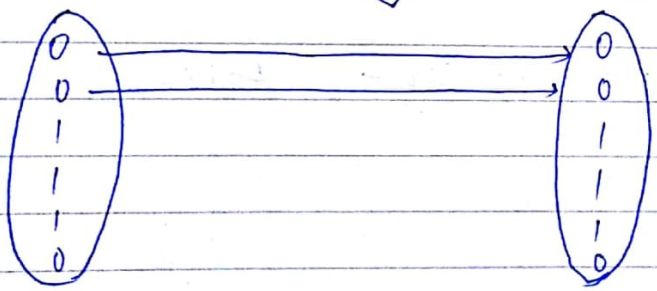
4) quaternary Relationship
when association exists among 4 entities types.



Connectivity of Relationship :

1) One to one
When only one instance of an entity is associated with the relationship



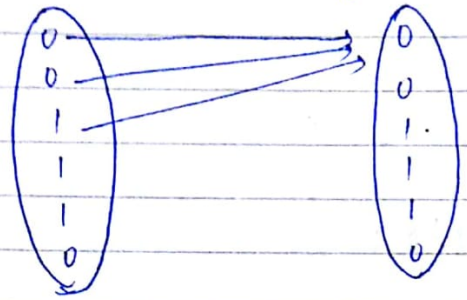


2) One to many
One entity from set A can be associated with more than one entities of set B.

Many to One :

More than one entities from set A can be associated with atmost one entity of set B.

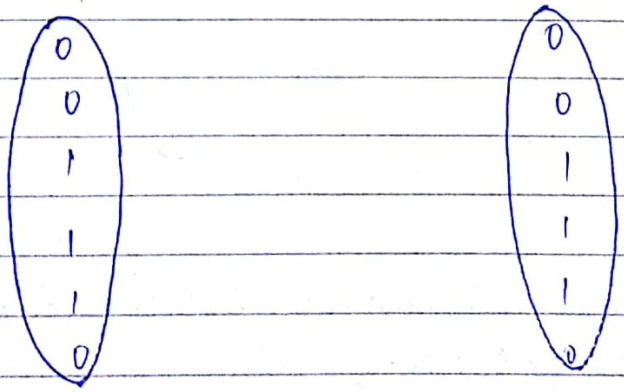
Department — Working — Faculty



Many to Many:

One entity of A set A can be associated with more than one entity from set B and vice versa.

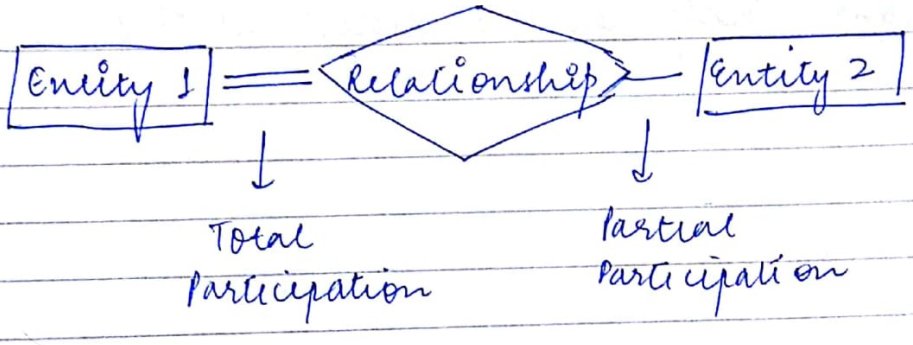
Author — Writer — Book



Participation Constraints:

1) Total Participation:
Each entity is involved in the relationship. It is represented by double lines

2) Partial Participation:
Not all the entities are involved in the relationship. It is represented by single line.



Entity Type:
It is a collection of entities having common attributes.

Entity Set:
It is the set of entities of same entity type.

Student Entity Type

Roll No.	Name	Marks
1	A	120
2	B	10
3	C	15
4	D	20

Entity

Entity Set

Weak Entity Set:
The entity set that doesn't have sufficient attributes to form a primary key and depend on other entity for its identification is called weak entity set.

Strong Entity Set:
An entity that has a primary key is called a strong entity set. It is the one that exists on it own independent of other entities.

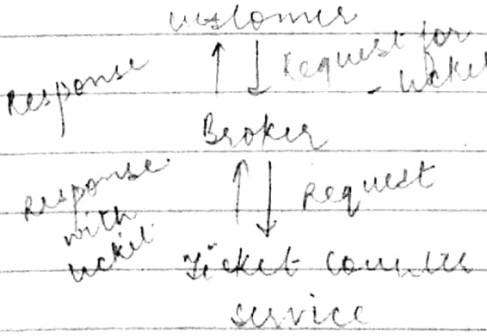
Q. Imagine we have to create a database for a company entities involve in it are company, department, its supplies, employs and projects. If we observe each of the entity, they have parent child relationship. Then draw detail data model which is best suited for this application. Hierarchical model (company, departments, employees, projects)

Q. If Operating system is upgraded to support some new functions on OS file (i.e ability to force some sequence of bytes to disk) which layer of DBMS would you have to rewrite to take the advantage of these new functions. Internal level

Q. What are the components of DBMS (Hardware, Software)

Q. Suppose you want to book ticket with the help of broker which n tier client server architecture will you prefer for given scenario. Explain working of that architecture.

Three tier client server



Requirement and Analysis:
During this step the database designers collect required data and documents according to their requirement.

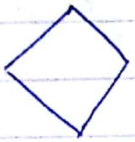
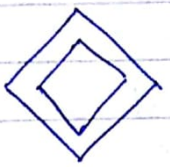
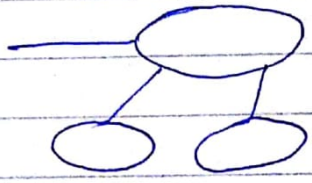
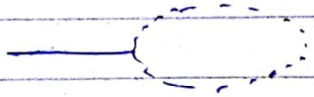
Functional Analysis:
In parallel with specifying the data requirements it is useful to specify the functional requirements of the application. This consist of user defined operations or transactions that will be applied to database.

Conceptual design:
Once all the requirements have been collected and analyzed, the next step is to create conceptual schema for database using high level conceptual data model.

Logical design:
This step is actual implementation of database.

Physical Design:
In this phase, internal storage, indexes, file organization for the database files are specified.
In parallel with these activities, application programs are designed and implemented.

Notations ...

Symbol	Meaning
	entity
	Weak entity
	Relationship
	Identifying Relationship
	Attribute
	key Attribute
	Multivalued Attribute
	composite Attribute
	Derived Attribute
	Total participation of E_2 in R

cardinality ratio
1:N
for $E_1 : E_2$ in R

Keys:
The minimal set of attributes which uniquely help identify the tuple (row) of a relation. These are used to create relationship among different database tables. It is used to fetch or retrieve data from table according to requirement.
It satisfies the following two properties:
- Unique identification
  i.e no duplication of tuples
- Irreducable
  cannot be further reduced

Types of Keys:
1) Candidate key / Minimal super key
It is a set of one or more columns that can identify a record uniquely in a table. There can be multiple candidate keys in one table. Each candidate key can work as a primary key. It is a subset of super key.

Eg:

Roll No.	Name	Age	Email

CK = Roll No.
Email

Primary Key:
It is a set of one or more columns of a table that uniquely identify record in database.

Condition:-
It cannot accept null values or duplicate values.
There can be only one primary key in a table.
If there is only one candidate key, it will be a primary key.

Eg: Student

Roll NO.	Name	Age	Email

Primary key:
Roll No.

fname	lname	Age
Abinav	Verma	10
Abhishek	Sharma	10
Abhishek	Verma	11

Primary key:
(fname, lname)

Super Key:
It is a set of one or more keys that can be used to identify the record in a table.
It is a unique for each and every row. It can also be a single column.
All the keys are subset of super key.

Alternate key / secondary key:
It is a key that can work as a primary key. If any table consist more than one candidate key then after choosing primary key rest of the candidate keys are known as alternate key/ secondary key.

Unique key:
It is like primary key but it can accept only one null value.

Composite key (concatenate key)
A key that consist 2 or more attributes to uniquely identify a row is known as composite key.
eg: Roll No. and Email Id (above example- student)
If there is no single attribute in your relation then concatenate two or more attributes to uniquely identify a record in your database.

Foreign key:
It is generally a primary key from one table that appear as an attribute in another table where first table has relationship
It acts as a cross reference between two tables.
The relation in which foreign key is created is known as dependent or child Table. The relation to which foreign key refer is known as parent table.

foreign key

Primary key

eg:

St_id	fname	C-ID
01	A	1
02	B	2
03	C	1

Relationship

Course ID	course_name
1	DBMS
2	OS

Parent table

Child Table

Primary key

dept ID	dept name
1	CSE
2	ECE
3	Mech

Parent table

Q. Consider a university contains many departments. Each department can offer any no. of courses. Many teachers can work in one dept. A teacher can work only in one department. For each dept there is a head. A teacher can be head of only department. Each teacher can take any no. of courses. A course can be taken by only one teacher. A student can enroll any no. of courses. Each course can have many no. of students. Identify entities, attributes, relationships between entities and draw ER diagram

NAMING CONVENTION AND DESIGN ISSUE OR STEPS IN DESIGNING ER DIAGRAM

1. Identify entities
2. Find relationship among entities.
3. Identify key attributes for each and every entity.
4. Identify other relevant attributes.
5. Draw ER diagram.

E.F Codd Rules

There are 12 rules which are infact 13 in number are developed by Dr Edgar F Codd who is the father of Relational Database Management System. There are 12 rules numbered from 0 to 12 i.e. 13 in numbers.
According to Dr. E.F Codd a RDBMS is fully relational if it follows all these rules.

1) Rule 0: Foundation Rule
Any relational DBMS should be able to manage the stored data entirely through its relational capabilities.

2) Rule 1: Information Rule
Relational database should store the data in the form of relations. Tables are relations in RDBMS.

3) Rule 2: Granted Access
The use of pointers to access data logically is strictly forbidden. Every data entity which is atomic in nature should be accessed by using a right combination of name of table, primary key represented by

a specific row value and column name represented by attribute value.

4) Rule 3: Systematic Treatment of null
Null values are completely supported in relational in databases. They should be considered as missing information. Null values are independent of any data type. But primary key cannot be null.

5) Rule 4: Active online catalogue
In the DBMS, the active online catalogue stores the matter data (the data about data) is called Data dictionary and this data dictionary is accessible only by authorised users who have the required privilages and query languages used for accessing the database.

6) Rule 5: Rule of comprehensive data language.
A single language should be able to define integrity constraints, dues, transactions, manipulations and authorisation.

7) Rule 6: Updating Views
A view is a logical table which shows restricted data; views help in data abstraction. Views make generally data readable but not modifiable.

8) Rule 7: Relational level operations
All the operations insert, update, delete, Select etc., should be supported.

9) Rule 8: Physical Data Independence
End users can change the database schema without having to recreate it or without any change at higher level.

10) Rule 9: Logical Data Independence
If the change in logical structure of database is there, then user view should not change.

11) Rule 10: Integrity Independence
Integrity constraints should be available and stored as meta data in data dictionary.

12) Rule 11: Distribution Independence
The data manipulation language of the relational system should not be concerned about the physical data storage and no alterations are required if the physical data is centralised or distributed.

13) Rule 12: Rule of Non-Subversion
Any row should obey the security and integrity constraints imposed. Locking or encryption should also be supported.

RELATIONAL ALGEBRA:

It is a theoritical or conceptual language with operations that work on one or more relations to define another relation without changing the original relation.

It means that queries created in it are not executed on computer so it is not used as commercial language. Its knowledge allow us to understand optimization of RDBMS and query execution.

Relational Algebra operations:

1) Set operations / Binary operations consist of
   1) Union
   2) Intersection
   3) Difference
   4) Cartesian product

2) Database operations / Special Relational operations consist of
   1) Selection] Unary operations
   2) Projection]
   3) Join
   4) Division

a) Union operation: (R ∪ S)

Relation R

E_ID	E_name
100	Ramesh
102	Akash
104	Mahesh

Relation S

E_ID	E_name
105	Vikram
110	Sushil
100	Ramesh
104	Mahesh

Relation R ∪ S

E_ID	E name
100	Ramesh
102	Akash
104	Mahesh
105	Vikram
110	Sushil

Relations are union compatibles if the degree is same of two relations.

Properties of Union operation:

1) The two relations must be union compatible
   i.e. Domain of Relation 1 should be equal to Domain of Relation 2.

2) Commutative law:

$$R \cup S = S \cup R$$

3) Associative law

$$P \cup (Q \cup R) = (P \cup Q) \cup R$$

b) Intersection Operation ($R \cap S$):

Relation R

E_id	E_name

Properties:
union compatible
Commutative law

$$R \cap S = S \cap R$$

Associative law

$$P \cap (Q \cap S) = (P \cap Q) \cap R$$

c) difference (R-S):

Properties:
1) Union compatible
2) Not commutative

$$R - S \neq S - R$$

3) Not Associative

d) cartesian product (R * S)

Relation R		Relation S	
E_ID	E_name	E_ID	E_name
100	Ramesh	105	Vikram
102	Akash	110	Sushil
104	Mahesh	100	Ramesh

Properties:

1) Input relations need not necessary to be Union compatible.
2) It may hold duplicate attributes.
3) Degree of resultant relation is equal to sum of degree of all input relations
4) Total no. of tuples (rows) = product of no. of rows of relation 1 and Relation 2.
5) commutative law

$$P * Q = Q * P$$

6) Associative

$$P * (Q * S) = (P * Q) * S$$

RELATIONAL ALGEBRA OPERATIONS:

i) Selection: It is also known as restriction equation results in a new relation that contains only those rows of relation that satisfy a specified condition. It is a unary operation which means that it can work on a single relation denoted by 'σ' (sigma)

$$\sigma \text{ condition}(R)$$

Q. Find out all the employees having a salary greater than equal to 8000 from employees table.

⇒ $\sigma_{\text{salary} \geq 8000}(\text{employee})$

Properties:

1) It is a unary operation.
2) Degree is same as resultant table.
3) Rows can be less than or equal to original table.

commutative law

$$\sigma_{<\text{condi1}>}(\sigma_{<\text{condi2}>}(R))$$

ii) Projection: It is denoted by π. It is also a unary operation. It results in a new table that contain a subset of columns of relations and eliminate any duplicate row that may result.

$$\pi_{<attribute1, attribute2 - \ldots>}(R)$$

Q. Find out the name and salary of all employes from employee table.

⇒ $\pi_{<name, salary>}(employee)$

Q. Find out the name of all employes whose salary is ≥ 8000.

⇒ $R_1 \rightarrow \sigma_{salary \geq 8000}$

$R_2 \rightarrow \pi_{name}(R_1)$

Result : $\pi_{name}(\sigma_{salary \geq 8000}(employee))$

Properties:

i) unary equation
ii) Degree = no. of attributes
iii) duplicacy elimination
iv) commutative law does not exist.

iii) Rename

Rename operation can be applied in 3 forms.

a) $P_x(R)$
b) $P_{(A_1, A_2, \ldots)}(R)$
c) $P_{x(A_1, A_2, \ldots)}(R)$

a) This expression returns the result of relation R under the name of X. i.e. Relation R having a new name X

b) Rename attributes

iv) Division

The division of R & S is denoted by $R \div S$

Q. Find out the name of subject i.e taught in all classes.

»

R			S		$R \div S$
Name	Course		Course		Name
System	BCS	$\div$	BCS		Database
Database	BCS		MCS		
Database	MCS				
Database	MCS				

v) Join

It is a method of combination of two or more relations in a single relation. It is denoted by ⋈ and represented by $P \bowtie_{<\text{join condition}>} Q$

Types of join:

i) θ join
ii) Equi join
iii) Natural join
} Inner join

iv) Cross join or Cartesian product
v) Outer join
vi) Left outer join
vii) Right outer join
viii) Full outer join

Consider two databases given below:

Student

id	Fname	Lname	Address
1	Rahul	Sharma	
2	Ajay	Thakur	
3	Mani	Sharma	
4	Rahul	Roy	
5	Nitin	Arora	

Subject allocated

Subject	id	Fname
101	1	Rahul
102	3	Mani
103	6	-
104	3	Mani
105	2	Ajay

- Theta join

It combines tuples from different relations provided they satisfy the theta condition denoted by θ.

Representation:

$$R_1 \bowtie_\theta R_2$$

where R_1 and R_2 are two relations.

It can be used all kinds of comparison operators.

- equi join

When Theta join uses only equality comparison operator, it is said to be equi join and if other comparison operators are used then the join is non equi join.

Student

S. id	Name	Std
101	Alex	10
102	Maria	11
103	Riza	12

Subject

class	subject
10	Math
10	English
11	Music
11	Sport

Student $\bowtie$ student std = subject class subject

S_id	Name	Std	Class	Subject
101	Alex	10	10	Math
101	Alex	10	10	English
102	Maria	11	11	Music
102	Maria	11	11	Sport

It doesnot use any comparison operator. It doesn't concatenate the way a cartesian product does.
We can perform Natural join if there is atleast one common attribute between 2 relation. Attribute must have same name and same domain.
Representation

$R_1 \bowtie R_2$

Courses

C_id	course	Dept
CS01	database	CS
ME01	Mechanical	ME
EE01	Electronics	EE

HOD

Dept	Head
CS	Alex
ME	Maya
EE	Maria

C-id	course	dept	Head
CS01	database	cs	Alex
ME01	Mechanics	ME	Maya
EE01	electronics	EE	Maria

- cross join
  Representation : (R X S)

- outer join
  Inner join includes only those tuples which are having matching attributes and same domain but rest tuples are discarded.
  ∴ We need to use outer join to include all tuples.

- left outer join
  All the tuples from left relation 'R' are included in resulting relation. If there are tuples in R without any matching tuple in the right relation 'S', then S attribute of resulting relation will be null.

course

A	B
100	DB
101	ME
102	EE

HOD

C	D
100	Alex
102	Maria
104	Maya

course ⟕ HOD

A	B	C	D
100	DB	100	Alex
101	ME	-	-
102	EE	102	Maria

- Right outer join (R ⟖ S)
All the tuples from right relation s are included in resulting relation but the tuples of left relation without any matching domain will be discarded.

Full Outer join.
All the tuples from both relations are included in resulting relation. If there are no matching tuples for both relation then their respective unmatched attribute will be null.

Q. employee (person_name, street, city)
works (person_name, company_name, salary
company (company_name, city)
manager (person_name, manager_name)

Consider the relational database where the primary keys are underlined. Give the expression in relational algebra to express each of the

1) Find the name of all employees who work for SBI
2) Find the name and city of residence of all employees who work for SBI.
3) Find the name of all employees who live in same city as the company for which they work.
4) Find the name, street address, city of all employees who work for SBI and earn more than 20,000 per month.

1) $\Pi_{person\_name}(\sigma_{company\_name = SBI}(works))$ (R1)

2) $\Pi_{person\_name, city}(employee \bowtie_{employee.person\_name = works.person\_name} (R1))$

3) (employee $\bowtie$ R1)

3) Π_{person_name} (employee $\bowtie$ works $\bowtie$ company)

4)

AGGREGATE FUNCTIONS AND GROUPING

Aggregate functions are the functions where multiple values are grouped together to form a single value.

representation:

$\mathcal{J}$

Notation of Aggregate function:

$$\langle \text{grouping attribute} \rangle \; \mathcal{J} \; \langle \text{function list} \rangle \; (R)$$

(function list: Ag-fn; (R): Relation)

Grouping attribute is a list of attributes of the relation specified in R.

Function list is the list of function attribute where function is aggregate function such as sum, avg, min, max, count, etc.

and attribute is attribute of relation specified by 'R'.

Q. Find out department no., no. of employees in each dept and their average salary from each department.

$\mathcal{J}$

⇒ Dept. No. count emp No., Avg, Salary (Employee)

RELATIONAL CALCULUS

Alternative way of formulating queries is called Relational calculus.

Two forms of Relational calculus:

1) Tupple RC
2) Domain RC

Tupple RC:

It is a non procedural query language which is based on no. of tupple variables for which predicate (condition) hold true.

A query in tupple relational calculus is expressed as

$\{t \mid cond(t)\}$ or $\{x \mid P(x)\}$ → conditional exp. involving tupple

where t and x are tupples

Q. Find all the employees working in dept_id = 20.

> Relational Algebra:

$\sigma_{dept\_id=20}(Emp)$

Tupple RC:

$\{t \mid emp(t) \wedge t.dept\_id = 20\}$

SQL:

```
Select * from emp where dept_id = 20
```

Q. Find out the employee id, employee name of all employees working in dept. id: 20.

⇒ SQL:

Select emp id ~~from~~ ename from emp where dept id=20

Relational Algebra:

$$\Pi_{\text{emp id, ename}}(\sigma_{\text{deptid:20}}(\text{Emp}))$$

Tuple RC:

$$\{t.\text{emp id}, t.\text{ename} \mid \text{Emp}(t) \wedge t.\text{dept\_id} = 20\}$$

EXISTENTIAL AND UNIVERSAL QUATIFIERS

It read as 'there exist' i.e in a given set of tuples there is atleast one occurrence whose values satisfy a given condition.

Eg:

$$\exists x(x.\text{salary} = 12000)$$

There exists one or more tuple in relation that have value 12000 for the attribute, salary.

It reads as "for all" which means that in a given set of tuples, all the tuples satisfy a given condition.

eg:

GENERAL EXPRESSION OF TRC

$\{t_1.A_1, t_2.A_2, \ldots t_n.A_n / cond(t_1, t_2, \ldots t_n, t_{n+1}, t_{n+2} \ldots t_{n+m}\}$

A_i = Attribute of Relation

↓

WFF (well formed formula of TRC)

Q. Find out all the employees having salary > 10,000 and dept_id = 10

Q. loan (loan_no., branch_name, amount)

a) Find loan_no., amount, for loan of $1200.

b) Find all the details of loan of $5000.

QUERY BY EXAMPLE (QBE)

Q. branch (branch_name, branch_city, assets)
customer (customer_name, customer_street, customer_city)
account (account_number, branch_name, balance)
loan (loan_number, branch_name, amount)
depositer (customer_name, account_number)
borrower (customer_name, loan_number)

Queries on one Relation:

Q. Find out all the loan numbers at SBI branch.

loan	loan_no.	branch_name	amount
	P.	SBI	

P. Print
P._x
P. All

Queries on multiple relation

Q. Find the name of all customers who have loan from SBI

loan	loan no.	branch_name	amount
	_x	SBI	

borrower	customer_name.	loan_number
	P._y	_x

Ascending Order - P.^D
descending Order - P.DD

Condition Box:

Q. Find out all the account no's with balance greater than 50,000 and less than 1 lakh.

account	account-number	branch-name	balance
	P.		-x

Conditions
- $x \geq 50{,}000$
- $x \leq$ 1 lakh

Result Relation:

Q. Find the customer name, account no. and balance for all customers who have an account at SBI.

account	account-number	branch-name	balance
	-y	SBI	-z

depositor	customer-name	account-no.
	-x	-y

result	customer-name	account-no.	balance
P	-x	-y	-z

Deletion

Q. Delete customer whose name is Smith.

customer	customer-name	cust-street	customer city
D.	Smith		

Insertion:

Q. Insert a new customer who lives in Chandigarh and whose name is Smith.

customer	customer-name	customer-street	c.city
I.	- u		- y

- u = 'Smith' ∧ - y = 'chd'

Updation (U.):

Q. Same Q. as above: (update city 'chd')

customer	c-name	c-street	c-city
U.	Smith ↕		- y chd

- u = 'Smith'

Aggregate operations:

Q. Find the total balance of all accounts maintained at SBI.

account	account no.	branch-name	balance
		SBI	P. sum ↓ for all sum P. sum. all

NORMALIZATION

It is process of spliting the relation into two or more relations with fewer attributes thereby minimizing the redendency of data and minimizing insertion, deletion and updation anomalies.

- The basic objective of Normalization is to reduce redendency.

Redendency:

It is unnecessary repetition of data which can cause problems with storage, with searching, insertion deletion and updation of data.

Normal forms:

It is an algorithm we use to test the structure of table. It helps to eliminate possible anomalies in a relation.

Types of Normal form:

1) First Normal Form (1NF)
2) 2nd Normal Form (2NF)
3) 3rd Normal Form (3NF)
4) 4th Normal Form (4NF)
5) 5th Normal Form (5NF)
6) Boyce Codd Normal Form (BCNF)
7) 6th Normal Form (6NF)

Functional dependencies:

A functional dependency exist between two fields A and B of a relation R when A distinct value of A is directly associated with distinct value of B.

Representation: $A \rightarrow B$

It means the value of A can determine the value of B

i.e. the value of B is functionally dependent on value of A.

eg:

Primary key functionally determines all other non-key attributes in a table.

For eg:

Student(Roll NO., Name, Age)

{Roll NO} → {Name}

{Roll NO.} → {Name, Age}

Types:

1) Trivial Functional Dependency
2) Non-Trivial Functional Dependency
3) Fully Functional Dependency
4) Partial Functional dependency
5) Transitive Functional Dependency
6) Multivalued Functional Dependency
7) Join Functional dependency

Trivial Functional dependency:

If FD

$A \to B$ holds where B is a subset ($\subseteq$) of A then it is known as Trivial Functional dependency.

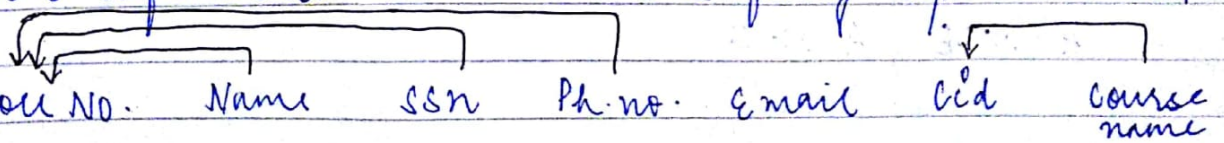
Non Trivial Functional dependency:

If FD

$A \to B$ holds, where B is not a subset of A then it is known as Non Trivial Functional Dependency.

Fully Functional Dependency

It is defined as $A \to B$ where B is fully dependent on A.



Partial dependency:
It is a functional dependency where the determinant consist of key attributes but not the entire primary key and determine consists of Non key attributes

Eg:
$A \rightarrow B$ holds where B is not fully dependent on A.

[fname, lname] → Address

fname, lname: primary key → determinant
Address: Non key → determine

Transitive dependency:
It is a functional dependency where determinant consist of non key attribute and determine also consist of Non key attribute.

Eg:

A	B	C
RollNo.	Name	Age

$A \rightarrow B$

$B \rightarrow C$

$\therefore A \rightarrow C$

Multivalued dependency:
It is the dependency where one attribute value is potentially having a multivalue fact about another value. It means that if a relation 'R' having A, B, C as attributes then B, C are multivalue facts about A which is represented as

$A \twoheadrightarrow B$, $A \twoheadrightarrow C$

Eg:

Course	Student Name	Textbook
Physics	Ankit	Mechanics
Physics	Ankit	Organic
Physics	Rahat	Mechanics
Chemistry	Ankit	Organic
Chemistry	Ankit	Inorganic

Join Dependency:
If a relation 'R' is equal to join of projections X, Y, Z where X, Y, Z are attributes of relation 'R' and if there is a loss-less decomposition.

Eg:

Agent	Company	Product Name
Sumeet	ABC	Nail
"	ABC	Screw
"	CDE	Bolt
Raj	ABC	Bolt

⇓

Agent	Company
Sumeet	ABC
"	CDE
Raj	ABC.

Agent	Product Name
Sumeet	Nail
"	Screw
"	Bolt
Raj	Bolt

Anomalies in Normal Forms:
There are 3 types of Anomalies:-
1) Insert Anomali
2) Delete Anomali
3) Update Anomali

Eg:
Employee
E_id E_name Rano. Deptno. Deptname Loc

Insert Anomali
It exist in a table when there is unnecessary or unreasonable constrain placed upon the task of adding a new record i.e. which can cause unnecessary data redendency.

Eg: You can't insert data for a new department until you have atleast one employe asigned to that department.

Delete Anomali
It exist when deleting a record could remove data not intended for deletion.

Eg: If there is only one employee in deptment 'Information services' and are want to delete the record of that employe then it will also delete the info. that you have of department of name 'Information Services'.

Update Anomali:
It occurs when we are not able to modify the records in the database without taking care of other facts.

Eg: If we want to update department name, then we have to update it in each and every record in that relation.

Normal Forms:

1) 1NF: There are two approaches to normalize table in first normal form.

UNF:

P_id	colour	Price
1.	Red, Green	45/-
2	Yellow.	23/-
3	Green	17/-
4.	Yellow, blue	39/-
5	red	29/-

ways to Normalize a table in 1NF:-

1) Flattening the table:
It removes repeating groups by filling the missing entries of each incomplete row of table with copy of their corresponding Non repeating attributes.

⇒

P_id	Colour	Price
1	Red	45/-
1	Green	45/-
2	Yellow	23/-
3	Green	17/-
4	Yellow	39/-
4	Blue	39/-
5	Red	29/-

2) Decomposition of table :-
In this table is to be decomposed into two new tables that will replace original table. Rules to decompose the table are :-
1) One of the two table contain the table identifier i.e primary key of original table and all non-repeating attributes.
2) The other table contain a copy of table identifier and all repeating attributes

P_id	Price
1	45/-
2	23/-
3	17/-
4	35/-
5	29/-

P_id	Colour
1	Red
1	Green
2	Yellow
3	Green
4	Yellow
4	Blue
5	Red

closure of a set of functional dependencies:
The set of functional dependency that is logically implied by F is called the closure of F and is written as F+.

Eg: R(A, B, C, D)
with F.D {A → B, A → C, BC → D}
To find closure:
$[A^+]$ = {A, B, C, D}
↓
It contains all individual attributes. ∴ it is called candidate key.

$[B^+]$ = [B] C^+ = [C]

$[D^+]$ = {D}

Second Normal Form (2NF)
The entity should be considered already in first Normal Form and all the attributes within the entity should depend only on unique identifier of the entity.
There must not be any partial dependency of any column on primary key.
In second NF, every non prime attribute should be fully functionally dependent on prime key attribute
i.e. If X → A holds
then there should not be any subset (Y) of X for which Y → A holds true.

Eg: Student - Project

Stu - id	Proj - id	Stu - Name	Proj - Name
1	101	A	DBMS
2	102	B	Java
3	101	C	DBMS
4	103	A	C++

[Stu-id, Proj-id]

↓ ↓

Stu-name Proj-name

Stu-Id	Proj-Id	Stu-Name
1	101	A
2	102	B
3	101	C
4	103	A

Proj-Id	Proj-Name
101	DBMS
102	Java
103	C++