DataBase Management System

Data :-

= Data is the collection of information, facts and figures that are stored in or used by computers.

= So, in the context of database refers to all the single items that are stored in database either individually or as a set.

Information:-

Information is the processed, organized data that is beneficital in ~~proving~~ providing useful knowledge.

Database :-

The collection of structured or organized data usually referred to us the database, contains information relevant to an enterprise.

Database Management System (DBMS):-

= To manage the database by a tool is known as Database Management System.

= DBMS is a software which is responsible for performing all the types of operations such as insertion, updation, retrieving, fetching and delete the database.

= DBMS is a intermediate between developer and database.

= A database management system is a collection of inter-related data and set of program to access this ~~pro~~ data

= Example of DBMS —

SQL Server, Oracle, MySQL, DB_2 etc.

AIM / GOALS OF DBMS :-

= Providing a way to store and retrieve database information i.e both convenient and efficient.

= Ensuring the safety of the information.

APPLICATIONS OF DATABASE :-

⇒ Banking ⇒ Airline

⇒ Railway ⇒ Tours

⇒ University. ⇒ Hotels

ADVANTAGES OF DBMS :- / Disadvantages of File System

In file processing system duplicate data is created in many places because all the programs have their own files. This creates data redundancy which in turns wastes labor and space. In DBMS all the files are integrated in a single data base.
Example - Saving A/c | current A/c

⇒ Data redundancy and inconsistency :-

⇒ Difficulties in accessing data :-

⇒ Data isolation :-

⇒ Integrity problem :-

⇒ Atomicity problem :-

All the transiction must be atomic means it's must happens in it's entirely or not at all.

It is difficult to ensure in a conventional file processing system

⇒ Concurrent access problems / anomalies :-

⇒ Security :-

In DBMS, the data is stored in hierical based where as in file system, everyone can access the data.

COMPONENTS OF DBMS :-

i) Hardware :- Actual Computer
Hard, disk

ii) Software :- Actual DBMS

iii) Users -

iv) Data - The most vital component is data

v) Procedure -

NOTE :-

DBMS allows users to do following task :-

i) Data Definition
ii) Data Updation
iii) Data retrieval
iv) User Administration

⇒ Procedure –

Procedure refers to the rules that govern the design of database along with that we require following procedures to run the system.

i) Logic to DBMS
ii) Use a particular DBMS facility.
iii) Start and stop the DBMS
iv) Make backup copies of database
v) Handle hardware or software failures.

DATABASE USERS :-

i) Naive User
ii) Application Programmer
iii) Sophisticated user
iv) Specialized user
v) DBA (Database Administrator ~~Application~~)

i) Naive User :-
Don't know or aware of database.

ii) Application Programmer :-
Those who are writting the code or to execute the program. to access the database

iii) Sophisticated user :-
Those who are designing database by using the query.

iv) Specialized user :-
Those who ~~are~~ access the database by writing the query.

v) DBA :- DBA can be a single person or it can be a group DBA ~~Those who~~ are responsible for everything that is related to database.

= DBA is responsible for authorizing access to the database by grant and revoke permissions to the user for co-ordinating and monitoring it's uses.

VIEW OF DATA :-

= A database system is collection of interrelated data and a set of programs that allow users to access and modify these data

= A major purpose of a database is to provide users with an abstract view of the data, that is the system hides certain details of how the data stored and maintained.

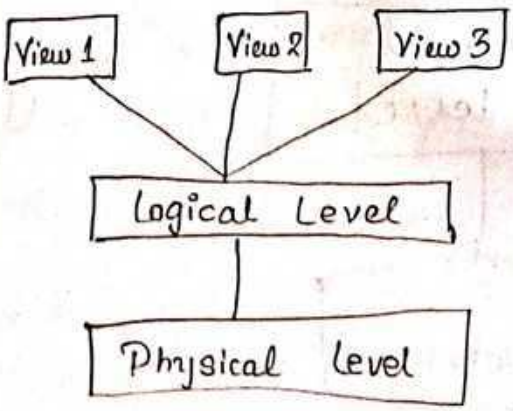
DATA ABSTRACTION :-

= Database systems are made-up of complex data structures. To ease the user interaction with database, the developers hide inter irrelevant details from users. This process of hiding irrelevant details from users is called data abstraction.

#3 Schema Architecture :-

Physical level / Internal Schema :-

= This is the lowest level of data abstraction. It describes how data is actually stored in database. You can get the complex data structure details at this level.



Logical Level / Conceptual Schema :-

= This is the middle level of 3-level data abstraction architecture. It describes what data is stored in database.

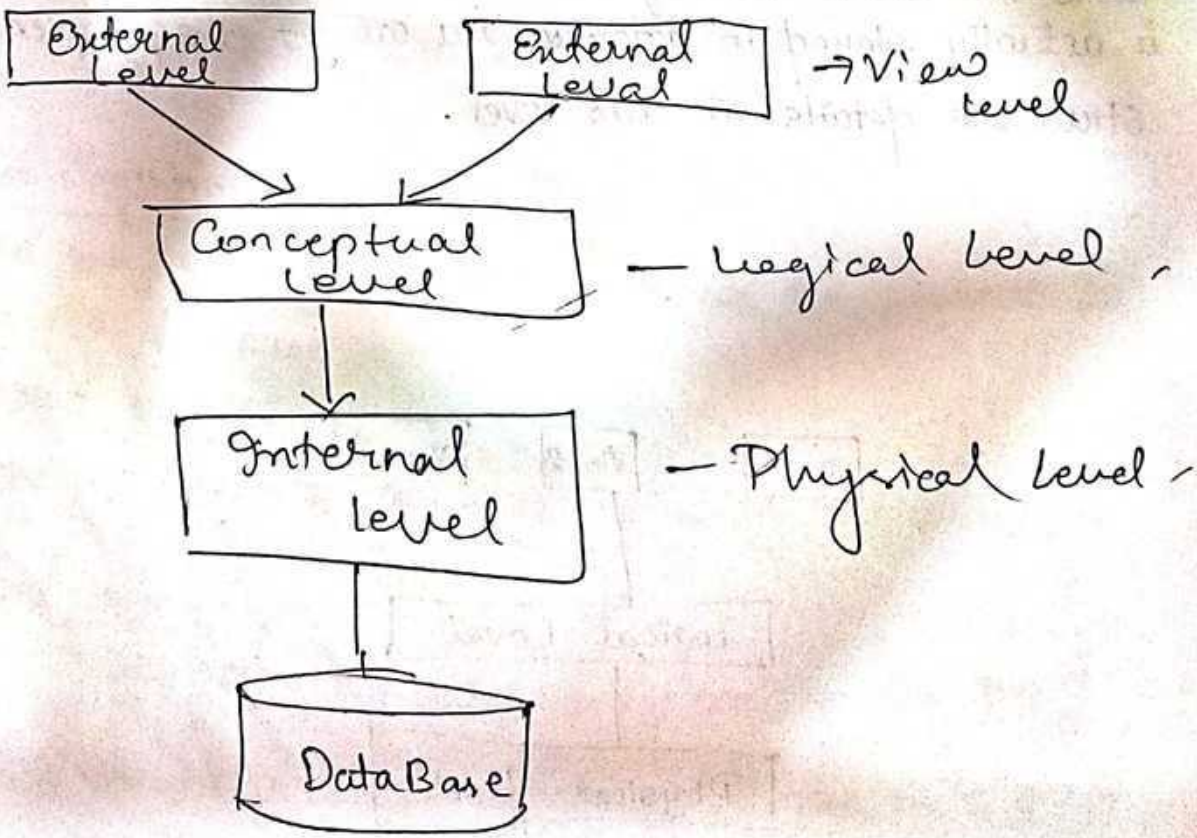
= At the logical level these records can be described as fields and attributes along with their data types, their relationship among each other can be logically implemented.

= The programers generally work at this level because they are aware of such things about database systems.

View Level / External Schema :-

= Highest level of data abstraction. This level describes the user interaction with database system.

= At view level, user just interact with system with the help of GUI and enter the details at the screen, they are not aware of how the data is stored and what data is stored; such details are hidden from them.



Database Schema and Instance :-

= The description or design of a database is called Schema, which is specified during database design and is not expected to change frequently.

= Most of the data models have certain conventions for displaying schemas in diagrams.

= A displayed schema is called Schema diagram.

= Database change over time as the information is inserted and deleted. The collection of information stored in the database at a particular moment is called an instance of the database.

DATA INDEPENDENCE :-

= Data Independence is the ability to modify the schema at one level of the database system without altering the schema at the next higher level.

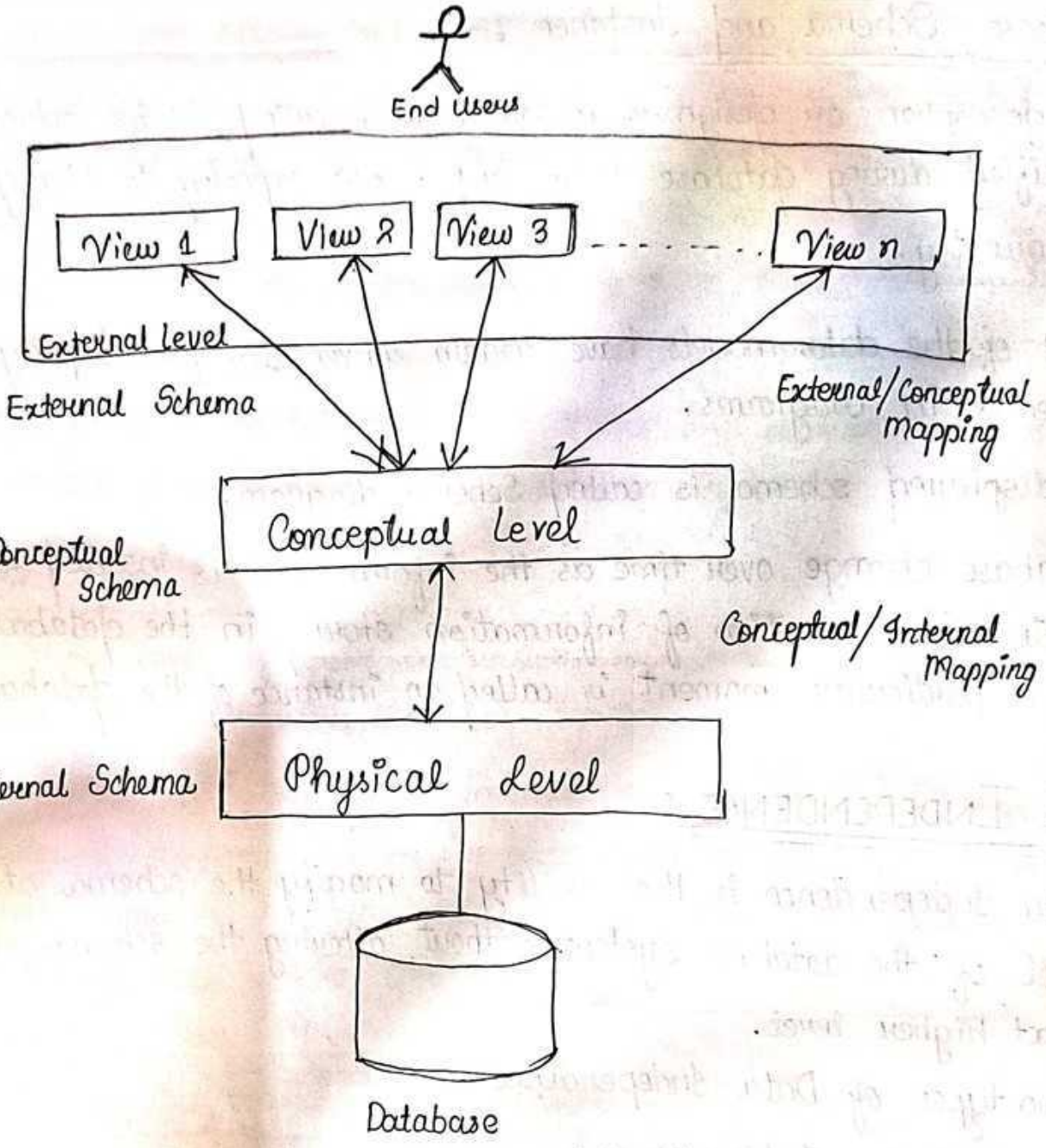
= Two types of Data Independence -

i) **Logical Data Independence :**

The ability to modify the logical schema without changing the external views and application.

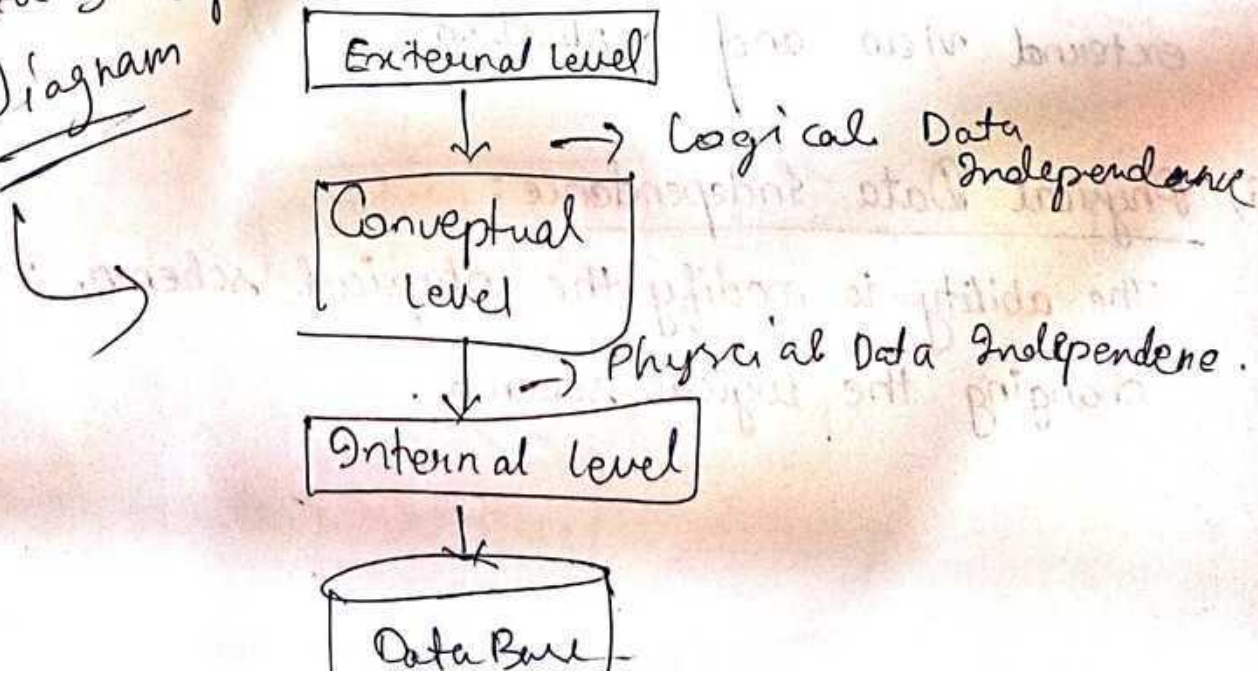
ii) **Physical Data Independence :**

The ability to modify the physical schema without changing the logical schema.



[3-level DBMS Architecture]

11 Data Independence Diagram



DBMS Architecture :-

= The design of a DBMS depends on it's architecture.

= It can be centralized, decentralized or hierarchical.

= The architecture of a DBMS can be seen as either single tier or multitier. An n-tier architecture divides the whole system into related but independent n modules, which can be independently modified, altered, changed or replaced.

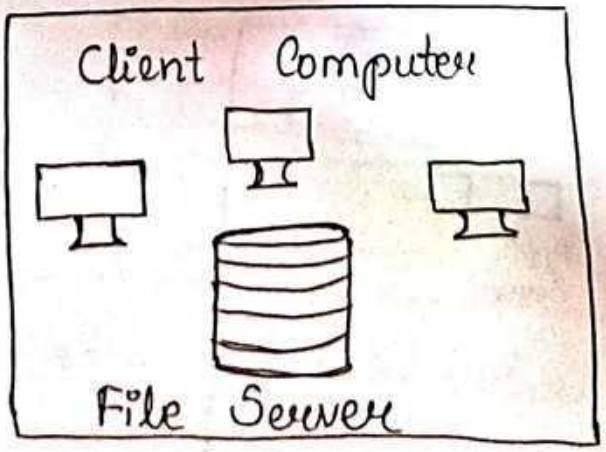
= The architecture of a DBMS can be seen as either single tier or multi-tier. The tier are classified as follows :

i) 1-tier architecture
ii) 2-tier architecture
iii) 3-tier architecture

i) 1-Tier Architecture :

= Basically, a one-tier architecture keeps all of the elements of an application, including the interface, Middleware and back-end data, in one place. Developers see these types of systems as the simplest and most direct way.

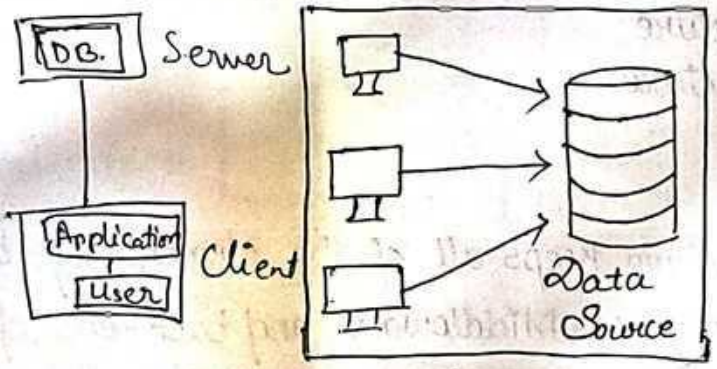
= In 1-tier architecture, the DBMS is the only entity where the user directly sits on the DBMS and uses it. Any changes done here will directly.



ii) 2-Tier Architecture :-

= The two-tier is based on Client Server architecture. The two-tier architecture is like client server application. The direct communication takes place between client and server. There is no intermediate between client and server.

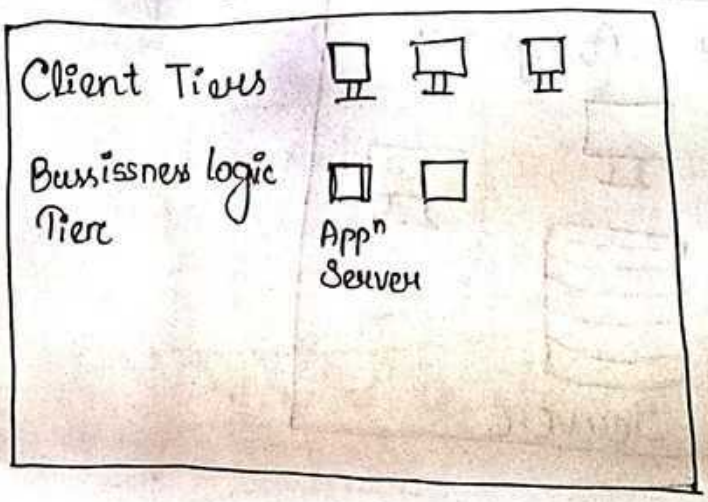
= If the architecture of DBMS is 2-tier, then it must have an application through which the DBMS can be accessed. Programmers use 2-tier architecture where they can access the DBMS ~~language~~ by means of an application. Here the application tier is entirely independent.



* API like ODBC & JDBC are used for this interaction.

iii) 3-Tier Architecture :-

A 3-tier architecture separates it tiers from each other based on the complexity of the users and how they use the data present in the database. It is the most widely used architecture to design a DBMS.

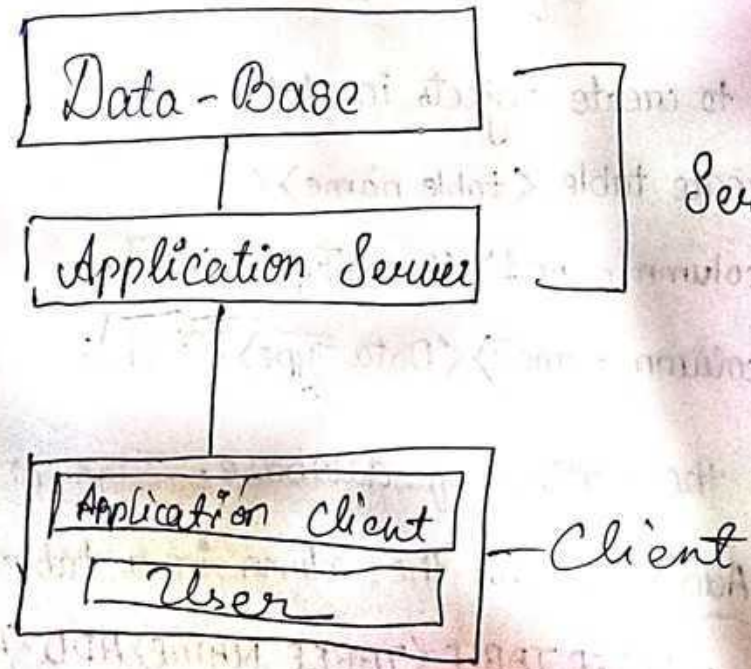


= The 3-tier architecture contains another layer between client and server. In this architecture the client cannot be communicate directly with the server.

= The application on the client end interactaet with applicatio server which further communicate with the database system.

= The end user has no no idea about the existance of the database beyond application server and the database also has no idea about any other user beyond the application.

= It is used in case of large applications.



Mapping between views:-

1) External Conceptual Mapping
2) Conceptual Internal Mapping.

Database Languages :-

= Database language are used for read, update and store data in a database. There are several such languages that can be used for this purpose.

= The types of database language are :-

i) DDL → Data Definition Language.
ii) DML → Data Manipulation Language
iii) DCL → Data Control Language.
iv) TCL → Transaction Control Language

i) DDL (DATA DEFINITION LANGUAGE) :-

= DDL is used to define database patterns or structures.

▶ CREATE - used to create objects in database.

Syntax :- Create table <table name> (
<column name1> <Data Type> [Size],
<column name2> <Data Type> [Size]);

▶ ALTER - alter the pattern of database.

i) Alter-Add - To add the column in a table.

Syntax : ALTER TABLE <TABLE NAME> ADD <NEW COLUMN NAME> <DATA TYPES> [SIZE];

ii) Alter-Drop - To drop a column from the existing table.

Syntax: ALTER TABLE <TABLE NAME> DROP <COLUMN> <COLUMN NAME>;

iii) ALTER-RENAME :- To rename a column name to new name.

Syntax : ALTER TABLE <TABLE NAME> RENAME COLUMN <
<OLD COLUMN NAME> TO <NEW COLUMN NAME>;

iv) ALTER-MODIFY :- To modify the datatype (size) of the existing column.

Syntax : ALTER TABLE <TABLE NAME> MODIFY <COLUMN NAME>
<NEW DATATYPES> [NEW SIZE];

▶ DROP :- helps in deleting objects.

Syntax :- DROP TABLE <TABLE NAME>;

▶ TRUNCATE - erase all records from the table.

Syntax - TRUNCATE TABLE <TABLE NAME>;

▶ RENAME - useful in renaming an object.

RENAME <OLD TABLE NAME> TO <NEW TABLE NAME>;

ii) DML (Data Manipulation language) :-

= DML is used for accessing and manipulating data in a database

▶ SELECT :- useful in holding data from a database

Syntax - SELECT * from Tab;

▶ Insert :- helps in inserting data into table

Syntax - i) Insert into tablename values (val1, val2, val3....)

ii) Insert into tablename (col1, col2, col3,...) values (val1, val2, val3...).

▶ Update :- used in updating the data.

Syntax - Update tablename
set col1=val1, col2=val2,...
where condition;

DELETE :- do the function of deleting the records.

Syntax - Delete from Tablename
where condition;

iii) DATA CONTROL LANGUAGE (DCL) :-

= DCL is used for granting and revoking user access on a database.

Grant - gives user's access privileges to database

Syntax - Grant privileges on object to user;

Ex - Grant select, insert on student to faculty.

Revoke - withdraw access privileges given with the GRANT command

Syntax - Revoke privileges on object from user;

Ex - Revoke select, insert on student from faculty.

iv) TRANSACTIONAL CONTROL LANGUAGE (TCL) :-

= Transaction Control Language has commands which are used to manage the transactions on the conduct of a database.

= They manage the changes made by data manipulation language statements and also group up the statements in a logical management.

Commit - use to save work permanent

Save Point - helps in identifying a point in the transaction, can be rolled back to the identified point.

Roll Back - It has the feature of restoring the database to the genuine point, since from the last COMMIT.

CONSTRAINTS :-

= Constraints are nothing but conditions.

= Every relation has some conditions that must hold for it to be a valid relation.

= Constraints enforce limits to the data or type of data that can be inserted, updated, deleted from a table.

= The whole purpose of constraints is to maintain the data integrity into a table.

= Constraints ~~avaible~~ available in SQL are —

Pre-defined constraints:
- i) NOT NULL
- ii) UNIQUE
- iii) PRIMARY KEY
- iv) FOREIGN KEY
- v) CHECK
- vi) DEFAULT
- vii) DOMAIN CONSTRAINTS

i) NOT NULL :-

= NOT NULL means empty i.e. the value is not available OR not applicable.

= Whenever a table's column is declared as NOT NULL, then the value for that column cannot be empty for any of the table's records.

= There must exist a value in the column to which the NOT NULL constraints is applied.

Syntax : NOT NULL (CONDITION)

```
CREATE TABLE student (StudentID INT NOT NULL,
        Student_FirstName VARCHAR(20));
```

Ex :- In a table one or more NOT NULL can be present.

ii) UNIQUE :-

= Duplicate values are not allowed in the columns to which the UNIQUE constraint is applied.

= The column with the unique constraint will always contain a unique value.

= This constraint can be applied to one or more than one column of a table, which means more than one unique constraint can exist on a single table.

Syntax:

```
CREATE TABLE student (StudentID INT UNIQUE,
        Student_FirstName VARCHAR(20));
```

iii) PRIMARY KEY :-

= Primary Key Constraint is a combination of NOT NULL and UNIQUE constraints.

= NOT NULL constraint and a UNIQUE constraint together forms a PRIMARY constraint

= The column to which we have applied the primary constraint will not contain a duplicate value and will not allow null values.

= In a table, only one column declare as a primary key.

Syntax -

```
CREATE TABLE student (StudentID INT PRIMARY KEY, Student_FirstName
VARCHAR(20), Student_LastName VARCHAR(20));
```

= If table doesn't contain Primary Key then we add Primary Key using

```
ALTER TABLE student ADD PRIMARY KEY (StudentID);
```

iv) FOREIGN KEY :-

= A foreign key is used for referential integrity.

= When we have two tables, and one table takes reference from another table i.e. the same column is present in both the tables and that column acts as a primary key in one table That particular column will acts as a foreign key in another table.

Primary Key

Student

Roll No.	Name	Course
101	Ally	Computer
105	Nobita	IT
107	Doremon	Biology
108	Ninja	Math

(Referencing Table)

Foreign Key

Department

Roll No.	Department
101	CSE
105	CSE
107	Science
108	Math.

(Referenced Table).

= In a table, one or more foreign key can be present.

Syntax :- (for Dept. Table)

CREATE TABLE ~~student~~ Department (StudentID INT REFERENCES ~~Student~~ Department, ~~Student~~ FIRSTNAME VARCHAR (20), ~~Student~~ Department_LastName VARCHAR (20));

= CREATE TABLE Department (
Department VARCHAR (20),
StudentID References, StudentID (Student));

NOTE :-

= In child table creation for using foreign key column name and table name from base table is required.

v) CHECK :-

= Whenever a check constraint is applied to the table's column, and the user wants to insert the value in it, then the value will first be checked for certain conditions before inserting the value into that column.

= Syntax :

CREATE TABLE student (StudentID INT, Student_Name VARCHAR (20), Age INT CHECK (Age <=15));

= If CHECK constraint is not given at the time of creation we can add using.

= ALTER TABLE student ADD CHECK (Age <=15);

- It can be use multiple time in a table.

vi) DEFAULT :-

= Whenever a default constraint is applied to the table's column, and the user has not specified the value to be inserted in it, then the default value which was specified while applying the default constraint will be inserted into that particular column.

= Syntax -

```
CREATE TABLE student (StudentID INT, Student_Name VARCHAR(20),
Student_Email_ID VARCHAR(40) DEFAULT "snsahu@gmail.com");
```

= Default constraint can be use multiple time in a table.

vii) DOMAIN CONSTRAINTS :-

⇒ Domain constraints are user defined data type and we can define them like this :

⇒ Domain constraint = datatype + Constraints

⇒ Example : For example I want to create a table "student info" with "stu id" field having value greater than 100, I can create a domain and table like this:

```
create domain id value int
check(value > 100)

create table student_info (
stu_id id value PRIMARY KEY,
stu name varchar (30),
stu age int
);
```

ENTITY :-

= An entity in database is a real world object which is having some characteristics.

= Example : Student is an entity and it's attributes are Name, RollNo, Id, etc.

ATTRIBUTE :-

= An attribute is nothing but characteristics of an entity.

= For example -
Attributes of employee can be EmpID, Ename, Dept, DOJ, Salary etc are attributes.

DEGREE :-

= Degree in SQL refers to the number of attributes or columns in a table.

CARDINALITY :-

= Cardinality means how the entities are arranged to each other or what is the relationship structure between entities in a relationship set.

= In a Database Management System, Cardinality represents a number that denotes how many times an entity is participating with another entity in a relationship set.

= In a table, the number of rows or tuples represents the cardinality.

Cardinality Mapping :- L·8

= This is also known as Cardinality Ratio.

= This represents the mapping of one entity set to another entity set in a relationship set.

Types Of Cardinality Mapping :- L·9

There are four types of cardinality mapping –

i) One to One

ii) Many to One

iii) One to Many

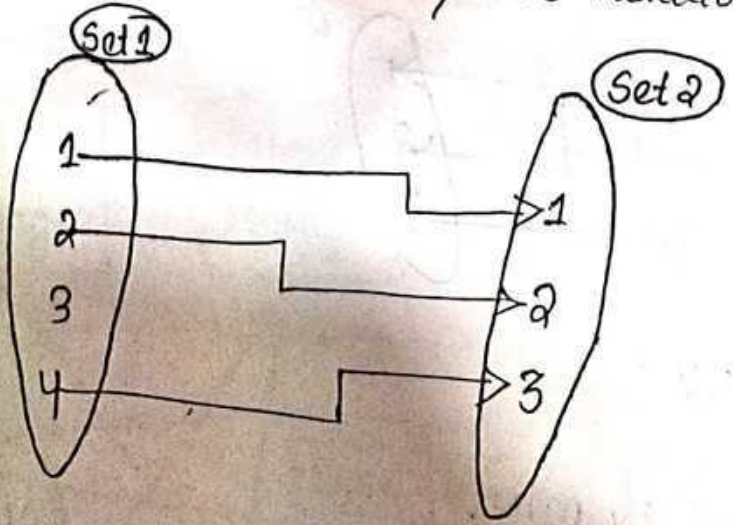
iv) Many to Many

i) ONE-TO-ONE :-

⇒ This is represented by a 1:1 symbol.

⇒ Here there is at most one relationship from one entity to another entity.

⇒ Eg :- A student can have only one relationship.



ii) Many-to-One :-

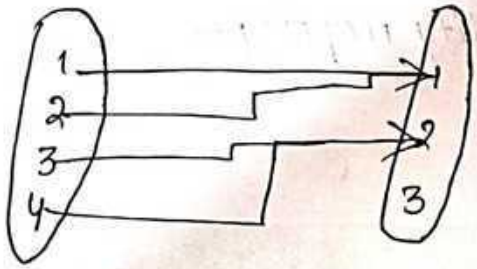
⇒ This mapping can be represented as n:1 / m:1.

⇒ In this mapping, from set 1, there can be multiple sets that can make relationships with a single entity of set 2.

⇒ For example -

i) There are many student sitting in a classroom.

iii)

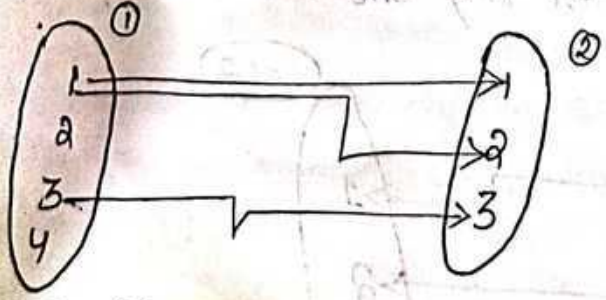


iii) ONE-TO-MANY :-

⇒ It can be represent by 1:n / 1:m.

⇒ In this mapping, from set 1 there can be a maximum single set that can make relationships with a single or more than one entity of set 2.

⇒ Eg :- In a classroom, a teacher is teaching to no. of students.

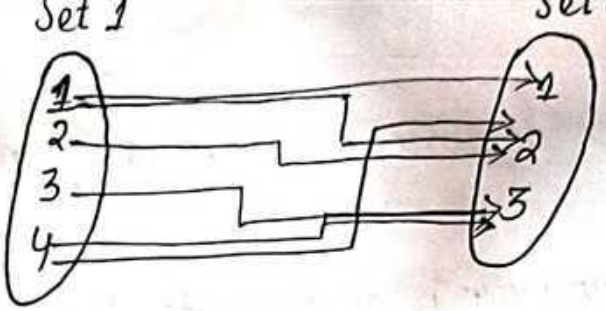


iv) MANY-TO-MANY :-

⇒ This is represented by M:N.

⇒ Here multiple entity of set 1 can be associate with multiple entity of set 2.

⇒ Eg - Multiple students can work of single project or a single student can work on multiple project.



DATABASE SYSTEM ARCHITECTURE :-

Native Users
Application Programmers
Sophisticated Users
Database Administrator
Application Interface
Application Programs
Query Tools
Administration Tools
Compiler & Linkers
DML Queries
DDL Interpreters
Application Program Object Code
DML Compiler & Organizers
Query Evaluation Engine
Query Processor
Buffer Manager
File Manager
Authorization Integrity Manager
Transaction Manager
Storage Manager
DISK STORAGE
Data
Indices
Data Dictionary
Statistical Data

Query Processor :-

⇒ Query Processor helps the database system, simplify and facilitate access to data.

⇒ The work of query processor is to execute the query successfully

⇒ It interprets the requests (queries) received from end users via an application program into instructions.

⇒ It also execute the user request which is received from the DML compiler.

Query Processor Components :-

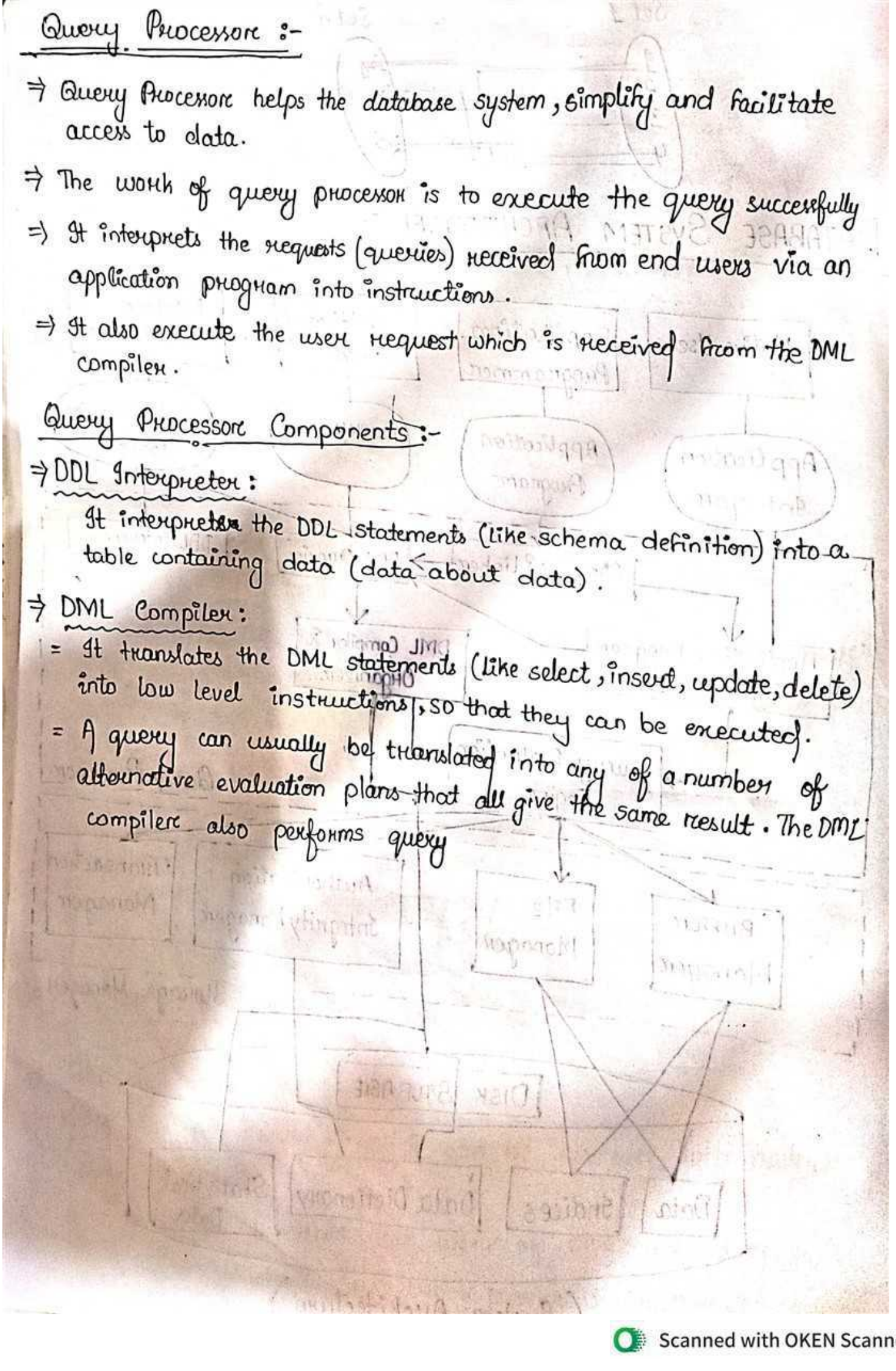
⇒ **DDL Interpreter :**

It interprets the DDL statements (like schema definition) into a table containing data (data about data).

⇒ **DML Compiler :**

= It translates the DML statements (like select, insert, update, delete) into low level instructions, so that they can be executed.

= A query can usually be translated into any of a number of alternative evaluation plans that all give the same result. The DML compiler also performs query



Storage Manager Components :-

⇒ **Authorization Manager and Integrity Manager** –

It checks the authority of users to access data and checks the integrity constraints when database is modified.

⇒ **Transaction Manager** –

It ensures that the database remains in a consistent state despite system failures and that concurrent transaction execution proceed without conflicting.

⇒ **File Manager** –

It manages the allocation of space on disk storage and the data structures used to represent information stored on disk.

⇒ **Buffer manager** –

It responsible for fetching data from disk-storage into main memory and deciding what data to cache in main memory.

DISK STORAGE :-

⇒ The storage manager implements the following data structures as a part of physical system implementations.

Data Files – It stores the database itself.

Data Dictionary – It contains the information about the structu of any database, or it stores metadata about the database, in o particular the schema of the database.

DATA MODELS IN DATABASE :

⇒ Data models define how the logical structure of database is modelled and defines how data will be stored, accessed and updates in database system.

⇒ Data Model can be defined as an integrated collection of concepts for describing and manipulating data, relationship between data and constraints on data in an organization.

⇒ The purpose of a data model is to represent data and to make the data understandable.

⇒ The different types of data models are :-

i) Hierarchical Model

ii) Network Model

iii) Entity-Relationship Model

iv) Relational Model.

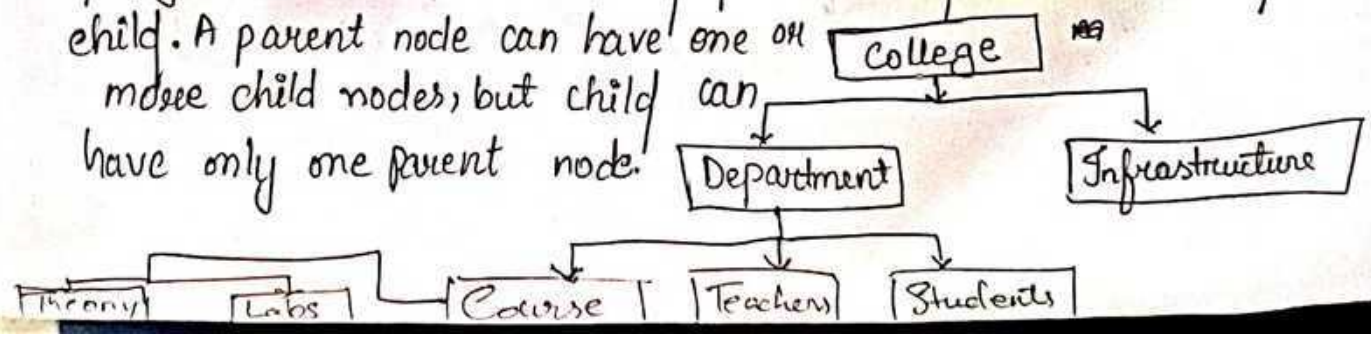
i) Hierarchical Model :- (Also known as Inverted Tree)

= The Hierarchical model arranges record in hierarchy like an organizational chart.

= Each record type in this model is called node or segment.

= A node represents a particular entity, the top most node is called root. Each node is a sub-ordinate of the node that is at the next higher level.

= A higher level node is called parent and lower level node is called child. A parent node can have one or more child nodes, but child can have only one parent node.



Drawback -

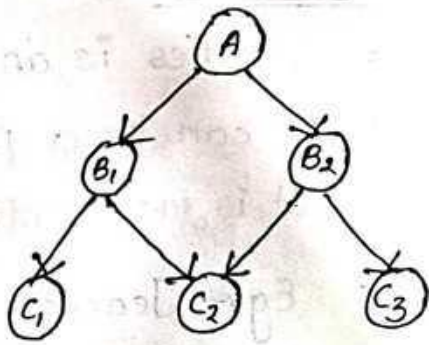
⇒ Relationship that are complex are not supported.

⇒ Because it only supports one parent per child node, if we have a complex relationship in which a child node needs to have two parents, we won't be able to describe using this model.

⇒ When a parent node is removed, the child node is also removed

(Diagram)

ii) Network Model :-



⇒ The Network model is similar to hierarchical model. The data are organized like graph.

⇒ The difference is that child node can have more than one parent nodes.

⇒ The child nodes are represented by arrows in network model.

⇒ It also provides more flexibility than hierarchical model.

Drawback -

⇒ It is slow complex and more difficult to maintain.

⇒ It requires more complex diagrams to represent a database.

iii) Entity-Relationship Model :-

⇒ ER model is a high level data model diagram.

⇒ ER model describes the structure of a database with the help of a diagram, which is known as ER diagram.

⇒ An ER model is a design or blueprint of a database that can later be implemented as a database.

⇒ It is based on the notation of real world entities and relationship among them.

⇒ It follows 3 components :-

i) Entities –

= Entities is an real world things or object.

= It can be a person, place or even a concept. It is represented by rectangular shape.

= Eg - Teacher, student.

ii) Attributes –

= An entity contains real world property called attributes. This is the characteristics of that attributes.

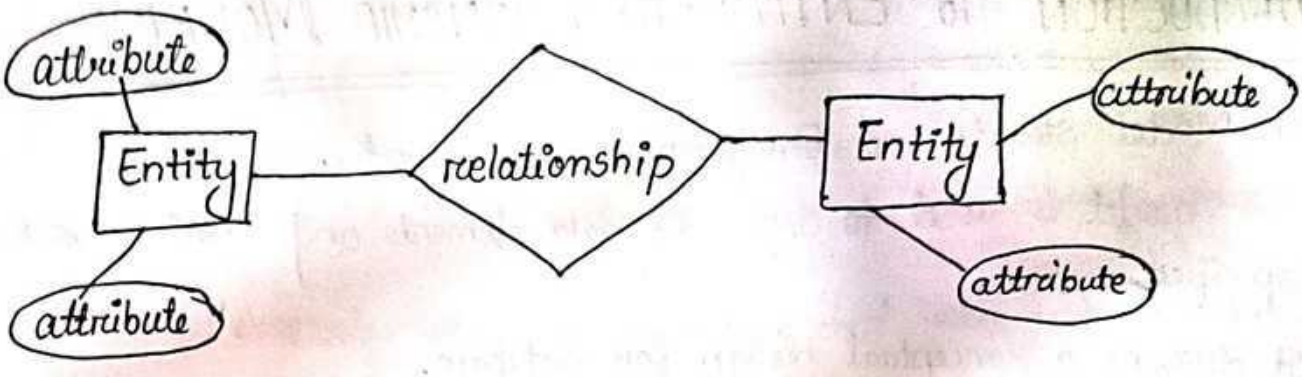
= This can be represented as eclipse.

= Ex :- The entity teacher has property like teacher_id, name, salary, age etc.

iii) Relationship –

= Relationship tells how the entities are related. Diamond or rhombus is used to represent the relationship.

= Ex :- Teacher works for department.



iv) Relationship Model :-

⇒ This model was initially described by Edgar F. Codd in 1969.

⇒ Most widely used model by commercial data processing applications.

⇒ It uses collection of tables for representing the data and the relationship among those data.

⇒ Data is stored in tables called Relations.

⇒ Each table is a group of column and rows where column represents attributes of an entity and rows represented records or tuples.

Attributes or Field –

= Each column in a relation is called an attribute. The values of an attribute should be from the same domain.

= Eg : Student_id, Student_name, Student_age.

Tuple or record –

= Each row in the relation called tuple.

= A tuple defines a collection of attribute values. So each row in a relation contains values.

= Each row has all the information about any specific individual.

Student_Id	Name	Age
1	A	22
2	B	33
3	C	14

Student_Id	Name	Teacher
1	Java	Mr. X
2	C++	Mr. Y
3	C++	Mr. Z

INTRODUCTION TO ENTITY-RELATIONSHIP MODEL :-

⇒ ER Model stands for Entity-Relationship model.

⇒ This model is used to define the data elements and relationship for a specified system.

⇒ It develops a conceptual design for database.

⇒ It works around real-world entities and the relationship among them.

ER Diagram :-

It is a graphical representation that depicts the relationships among people, objects, places, concepts or events within an information technology (IT) system.

Why ER diagram are used :-

⇒ ER diagram are used to represent the ER model in a database, which makes them easy to be converted into relations (tables).

⇒ ER diagram maps well to the relational model i.e an ER model can be easily transformed into relational table.

⇒ These diagrams are very easy to understand and easy to create even for a naive users.

⇒ It gives a standard solution for visualization of the data logically.

* ER modeling construction :-

⇒ These 3 are most important for ER model / basic of ER Model

i) Entity

ii) Attribute

iii) Relationship.

1) ENTITY :-

⇒ An 'Entity' may be a thing or object with a physical existence - a particular person, car, house or employee - or it may be an object with a conceptual existence - a company, a job or a university course.

⇒ Entity can be anything that has an independent existance and about which we collect data. It is known as entity type.

⇒ Entities are represented by means of rectangles.

Student	Teacher

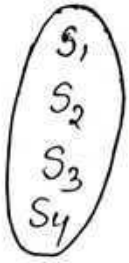
⇒ Entities have attributes that give them identity.

⇒ Entities became table in relational model.

Attributes → column

Entity Set :

An entity set is a collection of similar type of entities, that share some attributes.



→ Entity Set

Entity Type :

The collection of entity that have attributes is called Entity Type.

Question :-

1) What is ER-Model ?
2) What is ER-Diagram ?
3) Why ER-Diagram are used ?
4) Describe Entity with example.

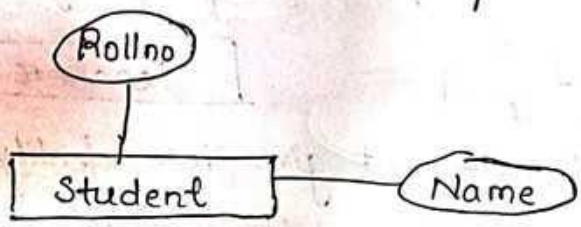
ii) Attribute :-

⇒ Attribute are the properties that define the entity type.

⇒ For example, Roll_No, Name, DOB, Age, Address and Mobile_No are the attributes that define entity type Student.

⇒ For each attributes there is a set of permitted values, called domain of that attribute.

⇒ In ER diagram, the attribute is represented in oval or ellipse.



⇒ Types of Attribute :-

i) Key Attribute
ii) Composite Attribute
iii) Multivalued Attribute
iv) Derived Attribute
v) Simple Attribute

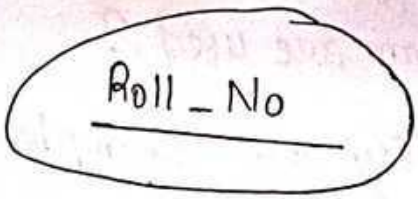
i) Key Attributes :-

⇒ Key Attributes is the attribute which uniquely identifies each entity in the entity set is called Key attribute.

⇒ It represents a primary key.

⇒ Key attribute is represented by an ellipse with underlying lines.

⇒ Ex: Roll_No / Roll_No



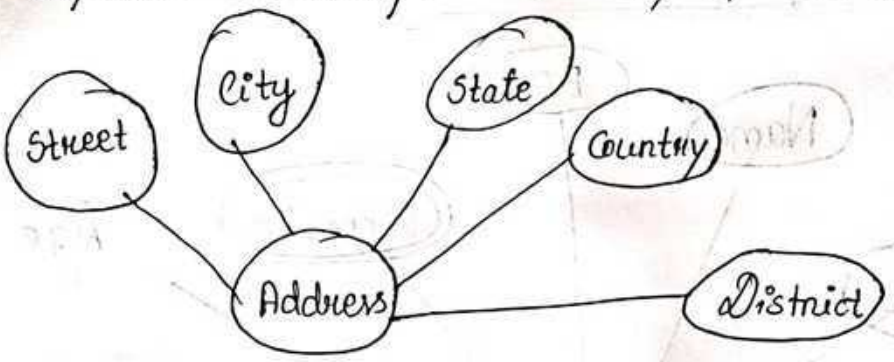
ii) Composite Attributes :-

⇒ The attribute which are made of more than one simple attribute wh is known as composite attributes.

⇒ A composite attributes is divided in a tree like structure.

⇒ Example : A student's complete name may have first_name and last_name

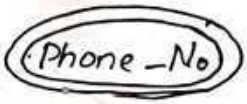
= An adress may have — street, city, state, country, pin code etc



iii) Multi-valued Attributes :-

⇒ Multi-valued attributes may contain more than one values.

⇒ Represented by double-ellipse.



⇒ Ex : A ~~phone~~ person can have more than one phone number, email etc.

iv) Derived Attributes :-

⇒ Derived Attributes are the attributes that do-not-exist in physical database but their values are derived from other attributes present in the database.

⇒ Derived attributes are depicted by dashed ellipshed

For Example-

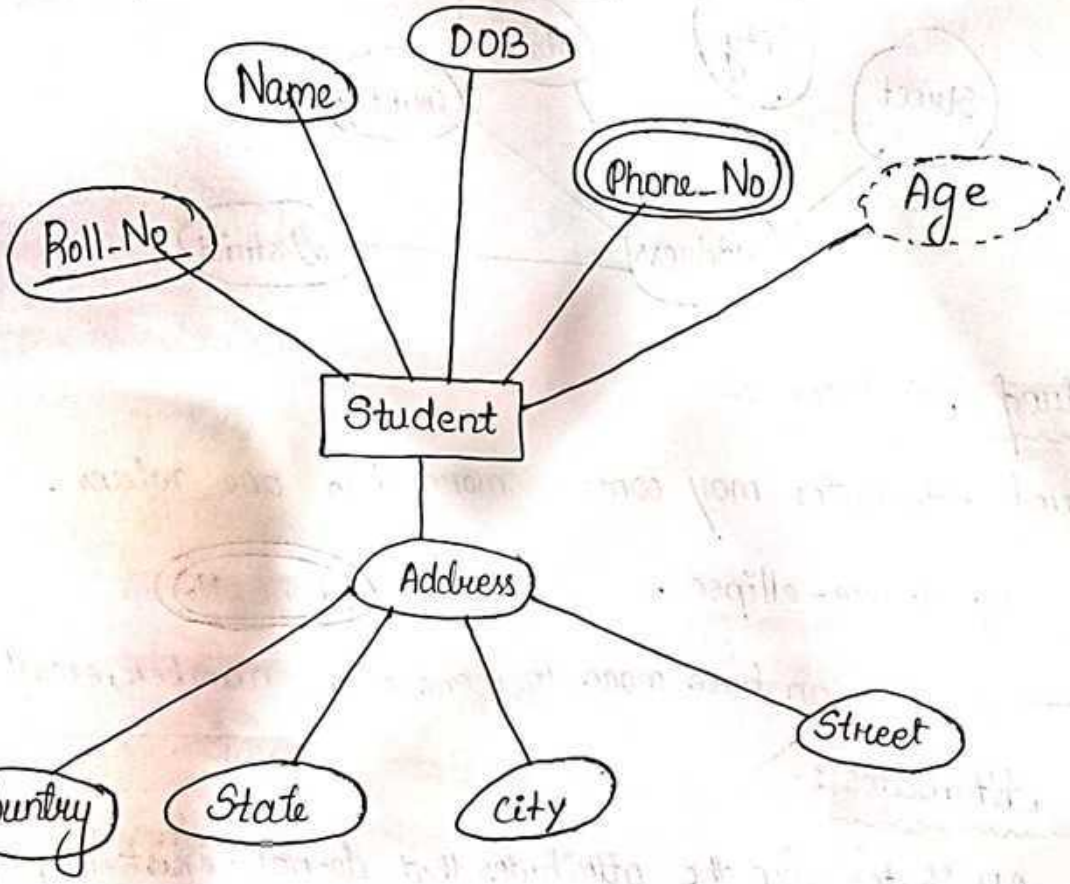
Age can be derived from date-of-birth.

v) Simple Attributes :-

⇒ Simple Attribute are atomic values, which can't be divided further

⇒ Ex :- Gender, D-O-B

ER Diagram Representation :-

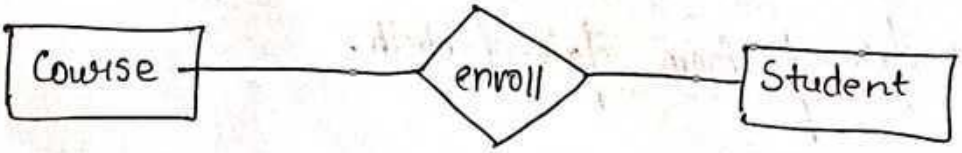


Relationship :-

⇒ When an entity is related to another entity they are said to have a relationship.

⇒ A relationship is an association among entities.

⇒ Example: An employee works_at department. A student enroll in a Course.

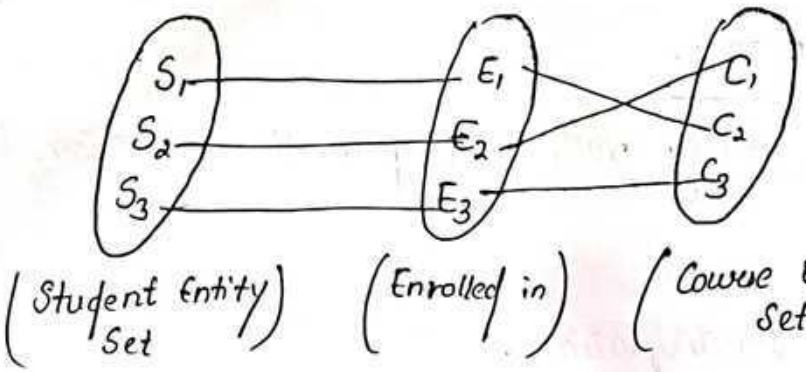


⇒ Relationship are represented by Diamond Set.

Relationship Set :-

⇒ A set of relationship of similar type is called relationship set.

⇒ The following relationship sets Enroll (E1, E2, E3) depicts.



Degree of Relationship :-

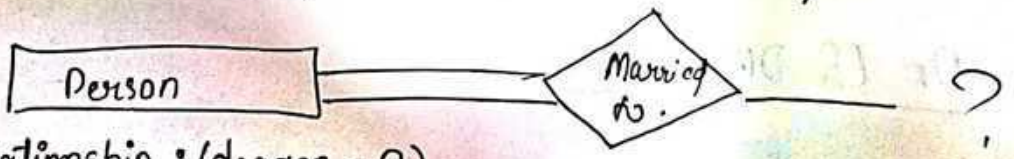
⇒ The number of different entity sets participating in a relationship set is called a degree of a relationship set.

⇒ = Unary
= Binary
= N-ary

i) Unary Relationship : (degree = 1)

= An unary relationship is only one entity participate in a relationship the relationship is called as unary relationship.

Ex: One person is married to single person only.



ii) Binary Relationship : (degree = 2)

A binary relationship is when two entities participating in a relationship and is the most common relationship degree.

Ex: Student is enrolled in course.

iii) N-ary Relationship (degree = n)

When there are n entities set participating in a relation, the relationship is called as n-ary relationship.

PARTICIPATION CONSTRAINTS :-

=> It is applied on the entity participating in the relationship set.

=> It has two types –

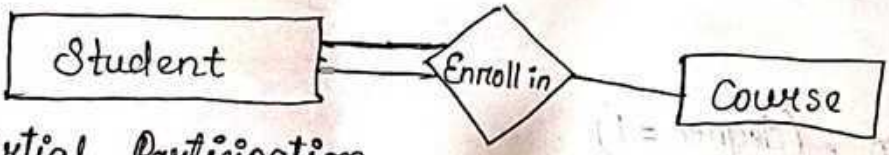
i) Total Participation.

ii) Partial Participation.

i) Total Participation –

=> Each entity in the entity set must participate in the relationship.

For ex – Each student must enroll in a course, the participation of student will be total participation.



ii) Partial Participation –

=> The entity in the entity set may or maynot participate in the relationship.

for ex – If some course aren't enrolled by any of the student, the participation of course is partial.

SYMBOLS OF ER DIAGRAM :-

i) Entity.

ii) Attributes

Keys in DBMS :-

⇒ A key is an attribute or set of attributes that uniquely identifies any record (or tuple) from the table.

⇒ Key is used to uniquely identify any record or row of data from the table.

⇒ It is also used to establish to identify relationship between table.

Types of Keys in DBMS :-

1) Super Key
2) Candidate Key
3) Primary Key
4) Alternate Key
5) Foreign Key
6) Composite Key

1) Super Key –

⇒ A superkey is a combination of all possible attribute that can uniquely identifies the rows (or tuples) in the given relation.

⇒ A superkey is superset of candidate key.

⇒ A table can have many super key.

⇒ A super key may have additional attribute that are not needed for unique identity.

2) Candidate Key :-

⇒ A candidate key is an attribute or set of attribute which can uniquely identify a tuple.

⇒ A candidate key is a minimal superkey ; or a super key with no redundant attributes.

⇒ A super key whose proper subset is not a super key.

⇒ Candidate keys are defined as distinct set of attributes from which primary key can be selected.

⇒ Candidate keys are not allowed to have NULL values.

⇒ Ex :- $S_1 = \{1, 2, 3\}$.

$S_2 = \{1, 2\}$.

$S_2 \subset S_2$ if $S_2 \subseteq S_1$ and $S1 \not\subseteq S_2$.

A	B	C
1	1	1
2	1	2
3	2	1
4	2	2

Superkey = A, AB, AC, ABC, BC.

Candidate Key = {A, B, C}

3) Primary Key :-

⇒ A primary key is one of the candidate key chosen by the database designer to uniquely identify the tuple in the relation.

4) Alternate Keys :-

⇒ Out of all candidate keys, only one gets selected as primary key, remaining keys are known as Alternate Keys.

⇒ In the Employee table :-

Emp_Id is the best suited for the primary key, rest of the attributes like aadhar_no and Email_Id are considered as alternate key.

5) Foreign Keys :-

⇒ An attribute or set of attribute in one table that refers to the primary key in another table.

⇒ It is used to link two tables together.

⇒ The purpose of the foreign key is to ensure referential integrity of data.

6) Composite Key :-

= A key that has more than one attributes is known as composite key.

= It is also known as "Compound key".

= Example : { customer_id, product_id }

Student_Id	Name	Course	Marks
201	S	SA	85
202	H	SD	90
201	S	QA	70

Eid	Name	Aadhar	Email	Dept (FK)
				1
				2
				3

Dept_Id (PK)	Dept_Name
1	Sales
2	Marketing
3	HR

Extended ER-Model :-

⇒ The EER model is a conceptual (or semantic) data model, capable of describing the data requirements for a new information system in a direct and easy to understand graphical notation.

⇒ As the complexity of database increases day by day, we need to use EER model to represent it.

⇒ Using ER-model for bigger data creates a lot of complexity while designing a database model.

⇒ So, in order to minimize the complexity, generalization, specialization and aggregation were introduced in the ER model.

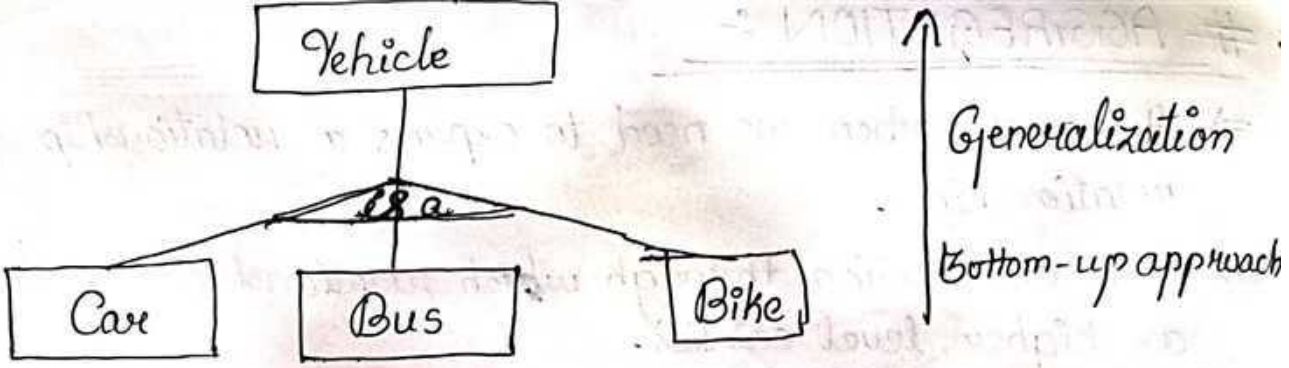
⇒ And these were used for data abstraction in which abstraction mechanism is used to hide details of a set of objects. Some of the terms were added to the enhanced ER-model where new concept are added. These concepts are generalization, specialization and aggregation inheritance.

GENERALISATION :-

⇒ It is the process of extracting common properties from a set of entities and create a generalized entity from it.

⇒ Generalisation is a "bottom-up approach" in which two or more entities can be combined to form a higher level entity if they have some attributes in common.

⇒ It is used to emphasized the similarities among lower-level entity set and to hide differences in the schema.



SPECIALIZATION :-

= It is opposite of Generalisation.

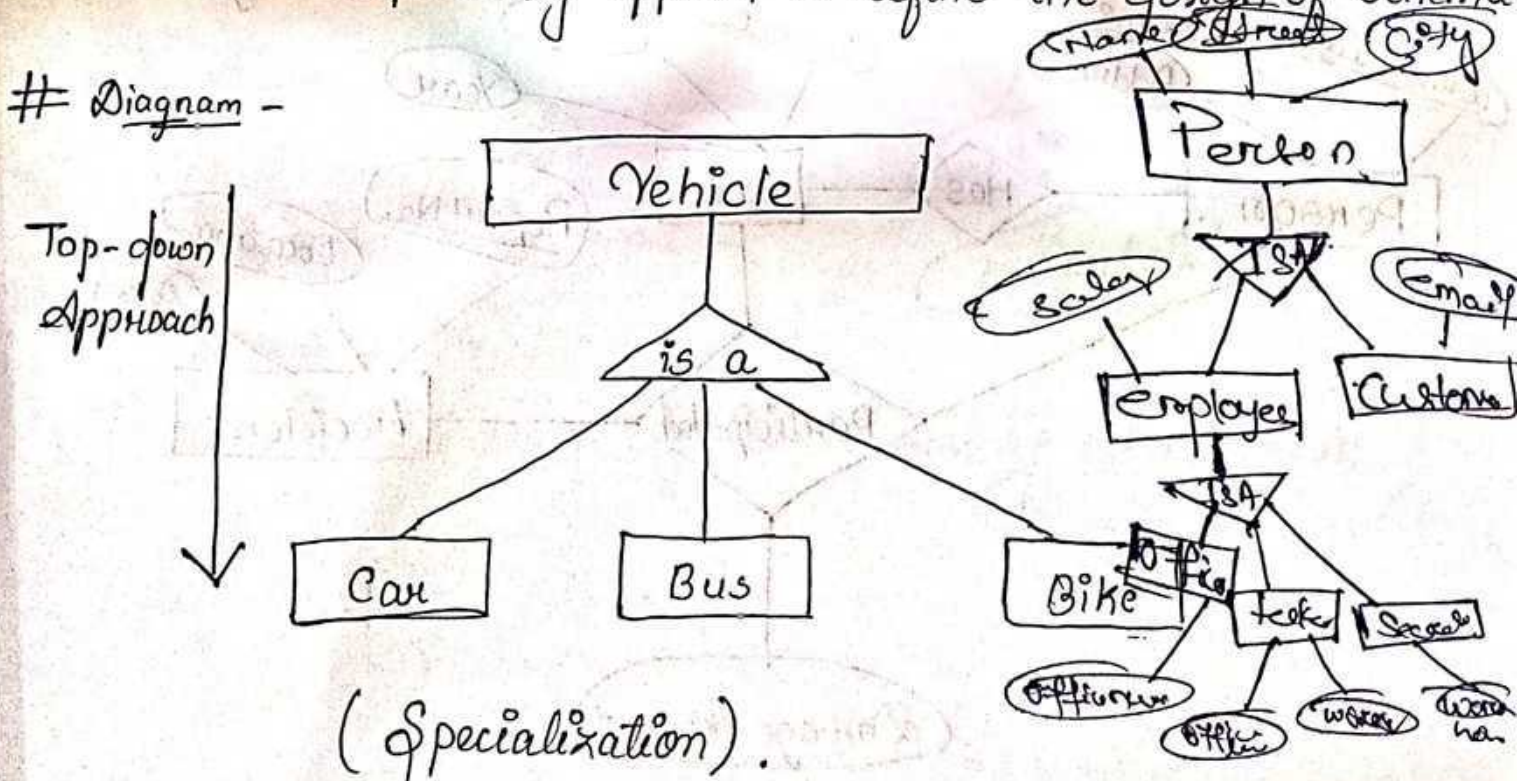
= In specialization, an entity is broken down into sub-entiti based on their characteristics.

= Specialization is "Top-down approach" where higher level entity is specialized into two or more lower level entities

= It is used to identify the subset of an entity set that shares some distinguishing characteristics.

= It can be repeatedly applied to refine the design of schema

Diagnam -



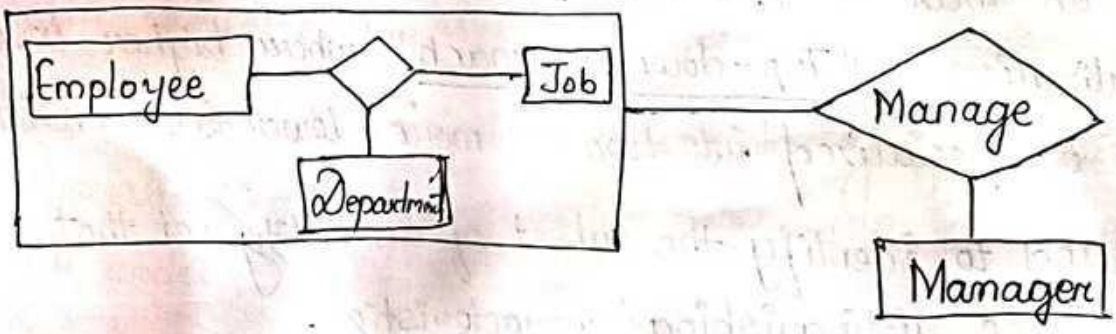
(Specialization).

AGGREGATION :-

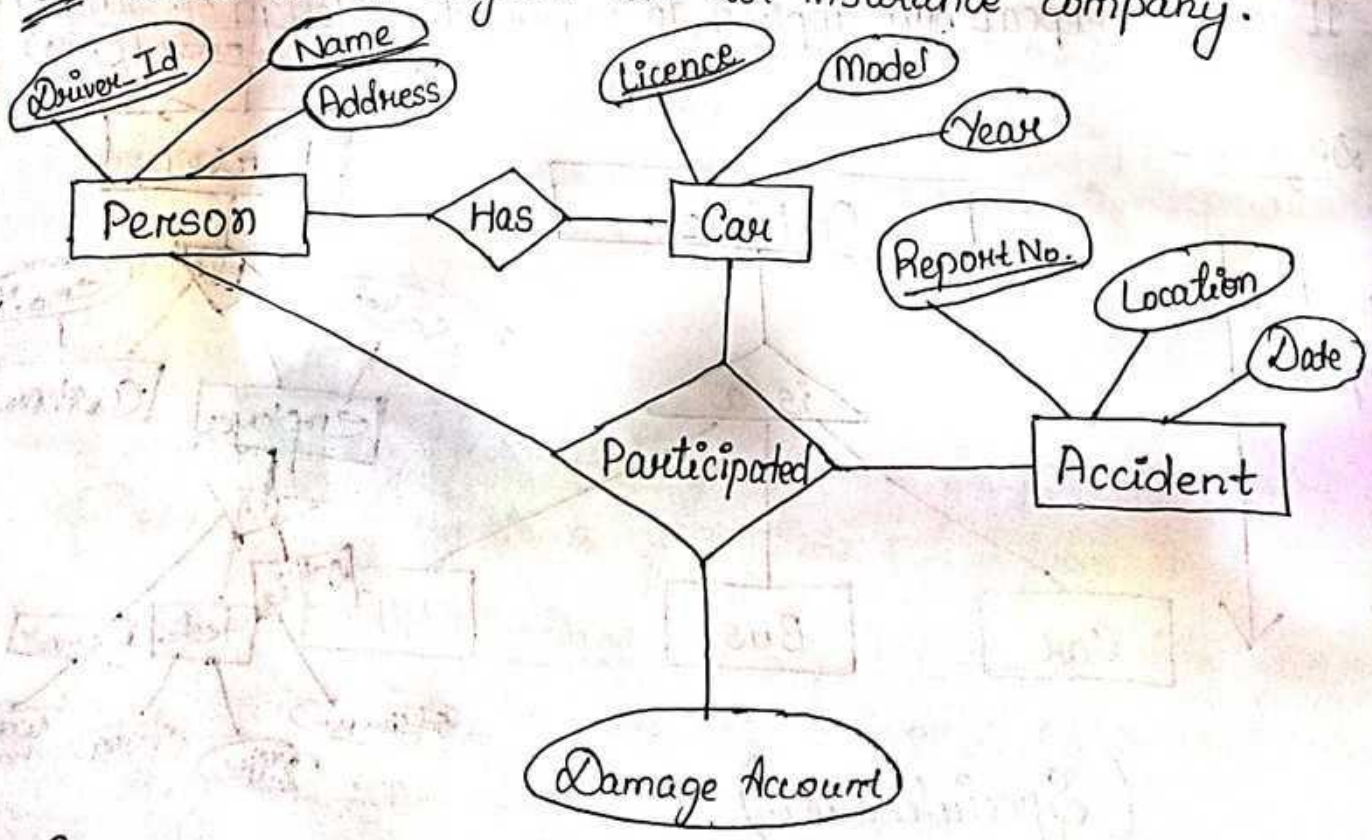
⇒ It is used when we need to express a relationship among relationships.

⇒ It is abstraction through which relationships are treated as higher level entities.

⇒ Aggregation is a process when a relationship between two entities is considered as a single entity and again this single entity has a relationship with another entity.

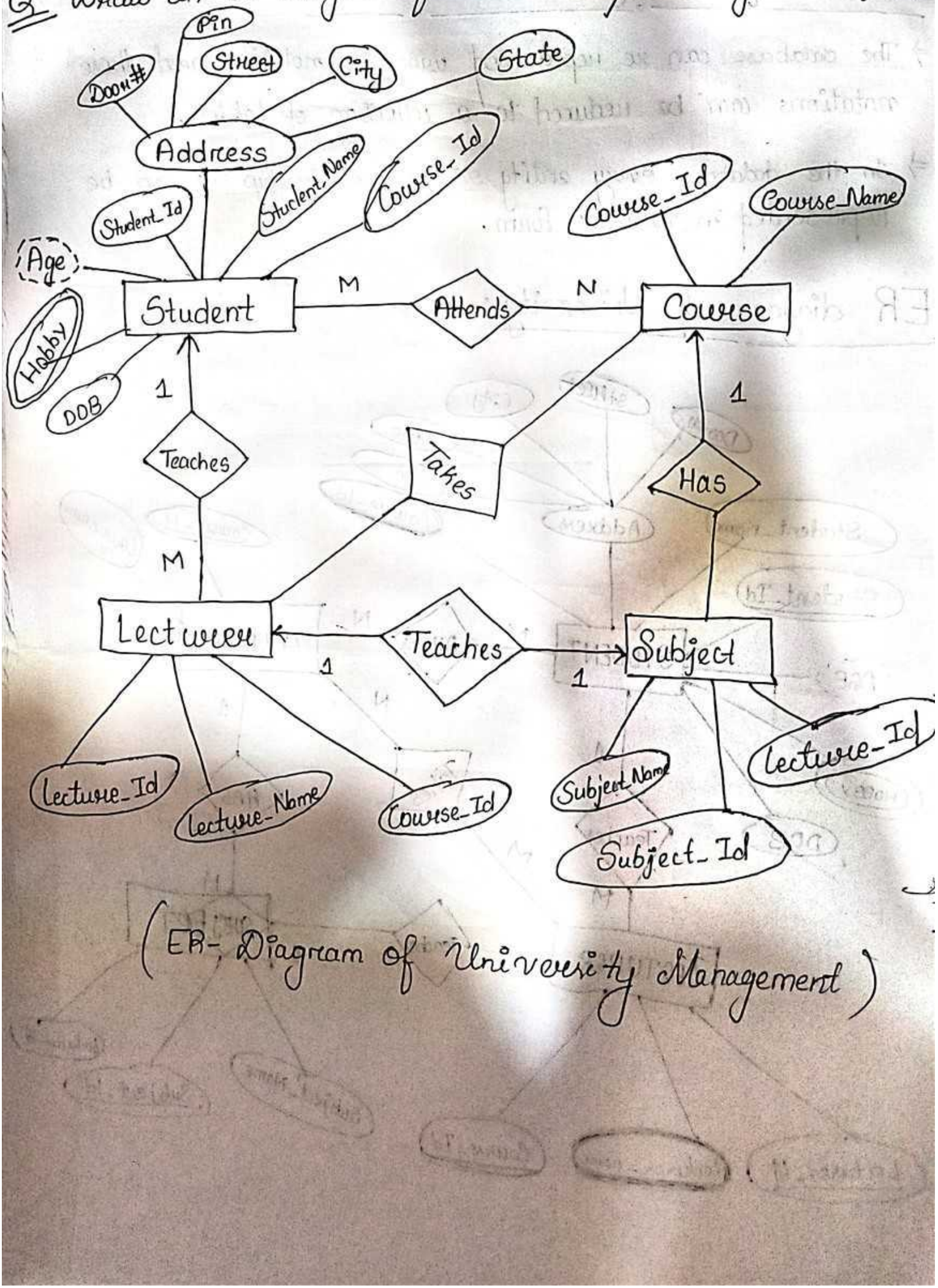


Q1 Draw an ER-diagram for car insurance company.



(ER Diagram of car Insurance)

Q Draw an ER-diagram for University Management system.



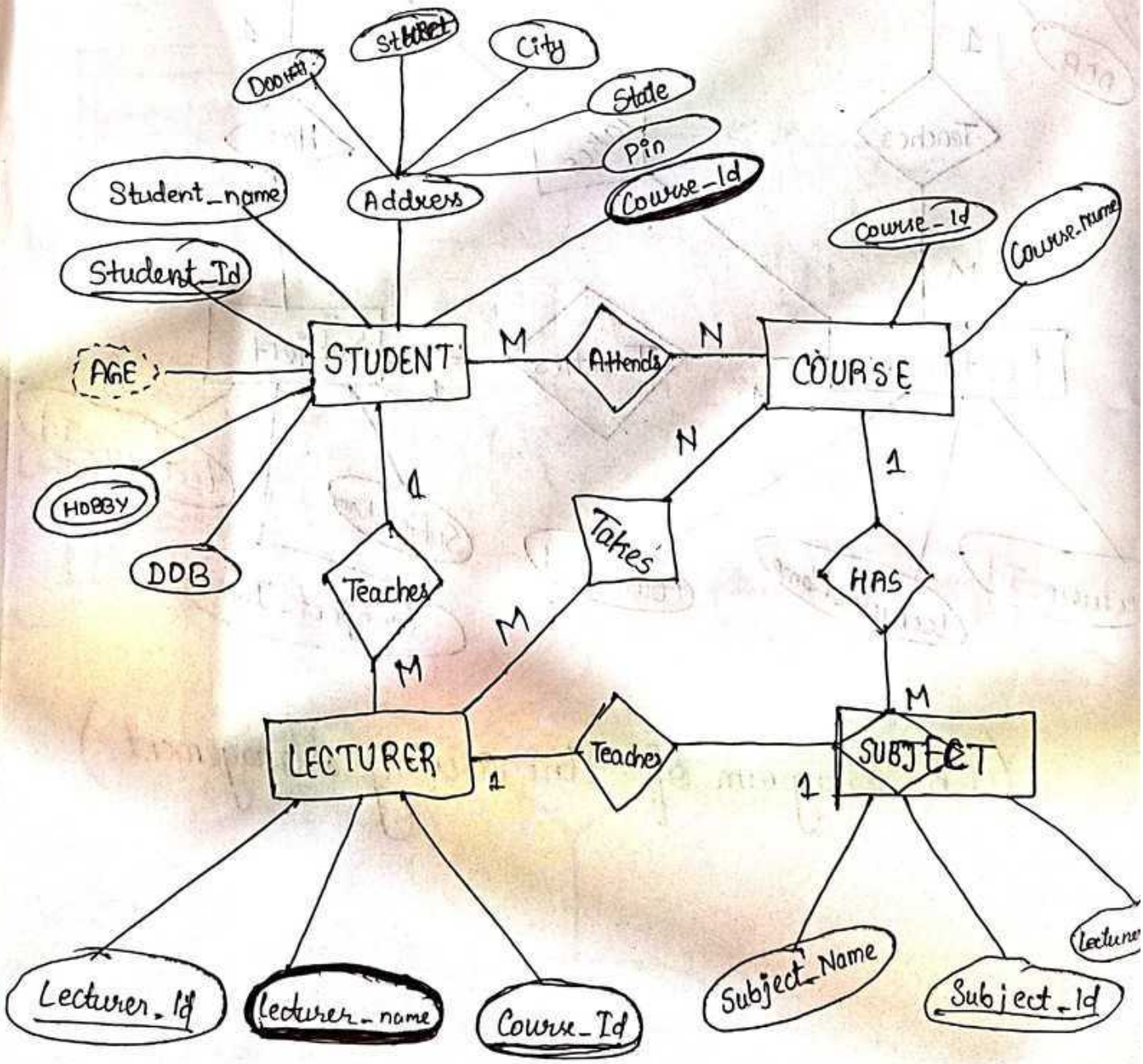
(ER- Diagram of University Management)

Reduction of ER diagram to Table/Schema :-

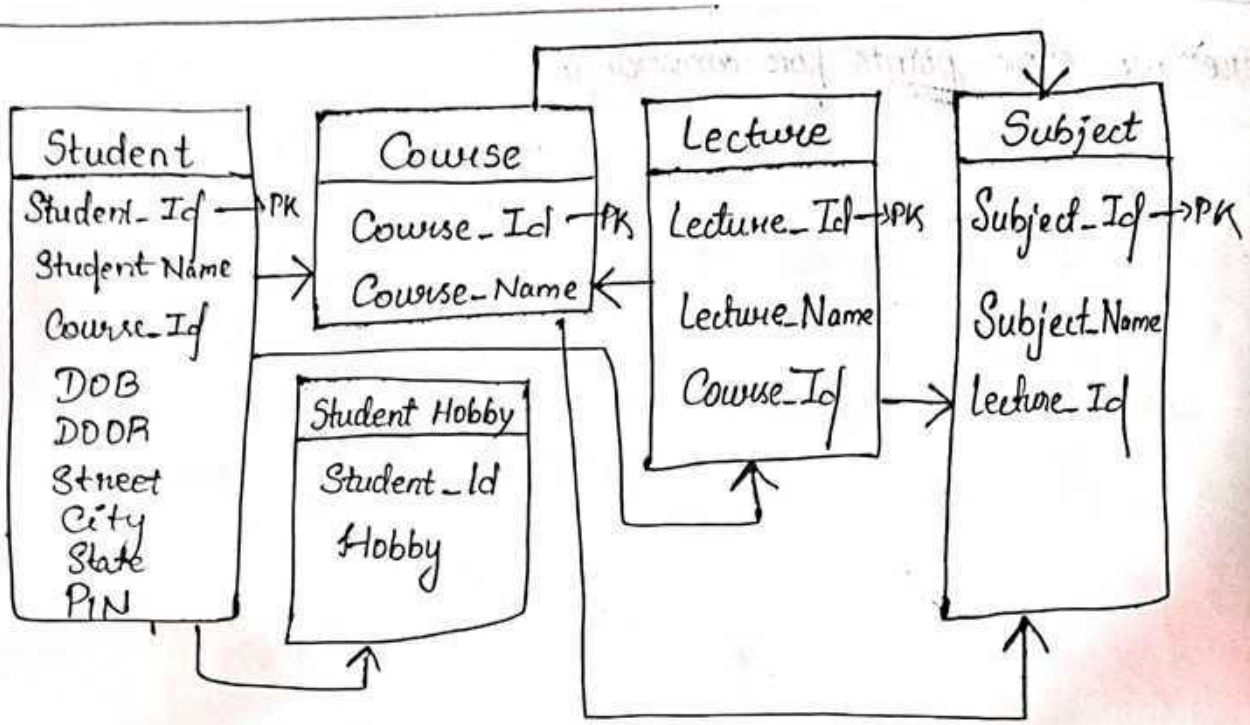
⇒ The database can be represented using the notations and these notations can be reduced to a collection of tables.

⇒ In the database every entity set or relationship set can be represented in tabular form.

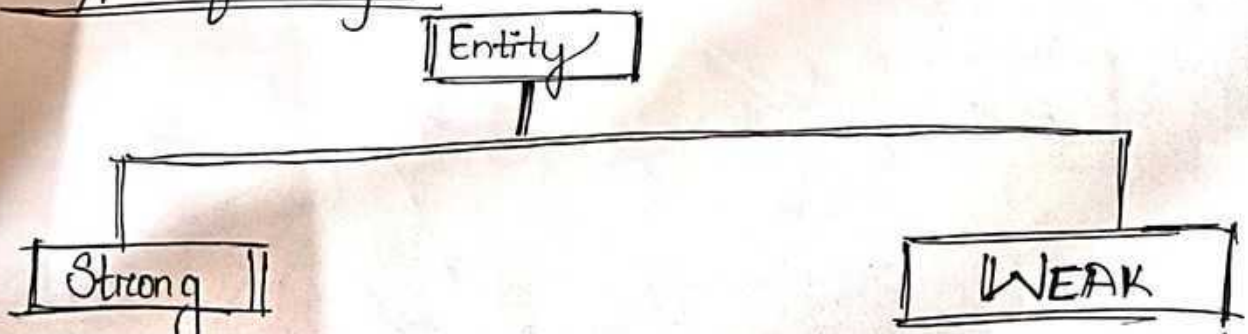
ER diagram of University :-



ER model to Relational model :-



Types of Entity :-



= It is a type of entity that has a key attribute

= It doesn't depend on other entity in the schema.

= It has a primary key, that helps in identifying it uniquely and it is represented by a rectangle. These are called Strong Entity Types.

= The weak entity doesn't have a sufficient attribute to form a primary key ie. Weak entity doesn't have a primary key.

= It is dependent on a strong entity to ensures it existance.

= Weak entity is represented by double rectangle.

Extended ER Model :-

⇒ The E-ER Model is a conceptual data model, capable of describing the data requirements for a new information system in a direct and easy to understand way by using the graphical notation.

DBMS Users :-

= The person who involved in the design, use and maintainance of any database are called as DBMS users.

= It can be classified into two types :-

i) Actors on the scene.
ii) Actors behind the scene.

i) Actors on the scene :-

= The people whose job involve the day to day use of database are called actors on the scene.

= This again categorized as :

i) Database Administrator (DBA)
ii) Database designer
iii) End user
   - → Casual User
   - → Naive user
   - → Sophisticated user
   - → Standalone user
iv) System Analyst
v) Application Programmer

DataBase Administraton (DBA):-

= The database administraton are the most important type of database user in DBMS.

= The DBA is an individual or team of users who defines the database schema and takes charge of controlling various levels of database within the organization.

= They are helping to design the overall structure of a database including layouts, functionalities, work flows etc.

= DBA can grant or revoke authorization permissions to all other users at any point of time. In order to access the database DBA had to provide logic credentials to all other users when required.

= The DBAs are responsible for providing security to the data base by restricting unauthorized user from accessing the database.

= The DBA have to update the database in terms of technology, functionality and workflow in order to meet the future requirement for making the database ready.

= DBAs are responsible to keep a check on data integrity and routine maintainance of the database.

= It monitors the back-up and recovery of the database and provides technical support.