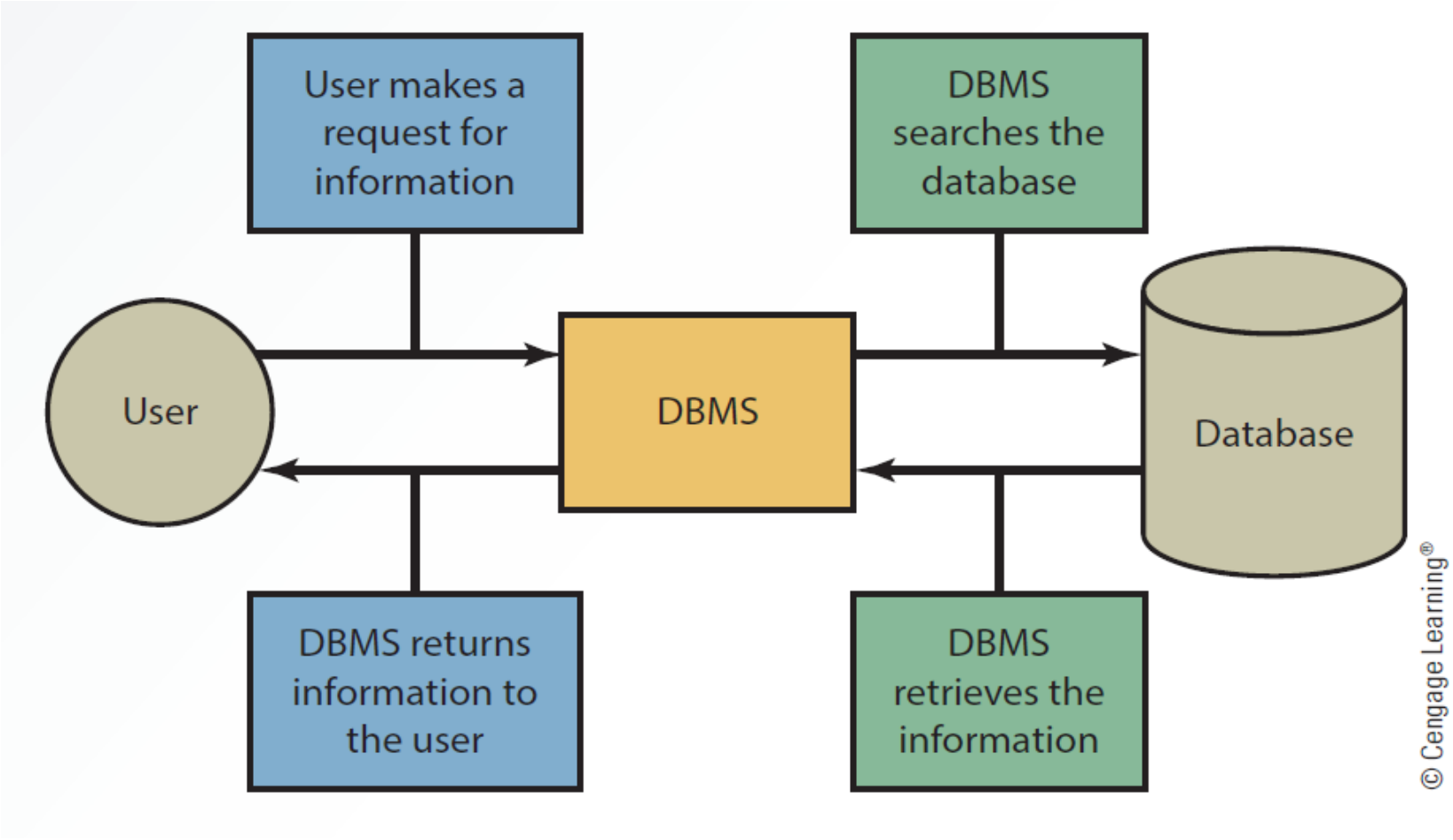
Unit I – INTRODUCTION OF DATABASE and SQL

Prof. Sonali Deshpande

- Database Concepts, Three-schema architecture of a database, Data Models : ER model, Relational Model, ER to Table Conversion.
- Relational Algebra: Select, Project, Union, Set difference, Join.
- SQL-Characteristics and advantages, SQL Data Types and Literals, DDL, DML, DCL, TCL, Views, Indexes.
- PLSQL : Concept of Stored Procedures, Functions, Cursors, Triggers.
- NOSQL- MongoDB CRUD Operations, SQL VsNoSQL Databases, Introduction of databricks

- **Database**
  - Collection of related data that is stored in a central location or in multiple locations
- **Data hierarchy**: Structure and organization of data involving fields, records, and files
- **Database management system (DBMS)**
  - Software for creating, storing, maintaining, and accessing database files
  - Makes using databases more efficient

Interaction Between the User, DBMS and Database



Disadvantage of File-oriented system

Data Redundancy:

- It is possible that the same information may be duplicated in different files. this leads to data redundancy results in memory wastage.

Data Inconsistency:

- Because of data redundancy,it is possible that data may not be in consistent state.

Difficulty in Accessing Data:

- Accessing data is not convenient and efficient in file processing system.

Limited Data Sharing:

- Data are scattered in various files.also different files may have different formats and these files may be stored in different folders may be of different departments.
- So, due to this data isolation, it is difficult to share data among different applications.

.

Integrity Problems:

- Data integrity means that the data contained in the database in both correct and consistent. for this purpose the data stored in database must satisfy correct and constraints.

Atomicity Problems:

- Any operation on database must be atomic.
- this means, it must happen in its **entirely** or not at all.

Concurrent Access Anomalies:

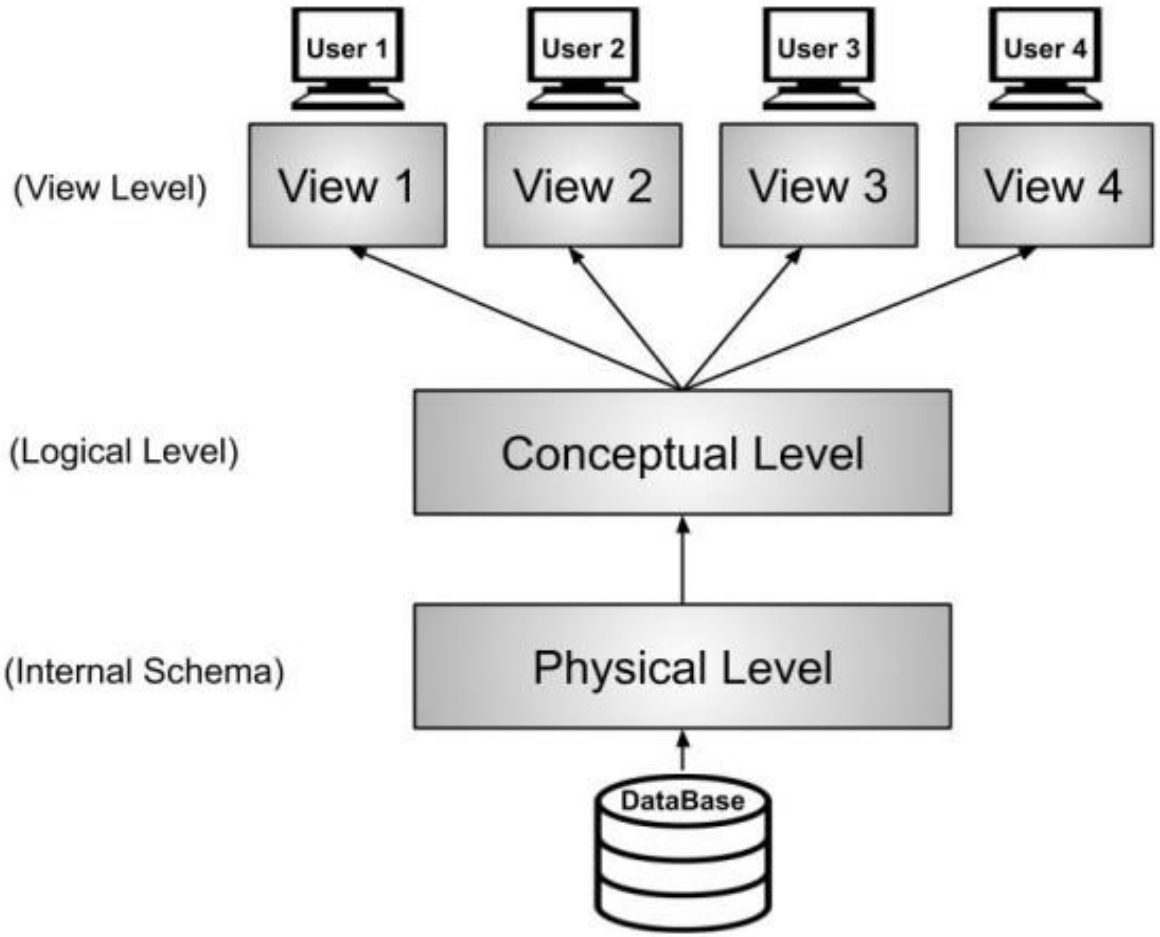
- Multiple users are allowed to access data simultaneously. this is for the sake of better performance and faster response.

Security Problems:

- Database should be accessible to users in limited way.
- Each user should be allowed to access data concerning his requirements only

Level of Abstraction

- There are mainly three levels of data abstraction:
- Internal Level: Actual PHYSICAL storage structure and access paths.
- Conceptual or Logical Level: Structure and constraints for the entire database
- External or View level: Describes various user views



Levels of Data Abstraction

- **Example**: Let's say we are storing customer information in a customer table. At **physical level** these records can be described as blocks of storage (bytes, gigabytes, terabytes etc.) in memory. These details are often hidden from the programmers.
- Gmail account

DBMS Three Level ArchitectureDiagram

- This architecture has three levels:
  1. External level
  2. Conceptual level
  3. Internal level

1. External level

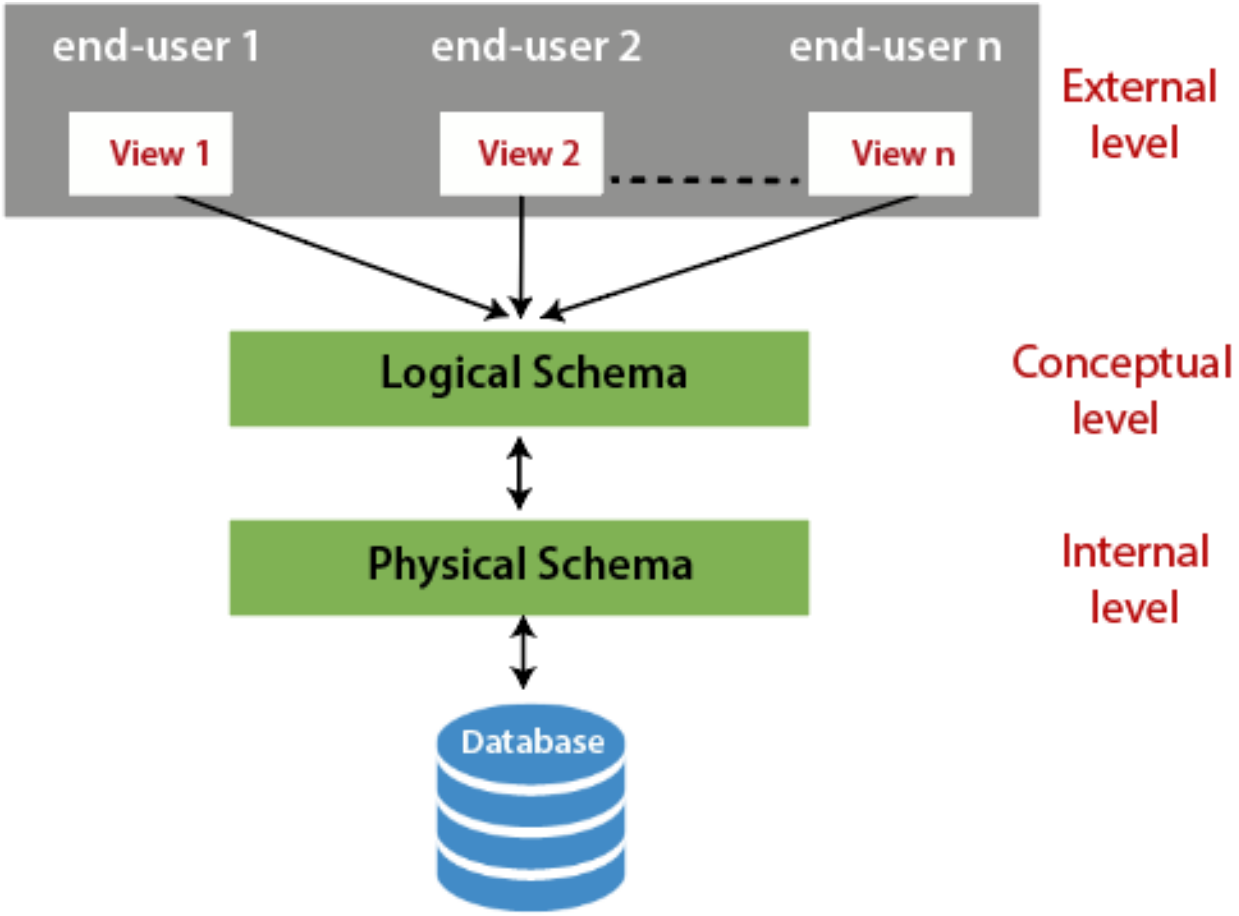
- It is also called **view level**. The reason this level is called "view" is because several users can view their desired data from this level which is internally fetched from database with the help of conceptual and internal level mapping.

2. Conceptual level

- It is also called **logical level**. The whole design of the database such as relationship among data, schema of data etc. are described in this level.

3. Internal level

- This level is also known as physical level. This level describes how the data is actually stored in the storage devices. This level is also responsible for allocating space to the data. This is the lowest level of the architecture.



DBMS Architecture

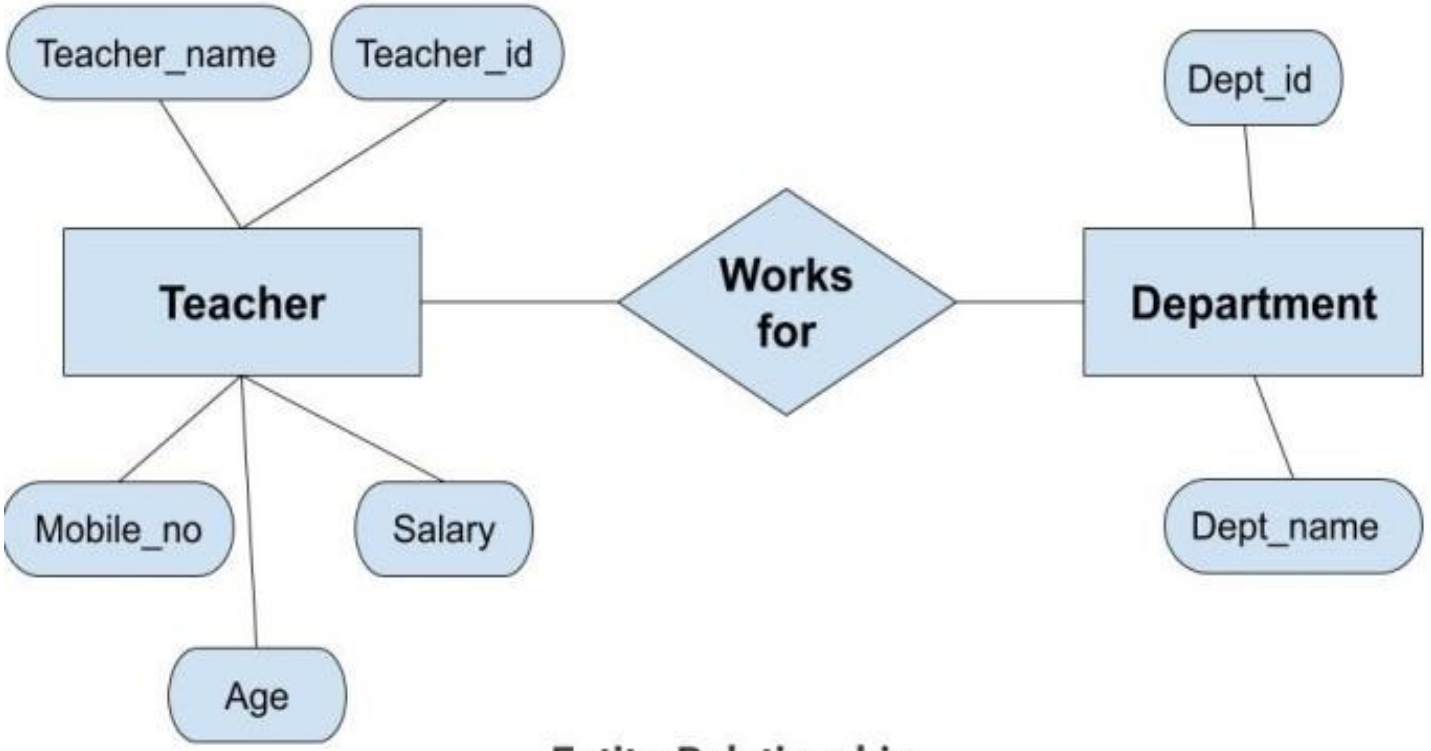
Data Models

- Data Model gives us an idea that how the final system will look like after its complete implementation.
- Data Models are used to show how data is stored, connected, accessed and updated in the database management system.
- Data models define how the logical structure of a database is modeled.
- It is classified into 3 types:

1. Conceptual Data Models -- ER Model
2. Representational Data Models – Relational Model
3. Physical Data Models

Entity-Relationship Model

- ER model stands for an Entity-Relationship model. It is a high-level data model. This model is used to define the data elements and relationship for a specified system.
- It develops a conceptual design for the database.
- We use the ER diagram as a visual tool to represent an ER Model. ER diagram has the following three components:
- ***Entities:*** Entity is a real-world thing. It can be a person, place, or even a concept. *Example:* Teachers, Students, Course, Building, Department, etc are some of the entities of a School Management System.
- ***Attributes:*** An entity contains a real-world property called attribute. This is the characteristics of that attribute. *Example:* The entity teacher has the property like teacher id, salary, age, etc.
- ***Relationship:*** Relationship tells how two attributes are related. *Example:* Teacher works for a department.



Entity-Relationship Model

Advantages of ER Model

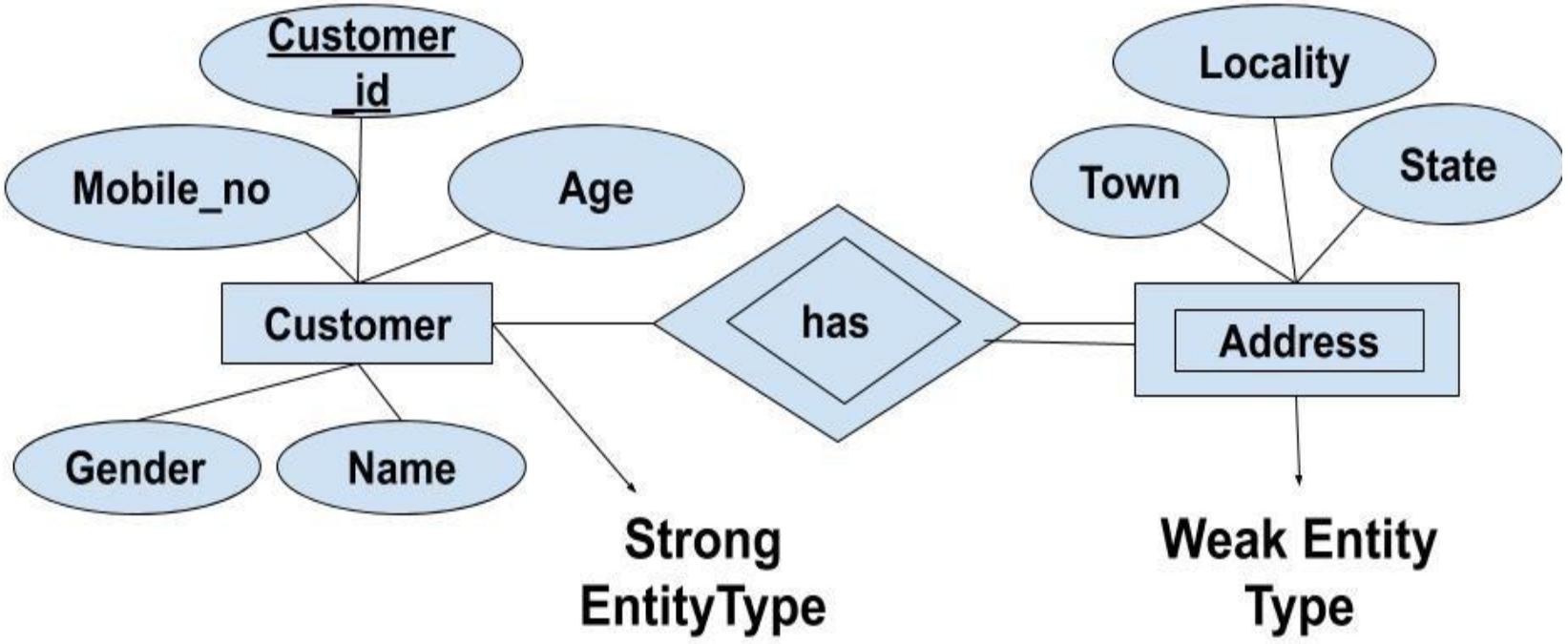
- ***Simple:*** Conceptually ER Model is very easy to build. If we know the relationship between the attributes and the entities we can easily build the ER Diagram for the model.
- ***Effective Communication Tool***: This model is used widely by the database designers for communicating their ideas.
- ***Easy Conversion to any Model***: This model maps well to the relational model and can be easily converted relational model by converting the ER model to the table. This model can also be converted to any other model like network model, hierarchical model etc.

Types of Entity type

- Strong Entity Type
- Weak Entity Type

- ***Strong Entity Type:*** Strong entity are those entity types which has a key attribute. The primary key helps in identifying each entity uniquely
- ***Weak Entity Type***: Weak entity type doesn't have a key attribute. Weak entity type can't be identified on its own. It depends upon some other strong entity for its distinct identity.

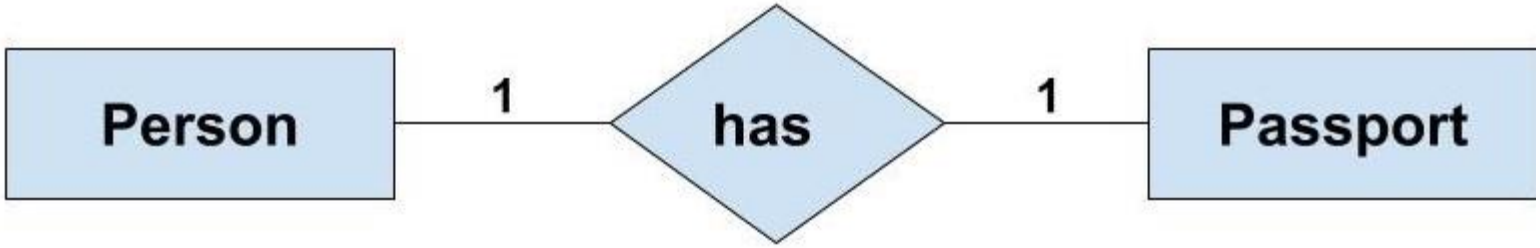
- If we have two tables of Customer and Address.
- Here we cannot identify the address uniquely as there can be many customers from the same locality.
- So, for this, we need an attribute of Strong Entity Type i.e 'Customer' here to uniquely identify entities of 'Address' Entity Type.



Relationship in DBMS

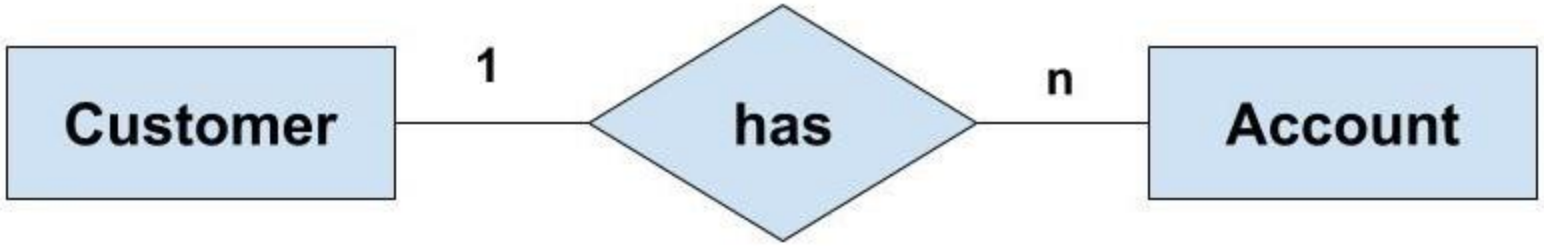
- Any association between two entity types is called a relationship.
- One-to-One Relationship
- One-to-Many or Many-to-One Relationship
- Many-to-Many Relationship
- **One-to-One Relationship**

Such a relationship exists when each record of one table is related to only one record of the other table.



- **One-to-Many or Many-to-One Relationship**

Such a relationship exists when each record of one table can be related to one or more than one record of the other table



If there are two entity type 'Customer' and 'Account' then each 'Customer' can have more than one 'Account' but each 'Account' is held by only one 'Customer'. In this example, we can say that each Customer is associated with many Account. So, it is a one-to-many relationship. But, if we see it the other way i.e many Account is associated with one Customer then we can say that it is a many-to-one relationship.

Many-to-Many Relationship

- If there are two entity type 'Customer' and 'Product' then each customer can buy more than one product and a product can be bought by many different customers.

Types of Participation Constraints-

- Total participation
- Partial participation
- **1.Total Participation-**
- It specifies that each entity in the entity set must compulsorily participate in at least one relationship instance in that relationship set.
- That is why, it is also called as **mandatory participation.**
- Total participation is represented using a double line between the entity set and relationship set.

- Double line between the entity set "Student" and relationship set "Enrolled in" signifies total participation.
- It specifies that each student must be enrolled in at least one course.

- **2. Partial Participation-**
- It specifies that each entity in the entity set may or may not participate in the relationship instance in that relationship set.
- That is why, it is also called as **optional participation.**
- Partial participation is represented using a single line between the entity set and relationship set.

- Single line between the entity set "Course" and relationship set "Enrolled in" signifies partial participation.
- It specifies that there might exist some courses for which no enrollments are made.

Types of Attributes-

- Simple attributes
- Composite attributes
- Single valued attributes
- Multi valued attributes
- Derived attributes
- Key attributes

!!!!!!!!!!Task for students!!!!!!!!!!

Relational Model

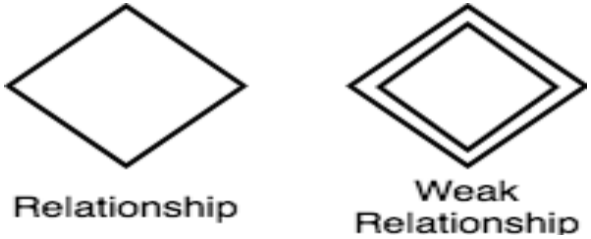
- Relational Model is the most widely used model.
- In this model, the data is maintained in the form of a two-dimensional table.
- All the information is stored in the form of row and columns. The basic structure of a relational model is tables.

Emp_id	Emp_name	Job_name	Salary	Mobile_no	Dep_id	Project_id
AfterA001	John	Engineer	100000	9111037890	2	99
AfterA002	Adam	Analyst	50000	9587569214	3	100
AfterA003	Kande	Manager	890000	7895212355	2	65

EMPLOYEE TABLE

- **Entity-**Simple rectangular box represents an Entity.

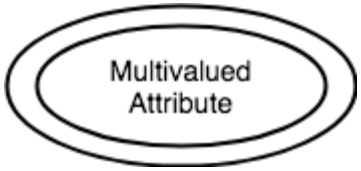
- **Relationships between Entities - Weak and Strong**



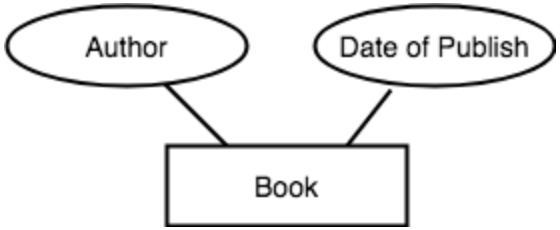
- **Weak Entity-**A weak Entity is represented using double rectangular boxes

- **Derived Attribute for any Entity**

- **Multivalued Attribute for any Entity**



- **Attributes for any Entity**



- **Composite Attribute for any Entity**

This image cannot currently be displayed.

SQL

- SQL stands for Structured Query Language
- SQL lets you access and manipulate databases

ER to Table conversion

- Entity type becomes a table.
- All single-valued attribute becomes a column for the table.
- A key attribute of the entity type represented by the primary key.
- The multivalued attribute is represented by a separate table.
- Composite attribute represented by components. (student address is a composite attribute, It contains CITY, PIN, DOOR#, STREET, and STATE, In the STUDENT table, these attributes can merge as an individual column.)
- Derived attributes are not considered in the table.

DDL-

- **Data Definition Language,** which deals with database schemas and descriptions, of how the data should reside in the database.
- CREATE- to create a database and its objects like (table, index, views, store procedure, function, and triggers)
- ALTER - alters the structure of the existing database
- DROP - delete objects from the database
- TRUNCATE - remove all records from a table, including all spaces allocated for the records are removed
- COMMENT - add comments to the data dictionary
- RENAME - rename an object

DML

- DML is short name of **Data Manipulation Language** which deals with data manipulation.
- SELECT - retrieve data from a database
- INSERT - insert data into a table
- UPDATE - updates existing data within a table
- DELETE - Delete all records from a database table

DCL AND TCL

- DCL is short name of **Data Control Language** which includes commands such as GRANT and mostly concerned with rights, permissions and other controls of the database system.
- **GRANT** - allow users access privileges to the database

  Eg-GRANT SELECT, INSERT, DELETE, UPDATE ON Users TO ‘sonali'@'localhost;
- **REVOKE** - withdraw users access privileges given by using the GRANT command
- Eg-REVOKE privilege_name ON object_name FROM {user_name};

- TCL is short name of **Transaction Control Language** which deals with a transaction within a database.
- **COMMIT** - commits a Transaction
- **ROLLBACK** - rollback a transaction in case of any error occurs
- **SAVEPOINT** - to rollback the transaction making points within groups
- **SET TRANSACTION** - specify characteristics of the transaction

SQL Datatypes
SQL Server String Data Type

char(n)	It is a fixed width character string data type. Its size can be up to 8000 characters.
varchar(n)	It is a variable width character string data type. Its size can be up to 8000 characters.
varchar(max)	It is a variable width character string data types. Its size can be up to 1,073,741,824 characters.
text	It is a variable width character string data type. Its size can be up to 2GB of text data.
nchar	It is a fixed width Unicode string data type. Its size can be up to 4000 characters.
nvarchar	It is a variable width Unicode string data type. Its size can be up to 4000 characters.

MySQL Date and Time Data Types

DATE	It is used to specify date format YYYY-MM-DD. Its supported range is from '1000-01-01' to '9999-12-31'.
DATETIME(fsp)	It is used to specify date and time combination. Its format is YYYY-MM-DD hh:mm:ss. Its supported range is from '1000-01-01 00:00:00' to 9999-12-31 23:59:59'.
TIMESTAMP(fsp)	It is used to specify the timestamp. Its value is stored as the number of seconds since the Unix epoch('1970-01-01 00:00:00' UTC). Its format is YYYY-MM-DD hh:mm:ss. Its supported range is from '1970-01-01 00:00:01' UTC to '2038-01-09 03:14:07' UTC.
TIME(fsp)	It is used to specify the time format. Its format is hh:mm:ss. Its supported range is from '-838:59:59' to '838:59:59'
YEAR	It is used to specify a year in four-digit format. Values allowed in four digit format from 1901 to 2155, and 0000.

MySQL Numeric Data Types

BIT(Size)	It is used for a bit-value type. The number of bits per value is specified in size. Its size can be 1 to 64. The default value is 1.
INT(size)	It is used for the integer value. Its signed range varies from -2147483648 to 2147483647 and unsigned range varies from 0 to 4294967295. The size parameter specifies the max display width that is 255.
INTEGER(size)	It is equal to INT(size).
FLOAT(size, d)	It is used to specify a floating point number. Its size parameter specifies the total number of digits. The number of digits after the decimal point is specified by **d** parameter.
FLOAT(p)	It is used to specify a floating point number. MySQL used p parameter to determine whether to use FLOAT or DOUBLE. If p is between 0 to24, the data type becomes FLOAT (). If p is from 25 to 53, the data type becomes DOUBLE().
DOUBLE(size, d)	It is a normal size floating point number. Its size parameter specifies the total number of digits. The number of digits after the decimal is specified by d parameter.

Views

- Views in SQL are kind of virtual tables.
- A view also has rows and columns as they are in a real table in the database.
- We can create a view by selecting fields from one or more tables present in the database.
- A View can either have all the rows of a table or specific rows based on certain condition.

```
CREATE VIEW DetailsView AS
SELECT NAME, ADDRESS
FROM StudentDetails
WHERE S_ID < 5;
```

- To see the data in the View, we can query the view in the same manner as we query a table.

```
SELECT * FROM DetailsView;
```

INDEXES

- Indexes are **special lookup tables** that the database search engine can use to speed up data retrieval.
- Simply put, an index is a pointer to data in a table. An index in a database is very similar to an index in the back of a book.
- An index helps to speed up **SELECT** queries and **WHERE** clauses, but it slows down data input, with the **UPDATE** and the **INSERT** statements. Indexes can be created or dropped with no effect on the data

Relational Algebra

- Relational algebra is a procedural query language.
- It uses operators to perform queries.
- They accept relations as their input and yield relations as their output.
- The operations of relational algebra are as follows −
- Select
- Project
- Union
- Set different
- Cartesian product
- Rename

- **Selection (σ)**
  Selection is used to select required tuples of the relations.
- **Projection (π)**
  Projection is used to project required column data from a relation.
- **Union (U)**
  Union operation in relational algebra is same as union operation in set theory, only constraint is for union of two relation both relation must have same set of Attributes.
- **Set Difference (-)**
  Set Difference in relational algebra is same set difference operation as in set theory with the constraint that both relation should have same set of attributes.
- **Rename (ρ)**
  Rename is a unary operation used for renaming attributes of a relation.

- **Cross Product (X)**
- Cross product between two relations let say A and B, so cross product between A X B will results all the attributes of A followed by each attribute of B.

Joins

- The join operations is one of the most useful and commonly used operations to extract information from two or more relations.
- The join operations denoted by ⋈ and it is used to join two relations to form new relations on the basis of common attributes present in two relations.
- Types of Join

1. Cross Join
2. Natural Join
3. Conditional Join
4. Equi Join
5. Self Join
6. Outer Join ---- 6.1 left join
   6.2 Right Join
   6.3 Full Join

Concept of Stored Procedures

- A stored procedure is a prepared SQL code that you can save, so the code can be reused over and over again.
- You can also pass parameters to a stored procedure, so that the stored procedure can act based on the parameter value(s) that is passed.

```
delimiter /
create procedure display()
Begin
select * from stud;
end;
/
```

Execute it as - call display/

- **Parameterized Procedure----**

```
Delimiter /
Create procedure display2(rn int)
Begin
Select * from stud where rollno=rn;
End;
/
```

Execute it by - Call display2(1)/

Functions

- A function is same as a procedure except that it returns a value.

```
DELIMITER //
CREATE FUNCTION SQUARE (val INT)
RETURNS  INT
BEGIN
DECLARE  result INT;
SET result=0;
SET result=val * val;
RETURN result;
END; //
DELIMITER ;
```

Execute it by - *Select square(2);*

Triggers

- Triggers are stored programs, which are automatically executed or fired when some events occur. Triggers are, in fact, written to be executed in response to any of the events like DDL and DML.

```
delimiter  /
create trigger t1
 before insert
 on purchase
 for each row
 begin
 set new.total=new.qnty*new.price;
end;
  /
```

Execute it by - ***Select *from purchase /***

Cursors

- A **cursor** is a pointer to this context area. PL/SQL controls the context area through a cursor. A cursor holds the rows (one or more) returned by a SQL statement.
- You can name a cursor so that it could be referred to in a program to fetch and process the rows returned by the SQL statement, one at a time. There are two types of cursors −
- Implicit cursors- implicit cursors are automatically created by Oracle whenever an SQL statement is executed
- Explicit cursors- Explicit cursors are programmer-defined cursors for gaining more control over the **context area**(When Oracle process an SQL statement, it needs to allocate memory called **context area**)

NOSQL- MongoDB CRUD Operations

- MongoDB is NoSql database based on document model where data objects are stored as separate documents inside the collection.
- MongoDB groups data together through collections. Collection is grouping of documents that have similar or related purpose.
- Document is representation of single entity of data in the mongoDB database. Collection is made up of one or more related objects.
- MongoDB documents are similar to Javascript Object Notation objects but use a variant called Binary JSON (BSON) that accommodates more data types.

BSON data format

```
{
 [name:'sonali',
 age:30,
 address:
          {
            city:'Mumbai'
          },
hobbies:[
            {name:'reading'},
            { name:'driving'}
]
}
```

MongoDB GUI Tools

- NoSQLBooster
- Studio 3T
- MongoDB Compass
- Nucleon Database Master
- NoSQL Manager
- Mongo Management Studio
- MongoJS Query Analyzer
- Nosqlclient
- Cluster control

- Sql--- database--- table---rows---column---index
- Nosql---database---collection---documents—field---index

- **Collections**

  'Collections' in Mongo are equivalent to tables in relational databases. They can hold multiple JSON documents.

- **Documents**

  'Documents' are equivalent to records or rows of data in SQL.

- **Fields**

  'Fields' or attributes are similar to columns in a SQL table.

- **Schema**

  While Mongo is schema-less, SQL defines a schema via the table definition. A Mongoose 'schema' is a document data structure (or shape of the document) that is enforced via the application layer.

- **Models**

  'Models' are higher-order constructors that take a schema and create an instance of a document equivalent to records in a relational database.

CRUD Operation

- To create database command **use DATABASE_NAME.**
- To check your currently selected database use the command **db**
- To check your databases list, use the command **show dbs**.
- To display database, you need to insert at least one document into it.
- To drop the database

  db.dropDatabase() command is used to drop a existing database.
- To create collection syntax is

  db.createCollection(“nameofCollection”)
- To insert

  db.nameofCollection.insert({“key”:”value”})

  db.nameofCollection.insertMany([{‘key’:’value’}, {‘key’:’value’}])

  db.nameofCollection.insertOne({name:’aaaa’,age:30, address:{street:’abcd’,city:’mumbai’}})
- To check your collection in database use command

  show collections

- To display all documents in collection

  db.nameofCollection.find()

- To display specific data from document (person with age 27)

  db.nameofCollection.find({age:27})

- To display all data from document whose age are greater than 26

  db.nameofCollection.find({age: {$gt: 26}})

- To update the document in collection

  db.nameofCollection.update(data to update,updated data)

  eg-- db.users.update({name:'sonali'},{$set:{age:30}})

  it will update the age of specific name

  db.users.update({name:'sonali'},{age:30})

  it will update the age but name will be replaced by only age.

- To delete

  db.users.deleteOne({name:’aaaa’}).

  db.users.deleteMany();

- **Printing in JSON format**
- JSON is a format called **JavaScript Object Notation**, and is just a way to store information in an organized, easy-to-read manner.
- db.users.find().forEach(printjson)