Database Design

Database Design is a collection of processes that facilitate the ***designing, development, implementation and maintenance of enterprise data management systems***. Properly designed database are easy to maintain, improves data consistency and are cost effective in terms of disk storage space. The database designer decides how the data elements correlate and what data must be stored.

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

The logical model concentrates on the data requirements and the data to be stored independent of physical considerations. It does not concern itself with how the data will be stored or where it will be stored physically.

The physical data design model involves translating the logical DB design of the database onto physical media using hardware resources and software systems such as database management systems (DBMS).

Why Database Design is Important?

- It helps produce database systems
- That meet the requirements of the users
- Have high performance.

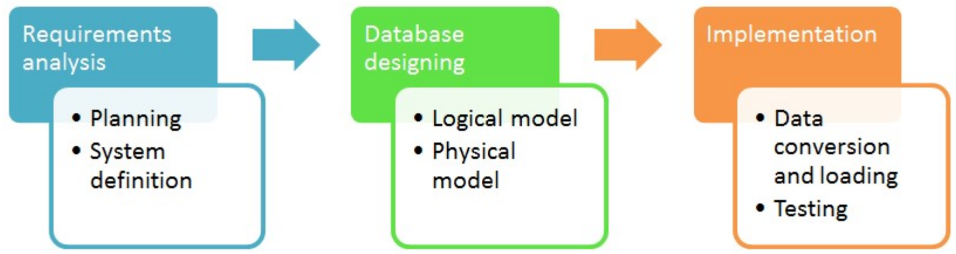
Database design process in DBMS is crucial for high performance database system.

The database development life cycle has a number of stages that are followed when developing database systems.

The steps in the development life cycle do not necessarily have to be followed religiously in a sequential manner.

On small database systems, the process of database design is usually very simple and does not involve a lot of steps.

In order to fully appreciate the above diagram, let's look at the individual components listed in each step for overview of design process in DBMS.



Requirement's analysis

Planning – These stages of database design concepts are concerned with planning of entire Database Development Life Cycle. It takes into consideration the Information Systems strategy of the organization.

System definition – This stage defines the scope and boundaries of the proposed database system.

Database designing

Logical model – This stage is concerned with developing a database model based on requirements. The entire design is on paper without any physical implementations or specific DBMS considerations.

Physical model – This stage implements the logical model of the database taking into account the DBMS and physical implementation factors.

Implementation

Data conversion and loading – this stage of relational databases design is concerned with importing and converting data from the old system into the new database.

Testing – this stage is concerned with the identification of errors in the newly implemented system. It checks the database against requirement specifications.

View of data in DBMS narrate how the data is visualized at each level of data abstraction? Data abstraction allow developers to keep complex data structures away from the users. The developers achieve this by hiding the complex data structures through levels of abstraction.

Data Abstraction

Data abstraction is **hiding the complex data structure** in order to simplify the user's interface of the system. It is done because many of the users interacting with the database system are not that much computer trained to understand the complex data structures of the database system.

To achieve data abstraction, we will discuss a Three-Schema architecture which abstracts the database at three levels discussed below:

Three-Schema Architecture:

The ANSI-SPARC Architecture ***(American National Standards Institute, Standards Planning And Requirements Committee)***, is an abstract design standard for a database management system (DBMS), first proposed in 1975. The main objective of this architecture is to have an effective separation between the user interface and the physical database. So, the user never has to be concerned regarding the internal storage of the database and it has a simplified interaction with the database system.

The three-schema architecture defines the view of data at three levels:

- Physical level (internal level)
- Logical level (conceptual level)
- View level (external level)

1. Physical Level/ Internal Level

The physical or the internal level schema describes how the data is stored in the hardware. It also describes how the data can be accessed. The physical level shows the data abstraction at the lowest level and it has complex data structures. Only the database administrator operates at this level.

2. Logical Level/ Conceptual Level

It is a level above the physical level. Here, the data is stored in the form of the entity set, entities, their data types, the relationship among the entity sets, user operations performed to retrieve or modify the data and certain constraints on the data. Well adding constraints to the view of data adds the security. As users are restricted to access some particular parts of the database.

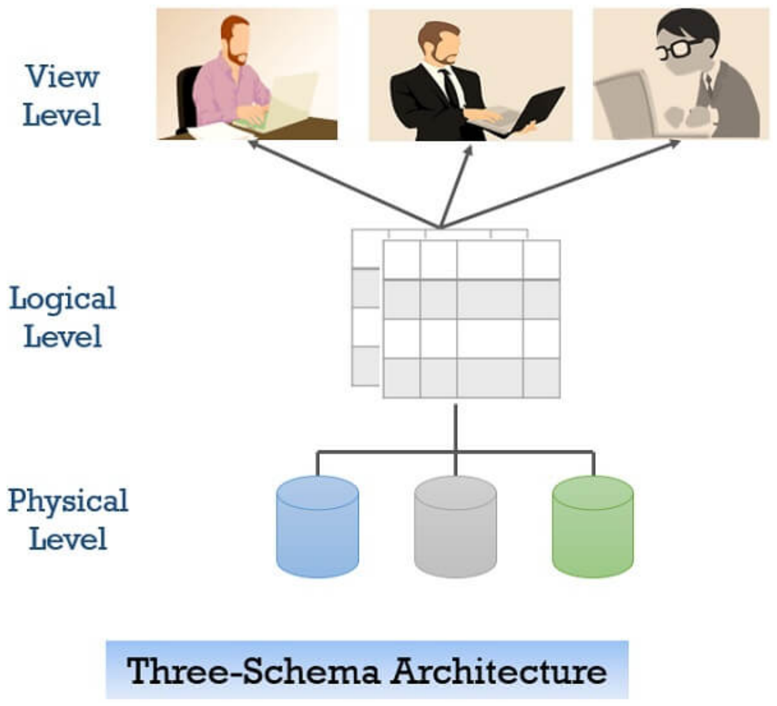
It is the developer and database administrator who operates at the logical or the conceptual level.

3. View Level/ User level/ External level

It is the highest level of data abstraction and exhibits only a part of the whole database. It exhibits the data in which the user is interested. The view level can describe many views of the same data. Here, the user retrieves the information using different application from the database.

The figure below describes the three-schema architecture of the database:

View of data three-schema architecture



In the figure above you can clearly distinguish between the three levels of abstraction. To understand it more clearly let us take an example:

We have to create a database of a college. Now, what entity sets would be involved? Student, Lecturer, Department, Course and so on…

- Now, the entity sets Student, Lecturer, Department, Course will be stored in the storage as the consecutive blocks of the memory location. This is the physical or internal level and is hidden from the programmers but the database administrator is it aware of it. It describes **how** data is actually stored in database. You can get the complex data structure details at this level. It deals with complex low level data structures, file structures and access methods in detail. It also deals with Data Compression and Encryption techniques, if used.
- At the logical level, the programmers define the entity sets and **relationship** among these entity sets using a programming language like SQL. So, the programmers work at the logical level and even the database administrator also operates at this level. It describes **what** data is stored in database.
- At the view level, the users have the set of applications which they use to retrieve the data they are interested in.

Structure of Database Management System

We saw how we can connect to the database. But how is the database laid to process all user requests? Since it is responsible to store huge amounts of data and is capable of handling multiple requests from users simultaneously, it should be arranged properly. One can imagine a database as a brain! How is the structure of the brain? Bit sophisticated and each part of the brain is responsible for some specific tasks. Similarly, Database is also designed.

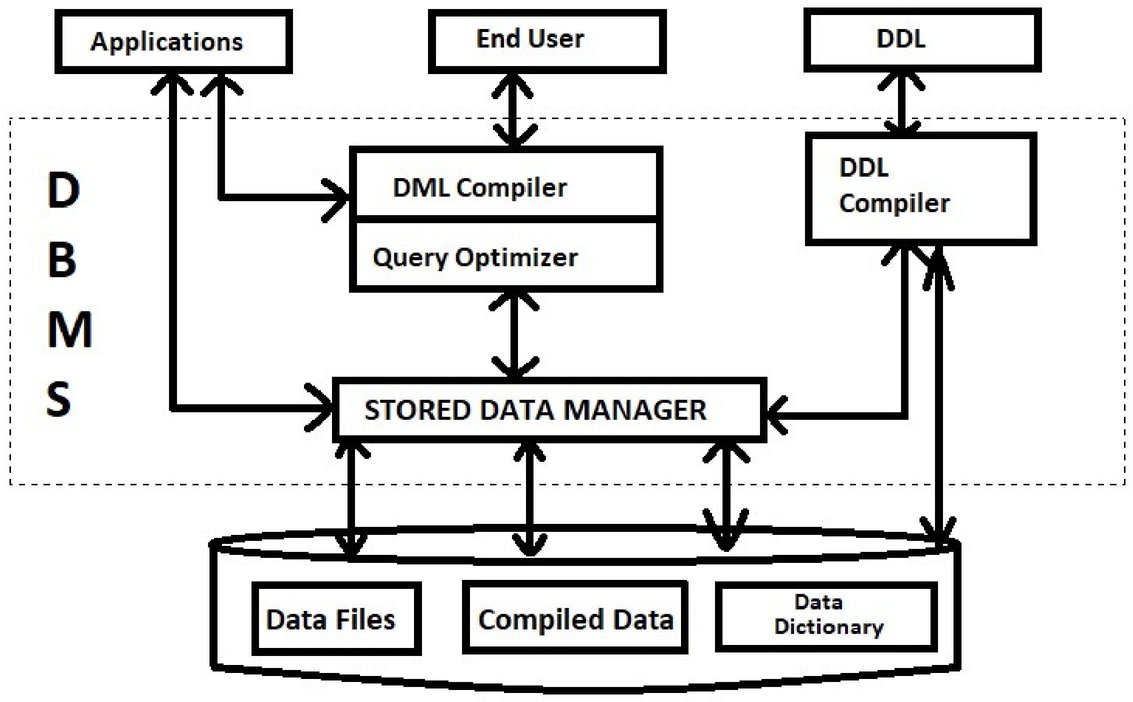


Fig: Structure of DBMS

Applications: – It can be considered as a user-friendly web page where the user enters the requests. Here he simply enters the details that he needs and presses buttons to get the data.

End User: – They are the real users of the database. They can be developers, designers, administrators, or the actual users of the database.

DDL: – Data Definition Language (DDL) is a query fired to create database, schema, tables, mappings, etc. in the database. These are the commands used to create objects like tables, indexes in the database for the first time. In other words, they create, alter, rename, etc. to the structure of the database.

DDL Compiler: – This part of the database is responsible for processing the DDL commands. That means this compiler actually breaks down the command into machine-understandable codes. It is also responsible for storing the metadata information like table name, space used by it, number of columns in it, mapping information, etc.

DML Compiler: – When the user inserts, deletes, updates or retrieves the record from the database, he will be sending requests which he understands by pressing some buttons. But for the database to work/understand the request, it should be broken down to object code. This is done by this compiler. One can imagine this as when a person is asked some question, how this is broken down into waves to reach the brain!

Query Optimizer: – Query optimization is *the overall process of choosing the most efficient means of executing a SQL statement*. When a user fires some requests, he is least bothered how it will be fired on the database. He is not all aware of the database or its way of performance. But whatever be the request, it should be efficient enough to fetch, insert, update, or delete the data from the database. The query optimizer decides the best way to execute the user request which is received from the DML compiler. It is similar to selecting the best nerve to carry the waves to the brain!

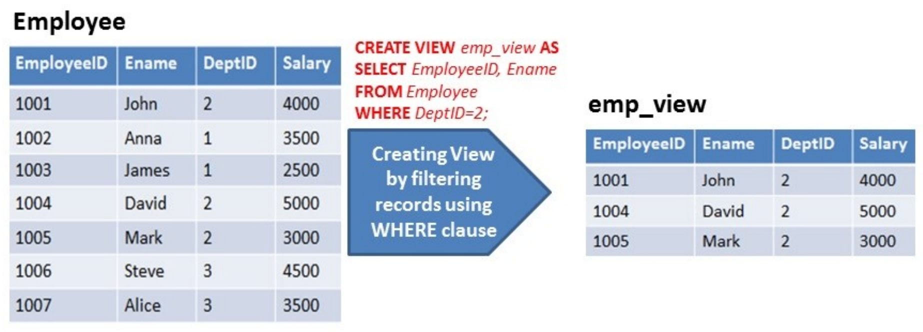
Stored Data Manager: – This is also known as Database Control System. It is one of the main central systems of the database. It is responsible for various tasks

- It converts the requests received from query optimizer to machine-understandable form. It makes actual requests inside the database. It is like fetching the exact part of the brain to answer.
- It helps to maintain consistency and integrity by applying the constraints. That means it does not allow inserting/updating / deleting any data if it has child entry. Similarly, it does not allow entering any duplicate value into database tables.
- It controls concurrent access. If there are multiple users accessing the database at the same time, it makes sure, all of them see correct data. It guarantees that there is no data loss or data mismatch happens between the transactions of multiple users.
- It helps to back up the database and recovers data whenever required. Since it is a huge database and when there is any unexpected exploit of the transaction, and reverting the changes is not easy. It maintains the backup of all data so that it can be recovered.

Data Files: – It has the real data stored in it. It can be stored as magnetic tapes, magnetic disks, or optical disks.

Compiled DML: – Some of the processed DML statements (insert, update, delete) are stored in it so that if there are similar requests, it will be re-used.

Data Dictionary: – It contains all the information about the database. As the name suggests, it is the dictionary of all the data items. It contains a description of all the tables, view, materialized views, constraints, indexes, etc.



Employee

EmployeeID	Ename	DeptID	Salary
1001	John	2	4000
1002	Anna	1	3500
1003	James	1	2500
1004	David	2	5000
1005	Mark	2	3000
1006	Steve	3	4500
1007	Alice	3	3500

```
CREATE VIEW emp_view AS
SELECT EmployeeID, Ename
FROM Employee
WHERE DeptID=2;
```

emp_view

EmployeeID	Ename	DeptID	Salary
1001	John	2	4000
1004	David	2	5000
1005	Mark	2	3000

/IEWS)

Note: Views are the virtual table for various complex data stored in the database. They do not cover any memory space to store the data.

Materialized View are the precomputed views that periodically cache the results of a query for increased performance and efficiency for bigdata Queries.

Constraints is the rule for data that are stored in DB and can be specified when the table is created with the CREATE TABLE statement, or after the table is created with the ALTER TABLE statement. Eg: Not Null, Default, Primary Key, unique, etc.

Data Independence

Data independence defines the extent to which the data schema can be changed at one level without modifying the data schema at the next level. Data independence can be classified as shown below:

Logical Data Independence:

Logical data independence describes the degree up to which the logical or conceptual schema can be changed without modifying the external schema. Now, a question arises what is the need to change the data schema at a logical or conceptual level?

Well, the changes to data schema at the logical level are made either to enlarge or reduce the database by adding or deleting more entities, entity sets, or changing the constraints on data.

Physical Data Independence:

Physical data independence defines the extent up to which the data schema can be changed at the physical or internal level without modifying the data schema at logical and view level.

Well, the physical schema is changed if we add additional storage to the system or we reorganize some files to enhance the retrieval speed of the records.

DDL:

Data definition or Data description language (DDL) is a syntax for creating and modifying database objects such as tables, indices, and users. DDL statements are similar to a computer programming language for defining data structures, especially database schemas. Common examples of DDL statements include CREATE, ALTER, and DROP.

Create Example:

```
CREATE TABLE table_name(
   column1 datatype,
   column2 datatype,
   column3 datatype,
   .....
   columnN datatype,
   PRIMARY KEY( one or more columns )
);

    CREATE TABLE EMPLOYEE(
   ID   INT               NOT NULL,
   NAME VARCHAR (20)      NOT NULL,
   AGE  INT               NOT NULL,
   ADDRESS  CHAR (25) ,
   SALARY   DECIMAL (10, 2),  //decimal (p,s, where p is presicion and s is scale)
   PRIMARY KEY (ID)
);
```

ALTER Example:

The ALTER TABLE statement is used to add, delete, or modify columns in an existing table. The ALTER TABLE statement is also used to add and drop various constraints on an existing table.

```
ALTER TABLE EMPLOYEE
ADD Email varchar(255);
```

DROP Example:

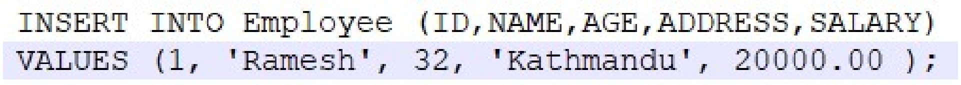
The DROP statement is used to drop/remove an existing table or database.

```
DROP TABLE EMPLOYEE;
```

DML:

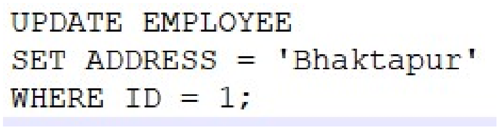
The SQL commands that deals with the manipulation of data present in the database belong to DML or Data Manipulation Language and this includes most of the SQL statements. It is the component of the SQL statement that controls access to data and to the database. Examples are: INSERT, UPDATE, DELETE, etc.

Insert Example:



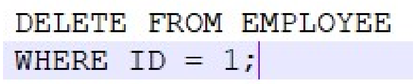
```
INSERT INTO Employee (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (1, 'Ramesh', 32, 'Kathmandu', 20000.00 );
```

Update Example:



```
UPDATE EMPLOYEE
SET ADDRESS = 'Bhaktapur'
WHERE ID = 1;
```

Delete Example:



```
DELETE FROM EMPLOYEE
WHERE ID = 1;
```

QBE (Query by Example):

In SQL we will get an error if the query is not correct but in the case of QBE if the query is wrong either we get a wrong answer or the query will not be going to execute but we will never get any error.

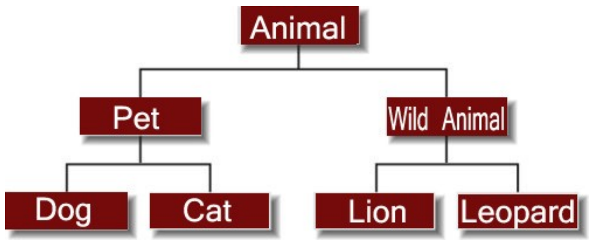
In QBE we don't write complete queries like SQL or other database languages it comes with some blank so we need to just fill that blank and we will get our required result.

Database Model:

Database Model provides us an idea that how the final system will look like after its complete implementation. It defines the data elements and the relationships between the data elements. Data Models are used to show how data is stored, connected, accessed and updated in the **Database management system**.

1. Hierarchical Model
2. NetworkModel
3. Entity-Relationship Model
4. Relational Model
5. Object-Oriented DataModel

1. Hierarchical Model: Hierarchical Model was the first DBMS model. This model organizes the data in the hierarchical tree structure. The hierarchy starts from the root which has root data and then it expands in the form of a tree adding child node to the parent node. This model easily represents some of the real-world relationships like food recipes, sitemap of a websiteetc.


Animal
Pet
Wild Animal
Dog
Cat
Lion
Leopard

Advantages of Hierarchical Model

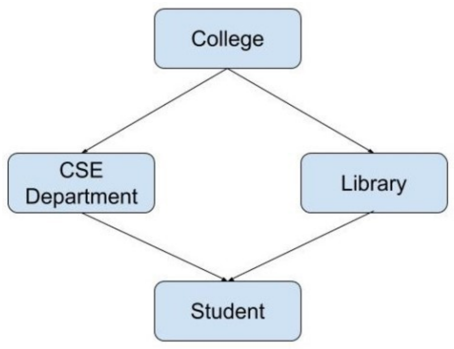
- It is very simple and fast to traverse through a tree-likestructure.
- Any change in the parent node is automatically reflected in the child node so, the integrity of data ismaintained.
- Database Security is easily provided by DBMS in this model.
- Its effective for 1:1 or 1:Mrelationships.

Disadvantages of Hierarchical Model

- Complex relationships are notsupported.
- As it does not support more than one parent of the child node so if we have some complex relationship where a child node needs to have two parent nodes then that can't be represented using thismodel.
- If a parent node is deleted then the child node is automaticallydeleted.

2. NetworkModel

This model is an extension of the hierarchical model. it allows many-to- many relationships to be managed in a tree-like structure that allows multiple parents.



Network Model

Advantages of Network Model

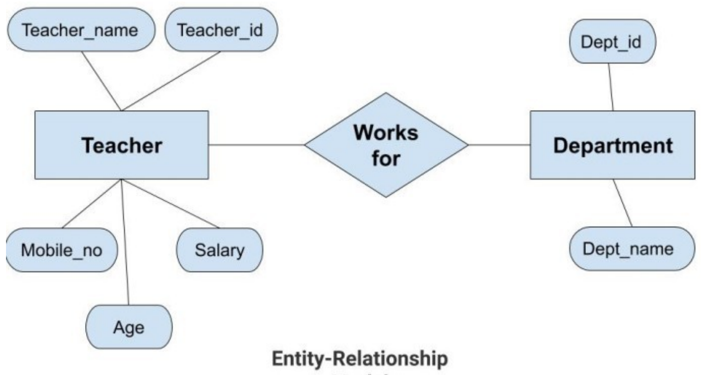
- The data can be accessed faster as compared to the hierarchical model. This is because the data is more related in the network model and there can be more than one path to reach a particular node. So the data can be accessed in many ways.
- As there is a parent-child relationship so data integrity is present. Anychange in parent record is reflected in the childrecord.

Disadvantages of Network Model

- As more and more relationships need to be handled the system might get complex. So, a user must be having detailed knowledge of the model to work with themodel.
- Any change like updation, deletion, insertion is verycomplex.

3. Entity-RelationshipModel

Entity-Relationship Model or simply ER Model is a high-level data model diagram. In this model, we represent the real-world problem in the pictorial form to make it easy for the stakeholders to understand. It is also very easy for the developers to understand the system by just looking at the ER diagram.



Entity-Relationship Model

Advantages of ER Model

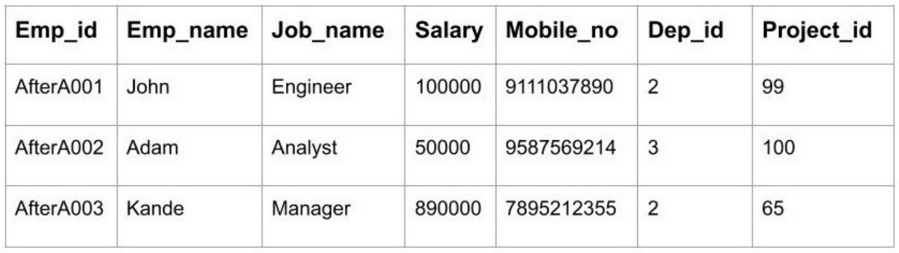
- ***Simple:*** Conceptually ER Model is very easy to build. If we know therelationship between the attributes and the entities, we can easily build the ER Diagram for themodel.
- ***Effective Communication Tool***: This model is used widely by the database designers for communicating theirideas.
- ***Easy 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 modeletc.

Disadvantages of ER Model

- Some information might be lost or hidden in the ER model. As it is ahigh-level view so there are chances that some details of information might behidden.

4. RelationalModel

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. So, the tables are also called *relations* in the relational model.



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

Advantages of Relational Model

- ***Simple:*** This model is simpler as compared to the network andhierarchical model.
- ***Scalable:*** This model can be easily scaled as we can add as manyrows and columns we want.
- ***Tabular View:*** The Tabular View help the data to be managed properly within rows and column and provides **simplicity** in theview.

Disadvantages of Relational Model

- ***Hardware Overheads:*** For hiding the complexities and making things easier for the user this model requires more powerful hardware computers and data storagedevices.
- ***Bad Design:*** As the relational model is very easy to design and use. So the users don't need to know how the data is stored in order to access it. This ease of design can lead to the development of a poor database which would slow down if the databasegrows.

5. Object Oriented DataModel

Object oriented data model is based upon real world situations. These situations are represented as objects, with different attributes. All these objects have multiple relationships between them.

Elements of Object-oriented data model

- **Objects**

The real-world entities and situations are represented as objects in the Object-oriented database model.

- **Attributes andMethod**

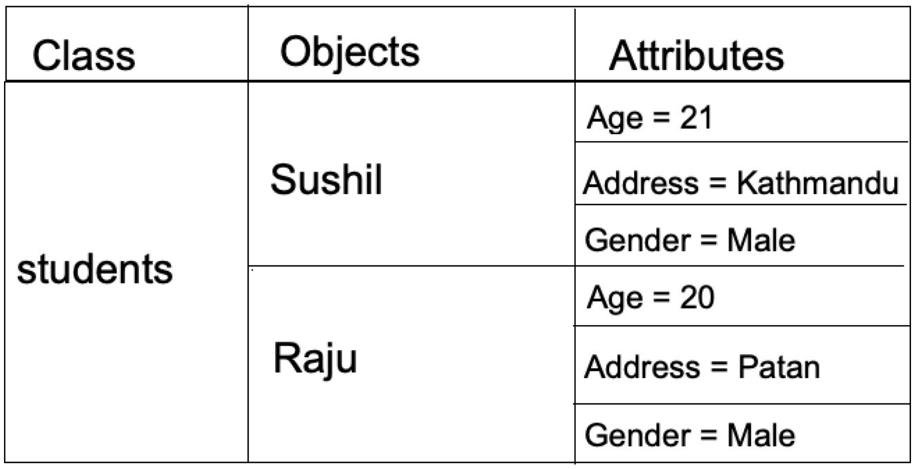
Every object has certain characteristics. These are represented usingAttributes. The behavior of the objects is represented usingMethods.

- **Class**

Similar attributes and methods are grouped together using a class. An object can be called as an instance of the class.

- **Relationship**

Relationship is based on two or more entities.



Class	Objects	Attributes
students	Sushil	Age = 21
		Address = Kathmandu
		Gender = Male
	Raju	Age = 20
		Address = Patan
		Gender = Male

Advantages of Object-Oriented Data Model

- Easy to save and retrieve data quickly.
- Logical content view isavailable.
- Supports OOPs language property like class, encapsulation, abstraction, etc.

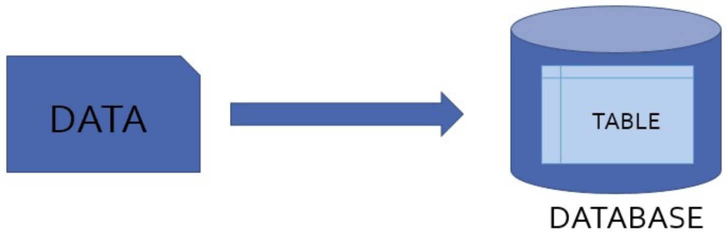
Disadvantages of Object-Oriented Data Model

- Not as widely adopted as relationaldatabases.
- High complexity causes performanceissues.
- Need more memory to representobjects.

ER Model to RDBMS:

What is relational model:

Relational Model represents how data is stored in database in the form of table.



1.Entity Set:

Consider we have entity STUDENT in ER diagram with attributes Roll Number, Student Name and Class. To convert this entity set into relational schema

1. Entity is mapped as relation in Relational schema

2. Attributes of Entity set are mapped as attributes for that Relation.
3. Key attribute of Entity becomes Primary key for that Relation.

2.Entity set with multi valued attribute:

Consider we have entity set Employee with attributes Employee ID, Name and Contact number.Here contact number is multivalued attribute as it has multiple values. as an employee can have more than one contact number for that we have to repeat all attributes for every new contact number. This will lead to data redundancy in table.

Hence to convert entity with multivalued attribute into relational schema, separate relation is created for multivalued attribute in which:

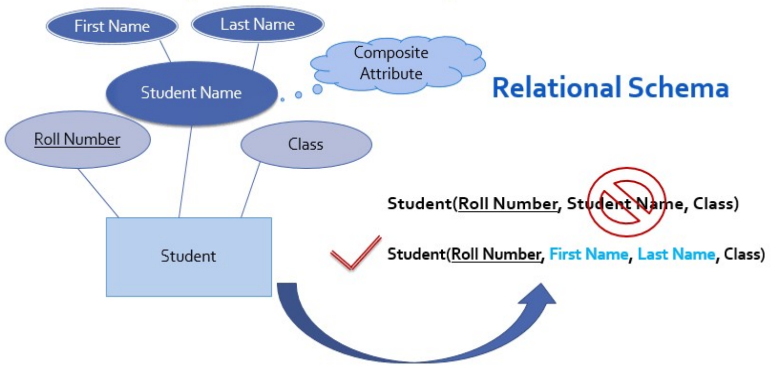
1. Key attribute and multivalued attribute of entity set becomes primary key of relation.
2. Separate relation employee is created with remaining attributes.

Due to this instead of repeating all attributes of entity now only one attribute is need to repeat.

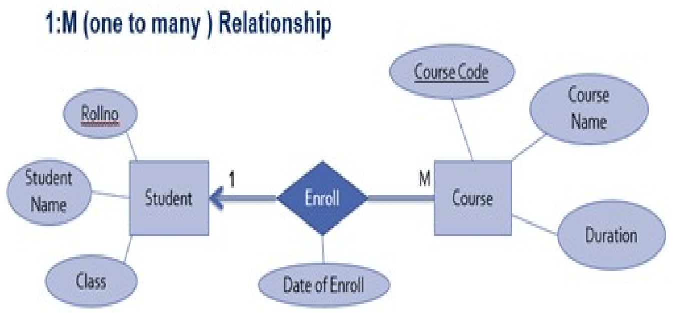
3. Entity set with Composite attribute:

Consider entity set student with attributes Roll Number, Student Name and Class. Here student name is composite attribute as it has further divided into First name, last name.

In this case to convert entity into relational schema, composite attribute student name should not be included in relation but all parts of composite attribute are mapped as simple attributes for relation


First Name
Last Name
Composite Attribute
Student Name
Relational Schema
Roll Number
Class
Student
Student(Roll Number, Student Name, Class)
Student(Roll Number, First Name, Last Name, Class)

4. 1:M (one to many) Relationship: Consider 1:M relationship set enrolled exist between entity sets student and course as follow,

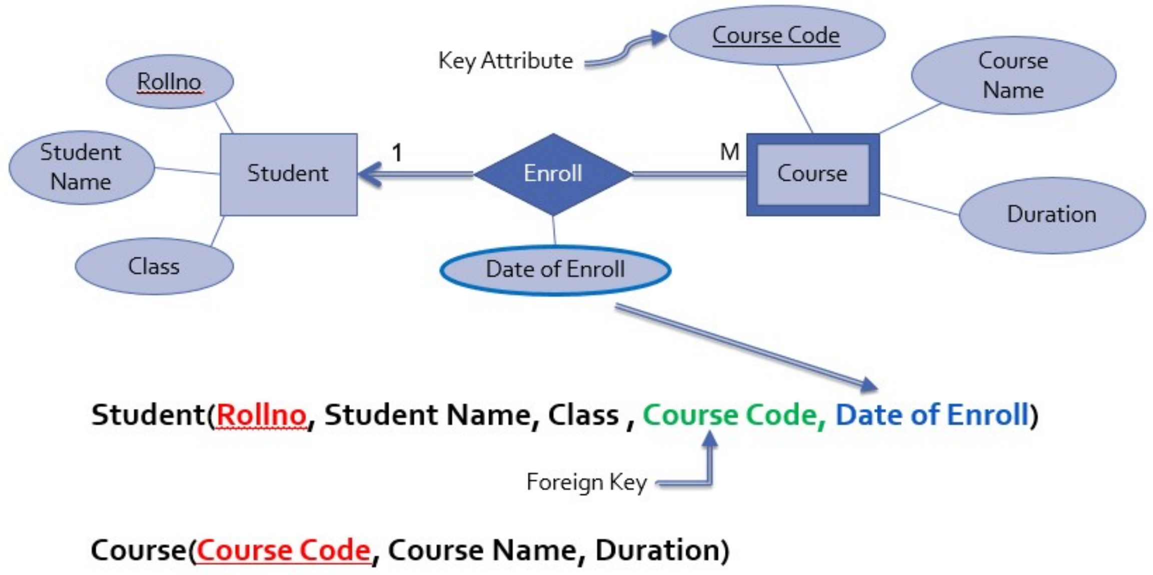


Attributes of entity set student are Roll no which is primary key, student name and class Attributes of entity set course are Course code which is primary key, Course name and duration And date of enroll is attribute of relationship set enroll.

Here Enroll is 1:M relationship exist between entity set student and course which means that *one student can enroll in multiple courses.* In this case to convert this relationship into relational schema,

1.Separate relation is created for all participating entity sets (student and course)
2. Key attribute of Many's side entity set (course) is mapped as foreign key in one's side relation (Student)

3.All attributes of relationship set are mapped as attributes for relation of one’s side entity set (student)

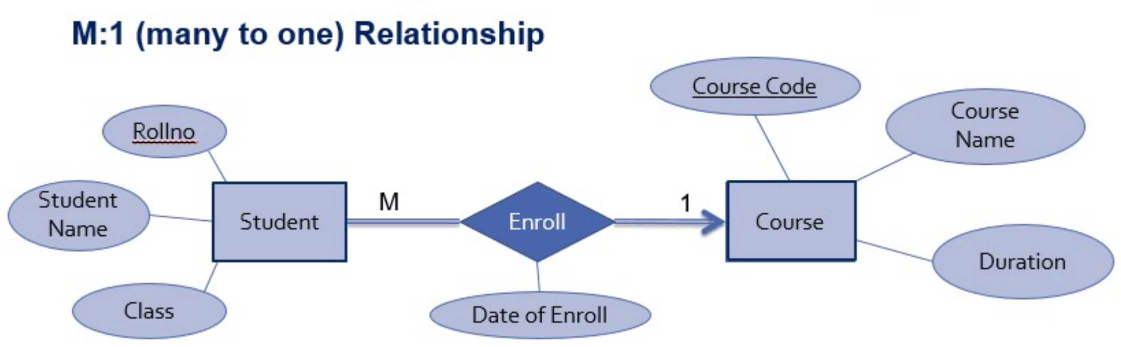


Student(Rollno, Student Name, Class , Course Code, Date of Enroll)

Course(Course Code, Course Name, Duration)

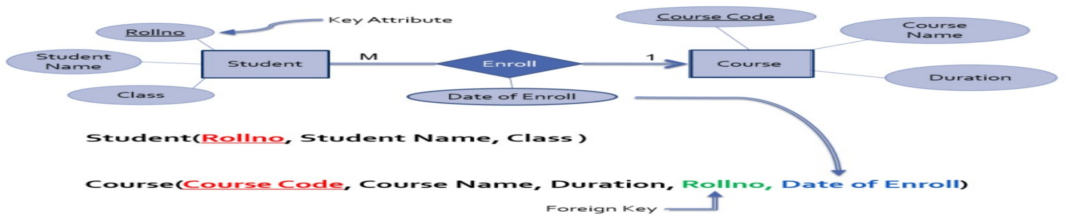
5. M:1 (many to one) Relationship:

Consider same relationship set enroll exist between entity sets student and course but here student is many side entity set while course is one side entity set. Which means many student can enroll in one course.



To convert this relationship set into relational schema:

1. Separate relation is created for all participating entity sets.
2. Key attribute of Many’s side entity set student is mapped as foreign key in one’s side relation
3. All attributes of relationship set are mapped as attributes for one’s side relation course.

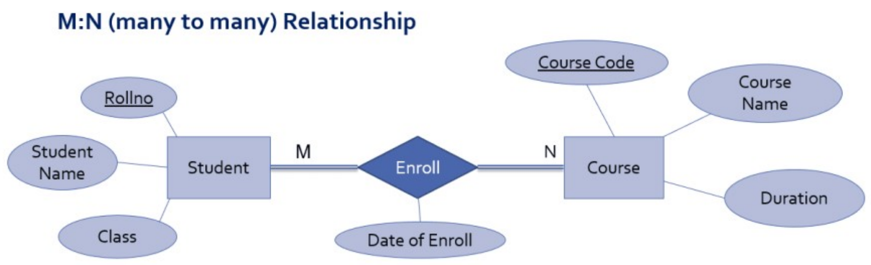


Student(Rollno, Student Name, Class)

Course(Course Code, Course Name, Duration, Rollno, Date of Enroll)

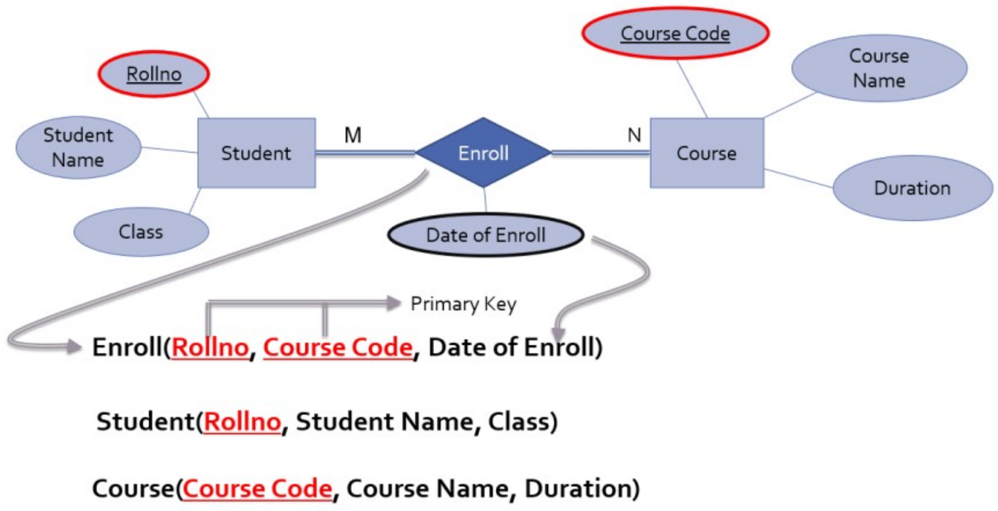
6. M:N (many to many) Relationship:

Consider same relationship set enrolled exist between entity sets student and course,which means multiple students can enroll in multiple courses.



To convert this Relationship set into relational schema,

1. Relationship set is mapped as separate relation
2. Key attributes of participating entity sets are mapped as primary key for that relation
3. Attribute of relationship set becomes simple attributes for that relation
4. And separate relation is created for other participating entities



Enroll(Rollno, Course Code, Date of Enroll)

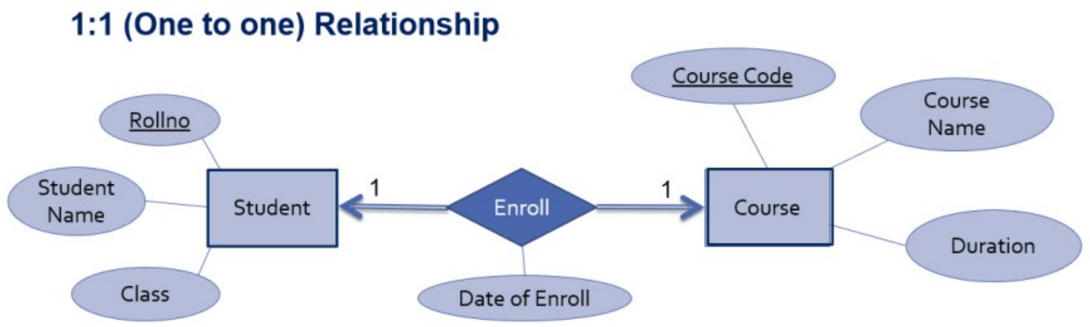
Student(Rollno, Student Name, Class)

Course(Course Code, Course Name, Duration)

5.

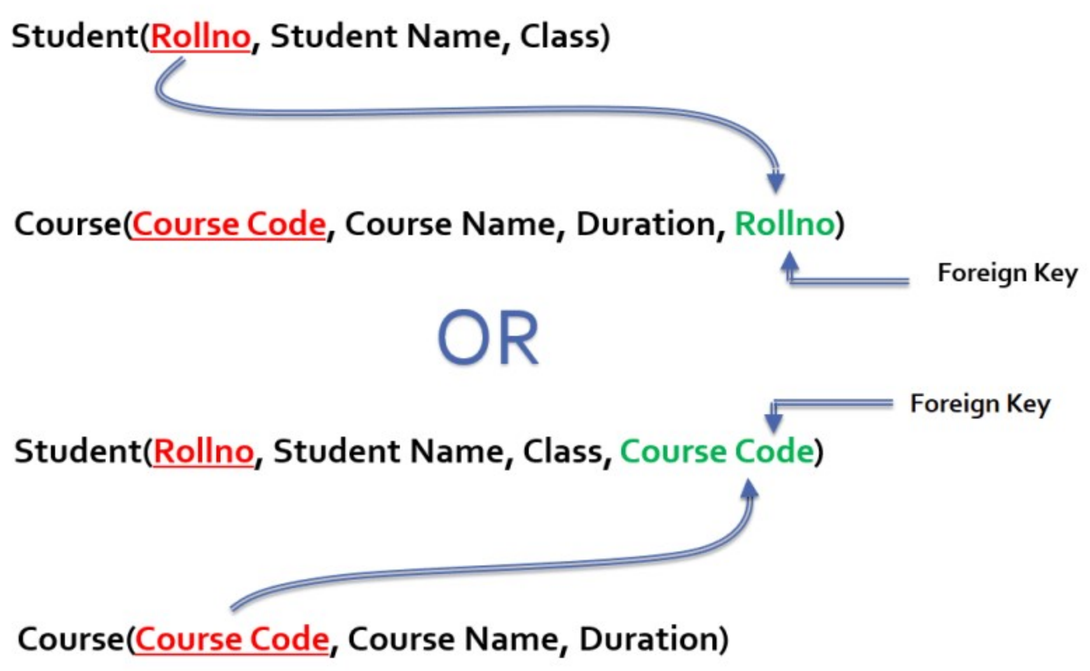
7. 1:1 (one to one) Relationship:

Consider same relationship set enroll exist between entity sets student and course, which means one student can enroll in only one courses



To convert this Relationship set into relational schema,

1. Separate relation is created for all participating entity sets.
2. Primary Key of Relation Student can be act as foreign key for relation Course OR Primary Key of Relation Course act as foreign key for relation Student.


Student(Rollno, Student Name, Class)
Course(Course Code, Course Name, Duration, Rollno)
Foreign Key
OR
Foreign Key
Student(Rollno, Student Name, Class, Course Code)
Course(Course Code, Course Name, Duration)

Aggregation:

Database management systems (DBMS) have replaced the traditional filing system by providing an easy, secure, efficient, and reliable way of storing, retrieving, accessing, and sharing data within databases.

DBMS is advantageous over the file system because it reduces data redundancy (through database normalization) and enhances data integrity. It also offers flexibility, privacy, and data security.

A DBMS consists of entities whose data can be stored. They can be people, things, objects, or places. Two or more entities are joined through a relationship, that is simply a way of connecting data sets. Some entities in a DBMS may have little value, which makes it difficult to use them for certain operations.

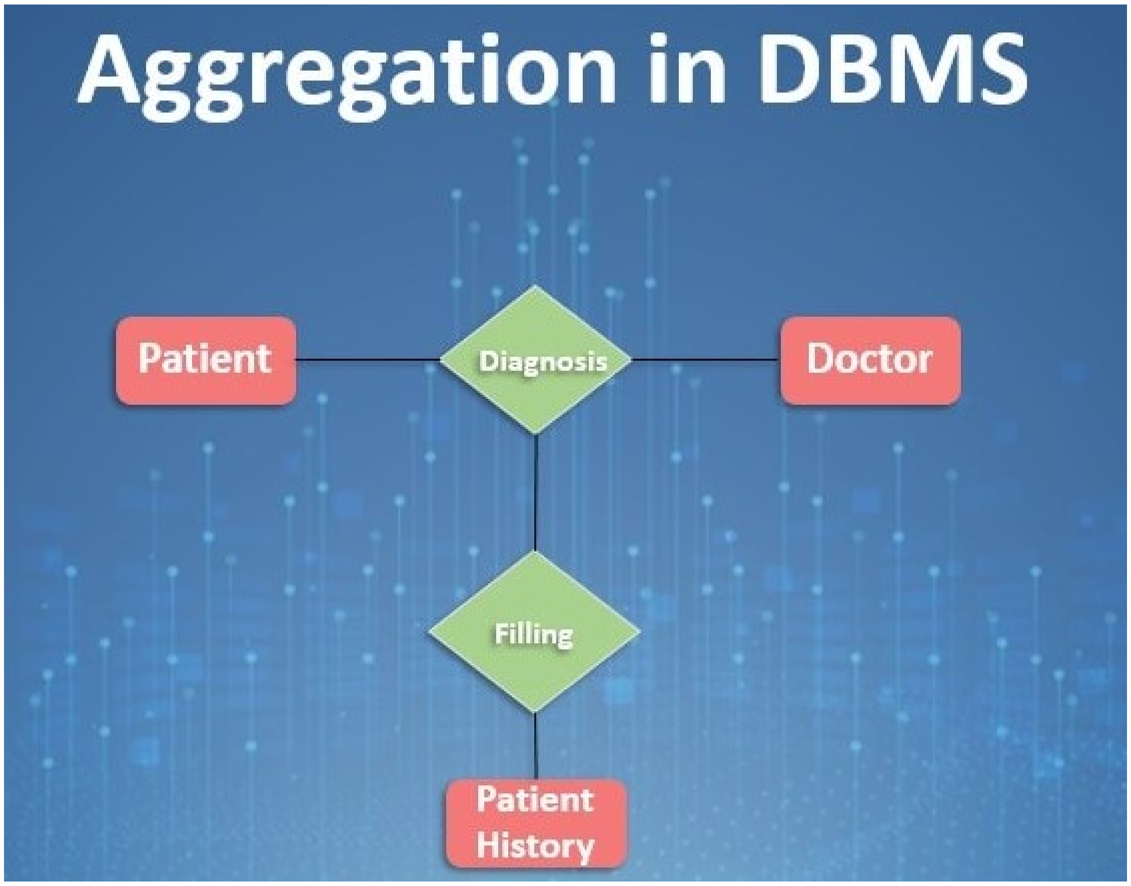
In such situations, we can combine these entities with other entities to form a complex one that makes sense. We can do this operation through a process called aggregation. Aggregation in DBMS links trivial (group and sub-groups) entities through relationships to ensure that the entire system functions well.

Aggregation refers to the process by which entities are combined to form a single meaningful entity. The specific entities are combined because they do not make sense on their own. To establish a single entity, aggregation creates a relationship that combines these entities. The resulting entity makes sense because it enables the system to function well.

When using data in the form of numerical values, the following operations can be used to perform DBMS aggregation:

- Average (AVG): This function provides the mean or average of the data values.
- Sum: This provides a total value after the data values have been added.

- Count: This provides the number of records.
- Maximum (Max): This function provides the maximum value of a given set of data.
- Minimum (Min): This provides the minimum value of a given set of data.
- Standard deviation (std dev): This provides the dispersion or variation of the sets of data. Let's take a simple example of a database of student marks. If the standard deviation is high, it means the average is obtained by lower number of students than usual, and the lowest and highest marks are higher.



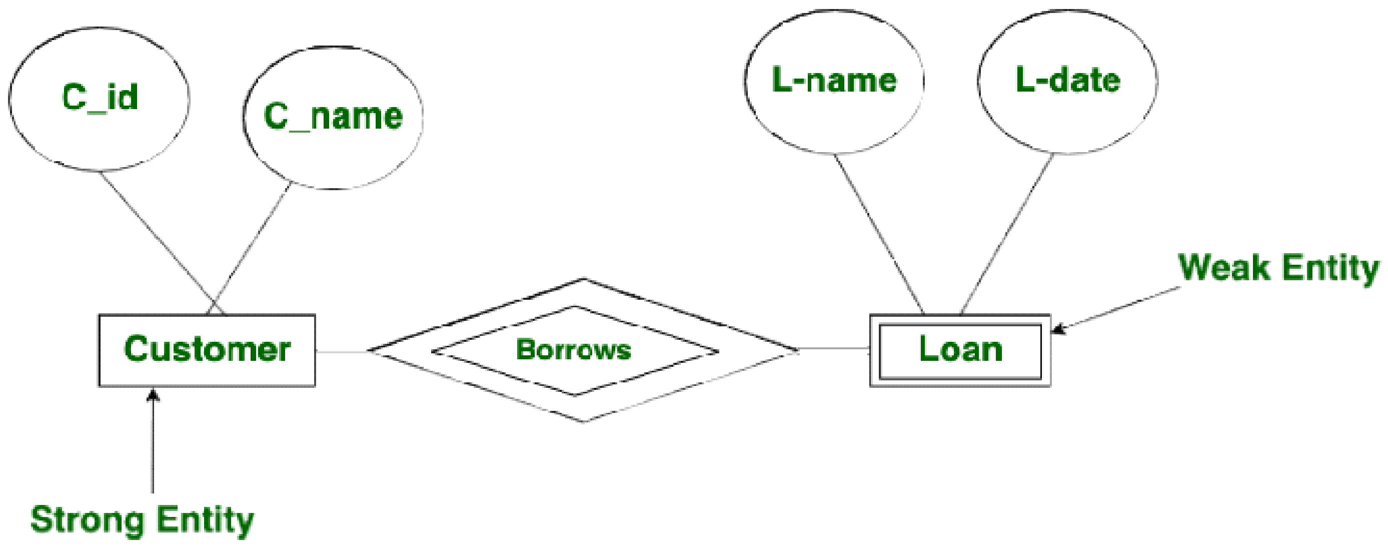
Strong Entity:

A strong entity is not dependent on any other entity in the schema. A strong entity will always have a primary key. Strong entities are represented by a single rectangle. The relationship of two strong entities is represented by a single diamond. Various strong entities, when combined together, create a strong entity set.

Weak Entity:

A weak entity is dependent on a strong entity to ensure its existence. Unlike a strong entity, a weak entity does not have any primary key. It instead has a partial discriminator key. A weak entity is represented by a double rectangle. It doesn't have sufficient attributes for its identification.

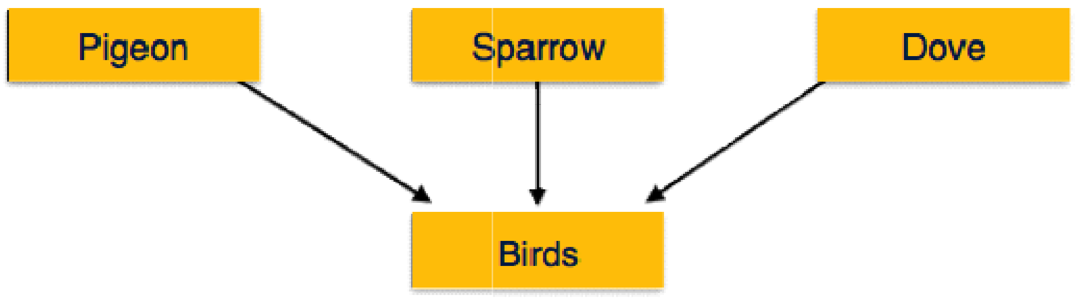
The relation between one strong and one weak entity is represented by a double diamond. This relationship is also known as identifying relationship.



Generalization and Specialization

The ER Model has the power of expressing database entities in a conceptual hierarchical manner. As the hierarchy goes up, it generalizes the view of entities, and as we go deep in the hierarchy, it gives us the detail of every entity included.

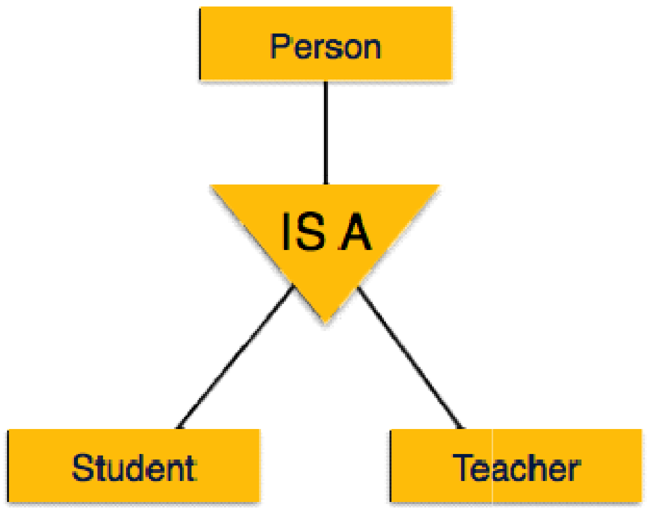
Going up in this structure is called **generalization**, where entities are clubbed together to represent a more generalized view. For example, a particular student named Kabindra can be generalized along with all the students. The entity shall be a student, and further, the student is a person. The reverse is called specialization where a person is a student, and that student is Kabindra.



As mentioned above, the process of generalizing entities, where the generalized entities contain the properties of all the generalized entities, is called generalization. In generalization, a number of entities are brought together into one generalized entity based on their similar characteristics. For example, pigeon, house sparrow, crow and dove can all be generalized as Birds.

Specialization

Specialization is the opposite of generalization. In specialization, a group of entities is divided into sub-groups based on their characteristics. Take a group 'Person' for example. A person has name, date of birth, gender, etc. These properties are common in all persons, human beings. But in a company, persons can be identified as employee, employer, customer, or vendor, based on what role they play in the company.



Similarly, in a school database, persons can be specialized as teacher, student, or a staff, based on what role they play in school as entities.