Unit 1

Introduction

Introduction to Database Management Systems

Data-It can be anything in the form of numbers, letters etc. To manage this data different application programs exist.

e.g. In order to write a letter we take the help of the application such as MS-Word similarly
for managing data of an enterprise we have to make use of Database Management System(DBMS)i.e. with DBMS software its possible

Before DBMS the data can be managed with File Management System (FMS)

In FMS permanent records are stored in various files and different application programs are written to extract records from and to add records to the appropriate files.

Introduction to Database Management Systems

A **DBMS** consists of a collection of interrelated data and a set of programs to access those data.

The **collection** of data usually referred to as a database which contains information about one particular enterprise.

The primary goal of **DBMS** is to provide an environment that is both convenient and efficient to use in retrieving and storing database information.

Introduction to Database Management Systems

Definitions: - 1. The scheme of grouping a set of integrated files together is called as **database.**

2. A **Data Base Management System** is a set of prewritten programs that are used to store, update and retrieve a database.

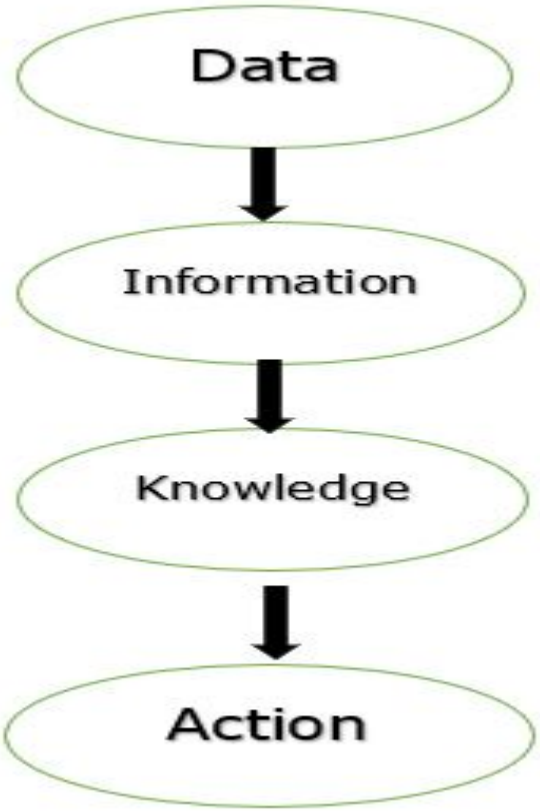
3. A **DBMS** is a collection of interrelated files and a set of programs to allow users to access and modify these files.

Purpose of Data base System

The purpose of DBMS is to transform the following −

- Data into information.
- Information into knowledge.
- Knowledge to the action.

Purpose of Data base System



Purpose of Data base System

In Hospital Management Information System If one application program is written which retrieve records of the patients such as patient_id, patient_nm,address now if a need arises to extract information like type of patient whether IPD or OPD then in that case we need to write a new application program which contains information like patient_id, patient_nm,address,type as well as we need to create a file which has the specified attributes

Purpose of Data base System

Take an application in a saving bank. Saving accounts and customer records are kept in permanent system files.

Application programs are written to manipulate files to perform some tasks like follows −

- Debit or credit an account.
- Add a new account.
- Find an account balance.
- Generate monthly statements.

Development of the system proceeds as new application programs must be written as the need arises, new permanent files are created as required, but over a long period of time files may be in different formats and application programs may be in different languages.

Purpose of Data base System

Before DBMS the data can be managed with File Management System (FMS).

In FMS permanent records are stored in various files and different application programs are written to extract records from and to add records to the appropriate files.

Purpose of Data base System

- This FMS suffers from major disadvantages and hence to overcome these disadvantages DBMS comes into existence

1.1 Need for DBMS

Different application programs are created in order to access data from files.

Example

In Hospital Management Information System If one application program is written which retrieve records of the patients such as patient_id, patient_nm,address now if a need arises to extract information like type of patient whether IPD or OPD then in that case we need to write a new application program which contains information like patient_id, patient_nm,address,type as well as we need to create a file which has the specified attributes.

This FMS suffers from major disadvantages and hence to overcome these disadvantages DBMS comes into existence

Disadvantages of FMS (Advantages of DBMS)

1. Data Redundancy and Inconsistency

Application programs and files are created by different programmers in different languages with different formats leads to these 2 problems Data duplication or data redundancy means multiple copies of the same data items. Data Inconsistency means different values for same data item

e.g. In Hospital Management Information System patient_id, patient_nm,address can gets replicated in different files. This will give rise to data duplication. Changed type of patient can be reflected in one file but may not be reflected in other file this will give rise to inconsistency

2. Difficulty in Accessing data

Conventional FMS does not allow needed data to be retrieved in a convenient and efficient manner. More responsive data retrieval systems must be developed for general use.

e.g. If a person ask for retrieval of patients past history and if this field is not present in the file then system programmer has to write a new application program which contains patient name and his/her history and then has to check manually for a patients history.

Disadvantages of FMS (Advantages of DBMS)

3. Data Isolation

Because data are scattered in various files and files may be in different formats it's difficult to write new application programs to retrieve appropriate data

4. Integrity Problems

Data values stored in database must satisfy certain types of consistency constraints

Example-Banks put certain kind of restrictions regarding balance i.e. balance of a customer should not fall down below 500 etc. Hence programmer has to write certain code to enforce such constraints

Disadvantages of FMS (Advantages of DBMS)

5. Atomicity Problems

The transaction done in the database should be atomic. It means it should get completed or it should remain incomplete.

Example Suppose customer A wants to transfer 500 Rs. amount to B with their current balance as 1500 and 1000 but before depositing this amount to B if a failure occurs then database will remain in an inconsistent state with balance of A as 1000 but balance of B as 1000 instead of correct balance as 1000 and 1500 respectively. So for maintaining consistency it is necessary that this transaction should takes place or remain incomplete with previous values.

6. Concurrent Access Anomalies

Many systems allow multiple users to update data simultaneously. Hence interaction of concurrent updates may result in inconsistent data.

Disadvantages of FMS (Advantages of DBMS)

5. Atomicity Problems

The transaction done in the database should be atomic. It means it should get completed or it should remain incomplete.

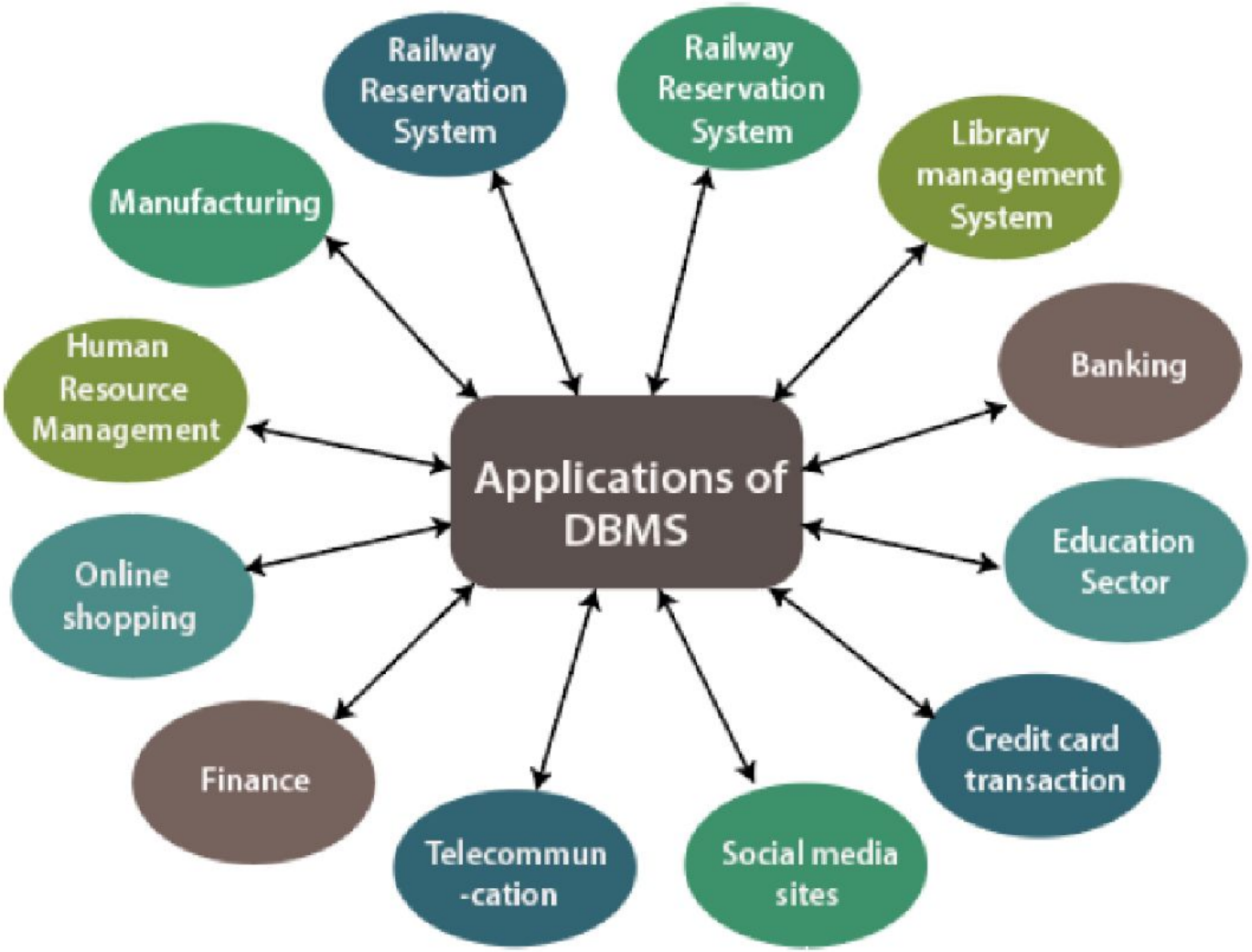
Example Suppose customer A wants to transfer 500 Rs. amount to B with their current balance as 1500 and 1000 but before depositing this amount to B if a failure occurs then database will remain in an inconsistent state with balance of A as 1000 but balance of B as 1000 instead of correct balance as 1000 and 1500 respectively. So for maintaining consistency it is necessary that this transaction should takes place or remain incomplete with previous values.

6. Concurrent Access Anomalies

Many systems allow multiple users to update data simultaneously. Hence interaction of concurrent updates may result in inconsistent data.

- To overcome these problems development of database management systems was introduced. So, DBMS means it is defined as a software system that allows the user to define, create and maintain the database and provide control access to the data.
- DBMS is a collection of programs used for managing data and simultaneously it supports different types of users to create, manage, retrieve, update and store information.

DBMS Applications



Database- System Applications

- **Railway Reservation System** − It is used to keep record of booking of tickets, departure of the train and the status of arrival and give updates to the passengers with the help of a database.
- **Library Management System** − There will be so many numbers of books in the library and it is very hard to keep a record of all the books in a register or a copy. So, DBMS is necessary to keep track of all the book records, issue dates, name of the books, author and maintain the records.

Database- System Applications

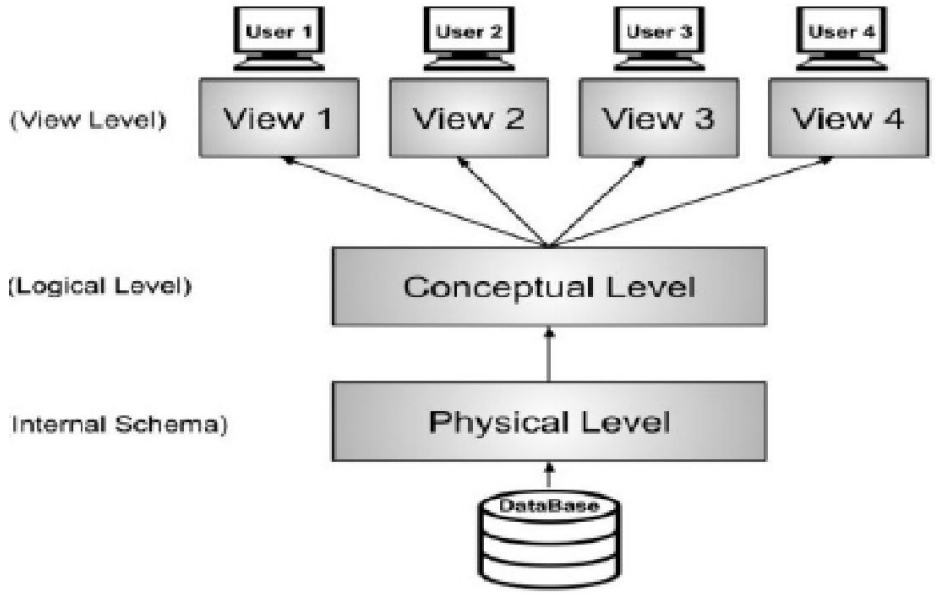
- **Banking** − We are doing a lot of transactions daily without directly going to the banks. The only reason is the usage of databases and it manages all the data of the customers over the database.
- **Educational Institutions** − All the examinations and the data related to the students maintained over the internet with the help of a database management system. It contains registration details of the student, results, grades and courses available. All these works can be done online without visiting an institution.

Database- System Applications

- **Social Media Websites** − By filling the required details we are able to access social media platforms. Many users daily sign up for social websites such as Facebook, Pinterest and Instagram. All the information related to the users are stored and maintained with the help of DBMS.

View of Data

A database system to be usable it must retrieve data efficiently. A database system provides a user an abstract view of the data. The database system users are not computer trained, developers hides the complexity from users through several levels of abstraction to simplify user's interaction with the system.



Levels of Data Abstraction

View of Data

1. **Physical Level**:-It is the lowest level of abstraction describes how the data are actually stored. Provides information regarding the physical organization of data in the database

2. **Logical Level**: - Next higher level of abstraction describes what data are stored in the database and what relationship exists among those data.
Logical level of abstraction is used by DBA (Data Base Administrator) who decides what information is to be kept in atabase.

3. **View Level**: - The highest level of abstraction describes only part of the entire database. Many users of database system will not be concerned with all this information. Instead such users need to access only a part of the database.

Instances and schemas

Definition of Instance:-Database changes over time as information is inserted and deleted. The
collection of information stored in database at a particular moment is called an instance.
Definition of Schema:-The overall design of the database is called the database schema.
Schemas are changed rarely.
Example: - In case of programming language the variable declared of a specific type is called a
schema and the value of that variable at a particular moment is called as Instance1

Data Independence

Definition: - The ability to modify a schema definition in one level without affecting a schema definition in the next higher level is called data independence.

There are 2 levels of data independence

1. **Physical Data Independence**:-It is the ability to modify physical schema without causing application programs to be rewritten. Modification at physical level is occasionally necessary to improve performance.

2. **Logical Data Independence**: - It is the ability to modify logical schema without causing application programs to be rewritten Modification at the logical level are necessary whenever the logical structure of the database is altered.

Logical data independence is difficult to achieve than physical. Because application programs heavily depend on logical structure of data that they access.

Database Languages

Database system provides two different types of languages

1. To specify database schema
2. To express database queries and updates.

1.Data Definition Language(DDL)

statements are used to classify the database structure or schema. It is a type of language that allows the DBA or user to depict and name those entities, attributes, and relationships that are required for the application along with any associated integrity and security constraints. Here are the lists of tasks that come under DDL:

- **CREATE** - used to create objects in the database
- **ALTER** - used to alters the structure of the database
- **DROP** - used to delete objects from the database
- **TRUNCATE** - used to remove all records from a table, including all spaces allocated for the records are removed
- **COMMENT** - used to add comments to the data dictionary
- **RENAME** - used to rename an object

Database Languages

2. Data Manipulation Language(DML)

DML is a language that enables users to access or manipulate data as organized by the appropriate data model. The types of access are retrieval, insertion, deletion and modification.

There are 2 types of DML's

1. **Procedural DML**- Requires a user to specify what data are needed and how to get those data.

Ex-PL/SQL

2. **Declarative DML**- What data are needed without specifying how to get those data. These DML easier to learn and use than procedural DML.

Ex- All SQL queries.

Database Languages

2. Data Manipulation Language(DML)

A language that offers a set of operations to support the fundamental data manipulation operations on the data held in the database. Data Manipulation Language (DML) statements are used to manage data within schema objects. Here are the lists of tasks that come under DML:

SELECT - It retrieves data from a database
INSERT - It inserts data into a table
UPDATE - It updates existing data within a table
DELETE - It deletes all records from a table, the space for the records remain
MERGE - UPSERT operation (insert or update)
CALL - It calls a PL/SQL or Java subprogram
EXPLAIN PLAN - It explains access path to data
LOCK TABLE - It controls concurrency

Database Languages

3. The Data Control Language (DCL)

There is another two forms of database sub-languages. The Data Control Language (DCL) is used to control privilege in Database. To perform any operation in the database, such as for creating tables, sequences or views we need privileges. Privileges are of two types,

System - creating a session, table etc are all types of system privilege.

Object - any command or query to work on tables comes under object privilege. DCL is used to define two commands. These are:

Grant - It gives user access privileges to a database.

Revoke - It takes back permissions from the user.

Database Languages

4. Transaction Control Language(TCL)

Transaction Control statements are used to run the changes made by DML statements. It allows statements to be grouped together into logical transactions.

- **COMMIT** - It saves the work done
- **SAVEPOINT** - It identifies a point in a transaction to which you can later roll back
- **ROLLBACK** - It restores database to original since the last COMMIT
- **SET TRANSACTION** - It changes the transaction options like isolation level and what rollback segment to use

Database Architecture/Database System Structure

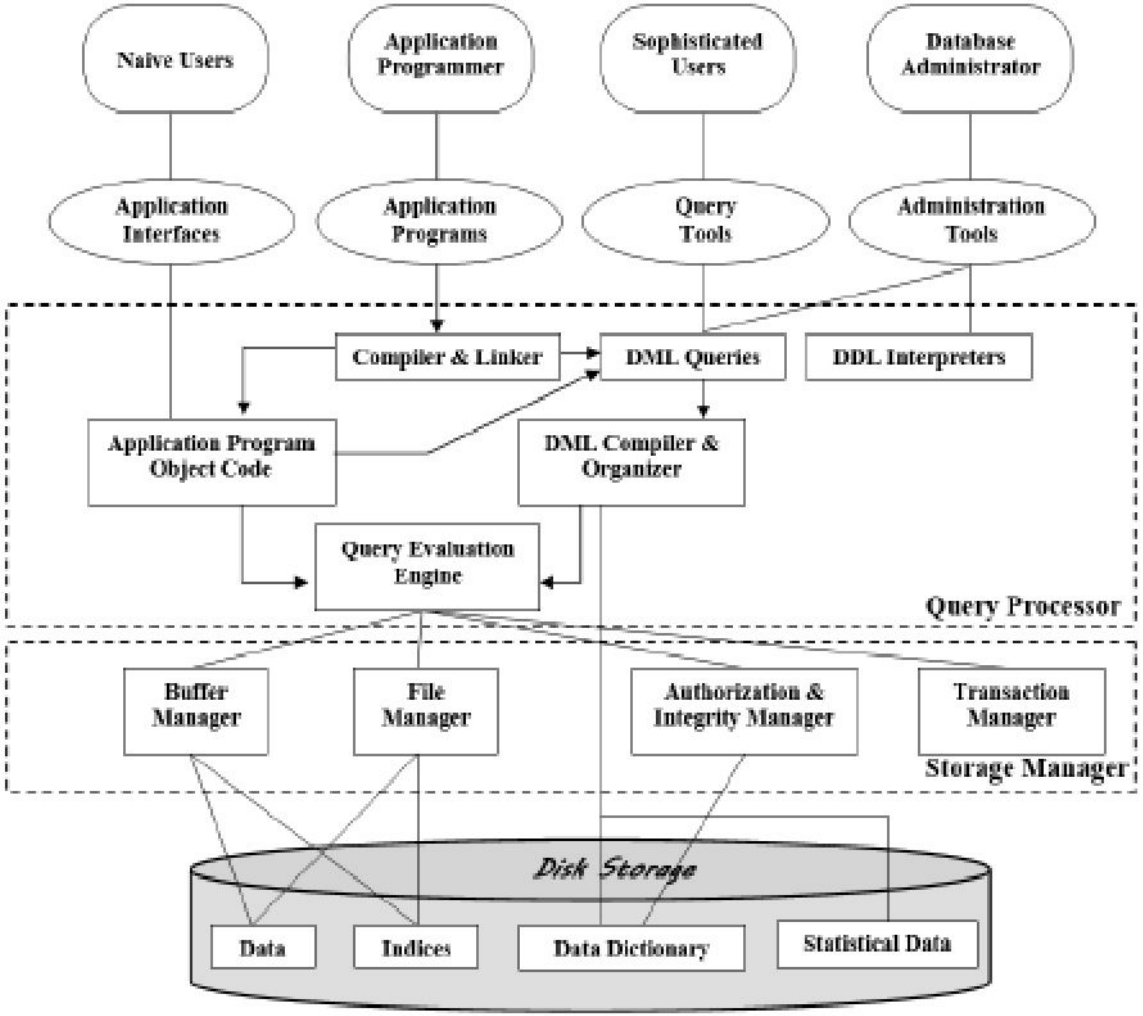


Figure: System Architecture

Database Architecture/Database System Structure

Database Users

There are four different types of database system users differentiated by the interfaces they use.

1. **Naive Users**- These are unsophisticated users who interact with the system by making use of one of the application program which is written by application programmer.

Ex- In order to transfer some amount from account A to B the bank teller in a bank will simply open an application interface of transfer amount and fill up the necessary details required to transfer the specific amount from account A to B.

The typical interface these people use is a forms interface.

2. **Application Programmers** – These are computer professionals who write application programs. These people choose many tools to develop user interfaces

Database Architecture/Database System Structure

Database Users

3. **Sophisticated Users-** These people interact with the system without writing programs. They form their request in a database query language by submitting queries to query processor. Mainly analysts fall under this category.

4. **Specialized Users-** These are sophisticated users who write specialized database applications. Some of the applications are computer aided design systems, knowledge based and expert systems and systems that store data with a complex data type.

Database Architecture/Database System Structure

Database Administrator (DBA)

DBA is concerned with central control of data and programs that access data. A person who has such control over the system is called as DBA.

Functions of DBA

1. Schema definition
2. Storage structure and access method definition
3. Schema and physical organization modification
4. Granting of authorization for data access

Database Architecture/Database System Structure

5. Routine maintenance

1. Backing up of data either onto tapes or on remote servers to prevent loss of data in
case of disasters such as flood, earthquake etc.
2. Ensuring enough disk space available for normal operations and upgrading disk space as required.
3. Monitoring jobs running on database and ensuring that performance is not degraded by very expensive tasks submitted by some users.
6. **Integrity constraint specification.**

Database Architecture/Database System Structure

Data Storage and Querying

A database system is partitioned into modules that deal with each of the responsibilities of the overall system. The functional components of a database system can be broadly divided into the storage manager and the query processor components.

The storage manager is important because a database will require a huge amount of storage space. Since main memory stores less data and hence data moved between disk and main memory.

Database system structures the data so as to minimize the need to move data between disk and main memory. Query processor helps database system simplify and facilitate access to data.

Database Architecture/Database System Structure

Storage Manager

It is a program module which acts as an interface between low level data stored in database and application program and queries submitted to system. Raw data stored on disk by using file system. It translates various DML statements into low level file system commands. It is also responsible for storing, retrieving and updating data in database.

It includes

1. Authorization and integrity manager
2. Transaction Manager
3. File Manager
4. Buffer Manager

Database Architecture/Database System Structure

Storage Manager Components

1. **Authorization and integrity manager**

It tests for satisfactions of integrity constraints and check authority of users to access data.

2. **Transaction Manager**

It ensures that database remains in an consistent state despite system failures and that concurrent transactions execution proceed without conflict.

3. **File Manager**

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

4. **Buffer Manager**

Responsible for fetching data from disk storage into main memory and deciding what data to cache in main memory. It's a critical part of database.

Database Architecture/Database System Structure

Data Components

Storage manager implements several data structures as part of the physical system implementation

1. **Data files**-Which stores the database itself.

2. **Data dictionary**-It is a metadata file, which stores the database schema. Since it is accessed very frequently a great emphasis needs to be placed on its design for efficiencies access of the metadata

3. **Indices**-Provides fast access to data items.

4. **Statistical** Data-It stores statistical information about processing of

Database Architecture/Database System Structure

Query Processor Components

1. **DDL Interpreter**–This interprets DDL statements and records the definitions in data dictionary.

2**. DML Compiler and Organizer**-It translates DML statements into low level instructions that are understood by the query evaluation engine understands. It also performs query optimization. One of the inputs for Query Optimization is the statistics gathered from the execution of previous queries

3. **Query Evaluation Engine**-Executes low level instructions generated by DML compiler and produce results.

4.Compiler and Linker- The Program written in fourth generation language(4GL) like PL/SQL will have the DML Queries embedded in the programs. The compiler compiles the programs and it interacts with the DML compiler to generate object code. This code could be directly executed or it could be saved for later execution as and when required

Database Design and ER Model

Entity

In a database, we would be grouping only related data together and storing them under one group name called Entity / Table.

This helps in identifying which data is stored where and under what name. It reduces the time to search for a particular data in a whole database.

Say, we are creating a School database. Then what all things come into our mind? It is school, students, teachers, subjects, class etc. If you observe what we have listed are items/names which are part of school, but each of them having their own group of related information? i.e.; Student has ID, name, address, DOB, class in which he is studying etc. for a Student.

Similarly, TEACHERS will have their ID, name, address subjects that they are teaching, class which they are teaching etc.

Hence each entity forms a group of related information. In real world each and every object forms an entity.

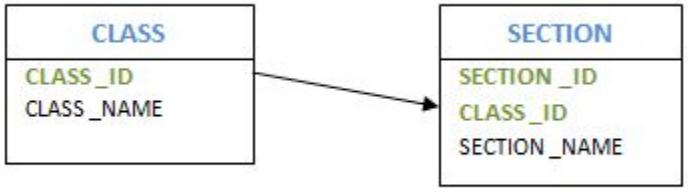
Database Design and ER Model

Entity

In short, all the living and non-living things in the world form entity. One can consider all nouns as entities

Strong Entity: Entities having its own attribute as primary keys are called strong entity. For example, STUDENT has STUDENT_ID as primary key. Hence it is a strong entity.

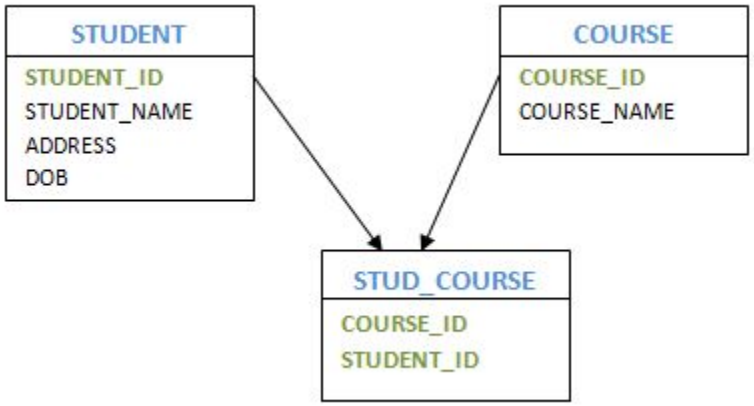
Weak Entity: Entities which cannot form their own attribute as primary key are known weak entities. These entities will derive their primary keys from the combination of its attribute and primary key from its mapping entity.



Database Design and ER Model

Entity

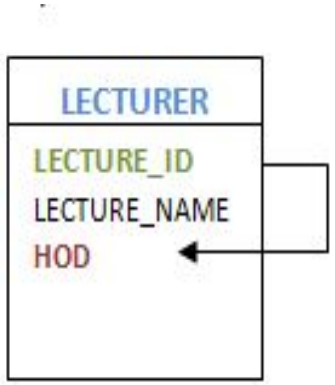
Composite Entity: Entities participating in the many to many relationships are called composite entity. In this case, apart from two entities that are part of relation, we will one more hidden entity in the relation. We will be creating a new entity with the relation, and create a primary key by using the primary keys of other two entities.



Database Design and ER Model

Entity

Recursive Entity: If a relation exists between the same entities, then such entities are called as recursive entity. For example, mapping between manager and employee is recursive entity. Here manager is mapped to the same entity Employee. HOD of the department is another example of having recursive entity.



Database Design and ER Model

Attribute

An attribute is a list of all related information of an entity, which has valid value.

It's also known as columns of the table.

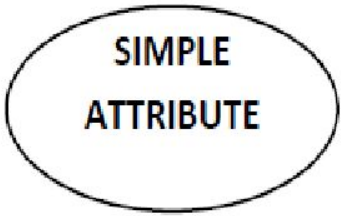
An attribute can have single value or multiple value or range of values. In addition, each attribute can contain certain type of data like only numeric value, or only alphabets, or combination of both, or date or negative or positive values etc. Depending on the values that an attribute can take, it is divided into different types

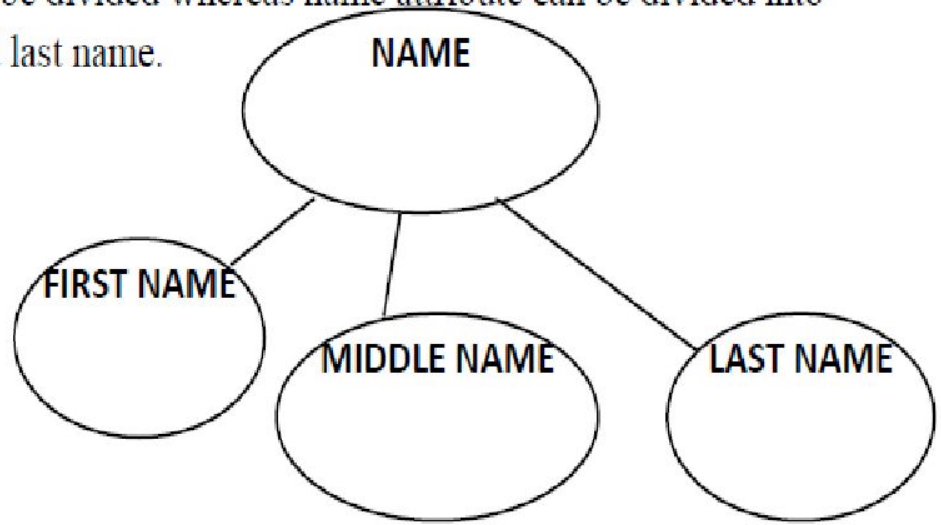
Types of Attributes

1. Simple and Composite Attribute

Simple attribute cannot be divided into sub parts whereas composite attributes can be divided into sub parts.

E.g. Students roll number attribute cannot be divided whereas name attribute can be divided into sub parts like first name, middle name and last name.



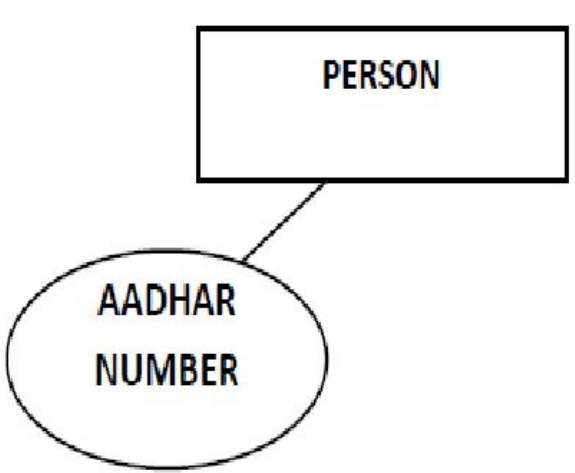


Composite Attribute

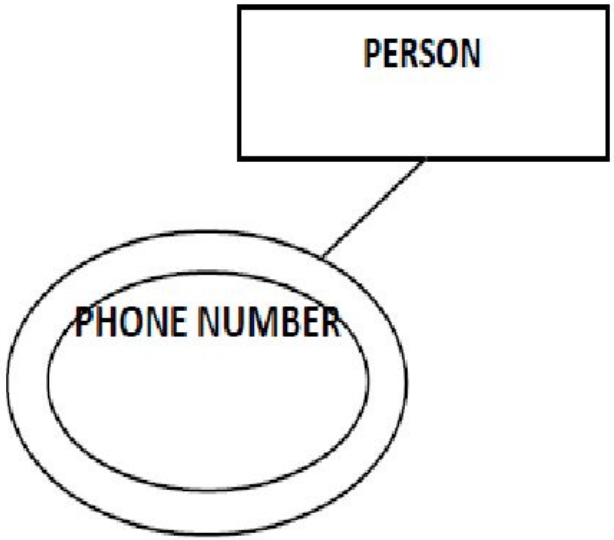
2. Single valued and multivalued attribute

The attribute which has single value is called single valued whereas attribute with multiple values can be called multivalued attribute.

E.g. Aadhar card is unique for a person and which has a single value whereas a person may have zero, one or two phone numbers.



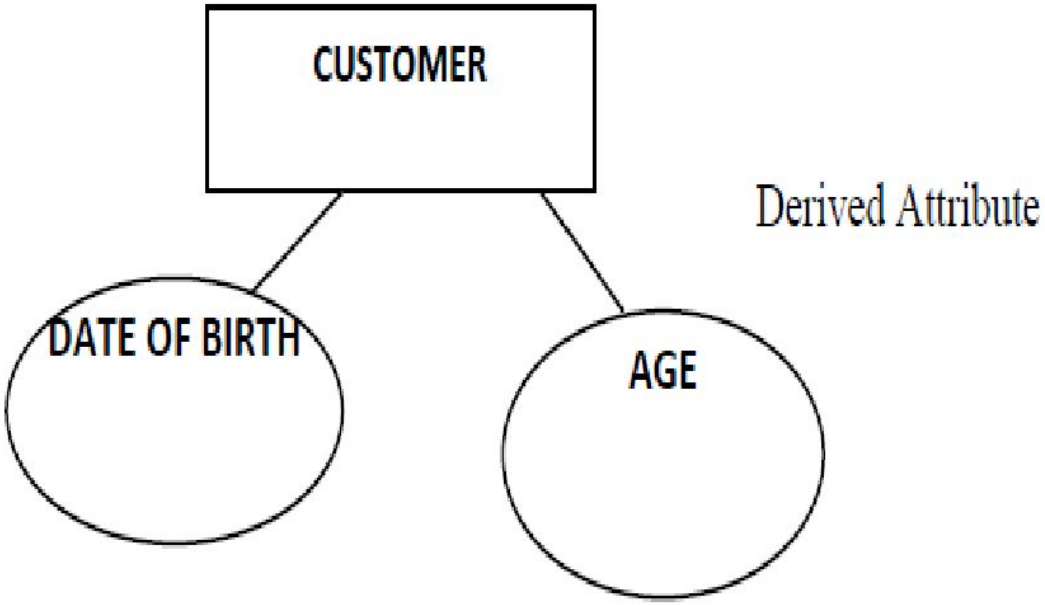
Single valued attribute



Multivalued attribute

3. Derived Attribute

The attribute which is derived or which is based on another attribute is called derived attribute.



Database Design and ER Model

Types of Attribute

Simple Attribute

These kinds of attributes have values which cannot be divided further. For example, STUDENT_ID attribute which cannot be divided further. Passport Number is unique value and it cannot be divided.

Composite Attribute

This kind of attribute can be divided further to more than one simple attribute. For example, address of a person. Here address can be further divided as Door#, street, city, state and pin which are simple attributes.

Derived Attribute

Derived attributes are the one whose value can be obtained from other attributes of entities in the database. For example, Age of a person can be obtained from date of birth and current date. Average salary, annual salary, total marks of a student etc are few examples of derived attribute.

Database Design and ER Model

Types of Attribute

Stored Attribute

The attribute which gives the value to get the derived attribute are called Stored Attribute. In example above, age is derived using Date of Birth. Hence Date of Birth is a stored attribute.

Single Valued Attribute

These attributes will have only one value. For example, EMPLOYEE_ID, passport#, driving license#, SSN etc have only single value for a person.

Multi-Valued Attribute

These attribute can have more than one value at any point of time. Manager can have more than one employee working for him, a person can have more than one email address, and more than one house etc is the examples.

Database Design and ER Model

Types of Attribute

Simple Single Valued Attribute

This is the combination of above four types of attributes. An attribute can have single value at any point of time, which cannot be divided further. For example, EMPLOYEE_ID – it is single value as well as it cannot be divided further.

Simple Multi-Valued Attribute

Phone number of a person, which is simple as well as he can have multiple phone numbers is an example of this attribute.

Composite Single Valued Attribute

Date of Birth can be a composite single valued attribute. Any person can have only one DOB and it can be further divided into date, month and year attributes.

Database Design and ER Model

Types of Attribute

Composite Multi-Valued Attribute

Shop address which is located two different locations can be considered as example of this attribute.

Descriptive Attribute

Attributes of the relationship is called descriptive attribute. For example, employee works for department. Here ‘works for’ is the relation between employee and department entities. The relation ‘works for’ can have attribute DATE_OF_JOIN which is a descriptive attribute.

Relationships

A relationship defines how two or more entities are inter-related. For example, STUDENT and CLASS entities are related as 'Student X **studies** in a Class Y'. Here 'Studies' defines the relationship between Student and Class. Similarly, Teacher and Subject are related as 'Teacher A **teaches** Subject B'. Here 'teaches' forms the relationship between both Teacher and Subject.

Degrees of Relationship

In a relationship two or more number of entities can participate. The number of entities who are part of a particular relationship is called degrees of relationship. If only two entities participate in the mapping, then degree of relation is 2 or binary. If three entities are involved, then degree of relation is 3 or ternary. If more than 3 entities are involved then the degree of relation is called n-degree or n-nary.

Relationships

Cardinality of Relationship

How many number of instances of one entity is mapped to how many number of instances of another entity is known as cardinality of a relationship. In a 'studies' relationship above, what we observe is only one Student X is studying in on Class Y. i.e.; single instance of entity student mapped to a single instance of entity Class. This means the cardinality between Student and Class is 1:1.

Based on the cardinality, there are 3 types of relationship.

One-to-One (1:1)

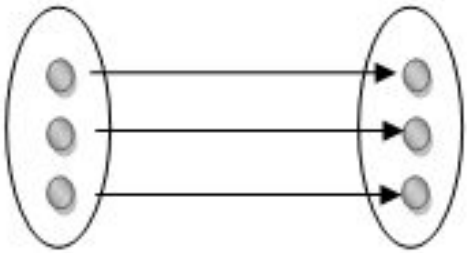
One-to-Many (1: M)

Many-to-Many (M: N)

Relationships

One-to-One (1:1): one instance of the entity is mapped to only one instance of another entity.

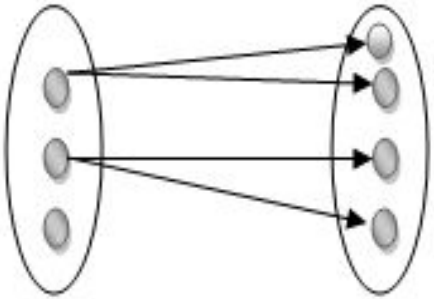
Consider, HOD of the Department. There is only one HOD in one department. That is there is 1:1 relationship between the entity HOD and Department.



Database Design and ER Model

Relationships

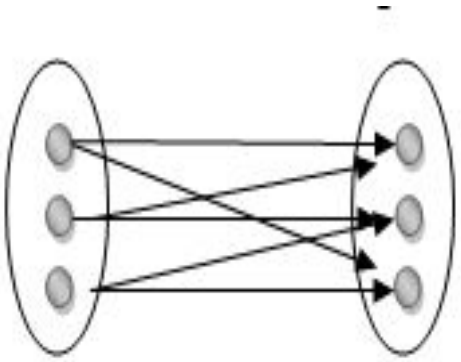
One-to-Many (1: M): one to many relationship has one instance of entity related to multiple instances of another entity. One manager manages multiple employees in his department. Here Manager and Employee are entities, and the relationship is one to many. Similarly, one teacher teaches multiple classes is also a 1: M relationship



Database Design and ER Model

Relationships

Many-to-Many (M: N): This is a relationship where multiple instances of entities are related to multiple instances of another entity. A relationship between TEACHER and STUDENT is many to many. How? Multiple Teachers teach multiple numbers of Students.



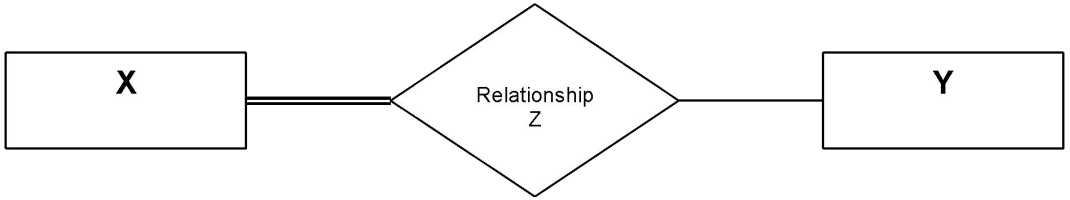
Constraints

Participation

Total Participation -

entity X has total participation in Relationship Z, meaning that every instance of X takes part in AT LEAST one relationship. (i.e. there are no members of X that do not participate in the relationship.

Example: X is Customer, Y is Product, and Z is a 'Purchases' relationship. The figure below indicates the requirement that every customer purchases a product.

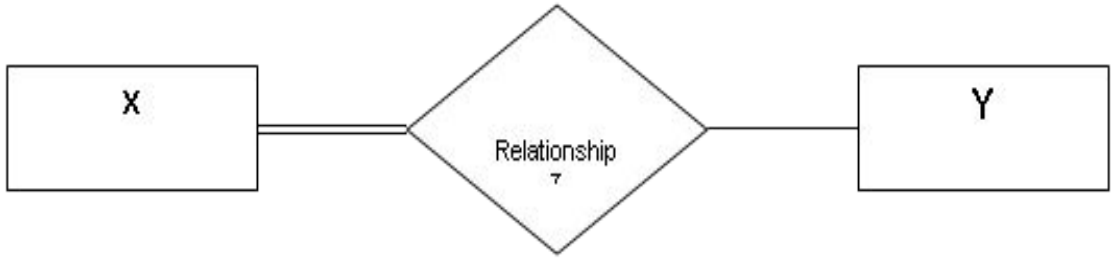


Constraints

Participation

Partial Participation - entity Y has partial participation in Relationship Z, meaning that only some instances of Y take part in the relationship.

Example: X is Customer, Y is Product, and Z is a 'Purchases' relationship. The figure below indicates the requirement that not every product is purchases by a customer. Some products may not be purchased at all.



Database Design and ER Model

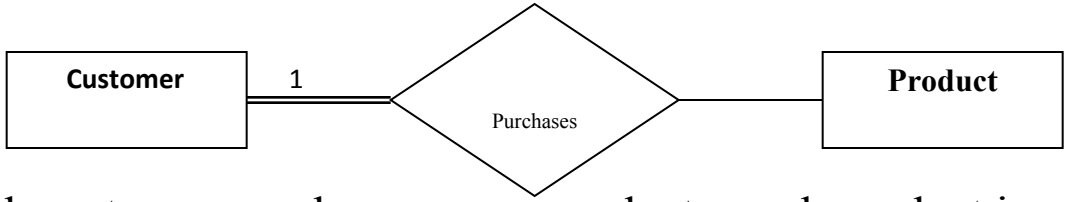
Constraints

Cardinality

1:N – One Customer buys many products, each product is purchased by only one customer

N:1 - Each customer buys at most one product, each product can be purchased by many customers.

1:1 – Each customer purchases at most one product, each product is purchased by only one customer.

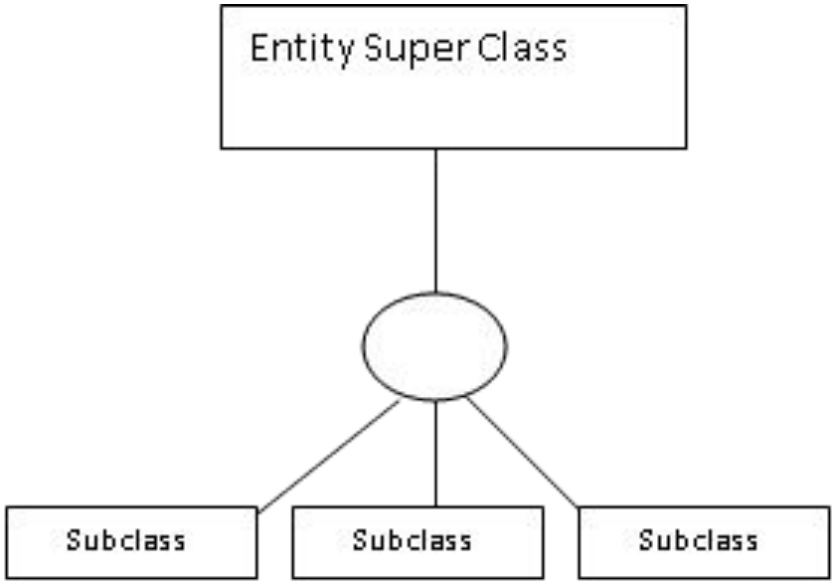


M:N – Each customer purchases many products, each product is purchased by many customers.

Constraints

Constraints on Specialization/Generalization

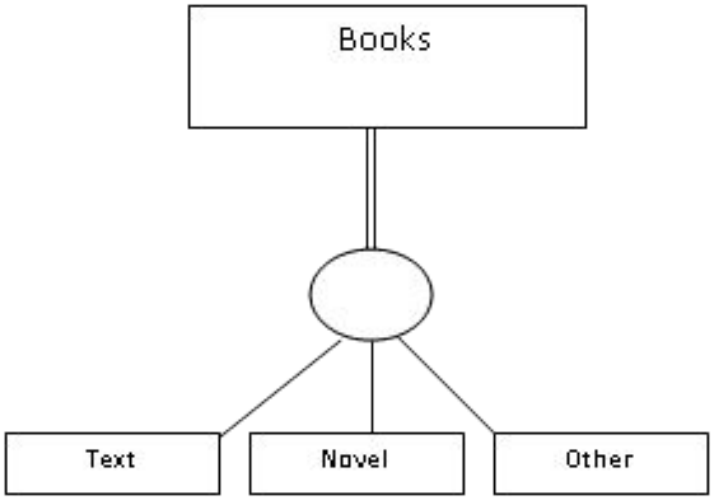
Each subclass inherits all relationships and attributes from the super-class.



Constraints

Constraints on Specialization/Generalization

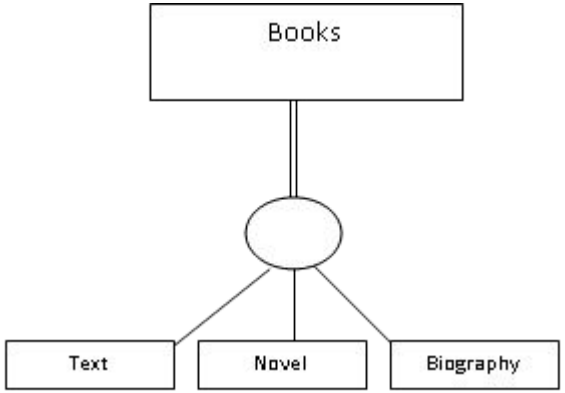
Total Specialization – Every member of the super-class must belong to at least one subclass. For example, any book that is not a text book, or a novel can fit into the "Other" category.



Constraints

Constraints on Specialization/Generalization

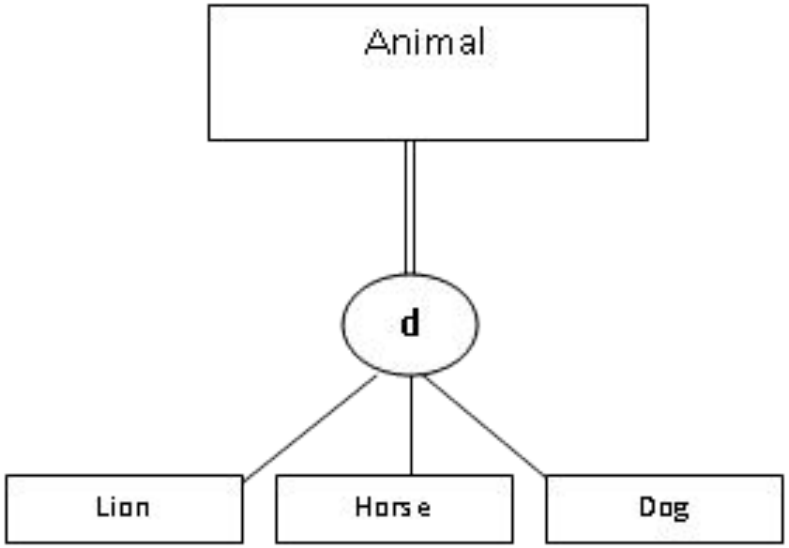
Partial Specialization – each member of the super-class may not belong to one of the subclasses. For example, a book on poetry may be neither a text book, a novel or a biography.



Constraints

Disjointness Constraint

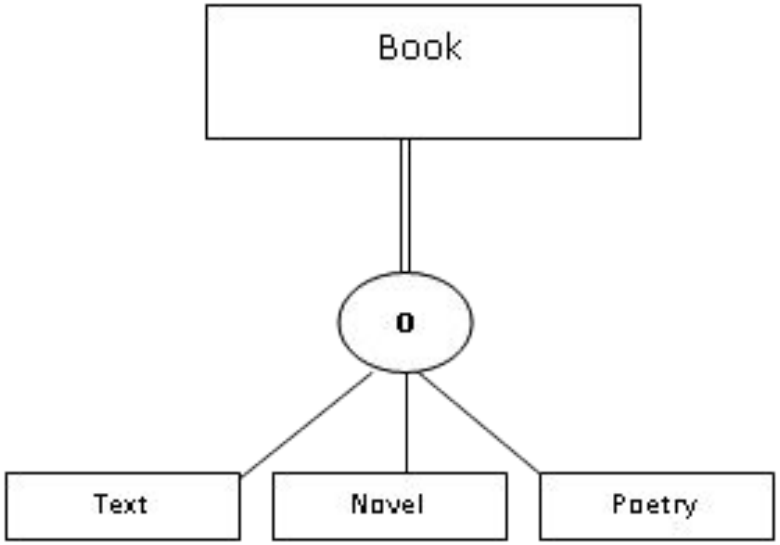
Disjoint – every member of the super-class can belong to at most one of the subclasses. For example, an Animal cannot be a lion and a horse, it must be either a lion, a horse, or a dog.



Constraints

Disjointness Constraint

Overlapping – every member of the super-class can belong to more than one of the subclasses. For example, a book can be a text book, but also a poetry book at the same time



Database Design and ER Model

Keys

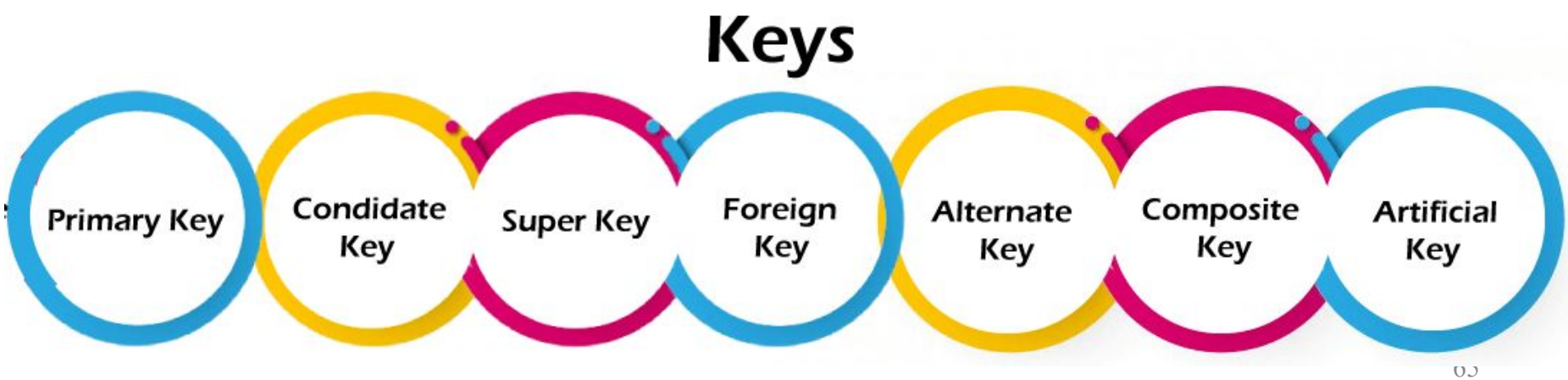
Keys play an important role in the relational database.

It is used to uniquely identify any record or row of data from the table. It is also used to establish and identify relationships between tables.

For example, ID is used as a key in the Student table because it is unique for each student. In the PERSON table, passport_number, license_number, SSN are keys since they are unique for each person.

STUDENT
ID
Name
Address
Course

PERSON
Name
DOB
Passport, Number
License_Number
SSN



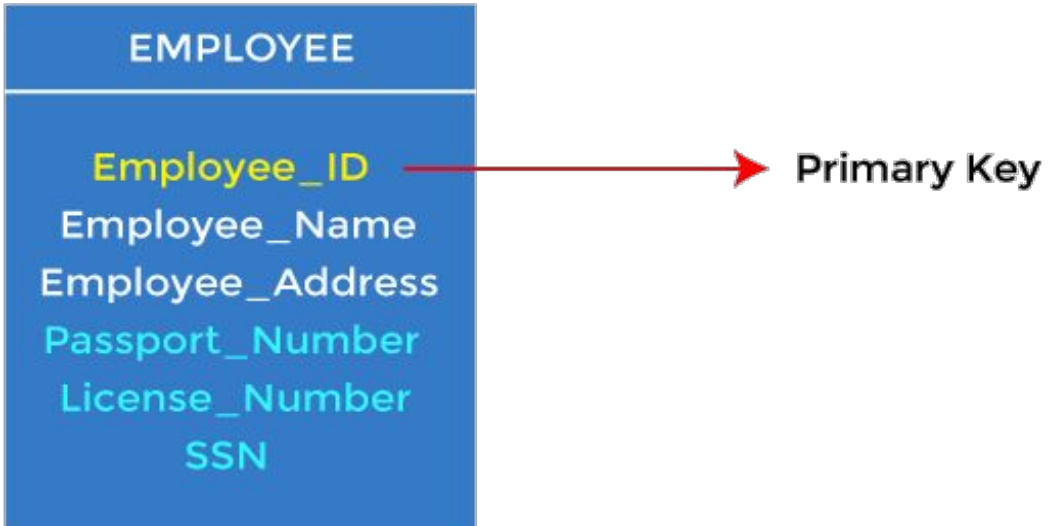
Database Design and ER Model

1. Primary key

It is the first key used to identify one and only one instance of an entity uniquely. An entity can contain multiple keys, as we saw in the PERSON table. The key which is most suitable from those lists becomes a primary key.

In the EMPLOYEE table, ID can be the primary key since it is unique for each employee. In the EMPLOYEE table, we can even select License_Number and Passport_Number as primary keys since they are also unique.

For each entity, the primary key selection is based on requirements and developers.



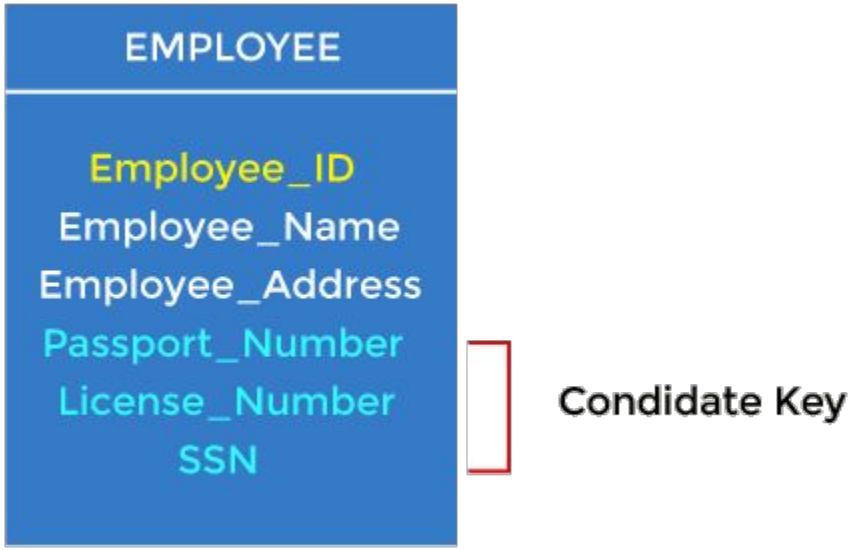
Database Design and ER Model

2. Candidate key

A candidate key is an attribute or set of attributes that can uniquely identify a tuple.

Except for the primary key, the remaining attributes are considered a candidate key. The candidate keys are as strong as the primary key.

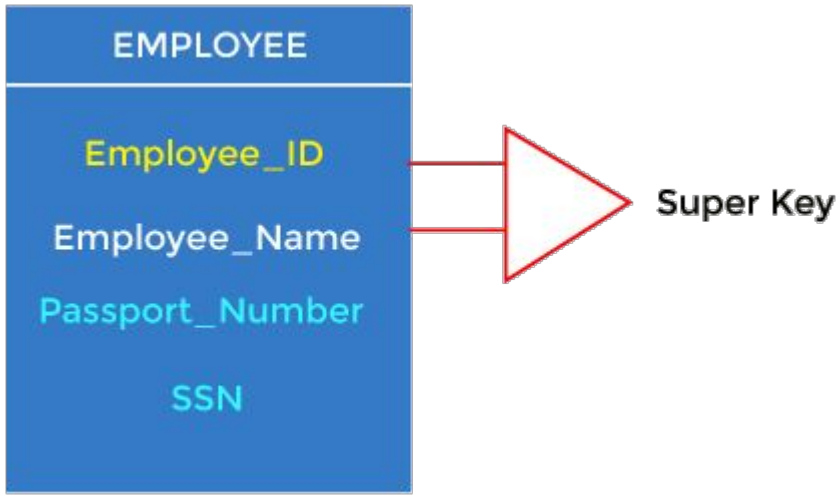
For example: In the EMPLOYEE table, id is best suited for the primary key. The rest of the attributes, like SSN, Passport_Number, License_Number, etc., are considered a candidate key.



Database Design and ER Model

3. Super Key

Super key is an attribute set that can uniquely identify a tuple. A super key is a superset of a candidate key.



For example: In the above EMPLOYEE table, for(EMPLOEE_ID, EMPLOYEE_NAME), the name of two employees can be the same, but their EMPLYEE_ID can't be the same. Hence, this combination can also be a key.

The super key would be EMPLOYEE-ID (EMPLOYEE_ID, EMPLOYEE-NAME), etc.

Database Design and ER Model

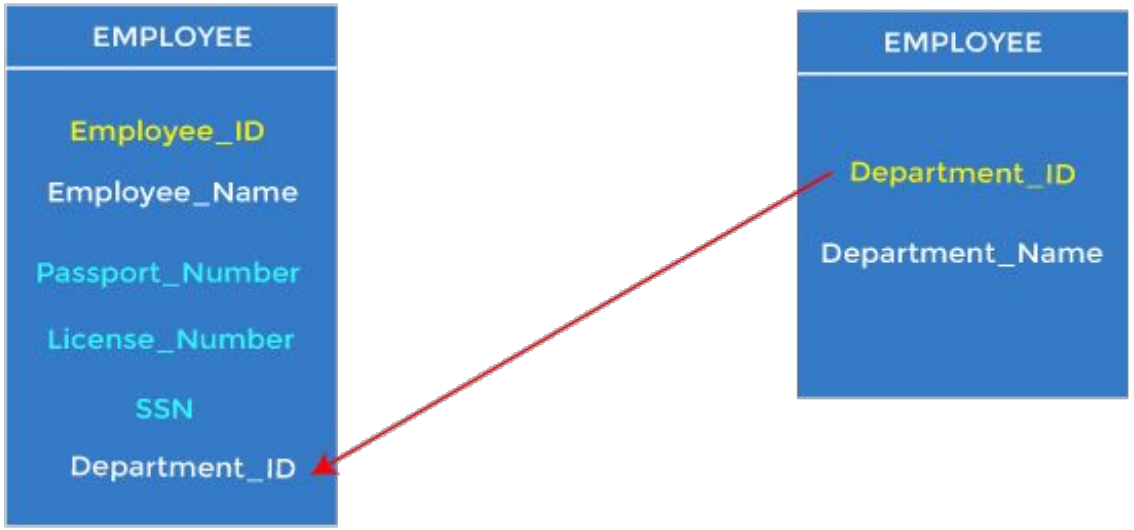
4. Foreign key

Foreign keys are the column of the table used to point to the primary key of another table.

Every employee works in a specific department in a company, and employee and department are two different entities. So we can't store the department's information in the employee table. That's why we link these two tables through the primary key of one table.

We add the primary key of the DEPARTMENT table, Department_Id, as a new attribute in the EMPLOYEE table.

In the EMPLOYEE table, Department_Id is the foreign key, and both the tables are related.

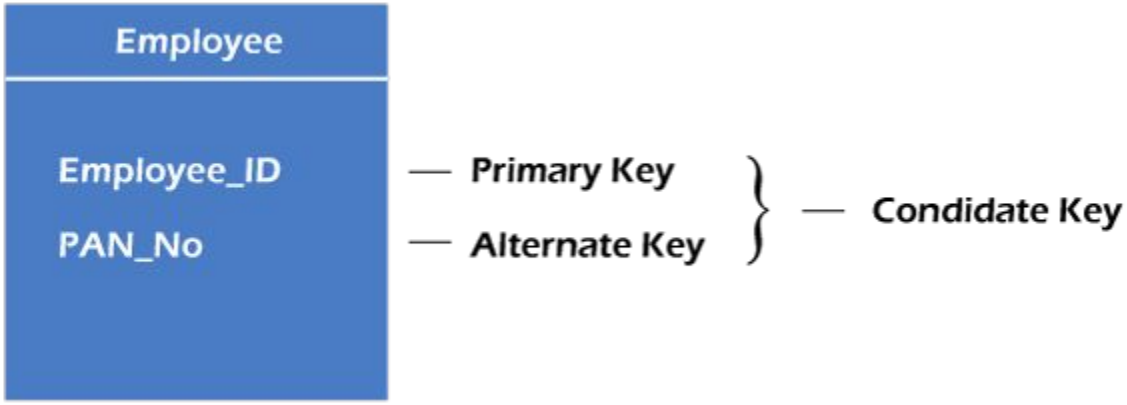


Database Design and ER Model

5. Alternate key

There may be one or more attributes or a combination of attributes that uniquely identify each tuple in a relation. These attributes or combinations of the attributes are called the candidate keys. One key is chosen as the primary key from these candidate keys, and the remaining candidate key, if it exists, is termed the alternate key. **In other words,** the total number of the alternate keys is the total number of candidate keys minus the primary key. The alternate key may or may not exist. If there is only one candidate key in a relation, it does not have an alternate key.

For example, employee relation has two attributes, Employee_Id and PAN_No, that act as candidate keys. In this relation, Employee_Id is chosen as the primary key, so the other candidate key, PAN_No, acts as the Alternate key.

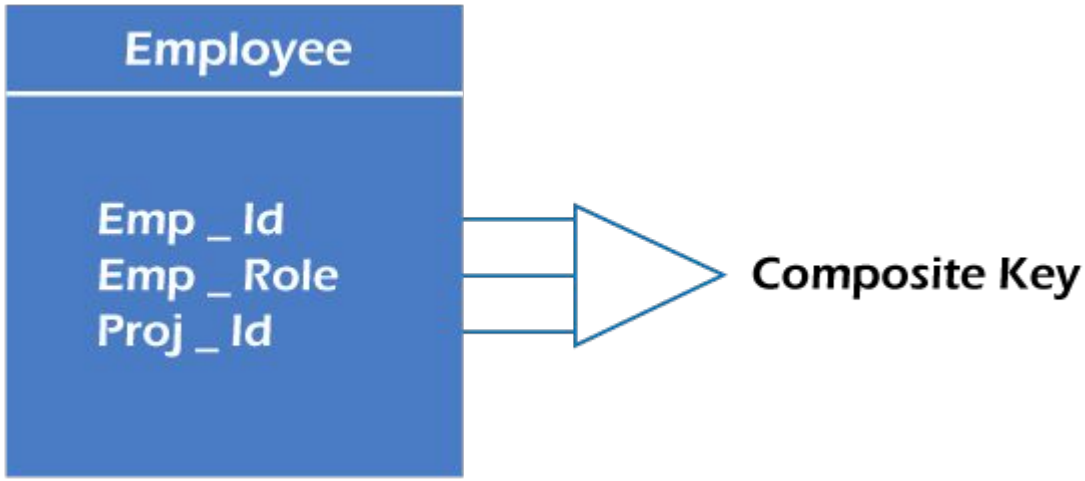


Database Design and ER Model

6. Composite key

Whenever a primary key consists of more than one attribute, it is known as a composite key. This key is also known as Concatenated Key.

For example, in employee relations, we assume that an employee may be assigned multiple roles, and an employee may work on multiple projects simultaneously. So the primary key will be composed of all three attributes, namely Emp_ID, Emp_role, and Proj_ID in combination. So these attributes act as a composite key since the primary key comprises more than one attribute.

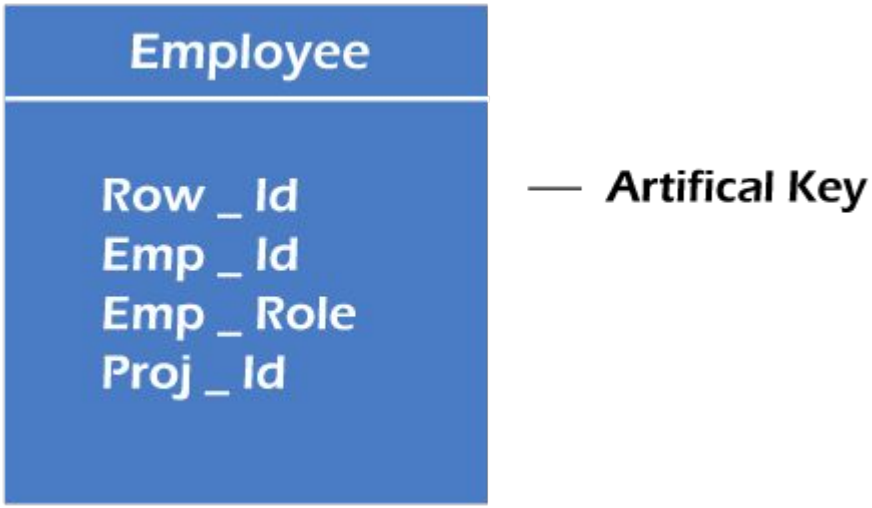


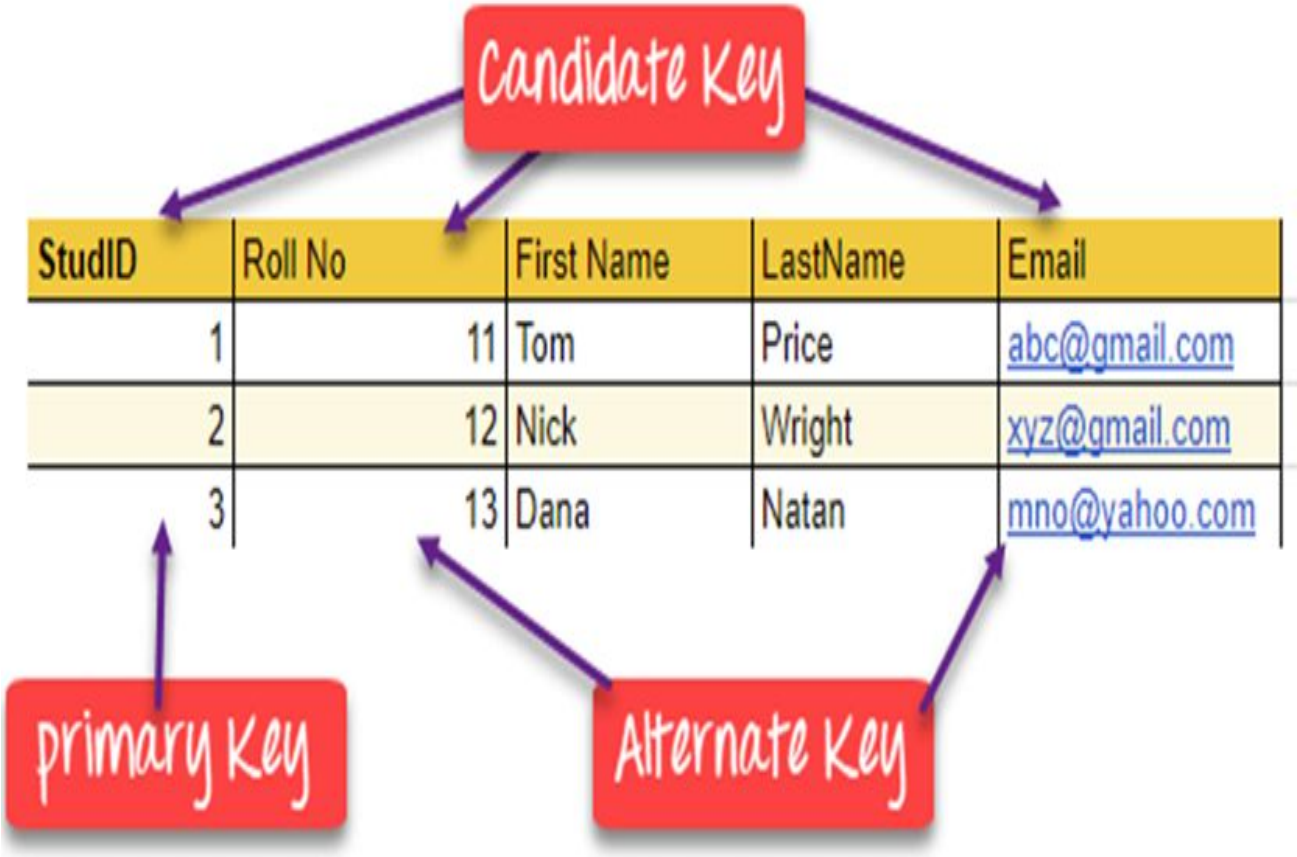
Database Design and ER Model

7. Artificial key

The key created using arbitrarily assigned data are known as artificial keys. These keys are created when a primary key is large and complex and has no relationship with many other relations. The data values of the artificial keys are usually numbered in a serial order.

For example, the primary key, which is composed of Emp_ID, Emp_role, and Proj_ID, is large in employee relations. So it would be better to add a new virtual attribute to identify each tuple in the relation uniquely.




Candidate Key
StudID
Roll No
First Name
LastName
Email
1
11
Tom
Price
abc@gmail.com
2
12
Nick
Wright
xyz@gmail.com
3
13
Dana
Natan
mno@yahoo.com
primary Key
Alternate Key

Database Design and ER Model

Design Process

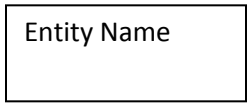
Database design can be generally defined as a collection of tasks or processes that enhance the designing, development, implementation, and maintenance of enterprise data management system.

The main objectives behind database designing are to produce physical and logical design models of the proposed database system.

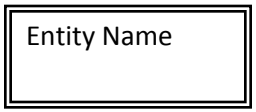
The physical database design model includes a translation of the logical design model of the database by keep control of physical media using hardware resources and software systems such as Database Management System (DBMS).

Entity Relationship Model

Strong Entities



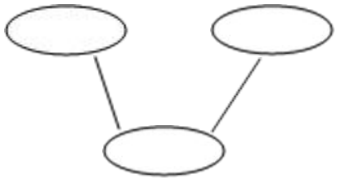
Weak Entities



Attributes

Multi Valued Attributes

Composite Attributes



Relationships

Database Design and ER Model

Entity Relationship Model

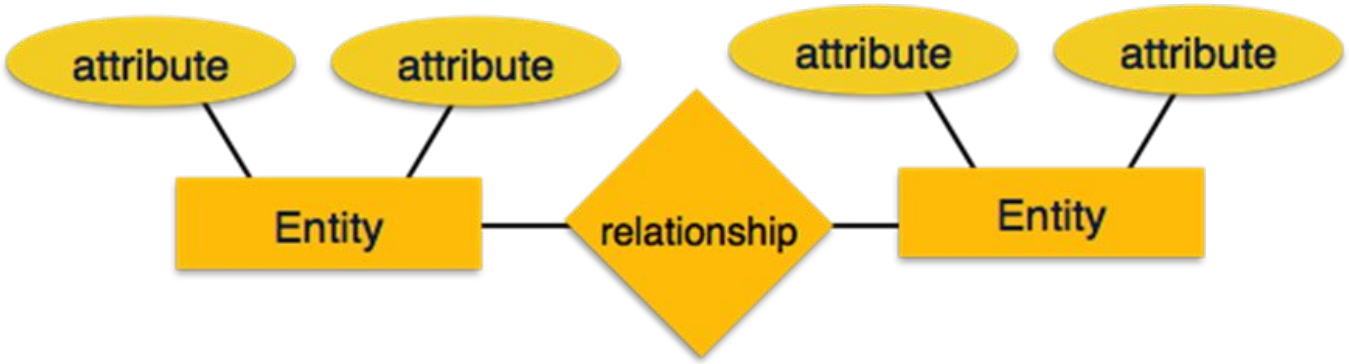
Entity-Relationship (ER) Model is based on the notion of real-world entities and relationships among them. While formulating real-world scenario into the database model, the ER Model creates entity set, relationship set, general attributes and constraints.

ER Model is best used for the conceptual design of a database.

ER Model is based on −

Entities and their *attributes*.

Relationships among entities.



Entity Relationship Model

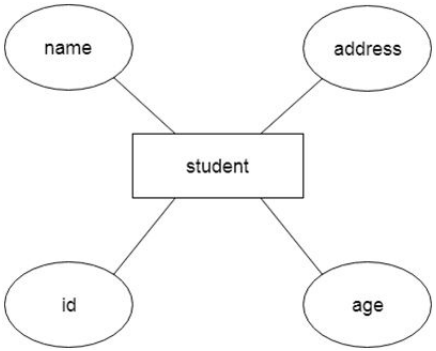
Entity − An entity in an ER Model is a real-world entity having properties called **attributes**. Every **attribute** is defined by its set of values called **domain**. For example, in a school database, a student is considered as an entity. Student has various attributes like name, age, class, etc.

Relationship − The logical association among entities is called***relationship***. Relationships are mapped with entities in various ways. Mapping cardinalities define the number of association between two entities.

Database Design and ER Model

ER Diagram 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. It also develops a very simple and easy to design view of data.
- In ER modeling, the database structure is portrayed as a diagram called an entity-relationship diagram.
- **For example,** Suppose we design a school database. In this database, the student will be an entity with attributes like address, name, id, age, etc. The address can be another entity with attributes like city, street name, pin code, etc and there will be a relationship between them.
-



Database Design and ER Model

ER Diagram

Components of the ER Diagram

This model is based on three basic concepts:

- Entities
- Attributes
- Relationships

Steps to Create an ERD

Following are the steps to create an ERD.

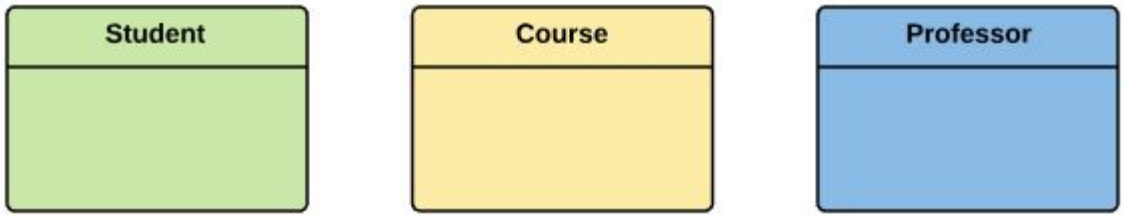
Database Design and ER Model

ER Diagram

In a university, a Student enrolls in Courses. A student must be assigned to at least one or more Courses. Each course is taught by a single Professor. To maintain instruction quality, a Professor can deliver only one course

Step 1) Entity Identification

- We have three entities
- Student
- Course
- Professor



Database Design and ER Model

Step 2) Relationship Identification

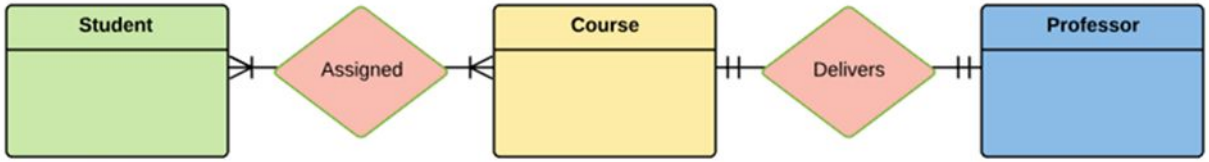
We have the following two relationships

- The student is **assigned** a course
- Professor **delivers** a course

Step 3) Cardinality Identification

For them problem statement we know that,

- A student can be assigned **multiple** courses
- A Professor can deliver only **one** course



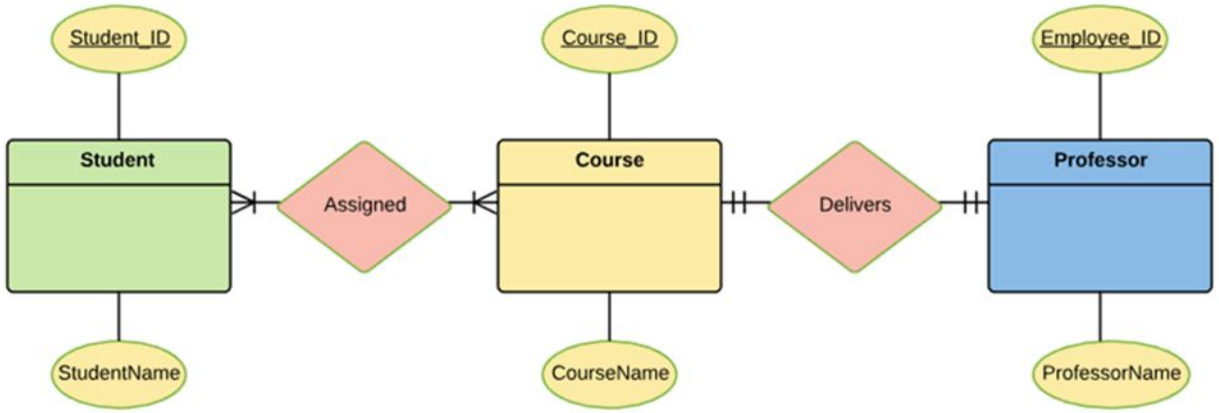
Database Design and ER Model

Step 4) Identify Attributes

You need to study the files, forms, reports, data currently maintained by the organization to identify attributes.

Once the mapping is done, identify the primary Keys. If a unique key is not readily available, create one.

Entity	Primary Key	Attribute
Student	Student_ID	StudentName
Professor	Employee_ID	ProfessorName
Course	Course_ID	CourseName



Database Design and ER Model

Step 5) Create the ERD

A more modern representation of ERD Diagram

Design Issues

Database Design and ER Model

Extended E-R Features

Enhanced entity-relationship models, also known as extended entity-relationship models, are advanced database diagrams very similar to regular ER diagrams. Enhanced ERDs are high-level models that represent the requirements and complexities of complex databases.

Features of EER Model

- EER creates a design more accurate to database schemas.
- It reflects the data properties and constraints more precisely.
- It includes all modeling concepts of the ER model.
- Diagrammatic technique helps for displaying the EER schema.
- It includes the concept of specialization and generalization.
- It is used to represent a collection of objects that is union of objects of different of different entity types.

Database Design and ER Model

Extended E-R Features

A. Sub Class and Super Class

Sub class and Super class relationship leads the concept of Inheritance.

The relationship between sub class and super class is denoted with symbol.

1. Super Class

Super class is an entity type that has a relationship with one or more subtypes.

An entity cannot exist in database merely by being member of any super class. **For example:** Shape super class is having sub groups as Square, Circle, Triangle.

2. Sub Class

Sub class is a group of entities with unique attributes. Sub class inherits properties and attributes from its super class. **For example:** Square, Circle, Triangle are the sub class of Shape super

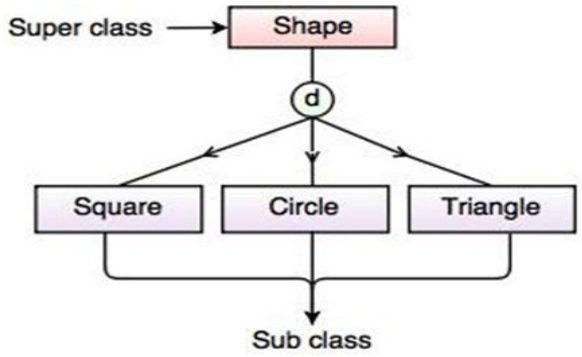


Fig. Super class/Sub class Relationship

Extended E-R Features

B. Specialization and Generalization

1. Generalization

Generalization is the process of generalizing the entities which contain the properties of all the generalized entities.

It is a bottom approach, in which two lower level entities combine to form a higher level entity.

Generalization is the reverse process of Specialization.

It defines a general entity type from a set of specialized entity type.

It minimizes the difference between the entities by identifying the common features.

For example:

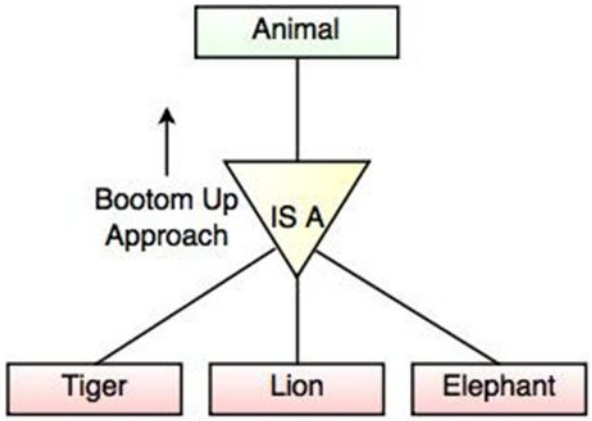


Fig. Generalization

In the above example, Tiger, Lion, Elephant can all be generalized as Animals

Extended E-R Features

2. Specialization

Specialization is a process that defines a group entities which is divided into sub groups based on their characteristic.

It is a top down approach, in which one higher entity can be broken down into two lower level entity.

It defines one or more sub class for the super class and also forms the superclass/subclass relationship.

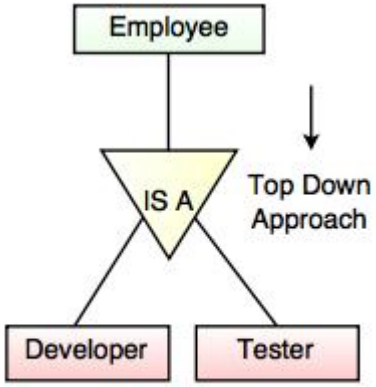


Fig. Specialization

In the above example, Employee can be specialized as Developer or Tester, based on what role they play in an Organization.

Database Design and ER Model

Extended E-R Features

C. Category or Union

Category represents a single super class or sub class relationship with more than one super class. It can be a total or partial participation.

For example

Car booking, Car owner can be a person, a bank (holds a possession on a Car) or a company. Category (sub class) → Owner is a subset of the union of the three super classes → Company, Bank, and Person. A Category member must exist in at least one of its super classes.

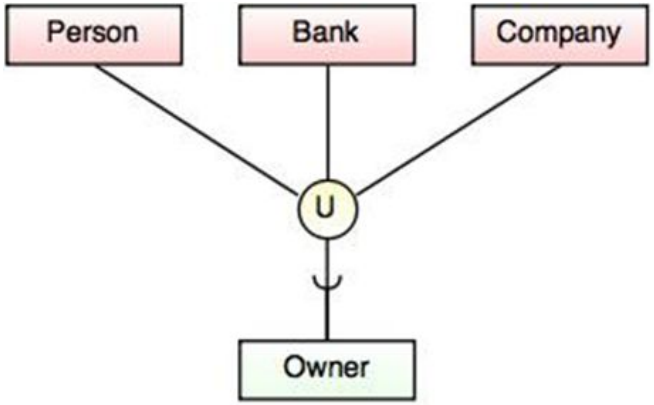


Fig. Categories (Union Type)

Database Design and ER Model

Extended E-R Features

D. Aggregation

Aggregation is a process that represent a relationship between a whole object and its component parts.

It abstracts a relationship between objects and viewing the relationship as an object.

It is a process when two entity is treated as a single entity.

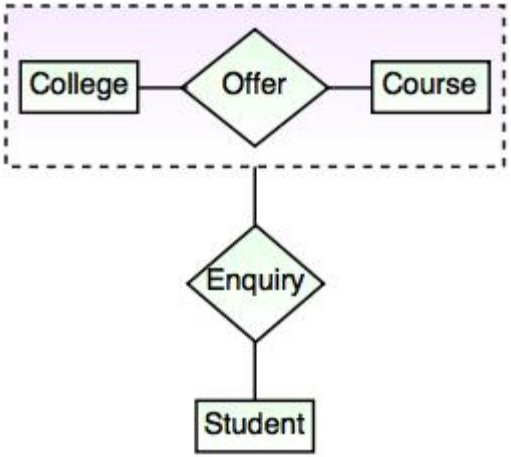


Fig. Aggregation

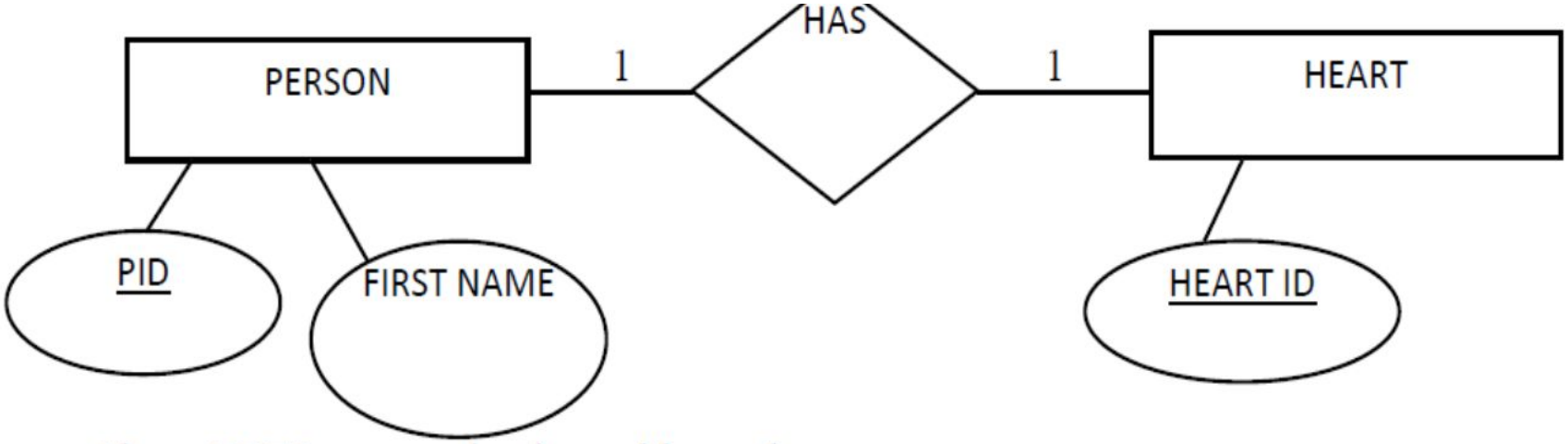
In the above example, the relation between College and Course is acting as an Entity in Relation with Student.

Converting Entity Relationship Diagram into table

1. One to One

If relationships between 2 entities (tables) are one to one then instead of creating 2 tables one can have a single table or two tables and we can adjust keys from either of the table.

e.g. If we consider the below relation which has a person with a heart then both keys can be pointing to a same record hence we can use either pid or heart id key in a single table



PID	HEART ID	FIRST NAME

OR

HEART ID	PID	FIRST NAME

AND

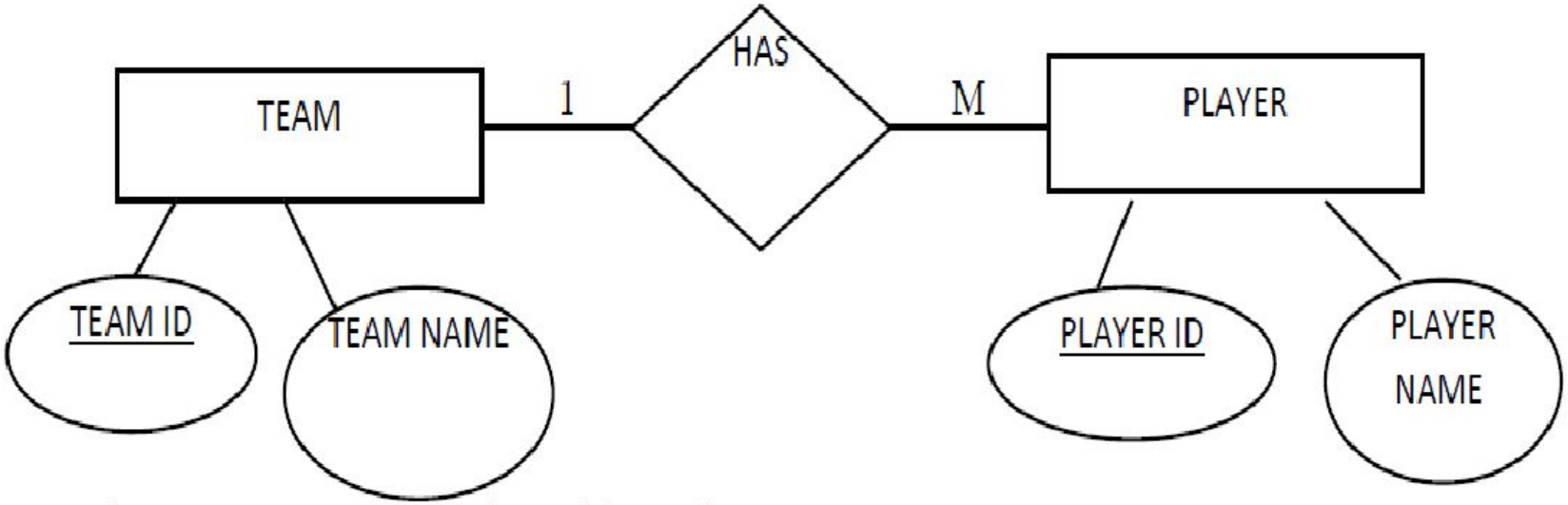
PID	FIRST NAME

2. **One to Many**

If relationships between 2 entities (tables) are one to many then in order to link 2 entities the 1's table key is copied in m's table key and treated as a foreign key.

E.g. If we consider the below relation which has a team with number of players then the 1's key

i.e. team id will be copied in the player table as a foreign key.



Above ERD's conversion into table can be

P.K. TEAM

TEAM ID	TEAM NAME

AND

P.K. PLAYER F.K.

PLAYER ID	PLAYER NAME	TEAM ID

3. Many to One

- Similar to One to Many i.e. 1's entity attributes copied in M's attributes.

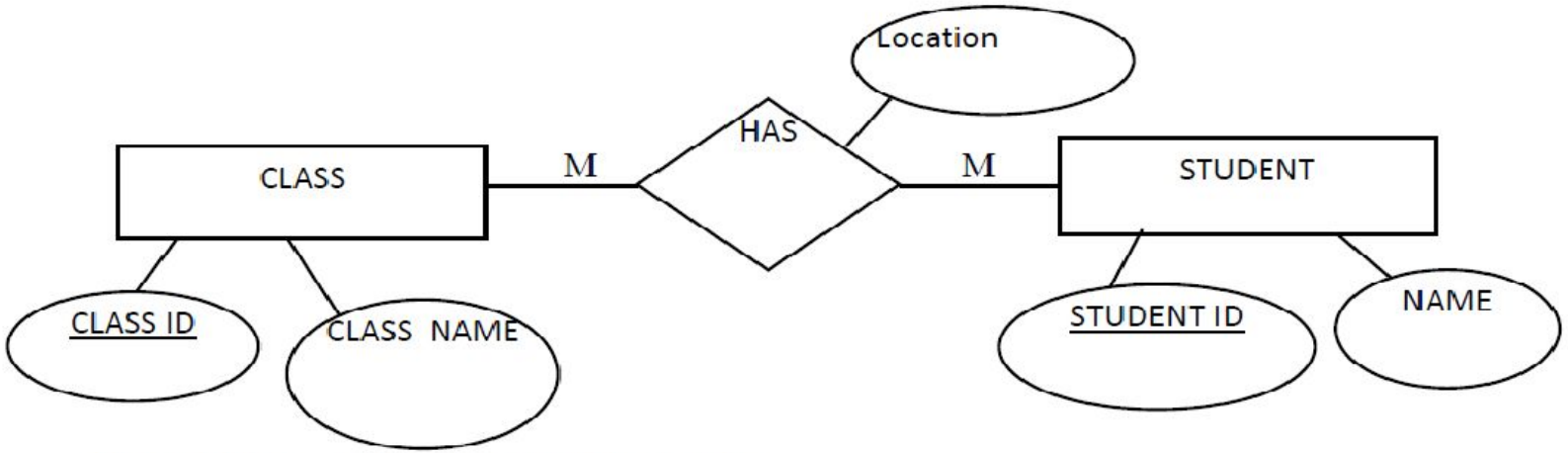
4. Many to Many

If relationships between 2 entities (tables) are many to many then a third table is created with the primary keys from both the tables and each entity is converted into its own schema i.e. tables

E.g. If we consider the below relation which has a class along with students then the first table is

created with class entity second created with student entity and a third table created by taking

into consideration keys from both the tables.



Above ERD's conversion into tables can be

P.K. CLASS

CLASS ID	CLASS NAME

P.K. STUDENT

STUDENT ID	NAME

CLASS-STUDENT

P.K. P.K.

CLASS ID	STUDENT ID	LOCATION