CHAPTER-4 DATABASE MANAGEMENT SYSTEM

What is Data?

Data is nothing but facts and statistics stored or free flowing over a network, generally it's raw and unprocessed. For example: When you visit any website, they might store your IP address, that is data, in return they might add a cookie in your browser, marking you that you visited the website, that is data, your name, it's data, your age, it's data.

Data becomes **information** when it is processed, turning it into something meaningful. Like, based on the cookie data saved on user's browser, if a website can analyse that generally men of age 20-25 visit us more, that is information, derived from the data collected.

What is a Database?

A **Database** is a collection of related data organised in a way that data can be easily accessed, managed and updated. Database can be software based or hardware based, with one sole purpose, storing data.

A database is defined as—(1) a collection, or repository of data, (2) having an organized structure, and (3) for a specific purpose. A database stores information, which is useful to an organization. It contains data based on the kind of application for which it is required. For example, an airline database may contain data about the airplane, the routes, airline reservation, airline schedules etc.; a college database may contain data about the students, faculty, administrative staff, courses, results etc.; a database for manufacturing application may contain data about the production, inventory, supply chain, orders, sales etc.; and a student database may contain data about students, like student names, student course etc.

What is DBMS?

A **DBMS** is a software that allows creation, definition and manipulation of database, allowing users to store, process and analyse data easily. DBMS provides us with an interface or a tool, to perform various operations like creating database, storing data in it, updating data, creating tables in the database and a lot more.

DBMS also provides protection and security to the databases. It also maintains data consistency in case of multiple users.

Here are some examples of popular DBMS used these days:

- MySQL
- Oracle
- SQL Server
- IBM DB2
- Postgre SQL
- Amazon Simple DB (cloud based) etc.

Characteristics of Database Management System

Data Redundancy is Minimized: Database system keeps data at one place in the database. The data is integrated into a single, logical structure. Different applications refer to the data from the centrally controlled location. The storage of the data, centrally, minimizes data redundancy.

Data Inconsistency is Reduced: Minimizing data redundancy using database system reduces data inconsistency too. Updating of data values becomes simple and there is no disagreement in the stored values. E.g. students' home addresses are stored at a single location and get updated centrally.

Data is Shared: Data sharing means sharing the same data among more than one user. Each user has access to the same data, though they may use it for different purposes. The database is designed to support shared data. Authorized users are permitted to use the data from the database. Users are provided with views of the data to facilitate its use. E.g. the students' home addresses stored in the database which is shared by student profile system and library system.

Data Independence: It is the separation of data description (metadata) from the application programs that use the data. In the database approach, data descriptions are stored in a central location called the
data dictionary. This property allows an organization's data to change and evolve (within limits) without changing the application programs that process the data.

Data Integrity is Maintained: Stored data is changed frequently for variety of reasons such asadding new data item types, and changing the data formats. The integrity and consistency of the database are protected using constraints on values that data items can have. Data constraint definitions are maintained in the data dictionary.

Data Security is Improved: The database is a valuable resource that needs protection. The database is kept secure by limiting access to the database by authorized personnel. Authorized users are generally restricted to the particular data they can access, and whether they can update it or not. Access is often controlled by passwords.

Backup and Recovery Support: Backup and recovery are supported by the software that logs changes to the database. This support helps in recovering the current state of the database in case of system failure.

Standards are Enforced: Since the data is stored centrally, it is easy to enforce standards on the database. Standards could include the naming conventions, and standard for updating, accessing and protecting data. Tools are available for developing and enforcing standards.

Application Development Time is Reduced: The database approach greatly reduces the cost and time for developing new business applications.Programmer can focus on specific functions required for the new application, without having to worry about design, or low-level implementation details; as related data have already been designed and implemented. Tools for the generation of forms and reports are also available.

Advantages of DBMS

- Segregation of application program.
- Minimal data duplicity or data redundancy.
- Easy retrieval of data using the Query Language.
- Reduced development time and maintenance need.
- With Cloud Datacenters, we now have Database Management Systems capable of storing almost infinite data.
- Seamless integration into the application programming languages which makes it very easier to add a database to almost any application or website.

Disadvantages of DBMS

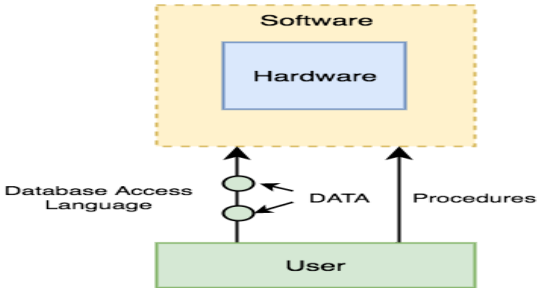
- It's Complexity
- Except MySQL, which is open source, licensed DBMSs are generally costly.
- They are large in size.

Components of DBMS

The database management system can be divided into five major components, they are:

1. Hardware
2. Software
3. Data
4. Procedures
5. Database Access Language

Let's have a simple diagram to see how they all fit together to form a database management system.



DBMS Components: Hardware
When we say Hardware, we mean computer, hard disks, I/O channels for data, and any other physical component involved before any data is successfully stored into the memory.
When we run Oracle or MySQL on our personal computer, then our computer's Hard Disk, our Keyboard using which we type in all the commands, our computer's RAM, ROM all become a part of the DBMS hardware.

DBMS Components: Software
This is the main component, as this is the program which controls everything. The DBMS software is more like a wrapper around the physical database, which provides us with an easy-to-use interface to store, access and update data.
The DBMS software is capable of understanding the Database Access Language and intrepret it into actual database commands to execute them on the DB.

DBMS Components: Data
Data is that resource, for which DBMS was designed. The motive behind the creation of DBMS was to store and utilise data.
In a typical Database, the user saved Data is present and **meta data** is stored.

Metadata is data about the data. This is information stored by the DBMS to better understand the data stored in it.
For example: When I store my **Name** in a database, the DBMS will store when the name was stored in the database, what is the size of the name, is it stored as related data to some other data, or is it independent, all this information is metadata.

DBMS Components: Procedures
Procedures refer to general instructions to use a database management system. This includes procedures to setup and install a DBMS, To login and logout of DBMS software, to manage databases, to take backups, generating reports etc.

DBMS Components: Database Access Language
Database Access Language is a simple language designed to write commands to access, insert, update and delete data stored in any database.
A user can write commands in the Database Access Language and submit it to the DBMS for execution, which is then translated and executed by the DBMS.
User can create new databases, tables, insert data, fetch stored data, update data and delete the data using the access language.

Users
- **Database Administrators:** Database Administrator or DBA is the one who manages the complete database management system. DBA takes care of the security of the DBMS, it's availability, managing the license keys, managing user accounts and access etc.
- **Application Programmer or Software Developer:** This user group is involved in developing and designing the parts of DBMS.
- **End User:** These days all the modern applications, web or mobile, store user data. How do you think they do it? Yes, applications are programmed in such a way that they collect user data and store the data on DBMS systems running on their server. End users are the one who store, retrieve, update and delete data.

Data Models, Schema and Instances

The information stored inside a database is represented using *data modeling*. The data model describes the structure of the database. A ***data model*** consists of components for describing the data, the relationships among them, and the semantics of data and the constraints that hold data. Many data models exist based on the way they describe the structure of database.

Schema is the logical structure of the database. A schema contains information about the descriptions of the database like the names of the record type, the data items within a record type, and constraints. A schema does not show the data in the database. The database schema does not change frequently.

Instances are the actual data contained in the database at a particular point of time. The content of the database may change from time to time.

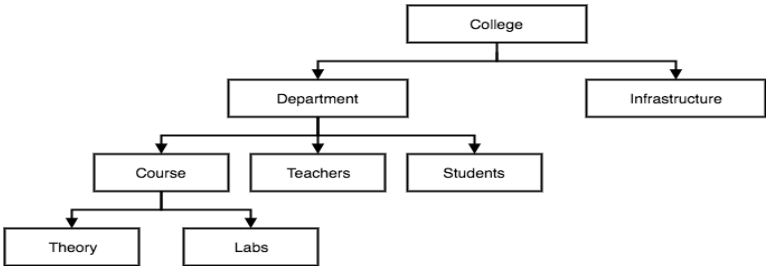
DBMS DATABASE MODELS

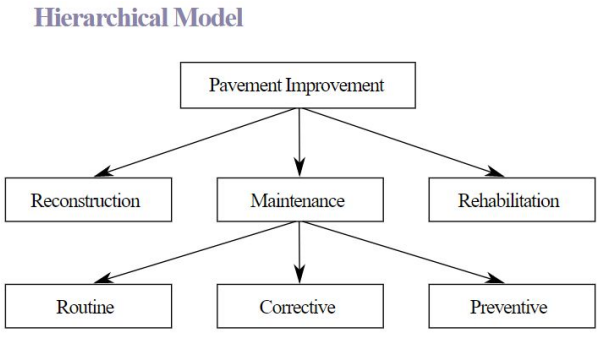
A Database model defines the logical design and structure of a database and defines how data will be stored, accessed and updated in a database management system. While the **Relational Model** is the most widely used database model, there are other models too:

- Hierarchical Model
- Network Model
- Entity-relationship Model
- Relational Model

Hierarchical Model or TREE Model

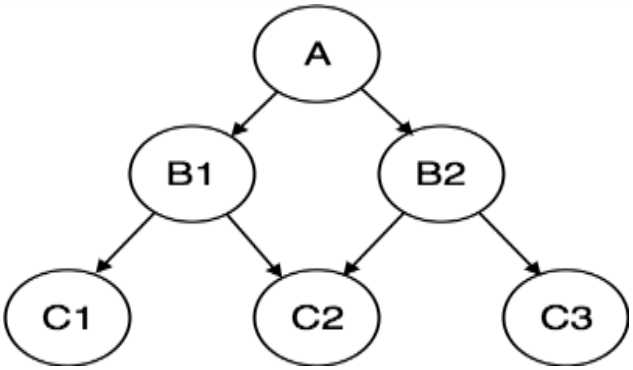
This database model organizes data into a tree-like-structure, with a single root, to which all the other data is linked. The hierarchy starts from the **Root** data, and expands like a tree, adding child nodes to the parent nodes.
In this model, **a child node will only have a single parent node.**
This model efficiently describes many real-world relationships like index of a book, recipes etc.
In hierarchical model, data is organized into tree-like structure with **one to one** and **one to many** relationship between two different types of data, for example, one department can have many courses, many professors and of-course many students. This is the **oldest model.** It was first developed by **IBM for IMS(Information Management System) record.**

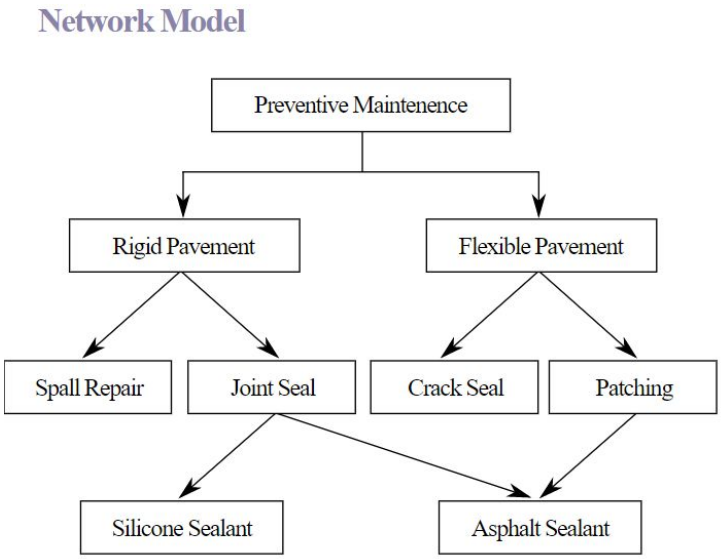




Network Model

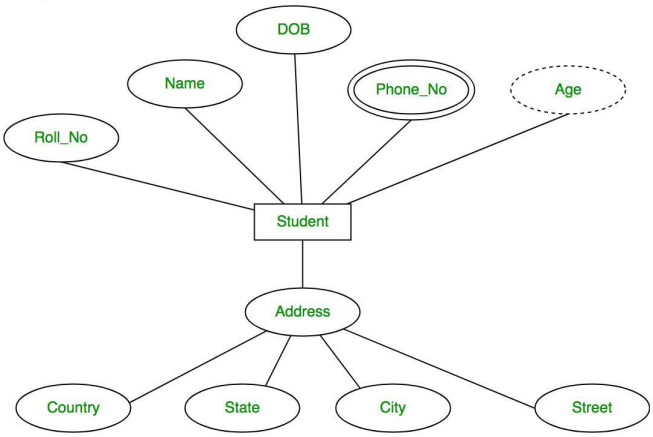
This is an extension of the Hierarchical model. In this model data is organized more like a graph, and are allowed to have more than one parent node. It gives additional many to many relationship. In this database model data is more related as more relationships are established in this database model. Also, as the data is more related, hence accessing the data is also easier and fast. This database model was used to map many-to-many data relationships.
This was the most widely used database model, before Relational Model was introduced.

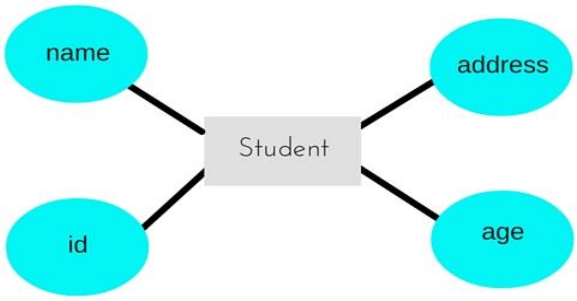




Entity-relationship Model

In this database model, relationships are created by dividing object of interest into entity and its characteristics into attributes. This shows the **logical view or conceptual model.**
Different entities are related using relationships.
E-R Models are defined to represent the **relationships into pictorial form** to make it easier for different stakeholders to understand.
This model is good to design a database, which can then be turned into tables in relational model(explained below).
Let's take an example, If we have to design a School Database, then **Student** will be an **entity** with **attributes** name, age, address etc. As **Address** is generally complex, it can be another **entity** with **attributes** street name, pincode, city etc, and there will be a relationship between them.
Relationships can also be of different types.





Relational Model

In this model, data is organised in two-dimensional **tables** and the relationship is maintained by storing a common field.

This model was introduced by E.F Codd in 1970, and since then it has been the most widely used database model, infact, we can say the only database model used around the world.

The basic structure of data in the relational model is tables. All the information related to a particular type is stored in rows of that table.

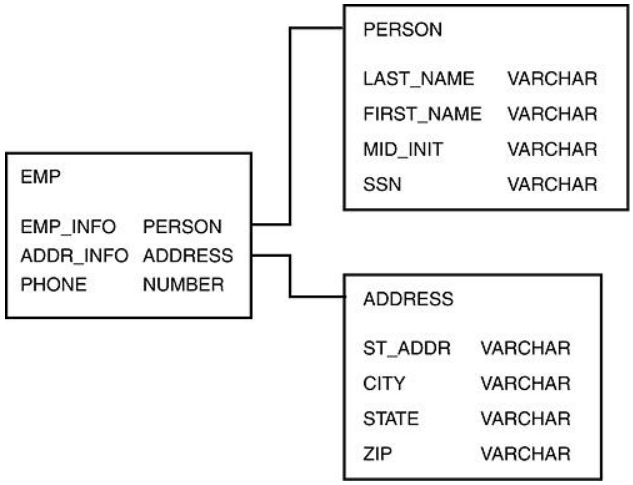
Hence, tables are also known as **relations** in relational model.

In the coming tutorials we will learn how to design tables, normalize them to reduce data redundancy and how to use Structured Query language to access data from tables.

student_id	name	age
1	Akon	17
2	Bkon	18
3	Ckon	17
4	Dkon	18

subject_id	name	teacher
1	Java	Mr. J
2	C++	Miss C
3	C#	Mr. C Hash
4	Php	Mr. P H P

student_id	subject_id	marks
1	1	98
1	2	78
2	1	76
3	2	88



- One-to-One: An entity in A is associated with at most one entity in B and vice versa.
- One-to-Many: An entity in A is associated with any number of entities in B, but an entity in B is associated with at most one entity in A.
- Many-to-One: An entity in A is associated with at most one entity in B, but an entity in B is associated with any number of entities in A.
- Many-to-Many: An entity in A is associated with any number of entities in B and vice versa.

The main highlights of this model are (ACID Properties)

Data is stored in tables called relations.

Relations can be normalized.

In normalized relations, values saved are atomic values.

Each row in a relation contains a unique value.

Each column in a relation contains values from a same domain.

INTRODUCTION TO SQL

Structure Query Language(SQL) is a database query language used for storing and managing data in Relational DBMS. SQL was the first commercial language introduced for E.F Codd's **Relational** model of database. Today almost all RDBMS(MySql, Oracle, Infomix, Sybase, MS Access) use **SQL** as the standard database query language. SQL is used to perform all types of data operations in RDBMS.

SQL Command

SQL defines following ways to manipulate data stored in an RDBMS.

DDL: Data Definition Language

This includes changes to the structure of the table like creation of table, altering table, deleting a table etc.

All DDL commands are auto-committed. That means it saves all the changes permanently in the database.

Command	Description
create	to create new table or database
alter	for alteration
truncate	delete data from table
drop	to drop a table
rename	to rename a table

DML: Data Manipulation Language

DML commands are used for manipulating the data stored in the table and not the table itself.

DML commands are not auto committed. It means changes are not permanent to database, they can be rolled back.

Command	Description
insert	to insert a new row
update	to update existing row
delete	to delete a row
merge	merging two rows or two tables

TCL: Transaction Control Language

These commands are to keep a check on other commands and their affect on the database. These commands can annul changes made by other commands by rolling the data back to its original state. It can also make any temporary change permanent.

Command	Description
commit	to permanently save
rollback	to undo change

savepoint	to save temporarily

DCL: Data Control Language

Data control language are the commands to grant and take back authority from any database user.

Command	Description
grant	grant permission of right
revoke	take back permission.

DQL: Data Query Language

Data query language is used to fetch data from tables based on conditions that we can easily apply.

Command	Description
select	retrieve records from one or more table

What is Database Design?

Database Design is a collection of processes that **facilitate the designing, development, implementation and maintenance of enterprise data management systems**

It helps produce database systems

1. That meet the requirements of the users
2. Have high performance.

The **main objectives of database designing 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 design of the database onto physical media using hardware resources and software systems such as database management systems (DBMS).

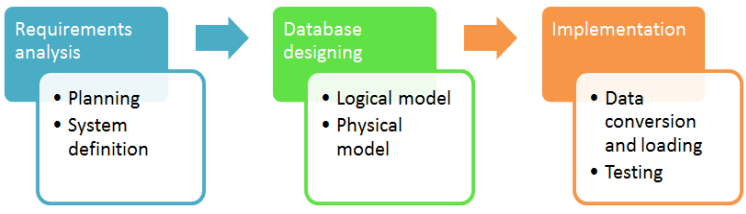
Why Database Design is Important ?

Database designing is crucial to **high performance** database system.

A**part from improving the performance, properly designed database are easy to maintain, improve data consistency and are cost effective in terms of disk storage space.**

Note , the genius of a database is in its design . Data operations using SQL is relatively simple

Database development life cycle



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 necessary have to be followed religiously in a sequential manner.
On small database systems, the database system development life cycle 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.

Requirements analysis

- **Planning** - This stages concerns 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 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.

Two Types of Database Techniques

1. **Normalization**
2. **ER Modeling**

Basic Concepts of ER Model in DBMS

As we described in the tutorial Database models, Entity-relationship model is a model used for design and representation of relationships between data.
The main data objects are termed as Entities, with their details defined as attributes, some of these attributes are important and are used to identify the entity, and different entities are related using relationships.
In short, to understand about the ER Model, we must understand about:

- Entity and Entity Set
- What are Attributes? And Types of Attributes.
- Keys
- Relationships

Let's take an example to explain everything. For a **School Management Software**, we will have to store **Student** information, **Teacher** information, **Classes**, **Subjects** taught in each class etc.

ER Model: Entity and Entity Set

Considering the above example, **Student** is an entity, **Teacher** is an entity, similarly, **Class**, **Subject** etc are also entities.
An Entity is generally a real-world object which has characteristics and holds relationships in a DBMS.
If a Student is an Entity, then the complete dataset of all the students will be the **Entity Set**

Proper Database Design can make database smaller, faster, scaleable, manageable.

ER Model: Attributes

If a Student is an Entity, then student's **roll no.**, student's **name**, student's **age**, student's **gender** etc will be its attributes.

An attribute can be of many types, here are different types of attributes defined in ER database model:

1. **Simple attribute:** The attributes with values that are atomic and cannot be broken down further are simple attributes. For example, student's **age**.
2. **Composite attribute:** A composite attribute is made up of more than one simple attribute. For example, student's **address** will contain, **house no.**, **street name**, **pincode** etc.
3. **Derived attribute:** These are the attributes which are not present in the whole database management system, but are derived using other attributes. For example, *average age of students in a class*.
4. **Single-valued attribute:** As the name suggests, they have a single value.
5. **Multi-valued attribute:** And, they can have multiple values.

ER Model: Keys

If the attribute **roll no.** can uniquely identify a student entity, amongst all the students, then the attribute **roll no.** will be said to be a key.

Following are the types of Keys:

1. Super Key
2. Candidate Key
3. Primary Key

We have covered Keys in details here in Database Keys tutorial.

ER Model: Relationships

When an Entity is related to another Entity, they are said to have a relationship. For example, A **Class** Entity is related to **Student** entity, because students study in classes, hence this is a relationship.

Depending upon the number of entities involved, a **degree** is assigned to relationships.

For example, if 2 entities are involved, it is said to be **Binary relationship**, if 3 entities are involved, it is said to be **Ternary** relationship, and so on.

In the next tutorial, we will learn how to create ER diagrams and design databases using ER diagrams.

Introduction to Database Keys

Keys are very important part of Relational database model. They are used to **establish and identify relationships between tables and also to uniquely identify any record or row of data inside a table.**

A Key can be **a single attribute or a group of attributes,** where the combination may act as a key.

Why we need a Key?

In real world applications, number of tables required for storing the data is huge, and the different tables are related to each other as well.

Also, tables store a lot of data in them. Tables generally extends to thousands of records stored in them, unsorted and unorganized.

Now to fetch any particular record from such dataset, you will have to apply some conditions, but what if there is duplicate data present and every time you try to fetch some data by applying certain condition, you get the wrong data. How many trials before you get the right data?

To avoid all this, **Keys** are defined to easily identify any row of data in a table.

Let's try to understand about all the keys using a simple example.

student_id	name	phone	age
1	Akon	9876723452	17
2	Akon	9991165674	19
3	Bkon	7898756543	18
4	Ckon	8987867898	19
5	Dkon	9990080080	17

Let's take a simple **Student** table, with field's student_id, name, phone and age.

Super Key

Super Key is defined as a **set of attributes within a table that can uniquely identify each record within a table. Super Key is a superset of Candidate key.**
In the table defined above super key would include **student_id**, **(student_id, name)**, **phone etc.**
Confused? The first one is pretty simple as student_id is unique for every row of data, hence it can be used to identity each row uniquely.
Next comes, (student_id, name), now name of two students can be same, but their student_id can't be same hence this combination can also be a key.
Similarly, phone number for every student will be unique, hence again, phone can also be a key.
So they all are super keys.

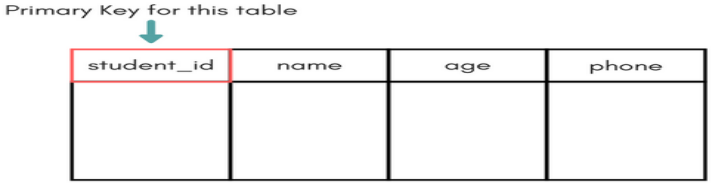
Candidate Key

Candidate keys are defined as the minimal set of fields which can uniquely identify each record in a table. It is a**n attribute or a set of attributes that can act as a Primary Key** for a table to uniquely identify each record in that table. There can be more than one candidate key.
In our example, **student_id** and **phone** both are candidate keys for table **Student**.

- A candiate key can never be NULL or empty. And its value should be unique.
- There can be more than one candidate keys for a table.
- A candidate key can be a combination of more than one columns(attributes).

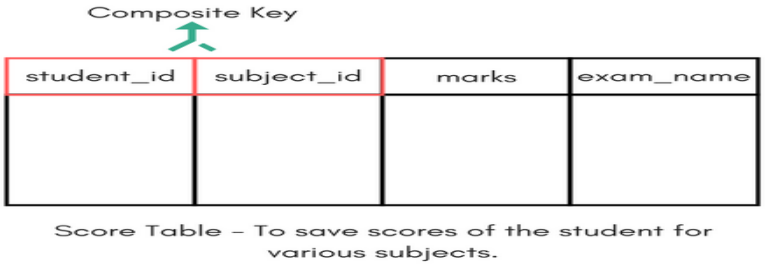
Primary Key

Primary key is a candidate key that is most appropriate to become the main key for any table. It is a key that can uniquely identify each record in a table.



For the table **Student** we can make the student_id column as the primary key.

Composite Key
Key that consists of **two or more attributes that uniquely identify any record** in a table is called **Composite key** . But the attributes which together form the **Composite key** are not a key independentely or individually.



Score Table – To save scores of the student for various subjects.

In the above picture we have a **Score** table which stores the marks scored by a student in a particular subject.
In this table student_id and subject_id together will form the primary key, hence it is a composite key.

Secondary or Alternative key
The **candidate key which are not selected as primary key** are known as secondary keys or alternative keys.
Non-key Attributes
Non-key attributes are the attributes or fields of a table, other than **candidate key** attributes/fields in a table.
Non-prime Attributes
Non-prime Attributes are attributes other than **Primary Key attribute(s).**.

Normalization:

If a database design is not perfect, it may contain anomalies, which are like a bad dream for any database administrator. Managing a database with anomalies is next to impossible.

- **Update anomalies** − If data items are scattered and are not linked to each other properly, then it could lead to strange situations. For example, when we try to update one data item having its copies scattered over several places, a few instances get updated properly while a few others are left with old values. Such instances leave the database in an inconsistent state.
- **Deletion anomalies** − We tried to delete a record, but parts of it was left undeleted because of unawareness, the data is also saved somewhere else.
- **Insert anomalies** − We tried to insert data in a record that does not exist at all.

Normalization is a method to remove all these anomalies and bring the database to a consistent state.
First Normal Form
First Normal Form is defined in the definition of relations (tables) itself. This rule defines that all the attributes in a relation must have atomic domains. The values in an atomic domain are indivisible units.

Course	Content
Programming	Java, c++
Web	HTML, PHP, ASP

We re-arrange the relation (table) as below, to convert it to First Normal Form.

Course	Content
Programming	Java
Programming	c++
Web	HTML
Web	PHP
Web	ASP

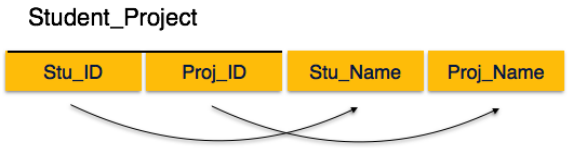
Each attribute must contain only a single value from its pre-defined domain.

Second Normal Form

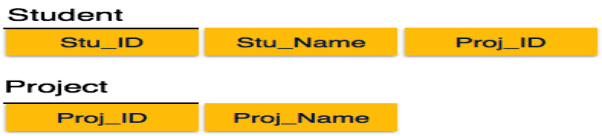
Before we learn about the second normal form, we need to understand the following −

- **Prime attribute** − An attribute, which is a part of the candidate-key, is known as a prime attribute.
- **Non-prime attribute** − An attribute, which is not a part of the prime-key, is said to be a non-prime attribute.

If we follow second normal form, then every non-prime attribute should be fully functionally dependent on prime key attribute. That is, if X → A holds, then there should not be any proper subset Y of X, for which Y → A also holds true.



We see here in Student_Project relation that the prime key attributes are Stu_ID and Proj_ID. According to the rule, non-key attributes, i.e. Stu_Name and Proj_Name must be dependent upon both and not on any of the prime key attribute individually. But we find that Stu_Name can be identified by Stu_ID and Proj_Name can be identified by Proj_ID independently. This is called **partial dependency**, which is not allowed in Second Normal Form.

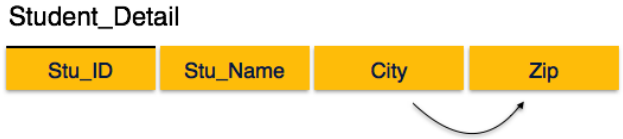


We broke the relation in two as depicted in the above picture. So there exists no partial dependency.

Third Normal Form

For a relation to be in Third Normal Form, it must be in Second Normal form and the following must satisfy −

- No non-prime attribute is transitively dependent on prime key attribute.
- For any non-trivial functional dependency, X → A, then either −
  - X is a superkey or,
  - A is prime attribute.



We find that in the above Student_detail relation, Stu_ID is the key and only prime key attribute. We find that City can be identified by Stu_ID as well as Zip itself. Neither Zip is a superkey nor is City a prime attribute. Additionally, Stu_ID → Zip → City, so there exists **transitive dependency**.

To bring this relation into third normal form, we break the relation into two relations as follows −

Boyce-Codd Normal Form
Boyce-Codd Normal Form (BCNF) is an extension of Third Normal Form on strict terms. BCNF states that −

- For any non-trivial functional dependency, X → A, X must be a super-key.

In the above image, Stu_ID is the super-key in the relation Student_Detail and Zip is the super-key in the relation ZipCodes. So,
Stu_ID → Stu_Name, Zip
and
Zip → City
Which confirms that both the relations are in BCNF.

Security Of DBMS

Security refers to activities and measures to ensure the c**onfidentiality, integrity, and availability of an information system and its main asset**, data.3 It is important to understand that securing data requires a comprehensive, company-wide approach.
• **Confidentiality** deals with ensuring that data is protected **against unauthorized access,** and if the data are accessed by an authorized user, that the data are used only for an authorized purpose. In other words, confidentiality entails safeguarding data against disclosure of any information that would violate the privacy rights of a person or organization. Data must be evaluated and classified according to the level of confidentiality: highly restricted (very few people have access), confidential (only certain groups have access), and unrestricted (can be accessed by all users).
• **Integrity,** within the data security framework, is concerned with **keeping data consistent, free of errors, or anomalies.** Integrity focuses on maintaining the data free of inconsistencies and anomalies. The DBMS plays a pivotal role in ensuring the integrity of the data in the database. However, from the security point of view, integrity deals not only with the data in the database but also with ensuring that organizational processes, users, and usage patterns maintain such integrity.
• **Availability** refers to the **accessibility of data** whenever required by **authorized users and for authorized purposes.** To ensure data availability, the entire system (not only the data component) must be protected from service degradation or interruption caused by any source (internal or external).

Security Policies:

A security policy is a collection of standards, policies, and procedures created to guarantee the security of a system and ensure auditing and compliance. The security audit process starts by identifying the security vulnerabilities in the organization's information system infrastructure and identifying measures to protect the system and data against those vulnerabilities.

Database Security:

Database security refers to the use of the DBMS features and other related measures to comply with the security requirements of the organization. From the DBA's point of view, security measures should be implemented to **protect the DBMS against service degradation and the database against loss, corruption, or mishandling**.

To protect the DBMS against service degradation there are certain minimum recommended security safeguards. For example:

- Change default system passwords.
- Change default installation paths.
- Apply the latest patches.
- Secure installation folders with proper access rights.
- Make sure only required services are running.
- Set up auditing logs.
- Set up session logging.
- Require session encryption.

Data Securing Technologies : HOW TO SECURE DATA???

What is Data Security?

In the most basic terms, Data Security is the process of keeping data secure and protected from not only unauthorized access but also corrupted access. The main focus of data security is to make sure that data is safe and away from any destructive forces. Data is stored as rows and columns in its raw form in the databases, PCs as well as over networks. While some of this data may be not that secretive, other might be of private value and importance. But unauthorized access
to such private information or data can cause many problems such as corruption, leakage of confidential information and violation of privacy.

Disk Encryption: Disk encryption is one of the most commonly opted for data security technology or methods. This is a technology through which encryption of data on a hard disk drive takes place. This technology takes place in two major ways – software or hardware. In disk encryption, data is converted into unreadable codes that cannot be accessed or deciphered by anyone who is unauthorized. There are several ways and tools to carry out disk encryption, and these tools may vary in the security offered and features used. Even though there are many benefits of using this method, there are also certain weaknesses or vulnerabilities.

Software and hardware based ways to protect data: Besides disk encryption, both software and hardware based ways can also be used to protect data. On one hand, software-based security solutions encrypt the data to protect it from theft, on the other, hardware-based solutions can prevent read and write access to data. Hardware based security solutions offer very strong protection against unauthorized access and tampering. But in the case of software-based solutions, a hacker or a malicious program can easily corrupt the data files and make the system unusable and files unreadable. This is why, hardware-based solutions are mostly preferred over software based ones. The hardware-based systems are more secure due to the physical access required to compromise them. This system is much more effective in the situation where an operating system is more vulnerable to threats from viruses and hackers.

Backups: One of the easiest yet most effective ways to avoid data loss or to lose important and crucial files is by taking a backup of your data regularly. There are many ways to take backup and it is up to you how many copies of your data you wish to keep. While external hard disks are a common way to take backup, these days cloud computing too proves to be a cheap and easy way to maintain a backup of all files at a safe location. Of course, a backup won't prevent data loss but would at least ensure that you don't lose any information of importance.

Data masking: Data masking is another data securing technology that can be brought into use by those who wish to secure their data. Another term that is used to refer to data masking is data obfuscation and is the process through which one can hide original data with random characters, data or codes. This method is especially very useful for situations where you wish to protect classified data and do not want anyone to access it or read it. This is a good way to let the data be usable to you but not to the unauthorized hacker or user.

Data erasure: Data erasure, which is only known as data wiping and data clearing is a software-based method of overwriting information or data and aims to totally destroy all data which may be present on a hard disk or any other media location. This method removes all data or information but keeps the disk operable.

Essentials Steps Every Buisness Must Take To Secure the Data

Establish strong passwords: The first step that every business must most take is to establish strong passwords for all your accounts, bank details and other kinds of accounts. Also, one must try to keep very strong passwords that may not be easily guessed by anyone. The passwords must be a combination of characters and numbers. The password must be easy to remember for you but should not be your birthday, your name, or any other personal detail that anyone else could guess.
The password must be between 8-12

Strong Firewall: Like antiviruses are for your files, firewalls are for protection. You must establish a strong firewall in order to protect your network from unauthorized access or usage. The firewall protects your network by controlling internet traffic that comes into and goes out of your business. A firewall works pretty much the same way across the board. Make sure you select a very strong firewall to ensure network safety.

Antivirus protection: Antivirus and antimalware solutions are also extremely important for data security and must be installed on your systems. You must opt for the strongest antivirus protection software not just on your PCs and laptops but also on your mobile devices. They help you to fight unwanted threats to your files and data.

Secure Systems: Data loss can also be caused in case your laptop or mobile device is stolen. Thus, you must take some extra steps to ensure the further safety of these devices. The easiest way to secure your laptops is to encrypt them. Encryption software help to make the information look coded so that no one who is unauthorized can view or access your data without a password. Besides this, you must protect your laptop falling into the wrong hands. Make sure you never leave it in your car on unattended in the office, etc.

Secure Mobile Phones: Smartphones too hold a lot of important and confidential data such as messages, bank account details, and emails, etc. thus it is important to secure mobile phones as well. There are many ways to secure your mobile phones and some of them include, establishing strong passwords, to have encryption software, to have remote wiping enabled and to opt for phone finding apps so that you can locate your mobile phone if it is lost or stolen.

Monitor well: Another practice that you must follow in order to secure your data is to monitor it well and diligently. You must always keep track of your data, know which data is stored where and use good monitoring tools that can help prevent data leakage. The data leakage software that you choose must have set up of key network touchpoints that help to look for specific information coming out of internal network. Such software can be easily configured or customized to look for codes, credit card numbers o

Surf Safely: Your data safety is in your hands, and if you are careful, there will be no way anyone would be able to violate it. Thus, it is important to be careful how you surf the net and what precautions you follow. It is common for us sometimes to click on certain links or attachments thinking that they are harmless, but they could lead to data hacking or planting of malicious files. This may infect your system and may squeeze out information. Thus, it is important to surf safely, use internet security

Data Warehouse and Data Mining:

The term "Data Warehouse" was first coined by Bill Inmon in 1990. According to Inmon, a data warehouse is a subject oriented, integrated, time-variant, and non-volatile collection of data. This data helps analysts to take **informed decisions in an organization.**

An operational database undergoes frequent changes on a daily basis on account of the transactions that take place. Suppose a business executive wants to analyze previous feedback on any data such as a product, a supplier, or any consumer data, then the executive will have no data available to analyze because the previous data has been updated due to transactions.

A data warehouses provides us generalized and consolidated data in multidimensional view. Along with generalized and consolidated view of data, a data warehouses also provides us Online Analytical Processing (OLAP) tools. These tools help us in interactive and effective analysis of data in a multidimensional space. This analysis results in data generalization and data mining.

Data mining functions such as association, clustering, classification, prediction can be integrated with OLAP operations to enhance the interactive mining of knowledge at multiple level of abstraction. That's why data warehouse has now become an important platform for data analysis and online analytical processing.

Understanding a Data Warehouse

- A data warehouse is a database, which is kept separate from the organization's operational database.
- There is no frequent updating done in a data warehouse.
- It possesses consolidated historical data, which helps the organization to analyze its business.
- A data warehouse helps executives to organize, understand, and use their data to take strategic decisions.
- Data warehouse systems help in the integration of diversity of application systems.
- A data warehouse system helps in consolidated historical data analysis.

Why a Data Warehouse is separated from Operational Databases

A data warehouses is kept separate from operational databases due to the following reasons −

- An operational database is constructed for well-known tasks and workloads such as searching particular records, indexing, etc. In contract, data warehouse queries are often complex and they present a general form of data.
- Operational databases support concurrent processing of multiple transactions. Concurrency control and recovery mechanisms are required for operational databases to ensure robustness and consistency of the database.
- An operational database query allows to read and modify operations, while an OLAP query needs only **read only** access of stored data.
- An operational database maintains current data. On the other hand, a data warehouse maintains historical data.

Data Warehouse Features

The key features of a data warehouse are discussed below −

- **Subject Oriented** − A data warehouse is subject oriented because it provides information around a subject rather than the organization's ongoing operations. T**hese subjects can be product, customers, suppliers, sales, revenue, etc.** A data warehouse does not focus on the ongoing operations, rather it focuses on modeling and analysis of data for decision making.
- **Integrated** − A data warehouse is **constructed by integrating data from heterogeneous sources such as relational databases, flat files, etc.** This integration enhances the effective analysis of data.
- **Time Variant** − The data collected in a data warehouse is identified with a particular time period. The data in a data warehouse provides information from the historical point of view.
- **Non-volatile** − Non-volatile means the **previous data is not erased when new data is added to it.** A data warehouse is kept separate from the operational database and therefore frequent changes in operational database are not reflected in the data warehouse.

Note − A data warehouse does not require transaction processing, recovery, and concurrency controls, because it is physically stored and separate from the operational database.

Data Warehouse Applications

As discussed before, a data warehouse helps business executives to organize, analyze, and use their data for decision making. A data warehouse serves as a sole part of a plan execute assess "closed-loop" feedback system for the enterprise management. Data warehouses are widely used in the following fields −

1. **Financial services** 2. **Banking services**
3. **Consumer goods** 4. **Retail sectors** 5. **Controlled manufacturing**

Types of Data Warehouse

Information processing, analytical processing and data mining are the three types of data warehouse applications that are discussed below −

- **Information Processing** − A data warehouse allows to process the data stored in it. The data can be processed by means of querying, basic statistical analysis, reporting using crosstabs, tables, charts, or graphs.
- **Analytical Processing** − A data warehouse supports analytical processing of the information stored in it. The data can be analyzed by means of basic OLAP operations, including slice-and-dice, drill down, drill up, and pivoting.
- **Data Mining** − Data mining supports knowledge discovery by finding hidden patterns and associations, constructing analytical models, performing classification and prediction. These mining results can be presented using the visualization tools.

S.N.	Data Warehouse (OLAP)	Operational Database(OLTP)
1	It involves historical processing of information.	It involves day-to-day processing.
2	OLAP systems are used by knowledge workers such as executives, managers, and analysts.	OLTP systems are used by clerks, DBAs, or database professionals.
3	It is used to analyze the business.	It is used to run the business.
4	It focuses on Information out.	It focuses on Data in.
5	It is based on Star Schema, Snowflake Schema, and Fact Constellation Schema.	It is based on Entity Relationship Model.
6	It focuses on Information out.	It is application oriented.
7	It contains historical data.	It contains current data.
8	It provides summarized and consolidated data.	It provides primitive and highly detailed data.
9	It provides summarized and multidimensional view of data.	It provides detailed and flat relational view of data.
10	The number of users is in hundreds.	The number of users is in thousands.
11	The number of records accessed is in millions.	The number of records accessed is in tens.
12	The database size is from 100GB to 100 TB.	The database size is from 100 MB to 100 GB.
13	These are highly flexible.	It provides high performance.

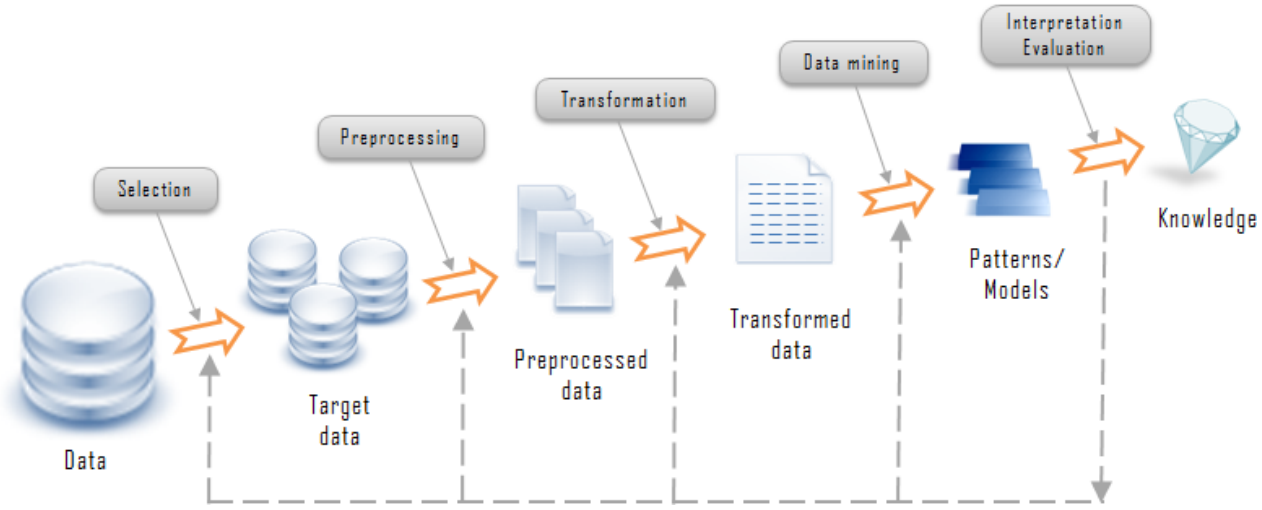


Fig: Data-mining Process

Database Administrator (DBA):

A Database Administrator, Database Analyst or Database Developer is the person responsible for managing the information within an organization. As most companies continue to experience inevitable growth of their databases, these positions are probably the most solid within the IT industry.

The DBA has many different responsibilities, but the overall goal of the DBA is to **keep the server up at all times and to provide users with access to the required information when they need it.** The DBA makes sure that the **database is protected and that any chance of data loss is minimized.**

A **DBA can be a programmer** who, by default or by volunteering, took over the responsibility of **maintaining a SQL Server** during project development and enjoyed the job so much that he switched.

A **DBA can be a system administrator** who was given the added responsibility of maintaining a SQL Server. DBAs can even come from unrelated fields, such as accounting or the help desk, and switch to Information Systems to become DBAs. To start your journey to becoming a Microsoft SQL Server DBA,

Responsibilities of DBA

Following are the responsibilities of Database Administrator (DBA),

- Installation, configuration and upgradation of databases like Microsoft SQL/ MySQL/ Oracle Server Software.
- Evaluating the features of various databases.
- Establishing and maintaining sound backup and recovery policies and procedures.
- Taking care of database design and implementation.
- Implementing and maintaining the database security.
- Database tuning, application tuning and performance monitoring.
- Maintaining documentation and standards.
- DBA does some technical trouble shooting and consultation to development teams.

Following skill set is required to be a successful Database Administrator,

1. Problem Management
2. Chain Management
3. Incident Management
4. Capacity Planning

Types of DBA

1. Administrative DBA

- Administrative DBA maintains the work on the server and keeps it running.
- Administrative DBA is mostly concerned with backups, security, replication etc.

2. Development DBA

- Development DBA builds queries, stored procedures etc. which mostly meet business needs.
- Development DBA is equivalent to a programmer.

3. Architect

- Architect builds table, design schema, foreign keys, primary keys etc. which meets the business needs.

4. Data Warehouse DBA

- Data Warehouse DBA is responsible for merging the data from multiple sources into a data warehouse.

5. OLAP DBA

- OLAP DBA builds multi-dimensional cubes for decision support or OLAP systems.
- The primary language in SQL Server is MDX.