**DBMS & RDBMS**

DBMS stands for Database Management System, and RDBMS is the acronym for the Relational Database Management system. **In DBMS, the data is stored as a file, whereas in RDBMS, data is stored in the form of tables**.

|  |  |  |
| --- | --- | --- |
| **No.** | **DBMS** | **RDBMS** |
| 1) | DBMS applications store **data as file**. | RDBMS applications store **data in a**  **tabular form**. |
| 2) | In DBMS, data is generally stored in either a hierarchical form or a navigational form. | In RDBMS, the tables have an identifier called primary  Keyword  data values are stored in the form of  tables. |
| 3) | **Normalization is not** present in DBMS. | **Normalization is** present in RDBMS. |
| 4) | DBMS does **not apply any security** with regards to data manipulation. | RDBMS **defines the integrity constraint** for the  purpose of ACID (Atomocity, Consistency, Isolation and  Durability) property. |
| 5) | DBMS uses file system to store data, so there will be **no relation between the tables**. | in RDBMS, data values are stored in the form of  tables, so a **relationship** between these data  values will be stored in the form of a table as well. |
| 6) | DBMS has to provide some uniform methods to access the stored information. | RDBMS system supports a tabular structure  of the data and a relationship between them to  access the stored information. |
| 7) | DBMS **does not support distributed database**. | RDBMS **supports distributed database**. |
| 8) | DBMS is meant to be for small organization and **deal with small data**. it supports **single user**. | RDBMS is designed to **handle large amount of**  **data**. it supports **multiple users**. |
| 9) | Examples of DBMS are file systems, **xml** etc. | Example of RDBMS are **mysql**, **postgre**, **sql server**,  **oracle** etc. |

After observing the differences between DBMS and RDBMS, you can say that RDBMS is an extension of DBMS. There are many software products in the market today who are compatible for both DBMS and RDBMS. Means today a RDBMS application is DBMS application and vice-versa.

DBMS (Data Base Management System)

Database management System is software which is used to store and retrieve the database. For example, Oracle, MySQL, etc.; these are some popular DBMS tools.

* DBMS provides the interface to perform the various operations like creation, deletion, modification, etc.
* DBMS allows the user to create their databases as per their requirement.
* DBMS accepts the request from the application and provides specific data through the operating system.
* DBMS contains the group of programs which acts according to the user instruction.
* It provides security to the database.

Advantage of DBMS

**Controls redundancy**

It stores all the data in a single database file, so it can control data redundancy.

**Data sharing**

An authorized user can share the data among multiple users.

**Backup**

It providesBackup and recovery subsystem. This recovery system creates automatic data from system failure and restores data if required.

**Multiple user interfaces**

It provides a different type of user interfaces like GUI, application interfaces.

Disadvantage of DBMS

**Size**

It occupies large disk space and large memory to run efficiently.

**Cost**

DBMS requires a high-speed data processor and larger memory to run DBMS software, so it is costly.

**Complexity**

DBMS creates additional complexity and requirements.

RDBMS (Relational Database Management System)

The word RDBMS is termed as 'Relational Database Management System.' It is represented as a table that contains rows and column.

RDBMS is based on the Relational model; it was introduced by **E. F. Codd**.

**A relational database contains the following components:**

* Table
* Record/ Tuple
* Field/Column name /Attribute
* Instance
* Schema
* Keys

An RDBMS is a tabular DBMS that maintains the security, integrity, accuracy, and consistency of the data.

# **Types of Databases**

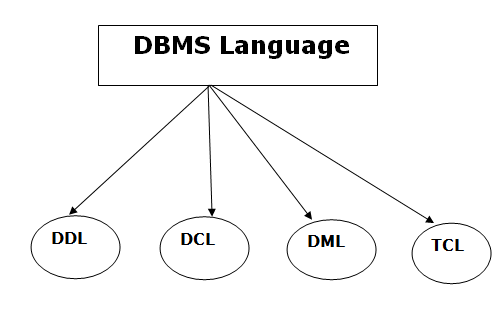
There are various types of databases used for storing different varieties of data:



# **Database Language**

* A DBMS has appropriate languages and interfaces to express database queries and updates.
* Database languages can be used to read, store and update the data in the database.

## Types of Database Language



### **1. Data Definition Language**

* **DDL** stands for **D**ata **D**efinition **L**anguage. It is used to define database structure or pattern.
* It is used to create schema, tables, indexes, constraints, etc. in the database.
* Using the DDL statements, you can create the skeleton of the database.
* Data definition language is used to store the information of metadata like the number of tables and schemas, their names, indexes, columns in each table, constraints, etc.

Here are some tasks that come under DDL:

* **Create:** It is used to create objects in the database.
* **Alter:** It is used to alter the structure of the database.
* **Drop:** It is used to delete objects from the database.
* **Truncate:** It is used to remove all records from a table.
* **Rename:** It is used to rename an object.
* **Comment:** It is used to comment on the data dictionary.

These commands are used to update the database schema that's why they come under Data definition language.

### **2. Data Manipulation Language**

**DML** stands for **D**ata **M**anipulation **L**anguage. It is used for accessing and manipulating data in a database. It handles user requests.

Here are some tasks that come under DML:

* **Select:** It is used to retrieve data from a database.
* **Insert:** It is used to insert data into a table.
* **Update:** It is used to update existing data within a table.
* **Delete:** It is used to delete all records from a table.
* **Merge:** It performs UPSERT operation, i.e., insert or update operations.
* **Call:** It is used to call a structured query language or a Java subprogram.
* **Explain Plan:** It has the parameter of explaining data.
* **Lock Table:** It controls concurrency.

### **3. Data Control Language**

* **DCL** stands for **D**ata **C**ontrol **L**anguage. It is used to retrieve the stored or saved data.
* The DCL execution is transactional. It also has rollback parameters.

(But in Oracle database, the execution of data control language does not have the feature of rolling back.)

Here are some tasks that come under DCL:

* **Grant:** It is used to give user access privileges to a database.
* **Revoke:** It is used to take back permissions from the user.

There are the following operations which have the authorization of Revoke:

CONNECT, INSERT, USAGE, EXECUTE, DELETE, UPDATE and SELECT.

### **4. Transaction Control Language**

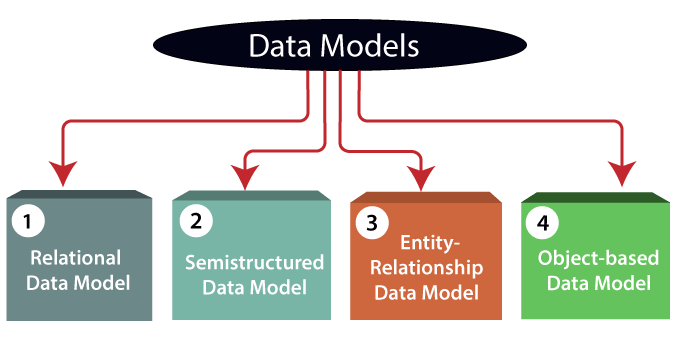
TCL is used to run the changes made by the DML statement. TCL can be grouped into a logical transaction.

Here are some tasks that come under TCL:

* **Commit:** It is used to save the transaction on the database.
* **Rollback:** It is used to restore the database to original since the last Commit.

# **Data Models**

Data Model is the modeling of the data description, data semantics, and consistency constra ints of the data. It provides the conceptual tools for describing the design of a database at each level of data abstraction. Therefore, there are following four data models used for understanding the structure of the database:



**1) Relational Data Model:** This type of model designs the data in the form of rows and columns within a table. Thus, a relational model uses tables for representing data and in-between relationships. Tables are also called relations. This model was initially described by Edgar F. Codd, in 1969. The relational data model is the widely used model which is primarily used by commercial data processing applications.

**2) Entity-Relationship Data Model:** An ER model is the logical representation of data as objects and relationships among them. These objects are known as entities, and relationship is an association among these entities. This model was designed by Peter Chen and published in 1976 papers. It was widely used in database designing. A set of attributes describe the entities. For example, student\_name, student\_id describes the 'student' entity. A set of the same type of entities is known as an 'Entity set', and the set of the same type of relationships is known as 'relationship set'.

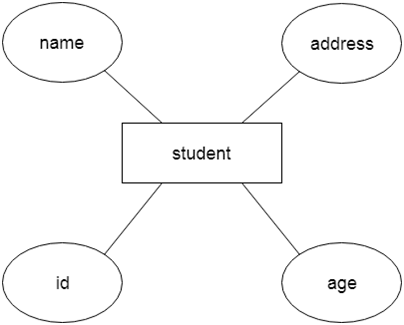
**3) Object-based Data Model:** An extension of the ER model with notions of functions, encapsulation, and object identity, as well. This model supports a rich type system that includes structured and collection types. Thus, in 1980s, various database systems following the object-oriented approach were developed. Here, the objects are nothing but the data carrying its properties.SQL CREATE TABLE

**4) Semistructured Data Model:** This type of data model is different from the other three data models (explained above). The semistructured data model allows the data specifications at places where the individual data items of the same type may have different attributes sets. The Extensible Markup Language, also known as XML, is widely used for representing the semistructured data. Although XML was initially designed for including the markup information to the text document, it gains importance because of its application in the exchange of data.

# **ER model**

* ER model stands for an Entity-Relationship model. It is a high-level data model. This model is used to define the data elements and relationship for a specified system.
* It develops a conceptual design for the database. 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.



## Component of ER Diagram



### **1. Entity:**

An entity may be any object, class, person or place. In the ER diagram, an entity can be represented as rectangles.

Consider an organization as an example- manager, product, employee, department etc. can be taken as an entity.



**a. Weak Entity**59.4M

1.1K

An entity that depends on another entity called a weak entity. The weak entity doesn't contain any key attribute of its own. The weak entity is represented by a double rectangle.



### **2. Attribute**

The attribute is used to describe the property of an entity. Eclipse is used to represent an attribute.

**For example,** id, age, contact number, name, etc. can be attributes of a student.



**a. Key Attribute**

The key attribute is used to represent the main characteristics of an entity. It represents a primary key. The key attribute is represented by an ellipse with the text underlined.



**b. Composite Attribute**

An attribute that composed of many other attributes is known as a composite attribute. The composite attribute is represented by an ellipse, and those ellipses are connected with an ellipse.



**c. Multivalued Attribute**

An attribute can have more than one value. These attributes are known as a multivalued attribute. The double oval is used to represent multivalued attribute.

**For example,** a student can have more than one phone number.



**d. Derived Attribute**

An attribute that can be derived from other attribute is known as a derived attribute. It can be represented by a dashed ellipse.

**For example,** A person's age changes over time and can be derived from another attribute like Date of birth.



### **3. Relationship**

A relationship is used to describe the relation between entities. Diamond or rhombus is used to represent the relationship.



Types of relationship are as follows:

**a. One-to-One Relationship**

When only one instance of an entity is associated with the relationship, then it is known as one to one relationship.

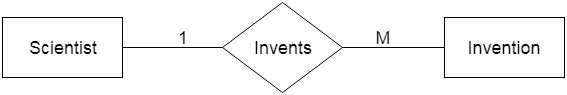
**For example,** A female can marry to one male, and a male can marry to one female.



**b. One-to-many relationship**

When only one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then this is known as a one-to-many relationship.

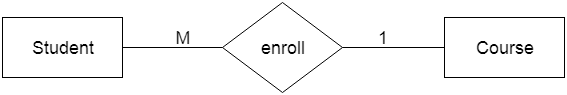
**For example,** Scientist can invent many inventions, but the invention is done by the only specific scientist.



**c. Many-to-one relationship**

When more than one instance of the entity on the left, and only one instance of an entity on the right associates with the relationship then it is known as a many-to-one relationship.

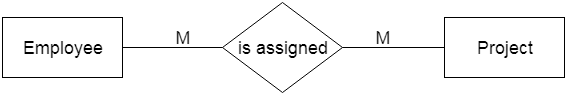
**For example,** Student enrolls for only one course, but a course can have many students.



**d. Many-to-many relationship**

When more than one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then it is known as a many-to-many relationship.

**For example,** Employee can assign by many projects and project can have many employees.



# **Normalization**

A large database defined as a single relation may result in data duplication. This repetition of data may result in:

* Making relations very large.
* It isn't easy to maintain and update data as it would involve searching many records in relation.
* Wastage and poor utilization of disk space and resources.
* The likelihood of errors and inconsistencies increases.

So to handle these problems, we should analyze and decompose the relations with redundant data into smaller, simpler, and well-structured relations that are satisfy desirable properties. Normalization is a process of decomposing the relations into relations with fewer attributes.

## What is Normalization?

* Normalization is the process of organizing the data in the database.
* Normalization is used to minimize the redundancy from a relation or set of relations. It is also used to eliminate undesirable characteristics like Insertion, Update, and Deletion Anomalies.
* Normalization divides the larger table into smaller and links them using relationships.
* The normal form is used to reduce redundancy from the database table.

Why do we need Normalization?

The main reason for normalizing the relations is removing these anomalies. Failure to eliminate anomalies leads to data redundancy and can cause data integrity and other problems as the database grows. Normalization consists of a series of guidelines that helps to guide you in creating a good database structure.

**Data modification anomalies can be categorized into three types:**

* **Insertion Anomaly:** Insertion Anomaly refers to when one cannot insert a new tuple into a relationship due to lack of data.
* **Deletion Anomaly:** The delete anomaly refers to the situation where the deletion of data results in the unintended loss of some other important data.
* **Updatation Anomaly:** The update anomaly is when an update of a single data value requires multiple rows of data to be updated.

## Types of Normal Forms:

Normalization works through a series of stages called Normal forms. The normal forms apply to individual relations. The relation is said to be in particular normal form if it satisfies constraints.

**Following are the various types of Normal forms:**



|  |  |
| --- | --- |
| **Normal Form** | **Description** |
| [1NF](https://www.javatpoint.com/dbms-first-normal-form) | A relation is in 1NF if it contains an atomic value. |
| [2NF](https://www.javatpoint.com/dbms-second-normal-form) | A relation will be in 2NF if it is in 1NF and all non-key attributes are fully functional  dependent on the primary key. |
| [3NF](https://www.javatpoint.com/dbms-third-normal-form) | A relation will be in 3NF if it is in 2NF and no transition dependency exists. |
| BCNF | A stronger definition of 3NF is known as Boyce Codd's normal form. |
| [4NF](https://www.javatpoint.com/dbms-forth-normal-form) | A relation will be in 4NF if it is in Boyce Codd's normal form and has no multi-valued  dependency. |
| [5NF](https://www.javatpoint.com/dbms-fifth-normal-form) | A relation is in 5NF. If it is in 4NF and does not contain any join dependency,  joining should be lossless. |

## Advantages of Normalization

* Normalization helps to minimize data redundancy.
* Greater overall database organization.
* Data consistency within the database.
* Much more flexible database design.
* Enforces the concept of relational integrity.

## Disadvantages of Normalization

* You cannot start building the database before knowing what the user needs.
* The performance degrades when normalizing the relations to higher normal forms, i.e., 4NF, 5NF.
* It is very time-consuming and difficult to normalize relations of a higher degree.
* Careless decomposition may lead to a bad database design, leading to serious problems.

# **Purpose of Normalization**

**Normalization** is the process of structuring and handling the relationship between data to minimize redundancy in the relational table and avoid the unnecessary anomalies properties from the database like insertion, update and delete. It helps to divide large database tables into smaller tables and make a relationship between them. It can remove the redundant data and ease to add, manipulate or delete table fields.

A normalization defines rules for the relational table as to whether it satisfies the normal form. A **normal form** is a process that evaluates each relation against defined criteria and removes the multivalued, joins, functional and trivial dependency from a relation. If any data is updated, deleted or inserted, it does not cause any problem for database tables and help to improve the relational table' integrity and efficiency.

## Objective of Normalization

1. It is used to remove the duplicate data and database anomalies from the relational table.
2. Normalization helps to reduce redundancy and complexity by examining new data types used in the table.
3. It is helpful to divide the large database table into smaller tables and link them using relationship.
4. It avoids duplicate data or no repeating groups into a table.
5. It reduces the chances for anomalies to occur in a database.

## Types of Anomalies

Following are the types of anomalies that make the table inconsistency, loss of integrity, and redundant data.

**1. Data redundancy** occurs in a relational database when two or more rows or columns have the same value or repetitive value leading to unnecessary utilization of the memory.

61M

1K

Hello Java Program for Beginners

**Student Table:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **StudRegistration** | **CourseID** | **StudName** | **Address** | **Course** |
| 205 | 6204 | James | Los Angeles | Economics |
| 205 | 6247 | James | Los Angeles | Economics |
| 224 | 6247 | Trent Bolt | New York | Mathematics |
| 230 | 6204 | Ritchie Rich | Egypt | Computer |
| 230 | 6208 | Ritchie Rich | Egypt | Accounts |

There are two students in the above table, 'James' and 'Ritchie Rich', whose records are repetitive when we enter a new CourseID. Hence it repeats the studRegistration, StudName and address attributes.

**2. Insert Anomaly:** An insert anomaly occurs in the relational database when some attributes or data items are to be inserted into the database without existence of other attributes. For example, In the Student table, if we want to insert a new courseID, we need to wait until the student enrolled in a course. In this way, it is difficult to insert new record in the table. Hence, it is called insertion anomalies.

**3. Update Anomalies:** The anomaly occurs when duplicate data is updated only in one place and not in all instances. Hence, it makes our data or table inconsistent state. For example, suppose there is a student 'James' who belongs to Student table. If we want to update the course in the Student, we need to update the same in the course table; otherwise, the data can be **inconsistent**. And it reflects the changes in a table with updated values where some of them will not.

**4. Delete Anomalies:** An anomaly occurs in a database table when some records are lost or deleted from the database table due to the deletion of other records. For example, if we want to remove Trent Bolt from the Student table, it also removes his address, course and other details from the Student table. Therefore, we can say that deleting some attributes can remove other attributes of the database table.

So, we need to avoid these types of anomalies from the tables and maintain the integrity, accuracy of the database table. Therefore, we use the normalization concept in the database management system.

## Types of Normalization

1. First Normal Form (1NF)
2. Second Normal Form (2NF)
3. Third Normal Form (3NF)
4. Boyce and Codd Normal Form (BCNF)
5. Fourth Normal Form (4NF)
6. Fifth Normal Form (5NF)

**First Normal Form (1NF):** The table will be in First Normal Form (1NF) if all the attributes of the table contain only atomic values. We can also say that if a table holds the multivalued data items in attributes or composite values, the relation cannot be in the first normal form. So, we need to make it first normal form by making the entries of the table atomic.

**Second Normal Form (2NF):** A Relation will be in 2NF if it follows the following condition:

1. The table or relation should be in 1NF or First Normal Form.
2. All the non-prime attributes should be fully functionally dependent on the candidate key.
3. The table should not contain any partial dependency.

**Third Normal Form (3NF):** The table will be in Third Normal Form (3NF) if it follows the given conditions:

1. The table or relation should be in 2NF.
2. It should not contain any transitive dependency. A **Transitive Dependency** is that any non-prime attribute determines or depends on the other non-prime attribute.  
   A relation is in 3NF if FD X determines Y ('X' -> 'Y') satisfies one of the following condition:
   1. If X -> Y is a trivial FD, i.e., Y is a subset of X.
   2. If X -> Y, where X is a Super key.
   3. If X -> Y, (Y - X) is a prime attribute.

#### **Note: A table should be in a 3 NF, if the Left-Hand Side (LHS) of all Functional dependency (FD) must be a Candidate Key (CK) / Super Key, Or the Right-hand side should be Prime attribute.**

**BCNF**: It stands for **Boyce Codd Normal form,** which is the next version of 3NF. Sometimes, it is also pronounced as **3.5 NF**. A normal form is said to be in BCNF if it follows the given conditions:

1. A table or relation must be in 3NF.
2. If a relation R has functional dependencies (FD) and if A determines B, where A is a super Key, the relation is in BCNF.

#### **Note: There should be one candidate/Super key on the left-hand side for each functional dependency for BCNF.**

**Fourth Normal Form (4 NF**): A relation is said to be Fourth Normal Form (4NF) if it follows the given conditions:

1. A table must be in BCNF.
2. There should be no multivalued dependency in the table.

For example, if the dependency A -> B, for a single value of A, more than one value of B exists. Then the relation is said to be a multivalued dependency.

Following is the condition for being multivalued dependency:

1. For a multivalued dependency A -> B, for a single value of A, there are multiple values of B.
2. If a table has at least three columns, they have a multivalued dependency.
3. For a relation with A, B and C columns, where B and C should be independent.

**Fifth Normal Form (5 NF):** A relation is said to be 5NF if it follows the given conditions:

1. The table should be in 4NF.
2. There should not be Join Dependency or further non-loss decomposed.

It is also known as **Project Join Normal Form** (PJNF).

**Join dependency**: A relation (R) is said to be a Join dependency if the relation (R) schema can be divided into smaller sets of tables R1, R2 … Rn that can be redesigned by joining multiple tables to the original table (R).

# **NoSQL Databases**

We know that MongoDB is a NoSQL Database, so it is very necessary to know about NoSQL Database to understand MongoDB throughly.

## What is NoSQL Database

Databases can be divided in 3 types:

1. RDBMS (Relational Database Management System)
2. OLAP (Online Analytical Processing)
3. NoSQL (recently developed database)

## NoSQL Database

NoSQL Database is used to refer a non-SQL or non relational database.

It provides a mechanism for storage and retrieval of data other than tabular relations model used in relational databases. NoSQL database doesn't use tables for storing data. It is generally used to store big data and real-time web applications.

## History behind the creation of NoSQL Databases

In the early 1970, Flat File Systems are used. Data were stored in flat files and the biggest problems with flat files are each company implement their own flat files and there are no standards. It is very difficult to store data in the files, retrieve data from files because there is no standard way to store data.

Then the relational database was created by E.F. Codd and these databases answered the question of having no standard way to store data. But later relational database also get a problem that it could not handle big data, due to this problem there was a need of database which can handle every types of problems then NoSQL database was developed.

## Advantages of NoSQL

* It supports query language.
* It provides fast performance.
* It provides horizontal scalability.

**Here are the 5 key features to look for in a NoSQL database:**

* Support for Multiple Data Models. ...
* Easily Scalable via Peer-to-Peer Architecture. ...
* Flexibility: Versatile Data Handling. ...
* Distribution Capabilities. ...
* Zero Downtime.

# Difference between Structured, Semi-structured and Unstructured data

**Big Data** includes huge volume, high velocity, and extensible variety of data. These are 3 types: Structured data, Semi-structured data, and Unstructured data. 

1. **Structured data –**   
   Structured data is data whose elements are addressable for effective analysis. It has been organized into a formatted repository that is typically a database. It concerns all data which can be stored in database SQL in a table with rows and columns. They have relational keys and can easily be mapped into pre-designed fields. Today, those data are most processed in the development and simplest way to manage information. *Example:* Relational data.
2. **Semi-Structured data –**   
   Semi-structured data is information that does not reside in a relational database but that has some organizational properties that make it easier to analyze. With some processes, you can store them in the relation database (it could be very hard for some kind of semi-structured data), but Semi-structured exist to ease space. *Example*: XML data.
3. **Unstructured data –**   
   Unstructured data is a data which is not organized in a predefined manner or does not have a predefined data model, thus it is not a good fit for a mainstream relational database. So for Unstructured data, there are alternative platforms for storing and managing, it is increasingly prevalent in IT systems and is used by organizations in a variety of business intelligence and analytics applications. *Example*: Word, PDF, Text, Media logs.

**Differences between Structured, Semi-structured and Unstructured data:**

| Properties | Structured data | Semi-structured data | Unstructured data |
| --- | --- | --- | --- |
| Technology | It is based on Relational database table | It is based on XML/RDF(Resource Description Framework). | It is based on character and binary data |
| Transaction management | Matured transaction and various concurrency techniques | Transaction is adapted from DBMS not matured | No transaction management and no concurrency |
| Version management | Versioning over tuples,row,tables | Versioning over tuples or graph is possible | Versioned as a whole |
| Flexibility | It is schema dependent and less flexible | It is more flexible than structured data but less flexible than unstructured data | It is more flexible and there is absence of schema |
| Scalability | It is very difficult to scale DB schema | It’s scaling is simpler than structured data | It is more scalable. |
| Robustness | Very robust | New technology, not very spread | — |
| Query performance | Structured query allow complex joining | Queries over anonymous nodes are possible | Only textual queries are possible |

# Difference between Relational database and NoSQL

**1. Relational Database :**   
RDBMS stands for Relational Database Management Systems. It is most popular database. In it, data is store in the form of row that is in the form of tuple. It contain numbers of table and data can be easily accessed because data is store in the table. This Model was proposed by E.F. Codd.

**2. NoSQL :**   
NoSQL Database stands for a non-SQL database. NoSQL database doesn’t use table to store the data like relational database. It is used for storing and fetching the data in database and generally used to store the large amount of data. It supports query language and provides better performance.

**Difference between Relational database and NoSQL :**

|  |  |
| --- | --- |
| **Relational Database** | **NoSQL** |
| It is used to handle data coming in low velocity. | It is used to handle data coming in high velocity. |
| It gives only read scalability. | It gives both read and write scalability. |
| It manages structured data. | It manages all type of data. |
| Data arrives from one or few locations. | Data arrives from many locations. |
| It supports complex transactions. | It supports simple transactions. |
| It has single point of failure. | No single point of failure. |
| It handles data in less volume. | It handles data in high volume. |
| Transactions written in one location. | Transactions written in many locations. |
| Deployed in vertical fashion. | Deployed in Horizontal fashion. |

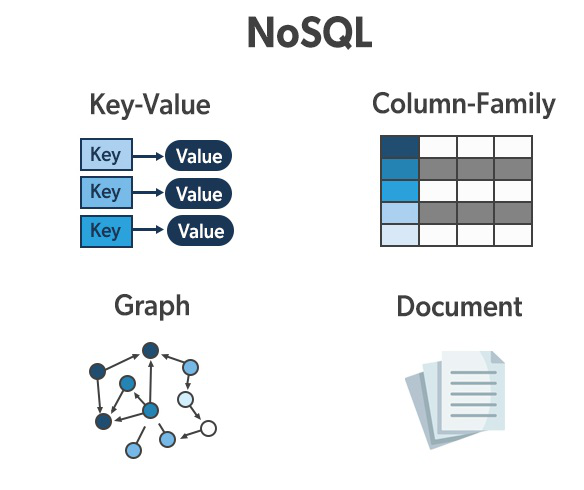
# Types of NoSQL Databases

A database is a collection of structured data or information which is stored in a computer system and can be accessed easily. A database is usually managed by a Database Management System (DBMS).

NoSQL is a non-relational database that is used to store the data in the nontabular form. NoSQL stands for Not only SQL. The main types are documents, key-value, wide-column, and graphs.

### Types of NoSQL Database:

* Document-based databases
* Key-value stores
* Column-oriented databases
* Graph-based databases



### Document-Based Database:

The document-based database is a nonrelational database. Instead of storing the data in rows and columns (tables), it uses the documents to store the data in the database. A document database stores data in JSON, BSON, or XML documents.

Documents can be stored and retrieved in a form that is much closer to the data objects used in applications which means less translation is required to use these data in the applications. In the Document database, the particular elements can be accessed by using the index value that is assigned for faster querying.

Collections are the group of documents that store documents that have similar contents. Not all the documents are in any collection as they require a similar schema because document databases have a flexible schema.

Key features of documents database:

* Flexible schema: Documents in the database has a flexible schema. It means the documents in the database need not be the same schema.
* Faster creation and maintenance: the creation of documents is easy and minimal maintenance is required once we create the document.
* No foreign keys: There is no dynamic relationship between two documents so documents can be independent of one another. So, there is no requirement for a foreign key in a document database.
* Open formats: To build a document we use XML, JSON, and others.

### Key-Value Stores:

A key-value store is a nonrelational database. The simplest form of a NoSQL database is a key-value store. Every data element in the database is stored in key-value pairs. The data can be retrieved by using a unique key allotted to each element in the database. The values can be simple data types like strings and numbers or complex objects.

A key-value store is like a relational database with only two columns which is the key and the value.

Key features of the key-value store:

* Simplicity.
* Scalability.
* Speed.

### Column Oriented Databases:

A column-oriented database is a non-relational database that stores the data in columns instead of rows. That means when we want to run analytics on a small number of columns, you can read those columns directly without consuming memory with the unwanted data.

Columnar databases are designed to read data more efficiently and retrieve the data with greater speed. A columnar database is used to store a large amount of data. Key features of columnar oriented database:

* Scalability.
* Compression.
* Very responsive.

### Graph-Based databases:

Graph-based databases focus on the relationship between the elements. It stores the data in the form of nodes in the database. The connections between the nodes are called links or relationships.

Key features of graph database:

* In a graph-based database, it is easy to identify the relationship between the data by using the links.
* The Query’s output is real-time results.
* The speed depends upon the number of relationships among the database elements.

## What is MongoDB

[MongoDB](https://www.javatpoint.com/mongodb-tutorial)

is an open-source document database that provides high performance, high availability, and automatic scaling.

In simple words, you can say that - Mongo DB is a document-oriented database. It is an open source product, developed and supported by a company named 10gen.

MongoDB is available under General Public license for free, and it is also available under Commercial license from the manufacturer.

The manufacturing company 10gen has defined MongoDB as:

"MongoDB is a scalable, open source, high performance, document-oriented database." - 10gen

MongoDB was designed to work with commodity servers. Now it is used by the company of all sizes, across all industry.

## History of MongoDB

The initial development of MongoDB began in 2007 when the company was building a platform as a service similar to window azure.

Window azure is a cloud computing platform and infrastructure, created by Microsoft, to build, deploy and manage applications and service through a global network.

MongoDB was developed by a NewYork based organization named 10gen which is now known as MongoDB Inc. It was initially developed as a PAAS (Platform as a Service). Later in 2009, it is introduced in the market as an open source database server that was maintained and supported by MongoDB Inc.

The first ready production of MongoDB has been considered from version 1.4 which was released in March 2010.

MongoDB2.4.9 was the latest and stable version which was released on January 10, 2014.

## Purpose of building MongoDB

It may be a very genuine question that - "what was the need of MongoDB although there were many databases in action?"

**There is a simple answer:**

All the modern applications require big data, fast features development, flexible deployment, and the older database systems not competent enough, so the MongoDB was needed.

**The primary purpose of building MongoDB is:**

* Scalability
* Performance
* High Availability
* Scaling from single server deployments to large, complex multi-site architectures.
* Key points of MongoDB
* Develop Faster
* Deploy Easier
* Scale Bigger

First of all, we should know what is document oriented database?

### **Example of document oriented database**

MongoDB is a document oriented database. It is a key feature of MongoDB. It offers a document oriented storage. It is very simple you can program it easily.

MongoDB stores data as documents, so it is known as document-oriented database.

1. FirstName = "John",
2. Address = "Detroit",
3. Spouse = [{**Name**: "Angela"}].
4. FirstName ="John",
5. Address = "Wick"

**There are two different documents (separated by ".").**

Storing data in this manner is called as document-oriented database.

Mongo DB falls into a class of databases that calls Document Oriented Databases. There is also a broad category of database known as [No SQL Databases](https://www.javatpoint.com/nosql-databases)

# **Features of MongoDB**

These are some important features of MongoDB:

**1. Support ad hoc queries**

In MongoDB, you can search by field, range query and it also supports regular expression searches.

**2. Indexing**

You can index any field in a document.

**3. Replication**

MongoDB supports Master Slave replication.

A master can perform Reads and Writes and a Slave copies data from the master and can only be used for reads or back up (not writes)

**4. Duplication of data**

MongoDB can run over multiple servers. The data is duplicated to keep the system up and also keep its running condition in case of hardware failure.

**5. Load balancing**

It has an automatic load balancing configuration because of data placed in shards.

**6. Supports map reduce and aggregation tools.**

**7. Uses JavaScript instead of Procedures.**

**8. It is a schema-less database written in C++.**

**9. Provides high performance.**

**10. Stores files of any size easily without complicating your stack.**

**11. Easy to administer in the case of failures.**

**12. It also supports:**

* JSON data model with dynamic schemas
* Auto-sharding for horizontal scalability
* Built in replication for high availability

**Mongodb command interface**

The MongoDB Command Line Interface ( mongocli ) **allows you to create and manage MongoDB Cloud deployments with MongoDB Atlas**. You can also use the MongoDB CLI to manage your on-premises MongoDB environments with MongoDB Cloud Manager and MongoDB Ops Manager.

MongoDB **Mongo shell** is the default client for the MongoDB database server. It's a command-line interface (CLI), where the input and output are all console-based. The Mongo shell is a good tool to manipulate small sets of data.

**What is a MongoDB compass?**

MongoDB Compass is **the official GUI for MongoDB, maintained by MongoDB itself**. MongoDB Compass helps users make clever decisions about the data structure, querying, indexing, and many more actions you can perform on the database.

MongoDB Compass is a powerful GUI for querying, aggregating, and analyzing your MongoDB data in a visual environment.

# **MongoDB Create Collection**

In MongoDB, db.createCollection(name, options) is used to create collection. But usually you don?t need to create collection. MongoDB creates collection automatically when you insert some documents. It will be explained later. First see how to create collection:

**Syntax:**

1. db.createCollection(**name**, options)

Here,

**Name:** is a string type, specifies the name of the collection to be created.

**Options:** is a document type, specifies the memory size and indexing of the collection. It is an optional parameter.

Following is the list of options that can be used.

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| Capped | Boolean | (Optional) If it is set to true, enables a capped collection. Capped collection is a fixed size  collecction that automatically overwrites its oldest entries when it reaches its maximum  size. If you specify true,you need to specify size parameter also. |
| AutoIndexID | Boolean | (Optional) If it is set to true, automatically create index on ID field. Its default value is false. |
| Size | Number | (Optional) It specifies a maximum size in bytes for a capped collection. Ifcapped is true, then  you need to specify this field also. |
| Max | Number | (Optional) It specifies the maximum number of documents allowed in the capped collection. |

Let's take an **example to create collection**. In this example, we are going to create a collection name SSSIT.

1. >use test

switched to db test

1. >db.createCollection("SSSIT")

{ "ok" : 1 }

To **check the created collection**, use the command "show collections".

1. >show collections

SSSIT

## How does MongoDB create collection automatically

MongoDB creates collections automatically when you insert some documents. For example: Insert a document named seomount into a collection named SSSIT. The operation will create the collection if the collection does not currently exist.

1. >db.SSSIT.**insert**({"name" : "seomount"})
2. >show collections
3. SSSIT

If you want to see the inserted document, use the find() command.

Syntax:

db.collection\_name.find()

MongoDB stores data records as BSON documents. BSON is a binary representation of [JSON](https://www.mongodb.com/docs/manual/reference/glossary/#std-term-JSON) documents, though it contains more data types than JSON. For the BSON spec, see [bsonspec.org](http://bsonspec.org/). See also [BSON Types](https://www.mongodb.com/docs/manual/reference/bson-types/).

## Document Structure

MongoDB documents are composed of field-and-value pairs and have the following structure:

|  |
| --- |
| { |
| field1: value1, |
| field2: value2, |
| field3: value3, |
| ... |
| fieldN: valueN |
| } |

The value of a field can be any of the BSON [data types](https://www.mongodb.com/docs/manual/reference/bson-types/), including other documents, arrays, and arrays of documents.

In MongoDB, the data records are stored as **BSON documents**. Here, BSON stands for binary representation of JSON documents, although BSON contains more data types as compared to JSON. The document is created using field-value pairs or key-value pairs and the value of the field can be of any BSON type.

# **JSON vs BSON**

Before knowing the differences between the JSON and BSON, we should know about JSON and BSON separately.

### **What is JSON?**

[JSON](https://www.javatpoint.com/json-tutorial)

is **Javascript Object Notation**. It is a standard format used for storing and interchanging the data. JSON is a user readable and it is a completely language format. It is light weight. When we read the JSON data of any application so that is language independent. We can send information through **JSON** very easily. JSON is basically a combination of key/value pair and arrays. In JSON, we can embed more than one json documents in a single json file. The maximum 100 documents can be embedded in a single json document.

The data types supported by JSON are string, number, object, array, true, false, and null.

**JSON Example:**

1. {
2. "\_id" : 100,
3. "name" : "John",
4. "subject" : [
5. "Maths",
6. "English",
7. "Hindi"
8. ],
9. "address" : {
10. "street" : "church road",
11. "city" : "faridabad",
12. "state" : "haryana"
13. }
14. }

In the above JSON example, "**id**" is defined. The name of the id starts with a '\_' underscore character. It is mandatory to provide the 'id' to the json document. If we do not provide the '**id**' to the json document, the system **id** is created. In the above, subject is an array that contains the values Maths, English, and Hindi. An address is a object as address contains the çurly bracket.

The '\_id' is a 12 bytes hexadecimal number that provides the uniqueness of every document. We can add the '\_id' field in a json document. If we do not provide the '\_id' field in a json document, then the system itself generates the '\_id'.

These 12 bytes are divided into four sections:

* First 4 bytes is the current timestamp.
* The next 3 bytes is the machine id.
* The next 2 bytes is the process id.
* The last 3 bytes is an incremental value.

### **What is BSON?**

|  |  |
| --- | --- |
| **JSON** | **BSON** |
| JSON is javascript object notation. | BSON is Binary Javascript Object notation. |
| It is a standard file format type. | It is a binary file format type. |
| JSON contains some basic data types like string, numbers, Boolean, null. | BSON contains some additional data types like date, timestamp, etc. |
| Databases like AnyDB, redis, etc. stores the data into the JSON format. | The data in MongoDB is stored in a BSON format. |
| JSON requires less space as compared to BSON. | BSON requires more space as compared to JSON. |
| It is comparatively less faster than BSON. | It is faster as compared to BSON. |
| It is used for the transmission of data. | It is used for the storage of data. |
| It does not have encoding and decoding technique. | It has faster encoding and decoding technique. |
| If we want to read any particular information from the JSON file then it needs to go through whole content. | In BSON, indexes concept is used that skips all the  content which are not in use. |
| JSON format does not require to be parsed as it is already human readable. | It requires to be parsed as it can be easily parsed by the  machines. |
| JSON is a combination of objects and arrays where an object is a collection of key-value pairs while array is an ordered list of elements. | The binary encoding technique provides some additional information such as length of the string and object subtypes. **BinData and Date** are the additional data types supported  by BSON over the JSON. |

BSON is a Binary **Javascript Object notation**. It is not in the human readable format as it is in the binary format. In MongoDB, BSON is used to encrypt the JSON data. It provides additional data types over the JSON data. It is also a language independent, and can be easily parsed.

It supports the adding of documents and arrays within other documents and arrays.

**BSON has following three characteristics:**

* Lightweight: When used over the network, the JSON keeps the overhead involved in processing extra header data to a minimum.
* Traversable: It is designed to traverse easily across the network. This helps in its role as the primary data representation for MongoDB.
* Efficient: It allows C data types that allow easy and quick encoding and decoding of data.

# **CRUD Operations in MongoDB**

MongoDB: Inserting a document into a collection(Create)

The command db.collection.insert() will perform an insert operation into a collection of a document.

Let us insert a document to a student collection. You must be connected to a database for doing any insert. It is done as follows:

db.student.insert({

regNo: "3014",

name: "Test Student",

course: {

courseName: "MCA",

duration: "3 Years"

},

address: {

city: "Bangalore",

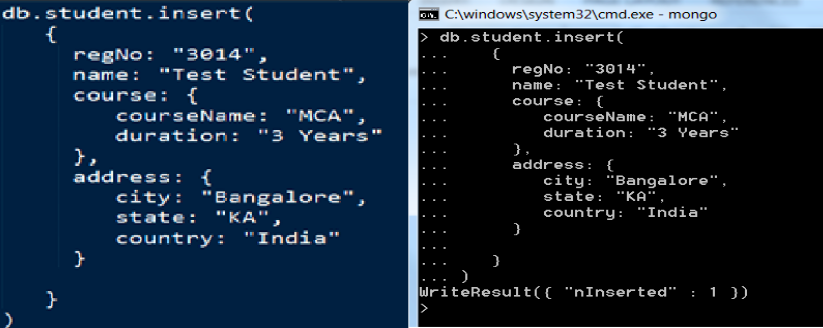
state: "KA",

country: "India"

}

})

Copy



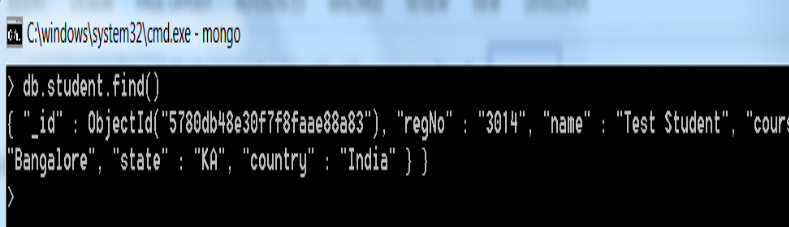
Note that an entry has been made into the collection called student.

MongoDB: Querying a document from a collection(Read)

To retrieve (Select) the inserted document, run the below command. The find() command will retrieve all the documents of the given collection.

db.collection\_name.find()

Copy



**NOTE:** Please observe that the record retrieved contains an attribute called \_id with some unique identifier value called **ObjectId** which acts as a document identifier.

If a record is to be retrieved based on some criteria, the find() method should be called passing parameters, then the record will be retrieved based on the attributes specified.

db.collection\_name.find({"fieldname":"value"})

Copy

For Example: Let us retrieve the record from the **student** collection where the attribute **regNo** is **3014** and the query for the same is as shown below:

db.students.find({"regNo":"3014"})

Copy

MongoDB: Updating a document in a collection(Update)

In order to update specific field values of a collection in MongoDB, run the below query.

db.collection\_name.update()

Copy

update() method specified above will take the fieldname and the new value as argument to update a document.

Let us update the attribute **name** of the collection **student** for the document with **regNo** 3014.

db.student.update({

"regNo": "3014"

},

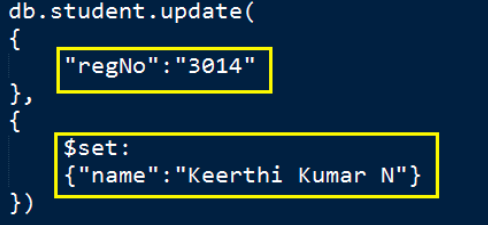
$set:

{

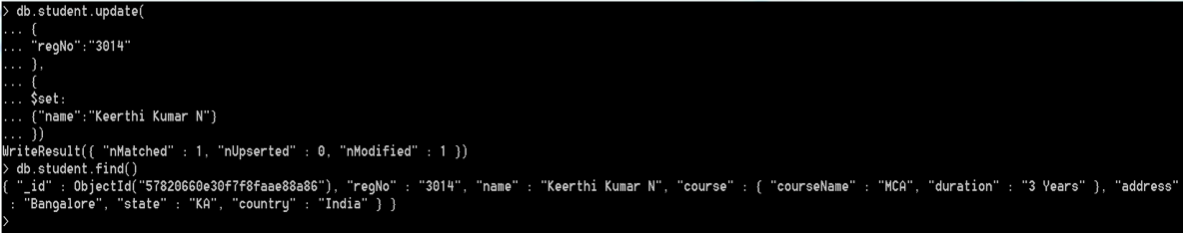
"name":"Viraj"

})

Copy



You will see the following in the Command Prompt:



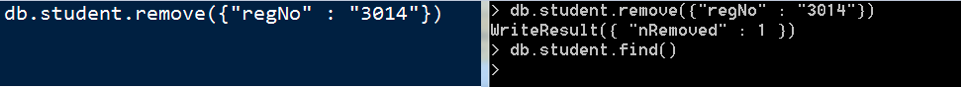
MongoDB: Removing an entry from the collection(Delete)

Let us now look into the deleting an entry from a collection. In order to delete an entry from a collection, run the command as shown below:

db.collection\_name.remove({"fieldname":"value"})

Copy

For Example: db.student.remove({"regNo":"3014"})



Note that after running the remove() method, the entry has been deleted from the student collection.

# **MongoDB sort() method**

In MongoDB, sort() method is used to sort the documents in the collection. This method accepts a document containing list of fields along with their sorting order.

The sorting order is specified as 1 or -1.

* 1 is used for ascending order sorting.
* -1 is used for descending order sorting.

**Syntax:**

1. db.COLLECTION\_NAME.find().sort({**KEY**:1})

## Scenario

Consider an example which has a collection name javatpoint

This collection has following fields within it.

1. [
2. {
3. Course: "Java",
4. details: { Duration: "6 months", Trainer: "Sonoo Jaiswal" },
5. Batch: [ { **size**: "Medium", qty: 25 } ],
6. category: "Programming Language"
7. },
8. {
9. Course: ".Net",
10. details: { Duration: "6 months", Trainer: "Prashant Verma" },
11. Batch: [ { **size**: "Small", qty: 5 }, { **size**: "Medium", qty: 10 }, ],
12. category: "Programming Language"
13. },
14. {
15. Course: "Web Designing",
16. details: { Duration: "3 months", Trainer: "Rashmi Desai" },
17. Batch: [ { **size**: "Small", qty: 5 }, { **size**: "Large", qty: 10 } ],
18. category: "Programming Language"
19. }
20. ];

Execute the following query to display the documents in descending order.

1. db.javatpoint.find().sort({"Course":-1})

This will show the documents in descending order.

{ "\_id" : ObjectId("564dbced8e2c097d15fbb603"), "Course" : "Web Designing", "det

ails" : { "Duration" : "3 months", "Trainer" : "Rashmi Desai" }, "Batch" : [ { "

size" : "Small", "qty" : 5 }, { "size" : "Large", "qty" : 10 } ], "category" : "

Programming Language" }

{ "\_id" : ObjectId("564dbced8e2c097d15fbb601"), "Course" : "Java", "details" : {

"Duration" : "6 months", "Trainer" : "Sonoo Jaiswal" }, "Batch" : [ { "size" :

"Medium", "qty" : 25 } ], "category" : "Programming Language" }

{ "\_id" : ObjectId("564dbced8e2c097d15fbb602"), "Course" : ".Net", "details" : {

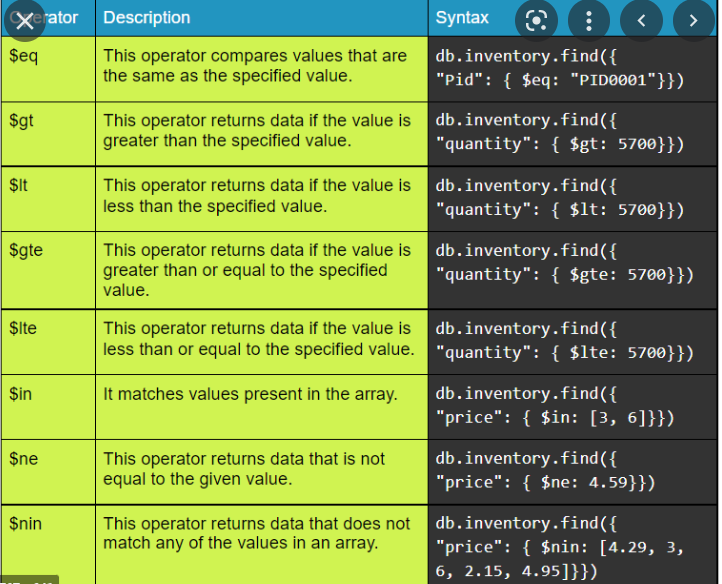
"Duration" : "6 months", "Trainer" : "Prashant Verma" }, "Batch" : [ { "size" :

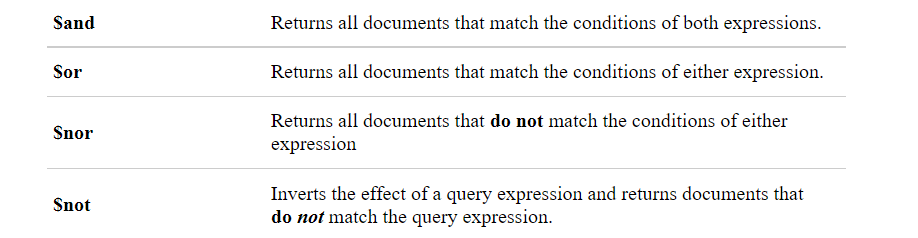
"Small", "qty" : 5 }, { "size" : "Medium", "qty" : 10 } ], "category" : "Progra

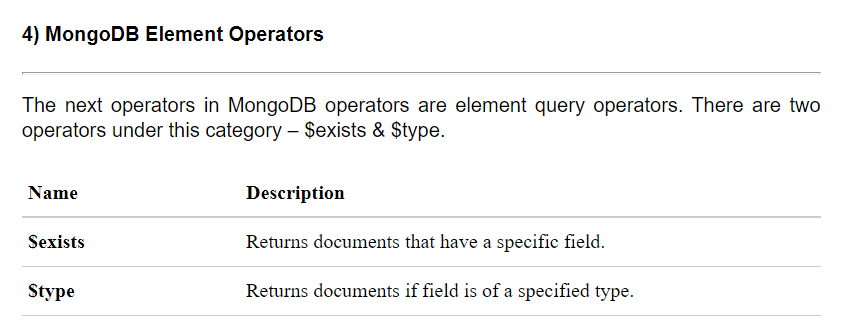
mming Language" }

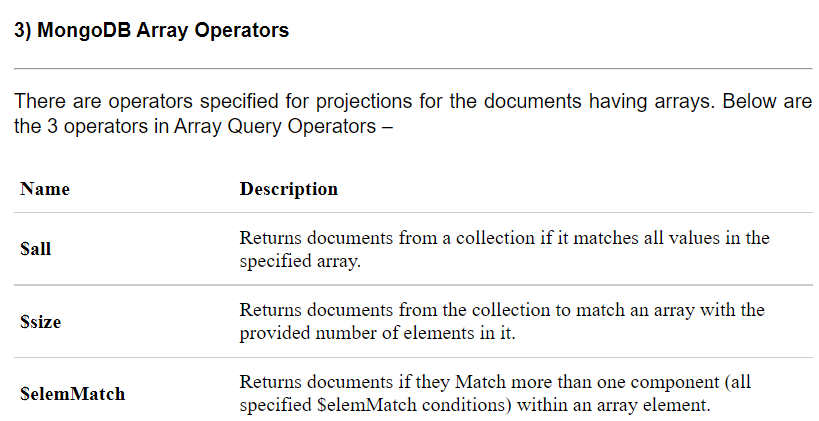
#### **Note: By default sort() method displays the documents in ascending order. If you don't specify the sorting preference, it will display documents in ascending order.**

**OPERATORS**

****

****

****

****

# Indexing in MongoDB

[MongoDB](https://www.geeksforgeeks.org/mongodb-an-introduction/) is leading NoSQL database written in C++. It is high scalable and provides high performance and availability. It works on the concept of collections and documents. Collection in MongoDB is group of related documents that are bound together. The collection does not follow any schema which is one of the remarkable feature of MongoDB.

**Indexing in MongoDB :**   
MongoDB uses indexing in order to make the query processing more efficient. If there is no indexing, then the MongoDB must scan every document in the collection and retrieve only those documents that match the query. Indexes are special data structures that stores some information related to the documents such that it becomes easy for MongoDB to find the right data file. The indexes are order by the value of the field specified in the index.

**Creating an Index :**   
MongoDB provides a method called createIndex() that allows user to create an index.

**Syntax –** 

db.COLLECTION\_NAME.createIndex({KEY:1})

The key determines the field on the basis of which you want to create an index and 1 (or -1) determines the order in which these indexes will be arranged(ascending or descending).

**Example –** 

db.mycol.createIndex({“age”:1})

{

“createdCollectionAutomatically” : false,

“numIndexesBefore” : 1,

“numIndexesAfter” : 2,

“ok” : 1

}

The createIndex() method also has a number of optional parameters.   
These include: 

* background (Boolean)
* unique (Boolean)
* name (string)
* sparse (Boolean)
* expireAfterSeconds (integer)
* hidden (Boolean)
* storageEngine (Document)

**Drop an index:**  
In order to drop an index, MongoDB provides the dropIndex() method.

**Syntax –** 

db.NAME\_OF\_COLLECTION.dropIndex({KEY:1})

The dropIndex() methods can only delete one index at a time. In order to delete (or drop) multiple indexes from the collection, MongoDB provides the dropIndexes() method that takes multiple indexes as its parameters.

**Syntax –** 

db.NAME\_OF\_COLLECTION.dropIndexes({KEY1:1, KEY2, 1})

The dropIndex() methods can only delete one index at a time. In order to delete (or drop) multiple indexes from the collection, MongoDB provides the dropIndexes() method that takes multiple indexes as its parameters.

**Get description of all indexes :**   
The getIndexes() method in MongoDB gives a description of all the indexes that exists in the given collection.

**Syntax –** 

db.NAME\_OF\_COLLECTION.getIndexes()

It will retrieve all the description of the indexes created within the collection.

# **What is Big Data**

Data which are very large in size is called Big Data. Normally we work on data of size MB(WordDoc ,Excel) or maximum GB(Movies, Codes) but data in Peta bytes i.e. 10^15 byte size is called Big Data. It is stated that almost 90% of today's data has been generated in the past 3 years.

## Sources of Big Data

These data come from many sources like

* **Social networking sites:** Facebook, Google, LinkedIn all these sites generates huge amount of data on a day to day basis as they have billions of users worldwide.
* **E-commerce site:** Sites like Amazon, Flipkart, Alibaba generates huge amount of logs from which users buying trends can be traced.
* **Weather Station:** All the weather station and satellite gives very huge data which are stored and manipulated to forecast weather.
* **Telecom company:** Telecom giants like Airtel, Vodafone study the user trends and accordingly publish their plans and for this they store the data of its million users.
* **Share Market:** Stock exchange across the world generates huge amount of data through its daily transaction.

# **Big Data Characteristics**

Big Data contains a large amount of data that is not being processed by traditional data storage or the processing unit. It is used by many **multinational companies** to **process** the data and business of many **organizations**. The data flow would exceed **150 exabytes** per day before replication.

There are five v's of Big Data that explains the characteristics.

### **5 V's of Big Data**

* **Volume**
* **Veracity**
* **Variety**
* **Value**
* **Velocity**

### **Volume**

The name Big Data itself is related to an enormous size. Big Data is a vast 'volumes' of data generated from many sources daily, such as **business processes, machines, social media platforms, networks, human interactions,** and many more.

**Facebook** can generate approximately a **billion** messages, **4.5 billion** times that the "**Like**" button is recorded, and more than **350 million** new posts are uploaded each day. Big data technologies can handle large amounts of data.

### **Variety**

Big Data can be **structured, unstructured, and semi-structured** that are being collected from different sources. Data will only be collected from **databases** and **sheets** in the past, But these days the data will comes in array forms, that are **PDFs, Emails, audios, SM posts, photos, videos,** etc.

**The data is categorized as below:**

1. **Structured data:** In Structured schema, along with all the required columns. It is in a tabular form. Structured Data is stored in the relational database management system.
2. **Semi-structured:** In Semi-structured, the schema is not appropriately defined, e.g., **JSON, XML, CSV, TSV**, and **email**. OLTP (**Online Transaction Processing**) systems are built to work with semi-structured data. It is stored in relations, i.e., **tables**.
3. **Unstructured Data**: All the **unstructured files, log files, audio files**, and **image** files are included in the unstructured data. Some organizations have much data available, but they did not know how to **derive** the value of data since the data is raw.
4. **Quasi-structured Data:**The data format contains textual data with inconsistent data formats that are formatted with effort and time with some tools.

**Example: Web server logs, i.e.,** the log file is created and maintained by some server that contains a list of **activities**.

### **Veracity**

Veracity means how much the data is reliable. It has many ways to filter or translate the data. Veracity is the process of being able to handle and manage data efficiently. Big Data is also essential in business development.

For example, **Facebook posts** with hashtags.

### **Value**

Value is an essential characteristic of big data. It is not the data that we process or store. It is **valuable** and **reliable** data that we **store, process**, and also **analyze**.

### **Velocity**

Velocity plays an important role compared to others. Velocity creates the speed by which the data is created in **real-time**. It contains the linking of incoming **data sets speeds, rate of change**, and **activity bursts**. The primary aspect of Big Data is to provide demanding data rapidly.

**Big data** velocity deals with the speed at the data flows from sources like **application logs, business processes, networks, and social media sites, sensors, mobile devices,** etc.