Distributed Database System

- Difficulty Level : Medium
- Last Updated : 17 Feb, 2022

A distributed database is basically a database that is not limited to one system, it is spread over different sites, i.e, on multiple computers or over a network of computers. A distributed database system is located on various sites that don't share physical components. This may be required when a particular database needs to be accessed by various users globally. It needs to be managed such that for the users it looks like one single database.

Types:
1. Homogeneous Database:
In a homogeneous database, all different sites store database identically. The operating system, database management system, and the data structures used – all are the same at all sites. Hence, they're easy to manage.
2. Heterogeneous Database:
In a heterogeneous distributed database, different sites can use different schema and software that can lead to problems in query processing and transactions. Also, a particular site might be completely unaware of the other sites. Different computers may use a different operating system, different database application. They may even use different data models for the database. Hence, translations are required for different sites to communicate.

Distributed Data Storage :
There are 2 ways in which data can be stored on different sites. These are:
1. Replication –
In this approach, the entire relationship is stored redundantly at 2 or more sites. If the entire database is available at all sites, it is a fully redundant database. Hence, in replication, systems maintain copies of data.
This is advantageous as it increases the availability of data at different sites. Also, now query requests can be processed in parallel.
However, it has certain disadvantages as well. Data needs to be constantly updated. Any change made at one site needs to be recorded at every site that relation is stored or else it may lead to inconsistency. This is a lot of overhead. Also, concurrency control becomes way more complex as concurrent access now needs to be checked over a number of sites.

2. Fragmentation –
In this approach, the relations are fragmented (i.e., they're divided into smaller parts) and each of the fragments is stored in different sites where they're required. It must be made sure that the fragments are such that they can be used to reconstruct the original relation (i.e, there isn't any loss of data).
Fragmentation is advantageous as it doesn't create copies of data, consistency is not a

problem.

Fragmentation of relations can be done in two ways:

- **Horizontal fragmentation – Splitting by rows –**
  The relation is fragmented into groups of tuples so that each tuple is assigned to at least one fragment.
- **Vertical fragmentation – Splitting by columns –**
  The schema of the relation is divided into smaller schemas. Each fragment must contain a common candidate key so as to ensure a lossless join.

In certain cases, an approach that is hybrid of fragmentation and replication is used.

Applications of Distributed Database:

- It is used in Corporate Management Information System.
- It is used in multimedia applications.
- Used in Military's control system, Hotel chains etc.
- It is also used in manufacturing control system.

References :

Distributed Database System

- Difficulty Level : Medium
- Last Updated : 17 Feb, 2022

A distributed database is basically a database that is not limited to one system, it is spread over different sites, i.e, on multiple computers or over a network of computers. A distributed database system is located on various sites that don't share physical components. This may be required when a particular database needs to be accessed by various users globally. It needs to be managed such that for the users it looks like one single database.

Types:
1. Homogeneous Database:
In a homogeneous database, all different sites store database identically. The operating system, database management system, and the data structures used – all are the same at all sites. Hence, they're easy to manage.
2. Heterogeneous Database:
In a heterogeneous distributed database, different sites can use different schema and software that can lead to problems in query processing and transactions. Also, a particular site might be completely unaware of the other sites. Different computers may use a different operating system, different database application. They may even use different data models for the database. Hence, translations are required for different sites to communicate.

Distributed Data Storage :
There are 2 ways in which data can be stored on different sites. These are:
1. Replication –
In this approach, the entire relationship is stored redundantly at 2 or more sites. If the entire database is available at all sites, it is a fully redundant database. Hence, in replication, systems maintain copies of data.
This is advantageous as it increases the availability of data at different sites. Also, now query requests can be processed in parallel.
However, it has certain disadvantages as well. Data needs to be constantly updated. Any change made at one site needs to be recorded at every site that relation is stored or else it may lead to inconsistency. This is a lot of overhead. Also, concurrency control becomes way more complex as concurrent access now needs to be checked over a number of sites.

2. Fragmentation –
In this approach, the relations are fragmented (i.e., they're divided into smaller parts) and each of the fragments is stored in different sites where they're required. It must be made sure that the fragments are such that they can be used to reconstruct the original relation (i.e, there isn't any loss of data).
Fragmentation is advantageous as it doesn't create copies of data, consistency is not a problem.

Fragmentation of relations can be done in two ways:

- **Horizontal fragmentation – Splitting by rows –**
  The relation is fragmented into groups of tuples so that each tuple is assigned to at least one fragment.
- **Vertical fragmentation – Splitting by columns –**
  The schema of the relation is divided into smaller schemas. Each fragment must contain a common candidate key so as to ensure a lossless join.

In certain cases, an approach that is hybrid of fragmentation and replication is used.

Applications of Distributed Database:
- It is used in Corporate Management Information System.
- It is used in multimedia applications.
- Used in Military's control system, Hotel chains etc.
- It is also used in manufacturing control system.

References :

Data Replication in DBMS

- Difficulty Level : Medium
- Last Updated : 20 Aug, 2019

Data Replication is the process of storing data in more than one site or node. It is useful in **improving the availability of data**. It is simply copying data from a database from one server to another server so that all the users can share the same data without any inconsistency. The result is a **distributed database** in which users can access data relevant to their tasks without interfering with the work of others.
Data replication encompasses duplication of transactions on an ongoing basis, so that the **replicate is in a consistently updated state** and synchronized with the source.However in data replication data is available at different locations, but a particular relation has to reside at only one location.
There can be full replication, in which the whole database is stored at every site. There can also be partial replication, in which some frequently used fragment of the database are replicated and others are not replicated.

Types of Data Replication –

1. **Transactional Replication –** In Transactional replication users receive full initial copies of the database and then receive updates as data changes. Data is copied in real time from the publisher to the receiving database(subscriber) in the same order as they occur with the publisher therefore in this type of replication, **transactional consistency is guaranteed.** Transactional replication is typically used in server-to-server environments. It does not simply copy the data changes, but rather consistently and accurately replicates each change.
2. **Snapshot Replication –** Snapshot replication distributes data exactly as it appears at a specific moment in time does not monitor for updates to the data. The entire snapshot is generated and sent to Users. **Snapshot replication is generally used when data changes are infrequent**. It is bit slower than transactional because on each attempt it moves multiple records from one end to the other end. Snapshot replication is a good way to perform initial synchronization between the publisher and the subscriber.
3. **Merge Replication –** Data from two or more databases is combined into a single database. Merge replication is the most complex type of replication because it allows both publisher and subscriber to independently make changes to the database. Merge replication is typically used in server-to-client environments. It allows changes to be sent from one publisher to multiple subscribers.

Replication Schemes –

1. Full Replication – The most extreme case is replication of the whole database at every site in the distributed system. This will improve the availability of the system because the system can continue to operate as long as atleast one site is up.

Advantages of full replication –

- High Availability of Data.
- Improves the performance for retrieval of global queries as the result can be obtained locally from any of the local site.
- Faster execution of Queries.

Disadvantages of full replication –

- Concurrency is difficult to achieve in full replication.
- Slow update process as a single update must be performed at different databases to keep the copies consistent.

2. No Replication – The other case of replication involves having No replication – that is, each fragment is stored at only one site.

Advantages of No replication –

- The data can be easily recovered.
- Concurrency can be achieved in no replication.

Disadvantages of No replication –

- Since multiple users are accessing the same server, it may slow down the execution of queries.
- The data is not easily available as there is no replication.

3. Partial Replication – In this type of replication some fragments of the database may be replicated whereas others may not. The number of copies of the fragment may range from one to the total number of sites in the distributed system. The description of replication of fragments is sometimes called the replication schema.

Advantages of Partial replication –

- The number of copies of the fragment depends upon the importance of data.

ADVANTAGES OF DATA REPLICATION – Data Replication is generally performed to:

- To provide a consistent copy of data across all the database nodes.
- To increase the availability of data.
- The reliability of data is increased through data replication.
- Data Replication supports multiple users and gives high performance.
- To remove any data redundancy,the databases are merged and slave databases are updated with outdated or incomplete data.
- Since replicas are created there are chances that the data is found itself where the transaction is executing which reduces the data movement.
- To perform faster execution of queries.

DISADVANTAGES OF DATA REPLICATION –

- More storage space is needed as storing the replicas of same data at different sites consumes more space.

- Data Replication becomes expensive when the replicas at all different sites need to be updated.
- Maintaining Data consistency at all different sites involves complex measures.

XML Database is used to store huge amount of information in the XML format. As the use of XML is increasing in every field, it is required to have a secured place to store the XML documents. The data stored in the database can be queried using **XQuery**, serialized, and exported into a desired format.

XML Database Types

There are two major types of XML databases −

- XML- enabled
- Native XML (NXD)

XML - Enabled Database

XML enabled database is nothing but the extension provided for the conversion of XML document. This is a relational database, where data is stored in tables consisting of rows and columns. The tables contain set of records, which in turn consist of fields.

Native XML Database

Native XML database is based on the container rather than table format. It can store large amount of XML document and data. Native XML database is queried by the **XPath**-expressions.

Native XML database has an advantage over the XML-enabled database. It is highly capable to store, query and maintain the XML document than XML-enabled database.

Example

Following example demonstrates XML database −

```
<?xml version = "1.0"?>
<contact-info>
   <contact1>
      <name>Tanmay Patil</name>
      <company>TutorialsPoint</company>
      <phone>(011) 123-4567</phone>
   </contact1>

   <contact2>
      <name>Manisha Patil</name>
      <company>TutorialsPoint</company>
      <phone>(011) 789-4567</phone>
   </contact2>
</contact-info>
```

Here, a table of contacts is created that holds the records of contacts (contact1 and contact2), which in turn consists of three entities − *name, company* and *phone*.

Previous Page

XML Schema is commonly known as **XML Schema Definition (XSD)**. It is used to describe and validate the structure and the content of XML data. XML schema defines the elements, attributes and data types. Schema element supports Namespaces. It is similar to a database schema that describes the data in a database.

Syntax

You need to declare a schema in your XML document as follows −

Example

The following example shows how to use schema −

```
<?xml version = "1.0" encoding = "UTF-8"?>
<xs:schema xmlns:xs = "http://www.w3.org/2001/XMLSchema">
   <xs:element name = "contact">
      <xs:complexType>
         <xs:sequence>
            <xs:element name = "name" type = "xs:string" />
            <xs:element name = "company" type = "xs:string" />
            <xs:element name = "phone" type = "xs:int" />
         </xs:sequence>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

The basic idea behind XML Schemas is that they describe the legitimate format that an XML document can take.

Elements

As we saw in the XML - Elements chapter, elements are the building blocks of XML document. An element can be defined within an XSD as follows −

```
<xs:element name = "x" type = "y"/>
```

Definition Types

You can define XML schema elements in the following ways −

Simple Type

Simple type element is used only in the context of the text. Some of the predefined simple types are: xs:integer, xs:boolean, xs:string, xs:date. For example −

```
<xs:element name = "phone_number" type = "xs:int" />
```

Complex Type

A complex type is a container for other element definitions. This allows you to specify which child elements an element can contain and to provide some structure within your XML documents. For example −

```
<xs:element name = "Address">
  <xs:complexType>
    <xs:sequence>
      <xs:element name = "name" type = "xs:string" />
      <xs:element name = "company" type = "xs:string" />
      <xs:element name = "phone" type = "xs:int" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

In the above example, *Address* element consists of child elements. This is a container for other **<xs:element>** definitions, that allows to build a simple hierarchy of elements in the XML document.

Global Types

With the global type, you can define a single type in your document, which can be used by all other references. For example, suppose you want to generalize the *person* and *company* for different addresses of the company. In such case, you can define a general type as follows −

```
<xs:element name = "AddressType">
  <xs:complexType>
    <xs:sequence>
      <xs:element name = "name" type = "xs:string" />
      <xs:element name = "company" type = "xs:string" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Now let us use this type in our example as follows −

```
<xs:element name = "Address1">
  <xs:complexType>
    <xs:sequence>
      <xs:element name = "address" type = "AddressType" />
      <xs:element name = "phone1" type = "xs:int" />
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name = "Address2">
  <xs:complexType>
    <xs:sequence>
      <xs:element name = "address" type = "AddressType" />
      <xs:element name = "phone2" type = "xs:int" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Instead of having to define the name and the company twice (once for *Address1* and once for *Address2*), we now have a single definition. This makes maintenance simpler, i.e., if you decide to add "Postcode" elements to the address, you need to add them at just one place.

Attributes

Attributes in XSD provide extra information within an element. Attributes have *name* and *type* property as shown below −

```
<xs:attribute name = "x" type = "y"/>
```

Previous Page

What is a NoSQL Database?

A non-relational database that stores data in non-tabular relations, a NoSQL database management system is a thing of the 21st century. Referring to non-SQL or non-relational databases, a NoSQL Database can store data in both traditional and non-traditional structural languages.

Perhaps this is why it is also referred to as 'not only SQL'. Before relational databases, large data used to be stored in database management systems (DBMS) that had a few drawbacks such as functional complications and slower recovery of data.

However, the emergence of SQL databases soon curbed these limitations and instead led to the development of relational database management system (RDBMS). This type of data storage led to the arrangement of data in the form of tables.

Yet, there was a need for a more flexible and simpler form of data storage. Therefore, NoSQL databases emerged that enhanced the way data was stored, making databases more flexible and simpler for large organizations.

Let us now understand the way a NoSQL database works. To begin with, one needs to note that NoSQL databases must be used when a large amount of data needs to be stored. Moreover, it does not require to be stored in a structured way that sets it free from any tabular form.

(Must read - Introduction to NoSQL)

Characteristics of NoSQL Database

Although there are different ways that can be incorporated to understand how NoSQL databases work, we will now look at some of the most common features that define a basic NoSQL database.

1. Complex-free working

Unlike SQL databases, NoSQL databases are not complicated. They store data in an unstructured or a semi-structured form that requires no relational or tabular arrangement. Perhaps they are easier to use and can be accomplished by all.

(Suggest blog: SQL: Applications and Commands)

2. Independent of Schema

Secondly, NoSQL databases are independent of schemas which implies that they can be run over without any predetermined schemas.

That said, they are far more efficient to work with and perhaps this particular feature works well for young programmers and organizations handling large amounts of heterogeneous data that requires no schemas to structure it.

(Must check: SQL vs NoSQL)

3. Better Scalability

One of the most prominent features of such a database is that it has high scalability that makes it suitable for large amounts of data.

Needless to mention that the contemporary data scientists often prefer to work with NoSQL databases due to this feature since it allows them to accommodate humongous data without rupturing its efficacy.

4. Flexible to accommodate

Since such databases can accommodate heterogeneous data that requires no structuring, they are claimed to be flexible in terms of their usage and reliability.

For beginners intending to try their hands in the field, NoSQL databases are easy to handle yet very useful.

(Read also: Top sites to learn SQL)

5. Durable

If durability is not one of its most striking features, then what is? NoSQL databases are highly durable as they can accommodate data ranging from heterogeneous to homogeneous.

Not only can they accommodate structured data, but they can also incorporate unstructured data that requires no query language. Undoubtedly, these databases are durable and efficient.

(Must check: Introduction to MySQL)

Types of NoSQL Databases

As we have gained some useful insights from the features of the NoSQL databases as to how they work, let us now jump on to the various NoSQL database types to understand the concept in a better manner.

To begin with, NoSQL databases can be divided into 4 types. They are as follows -

1. Document Database

   As the title itself indicates, the document database stores data in the form of documents. This implies that data is grouped into files that make it easier to be recognized when it is required for building application software.

   One of the major benefits of a document database is that it allows the developer to store data in a particular format of documents according to the same format they follow for their applications.

   It is a semi-structured and hierarchical NoSQL database that allows efficient storage of data. Especially when it comes to user profiles or catalogs, this type of NoSQL database works very well. A typical NoSQL database example is Mongodb.

   (Also read - Hadoop vs Mongodb)

2. Key-Value Database

Termed to be the simplest form of NoSQL database of all other types, the key-value database is a database that stores data in a schema-less manner. This type of database stores data in the key-value format.

Herein, a data point is categorized as a key to which a value (another data point) is allotted. For instance, a key data point can be termed as 'age' while the value data point can be termed as '45'.

This way, data gets stored in an organized manner with the help of associative pairing. A typical example of this type is Amazon's Dynamo database.

"Hundreds of thousands of AWS customers have chosen DynamoDB as their key-value and document database for mobile, web, gaming, ad tech, IoT, and other applications that need low-latency data access at any scale."- Amazon's Dynamo

3. Column-oriented Database

Another type of NoSQL database is the column-oriented database. This type of database stores data in the form of columns that segregates information into homogenous categories.

This allows the user to access only the desired data without having to retrieve unnecessary information.

When it comes to data analytics in social media networking sites, the column-oriented database works very efficiently by showcasing data that is prevalent in the search results.

Since such types of databases accommodate large amounts of data, it is better to filter out information. This is exactly what the column-oriented database does. A typical example of a column-oriented NoSQL database is Apache HBase.

4. Graph Database

The 4th type of NoSQL database is the graph database. Herein, data is stored in the form of graphical knowledge and related elements like edges, nodes, etc.

Data points are placed in such a manner that nodes are related to edges and thus, a network or connection is established between several data points.

This way, one data point leads to the other without the user having to retrieve individual data points. In the case of software development, this type of database works well since connected data points often lead to networked data storage.

This, in turn, makes the functioning of software highly effective and organized. An example of the graph NoSQL database is Amazon Neptune.

"Amazon Neptune is a fast, reliable, fully managed graph database service that makes it easy to build and run applications that work with highly connected datasets. "

Applications of NoSQL Databases

1. Data Mining

When it comes to data mining, NoSQL databases are useful in retrieving information for data mining uses. Particularly when it's about large amounts of data, NoSQL databases store data points in both structured and unstructured formats leading to efficient storage of big data.

Perhaps when a user wishes to mine a particular dataset from large amounts of data, one can make use of NoSQL databases, to begin with. Data is the building block of technology that has led mankind to such great heights.

Therefore, one of the most essential fields where NoSQL databases can be put to use is data mining and data storage.

(Also read - Top 10 Data Mining Tools)

2. Social Media Networking Sites

Social media is full of data, both structured and unstructured. A field that is loaded with tons of data to be discovered, social media is one of the most effective applications of NoSQL databases.

From comments to posts, user-related information to advertising, social media marketing requires NoSQL databases to be implemented in certain ways to retrieve useful information that can be helpful in certain ways.

Social media sites like Facebook and Instagram often approach open-source NoSQL databases to extract data that helps them keep track of their users and the activities going on around their platforms.

3. Software Development

The third application that we will be looking at is software development. Software development requires extensive research on users and the needs of the masses that are met through software development.

However, a developer must be able to scan through data that is available.

Perhaps NoSQL databases are always useful in helping software developers keep a tab on their users, their details, and other user-related data that is important to be noted. That said, NoSQL databases are surely helpful in software development.

Summing Up

To sum up, there are plenty of NoSQL databases out there to assist in data mining for one purpose or the other. However, a true NoSQL database is identified by its features like scalability, flexibility, and efficiency to accommodate data.

(Related blog: SQL

Yet, there was a need for a more flexible and simpler form of data storage. Therefore, NoSQL databases emerged that enhanced the way data was stored, making databases more flexible and simpler for large organizations.

Let us now understand the way a NoSQL database works. To begin with, one needs to note that NoSQL databases must be used when a large amount of data needs to be stored. Moreover, it does not require to be stored in a structured way that sets it free from any tabular form.

(Must read - Introduction to NoSQL)

Characteristics of NoSQL Database

Although there are different ways that can be incorporated to understand how NoSQL databases work, we will now look at some of the most common features that define a basic NoSQL database.

1. Complex-free working

Unlike SQL databases, NoSQL databases are not complicated. They store data in an unstructured or a semi-structured form that requires no relational or tabular arrangement. Perhaps they are easier to use and can be accomplished by all.

(Suggest blog: SQL: Applications and Commands)

2. Independent of Schema

Secondly, NoSQL databases are independent of schemas which implies that they can be run over without any predetermined schemas.

That said, they are far more efficient to work with and perhaps this particular feature works well for young programmers and organizations handling large amounts of heterogeneous data that requires no schemas to structure it.

(Must check: SQL vs NoSQL)

3. Better Scalability

One of the most prominent features of such a database is that it has high scalability that makes it suitable for large amounts of data.

Needless to mention that the contemporary data scientists often prefer to work with NoSQL databases due to this feature since it allows them to accommodate humongous data without rupturing its efficacy.

4. Flexible to accommodate

Since such databases can accommodate heterogeneous data that requires no structuring, they are claimed to be flexible in terms of their usage and reliability.

For beginners intending to try their hands in the field, NoSQL databases are easy to handle yet very useful.

(Read also: Top sites to learn SQL)

5. Durable

If durability is not one of its most striking features, then what is? NoSQL databases are highly durable as they can accommodate data ranging from heterogeneous to homogeneous.

Not only can they accommodate structured data, but they can also incorporate unstructured data that requires no query language. Undoubtedly, these databases are durable and efficient.

(Must check: Introduction to MySQL)

Types of NoSQL Databases

As we have gained some useful insights from the features of the NoSQL databases as to how they work, let us now jump on to the various NoSQL database types to understand the concept in a better manner.

To begin with, NoSQL databases can be divided into 4 types. They are as follows -

1. Document Database

As the title itself indicates, the document database stores data in the form of documents. This implies that data is grouped into files that make it easier to be recognized when it is required for building application software.

One of the major benefits of a document database is that it allows the developer to store data in a particular format of documents according to the same format they follow for their applications.

It is a semi-structured and hierarchical NoSQL database that allows efficient storage of data. Especially when it comes to user profiles or catalogs, this type of NoSQL database works very well. A typical NoSQL database example is Mongodb.

(Also read - Hadoop vs Mongodb)

2. Key-Value Database

Termed to be the simplest form of NoSQL database of all other types, the key-value database is a database that stores data in a schema-less manner. This type of database stores data in the key-value format.

Herein, a data point is categorized as a key to which a value (another data point) is allotted. For instance, a key data point can be termed as 'age' while the value data point can be termed as '45'.

This way, data gets stored in an organized manner with the help of associative pairing. A typical example of this type is Amazon's Dynamo database.

"Hundreds of thousands of AWS customers have chosen DynamoDB as their key-value and document database for mobile, web, gaming, ad tech, IoT, and other applications that need low-latency data access at any scale."- Amazon's Dynamo

3. Column-oriented Database

Another type of NoSQL database is the column-oriented database. This type of database stores data in the form of columns that segregates information into homogenous categories.

This allows the user to access only the desired data without having to retrieve unnecessary information.

When it comes to data analytics in social media networking sites, the column-oriented database works very efficiently by showcasing data that is prevalent in the search results.

Since such types of databases accommodate large amounts of data, it is better to filter out information. This is exactly what the column-oriented database does. A typical example of a column-oriented NoSQL database is Apache HBase.

4. Graph Database

The 4th type of NoSQL database is the graph database. Herein, data is stored in the form of graphical knowledge and related elements like edges, nodes, etc.

Data points are placed in such a manner that nodes are related to edges and thus, a network or connection is established between several data points.

This way, one data point leads to the other without the user having to retrieve individual data points. In the case of software development, this type of database works well since connected data points often lead to networked data storage.

This, in turn, makes the functioning of software highly effective and organized. An example of the graph NoSQL database is Amazon Neptune.

"Amazon Neptune is a fast, reliable, fully managed graph database service that makes it easy to build and run applications that work with highly connected datasets. "

Applications of NoSQL Databases

1. Data Mining

When it comes to data mining, NoSQL databases are useful in retrieving information for data mining uses. Particularly when it's about large amounts of data, NoSQL databases store data points in both structured and unstructured formats leading to efficient storage of big data.

Perhaps when a user wishes to mine a particular dataset from large amounts of data, one can make use of NoSQL databases, to begin with. Data is the building block of technology that has led mankind to such great heights.

Therefore, one of the most essential fields where NoSQL databases can be put to use is data mining and data storage.

(Also read - Top 10 Data Mining Tools)

2. Social Media Networking Sites

Social media is full of data, both structured and unstructured. A field that is loaded with tons of data to be discovered, social media is one of the most effective applications of NoSQL databases.

From comments to posts, user-related information to advertising, social media marketing requires NoSQL databases to be implemented in certain ways to retrieve useful information that can be helpful in certain ways.

Social media sites like Facebook and Instagram often approach open-source NoSQL databases to extract data that helps them keep track of their users and the activities going on around their platforms.

3. Software Development

The third application that we will be looking at is software development. Software development requires extensive research on users and the needs of the masses that are met through software development.

However, a developer must be able to scan through data that is available.

Perhaps NoSQL databases are always useful in helping software developers keep a tab on their users, their details, and other user-related data that is important to be noted. That said, NoSQL databases are surely helpful in software development.

Summing Up

To sum up, there are plenty of NoSQL databases out there to assist in data mining for one purpose or the other. However, a true NoSQL database is identified by its features like scalability, flexibility, and efficiency to accommodate data.

(Related blog: SQL