Why NoSQL?

- Let’s take an example to understand how a database in NoSQL is different from its SQL counterpart
- Consider an E-commerce company that needs to store its information in a database
- How would an RDBMS solution look like?

Why NoSQL?

- Let’s take an example to understand how a database in NoSQL is different from its SQL counterpart
- Consider an E-commerce company that needs to store its information in a database
- How would an RDBMS solution look like?

Orders:
Order_id,
Category,
…..

TABLE 1

Sales:
Order_id,
Sales
…..

TABLE 2

Profit:
Order_id,
Profit,
…..

TABLE 3

Why NoSQL?

- We perform joins between all these tables and get the final table
- Each record in this table would look somewhat as follows

Order_ID	Name	Category	Sales	Profit
TEC-103	Plantronics	Technology	2400	700

Why NoSQL?

- We perform joins between all these tables and get the final table
- Each record in this table would look somewhat as follows

Order_ID	Name	Category	Sales	Profit
TEC-103	Plantronics	Technology	2400	700

- But as the size of data increases, joins become computationally expensive.

Why NoSQL?

- This is an example of how records are stored in a NoSQL database. As all the information pertaining to a single record can be stored in one place, querying doesn’t necessarily require joining multiple tables

```
{

  "_id" : ObjectId("5eec2adaee4bfd8f653b24dc"),
  "Order_ID" : "TEC-103",
  "Name" : "Plantronics",
  "Sales" : 2400,
  "Profit" : 760

 }
```

Why NoSQL?

- The structure that you see is called the '**document**' structure or **JSON** structure.
- The elements are mapped in the form of a '**key : value**' pair.

```
{

  "_id" : ObjectId("5eec2adaee4bfd8f653b24dc"),
  "Order_ID" : "TEC-103",
  "Name" : "Plantronics",
  "Sales" : 2400,
  "Profit" : 760

 }
```

Why NoSQL?

- The document structure is quite flexible and doesn't restrict itself to a well-defined schema like in a SQL table.

- For example in the following record

Order_ID	Name	Category	Sales	Profit
TEC-103	Plantronics	Technology	2400	0

you can't remove the Profit field from this record even if its value is 0,

- The schema is quite rigid, and you need to show all the fields for all the records.

Why NoSQL?

- But in the document structure, you have the flexibility to show as many fields as you want for each record.

```
{

  "_id" : ObjectId("5eec2adaee4bfd8f653b24dc"),
  "Order_ID" : "TEC-103",
  "Name" : "Plantronics",
  "Sales" : 2400,

}
```

- NoSQL databases have a dynamic schema and therefore you can add or remove attributes for different records

SQL vs NoSQL

- SQL databases store information in form of tables with a fixed schema, whereas NoSQL databases can follow a dynamic schema.

SQL vs NoSQL

- SQL databases store information in form of tables with a fixed schema, whereas NoSQL databases can follow a dynamic schema.

- For scaling SQL databases, you generally do it ***vertically*** - by increasing the CPU, RAM to increase the power of the existing database. NoSQL databases are more suitable for ***horizontal*** scaling, where you keep on adding more servers to improve the performance.

SQL vs NoSQL

- SQL databases store information in form of tables with a fixed schema, whereas NoSQL databases can follow a dynamic schema.

- For scaling SQL databases, you generally do it ***vertically*** - by increasing the CPU, RAM to increase the power of the existing database. NoSQL databases are more suitable for ***horizontal*** scaling, where you keep on adding more servers to improve the performance.

- SQL databases are designed to handle advanced querying requirements, whereas NoSQL databases are utilised to handle complex databases and scale them as per requirement.

Introduction to MongoDB

- A brief history of MongoDB
  - Open source, document-based NoSQL database
  - Created in 2007, with the idea of building databases that can support ***humongous*** amounts of data to ensure scalability.
  - Provides high performance, scalability and data modelling features.

Structure of MongoDB databases

Structure of MongoDB databases

- Earlier in a SQL database, the data was stored in tables.

Structure of MongoDB databases

- Earlier in a SQL database, the data was stored in tables.
- Data is stored in the form of **collections** here. Each collection stores information on several records.

Structure of MongoDB databases

- Earlier in a SQL database, the data was stored in tables.
- Data is stored in the form of **collections** here. Each collection stores information on several records.
- Each record in the collection is stored as a **document**. The information is recorded in the form a **key : value** pair.

Getting started with MongoDB

- To select an existing database or create a new database we use the following command

```
use testdb
```

where `use` is the command that either creates a new database called `testdb` or we select it if it already exists

Getting started with MongoDB

- To select an existing database or create a new database we use the following command

```
use testdb
```

where `use` is the command that either creates a new database called `testdb` or we select it if it already exists

- To create a collection to start storing the data we can use the following command

```
db.createCollection('Orders')
```

where `'Orders'` is the name of the new collection that we want to create.

Getting started with MongoDB

- To add a document to the collection, we need to use the following command

```
 db.Orders.insert(
{
        "Order_ID" : "TEC-103",
        "Name" : "Plantronics",
        "Sales" : 2400,
        "Profit" : 760
})
```

The `insert` command helps in adding documents

- To view the document that we have added we can use the `db.Orders.find()` command

Getting started with MongoDB

- When we use the `db.Orders.find()` command we get an output like this

```
{
      "_id" : ObjectId("5ef58d5d1564fba7f8bea373"),
      "Order_ID" : "TEC-103",
      "Name" : "Plantronics",
      "Sales" : 2400,
      "Profit" : 760
}
```

- Each of the items in the document is in the form of a **key : value** pair.

Introduction to CRUD

Essential steps in a database management system

- **C –** Create a new record in the database
- **R –** Read a record and understand its contents
- **U –** Update some values in the record
- **D –** Delete a record from the database

Introduction to CRUD

- Create a new document using the `db.collection.insert()`
  - Create or select a database, choose the collection and then insert the document by passing the key – value pairs.

Introduction to CRUD

- Create a new document using the `db.collection.insert()`
  - Create or select a database, choose the collection and then insert the document by passing the key – value pairs.

- Read the documents using `db.collection.find()`
  - When you use this command, all the documents in the collection will be returned.

Introduction to CRUD

- Create a new document using the `db.collection.insert()`
  - Create or select a database, choose the collection and then insert the document by passing the key – value pairs.

- Read the documents using `db.collection.find()`
  - When you use this command, all the documents in the collection will be returned.
  - Can be used to filter documents based on a specific field and value pair.

Introduction to CRUD

- Create a new document using the `db.collection.insert()`
  - Create or select a database, choose the collection and then insert the document by passing the key – value pairs.

- Read the documents using `db.collection.find()`
  - When you use this command, all the documents in the collection will be returned.
  - Can be used to filter documents based on a specific field and value pair.
  - Also, additional conditional statements and logical operators can also be added to further filter the documents.

Introduction to CRUD

- Create a new document using the `db.collection.insert()`
  - Create or select a database, choose the collection and then insert the document by passing the key – value pairs.

- Read the documents using `db.collection.find()`
  - When you use this command, all the documents in the collection will be returned.
  - Can be used to filter documents based on a specific field and value pair.
  - Also, additional conditional statements and logical operators can also be added to further filter the documents.

- Update the values using `db.collection.update()`

Introduction to CRUD

- Create a new document using the `db.collection.insert()`
  - Create or select a database, choose the collection and then insert the document by passing the key – value pairs.

- Read the documents using `db.collection.find()`
  - When you use this command, all the documents in the collection will be returned.
  - Can be used to filter documents based on a specific field and value pair.
  - Also, additional conditional statements and logical operators can also be added to further filter the documents.

- Update the values using `db.collection.update()`

- Delete a document using `db.collection.remove()`

THE BASE PROPERTY

- The abbreviation **BASE** is used to describe the properties of NoSQL databases.
- **BASE** consist of three properties

1. **B**asic **A**vailability
2. **S**oft State
3. **E**ventual Consistency

THE BASE PROPERTY

1

Basic Availability

1. System is mostly available, but without any kind of consistency guarantee.

2

Soft State

1. Even without input query, the state of the system may change over time.
2. This happens because a NoSQL database keeps on trying to make it consistent and available by synchronizing with other systems

3

Eventual Consistency

1. The system will eventually become consistent once it stops receiving input.
2. So, if we wait long enough for any give input, we will get consistent reads.