# NoSQL

- “NoSQL” stands for “Not Only SQL,” emphasizing that these databases can handle a variety of data models, not just relational.
- Unlike relational databases, which store data in tables with rows and columns, NoSQL databases use more flexible storage methods.

## How NoSQL Differs from SQL

| Feature | SQL | NoSQL |
|---|---|---|
| **Schema** | Fixed schema (tables) | Flexible schema (documents/keys) |
| **Scalability** | Vertical (more power to one server) | Horizontal (add servers) |
| **Joins** | Complex joins supported | Not typically supported |
| **Data Storage** | Tables (rows & columns) | Documents, key-value pairs, etc. |
| **Query Language** | SQL (Structured Query Language) | Database-specific APIs or queries |
| **Best For** | Structured, relational data | Large, unstructured/semi-structured data |

- Here we use Mongo DB for NoSQL
- An open-source document database and leading NoSQL database.
- Used In:
    - Big Data
    - Content Management and Delivery
    - Mobile and Social Infrastructure
    - User Data Management
    - Data Hub
- Installation
    - https://www.mongodb.com/download-center/community
    - C:\Program Files\MongoDB\Server\5.0\bin
    - mongo.exe: client

| RDBMS | MongoDB |
|---|---|
| Database | Database |
| Table | Collection |
| Tuple/Row | Document |
| column | Field |

## Commands in Mongo DB

- **To see default Databases**

```
show dbs
```

- **To create a Database or enter a Database**

```
use database_name
```

- **To check currently selected database**

```
db
```

- **To drop a database**

```
use database_name (Enter into a database)
db.dropDatabase()
```

- **Create Collection(table)**

```
db.createCollection('collectionname')
```

- **To show all collections**

```
show collections
```

- **To drop collection**

```
db.collectionname.drop()
```

- **Insert a document into the collection (adding rows to the table)**

```
db.collectionname.insert({propertyname:value}) (set property name as column_name)
```

- **To display documents in a collection**

```
db.collectionname.find()
Or
db.collection_name.find().pretty() (used to show document in an organized manner)
```

- **To get first two documents**

```
db.collection_name.find().limit(2)
```

- **To skip the first one and display the rest two**

```
db.collection_name.find().limit(2).skip(1)
```

- **To get the documents in ascending order**

```
db.collection_name.find().sort({"name": 1}) (for descending order, change to -1)
```

- **To update the document**

```
db.collection_name.update({based on},{operator:{which all / what all}})
```

- **To rename a property**

```
db.sharon.update({name:"sharon"},{$rename:{"contact": "phone"}})
```

To Delete a document based on condition

```
db.collection_name.remove({property:value})
```

- **To Delete a collection**

```
db.collection_name.remove()
```

- **To rename a collection**

```
db.collection_name.renameCollection(new_collection_name)
```

## Operators in MongoDB

- **Comparison Operators**

| Symbol | MongoDB |
|---|---|
| **<** | $lt |
| **>** | $gt |
| **<=** | $lte |
| **>=** | $gte |
| **==** | $eq |
| **!=** | $ne |

**Example:**

```
db.collection_name.find({"age":{$eq:4}})
```

- **Membership Operators**

| Symbol | MongoDB |
|---|---|
| **in** | $in |
| **not in** | $nin |

**Example:**

```
db.collection_name.find({"age":$in:[4,45,43]})
```

- **Logical Operators**

| Symbol | MongoDB |
|---|---|
| **and** | $and |
| **or** | $or |
| **not** | $not |

**Example:**

```
db.collection_name.find({$or:[{"name":"appu"},{"place":"ekm"}]})
db.collection_name.find({$and:[{"name":"appu"},{"place":"ekm"}]})
```