No Schema

"No Schema" refers to the flexibility it provides in terms of document structure within a collection. Unlike traditional relational databases where you define a fixed schema with tables, columns, and relationships, MongoDB allows you to store documents in collections without enforcing a rigid schema.

This means that documents within a collection can have varying structures, and fields can be added, modified, or removed dynamically without the need to define a schema beforehand. Each document can have its own unique structure, with different fields and data types.

How MongoDB Works

MongoDB is a document-oriented database, classified as a NoSQL database. It stores data in flexible, JSON-like documents, meaning fields can vary from document to document and data structure can be changed over time. MongoDB doesn't require a schema to be defined before adding data; however, you can enforce a schema if needed.

MongoDB works on the principle of collections and documents. A collection is a grouping of MongoDB documents and is the equivalent of a table in a relational database. Documents are individual records within a collection and are similar to rows in a table.

Insert First Data

To insert data into MongoDB, you can use the insertOne() method. Here's a basic example:

```
db.collection("myCollection").insertOne({ "name": "John", "age": 30 });
```

This command inserts a document with the fields "name" and "age" into the "myCollection" collection.

CRUD Operations

CRUD stands for Create, Read, Update, Delete. MongoDB provides methods to perform these operations:

- Create: insertOne() or insertMany() to add new documents.
- Read: find() to retrieve documents.
- Update: updateOne() or updateMany() to modify existing documents.
- Delete: deleteOne() or deleteMany() to remove documents.

Insert Many

To insert multiple documents at once, you can use the insertMany() method. Here's an example:

```
db.collection("myCollection").insertMany([
  { "name": "Alice", "age": 25 },
  { "name": "Bob", "age": 35 },
  { "name": "Charlie", "age": 40 }
]);
```

This inserts three documents into the "myCollection" collection.

Update and Update Many

Update: The updateOne() method in MongoDB updates the first document that matches the specified criteria.

To update documents, you can use the updateOne() or updateMany() method. Here's an example:

Ex1:

```
db.collection("myCollection").updateOne(
   { "name": "Alice" }, // Filter
  { $set: { "age": 26 }} // Update
);
```

This updates the age of the document where the name is "Alice" to 26.

Update Many: The updateMany() method updates all documents that match the specified criteria

Ex2:

```
db.employees.updateMany(
 { "jobTitle": "Software Engineer" }, // Filter criteria
 { $set: { "department": "Engineering" }} // Update operation
);
```

All documents where the job title is "Software Engineer" will have their department updated to "Engineering".

Delete and Delete Many

To delete documents, you can use the deleteOne() or deleteMany() method. Here's an example:

Ex1:

```
db.collection("myCollection").deleteOne({ "name": "Alice" });
```

This deletes the document where the name is "Alice" from the collection.

Ex2:

```
db.employees.deleteMany(
  { "age": { $gt: 50 } } // Filter criteria
);
```

this operation, all documents where the age is greater than 50 will be deleted from the collection.