DOCUMENT BASED MONGODB DATA MODEL AND CRUD OPERATIONS

Document-based data modeling in MongoDB revolves around storing and managing data as **documents** in collections. MongoDB is a NoSQL database, and its document-based approach provides flexibility, scalability, and ease of use for a wide range of applications.

Here's a breakdown of MongoDB's document-based data model and how CRUD (Create, Read, Update, Delete) operations are performed:

---

1. MongoDB Data Model: Document-Based

- **Documents**: Data is stored in **JSON-like objects** called documents, which are key-value pairs.
  - Example document:

```
json
Copy code
{
  "_id": "12345",
  "name": "John Doe",
  "email": "johndoe@example.com",
  "age": 30,
  "address": {
    "street": "123 Main St",
    "city": "Anytown",
    "state": "CA",
    "zip": "12345"
  },
  "orders": [
    { "order_id": "A001", "product": "Laptop", "amount": 1200 },
    { "order_id": "A002", "product": "Mouse", "amount": 25 }
  ]
}
```

- **Collections**: Documents are grouped into collections. Each collection can hold documents with similar structures but no strict schema.

Key Features:

1. **Schema Flexibility**: Collections do not enforce a strict schema, allowing documents with different fields to coexist.
2. **Embedded Documents**: Related data can be nested within a document, reducing the need for joins.
3. **Scalability**: MongoDB supports horizontal scaling through sharding.

---

2. CRUD Operations in MongoDB

A. Create

- Used to insert documents into a collection.
- **Command**: `insertOne()` or `insertMany()`
- Example:

```
javascript
Copy code
db.users.insertOne({
  name: "Alice Smith",
  email: "alice@example.com",
  age: 25,
  preferences: { newsletter: true, theme: "dark" }
});
```

B. Read

- Used to retrieve documents from a collection.
- **Command**: `find()` or `findOne()`
- Example:

```
javascript
Copy code
// Retrieve all documents
db.users.find();

// Retrieve documents with a filter
db.users.find({ age: { $gte: 18 } });

// Retrieve a specific document
db.users.findOne({ email: "alice@example.com" });
```

C. Update

- Used to modify existing documents.
- **Command**: `updateOne()`, `updateMany()`, or `replaceOne()`
- Example:

```
javascript
Copy code
// Update a single document
db.users.updateOne(
  { email: "alice@example.com" },
  { $set: { age: 26, "preferences.newsletter": false } }
);

// Update multiple documents
db.users.updateMany(
  { age: { $lt: 18 } },
  { $set: { status: "minor" } }
);
```

D. Delete

- Used to remove documents from a collection.
- **Command**: `deleteOne()` or `deleteMany()`
- Example:

```
javascript
Copy code
// Delete a single document
db.users.deleteOne({ email: "alice@example.com" });

// Delete multiple documents
db.users.deleteMany({ age: { $lt: 18 } });
```

---

3. Advantages of MongoDB Document Model

1. **Flexibility**: Schema-less design allows rapid iterations and changes.
2. **Embedded Data**: Reduces the need for costly joins.
3. **Dynamic Queries**: Supports rich query language and indexing.
4. **Scalability**: Easily handles large volumes of data.
5. **Developer-Friendly**: JSON-like documents are intuitive and widely used in modern web development.