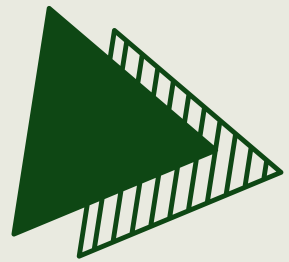


# NOSQL DATABASES

Introduction to NoSQL Data  
Models & MongoDB





## NoSQL Databases

*NoSQL databases are non-relational databases designed for flexible, scalable storage of structured, semi-structured, or unstructured data.*

# NOSQL TYPES

Q1

*Document Store*

**(e.g., MongoDB, CouchDB):**

Stores data as JSON-like documents. Good for flexible and nested data.

Q2

*Key-Value Store*

**(e.g., Redis, DynamoDB):**

Very simple: a key and its value. Super fast lookups.

Q3

*Column Store*

**(e.g., Cassandra, HBase):**

Data stored in columns instead of rows. Great for analytics on big datasets.

Q4

*Graph Database*

**(e.g., Neo4j, ArangoDB):**

Stores nodes and relationships. Best for complex relationships (like social networks).

# WHAT IS MONGODB?

## WHAT DIFFERENTIATES?

---

### Data Format

Documents are stored in **BSON** (Binary JSON) format. BSON takes the same structure as JSON but converts it into binary, making it faster for computers to parse and store.

---

### Collections

Documents are grouped into collections (like tables in SQL).

---

### Flexible Schema

Fields can vary across documents; ***no need to predefine a structure.***

---

# DATATYPES

TYPE	DESCRIPTION	EXAMPLE
String	Text data	"name": "Alice"
Integer	Whole numbers (32-bit or 64-bit)	"age": 25
Double	Decimal numbers (floating point)	"price": 19.99
Boolean	True or false values	"isActive": true

# DATATYPES

TYPE	DESCRIPTION	EXAMPLE
Array	List of values	"skills": ["JS", "Python"]
Embedded Document	A nested document (JSON inside JSON)	"address": { "city": "NY" }
Null	Represents no value	"middleName": null
ObjectId	A unique ID generated automatically for each document	"_id": ObjectId("507f...")
Date	Date and time	"createdAt": ISODate("2024-04-26T12:00:00Z")



# DATA MODELING

MongoDB is schema-less (flexible), you choose how to organize the data depending on what your application needs.

## 1. Embedding (Denormalization)

- Put related data inside the same document.
- Best when the related data is mostly read together.

- ✓ Faster reads
- ✓ Fewer queries
- ✗ Document size limit (16MB)

### Users Collection

```
{
  "name": "Alice",
  "orders": [
    { "product": "Laptop", "price": 1200 },
    { "product": "Phone", "price": 700 }
  ]
}
```

# DATA MODELING

MongoDB is schema-less (flexible), you choose how to organize the data depending on what your application needs.

## 2. Referencing (Normalization)

- Store related data in different documents and link them using IDs.
- Best when related data changes often or is very large.

- ✓ Smaller documents
- ✓ Better for complex, growing data
- ✗ Requires extra queries (joins)

### Users Collection

```
{ "_id": ObjectId("user1"), "name": "Alice" }
```

### Products Collection

```
{ "userId": ObjectId("user1"), "product":  
  "Laptop", "price": 1200 }
```



# DATABASE QUERIES

Create / Switch  
Database

**use databaseName**

Show all Databases

**show databases**

Drop Database

**db.dropDatabase()**

# COLLECTION CRUD

Create Collection  
(manually)

```
db.createCollection("C1")
```

Show Collections

```
show collections
```

Drop Collection

```
db.collectionName.drop()
```

# DOCUMENT (OBJECT) CRUD

Insert One  
Document

```
db.collName.insertOne({  
  field: value  
})
```

Insert Many  
Documents

```
db.collectionName.insert  
Many([ {...}, {...} ])
```

Find All Documents

```
db.collectionName.find()
```

Find with Condition

```
db.collectionName.find({  
  field: value })
```

Update One  
Document

```
db.cName.updateOne(  
  { filter },  
  { $set: { field: newValue } }  
)
```

Update Many  
Document

```
db.cName.updateMany(  
  { filter },  
  { $set: { field: newValue } }  
)
```

Delete One  
Document

```
db.cName.deleteOne(  
  { filter }  
)
```

Delete Many  
Document

```
db.cName.deleteMany(  
  { filter }  
)
```