

Introduction to MongoDB

1. Apa itu MongoDB?

MongoDB adalah sebuah database NoSQL (Not Only SQL) yang bersifat open-source dan document-oriented. MongoDB menyimpan data dalam format dokumen JSON-like yang fleksibel, berbeda dengan database relasional tradisional yang menggunakan tabel dan baris.

Karakteristik Utama MongoDB:

- **Document-Oriented:** Data disimpan dalam bentuk dokumen BSON (Binary JSON)
- **Flexible Schema:** Struktur dokumen dapat berbeda-beda dalam satu koleksi
- **Scalability:** Mendukung horizontal scaling melalui sharding
- **High Performance:** Optimisasi untuk operasi baca dan tulis yang cepat
- **Replication:** Mendukung replica set untuk redundansi data

2. Konsep Dasar MongoDB

Istilah Penting:

MongoDB	SQL	Penjelasan
Database	Database	Kontainer utama untuk menyimpan data
Collection	Table	Kumpulan dokumen (mirip tabel)
Document	Row	Satu record data dalam format JSON
Field	Column	Properti atau atribut dalam dokumen
Index	Index	Struktur untuk mempercepat pencarian

Contoh Struktur Data:

```
// Database: toko_online
// Collection: produk
// Document:
{
  "_id": ObjectId("507f1f77bcf86cd799439011"),
  "nama": "Laptop Gaming",
  "harga": 15000000,
  "kategori": "elektronik",
  "stok": 25,
  "spesifikasi": {
    "prosesor": "Intel i7",
    "ram": "16GB",
    "gpu": "RTX 3070"
  },
  "tags": ["gaming", "laptop", "high-end"],
}
```

```
"tersedia": true,
"tanggal_dibuat": ISODate("2024-01-15T10:30:00Z")
}
```

3. Perbedaan Desain Relasional vs. MongoDB

Karakteristik Tabel:

- Schema rigid dan terdefinisi dengan jelas
- Setiap row harus memiliki struktur yang sama
- Foreign key digunakan untuk relasi
- Normalisasi: data dipecah ke banyak tabel untuk menghindari duplikasi
- Memerlukan JOIN untuk menggabungkan data dari berbagai tabel

Koleksi (MongoDB) :

- Koleksi adalah kumpulan dokumen JSON/BSON yang fleksibel
- Setiap dokumen dapat memiliki struktur yang berbeda
- Field dapat bervariasi antar dokumen dalam satu koleksi

Tabel: alumni

id	nim	nama	email	jurusan	tahun
1	2019001	Ahmad Wijaya	ahmad@example.com	Tek. Inf.	2023
2	2019002	Budi Santoso	budi@example.com	Sistem Info	2022
3	2019003	Citra Dewi	citra@example.com	Tek. Inf.	2023

Tabel: pekerjaan_alumni

id	alumni_id	nama_perusahaan	posisi	gaji	aktif
1	1	PT Teknologi	Backend	15000000	true
2	2	PT Digital Solutions	Frontend	12000000	true
3	1	PT lain	DevOps	18000000	false

```
// Koleksi: alumni
[
  {
    "_id": ObjectId("507f1f77bcf86cd799439011"),
    "nim": "2019001",
    "nama": "Ahmad Wijaya",
    "email": "ahmad@example.com",
    "jurusan": "Teknik Informatika",
```

```

    "tahun_lulus": 2023,
    "no_telepon": "08123456789",
    "alamat": {
      "kota": "Jakarta",
      "provinsi": "DKI Jakarta"
    },
    "pekerjaan_aktif": [
      {
        "nama_perusahaan": "PT Teknologi",
        "posisi": "Backend Developer",
        "gaji": 15000000
      }
    ]
  },
  {
    "_id": ObjectId("507f1f77bcf86cd799439012"),
    "nim": "2019002",
    "nama": "Budi Santoso",
    "email": "budi@example.com",
    "jurusan": "Sistem Informasi",
    "tahun_lulus": 2022,
    "no_telepon": "08123456790"
    // Bisa berbeda struktur dari dokumen lain
  },
  {
    "_id": ObjectId("507f1f77bcf86cd799439013"),
    "nim": "2019003",
    "nama": "Citra Dewi",
    "email": "citra@example.com",
    "jurusan": "Teknik Informatika",
    "tahun_lulus": 2023,
    "status_verifikasi": true, // Field tambahan yang unik
    "sertifikasi": ["AWS", "GCP"] // Array data
  }
]

// Koleksi: pekerjaan_alumni (terpisah atau embedded)
[
  {
    "_id": ObjectId("507f1f77bcf86cd799439021"),
    "alumni_id": ObjectId("507f1f77bcf86cd799439011"),
    "nama_perusahaan": "PT Teknologi Indonesia",
    "posisi": "Backend Developer",
    "gaji": 15000000,
    "status_aktif": true,
    "tanggal_mulai": "2023-06-01"
  }
]

```

```

-- Query untuk mendapatkan alumni dengan pekerjaan
SELECT a.*, p.nama_perusahaan, p.posisi, p.gaji
FROM alumni a
LEFT JOIN pekerjaan_alumni p ON a.id = p.alumni_id
WHERE a.tahun_lulus = 2023;

// Query sederhana tanpa JOIN
db.alumni.find({ tahun_lulus: 2023 });

// Query dengan $lookup (similar to JOIN)
db.pekerjaan_alumni.aggregate([
  {
    $lookup: {
      from: "alumni",
      localField: "alumni_id",
      foreignField: "_id",
      as: "alumni_info"
    }
  },
  { $match: { "alumni_info.tahun_lulus": 2023 } }
]);

```

4. Keuntungan dan Keterbatasan MongoDB

Keuntungan

- Fleksibilitas: Schema tidak rigid, mudah untuk perubahan struktur
- Developer-Friendly: Format JSON mudah dipahami programmer
- Scalability: Dapat menangani volume data yang sangat besar
- Performance: Query cepat dengan indexing yang efisien
- Replication & High Availability: Built-in redundansi data
- Aggregation Framework: Query kompleks dengan pipeline aggregation

Hal yang Perlu Diperhatikan:

- Konsumsi Memory: Lebih boros memori dibanding database relasional
- Tidak Ada JOIN: Tidak mendukung operasi JOIN seperti SQL
- Transactional Support: Untuk transaksi multi-dokumen, perlu perhatian khusus (mulai MongoDB 4.0)
- Redundansi Data: Karena tidak ada JOIN, data sering diduplikasi

- Size Limit: Satu dokumen maksimal 16MB

5. Instalasi dan Setup

Instalasi MongoDB (Windows):

1. Download installer dari <https://www.mongodb.com/try/download/community>
2. Jalankan installer dan ikuti petunjuk wizard
3. Pilih "Install MongoDB as a Service"
4. Selesai, MongoDB siap digunakan

Instalasi MongoDB (Linux - Ubuntu):

```
# Import public key
wget -qO - https://www.mongodb.org/static/pgp/server-7.0.asc | sudo apt-key
add -

# Tambahkan repository
echo "deb [ arch=amd64 ] https://repo.mongodb.org/apt/ubuntu focal/mongodb-
org/7.0 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-7.0.list

# Update dan install
sudo apt-get update
sudo apt-get install -y mongodb-org

# Start service
sudo systemctl start mongod
```

Akses MongoDB Shell:

```
# Gunakan perintah ini untuk membuka MongoDB shell
mongosh
```

6. Operasi Dasar (CRUD)

CREATE - Menambah Data:

```
// Insert satu dokumen
db.produk.insertOne({
  nama: "Mouse Gaming",
  harga: 500000,
  kategori: "aksesori"
})

// Insert banyak dokumen
db.produk.insertMany([
```

```
{ nama: "Keyboard", harga: 750000 },
{ nama: "Monitor", harga: 2000000 }
])
```

READ - Membaca Data:

```
// Tampilkan semua dokumen
db.produk.find()

// Tampilkan dengan kriteria tertentu
db.produk.find({ kategori: "elektronik" })

// Tampilkan satu dokumen pertama
db.produk.findOne({ nama: "Laptop Gaming" })

// Dengan projection (pilih field tertentu)
db.produk.find({}, { nama: 1, harga: 1, _id: 0 })
```

UPDATE - Mengubah Data:

```
// Update satu dokumen
db.produk.updateOne(
  { _id: ObjectId("...") },
  { $set: { harga: 14000000 } }
)

// Update banyak dokumen
db.produk.updateMany(
  { kategori: "elektronik" },
  { $set: { diskon: 10 } }
)
```

DELETE - Menghapus Data:

```
// Hapus satu dokumen
db.produk.deleteOne({ _id: ObjectId("...") })

// Hapus banyak dokumen
db.produk.deleteMany({ kategori: "kadaluarsa" })
```

7. Query Operators

Comparison Operators:

```
// $eq - sama dengan
db.produk.find({ harga: { $eq: 1000000 } })
```

```
// $gt - lebih besar dari
db.produk.find({ harga: { $gt: 1000000 } })

// $gte - lebih besar atau sama dengan
db.produk.find({ harga: { $gte: 1000000 } })

// $lt - lebih kecil dari
db.produk.find({ harga: { $lt: 1000000 } })

// $ne - tidak sama dengan
db.produk.find({ kategori: { $ne: "elektronik" } })

// $in - ada di dalam array
db.produk.find({ kategori: { $in: ["elektronik", "aksesori"] } })
```

Logical Operators:

```
// $and - semua kondisi harus terpenuhi
db.produk.find({ $and: [
  { harga: { $gt: 500000 } },
  { kategori: "elektronik" }
]})

// $or - salah satu kondisi terpenuhi
db.produk.find({ $or: [
  { kategori: "elektronik" },
  { kategori: "aksesori" }
]})

// $not - negasi kondisi
db.produk.find({ harga: { $not: { $gt: 5000000 } } })
```

8. Aggregation Pipeline

Aggregation adalah teknik untuk memproses dan mengubah data kompleks.

```
db.produk.aggregate([
  // Stage 1: Filter
  { $match: { kategori: "elektronik" } },

  // Stage 2: Kelompokkan dan hitung
  { $group: {
    _id: "$kategori",
    jumlah_produk: { $sum: 1 },
    harga_rata: { $avg: "$harga" },
    harga_max: { $max: "$harga" }
  }},
])
```

```
// Stage 3: Urutkan
{ $sort: { jumlah_produk: -1 } }
])
```

9. Indexing

Index mempercepat query dengan membuat struktur pencarian yang lebih efisien.

```
// Buat index pada field nama
db.produk.createIndex({ nama: 1 })

// Buat index compound (multiple fields)
db.produk.createIndex({ kategori: 1, harga: -1 })

// Buat text index untuk pencarian text
db.produk.createIndex({ deskripsi: "text" })

// Lihat semua index
db.produk.getIndexes()

// Hapus index
db.produk.dropIndex({ nama: 1 })
```

11. Best Practices MongoDB

1. Pilih _id dengan bijak: Gunakan default ObjectId atau custom ID yang bermakna
2. Denormalisasi dengan hati-hati: Embed dokumen untuk akses cepat, reference untuk menghemat storage
3. Gunakan Indexes: Buat index pada field yang sering di-query
4. Monitor Performance: Gunakan explain() untuk analisis query
5. Data Validation: Validasi data di aplikasi atau gunakan schema validation
6. Backup Regular: Lakukan backup berkala untuk keamanan data
7. Optimize Query: Hindari full collection scan, gunakan filter yang efisien
8. Limit Result: Gunakan limit() dan skip() untuk pagination

Latihan 1.1: Membuat Database dan Collection

Task:

1. Buka MongoDB Shell (mongosh)
2. Buat database baru bernama alumni_management_db
3. Buat 2 collection:
 - o alumni - untuk data alumni
 - o pekerjaan_alumni - untuk data pekerjaan alumni

Kode yang harus dijalankan:

```
// Membuat database
use alumni_management_db

// Membuat collection alumni
db.createCollection("alumni")

// Membuat collection pekerjaan_alumni
db.createCollection("pekerjaan_alumni")

// Verifikasi collection yang dibuat
show collections
```

Latihan 1.2: Insert Sample Data Alumni

Task: Insert minimal 10 dokumen alumni dengan data beragam (berbagai jurusan, tahun lulus, dll)

- Minimal 10 alumni
- Minimal 3 jurusan berbeda
- Minimal 4 tahun lulus berbeda
- Email unik untuk setiap alumni

```
use alumni_management_db

db.alumni.insertMany([
  {
    nim: "2019001",
    nama: "Ahmad Wijaya",
    email: "ahmad@example.com",
    jurusan: "Teknik Informatika",
    tahun_lulus: 2023,
```

```

        no_telepon: "08123456789"
    },
    {
        nim: "2019002",
        nama: "Budi Santoso",
        email: "budi@example.com",
        jurusan: "Sistem Informasi",
        tahun_lulus: 2022,
        no_telepon: "08123456790"
    },
    // TODO: Tambahkan 8 dokumen lagi dengan data yang beragam
    // Pastikan ada alumni dari berbagai jurusan:
    // - Teknik Informatika
    // - Sistem Informasi
    // - Teknik Komputer
    // - Manajemen Informatika
    //
    // Dan tahun lulus yang berbeda:
    // - 2020, 2021, 2022, 2023, 2024
])

```

Latihan 1.3: Insert Sample Data Pekerjaan Alumni

Task: Insert minimal 15 dokumen pekerjaan alumni dengan data beragam

- Minimal 15 pekerjaan
- Minimal 8 alumni berbeda yang memiliki pekerjaan
- Mix antara status_aktif true dan false
- Gaji range: 5juta - 30juta
- Minimal 5 posisi berbeda

```

use alumni_management_db

var alumni_ids = db.alumni.find({}, {_id: 1}).limit(10).toArray().map(x => x._id)

db.pekerjaan_alumni.insertMany([
  {
    alumni_id: alumni_ids[0],
    nama_perusahaan: "PT Teknologi Indonesia",
    posisi: "Backend Developer",
    departemen_usaha: "Engineering",
    tanggal_mulai: "2023-06-01",
    gaji: 15000000,
    status_aktif: true
  },
  {
    alumni_id: alumni_ids[1],

```

```

    nama_perusahaan: "PT Digital Solutions",
    posisi: "Frontend Developer",
    departemen_usaha: "Engineering",
    tanggal_mulai: "2022-08-15",
    gaji: 12000000,
    status_aktif: true
  },
  // TODO: Tambahkan 13 dokumen lagi
  // Pastikan ada pekerjaan dengan:
  // - Status aktif dan tidak aktif
  // - Gaji bervariasi (5jt - 30jt)
  // - Berbagai posisi: Backend Developer, Frontend Developer,
  //   Full Stack Developer, DevOps Engineer, Data Analyst, dll
  // - Berbagai perusahaan
])

```

Latihan 2.1: Buat query untuk menampilkan semua dokumen alumni

Latihan 2.2: Find Alumni Tertentu (Buat query untuk menemukan semua alumni dari jurusan "Teknik Informatika")

Latihan 2.3: Find dengan Projection : Buat query untuk menampilkan nama dan email alumni saja, tanpa field lainnya dan tanpa _id

```

{ "nama": "Ahmad Wijaya", "email": "ahmad@example.com" }
{ "nama": "Budi Santoso", "email": "budi@example.com" }

```

Latihan 2.4 FindOne untuk mencari alumni dengan nama "Ahmad Wijaya"

Latihan 2.5 Find dengan Multiple Conditions (AND) untuk menemukan alumni yang:

- Lulus pada tahun 2023
- Dari jurusan Teknik Informatika

Latihan 3.1 Operator \$gt (Greater Than) : Buat query untuk menemukan pekerjaan dengan gaji lebih besar dari 10 juta

Latihan 3.2: Operator \$lt (Less Than) Buat query untuk menemukan pekerjaan dengan gaji kurang dari 10 juta

Latihan 3.3: Operator \$in Buat query untuk menemukan alumni dari jurusan "Teknik Informatika" ATAU "Sistem Informasi"

Latihan 3.4: Operator \$or Buat query untuk menemukan alumni yang:

- Lulus tahun 2023 ATAU
- Dari jurusan Teknik Informatika

Latihan 3.5: Operator \$regex (Pattern Matching) Buat query untuk menemukan alumni yang email-nya mengandung kata "example" (case-insensitive)