# Panduan Praktik Proyek Sederhana MongoDB dengan Mongo Express

## Daftar Isi

1. [Pendahuluan](#pendahuluan)
2. [Prasyarat](#prasyarat)
3. [Konsep Dasar MongoDB](#konsep-dasar-mongodb)
4. [Instalasi Docker](#instalasi-docker)
5. [Setup MongoDB dengan Docker](#setup-mongodb-dengan-docker)
6. [Instalasi Mongo Express](#instalasi-mongo-express)
7. [Membuat Proyek Sederhana: Sistem Manajemen Kontak](#Xe62dfacbbb9de742f53b2e635b7f482ef2549f5)
8. [Operasi CRUD Dasar](#operasi-crud-dasar)
9. [Query Lanjutan](#query-lanjutan)
10. [Indexing dan Optimasi](#indexing-dan-optimasi)
11. [Backup dan Restore](#backup-dan-restore)
12. [Troubleshooting](#troubleshooting)
13. [Latihan Tambahan](#latihan-tambahan)
14. [Referensi](#referensi)

## Pendahuluan

MongoDB adalah database NoSQL yang populer dan banyak digunakan dalam industri modern. Dokumen ini akan memandu mahasiswa langkah demi langkah dalam membuat proyek sederhana menggunakan MongoDB dengan antarmuka grafis Mongo Express yang berjalan di dalam container Docker.

### Tujuan Pembelajaran

Setelah menyelesaikan praktik ini, mahasiswa diharapkan dapat:

* Memahami konsep dasar database NoSQL dan MongoDB
* Menginstal dan mengonfigurasi MongoDB dengan Docker
* Menggunakan Mongo Express untuk manajemen database
* Membuat aplikasi sederhana dengan operasi CRUD
* Melakukan query dan indexing dasar
* Memahami best practices dalam pengembangan aplikasi dengan MongoDB

### Mengapa Docker?

Docker menyederhanakan proses instalasi dan pengelolaan lingkungan pengembangan. Dengan Docker, kita tidak perlu menginstal MongoDB secara langsung di sistem operasi, sehingga menghindari konflik dengan software lain dan memudahkan reproduksi lingkungan yang sama di berbagai mesin.

## Prasyarat

Sebelum memulai praktik ini, pastikan Anda telah memenuhi persyaratan berikut:

### Sistem Operasi yang Didukung

* **Windows 10/11** (dengan WSL2 enabled)
* **macOS** (versi 10.14 atau lebih baru)
* **Linux** (Ubuntu 18.04+, CentOS 7+, atau distribusi lainnya)

### Spesifikasi Minimum Sistem

| Komponen | Minimum | Rekomendasi |
| --- | --- | --- |
| RAM | 4 GB | 8 GB atau lebih |
| Ruang Disk | 10 GB | 20 GB atau lebih |
| Processor | 2 core | 4 core atau lebih |

### Software yang Diperlukan

1. **Docker Desktop** (versi 20.10 atau lebih baru)
2. **Docker Compose** (biasanya sudah terinstal dengan Docker Desktop)
3. **Text Editor** (VS Code, Sublime Text, atau Atom)
4. **Terminal/Command Prompt**
5. **Web Browser** (Chrome, Firefox, atau Edge)

### Verifikasi Instalasi Docker

Buka terminal dan jalankan perintah berikut untuk memverifikasi instalasi Docker:

# Cek versi Docker  
docker --version  
  
# Cek versi Docker Compose  
docker-compose --version  
  
# Test Docker dengan menjalankan container hello-world  
docker run hello-world

Jika semua perintah berjalan dengan baik, sistem Anda siap untuk praktik ini.

## Konsep Dasar MongoDB

Sebelum memulai praktik, penting untuk memahami konsep fundamental MongoDB yang membedakannya dari database relasional tradisional [(PRAKTIKUM 10 DATABASE NOSQL, n.d.)](https://yunia.lecturer.pens.ac.id/Praktikum%20Basis%20Data%20Lanjut/10%20Praktikum%20Database%20NoSQL.pdf).

### Arsitektur MongoDB

MongoDB menggunakan arsitektur yang berbeda dengan database SQL. Berikut adalah perbandingannya:

graph TD  
 A[Database SQL] --> B[Tables]  
 A --> C[Rows]  
 A --> D[Columns]  
   
 E[Database MongoDB] --> F[Collections]  
 E --> G[Documents]  
 E --> H[Fields]  
   
 style A fill:#ffcccc  
 style E fill:#ccffcc

### Terminologi MongoDB

| Terminologi SQL | Terminologi MongoDB | Deskripsi |
| --- | --- | --- |
| Database | Database | Kontainer untuk collections |
| Table | Collection | Grup dokumen MongoDB |
| Row | Document | Struktur data tunggal dalam BSON |
| Column | Field | Pasangan key-value dalam dokumen |
| Primary Key | \_id | Unique identifier otomatis |

### Struktur Data MongoDB

MongoDB menyimpan data dalam format BSON (Binary JSON). Contoh struktur dokumen:

{  
 "\_id": ObjectId("635a1b2c3d4e5f6789abc123"),  
 "nama": "John Doe",  
 "email": "john@example.com",  
 "umur": 25,  
 "alamat": {  
 "jalan": "Jl. Sudirman No. 123",  
 "kota": "Jakarta",  
 "kode\_pos": "12345"  
 },  
 "hobi": ["membaca", "berenang", "coding"],  
 "tanggal\_registrasi": ISODate("2023-10-30T10:00:00Z")  
}

### Keuntungan MongoDB

1. **Schema Flexibility**: Struktur dokumen dapat bervariasi dalam satu collection
2. **Scalability**: Mudah untuk scaling horizontal
3. **Performance**: Optimized untuk read operations
4. **Rich Queries**: Mendukung query yang kompleks dan aggregation
5. **Document-oriented**: Cocok untuk aplikasi modern dengan data kompleks

## Instalasi Docker

Proses instalasi Docker berbeda untuk setiap sistem operasi. Ikuti panduan berikut sesuai dengan sistem yang Anda gunakan.

### Instalasi di Windows

1. **Download Docker Desktop**
   * Kunjungi [docker.com/products/docker-desktop](https://www.docker.com/products/docker-desktop)
   * Download installer untuk Windows
   * Pastikan sistem Anda menggunakan Windows 10/11 dengan fitur WSL2
2. **Enable WSL2**

# Buka PowerShell sebagai Administrator  
dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart  
dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart

1. **Install Docker Desktop**
   * Jalankan installer yang telah diunduh
   * Restart komputer setelah instalasi
   * Launch Docker Desktop dari Start Menu

### Instalasi di macOS

1. **Download Docker Desktop**
   * Kunjungi [docker.com/products/docker-desktop](https://www.docker.com/products/docker-desktop)
   * Pilih versi untuk Mac dengan chip Intel atau Apple Silicon
2. **Install Docker Desktop**
   * Buka file .dmg yang telah diunduh
   * Drag Docker.app ke folder Applications
   * Launch Docker dari Launchpad
3. **Konfigurasi**
   * Ikuti wizard setup awal
   * Berikan password sistem saat diminta
   * Tunggu hingga Docker selesai menginisialisasi

### Instalasi di Linux (Ubuntu)

1. **Update System Packages**

sudo apt-get update  
sudo apt-get install ca-certificates curl gnupg

1. **Add Docker’s GPG Key**

sudo install -m 0755 -d /etc/apt/keyrings  
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg  
sudo chmod a+r /etc/apt/keyrings/docker.gpg

1. **Add Docker Repository**

echo \  
 "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \  
 $(. /etc/os-release && echo "$VERSION\_CODENAME") stable" | \  
 sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

1. **Install Docker**

sudo apt-get update  
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

1. **Configure Docker to Run Without sudo**

sudo groupadd docker  
sudo usermod -aG docker $USER  
newgrp docker

### Verifikasi Instalasi

Setelah instalasi selesai, verifikasi dengan perintah berikut:

# Test instalasi Docker  
docker run hello-world  
  
# Check Docker info  
docker info  
  
# List running containers  
docker ps

## Setup MongoDB dengan Docker

Setelah Docker terinstal, langkah selanjutnya adalah mengatur lingkungan MongoDB menggunakan container Docker.

### Metode 1: Menggunakan Docker Run

Cara tercepat untuk menjalankan MongoDB adalah dengan perintah docker run:

# Pull image MongoDB  
docker pull mongo:latest  
  
# Jalankan container MongoDB  
docker run --name mongodb-container \  
 -p 27017:27017 \  
 -d mongo:latest  
  
# Verifikasi container berjalan  
docker ps

### Metode 2: Menggunakan Docker Compose (Rekomendasi)

Docker Compose memungkinkan kita mendefinisikan layanan dalam file YAML. Buat file docker-compose.yml:

version: '3.8'  
  
services:  
 mongodb:  
 image: mongo:latest  
 container\_name: mongodb-server  
 restart: always  
 ports:  
 - "27017:27017"  
 environment:  
 MONGO\_INITDB\_ROOT\_USERNAME: admin  
 MONGO\_INITDB\_ROOT\_PASSWORD: password123  
 volumes:  
 - mongodb\_data:/data/db  
 - ./init-mongo.js:/docker-entrypoint-initdb.d/init-mongo.js:ro  
  
volumes:  
 mongodb\_data:

### Konfigurasi Lanjutan

Untuk penggunaan produksi, tambahkan konfigurasi berikut:

version: '3.8'  
  
services:  
 mongodb:  
 image: mongo:6.0  
 container\_name: mongodb-server  
 restart: always  
 ports:  
 - "27017:27017"  
 environment:  
 MONGO\_INITDB\_ROOT\_USERNAME: admin  
 MONGO\_INITDB\_ROOT\_PASSWORD: password123  
 MONGO\_INITDB\_DATABASE: aplikasi\_db  
 volumes:  
 - mongodb\_data:/data/db  
 - ./mongo-config:/etc/mongo  
 - ./init-scripts:/docker-entrypoint-initdb.d  
 command: mongod --auth  
   
 mongo-express:  
 image: mongo-express:latest  
 container\_name: mongo-express-ui  
 restart: always  
 ports:  
 - "8081:8081"  
 environment:  
 ME\_CONFIG\_MONGODB\_ADMINUSERNAME: admin  
 ME\_CONFIG\_MONGODB\_ADMINPASSWORD: password123  
 ME\_CONFIG\_MONGODB\_URL: mongodb://admin:password123@mongodb:27017/  
 depends\_on:  
 - mongodb  
  
volumes:  
 mongodb\_data:

### Inisialisasi Database

Buat file init-mongo.js untuk inisialisasi awal database:

// init-mongo.js  
db = db.getSiblingDB('aplikasi\_db');  
  
// Buat user aplikasi  
db.createUser({  
 user: 'app\_user',  
 pwd: 'app\_password',  
 roles: [  
 {  
 role: 'readWrite',  
 db: 'aplikasi\_db'  
 }  
 ]  
});  
  
// Buat collection awal  
db.createCollection('users');  
db.createCollection('produk');  
db.createCollection('transaksi');  
  
// Insert data awal  
db.users.insertOne({  
 nama: "Admin User",  
 email: "admin@example.com",  
 role: "admin",  
 created\_at: new Date()  
});

### Menjalankan Container

# Jalankan semua services  
docker-compose up -d  
  
# Cek logs  
docker-compose logs -f mongodb  
  
# Cek status container  
docker-compose ps

### Koneksi ke MongoDB

Anda dapat terhubung ke MongoDB menggunakan beberapa metode:

1. **Mongo Shell**

# Connect ke container  
docker exec -it mongodb-server mongosh  
  
# Connect dengan authentication  
docker exec -it mongodb-server mongosh -u admin -p password123 --authenticationDatabase admin

1. **MongoDB Compass** (GUI Desktop)
   * Connection string: mongodb://admin:password123@localhost:27017/
2. **Dari Aplikasi**

// Node.js example  
const { MongoClient } = require('mongodb');  
const uri = "mongodb://admin:password123@localhost:27017/?authSource=admin";

## Instalasi Mongo Express

Mongo Express adalah antarmuka web berbasis Node.js untuk administrasi MongoDB. Ini menyediakan GUI yang user-friendly untuk mengelola database MongoDB.

### Cara Instalasi Mongo Express

#### Metode 1: Standalone

# Pull image Mongo Express  
docker pull mongo-express:latest  
  
# Jalankan Mongo Express  
docker run --name mongo-express \  
 --link mongodb-server:mongo \  
 -p 8081:8081 \  
 -e ME\_CONFIG\_MONGODB\_ADMINUSERNAME=admin \  
 -e ME\_CONFIG\_MONGODB\_ADMINPASSWORD=password123 \  
 -e ME\_CONFIG\_MONGODB\_URL=mongodb://admin:password123@mongodb:27017/ \  
 -d mongo-express

#### Metode 2: Dengan Docker Compose (Sudah termasuk di file sebelumnya)

Jika Anda menggunakan file docker-compose.yml dari bagian sebelumnya, Mongo Express sudah terkonfigurasi dan akan otomatis terinstall saat menjalankan docker-compose up.

### Konfigurasi Mongo Express

#### Environment Variables

| Variable | Deskripsi | Default |
| --- | --- | --- |
| ME\_CONFIG\_MONGODB\_SERVER | MongoDB server hostname | localhost |
| ME\_CONFIG\_MONGODB\_PORT | MongoDB port | 27017 |
| ME\_CONFIG\_MONGODB\_ADMINUSERNAME | Admin username | - |
| ME\_CONFIG\_MONGODB\_ADMINPASSWORD | Admin password | - |
| ME\_CONFIG\_MONGODB\_URL | Connection string | - |
| ME\_CONFIG\_BASICAUTH\_USERNAME | Basic auth username | - |
| ME\_CONFIG\_BASICAUTH\_PASSWORD | Basic auth password | - |

#### Konfigurasi Advanced

Buat file config.default.js untuk konfigurasi lanjutan:

module.exports = {  
 mongodb: {  
 server: 'mongodb',  
 port: 27017,  
 // SSL options  
 ssl: false,  
 sslValidate: false,  
 sslCA: null,  
 authentication: {  
 authSource: 'admin'  
 }  
 },  
 site: {  
 baseUrl: '/',  
 cookieKeyName: 'mongo-express',  
 sessionSecret: process.env.ME\_CONFIG\_SITE\_SESSION\_SECRET || 'sessionscret',  
 cookieSecret: process.env.ME\_CONFIG\_SITE\_COOKIE\_SECRET || 'cookiesecret'  
 },  
 setBasicAuth: {  
 username: process.env.ME\_CONFIG\_BASICAUTH\_USERNAME,  
 password: process.env.ME\_CONFIG\_BASICAUTH\_PASSWORD  
 }  
};

### Mengakses Mongo Express

1. **Buka Browser**
   * Navigate ke http://localhost:8081
   * Login dengan credentials yang telah diset
2. **Interface Overview**
   * **Database List**: Daftar semua database
   * **Collection View**: Tampilan dokumen dalam collection
   * **Query Builder**: Interface untuk membuat query
   * **Index Manager**: Manajemen index
   * **User Management**: Manajemen user database

### Fitur-Fitur Utama Mongo Express

![Mongo Express Interface](data:image/png;base64;base64,)

Mongo Express Interface

#### 1. Database Management

* View semua database yang tersedia
* Create new database
* Drop database
* View database statistics

#### 2. Collection Operations

* Create new collection
* Insert documents
* View documents dengan pagination
* Edit documents
* Delete documents
* Import/Export documents

#### 3. Query Builder

Interface visual untuk membuat query MongoDB:

// Example query di Mongo Express  
{  
 "nama": "John Doe",  
 "umur": { "$gt": 18 }  
}

#### 4. Index Management

* Create indexes
* View existing indexes
* Drop indexes
* Compound indexes

### Security Best Practices

1. **Jangan expose ke public internet tanpa authentication**
2. **Gunakan HTTPS di production**
3. **Batasi akses dengan firewall**
4. **Gunakan environment variables untuk sensitive data**
5. **Regular backup database**

### Tips Penggunaan Mongo Express

* **Use the Query Builder untuk query kompleks**
* **Export data sebagai JSON atau CSV**
* **Use aggregation pipeline untuk analisis data**
* **Monitor performance dengan explain()**
* **Backup regular melalui interface**

## Membuat Proyek Sederhana: Sistem Manajemen Kontak

Sekarang kita akan membuat proyek sederhana berupa Sistem Manajemen Kontak menggunakan MongoDB dan Mongo Express. Proyek ini akan mendemonstrasikan konsep dasar database NoSQL dalam aplikasi nyata.

### Spesifikasi Proyek

Sistem Manajemen Kontak akan memiliki fitur:

1. **Manajemen Kontak** - Tambah, edit, hapus kontak
2. **Kategorisasi** - Grup kontak (keluarga, teman, kerja)
3. **Pencarian** - Cari kontak berdasarkan nama atau email
4. **Export Data** - Export kontak ke format CSV/JSON
5. **Import Batch** - Import kontak dari file

### Desain Database

#### Collection: kontak

{  
 "\_id": ObjectId("635a1b2c3d4e5f6789abc123"),  
 "nama\_depan": "John",  
 "nama\_belakang": "Doe",  
 "email": "john.doe@example.com",  
 "telepon": [  
 {  
 "tipe": "mobile",  
 "nomor": "+628123456789",  
 "utama": true  
 },  
 {  
 "tipe": "office",  
 "nomor": "+622212345678",  
 "utama": false  
 }  
 ],  
 "alamat": {  
 "jalan": "Jl. Sudirman No. 123",  
 "kota": "Jakarta",  
 "provinsi": "DKI Jakarta",  
 "kode\_pos": "12345",  
 "negara": "Indonesia"  
 },  
 "tanggal\_lahir": ISODate("1990-01-15T00:00:00Z"),  
 "grup": "teman",  
 "foto": "https://example.com/foto/johndoe.jpg",  
 "catatan": "Kolega dari kantor lama",  
 "tags": ["programming", "javascript", "nodejs"],  
 "dibuat\_pada": ISODate("2023-10-30T10:00:00Z"),  
 "diperbarui\_pada": ISODate("2023-10-30T10:00:00Z"),  
 "status": "aktif"  
}

#### Collection: grup

{  
 "\_id": ObjectId("635a1b2c3d4e5f6789abc124"),  
 "nama": "keluarga",  
 "deskripsi": "Anggota keluarga dekat",  
 "warna": "#FF5733",  
 "dibuat\_pada": ISODate("2023-10-30T10:00:00Z"),  
 "status": "aktif"  
}

### Langkah 1: Membuka Mongo Express

1. Buka browser
2. Navigate ke http://localhost:8081
3. Login dengan credentials yang telah diset

### Langkah 2: Membuat Database

1. Klik **Create Database**
2. Masukkan nama database: kontak\_manager
3. Klik **Create**

### Langkah 3: Membuat Collections

#### Buat Collection grup

1. Pilih database kontak\_manager
2. Klik **New Collection**
3. Masukkan nama collection: grup
4. Klik **Create**

#### Buat Collection kontak

1. Klik **New Collection**
2. Masukkan nama collection: kontak
3. Klik **Create**

### Langkah 4: Insert Data Awal

#### Insert Data Grup

Klik collection grup → **Insert Document**:

{  
 "nama": "keluarga",  
 "deskripsi": "Anggota keluarga dekat",  
 "warna": "#FF5733",  
 "dibuat\_pada": {  
 "$date": "2023-10-30T10:00:00.000Z"  
 },  
 "status": "aktif"  
}

{  
 "nama": "teman",  
 "deskripsi": "Teman-teman dekat",  
 "warna": "#33FF57",  
 "dibuat\_pada": {  
 "$date": "2023-10-30T10:00:00.000Z"  
 },  
 "status": "aktif"  
}

{  
 "nama": "kerja",  
 "deskripsi": "Rekan kerja",  
 "warna": "#3357FF",  
 "dibuat\_pada": {  
 "$date": "2023-10-30T10:00:00.000Z"  
 },  
 "status": "aktif"  
}

#### Insert Data Kontak

Klik collection kontak → **Insert Document**:

{  
 "nama\_depan": "Ahmad",  
 "nama\_belakang": "Pratama",  
 "email": "ahmad.pratama@example.com",  
 "telepon": [  
 {  
 "tipe": "mobile",  
 "nomor": "+6281234567890",  
 "utama": true  
 }  
 ],  
 "alamat": {  
 "jalan": "Jl. Thamrin No. 45",  
 "kota": "Jakarta Pusat",  
 "provinsi": "DKI Jakarta",  
 "kode\_pos": "10230",  
 "negara": "Indonesia"  
 },  
 "tanggal\_lahir": {  
 "$date": "1992-05-15T00:00:00.000Z"  
 },  
 "grup": "teman",  
 "tags": ["developer", "javascript"],  
 "dibuat\_pada": {  
 "$date": "2023-10-30T10:00:00.000Z"  
 },  
 "diperbarui\_pada": {  
 "$date": "2023-10-30T10:00:00.000Z"  
 },  
 "status": "aktif"  
}

{  
 "nama\_depan": "Siti",  
 "nama\_belakang": "Nurhaliza",  
 "email": "siti.nurhaliza@example.com",  
 "telepon": [  
 {  
 "tipe": "mobile",  
 "nomor": "+6282345678901",  
 "utama": true  
 },  
 {  
 "tipe": "home",  
 "nomor": "+622134567890",  
 "utama": false  
 }  
 ],  
 "alamat": {  
 "jalan": "Jl. Gatot Subroto No. 78",  
 "kota": "Bandung",  
 "provinsi": "Jawa Barat",  
 "kode\_pos": "40111",  
 "negara": "Indonesia"  
 },  
 "tanggal\_lahir": {  
 "$date": "1990-12-20T00:00:00.000Z"  
 },  
 "grup": "keluarga",  
 "tags": ["designer", "ui/ux"],  
 "dibuat\_pada": {  
 "$date": "2023-10-30T10:00:00.000Z"  
 },  
 "diperbarui\_pada": {  
 "$date": "2023-10-30T10:00:00.000Z"  
 },  
 "status": "aktif"  
}

### Langkah 5: Membuat Index

Index penting untuk optimasi query performance:

// Index untuk pencarian nama  
db.kontak.createIndex({ "nama\_depan": 1, "nama\_belakang": 1 });  
  
// Index untuk email (unique)  
db.kontak.createIndex({ "email": 1 }, { unique: true });  
  
// Index untuk grup  
db.kontak.createIndex({ "grup": 1 });  
  
// Index untuk tags  
db.kontak.createIndex({ "tags": 1 });  
  
// Index compound untuk pencarian lanjutan  
db.kontak.createIndex({   
 "grup": 1,   
 "status": 1,   
 "dibuat\_pada": -1   
});

### Langkah 6: Validasi Schema

MongoDB mendukung schema validation untuk menjaga konsistensi data:

db.runCommand({  
 collMod: "kontak",  
 validator: {  
 $jsonSchema: {  
 bsonType: "object",  
 required: ["nama\_depan", "email", "telepon", "grup"],  
 properties: {  
 nama\_depan: {  
 bsonType: "string",  
 minLength: 2,  
 maxLength: 50  
 },  
 email: {  
 bsonType: "string",  
 pattern: "^[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,}$"  
 },  
 telepon: {  
 bsonType: "array",  
 minItems: 1,  
 items: {  
 bsonType: "object",  
 required: ["tipe", "nomor"],  
 properties: {  
 tipe: {  
 enum: ["mobile", "home", "office", "other"]  
 },  
 nomor: {  
 bsonType: "string",  
 pattern: "^\\+?[0-9]{10,15}$"  
 }  
 }  
 }  
 }  
 }  
 }  
 },  
 validationLevel: "moderate",  
 validationAction: "error"  
});

### Langkah 7: Aggregation Pipeline

Contoh aggregation untuk analisis data:

// Hitung jumlah kontak per grup  
db.kontak.aggregate([  
 {  
 $match: { status: "aktif" }  
 },  
 {  
 $group: {  
 \_id: "$grup",  
 jumlah: { $sum: 1 },  
 email\_list: { $push: "$email" }  
 }  
 },  
 {  
 $sort: { jumlah: -1 }  
 }  
]);  
  
// Cari kontak dengan umur tertentu  
db.kontak.aggregate([  
 {  
 $addFields: {  
 umur: {  
 $dateDiff: {  
 startDate: "$tanggal\_lahir",  
 endDate: "$$NOW",  
 unit: "year"  
 }  
 }  
 }  
 },  
 {  
 $match: { umur: { $gte: 25 } }  
 },  
 {  
 $project: {  
 nama: { $concat: ["$nama\_depan", " ", "$nama\_belakang"] },  
 email: 1,  
 umur: 1,  
 grup: 1  
 }  
 }  
]);

### Langkah 8: Testing Query

Test berbagai query untuk memastikan sistem berfungsi:

// Cari kontak berdasarkan nama  
db.kontak.find({  
 $or: [  
 { nama\_depan: /Ahmad/i },  
 { nama\_belakang: /Ahmad/i }  
 ]  
});  
  
// Cari berdasarkan grup  
db.kontak.find({ grup: "teman" });  
  
// Cari dengan multiple tags  
db.kontak.find({   
 tags: { $all: ["developer", "javascript"] }  
});  
  
// Update kontak  
db.kontak.updateOne(  
 { \_id: ObjectId("635a1b2c3d4e5f6789abc123") },  
 {   
 $set: { diperbarui\_pada: new Date() },  
 $push: { tags: "updated" }  
 }  
);

## Operasi CRUD Dasar

CRUD (Create, Read, Update, Delete) adalah operasi fundamental dalam database management. MongoDB menyediakan metode yang intuitif untuk melakukan operasi ini [(Tugas Praktikum Mongodb, n.d.)](https://id.scribd.com/document/730001683/TUGAS-PRAKTIKUM-MONGODB-4).

### Create Operations

#### Single Document Insert

// Method 1: insertOne()  
db.kontak.insertOne({  
 "nama\_depan": "Budi",  
 "nama\_belakang": "Santoso",  
 "email": "budi.santoso@example.com",  
 "telepon": [  
 {  
 "tipe": "mobile",  
 "nomor": "+6283456789012",  
 "utama": true  
 }  
 ],  
 "grup": "kerja",  
 "dibuat\_pada": new Date(),  
 "status": "aktif"  
});  
  
// Method 2: insert() (deprecated)  
db.kontak.insert({  
 "nama\_depan": "Dewi",  
 "nama\_belakang": "Lestari",  
 "email": "dewi.lestari@example.com",  
 "grup": "teman"  
});

#### Multiple Document Insert

// Insert multiple documents  
db.kontak.insertMany([  
 {  
 "nama\_depan": "Rizki",  
 "nama\_belakang": "Putra",  
 "email": "rizki.putra@example.com",  
 "grup": "teman",  
 "tags": ["gamer", "streamer"]  
 },  
 {  
 "nama\_depan": "Maya",  
 "nama\_belakang": "Sari",  
 "email": "maya.sari@example.com",  
 "grup": "kerja",  
 "tags": ["marketing", "sales"]  
 },  
 {  
 "nama\_depan": "Fajar",  
 "nama\_belakang": "Hidayat",  
 "email": "fajar.hidayat@example.com",  
 "grup": "keluarga",  
 "tags": ["student", "programming"]  
 }  
]);  
  
// Insert dengan ordered flag  
db.kontak.insertMany(  
 [  
 { "nama\_depan": "User1", "email": "user1@example.com" },  
 { "nama\_depan": "User2", "email": "user2@example.com" },  
 { "nama\_depan": "User3", "email": "user1@example.com" } // Duplicate email  
 ],  
 { ordered: false } // Continue on error  
);

#### Best Practices untuk Create Operations

1. **Gunakan insertOne() untuk single document**
2. **Gunakan insertMany() untuk batch insert**
3. **Validasi data sebelum insert**
4. **Gunakan transactions untuk multiple collection operations**
5. **Handle duplicate key errors**

### Read Operations

#### Basic Find

// Find all documents  
db.kontak.find();  
  
// Find with criteria  
db.kontak.find({ grup: "teman" });  
  
// Find with projection  
db.kontak.find(  
 { grup: "teman" },  
 { nama\_depan: 1, email: 1, \_id: 0 }  
);  
  
// Find with pretty print  
db.kontak.find().pretty();

#### Query Operators

// Comparison operators  
db.kontak.find({ umur: { $gt: 25 } }); // Greater than  
db.kontak.find({ umur: { $gte: 25 } }); // Greater than or equal  
db.kontak.find({ umur: { $lt: 30 } }); // Less than  
db.kontak.find({ umur: { $lte: 30 } }); // Less than or equal  
db.kontak.find({ umur: { $ne: null } }); // Not equal  
db.kontak.find({ umur: { $in: [25, 30, 35] } }); // In array  
db.kontak.find({ umur: { $nin: [20, 21] } }); // Not in array  
  
// Logical operators  
db.kontak.find({ $and: [{ grup: "teman" }, { status: "aktif" }] });  
db.kontak.find({ $or: [{ grup: "teman" }, { grup: "kerja" }] });  
db.kontak.find({ $nor: [{ grup: "teman" }, { status: "nonaktif" }] });  
  
// Element operators  
db.kontak.find({ email: { $exists: true } });  
db.kontak.find({ tags: { $type: "array" } });

#### Array Operations

// Find documents with specific array value  
db.kontak.find({ tags: "programming" });  
  
// Find documents with multiple array values  
db.kontak.find({ tags: { $all: ["programming", "javascript"] } });  
  
// Find by array size  
db.kontak.find({ telepon: { $size: 2 } });  
  
// Array element matching  
db.kontak.find({ "telepon.tipe": "mobile" });  
  
// Nested array query  
db.kontak.find({  
 telepon: {  
 $elemMatch: {  
 tipe: "mobile",  
 utama: true  
 }  
 }  
});

#### Regular Expressions

// Case insensitive search  
db.kontak.find({ nama\_depan: /ahmad/i });  
  
// Starts with  
db.kontak.find({ email: /^ahmad/ });  
  
// Ends with  
db.kontak.find({ email: /example\.com$/ });  
  
// Contains  
db.kontak.find({ nama\_depan: /a/ });

### Update Operations

#### Update Single Document

// updateOne - modifies first matching document  
db.kontak.updateOne(  
 { email: "ahmad.pratama@example.com" },  
 { $set: { status: "updated" } }  
);  
  
// Add field to document  
db.kontak.updateOne(  
 { \_id: ObjectId("635a1b2c3d4e5f6789abc123") },  
 {  
 $set: { "alamat.negara": "Indonesia" },  
 $push: { tags: "verified" },  
 $inc: { login\_count: 1 }  
 }  
);

#### Update Multiple Documents

// updateMany - modifies all matching documents  
db.kontak.updateMany(  
 { grup: "teman" },  
 { $set: { kategori\_prioritas: "medium" } }  
);  
  
// Update with conditions  
db.kontak.updateMany(  
 {   
 grup: "kerja",  
 "telepon.tipe": "mobile"  
 },  
 { $set: { "telepon.$[].verified": true } }  
);

#### Update Operators

// $set - Set or update field value  
db.kontak.updateOne(  
 { \_id: ObjectId("...") },  
 { $set: { last\_login: new Date() } }  
);  
  
// $unset - Remove field  
db.kontak.updateOne(  
 { \_id: ObjectId("...") },  
 { $unset: { temporary\_field: "" } }  
);  
  
// $inc - Increment number  
db.kontak.updateOne(  
 { \_id: ObjectId("...") },  
 { $inc: { login\_count: 1 } }  
);  
  
// $push - Add element to array  
db.kontak.updateOne(  
 { \_id: ObjectId("...") },  
 { $push: { tags: "new\_tag" } }  
);  
  
// $pull - Remove element from array  
db.kontak.updateOne(  
 { \_id: ObjectId("...") },  
 { $pull: { tags: "old\_tag" } }  
);  
  
// $addToSet - Add unique element to array  
db.kontak.updateOne(  
 { \_id: ObjectId("...") },  
 { $addToSet: { tags: "unique\_tag" } }  
);

#### Replace Document

// replaceOne - replaces entire document  
db.kontak.replaceOne(  
 { \_id: ObjectId("635a1b2c3d4e5f6789abc123") },  
 {  
 "nama\_depan": "Ahmad",  
 "nama\_belakang": "Pratama",  
 "email": "ahmad.new@example.com",  
 "grup": "updated",  
 "version": 2  
 }  
);

### Delete Operations

#### Delete Single Document

// deleteOne - deletes first matching document  
db.kontak.deleteOne({ email: "old.email@example.com" });  
  
// Delete by \_id  
db.kontak.deleteOne({ \_id: ObjectId("635a1b2c3d4e5f6789abc123") });

#### Delete Multiple Documents

// deleteMany - deletes all matching documents  
db.kontak.deleteMany({ status: "inactive" });  
  
// Delete all documents (use with caution!)  
db.kontak.deleteMany({});

#### Safe Delete Pattern

// First check what will be deleted  
db.kontak.find({ status: "inactive" });  
  
// Confirm deletion  
var result = db.kontak.deleteMany({ status: "inactive" });  
print("Deleted " + result.deletedCount + " documents");

### Bulk Operations

// Initialize bulk operations  
var bulk = db.kontak.initializeUnorderedBulkOp();  
  
// Add operations  
bulk.insert({ nama\_depan: "Bulk1", email: "bulk1@example.com" });  
bulk.find({ email: "old@example.com" }).updateOne({ $set: { status: "updated" } });  
bulk.find({ status: "temp" }).deleteOne();  
  
// Execute bulk operations  
var result = bulk.execute();  
print("Inserted: " + result.nInserted);  
print("Updated: " + result.nUpserted);  
print("Deleted: " + result.nRemoved);

## Query Lanjutan

Setelah memahami operasi CRUD dasar, kita akan mendalami query yang lebih kompleks untuk mengekstrak insight dari data MongoDB.

### Aggregation Framework

Aggregation framework adalah fitur powerful MongoDB untuk processing dan analisis data.

#### Pipeline Stages

graph LR  
 A[Input Documents] --> B[$match]  
 B --> C[$group]  
 C --> D[$sort]  
 D --> E[$limit]  
 E --> F[$project]  
 F --> G[Output Documents]

#### Basic Aggregation Example

// Contoh: Hitung jumlah kontak per grup  
db.kontak.aggregate([  
 // Stage 1: Filter hanya kontak aktif  
 {  
 $match: {  
 status: "aktif"  
 }  
 },  
 // Stage 2: Group by grup  
 {  
 $group: {  
 \_id: "$grup",  
 total\_kontak: { $sum: 1 },  
 nama\_kontak: { $push: { $concat: ["$nama\_depan", " ", "$nama\_belakang"] } }  
 }  
 },  
 // Stage 3: Sort descending  
 {  
 $sort: {  
 total\_kontak: -1  
 }  
 },  
 // Stage 4: Rename fields  
 {  
 $project: {  
 grup: "$\_id",  
 jumlah: "$total\_kontak",  
 daftar\_nama: "$nama\_kontak",  
 \_id: 0  
 }  
 }  
]);

#### Advanced Aggregation Operations

// Multi-stage aggregation dengan computed fields  
db.kontak.aggregate([  
 // Add computed fields  
 {  
 $addFields: {  
 nama\_lengkap: { $concat: ["$nama\_depan", " ", "$nama\_belakang"] },  
 jumlah\_telepon: { $size: "$telepon" },  
 usia: {  
 $dateDiff: {  
 startDate: "$tanggal\_lahir",  
 endDate: "$$NOW",  
 unit: "year"  
 }  
 },  
 kategori\_usia: {  
 $switch: {  
 branches: [  
 { case: { $lt: ["$usia", 18] }, then: "remaja" },  
 { case: { $lt: ["$usia", 35] }, then: "dewasa" },  
 { case: { $lt: ["$usia", 60] }, then: "paruh baya" }  
 ],  
 default: "lansia"  
 }  
 }  
 }  
 },  
 // Filter berdasarkan kategori  
 {  
 $match: {  
 kategori\_usia: { $in: ["dewasa", "paruh baya"] }  
 }  
 },  
 // Group dengan multiple aggregations  
 {  
 $group: {  
 \_id: {  
 grup: "$grup",  
 kategori\_usia: "$kategori\_usia"  
 },  
 rata\_rata\_usia: { $avg: "$usia" },  
 total\_kontak: { $sum: 1 },  
 kontak\_tertua: {  
 $max: "$nama\_lengkap"  
 },  
 email\_list: { $push: "$email" }  
 }  
 },  
 // Sort hasil  
 {  
 $sort: {  
 "\_id.grup": 1,  
 rata\_rata\_usia: -1  
 }  
 }  
]);

### Lookup (Join Operations)

MongoDB mendukung operasi join antar collections dengan $lookup.

// Join kontak dengan grup  
db.kontak.aggregate([  
 {  
 $lookup: {  
 from: "grup",  
 localField: "grup",  
 foreignField: "nama",  
 as: "info\_grup"  
 }  
 },  
 {  
 $unwind: "$info\_grup" // Flatten array result  
 },  
 {  
 $project: {  
 nama\_lengkap: { $concat: ["$nama\_depan", " ", "$nama\_belakang"] },  
 email: 1,  
 grup\_warna: "$info\_grup.warna",  
 grup\_deskripsi: "$info\_grup.deskripsi"  
 }  
 }  
]);  
  
// Multiple lookup  
db.kontak.aggregate([  
 {  
 $lookup: {  
 from: "grup",  
 localField: "grup",  
 foreignField: "nama",  
 as: "grup\_info"  
 }  
 },  
 {  
 $lookup: {  
 from: "transaksi",  
 localField: "\_id",  
 foreignField: "kontak\_id",  
 as: "transaksi\_history"  
 }  
 },  
 {  
 $project: {  
 nama: { $concat: ["$nama\_depan", " ", "$nama\_belakang"] },  
 email: 1,  
 grup\_nama: { $arrayElemAt: ["$grup\_info.nama", 0] },  
 total\_transaksi: { $size: "$transaksi\_history" }  
 }  
 }  
]);

### Text Search

MongoDB memiliki built-in full-text search capability.

// Create text index  
db.kontak.createIndex({  
 nama\_depan: "text",  
 nama\_belakang: "text",  
 catatan: "text"  
});  
  
// Text search  
db.kontak.find({  
 $text: { $search: "ahmad developer" }  
});  
  
// Text search dengan score  
db.kontak.find(  
 { $text: { $search: "developer programming" } },  
 { score: { $meta: "textScore" } }  
).sort({ score: { $meta: "textScore" } });  
  
// Advanced text search  
db.kontak.find({  
 $text: {  
 $search: "developer -designer", // Exclude designer  
 $caseSensitive: false,  
 $diacriticSensitive: false  
 }  
});

### Geospatial Queries

Untuk aplikasi yang membutuhkan lokasi geografis:

// Create 2dsphere index  
db.kontak.createIndex({  
 "alamat.location": "2dsphere"  
});  
  
// Insert dengan GeoJSON  
db.kontak.insertOne({  
 nama: "Jakarta Office",  
 "alamat": {  
 location: {  
 type: "Point",  
 coordinates: [106.8196, -6.2088] // [longitude, latitude]  
 }  
 }  
});  
  
// Find nearby locations  
db.kontak.find({  
 "alamat.location": {  
 $near: {  
 $geometry: {  
 type: "Point",  
 coordinates: [106.8227, -6.1751] // Jakarta coordinates  
 },  
 $maxDistance: 5000 // 5km radius  
 }  
 }  
});  
  
// GeoWithin for area search  
db.kontak.find({  
 "alamat.location": {  
 $geoWithin: {  
 $geometry: {  
 type: "Polygon",  
 coordinates: [[  
 [106.8, -6.2],  
 [106.9, -6.2],  
 [106.9, -6.1],  
 [106.8, -6.1],  
 [106.8, -6.2]  
 ]]  
 }  
 }  
 }  
});

### Performance Optimization

// Explain query performance  
db.kontak.find({ grup: "teman" }).explain("executionStats");  
  
// Use covered queries  
db.kontak.createIndex({ grup: 1, email: 1 });  
db.kontak.find(  
 { grup: "teman" },  
 { email: 1, \_id: 0 } // Only indexed fields  
);  
  
// Use hint for index selection  
db.kontak.find({ grup: "teman" }).hint({ grup: 1 });  
  
// Limit and skip for pagination  
var page = 1;  
var pageSize = 10;  
db.kontak.find()  
 .sort({ dibuat\_pada: -1 })  
 .skip((page - 1) \* pageSize)  
 .limit(pageSize);

### Complex Query Patterns

// Find duplicates  
db.kontak.aggregate([  
 {  
 $group: {  
 \_id: "$email",  
 count: { $sum: 1 },  
 docs: { $push: "$\_id" }  
 }  
 },  
 {  
 $match: { count: { $gt: 1 } }  
 }  
]);  
  
// Conditional aggregation  
db.kontak.aggregate([  
 {  
 $group: {  
 \_id: "$grup",  
 kontak\_dengan\_foto: {  
 $sum: {  
 $cond: [  
 { $ifNull: ["$foto", false] },  
 1,  
 0  
 ]  
 }  
 },  
 total\_kontak: { $sum: 1 }  
 }  
 },  
 {  
 $addFields: {  
 persentase\_berfoto: {  
 $multiply: [  
 100,  
 { $divide: ["$kontak\_dengan\_foto", "$total\_kontak"] }  
 ]  
 }  
 }  
 }  
]);  
  
// Time-based analysis  
db.kontak.aggregate([  
 {  
 $group: {  
 \_id: {  
 year: { $year: "$dibuat\_pada" },  
 month: { $month: "$dibuat\_pada" }  
 },  
 total\_registrasi: { $sum: 1 },  
 grup\_populer: {  
 $push: { $arrayElemAt: ["$grup", 0] }  
 }  
 }  
 },  
 {  
 $sort: { "\_id.year": -1, "\_id.month": -1 }  
 }  
]);

## Indexing dan Optimasi

Indexing adalah salah satu fitur paling penting untuk optimasi performa query MongoDB. Index memungkinkan database untuk menemukan dokumen dengan cepat tanpa melakukan scan seluruh collection.

### Konsep Dasar Indexing

#### Bagaimana Index Bekerja

graph TD  
 A[Query] --> B{Index Exists?}  
 B -->|Yes| C[Use Index]  
 B -->|No| D[Collection Scan]  
 C --> E[Fast Result]  
 D --> F[Slow Result]

#### Jenis-Jenis Index

// 1. Single Field Index  
db.kontak.createIndex({ email: 1 });  
  
// 2. Compound Index  
db.kontak.createIndex({ grup: 1, nama\_depan: 1 });  
  
// 3. Multikey Index (for arrays)  
db.kontak.createIndex({ tags: 1 });  
  
// 4. Text Index  
db.kontak.createIndex({   
 nama\_depan: "text",   
 nama\_belakang: "text"   
});  
  
// 5. Hashed Index  
db.kontak.createIndex({ email: "hashed" });  
  
// 6. Geospatial Index  
db.kontak.createIndex({ "alamat.location": "2dsphere" });  
  
// 7. TTL Index (Time-To-Live)  
db.sessions.createIndex({ created\_at: 1 }, { expireAfterSeconds: 3600 });

### Index Strategy

#### Menentukan Field untuk Di-index

**Candidate fields untuk indexing:**

1. **Fields yang sering di-query**
2. **Fields untuk sorting**
3. **Fields untuk range queries**
4. **Fields yang unik (untuk unique index)**
5. **Array fields untuk array queries**

#### Compound Index Best Practices

// Rule of thumb: ESR (Equality, Sort, Range)  
db.kontak.createIndex({   
 status: 1, // Equality field  
 grup: 1, // Another equality  
 dibuat\_pada: -1 // Range field (sorting)  
});  
  
// Query yang optimal untuk index di atas:  
db.kontak.find({  
 status: "aktif",  
 grup: "teman"  
}).sort({ dibuat\_pada: -1 });  
  
// Index prefix rules:  
// Index: { a: 1, b: 1, c: 1 }  
// Queries yang bisa menggunakan index:  
// { a: value }  
// { a: value, b: value }  
// { a: value, b: value, c: value }  
// Query yang TIDAK bisa menggunakan index:  
// { b: value }  
// { b: value, c: value }

### Index Management

#### View Existing Indexes

// List all indexes on collection  
db.kontak.getIndexes();  
  
// Get index stats  
db.kontak.getIndexStats();  
  
// Detailed index information  
db.kontak.getIndexes().forEach(function(index) {  
 print("Index: " + JSON.stringify(index.key));  
 print("Size: " + index.size);  
 print("Usage: " + index.accesses.ops);  
});

#### Drop Indexes

// Drop specific index  
db.kontak.dropIndex({ email: 1 });  
  
// Drop index by name  
db.kontak.dropIndex("email\_1");  
  
// Drop all indexes except \_id  
db.kontak.dropIndexes();

### Special Index Types

#### Unique Index

// Unique index untuk mencegah duplikasi  
db.kontak.createIndex({ email: 1 }, { unique: true });  
  
// Compound unique index  
db.kontak.createIndex({   
 nama\_depan: 1,   
 nama\_belakang: 1   
}, { unique: true });  
  
// Partial unique index  
db.kontak.createIndex(  
 { email: 1 },  
 {   
 unique: true,  
 partialFilterExpression: { status: "aktif" }  
 }  
);

#### Sparse Index

// Index hanya untuk dokumen yang memiliki field tersebut  
db.kontak.createIndex({ fax: 1 }, { sparse: true });  
  
// Sparse vs Regular Index:  
// Regular: null untuk missing values  
// Sparse: tidak index dokumen tanpa field

#### TTL Index

// Auto-expire documents after certain time  
db.sessions.createIndex(  
 { created\_at: 1 },  
 { expireAfterSeconds: 3600 } // 1 hour  
);  
  
// Capped collection dengan TTL  
db.logs.createIndex(  
 { timestamp: 1 },  
 { expireAfterSeconds: 86400 } // 24 hours  
);

### Query Optimization

#### Explain Plan

// Basic explain  
db.kontak.find({ email: "test@example.com" }).explain();  
  
// Detailed execution stats  
db.kontak.find({ grup: "teman" }).explain("executionStats");  
  
// All execution plans  
db.kontak.find({ tags: "javascript" }).explain("allPlansExecution");

#### Reading Explain Output

// Check if index is used  
var explain = db.kontak.find({ grup: "teman" }).explain("executionStats");  
print("Total docs examined: " + explain.executionStats.totalDocsExamined);  
print("Total docs returned: " + explain.executionStats.totalDocsReturned);  
print("Execution time (ms): " + explain.executionStats.executionTimeMillis);  
  
// Index usage ratio  
var ratio = explain.executionStats.totalDocsReturned /   
 explain.executionStats.totalDocsExamined;  
print("Efficiency ratio: " + ratio);  
// Ratio close to 1 = efficient query

#### Covered Queries

// Create covering index  
db.kontak.createIndex({   
 grup: 1,   
 email: 1,   
 nama\_depan: 1   
});  
  
// Query that uses covered index  
db.kontak.find(  
 { grup: "teman" },  
 { email: 1, nama\_depan: 1, \_id: 0 } // Only indexed fields  
);  
  
// Verify covered query  
db.kontak.find(  
 { grup: "teman" },  
 { email: 1, nama\_depan: 1, \_id: 0 }  
).explain("executionStats").executionStats.totalDocsExamined ===   
db.kontak.find(  
 { grup: "teman" },  
 { email: 1, nama\_depan: 1, \_id: 0 }  
).explain("executionStats").executionStats.totalDocsReturned;

### Performance Monitoring

#### Index Usage Statistics

// Monitor index usage over time  
db.runCommand({  
 collStats: "kontak",  
 indexDetails: true  
});  
  
// Check unused indexes  
db.kontak.aggregate([  
 { $indexStats: {} },  
 {   
 $project: {  
 name: "$name",  
 usage: "$accesses.ops",  
 lastUsed: "$accesses.since"  
 }  
 },  
 { $sort: { usage: -1 } }  
]);

#### Slow Query Analysis

// Enable profiler (use with caution in production)  
db.setProfilingLevel(2); // Log all operations  
db.setProfilingLevel(1, { slowms: 100 }); // Log operations > 100ms  
  
// View slow queries  
db.system.profile.find().sort({ ts: -1 }).limit(5);  
  
// Get profiler status  
db.getProfilingStatus();

### Index Maintenance

#### Rebuild Indexes

// Rebuild all indexes  
db.kontak.reIndex();  
  
// Check index fragmentation  
db.kontak.aggregate([ { $indexStats: {} } ]);  
  
// Compact collection (requires exclusive lock)  
db.runCommand({ compact: "kontak" });

#### Index Size Management

// Check index size  
db.kontak.totalIndexSize();  
  
// Detailed size info  
db.kontak.stats().indexSizes;  
  
// Reduce index size with selective indexing  
db.kontak.createIndex(  
 { tags: 1 },  
 {   
 sparse: true,  
 partialFilterExpression: {   
 tags: { $exists: true, $ne: [] }   
 }  
 }  
);

## Backup dan Restore

Backup dan restore adalah operasi krusial dalam manajemen database untuk melindungi data dari kehilangan dan corruption.

### Backup Methods

#### 1. Mongodump

Mongodump adalah tool untuk membuat backup binary dari MongoDB database.

# Backup single database  
mongodump --host localhost:27017 --db kontak\_manager  
  
# Backup dengan authentication  
mongodump --host localhost:27017 \  
 --username admin \  
 --password password123 \  
 --authenticationDatabase admin \  
 --db kontak\_manager  
  
# Backup specific collections  
mongodump --host localhost:27017 \  
 --db kontak\_manager \  
 --collection kontak \  
 --collection grup  
  
# Backup dengan compression  
mongodump --host localhost:27017 \  
 --db kontak\_manager \  
 --gzip \  
 --out /backup/$(date +%Y%m%d)  
  
# Backup dengan query filter  
mongodump --host localhost:27017 \  
 --db kontak\_manager \  
 --collection kontak \  
 --query '{"status": "aktif"}'

#### 2. Mongoexport

Mongoexport untuk export data dalam format JSON, CSV, atau TSV.

# Export collection ke JSON  
mongoexport --host localhost:27017 \  
 --db kontak\_manager \  
 --collection kontak \  
 --out kontak.json  
  
# Export ke CSV  
mongoexport --host localhost:27017 \  
 --db kontak\_manager \  
 --collection kontak \  
 --type=csv \  
 --fields=nama\_depan,nama\_belakang,email,grup,status \  
 --out kontak.csv  
  
# Export dengan query  
mongoexport --host localhost:27017 \  
 --db kontak\_manager \  
 --collection kontak \  
 --query '{"grup": "teman"}' \  
 --out kontak\_teman.json  
  
# Export array ke CSV  
mongoexport --host localhost:27017 \  
 --db kontak\_manager \  
 --collection kontak \  
 --type=csv \  
 --fields=nama\_depan,email,grup \  
 --out kontak.csv

### Automated Backup Scripts

#### Bash Script untuk Daily Backup

#!/bin/bash  
# backup\_mongodb.sh  
  
BACKUP\_DIR="/backup/mongodb"  
DATE=$(date +%Y%m%d\_%H%M%S)  
DB\_NAME="kontak\_manager"  
RETENTION\_DAYS=7  
  
# Create backup directory  
mkdir -p $BACKUP\_DIR  
  
# Backup dengan mongodump  
mongodump \  
 --host localhost:27017 \  
 --username admin \  
 --password password123 \  
 --authenticationDatabase admin \  
 --db $DB\_NAME \  
 --gzip \  
 --out $BACKUP\_DIR/backup\_$DATE  
  
# Compress backup folder  
tar -czf $BACKUP\_DIR/backup\_$DATE.tar.gz -C $BACKUP\_DIR backup\_$DATE  
rm -rf $BACKUP\_DIR/backup\_$DATE  
  
# Delete old backups  
find $BACKUP\_DIR -name "backup\_\*.tar.gz" -mtime +$RETENTION\_DAYS -delete  
  
echo "Backup completed: backup\_$DATE.tar.gz"

#### Docker Backup Script

#!/bin/bash  
# mongodb\_docker\_backup.sh  
  
CONTAINER\_NAME="mongodb-server"  
BACKUP\_DIR="/backup/mongodb"  
DATE=$(date +%Y%m%d\_%H%M%S)  
  
# Create backup directory  
mkdir -p $BACKUP\_DIR  
  
# Backup from Docker container  
docker exec $CONTAINER\_NAME mongodump \  
 --db kontak\_manager \  
 --gzip \  
 --out /backup/backup\_$DATE  
  
# Copy backup from container  
docker cp $CONTAINER\_NAME:/backup/backup\_$DATE $BACKUP\_DIR/  
  
# Compress  
tar -czf $BACKUP\_DIR/backup\_$DATE.tar.gz -C $BACKUP\_DIR backup\_$DATE  
rm -rf $BACKUP\_DIR/backup\_$DATE  
  
echo "Docker backup completed: backup\_$DATE.tar.gz"

### Restore Operations

#### 1. Mongorestore

# Restore entire database  
mongorestore --host localhost:27017 \  
 --db kontak\_manager \  
 /backup/20231030/backup\_20231030/kontak\_manager  
  
# Restore dengan overwrite  
mongorestore --host localhost:27017 \  
 --db kontak\_manager \  
 --drop \  
 /backup/20231030/backup\_20231030/kontak\_manager  
  
# Restore specific collection  
mongorestore --host localhost:27017 \  
 --db kontak\_manager \  
 --collection kontak \  
 /backup/20231030/backup\_20231030/kontak\_manager/kontak.bson  
  
# Restore dari compressed backup  
mongorestore --host localhost:27017 \  
 --db kontak\_manager \  
 --gzip \  
 /backup/20231030/backup\_20231030.tar.gz  
  
# Restore dengan authentication  
mongorestore --host localhost:27017 \  
 --username admin \  
 --password password123 \  
 --authenticationDatabase admin \  
 --db kontak\_manager \  
 /backup/20231030/backup\_20231030/kontak\_manager

#### 2. Mongoimport

# Import dari JSON  
mongoimport --host localhost:27017 \  
 --db kontak\_manager \  
 --collection kontak \  
 --file kontak.json  
  
# Import dari CSV  
mongoimport --host localhost:27017 \  
 --db kontak\_manager \  
 --collection kontak \  
 --type=csv \  
 --headerline \  
 --file kontak.csv  
  
# Import dengan mode upsert  
mongoimport --host localhost:27017 \  
 --db kontak\_manager \  
 --collection kontak \  
 --file kontak.json \  
 --mode=upsert \  
 --upsertFields=email  
  
# Import ke collection baru  
mongoimport --host localhost:27017 \  
 --db kontak\_manager \  
 --collection kontak\_backup \  
 --file kontak.json

### Point-in-Time Recovery

#### Oplog untuk Point-in-Time Recovery

// Enable oplog (replica set required)  
// Check oplog size  
db.getReplicationInfo();  
db.printReplicationInfo();  
  
// Query oplog  
use local;  
db.oplog.rs.find().sort({ ts: -1 }).limit(5);  
  
// Restore to specific timestamp  
mongorestore --host localhost:27017 \  
 --db kontak\_manager \  
 --oplogReplay \  
 --oplogLimit 1672444800:1 \  
 /backup/base\_backup/

### Backup Strategies

#### 1. Full Backup Strategy

# Weekly full backup  
#!/bin/bash  
# weekly\_full\_backup.sh  
  
BACKUP\_DIR="/backup/mongodb/full"  
DATE=$(date +%Y%m%d)  
  
mongodump --host localhost:27017 \  
 --db kontak\_manager \  
 --gzip \  
 --out $BACKUP\_DIR/full\_$DATE  
  
# Keep 4 weeks of full backups  
find $BACKUP\_DIR -name "full\_\*" -mtime +28 -delete

#### 2. Incremental Backup Strategy

# Daily incremental backup menggunakan oplog  
#!/bin/bash  
# incremental\_backup.sh  
  
BACKUP\_DIR="/backup/mongodb/incremental"  
DATE=$(date +%Y%m%d)  
LAST\_BACKUP=$(cat $BACKUP\_DIR/last\_backup.txt)  
  
# Extract oplog since last backup  
mongodump --host localhost:27017 \  
 --db local \  
 --collection oplog.rs \  
 --query '{"ts":{"$gt":{"$timestamp":{"t":'$LAST\_BACKUP',"i":1}}}}' \  
 --out $BACKUP\_DIR/inc\_$DATE  
  
# Update last backup timestamp  
date +%s > $BACKUP\_DIR/last\_backup.txt

#### 3. Cloud Backup Strategy

# Backup ke AWS S3  
#!/bin/bash  
# s3\_backup.sh  
  
S3\_BUCKET="mongodb-backups-bucket"  
LOCAL\_BACKUP="/backup/mongodb"  
DATE=$(date +%Y%m%d)  
  
# Create backup  
mongodump --host localhost:27017 \  
 --db kontak\_manager \  
 --gzip \  
 --out $LOCAL\_BACKUP/backup\_$DATE  
  
# Upload to S3  
aws s3 cp $LOCAL\_BACKUP/backup\_$DATE.tar.gz \  
 s3://$S3\_BUCKET/mongodb/backup\_$DATE.tar.gz  
  
# Clean local backup  
rm -rf $LOCAL\_BACKUP/backup\_$DATE\*

### Verification and Testing

#### Backup Verification

#!/bin/bash  
# verify\_backup.sh  
  
BACKUP\_FILE=$1  
TEST\_DB="kontak\_manager\_test"  
  
# Create test database  
mongorestore --host localhost:27017 \  
 --db $TEST\_DB \  
 --drop \  
 $BACKUP\_FILE  
  
# Verify data integrity  
mongo --host localhost:27017 --eval "  
db = db.getSiblingDB('$TEST\_DB');  
var originalCount = db.getSiblingDB('kontak\_manager').kontak.count();  
var restoredCount = db.kontak.count();  
print('Original count: ' + originalCount);  
print('Restored count: ' + restoredCount);  
print('Verification: ' + (originalCount === restoredCount ? 'PASSED' : 'FAILED'));  
db.dropDatabase();  
"

#### Disaster Recovery Testing

#!/bin/bash  
# disaster\_recovery\_test.sh  
  
# 1. Simulate data corruption  
mongo kontak\_manager --eval "  
db.kontak.updateMany({}, {\$set: {corrupted: true}});  
"  
  
# 2. Restore from latest backup  
LATEST\_BACKUP=$(ls -t /backup/mongodb/backup\_\*.tar.gz | head -1)  
tar -xzf $LATEST\_BACKUP  
mongorestore --host localhost:27017 \  
 --db kontak\_manager \  
 --drop \  
 backup\_\*/kontak\_manager  
  
# 3. Verify recovery  
mongo kontak\_manager --eval "  
var count = db.kontak.count({corrupted: {\$exists: false}});  
print('Recovered documents: ' + count);  
"

### Monitoring Backup Health

#### Backup Health Check Script

#!/bin/bash  
# backup\_health\_check.sh  
  
BACKUP\_DIR="/backup/mongodb"  
ALERT\_EMAIL="admin@example.com"  
MAX\_AGE\_HOURS=24  
  
# Check latest backup  
LATEST\_BACKUP=$(ls -t $BACKUP\_DIR/backup\_\*.tar.gz | head -1)  
BACKUP\_AGE=$(($(date +%s) - $(stat -c %Y $LATEST\_BACKUP)))  
BACKUP\_AGE\_HOURS=$((BACKUP\_AGE / 3600))  
  
if [ $BACKUP\_AGE\_HOURS -gt $MAX\_AGE\_HOURS ]; then  
 echo "ALERT: Latest backup is $BACKUP\_AGE\_HOURS hours old!" | \  
 mail -s "MongoDB Backup Alert" $ALERT\_EMAIL  
fi  
  
# Check backup file integrity  
if ! tar -tzf $LATEST\_BACKUP > /dev/null; then  
 echo "ALERT: Backup file is corrupted!" | \  
 mail -s "MongoDB Backup Alert" $ALERT\_EMAIL  
fi  
  
echo "Backup health check completed"

## Troubleshooting

Dalam praktik pengembangan dengan MongoDB, berbagai masalah dapat terjadi. Bagian ini membahas troubleshooting umum dan solusinya.

### Connection Issues

#### Problem: Cannot Connect to MongoDB

**Symptoms:**

* Connection timeout error
* “Cannot connect to MongoDB” message
* Docker container not accessible

**Solutions:**

# 1. Check if MongoDB container is running  
docker ps | grep mongo  
  
# 2. Check container logs  
docker logs mongodb-container  
  
# 3. Restart MongoDB container  
docker restart mongodb-container  
  
# 4. Check port availability  
netstat -tlnp | grep 27017  
  
# 5. Test connection with telnet  
telnet localhost 27017  
  
# 6. Check Docker network  
docker network ls  
docker network inspect bridge

#### Problem: Authentication Failed

**Symptoms:**

* “Authentication failed” error
* Cannot login with credentials

**Solutions:**

// 1. Check if users exist  
use admin  
db.getUsers();  
  
// 2. Create admin user if not exists  
use admin  
db.createUser({  
 user: "admin",  
 pwd: "password123",  
 roles: ["userAdminAnyDatabase", "dbAdminAnyDatabase"]  
});  
  
// 3. Check authentication database  
mongo mongodb://admin:password123@localhost:27017/?authSource=admin

### Performance Issues

#### Problem: Slow Queries

**Diagnosis:**

// 1. Enable profiler  
db.setProfilingLevel(2);  
  
// 2. Check slow queries  
db.system.profile.find().sort({millis: -1}).limit(5);  
  
// 3. Explain slow query  
db.kontak.find({grup: "teman"}).explain("executionStats");  
  
// 4. Check index usage  
db.kontak.getIndexes();  
db.kontak.getIndexStats();

**Solutions:**

// 1. Create appropriate indexes  
db.kontak.createIndex({grup: 1, status: 1});  
  
// 2. Use covered queries  
db.kontak.find({grup: "teman"}, {email: 1, \_id: 0});  
  
// 3. Optimize aggregation pipeline  
db.kontak.aggregate([  
 {$match: {status: "aktif"}}, // Early filtering  
 {$group: {\_id: "$grup", count: {$sum: 1}}},  
 {$sort: {count: -1}}  
]);

#### Problem: High Memory Usage

**Diagnosis:**

// Check memory usage  
db.serverStatus().mem;  
  
// Check working set  
db.serverStatus().wiredTiger.cache;  
  
// Check connection stats  
db.serverStatus().connections;

**Solutions:**

# 1. Limit connections in docker-compose.yml  
environment:  
 - MONGO\_INITDB\_ROOT\_USERNAME=admin  
 - MONGO\_INITDB\_ROOT\_PASSWORD=password123  
 - WIRED\_TIGER\_CACHE\_SIZE\_GB=1  
  
# 2. Monitor with docker stats  
docker stats mongodb-container  
  
# 3. Configure max connections  
command: mongod --maxConns 1000

### Data Issues

#### Problem: Duplicate Documents

**Detection:**

// Find duplicate emails  
db.kontak.aggregate([  
 {$group: {\_id: "$email", count: {$sum: 1}, docs: {$push: "$\_id"}}},  
 {$match: {count: {$gt: 1}}}  
]);

**Solution:**

// Remove duplicates keeping latest  
db.kontak.aggregate([  
 {$sort: {dibuat\_pada: -1}},  
 {$group: {  
 \_id: "$email",  
 latest: {$first: "$$ROOT"},  
 dups: {$push: "$$ROOT"}  
 }},  
 {$match: {"dups.1": {$exists: true}}},  
 {$replaceRoot: {newRoot: "$latest"}}  
]).forEach(function(doc) {  
 db.kontak.deleteMany({\_id: {$ne: doc.\_id}, email: doc.email});  
});

#### Problem: Schema Inconsistency

**Detection:**

// Find documents missing required fields  
db.kontak.find({email: {$exists: false}});  
  
// Find documents with wrong field types  
db.kontak.find({umur: {$not: {$type: "number"}}});

**Solution:**

// Add missing fields with default values  
db.kontak.updateMany(  
 {status: {$exists: false}},  
 {$set: {status: "aktif"}}  
);  
  
// Fix type issues  
db.kontak.find({umur: {$type: "string"}}).forEach(function(doc) {  
 db.kontak.updateOne(  
 {\_id: doc.\_id},  
 {$set: {umur: parseInt(doc.umur)}}  
 );  
});

### Docker Issues

#### Problem: Container Keeps Restarting

**Diagnosis:**

# Check container status  
docker ps -a | grep mongo  
  
# Check logs for errors  
docker logs mongodb-container  
  
# Inspect container  
docker inspect mongodb-container

**Common Solutions:**

# 1. Fix volume permissions in docker-compose.yml  
volumes:  
 - mongodb\_data:/data/db  
 - ./init-scripts:/docker-entrypoint-initdb.d:ro  
  
# 2. Add health check  
healthcheck:  
 test: ["CMD", "mongosh", "--eval", "db.adminCommand('ping')"]  
 interval: 30s  
 timeout: 10s  
 retries: 3

#### Problem: Data Persistence Issues

**Diagnosis:**

# Check volume mounts  
docker volume ls  
docker volume inspect mongodb\_mongodb\_data  
  
# Check data directory  
docker exec mongodb-container ls -la /data/db

**Solution:**

# Ensure proper volume configuration  
volumes:  
 mongodb\_data:  
 driver: local  
  
services:  
 mongodb:  
 volumes:  
 - mongodb\_data:/data/db  
 - ./backup:/backup

### Mongo Express Issues

#### Problem: Cannot Access Mongo Express

**Solutions:**

# 1. Check if container is running  
docker ps | grep mongo-express  
  
# 2. Check port conflict  
netstat -tlnp | grep 8081  
  
# 3. Verify network connection  
docker network ls  
docker network connect mongodb\_default mongo-express-container  
  
# 4. Check environment variables  
docker logs mongo-express-ui

#### Problem: Mongo Express Shows “Cannot GET /”

**Solution:**

# Fix ME\_CONFIG\_MONGODB\_URL format  
environment:  
 ME\_CONFIG\_MONGODB\_URL: "mongodb://admin:password123@mongodb:27017/"  
 ME\_CONFIG\_MONGODB\_ENABLE\_ADMIN: "true"

### Common Error Messages

#### “Operation time out”

// Solution: Increase timeout or optimize query  
db.adminCommand({  
 setParameter: 1,  
 internalQueryExecMaxBlockingSortBytes: 104857600  
});

#### “Document not found”

// Check if document exists before update  
var doc = db.kontak.findOne({\_id: ObjectId("...")});  
if (doc) {  
 db.kontak.updateOne({\_id: doc.\_id}, {$set: {field: "value"}});  
}

#### “Write conflict”

// Use retry logic  
var attempts = 0;  
var maxAttempts = 3;  
  
while (attempts < maxAttempts) {  
 try {  
 db.kontak.updateOne({\_id: id}, {$set: {field: value}});  
 break;  
 } catch (e) {  
 attempts++;  
 sleep(100); // Wait 100ms  
 }  
}

### Monitoring and Diagnostics

#### Health Check Script

#!/bin/bash  
# mongodb\_health\_check.sh  
  
CONTAINER="mongodb-server"  
PORT=27017  
  
# Check container status  
if ! docker ps | grep -q $CONTAINER; then  
 echo "ERROR: MongoDB container is not running"  
 exit 1  
fi  
  
# Check port connectivity  
if ! nc -z localhost $PORT; then  
 echo "ERROR: Cannot connect to MongoDB port $PORT"  
 exit 1  
fi  
  
# Check database connectivity  
if ! docker exec $CONTAINER mongosh --eval "db.adminCommand('ping')" > /dev/null; then  
 echo "ERROR: MongoDB not responding to ping"  
 exit 1  
fi  
  
# Check disk space  
DISK\_USAGE=$(docker exec $CONTAINER df /data/db | tail -1 | awk '{print $5}' | sed 's/%//')  
if [ $DISK\_USAGE -gt 80 ]; then  
 echo "WARNING: Disk usage is ${DISK\_USAGE}%"  
fi  
  
echo "MongoDB health check passed"

#### Performance Monitoring

// Create monitoring script in MongoDB  
var monitoring = {  
 checkConnections: function() {  
 var stats = db.serverStatus().connections;  
 print("Current connections: " + stats.current);  
 print("Available connections: " + stats.available);  
 if (stats.current / stats.available > 0.8) {  
 print("WARNING: High connection usage");  
 }  
 },  
   
 checkOperations: function() {  
 var stats = db.serverStatus().opcounters;  
 print("Operations since start:");  
 print(" Insert: " + stats.insert);  
 print(" Query: " + stats.query);  
 print(" Update: " + stats.update);  
 print(" Delete: " + stats.delete);  
 },  
   
 checkMemory: function() {  
 var mem = db.serverStatus().mem;  
 print("Memory usage:");  
 print(" Resident: " + (mem.resident / 1024) + " MB");  
 print(" Virtual: " + (mem.virtual / 1024) + " MB");  
 print(" Mapped: " + (mem.mapped / 1024) + " MB");  
 }  
};  
  
// Run monitoring  
monitoring.checkConnections();  
monitoring.checkOperations();  
monitoring.checkMemory();

## Latihan Tambahan

Setelah menyelesaikan praktik dasar, lanjutkan dengan latihan berikut untuk memperdalam pemahaman MongoDB.

### Latihan 1: Advanced Schema Design

**Tujuan:** Mendesain schema yang optimal untuk kasus kompleks

**Scenario:** Buat sistem e-commerce sederhana dengan collections:

1. **products** - Produk dengan variasi dan review
2. **orders** - Order dengan multiple items
3. **users** - User dengan shopping cart dan wishlist

**Task:**

// 1. Buat collection products dengan embedded reviews  
db.products.insertOne({  
 name: "Laptop XYZ",  
 category: "electronics",  
 price: 15000000,  
 specifications: {  
 brand: "XYZ Corp",  
 cpu: "Intel i7",  
 ram: "16GB",  
 storage: "512GB SSD"  
 },  
 variations: [  
 {  
 color: "black",  
 stock: 10,  
 sku: "XYZ-LAP-BLK"  
 },  
 {  
 color: "silver",   
 stock: 5,  
 sku: "XYZ-LAP-SLV"  
 }  
 ],  
 reviews: [  
 {  
 user\_id: ObjectId("..."),  
 rating: 5,  
 comment: "Excellent laptop!",  
 date: new Date("2023-10-01")  
 }  
 ],  
 tags: ["laptop", "gaming", "professional"],  
 created\_at: new Date()  
});  
  
// 2. Buat aggregation untuk hitung rata-rata rating per produk  
db.products.aggregate([  
 {  
 $addFields: {  
 avg\_rating: { $avg: "$reviews.rating" },  
 total\_reviews: { $size: "$reviews" }  
 }  
 },  
 {  
 $project: {  
 name: 1,  
 price: 1,  
 avg\_rating: 1,  
 total\_reviews: 1  
 }  
 }  
]);  
  
// 3. Query produk dengan variasi tertentu  
db.products.find({  
 "variations.color": "black",  
 "variations.stock": { $gt: 0 }  
});

### Latihan 2: Performance Optimization

**Tujuan:** Mengoptimasi performa query dengan indexing

**Task:**

// 1. Identifikasi slow queries  
db.products.find({   
 category: "electronics",  
 "specifications.brand": "XYZ Corp"  
}).explain("executionStats");  
  
// 2. Buat compound index yang optimal  
db.products.createIndex({  
 category: 1,  
 "specifications.brand": 1,  
 price: -1  
});  
  
// 3. Test performa dengan explain  
db.products.find({  
 category: "electronics",  
 "specifications.brand": "XYZ Corp"  
}).sort({ price: -1 }).explain("executionStats");  
  
// 4. Implementasi pagination yang efisien  
function getProductsPage(pageNumber, pageSize) {  
 var skip = (pageNumber - 1) \* pageSize;  
 return db.products.find()  
 .sort({ created\_at: -1 })  
 .skip(skip)  
 .limit(pageSize)  
 .toArray();  
}  
  
// 5. Implementasi search dengan text index  
db.products.createIndex({  
 name: "text",  
 "specifications.cpu": "text",  
 tags: "text"  
});  
  
db.products.find({  
 $text: { $search: "laptop gaming" }  
}, { score: { $meta: "textScore" } })  
.sort({ score: { $meta: "textScore" } });

### Latihan 3: Data Validation and Security

**Tujuan:** Implementasi schema validation dan security best practices

**Task:**

// 1. Buat schema validation untuk products  
db.runCommand({  
 collMod: "products",  
 validator: {  
 $jsonSchema: {  
 bsonType: "object",  
 required: ["name", "category", "price"],  
 properties: {  
 name: {  
 bsonType: "string",  
 minLength: 3,  
 maxLength: 100  
 },  
 price: {  
 bsonType: "number",  
 minimum: 0  
 },  
 category: {  
 enum: ["electronics", "clothing", "books", "home"]  
 }  
 }  
 }  
 }  
});  
  
// 2. Test validation  
try {  
 db.products.insertOne({  
 name: "Invalid Product",  
 category: "invalid\_category",  
 price: -100  
 });  
} catch (e) {  
 print("Validation error: " + e.message);  
}  
  
// 3. Buat role-based access control  
use admin  
db.createRole({  
 role: "productManager",  
 privileges: [  
 {  
 resource: { db: "ecommerce", collection: "products" },  
 actions: ["find", "insert", "update"]  
 }  
 ],  
 roles: []  
});  
  
// 4. Create user dengan role tertentu  
db.createUser({  
 user: "product\_manager",  
 pwd: "secure\_password",  
 roles: [  
 { role: "productManager", db: "ecommerce" }  
 ]  
});

### Latihan 4: Advanced Aggregation

**Tujuan:** Master aggregation pipeline untuk analisis data kompleks

**Task:**

// 1. Sales analytics dashboard  
db.orders.aggregate([  
 // Stage 1: Filter date range  
 {  
 $match: {  
 order\_date: {  
 $gte: new Date("2023-01-01"),  
 $lte: new Date("2023-12-31")  
 },  
 status: "completed"  
 }  
 },  
 // Stage 2: Unwind items  
 { $unwind: "$items" },  
 // Stage 3: Lookup product details  
 {  
 $lookup: {  
 from: "products",  
 localField: "items.product\_id",  
 foreignField: "\_id",  
 as: "product"  
 }  
 },  
 // Stage 4: Calculate metrics  
 {  
 $group: {  
 \_id: {  
 month: { $month: "$order\_date" },  
 product\_category: { $arrayElemAt: ["$product.category", 0] }  
 },  
 total\_revenue: {  
 $sum: {  
 $multiply: ["$items.quantity", "$items.price"]  
 }  
 },  
 total\_items\_sold: { $sum: "$items.quantity" },  
 unique\_orders: { $addToSet: "$\_id" }  
 }  
 },  
 // Stage 5: Final calculations  
 {  
 $project: {  
 month: "$\_id.month",  
 category: "$\_id.product\_category",  
 revenue: "$total\_revenue",  
 items\_sold: "$total\_items\_sold",  
 order\_count: { $size: "$unique\_orders" },  
 avg\_order\_value: {  
 $divide: ["$total\_revenue", { $size: "$unique\_orders" }]  
 }  
 }  
 },  
 // Stage 6: Sort results  
 { $sort: { month: 1, revenue: -1 } }  
]);

### Latihan 5: Replication and High Availability

**Tujuan:** Memahami konsep replica set untuk high availability

**Task:**

# 1. Setup replica set dengan Docker Compose  
# docker-compose.replica.yml  
version: '3.8'  
services:  
 mongo-primary:  
 image: mongo:6.0  
 container\_name: mongo-primary  
 command: mongod --replSet rs0 --bind\_ip\_all  
 ports:  
 - "27017:27017"  
   
 mongo-secondary1:  
 image: mongo:6.0  
 container\_name: mongo-secondary1  
 command: mongod --replSet rs0 --bind\_ip\_all  
 ports:  
 - "27018:27017"  
   
 mongo-secondary2:  
 image: mongo:6.0  
 container\_name: mongo-secondary2  
 command: mongod --replSet rs0 --bind\_ip\_all  
 ports:  
 - "27019:27017"

// 2. Initialize replica set  
rs.initiate({  
 \_id: "rs0",  
 members: [  
 { \_id: 0, host: "mongo-primary:27017", priority: 2 },  
 { \_id: 1, host: "mongo-secondary1:27017", priority: 1 },  
 { \_id: 2, host: "mongo-secondary2:27017", priority: 1 }  
 ]  
});  
  
// 3. Check replica set status  
rs.status();  
rs.conf();  
  
// 4. Test failover  
// Step down primary  
rs.stepDown();  
  
// 5. Configure read preferences  
db.collection.find().readPref("secondaryPreferred");

### Latihan 6: Sharding for Scalability

**Tujuan:** Memahami horizontal scaling dengan sharding

**Task:**

// 1. Enable sharding for database  
sh.enableSharding("ecommerce");  
  
// 2. Shard collection based on user\_id  
sh.shardCollection("ecommerce.orders", { user\_id: 1 });  
  
// 3. Check sharding status  
sh.status();  
  
// 4. Monitor shard distribution  
db.runCommand({ listCollections: 1 });  
db.runCommand({ shardConnPoolStats: 1 });

### Latihan 7: Change Streams

**Tujuan:** Implementasi real-time data change notifications

**Task:**

// 1. Open change stream on collection  
var changeStream = db.products.watch();  
  
// 2. Listen for changes  
changeStream.on('change', function(change) {  
 print("Change detected:");  
 print(JSON.stringify(change, null, 2));  
   
 // Process different operation types  
 if (change.operationType === 'insert') {  
 print("New product added: " + change.fullDocument.name);  
 } else if (change.operationType === 'update') {  
 print("Product updated: " + change.documentKey.\_id);  
 }  
});  
  
// 3. Filter change stream  
var filteredStream = db.products.watch([  
 { $match: { operationType: 'insert' } }  
]);  
  
// 4. Change stream with full document  
var fullDocStream = db.products.watch(  
 [], // Pipeline  
 { fullDocument: 'updateLookup' }  
);

### Latihan 8: Transactions

**Tujuan:** Implementasi multi-document ACID transactions

**Task:**

// 1. Start transaction  
session = db.getMongo().startSession();  
session.startTransaction();  
  
try {  
 // 2. Update inventory  
 session.getDatabase("ecommerce").products.updateOne(  
 { \_id: ObjectId("..."), "variations.stock": { $gt: 0 } },  
 { $inc: { "variations.$.stock": -1 } }  
 );  
   
 // 3. Create order  
 session.getDatabase("ecommerce").orders.insertOne({  
 user\_id: ObjectId("..."),  
 items: [{  
 product\_id: ObjectId("..."),  
 quantity: 1,  
 price: 15000000  
 }],  
 total: 15000000,  
 status: "pending",  
 created\_at: new Date()  
 });  
   
 // 4. Commit transaction  
 session.commitTransaction();  
 print("Transaction committed successfully");  
   
} catch (error) {  
 // 5. Rollback on error  
 session.abortTransaction();  
 print("Transaction rolled back: " + error);  
   
} finally {  
 session.endSession();  
}  
  
// 6. Retry transaction on write conflict  
function executeTransactionWithRetry(retryCount) {  
 var session = db.getMongo().startSession();  
   
 for (var i = 0; i < retryCount; i++) {  
 try {  
 session.startTransaction();  
   
 // Transaction operations here  
 session.getDatabase("ecommerce").products.updateOne(  
 { \_id: productId },  
 { $inc: { stock: -quantity } }  
 );  
   
 session.commitTransaction();  
 return true;  
   
 } catch (error) {  
 session.abortTransaction();  
 if (i === retryCount - 1) throw error;  
 sleep(100 \* (i + 1)); // Exponential backoff  
 }  
 }  
   
 session.endSession();  
}

## Referensi

Berikut adalah referensi yang digunakan dalam penyusunan dokumen ini:

1. Mari Belajar MongoDB - Product Engineering. (2023, February). Diambil dari <https://waresix.engineering/mongodb-9674028e545d>
2. PRAKTIKUM 10 DATABASE NOSQL. (n.d.). Diambil dari <https://yunia.lecturer.pens.ac.id/Praktikum%20Basis%20Data%20Lanjut/10%20Praktikum%20Database%20NoSQL.pdf>
3. Tugas Praktikum Mongodb | PDF. (n.d.). Diambil dari <https://id.scribd.com/document/730001683/TUGAS-PRAKTIKUM-MONGODB-4>

## Lampiran

### A. Cheatsheet Perintah MongoDB

#### Basic Operations

// Insert  
db.collection.insertOne({field: "value"});  
db.collection.insertMany([{field: "value1"}, {field: "value2"}]);  
  
// Find  
db.collection.find();  
db.collection.findOne({field: "value"});  
db.collection.find({field: {$gt: 10}});  
  
// Update  
db.collection.updateOne({field: "value"}, {$set: {newField: "newValue"}});  
db.collection.updateMany({field: "value"}, {$set: {newField: "newValue"}});  
  
// Delete  
db.collection.deleteOne({field: "value"});  
db.collection.deleteMany({field: "value"});

#### Index Operations

// Create indexes  
db.collection.createIndex({field: 1});  
db.collection.createIndex({field1: 1, field2: -1});  
db.collection.createIndex({field: "text"});  
  
// View indexes  
db.collection.getIndexes();  
db.collection.getIndexStats();  
  
// Drop indexes  
db.collection.dropIndex({field: 1});  
db.collection.dropIndexes();

#### Aggregation

// Basic aggregation  
db.collection.aggregate([  
 {$match: {field: "value"}},  
 {$group: {\_id: "$groupField", count: {$sum: 1}}},  
 {$sort: {count: -1}}  
]);  
  
// Lookup (join)  
db.collection.aggregate([  
 {$lookup: {  
 from: "otherCollection",  
 localField: "field",  
 foreignField: "field",  
 as: "joinedData"  
 }}  
]);

### B. Docker Commands Reference

#### Container Management

# Run container  
docker run -d --name container-name image  
  
# Stop container  
docker stop container-name  
  
# Start container  
docker start container-name  
  
# Remove container  
docker rm container-name  
  
# View logs  
docker logs container-name

#### Docker Compose

# Start services  
docker-compose up -d  
  
# Stop services  
docker-compose down  
  
# View logs  
docker-compose logs -f  
  
# Rebuild services  
docker-compose up --build  
  
# Execute command in container  
docker-compose exec service-name command

### C. Configuration File Templates

#### MongoDB Configuration

# docker-compose.yml  
version: '3.8'  
services:  
 mongodb:  
 image: mongo:6.0  
 container\_name: mongodb  
 restart: always  
 ports:  
 - "27017:27017"  
 environment:  
 MONGO\_INITDB\_ROOT\_USERNAME: admin  
 MONGO\_INITDB\_ROOT\_PASSWORD: password  
 volumes:  
 - mongodb\_data:/data/db  
  
volumes:  
 mongodb\_data:

#### Mongo Express Configuration

# docker-compose.yml (extended)  
version: '3.8'  
services:  
 mongodb:  
 image: mongo:6.0  
 container\_name: mongodb  
 restart: always  
 environment:  
 MONGO\_INITDB\_ROOT\_USERNAME: admin  
 MONGO\_INITDB\_ROOT\_PASSWORD: password  
 volumes:  
 - mongodb\_data:/data/db  
  
 mongo-express:  
 image: mongo-express  
 container\_name: mongo-express  
 restart: always  
 ports:  
 - "8081:8081"  
 environment:  
 ME\_CONFIG\_MONGODB\_ADMINUSERNAME: admin  
 ME\_CONFIG\_MONGODB\_ADMINPASSWORD: password  
 ME\_CONFIG\_MONGODB\_URL: mongodb://admin:password@mongodb:27017/  
 depends\_on:  
 - mongodb  
  
volumes:  
 mongodb\_data: