**Hands-on: Menyimpan Data IoT ke MongoDB & Querying via MongoShell**

**Prerequisites**

* Docker dan Docker Compose terinstall
* Text editor (VS Code, Sublime, dll)
* Terminal/Command Prompt

**1. Setup MongoDB dengan Docker**

**1.1 Buat Docker Compose File**

Buat file docker-compose.yml:

version: '3.8'

services:

mongodb:

image: mongo:7.0

container\_name: mongodb-iot

restart: always

ports:

- "27017:27017"

environment:

MONGO\_INITDB\_ROOT\_USERNAME: admin

MONGO\_INITDB\_ROOT\_PASSWORD: password123

MONGO\_INITDB\_DATABASE: iot\_db

volumes:

- mongodb\_data:/data/db

- ./init-mongo.js:/docker-entrypoint-initdb.d/init-mongo.js:ro

networks:

- iot-network

mongo-express:

image: mongo-express:1.0.0

container\_name: mongo-express-iot

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/

ME\_CONFIG\_BASICAUTH: false

depends\_on:

- mongodb

networks:

- iot-network

volumes:

mongodb\_data:

networks:

iot-network:

driver: bridge

**1.2 Buat Script Inisialisasi Database**

Buat file init-mongo.js:

// Beralih ke database iot\_db

db = db.getSiblingDB('iot\_db');

// Buat user untuk aplikasi

db.createUser({

user: 'iot\_user',

pwd: 'iot\_password',

roles: [

{

role: 'readWrite',

db: 'iot\_db'

}

]

});

// Buat collection untuk sensor data

db.createCollection('sensor\_data');

db.createCollection('devices');

db.createCollection('alerts');

// Insert sample devices

db.devices.insertMany([

{

device\_id: 'ESP32\_001',

device\_name: 'Temperature Sensor Living Room',

location: 'Living Room',

device\_type: 'temperature\_humidity',

status: 'active',

created\_at: new Date(),

last\_seen: new Date()

},

{

device\_id: 'ESP32\_002',

device\_name: 'Motion Sensor Bedroom',

location: 'Bedroom',

device\_type: 'motion',

status: 'active',

created\_at: new Date(),

last\_seen: new Date()

},

{

device\_id: 'ESP32\_003',

device\_name: 'Light Sensor Garden',

location: 'Garden',

device\_type: 'light',

status: 'active',

created\_at: new Date(),

last\_seen: new Date()

}

]);

print('Database initialization completed!');

**1.3 Jalankan MongoDB**

# Jalankan container

docker-compose up -d

# Cek status container

docker-compose ps

# Lihat logs

docker-compose logs mongodb

**2. Simulasi Data IoT**

**2.1 Buat Script Python untuk Generate Data**

Buat file iot\_data\_generator.py:

import pymongo

import random

import time

from datetime import datetime, timedelta

import json

# Connection ke MongoDB

client = pymongo.MongoClient("mongodb://iot\_user:iot\_password@localhost:27017/iot\_db")

db = client.iot\_db

def generate\_temperature\_data(device\_id):

"""Generate data sensor suhu dan kelembaban"""

return {

'device\_id': device\_id,

'timestamp': datetime.utcnow(),

'sensor\_type': 'temperature\_humidity',

'data': {

'temperature': round(random.uniform(20.0, 35.0), 2),

'humidity': round(random.uniform(40.0, 80.0), 2),

'heat\_index': round(random.uniform(25.0, 40.0), 2)

},

'location': 'Living Room',

'battery\_level': random.randint(60, 100),

'signal\_strength': random.randint(-80, -30)

}

def generate\_motion\_data(device\_id):

"""Generate data sensor gerak"""

return {

'device\_id': device\_id,

'timestamp': datetime.utcnow(),

'sensor\_type': 'motion',

'data': {

'motion\_detected': random.choice([True, False]),

'motion\_intensity': random.randint(1, 10) if random.choice([True, False]) else 0

},

'location': 'Bedroom',

'battery\_level': random.randint(70, 100),

'signal\_strength': random.randint(-75, -35)

}

def generate\_light\_data(device\_id):

"""Generate data sensor cahaya"""

return {

'device\_id': device\_id,

'timestamp': datetime.utcnow(),

'sensor\_type': 'light',

'data': {

'light\_intensity': random.randint(0, 1000),

'uv\_index': round(random.uniform(0, 11), 1)

},

'location': 'Garden',

'battery\_level': random.randint(50, 100),

'signal\_strength': random.randint(-70, -40)

}

def generate\_alert(device\_id, alert\_type, message):

"""Generate alert jika ada kondisi tertentu"""

return {

'device\_id': device\_id,

'timestamp': datetime.utcnow(),

'alert\_type': alert\_type,

'severity': random.choice(['low', 'medium', 'high']),

'message': message,

'acknowledged': False,

'created\_at': datetime.utcnow()

}

def main():

print("Starting IoT Data Generator...")

try:

while True:

# Generate data untuk setiap device

temp\_data = generate\_temperature\_data('ESP32\_001')

motion\_data = generate\_motion\_data('ESP32\_002')

light\_data = generate\_light\_data('ESP32\_003')

# Insert data ke MongoDB

db.sensor\_data.insert\_one(temp\_data)

db.sensor\_data.insert\_one(motion\_data)

db.sensor\_data.insert\_one(light\_data)

# Generate alerts berdasarkan kondisi tertentu

if temp\_data['data']['temperature'] > 30:

alert = generate\_alert('ESP32\_001', 'high\_temperature',

f"High temperature detected: {temp\_data['data']['temperature']}°C")

db.alerts.insert\_one(alert)

if motion\_data['data']['motion\_detected'] and motion\_data['data']['motion\_intensity'] > 7:

alert = generate\_alert('ESP32\_002', 'motion\_detected',

f"High motion intensity detected: {motion\_data['data']['motion\_intensity']}")

db.alerts.insert\_one(alert)

if light\_data['data']['light\_intensity'] < 100:

alert = generate\_alert('ESP32\_003', 'low\_light',

f"Low light detected: {light\_data['data']['light\_intensity']} lux")

db.alerts.insert\_one(alert)

print(f"Data inserted at {datetime.now().strftime('%Y-%m-%d %H:%M:%S')}")

time.sleep(10) # Generate data setiap 10 detik

except KeyboardInterrupt:

print("\\nData generator stopped.")

finally:

client.close()

if \_\_name\_\_ == "\_\_main\_\_":

main()

**2.2 Install Dependencies dan Jalankan Generator**

# Install pymongo

pip install pymongo

# Jalankan generator (biarkan berjalan di background)

python iot\_data\_generator.py

**3. MongoDB Shell Commands**

**3.1 Connect ke MongoDB Shell**

# Connect ke MongoDB container

docker exec -it mongodb-iot mongosh

# Atau connect dengan authentication

docker exec -it mongodb-iot mongosh -u admin -p password123 --authenticationDatabase admin

**3.2 Basic Commands**

// Beralih ke database iot\_db

use iot\_db

// Authenticate sebagai iot\_user

db.auth('iot\_user', 'iot\_password')

// Lihat collections

show collections

// Lihat devices

db.devices.find().pretty()

// Count total sensor data

db.sensor\_data.countDocuments()

**4. Querying Data IoT**

**4.1 Basic Queries**

// Ambil 10 data sensor terbaru

db.sensor\_data.find().sort({timestamp: -1}).limit(10).pretty()

// Filter berdasarkan device\_id

db.sensor\_data.find({device\_id: "ESP32\_001"}).pretty()

// Filter berdasarkan sensor\_type

db.sensor\_data.find({sensor\_type: "temperature\_humidity"}).pretty()

// Filter berdasarkan range waktu (1 jam terakhir)

db.sensor\_data.find({

timestamp: {

$gte: new Date(Date.now() - 60\*60\*1000)

}

}).pretty()

// Filter berdasarkan suhu > 25°C

db.sensor\_data.find({

"data.temperature": {$gt: 25}

}).pretty()

**4.2 Advanced Queries**

// Aggregation: Rata-rata suhu per device

db.sensor\_data.aggregate([

{$match: {sensor\_type: "temperature\_humidity"}},

{$group: {

\_id: "$device\_id",

avg\_temperature: {$avg: "$data.temperature"},

avg\_humidity: {$avg: "$data.humidity"},

count: {$sum: 1}

}},

{$sort: {avg\_temperature: -1}}

])

// Agregasi data per hari

db.sensor\_data.aggregate([

{$match: {sensor\_type: "temperature\_humidity"}},

{$group: {

\_id: {

year: {$year: "$timestamp"},

month: {$month: "$timestamp"},

day: {$dayOfMonth: "$timestamp"}

},

avg\_temp: {$avg: "$data.temperature"},

min\_temp: {$min: "$data.temperature"},

max\_temp: {$max: "$data.temperature"},

count: {$sum: 1}

}},

{$sort: {"\_id.year": -1, "\_id.month": -1, "\_id.day": -1}}

])

// Find motion events dalam 24 jam terakhir

db.sensor\_data.find({

sensor\_type: "motion",

"data.motion\_detected": true,

timestamp: {

$gte: new Date(Date.now() - 24\*60\*60\*1000)

}

}).sort({timestamp: -1})

// Time series analysis: Data per jam

db.sensor\_data.aggregate([

{$match: {

sensor\_type: "temperature\_humidity",

timestamp: {$gte: new Date(Date.now() - 24\*60\*60\*1000)}

}},

{$group: {

\_id: {

year: {$year: "$timestamp"},

month: {$month: "$timestamp"},

day: {$dayOfMonth: "$timestamp"},

hour: {$hour: "$timestamp"}

},

avg\_temperature: {$avg: "$data.temperature"},

avg\_humidity: {$avg: "$data.humidity"},

readings\_count: {$sum: 1}

}},

{$sort: {"\_id.year": 1, "\_id.month": 1, "\_id.day": 1, "\_id.hour": 1}}

])

**4.3 Alert Queries**

// Semua alerts yang belum di-acknowledge

db.alerts.find({acknowledged: false}).sort({timestamp: -1})

// Alerts berdasarkan severity

db.alerts.find({severity: "high"}).sort({timestamp: -1})

// Count alerts per device

db.alerts.aggregate([

{$group: {

\_id: "$device\_id",

total\_alerts: {$sum: 1},

high\_severity: {$sum: {$cond: [{$eq: ["$severity", "high"]}, 1, 0]}},

latest\_alert: {$max: "$timestamp"}

}},

{$sort: {total\_alerts: -1}}

])

// Acknowledge alert

db.alerts.updateOne(

{\_id: ObjectId("YOUR\_ALERT\_ID")},

{$set: {acknowledged: true, acknowledged\_at: new Date()}}

)

**4.4 Performance Queries**

// Buat index untuk performance

db.sensor\_data.createIndex({device\_id: 1, timestamp: -1})

db.sensor\_data.createIndex({sensor\_type: 1, timestamp: -1})

db.sensor\_data.createIndex({timestamp: -1})

db.alerts.createIndex({device\_id: 1, timestamp: -1})

// Lihat execution plan

db.sensor\_data.find({device\_id: "ESP32\_001"}).explain("executionStats")

// Device status dengan last\_seen update

db.devices.updateOne(

{device\_id: "ESP32\_001"},

{$set: {last\_seen: new Date()}}

)

// Find devices yang tidak aktif (tidak ada data > 5 menit)

db.devices.aggregate([

{$lookup: {

from: "sensor\_data",

localField: "device\_id",

foreignField: "device\_id",

as: "recent\_data",

pipeline: [

{$match: {timestamp: {$gte: new Date(Date.now() - 5\*60\*1000)}}},

{$limit: 1}

]

}},

{$match: {"recent\_data": {$size: 0}}},

{$project: {device\_id: 1, device\_name: 1, last\_seen: 1}}

])

**5. Data Maintenance**

**5.1 Cleanup Commands**

// Delete data older than 30 days

db.sensor\_data.deleteMany({

timestamp: {$lt: new Date(Date.now() - 30\*24\*60\*60\*1000)}

})

// Archive old data to separate collection

db.sensor\_data.aggregate([

{$match: {timestamp: {$lt: new Date(Date.now() - 7\*24\*60\*60\*1000)}}},

{$out: "sensor\_data\_archive"}

])

// Delete archived data from main collection

db.sensor\_data.deleteMany({

timestamp: {$lt: new Date(Date.now() - 7\*24\*60\*60\*1000)}

})

**5.2 Backup Commands**

# Backup database

docker exec mongodb-iot mongodump --db iot\_db --out /data/backup

# Restore database

docker exec mongodb-iot mongorestore --db iot\_db /data/backup/iot\_db

**6. Monitoring & Web Interface**

**6.1 Akses Mongo Express**

Buka browser dan akses http://localhost:8081 untuk interface web MongoDB.

**6.2 Monitoring Queries**

// Database stats

db.stats()

// Collection stats

db.sensor\_data.stats()

// Current operations

db.currentOp()

// Server status

db.serverStatus()

**7. Tips & Best Practices**

1. **Indexing**: Selalu buat index pada field yang sering di-query
2. **TTL Collections**: Gunakan TTL index untuk auto-delete data lama
3. **Sharding**: Pertimbangkan sharding untuk data yang sangat besar
4. **Aggregation Pipeline**: Gunakan untuk analisis data kompleks
5. **Connection Pooling**: Gunakan connection pooling untuk aplikasi production

**Troubleshooting**

# Restart containers

docker-compose restart

# View logs

docker-compose logs -f mongodb

# Connect to container

docker exec -it mongodb-iot bash

# Check MongoDB process

docker exec mongodb-iot ps aux | grep mongo

**Cleanup**

# Stop dan remove containers

docker-compose down

# Remove volumes (hati-hati, akan menghapus data!)

docker-compose down -v

# Remove images

docker rmi mongo:7.0 mongo-express:1.0.0

Selamat mencoba! Panduan ini memberikan foundation yang solid untuk bekerja dengan data IoT menggunakan MongoDB dalam environment Docker.