LAPORAN PRAKTIKUM
PRAKTIKUM DATABASE PERTEMUAN 10

Disusun untuk Memenuhi Matakuliah Praktikum {Nama Matakuliah}
Dibimbing oleh : Sulaibatul Aslamiyah, M.Kom

Oleh:

RAFLI RAHMAN.EFENDY

1124102162

PROGRAM STUDI S1 TEKNIK INFORMATIKA
SEKOLAH TINGGI ILMU KOMPUTER PGRI BANYUWANGI
2025

LEMBAR PENGESAHAN

LAPORAN PRAKTIKUM

Matakuliah : Praktikum Database

Oleh : Rafli Rahman.Efendy

NIM : 1124102162

Telah disahkan pada

Hari : JUM'AT

Tanggal : 30/05/2025

Mengetahui/ Menyetujui :

Dosen Pengampu Mata Kuliah

Sulaibatul Aslamiyah, M.Kom

NIDN. 0712058304

KATA PENGANTAR

Berisikan kata pengantar yang dituliskan oleh mahasiswa untuk laporan praktikum matakuliah yang diikuti.

MODUL PRAKTIKUM 1

1.1 Capaian Praktikum Pertemuan Bulk Write Operation 1

Bulk write di MongoDB itu salah satu cara efisien untuk melakukan banyak operasi database sekaligus dalam satu panggilan ke server.

Biasanya dipakai kalau kita mau melakukan banyak insert, update, delete, atau kombinasi ketiganya, biar lebih cepat daripada kirim request satu per satu.

MongoDB mendukung Bulk Write Operation, yaitu operasi bulk yang dalam satu request, kita bisa mengirim banyak perintah sekaligus dalam satu waktu

1.2 Indikator Capaian

- Menjelaskan konsep dasar bulkwrite
- mengidentifikasi jenis-jenis operasi bulk write seperti insertMany, updateMany, deleteMany dan kombinasi operasi lainnya
- Menerapkan operasi bulkwrite pada sistem basis data

1.3 Landasan Teori

Databases adalah kumpulan data yang terstruktur dan disimpan secara terorganisir untuk memudahkan pengelolaan dan pencarian informasi

1.4 Pelaksanaan Praktikum

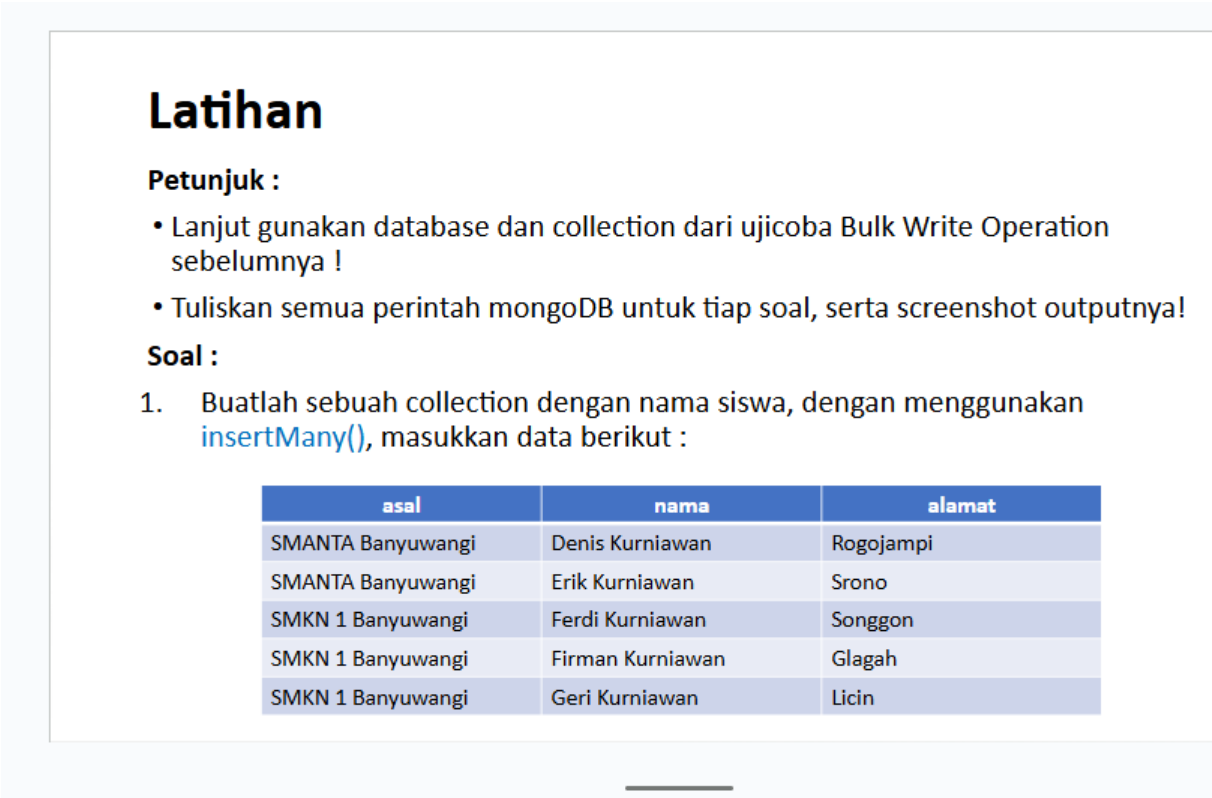
Model data embedded cocok untuk digunakan ketika : Antar document saling ketergantungan Kita jarang melakukan perubahan isi data pada embedded document Embedded document selalu dibutuhkan ketika mengambil data document

1.4.1 Percobaan Pertama

Latihan Pratikum Data Base MongoDB

Bulk Write Operation

- Bulk write di MongoDB itu salah satu cara efisien untuk melakukan banyak operasi database sekaligus dalam satu panggilan ke server.
- Biasanya dipakai kalau kita mau melakukan banyak insert, update, delete, atau kombinasi ketiganya, biar lebih cepat daripada kirim request satu per satu.
- MongoDB mendukung Bulk Write Operation, yaitu operasi bulk yang dalam satu request, kita bisa mengirim banyak perintah sekaligus dalam satu waktu



Latihan

Petunjuk :

- Lanjut gunakan database dan collection dari ujicoba Bulk Write Operation sebelumnya !
- Tuliskan semua perintah mongoDB untuk tiap soal, serta screenshot outputnya!

Soal :

1. Buatlah sebuah collection dengan nama siswa, dengan menggunakan insertMany(), masukkan data berikut :

asal	nama	alamat
SMANTA Banyuwangi	Denis Kurniawan	Rogojampi
SMANTA Banyuwangi	Erik Kurniawan	Srono
SMKN 1 Banyuwangi	Ferdi Kurniawan	Songgon
SMKN 1 Banyuwangi	Firman Kurniawan	Glagah
SMKN 1 Banyuwangi	Geri Kurniawan	Licin

Latihan Soal 1

Membuat Collection siswa dan insert data dengan insertMany()

```
db.siswa.insertMany([
  { asal: "SMANTA Banyuwangi", nama: "Denis Kurniawan", alamat: "Rogojampi" },
  { asal: "SMANTA Banyuwangi", nama: "Erik Kurniawan", alamat: "Srono" },
  { asal: "SMKN 1 Banyuwangi", nama: "Ferdi Kurniawan", alamat: "Songgon" },
  { asal: "SMKN 1 Banyuwangi", nama: "Firman Kurniawan", alamat: "Glagah" },
  { asal: "SMKN 1 Banyuwangi", nama: "Geri Kurniawan", alamat: "Licin" }
]);
```

Penjelasan nya:

Sebelum membuat data collection nya kita harus membuat data nya terlebih dahulu dengan perintah

“Use” Untuk membuat sebuah Data.

```
biodata> show dbs
PertemuanTujuh  184.00 KiB
admin            40.00 KiB
biodata          80.00 KiB
config           72.00 KiB
local            72.00 KiB
mahasiswa        40.00 KiB
mahasiwa         72.00 KiB
matakuliah       80.00 KiB
soallatihan     184.00 KiB
biodata> use  bulkwrite
switched to db bulkwrite
bulkwrite> db.createCollection("bulkwrite")
{ ok: 1 }
```

use bulkwrite

Perintah ini digunakan untuk berpindah atau membuat database bernama bulkwrite.

Jika database bulkwrite belum ada, MongoDB akan membuatnya secara otomatis saat kamu pertama kali menyimpan data di dalamnya.

2 Penjelasan nya:

db.createCollection("bulkwrite")

Perintah ini digunakan untuk membuat koleksi (collection) bernama "bulkwrite" di dalam database bulkwrite.

- koleksi di MongoDB seperti tabel dalam database relasional.
- Di dalam koleksi ini nantinya kamu bisa insert, find, update, delete, dll.

Hasil nya:

{ ok: 1 }

Artinya:

- Koleksi berhasil dibuat.
- ok: 1 menunjukkan operasi sukses tanpa error.

Petunjuk :

- Lanjut gunakan database dan collection dari ujicoba Bulk Write Operation sebelumnya !
- Tuliskan semua perintah mongoDB untuk tiap soal, serta screenshot outputnya!

1.Buatlah sebuah collection dengan nama siswa, dengan menggunakan insertMany(), masukkan data berikut :

```
bulkwrite> db.siswa.insertMany([
...   { asal: "SMANTA Banyuwangi", nama: "Denis Kurniawan", alamat: "Rogojampi" },
...   { asal: "SMANTA Banyuwangi", nama: "Erik Kurniawan", alamat: "Srono" },
...   { asal: "SMKN 1 Banyuwangi", nama: "Ferdi Kurniawan", alamat: "Songgon" },
...   { asal: "SMKN 1 Banyuwangi", nama: "Firman Kurniawan", alamat: "Glagah" },
...   { asal: "SMKN 1 Banyuwangi", nama: "Geri Kurniawan", alamat: "Licin" }
... ]);
...
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('683926ca77a581b6f6b71237'),
    '1': ObjectId('683926ca77a581b6f6b71238'),
    '2': ObjectId('683926ca77a581b6f6b71239'),
    '3': ObjectId('683926ca77a581b6f6b7123a'),
    '4': ObjectId('683926ca77a581b6f6b7123b')
  }
}
```

```
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('683926ca77a581b6f6b71237'),
    '1': ObjectId('683926ca77a581b6f6b71238'),
    '2': ObjectId('683926ca77a581b6f6b71239'),
    '3': ObjectId('683926ca77a581b6f6b7123a'),
    '4': ObjectId('683926ca77a581b6f6b7123b')
  }
}
bulkwrite> db.bulkwrite.find();
```

Penjelasan Output nya:

acknowledged: true,
insertedIds: {
'0': ObjectId('683926ca77a581b6f6b71237'),
'1': ObjectId('683926ca77a581b6f6b71238'),
'2': ObjectId('683926ca77a581b6f6b71239'),
'3': ObjectId('683926ca77a581b6f6b7123a'),
'4': ObjectId('683926ca77a581b6f6b7123b')
}

1. acknowledged: true

- Artinya permintaan insert berhasil diterima dan diproses oleh MongoDB server.
- Ini adalah konfirmasi bahwa operasi insertMany() berjalan sukses.

2. insertedIds

- Ini adalah objek yang menunjukkan ID (ObjectId) dari setiap dokumen yang baru saja dimasukkan ke dalam koleksi.
- Key '0', '1', dan Seterus snya'4' mewakili urutan index dari dokumen yang kamu masukkan.
- Nilai ObjectId(...) adalah ID unik otomatis yang diberikan MongoDB ke setiap dokumen baru.

Contoh Konteks: Misalnya kamu menjalankan

```
db.siswa.insertMany([
  { nama: "Joko" },
  { nama: "Budi" },
  { nama: "Ani" },
  { nama: "Tini" },
  { nama: "Rudi" }
]);
```

Maka MongoDB akan mengembalikan insertedIds untuk masing-masing dokumen tersebut, sesuai urutan inputnya, beserta ObjectId unik mereka.

Kesimpulan: Output ini memberitahumu bahwa:

- Kamu berhasil menyimpan 5 dokumen.
- Masing-masing dokumen sudah diberikan ObjectId.
- Kamu bisa menggunakan ObjectId ini untuk mengakses atau memanipulasi dokumen secara spesifik di kemudian hari.

2.Dengan menggunakan operasi bulkWrite( ) :

```
bulkwrite> db.siswa.bulkWrite([
...   {
...     insertOne: {
...       document: {
...         asal: "MAN Banyuwangi",
...         nama: "Harun Kurniawan",
...         alamat: "Kabat"
...       }
...     }
...   },
...   {
...     insertOne: {
...       document: {
...         asal: "MAN Banyuwangi",
...         nama: "Iwan Kurniawan",
...         alamat: "Kalipuro"
...       }
...     }
...   },
...   {
...     updateMany: {
...       filter: { asal: "MAN Banyuwangi" },
...       update: { $set: { asal: "MAN 1 Banyuwangi" } }
...     }
...   },
...   {
...     replaceOne: {
...       filter: { nama: "Denis Kurniawan" },
...       replacement: {
...         asal: "SMAN 1 Giri",
...         nama: "Joko Kurniawan",
...         alamat: "Muncar"
```

```
...     }
...   }
... },
... {
...   deleteOne: {
...     filter: { nama: "Ferdi Kurniawan" }
...   }
... }
... ]);
```

```
bulkwrite> db.siswa.bulkWrite([
...   {
...     insertOne: {
...       document: {
...         asal: "MAN Banyuwangi",
...         nama: "Harun Kurniawan",
...         alamat: "Kabat"
...       }
...     }
...   },
...   {
...     insertOne: {
...       document: {
...         asal: "MAN Banyuwangi",
...         nama: "Iwan Kurniawan",
...         alamat: "Kalipuro"
...       }
...     }
...   },
...   {
...     updateMany: {
...       filter: { asal: "MAN Banyuwangi" },
...       update: { $set: { asal: "MAN 1 Banyuwangi" } }
...     }
...   },
...   {
...     replaceOne: {
...       filter: { nama: "Denis Kurniawan" },
...       replacement: {
...         asal: "SMAN 1 Giri",
...         nama: "Joko Kurniawan",
...         alamat: "Muncar"
...       }
...     }
...   },
...   {
...     deleteOne: {
...       filter: { nama: "Ferdi Kurniawan" }
...     }
...   }
... ]);
...
{
```

Fungsi bulkWrite([...])

bulkWrite digunakan untuk melakukan banyak operasi sekaligus (seperti insertOne, updateMany, replaceOne, deleteOne) dalam satu eksekusi.

Penjelasan per bagian:

1. insertOne: Menambahkan Harun Kurniawan: Menambahkan data baru siswa bernama Harun Kurniawan.

```
{
  insertOne: {
    document: {
      asal: "MAN Banyuwangi",
      nama: "Harun Kurniawan",
      alamat: "Kabat"
    }
  }
}
```

2. insertOne: Menambahkan Iwan Kurniawan: Menambahkan data baru siswa bernama Iwan Kurniawan.

```
{
  insertOne: {
    document: {
      asal: "MAN Banyuwangi",
      nama: "Iwan Kurniawan",
      alamat: "Kalipuro"
    }
  }
}
```

3. updateMany: Update asal semua siswa dari "MAN Banyuwangi" ke "MAN 1 Banyuwangi": Mengubah field asal dari semua siswa yang sebelumnya "MAN Banyuwangi" menjadi "MAN 1 Banyuwangi".

```
{
  updateMany: {
    filter: { asal: "MAN Banyuwangi" },
    update: { $set: { asal: "MAN 1 Banyuwangi" } }
  }
}
```

4. replaceOne: Ganti seluruh data siswa bernama Denis Kurniawan: Jika ada siswa bernama "Denis Kurniawan", maka seluruh dokumen dia diganti sepenuhnya menjadi data baru bernama "Joko Kurniawan".

```
{
  replaceOne: {
    filter: { nama: "Denis Kurniawan" },
    replacement: {
      asal: "SMAN 1 Giri",
      nama: "Joko Kurniawan",
      alamat: "Muncar"
```

```
   }
  }
}
```

5. deleteOne: Hapus siswa bernama Ferdi Kurniawan: Menghapus satu dokumen pertama yang cocok dengan nama: "Ferdi Kurniawan".

```
{
  deleteOne: {
    filter: { nama: "Ferdi Kurniawan" }
  }
}
```

Penggunaan bulkWrite( )

- Lebih efisien karena hanya satu komunikasi ke server.
- Dapat menggabungkan berbagai jenis operasi (insert, update, delete, replace).
- Cocok untuk proses data besar atau migrasi data.

Kalau kamu ingin lihat hasil dari bulkWrite() (seperti berapa data yang berhasil di-insert, di-update, atau dihapus), bisa tambahkan log seperti:

```
const hasil = db.siswa.bulkWrite([...]);
printjson(hasil);
```

A. Tambahkan data dibawah dengan menggunakan insertOne()

```
db.siswa.insertMany([
  {
    _id: ObjectId('6839285b77a581b6f6b7123c'),
    asal: 'MAN 1 Banyuwangi',
    nama: 'Harun Kurniawan',
    alamat: 'Kabat'
  },
  {
    _id: ObjectId('6839285b77a581b6f6b7123d'),
    asal: 'MAN 1 Banyuwangi',
    nama: 'Iwan Kurniawan',
    alamat: 'Kalipuro'
  }
]);
```

```
bulkwrite> db.siswa.insertOne({
...   asal: "SMA Negeri 2 Genteng",
...   nama: "Aldi Pratama",
...   alamat: "Gambiran"
... });
...
{
  acknowledged: true,
  insertedId: ObjectId('683926aa77a581b6f6b71236')
}
```

```
bulkwrite> db.siswa.insertMany([
...   {
...     _id: ObjectId('6839285b77a581b6f6b7123c'),
...     asal: 'MAN 1 Banyuwangi',
...     nama: 'Harun Kurniawan',
...     alamat: 'Kabat'
...   },
...   {
...     _id: ObjectId('6839285b77a581b6f6b7123d'),
...     asal: 'MAN 1 Banyuwangi',
...     nama: 'Iwan Kurniawan',
...     alamat: 'Kalipuro'
...   }
... ]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('6839285b77a581b6f6b7123c'),
    '1': ObjectId('6839285b77a581b6f6b7123d')
  }
}
bulkwrite>
```

```
bulkwrite> db.siswa.find({
...   nama: { $in: ['Harun Kurniawan', 'Iwan Kurniawan'] }
... });
...
[
  {
    _id: ObjectId('6839285b77a581b6f6b7123c'),
    asal: 'MAN 1 Banyuwangi',
    nama: 'Harun Kurniawan',
    alamat: 'Kabat'
  },
  {
    _id: ObjectId('6839285b77a581b6f6b7123d'),
    asal: 'MAN 1 Banyuwangi',
    nama: 'Iwan Kurniawan',
    alamat: 'Kalipuro'
  }
]
```

Penjelasan nya Baris per Baris:

db.siswa.insertMany([...])

- Memasukkan lebih dari satu dokumen ke dalam koleksi siswa.
- insertMany() adalah metode MongoDB untuk menambahkan beberapa data sekaligus.

Dokumen Pertama:

```
{
  _id: ObjectId('6839285b77a581b6f6b7123c'),
  asal: 'MAN 1 Banyuwangi',
  nama: 'Harun Kurniawan',
  alamat: 'Kabat'
}
```

- bjectId('6839285b77a581b6f6b7123c'): Nilai unik sebagai _id (kamu tentukan sendiri, bukan otomatis dibuat MongoDB).
- asal: Asal sekolah, yaitu MAN 1 Banyuwangi
- nama: Nama siswa, Harun Kurniawan
- alamat: Alamat rumah, yaitu Kabat

Dokumen Kedua: Sama seperti dokumen pertama, tapi data berbeda (nama dan alamat).

```
{
  _id: ObjectId('6839285b77a581b6f6b7123d'),
  asal: 'MAN 1 Banyuwangi',
  nama: 'Iwan Kurniawan',
  alamat: 'Kalipuro'
}
```

Hasil Akhir:

- Dua data siswa ditambahkan ke koleksi siswa.
- Jika _id belum ada, maka data ditambahkan berhasil.
- Jika _id sudah ada di database, maka insertMany akan gagal untuk seluruh operasi, kecuali kamu tambahkan opsi ordered: false.

Tips:

Untuk menghindari error karena _id sudah ada, kamu bisa hapus dulu data sebelumnya:

```
db.siswa.deleteMany({
  _id: {
    $in: [
      ObjectId('6839285b77a581b6f6b7123c'),
      ObjectId('6839285b77a581b6f6b7123d')
    ]
  }
});
```

Penjelasan Output nya:

Output berikut muncul setelah kamu menjalankan perintah db.siswa.insertMany(...) di MongoDB:

```
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('6839285b77a581b6f6b7123c'),
    '1': ObjectId('6839285b77a581b6f6b7123d')
  }
}
```

Penjelasan Baris per Baris:

acknowledged: true

- Artinya perintah berhasil dijalankan dan MongoDB mengakui (acknowledge) bahwa data berhasil dimasukkan ke koleksi**.**

insertedIds

- Ini adalah objek yang menunjukkan ID dari setiap dokumen yang berhasil dimasukkan.
- Disusun dalam bentuk pasangan key: value, di mana:
    - key = indeks array dari dokumen saat dimasukkan
    - value = _id dari dokumen tersebut

Detailnya:

```
'0': ObjectId('6839285b77a581b6f6b7123c'),
'1': ObjectId('6839285b77a581b6f6b7123d')
```

- '0' dokumen pertama di array insertMany
- (yaitu data Harun Kurniawan)
- '1' dokumen kedua di array
- (yaitu data Iwan Kurniawan)
- ObjectId(...) adalah ID unik yang kamu tentukan sendiri saat insert.

Kesimpulan:

Output ini memberitahumu bahwa kedua data berhasil dimasukkan ke koleksi siswa, dan masing-masing memiliki _id yang tercatat.Kalau kamu tidak menyertakan _id secara manual, MongoDB akan membuatkan otomatis dan menampilkannya di sini juga.

```
bulkwrite> db.siswa.find({
...   nama: { $in: ['Harun Kurniawan', 'Iwan Kurniawan'] }
... });
...
[
  {
    _id: ObjectId('6839285b77a581b6f6b7123c'),
    asal: 'MAN 1 Banyuwangi',
    nama: 'Harun Kurniawan',
    alamat: 'Kabat'
  },
  {
    _id: ObjectId('6839285b77a581b6f6b7123d'),
    asal: 'MAN 1 Banyuwangi',
    nama: 'Iwan Kurniawan',
    alamat: 'Kalipuro'
  }
]
```

B. Dengan perintah updateMany(), ubah data yang asal = “MAN Banyuwangi” menjadi asal = “MAN 1 Banyuwangi”.

1.Update Many Man 1 Banyuwangi ke Man Banyuwangi:

```
bulkwrite> db.siswa.updateMany(
...   { asal: "MAN 1 Banyuwangi" },  // filter dokumen yang ingin diubah
...   { $set: { asal: "MAN Banyuwangi" } }  // nilai baru yang akan diset
... );
...
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 2,
  modifiedCount: 2,
  upsertedCount: 0
}
bulkwrite>
```

```
bulkwrite> db.siswa.find({ asal: "MAN Banyuwangi" }).pretty();
[
  {
    _id: ObjectId('6839285b77a581b6f6b7123c'),
    asal: 'MAN Banyuwangi',
    nama: 'Harun Kurniawan',
    alamat: 'Kabat'
  },
  {
    _id: ObjectId('6839285b77a581b6f6b7123d'),
    asal: 'MAN Banyuwangi',
    nama: 'Iwan Kurniawan',
    alamat: 'Kalipuro'
  }
]
```

```
db.siswa.updateMany(
  { asal: "MAN 1 Banyuwangi" },      // filter dokumen yang ingin diubah
  { $set: { asal: "MAN Banyuwangi" } }  // nilai baru yang akan diset
);
```

Penjelasan baris per baris:

1.db.siswa.updateMany(...)

- Perintah untuk mengubah banyak dokumen sekaligus di koleksi siswa.
- updateMany berbeda dengan updateOne, karena ini mengupdate semua dokumen yang memenuhi filter, bukan hanya satu.

2. { asal: "MAN 1 Banyuwangi" }

- Filter atau kriteria pencarian dokumen yang ingin diubah.
- Artinya: cari semua dokumen yang field asal-nya bernilai persis "MAN 1 Banyuwangi".

3. { $set: { asal: "MAN Banyuwangi" } }

- Ini adalah operasi update yang dilakukan pada dokumen yang cocok dengan filter.
- $set adalah operator untuk mengubah/mengganti nilai dari field yang ditentukan.
- Jadi, nilai field asal akan diubah dari "MAN 1 Banyuwangi" menjadi "MAN Banyuwangi".

Apa yang terjadi?

- Semua dokumen di koleksi siswa yang punya asal = "MAN 1 Banyuwangi" akan diupdate supaya nilai asal menjadi "MAN Banyuwangi".
- Dokumen lain yang tidak sesuai filter tidak akan terpengaruh.

2.Update Many Man Banyuwangi ke Man 1 Banyuwangi:

```
bulkwrite> db.siswa.updateMany(
...   { asal: "MAN Banyuwangi" },           // kondisi dokumen yang ingin diubah
...   { $set: { asal: "MAN 1 Banyuwangi" } } // perubahan nilai field
... );
...
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 2,
  modifiedCount: 2,
  upsertedCount: 0
}
bulkwrite>
```

```
bulkwrite> db.siswa.find({ asal: "MAN 1 Banyuwangi" }).pretty();
[
  {
    _id: ObjectId('6839285b77a581b6f6b7123c'),
    asal: 'MAN 1 Banyuwangi',
    nama: 'Harun Kurniawan',
    alamat: 'Kabat'
  },
  {
    _id: ObjectId('6839285b77a581b6f6b7123d'),
    asal: 'MAN 1 Banyuwangi',
    nama: 'Iwan Kurniawan',
    alamat: 'Kalipuro'
  }
]
```

```
db.siswa.updateMany(
  { asal: "MAN Banyuwangi" },          // kondisi dokumen yang ingin diubah
  { $set: { asal: "MAN 1 Banyuwangi" } } // perubahan nilai field
);
```

Penjelasan:

1. db.siswa.updateMany(...)

- Perintah untuk mengupdate banyak dokumen sekaligus di koleksi siswa.

- Berbeda dengan updateOne, updateMany akan mengubah semua dokumen yang memenuhi kondisi filter.

2. { asal: "MAN Banyuwangi" }

- Ini adalah filter/kondisi yang menentukan dokumen mana yang akan diubah.
- Hanya dokumen yang memiliki field asal dengan nilai "MAN Banyuwangi" saja yang akan diproses.

3. { $set: { asal: "MAN 1 Banyuwangi" } }

- Operator $set digunakan untuk mengubah nilai field tertentu dalam dokumen.
- Dalam hal ini, nilai asal akan diubah menjadi "MAN 1 Banyuwangi".

Apa yang terjadi?

Semua dokumen dalam koleksi siswa yang field asal nya adalah "MAN Banyuwangi" akan diubah menjadi "MAN 1 Banyuwangi".

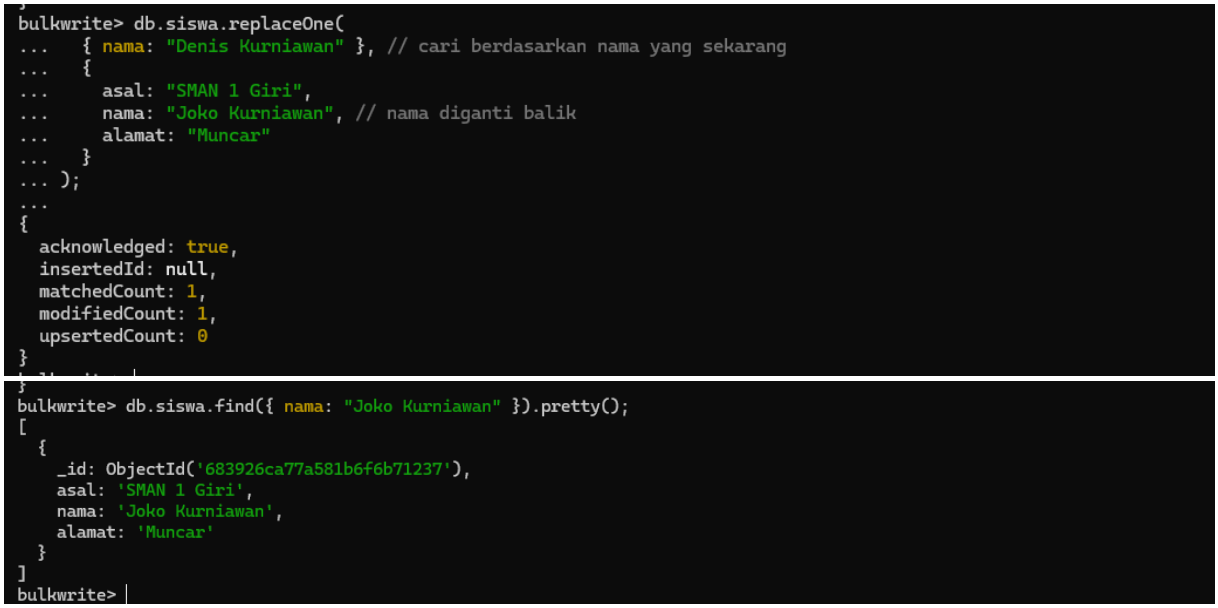
Latihan – Lanj.

d) Dengan menggunakan replaceOne(), ganti data nama =“Denis Kurniawan” dengan :

asal	nama	alamat
SMAN 1 Giri	Joko Kurniawan	Muncar

e) Dengan menggunakan perintah deleteOne(), hapus data dengan nama = “Ferdi Kurniawan”

D. Dengan menggunakan replaceOne(), ganti data nama =“Denis Kurniawan” dengan :




```
db.siswa.replaceOne(
  { nama: "Denis Kurniawan" },  // filter dokumen lama
  {
    asal: "SMAN 1 Giri",
    nama: "Joko Kurniawan",
    alamat: "Muncar"
  }
);
```

Penjelasan Baris per Baris:

db.siswa.replaceOne(...)

- Ini adalah perintah untuk mengganti seluruh isi satu dokumen dalam koleksi siswa.
- Digunakan jika kamu ingin menghapus semua field lama (kecuali _id) dan menggantinya dengan isi baru.

{ nama: "Denis Kurniawan" }

- Ini adalah filter untuk mencari dokumen yang ingin diganti.
- MongoDB akan mencari satu dokumen yang memiliki nama = "Denis Kurniawan".

Dokumen Baru:

```
{
  asal: "SMAN 1 Giri",
  nama: "Joko Kurniawan",
  alamat: "Muncar"
}
```

- Ini adalah pengganti dari dokumen yang lama.

- Seluruh dokumen lama akan diganti total menjadi data ini.
- Jika dokumen lama punya field tambahan (misalnya usia, kelas, dll), maka field itu akan hilang.

Penting:

- Jika tidak ada dokumen dengan nama: "Denis Kurniawan", maka tidak ada perubahan yang terjadi.
- Jika ingin menyimpan _id yang lama, kamu harus menyertakannya secara eksplisit di dokumen baru, karena replaceOne akan menghapus semua field kecuali _id.

```
bulkwrite> db.siswa.replaceOne(
...   { nama: "Joko Kurniawan" }, // filter dokumen yang ingin diganti
...   {
...     asal: "SMAN 1 Giri",
...     nama: "Denis Kurniawan", // nama diganti
...     alamat: "Muncar"
...   }
... );
...
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

```
bulkwrite> db.siswa.find({ nama: "Denis Kurniawan" }).pretty();
[
  {
    _id: ObjectId('683926ca77a581b6f6b71237'),
    asal: 'SMAN 1 Giri',
    nama: 'Denis Kurniawan',
    alamat: 'Muncar'
  }
]
```

E. Dengan menggunakan perintah deleteOne(), hapus data dengan nama = “Ferdi Kurniawan”

```
]
bulkwrite> db.siswa.deleteOne({ nama: "Ferdi Kurniawan" });
{ acknowledged: true, deletedCount: 1 }
bulkwrite>
```

```
]
bulkwrite> db.siswa.deleteOne({ nama: "Ferdi Kurniawan" });
{ acknowledged: true, deletedCount: 1 }
bulkwrite> db.siswa.find({ nama: "Ferdi Kurniawan" });

bulkwrite>
```

```
bulkwrite> db.siswa.insertOne({
...   asal: "SMKN 1 Banyuwangi",
...   nama: "Ferdi Kurniawan",
...   alamat: "Songgon"
... });
...
{
  acknowledged: true,
  insertedId: ObjectId('6839330977a581b6f6b7123f')
}
bulkwrite> db.siswa.find({ nama: "Ferdi Kurniawan" }).pretty();
[
  {
    _id: ObjectId('6839330977a581b6f6b7123f'),
    asal: 'SMKN 1 Banyuwangi',
    nama: 'Ferdi Kurniawan',
    alamat: 'Songgon'
  }
]
```

Penjelasan Baris per Baris:

db.siswa.deleteOne(...)

- Ini adalah perintah untuk menghapus satu dokumen dari koleksi siswa.
- Metode deleteOne() akan mencari satu dokumen pertama yang cocok dengan filter, lalu menghapusnya.

{ nama: "Ferdi Kurniawan" }

- Ini adalah filter untuk menentukan dokumen mana yang akan dihapus.
- MongoDB akan mencari dokumen pertama di mana nama bernilai "Ferdi Kurniawan".

Hasil Output nya:

{ acknowledged: true, deletedCount: 1 **}**

acknowledged: true

- Menunjukkan bahwa MongoDB menerima dan memproses perintah penghapusan dengan sukses.

deletedCount: 1

- Artinya 1 dokumen berhasil dihapus dari koleksi.

Catatan Penting:

- Jika tidak ada dokumen dengan nama "Ferdi Kurniawan", maka deletedCount akan menjadi 0.
- deleteOne() hanya menghapus satu dokumen saja, meskipun ada beberapa yang cocok.
- Jika ingin menghapus semua dokumen yang cocok, gunakan deleteMany().

Di MongoDB, data yang sudah dihapus tidak bisa dikembalikan atau dicek lagi secara langsung, karena MongoDB tidak menyimpan histori penghapusan secara default.

1. Cek Apakah Data Masih Ada

Jika kamu ingin memastikan bahwa data benar-benar terhapus, kamu bisa cari ulang datanya:

```
db.siswa.find({ nama: "Ferdi Kurniawan" });
```

Jika hasilnya kosong ([]), maka data tersebut memang sudah terhapus.

2. Gunakan deletedCount dari Perintah deleteOne() atau deleteMany() Seperti ini:

```
{ acknowledged: true, deletedCount: 1 }
```

MongoDB berhasil menghapus 1 data yang cocok dengan filter.

3. Log Manual Sebelum Hapus (Opsional)

Agar bisa tahu data apa saja yang dihapus, kamu bisa tampilkan dulu data sebelum menghapusnya:

```
db.siswa.find({ nama: "Ferdi Kurniawan" });
```

Lalu jika yakin, baru jalankan:

```
db.siswa.deleteOne({ nama: "Ferdi Kurniawan" });
```

4. Backup Koleksi Sebelum Hapus (Opsional)

Kalau kamu ingin jaga-jaga agar data yang dihapus bisa dikembalikan, kamu bisa copy isi koleksi ke koleksi lain dulu:

```
db.siswa.find({}).forEach(doc => db.siswa_backup.insertOne(doc));
```

Dengan begitu, kamu bisa mengakses data lama di siswa_backup.

Penting:

MongoDB tidak punya fitur "Recycle Bin" atau "Trash" seperti di sistem operasi.

Jika datanya sudah terhapus dan belum dibackup, maka tidak bisa dikembalikan lagi.

Kesimpulan

Kesimpulan Percobaan 1

Dengan study kasus ini Saya bisa membuat Relasi One-to-One adalah relasi dimana suatu baris tabel A hanya berhubungan dengan suatu baris tabel B. Kita dapat menerapkan model data Embedded Document untuk mempresentasikan relasi One-to-One tersebut pada database Non Relational.

Kesimpulan Percobaan 2

Dengan study kasus ini Saya bisa membuat Mirip dengan implementasi embedded document pada relasi One-to-One, pada kasus One-to-Many, document dengan relasi “Many” akan di-embedded kedalam document dengan relasi “One”.

Kesimpulan Percobaan 3

Dengan study kasus ini Saya bisa membuatUntuk menampilkan data kita dapat menggunakan method $lookup dengan aggregation.

Kesimpulan Percobaan 4

Dengan study kasus ini Saya bisa membuat Collection dan juga menambahkan dukumen menggunakan one to one dan many to one, dan juga query untuk menampilkan isi document pada collection

BAB 1

PENUTUP

Kesimpulan

teori MySQL adalah Structured Query Language (SQL) sebagai bahasa interaktif untuk mengelola data. MySQL adalah sistem manajemen basis data relasional (RDBMS) yang menggunakan SQL untuk menjalankan fungsinya..

Saran

Saran saya tidak ada perubahan pembelajaran sama seperti si Semester 1 dan masih masuk di praktikum kali kecuali pada pengisian laporan praktikum database ke word itu yang perlu di tanyakan dan cara pengisian laporan untuk kedepannya.

DAFTAR PUSTAKA

Tuliskan rujukan yang anda gunakan baik website maupun buku seperti contoh dibawah.

1. Tim Asisten Dosen. 2014. Modul 1 Pengenalan Sistem Operasi, Ide Visual C++, Dan Algoritma Pemrograman. Malang: Unversitas Negeri Malang.
2. Program Konversi Suhu (online) http://bondanoky.blogspot.com/2012/10/program-konversi-suhu-c.html. Di akses 8 September.

LAMPIRAN

Berisikan syntax atau gambar yang dibutuhkan dalam tiap pertemuan praktikum.