



# Modul 08

## MongoDB and No-SQL Databases

### TUJUAN PEMBELAJARAN

1. Mahasiswa mampu membandingkan basis data NoSQL dengan basis data SQL tradisional.
2. Mahasiswa mampu membangun database berbasis NoSQL menggunakan MongoDB

### HARDWARE & SOFTWARE

1. PC (*Personal Computer*) dengan akses Internet
2. JavaScript
3. Visual Studio Code atau IDE lainnya yang mendukung JavaScript
4. Node.js
5. MongoDB

### URAIAN MATERI

#### A. Database Server

Database server adalah sistem komputer khusus atau aplikasi perangkat lunak yang dirancang untuk mengelola, menyimpan, mengambil, dan mengatur data dalam database. Hal ini bertindak sebagai tempat penyimpanan pusat untuk data dan memungkinkan beberapa pengguna atau aplikasi untuk berinteraksi dengan data tersebut dengan cara yang terstruktur dan efisien. Server database memainkan peran penting dalam banyak aplikasi perangkat lunak dan sistem informasi, menyediakan solusi penyimpanan dan pengambilan data yang aman dan dapat diperluas. Karakteristik dan fungsi dari server database meliputi:

1. **Penyimpanan Data:** Server database menyimpan data dalam format yang terstruktur, biasanya menggunakan tabel, baris, dan kolom. Penyimpanan terstruktur ini memungkinkan pengajuan pertanyaan dan pengambilan data yang efisien.
2. **Manajemen Data:** menyediakan mekanisme untuk menambah, memperbarui, dan menghapus data, memastikan integritas dan konsistensi data.

3. Pengolahan Permintaan: Server database memproses permintaan dan permintaan dari pengguna atau aplikasi, memungkinkan mereka untuk mengambil data tertentu berdasarkan kriteria yang telah ditentukan (misalnya, permintaan SQL).
4. Kontrol Kesesuaian: Server database mengelola beberapa pengguna atau aplikasi secara bersamaan, memastikan bahwa data diakses dan dimodifikasi dengan terkendali dan koordinasi untuk mencegah kerusakan data dan konflik.
5. Keamanan: mencakup fitur keamanan seperti otentikasi pengguna, kontrol akses, dan enkripsi untuk melindungi data dari akses atau penyusupan yang tidak sah.
6. Indeks Data: Server database membuat dan menjaga indeks untuk mempercepat pengambilan data, terutama untuk dataset besar.
7. Cadangan dan Pemulihan: Server database menawarkan mekanisme pencadangan dan pemulihan data untuk melindungi data dari kerusakan perangkat keras atau masalah lainnya.
8. Optimisasi Kinerja: menggunakan berbagai teknik seperti optimisasi permintaan, caching, dan manajemen transaksi untuk meningkatkan kinerja database.
9. Replikasi Data: Dalam beberapa kasus, server database mendukung replikasi data untuk membuat salinan database untuk keperluan redundansi, penyeimbangan beban, dan pemulihan bencana.

Contoh umum perangkat lunak server database meliputi:

1. Microsoft SQL Server
2. Oracle Database
3. MySQL
4. PostgreSQL
5. MongoDB (basis data NoSQL)
6. IBM Db2
7. SQLite
8. Redis (penyimpanan data dalam memori)

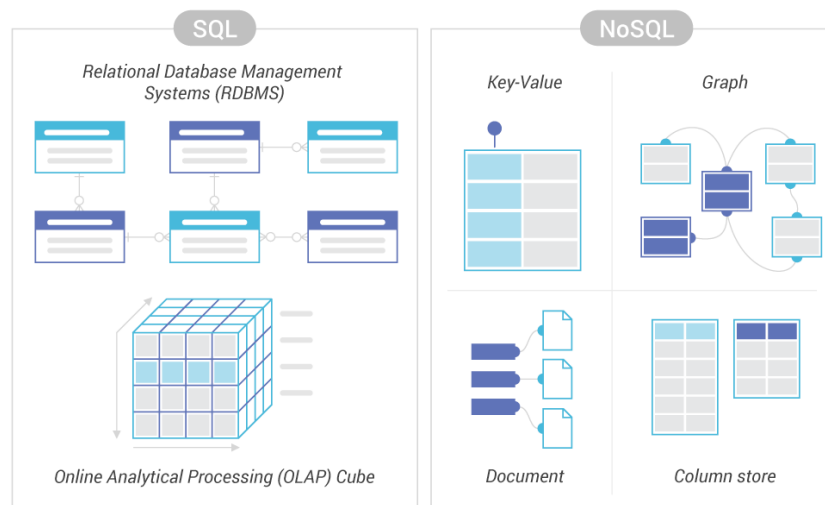
Server-server database ini dapat digunakan dalam berbagai aplikasi, mulai dari sistem manajemen konten sederhana hingga sistem perencanaan sumber daya perusahaan (ERP) yang kompleks dan platform analitik big data. Pemilihan server database tertentu bergantung pada persyaratan aplikasi tertentu, termasuk faktor seperti volume data, kompleksitas, kinerja, dan skalabilitas.

## B. SQL and No-SQL Databases

SQL (Structured Query Language) adalah bahasa pemrograman khusus yang digunakan untuk mengelola dan mengakses database relasional. SQL digunakan untuk berinteraksi dengan database relasional untuk melakukan operasi seperti pengambilan data, pembaruan data, penambahan data, dan penghapusan data. Ini adalah bahasa standar yang digunakan oleh hampir semua database relasional, dan digunakan untuk

menghasilkan, mengelola, dan mengoptimalkan struktur dan konten dalam database tersebut. Relational database adalah jenis database yang terdiri dari tabel dan kolom yang terkait satu sama lain melalui kunci-kunci asing.

Database SQL didasarkan pada model data relasional yang diperkenalkan oleh Edgar F. Codd pada tahun 1970. Dalam database relasional, data disimpan dalam tabel dengan skema yang telah ditentukan sebelumnya. Setiap tabel memiliki kunci utama yang memungkinkan penghubungan antara tabel-tabel tersebut. Contoh database relasional yang terkenal adalah MySQL, PostgreSQL, Oracle Database, SQL Server, dan SQLite.



Gambar 1. Perbedaan Struktur dalam SQL dan NoSQL

Sementara itu, NoSQL (Not Only Structred Query Language) databases adalah jenis database yang berbeda dari database relasional. Database ini dirancang untuk mengatasi beberapa batasan database relasional dan memungkinkan fleksibilitas yang lebih besar dalam penyimpanan dan pengambilan data. NoSQL berfokus pada penyimpanan dan pengambilan data yang tidak sesuai dengan model relasional tradisional. Hal ini mendukung berbagai model data, termasuk dokumen, key-value, kolom, dan grafik, yang membuat mereka lebih fleksibel dalam menangani jenis data yang beragam dan berukuran besar.

Document database adalah salah satu model data yang digunakan dalam lingkungan NoSQL. Dalam document database, data disimpan dalam dokumen, yang biasanya dalam format JSON atau BSON (Binary JSON). Setiap dokumen adalah unit data tunggal dan dapat memiliki struktur yang berbeda dari dokumen lain dalam database. Berikut beberapa perbedaan istilah dalam basisdata SQL dan No-SQL

Tabel 1. Istilah-istilah dalam relational dan document database

Relational DB	Document DB (MongoDB)
Database	Database
Table	Collection
Column	Field
Row, Record	Document (JSON, XML, dan lain-lain)
Join Table	Embedded Document, Reference
SQL	JavaScript (MongoDB)

SQL databases dan NoSQL databases memiliki keunggulan dan kelemahan masing-masing terkait implementasinya dalam membangun aplikasi. Berikut adalah beberapa contoh aplikasi di mana SQL dan NoSQL databases dapat digunakan:

a) SQL (Structure Query Language)

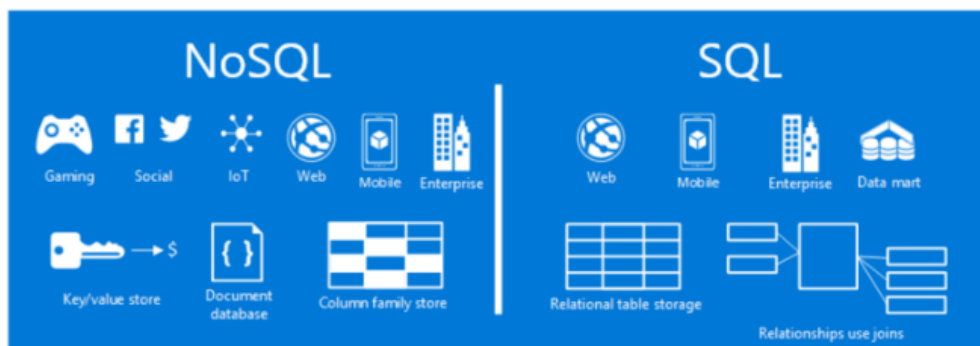
1. Aplikasi Bisnis: SQL databases seperti MySQL, PostgreSQL, dan SQL Server cocok untuk aplikasi bisnis yang memerlukan transaksi yang konsisten dan data dengan skema yang ketat, seperti sistem manajemen inventaris, manajemen pelanggan, dan akuntansi.
2. Aplikasi Keuangan: SQL databases sering digunakan dalam aplikasi keuangan yang memerlukan tingkat keamanan, konsistensi, dan integritas data yang tinggi.
3. Sistem Manajemen Konten (CMS): CMS seperti WordPress, Drupal, dan Joomla menggunakan SQL databases untuk menyimpan konten, pengguna, dan pengaturan aplikasi.
4. Aplikasi Berbasis Analitik: SQL databases digunakan dalam aplikasi analitik untuk mengumpulkan, menyimpan, dan mengambil data yang diperlukan untuk analisis bisnis.
5. Aplikasi yang Memerlukan Relasi Data yang Kuat: SQL databases ideal untuk aplikasi yang memerlukan relasi data yang kompleks, seperti aplikasi e-commerce dengan hubungan produk, pesanan, dan pelanggan.

b) Aplikasi NoSQL (Not Only Structured Query Language)

1. Aplikasi Media Sosial: NoSQL databases seperti MongoDB digunakan dalam aplikasi media sosial untuk mengelola data pengguna, posting, komentar, dan berbagai jenis konten yang bisa memiliki struktur yang berbeda.
2. Aplikasi E-commerce Skala Besar: NoSQL databases cocok untuk aplikasi e-commerce yang perlu menangani jumlah data yang sangat besar dan beragam seperti data produk, penilaian produk, dan catatan pembelian.
3. IoT (Internet of Things): NoSQL databases dapat digunakan untuk mengumpulkan dan menyimpan data dari perangkat IoT yang menghasilkan data dalam volume besar dan dengan format yang bervariasi.

4. Aplikasi Analitik Data Besar (Big Data): NoSQL databases sering digunakan dalam aplikasi yang mengolah dan menganalisis data besar karena kemampuannya untuk diskalakan secara horizontal.
5. Aplikasi yang Memerlukan Skalabilitas Cepat: NoSQL databases sangat berguna untuk aplikasi yang memerlukan skalabilitas horizontal yang cepat, terutama jika pertumbuhan data adalah faktor utama.

Namun, perlu dicatat bahwa tidak selalu harus memilih satu jenis database secara eksklusif. Beberapa aplikasi menggabungkan baik SQL dan NoSQL databases untuk mengatasi berbagai kebutuhan data dalam lingkungan yang sama. Keputusan tentang jenis database yang tepat untuk sebuah aplikasi harus mempertimbangkan kompleksitas aplikasi, struktur data, dan skala yang diperlukan.



Gambar 2. Implementasi NoSQL dan SQL dalam berbagai aplikasi

### C. MongoDB

MongoDB, sebuah database NoSQL yang terkenal, dirancang untuk menyimpan data dalam bentuk dokumen JavaScript Object Notation (JSON). Database ini telah menjadi pilihan yang umum digunakan dalam pengembangan situs web yang digunakan oleh perusahaan-perusahaan besar seperti Google, Adobe, dan eBay. MongoDB juga dirancang dengan teliti untuk mengatasi beberapa tantangan yang dihadapi oleh database relasional tradisional. Hal ini mencakup skalabilitas horizontal, fleksibilitas skema, dan kinerja yang unggul untuk aplikasi yang berurusan dengan volume data besar dan kompleks. MongoDB menyimpan data dalam dokumen yang dapat berisi berbagai jenis data, termasuk string, angka, daftar, dan bahkan dokumen lain. Dokumen-dokumen ini kemudian dikelompokkan ke dalam koleksi, yang setara dengan tabel dalam database relasional tradisional.

Salah satu keunggulan terbesar MongoDB adalah kemampuannya untuk skalabilitas horizontal, yang memungkinkan penambahan server lebih mudah untuk meningkatkan kinerja dan kapasitas penyimpanan. Selain itu, MongoDB juga menyediakan berbagai fitur untuk manajemen data, termasuk indeks, replikasi, dan sharding. Komponen-komponen kunci dalam sistem database MongoDB meliputi:

- Basis data, yang berfungsi sebagai wadah dengan struktur penyimpanan yang disebut koleksi.
- Koleksi, yang merupakan tempat penyimpanan data dalam bentuk dokumen.
- Dokumen, unit terkecil dalam struktur data MongoDB.

Berikut contoh data dalam MongoDB

```
{
  "_id": ObjectId("653be4158f34d12b7d8c5b98"),
  "nama": "Randi",
  "usia": 26
}
```

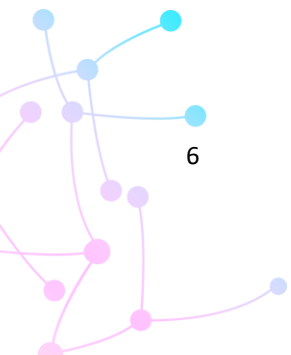
Dokumen diatas memiliki beberapa atribut yang memiliki makna berikut:

1. "\_id": Ini adalah atribut khusus yang biasanya digunakan untuk menyimpan identifikasi unik untuk setiap dokumen dalam koleksi MongoDB. Nilai "\_id" biasanya dibuat oleh database dan berfungsi sebagai kunci utama atau identifikasi unik. Dalam contoh ini, "\_id" adalah nilai ObjectId yang unik.
2. "nama": Ini adalah atribut yang menyimpan nama seseorang. Dalam contoh ini, nama yang disimpan adalah "Randi."
3. "usia": Ini adalah atribut yang menyimpan usia seseorang. Dalam contoh ini, usia yang disimpan adalah 26 tahun.

Setiap data pada database berbasis MongoDB di identifikasi dengan **ObjectId()** yang unik. ObjectId terdiri dari 12 byte memiliki struktur sebagai berikut:

1. Timestamp 4 byte, yang merepresentasikan waktu penciptaan ObjectId dalam detik sejak awal epoch Unix.
2. Nilai acak 5 byte yang dihasilkan satu kali per proses. Nilai acak ini bersifat unik untuk setiap mesin dan proses.
3. Counter (pencacah) 3 byte yang terus bertambah, diinisialisasi dengan nilai acak.

Pada nilai timestamp dan counter, byte yang memiliki signifikansi tertinggi muncul pertama dalam urutan byte (big-endian). Hal ini berbeda dengan nilai BSON (Binary JSON) lainnya, di mana byte yang memiliki signifikansi paling rendah muncul pertama (little-endian). Jika sebuah nilai bilangan bulat digunakan untuk menciptakan ObjectId, nilai bilangan bulat tersebut akan menggantikan nilai timestamp.

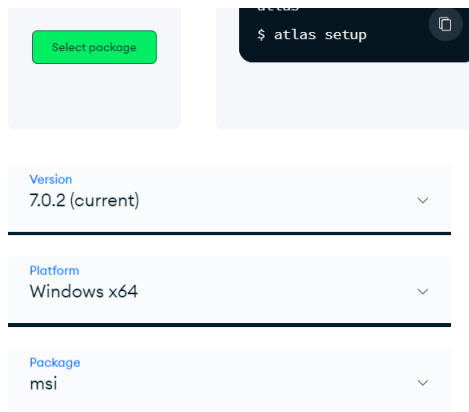


## LATIHAN

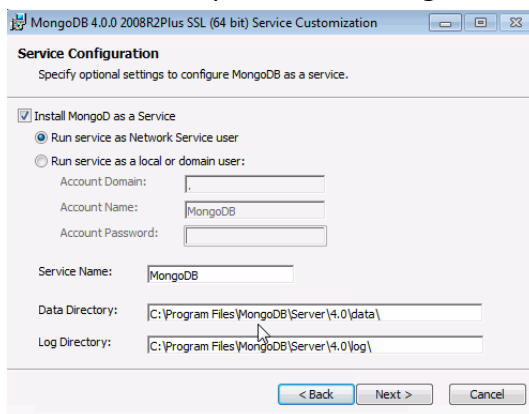
---

### 1. Instalasi MongoDB

- Silakan download **MongoDB Community Server** melalui link berikut ini <https://www.mongodb.com/try/download/community>
- Silakan klik select package hingga muncul seperti gambar berikut ini. Silakan pilih versi terbaru dan sesuaikan tipe sistem operasi anda. Pada **package**, pastikan anda memilih **msi** untuk memudahkan instalasi. Kemudian klik tombol **Download**

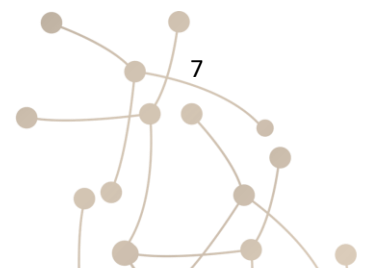


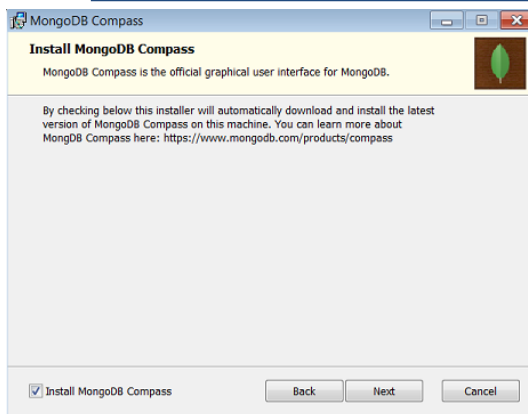
- Setelah selesai download, silakan lakukan instalasi. Jika muncul tampilan seperti gambar berikut, silakan pilih Install MongoD as a service



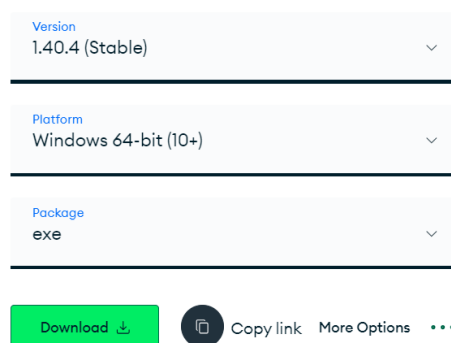
Keterangan lebih lanjut terkait ini, bisa dibaca melalui link berikut <https://www.mongodb.com/docs/manual/tutorial/install-mongodb-on-windows/>

- Jika muncul tampilan berikut, silakan centan **Install MongoDB Compass**, lalu klik **Next**





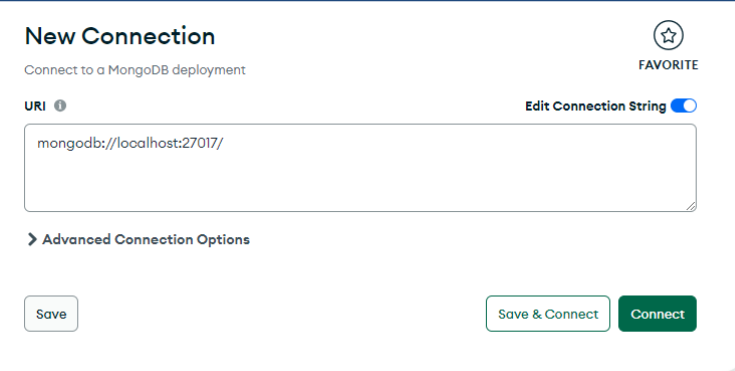
- e. MongoDB Compass adalah tools berbasis GUI yang berguna untuk manajemen database MongoDB dengan mudah. Jika anda terlanjur melewati bagian tersebut d, silakan download MongoDB Compass melalui link berikut ini <https://www.mongodb.com/try/download/compass>
- f. Pastikan anda mendownload versi terbaru dan memilih package .exe



- g. Download dan lakukan proses instalasi hingga selesai
- h. Alternatif dari MongoDB Compass adalah **Studio 3T** dengan fitur yang lebih lengkap dan dapat didownload melalui link berikut <https://studio3t.com/>
- i. Untuk memastikan bahwa MongoDB server anda telah terinstall dengan baik dan telah berjalan, silakan buka **Windows Task Manager** (tekan **Ctrl + Shift + Esc**) dan perhatikanlah pada bagian **Processes** bahwa **MongoDB Database Server** telah berjalan.
- j. Jika belum berjalan, silakan klik **Services**, lalu carilah **MongoDB**, klik **Kanan**, lalu Pilih **Start**

## 2. Koneksi Ke Database Dan Memasukan Data Dokumen (INSERT)

- a. **PENTING!** Semua dokumentasi dan penjelasan terkait perintah-perintah MongoDB yang akan digunakan dalam praktik ini telah tersedia melalui link berikut <https://mongodb.github.io/node-mongodb-native/6.2/>
- b. Bukalah aplikasi **MongoDB Compass** anda dan ketikkan **mongodb://localhost:27017** pada bagian **URI** seperti gambar dibawah ini. Lalu klik **Connect**



The screenshot shows the 'New Connection' window in MongoDB Compass. It has a title bar 'New Connection' and a subtitle 'Connect to a MongoDB deployment'. There is a 'FAVORITE' icon in the top right. The 'URI' field is populated with 'mongodb://localhost:27017/'. Below the URI field is a link to 'Advanced Connection Options'. At the bottom, there are three buttons: 'Save', 'Save & Connect', and 'Connect'.

- Biarkan aplikasi tersebut tetap terbuka. Lalu, buatlah **folder baru** dengan nama **task-manager** pada visual studio code anda.
- Buka terminal pada visual studio code anda dan ketikkan **npm init -y** untuk meng-generate file package.json
- Lakukan **instalasi library** mongodb dengan perintah **npm i mongodb@6.2.0**. Berikut adalah link library mongodb <https://www.npmjs.com/package/mongodb>
- Setelah itu, buatlah **file JavaScript baru** dengan nama **insertDocument.js** dalam folder task-manager
- Ketikkanlah *source code* berikut ini pada file **insertDocument.js** untuk melakukan koneksi dan memasukkan data (*insert*) ke database mongodb

```
// Mengimport modul MongoClient dan ObjectId dari 'mongodb'.
const { MongoClient, ObjectId } = require('mongodb');

// Mendefinisikan URL MongoDB server yang akan digunakan untuk koneksi.
const url = 'mongodb://127.0.0.1:27017';

// Membuat instance MongoClient dengan URL koneksi yang telah didefinisikan sebelumnya.
const client = new MongoClient(url);

// Mendefinisikan nama database yang akan digunakan.
const namaDatabase = 'testsaja';

// Membuat instance ObjectId baru. ObjectId digunakan untuk menghasilkan unik identifier untuk dokumen MongoDB.
const id = new ObjectId();

//BAGIAN INI MENCETAK INFORMASI DARI ObjectId()
// Mencetak ObjectId yang baru dibuat ke konsol.
console.log(id);

// Mencetak representasi hexadecimal dari ObjectId ke konsol.
console.log(id.id);

// Mencetak panjang (jumlah karakter) dari representasi hexadecimal ObjectId ke konsol.
console.log(id.id.length);
```

```
// Mencetak timestamp yang terkait dengan ObjectId ke konsol. Kode ini akan
memberikan data waktu kapan ObjectId tersebut dibuat.
console.log(id.getTimestamp());

// Mencetak panjang dari representasi ObjectId dalam bentuk string
heksadesimal.
console.log(id.toHexString().length);

// BAGIAN INI ADALAH FUNGSI UTAMA YANG BERJALAN SECARA ASYNCHRONOUS
// Mendefinisikan fungsi async 'main' untuk melakukan operasi-operasi terkait MongoDB.
async function main() {
  try {
    //BAGIAN INI TERKAIT KONEKSI KE DATABASE DAN MEMASUKAN DATA
    // Menggunakan 'await' untuk menghubungkan ke server MongoDB.
    await client.connect();
    console.log('Berhasil terhubung ke MongoDB database server');

    // Memilih database dengan nama 'task-manager' yang telah didefinisikan sebelumnya.
    const db = client.db(namaDatabase);

    // Memilih koleksi 'pengguna' di dalam database.
    const clPengguna = db.collection('pengguna');

    // Memilih koleksi 'tugas' di dalam database.
    const clTugas = db.collection('tugas');

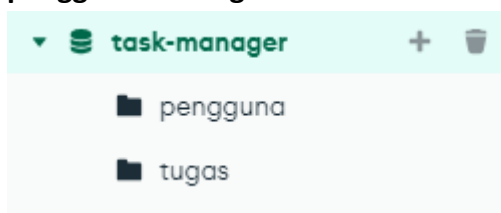
    // MEMASUKAN SATU DATA (DOKUMEN)
    // Memasukkan dokumen ke dalam koleksi 'pengguna'.
    const insertPengguna = await clPengguna.insertOne({
      _id: id,
      nama: 'Randi',
      usia: 25
    });
    console.log('Memasukkan data Pengguna ke koleksi =>', insertPengguna);

    // MEMASUKAN BANYAK DATA (DOKUMEN)
    // Memasukkan beberapa dokumen ke dalam koleksi 'tugas'.
    const insertTugas = await clTugas.insertMany([
      {
        Deskripsi: 'Membersihkan rumah',
        StatusPenyelesaian: true
      }, {
        Deskripsi: 'Mengerjakan tugas kuliah',
        StatusPenyelesaian: false
      }, {
        Deskripsi: 'Memberikan bimbingan',
        StatusPenyelesaian: false
      }
    ]);
    console.log('Memasukkan data Tugas ke koleksi =>', insertTugas);
    // Mengembalikan pesan sukses setelah operasi selesai.
    return 'Data selesai dimasukkan.';
  }
}
```

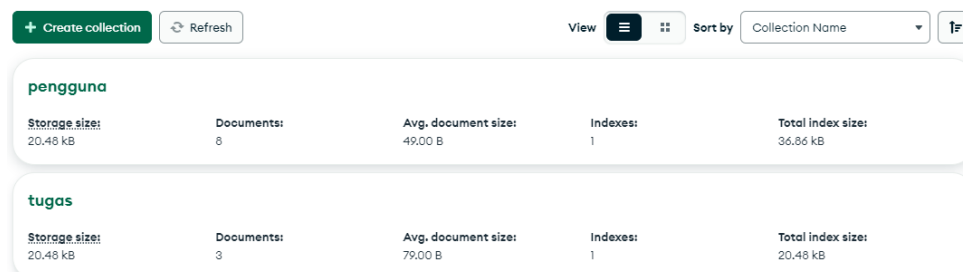
```
// BAGIAN INI MENANGANI ERROR
} catch (err) {
  // Menangani kesalahan dengan mencetak pesan kesalahan ke konsol.
  console.error(err);
} finally {
  // Selalu menutup koneksi ke server MongoDB setelah operasi selesai, baik
  // sukses maupun gagal.
  client.close();
}
}

// Memanggil fungsi 'main' dan menangani hasilnya menggunakan 'then' dan
// 'catch' untuk mencetak hasil atau pesan kesalahan ke konsol.
main().then(console.log).catch(console.error);
```

- h. Jalankan coding tersebut dengan mengetikkan perintah **node insertDocument.js** dan perhatikan apa yang ditampilkan pada terminal
- i. Lalu, silakan buka **MongoDBCompass** anda dan perhatikanlah bahwa **database** terbaru dengan nama **task-manager** telah dibuat lengkap dengan **koleksi** yang diberi nama **pengguna** dan **tugas**

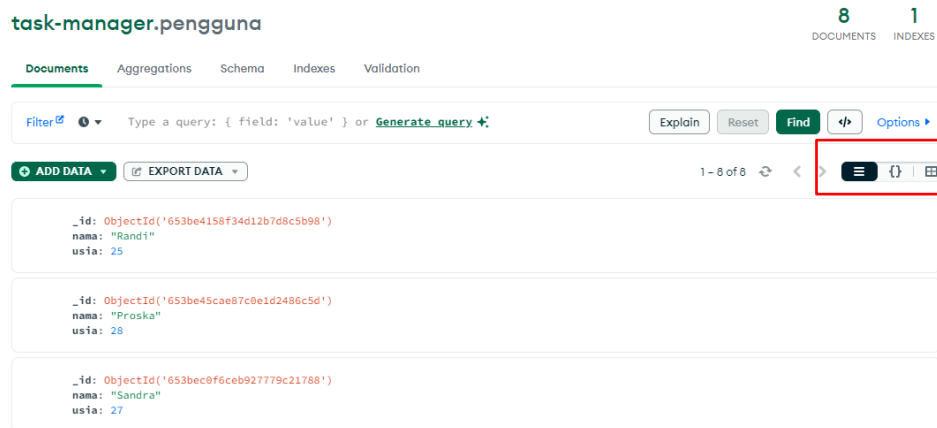


- j. Anda dapat mengklik database tersebut agar muncul tampilan seperti berikut ini

A screenshot of the MongoDB Compass interface showing the details of two collections. The top collection is 'pengguna' and the bottom is 'tugas'. Each collection has a table of statistics including storage size, number of documents, average document size, number of indexes, and total index size.

Collection Name	Storage size	Documents	Avg. document size	Indexes	Total index size
pengguna	20.48 kB	8	49.00 B	1	36.86 kB
tugas	20.48 kB	3	79.00 B	1	20.48 kB

- k. Kliklah salah satu koleksi, misalnya '**pengguna**' untuk melihat data yang telah anda masukan. Anda dapat mengubah tampilan data dengan mengklik **ikon yang diberi lingkaran merah** pada gambar dibawah ini. Pilihlah tampilan yang paling nyaman bagi anda.



### 3. Query Dokumen (READ)

- a. Buatlah file JavaScript baru dengan nama **readDocument.js**, lalu masukkanlah source code berikut ini

```
const { MongoClient, ObjectId } = require('mongodb');
const url = 'mongodb://127.0.0.1:27017';
const client = new MongoClient(url);
const namaDatabase = 'task-manager';

async function main() {
  try {
    await client.connect();
    console.log('Berhasil terhubung ke MongoDB database server');
    const db = client.db(namaDatabase);

    // Mencari satu dokumen dalam koleksi 'pengguna' berdasarkan nama 'Randi'.
    const byNama = await db.collection('pengguna').findOne({ nama: 'Randi' });

    // Mencari satu dokumen dalam koleksi 'pengguna' berdasarkan ID objek tertentu.
    const byObjectID = await db.collection('pengguna').findOne({ _id: new ObjectId('653bec0f6ceb927779c21788') });

    // Mencari beberapa dokumen dalam koleksi 'pengguna' dengan kriteria usia 28 dan mengubahnya menjadi array.
    const toArray = await db.collection('pengguna').find({ usia: 28 }).toArray();

    // Menggunakan if statement dengan kondisi yang salah. (Ini tidak akan berfungsi sebagaimana yang diharapkan)
    if (byNama && byObjectID && toArray) {
      // Menampilkan hasil pencarian berdasarkan nama, ID objek, dan kriteria usia.
      console.log('Data Pengguna ditemukan (berdasarkan nama):', byNama);
      console.log('Data Pengguna ditemukan (berdasarkan ID Objek):', byObjectID);
      console.log('Data Pengguna ditemukan (dalam format Array):', toArray);
    } else {
      // Menampilkan pesan bahwa data pengguna tidak ditemukan.
    }
  } catch (err) {
    console.error(err);
  }
}
```

```
        console.log('Data Pengguna tidak ditemukan');
    }
} catch (err) {
    console.error(err);
} finally {
    await client.close();
}
}

// Memanggil fungsi 'main' dan menangani kesalahan (jika ada) dengan mencetak
pesan kesalahan ke konsol.
main().catch(console.error);
```

- Silakan ganti teks yang berwarna kuning dengan data **nama** yang ingin anda cari
- Ganti juga teks warna merah dengan **ID objek** yang ingin anda cari. ID objek ini dapat ditemukan pada data anda di MongoDBCompass. **Perhatikanlah** gambar pada tutorial sebelumnya terkait Insert Document bagian j
- Lalu, ganti juga teks warna biru dengan data **usia** yang ingin anda cari
- Jalankan kode diatas dengan perintah **node readDocument.js** dan **perhatikan** apa yang ditampilkan pada terminal

#### 4. Memperbaharui Dokumen (UPDATE)

- Buatlah file baru dalam folder task-manager anda dan beri nama **updateDocument.js**, lalu masukanlah source code berikut ini

```
const { MongoClient, ObjectId } = require('mongodb');

const url = 'mongodb://127.0.0.1:27017';
const client = new MongoClient(url);
const namaDatabase = 'task-manager';

async function main() {
    try {
        await client.connect();
        console.log('Berhasil terhubung ke MongoDB database server');
        const db = client.db(namaDatabase);

        // //Memperbaharui Data dengan perintah updateOne
        const updateOnePromise = db.collection('pengguna').updateOne(
            { _id: new ObjectId('553be4158f34d12b7d8c5b98') },
            { $set: { nama: 'Randikun' } },
            // { $inc: { usia: 1 } }
        )
        updateOnePromise.then((result) => {
            console.log(result);
        }).catch((error) => {
```

```
        console.error(error);
    }).finally(() => {
        client.close();
    });

    // //Memperbaharui Data dengan perintah updateMany

    // db.collection('tugas').updateMany(
    //   { StatusPenyelesaian: false },
    //   { $set: { StatusPenyelesaian: true} }
    // ).then((result) => {
    //   console.log(result.modifiedCount);
    // }).catch((error) => {
    //   console.error(error);
    // }).finally(() => {
    //   client.close();
    // });
  } catch (error) {
    console.error(error);
  }
}

main();
```

- b. Gantilah **teks warna kuning** dengan salah satu **objectId** data anda dalam collection **pengguna**
- c. Gantilah **teks warna merah** dengan nama baru yang ingin anda perbaharui
- d. Jalankan kode diatas dengan mengetikan perintah **node updateDocument.js** dan perhatikanlah bahwa terminal akan menampilkan pesan seperti gambar berikut ini.

```
Berhasil terhubung ke MongoDB database server
{
  acknowledged: true,
  modifiedCount: 1,
  upsertedId: null,
  upsertedCount: 0,
  matchedCount: 1
}
```

- e. **ModifiedCount** ditandai dengan **angka 1** menunjukkan bahwa 1 data telah diubah. Silakan cari tau makna dari masing-masing item yang lainnya (*acknowledged*, *upsertedId*, *upsertedCount* dan *matchedCount*) melalui internet untuk memperdalam pemahaman anda
- f. Setelah mengecek terminal anda, silakan buka juga aplikasi MongoDBCompass anda dan perhatikanlah bahwa data telah berubah

	_id ObjectId	nama String	usia Int32
1	ObjectId('653be4158f34d12b7d8...	"Randikun"	25

- g. Selanjutnya silakan jadikan baris kode berikut `{ $set: { nama: 'Randikun' } }`, menjadi komentar. Lalu uncomment lah baris yang mengandung teks `$inc`.
- h. Jalankan kembali file `updateDocument.js`, lalu perhatikan apa yang ditampilkan pada terminal dan hasilnya data usia telah berubah pada aplikasi MongoDBCompass.

1	<code>ObjectId('653be4158f34d12b7d8...</code>	<code>"Randikun"</code>	<code>26</code>
---	---	-------------------------	-----------------

- i. Selanjutnya jadikan semua baris kode yang terkait dengan memperbaharui dengan perintah **updateOne** menjadi **komentar**, lalu **uncommentlah** semua baris kode yang terkait **updateMany**
- j. Jalankan kembali file `updateDocument.js` dan perhatikan apa yang ditampilkan pada terminal
- k. Lalu cek data anda pada MongoDBCompass, perhatikanlah semua data yang terkait **StatusPenyelesaian** telah berubah menjadi **True** dari yang sebelumnya **false**
- l. **CHALLENGE:** Perbaharui semua data pada collection **pengguna** dan buatlah semua data menjadi unik agar tidak ada satupun data yang sama baik **usia** maupun **nama**. Berikut adalah contoh tampilan beberapa data yang sama

🏠 pengguna			
	_id ObjectId	nama String	usia Int32
1	<code>ObjectId('653be4158f34d12b7d8...</code>	<code>"Randikun"</code>	<code>26</code>
2	<code>ObjectId('653be45cae87c0e1d24...</code>	<code>"Proska"</code>	<code>28</code>
3	<code>ObjectId('653bec0f6ceb927779c...</code>	<code>"Sandra"</code>	<code>27</code>
4	<code>ObjectId('653bec0f6ceb927779c...</code>	<code>"Dr. While"</code>	<code>35</code>
5	<code>ObjectId('653bec0f6ceb927779c...</code>	<code>"Dr. Node"</code>	<code>46</code>
6	<code>ObjectId('653bec43989bb78bae6...</code>	<code>"Sandra"</code>	<code>26</code>
7	<code>ObjectId('653bec43989bb78bae6...</code>	<code>"Randi"</code>	<code>35</code>
8	<code>ObjectId('653bec43989bb78bae6...</code>	<code>"Randi"</code>	<code>28</code>

## 5. Menghapus Dokumen (DELETE)

- a. Silakan buat file baru dalam folder task-manager anda dengan nama **deleteDocument.js**, lalu masukanlah kode berikut ini

```
const { MongoClient, ObjectId } = require('mongodb');

const url = 'mongodb://127.0.0.1:27017';
const client = new MongoClient(url);
const namaDatabase = 'task-manager';

async function main() {
  try {
    await client.connect();
    console.log('Berhasil terhubung ke MongoDB database server');
    const db = client.db(namaDatabase);

    db.collection('pengguna').deleteMany({
```

```
    usia: 28
  }).then((result) => {
    console.log(result);
  }).catch((error) => {
    console.error(error);
  })
} catch (error) {
  console.error(error);
}
}
main();
```

- b. Jalankan kode tersebut dengan mengetikkan perintah **node deleteDocument.js** pada terminal
- c. Baris kode diatas digunakan untuk menghapus semua data **pengguna** yang **berusia 28**. **Perhatikanlah** apa yang ditampilkan pada terminal ketika anda menjalankan kode tersebut.

```
Berhasil terhubung ke MongoDB database server
{ acknowledged: true, deletedCount: 2 }
```

- d. Setelah itu, silakan cek data anda pada aplikasi **MongoDBCompass** dan perhatikanlah data apa yang telah dihapus. Jika anda masih bingung, silakan ganti angka **28** dengan data yang **paling banyak muncul** dalam data anda.

	_id ObjectId	nama String	usia Int32
1	ObjectId('653be4158f34d12b7d8...')	"Randikun"	26
2	ObjectId('653bec0f6ceb927779c...')	"Sandra"	27
3	ObjectId('653bec0f6ceb927779c...')	"Dr. While"	35
4	ObjectId('653bec0f6ceb927779c...')	"Dr. Node"	46
5	ObjectId('653bec43989bb78bae6...')	"Sandra"	26
6	ObjectId('653bec43989bb78bae6...')	"Randi"	35

- e. **CHALLENGE:** Silakan tambahkan kode yang menggunakan perintah **deleteOne** untuk menghapus salah satu data **tugas** pada database task-manager anda.

## REFERENCES

1. Mead, A., & Percival, R. (2019, March). The Complete Node.js Developer Course (3rd ed.) [Oreilly.com]. Packt Publishing.
2. Stevens, R. W., Fenner, B., Rudoff, A. M., & Stevens, W. R. (2003, November 14). UNIX Network Programming: The Sockets Networking API. <https://doi.org/10.1604/9780131411555>
3. *What is MongoDB? — MongoDB Manual*. (n.d.). <https://www.mongodb.com/docs/manual/>
4. *Node MongoDB Native Driver Documentation*. (n.d.). <https://mongodb.github.io/node-mongodb-native/6.2/>