Full Stack Engineer

Module 3.1: Intro to MongoDB

Prerequisite

Generasi Gigih

Install MongoDB Community Edition

Ensure that you have installed MongoDB Community Edition before the classroom started. Follow instructions in the pages below:

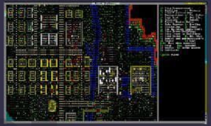
- Linux
- MacOS
- Windows

Generasi Gigih

Before we begin... let's start with a question: what is a database?

Generasi Gigih

Database Alignment Chart

	Access Purist (must be queryable with a query language)	Access Neutral (must be queryable with a language)	Access Radical (queryable in any way)
Function Purist (must contain digital data)	PostgresSQL is a database	Excel is a database	 Dwarf Fortress is a database
Function Neutral (must contain information)	A library is a database	A senior engineer is a database	A file cabinet is a database
Function Radical (can contain anything)	Battleship is a database	Subway checkout counter is a database	A fridge is a database

Generasi Gigih

MongoDB

A document-oriented, general purpose database.

Generasi Gigih

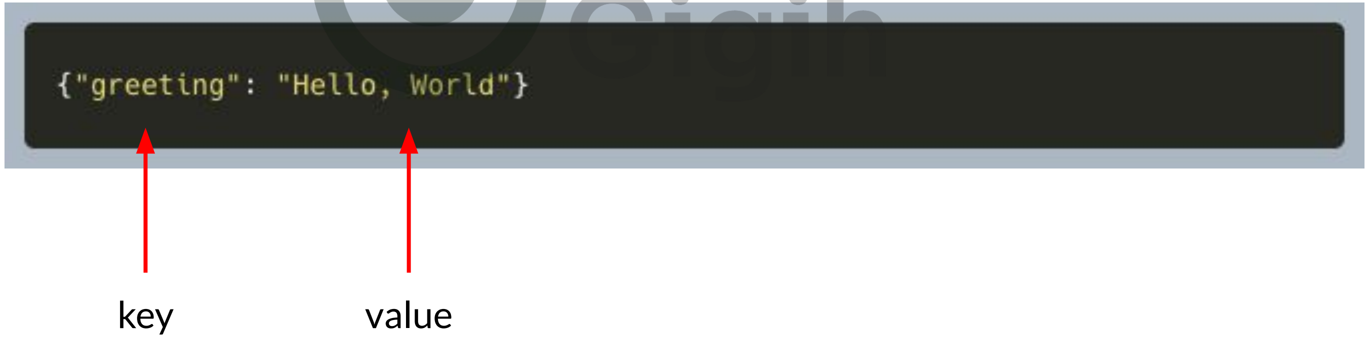
Documents (1)

- A document is the basic unit of data for MongoDB
- An ordered set of keys with associated values
- Representation in programming languages varies: map, hash, dictionary
- In Javascript, documents are represented as objects

Generasi Gigih

Documents (2)

An example of a simple document:

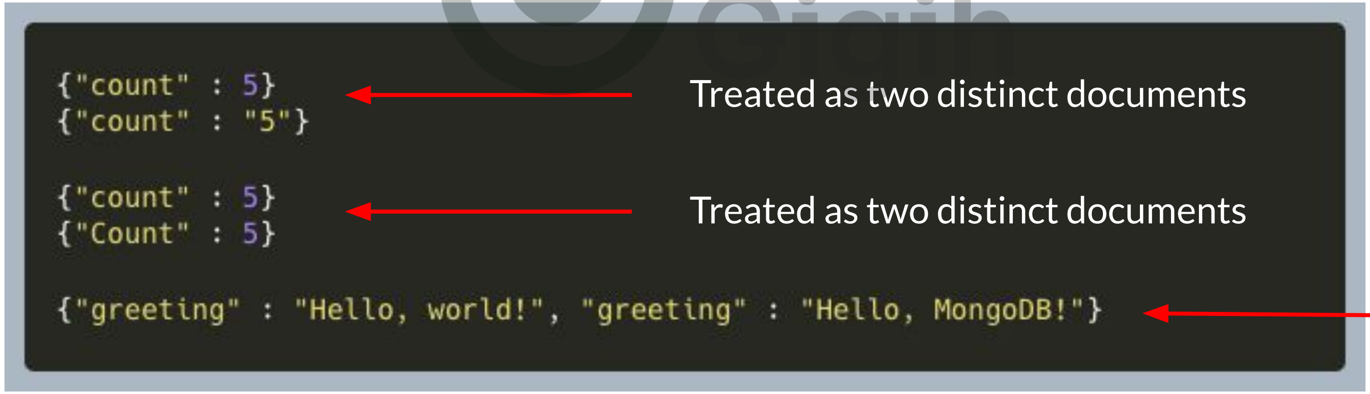


```
{"greeting": "Hello, World"}
```

Generasi Gigih

Documents (3)

Type-sensitive, case-sensitive, can't contain duplicate keys



```
{"count" : 5}
{"count" : "5"}

{"count" : 5}
{"Count" : 5}

{"greeting" : "Hello, world!", "greeting" : "Hello, MongoDB!"}
```

Treated as two distinct documents

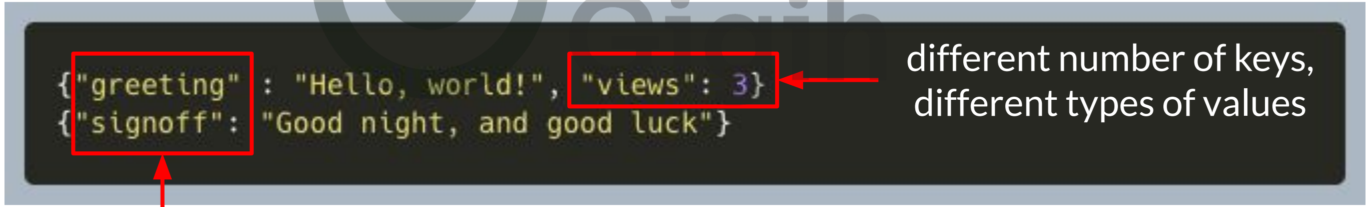
Treated as two distinct documents

not a legal document

Generasi Gigih

Collections (1)

A group of documents with dynamic schemas:



```
{"greeting" : "Hello, world!", "views": 3}
{"signoff": "Good night, and good luck"}
```

different number of keys, different types of values

different keys

Generasi Gigih

Collections (2)

Naming:

- Empty string ("") is not a valid collection name
- Collection names may not contain the character \0 (the null character)
- You should not create any collections with names that start with “system.”
- Collections should not contain the reserved character $ in their names

Generasi Gigih

Subcollections

One convention for organizing collections is to use namespaced subcollections separated by the "." character.

For example, an application containing a blog might have a collection named "blog.posts" and a separate collection named "blog.authors". This is for organizational purposes only—there is no relationship between the blog collection (it doesn't even have to exist) and its subcollections.

Generasi Gigih

Databases (3)

In addition to grouping documents by collection, MongoDB groups collections into databases.

A single instance of MongoDB can host several databases, each grouping together zero or more collections.

Generasi Gigih

Databases (3)

Naming:

- The empty string (“”) is not a valid database name.
- A database name cannot contain any of these characters: /, \, ., ", *, <, >, :, |, ?, $, (a single space), or \0 (the null character)
- Database names are case-insensitive
- Database names are limited to a maximum of 64 bytes

Generasi Gigih

Document-Oriented vs Relational Database

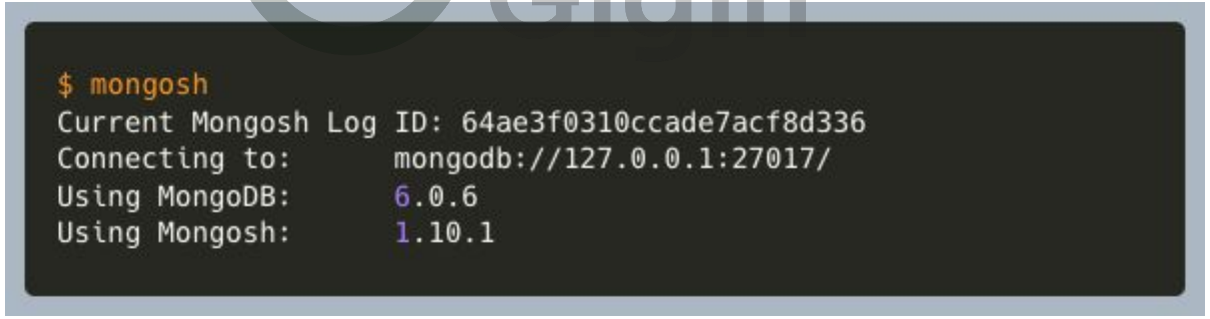
Document-Oriented Database	Relational Database
Document	Row
Collection	Table
Database	Database

Generasi Gigih

MongoDB Shell

Generasi Gigih

A full-featured JavaScript interpreter, capable of running arbitrary JavaScript programs

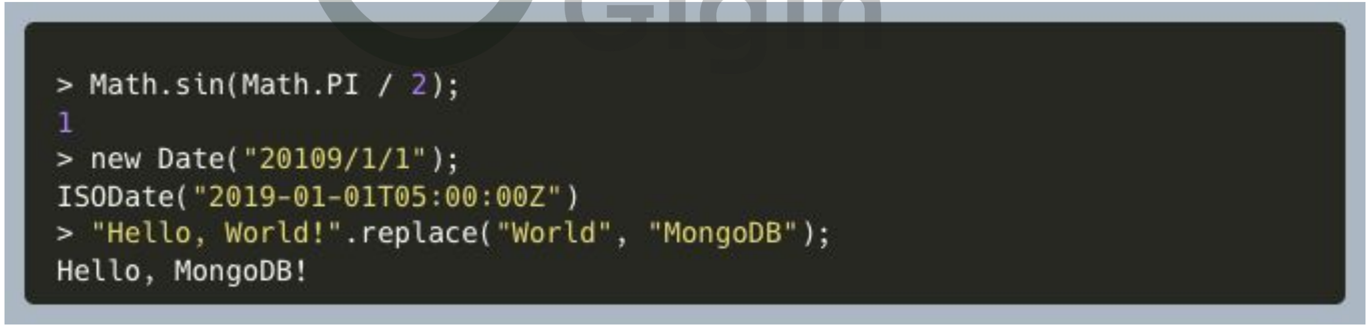


```
$ mongosh
Current Mongosh Log ID: 64ae3f0310ccade7acf8d336
Connecting to:          mongodb://127.0.0.1:27017/
Using MongoDB:          6.0.6
Using Mongosh:          1.10.1
```

MongoDB Shell

Generasi Gigih

Example of running standard Javascript libraries in MongoDB shell:

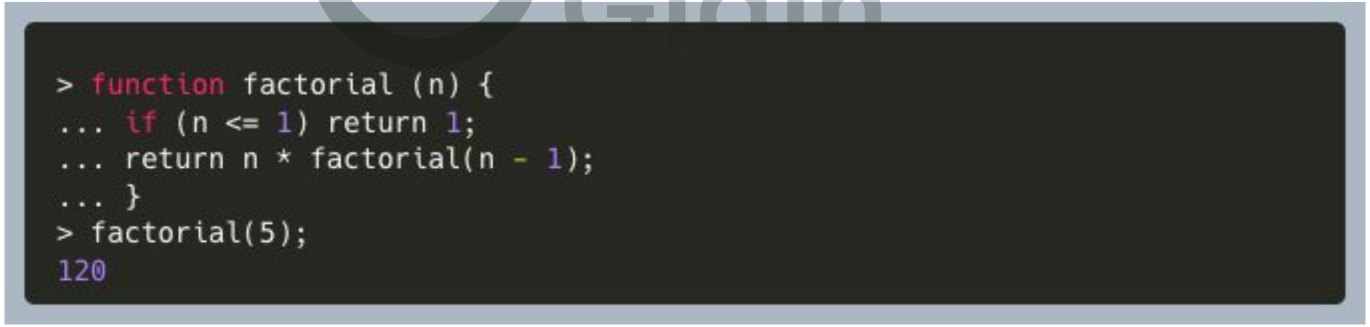


```
> Math.sin(Math.PI / 2);
1
> new Date("20109/1/1");
ISODate("2019-01-01T05:00:00Z")
> "Hello, World!".replace("World", "MongoDB");
Hello, MongoDB!
```

MongoDB Shell

Generasi Gigih

Example of defining and calling a Javascript function in MongoDB shell:

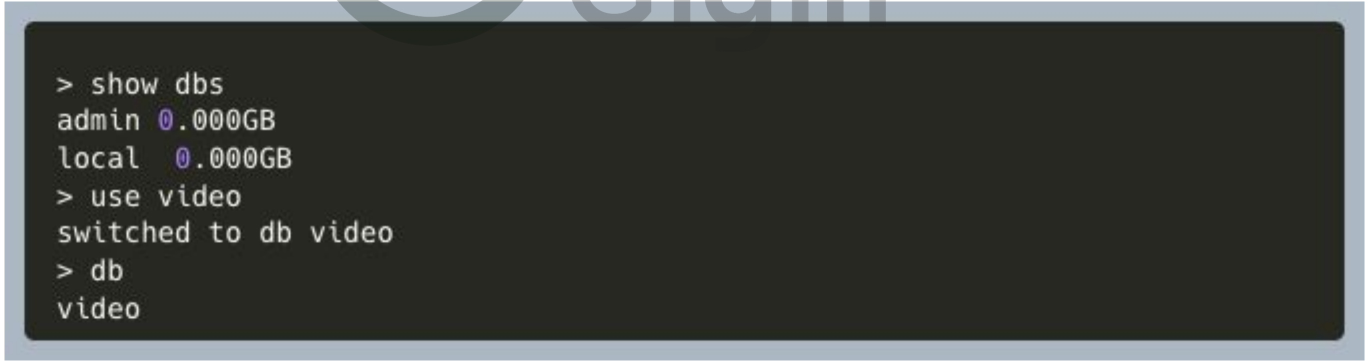


```
> function factorial (n) {
... if (n <= 1) return 1;
... return n * factorial(n - 1);
... }
> factorial(5);
120
```

Basic Operations - Create a Database

Generasi Gigih

No “create” command in MongoDB, to create a database, simply switch context to a non-existing database:

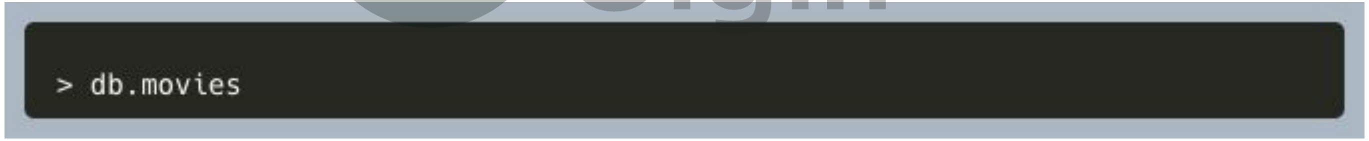


```
> show dbs
admin 0.000GB
local  0.000GB
> use video
switched to db video
> db
video
```

Basic Operations - Create a Collection

Generasi Gigih

While there is a command to create a collection in MongoDB, we can also simply use a non-existent collection to create one:

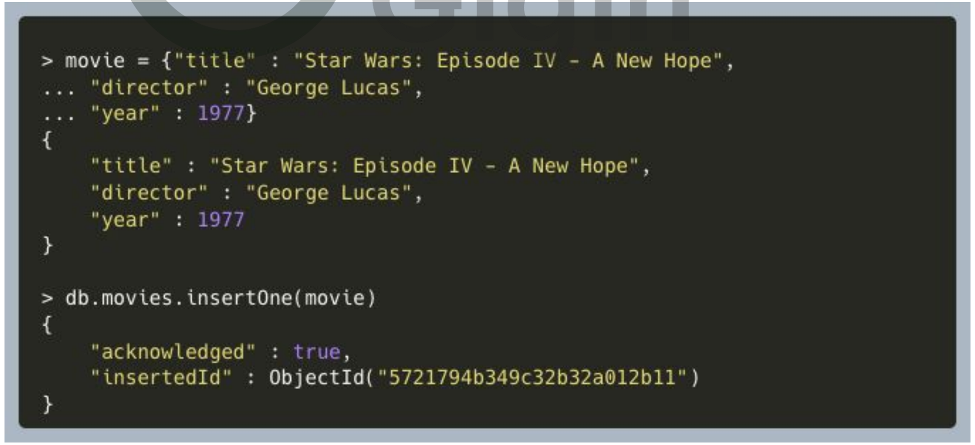


```
> db.movies
```

Basic Operations - Create a Collection

Generasi Gigih

The insertOne function adds a document to a collection. For example, suppose we want to store a movie. First, we'll create a local variable called movie that is a JavaScript object representing our document. It will have the keys "title", "director", and "year" (the year it was released):



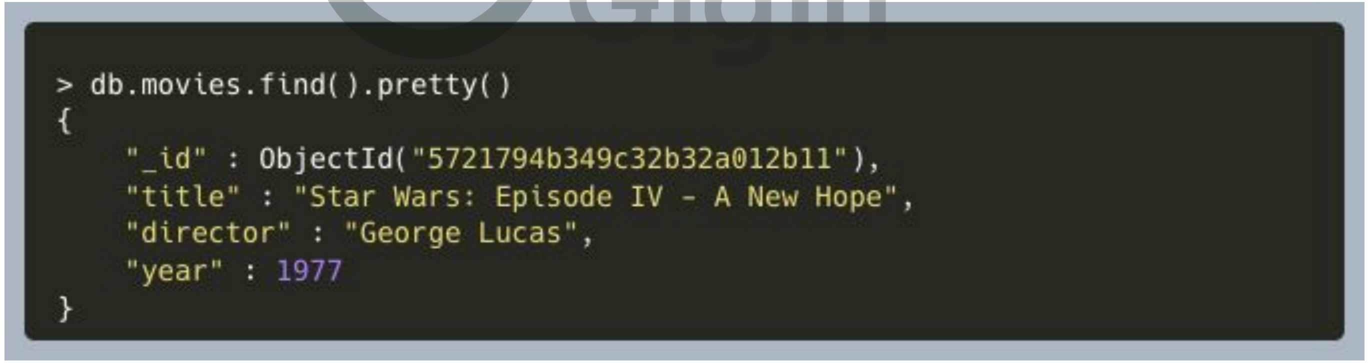
```
> movie = {"title" : "Star Wars: Episode IV - A New Hope",
... "director" : "George Lucas",
... "year" : 1977}
{
    "title" : "Star Wars: Episode IV - A New Hope",
    "director" : "George Lucas",
    "year" : 1977
}

> db.movies.insertOne(movie)
{
    "acknowledged" : true,
    "insertedId" : ObjectId("5721794b349c32b32a012b11")
}
```

Basic Operations - Read a Collection

Generasi Gigih

The find function reads an entire collection. The following example can be used to read the entire collection of movies:

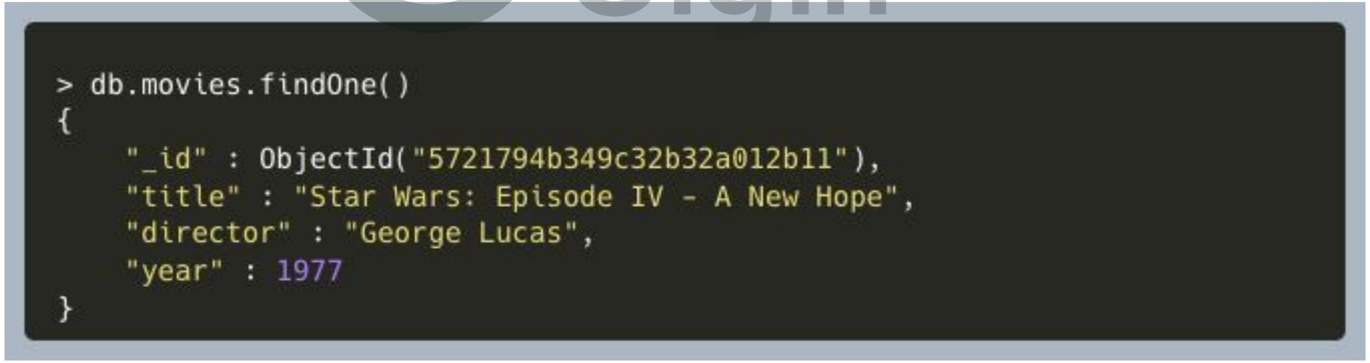


```
> db.movies.find().pretty()
{
    "_id" : ObjectId("5721794b349c32b32a012b11"),
    "title" : "Star Wars: Episode IV - A New Hope",
    "director" : "George Lucas",
    "year" : 1977
}
```

Basic Operations - Read a Document

Generasi Gigih

The findOne function reads a document from a collection. The following snippet will get us the first document in movies collection:

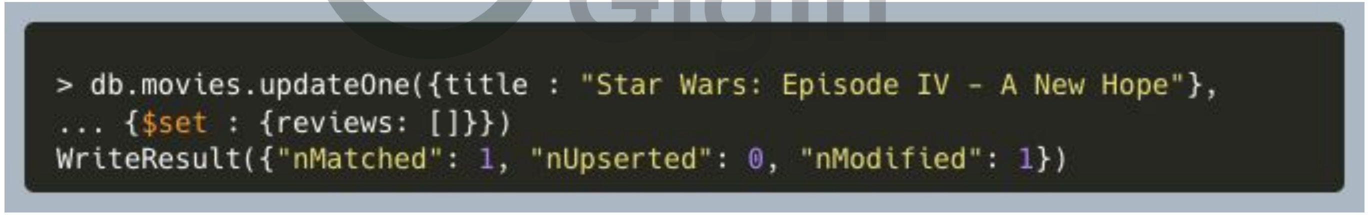


```
> db.movies.findOne()
{
    "_id" : ObjectId("5721794b349c32b32a012b11"),
    "title" : "Star Wars: Episode IV - A New Hope",
    "director" : "George Lucas",
    "year" : 1977
}
```

Basic Operations - Update a Document (1)

Generasi Gigih

To update a document, we can use updateOne function:

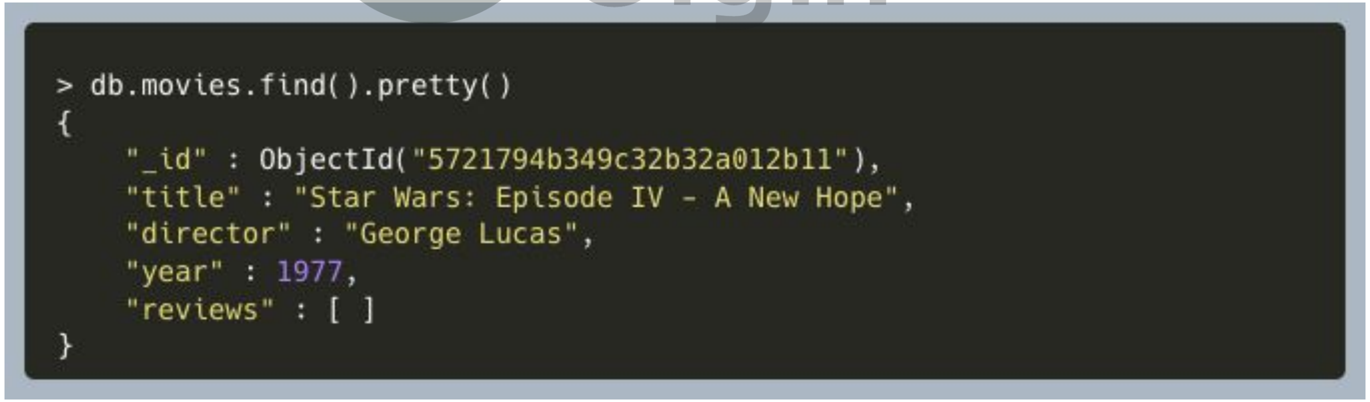


```
> db.movies.updateOne({title : "Star Wars: Episode IV - A New Hope"},
... {$set : {reviews: []}})
WriteResult({"nMatched": 1, "nUpserted": 0, "nModified": 1})
```

Basic Operations - Update a Document (2)

Generasi Gigih

As the result of the previous snippet, if we try to read the first document in movies collection, we'll get the following output:

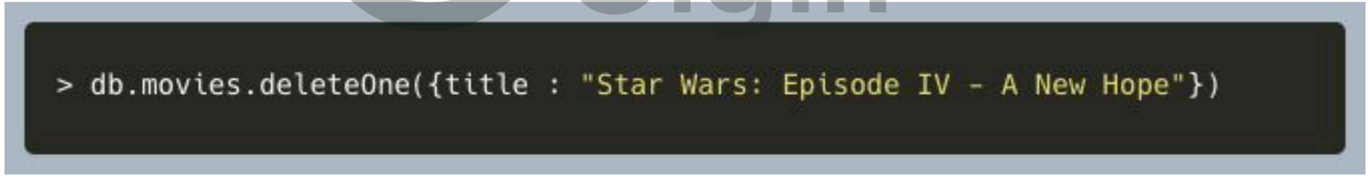


```
> db.movies.find().pretty()
{
    "_id" : ObjectId("5721794b349c32b32a012b11"),
    "title" : "Star Wars: Episode IV - A New Hope",
    "director" : "George Lucas",
    "year" : 1977,
    "reviews" : [ ]
}
```

Basic Operations - Delete a Document

To delete a document, we can use deleteOne function as follows:

Generasi Gigih



```
> db.movies.deleteOne({title : "Star Wars: Episode IV - A New Hope"})
```

Basic Data Types (1)

Documents in MongoDB can be thought of as “JSON-like” in that they are conceptually similar to objects in JavaScript. JSON is a simple representation of data, its specification can be described in about one paragraph. This is a good thing as it’s easy to understand, parse, and remember. On the other hand, JSON’s expressive capabilities are limited because its only types are:

- null
- boolean
- numeric
- string
- array
- object

Generasi Gigih

Basic Data Types (2)

Generasi Gigih

MongoDB adds support for a number of additional data types while keeping JSON's essential key/value–pair nature. Exactly how values of each type are represented varies by language, but this is a list of the commonly supported types and how they are represented as part of a document in the shell. The most common types are:

- null
- boolean
- number
- string
- date
- regular expressions
- array
- embedded document
- object ID
- binary data
- code

Questions?

Homework

Homework

1. Create a database that stores the following information:
    - Songs, containing the following data: the title of the song, the name of the artist(s), and the album
    - Artists, containing the following data: name, date of birth, genre(s)
    - Popular Songs, containing the following data: the title of the song, how many times it's played, period of time
2. Populate the database you've created above with at least 10 data for each collection