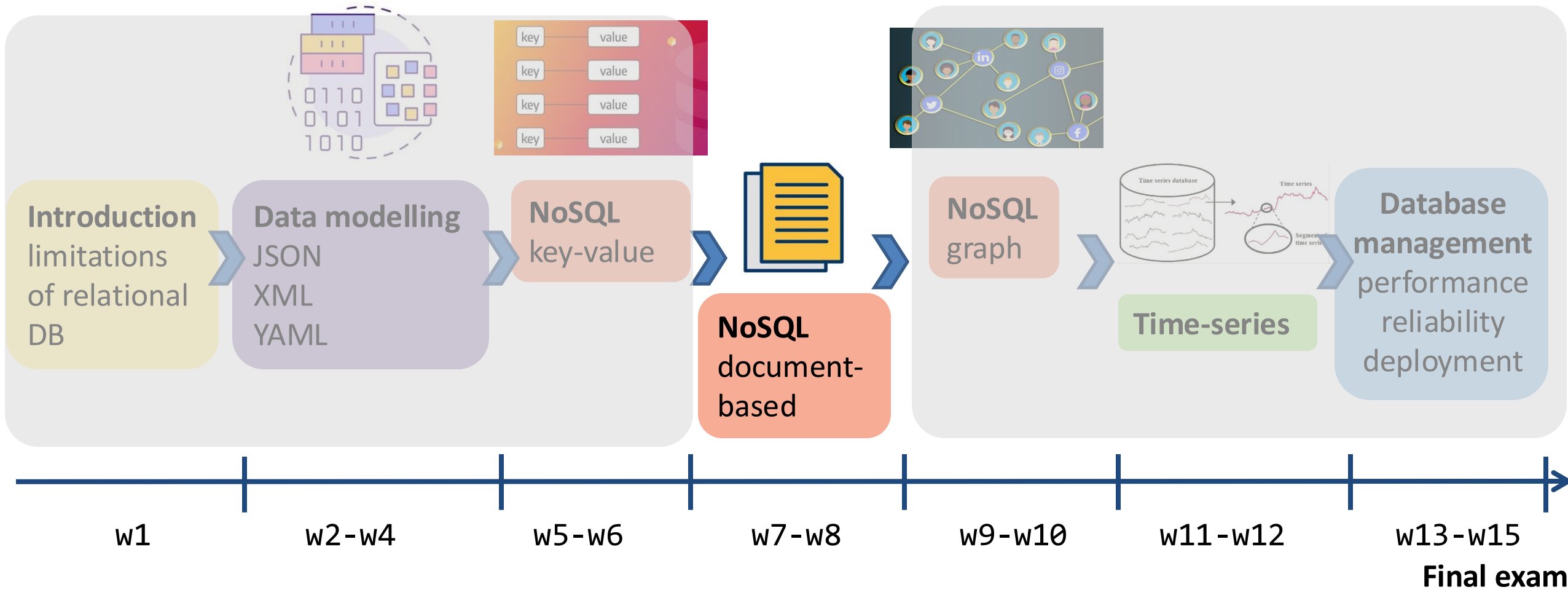




Document-based DBs

Beyond relational-DBs

Planning of the course



Memento: NoSQL families

- ▶ Key value stores
- ▶ Document databases
- ▶ Graph databases
- ▶ Wide column stores



Remember: Key-value database management store

- ▶ Various vendors provide key-value database management store



Voldemort



DynamoDB



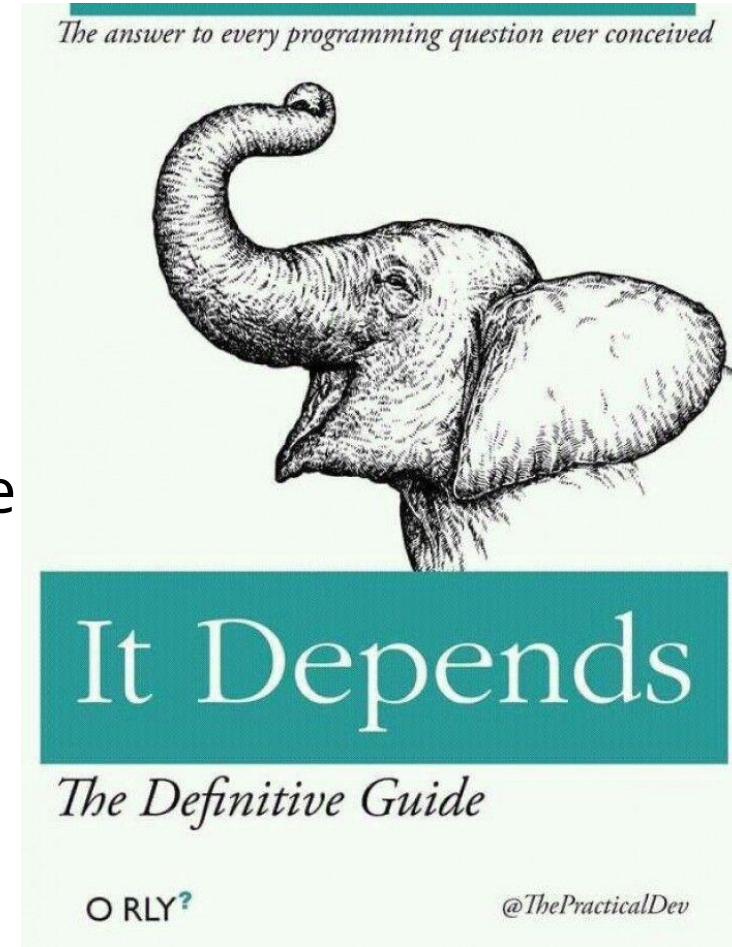
wait, wasn't this document-based?

Document stores

- ▶ **What is it**
 - ▶ Technically, a special form of key-value store

So what is special ?

- ▶ Normal key-value store **do not** assume any structure of the value
- ▶ Okay, is that a problem?
- ▶ Well, it depends...
- ▶ **Consequence:** No queries/ index to be meaningfully created based on the value



Document stores

- ▶ BUT is it true ?
 - ▶ Is there really no structure.
- ▶ Well, sometimes...

```
#Student record
---
firstname: John
lastname: Smith
age: 26
programming_languages:
  scala: Advanced
  python: Intermediate
  C: Beginner
hobbies:
  - gaming
  - gym
  - playing_guitar
favorite_food:
  vegetables:
  tomatoes
  fruits:
    citrus: lemon
    nuts: peanuts
```



```
{
  "firstname": "John",
  "lastname": "Smith",
  "age": 26,
  "programming_languages": {
    "scala": "Advanced",
    "python":
      "Intermediate",
    "C": "Beginner"
  },
  "hobbies": [
    "gaming",
    "gym",
    "playing_guitar"
  ],
  "favorite_food": {
    "vegetables":
      "tomatoes",
    "fruits": {
      "citrus": "lemon",
      "nuts": "peanuts"
    }
  }
}
```

Document storage

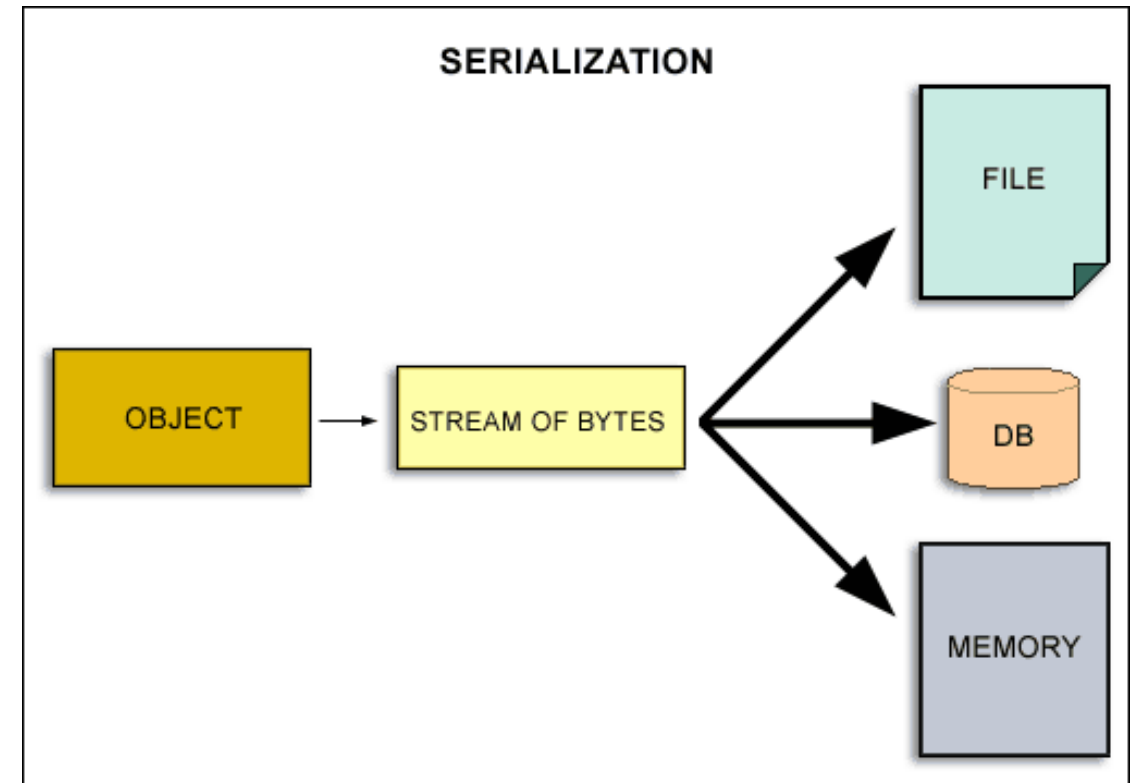
- ▶ Well, if there is structure... what is it...
- ▶ That is what we call here “document”
- ▶ We need to encode our documents...
 - Typically, this is done in
 - XML
 - JSON
 - YAML

What is a document

- ▶ So we have
 - ▶ Key -> document
- ▶ Not all documents have to have the same structure (schema-free, NoSQL)
- ▶ But, documents do have a structure, which allows for supporting queries depending on the content of the document

When it can be an advantage










- ▶ When storing objects..
 - ▶ You do not need directly an OR-Mapper... but e.g. use your JSON mapper
- ▶ You get data as documents anyway (e.g. JSON via browsers or other libraries)



- ▶ Documents are organized in
 - ▶ Databases (as usual a DBMS can manage several DBs)
 - ▶ In Insight DBs we have **collections**
 - ▶ A collection than can contain documents
 - Note they do not have to have the same structure!

We are looking at MongoDB

Probably the most-popular document-store

Rank			DBMS	Database Model	Score		
Mar 2025	Feb 2025	Mar 2024			Mar 2025	Feb 2025	Mar 2024
1.	1.	1.	MongoDB 	Document, Multi-model 	396.42	-0.21	-28.11
2.	2.	 3.	Databricks	Multi-model 	96.01	+5.97	+21.67
3.	3.	 2.	Amazon DynamoDB	Multi-model 	76.57	+0.99	-1.14
4.	4.	4.	Microsoft Azure Cosmos DB	Multi-model 	22.27	+0.14	-8.12
5.	5.	5.	Couchbase	Multi-model 	15.05	-0.58	-4.11
6.	6.	6.	Firebase Realtime Database	Document	13.49	+0.17	-1.58
7.	7.	7.	CouchDB	Document, Multi-model 	7.34	-0.47	-4.39
8.	8.	8.	Google Cloud Firestore	Document	7.11	+0.20	-2.86

Mongo DB

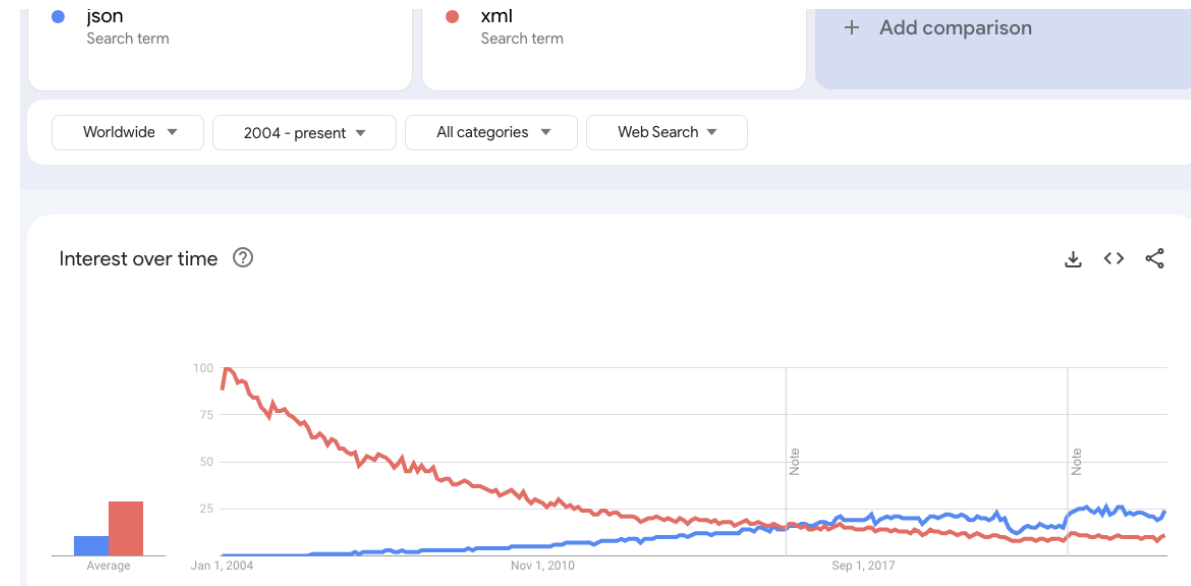
- And also among “normal” DBs one of the most popular these days

Rank			DBMS	Database Model	Score		
Mar 2025	Feb 2025	Mar 2024			Mar 2025	Feb 2025	Mar 2024
1.	1.	1.	Oracle	Relational, Multi-model ⓘ	1253.08	-1.74	+32.02
2.	2.	2.	MySQL	Relational, Multi-model ⓘ	988.13	-11.86	-113.37
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model ⓘ	788.14	+1.27	-57.67
4.	4.	4.	PostgreSQL +	Relational, Multi-model ⓘ	663.42	+3.81	+28.52
5.	5.	5.	MongoDB +	Document, Multi-model ⓘ	396.42	-0.21	-28.11
6.	↑ 7.	↑ 9.	Snowflake	Relational	161.78	+6.20	+36.40
7.	↓ 6.	↓ 6.	Redis	Key-value, Multi-model ⓘ	155.36	-2.55	-1.64
8.	8.	↓ 7.	Elasticsearch	Multi-model ⓘ	131.38	-3.25	-3.41
9.	9.	↓ 8.	IBM Db2	Relational, Multi-model ⓘ	126.57	+1.14	-1.18
10.	10.	10.	SQLite	Relational	113.08	-0.74	-5.08

- ▶ We often have JSON files, we want to store...
 - ▶ ... whatever (in the following what we did in the past)
 - ▶ Traffic data of the Cantons
 - ▶ AirBnB data
 - ▶ Or for example mouse-tracking

Mongo-DB

- ▶ From a previous lecture: Json became the most used data exchange format for data transfer on the web
- ▶ MongoDB is a document-store for JSON
- ▶ Note there are other systems for other document types, in particular XML



Some MongoDB Hands on

- ▶ We assume you know Postgres & Redis by now....
- ▶ Some examples now for Mongo
- ▶ Attention: I used an Ubuntu VM

Start your engines

- ▶ We need to install MongoDB, e.g.

```
~$ sudo apt-get install -y mongodb-org
```

- ▶ Start-up

```
$ sudo systemctl start mongod
```

- ▶ Check it

```
$ sudo systemctl status mongod
```


A small JSON Example: Mouse-Tracking

- ▶ We have
 - ▶ 1 server
 - ▶ N clients (using our website)
 - ▶ We want to know where who is clicking, (and when...)
- ▶ For what this can be good
 - ▶ Marketing
 - ▶ Psychology research
 - ▶ Security

Empty DBs are boring

```
1 import random
2 import pymongo
3
4 # Connect to the local MongoDB (assuming it's running on the default port 27017)
5 client = pymongo.MongoClient("mongodb://localhost:27017/")
6
7 # Create or select a database (e.g., "mydb")
8 db = client["mydb"]
9
10 # Create or select a collection (e.g., "mycollection")
11 collection = db["mycollection"]
12
13
14
15 # Generate a random number between 1 and 100 (inclusive)
16 for i in range(10):
17     random_x = random.randint(1, 100)
18     random_y = random.randint(1, 100)
19     random_user = random.randint(1,10)
20
21
22     json_document = {
23         "x-pos" : random_x,
24         "y-pos" : random_y,
25         "user" : random_user
26     }
27     collection.insert_one(json_document)
28     print(json_document)
29
```

How to get something out of the DB

```
e$ mongosh
```

```
> db.mycollection.find({"user":10})
```

Reading by program










```
1 import random
2 import pymongo
3
4 # Connect to the local MongoDB (assuming it's running on the default port 27017)
5 client = pymongo.MongoClient("mongodb://localhost:27017/")
6
7 # Create or select a database (e.g., "mydb")
8 db = client["mydb"]
9
10 # Create or select a collection (e.g., "mycollection")
11 collection = db["mycollection"]
12
13 retval = collection.find({"user":10})
14
15 for data in retval:
16     print(data)
17
```

TASK










- ▶ Make the Mongo DB work

Everyone wants to be a jack of all trades..

- ▶ Redis has many modules that support different things (e.g. JSON,)
- ▶ PostgreSQL has support for documents
- ▶ MongoDB becomes more like a traditional RDBMS, adding multi-document ACID transaction, secondary-index, advanced query capabilities.
- ▶ Postgres keeps improving its JSON capabilities such as indexing, query optimization, and more operators, which makes people wonder whether MongoDB becomes obsolete.

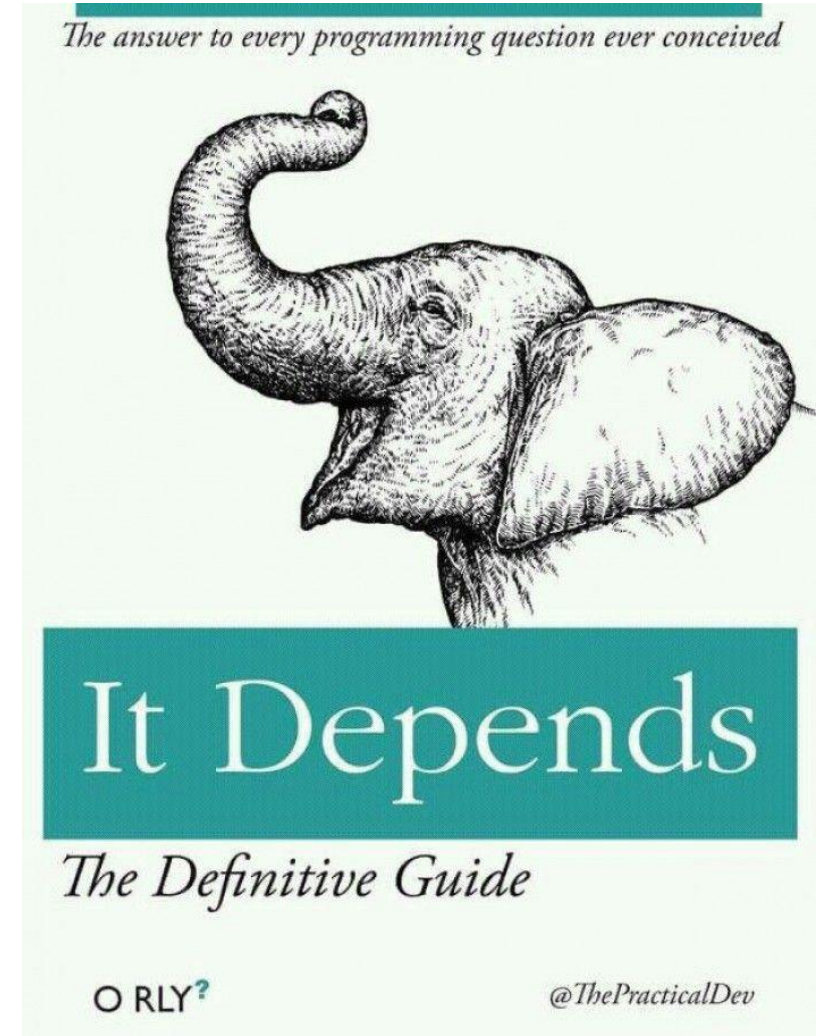
Rank			DBMS	Database Model	Score		
Mar 2025	Feb 2025	Mar 2024			Mar 2025	Feb 2025	Mar 2024
1.	1.	1.	MongoDB 	Document, Multi-model 	396.42	-0.21	-28.1
2.	2.	 3.	Databricks	Multi-model 	96.01	+5.97	+21.6
3.	3.	 2.	Amazon DynamoDB	Multi-model 	76.57	+0.99	-1.1
4.	4.	4.	Microsoft Azure Cosmos DB	Multi-model 	22.27	+0.14	-8.1
5.	5.	5.	Couchbase	Multi-model 	15.05	-0.58	-4.1
6.	6.	6.	Firebase Realtime Database	Document	13.49	+0.17	-1.5
7.	7.	7.	CouchDB	Document, Multi-model 	7.34	-0.47	-4.3
8.	8.	8.	Google Cloud Firestore	Document	7.11	+0.20	-2.8

But if everyone can do anything... what should I use.

Rank			DBMS	Database Model	Score		
Mar 2025	Feb 2025	Mar 2024			Mar 2025	Feb 2025	Mar 2024
1.	1.	1.	MongoDB 	Document, Multi-model 	396.42	-0.21	-28.11
2.	2.	 3.	Databricks	Multi-model 	96.01	+5.97	+21.67
3.	3.	 2.	Amazon DynamoDB	Multi-model 	76.57	+0.99	-1.14
4.	4.	4.	Microsoft Azure Cosmos DB	Multi-model 	22.27	+0.14	-8.12
5.	5.	5.	Couchbase	Multi-model 	15.05	-0.58	-4.11
6.	6.	6.	Firebase Realtime Database	Document	13.49	+0.17	-1.58
7.	7.	7.	CouchDB	Document, Multi-model 	7.34	-0.47	-4.39
8.	8.	8.	Google Cloud Firestore	Document	7.11	+0.20	-2.86

db-engines.com/en/ranking

- So, who is really the best....
 - Just what everyone else is doing?
 - What is best for me?



Okay... but we need to select one

- ▶ How to choose?
 - ▶ What do we need to compare?
 - Execution speed of CRUD operators (and to know which one are the relevant in your case)
 - Create
 - Read
 - Update
 - Delete
 - Development speed
 - How long to learn the technology
 - How long to develop something (also depends on how often to change)

When you need to compare?

- ▶ Which one is faster



- ▶ If your track looks like this:




Visibility of the context?

- ▶ Will I get wet when it rains?



- ▶ Your context is often not completely visible... so make sure you have all information ..
 - ▶ Ask questions?
 - ▶ Be aware that information might could be missing

Define KPIs



**"If you can't measure
it, you can't improve
it."**

Peter Drucker

- ▶ Know something is relevant is one thing... measure it another...
 1. Define KPIs relevant for our scenario
 2. Define measures for that

Measuring things

- ▶ Execution speed
 - ▶ Well, that is easy... (is it?)
 - ▶ ms to handle x query
 - ▶ BUT: what query
 - CRUD
 - Complexity of queries
 - Mix of types of queries
- ▶ Development effort
 - ▶ Well, that is easy (is it?)
 - ▶ Hours my developer spend on it
 - ▶ BUT: how is the time spend
 - Learning is an investment
 - Initial development vs. maintenance
 - Pre-existing knowledge



And what is the tradeoff??

Task:

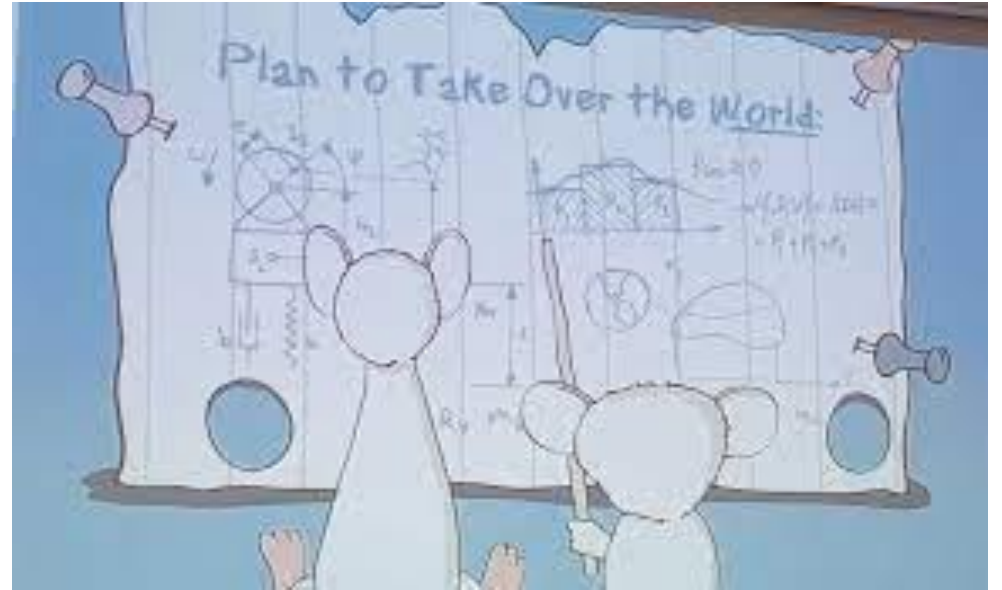
- ▶ We need to store the data for later analysis.
- ▶ We want to use a DB as (as large amount of files become cumbersome)
- ▶ We have three alternatives (those you already know)
 - ▶ Postgres
 - ▶ Redis
 - ▶ Mongo

You need to compare the different options

- ▶ It will be on you to give a recommendation, based on your comparison
 - ▶ Technology choices are key! (and often part of your future job)
 - ▶ You need to show it for the specific case, not based on what “others” say, only...

WHAT DO WE NEED TO DO???

- ▶ Your task of the day!
- ▶ Make a plan
 - ▶ Which aspects are relevant (and why)?
 - ▶ How can the relevant aspects be quantified?
 - ▶ How do you can measure them?
 - ▶ How can you make a **“fair” comparison**?
- ▶ Note: This is the first part of the Lab (graded part for B-DB CC)



Make the comparison

- ▶ 1) Make a plan on how to compare faire (e.g. same functionality, same environment....)
- ▶ 2) Make a plan to perform the same task in the other DBsx

Make the comparison

- ▶ 3) Compare the approaches, based on your comparison plan

Make the comparison

- ▶ 4 Make a report about the results

What shall be in the report

- ▶ 1 How do your solutions look like (for the different approaches)
- ▶ 2 What are your measures and how do you measure
- ▶ 3 What did you do to ensure fairness
- ▶ 4 What would be your recommendation
- ▶ 5 What did not work as you would have thought, and maybe why

ANY
QUESTIONS
?