Introduction to Databases
Part 2: MongoDB

Ashok Dudhat
Germany Startup Jobs
www.germanystartupjobs.com

What is NOSQL?

A variety of technologies that are alternatives to **Tables** and Structured Query Language (**SQL**). Stands for Not Only SQL.

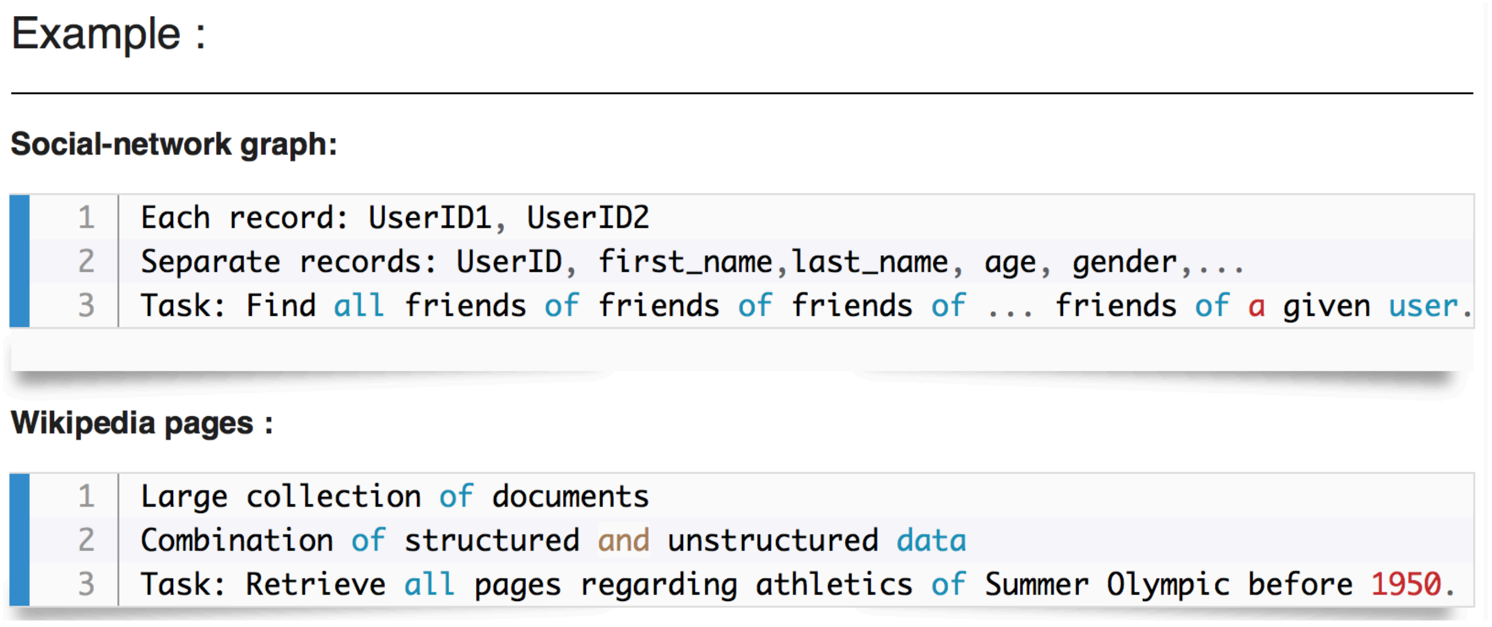
NoSQL is a **non-relational database management systems**, different from traditional relational database management systems in some significant ways. It is designed for distributed data stores where very large scale of data storing needs (for example Google or Facebook which collects terabits of data every day for their users). These type of data storing may **not require fixed schema**, **avoid join operations** and typically **scale horizontally**.

Why NoSQL?

In today's time data is becoming easier to access and capture through third parties such as Facebook, Google+ and others.

Personal user information, social graphs, geo location data, user-generated content and machine logging data are just a few examples where the data has been increasing exponentially.

To avail the above service properly, it is required to process huge amount of data. Which SQL databases were never designed. The evolution of NoSql databases is to handle these huge data properly.



Example :

Social-network graph:

```
Each record: UserID1, UserID2
Separate records: UserID, first_name,last_name, age, gender,...
Task: Find all friends of friends of friends of ... friends of a given user.
```

Wikipedia pages :

```
Large collection of documents
Combination of structured and unstructured data
Task: Retrieve all pages regarding athletics of Summer Olympic before 1950.
```

The Big Picture Differences

The Language

Think of a town - we'll call it **Town A** - where everyone speaks the **same language**. All of the businesses are built around it, every form of communication uses it - in short, it's the only way that the residents understand and interact with the world around them. **Changing that language** in one place would be c**onfusing and disruptive for everyone**.

Now, think of another town, **Town B**, where every home can speak a **different language**. Everyone interacts with the world differently, and there's no "universal" understanding or set organization. I**f one home is different, it doesn't affect anyone else at all**.

SQL databases

SQL databases use structured query language (SQL) for defining and manipulating data. On one hand, this is extremely powerful: SQL is one of the most versatile and widely-used options available, making it a safe choice and especially great for complex queries. On the other hand, it can be restrictive. SQL requires that you use **predefined schemas** to determine the structure of your data before you work with it. In addition, **all of your data must follow the same structure**.

This can require significant up-front preparation, and, as with **Town A**, it can mean that a **change in the structure** would be both **difficult and disruptive to your whole system**.

NoSQL database

A NoSQL database, on the other hand, has **dynamic schema for unstructured data**, and data is stored in many ways: it can be column-oriented, document-oriented, graph-based or organized as a KeyValue store. This flexibility means that:

- You can create documents without having to first define their structure.
- Each document can have its own unique structure.
- The syntax can vary from database to database.
- You can add fields as you go.

The Scalability

In most situations, **SQL databases are vertically scalable**, which means that you can increase the load on a single server by increasing things like CPU, RAM or SSD.

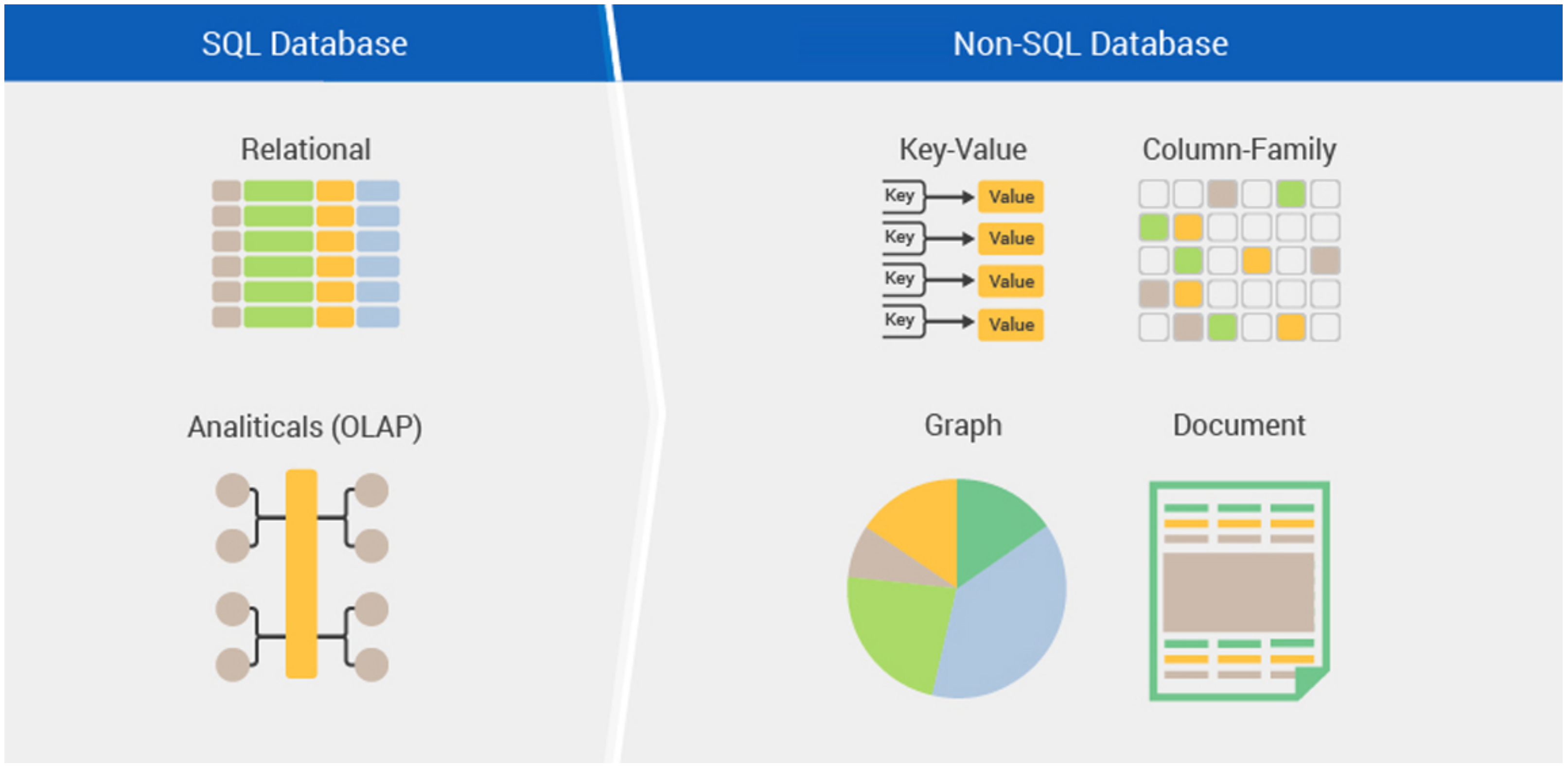
NoSQL databases, on the other hand, **are horizontally scalable**. This means that you handle more traffic by sharding, or adding more servers in your NoSQL database.

It's like **adding more floors to the same building** versus **adding more buildings to the neighborhood**.

The latter can ultimately become larger and more powerful, making NoSQL databases the preferred choice for large or ever-changing data sets.

The Structure

SQL databases are **table-based**, while **NoSQL databases** are **either document-based, key-value pairs, graph databases or wide-column stores**.



Examples of NoSQL

Type	Example	
Key-Value Store	redis	riak
Wide Column Store	H·BASE	cassandra
Document Store	mongoDB	CouchDB relax
Graph Store	Neo4j	InfiniteGraph The Distributed Graph Database

Core NOSQL Systems

Riak API: tons of languaes, JSON, Protocol: REST, Query Method: MapReduce term matching , Scaling: Multiple Masters; Written in: Erlang, Concurrency: eventually consistent (stronger then MVCC via Vector Clocks).

Redis API: Tons of languages, Written in: C, Concurrency: in memory and saves asynchronous disk after a defined time. Append only mode available. Different kinds of fsync policies. Replication: Master / Slave, Misc: also lists, sets, sorted sets, hashes, queues.

Hadoop / HBase API: Java / any writer, Protocol: any write call, Query Method: MapReduce Java / any exec, Replication: HDFS Replication, Written in: Java, Concurrency

Cassandra massively scalable, partitioned row store, masterless architecture, linear scale performance, no single points of failure, read/write support across multiple data centers & cloud availability zones. API / Query Method: CQL and Thrift, replication: peer-to-peer, written in: Java, Concurrency: tunable consistency, Misc: built-in data compression, MapReduce support, primary/secondary indexes, security features.

MongoDB **API: BSON**, Protocol: C, **Query Method: dynamic object-based language & MapReduce**, Replication: Master Slave & Auto-Sharding, **Written in: C++**,Concurrency: Update in Place. Misc: Indexing, GridFS, Freeware + Commercial License.

CouchDB API: JSON, Protocol: REST, Query Method: MapReduceR of JavaScript Funcs, Replication: Master Master, Written in: Erlang, Concurrency: MVCC.

Neo4J API: lots of langs, Protocol: Java embedded / REST, Query Method: SparQL, nativeJavaAPI, JRuby, Replication: typical MySQL style master/slave, Written in: Java, Concurrency: non-block reads, writes locks involved nodes/relationships until commit.

Infinite Graph (by Objectivity) API: Java, Protocol: Direct Language Binding, Query Method: Graph Navigation API, Predicate Language Qualification, Written in: Java (Core C++), Data Model: Labeled Directed Multi Graph, Concurrency: Update locking on subgraphs, concurrent non-blocking ingest, Misc: Free for Qualified Startups.

SQL vs NoSQL: MySQL vs MongoDB

MySQL: The SQL Relational Database

The following are some MySQL benefits and strengths:

- **Maturity**: MySQL is an extremely established database, meaning that there's a huge community, extensive testing and quite a bit of stability.
- **Compatibility:** MySQL is available for all major platforms, including Linux, Windows, Mac, BSD and Solaris. It also has connectors to languages like Node.js, Ruby, C#, C++, Java, Perl, Python and PHP, meaning that it's not limited to SQL query language.
- **Cost-effective:** The database is open source and free.
- **Replicable:** The MySQL database can be replicated across multiple nodes, meaning that the workload can be reduced and the scalability and availability of the application can be increased.
- **Sharding:** While sharding cannot be done on most SQL databases, it can be done on MySQL servers. This is both cost-effective and good for business.

SQL vs NoSQL: MySQL vs MongoDB

MongoDB: The NoSQL Non-Relational Database.

The following are some of MongoDB benefits and strengths:

- **Dynamic schema:** As mentioned, this gives you flexibility to change your data schema without modifying any of your existing data.
- **Scalability:** MongoDB is horizontally scalable, which helps reduce the workload and scale your business with ease.
- **Manageability:** The database doesn’t require a database administrator. Since it is fairly user-friendly in this way, it can be used by both developers and administrators.
- **Speed:** It’s high-performing for simple queries.
- **Flexibility:** You can add new columns or fields on MongoDB without affecting existing rows or application performance.

In Short : RDBMS vs NoSQL

RDBMS

- Structured and organized data.
- Structured query language (SQL).
- Data and its relationships are stored in separate tables.
- Data Manipulation Language, Data Definition Language.
- Tight Consistency.

NoSQL

- Stands for Not Only SQL.
- No declarative query language.
- No predefined schema.
- Key-Value pair storage, Column Store, Document Store, Graph databases.
- Eventual consistency rather ACID property (**A**tomic, **C**onsistency, **I**solation, **D**urable).
- Unstructured and unpredictable data.
- Prioritises high performance, high availability and scalability.
- BASE Transaction (**B**asically **A**vailable **S**oft state **E**ventual consistency).

Which Database Is Right For Your Business?

MySQL is a strong choice for any business that will benefit from its pre-defined structure and set schemas. For example, applications that require **multi-row transactions** - like **accounting systems** or **systems that monitor inventory** - or that run on legacy systems will thrive with the MySQL structure.

MongoDB, on the other hand, is a good choice for businesses that have rapid growth or databases with no clear schema definitions. More specifically, **if you cannot define a schema for your database**, if you find yourself denormalizing data schemas, or **if your schema continues to change** - as is often the case with **mobile apps**, **real-time analytics**, **content management systems**, etc.- MongoDB can be a strong choice for you.

Production Deployment Examples

There is a large number of companies using NoSQL.

- Google
- Facebook
- Mozilla
- Adobe
- Foursquare
- LinkedIn
- Digg
- McGraw-Hill Education
- Vermont Public Radio

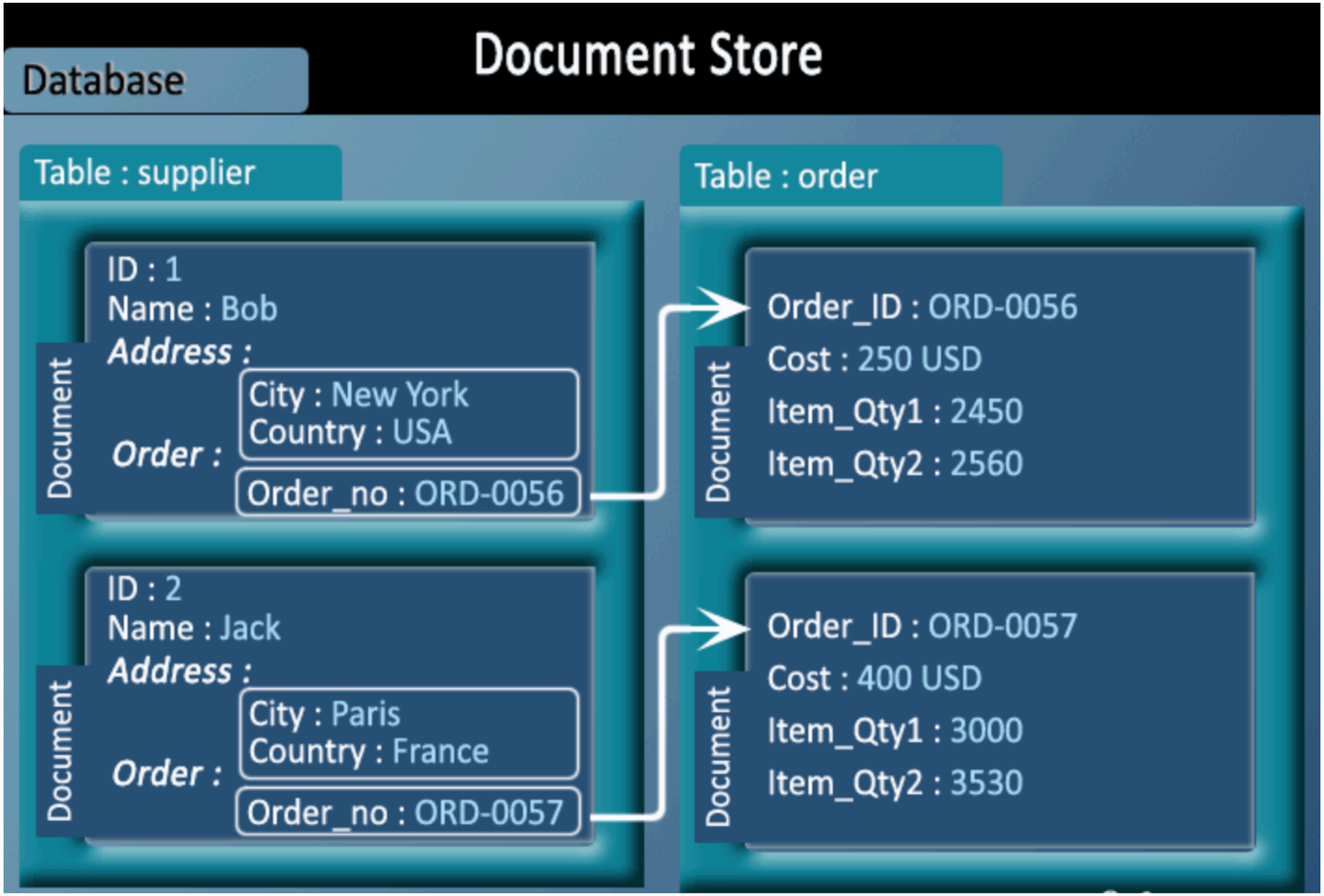
Document Oriented databases

- A **collection** of **documents**.
- **Data** in this model is stored inside **documents**.
- A document is a **key value collection** where the key allows access to its value.
- Documents are **not typically forced to have a schema** and therefore are flexible and easy to change.
- **Documents are stored into collections** in order to group different kinds of data.
- Documents can contain many different key-value pairs, or key-array pairs, or even nested documents.

Comparison between Classic relational model and Document model

MySQL (Relational Model)	MongoDB (Document Model)
Tables	Collections
Rows	Documents
Columns	Key/value pairs
Joins	Embedded documents, $lookup & $graphLookup
GROUP_BY	Aggregation Pipeline
ACID Transactions	ACID Transactions
Secondary Index	Secondary Index

Pictorial Presentation



Introduction : MangoDB

Document: „A record in a MongoDB collection and the basic unit of data in MongoDB. Documents look like JSON objects but exist as BSON“.

- BSON

```
{
   “title“: “Article two“,
   “category“: “Education“,
    “body“: “this is the body“
}
```

Documents

- BSON **is JSON saved binarily**

```
{
  “title“: “Article two“,
  “category“: “Education“,
   “body“: “this is the body“
}
```

- We do not have a JSON file, we have binary BSON file. Not human-readable.

Documents

- BSON is JSON saved binarily

{

“titl

“cat

“bo

}

Each document is JSON which is saved as a BSON file.

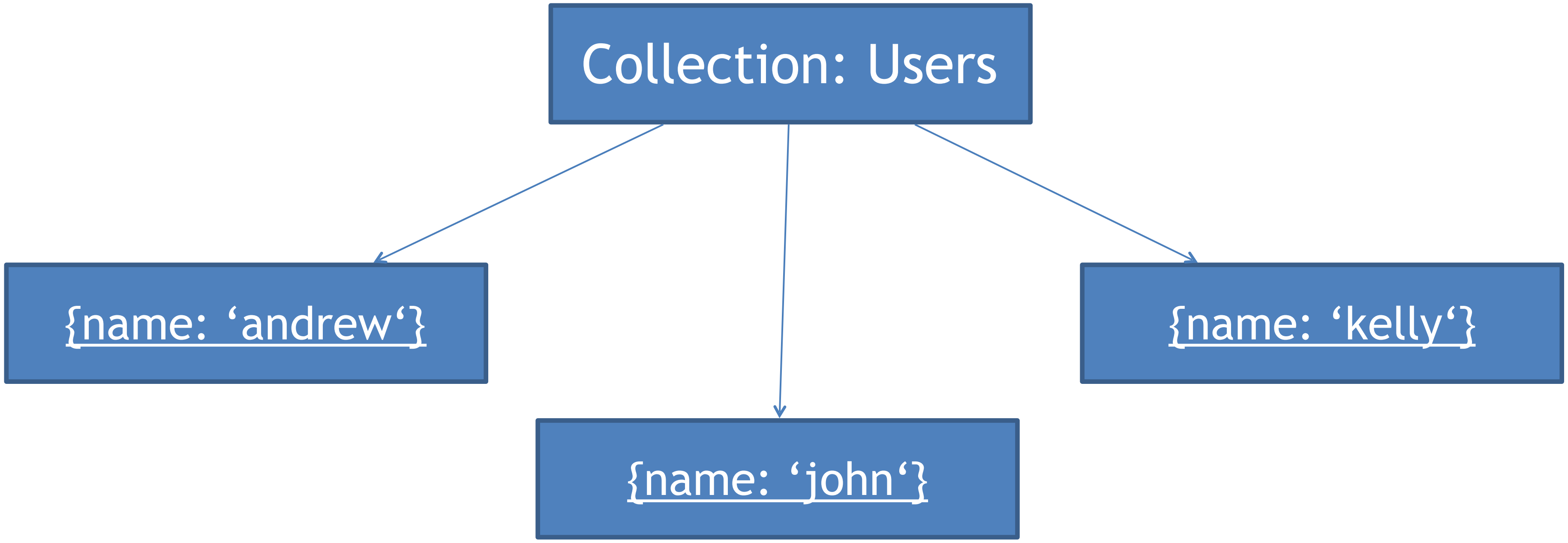
- We do not have a JSON file, we have binary BSON file. Not human-readable.

Collections

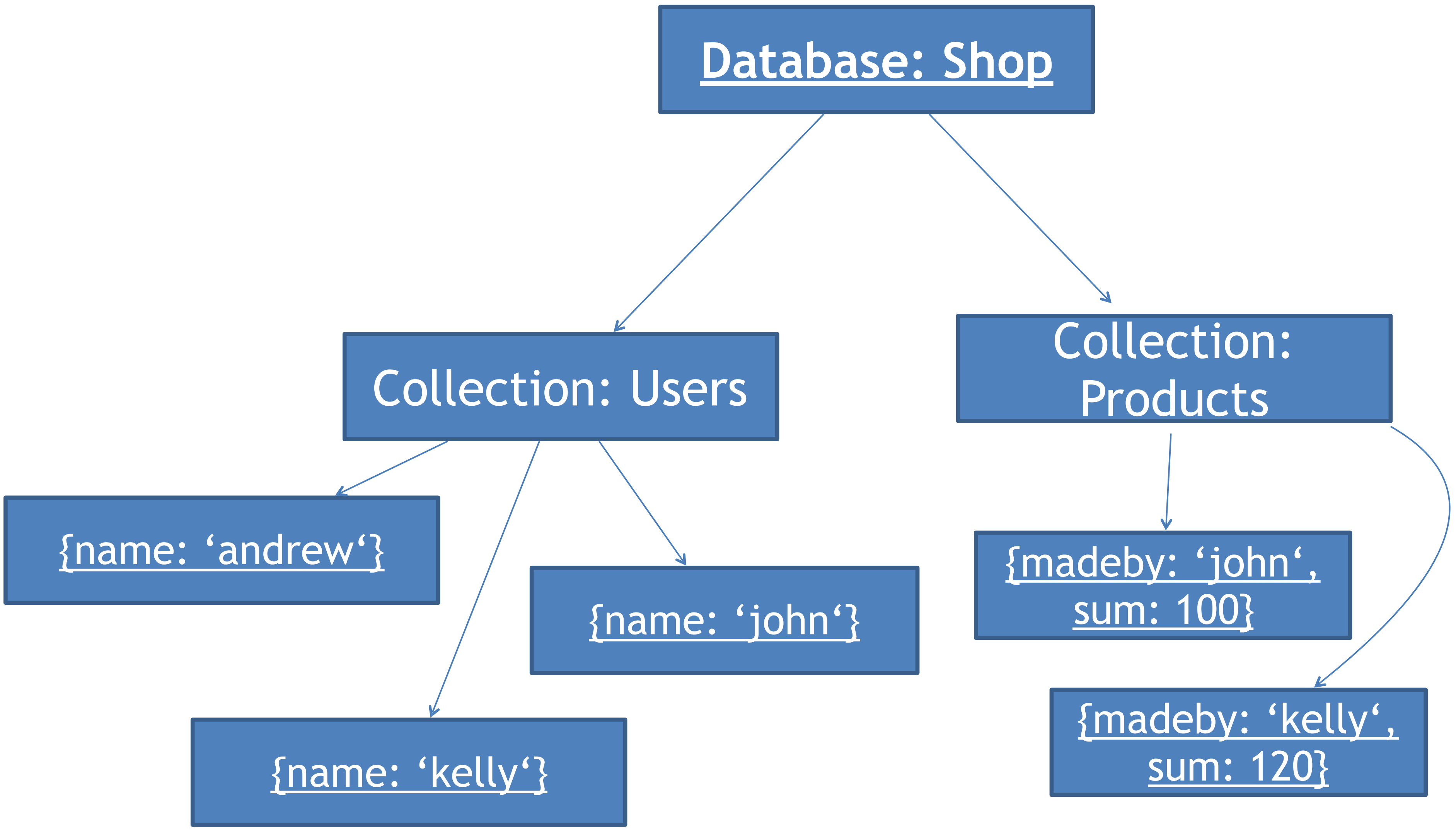
- Collection: A group of MongoDB documents. Typically, all documents in a collection have a similar related purpose.

Collections

- Collection: A group of MongoDB documents. Typically, all documents in a collection have a similar related purpose.



Collections



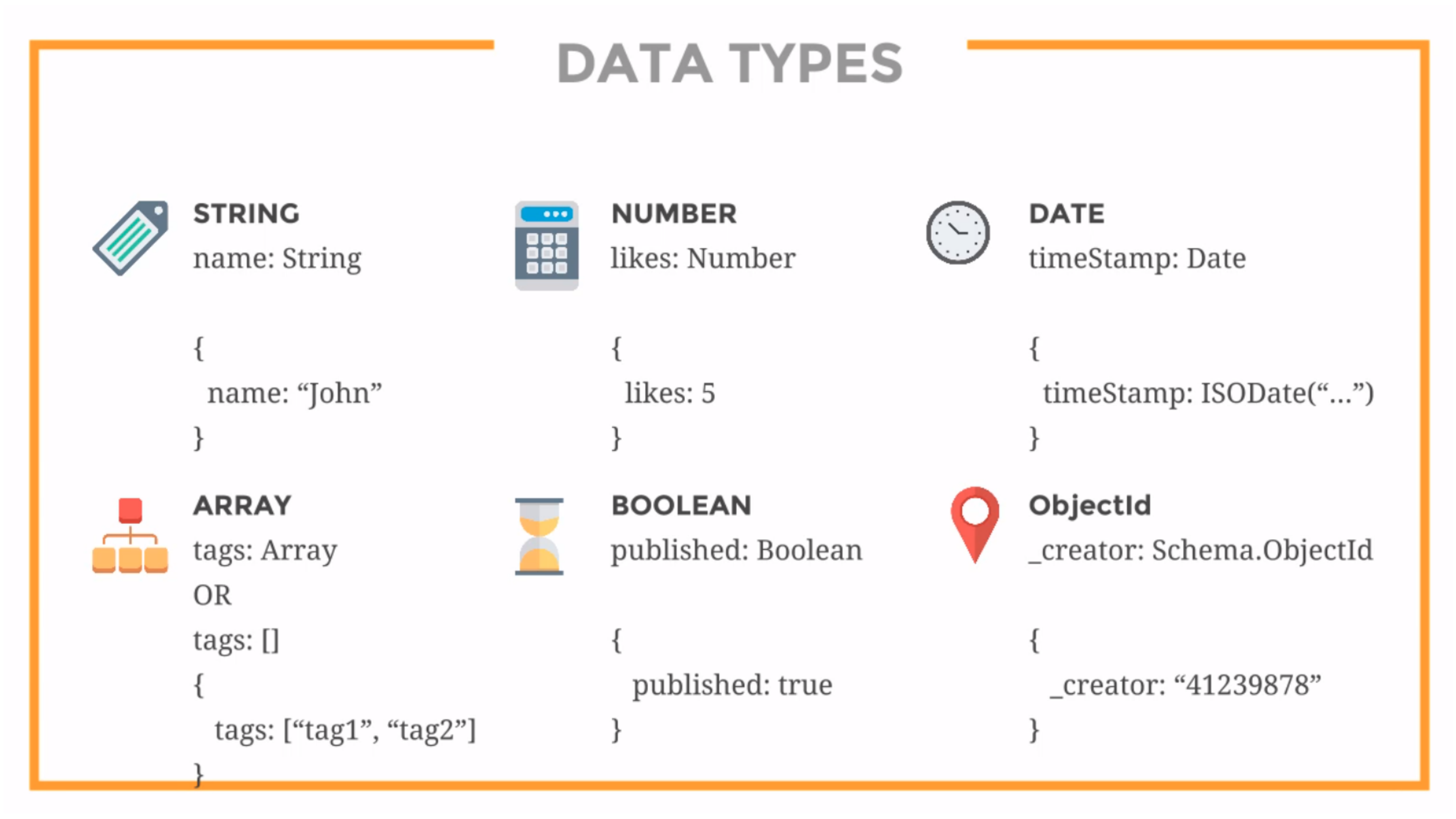
Biggest difference between SQL and MongoDB

SQL	MongoDB
JOINS	REFERENCES

Installation : MongoDB

- sudo apt-get update
- sudo apt-get install –y mongodb-org
- sudo systemctl start mongod
- sudo systemctl status mongod

Data-Types



DATA TYPES

STRING

name: String

```
{
  name: "John"
}
```

NUMBER

likes: Number

```
{
  likes: 5
}
```

DATE

timeStamp: Date

```
{
  timeStamp: ISODate("…")
}
```

ARRAY

tags: Array

OR

tags: []

```
{
  tags: ["tag1", "tag2"]
}
```

BOOLEAN

published: Boolean

```
{
  published: true
}
```

ObjectId

_creator: Schema.ObjectId

```
{
  _creator: "41239878"
}
```

Mongo Shell

Let Create Database, Collection, Document and play with some function/queries…

1. Create text file mongo_shell_1.txt
2. Create text file mongo_shell_2.txt

First We can start with mongo_shell_1.txt

Quiz

1. How do we access the shell?

A: by typing ‘mongodb‘
B: by typing ‘mongo‘
C: by loading the browser
D: by typing ‘mongo start‘

Quiz

1. How do we access the shell?

A: by typing ‘mongodb‘
B: by typing ‘mongo‘
C: by loading the browser
D: by typing ‘mongo start‘

Quiz

2. How do we add a new document if the to-be-updated document is not found?

A: $set

B: $in

C: upsert=true

D: $upsert=true

Quiz

2. How do we add a new document if the to-be-updated document is not found?

A: $set

B: $in

C: upsert=true

D: $upsert=true

Quiz

3. What command do we use to indicate which database we want to access?

A: use

B: show

C: find

D: list

Quiz

3. What command do we use to indicate which database we want to access?

A: use

B: show

C: find

D: list

Quiz

4. What method is used to display our documents in a clean and organized way?

A: insert
B: find
C: pretty
D: clean
E: style

Quiz

4. What method is used to display our documents in a clean and organized way?

A: insert

B: find

C: pretty

D: clean

E: style

Quiz

5. Which one of these is not one of the 6 main data types commonly used within the model of our collection?

A: String
B: Boolean
C: Number
D: Date
E: Buffer
F: Array

Quiz

5. Which one of these is not one of the 6 main data types commonly used within the model of our collection?

A: String
B: Boolean
C: Number
D: Date
E: Buffer
F: Array