



UBKUBK • 15h ago

Some might be taking your last sentence as a positive. I think it is a negative and for this discussion better phrased as "For all nba teams all he had a was a single third team..."



Vote



Reply



Share



YoungClint_TrapLord • 17h ago

His resume does seem extremely weak. He also doesn't have any college or international accomplishments to supplement it either



Vote



Reply



Share



HoopsAndBooks • 17h ago

Extremely weak? Lmfao top 15 all time in assists. Top tier defender.

Would've been finals MVP in 2010 if Perk didn't tear his ACL



Vote



Reply



Share



YoungClint_TrapLord • 16h ago

4x all star 1x all nba 3rd team averaging less than 10 points a game for his career is pretty weak.

Billups has a FMVP 3x all NBA 5x all star and 2 gold medals and he isn't in yet.



Vote



Reply



Share



Drummallumin • 15h ago

People really overestimate how much international matters, especially if it's for Team USA.

Chauncey is also normally near the top of any HOF snub list.



Vote



Reply



Share



```
CREATE TABLE comments(
  id INT PRIMARY KEY,
  id_of_replied INT PRIMARY KEY REFERENCES
comments.id -- for root comments it will be equal to id
)
```

```
CREATE OR REPLACE FUNCTION
retrieve_root_comment (id INTEGER)
  RETURNS parent_id INT
AS $$
DECLARE
  currentComment comments
BEGIN
  LOOP
    IF currentComment.id_of_replied ==
currentComment.id THEN
      EXIT;
    END IF;
    -- set currentComment to comment with
id_of_replied
  END LOOP;
END; $$
LANGUAGE 'plpgsql';
```

```
CREATE TABLE public.messages (
  id integer not null primary key unique generated
always as identity,
  message varchar(40)
);
CREATE TABLE public.reply (
  parent_id integer not null references messages
(id),
  child_id integer not null references messages (id)
);
insert into messages (message) values ('Message1');
insert into messages (message) values ('Message1');
insert into messages (message) values ('Message1');
insert into messages (message) values ('Message1');
insert into messages (message) values ('Message1');
insert into messages (message) values ('Message1');

insert into reply (parent_id, child_id) values (1,2);
insert into reply (parent_id, child_id) values (2,3);
insert into reply (parent_id, child_id) values (3,4);
insert into reply (parent_id, child_id) values (5,6);
WITH RECURSIVE r AS (
  SELECT child_id, parent_id, 1 AS level
  FROM reply r1
  WHERE parent_id = 1
  UNION ALL
  SELECT r1.child_id, r1.parent_id, r.level + 1 AS level
  FROM reply r1
  JOIN r
    ON r1.parent_id = r.child_id
)
SELECT * FROM r;
```

```
CREATE TABLE IF NOT EXISTS public.topic
(
  topic_id int NOT NULL,
  PRIMARY KEY (topic_id)
);
ALTER TABLE public.topic
  OWNER to postgres;
CREATE TABLE IF NOT EXISTS public.comment
(
  reply_id int NOT NULL,
  message text,
  topic_id int,
  reply_to int,
  PRIMARY KEY (reply_id),
  FOREIGN KEY (reply_to) REFERENCES
public.comment(reply_id),
  FOREIGN KEY (topic_id) REFERENCES
public.topic(topic_id)
);
ALTER TABLE public.comment
  OWNER to postgres;
SELECT * FROM comment WHERE reply_to =
{reply_id};
```

```
create table comments (  
    id int,  
    replied int,  
    comment text  
);  
insert into comments(id, replied, comment) VALUES (0,  
NULL, 'asdf');  
insert into comments(id, replied, comment) VALUES (1,  
0, 'asdf1');  
insert into comments(id, replied, comment) VALUES (2,  
1, 'asdf2');  
insert into comments(id, replied, comment) VALUES (3,  
NULL, 'asdf2');  
insert into comments(id, replied, comment) VALUES (4,  
3, 'asdf2');  
insert into comments(id, replied, comment) VALUES (5,  
4, 'asdf2');  
WITH RECURSIVE r AS (  
    SELECT id, replied, comment  
    FROM comments  
    WHERE id = 0  
    UNION  
    SELECT comments.id, comments.replied,  
comments.comment  
    FROM comments  
    JOIN r  
        ON comments.replied = r.id  
)  
SELECT * FROM r;
```

Databases - Tutorial 10

NOSQL - Mongoddb

Hamza Salem - Innopolis University

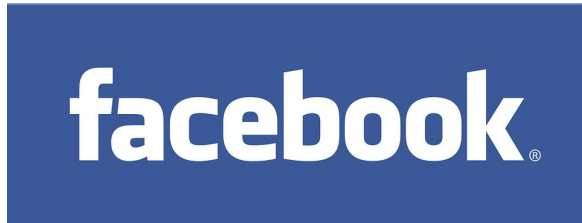


At least No SQL
injection in NoSQL

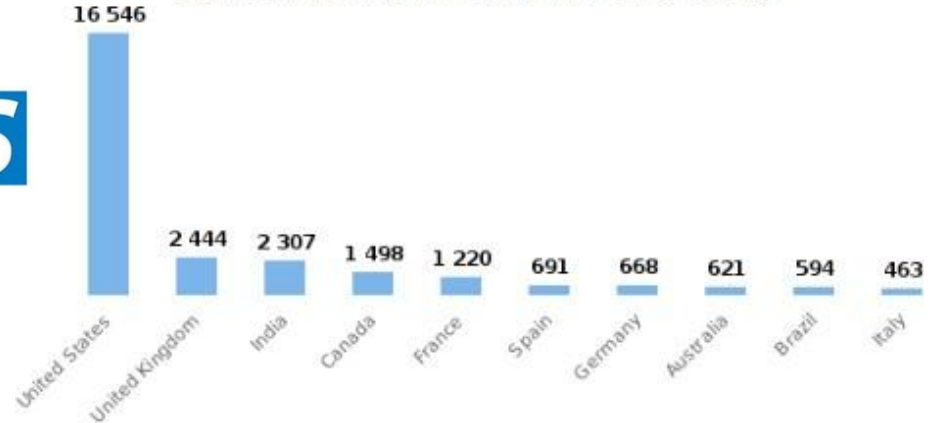
Introduction to NoSQL Systems

- Who is using NoSQL (Not Only SQL)?



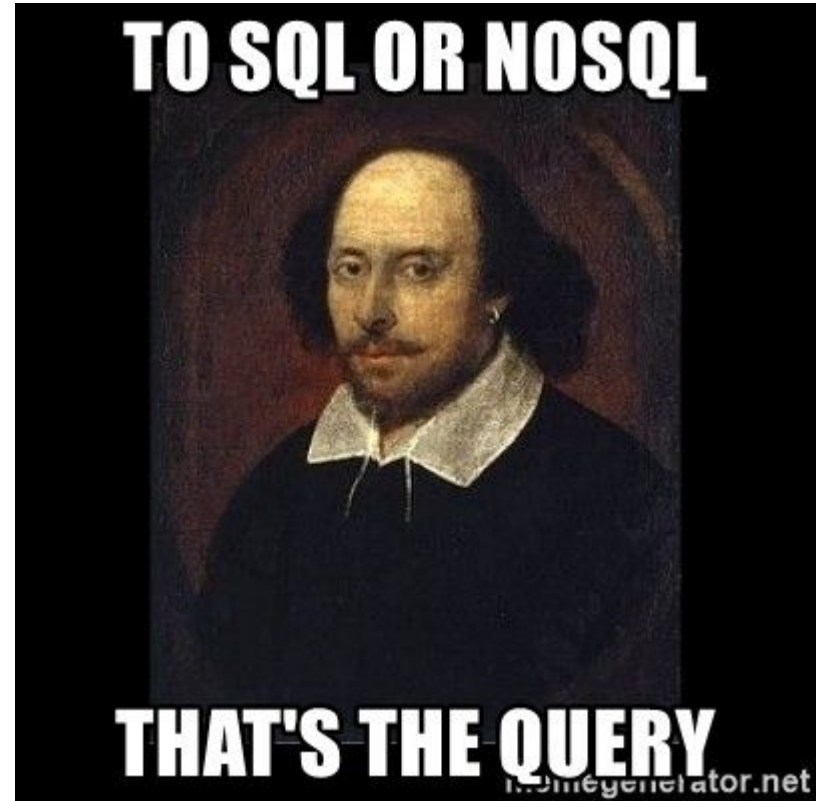


Distribution of companies using MongoDB by Country



Introduction to NoSQL Systems

- Motivation
 - Vast amounts of data and structured relational SQL system may not be appropriate
 - Restrictions regarding schemas
 - Use of data which is not suitable to traditional relational systems



Introduction to NoSQL Systems

```
personal.json x
1 {
2   "$schema": "personal-schema.json",
3   "personnel": {
4     "person": [
5       {
6
7         "id": "Big.Boss",
8         "name": {
9           "family": "Boss",
10          "given": "Big"
11        },
12        "email": "chief@oxygenxml.com",
13        "link": {
14          "subordinates": [
15            "one.worker",
16            "two.worker",
17            "three.worker",
18            "four.worker",
19            "five.worker"
20          ]
21        }
22      }
23    ]
24  }
25 }
```

```
<?xml version="1.0" encoding="iso-8859-8" standalone="yes" ?>
<CURRENCIES>
  <LAST_UPDATE>2004-07-29</LAST_UPDATE>
  <CURRENCY>
    <NAME>dollar</NAME>
    <UNIT>1</UNIT>
    <CURRENCYCODE>USD</CURRENCYCODE>
    <COUNTRY>USA</COUNTRY>
    <RATE>4.527</RATE>
    <CHANGE>0.044</CHANGE>
  </CURRENCY>
  <CURRENCY>
    <NAME>euro</NAME>
    <UNIT>1</UNIT>
    <CURRENCYCODE>EUR</CURRENCYCODE>
    <COUNTRY>European Monetary Union</COUNTRY>
    <RATE>5.4417</RATE>
    <CHANGE>-0.013</CHANGE>
  </CURRENCY>
</CURRENCIES>
```


Introduction to NoSQL Systems

Characteristics related to data models and query languages:

- Less Powerful Query Languages:
 - Only a subset of SQL querying capabilities would be provided
 - In particular, **many NOSQL systems do not provide join operations** as part of the query language itself; the joins need to be implemented in the application programs



NoSQL,
RDMS, etc



JSON file
database

Introduction to NoSQL Systems

Characteristics related to data models and query languages:

- Versioning:
 - Some NoSQL systems provide storage of multiple versions of the data items, with the timestamps of when the data version was created.

Time Series

```
{
  _id: "20130310/resource/home..htm",
  metadata: {
    date: ISODate("2013-03-10T00:00:00Z"),
    site: "main-site",
    page: "home..htm",
    ...
  },
  month : 3,
  total : 9120637,
  hourly: {
    0 : 361012,
    1 : 399034,
    ...,
    23 : 387010 },
  hour-minute: {
    0 : { 0 : 5678,
          1 : 6745,
          2 : 9212,
          ...,
          59 : 6823 },
    1 : { 0 : 8765,
          1 : 8976,
          2 : 8345,
          ...,
          59 : 9812 },
    ...,
    23 : { 0 : 7453,
           1 : 7432,
           2 : 7901,
```

Document-Based NoSQL Systems

- These systems store data in the form of documents using well-known formats, such as JSON (JavaScript Object Notation) or XML documents
- Documents are accessible via their document id, but can also be accessed rapidly using other indexes
- The documents are specified as self-describing data
- New documents can have new data elements that do not exist in any of the current documents in the collection
- Some of the top NoSQL Document Databases: MongoDB, CouchDB, MarkLogic, OrientDB, IBM Coudant, Azure Cosmos DB, IBM Informix

Document-Based NoSQL Systems

- Example of code on MongoDB:

<https://mkyong.com/mongodb/mongodb-hello-world-example/>

<https://metacpan.org/pod/MongoDB::Examples>

https://www.w3schools.com/python/python_mongodb_insert.asp

- Example of code on CouchDB:

<https://www.ibm.com/developerworks/opensource/library/os-couchdb/os-couchdb-pdf.pdf>

- <https://www.javatpoint.com/python-couchdb>

NoSQL Key-Value Stores

- These systems have a simple data model based on **fast access by the key to the value associated with a unique key**
- The value can be a record or an object or a document or even have a more complex data structure
- In many of these systems, there is no query language but rather a set of operations that can be used by the application programmers
- Top NoSQL Key-Value Databases: Amazon DynamoDB, Oracle NoSQL DB, Redis, Aerospike, Oracle Berkeley DB, Voldemort, Riak KV

NoSQL Key-Value Stores

- Example of code in Aerospike:

<https://www.aerospike.com/docs/guide/single.html>

- Example of code in Redis:

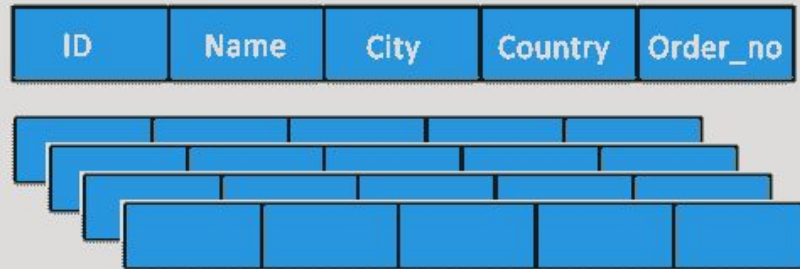
<https://redislabs.com/resources/how-to-redis-enterprise/>

- <https://redis.io/topics/data-types-intro>

Column-Based or Wide Column NoSQL Systems

- One of the main differences that distinguish column-based systems from key-value stores is the nature of the key
- In column-based systems, the **key is multidimensional** and so has several components: typically, a combination of table name, row key, column, and timestamp
- The column is typically composed of two components: column family and column qualifier.
- Each column family is stored in its own files
- Top Column-Based DB: MariaDB, ClickHouse, Apache Hbase, HyperTable, MonetDB

row-store



- + easy to add/modify a record
- might read in unnecessary data

column-store



- + only need to read in relevant data
- tuple writes require multiple accesses

=> suitable for read-mostly, read-intensive, large data repositories

Column-Based or Wide Column NOSQL Systems

- Examples of queries on MariaDB:
<https://mariadb.com/kb/en/useful-mariadb-queries/>
<https://mariadb.com/kb/en/basic-sql-statements/>
- Examples of queries on ClickHouse:
https://clickhouse.yandex/docs/en/query_language/select/

Row-oriented

ID	Name	Grade	GPA
001	John	Senior	4.00
002	Karen	Freshman	3.67
003	Bill	Junior	3.33

Column-oriented

Name	ID
John	001
Karen	002
Bill	003

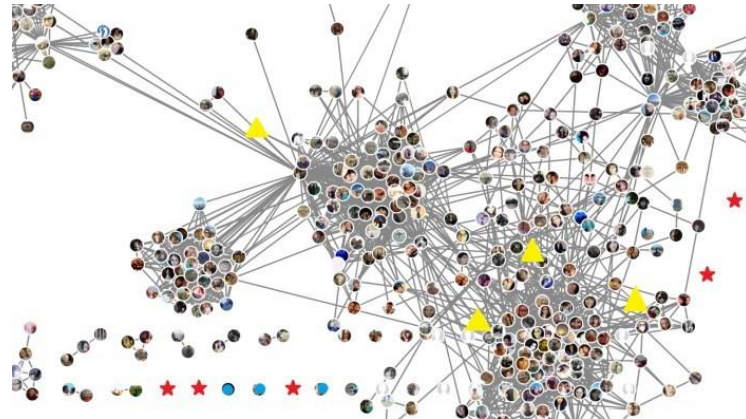
Grade	ID
Senior	001
Freshman	002
Junior	003

GPA	ID
4.00	001
3.67	002
3.33	003

NoSQL Graph Databases

- Data is represented as graphs, and related nodes can be found by traversing the edges using path expressions
- Both nodes and edges can be labelled to indicate the types of entities and relationships they represent, and it is generally possible to store data associated with both individual nodes and individual edges
- Top NoSQL Graph DB: Neo4j, ArangoDB, OrientDB, Amazon Neptune, FlockDB, DataStax, Cassandra, Cayley

Trivia:
<https://www.technologyreview.com/s/609091/first-evidence-that-online-dating-is-changing-the-nature-of-society/>



NoSQL Graph Databases

- Example of queries on Neo4J:

<https://neo4j.com/docs/cypher-manual/current/introduction/quering-updating-administering/>

<https://neo4j.com/developer/cypher-basics-i/>

- Example of queries in Cassandra:

<https://www.guru99.com/cassandra-query-language-cql-insert-update-delete-read-data.html>

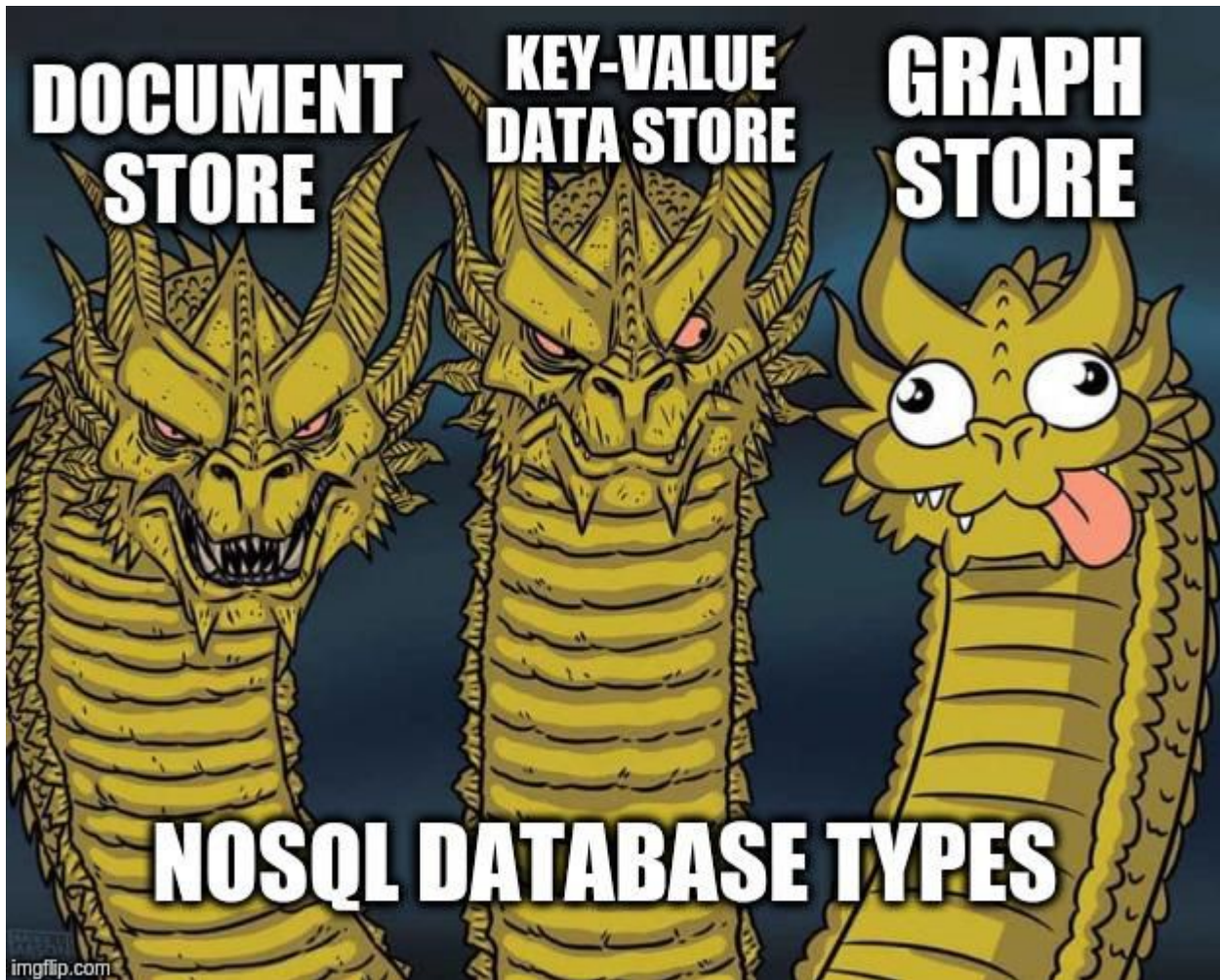
<http://cassandra.apache.org/doc/latest/cql/dml.html#select>

**DOCUMENT
STORE**

**KEY-VALUE
DATA STORE**

**GRAPH
STORE**

NOSQL DATABASE TYPES



HOW DO YOU SPELL YOUR NAME ?



MongoDB Inc. / Founders



Eliot Horowitz



Kevin P. Ryan



Dwight Merriman

Introduction

MongoDB is a document-oriented DB, which means:

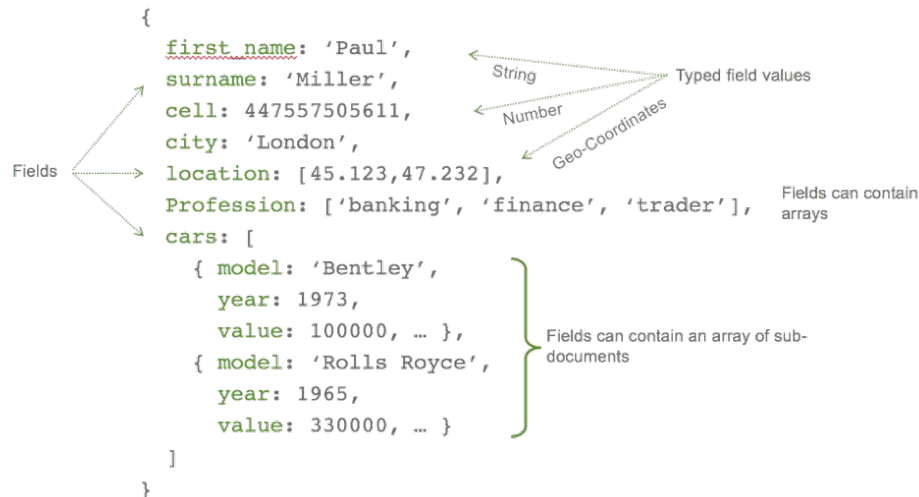
- It is a subclass of key-value databases
 - However, MongoDB relies on the structure of the data to extract metadata for optimization
- It is different from DB in how objects are stored
 - In DB, an object can be stored in multiple tables
 - In Document DB, the object is always stored in a single instance and every object can be different from other
 - In MongoDB documents are similar to JSON objects
- It supports CRUD operations

MongoDB main features

- Ad hoc queries
- Indexing
- Replication
- Load balancing
- File storage
 - See grid filesystem
- Aggregation
 - Built-in MapReduce
- Capped collections (circular queues)

Documents

- A record in MongoDB is a document, which is a data structure composed of field and value pairs. MongoDB documents are similar to JSON objects.



Collections

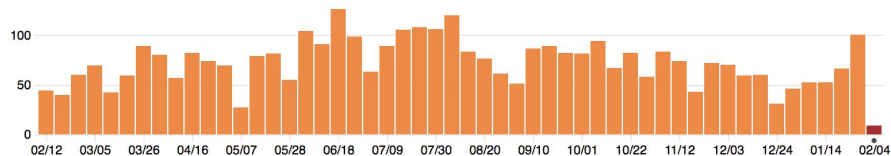
- MongoDB stores documents in collections. Collections are analogous to tables in relational databases. Unlike a table, however, a collection does not require its documents to have the same schema.



MongoDB, a closer look

● C++ 74.8% ● JavaScript 17.5% ● Python 4.4% ● Go 3.1% ● Shell 0.1% ● C 0.1%

- Written in C++
- Actively maintained (last commit was an day ago at the moment of writing)
- Average commit frequency is above 50 per week
- <https://github.com/mongodb/mongo>



Data as Documents

- Data is stored in BSON(Binary JSON)
- BSON documents contain fields, fields contain values (including arrays, binary data and sub documents)
- Closely aligned to the structure of objects in programming languages
 - Faster for developers to map data
- Keep all data for a given record in a single document

Data as Documents

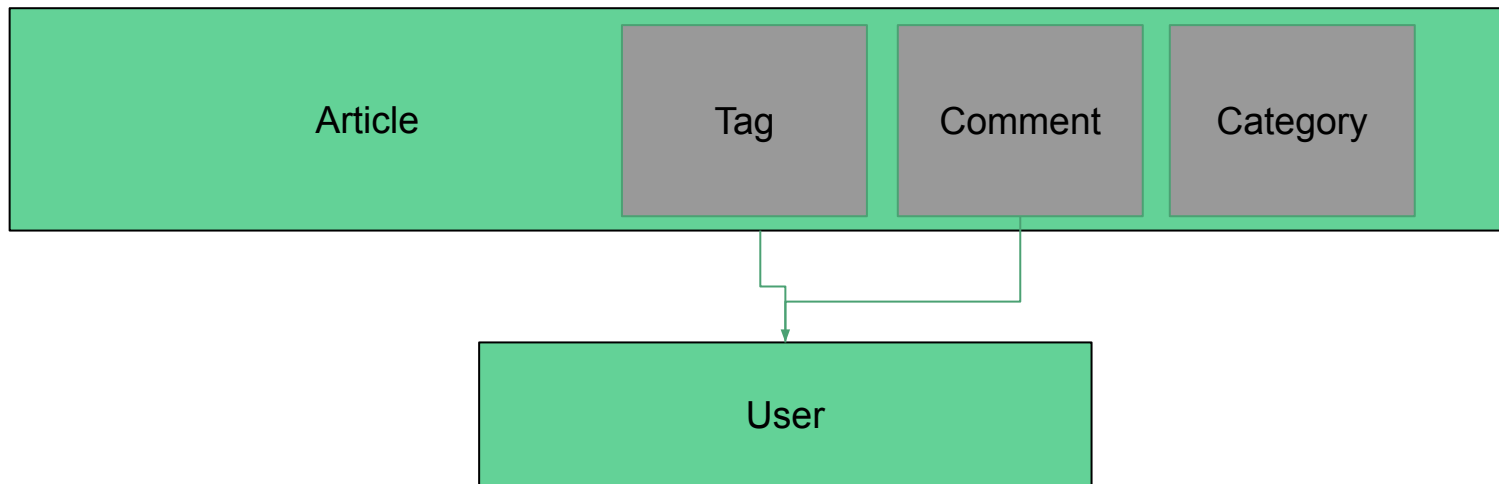
- Consider for example a data model for blogging application:
 - In RDB, it would involve having separate tables for **Users, Categories, Articles, Comments** etc
 - In MongoDB this could be modelled by two collections: one for Users and one for Articles:

Data as Documents

- Consider for example a data model for blogging application:
 - In RDB, it would involve having separate tables for Users, Categories, Articles, Comments etc
 - In MongoDB this could be modelled by two collections: one for Users and one for Articles
- Pros:
 - Data is more localized, no need to join
 - A single read can return the whole document
 - Higher scalability
- Cons:
 - What happens if you have many transactions?

Data as Documents

- Consider for example a data model for blogging application:
 - In RDB, it would involve having separate tables for Users, Categories, Articles, Comments etc
 - In MongoDB this could be modelled by two collections: one for Users and one for Articles:



Dynamic Data

- MongoDB allows variations in data structure
 - MongoDB allows developers to evolve data schema as the application becomes more mature
 - If a new field is needed to be introduced to a document, it can be introduced without affecting other documents and without taking the system offline

Dynamic Data

- A problem with having dynamic data - there may be situations when a strict schema must be guaranteed
- MongoDB provides schema validation within the database via syntax derived from the IETF JSON Schema standard
- Schema validation allows developers to create a flexible rules on which to save or reject a document
- Structure can be imposed on a subset of fields

Dynamic Data

Benefits of data validations

- Application logic simplification: With a strictly defined definition of what a collection looks like, there isn't a need for the application to handle the guarantees and associated errors.
- Control of data: Client applications must adhere to the rules set forth by the collection. No need to worry about data being updated or inserted that has incorrect field names or data types.
- Governmental compliance: There are applications and data models in a variety of industries and locales that require data to be stored in specific formats. For example, the EU General Data Protection Regulation in the European Union.

Query Model

Query types:

- **Key-value queries** return results based on any field in the document, often the primary key
- **Range queries** return results based on values defined as inequalities (e.g. greater than, less than or equal to, between)
- **Proximity queries** return results based on proximity criteria, intersection and inclusion as specified by a point, line, circle or polygon
- **Geospatial** return results based on proximity criteria
- **Search queries** queries return results in relevance order and in faceted groups, based on text arguments using Boolean operators (e.g., `AND`, `OR`, `NOT`)
- **Aggregation Pipeline** queries return aggregations and transformations of documents and values
- **JOINS and graph traversals**

Sharding

- Sharding is a way to achieve horizontal scalability
- MongoDB supports following:
 - **Range Sharding.** Documents are partitioned across shards according to the shard key value
 - **Hash Sharding.** Documents are distributed according to an MD5 hash of the shard key value
 - **Zone Sharding.** Provides the ability for DBAs and operations teams to define specific rules governing data placement in a sharded cluster

Consistency and availability model

- MongoDB provides ACID(atomicity, consistency, isolation, durability) properties at the document level.
 - The ACID ensure complete isolation as a document is updated; any errors cause the operation to roll back so that clients receive a consistent view of the document.
- MongoDB's Write Concerns allow to commit to the application only after they have been flushed to the journal file on disk
- Each query can specify the appropriate write concern, such as writing to at least two replicas in one data center and one replica in a second data center

Indexing

- MongoDB includes support for many types of secondary indexes that can be declared on any field in the document
- The list of supported indexes:
 - Unique Indexes
 - Compound Indexes
 - Array Indexes
 - TTL Indexes
 - Geospatial Indexes
 - Partial Indexes
 - Sparse Indexes
 - Text Search Indexes.

MongoDB, components & utils

COMPONENTS

- `mongod` - The database server.
- `mongos` - Sharding router.
- `mongo` - The database shell (uses interactive javascript).

UTILITIES

- | | |
|--------------------------|---|
| <code>mongodump</code> | - Create a binary dump of the contents of a database. |
| <code>mongoexport</code> | - Export the contents of a collection to JSON or CSV. |
| <code>mongoimport</code> | - Import data from JSON, CSV or TSV. |
| <code>mongofiles</code> | - Put, get and delete files from GridFS. |
| <code>mongostat</code> | - Show the status of a running mongod/mongos. |
| <code>bsondump</code> | - Convert BSON files into human-readable formats. |

Installing MongoDB

- Binaries are available for all major platforms (Linux, OS X, Windows)
 - Easy way
 - <https://docs.mongodb.com/tutorials/>
- You can also build from sources
 - <https://github.com/mongodb/mongo>
 - <https://github.com/mongodb/mongo/wiki/Build-Mongodb-From-Source>
 - Hard way, but you can grab the latest version
- Go ahead and install MongoDB on your laptop.
- Run with `mongod`

Importing data

Once you are done, import data from the following link:

<https://raw.githubusercontent.com/mongodb/docs-assets/primer-dataset/primer-dataset.json>

With the following command:

```
mongoimport --db test --collection restaurants --drop --file ~/downloads/primer-dataset.json
```

To import data into a mongod instance running on a different host or port, specify the hostname or port by including the `--host` and the `--port` options in your `mongoimport` command

Connecting to MongoDB from Python

- Install Python interface with
 - `pip install pymongo`
- Import MongoClient
 - `from pymongo import MongoClient`
- Connect to running MongoDB instance
 - `client = MongoClient("mongodb://hostname:port")`
 - `client = MongoClient("mongodb://localhost")` # will connect to localhost and default port 27017
- Access database of interest
 - `db = client['db_name']`

Query data

- Find data inside collection:

- `cursor = db.collection_name.find({key1: value, key2: value2 ...})`
- if no parameters are supplied, all data will be returned (as in `SELECT *`)
- cursor is a generator which can be iterated through using standard Python syntax

Useful links

- <https://www.mongodb.com/docs/compass/current/query/filter/>
- <https://www.postgresqltutorial.com/postgresql-tutorial/postgresql-recursive-query/>