NoSQL : Unleash the Power of MongoDB

Abhishek Bagga

24th September 2019

Abhishek Bagga

Solution Architect

abhishek.bagga@outlook.com

linkedin.com/in/abhishekbagga/

@abhishekbagga28

nagarro

Session Contents

NOSQL: What is NoSQL???

NOSQL: What is NotOnlySQL???

- Different Types of NoSQL Databases
  - **Document Store** – MongoDB, Elastic Search
  - **Wide Column Store** – Hadoop, Cassandra
  - **Key Value/ Tuple Store** – DynamoDB, Redis
  - **Graph Stores** – Neo4j, InfiniteGraph

NOSQL: Benefits

- NoSQL databases are more scalable, and provide superior performance
- NoSQL Data model addresses several issues that the relational model is not designed to address:
  - Large volumes of rapidly changing structured, semi-structured, and unstructured data
  - Agile sprints, quick schema iteration, and frequent code pushes
  - Object-oriented programming that is easy to use and flexible
  - Geographically distributed scale-out architecture instead of expensive, monolithic architecture


mongoDB

MongoDB: The database for modern applications

- General purpose
- Document-based
- Scalable
- Distributed

MongoDB: Features

- Rich JSON Documents
- Powerful Query Language
- All the power of a relational database, and more...
- Made for The Cloud

MongoDB: Features

- Rich JSON Documents
  - The most natural and productive way to work with data.
  - Supports arrays and nested objects as values.
  - Allows for flexible and dynamic schemas
- Powerful Query Language
- All the power of a relational database, and more...
- Made for The Cloud

MongoDB: Features

- Rich JSON Documents
- Powerful Query Language
  - Rich and expressive query language
  - Allows filter and sort by any field
  - Supports aggregations
  - Geo-based search, graph search, and text search.
  - Queries are easily composable (JSON)
- All the power of a relational database, and more...
- Made for The Cloud

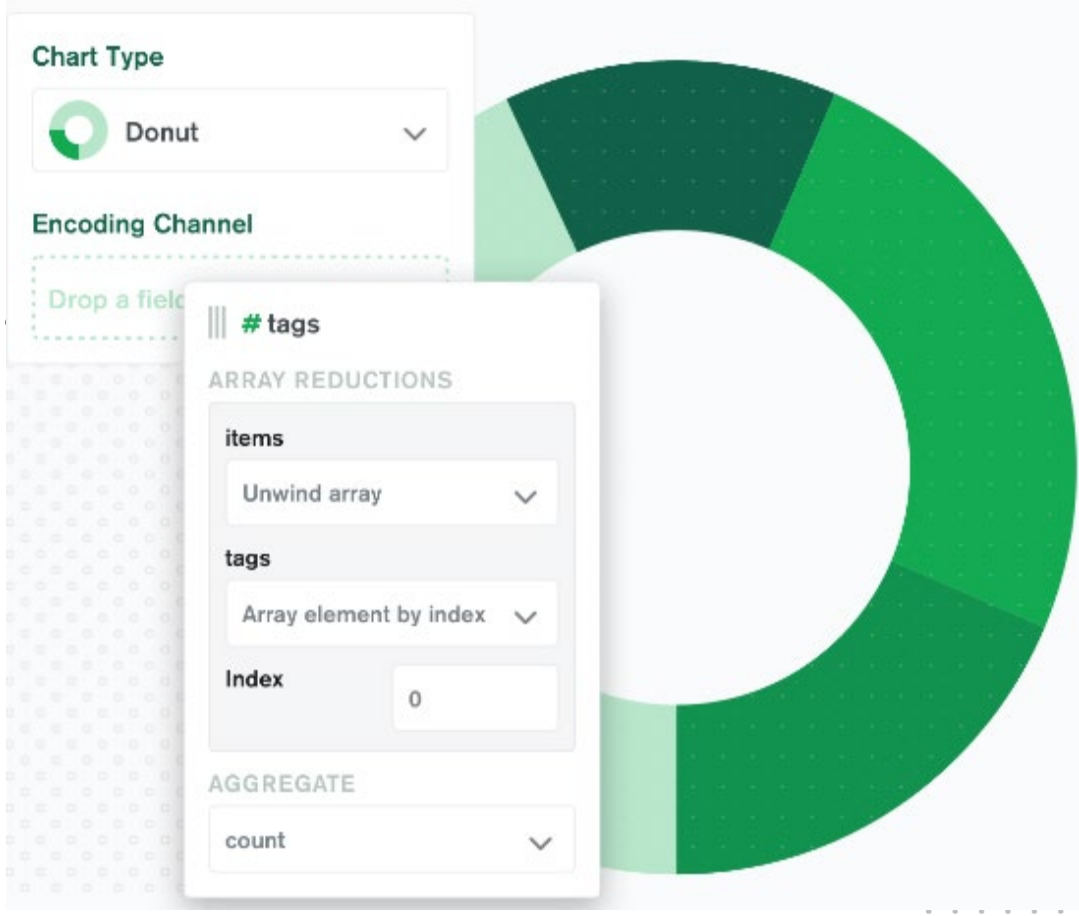
MongoDB: Features

- Rich JSON Documents
- Powerful Query Language
- All the power of a relational database, and more...
  - Full ACID transactions.
  - Support for joins in queries.
  - Two types of relationships instead of one:
    >reference and embedded.
- Made for The Cloud

```
session.start_transaction()
order = { line_items : [ { item : 5, quantity: 6 } ] }
db.orders.insertOne( order, session=session );
for x in order.line_items:
  db.inventory.update(
    { _id  : x.item } ,
    { $inc : { number : -1 * x.quantity } },
    session=session
  )
session.commit_transaction()
```

MongoDB: Features

- Rich JSON Documents
- Powerful Query Language
- All the power of a relational database, and more..
- Made for The Cloud
  - MongoDB Atlas
  - MongoDB Charts
  - MongoDB Stitch

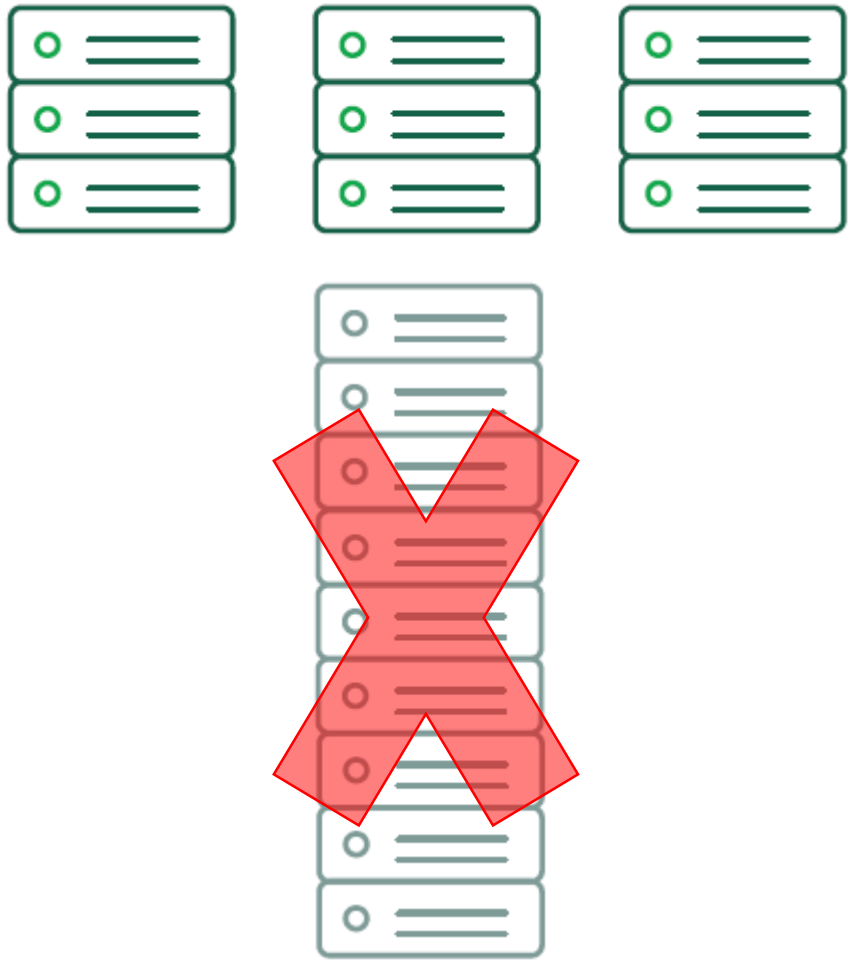


MongoDB: Major Advantages

- Highly Scalable
- Code/ Program Faster
- Query Faster
- Dynamic Schema

MongoDB: Major Advantages

- Highly Scalable
  - Scale Cheaper
  - As the database grows, Scale horizontally.
- Code/ Program Faster
- Query Faster
- Dynamic Schema



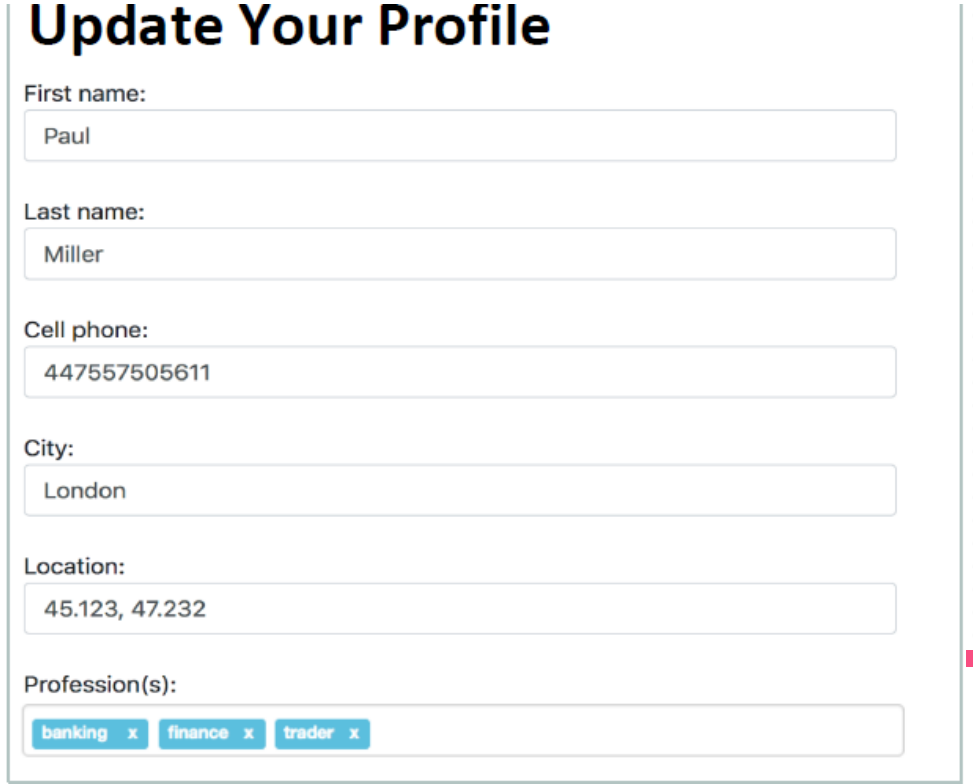
MongoDB: Major Advantages

- Highly Scalable
- Code/ Program Faster
  - Documents map to data structures in most popular languages
  - Avg 60% reduction in lines of code
- Query Faster
- Dynamic Schema

MongoDB: Major Advantages

- Highly Scalable
- Code/ Program Faster
  - Documents map to data structures in most popular languages

```
{
    first_name: "Paul",
    surname: "Miller",
    cell: "447557505611",
    city: "London",
    location: [45.123,47.232],
    profession: ["banking", "finance", "trader"],
}
```



MongoDB: Major Advantages

- Highly Scalable
- Code/ Program Faster
  - Documents map to data structures in most popular languages
  - Avg 60% reduction in lines of code

```
# UPDATE THE USER'S PROFILE IN THE DATABASE

### Since the user's data is stored in a single document, we only
have to make one update

result = db['Users'].update_one(
  {"_id": userId}, {"$set": user})
```

```
# UPDATE THE USER'S PROFILE IN THE DATABASE

### First update what is stored in the Users table

sql = "UPDATE Users SET first_name=%s, surname=%s, cell=%s,
city=%s, location_x=%s, location_y=%s WHERE (ID=%s)"
values = (
  user["first_name"],
  user["surname"],
  user["cell"],
  user["city"],
  user["location_x"],
  user["location_y"],
  userId)
mycursor.execute(sql, values)

mydb.commit()
```

MongoDB: Major Advantages

- Highly Scalable
- Code/ Program Faster
- Query Faster
  - No Expensive Joins
  - JSON based query language
- Dynamic Schema

```
{
    first_name: "Paul",
    surname: "Miller",
    cell: "447557505611",
    city: "London",
    location: [45.123,47.232],
    profession: ["banking", "finance", "trader"],
}
```

Users

ID	first_name	surname	cell	city	location_x	location_y
1	Paul	Miller	447557505611	London	45.123	47.232

Professions

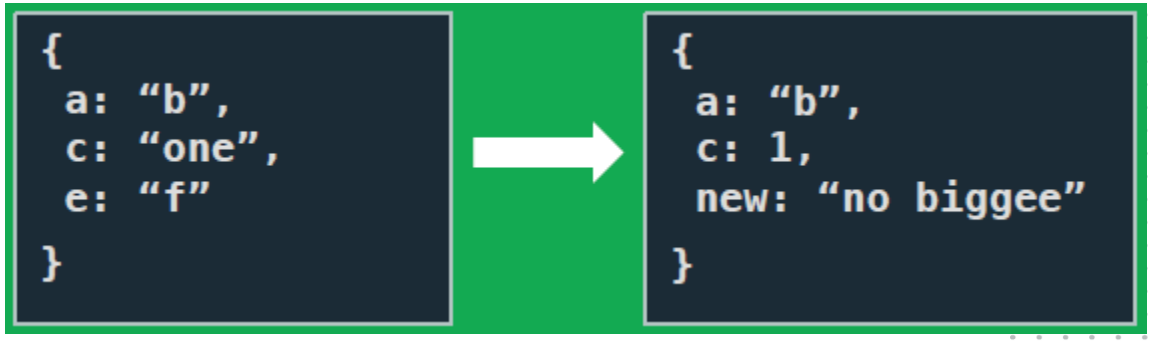
ID	user_id	profession
10	1	banking
11	1	finance
12	1	trader

MongoDB: Major Advantages

- Highly Scalable
- Code/ Program Faster
- Query Faster
  - No Expensive Joins
  - JSON based query language
- Dynamic Schema
  - Easily change the shape of your data as your app evolves

MongoDB: Major Advantages

- Highly Scalable
- Code/ Program Faster
- Query Faster
  - No Expensive Joins
  - JSON based query language
- Dynamic Schema
  - Easily change the shape of your data as your app evolves




```
ALTER TABLE `mydb`.`letters_table`
DROP COLUMN `e`,
ADD COLUMN `New` VARCHAR(45) NULL AFTER `C`,
CHANGE COLUMN `C` `C` INT NULL DEFAULT NULL ;
```

MongoDB: Major Advantages

```
{
    first_name: "Paul",
    surname: "Miller",
    cell: "447557505611",
    city: "London",
    location: [45.123,47.232],
    profession: ["banking", "finance", "trader"],
    cars: [
        {
            model: "Bentley",
            year: 1973
        },
        {
            model: "Rolls Royce",
            year: 1965
        }
    ]
}
```

```
{
    first_name: ”Lauren",
    surname: ”Schaefer",
    cell: ”1235552222",
    city: ”Lancaster",
    profession: [”software engineer", ”developer advocate"],
}
```

```
{
    first_name: ”Sydney",
    surname: ”Schaefer",
    city: ”Lancaster",
    school: ”Daisy’s Daycare”
}
```

- Not all documents in a collection need to have the same fields

MongoDB: Major Advantages

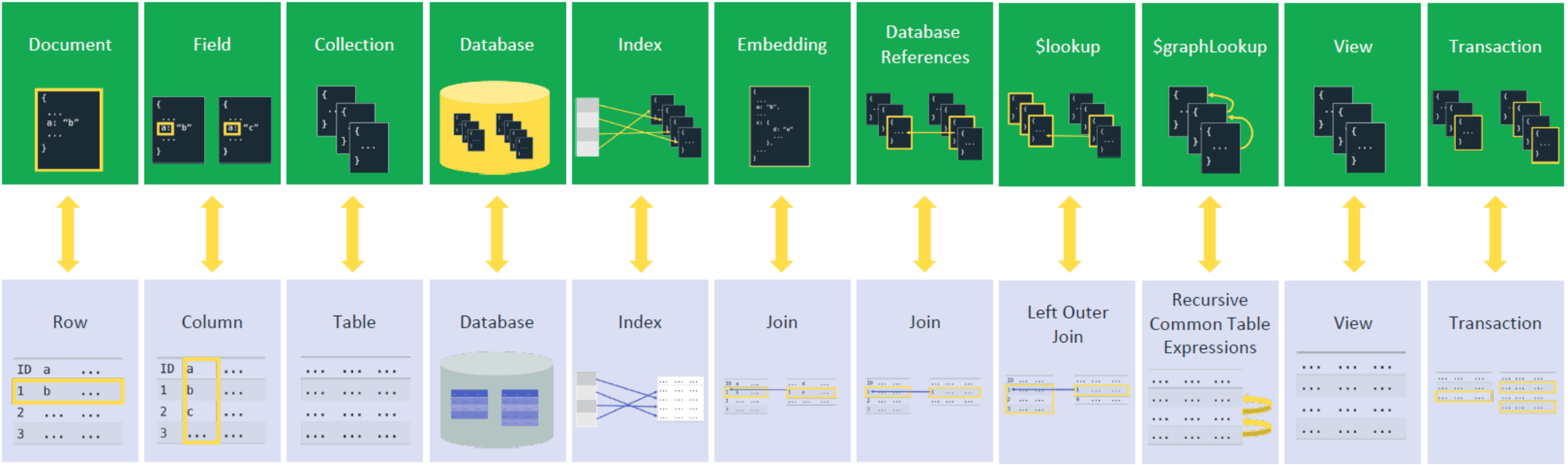
```
{
    _id: "Lauren_Schaefer",
    displayName: "Lauren Schaefer",
    numFollowers: 1310
    followers: [
       "naomi_pen",
       "kenwalger",
       "mylynn"
       ...
    ]
}
```

```
{
    _id: "Nick_Offerman",
    displayName: "Nick Offerman",
    numFollowers: 1730332
    followers: [
       "c_hotaling",
       "IAmJerdog",
       "ChloeCondon"
       ...
    ],
    has_extras: true
}
```

```
{
    _id: "Nick_Offerman_1",
    twitter_id: "Nick_Offerman",
    is_overflow: true,
    followers: [
       "StephenAtHome",
       "TheEllenShow",
       "hulu"
       ...
    ]
}
```

- Dynamic Schema
  - Easily change the shape of your data as your app evolves
  - Not all documents in a collection need to have the same fields

SQL to MongoDB Mapping



Courtesy: Lauren Schaefer

References

1. MongoDB Docs
   https://docs.mongodb.com/

2. SQL to MongoDB Mapping:
   https://docs.mongodb.com/manual/reference/sql-comparison/

SQL Terms, Functions, and Concepts	MongoDB Aggregation Operators
WHERE	$match
GROUP BY	$group
HAVING	$match
SELECT	$project
ORDER BY	$sort
LIMIT	$limit
SUM()	$sum
COUNT()	$count $sortByCount
JOIN	$lookup

MongoDB.LOCAL.LONDON

- MongoDB is coming to London on 25th September
- Full day of deep-dive technical sessions
- One-on-one consulting with MongoDB experts
- Learn what's new in MongoDB
- Registration: https://www.mongodb.com/local/london
- Use code 'Abhishek40' to get 40% off ticket prices
- Student can get a FREE TICKET
  - DM Naomi - @naomi_pen (Twitter)
  - DM Natasha Wilson - https://www.linkedin.com/in/natashawilson2/

Tip of the Iceberg

MongoDB has immense
Capabilities & Power

Q&A

Abhishek Bagga

abhishek.bagga@outlook.com

linkedin.com/in/abhishekbagga/

@abhishekbagga28