Database Applications
Lecture 8: NoSQL and MongoDB

Santha Sumanasekara
September 2022

RMIT UNIVERSITY

1

1

Announcements

- Mid-Semester Test – Thursday 22 Sep --
  - Must bring in your laptop (fully charged)
  - Topics covered in Week 1 – 7 are examinable. No PHP coding
  - Multiple-choice and short-answer questions
- Assignment 2 – will be released next week.

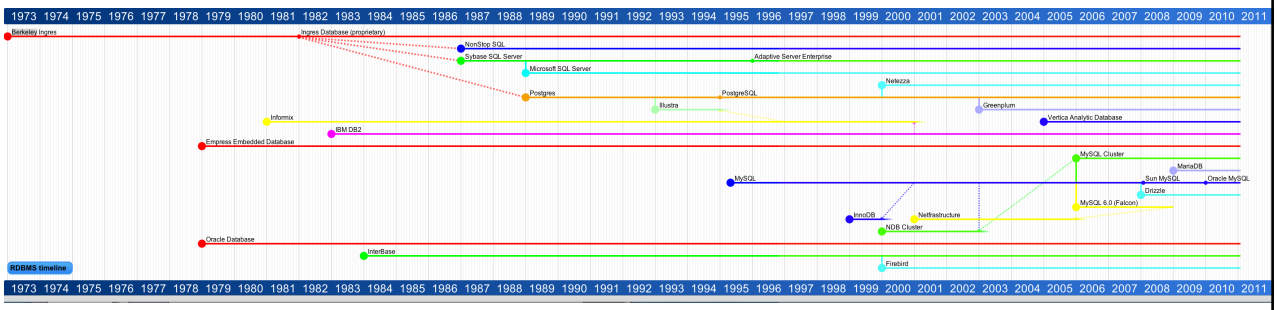
RMIT UNIVERSITY

2

2

A (not so) short history of DBMS

- http://upload.wikimedia.org/wikipedia/commons/2/22/RDBMS_timeline.svg



RMIT UNIVERSITY

3

3

Re-invent the wheel?

- In modern times, there were only very few re-inventions that made a great impact!
  - Propellor engine → Jet engine (in aircraft)
  - Film Camera → Digital Camera
  - …
- Is NoSQL the database equivalent to the jet engine in the aeronautical world?
- Why do you reinvent this new database model?

RMIT UNIVERSITY

4

4

Re-invent the wheel?

- Data is everywhere. The digital transformation drives us to the frontiers that never reached.
- The digital transformation demands for rich content and big data!
- The structured database model, that catered the data management needs for almost half a century, has its inherent limitations.

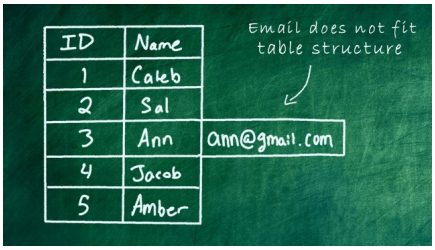
RMIT UNIVERSITY

5

5

An Example

- Consider the following example: Consider a database schema developed in early 1990s where use of email was not widespread, however, now it has become an essential piece of information required for a student database.



- This is not easily done with a relational database model.

RMIT UNIVERSITY

6

6

Another Example

- Consider the following two AirBnB listings:
- https://www.airbnb.com.au/experiences/79594
- https://www.airbnb.com.au/rooms/10082422

- How different are they?
- Can you easily store them in a table in a relational database?

- We will explore these databases in our future labs. (https://docs.atlas.mongodb.com/sample-data/sample-airbnb/#sample-document)

RMIT UNIVERSITY

7

7

NoSQL – Not just one model

https://en.wikipedia.org/wiki/NoSQL

NoSQL is a collective noun used for data models other than tabular relations.

Some of the common NoSQL data models are:

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

RMIT UNIVERSITY

8

8

NoSQL Database Types – Document Databases

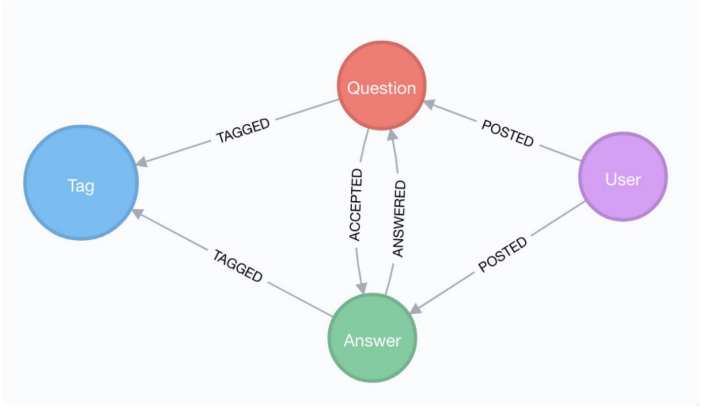
- **Document databases**: pair each key with a complex data structure known as a document. Documents can contain many different key-value pairs, or key-array pairs, or even nested documents.

```
{
  "_id": "10006546",
  "listing_url": "https://www.airbnb.com/rooms/10006546",
  "name": "Ribeira Charming Duplex",
  "summary": "Fantastic duplex apartment with three bedrooms,
            located in the historic area of Porto, Ribeira (Cube)...",
  "interaction": "Cot - 10 € / night Dog - € 7,5 / night",
  "house_rules": "Make the house your home...",
  "property_type": "House",
  "room_type": "Entire home/apt",
    ...
     ...
```

9

NoSQL Database Types – Graph Stores

- **Graph stores** are used to store information about networks of data, such as social connections. Graph stores include Neo4J and Giraph.



RMIT UNIVERSITY

10

10

NoSQL Database Types – Key-value stores

- **Key-value stores** are the simplest NoSQL databases. Every single item in the database is stored as an attribute name (or 'key'), together with its value. Examples of key-value stores are Riak and Berkeley DB. Some key-value stores, such as Redis, allow each value to have a type, such as 'integer', which adds functionality.

RMIT UNIVERSITY

11

11

NoSQL Database Types – Wide-column Stores

- **Wide-column stores** such as Cassandra and HBase are optimized for queries over large datasets, and store columns of data together, instead of rows.

RMIT UNIVERSITY

12

12

Benefits of NoSQL

- Dynamic Schemas
  - Relational databases require that schemas be defined before you can add data.
  - NoSQL databases are built to allow the insertion of data without a predefined schema.
  - Nicely fits with agile development process.
  - Data validation is delegated to the application.

RMIT UNIVERSITY

13

13

Benefits of NoSQL

- Auto-sharding
  - Partitioning can be done in the relational database systems, however, they require quite sophisticated mechanisms to ensure joins etc are executed efficiently.
  - NoSQL databases, on the other hand, usually support auto-sharding, meaning that they natively and automatically spread data across an arbitrary number of servers, without requiring the application to even be aware of the composition of the server pool.

RMIT UNIVERSITY

14

14

Benefits of NoSQL

- **Replication**
  - Most NoSQL databases also support automatic database replication to maintain availability in the event of outages or planned maintenance events.
  - Unlike relational databases, NoSQL databases generally have no requirement for separate applications or expensive add-ons to implement replication.

RMIT UNIVERSITY

15

15

MongoDB

What a name for a database!

The name of the database was derived from the word *humongous* to support the idea of processing large amount of data.

RMIT UNIVERSITY

16

MongoDB

- **MongoDB** is a cross-platform document-oriented database
- MongoDB uses JSON-like documents. A document is a record (sort of like a row in a structured table). For example, if we have a database for users, a document would be one individual user.
- A document is made out of a list of key-value pairs.

```
{
name: "sue",
age: 26,
status: "A",
Groups: [ "news", "sports" ]
}
```

RMIT UNIVERSITY

17

MongoDB – similar but different!

Relational	MongoDB
database	database
table	collection
row	document
column	field
CREATE TABLE people (user_id Varchar(30), age Number, status char(1), PRIMARY KEY (user_id));	db.people.insertOne({ user_id: "abc123", age: 55, status: "A" })

18

18

Demo time!

MongoDB
MongoDB Compass

RMIT UNIVERSITY

19

Insert a document

- In SQL:

```
INSERT INTO people (user_id, age, status)
    VALUES ('A1234',23,'A');
```

- In MongoDB Shell:

```
db.people.insertOne(
   { user_id: "A1234", age: 23, status: "A" }
)
```

RMIT UNIVERSITY

20

20

Insert a document

- In SQL:

```
INSERT INTO people (user_id, age, status)
    VALUES ('A1234',23,'A');
```

- In MongoDB Compass:

RMIT UNIVERSITY

21

21

A complex insertion

- In SQL: We **cannot** insert composite objects or arrays into tables

```
INSERT INTO people (user_id, name, address, age, sports)
    VALUES ('A1234','Sam', {1}{Happy St}{Happyville}{3999},
              23,['AFL', 'Cricket', 'Rugby']);
```

- In MongoDB Shell:

```
db.people.insertOne(
   { user_id: "A1234", name: "Sam",
     address: { no: 28, street: "Happy St",
                suburb: "Happyville"},
     age: 23, sports: ["AFL", "Cricket", "Rugby"] }
)
```

Address is a composite object. Sports is an array.

22

22

A complex insertion

- In MongoDB Compass:

Address is a composite object. Sports is an array.

RMIT UNIVERSITY

23

23

Find a document

- In SQL we use SELECT to find/ retrieve data. In MongoDB, we use find() method.

```
db.<<collection-name>>.find (<<query>>, <<projection>>)
```

This specifies the conditions. Similar to WHERE clause in SQL

- Next few slides demonstrate how queries and projections are formed.

This specifies which fields to display. Similar to SELECT clause in SQL.

RMIT UNIVERSITY

24

24

Find a document

- In SQL:

```
SELECT *
    FROM people;
```

- In MongoDB Shell:

```
db.people.find()
```

RMIT UNIVERSITY

25

25

Find some fields in a document

- In SQL:

```
SELECT ROWID, user_id, status
    FROM people;
```

- In MongoDB Shell:

```
db.people.find(
    { },
    { user_id: 1, status: 1 }
)
```

Leave this empty, as there isn't any filtering condition.

If you do not want to see the document ID, add _id: 0 to the projection (field list).

RMIT UNIVERSITY

26

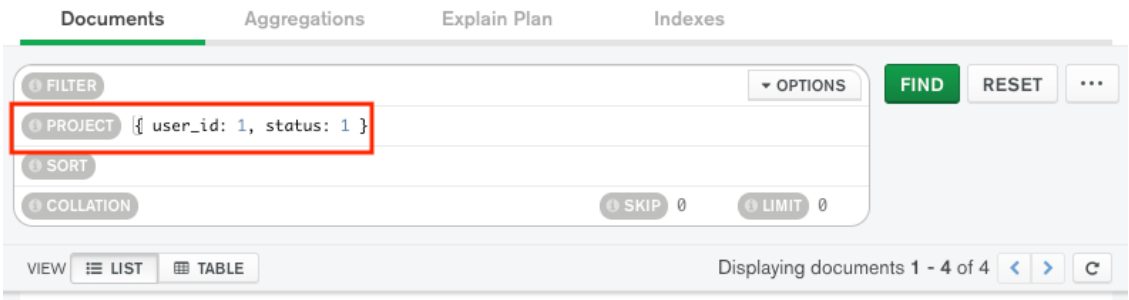
26

Find a document

- In SQL:

```
SELECT ROWID, user_id, status
    FROM people;
```

- In MongoDB Compass:



27

Filter a document

filtering condition.

- In SQL:

```
SELECT user_id, status
    FROM people
    WHERE status = 'A';
```

- In MongoDB Shell:

```
db.people.find(
    { status: "A" },
    { user_id: 1, status: 1, _id: 0 }
)
```

RMIT UNIVERSITY

28

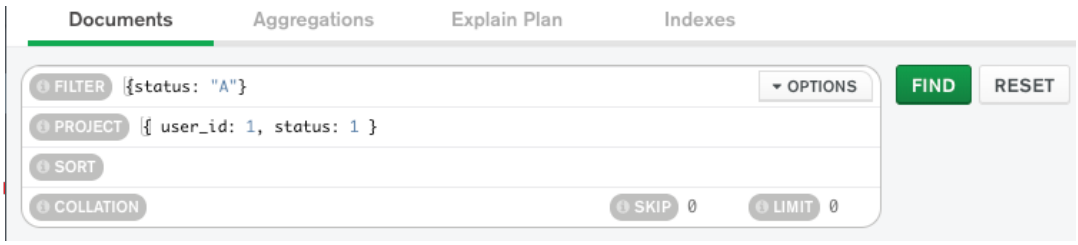
28

Filter a document

- In SQL:

```
SELECT user_id, status
    FROM people
    WHERE status = 'A';
```

- In MongoDB Compass:



29

29

Filter a document – with multiple conditions

- In SQL:

```
SELECT *
    FROM people
    WHERE status = 'A' AND
          age = 25;
```

AND is the default.

- In MongoDB Shell:

```
db.people.find(
    { status: "A",
      age: 25 }
)
```

RMIT UNIVERSITY

30

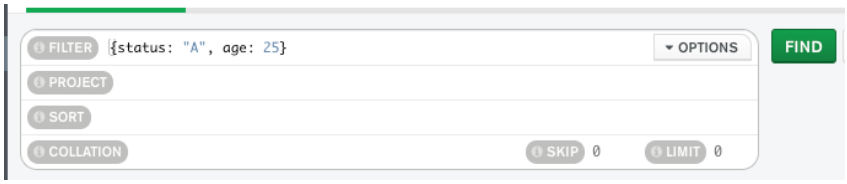
30

Filter a document

- In SQL:

```
SELECT *
    FROM people
    WHERE status = 'A' AND
          age = 25;
```

- In MongoDB Compass:



RMIT UNIVERSITY

31

Filter a document – with OR conditions

- In SQL:

```
SELECT *
    FROM people
    WHERE status = 'A' OR
          age = 25;
```

OR is applied to an array of conditions. [ ] denotes an array.

- In MongoDB Shell:

```
db.people.find(
    { $or: [ { status: "A" } , { age: 25 } ] }
)
```

RMIT UNIVERSITY

32

Filter a document – with OR conditions

- In SQL:

```
SELECT *
    FROM people
    WHERE status = 'A' OR
          age = 25;
```

- In MongoDB Compass:

RMIT UNIVERSITY

33

33

Filter a document – with "Not Equal"

$ne means "not equal"

- In SQL:

```
SELECT *
    FROM people
    WHERE  age <> 25;
```

- In MongoDB Shell:

```
db.people.find(
     { age: { $ne: 25 } }
)
```

RMIT UNIVERSITY

34

34

Filter a document – middle-aged people!

- In SQL:

```
SELECT *
    FROM people
    WHERE  age > 35 AND age <= 60;
```

- In MongoDB Shell:

```
db.people.find(
     { age: { $gt: 35, $lte: 60 }}
)
```

$gt: greater than
$gte: greater than or equal
$lt: less than
$lte: less than or equal

RMIT UNIVERSITY

35

35

Filter a document – Partial Matches

- In SQL:

```
SELECT *
    FROM people
    WHERE  name LIKE '%Sam%;
```

- In MongoDB Shell:

```
db.people.find(
     {name: /Sam/}
)
```

More complex regular expressions are possible. To be discussed later. E.g.
`{name: {$regex: /Sam/ } }`

RMIT UNIVERSITY

36

36

Filter a document – Using Composite Fields

- In SQL: Cannot be done!

```
SELECT *
    FROM people
    WHERE  address.suburb = 'Happyville';
```

- In MongoDB Shell:

```
db.people.find(
     {"address.suburb": "Happyville"}
)
```

Make sure to use double quotes.

RMIT UNIVERSITY

37

37

Filter a document – Using Array Values

- In SQL: Cannot be done!

```
SELECT *
    FROM people
    WHERE  sport = ['AFL', 'Cricket']
```

- In MongoDB Shell:

```
db.people.find(
    {sports:"AFL", "Cricket"}
)
```

Return documents that exactly contain this array.

RMIT UNIVERSITY

38

38

Aggregations -- count()

- In SQL:

```
SELECT COUNT(*)
    FROM people;
```

- In MongoDB Shell:

```
db.people.find().count()
```

Object-oriented. Two ways to do it:
`db.collection.find().count()`
`db.collection.count()`

RMIT UNIVERSITY

39

Sorting

- In SQL:

```
SELECT *
    FROM people
    ORDER BY name;
```

- In MongoDB Shell:

```
db.people.find().sort({name: 1})
```

```
db.people.find().sort({name: -1})
```

Descending order

RMIT UNIVERSITY

40

Sorting

- In SQL:

```
SELECT *
    FROM people
    ORDER BY name;
```

- In MongoDB Compass:

RMIT UNIVERSITY

41

41

Further Readings

- https://docs.mongodb.com/manual/reference/sql-comparison/
- https://docs.mongodb.com/manual/tutorial/insert-documents/
- https://docs.mongodb.com/manual/tutorial/query-documents/

RMIT UNIVERSITY

42

42

Next Week

- Mid-semester Test
- We will continue MongoDB discussion in Week 9. We discuss further on aggregations, etc.

RMIT UNIVERSITY

43

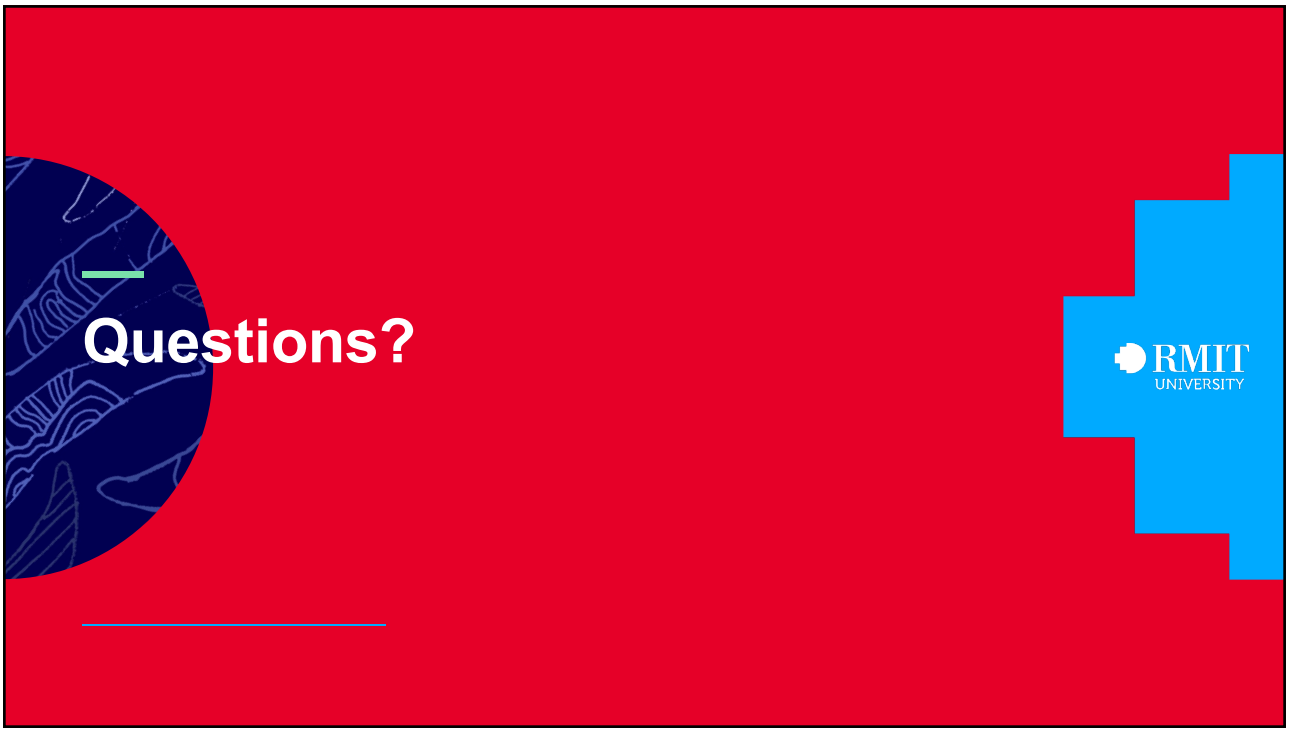
43

Menti-time!

Menti.com
Code: to be supplied

RMIT UNIVERSITY

44



45