FORT HAYS STATE
UNIVERSITY

Forward thinking. World ready.

Back-end development II
10
Understating MongoDB

Ravindra Ramsey Muvva
Department of Informatics
Robbins College of Business and Entrepreneurship

NoSQL

Not only SQL

- **NoSQL database** does not adhere to the **relational model**.
  This is evident in that MongoDB has no fixed schema, no tables, columns, or rows.
- NoSQL databases are driven by the needs of big data.
- NoSQL databases tend to be scalable as well as distributed.

NoSQL databases

Graph: A database technology designed in an almost visual manner, where basic entities that need to be stored are represented as node.

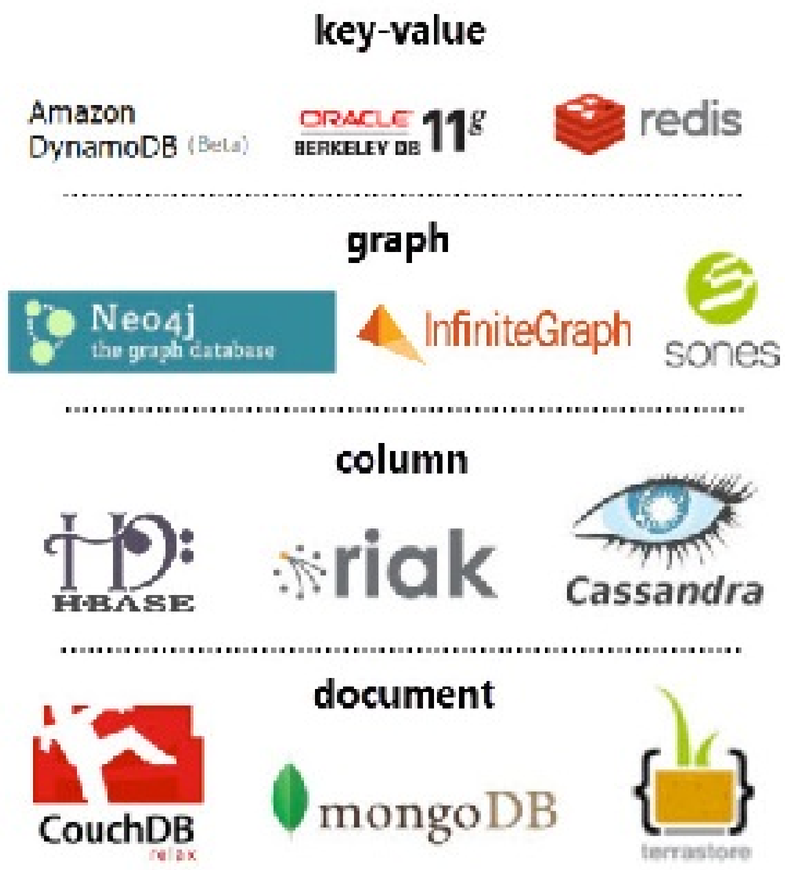
Key/Value: Data is stored as keys and values.

Wide column: This technology is also analogous to an array with key/value pairs.

Document: The document-style database uses what amounts to an object as its smallest logical unit.

Different Technologies

- HBase’s column structure
- Redis’s key/value structure
- Neo4j’s graph structure.



MongoDB

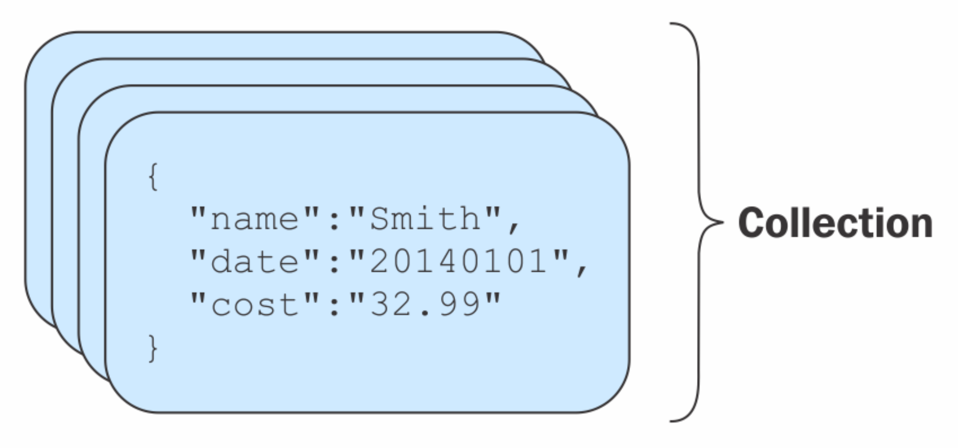
Understanding MongoDB

Document Oriented Database: A document-oriented database replaces the concept of a “row” with a more flexible model, the “document.” By allowing embedded documents and arrays, the document-oriented approach makes it possible to represent complex hierarchical relationships with a single record.

Schema-less: There are also no predefined schemas a document’s keys and values are not of fixed types or sizes. Without a fixed schema, adding or removing fields as needed becomes easier

Collection

- Grouping of Documents
- Similar to Table in SQL

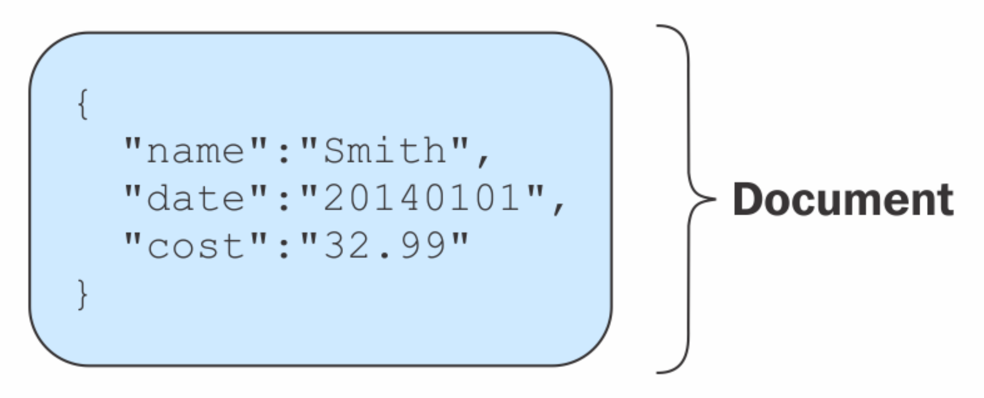


```
{
  "name":"Smith",
  "date":"20140101",
  "cost":"32.99"
}
```

Documents

- A document is a representation of a single entity of data in the MongoDB database.

- MongoDB, documents can contain embedded subdocuments, thus providing a much closer inherent data model to your applications.



```
{
  "name":"Smith",
  "date":"20140101",
  "cost":"32.99"
}
```

BSON

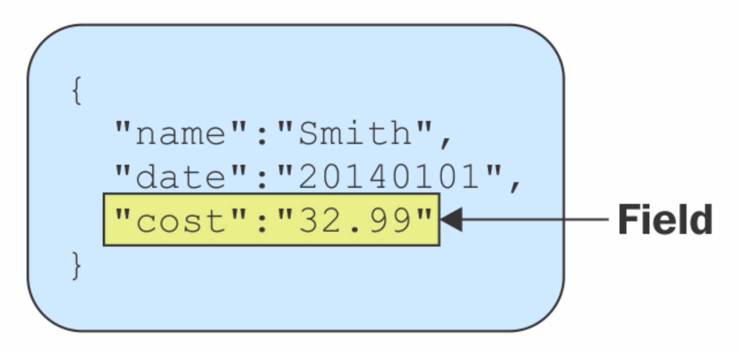
- MongoDB that represent documents are stored as **BSON**, which is a lightweight binary form of JSON

- These **field: value** pairs define the values stored in the document

- Document can contain strings, integers, arrays, and objects, just like a JavaScript object

- The field names cannot contain null characters,
  . (dots), $ (dollar signs), and **_id**

- The maximum size of a document in MongoDB is 16MB



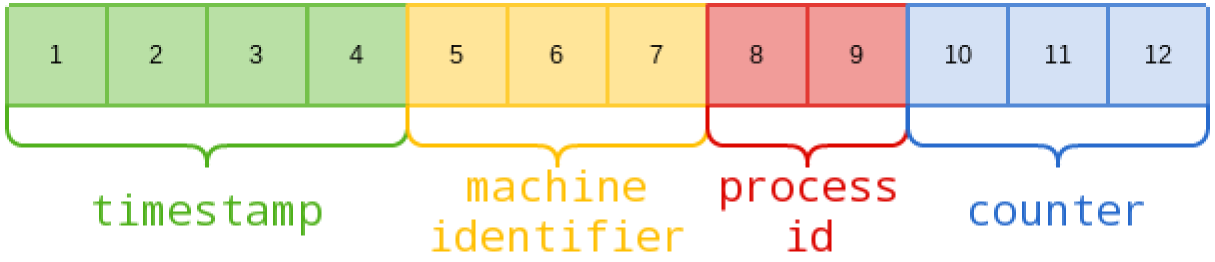
```
{
  "name":"Smith",
  "date":"20140101",
  "cost":"32.99"
}
```

Reference: *http://bsonspec.org/*

Object ID

ObjectId('5b47108b533b8406ac227798’)

- A 4-byte value representing the seconds since the last epoch
- A 3-byte machine identifier
- A 2-byte process ID
- A 3-byte counter, starting with a random value



1 2 3 4 5 6 7 8 9 10 11 12

timestamp

machine identifier

process id

counter

Data Types

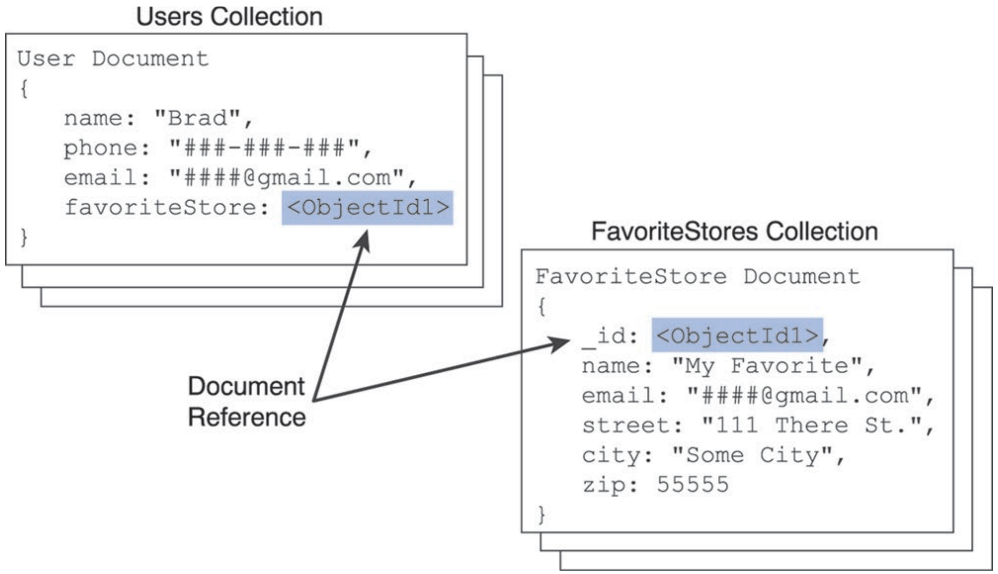
BSON data format provides several different types that are used when storing the JavaScript objects to binary form.

Type	Number
Double	1
String	2
Object	3
Array	4
Min Key	-1
Max Key	127

Reference: https://docs.mongodb.com/manual/reference/bson-types/

Data Normalization

- Data normalization is the process of organizing documents and collections to minimize redundancy and dependency.
- The advantage of normalizing data is that the database size will be smaller because only a single copy of an object will exist in its own collection instead of being duplicated on multiple objects in a single collection.



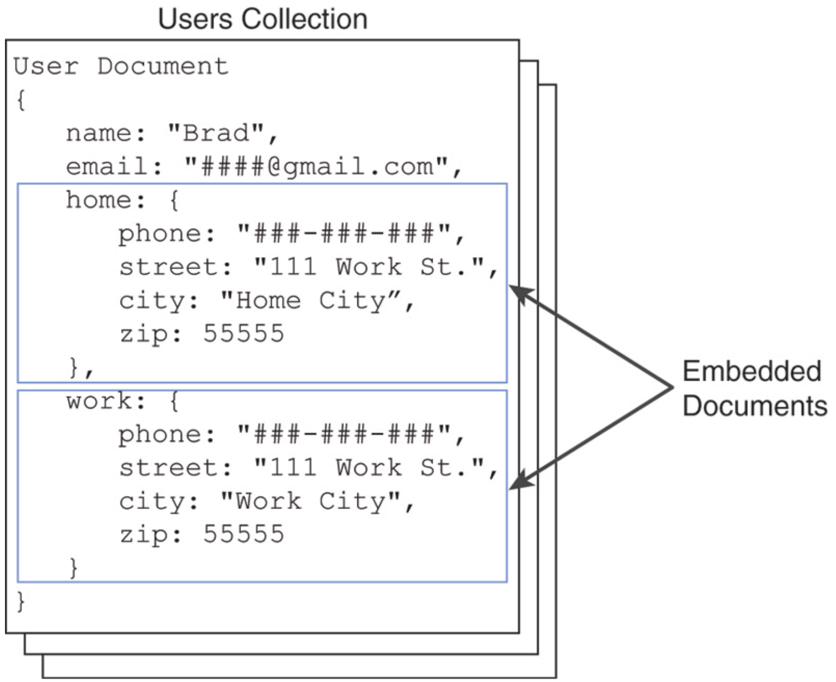
```
User Document
{
    name: "Brad",
    phone: "###-###-###",
    email: "####@gmail.com",
    favoriteStore: <ObjectId1>
}
```

```
FavoriteStore Document
{
    _id: <ObjectId1>,
    name: "My Favorite",
    email: "####@gmail.com",
    street: "111 There St.",
    city: "Some City",
    zip: 55555
}
```

Denormalizing Data

- Denormalizing data is the process of identifying sub-objects of a main object that should be embedded directly into the document of the main object.

- The major advantage of denormalized documents is that you can get the full object back in a single lookup without the need to do additional lookups to combine sub-objects from other collections.

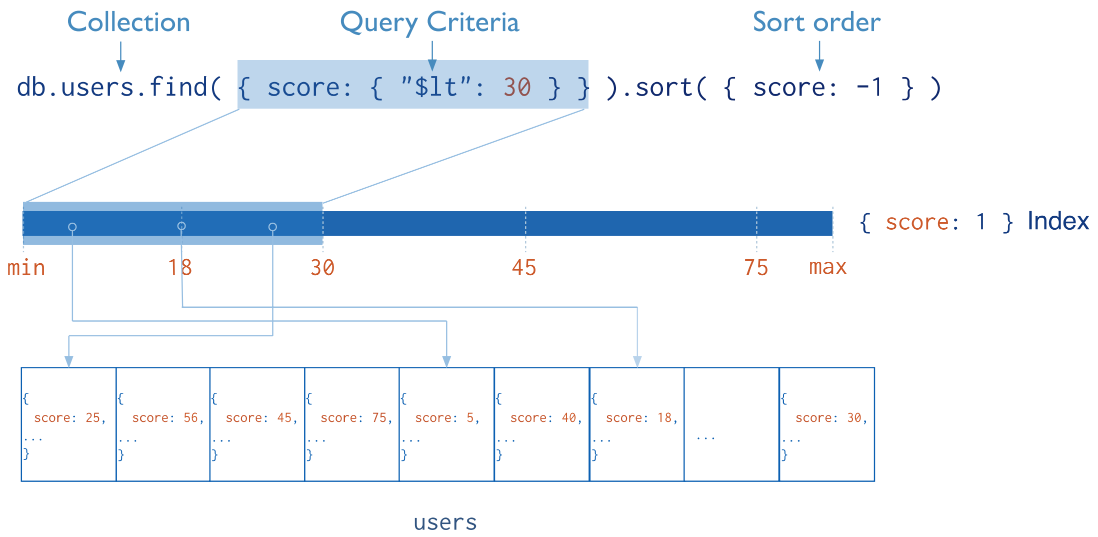


```
User Document
{
   name: "Brad",
   email: "####@gmail.com",
   home: {
      phone: "###-###-###",
      street: "111 Work St.",
      city: "Home City”,
      zip: 55555
   },
   work: {
      phone: "###-###-###",
      street: "111 Work St.",
      city: "Work City",
      zip: 55555
   }
}
```

Features of MongoDB

Indexing

- Indexes improve performance for frequent queries by building a lookup index that can be easily sorted.
- The _id property of a collection is automatically indexed on since it is a common practice to look items up by ID.

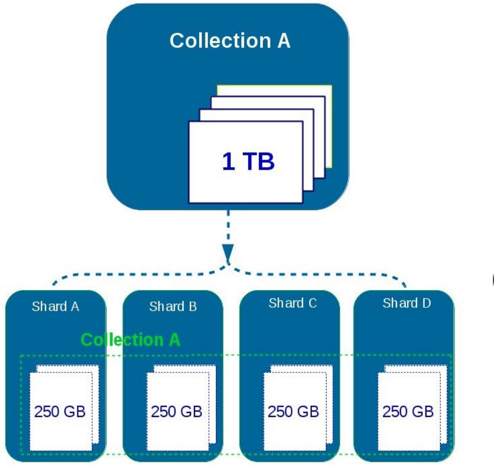


```
db.users.find( { score: { "$lt": 30 } } ).sort( { score: -1 } )
```

Reference: https://docs.mongodb.com/manual/indexes/

Sharding

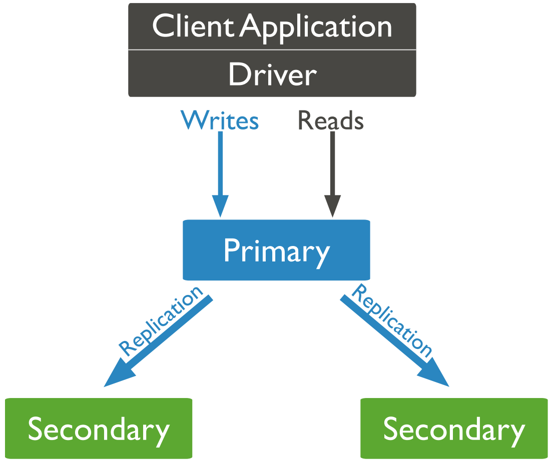
- Sharding is the process of slicing up large collections of data that can be split between multiple MongoDB servers in a cluster.
- Each MongoDB server is considered a shard.
- This provides the benefit of using multiple servers to number of requests to a large system, thus providing your database.



Reference: https://docs.mongodb.com/manual/sharding/

Replication

- Replication is the process of duplicating data on multiple MongoDB instances in a cluster.
- When considering the reliability aspect of your database, you should implement replication to ensure that a backup copy of critical data is always readily available.



Reference: https://docs.mongodb.com/manual/replication/

Aggregations

MongoDB provides an aggregation framework based on the concept of data processing pipelines. Aggregation pipelines allow you to build complex analytics engines by processing data through a series of relatively simple stages on the server side and with the full advantage of database optimizations.

$project —Specify fields to be placed in the output document (projected).
$match —Select documents to be processed, similar to find().
$limit —Limit the number of documents to be passed to the next step.
$skip —Skip a specified number of documents.
$unwind —Expand an array, generating one output document for each array entry.
$group —Group documents by a specified key.
$sort —Sort documents.
$geoNear —Select documents near a geospatial location.
$out —Write the results of the pipeline to a collection.
$redact —Control access to certain data.

https://docs.mongodb.com/manual/aggregation/

Installing MongoDB

MongoDB

MongoDB Download Center Reference:
https://www.mongodb.com/download-center/community

Mongo shell method Reference:
https://docs.mongodb.com/manual/reference/method/#native

In-class Exercise

What are the Important things to consider when designing a MongoDB database

Example:

What are the basic objects that my application will be using?
What is the relationships between the different object types?
How often will new objects be added to the database?
How often will objects be deleted from the database?
How often will objects be changed?