UCS677: Data Engineering

Course Instructor: Dr. Mandeep Kaur

Syllabus

1. Getting started with MongoDB
2. CRUD operations
3. Collections
4. Indexes
5. Sharding and Replication

Course Learning Outcomes (CLOs) / Course Objectives (COs):

1. Learn NoSQL, Mongo DB, and Javascript.
2. Apply basic concepts of indexing and collections using MongoDB.
3. Formulate the basic usage of database such as graph-based database, creating and dropping of collections, and indexing and solving real-world problems.
4. Able to handle and update multiple documents using Mongo DB.
5. Able to manage database for availability and performance, and scaling of database.

L	T	P	Cr
2	0	2	3.0

Marks Distribution:

EST : 40 marks

MST : 30 marks

Quiz : 15 marks

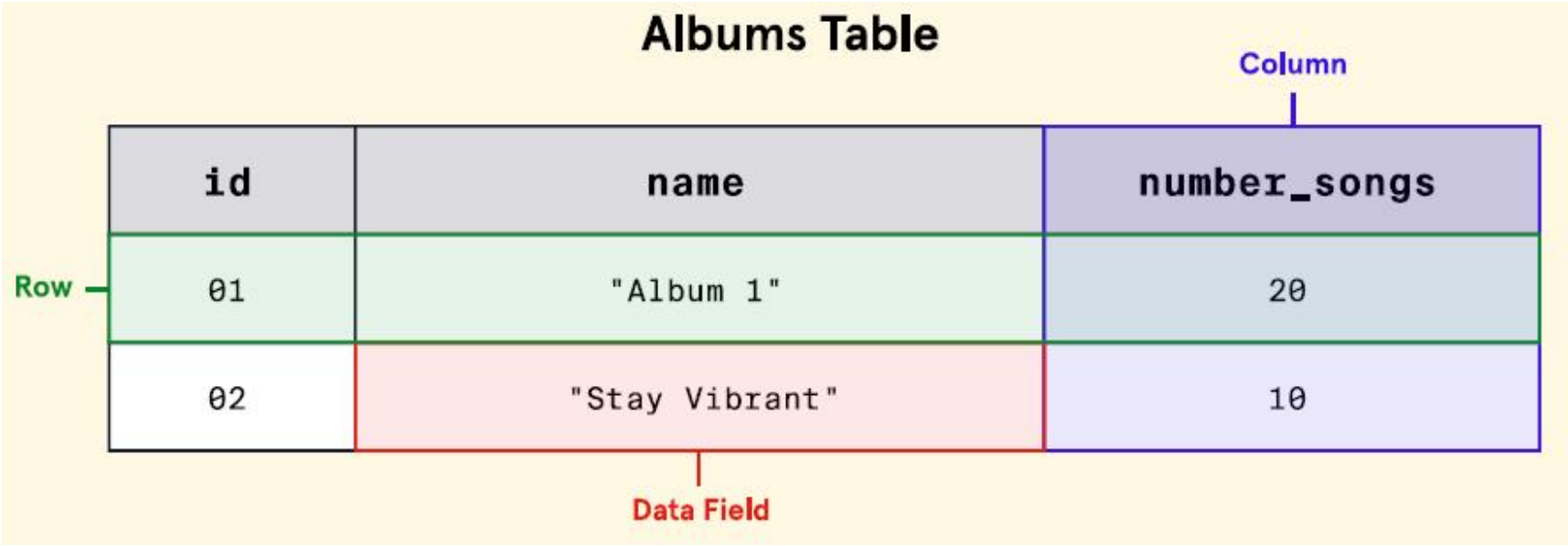
Lab Project : 15 marks

Introduction

- **Databases** are systems that store, modify, and access collections of information electronically.
- **Data Types:** Databases can store a variety of data types including strings, numeric data, booleans, dates and times, and files of various extensions.
- **Database Management System (DBMS):** A database management system (DBMS) is software designed to manage a database.
- Database management systems allow developers to communicate with a database, via code or a graphical user interface, to perform database operations and administrative actions such as user management.

Tabular Form

- Tabular form refers to a database structure that stores data as a collection of tables with each table consisting of several rows and columns.



Relational or SQL Databases

- Relational databases organize data into individual tables via rows and columns.
- Each column represents an attribute of the data stored and each row is an individual record of the data.
- Many relational databases use Structured Query Language (SQL) to store and retrieve data, and as such are often referred to as SQL databases.

Relational Database Relationship Types:

- “one-to-one”, “one-to-many”, and “many-to-many”

Database Schema:

- A database schema is a blueprint that outlines how a database’s data will be structured and organized. It is always a good idea to have a database schema in place before actually inserting data.
- Relational databases typically follow predefined schemas. Developers outline the tables and their associated relationships before inserting any data.

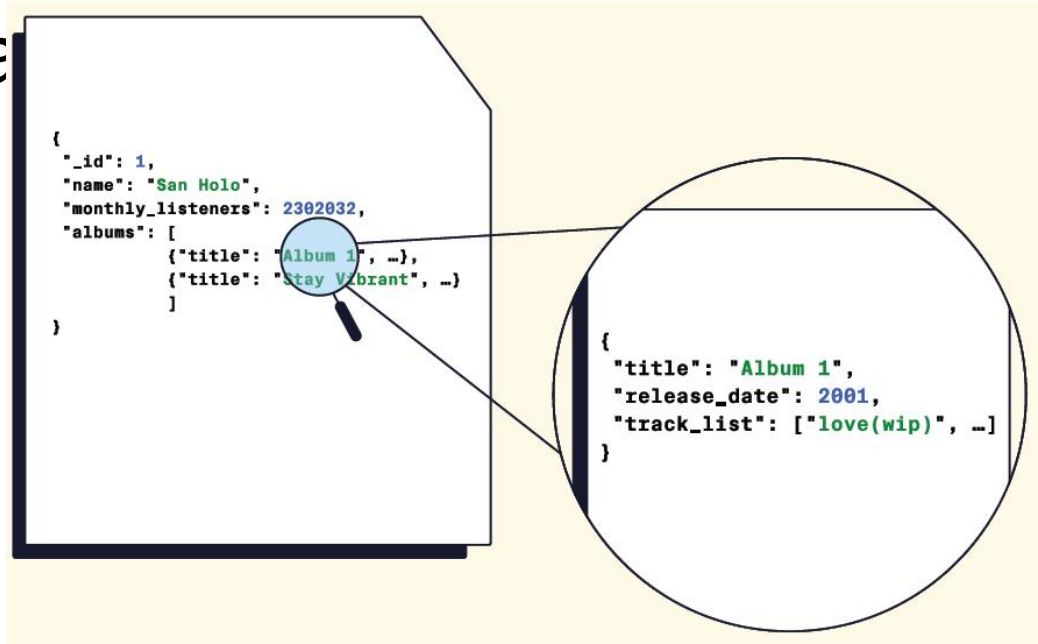
SQL Databases: Disadvantages

Two considerable disadvantages of working with relational databases are:

- They can be costly to set up, scale, and maintain.
- The organization of data into tables consumes a great deal of space which can negatively impact cost and performance.

Non-Relational or NoSQL Databases

- Non-relational, or NoSQL, databases is an umbrella term that refers to any database that does not follow the relational model provided by traditional database management systems.
- Databa N format.



NoSQL

Non SQL

Not only SQL

Employee

Empid	Name	Age	Contact no	location

NOSQL

Empid = 01
Name = ABC
Age = 36

- In SQL, unnecessarily, it takes more space

Empid = 02
Location = Indra
Qualificatim = B.Tech

NoSQL

- It does not have any fixed scheme
- It is easy to scale (Horizontal scaling)
- It is needed for distributed data storage
- Used for Bigdata and real time web app (Google, Twitter, Facebook)

Empid = 03
Name = XYZ
Age = 32

features of NoSQL

(1) Non-relational

(a) Never follows the relational model
(b) Never provide tables with fixed column & records
(c) No complex features like query language, join, ACID

(2) Scheme free

(a) Either schema-free or related scheme
(b) Do not require the definition of the scheme of data

(3) Distributed

Auto scaling

(4) NoSQL query language

(No standard based

1.2 Benefits of NoSQL

NoSQL vs. RDBMS

Criteria	NoSQL	RDBMS
Types	Different types available, according to the use case	One type (SQL Database) with minor variations
Development	Developed in late 2000s to address the challenges associated with SQL databases, like scalability, multi-structured data, distribution and development sprints	Developed in 1970s to work with the first wave of data storage applications
Development model	Open-source	Mix of open-source (Postgres, MySQL) and closed-source (Oracle)
Storage	Depends on database type For example, key-value stores are similar to SQL databases, but have only two columns ('key' and 'value'), with more complex information sometimes stored as BLOBs within the 'value' columns.	Data is typically stored in a relational model where columns with data points and rows with all the information concerning a single entity. Similar to a spreadsheet.
Schemas	Schemas are dynamic, information can be updated on the go. Dissimilar data can also be stored together as necessary. Only for some databases, e.g., wide-column stores, adding new fields is challenging.	Structure and data types are predefined, where the columns must be determined and locked before data is entered and cannot be amended later without taking the DB offline and modifying the entire database.

NoSQL vs. RDBMS (Contd.)

Criteria	NoSQL	RDBMS
ACID Compliance	NoSQL emphasizes performance over data integrity and most NoSQL systems compromise on ACID compliance for performance.	SQL databases default to enabling ACID compliance though most offer options to favor performance over data integrity for some operations.
Access	Access is allowed in well-defined and narrow patterns which make performance and scalability dependable and expected.	Not known beforehand and hence requires assumptions that are then translated into index definitions.
Scaling	Horizontally, i.e., while adding capacity, a database administrator can simply add more commodity servers or cloud instances. The database automatically spreads data across servers as necessary.	Vertically, meaning a single server must be made increasingly powerful in order to deal with increased demand. It is possible to spread SQL databases over many servers, but significant additional engineering is generally required, and core relational features such as JOINs, referential integrity and transactions are typically lost.
Data Manipulation	Done by object-oriented APIs.	Using Select, Insert, and Update statements of the language used.
Consistency	Some databases offer strong consistency, where others offer eventual consistency.	Can be configured for strong consistency.

Types of Databases

Key-Values Stores

A key-value NoSQL database uses individual records consisting of:

- Name, which is the key
- Data, which is the value

The data is assigned to a name when stored.

Features

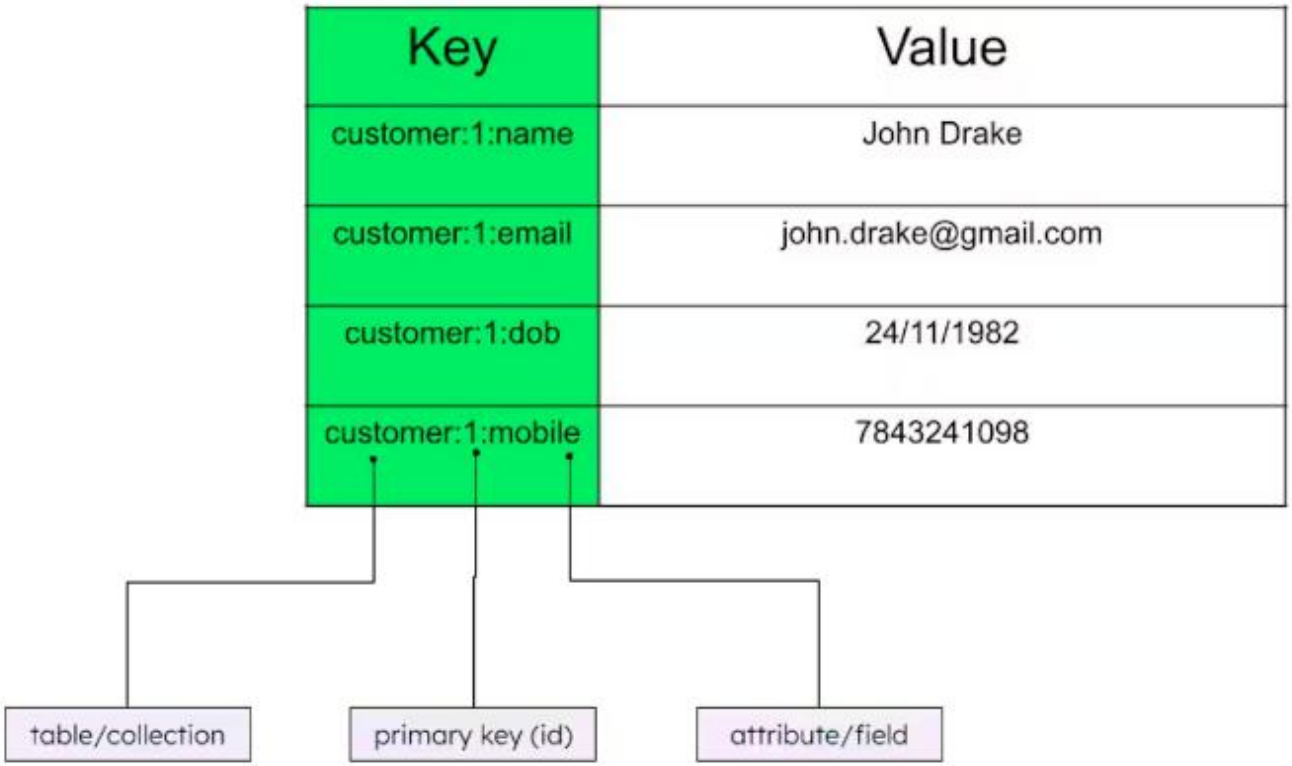
- Similar to relational databases, but there is no schema and the value of the data is opaque.
- Values are identified using a key, values can be numbers, strings, counters, JSON, XML, HTML, binaries, images, short videos, and more.
- Application has complete control over the data stored in the value.

Use Cases

- For storing user session data
- Maintaining schema-less user profiles
- Storing user preferences
- Storing shopping cart data

Examples

- Redis, MemcacheDB and Riak



- put(key, value): insert or update a value into the database.
- get(key): read a value from the database.
- delete(key): delete a value from the database.

customer_id	name	email	dob	mobile
1	John Drake	john.drake@gmail.com	24/11/1982	7843241098
2	Mary Chile	mary.chile@outlook.com	05/06/1981	8903424531
3	Mac Adams	mac_1979@gmail.com	23/04/1979	0920421454
4	Jill Smith	jellyjill@gmail.com	14/02/1987	8795092014

Key	Value

customer_1	{
	"name": "John Drake",
	"email": "john.drake@gmail.com",
	"dob": "24/11/1982",
	"mobile": 7843241098
	}

customer_2	{
	"name": "Mary Chile",
	"email": "mary.chile@outlook.com",
	"dob": "05/06/1981",
	"mobile": 8903424531
	}

customer_3	{
	"name": "Mac Adams",
	"email": "mac_1979@gmail.com",
	"dob": "23/04/1979",
	"mobile": 0920421454
	}

customer_4	{
	"name": "Jill Smith",
	"email": "jellyjill@gmail.com",
	"dob": "14/02/1987",
	"mobile": 8795092014
	}

Wide-Column Stores/ Columnar Databases

- A column-oriented NoSQL database groups data into columns instead of using the rows of a traditional database.
- The columns can consist of different sizes and stored data types.

Features

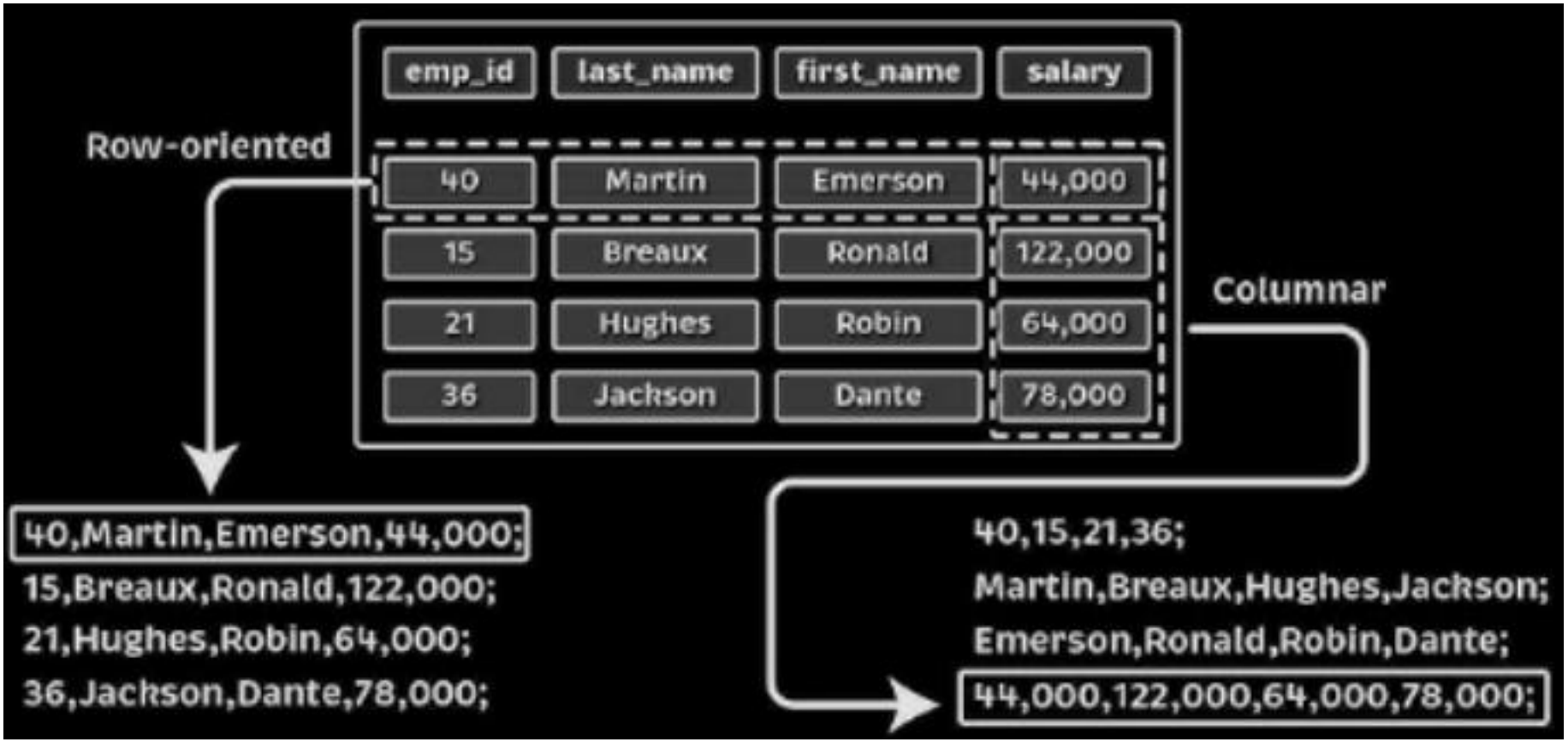
- Data stored in cells are logically grouped as data columns, and columns in turn to column families.
- Column families may contain virtually an unlimited number of columns.
- Column family is similar to a container of rows in an RDBMS table, the key identifies the row and the row consists of multiple columns.

Use Cases

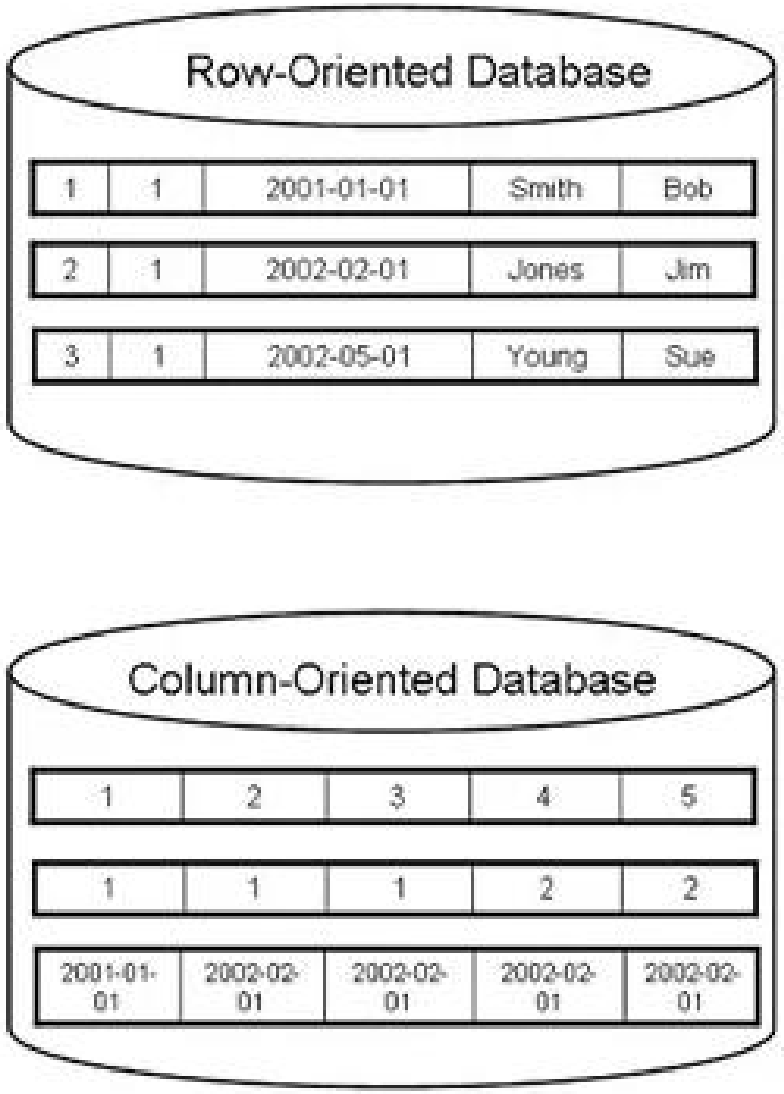
- Content management systems
- Blogging platforms
- Systems that maintain counters
- Services that have expiring usage
- Systems that require heavy write requests (like log aggregators)

Examples

- Cassandra, Apache Hadoop Hbase


emp_id
last_name
first_name
salary
40
Martin
Emerson
44,000
15
Breaux
Ronald
122,000
21
Hughes
Robin
64,000
36
Jackson
Dante
78,000
Row-oriented
Columnar
40,Martin,Emerson,44,000;
15,Breaux,Ronald,122,000;
21,Hughes,Robin,64,000;
36,Jackson,Dante,78,000;
40,15,21,36;
Martin,Breaux,Hughes,Jackson;
Emerson,Ronald,Robin,Dante;
44,000,122,000,64,000,78,000;

Emp_no	Dept_id	Hire_date	Emp_ln	Emp_fn
1	1	2001-01-01	Smith	Bob
2	1	2002-02-01	Jones	Jim
3	1	2002-05-01	Young	Sue
4	2	2003-02-01	Stemle	Bill
5	2	1999-06-15	Aurora	Jack
6	3	2000-08-15	Jung	Laura



Document/Document-Store/Document-Oriented Databases

A document-based NoSQL database stores objects which contain data in standard encodings such as JSON or XML.

Features

- Used for storing, retrieving, and managing semi-structured data
- Uses documents as the structure for storage and queries.
- Data can be added by adding objects to the database.
- Documents are grouped into "collections," similar to a table in RDBMS.

Use Cases

- E-commerce platforms
- Content management systems
- Analytics platforms
- Blogging platforms

Examples

- MongoDB, Apache CouchDB and Elasticsearch

Graph-based Databases

- A graph-based NoSQL database connects many data objects (nodes) together through relationships (edges).
- This type of database combines the querying of relational databases with NoSQL flexibility and scalability.

Features

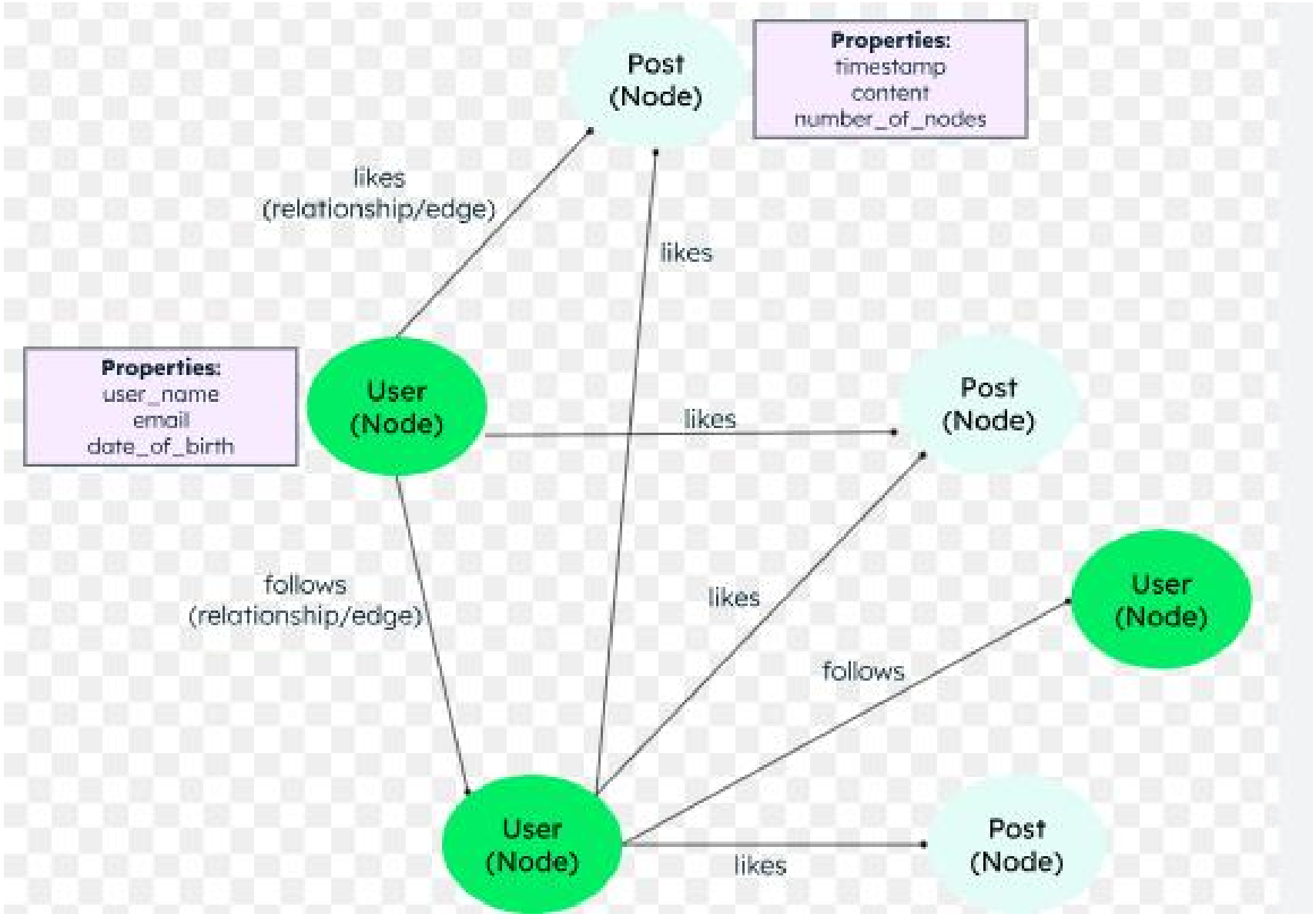
- Built upon Entity - Attribute - Value model.
- Useful in describing relationships between data.
- Entities, also called nodes, store data and each node has properties.
- Relationships describe a relationship between nodes, and a property is the node on the opposite end of the relationship.

Use Cases

- Fraud detection
- Graph based search
- Network and IT operations
- Social networks, etc.

Examples

- Neo4j, ArangoDB and Apache Giraph


Post
(Node)
Properties:
timestamp
content
number_of_nodes
likes
(relationship/edge)
likes
Properties:
user_name
email
date_of_birth
User
(Node)
likes
Post
(Node)
follows
(relationship/edge)
likes
User
(Node)
follows
User
(Node)
likes
Post
(Node)

Introduction to MongoDB

- MongoDB is a non-relational, document-oriented, open source, cross platform, and distributed database imparted with high performance and easy scalability database system.
- It's the scalability feature of this database which gives it the name Mongo taken from the word "humongous" which means large-sized.
- It requires "Collection" (tables in MySQL) of "Documents" (Rows and columns in MySQL) to build the database.
- Data is formatted as JSON or BSON and is stored inside documents.

BSON and JSON

- Binary Javascript Object Notation (BSON) is a non-human-readable data format that MongoDB uses to store data more efficiently than JSON. BSON takes up less space, is faster to parse, and can store more data types than JSON.
- JavaScript Object Notation (JSON) is a human-readable, text-based data format that MongoDB uses for insertion and retrieval of data.
- Data is inserted and retrieved in MongoDB documents as .

```
{
 _id:
ObjectId("9021032303df34948d67wd391"),
 name: { first: "Jin", last: "Sakai" },
 age: 21,
 major: [ "Japanese", "Chemistry"]
}
```

BSON is not JSON

- JSON is text format that requires character by character parsing (like CSV) and provides fewer data types (integers, strings etc.)
- BSON is a serialized binary objects with types, lengths (variable-length strings).
- BSON is faster to read and traverse.
- BSON is similar to how we used to save objects to disk before putting them in database

JSON/BSON Relationship in MongoDB:

- MongoDB allows developers to insert or retrieve data in readable JSON format. Internally, MongoDB stores data as BSON, a format that is not readable by humans, but is a more efficient way for MongoDB to store and parse data.
- **BSON Advantages Over JSON:**
- Storage efficiency and supporting data formats that JSON does not - like dates. BSON is also faster to parse than JSON.
- **JSON Advantages Over BSON:** it is highly structured and parsable by humans.

- **MongoDB Document:** A MongoDB document is an individual record of data stored as “field-value” pairs.
- A field uniquely identifies a data point while a value is the data point itself.
- Documents containing related content are grouped into collections. The document model used by MongoDB is in contrast to the relational database model where d
ro s.

```
{
   <field1>: <value1>,
   <field2>: <value2>,
   ...
}
```

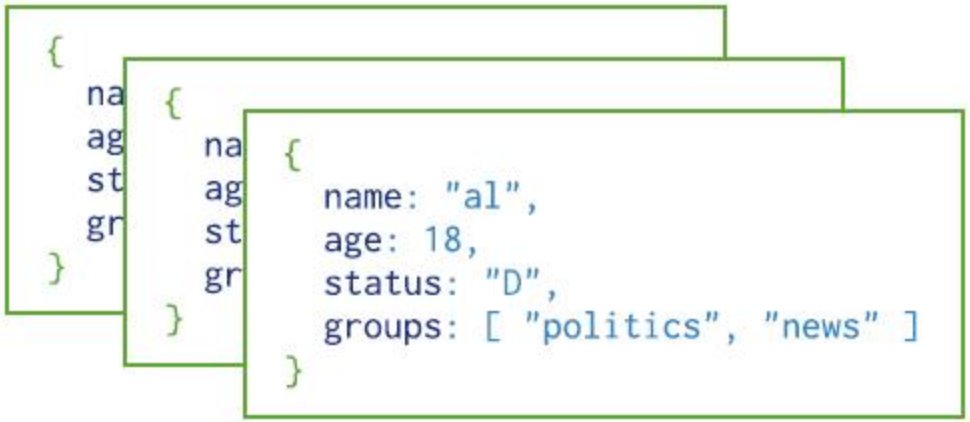
```
{
 _id: ObjectId("98232303df34948b4
 name: { first: "Ezio", last: "Au
 age: 33,
 major: [ "Italian", "Physics"]
}
```

MongoDB - Document Modification Effects

- Within a document-oriented database, like MongoDB, modifications made to a single document will only impact that document. This flexibility is an advantage over relational databases where changing a column of a table will impact every record in the table.

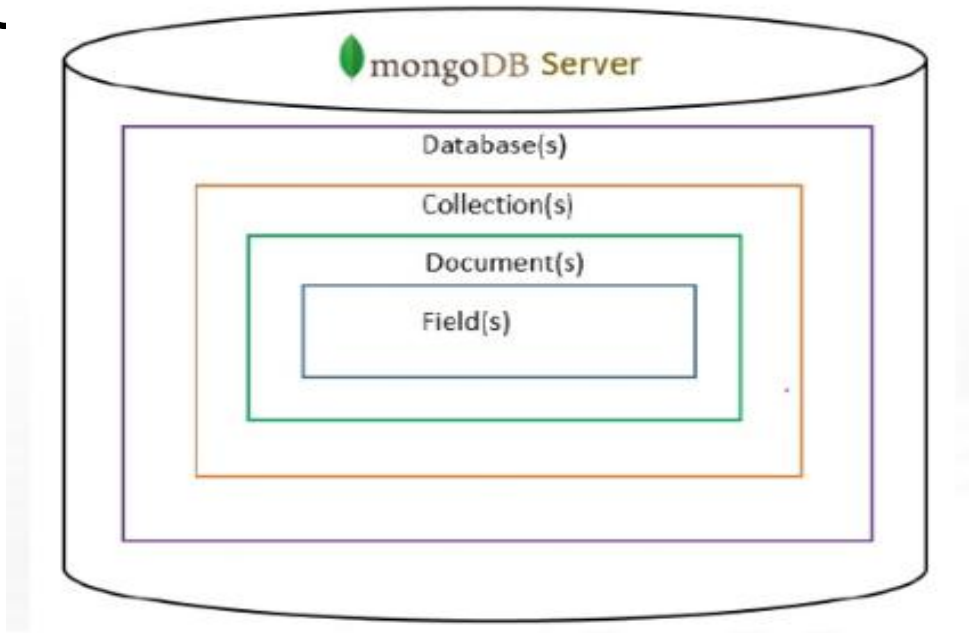
MongoDB Collection

- A MongoDB collection is a group of documents containing similar information. Documents within a collection can have different fields, though they tend to share a similar structure.



MongoDB Data Hierarchy

- A MongoDB database is a number of collections grouped together for a specific use case. A database is made of many collections and collections are made of many dc



MongoDB Atlas

- MongoDB Atlas is MongoDB's database-as-a-service or DBaaS platform. Using Atlas, MongoDB databases can be created, managed, and deployed from the cloud. Data analytics and data visualization are also available on this platform.

Features of MongoDB

MongoDB supports BSON

- MongoDB supports BSON object notation as a storage and data interchange format.
- BSON is a Binary variant of JSON accommodating more data types.
- Codes creating collection and documents are always written in JSON format, which is converted into and stored in its binary form by MongoDB behind the scenes.

Automatic Sharding

- Allows data in ***Collection*** to be spread across many systems for horizontal scalability as the volume of data volumes and in turn, the throughput requirements increase.

Using NoSQL Database Design

- NoSQL stands for Not Only SQL, useful for maintaining large-scale databases.
- Makes use of a single master architecture for consistency in data storage having secondary databases maintaining copies of primary databases.

Comparison of MySQL and NoSQL

Comparison	MySQL	NoSQL
Nature	Database is "Relational" in Nature	Database is "non-relational" in nature
Design	Comprises Tables and Columns	Comprises Collections & Documents
Scalable	Relatively difficult to scale-up for handling large data	Easy scale-up to manage large data
Model	Requires creating a detailed database model	Not required to generate a detailed database model
Standardization	SQL is standard language	Lack of a standard qury language
Schema	Schema is rigid	Schema is dynamic

CAP Theorem

According to Eric Brewer a distributed system has 3 properties:

- **C** - Consistency
- **A** - Availability
- **P** - Partitions

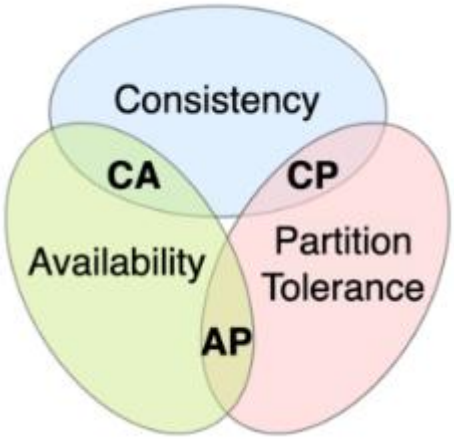
We can have at most two of these properties for any shared-data system

To scale out, we have to partition. It leaves a choice between consistency and availability (mostly in all cases, we would prefer availability over consistency)

Everyone who builds big applications builds them on CAP: **Google, Yahoo, Facebook, Amazon** etc

Consistency: All the nodes see the same data at the same time.
Availability: Every request receives a response.
Partition Tolerance: The system continues to operate even if the part of the system fails/message being dropped or delayed.

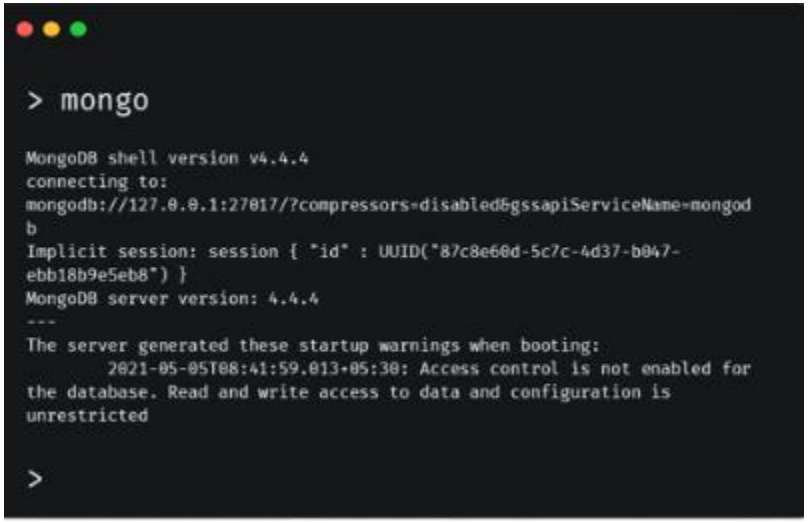
MongoDB as Per CAP Theorem:
• MongoDB is strongly consistent by default .
• MongoDB also gets high availability.



Scenario	Main Focus	Description
No partition	CA	The system is available and provide string consistency
Partition, majority connected	AP	Not synchronized writes from the old primary are ignored
Partition, majority not connected	CP	Only read access is provided to avoid separated and inconsistent system

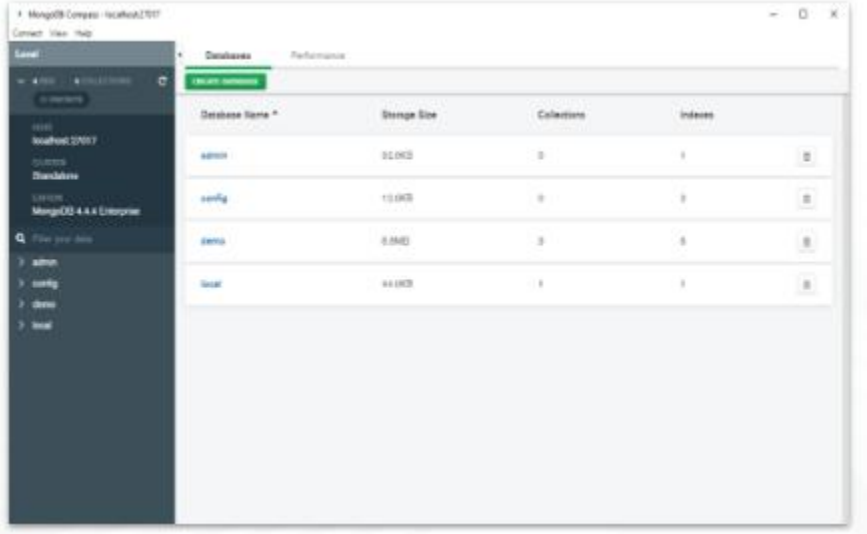
Mongo shell

- The mongo shell is an interactive JavaScript interface to MongoDB.
- It is used to query and update data as well as perform administrative operations.
- Performs CRUD operations.



MongoDB Compass

- MongoDB Compass is a powerful GUI for querying, aggregating, and analyzing MongoDB data in a visual environment.
- Compass is free to use and source available, and can be run on macOS, Windows, and Linux.



Commands

Create database

```
> use demo

switched to db demo
```

Display the list of database

```
> show dbs

admin   0.000GB
config  0.000GB
local   0.000GB
```

Drop database

```
> db
demo

> use demo
switched to db demo

> db.dropDatabase()

{ "dropped" : "demo", "ok" : 1 }
```

To drop database first make sure right database must be selected.

Create Collection

```
db.createCollection("employee")
db.<collection_name>.insert("name": "XYZ")
```

Capped Collection

It is fixed size collection

```
db.createCollection("users", {capped: true, size: 6142800,
                               max: 10000})
```

bytes

Drop Collection

```
db.collection_name.drop()
```

Capped collection
It is fixed size collection

bytes

db.createCollection("users", {capped: true, size: 6142800, max: 10000})

Parameter	Type	Description
capped	Boolean	If true, enables a capped collection. Capped collection is a fixed size collection that automatically overwrites its oldest entries when it reaches its maximum size. If you specify true, you need to specify size parameter also.
autoIndexId	Boolean	If true, automatically create index on _id field. Default value is false.
size	number	Specifies a maximum size in bytes for a capped collection. If capped is true, then you need to specify this field also.
max	number	Specifies the maximum number of documents allowed in the capped collection.