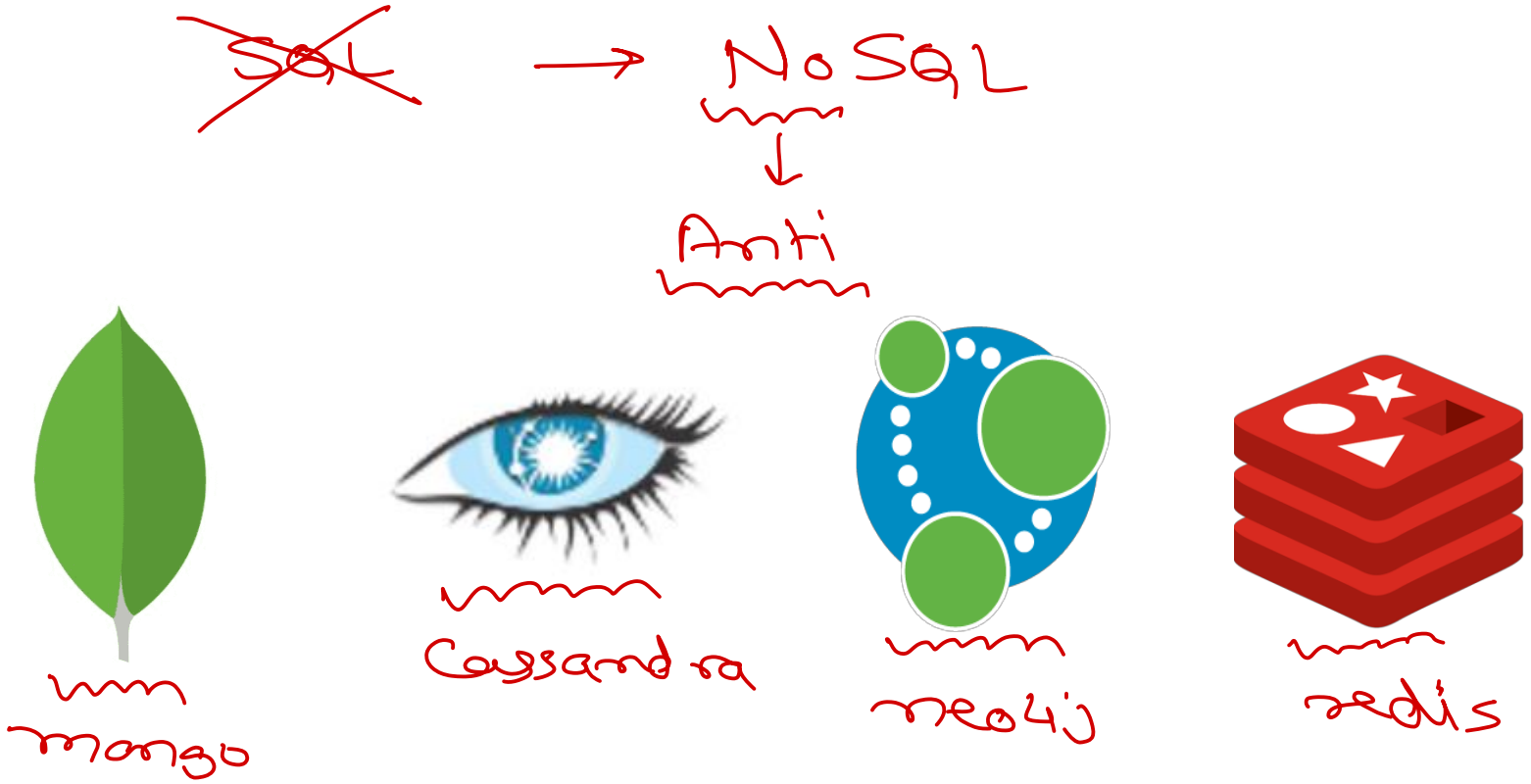
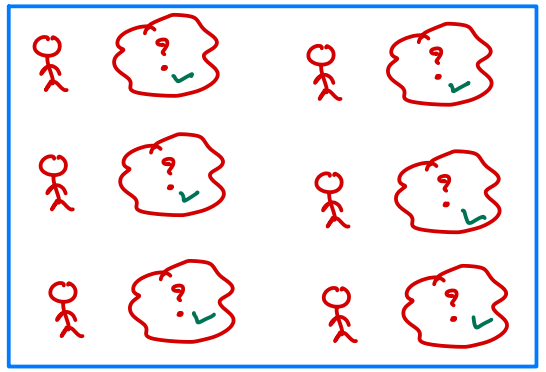


NoSQL Databases

Trainer: Mr. Nilesh Ghule

NoSQL

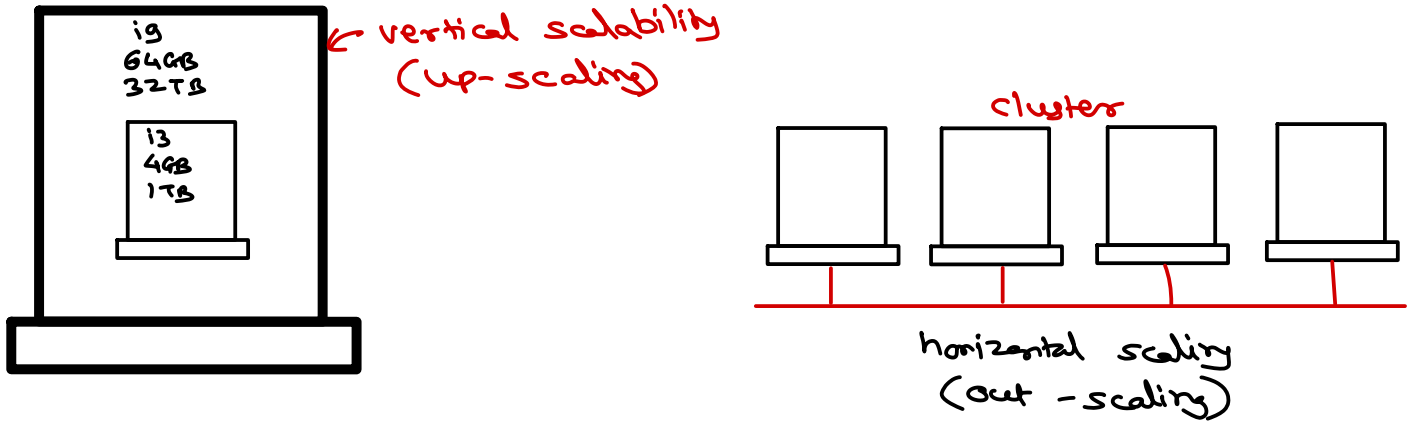
- Refer to non-relational databases
- Stands for Not Only SQL
- Term NoSQL was first used by Carlo Strozzi in 1998.
- No declarative query language
- No predefined schema, Unstructured and unpredictable data
- Eventual consistency rather ACID property
- Based on CAP Theorem
- Prioritizes high performance, high availability and scalability
- BASE Transaction
  - Basically Available
  - Soft state
  - Eventual consistency



last.fm → international Conference → 2004 → twitter invitation #nosql → nosql

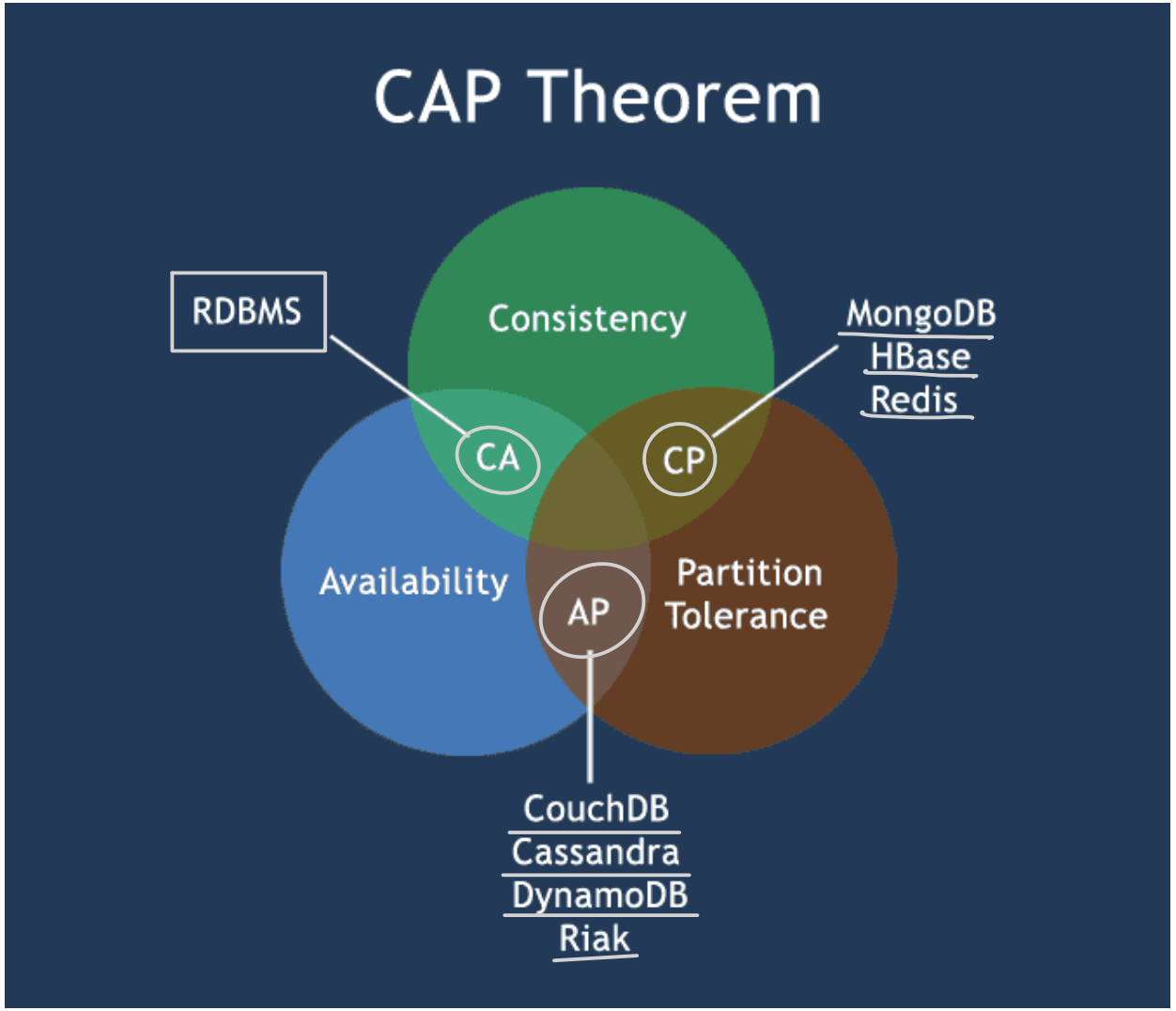
Scaling

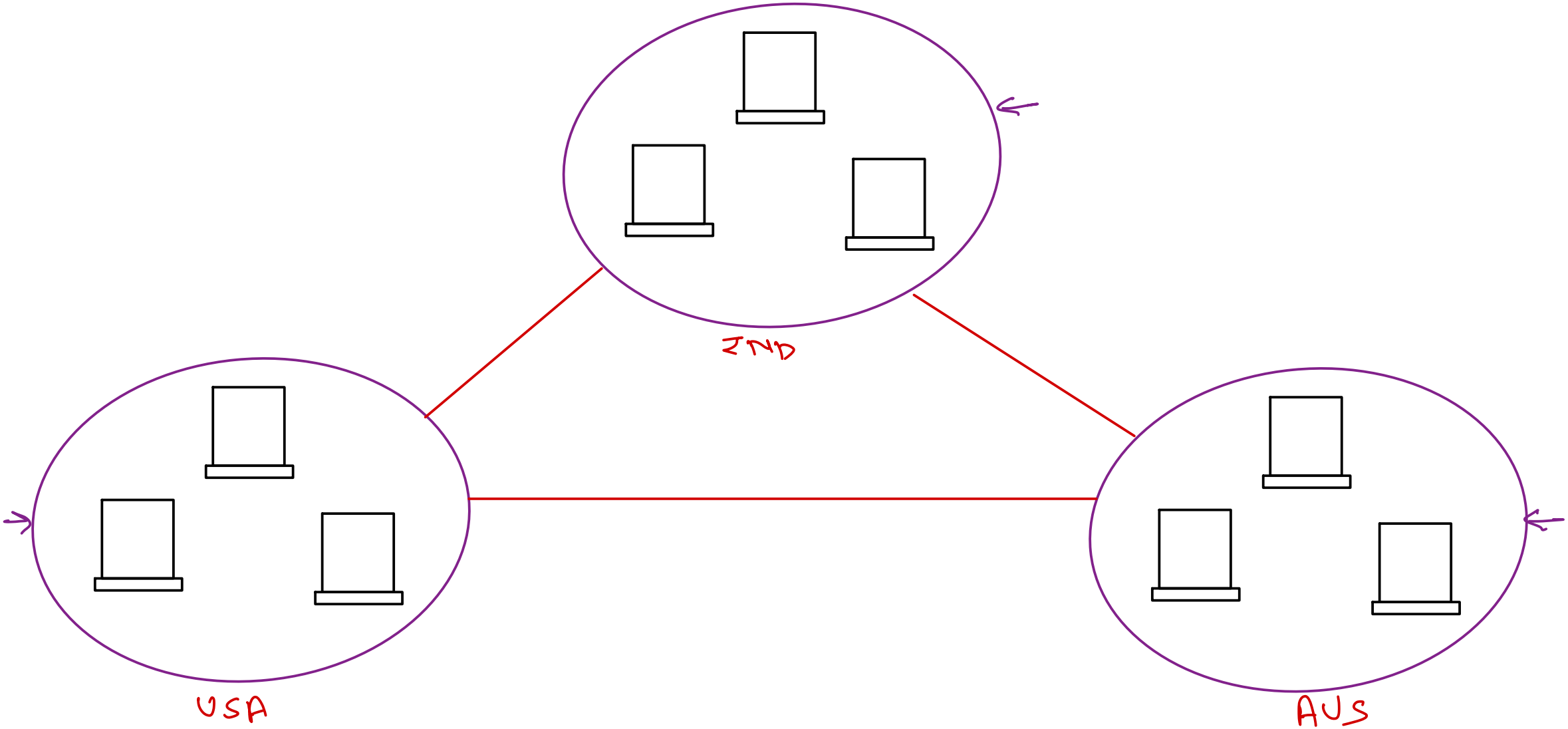
- Scalability is the ability of a system to expand to meet your business needs.
- E.g. scaling a web app is to allow more people to use your application.
- Types of scaling
  - Vertical scaling: Add resources within the same logical unit to increase capacity. E.g. add CPUs to an existing server, increase memory in the system or expanding storage by adding hard drives.
  - Horizontal scaling: Add more nodes to a system. E.g. adding a new computer to a distributed software application. Based on principle of distributed computing.
- NoSQL databases are designed for Horizontal scaling. So they are reliable, fault tolerant, better performance (at lower cost), speed.



CAP (Brewer's) Theorem

- **Consistency** - Data is consistent after operation. After an update operation, all clients see the same data.
- **Availability** - System is always on (i.e. service guarantee), no downtime. 24x7
- **Partition Tolerance** - System continues to function even the communication among the servers is unreliable.
- Brewer's Theorem
  - It is impossible for a distributed data store to simultaneously provide more than two out of the above three guarantees.




IND
USA
AUS

Advantages of NoSQL

- High scalability
  - This scaling up approach fails when the transaction rates and fast response requirements increase. In contrast to this, the new generation of NoSQL databases is designed to scale out (i.e. to expand horizontally using low-end commodity servers).
- Manageability and administration
  - NoSQL databases are designed to mostly work with automated repairs, distributed data, and simpler data models, leading to low manageability and administration.
- Low cost
  - NoSQL databases are typically designed to work with a cluster of cheap commodity servers, enabling the users to store and process more data at a low cost.
- Flexible data models
  - NoSQL databases have a very flexible data model, enabling them to work with any type of data; they don't comply with the rigid RDBMS data models. As a result, any application changes that involve updating the database schema can be easily implemented.

Disadvantages of NoSQL

- Maturity
  - Most NoSQL databases are pre-production versions with key features that are still to be implemented. Thus, when deciding on a NoSQL database, you should analyse the product properly to ensure the features are fully implemented and not still on the To-do list.
- Support
  - Support is one limitation that you need to consider. Most NoSQL databases are from start-ups which were open sourced. As a result, support is very minimal as compared to the enterprise software companies and may not have global reach or support resources.
- Limited Query Capabilities
  - Since NoSQL databases are generally developed to meet the scaling requirement of the web-scale applications, they provide limited querying capabilities. A simple querying requirement may involve significant programming expertise.
- Administration
  - Although NoSQL is designed to provide a no-admin solution, it still requires skill and effort for installing and maintaining the solution.
- Expertise
  - Since NoSQL is an evolving area, expertise on the technology is limited in the developer and administrator community.

Applications

- When to use NoSQL?
  - Large amount of data (TBs)
  - Many Read/Write ops
  - Economical Scaling
  - Flexible schema
- Examples:
  - Social media
  - Recordings
  - Geospatial analysis
  - Information processing

- When Not to use NoSQL?
  - Need ACID transactions
  - Fixed multiple relations
  - Need joins
  - Need high consistency
- Examples
  - Financial transactions
  - Business operations

RDBMS vs NoSQL

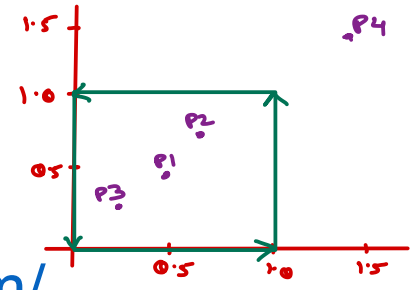
	RDBMS	NoSQL
Types	All types support SQL standard	Multiple types exists, such as document stores, key value stores, column databases, etc
History	Developed in 1970	Developed in 2000s
Examples	SQL Server, Oracle, MySQL	MongoDB, HBase, Cassandra, Redis, Neo4J
Data Storage Model	Data is stored in rows and columns in a table, where each column is of a specific type	The data model depends on the database type. It could be Key-value pairs, documents etc
Schemas	Fixed structure and schema	Dynamic schema. Structures can be accommodated
Scalability	Scale up approach is used	Scale out approach is used
Transactions	Supports ACID and transactions	Supports partitioning and availability
Consistency	Strong consistency	Dependent on the product [Eventual Consistency]
Support	High level of enterprise support	Open source model
Maturity	Have been around for a long time	Some of them are mature; others are evolving

MongoDb Databases

Trainer: Mr. Nilesh Ghule

Mongo - GeoSpatial Queries

- mongodb support three types of geo-spatial queries:
  - 2-d index: traditional long-lat. used in older mongodb (2.2-).
  - 2-d sphere index: data can be stored as GeoJSON.
  - geo-haystack: query on very small area. not much used.
- GeoJSON stores geometry and coordinates: http://geojsonlint.com/
  - { type: "geometry", coordinates: [long, lat] };
  - { type: "Point", coordinates: [73.86704859999998, 18.4898445] };
- Possible geometry types are:
  - Point, LineString, MultiLineString, Polygon
- allowed queries:
  - inclusion - $geoWithin
  - intersection - $geoIntersects
  - proximity - $near



Mongo - GeoSpatial Queries

- For faster execution create geo-index :
  - db.busstops.createIndex({ location : "2dsphere" });
- Example proximity query:

```
db.busstops.find( { location : { $near : {
        $geometry : { type : "Point" ,
                coordinates : [73.86704859999998, 18.4898445]
                },
                $maxDistance : 200
    } } } );
```

Mongo - Capped Collections

- Capped collections are fixed sized collections for high-throughput insert and retrieve operations.
- They maintain the order of insertion without any indexing overhead.
- The oldest documents are auto-removed to make a room for new records. The size of collection should be specified while creation.
- The update operations should be done with index for better performance. If update operation change size, then operation fails.
- Cannot delete records from capped collections. Can drop collection.
- Capped collections cannot be sharded.
- db.createCollection("logs", { capped: true, size: 4096 }); → if size is below 4096, 4096 is considered. Higher sizes are roundup by 256.

Thank you!

Nilesh Ghule <nilesh@sunbeaminfo.com>