**Group A : Assignment 1**

Name : Anushka M Mulay

Roll No : 35040

Group : A

Batch : B

**Title : -** Introduction to databases (Study Assignment )

**Problem Statement** : - Study and design a database with suitable example using following database systems:

* Relational: SQL/ MySQL
* Key-value: Riak/ Redis
* Columnar: HBase
* Document: MongoDB/ CouchDB
* Graph: Neo4j

Compare the different database systems based on points like efficiency, scalability, characteristics and performance.

**Theory** :-

1. **MySQL :** MySQL is a fast, easy-to-use RDBMS being used for many small and big businesses. MySQL is developed, marketed and supported by MySQL AB, which is a Swedish company. MySQL is a very powerful program in its own right. It handles a large subset of the functionality of the most expensive and powerful database packages. MySQL uses a standard form of the well-known SQL data language. MySQL works on many operating systems and with many languages including PHP, PERL, C, C​++, JAVA, etc.

Languages in MySQL:

DDL Statements:-Allows the user to define data and their relationship to other types of data

create​:-to create the database instance. **alter**​:-to alter the structure of database. drop​:-to drop database instances.

rename​:-to rename database instances

DML (Data Manipulation Language):-Use for accessing and manipulating data in database.

DML Statement:-

select:-to access or read record from table.

insert:-to insert record into table.

update:-update the record in table.

delete:-delete the record from table.

DCL (DATA CONTROL LANGUAGE):- Used to control the user access to database related elements like tables, views, functions, procedures etc.

DCL Statements:-

i.grant:-used to provide the privileges to the user on the database objects. The privileges could be select, delete, update, insert on the tables and views.

ii.revoke:-helps to remove the privileges given on database objects.

**Efficiency** : -

●High speed- MySQL Queries can be used to retrieve large amounts of records from a database quickly and efficiently.

●Well defined standard exists- MySQL databases use long-established standard, which is being adopted by ANSI & ISO. Non-MySQL databases do not adhere to any clear standard

● No coding required- Using standard MySQL, it is easier to manage database systems without having to write substantial amount of code.

**Scalability** : - The response time of an MYSQL query depends on many factors. The data volume is one of them. If a query is fast enough under certain testing conditions, it does not mean it will be fast enough in production. That is especially the case in development environments that have only a fraction of the data of the production system.

MYSQL is used to share and manage data, particularly data that is found in relational database management systems - data is organized into tables, and multiple files, each containing tables of data, may be related together by a common field.

## Characteristics : -

● MySQL is an ANSI and ISO standard computer language for creating and manipulating databases.

● MySQL allows the user to create, update, delete, and retrieve data from a database.

● MySQL is very simple and easy to learn.

● MySQL works with database programs like DB2, Oracle, MS Access, Sybase, MS MySQL Server etc.

● Allows users to define the data in a database and manipulate that data.

● Allows to embed within other languages using MYSQL modules, libraries & pre-compilers.

**Performance** : -

The truth is, common mistakes underlie most MYSQL performance problems. To ensure your MYSQL server hums along at top speed, providing stable and consistent performance, it is important to eliminate these mistakes, which are often obscured by some subtlety in your workload or a configuration trap.

To function, a database server needs four fundamental resources: CPU, memory, disk, and network. If any of these is weak, erratic, or overloaded, then the database server is very likely to perform poorly.

**Conclusion** : Thus, we have learnt about relational database MySql.

# RIAK

**Riak** (pronounced "ree-ack") is a distributed NoSQL key-value data store that offers high​ availability, fault tolerance, operational simplicity, and scalability. In addition to the open-source version, it comes in a supported enterprise version and a cloud storage version. Riak implements the principles from Amazon's Dynamo paper with heavy influence from the CAP Theorem. Written in Erlang, Riak has fault tolerance data replication and automatic data distribution across the cluster for performance and resilience.

## Efficiency : - The efficiencies of Riak and improve the cost basis for the Cisco Inter cloud service offering and increasing customer satisfaction by creating an easy to consume data service for application developers. In addition, the entire community benefits as this work is fed into the open source project.

## Scalability : -​ The masterless architecture of Riak makes it easy to add and remove nodes from your cluster. You can achieve predictable and near-linear scale by adding nodes using commodity hardware.

## When nodes are added or removed, data is rebalanced automatically without requiring human intervention. New machines assume ownership of some of the partitions, and existing machines hand off relevant partitions and associated data until data ownership is equal across nodes.

## Replicating data eliminates manual sharding; Riak makes it significantly easier for applications to scale.

## Riak TS is designed to handle massive amounts of Io or time series data. Clusters can scale to multi-petabytes of data.

**Characteristics :**

●Fault-tolerant availability : Riak replicates key/value stores across a cluster of nodes with a default n\_val of three. In the case of node outages due to network partition or hardware failures, data can still be written to a neighbouring node beyond the initial three, and read-back due to its &quot;masterless&quot; peer-topeer architecture.

●Queries : Riak provides a REST-full API through HTTP and Protocol Buffers for basic PUT, GET, POST, and DELETE functions. More complex queries are also possible, including secondary indexes, search (via Apache Solr), and Map Reduce. Map Reduce has native support for both JavaScript (using the Spider Monkey runtime) and Erlang.

●Predictable latency : Riak distributes data across nodes with hashing and can provide latency profile, even in the case of multiple node failures.

●Storage options: Keys/values can be stored in memory, disk, or both.

## Performance : -

Firstly, performance will probably depend a lot on the number of cores available, and the memory. These systems are usually heavily pipelined and concurrent and benefit from a lot of cores. 4+ cores and 4GB+ of RAM would be a good starting point.

Secondly, Map Reduce is designed for batch processing, not real-time queries.

Riak and all similar Key-Value stores are designed for high write performance, high read performance for simple lookups, no complex querying at all.

**Conclusion** : Thus, we have learnt about key-value database Riak.

## HBASE : HBase is a distributed column-oriented database built on top of the Hadoop file system. It is an open-source project and is horizontally scalable. To scale horizontally (or scale out/in) means to add more nodes to (or remove nodes from) a system, such as adding a new computer to a distributed software application.

## HBase is a data model that is similar to Google’s big table designed to provide quick random access to huge amounts of structured data. It leverages the fault tolerance provided by the Hadoop File System (HDFS).

## It is a part of the Hadoop ecosystem that provides random real-time read/write access to data in the Hadoop File System.

### Efficiency : -

What HBase is that it is basically a column-oriented key-value data store, and the since it works extremely fine with the kind of data that Hadoop process it is natural fit for deploying as a top layer on HDFS. It is extremely fast when it comes to both read and write operations and does not lose this extremely important quality even when the datasets are humongous. Therefore it is being widely used by corporations for its high throughput and low input/output latency. It cannot work as a replacement for the SQL database but it is perfectly possible to have an SQL layer on top of HBase to integrate it with the various business intelligence and analytics tools.

### Scalability : -

Scaling a database requires horizontally scaling the database contents across multiple nodes. In some databases, it is only possible to replicate complete copies of a database onto multiple nodes, which restricts scalability to the capacity of a single node - this is effectively scaling up. Other databases support horizontal partitions, or sharding, to scale data on to multiple nodes. In addition, some databases make it possible to also replicate partitions to scale both read and write loads.

### Characteristics : -

* HBase is linearly scalable.
* It has automatic failure support.
* It provides consistent read and writes.
* It integrates with Hadoop, both as a source and a destination.
* It has easy java API for client.
* It provides data replication across clusters.

### Performance : -

HBase comes with very good scalability and performance for this workload with a simpler consistency model than Cassandra. While they found HBase to be the most suitable in terms of their requirements like auto load balancing and failover, compression support, multiple shards per server, etc.

**Conclusion** : Thus, we have learnt about columnar database HBase.

## MONGODB : MongoDB is a document database with the scalability and flexibility that you want with the querying and indexing that you need MongoDB stores data in flexible, JSON-like documents, meaning fields can vary from document to document and data structure can be changed over time

## The document model maps to the objects in your application code, making data easy to work with Ad hoc queries, indexing, and real time aggregation provide powerful ways to access and analyse your data.

## MongoDB is a distributed database at its core, so high availability, horizontal scaling, and geographic distribution are built in and easy to use.

## This data model gives you the ability to represent hierarchical relationships, to store arrays, and other more complex structures easily. MongoDB was also designed with high availability and scalability in mind, and includes out-of-the-box replication and auto-sharding

### Efficiency : -

One of the most powerful features of MongoDB is its rich indexing functionality. Users can specify secondary indexes on any field, compound indexes, geospatial, text, sparse, TTL, and others. Having extensive indexing functionality makes it easier for developers to build apps that provide rich functionality and low latency.

### Scalability : -

Scalability is not just about speed. It&#39;s about 3 different metrics, which often work together:

* Cluster Scale: Distributing the database across 100+ nodes, often in multiple data centres
* Performance Scale: Sustaining 100,000+ database read and writes per second while maintaining strict latency SLAs
* Data Scale: Storing 1 billion+ documents in the database

### Characteristics : -

* Support ad hoc queries: In MongoDB, you can search by field, range query and it also​ supports regular expression searches.​
* Indexing: You can index any field in a document
* Replication: MongoDB supports Master Slave replication. A master can perform Reads and Writes and a Slave copies data from the master and can only be used for reads or back up (not writes)
* Duplication of data: MongoDB can run over multiple servers. The data is duplicated to keep the system up and also keep its running condition in case of hardware failure.
* Load balancing: It has an automatic load balancing configuration because of data placed in shards.

### Performance : -

As you develop and operate applications with MongoDB, you may need to analyze the performance of the application and its database. When you encounter degraded performance, it is often a function of database access strategies, hardware availability, and the number of open database connections.

* Locking Performance
* Memory and the MMAPv1 Storage Engine
* Number of Connection
* Database Profiling
* Additional Resources

**Conclusion** : Thus, we have learnt about document database MongoDB.

## NEO4J : Neo4j is one of the popular Graph Databases and Cypher Query Language (CQL). Neo4j is written in Java Language.

## In Neo4j too, relationships should be directional. If we try to create relationships without direction, then Neo4j will throw an error message saying that &quot;Relationships should be directional&quot;.

## Neo4j Graph Database stores all of its data in Nodes and Relationships. We neither need any additional RRBMS Database nor any SQL database to store Neo4j database data. It stores its data in terms of Graphs in its native format.

## Neo4j uses Native GPE (Graph Processing Engine) to work with its Native graph storage format.

**Efficiency** : - Neo4j effectively connects people with our products and services, based on personal information their profiles on social networks and their recent online activity. In this respect, graph databases​​are interesting because they are able to connect people and interests.

### Scalability : -

• Scale is: redundancy, increased write load, increased read load, increased data size.

* Neo4j’s scalability package is known as High Availability (or HA).
* In every Neo4j HA cluster, the full graph is replicated to each instance of the cluster.
* The read capacity of each HA cluster increases linearly with the number of servers.
* Neo4j can commit tens of thousands of writes per second while maintaining fully ACID transactions.
* At scale, Neo4j best practices include online backups, reporting instances, data locality and disaster recovery.

### Characteristics : -

* ACID for Data Integrity
* Flexible Schema
* High Performance Query Execution
* Cypher Query Language
* Scale and Performance
* Advanced Casual Clustering
* Built-in Tooling and Visualization
* Drivers for popular languages and frameworks
* Seamless Data Import
* Cloud Ready Deployment
* Elastic Scalability

**Performance** : - Neo4j uses the Raft protocol. This is a consensus algorithm used for distributing a state machine across a cluster of computing systems. In Neo4j this is what enables fault tolerance and related capabilities such as causal consistency (which is to be preferred - in my opinion - to eventual consistency). That&#39;s your central cluster but then you can define what are known as Read Replicas that can depend from the main cluster and you can have multiple such replicas that can be defined for different purposes. From a performance perspective, the company has re-written both its browser and its label indexing in the latest release (3.2), in the latter case so that they are graph native, as opposed to being based on Lucene (which is greedy on resources).

**Conclusion** : Thus, we have learnt about graph database Neo4j.