

REDIS OR NEO 4J ? A COMPREHENSIVE STUDY OF NO SQL DATABASES: KEY-VALUE VS GRAPH DATABASE PERFORMANCE

Dr. Deepa Nyayadhish, Asst. Professor V. K. K. Menon College
Mrs. Hiral Joshi, Asst. Professor V. K. K. Menon College

ABSTRACT:

Nowadays we have a variety of databases designed to meet our specific data requirements. Although traditional relational databases are commonly used, the flexibility and scalability of NoSQL databases have made them increasingly popular. These databases come in various types, including document databases, key-value stores, and column-family stores. Among the NoSQL databases, the popularity of graph databases is on the rise. This paper will delve into the differences between graph and Key-value databases, examining their workings.

This study aims to evaluate two popular databases and compare the performance difference between Redis and Neo4j.[1]

Key Words: Redis, Neo4j, NoSQL, Graph, Eclipse IDE, CQL, Database.

INTRODUCTION:

The major challenge with the growing data is its nonuniformity. Due to this problem, in recent years, a nonrelational database is needed to scale the growing need of industry and at the same time, must be highly efficient. This gave rise to NoSQL databases which are highly scalable, efficient and can store large amounts of data.[2]

A.IMPORTANCE OF NOSQL

NoSQL databases rigid schemes and many other limitations are avoided. They were initially introduced as databases to provide an alternative to the long existing relational databases. For these NoSQL databases scalability, fault tolerance and availability are the most important deciding factors. They do not follow the strict schema approach of RDBMSs .

There are four general types of NoSQL databases where every database has its own properties:

- **Graph database:** The basis of this type of database is graph theory. Examples: Neo4j and Titan.
- **Key-Value store:** In this database, we store the data in two parts, namely key and value. Examples: Redis, DynamoDB, Riak.
- **Column store:** Here, data is stored in the form of sections of columns of data. Examples: HBase, BigTable and Cassandra.
- **Document database:** This database is a higher version of key-value stores. Here values are saved as documents which are data in the form of complex structures (like JSON). Examples: MongoDB and CouchDB.

REDIS: WHAT IT IS, WHAT IT DOES:

A.What is Redis?

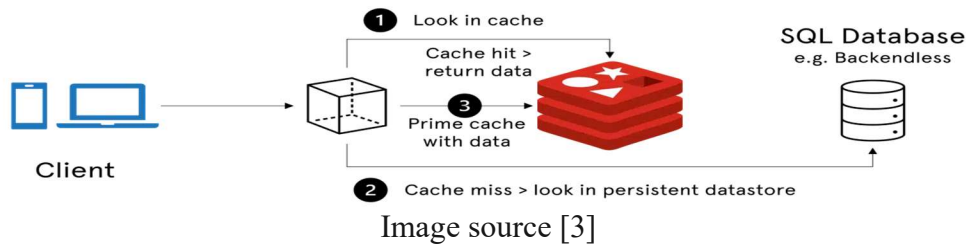
Redis is an open source in-memory data store that can be used as a database, cache or message broker, but it can also be used as a database when you don't need all the features of a traditional database. It offers excellent performance, with the ability to quickly read and write data to memory. Additionally, Redis supports atomic operations, making it ideal for caching scenarios where you need fast access time.[3]

B.In-memory database

An in-memory database is a type of database that stores data entirely in main memory (RAM) rather than on disk. In-memory databases are designed to provide fast access to data by leveraging the high speed of main memory, which is several orders of magnitude faster than disk storage.

In-memory databases are commonly used in applications that require fast access to large amounts of data, such as real-time analytics, online gaming, e-commerce, and social media. They are also used in applications that require high performance and scalability, as in-memory databases can handle high volumes of data and transactions without sacrificing performance.[3]

How Redis is typically used



C.What are key-value pairs?

In Redis, a key-value pair is a data structure that consists of a unique key, which is used to identify the data, and a value, which is the data itself. Key-value pairs are the most basic data structure in Redis, and they are used to store and manage data in the database.

Redis supports a wide range of data types for keys and values, including strings, hashes, lists, sets, and sorted sets. This allows developers to store and manipulate a variety of data types in Redis, such as text, numbers, arrays, and complex data structures.

Redis provides a rich set of commands for working with key-value pairs, such as SET, GET, and DEL for strings, HSET, HGET, and HDEL for hashes, and LPUSH, LGET, and LREM for lists. These commands enable developers to store, retrieve, and manipulate data in Redis efficiently and easily.[3]

HOW KEY-VALUE PAIR WORKS:

Following image1 demonstrates the working of storing and retrieving data with key-value pairs of SET,GET and MSET,MGET commands .

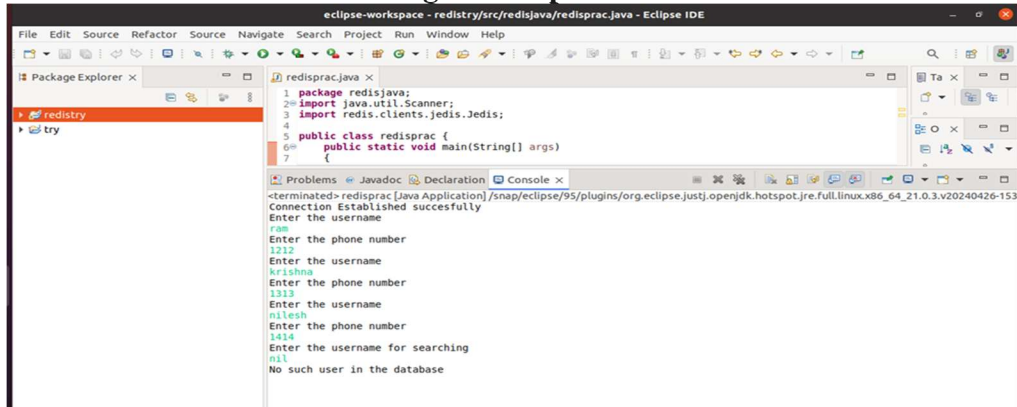
Image1:Redis Commands

```

lab@lab-H410M-S2-V2: ~
lab@lab-H410M-S2-V2:~$ redis.cli
127.0.0.1:6379> ping
PONG
127.0.0.1:6379> set satish 8888888
OK
127.0.0.1:6379> mset jeff 4444444 tom 333333 ram 766666
OK
127.0.0.1:6379> mget satish jeff tom ram
1) "8888888"
2) "4444444"
3) "333333"
4) "766666"
127.0.0.1:6379> get satish
"8888888"
127.0.0.1:6379>
  
```

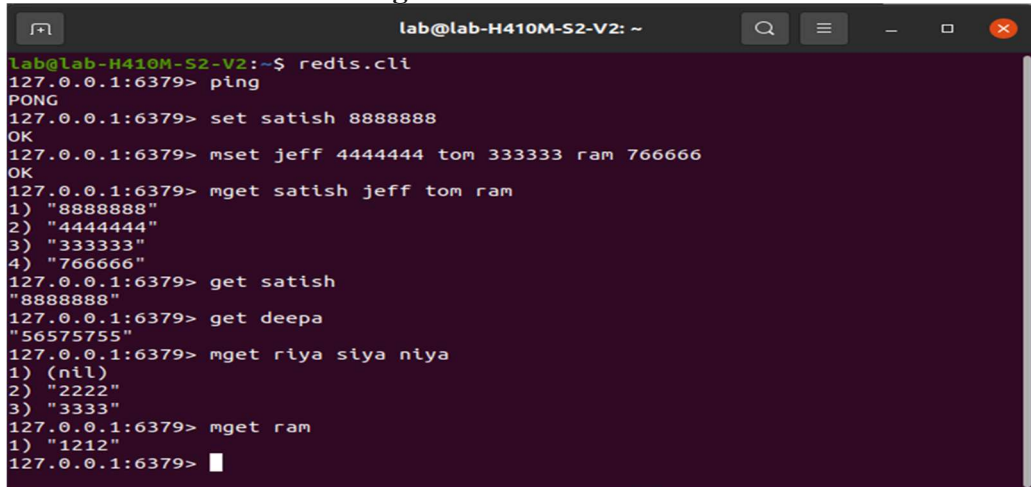
In Image 2 Eclipse IDE is used to implement Redis as a back end with Java. It describes that after establishing a successful connection, the user has entered name and phone number as an input.

Image 2: Eclipse IDE



As it can be clearly seen in Image 3 that values entered in java can be accessed through the Redis command prompt using GET and MGET commands.

Image 3 : Redis commands



NEO4J: WHAT IT IS, WHAT IT DOES :

A.What is Neo4j?

Neo4j is an open-source graph database management system, designed to store and query data structured as a graph, which consists of nodes (entities) and relationships (edges) connecting them. Traditional databases organize data in tables whereas Neo4j represents data as a network of interconnected nodes, making it highly efficient for complex querying and traversals of relationships between entities which are crucial. Additionally, Neo4j supports full ACID operations, indexing and unique constraints to optimize queries. This structure excels at querying complex, interconnected data, such as social networks, recommendation engines, and fraud detection systems. [1]

B.Architecture of Graph Databases:

Graph data is stored in a native graph storage that ensures fast CRUD operation. Data is stored on disk or in Page cache memory. Indexes are used in graph which Optimizes data retrieval by indexing node and edge properties (e.g., name, ID). In-built algorithms like shortest path is used to traverse the graph. External Devices interact with the graph database through REST APIs and provide graphical

visualization. Transaction Log ensures any changes to database are permanently saved and consistent ,even if there are system failure .[7]

C.What is Cypher Query Language?:

In Neo4j a Cypher query language is used to retrieve graph data from a database,a single Traverse with no lookups involved. It is a declarative pattern-matching language that supports SQL syntax ,clauses like WHERE,ORDER-BY and functions such as string, aggregation to simplify complex queries.The nodes are represented using simple pair of round () parentheses and relationship between nodes is represented using a pair of dashes both undirected – and directed using arrow head -->.

CYPHER ALSO USES BASIC COMMANDS LIKE:

Create : It adds nodes or relationship among existing nodes in graph

CREATE (Variable name :label to node{property: key-value pair})

MATCH : It finds nodes and their relationship based on pattern

Match(n) return(n)[6]

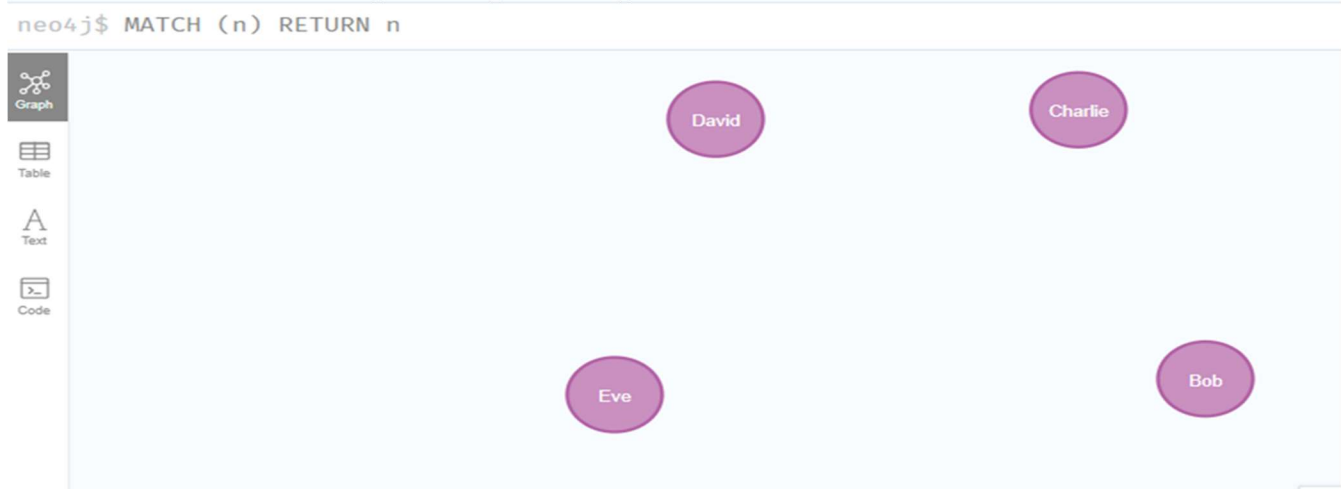
HOW CQL WORKS ON GRAPH DATABASE:

Following image 4 shows the working of storing and retrieving data with CQL commands to create nodes and relationships[8]

Image 4:CQL command

```
1 CREATE (alice:Person {name: 'Alice', age: 30}),
2      (bob:Person {name: 'Bob', age: 25}),
3      (charlie:Person {name: 'Charlie', age: 35}),
4      (david:Person {name: 'David', age: 40}),
5      (eve:Person {name: 'Eve', age: 28});
6
```

Image 5:Graphical Representation of CQL command

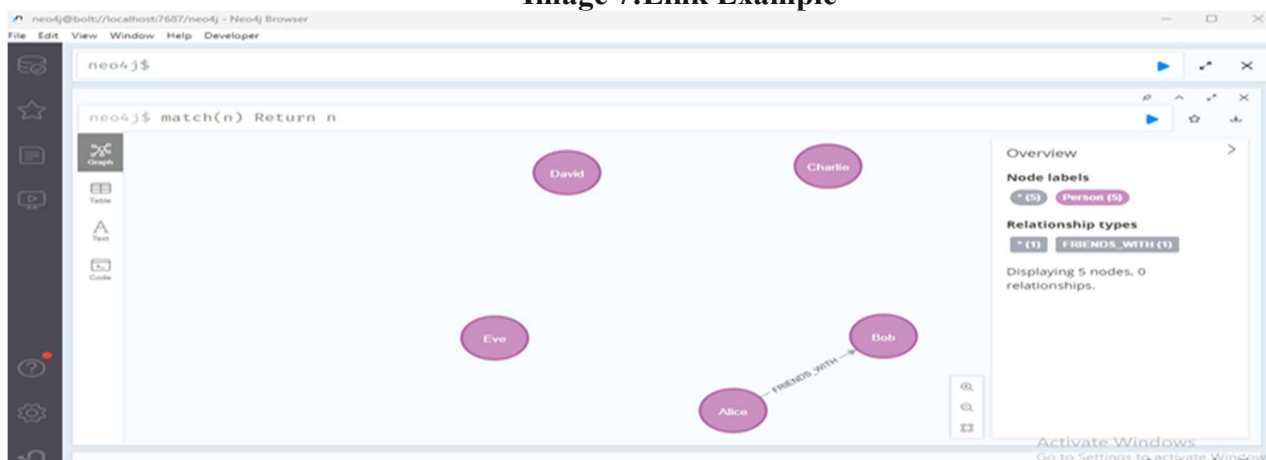
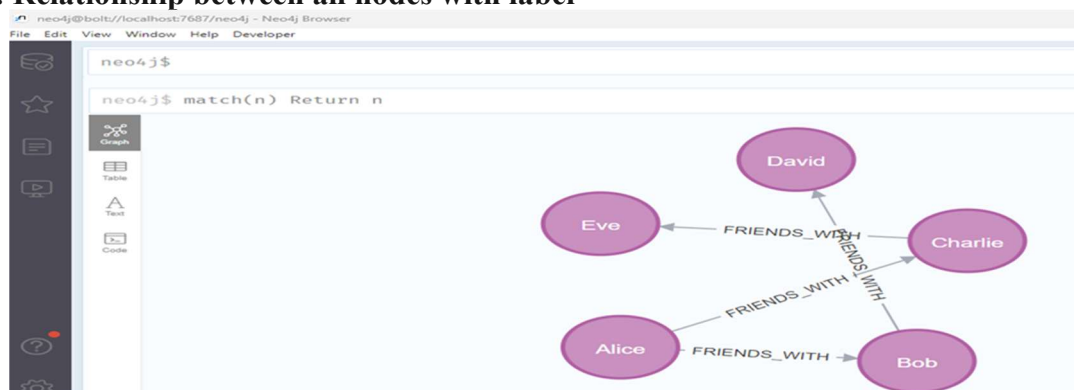


Neo4j is used to implement Real world examples of graph databases . Image5 Provides Graphical view of 5 nodes with label which we have created in image 4.[8]

Image 6: Demonstrates graphical relationship between different nodes like Alice and Bob using match and Return commands

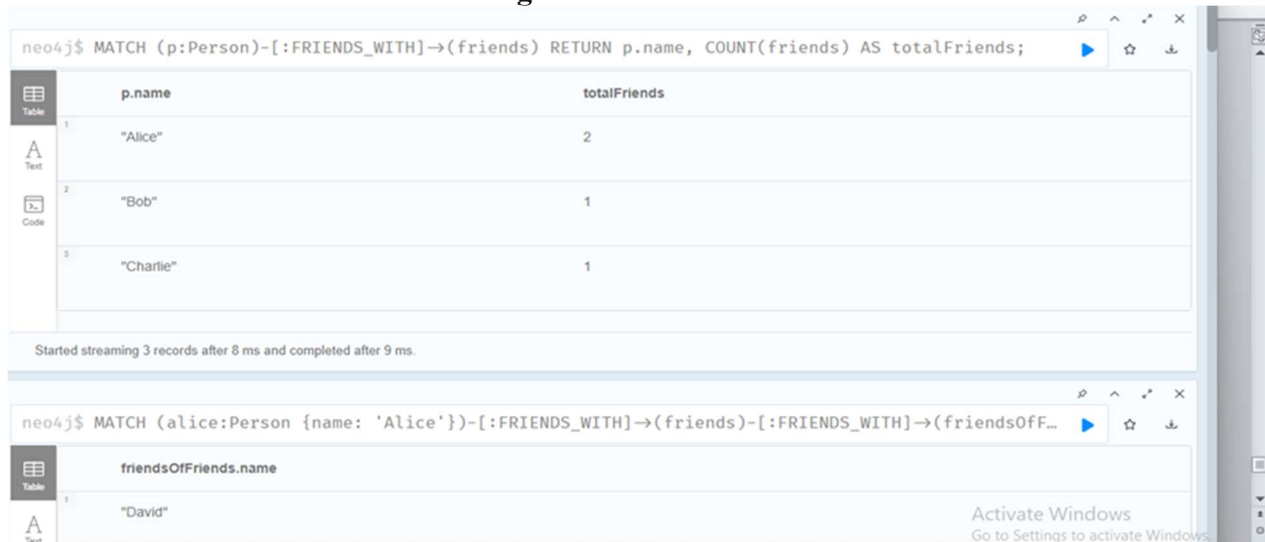
Image 6: Match and Return command

Image 7 shows Graphical link between Alice and Bob with reference to image 6 command.

Image 7: Link Example**Image 8: Relationship between all nodes with label**

As it can be clearly seen in Image 8 we can create multiple links between different nodes. We can perform various exploration queries using Find and Aggregation command. This allows you to analyze the graph structure and relationships effectively.

Image 9 illustrates how many friends each node has with other nodes using Find and count command

Image 9 : Find and Count


neo4j\$ MATCH (p:Person)-[:FRIENDS_WITH]->(friends) RETURN p.name, COUNT(friends) AS totalFriends;

| | p.name | totalFriends |
|---|-----------|--------------|
| 1 | "Alice" | 2 |
| 2 | "Bob" | 1 |
| 3 | "Charlie" | 1 |

Started streaming 3 records after 8 ms and completed after 9 ms.

neo4j\$ MATCH (alice:Person {name: 'Alice'})-[:FRIENDS_WITH]->(friends)-[:FRIENDS_WITH]->(friendsOfF...

| | friendsOfFriends.name |
|---|-----------------------|
| 1 | "David" |

CONCLUSION:

We showcased the performance of Redis and Neo4j database with suitable examples and various images. Both the databases have different strengths and weaknesses.

Redis is a key-value store that stores data with a simple key-value pair structure. It excels in caching and real-time data processing applications. Redis is known for its exceptional performance and it can handle high write and read loads. It also supports replication and sharding for scalability, but it is not specifically built for handling large-scale datasets as in Neo4j.

On the other hand Neo4j is a graph database that represents data in a graph structure, consisting of nodes and relationships. It is highly suited for managing complex relationships and performing graph-based operations efficiently. It is designed to handle large-scale graph datasets efficiently, with the ability to scale horizontally across multiple machines.

Neo4j and Redis are two popular database management systems, each with its own unique features and benefits, and the choice of which one is to be used depends on the specific requirements of user application.[1][9]

REFERENCES:

1. <https://dev.to/documatic/nosql-databases-vs-graph-databases-which-one-should-you-use-4jpg>
2. Roshani Parate, "NoSQL Databases: Facebook Case study and Analysis", International Journal of Creative Research Thoughts (IJCRT), Volume 11, Issue 7 July 2023 | ISSN: 2320-2882.
3. <https://backendless.com/redis-what-it-is-what-it-does-and-why-you-should-care/>
4. https://books.google.co.in/books?hl=en&lr=&id=RTvcCQAAQBAJ&oi=fnd&pg=PR2&dq=+Graph+Databases&ots=fM9VGDs4mK&sig=k8xvRSKm2BGfq919ZhnuUO08Qgk&redir_esc=y#v=onepage&q=Graph%20Databases&f=false
5. Neo4j: From:<https://neo4j.com/>
6. <https://neo4j.com/docs/getting-started/cypher/>
7. <https://image.slidesharecdn.com/neo4j-131011114420-phpapp02/95/neo4j-graph-storage-7-638.jpg?cb=1381492129>
8. <https://neo4j.com/docs/getting-started/cypher-intro/subqueries/>
9. <https://stackshare.io/stackups/neo4j-vs-redis>