Non-Relational Data Storage and Retrieval Systems

A relational database is a list of linked data items that are organized in tables with rows and columns (Thakur & Gupta, 2021). These rows and columns store information within them, where each column, aside from the first column, has a data type (Thakur & Gupta, 2021). The relational model was first introduced by Edgar F. Codd in 1970 (Codd, 1970) and has since become the most common type of database (Gupta et al., 2017). Relational databases were created to allow a high degree of the data to be stored in the database and not in the application, the ease of adding, updating, or deleting data, is used for data summarization, retrieval, and reporting, and it is easy to make changes to the schema of the database (Jatana et al., 2012). One of the most widely used relational database models is SQL (Anderson & Nicholson, 2022).

SQL, which stands for "Structured Query Language" is a querying language that has been used by many to manage relational database data since the 1970s (Anderson & Nicholson, 2022). SQL databases are adept at handling data that has relationships with other associated data (Anderson & Nicholson, 2022). SQL databases scale vertically by moving to a larger server that adds more CPU, RAM, or SSD capability (Anderson & Nicholson, 2022). SQL databases must display the acronym ACID (Atomicity – all transactions must succeed or fail; Consistency – the database must follow rules that validate and prevent corruption; Isolation – concurrent transactions cannot affect each other, and Durability – transactions are final), properties that ensure transactions are processed successfully and the database has high reliability (Anderson & Nicholson, 2022). A few examples of SQL databases are MySQL, PostgreSQL, CockroachDB, Oracle Database, and Microsoft SQL Server (Anderson & Nicholson, 2022).

SQL is a good option when working with relational data – when the data within the database have a relation to each other (Anderson & Nicholson, 2022). What happens when you have data that's not relational? What about when the amount of data is large? Non-uniformity can pose a challenge with growing data, hence the need for non-relational databases, and databases that can scale, and manage large amounts of data (Gupta et al., 2017). Enter NoSQL databases. NoSQL, also known as "not only SQL", are database that provides an alternative form of data storage in a non-tabular form that's different from relational tables (Gupta et al., 2017). NoSQL databases are often categorized within one of four categories according to how they store the data they store (Anderson & Nicholson, 2022). These database categories include key-value stores, column-oriented, document store databases, and graph databases (Anderson & Nicholson, 2022). A few examples of NoSQL databases are MongoDB, Hypertable, Apache

CouchDB, Apache Cassandra, and RavenDB (Li & Manoharan, 2013). NoSQL databases house data within one data structure, such as a JSON document, and since this non-relational database design does not require a schema, it offers rapid scalability to manage large and typically unstructured data sets (Anderson & Nicholson, 2022).

The problems NoSQL databases were created to solve were challenges relational databases couldn't solve (Gupta & Patel, 2023). It's best to use a NoSQL database when the availability of big data is a priority (Anderson & Nicholson, 2022). NoSQL databases are great choices for when data sets often change, and when working with unstructured data that don't fit into the relational database model (Anderson & Nicholson, 2022). NoSQL databases also have the added benefit of scalability (Anderson & Nicholson, 2022). NoSQL databases can take on large amounts of data spread across many servers, which allows them to have no single points of failure (Anderson & Nicholson, 2022).

NoSQL can store the same kind of data used in relational databases, the data is just stored differently (Gupta & Patel, 2023). As previously stated, there are 4 different types of NoSQL database categories, or structures. There's column-oriented, where data is stored in an unlimited number of columns; key-value stores that utilize a map, and represents a collection of key-value pairs; document stores, which utilizes document to hold and encode data in formats such as XML, YAML JSON, and BSON; and graph databases that represent data on a graph showing how the data relates to each other (Anderson & Nicholson, 2022). A few examples of NoSQL databases are MongoDB, Cassandra, and CouchDB (Anderson & Nicholson, 2022).

MongoDB, which came about in the mid-2000s, is a NoSQL high-volume data storage database that uses collections and documents instead of tables (Taylor, 2023). The documents MongoDB utilizes consist of key-value pairs which are the basic unit of data (Taylor, 2023). The documents can each be of a different size and the data can all be different within the document (Taylor, 2023). The fields the documents utilize can be created on the spot since the documents don't need to have a schema defined prior (Taylor, 2023). MongoDB is highly scalable, with companies defining clusters with millions of documents within the database (Taylor, 2023). Some of the key components of MongoDB are the collection – a group of MongoDB documents (the equivalent to a table in a relational database); a cursor – the cursor points to the result of a query, the database – a container for collections, whereas a MongoDB server can have multiple databases; the document – consisting of field names and values, this is a record in a MongoDB collection; and the field – the name-value pair in a document (Taylor, 2023).

MongoDB is a document-oriented database, making MongoDB very flexible and adaptable (Taylor, 2023). MongoDB allows the user to search by field, range queries, and regular expressions (Taylor, 2023). MongoDB supports indexing, making searching for documents easier (Taylor, 2023).

CassandraDB is another NoSQL database. Managed by the non-profit organization Apache, this data storage system utilizes a distributed architecture, which allows Cassandra to provide high availability and reliability (BasuMallick, 2023). Created for Facebook, the Apache Software Foundation currently maintains it (BasuMallick, 2023). Cassandra has been used by companies such as Netflix, Twitter, and Reddit (BasuMallick, 2023).

Casandra's architecture is made up of clusters of nodes, and each node is where the data is stored (BasuMallick, 2023). Cassandra's architecture can be expanded to hold more data without overwhelming the system, which allows for easy scaling up or downwards (BasuMallick, 2023). Cassandra uses a partitioning system where a partitioner decides where data is stored (BasuMallick, 2023). Nodes in the cluster have tokens based on a partition key, and a hash function is applied to the key when data enters the cluster (BasuMallick, 2023). The coordinator node ensures data is sent to the node with the matching token for that partition (BasuMallick, 2023).

Some of the key features of Cassandra are: Cassandra is open-source and free to use; the data is distributed across multiple equal nodes, preventing bottlenecks and ensuring data accuracy; easy horizontal scaling without service interruptions, providing smooth operation; utilizes Cassandra Query Language, which is similar to SQL but tailored for tabled-based data; data replication across nodes guarantees high availability and resilience against faults; permits the creation of rows and columns as needed, so Cassandra is schema-optional; Cassandra offers both eventual and strong consistency, allowing developers to choose as per requirements; efficient data handling, including writing to commit logs and Memtables, enables rapid writes; and all nodes are equal and communicate as peers, eliminating single points of failure (BasuMallic, 2023).

CouchDB is an open-source NoSQL document-oriented database (Kumarmanish, 2022). It stores data in documents with fields represented as key-value maps (Kumarmanish, 2022). Data is stored in JSON format, queries are written in JavaScript using MapReduce, and it utilizes HTTP for its API. CouchDB supports various data formats and protocols for storage, transfer, and data processing (Kumarmanish, 2022). CouchDB uses a unique identifier and revision number for each document, enabling change tracking (Kumarmanish, 2022). Unlike relational databases, it doesn't use tables; each database comprises independent documents with self-contained schemas (Kumarmanish, 2022). Multiple databases can be accessed, and document metadata facilitates merging changes after

disconnection (Kumarmanish, 2022). CouchDB employs MVCC to avoid locking during writes, leaving conflict resolution to the application, typically involving data merging and deletion of outdated versions (Kumarmanish, 2022).

CouchDB offers the following features: CouchDB simplifies replication, making it exceptionally easy to replicate data; embracing NoSQL, CouchDB stores data as documents with uniquely named fields, accommodating various data types; CouchDB's file layout adheres to all ACID properties, ensuring data integrity; CouchDB offers database-level security with distinct permissions for readers and admins, allowing both read and write access; CouchDB's popularity is driven by its robust map/reduce system for data processing; CouchDB offers open authentication via session cookies, similar to web applications; CouchDB supports replication to offline-capable devices, ensuring seamless data synchronization when they reconnect; CouchDB guarantees eventual consistency, combining availability and partition tolerance; and each item has a unique URI exposed through HTTP, with support for CRUD operations (Create, Read, Update, Delete) using HTTP methods like POST, GET, PUT, DELETE (Kumarmanish, 2022).

CouchDB stands out for its user-friendliness, which is attributed to its HTTP-based REST API, which simplifies interactions with the databases (Kumarmanish, 2022). CouchDB follows a straightforward structure of HTTP resources and methods (GET, PUT, DELETE) making it easily comprehensible and usable (Kumarmanish, 2022). CouchDB is a flexible document-based data storage that eliminates concerns about data structure (Kumarmanish, 2022). Additionally, CouchDB offers robust data mapping capabilities for querying, combining, and filtering information. Moreover, it supports hassle-free replication, enabling data copying, sharing, and synchronization across databases and devices.

One of the biggest advantages of NoSQL databases is they offer scalability, accommodate various data formats, and make data storage and retrieval more straightforward (Foote, 2022). NoSQL databases are particularly well-suited for cloud computing and the internet, enabling a scale-out architecture through data distribution across a cluster of computers (Foote, 2022). They feature flexible schemas, easy updates, high-speed data processing, scalability, and the ability to handle structured, semi-structured, and unstructured data (Foote, 2022). NoSQL databases foster strong user communities for sharing experiences and solutions (Foote, 2022). They are developer-friendly, allowing for easy application development and modification, and using JSON to control data structure

(Foote, 2022). NoSQL databases store data in their native format, eliminating the need for data adaptation and simplifying the application development process (Foote, 2022).

Despite the advantages of NoSQL databases, NoSQL databases also have their drawbacks. One of the drawbacks is that NoSQL databases lack SQL, which is a well-established technology for data organization, causing challenges for analysts accustomed to SQL (Foote, 2022). Each NoSQL database has its unique schema, making it necessary to understand the strengths and weaknesses of each system before choosing the right one (Foote, 2022). NoSQL databases do not support ACID properties, impacting data integrity. ACID includes Atomicity, Consistency, Isolation, and Durability (Foote, 2022). Unlike SQL databases, NoSQL databases lack "JOIN" operations, which complicates data reliability and complex queries (Foote, 2022).

Another example of a NoSQL database would be the graph database. In the graph database, information is organized as nodes connected by relationships (Memgraph, 2023). These databases, following a labeled-property graph data model, consist of four key elements:

1. Nodes – representing primary entities in the graph, they may also be called vertices or points (Memgraph, 2023).
2. Relationships – serving as links between these entities, sometimes referred to as edges or links (Memgraph, 2023).
3. Labels – attributes used for grouping similar nodes (Memgraph, 2023).
4. Properties – key-value pairs stored within nodes or relationships (Memgraph, 2023).

Graph databases mostly use Cyber Query Language (CQL) since it is widely adopted, fully specified, and is an open query language for property graph databases (Memgraph, 2023). Although other graph query languages exist, Cypher stands as the most popular choice due to its user-friendly approach to working with property graphs (Memgraph, 2023).

The problems graph databases were created to solve include: when data has intricate relationships and complex connections, traditional relational databases struggle with slow and resource-intensive join operations (Memgraph, 2023). Graph databases are designed to efficiently manage and analyze such highly interconnected data, making them the solution of choice (Memgraph, 2023). Graph databases excel at retrieving data and performing in-depth graph analytics to uncover hidden patterns within the data. They are ideal when data retrieval and complex analysis are a priority, especially in scenarios with numerous data connections (Memgraph, 2023). Graph databases are well-

suited for data models that evolve frequently and lack a consistent structure (Memgraph, 2023). They offer flexibility, making them beneficial when additional attributes are likely to be added, not all entities have all attributes, or attribute types are not strictly defined (Memgraph, 2023).

This paper will discuss two graph databases, Neo4j and ArangoDB. Neo4j is a unique type of database known as a native graph database (Sekhon, 2020). Unlike traditional databases that use rigid tables, Neo4j offers a flexible structure that preserves the relationships between data points. It stores data in a way that mirrors how it's conceptualized on a whiteboard, using pointers to navigate and traverse the graph (Sekhon, 2020). Neo4j combines the benefits of graph processing with full database features, including ACID transaction support, cluster capabilities, and runtime failover, making it suitable for production data scenarios (Sekhon, 2020).

Using Neo4j offers several advantages such as Neo4j's ability to identify intricate patterns that are hard to spot with traditional databases (Sekhon, 2020). This makes it invaluable for fraud detection (Sekhon, 2020). It excels in analyzing the complex relationships within data, enabling the detection of fraud networks and their elusive mechanics (Sekhon, 2020). Neo4j is a solution to the data explosion in data center management. Its flexibility, high performance, and scalability allow organizations to effectively handle vast amounts of data, including information, devices, and users (Sekhon, 2020). This makes it possible to manage, monitor, and optimize both physical and virtual networks, even when dealing with large data volumes (Sekhon, 2020). Neo4j is a valuable tool for addressing the challenges of master data management. It helps companies create a centralized and reliable information system, ensuring consistency in data formats and applications across the organization. This streamlined approach fosters a working protocol that benefits the entire organization (Sekhon, 2020).

ArangoDB sets itself apart as a native multi-model database, offering the flexibility to store data as key-value pairs, graphs, or documents (Singh, 2022). Its unique feature is the ability to access and combine these data models using a single declarative query language (Singh, 2022). This native multi-model approach enables the creation of high-performance applications and horizontal scalability across all three data models (Singh, 2022).

ArangoDB is a versatile graph database that offers a unique combination of benefits. It stands out by providing flexible data modeling, where both vertices and edges are stored as full JSON documents (Singh, 2022). This flexibility allows for complex nested properties within these documents (Singh, 2022). ArangoDB ensures efficient graph query performance through specialized hash indexes on key attributes, enabling constant look times (Singh, 2022). What sets it apart further is its capability for horizontal scaling, a feature not common in many graph

databases (Singh, 2022). Additionally, ArangoDB offers a wide array of graph-related features, such as graph traversals, shortest path calculations, and pattern matching (Singh, 2022). It also provides visualization tools like Graph Viewer (Singh, 2022). What makes ArangoDB truly exceptional is its seamless integration with other data models, allowing users to combine graph analysis with document and key-value data models in a single query, making it a versatile choice for a range of data analysis needs (Singh, 2022).

Database strength is evaluated based on four key factors: integrity, performance, efficiency, and scalability (Technical Matters, 2019). Graph databases excel in delivering quicker and simpler data queries, especially when compared to relational databases which may encounter limitations as complexity and data volume increase (Technical Matters, 2019). The graph-based model aligns well with the natural way facts are stored, mirroring human thinking and offering clear links between data points (Technical Matters, 2019). However, graph databases do have limitations, particularly in terms of scalability when they are primarily designed for one-tier architecture (Technical Matters, 2019). Additionally, the lack of standardized query language remains a challenge in this database model.

This paper explores NoSQL and Graph databases. These two database models provide their particular advantages, but there are also some drawbacks as well that you need to be aware of while choosing the database model to use for your application. Make sure you read through the database model's documentation to find out if it's the best database model for you.

1. Gupta, A., Tyagi, S., Panwar, N., Sachdeva, S., & Saxena, U. (2017). NoSQL databases: Critical Analysis and comparison. *2017 International Conference on Computing and Communication Technologies for Smart Nation (IC3TSN)*. https://doi.org/10.1109/ic3tsn.2017.8284494

2. Berg, K. L., Seymour, T., & Goel, R. (2012). History of databases. *International Journal of Management & Information Systems (IJMIS)*, *17*(1), 29–36. https://doi.org/10.19030/ijmis.v17i1.7587

3. Li, Y., & Manoharan, S. (2013). A performance comparison of SQL and NoSQL databases. *2013 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*. https://doi.org/10.1109/pacrim.2013.6625441

4. Anderson, B., & Nicholson, B. (2022, June 12). *ibm.com*. October 28, 2023, ibm.com/blog/sql-vs-nosql/

5. Thakur, N., & Gupta, N. (2021). Relational and Non Relational Databases: A Review. *Journal of the University of Shanghai for Science and Technology*, *23*(08), 117–121. https://doi.org/10.51201/jusst/21/08341

6. Codd, E. F. (1970). A relational model of data for large shared data banks. *Communications of the ACM*, *13*(6), 377–387. https://doi.org/10.1145/362384.362685

7. Jatana, N., Puri, S., Ahuja, M., Kathuria, I., & Gosain, D. (2012). A survey and comparison of relational and non-relational databases. *International Journal of Engineering Research & Technology*, *1*(6), 1-5.

8. Gupta, P., & Patel, P. (2023). Demystifying databases: Exploring their use cases. *International Journal of Computer Science and Engineering*, *10*(6), 43–53. https://doi.org/10.14445/23488387/ijcse-v10i6p106

9. BasuMallick, C. (2023, August 17). *What is Cassandra? working, features, and uses*. Spiceworks. https://www.spiceworks.com/tech/big-data/articles/what-is-cassandra/

10. Kumarmanish. (2022, December 23). *What is CouchDB?* DevOpsSchool.com. https://www.devopsschool.com/blog/what-is-couchdb/#google_vignette

11. Foote, K. D. (2022, November 17). *NoSQL databases: Advantages and disadvantages*. DATAVERSITY. https://www.dataversity.net/nosql-databases-advantages-and-disadvantages/

12. *Graph database vs relational database*. Memgraph. (2023, May 8). https://memgraph.com/blog/graph-database-vs-relational-database

13. Sekhon, S. (2020, December 24). *What is neo4j?*. DEV Community. https://dev.to/sukhbirsekhon/what-is-neo4j-8jc

14. Singh, P. (2022, December 29). *ArangoDB – A graph database*. StatusNeo. https://statusneo.com/arangodb-a-graph-database/

15. Technical matters. (2019, October 30). *Graph databases explained*. IONOS Digital Guide. https://www.ionos.com/digitalguide/hosting/technical-matters/graph-database/