JCBI
Journal of Computing & Biomedical Informatics
ISSN: 2710 - 1606
Volume 07 Issue 02
2024
Research Article
https://doi.org/10.56979/702/2024

A Systematic Literature Review on Performance Evaluation of SQL and NoSQL Database Architectures

Muqaddas Salahuddin[1], Samra Majeed[1], Sammia Hira[1], and Gohar Mumtaz[1*]

[1]Faculty of Computer Science and Information Technology, Superior University, Lahore, 54000, Pakistan.
[*]Corresponding Author: Gohar Mumtaz. Email: gohar.m@superior.edu.pk

Received: April 12, 2024 **Accepted:** July 20, 2024 **Published:** September 01, 2024

Abstract: The Maintaining large volumes of data in SQL and NoSql databases depends on programming architecture. NoSql databases excel in horizontal scalability and handling unstructured data, while SQL databases are designed for data organization and horizontal scalability. Choosing between SQL and NoSql databases can be challenging due to differences in database design and hierarchical needs. NoSql databases use a mixed-model strategy, complicating data movement between cloud storage providers, as different platforms experiment with various concepts. Systematic literature reviews (SLRs) analyze papers on NoSql and SQL database designs, interoperability, and cloud data portability. Recent research comparing Oracle RDBMS and MongoDB suggests that NoSql databases are better for big data analytics due to their customized architectures, whereas SQL databases are more suitable for online transaction processing (OTP).

Keywords: MapReduce; DBaaS; ACID; NoSql Database and Sql Databases; BASE; Big Data; Aggregation.

1. Introduction

An application's architecture includes non-functional elements like data sharding, interoperability, scalability, performance, dependability, and usability. A software architect must balance these quality attributes. Distributed systems, often big data systems, face challenges with data availability and consistency due to sharding and replication. Database systems have evolved with the rise of data applications. Over the past decade, NoSQL databases have rapidly developed, offering more flexible models compared to traditional databases, which struggle with scalability due to rigid schema structures. Key features of NoSQL database designs include:

- Permitting the efficient and dynamic growth of data representations
- Structure without Schema
- Horizontal scaling over large clusters with sharding and data replication collections.

In recent years, businesses have gathered large amounts of data that relational databases struggle to handle. Software architects must balance constant quality attributes. Relational databases have been widely used for over forty years, ensuring availability, isolation, consistency, and durability through ACID principles. "Big data" involves scalable techniques to manage vast amounts of data, defined by the five Vs: variety, value, veracity, velocity, and volume. This encompasses a wide range of unstructured and heterogeneous data. Frameworks like Spark, Flink, Hadoop/MapReduce, and Samza process these massive datasets [1].

Recently, there has been a growing interest in enterprise, production, parallel databases, big data, and SQL query optimization. Inefficient searches can waste system resources and cause database locking and data loss. Data mining goes beyond simple data analysis by uncovering correlation patterns in datasets. Query optimization, which determines the most efficient query execution strategy, is crucial. Data mining methods are essential for extracting patterns and knowledge from large datasets, requiring complex and resource-intensive searches [2].

Over the last decade, the one-size-fits-all approach to data storage has been questioned, especially in scientific and online retail sectors. NoSQL ("Not only SQL") emerged to address these concerns. The term NoSQL appeared in 1998 and gained popularity at a San Francisco conference on non-relational databases. NoSQL databases, primarily used by web developers, differ from traditional relational databases [3]. Figure 1 summarizes their main traits.

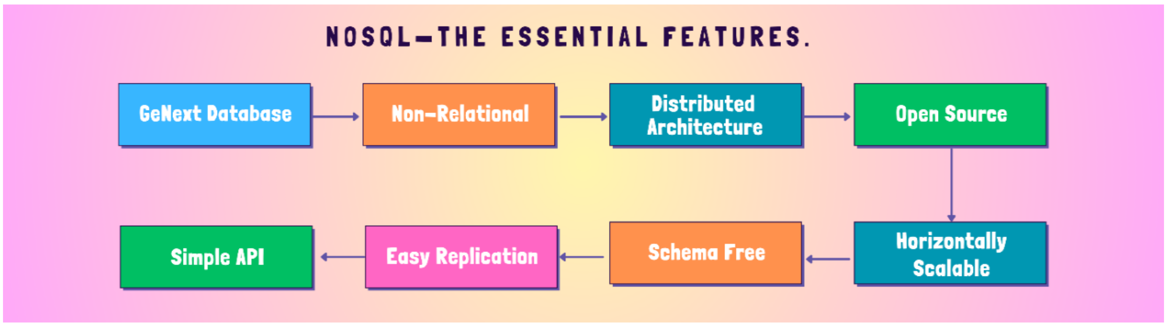

Figure 1. Essential features of database of NoSql

The data consistency, the availability of data, and Partition Tolerance (CAP) hypothesis was introduced by Eric Brewer [4]. Table 1 lists the primary features of THEORY of CAP.

Table 1. Theory of CAP

Consistency of data	Availability of data	Partition tolerance
• The term of consistency refers to fact that after an operation is completed, in database the data is stays consistent. • For example, after an update transaction, the identical data is visible to every client.	• Availability guarantees 100% service uptime, meaning there won't be any downtime for the system. • Every node always does the query, assuming nothing goes wrong.	• Partition tolerance refers to the system's ability to keep running even in the event that server connection is unstable. • It's possible to divide the servers up into several groups that aren't able to speak to each other.

Achieving all three requirements at the same time is considered theoretically impossible. As a result, a distributed system must give two of the three components priority according to CAP theory. In Figure 2 as a result, all current NoSQL databases support different partition tolerance (P), availability (A), and consistency (C) combinations according to the CAP.

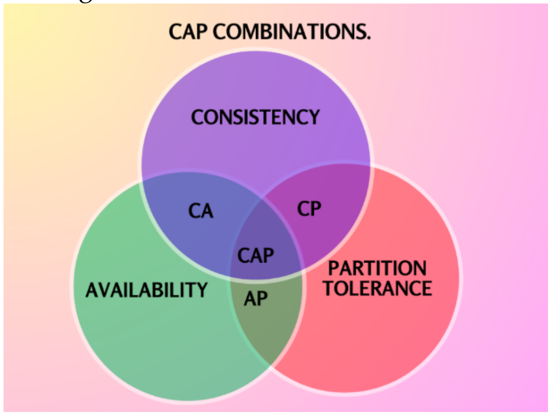

Figure 2. CAP Diagram

Database of NoSql advantages are as follows:

- Volume: Existing information that needs to be processed, ranging from terabytes to exabytes.
- Velocity: Data in motion; streaming data; reaction times ranging from milliseconds to seconds.
- Variability: Data can be textual, unstructured, structured, or in other formats.
- Veracity: Uncertainty resulting from delays, deceit, etc.

- Not based on tables and doesn't use SQL for data manipulation.
- Columnar, graphs, Key-values, documents, etc. are all included in a schema.
- An alternative to relational databases as they are known.
- Database to manage disorderly, unexpected, and unstructured data.
- Useful when handling huge distributed data sets.

Relational databases and NoSql databases have different architectures and models, therefore when implementing NoSQL databases in the cloud, interoperability, portability, and security must be carefully taken into account. Because NoSQL databases are heterogeneous, even though cloud service providers (CSPs) offer scalability, availability, and privacy features [5, 6].

The following are this Systematic Literature Review primary contributions:

- The analysis of performance, scalability, and architecture evaluations of NoSql databases and Sql databases—specifically, RDBMS Oracle and the Document of NoSql DB (MongoDB)—is covered in this systematic literature review (SLR). The evaluation also looks at data migration procedures from one database to another that is housed on a different cloud platform.
- The aforementioned study objectives have been achieved through the analysis of 33 publications in total.
- This article identifies the underlying causes of gaps of research within related architectures.

1.1. Current Problem Situation

The growth of big data has increased the need for scalable systems to manage large volumes of information. While relational databases using SQL can handle semi-structured, unstructured, and structured data, they have scaling issues. NoSQL databases excel with massive datasets and adapting to changing data types due to their horizontal scalability and BASE principle.

There are several types of NoSQL databases: document, column, graph, and key-value. Graph databases, for instance, are optimized for relationships between nodes, making them ideal for managing densely interrelated data.

MongoDB is a NoSQL database known for its scalability, effectively handling large data volumes. However, migrating from OLTP databases to MongoDB can lead to issues like distinct indices, combined keys, inconsistent data, and duplicates.

A significant concern when transferring to cloud storage is the security of personal data. Developing a unified cloud solution for NoSQL models is challenging due to interoperability and portability issues in current cloud service provider designs. This SLR includes a thorough analysis of modern security methods and guidelines for NoSQL databases.

1.2. Method

This systematic literature review follows PRISMA guidelines, integrating top primary studies related to specific research questions. SLRs are widely used in software engineering research; our work focuses on data portability and interoperability across cloud platforms, and performance evaluations of SQL and NoSQL databases. Our study is notable for its systematic data collection, comprehensive study coverage, focused investigation, and detailed classification and analysis of selected studies. The following sections of this document are organized as follows:

- The study topics, search terms, inclusion/exclusion criteria, data extraction procedure, and classification methods are all described in depth in Sec 2.
- The conclusions drawn from the chosen papers are presented in Sec 3.
- Research gaps are identified and talks are held in Sec 4.
- In conclusion, Sec 5 shows a synopsis of the findings and suggests next directions.

2. The Objectives and Questions for Research

The current SLR's goals, focus, and purpose are as follows:

1. Examine the current methodologies and strategies for handling SQL and NoSql documents while taking big data processing into account.
2. Conduct a thorough review of the literature on NoSql and SQL databases.
3. Examine a few chosen study subsets in detail.
4. Based on the data gathered and examined from these research, evaluate the merits and demerits of NoSql and Sql databases.

5. Emphasize the gaps in the field's research.
6. Determine the directions for further research.
7. Create the following research inquiries in order to fulfill our study's primary goal:

- Why is NoSql necessary when dealing with big data sets, including both structured and unstructured data?
- Why does the BASE property apply to NoSql databases whereas the ACID property applies to SQL databases?
- Does DBaaS effectively address data portability and interoperability across different databases of NoSql?

2.1. Search Requirements

The major study research requirements entail the identification and gathering of literature that satisfies all exclusion and all inclusion criteria. Many search strategies were employed to achieve this, including manual, electronic database, snowballing, and searches of related publications and conference proceedings. We adhered to the methodology suggested by [7] and the four phases examined in [8] for our SLR. Figure 3 shows the studies that were chosen for association.

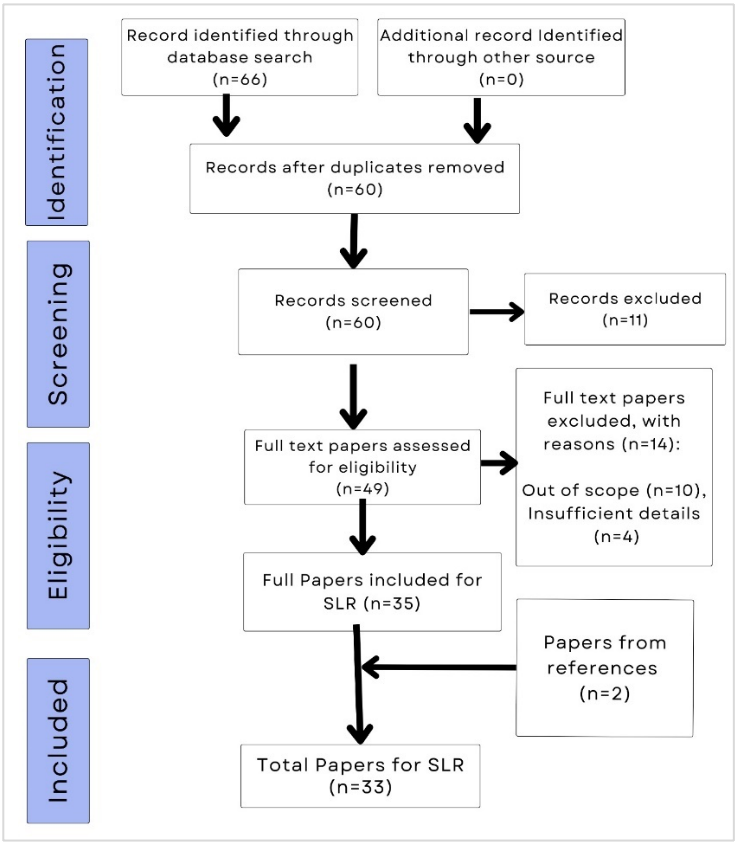
Identification
Record identified through database search (n=66)
Additional record Identified through other source (n=0)
Records after duplicates removed (n=60)
Screening
Records screened (n=60)
Records excluded (n=11)
Eligibility
Full text papers assessed for eligibility (n=49)
Full text papers excluded, with reasons (n=14): Out of scope (n=10), Insufficient details (n=4)
Full Papers included for SLR (n=35)
Included
Papers from references (n=2)
Total Papers for SLR (n=33)

Figure 3. SLR Flow Diagram

First, we performed the database search that [9] suggested. Tools, techniques, and frameworks were categorized and classified using the search strings covered in the section on search strategies.

2.1.1. Resources for Searching

For database searches, we came up with a list of pertinent search phrases. We then used these search terms to find relevant documents. To guarantee comprehensive investigation, we employed significant databases like:

- springerlink.com web link of (Springer)
- dl.acm.org web link of (Digital Libraries ACM on Google Scholar)
- ieeexplore.ieee.org web link of (Digital Libraries DBLP for IeeeXplore)
- onlinelibrary.wiley.com web link of (Web Library at Wiley)
- sciencedirect.com web link of (Elsevier)

2.2. Selection Procedure and Criteria

By using inclusion/exclusion criteria, we were able to include all related papers. Database search involved finalizing the selected list after evaluating the papers' qualities and features in light of our research topics. The sources used in our research article are arranged as below:

Step1: All number of papers determined by:

1. Titles of Papers.

Abstract of Papers.

a. Full reading of Associated papers

(1) Check the attached papers' effect and quality.

- ✓ Examine the paper in the catalog for possible repetition.
- ✓ Add the product to the catalogue of finished article.

(2) Snowballing and Manual search

(3) Completely repeat the procedure; return to Step 1

The number of related publications to the previously indicated points following each steps filtering is shown in Table 2.

Table 2. Criteria and Paper selection articles

Search Source	Based on removed after duplicates	Based on Abstract	Total Papers for SLR
Total No. of Articles	60	35	33

The Table 2 demonstrates that 46 articles were first chosen based on the elimination of duplicates. The first author picked 36 publications after looking over their abstracts; 33 of those were chosen for full reading for SLR. After reading the 33 articles of different researchers, we collected the required data from those papers and described it in literature view with references. The majority of the publications drew from empirical studies, such as the following:

- Research purposes
- Related research and substantiated theories
- Measurement of hypotheses
- Method, design, strategy, dimension, and data gathering that are suggested
- Analysis of data results
- Conclusion

2.2.1. Criteria for Inclusion

The questions we were investigating led us to include the following categories of papers:

- CI1: survey papers and related SLRs
- CI2: newly suggested methods and strategies pertinent to our suggested SLR
- CI3: the suggested study's presentation of efficient research techniques

The related author analyzed and considered the methodology and effects of the included papers to improve the SLR's dependability and effectiveness.

2.2.2. Criteria for Exclusion

The following criteria were used to exclude the research papers:

- CE1: articles unrelated to the specified domain
- CE2: unrelated papers
- CE3: a few papers selected from the abstract and title
- Non-peer reviewed articles and materials (CE4)
- CE4: duplicate articles and articles not written in English

The second and third co-authors reviewed the omitted papers in accordance with the exclusion criteria checklist in an effort to lessen the risk to the SLR's dependability.

When the proposed SLR is presented at a conference or published in a journal, the most recent version will be utilized. The co-authors and corresponding authors reviewed the document quality.

2.3. Data Collection and Extraction

Two reviewers examined the 33 related papers once they had been gathered, extracting important information that addressed the questions of research [10]. The following information was taken from every chosen paper:

- Paper of Title
- Paper of Abstract
- Source of Paper (journal or conference)
- Year of Publication
- Classification of Paper (type, scope)
- SLR proposed to the Relatedness
- Research question issues and Proposed SLR objectives
- Method and Paper summary

We selected a large number of empirical papers with an emphasis on reviews, experiments, discussions, and evaluations of state-of-the-art methods (NoSql and Sql). These empirical studies served as the foundation for our study.

2.4. Data Collection and Extraction

Additionally, the following arrangement and tabulation of data was made for each retrieved data set based on the study research questions. After categorizing all of the collected strategies throughout analysis, we assembled our results into three groups: research methodologies, phases of the evaluation and research process.

2.5. Data Collection and Extraction

According to [11], threats to the SLR process ought to be regularly assessed. These risks were divided into several categories, such as repeatability, generalizability, theoretical validity, interpretive validity, and descriptive validity. Anhui University's Professor Zhang Cheng examined the SLR validity checks, and we changed the technique to reflect his recommendations.

3. Method Results

This section provides an overview of the research we have selected, broken down by publication year, paper type, and the total number of papers we have selected from a certain digital library. The majority of the empirical research articles were picked in accordance with the selection process and criteria. The researcher used both kinds of databases for their suggested procedures and investigations, according to the research literature. Apart from empirical research, we also found survey articles related to NoSql and SQL databases. We divided the chosen research and publications into three primary groups after completing the related review selection process. The studies' category pie chart is shown in Figure 4.

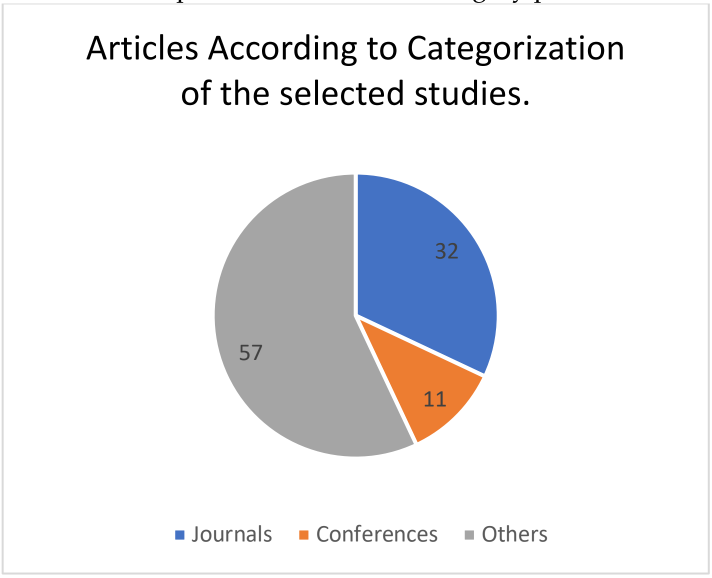

Figure 4. Articles Categorized According to Selected Studies

The chosen research on NoSQL and SQL databases covers the period from 2000 to 2024. The distribution of published articles by year is shown in Figure 5, which also shows a noticeable uptick in interest in NoSQ databases after 2008.

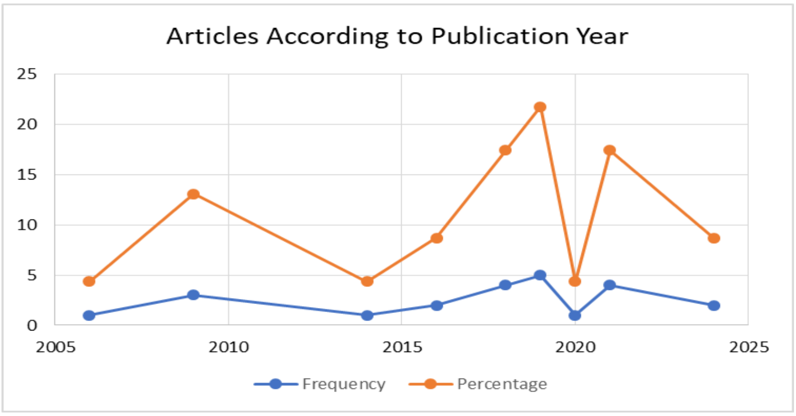

Figure 5. Articles According to Publication Year

The number of articles on NoSQL and relational databases from various digital libraries is shown in Figure 6. Many of the pertinent articles are available in the libraries of digital of IEEE and Springer. Figure 6 also covers a variety of additional sources from publishers like MIT, Oracle, SciTePress, Academic Journal, IJACSA, and IOPScience, including technical reports, book chapters, and white papers.

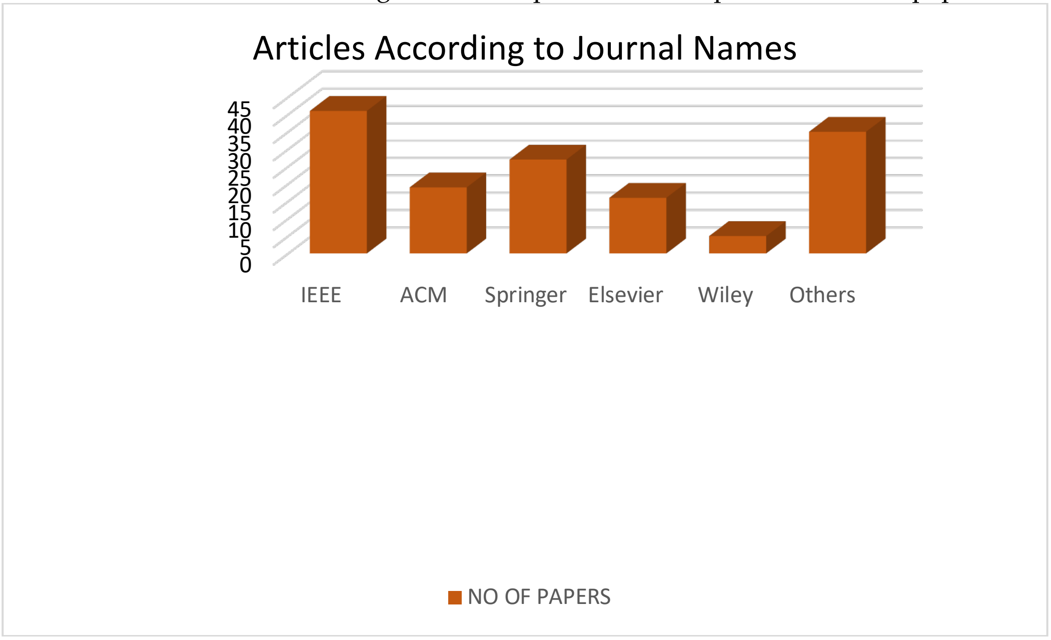

Figure 6. Count of Papers across Digital Libraries

The features and performance of databases such as Sql Server, HBase, MySql, MongoDB and PostgreSql have been compared in several research papers. As shown in Figure 7, the chosen research usually included one or more of these databases.

3.1. Analysis of Empirical Studies:

Empirical studies and publications are classified as evaluations, experiments, categorizations, dialogues, surveys, and comparisons. We evaluated articles based on our selection criteria and hypotheses after selecting them from the literature that we thought would be pertinent to our study themes.

RQ1: Why is NoSql necessary when dealing with big data sets, including both structured and unstructured data?

RQ2: Why does the BASE property apply to NoSql databases whereas the ACID property applies to SQL databases?

NoSQL ("Not only SQL") is a DBM strategy adept at handling large volumes of unstructured data and analysis. Unlike relational databases that use SQL, NoSQL databases offer flexibility with various query languages and dynamic schema structures. Document-oriented NoSQL databases like MongoDB and CouchDB store and retrieve documents in formats such as XML, PDF, JSON, and BSON. Both are open-source, but MongoDB excels with JSON and in distributed environments, using a dynamic schema for faster and more accurate data analysis. MongoDB, powered by C++, also features strong security, backup, and recovery tools. Figure 8 shows NoSQL and SQL databases with their data storage systems [12].

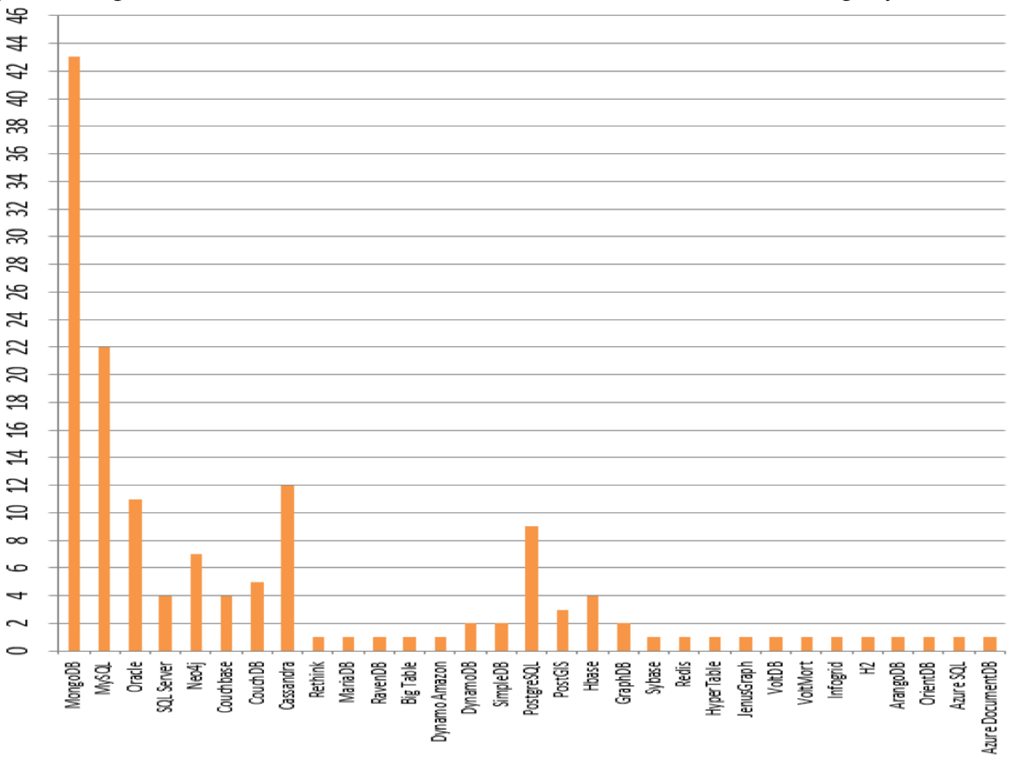

Figure 7. Selected studies used in Databases

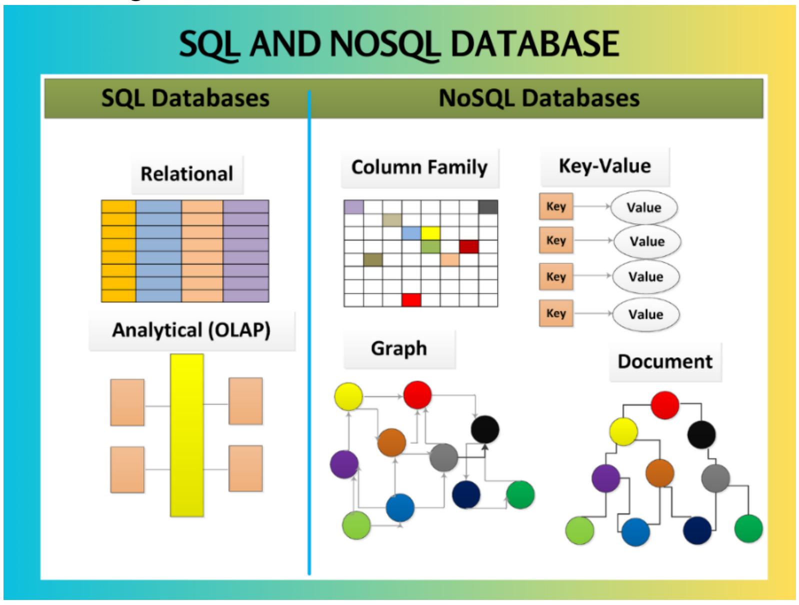

Figure 8. Sql database and NoSql Database

Numerous studies have focused on the structure, design, and performance of NoSQL (e.g., MongoDB, Neo4j) and SQL (e.g., Oracle, MySQL, SQL Server) databases. These studies examined the application of suggested methods to SQL and NoSQL scenarios. NoSQL databases offer unique functionalities and use cases, with horizontal scalability, though they can't maintain the ACID property. Neo4j, a graph database, efficiently organizes and stores complex dependency data. MongoDB focuses on documents, using the BSON format, a JSON variant [13].

MongoDB has proven superior for managing large datasets compared to other databases. A study comparing PostgreSQL and MongoDB for social network services and streaming sensor data found MongoDB performed better. MongoDB excels at handling large, diverse datasets and is more efficient in cluster contexts than traditional RDBMS like MySQL. Another study showed MongoDB managed vast amounts of unstructured data more effectively than MySQL. Detailed analyses confirmed MongoDB's superiority over MySQL for unstructured data. While MongoDB often outperforms relational databases like MySQL and Oracle, a study cautioned against viewing NoSQL as a simple substitute for SQL databases, advising businesses to consider their specific needs when choosing a DBMS [14].

Many studies and publications address the transition from RDB to NoSQL databases, emphasizing data availability, high performance, and scalability. MongoDB is favored for handling large data volumes due to its scalability, security, integrity, and customizable designs. It stores data in BSON format, efficiently managing organized, semi-structured, and unstructured data without the need for joins.

Despite Oracle's RDBMS performing better than MongoDB's MapReduce for specific aggregation tasks, MongoDB is ranked higher in popularity and usage. However, it is important to note that some studies used datasets not suitable for large-scale analytics [15]. Figure 9 illustrates the process SQL statement command flow in the Oracle RDBMS.

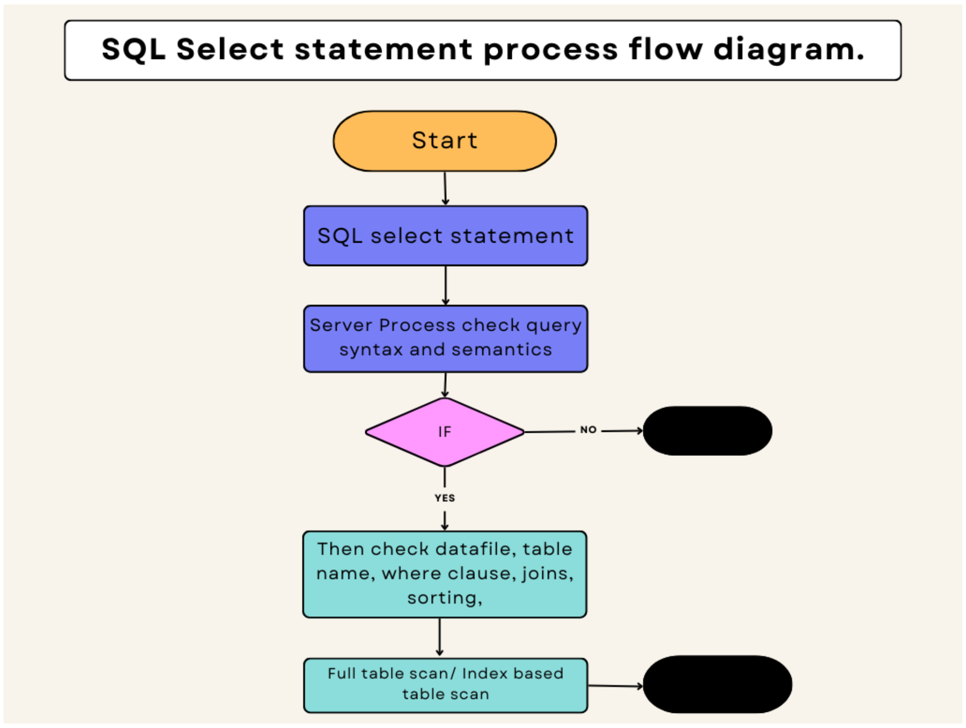
SQL Select statement process flow diagram.
Start
SQL select statement
Server Process check query syntax and semantics
IF
NO
YES
Then check datafile, table name, where clause, joins, sorting,
Full table scan/ Index based table scan

Figure 9. Select query of SQL flow diagram

Because of its sub-document structure, data retrieval in MongoDB is simple and does not involve Verifying limitations or any conditions, unlike the SELECT query Sql of the Oracle 11g RDBMS. The INSERT query of Sql flow is shown in Figure 10.

MongoDB insertion is quicker than Oracle RDBMS since it doesn't need to confirm the actions indicated in Figure 10 of the SQL Insert statement.

MapReduce is a programming model that excels in distributed environments, making it suitable for big data processing compared to simple aggregation. MapReduce consists of "Map" and "Reduce" stages, where documents are processed and sorted, then combined and stored. The MapReduce-Merge framework and MRShare framework have improved MapReduce efficiency and query performance on clusters. Despite these advancements, Oracle RDBMS still outperforms MongoDB in aggregation tasks. Over the past decade, businesses have widely adopted Oracle RDBMS, though its architecture struggles with unstructured data. Studies have examined NoSQL databases on Hadoop, categorizing them by scalability and data type. Table 3 compares the main features of MongoDB and Oracle RDBMS [16].

The volume of geospatial and geolocated data has significantly increased, necessitating robust DBMS to process this data efficiently. NoSQL databases are preferred over SQL for online applications handling large data volumes, including geospatial data. Studies have shown that NoSQL databases efficiently process massive volumes of unstructured data and manage location and geospatial data effectively. Traditional SQL optimization techniques struggle with geographic queries, highlighting the need for well-established methods to handle extensive geographic data.

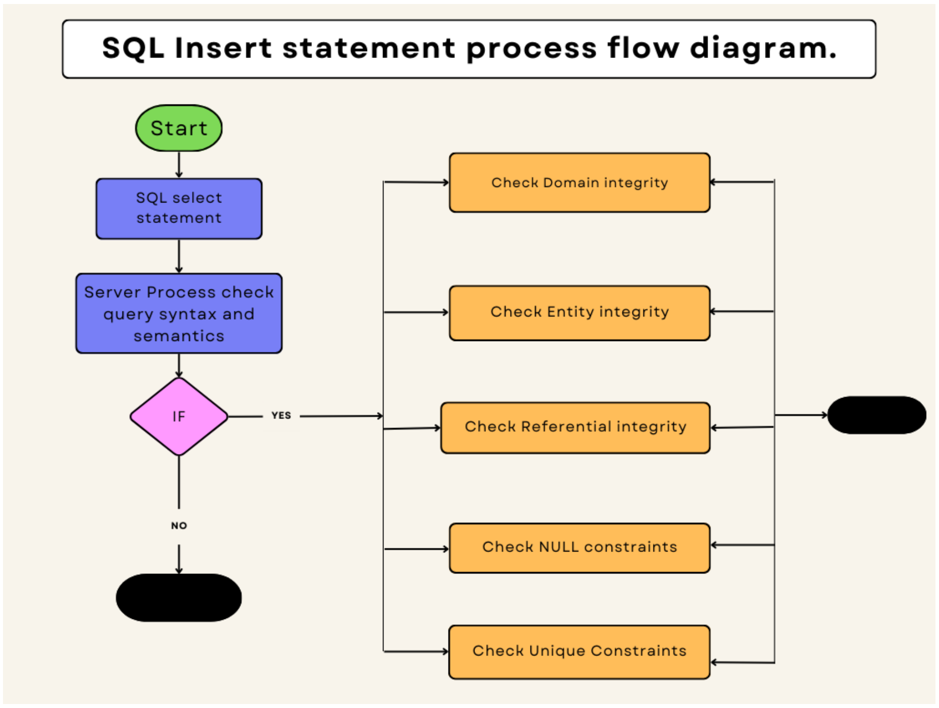

Figure 10. Flow diagram of SQL Insert statement process

Table 3. SQL Database Characteristics and MongoDB Database

SQL	Mongo DB
• Column	• Flexible schema
• Tables	• Collection
• Rows	• Documents
• Rigid schema	• Field
• SQL (Structured Query Language)	• MongoDB Query Language (MQL)
• Manual indexing for optimal performance	• Automatic indexing on _id field
• Full ACID compliance across transactions	• ACID transactions limited to single document operations
• Supports relational algebra	• Supports relational algebra

Table 4. Select query of SQL and MongoDB

SQL	Mongo DB
Select * from students	db.students.find()

Table 5. Insert query of SQL and MongoDB

SQL	Mongo DB
INSERT INTO students VALUES("ali","355","BSCS")	db.students.insert(name : "ali", Roll: "355", degree: "BSCS")

Table 6. Create query of SQL and MongoDB

SQL	Mongo DB
Create Table students(id int,name varchar(21),roll-no int)	db.createCollection("students")

Table 7. Drop query of SQL and MongoDB

SQL	Mongo DB
DROP TABLE students	db.students.drop()

MongoDB has been shown to outperform PostGIS in processing geospatial data. NoSQL databases like Azure DocumentDB and MongoDB provide geographic capabilities, while SQL Server 2016 and Azure SQL Database offer similar functions in the cloud. PostgreSQL, with its PostGIS plugin, handles geospatial data as a spatial database. Azure DocumentDB, developed by Microsoft, supports MongoDB's features and geographic operations using the GeoJSON standard format. Performance studies indicate that Azure DocumentDB performs faster than Azure SQL Database but is less scalable. Table 4 lists the main geographical attributes of popular NoSQL and SQL databases [17].

Table 8. Geospatial characteristics of NoSQL and SQL databases

Database	PostGIS	Oracle	MongoDB	Azure SQL	Document DB
Supported Geometries Objects:	Polygon, LineString, MultiPoint, MultiPolygon, Point, MultiLinePoint, GeometryCollection	LineString, MultiPolygon, Polygon,MultiPoint, MultiLinePoint, GeometryCollection, Point,	Polygon, Point, MultiLinePoin, GeometryCollection, MultiPoint, MultiPolygon, LineString,	MultiPolygon, Point, MultiLinePoint, MultiPoint, GeometryCollection, LineString, Polygon,	LineString, Polygon, Point, MultiPolygon, Geometry Collection, MultiLine Point, MultiPoint,
Geometry Functionalities Supported	Oracle supports Open Geospatial Consortium (OGC) for handling geometry instances.	Oracle supports Open Geospatial Consortium (OGC) for handling geometry instances.	Inclusion, Intersection, and Distance/Proximity are key operations	Oracle handles geometry objects via supporting Open Geospatial Consortium (OGC).	Inclusion, Intersection, and Distance/Proximity are key operations
Spatial Indexes Supported	R-Tree indexes GiST indexes, B-Tree indexes,	B-Tree and parallel indexes build are utilized for spatial R-trees indexe/indexes	2D sphere indexes, 2D indexes	B-Trees, 2D plane indexes	Quadtree, Plane 2D indexes,
DBaaS	×	✓	✓	✓	✓
GeoServer Compatibility	✓	✓	✓	✓	✓
Horizontal Scalability	×	×	✓	×	✓

3.2. MongoDB Data Modeling in NoSql

The routing servers, configuration servers, and shard nodes (sometimes referred to as "mongos") that make up architecture of the MongoDB are depicted in Figure 11 and further explained in [18].

In a MongoDB cluster, data is stored in shards, each with replicas on different nodes to ensure availability in case of failure. Read/write transactions select the appropriate shard, with a primary server mirrored by secondary nodes. If the primary server fails, a backup takes over. Configuration servers manage metadata, identifying and broadcasting shard data. User tasks are routed by MongoDB, grouped by type, allocated to relevant shards, and combined before client confirmation. Mongos can be used in a distributed environment as they are stateless.

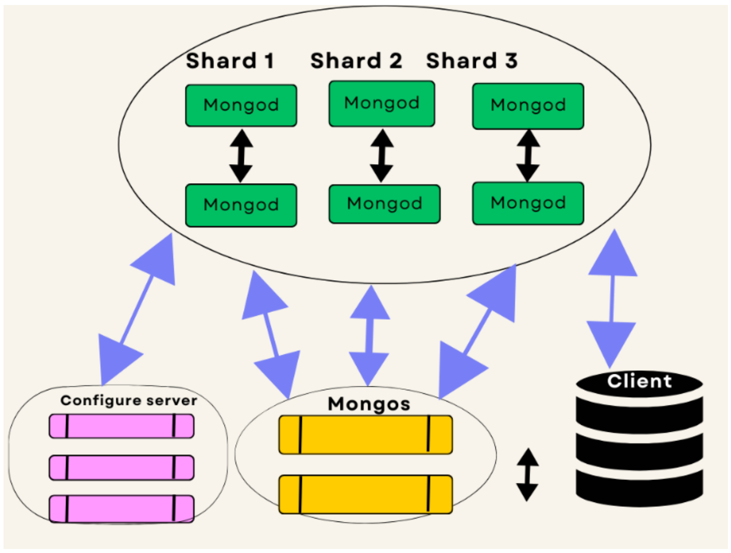

Figure 11. Architecture of MongoDB

RQ3: Does DBaaS effectively address interoperability and data portability across different NoSql databases?

A thorough literature review of DBaaS architecture revealed that cloud DBaaS designed for RDB is not optimal for NoSQL databases. Standard APIs eliminate the need to re-engineer applications for different CSPs. The main challenges are interoperability and data portability between cloud providers. Interoperability is defined differently across PaaS, SaaS, and IaaS paradigms, with our focus on the IaaS layer. Unified APIs are required for data transfers across cloud providers, as not all use the same data storage model. Figure 12 shows the high-level architecture for data exchange during CSP migration [19].

Efficient management and consistent databases are crucial in today's IT landscape. Declarative query capabilities ensure data independence from physical storage in database systems, supporting various data models like relational, XML, and NoSQL, which handle large data volumes with the BASE characteristic. Cloud services providers offer new capabilities efficiently and cost-effectively but use different implementations, leading to portability and interoperability issues [20].

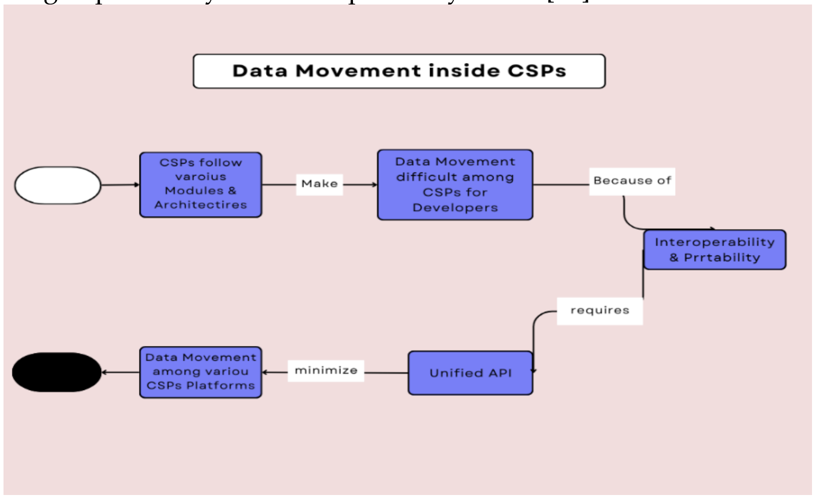
Data Movement inside CSPs
CSPs follow varoius Modules & Architectires
Make
Data Movement difficult among CSPs for Developers
Because of
Interoperability & Prrtability
requires
Unified API
minimize
Data Movement among variou CSPs Platforms

Figure 12. CSPs inside Data Movement

Interoperability across PaaS, SaaS, and IaaS paradigms is challenging, and switching cloud services providers can be driven by various factors like outages or cost. Vendor lock-in and security risks complicate this further. Open standards like OVF and CIMI aim to address these issues, along with initiatives like MOSAIC, MODACLOUDS, and Cloud4SOA. However, unique PaaS APIs limit their effectiveness.

Migration tools, cloud infrastructures, and SDCPs facilitate data transfers but don't fully resolve portability issues [21,22]. Major cloud services providers like Microsoft Azure, Google App Engine, and AWS help develop cloud applications but require standardized APIs for seamless data transfers [23].

Unified API frameworks for SQL and NoSQL databases, like CdPort and Se-cloudDB, enhance data portability, security, and interoperability. These frameworks protect user data and convert requests into compatible models for different databases, ensuring secure and authorized data access [24].

4. Discussion

The debate between SQL and NoSQL databases isn't about relational versus non-relational models but their transaction models. SQL databases adhere to the ACID properties—Atomism, Consistency, Isolation, and Durability—for every transaction. In contrast, NoSQL databases, seeing ACID as a barrier, adopted Eric Brewer's CAP theorem principles: Consistency, Availability, and Partition tolerance. This theorem allows developers to design partition-tolerant databases that can ensure either availability or consistency. Table 5 outlines the primary functionalities of each database type.

Table 9. Features of Database Management Systems

Features of Database Management Systems							
DB MSs	**The Data**	**The Schema**	**The Scalability**	**The Compliance**	**The Architecture**	**The Consistency**	**The Perfor mance**
NoS ql	Structured ,Un Structured , semi Structured	Dynamic	Horizontal	BASE	Distributed	Eventual	Fast
RDB MS	Structured	Fixed	Vertical	ACID	Centralized	Strict	Slow

The classification of DBMSs has multiple subcategories, with the relational data model being key to many modern systems. NoSQL databases differ from traditional SQL-based systems in their data model and querying techniques, often requiring developers to handle query execution, data verification, and consistency tasks. Brewer's CAP theorem outlines requirements for distributed databases: in the presence of network partitions, consistency (C) must take precedence over availability (A), while the reverse is true in their absence. NoSQL databases excel in handling large, unstructured data due to their scalability, real-time access, flexible schemas, and storage capacity, following the BASE characteristics [25]. They prioritize read/write performance over data consistency, making them ideal for big datasets without strict data-level constraints.

This study reviewed approximately 33 previous studies comparing the productivity, reliability, and utility of NoSQL and SQL databases. Our research shows that NoSQL databases offer more scalability, better handling of diverse datasets, and require fewer resources for data integrity and consistency compared to SQL databases. However, SQL databases like MySQL and Oracle, built on rigorous theoretical models such as relational algebra, are more suitable for transactional applications and require more maintenance. NoSQL databases prioritize data accessibility and excel in cluster environments, making them ideal for parallel computing with the MapReduce programming module. Ultimately, the choice between NoSQL and SQL depends on an organization's specific needs [26,27].

Relational databases rely on a fixed schema (tabular format), while NoSQL databases use a more flexible and dynamic schema. For example, relational databases require predefined fields like StdRegNo, StdName, and StdAddress for student data, adhering to strict integrity and domain rules.

Both SQL and NoSQL databases use various models, each with different underlying data structures. Data portability is challenging due to the diversity of cloud service providers (CSPs). The rapid growth of business databases adds complexity to cloud storage, with AWS DBaaS offering limited storage extension options for Azure databases [28]. Table 6 lists product languages, categories, database types, and architectures.

Table 10. Database architecture type, product category and its languages

Databases Name	Databases Types	Databases Architectures	Databases Category	Written in
MySQL	Open Source		Sql	C, C++
MongoDB	Open Source	Multi-Model of Distributed	Store of NoSql-Document	C++, Go, JavaScript, Python
Oracle DB			Sql	C, Assembly language, C++
SQL Servers			Sqk	C++, C
Couchbase	Open Source	Multi-Model of Distributed	Store of NoSql-Document	C++, Erlang, C, Go
Neo4j	Open Source		Family of NoSql-Graph	Java
CouchDB	Open Source	Multi-Model of Distributed	Store of NoSql-Document	Erlang, JavaScript, C, C++
Rethink	Open Source	Multi-Model of Distributed	Store of database NoSql-Document	C++, JavaScript, Python, Bash, Java
Cassandra	Open Source	Multi-Model of Distributed	Based of NoSql-Document	Java
RavenDB	Open Source	Open Source	Store of NoSql-Document	C#

4.1. Gaps of Research

Significant availability and scalability requirements necessitate complex distribution systems. Sharding and partitioning occur at the application, caching, and back-end storage tiers. The software must handle data replicas and inconsistencies from concurrent updates. Each type of NoSQL database has different shortcomings regarding consistency, durability, performance, and scalability. To choose the right database, architects must examine the features of each candidate database. This includes analyzing gaps in each type of NoSQL database [29].

Gaps among databases of NoSql.

- If your applications primarily involve storing and retrieving data items identifiable by a key, use key-value stores. However, querying by an attribute other than the key can cause crashes, and individual fields cannot be modified or retrieved.
- For applications needing detailed management of record selection and specific fields, document databases are better. They offer more query flexibility than key-value stores, allowing retrieval using criteria besides the primary key.
- Column-family stores are suitable for applications storing data with numerous fields but requiring access to only a portion. These stores handle extensive datasets well.
- Graph databases are ideal when entities and their connections are equally important.

Big data from sensor networks or social media often overwhelms traditional relational databases due to their inability to handle massive, unstructured data sets. NoSql databases, lacking a centralized schema, offer greater scalability, availability, fault tolerance, and performance for big data [30,31].

The study concludes that NoSql is not a complete replacement for relational databases but is well-suited for heterogeneous big data. Further research is needed in NoSql performance, scalability, simplicity, and schema design. Horizontal scalability makes NoSql ideal for handling large, diverse data, whereas SQL databases scale vertically. Research also focuses on integrating non-relational and relational database features and developing frameworks for data migration from SQL to NoSQL, which has significant business implications.

4.2. Forecasting and incidents involving DBMSs in comparison to a specific DBMS

We generated a set of (x, y) data pairs and constructed 301 combinations for database names. We preprocessed the data by converting x and y from strings to numbers using a label encoder. We then trained a Gaussian Naïve Bayes model on the encoded data. This model identified distinct database names and calculated the probability for each class. The results were represented in an n by n table, where n is the number of distinct databases. The model was trained on this data, but it correctly identified MongoDB for every database name except its own [32, 33].

5. Conclusions

The study concludes that switching from relational databases to NoSql databases isn't always necessary. Organizations should choose the database management system (DBMS) that best fits their needs. SQL databases are ideal for maintaining data consistency and uniformity, while NoSql databases excel in handling large amounts of unstructured data and providing accessibility. Relational databases might be better for consolidating smaller datasets, whereas NoSql databases are suited for big data analytics and applications producing large quantities of data due to their distributed and scalable architecture.

Relational databases perform better with geospatial data, though they are slower than NoSql databases in processing extensive geographical information. Despite the advantages of NoSql databases, many enterprises may still hesitate to replace traditional RDBMSs entirely. NoSql databases, being a newer addition to the database industry, lack universally recognized standards and strict ACID properties, but they offer a flexible and dynamic schema, facilitating quick development. However, the diverse models and interfaces used by different NoSql databases and cloud service providers (CSPs) pose challenges for data portability and interoperability. A standardized cloud-based solution is needed to address these issues.

Future research could focus on denormalized techniques for SQL RDBMS and compare performance metrics like data insertion, updating, and retrieval between MongoDB and other systems. NoSql databases should also consider more effective parallel geographic strategies to serve large user groups better. Additionally, the development of big data techniques can benefit applications in deep learning, such as object detection, signal classification, and computer vision.

References

1. Roy, A. M., & Bhaduri, J. (2021). A deep learning enabled multi-class plant disease detection model based on computer vision. AI, 2(3), 413–428. https://doi.org/10.3390/ai2030025
2. Roy, A. M. (2022). An efficient multi-scale CNN model with intrinsic feature integration for motor imagery EEG subject classification in brain-machine interfaces. Biomedical Signal Processing and Control, 74, 103496. https://doi.org/10.1016/j.bspc.2021.103496
3. Singh, A., Raj, K., Kumar, T., Verma, S., & Roy, A. M. (2023). Deep learning-based cost-effective and responsive robot for autism treatment. Drones, 7(2), 81. https://doi.org/10.3390/drones7020081
4. Siddiqa, A., Hashem, I. A. T., Yaqoob, I., Marjani, M., Shamshirband, S., Gani, A., & Nasaruddin, F. (2016). A survey of big data management: Taxonomy and state-of-the-art. Journal of Network and Computer Applications, 71, 151–166. https://doi.org/10.1016/j.jnca.2016.09.001
5. Kong, X., Shi, Y., Yu, S., Liu, J., & Xia, F. (2019). Academic social networks: Modeling, analysis, mining and applications. Journal of Network and Computer Applications, 132, 86–103. https://doi.org/10.1016/j.jnca.2019.01.003
6. Ordonez, C. (2009). Optimization of linear recursive queries in SQL. IEEE Transactions on Knowledge and Data Engineering, 22(2), 264–277. https://doi.org/10.1109/TKDE.2009.80
7. Strozzi, C. (2007–2010). NoSQL—A relational database management system. Retrieved from http://www.strozzi.it/cgi-bin/CSA/tw7/I/en_US/nosql/Home%20Page
8. Brewer, E. A. (2000). Towards robust distributed systems. In Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC), Volume 7. Foster City, CA: Inktomi.
9. Kumari, A., Tanwar, S., Tyagi, S., Kumar, N., Parizi, R. M., & Choo, K.-K. R. (2019). Fog data analytics: A taxonomy and process model. Journal of Network and Computer Applications, 128, 90–104. https://doi.org/10.1016/j.jnca.2018.11.002
10. Alsolai, H., & Roper, M. (2020). A systematic literature review of machine learning techniques for software maintainability prediction. Information and Software Technology, 119, 106214. https://doi.org/10.1016/j.infsof.2019.106214
11. Rodrigues, M., Santos, M. Y., & Bernardino, J. (2019). Big data processing tools: An experimental performance evaluation. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 9(2), e1297. https://doi.org/10.1002/widm.1297
12. Băzăr, C., & Iosif, C. S. (2014). The transition from RDBMS to NoSQL: A comparative analysis of three popular non-relational solutions: Cassandra, MongoDB, and Couchbase. Database Systems Journal, 5(2), 49–59.
13. Mukherjee, S. (2019). The battle between NoSQL databases and RDBMS. Williamsburg, KY: University of the Cumberlands.
14. Chopade, R., & Pachghare, V. K. (2019). Ten years of critical review on database forensics research. Digital Investigation, 29, 180–197. https://doi.org/10.1016/j.diin.2019.04.009
15. Kitchenham, B., & Brereton, P. (2013). A systematic review of systematic review process research in software engineering. Information and Software Technology, 55(12), 2049–2075. https://doi.org/10.1016/j.infsof.2013.07.010
16. Dean, J., & Ghemawat, S. (2008). MapReduce: Simplified data processing on large clusters. Communications of the ACM, 51(1), 107–113. https://doi.org/10.1145/1327452.1327492
17. Pokorný, J. (2015). Database technologies in the world of big data. In Proceedings of the 16th International Conference on Computer Systems and Technologies (pp. 1–12). Dublin, Ireland.
18. Baralis, E., Dalla Valle, A., Garza, P., Rossi, C., & Scullino, F. (2017). SQL versus NoSQL databases for geospatial applications. In Proceedings of the 2017 IEEE International Conference on Big Data (Big Data) (pp. 3388–3397). Boston, MA, USA.
19. Okman, L., Gal-Oz, N., Gonen, Y., Gudes, E., & Abramov, J. (2011). Security issues in NoSQL databases. In Proceedings of the 2011 IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications (pp. 541–547). Changsha, China.
20. Roy-Hubara, N., & Sturm, A. (2018). Exploring the design needs for the new database era. In Enterprise, Business-Process and Information Systems Modeling (pp. 276–290). Cham, Switzerland: Springer.
21. Mansouri, Y., Prokhorenko, V., & Babar, M. A. (2020). An automated implementation of hybrid cloud for performance evaluation of distributed databases. Journal of Network and Computer Applications, 167, 102740. https://doi.org/10.1016/j.jnca.2020.102740

22. Ravi, K., Khandelwal, Y., Krishna, B. S., & Ravi, V. (2018). Analytics in/for cloud-an interdependence: A review. Journal of Network and Computer Applications, 102, 17–37. https://doi.org/10.1016/j.jnca.2017.11.006
23. Wiese, L., Waage, T., & Brenner, M. (2019). CloudDBGuard: A framework for encrypted data storage in NoSQL wide column stores. Data & Knowledge Engineering, 126, 101732. https://doi.org/10.1016/j.datak.2019.101732
24. Ribas, M., Furtado, C., de Souza, J. N., Barroso, G. C., Moura, A., Lima, A. S., & Sousa, F. R. (2015). A Petri net-based decision-making framework for assessing cloud services adoption: The use of spot instances for cost reduction. Journal of Network and Computer Applications, 57, 102–118. https://doi.org/10.1016/j.jnca.2015.08.004
25. Yoon, J., Jeong, D., Kang, C.-H., & Lee, S. (2016). Forensic investigation framework for the document store NoSQL DBMS: MongoDB as a case study. Digital Investigation, 17, 53–65. https://doi.org/10.1016/j.diin.2016.02.002
26. Shirazi, M. N., Kuan, H. C., & Dolatabadi, H. (2012). Design patterns to enable data portability between clouds' databases. In Proceedings of the 2012 12th International Conference on Computational Science and Its Applications (pp. 117–120). Salvador, Bahia, Brazil.
27. Dyba, T., Kitchenham, B. A., & Jorgensen, M. (2005). Evidence-based software engineering for practitioners. IEEE Software, 22(1), 58–65. https://doi.org/10.1109/MS.2005.7
28. Yang, H., Dasdan, A., Hsiao, R.-L., & Parker, D. S. (2007). Map-reduce-merge: Simplified relational data processing on large clusters. In Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data (pp. 1029–1040). Portland, OR, USA. https://doi.org/10.1145/1247480.1247602
29. Zeng, N., Zhang, G.-Q., Li, X., & Cui, L. (2017). Evaluation of relational and NoSQL approaches for patient cohort identification from heterogeneous data sources. In Proceedings of the 2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM) (pp. 1135–1140). Kansas City, MO, USA. https://doi.org/10.1109/BIBM.2017.8217826
30. Tear, A. (2014). SQL or NoSQL? In Contrasting approaches to the storage, manipulation and analysis of spatio-temporal online social network data. In Proceedings of the International Conference on Computational Science and Its Applications (pp. 221–236). Guimaraes, Portugal. https://doi.org/10.1007/978-3-319-09150-1_17
31. Fraczek, K., & Plechawska-Wojcik, M. (2017). Comparative analysis of relational and non-relational databases in the context of performance in web applications. In Proceedings of the International Conference: Beyond Databases, Architectures and Structures (pp. 153–164). Ustroń, Poland. https://doi.org/10.1007/978-3-319-52465-8_15
32. Hricov, R., Šenk, A., Kroha, P., & Valenta, M. (2017). Evaluation of XPath queries over XML documents using SparkSQL framework. In Proceedings of the International Conference: Beyond Databases, Architectures and Structures (pp. 28–41). Ustroń, Poland. https://doi.org/10.1007/978-3-319-52465-8_3
33. Płuciennik, E., & Zgorzałek, K. (2017). The multi-model databases–A review. In Proceedings of the International Conference: Beyond Databases, Architectures and Structures (pp. 141–152). Ustroń, Poland. https://doi.org/10.1007/978-3-319-52465-8_14.