IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 6.252
Volume 2, Issue 1, April 2022

Efficient Data Modeling and Storage Solutions with SQL and NoSQL Databases in Web Applications

Vasudhar Sai Thokala
Independent Researcher
vasudharsai7@gmail.com

Abstract: *In today's web application landscape, efficient data modelling and storage solutions are essential for ensuring scalability, performance, and user satisfaction. SQL and NoSQL databases have emerged as prominent technologies, each providing unique benefits for managing structured and unstructured data, respectively. SQL databases offer a stable, ACID-compliant environment suitable for applications requiring transactional integrity and complex querying, whereas NoSQL databases provide a flexible, schema-less architecture designed for horizontal scalability, high availability, and handling large volumes of diverse data. With the rise of data-centric applications, a hybrid approach that combines SQL's strong consistency and NoSQL's adaptability has become increasingly common. This paper examines the architectural distinctions, performance trade-offs, and integration strategies of SQL and NoSQL databases in web applications, providing a comprehensive analysis of their roles, benefits, and limitations. By understanding these aspects, developers and data architects can make informed decisions in selecting or combining SQL and NoSQL databases to meet the dynamic requirements of modern digital platforms.*

Keywords: Data modelling, SQL and NoSQL databases, ACID Compliance, Web Application Architecture

I. INTRODUCTION

In the modern landscape of web applications, efficient data modelling and storage solutions play a crucial role in determining the scalability, performance, and user satisfaction of digital platforms. The continuous evolution of data-centric applications necessitates robust and flexible databases capable of handling large, diverse data sets with varying levels of complexity and structure. SQL and NoSQL databases have emerged as prominent solutions, each offering unique advantages and design principles[1]. SQL databases, known for their strong consistency and structured schema, have long been the backbone of relational data storage. In contrast, NoSQL databases present an adaptable approach to handling unstructured data, emphasising scalability and performance for a wide array of web applications. Understanding the strengths and limitations of both SQL and NoSQL systems is critical for developers and data architects when designing storage solutions that meet the demands of today's applications.

SQL Database stores data according to the relational data model. Data is organised in a tabular fashion using rows and columns in this paradigm. There is a way to connect related tables. Database management systems such as MySQL, Oracle, SQL Server, and many more are accessible. SQL has been around for a while; therefore, most of the problems have been fixed. Incorporating security elements such as authentication, data confidentiality, and integrity, SQL [2]. Tables in SQL are connected to one another. The user must use JOIN statements, which generate Views, in order to get data from many tables. Doing this takes a lot of time.

The non-relational data paradigm is followed by NoSQL. Data in many formats, including documents and graphs, may be stored schema-free using a non-relational paradigm. Not only is NoSQL capable of storing data from the Internet of Things, but it also supports schema-less storage, horizontal scalability, and unstructured data[3]. The distributed design, quick access, and high scalability of NoSQL databases are the main reasons for their rising popularity. Not relational database systems are widely used; examples include MongoDB, Redis, CouchDB, and Hbase. When compared to the conventional ways used by relational database systems, the way data is stored in NoSQL databases is very different[4].

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 6.252
Volume 2, Issue 1, April 2022

This data type is designed for schema-less structures. The distributed nature of NoSQL databases makes them more scalable and available [5].

The complementary strengths of SQL and NoSQL databases have led to a hybrid approach in many modern web applications. Instead of viewing these technologies as mutually exclusive, data architects and developers increasingly explore their combined use to achieve optimal data management. SQL databases[6] are often employed for critical, structured data with well-defined relationships, while NoSQL databases manage more fluid, unstructured, or semi-structured data[7].

A. Structure of the paper

The paper is structured as follows: Section II reviews SQL database systems, their structure, and optimisation. Section III covers core concepts and types of NoSQL databases. Section IV explores data modelling and storage solutions for both SQL and NoSQL. Section V compares their roles in web applications. Section VI presents the previous study. Finally, Section VII concludes with key findings and future directions.

II. FUNDAMENTAL OVERVIEW OF SQL DATABASES SYSTEM

SQL, or Structured Query Language, is a programming language created by IBM in the early 1970s to interact with relational databases, specifically for their System R database system. An SQL database uses SQL to manage and modify data, which is organised in a systematic way for simple access, maintenance, and updating. An SQL database is structured like a table, with rows representing records and columns representing attributes [8]. These tables are typically linked by primary keys (unique identifiers) and foreign keys, establishing relationships between data across tables—a feature that gives SQL databases the term "relational." SQL databases adhere to ACID properties (Atomicity, Consistency, Isolation, Durability), ensuring secure and reliable transactions. Using standardised commands, SQL allows users to perform complex queries, data manipulations[9], and aggregations, making it a core technology for applications across industries. Today, SQL-based systems like MySQL, PostgreSQL, Oracle, and Microsoft SQL Server are widely used due to their robust data management capabilities, flexible querying, and compatibility with numerous applications

A. Structure of SQL Database

The SQL framework is used to construct both commercial and open-source relational database management systems. Many organisations make use of these databases. Among the most popular are PostgreSQL, Integrated Business Machines DB2, SAP HANA, Oracle Database, MySQL (owned by Oracle), MS SQL Server, and SAP Adaptive Server[10]. A large number of DBMSs are SQL-centric, meaning they use SQL as their primary language for all related functions and programming. Figure 1 depicts the structure of SQL databases as shown below:

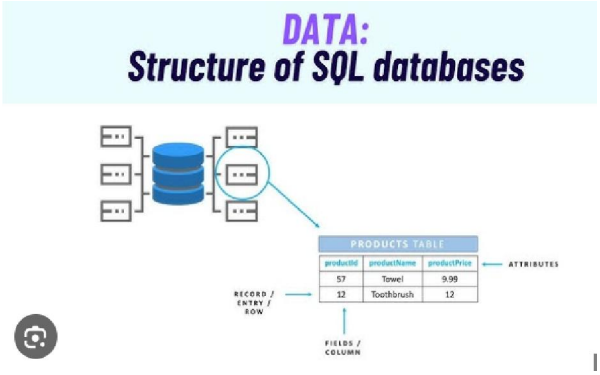

Figure 1: Structure of SQL databases.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 6.252
Volume 2, Issue 1, April 2022

B. Performance and Optimization in Relational Databases

- The last forty years have seen remarkable performance gains from relational databases. Structured data is their speciality, and they adhere to the ACID (Availability, Consistency, Isolation, and Durability) property in their design.
- SQL query optimisation and performance for business, production, and parallel databases, as well as large data, has been an increasingly hot topic in recent years [11].
- Ineffective and poorly optimised queries might use up server and system resources, cause database locking, and result in data loss issues.
- Choosing an efficient query execution plan that uses fewer system resources and costs the least is what we mean when we talk about query optimisation [12].

C. Common SQL databases (e.g., MySQL, PostgreSQL)

1. MySQL

Ever since the site began to gain notoriety in 2012 for its database, MySQL has been the most popular open-source RDBMS. Twitter, Facebook, Netflix, and Spotify are just a few of the many popular websites and apps that rely on this component-rich item [13][14]. The abundance of MySQL-related resources online, together with the project's extensive documentation and large network of designers, make getting started with MySQL a very simple process. Although it was designed to comply with standard SQL, MySQL prioritised speed and unchanging quality [15].

2. PostgreSQL

PostgreSQL is an object-relational database management system (ORDBMS) that originated from the POSTGRES project developed at UC Berkeley in the 1980s. In 1994, SQL support was added, and it was re-released as Postgres95 in 1996, eventually becoming PostgreSQL[16][17]. It adheres to much of the SQL standard and includes features like complex queries, foreign keys, triggers, views, and transactional integrity. PostgreSQL supports multiple programming interfaces (e.g., C/C++, Java, Python) and international character sets, including Unicode. It is ACID-compliant and compatible with major operating systems such as Linux, UNIX, and Windows[18][19].

D. Basic steps for executing SQL Queries

Executing SQL queries involves several fundamental steps to connect to a database, perform the query, and retrieve the results. Here's a general outline of these steps:

1. BASIC SQL — Select, Where, Sorting

Syntax of the SQL SELECT command, including the following clauses: AS, LIKE, DISTINCT, ORDER BY, AND, OR, NOT, and WHERE. The SQL queries that follow are derived from Table I, a dataset that represents an organisation's employees.

- Every column for every employee is shown in the following SQL query. SELECT * FROM Staff;
- Any employee with the name Mary may see their earnings and department in the following SQL query. SELECT salary, dept FROM Staff WHERE name='Mary';
- Any employee whose name includes the letter "e" will have their unique department shown using the following SQL query. SELECT distinct dept from Staff WHERE name like '%e%';
- The following SQL query lists all female employees with salaries more than or equal to 40,000.SELECT name FROM Staff WHERE sex='F' and salary>= 40000;
- The following Table I SQL query lists all IT department employees by name and pay, with the results ordered from highest to lowest[20]. SELECT name, salary FROM Staff WHERE dept='IT' ORDER BY salary DESC;

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 6.252
Volume 2, Issue 1, April 2022

Table 1: Staff table

sid	name	dept	sex	salary	ext
101	Mary	IT	F	45000	121
102	Smith	HR	M	37000	NULL
103	Tony	IT	M	38000	115
104	Sarah	PJ	F	35000	311
105	Mark	IT	M	45000	153
106	Alice	HR	F	39000	433
107	Ben	IT	F	NULL	NULL

The basic steps are as follows:

- Use a valid username and password to establish a connection to the database on a server.
- Send the query string.
- Gather the outcomes and store them in an array in PHP.
- Check the number of rows in the array.
- Get every record in the array of results.
- Break the connection and release the result array.

Figure 2 shows an example program that followed this technique. Using the credentials "tester" and "1234", the application establishes a connection to the "test" database on the "localhost" server

```php
<?php
$value=$_GET['keyword'];
$server = "localhost";
$user = "tester";
$pass = "1234";
$db = "test";

$conn = mysqli_connect($server,$user,$pass,$db);
if (!$conn)
  die("Connection failed: " . mysqli_connect_error());

$sql = "SELECT name, salary FROM Staff where dept='$value'";
$result = mysqli_query($conn, $sql);

if (mysqli_num_rows($result) > 0) {
  echo "Records found for dept: $value\n";
  echo "<TABLE border=1>\n";
  echo "<TR><TH>Name<TH>Salary\n";
  while($row = mysqli_fetch_assoc($result)) {
    echo "<TR><TD>".$row['name']."<TD>".$row['salary']."\n";
  }
  echo "</TABLE>\n";
} else
  echo "No record is found for dept: $value\n";

mysqli_free_result($result);
mysqli_close($conn);
?>
```

Figure 2: Establishment of a connection to the database on a server.

III. CORE CONCEPTS IN NOSQL DATABASES

Traditional relational databases rely on table relationships to store and retrieve model data; however, a NoSQL database (formerly known as "non-SQL" or "non-relational") offers an alternative. Data storage and recovery should be made simple regardless of its structure or content, and that is the main goal of the NoSQL movement [21]. Which is designed for scalability, speed, and high availability. NoSQL databases accommodate diverse data types, such as unstructured or semi-structured data, making them ideal for handling large volumes of information and complex, hierarchical relationships[22]. The NoSQL model includes various types: document-based databases, such as MongoDB, that store data in flexible document formats; key-value stores, like Redis, for rapid data retrieval by key; column-family stores, such as Cassandra, that optimise read and write operations for analytical workloads; and graph databases, like Neo4j, that excel at modelling complex relationships in social networks and recommendation systems. NoSQL's flexibility allows developers to easily adapt to changing application requirements without significant data restructuring, making it particularly useful for Big Data and real-time web applications where relational databases may fall short[23].

NoSQL database types

Figure 3 shows the four main kinds of NoSQL databases that have evolved: graph databases, document databases, key-value databases, and wide-column stores. Additionally, multi-model databases are booming in popularity these days.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 6.252
Volume 2, Issue 1, April 2022

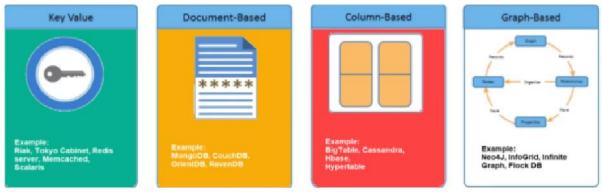

Figure 3: Types of NoSQL Databases.

1. Document-Oriented Databases

Data stored in a document-oriented database resembles JSON (JavaScript Object Notation) objects. There are two fields and two values in every document. The values may usually take several forms, such as texts, integers, Booleans, arrays, or even actual objects [24][25]. Document databases provide a versatile data paradigm that works well with semi-structured and often unstructured data collections. The ability to create nested structures is another great feature that makes it easier to express data with complicated connections or hierarchies.

2. Key-Value Pairs store

The simplest kind of database is the key-value store, in which each object has both a key and a value. As shown in Table II, every key has its own distinct value. Their tendency to hold items in memory makes them ideal for caching and session management, where they provide great read/write speed.[26][27].

Table II: Example of how key pairs are stored in a database.

Key	Value
Book Title	Business Intelligence and Analytics: Systems for De-cision Support
Author (set)	Ramesh Sharda
	Dursun Delen
	Efraim Turban
Publication Date	2015
Edition	10th
Publisher	Pearson
...	...

3. Column family stores

The information will be stored and displayed in these databases using column families as rows. The key of each row is associated with a certain number of columns in these rows. The set of interconnected data that may be accessed individually. In RDBMS, each column family is analogous to a table's row container [28][29].

4. Graph Database stores

Graph databases specialise in handling highly connected data using three core elements: nodes, edges, and properties. Nodes store data about entities, while edges define relationships, each with a type, direction, and properties. Relationships in graph databases are directional, determining the traversal path from one node to another and may link to the same or different nodes. Properties, attached to both nodes and relationships, provide additional context[30]. Unlike relational databases, graph databases are optimised for querying complex relationships, as illustrated in Figure 4, where each node represents an employee connected by a “knows” relationship, with “Duration” as a property.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 6.252
Volume 2, Issue 1, April 2022

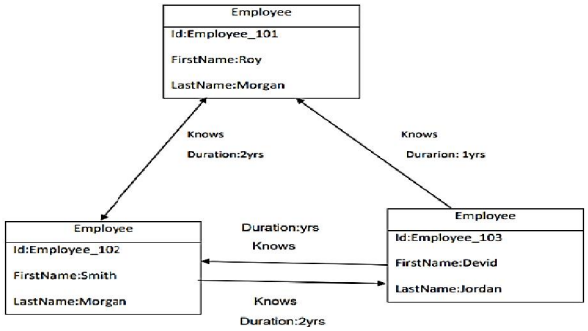

Figure 4: 'Employee' database modelled as a graph database.

B. Benefits in terms of flexibility, scalability, and performance

In this section, provides the benefits of NoSQL databases in the form of flexibility, scalability and performance.

- Various online applications and corporate systems rely on NoSQL databases to meet their storage and retrieval needs quickly and efficiently.
- They provide a range of extra characteristics that set them apart from traditional RDBMS, such as the capacity to easily scale horizontally and the fact that they do not need the definition of a strict database structure [31].
- The horizontal scalability of NoSQL databases makes it simpler to use clusters with data dispersed across multiple nodes, and the ability to store structured, unstructured, and semi-structured data from many sources is a major bonus.
- Distributing data across several servers is a representation of shared-nothing horizontal scalability, which is a feature of NoSQL databases.
- Data scalability improves performance, which in turn speeds up query execution and enables a greater volume of attended client requests [32].
- Data is further distributed as the number of servers increases, and requests are processed in parallel.
- A speedup occurs when the amount of time it takes to execute a request is reduced due to the parallelisation of query processing.

C. Common NoSQL databases

NoSQL databases in order to compare them with Microsoft SQL and with each other. The NoSQL databases we have chosen for comparison are briefly described in the remaining portion of this section.

- **MongoDB** is a C++-based document-oriented database. BSON serialisation is used to store the items. Flexible schema storage is made possible by the fact that the objects do not have to have the same fields or structure, and the shared fields do not have to be of the same type. Auto-sharding, which divides data sets and distributes them across available servers, is supported by MongoDB[33]. The outcome is a load that is balanced dynamically.
- **Hypertable** is a free and open-source NoSQL database that makes use of Google Big Table. The database is crafted in C++ and belongs to the column-family. Hypertable, like MongoDB, has auto-sharding capabilities.
- **Apache CouchDB** is an Erlang-based document-oriented database. The JSON format is used for storing the items. A protocol known as RESTful HTTP allows access to the database. "Database that completely embraces the web"3 is the assertion made in regard to CouchDB.
- **Apache Cassandra** was created by Facebook and is a database that uses column families. Data replication to improve fault tolerance and decentralisation to decrease failures are two of its main characteristics [34].
- **RavenDB and Couchbase** examine two other document-oriented databases. These databases, like MongoDB and CouchDB, employ JSON as their serialised object format and provide flexible schema storage. Data sharding and replication are added services they provide [35].

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 6.252
Volume 2, Issue 1, April 2022

IV. DATA MODELING AND STORAGE SOLUTIONS FOR SQL AND NOSQL

Data modelling and storage are critical components of any database management system, determining how data is structured, stored, accessed, and maintained. SQL and NoSQL databases each use different approaches to data modeling and storage to suit different types of applications and use cases and schema design principles are as described below.

A. SQL (Relational Databases)

Originally, papers were used to hold data. However, utilising files to obtain the information became more difficult as its volume expanded. It was an ineffective and sluggish procedure [36]. Maintaining records and storing the information became more challenging as the amount of data increased. With their focus on storage, hierarchical and network databases failed to provide a standard method for retrieving data[37]. The need to manage data and the demand for a standard method of retrieving data gave rise to SQL.

B. NoSQL (Non-relational Databases):

'NoSQL' databases, which stand for non-relational, cloud, or alternative database management models, are quickly becoming the popular choice. The fundamental goals of this approach are improved accessibility control, easier 'horizontal' scalability to machine clusters—a problem with relational databases—and a simpler architecture. Some operations in NoSQL are faster than in relational databases because the default information structures utilised by NoSQL databases (such as key-value, graph, or document) are somewhat different. A further argument in favour of NoSQL databases is the fact that their data structures are ' more flexible' than those of relational databases. More and more, NoSQL databases are being employed in Big Data and real-time web applications[38].

C. NoSQL Data Modeling techniques:

Although NoSQL databases may not have a rigid schema design, it is nevertheless imperative that you adhere to NoSQL-specific data modelling best practices. These could differ among technologies, but in general, the following are some conceptual approaches to NoSQL data modelling:

1. Denormalization

Denormalisation is a popular method for simplifying data that entails replicating it into many tables or formats. All of the data you need to query may be conveniently grouped in one location by using denormalisation. Unfortunately, this results in a significant increase in the amount of data for a variety of factors [39].

2. Aggregates

This opens the door for users to modify their own systems and build layered entities with intricate frameworks. By reducing the number of one-to-one linkages, aggregation minimises connections. When it comes to NoSQL databases, this soft schema method is present in almost every model. For instance, since these data models impose no limitations on the subject, values in graph and key-value store databases may be in any format [40].

3. Application Side Joins.

NoSQL often does not allow joins since its databases are question-oriented and joins are done at design time. This is carried out during query execution, as opposed to relational databases. It is occasionally inevitable and, of course, often involves a performance penalty.

V. SQL AND NOSQL DATABASES IN WEB APPLICATIONS

SQL and NoSQL databases play critical roles in modern web applications, each offering distinct strengths that can impact application performance, scalability, and data management capabilities. Considerations such data structure, scalability demands, query complexity, and performance should be considered while deciding between SQL and NoSQL databases, or combining the two in a hybrid method.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 6.252
Volume 2, Issue 1, April 2022

A online database application need to be one of the primary foci of projects in any database course, given the increasing importance of web-based applications across numerous sectors [41].

In web application development, numerous SQL and NoSQL databases are commonly used. These include:

- **Oracle:** The most popular commercial RDBMS, Oracle, uses assembly languages such as Java, C++, and C.
- **MySQL:** MySQL is still one of the most popular database options, and for good reason: it is mature, stable, and robust—three qualities that are highly prized in the creation of online applications.
- **PostgreSQL:** PostgreSQL has a long and storied history, having been originally called POSTGRES. Michael, who made substantial contributions to the creation of PostgreSQL, was honoured with the prestigious Turing Award.
- **MongoDB:** The first document database management system, MongoDB, was released in 2009. Problems with utilising object-oriented programming languages to load and retrieve data in RDBMS led to the creation of MongoDB.
- **Cassandra:** Applications demanding strong database solutions often use Cassandra, an open-core distributed wide-column store that was launched in 2008 [42].
- **SQLite:** A robust open-source SQL database, SQLite has been providing customers with an integrated RDBMS since its introduction in the year 2000.

A. NoSQL Database in Web Application

Businesses often switch to NoSQL databases when conventional SQL systems don't meet their needs for scalability, flexibility, or performance. NoSQL databases may store data in a variety of ways, including column-oriented, document-oriented, graph-based, and key-value stores, and can feature dynamic schemas for unstructured data, unlike SQL databases.

- ***Scalability:*** Adding servers (sharding) instead of updating hardware allows NoSQL databases to accommodate increased traffic, making them horizontally scalable. This makes them ideal for large or evolving datasets.
- ***Flexibility:*** Different data formats may be stored in the same table in NoSQL databases since they don't need a predetermined schema. However, most NoSQL systems don’t support SQL-like query languages, offering flexibility at the cost of complex querying[43].
- ***Performance:*** NoSQL databases, such as MongoDB and CouchDB, are designed to handle high data volumes and provide fast data access, essential for web applications needing quick response times. Performance is influenced by factors like data type, volume, and processing needs, which can impact the overall speed and efficiency of a web application.

B. SQL Databases in Web Applications

SQL databases, or relational databases, are widely used in web applications requiring structured data, transactional integrity, and complex querying capabilities. They organise data in tables with predefined schemas, enforcing relationships between entities. SQL databases are popular in applications where data consistency, integrity, and complex querying are essential.

- ***Data Consistency and Integrity:*** SQL databases are ACID-compliant (Atomicity, Consistency, Isolation, Durability), ensuring data reliability and consistency. This makes them ideal for applications needing strict accuracy, such as financial, e-commerce, and inventory management systems[44].
- ***Complex Querying and Reporting:*** SQL databases support complex queries with joins, filtering, aggregations, and subqueries. This is useful for applications that require data analysis, reporting, and handling of complex relationships between data.
- ***Standardized Language (SQL):*** SQL is a standard language for data manipulation that SQL databases utilise. It simplifies data management, updates, and queries. This standardisation is advantageous for developers as it is widely understood and supported[45].

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 6.252
Volume 2, Issue 1, April 2022

- ***Scalability:*** SQL databases are often vertically scalable, which means that by boosting a single server's hardware capacity (CPU, RAM), they can manage larger loads. Some modern SQL databases also support limited horizontal scaling via sharding and replication.

VI. LITERATURE REVIEW

This section reviews studies on modelling and storage solutions, comparing the structured approach of SQL databases with the flexible, schema-less design of NoSQL systems. It highlights performance, scalability, and use-case suitability to guide database selection.

In, Laksono, (2018), compares and contrasts the performance of SQL and NoSQL (MongoDB) databases when it comes to processing geographic large data quantitatively. To evaluate how well MongoDB and PostGIS handle massive amounts of synthetic geographic data, we built an angular-framework web app that runs on Node.js. Both databases were tested for their capacity to store and load geographic data, but with differing numbers of points. In order to carry out the test, the results of each XHR (XML HTTP Request) from both databases were compared. The results demonstrated that when it comes to loading large amounts of geographical data, NoSQL databases, such as MongoDB, outperform conventional SQL databases when using PostGIS [46].

In, Vonitsanos et al. (2018) provide a method for modelling software architecture and data that makes use of NoSQL databases to handle semi-structured and heterogeneous data. Using data mining methods, we integrated Apache Spark and Apache Cassandra to build a strong analytics framework. Then, we presented a model that could predict the link between the number of tourists and the number of nights they stayed in Greece. This model makes use of a dataset that was built using data from Eurostat and the Hellenic Statistical Authority. After fitting the present dataset to the suggested data model, the assessment revealed that it accurately predicted tourist behaviours[47].

In, Flores et al. (2018) used the experimental database system given by the Función Judicial del Ecuador to investigate response times on relational and non-relational database types. The data is prepared for loading into a NoSQL allocation, and a document-oriented data model is used to achieve benchmarks for performance. Our goal in conducting these tests is to identify the optimal database by analysing the time it takes for data queries to provide results from performance evaluations. Beyond that, we compare pre- and post-RDBMS performance and find that, in the first test, MongoDB queries significantly outperform SQL Server queries, but in subsequent tests, the times are comparable[48].

In, Gupta et al. (2019), Integrated encryption methods are used to protect online application databases from SQL Injection Attacks. Interestingly, if an intruder manages to breach the database's defences, they may unleash havoc while stealing sensitive information. To counteract these kinds of attacks, a hybrid strategy is being considered. One option is to use Advanced Encryption Standard (AES) during the login process to secure databases from unauthorised access. Another option is to use Elliptical Curve Cryptography (ECC) to encrypt databases, making them unreadable without the key. The method to avoid SQL Injection Attacks is shown in this study article[49].

In, S. Reetishwaree (2020) study examines the performance of three database formats in an IoT setting: relational on-disk, relational in-memory, and noSQL on-disk/in-memory. The operations that are compared include insert, select, update, and delete, as well as indexes, stored procedures, and triggers. Sensor data, including temperature/humidity, humidity, and water level, has allowed for the real-time monitoring of a plant environment. Data was saved in many databases, and the time it took to complete different tasks was documented. The results of experiments with the water level parameter show that no one database performs optimally for every data process [50].

In, Namdeo and Suman (2021), offered a methodology for migrating databases that can parallelly transfer both historical and real-time data. The suggested paradigm is used in Java for the purpose of connecting MySQL (RDBMS) to MongoDB, a document database. The prototype simultaneously migrates active data and database snapshots, which are instances of the database. In conclusion, as compared to other methodologies, our experimental study of snapshot and live data transfer demonstrates superior performance [51].

Table III provide Efficient Data Modeling and Storage Solutions with SQL and NoSQL Databases in Web Applications

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 6.252
Volume 2, Issue 1, April 2022

Table III: Efficient Data Modeling and Storage Solutions with SQL and NoSQL Databases in Web Applications

Reference	Focus Area	Databases Compared	Methodology	Findings	Limitations
Laksono (2018)	Geospatial big data handling performance	MongoDB (NoSQL), PostGIS (SQL)	Developed NodeJS-based web app to compare XML HTTP Request (XHR) results for geospatial data storage and loading.	MongoDB outperforms PostGIS in loading large geospatial datasets.	Limited to simulated geospatial data; real-world variability not tested.
Vonitsanos et al. (2018)	Data modelling and analytics for tourist behaviour prediction	Apache Cassandra (NoSQL)	Integrated Apache Spark with Cassandra to predict relationships between tourist arrivals and overnight stays in Greece	The model predicts tourist behaviour with high accuracy using heterogeneous datasets from statistical authorities.	Focused only on specific tourism datasets in Greece; generalizability may be limited.
Flores et al. (2018)	Response time comparison between relational and non-relational databases	SQL Server, MongoDB (NoSQL)	Experimental database migration to a document-oriented model, performance tests based on query response times	MongoDB performs better than SQL Server in initial tests but shows similar performance in subsequent queries.	Focuses primarily on response times without evaluating scalability or operational complexity.
Gupta et al. (2019)	Prevention of SQL Injection Attacks	SQL databases	Integrated encryption using AES and ECC to secure web application databases	Proposed method effectively prevents SQL Injection by encrypting database access and storage.	Concentrated on security, not performance or storage optimisation.
S. Reetishware (2020)	Comparison of database types for IoT real-time monitoring	Relational On-disk, In-memory, NoSQL On-disk/In-memory	Monitored IoT sensor data (Temperature, Humidity, Water level) and evaluated Insert, Select, Update, Delete operations	No single database consistently outperforms others; performance varies by operation and workload.	Limited to IoT sensor environments; results may not extend to other use cases.
Namdeo and Suman (2021)	Real-time and historical data migration	MySQL (SQL), MongoDB (NoSQL)	Developed a Java-based migration model for snapshot and live database instances	Parallel migration model improves performance for both real-time and snapshot migrations compared to existing approaches.	Focuses solely on migration performance without exploring post-migration query efficiency.

VII. CONCLUSION AND FUTURE WORK

For various kinds of data storage, SQL and NoSQL databases have their own set of benefits: SQL databases provide robust consistency, complex querying, and structured data management, making them ideal for applications requiring high data integrity. NoSQL databases, in contrast, offer flexibility, schema-less structures, and horizontal scalability, excelling in handling unstructured or rapidly evolving data. Increasingly, modern web applications leverage a hybrid

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 6.252
Volume 2, Issue 1, April 2022

approach, combining SQL's data reliability with NoSQL's adaptability to balance performance, scalability, and versatility, optimising the data infrastructure for dynamic demands.

Future work could further refine SQL and NoSQL integration by exploring advanced methods for seamless data synchronisation, unified querying, and robust cross-database transaction management. Enhancing compatibility between SQL and NoSQL with AI-driven adaptive modelling and resource optimisation could dynamically balance performance in response to usage patterns. Additionally, developing hybrid databases that merge the ACID compliance of SQL with NoSQL's scalability and distribution capabilities would streamline data management, making it possible to support the increasingly complex requirements of real-time and data-intensive applications.

REFERENCES

[1] S. T. March, "Data Modeling: Entity-Relationship Data Model," in *Encyclopedia of Information Systems*, 2003. doi: 10.1016/b0-12-227240-4/00034-4.

[2] R. Goyal, "The Role Of Business Analysts In Information Management Projects," *Int. J. Core Eng. Manag.*, vol. 6, no. 9, pp. 76–86, 2020.

[3] V. V. Kumar and F. T. S. Chan, "A superiority search and optimisation algorithm to solve RFID and an environmental factor embedded closed loop logistics model," *Int. J. Prod. Res.*, 2011, doi: 10.1080/00207543.2010.503201.

[4] V. V. Kumar, F. W. Liou, S. N. Balakrishnan, and V. Kumar, "Economical impact of RFID implementation in remanufacturing: a Chaos-based Interactive Artificial Bee Colony approach," *J. Intell. Manuf.*, 2015, doi: 10.1007/s10845-013-0836-9.

[5] S. D. Dhasade, "NOSQL DATABASE," *Int. J. Innov. Res. Sci. Eng. Technol. (An ISO*, vol. 3297, no. 5, 2007, doi: 10.15680/IJIRSET.2016.0505350.

[6] V. S. Thokala, "A Comparative Study of Data Integrity and Redundancy in Distributed Databases for Web Applications," *IJRAR*, vol. 8, no. 4, pp. 383–389, 2021.

[7] V. V. Kumar, M. Tripathi, M. K. Pandey, and M. K. Tiwari, "Physical programming and conjoint analysis-based redundancy allocation in multistate systems: A Taguchi embedded algorithm selection and control (TAS&C) approach," *Proc. Inst. Mech. Eng. Part O J. Risk Reliab.*, 2009, doi: 10.1243/1748006XJRR210.

[8] V. V. Kumar, S. R. Yadav, F. W. Liou, and S. N. Balakrishnan, "A digital interface for the part designers and the fixture designers for a reconfigurable assembly system," *Math. Probl. Eng.*, 2013, doi: 10.1155/2013/943702.

[9] J. Shute *et al.*, "F1: A distributed SQL database that scales," *Proc. VLDB Endow.*, 2013, doi: 10.14778/2536222.2536232.

[10] V. V. Kumar, A. Sahoo, and F. W. Liou, "Cyber-enabled product lifecycle management: A multi-agent framework," 2019. doi: 10.1016/j.promfg.2020.01.247.

[11] S. K. R. A. Sai Charan Reddy Vennapusa, Takudzwa Fadziso, Dipakkumar Kanubhai Sachani, Vamsi Krishna Yarlagadda, "Cryptocurrency-Based Loyalty Programs for Enhanced Customer Engagement," *Technol. Manag. Rev.*, vol. 3, no. 1, pp. 46–62, 2018.

[12] V. Kumar, V. V. Kumar, N. Mishra, F. T. S. Chan, and B. Gnanasekar, "Warranty failure analysis in service supply Chain a multi-agent framework," 2010.

[13] V. V Kumar, M. Tripathi, S. K. Tyagi, S. K. Shukla, and M. K. Tiwari, "An integrated real time optimization approach (IRTO) for physical programming based redundancy allocation problem," *Proc. 3rd Int. Conf. Reliab. Saf. ...*, no. November 2014, 2007.

[14] V. V Kumar, "An interactive product development model in remanufacturing environment : a chaos-based artificial bee colony approach," *MASTER Sci. Manuf. Eng.*, 2014.

[15] D. Kengalagutti, "Comparing Database Management Systems : MySQL , PostgreSQL , SQLite," *Int. Res. J. Eng. Technol.*, 2020.

[16] K. Mullangi, V. K. Yarlagadda, N. Dhameliya, and M. Rodriguez, "Integrating AI and Reciprocal Symmetry in Financial Management: A Pathway to Enhanced Decision-Making," *Int. J. Reciprocal Symmetry Theor. Phys.*, vol. 5, no. 1, pp. 42–52, 2018.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 6.252
Volume 2, Issue 1, April 2022

[17] V. K. Yarlagadda and R. Pydipalli, "Secure Programming with SAS: Mitigating Risks and Protecting Data Integrity," *Eng. Int.*, vol. 6, no. 2, pp. 211–222, Dec. 2018, doi: 10.18034/ei.v6i2.709.
[18] S. Andjelic, S. Obradovic, and B. Gacesa, "A performance analysis of the DBMS - MySQL vs PostgreSQL," *Communications - Scientific Letters of the University of Žilina*. 2008. doi: 10.26552/com.c.2008.4.53-57.
[19] S. K. R. Anumandla, V. K. Yarlagadda, S. C. R. Vennapusa, and K. R. V Kothapalli, "Unveiling the Influence of Artificial Intelligence on Resource Management and Sustainable Development: A Comprehensive Investigation," *Technol. \& Manag. Rev.*, vol. 5, no. 1, pp. 45–65, 2020.
[20] C. Y. Huang, "Learning database through developing database web applications," *Int. J. Inf. Educ. Technol.*, 2019, doi: 10.18178/ijiet.2019.9.4.1207.
[21] V. K. Y. Mohamed Ali Shajahan, Nicholas Richardson, Niravkumar Dhameliya, Bhavik Patel, Sunil Kumar Reddy Anumandla, "AUTOSAR Classic vs. AUTOSAR Adaptive: A Comparative Analysis in Stack Development," *Eng. Int.*, vol. 7, no. 2, pp. 161–178, 2019.
[22] V. K. Yarlagadda, S. S. Maddula, D. K. Sachani, K. Mullangi, S. K. R. Anumandla, and B. Patel, "Unlocking Business Insights with XBRL: Leveraging Digital Tools for Financial Transparency and Efficiency," *Asian Account. Audit. Adv.*, vol. 11, no. 1, pp. 101–116, 2020.
[23] N. Richardson, R. Pydipalli, S. S. Maddula, S. K. R. Anumandla, and V. K. Yarlagadda, "Role-Based Access Control in SAS Programming: Enhancing Security and Authorization," *Int. J. Reciprocal Symmetry Theor. Phys.*, 2019.
[24] M. Z. Hasan, R. Fink, M. R. Suyambu, and M. K. Baskaran, "Assessment and improvement of intelligent controllers for elevator energy efficiency," 2012. doi: 10.1109/EIT.2012.6220727.
[25] N. G. Singh, Abhinav Parashar A, "Streamlining Purchase Requisitions and Orders : A Guide to Effective Goods Receipt Management," *J. Emerg. Technol. Innov. Res.*, vol. 8, no. 5, 2021.
[26] A. Nayak, A. Poriya, and D. Poojary, "Type of NOSQL databases and its comparison with relational databases," *Int. J. Appl. Inf. Syst.*, 2013.
[27] R. T. Mason, "NoSQL Databases and Data Modeling Techniques for a Document-oriented NoSQL Database," no. July, 2015, doi: 10.28945/2245.
[28] V. Sumalatha, "Overview of NoSQL Databases and A Concise Description of MongoDB," vol. 9, no. 12, pp. 295–299, 2020.
[29] Mani Gopalsamy, "Enhanced Cybersecurity for Network Intrusion Detection System Based Artificial Intelligence (AI) Techniques," *Int. J. Adv. Res. Sci. Commun. Technol.*, vol. 12, no. 1, pp. 671–681, Dec. 2021, doi: 10.48175/IJARSCT-2269M.
[30] S. G. Jubin Thomas, Kirti Vinod Vedi, "The Effect and Challenges of the Internet of Things (IoT) on the Management of Supply Chains," *Int. J. Res. Anal. Rev*, vol. 8, no. 3, pp. 874–878, 2021.
[31] S. G. Thomas Jubin, Kirti Vinod Vedi, "Enhancing Supply Chain Resilience Through Cloud-Based SCM and Advanced Machine Learning : A Case Study of Logistics," *JETIR*, vol. 8, no. 9, pp. 357–364, 2021.
[32] K. Patel, "Quality Assurance In The Age Of Data Analytics: Innovations And Challenges," *Int. J. Creat. Res. Thoughts*, vol. 9, no. 12, pp. f573–f578, 2021.
[33] M. Z. Hasan, R. Fink, M. R. Suyambu, and M. K. Baskaran, "Assessment and improvement of elevator controllers for energy efficiency," 2012. doi: 10.1109/ISCE.2012.6241747.
[34] M. Z. Hasan, R. Fink, M. R. Suyambu, M. K. Baskaran, D. James, and J. Gamboa, "Performance evaluation of energy efficient intelligent elevator controllers," 2015. doi: 10.1109/EIT.2015.7293320.
[35] Y. Li and S. Manoharan, "A performance comparison of SQL and NoSQL databases," pp. 15–19, 2013.
[36] S. Konkimalla, S. Bauskar, H. K. Gollangi, and E. P. Galla, "Predicting tomorrow ' s Ailments : How AI / ML Is Transforming Disease Forecasting," *ESP J. Eng. Technol. Adv.*, vol. 1, no. 2, pp. 188–200, 2021, doi: 10.56472/25832646/JETA-V1I2P120.
[37] V. N. Boddapati, E. P. Galla, and J. R. Sunkara, "Harnessing the Power of Big Data : The Evolution of AI and Machine Learning Harnessing the Power of Big Data : The Evolution of AI and Machine Learning in Modern Times," *ESP J. Eng. Technol. Adv.*, vol. 1, no. 2, pp. 134–146, 2021, doi: 10.56472/25832646/JETA-V1I2P116.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 6.252
Volume 2, Issue 1, April 2022

[38] W. Ali, M. U. Shafique, M. A. Majeed, and A. Raza, "Comparison between SQL and NoSQL Databases and Their Relationship with Big Data Analytics," *Asian J. Res. Comput. Sci.*, no. October, pp. 1–10, 2019, doi: 10.9734/ajrcos/2019/v4i230108.

[39] M. Gopalsamy, "Artificial Intelligence (AI) Based Internet-ofThings (IoT)-Botnet Attacks Identification Techniques to Enhance Cyber security," *Int. J. Res. Anal. Rev.*, vol. 7, no. 4, pp. 414–420, 2020.

[40] M. Gopalsamy, S. Cyber, and S. Specialist, "Advanced Cybersecurity in Cloud Via Employing AI Techniques for Effective Intrusion Detection," *IJRAR*, vol. 8, no. 1, pp. 187–192, 2021.

[41] M. S. Rajeev Arora, Sheetal Gera, "Impact of Cloud Computing Services and Application in Healthcare Sector and to provide improved quality patient care," *IEEE Int. Conf. Cloud Comput. Emerg. Mark. (CCEM), NJ, USA, 2021*, pp. 45–47, 2021.

[42] R. Arora, "Mitigating Security Risks on Privacy of Sensitive Data used in Cloud-based Mitigating Security Risks on Privacy of Sensitive Data used in Cloud-based ERP Applications," *8th Int. Conf. "Computing Sustain. Glob. Dev.*, no. March, pp. 458–463, 2021.

[43] P. ku. P. Biswajeet Sethi, Samaresh Mishra, "A Study of NoSQL Database," *Int. J. Eng. Res. Technol.*, vol. 67, no. 6, pp. 14–21, 2014.

[44] R. Bishukarma, "Impact of Cloud Computing Services and Application in Healthcare Sector and to provide improved quality patient care," *IJRAR*, vol. 8, no. 2, pp. 846–851, 2021.

[45] R. A. Anoop Kumar, Ramakrishna Garine, Arpita Soni, "Leveraging AI for E-Commerce Personalization : Insights and Challenges from 2020," pp. 1–6, 2020.

[46] D. Laksono, "Testing Spatial Data Deliverance in SQL and NoSQL Database Using NodeJS Fullstack Web App," 2018. doi: 10.1109/ICSTC.2018.8528705.

[47] G. Vonitsanos, A. Kanavos, P. Mylonas, and S. Sioutas, "A NoSQL database approach for modeling heterogeneous and semi-structured information," 2018. doi: 10.1109/IISA.2018.8633658.

[48] A. Flores, S. Ramirez, R. Toasa, J. Vargas, R. U. Barrionuevo, and J. M. Lavin, "Performance Evaluation of NoSQL and SQL Queries in Response Time for the E-government," 2018. doi: 10.1109/ICEDEG.2018.8372362.

[49] H. Gupta, S. Mondal, S. Ray, B. Giri, R. Majumdar, and V. P. Mishra, "Impact of SQL Injection in Database Security," 2019. doi: 10.1109/ICCIKE47802.2019.9004430.

[50] S. Reetishwaree and V. Hurbungs, "Evaluating the performance of SQL and NoSQL databases in an IoT environment," 2020. doi: 10.1109/ELECOM49001.2020.9297028.

[51] B. Namdeo and U. Suman, "A Model for Relational to NoSQL database Migration: Snapshot-Live Stream Db Migration Model," 2021. doi: 10.1109/ICACCS51430.2021.9441829.