International Journal of Innovation Scientific Research and Review
Vol. 07, Issue, 05, pp.8432-8447, May 2025
Available online at http://www.journalijisr.com
SJIF Impact Factor 2025: 7.913

Research Article

ISSN: 2582-6131

THE EVOLUTION OF RELATIONAL DATABASES: FROM TRADITIONAL MODELS TO CLOUD-BASED AND DISTRIBUTED SYSTEMS

***Chreesk Sabah M. Ali and Dr. Hajar Maseeh Yasin**

Akre University for Applied Sciences, Technical College of Informatics, Department of Information Technology, Duhok, Kurdistan Region, Iraq.

Received 08th March 2025; Accepted 09th April 2025; Published online 20th May 2025

ABSTRACT

This study examines the progression of relational databases from conventional, centralized RDBMS frameworks to contemporary distributed and cloud-based systems. It describes basic concepts such as indexing, transaction management, and organized schemas, while also highlighting drawbacks that have prompted the development of more adaptable data models. The study examines the transition from traditional SQL databases to newer paradigms, such as NoSQL and NewSQL, highlighting the significance of distributed systems and scalable cloud migration. Additionally, the study examines new developments that promise to improve real-time processing and data management flexibility, such as server less computing and AI integration. Overall, it offers a succinct but thorough summary of past developments and potential paths for database technology, providing practitioners and academics with insightful information to maximise contemporary data processing and storage options.

Keywords: Evolution of Relational Databases: Advancing from Traditional SQL to Scalable, Cloud-Driven, NoSQL & Distributed Data Management Systems.

INTRODUCTION

This paper examines the significant evolution of the relational database, exploring the various challenges it has presented throughout its history, as well as the recent developments that have emerged in the ever-changing landscape of database management systems. To set the stage, it is essential first to introduce the most fundamental principles that underpin relational databases. A relational database management system, commonly referred to as an RDBMS, is a sophisticated computer system equipped with applications that abstract, organize, and maintain data, all while invoking transactions in a well-defined and structured manner. Conceptually, users interact with this complex system through a query language, a declarative, question-oriented language designed for ease of use and efficiency. An RDBMS features a schema that precisely defines the structure of the records, or rows, to be stored, alongside the specific types of information associated with each entity or table. In general, a relational database will also generate a variety of indices. These indices are, in essence, search structures that serve to accelerate frequent queries and ensure that checkpoints do not prolong the retrieval process, resulting in a more streamlined and efficient user experience [1]. The system promises full transaction mapping, which is critical for maintaining data integrity. When performing a transaction, a series of changes is applied across multiple databases based on a logical view of the data. Physically, a transaction is requested from a primitive operation to be atomically addressed, analogous to ensuring that either all changes in a transaction are executed successfully and permanently stored on the disk, or none are if the transaction fails. The core of the RDBMS comprises the module responsible for managing transactions and scheduling these operations, in addition to controlling user queries to facilitate concurrent execution [2]. This crucial component is known as the transaction manager, which ensures the smooth operation of the system. Traditionally, records are stored using keys, with each record possessing scheme-compatible data types that adhere to the structured format of the RDBMS. However, there has been a growing interest in modern systems that prioritize essential transactional capabilities while not adhering strictly to the conventional RDBMS-style synchronization methods. Instead, these newer models operate on a data structure that is significantly more flexible than the rigid record/attribute storage characteristic of the traditional relational model. For the time being, though, the focus remains on the traditional RDBMS framework. As the project culminates, it offers a brief introduction to the realm of modern cloud-based and distributed database systems. These contemporary innovations represent the motivation behind addressing many of the longstanding issues associated with traditional RDBMS that are currently being highlighted in discussions within the field.[3]

***Corresponding Author: Chreesk Sabah M. Ali,**
Akre University for Applied Sciences, Technical College of Informatics, Department of Information Technology, Duhok, Kurdistan Region, Iraq.

HISTORICAL BACKGROUND

Automatic data migration to the cloud from relational databases is vital. Organizations often begin with private database servers and require a combination of database and application configurations to support growth. Many establish regional locations for broader coverage. RDBMS operates on the DB server for data operations using 2-phase committed transactions. Since access to data objects is unpredictable, the query processing path is also on the DB server to reduce performance issues. Initially, indexing is provided for data objects. Kernel-space operations are theoretically unlimited, so in-memory caching is utilized. Without a public API, many RDBMS optimizations are vendor-specific.[4]

The automatic data migration to cloud database architecture results in two sets of quadruples for each table: one for the number of hash indexes and one for the number of indexes on time, label, and primary key mapping fields. For consortia involving inter-commodities, the ability to decide under which table each object is located in the information retrieval path exists. A flat index file has a fixed schema with a list of keyed records, including each interpolation and configuration generating 4 4-tuple. Because the cloud database cluster may be periodically expanded and shrunk, a feature is

included that exports the RDBMS field's unique hashes of the data and non-intersecting divides the object hash space. Thus, the cloud database cluster implementation can deduce which quadruples describe object aggregation and de-allocation automatically [5]. Upon receiving a command flow that requires modifications to the target database configuration, the client-side app parses the command and, probabilistically, checks the corresponding indexes. At some point, applications exceed a few hundred objects, each in the hundreds of Gigabytes, and maintaining appropriate statistics without a very large storage is challenging. To overcome this problem, a subset of base objects with manually configured relevant indexes is designated. The preserved hash of the object's time field is calculated, and the output hash index, resulting in the cloud location, is derived. For the newly added hashing indexes during the cloud migration, as the app ages, it will attempt new hashes for an extended period until an appropriate hash space is established. [6]

TRADITIONAL RELATIONAL DATABASE MODELS

Each of the proposed frameworks relies on the design of powerful, flexible, and scalable methods for representing a wide range of Ratio Temporal Relations that will be encountered in practice. In this paper, two methods are described: the initiation of Duration Calculus-based algebra and the compilation of Constraint Models in a Constraint Logic Programming language. The aim is to establish a framework for representing and reasoning about rich sets of constraints involving future instances and time periods, as well as temporal relations. As technology continues to grow at an exponential rate, databases must keep pace, continually redesign, and expand. A decade earlier, the keyword used was centralized database. Currently, the term that is gaining significant momentum in the database sector is cloud-based databases. The cloud-based database enables the database owner to access advanced development tools, fast data retrieval, and global data sharing. Traditionally, databases are stored in a traditional record system previously known as a flat file system. Later, with the revolutionary growth in data requirements through databases, relational database modeling was introduced as a response. A framework is designed, and a formal description is provided for the temporal requirements problem in relational data modeling. Since the advent of the traditional record system, to address flat file issues and enhance data retrieval, a Relational Data Model was introduced. This work is based on the relational framework of RDM. [7]

Key Features of Traditional Models

Before the Big Data era, traditional relational database management systems (RDBMSs), such as MySQL and SQL Server, were widely used for data storage due to their transactional capabilities. Premium options included Oracle, IBM DB2, and PostgreSQL. As organizations encountered vast and diverse data volumes, RDBMS became less effective, leading to a shift toward more scalable models. Initially, Big Data focused on batch processing and data warehouses. The Hadoop community introduced HDFS and MapReduce as low-cost storage solutions, but faced challenges such as complexity and a lack of real-time query processing. Consequently, enterprises began to move away from them. To address these limitations, Apache projects such as HBase, Sqoop, Flume, Hive, Oozie, and Spark emerged; however, overcoming these challenges remained a significant challenge. Software vendors, such as Microsoft and Oracle, responded by launching APIs and proprietary solutions, yet they still encountered constraints. Oracle and IBM also addressed the open-source threat with cloud-based data processing systems, though issues persisted. Oracle faced challenges with column family solutions and secondary index support for range queries, while performance in cross-data center replication added more complexity. Systems often experienced backlogs during scaling, requiring data from both old and new sources, complicating performance. Ultimately, while robust, object storage lacked the write restrictions typically found in traditional file systems. [8][9]

Limitations of Traditional Models

The end of the 20th century sees us, in an increasingly frequent manner, new requirements and scenarios emerge that no longer fit properly within this relational model. Reasons for this are manifold. For once, the considerable penetration of the Internet and the widespread sharing and exchange of data that results. The ever-growing importance of this network also arises in connection with the analysis and management of vast amounts of data and activity, which must be stored and maintained. Another motivation is that, in this context, the scenarios change, and a new well-established tendency towards distribution emerges [10]. How to store and deal with the necessary vast amounts of data on the network within this environment without neglecting desirable scalability criteria. Additionally, properly following relational submission and distribution simultaneously makes a shift to other models or premises an attractive option. From here – in the following sections – there begins, then, after a short introduction and main topic, to detail and work on a more affine model, which gives attention to flat relations, intended to include a system to “emulate” the vision of today in both query time and space. [11]

THE RISE OF SQL

Over the past 20 years, exploring the history of databases has been akin to engaging with a well-loved aristocratic family. Initially, it feels like being an entertained guest or an eager student. As time passes, it evolves into a familiar conversation with an old friend, marked by humor, debate, and captivating stories. Although there is concern about political correctness, discussions can stray into uncomfortable territory, reflecting broader societal issues. This document aims to recount the history of databases without implying any structural equivalence. As noted by Breslauer, observing databases provides valuable insights. Ultimately, the experience reveals that databases represent both an “object of desire” and a rich source for historical understanding. [12]

Developers began to diversify their database spend on newer rivals to the tried-and-tested SQL Database. NoSQL databases emerged as an alternative to traditional relational systems, and the rise of big data and the cloud necessitated a different approach. These engines emerged in the mid-2000s, initially supporting specific use cases, such as search or column storage. By 2024, they had evolved to handle high-velocity, high-ingestion workloads and are widely considered faster and more flexible. The responders also said that SQL models had already had a good life, and now they needed something more straightforward. One of these solutions was the adoption of the NoSQL Data store. This more flexible and fast methodology allowed for modeling entities and properties without the need for data normalization, and for easily creating new attributes or data, while handling the high-performance requirements. [13]

SQL Language Overview

This chapter covers structured data processing with SQL, the generalizability of database extensions for training and inference of neural networks, and common strategies for embedding table data in neural network architectures. Relational database management systems are reliable, mature data stores. Information in the database can be retrieved and modified using SQL. With the SQL/MDM

approach, continuous data is retrieved from the database for training and querying, which is the typical machine learning use case. Additionally, database systems are efficient for expressive relational queries or machine learning algorithms on graphs by transforming the data offline into their relational representation.

Structured Query Language (SQL) was introduced in 1974 as an English-like query language and has been an ANSI/ISO standard since 1986. Initially, there was no "language spec," prompting database vendors to create their own dialects. Standardization began in the late 1980s, resulting in the first official standard in 1986, followed by a comprehensive rewrite in 1989 to address ambiguities. The third standard, SQL-92, was published in 1992 and marked the first normalization, removing redundant features. SQL-92 (ISO/IEC 9075-1:1992) is included in DB2 and MySQL implementations, with DB2 being IBM's most advanced database, although it only provides partial support for SQL-92. Major commercial relational database management systems support features like transactions and advanced SQL queries. MySQL, an open-source DBMS, supports advanced SQL queries but lacks transaction-related features. The MySQL server is started with "mysqld-nt -O my.cnf," while DB2 uses "db2start." [14]

Impact of SQL on Database Management

The SQL platform emerged in the mid-1970s as a response to existing database management systems, making it easier for users to access and interact with those systems. Structured Query Language (SQL) is the fundamental language for storing and managing data in relational databases. This idea was powerful because now people who wanted to use databases didn't have to be programmers; they could write SQL expressions to retrieve data. SQL enabled people who were focusing on business questions and not code to use the system. However, for a long time, a divide has existed in database management. There is a group of SQL users, those who write SQL expressions and submit them to the database server for execution. There is another group, consisting of more experienced industrial database administrators (DBAs) or professionally trained computer scientists, who understand what is happening inside the database, including storage, query execution, the query planner, and so on. The latter group is responsible for ensuring that databases remain operational and writes software used to manage databases. To be a good database programmer who writes high-performance applications, one typically needs to understand both sides. Too often, a divide exists between these groups for various reasons. [15][16]

ADVANCEMENTS IN RELATIONAL DATABASE TECHNOLOGY

Database management systems aim to enhance data processing performance. Various types include hierarchical, network, and relational database management systems (RDBMSs), which are preferred for storing structured data. RDBMSs utilize SQL for efficient data management and operations. Organizations also utilize data warehousing techniques for large datasets. However, locks in RDBMSs ensure transaction isolation, which can slow down performance as transaction volume increases. NewSQL offers a centralized lock manager, which can lead to bottlenecks. NewQ, a hybrid database system, combines the advantages of RDBMS with those of data warehousing for enhanced transaction monitoring. With the growth of the Internet, data scale, user numbers, and server capacity have surged, making databases essential for maintaining data integrity and processing. RDBMS remains the leading choice for structured data due to its effectiveness; however, traditional RDBMSs' locking mechanisms introduce overheads, such as deadlocks, which are crucial for sharing data amidst numerous concurrent queries. Alternatives have been proposed to improve scalability and minimize deadlocks without relying on locks. [17]

Normalization and Data Integrity

A relational database management system connects to databases and executes SQL queries to manipulate data. SQL statements are used to retrieve or update information within the database, with a significant focus on data settings and attributes. Data is organized in relational tables, and algorithms materialize query results in two-dimensional formats, even for multi-way joins. The algorithm's operator set includes various relational database system operators, while additional metadata, physical constraints on tables, and data access overheads are not part of the same systems. Modern data is typically stored in relational databases, which facilitate efficient complex queries. Just as homes can use gas or electricity for heating, databases can also utilize various formats to store diverse data types. However, fixed schemas enforce a standard structure while flexible formats often lack this capability. A golden record represents an idealized aggregation of information from multiple sources, raising the question of how to map these data sources to the golden record automatically. [18]

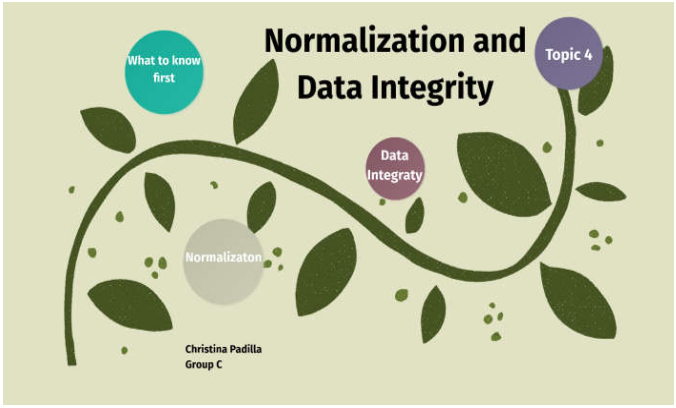

Figure 1: The figure representation about Normalization and Data Integrity.

Transaction Management

The complexity of current-day setups is becoming a burden for traditional recovery and concurrency control algorithms. A strategy is suggested to help keep pace with the proliferation of diverse hardware and workloads: unbundling these services from the underlying data management and making them available as cloud services. In this way, recovery and concurrency control can be provided outside normal DBMSs—a separation spanning all four layers of the classical architecture: storage, query processing, transaction management, and interface. [19]

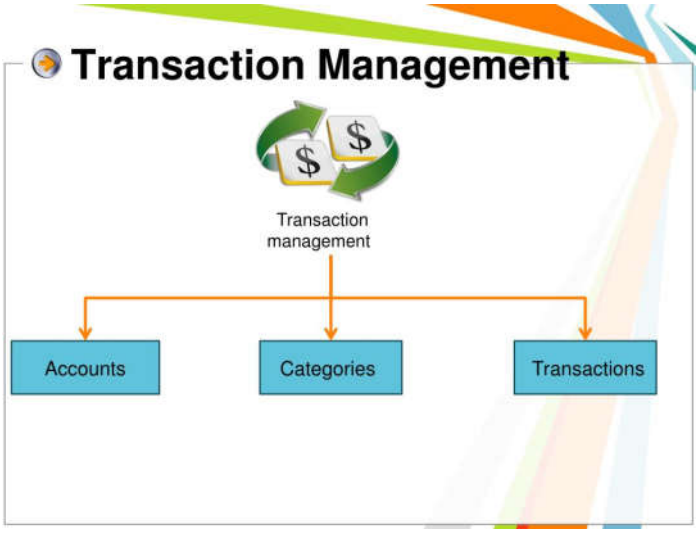

Figure 2: The figure representation about Transaction Management.

The emerging paradigm of cloud computing offers a robust platform for running various services, including those that are long-lived. Regarding transaction management, a radical, workload-adaptive approach is proposed. There are no hard-wired recovery and locking protocols in place. Each TC or DC service instance starts from scratch with a fresh log segment. The log is the only fixed point of recovery. Transaction semantics are deduced from log cleanliness. In current deployments, a DC should submit only Commit or Depend commands. The coordinate node acts as a journaling file system for those commands.

Concurrency management is still in the interface layer of a DC. Further evolution may move that support either higher up to a concurrency cloud outside the DC service proper, or lower down into the underlying TCs. Concurrency control per se may be handled in the cloud. The client API is then modelled with one of four degrees of express oblivious concurrency control. At the extreme, the database merely promises to attempt to fulfill read and write requests unless they require unobtainable resources. The ensuing speculation on the need for corresponding decision support among clients is now believed unnecessary. [20]

INTRODUCTION TO CLOUD-BASED DATABASES

Traditionally, database management systems have been used as back-end storage systems for web application servers. The conventional model involves installing the web server, middleware, and database server in a single location. The emergence of efficient web distribution tools and the increased cost efficiency and ease of availability of cloud resources have led companies to consider creating tools and storage within the cloud, accessible over the network. Cloud-based databases have been developed to facilitate the storage and retrieval of application data over a network. These databases are provided as services and account for the transactions and the storage capacity they have. Although cloud-based databases are increasingly mainstream, they have not been widely studied in conjunction with other cloud components and network infrastructure from several perspectives. This is because cloud computing, in general, and database systems, in particular, are multifaceted research domains.

The rise of cloud technology has led to the development of various distributed and NoSQL databases that address the limitations of traditional systems. These databases enhance throughput and uptime, offering network accessibility and scalability to meet high demand. They are designed for fault tolerance, ensuring continuity even if certain parts fail due to their high-availability architecture. Distributed NoSQL document-based databases are prevalent in cloud environments. However, the use of central cloud storage limits hybrid cloud applications, as remote caches access data more slowly than local databases. Data is replicated across various data centers in one or more clouds, with central databases serving as backends for distributed cloud systems. Popular NoSQL databases partition data into buckets on separate nodes. [21]

Definition and Characteristics

This section examines the development and functioning of DBMSs. For years, the database community has focused on creating scalable DBMSs. The approach to building a 1000-node DBMS is now well-defined. It highlights two prominent database service providers: Google Big table and Apache HBase. Database research aims to create a DBMS that is easier to deploy, maintain, and extend than existing systems. A key technology for scalable systems is the distributed hash table, with Ballista serving as a simple example on a small cluster. It supports two scalable systems: one focuses on robust, predictable performance DBMS clusters, while the other explores multi-data center systems, leading to the development of Cloud Span and Spanner. The discussion concludes with open challenges in the DBMS domain. [22]

There is a wide variety of database systems available in the cloud, each with its own distinct emphasis and target applications. Researchers struggle with the question of what characterizes the "database-ness" of a system. A multidimensional classification of the cloud database space is proposed, identifying ten dimensions that vary widely among the systems investigated. These dimensions include "strong" multi-row transactions, dynamic schema updates, and non-deterministic query evaluation. The survey assesses the premises, technical details, and impact of the most representative space systems in these dimensions. Some preliminary findings of the research are presented. [23]

Advantages of Cloud Solutions

This section discusses the evolution of traditional relational databases to the current cloud-based and distributed database systems. The basic concepts of various new SQL and NoSQL systems are also reviewed. Traditional databases have been relational databases for decades. The basic building block of these systems is a table in which data is stored in an ACID-compliant way. In this model, each table has a schema that defines the fields and their types. The schema must be defined in advance so existing fields cannot be easily modified or deleted. There are well-established relational database management systems. The language used to query relational databases is SQL, which is based on which SQL database processing engines are built. SQL is known for its set-based interface, where a single SQL statement can operate on hundreds, thousands, or even millions of records. [24]

Databases moved to the Web and web applications. Web-interpreted programming languages can be used to generate HTML. To add dynamics to these generated pages, interaction with the database is necessary. In the PHP world, the most famous database engines are MySQL and PostgreSQL. These engines did not have built-in database drivers, but instead, DB interactivity was achieved through a myriad of API functions. In the ASP world, access to the database was through ADO. Java introduced some frameworks that abstract the resource allocation and management from the programmer.

Agility? To avoid redundancy, ensure better control of updates, and design the system for better performance.

With the advent of cloud computing, the application of database concepts has undergone significant changes. Cloud databases are designed for thin clients that send their queries to powerful cloud servers. How do cloud databases work? Instead of being stored in flat text files, the data is stored in relational database servers functioning in a private cloud. Each database server collects information from multiple tables and executes complex SQL queries. Since the database format is hidden, it cannot be queried by a generic database engine. A cloud server can communicate with many database servers and with other cloud services. A reply is formatted, containing the requested data. With the plurality of service providers, there is no specific formatting convention. The data may need to go through various language parsers. This format is complex and difficult to parse. [25]

DISTRIBUTED DATABASE SYSTEMS

Over the last few decades, the relational database has become the dominant type of database used by many companies, institutions, and, more recently, personal users who generate vast volumes of data through their activities, including e-commerce, internet usage, and video gaming. However, the relational model has been evolving, playing catch-up with the evolving relational database systems. Then, in the next section, we will dive deeper into the exploration of this evolution. However, this evolution compelled some companies to select new database engines that diverged from the traditional relational database model. [26]

The traditional database model was initially designed for centralized architectures; however, the growing amount of data that companies must store has led to the development of quantitatively superior business rules, algorithms, and software applications that the traditional database model cannot handle. In this scenario, relational databases, which support queries written in the SQL language, began to reach computational and storage limits as data volumes grew. This led to an investigation of novel database models capable of efficient parallel and/or distributed processing, designed to scale out with new computing paradigms.

In a distributed database system (DDBS), a database is stored across geographically distributed sites, and each site is managed by a local Database Management System (DBMS) that can access and update the data in its own site as well as the data located at other sites. The DB of a DDBS can provide the illusion to a user that the data stored is all centralized. Furthermore, a DDBS must provide transparency for data distribution or replication, fragmentation, and remoting. [27]

Concepts and Architectures

Database system architectures are undergoing significant changes as algorithms and data integrate with programming languages, transforming each DBMS into a web service. Customers can access stored procedures and manipulate data online. Strong servers run User-Defined Functions (UDFs) as compiled code due to the I/O-bound nature of HTTP. DBMSs function as object containers, housing collections of objects and tables, where each object comprises various components with distinct schemas and methods. These containers are divided into schemas that allow inter-object referencing via fully qualified names. Transaction processing and workflow applications are based on queues, with asynchronous flows defined by chains of stored procedures interacting with queues. DBMSs have incorporated queue management into their architecture for over a decade, with OS-level support in Windows NT and UNIX, and have introduced declarative queues that extend T-SQL. Most DBMSs now include integrated support for data cubes and online analytic processing, along with frameworks for data mining and machine learning. Data mining adjusts model parameters to fit data. Offline data mining separates model training and production queries, enabling runtime queries against specialized databases and models, with the scoring operation supported by the DBMS.[28]

Challenges in Distributed Systems

The last decade of the 20th century witnessed the rise of object-oriented and object-relational database management systems, which merged relational DBMSs with SQL access and complex data types. This evolution led to extensive analysis of relational database technology, exploring its history to recent developments in multi-strata and heterogeneous query languages. Trends in distributed and federated data management were also examined, highlighting the challenges posed by scalable system solutions in modern information systems. New database applications often outperformed prior storage technologies and DBMS products, benefiting from advancements within existing hardware-software constraints. Research shifted toward intelligent databases, particularly deductive and federated systems. [29]

The primary conceptual schema design policy for very large databases was initially established through partitioning and replication, and has since been refined by newer methodological or heuristic contributions, which currently dominate use and acceptance. Some other claimed principles now seem dated. The early manifestations of some important future research and application trends have been overlooked, including the fact that extensively studied optimization problems are NP-complete for large databases and non-trivial queries, but not for the most uncomplicated practical cases. The same is true for the latter-day concerns about the technology lock-in effects.

COMPARATIVE ANALYSIS OF DATABASE MODELS

Databases have a significant role in the development of computer technologies. The first databases were developed in the 1960s, utilizing hierarchical and network data organization models. This model consists of n-to-n relationships for data that is awkward for many structures and therefore cannot be managed effectively. Furthermore, hierarchical and network data organization models should consider the relationships between the tables. Making a change in one table can create a chain of changes in other tables that have relationships with the altered table. [30]

A researcher working at IBM developed a new system in which the data was kept in tables. He introduced the inner join and outer join, which relate rows in two or more tables. SQL is developed for DBMSs, provides the possibility of working with multiple tables together, and cleaning the data to be ready for analysis. A relational DBMS consists of a set of tables, each with a unique name. The following queries are used for making operations on a database: (1) Inserting data, (2) Deleting data, (3) Updating the data, and (4) Getting data from the database using queries. Later in 2000, JSON-based DBMSs, such as MongoDB, were introduced as an alternative to relational DBMSs. With the advent of cloud and distributed systems, this model is preferred for use in conjunction with NoSQL DBMS. With the increasing volume and complexity of data, these systems manage data efficiently and effectively. [31]

Traditional vs. Cloud-Based

Relational Database Management Systems (RDBMS) have long been the standard for data management. Still, the Internet's explosion of data has rendered them inadequate for the speed and volume required by modern web applications. This has driven many enterprises toward NoSQL databases. In RDBMS, queries and storage are processed by a single server, which creates inefficiencies, particularly when scaling to meet high data request ratios. After the dot-com bubble, server availability became a constraint, forcing companies to invest heavily in larger machines to function as DB Servers, often leading to high licensing costs as CPU counts increased. Scaling out by adding multiple machines has become more common in cloud environments, offering a more economical solution through pay-per-use pricing and reduced hardware costs. Smaller machines can boost computing power and resilience. Companies like Google and Amazon utilize key-value pair systems for their scalable infrastructure, with the "Web Scale Principle" advocating for distribution and rapid recovery from failures. However, these systems often fall short in transaction guarantees, which can be critical. As data demands evolve with the growth of the Internet, suitable data models have become essential, reflecting the importance of specialized roles, as seen in sports. As RDBs became dominant through Codd's relational model, their limitations in handling large-scale, high-concurrency environments have prompted a move to NoSQL solutions. One challenge in this transition is the reliance on SQL in existing RDBs, coupled with resistance from vendors regarding data exports. While transferring data between storage types can be straightforward, migration may require API updates and modifications to outdated data models to accommodate new storage solutions. [32]

Cloud-Based vs. Distributed

Relational database management systems have been the standard for data persistence and management. However, a plethora of NoSQL and NewSQL systems have emerged over the past decade, offering interesting persistence options for various types of domains. This new landscape comprises a family of data stores and access paradigms that differ from the traditional relational model. NoSQL databases are recognized for their performance, flexibility, and scalability advantages in various use cases, suggesting that these properties are crucial for the needs of modern applications. NewSQL database systems are a relatively recent development that employs a SQL-like query language to ensure operational consistency on a horizontally scalable, clustered system. This ensures that loads are uniformly distributed among servers and that requests are executed quickly enough to validate the service rate agreement's strictly predefined objectives. NoSQL and NewSQL database management systems are commonly designed as distributed database management systems, offering a single database management system across multiple nodes. Cloud computing, characterized by providing faster internet access to archetypal computer systems and processing tools, is beneficial for developing on-demand usage spikes. Owing to its eligibility requirements, cloud computing affords flexibility in the resources that provide a suitable environment for deploying DDBMSs. [33]

FUTURE TRENDS IN RELATIONAL DATABASES

Relational database management systems (RDBMS) have been the standard for data persistence for decades. However, the last decade has seen the rise of new database management systems (DBMS) such as NoSQL and NewSQL. These systems provide robust solutions for Web applications and emerging domains, including Big Data and IoT. NewSQL systems are based on relational models, while NoSQL offers various storage types, including key-value, document-oriented, column-oriented, and graph-oriented stores. Both NoSQL and NewSQL are optimized for high performance and scalability, functioning as distributed database management systems (DDBMS). Cloud computing has facilitated this evolution by providing quick access to commodity hardware through elastic, on-demand resource provisioning. Infrastructure as a Service (IaaS) is favored for deploying DDBMS, offering flexibility in compute, storage, and network resources. [34]

The database landscape has undergone significant changes over the last decade. A plethora of new database management systems, typically classified into NoSQL and NewSQL, have evolved. A NewSQL DBMS is a scalable, distributed system designed for online transaction processing (OLTP) operations, providing ACID guarantees inspired by the relational model. On the contrary, NoSQL DBMSs do not support transactions and joined operations, and they typically organize and operate on large datasets. A system is not a "classical" DGBMS as long as it either lacks a SQL-like query language or does not purely store relations and enforce the respective relational model features. Aside from this separation, the classification criteria used to categorize databases include the storage model, intended purpose, persisting abstractions, distribution model, distribution of data storage abstractions, and transactional properties.

Artificial Intelligence Integration

RDBs have been the most popular DBMSs for many years due to their simplicity and the ability to efficiently manage the ever-growing volumes of data. However, with the growing volumes of data and its increasing complexity, the RDBMS has proven to be inefficient. Therefore, the DBMS industry has been compelled to develop alternative systems for addressing such tasks. There are numerous modern DBMS systems for big data on the market, ranging from traditional relational model-based DBMSs to modern cloud-based and distributed data processing systems, such as Aurora DB and Snowflake DB. All modern cloud-based and distributed systems, such as Apache Cassandra, Amazon Redshift, and Google BigQuery, are generally built on a distributed architecture.

AI is generating global interest as "smart data" has transformed working paradigms. Recently, the impact of AI and machine learning on databases has increased in tandem with the surge in data volumes, making large-scale data technologies a preferred industry model. AI addresses the challenges of extensive data spreads and quickly extracts information, particularly with datasets that are unsuitable for traditional relational database management systems (RDBMSs). This is evident in filtering web information and integrating distant sources. The key goal is to develop adaptable, intuitive, and efficient modeling schemes, with the evolution of various data models driving a comparison approach. Historically, the database and AI fields have collaborated, characterized by established semantics and the integration of object-oriented systems with knowledge-based frameworks. [35]

Given the ongoing trend of data growth with an increasing ratio of unstructured data, it is compelling for modern enterprises to store, index, and efficiently retrieve their new datasets. Database engines have historically absorbed many innovations in data processing paradigms and become the de facto execution back-end. In this paper, it is claimed that achieving a truly AI-centric database engine requires moving the DBMS engine from a low-level relational to a high-level tensor abstraction. This enables multi-modal data

processing and leverages innovations in hardware and runtimes developed for tensor computation.

Serverless Database Architectures

Serverless Database Architectures. Its revolutionary on-demand pricing combined with easy-to-use, scalable compute made event-driven architectures much more accessible. Consequently, serverless computing has seen rapid growth, and many cloud providers now support it. Nevertheless, server-side databases still rely on fixed provisioning and are inextricably tied to compute serving the same purpose. This paper opens the black box on the database aspects of Serverless and presents AnyDB. On an abstract level, it proposes changes in how events are handled and materialized, which in turn introduces a novel ecosystem that allows all DBMS systems to work with a designated event artist. From an infrastructural perspective, it suggests the beneficial requirements for future Serverless-ready NoSQL databases and stream handling, which should evolve to include concepts from the event-driven programming paradigm. [36]

CASE STUDIES OF MODERN RELATIONAL DATABASES

For approximately fifty years, relational databases have been the primary solution for storing, retrieving, and managing data. However, since the early 2000s, the rise of the web, big data, and cloud technologies has led to the emergence of NoSQL databases, which are characterized by fault tolerance, high availability, and scalability. These technologies have made NoSQL databases more accessible and affordable for managing big data. There are four primary types of NoSQL databases: document databases, column families (also known as wide-column stores), key-value stores, and graph databases. Despite their benefits, NoSQL databases do not provide all the features of relational databases, which remain popular for OLTP applications requiring ACID guarantees or complex queries with sophisticated joins. The development of relational database management systems has also advanced, incorporating features such as procedural objects (functions and triggers) written in various procedural languages. These objects help maintain database integrity by centralizing constraints, akin to T-SQL for application architects. However, numerous studies indicate that these procedural objects can adversely affect database performance, even degrading the execution of simple SQL queries. This performance drop is tied to the optimization process, which relies on heuristic rules to create execution plans, leading to increased complexity and maintenance challenges for stored procedures. [37]

Successful Implementations

The concept of scaling up has evolved with the rise of affordable, shared-nothing commodity servers, moving beyond traditional SQL and enterprise databases. While some enterprise solutions demonstrate effective scaling, factors such as lower long-term costs, administrative expertise, and multiple robust out-of-the-box solutions have led many startups to favor these machines. Consequently, a regular relational schema is necessary, which entails a slight scalability trade-off due to the need for specialized management of buffering, query optimization, and concurrency to maintain ACID properties. Developers often prefer NoSQL systems to avoid the bottleneck of centralized databases and the requirement for predefined schemas.

A relational DBMS is essential and sufficient for most applications. Scalability may be needed to add servers while accommodating multiple TBs of data. These requirements render most NoSQL data stores unviable, making it crucial to find a 3-tier middleware implementation to enhance the scalability of MySQL, PostgreSQL, or SQL Server. Although relying on tested software in production is ideal, that's often not feasible. In-house development might be preferred, similar to client and server code, or existing cloud services complicate middleware control. Therefore, the focus is on analyzing available off-the-shelf systems. [38]

Lessons Learned from Failures

Distributed Computer Systems Networks span the Web, taking advantage of the ubiquity of TCP/IP and other IP protocols in Local and Wide Area Networks. Provisioning and management of enterprise data centers are enhanced through the utilization of the utility model, both for hardware procurement and operations, as well as by renting services. Information Technology now seems so pervasive within all types of organizations that business models for Information Technology Services truly appear to represent the wave of the Future. In a world that is networked end-to-end, everything, from desktops to large server installations, will be accessed through networks from unknown and unknowable parts of the globe. The Web is seen as a superb enabler for many applications that benefit from the ability to efficiently, reliably, and robustly move data to and from clients and servers.

While Networks and End Systems have advanced significantly, the Model of Application Interaction has largely remained unchanged. Applications utilize protocols such as NFS, FTP, HTTP, CORBA, DCOM, and SQL for remote resource interfacing, relying on a transport layer that ensures reliable delivery. This places the responsibility for fault tolerance and reliability on application servers and end systems, which offers clean abstractions. However, as dependence on these services grows, applications often fail in unpredictable ways, revealing the fragility of distributed systems and limitations of current models. These failures underscore the urgent need for robust, secure, and reliable network systems, driving significant research in distributed services and systems, and emphasizing the importance of cross-sector dialogue to address core challenges. [39]

Best Practices for Database Management

This section presents 11 areas of current good practice and makes connections between them and the evolution of technology.

- Within an organization, multiple databases will exist, and standards are necessary for them to communicate effectively with each other over the internet. Online transaction protocols are necessary for this, and new ones will emerge.
- Clients and intermediaries of a service may have a large amount of personal and access-restricted data that needs to be accessed selectively by the service. A new public and open agreement for brokering remote feeds from private or authenticated sources may fulfill that role.
- Often, write access to databases is granted to services that should not have it, e.g., to services with a public API. In this context, restrict the possibilities when providing a database service manager URI.
- In scientific instruments and any measurement, the raw data is collected first. Relating that data to a sample or subject later, after it has been generated, is necessary for reproducibility. There needs to be a simple way to annotate raw scientific measurements with externally resolvable URIs that has a very low barrier to entry, is valid to disclose, and unambiguous, yet

allows a machine to uniquely identify the generated data, the measurement, and the thing measured. [40]

- There should be a clean, simple, predictable, and consistent way to represent dates, date times, and durations in formats that are easy and stable to generate in any programming language.
- There are many services related to projects, deliverables, products, or the like that are open and list able; however, for various reasons, it may still be desirable to create a local replica. That generally needs a compatible representation. For an object with a series of events, additional cases may then arise for representing actions and their consequences.
- A point should be reached where any curated information has machine-process able metadata about who did it, when, how, and so forth, such that the trust can be extended via transitivity.
- Once a database system reaches a certain size, debugging one user's problems within it can become tedious. An experimental approach to having contracting customers and DBMS teams agree on handover boundary files that can be quickly checked post-issue for potential problems is described.
- Since scientific data comes in a wider variety of formats and encodings than web documents, there is a need to update the data mediation approach to determine when the original fetch was a non-CSV file. This will identify the first worksheet of the file, list the full contents of the used range in the representation, and provide HTML with a simple form to submit for downloading the used range in the requested format.
- Based on archaeological evidence, as well as historical data representations of city names, it is revealed that even median ages of 50 years are prone to inaccuracies in interpretation. For improved parts amalgamation or spatial analysis, this lack of certainty is problematic, and it is proposed that a more comprehensive change-tracked method be mandated for the addition and deletion of named entities on records.
- During the lifetime of a project, it is common to work with a series of similar or recurring entities, such as social scientists performing repeated services to a sequence of projects. Often, such new entities should be based on an existing one, but being distinct, they should be recognizable by their own URI. A simple mechanism is needed to derive new URIs from existing ones. [41]

Security Considerations

Databases are essential for individuals and small businesses in modern computing, although their usage can be subtle. Several unresolved issues exist in this field, particularly regarding the management of big data and the development of efficient technological infrastructures. This subsection reviews computer science efforts in database management, highlighting current research and its proximity to active projects. The chapter emphasizes time constraints, structured into five parts: initial challenges for real-time data processing, emerging research questions, development of two key areas, and a concluding overview.

There are numerous directions and issues proposed for study, as they align with the author's research and are believed to be potential starting points for developing new and original ideas. Ensuring that adequate security and privacy mechanisms are in place is important for the correct operation and safe use of database technology. In particular, databases have been the target of three main reasons for many successful attacks: they can hold large quantities of files, often contain sensitive information, and are open to network access. Unfortunately, despite intense attention and effort on the matter over the past thirty years, some welcome improvements have been made; however, the growth of threats has far outpaced the technology used to prevent them [42].

Performance Optimization Techniques

In this section, the study outlines the storage methods used for PDB data and the techniques employed to optimize the database. Further details concerning the techniques employed will also be discussed.
To achieve the goal, it is crucial to understand certain aspects. A literature review is conducted on PDB data storage methods and database optimization techniques for efficient querying. The study provides details on the optimization techniques employed, serving as a reliable source. The literature encompasses relational databases for storing protein structure data, including amino acids, ligands, and secondary structures, as well as graph databases for protein information. However, many studies overlooked database optimization or the overall system efficiency. Thus, a set of improvements to the system's design is presented and evaluated.

Two particular papers may be cited. The first study outlines a methodology for optimizing relational databases through denormalization, a phase that bridges the gap between logical and physical modeling. The second technique optimizes relational databases by combining multiple tables into a single table, thereby reducing the data collection needs for queries. Additionally, another form, partitioning, is discussed for its features. An algorithm for optimizing databases using denormalization strategies aims to enhance analytical query processing of in-memory data. Furthermore, a study explores various optimization methods for relational databases, with a focus on improving the data mining process. The examination reveals that denormalization and indexing can significantly enhance database performance. However, caution is needed in using these methods, particularly denormalization. Overall, if configured correctly, a relational database may offer superior querying speeds compared to other options.

LITERATURE REVIEW

Al-Khatib et al. (2023) [43] present a novel framework that unifies disparate databases using an object-oriented approach. Three phases comprise their architecture: a preprocessing phase that efficiently gathers data using materialized views, an OODB construction phase that builds a single object-oriented database, and a deployment phase that leverages cloud and web-based computing resources. This comprehensive method outperforms conventional methods by a factor of 2.49, significantly enhancing query performance while simplifying the integration of multiple data sources. For large data scenarios where scalability and agility are crucial, the paper offers practical management guidance.

A cutting-edge data processing platform, designed explicitly for pavement quality management, based on the Internet of Things (IoT), is presented by **Hong et al. (2023)** [44]. By utilizing a NoSQL database design, which enables rapid data ingestion and real-time analysis, the solution mitigates the limitations of conventional relational databases. The technology significantly reduces data processing durations, which can exceed 21 ms in traditional systems, to approximately 0.405 ms, thereby facilitating swift decision-making on construction sites. This study exemplifies the utilization of contemporary data architectures to optimize civil engineering maintenance practices and enhance infrastructure monitoring.

Syeda Husna et al. (2024)[45] develop a comprehensive cybersecurity trust model in their paper, addressing the numerous security issues associated with cloud computing. The approach

combines Quantum Key Distribution (QKD) with a Modified Advanced Encryption Standard (MAES) to provide multi-layered protection mechanisms and integrates blockchain technology to build a tamper-resistant ledger. The framework's remarkable accuracy rate of 99.84% and encryption and decryption operations that take only milliseconds (2.25 ms and 1.071 ms, respectively) demonstrate its effectiveness in protecting data from both established and emerging threats. This study highlights the benefits of decentralized security solutions in enhancing the integrity and trust of cloud infrastructure.

Chinnasamy et al. (2023)[46] utilize smart contracts and blockchain technology to address the critical issue of secure data exchange for health purposes. Their framework offers a decentralized, trust-based access control system that guarantees the integrity and security of electronic health records (EHRs). The suggested model utilizes distributed ledger technology in place of centralized authentication mechanisms to mitigate the risks of unauthorized access and data breaches. By enhancing the dependability of mobile cloud-based e-health services and facilitating real-time monitoring and sharing of private medical data, this secure sharing mechanism helps to offer healthcare more safely and effectively.

Haut et al. (2024) [47], concentrating on the 2010 Gulf of Mexico event, offer a state-of-the-art cloud-based method for evaluating hyperspectral data to identify oil leaks. To accurately identify oil spills, especially in complex coastal habitats, the study utilizes the normalized difference oil index (NDOI) to extract crucial spectral information. MapReduce with HDFS and Apache Pig are two examples of distributed computing techniques that the framework utilizes to efficiently handle the massive amounts of data inherent in remote sensing applications. This scalable system enables prompt and precise decision-making in response to ecological emergencies by accelerating processing times and providing a robust foundation for real-time environmental monitoring and disaster management.

Using distributed representation models, **Khan et al. (2023)**[48] offer a unique method for identifying emergent topic patterns in streaming news data. To analyze trending topics in real-time as they evolve, their News Sequential Evolution Model (NSEM) utilizes word2vec to capture semantic linkages across sequential datasets. The framework enhances trend detection accuracy and provides decision-makers in dynamic situations with intuitive insights by incorporating a visual display model and creating a knowledge graph. This paper addresses significant challenges in processing massive text streams and offers a scalable solution for various market analysis and social media monitoring applications.

A thorough analysis of how cloud computing technologies are changing industrial operational technology (OT) networks can be found in **Perducat et al. (2023)** [49]. This research examines the evolution of conventional models, such as the ISA-95 framework, in light of new developments, including zero-trust architectures and policy-based software-defined networks. By investigating the integration of cloud solutions with legacy OT infrastructures, the study highlights the security concerns prevalent in contemporary industrial environments, as well as the opportunities for enhanced connectivity. This thorough research is a crucial tool for understanding how OT networks are evolving in response to digital convergence.

Matthew et al. (2024) [50] propose an inventive architecture for creating e-health data warehouses that utilizes machine learning techniques to enhance data mining procedures. To facilitate risk management, fraud detection, and identity authentication, the project focuses on integrating multimodal healthcare records with smart sensor data within a cloud-based system. The authors employ a bimodal sensor access technique, supported by artificial neural networks, to demonstrate how enhanced data mining can improve decision support in cutting-edge e-healthcare services. This work highlights the significance of scalable data architectures in addressing the complex challenges of contemporary healthcare data management.

The NODW framework, presented by **Imran et al. (2021)** [51], promotes a paradigm change in data warehousing in significant data contexts by moving away from conventional relational databases and toward NoSQL-based solutions. The framework maximizes data aggregation and query efficiency in online analytical processing scenarios by utilizing the capabilities of column-oriented storage systems, particularly Apache Cassandra. By overcoming the drawbacks of traditional ETL procedures and relational join operations, the suggested paradigm improves scalability and efficiency. It is ideally adapted to manage the growing amounts of diverse data. This study presents a compelling perspective on integrating NoSQL technology into modern data warehousing architectures.

Ionescu and Diaconita (2023) [52] investigate how financial decision-making is being revolutionized by the convergence of artificial intelligence, cloud computing, and improved data management technologies. This thorough assessment examines the transition from conventional relational database systems to contemporary big data designs, which utilize technologies such as Hadoop, Apache Spark, and Blockchain for secure, real-time data processing. The paper addresses the challenges of data security and system integration, while highlighting the notable gains in operational efficiency, risk management, and predictive analytics that are made possible by combining these technologies. For financial organizations seeking to establish more data-driven and responsive operational frameworks, this synthesis provides valuable insights.

To handle the complexity of distributed systems, such as blockchains and distributed ledgers, **Di Pilla et al. (2023)**[53] propose a novel auditing architecture. To monitor, analyze, and categorize log data in real-time, the article introduces the DELTA tool, a Distributed Elastic Log Text Analyzer that leverages artificial intelligence, natural language processing, and advanced data analytics. The study demonstrates how systematic log auditing can identify vulnerabilities, detect anomalies, and facilitate proactive mitigation techniques across various distributed contexts by integrating Docker-based log collection with the ELK stack for storage and visualization.

The revolutionary significance of data engineering at the intersection of artificial intelligence, machine learning, and cloud computing is examined by Muthusubramanian and Jeyaraman (2023) [54]. The study examines the importance of robust data engineering methods in developing scalable cloud architectures and optimizing advanced machine learning and artificial intelligence applications. The authors emphasize the importance of integrated approaches in achieving enhanced system performance and operational efficiency by thoroughly examining current trends and challenges and highlighting the synergistic relationship between effective data pipelines and cutting-edge computational models.

Revathi et al. (2023) [55] present an automatic translation approach that transforms relational database schemas into non-relational JSON documents. Without interfering with existing configurations, the proposed algorithm facilitates a seamless transition from conventional SQL-based systems to document-oriented NoSQL databases. When tested on a dataset from a hospital management system, the model demonstrates that it can maintain important relational data while

providing the performance, scalability, and flexibility characteristic of NoSQL environments, thereby meeting the increasing demands of big data applications.

Wang et al. (2024)[56]offer a hybrid approach that combines robust cloud computing infrastructures with cutting-edge deep learning algorithms to improve tailored search. To capture complex user preferences and query semantics, their method employs a multi-layer transformer model enhanced by hierarchical attention networks. Cloud-based deployment ensures the system's scalability, enabling real-time processing and responsiveness. Significant gains in search accuracy and user satisfaction are demonstrated by experimental evaluations, establishing this integrated approach as a viable solution for contemporary search engines addressing complex and large-scale data challenges.

The strategic change of SAP's business model within the dynamic database management system (DBMS) market is examined in a teaching case presented by **Balodi et al. (2023)** [57]. In line with broader industry trends toward cloud usage and digital convergence, the paper documents the transition from conventional, pipeline-based database management system (DBMS) solutions to a holistic platform approach. It examines the pressures from competitors, especially Oracle. It discusses the crucial strategic decisions SAP must make to maintain its market leadership and adapt to new customer value propositions in a rapidly changing environment.

A viewpoint on the evolution of control systems in the cloud computing era is provided by **Xia (2024)** [58]. Cloud-based control systems (CCSs) are proposed as the architecture of the future, following a study that examines how the enormous amounts of data and real-time requirements of modern industrial applications challenge traditional networked control systems. The study emphasizes intriguing pathways, such as data-driven cloud control and model-predictive cloud management, by leveraging the computational power and scalability of cloud computing. These methodologies are particularly relevant for complex applications, such as industrial automation and intelligent connected vehicles. This paper establishes the theoretical foundation for subsequent research on integrating control theory with cloud architectures.

The query optimisation technique presented by **Alyas et al. (2022)** [59] is designed for graph databases operating in cloud dew environments. The framework addresses the specific issues posed by distributed systems in relation to graph data structures, encompassing resource allocation and efficient query processing. The authors utilize cloud-based techniques to demonstrate improved performance when handling complex graph queries, thereby increasing scalability and reducing response times. Since it offers insightful information on how to optimize graph database operations in cloud settings, this work is a significant step towards more efficient big data analytics in networked systems.

The revolutionary shift from cloud-native to AI-native computing is examined by **Lu et al. (2024)** [60] in the context of massive generative models. To better meet the resource-intensive needs of large AI models, such as ChatGPT and similar programs, the paper explains how traditional cloud infrastructures are being transformed. The authors demonstrate the challenges of cost, scalability, and GPU resource availability by contrasting large-model-as-a-service with cloud database services. They propose an AI-native paradigm that integrates advanced machine learning runtime frameworks with cloud-native architectures. This paper offers a prospective view on the convergence of AI with cloud computing, delineating potential research avenues and practical applications.

Through data-driven decision-making, **Naeem et al. (2024)**[61] investigate how sophisticated database systems might improve corporate intelligence. The paper examines different database architectures, ranging from contemporary big data platforms to conventional relational systems, and highlights how their integration can facilitate strategic decision-making and real-time analytics. The study compares various systems, highlighting their advantages in terms of processing speed, scalability, and the ability to handle diverse data sources. This thorough analysis highlights the importance of developing database technologies in facilitating informed decision-making in today's competitive commercial environment.

Al-Kateeb (2024) [62] explores the potent combination of big data analytics, cloud computing, and the Internet of Things (IoT). The study describes how these technologies work together to create a networked ecosystem that can revolutionize data gathering, processing, and analysis across various sectors. By addressing the potential and challenges, such as scalability, data protection, and effective resource management, the study demonstrates how integrated solutions may promote operational efficiency, innovation, and proactive decision-making. This study demonstrates how a coordinated strategy can yield valuable insights and foster a more adaptable technological future.

The impact of cloud-based accounting education on academic achievement at Omani universities, both during and after the COVID-19 pandemic, is evaluated by **Tawfik and Elmaasrawy (2023)** [63]. The study highlights the important human, cultural, social, technological, and economic factors that influence the evolution of cloud-based accounting education, as well as student outcomes, through a structured survey and partial least squares data analysis. For governments and educational institutions seeking to enhance e-learning strategies, particularly in addressing post-pandemic educational challenges, the findings provide crucial insights.

A cloud-based approach designed to manage the large and intricate datasets produced by satellite remote sensing is presented by **Wang et al. (2024)** [64]. The study examines the volume, variety, velocity, authenticity, and utility of big data derived from remote sensing, as well as the resulting challenges in processing, storing, and analyzing it. The authors provide a comprehensive platform that improves data fusion, administration, and accessibility by utilizing cloud computing technology. The study highlights the relevance of cloud solutions in supporting crucial decision-making processes in domains such as resource management and environmental monitoring, while outlining the technological challenges and potential opportunities for enhancing remote sensing applications.

DISCUSSION AND COMPARISON

Table 1: Comparative analysis of all evaluated papers.

No.	Database Type / Focus	Performance Evaluation Metrics	Technology & Tools	Data Model / Conceptual Framework	Scalability	Flexibility	Limitations
[43]	Integration of cloud objects for heterogeneous data	assessed using integration efficiency (Details are not entirely clear.)	Cloud computing: design that is focused on objects	Object-focused	made to handle diverse data; probably scalable	High: accommodates a range of data formats	Complexity of integration; details are not quite clear
[44]	NoSQL for managing the quality of IoT pavements	Speed of data collection: comparison between 0.405 ms and 21.146 ms	Cloud computing, big data analytics, and the Internet of Things	NoSQL, or not-relational	Designed to handle large volumes of data in real time	Schema-free structure allows for flexibility.	Conflicts between consistency and support for complicated queries
[45]	Based on a structure of trust for cybersecurity	Qualitative indicators of reliability and danger mitigation	Cloud technology and crypto	A distributed database	Potential issues with scaling in blockchain technology	Sturdy construction	Limitations with flexibility and connection with older systems
[46]	Safe communication in smart electronic health systems was made possible by smart contracts.	Evaluation of security efficiency and access management qualitatively	the digital ledger, smart contracts, and mobile clouds	Distributed (influenced by blockchain)	scalable in situations involving mobile devices	Depending on modifications to the smart contract	Dependence on cryptocurrency efficiency and network latency
[47]	Based on the cloud hyperspectral analysis of images for identifying oil spills	Metrics of memory usage and processing performance were provided.	Big data, satellite imaging, and the Internet of Things	Multifaceted arrays of images	Large datasets are supported via cloud infrastructure.	Extremely flexible for a variety of picture data	Computational overhead for hyperspectral information with extremely high resolution
[48]	Analyzing streaming text data to identify patterns in popular topics	The style detection reliability and delays (qualitative evaluations)	NLP, streamed analytics, as well as distributed models (like word2vec)	Vector interpretations in the form of text	Developed for flow analysis in real time	Adjusts to changing subjects	Capturing small semantic changes over time might be challenging.
[49]	Evolution of industrial operational networks (with an emphasis on cloud integration)	Measurement of qualitative performance of networks (throughput, latency)	Cloud computing and the integration of industry OT	Process-oriented (not specifically defined)	allows for the integration of many OT devices.	Connects to older OT devices	Convergence between information technology and operational technology and security standardization challenges
[50]	E-health data analysis and designing warehouses	Metrics regarding query effectiveness and data analysis accuracy were published.	Usage in e-health, analysis of data, and artificial intelligence	multifaceted; star structure	Designing scalable warehouses for big health information	A structured framework may limit rapid modifications.	Integrating diverse health data while preserving data quality
[51]	Big data from a NoSQL approach for information warehouses	Enhancements to query speed; evaluation of the impact of join procedures	NoSQL technology (like Apache Cassandra) with big data analytics	Complex, schema-less star models	Scalable distributed structure	High adaptability in schema architecture	Trade-offs for reliability, difficulties with intricate join operations
[52]	a hybrid system that combines NoSQL and traditional components	Evaluation of the effectiveness of real-time processing and risk estimation	Large amounts of data, the blockchain, cloud-based computing, and AI	Models of hybrid data	extremely scalable using cloud-based solutions	uses a variety of data management techniques	Problems of unification; difficulties with combining data and security
[53]	Distribution of system auditing (log analysis for cybersecurity)	Qualitative enhancements in recognizing anomalies and analysis	Advanced analytics, AI, natural language processing, and the Docker and ELK stack for collecting logs	A traceability scheme based on logs	Scaling with attention to distributed systems	Adaptable integration with different frameworks	Dependent on log styles, limited consistency
[54]	Advances in data engineering that combine cloud, machine learning, and AI	Perspectives on efficiency and optimization (not precisely measured)	AI, ML, computing in the cloud, and sophisticated data engineering methods	Data engineering information model/pipeline	Scalable through cloud-based systems	Modular and flexible methods for data engineering	Heterogeneous system integration challenges and changing approaches

[55]	Relationship documents are automatically converted to non-relational entities.	Evaluation of the transformation's accuracy and efficiency (minimum details)	Techniques for data conversion that automate translation	Relationship → text-oriented translation type manages enormous databases when appropriately automated	manages enormous databases when appropriately automated	transforms automatically while requiring little physical labor.	Limited assessment of various database structures; customization might be necessary
[56]	Using AI to improve tailored search	Average precision increased by 15%, cost effectiveness improved by 12%, and 95% of inquiries had a latency of less than 200 ms	Custom software, cloud computing, and deep learning	Customized search model (multi-layer, dynamic)	scalable by the use of cloud resources (like AWS)	Extremely flexible about user information and search settings	Privacy issues and the difficulty of integrating with outdated search systems
[57]	The development of SAP's business model and the DBMS industry	Market research and competitive positioning are used to evaluate the strategic effectiveness.	The business model evaluation, migration to the cloud, and a case study of the DBMS industry	Business model transition from funnel to interface	Migration to the cloud improves flexibility	Adaptable change of business models	Pressures from competitors and difficulties in winning the trust of customers
[58]	Controllers that run on the cloud for contemporary industrial applications	Qualitative evaluation of management effectiveness and system effectiveness	Integration of sensor data, management systems, and cloud computing	Cloud resource integration in the control design	Scalable if more IoT and control devices are added	Capable of adjusting to intricate industrial procedures	Real-time processing difficulties and computational constraints
[59]	A technique for optimizing queries in cloud dew graph databases	reported gains in optimization effectiveness and query time of execution	Optimization of query strategies, cloud and rain computing platforms, and graph databases	A data structure that is graph-oriented	Designed for dispersed cloud dew situations in mind	Adjusts to changing patterns of graph queries	For really big graphs, more tuning might be necessary; integration overhead
[60]	Using AI-native computing instead of cloud-native computing for big models that generate	explains throughput enhancements, such as up to 14x in specific techniques.	Containerization, distributed computing, creative AI models, and cloud native designs	A hybrid strategy that transitions from cloud-based to AI-native	Auto-scalable and scalable with container orchestration	Adaptable via dynamic scalability and multi-tenant architectures	High interpretation costs, difficulties accessing resources, and the complexity of integration
[61]	sophisticated databases for company intelligence (decisions based on data)	evaluated using profitability and processing performance (both qualitative and quantitative findings).	Hadoop, Spark, big data analytics, sophisticated DBMS technology, and hybrid systems	Mixed and multifaceted models	Extremely scalable with cloud-based and distributed platforms	Adaptable combination of various data storage strategies	Complexity of integrating with old systems; difficulties brought on by the quick advancement of technology
[62]	Combining big data, cloud computing, and IoT to get revolutionary results	assessed using operational effectiveness, security of information, and continuous processing (qualitative insights)	The cloud, computing at the edge, big data analytics, and the Internet of Things	The big-data environment with integrated IoT and cloud	Distributed and cloud-based technologies that are very scalable	Adaptable integration between various data and sensor sources	Challenges with data security and privacy, difficulties with managing enormous amounts of data
[63]	Oman's academic achievement and accounting through cloud instruction	utilizes PLS path coefficients, which reveal strong positive correlations between variables.	Cloud technology, online learning environments, and survey-based analyses (PLS)	Conceptual framework combining economic, technological, cultural, and social aspects that impact cloud-based learning	Prioritized uptake of education over scale.	encourages the use of hybrid (online and in-person) teaching methods	Only students are included in the sample; faculty viewpoints are absent, and survey-based self-report biases
[64]	Cloud-based system and solutions for remotely collected Big Data (RSBD)	displays efficiency measures and real-time processing information, such as query response time of less than one second.	Integrated tables, geographic information systems, internet computing, simultaneous processing, and flexible data sharing	Multifaceted picture and metadata model for remote sensing	Extremely scalable (made for worldwide coverage and PB-level data)	Incredibly adaptable by combining data from multiple sources and scales	Integration difficulties, perhaps exorbitant cloud storage expenses, and intricate data mining specifications

An extensive table presents an integrated review of 22 research publications covering various facets of contemporary data management and processing systems. Cloud-based object-oriented integration, NoSQL architectures for Internet of Things applications, blockchain-based security models, sophisticated data warehousing systems for business intelligence, and even big data solutions for remote sensing are all included in the table, categorized by their respective areas of focus. Key performance indicators, including processing speeds, query execution improvements, and throughput upgrades, are compiled along with the underlying technologies (such as blockchain, cloud computing, IoT, and AI) and the corresponding data models or conceptual frameworks used. Additionally, the chart outlines the flexibility and scalability of each strategy, highlighting how these systems manage massive, diverse datasets, as well as their drawbacks, such as trade-offs in consistency or complexity, and integration difficulties. The table provides a helpful overview of the various ongoing research initiatives and technological advancements in the domains of cloud computing, data engineering, and related areas.

EXTRACT STATISTICS

The assessment metrics integrate both qualitative and quantitative insights to deliver a holistic perspective on system performance. Efficacy and efficiency are significant topics, with a particular focus on processing performance and query speed. Regular qualitative evaluations evaluate managerial effectiveness, risk estimation, and reliability. Quantitative metrics highlight enhancements, including data collection speeds ranging from 0.405 ms to 21.146 ms, a 15% increase in average precision, and a 12% improvement in cost efficiency. The strategic focus on operational efficiency and security is further underscored by other key metrics, including memory utilization, real-time query response times of under one second, and throughput enhancements of up to 14 times. When combined, these metrics provide a robust framework for assessing overall performance, identifying anomalies, and optimizing systems in both technical and strategic contexts, as demonstrated in Figure. 3.

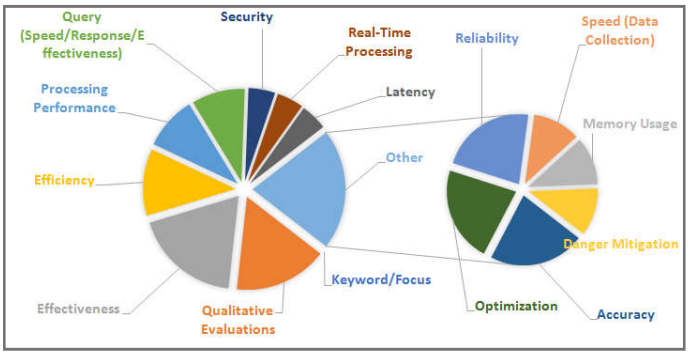

Figure 3: A statistical depiction of the frequency of Evaluation Metrics for Performance.

Scalability is a recurring theme in the frequency analysis, as evidenced by the 14 instances where the terms "scalable" or "scaling" are explicitly used, highlighting its crucial importance in system design. Significant focus is placed on cloud-based scalability, referenced on seven occasions, underscoring the dependence on cloud infrastructure for accommodating substantial data quantities and facilitating real-time processing. Distributed architectures are a prevalent theme, with three references emphasising the imperative for distributed solutions to manage data flow and provide resilient performance across geographically scattered systems. Additionally, scalability difficulties in developing domains such as mobile and IoT settings are acknowledged, noted on two occasions. At the same time, the notion of auto-scalability via container orchestration emerges as a specialised yet significant method. The overall frequency makes it abundantly evident that contemporary system designs must be dynamically adaptable and able to handle ever-increasing and diversified data demands, even though one example juxtaposes scalability with the importance of education (see Figure 4).

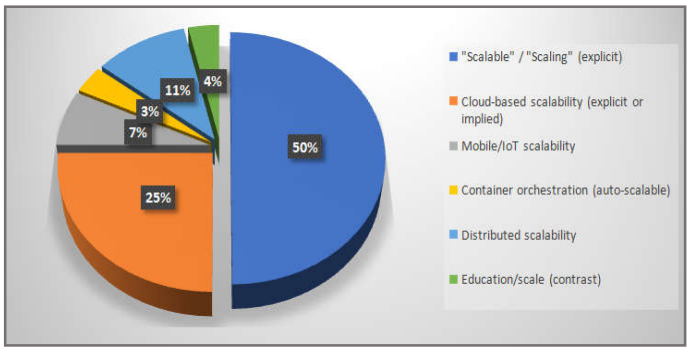

Figure 4: A statistical depiction of the prevalence of Scalability.

System adaptability is heavily emphasised, according to the frequency analysis of keywords related to flexibility. The word "adaptable" is used seven times, indicating that the design prioritizes flexibility to easily interact with other frameworks, data sources, and changing requirements. The system's ability to support a variety of data formats and user preferences, including schema-free structures and adjustable search options, is highlighted by the four instances in which the term "flexible" is used. Furthermore, three uses of the word "adjust" (or its variations) emphasise the significance of dynamic responsiveness, whether that be adjusting to shifting topics, business models, or data patterns. All things considered, as shown in Figure 5, these frequency counts indicate that flexibility is a crucial design objective, ensuring the system can adapt to a wide range of rapidly changing operational environments.

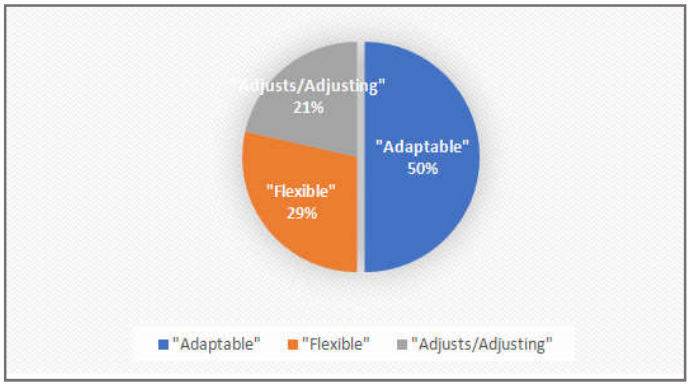

Figure 5: Flexibility's frequency represented statistically.

Operational issues and system integration emerge as key topics in the frequency table, which clearly shows a pattern of recurring difficulties. "Difficulties" is the most commonly used term, implying a broad variety of challenges that continuously impact many facets of system deployment and maintenance. Additionally, "integration" and its variations receive considerable attention, highlighting the difficulties of combining different systems. The constant reference to complexity and data management suggests that managing large, complex datasets is a persistent challenge that warrants serious consideration and innovative solutions. Furthermore, the existence of security and privacy issues underscores the importance of information protection in this rapidly evolving technological environment. Overall, the frequency counts suggest that resolving operational obstacles, ensuring smooth system integration, and

maintaining strong data security are interconnected problems that require coordinated plans and creative solutions, as illustrated in Figure. 6.

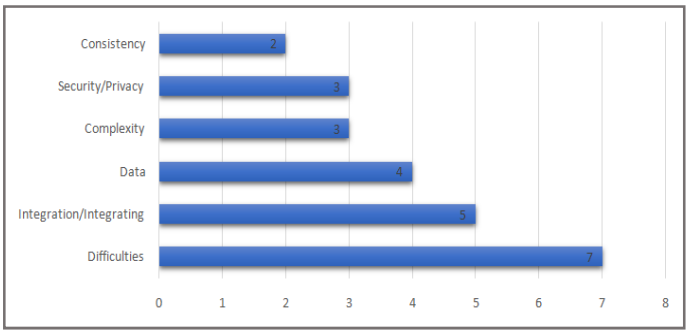

Figure 6: Statistical representations of frequencyLimitations.

Recommendation

Anyone working in database administration or research should read this paper since it provides a thorough examination of the development of relational databases. It offers a critical study of the move towards cloud-based and distributed architectures in addition to revisiting the fundamental ideas of conventional RDBMS.

Principal strengths encompass:

The evolution of relational databases from centralised systems to the more dynamic, scalable forms we see today is explained in a great historical perspective. This background clarifies for readers why contemporary problems call for more recent models.

A thorough analysis of the shortcomings of old models paved the way for the development of NoSQL and NewSQL systems. The advantages and disadvantages of SQL-based systems and these more recent paradigms are skillfully contrasted in the study.

Perceptive segments on contemporary topics, like AI integration and serverless database systems. These anticipatory insights are particularly beneficial for IT professionals and system architects devising future technological plans.

The difficulties of integrating data across dispersed systems, security, and performance optimisation are all covered in this thorough analysis of database administration best practices.

Overall, the article offers practical insights and a robust framework for understanding how evolving data demands are shaping new technologies, serving as both a retrospective and a roadmap for the future of database management.

CONCLUSION

This study concludes by demonstrating the amazing evolution from conventional, centralised relational databases to contemporary distributed and cloud-based systems. It illustrates how persistent issues, such as strict schemas, scalability constraints, and intricate transaction management, have prompted the creation of data structures that are more adaptable, scalable, and effective. The way businesses handle and process large, varied datasets has undergone significant changes with the adoption of NoSQL and NewSQL paradigms, as well as the integration of AI and serverless technologies. Ultimately, the paper emphasizes that although classic RDBMS frameworks remain fundamental, adopting contemporary technologies is crucial for meeting the increasing demands of global connectivity and real-time data processing in today's digital landscape.

REFERENCES

[1] V. Govindaraj, "The Future of Mainframe IDMS: Leveraging Artificial Intelligence for Modernization and Efficiency," Journal of Advanced Computer Science & Applications, 2024. researchgate.net

[2] M. R. Anwar, R. Panjaitan, and R. Supriati, "Implementation of Database Auditing by Synchronization DBMS," in IT Service Management, 2021. academia.edu

[3] L. Li and J. Zhang, "Research and analysis of an enterprise E-commerce marketing system under the big data environment," Journal of Organizational and End User Computing, vol. 2021. igi-global.com

[4] Y. Himeur, M. Elnour, F. Fadli, N. Meskin, I. Petri, "AI-big data analytics for building automation and management systems: a survey, actual challenges and future perspectives," Artificial Intelligence, vol. 2023, Springer, 2023. springer.com

[5] Y. Aldwyan and R. O. Sinnott, "Elastic deployment of container clusters across geographically distributed cloud data centers for web applications," Software: Practice and Experience, vol. 51, no. 5, pp. 1021-1042, 2021. [HTML]

[6] Y. Mansouri, F. Ullah, S. Dhingra, "Design and implementation of fragmented clouds for evaluation of distributed databases," Transactions on Cloud, vol. 2023. [PDF]

[7] C. Xu, X. Du, X. Fan, G. Giuliani, Z. Hu, "Cloud-based storage and computing for remote sensing big data: a technical review," Journal of Digital, vol. 2022, Taylor & Francis. tandfonline.com

[8] E. S. Kumar, S. Kesavan, and R. C. A. Naidu, "Comprehensive analysis of cloud-based databases," in IOP Conference Series, 2021. iop.org

[9] J. J. K. Behan, A. Inam, M. Ali, and M. T. Khan, "Comparative analysis of RDBMS and NoSQL databases," Knowledge and Data, 2022. upv.es

[10] H. Yu, "The application and challenges of ChatGPT in educational transformation: New demands for teachers' roles," Heliyon, 2024. cell.com

[11] J. C. Recker and R. Lukyanenko, "From representation to mediation: a new agenda for conceptual modeling research in a digital world," MIS Quarterly, vol. 2021. qut.edu.au

[12] M. Nielsen, "Propertius through Wilson: Books of Latin Love Poetry as Objects," 2023. unc.edu

[13] K. Tatsis, "A quantitative study on the popularity and performance of SQL and NoSQL DBMS," 2022. diva-portal.org

[14] S. Morozova, "Uses and relevancy of old and new programming languages," 2023. theseus.fi

[15] A. Shuparskyy and Y. Furgala, "SELECTED ASPECTS OF DIGITAL REPRESENTATION OF INFORMATION SYSTEMS," Electronics and Information, 2024. lnu.edu.ua

[16] G. O'Regan, "Birth of software industry and human computer interaction," A Brief History of Computing, 2021. [HTML]

[17] N. K. Miryala, "Evolving Trends in Open-Source RDBMS: Performance, Scalability and Security Insights," researchgate.net, researchgate.net

[18] V. Yatsyshyn, O. Pastukh, R. Zharovskyi, "Software tool for productivity metrics measure of relational database management system," Математичне …, 2023. irbis-nbuv.gov.ua

[19] J. Zhou, M. Xu, A. Shraer, B. Namasivayam, A. Miller, "FoundationDB: a distributed key value store," in ACM SIGMOD, 2022. sigmodrecord.org

[20] B. Modu, M. P. Abdullah, M. A. Sanusi, et al., "DC-based microgrid: Topologies, control schemes, and implementations," Alexandria Engineering Journal, vol. 2023, Elsevier. sciencedirect.com

[21] A. K. Sandhu, "Big data with cloud computing: Discussions and challenges," Big Data Mining and Analytics, 2021. ieee.org

[22] H. M. Elgohary, S. M. Darwish, and S. M. Elkaffas, "Improving uncertainty in chain of custody for image forensics investigation applications," IEEE Access, 2022. ieee.org

[23] F. Song, Z. Qin, L. Xue, J. Zhang, and X. Lin, "Privacy-preserving keyword similarity search over encrypted spatial data in cloud computing," *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3456-3468, 2021. uwaterloo.ca

[24] W. Khan, T. Kumar, C. Zhang, K. Raj, and A. M. Roy, "SQL and NoSQL database software architecture performance analysis and assessments—a systematic literature review," *Big Data and Cognitive Computing*, vol. 2023. mdpi.com

[25] M. Ismail, N. El-Rashidy, and N. Moustafa, "Mobile cloud database security: problems and solutions," Fusion: Practice and …, 2021. researchgate.net

[26] J. Robinson, R. Ranjan, W. Hu, "Relbench: A benchmark for deep learning on relational databases," in Advances in ..., 2024. neurips.cc

[27] S. H. Alfred, O. A. Augusta, and L. K. Egi, "Impact of distributed database management system to individuals, institutions, and organizations all over the world," BW Academic Journal, 2022. bwjournal.org

[28] A. Skiadopoulos, Q. Li, P. Kraft, K. Kaffes, D. Hong, "DBOS: A DBMS-oriented operating system," 2021. mit.edu

[29] K. Siau, C. Woo, V. C. Storey, R. H. L. Chiang, and C. E. H. Chua, "Information systems analysis and design: past revolutions, present challenges, and future research directions," 2022. mst.edu

[30] G. W. Kyro, A. M. Smaldone, Y. Shee, C. Xu, "T-ALPHA: A Hierarchical Transformer-Based Deep Neural Network for Protein–Ligand Binding Affinity Prediction with Uncertainty-Aware Self-Learning for Protein …," in *Journal of Chemical Information and Modeling*, 2025. biorxiv.org

[31] P. Vikström, M. Larsson, E. Engberg, and S. Edvinsson, "The Demographic Database—History of Technical and Methodological Achievements," SOWING, 2023. oapen.org

[32] S. Youssef, "… for Large-Scale Enterprise Applications: A Comprehensive Study on Techniques, Challenges, and the Integration of SQL and NoSQL Databases in Modern …," ResearchGate, . researchgate.net

[33] T. Taipalus and H. Grahn, "NewSQL database management system compiler errors: Effectiveness and usefulness," *International Journal of Human–Computer Interaction*, vol. 39, no. 1, pp. 1-12, 2023. tandfonline.com

[34] W. Qi, M. Sun, and S. R. A. Hosseini, "Facilitating big-data management in modern business and organizations using cloud computing: a comprehensive study," Journal of Management & Organization, 2023. researchgate.net

[35] M. M. Rathore, S. A. Shah, D. Shukla, and E. Bentafat, "The role of AI, machine learning, and big data in digital twinning: A systematic literature review, challenges, and opportunities," in *IEEE*, 2021. ieee.org

[36] A. Mampage, S. Karunasekera, and R. Buyya, "A holistic view on resource management in serverless computing environments: Taxonomy and future directions," ACM Computing Surveys, vol. 2022. [PDF]

[37] M. Naeem, T. Jamal, J. Diaz-Martinez, and S. A. Butt, "Trends and future perspective challenges in big data," in *Intelligent Data Analysis*, vol. 2022, Springer. minciencias.gov.co

[38] K. Reddy, "Scalable Data Management in Distributed Systems: A Sharding-Based Approach for Multi-Tenant Architectures," International Journal of Emerging Research in …, 2024. ijeret.org

[39] A. Sunyaev, "Applications and Systems Integration," in Internet Computing: Principles of Distributed Systems, 2024, Springer. [HTML]

[40] S. Mukherjee, D. Stamatis, J. Bertsch, et al., "Genomes OnLine Database (GOLD) v. 8: overview and updates," Nucleic Acids Research, vol. 49, no. D1, pp. D1-D6, 2021. oup.com

[41] C. J. Markiewicz, K. J. Gorgolewski, F. Feingold, R. Blair, et al., "The OpenNeuro resource for sharing of neuroscience data," Elife, 2021. elifesciences.org

[42] Ö Aslan, S. S. Aktuğ, M. Ozkan-Okay, A. A. Yilmaz et al., "A comprehensive review of cyber security vulnerabilities, threats, attacks, and solutions," Electronics, 2023. mdpi.com

[43] Al-Khatib, R. M., El-Omari, N. K. T., & Al-Betar, M. A. (2023). An innovative cloud computing object-oriented model to unify heterogeneous data. International Journal of Operational Research, 46(3), 289–322. https://doi.org/10.1504/IJOR.2023.129410.

[44] Hong, S.-S., Lee, J., Chung, S., & Kim, B. (2023). Fast real-time data process analysis based on NoSQL for an IoT pavement quality management platform. Applied Sciences, 13, 658. https://doi.org/10.3390/app13010658

[45] Husna, S., Barmavatu, P., et al. (2024). A blockchain-based cybersecurity trust model with a multi-risk protection scheme for secure data transmission in cloud computing. Cluster Computing. https://doi.org/10.1007/s10586-024-04481-9.

[46] Chinnasamy, P., Albakri, A., Khan, M., Raja, A. A., Kiran, A., & Babu, J. C. (2023). Smart contract-enabled secure sharing of health data for a mobile cloud-based e-health system. Applied Sciences, 13, 3970. https://doi.org/10.3390/app13063970.

[47] Haut, J. M., Moreno-Alvarez, S., Pastor-Vargas, R., Perez-Garcia, A., & Paoletti, M. E. (2024). Cloud-based analysis of large-scale hyperspectral imagery for oil spill detection. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 17, 2461. https://doi.org/10.1109/JSTARS.2023.3344022.

[48] Khan, Z. A., Xia, Y., Ali, S., Khan, J. A., Askar, S. S., Abouhawwash, M., & El-Rashidy, N. (2023). Identifying hot topic trends in streaming text data using news sequential evolution model based on distributed representations. IEEE Access, 11. https://doi.org/10.1109/ACCESS.2023.3312764.

[49] Perducat, C., Mazur, D. C., Mukai, W., Sandler, S. N., Anthony, M. J., & Mills, J. A. (2023). Evolution and Trends of Clouds on Industrial OT Networks. OJIA, 4. https://doi.org/10.1109/OJIA.2023.3309669.

[50] Matthew, U. O., Onumaku, V. C., Fatai, L. O., & Adekunle, T. (2024). E-Healthcare data warehouse design and data mining using ML approach. International Journal of Advanced Computer Science and Applications, 1, Article 84. https://doi.org/10.4018/979-8-3693-4439-2.ch013.

[51] Imran, S., Mahmood, T., Khan, A. H., Qamar, A. M., Siddiqui, A. J., Ahmed, I., & Rehman, N. (2021). NODW framework for data warehousing – A NoSQL big data perspective. Journal of LaTeX Class Files, 14(8).

[52] Ionescu, S.-A., & Diaconita, V. (2023). Transforming financial decision-making: The interplay of AI, cloud computing and advanced data management technologies. International Journal of Computers Communications & Control, 18(6), Article 5735. https://doi.org/10.15837/ijccc.2023.6.5735.

[53] Di Pilla, P., Pareschi, R., Salzano, F., & Zappone, F. (2023). Listening to what the system tells us: Innovative auditing for distributed systems. Frontiers in Computer Science, 4, 1020946. https://doi.org/10.3389/fcomp.2022.1020946

[54] Muthusubramanian, M., & Jeyaraman, J. (2023). Data engineering innovations: Exploring the intersection with cloud computing, machine learning, and AI. Journal of Knowledge Learning and Science Technology, 1(1). https://doi.org/10.60087/jklst.vol1.n.p84.

[55] Revathi, K., Tamilselvi, T., Dhanwanth, B., & Dhivya, M. (2023). Auto JSON: An automatic transformation model for converting relational databases to non-relational documents. International Journal of Advanced Computer Science and Applications, 14(3). https://doi.org/10.14569/IJACSA.2023.0140377.

[56] Wang, J., Lu, T., Li, L., & Huang, D. (2024). Enhancing personalized search with AI: A hybrid approach integrating deep learning and cloud computing. Journal of Advanced Computing Systems, 4(10), 1–13. https://doi.org/10.69987/JACS.2024.41001

[57] Balodi, K. C., Jain, R., Kiran Kumar, T. B., & Banerjee, D. (2023). Platform revolution in the database management system industry: Evolution of SAP's business model [Teaching case]. Journal of Information Technology Teaching Cases, 13(1), 126–133. https://doi.org/10.1177/20438869221106798

[58] Xia, Y. Q. (2024). Cloud-based control systems: Towards the control architecture in cloud computing era. Science China Information Sciences, 67(10), 206201:1–206201:3. https://doi.org/10.1007/s11432-023-4156-5.

[59] Alyas, T., Alissa, K., Niazi, Q. A., & Tabassum, N. (2022). Query optimization framework for graph database in cloud dew environment. Computers, Materials & Continua. https://doi.org/10.32604/cmc.2023.032454.

[60] Lu, Y., Bian, S., Chen, L., He, Y., Hui, Y., Lentz, M., Li, B., Liu, F., Li, J., Liu, Q., … Zhuo, D. (2024). Computing in the era of large generative models: From cloud-native to AI-native.

[61] Naeem, Z., Folorunso, E. O., Chu, T. S., Mamun, M. A. A., & colleagues. (2024). Data-driven decision making: Advanced database systems for business intelligence. Nanotechnology Perceptions, 20(S3), 687–704. https://doi.org/10.62441/nano-ntp.v20iS3.51

[62] Al-kateeb, Z. N. (2024). Unlocking the potential: Synergizing IoT, cloud computing, and big data for a bright future. Iraqi Journal for Computer Science and Mathematics, 5(3), 1–13. https://doi.org/10.52866/ijcsm.2024.05.03.001

[63] Tawfik, O. I., & Elmaasrawy, H. E. (2023). Assessing the factors that affected the development of cloud-based accounting education and students' academic performance in Oman. Arab Gulf Journal of Scientific Research, 41(2), 141–157. https://doi.org/10.1108/AGJSR-07-2022-0102

[64] Wang, H., Shi, S., Liu, H., & Ma, X. (2024). A cloud-based solution and platform for remote sensing big data applications: Challenges and opportunities.

****