Information Research Communications, 2025; 2(1):71-97.
https://inforescom.org

Research Article

A Comparative Study of Relational, Graph, Wide Column, Key-Value, and Document Models in Retail Industries

Lauren Hyun-Ee Wong[1], Jia Wen Ng[1], Angel Xian Theng Tan[1], Mae Jhin Yong[1], Su-Lyn Lim[1], Sathishkumar Veerappampalayam Easwaramoorthy[1,*], Usha Moorthy[2]

[1]School of Engineering and Technology, Sunway University, Jalan Universiti, Bandar Sunway, Selangor Darul Ehsan, MALAYSIA.
[2]Department of Information Technology, Manipal Institute of Technology Bengaluru, Manipal Academy of Higher Education, Manipal, Karnataka, INDIA.

ABSTRACT

Aim/Background: The study addresses the growing need for efficient database management systems in the rapidly expanding online retail sector, especially post-COVID-19. It aims to identify the most suitable data model for handling retail operations by comparing five types: relational, graph, wide-column, key-value, and document databases. The focus is on evaluating their performance in retail-specific functionalities such as order processing and Customer Relationship Management (CRM). **Methodology:** Researchers created test environments using five database platforms-Oracle APEX, ArangoDB, AstraDB, Redis, and CouchDB. Each was populated with 200–1000 records and assessed under two retail scenarios: order management and CRM systems. Data was inserted via JSON files and evaluated based on CPU usage, processing speed, and memory consumption. Performance data was collected using both database logs and system monitoring tools. **Results:** Oracle APEX (relational) demonstrated the most consistent performance across all tested metrics, including data integrity, query speed, and memory efficiency. ArangoDB excelled in handling complex relationships but required tuning. AstraDB showed high throughput and scalability, while Redis performed best in speed for simple operations. CouchDB offered schema flexibility but lagged in complex query handling due to its reliance on MapReduce. **Discussion:** Relational databases proved most effective in structured data environments where strict access control and data integrity are essential. In contrast, NoSQL models provided better flexibility and scalability but lacked advanced features like views and joins. While some prior research favors No SQL, this study highlights the continued relevance of SQL models in structured, high-volume retail applications. **Conclusion:** The research concludes that relational databases, particularly Oracle APEX, are the most balanced and reliable option for online retail systems. Despite the appeal of No SQL models for specific use cases, SQL databases offer superior overall performance, especially in maintaining data consistency and supporting complex queries in dynamic retail environments.

Keywords: Database Model, E-Commerce, NoSQL, Retail Industry, SQL.

Correspondence:
Dr. Sathishkumar Veerappampalayam Easwaramoorthy
School of Engineering and Technology, Sunway University, Jalan Universiti, Bandar Sunway, 47500 Selangor Darul Ehsan, MALAYSIA.
Email: sathishv@sunway.edu.my

Received: 03-01-2025;
Revised: 24-02-2025;
Accepted: 14-04-2025.

INTRODUCTION

The application of e-commerce has gradually become a norm for people to meet the demands of obtaining both their necessities and desires. It bloomed during the COVID-19 pandemic when consumers had restricted mobility to purchase goods, offering optimal convenience through internet-enabled devices. As time progresses, it has opened more opportunities for business owners to expand their market reach internationally, allowing consumers a variety of products to purchase from. However, the trend of online shopping is foreseen to grow even bigger due to its active digitalisation on the economy (Akhmetova *et al.*, 2022). The dynamic changes in consumer behaviours also demands for advancements in technologies and the emergence of new business models (Klemová, 2024). This calls for an increasing need for robust and scalable database management systems to manage these vast amounts of data and digital transactions happening within the retail and e-commerce industries.

Significance of Data in Retail and E-commerce

With the emergence of big data and the concept of the 5 V's (Volume, Variety, Velocity, Veracity, and Value), big data analysis in retail and e-commerce businesses has become a crucial area of study. According to research by Niranjan *et al.*, (2016), a data-driven approach, also known as Data-Driven Decision-making (DDD), is increasingly important for retail and e-commerce businesses. Given the massive amounts of

DOI: 10.5530/irc.2.1.6

data generated by online transactions (retails purchases via e-commerce and m-commerce), DDD enables companies to make informed decisions that can significantly impact their operations and success in the industry.

Utilising DDD, retail companies and those engaged in e-commerce can make more informed decisions by providing insights into customer behaviour, preferences, and trends (Niranjan *et al.*, 2016). This enhances customer experience, increases personalization and targeted marketing, as well as optimizes operations for inventory management. Additionally, research by Kameswari *et al.*, (2024) supports this by emphasising the role of data in the retail industry, encompassing data sourcing, data analysis, data storage, data security and privacy, as well as real-time data processing, and highlighting how such business utilize data to boost their income (increased sales).

Data can be leveraged as a competitive advantage, allowing businesses to predict trends and forecast demands, and according to Trachim, (2024), e-commerce companies compile huge datasets, including evaluating data from previous years. These datasets can be broken down to transactional data, such as sales and transaction records, and non-transactional data, such as customer data (customer information, behaviour, references, product data (product catalogues), inventory data (inventory management), and order data (order fulfilment, logistics, supply chain management) (Trachim, 2024).

Challenges in Data Management

Given the large amount of data, handlings such large volumes of diverse data can make data management for retail and e-commerce companies challenging. These challenges, as indicated by Smith, (2024), may include data silos, data synchronization, data quality and governance issues, and data security and privacy issues. It is crucial as it determines the businesses' capability to leverage these vast amounts of data for empowering strategic decision making (Jaleel and Abbas, 2020). It is important for these companies to ensure fast, reliable, and secure data access, especially in adapting to the dynamic nature of e-commerce trends and customer behaviours.

Not only that, but data security is also a rising concern e-commerce business. Malicious attacks are constantly occurring on e-commerce platforms with intentions of stealing private customer information (Jamra *et al.*, 2020). This demands businesses to employ robust technologies that can safeguard customer data and comply with privacy regulations (Smith, 2024). Hence, this highlights the significance of data in retail and e-commerce, thereby reinforcing our research objectives on the importance of selecting the right database model for these industries.

Research Objectives and Questions

The aim of this research is to discover the suitability of the five database models (Relational, Graph, Wide Column, Key-value, Document) in application to the retail and e-commerce industry. Particularly, the literature seeks to understand the performance and the strengths and weaknesses of each database model in handling various aspects of retail and e-commerce functionalities, such as customer relationship management, order processing, product development, and inventory management. With this purpose in mind, the study poses the following research questions:

How do different database models impact data creation, manipulation, retrieval, access control and integrity?

Which database model provides the best performance for high volume transactions and operations in retail and e-commerce?

As a result, findings of these research questions with demonstrations of scenarios applied will enable retailers and e-commerce business owners to gain insights into choosing the best database model to optimize their business operations and boost customer satisfaction.

LITERATURE REVIEW

Database Management Aspects

Data Creation

Data creation refers to the process of creating and defining data structures in a database. This process starts with the design of the database schema, which is the blueprint for the database. Schemas define the types of databases objects, also known as tables and entities, the relationship among these objects such as foreign key referencing and relationship multiplicity, and the constraints imposed on the tables such as integrity constraints (Liu and Özsu, 2018). Depending on the type of database model, the definition of each schema will differ slightly as the nature of NoSQL schemas are typically more flexible (Nayak *et al.*, 2024). Following the completion of the schema, data entries will be inserted into each table. By defining the schema before populating the table, data integrity is maintained.

Data Manipulation

Data manipulation is the update, insert, or delete actions carried out when handling data. These processes are formally referred to as the Create, Read, Update, and Delete (CRUD) operations in database management (Gudivada *et al.*, 2018). Carrying out data manipulation can help ensure the accuracy of data. Different database models will use different data manipulation methods to modify the table contents. Consistently maintaining a database through data manipulation will product records that are reliable and efficient for daily operations.

Data Retrieval

Data retrieval is said to play a major role in achieving high performance and efficiency. It incorporates using a number of commands to request data to get the information from the database. Several database models enable different query languages and techniques to be used in a system. For example, relational databases are written in SQL, while some NoSQL databases, such as wide-column stores, use their own language, Cassandra Query Language (CQL). Some of the NoSQL databases do not follow a schema, allowing users to query and get rows and particular columns or a sequence of columns. As the size of the databases grow, indexes enable efficient queries resulting in fast data access (Yusof, 2017). Efficient data acquisition and retrievals guarantees that the required data is made available for use by the applications and users at required times.

Access Control

Access control enables only those personnel who are permitted to access or modify data to do so, thus, protecting data from the outsiders. This is done through mechanisms like Role-Based Access Control (RBAC), Discretionary Access Control (DAC) and Mandatory Access Control (MAC). Users are defined in RBAC based on the role model where each role relates to privileges that are the actions the users can carry out in the database (Le *et al.,* 2023). Another is the Data Access Control (DAC) enables data owners to set their access control policies for their data and the Mandatory Access Control (MAC) ensure that access policies of data are in compliance with the security labels assigned to the data and users. Strong access control measures need to be put in place to minimize the risks associated with privacy and security of information and policy noncompliance.

Data Integrity

Data integrity is essential as it ensures that the data remains accurate, consistent and reliable throughout its entire life cycle (Cai and Zhu, 2015). In other words, the data must remain credible and cannot be compromised or corrupted in any way Duggineni, (2023). conducted research to identify measures that can ensure data integrity and prevent unauthorized access. One of the methods is to verify input data to ensure the accuracy of the data before it enters the database. Additionally, any form of unauthorized data modification can be prevented through access controls found within a DBMS. It is important for an organization to implement these methods and ensure that their data is trustworthy and secure.

General Concepts of Data Models

What is a data model?

A data model is a simplified diagram or flow chart that shows the entities, attributes and the relationships of a software system (Gillis *et al.,* 2024). Data models contain various useful information about the data stored, such as the definition, type and format of the data, to support the development of the information system (Palanisamy *et al.*, 2020). Through the data modelling process, various data validation techniques have been performed to ensure that the data stored in the data model is consistent and accurate. This is especially useful as it provides a standardized view of the data to all stakeholders involved in the business process. Thus, data models can be used to facilitate communication between the business and IT professionals as it provides a common reference point for discussion and setting the business requirements.

How are data models represented?

Data models can be represented by three common types which are known as conceptual data model, logical data model and physical data model. Each one of these types represent different phases during the data modelling process.

Conceptual data model

A conceptual data model is a high-level model that provides an abstract view of the data. The purpose of a conceptual data model is for business stakeholders to understand the structures and relationships within the data. This model focuses on essential data that is important to the business and omits the finer details (Tupper, 2011). Conceptual data models typically use notations such as entities and relationships.

Logical data model

Logical data models provide a more technical and detailed view of the data. For example, an Entity-Relationship Diagram

Table 1: Summary Comparison between data models.

	Conceptual	Logical	Physical
Purpose	Show the structures and relationships within the data from a business point of view.	Provide a visual representation of the structure and relationship of the data elements needed to fulfil the business requirements.	Focus on designing data structures that optimize resources while maintaining an optimal database performance.
Level of detail	Abstract, less detailed	Detailed	Very detailed
Target group	Business stakeholders	Data architects, database designers	Developers
Elements	Entities, relationships	Entities, attributes, relationships, primary and foreign keys	Tables, columns, constraints and triggers

(ERD) would include the optionality and cardinality in a logical data model (Luisi, 2014). The goal of logical data models is to provide a visual representation of the structure and relationship of the data elements needed to fulfil the business requirements. Logical data models use notations such as entities, relationships, attributes, primary and foreign keys.

Physical data model

The physical data model is the most detailed model amongst the three data models. This is because the model shows how data is stored in a specific Database Management System (DBMS) (West, 2011). Unlike the conceptual or logical data model that use notations like entities and attributes, the physical data model includes the tables, columns and constraints (Hughes, 2016). This model focuses on designing data structures in a way that optimizes resources while maintaining an optimal database performance.

Table 1 shows a summarized table of the comparisons between data models to allow researchers to identify the optimal model for their specific needs.

How are data models related to logical database design and physical implementation?

In a logical database design, data models such as ERDs, define the entities, attributes and relationships involved. The data is organized in a logical manner such that the relationships are formed based on the business requirements. This stage prepares the data to be physically implemented in a DBMS. The physical implementation stage then translates the logical design into a real database structure, considering specific storage methods and optimization techniques to ensure efficient data handling and retrieval.

Data Models and their Characteristics (Creation, Manipulation, Retrieval, Access Control, Integrity)

Relational

A Relational Database (RDBMS), also known as an SQL database, is a logically structured database that comprises of a collection of relations (Harrington, 2009). RDBMS are most commonly used today as each data entry is stored in tables with columns to define the nature of the data (Bourgeois, 2014). This improves the visualization of the database as users can immediately extract the data entities and entries they want. Relational databases are also known for using structured approaches to allow data creation, data manipulation, data retrieval, access control, and data integrity. SQL language is used to manage operations related to the characteristics mentioned below (Oracle Australia, 2024).

Data Creation

To create data, a schema must first be defined. Entities are known as tables. Tables are created using *CREATE* command after defining the columns that represents an attribute. The relationships that link a table to another are also defined through *FOREIGN KEY* referencing. Lastly, constraints are given to each table to maintain data integrity. For example, integrity constraints or *PRIMARY KEY* constraints are applied to each table to ensure that each entry is unique. After the completion of these tasks, data records will be entered, making up the rows of the tables. Users can modify or remove the table at any time using *ALTER* and *DROP* commands.

Data Manipulation

Data manipulation involves the operations that modify the contents of each table. These operations include the *INSERT* command for entering new data, *UPDATE* command for modifying existing data entries, and *DELETE* command for deleting entries from the table.

Data Retrieval

This characteristic refers to the process of querying the database for specific data based on a user's specification. Data querying is performed using the *SELECT* command.

Access Control

Relational databases offer access control commands to restrict users from viewing or modifying certain tables, columns, or rows. This is done through the *GRANT* and *REVOKE* commands.

Data Integrity

Data integrity should always be maintained to ensure that the database remains consistent, accurate, and reliable. *PRIMARY KEY* constraints and *FOREIGN KEY* constraints are implemented to enforce data integrity and establish relationships between tables, which enforces the reliability of each data record.

Graph

Graph model databases like ArangoDB stores entities as collections, which are equivalent to tables in RDBMS (Belgundi *et al.*, 2023; https://arangodb.com/graph-database/). Within each collection (table), there are multiple documents, which are parallel to records in RDBMS. Documents can contain single or multiple attributes. Unlike RDBMS, ArangoDB does not require schemas, in the sense that each attribute must be defined, to build an entity. Documents can be directly entered into collections. There are two types of collections in ArangoDB, which are document (vertex) collection and edge collection.

Data Creation

Document collections, also known as vertex collections, are analogous to data tables in RDBMS. Vertex collections store the collections (tables), which are the primary data elements of the database. Keys are assigned within collections, such as *_key,*

_id, and *_rev*, and are equivalent to primary key referencing, foreign key referencing, and document revision maintenance respectively. Meanwhile, edge collections are analogous to data relations in RDBMS. Every relation that links one collection to another are stored in edge collections.

These vertex and edge collections are represented as graphs in graph-based models like ArangoDB. Documents from both vertex and edge collections are represented as a node/vertex in a graph. Meanwhile, the relations between each collection are represented as edges in the graph. These edges will use _from and _to components to identify the direction of relation. Storing databases as graphs offers a user-friendly approach due to its intuitive representation of data and relationships.

Data Manipulation

Like other databases, each graph database has its unique query language. For ArangoDB, ArangoDB Query Language (AQL) is used (https://docs.arangodb.com/3.11/aql/data-queries/). Users can modify existing collections and documents by using 5 different modification queries. Firstly, *INSERT* allows users to enter new documents into a collection. *UPDATE* modifies existing documents in the collections, while *REPLACE* complete replaces existing documents with new ones. *UPSERT*, a combination of update and insert, is used to insert a new record if the specified unique key does not exist or update the existing record if it does. This is an efficient way to check for the existence of the document without explicitly checking for it. Lastly, *REMOVE* deletes documents from a collection.

Data Retrieval

Data retrieval on ArangoDB is carried out with the *RETURN* command.

Access Control

Graph-based models like ArangoDB assigns user roles from the Access Controls page in the navigation menu (https://docs.arangodb.com/3.11/aql/data-queries/). Pre-defined roles created by ArangoDB and group related permissions are available for use. Users can also create new roles for other restrictions.

Data integrity

Data integrity on graph-based models like ArangoDB are reinforced with edges and _key commands. This ensures rules governing relationships between vertices are strictly followed.

Wide-Column

Wide column stores also known as column family stores are implemented to address the needs of distributing massive amounts of data across different systems (https://www.scylladb.com/glossary/wide-column-database/). They group data by tables, rows, and dynamic columns where every row can contain a different number of columns. This structure has the advantage of flexibility in their storage and thus makes it possible for wide column stores to be applicable in applications that are represented by changing models of data. One main example that can be outlined is the AstraDB that is a serverless vector-based database developed on the basis of Apache Cassandra (https://python.langchain.com/v0.2/docs/integrations/vectorstores/astradb/). A kind of wide-column store which is highly scalable and is primarily designed for the distributed data centre. It is scalable as it deals with petabytes of data and has high availability because of the distributed nature of the system. Their schema flexibility makes them suitable in applications such as real time analysis, time series data analysis and data warehousing since they perform well in data manipulation as well as retrieval of data (Akhtar, 2024).

Data Creation

In a column store like AstraDB, data is stored in column families, where a column family assumes the role of a table in a relational database. Each column family can contain any number of columns, and it is possible to add a new column family dynamically, without extending the length of the rows (Akhtar, 2024). This characteristic makes it possible to store sparse data that are characterized by many empty locations with minimum storage space. To create the column family in AstraDB, the operation used is CREATE TABLE and the columns are created along with their data types. They also define primary keys, which uniquely identify each row in the table.

Data Manipulation

Data manipulation in wide column stores refers to the operation of inserting, updating, and deleting data within the column families. AstraDB performs these operations using CQL, which stands for Cassandra Query Language (Vohra, 2016). The INSERT command is used to insert new rows, the UPDATE command is used to update the existing rows, and the DELETE command is used to delete rows or a specific column from a given row. These commands make it possible for the database to have high write throughput per second, and at the same time, have high performance.

Data Retrieval

Data retrieval in wide column stores is optimized for read-heavy workloads. AstraDB uses the SELECT command to query data. The schema-less nature of wide column stores allows for flexible queries, where users can retrieve entire rows, specific columns, or ranges of columns (Vohra, 2016). Indexes can be created to improve query performance, ensuring efficient data access even in large datasets.

Access Control

Another feature in wide column stores is the ability to control access to the data, thus preventing anyone who is not authorized from accessing the data. From the study conducted on AstraDB, it was observed that the access control model implemented was the role-based access control (RBAC) model (DataStax, 2024). Users have privileges that may include viewing or writing onto specific column families of the key space. These include the GRANT and REVOKE statements which help in enforcing the permissions in order to limit the users who execute some tasks (DataStax, 2024).

Data Integrity

In wide column stores, the data integrity is ensured by the consistency levels and replication. AstraDB as a database supports different levels of consistency, which means that users can set priorities to converge on consistency instead of availability and vice versa (Kitterman, 2024). To provide protection against loss of data and to make the data more durable, it is replicated over the nodes. Primary keys and unique constraints also play their role in keeping the record free from redundancy and any adverse effects on the accurate collection of data.

Key-Value

Key-value stores which also fall under the category of NoSQL databases are the simplest, and the data is stored in the form of key-value pairs. It gives an extremely high data access speed, and it can be useful in any application that requires fast lookups such as caching and session handling (https://aerospike.com/what-is-a-key-value-store/). One of the examples of key-value stores is Redis. It is a concern system implemented as a key-value store, which is suitable for various purposes due to its efficiency. It is compatible with almost all types of data and has many extras such as transactions, pub/sub, and Lua scripting. It is widely implemented in caching, probably in session handling, and in real-time analytics applications in which real-time data is vital (https://aerospike.com/what-is-a-key-value-store/).

Data Creation

The data in key-value stores such as Redis is stored in the form of a key-value pair where the key is unique and is associated with a specific value. It can be any type or kind of data inclusive of strings, hashes, lists, sets, sorted sets, etc. For the creation of data, the SET command is utilized, before which a value is assigned to a key (https://redis.io/nosql/key-value-databases/). Directions are rather unique labels in the database while extents are linked with

Table 2: Summary of SQL and NoSQL databases.

Database Model	Structure	Schema	Query Language	Examples	Characteristics
Relational	Tables (relations) with rows and columns	Strict	SQL	Oracle, MySQL, PostgreSQL	Data Integrity: Enforced through constraints and keys. ACID Compliance Handles complex queries and joins.
Graph	Nodes (entities) and edges (relationships) forming a graph	Flexible	Cypher, Gremlin, AQL	Neo4j, Amazon Neptune, ArangoDB	Efficient for complex relationships Optimized for traversing relationships. Schema-less or schema-light.
Wide-Column	Tables with rows and dynamic columns	Flexible	CQL	Apache Cassandra, HBase, AstraDB	Designed for horizontal scalability. Optimized for write-heavy operations. Efficient for large datasets with varying data points.
Key-Value	Key-value pairs	Schema-less	Simple API calls	Redis, Amazon DynamoDB, Riak	Simple and fast for read/write operations. Ideal for session management, user profiles, caching Easy to implement and use.
Document	Documents (JSON, BSON, XML)	Schema-less or dynamic	MongoDB Query Language (MQL)	MongoDB, CouchDB, Amazon DocumentDB	Supports complex and nested data structures. Optimized for read-heavy and complex queries. Designed for horizontal scaling via sharding.

these directions without reference to a pre-designated structure. Redis's flexibility makes it suitable for a wide range of applications because it allows it to handle different data types and structures with ease.

Data Manipulation

Manipulation in key-value stores signifies creating, updating, and deleting key-value pairs. SET is one of the commands that allows the setting or updating of a key-value pair while DEL is a command used in the removal of a key as well as its value (Kaur and Kaur, 2018). These bench operations are highly efficient and prompt with the result that key value stores perform well in read and write operations. The high-performance demands of modern applications depend heavily on these techniques' efficiency. Redis uses advanced techniques to improve the speed and effectiveness of these processes, including instruction-level parallelism and cache-friendly hash index structures (Kaur and Kaur, 2018).

Data Retrieval

Similarly, data retrieval is quite easy in key-value stores as there is no complex model as presented in other databases. GET is one of the interrogation commands whereby Redis helps in retrieving the value of a specific key typed in. Due to their high efficiency, key-value stores are used in applications in which the speed of data access is paramount (Kaur and Kaur, 2018). Redis's effective caching and in-memory data structures improve retrieval performance significantly. Redis can be used for use cases including real-time analytics and caching, which require quick access to data and low latency.

Access Control

The authorization which takes place on the key-value stores, can be achieved mainly through the identification of a user and the functionalities that have been granted/restricted. For instance, Redis has a configuration file when it comes to the authentication of the roles of users (Liu, 2019). These roles can specify which command is available to the given user, and thus they can deny access to the required information. This access control system is very important when it comes to ensuring that the private information is secure and only those who are allowed can perform some operations. These security features complement Redis's security features such as RBAC and encryption activities to boost the security of the platform.

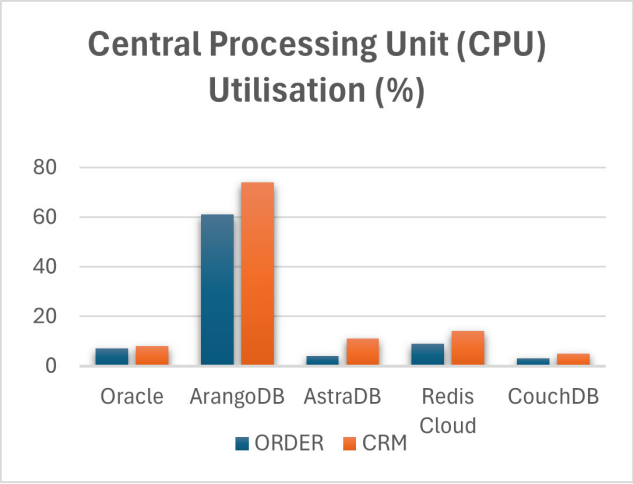


Figure 1: CPU Performance for Data Creation Across All Database Models.

Data Integrity

It is important to maintain the consistency of the data in key-value stores and this can be achieved through persistence and replication. The advantages of Redis include a snapshotting with a fast data persistence and Append-Only File (AOF) (Liu, 2019). Replication also assists in the copying of data from one instance to another hence its availability in case of failure. Redis cluster availability and dependability are further improved by innovative techniques like the dynamic cuckoo filter system.

Document

A NoSQL document-oriented database such as Apache CouchDB stores data in JavaScript Object Notation (JSON), Binary JSON (BSON) or eXtensible Markup Language (XML) document (Sadalage and Fowler, 2012). Document-oriented database stores their data in a collection of documents (Olivera, *et al.*, 2019). This is completely different from RDBMS where the data is stored in a table with rows and columns. Document databases are known to be flexible and scalable. This is because they do not require a fixed schema. In other words, different documents in a collection can contain different types of data and each document can have a different data structure within a collection (Patel, 2024). Additionally, new data can be added to the document without affecting the pre-existing documents.

Data Creation

For CouchDB, a database can be created by selecting "Create Database" from the Fauxton interface. Fauxton is the default administration interface used by CouchDB that allows users to manage their databases and documents. Once a database is created, the "New Doc" function will allow the user the create a new document (https://docs.couchdb.org/en/stable/intro/tour.html). Another method is through the client URL (cURL), a command line tool that allows access to the HTTP protocol. A database can be created through this method by using the PUT operation. Similarly, the PUT operation can also be used to create a new document (https://docs.couchdb.org/en/stable/intro/curl.html). Each document in CouchDB is uniquely identified by its _id field. This _id field is used as a primary key to identify the different documents. As a fixed schema is not needed, a new field can be added within a document by using the editor provided in CouchDB (https://docs.couchdb.org/en/stable/intro/tour.html).

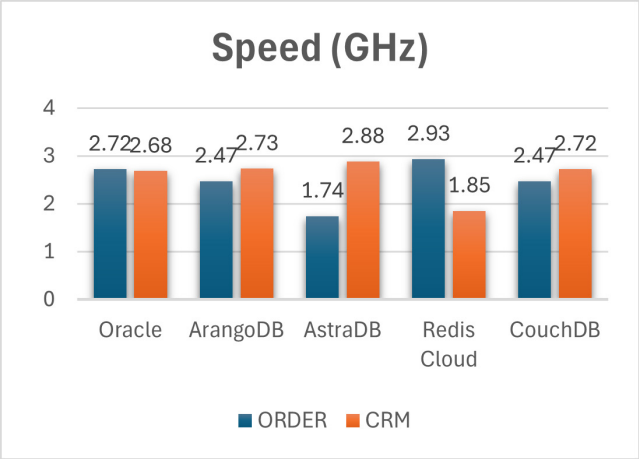


Figure 2: Speed Performance for Data Creation Across All Database Models.

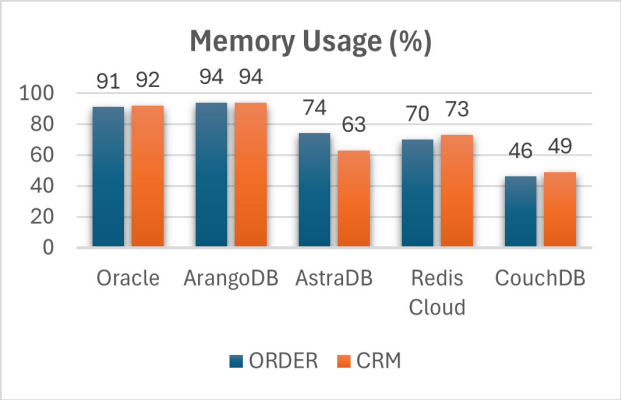


Figure 3: Memory Usage for Data Creation Across All Database Models.

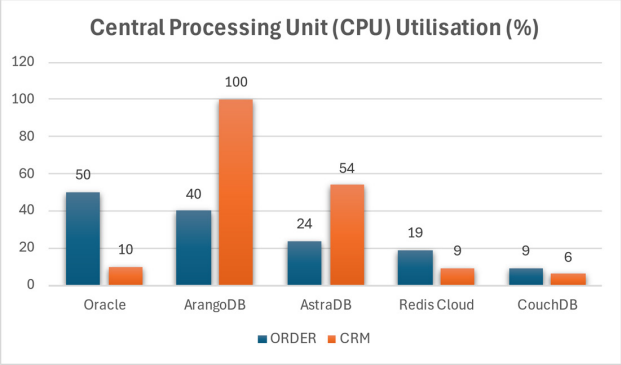


Figure 4: CPU Performance for Data Manipulation Across All Database Models.

Data Manipulation

Data manipulation techniques to update and delete the data can be done using the Fauxton interface or via cURL. As for updating the document, the PUT operation can be used to send a HTTP request to the server (https://www.tutorialspoint.com/couchdb/couchdb_updating_a_document.htm). Deleting a document works in the same manner but it uses the DELETE operation (https://www.tutorialspoint.com/couchdb/couchdb_deleting_a_document.htm). A notable feature that document-oriented database can do is to attach files like pictures to the document. This can be done by using the PUT operation and including the relevant information such as the document ID, the name of the database and the name of the attachment (https://docs.couchdb.org/en/stable/api/document/attachments.html).Data RetrievalData retrieval in CouchDB can be done through the cURL utility. The HTTP GET command can be used to retrieve the data within a document by specifying the unique _id in the URL (https://docs.couchdb.org/en/stable/intro/curl.html).Access ControlThere are different methods in CouchDB to give a user permission to access certain databases and documents. The straightforward method is through the Fauxton interface where users would have to click on the "Permissions" section and set it to their preferences.As for the cURL method, there are basic authentication measures in place. By adding the admin's username and password in the URL, CouchDB can determine if the user is authorized to access the document. There are two main roles on CouchDB, namely admin and member. Admins have full control over database that they are administrating and are able to change users' roles while members can read, create and modify document but are unable to modify the designs of a documents such as the views and indexes. The permissions of a database can be modified through the command line by editing the security document within the database (https://docs.couchdb.org/en/stable/intro/security.html).Data IntegrityCouchDB has several data integrity controls in place, namely, Multi-Version Concurrency Control (MVCC) and data validation. Traditional database can only be modified by one person at a time, the person who is editing holds the "lock" and everyone else must wait for the "lock". This method is known as locking. On the other hand, CouchDB uses MVCC, a method that creates multiple versions of the document and allows simultaneous access to the database while maintaining full speed. This method ensures that data is consistent as it prevents conflicts. Moving on, data validation is

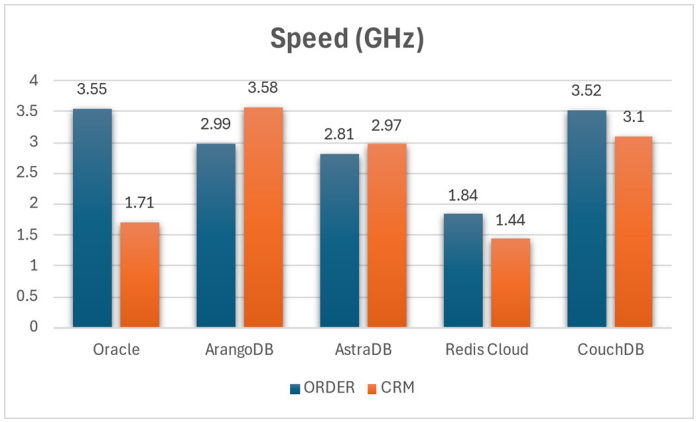


Figure 5: Speed Performance for Data Manipulation Across All Database Models.

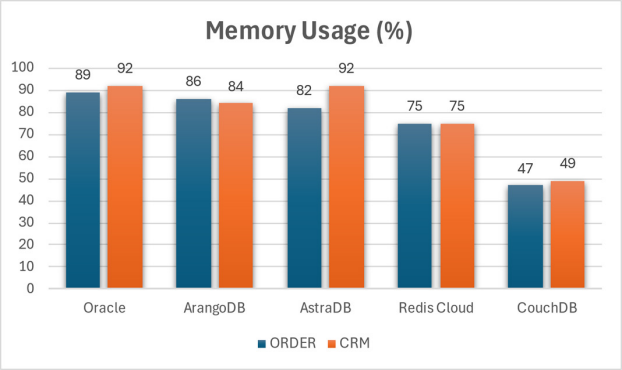


Figure 6: Memory Usage for Data Manipulation Across All Database Models.

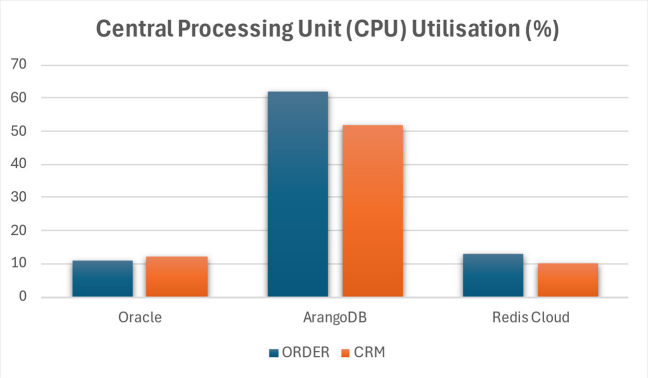


Figure 7: CPU Performance for Data Retrieval Across All Database Models.

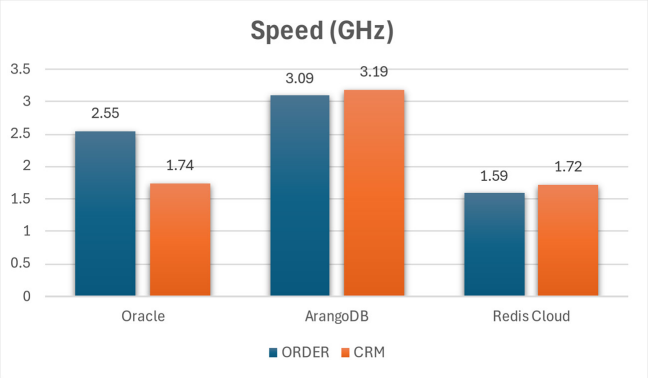


Figure 8: Speed Performance for Data Retreival Across All Database Models.

performed by CouchDB through the use of JavaScript functions. This process ensures that only valid data modifications are allowed, thus maintaining the overall data quality and consistency (https://docs.couchdb.org/en/stable/intro/consistency.html).After continuous research synthesis, Table 2 is created to depict the distinct characteristics and features summarized for easier readability.

Reviewing Existing StudiesThere are many existing studies that analyze the compatibility of different database models in the retail industry. For example, retail stores with a lot of data to

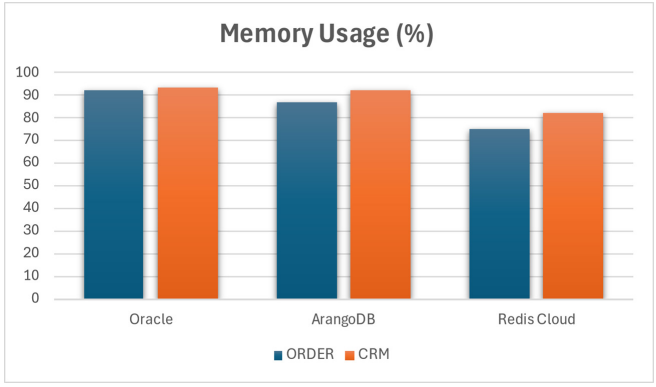


Figure 9: Memory Usage for Data Retrieval Across All Database Models.

process would be more suited to use document databases such as MongoDB instead of traditional SQL databases (Ramesh *et al.*, 2016). The research states that NoSQL databases show higher flexibility and scalability, which makes them the preferred database model for online retail industries. This is because document databases provide more flexibility in schema implementation as schemas are not necessary. In fact, document databases, like other NoSQL databases, normally operate by populating the tables before checking for constraints (Soares, 2015).Additionally, other research concludes that graph databases like ArangoDB benefit the e-commerce industry. This is because graph databases are built to handle large volumes of data, and they can manage the relationships between the data effectively through building knowledge graphs. Unlike the traditional relational database that use tables and joins, graph databases use nodes and edges. This makes it easy to process complex queries in real-time (Yeung *et al.*, 2018). This is supported by Thakare *et al.*, (2023) whose study revealed that NoSQL databases are suited to handle and analyze large volumes of data generated by e-commerce platforms.Furthermore, wide-column and key-value stores are essential in retail and commerce fields due to the scalability of data and usage of efficient structures in applications. Wide-column databases, like Astra DB (Cassandra), are useful for huge datasets keeping in mind the scalability and distribution of the data among many nodes (Agrawal *et al.*, 2024). These databases are well suited for applications where raw processing power is needed, for instance, in real-time analysis of e-commerce platforms. Essentially, key-value stores such as Redis are simple and fast, which can be appropriate in caching and session use cases in retail applications. These models enable the variation of the data that is stored and their accessibility, which is essential in e-commerce data such as product lists, customer transactions, and orders. According to research, wide-column and key-value databases can also greatly improve efficiency in cases of large-scale, reads, and real-time data processing (Reddy, 2024).In short, most existing studies conclude that NoSQL databases are more suited for large volume retail databases. This study aims to add on to these current findings by identifying the

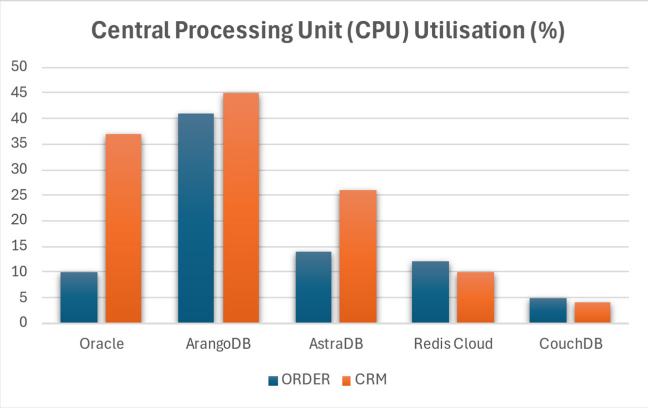


Figure 10: CPU Performance for Access Control Across All Database Models.

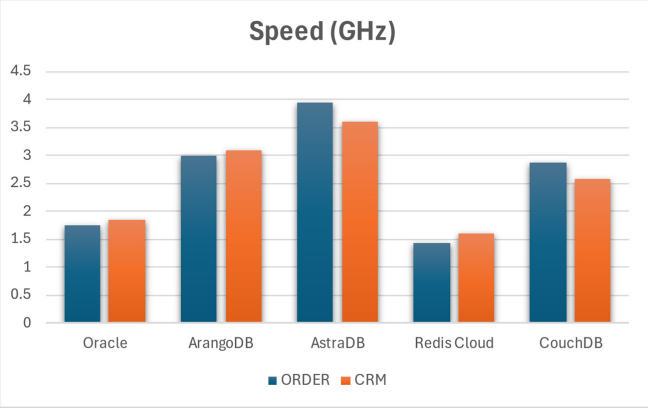


Figure 11: Speed Performance for Access Control Across All Database Models.

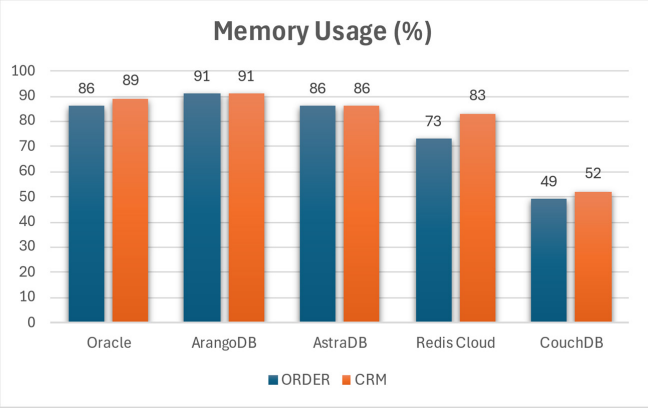


Figure 12: Memory Usage for Access Control Across All Database Models.

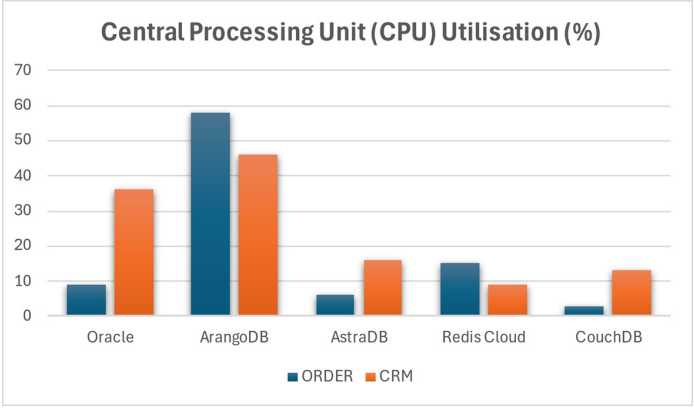


Figure 13: CPU Performance for Data Integrity Across All Database Models.

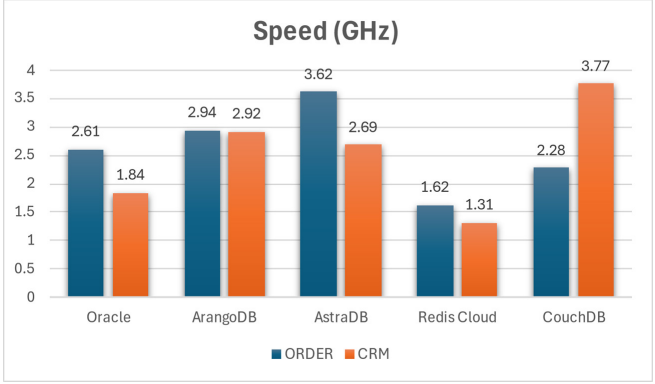


Figure 14: Speed Performance for Data Integrity Across All Database Models.

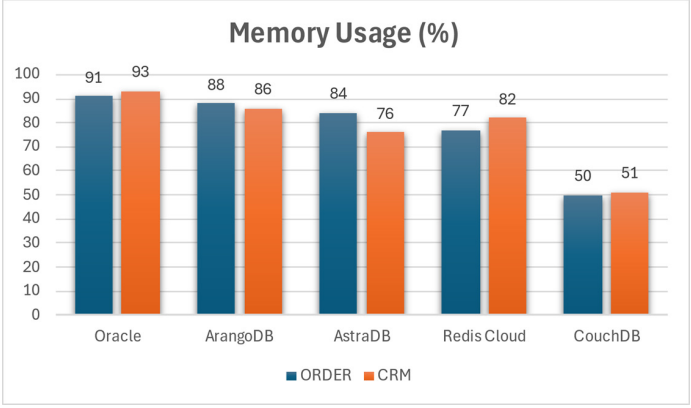


Figure 15: Memory Usage for Data Integrity Across All Database Models.

optimal database for online retail industries.MethodologyIntroduction

Initially, an outline was created to provide structure to the introduction. This outline splits the introduction into four sections that focused on different aspects. These four sections include the background of e-commerce and retail, importance of data in retail, challenges of data management and the research questions of this study. The team gained key insights for these topics by reading various database management academic papers that were relevant to the e-commerce and retail industry. Other sources that were referred to by the team include journal articles and books. Key points from the academic materials were extracted and compiled to form a draft for the introduction. These points were also supported by relevant statistics and real-world examples found in the materials. The research questions and objectives were decided after thorough research and discussion among team members, ensuring that they are directly relevant to the research title. Multiple revisions were made to ensure clarity, consistency

and adherence to academic writing standards throughout the writing process.

Literature Review

Literature analysis was carried out to identify and understand the different model types. Firstly, researchers identified the useful research papers by browsing through online academic databases such as Google Scholar and Tun Hussein Onn Sunway Library. Researchers then used specific keywords such as "Database Model Comparison" and "Retail" to obtain the most relevant journals that aid in the comparison analysis for this research paper. These journals were then checked for their credibility by ensuring that they have been peer-reviewed and are not too dated. After that, researchers carefully read through every article to identify the common and contrasting themes mentioned throughout the papers. These findings were synthesized and documented in the paper later on. This thorough investigation helped highlight the differences between SQL and NoSQL databases, which helped researchers gather more comprehensive insights into the how different database models affect CRUD operations, access controls, data integrity and other aspects.

Results

Sample databases were created after reviewing data structures and Entity Relationship Diagrams (ERDs) that focused on order processing and customer relationship management from existing studies. The selected database models are as follow: Oracle APEX for Relational, ArangoDB for Graph, AstraDB for Wide-Column, Redis for Key-Value, CouchDB for Document. Each database approximately had 200 - 1000 records on each table that will be applied and inserted using JSON files and the command languages of the respective database model to compare between the different database models. Once the databases were established, queries narrating various scenarios were made and implemented to evaluate the performances of each database model in applications [Appendix A]. Apart from capturing the performance indicators provided by the database model, performance indicators from "Task Manager" on laptops were also recorded and summarized into concise graphs to contrast each database model in various aspects.

Discussion and Conclusion

The discussion section consists of a comparison analysis between the five database models. It analyzes the key aspects such as data creation, manipulation, retrieval, access control and data integrity of each database model and how they compare to one another. The aim of this discussion is to identify the strengths and weaknesses of each database model under different scenarios that are relevant to the e-commerce and retail industry. Additionally, the discussion answers the research questions introduced at the beginning of this study by reflecting on the findings gained from the comparative analysis and correlating it with the results obtained from the experiment.

The conclusion compares the result of our study to other existing studies that were mentioned in the literature review. It highlights how our results either contradict or support the previous studies and explores the reason behind this. This section also mentions the insights gained and lessons learned by our team during the whole process. The limitations of the study as well as potential future avenues of our research will also be mentioned in the conclusion.

RESULTS

The following results depict how each database model executes data creation, data manipulation, data retrieval, access controls, and data integrity in different scenarios. The tables were populated with over 1000 records. These records were randomly generated using a Python program [Appendix B]. The constraint schema used are also documented in this report [Appendix C].

The first scenario-Order Management Database, is used to reflect real-world situations where thousands of data are used in a customer order database. This database consists of 4 tables, which are the User, Goods, Order, and OrderGoods table. This database was adopted from a prior case study by Wei and Zhang, (2018).

The second scenario used in this report is a Customer Relationship Management (CRM) Database. This database is used to depict the data management for the reviews and complaints given by online customers. It consists of 7 tables, which are Company, Office, Coworker, Person, Deal, History, and Document. These tables were adopted from previous research in Sweden (Winberg and Zubac, 2019).

Comparison of Performance Across Different Queries in Different Applications

The performance of each database model is based on CPU utilization, speed, and memory usage. The full screenshot of the performance is documented in the appendix [Appendix D-H]. Performance in terms of seconds taken for server platform to execute query is not documented as not all database models have that feature [Appendix I]. Hence, for consistency, only these 3 features are used to assess the performance.

Data Creation

Figures 1-6 illustrate how relational databases are leading in terms of various performances. They highlight how Oracle utilizes low CPU space, high speed, and higher memory usage.

Data Manipulation

For CRUD operations, it is clear that Oracle again shows better performance overall compared to the other database models, as

Table 3: Summary discussion of key aspects of SQL and NoSQL databases.

	Relational	Graph	Wide Column	Key-Value	Document
Data Creation	Strength: Easy bulk data upload with CSV, strong data integrity. Weakness: Challenging schema adjustments.	Strength: Easy bulk data creation with JSON, flexible schema. Weakness: Limited schema validation.	Strength: Easy data loading, high write throughput. Weakness: Complex initial setup.	Strength: Fast batch processing with HSET. Weakness: No data validation measures.	Strength: Easy bulk data creation, flexible schema. Weakness: Potential data inconsistencies.
Data Manipulation	Strength: Easy data insertion and updates. Weakness: Performance varies with dataset size.	Strength: Flexible querying with AQL. Weakness: Steep learning curve for AQL.	Strength: Fast data creation, supports basic CQL operations. Weakness: Limited complex manipulations.	Strength: Fast retrieval with HSET. Weakness: Limited to key-specific operations.	Strength: Supports MapReduce for queries. Weakness: Must update entire document for changes.
Data Retrieval	Strength: Handles complex queries. Weakness: Performance may decline with large datasets.	Strength: Supports joins, subqueries with AQL. Weakness: Requires performance tuning for large datasets.	Strength: Fast read operations. Weakness: Lacks support for complex queries.	Strength: Fast retrieval using indexes. Weakness: Limited relational linking.	Strength: Supports views with MapReduce. Weakness: Cannot join databases directly.
Access Control	Strength: Strong control through SQL Command Line. Weakness: Administrator access required.	Strength: User-friendly access control. Weakness: Limited access control options.	Strength: User-friendly access control via web interface. Weakness: Keyspace-level permissions only.	Strength: User-friendly access control. Weakness: Limited to predefined roles.	Strength: Flexible access control via design documents. Weakness: Roles must be user-assigned.
Data Integrity	Strength: Enforces data integrity with constraints. Weakness: Inflexible schema constraints.	Strength: Strict validation rules when defined. Weakness: Potential inconsistencies with schema-free setup.	Strength: Enforces primary key constraints. Weakness: Lacks support for some constraints.	Strength: Unique entries. Weakness: No schema structure, potential inconsistencies.	Strength: Customizable _id field for uniqueness. Weakness: Potential inconsistencies with schema-free setup.

shown in Figures. Relational databases use lower CPU storage; have higher processing speed, and high memory usage.

Data Retrieval

Figures 7-9 display Oracle and ArangoDB as similar performing databases for data retrieval. This is because ArangoDB shows inefficient CPU usage, with higher speed, whereas Oracle uses less CPU usage and lower speed. Other NoSQL databases such as column (AstraDB) and document databases (CouchDB) have no performance available as they do not support view functions, which is what the query contains.

Access Control

Figures 10-15 highlight ArangoDB as the database model with the better performance overall. This is because of the efficient utilization of CPU, the higher processing speed, and the comprehensive usage of memory storage.

Data Integrity

Figures display Oracle as the better performing database models due to its lower CPU utilization, and higher memory usage. Even though it has relatively lower speed compared to the other database models, it still exceeds in performance from an overall view.

Table 4: Summary discussion of key aspects of SQL and NoSQL databases.

Database Models	Data Creation	Data Manipulation	Data Retrieval	Access Control	Data Integrity
Relational (Oracle APEX)	Bulk Data Creation and Variation of Data Uploads: Supports large dataset insertions (> 5000 records) using csv files or copy pasting the data itself. Using SQL Scripts is also possible to insert data values. Schema definition and table creation also comes with built-in constraints to ensure data integrity. Schema Management: Highly flexible when providing schema definition. However, making adjustments or additions to rigid schemas will be slightly complex. Alternatively, predefined constraints help maintain data accuracy but can limit schema flexibility.	Ease of Use: Inserting and Updating data through SQL Commands can be easily executed due to its user-friendly interface and familiarity. Performance Variability: Execution of SQL Queries should generally be efficient, but it will still depend on the size of the dataset being analyzed as larger datasets tend to reduce performance in certain circumstances.	Query Complexity: Oracle APEX (SQL Commands) allows complex queries, including joins, aggregations, and subqueries. This enables a large scope of data retrieval operations. Performance Considerations: Oracle Apex is known for its high performance. However, it is also important to note that it can decline with very large datasets or highly complex queries, which may require further optimization and adjustments.	Permission Management: Permissions can be managed, and GRANT commands can be executed as long as administrative access is provided. An alternative method is to use SQL Command Line if there is not any administrative access given. Oracle APEX provides detailed control over user roles and access levels. Access (Level of Detail): Detailed role-based access control is possible. Nonetheless, managing permissions for specific tables or operations is usually more complex without proper administrative access.	Constraint Enforcement: Data integrity can be maintained through primary and foreign key constraints. Although they are strong, schema constraints can be rigid and need to be executed with caution. Schema Constraints: Constraints such as NOT NULL and UNIQUE are enforced at the schema level. This will guarantee data validity but also it may limit flexibility in schema design.
Graph (ArangoDB)	Schema-free document creation: Allows collections (tables) and documents (rows) to be created without predefining a schema, allowing users to add new fields to the document anytime. User-friendliness: Simple interface for creation of collection through point-and-click method, eliminating the need for AQL coding. Limitation: Bulk data import lacks visibility in terms of performance and success status	Querying and manipulation complexity: Data manipulation is handled directly through AQL, which has a steep learning curve for users unfamiliar with it. Any insertion and updates of records require specific AQL queries. Capabilities: Supports comprehensive data manipulation through AQL to perform complex operations. Also provides robust features for managing graph data to handle complex relationships and hierarchies.	Efficiency of complex queries: Excels at handling complex queries especially those involving graph data structures. However, the absence of a dedicated view creation feature requires the need of users to construct queries to simulate views. Performance tuning: Proper indexing and query optimization are crucial for maintaining performance, otherwise performance may degrade if not properly tuned.	Access control options: Provides user-friendly interface for managing access controls, without needing AQL commands. Can be granted at the database level with three main options-administrative, access and no access. Limitations: Access controls are applied to the entire database rather than individual collections. The lack of CRUD specific permissions also prohibits users from having selective access to perform specific operations.	Data consistency and schema enforcement: Allows the creation of optional schemas to enforce data integrity rules such as NOT NULL and UNIQUE constraints. Constraints: Primary keys must be explicitly defined, and they are enforced using the "_key" attribute, and the NOT NULL constraint is defined as "required". The absence of support for SQL-like constraints such as foreign keys and triggers requires complex integrity rules to be managed at the application level.

Database Models	Data Creation	Data Manipulation	Data Retrieval	Access Control	Data Integrity
Wide Column (AstraDB)	Bulk data creation: Allows easy bulk data loading/ creation through CSV file uploads or data migration from Dynamo DB and allows efficient handling of large datasets-i.e. has the ability to load 1000 records in 4.65 seconds. However, comes with the limitation that it does not provide real-time performance metrics or detailed status updates during the data load process. Schema-flexibility: High schema flexibility allows new columns to be added dynamically without altering existing rows (+ no limitation in columns). Requires schema definitions-must define tables with primary keys, column names, and data types.	Ease of data manipulation: Data manipulation is performed using CQL (Cassandra Query Language), which provides SQL-like syntax, making it relatively easy for users familiar with SQL to perform basic CRUD operations, but not complex operations which needs to be handled at the application level. Performance: Performance is optimized for high-write throughput, ensures low-latency writes and reads which is suitable for applications with heavy data, Application-level logic: Requires application-level logic to handle complex scenarios that CQL cannot directly support.	Efficiency: Optimized for high scalability and efficient data retrieval (for simple queries). However, complex query operations such as joins or views are not supported and is required to be handled at the application level. Query patterns: Performs well at read operations with appropriate use of primary keys.	User access management: Offers user-friendly RBAC system, which allows administrators to create, assign or revoke roles with specific permissions through the interface with predefined roles covering a wide range of access needs like administrative roles, read/write roles, and etc. access control can be. Limitations: Table-specific permissions are managed at the keyspace level and not at the individual table level. CQL does not include commands for managing permissions. All access controls must be managed through the web interface.	Schema enforcement: Primary keys (partition keys) are enforced at the database level. SQL-like constraints like NOT NULL and UNIQUE constraints are not supported by the system and must be enforced at the application level. Database triggers and complex constraints are also not supported. Integrity maintenance: Application-level validations are necessary to maintain data integrity. This involves implementing validation logic and integrity checks at the application level.
Key-Value (Redis)	Schema-free data creation: Allows the creation of key-value pairs without predefined schemas where users can insert data of different types without specifying the schema prior, which provides flexibility to add fields to any key at any time. Bulk data import: Bulk data creation can be achieved through the upload of text files or JSON files. Bulk loads are processed efficiently, i.e. can handle 1000 records in 0.725 seconds.	Indexing: Data manipulation relies indexing for efficient retrieval where users must create indexes to search for key-value pairs based on specific values. Limitations: Automatic updates with arithmetic operations are not supported. Data manipulation tasks such as updating specific fields require manual intervention. Performance: Performs well for basic CRUD operations, however, requires the need for manual updates and external scripting such as Python for more complex operations.	Retrieval speed: Excels in fast data retrieval (with or without indexes), with a typical retrieval time in the range of 80-90 milliseconds. Indexing: Creating indexes can significantly enhance retrieval efficiency, especially for searching based on field values. Limitations: complex queries, joins or views are not supported. Users must rely on patterns or indexes to retrieve data.	Simplicity: Offers a user-friendly and straightforward method to create or revoke access controls without coding, directly through the web interface. Limitations: Access control options are limited to full access, read-only, and read-write permissions which is applied to the entire database. Table-specific access control is not supported.	Constraints: Each key is ensured to be unique where basic data integrity is maintained at the key level. However, SQL-like constraints such as NOT NULL or UNIQUE constraints are not enforced. Use of application-level management: Data integrity such as ensuring no null values and uniqueness of key level must be managed explicitly at the application level.

Database Models	Data Creation	Data Manipulation	Data Retrieval	Access Control	Data Integrity
Document Model (CouchDB)	Bulk Data Import: Handles bulk data creation by inserting documents in JSON format. Schema Flexibility: Schema validation is not enforced, allowing flexible document structures. Users can store and retrieve JSON documents without defining a schema.	Ease of Use: Data manipulation is performed through JSON documents, providing flexibility in document structure. Querying and Indexing: Relies heavily on MapReduce functions for querying and indexing data. Users need to learn these functions to manage their database efficiently.	Views and Joins: Views can be created for documents within the same database. Joins are limited to documents within the same database and cannot span across different databases. Performance: Performance of data retrieval depends on the efficiency of MapReduce functions and the design of views.	User Access Management: User-friendly access management through the Fauxton interface, with preset roles such as admin (full access) and member (read/write). No coding is required for basic roles. Customizability: Access control can be customized for each database. However, limited options include full access and read/write. More granular control, such as read-only access, requires creating new design documents and cannot be applied to standalone roles.	Constraint Enforcement: No built-in enforcement for constraints like NOT NULL or unique fields. Constraints must be managed through custom validation functions implemented in design documents. Unique Identifier: Uses a unique identifier (_id key) for each document, which is customizable.

STRENGTHS AND WEAKNESSES OF DATABASE MODELS

Overall, Oracle shows the best performance in terms of memory usage, CPU usage, and speed. To dive deeper into the strengths and weaknesses, Table 3 shows the characteristics of each database model observed in both scenarios.

Relational databases are known for their robust data integrity and ability to handle complex queries, primarily due to their rigid schema design and enforcement of constraints. However, these same features can become limitations when dealing with large datasets, as performance may decline, and the inflexible schema can make adapting to changes challenging. On the other hand, graph databases stand out for their powerful querying capabilities enabled by flexible schemas and the use of AQL (ArangoDB Query Language). Despite these strengths, they come with a steep learning curve and may require ongoing performance tuning, especially as dataset sizes increase.

Wide Column stores, like Cassandra, are designed for high write throughput and straightforward data operations, making them ideal for applications requiring fast, large-scale data processing. Yet, their simplicity in handling queries means they struggle with more complex data manipulation tasks, often necessitating additional setup and management efforts. Key-value stores excel in scenarios requiring rapid processing and retrieval, especially in batch operations, but they lack built-in data validation and the ability to perform complex queries, which can limit their use in more sophisticated applications.

Lastly, document databases offer a great deal of flexibility in schema design and ease of data creation, particularly for unstructured or semi-structured data. However, this flexibility comes with a trade-off as it can lead to data inconsistencies without careful management. Hence, it is crucial for users to implement stringent data validation practices. Overall, each database type has its unique strengths and challenges, and the choice of which to use depends largely on the specific requirements of the project or application.

DISCUSSION

The objective of the section is to present a thorough analysis of the results which compared the 5 database models-relational, graph, wide-column, key-value, and document-with emphasis on how well these models work in the retail industry, specifically for optimizing order processing and Customer Relationship Management (CRM).

To aid in answering the research questions defined in the Introduction section, a table which discusses the key aspects of each of the 5 database models is as shown in Table 4.

Research Question 1: How do different database models impact data creation, manipulation, retrieval, access control and integrity?

Data Creation and Manipulation

Relational databases like Oracle APEX are optimized in data consistency and integrity through structured schemas and SQL constraints. This makes data creation and manipulation simple

Table 5: Summary Database Model Suitability Relating to Data Creation and Manipulation, Data Retrieval (Query Performance), and Data Integriy and Security

Aspect	Database Model	Key Strengths
Data Creation and Manipulation	Relational (Oracle APEX)	Strong support for bulk data creation, built-in constraints, and robust transaction management.
	Wide-Column (AstraDB)	High write throughput, flexible schema, efficient handling of large-scale data ingestion.
Data Retrieval (Query Performance)	Relational (Oracle APEX)	Ability to handle complex queries and relationships with optimized performance.
	Graph (ArangoDB)	Ability to handle complex queries involving relationships and graph traversals.
Data Integrity and Security	Relational (Oracle APEX)	Highest level of data integrity, detailed access control, strict data validation measures.

in Oracle APEX, but also contributes to the inflexibility of the database. Graph databases like ArangoDB offer flexible schemas and are more semi structured. ArangoDB has adequate capabilities when it comes to dealing with complex data relationships, which is ideal for highly relational applications. Wide-column databases like AstraDB, which is built on Apache Cassandra, provide high write throughput and efficient bulk data operations. However, it does not handle complex operations well and need to be processed using application logic such as Python. Key-value databases, such as Redis, offers fast data creation and retrieval through key-value stores with the use of basic data types like simple key. However, it lacks features for complex data manipulation and validation. Document databases like CouchDB supports flexible document schema and handling of nested data. However, its operation may be slower in complex data manipulations as it relies heavily on MapReduce.

Data Retrieval (Query Performance)

Relational databases like Oracle APEX are efficient for data retrieval and have an effective querying language know as SQL. However, it is not suitable for complex querying capabilities on large datasets. Graph databases like ArangoDB are highly efficient in querying complex data relationships. It is particularly useful for hierarchical and networked data retrieval. Wide-column databases like AstraDB are mostly designed for read-intensive workloads on large datasets. It lacks support for complex join operations and queries which need to be handled at the application level. Key-value databases like Redis are highly

Table 6: Summary of Suitable Database Models for High Volume Transactions.

Database Model	Suitability for High-Volume Transactions in Online Retail	Limitations of Database Model
Wide Column (AstraDB)	High write throughput and efficient bulk data handling. Ideal for handling large scale data ingestion and retrieval with high scalability. Suitable for retail environments requiring rapid data processing.	Limited support for complex operations in CQL, requires the use of application-level logic like Python.
Key-Value (Redis)	In-memory data storage ensuring low latency. Optimal for retail scenarios requiring fast access and rapid processing, such as caching and session management.	Lacks advanced query manipulation and query support.
Document (CouchDB)	Efficient handling of nested documents and flexible document structure without the need of predefined schemas. Suitable for complex, nested data structures and high scalability. Beneficial for diverse retail scenarios with flexible document management.	Requires learning MapReduce functions. Limited to scenarios requiring high flexibility and scalability.

efficient for simple-based queries. It is ideal for caching but are not suitable for looking up complex queries. Document databases like CouchDB are good for retrieving nested and hierarchical data. However, it is inefficient for looking up data due to its need for indexing and MapReduce operations.

Access Control and Integrity

Relational databases like Oracle APEX provide robust access control and high data integrity using constraints which can be defined using SQL. Graph databases like ArangoDB offers a flexible way to implement access control. It also supports data integrity through application-level enforcement. Wide-column databases like AstraDB enforce primary key constraints through partition keys. It provides role-based access controls but requires data integrity constraints to be managed at the application level. Key-value databases like Redis mostly provide simple access control and has minimal data integrity enforcement. It focuses more on performance. Document databases like CouchDB

support flexible access controls. It ensures data integrity through unique document identifiers but lack built-in constraints.

Based on the comparative analysis, the most suitable data model for the different key aspects of required for e-commerce or online retail industries is summarized in Table 5 below:

Research Question 2: Which database model provides the best performance for high volume transactions and operations in retail and e-commerce?

Based on the results section, the various types of database models analyzed suggest different capabilities for managing large volumes of transaction and operations in retail and e-commerce. Each database model presents its unique strengths, but three database models stand out in this area:

Wide-Column Database (AstraDB)

Wide-column databases such as AstraDB are suitable for volume transactions due to its ability of high write throughput, of handling bulk data and for horizontal scaling, which is crucial for handling the large data loads which are common in retail and e-commerce applications. The database is designed to handle large-scale applications with massive volumes of data that requires frequent read and write request. AstraDB has a user-friendly web interface which allows each data creation and load with support for CSV files and data migration from DynamoDB, which is beneficial for large-scale retail operations that require fast data loading with high scalability. Additionally, AstraDB which is built on one of the most popular wide-column databases, Apache Cassandra, uses CQL (Cassandra Query Language), which has similar syntax to SQL. This simplifies data manipulation for users that are familiar with relational databases. However, it comes with the limitation that CQL lacks support for complex operations and requires them to be handled at the application level such as using Python.

Key-Value Database (Redis)

Using key-value databases like Redis can be useful in high transactional environments, specifically in caching, session management, and simple-key operations. Redis features an in-memory data storage which can help ensure low latency, and this makes it suitable applications which requires rapid read and write operations, which is critical for real-time processing needs in e-commerce platforms. For instance, Redis has the ability to process 1000 records in just 0.725 seconds, and this highlights its efficiency in handling bulk operations. However, it comes with the limitation that it lacks advanced data manipulation capabilities and query support, and this limits its use for simple and more straightforward operations, limiting the range of operations needed in complex retail scenarios.

Document Database (CouchDB)

Document databases such as CouchDB are well-suited for high volume operations. It is highly efficient in dealing with nested documents which are fairly common in e-commerce or retail applications especially when dealing with diverse product catalogs and user or customer profiles. CouchDB allows for flexible document structure and does not require the need for predefined schemas, which is beneficial for applications that require high scalability, making it suitable in accommodating the dynamic nature of retail data. CouchDB uses MapReduce functions for querying and indexing for efficient data retrieval which is crucial for large datasets. However, it is noted that users must familiarize themselves with the functions to be able to manage the database effectively. Other than that, CouchDB has the ability to JSON documents through its user-friendly access management through the Fauxton interface. This makes it highly suitable for e-commerce applications and scenarios which demands high flexibility.

In conclusion, the best database model for high volume transactions and operations in retail and e-commerce depends on the specific use case, but generally the three database models discussed present the best performance and their key strengths along with its limitations is summarized in Table 6 below:

In short, the selection of the right database model for high volume transactions and daily operations in the retail and e-commerce industries will depend on the needs and preferences of individual businesses. Wide-column databases such as AstraDB perform well in write throughput and work well in a setting that requires loading large amounts of data and processing them quickly, but do not support detailed operations without further application-level enjoyment. Some of the popular NoSQL databases like Redis are great for real-time data processing and caching since they store data in memory but are not very good at querying. CouchDB based document databases are flexible and easy to manage with nested documents and can handle dynamic structure of retail data while they are functioning but one needs to understand MapReduce functions. All the models have their own advantages and disadvantages in stability, query complexity and data manipulation. Hence, the selection will depend on the requirement of scalability and the query complexity of the retail or e-commerce application in question.

CONCLUSION

In conclusion, each database model possesses different characteristics that impact data creation, manipulation, retrieval, access control, and integrity. To answer the research questions, relational databases appear to be the most suitable database model for conducting data creation, manipulation, retrieval, access control, and integrity. Meanwhile, online retail organizations are encouraged to select NoSQL databases if they seek models solely for high performances. Overall, relational (SQL) databases prove

to be the best performing database model while balancing high volume transactions and operations in retail and e-commerce. This is because compared to NoSQL databases, relational database easily allows for the creation of views and indexes. These features are vital for the easier data management and access in the e-commerce industry where tons of data are collected. As a result, the online commerce industry will greatly enhance database management performance by implementing relational databases such as Oracle throughout the company.

These findings greatly contradict prior research where NoSQL databases such as graph and document models are suggested for higher management efficiency (Ramesh, *et al.*, 2016; Thakare *et al.*, 2023; Agrawal *et al.*, 2001). Even though this research has contrasting conclusions compared to other research papers, it does not mean that one is incorrect. This is because different scenarios and industries will require different database models, according to their respective needs. Hence, this report concludes that relational databases would be preferred in the retail industry because SQL databases will impose strict data integrity constraints, while other NoSQL databases do not.

This conclusion does not imply that another research is incorrect. Instead, it highlights how different database models cater to varying needs. It is up to researchers to determine which database model best suits their objectives in terms of performance and scalability. To maintain strict data integrity and consistency, relational databases would be the best choice for online retail industries.

Moving on, this report further added to existing studies by highlighting the efficiencies of relational database models when compared to NoSQL models, showing how the lack of features such as access controls, views, and join functions will impact the decision on the preferred database model in a retail industry. This is one of the reasons why relational databases are preferred in this research study even though NoSQL databases have a more flexible schema validation, data creation, and table population. This study also contributed information on the workings of different NoSQL databases such as ArangoDB and AstraDB as not much study has been conducted on these database models.

This study has made the realization that there is no such thing as a best database model. Instead, there is the better performing database model that best suits the needs and objectives of an organization. Therefore, database managers must constantly evaluate the objectives in relation to the schema, data type, and data volume of a database before deciding on the database model to use. We learned that to be a database engineer or relevant roles, one must be meticulous and patient when handling a database to ensure that the database model will align with company needs. Furthermore, we have also learned the importance of balancing flexibility with data consistency. This is because although relational databases have higher data consistency due to strict constrain rules, they lose out on the flexibility of data handling compared to NoSQL database models. Hence, understanding this concept will allow researchers to have clearer understanding on which database model to use in specific scenarios. As a result, this paper allowed us to enhance our understanding in both SQL and NoSQL database models.

Lastly, the team has also learned the importance of refining current database knowledge to keep up with current world changes. By keeping an open mind to learning new things, the team is more prepared to handle changes in the database industry. With the report, we have also enhanced our experience in team collaboration as we consistently maintained clear communication with each other. Each member has learned to balance our personal tasks with this report, ensuring that every deadline is met consistently. As a result, the time management skills of each member are also greatly increased to maintain the consistent work pace in completing this research paper.

One of the limitations of the study is the usage of different devices with varying specifications. Due to location constraints, the team utilized their own personal laptops with different hardware configurations to test the performance of the database that they were assigned to. Different devices have different RAM and CPU that affect the performance rate of the database. This difference in hardware may have led to inconsistencies in the database performance result. Additionally, another limitation is that our research only focuses on two main scenarios specifically for the retail industry which are company supply order management and CRM of a company. Other studies that utilize these database models while focusing on other industries such as healthcare, may yield different results due to differences in data volumes and other factors. Furthermore, the team's lack of proficiency with their respective databases is another one of the drawbacks of the study. This is because this was their first time using the database; thus, they were faced with a steep learning curve. Thus, this leads to user errors and inefficiencies in managing the database. Despite these limitations, the findings from the study provide a deeper understanding of the strengths and weaknesses of each database model in the retail industry.

Future research in this field could focus on the database models' performance under larger volumes of data. Researchers can identify at which point does the database model begin to experience a decrease in performance. This will allow researchers to determine the limitations of each database model and identify the best-performing model for large datasets. One aspect that was not tested in this study was the database models' backup and restore functions. Future investigations can be conducted to determine the frequency of the backup system and how it impacts the database model's performance. Additional studies can be carried out to identify the time taken to restore the data as well as to what extent and how far back in time data could be recovered. The study of the backup and recovery system would benefit

organisations as they would be able to make well-informed decisions when choosing the database model that would suit the needs of their own organisation. Moreover, it would be beneficial to research the ability of the five database models to process and query real-time data. This would be incredibly useful for the e-commerce and retail industry where data is constantly generated. Processes like inventory management, order processing and customer behavior analysis would benefit from real-time data processing as it enables the organisation to make quick decisions that may potentially affect their market positioning. Overall, researching these areas would provide further insights into the potential of each database model.

CONFLICT OF INTEREST

The authors declare that there is no conflict of interest.

REFERENCES

Aerospike, Inc. What is a key-value store or key-value database? Aerospike. Retrieved July 9, 2024, https://aerospike.com/what-is-a-key-value-store/#:~:text=A%20key%2Dvalue%20store%2C%20or,as%20a%20key%2Dvalue%20pair

Agrawal, R., Somani, A., & Xu, Y. (2001). Storage and querying of e-commerce data. In Proceedings of the 27th International Conference on Very Large Data Bases (pp. 149–158) [Online]. Retrieved July 12, 2024, https://www.vldb.org/conf/2001/P149.pdf

Akhmetova, Z. B., Kozhamkulova, J. T., & Kim, I. A. (2022, December). 'Consumer behavior transformation in the digital economy on the basis of quantitative analysis,' Bulletin of "Turan" University, no. 4. Bulletin of "Turan" University, (4), 116–129, Dec. 2022. https://doi.org/10.46914/1562-2959-2022-1-4-116-129 [Online Abstract].

Akhtar, Z. Wide column database (use cases, example, advantages & disadvantages) [Database]. Town. Retrieved July 8, 2024, https://databasetown.com/wide-column-database-use-cases/.

AQL data queries. ArangoDB Documentation. Retrieved July 8, 2024, https://docs.arangodb.com/3.11/aql/data-queries/

Astra DB vector store. Python LangChain. Retrieved July 8, 2024, https://python.langchain.com/v0.2/docs/integrations/vectorstores/astradb/

Australia, O. SQL for accessing, defining, and maintaining data. Oracle. Retrieved July 8, 2024, https://www.oracle.com/au/database/technologies/appdev/sql.html.

Belgundi, R., Kulkarni, Y., & Jagdale, B. (February 23, 2023). Analysis of native multi-model database using ArangoDB. In Proceedings of the Third International Conference on Sustainable Expert Systems, Singapore (pp. 929–935) [Online]. https://doi.org/10.1007/978-981-19-7874-6_68

Bourgeois, D. T. (2014), Chapter 4. Data and databases. Information systems for business and beyond, Saylor academy (pp. 39–51) [Online]. Retrieved July 8, 2024, https://pressbooks.pub/bus206/chapter/chapter-4-data-and-databases/.

Cai, L., & Zhu, Y. (May 2015). The challenges of data quality and data quality assessment in the big data era. Data Science Journal, 14(2), 1–10. https://doi.org/10.5334/dsj-2015-002 [Online].

Chapters 1.3 Eventual consistency. In CouchDB documentation. Retrieved July 9, 2024, https://docs.couchdb.org/en/stable/intro/consistency.html

Chapters 1.4. cURL: Your command line friend. In CouchDB documentation. Retrieved July 9, 2024, https://docs.couchdb.org/en/stable/intro/curl.html

Chapters 1.5. Security. In CouchDB documentation. Retrieved July 9, 2024, https://docs.couchdb.org/en/stable/intro/security.html

Chapters 1.6. Getting started. In CouchDB documentation. Retrieved July 9, 2024, https://docs.couchdb.org/en/stable/intro/tour.html

CouchDB – Deleting a document. Tutorialspoint. Retrieved July 9, 2024, https://www.tutorialspoint.com/couchdb/couchdb_deleting_a_document.htm

CouchDB – Updating a document. Tutorialspoint. Retrieved July 9, 2024, https://www.tutorialspoint.com/couchdb/couchdb_updating_a_document.htm

"1.4.2. /db/doc/attachment," CouchDB Documentation. Retrieved July 9, 2024, https://docs.couchdb.org/en/stable/api/document/attachments.html

DataStax, "Manage roles and permissions." DataStax Documentation. Retrieved July 8, 2024, https://docs.datastax.com/en/astra-db-serverless/administration/manage-database-access.html.

Duggineni, S. (2023) [Online]. Impact of controls on data integrity and information systems. Science and Technology, 13(2) (pp. 29–35). Retrieved July 8, 2024, https://www.researchgate.net/profile/Sasidhar-Duggineni/publication/372193665_Impact_of_Controls_on_Data_Integrity_and_Information_Systems/links/64a8d256b9ed6874a5046bc3/Impact-of-Controls-on-Data-Integrity-and-Information-Systems.pdf

Gillis, A. S., Stedmand, C., & Vaughan, J. What is data modelling? Tech. Target. Retrieved July 8, 2024, https://www.techtarget.com/searchdatamanagement/definition/data-modeling

Graph database basics. ArangoDB Documentation. Retrieved July 8, 2024, https://arangodb.com/graph-database/

Gudivada, V., Apon, A., & Rao, D. L. (2018), Chapter 3. Database systems for big data storage and retrieval. Handbook of research on big data storage and visualization techniques. In R. S. Segall & J. S. Cook (Eds.).Engineering Science Reference, 2 (pp. 76–100) [Online]. https://doi.org/10.4018/978-1-5225-3142-5.ch003

Harrington, J. L. (2009), Chapter 5. The relational data model. Relational database design: Clearly explained (3rd ed.). In Morgan Kaufmann Publishers (pp. 85–101) [Online]. https://doi.org/10.1016/B978-0-12-374730-3.00005-X

Hughes, R. (2016), Chapter 4. Essential DW/BI background and definitions. Agile data warehousing for the enterprise: A guide for solution architects and project leaders. In Morgan Kaufmann Publishers (pp. 59–84) [Online]. https://doi.org/10.1016/B978-0-12-396464-9.00004-7

Jaleel, R. A., & Abbas, T. N. J. (2020). Design and implementation of efficient decision support system using data mart architecture International Conference on Electrical, Communication, and Computer Engineering (ICECCE), Istanbul, Turkey, 2020 (pp. 1–6) [Online]. https://doi.org/10.1109/icecce49384.2020.9179313

Jamra, R. K., Anggorojati, B., & Kautsarina, D. I. (2020). Sensuse and R.R. Suryono. "Systematic review of issues and solutions for security in e-commerce," International Conference on Electrical Engineering and Informatics (ICELTICs), Aceh, Indonesia, 2020 (pp. 1–5) [Online]. https://doi.org/10.1109/ICELTICs50595.2020.9315437

Kameswari, J., Ramesh, P., Bhavikatti, V., Omnamasivaya, B., Chaitanya, G., & T. Bastray, S. Hiremath, G. S. Gondesi, "Analyzing the role of big data and its effects on the retail industry," Web Intelligence, vol. 22, no. 1, pp. 45–63, Mar. 26, 2024, doi: 10.3233/WEB-230027. [Online Abstract].

Kaur, G., & Kaur, J. (March 2018). In-memory data processing using redis database. International Journal of Computer Applications, 180(25), 26–31. https://doi.org/10.5120/ijca2018916589

Kitterman, C. Enhanced multi-region database consistency in Astra DB. DataStax. Retrieved July 8, 2024, https://www.datastax.com/blog/enhanced-multi-region-database-consistency-astra-db.

Klemová, P. The evolving landscape of e-commerce: Trends and innovations shaping the future of online shopping," ui42. Retrieved July 1, 2024, https://www.ui42.com/blog/the-evolving-landscape-of-e-commerce, Retrieved July 1, 2024

Le, D. M., Pham, V. D., Lunven, C., & Ho, A. (2023). On the principles of microservice-NoSQL-based design for very large scale software: A cassandra case study. In T. D. L. Nguyen, E. Verdú, A. N. Le, M. Ganzha (Eds.).Intelligent Systems and Networks (pp. 591–602). Springer Nature Singapore. https://doi.org/10.1007/978-981-99-4725-6_70

Liu, L., & Özsu, M. T. (2018) [Online]. Encyclopedia of database systems (2nd ed.). Springer New York. https://doi.org/10.1007/978-1-4614-8265-9

Liu, Q. (January 2019). A high performance memory key-value database based on Redis. Journal of Computers, 14(3), 170–183. https://doi.org/10.17706/jcp.14.3.170-183

Luisi, J. V. (2014), ch. Information architecture. Pragmatic Enterprise Architecture: Strategies to Transform Information Systems in the Era of Big Data. In Morgan Kaufmann Publishers, 4 (pp. 189–261) [Online]. https://doi.org/10.1016/B978-0-12-800205-6.00004-4

Nayak, A., Poriya, A., & Poojary, D. (March 2013) [Online]. Type of NoSQL databases and its comparison with relational databases. International Journal of Applied Information Systems (IJAIS), 5(4) (pp. 16–19). Retrieved July 5, 2024, https://www.researchgate.net/publication/302557703_Article_Type_of_nosql_databases_and_its_comparison_with_relational_databases

Olivera, H. V., Holanda, M. T., Guimarães, V., Hondo, F., & Boaventura, W. (2019). Data modeling for NoSQL document-oriented databases. In Symposium on Information Management and Big Data (pp. 129–135) [Online]. Retrieved July 9, 2024, https://ceur-ws.org/Vol-1478/paper17.pdf

Palanisamy, A. M., Nataraj, R. V., S. S., & Sountharrajan, S. (2020). Virtual datastack for application domains: Concepts, challenges and generation techniques 3rd International Conference on Intelligent Sustainable Systems (ICISS), Thoothukudi, India, 2020 (pp. 1314–1318) [Online]. https://doi.org/10.1109/ICISS49785.2020.9315915

Patel, A. Document database in NoSQL. Tutorialspoint. Retrieved July 9, 2024, https://www.tutorialspoint.com/document-database-in-nosql

Ramesh, D., Khosla, E., & Bhukya, S. N. (2016). Inclusion of e-commerce workflow with NoSQL DBMS: MongoDB document store IEEE International Conference on Computational Intelligence and Computing Research (ICCIC), Chennai, India, 2016 (pp. 1–5). https://doi.org/10.1109/ICCIC.2016.7919652

Reddy. (2024). The role of NoSQL databases in scaling e-commerce platforms International Journal of Advanced Engineering Technologies and Innovative Research. Retrieved July 12, 2024, https://ijaeti.com/index.php/Journal/article/view/302

Sadalage, P. J., & Fowler, M. (2012) [Online]. NoSQL distilled: A brief guide to the emerging world of polyglot persistence. Pearson Education. Retrieved July 9, 2024, https://bigdata-ir.com/wp-content/uploads/2017/04/NoSQL-Distilled.pdf

Seetharaman, A., Niranjan, I., Tandon, V., & Saravanan, A. S. (2016). Impact of big data on the retail industry. Corporate Ownership and Control, 14(1), 506–518. https://doi.org/10.22495/cocv14i1c3p11 [Online].

Smith, J. 5 challenges for e-commerce & retail that our data management solutions can help with. Staedean. Retrieved July 3, 2024, https://www.to-increase.com/business-integration/blog/challenges-retail-ecommerce-solutions-data-management-solutions

Soares, D. (December 2015). doi. https://d117h1jjiq768j.cloudfront.net/docs/default-source/marklogic-docs/tdwi-ecommerce-nosql-retail.pdf?sfvrsn=e370502e_7. E-commerce and NoSQL in retail. TDWI's Business Intelligence Journal, 20(4), 51–55.

Thakare, A., Tembhurne, O. W., Thakare, A. R., & Narasimha Reddy, S. N. (February 17, 2023). NoSQL databases: Modern data systems for big data analytics – Features, categorization and comparison. International Journal of Electrical and Computer Engineering Systems, 14(2), 207–216. https://doi.org/10.32985/ijeces.14.2.10 [Online].

Trachim, A. Big data in e-commerce: A quick guide. In Data Laboratories. Retrieved July 3, 2024, https://indatalabs.com/blog/big-data-analytics-in-ecommerce

Tupper, C. D. (2011), Chapter 11. Model constructs and model types. Data architecture: From Zen to reality. In Morgan Kaufmann Publishers (pp. 207–221) [Online]. https://doi.org/10.1016/b978-0-12-385126-0.00011-5

Vohra, D. (2016). Using apache cassandra. In (pp. 81–93). Apress. https://doi.org/10.1007/978-1-4842-1830-3_6

Wei, F., & Zhang, Q. (2018). Design and implementation of online shopping system based on B/S model. MATEC Web of Conferences, 246, Article 03033. https://doi.org/10.1051/matecconf/201824603033

West, M. (2011), ch. Some types and uses of data models. Developing High Quality Data Models, Morgan. In Kaufmann, 3 (pp. 23–36) [Online]. https://doi.org/10.1016/B978-0-12-375106-5.00003-8

What is a key-value database? Redis. Retrieved July 9, 2024, https://redis.io/nosql/key-value-databases/

What is a wide-column database? Definition & FAQs. ScyllaDB. Retrieved July 8, 2024, https://www.scylladb.com/glossary/wide-column-database/

Winberg, V., & Zubac, J. (2019) [Online]. A comparison of relational and graph databases for CRM systems [Unpublished MS thesis]. Department of Computer Science, Lunds Universitet. Retrieved July 14, 2024, https://lup.lub.lu.se/luur/download?func=downloadFile&recordOId=8987929&fileOId=8987933

Yeung, J., Wong, S., & Wu, J. (April 2018). Graph databases for e-commerce platforms. Presented at the 2018 International Conference on Internet Studies, Takamatsu, Japan [Online]. https://www.researchgate.net/publication/338710108_Graph_Databases_for_E-Commerce_Platforms

Yusof, M. K. (July 2017). Efficiency of JSON for data retrieval in big data. Indonesian Journal of Electrical Engineering and Computer Science, 7(1), 250. https://doi.org/10.11591/ijeecs.v7.i1.pp250-262

Cite this article: Wong LHE, Ng JW, Tan AXT, Yong MJ, Lim SL, Sathishkumar VE, *et al*. A Comparative Study of Relational, Graph, Wide Column, Key-Value, and Document Models in Retail Industries. Info Res Com. 2025;2(1):71-97.

APPENDIX A

Queries for Order Scenario

Data Creation

Create "User" database.

Data Manipulation

(Insertion)

Insert the following data into the "Goods" database.

_id: G_0001

Good_id: G_0001

Good_name: Coffee Maker

Good_price: 114.35

Description: High-quality coffee maker for office use.

(Modification)

Price of good "Laminating Sheets" increased by 10%.

Data Retrieval

Create a view called "CustomerOrderHistory". This view combines information from the "User", "Order", "Goods" and "OrderGoods" database.

Access Control

Create a role called "auditor" and a role called "employee". Grant read-only access to the "Order" and "User" database to the auditor role. Revoke all privileges from those with the "employee" role.

Data Integrity

Ensure that the "User" database adheres to the UNIQUE constraint.

Queries for CRM Scenario

Data Creation

Create "Deal" database.

Data Manipulation

(Insertion)

Insert the following data into the "Deal" database.

_id: D_0001

name: Cross-group high-level array

value: 409100

probability: 0.62

(Modification)

Update "Wayne Hall" phone number to "1234567890" and email to "waynehall@gmail.com".

Data Retrieval

Retrieves the names, phone numbers, and email addresses of people who have deals with a probability greater than 0.8.

Access Control

Create a role called "associate consultant". Grant read-only access to the "Deal" database to associate consultant. Revoke the insert privileges from the associate consultant.

Data Integrity

Ensure that the "Document" database adheres to the NOT NULL constraint.

APPENDIX B

Python Code Snippet to Generate Records for Order Scenario

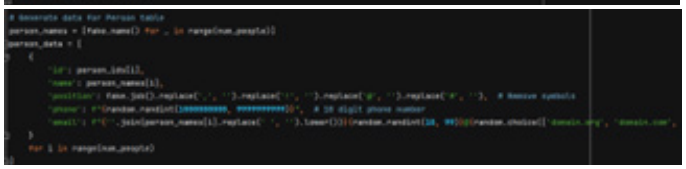
```
# Set seed to 1000
random.seed(1000)

# Generate data in tabular format
data = []

for i in range(1, 1001):
    user_id = f"U_{i:04d}"
    first_name = random.choice(first_names)
    last_name = random.choice(last_names)
    user_name = f"{first_name}{last_name}"
    user_email = random_email(first_name, last_name)
    user_tel = ''.join(random.choices(string.digits, k=10))
    user_birth = random_date(1955, 2000)  # Adjusted birth year range
    user_pass = ''.join(random.choices(string.ascii_letters + string.digits, k=8))
    street_number = random.randint(100, 999)
    street_name = random.choice(streets)
    user_address = f"{street_number} {street_name}"

    data.append([user_id, user_name, user_email, user_tel, user_birth, user_pass, user_address])
```

```
# Function to generate realistic goods records
def generate_goods_records(num_records):
    goods_records = []
    for i in range(num_records):
        good = random.choice(goods)
        good_id = f"G_{i+1:04d}"
        good_name = good["name"]
        good_price = round(random.uniform(good["min_price"], good["max_price"]), 2)
        description = f"High-quality {good_name.lower()} for office use."
        goods_records.append((good_id, good_name, good_price, description))
    return goods_records
```

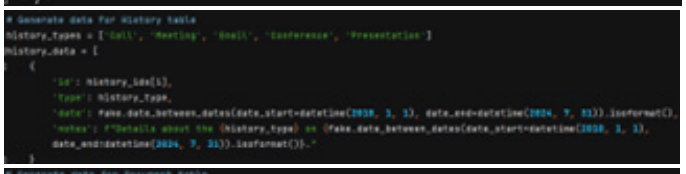
```
orders_data = []
for i in range(1, 1001):
    order_id = f"O_{i:04d}"
    date = random_date(2018, 2023)  # Adjusted date range from 2018 to current year
    consumption = round(random.uniform(10.0, 1000.0), 2)
    delivery_address = f"{random.randint(100, 999)} {random.choice(streets)}"  # Random street address format
    user_id = f"U_{random.randint(1, 1000):04d}"

    orders_data.append((order_id, date, consumption, delivery_address, user_id))
```

```
orderGoods = []
for i in range(1, 1001):
    order_id = f"O_{i:04d}"  # Generates order IDs like O0001 to O1000
    num_goods = random.randint(1, 4)  # Randomly choose 1 to 4 goods IDs per order
    for _ in range(num_goods):
        good_id = f"G_{random.randint(1, 1000):04d}"
        orderGoods.append((order_id, good_id))
```

Python Code Snippet to Generate Records for CRM Scenario



```
# Generate data for Deal table
deal_data = [
    {
        'id': deal_ids[i],
        'name': fake.catch_phrase(),
        #'value': f"{random.randint(1000, 999999):,}".replace(',', ''),  # 4-7 digits, ending in 00
        'value': f"{random.randint(10, 9900) * 100:05d}",
        'probability': round(random.uniform(0, 1), 2)  # Probability between 0.01 and 1.00
    }
```



APPENDIX C

Schema Constraints on Order Scenario

User Table

Pk (or not null + unique): User_id
Not null: User_name, User_email, User_tel, User_pass, User_address
Unique: User_email, User_tel

Goods Table

Pk (or not null +unique): Good_id
Not null: Good_name, Good_price, Description

Order Table

Pk (or not null +unique): Order_id
Not null: Date, Consumption, Delivery_address, User_id

OrderGoods Table

Pk (or not null +unique): Order_id
Not null: Good_id

Schema Constraints on CRM Scenario

Company Table

PK (or not null + unique): id
Not null + Unique: name, phone, website, address

Office Table

PK (or not null + unique): id
Not null + Unique: name, phone, address

Coworker Table

PK (or not null + unique): id
Not null: name, phone, email
Unique: phone, email

Person Table

PK (or not null + unique): id
Not null: name, position, phone, email
Unique: phone, email

Deal Table

PK (or not null + unique): id
Not null: name, value, probability

History Table

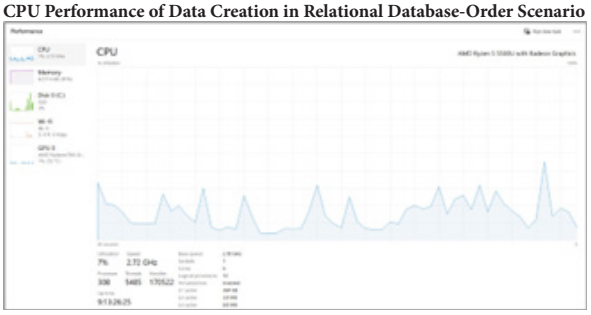
PK (or not null + unique): id
Not null: type, date, notes

Document Table

PK (or not null + unique): id
Not null: type, description, document

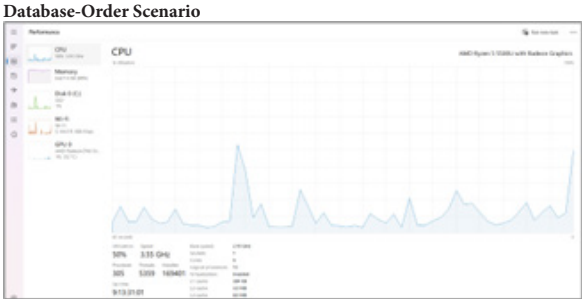
APPENDIX D

CPU Performance of Data Creation in Relational Database-Order Scenario

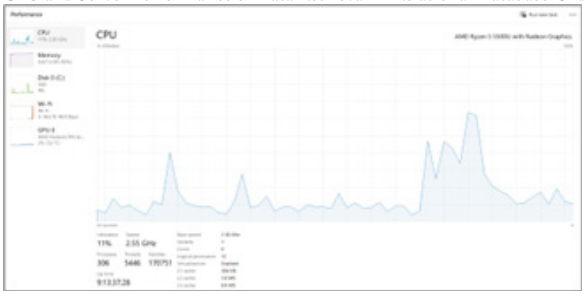


CPU and Server Performance of Data Manipulation (Insertion) in Relational Database-Order Scenario

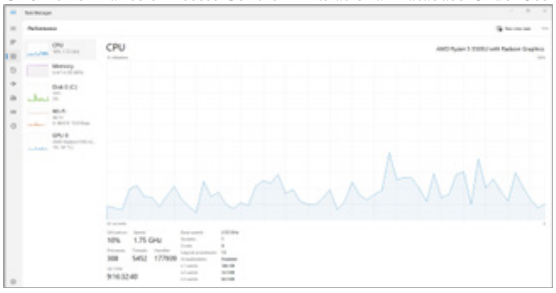
CPU and Server Performance of Data Manipulation (Modification) in Relational Database-Order Scenario



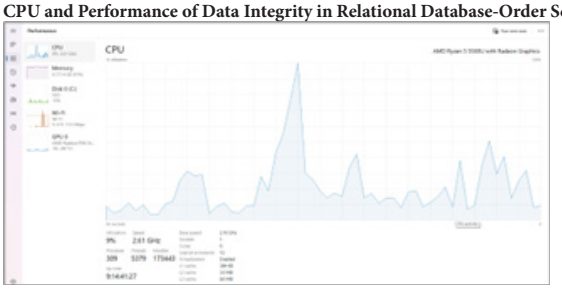
CPU and Server Performance of Data Retrieval in Relational Database-Order Scenario



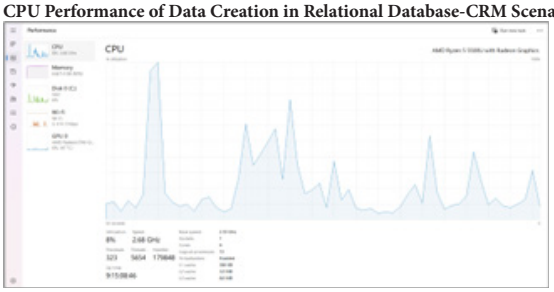
CPU Performance of Access Control in Relational Database-Order Scenario



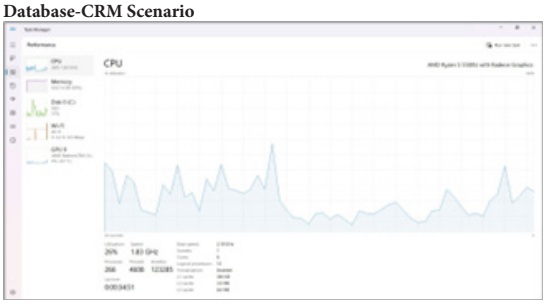
CPU and Performance of Data Integrity in Relational Database-Order Scenario



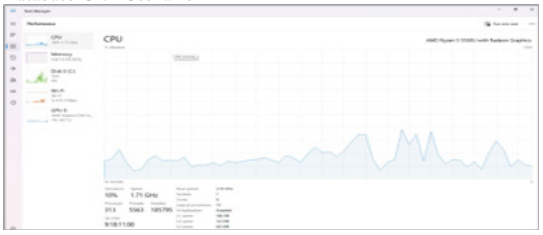
CPU Performance of Data Creation in Relational Database-CRM Scenario



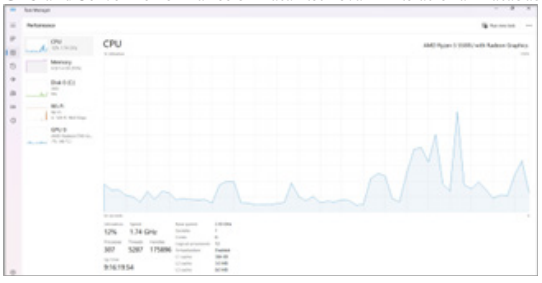
CPU and Server Performance of Data Manipulation (Insertion) in Relational Database-CRM Scenario



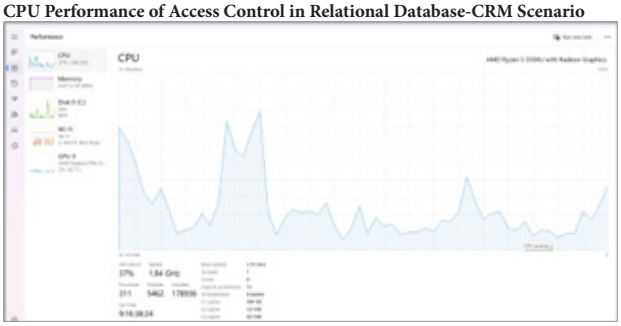
CPU and Server Performance of Data Manipulation (Modification) in Relational Database-CRM Scenario



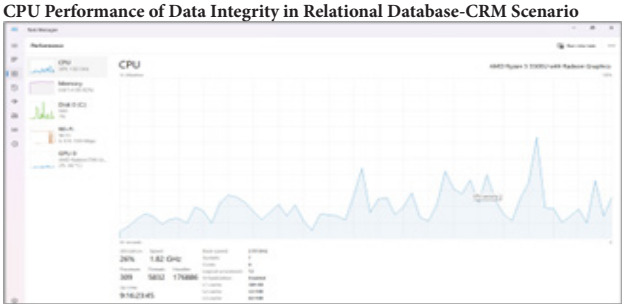
CPU and Server Performance of Data Retrieval in Relational Database-CRM Scenario



CPU Performance of Access Control in Relational Database-CRM Scenario

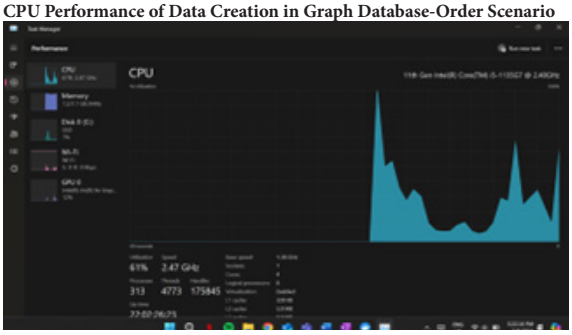


CPU Performance of Data Integrity in Relational Database-CRM Scenario

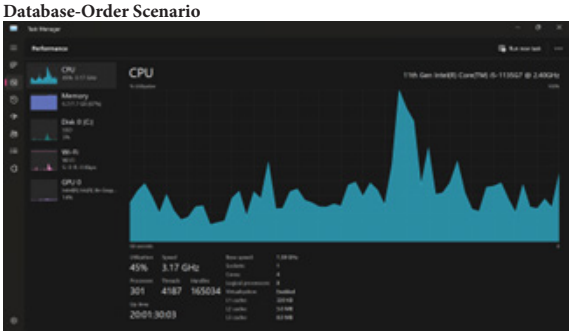


APPENDIX E

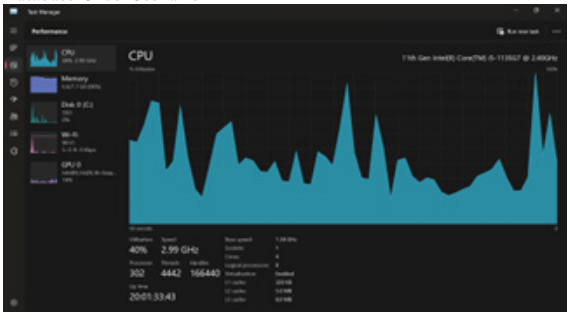
CPU Performance of Data Creation in Graph Database-Order Scenario

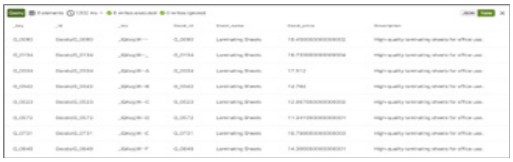


CPU and Server Performance of Data Manipulation (Insertion) in Graph Database-Order Scenario

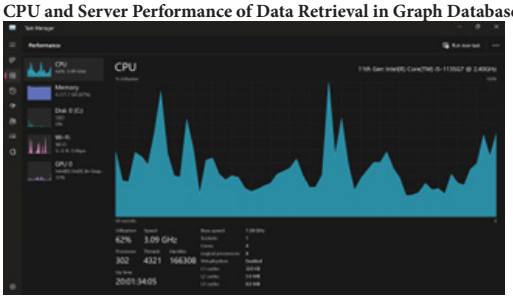


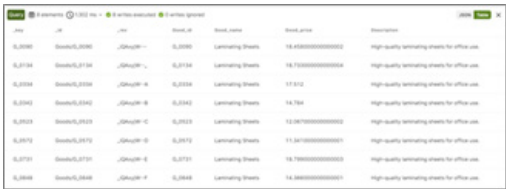
CPU and Server Performance of Data Manipulation (Modification) in Graph Database-Order Scenario



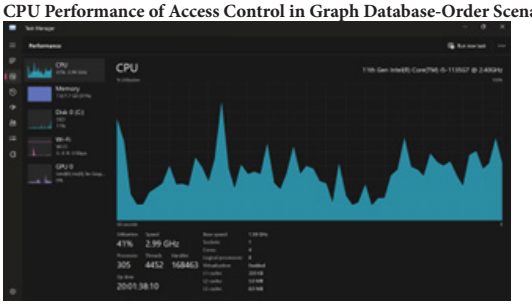


CPU and Server Performance of Data Retrieval in Graph Database-Order Scenario

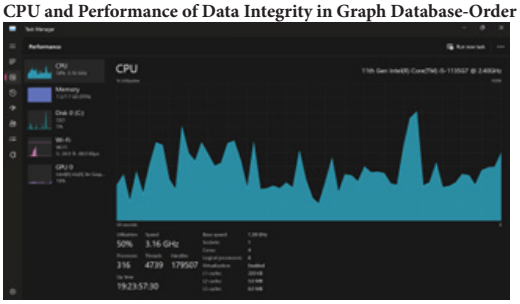




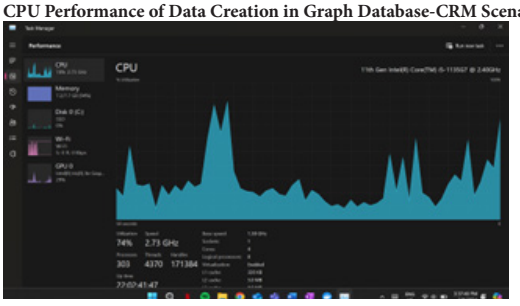
CPU Performance of Access Control in Graph Database-Order Scenario



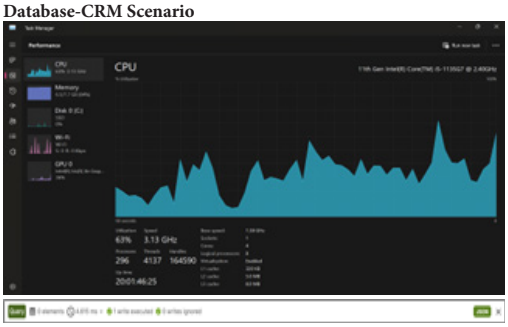
CPU and Performance of Data Integrity in Graph Database-Order Scenario



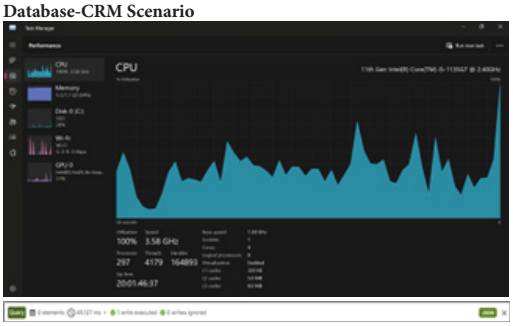
CPU Performance of Data Creation in Graph Database-CRM Scenario



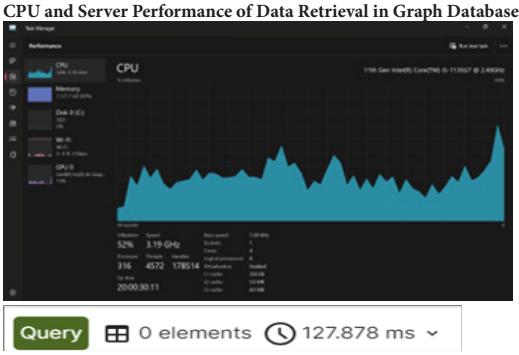
CPU and Server Performance of Data Manipulation (Insertion) in Graph Database-CRM Scenario



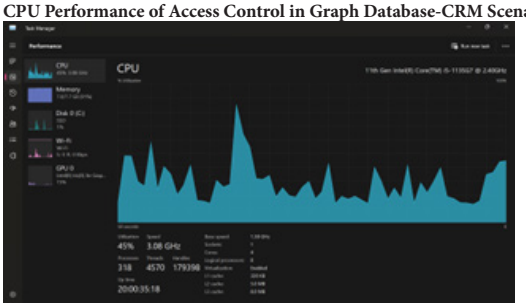
CPU and Server Performance of Data Manipulation (Modification) in Graph Database-CRM Scenario



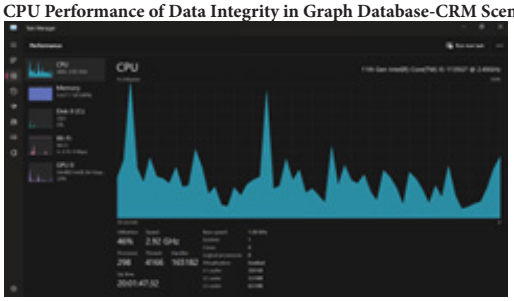
CPU and Server Performance of Data Retrieval in Graph Database-CRM Scenario



CPU Performance of Access Control in Graph Database-CRM Scenario

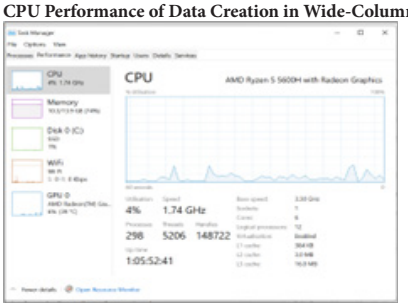


CPU Performance of Data Integrity in Graph Database-CRM Scenario



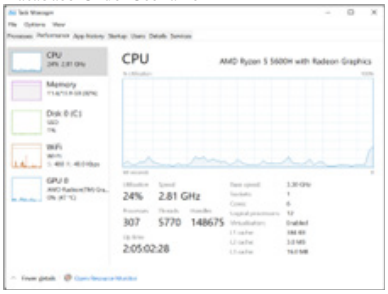
APPENDIX F

CPU Performance of Data Creation in Wide-Column Database-Order Scenario

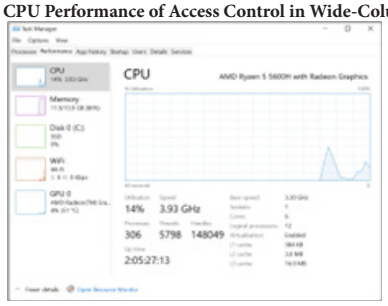


CPU and Server Performance of Data Manipulation (Insertion) in Wide-Column Database-Order Scenario

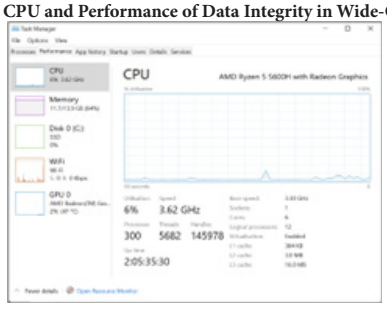
CPU and Server Performance of Data Manipulation (Modification) in Wide-Column Database-Order Scenario



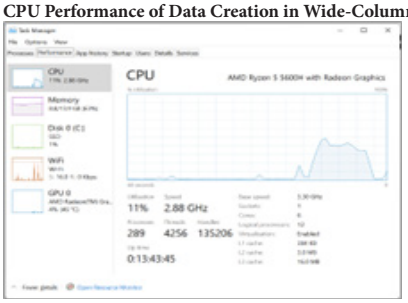
CPU Performance of Access Control in Wide-Column Database-Order Scenario



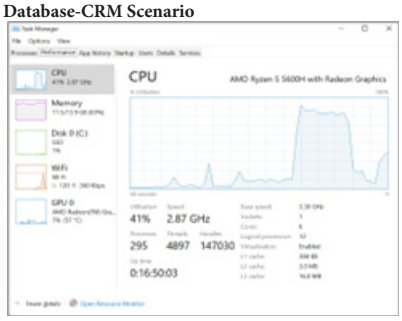
CPU and Performance of Data Integrity in Wide-Column Database-Order Scenario



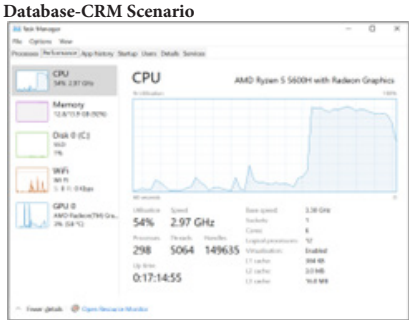
CPU Performance of Data Creation in Wide-Column Database-CRM Scenario



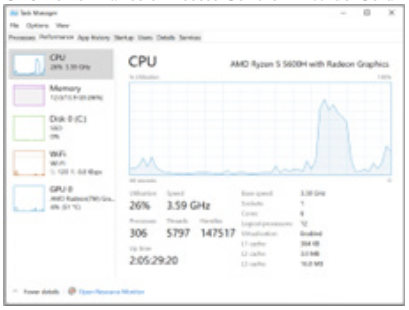
CPU and Server Performance of Data Manipulation (Insertion) in Wide-Column Database-CRM Scenario



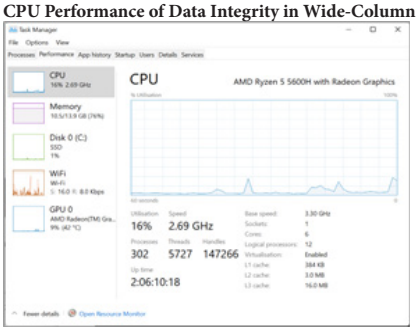
CPU and Server Performance of Data Manipulation (Modification) in Wide-Column Database-CRM Scenario



CPU Performance of Access Control in Wide-Column Database-CRM Scenario

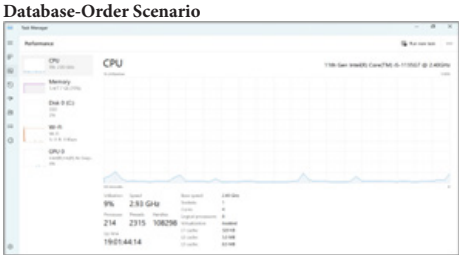


CPU Performance of Data Integrity in Wide-Column Database-CRM Scenario



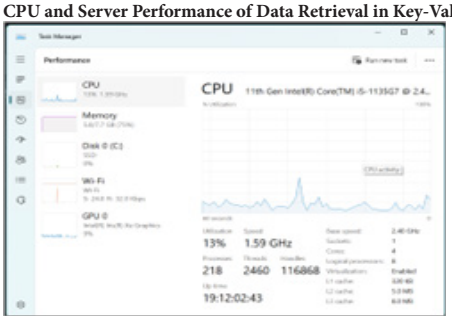
APPENDIX G

CPU and Server Performance of Data Manipulation (Insertion) in Key-Value Database-Order Scenario

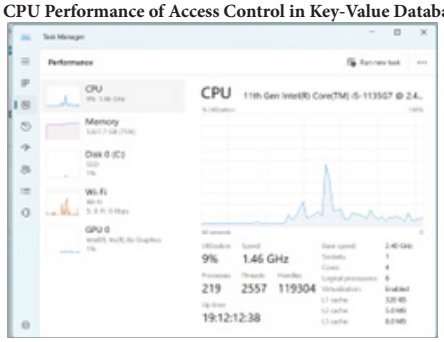


CPU and Server Performance of Data Manipulation (Modification) in Key-Value Database-Order Scenario

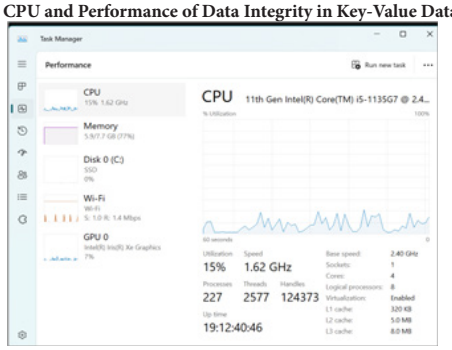
CPU and Server Performance of Data Retrieval in Key-Value Database-Order Scenario



CPU Performance of Access Control in Key-Value Database-Order Scenario

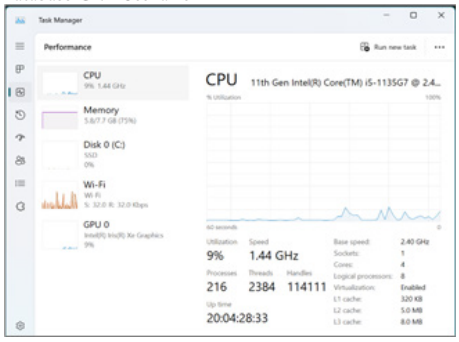


CPU and Performance of Data Integrity in Key-Value Database-Order Scenario

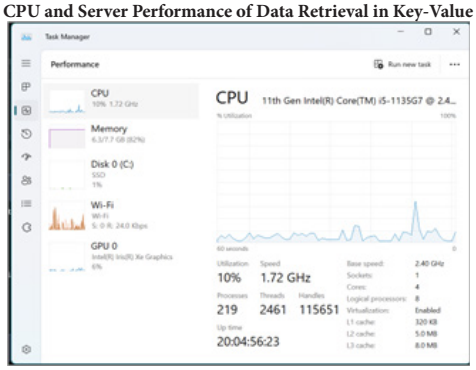


CPU and Server Performance of Data Manipulation (Insertion) in Key-Value Database-CRM Scenario

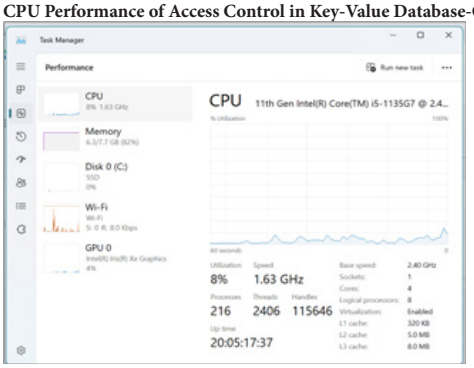
CPU and Server Performance of Data Manipulation (Modification) in Key-Value Database-CRM Scenario



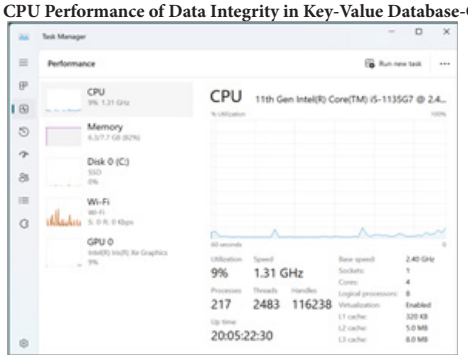
CPU and Server Performance of Data Retrieval in Key-Value Database-CRM Scenario



CPU Performance of Access Control in Key-Value Database-CRM Scenario

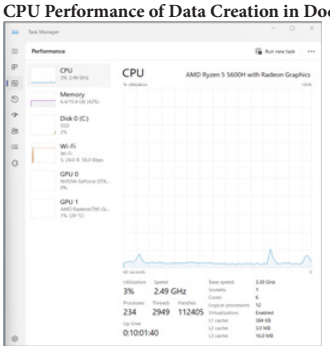


CPU Performance of Data Integrity in Key-Value Database-CRM Scenario

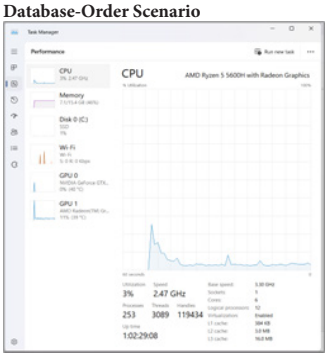


APPENDIX H

CPU Performance of Data Creation in Document Database-Order Scenario

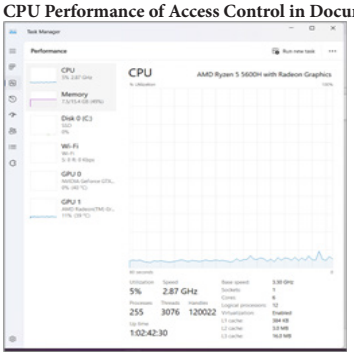


CPU and Server Performance of Data Manipulation (Insertion) in Document Database-Order Scenario

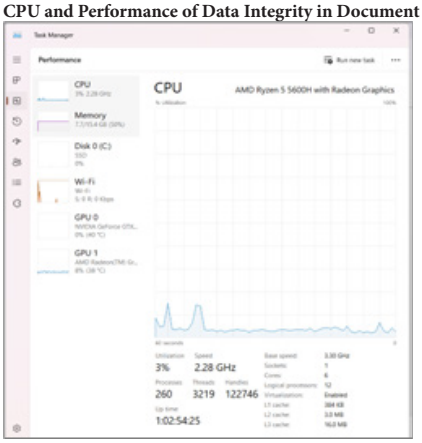


CPU and Server Performance of Data Manipulation (Modification) in Document Database-Order Scenario

CPU Performance of Access Control in Document Database-Order Scenario

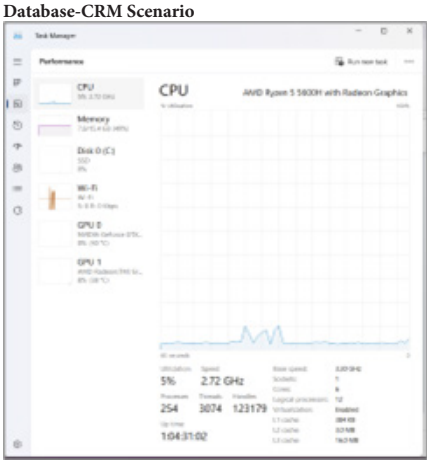


CPU and Performance of Data Integrity in Document Database-Order Scenario

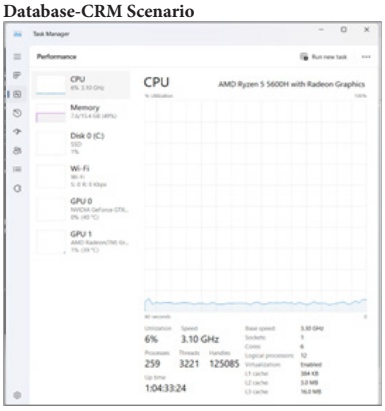


CPU Performance of Data Creation in Document Database-CRM Scenario

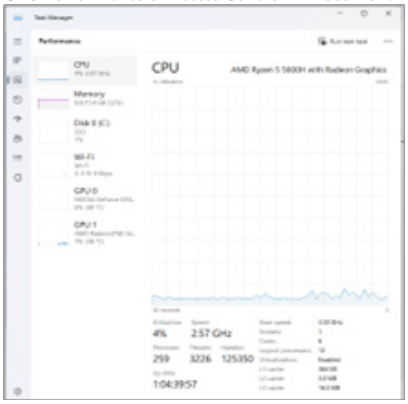
CPU and Server Performance of Data Manipulation (Insertion) in Document Database-CRM Scenario



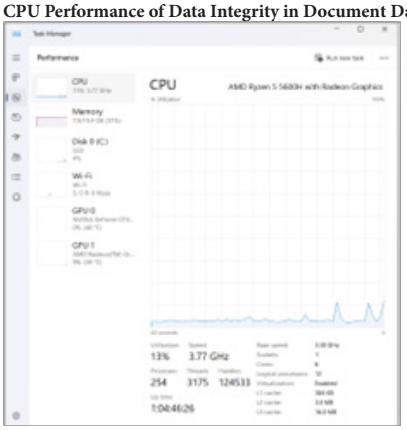
CPU and Server Performance of Data Manipulation (Modification) in Document Database-CRM Scenario



CPU Performance of Access Control in Document Database-CRM Scenario

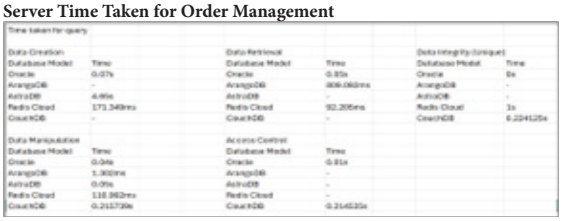


CPU Performance of Data Integrity in Document Database-CRM Scenario



APPENDIX I

Server Time Taken for Order Management



Server Time Taken for CRM Management