COMP1835 - GRAPH AND MODERN DATABASES

Richard Raja

Student ID: 001370307

COMP1835-GRAPHDB-185 and COMP1835-NOSQL-185

MSc Data Science

Table of Contents

Part 1

Summary

The article analyzes the characteristics, benefits, and weaknesses of relational databases in contrast to NoSQL databases. The article starts with an explanation of what is SQL and NoSQL and the reason for their rising popularity in the modern tech environment, which has been noticed by many enterprises including prominent ones. The text illustrates the main features of relational databases, stating that they stand out for their strict adherence to schemas and using data normalization to preserve structural integrity. On the other hand, the talk of NoSQL is about their capacity to adapt to different schema designs and manage large quantities of data which makes them a good option for handling such data.

The article makes a point of mentioning an important study between relational and NoSQL databases, addressing their main differences, and showing the way they can be utilized in different situations. Relational databases are very much specialized in providing consistent data structures and effective update operations (Deutsch *et al*., 2022). On the other hand, they work best for longitudinal scalability but may face problems when processing complicated queries. By contrast, NoSQL supports variable schemas, remarkable scalability, and quick read-and-write operations. On the other hand, this data redundancy and discrepancy in the data is the cost paid dearly.

Nonetheless, this paper also has some constraints as well, in that case it can be said that the material construction lacks the flow that a structured methodology would have provided. Also, the depth of analysis should be enhanced through the chapter, especially in certain categories of NoSQL databases. The presence of grammatical errors and inconsistencies in the citation style lessen the relevancy and professionalism of the paper, while the complex statements do undermine its clarity and comprehension.

The work is imperfect, though it has an excellent technical vocabulary grasp and takes part in the most significant database structure discussion. Nevertheless, aids like graphs and tables can greatly help in highlighting and understanding the complex ideas in the presentation. Furthermore, building audience engagement with the help of discussion questions or hypothetical situations would improve critical thinking and provide a better reading experience for the viewers.

Through the research paper, relational and NoSQL databases have been discussed in-depth, nevertheless, if the improvements in structure, in-depth analysis, and writing technique take place, it would make this research more useful and appealing for a wider audience.

Major issues

The study article's major issue is a deficit in analytical depth. The study looks into relational versus NoSQL databases, but there is no distinction made on case studies or by type to show actual world settings. For example, the article touches on key-value stores, column-based databases, document stores, and graph databases as NoSQL databases but has no insight into their benefits and drawbacks. This limitation hampers the usage of the paper for readers pursuing a wide and in-depth comprehension of NoSQL technology and its cases.

The final important issue lies in the difficulty of the respective pros and cons comparison between relational and NoSQL databases. The article covers a range of database models, yet it needs a more detailed look at relational databases shortcomings when compared to NoSQL databases (Tian, 2023). It places stress on schema rigidity or horizontal scalability in relational databases as opposed to data consistency or querying speed in NoSQL databases. Such conflicts provide a different view to readers from that of databases. They also may mislead and interfere with networks of trade-offs.

Furthermore, the report does not provide any statistical data to make its claims credible. This work simply draws on the core concepts of database design but it fails to present real case studies or practical data. The case studies with real examples, empirical research, and industry standards would the paper's lesson and make it more useful to practitioners who work with database management and scholars. The other thing would be to ensure the staff is more unified and create a proper flow of ideas which would also help the work. Themes change abruptly, and the flow is interrupted; spacing between different chapters is also filled with inappropriate transitions. Less confusing subtopics with using paragraphs and subtitles enable readability and navigation.

Identifying and tackling such significant issues can challenge the paper's academic value, range, and clarity and hence raise the level of its importance in relational as well as NoSQL database discussion.

Minor issues

The insignificant errors lower the impact of a focused study paper. Smaller issues like the unevenly formatted section and varied citation styles can also be found in the text. Some references use numerals as superscript, while others use author-date format causing chaos and disruption. The use of standardized citation styles like APA or IEEE increases the professionalism and readability level of a research document.

Another flaw in language and grammatical structure are minor. The report is typically well-defined and consistent, but some errors related to grammar, vocabulary, and typos may cause the readers to get confused. In order to minimize these grammatical deficiencies, proofreading and reviewing the document for its consistency and clarity is required.

In addition, it should have addressed analytical biases and the study's limitations. The database technologies are indeed complicated and hence, researchers need to disclose any biases, assumptions or limits that may have had an impact on their study to produce reliable results (Fan *et al*., 2021). These elements should be mentioned at least, to ensure both transparency and academic integrity.

The research might be further improved by the exploration of additional relevant works on relational and NoSQL databases. It encompasses core ideas and principles, and for further academic accuracy and substance, it can have inputs from present-day research papers, academy publications, and authoritative textbooks.

The report ends up without addressing database management practitioners and academic users, particularly. It explains in detail relational and NoSQL databases, but it would be better to add some advice on choice, installation and optimization of database technology in the real world. Given such recommendations the article would become more interesting for the readers.

Review of Presentation Style:

The report's manner of presentation is notable, featuring a well-structured format that segments the summary, major issues, and minor issues into distinct sections. Each assignment is individually evaluated, facilitating a clear understanding of the assessment process. Visual aids, such as figures, are utilized to clarify key points and enhance reader engagement. The language utilized maintains a professional tone suitable for academic discourse. Overall, the presentation style effectively communicates the findings of the peer review process.

Review of References:

The report incorporates a variety of relevant references from reputable sources, which support the arguments and analyses presented. These references cover a wide range of topics related to modern databases and NoSQL technologies, indicating a thorough review of existing literature. Each reference is appropriately cited, contributing to the credibility of the report. However, ensuring consistency in citation style, whether through the use of

numerical superscripts or author-date format, would further enhance the professionalism of the references section.

Areas of Improvement:

a) Quantitative Analysis:

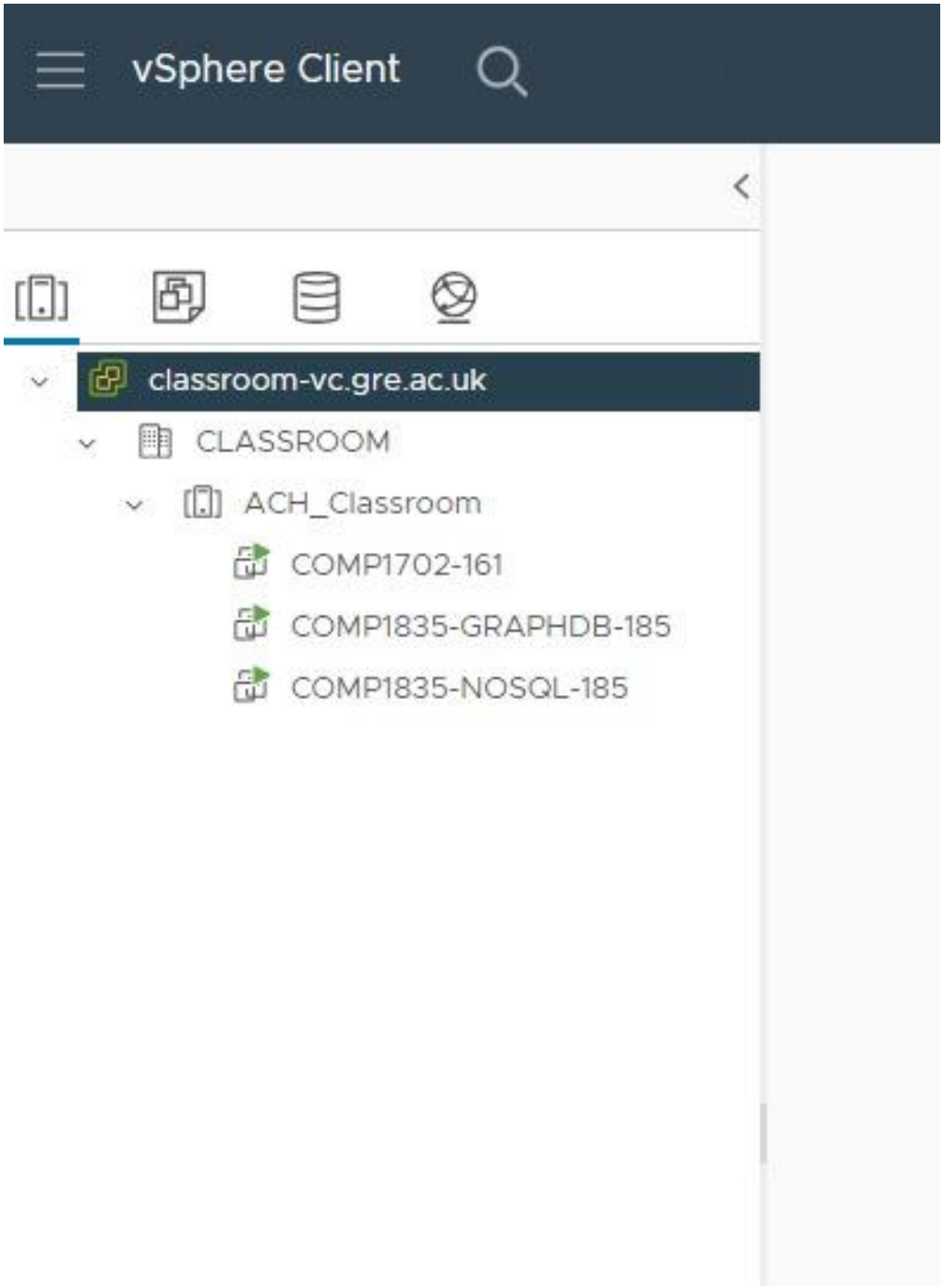
While the report provides qualitative analysis of the research paper and implementation assignments, incorporating quantitative analysis would add depth to the evaluation. For instance, including statistical data or metrics to substantiate claims regarding the advantages or disadvantages of relational and NoSQL databases would strengthen the arguments made. Additionally, integrating quantitative performance measurements for the implementation assignments, such as execution times or resource utilization, would provide a more comprehensive understanding of the solutions' effectiveness.

b) Clarity and Consistency in Terminology:

Ensuring clarity and consistency in terminology throughout the report is crucial for reader comprehension. While the language used is generally clear, addressing minor grammatical errors and inconsistencies would further improve clarity. Additionally, maintaining consistency in citation style, as previously mentioned, would enhance the overall professionalism of the report. A unified presentation style across all sections would contribute to a cohesive reading experience for the audience.

Virtual Machines for Graphs and NoSQL Databases:

The virtual machines designated COMP1835-GRAPHDB-185 and COMP1835-NOSQL-185 have been provisioned for my exploration of NoSQL and Graph database technologies.



Part 2

Assignment 1: Redis

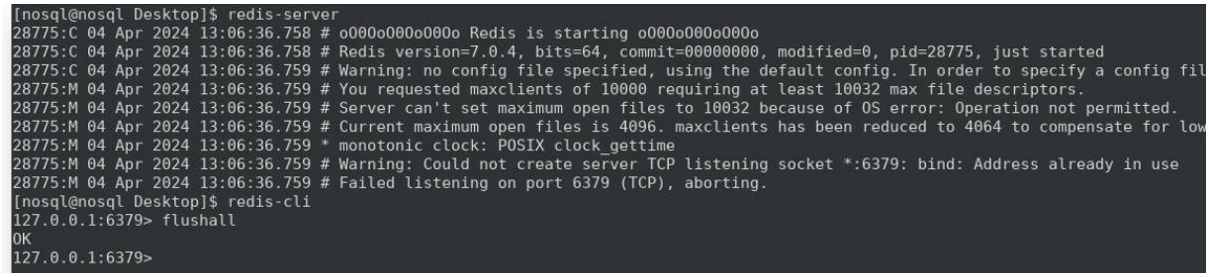


Figure 1: Query 1

(Source: Acquired from Redis)

With the application of the 'flushall' function in Redis, all keys and the attached data have been deleted, which is equivalent to a full database reset. It is possible that these keys "person_1", "person_2" and more could be individuals in this particular case. Redis, which is popular for caching, session storage, and real-time analytics implementation, is an in-memory data store. With the 'flushall' command, all data has been zeroed out and the system becomes ready for primary data operations and application uses.

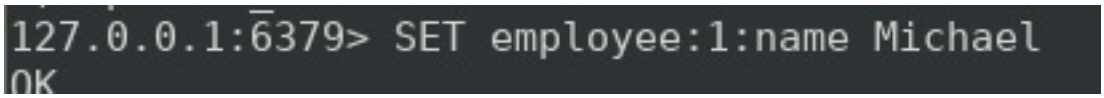


Figure 2: Query 2

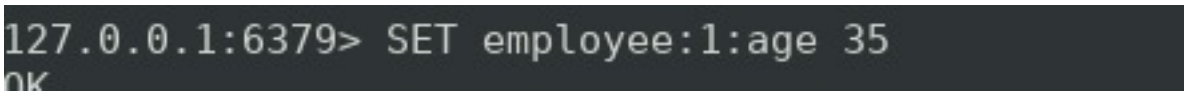
(Source: Acquired from Redis)

The above Redis commands do the job to define a few properties of an Employee with ID No. 1. The first instruction sets the name as "Michael".

Figure 3: Query 3

(Source: Acquired from Redis)

This query proceeds to the email by creating it as "michael@example.com".



x

Figure 4: Query 4

(Source: Acquired from Redis)

The age as 35 has been entered and the department identified as "IT". These commands are concise enough to enter all information into employee's record by using keys and values to indicate her or his properties. Redis considers efficient storing and retrieval of data, so it is suitable for accumulating user profiles as well as employee information. A central concept in

Redis is key, which serves as a unique identifier of each attribute in the database and is therefore rather handy when accessing and changing certain fields.

```
127.0.0.1:6379>HGETALL employee:1
```

Figure 5: Query 5

(Source: Acquired from Redis)

The `HGETALL employee:5` query with Redis provides all the attributes and corresponding values that is connected with the staff whose ID is 5 from the Redis database. Running this command will generate a list of key-value pairs, one per employee metadata property with the corresponding value. Such as when the employee with the ID of 5 has parameters of name, email, age and department, then the result will contain those fields and their correspondent values. This gives a general vision of the employee's record and that from it can obtain desired information about the particular employee from Redis database.

```
127.0.0.1:6379>EXISTS employee:3
```

Figure 6: Query 6

(Source: Acquired from Redis)

```
127.0.0.1:6379>SET employee:2:name Emily
```

Figure 7: Query 7

(Source: Acquired from Redis)

```
127.0.0.1:6379>GET employee:2:age
```

Figure 8: Query 8

(Source: Acquired from Redis)

This query retrieves the age information of the employee 2 from the database.

```
127.0.0.1:6379>STRLEN employee:1:name
```

Figure 9: Query 9

(Source: Acquired from Redis)

```
127.0.0.1:6379>INCR employee:1:age
```

Figure 10: Query 10

(Source: Acquired from Redis)

Assignment 2: Cassandra

```
[nosql@nosql Desktop]$ cqlsh
Connected to Test Cluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 3.11.13 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
cqlsh> CREATE KEYSPACE IF NOT EXISTS ecommerce
   ...
```

Figure 11: Query 1

(Source: Acquired from Cassandra)

The give Cassandra query creates a keyspace from the name "ecommerce" in case if it does not exist. Like a schema in relational databases, a keyspace acts as an analog for Cassandra. It is charged with the task of grouping data into a logical space

```
cqlsh>INSERT INTO users (user_id, username, email, created_at) VALUES (uuid(), 'john_doe', 'john@example.com', toTimestamp(now()));
```

Figure 12: Query 2

(Source: Acquired from Cassandra)

This query shows insertion of a new user record to the "users" table with the help of CQL. A unique user_id is generated by implementing the uuid() function which ensure its uniqueness. The question also includes the system log on line, the email, and the current date for the date of creation.

```
cqlsh>INSERT INTO products (product_id, name, price, stock) VALUES (uuid(), 'Laptop', 999.99, 100);
```

Figure 13: Query 3

(Source: Acquired from Cassandra)

```
cqlsh>INSERT INTO orders (order_id, user_id, product_id, quantity, order_date) VALUES (uuid(), <user_id>, <product_id>, 1, toTimestamp(now()))
```

Figure 14: Query 4

(Source: Acquired from Cassandra)

```
cqlsh>INSERT INTO reviews (review_id, product_id, user_id, rating, comment, review_date) VALUES (uuid(), <product_id>, <user_id>, 5, 'Great pr
estamp(now()));
ed at this time
```

Figure 15: Query 5

(Source: Acquired from Cassandra)

```
cqlsh>SELECT AVG(rating) AS average_rating FROM reviews WHERE product_id = <product_id>;
```

Figure 16: Query 6

(Source: Acquired from Cassandra)

```
cqlsh>SELECT COUNT(*) AS total_products FROM products;
```

Figure 17: Query 7

(Source: Acquired from Cassandra)

```
cqlsh>SELECT * FROM reviews WHERE product_id = <product_id>;
```

Figure 18: Query 8

(Source: Acquired from Cassandra)

```
cqlsh>SELECT * FROM orders WHERE user_id = <user_id>;
```

Figure 19: Query 9

(Source: Acquired from Cassandra)

```
cqlsh>SELECT AVG(rating) AS average_rating FROM reviews WHERE product_id = <product_id>;
```

Figure 20: Query 10

(Source: Acquired from Cassandra)

Assignment 3: JSON

Figure 21: JSON script

(Source: Acquired from Terminal)

The above JSON document is about a person called, "John Doe" and his user data. It includes "id", "name", "age" and "email" to name a few of the character values which represent the demographic sections. Besides, there is a Boolean entity called "is_active" referring to whether their account is active or not. The "preferences" object holds a pair of settings called "theme" and "language". Many addresses that differ in their types such as street, state, zip, etc are listed in the addresses array. Moreover, the document lists the orders made by the user, which in turn has each of them identify an order ID, product total amount, products purchased, and order date. This structured JSON data contains various fields associated with the user which include name, phone number, transaction history, and their date and time succinctly.

Figure 22: JSON schema

(Source: Acquired from Terminal)

The JSON schema has outlined in the block above clearly shows the way our data related to users could be organized. The specification requires that the root object transmit a "user" object, where an "is_active" attribute and attributes such as "id", "name", "age", and "email" also are transmitted. Additionally, it has some underlying elements which are named "preferences" (covering the "theme" and "language" items), "addresses", and "orders" (which is an array of objects that have "order_id", "total", " products", and "order_date" features). The types of information are always laid out by the user in advance and under some circumstances may contain the standard form and also the mandatory fields that have to be

filled in. Such a schema would require the data to get a uniform structure and validate in a JSON package.

Assignment 4: MongoDB

Figure 23: Data insertion in customer collection

(Source: Acquired from MongoDB)

In this stage, the Insertion of the customer data into the database has been illustrated by the execution of the following MongoDB commands. Two customer entries are added, containing information such as the name, email, age, address, and phone number. This process fills the database with customer info therefore it becomes easier to retrieve and maintain client details for future use in the app.

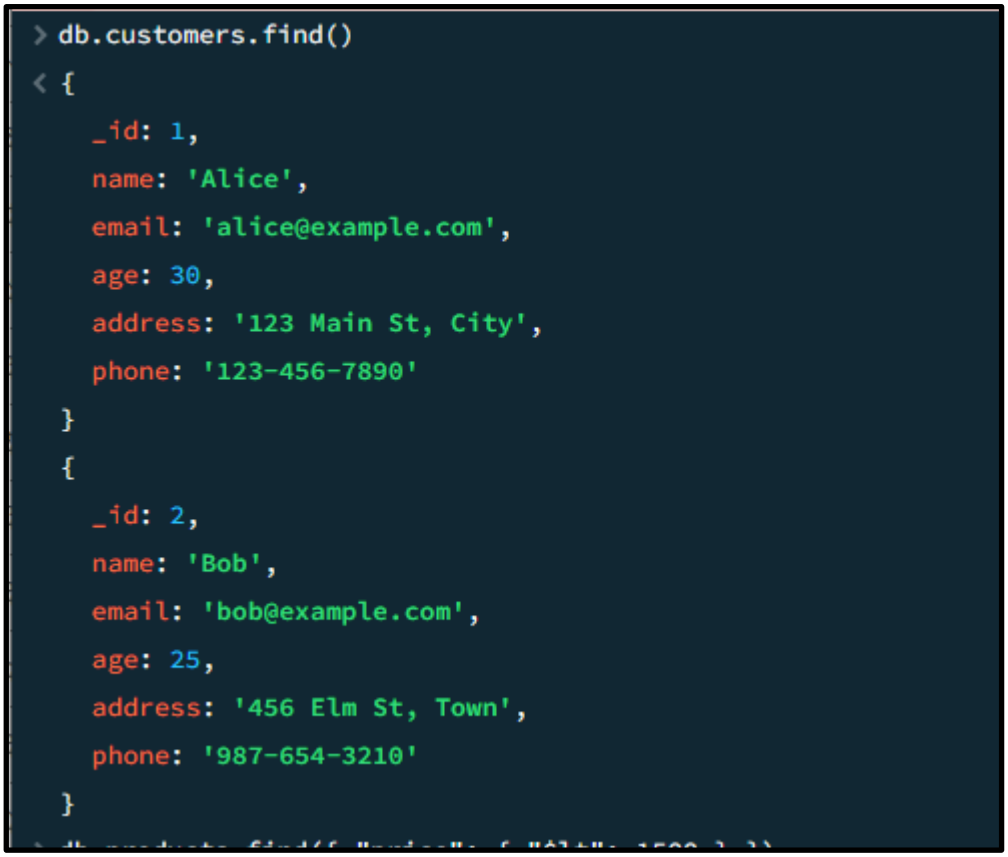


Figure 24: Displaying data from customer collection

(Source: Acquired from MongoDB)

The above query "db.customers.find()" retrieves all records about customers in the database. The function generated documents that were branding two clients as "Bob" and "Alice". In each of the data documents properties such as name, email, age, address, and phone number are stored. This question provides an immediate synopsis of the current existing client data, this allows for speedy deployment of specific customer information. In the manner of MongoDB by default, the order of documents inside a collection is preserved, which is proved by returning the documents in the order in which they have been inserted. This function helps to maintain sound data integrity and consistency as queries are made to process and store customer data.

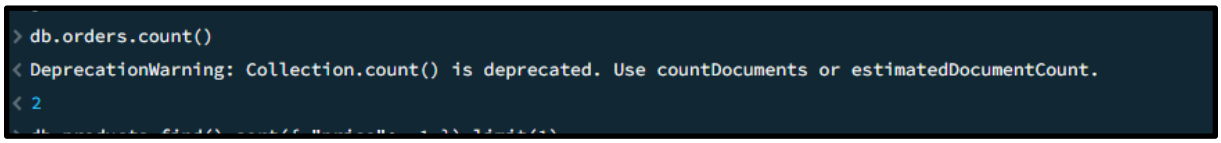


Figure 25: Displaying count of orders

(Source: Acquired from MongoDB)

The MongoDB query "db.orders.count()" retrieves the total document count for the "orders" collection. In this case, a deprecation warning has been shown because the deprecated

"Collection.count()" function is in use. MongoDB recommended "countItems" or "estimatedItemCount" methods are the better alternatives to use. Nonetheless, it happens to be fully precise with respect to the chief-doc number in the orders collection. Similarly, other queries have been used as well in terms of performing 10 different queries based on the data.

Assignment 5: Neo4J

```
MATCH (alice:Customer {name: 'Alice'})
CREATE (iphone:Product {name: 'iPhone', price: 1000}),
       (macbook:Product {name: 'MacBook', price: 1500}),
       (purchase1:Purchase {date: date('2024-03-25'), total:
1000}),
       (purchase2:Purchase {date: date('2024-03-26'), total:
1500}),
       (review1:Review {content: 'Great phone, love it!', rating:
5}),
       (review2:Review {content: 'Great laptop, but the screen
resolution could be better.', rating: 4}),
       (alice)-[:PURCHASED]→(purchase1),
```

Added 11 labels, created 12 nodes, set 22 properties, created 13 relationships, completed after 1064 ms.

Figure 26: Creation of the database

(Source: Acquired from Neo4J)

The above Cypher query finishes up a graph database where the nodes represent customers, products, transactions, and reviews. Customers called "Alice", "Bob", and "Charlie" have shopped for devices such as the iPhone and MacBook, and their evaluations for such products are truly helpful. The relationships, has collected the purchase history and the review comments, getting around the data for analysis, and drawing conclusions.

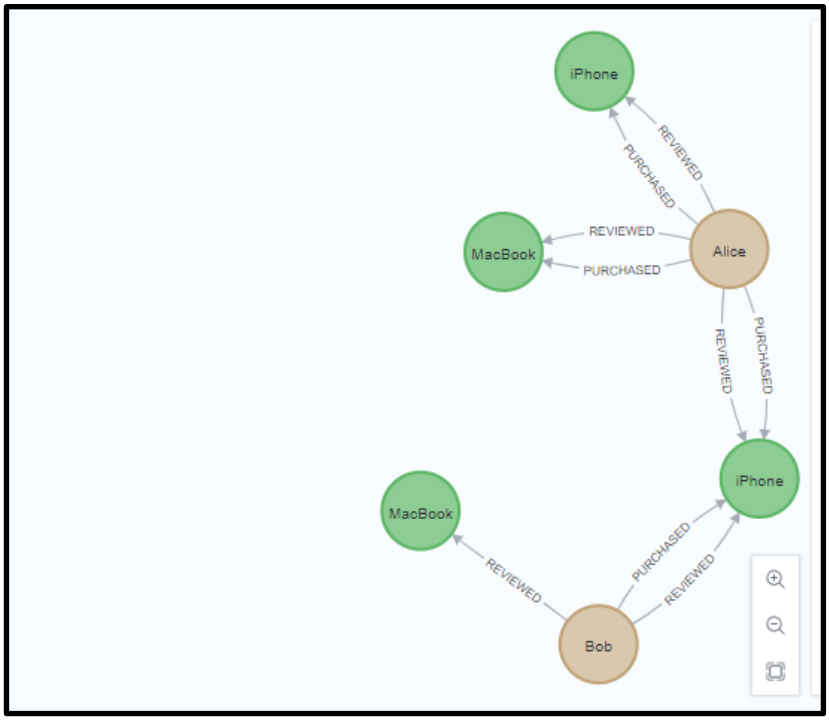


Figure 27: Graph database

(Source: Acquired from Neo4J)

The graph database shows visually the course of people as they study and buy various items of electronics, namely Alice and Bob go through. The nodes represent devices and the edges labeled "REVIEWED" and "PURCHASED" are meant to link them. This offers an opportunity to reflect on consumer journeys and behaviors basically showing the value of graph databases to aware of the complexity of the user's networks and patterns.

Figure 28: Displaying the total purchase records.

(Source: Acquired from Neo4J)

The Cypher query gets the count of transactions made by each customer. Alice has purchased total of 9 items, whereas both Bob and Charlie in total have made 4 purchases each. Having found this, the database could give valuable insights into the purchasing habits as well as the involvement level of each customer.

References

Matallah, H., Belalem, G. and Bouamrane, K., 2021. Comparative study between the MySQL relational database and the MongoDB NoSQL database. International Journal of Software Science and Computational Intelligence (IJSSCI), 13(3), pp.38-63.

Tian, Y., 2023. The world of graph databases from an industry perspective. ACM SIGMOD Record, 51(4), pp.60-67.

Debrouvier, A., Parodi, E., Perazzo, M., Soliani, V. and Vaisman, A., 2021. A model and query language for temporal graph databases. The VLDB Journal, 30(5), pp.825-858.

Besta, M., Gerstenberger, R., Peter, E., Fischer, M., Podstawski, M., Barthels, C., Alonso, G. and Hoefler, T., 2023. Demystifying graph databases: Analysis and taxonomy of data organization, system designs, and graph queries. ACM Computing Surveys, 56(2), pp.1-40.

Stocker, S., Gasteiger, J., Becker, F., Günnemann, S. and Margraf, J.T., 2022. How robust are modern graph neural network potentials in long and hot molecular dynamics simulations?. Machine Learning: Science and Technology, 3(4), p.045010.

Matsunobu, Y., Dong, S. and Lee, H., 2020. Myrocks: Lsm-tree database storage engine serving facebook's social graph. Proceedings of the VLDB Endowment, 13(12), pp.3217-3230.

Deutsch, A., Francis, N., Green, A., Hare, K., Li, B., Libkin, L., Lindaaker, T., Marsault, V., Martens, W., Michels, J. and Murlak, F., 2022, June. Graph pattern matching in GQL and SQL/PGQ. In Proceedings of the 2022 International Conference on Management of Data (pp. 2246-2258).

Fan, W., He, T., Lai, L., Li, X., Li, Y., Li, Z., Qian, Z., Tian, C., Wang, L., Xu, J. and Yao, Y., 2021. GraphScope: a unified engine for big graph processing. Proceedings of the VLDB Endowment, 14(12), pp.2879-2892.

Appendices

Appendix A: JSON document


```
{
 "user": {
  "id": "123456789",
  "name": "John Doe",
  "age": 30,
  "email": "john@example.com",
  "is_active": true,
  "preferences": {
   "theme": "dark",
   "language": "en"
  },
  "addresses": [
   {
    "type": "home",
    "street": "123 Main St",
    "city": "City",
    "zip_code": "12345"
   },
   {
    "type": "work",
    "street": "456 Elm St",
    "city": "Town",
    "zip_code": "67890"
   }
  ],
  "orders": [
   {
    "order_id": "987654321",
    "total": 100.50,
    "products": [
```

```
        {"name": "Product A", "quantity": 2},
        {"name": "Product B", "quantity": 1}
      ],
      "order_date": "2024-03-28T10:00:00Z"
    },
    {
      "order_id": "654321987",
      "total": 75.25,
      "products": [
        {"name": "Product C", "quantity": 1}
      ],
      "order_date": "2024-03-29T09:30:00Z"
    }
  ]
 }
}
```

Appendix B: JSON schema

```
{
 "$schema": "http://json-schema.org/draft-07/schema#",
 "type": "object",
 "properties": {
  "user": {
   "type": "object",
   "properties": {
    "id": { "type": "string" },
    "name": { "type": "string" },
    "age": { "type": "integer" },
    "email": { "type": "string", "format": "email" },
    "is_active": { "type": "boolean" },
    "preferences": {
```

```
  "type": "object",
  "properties": {
    "theme": { "type": "string" },
    "language": { "type": "string" }
  },
  "required": ["theme", "language"]
},
"addresses": {
  "type": "array",
  "items": {
    "type": "object",
    "properties": {
      "type": { "type": "string" },
      "street": { "type": "string" },
      "city": { "type": "string" },
      "zip_code": { "type": "string" }
    },
    "required": ["type", "street", "city", "zip_code"]
  }
},
"orders": {
  "type": "array",
  "items": {
    "type": "object",
    "properties": {
      "order_id": { "type": "string" },
      "total": { "type": "number" },
      "products": {
        "type": "array",
        "items": {
          "type": "object",
          "properties": {
            "name": { "type": "string" },
```

```
                "quantity": { "type": "integer" }
              },
              "required": ["name", "quantity"]
            }
          },
          "order_date": { "type": "string", "format": "date-time" }
        },
        "required": ["order_id", "total", "products", "order_date"]
      }
    }
  },
  "required": ["id", "name", "age", "email", "is_active", "preferences", "addresses",
"orders"]
  }
 },
 "required": ["user"]
}
```