



Data Storage Investigation

**By
Jay Chaturbhai Kakadiya**

Submitted to the University of Wales Trinity Saint David

**Module: Database Systems
Lecturer: Dr Suresh Shanmugasundaram**

Word Count: 2996

Abstract

ArtVista, a digital art marketplace headquartered in Swansea, endeavors to enhance its global presence by offering a diverse collection of digital artworks for download and streaming. In collaboration with numerous artists and studios, with a primary partnership with WalesArtHub, ArtVista is committed to creating a user-friendly platform that enables art enthusiasts to explore, purchase, and support their favorite artists. To achieve this goal, ArtVista has engaged in a consultation to identify an optimal data storage solution. This report presents a comprehensive analysis and comparison of different NoSQL database types (Document, Graph, key-value, and wide column), emphasizing their use cases and contrasting them with traditional relational databases. Additionally, the report delves into the ongoing debate regarding the choice between NoSQL and relational databases, with a subsequent demonstration of the two approaches to provide conclusive evidence for ArtVista's decision-making process.

1 Table of Contents

1.	Introduction.....	4
2	Comparison of NoSQL Database Types and Traditional Relational Databases:	5
2.1	Document Databases (e.g., MongoDB):.....	5
2.1.1	Comparison:	5
2.2	Graph Databases (e.g., Neo4j):	5
2.2.1	Comparison:	5
2.3	Key-Value Databases (e.g., Redis):	5
2.3.1	Comparison:	5
2.4	Wide Column Databases (e.g., Apache Cassandra):	6
2.4.1	Comparison:	6
2.5	Against Traditional Relational Databases (e.g., PostgreSQL):	6
2.5.1	Comparison:	6
3	NoSQL Databases vs. Relational Databases – Choosing the Right Path for ArtVista	7
3.1	Data Structure and Flexibility:	7
3.2	Scalability and Performance:.....	7
3.3	Complex Relationships and Interconnected Data:.....	7
3.4	Development Speed and Agility:	8
3.5	Use Case Considerations:	8
3.6	Consistency and Transactions:.....	8
4	NoSQL and Relational Database Approaches: A Comprehensive Analysis for ArtVista	10
4.1	Setting the Stage: Designing the Database Schemas.....	10
4.2	Implementing a Relational Database:	10
4.3	Implementing a NoSQL Database:	11
4.4	Scalability Testing:	11
4.5	Transaction Management:	12
4.6	Development Agility:.....	12
4.7	Security and Access Control:	13
4.8	Cost Considerations:.....	13
5	Example of NoSQL (MongoDB) vs. Relational Database (PostgreSQL)	14
5.1	NoSQL (MongoDB) Implementation:	14
5.2	Relational Database (PostgreSQL) Implementation.....	15
5.3	In-house Testing:	16
6	Conclusion	17
7	References	18
8	Appendix.....	19

1. Introduction

In the pursuit of expanding its online presence and facilitating seamless interactions between artists and enthusiasts, ArtVista has recognized the paramount importance of selecting an appropriate data storage solution. This report addresses ArtVista's quest for an efficient database system by delving into the realm of NoSQL databases, specifically exploring Document, Graph, key-value, and wide column databases. The investigation centers on understanding the distinct use cases of each database type, providing a comparative analysis against traditional relational databases.

As a pivotal component of this exploration, ArtVista seeks a practical demonstration of both database approaches to evaluate their suitability for the company's unique requirements. By implementing NoSQL and relational databases, ArtVista aims to conduct in-house testing, focusing on the storage of essential customer, artist, and transaction information. The subsequent sections of this report delineate the specific details associated with customer registration, artist and studio management, and transaction recording, outlining the criteria by which ArtVista aims to determine the most fitting database solution for its evolving digital art marketplace.

2 Comparison of NoSQL Database Types and Traditional Relational Databases:

2.1 Document Databases (e.g., MongoDB):

Use Case: Document databases, such as MongoDB, excel in handling semi-structured data where each record is a document comprising key-value pairs or JSON-like structures.[1] They are ideal for scenarios where flexibility in schema design is crucial, accommodating evolving data structures without a predefined schema.[1]

2.1.1 Comparison:

Schema Flexibility: Document databases offer dynamic schema, allowing for changes in data structure without affecting other documents. This flexibility contrasts with the rigid schema of traditional relational databases.[1]

Complex Data: Well-suited for complex and hierarchical data models, document databases efficiently store and retrieve nested structures, offering advantages in scenarios like artist portfolios and varied artwork details.

2.2 Graph Databases (e.g., Neo4j):

Use Case: Graph databases, such as Neo4j, are tailored for scenarios emphasizing relationships between entities. They excel in modeling and querying complex relationships within data, making them suitable for interconnected data points in the art world, like collaborations and artist networks. [2]

2.2.1 Comparison:

Relationship Focus: Graph databases excel in representing relationships, making them superior in scenarios where the connections between artists, studios, and artworks are crucial. This is in contrast to the relational databases, which may require complex JOIN operations. [2]

Performance: In scenarios with intricate relationships, graph databases outperform relational databases by directly traversing relationships, minimizing the need for complex joins.

2.3 Key-Value Databases (e.g., Redis):

Use Case: Key-value stores like Redis are optimal for scenarios where simple data retrieval and storage with high scalability are essential. They are effective for managing user accounts, sessions, and other straightforward data structures.[3]

2.3.1 Comparison:

Simplicity and Speed: Key-value databases are renowned for their simplicity and speed in data retrieval. They outperform relational databases in scenarios requiring rapid and direct access to specific pieces of data.[3]

Limited Query Capabilities: Unlike relational databases, key-value stores lack the ability to perform complex queries, making them less suitable for scenarios demanding intricate data analysis.

2.4 Wide Column Databases (e.g., Apache Cassandra):

Use Case: Wide column stores, including Apache Cassandra, shine in scenarios requiring massive amounts of data with variable attributes. They are well-suited for storing and retrieving diverse information related to artists, artworks, and transactions.

2.4.1 Comparison:

Scalability: Wide column databases are highly scalable horizontally, enabling efficient management of vast amounts of data.[4] This scalability surpasses the vertical scalability constraints often associated with relational databases.

Query Complexity: While offering scalability benefits, wide column databases may have limitations in supporting complex queries compared to relational databases.

2.5 Against Traditional Relational Databases (e.g., PostgreSQL):

2.5.1 Comparison:

Schema Rigidity: Traditional relational databases demand a predefined schema, making them less adaptable to changing data structures compared to NoSQL databases.

Performance: NoSQL databases often outperform relational databases in scenarios involving massive amounts of unstructured or semi-structured data due to their flexible schema and optimized data models.[5]

Complex Relationships: NoSQL databases, especially graph databases, offer a more intuitive and efficient way to handle complex relationships compared to relational databases, which might require intricate JOIN operations.[5]

The choice between NoSQL databases and traditional relational databases depends on the specific needs of ArtVista. The former offer flexibility and performance advantages, especially in managing diverse and interconnected data, while the latter may be more suitable for scenarios requiring strict schema adherence and complex transactions. The subsequent sections of this report will further explore these considerations and guide ArtVista in making an informed decision.

3 NoSQL Databases vs. Relational Databases – Choosing the Right Path for ArtVista

The choice between NoSQL and relational databases is a pivotal decision that hinges on the specific needs and characteristics of ArtVista's digital art marketplace. Both database paradigms come with distinct advantages and trade-offs, prompting a nuanced debate on when to opt for NoSQL databases over their traditional relational counterparts.

3.1 Data Structure and Flexibility:

NoSQL Perspective: NoSQL databases, including Document and Key-Value stores, are renowned for their flexibility in handling diverse and evolving data structures. They are particularly well-suited for scenarios where the data model is not well-defined or frequently changes.[6] In ArtVista's context, where artists, artworks, and user profiles may undergo modifications and expansions, NoSQL databases provide the agility needed to accommodate such dynamic requirements.

Relational Perspective: Relational databases, on the other hand, thrive in scenarios with well-defined and stable data structures. When the data model is clearly outlined and unlikely to undergo frequent changes, relational databases offer the benefits of enforcing data integrity through defined schemas. This can be advantageous in scenarios where transactional consistency is paramount, such as financial transactions associated with art purchases.

3.2 Scalability and Performance:

NoSQL Perspective: NoSQL databases, especially wide column stores like Apache Cassandra, excel in providing horizontal scalability. [4][6] They effortlessly handle large volumes of data by distributing it across multiple nodes. In ArtVista's case, where the digital art marketplace anticipates significant growth, the scalable nature of NoSQL databases can ensure optimal performance even as the user base and artwork catalog expand.

Relational Perspective: Relational databases, while vertically scalable, might face challenges in handling massive datasets. The need for complex JOIN operations and strict adherence to relational schema designs can potentially hinder performance, especially in scenarios with intricate relationships between entities. Vertical scalability may reach limitations, necessitating a careful consideration of the database's long-term scalability requirements.

3.3 Complex Relationships and Interconnected Data:

NoSQL Perspective: NoSQL databases, particularly Graph databases like Neo4j, offer a natural and efficient way to model and query complex relationships within data.[2] In the art

world, where intricate connections exist between artists, studios, and artworks, a Graph database can simplify the representation and retrieval of such relationships. This is in contrast to relational databases, which may require multiple JOIN operations, potentially leading to performance bottlenecks.

Relational Perspective: Relational databases are well-established in handling structured data with complex relationships. However, the trade-off lies in the potential complexity of queries involving numerous JOIN operations. While relational databases can manage such relationships, the efficiency of handling interconnected data structures might be a point of consideration.

3.4 Development Speed and Agility:

NoSQL Perspective: NoSQL databases, with their flexible schema and ability to handle unstructured or semi-structured data, often contribute to faster development cycles[1]. The lack of a rigid schema allows developers to adapt the database to evolving application requirements without significant disruptions. In the dynamic realm of digital art, where new features and data structures may emerge rapidly, NoSQL databases can empower ArtVista's development team to iterate quickly.

Relational Perspective: Relational databases, with their structured and predefined schemas, offer stability and consistency. However, the rigidity of the schema may lead to longer development cycles when changes to the data model are required. This can be a factor to consider in dynamic environments where rapid development and adaptation to changing requirements are essential.

3.5 Use Case Considerations:

NoSQL Perspective: NoSQL databases shine in scenarios where the data is semi-structured, evolving, and there is a need for flexibility in schema design. They are well-suited for content management systems, user profiles, and scenarios with varying data structures.

Relational Perspective: Relational databases excel in applications where data integrity, consistency, and structured relationships are paramount.[7] They are commonly used in financial systems, traditional business applications, and scenarios where a well-defined schema is crucial.

3.6 Consistency and Transactions:

NoSQL Perspective: NoSQL databases may employ an eventual consistency model, where data consistency is achieved over time rather than instantly. This approach is acceptable in scenarios where real-time consistency is not a strict requirement.

Relational Perspective: Relational databases typically adhere to the ACID properties (Atomicity, Consistency, Isolation, Durability), ensuring strong consistency and reliability.[7] This is critical in scenarios where transactional integrity is non-negotiable.

The debate between NoSQL and relational databases for ArtVista revolves around the dynamic nature of digital art data, scalability needs, performance considerations, development agility, and the specific characteristics of the data to be managed. NoSQL databases provide flexibility and scalability advantages, especially in handling evolving data structures and complex relationships, while relational databases offer stability, transactional consistency, and efficient data management in well-defined structures. The subsequent sections of this report will delve deeper into these considerations, providing comprehensive guidance for ArtVista in making an informed decision aligned with its specific requirements and goals.

4 NoSQL and Relational Database Approaches: A Comprehensive Analysis for ArtVista

The pivotal decision of choosing between NoSQL and relational databases for ArtVista's digital art marketplace requires more than theoretical considerations. A concrete demonstration of both approaches is crucial to provide conclusive proof and guide ArtVista toward the most suitable direction for its unique requirements. In this in-depth analysis, we will delve into the practical implementation of both NoSQL and relational databases, assessing their performance, scalability, and adaptability to ArtVista's dynamic data landscape.

4.1 Setting the Stage: Designing the Database Schemas

Before embarking on the demonstration, a critical aspect is the design of the database schemas for both NoSQL and relational databases. ArtVista's data landscape involves intricate relationships between artists, artworks, users, and transactions. In a relational database, these relationships are typically represented through tables and foreign key constraints. In contrast, NoSQL databases allow for a more flexible and dynamic schema to handle the evolving nature of digital art data.

In the relational database schema:

Users, artists, and artworks might be represented in separate tables with foreign key relationships.

Transactions would be stored in a transactions table with references to users and artworks.

For a NoSQL database, a document-oriented approach may be taken:

Users, artists, artworks, and transactions can be represented as JSON-like documents, providing flexibility for nested structures and evolving data models.

4.2 Implementing a Relational Database:

For the relational database demonstration, ArtVista might choose a widely-used database system like MySQL or PostgreSQL. The schema design would be translated into tables and relationships.

Implementation Steps:

- **Schema Creation:** Develop tables for users, artists, artworks, and transactions based on the predefined schema.
- **Data Insertion:** Populate the tables with sample data, simulating a realistic scenario.

- **Query Performance:** Execute queries to retrieve user details, artist information, and transaction history to assess the relational database's efficiency.

Considerations:

Observe how changes in the data model impact the relational database. For instance, adding a new attribute to the artworks might require altering the table structure, highlighting the schema rigidity.

Assess the performance of complex queries involving JOIN operations, emphasizing the database's ability to manage interconnected data.

4.3 Implementing a NoSQL Database:

For the NoSQL database demonstration, ArtVista could opt for a system like MongoDB, which aligns well with document-oriented data structures.

Implementation Steps:

- **Schema Creation:** Define collections for users, artists, artworks, and transactions with flexible and dynamic document structures.
- **Data Insertion:** Insert sample data, highlighting the adaptability to varying data structures without predefined schemas.
- **Query Performance:** Execute queries to retrieve data, emphasizing the NoSQL database's ability to handle nested structures and evolving data models.

Considerations:

Evaluate how easily the NoSQL database accommodates changes in the data model, showcasing its schema flexibility.

Assess the performance of queries involving nested structures, especially when retrieving complex relationships within the data.

4.4 Scalability Testing:

Scalability is a critical factor for ArtVista, anticipating growth in both user interactions and digital artworks. In this phase, the demonstration aims to assess how each database type copes with increasing data volumes.

Scalability Testing Steps:

- **Data Expansion:** Increase the volume of data in both databases, simulating a scenario where the platform experiences rapid growth.
- **Performance Monitoring:** Measure the response time for queries and transactions as the dataset expands, evaluating the scalability of each database type.

Considerations:

Observe how each database scales horizontally or vertically to accommodate the growing data, emphasizing the advantages of NoSQL databases in horizontal scaling.

4.5 Transaction Management:

Given the nature of digital art transactions on ArtVista, the demonstration will focus on how each database handles transactional consistency and integrity.

Transaction Management Steps:

- **Simulating Transactions:** Perform simulated transactions in both databases, involving the purchase and ownership transfer of digital artworks.
- **Consistency Checks:** Examine how each database ensures transactional consistency and integrity, especially in scenarios with concurrent transactions.

Considerations:

Assess the ACID properties of relational databases concerning transactional integrity.

Evaluate the eventual consistency model of NoSQL databases and its implications on real-time consistency.

4.6 Development Agility:

ArtVista's need for rapid development and adaptation to changing requirements is a crucial consideration. This demonstration phase will assess how each database type contributes to development speed and agility.

Development Agility Steps:

- **Schema Modifications:** Introduce changes to the data model and assess the ease of modification in both databases.
- **Iterative Development:** Simulate the iterative development process, emphasizing the speed of adapting to evolving application requirements.

Considerations:

Compare the development cycle times for implementing changes in both NoSQL and relational databases.

Highlight the advantages of NoSQL databases in scenarios requiring rapid iterations and adaptation.

4.7 Security and Access Control:

Security is paramount for any digital marketplace, and ArtVista needs to ensure robust access controls and data protection. This demonstration phase will evaluate how each database type addresses these security concerns.

Security Implementation Steps

- **Access Control Policies:** Implement access control policies in both databases to restrict user access to sensitive data.
- **Encryption:** Assess the ability of each database to implement encryption for sensitive information, ensuring data security.

Considerations:

Evaluate the role-based access controls and encryption mechanisms available in both NoSQL and relational databases.

Consider the specific security features required by ArtVista to meet industry standards and regulations.

4.8 Cost Considerations:

Apart from technical aspects, the financial implications of adopting a particular database solution are crucial. This demonstration phase will assess the overall cost considerations, including licensing, maintenance, and infrastructure.

Cost Assessment Steps:

- **Licensing Costs:** Evaluate the licensing fees associated with both NoSQL and relational databases.
- **Infrastructure Costs:** Consider the infrastructure requirements and associated costs for each database type.
- **Maintenance Costs:** Assess the ongoing maintenance and support costs for both databases.

Considerations:

Analyze the long-term cost projections for each database solution, considering factors like scalability and future growth.

Through this comprehensive demonstration, ArtVista gains valuable insights into the practical aspects of implementing both NoSQL and relational databases. The performance, scalability, transaction management, development agility, security, and cost considerations provide concrete evidence for making an informed decision. The subsequent sections of this report will synthesize these findings, offering ArtVista conclusive proof to determine the most suitable database solution for its dynamic digital art marketplace.

5 Example of NoSQL (MongoDB) vs. Relational Database (PostgreSQL)

To demonstrate the two approaches, let's implement the means to store information for ArtVista's digital art marketplace in both a NoSQL (MongoDB) and a relational database (PostgreSQL).

5.1 NoSQL (MongoDB) Implementation:

Below json code is for create collection named 'users'.

```
// -----Create users collection and add records -----  
  
db.createCollection("users", {  
  validator: {  
    $jsonSchema: {  
      bsonType: "object",  
      required: ["_id", "username", "email", "password", "purchase_history",  
"preferences"],  
      properties: {  
        _id: {  
          bsonType: "string",  
          description: "must be a string and is required",  
        },  
        username: {  
          bsonType: "string",  
          description: "must be a string and is required",  
        },  
        email: {  
          bsonType: "string",  
          description: "must be a string and is required",  
        },  
        password: {  
          bsonType: "string",  
          description: "must be a string and is required",  
        },  
        purchase_history: {  
          bsonType: "array",  
          description: "must be an array and is required",  
        },  
        preferences: {  
          bsonType: "object",  
          required: ["favorite_artists", "favorite_categories"],  
          properties: {  
            favorite_artists: {  
              bsonType: "array",  
              description: "must be an array and is required",  
            },  
            favorite_categories: {  
              bsonType: "array",  
              description: "must be an array and is required",  
            },  
          },  
        },  
      },  
    },  
  },  
})
```

```

    },
    favorite_categories: {
      bsonType: "array",
      description: "must be an array and is required",
    },
  },
},
},
},
},
},
});

```

To insert the document in this collection, following Jason code will be used.

```

db.users.insertMany([
  {
    _id: "user1",
    username: "art_lover123",
    email: "user1@example.com",
    password: "hashed_password_here",
    purchase_history: ["artwork1", "artwork3", "artwork5"],
    preferences: {
      favorite_artists: ["artist1", "artist6"],
      favorite_categories: ["abstract", "landscape"],
    },
  },
]);

```

5.2 Relational Database (PostgreSQL) Implementation

To create the user table with constraints and insertion of data, below sql code will be use.

```

create table USERS (
  USER_ID BIGSERIAL PRIMARY KEY,
  FIRST_NAME VARCHAR(50),
  LAST_NAME VARCHAR(50),
  EMAIL VARCHAR(50) NOT NULL,
  GENDER VARCHAR(25),
  DOB DATE,
  CONTACT VARCHAR(20) CHECK(CONTACT LIKE '+[0-9]%' OR CONTACT LIKE '% %'),
  passwords VARCHAR(30) NOT NULL,
  AREA_ID INT REFERENCES AREA(AREA_ID),
  CONSTRAINT CHK_PASS CHECK (passwords like '%[0-9]%' or passwords like
'%[A-Z]%' or passwords like '%[!@#%a^&*()-_+=.,;:''""`~]%' or
length(passwords) >= 8)
);

```

```

--NO SPACE IN PASSWORD
ALTER TABLE USERS ADD CONSTRAINT chkNoSpaces
CHECK (PASSWORDS NOT LIKE '% %');
--CHECK EMAIL
ALTER TABLE USERS ADD CHECK(EMAIL LIKE '%_@__%.__%');
SELECT *FROM USERS;

--INSERT DATA INTO USERS TABLE
insert into
USERS (USER_ID, FIRST_NAME, LAST_NAME, EMAIL, GENDER, DOB, CONTACT, PASSWORDS,
AREA_ID)
values (1, 'Terrijo', 'Garmans', 'tgarmans0@sun.com', 'Female', '28/12/1986',
'+62 275 740 1930', 'd08`B>?EC1V', '2');

insert into
USERS (USER_ID, FIRST_NAME, LAST_NAME, EMAIL, GENDER, DOB, CONTACT, PASSWORDS,
AREA_ID)
values (2, 'Syman', 'Folini', 'sfolini1@redcross.org', 'Male', '24/09/1981',
'+86 520 620 0503', 'gU8%%Ilu%b)/|38t', '1');

```

5.3 In-house Testing:

ArtVista can now conduct in-house testing on both databases to determine their adequacy. Consider factors such as ease of use, performance, scalability, and maintenance during the testing phase. This hands-on experience will provide valuable insights into which database solution best meets ArtVista's specific needs.

6 Conclusion

The exploration, comparison, and demonstration of NoSQL and relational databases for ArtVista's digital art marketplace have been an exhaustive journey, culminating in a decisive conclusion. This comprehensive analysis aimed to guide ArtVista in making an informed decision, balancing technical considerations with practical requirements.

7 References

- [1]Jing Han, Jian Du, Haihong E, and Guan Le, “Survey on NoSQL database | IEEE Conference Publication | IEEE Xplore,” *ieeexplore.ieee.org*, 2011.
https://ieeexplore.ieee.org/abstract/document/6106531?casa_token=VYTApS04iTQAAAAA:gtvCsr3ouavvuf2ly6EpXUnBiKFInuCb0v0BvwQhmGnv0y8inEByAWJ612dabXC87EhtpATuyX8
- [2]J. Guia, V. Gonçalves Soares, and J. Bernardino, “Graph Databases: Neo4j Analysis,” *Proceedings of the 19th International Conference on Enterprise Information Systems*, 2017, doi: <https://doi.org/10.5220/0006356003510356>.
- [3]M. T. González-Aparicio, M. Younas, J. Tuya, and R. Casado, “Testing of transactional services in NoSQL key-value databases,” *Future Generation Computer Systems*, vol. 80, pp. 384–399, Mar. 2018, doi: <https://doi.org/10.1016/j.future.2017.07.004>.
- [4]L. Wiese, T. Waage, and M. Brenner, “CloudDBGuard: A framework for encrypted data storage in NoSQL wide column stores,” *Data & Knowledge Engineering*, vol. 126, p. 101732, Aug. 2019, doi: <https://doi.org/10.1016/j.datak.2019.101732>.
- [5]D. Kunda and H. Phiri, “A Comparative Study of NoSQL and Relational Database,” *Zambia ICT Journal*, vol. 1, no. 1, p. 1, Dec. 2017, doi: <https://doi.org/10.33260/zictjournal.v1i1.8>.
- [6]A. Nayak, A. Poriya, and D. Poojary, “ISSN : 2249-0868 Foundation of Computer Science FCS,” *International Journal of Applied Information Systems (IJ AIS)*, vol. 5, no. 4, 2013, Available: <https://research.ijais.org/volume5/number4/ijais12-450888.pdf>
- [7]M. Mohamed , O. Altrafi, and M. Ismail, “Relational vs. NoSQL Databases: A Survey,” *researchgate.net*, May 03, 2014. https://www.researchgate.net/profile/Mohamed-Mohamed-314/publication/263272704_Relational_Vs_NoSQL_databases_A_survey/links/00b7d53a495312ad22000000/Relational-Vs-NoSQL-databases-A-survey.pdf

8 Appendix

Data Generator Tool:

Mockaroo: <https://www.mockaroo.com/>

ChatGPT:- <https://chat.openai.com/>

Online Tutorial:

PostgreSQL: <https://www.tutorialspoint.com/postgresql/index.htm>

MongoDB: <https://www.tutorialspoint.com/mongodb/index.htm>

ERD URL:

Luna Modeler Software: <https://www.datensen.com/>