Fachhochschule für Oekonomie und Management
IT Infrastructure
Prof. Dr. Karl Kurbel
WS 2022
BWIt_BS WS20 B

NoSQL Databases for Multimedia Applications

Seminar Paper

Presented by:	Emil Triest Westendallee 97A 14052 Berlin emil@triest.de
Field of Studies:	Bachelor of Science (B.Sc.) - Wirtschaftsinformatik - Business Information Systems - 5. Semester
Matriculation Number:	566224
Reviewed By:	Prof. Dr. Karl Kurbel
Place and Date:	Berlin, 04.03.2023

Table of Contents

Index of Tables

Index of Figures

Index of Abbreviations

Abbreviation	Definition
API	Application Programming Interface
AWS	Amazon Web Services
CAP	Consistency, Availability, Partition Tolerance
CRUD	Create, Read, Update and Delete
DB	Database
DBMS	Database Management System
GB	Gigabyte
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
NoSQL	Nonrelational Databases, commonly interpreted as ‘Not only SQL’
RAM	Random Access Memory
RDBMS	Relational Database Management System
SQL	Structured Query Language
SSD	Solid State Drive

1. Introduction

1.1 What are NoSQL Databases?

The encyclopedia of database systems defines a database as a "*structured* and *persistent* collection of information about some aspect of the real world organized and stored in a way that facilitates efficient retrieval and modification. The structure of a database is determined by an abstract data model. Primarily, it is this structure that differentiates a database from a data file." [1]

NoSQL is commonly interpreted as an acronym for 'Not only SQL'. Although "the term .. was first used in 1998 for a database that (although relational) did not have an SQL interface" [i] (p. 201), the definition for the term has since developed into "any *nonrelational data management approaches* meeting two criteria: Firstly: The data is not stored in tables. Secondly: The database language is not SQL." [i] (p. 16)

The common interpretation 'Not only SQL' does however, in contradiction to the above-mentioned definition, implicate that SQL can still be used in combination with NoSQL database systems. The combination of these two technologies is called polyglot persistence.

For a NoSQL database system to be considered as such, it must meet a list of requirements: [i] (p.18) "

- **Model:** The underlying database model is not relational.
- **At least three Vs**: The database system includes a large amount of data (volume), flexible data structures (variety), and real-time processing (velocity).
- **Schema**: The database management system is not bound by a fixed database schema.
- **Architecture**: The database architecture supports massively distributed web applications and horizontal scaling.
- **Replication**: The database management system supports data replication.
- **Consistency assurance**: According to the CAP (Consistency, Availability, Partition Tolerance) theorem, consistency may be ensured with a delay to prioritize high availability and partition tolerance."

[1] Özsu, M.T. (2018). Database. In: Liu, L., Özsu, M. (eds) Encyclopedia of Database Systems. Springer, New York, NY. https://doi.org/10.1007/978-1-4899-7993-3_80734-1

1.2 Short History of Nonrelational Databases

„Before Ted Codd's introduction of the relational model, nonrelational databases such as hierarchical or network-like databases existed" [i] (p. 16), however these were only sparsely used for specific use-cases, while Ted Codd's relational model grew immensely in popularity and widespread appeal. With the rise of the internet and web-based applications however, the scale of 'nonrelational vs relational' began to tip back to nonrelational. The same factors that had caused widespread appeal for relational databases such as data consistency and security now proved problematic, due to their high requirements of work and processing power. The larger the amount of data, the more apparent the problem became: "Big Data applications with relational database technology is difficult to impossible."[i] (p. 16)
The age of the internet had begun, meaning efficiency and performance now trumped consistency and nonrelational concepts were on the rise.

1.3 The most common types of NoSQL Databases

The four most common database models for NoSQL Databases are: [i] (p. 202) "

- Key-value stores
- Column family databases
- Document Stores
- Graph Databases"

1.4 Multimedia & Big Data

Multimedia can be defined as "the use of a combination of text, graphic and audio aids in the presentation of information." [2] With almost every modern application using a combination of text, graphics and audio aids to present information, it has become more difficult to find an example of an application, that isn't a multimedia application, than one that is. Social Media, modern websites and streaming platforms are all examples of multimedia applications.

[2] "Introduction to Computer Science, 2nd Edition." O'Reilly Online Learning. Pearson Education India. Accessed February 25, 2023. https://www.oreilly.com/library/view/introduction-to-computer/9788131760307/xhtml/chapter015.xhtml#head0570

The Gartner Glossary defines Big Data as “high-volume, high-velocity and/or high-variety information assets that demand cost-effective, innovative forms of information processing that enable enhanced insight, decision making, and process automation.” [3] This definition, in combination with the knowledge that relational databases tend to lack speed and performance for large amounts of data, shows where nonrelational or ‘NoSQL’ databases come in to play. It is safe to say that in general, for the use-case of multimedia applications, NoSQL databases tend to be the smarter choice due to their efficiency and performance handling large amounts of data of various types.

1.5 Research Objective

The background information above led to the following research objective:
to find out which type of NoSQL database model is best suited for different types of multimedia applications and different types of application scenarios.

[3] Gartner_Inc, “Definition of Big Data - Gartner Information Technology Glossary,” Gartner, accessed February 25, 2023, https://www.gartner.com/en/information-technology/glossary/big-data

2. Analysis of NoSQL database models

While there are plenty of NoSQL database models in existence, for the purpose of this paper the 'core NoSQL models' from section **1.3** will be used for further analysis. Although this will omit many other database models from the analysis, it will allow for a more detailed and in-depth analysis of these four most common NoSQL database models.

2.1 Key-Value Stores

A database can be defined as a key-value store database, if it has the following properties: [i] (p. 203) "

- There is a set of identifying data objects, the keys.
- For each key, there is exactly one associated descriptive data object, the value for that key.
- Specifying a key allows querying the associated value in the database."

Currently, the most popular key-value store[ii] DBMS is 'redis'[4].
An important note to make here is that in comparison to relational databases, where the schema of the data is predefined and has to be altered in order to accommodate new attributes, key-value stores do not have these restrictions and offer a lot more flexibility with regards to the values that are stored. While this is extremely useful for heterogenous data, it can be a cause for troubles with regards to data management.

Key	Value
Example-user	"Emil Triest"
Example-image.jpeg	<image file>
Example-song.mp3	<audio file>
Example-document-file.pdf	<pdf document>

Table 1 - Table showing an example of a key value store database with different typed value data

Additionally, since there is no need to check referential integrity[5], read and write operations can be performed at much faster rates. Concepts like 'in-memory'[6] and 'sharding'[7] can be

[4] https://redis.io/
[5] "Referential Integrity," IBM, accessed February 28, 2023, https://www.ibm.com/docs/en/informix-servers/14.10?topic=integrity-referential.
[6] Claude Aveline, "What Is an In-Memory Database?," Amazon (Corti, 1967), accessed February 28, 2023, https://aws.amazon.com/de/nosql/in-memory/.
[7] "Sharding," Sharding - MongoDB Manual, accessed February 28, 2023, https://www.mongodb.com/docs/manual/sharding/.

used in connection with key-value store databases to even further improve efficiency and speed.

2.2 Column-Family Stores

Column-family stores can be defined as NoSQL tables that meet the following requirements: [i] (p. 207) “

- The data is stored in multidimensional tables.
- Data objects are addressed with row keys.
- Object properties are addressed with column keys.
- Columns of the tables are grouped into column families.
- A table’s schema only refers to the column families; within one column family, arbitrary column keys can be used.
- In distributed, fragmented architectures, the data of a column family is preferably physically stored at one place (co-location) in order to optimize response times.”

Although the imagination of this concept may be a little complicated, a way to describe Column-family stores is as the inverse of a relational table. Rather than storing the data in fixed tables the column-family approach creates, as the name already reveals, column families. Similar to key-value stores, a key is defined for a set of data. These sets of data are grouped in column families which allows for a more clearly defined structure.

At the time of this writing, ‘Cassandra’[8] is the most popular column-family DBMS[ii]. The following table compares traditional relational databases to Cassandra:

Relational Database	Cassandra
Database	Keyspace
Database instance	Cluster
Table	Column family
Row	Row
Column	Column

Table 2 - Table comparing features of relational databases to Cassandra, a popular NoSQL database

It is however important to note, that while with relational databases each row has the same columns, this is not the case for Cassandra and column-family databases in general. This concept is outlined in Figure 1.

[8] https://cassandra.apache.org/_/index.html

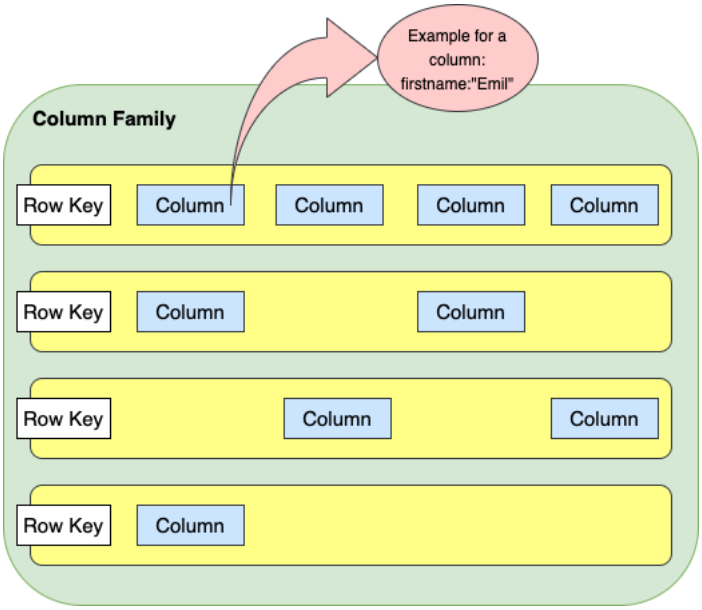

Figure 1 - Diagram showing an example of a column family where not all rows have the same columns.

As can be seen in the diagram above while these four sets of data are in the same column family, they do not all contain the same four columns.

2.3 Document Stores

A document store database can be defined as any NoSQL database that has the following properties: [i] (p. 208-209) “

- It is a key-value store.
- The data objects stored as values for keys are called documents; the keys are used for identification.
- The documents contain data structures in the form of recursively nested attribute-value pairs without referential integrity.
- These data structures are schema-free, i.e., arbitrary attributes can be used in every document without defining a schema first.”

The most popular document store database[ii] at the time of this writing is ‘MongoDB’[9].
A common misconception with regards to document stores is that they are indifferent to key-value store databases. Although it is possible to store the same data that would be stored in a document in text form and use a simple key-value database, however this would mean that the structures would need to be processed on the client side, impeding efficiency and speed.

[9] https://www.mongodb.com/

Furthermore, this would neglect key features that are made available by document store DBMSs such as the ones listed on the MongoDB website:[10] “

- Document model: Data is stored in documents (unlike other databases that store data in structures like tables or graphs). Documents map to objects in most popular programming languages, which allows developers to rapidly develop their applications.
- Flexible schema: Document databases have a flexible schema, meaning that not all documents in a collection need to have the same fields. Note that some document databases support schema validation, so the schema can be optionally locked down.
- Distributed and resilient: Document databases are distributed, which allows for horizontal scaling (typically cheaper than vertical scaling) and data distribution. Document databases provide resiliency through replication.
- Querying through an API or query language: Document databases have an API or query language that allows developers to execute the CRUD operations on the database. Developers have the ability to query for documents based on unique identifiers or field values.”

2.4 Graph Databases

Graph Databases can be defined as NoSQL databases that meet the following requirements: [i] (p. 215-216) “

- The data and/or the schema are shown as graphs … or graph-like structures, which generalize the concept of graphs (e.g., hypergraphs)
- Data manipulations are expressed as graph transformations, or operations which directly address typical properties of graphs (e.g., paths, adjacency, subgraphs, connections, etc.).
- The database supports the checking of integrity constraints to ensure data consistency. The definition of consistency is directly related to graph structures (e.g., node and edge types, attribute domains, and referential integrity of the edges).”

The current most popular graph DBMS[ii] is ‘Neo4j’[11].

[10] “Document Database - Nosql,” MongoDB, accessed February 28, 2023, https://www.mongodb.com/document-databases.
[11] https://neo4j.com/

Graph Databases do quite literally what the name implies, they store data in graph formats using nodes to represent and store data, and edges to represent relationships. This is outlined in the example below.

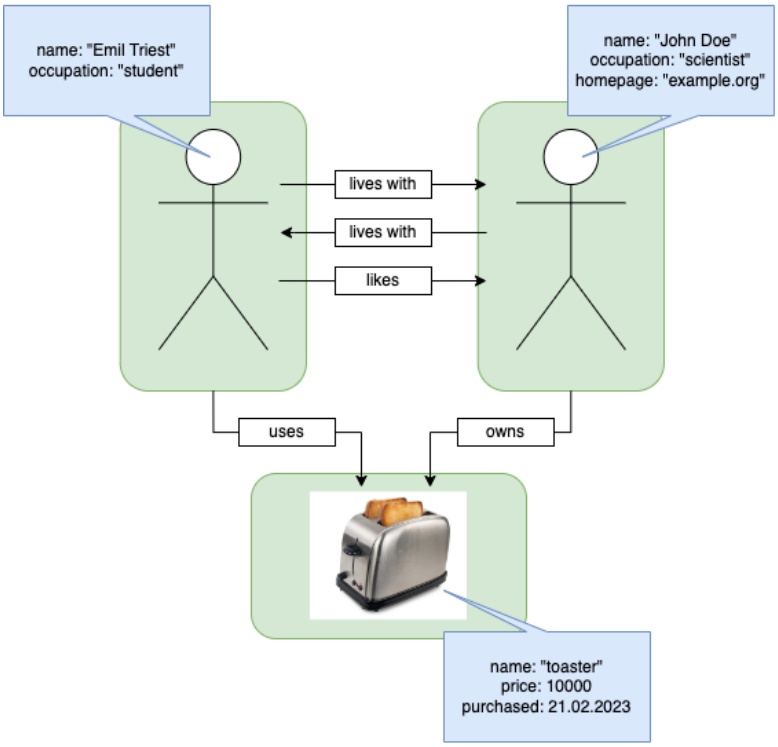

Figure 2 - Illustration showing a relationship between two people and a toaster using a graph model.

The example above uses a graph structure to represent the relationship between a student, a scientist and a toaster. The green boxes represent nodes, the blue boxes represent data stored in said nodes and the white boxes represent edges or relationships between these nodes. It can be seen that the student and scientist both live with each other, however only the student likes the scientist. This is probably due to the student using the expensive toaster the scientist recently purchased. As roughly demonstrated by this example relationship, graphs are quite handy for illustrating and storing complicated relationships.

3. Cost/Benefit Analysis

Since it is difficult to define a price for a type of data model, the following approach was taken: the pricing for the most popular framework was analyzed for each type of NoSQL model. The following variables were attempted to control as best as possible:

- Type of tier (professional tier if available)
- Type of billing (monthly if available)
- Amount of storage (1 GB if available)

3.1 Costs of different NoSQL databases

3.1.1 Key-value

Figure 3: [iii]

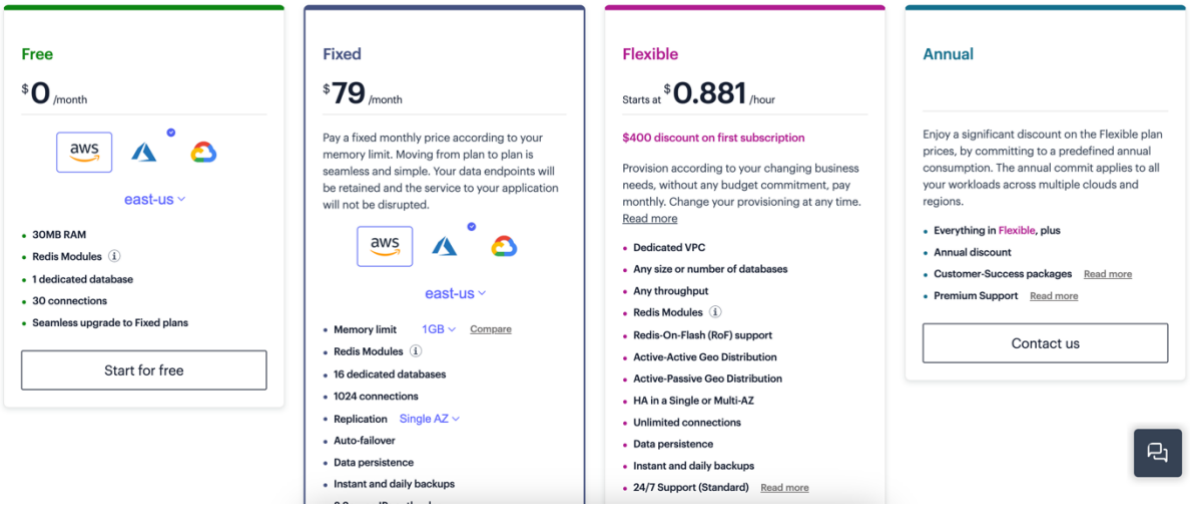

Figure 3 - Pricing models shown on the redis website

The image above shows the different pricing models that are offered on the website of the most popular key-value DBMS; redis. As can be seen, a free tier, a fixed tier, a flexible tier and an annual tier is offered. The price that will be focused on for the purpose of this paper is the monthly fixed price for one GB of storage: $ 79. While there are other options available in the fixed tier, which are shown in Figure 4, for the sake of uniformity and integrity of the comparison the option of 1 GB has been chosen for further analysis.

Figure 4: [iii]

Memory Limit	Dedicated Databases	Connections	Source IP auth. rules	Price
100MB	4	256	4	$8/mo
250MB	8	256	4	$20/mo
500MB	12	512	4	$40/mo
1GB	16	1024	8	$79/mo
2.5GB	24	2500	8	$192/mo
5GB	32	5000	16	$375/mo
10GB	64	10000	32	$732/mo

Figure 4 - Image of the redis fixed tier pricing table

As can be seen above the prices vary according features such as memory limit, dedicated databases, connections and source IP authentication rules. Certainly, a different price per GB can be used for larger or smaller memory limits, however for the purpose of this study the 1 GB memory limit will be used for further comparison and analysis. This means that for the most popular key-value NoSQL database, we can expect a price of roughly $ 79 per GB per month.

3.1.2 Column-family

Figure 5: [iv]

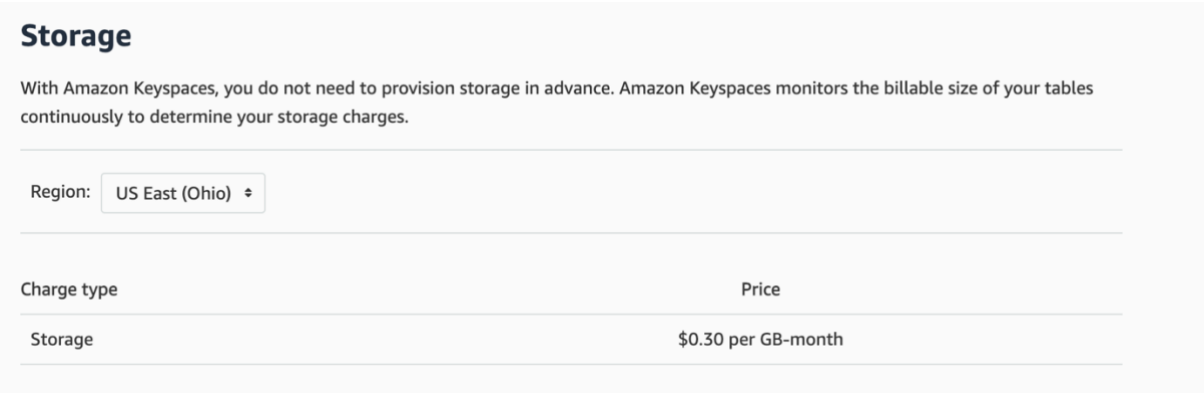

Figure 5 - Image showing the per GB-month price of a aws hosted cassandra database

The image above shows the pricing of the most popular column-family database, Cassandra, hosted on amazon web services (AWS). At $ 0.30 per GB per month, Cassandra is exceedingly less expensive than redis.

3.1.3 Document

Figure 6: [v]

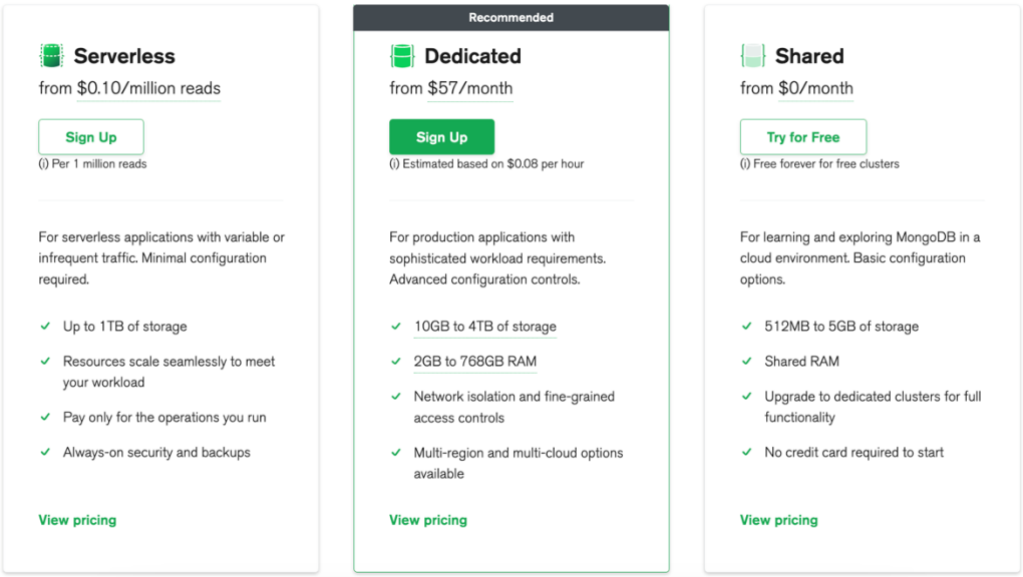

Figure 6 - Image showing the different pricing tiers for MongoDB databases

As can be seen from the image above, the most popular document DB, mongoDB, offers three different tiers: Serverless, Dedicated and Shared. For the purpose of this comparison the dedicated tier will be used to calculate a $/GB/Month value. The dedicated tier starts at $57 per month for 10 GB, meaning that the dollar per GB per Month value would be: 57/10= $ 5.70.

3.1.4 Graph

Figure 7: [vi]

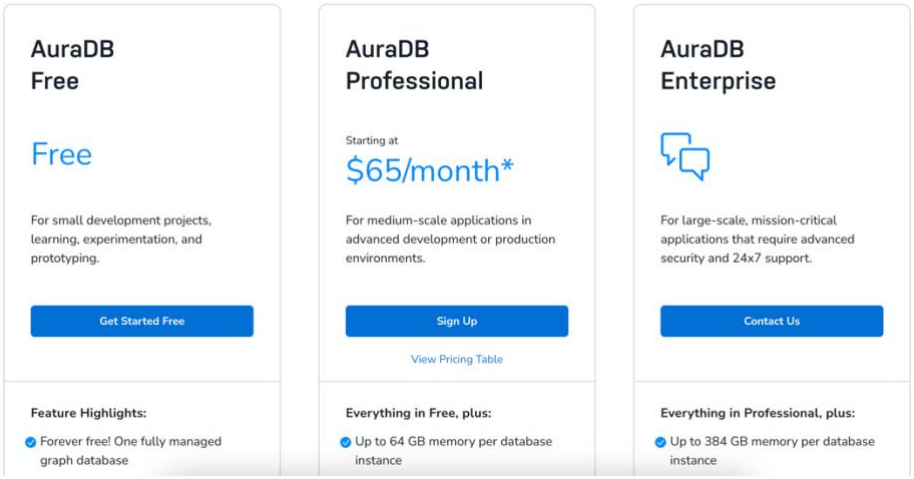

Figure 7 - Pricing models available for a neo4j database

As can be seen in the image above, the most popular graph database, neo4j, offers three different types of tiers: free, professional and enterprise. The professional tier starts at $ 65 per month and comes with 2 GB of storage, as can be seen in Figure 8.

This results in a price of $ 32.40 per GB per month ($64.80/2 GB).

Figure 8: [vi]

Memory	CPU	Storage	Hourly	Monthly* (30 days)
1GB	1CPU	2GB	$0.09/hour	$64.80/month
2GB	1CPU	4GB	$0.18/hour	$129.60/month
4GB	1CPU	8GB	$0.36/hour	$259.20/month
8GB	2CPU	16GB	$0.72/hour	$518.40/month
16GB	3CPU	32GB	$1.44/hour	$1036.80/month
24GB	5CPU	48GB	$2.16/hour	$1555.20/month
32GB	6CPU	64GB	$2.88/hour	$2073.60/month
48GB	10CPU	96GB	$4.32/hour	$3110.40/month
64GB	12CPU	128GB	$5.76/hour	$4147.02/month

For sizing larger than 64GB contact us for Enterprise pricing

Close ×

Figure 8 - Image of the professional tier pricing tier from the neo4j webite

3.2 Overview of Cost comparison

DBMS	Redis	Cassandra	MongoDB	Neo4j
Price ($/GB/Month)	79.00	0.30	5.70	32.40

Table 3 - Table showing the prices of the four most popular DBMS for each database model

From a cost perspective Cassandra (column-family) is the clear winner. Although all the alternatives offer free tiers, these free tiers usually include restrictions that make them unviable as an option for companies or larger applications. It should also be noted that all of these databases are free to use on local machines. Neo4j and Redis are significantly more expensive. For redis, this can be explained by the fact that the data is stored 'in-memory', and RAM storage tends to be more expensive than SSD. Basically, Redis trades speed for storage capacity and price. Not all key-value databases store their data 'in-memory'. Neo4j's higher price tag could be justified by their extensive customer support, however a clear reason was difficult to find.

3.3 Benefits/tradeoffs of different NoSQL databases

3.3.1 Key-Value

Benefits of key-value databases include [vii] high performance, simplicity of integration, scalability as they are often horizontally scalable and flexibility with respect to data types. Tradeoffs and limitations of key-value databases include [vii] limited querying capabilities, no support for more complex data structures and or relationships, and the consistency and integrity of the data is not always guaranteed.

3.3.2 Column-Family

Benefits of column-family databases, also known as wide-column databases, also include [viii] high performance, especially with regards to queries, a flexible and efficient data model, they are also often horizontally scalable and strong at supporting high write throughput. Tradeoffs and limitations of column-family databases are very similar to those of key-value databases, they include [viii] that their querying capabilities are limited, their data modelling can prove to be problematic for complex data structures and or relationships, data consistency and

integrity is difficult to guarantee and lastly data migration from column-family databases to other models can be difficult and time consuming.

3.3.3 Document

Benefits of document databases include [ix] a flexible schema which allows for easier adaptation to changing requirements and handing unstructured or semi-structured data, high performance for both read and write operations, horizontal scalability, they are often easy to use and integrate for developers due to their intuitiveness, and lastly their nested data structures make it easier to store and query complex data. Tradeoffs and limitations of document databases include [ix] data consistency and integrity, more complex backup and recovery processes.

3.3.4 Graph

Benefits of graph databases include [x] that the query speed is only dependent on the number of relationships of a node, and not on the amount of data, a clear and manageable representation of relationships is possible, and the graph model allows for flexible and agile data structures. However, one of the most prominent drawbacks [x] of graph databases is the complexity and cost intensity of scaling them, due to their one-tier architecture design. Furthermore, their importance and use are heavily dependent on the use case, specifically whether the data has many complex relationships that would be difficult to model in different NoSQL databases.

4. Analysis of database requirements for multimedia applications

A fantastic and detailed journal issue[xi] of the IEEE 'MultiMedia' journal[12] examines and analyses the requirements and issues regarding multimedia database management. The requirements, as described by the above-mentioned journal, were defined as the following:[xi] "

- Traditional DBMS capabilities
- Huge capacity storage management
- Information retrieval capabilities
- Media integration, composition, and presentation
- Multimedia query support
- Multimedia interface and interactivity
- Performance"

The points above are examined in more detail in the paper, however for the sake of this seminar paper these points will define the requirements of multimedia applications to their DBMSs.

[12] Accessed March 1, 2023 https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=93

5. Scenario-based Discussion of NoSQL Databases for Multimedia Applications

In order to discuss NoSQL databases for Multimedia Applications, the following three scenarios were defined:

- Small userbase, < 100,000 monthly active users
- Medium userbase, > 1 million monthly active users
- Large userbase, > 100 million monthly active users

To put these numbers into perspective; Facebook, the most popular social network as of January 2023, had 2,958 million monthly active users[13].
All scenario discussions, except for scenario 4, assume that the application is not yet productive, but rather what the end goal for their userbase looks like. The goal is to discuss what should be taken into account for the choice of their data model in the planning stage.

5.1 Scenario 1: Multimedia Application with small userbase

A multimedia application that will operate with a smaller userbase has the luxury of being able to prioritize scaling less and focus more on which data model suits their use-case. A small social media platform for instance may favor a graph model, since scalability isn't the main facto, and graphs are great at representing complex relationships such as those found in social media platforms. An ecommerce business might prefer a column-family model since the complexity of the queries tend to be simple and further scalability remains an option. A key-value database could be an excellent option for a gaming platform, due to its speed.

The choice heavily depends on the use case. With that being said due to the size of the userbase and due to the low cost associated with column-family databases, they are likely to be a better choice for multimedia applications with smaller userbases.

[13] Published by S. Dixon. "Biggest Social Media Platforms 2023,". Accessed March 1, 2023. Statista, February 14, 2023. https://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/

5.2 Scenario 2: Multimedia Application with midsize userbase

A larger userbase tends to accumulate more data, more data means for the choice of the data model the scalability needs to be weighted stronger. This tends to exclude graph models. It should however be noted that for specific use-cases, the implementation of graph models can still be justified. Document databases are a great choice both with respect to data modelling possibilities and price. Key-value database are probably going to be more expensive, depending on the importance of speed a midsize userbase application might want to prioritize speed over cost. Column-family stores are going to be less expensive, integration and maintenance costs for developers may be higher due to more complex and less intuitive data modelling possibilities.

5.3 Scenario 3: Multimedia Application with large userbase

For multimedia applications with large userbases, the number one priority is scalability. If the model doesn't scale, content will take longer to load or won't load at all and the application loses its lifeblood; its users. Again, due to the specific use-case importance for each data model, a multimedia application with this size of userbase will probably benefit the most from a combination of the database models specific to each feature. Facebook for example might use key-value databases for caching user specific data (fast & low latency), column-family databases to analyze traffic (more query potential and speed), a document store for search features (many filtering and data modelling options) and a graph store for recommendations and ads (clear representation of relationships).

5.4 Scenario 4: Migrating a Multimedia Application from a RDBMS to a NoSQL database

For applications that already integrate a RDBMS and want to switch to a NoSQL model, the choice can be difficult. For this use-case column-family databases tend to be a bad choice. Depending on the amount of data and the impact of per GB cost either document or key-value stores tend to be smarter choices due to ease of migration, existing documentation and migration solutions.

6. Conclusions

Big data and multimedia have had a significant impact on how data is stored. Nonrelational database models, or NoSQL databases, are often summarized as one category sparing the details of core NoSQL data models and concepts. Details that deserve attention, because all NoSQL database models are not the same, quite the opposite. Each core NoSQL database model comes with a different set of benefits and tradeoffs. This makes each model more, or less, suitable for different scenarios and different types of applications or use-cases.

At the beginning of this research paper, the following research objective was defined:

to find out which type of NoSQL database model is best suited for different types of multimedia applications and different types of application scenarios.

While it is difficult to define a 'winner-take-all'[14] NoSQL database model, the information gathered within the scope of pursuing the research objective above have allowed for a deeper insight into the four core NoSQL database models, their strengths, their weaknesses, their costs and their applications. It may be easy to say that MongoDB is the most popular NoSQL database, and it may be easier to simply appoint document stores as the winner of NoSQL database models, however this would be cheating the research objective that was defined. The easy conclusion doesn't encompass the speed that key-value databases provide, it doesn't encompass the relationship representation that graph databases provide, and it doesn't reflect the querying power and cost efficiency of column-family databases.

Every single one of these four models still see widespread use today, and for a good reason. There are millions of different software use-cases and application scenarios that have immensely varying requirements. One solution cannot solve all of them. However, when planning to integrate a DBMS into a multimedia application, the information gathered above can help to decide which database model is the best fit for the given application scenario.

[14] Accessed 3 March, 2023. https://dictionary.cambridge.org/de/worterbuch/englisch/winner-take-all

Declaration in lieu of Oath

I hereby guarantee that the work presented here is my own work and that it has been produced without unauthorised assistance, and in particular that I have indicated as such all sections which have been taken from publications verbatim or nearly verbatim. I also guarantee that the written version I have submitted is the same as the digital version. I furthermore declare that the work has not previously been submitted to any examination authority/body in the same or a similar form. I grant my consent for the work to be made accessible to the public. I declare my consent for the digital version of this work to be uploaded to the server of an external service provider for the purposes of a plagiarism check. The plagiarism check shall in no way act as a form of disclosure of this work to the public.

Berlin, 02.03.2023

Signed, Emil Triest

References

- C. Strauch U.-L. S. Sites and W. Kriha. “Nosql databases,”. Lecture Notes. Stuttgart Media University. 2011.
- I. Robinson, J. Webber, and E. Eifrem. “Graph databases.”. O’Reilly Media, Inc.. 2013.
- Edlich, S., Friedland, A., Hampe, J., Brauer B., Brückner M. “NoSQL—Einstieg in die Welt nichtrelationaler Web 2.0 Datenbanken”. Hanser, München. 2011.
- McCreary, D., Kelly, A.: “Making Sense of NoSQL - A Guide for Managers and the Rest of Us”. Manning, Shelter Island. 2014.

Endnotes

[i] Meier. *SQL & NoSQL Databases*. Springer Fachmedien Wiesbaden, 2019. https://doi.org/10.1007/978-3-658-24549-8_7

[ii] “Engines Ranking.” DB. Accessed February 28, 2023. https://db-engines.com/en/ranking/

[iii] Accessed March 1, 2023. https://redis.com/redis-enterprise-cloud/pricing/

[iv] Accessed March 1, 2023. https://aws.amazon.com/keyspaces/pricing/?nc1=h_ls

[v] Accessed March 1, 2023. https://www.mongodb.com/pricing

[vi] Accessed March 1, 2023. https://neo4j.com/pricing/

[vii] “Key-Value Database (Use Cases, List, Pros & Cons).” Accessed March 1, 2023. DatabaseTown, January 30, 2023. https://databasetown.com/key-value-database-use-cases/

[viii] “Wide Column Database (Use Cases, Example, Advantages & Disadvantages).” Accessed March 1, 2023. DatabaseTown, January 29, 2023. https://databasetown.com/wide-column-database-use-cases/

[ix] “What Is Document Database? (Document Oriented Database) Uses Cases, Operations, Model.” Accessed March 1, 2023. DatabaseTown, January 26, 2023. https://databasetown.com/what-is-document-database/

[x] “Graph Databases Explained.” IONOS Digital Guide. Accessed March 1, 2023. http://ionos.com/digitalguide/hosting/technical-matters/graph-database/

[xi] Adjeroh, D.A., and K.C. Nwosu. “Multimedia Database Management-Requirements and Issues.” *IEEE Multimedia* 4, no. 3 (1997): 24–33. https://doi.org/10.1109/93.621580