Applied research document (SQL or NoSQL)

Problem statement

I am a junior software developer and I am currently working on a web solution that is composed of a ReactJS single-page application and a Spring boot REST API. The app is connected to a MySQL database but I am not happy with how things are turning out and I want to switch to a different style of database

Main question

Is a relation or non-relational database most suitable for small, medium and large scale applications

Sub questions

What exactly is a relational database?

What exactly is a non-relational database?

How do the 2 types of databases resolve transactions?

What are the most common applications for these databases?

Which type of database is most widely used?

Which database has a faster performance?

Which database is more secure?

What are the scalability options for these databases?
Which type of database has a larger support/community

Why not use both types of databases in one application?

What exactly is a relational database?

Source 1: Google cloud

On google cloud's website, a relational database is defined as the following:

"Developed by E.F. Codd from IBM in the 1970s, the relational database model allows any table to be related to another table using a common attribute. Instead of using hierarchical structures to organize data, Codd proposed a shift to using a data model where data is stored, accessed, and related in tables without reorganizing the tables that contain them.

A relational database (RDB) is a way of structuring information in tables, rows, and columns. An RDB has the ability to establish links—or relationships–between information by joining tables, which makes it easy to understand and gain insights about the relationship between various data points.

Attributes (columns) specify a data type, and each record (or row) contains the value of that specific data type. All tables in a relational database have an attribute known as the **primary key**, which is a unique identifier of a row, and each row can be used to create a relationship between different tables using a **foreign key** — a reference to a primary key of another existing table." *(What Is a Relational Database (RDBMS)? |, n.d.-c)*

Source 2: Oracle's database documentation

Oracle is the biggest relational database provider today. In their documentation, the following is stated:

"In his seminal 1970 paper "A Relational Model of Data for Large Shared Data Banks," E. F. Codd defined a relational model based on mathematical set theory. Today, the most widely accepted database model is the relational model.

A **relational database** is a database that conforms to the relational model. The relational model has the following major aspects:

- Structures

  Well-defined objects store or access the data of a database.
- Operations

  Clearly defined actions enable applications to manipulate the data and structures of a database.
- Integrity rules

  Integrity rules govern operations on the data and structures of a database.

A relational database stores data in a set of simple relations. A **relation** is a set of tuples. A **tuple** is an unordered set of attribute values.

A **table** is a two-dimensional representation of a relation in the form of rows (tuples) and columns (attributes). Each row in a table has the same set of columns. A relational database is a database that stores data in relations (tables). For example, a relational database could store information about company employees in an employee table, a department table, and a salary table." *(Introduction to Oracle Database, n.d.-b)*

What exactly is a non-relational database?

Source 1: Microsoft's guide to non-relational data

According to a guide on microsoft's website:

"A **non-relational database** is a database that does not use the tabular schema of rows and columns found in most traditional database systems. Instead, non-relational databases use a storage model that is optimized for the specific requirements of the type of data being stored. For example, data may be stored as simple key/value pairs, as JSON documents, or as a graph consisting of edges and vertices.

What all of these data stores have in common is that they don't use a relational model. Also, they tend to be more specific in the type of data they support and how data can be queried. The term **NoSQL** refers to data stores that do not use SQL for queries. Instead, the data stores use other programming languages and constructs to query the data. In practice, "NoSQL" means "non-relational database," even though many of these databases do support SQL-compatible queries. " (*EdPrice-MSFT, 7.25.2022)*

Source 2: MongoDB's documentation

MongoDB is one of the most popular non-relational database providers out there. In their documentation, the following can be read:

"Most databases can be categorized as either relational or non-relational. Non-relational databases are sometimes referred to as "NoSQL," which stands for Not Only SQL. The main difference between these is how they store their information.

A non-relational database stores data in a non-tabular form, and tends to be more flexible than the traditional, SQL-based, relational database structures. It does not follow the relational model provided by traditional relational database management systems. Non-relational databases might be based on data structures like documents. A document can be highly detailed while containing a range of different types of information in different formats." *(MongoDB, n.d.-b)*

Which database resolves transactions in a better way?

Source 1: SQL vs NoSQL: Differences, Databases, and Decisions

Talend is an organization that helping companies develop better data agility, and data trust, and data culture. In an article on their website, it is explained how the ACID and CAP properties work:

"At a high level, SQL and NoSQL comply with separate rules for resolving transactions. RDBMSs must exhibit four "ACID" properties:

- **Atomicity** means all transactions must succeed or fail completely. They cannot be partially-complete, even in the case of system failure.
- **Consistency** means that at each step the database follows invariants: rules which validate and prevent corruption.
- **Isolation** prevents concurrent transactions from affecting each other. Transactions must result in the same final state as if they were run sequentially, even if they were run in parallel.
- **Durability** makes transactions final. Even system failure cannot roll-back the effects of a successful transaction.

NoSQL technologies adhere to the "CAP" theorem, which says that in any distributed database, only two of the following properties can be guaranteed at once:

- **Consistency:** Every request receives the most recent result, or an error. (Note this is different than in ACID)
- **Availability:** Every request has a non-error result, regardless of how recent that result is.

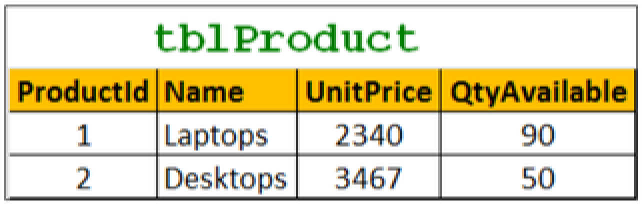
- **Partition tolerance:** Any delays or losses between nodes will not interrupt the system's operation." *(Talend, n.d.-b)*

Source 2: Testing scenarios

Even though these properties are very clearly documented, I decided to test them for myself.

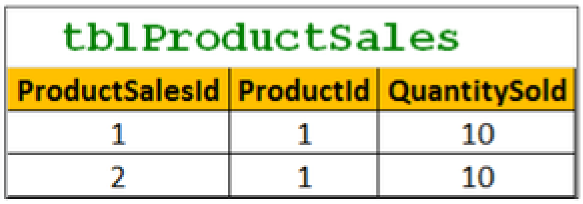
The ACID properties test

The scenario is as follows: certain queries will be executed in order to test the ACID properties of a relational database. The table structure is as follows:



tblProduct

ProductId	Name	UnitPrice	QtyAvailable
1	Laptops	2340	90
2	Desktops	3467	50



tblProductSales

ProductSalesId	ProductId	QuantitySold
1	1	10
2	1	10

Atomic test:

I executed the following query

```
Begin
  Begin Try
    Begin Transaction
      Update tblProduct set QtyAvailable = (QtyAvailable - 10)
      where ProductId = 500

      Insert into tblProductSales values(3, 1, 10)
    Commit Transaction
  End Try
  Begin Catch
    Rollback Transaction
  End Catch
End
```

The result of the test is as follows: a product with id does not exist and an error occurs, scrapping the whole transaction. An entry in the sales table is **NOT** created, thus proving the **Atomic** part of **A**CID

Consistent test

I executed the following query:

```
Begin
  Begin Try
    Begin Transaction
      Update tblProduct set QtyAvailable = (QtyAvailable - 10)
      where ProductId = 1

      Insert into tblProductSales values(3, 1, 10)
    Commit Transaction
  End Try
  Begin Catch
    Rollback Transaction
  End Catch
End
```

Here everything goes well, 10 units are deducted from the product with id=1 and a new entry is created in the sales table, thus proving the **Consis tent** part of A**C**ID

Isolation test

The isolation test is done by just executing two of the previous queries at the same time. Everything goes well and there are 20 units deducted from the product with id=1, and two new entries in the sales table. This is done because the database utilizes locking to maintain transaction **isola tion** in order to make sure that they don't interfere with each other.

Durability test

In the midst of the following query being executed I closed down MySQL workbench and therefore interrupted the transaction half-way

```
Begin
  Begin Try
    Begin Transaction
      Update tblProduct set QtyAvailable = (QtyAvailable - 10)
      where ProductId = 1

      Insert into tblProductSales values(3, 1, 10)
    Commit Transaction
  End Try
  Begin Catch
    Rollback Transaction
  End Catch
End
```

The result: nothing changed, therefore proving the **Durability** part of ACI**D**

Answer: Relational and non-relational databases have different properties that they abide to. Relational databases handle transactions in a more consistent way, which is better if consistency is key. Non-relational databases can follow either consistency or availability, but not both. Most often the choice is availability, which makes them more suitable for systems that experience a lot of traffic.

What are the most common applications for these databases?

Source 1: SQL vs NoSQL: Differences, Databases, and Decisions

Continuing to read talend's article, the following is stated:

"Generally, NoSQL is preferred for:

- Graph or hierarchical data
- Data sets which are both large and mutate significantly,
- Businesses growing extremely fast but lacking data schemata.

In terms of use cases, this might translate to **social networks, online content management, streaming analytics, or mobile applications.**

SQL is more appropriate when the data is:

- Small
- Conceptually modeled as tabular
- In systems where consistency is critical.

Think small business' **accounting systems, sales databases, or transactional systems like payment processing in e-commerce**. When in doubt, SQL is also more appropriate, as RDBMSs are better supported and fault-tolerant." *(Talend, n.d.-b)*

Source 2: Survey of database applications

A survey on careerkarma.com shows an overview of which companies use SQL. The table below lets us see that very big and familiar companies such as Adobe, Dell, Microsoft, Facebook and LinkedIn use it mainly for **Data analysis and data management**

Companies That Use SQL	Who Uses SQL at This Company?	What Does This Company Use SQL For?	Estimated Number of Employees
Accenture	Microsoft SQL Server Database Administration, Application Developer PL/SQL, Production Support Engineer	Data analysis, data management	624,000
Adobe Systems, Inc.	Software Development Engineer, Finance Manager, Software Engineer Full Stack	Data analysis, data management, marketing	22,516
Cigna	Software Development Engineer, Data Management Analyst, Senior Java Services Engineer	Data analysis, data management	74,000
Cognizant	SAP BO Developer, Oracle PL/SQL Developer, Data Engineer with Oracle PL /SQL and Unix Scripting	Data analysis, data management	289,500
Dell Technologies	SQL Server and Oracle Technical Marketing Engineer Integrated Solutions, Senior Software Reliability Engineer, Data Engineer	Data analysis, data management, marketing	158,000
Facebook	Staffing Optimization Specialist, Data Analyst Global Operations Central Analytics, Research Data Scientist	Data analysis, data management, marketing	58,604

LinkedIn	Staff Knowledge Engineer, Strategic Financial Analyst (Revenue), Sales Strategy and Operations Analyst	Data analysis, data management	16,000
Microsoft	Data Engineer, Senior Program Manager, Principal Data Architect	Data analysis, data management	103,000
NTT Data	Senior Data Analyst, Senior .Net Developer, Senior Data Scientist	Data analysis, data management	139,680
Seagate	Staff Analyst-Native Hana, Senior Engineer Characterization and Analytics Group, Senior Data Scientist	Data analysis, data management	40,000

(MAIA ESTRERA **- AUGUST 05, 2022)**

According to a blog on bangdb.com, popular applications that take advantage of NoSQL are:

- Uber
- Cisco
- Netflix
- Forbes
- Facebook messenger
- Google mail
- Linked in

Most of these applications share things in common, such as being **mobile apps, discussion threads, social media, which need to scale rapidly and process large amounts of data.**

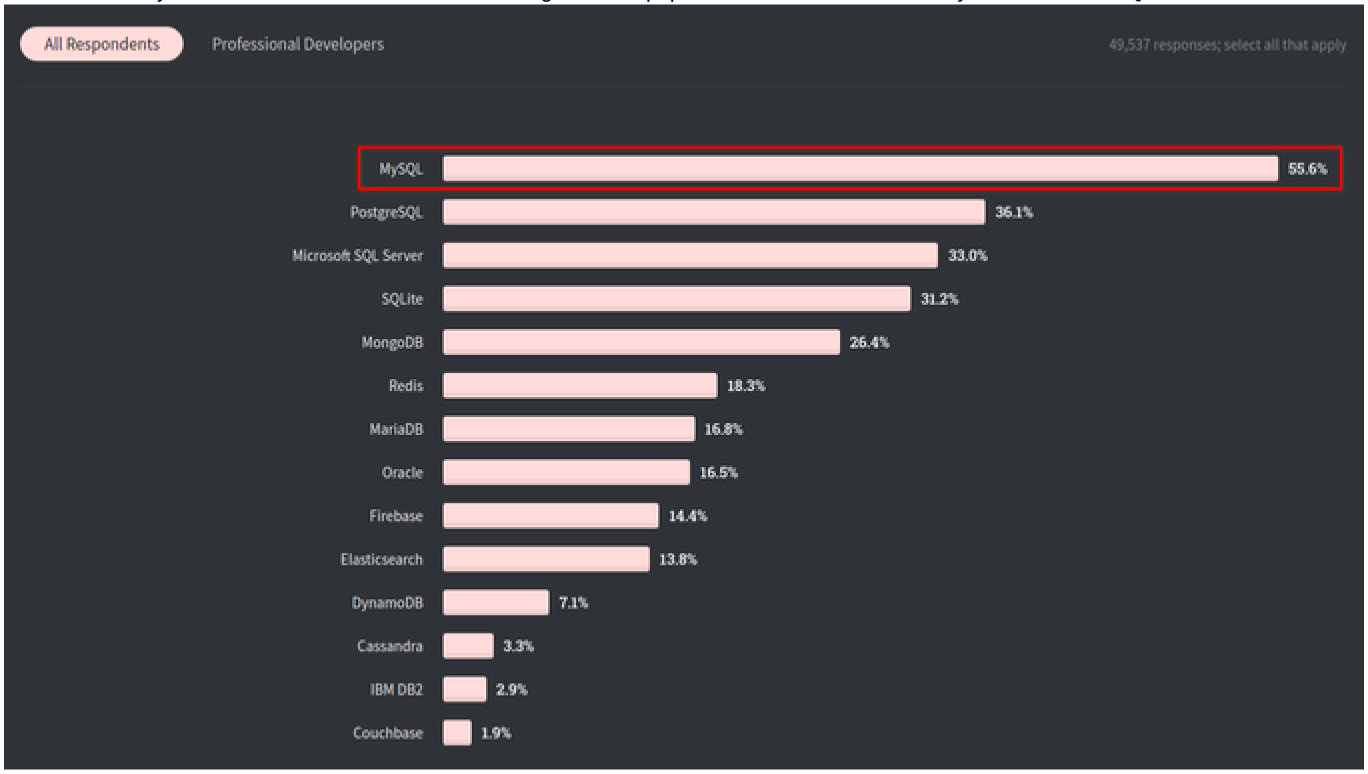
***Answer*:** SQL is most widely used In systems where consistency is critical, such as data analytics, sales databases, or transactional systems

NoSQL is used for applications that need to store large amounts of data and grow extremely fast

Which type of database is most widely used?

Source 1: Stack overflow survey

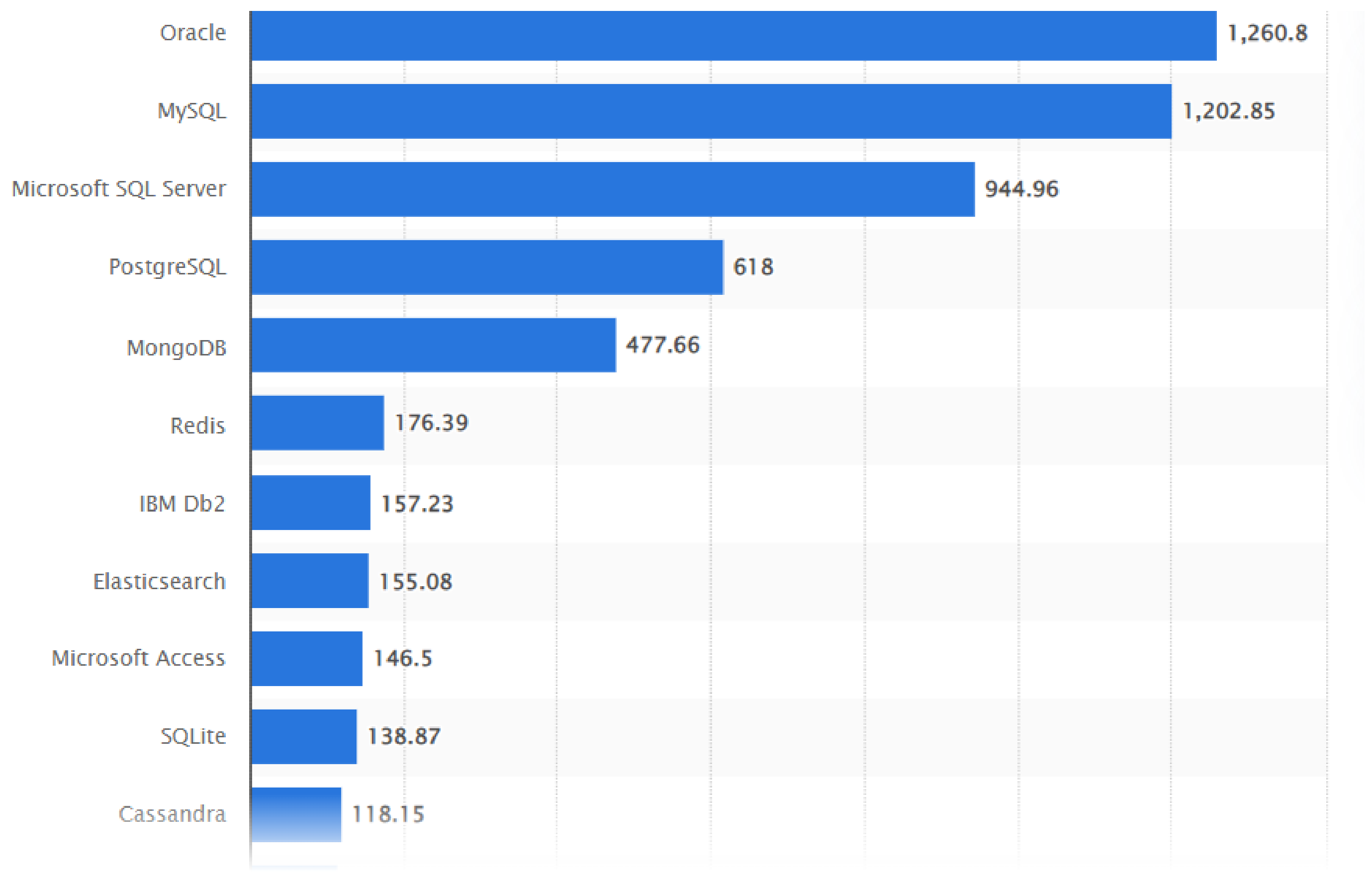
Conducted surveys on stack overflow show that the among the most popular databases are RDBMS systems that use SQL



The most popular NoSQL database is MongoDB ranking 5th on the survey.

Source 2: Survey on Statista

Conducted surveys on Statista show the ranking of the most popular database management systems worldwide, as of August 2022



In the top 4 again are relational databases such belonging to Oracle, MySQL, Microsoft and PostgreS

Again, the first non-relational database is MongoDB, scoring 5th place

Answer: Relational databases are a lot more popular that non-relational ones

Which database has a faster performance?

Source 1: Teacher interview

When asked about this topic, Marcel Boelaars, a teacher in Fontys who has used who has used both types of databases during his work experience synthesized:

A relational database is going to have a better write speed than a non-relational because of the way that it stores information and indexes it. SQL databases need to write the data in only one place, where as NoSQL have to write it to multiple nodes.

He also said that NoSQL databases are specifically designed for unstructured data which can be document-oriented, column-oriented, graph-based, etc. In this case, the NoSQL database would have a faster read because it can talk to any of the nodes to retrieve information. If I, however, was looking to do complex joins and statistics, that would be hard and slow in a non-relational database.

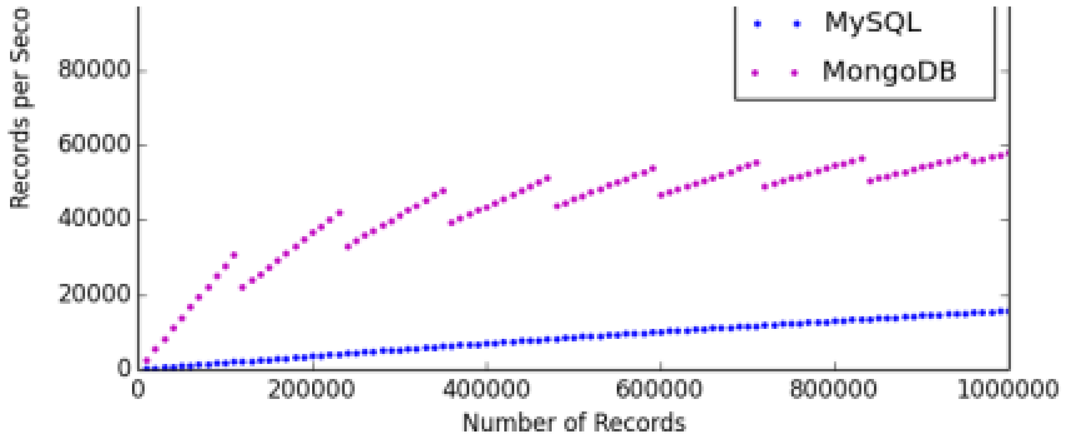
Source 2: Performing tests

To evaluate the performance of relational databases and non-relational databases, I conducted some tests using a benchmark tool from stssoft to test the read and write speeds among relational database (MySQL) and document storage (MongoDB). The test platform configuration is:

- OS: Microsoft Windows 11 64 bit
- CPU: Intel (R) Core (TM) i7-8750H CPU @ 2.20GHz
- RAM: DDR4 16GB 2667MHz
- Storage: SAMSUNG MZ7PC128HAFU-000H1
- Database Settings: InsertsPerQuery: 5000;
- Indexing Technology: BTree;
- Test Data: 1,000,000 Records, 100% randomness

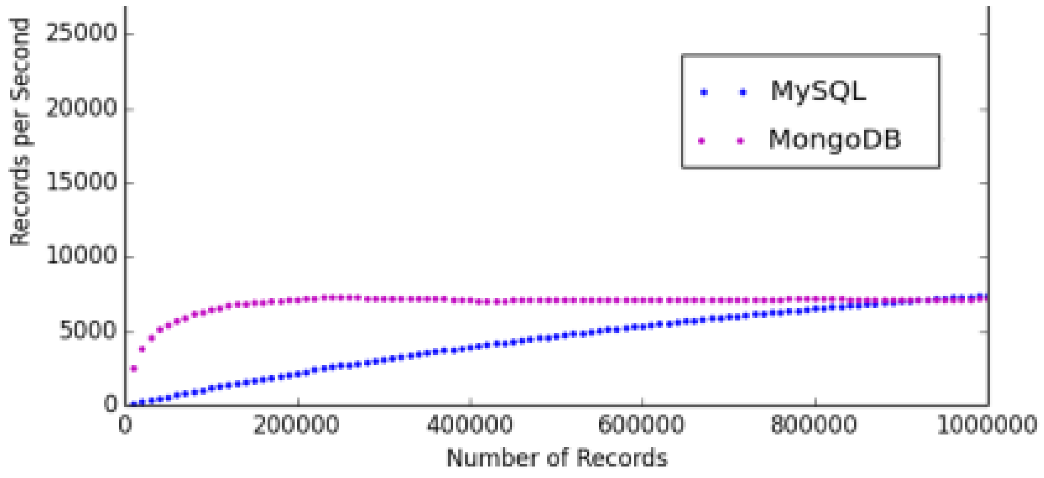
The information will be saved normally in the MySQL database, while MongoDB uses the JSON document format as its storage

Fig. 1 - Read speed benchmark



We can see in Fig. 4 that in a scenario of just reading a big amount of records without doing any complex queries the non-relational database performs much faster, although the relational one is still not too slow and would be fine to use in smaller use-case applications

Fig. 2 - Write speed



We can see on fig. 5 that the non-relational database is ahead of the relational one when it comes to a small amount of records and it takes some time for MySQL to catch up to its write speed when writing simple records.

Answer: NoSQL is generally faster than SQL, especially for large amounts of data, as seen by the conducted benchmark

Which database is more secure?

Source 1: http://softwaretestinghelp.com

In an article on softwaretestinghelp.com, the following can be read:

“Most of the enterprise-based relational or SQL databases like Oracle and MySQL have strong security features integrated into them. They abide by the **ACID** properties which ensure secure and reliable database transactions. RDBMS also has features like role-based security, access-control via user-level permissions, encrypted messages, support for row and column access control, etc. However, these security features do need a significant licensing fee and affect the speed of data access.

NoSQL databases security is not as robust as relational databases security. NoSQL does not strictly follow ACID properties. The one in NoSQL is known as the BASE (Basically available, soft state, eventually consistent) properties. Instead of being consistent after every transaction, it is okay here for the database to be in a consistent state eventually. It may not be the case that you will always see the current data in NoSQL databases. You may be seeing the data as per last taken snapshot and a simultaneous transaction can interfere with each other.

Unlike SQL databases, the NoSQL databases have very few inbuilt security features in order to allow faster data access. They lack confidentiality and integrity attributes. Also, as they don’t have a fixed and well-defined schema, you can’t segregate the permissions.” *(softwaretestinghelp, 29.10.2022)*

Source 2: Answer to sub-question - **What are the most common applications for these databases?**

Further analyzing the answer to one of the previous sub-questions gives us more information about the security of the 2 types of databases. Relational databases are crucial in scenarios where security is needed such as applications that deal with online payments, transactions, and data analytics. Not a single successful software solution in the field can be found that uses a non-relational database for such confidential information.

On the other hand, non-relational databases are used in applications where there is less consistency and generally the data is not sensitive, such as social media and mobile apps.

We can therefore come to the conclusion that a relational database is far more secure than a non-relational one

Answer: Relational databases are more secure than a non-relational ones

What are the scalability options for these databases?

Source 1 - linkedin.com

In an article on linkedin.com, a lead back-end engineer says:

“**RDMS** is designed to scale vertically. When a database begins to experience performance issues, the common solution is to migrate it to a larger hard drive or faster server. In the modern era of **Big Data**, the rapid growth of the database often exceeds the speed at which such a painstaking migration process can occur. RDBMS databases were designed in a period when data could be kept small, neat, and orderly. That assumption no longer holds anymore. Achieving scalability and elasticity is a huge challenge for relational databases. Scalability with **RDBMS** co mes with hardware upgrades that are incredibly expensive.

With non-relational databases, however, one can scale horizontally, which is overall a lot easier. NoSQL scale horizontally, distributed across multiple hosts rather than a single server to avoid a single point of failure. The decentralized nature of NoSQL also makes it a far better option for organizations that need or want to migrate their database to the cloud. Instead of upgrading costly hardware, the difference with this database is that you can expand cheaply by simply adding cloud instances or commodity servers.” *(Oluwaseyi Otun, 10.01.2020)*

Source 2 - Teacher interview

When asked about this topic, Marcel Boelaars, a teacher in Fontys who has used both types of databases during his work experience, gave the following answer:

“Relational databases are much harder to scale because you need to upgrade the hardware components of the system. You cannot split up the database into multiple servers - everything needs to be in one place. Non-relational databases, however, scale horizontally - basically having multiple of them on different servers that keep in sync, which is a lot cheaper than scaling vertically“ (Marcel Boelaars, 7.10.2022)

Answer: Relational databases scale vertically and non-relational scale horizontally. The latter is the better option.

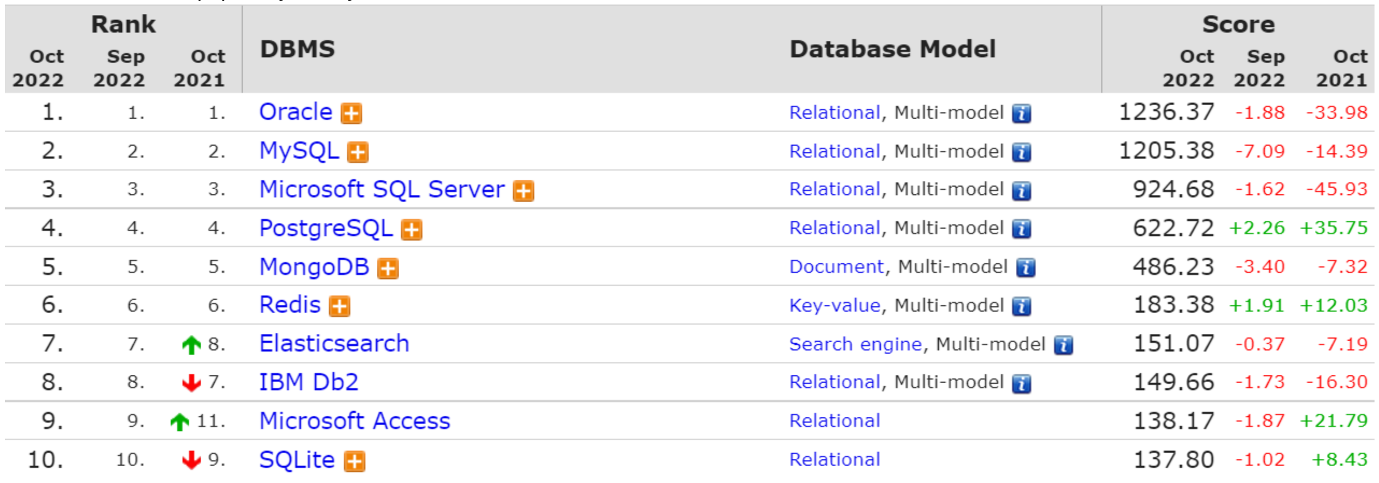
Which type of database has a larger support/community

Source 1: talend.com

“Since relational databases have been around for ages, they have an incredibly large following. SQL databases represent massive communities, stable codebases, and proven standards. Multitudes of examples are posted online and experts are available to support those new to programming relational data.

NoSQL technologies are being adopted quickly, but communities remain smaller and more fractured. However, many SQL languages are proprietary or associated with large single-vendors (such as Mircosoft and Oracle), while NoSQL communities benefit from open systems and concerted commitment to onboarding users.” *(Talend, n.d.-b)*

Source 2: Database popularity survey



Rank			DBMS	Database Model	Score		
Oct 2022	Sep 2022	Oct 2021			Oct 2022	Sep 2022	Oct 2021
1.	1.	1.	Oracle	Relational, Multi-model	1236.37	-1.88	-33.98
2.	2.	2.	MySQL	Relational, Multi-model	1205.38	-7.09	-14.39
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	924.68	-1.62	-45.93
4.	4.	4.	PostgreSQL	Relational, Multi-model	622.72	+2.26	+35.75
5.	5.	5.	MongoDB	Document, Multi-model	486.23	-3.40	-7.32
6.	6.	6.	Redis	Key-value, Multi-model	183.38	+1.91	+12.03
7.	7.	↑ 8.	Elasticsearch	Search engine, Multi-model	151.07	-0.37	-7.19
8.	8.	↓ 7.	IBM Db2	Relational, Multi-model	149.66	-1.73	-16.30
9.	9.	↑ 11.	Microsoft Access	Relational	138.17	-1.87	+21.79
10.	10.	↓ 9.	SQLite	Relational	137.80	-1.02	+8.43

Trust-proven rankings on db-engines show that from the top 10 most popular database management systems, 7 are relational databases, with only 3 being non-relational. The score of the relational databases adds up to 4,411, which is about 5 times more than the score of non-relational databases - 820. We can therefore make the conclusion that relational DBMSs are a lot more widespread and therefore offer more support to any potential issues that arise.

Answer: Relational databases have a far larger support and community than non-relational ones

Why not use both types of databases in one application?

Source 1 - quora.com

In quora, an experienced project lead shared the following:

“It perfectly makes sense. At least this is how I make the best of the both technologies.” *(Simone Scarduzio, 2013)*

NoSQL is simple and quick, but has little to no structure to enforce constraints on the data you are working with. RDBMS satisfies all ACID properties, keeping your data safe and clean. Performance, however, goes down rapidly as traffic and datasets grow.

“You can afford it as long as your NoSQL layer manages to intercept the bulk of the traffic directed to the RDBMS.

There's a lot of tricks to do that using a NoSQL solution:

- **caching** denormalized entities (index by ID the content of pre-joined tables in a key-value store),
- **hosting** all the **derived** data (like precalculated recommendation matrices)
- **hosting** all the **temporary**, fast changing data (like user sessions, shopping carts)

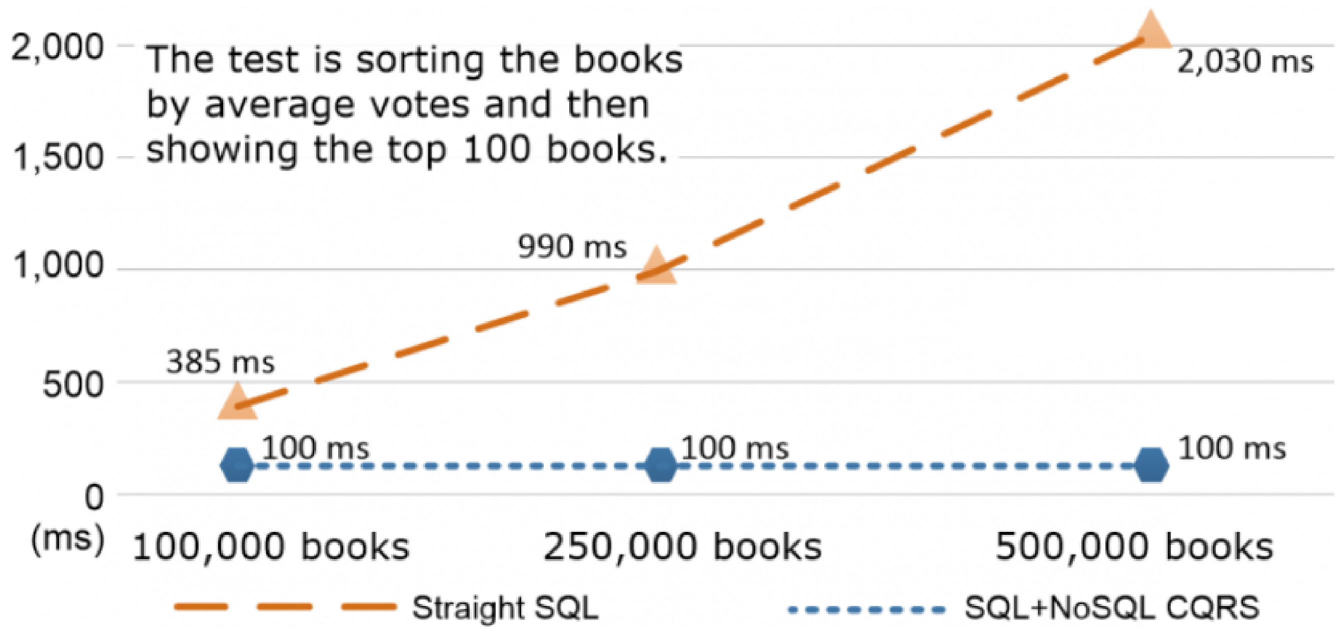
Sometimes you might decide to completely skip the RDBMS part, since your data set is particularly big and you decide you can give up guarantees on consistency, or simply, your data does not have a complex structure and you gladly keep it simple.” ” *(Simone Scarduzio, 2013)*

Source 2 - Gathering test information

A performance test was published by RavenDB, a non-relational database provider that with the following scenario:

The application that was used is a ASP.NET Core application that uses EF Core to display a set of books. Initially, the database was using only a SQL integration. The performance was measured and then it was changed to feature CQRS architecture, utilizing both a SQL database and RavenDB.

Read test



We can see from the read test that the first solution, which utilizes only SQL, gets progressively slowed down as the dataset grows, whereas the SQL+NoSQL integration is unfazed.

Write Test

Solution type	**Total time**	**Notes**
SQL	13 ms	Simple addition of Review entity to the database and a recalculate of the cached values.
SQL+NoSQL CQRS	35 ms	The extra time is mainly around writing to the NoSQL database. RavenDB update measured taking 25ms, which is quite long compared to a SQL write.

From this test we can see that the time it takes to write increases, solely because of the fact that you have to write to two databases rather than one. But there are some ways to handle this. One way would be to pass the update a background task to be executed so that the application returns immediately to the user. The only down side of that approach is the user may be shown data that is not updated. The increase in time is still negligible, though, compared to the time saved in reading from the database.

Effort taken

The table below compares the development effort, complexity and robustness of the two designs: SQL solution and the SQL+NoSQL CQRS.

Solution type	Effort	Complexity	Robustness
SQL	~ 3 days	Complex concurrency	Good
SQL+NoSQL CQRS	~ 8 days	Complex structure	Very good (see below)

The effort was bigger for the SQL+NoSQL CQRS solution, but still in a very sensible time. The big plus was on the robustness of the application. Keeping cached values correct is really difficult, and the SQL+NoSQL does a much better job of that than the SQL did. This is because the SQL+NoSQL CQRS's architecture takes care of any concurrency issues.

Answer: Combining both relational and non-relational databases is beneficial for an an application, but it increases its complexity and is only worth it on a larger dataset.

Results

From answering all of the sub-questions, we can make the following conclusion:

Relational and non-relational databases both have their strengths and weaknesses.

Relational databases are better suited for small business' accounting systems, sales databases, or transactional systems like payment processing in e-commerce. They do not scale well and are not capable of handling large amounts of data.

This is why in recent years many tech giants have been moving to non-relational databases, because of their capability to handle massive amounts of data, while still being easily scalable. They are not as secure as consistent as relational DBMSs, so they are better suited for applications like social media, discussion threads and mobile applications, where the lack of consistency and security is not that big of an issue.

You can, however, get the best of both worlds if you choose to combine these 2 types of databases and utilize their pros while avoiding their cons. It takes more development time to integrate 2 databases in one application, which is only suitable for large scale applications where performance would start to suffer with just a SQL database.

Recommendation

With an application of such a small scale, it does not matter for performance which database would be used. The better choice would be the database that the developer is most comfortable with. In case that the developer wants to upscale the project and it becomes an enterprise-level application, either changing to NoSQL or adapting both SQL and NoSQL databases would be required, as the performance would start to suffer and a simple SQL database would not be able to handle all the data.

References

What Is A Relational Database (RDBMS)? |. (n.d.-b). Google Cloud. Retrieved October 10, 2022, from https://cloud.google.com/learn/what-is-a-relational-database

Introduction to Oracle Database. (n.d.). Retrieved October 10, 2022, from https://docs.oracle.com/cd/E11882_01/server.112/e40540/intro.htm

Non-relational data and NoSQL - Azure Architecture Center. (n.d.). Microsoft Learn. Retrieved October 10, 2022, from https://learn.microsoft.com/en-us/azure/architecture/data-guide/big-data/non-relational-data

MongoDB. (n.d.). *What Is A Non-Relational Database?* Retrieved October 10, 2022, from https://www.mongodb.com/databases/non-relational

Talend. (n.d.). *SQL vs NoSQL: Differences, Databases, and Decisions*. Talend - a Leader in Data Integration & Data Integrity. Retrieved October 10, 2022, from https://www.talend.com/resources/sql-vs-nosql/

ACID properties of transactions. (n.d.). Retrieved October 10, 2022, from https://www.ibm.com/docs/en/cics-ts/5.4?topic=processing-acid-properties-transactions

Estrera, M. (2022, August 5). *Who Uses SQL? Companies That Use SQL and What SQL Is Used For*. Career Karma. Retrieved October 10, 2022, from https://careerkarma.com/blog/who-uses-sql/

Did You Know? Popular Applications that Use NoSQL. (2022, November 10). BangDB. Retrieved October 11, 2022, from https://bangdb.com/blog/applications-that-use-nosql/

Statista. (2022, August 10). *Most popular database management systems worldwide 2022*. Retrieved October 11, 2022, from https://www.statista.com/statistics/809750/worldwide-popularity-ranking-database-management-systems/

GeeksforGeeks. (2020, September 15). *SQL vs NoSQL: Which one is better to use?* Retrieved October 11, 2022, from https://www.geeksforgeeks.org/sql-vs-nosql-which-one-is-better-to-use/

SQL vs NoSQL Exact Differences. (2022, September 29). https://www.softwaretestinghelp.com . Retrieved October 11, 2022, from https://www.softwaretestinghelp.com/sql-vs-nosql/

Otun, O. (2020, January 11). *Relational Database VS NoSQL. RDBMS Are Not Designed For Scale.* Retrieved October 11, 2022, from https://www.linkedin.com/pulse/relational-database-vs-nosql-rdbms-designed-scale-oluwaseyi-otun

Does it make sense to combine both NoSQL and SQL? Why? (n.d.). Quora. Retrieved October 11, 2022, from https://www.quora.com/Does-it-make-sense-to-combine-both-NoSQL-and-SQL-Why

NoSQL Database | *RavenDB ACID NoSQL Document Database*. (2022, September 27). RavenDB NoSQL Database. Retrieved October 11, 2022, from https://ravendb.net