Research Paper on choosing a Database

By STEP 8

June 28, 2023

1 Art of choosing a database

In recent years, the new generation of applications like business intelligence, data warehouses, social networking etc requires processing of terabytes or petabytes of data. This has sparked the market with lots of new types of databases that provide different features for different problems. Choosing a perfect database, among all these databases for your service is one of the most important parameter to consider as it will affect the future of the business and the application. This is where this research paper might help you, by the end of this research paper you would get some knowledge regarding choosing of a database.

Before going directly to the criteria that you might need to consider while choosing a database. Let's build up context around database, its evolution , types and so on. Please be patience and go through all the points as each one of them is as important as other.

2 What is a database?

A database is an organized collection of structured information or data that is stored and managed in a computer system. It is designed to efficiently store, retrieve, and manipulate large amounts of data. It allows users and applications to access and work, while ensuring data integrity, consistency, and security. Databases are used in various domains and industries to store different types of data, such as customer information, financial records, inventory data, scientific data, and more.

Databases are managed by database management systems (DBMS), which provide the tools and functionality to create, modify, and retrieve data from the database. Common types of DBMS include relational database management systems (RDBMS) like MySQL, Oracle, and SQL Server, as well as non-relational or NoSQL databases like MongoDB and Cassandra.

3 Evolution of database

The database started with a file-based system about 50 years ago. In the due time, it has gone through generations of evolution.

- Databases were first introduced in 1968 as flat-file-based databases.
- Then the Hierarchical Database came into existence and lasted until 1980. IBM's first database, IMS (Information Management System) was based on this.
- Charles Bachman developed the first Network data model, called Integrated Data Store (IDS). It was introduced in the early 1960s and standardized in 1971.
- In 1970, the Relational Database was introduced.
- Today, it is the era of Relational Database and Database Management.

4 Types of databases

The following are the types of databases categorized based on various factors:

4.1 Model

4.1.1 Relational database

A relational database stores and provides access to data points that are related to one another. It is based on the relational model and use tables to store and organize data. In a relational database, each row in the table is a record with a unique ID called the key. The columns of the table hold attributes of the data, and each record usually has a value for each attribute, making it easy to establish the relationships among data points. Popular relational database management systems (RDBMS) include MySQL, Oracle, Microsoft SQL Server and PostgreSQL.

4.1.2 Non-relational database

The non-relational database or NoSQL (Not Only SQL) databases are non-relational databases that provide flexible data models and scalability. They are suitable for handling large volumes of unstructured or semi-structured data. However, unlike the relational database, there are no tables, rows, primary keys or foreign keys. Instead, the non-relational database uses a storage model optimized for specific requirements of the type of data being stored.

NoSQL databases can be classified into categories such as

Document databases

Document databases make it easier for developers to store and query data in a database by using the same document-model format they use in their application code. The flexible, semistructured, and hierarchical nature of documents and document databases allows them to evolve with applications' needs. The document model works well with use cases such as catalogs, user profiles and content management systems where each document is unique and evolves over time. Document databases enable flexible indexing, powerful ad hoc queries and analytics over collections of documents. Widely used document databases are Amazon DocumentDB and MongoDB.

Key-value stores

A key-value database uses a simple key-value method to store data. A key-value database stores data as a collection of key-value pairs in which a key serves as a unique identifier. Both keys and values can be anything, ranging from simple objects to complex compound objects. Key-value databases are highly partitionable and allow horizontal scaling at scales that other types of databases cannot achieve. Redis and AmazonDynamo DB are some of the key-value store databases.

Column-based databases

A column-based database helps to store data in columns rather than rows. It is responsible for speeding up the time required to return a particular query. It also is responsible for greatly improving the disk I/O performance. It is helpful in data analytics and data warehousing. Also the major motive of column-based database is to effectively read and write data. Here are some examples for column-based database like Monet DB, Apache Cassandra, SAP Hana, Amazon Redshift.

Graph databases

A graph database's purpose is to make it easy to build and run applications that work with highly connected data sets. Typical use cases for a graph database include social networking, recommendation engines, fraud detection and knowledge graphs. Some of the examples of graph databases are Neo4j, DGraph and JanusGraph.

In-memory databases

In-memory databases are purpose-built databases that rely primarily on memory for data storage, in contrast to databases that store data on disk or SSDs. In-memory data stores are designed to enable minimal response times by eliminating the need to access disks. Because all data is stored and managed exclusively in main memory, in-memory databases risk losing data upon a process or server failure. In-memory databases can persist data on disks by storing each operation in a log or by taking snapshots. In-memory databases are ideal for applications that require microsecond response times or have large spikes in traffic such as gaming leaderboards, session stores, and real-time analytics. Following are some of the in-memory databases, SAP HANA, VoltDB and MemSQL.

4.1.3 Object-oriented database

An object-oriented database is a type of database that is based on the principles of object-oriented programming (OOP). In an object-oriented database, data is organized and stored as objects, which are self-contained units that contain both data and the operations or methods that can be performed on that data. This allows for the efficient representation and management of complex data structures and relationships. Popular object-oriented databases include ConceptBase, Db4o, ObjectDB, etc.

4.2 Location

4.2.1 Centralized

A centralised database is a database in which the data required to complete all of your day-to-day business activities are located, stored and maintained in a single location. Multiple users are able to access the database and it's easier for them to get a complete view of the data due to its single location. Examples of a centralized database are a desktop or server CPU or mainframe computer that users access through a computer network such as a LAN or WAN.

4.2.2 Distributed

Distributed databases store information across different physical sites. The database resides on multiple CPUs on a single site or spread out across various locations. Due to the connections between the distributed databases, the information appears as a single database to end-users. Additionally, if one database fails, users can still access the system through other systems. Common examples of distributed databases include Apache Ignite, Apache Cassandra, Couchbase Server, etc.

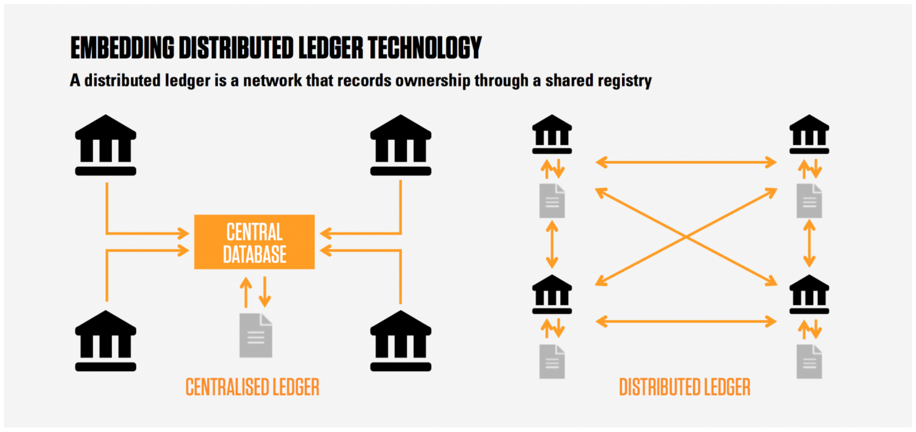
EMBEDDING DISTRIBUTED LEDGER TECHNOLOGY
A distributed ledger is a network that records ownership through a shared registry
CENTRAL DATABASE
CENTRALISED LEDGER
DISTRIBUTED LEDGER

Figure 1: Centralized v/s Distributed

4.3 Design

4.3.1 OLTP

Online Transaction Processing (OLTP) is a type of database system used in transaction-oriented applications, such as many operational systems. 'Online' refers to that such systems are expected to respond to user requests and process them in real-time. Online transaction processing systems are used in business for handling processes like financial transactions, order entries, customer relationship management, and retail sales. OLTP databases need to be ACID-compliant as well. ACID refers to a standardized set of properties that guarantee database transactions are processed reliably, accurately, and consistently.

4.3.2 OLAP

Online Analytical Processing (OLAP) is software technology you can use to analyze business data from different points of view. Organizations collect and store data from multiple data sources, such as websites, applications, smart meters, and internal systems. OLAP combines and groups this data into categories to provide actionable insights for strategic planning. For example, a retailer stores data about all the products it sells, such as color, size, cost, and location. The retailer also collects customer purchase data, such as the name of the items ordered and total sales value, in a different system. OLAP combines the datasets to answer questions such as which color products are more popular or how product placement impacts sales.

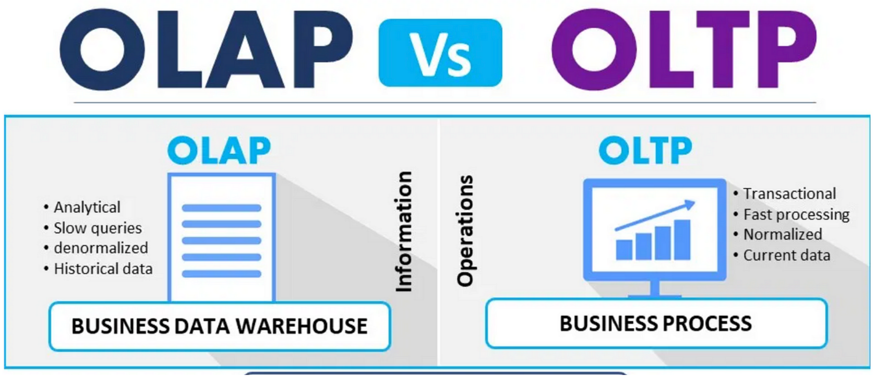

Figure 2: OLAP v/s OLTP

4.4 Hosting

4.4.1 On-premises

On-premises software is installed and runs on computers on the premises of the person or organization using the software, rather than at a remote facility such as a server farm or cloud. Businesses have more control of on-premises assets by maintaining the performance, security and upkeep, as well as the physical location.

4.4.2 Cloud

A cloud database is a database built to run in a public or hybrid cloud environment to help organize, store, and manage data within an organization. Cloud databases organize and store structured, unstructured, and semi-structured data just like traditional on-premises databases. However, they also provide many of the same benefits of cloud computing, including speed, scalability, agility and reduced costs. Popular cloud databases are Microsoft Azure, Amazon Web Service (AWS), Google cloud, etc.

4.5 Processing power

4.5.1 Personal

Personal databases have single-user access and process on low to medium-powered machines. Simpler database applications benefit from this database type due to the low cost and maintenance.

4.5.2 Commercial

A commercial database has multiple users with various permissions as well as numerous applications on high-powered machines. High availability commercial databases are costly and require constant maintenance as well as support.

5 Database selection criteria

Consider the following criteria for choosing the right database for your application:

5.1 CAP consideration

CAP stands for Consistency, Availability, and Partition tolerance. The theorem states that you cannot achieve all the properties at the best level in a single database, as there are natural trade offs between the items. You can only pick two out of three at a time and that totally depends on your prioritize based on your requirements. For example, if your system needs to be available and partition tolerant, then you must be willing to accept some latency in your consistency requirements. Traditional relational databases are a natural fit for the CA side whereas non-relational database engines mostly satisfy AP and CP requirements.

- Consistency means that any read request will return the most recent write. Data consistency is usually “strong” for SQL databases and for NoSQL database consistency may be anything from ’eventual’ to ’strong’.
- Availability means that a non-responding node must respond in a reasonable amount of time. Not every application needs to run 24/7 with 99.999% availability but most likely you will prefer a database with higher availability.
- Partition tolerance means the system will continue to operate despite network or node failures.

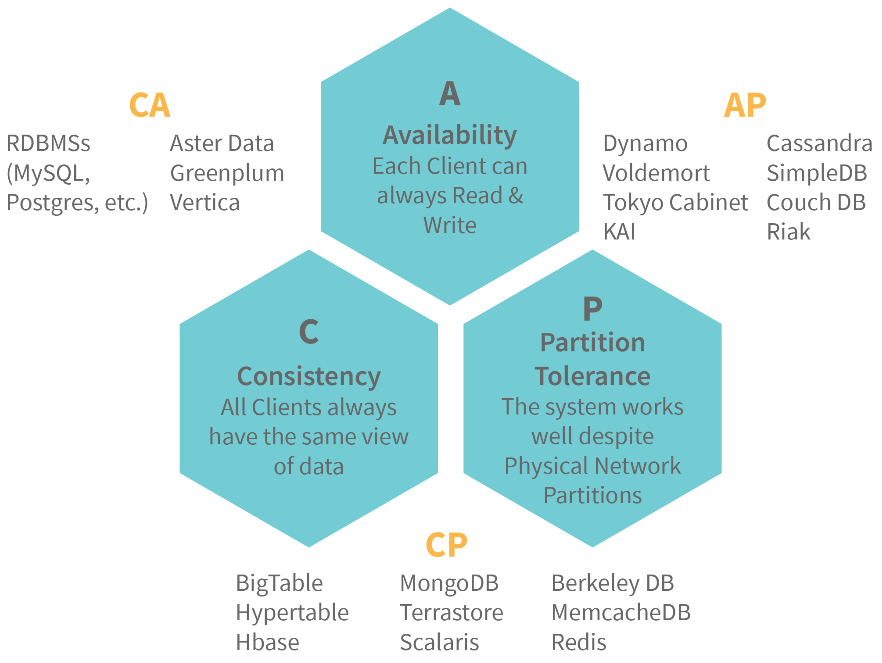

Figure 3: CAP Theorem

5.2 Cost

Cost is always a huge factor and something to consider for a database system. But the cost goes beyond the obvious license costs which need to be considered and evaluated based on the value that is added.

- For buying a software license in the market, there is a large upfront fee usually in the hundreds of thousands of dollars plus a support and maintenance fee.

- There is another cost that is attached while selecting a database that is hardware cost. There is need to consider how much infrastructure needed to buy to run the database system, or if considering a cloud service, consider the cost of hosting the database, whether it adds enough value compared to an on-premises installation.
- On top of hardware and software costs, there are day-to-day costs incurred by running the database. These costs come under operational cost, which has the following things under it:
  - The cost of a temporary or catastrophic challenge needs to be considered when choosing your database for your application. Some of the disasters are old versions, write bottlenecks, memory issues, locked transactions, misconfigurations and hardware failure.
  - Your database does not exist in a vacuum instead integrated with different parts of your IT platform. So, integration between your database and other tools is important and can be costly depending on the database.

5.3 Query Patterns

The way of fetching the data is one of the main ways to find the best database for a particular use case. Some of the types of databases that can be used for fetching the data are listed below :
Key-value store : Key-value store is used to fetch the data using key.
Wide-column databases : Sometimes one or more fields are used to fetch the data, then wide-column databases can be used.
Document or relational databases : There might be a requirement to query using many different fields, then document or relational database can be used.
Search engines : In case of a fuzzy search query capabilities or free text search, search engines can be used.

5.4 Storage Capacity

Most database systems are limited by the amount of space on disk or struggle with performance as amount of nodes and shards grows into the hundreds. Some of the things that can be taken into consideration while choosing database:

- If the data volume fits within gigabytes or less, then any database like in-memory databases should suffice. VoltDB and MemSQL can be used here.
- If the data volume is in the terabytes range, which is equivalent to thousands of gigabytes, there are many more options available such as cloud databases.

- If the data volume grows and touches petabytes, which are millions of gigabytes, then the options of databases shrink. PostgreSQL and Oracle are some of the databases that can be used.

5.5 Security

Security of databases refers to the array of controls, tools, and procedures designed to ensure and safeguard confidentiality, integrity, and accessibility. Security of databases is a complicated and challenging task that requires all aspects of security practices and technologies. The more usable and accessible the database is, the more susceptible we are to threats from security.

Why Database Security is Important?

- Recovering reputation after customer's data leak is difficult
- Penalties or fines are to be paid for not complying with international regulations.
- Costs for repairing breaches and notifying consumers about them is high
- Business can be on hold until the breach is resolved.

Common Threats and Challenges

- An insider threat can be an attack on security from any sources having an access privilege to the database.
- Exploitation of Database Software Vulnerabilities
- SQL/NoSQL Injection Attacks
- Buffer Overflow Attacks
- Denial of service (DoS/DDoS) attacks

5.6 Performance

Performance is another essential factor in choosing a database. It directly impacts the user experience. Query efficiency and the balance between read and write performance should be considered.

Five factors influence database performance: workload, throughput, optimization and contention.

- Workload : Workload measures the level of activity in the database. There are few units that helps measuring the load a database can take:
  - Queries Per Second (QPS): This unit refers to how many queries the database can execute in a given period of time.

  - Transactions Per Second (TPS) : This unit refers to how many transactions an user can execute in a given time period.
  - Latency : Latency is the amount of time taken to process a query.

- Throughput: Throughput defines the overall capability of the hardware and software to process data. It depends on I/O speed, CPU, parallel capabilities of the machine and the efficiency of the operating system. The higher the throughput a database can handle, the more users can simultaneously query it without hurdling the speed of processing.

- Optimization: All types of systems can be optimized, but many database systems can perform query optimization primarily accomplished internal to the DBMS. Yet there are other factors that need to be optimized (SQL formulation, database parameters, database organization, etc.) to enable the database optimizer to create the most efficient access paths to the data.

- Contention: When the demand for a particular resource is high, contention can result. Contention is the condition in which two or more components of the workload are attempting to use a single resource in a conflicting way. As contention increases, throughput decreases.

Therefore, database performance can be defined as the optimization of resource use to increase throughput and minimize contention, enabling the largest possible workload to be processed.

5.7 Concurrency

Data concurrency is the ability to allow multiple users to affect multiple transaction within a database. Simply, data concurrency allows multiple users to access data at the same time.

There are two type of database concurrency used in businesses daily:

- Simultaneous access to data – This kind of concurrency is important because it's all about multiple users accessing data at the same time without causing inconsistencies.

- Coexistent query workload – This type of concurrency is a fundamental measure of system performance. Businesses use the term 'concurrency' to measure how many units of work are co-executing actively and simultaneously progressing at the same time.

For example, when one user is changing data but has not yet saved that data, then the database should not allow other users who query the same data to view the changed, unsaved data. Instead the user should only view the original data.

5.8 Data localization

Data localization is the act of storing the data on any physical device that is present within the borders of a specific country where the data was generated. Free flow of digital data which could impact government operations in a region is restricted by the government. This attempt of providing security for data across borders encourages **data localization**.

While some arguments support data localization, some feel that data localization could have serious harmful consequences to citizens and economics. There are effects of data localization, depending on the context in which it is used. Some of those effects are as follows:

- **Data security** - Data localization can enhance the security of data by keeping it within the borders of a particular country. This is particularly important to handle sensitive information.
- **Economic considerations** - Data localization can provide economic benefits, which helps in economic growth of a particular country.
- **Improved performance** - Data localization can improve the performance of online applications and websites, by keeping data closer to users.
- **Cost** - One of the main challenge in data localization is cost, which includes building or renting data centers, as well as cost of complying with multiple laws and regulations.
- **Limited Access** - Data localization can limit access to data as it may not be accessible to users outside the country or region in which it is stored.
- **Complexity** - It can also be complex to implement as it involves transferring data between different locations.

5.9 API Compatibility

API compatibility of database refers to the ability of different versions of a database to interact with each other using a consistent set of **Application Programming Interfaces**(APIs). API Compatibility ensures that applications built in one version can seamlessly migrate with another version without significant modifications. The level of the API Compatibility can vary depending on the specific database management system(DBMS) being used. Some databases like MySQL and PostgreSQL, have well-defined and widely adopted APIs which are supported by various programming languages and frameworks.

On the other hand, proprietary databases with unique features may have their own APIs that are specific to that particular system. In such cases compatibility may be limited to the applications developed for that database.

To ensure API Compatibility, it is important to consider factor like version of the database software, language and framework being used. It is beneficial to consider the documentation and resources provided by the database to understand the APIs and its compatibility.

5.10 Integration with outside data

Integration with outside data refers to the process of bringing data from multiple source together across an organization to provide a complete, accurate and up-to-date applications and business processes. It includes data replication, ingestion and transformation to combine different type of data to be stored in a target place such as a data warehouse or data lake-house.

Following are the primary approaches to integrate data with external services:

- **ETL** - It is a traditional type of data-pipeline which converts raw data to match the target system via extract, transform and load. This allows for fast and accurate data analysis in the target system and is most appropriate for small datasets which requires complex transformations.
- **ELT** - This is also a data pipeline that supports immediately load and transformation of data within target system. This approach is more appropriate when datasets are large and timeliness is important, since loading is quite quicker.
- **Data Streaming** - Instead of loading data into a new target system, data streaming provides continuous flowing of data from source to target.
- **Application Integration** - Application Integration allows separate applications to work together by moving or syncing data between them. Mostly the applications have their APIs for transactions of data, which helps automation tools to maintain integration efficiently.
- **Data Virtualization** - Like Data Streaming, Data Virtualizations also delivers data in time, but only when it is requested by a user or an application. This unifies view of data and makes data available on demand by combining data from different systems.

5.11 Replication

Database replication involves copying, transferring, or integrating data from one database in a server or computer to another, eventually creating a distributed database. Replication allows users to access the same information, which improves consistency, reliability, and performance. Data replication can either happen once or it can be a continuous process. The result is one or more distributed databases, where users have access to the same information across all database nodes. The original database is called the 'Publisher'. The replicated database is called the 'Subscriber'.

Data Replication Types

- Transactional Replication - The distributed database management system (DDBMS) replicates changes or transactions made to the original database on the receiving database in a sequence in near real time.

- Snapshot Replication - The DDBMS captures a 'snapshot' of data from the original database and overwrites it on the receiving database via the same server.
- Merge Replication - The DDBMS merges data from two or more databases and combines it into a new receiving database.

5.12 Hosting

Database hosting is providing a managed server that is optimized for running a database. A host can run a web server, mail server, file share, or other file-based application. Databases are different in that they demand specific hardware and software configurations to support the database.

- A database host must optimize I/O for the specific database and storage platform for optimal performance, growth and internal resiliency.
- A database host must run certified versions of certified operating systems to ensure your database support licenses are valid.
- Database hosting providers should be able to help you design a specific backup plan for your database and host based on your recovery requirements.

5.13 Effect on environment

To store, process and transfer data, a server needs a lot of electricity and water. These tasks constitute only half of its actually energy consumption: nearly 50% of the electricity is used to cool the servers.

Followings are the links between data center and climate change.

- Data centers use nearly 3% of the world's electricity consumption.
- Data centers account for 2% of total greenhouse gas emissions.
- Data centers are responsible for 15% of the IT sector's carbon footprint and 18% of the digital pollution.
- In 2018, it was estimated that the carbon footprint of data centers was equivalent to the entire aviation - industry!
- Some of the largest data centers can contain hundreds of thousands of servers and IT devices using more than - 100 megawatts. That's what 80,000 American homes consume in electricity.
- To cool its 800 data centers, California alone uses the equivalent of 158,000 Olympic-sized swimming pools per year to cool its servers. The water used by the data centers is mostly discharged directly into the sewer - system, causing damage to the environment, the economy and the population.

- Per year, the water used by a medium-sized data center is equivalent to that of three hospitals!
- In 2016 (that was 7 years ago!), it was calculated that, data centers consumed globally more electricity than Great Britain alone (416.2 vs 300 TWh).

5.14 Maturity and Stability

Choosing the most trendy, powerful, and fully featured database to self-host maybe tempting, but as long as your organization doesn't have experience with this database, you may end up regretting it. SQL databases are built on mature technologies that are well known and supported by large developer communities. NoSQL products are not as mature and the technologies not as well supported as SQL products, but NoSQL technologies are making fast inroads into the industry, with developer communities constantly growing.

Setup, configuration and fine tuning of databases is a lengthy and risky ordeal. Sometimes choosing the "old" organization self-hosted work-horse will pay bigger dividends in the long term when it comes to production stability.

5.15 Learning curve

Evaluate the ease of adoption by assessing factors like documentation, learning curves, and available tools. Choosing a database with a supportive community and an easy learning curve will streamline your development process and accelerate the integration of the database into your project.

Some databases structure and style are very plain because of which they share a gentle learning curve, which is much easier to form a team to manage your database. Whereas some databases, while covering many issues, can sometimes be overwhelming and even confusing. So, to install and run such databases, you'll have to consider hiring dedicated experts.

5.16 Community

Consider the strength and vibrancy of the community surrounding a database. A thriving community offers support, resources, and active development, which can be invaluable when facing challenges or seeking enhancements. It also ensures that your requests and feedback are heard and considered in the road-map. A strong database community saves the developer from wasting his time, which indeed results in a quick and smooth development cycle. This criteria plays an important role if you are in a rush or have delivery pressure.

5.17 Scalability

The database is an ever-growing asset for an organization, so it is important to make sure that the DBMS that is being chosen can scale up to accommodate the growing volume as the enterprise grows. While considering this, there might be a need to keep the changing nature of data from structured database formats to unstructured data in mind. So, ensure that unconventional DBMS systems are considered too, which can effectively handle unstructured or semi-structured data. The NoSQL and NewSQL databases are now focused on these needs.

6 Conclusion

To conclude, we will just say that a lot of people think that choosing a database is basically choosing between SQL and no SQL, this is incorrect. Selecting the perfect database requires a comprehensive evaluation of multiple factors, as you saw. Remember, each project is different and the ideal database choice may vary based on specific requirements. The most important factors are to consider the structure of the data, the size of the database, the speed you need to access the data, as well as the scaling of the database. So take the time to thoroughly assess each consideration, and seek expert advice if needed.

With the right database powering the project, there'll be a solid foundation for success and will unlock the full potential of an application.

7 References

- Basics to choose the database
- Few tips to choose database
- CAP Theorem
- Data Categories
- How to efficiently choose a database
- What is data concurrency?
- Evolution of database
- Types of databases
- What is a relational database
- Non-relational database
- ODBMS
- Location databases
- OLAP
- Cloud
- Database Security

- Replication
- Cost
- Hosting
- Effect on environment
- Performance
- How to Choose a Database for Your Needs?
- Scalability

Authors : Subhash Yadav, Vivek Bisht, Prajakta, Abin, Sourav B
28 June 2023