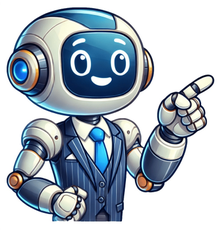
Click to prove you're human



# Examples of non relational databases

Databases evolved significantly over the years, with most being non-relational until the mid-1980s when DB2 and Oracle emerged. Prior to this, databases were mostly flat files or hierarchical structures. Hierarchical databases were more complex but offered better performance than flat files. DB/1, IBM's predecessor to DB2, was a hierarchical database that used a file with 'root' records accessible by key. Each root record could have child records, forming a tree-like structure. This allowed for efficient data access, despite limitations such as the need to traverse the tree. The use of hierarchical databases had drawbacks, including the requirement for full programs to perform tasks, which was time-consuming compared to modern SQL. However, DB2's implementation on top of VSAM provided improved execution speed, making it a valuable solution in mainframe systems where processing power and CPU time were comparable in value. VSAM remains an important consideration in mainframe systems today, while DB/1's hierarchical structure paved the way for future database developments. NoSQL databases offer flexibility and scalability to handle large-scale, unstructured, and semi-structured data efficiently. Unlike traditional relational databases, NoSQL databases provide high-performance solutions for modern applications such as big data analytics, real-time systems, cloud computing, and distributed systems. NoSQL databases can be classified into four main types: Document-based databases, Key-value stores, Column-oriented databases, and Graph-based databases. Each type has unique advantages and use cases, making NoSQL a preferred choice for specific applications. Document-based databases store data in JSON, BSON, or XML documents, providing a flexible schema that allows for easy querying and retrieval of data. Collections are groups of documents with similar contents, and document databases have features such as faster creation and maintenance, no foreign keys, and open formats like XML and JSON. The advantages of document databases include their flexibility, ease of use, and fast query performance. They are often used in content management, product catalogs, and user data storage. NoSQL database types are categorized into four main types - Key-Value Stores, Column Oriented Databases, Graph Based Databases and Document Based. NoSQL means "not only SQL" which is a relational database technology that uses SQL to interact with the data stored in the database. NoSQL databases offer flexibility, scalability, and high performance, making them crucial for modern applications handling big data, real-time analytics, and distributed systems. Selecting the right NoSQL database type depends on data structure, scalability requirements, and query performance needs. Understanding these types and their advantages enables businesses and developers to make informed decisions to optimize performance and scalability. Non-relational databases, also known as NoSQL databases, offer greater flexibility when storing data compared to traditional relational databases. They are particularly useful when dealing with unknown or varying data structures, large volumes of data, and the need to compare complex attributes side-by-side. When choosing a non-relational database, consider factors such as latency, complexity, stability, price, and scalability. NoSQL databases scale well horizontally, making them better suited for distributed systems like cloud infrastructure. They can handle complex queries and large datasets efficiently. In contrast to relational databases, which rely on rigid table structures and predefined schema, NoSQL databases store data in a more flexible manner, allowing for faster lookups and easier management of varying data types. Non-relational databases encompass four main types that function differently from one another. We'll examine each of these categories below. Graph databases store data as nodes and edges with equal importance. They're ideal for analyzing large datasets to identify relationships between seemingly unrelated points. Applications include detecting financial fraud, tracking disease trajectories in healthcare, and managing user relationships on social networks. Neo4j is a prominent graph database example. A sample graph database works like this: imagine mapping ice cream flavors, ingredients, and brands onto a graph using CONTAINS and SELLS relationships. These connections enable traversing the graph, making queries, and analyzing data. Key-value databases feature unique keys pointing to specific values, similar to Javascript objects or Python dictionaries. They're known for their speed and are often used for caching, message queues, and managing user profiles. Examples include Redis and Amazon DynamoDB. A sample key-value database query would retrieve the entire JSON object associated with a given key. Document-oriented databases are multi-purpose, storing data as JSON documents with exposed keys for querying. Documents can have varying structures, but collections offer hierarchical organization, enabling relational queries without sacrificing flexibility. Document databases are used in online shopping carts, gaming, and content management. Examples include MongoDB and Amazon DynamoDB (which is also a hybrid key-value database). Given article text here 1. Data Flexibility and Adaptability: Databases can store varying amounts of data and allow users to add or remove fields as needed. 2. Wide-Column Database Structure: Each piece of data is organized in a table-like format, with flexible column types that may change for each row. 3. Characteristics Similar to Hybrid Databases: Wide-column databases combine elements of key-value databases and relational databases, ideal for handling diverse data types and large datasets. 4. Common Usage and Popular Options: Cassandra and HBase are frequently employed due to their ability to handle varied data structures. 5. Evolution of Database Types: Originally non-relational, the advent of DB2 and Oracle in the 1980s led to widespread adoption, replacing earlier flat-file or hierarchical database systems. 6. Historical Context: Early databases were either flat files or hierarchical, with limitations such as requiring full program execution for tasks that can now be accomplished through SQL in mere minutes. 7. Execution Speed Advantages: Hierarchical databases excelled in performance, particularly on mainframe systems, due to their optimized data I/O capabilities and lower CPU requirements compared to programmer time. DB2's hierarchical implementation, built on top of VSAM, is still relevant today. VSAM, used on mainframe systems, remains a significant consideration. DB/1, IBM's predecessor to DB2, was a prime-time database with a hierarchical structure, featuring root records that could have child records, forming tree-like structures. Accessing child records required traversing the tree due to limitations on direct access, leading to inefficiencies in data management and query performance. Although execution speed was an advantage, poor queries could be detrimental to expensive installations. DB/1's cost-effectiveness, with programmer costs outweighing CPU time, made sense at the time.