Distributed Databases: SQL vs. NoSQL

Gul Seda Unal [21], Yuchen Zheng [87]
University of Rochester, New York

Introduction

Distributed Database is a database where portions of the database are stored in multiple physical locations and processing is distributed among multiple database nodes.
SQL Structured Query Language, is the standard language for relational database management systems.
NoSQL The need to store, process and analyze the unstructured data led to the development of schema-less alternatives to SQL, namely NoSQL known as not only SQL.
SQL vs. NoSQL for Distributed Databases a thorough comparison between SQL and NoSQL for distributed databases is provided on many aspects from model features, data integrity and flexibility to scalability.

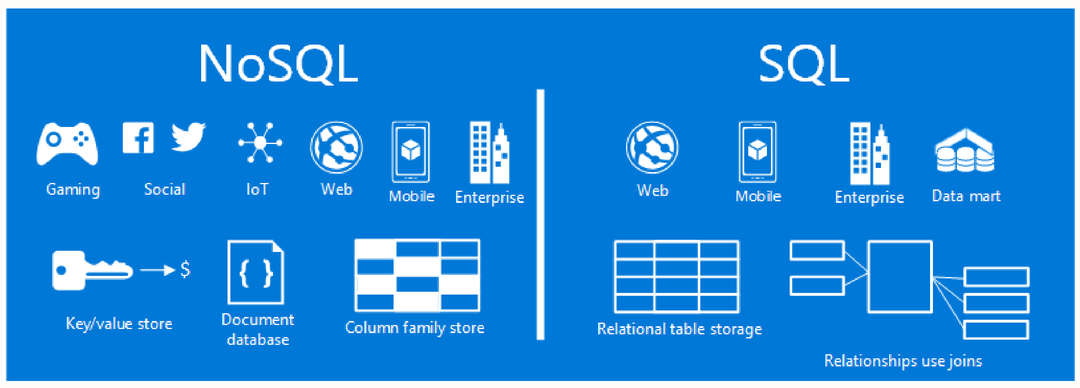

Figure 1: SQL Database for Social Media [1]

SQL vs. NoSQL in Distributed Databases

SQL and NoSQL are both great inventions to keep data storage and retrieval optimized and smooth with one having some advantages over the other in certain scenarios.
NoSQL has gained greater interest with the increasing demand in social networks. Therefore, a nice comparison of SQL and NoSQL can be achieved through a social media database where users interact with each other by posts and comments.
Consider a social platform where users engage with each other in terms of posts associated with images, audio, video, comments, links to websites in a live stream. Using SQL a query with many joins is necessary to retrieve the content. With SQL, many queries and joins will be necessary to complete the task.

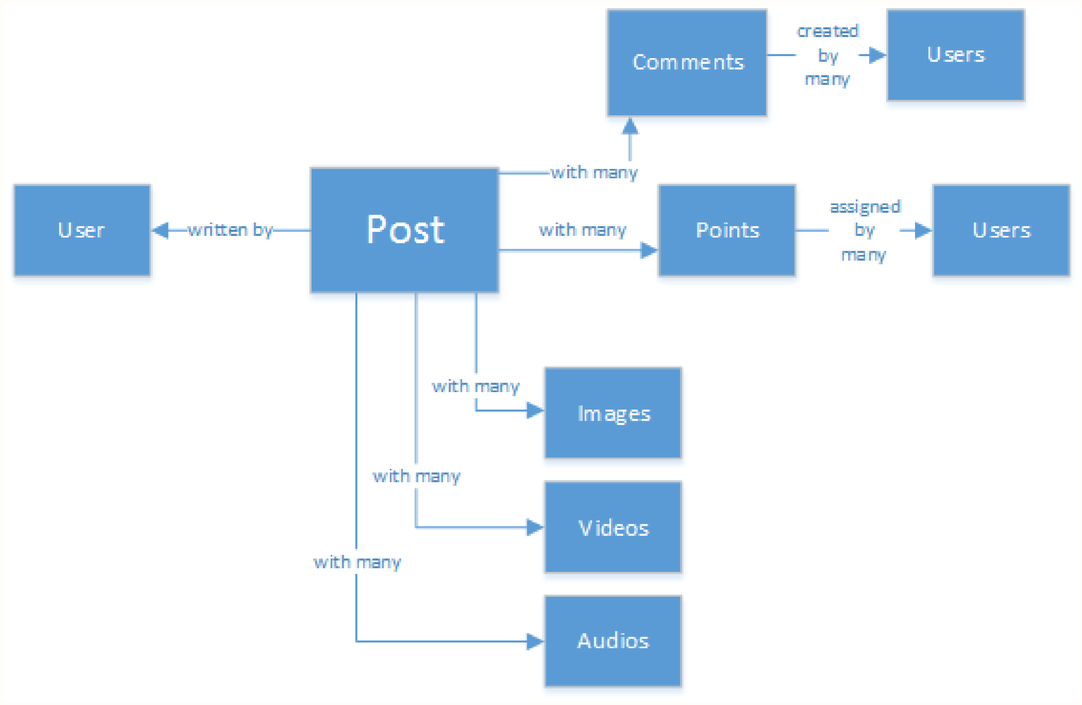

Figure 2: SQL Database for Social Media [1]

A solution to this problem from a relational perspective can be using JSON, as it is the supported dynamic data format in SQL. Another approach, from a nonrelational perspective would be using NoSQL. NoSQL simplifies the approach for this specific scenario.
Due to its simplicity, the use of NoSQL has grown with the social media platforms in order to successfully handle the growing need of IoT (Internet of Things).

SQL

Scalability in relational SQL databases has been an important issue until Google announcing F1, a SQL database that is trivial to scale up, to run at the core of AdWords business; SQL is guaranteed to be available for distributed databases [2].

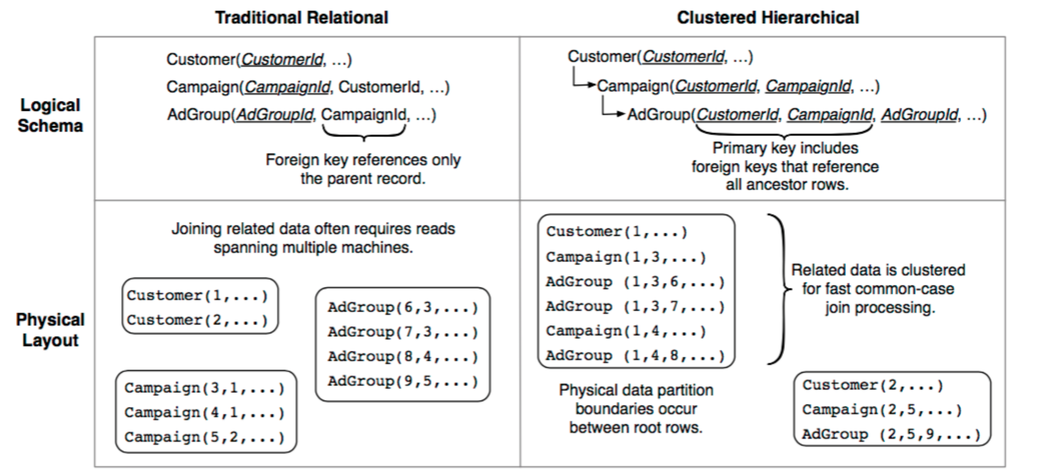

Figure 3: Google's AdWords business running on SQL distributed (scalable) database [2]

NoSQL

SQL vs. NoSQL

	NoSQL	SQL
Model	Non-relational	Relational
	Stores data in JSON documents, key/value pairs, wide column stores, or graphs	Stores data in a table
Data	Offers flexibility as not every record needs to store the same properties	Great for solutions where every record has the same properties
	New properties can be added on the fly	Adding a new property may require altering schemas or backfilling data
	Relationships are often captured by denormalizing data and presenting all data for an object in a single record	Relationships are often captured in normalized model using joins to resolve references across tables
	Good for semi-structured, complex, or nested data	Good for structured data
Schema	Dynamic or flexible schemas	Strict schema
	Database is schema-agnostic and the schema is dictated by the application. This allows for agility and highly iterative development	Schema must be maintained and kept in sync between application and database
Transactions	ACID transaction support varies per solution	Supports ACID transactions
Consistency & Availability	Eventual to strong consistency supported, depending on solution	Strong consistency enforced
	Consistency, availability, and performance can be traded to meet the needs of the application (CAP theorem)	Consistency is prioritized over availability and performance
Performance	Performance can be maximized by reducing consistency, if needed	Insert and update performance is dependent upon how fast a write is committed, as strong consistency is enforced. Performance can be maximized by using scaling up available resources and using in-memory structures.
	All information about an entity is typically in a single record, so an update can happen in one operation	Information about an entity may be spread across many tables or rows, requiring many joins to complete an update or a query
Scale	Scaling is typically achieved horizontally with data partitioned to span servers	Scaling is typically achieved vertically with more server resources

Figure 4: SQL vs NoSQL [1]

Conclusions

Distributed databases have better reliability, higher speed and low communication cost compared to the traditional databases. With big data and increasing user activity in social platforms distributed databases are in increasing demand.
NoSQL supporting a nonrelational, flexible, dynamic and horizontally scalable databases is the frequently selected programming language for distributed database systems. SQL, on the other hand, enforces strict schema, structured data, strong consistency and vertical scalability. Depending on the needs, the best fit between SQL and NoSQL can be decided. In distributed systems, due the the scalability problems of SQL, mostly NoSQL is preferred. However, there is evidence that scalability in SQL can be achieved with clustered hierarchical distributed databases [2].

References

[1] NoSQL vs. SQL, Microsoft Azure: https://docs.microsoft.com/en-us/azure/documentdb/documentdb-nosql-vs-sql
[2] Ian Rae et.al. August 26th 2013, Proceedings of the VLDB Endowment, Vol. 6, No. 11