NoSQL databases have been quite successful in very specific domains. The most popular avenue for NoSQL systems is the storage of big data for web-scale companies whose applications that can tolerate reduced consistency guarantees. However, this approach pushes some of the storage challenges to the application developer. As an example, the non-relational data model means that developers may have to implement their own joins if they ever need to combine data from two tables. They also have to handle eventually consistent data and ensure that their applications do not face any correctness issues with the lack of transactions. However, not all applications can give up these strong transactional semantics. There is a demand for database systems that can combine the best of both the relational and NoSQL worlds. These systems work using a relational model and SQL, with ACID transactions, while offering scalable performance that is similar to NoSQL systems.

NewSQL databases refer to the next generation of relational DBMSs that can scale like a NoSQL system, without fully giving up on SQL or some level of ACID properties for transactions. This can be done using multiple techniques, the most popular of which we will discuss here:

- **Shared-nothing architectures**: A common design pattern in NewSQL systems is the adoption of a shared-nothing architecture. This is a system in which each node is completely independent (sometimes down to the level of individual threads), thus ensuring that there is no single point of contention across the system. This enables systems to have a high performance at scale because there is no need for expensive locking protocols.
- **In-memory systems**: Another avenue for improving performance in database systems is to reduce the number of trips made to disk for a given query. An extreme version of this philosophy is to operate an entire database from memory, so that the database never has to go to disk.

H-Store and VoltDB

H-Store is an experimental example of a NewSQL system, designed by a team at Brown University, Carnegie Mellon University, the Massachusetts Institute of Technology, and Yale University. H-Store is deployed on a cluster of nodes, using a shared-nothing architecture. At the heart of H-Store is a highly optimized single-threaded database engine that quickly processes individual queries. The database is then sharded across the cluster such that every individual core is responsible for a disjoint subset of the data. The data in H-Store is stored in the memory of each of the nodes in the system.

Because each engine has exclusive access to all of the data at a partition, only one transaction at a time is able to access the data stored in the partition, eliminating the need for locks and latches in the system. As a result, no transaction will stall waiting for another transaction once started, at least for queries that do not span a single partition.

H-Store has its limitations. First, the lack of durable storage (as H-Store stores all data in-memory), coupled with the shared-nothing architecture, means that node failures can cause loss of data.

While H-Store was an academic project, the commercial realization of H-Store emerged as VoltDB. VoltDB has been improving and adding features to H-Store in order to address the shortcomings of VoltDB, including a logging mode to enhance the durability of the storage system.

Check your knowledge

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue.