VOLTDB

Oracle vs. NoSQL vs. NewSQL

Comparing Database Technology

John Ryan
Senior Solution Architect, Snowflake Computing

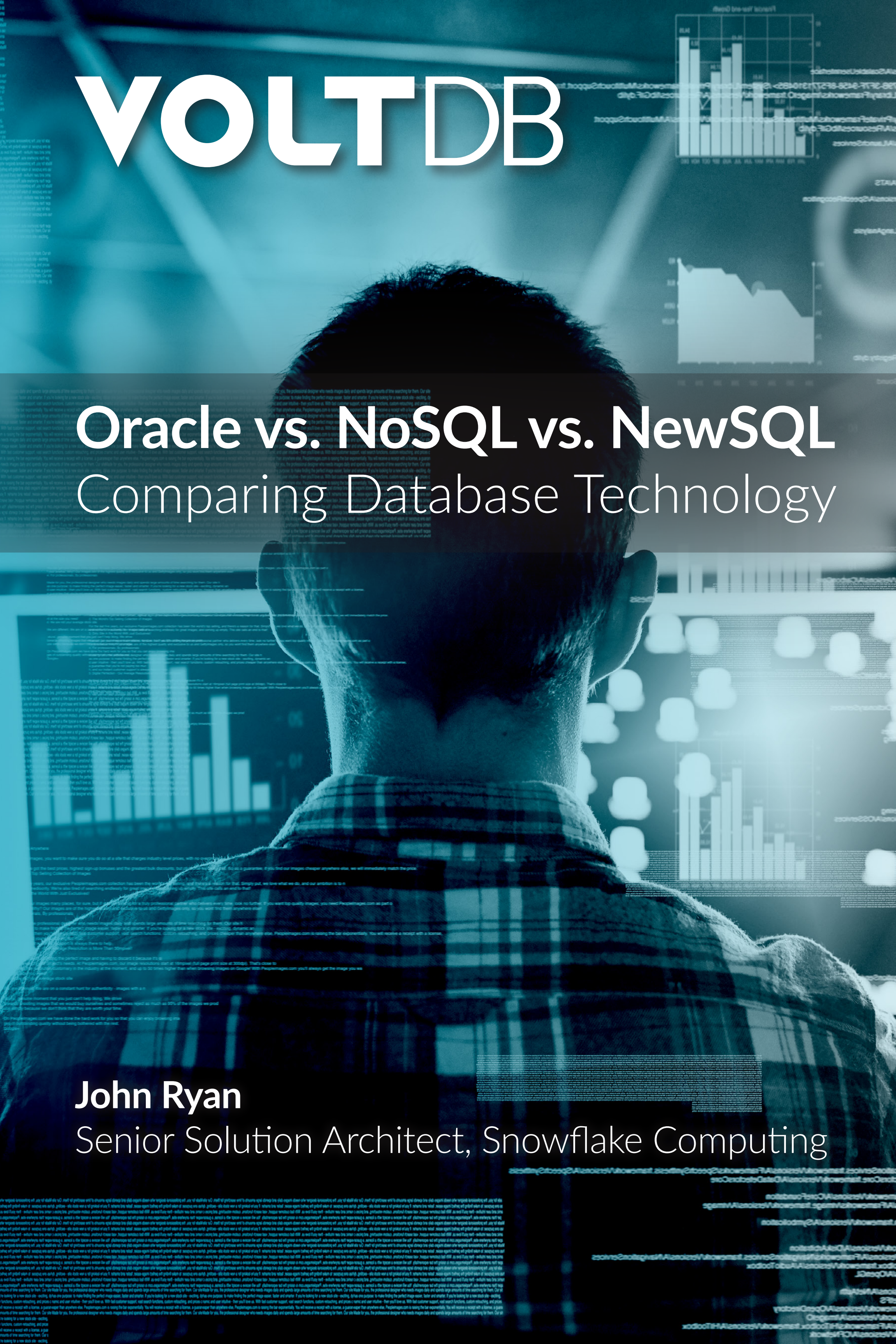


Table of Contents

The World has Changed

The world has changed massively in the past 20 years. Back in the year 2000, a few million users connected to the web using a 56k modem attached to a PC, and Amazon only sold books. Now billions of people are using to their smartphone or tablet 24x7 to buy just about everything, and they're interacting with Facebook, Twitter and Instagram. The pace has been unstoppable.

Expectations have also changed. If a web page doesn't refresh within seconds we're quickly frustrated, and go elsewhere. If a web site is down, we fear it's the end of civilisation as we know it. If a major site is down, it makes global headlines.

> *Instant gratification takes too long!*
>
> *— Ladawn Clare-Panton*

Aside: *If you're not a seasoned Database Architect, you may want to start with my previous articles on Scalability and Database Architecture*.

What's Changed?

The above leads to a few observations:

- **Scalability** — With potentially explosive traffic growth, IT systems need to quickly grow to meet exponential numbers of transactions
- **High Availability** — IT systems must run 24x7, and be resilient to failure. (A failure at Bank of America in 2011 affected 29 million customers over six days).
- **High Performance** — In tandem with incremental scalability, performance must remain stable, and fast. At the extreme end, Amazon estimates it loses $1.6B a year for each additional second it takes a page to load.
- **Velocity** — As web connected sensors are increasingly built into machines (your smartphone being the obvious one), transactions can repeatedly arrive at millions of transactions per second.
- **Real Time Analytics** — Nightly batch processing and Business Intelligence is no longer acceptable. The line between analytic and operational processing is becoming blurred, and increasingly there are demands for real time decision making.

The Internet of Things is sending velocity through the roof!
— Dr Stonebraker (MIT).

The above demands have led to the truly awful marketing term *Translytical Databases* which refer to hybrid solutions that handle both high throughput transactions and real time analytics in the same solution.

What's the Problem?

The challenge faced by all database vendors is to provide high performance solutions while reducing costs (perhaps using commodity servers). But there are conflicting demands:

- **Performance** — To minimise latency, and process transactions in milli-seconds.
- **Availability** — The ability to keep going, even if one or more nodes in the system fail, or are temporarily disconnected from the network.
- **Scalability** — The ability to incrementally scale to massive data volumes and transaction velocity.
- **Consistency** — To provide consistent, accurate results — particularly in the event of network failures.
- **Durability** — To ensure changes once committed are not lost.
- **Flexibility** — Providing a general purpose database solution to support both transactional and analytic workloads.

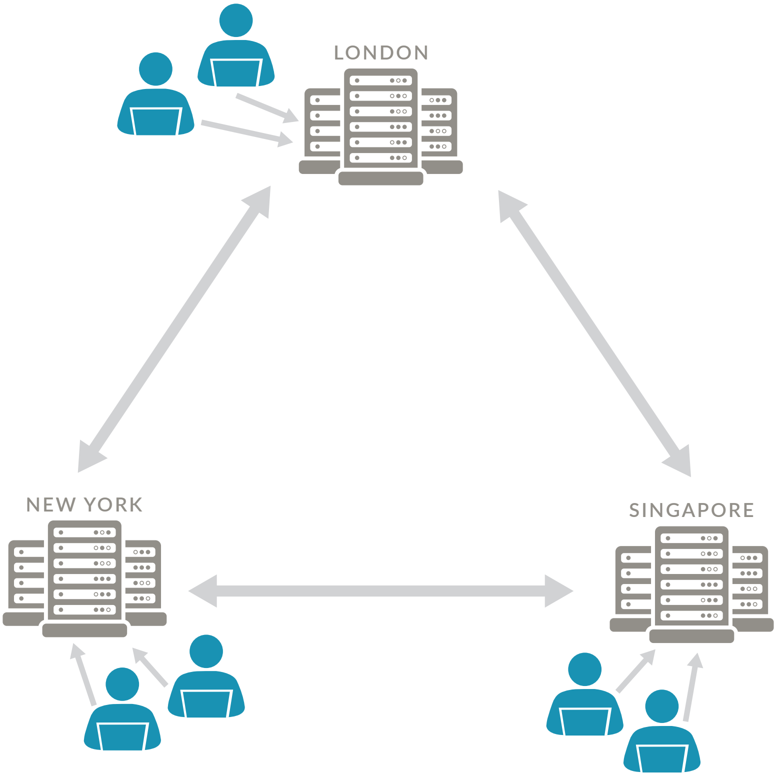
The only realistic way to provide massive incremental scalability is to deploy a scale-out distributed system. Typically, to maximise availability, changes applied on one node are immediately replicated to two or more others. However, once you distribute data across servers, you face trade-offs.

For example:

Performance vs. Availability and Durability

Many NoSQL databases replicate data to other nodes in the cluster to improve availability. If immediately following a write, the database node crashes, the data is available on other machines, and changes are therefore durable. It's possible, however, to relax this requirement, and return immediately. This maximises performance at the risk of losing the change. The change may not be durable after all.

Geographically Distributed System



Consistency vs. Availability

NoSQL databases support eventual consistency. For example, in the above diagram, if network connectivity to New York temporarily fails there are two options:

- **Stop Processing** — But availability suffers in New York
- **Accept Reads/Writes** — And resolve the differences once reconnected. But this risks giving out-of-date or incorrect results, and conflicting writes need to be resolved.

Clearly, NoSQL databases trade consistency for availability ability.

Flexibility vs. Scalability

Compared to general purpose relational systems like Oracle and DB2, NoSQL databases are relatively inflexible, and don't (for example) support join operations. In addition to many not supporting the SQL language, some (eg. Neo4J and MongoDB) are designed to support specific problem spaces — Graph processing and JSON data structures.

Even databases like HBase, Cassandra, and Redis abandon relational joins, and many limit access to a single primary key with no support for secondary indexes.

> *Many databases claim 100% ACID transactions.*
> *In reality few provide formal ACID guarantees.*
> *— Dr Peter Bailis (University of Stanford)*

ACID vs. Eventual Consistency

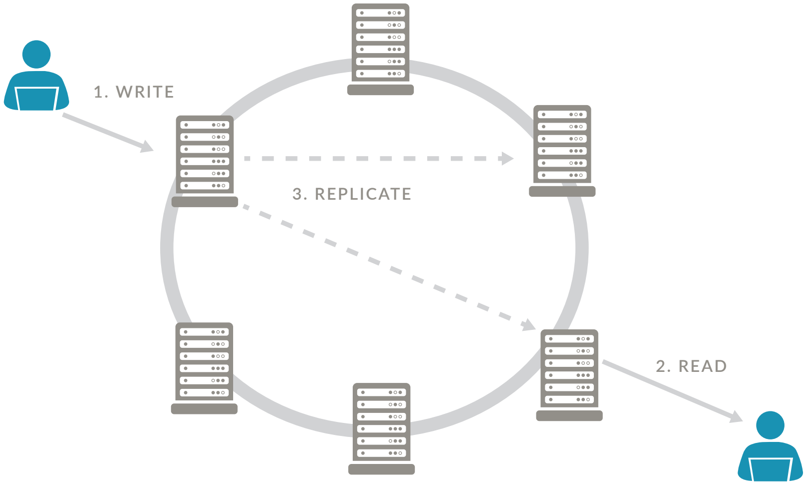
One of the major challenges in scaling database solutions is maintaining ACID consistency. Amazon solved the performance problem with the DynamoDB database by relaxing the consistency constraints in favour of speed which led to a raft of NoSQL databases.

As an aside, even the most successful databases (including Oracle), don't provide true ACID isolation. Of 18 databases surveyed, only three databases (VoltDB, Ingres, and Berkeley DB) were found to support Serializability by default. The primary reason is it's difficult to achieve while maintaining performance.

> *Eventual consistency is a particularly weak model.*
> *The system can return any data, and still be eventually consistent.*
> *— Dr Peter Bailis (Stanford)*

Eventual consistency, on the other hand, provides almost no consistency guarantees. The diagram below illustrates the problem with eventual consistency. One user deducts $1m from a bank account, but before the changes are replicated, a second user checks the balance. The only guarantee, provided there are no further writes, is the system will eventually provide a consistent result. How this this even useful, let alone acceptable?

Cassandra — Eventual Consistency



The OLTP Database Reimagined

Ten years ago, Dr. Michael Stonebraker wrote the paper The End of an Architectural Era, where he argued the 1970s architecture of databases from Oracle, Microsoft and IBM were no longer fit for purpose.

He stated an OLTP database should be:

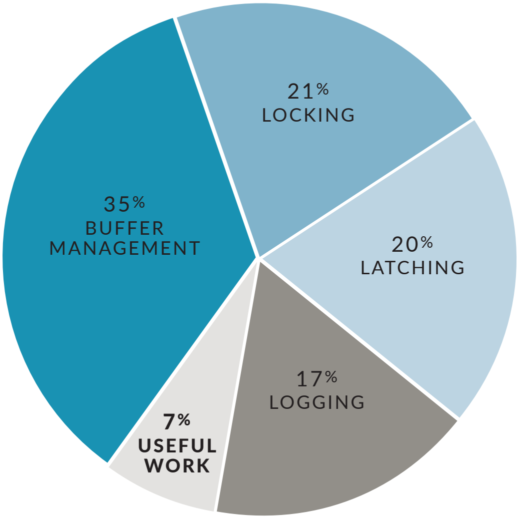
- **Dedicated to a Single Problem** — To quickly execute short lived, predefined (not ad-hoc), transactions with a relatively simple query plan. In short, a dedicated OLTP platform.
- **ACID Compliant** — With all transactions running single threaded, providing full Serializability by default.
- **Always Available** — Using data replication (not a hot standby) to provide high availability at almost no additional cost.
- **Geographically Distributed** — Running seamlessly on a grid of dispersed machines (adding further resilience, and providing local performance benefits)
- **A Shared Nothing Architecture** — With load dispersed across multiple machines connected as a peer-to-peer grid. Adding a machine is a seamless operation with zero downtime, and loss of a node results in a marginal performance degradation rather than full system down time.
- **Memory Based** — Entirely run in memory for absolute speed, with durability provided by in-memory data replication other nodes.
- **Eliminate Bottlenecks** — Achieve massive throughput by completely re-designing the database internals to run single threaded while removing the need for redo logs, and locking and latching — the most significant constraints on database performance.

To demonstrate the above was feasible, he built a prototype, the H-Store database, and demonstrated a TPC-C benchmark performance of 82 times faster than an unnamed commercial rival on the same hardware. The H-Store prototype achieved a remarkable 70,000 transactions per second compared to just 850 from the commercial rival, despite significant DBA tuning effort.

Achieving the Impossible!

Dr. Stonebraker's achievement is remarkable. The previous TCP-C world record was around 1,000 transactions per CPU core, and yet H-Store achieved 35 times that on a dual-core 2.8GHz desktop machine. In his 2008 paper OLTP through the Looking Glass, he went on explain why commercial databases (including Oracle) perform so badly.

Relational Database Processing Effort



The diagram above illustrates the 93% overhead built in to a traditional (legacy) database including locking, latching and buffer management. In total, just 7% of machine resources are dedicated to the task at hand.

H-Store was able to achieve the seemingly impossible task of full ACID transactional consistency, orders of magnitude faster, by simply eliminating these bottlenecks, and using memory rather than disk based processing.

NewSQL Database Technology

First released in 2010, VoltDB is the commercial implementation of the H-Store prototype, and is a dedicated OLTP platform for web scale transaction processing and real time analytics. As this infographic demonstrates, there are 250 commercially available database solutions, of which just 13 are classified as NewSQL technology.

VoltDB

In common with other NewSQL databases, VoltDB aims to run entirely in-memory with optional periodic disk snapshots. It runs on 64 bit Linux on premises, AWS, Google and Azure cloud services, and implements a horizontally scalable architecture.

Unlike traditional relational databases where data is written to disk-based log files, VoltDB applies changes in parallel to multiple machines in memory. For example, a K-Safety of two guarantees no data loss even if two machines fail, as data is committed to at least three in-memory nodes.

Transactions are submitted as Java stored procedures which can be executed asynchronously in the database, and data is automatically partitioned (sharded) across nodes in the system, although reference data can be duplicated to maximise join performance. Unusually, VoltDB also supports semi-structured data in the form of JSON data structures.

In terms of performance, a 2015 benchmark demonstrates VoltDB at almost double the processing speed of NoSQL database Cassandra, while also six times less expensive in AWS cloud processing costs.

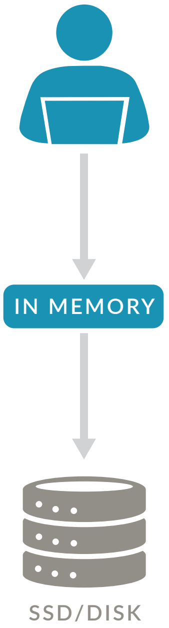
Finally, VoltDB version 6.4 passed the remarkably stringent Jepsen distributed safety tests.

To put this in context, a previous test with NoSQL database Riak demonstrated dropping 30-70% of writes, even with the strongest consistency setting. Meanwhile, Cassandra lost up to 5% of writes using lightweight transactions.

MemSQL

In common with VoltDB, MemSQL is a scale out, in-memory distributed database designed for fast data ingestion and real time analytics. It also runs on premises and the cloud, and provides automatic sharding across nodes, with queries executed in parallel on each CPU core.

MemSQL Database Architecture



In memory row store (OLTP optimized)

ID	Continent	City	Dept	Value
1	Europe	Paris	Sales	£500
2	USA	New York	Sales	£300
3	Europe	Paris	Sales	£700
4	Europe	London	Sales	£500
5	USA	New York	Sales	£200
6	Europe	London	Web	£100

Disk based column store (OLAP optimized)

ID	1	2	3	4	5	6
Continent	Europe	USA	Europe	Europe	USA	Europe
City	Paris	New York	Paris	London	New York	London
Team	Sales	Sales	Sales	Sales	Sales	Web
Value	£500	£300	£700	£500	£200	£100

While there are many similarities with VoltDB, the diagram above illustrates a key difference. MemSQL attempts to balance conflicting demands of real time transactions with data warehouse style historical data processing. To achieve this, MemSQL organises data in memory as a row store, backed by a column oriented disk store to combine real time (recent) data with historical results.

This places it firmly in the OLTP and Data Warehouse space, although both solutions target the real time data ingestion and analytics market.

Which Applications Need NewSQL Technology?

Any application which requires very high ingest rates and fast response times (average 1-2 milliseconds), but also demands transactional accuracy provided by ACID guarantees — for example, customer billing.

Typical applications include:

- **Real Time Authorisation** — For example, validating, recording and authorising mobile phone calls for analysis and billing purposes. Typically, 99.999% of database operations must complete within 50 milliseconds.

- **Real Time Fraud Detection** — Used to complete complex analytic queries to accurately determine the likelihood of fraud before the transaction is authorised.

- **Gaming Analytics** — Used to dynamically modify gaming difficulty in real time based upon ability and typical customer behaviour. The aim is to retain existing customers, and convert others from free to paying players. One client was able increase customer spend by 40% using these techniques, where speed, availability and accuracy are critical.

- **Personalised Web Adverts** — To dynamically select personalised web based adverts in real time, recording the event for billing purposes, and maintaining the outcome for subsequent analysis.

While these initially may seem like edge cases compared to the majority of OLTP applications, in a 24x7 web connected world, these present the new frontier for real time analytics, and with the advent of the Internet of Things — a massive opportunity.

Conclusion

Although Hadoop is more closely associated with Big Data, and has received huge attention of late, database technology is the cornerstone of any IT system.

Likewise, NoSQL databases appear to provide a fast, scalable alternative to the relational database, but despite the lure of licence-free open source databases, it really does seem you get what you pay for. And, as VoltDB demonstrates, it may actually be cheaper than the NoSQL alternatives in the long run.

In conclusion, if you have a web scale, OLTP and/or real time analytics requirement, the NewSQL class of databases need serious consideration.

About the Author

John Ryan is an experienced Data Warehouse architect and designer. Currently working as a Senior Solution Architect for Snowflake Computing, he regularly writes on subjects including data warehousing and database processing on his blog site at: www.analytics.today

We predict the next fifteen years will be a period of intense debate and considerable upheaval.

Dr. Michael Stonebraker (2007)