InDetail

InDetail Paper by Bloor
Author **Philip Howard**
Date **October 2014**

NuoDB Swifts Release 2.1

“

NuoDB is a very interesting product, both from a conceptual and an architectural point of view

”

Author **Philip Howard**

Executive summary

Fast facts

NuoDB is a so-called **'NewSQL'** database, developed by the company of the same name. A NewSQL database is, in effect, a database that looks relational and leverages SQL but isn't necessarily relational under the covers. In theory, NewSQL databases could be used for any task currently supported by relational databases but, in practice, developers of these products have stuck to OLTP (online transaction processing) environments in the first instance. With its initial release NuoDB was no exception but, with its latest version (2.1), the company is now starting to offer a hybrid environment that not only supports transaction processing but also provides analytic capabilities. It should be noted that the product was designed from the outset to offer general-purpose capabilities—that is, support both transaction processing and analytics—but with the latter only being formally introduced with this release.

It is briefly worth discussing why NewSQL databases have emerged at all and the simple reason is that relational databases such as Oracle or DB2 have been around for 30 or more years and over that time they have acquired a lot of features that were designed to overcome the limitations that hardware and other constraints placed upon the database at the time those features were developed. For compatibility and migration reasons these features have remained in the database and been built upon over the years, resulting in products that are burdened with superfluous and out-of-date functionality, which means that those products are over-complex, have very significant footprints, are difficult to administer and do not perform as well as they might. The approach taken by NewSQL database vendors has been to say "what do we need to provide a high-performance, low administration, transaction processing, ACID compliant, SQL database on the basis of current—as opposed to obsolete—hardware and infrastructure?"

In so far as NuoDB is concerned it distinguishes itself from other NewSQL databases, not only with its newly introduced analytic capabilities but, more specifically, because it offers a (geographically) distributed database environment that can easily scale out, as opposed to scaling up.

"With its latest version (2.1), the company is now starting to offer a hybrid environment that not only supports transaction processing but also provides analytic capabilities."

Key findings

In the opinion of Bloor Research, the following represent the key facts of which prospective users should be aware:

- NuoDB has cracked the problem of how to distribute a database across multiple physical locations (which may be geographically separated and/or in the cloud) in order to support transaction processing in a familiar SQL-based environment while providing performance, scalability and relatively low cost.
- Despite the fact that NuoDB presents a SQL face to the world it is not actually a relational database under the covers. However, you won't see that and users don't need to know about it. Under the covers the product uses a distributed object model and NuoDB is actually a multi-model database with the company planning to introduce additional interfaces during the course of 2015.
- Data may be stored in flat file systems or in key-value stores such as HDFS or Amazon S3. Thus storage is relatively low cost. This also implies, correctly, that the product is available both as in-cloud and as an on-premise solution, or in hybrid environments.
- A typical environment consists of multiple NuoDB domains within which each storage manager has its own copy of the (same) database. Processing is logically distinct from storage and is performed by transaction engines based on a memory-centric architecture that

performs at in-memory speeds if all the data is in memory. You can have as many or as few transaction engines and storage managers within a single domain as you wish, and there is no necessary relationship between the numbers of each.

- With the latest 2.1 release, NuoDB now offers HTAP (hybrid transaction/analytical processing). This is enabled by allocating transaction engines and storage managers within the cluster to do analytic tasks that do not interfere with operational processes.
- NuoDB is an append-only database that makes extensive use of multi-version concurrency control (MVCC). Corollaries to this approach mean that you can make point-in-time enquiries (for compliance purposes, say) and that the product supports schema evolution.
- The database is ACID compliant and provides support for concurrency, locking and commit processes, albeit in a somewhat different way from traditional approaches.
- The database is continuously available (that is, regardless of planned or unplanned downtime) and it is easy (and mostly automatic) to scale the environment up or down (there are facilities to quiesce database instances) on demand.

“NuoDB has a significantly different architecture from traditional row-based relational databases. However, from a developer and user’s perspective, and even from that of a database administrator, it looks like just like a conventional database of this type.”

The bottom line

NuoDB has a significantly different architecture from traditional row-based relational databases. However, from a developer and user’s perspective, and even from that of a database administrator, it looks like just like a conventional database of this type. In particular, it has a viable solution for large, geographically dispersed, web-facing, transaction-oriented and hybrid environments. Historically, conventional database products have had a problem meeting such requirements at scale and NuoDB meets this need. If you live and work in this sort of environment, or where you are migrating to cloud and hybrid cloud environments, then NuoDB is certainly worth serious consideration.

Actually, we would argue that NuoDB is worth serious consideration in other, less dispersed, transaction/hybrid processing environments as well. In particular, it should be much less expensive than a merchant database both to install initially and to scale.

The product

NuoDB was first made generally available in January 2013 following an extensive beta programme (with 9 beta releases). Version 2.0 of the product was released in October 2013 followed by several maintenance releases and 2.1 a year later. In terms of platform it is important to understand that, in effect, NuoDB is modular with storage separated from processing. Thus storage may be in the cloud (such as Amazon S3—the product supports multi-tenancy), in a data centre, or you may employ a hybrid model. The storage itself is not relational (though it looks that way) and may be implemented on a file system or any key-value store such as HDFS (Hadoop File System), which means that it will run on low cost commodity hardware. Processing modules may run in Windows, Linux or MacOS environments.

There is extensive support for different programming languages, including Java, .NET, Perl, Python, PHP and so on, as well as the aforementioned SQL, which conforms to the ANSI 92 standard plus extensions, including SQL and Java stored procedures. There is a server-side programming model so that Java code (for example) can be executed with direct access to the data. The database works with popular ORM (object relational mapping) tools such as Hibernate and PDO (PHP Data Objects) and comes with JDBC and ODBC drivers. In the latest release there is support for Java Management Extensions.

The product includes major relational database compatibility features for Oracle, SQL Server and MySQL to enable migrations from these environments to NuoDB. In the latest release there is also a new SQL-based parallel loading facility to improve performance for migrations from third party RDBMS environments.

There are several editions of the product that are available ranging from free to enterprise to cloud editions.

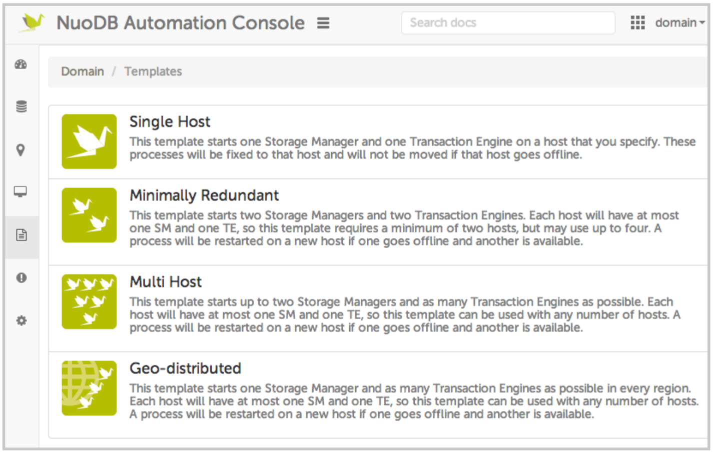
Architecture

The most important thing to understand about NuoDB is that it has a distributed architecture. This is important where you have geographical dispersed operations or cloud-based deployments where you wish to operate off a single logical database. Since the product's launch, this is where NuoDB has gained the most traction and, as a result, the company has implemented a number of optimisation and performance features to support such environments. For example, features to optimise performance even where there are poor quality (high latency) connections.

The big question is: how has NuoDB managed to implement this distributed architecture where others have failed to do so? In the case of traditional relational database vendors the simple answer is that NuoDB has been able to design its solution based on modern techniques and capabilities and without being saddled with a legacy past that it needs to support. In fact, the way that the product is structured is that it is separated into three tiers:

- An administrative tier that consists of 'brokers' that provide functions such as load balancing and to which applications and users connect (via JDBC typically). There is also a NuoDB Automation Console that provides management and monitoring capabilities. A major feature of NuoDB is the use of auto-administration. The basic idea is that you don't want to have to administer individual servers so the company has introduced rule-based capabilities so that you can define rules that automatically fire up additional resources under relevant conditions. These can then be combined into templates, as illustrated in **Figure 1**. It is worth noting that the Automation Console has been significantly enhanced in this release—not least in supporting SLA-based templates.
- A processing layer consisting of 'transaction engines' that run in memory (data is loaded on demand) and can be thought of as providing a loosely coupled distributed cache. Note that, as with the other tiers, you can have as few or as many transaction engines as you like.
- A storage layer consisting of 'storage managers'. Each storage manager within a 'domain' has its own copy of the database thus, in effect, you have multiple physical instances of the same database but these are presented to the users (the transaction engines) as a single, logical instance.

Figure 1: NuoDB 2.1 out-of-the-box automation templates



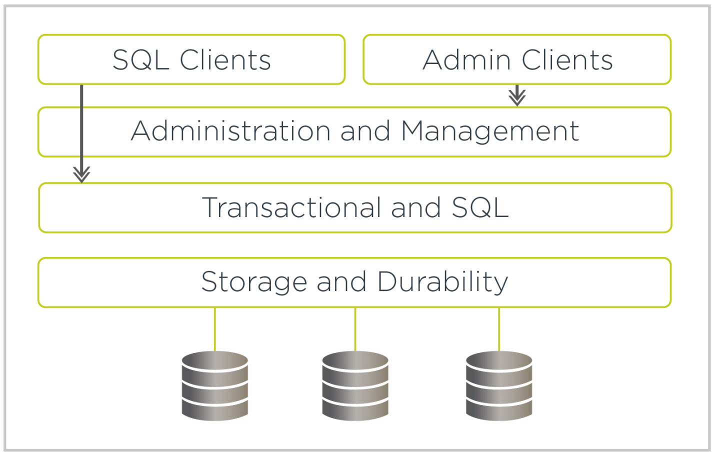
Single Host
This template starts one Storage Manager and one Transaction Engine on a host that you specify. These processes will be fixed to that host and will not be moved if that host goes offline.

Minimally Redundant
This template starts two Storage Managers and two Transaction Engines. Each host will have at most one SM and one TE, so this template requires a minimum of two hosts, but may use up to four. A process will be restarted on a new host if one goes offline and another is available.

Multi Host
This template starts up to two Storage Managers and as many Transaction Engines as possible. Each host will have at most one SM and one TE, so this template can be used with any number of hosts. A process will be restarted on a new host if one goes offline and another is available.

Geo-distributed
This template starts one Storage Manager and as many Transaction Engines as possible in every region. Each host will have at most one SM and one TE, so this template can be used with any number of hosts. A process will be restarted on a new host if one goes offline and another is available.

Figure 2: NuoDB 3-tiered distributed architecture



It is this tiered approach, which is illustrated in **Figure 2** that supports the distributed architecture that NuoDB provides. However, the concept of a three-tiered (or four, if you count the brokers as a separate tier from the applications) architecture is hardly new: so what has NuoDB done differently? The key point is it offers a scale-out architecture rather than one that is based on scaling up. By using file systems and key-value stores to store data it can be very much more cost effective in the storage layer. On the other hand, its use of in-memory technology (combined with conventional indexing—the product uses Btrees—and, in this release, an enhanced cost-based optimiser) gives it the performance that you require. Nevertheless, there will be questions about how replication works, how you ensure consistency and what happens with respect to locking in such a distributed environment: because the product architecture sounds all well and good but if it can't address these sorts of issues then it will fail. In order to address these questions we need to delve a little deeper.

“The big question is: how has NuoDB managed to implement this distributed archtecture where others have failed to do so?”

In detail

The first thing to understand is that NuoDB is an append-only database. That means that you cannot update a record. Instead you create a new version of that record although actually you only write changes to the record with the most recent changes defining the current version of that record. This technique is known as reverse deltas—with deltas you move forward from the earliest instance of the record, with reverse deltas you move backwards from the current state—the latter is obviously preferable in transaction processing environments for performance reasons. Older versions can be reconstructed by iterating backwards through the deltas, which means that you can do point-in-time queries and have support for (logical) schema evolution. So NuoDB is heavily reliant on versioning and uses MVCC (multi-version concurrency control).

The second major point is that although NuoDB looks and acts like a SQL database, it actually isn't. It would be more correct to call it a distributed object database with messaging and a SQL layer on top. In fact, in theory, you could have any interface layer on top: a MongoDB layer, for example. Everything (data, metadata, indexes, schema and so forth) in NuoDB is known as an "atom" (sort of like an object but without object oriented properties such as polymorphism and inheritance) and transaction engines know where all the atoms are, whether in-memory or on disk, and when a transaction engine needs a particular atom it will retrieve it from the optimal source (its own memory, the memory of another engine or from disk). When changes or appends are made that change is propagated to all other transaction engines using the same atom and, once 'agreed' (see next), that update will be replicated to all the storage managers.

Transaction engines are linked in a peer-to-peer fashion and there is no master engine. However, this raises questions about locking and different techniques are used depending on the situation. In general, NuoDB uses optimistic locking as well as other techniques, not least because traditional distributed lock management tends to become a bottleneck. In cases where locks need to be applied, if there is a conflict between a read and a write then the conflict is resolved based on the time at which that request was made with the reader and writer respectively seeing the version of the data that was live at that point in time. However, say that two different processes both want to take the last $1,000 out of a checking account. In this case, NuoDB makes use of it what it calls "chairmen". In effect, for each process, a particular transaction engine takes over the role of chairman and this communicates with the other transaction engines via asynchronous messages. In the case of the checking account, the chairman decides which request succeeds and which fails and notifies the other engines which process to commit and which engines should drop their new versions. They can then confirm back to the chairman that the commits or rollbacks have taken place. Storage managers go through a similar process.

The idea of chairmen is also critical for the continuous availability that NuoDB offers. This is because there is a hierarchy of chairmen, so that the next in line can take over if one transaction engine goes down or if you want to take it down for planned maintenance. Conversely, you can add new transaction engines whenever you wish: all that basically happens is that you plug it in, it provides the brokers with its security credentials and off you go. The position is slightly different with storage managers. It is common, especially in geographically dispersed implementations, that only a certain percentage of storage managers will be active at any one time (there is a built-in facility to hibernate databases when they are not necessary). When you bring a storage manager back online (a process that takes 30 milliseconds according to the company) it needs to get its instance of the database updated: it will do this automatically and in the background and then, once this is accomplished, it will notify the transaction engines that it is ready for work. In other respects, with their support for both planned and unplanned downtime, storage engines offer the same sort of continuous availability and scalability as the transaction engines.

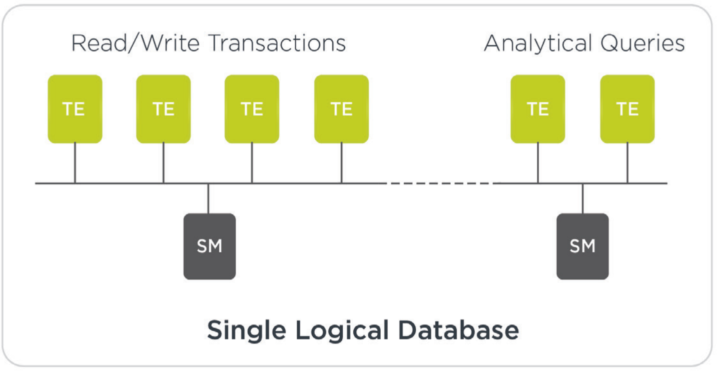
One further notable feature is that, along with hibernation, NuoDB also supports the concept of a 'bursting' database. That is, when the database has got over-busy. In this case there is a feature that enables that to be automatically moved to a server with a faster processor.

Hybrid transaction/analytical processing

There are significant advantages to being able to run analytics on the same system that you use to process transactions. In particular, you do not need ETL (extract, transform and load) or other data integration tools, there is no latency associated with moving the data from one place to another and, of course, you do not need a separate data warehouse. This does, of course, depend on the type of analytics you want to do but where you want to perform analytics or business intelligence processes against transactional data, and especially if results are required in real-time, then a hybrid transaction and analytic environment can make a lot of sense.

As far as hybrid transaction/analytical processing (HTAP) is concerned, you designate a number of your transaction engines and storage managers as specific to analytics, as is illustrated in **Figure 3**. In effect, a sub-cluster is being used for transaction processing and another sub-cluster for analytics. As far as the data being processed is concerned, there is no conflict between the data being used for analytics as opposed to that used to support transaction processing, because of NuoDB's nature as an append-only database. This means that analytics can run against the latest version of the data while not impeding updates or inserts that may be occurring within the transactional environment. It is worth commenting that the original design of NuoDB specifically allowed for the introduction of HTAP and the company is planning a number for enhancements to its analytic processing capabilities in future releases.

Figure 3: Hybrid transaction/analytical architecture showing Transaction Engines (TE) and Storage Managers (SM)



NuoDB's in-memory Transaction Engines (TEs) can be dedicated to perform either transactional or analytical functions. The Storage Managers (SMs) make the data durable and can automatically replicate the data among themselves.

The vendor

“NuoDB claims to be the most innovative company in this space: while such claims are, to a certain extent, in the eyes of the beholder, we would not disagree with this statement.”

The database veterans behind NuoDB first started to explore its concepts formally in 2008 and the company was founded, at that time named NimbusDB, in 2010. The company changed its name to the present NuoDB in 2011. The company is backed by venture capital and is based in Cambridge, MA.

Proof points quoted by the company include 1.82 million transactions per second running across 32 machines and a 72,000 database instance implementation on a single HP Moonshot system (incidentally, HP is a partner, along with Amazon, for which there is a quick start kit for scale-out provisioning on AWS). Further, NuoDB claims to be the most innovative company in this space: while such claims are, to a certain extent, in the eyes of the beholder, we would not disagree with this statement.

The company's website lists a dozen or so customers and although some of these are major organisations none of them are household names. On the other hand we do know of large well-known companies that are using NuoDB in production (NuoDB worked extensively with several of them during its beta phase and is continuing to do so) but have declined to make their names public. The company is heavily targeting ISVs and application providers, especially in the financial and telecommunications markets. Notable OEM and technology partners include Dassault Systèmes, EnterpriseWeb, Pentaho (for both BI and ETL), New Relic and Zabbix (for application performance management) and Drupal.

www.NuoDB.com

Summary

NuoDB is a very interesting product, both from a conceptual and an architectural point of view. On the other hand, the bottom line for users is whether the product can perform and scale in a cost-effective manner. There seems to be no doubt that this should be the case.

However, NuoDB faces a crowded market both in terms of established 800lb gorillas and in other new ventures. Perhaps the best thing going for NuoDB is that it has identified a sub-sector of the transaction processing market that has not been well-served hitherto and for which its product is a good fit. Its expansion into supporting hybrid query/transaction environments is also a significant step forward that other NewSQL databases cannot offer.

FURTHER INFORMATION
Further information about this subject is available from
http://www.BloorResearch.com/update/2236

About the author

PHILIP HOWARD

Research Director / Information Management

Philip started in the computer industry way back in 1973 and has variously worked as a systems analyst, programmer and salesperson, as well as in marketing and product management, for a variety of companies including GEC Marconi, GPT, Philips Data Systems, Raytheon and NCR.

After a quarter of a century of not being his own boss Philip set up his own company in 1992 and his first client was Bloor Research (then ButlerBloor), with Philip working for the company as an associate analyst. His relationship with Bloor Research has continued since that time and he is now Research Director focused on Data Management.

Data management refers to the management, movement, governance and storage of data and involves diverse technologies that include (but are not limited to) databases and data warehousing, data integration (including ETL, data migration and data federation), data quality, master data management, metadata management and log and event management. Philip also tracks spreadsheet management and complex event processing.

In addition to the numerous reports Philip has written on behalf of Bloor Research, Philip also contributes regularly to IT-Director.com and IT-Analysis.com and was previously editor of both ***"Application Development News"*** and ***"Operating System News"*** on behalf of Cambridge Market Intelligence (CMI). He has also contributed to various magazines and written a number of reports published by companies such as CMI and The Financial Times. Philip speaks regularly at conferences and other events throughout Europe and North America.

Away from work, Philip's primary leisure activities are canal boats, skiing, playing Bridge (at which he is a Life Master), dining out and walking Benji the dog.

Bloor overview

Bloor Research is one of Europe's leading IT research, analysis and consultancy organisations, and in 2014 celebrates its 25th anniversary. We explain how to bring greater Agility to corporate IT systems through the effective governance, management and leverage of Information. We have built a reputation for 'telling the right story' with independent, intelligent, well-articulated communications content and publications on all aspects of the ICT industry. We believe the objective of telling the right story is to:

- Describe the technology in context to its business value and the other systems and processes it interacts with.
- Understand how new and innovative technologies fit in with existing ICT investments.
- Look at the whole market and explain all the solutions available and how they can be more effectively evaluated.
- Filter 'noise' and make it easier to find the additional information or news that supports both investment and implementation.
- Ensure all our content is available through the most appropriate channel.

Founded in 1989, we have spent 25 years distributing research and analysis to IT user and vendor organisations throughout the world via online subscriptions, tailored research services, events and consultancy projects. We are committed to turning our knowledge into business value for you.