NewSQL - What's the Point?

Aden Kenny
School of Engineering and Computer Science
Victoria Unversity of Wellington
Wellington, New Zealand
kennyaden@ecs.vuw.ac.nz

I. Introduction

NewSQL systems are a class of relational database management systems that maintain ACID (Atomicity, Consistency, Isolation, Durability) principles, as well as have the horizontal scalability of NoSQL systems [1]. The aim of NewSQL systems is to have the scalability of NoSQL systems whilst keeping the relational model from RDBMSs (relational database management systems). One of the main selling points of NewSQL systems is that they tout their ability to scale modern online transaction processing workloads in a way that is not possible with legacy (relational, NoSQL) systems [1].

The term "NewSQL" was coined in 2011 research paper written by Aslett [2]. Therefore NewSQL DBMSs are a fairly recent concept, but the field has moved quickly since Aslett first defined them in 2011.

This essay will cover a major question that can be posed about NewSQL DBMSs - "What are the use cases of NewSQL database management systems?".

II. Literature Review

This section will review major pieces of literature in the body of literature on NewSQL systems.

A. "What we talk about when we talk about NewSQL" - Aslett

The first piece of literature to be reviewed will be Aslett's 2011 report [2] that introduced the concept of NewSQL systems. In this report Aslett introduces NewSQL systems as "shorthand for the various new scalable/high performance SQL database vendors". Aslett states that NewSQL systems in the past have been referred to as "ScalableSQL" systems, and it is stated that this name implies horizontal scalability. This is not always the case all the products that are classified as NewSQL systems in this report therefore the name "NewSQL" is used.

Aslett makes the claim that "NewSQL is not to be taken too literally: the new thing about the NewSQL vendors is the vendor, not the SQL". Aslett then introduces the idea that there is a group of vendors who are developing "new relational database products and services designed to bring the benefits of the relational model to distributed architectures, or to improve the performance of relational databases to the extent that horizontal scalability is no longer a necessity". In this category Aslett includes DBMSs such as GenieDB, NimbusDB, and ScaleDB.

Aslett also states there is a second category of NewSQL DBMSs which focus on "NewSQL-as-a-service". Examples of systems that would fall into this category include Amazon Relational Database Service and Microsoft SQL Azure. This category of DBMSs (database management systems) focuses on selling database access or usage to businesses which allows the business to not worry about replication, backups or other difficult and time consuming database management tasks.

Aslett then states that there is a clear potential for overlap with NoSQL systems. It is stated that RethinkDB, one of the DBMSs that is in the first category Aslett defined, is planning to enable "the use of its database as a schema-less store". Aslett then notes that they expect to see support for SQL queries to come to some NoSQL databases, primarily as a result of the NewSQL movement. Aslett then discusses Citrusleaf, a NoSQL DBMS that claims to support ACID transactions, and states that he is sure that it will not be last NoSQL vendor to do so.

Aslett then states that "NewSQL is not about attempting to re-define the database market using our own term, but it is useful to broadly categorize the various emerging database products at this particular point in time.". Aslett then concludes the report by stating that they have noted the "beginning of the end of NoSQL", and that they foresee the labels NoSQL and NewSQL to become useless as the two combine.

The main things to take away from Aslett's report are the definitions of NewSQL, mainly that NewSQL systems are "new scalable/high performance SQL database vendors". This isn't a very specific definition which is a flaw, but it can be forgiven, as this is, chronologically, the first piece of literature to discuss NewSQL systems.

Another thing to consider from the report is the DBMSs that Aslett identified as NewSQL systems. Aslett lists fifteen DBMSs in the "new NewSQL" category, but very few of these systems have had much uptake as of the writing of this essay in 2018.

In the second category of "NewSQL-as-a-service" DMBSs Aslett lists five systems, but the two that jump out are Amazon Relational Database Service and Microsoft SQL Azure, both of which are extremely popular and have an extremely large user base.

The final thing to take away from Aslett's report is the prediction that NoSQL will die out due to NewSQL databases. Since 2011 NoSQL DBMS market share has increased [3] rather than decreased as Aslett predicted. NewSQL systems have not gained much of a market share either. Overall Aslett's conclusions on the future of NewSQL databases seem to be

quite far off the mark.

B. *"What's Really New with NewSQL?" - Pavlo and Aslett*

"What's Really New With NewSQL?" [1] is a 2016 paper written by Pavlo and Aslett. Aslett is also the author of *"What we talk about when we talk about NewSQL"*, the first piece of literature reviewed in this essay. This paper is an extremely important in the collection of NewSQL literature as it provides a concrete definition for a NewSQL DBMS, something which is lacking in Aslett's *"What we talk about when we talk about NewSQL"*. Pavlo and Aslett also introduce categories of NewSQL systems, and some of the features that are common in NewSQL systems.

In the abstract of this paper Pavlo and Aslett ask the question "[if NewSQL's] superiority is actually true or whether it is simply marketing?" They also ask if "they [NewSQL systems] are indeed able to get better performance, ... [is] there is anything scientifically new about them that enables them to achieve these gains or is it just that hardware has advanced so much that now the bottlenecks from earlier years are no longer a problem?". Overall it can clearly be seen that this paper establishes a definition, a categorisation, and a common feature set in order to help answer the questions they put forward in the abstract.

The first point to note in this paper is the working definition of the term "NewSQL" that is introduced. Pavlo and Aslett state that NewSQL is "a class of modern relational DBMSs that have the scalability of NoSQL while still maintaining ACID" [1]. Furthermore, they state that NewSQL systems "want to achieve the same scalability of NoSQL DBMSs ... but keep the relational model. [1]" Finally, they give their final overarching definition of a NewSQL system by stating that "a NewSQL system's implementation has to use a lock-free concurrency control scheme and a shared-nothing distributed architecture. [1]"

The abstract therefore introduced concrete definition of what a NewSQL system actually is can then be used to categorise and discuss features common to most, if not all NewSQL systems.

In the next section Pavlo and Aslett go over the history DBMSs, and this section also introduces some of the issues with past DBMSs that led to the creation of NewSQL systems. They introduce those problems as the difficulty of scaling RDBMS systems, and the fact that many applications are unable to use NoSQL systems as they "cannot give up strong transactional and consistency requirements. [1]" This meant that an ideal solution to these problems would be a DBMS that allowed easy scaling and supported ACID. This description is basically the same as a definition stated by Pavlo and Aslett in this paper. From this section we can clearly see the motivations for the development of NewSQL systems.

The next section introduces three different categories of NewSQL systems. The three categories are

- New Architecture
- Sharding Middleware
- Database-as-a-Service

1) New Architecture Systems: New Architecture NewSQL systems are stated to be DBMSs that are built from scratch to fit unique specifications, which means they lack the of the architectural baggage or constraints of legacy systems. All NewSQL systems that fit into this category are "based on distributed architectures that operate on shared-nothing resources, a clear departure from RDBMS systems, but it is noted that like RDBMS systems, these "New Architecture" systems support ACID" [1].

Pavlo and Aslett state that all of the DBMSs in this category are "based on distributed architectures that operate on shared-nothing resources [1]". This means that these systems can easily be scaled horizontally. Due to the fact that they are distributed systems, they all contain components to support multi-node concurrency control, and fault tolerance, specifically through replication.

Pavlo and Aslett then state that the advantage of using a New Architecture NewSQL DBMS is that "it is built for distributed execution in that all parts of the system can be optimized for multi-node environments." [1] A consequence of this is that that New Architecture systems tend to have better query and overall performance when contrasted with NoSQL DBMSs or other categories of NewSQL DBMSs.

A optimisation that is common in New Architecture NewSQL DBMSs is the fact that queries and data can be sent between nodes rather than having to be routed through a central middleman or master node [1]. In other words, the DBMS can "send the query to the data" rather than "bringing the data to the query". This means that in the case of queries that examine or require large amounts of data, time is not spent sending the data between nodes via a middleman. The query can instead be sent to the node that contains the data. The query will almost certainly be smaller than the data that would be required to fulfil the query. This means that there will be a performance increase as the network latency impact about query resolution will be minimised when compared to more traditional "bring the data to the query" approach. Network traffic will also be reduced due to the fact that queries are far more likely to be smaller than the data.

2) Sharding Middleware Systems: The second category of systems that Pavlo and Aslett introduce are Sharding Middleware Systems. These are systems that build upon the sharding middleware that was developed in the early 2000s by companies such as eBay, Google, and Facebook. This class of systems splits a database into multiple shards that are stored across a cluster of single-node DBMS instances.

Pavlo and Aslett do emphasise that Sharding Middleware NewSQL DBMSs (new sharding) are different from past sharding middleware DBMSs (old sharding) [1]. They offer three main differences. Firstly they state that each node in a new sharding system has to run the same the DBMS, which is in contrast to older technologies where this was not always required. A second difference is that each node in a new sharding system contains only a "portion of the overall database", in other words they operate on shared-nothing resources. This is in contrast to some old sharding systems

where some of the DBMSs were not shared-nothing, data could be shared across nodes. The third and final difference that Pavlo and Aslett point out is that unlike some older systems, each node of a new sharding system is not meant to be accessed and updated independently [1]. This is because new sharding systems have a centralised master node.

All new sharding systems have a "centralized middleware component" [1]. This component routes queries, coordinates transactions, as well as manages data placement, replication, and partitioning across the nodes. This means that all the "smart work" of a new sharding DBMS is done by the middleware component and individual nodes do not have to be concerned with this work. Each node in the DBMS will typically have a "communication layer" on top of it that is designed to communicate with the middleware component.

Pavlo and Aslett state that the key advantage of new sharding systems is that they are often a drop-in replacement for an application that is already using an existing single-node DBMS [1]. This means that developers or database administrators only need to make minimal changes to their application to use a new sharding system.

Pavlo and Aslett mention that this centralised middleware solution does result in a major issue [1]. This issue is that a new sharding system is the middleware itself, but the specific nodes of the system still have to run a traditional DBMS on each node. This means that some of the benefits of NewSQL DBMSs are not available to new sharding systems as they still partially rely on traditional NoSQL or RDBMS systems.

3) Online Service Systems: Pavlo and Aslett's third and final category of NewSQL DBMSs are Online Services systems. This category consists of cloud computing providers that offer NewSQL database-as-a-service (DBaaS) products. These services mean that organizations do not have to maintain the DBMS on either their own private hardware or on a cloud-hosted virtual machine (VM).

This means that the DBaaS provider is responsible for maintaining the physical configuration of the database, including system tuning, replication, and backups. This takes a significant amount of work off the shoulders of the customer of the provider. This means that NewSQL systems can be accessible to smaller companies that do not have the expertise or willpower to host their own database solution.

The customer is generally provided with an API or dashboard to control the database system [1]. This means that the customer can still access and use their database while being able to ignore the more technical maintenance and set-up tasks on their database.

Pavlo and Aslett state that some systems that had previously been considered in Aslett's 2011 paper "*What we talk about when we talk about NewSQL*" [2] to be Online Services NewSQL systems, are no longer considered to be NewSQL systems. These are systems that still use the same underlying disk-orientated architecture from the 1970s. Pavlo and Aslett give the examples of Microsoft Azure SQL, and Heroku, as DBMSs that are no longer considered NewSQL systems due to the previously stated reason [1].

Pavlo and Aslett state that they only regard DBaaS products that are based on a new architecture as NewSQL systems. The also state that "DBaaS has the advantage that it can distribute a database across different providers in the same geographical region to avoid downtimes due to service outages." [1] This means that Online Service system NewSQL DBMSs have an advantage in terms of geographic fault tolerance.

4) NewSQL Features: Pavlo and Aslett then discuss the features of NewSQL systems, and they attempt to identify what is novel in NewSQL systems. These features mainly revolve around the storage structures, query processing, and fault tolerance of NewSQL DBMSs.

The first feature discussed is main memory storage. Pavlo and Aslett state that currently all the major (non-NewSQL) DBMSs use a disk-oriented storage architecture based on the original DBMSs from the 1970s, where the data is stored on an HDD or SSD [1]. Since reading and writing to these devices is slow, DBMSs use memory to cache blocks read and to buffer from transactions. This was necessary because historically memory was much more expensive and had a limited capacity compared to disks. The point has been reached, where capacities and prices are such that it is affordable to store all but the largest databases in memory. The benefit of this approach is that the DBMS no longer has to assume that a transaction could access data at any time that is not in memory and will have to stall. This means faster querying times for DBMSs that store the entirety of their data in RAM rather than on a HDD or SSD. They do state that a new feature of main memory NewSQL systems, is the ability to evict a subset of the database out to persistent storage to reduce its memory footprint. This allows the DBMS to support databases that are larger than the amount of memory available without having to switch back to a disk-oriented architecture.

The second feature introduced by Pavlo and Aslett is Sharding. This the fact that almost all of the distributed NewSQL DBMSs scale out is to split a database up into disjoint subsets, generally called shards but also sometimes known as partitioning. They state that distributed transaction processing on partitioned databases is not a new idea. The DBMS assigns each tuple to a fragment based on the values of its attributes using partitioning. This means that a single piece of data will only be on a single node, which is means that most transactions need to access data at a single partition.

The third major feature that is discussed by Pavlo and Aslett is Concurrency Control. Pavlo and Aslett state that concurrency control is the most important implementation detail of a DBMS as it affects almost all aspects of the system. NewSQL DBMSs concurrency control system provides the atomicity and isolation for ACID of the systems. Pavlo and Aslett state that concurrency control permits end-users to access a database in a multi-program fashion while preserving the illusion that each of them is executing their transaction alone.

Overall Pavlo and Aslett state that they "we find that there is nothing significantly new about the core concurrency control schemes in NewSQL systems." [1]

The fourth major feature that is discussed by Pavlo and Aslett is Secondary Indexing. They describe a secondary index as containing a subset of attributes from a table that are different than its primary key(s). They state that this allows the DBMS to support fast queries beyond primary key or partitioning key look-ups. This helps to improve the overall performance of the DBMS.

They state that implementing this in a non-partitioned DBMS is fairly trivial due to the fact that the entire database is located on a single node. Secondary indexing can be a challenge in a distributed DBMS. This is due to the fact that the indices cannot always be partitioned in the same manner as with the rest of the database, in other words, the indexing may not be consistent. Pavlo and Aslett state that "all of the NewSQL systems based on new architectures are decentralized and use partitioned secondary indexes." This means that each node stores a portion of the index, rather than each node having a complete copy of it.

The fifth feature identified by Pavlo and Aslett is Replication. Replication is obviously not a unique feature of NewSQL databases, but it is stated that all NewSQL databases have replication features. Replication is extremely important as data loss can be extremely costly, especially in a competitive business environment. Pavlo and Aslett state that "in a strongly consistent DBMS, a transaction's writes must be acknowledged and installed at all replicas before that transaction is considered committed." [1] They then make the claim that the advantage of this approach is that replicas can serve read only queries and still be consistent. Furthermore it is stated that it also means that when a replica fails, there are no lost updates because all the other nodes are synchronized.

Pavlo and Aslett do mention a downside to this model by stating that "maintaining this synchronization requires the DBMS to use a two-phase commit to ensure that all replicas agree with the outcome of a transaction" [1]. This can add a large amount of overhead to the database system and can reduce the overall performance of the DBMS. This is a trade-off that must be taken into account when considering the use of a NewSQL system, especially compared to traditional non-partitioned which will not have this issue. As a closing statement on replication, Pavlo and Aslett state that all of the NewSQL systems that they are aware of support strongly consistent replication." [1]

The sixth and final feature that Pavlo and Aslett discuss is Crash Recovery. This provides fault tolerance by being able to recover from a crash. This differs from traditional DBMSs where the main concern of fault tolerance is to ensure that no updates are lost. Pavlo and Aslett state that newer DBMSs must also minimise downtime, as well as providing the previously mentioned crash recovery functionality. This is due to the fact that modern web applications are expected to be online all the time as site outages are costly, both financially and in user numbers.

New Architecture NewSQL DBMSs have a different approach to crash recovery when compared to Sharding Middleware and Online Service NewSQL DBMSs. New Architecture systems generally rely on recovering the last checkpoint node from local storage and then pulling logs it missed from other nodes that did not crash. Pavlo and Aslett note that as long as the node can process the log faster than new updates are appended to it, the node will eventually converge to the same state as the other replica nodes.

In contrast Sharding Middleware and Online Service NewSQL systems generally rely on the on built-in mechanisms of their underlying single-node DBMSs for crash recovery. However it is noted that additional infrastructure for leader election and other such management capabilities can be added.

5) Pavlo and Aslett's Conclusions on NewSQL: Pavlo and Aslett then discuss the current status and future of NewSQL DBMSs (note that the paper was published in 2016). They state that NewSQL systems are facing an uphill battle to gain market share, and give examples of several NewSQL focused companies that have either folded or pivoted focus to other areas. They then note that NewSQL systems still lack market share, especially when compared to Aslett's forecast from 2012 [4]. This is especially true when compared to NoSQL systems, where the growth is primarily considered to be developer driven [4].

Pavlo and Aslett suggest the reason for the slow uptake of NewSQL systems is that NewSQL DBMSs are designed to support the workloads that are mostly found in enterprise applications. This means that decisions regarding database choices for these enterprise applications are likely to be more conservative than for new Web application workloads. This means that smaller business are going to be less likely to pick up NewSQL systems, as they may lack some new features. Furthermore, it is quite likely that the conservative design of the system could make the system as a whole harder to use, which would also turn off potential users.

Pavlo and Aslett state that they have found that NewSQL DBMSs are not being used for new development, when NewSQL DBMSs are used, they are generally used to complement or replace existing RDBMSs [1]. They have also found that NoSQL DBMSs are the main class of systems being used for new application development. This is obviously an issue for NewSQL systems, as to gain high market share, NewSQL DBMSs need to be being used for new development.

In the conclusion of the paper Pavlo and Aslett state that NewSQL is not a radical departure from previous systems. They state that they are the next chapter in the continuous development of database technologies just as RDBMSs were and NoSQL DBMSs were. Many of the technologies that are used in NewSQL systems have existed in previous DBMSs, both from academia and from industry. They do note that many of the features were only implemented one-at-a-time in a single system and never all together.

Pavlo and Aslett then state that NewSQL systems are by-products of a new era where distributed computing resources are plentiful and affordable, but at the same time the demands of applications is much greater. They believe there will be a convergence of features from RDBMSs, data warehouses, NoSQL DBMSs, and NewSQL DBMSs. DBMSs that spawn

from this convergence will support the relational data model, and will support SQL. They claim that when this happens labels will become meaningless as DBMSs will become very similar to each other [1].

III. NewSQL use cases

The main question that will be posed in this essay is "what are the use cases for NewSQL systems?", and this section will attempt to answer this question.

In order to understand where NewSQL DBMSs could best be used, we need to recall the advantages and features of NewSQL systems. They are designed to overcome the scaling issues of RDBMSs and the consistency issues of NoSQL DBMSs. This means that any use case of NewSQL systems must require a relational data model, where consistency is needed, and must have a significant amount of query volume (meaning scaling is important).

It is trivial to imagine a situation where the having relational data model, isolation, and consistency are important, if not vital. The real world is full of these situations, and examples include banking, finance, and many more. These situations generally value These are prime use cases for RDMBSs.

It is also simple to think of situations where the query volume is sufficient to need horizontal scaling. Examples of these include websites with extremely large user bases, social networks, and systems where high availability is needed. These are good use cases for NoSQL DBMSs.

In order to find a use case for NewSQL DBMSs we need to find the intersection of the set of use cases where the relational model is needed, and the set where horizontal scaling is required. Therefore we need to find a use case where there is a large amount of query volume and a relational model is required.

One thing that must be considered is the potential advantage of being schema-less that NoSQL DBMSs present. This can be extremely useful in some situations such social networks, where the data may not have an underlying natural structure, and where different users may have different types of data stored when compared with other users.

Therefore we need to add yet another condition to find use cases for NewSQL DBMSs. This use case must:

- Require the relational model - or at least not be so negatively impacted by using the relational model to become unusable
- Require the ability, and have the need to horizontally scale
- Not be so negatively impacted by having a schema to become unusable

One place where NewSQL DBMSs could feasibly be used is in stock markets [5]. A central database that handles all trading and price information would almost certainly require strong consistency and isolation. This is because when making trades, it is vital that any database constraints are not violated, and that any transactions see the effects of transactions in the past [6]. Transaction integrity would also be extremely important in this situation, especially in regards to the ACID concept of isolation.

A centralised stock market database could be required to serve a large number of users/customers [5]. The transaction volume could pass the volume that a standard RDBMS could handle through traditional vertical scaling. The natural solution would be to use a NoSQL database so horizontal scaling could be taken advantage of. Unfortunately as per the CAP theorem, it is not possible to have consistency, availability and partition tolerance. This means that for our hypothetical stock market database, a NoSQL solution would be impossible.

This means that a centralised stock market database is at intersection of the two sets of previous discussed requirements, and this makes it an ideal candidate for a NewSQL solution. This is because it would require the relational data model that a NewSQL DBMS offers, as well as benefiting from the ability to horizontally scale through sharding that a NewSQL provides. A centralised stock market database is probably the ideal candidate for a NewSQL DBMS.

Another possible use case of a NewSQL DBMS is for a bank back end. A bank system obviously needs a relational data model as ACID principles are core to any system that can successfully handle banking scenarios. It can fairly safely be assumed that all banks use a RDBMS solution (most often Oracle) [7]. This shows that horizontal scaling is not especially important or needed in a traditional banking setting, but it is possible that could change in the future. If a bank has so many transactions that a traditional RDBMS cannot handle the transaction volume the bank would need to scale the system, and when they reach the limit of vertical scaling, horizontal scaling will be required. This could be a use case for a NewSQL system as it could provide the relational data model as well as the ability to horizontally scale.

It also possible that any situation that is served by an RDBMS could potentially be served by a NewSQL DBMS. This is because a NewSQL system generally has all the features of an RDBMS with added features such as sharding added on top. These added features bring extra complexity which does present a downside to using a NewSQL DBMS. Therefore we can conclude that a NewSQL database could probably be used in all situations where an RDBMS is used, it is probably unwise to use a NewSQL DBMS in this situation unless the horizontal scalability (or any of the other features of NewSQL) is actually needed, primarily due to the added complexity, cost, and relative lack of large scale industry use of NewSQL DBMSs.

IV. Conclusion

This essay reviewed major pieces of literature in the NewSQL space, and drew data and conclusions from them. A question - "What are the use cases of NewSQL database management systems?" was posed, and it was partially answered by taking information from the reviewed literature and extrapolating this said information.

Three use cases for NewSQL DBMSs were proposed by this essay:

- Stock Exchange Database
- Large Scale Banking Database
- Any Use Case of an RDBMS

The first use case was a centralised stock exchange database. This could be a use case for NewSQL system as it requires the relational data model and ACID principles as well as potentially benefiting from the horizontal scalability of a NewSQL DBMS. This use case would also probably benefit from a conservative design process.

The second proposed use case was for banking databases with many users. This could be a major use case for NewSQL systems as they require the relational data model, and it is plausible that the transaction volume will be large enough that the horizontal scaling of a NewSQL system could be beneficial.

The third and final proposed use case was any system where an RDBMS is currently used. This suggest is less serious than the others, but a NewSQL DBMS can generally do what an RDBMS does, but more. NewSQL systems are not a silver bullet though and do present some downsides. This means that it is possible to use a NewSQL DBMS in this use case, but it is probably not always the best choice, and RDBMS should still be used.

These use cases are by no means exhaustive, there are probably many more scenarios that exist where a NewSQL system could be used, and could present advantages. This essay was not meant to provide an exhaustive list, it was meant to provide a few use cases for NewSQL systems.

Overall it can be seen that NewSQL systems are fairly niche in their current state. While there are potential uses for them, RDBMSs and NoSQL DBMSs are probably still the better choice in most situations. This may change as requirements change over time, or technological advances mean that NewSQL databases have a feature that means they present more advantages than disadvantages when compared with RDBMSs and NoSQL DBMSs.

Overall we can conclude that there is a point to NewSQL databases, and they do have a place in the modern database industry. However it must be remembered and taken into consideration that NewSQL DBMSs are not a silver bullet, they will not solve all problems. They are simply another class of database systems that have their own strengths and weaknesses.

REFERENCES

[1] A. Pavlo and M. Aslett, "What's Really New with NewSQL?," *ACM Sigmod Record*, vol. 45, no. 2, pp. 45–55, 2016.

[2] "What we talk about when we talk about NewSQL." https://blogs.the451group.com/information_management/2011/04/06/what-we-talk-about-when-we-talk-about-newsql/, 2011. [Online; accessed 14-October-2018].

[3] "The State of NoSQL Databases in 2018: Is NoSQL Usage No Longer Cool?." https://www.itprotoday.com/cloud-data-center/state-nosql-databases-2018-nosql-usage-no-longer-cool, 2018. [Online; accessed 15-October-2018].

[4] "451 Research delivers market sizing estimates for NoSQL, NewSQL and MySQL ecosystem." https://blogs.the451group.com/information_management/2012/05/22/mysql-nosql-newsql/, 2012. [Online; accessed 16-October-2018].

[5] "U.S. Stocks Database." https://www.globalfinancialdata.com/GFD/Article/us-stocks-database, 2017. [Online; accessed 16-October-2018].

[6] "History of the American and NASDAQ Stock Exchanges." https://www.loc.gov/rr/business/amex/amex.html, 2015. [Online; accessed 16-October-2018].

[7] "Is there a large bank using Mysql or PostgreSQL?." https://softwareengineering.stackexchange.com/questions/83363/is-there-a-large-bank-using-mysql-or-postgresql, 2011. [Online; accessed 16-October-2018].