datAvail
DATABASE SERVICES

MySQL Architecture Options:

Which Database Structure is Ideal for You?

By Charleste King

ABSTRACT:

A database is a collection of information organized to be easily accessed, managed and updated. Databases have evolved dramatically over time. We have moved from scrolls and papyruses found in the Library of Alexandria to virtual file cabinets in the digital realm.

When we have a database, we want all our data to be safely stored and easily retrieved. The data should be available using a systematic and repeatable method, and the retrieval process should be as fast and as reliable as possible to benefit our end users.

Underpinning and enabling this is the database architecture. This consists of the data storage, replication and, for high availability, the architecture must include a failover architecture as well as some sort of data redundancy. With these in place, the data is safely stored and can be retrieved easily and quickly in a systematic, repeatable fashion.

Which type of database architecture will enable your organization to fully access, manage, and update your data resources through MySQL? This whitepaper discusses storage options; cluster solutions, including Galera and MySQL Cluster; as well as redundancy, speed, failover, and other parameters.

datAvail

DATABASE SERVICES

MySQL Architecture Options: Which Database Structure is Ideal for You?

content

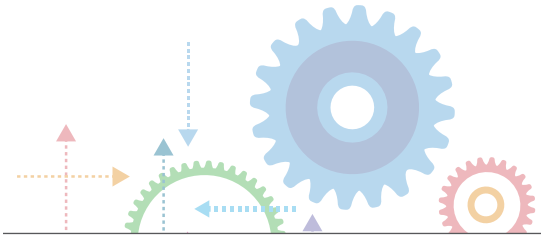
MySQL Architecture Options: Which Database Structure Is Ideal for You?

The open-source software, MySQL, is the dominant platform for many enterprise databases. What are the aspects and parameters of the database that allow each of these key features to be enabled? What is the best possible database structure available? We will discuss each of these so you can evaluate and create a MySQL database architecture that meets your organization's unique needs.

Database Architecture

When we talk about data being "safely stored," this means it is protected. It does not become corrupt, and it is not changed without our instructions. The term "protected" is very wide and is often vague. It can mean ensuring that prying eyes can't see it. It can also mean data is secure and cannot be changed readily. We want our data to remain intact.

We also want to find and retrieve our data easily. We should be able to cross-reference our data, but data must retain its integrity. We must have a way to systematically store and find our data in a predictable, replicable fashion. That is the primary goal of database architecture, regardless of the server or application being used. And it starts with data storage.



Data Storage With MySQL

Data storage is a key aspect of any database architecture. Each data storage method has its pros and cons. In MySQL, the term "storage engines" is used. MySQL uses two different formats: transactional, which means the data can be rolled back when/if a transaction fails, and non-transactional, in which the data is altered as the statements.

Although there have been several different ways in which databases traditionally store data -- within flat files, in spreadsheets, or relational databases -- there are many different options and, with those, different concerns. Choices you make regarding data storage will impact subsequent choices in the software used to organize the data.

Today, we need to think about:

- Storage size
- Memory
- Indexes
- Redundancy

We need to look at each of these within the context of the numerous different storage types available to MySQL users. The most commonly used MySQL storage engines are MyISAM, InnoDB, memory, NDB, CSV, and XtraDB. Of these, XtraDB is commonly used with Percona and with Galera. MyISAM, InnoDB, Memory, NDB, and CSV are used with MySQL.

There are many other storage options available. We aren't going to address all of these. Some -- like Blackhole -- may be very handy for a specific one-off task. Blackhole is useful, for example, for upgrading from a MySQL 5.0 instance to 5.6. If you want to use replication for minimal downtime, the Blackhole engine enables you to write only the binary logs and complete the data chain replication. This means not having to pay for all the storage for 5.1 or 5.5 instances. It enables you to have and use replication affordably. That's just one example of a specialized solution. Let's look at some of the MySQL storage options briefly.

MyISAM is a staple storage engine for MySQL. It is a non-transactional or atomic storage engine. When a statement is submitted, it is immediately processed, making this option very fast for data retrieval.

There are several caveats associated with its use. The most onerous one is, the problems that occur with row locking versus table locking. Because table locking is initiated by MyISAM when someone is backing up, inserting, updating, or deleting data within a table, if another user needs to do something to that table, they must wait their turn. In a very high-transaction environment MyISAM becomes a bottleneck.

Additionally, bottlenecks occur when backups are needed. Given the number of other storage engine options available today, the primary reason you might wish to use MyISAM is for delayed inserts -- something no other storage engine provides.

At Datavail, we often recommend InnoDB, the storage engine that now serves as the default storage engine for MySQL. It has the benefit of being both transactional and ACID-compliant. It uses row-level locking. In versions prior to 5.7, it uses gap-locking, which allows users to make a hot backup without losing state.

InnoDB has multi-version concurrency control, which means that when it's modifying a row and another function is coming in, it is able to mark the old row and archive it. It also writes and removes the old data. This eliminates any "dirty reads" from occurring. The primary caveat: InnoDB doesn't work well for fast access.

CSV is a handy storage engine to use. It is basically a text file with Comma-Separated Values (CSV). This classic data storage format is always available. CSV data files can be opened in Excel or by a text editor. You can write a little MySQL script that will be easily able to access the data using no additional libraries. This can be a very handy way to allow programs to access your data if you don't want to use a specific relational database management system or application.

Another option is to use the memory storage engine, which is very, very fast because the data and indexes are stored in memory. The problem is, the data is not stored anywhere physically. If there is a power loss -- whether from a restart or catastrophic failure -- the data is not retained. This option could be handy for holding session data, as having data in memory makes it readily accessible.

NDB is a storage engine used exclusively with MySQL Cluster. It stores data similarly to the way in which data nodes do. Here, the data nodes are managed by the NDB administrative node.

Comparing Database Storage Engines

When we compare these different storage engines, we want to look at several different metrics. In particular, we want to compare the maximum storage limit, transaction types, foreign keys, multi-version concurrency control, data compression, scalability, whether the engine is ACID-compliant, and allows parallel writes. You can see some of these in the chart provided.

Storage Engine	Innodb	NDB	MyISAM
Max Storage Limit	48 TB	3 TB	Disk and filesize limited
Transactional Types	All	READ_COMMITTED	Atomic
Foreign Keys	Yes	7.3+	Ignored
MVCC	Yes	No	No
Data Compression	Yes	No	No
Scalability	Application-level sharding	Yes	Application-level sharding
ACID Compliant	Yes	Yes	No
Parallel Writes	No	Yes	No

You will note that only three database engines are compared in the accompanying chart. We did not compare CSV or memory because these are not typically used. XtraDB has its own parameters, the specifics of which were not available from Percona.

In looking at the storage limit, for example, InnoDB has a maximum of 48 terabytes; NDB is limited to a maximum of 3 TB; and MyISAM is limited by the disk and file size. Additionally, InnoDB allows data compression while neither NDB nor MyISAM do.

You will need to determine those metrics or features that are most important to your organization and weigh those carefully before making a storage engine selection. Having ACID-compliant data storage, for example, may be a priority. This will narrow your possible choices to InnoDB or NDB. Issues such as the need to comply with federal or industry data retention policies may also guide your selection of a database storage solution.

MySQL Cluster

There are two different cluster solutions available to MySQL users: Galera and MySQL Cluster. Both have some very nice features, but they also have issues.

MySQL Cluster offers synchronous replication and is highly scalable. It operates in real-time and is ACID-compliant. It claims to offer 99.999% availability. It has a distributed, multi-master architecture with no single point of failure. It is also able to scale horizontally, and it has both SQL and NoSQL interfaces. MySQL Cluster uses the NDB storage engine.

Having a multi-master architecture with no single point of failure allows users to structure the cluster with a single data node, one management node, and one API node from which MySQL is run. Ideally, you would have at least two, if not three, data nodes, as well as multiple management nodes and as many of the API nodes as you would like.

Typically, the data node would be installed with an API node on each instance. Users typically have one management node, but that's risky. That would be your single point of failure. You will need at least two management nodes if you wanted to have no single point of failure.

What hardware does MySQL Cluster require? If you want to have two data nodes and two management nodes, four servers will be needed to implement this. Some database professionals prefer to have multiple redundancy, which would require at least five servers. If you want to keep your API nodes separate from your data and your management nodes, each API node will require an additional server.

The first thing to think about with MySQL Cluster is latency. When a transaction is submitted, the cluster needs to have confirmed that all the components within the system "know" the transaction is completed. The lag time associated with this verification and confirmation creates some inherent latency.

At Datavail, we have seen latency issues arise, some of which have resulted in a cascade of increasingly problematic issues throughout the system. In one situation, I was examining a bottleneck and needed to look in the error log. The process of looking at the nodes resulted in a series of events causing corruption in both nodes as well as an unsuccessful repair requiring the restoration of data from backup files.

Galera Cluster

Galera Cluster is a second option. It offers synchronous replication, has an active-active multi-master topology, and reads and writes to any cluster node. Because it has automatic membership control, any failed nodes are automatically dropped from the cluster. It also offers automatic node joining and true parallel replication on the row level. Among its many other attributes, it has a native MySQL look and feel, and you can use Galera Cluster with either the InnoDB or XtraDB storage engines.

How does this differ from MySQL Cluster? Galera Cluster uses no management node. In a cluster, each of the different nodes communicates with one another. This can result in latency, particularly if you have many nodes in your system and if you are sending a large transaction.

That entire transaction has to be completed on every single node, and each of these nodes must confirm that the transaction has been completed before the data is considered committed. These attributes make Galera Cluster very transactional and ACID-compliant, which is great, save for the inherent latency.

You can also use it, for example, to write to a single node and propagate that data to other nodes as a backup strategy. Or you could do all your writes to one node and do all your reads to another node. You can, in other words, structure the cluster to operate in a way that best meets your organization's needs.

Data Replication Strategies

Another aspect of your data architecture is replication. The standard types of replication strategies are:

- Master/Slave (or Multi-Slave)
- Master/Master
- Ring Replication

Ring replication is a complicated structure. It does exist as an option for replication; however, because of all the inherent problems associated with its ongoing use -- especially when maintenance or upgrades are needed -- Datavail recommends using one of the other replication methods.

1. Master/Slave Replication

Master/Slave is perhaps the most common means of ensuring redundancy. Within Master/Slave, the readability of the slave has no impact on the master and backups can also be made without any effect on the master. Downtime can be averted when the slave is taken offline for maintenance, and the slave can be easily re-synced.

That it is an asynchronous type of replication becomes a sticking point for some people, but there is neither any automatic failover in a Master/Slave environment, nor is this truly possible in any type of replication. There is always some downtime and possible data loss if the master fails.

The Master/Slave architecture is very straightforward. You can have a single master database and one or more copies or "slave" databases. You can choose the number of slaves based on factors such as your need to do read write splits. Ideally, you want to have your replication so that your slave is not lagging very far behind the master.

The greatest problem that occurs from slave lag is when you are first starting the system. If you have your slave, and if you can keep your binary logs, then you actually have all of your data after it was written. This is a reliable way to ensure you have no data loss.

2. Master/Master Replication

Master/Master is a little different. Each master serves as a master to the other master. In other words, A is a master of B; B is a master of A. You can have additional slaves off of either one of those masters. The writes can be distributed and split as you wish using a load balancer or a tool such as MMM or HAProxy. Using a load balancer can help keep the system running automatically in the event of failover compared to the steps needed to be instigated by a system administrator in a Master/Slave structure to maintain a state.

The caveat associated with using Master/Master is that it requires a stepped primary key for completing those automatic increments. You can institute multi-master or ring replication, but you will need to have the correct offsets for those and you will need more servers to execute this type of architecture. It is asynchronous and more complex than a Master/Slave to deploy. Once this replication structure is active and working, it can be a simple, good option.

Increasing Database Availability

To ensure your system has high availability, you need to be certain the system has failover ability. Data redundancy is also important in this context as having data redundancy prevents data loss. For failover to be successful, the system needs to be back up and running as quickly as possible with little to no interruption.

Different architectures have different requirements for failover to be effective. Here, we will look at standalone instances as well as Master/Slave, Master/Master, MySQL Cluster, and Galera Cluster.

In a standalone data architecture, a full, consistent backup is required; for failover, you need binary logs for point-in-time recovery from your last backup. Although there is no slave component to this architecture, you need binary logs to record transactions.

Restoration is going to be time-consuming, particularly if a large dataset is involved. You must have sufficient hardware available for restoration. If there is no binary log, there is a high probability of data loss. Even with a binary log, there remains a good probability of some data loss. In either event, the downtime for your organization could be considerable.

With a Master/Slave configuration, a hot standby is used, which means there's no immediate restore required. There is, however, the possibility for data loss if the slave is not in sync with the master. You also need to have some means for promoting the slave to a master, such as a load balancer as previously mentioned.

There is a caveat to the use of Master/Slave. You have to re-synch both the newly promoted master and the slave.

Any issues that occur around the promotion -- which could be an application connection change or a change of IP address -- must be resolved to ensure the application continues working properly after these changes occur.

The Master/Master configuration also uses a hot standby. No immediate restore is required nor is promotion needed. As with Master/Slave, some kind of proxy load balancer or an IP change will be needed for the changes to be effected properly. The server that is down will need to be fixed as quickly as possible to maintain redundancy and keep all the tools associated with the database in place.

MySQL Cluster offers an automatic internal failover. Replicants are manual failovers. The nature of the MySQL Cluster means that -- although there is a management node, multiple data nodes and multiple API nodes -- a single point of failure exists: the management node.

Galera Cluster, as discussed previously, has no management node. Failover is automatic, accomplished internally. Replicants also have a manual failover. If the application is going to a particular IP address in lieu of a pool, then a floating IP or a manual failover will be required for an application.

Many people simply like to point their applications to a single node. If using a load balancer or a virtual or floating IP address, that application can be moved to a different server with little to no problem.

Data Redundancy Configurations

Let's look at how data redundancy works with each architecture type. Redundancy must be properly executed so you or another database administrator in the organization can restore data in the event of some type of failure.

In a standalone database architecture, there is no real data redundancy. Any existing redundancy depends on any backups and/or binary logs that exist. With this approach, there is a very high probability of data loss and database failure. In the event of any type of failure, the entire system would have to be restored. Hopefully, you have those binary log shadows somewhere so that if there's a catastrophic failure, you can at least restore the data to that point in time.

Within a master-slave database architecture, the slave is a hot replica of the master.

This means it offers a hot backup and is on standby. There remains a possibility of data loss if there is slave lag. There can also be binary lag corruption, which can cause data drift or loss.

If any of these occur, you cannot replicate the corrupted data. Improper binary logging and the use of triggers, which may create issues when there are unsafe replication statements used, can also cause data drift.

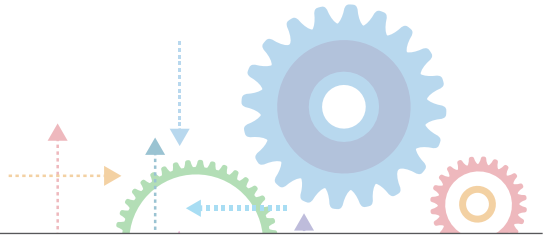
In the master-master architecture, the slave is a hot replica of the master. You can have data redundancy issues similar to those you experience with the master-slave database architecture as they are actually both masters and both slaves.
Data redundancy within MySQL Cluster relies on mirrored data. There are several types of potential challenges. Individual data nodes, for example, can fail when transactions are aborted. You must be aware that transactional applications are expected to handle any transaction failures.

How a specific application might protect your data in the event of failure differs for each of these. If, for example, you want to put triggers in the database and you're using some kind master-slave or master-master redundancy, you must be certain your application is using replication-safe statements in MySQL Cluster.

The application needs to be able to handle those transactions that have failed. We have seen many instances of data loss when adding nodes, for example.

Galera Cluster works by not considering a transaction complete until all of its members agree. Large transactions are not advised. If you're sending something very large off to four or more nodes, this would be classified as a large transaction.

With Galera Cluster, you must wait until all the members agree before the transaction is complete. Huge latency can occur with large transactions, which is why large transactions are not advised. Replication may prove challenging if you are not aware of how transactions work.



What Architecture Should I Use?

Now that we've reviewed a few of the benefits and caveats of these types of database architectures, what is the best database architecture available to optimize your use of MySQL? Good question; start by thinking of your business requirements.

You will need to evaluate your specific database architecture needs -- the required redundancy, speed, failover, and other parameters -- against issues such as costs and available infrastructure. You will need to answer questions such as, "Where is MySQL going to be hosted?" "Will I have my own servers?" "Will it be hosted in the cloud or via a hosting service?" "Will I need more hardware?"

You will also be selecting the best possible architecture based on your own expertise. The easiest architecture to implement will be some variant of the master-slave architecture. The safest architecture to use is probably Galera Cluster, but you will need to weigh its use against performance. Your choice depends on what you need and want from a database. MySQL Cluster could be a solution for you, but so could Galera.

It is best not to use a standalone server. Having a single database server is a very, very high-risk strategy. Datavail has seen numerous problems resulting from the failure of a standalone server. A catastrophic failure makes it challenging, if not impossible, to recover data. You always need redundant architecture that allows recovery from any disaster. It is not a matter of if a disaster will occur, but when. Better to be prepared!

Think about what your organization's needs are. Think about the different approaches and which of these might be the best option for your database environment.

There is no single perfect solution for every organization or database.

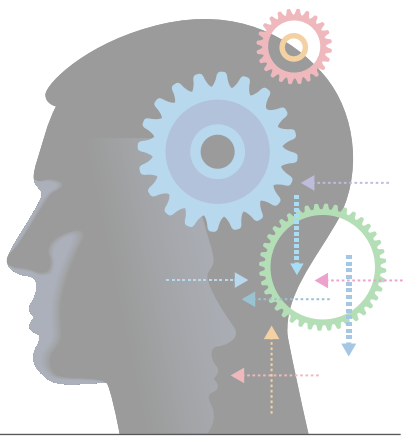
There are several flavors of MySQL out there and they all work very well. Each has its pros and cons, endorsements and caveats associated with it. Ultimately, you will need to select an architecture that will enable your organization to fully access, manage, and update your data resources through MySQL.

More Information on MySQL Database Solutions

There are a host of resources available for MySQL users. If you are still uncertain about the best MySQL architecture options for your organization, Datavail can help you evaluate your current database environment against your company's business priorities and future goals.

The Datavail website provides many resources for those new to MySQL as well as long-time users. We have several options for MySQL and primary benefits related to using MySQL, including how to optimize MySQL for high availability, how to configure it for scalability, and how to use MySQL performance analysis tools.

With nearly 500 database administrators worldwide, Datavail is the largest database services provider in North America. With 24x7 managed database services, including database design, architecture and staffing, Datavail can support your organization as it works with MySQL, regardless of the build you ultimately select. Contact Datavail to discuss a custom MySQL solution designed for your enterprise.



Biography

Charleste King
Lead MySQL DBA for Datavail

Charleste has more than 15 years of experience in the IT industry in a myriad of areas from software development to data analysis, architecture, and administration. She has worked supporting organizations from the very small to enterprise level in aerospace, agriculture, medicine, education, and other industries. She has developed solutions to unique problems for clients ranging from multi-level upgrades with minimal downtime, compliance conversions, documentation, monitoring, alerting, stabilization, trending, and forecasting problem areas, as well as tuning and performance monitoring.

About Datavail

Datavail Corporation is the largest provider of remote database administration (DBA) services in North America, offering database design and architecture, administration and 24x7 support. The company specializes in Oracle, Oracle E-Business Suite, Microsoft SQL Server, MySQL, MongoDB, DB2 and SharePoint, and provides flexible on-site/off-site, onshore/offshore service delivery options to meet each customer's unique business needs.

Contact Us

General Inquiries:
877-722-8247
Fax Number: 303-469-2399
Email: info@datavail.com

Corporate Headquarters:
Datavail Corporation
11800 Ridge Parkway
Suite 125
Broomfield, CO 80021

Database Operations Control Center:
Datavail Infotech Pvt. Ltd
3rd Floor, Unit No. B-3
Ashar IT Park, Road No. 16Z
Wagale Estate
Thane (West), Thane 400604
Direct Telephone Number: 022-61517000

Bangalore Office
Datavail Infotech Pvt. Ltd
Concept Business Park
#319/9, 1st floor, Block A
Hosur Main Road
Bommanahalli, Bangalore 560100

datAvail

DATABASE SERVICES

www.datavail.com | 877.634.9222