Glossary

Colored Contents	Represents
BLACK	Normal DB Activities (Managed Services)
RED	New Implementations (Professional Services)

Support Type	Related Tasks
Full Support	Covers all **BLACK** + **RED** Colored tasks mentioned in this document. **NOTE**: All **RED** colored contents will be executed in Serial manner (one task at a time). If any multiple **RED** colored tasks needed to be executed in parallel will be carried over with additional costs based on the workload proposed.
Normal Support	Covers ONLY BLACK colored tasks mentioned in this document

---

Equivalent Work Experience

Tasks Level	
Level-0	Freshers
Level-1	1-2 years exp
Level-2	3-5 years exp
Level-3	5+ years exp
Level-4	8+ Technology & Leadership exp
ITIL	Process Expertise

MYSQL/MARIA Database Project & Support Services

Database Health Check (level-1)

1. Daily check the status of mysql process is running in prod server
2. To monitor the error log in mysql and find any ERROR / WARNINGS report to DBA team.
3. Check the backup status (MySQLDump, Percona)
4. To check the status of the replication master and slave.
5. Database Space check
6. OS Space check
7. To monitor the binlog file growth per day and storage usage.
8. Proactive basic Performance check
   a. Connection Thread Caching
   b. CPU/MEMORY utilization
   c. Blocking/Locking Reports
   d. I/O Monitoring

Database Design, Installation & Configuration. (Level-3)

1. Mysql/MariaDB installation with source binary files/tar file on different OS linux, centos, Ubuntu.
2. Design OS requirements: Choosing & configuring the appropriate CPU, Memory, storage and ancillary software
3. Design Database requirements: Creating scalable database architectures that allows for expansion
4. Conducting design reviews
5. Designing database for high-speed, high-volume transactions
6. Installation and configuration of database from source or Packages on all types Operating Systems.
7. Preparing documentation
8. Installation & configuration of all types Database client side & server tools for both windows & Linux.

MySQL Storage Engines

- **InnoDB:** The default storage engine in MySQL 8.0. InnoDB is a transaction-safe (ACID compliant) storage engine for MySQL that has commit, rollback, and crash-recovery capabilities to protect user data. InnoDB row-level locking (without escalation to coarser granularity locks) and Oracle-style consistent nonlocking reads increase multi-user concurrency and performance.

- **MyISAM:** These tables have a small footprint. Table-level locking limits the performance in read/write workloads, so it is often used in read-only or read-mostly workloads in Web and data warehousing configurations.

- **Memory:** Stores all data in RAM, for fast access in environments that require quick lookups of non-critical data. This engine was formerly known as the HEAP engine. Its use cases are decreasing; InnoDB with its buffer pool memory area provides a general-purpose and durable way to keep most or all data in memory, and NDBCLUSTER provides fast key-value lookups for huge, distributed data sets.

- **Archive:** These compact, unindexed tables are intended for storing and retrieving large amounts of seldom-referenced historical, archived, or security audit information.

- **NDB:** (also known as NDBCLUSTER): This clustered database engine is particularly suited for applications that require the highest possible degree of uptime and availability.

Feature	MyISAM	Memory	InnoDB	Archive	NDB
B-tree indexes	Yes	Yes	Yes	No	No
Backup/point-in-time recovery (note 1)	Yes	Yes	Yes	Yes	Yes
Cluster database support	No	No	No	No	Yes
Clustered indexes	No	No	Yes	No	No
Compressed data	Yes (note 2)	No	Yes	Yes	No
Data caches	No	N/A	Yes	No	Yes
Encrypted data	Yes (note 3)	Yes (note 3)	Yes (note 4)	Yes (note 3)	Yes (note 3)
Foreign key support	No	No	Yes	No	Yes (note 5)
Full-text search indexes	Yes	No	Yes (note 6)	No	No
Geospatial data type support	Yes	No	Yes	Yes	Yes
Geospatial indexing support	Yes	No	Yes (note 7)	No	No
Hash indexes	No	Yes	No (note 8)	No	Yes
Index caches	Yes	N/A	Yes	No	Yes
Locking granularity	Table	Table	Row	Row	Row
MVCC	No	No	Yes	No	No
Replication support (note 1)	Yes	Limited (note 9)	Yes	Yes	Yes
Storage limits	256TB	RAM	64TB	None	384EB
T-tree indexes	No	No	No	No	Yes
Transactions	No	No	Yes	No	Yes
Update statistics for data dictionary	Yes	Yes	Yes	Yes	Yes

Backup & Disaster Recovery Implementation & Management (Level-2)

1. Configure new backups
2. Re-Run failed backups
3. Ensure, all backups (data & archive logs) are stored in a safer remote location to restore consistency.
4. Install percona backup tool to initiate the full backup and incremental backup, to improve the speedy and for recovery ie. For big database size ex. 1TB
5. PITR for Database & table restoration using backup and binlogfiles.
6. Resolving table and index level database corruption using backup techniques.

Backup Types

Hot Backup for InnoDB

Parallel backup, recovery operations

Compressed Backup

Full, Incremental, Partial Backups & Full, Partial Restore

Scriptable, command line interface

Integrated backup with Oracle Secure Backup, NetBackup, Tivoli

Recovery Types

Point in Time Recovery

Auto-Restart/Recovery

Standby Database Management (Database Replication) (Level-2 & 3)

1. Architecting, Installation & Configuration of a new standby database.
2. Re-sync if in case of any gaps
3. Installing and configuring proxysql for master and slave for load balancer/HA.
4. Perform DR drill activity every 6-month DC/DR activity.

Replication Method: MySQL supports two (or three, depending on how you look at it) different methods of replicating databases from master to slave. All these methods use the binary log; however, they differ in the type of data that is written to the master's binary log.

1. **Statement-based replication** Under this method, the binary log stores the SQL statements used to change databases on the master server. The slave reads this data and re-executes these SQL statements to produce a copy of the master database. This is the default replication method in MySQL 5.1.11 and earlier and MySQL 5.1.29 onwards.

2. **Row-based replication** Under this method, the binary log stores the record-level changes that occur to database tables on the master server. The slave reads this data and manipulates its records accordingly to produce a copy of the master database.

3. **Mixed-format replication** Under this method, the server can dynamically choose between statement-based replication and row-based replication, depending on certain conditions. Some of these conditions include using a user-defined function (UDF), using an INSERT command with the DELAYED clause, using temporary tables, or using a statement that uses system variables. This is the default replication method in MySQL 5.1.12 to MySQL 5.1.28.

Types of Replications

Master - slave replication configuration
Master - Master replication configuration
Circular replication
Cascade standby replication.
Built-in Replication Engine
Row-based Replication
Multi-source Replication
Time-delayed Replication
Replica Failover, Recovery
Multi-threaded replicas
Group Replication

Database User and Security Management (Level-1 & 2)

1. User creation with proper authentication(credentials) & authorization (Privileges).
2. User Management in existing setup.
3. Configuring application connections & troubleshooting.
4. OpenSSL by Default
5. SQL Roles
6. Password management
7. Proxy Users
8. Setting Account Resource Limits
9. SQL-Based Account Activity Auditing
10. The Connection-Control Plugins
    - MySQL Server includes a plugin library that enables administrators to introduce an increasing delay in server response to connection attempts after a configurable number of consecutive failed attempts

11. Authentication Plugins

- Native Pluggable Authentication
- Caching SHA-2 Pluggable Authentication
- SHA-256 Pluggable Authentication
- Client-Side Cleartext Pluggable Authentication
- PAM Pluggable Authentication
- Windows Pluggable Authentication
- LDAP Pluggable Authentication
- Kerberos Pluggable Authentication
- No-Login Pluggable Authentication
- Socket Peer-Credential Pluggable Authentication
- FIDO Pluggable Authentication
- Test Pluggable Authentication
- Pluggable Authentication System Variables

12. MySQL Keyring
13. MySQL Enterprise Firewall
14. Security-Related mysqld Options and Variables
    - allow-suspicious-udf
    - automatic_sp_privileges
    - chroot
    - local_infile
    - safe-user-create
    - secure_file_priv
    - skip-grant-tables
    - skip_name_resolve
    - skip_networking
    - skip_show_database

Database Encryption

1. Enabling/Disabling table wise encryption

Pros of database level encryption

- Full power of DBMS is available
- Easy to implement
- Only database can see the data
- Per-table encryption, per-table keys, performance

Cons of database level encryption

- Cannot be done per-user
- Does not protect against malicious root user

SECURITY STANDARDS

- **Modify port mappings** – avoid keeping default ports because they are known by attackers.
- **Avoid running MySQL with root privileges** – this can allow the compromise of the database to lead to privilege escalation on the host machine.
- **Secure MySQL in the cloud** – accidental exposure to cloud-based databases can have catastrophic consequences.
- **Disable and delete MySQL history** – this prevents attackers who gain access to a database account from gleaning valuable data.
- **Lock users' accounts on suspicious activity** – use built-in MySQL features to lock accounts after too many failed login attempts.
- **Use authentication plugins** – ensure your database has strong authentication.
- **Restrict or disable database visibility** – disable the SHOW DATABASES command that can provide sensitive information to attackers.
- **Encrypt data at rest and in transit** – encryption can prevent exposure of sensitive data even if the database is compromised.
- **Enforce secure password policies** – MySQL lets you set policies to ensure that users and administrators have passwords that are not easily guessable

Database Capacity Planning & Space Management (Level-3)

1. **Designing database storage requirements by considering immediate future growth.**
2. Performing periodic capacity reviews (daily, Weekly, Monthly, and Quarterly based on customer requirements & send database growth report regularly.
3. Add additional space by taking approval from management.
4. Pro-active housekeeping activities to clear old logs, Backup files & Backup logs from the OS database drive, as per customer standards to avoid last-minute space hiccups.
5. Monitoring the DB growth for every quarter and planning to implement the table partitioning for better performance.

Database Performance Tuning: (level 2 & 3)

1. To check the long running queries in slow query logs which is taking more than 10 mins will report to the application team to find any blocking session/wait events
2. Max connections monitoring and clean up sleep connections.
3. To find slow, long running, blocking queries will capture and report to application team.
4. OS Tuning – hugepages/kernel parameters/open file limits.
5. Optimizing databases and queries in MySQL server environment.
6. Tuning innodb buffer size and other tuning parameters based on hardware and memory.
7. Performed maintenance on MySQL databases, running MySQL check, analyze and optimize as needed.
8. Analyzing database logs for errors and performing routine table and index maintenance tasks.
9. Partitioning MySQL to improve database performance.

   Types: -

   - RANGE partitioning
   - LIST partitioning
   - HASH partitioning
   - KEY partitioning

10. **High Availability Performance Tunning: -**

    innodb_buffer_pool_size: is where data and indexes are cached: having it as large as possible will ensure you use memory and not disks for most read operations.

    innodb_log_file_size: size of the redo logs. The redo logs are used to make sure writes are fast and durable and also during crash recovery.

    max_connections: if you are often facing the 'Too many connections' error, max_connections is too low. It is very frequent that because the application does not close connections to the database correctly, you need much more than the default 151 connections.

innodb_file_per_table: this setting will tell InnoDB if it should store data and indexes in the shared tablespace (innodb_file_per_table = OFF) or in a separate .ibd file for each table (innodb_file_per_table= ON). Having a file per table allows you to reclaim space when dropping, truncating or rebuilding a table.

innodb_flush_log_at_trx_commit: the default setting of 1 means that InnoDB is fully ACID compliant. It is the best value when your primary concern is data safety, for instance on a master. However, it can have a significant overhead on systems with slow disks because of the extra fsyncs that are needed to flush each change to the redo logs.

innodb_flush_method: this setting controls how data and logs are flushed to disk. Popular values are O_DIRECT when you have a hardware RAID controller with a battery-protected write-back cache and fdatasync (default value) for most other scenarios. sysbench is a good tool to help you choose between the 2 values.

innodb_log_buffer_size: this is the size of the buffer for transactions that have not been committed yet. The default value (1MB) is usually fine but as soon as you have transactions with large blob/text fields, the buffer can fill up very quickly and trigger extra I/O load. Look at the Innodb_log_waits status variable and if it is not 0, increase innodb_log_buffer_size

log_bin: enabling binary logging is mandatory if you want the server to act as a replication master. If so, don't forget to also set server_id to a unique value. It is also useful for a single server when you want to be able to do point-in-time recovery: restore your latest backup and apply the binary logs.

key_buffer_size – Very important if you use MyISAM tables. Set up to 30-40% of available memory if you use MyISAM tables exclusively. Right size depends on amount of indexes, data

size and workload – remember MyISAM uses OS cache to cache the data so you need to leave memory for it as well, and data can be much larger than indexes in many cases

table_cache – Opening tables can be expensive. For example MyISAM tables mark MYI header to mark table as currently in use. You do not want this to happen so frequently and it is typically best to size your cache so it is large enough to keep most of your tables open. It uses some OS resources and some memory but for modern hardware it is typically not the problem. 1024 is good value for applications with couple hundreds tables (remember each connection needs its own entry) if you have many connections or many tables increase it larger. I've seen values over 100.000 used.

thread_cache – Thread creation/destructions can be expensive, which happen at each connect/disconnect. I normally set this value to at least 16. If application has large jumps in amount of concurrent connections and I see fast growth of

Threads_Created variable I boost it higher. The goal is not to have threads created in normal operation.

query_cache_size – If your application is read intensive and you do not have application level caches this can be great help to optimize your MySQL database. Do not set it too large as it may slow things down as its maintenance may get expensive. Values from 32M to 512M normally make sense. Check it however after a while and see if it is well used. For certain workloads cache hit ratio is lower than would justify having it enabled.

Database Patch & Upgrade (Level-2)

1. Upgrade Mysql and MariaDB different version on linux/centos/Ubuntu.
2. Apply Security patching regularly.

Database High-Availability configuration & Management (Level-2 & 3)

1. Galera cluster installation and configuration for MariaDB cluster.
2. Installation and configuration for MariaDB maxscale for HA.
3. Managing existing HA setup

HA Types

a) **NDB Cluster**: Technology that enables clustering of in-memory databases in a shared-nothing system. The shared-nothing architecture enables the system to work with very inexpensive hardware, and with a minimum of specific requirements for hardware or software.

b) **Inno DB Cluster**: MySQL InnoDB ClusterSet provides disaster tolerance for InnoDB Cluster deployments by linking a primary InnoDB Cluster with one or more replicas of itself in alternate locations, such as different datacenters.

Feature	InnoDB (MySQL 5.7)	NDB 7.5/7.6
MySQL Server Version	5.7	5.7
InnoDB Version	InnoDB 5.7.41	InnoDB 5.7.41
NDB Cluster Version	N/A	NDB 7.5.30/7.6.26
Storage Limits	64TB	128TB (as of NDB 7.5.2)
Foreign Keys	Yes	Yes
Transactions	All standard types	READ COMMITTED
MVCC	Yes	No

Feature	InnoDB (MySQL 5.7)	NDB 7.5/7.6
Data Compression	Yes	No (NDB checkpoint and backup files can be compressed)
Large Row Support (> 14K)	Supported for VARBINARY, VARCHAR, BLOB, and TEXT columns	Supported for BLOB and TEXT columns only (Using these types to store very large amounts of data can lower NDB performance)
Replication Support	Asynchronous and semisynchronous replication using MySQL Replication; MySQL Group Replication	Automatic synchronous replication within an NDB Cluster; asynchronous replication between NDB Clusters, using MySQL Replication (Semisynchronous replication is not supported)
Scaleout for Read Operations	Yes (MySQL Replication)	Yes (Automatic partitioning in NDB Cluster; NDB Cluster Replication)
Scaleout for Write Operations	Requires application-level partitioning (sharding)	Yes (Automatic partitioning in NDB Cluster is transparent to applications)
High Availability (HA)	Built-in, from InnoDB cluster	Yes (Designed for 99.999% uptime)
Node Failure Recovery and Failover	From MySQL Group Replication	Automatic (Key element in NDB architecture)
Time for Node Failure Recovery	30 seconds or longer	Typically < 1 second
Real-Time Performance	No	Yes
In-Memory Tables	No	Yes (Some data can optionally be stored on disk; both in-memory and disk data storage are durable)
NoSQL Access to Storage Engine	Yes	Yes (Multiple APIs, including Memcached, Node.js/JavaScript, Java, JPA, C++, and HTTP/REST)
Concurrent and	Yes	Up to 48 writers, optimized for

Feature	InnoDB (MySQL 5.7)	NDB 7.5/7.6
Parallel Writes		concurrent writes
Conflict Detection and Resolution (Multiple Replication Surces)	Yes (MySQL Group Replication)	Yes
Hash Indexes	No	Yes
Online Addition of Nodes	Read/write replicas using MySQL Group Replication	Yes (all node types)
Online Upgrades	Yes (using replication)	Yes
Online Schema Modifications	Yes, as part of MySQL 5.7	Yes

Database Refresh (Level-2)

a. Perform DB refresh using physical backup/restore.
b. Refresh data using the logical method
c. Refresh data from TAPE (storage)
d. Tables and database refresh activity using mysqldump and mysql.

Database Migration (Level-3)

Methods

Straight Platform: One Server to Another

Cloud Migration: On premises to any cloud & vice-versa

Cross Technologies: Between different database Technologies

Cross Platform: Between different Operating System types.

Between Datacentres

1. Migration from MySQL to MariaDB
2. The MySQL Workbench Migration: configuring and managing a complex migration process, that supports:

- **Database migrations -** enables migrations from Microsoft SQL Server, Microsoft Access, PostgreSQL, Sybase ASE, Sybase SQL Anywhere, SQLite, and more.
- **Manage Migration Projects -** allows migrations to be configured, copied, edited, executed, and scheduled.
- **Source and Target selection -** allows users to define specific data sources and to analyze source data in advance of the migration.
- **Object migration -** allows users to select objects to migrate, assign source to target mappings where needed, edit migration scripts, and create the target schema.
- **Version Upgrades -** using migration users can easily move databases off older MySQL versions to the latest.

Vendor & Licensing Management (Level-2)

a. Utilizing our vendor relationships, we increase turn-around speed and get licensing information quickly.
b. We navigate the licensing maze, providing you with all the licensing options available.
c. We regularly work with software vendors to get the "best pricing" for our clients.

Database Continuous Monitoring (Level-1)

a. Implementation of Customised Monitoring using Shell & PowerShell.
b. MySQL Enterprise Monitoring: Database Availability, Lister, Storage and process, DB performance.
c. Splunk: Monitor the logs and raise on-call alerts to DBAs at the time of an incident.
d. Proactive checks based on the monitoring alerts help DBAs to keep the system healthy.

Database Process Implementation, Documentation & Orientation (ITIL)

- Introducing best-fit processes for database administration using ITIL global standards.
- Deriving & implementing new processes & periodically reviewing existing processes, in Incident Management, Problem Management, Change Management & Release management.
- Can Manage ITIL portfolio by supplying, Incident Manager, Problem Manager, Change Manager, Release Manager.
- Documenting the SOP (Standard Operating Procedures) for end-to-end database administration portfolio for your various environments Prod, QA & Development.
- Also we educate your IT / Non-IT staffs for the Process & SOP structured.

Database Status Reporting to IT Management (Level 1)

1. **Backup Status Report:** (Daily/Weekly/Monthly/Quarterly) backup status reports for end-to-end database environment.
2. **Daily Status Check Report:** Services Availability Status, Storage Status & Standby Sync status, HA Availability Status.
3. **Performance Status Report:** Periodic database Locking, Blocking, Inactive sessions status.
4. **Capacity Growth Report:** Periodic database storage growth (Daily/Weekly/Monthly/Quarterly) for Production databases.
5. **Others:** Other Database related report as the customers customized requirements.

Cloud 'Database as Service' support (Level-2 & 3)

End to End Database Consultation, Administration & Management in all Cloud Technologies (as well as in Hybrid Model)

We are expertise in supporting below cloud environments

Cloud We Support

Also we are in Technical Partnership with **SpeedCloud** cloud company and provide special discounted rates for our customers on our Techno-partner **SpeedCloud** cloud platform.

Also we are in Technical Partnership with xxx cloud company and provide special discounted rates for our customers on our Techno-partner xxx cloud platform. CLICK HERE

Our Database Support in cloud includes: -

- **Rehost patterns** (from on-perm database to self managed in Amazon EC2, Azure VM, Compute Engine in GCC etc.)
- **Re-platform patterns** (from on-perm database to cloud DBaaS*)
- **Re-architect patterns** (from on-perm one database Technology to other open-source or cloud-native database Technologies)
- Setup monitoring of DBs using cloud native tools. (Eg: Cloud watch)
- Setup and manage DB in muti, hybrid cloud environment like AWS, AZURE & GCC along with On-Perm
- **Database Consulting in Cloud Infra**
    - Start with a comprehensive plan and a governance framework
    - Run the right database in the right cloud
    - Use data services that support multi-cloud environments
    - Exploit managed database services, or DBaaS
    - Consider database portability across multiple clouds
    - Optimize data access for applications and end users
    - Connect cloud networks to reduce data latency

- **Cloud Security**
    - Define standards, security, and compliance policies. Cloud database vendors rarely enforce more than the most obvious weaknesses in the out-of-the-box installations of their platforms
    - Run vulnerability assessments. Since databases are often an organization's largest repository of sensitive information, they should be evaluated to not only search for potential vulnerabilities but also to ensure they fulfill any relevant regulatory compliance requirements.
    - Understand user privilege and access. As people change roles or leave an organization, user privileges are often not kept up-to-date, and, as a result, organizations lack a full understanding of who has access to sensitive data.
    - Use data analytics to mitigate risks. Remediating high-risk vulnerabilities and misconfigurations within your databases not only reduces your risk of compromise, but it also narrows the scope of any required compensating controls you might need, such as exploit monitoring.
    - Respond to policy violations in real time. For vulnerabilities that cannot be remediated or patched in a timely manner, real-time database activity monitoring (DAM) can be an appropriate compensating control.
    - Database encryption
    - Data Masking
    - Multi-Factor Authentication (MFA)

- **Cloud Database Migration**

    - Migrate your existing database platform from On-premises datacenter to any cloud Technologies includes (AWS, Azure, Google Cloud, Oracle Cloud & other 3rd party cloud services) & Vice versa, Cloud to On-premises datacenter
    - New database design and implementation in Cloud platform.
    - Cloud database integration with other cloud services as per your business functional requirements.

- **Cloud Database Performance Tuning**

- **MySQL native Cloud DB Support & Machine Learning. Below are its benefits:**

  Single Database for OLTP, OLAP, and ML

  - Native, real-time analytics
  - Single database service for OLTP, OLAP, and ML
  - Eliminate separate analytics databases and ML services
  - Eliminate the cost, complexity, and risk of ETL

  HeatWave AutoML

  - Native, In-Database Machine Learning
  - Build ML Models using SQL Commands
  - Automate the ML Lifecycle
  - 25x Faster than Redshift ML

  MySQL HeatWave LakeHouse (beta)

  - 400 TB of Data, MySQL syntax
  - Query Data in MySQL and Object Store

  Faster Performance

  - 5400x Faster than Amazon RDS
  - 1400x Faster than Amazon Aurora
  - 6.5x Faster than Amazon Redshift AQUA
  - 17X Faster than Snowflake
  - 6X Faster than Amazon Redshift

  Lower Total Cost of Ownership

- 2/3 the cost of Amazon RDS
- 1/2 the cost of Amazon Aurora
- 1/2 the cost of Amazon Redshift AQUA
- 1/5 the cost of Snowflake

Multi-cloud

- Oracle Cloud Infrastructure (OCI)
- Amazon Web Services (AWS)
- Microsoft Azure

DBaaS* (Database as a service), Managed database service that are fully managed by the vendor, which could be a cloud platform provider or another database vendor that runs its cloud DBMS on a platform provider's infrastructure

Self-managed database*: This is an infrastructure as a service (IaaS) environment, in which the database runs in a virtual machine on a system operated by a cloud provider.

MySQL Server Plugins (Level-3)

1. Installing Plugins
2. Controlling Plugin Activation State
3. Uninstalling Plugins
4. Plugins and Loadable Functions
    - The ddl_rewriter Plugin: MySQL 8.0.16 and higher includes a ddl_rewriter plugin that modifies CREATE TABLE statements received by the server before it parses and executes them.
    - The Clone Plugin: The clone plugin, introduced in MySQL 8.0.17, permits cloning data locally or from a remote MySQL server instance. Cloned data is a physical snapshot of data stored in InnoDB that includes schemas, tables, tablespaces, and data dictionary metadata.
    - The Keyring Proxy Bridge Plugin: MySQL Keyring originally implemented keystore capabilities using server plugins, but began transitioning to use the component infrastructure in MySQL 8.0.24. The transition includes revising the underlying implementation of keyring plugins to use the component infrastructure.