Yash Khokhar

SOLARWINDS

Contents

[SolarWinds Observability SaaS & DPA for MySQL 2](#_Toc193282391)

[Introduction 2](#_Toc193282392)

[SolarWinds Observability SaaS - MySQL Monitoring Setup 2](#_Toc193282393)

[Create a SolarWinds Observability SaaS Account 2](#_Toc193282394)

[Deploy a New SolarWinds Observability Agent 2](#_Toc193282395)

[Configure MySQL for SolarWinds Monitoring 3](#_Toc193282396)

[Register MySQL in SolarWinds Observability SaaS 4](#_Toc193282397)

[Database Performance Analyzer (DPA) - MySQL Setup 5](#_Toc193282398)

[Install SolarWinds DPA 5](#_Toc193282399)

[Register a MySQL Instance in DPA 5](#_Toc193282400)

[Configure the Monitoring User in MySQL 5](#_Toc193282401)

[Configure Connection Properties 6](#_Toc193282402)

[Complete Registration & Monitor MySQL 6](#_Toc193282403)

[Comparison: SolarWinds Observability SaaS vs. DPA 6](#_Toc193282404)

[SolarWinds Observability SaaS 6](#_Toc193282405)

[SolarWinds Database Performance Analyzer (DPA) 6](#_Toc193282406)

[Top 10 Features of SolarWinds Observability SaaS for MySQL 7](#_Toc193282407)

[Top 10 Features of SolarWinds Database Performance Analyzer (DPA) for MySQL 9](#_Toc193282408)

# SolarWinds Observability SaaS & DPA for MySQL

## Introduction

* **Purpose**: This guide explains how to set up SolarWinds Observability SaaS and Database Performance Analyzer (DPA) to monitor MySQL databases effectively.
* **Who It’s For**: The guide is intended for IT administrators, database professionals, and system engineers who need to track MySQL performance and infrastructure health.
* **What’s Included**: It provides detailed steps for setting up both tools and includes a comparison to help you choose the right one for your needs.
* **Goal**: The aim is to ensure you can monitor MySQL with clear instructions and actionable insights.

## SolarWinds Observability SaaS - MySQL Monitoring Setup

### Create a SolarWinds Observability SaaS Account

* **Action**: Sign up for a new account or log in at [SolarWinds Observability SaaS](https://www.solarwinds.com/solarwinds-observability).
* **Navigation**: Once logged in, navigate to **Infrastructure > Databases** in the user interface (UI).
* **Insight**: This section serves as your starting point to manage all monitored resources, including MySQL databases.

### Deploy a New SolarWinds Observability Agent

* **Why**: You need an agent on your MySQL host to collect performance data; check if one already exists under **Infrastructure**.
* **Purpose**: The agent connects MySQL (and other services like Kafka) to the Observability platform for comprehensive monitoring.

#### Recommended Method: Script-Based Installation

* **Start**: In the UI, go to **Add Data > Agent > Deploy New Agent** to begin the process.
* **Choose**: Select **Script-based installation**—it’s recommended because it automatically validates your operating system.
* **Generate Token**:
  + **Step**: Click to create a new ingestion token.
  + **Name**: Give the token a clear name, such as mysql-monitoring-token, for easy identification.
  + **Optional Tags**: After setup, visit **Settings > API Tokens** to add tags (e.g., mysql, prod) to organize your data—refer to the documentation for details.
* **Install the Agent**:
  + **OS Selection**: Choose your host’s operating system—Linux or Windows.
  + **Architecture**: Select the appropriate architecture—AMD64 or ARM—based on your hardware.
  + **Script**: Copy the script provided in the UI; it includes your unique token and endpoint (e.g., na-01.cloud.solarwinds.com).
  + **Example (Windows)**: Here’s a sample script for Windows:
  + [Net.ServicePointManager]::SecurityProtocol = "Tls11, Tls12"; irm https://agent-binaries.cloud.solarwinds.com/uams/latest/uamsclient\_install.ps1 | iex; install -AccessToken "NDby8LcFFvbxV4JCSynxjtlgjVhsi\_FTkeOLwZBlBe3MCb8Yvi91zFQlxPY8LxSx6nS5F5k" -Metadata "role:host-monitoring,installationSessionId:9cce1f15-3531-459c-adf4-e8bf4ccb4eea" -SwoUrl "na-01.cloud.solarwinds.com"
  + **Execution**: Run the script on the target host with Administrator privileges on Windows or root privileges on Linux.
* **Status Check**:
  + **Wait**: After executing the script, wait for the message “Agent installation successful” to appear in the UI.
  + **Next**: The UI will automatically redirect you to configure the agent further.
* **Optional Configuration**:
  + **Host Name**: Set a custom name for the host (e.g., mysql-server-01) instead of using the default.
  + **Host Monitoring**: Enable this option to collect broader server metrics like CPU and memory usage.
  + **Proxy**: If your network requires an HTTP proxy, configure it—see [Configure Proxy for Agents](https://docs.solarwinds.com) for instructions.
* **Insight**: The script-based method is reliable and fast, minimizing manual errors during installation.

#### Alternative Deployment Methods

* **Manual Installer**: Download the agent installer from the UI and run it manually on Linux or Windows; this works well for single hosts.
* **Automated Tools**: Use tools like Ansible, Chef, Puppet, or SaltStack to deploy the agent across multiple hosts; this is ideal for large setups.
* **Docker**: Deploy the agent as a Docker container; this suits containerized environments.
* **Kubernetes**: Install the agent in a Kubernetes cluster; this is perfect for cloud-native MySQL deployments.
* **Recommendation**: Stick to the script-based method unless you require automation or container-specific support.

### Configure MySQL for SolarWinds Monitoring

* **Enable Performance Schema**:
  + **Why**: The Performance Schema provides detailed query and performance data essential for monitoring.
  + **Command**: Execute this SQL command:
  + UPDATE performance\_schema.setup\_consumers SET ENABLED = 'YES' WHERE NAME LIKE 'events\_statements%';
  + **Check**: If the Performance Schema is disabled, edit the my.cnf file to include performance\_schema=ON and restart MySQL.
* **Create Monitoring User**:
  + **Purpose**: You need a dedicated user with minimal privileges for SolarWinds to access MySQL securely.
  + **Commands**: Run these SQL commands:
  + CREATE USER 'solarwinds'@'agent\_host\_ip' IDENTIFIED BY 'your\_password';  
    GRANT PROCESS, SELECT ON \*.\* TO 'solarwinds'@'agent\_host\_ip';  
    FLUSH PRIVILEGES;
  + **Security Tip**: Replace agent\_host\_ip with the agent’s actual IP address (e.g., 192.168.1.20) instead of using % to restrict access in production environments.
* **Insight**: These steps ensure SolarWinds can collect data securely without granting excessive privileges to the monitoring user.

### Register MySQL in SolarWinds Observability SaaS

* **Navigate**: Go to **Infrastructure > Databases > Add Database** in the UI.
* **Details**:
  + **Hostname/IP**: Enter the MySQL server’s address (e.g., 192.168.1.10).
  + **Port**: Use the default port 3306 unless it’s been customized.
  + **Credentials**: Input the solarwinds username and password created earlier.
* **Test Connection**:
  + **Action**: Click **Test Connection** to verify the setup.
  + **Fails?**: If the test fails, check if port 3306 is open in your firewall and confirm the user has the correct permissions.
* **Register**:
  + **Success**: If the test passes, click **Register** to complete the process.
* **Verify**: Look for the MySQL instance in the **Databases** dashboard to confirm it’s being monitored.
* **Insight**: Registering your MySQL instance links it to the platform, enabling real-time metrics once completed.

## Database Performance Analyzer (DPA) - MySQL Setup

### Install SolarWinds DPA

* **Download**: Obtain DPA from the [SolarWinds Customer Portal](https://www.solarwinds.com/database-performance-analyzer).
* **System Requirements**: Ensure the server has at least 4GB of RAM, 2 CPUs, and 10GB of disk space.
* **Install Location**: Install DPA on a separate Windows or Linux server (not the MySQL server) to optimize performance.
* **Process**: Run the installer and follow the on-screen prompts to complete the installation.
* **Access**: After installation, open the DPA web console (e.g., at http://dpa\_host:8123).
* **Insight**: Keeping DPA on a separate server prevents resource conflicts with your MySQL instance.

### Register a MySQL Instance in DPA

* **Start**: In the DPA web UI, click **Register DB Instance for Monitoring** to begin.
* **Select**: Choose **MySQL or Percona** from the options and click **Next**.
* **Enter Details**:
  + **Host/IP**: Provide the MySQL server address (e.g., mysql-server.example.com).
  + **Port**: Use the default port 3306 unless it’s been changed.

### Configure the Monitoring User in MySQL

#### Option 1: Let DPA Create the User (Recommended)

* **Choose**: In the registration wizard, select **Let DPA create a new user**.
* **Provide**: Enter the credentials for a privileged user (e.g., root username and password).
* **Next**: Click **Next**, and DPA will automatically create the monitoring user for you.
* **Insight**: This is the simplest option, reducing manual configuration steps.

#### Option 2: Manually Create the User

* **Purpose**: Create the monitoring user yourself if you prefer more control over the setup.
* **Commands**: Execute these SQL commands:
* CREATE USER 'dpa\_user'@'dpa\_host\_ip' IDENTIFIED WITH mysql\_native\_password BY 'your\_password';  
  GRANT PROCESS, SELECT ON \*.\* TO 'dpa\_user'@'dpa\_host\_ip';  
  GRANT SELECT ON performance\_schema.\* TO 'dpa\_user'@'dpa\_host\_ip';  
  FLUSH PRIVILEGES;
  + **Note**: This setup works with MySQL 5.6 and later; avoid granting unnecessary permissions like INSERT.
* **Enter**: Input the created username and password into the DPA registration wizard.
* **Insight**: Manual creation allows you to enforce custom security settings tailored to your environment.

### Configure Connection Properties

* **SSL (if MySQL enforces it)**:
  + **Add**: In the DPA connection settings, include trustServerCertificate=true;useSSL=true to enable SSL.
* **Non-SSL or Custom**:
  + **String**: Modify the connection string to jdbc:mysql://your-mysql-host:3306?allowPublicKeyRetrieval=true&useSSL=false if SSL isn’t required.
* **Insight**: Adjust these properties based on your MySQL security configuration to ensure a stable connection.

### Complete Registration & Monitor MySQL

* **Finalize**: Click **Next**, review your settings, and then click **Register Database Instance** to complete the process.
* **Monitor**: Visit the **DPA Dashboard** to view MySQL performance data, such as wait times and query statistics.
* **Insight**: Registration activates DPA’s agentless monitoring through JDBC, providing immediate insights into MySQL performance.

## Comparison: SolarWinds Observability SaaS vs. DPA

### SolarWinds Observability SaaS

* **What**: This is a cloud-based tool designed for full-stack observability, covering databases, applications, and infrastructure.
* **Key Features**:
  + It monitors query latency, IOPS, and slow queries in real-time.
  + It provides comprehensive dashboards for CPU, memory, disk, and database health.
  + It uses AI for anomaly detection and predictive analytics.
  + It supports multi-cloud environments (AWS, Azure) and hybrid setups.
  + It integrates with tools like Prometheus and AWS CloudWatch.
  + It sends alerts via email, Slack, and webhooks.

### SolarWinds Database Performance Analyzer (DPA)

* **What**: DPA is an on-premises or cloud-compatible tool focused on database performance monitoring and query tuning.
* **Key Features**:
  + It analyzes query performance with detailed execution plans.
  + It provides in-depth wait-time analysis for CPU, disk, and memory bottlenecks.
  + It tracks historical query trends over time.
  + It supports multiple databases, including MySQL, PostgreSQL, SQL Server, and Oracle.
  + It uses a lightweight, agentless architecture via JDBC connections.
  + It offers custom alerts and automated performance reports.

### **Top 10 Features of SolarWinds Observability SaaS for MySQL**

#### **1. Real-Time Query Latency Monitoring**

* **What It Does**: Tracks the time MySQL queries take to execute in real time, highlighting slow queries immediately.
* **How It Works**: The agent collects query performance data via MySQL’s Performance Schema and sends it to the cloud platform, displaying latency metrics in the UI under **Infrastructure > Databases**.
* **Why It’s Valuable**: Allows instant detection of performance issues, reducing downtime and improving user experience without manual checks.
* **Example**: Jane notices a checkout query on her e-commerce site jumps from 50ms to 500ms during a sale. She uses this data to identify and fix it within minutes.

#### **2. Comprehensive Infrastructure Dashboards**

* **What It Does**: Provides a visual overview of MySQL and server health, including CPU, memory, disk, and network metrics.
* **How It Works**: Aggregates agent-collected data into customizable dashboards accessible from the UI, with options to drill into specific metrics.
* **Why It’s Valuable**: Offers a single-pane view of the entire system, speeding up diagnosis of issues affecting MySQL.
* **Example**: Jane sees a disk usage spike alongside slow queries on her dashboard, linking it to a log file overflow she quickly clears.

#### **3. AI-Driven Anomaly Detection**

* **What It Does**: Uses AI to identify unusual MySQL performance patterns, like sudden latency increases or resource spikes.
* **How It Works**: Analyzes real-time and historical data to set baselines, flagging anomalies and sending alerts via email, Slack, or webhooks.
* **Why It’s Valuable**: Proactively catches problems before they escalate, saving time and preventing outages.
* **Example**: Jane gets a Slack alert at 3 AM about a 10x latency spike. She logs in, finds a runaway query, and kills it before customers notice.

#### **4. Script-Based Agent Deployment**

* **What It Does**: Simplifies agent installation on MySQL hosts using an automated script.
* **How It Works**: From **Add Data > Agent**, users generate a token (e.g., mysql-monitoring-token), select OS/architecture, and run a provided script (e.g., PowerShell for Windows) to install the agent.
* **Why It’s Valuable**: Reduces setup errors and time, ensuring reliable data collection with minimal effort.
* **Example**: Jane runs a script on her Windows server, and in 2 minutes, the agent is live, showing “Agent installation successful” in the UI.

#### **5. Multi-Cloud Environment Support**

* **What It Does**: Monitors MySQL across AWS, Azure, or hybrid setups seamlessly.
* **How It Works**: The agent adapts to cloud or on-premises hosts, integrating with cloud-native tools like AWS CloudWatch for unified monitoring.
* **Why It’s Valuable**: Ensures consistent visibility in modern, distributed infrastructures without needing multiple tools.
* **Example**: Jane’s company moves MySQL to AWS. The agent keeps tracking it, showing no latency change post-migration.

#### **6. Flexible Alert Notifications**

* **What It Does**: Sends alerts for MySQL issues via multiple channels like email, Slack, or webhooks.
* **How It Works**: Users configure thresholds (e.g., query latency > 200ms) in the UI, and the system triggers notifications when exceeded.
* **Why It’s Valuable**: Keeps teams informed in real-time, enabling rapid response regardless of their preferred platform.
* **Example**: Jane sets a Slack alert for high IOPS. During a peak load, she’s notified and adjusts a query to lower disk strain.

#### **7. Host Monitoring Integration**

* **What It Does**: Collects broader server metrics (e.g., CPU, memory) alongside MySQL data.
* **How It Works**: Enabled during agent setup (e.g., via script with -Metadata "role:host-monitoring"), it correlates database and host performance in the UI.
* **Why It’s Valuable**: Provides context for MySQL issues tied to server health, improving root cause analysis.
* **Example**: Jane sees MySQL slowdowns match CPU spikes from a backup process, rescheduling it to off-hours.

#### **8. Secure Monitoring User Setup**

* **What It Does**: Creates a dedicated MySQL user with minimal privileges for secure data collection.
* **How It Works**: Users run SQL commands (e.g., CREATE USER 'solarwinds'@'192.168.1.20') to grant only PROCESS and SELECT permissions, limiting access risks.
* **Why It’s Valuable**: Enhances security by restricting agent access, critical for production environments.
* **Example**: Jane configures a solarwinds user for her MySQL server, ensuring the agent can’t alter data even if compromised.

#### **9. Performance Schema Integration**

* **What It Does**: Leverages MySQL’s Performance Schema for detailed query and performance insights.
* **How It Works**: Users enable it with UPDATE performance\_schema.setup\_consumers SET ENABLED = 'YES', allowing the agent to collect rich data like query execution stats.
* **Why It’s Valuable**: Unlocks deep visibility into query behavior, essential for effective monitoring.
* **Example**: Jane enables Performance Schema and spots a query with high I/O waits, optimizing it with an index.

#### **10. Proxy Configuration Support**

* **What It Does**: Allows the agent to work through an HTTP proxy for restricted networks.
* **How It Works**: Configured post-installation via UI settings (referenced in docs), ensuring connectivity in complex environments.
* **Why It’s Valuable**: Enables monitoring in firewalled or enterprise setups without network changes.
* **Example**: Jane’s company uses a proxy. She configures it, and the agent connects to the cloud platform without opening new ports.

### **Top 10 Features of SolarWinds Database Performance Analyzer (DPA) for MySQL**

#### **1. Wait-Time Analysis**

* **What It Does**: Measures time between query requests and responses, breaking it into wait types (e.g., CPU, disk) to identify bottlenecks.
* **How It Works**: Uses JDBC to collect second-by-second data, displaying wait events in the DPA Dashboard with detailed breakdowns.
* **Why It’s Valuable**: Pinpoints exact causes of delays, offering a forensic approach beyond basic metrics.
* **Example**: Bob sees a 10-second query wait. DPA shows 8 seconds are disk I/O, prompting him to optimize storage.

#### **2. Detailed Query Execution Plans**

* **What It Does**: Provides MySQL query execution plans to analyze and optimize performance.
* **How It Works**: Captures query details via JDBC, ranks them by impact, and shows EXPLAIN plans in the UI for tuning insights.
* **Why It’s Valuable**: Gives actionable steps to fix inefficient queries, directly improving database speed.
* **Example**: Bob finds a slow JOIN query with a full table scan in the plan. He adds an index, cutting runtime from 3 seconds to 0.1 seconds.

#### **3. Historical Trend Tracking**

* **What It Does**: Stores MySQL performance data over time (e.g., 30 days) for trend analysis.
* **How It Works**: Continuously collects and condenses data, viewable in time-series graphs on the DPA Dashboard.
* **Why It’s Valuable**: Reveals recurring issues or patterns, aiding proactive optimization and planning.
* **Example**: Bob notices a query slows every Friday. Historical data ties it to a report, so he shifts it to midnight.

#### **4. Agentless Monitoring**

* **What It Does**: Monitors MySQL without host agents, using JDBC with minimal overhead (<1%).
* **How It Works**: Connects directly to MySQL via the DPA server, pulling metrics and query data without impacting the host.
* **Why It’s Valuable**: Simplifies setup and preserves server resources, ideal for sensitive production systems.
* **Example**: Bob deploys DPA on a separate server, monitoring his MySQL instance with no performance hit.

#### **5. Custom Alerts**

* **What It Does**: Triggers notifications for specific MySQL performance thresholds (e.g., wait time > 500ms).
* **How It Works**: Users set rules in the UI under **DPA Alerts**, choosing email, Slack, or Teams for delivery.
* **Why It’s Valuable**: Ensures rapid response to issues, tailored to team workflows.
* **Example**: Bob sets an alert for waits over 1 second. He gets an email during a spike and fixes a locking issue fast.

#### **6. Automated Performance Reports**

* **What It Does**: Generates scheduled reports (e.g., Top SQL) to track MySQL performance over time.
* **How It Works**: Configured in the UI under **DPA Reports**, reports can be emailed weekly or monthly with custom metrics.
* **Why It’s Valuable**: Provides long-term insights for tuning and stakeholder updates without manual effort.
* **Example**: Bob schedules a weekly Top SQL report, showing a 20% performance gain after optimization.

#### **7. SQL Statement Search**

* **What It Does**: Lets users find specific MySQL queries by text, user, or time range.
* **How It Works**: Available since DPA 2021.1, the UI offers simple or advanced search options to locate queries not in Trends charts.
* **Why It’s Valuable**: Speeds up analysis of obscure or problematic queries, enhancing troubleshooting.
* **Example**: Bob searches for a query by “app\_user” and finds it’s causing sporadic slowdowns, then optimizes it.

#### **8. Resource Metrics Correlation**

* **What It Does**: Tracks CPU, memory, disk I/O, and network usage alongside MySQL performance.
* **How It Works**: Displays metrics on the Resources tab with historical baselines, linking them to query waits.
* **Why It’s Valuable**: Connects database issues to server conditions, improving holistic diagnosis.
* **Example**: Bob sees a CPU spike matches a slow query, identifying a resource contention issue he resolves.

#### **9. Database Instance Grouping**

* **What It Does**: Organizes MySQL instances into groups (e.g., by type or location) for easier management.
* **How It Works**: Auto-groups clusters like RAC or allows manual grouping in the UI, visible on the DPA home page.
* **Why It’s Valuable**: Simplifies applying settings or alerts across multiple instances, saving time in large setups.
* **Example**: Bob groups his 3 MySQL servers as “Production,” applying a wait-time alert to all at once.

#### **10. REST API Automation**

* **What It Does**: Automates DPA tasks like registering instances or generating reports via API.
* **How It Works**: Uses HTTPS requests (detailed in the admin guide) to script actions, integrating with other systems.
* **Why It’s Valuable**: Streamlines management in large or dynamic environments, reducing manual work.
* **Example**: Bob scripts a nightly report generation via API, saving 30 minutes of daily effort.