Quest®

Foglight for PostgreSQL

Cartridge Guide

TABLE OF CONTENTS

Introduction to PostgreSQL

PostgreSQL is a powerful, open source object-relational database system originally developed at UC Berkeley by computer science professor Michael Stonebraker, who went on to become the CTO of Informix Corporation. Stonebraker started Postgres in 1986 as a follow-up project to its predecessor, Ingres, now owned by Computer Associates. With more than 15 years of active development, it has earned a reputation for ensuring reliability and data integrity. PostgreSQL supports all major operating systems, including Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64), and Windows.

PostgreSQL has evolved into an enterprise class database and boasts sophisticated features such as Multi-Version Concurrency Control (MVCC), point in time recovery, tablespaces, asynchronous replication, nested transactions (savepoints), online/hot backups, a sophisticated query planner/optimizer, and write ahead logging for fault tolerance. It supports international character sets, multibyte character encodings, Unicode, and it is locale-aware for sorting, case-sensitivity, and formatting. It is highly scalable both in the sheer quantity of data it can manage and in the number of concurrent users it can accommodate. There are active PostgreSQL systems in production environments that manage in excess of 4 terabytes of data.

Many organizations, government agencies and companies use PostgreSQL. You will find installations in ADP, CISCO, NTT Data, NOAA, Research In Motion, The US Forestry Service, and The American Chemical Society. Today, it's rare to find a large corporation or government agency which isn't using PostgreSQL in at least one department.

Foglight for PostgreSQL Overview

Foglight and PostgreSQL - A Powerful Combination

Foglight for PostgreSQL brings, true enterprise-class monitoring for PostgreSQL to the industry's leading cross-platform database management solution. On its own, Foglight for Postgres is a powerful monitoring solution. When used in conjunction with other components from the Foglight suite, it has the ability to provide true end-to-end transaction insight. Response times can be collected for end user sessions and transactions and can be correlated with metrics from the application and database layers as well as the underlying physical and virtual infrastructure. The ability to collect and correlate information across infrastructure layers further increases the value of the solution, making Foglight for PostgreSQL an invaluable part of this industry-leading Application Performance Monitoring (APM) solution.

From high-level server operations to database and table-level statistics, Foglight for PostgreSQL provides real-time monitoring and alerting, offering users real-time insight into the performance and availability of their PostgreSQL environment. The solution leverages the holistic data analysis and presentation that have made Foglight a leading Gartner Magic Quadrant APM solution. The addition of PostgreSQL monitoring to Foglight provides customers with the ability to monitor PostgreSQL availability and performance alongside Oracle, MS-SQL Server, MySQL, DB2/UDB, and Sybase. Databases can be managed using a common interface, making it the leading cross-platform database management solution on the market today. With the ability to be used as a standalone solution or in conjunction with other solutions available from Dell, Foglight for PostgreSQL is a unique solution offering best-in-class monitoring and an unmatched value proposition.

Ease of Installation

Installation of the solution is straightforward and can be accomplished in minutes. It can be deployed as an agent-based or agentless solution. The agent-based deployment requires that agent code run locally (on the DB host) whereas the agentless configuration allows the agent to be run remotely from another host. Once installed, the solution provides granular control of collection intervals and the option to disable collections for entire databases or database child objects for which monitoring is deemed unnecessary. Within minutes, Information, Dashboards, and Expert advice is available, providing a platform that will ensure the heath and availability of PostgreSQL for years to come.

Data Visualization and Reporting

Foglight for PostgreSQL was designed to provide quick access to vital database statistics. Utilizing rich dashboards and robust data analysis, the solution uncovers critical information on server operations, including the background writer, write ahead logs, connections, locks, and replication status. The solution integrates data from Foglight host monitoring and incorporates these metrics into key dashboards, permitting system data to be presented alongside rich database metrics. Dashboards display server health and availability information alongside vital statistics such as resource allocation and system metrics from the host machine.

Database and table-level statistics are exposed through the use of both standard and derived metrics. These metrics include memory hit rates, read rates, tuple modifications, transaction commits/rollbacks, standby server conflicts, disk sizes, and deadlocks. Additional information is provided for databases, tables, and indexes. Function comparisons show frequency of use and execution time statistics. Information displayed in dashboards can be used to create and schedule reports. Operators then have

the ability to utilize the data as-is, or define their own dashboards and reports using the built-in drag-and-drop functionality.

Administration and Workflow Automation

For those customers who wish to administer their PostgreSQL environment directly from Foglight, some basic administrative actions are available. With properly assigned permissions, these actions can be taken directly from the Foglight console. Actions can also be triggered as a result of a condition, or grouped together in workflows. Customers who wish to identify and store additional actions may do so on demand.

Capacity Planning and Alerting

The solution incorporates functionality for both trending/capacity planning and alerting. To facilitate capacity planning, a tablespaces dashboard exposes growth trends over time, a breakdown of disk space usage on the host machine, and remaining available space. Alerting is based on a robust set of rules and definitions that are included with the solution. Rules can be triggered based on both static and dynamic thresholds. Dynamic thresholds are triggered when the system detects a deviation from normal behavior patterns. The system will automatically create a baseline for usage and alert on environmental anomalies.

Paying Dividends

With feature-rich out-of-the-box dashboards and rules, PostgreSQL for Foglight provides unparalleled insight into your PostgreSQL infrastructure and offers immediate benefits to DBAs and Administrators. Over time, the solution continues to pay dividends by offering historical trend analysis, scheduled reporting, alert management, and the ability to create user-defined dashboards.

Combined, these features bring great insight into database and application availability and ensure improved access to mission-critical data.

Foglight for PostgreSQL Requirements

Foglight for PostgreSQL is compatible with PostgreSQL version 9.1+. However, some data may be missing from versions prior to 9.2, most notably I/O timing metric statistics, which are unavailable in those versions.

The agent may be run on a FglAM that is either local or remote to the PostgreSQL server. More information on configuring a PostgreSQL DB user for monitoring can be found in the PostgreSQL Agent User Permissions section of this document.

For workload and other I/O metrics, track_io_timing must be set to "on". More info is available here: https://www.postgresql.org/docs/current/static/runtime-config-statistics.html.

Installation of the pg_stat_statements module and creation of the extension in at least one database is required for statement tracking. Basic steps follow below. For more information on how to install this, refer to the relevant PostgreSQL documentation for your version.

1) In postgresql.conf, set these sample properties:

shared_preload_libraries = 'pg_stat_statements'
pg_stat_statements.max = 10000
pg_stat_statements.track = all

More info on properties: https://www.postgresql.org/docs/current/static/pgstatstatements.html

2) Restart server

3) In PSQL or pgAdmin, execute "CREATE EXTENSION pg_stat_statements;" pg_stat_statements view should appear in DB's public schema.

4) Set agent properties to turn statement tracking on and provide database where extension was created.

PostgreSQL Cartridge Installation

1. Open Foglight Management Console.
2. *From the navigation pane, select:* **Dashboards** > **Administration** > **Cartridges** > **Cartridge Inventory**. The Cartridge Inventory screen appears. For more information on agents, see the *Foglight User Guide.*
3. Load the ***PostgreSQLAgent-5_6_4_xx.car*** file by browsing to the location where the .car file exists and then clicking on "Install Cartridge". Leave the "Enable on Install" check box checked.
4. Once the installation is completed on the Foglight Management Server, the PostgreSQL Agent Cartridge will appear in this list below as an installed cartridge.

PostgreSQL Agent User Permissions

The PostgreSQL agent requires use of a database user with certain minimum privileges for the agent to be able to fully monitor the database.

Create a new user with an existing user that has the appropriate permissions or a utility program of your choice. The new user must have access to all databases you wish to monitor. You must also ensure that any firewalls and the pg_hba.conf file allow the created user access from whichever FglAM host the agent is running from. The agent will only query system tables and views and make use of a few system functions in order to gather operational data.

User privileges required for the PostgreSQL agent:

- LOGIN
- SUPERUSER
- CONNECT (on all databases)
- SELECT (on all databases)

The SUPERUSER privilege is required in order to make use of administration functions through the Foglight console. Usage of the functions is restricted to Foglight users who have been granted the "PostgreSQL Administrator" role.

Example:

CREATE USER user1 WITH PASSWORD 'password1' LOGIN SUPERUSER;

PostgreSQL Agent Properties

When an agent connects to the Foglight Management Server, it is provided a set of properties that is then used to configure its correct running state.

Default Agent properties are installed with the Foglight Cartridge for PostgreSQL. However, the user will typically edit the default agent properties. Agent properties may apply only to a specific agent instance, or may be applicable across multiple agents.

For more information about working with agent properties, see the *Foglight Administration and Configuration Guide.*

To modify the agent properties for a new agent instance:
Open Foglight.
1. From the navigation pane, select: **Dashboards** > **Administration** > **Agents** > **Agent Status**. The Agent Status screen appears.
2. Deploy the PostgreSQL agent package.
3. Create a PostgreSQL agent instance.
4. Click on a PostgreSQL agent type row. The selected row is highlighted with a yellow background.
5. Click the **Edit Properties** button.
6. Select **Modify the default properties for this agent**.
7. Edit the agent properties for the PostgreSQL agent instance:
   - Setting Server Connection Parameters (mandatory)

-
8. Click the **Activate** button.

Setting Connection Parameters (mandatory)

- **IP or Hostname** – Host where PostgreSQL database server is running. Default is "localhost". (e.g.<hostname> or <IP address>)
- **Port** – Port the PostgreSQL database is running. Default is 5432.
- **Database** – Name of PostgreSQL database for server data collection. The agent will monitor all databases by default, but this is the database that will be accessed for primary data collection.
- **Username** – User that can connect to the PostgreSQL server being monitored
- **Password** – Password of the user that can connect to the PostgreSQL server being monitored.
- **Use SSL?** – Whether to connect to the PostgreSQL server using SSL. If using an SSL client certificate, it must be imported into the FglAM's JRE keystore.

Setting Administration Options (optional)

The Administration options allow for certain server operations to be performed on the monitored server from the FMS console. These operations are restricted to Foglight users with the PostgreSQL Administrator role.

- **Enable Administration** – Allows server operations to be performed from the FMS console.
- **Enable Explain Plans** – Allows explain plans to be requested on demand through the FMS console. Does not require that Administration also be enabled.
- **Enable Explain Analyze** – If this is enabled, the Analyze option in the Explain Plan request form will be selectable. The EXPLAIN ANALYZE command will carry out the command on the server and return actual run times and other statistics. Administrators should be very careful when using this option.
- **Admin User** – User with the SUPERUSER privilege that is able to perform the enabled administrative operations. This user will only be used in order to carry out these operations, not for normal data collection.
- **Admin Password** – Password for the admin user.

Setting Collection Intervals (optional)

The Collection Interval fields in the agent properties are used to set the sample frequencies. You can turn off a collection by setting the interval to 0.

- **DB List Refresh (min)** – Checks for databases on the server in order to schedule collections for those databases. As databases are not often added or removed, this does not have to happen frequently. Set to 30 minutes by default.
- **Availability** – A basic heartbeat collection that checks whether the server is running and how long it takes to connect. Should happen frequently. Set to 60 seconds by default.
- **Server and DBs** – Collects data on server metrics and configuration as well as database-level statistics. Set to 300 seconds by default.
- **Tables, Indexes, & Functions** – Collects table, index, and function statistics from each database. Set to 300 seconds by default.

Setting DB Overrides (optional)
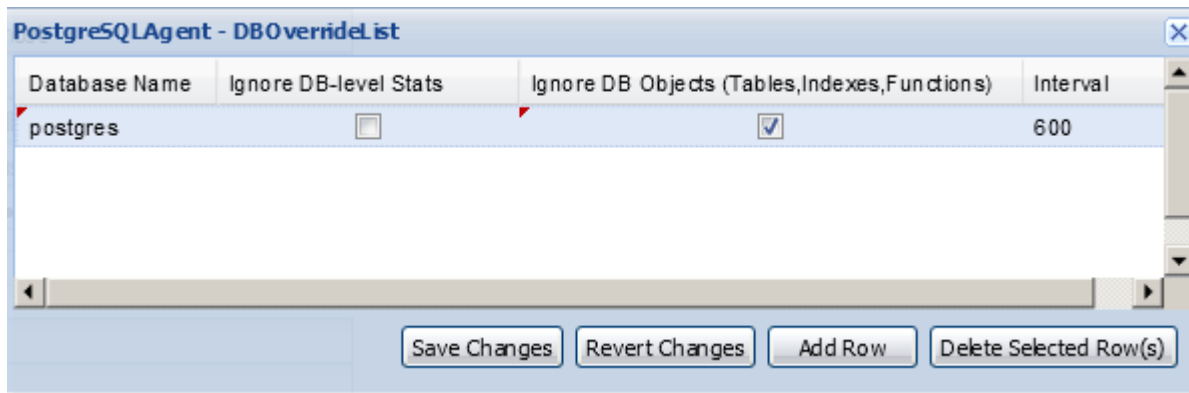
The DB override properties allow you finer-grained control over database collections. If a database is not specified here, it will use the parameters set in the Collection Intervals section.

- **Database Name** – Name of the database where collections are being overridden.
- **Ignore DB-level Stats** – If set to true, database-wide statistics (aggregated from objects belonging to the database) will not be collected for this database.
- **Ignore DB Objects** – If set to true, statistics for individual tables, indexes, and functions belonging to the database will not be collected.
- **Interval** – This collection interval will override the default Tables, Indexes, and Functions setting in the Collection Intervals section for this database.

Statement Tracking (optional)

If the pg_stat_statements module and extension is enabled in the PostgreSQL server, this section can be used to configure statement statistics tracking.

- **Enable Statement Tracking** – Enables or disables use of statement tracking.
- **DB w/ pg_stat_statements view** – While statement tracking is performed across all databases, the pg_stat_statements extension is only available in databases where it has been created. If one or multiple databases have the pg_stat_statements view, you may supply the name of any of them in this field.
- **# of Top Statements** – The number of statements for the agent to collect. If you wish to collect all of them, provide the number of the pg_stat_statements.max setting, or anything higher.
- **Sort By** – The statements are sorted in order to gather the # of top statements specified in the previous property. If all statements are being gathered, this field is unimportant. Otherwise, several self-explanatory options are available based on existing or calculated column data in the view.

Setting Options (optional)

- **Agent Host Name** – The hostname property for the PostgreSQL server topology object is by default set to the connection parameter value defined above. However, you can override it with this setting. If you are monitoring the OS, it is important that the hostname provided by the PostgreSQL agent and OS agent match so that they can be linked.

PostgreSQL Dashboards

The installation of the PostgreSQL Cartridge includes the PostgreSQL Dashboards. The PostgreSQL Cartridge offers four principal dashboards:

- PostgreSQL Servers
- Server Overview
- Databases
- Tablespaces

Drill downs from these dashboards are available to expose more granular data. It should be noted that there is a great deal of additional information that is collected that is not displayed in these primary dashboards. Should an operator wish to have that information displayed, Foglight allows for the creation of simple drag and drop dashboards.

PostgreSQL Servers

Instance				Alarms												
	Name	Host	Version	F	C	W	Availability	Uptime	Conn Time	Connections	Workload	Databases	Tables	Tablespace Size	Statements	Admin
⊗	PostgreSQL_on_WIN2008x64-4	WIN2008x64-4	9.2.4	10	1		Connected	1.4 hr	15.0 ms	1	0.00	2	8	24.5 MB	39	✂
◈	PostgreSQL_91_on_WIN2008x64-4	WIN2008x64-4	9.1.13		1		Connected	15.1 d	16.0 ms	2	0.00	1	0	17.7 MB	48	✂

Overview

The PostgreSQL Servers dashboard displays a row for each PostgreSQL agent instance deployed. This table gives a brief overview of each server's health, availability, response time, and size.

Data

- **Instance**
  - **Health** – Shows the overall health of the deployed instance.
  - **Name** – The name of the PostgreSQL agent. This is the name entered when the PostgreSQL agent instance was created.
  - **Host** – Hostname of the machine where the PostgreSQL server is located.
  - **Version** – The version number of the PostgreSQL server being monitored.

- **Alarms**
  - **Fatal** – Provides a count of the "Fatal" alerts for this agent.
  - **Critical** – Provides a count of the "Critical" alerts for this agent.
  - **Warning** – Provides a count of the "Warning" alerts for this agent.

- **Other Information**
  - **Availability** – Whether the agent is able to connect to the PostgreSQL server.
  - **Uptime** – How long the server has been running for.
  - **Conn Time** – The amount of time needed for the agent to establish a successful JDBC connection to the server.
  - **Connections** – The current number of backends connected to the server.
  - **Workload** – The combined workload of database-level workload metrics, workload is a calculation of time spent reading and writing data file blocks by backends in a sample period, divided by the sample period time. This represents the load on the server and may exceed 1 because of concurrent processing. This is not an estimation of resource usage or availability. Requires track_io_timing to be enabled, available only in v9.2+.
  - **Databases** – The current number of databases contained by the server.
  - **Tables** – The current total number of user tables in the server.
  - **Tablespace Size** – The current size on disk of the server's combined tablespace files.
  - **Statements** – The number of statements gathered in the last sample. This collection may be limited due to the # of Top Statements property in the Agent Properties or the pg_stat_statements.max setting on the PostgreSQL server.

Actions

The PostgreSQL Servers dashboard is conveniently linked to several pages and also contains popup information to provide a natural workflow while investigating server performance. Several fields in the

table have dwell (hovering over the cell value) or drilldown (clicking on the cell value) capability. A list of available interactions with the table is displayed below:

- **Health** (dwell or drilldown) - Shows a breakdown of child objects in a warning, critical, or fatal state.
- **Name** (drilldown) – Navigates to the Server Overview dashboard for the selected server.
- **Host** (drilldown) – Navigates to the Host Monitor page for the selected host. This page will only show data if you are monitoring the OS for this host. For more information on this page, see the related documentation for the infrastructure cartridge.
- **Alarms** (dwell or drilldown) – Displays a list of alarms in the selected severity level.
- **Conn Time** (dwell or drilldown) – A popup time plot of the connection time.
- **Connections**
    - Dwell - A popup time plot of connections.
    - Drilldown – Navigates to the Current Backends page.
- **Workload** (dwell or drilldown) – Displays a popup time plot of workload.
- **Databases** (drilldown) – Navigates to the Databases dashboard.
- **Tables**
    - Dwell - A popup time plot of total tables.
    - Drilldown – Navigates to the Tables page.
- **Tablespace Size**
    - Dwell - A popup time plot of the tablespace size.
    - Drilldown – Navigates to the Tablespaces page.
- **Statements** (drilldown) – Navigates to the Statements dashboard.

<u>Admin Actions</u>

If your user account has the PostgreSQL Administrator role and the PostgreSQL user provided to the agent has the appropriate permissions, you can perform administrative actions on the server from the dashboard. Click the icon in the Admin table column to perform one of the following actions:

- **Statement Stat Reset** – If using the pg_stat_statements module, this action empties the pg_stat_statements view and resets all statement statistics. The extension will then repopulate the view based on new statements. The next sample by the PostgreSQL agent will collect these as normal.

Server Overview
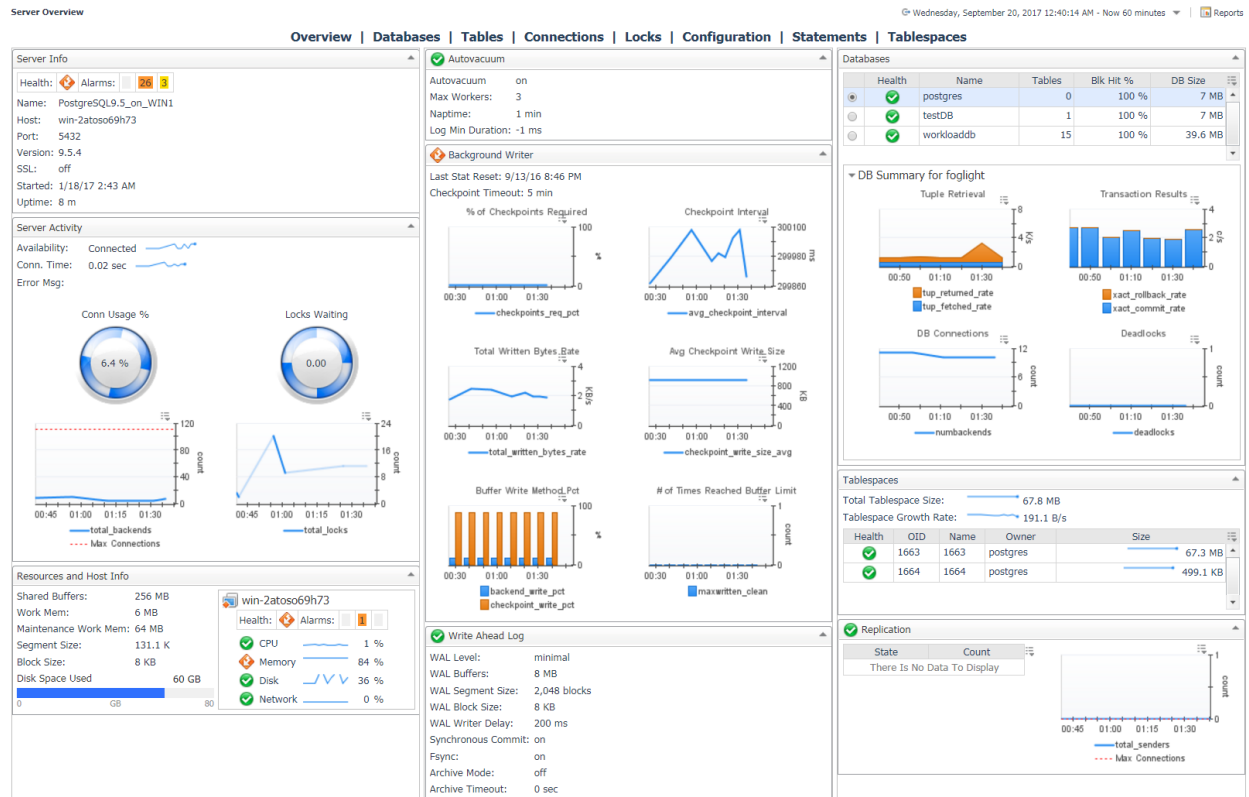
Overview

The Server Overview dashboard gives an overview of all key segments for the PostgreSQL server, including host metrics. In addition, many elements on this dashboard serve as drilldown buttons, allowing users to quickly see potential issues and click through page elements to find more information. In the action panel on the right pane, the PostgreSQL Agent Selector allows you to switch between PostgreSQL servers.

Note: Some PostgreSQL server configuration information available on this page is not shown by default in order to conserve space. This information can be seen by hovering the mouse over the related section. For better readability, an explanation of the data is presented in this document's Data section (found immediately below this one) rather than the Actions section.

Data

Left Column

- **Server Info**
    - **Health** – Shows the overall health of the background writer.
    - **Alarms** – Provides counts of the alarms for this agent, separated by severity.
    - **Name** – Name of the agent monitoring this PostgreSQL server.
    - **Host** – Host where the server is located.
    - **Port** – SQL listening port of the server.
    - **Version** – Version of the server.

- o **Host** – Host where the server is located.
- o **SSL** – Whether the server is configured to accept SSL connections.
- o **Started** – Time of the last server restart.
- o **Uptime** – How long the server has been running since the last restart.
- o **Listen Addr.** – Addresses from which the server will accept connections.
- o **Max Prep. Trans.** – The maximum number of transactions that can be in the "prepared" state simultaneously.
- o **Statement Timeout** – Any statement that takes over the specified number of milliseconds will be aborted, starting from the time the command arrives at the server from the client. A value of zero (the default) turns this off.
- o **Def. Trans. Isolation** – Isolation level applied to each SQL transaction. Can be either "read uncommitted", "read committed", "repeatable read", or "serializable".
- o **Def Trans. Read-Only** – Whether SQL transactions are read-only by default.
- o **Restart After Crash** – Whether server will automatically reinitialize after a backend crash.
- **Server Activity**
  - o **Availability** – Whether agent is able to connect to server with provided credentials.
  - o **Conn. Time** – How much time required to create a new SQL connection.
  - o **Error Msg** – Text of the error message received when failing to create a SQL connection.
  - o **Total Backends** – Shows number of backends connected to the server over specified time range. The red dotted line shows the configured max connections setting, the upper limit for connections to the server.
  - o **Total Locks** – Shows number of locks active on the server.
- **Replication**
  - o **State Table** – Shows current states of all WAL senders and a count of how many are in that state.
  - o **Total Senders** – Shows number of WAL senders active on the server. The red dotted line shows the maximum amount of concurrent connections from standby servers or streaming base backup clients the server is configured to allow.
- **Resources and Host Info**
  - o **Shared Buffers** – The amount of space allocated for memory use by the PostgreSQL server.
  - o **Work Mem** – Amount of memory allocated for internal sorts and hash tables.
  - o **Maintenance Work Mem** – Maximum amount of memory used by maintenance operations like vacuum and create index.
  - o **Segment Size** – The number of blocks that can be stored within a file segment.
  - o **Block Size** – The size of a memory block.
  - o **Disk Space Used** – Shows the total and used disk space detected on the host.
  - o **Host Summary** – Displays health, alarms, and key metric information on the host.

Center Column
- **Autovacuum**
  - o **Autovacuum** – Whether the server is running the autovacuum daemon.
  - o **Max Workers** – The maximum number of autovacuum processes (other than the autovacuum launcher) which may be running at any one time.
  - o **Naptime** –The minimum delay between autovacuum runs on any given database.

- o **Log Min Duration** – If an autovacuum action runs for longer than this setting, it will be logged.
  - o **Analyze Threshold** – The minimum number of inserted, updated or deleted tuples needed to trigger an ANALYZE in any one table.
  - o **Analyze Scale Factor** – The fraction of the table size to add to autovacuum_analyze_threshold when deciding whether to trigger an ANALYZE.
  - o **Vacuum Threshold** – The minimum number of updated or deleted tuples needed to trigger a VACUUM in any one table.
  - o **Vacuum Scale Factor** – The fraction of the table size to add to autovacuum_vacuum_threshold when deciding whether to trigger a VACUUM.
  - o **Cost Delay** – The cost delay value that will be used in automatic VACUUM operations. If -1 is specified, the regular vacuum_cost_delay value will be used.
  - o **Cost Limit** – The cost limit value that will be used in automatic VACUUM operations. If -1 is specified (which is the default), the regular vacuum_cost_limit value will be used.
  - o **Freeze Max Age** – The maximum age (in transactions) that a table's pg_class.relfrozenxid field can attain before a VACUUM operation is forced to prevent transaction ID wraparound within the table.
- **Background Writer**
  - o **Last Stat Reset** – Time when statistic counters for the background writer were last reset.
  - o **Checkpoint Timeout** – Maximum amount of time between automatic WAL checkpoint writes. If checkpoint segment limit is not reached before this time, a checkpoint will be triggered.
  - o **% of Checkpoints Required** - Percent of checkpoints executed that were required. Low percent indicates system has not been active enough to reach WAL segment limit before reaching the checkpoint timeout limit.
  - o **Checkpoint Interval** – Average elapsed time between checkpoints.
  - o **Total Written Bytes Rate** – Rate of total data written into Write Ahead Log.
  - o **Avg Checkpoint Write Size** – Average amount of data written into WAL during checkpoints.
  - o **Buffer Write Method Pct** – Compares percent of buffers written by a backend vs. percent of buffers written during normal checkpoints.
  - o **# of Time Reached Buffer Limit** – Number of times the background writer stopped a cleaning scan because it had written too many buffers.
- **Write Ahead Log**
  - o **WAL Level** – Detail level of information being written to the Write Ahead Log.
  - o **WAL Buffers** – The amount of shared memory used for WAL data that has not yet been written to disk.
  - o **WAL Segment Size** – The number of blocks (pages) in a WAL segment file. The total size of a WAL segment file in bytes is equal to wal_segment_size multiplied by wal_block_size; by default this is 16MB.
  - o **WAL Block Size** – The size of a WAL disk block. It is determined by the value of XLOG_BLCKSZ when building the server.
  - o **WAL Writer Delay** – The delay between activity rounds for the WAL writer
  - o **Synchronous Commit** – Whether transaction commits wait for WAL records to be written to disk before the command returns "success".
  - o **Fsync** – Whether fsync calls are invoked by the server. This ensures updates are physically written to disk.

- o **Archive Mode** – When enabled, completed WAL segments are sent to archive storage.
- o **Archive Timeout** – Since only completed WAL segments are archived, this timeout forces the server to switch to a new WAL segment file periodically if no new segments have been completed.
- o **Checkpoint Completion Target** – Specifies the target of checkpoint completion, as a fraction of total time between checkpoints.
- o **Checkpoint Segments** – Maximum number of log file segments between automatic WAL checkpoints (each segment is normally 16MB).
- o **Checkpoint Timeout** – Maximum time between automatic WAL checkpoints.
- o **Checkpoint Warning** – Writes a message to the server log if checkpoints caused by the filling of checkpoint segment files happen closer together than this many seconds.

Right Column
- • **Databases**
  - o **DB Table**
    - ▪ **Health** – Shows the overall health of the database.
    - ▪ **Name** – The name of the database.
    - ▪ **Tables** – Number of tables in the database.
    - ▪ **Blk Hit %** - Percentage where a table read is successfully sourced from buffer cache rather than needing physical disk I/O.
    - ▪ **DB Size** – Disk size of this database.
  - o **DB Summary**
    - ▪ **Tuple Retrieval** – Rate of tuples retrieved with sequential scans (tup_returned_rate) vs. bitmap or simple scans (tup_fetched_rate).
    - ▪ **Transaction Results** – Rate of transactions finished with commits or rollback.
    - ▪ **DB Connections** – Number of backends connected to the selected database.
    - ▪ **Deadlocks** – Number of deadlocks detected in the database.
- • **Tablespaces**
  - o **Total Tablespace Size** – The combined disk size of all tablespaces used by the server.
  - o **Tablespace Growth Rate** – Rate of tablespace growth.
    - ▪ **Health** – Shows the overall health of the tablespace.
    - ▪ **OID** – OID of the tablespace.
    - ▪ **Name** – Name of the tablespace.
    - ▪ **Owner** – Owner of the tablespace.
    - ▪ **Size** – Disk size of the tablespace.

Actions

- • **Server Info**

  - o **Health** (dwell or drilldown) - Shows a breakdown of child objects in a warning, critical, or fatal state.
  - o **Alarms** (dwell or drilldown) – Displays a list of alarms in the selected severity level.
  - o **Other Data** (dwell) – See additional general server configuration information.
- • **Server Activity**
  - o **Availability** (dwell) – A popup time plot of the server's availability.
  - o **Conn. Time** (dwell) – A popup time plot of the connection time.
  - o **Total Backends** (drilldown) – Navigates to the Current Backends page.

- o **Total Locks** (drilldown) – Navigates to the Current Locks page.
- **Replication**
  - o **Total Senders** (drilldown) – Navigates to the WAL senders page.
- **Resources and Host Info**
  - o **Host Summary**
    - ▪ **Health** (dwell or drilldown) - Shows a breakdown of child objects in a warning, critical, or fatal state.
    - ▪ **Alarms** (dwell or drilldown) – Displays a list of alarms in the selected severity level.
    - ▪ **Metrics**
      - • Dwell – Time plot of metric-related information.
      - • Drilldown – Navigates to a related page in the Infrastructure cartridge.
- **Autovacuum**
  - o **All Data** (dwell) – See additional autovacuum configuration information.
- **Background Writer**
  - o **Graphs** (drilldown) – Navigates to the Background Writer page.
- **Write Ahead Log**
  - o **All Data** (dwell) – See additional WAL configuration information.
- **Databases**
  - o **DB Table**
    - ▪ **Health** (dwell or drilldown) - Shows a breakdown of child objects in a warning, critical, or fatal state.
    - ▪ **Name** (drilldown) – Navigates to the Databases dashboard with the selected DB auto-selected in the dashboard.
    - ▪ **Tables**
      - • Dwell – Popup time plot of the database's table count.
      - • Drilldown – Navigates to the Tables page for the selected database.
    - ▪ **Blk Hit %** (dwell) – A popup time plot of the block hit percentage.
    - ▪ **DB Size** (dwell) – A popup time plot of the database size.
  - o **DB Summary**
    - ▪ **All Graphs** (drilldown) – Navigates to the Databases dashboard with the selected DB auto-selected in the dashboard.
- **Tablespaces**
  - o **Total Tablespace Size** (dwell) – A popup time plot of the total tablespace size.
  - o **Tablespace Growth Rate** (dwell) – A popup time plot of the tablespace growth rate.
  - o **Tablespaces Table**
    - ▪ **Health** (dwell or drilldown) - Shows a breakdown of child objects in a warning, critical, or fatal state.
    - ▪ **Name** (drilldown) – Navigates to the Tablespaces page.
    - ▪ **Size** (dwell) – A popup time plot of the tablespace disk size.

Current Backends

Overview | Databases | Tables | Connections | Locks | Configuration | Statements | Tablespaces

PID	User	App Name	DB	State	State Change	Runtime	Waiting	Query	Query Start	Xact Start	Admin
11308	postgres	pgAdmin III - Browser	postgres	idle	9/13/17 4:04 PM	6.4 d	f	select pg_reload_conf()	9/13/17 4:04 PM	n/a	✗
11364	foguser		postgres	idle	9/20/17 1:41 AM	0 sec	f	SELECT pg_postmaster_start_time() as start_time, EXTRACT('epoch' FROM (now() - pg_postmaster_sta...	9/20/17 1:41 AM	n/a	✗
2236	foguser		workloaddb	idle	9/18/17 3:32 PM	1.4 d	f	SELECT 1	9/18/17 3:32 PM	n/a	✗
3428	postgres	pgAdmin III - Browser	postgres	idle	8/1/17 2:33 PM	1.6 m	f	SELECT t.oid, t.xmin, t.*, relname, CASE WHEN relkind = 'r' THEN TRUE ELSE FALSE END AS parentista...	8/1/17 2:33 PM	n/a	✗
9780	foguser		workloaddb	idle	9/20/17 1:40 AM	27 sec	f	SELECT * FROM testtable14	9/20/17 1:40 AM	n/a	✗

Overview

The Current Backends page shows active backends in the server from the last server data sample and related information. In the action panel on the right pane, the DB Selector allows you to switch between databases or show backends from all databases.

Data

- **PID** – Process ID of the backend.
- **User** – The user name that owns the backend.
- **App Name** – The application where the backend originated from, if provided in the connection string.
- **DB** – Name of the database the backend is connected to.
- **State** – Current overall state of the backend.
- **State Change** – Time when the last time the backend state was changed.
- **Runtime** – How long the backend has existed.
- **Waiting** – Whether or not the backend is waiting for a lock to open in order to perform its query.
- **Wait Event** – Wait event type if the backend is waiting (9.6+).
- **Wait Event Type** – Wait event name if the backend is waiting (9.6+).
- **Query** – Current or last query executed by the backend. Shown in an abbreviated state in order to fit in the table cell.
- **Query Start** – Time when the currently active query was started or start of the last query if not active.
- **Xact Start** – Time when the backend's current transaction was started or null if no transaction is active.

Actions

- **DB** (drilldown) – Navigates to the Databases dashboard with DB selected.
- **Query** (dwell or drilldown) - Shows the full query text in a formatted, colored state for better readability.

Admin Actions

If your user account has the PostgreSQL Administrator role and the PostgreSQL user provided to the agent has the appropriate permissions, you can perform administrative actions on the server from the dashboard. Click the icon in the Admin table column to perform one of the following actions:

- **Cancel** – Cancel the backend's current query.
- **Terminate** – Terminates the backend, i.e. kills the process.

Current Locks

Overview | Databases | Tables | Connections | **Locks** | Configuration | Statements | Tablespaces

Deadlock Timeout	1 sec
Max Locks Per Transaction	64
Max Pred Locks Per Transaction	64

Total Locks

Locks Waiting / Granted

—total_locks

■ locks_waiting
■ locks_granted

PID	Mode	Lock Type	User	Virtual Transaction	Transaction ID	Granted	Lock Age	Nsp Name	Relname	Query	Query Start
3260	AccessShareLock	relation	foguser	9/1191881	n/a	t	0 sec	pg_catalog	pg_shadow	SELECT t.oid as oid, spcname as name, u.usename as owner, pg_tablespace_size(t.oid) as size FROM p...	9/20/17 12:41 AM
3260	AccessShareLock	relation	foguser	9/1191881	n/a	t	0 sec	pg_catalog	pg_user	SELECT t.oid as oid, spcname as name, u.usename as owner, pg_tablespace_size(t.oid) as size FROM p...	9/20/17 12:41 AM
3260	ExclusiveLock	virtualxid	foguser	9/1191881	n/a	t	0 sec	n/a	n/a	SELECT t.oid as oid, spcname as name, u.usename as owner, pg_tablespace_size(t.oid) as size FROM p...	9/20/17 12:41 AM
3260	AccessShareLock	relation	foguser	9/1191881	n/a	t	0 sec	pg_catalog	pg_tablespace_spcname_index	SELECT t.oid as oid, spcname as name, u.usename as owner, pg_tablespace_size(t.oid) as size FROM p...	9/20/17 12:41 AM
3260	AccessShareLock	relation	foguser	9/1191881	n/a	t	0 sec	pg_catalog	pg_authid_oid_index	SELECT t.oid as oid, spcname as name, u.usename as owner, pg_tablespace_size(t.oid) as size FROM p...	9/20/17 12:41 AM
3260	AccessShareLock	relation	foguser	9/1191881	n/a	t	0 sec	pg_catalog	pg_authid_rolname_index	SELECT t.oid as oid, spcname as name, u.usename as owner, pg_tablespace_size(t.oid) as size FROM p...	9/20/17 12:41 AM
3260	AccessShareLock	relation	foguser	9/1191881	n/a	t	0 sec	pg_catalog	pg_tablespace	SELECT t.oid as oid, spcname as name, u.usename as owner, pg_tablespace_size(t.oid) as size FROM p...	9/20/17 12:41 AM
3260	AccessShareLock	relation	foguser	9/1191881	n/a	t	0 sec	pg_catalog	pg_db_role_setting	SELECT t.oid as oid, spcname as name, u.usename as owner, pg_tablespace_size(t.oid) as size FROM p...	9/20/17 12:41 AM
3260	AccessShareLock	relation	foguser	9/1191881	n/a	t	0 sec	pg_catalog	pg_authid	SELECT t.oid as oid, spcname as name, u.usename as owner, pg_tablespace_size(t.oid) as size FROM p...	9/20/17 12:41 AM
3260	AccessShareLock	relation	foguser	9/1191881	n/a	t	0 sec	pg_catalog	pg_db_role_setting_databaseid_rol_index	SELECT t.oid as oid, spcname as name, u.usename as owner, pg_tablespace_size(t.oid) as size FROM p...	9/20/17 12:41 AM
3260	AccessShareLock	relation	foguser	9/1191881	n/a	t	0 sec	pg_catalog	pg_tablespace_oid_index	SELECT t.oid as oid, spcname as name, u.usename as owner, pg_tablespace_size(t.oid) as size FROM p...	9/20/17 12:41 AM

Overview

The Current Locks page shows current locks in the server from the last server data sample and related information. In the action panel on the right pane, the PostgreSQL Agent Selector allows you to switch between PostgreSQL servers.

Data

Top-left Table Section
- **Deadlock Timeout** – The amount of time that the server waits on a lock before checking to see if there is a deadlock condition.
- **Max Locks Per Transaction** – The average number of object locks allocated for each transaction. Individual transactions can lock an infinite amount of objects, but as a whole, the server cannot allow an average of more than this number across all active transactions.
- **Max Pred Locks Per Transaction** - The average number of predicate locks allocated for each transaction. Individual transactions can lock an infinite amount of objects, but as a whole, the server cannot allow an average of more than this number across all active transactions.

Top Graphs
- **Total Locks** – Total locks on the server.
- **Locks Waiting/Granted** – Shows number of locks waiting and granted.

Table
- **PID** – Process ID of the lock.
- **Mode** – Name of the lock mode held or desired by the process.
- **Lock Type** – Type of the lockable object.
- **User** – User controlling the backend that requested the lock.
- **Virtual Transaction** – Virtual ID of the transaction.
- **Transaction ID** – ID of the transaction targeted by the lock or null if target is not a transaction.
- **Granted** – True if lock is held, false if it is waiting to be granted.

- **Lock Age** – How long since lock process started.
- **Nsp Name** – Name of the namespace.
- **Relname** – Name of the object targeted by the lock.
- **Query** – Current or last query executed by the backend. Shown in an abbreviated state in order to fit in the table cell.
- **Query Start** – Time when the currently active query was started or start of the last query if not active.

Actions

- **Query** (dwell or drilldown) - Shows the full query text in a formatted, colored state for better readability.

Statements

Overview | Databases | Tables | Connections | Locks | Configuration | Statements | Tablespaces

Order By: [Average Time ▼] [5] [Apply]

		Statement						Memory Blocks		Shared Blocks
User	DB	Query	Calls	Average ▼	Total	Avg Rows	Write Time	Read Time	Read Time per Query	Read Time per Row	Hit Pct
foguser	testDB	SELECT inhparent as parent, array_agg(inhrelid) as children FROM pg_c...	1	89.23 ms	89.2 ms	0	0 ms	0 ms	0.00 ms	0.00 ms	n/a
foguser	postgres	SELECT t.oid as oid, spcname as name, u.usename as owner, pg_tables...	12	76.72 ms	920.6 ms	2	0 ms	0 ms	0.00 ms	0.00 ms	100.00 %
foguser	workloaddb	SELECT pg_is_in_recovery();	10	69.27 ms	692.7 ms	1	0 ms	0 ms	0.00 ms	0.00 ms	n/a
foguser	postgres	SELECT d.datid as dbid, d.datname as name, t.spcname as tablespaceId...	13	61.68 ms	801.8 ms	3	0 ms	0 ms	0.00 ms	0.00 ms	100.00 %
foguser	testDB	SELECT pg_is_in_recovery();	1	45.54 ms	45.5 ms	1	0 ms	0 ms	0.00 ms	0.00 ms	n/a
foguser	postgres	SELECT *,(buffers_checkpoint+buffers_clean+buffers_backend) as total...	12	9.35 ms	112.1 ms	1	0 ms	0 ms	0.00 ms	0.00 ms	n/a
foguser	postgres	SELECT pg_stat_statements.*, pg_user.usename as usename, pg_datab...	12	6.85 ms	82.2 ms	200	0 ms	0 ms	0.00 ms	0.00 ms	100.00 %
foguser	postgres	SELECT name, setting, unit, category, short_desc, extra_desc, context, ...	100	2.65 ms	264.7 ms	251	0 ms	0 ms	0.00 ms	0.00 ms	n/a
foguser	postgres	SELECT funcid, schemaname, funcname, calls, total_time, self_time FR...	13	0.88 ms	11.4 ms	2	0 ms	0 ms	0.00 ms	0.00 ms	100.00 %
foguser	workloaddb	SELECT * FROM testtable0;	120	0.85 ms	101.8 ms	57.6	0 ms	0 ms	0.00 ms	0.00 ms	100.00 %
foguser	workloaddb	DELETE FROM testtable0 WHERE id%?=?;	14	0.84 ms	11.8 ms	62.9	0 ms	0 ms	0.00 ms	0.00 ms	100.00 %
foguser	workloaddb	DELETE FROM testtable7 WHERE id%?=?;	14	0.83 ms	11.6 ms	76.7	0 ms	0 ms	0.00 ms	0.00 ms	100.00 %
foguser	workloaddb	SELECT * FROM testtable7;	120	0.81 ms	97.6 ms	79.5	0 ms	0 ms	0.00 ms	0.00 ms	100.00 %
foguser	workloaddb	DELETE FROM testtable14 WHERE id%?=?;	14	0.80 ms	11.2 ms	62.6	0 ms	0 ms	0.00 ms	0.00 ms	100.00 %
foguser	workloaddb	DELETE FROM testtable5 WHERE id%?=?;	14	0.80 ms	11.1 ms	75.8	0 ms	0 ms	0.00 ms	0.00 ms	100.00 %

Overview

If Statement Tracking is enabled, the Statements dashboard shows a number of the top statements collected from the server, sorted and limited by fields specified in the Agent Properties. The bar graph can be customized using the fields above it to show top statements by several categories. The graphs below show workload, statement counts, and system utilization metrics (if host is being monitored), in order to allow the user to correlate potentially problematic statements with impact on the system. Below, a table lists all collected statements that have been called in the selected time period. Many of these fields are hidden by default for space considerations, but can be made visible by using the customizer at the top right of the table. In the action panel on the right pane, the PostgreSQL Agent Selector allows you to switch between servers without leaving the page.

Data

Statements Table

- **Statement**
    - **User** – Name of the user that executed the statement.
    - **User ID** – ID of the user that executed the statement.
    - **DB** – Name of the database in which the statement was executed.
    - **DB ID** – ID of the database in which the statement was executed.
    - **Query** – Text of a representative statement (up to track_activity_query_size bytes). Statements with the same operations plan may have slight differences in the original text, but are represented by a single statement.
    - **Calls** – Number of times the statement has been executed.
    - **Rows** - Total number of rows retrieved or affected by the query.
    - **Average** – Average time spent executing the statement per call.

- o **Total** – Total time spent executing statements.
- **Memory Blocks**
  - o **Write Time** - Total time the statement spent writing blocks (if track_io_timing is enabled, otherwise zero).
  - o **Read Time** - Total time the statement spent reading blocks (if track_io_timing is enabled, otherwise zero).
  - o **Read Time per Query** - Time the statement spent reading blocks per query.
  - o **Read Time per Query** - Time the statement spent reading blocks per row retrieved/affected.
- **Shared Blocks**
  - o **Hit Pct** – Percentage of shared block cache hits of total blocks read.
  - o **Read** – Number of shared blocks read.
  - o **Hit** – Number of shared block cache hits.
  - o **Written** – Number of shared blocks written.
  - o **Dirtied** – Number of shared blocks dirtied.
- **Local Blocks**
  - o **Read** – Number of local blocks read.
  - o **Hit** – Number of local block cache hits.
  - o **Written** – Number of local blocks written.
  - o **Dirtied** – Number of local blocks dirtied.
- **Temp Blocks**
  - o **Read** – Number of temp blocks read.
  - o **Written** – Number of temp blocks written.

Actions

Statements Graph
- **Bar Section** (dwell) – Popup of selected statement details.
- **Bar Section** (drilldown) – Navigates to the Statement Details page.

Statements Table
- **DB** (drilldown) – Navigates to the Databases dashboard with DB selected.
- **Query** (drilldown) - Shows the full query text in a formatted, colored state for better readability.
- **Other metrics** (drilldown or dwell) – Shows a popup time plot for the selected metric.

Statement Details
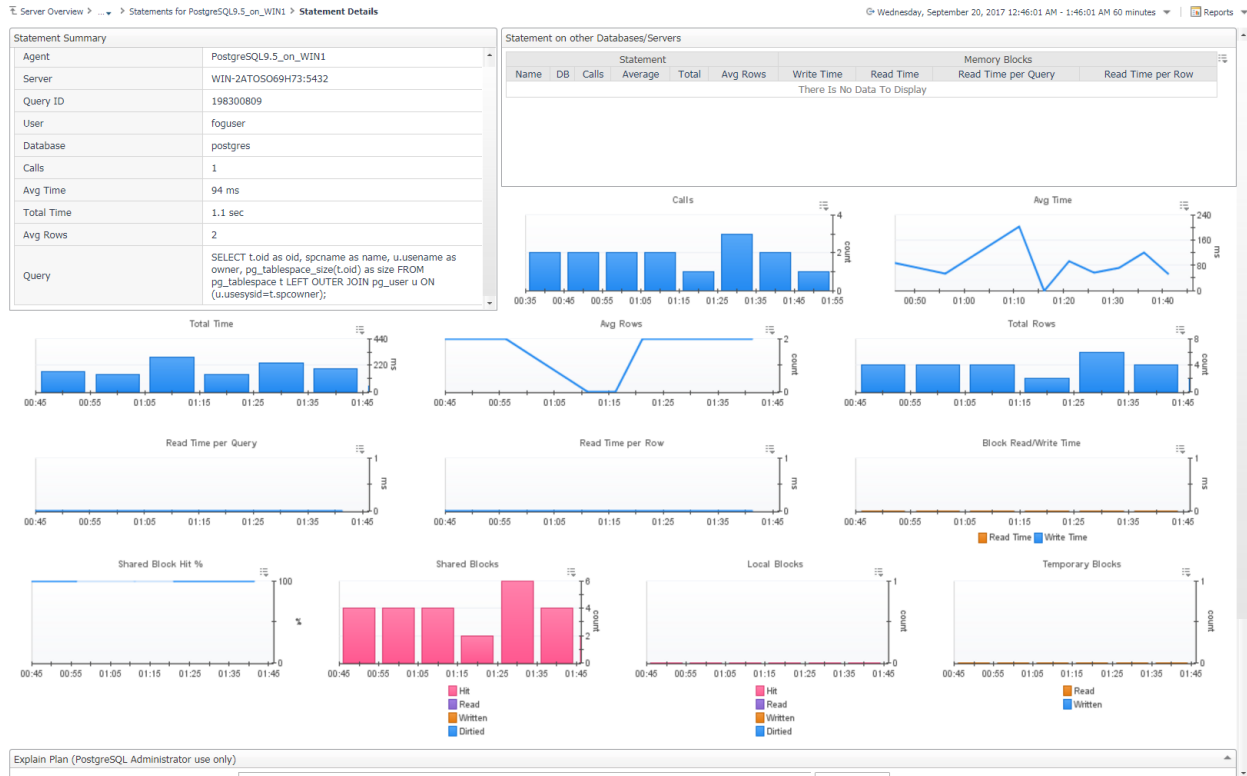
Overview

The Statement Details page provides a more in-depth look at the operations and history of a particular statement as well as its performance on other monitored servers, if the same queryid has been found. This page also features the Explain Plan Request form, which can be used to request an explain plan for the statement if it has been enabled in the agent properties and the Foglight user has the PostgreSQL Administrator role.

Data

- **Statement Summary**
  - **Agent** – Agent monitoring the PostgreSQL Server.
  - **Server** – The host:port of the PostgreSQL server.
  - **Query ID** – An internally-generated hash calculation. For versions below 9.4, where queryid is not available from the server, a hash code has been generated by the agent based solely on a concatenation of the database name and query text
  - **User** – Name of the first user that executed the statement.
  - **Database** – Name of the database in which the statement was executed.
  - **Calls** – Number of times the statement has been executed.
  - **Average Time** – Average time spent executing the statement per call.
  - **Total Time** – Total time spent executing statements.
  - **Avg Rows** – Average number of rows retrieved or affected by the query.

- o **Query** – Text of a representative statement (up to track_activity_query_size bytes). Statements with the same operations plan may have slight differences in the original text, but are represented by a single statement.
- **Statements On Other Servers**
  - o This table shows performance of the same query on other monitored PostgreSQL servers on which it may be performed, for comparison to the selected server.

    In order to match the queries across servers, this table relies on the queryid which is not guaranteed to be the same across servers.

    Per PostgreSQL documentation (https://www.postgresql.org/docs/current/static/pgstatstatements.html):

    "As a rule of thumb, queryid values can be assumed to be stable and comparable only so long as the underlying server version and catalog metadata details stay exactly the same. Two servers participating in replication based on physical WAL replay can be expected to have identical queryid values for the same query. However, logical replication schemes do not promise to keep replicas identical in all relevant details, so queryid will not be a useful identifier for accumulating costs across a set of logical replicas. If in doubt, direct testing is recommended."
- **Other Graphs**
  - o Represent historical views of metrics listed on the Statements page table.

Actions

Statements On Other Servers Table
- **Name** (drilldown) – Switches the page to represent the same statement as performed on the selected server.

Explain Plan Request

If Enable Explain Plans is set to true in the agent properties, this form allows a user with the PostgreSQL Administrator role to request explain plans. The text area field will be pre-filled with the normalized query text of the statement being investigated. It can (or must be) modified to an appropriate form for the PostgreSQL server to consider it a legitimate statement (? replaced with real numerical or string values). The database is also auto-selected, but may be switched.

If Enable Explain Analyze has been set to true in the agent properties, the Analyze option in this form will be selectable. If selected, the EXPLAIN ANALYZE command will carry out the command on the server and return actual run times and other statistics. Administrators should be very careful when using this

option. If unset, the server will not actually run the query and will instead return an expected execution plan.

When done, click the Execute button to request the Explain plan and wait. After a short time, a popup should launch with the results of the explain plan request similar to the screenshot below. The Foglight user and statement performed will also be logged to the Agent Log.
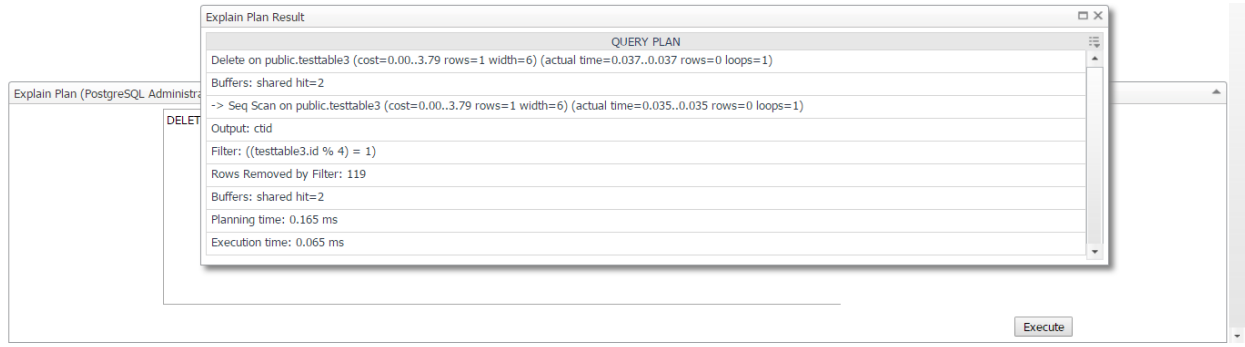
Explain Plan Result

QUERY PLAN
Delete on public.testtable3 (cost=0.00..3.79 rows=1 width=6) (actual time=0.037..0.037 rows=0 loops=1)
Buffers: shared hit=2
-> Seq Scan on public.testtable3 (cost=0.00..3.79 rows=1 width=6) (actual time=0.035..0.035 rows=0 loops=1)
Output: ctid
Filter: ((testtable3.id % 4) = 1)
Rows Removed by Filter: 119
Buffers: shared hit=2
Planning time: 0.165 ms
Execution time: 0.065 ms

Explain Plan (PostgreSQL Administra

DELET

Execute

Replication

PID	App Name	User	User ID	Host	IP	Port	State	Backend Start	Synch State	Sync Priority	Standby Lag	Write Location	Sent Location	Flush Location	Replay Location	Delete
3676	walreceiver	rep_user	24611	WIN2008x64-3	192.168.182.168	49374	streaming	2014-06-16 05:25:45.614-04	async	0	+0	E/50000000	E/50000000	E/50000000	E/50000000	⊖

Overview

The Replication page contains a table that displays one WAL sender per row with configuration and current status information.

Data

- **PID** – The process ID of the WAL sender.
- **App Name** – Name of the application connected to the WAL sender process.
- **User** – Name of the user logged into the WAL sender process
- **User ID** – ID of the user logged into the WAL sender process
- **Host** – Host of the connected WAL receiver client.
- **IP** – IP address of the connected WAL receiver client.
- **Port** – Port of the connected WAL receiver client.
- **State** – Current WAL sender state.
- **Backend Start** – Time when the client first connected to the WAL sender.
- **Synch State** – Synchronous state of the standby server.
- **Sync Priority** – Priority of the current standby server client being chosen as the synchronous standby.
- **Standby Lag** – Number of WAL segments the standby server is behind as compared to the primary server. If this number increases, it means that the standby server is not able to keep up and replication may eventually be broken.
- **Write Location** - Last transaction log position written to disk by this standby server.
- **Sent Location** - Last transaction log position sent on this connection.
- **Flush Location** - Last transaction log position flushed to disk by this standby server.
- **Replay Location** - Last transaction log position replayed into the database on this standby server.

Actions

- **Standby Lag** (dwell) - Shows a time plot of the standby lag.

Admin Actions

Click the icon in the Delete table column to perform the following action:

- **Delete** – Delete the WAL Sender topology object if it becomes unnecessary. This will not affect the PostgreSQL server. Because the topology objects are persisted on the FMS, an object may become redundant or unnecessary if the WAL sender is no longer in use. You can manually delete it with this action. If it is still in use, the object will just be recreated.

Background Writer
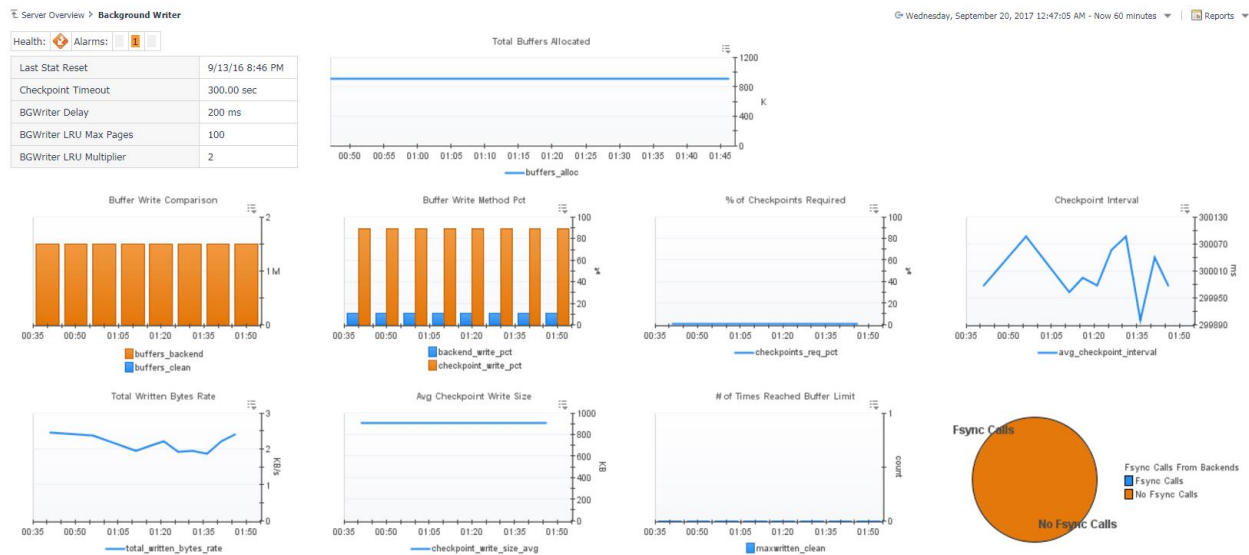
Overview

The Background Writer page shows important configuration information as well as performance metrics related to the background writer.

Data

Top-left Table Section
- **Health** – Shows the overall health of the background writer.
- **Alarms** – Provides counts of the alarms for this agent, separated by severity.
- **Last Stat Reset** – Time when statistic counters for the background writer were last reset.
- **Checkpoint Timeout** – Maximum amount of time between automatic WAL checkpoint writes. If checkpoint segment limit is not reached before this time, a checkpoint will be triggered.
- **BGWriter Delay** – Delay between activity rounds for the background writer.
- **BGWriter LRU Max Pages** – Maximum number of pages written per activity round.
- **BGWriter LRU Multiplier** –Multiplier used to calculate the number of dirty buffers written for a given activity round

Graphs
- **Total Buffers Allocated** – Number of buffers allocated for use by the background writer.
- **Buffer Write Comparison** – Compares number of buffers written by a backend vs. being written by the background writer.
- **Buffer Write Method Pct** – Compares percent of buffers written by a backend vs. percent of buffers written during normal checkpoints.
- **% of Checkpoints Required** - Percent of checkpoints executed that were required. Low percent indicates system has not been active enough to reach WAL segment limit before reaching the checkpoint timeout limit.
- **Checkpoint Interval** – Average elapsed time between checkpoints.

- **Total Written Bytes Rate** – Rate of total data written into Write Ahead Log.
- **Avg Checkpoint Write Size** – Average amount of data written into WAL during checkpoints.
- **# of Time Reached Buffer Limit** – Number of times the background writer stopped a cleaning scan because it had written too many buffers.
- **Fsync Calls From Backends** – Compares number of times a backend had to execute its own fsync call (normally handled by the background writer even when the backend does its own write).
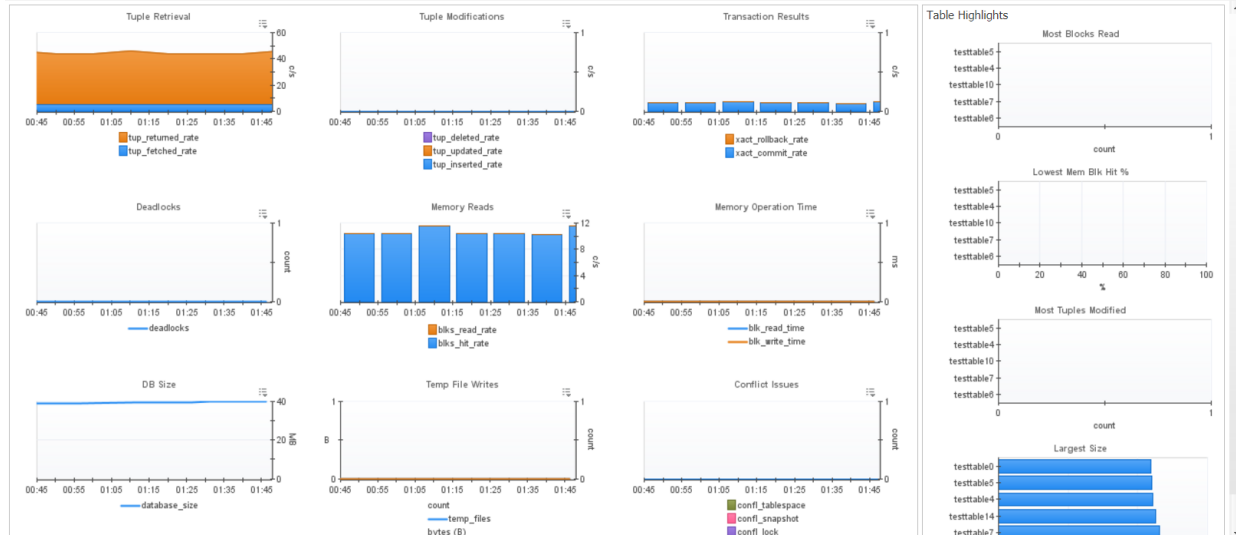
Databases

Overview | Databases | Tables | Connections | Locks | Configuration | Statements | Tablespaces

	Health	Name	Connections	Tables	Functions	Workload		Blk Hit Rate	Blk Read Rate	Blk Hit %	DB Size	Conflicts Rate	Age	Tablespace	Statements	Admin
○	✔	postgres	6	0	3		0.00	20.5 c/s	0 c/s	100 %	7 MB	0.0 c/s	4.6 M	pg_default	13	✖
○	✔	testDB	0	1	0		0.00	9.1 c/s	0 c/s	100 %	7 MB	0.0 c/s	4.6 M	pg_default	5	✖
◉	✔	workloaddb	2	15	0		0.00	10.5 c/s	0 c/s	100 %	39.8 MB	0.0 c/s	4.6 M	pg_default	56	✖

DB Summary for workloaddb - OID 19004 - Stats Reset 5/5/17 3:56 PM

Overview

The Databases dashboard contains 2 sections separated by a horizontal splitter: the top section, which displays a row for each database as well as some key information, and the bottom section, which displays metric information related to the database selected in the top section as well as highlighting tables in that database with the highest or lowest value for a particular metric. The size of the two sections can be increased or decreased by dragging the horizontal splitter bar up or down. In the action panel on the right pane, the PostgreSQL Agent Selector allows you to switch between PostgreSQL servers.

Data

Top Section – DB Table

- **Health** – Shows the overall health of the database.
- **Name** – The name of the database.
- **Connections** – Number of connections to this database.
- **Tables** – Number of tables in the database.
- **Functions** – Number of functions in the database.
- **Workload** – A calculation of time spent reading and writing data file blocks by backends in this database in a sample period, divided by the sample period time. This represents the load on the server and may exceed 1 because of concurrent processing. This is not an estimation of resource usage or availability. Requires track_io_timing to be enabled, available only in v9.2+.

- **Blk Hit Rate** – Rate at which disk blocks are found already in the buffer cache, so that a read was not necessary.
- **Blk Read Rate** – Rate at which disk blocks are read from the database.
- **Blk Hit %** - Percentage where a table read is successfully sourced from buffer cache rather than needing physical disk I/O.
- **DB Size** – Disk size of this database.
- **Conflicts Rate** – Rate of queries cancelled due to conflicts with recovery in this database.
- **Tablespace** – Name of the default tablespace for objects created in this database.
- **Statements** – The number of statements gathered in the last sample for this database. This collection may be limited due to the # of Top Statements property in the Agent Properties or the pg_stat_statements.max setting on the PostgreSQL server.

Bottom Section – DB Metrics
- **Tuple Retrieval** – Rate of tuples retrieved with sequential scans (tup_returned_rate) vs. bitmap or simple scans (tup_fetched_rate).
  - **Tuple Modifications** – Rate of tuples modified by deletion, update, or insert operations.
  - **Transaction Results** – Rate of transactions finished with commits or rollback.
  - **Deadlocks** – Number of deadlocks detected in the database.
  - **Memory Reads** – Rate of memory blocks read from buffer cache vs. physical disk.
  - **Memory Operation Time** – Time spent reading or writing data file blocks by backends.
  - **DB Size** – Disk size of the database.
  - **Temp File Writes** – Number of temporary files created by queries in this database and the total data size.
  - **Conflict Issues** – Breakdown of cancelled queries due to conflict by reason.

Bottom Section – Table Highlights
- **Most Blocks Read** – Tables with the most disk blocks read during queries.
- **Lowest Mem Blk Hit %** - Tables with the lowest memory block hit %, i.e. tables where data is being read from disk rather than buffer cache.
- **Most Tuples Modified** – Tables most actively being modified.
- **Largest Size** – Tables with the largest disk size.

Actions

Database Table

- **Health** (dwell or drilldown) - Shows a breakdown of child objects in a warning, critical, or fatal state.
- **Connections**
  - Dwell - A popup time plot of connections.
  - Drilldown – Navigates to the Current Backends page filtered for selected database.
- **Tables** (drilldown) – Navigates to the Tables page for the selected database.
- **Functions** (drilldown) – Navigates to the Functions page for the selected database.
- **Blk Hit Rate** (dwell) – A popup time plot of the block hit rate.
- **Blk Read Rate** (dwell) – A popup time plot of the block read rate.
- **Blk Hit %** (dwell) – A popup time plot of the block hit percentage.
- **DB Size** (dwell) – A popup time plot of the database size.

- **Conflicts Rate** (dwell) – A popup time plot of the conflicts rate.
- **Tablespace**
  - Dwell - A popup table with the tablespace's health, OI, owner, and disk size.
  - Drilldown – Navigates to the Tablespaces page.
- **Statements** (drilldown) – Navigates to the Statements dashboard, filtered for statements only performed on the selected database.

Table Highlights

Clicking on an area of the graph reserved for displaying the metric value for a table will drilldown to the Tables page for that database with the selected table highlighted.

Admin Actions

If your user account has the PostgreSQL Administrator role and the PostgreSQL user provided to the agent has the appropriate permissions, you can perform administrative actions on the server from the dashboard. Click the icon in the Admin table column to perform one of the following actions:

- **Vacuum + Analyze** – Perform a VACUUM ANALYZE operation on the database.
- **Vacuum** – Perform a VACUUM operation on the database.
- **Analyze** – Perform an ANALYZE operation on the database.
- **Stat Reset** – Reset the statistic counts on the database.

Tables

Overview | Databases | **Tables** | Connections | Locks | Configuration | Statements | Tablespaces

Health	Name	DB	Schema	Tuples	Indexes	Memory Hit %	Idx Mem Hit %	Idx/Seq Scan Tuple %	HOT Update %	Idx Size	Table Size	Idx/Table Size	Age	Owner	Tablespace	Admin
✅	acl_class	foglight	foglight	0	2	0.0 %	0.0 %	0.0 %	0.0 %	16.0 KB	0.0 B	n/a	44.9 M	foglight	pg_default	✕
✅	acl_entry	foglight	foglight	0	4	0.0 %	0.0 %	0.0 %	0.0 %	32.0 KB	0.0 B	n/a	44.9 M	foglight	pg_default	✕
✅	acl_object_identity	foglight	foglight	0	5	0.0 %	0.0 %	0.0 %	0.0 %	40.0 KB	0.0 B	n/a	44.9 M	foglight	pg_default	✕
✅	acl_sid	foglight	foglight	0	2	0.0 %	0.0 %	0.0 %	0.0 %	16.0 KB	0.0 B	n/a	44.9 M	foglight	pg_default	✕
✅	agent_client_defaults	foglight	foglight	0	1	0.0 %	0.0 %	0.0 %	0.0 %	16.0 KB	16.0 KB	100.0 %	44.9 M	foglight	pg_default	✕
✅	agent_config_binder	foglight	foglight	16	3	93.8 %	72.7 %	100.0 %	0.0 %	48.0 KB	48.0 KB	100.0 %	44.9 M	foglight	pg_default	✕
✅	agent_dc_manager_schedule_ids	foglight	foglight	0	2	0.0 %	0.0 %	0.0 %	0.0 %	16.0 KB	0.0 B	n/a	44.9 M	foglight	pg_default	✕
✅	agent_dc_manager_state	foglight	foglight	50	1	0.0 %	0.0 %	0.0 %	0.0 %	16.0 KB	40.0 KB	40.0 %	44.9 M	foglight	pg_default	✕
✅	agent_manager_state	foglight	foglight	86	1	0.0 %	0.0 %	0.0 %	0.0 %	16.0 KB	72.0 KB	22.2 %	44.9 M	foglight	pg_default	✕
⦿ ✅	alarm_alarm	foglight	foglight	283.2 K	7	98.7 %	98.6 %	100.0 %	0.0 %	162.8 MB	173.9 MB	93.6 %	44.9 M	foglight	pg_default	✕
✅	alarm_annotations	foglight	foglight	0	2	0.0 %	0.0 %	0.0 %	0.0 %	32.0 KB	16.0 KB	200.0 %	44.9 M	foglight	pg_default	✕
✅	alarm_loc_msg	foglight	foglight	0	2	0.0 %	100.0 %	0.0 %	0.0 %	16.0 KB	8.0 KB	200.0 %	44.9 M	foglight	pg_default	✕
✅	auditing_log	foglight	foglight	28.1 K	3	0.0 %	0.0 %	0.0 %	0.0 %	4.1 MB	11.2 MB	36.4 %	44.9 M	foglight	pg_default	✕
✅	baseline_config	foglight	foglight	596	1	0.0 %	0.0 %	0.0 %	0.0 %	56.0 KB	192.0 KB	29.2 %	44.9 M	foglight	pg_default	✕
✅	baseline_config_properties	foglight	foglight	1.3 K	2	0.0 %	0.0 %	0.0 %	0.0 %	192.0 KB	144.0 KB	133.3 %	44.9 M	foglight	pg_default	✕

Table Summary for alarm_alarm - OID 16449
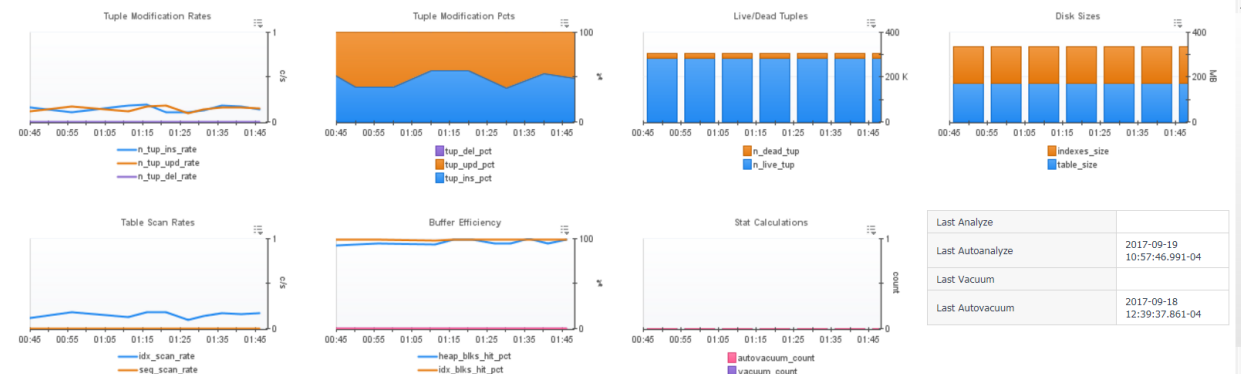
Overview

The Tables page contains 2 sections separated by a horizontal splitter: the top section, which displays a row for each table as well as some key information, and the bottom section, which displays metric information related to the table selected in the top section. The size of the two sections can be increased or decreased by dragging the horizontal splitter bar up or down. In the action panel on the right pane, the DB Selector allows you to switch between databases or show tables from all databases.

Data

Top Section – Tables Table

- **Health** – Shows the overall health of the table.
- **Name** – The name of the table.
- **DB** – The database that contains the table.
- **Schema** – The schema that contains the table.
- **Tuples** – The number of tuples in the table.
- **Indexes** – The number of indexes in the table.
- **Memory Hit %** - Percentage where a read from the table or any related resource is successfully sourced from buffer cache rather than needing physical disk I/O.
- **Idx Mem Hit %** – Percentage where a read from the table indexes is successfully sourced from buffer cache rather than needing physical disk I/O.
- **Idx/Seq Scan Tuple %** – Percentage of tuples returned from index vs. sequential scans.
- **HOT Update %** – Percentage of rows updated by HOT.

- **Idx Size** – Disk size of all index tables associated with the table**.**
- **Table Size** – Disk size of the table.
- **Idx/Table Size** – Ratio of index size to table size.
- **Owner** – Owner of the table.
- **Tablespace** – Tablespace where table data is stored.

Bottom Section – Table Metrics
- **Tuple Modifications Rates** – Rate of tuples modified by deletion, update, or insert operations.
- **Tuple Modification Pcts** – Distribution of modification operations affecting table tuples.
- **Live/Dead Tuples** – Counts of live and dead tuples in table memory. Dead tuples are deleted rows that have not yet been cleaned up.
- **Disk Sizes** – Disk sizes of table, table indexes, and total table size, including related resources.
- **Table Scan Rates** – Rates of index and sequential scans on this table.
- **Buffer Efficiency** – Breakdown of memory block hits by table resource.
- **Stat Calculations** – Counts of vacuum and analyze operations.
- **Last…Operation** – Timestamp of last operation on the table.

Actions

Table Table

- **Health** (dwell or drilldown) - Shows a breakdown of child objects in a warning, critical, or fatal state.
- **DB** (drilldown) – Navigates to the Database page for the selected table.
- **Tuples** (dwell) – A popup time plot of the tuple count.
- **Indexes**
    - Dwell – A popup time plot of the index count.
    - Drilldown – Navigates to the indexes page.
- **Memory Hit %** (dwell or drilldown) - A popup time plot of the memory hit percentage.
- **Idx Mem Hit %** (dwell or drilldown) - A popup time plot of the index memory hit percentage.
- **Idx/Seq Scan Tuple %** (dwell or drilldown) - A popup time plot of the index/sequential scan tuple percent.
- **HOT Update %** (dwell or drilldown) - A popup time plot of the HOT update percent.
- **Idx Size** (dwell or drilldown) - A popup time plot of the index size.
- **Table Size** (dwell or drilldown) - A popup time plot of the table size.
- **Idx/Table Size** (dwell or drilldown) - A popup time plot of the index/table size.
- **Tablespace**
    - Dwell - A popup table with the tablespace's health, OI, owner, and disk size.
    - Drilldown – Navigates to the Tablespaces page.

Admin Actions

If your user account has the PostgreSQL Administrator role and the PostgreSQL user provided to the agent has the appropriate permissions, you can perform administrative actions on the server from the dashboard. Click the icon in the Admin table column to perform one of the following actions:

- **Vacuum + Analyze** – Perform a VACUUM ANALYZE operation on the table.

- **Vacuum** – Perform a VACUUM operation on the table.
- **Analyze** – Perform an ANALYZE operation on the table.
- **Stat Reset** – Reset the statistic counts on the table.

Indexes

Health	Name ▲	Size	isPrimary	isUnique	isValid	Attributes	Avg Tuples	Memory Hit %	Hit Rate	Scan Rate	Tuple Read Rate	Tuple Fetch Rate
✅	alarm_alarm_cl_cre	20.6 MB	f	f	t	2	649.1 count	97.1 %	0.9 c/s	0.0 c/s	6.4 c/s	0.0 c/s
✅	alarm_alarm_cleared_cretd_sev	15.7 MB	f	f	t	3	349.7 count	96.6 %	0.9 c/s	0.0 c/s	0.0 c/s	0.0 c/s
✅	alarm_alarm_created_time	13.8 MB	f	f	t	1	732.9 count	98.7 %	0.9 c/s	0.0 c/s	0.0 c/s	0.0 c/s
✅	alarm_alarm_id	25.4 MB	f	f	t	1	1.0 count	83.9 %	0.8 c/s	0.0 c/s	0.0 c/s	0.0 c/s
✅	alarm_alarm_pkey	13.8 MB	t	t	t	1	1.0 count	99.3 %	1.3 c/s	0.1 c/s	0.1 c/s	0.1 c/s
✅	alarm_alarm_rule_id	41.8 MB	f	f	t	1	35.3 count	97.0 %	1.2 c/s	0.0 c/s	0.0 c/s	0.0 c/s
✅	alarm_alarm_topid_iscleared	31.5 MB	f	f	t	2	358.5 count	92.2 %	0.9 c/s	0.0 c/s	0.0 c/s	0.0 c/s

Overview

The Indexes page displays key information on indexes for the selected database table as well as providing comparisons between multiple table indexes in the four graphs at the top of the page.

Data

Top Section – Index Comparisons

- **Indexes v Table Size Comparison** – Compares disk size of table vs. combined table indexes.
- **Index sizes** – Compares disk sizes of table indexes.
- **Memory Hit %** - Compares memory hit percentage of indexes.
- **Average Tuples Returned** – Average number of tuples returned per query.
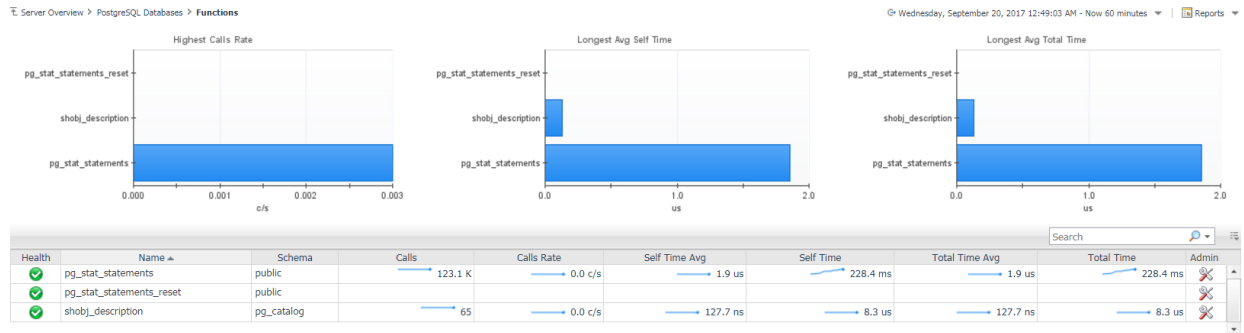
Bottom Section – Index Table

- **Health** – Shows the overall health of the index.
- **Name** – The name of the index.
- **Size** – Disk size of the index.
- **isPrimary** – Whether index is the primary index in the table.
- **isUnique** – Whether index values must be unique.
- **isValid** – Whether index is configured correctly and can be used as an index.
- **Attributes** – The number of columns used in the index.
- **Avg Tuples** – Average number of tuples returned from an index scan.
- **Memory Hit %** - Percentage where a read from the index is successfully sourced from buffer cache rather than needing physical disk I/O.
- **Hit Rate** – Rate of buffer hits in this index.
- **Scan Rate** – Rate of index scans initiated on the index.
- **Tuple Read Rate** – Number of index entries returned by scans on the index.
- **Tuple Fetch Rate** – Number of live table rows fetched by simple index scans using the index.

Actions

Index Table

- **Health** (dwell or drilldown) - Shows a breakdown of child objects in a warning, critical, or fatal state.
- **Size** (dwell) - A popup time plot of the index size.
- **Attributes** (dwell) – A popup table of the columns used in the index.
- **Avg Tuples** – A popup time plot of the average number of tuples returned.
- **Memory Hit %** - A popup time plot of the memory hit percentage.
- **Hit Rate** – A popup time plot of the hit rate.
- **Scan Rate** – A popup time plot of the scan rate.
- **Tuple Read Rate** – A popup time plot of the tuple read rate.
- **Tuple Fetch Rate** – A popup time plot of the tuple fetch rate.

Functions

Health	Name ▲	Schema	Calls	Calls Rate	Self Time Avg	Self Time	Total Time Avg	Total Time	Admin
✓	pg_stat_statements	public	123.1 K	0.0 c/s	1.9 us	228.4 ms	1.9 us	228.4 ms	✂
✓	pg_stat_statements_reset	public							✂
✓	shobj_description	pg_catalog	65	0.0 c/s	127.7 ns	8.3 us	127.7 ns	8.3 us	✂

Overview

The Functions page displays statistics on functions for the selected database as well as providing comparisons between functions in the three graphs at the top of the page. Tracking of function call counts and timing requires that track_functions is enabled.

Data

Top Section – Function Comparisons
- **Highest Calls Rate** – Compares call rates of most frequently called functions.
- **Longest Avg Self Time** – Compares self time for functions with highest average time.
- **Longest Avg Total Time** – Compares total time for functions with highest average total time.

Bottom Section – Functions Table
- **Health** – Shows the overall health of the function.
- **OID** – OID of the function.
- **Name** – The name of the function.
- **Schema** – Schema that owns the function.
- **Calls** – Number of calls to the function.
- **Calls Rate** – Rate at which the function is called.
- **Self Time Avg** – Average time spent in the function, not including calls to other functions.
- **Self Time** – Total time spent in the function, not including calls to other functions.
- **Total Time Avg** – Average time spent in the function, including calls to other functions.
- **Total Time** – Total time spent in the function, including calls to other functions.

Actions

Function Table

- **Health** (dwell or drilldown) - Shows a breakdown of child objects in a warning, critical, or fatal state.
- **Calls** – A popup time plot of the number of calls.
- **Calls Rate** – A popup time plot of the calls rate.
- **Self Time Avg** – A popup time plot of the average self time.
- **Self Time** – A popup time plot of the self time.
- **Total Time Avg** – A popup time plot of the average total time.

- **Total Time** – A popup time plot of the total time.
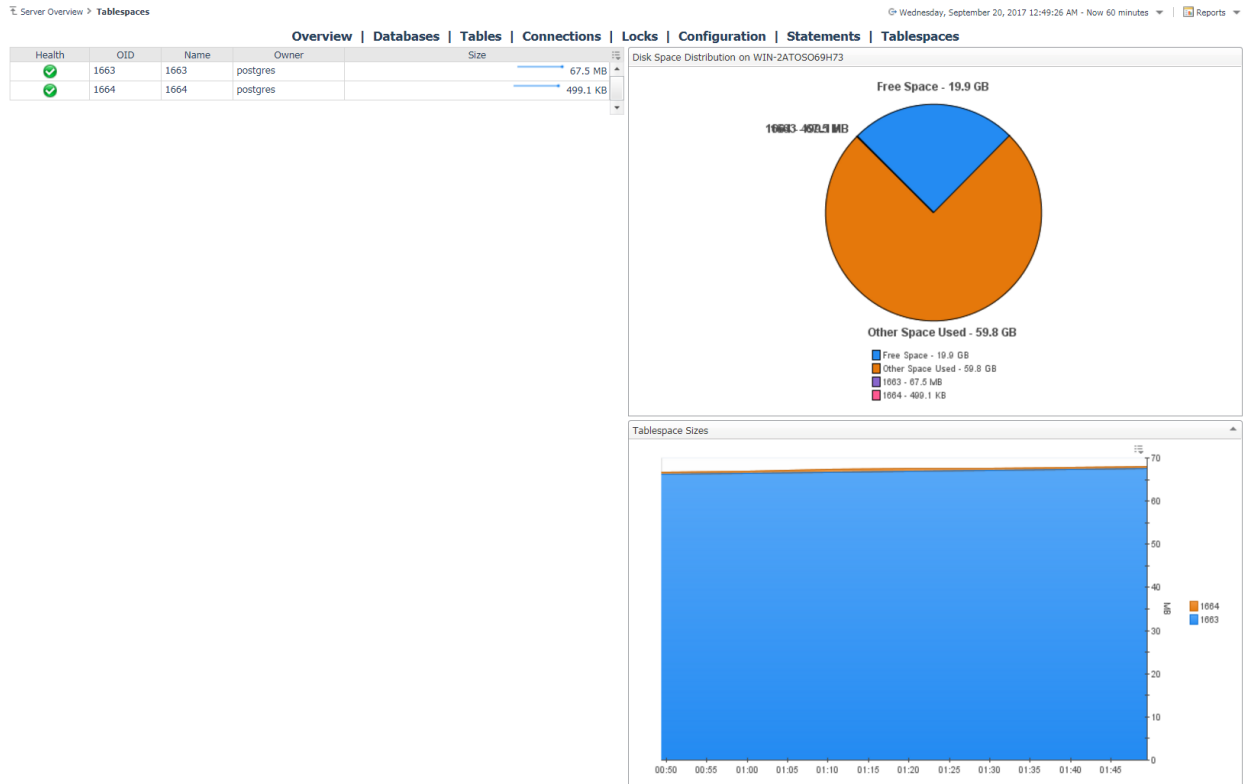
Admin Actions

If your user account has the PostgreSQL Administrator role and the PostgreSQL user provided to the agent has the appropriate permissions, you can perform administrative actions on the server from the dashboard. Click the icon in the Admin table column to perform one of the following actions:

- **Stat Reset** – Resets statistics for the selected function.

Tablespaces

Overview | Databases | Tables | Connections | Locks | Configuration | Statements | Tablespaces

Health	OID	Name	Owner	Size	
✓	1663	1663	postgres		67.5 MB
✓	1664	1664	postgres		499.1 KB

Disk Space Distribution on WIN-2ATOSO69H73

Free Space - 19.9 GB

1663 - 67.5 MB

Other Space Used - 59.8 GB

- Free Space - 19.9 GB
- Other Space Used - 59.8 GB
- 1663 - 67.5 MB
- 1664 - 499.1 KB

Tablespace Sizes

- 1664
- 1663

Overview

The Tablespaces page shows tablespaces used by the selected PostgreSQL server. The table on the left displays a row for each tablespace while the graphs on the right show a visual distribution of disk spaze. In the action panel on the right pane, the PostgreSQL Agent Selector allows you to switch between PostgreSQL servers.

Data

Table

- **Health** – Shows the overall health of the tablespace.
- **OID** – OID of the tablespace.
- **Name** – Name of the tablespace.
- **Owner** – Owner of the tablespace.
- **Size** – Disk size of the tablespace.

Graphs

- **Disk Space Distribution** – Shows the distribution of disk space on the host between tablespaces, other used space, and free space. If host storage information is not being collected, only tablespace sizes are shown.
- **Tablespace Sizes** – Compares tablespace sizes to each other and shows combined total space used.

<u>Actions</u>

- **Health** (dwell or drilldown) - Shows a breakdown of child objects in a warning, critical, or fatal state.
- **Size** (dwell) – A popup time plot of the disk size.

PostgreSQL Reports

PostgreSQL Executive Summary – Executive summary of PostgreSQL instance with high-level information including connection, workload, availability, and performance metrics, as well as top 10 alarms by severity.

PostgreSQL Health Check – Report showing health indicators for the PostgreSQL instance, including backends, workload, system, and performance metrics, top 10 statement digests by response time and top 10 alarms by severity.

PostgreSQL Server Configuration – Displays variables for the PostgreSQL instance with values at the start and end of the selected time range and when they were modified.

PostgreSQL Server Configuration Comparison – Compares current variable values for all PostgreSQL instances contained in a Service against a selected PostgreSQL instance (template server) to ensure configuration compliance. The report will optionally return all variables or just the variables with at least one difference in values.

PostgreSQL Top Servers by Connections – Displays top *X* PostgreSQL instances by the selected connection metric: availability percent, connection time, total backends, or percent used.

PostgreSQL Top Statements – Displays top *X* statement digests for a PostgreSQL instance by the selected metric: calls, average/total/max time, average rows, or shared blocks hit percent.

PostgreSQL Top Tables – Displays top *X* tables in a PostgreSQL instance by the selected table metric: tuples, indexes, memory hit percent, index memory hit percent, index size, or table size.

PostgreSQL Cartridge Rules

Approaching Connection Limit

Severity	Expression	Message

Fatal	def connUtil = (#total_backends#/scope.monitoringAgent.max _connections) * 100;
	
	if(connUtil>registry("connUtilFatal")){return true;}
		The connection utilization for this server is at @connUtil%. Your max_connections setting is @maxConnections. You may wish to increase this setting in order to ensure there are enough connections available.

Critical	def connUtil = (#total_backends#/scope.monitoringAgent.max _connections) * 100;
	
	if(connUtil>registry("connUtilCrit")){return true;}
		The connection utilization for this server is at @connUtil%. Your max_connections setting is @maxConnections. You may wish to increase this setting in order to ensure there are enough connections available.

Warning	def connUtil = (#total_backends#/scope.monitoringAgent.max _connections) * 100;
	
	if(connUtil>registry("connUtilWarn")){return true;}
		The connection utilization for this server is at @connUtil%. Your max_connections setting is @maxConnections. You may wish to increase this setting in order to ensure there are enough connections available.

Default Registry Variables: connUtilWarn = 85%, connUtilCrit = 95%, connUtilFatal=99%

Backend Write Percentage High

Severity	Expression	Message
Fatal	#backend_write _pct#>registry(" backendWriteFa tal")	A high backend write percentage means that backends are writing their own WAL buffers to disk rather than waiting for a checkpoint to do this. This may happen normally if your application performs many bulk insert statements. Otherwise, you may need to increase your background writer activity or increase your WAL Buffer allocation size.
Critical	#backend_write _pct#>registry(" backendWriteCri t")	A high backend write percentage means that backends are writing their own WAL buffers to disk rather than waiting for a checkpoint to do this. This may happen normally if your application performs many bulk insert statements. Otherwise, you may need to increase your background writer activity or increase your WAL Buffer allocation size.
Warning	#backend_write _pct#>registry(" backendWriteW arn")	A high backend write percentage means that backends are writing their own WAL buffers to disk rather than waiting for a checkpoint to do this. This may happen normally if your application performs many bulk insert statements. Otherwise, you may need to increase your background writer

		activity or increase your WAL Buffer allocation size.

Default Registry Variables: backendWriteWarn = 60%, backendWriteCrit = 75%, backendWriteFatal = 90%

DB Running Out of Disk Space

Severity	Expression	Message	

Fatal	def ds = server.DataService;
	
	def free;
	try{
	free = ds.retrieveLatestValue(scope.monitoredHost.storage,"spaceAvailable").value.avg;
	}
	catch(Exception e){return;}
	if(free==null){return;} //Must be collected
	
	def endTime = new java.sql.Timestamp(System.currentTimeMillis());
	def startTime = endTime.minus(7).toTimestamp();
	
	def metricStart = ds.retrieveEarliestTime(scope,"tablespace_growth_rate");
	
	if(metricStart.after(startTime)){return;} //Metric must be collected for longer than the start time for a representative average
	
	def growthRate = ds.retrieveAggregate(scope,"tablespace_growth_rate",startTime,endTime).value.avg;
	if(growthRate<=0){return;}
	
	//Convert to expected daily growth in MB
	def dayGrowth = (growthRate*60*60*24)/(1024*1024);
	
	//Divide free space by day growth
	def daysToFull = free / dayGrowth;
	
	if(daysToFull < registry("dbCapacityFatal")){return true;}
		If the average growth of the tablespaces of this database server over the last week continues at its current rate, it is predicted that the DB host will run out of disk space in @daysToFull day(s).	

Critical	def ds = server.DataService;
	
	def free;
	try{
	free = ds.retrieveLatestValue(scope.monitoredHost.storage,"spaceAvail
		If the average growth of the tablespaces of this database server over the last week continues at its current rate, it is predicted that the DB	

	able").value.avg;
	}
	catch(Exception e){return;}
	if(free==null){return;} //Must be collected
	
	def endTime = new
	java.sql.Timestamp(System.currentTimeMillis());
	def startTime = endTime.minus(7).toTimestamp();
	
	def metricStart =
	ds.retrieveEarliestTime(scope,"tablespace_growth_rate");
	
	if(metricStart.after(startTime)){return;} //Metric must be collected
	for longer than the start time for a representative average
	
	def growthRate =
	ds.retrieveAggregate(scope,"tablespace_growth_rate",startTime,
	endTime).value.avg;
	if(growthRate<=0){return;}
	
	//Convert to expected daily growth in MB
	def dayGrowth = (growthRate*60*60*24)/(1024*1024);
	
	//Divide free space by day growth
	def daysToFull = free / dayGrowth;
	
	if(daysToFull < registry("dbCapacityCrit")){return true;}
		host will run out of disk
		space in @daysToFull
		day(s).

Warning	def ds = server.DataService;
	
	def free;
	try{
	free =
	ds.retrieveLatestValue(scope.monitoredHost.storage,"spaceAvail
	able").value.avg;
	}
	catch(Exception e){return;}
	if(free==null){return;} //Must be collected
	
	def endTime = new
	java.sql.Timestamp(System.currentTimeMillis());
	def startTime = endTime.minus(7).toTimestamp();
	
	def metricStart =
	ds.retrieveEarliestTime(scope,"tablespace_growth_rate");
	
	if(metricStart.after(startTime)){return;} //Metric must be collected
		If the average growth of
		the tablespaces of this
		database server over the
		last week continues at
		its current rate, it is
		predicted that the DB
		host will run out of disk
		space in @daysToFull
		day(s).

| | for longer than the start time for a representative average

def growthRate =
ds.retrieveAggregate(scope,"tablespace_growth_rate",startTime,
endTime).value.avg;
if(growthRate<=0){return;}

//Convert to expected daily growth in MB
def dayGrowth = (growthRate*60*60*24)/(1024*1024);

//Divide free space by day growth
def daysToFull = free / dayGrowth;

if(daysToFull < registry("dbCapacityWarn")){return true;} | |

Default Registry Variables: dbCapacityWarn = 60 days, dbCapacityCrit = 30 days, dbCapacityFatal = 15 days

High Percentage of Queries Waiting

Severity	Expression	Message

Fatal	def conList = #Current_Backends#;
	
	int total = #Current_Backends#.size()
	if(total==0){return false;}
	int waitNum = 0;
	for(conItem in conList)
	{
	if(conItem.waiting=="t"){waitNum++;}
	}
	
	def waitPct = 100*(waitNum/total);
	if(waitPct>registry("queryWaitingFatal")){return true;}
		In the last 3 out of 5 evaluations, the percentage of queries in a waiting state have exceeded this severity threshold. The percentage of queries in a waiting state when this alarm was triggered is @waitPct%.

Critical	def conList = #Current_Backends#;
	
	int total = #Current_Backends#.size()
	if(total==0){return false;}
	int waitNum = 0;
	for(conItem in conList)
	{
	if(conItem.waiting=="t"){waitNum++;}
	}
	
	def waitPct = 100*(waitNum/total);
		In the last 3 out of 5 evaluations, the percentage of queries in a waiting state have exceeded this severity threshold. The percentage of queries in a waiting state when this alarm was triggered is @waitPct%.

Severity	Expression	Message
	if(waitPct>registry("queryWaitingCrit")){return true;}	

Warning	def conList = #Current_Backends#;
	
	int total = #Current_Backends#.size()
	if(total==0){return false;}
	int waitNum = 0;
	for(conItem in conList)
	{
	if(conItem.waiting=="t"){waitNum++;}
	}
	
	def waitPct = 100*(waitNum/total);
	if(waitPct>registry("queryWaitingWarn")){return true;}
		In the last 3 out of 5 evaluations, the percentage of queries in a waiting state have exceeded this severity threshold. The percentage of queries in a waiting state when this alarm was triggered is @waitPct%.

Default Registry Variables: queryWaitingWarn = 10%, queryWaitingCrit = 20%, queryWaitingFatal = 30%

Index Bloat

Severity	Expression	Message
Fatal	#idx_size_pct#>registry("indexBloatFatal")	The index size for this table is currently @idxPct% of the total table size. You may want to ensure the index table is being vacuumed in order to compact its size.
Critical	#idx_size_pct#>registry("indexBloatCrit")	The index size for this table is currently @idxPct% of the total table size. You may want to ensure the index table is being vacuumed in order to compact its size.
Warning	#idx_size_pct#>registry("indexBloatWarn")	The index size for this table is currently @idxPct% of the total table size. You may want to ensure the index table is being vacuumed in order to compact its size.

Default Registry Variables: indexBloatWarn = 20%, indexBloatCrit = 25%, indexBloatFatal = 40%

Low Buffer Hit Percentage – Database

Severity	Expression	Message
Fatal	if(#blks_hit_pct#<registry("bufferHitFatal")){return true;}	Buffer hit percentage for this object has been low for the last three consecutive evaluations. A higher hit percentage means that tuples are being returned from pages in memory rather than being read from disk, which consumes more system resources. The current buffer hit

		percentage is @hitPct%
Critical	if(#blks_hit_pct#<registry("bufferHitCrit")){return true;}	Buffer hit percentage for this object has been low for the last three consecutive evaluations. A higher hit percentage means that tuples are being returned from pages in memory rather than being read from disk, which consumes more system resources. The current buffer hit percentage is @hitPct%
Warning	if(#blks_hit_pct#<registry("bufferHitCrit")){return true;}	Buffer hit percentage for this object has been low for the last three consecutive evaluations. A higher hit percentage means that tuples are being returned from pages in memory rather than being read from disk, which consumes more system resources. The current buffer hit percentage is @hitPct%

Default Registry Variables: bufferHitWarn = 75%, bufferHitCrit = 50%, bufferHitFatal = 25%

Low Buffer Hit Percentage – Table

Severity	Expression	Message
Fatal	if(#total_blks_hit_pct#<registry("bufferHitFatal")){return true;}	Buffer hit percentage for this object has been low for the last three consecutive evaluations. A higher hit percentage means that tuples are being returned from pages in memory rather than being read from disk, which consumes more system resources. The current buffer hit percentage is @hitPct%
Critical	if(#total_blks_hit_pct#<registry("bufferHitCrit")){return true;}	Buffer hit percentage for this object has been low for the last three consecutive evaluations. A higher hit percentage means that tuples are being returned from pages in memory rather than being read from disk, which consumes more system resources. The current buffer hit percentage is @hitPct%
Warning	if(#total_blks_hit_pct#<registry("bufferHitWarn")){return true;}	Buffer hit percentage for this object has been low for the last three consecutive evaluations. A higher hit percentage means that tuples are being returned from pages in memory rather than being read from disk, which consumes more system resources. The current buffer hit percentage is @hitPct%

Default Registry Variables: bufferHitWarn = 75%, bufferHitCrit = 50%, bufferHitFatal = 25%

Low Percent of Checkpoints Required

Severity	Expression	Message
Critical	#checkpoints_req_pct#<registry("checkReqCrit")	A low percentage of checkpoints that are being performed are required, i.e. the checkpoint_timeout limit has been reached before the checkpoint_segments limit. This may indicate that your settings should be optimized for the level of activity normally seen on this DB server. A lower segment limit can also mean shorter crash recovery time if that is a concern.
Warning	#checkpoints_req_pct#<registry("checkReqWarn")	A low percentage of checkpoints that are being performed are required, i.e. the checkpoint_timeout limit has been reached before the checkpoint_segments limit. This may indicate that your settings should be optimized for the level of activity normally seen on this DB server. A lower segment limit can also mean shorter crash recovery time if that is a concern.

Default Registry Variables: checkReqWarn = 35%, checkReqCrit = 15%

PostgreSQL Server Availability

Severity	Expression	Message
Fatal	if(#availability#.equals("Not Available")){return true;}	if(#availability#.equals("Not Available")){return true;}

Potential Deadlock Issue

Severity	Expression	Message

Critical	def locksWaiting = #locks_waiting#;
	def timeout = scope.deadlock_timeout/1000;
	
	if(locksWaiting==0){return false;}
	
	for(def lockRow in scope.Current_Locks.current.value)
	{
	 if(lockRow.granted=="f"){
	  if(lockRow.lock_age>timeout){return true;}
		You may have a potential deadlock issue. Currently, there are @lockNum locks that have not been granted and are older than the deadlock timeout setting of @timeout

	}def locksWaiting = #locks_waiting#;
	def timeout = scope.deadlock_timeout/1000;
	
	if(locksWaiting==0){return false;}
	
	for(def lockRow in scope.Current_Locks.current.value)
	{
	 if(lockRow.granted=="f"){
	  if(lockRow.lock_age>timeout){return true;}
	 }
	}
		milliseconds. The oldest waiting lock age is @maxAge seconds and the average age is @avgAge seconds.

Replication Lag

Severity	Expression	Message
Critical	#standby_log_diff# >=registry("repLag Crit")	The difference in WAL segments between your primary's sent_location and standby server's replay_location has exceeded this threshold. If the standby server continues to fall behind, this may cause problems, especially if the primary server goes down in a high-availability situation.
Warning	#standby_log_diff# >=registry("repLag Warn")	The difference in WAL segments between your primary's sent_location and standby server's replay_location has exceeded this threshold. If the standby server continues to fall behind, this may cause problems, especially if the primary server goes down in a high-availability situation.

Default Registry Variables: repLagWarn = 5, repLagCrit = 10

Slow Connection

Severity	Expression	Message
Fatal	#connection_time#>registr y("connTimeFatal")	The agent's last two connections to this PostgreSQL server have exceeded this rule's threshold for connection time. The latest connection time was @connTime seconds.
Critical	#connection_time#>registr y("connTimeCrit")	The agent's last two connections to this PostgreSQL server have exceeded this rule's threshold for connection time. The latest connection time was @connTime seconds.
Warning	#connection_time#>registr y("connTimeWarn")	The agent's last two connections to this PostgreSQL server have exceeded this rule's threshold for connection time. The latest connection time was @connTime

		seconds.

Default Registry Variables: connTimeWarn = 1.5, connTimeCrit = 3, connTimeFatal = 5