

**Specification**

**for**

**8/6/10**



Proprietary and Confidential to BBS Technologies, Inc.

**© 2006 BBS Technologies, Inc.; all rights reserved.**

# Revision history:

|  |  |  |
| --- | --- | --- |
| **When?** | **Who?** | **What?** |
| 8/19/2010 | Vicky | Initial Draft |
| 8/23/2010 | Vicky | Updated with review notes |
| 8/27/2010 | Mike and Vicky | Updated with implementation details |

# Table of Contents

1. Revision history: ii

2. Table of Contents iii

3. Requirements 1

3.1. Overview/Purpose 1

3.1.1. Related Customer Requests 1

3.2. Target Users 1

3.3. Feature/Function Market Requirements 2

3.3.1. Required Functions 2

3.3.2. Non-Supported Functions 2

3.4. FAQ 2

3.5. Open Issues 3

4. Functional Design 3

4.1. User Interfaces 3

4.1.1. Queries View 3

4.1.2. Query History View 5

4.1.3. Query History View 6

4.2. Installation and Upgrade 6

4.3. Permissions and other Required Configuration 6

5. Internal Design 6

5.1. Architecture 6

5.1.1. Collection Service 6

5.1.2. Management Service 7

5.1.3. Repository 7

5.1.4. Reports 7

5.2. Installation Issues 7

5.3. Schedule 8

5.3.1. Work Breakdown and Sizings 8

5.3.2. Areas of Risk 8

6. Quality Assurance Considerations 8

6.1.1. Overview 8

6.1.2. Developer-Created Unit Tests 9

7. Documentation Considerations 9

8. Bibliography 9

# Requirements

## Overview/Purpose

The primary goal of this feature is to make the Queries view more usable for users. By improving this view we can make the feature more useful to our current users and capture new users who have heretofore been unable to make good use of the view. The current Queries view suffers from a number of problems which limit its usefulness:

* The large mass amount of queries shown in a list is almost impossible to sift through
* Users really have no way of truly finding the Top 5-10 worst performing queries
* Our filters are hidden
* The view is not interesting – just a data grid
* The view shows nothing if the Query Monitor is not running or not working
* The reports are not a good substitute because of limited filters
* Both the views and the reports are slow

This will be addressed by introducing interesting new data graphs and by limiting data grids to only the most pertinent information. Another innovation in this version will be the introduction of a universal identifier for both a query statement and its signature. With this improvement it will become possible to correlate and allow drill-down from disparate views into the Queries view in a way we have not been able to provide in the past.

A secondary goal of this feature is to reduce the amount of data gathered by the Query Monitor and improve the efficiency of its storage. This is a large topic but not a compelling one: while it is necessary to improve our storage mechanism from a performance standpoint, this change alone will not “sell” the feature. The primary problems that need to be addressed from a performance standpoint are:

* The amount of data we are storing is excessive
* We have more than we can possibly use and therefore cannot use any
* The table structure is very flat: easy to work with but slow
* The size of the Query Monitor data causes problems during upgrade and grooming
* The SQL\_Signature function has outlived its usefulness: it is slow and has irritating bugs

### Related Customer Requests

14665 - 'p\_GetQueryMonitorStatements is really expensive.'  
14348 - FRQ: Reinstate the Advanced filters in the Worst Performing Queries reports13268 – Queries view cause SQLdm Console to hand and use 50% of CPU

## Target Users

Through the improvements to this feature we hope to capture the attention of users who previously have found the Queries view too cumbersome or confusing to use.

## Feature/Function Market Requirements

### Required Functions

* Provide a more graphical user interface for the Queries view, with charts to supplement the data grid
* Bring in data from sources other than the traditional Query Monitor trace
  + Query Waits
  + Blocking Chains
  + Deadlocks
  + Plan Reuse
* Allow users to browse worst performing on several axes of performance
  + CPU
  + Reads
  + Writes
  + Waits
  + Deadlocks
  + Blocking
  + Rate of CPU
  + Rate of IO
  + Plan reuse
* Allow users to group worst performing on several identifiers
  + SQL text
  + SQL signature
  + Application
  + User
  + Host
* Improve the performance of the Queries view
  + Calculate the query signature at collection time
  + Streamline the repository storage structure
  + Improve grooming
  + Add aggregation logic to reduce data being stored
* Simplify filtering and bring it into the foreground instead of hiding it in a dialog box
* Allow a user to see the performance of a query over time

### Non-Supported Functions

* For the purposes of this specification, no additional tracing will be added to capture short-duration, high-frequency queries. This should generally be covered by Query Wait Monitoring.
* This feature presents an opportunity to expand into continuous blocking monitoring. However, there are implications for other views (specifically the Blocking view) as well as alerting. Therefore this probably belongs in its own detailed spec.

## FAQ

None at present.

## Open Issues

Continuous Blocking Monitoring (see 3.3.2) – need a spec?

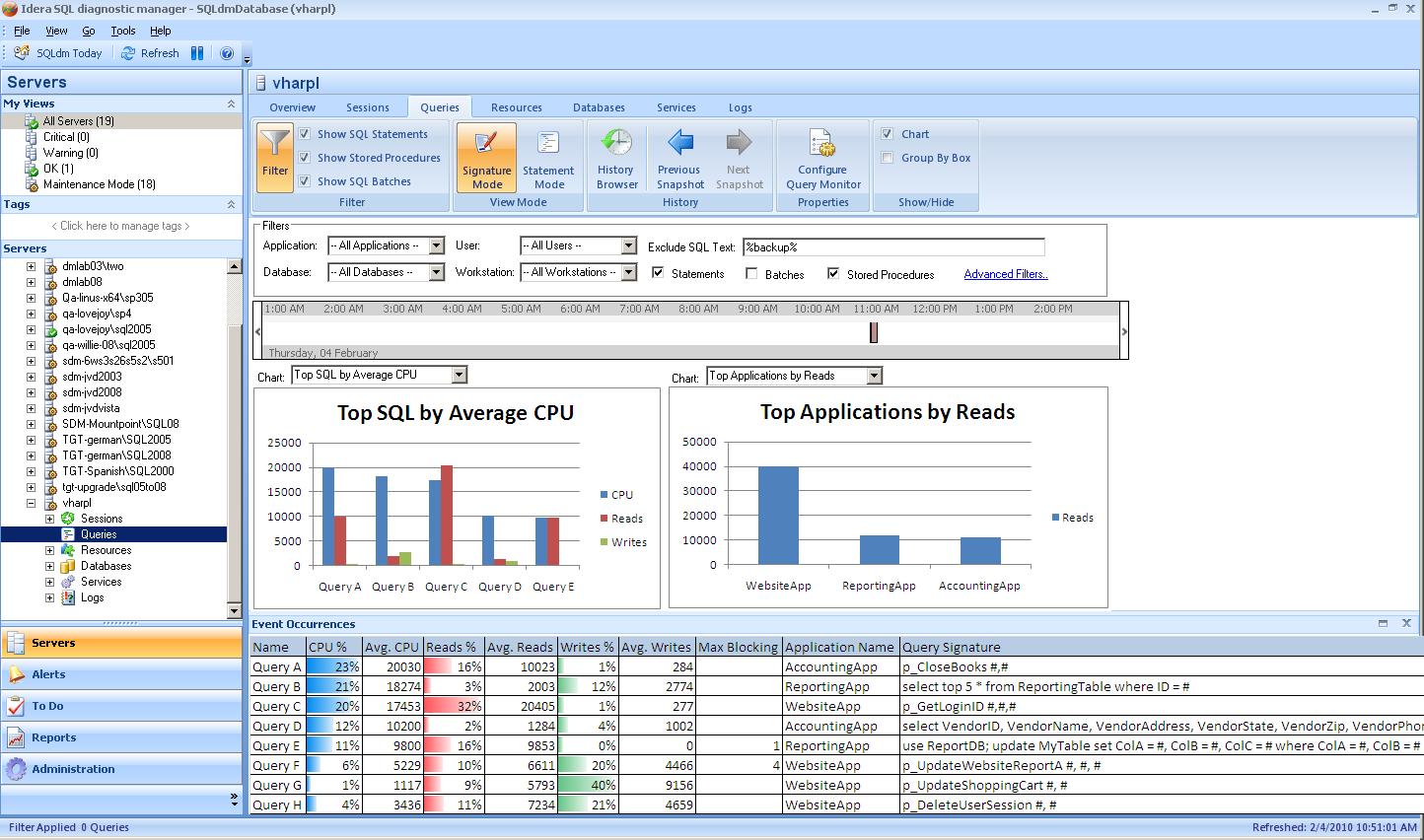
Do we support statement mode for the Query History view? (see 4.1.2)

What is the default query to show in the Query History view if it was not drilled in to? (see 4.1.2)

# Functional Design

## User Interfaces

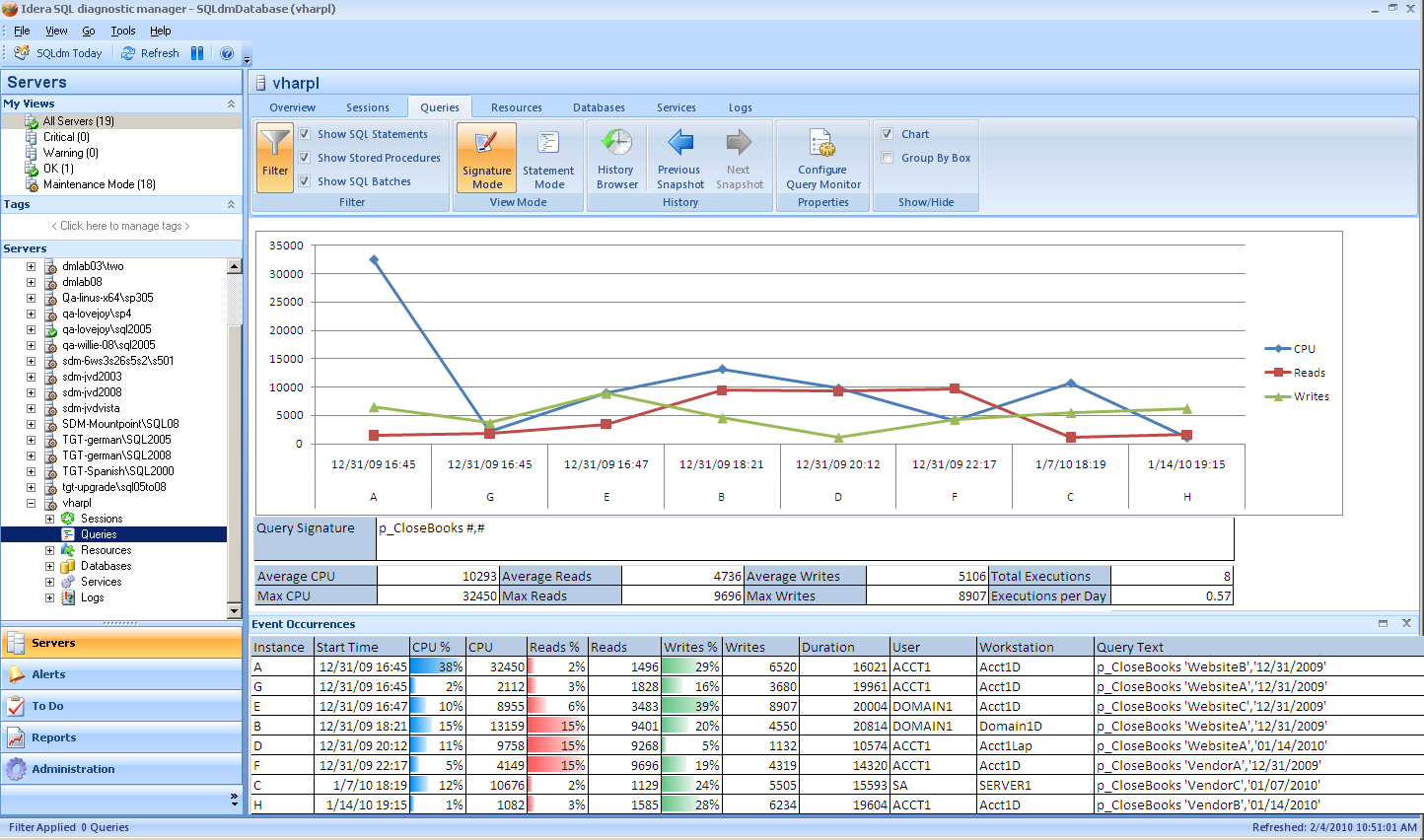
### Queries View



The main Queries view will be altered as follows:

* The most common filters for the view will be clearly located across the top in a fixed sized box unless minimized as in the reports views. By clicking “Advanced Filters” the user may access all filter criteria that are in the current filter dialog. Clicking the Filter button in the ribbon will bring up Advanced Filters. Filter values will all be text boxes as in the dialog and not drop downs as shown in the image because of issues with volume and changing values on refresh or in different history snapshots. Filter needs a “reset/clear” button. Filters that don’t apply in Signature Mode will be grayed out or hidden in the filter area just as the columns are in the grid.
  + Filters that don’t apply in Signature Mode include User and Workstation. The spid column also does not apply in signature mode, though there is not a filter for it.
* The view will feature a time selection area similar to the one in SQLsafe that will allow the user to more easily visualize the filtered timeframe. The time selector stands in for the old sart and end time filters.
* The upper pane will now consist of two large, clear “Top 5” graphs over the period selected. The user will be able to select from each of the following graphs:
  + CPU
  + Reads
  + Writes
  + Waits
  + Deadlocking
  + Blocking
  + Plan Reuse
  + CPU Per Second
  + IO Per Second
* Each of the graphs will offer the ability to view the data grouped by any of the following in a second selection box over each chart. Values will be dependent on Mode:
  + SQL (Text in Statement Mode or Signature in Signature Mode)
  + ApplicationDatabase
  + User
  + Host
* Each series on the graph will allow drill-down. In the case of a series that represents individual queries, the navigation will apply a filter to the Queries view (for instance, clicking on the top application by reads will filter the Queries view to show Top SQL by Average Reads filtered on that single application).
* The lower pane will be a sortable, groupable grid view which is pre-sorted to match the initial leftmost graph if not loaded from a persisted value. Changing the chart will not affect the grid sort thereafter. The CPU, Reads, and Writes will be displayed as percentages of the filtered set for ease in isolating the most critical queries. Drill-through on any query will go to the Query Statistics view. This lower pane should be collapsible and there should be a splitter between the upper and lower panes.
* The grid will have a selection for the number of queries to return that defaults to 100 in the title bar. This value should be persisted across sessions.
* The user will be able to switch between Signature and Statement mode in the filters region of the screen.
* Query Names will be assigned to each query for identification between the grid and charts and maintained for the duration of the session to prevent confusion with them changing through refresh or as sort orders change in the charts or grid. The values will be unique within each instance. They will not be saved or persisted across sessions.
* The view should have a “SQL Include” filter in addition to the SQL Exclude filter shown in the prototype.
* The view needs to attempt to show relevant data based on the Query Monitor settings, available data, and filter settings. For instance, when a user filters by Application it makes no sense to show a Top Applications chart. In this case, the next selection in the list will be chosen as it will when switching back to Signature Mode and being current on a chart that no longer makes sense.
* All filter columns applicable to the selected mode will be available in the column chooser on the grid.
* The chart selections in each position should persist across sessions for each Mode just like the grid settings do currently. The selected Mode should be persisted as well.
* The view also needs to inform the user why data is not available: is it in the repository or not upgraded, is it not available in the repository, is the Query Monitor not enabled?
* Need to be able to change the “Top X” number – 5 by default – 10 is max. Will try to put this over each chart as well if there is room. Otherwise it will go in the ribbon for all charts.

### Query History View

 A secondary Query History view will be provided for drill-down purposes from both the Queries view and from other views. Ideally whenever there is query text being shown it should be possible to navigate through to this view. For this release I suggest we add drill-throughs from Queries, Sessions, Locks, and Query Waits.

The upper pane will show a history of the runtimes of the query and their CPU, IO, and duration for each run. Information on the average and maximum CPU, IO, duration, and execution frequency will be shown below the graph, along with the query signature and an example of the query with parameters still in place (this latter part is not in the image above). I suggest we only support signature mode for this chart because most statements are not going to repeat frequently enough to provide a chart – open for discussion.

The lower pane will show the detailed and aggregated history of specific runs for the query. Detailed history will only be available for a few days before it is aggregated, but the data will be combined into the same grid. Example (with a subset of columns):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Instance | Start | Executions | Avg Duration | Avg CPU | Workstation |
| A | 8/18/2010 11:28 PM | 1 | 6520 | 2210 | ACCT1 |
| B | 8/18/2010 10:15 PM | 1 | 6320 | 2100 | ACCT2 |
| C | 8/1/2010 | 2 | 6100 | 1975 | ACCT1 |
| D | 7/25/2010 | 2 | 6075 | 1980 | Multiple Workstations |

From this view the user should also be able to choose to “Keep detailed history for this query” and flag the current signature to prevent the aggregation job from affecting it (the data will still be groomed). This should be possible to un-flag as well.

Open issue: What should be the default query in this view?

Though not present in the mock-up, the same timeline selector from the Queries view will be available in this view for time selection.

### Grooming View

The grooming view will need to be updated to allow configuration of how many days of detailed queries information to maintain and to allow a “every X hours” configuration of the query aggregation job. The last run time and status of the query aggregation job will need to be displayed.

### Query Monitor Configuration Properties

* Need to make the More Options button more consistent in name and location with the Wait Monitoring Filter Options… button.

## Installation and Upgrade

Upgrading old Query Monitor data will be a protracted process and will undertaken incrementally in the days following upgrade. Further technical details are available in section 5.2.

## Permissions and other Required Configuration

No special considerations. No changes will be made to application security rules for query monitor configuration or for data viewing.

# Internal Design

## Architecture

### Collection Service

No significant change will be made to the batches for tracing as part of this feature. There should be no impact on the monitored servers.

Service-side trace processing will be modified as follows:

* The signature of each query monitor event will be calculated at collection time, rather than the former practice of calculating the signature in the repository only when requested.
  + The method used to create signatures in SQLdoctor will be used to simplify future integration
* Each query monitor statement will have a hash created from both the signature and raw event strings in order to facilitate easy statement aggregation in all parts of the product. This is already being done in Query Waits.

### Management Service

The management service will manage the insertion of signature and raw queries into the new repository. Furthermore, the management service will need to store non-serialized blocking and deadlock records so that they may be included in the new Queries view.

As described in 5.2 below, the management service will also upgrade old data from the respository into the new data structure.

### Repository

The repository will undergo a major overhaul with respect to Query Monitor storage. Currently all Query Monitor data is stored in one table, with a foreign key for the SQL Server ID and the Database ID being the only normalized fields. In 6.2 a new normalized table structure was introduced to store Query Wait data and this structure will be extended to absorb Query Monitor data.

The existing QueryMonitor table and the associated SQL\_Signature function will be left in place for 1 version in order to allow old data to be gradually moved to the new format (see 5.2).

The repository will also be modified to allow the separate tracking of individual query executions and aggregated histories for a given signature. This means that as data grows older it will become less granular: I propose that after 14 days the data is aggregated up into 1-day buckets. We had initially discussed 7 days but with 14 days there will be better insight into weekly activity. As this is configurable it can be changed to 7 days in environments where it is warranted.

New tables will be added for Blocking Queries and Deadlocks, though the old tables will remain in place to support existing functionality.

There will be a significant modification to the grooming job, which will need to groom both the existing QueryMonitor table and the accumulated new tables. A second aggregation job will be added to move data into the aggregation tables: if during implementation this proves to be overkill the job may instead be added to the existing grooming job, but the plan is for them to be separate.

The stored procedure that returns data to the Query Monitor view will be expected to provide the correct columns for signature versus statement mode and also provide a reason if no data is found. The procedure will detect whether the upgrade is in progress or whether the Query Monitor is currently disabled.

The upgrade will need some sort of limiter to prevent performance problems.

### Reports

The existing reports will be upgraded to support the new data structure and any necessary changes will be made to support the data aggregation functionality, but no new reports and no new functionality to existing reports is planned.

## Installation Issues

The tremendous changes to the repository will require an extended upgrade process for past Query Monitor data. In order to avoid a protracted upgrade process, the Management Service will incrementally upgrade data and insert it into the new tables, working on a small (TBD based on performance) number of rows per run of the job and moving back in time until all old data is either upgraded or groomed out. The details of this process will be solidified once the new table structure is in place but broadly speaking it will work similarly to the old 4.x Metric Repository Service. While the upgrade is in progress a notice should be visible on the Queries view to explain to the user that not all data may be ready for viewing.

## Schedule

### Work Breakdown and Sizings

Insert areas that put the implementation or schedule at risk – Assumptions made in design, new areas that have a learning curve etc. Break the project into logical components and add time for design, coding, integration, unit testing. These estimates should not take into account outside forces like maintenance work. These will be applied in building the overall project schedule. A general rule of thumb is to break any unit of work greater then 1 week into smaller pieces to ensure accuracy and to allow measurement of progress during the project.

|  |  |  |
| --- | --- | --- |
| Component | Who | Sizing |
| Desktop Client | Michael (10) Vicky(1) | 11 days |
| Services Work | Vicky | 5 days |
| Repository Work | Vicky | 4 days |
| Grooming Job and Upgrade Changes | Vicky | 2 days |
| Report Upgrades | Anyone | 2 days |
| Unit and Integration Testing | Michael (1.5) Vicky (1.5) | 3 days |
| **Total** |  | **27 days** |

### Areas of Risk

The primary area of risk is the change to repository structure and the upgrade logic that goes with that. The UI being entirely new is also risky. The data collection and processing changes are low-risk.

# Quality Assurance Considerations

### Overview

* It would be prudent to compare the speed and repository size of 6.1 versus 6.5 for this feature, probably by monitoring the same workload with both versions.
* In evaluating this feature it will be important to create a varied workload for tracing: the behavior of the view will not be apparent and will not emulate the user experience if we are using the same few worst performing queries over and over. Furthermore, if there are only a handful of signatures it could cause the repository changes to appear to be far more effective than they might really be.
  + This is conceptually the same as the Query Waits testing
* The upgrade process will need to be thoroughly vetted to ensure that it does not cause problematic locking or blocking, especially on servers which are upgrading large amounts of data. The performance of the management service, especially with notification and alerting, should not be affected.

### Developer-Created Unit Tests

None.

# Documentation Considerations

This is a significant departure from the old view so existing documentation will be made obsolete for the Queries view.

The upgrade process may need to be documented in the release notes so that users understand why their previously-collected data is not necessarily available right away after upgrade.

# Bibliography