****

**Specification**

**for**

**02/05/2010**

IderaLogo

Proprietary and Confidential to BBS Technologies, Inc.

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

# Revision history:

|  |  |  |
| --- | --- | --- |
| **When?** | **Who?** | **What?** |
| 2/5/2010 | Vicky Harp | First draft |
| 2/10/2010 | Vicky Harp | Incorporating comments before review |
| 2/24/2010 | Vicky Harp | Included revisions from review |

# 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

4.2. Installation and Upgrade 5

4.3. Permissions and other Required Configuration 5

5. Internal Design 6

5.1. Architecture 6

5.1.1. Collection Service 6

5.1.2. Management Service 6

5.1.3. Repository 6

5.1.4. Desktop Client 8

5.1.5. Reports 8

5.2. Installation Issues 8

5.3. Schedule 8

5.3.1. Work Breakdown and Sizings 8

5.3.2. Areas of Risk 8

6. Quality Assurance Considerations 9

6.1.1. Overview 9

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

* Improve the performance of the Queries view from an end-user perspective
* Bubble up the ‘worst performers’ by resource usage, blockers/blocked, deadlock, # executions, waits, etc.
* Bubble up our filters to allow more graphical selection of timeframe (date and even time of day), instance, database, etc.
* Incorporate non-Query Monitor data into the Queries view so that we are not entirely dependent on the trace
* Improve the storage structure of the Query Monitor data so that the repository does not grow unmanageably large

### Non-Supported Functions

* For the purposes of this specification, no additional tracing will be added to capture short-duration, high-frequency queries.
* While Wait Statistics are of interest to incorporate into this view, the research on Wait Monitoring is ongoing and thus this is not covered in this specification.
* While this spec references long blocking queries, this feature does not have anything to do with adding a blocking trace to the Query Monitor. Blocked queries will be detected in the same fashion as previous versions.
* No alerts will be added for this feature.

## FAQ

Desktop Client

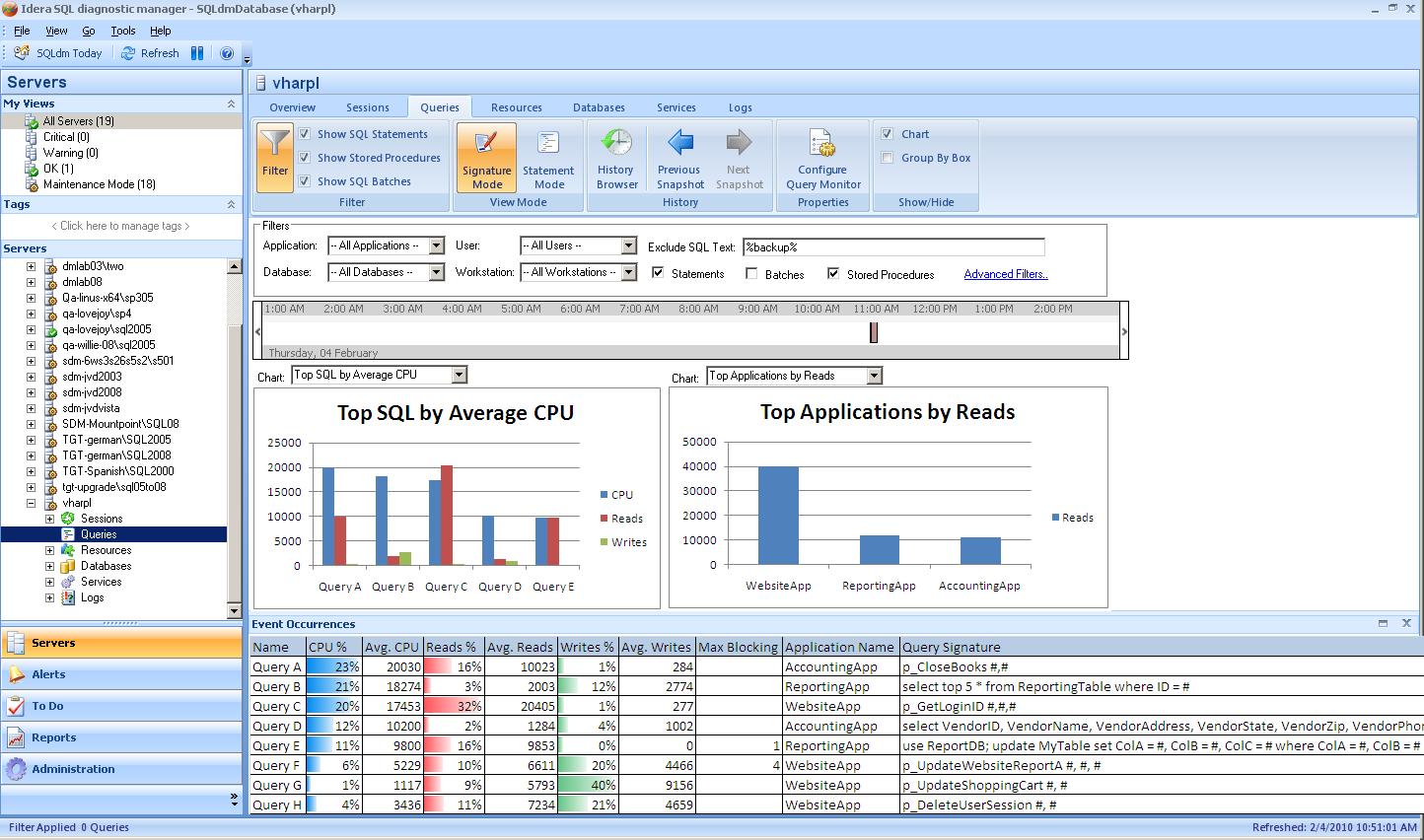
* What is our default view ? What view will be shown if no query monitor data is being collected? Can we detect that?
  + The default view is open for discussion: the two graphs in the prototype are not a bad starting point.
  + We can easily detect whether the Query Monitor is running. If no query monitor data is being collected I would recommend showing blocking data along with a message about how more data would be available if the Query Monitor were running. If the Query Monitor is running and there’s just no data available that should be noted as well. (Common on first install and upgrade )
* Once we have a specific query is it possible to see where the ‘duration’ was spent? IOW, where is waited the most?
  + Not with the data we currently collect. We also do not presently know of any way in which to collect this without very frequent refreshes.

## Open Issues

* Is it possible to link this view to the new Wait Statistics view, and if so, what would be required of this feature to facilitate that linkage? Resolved – No link will be made
* Aggregation Times – Would we like to aggregate queries into hourly or 10-minute periods? Resolved - Hourly
* Detail Retention – How long should full details be retained before aggregation? Can this be modified? See 5.1.3 under QueryStatements. Resolved – Default will be 3 days, and this can be changed in the Grooming dialog
* Should the Query Monitor aggregation and grooming job be separated from the main grooming job? My feeling is yes, as ideally it should run more often than the regular grooming job, possibly every hour. Resolves - yes
* Are new alerts necessary with this feature? Resolved - No
* Are new reports necessary with this feature? Resolved – No new reports. We will make the filters in the existing reports better match the view.

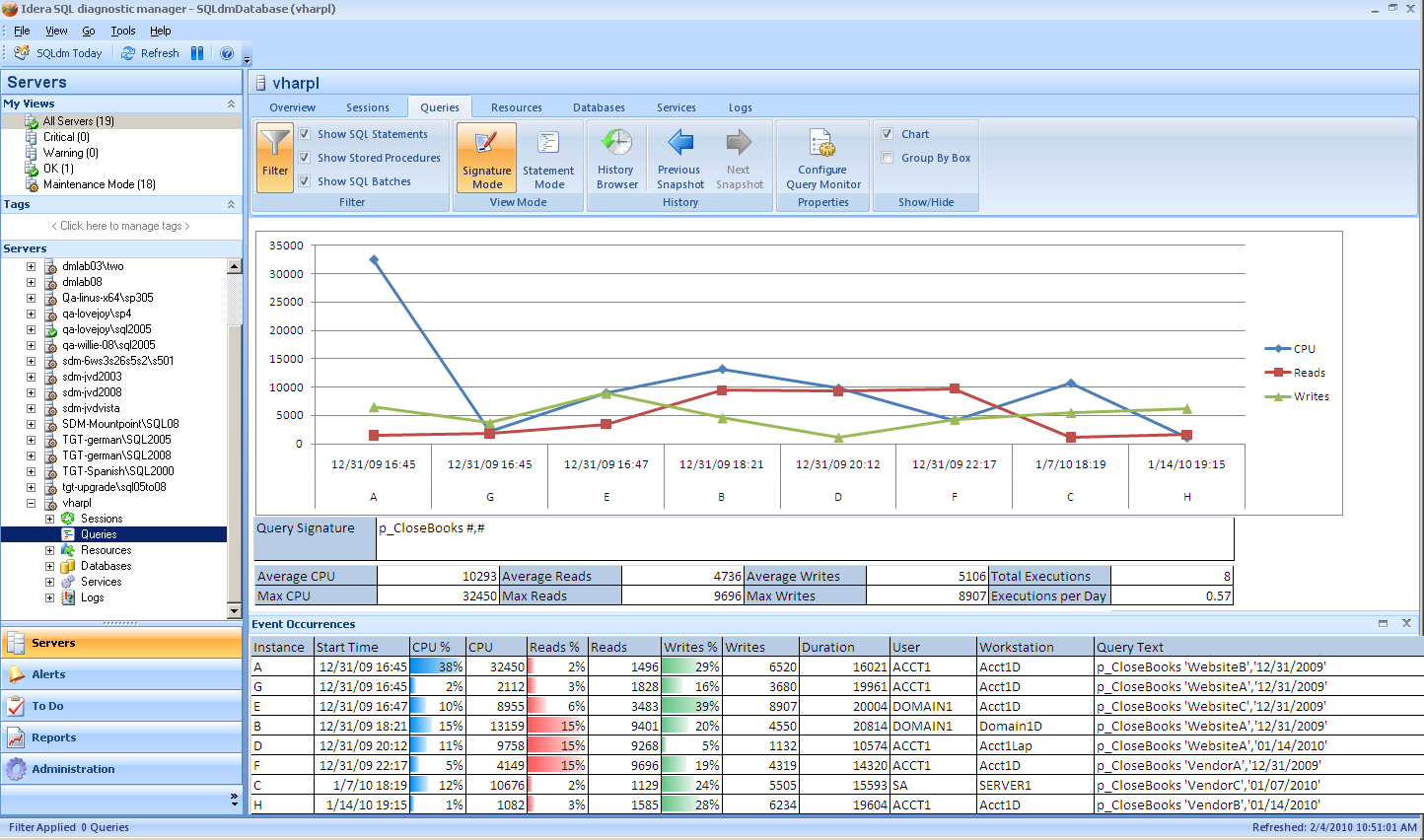
# Functional Design

## User Interfaces



The main Queries view will be altered as follows:

* The most common filters for the view will be clearly located across the top unless minimized. By clicking “Advanced Filters” the user may access additional filter criteria.
* 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:
  + Longest Running Queries (Max, Average, or Cumulative)
  + Highest CPU Consuming Queries (Max, Average, or Cumulative) \*default
  + Highest IO Consuming Queries by Reads (Max, Average, or Cumulative)
  + Highest IO Consuming Queries by Writes (Max, Average, or Cumulative)Highest CPU Consuming Applications (Max, Average, or Cumulative) \*defaultHighest IO Consuming Applications by Reads (Max, Average, or Cumulative)Highest IO Consuming Applications by Writes (Max, Average, or Cumulative)Highest CPU Per Second Queries (Max or Average)
  + Highest IO Per Second Queries (Max or Average)
  + Most Frequently Blocking Queries
  + Longest Blocking Queries
* Most Frequently Deadlocking QueriesEach 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 leftmost graph. 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.
* The user will be able to switch between Signature and Statement mode in the filters region of the screen.
* The view should have a “SQL Include” filter (shown in the prototype is a SQL Exclude filter).
* 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. 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



A secondary Query Statistics view will be provided for drill-down purposes from both the Queries view and from other views such as Sessions or Locks.

* 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
* 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.

The same timeline selector from the Queries view will be available in this view for time selection

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 queries grooming job.

## 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.

Several changes will be made to service-side trace processing:

* The signature of each query monitor event will be calculated using a regular expression, rather than the former practice of calculating the signature in the repository only when requested.
* Each query monitor statement will have an SHA-1 hash created from both the signature and raw event strings in order to facilitate easy statement aggregation in all parts of the product
* The raw and signature versions of query monitor events will be aggregated separately from run-time information within the collection service in order to reduce duplication of data
* The collection service will do minor aggregation of statements as they are read to prevent duplication of effort by the management service later in the process

### Management Service

Query Monitor Processing

* The management service will insert the collected raw and signature text from the refresh events before inserting the query execution data.
* The aggregated data from the refresh will be used to update the running aggregation in the repository.

### 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. Moving forward this area of the repository will be highly normalized in order to optimize our space usage. The proposed tables are as follows (subject to change during implementation):

* QueryStatements
  + Contains the per-execution statistics for trace data in a form close to the existing QueryMonitor table, but with all text fields extracted to foreign tables
  + This table is intended to be deeply groomed in a multi-step process:
    - Retain the single worst statement for each of duration, CPU, reads, and writes
    - Reduce the remaining queries to the single worst execution for any individual signature
    - Retain only the top 10% by duration (arguable) of the statements remaining after step 2
  + The frequency of this grooming is open to discussion.
* LeadBlockers
  + This is a new table that will be used to store the deserialized lead blockers which were detected during the refresh. This allows us to provide a “worst blockers” list
  + This table will link to the QuerySignature and QueryText tables
  + This data will be groomed out with the rest of the session data
* DeadlockStatements
  + This is a new table which will store the time of a deadlock and the hash of the signature of the statements involved with the deadlock. This allows us to provide a “most often deadlocked” list.
  + This table will link to the QuerySignature, QueryText, and Deadlocks tables
* QueryAggregation
  + Contains the aggregated statistics for a query signature. No aggregation will be provided for individual executions of a query.
  + The aggregation period (hourly, every 10 minutes, daily) is open to discussion
  + Intended to be groomed at 30 days, or the configured retention period for Query Monitor stats
* QueryStatistics – If needed
  + This contains information similar to the graph at the bottom of the old Queries view. If such a graph is not kept in the new view, this table may be superfluous
  + Contains the aggregated statistics for all queries of a given type (batch, stored procedure, or statement)
  + Intended to be groomed at 30 days, or the configured retention period for Query Monitor stats
* QuerySignature
  + Contains the first 8000 characters of a query signature, which may be linked to both QueryStatements and QueryAggregation
  + Unicode data and values beyond 8000 characters will be available in the overflow table, QuerySignatureOverflow
  + Also contains a link to the single worst execution of QueryText
  + Groom out when no longer linked to any other table
* QueryText
  + Contains the first 8000 characters of an individual execution of a query, which may be linked to both QueryStatements and QuerySignature
  + The link to QuerySignature exists in order to provide a single example of a fleshed-out query once the
  + Unicode data and values beyond 8000 characters will be available in the overflow table, QuerySignatureOverflow
  + Groom out when no longer linked to any other table
* ApplicationName, UserName, ClientComputerName
  + Lookup tables for string values which would otherwise be fully expanded in QueryStatements
  + These are mostly of value for reporting purposes and do not need to be included in the main table
  + Groom out when no longer linked to any other table

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).

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. It will also be used to upgrade the old data (see 5.2).

### Desktop Client

The user interface will be modified as detailed in 4.1 above.

No changes will be made to the trace configuration screens.

### Reports

Due to the significant change in repository structure, all Query Monitor related reports will need extensive re-working in order to continue to function. Existing PRs with regard to adding additional filters to the report may be undertaken at this time.

## 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 grooming job will incrementally upgrade data and insert it into the new tables, working on a small (TBD) number of rows per run of the job and moving back in time until all old data is either upgraded or groomed out. The management service may need to be involved in the upgrade process in order to provide proper hashes for stored queries. The details of this process will need to be solidified once the new table structure is in place.

## Schedule

### Work Breakdown and Sizings

|  |  |  |
| --- | --- | --- |
| Component | Who | Sizing |
| Desktop Client |  | 7 days |
| Services Work |  | 4 days |
| Repository Work |  | 4 days |
| Grooming Job and Upgrade Changes |  | 3 days |
| Report Upgrades |  | 2 days |
| Unit and Integration Testing |  | 4 days |
| **Total** |  | **24 days** |

### Areas of Risk

The primary area of risk for this feature is in the repository changes and the associated upgrade and grooming job alterations that are required.

# Quality Assurance Considerations

### Overview

* It would be prudent to compare the speed and repository size of 6.1 versus 6.2 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.
* 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.

### Developer-Created Unit Tests

None anticipated

# 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