



# Quick Start Guide

Ignite for SQL Server

---

Confio Software  
4772 Walnut Street, Suite 100  
Boulder, CO 80301  
866.CONFIO.1

[www.confio.com](http://www.confio.com)

## Introduction

Confio Ignite gives DBAs the ability to quickly answer critical performance questions about their database. Ignite is the fastest and most effective solution for finding the root cause of database delays that impact business applications.

Even the simplest tools require you to start somewhere. This Quick Start Guide helps DBAs immediately begin using Ignite to identify and resolve the urgent performance problems in their database.

You will learn how to:

1. Identify the top three queries with the highest impact on application response time
2. Correlate these queries with system resources to understand the complete picture
3. Identify the specific bottlenecks slowing response time
4. Diagnose current problems happening right now
5. Become proactive with by using alerts and automatic report distribution

These first steps cover the basics of Ignite, but once you have completed the Quick Start Guide you will have acquired enough knowledge and experience to begin performing more advanced analysis. The examples used herein apply to Ignite for SQL Server. There is another Quick Start Guide available Oracle, DB2 / LUW, and Sybase at [www.confio.com](http://www.confio.com).

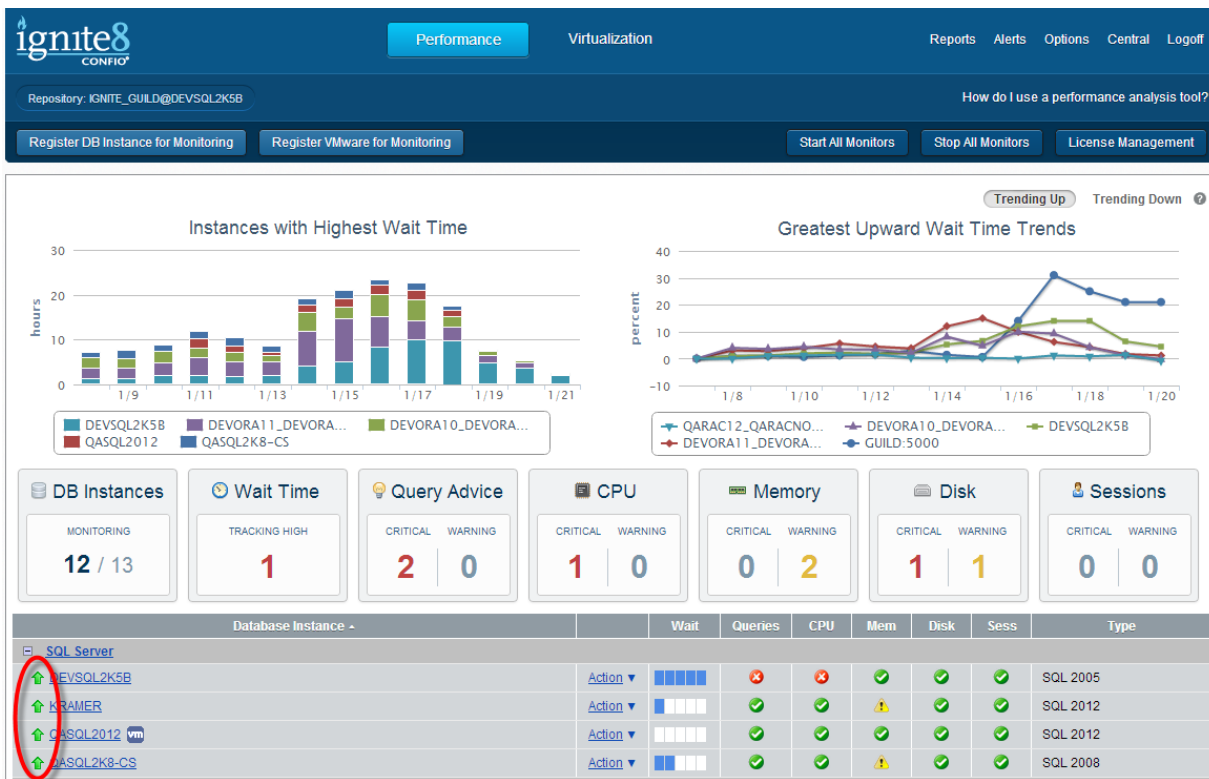
## The Quick Start Worksheet

For the best results, be sure to record your results at each step of Quick Start process. Use the [Worksheet](#) on the last page to track your progress.

Results - Record your Action Steps	
<b>Step 1</b>	<b>Find Top 3 SQL Queries Causing User Del</b>
<i>Action: Identify 3 SQLs by hash values</i>	
	SQL1
	SQL2
	SQL3
<i>Action: Assian Names to 3 SQLs Identified above</i>	

## Installation

Ignite typically takes less than 20 minutes to install and configure. Rest assured you can be comfortable setting up Ignite to monitor a busy production database. Ignite is an agentless solution that places less than 1% load on the monitored databases. You will see best results on a heavily used database where there are problems to solve. This guide assumes Ignite is currently installed and you have started monitoring a database.



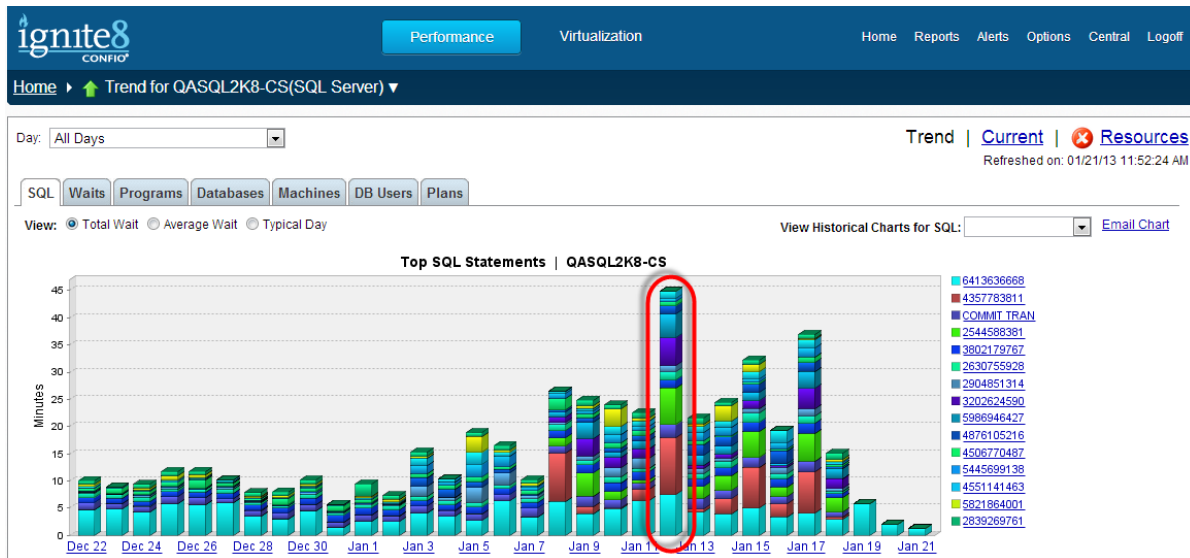
*Screen 1: Main screen with instances and monitor status*

Note the green icon left of the instance name indicating your Ignite monitor is running and capturing performance data. Before continuing with the first step in the Quick Start Guide, let Ignite capture at least two days of performance data so you have a representative sample of data to begin.

### Step 1 – Identify Top 3 Query Problems

The efficiency of processing SQL queries is one of the most important indicators of database performance, so start by determining which queries are creating the largest delays in the database. This response time measure is critical, because it reflects where end-users are waiting on the database. By identifying and improving the response time for SQL queries, you have the biggest impact on the service delivered by your database.

Click on the instance name to bring up the Trend View showing Top SQL Statements for the days you have gathered data.



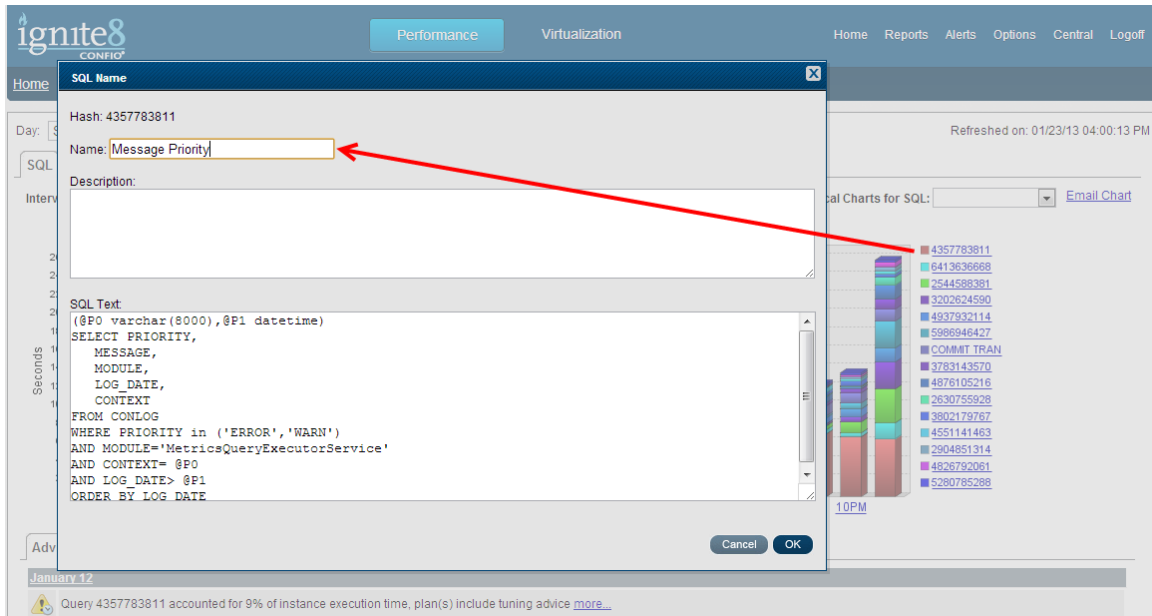
*Screen 2: Top SQL queries by daily wait*

Screen 2 displays data for each day the monitors have been running. The length of each bar represents the total wait time across all sessions. The bigger the bar, the longer the users had to wait for the database. The colors represent the individual SQL queries, identified by hash values, accumulating the response time delay. In Screen 2, it is immediately obvious that a combination of queries on January 12 is creating the most significant wait in the instance QASQL2K8-CS.

From this first screen you can identify the three SQL queries most responsible for database delays. Which SQL queries were consistently contributing to response time on the days where data was captured? If there was a day with particularly poor performance day, which SQL queries were most prevalent on that day? The legend entries are sorted by total database wait time, so the first three entries on the list should be your highest priorities.

**Action:** Note the top three SQL query hash values with highest performance priority and enter this information into the [Quick Start Worksheet](#).

Ignite lets you assign names to the SQL queries so reports and charts are more understandable. In Screen 3, the legend shows SQL names, not just numerical hash values. Before you move on to more detailed analysis, assign names to the SQL queries you have chosen.




*Screen 3: Assign a name to SQL hash value*

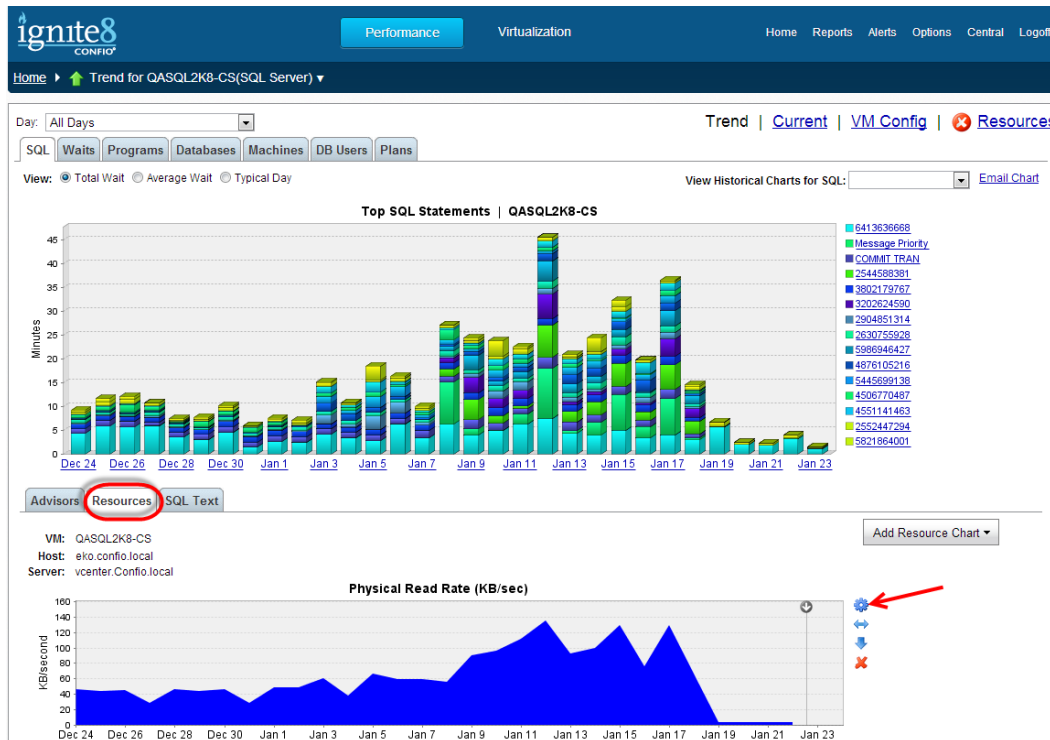
Click on the SQL hash number in the legend to get the SQL Name dialogue box. Assign an understandable name to this query as it will be used throughout all Ignite reports.

**Action:** Assign names to your top three SQL query problems.

## Step 2 Correlate Response Time with System Resources

Now it's time to compare the response time results with system resource usage such as CPU, I/O and cache. You will learn if specific response time spikes are related to server resources by comparing chart results at the same times.

Click the Resource tab on the Trend View screen. A default set of resources are shown, but you can change them by using the tool  on the right.



Screen 4: Resource tab correlates resources with response time

Choose from the resources of most interest with the Add Resource Chart button. Customize this resource page so you can quickly return and correlate with the response time chart.

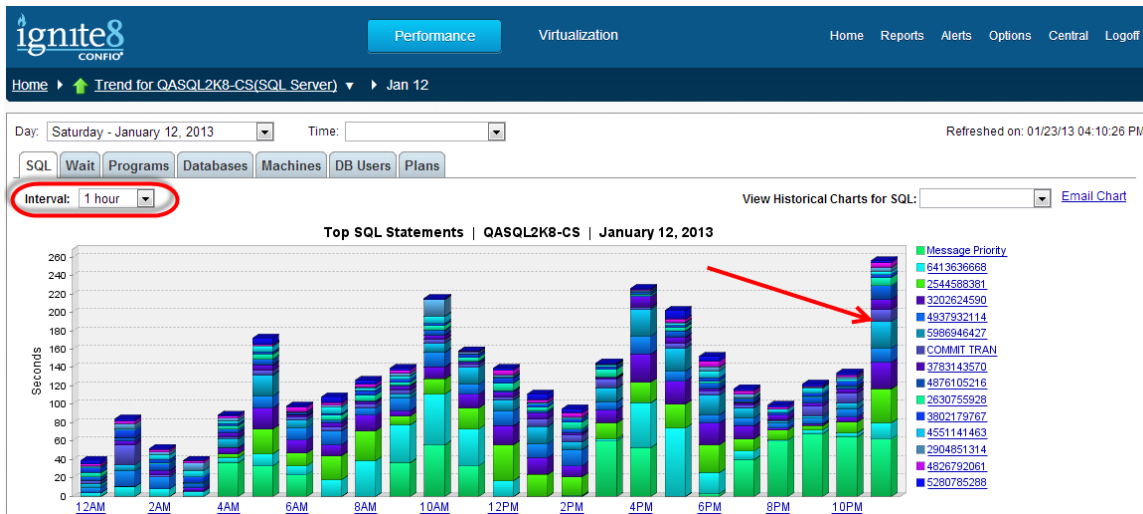
**Note:** Viewing the specific top Wait Types, as explained below, will help you determine which resources are having the biggest impact on database response time. For example if you have an SQL Server db\_file\_sequential\_read as a top Wait Type you would want to monitor storage layer resource metrics.

**Action:** Choose the two or three resources of most interest at your site and customize your tab. Note the resource names in the [Worksheet](#).

### Step 3 – Determine Specific Wait Bottlenecks Slowing Response Time

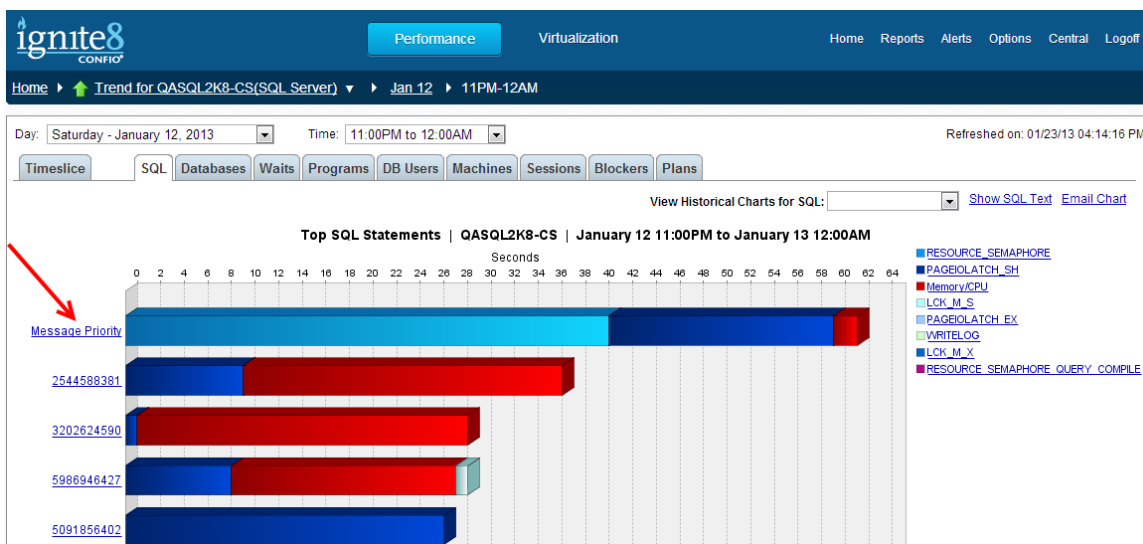
Once you have the basics of identifying problem SQL queries based on response time analysis and impact, you can drill down for more detail about the problem and identify the cause of the bottleneck.

From Screen 2, Top SQL by Daily Wait, click on your day of concern to drill in for more detail. See the daily wait time chart, Screen 5, showing the response time delay in each hour of the day and the SQL queries that created the delay. This is similar data, but is now focused on one day with each bar representing an hour. The *Change Time Interval* link up top can be used to adjust the time interval. Drill into a specific hour then use the Timeslice tab to view a range of intervals down to one second.



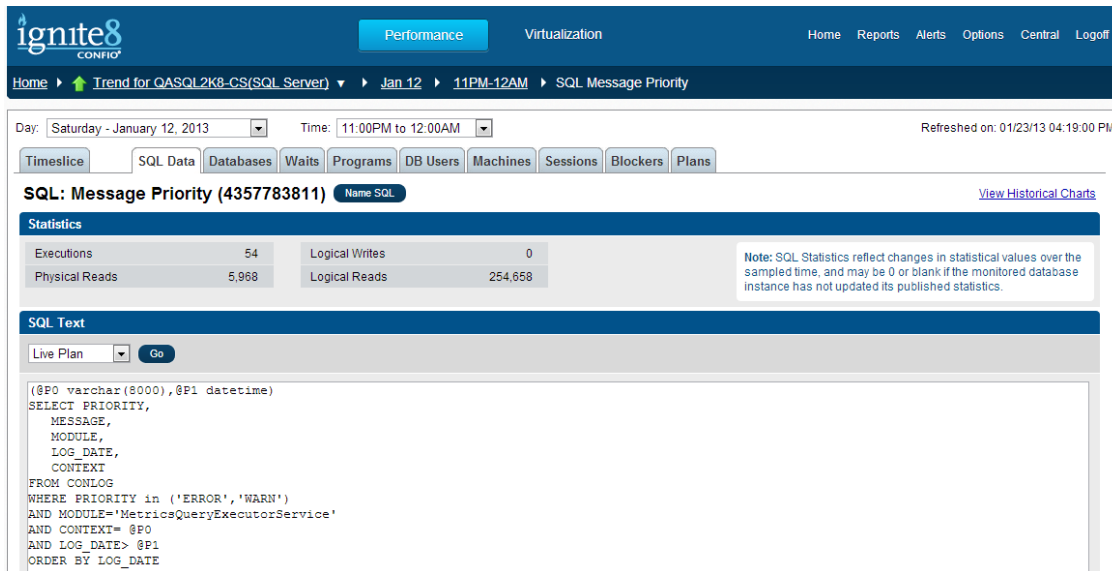
*Screen 5: SQL statements causing delays in each hourly period*

Was there a particular hour that had problems or that represents commonly found problems? From Screen 5, choose an hourly period of concern. Click on the hourly link to bring up the Top SQL chart, Screen 6.



*Screen 6: Top SQL statements causing delays in the selected hour*

Screen 6, Top SQL Statements, ranks the specific query bottlenecks in the chosen period. Here the SQL queries with most accumulated wait time across all sessions are charted to show the total delay for each. If they were running during this time period, your named SQL statements from Step 1 may appear with long bars. Click the name (or hash) to get detail on the SQL query.



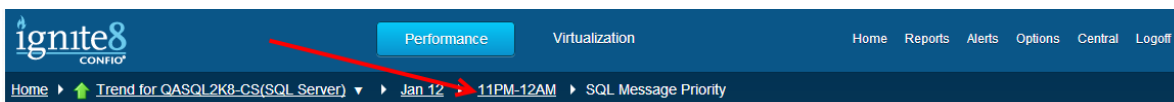
Screen 7: Detail and statistics on problem SQL statements

The SQL text and execution statistics are shown in screen 7. The tabs along the top row allow you to drill down based on Programs, Sessions, Objects and other dimensions captured in the Ignite repository.

**Action:** Click the Programs tab and see which Programs were executing the query you have chosen. Note the SQL program name in the [Worksheet](#).

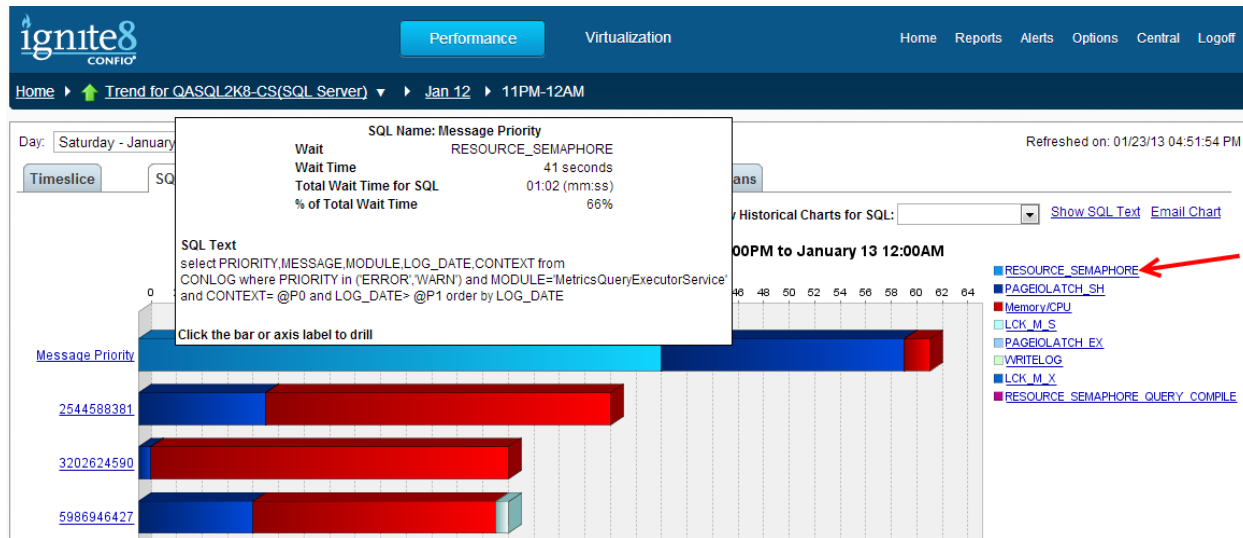
Now you have detailed visibility into the problem SQL query. Use the Ignite screens illustrated to identify the specific cause of the bottleneck.

For easy navigation, use the breadcrumb trail in the top banner to move to previous screens and keep track of where you are. Click the hourly period to go back to Top SQL Statements for that hour.



Now go back to the Top SQL Statement chart again, shown as Screen 8. For the first query, 'Message Priority', note how the majority of the bar corresponds to a colored "Wait Type" in the legend, "RESOURCE\_SEMAPHORE".

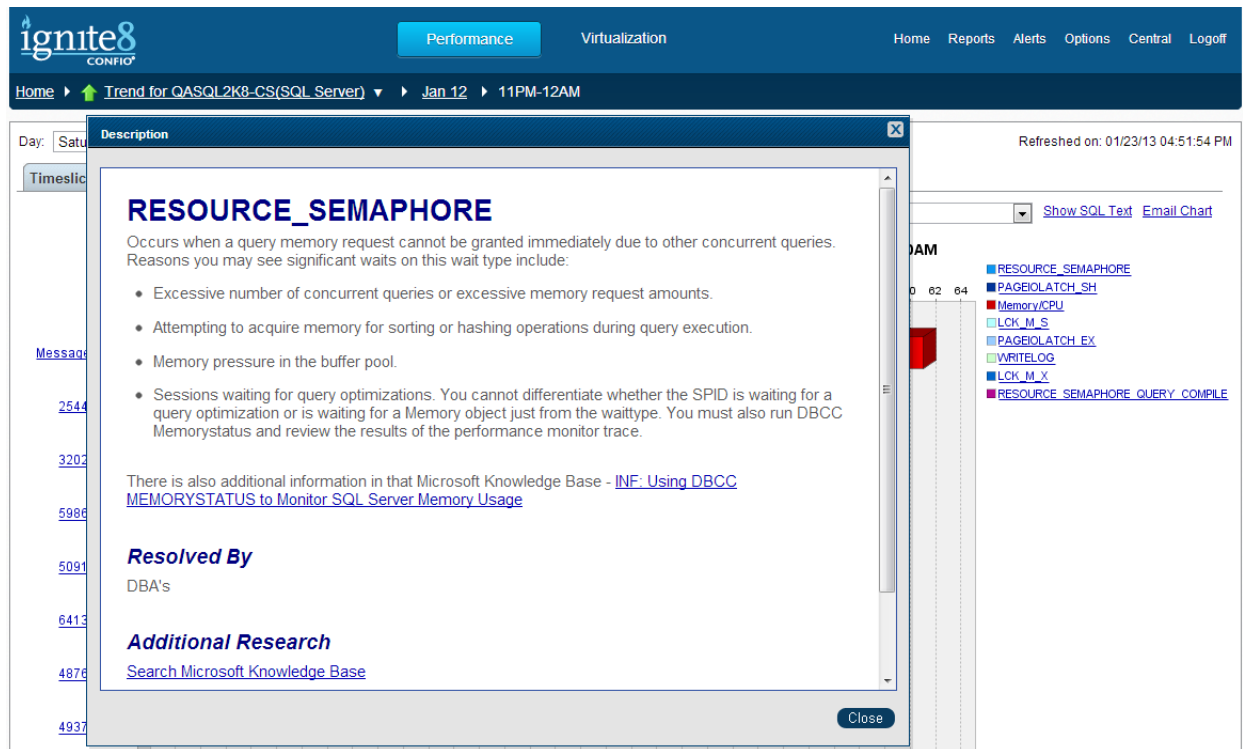




Screen 8: Top SQL statements causing delays in selected hour

This is an important piece of information. Ignite has shown the specific bottleneck causing delay in the query 'Message Priority'. The "RESOURCE\_SEMAPHORE" wait type is the one item causing delay, here shown as responsible for 66% of the response time. What is this wait type?

Click on the wait type in the legend to see an explanation of the specific bottleneck as shown in Screen 9.



Screen 9: Description of the wait type and details on how to best resolve the issue

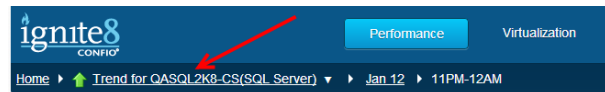
Now Ignite has isolated the top SQL queries creating increased user response time, and has shown you the wait types uniquely responsible for the response time in each one.

**Action:** For the three SQL queries of interest identify the two or three wait types that comprise more than 75% of the total response time. In many cases, one single Wait bottleneck will be responsible for nearly all of the delay.

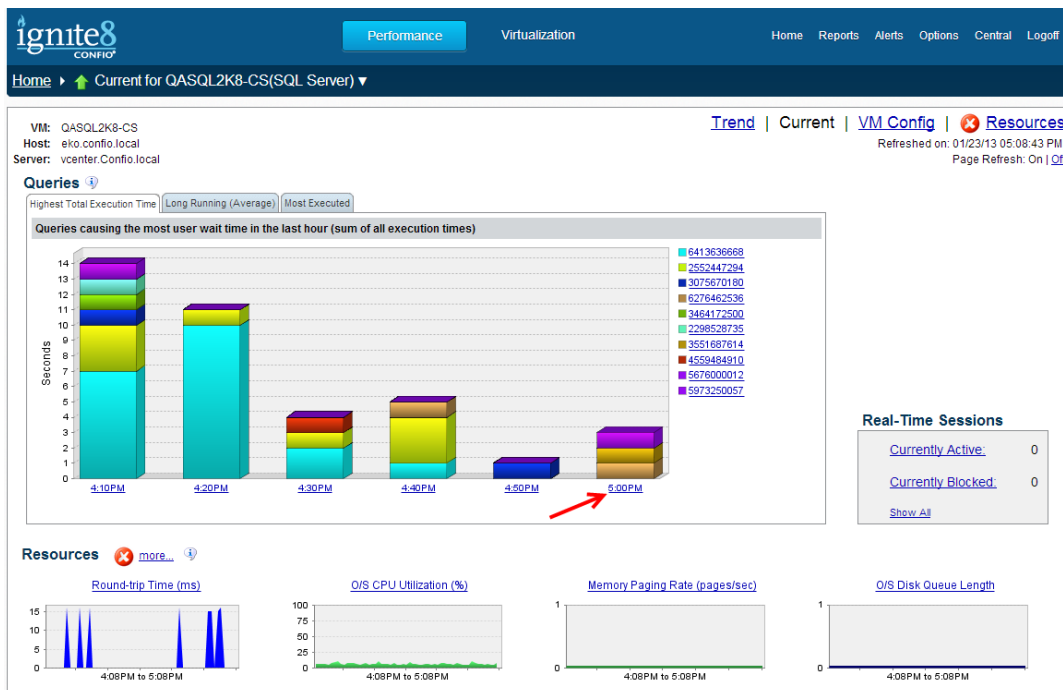
### Step 4 – Diagnose the Situation Right Now

Up to this point all the analysis has focused on historical Trend View data. But when a problem is actually occurring, you need to be able to instantly diagnose it in real-time. To see immediate performance issues simply switch to Ignite Current View and examine response time and resources within the past minutes and seconds.

Go back to Screen 2: Top SQL by Daily Wait then click the Current link from the upper right of the Trend screen to see the last hour of collected data.

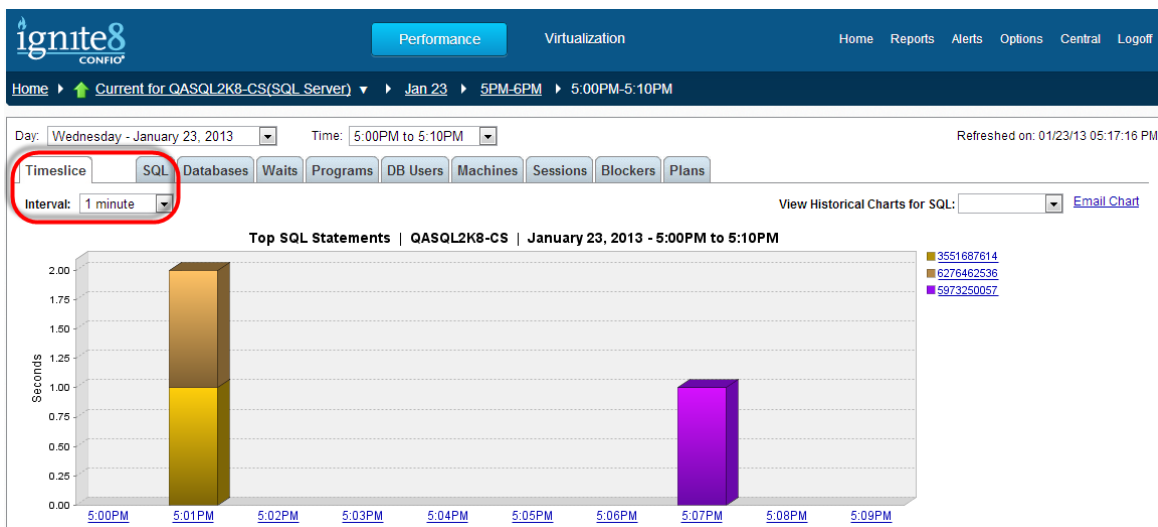
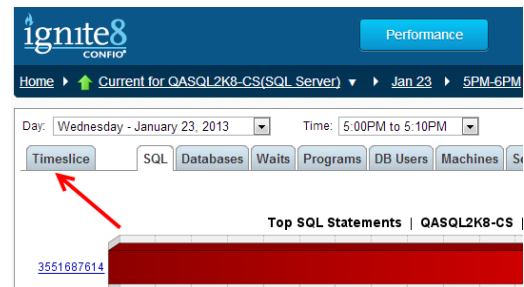


The Current View starts off by giving you an overview of the past hour broken into 10 minute intervals. Click it to drill down into the top SQL queries during a recent 10 minute interval.



Screen 10: Current View shows current conditions

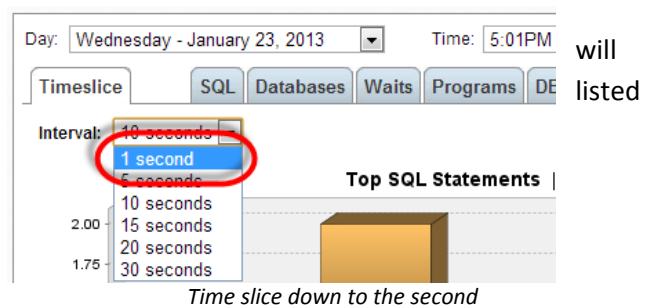
Then click the Timeslice tab, upper left, to see a minute by minute breakdown of the response time and resource data, shown in Screen 11.



Screen 11: Timeslice to see performance over small time increments

The Timeslice view breaks the current data into one minute intervals, for both response time and system resources. Examine the response time charts, then scroll down to the resources and identify any correlation.

**Note:** To get to a one second time slice you need to click on one of the one minute times below the chart in Screen 11 (5:01 PM for example) then click on the Timeslice tab.



Time slice down to the second

**Action:** Evaluate the current performance situation. If there are queries and resources that are showing performance problems right now, record them in the [Worksheet](#).

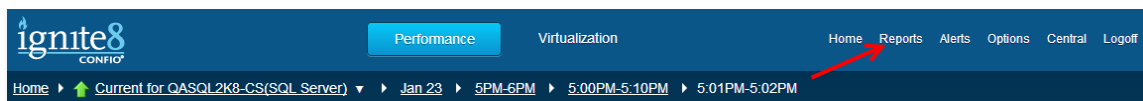
## Step 5 – Set up Proactive Management

Proactive response time monitoring is one of the best ways to maintain top database performance. Ignite’s automated reports and alerts help keep DBAs, managers and developers focused on improving database response time, while immediately warning the appropriate people when significant performance degradation occurs. Many times this results in resolving issues before they have critical impact end users.

In this section you will setup reports to automatically run daily and email detail on your top three queries.

### Reports

Select the Reports link from the top of the Ignite main screen.



Now select the database instance to use for the report and Report Type = Top SQLs. Then click Advanced Options.

**ignite8**  
CONFIO

Home Reports Alerts Options Central Logoff

**Report - Advanced Options**

**New Report**

Database Instance: QASQL2K8-CS  
Report Type: Top SQLs

Display Report

**SQL Statements to Display**

Top SQL Statements Ranked by Cumulative Wait Time  
Top 3 SQL Statements

User-Defined SQL Statements

**Dates to Display**

Date Range: Current Week  
Hour Range: 12:00am to 12:00am  
Granularity:  Auto  
 Hours  
 Days  
 Weeks  
 Months

Date Range: January 20, 2013 - January 26, 2013  
Days of Week: Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday

Available Data  
Daily: None Hourly: December 18, 2012 - January 23, 2013

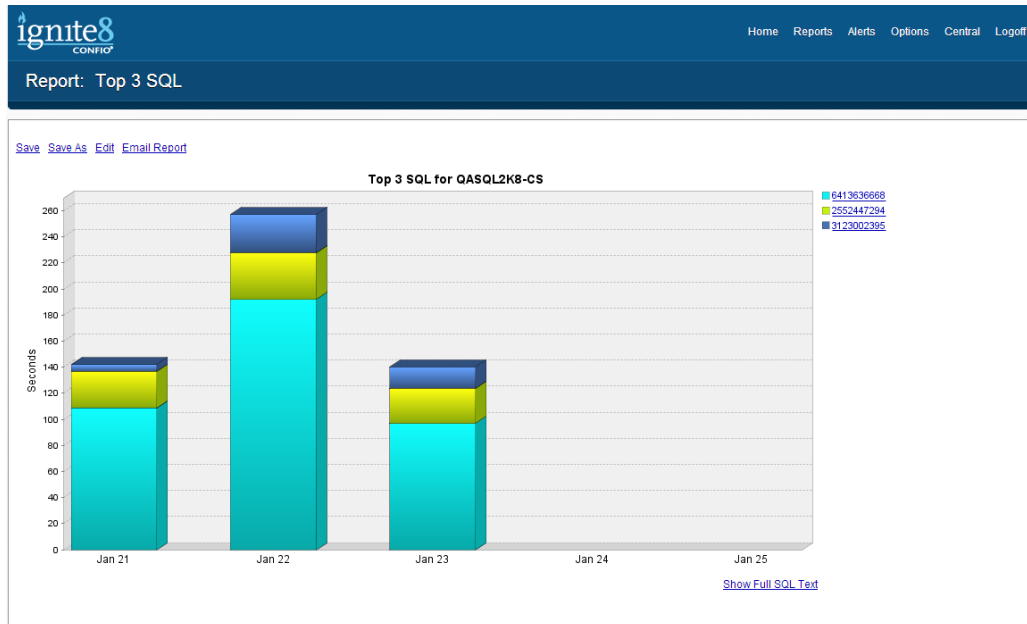
**General**

Report Name: Top 3 SQL  
Report Title: Top 3 SQL for QASQL2K8-CS  
Report Description:

*Screen 12: Report creation*

Now you can choose the Top SQL Statement Option and number of SQL statements to display. Below specific a date range and name the report. Of course you can choose any options and make changes, but this will get you started.

Then click the 'Display Report' button near the top of the Report Creation Window. Screen 13 shows the report with just the chosen queries for the most recent days.



*Screen 13: Custom report for three critical queries*

Make sure to use the Save As link with a report name to keep selections you have chosen for repeated use and scheduled distribution.

Now you want to schedule the report to be delivered automatically via email to you and your manager on a weekly basis. Click the Options link at the top right, then from the Monitor Setup tab click Report Schedules.

**Note:** Ignite has a default SMTP email configuration that works in most cases. If your network or firewall requires use of an internal SMTP server, see the Knowledge Base article at (<http://support.confio.com/kb/1524> on configuring SMTP for outgoing email).

Creating a report delivery schedule just requires a few pieces of information on this screen:

- Name the schedule and descriptive text for the email
- Choose when to deliver the report – daily, weekly or monthly
- Choose the report you want, using the name created earlier
- Choose the recipients by adding their email addresses

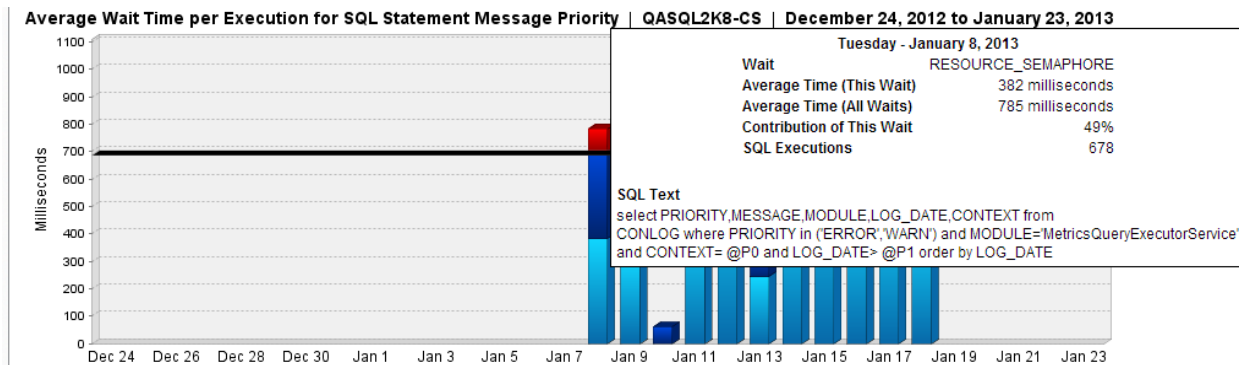
**Action:** Create the report “Total Response Time for 3 Critical SQLs” for the queries identified in Step 1. Create a report schedule to deliver it via email to you, colleagues or managers every Monday morning.

### Setting Alerts

Now that you can find the routine performance history at any time and can generate summary reports to watch the queries of interest, take the final step of establishing proactive alerts to watch for major increases in SQL response time.

You will determine the average execution response time for each of your SQL queries, and then set Alerts to warn you of significant deviations.

First, determine the normal execution time for your queries. Go back to Screen 2: Top SQL by Daily Wait. See the dropdown menu at the top, “View Historical Charts for SQL”. Choose your first query and view the historical chart, as in Screen 14. You will see this chart with a black bar through the center located near the bottom on the page, Historical Charts for SQL. This chart shows the average execution time for the SQL on each day, and the black bar denotes the average wait for this query across all the days displayed in the chart.



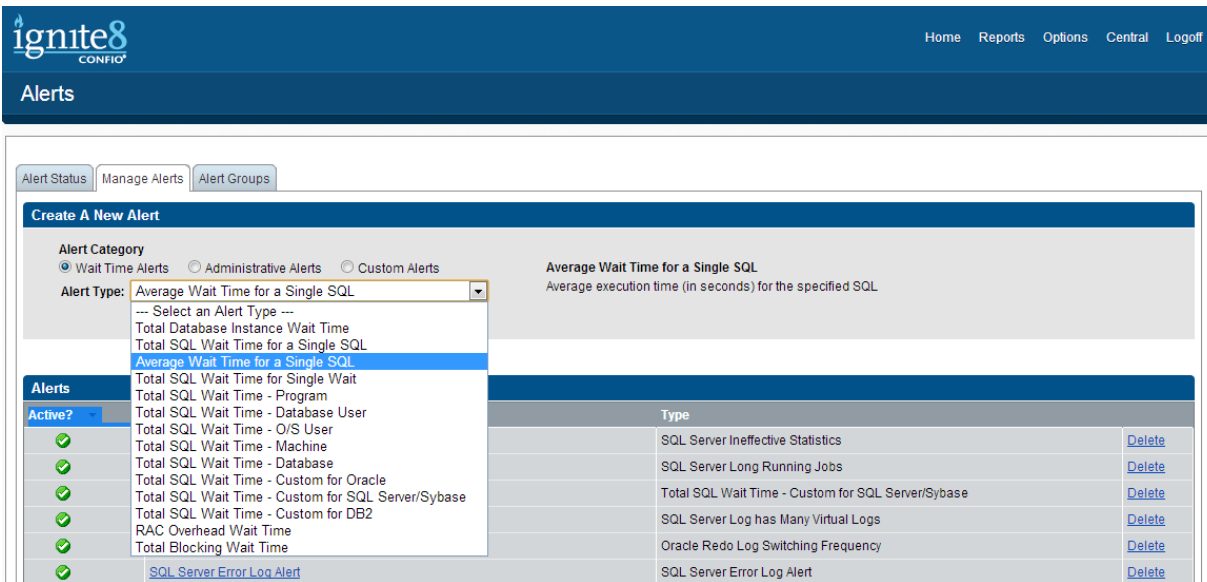
Screen 14: Daily average wait time on historical SQL page

Rolling the cursor over each bar shows the popup window with details specific to this day including the Average time for this wait type, plus the average time for all waits on this day. In this case, the average wait time was 785 milliseconds for the example in Screen 14. (Note that execution time, wait time, and response time all refer to the same concept of how long it takes for the database to return a response to the application).

The black bar shows an average time of 700 milliseconds wait for this query for all executions for the entire month. A best practice is to alert if the average response during a period is greater than 100% higher than this average during a sample period. In this case that would be roughly two seconds.

**Action:** For each of the three chosen SQL queries, estimate the typical response time. Choose a threshold, approximately twice as long, that you will use to set alerts for notification.

To create the alert, go to the top right navigation and click Alerts, then the Manage Alerts tab. Here you can choose from Under Create New Alert, select category Wait Time Alert and type Average SQL Wait Time. Ignite is unique in its capability to perform alerting on wait time/response time for SQL queries.



*Screen 15: Create a new alert*

Press Create Alert and fill in the details as in Screen 16 to match your SQL query. In this example, average response times between two and four seconds for the query named ‘Message Priority’ will generate a medium alert message to a specific DBA. If the average wait is over four seconds it will cause a high alert message to be sent to the On Call group.

Choose the database instance and one of the three important SQL queries you selected in Step 1 as the target for your Alert.

- Set low and high level alert thresholds that are approximately twice as long as the typical execution times for your query.
- Run the alert every 60 minutes or more, allowing enough time to get valid samplings and ensure you don’t get unnecessary warnings from a single slow execution.
- Make sure you test the alert (see Test button) to ensure it works properly and can send email messages from your network.
- Name the alert, give it a short description, and save it.



Create Alert

Type: Average Wait Time for a Single SQL  
 Description: Average execution time (in seconds) for the specified SQL  
 Alert Name:   
 Active:   
 Execution Interval:    
 Notification Text - Explanation or resolution steps to be sent with alert email:

**Available Database Instances**

GIBSON:5000  
 GIBSON:5000/QASAMPLE  
 GUILD817\_GUILD  
 GUILD:5000  
 KRAMER  
 MARTIN:5000/TOOLSDB  
 QARAC11\_QARACNODE1  
 QARAC12\_QARACNODE2  
 QASQL2012

**Selected Database Instances** View All

QASQL2K8-CS

SQL Hash:   For Oracle instances, use search button to find SQL Hash by SQL ID

**Configure Alert Levels and Recipients**

<div style="margin-bottom: 5px;"><span style="color: red;">✘</span> HIGH</div> <div style="margin-bottom: 5px;"><span style="color: orange;">✘</span> MEDIUM</div> <div style="margin-bottom: 5px;"><span style="color: yellow;">⚠</span> LOW</div> <div style="margin-bottom: 5px;"><span style="color: blue;">i</span> INFO</div> <div style="margin-bottom: 5px;"><span style="color: grey;">✘</span> BROKEN</div>	<table border="0"> <tr><th>Min (seconds)</th><th>Max (seconds)</th></tr> <tr><td><input type="text" value="4"/></td><td><input type="text"/></td></tr> <tr><td><input type="text" value="2"/></td><td><input type="text" value="4"/></td></tr> <tr><td><input type="text"/></td><td><input type="text"/></td></tr> <tr><td><input type="text"/></td><td><input type="text"/></td></tr> </table>	Min (seconds)	Max (seconds)	<input type="text" value="4"/>	<input type="text"/>	<input type="text" value="2"/>	<input type="text" value="4"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<p>Notification Group or Contact</p> <div style="margin-bottom: 5px;"><input type="text" value="(GROUP)On Call"/></div> <div style="margin-bottom: 5px;"><input type="text" value="Rob M"/></div> <div style="margin-bottom: 5px;"><input type="text" value="-- Select a recipient --"/></div> <div style="margin-bottom: 5px;"><input type="text" value="-- Select a recipient --"/></div> <div style="margin-bottom: 5px;"><input type="text" value="-- Select a recipient --"/></div>
Min (seconds)	Max (seconds)											
<input type="text" value="4"/>	<input type="text"/>											
<input type="text" value="2"/>	<input type="text" value="4"/>											
<input type="text"/>	<input type="text"/>											
<input type="text"/>	<input type="text"/>											

Screen 16: Alert creation details

**Notes:**

- You can search for the query to alert on by name or hash value.
- The hash value is always returned from search, even when a named query is selected.
- The average wait time is specific to the execution interval - so if the interval is 60 minutes, Ignite will look at the average wait time for the SQL statement chosen for that 60 minute period and compare these values to your thresholds.

**Action:** For the three selected SQL queries, and the criteria established already, create, name and save three alerts that send you warning messages when the SQL queries exhibit excessively long response times.

## **Congratulations!**

You have now completed the steps and have successfully identified your top problems, drilled down for specific roots causes, created reports to monitor changes, and set up alerts for proactive notification. You are well on your way to becoming an expert user of Ignite. The same principles now can be used to take a deeper look at all of the Ignite dimensions, including programs, blockers, plans, and top objects. You can create your own reports, and establish custom alerts on business processes. And you will always be able to answer the key question –“what is the top performance problem in my database and why?”

## **About Confio Software**

Confio™ Software is focused on database performance monitoring. Ignite is a comprehensive database performance solution for DBAs, IT managers, and database application developers. Ignite helps eliminate performance bottlenecks, speed problem resolution, and reduce the cost of operations for Oracle, SQL Server, DB2, and Sybase databases and VMware Servers.

Confio Software  
Boulder, Colorado, USA  
(303) 938-8282  
info@confio.com  
www.confio.com

### Ignite Quick Start Worksheet

<b>Results - Record your Action Steps</b>			
<b>Step 1</b>	<b>Find Top 3 SQL Queries Causing User Delays</b>		
<i>Action: Identify 3 SQLs by hash values</i>			
SQL1			
SQL2			
SQL3			
<i>Action: Assign Names to 3 SQLs Identified above</i>			
SQL1			
SQL2			
SQL3			
<b>Step 2</b>	<b>Identify Critical System Resources</b>		
<i>Action: Note System Resources that Correlate with Response Time Bottlenecks</i>			
Resource 1			
Resource 2			
Resource 3			
<b>Step 3</b>	<b>Determine Specific Performance Bottlenecks</b>		
<i>Action: Identify the Program generating most Wait Time</i>			
Program			
<i>Action: Identify Top Waits for each SQL Query</i>			
SQL1 Name		% of Time	
Wait1			
Wait2			
Wait3			
SQL2 Name		% of Time	
Wait1			
Wait2			
Wait3			
SQL3 Name		% of Time	
Wait1			
Wait2			
Wait3			
<b>Step 4</b>	<b>Current Bottlenecks</b>		
<i>Action: Identify Current Queries and Wait Events</i>			
Current Queries			
Current Wait Events			

