

**Foglight Skills 101**

**Episode 4**

# Mastering SQL Server Performance with Quest Foglight

**Janis Griffin**  
Senior Systems Consultant



**Quest**  
Where Next Meets Now.



# Who Am I?

**Current – 30+ Years in Oracle®, DB2®, ASE,  
SQL Server®, MySQL®, PostgreSQL**

**DBA and Developer**

- Specialize in Performance Tuning
- Customers Common Question: How do I tune it?



**Janis.Griffin@quest.com**

Twitter® - @DoBoutAnything

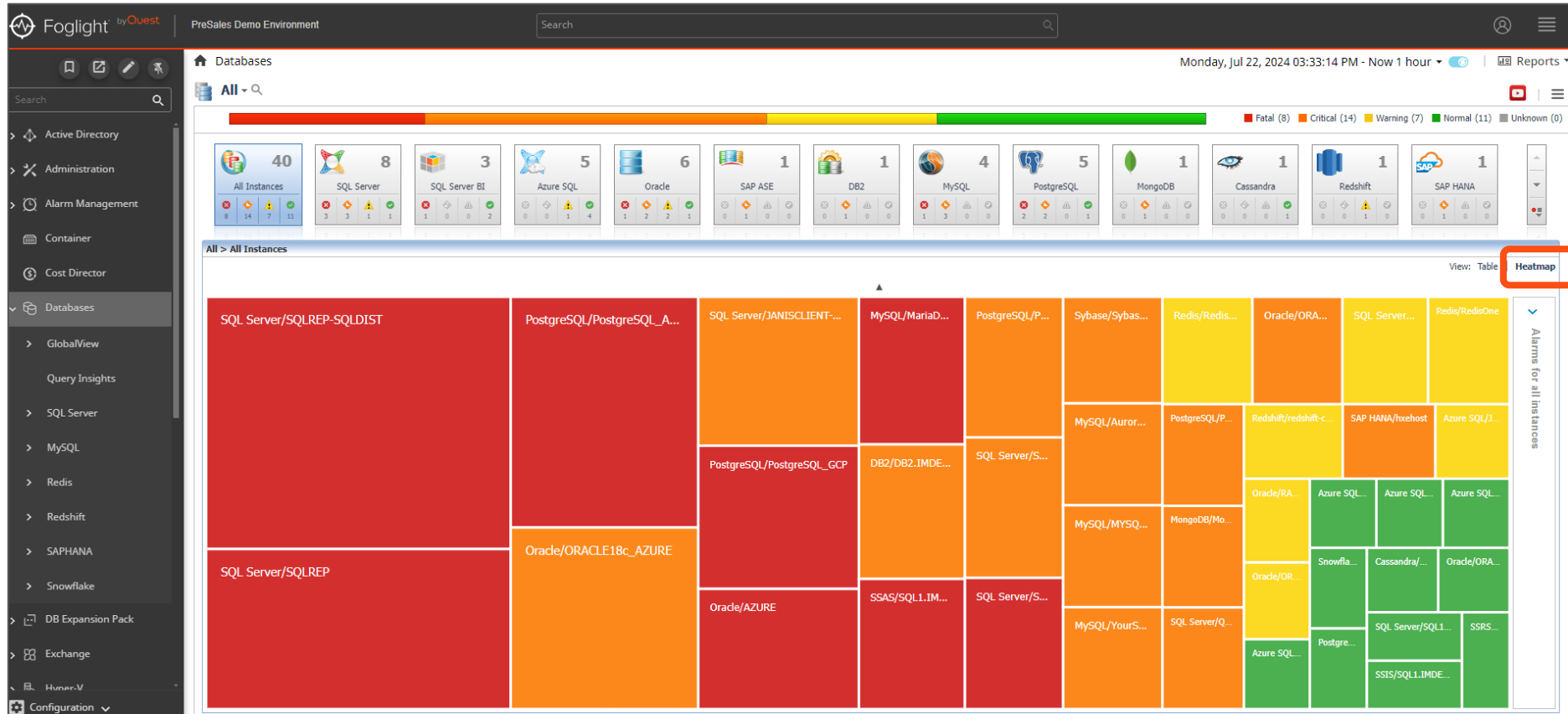


# Today's Focus

---

- **Utilize Query Insights**
- **Monitor & evaluate adaptive baselines & change tracking**
  - Including historical examination & decision making
- **Review key metrics & query execution plans**
- **Use wait types to quickly identify bottlenecks**
  - To gain insights on the best tuning approach
- **Consider tuning with SQL Optimizer**

# Foglight Shows Company Level Database Pain



# Global Company or Database Specific Views

The screenshot displays the Foglight by Quest interface for a 'PreSales Demo Environment'. The top navigation bar includes the Foglight logo, environment name, a search bar, and user information. The left sidebar shows navigation options like Active Directory, Administration, Alarm Management, Container, Cost Director, and Databases. The main content area is titled 'Databases' and features a horizontal bar with status counts: Fatal (10), Critical (12), Warning (7), Normal (11), and Unknown (0). Below this is a grid of database icons with instance counts: All Instances (40), SQL Server (8), SQL Server BI (3), Azure SQL (5), Oracle (6), SAP ASE (1), DB2 (1), MySQL (4), PostgreSQL (5), MongoDB (1), Cassandra (1), Redshift (1), and SAP HANA (1). A red arrow points to the 'SQL Server' icon. Below the grid, the 'All > SQL Server' view is shown in 'Table' mode, with a red box around the 'View: Table' button. The table lists various SQL Server instances with columns for Severity, Name, Database, Version, Up Since, Workload, DB Alarms, Host, System Utilization (CPU Load), and Monitoring Agent. Red arrows point to the 'Workload' and 'DB Alarms' columns in the table.

Sev	Name	Database	Version	Up Since	Workload	DB Alarms	Host	System Utilization	Monitoring Agent
Warning	SQL_GCP_DBaaS_2019	Microsoft SQL Server	15.0.4365.2	05/19/24 03:39	5.31	3 Critical, 9 Warning	1189db3cb1a3e48	---	SQL PI
Warning	SQLAG1.IMDEMO.LOCAL	Microsoft SQL Server	15.0.4382.1	07/14/24 02:40	0.21	10 Warning	sqlag1.imdemo.local	2%	SQL PI
Normal	SQL1.IMDEMO.LOCAL	Microsoft SQL Server	15.0.4382.1	07/13/24 23:42	0.08	---	sql1.imdemo.local	14%	SQL PI
Warning	QUESTSTC.CGKVB6DJBW1.US-EAST-2.RDS.AMAZONAWS.COM	Microsoft SQL Server	14.0.3381.3	06/23/24 00:09	0.01	4 Warning	ec2amaz-mp30ifd	---	SQL PI
Critical	SQLREP-SQLDIST	Microsoft SQL Server	16.0.4105.2	07/10/24 03:47	0.01	6 Critical, 13 Warning, 9 Normal	sqlrep	3%	Enable PI
Critical	SQLREP	Microsoft SQL Server	16.0.4105.2	07/10/24 03:45	0.01	4 Critical, 6 Warning, 8 Normal	sqlrep	3%	Enable PI
Critical	SQLAG2.IMDEMO.LOCAL	Microsoft SQL Server	15.0.4382.1	07/14/24 02:37	0.00	2 Critical, 2 Warning	sqlag2.imdemo.local	2%	SQL PI
Critical	JANISCLIENT-MSSQLSERVER01	Microsoft SQL Server	16.0.1000.6	-	-	4 Critical, 10 Warning, 2 Normal	janisclient.imdemo.local	---	SQL PI

# Quick View of Workloads / Baselines

The dashboard displays a top navigation bar with the date and time: Monday, Jul 22, 2024 03:30:14 PM - Now 1 hour. Below this is a search bar and a status bar showing 6 Fatal, 14 Critical, 6 Warning, 11 Normal, and 0 Unknown alerts. A row of database icons follows, including All Instances (40), SQL Server (8), SQL Server BI (3), Azure SQL (5), Oracle (6), SAP ASE (1), DB2 (1), MySQL (4), PostgreSQL (5), MongoDB (1), Cassandra (1), Redshift (1), and SAP HANA (1).

The main section is titled "All > SQL Server" and contains a table of database instances. The table has columns for Severity, Name, Database, Version, Up Since, Workload, DB Alarms, System Utilization, Host, and Monitoring Agent. The instance "SQL\_GCP\_DBaaS\_2019" is highlighted with a red border. Below the table, a detailed view for "SQL\_GCP\_DBaaS\_2019" is shown, including General information, HADR settings, System Utilization, Storage usage, and a Workload graph.

Sev	Name	Database	Version	Up Since	Workload	DB Alarms	System Utilization	Host	Monitoring Agent
Warning	SQLREP-SQLDIST	SQL Server	16.0.4105.2	07/10/24 03:47	0.01	4 Critical, 13 Warning	sqlrep	sqlrep	Enable PI
Warning	SQLREP	SQL Server	16.0.4105.2	07/10/24 03:45	0.01	4 Critical, 6 Warning	sqlrep	sqlrep	Enable PI
Warning	SQLAG2.IMDEMO.LOCAL	SQL Server	15.0.4382.1	07/14/24 02:37	0.00	1 Critical, 2 Warning	sqlag2.imdemo.local	sqlag2.imdemo.local	SQL PI
Warning	JANISCLIENT-MSSQLSERVER01	SQL Server	16.0.1000.6	07/22/24 11:24	0.00	1 Critical, 10 Warning	janisclient.imdemo.local	janisclient.imdemo.local	SQL PI
Warning	<b>SQL_GCP_DBaaS_2019</b>	SQL Server	<b>15.0.4365.2</b>	<b>05/19/24 03:39</b>	<b>5.68</b>	<b>3 Critical, 8 Warning</b>	<b>1189db3cb1a3e48</b>	<b>ec2amaz-mp30fd</b>	SQL PI
Warning	QUESTSTC.CGKV/B6DJB.JWI.US-EAST-2.RDS.AMAZONAWS.COM	SQL Server	14.0.3381.3	06/23/24 00:09	0.00	1 Critical, 4 Warning	ec2amaz-mp30fd	ec2amaz-mp30fd	SQL PI
Warning	SQLAG1.IMDEMO.LOCAL	SQL Server	15.0.4382.1	07/14/24 02:40	0.00	1 Critical, 10 Warning	sqlag1.imdemo.local	sqlag1.imdemo.local	SQL PI
Normal	SQL1.IMDEMO.LOCAL	SQL Server	15.0.4382.1	07/13/24 23:42	0.03	0 Critical, 0 Warning	sql1.imdemo.local	sql1.imdemo.local	SQL PI

**SQL\_GCP\_DBaaS\_2019** Details:

- General:** DB type: SQL Server, Version: 15.0.4365.2, Up since: 05/19/24 03:39, Host: 1189db3cb1a3e48, OS Cluster: Disabled.
- HADR:** Log shipping: Disabled, Mirroring: Disabled, Replication: Disabled, Always On: Enabled, Cluster Aware: Disabled.
- System Utilization:** CPU Load (%): n/a, Memory (%): n/a, Disk (% Busy): n/a.
- Storage:** Total Used Space Percent: 24%, File Groups: 7, Files: 17, 15 GB.
- Alarms:** 0 Critical, 3 Warning, 8 Normal.
- Performance:** Active sessions: 1, Total sessions: 85.
- Workload:** Graph showing workload over time from 15:30 to 16:25.

# Get Query Insights over All or Selected Domains

**Foglight** by Quest | PreSales Demo Environment | Search | Monday, Jul 22, 2024 03:44:15 PM - Now 1 hour

### Query Insights

Display: Top 50 Queries ▾ by Impact

SQL Server ▾ Impact ▾ Response Time ▾ Elapsed Time ▾ Executions ▾ Search text [ ] Apply [ ] Reset [ ]

Query	Impact on target	Impact	Elapsed Time	Executions	Response Time	Domain	Target
UPDATE [dbo].[BANK] SET [BANK_NAME] = LTRIM(RTRIM(BAN	46.53 %	1.91 hr	40	2.87 min	SQL Server	SQL_GCP_D	
SELECT LAST_NAME, STREET INTO #tmp_sales FROM EMPLOYE	18.48 %	4.38 min			SQL Server	SQLAG1.IM	
OPEN getsumamount;	14.39 %	3.39 min			SQL Server	SQLAG1.IM	
ROLLBACK TRAN;	9.01 %	5.7 min			SQL Server	SQLAG1.IM	
INSERT INTO #GETSUMAMOUNT( CUSTOMER_NAME, ORDER_	8.76 %	4.38 min			SQL Server	SQLAG1.IM	
UPDATE [BB] set [B] = @1	8.43 %	20.85 min	242	5.1 sec	SQL Server	SQL_GCP_D	
UPDATE [AA] set [A] = @1	7.19 %	17.76 min	176	6 sec	SQL Server	SQL_GCP_D	
INSERT INTO [dbo].[ORDL1] SELECT * FROM [dbo].[ORDER_LI	6.23 %	18.05 min	71	15.9 sec	SQL Server	SQL_GCP_D	
INSERT INTO [dbo].[ORDL2] SELECT * FROM [dbo].[ORDER_LI	5.7 %	19.69 min	57	20.9 sec	SQL Server	SQL_GCP_D	
INSERT INTO [dbo].[ORDL3] SELECT * FROM [dbo].[ORDER_LI	5.66 %	19.32 min	57	20.5 sec	SQL Server	SQL_GCP_D	
DELETE FROM [dbo].[ORDL1]	5.4 %	12.64 min	70	11.1 sec	SQL Server	SQL_GCP_D	
FETCH getsumamount INTO @CUSTNME, @ORDERNBR, @PRC	5.35 %	2.56 min	98	1.5 sec	SQL Server	SQLAG1.IM	
DELETE FROM [dbo].[ORDL3]	5.09 %	11.85 min	56	12.6 sec	SQL Server	SQL_GCP_D	

**Summary:**  
Min: 28.54 min  
Avg: 28.75 min  
Max: 28.96 min

### Statement Details

Name: UPDATE [dbo].[BAN] SET [BANK\_NAME] = LTRIM(RTRIM(BANK\_NAME)) FROM BANK WHERE [BANK\_CODE] = '\*\* Removed by Foglight \*\*  
Query ID: 0x0bd0a4d0d5e94049

SQL Text: [ ] Copy [ ]

**Investigate**

#### Elapsed Time

Time	Elapsed Time (ms)
July 22, 2024 3:45 PM	28.55 min

#### Executions

Time	Executions
July 22, 2024 3:45 PM	10

# Query Insights – Investigate SQL PI

**Query Insights > SQL PI**

Monday, Jul 22, 2024 03:59:08 PM - Now 1 hour

SQL\_GCP\_DBaaS\_2019 | Overview | SQL PI Memory | Activity | Databases | Services | HADR | Logs | Configuration | User-defined

Powered by SQL PI

View as PDF

Dimension Filter: Instance View > SQL Statements > UPDATE [dbo].[BANK] ...

**Performance Tree** | Tops: 25 | History | Advanced Analytics

UPDATE [dbo].[BANK] SET [BANK\_NAME] = LTRIM(RTRIM([BANK\_NAME])) FROM BANK WHERE [BANK\_CODE] = "" Removed Foglight \*\*

**Top Wait Events**

Category	Event Name	% of Total Wait Time	Wait Time
Lock Wait	LCK_M_U	63.14	9,443.89
CPU Wait	SOS_SCHEDULER_YIELD	3.02	452.44
Memory Wait	MEMORY_ALLOCATION_EXT	1.52	227.84
Log Wait	LOGMGR_FLUSH	0.61	90.55
Memory Wait	RESERVED_MEMORY_ALLOCATION_EXT	0.39	58.87
Other Wait	CXPACKET	0.10	14.79
Network Wait	ASYNC_NETWORK_IO	0.04	6.43

**Workload related Metrics**

Metric	Resource	Total
Active Time	Workload	14,958.01
Average SQL Response Time	Workload	14.58
Batches Rate	Workload	11.66
CPU Usage	CPU	4,936.06
Deadlocks	Lock	19.33
Executions	Workload	1,564.00
Logical Reads	I/O	1,387,327,112.00
Wait Time Percent	Workload	67.00



# SQL PI – Analyze Plan

The screenshot displays the Foglight SQL PI interface. At the top, the navigation bar shows 'Query Insights > SQL PI' highlighted with a red box. The main content area is divided into two sections: 'Execution Plan' and 'SQL Text'.

**Execution Plan Section:**

- Statement:** UPDATE [dbo].[BANK] SET [BANK\_NAME] = LTRIM(RTRIM([BANK\_NAME])) FROM BANK WHERE [BANK\_CODE] = '\*\* Removed by Foglight \*\*'
- Plan Analysis:** Total cost: 0.0165715 | Total I/O cost: 0.0162500 | Total CPU cost: 0.0003215
- Object Analysis Table:**

Name	Database	Type	Associated Operators
dbo.BANK	sales	Table	Table Update, RID Lookup
dbo.BANK.PK_BANK	sales	Index (NonClustered)	Index Seek

**SQL Text Section:**

```
UPDATE [dbo].[BANK]
SET [BANK_NAME] = LTRIM (RTRIM (BANK_NAME))
FROM BANK
WHERE [BANK_CODE] = '** Removed by Foglight **'
```

# SQL PI - Compare

Query Insights > SQL PI Monday, Jul 22, 2024 04:54:35 PM - Now 1 hour

SQL\_GCP\_DBaaS\_2019 | Overview SQL PI Memory Activity Databases Services HADR Logs Configuration User-defined

Workload CPU I/O Memory Network Lock Latch Log CLR Remote Provider XTP Other

### Performance Tree

Tops: 25 | History | Advanced Analytics

Instance View

- SQL Statements
  - UPDATE [dbo].[BANK] SET [BANK\_...]
  - UPDATE [BB] set [B] = @1
  - UPDATE [AA] set [A] = @1
  - INSERT INTO [dbo].[ORDL1] SELEC...
  - INSERT INTO [dbo].[ORDL3] SELEC...
  - INSERT INTO [dbo].[ORDL2] SELEC...
  - DELETE FROM [dbo].[ORDL3]
  - DELETE FROM [dbo].[ORDL1]
  - DELETE FROM [dbo].[ORDL2]
  - SELECT [customer\_id], Sum(amount...
  - Insert Into #ResourceDB Select Dist...
  - SELECT TOP (@RowLimit) CONVERT...
  - Select Distinct A2.request\_session\_i...
  - insert into #temp\_trace select top 1...
  - Select Top 1 @DBID = B,[dbid] , @C...
  - insert into #temp\_trace select top 1...
  - select top 100 DatabaseName, File...
  - (@PO bigint,@P1 int,@P2 int,@P3 m...
  - xp\_readererrorlog
  - xp\_instance\_regread
  - SELECT top (@PO) t2.spid, t2.log\_in...
  - Select @DBName As DBName , DB\_...
  - DROP TABLE #QS\_sysfiles
  - select create\_time from sys.dm\_xe\_...
- TSQL Batches
- Databases
- Programs
- Users
- Client Machines
- Context Infos

### Resource Consumption

Dimension Filter: Instance View | SQL Statements | UPDATE [dbo].[BANK] ...

UPDATE [dbo].[BANK] SET [BANK\_NAME] = LTRIM(RTRIM(BANK\_NAME)) FROM BANK WHERE [BANK\_CODE] = '\*\*\* Removed by Foglight \*\*\*'

### Resource Breakdown

Lock Wait 100.00%

Sum of all the active waits and cpu usage, equal to the session total activity within the current interval.

### Active Time

### Workload related Metrics

Select Metric View SQL Text Analyze Plan Tune SQL Compare

Metric	Resource	Total
Average SQL Response Time	Workload	175.22
CPU Usage	CPU	< 0.01
Executions	Workload	39.00
Lock Update	Lock	6,834.83
Lock Wait	Lock	6,834.83
Logical Reads	I/O	135.00
Row count	Workload	13.00
Wait Time Percent	Workload	100.00

# SQL PI – Advanced Analytics

Query Insights > SQL PI > Compare Monday, Jul 22, 2024 05:03:56 PM - Now 1 hour

SQL\_GCP\_DBaaS\_2019 | Overview | SQL PI | Memory | Activity | Databases | Services | HADR | Logs | Configuration | User-defined

Comparison Parameters | Workload

---

Query Insights > SQL PI Time Period: Monday, Jul 22, 2024 05:03 PM - 06:03 PM 1 hour

SQL\_GCP\_DBaaS\_2019 | Overview | SQL PI | Memory | Activity | Databases | Services | HADR | Logs | Configuration | User-defined

Workload | CPU | I/O | Memory | Network | Lock | Latch | Log | CLR | Remote Provider | XTP | Other

Performance Tree | Tops: 25 | History | **Advanced Analytics**

Instance View | SQL Statements | TSQL Batches | Databases

Resource Consumption

Categories

---

Query Insights > SQL PI Time Period: Monday, Jul 22, 2024 05:03 PM - 06:03 PM 1 hour

SQL\_GCP\_DBaaS\_2019 | Overview | SQL PI | Memory | Activity | Databases | Services | HADR | Logs | Configuration | User-defined

Workload | CPU | I/O | Memory | Network | Lock | Latch | Log | CLR | Remote Provider | XTP | Other

Performance Tree | Tops: 25 | History | **Advanced Analytics**

Instance View | SQL Statements | TSQL Batches | Databases | Programs | Users | Client Machines | Context Infos | Command Types | Sessions | Locked Objects | Objects I/O | Files | Disks

Resource Consumption

Baseline | Breakdown

Change Tracking | **Advisories**

Priority	Advisory	Name
High	Objects Experiencing Lock Waits (1)	
High	Deadlock Observed (1)	

# Advisories

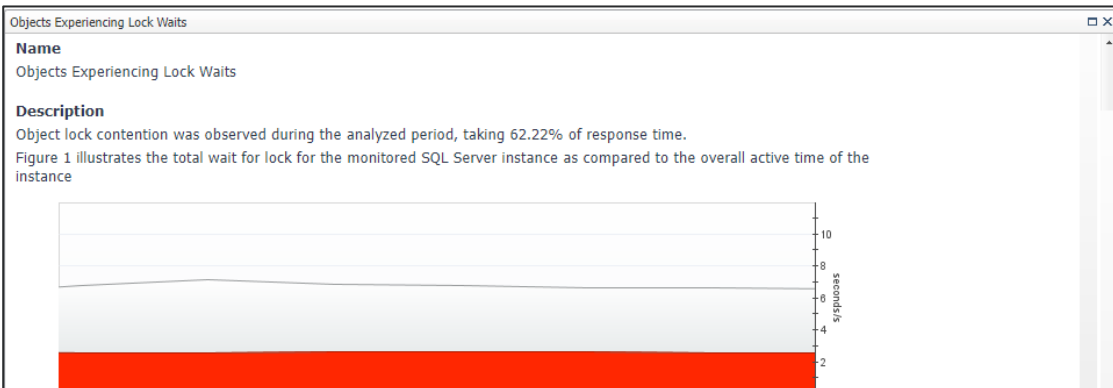


Table 1 shows the list of top objects that experienced Lock Contention.

Locked Objects	Lock Wait Time (seconds)	% of Overall Response Time
sales.dbo.BANK.PK_BANK [INDEX]	7,026.65	75.43
sales.dbo.BB [TABLE]	1,251.99	13.44
sales.dbo.AA [TABLE]	1,104.32	11.85

Click on the Locked Object name to find out more about the identity of the blockers of that object, the amount of sessions involved, and the identity of the locked sessions.

## Key Metrics

Table 2 displays the trend of related key metrics, recorded during the analyzed period.

Metric	Average	Min	Max
Blocked Lock Requests	16.91	6.00	33.00
Table Lock Escalations	10.70	0.00	40.00
Lock Timeouts	23.78	6.00	56.00
Average Lock Wait Time (ms)	12,248,568,325.27	12,244,140,552.00	12,253,095,061.00

## Recommendation

Review the application logic and consider techniques to reduce contention caused by concurrent updates to data.

Some of the main causes of table locking include the following:

- Contention for a specific row in the database. The application design may require that many processes update or lock the same row in the database. For example, when primary keys are generated using a sequence table.
- Table locks caused by non-indexed foreign keys, or ineffective / missing indexes, or out-of-date statistics. When a non-indexed foreign key is updated, the parent table could be subjected to a table lock until the transaction is complete. If the indexes defined are ineffective or missing, the SQL optimizer chooses non-optimal query plans. Non-optimal plans can also be chosen when a qualifying index exists, if statistics are out of date.
- Long-running transactions. Individual transact-SQL statements may perform well, but maintaining transactions for extended operations can dramatically increase the amount of contention within a database. Investigate transaction control logic to determine where operations can be broken up into smaller, component transactions. In addition, review the use of transaction isolation levels in application code to determine whether a less restrictive isolation level can be used.

## Next steps

- Review [Blocking History](#) for a detailed report of all sessions involved in the blocking scenario during the period described.
- Review [Activity Highlights](#) to find out top contributors to the lock wait.

# Blocking History

Home Databases > Overview > SQL PI

Time Period: Wednesday, Jul 24, 2024 06:08 PM - 07:08 PM 1 hour | Reports

SQL\_GCP\_DBaaS\_2019 | Overview SQL PI Memory Activity Databases Services HADR Logs Configuration User-defined

Worldload CPU I/O Memory Network Lock Latch Log CLR Remote Provider XTP Other

Powered by SQL PI | View as PDF

Performance Tree Instance View | Dimension Filter: Instance View

Resource Consumption | Top Wait Events

There Is No Data To Display

Baseline | Breakdown | Resource Breakdown

There Is No Data To Display

Overview **Blocking History** Activity Highlights

View Full Text

	Event Start	SPID	Blocked By	Resource	Lock Resource Type	Status	Duration	Program	User	
🔍	7/24/24, 6:39 PM	84 [2024-07-24 18:36:02.61]				Blocking	173.22	SQLAgent - TSQL JobStep (Job 0xC31233C4311984AA502BF9D0924748 : Step 1)	sqlserver	BEGIN TRAN
○	7/24/24, 6:39 PM	67 [2024-07-24 18:39:01.593]	84 [2024-07-24 18:36:02.61]	sales.dbo.BANK_PK_BANK [INDEX]	KEY	Blocked	173.22	SQLAgent - TSQL JobStep (Job 0x2BE2C65EE58F9A4FB5D10B3B9C206BEF : Step 1)	sqlserver	UPDATE [dbo]
🔍	7/24/24, 6:21 PM	85 [2024-07-24 18:18:02.443]				Blocking	173.03	SQLAgent - TSQL JobStep (Job 0xC31233C4311984AA502BF9D0924748 : Step 1)	sqlserver	BEGIN TRAN
○	7/24/24, 6:21 PM	69 [2024-07-24 18:21:01.873]	85 [2024-07-24 18:18:02.443]	sales.dbo.BANK_PK_BANK [INDEX]	KEY	Blocked	173.03	SQLAgent - TSQL JobStep (Job 0x2BE2C65EE58F9A4FB5D10B3B9C206BEF : Step 1)	sqlserver	UPDATE [dbo]
🔍	7/24/24, 7:06 PM	81 [2024-07-24 19:03:03.11]				Blocking	172.77	SQLAgent - TSQL JobStep (Job 0xC31233C4311984AA502BF9D0924748 : Step 1)	sqlserver	BEGIN TRAN
○	7/24/24, 7:06 PM	74 [2024-07-24 19:06:02.147]	81 [2024-07-24 19:03:03.11]	sales.dbo.BANK_PK_BANK [INDEX]	KEY	Blocked	172.77	SQLAgent - TSQL JobStep (Job 0x2BE2C65EE58F9A4FB5D10B3B9C206BEF : Step 1)	sqlserver	UPDATE [dbo]
🔍	7/24/24, 6:42 PM	67 [2024-07-24 18:39:01.593]				Blocking	172.45	SQLAgent - TSQL JobStep (Job 0x2BE2C65EE58F9A4FB5D10B3B9C206BEF : Step 1)	sqlserver	BEGIN TRAN
○	7/24/24, 6:42 PM	78 [2024-07-24 18:42:02.487]	67 [2024-07-24 18:39:01.593]	sales.dbo.BANK_PK_BANK [INDEX]	KEY	Blocked	172.45	SQLAgent - TSQL JobStep (Job 0x45028A9858FE674999EDC6294A79162C : Step 1)	sqlserver	UPDATE [dbo]
🔍	7/24/24, 6:48 PM	81 [2024-07-24 18:45:02.64]				Blocking	172.37	SQLAgent - TSQL JobStep (Job 0xC31233C4311984AA502BF9D0924748 : Step 1)	sqlserver	BEGIN TRAN
○	7/24/24, 6:48 PM	78 [2024-07-24 18:48:02.477]	81 [2024-07-24 18:45:02.64]	sales.dbo.BANK_PK_BANK [INDEX]	KEY	Blocked	172.37	SQLAgent - TSQL JobStep (Job 0x2BE2C65EE58F9A4FB5D10B3B9C206BEF : Step 1)	sqlserver	UPDATE [dbo]
🔍	7/24/24, 6:18 PM	69 [2024-07-24 18:15:02.787]				Blocking	172.36	SQLAgent - TSQL JobStep (Job 0x45028A9858FE674999EDC6294A79162C : Step 1)	sqlserver	BEGIN TRAN
○	7/24/24, 6:18 PM	85 [2024-07-24 18:18:02.443]	69 [2024-07-24 18:15:02.787]	sales.dbo.BANK_PK_BANK [INDEX]	KEY	Blocked	172.35	SQLAgent - TSQL JobStep (Job 0xC31233C4311984AA502BF9D0924748 : Step 1)	sqlserver	UPDATE [dbo]
🔍	7/24/24, 6:12 PM	72 [2024-07-24 18:09:02.81]				Blocking	172.35	SQLAgent - TSQL JobStep (Job 0xC31233C4311984AA502BF9D0924748 : Step 1)	sqlserver	BEGIN TRAN
○	7/24/24, 6:12 PM	76 [2024-07-24 18:12:02.147]	72 [2024-07-24 18:09:02.81]	sales.dbo.BANK_PK_BANK [INDEX]	KEY	Blocked	172.35	SQLAgent - TSQL JobStep (Job 0x2BE2C65EE58F9A4FB5D10B3B9C206BEF : Step 1)	sqlserver	UPDATE [dbo]
🔍	7/24/24, 6:30 PM	69 [2024-07-24 18:27:03.04]				Blocking	172.27	SQLAgent - TSQL JobStep (Job 0xC31233C4311984AA502BF9D0924748 : Step 1)	sqlserver	BEGIN TRAN

# Activity Highlights

Databases > Overview > SQL PI

Time Period: Wednesday, Jul 24, 2024 06:08 PM - 07:08 PM 1 hour | Reports

SQL\_GCP\_DBaaS\_2019 | Overview | SQL PI Memory | Activity | Databases | Services | HADR | Logs | Configuration | User-defined

Workload CPU I/O Memory Network **Lock** Latch Log CLR Remote Provider XTP Other

Powered by SQL PI

View as PDF

### Performance Tree

Tops: 25

- Instance View
  - SQL Statements
  - TSQL Batches
  - Databases
  - Programs
  - Users
  - Client Machines
  - Context Infos
  - Command Types
  - Sessions
  - Locked Objects

### History | Advanced Analytics

Dimension Filter: Instance View

#### Resource Consumption

Baseline | Breakdown

#### Resource Breakdown

■ Lock Update 49.75%  
■ Non-Lock Ac... 50.25%

### Activity Highlights

The Instance consumed 5,762 seconds waiting for Lock. **49.75%** of its db-time spent waiting for **Lock Update**

#### Highlights

Statement/Program	Resource Breakdown	Top Wait Event
SQL Statement UPDATE [dbo].[BANK] SET [BANK_NAME] = LTRIM(R...	3,470.61	LCK_M_U 100 %
SQL Statement UPDATE [BB] set [B] = @1 is attributed to 22.41% of the instance Lock Wait	1,291.37	LCK_M_U 100 %
SQL Statement UPDATE [AA] set [A] = @1 is attributed to 19.01% of the instance Lock Wait	1,095.24	LCK_M_U 100 %
TSQL Batch BEGIN TRAN UPDATE [dbo].[BANK] SET [BANK_NAME...	2,262.37	LCK_M_U 100 %
TSQL Batch (@1 int)UPDATE [BB] set [B] = @1 is attributed to 22.41% of the instance Lock Wait	1,291.37	LCK_M_U 100 %
TSQL Batch BEGIN TRAN UPDATE [dbo].[BANK] SET [BANK_NAME...	1,208.24	LCK_M_U 100 %
Database sales is attributed to 101.66% of the instance Lock Wait	11,449.77	LCK_M_U 51 %
Program SQLAgent - TSQL JobStep (Job 0x2BE2C65EE58F9A... is attributed to 20.97% of the instance Lock Wait	1,208.24	LCK_M_U 100 %
Program SQLAgent - TSQL JobStep (Job 0xCC31233C431198... is attributed to 20.88% of the instance Lock Wait	1,202.74	LCK_M_U 100 %
Program SQLAgent - TSQL JobStep (Job 0x198CA7AC67AF5E... is attributed to 19.01% of the instance Lock Wait	1,096.25	LCK_M_U 100 %
User sqlserver is attributed to 101.66% of the instance Lock Wait	11,731.34	LCK_M_U 50 %

# SQL PI – Tune SQL

Query Insights > SQL PI Monday, Jul 22, 2024 04:54:35 PM - Now 1 hour

SQL\_GCP\_DBaaS\_2019 | Overview SQL PI Memory Activity Databases Services HADR Logs Configuration User-defined

Workload CPU I/O Memory Network Lock Latch Log CLR Remote Provider XTP Other

### Performance Tree

Instance View SQL Statements

- UPDATE [dbo].[BANK] SET [BANK\_NAME] = LTRIM(RTRIM(BANK\_NAME)) FROM BANK WHERE [BANK\_CODE] = '\*\*\* Removed by Foglight \*\*\*'
- UPDATE [B8] set [B] = @1
- UPDATE [AA] set [A] = @1
- INSERT INTO [dbo].[ORDL1] SELEC
- INSERT INTO [dbo].[ORDL3] SELEC
- INSERT INTO [dbo].[ORDL2] SELEC
- DELETE FROM [dbo].[ORDL3]
- DELETE FROM [dbo].[ORDL1]
- DELETE FROM [dbo].[ORDL2]
- SELECT [customer\_id], Sum(amount
- Insert Into #ResourceDB Select Dist
- SELECT TOP (@RowLimit) CONVERT
- Select Distinct A2.request\_session\_k
- insert into #temp\_trace select top 1
- Select Top 1 @DBID = B.[dbid] , @C
- insert into #temp\_trace select top 1
- select top 100 DatabaseName, FileN
- [@P0] bigint,@P1 int,@P2 int,@P3 m
- xp\_readerrorlog
- xp\_instance\_regread
- SELECT top (@P0) t2.spid, t2.login\_
- Select @DBName As DBName , DB\_
- DROP TABLE #QS\_sysfiles
- select create\_time from sys.dm\_xe\_

### Resource Consumption

UPDATE [dbo].[BANK] SET [BANK\_NAME] = LTRIM(RTRIM(BANK\_NAME)) FROM BANK WHERE [BANK\_CODE] = '\*\*\* Removed by Foglight \*\*\*'

### Resource Breakdown

Lock Wait 100.00%

### Active Time

Sum of all the active waits and cpu usage, equal to the session total activity within the current interval.

### Workload related Metrics

Select Metric View SQL Text Analyze Plan **Tune SQL** Compare

Metric	Resource	Total
Average SQL Response Time	Workload	175.22
CPU Usage	CPU	< 0.01
Executions	Workload	39.00
Lock Update	Lock	6,834.83
Lock Wait	Lock	6,834.83
Logical Reads	I/O	135.00
Row count	Workload	13.00
Wait Time Percent	Workload	100.00

# SQL Optimizer

Quest SQL Optimizer for SQL Server 10.1.2

Optimize SQL | Optimize Indexes | Find SQL | Scan SQL | Manage Plan Guides

SQL Rewrite | SQL Details | Compare | Report

JANISCLIENT.MSSQLSERVER01 (IMDEMO\janis) | DOT | <Default> | Custom | 4

```
UPDATE [dbo].[BANK]
SET [BANK_NAME] = LTRIM(RTRIM([BANK_NAME]))
FROM BANK
WHERE [BANK_CODE] + '!' >= '*** Removed by Foglight ***'
AND [BANK_CODE] <= '*** Removed by Foglight ***'
```

Schema Information

Summary

Object Used in SQL

- DOT.dbo.bank

Summary Details

Table Name	Table Owner	Type	Created Datetime	Cardinality	Pages
bank	dbo	user table	7/23/2024 2:52:37 PM	7764288	73996

Alternatives

0 faster found (19 not tested)

Show faster alternatives only

Scenario Name	Plan C...	Executions	Record Count	Status	Execution Elapsed Time	Test Run Label	Execution Elapsed Time	Total Elapsed Time	Response Time	Execution CPU Time	Logical Reads	Physical Reads	Scans	Read-Ahead Reads	Complie Elapsed Time	Complie CPU Time	Buffers Received	Bytes Received
Original	0.0165715	2	0		00:00:00.000	12:41:12 PM	00:00:00.000	00:00:00.000	00:00:00.001	00:00:00.000	3	0	0	0	00:00:00.000	00:00:00.000	4	18,501
Alt4	0.0165722	2	0		00:00:00.000	12:41:12 PM	00:00:00.000	00:00:00.000	00:00:00.000	00:00:00.000	3	0	1	0	00:00:00.000	00:00:00.000	4	22,107
Alt32	0.0165727	2	0		00:00:00.000	12:41:12 PM	00:00:00.000	00:00:00.000	00:00:00.000	00:00:00.015	3	0	0	0	00:00:00.000	00:00:00.000	4	22,495
Alt44	0.0165727	2	0		00:00:00.000	12:41:12 PM	00:00:00.000	00:00:00.000	00:00:00.000	00:00:00.000	3	0	1	0	00:00:00.000	00:00:00.000	5	24,783

Total 138 Selected 1



# SQL Optimizer

Optimize SQL | Optimize Indexes | Find SQL | Scan SQL | Manage Plan Guides

SQL Rewrite 1 x

SQL Rewrite | SQL Details | Compare | Report

JANISCLIENT.MSSQLSERVER01 (IMDEMO\janis) | DOT | <Default> | Custom | 4

### Alternatives

Scenario Name	Plan Cost	Executions	Record Count	Status	Execution Elapsed Time	Test Run Label	Execution Elapsed Time	Total Elapsed Time	Response Time	Execution CPU Time	Logical Reads	Physical Reads	Scans	Read-Ahead Reads	Compile Elapsed Time	Compile CPU Time	Buffers Received	Bytes Received
Original	0.0165715	2	0	✓	00:00:00.000	12:41:12 PM	00:00:00.000	00:00:00.000	00:00:00.001	00:00:00.000	3	0	0	0	00:00:00.000	00:00:00.000	4	18,501
Alt4	0.0165722	2	0	✓	00:00:00.000	12:41:12 PM	00:00:00.000	00:00:00.000	00:00:00.000	00:00:00.000	3	0	1	0	00:00:00.000	00:00:00.000	4	22,107
Alt3	111.9610...	2	0	✓	00:00:02.094	12:41:12 PM	00:00:02.094	00:00:02.094	00:00:02.045	00:00:07.891	73,996	0	5	0	00:00:00.000	00:00:00.000	8	50,349
Alt2	114.7170...	2	0	✓	00:00:00.695	12:41:12 PM	00:00:00.695	00:00:00.695	00:00:00.653	00:00:02.593	73,996	0	5	0	00:00:00.000	00:00:00.000	4	19,752
Alt1	24.63000...	2	0	✓	00:00:00.335	12:41:12 PM	00:00:00.335	00:00:00.335	00:00:00.326	00:00:01.313	24,212	0	5	0	00:00:00.000	00:00:00.000	6	33,459

### Comparison

Layout: SQL and Plan (Left-Right) | Swap Panes | Maximize

Original

```
UPDATE [dbo].[BANK]
SET [BANK_NAME] = LTRIM(RTRIM([BANK_NAME]))
FROM BANK
WHERE [BANK_CODE] <= '** Removed by Foglight **'
```

Alt4

```
UPDATE [dbo].[BANK]
SET [BANK_NAME] = LTRIM(RTRIM([BANK_NAME]))
FROM BANK
WHERE [BANK_CODE] <= '** Removed by Foglight **'
AND [BANK_CODE] <= '** Removed by Foglight **'
```

### Actual Plan / Estimated Plan

Plan	Act rows	Est rows	A/E Row...	Est cost
4 Compute Scalar Est cost %: 0.00; Est subtree cost: 0.006570; Est subtree cost %: 39.65; Est I/O cost: 0.000000; Est CPU cost: 0.000000; Est avg row size: 44; Est executions: 1;		1.00		0.000000
3 Nested Loops / Inner Join Act executions: 1;	0	1.00	-100.0%	0.000004
1 Index Seek [DOT].[dbo].[bank].[pk_bank2] Act executions: 1; Est cost %: 19.81; Est subtree cost: 0.003283; Est subtree cost %: 19.81; Est I/O cost: 0.003125; Est CPU cost: 0.000158; Est avg row size: 15; Est executions: 1;	0	1.00	-100.0%	0.003283
2 RID Lookup [DOT].[dbo].[bank] Act executions: 0;	0	1.00	-100.0%	0.003283

Step 6 Rows were updated by the UPDATE statement. Total 138 Selected 1

Plan	Act rows	Est rows	A/E Row...	Est cost
4 Nested Loops / Inner Join Act executions: 1;	0	1.00	-100.0%	0.000004
2 Compute Scalar Est cost %: 0.00; Est subtree cost: 0.003284; Est subtree cost %: 19.81; Est I/O cost: 0.000000; Est CPU cost: 0.000001; Est avg row size: 15; Est executions: 1;		1.00		0.000001
1 Index Seek [DOT].[dbo].[bank].[pk_bank2] Act executions: 1; Est cost %: 19.81; Est subtree cost: 0.003283; Est subtree cost %: 19.81; Est I/O cost: 0.003125; Est CPU cost: 0.000158; Est avg row size: 25; Est executions: 1;	0	1.00	-100.0%	0.003283
3 RID Lookup [DOT].[dbo].[bank] Act executions: 0;	0	1.00	-100.0%	0.003283

Step 7 Rows were updated by the UPDATE statement.

# Change Heap to Clustered Index

```
CREATE TABLE [dbo].[bank](
  [bank_code] [varchar](20),
  [bank_name] [varchar](50) NULL,
  [address] [varchar](50) NULL,
  [state] [varchar](2) NULL,
  [zip] [varchar](10) NULL,
  CONSTRAINT [PK_BANK] PRIMARY KEY CLUSTERED
  ([bank_code])
)ON [PRIMARY]

insert into bank select * from bank_old;
```

7/24/24, 8:09 PM | Cache Collected | 0x0600070086d9220d20db61036b010000010000...

Date | Type | Plan Handle

Statement

```
UPDATE [dbo].[BANK] SET [BANK_NAME] = LTRIM(RTRIM(BANK_NAME)) FROM BANK WHERE [BANK_CODE] = '*** Removed by Foglight ***'
```

Before

Plan Analysis

Total cost: 0.0165715 | Total I/O cost: 0.0162500 | Total CPU cost: 0.0003215

Plan Details | Operator Analysis | Object Analysis

Operator	Object	Operator Cost	Subtree Cost	I/O Cost	CPU Cost	Rows	Arguments
Table Update (Update)	dbo.bank	60.35 %	0.0165715	0.0100000	0.0000010	1	[DOT].[dbo].[bank].[bank_name] = [Expr1003]
Compute Scalar		0.00 %	0.0065705	0.0000000	0.0000001	1	
Nested Loops (Inner Join)		0.03 %	0.0065704	0.0000000	0.0000042	1	
Index Seek	dbo.bank.pk_bank2	19.81 %	0.0032831	0.0031250	0.0001581	1	
RID Lookup	dbo.bank	19.81 %	0.0032831	0.0031250	0.0001581	1	

7/24/24, 8:04 PM | Cache Collected | 0x060007004ad3961f50d9da446b010000010000...

Date | Type | Plan Handle

Statement

```
UPDATE [dbo].[BANK] SET [BANK_NAME] = LTRIM(RTRIM(BANK_NAME)) FROM BANK WHERE [BANK_CODE] = '*** Removed by Foglight ***'
```

After

Plan Analysis

Total cost: 0.0132842 | Total I/O cost: 0.0100000 | Total CPU cost: 0.0000010

Plan Details | Operator Analysis | Object Analysis

Operator	Object	Operator Cost	Subtree Cost	I/O Cost	CPU Cost	Rows	Arguments
Clustered Index Update (Update)	dbo.bank.PK_BANK	100.00 %	0.0132842	0.0100000	0.0000010	1	[DOT].[dbo].[bank].[bank_name] = [Expr1002]

# Create Reusable Stored Procedure

Home Databases > Overview > SQL PI

Thursday, Jul 25, 2024 05:00:08 PM - Now 1 hour

JANISCLIENT-MSSQLSERVER01 Overview SQL PI Memory Activity Databases Services HADR Logs Configuration User-defined

Workload CPU I/O Memory Network Lock Latch Log CLR Remote Provider XTP Other

### Performance Tree

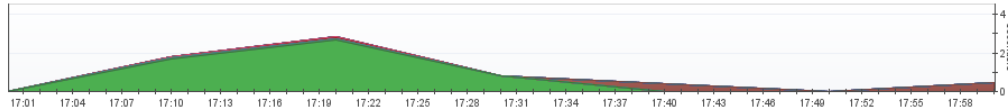
- Instance View
- SQL Statements
- TSQL Batches
- Databases
- Programs
- Users
- Client Machines
- Context Infos
- Command Types
- Sessions
- Locked Objects
- Objects I/O
- Files
- Disks

Tops: 25

History | Advanced Analytics

Dimension Filter: Instance View

### Resource Consumption



Baseline | Breakdown

### Resource Breakdown



CPU Usage	83.40%
Log Wait	12.67%
CPU Wait	1.28%
Latch Wait	1.12%

```
CREATE PROCEDURE bank_name_upd
    @bc varchar(20)
AS
UPDATE [dbo].[BANK] SET [BANK_NAME] = LTRIM(RTRIM([BANK_NAME]))
FROM BANK
WHERE bank_code = @bc;
go
```

Overview | Blocking History | Activity Highlights

### Active Time



Sum of all the active waits and cpu usage, equal to the session total activity within the current interval.

### Workload related Metrics

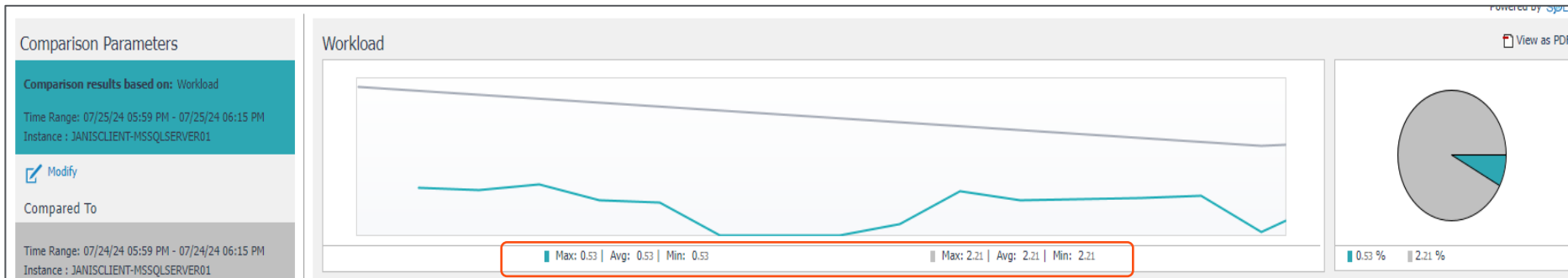
Select Metric Compare

Metric	Resource	Total
Active Time	Workload	3,750.87
Average SQL Response Time	Workload	< 0.01
Batches Rate	Workload	1.86
CPU Usage	CPU	3,128.37
Executions	Workload	175,548.00
Logins Rate	Workload	0.03
Wait Time Percent	Workload	16.60



# Comparison of Changes = Improvement

- **Heap to Clustered Index (on bank\_code)**
- **Change query from passing literals to parameters**
  - With cached stored procedure



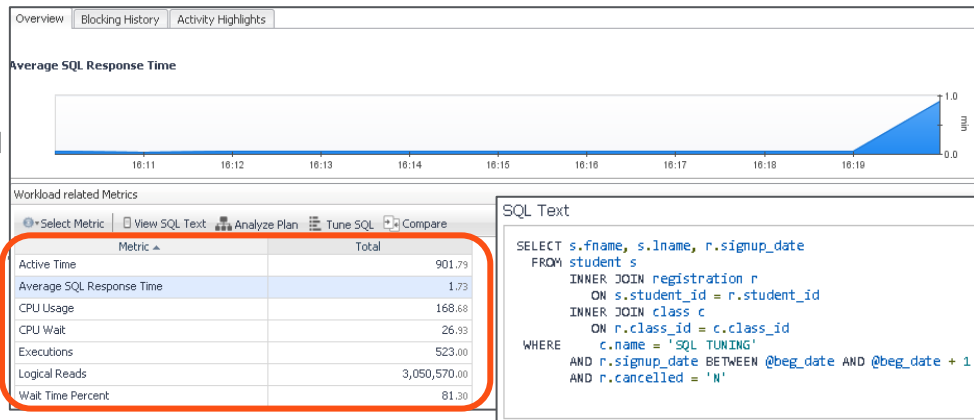
# Another SQL Optimizer Example

Who registered yesterday for SQL Tuning?

```
SELECT s.fname, s.lname, r.signup_date
FROM student s
     INNER JOIN registration r ON s.student_id = r.student_id
     INNER JOIN class c ON r.class_id = c.class_id
WHERE c.name = 'SQL TUNING'
AND r.signup_date BETWEEN :beg_date AND :beg_date + 1
AND r.cancelled = 'N'
```

Execution Stats – 3,050,570 Logical Reads  
Execution Time – Approx. 2 seconds  
Wait Type – ASYNC\_NETWORK\_IO

10 Minute Time slice



Category	Event Name	% of Total Wait Time	Wait Time
Network Wait	ASYNC_NETWORK_IO	77.85	702.04
CPU Wait	SOS_SCHEDULER_YIELD	2.99	26.93
I/O Wait	PAGEIOLATCH_SH	1.19	10.77
Memory Wait	MEMORY_ALLOCATION_EXT	0.36	3.21
Memory Wait	RESERVED_MEMORY_ALLOCATION_EXT	0.07	0.59

# Sql Optimizer Indexes .011 to .001

The screenshot displays the Toad SQL Optimizer interface. The main window shows a query with several inner joins and a filter condition. Below the query, the 'Alternatives' section shows a comparison between the 'Original' query and several indexed alternatives. A table at the bottom provides detailed execution statistics for each alternative.

```
SELECT s.fname,
       s.lname,
       r.signup_date
FROM student s
     INNER JOIN registration r
           ON s.student_id = r.student_id
     INNER JOIN class c
           ON r.class_id = c.class_id
WHERE c.name = @c1_name
     AND r.signup_date BETWEEN @beg_date and @beg_date + 1
     AND r.cancelled = 'N'
```

Alternatives:

- Not optimized (Click to start)
- Index2
- Index
- User Alternative
- Test Run - All

General		Execution Statistics																
Scenario Name	Plan Cost	Executions	Record Count	Status	Execution Elapsed Time	Test Run Label	Execution Elapsed Time	Total Elapsed Time	Response Time	Execution CPU Time	Logical Reads	Physical Reads	Scans	Read-Ahead Reads	Compile Elapsed Time	Compile CPU Time	Buffers Received	Bytes Received
Original	0.6506120	2	20		00:00:00.011	1:35:19 PM	00:00:00.011	00:00:00.011	00:00:00.013	00:00:00.016	477	0	2	0	00:00:00.000	00:00:00.000	7	35,147
Index1	0.2182100	2	20		00:00:00.006	1:35:19 PM	00:00:00.006	00:00:00.006	00:00:00.008	00:00:00.000	97	0	2	0	00:00:00.000	00:00:00.000	7	34,311
Index2	0.0557958	2	20		00:00:00.001	1:35:19 PM	00:00:00.001	00:00:00.001	00:00:00.004	00:00:00.000	61	0	3	0	00:00:00.000	00:00:00.000	6	30,111
Index3	0.5868010	2	20		00:00:00.010	1:35:19 PM	00:00:00.010	00:00:00.010	00:00:00.013	00:00:00.000	425	0	2	0	00:00:00.000	00:00:00.000	7	35,489
Index5	0.0956162	2	20		00:00:00.001	1:35:19 PM	00:00:00.001	00:00:00.001	00:00:00.004	00:00:00.000	121	0	3	0	00:00:00.000	00:00:00.000	7	40,069
Index6	0.2077420	2	20		00:00:00.004	1:35:19 PM	00:00:00.004	00:00:00.004	00:00:00.006	00:00:00.000	84	0	2	0	00:00:00.000	00:00:00.000	7	34,391
Index7	0.0453280	2	20		00:00:00.001	1:35:19 PM	00:00:00.001	00:00:00.001	00:00:00.001	00:00:00.000	48	0	3	0	00:00:00.000	00:00:00.000	6	30,185
Index9	0.0851484	2	20		00:00:00.001	1:35:19 PM	00:00:00.001	00:00:00.001	00:00:00.004	00:00:00.000	108	0	3	0	00:00:00.000	00:00:00.000	7	40,147

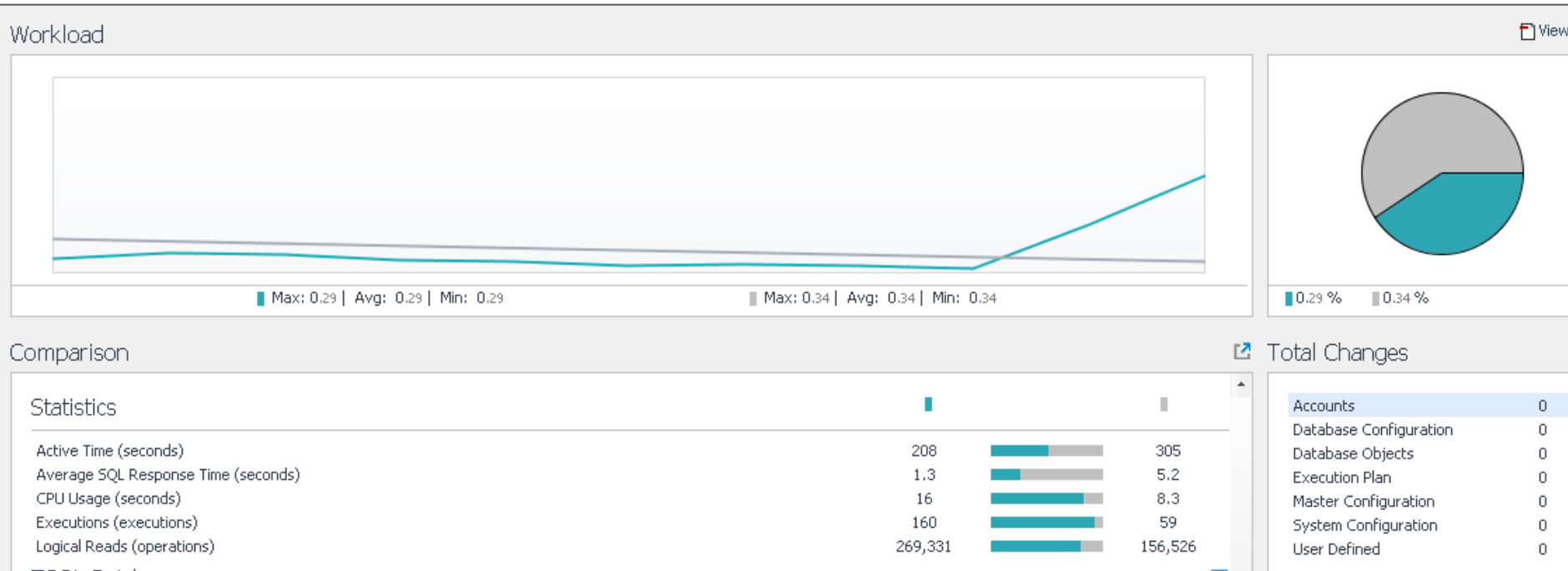
# Sql Optimizer Indexes .011 to .001

The screenshot displays the SQL Optimizer interface. The top menu includes 'Optimize SQL', 'Optimize Indexes', 'Find SQL', 'Scan SQL', 'Manage Plan Guides', and 'Community'. The main window shows the configuration for an index named 'Index2' on the 'registration' table. The indexed columns are 'class\_id', 'cancelled', and 'signup\_date', all in ascending order. The index type is set to 'UNIQUE'. Below the configuration, the 'Alternatives' section shows a list of execution alternatives with their respective statistics.

General		Execution Statistics																
Scenario Name	Plan Cost	Executions	Record Count	Status	Execution Elapsed Time	Test Run Label	Execution Elapsed Time	Total Elapsed Time	Response Time	Execution CPU Time	Logical Reads	Physical Reads	Scans	Read-Ahead Reads	Compile Elapsed Time	Compile CPU Time	Buffers Received	Bytes Received
Alt32	0.7938320	2	20	✓	00:00:00.019	1:35:19 PM	00:00:00.019	00:00:00.019	00:00:00.021	00:00:00.031	584	0	3	0	00:00:00.000	00:00:00.000	7	38,041
Alt33	0.6281010	2	20	✓	00:00:00.011	1:35:19 PM	00:00:00.011	00:00:00.011	00:00:00.013	00:00:00.016	477	0	2	0	00:00:00.000	00:00:00.000	8	42,807
Alt34	0.8185790	2	20	✓	00:00:00.026	1:35:19 PM	00:00:00.026	00:00:00.026	00:00:00.028	00:00:00.031	579	0	3	0	00:00:00.000	00:00:00.000	8	43,693
Index1	0.2182100	2	20	✓	00:00:00.006	1:35:19 PM	00:00:00.006	00:00:00.006	00:00:00.008	00:00:00.000	97	0	2	0	00:00:00.000	00:00:00.000	7	34,311
<b>Index2</b>	<b>0.0657958</b>	<b>2</b>	<b>20</b>	<b>⚠</b>	<b>00:00:00.001</b>	<b>1:35:19 PM</b>	<b>00:00:00.001</b>	<b>00:00:00.001</b>	<b>00:00:00.004</b>	<b>00:00:00.000</b>	<b>61</b>	<b>0</b>	<b>3</b>	<b>0</b>	<b>00:00:00.000</b>	<b>00:00:00.000</b>	<b>6</b>	<b>30,111</b>
Index3	0.5868010	2	20	✓	00:00:00.010	1:35:19 PM	00:00:00.010	00:00:00.010	00:00:00.013	00:00:00.000	425	0	2	0	00:00:00.000	00:00:00.000	7	35,489
Index4	0.6401440	2	20	✓	00:00:00.012	1:35:19 PM	00:00:00.012	00:00:00.012	00:00:00.015	00:00:00.016	464	0	2	0	00:00:00.000	00:00:00.000	7	35,065
Index5	0.0956162	2	20	✓	00:00:00.001	1:35:19 PM	00:00:00.001	00:00:00.001	00:00:00.004	00:00:00.000	121	0	3	0	00:00:00.000	00:00:00.000	7	40,069
Index6	0.2077420	2	20	✓	00:00:00.004	1:35:19 PM	00:00:00.004	00:00:00.004	00:00:00.006	00:00:00.000	84	0	2	0	00:00:00.000	00:00:00.000	7	34,391
Index7	0.0453280	2	20	✓	00:00:00.001	1:35:19 PM	00:00:00.001	00:00:00.001	<b>00:00:00.003</b>	00:00:00.000	<b>48</b>	0	3	0	00:00:00.000	00:00:00.000	6	30,185
Index8	0.5763330	2	20	✓	00:00:00.011	1:35:19 PM	00:00:00.011	00:00:00.011	00:00:00.014	00:00:00.016	412	0	2	0	00:00:00.000	00:00:00.000	7	35,565
Index9	0.0851484	2	20	✓	00:00:00.001	1:35:19 PM	00:00:00.001	00:00:00.001	00:00:00.004	00:00:00.000	108	0	3	0	00:00:00.000	00:00:00.000	7	40,147

# Improvement?

- 4x faster, 3x throughput









- 
- 1. Focus on the queries that impact performance the most**
    - Query Insights
  - 2. Review baselines & changes**
    - Utilize compare & change tracking features
  - 3. Review key metrics & query execution plans**
    - Response time, executions, logical\_reads & row\_counts
  - 4. Use wait types to identify bottlenecks**
    - They give clues to best tuning approach
  - 5. Determine if it's a poorly written query or database design flaw**
  - 6. Consider tuning with SQL Optimizer**

# More Performance Tuning Help...

Quest

Resources Blogs Forums    

Quest

Products

Solutions

Support & Services

Partners

About

Free Trials

Request Pricing

## The Zombie Survival Guide to Database Performance Tuning

You know that feeling of banging your head against a SQL query or an entire database for minutes – then hours – and watching your day vanish and feeling your brain turn to mush?

Nobody wants you to be a database zombie.

That's why we've pulled together a few decades of our experience in improving database performance to bring you this guide. You'll find six steps on database tuning, with plenty for both newbies and old hands to latch onto and learn from. We've also included a section with variations for Microsoft SQL Server, Oracle Database, MySQL and PostgreSQL.

[Zombie Survival Guide](#)



Download Your Free E-book

Business Email

Download E-book

By downloading, you are registering to receive marketing email from us. To opt-out, follow steps described in our [Privacy Policy](#).

reCAPTCHA protects this site. See Google's [Privacy Policy](#) and [Terms of Use](#).

# Thank You



Looking Forward to Seeing You There

**Register Today!**

Foglight 101 Series: Episode 5

**Exploring REST API Functionality**

Clay Jackson

May 21<sup>st</sup>

**Quest**  
Where Next Meets Now.