



Developer's Guide to Tuning Someone Else's SQL Server

Post-conference Workshop #05

SQLintersection Fall 2014

Workshop: Friday, November 14, 2014

written and presented by

Brent Ozar, Jeremiah Peschka, Kendra Little

<http://www.brentozar.com>



BRENT OZAR
UNLIMITED®



BRENT OZAR
UNLIMITED®

Developer's Guide to Tuning Someone Else's SQL Server

Brent Ozar, Jeremiah Peschka, Kendra Little
SQL Intersection, November 2014, Las Vegas

BRENT OZAR



Microsoft
CERTIFIED
Master

SQL Servers 2008

DOUG LANE



Microsoft
CERTIFIED
Professional

KENDRA LITTLE



Microsoft
CERTIFIED
Master

SQL Servers 2008

JEREMIAH PESCHKA



Microsoft
CERTIFIED
IT Professional

Database Developer 2008
Database Administrator 2008

JES BORLAND



Microsoft
CERTIFIED
Professional



X4

Microsoft Partner
Silver Data Platform



About you

Your company sells software that relies on SQL Server, and your customers want help with performance and reliability.

You have some developers on staff, plus a handful of help desk support folks, but none of them are DBAs, and none have had SQL Server training.

Your customers call up complaining about slow queries, and you're tired of the calls being escalated to you.



© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

What you'll learn today

How to identify the SQL Server's real bottleneck

How to prove that it's not your code, or

If it is your code, how to fix it quickly with indexes

How to communicate it in a way customers will love

What hardware and HA/DR solutions to recommend to your clients

How to know if your customers will be happy with virtualization

You'll even get a First Responder Kit for your help desk team and developers to quickly triage the customer's performance issue and get them back on track.



© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Today's agenda

Introduction

Assessing Your Workload – Kendra

Indexing to Fix Performance Problems – Kendra

T-SQL Anti-patterns – Jeremiah

How to Build a Health Check Report – Jeremiah

RPO & RTO Basics – Brent

ISV Best Practices - Brent



5

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

FAQ, BBQ, LOL

We're at level 200-300 today

- But ask your questions no matter what!
- Look for your “Montel”

These chairs get old!

- Get up and walk around the back/sides of the room
- We will take frequent breaks (not just the scheduled ones)
- Watch the timer to know when to get back in your seat

We love questions.

Asking questions can win you a prize.



6

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Tell us how you really feel.

Use EventBoard to fill out a session evaluation.

We want to improve, and your feedback helps.

Questions before we get started?



© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>



BRENT OZAR
UNLIMITED®

Assess a Workload and Find Out What's Slow

How to assess a workload

Two part methodology

1. Measure and interpret bottlenecks using SQL Server wait statistics
2. Identify top queries and assess how they relate to the bottlenecks

Measure bottlenecks with wait statistics



© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

SQL Server loves cooperation



© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Meet the SQLOS

SQL Server is uppity– it takes over work from the OS

Why not let Windows do the work?

- Windows now prevents any single thread from taking over
- Threads get a time-slice, then Windows pulls ‘em off the processor (this thread has been pre-empted!)

SQL Server is different: threads co-operatively yield to each other

SQL “Schedulers” manage the process of putting threads into action when another thread yields



© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Why yield?

CPU is limited

When a query is waiting on another resource, it doesn't need to be active on the CPU



© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Executing != Running

An executing query may be waiting:

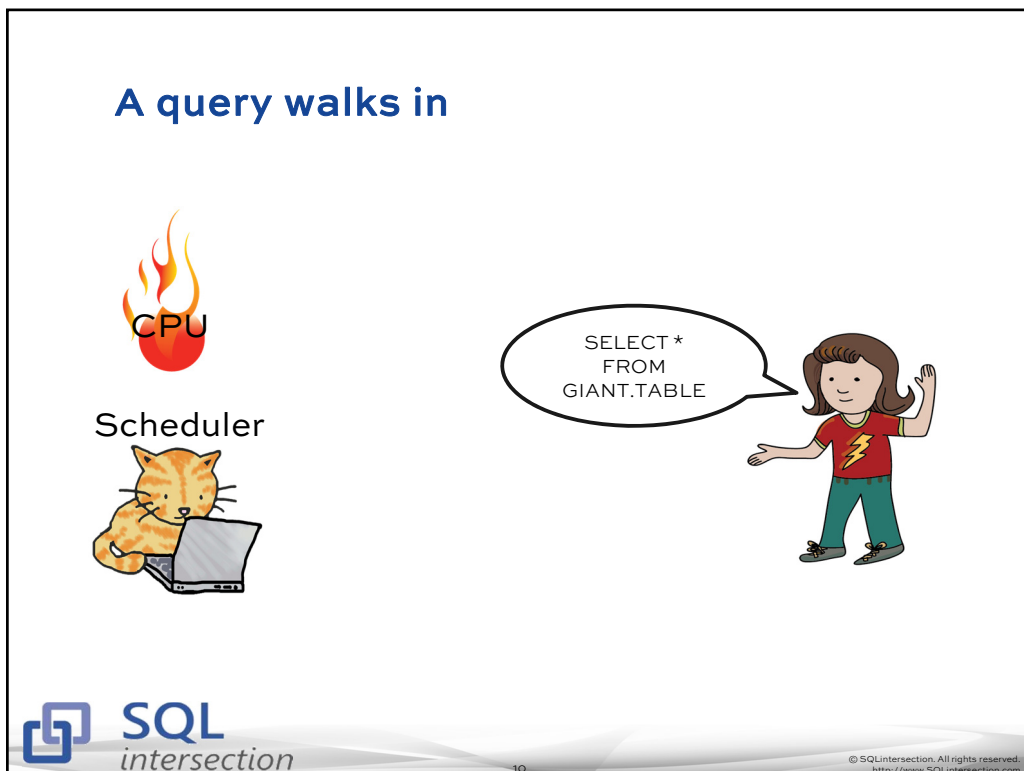
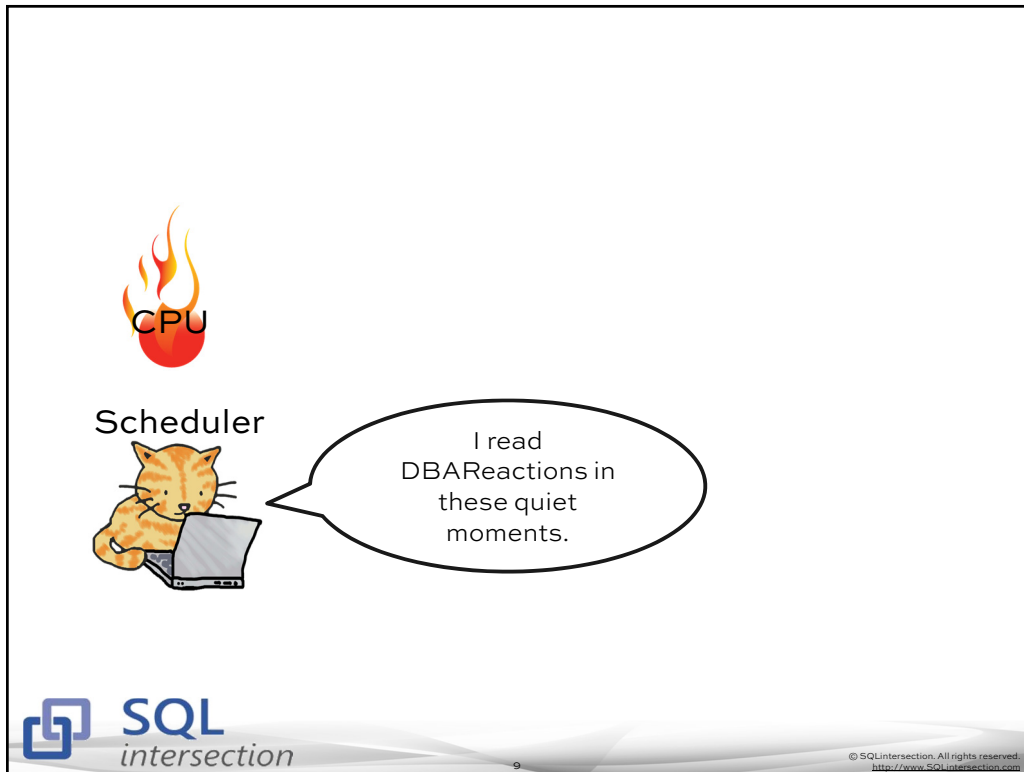
- For a resource (a lock, a page from storage, etc)
- For a signal/CPU (if it's ready to go again)

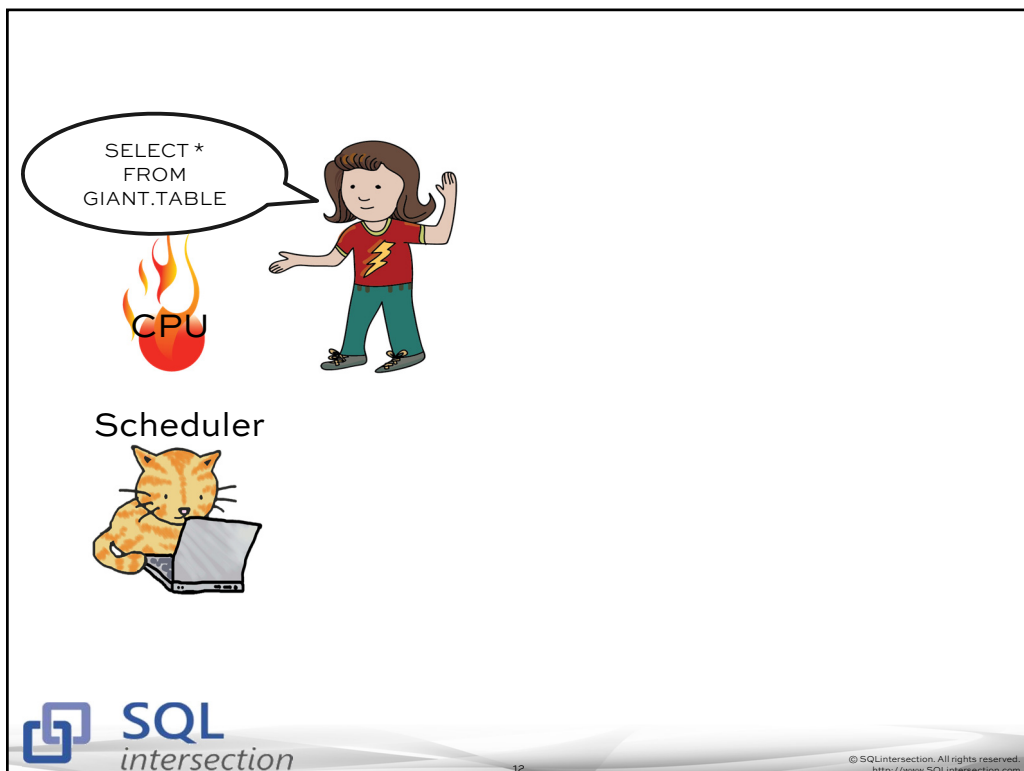
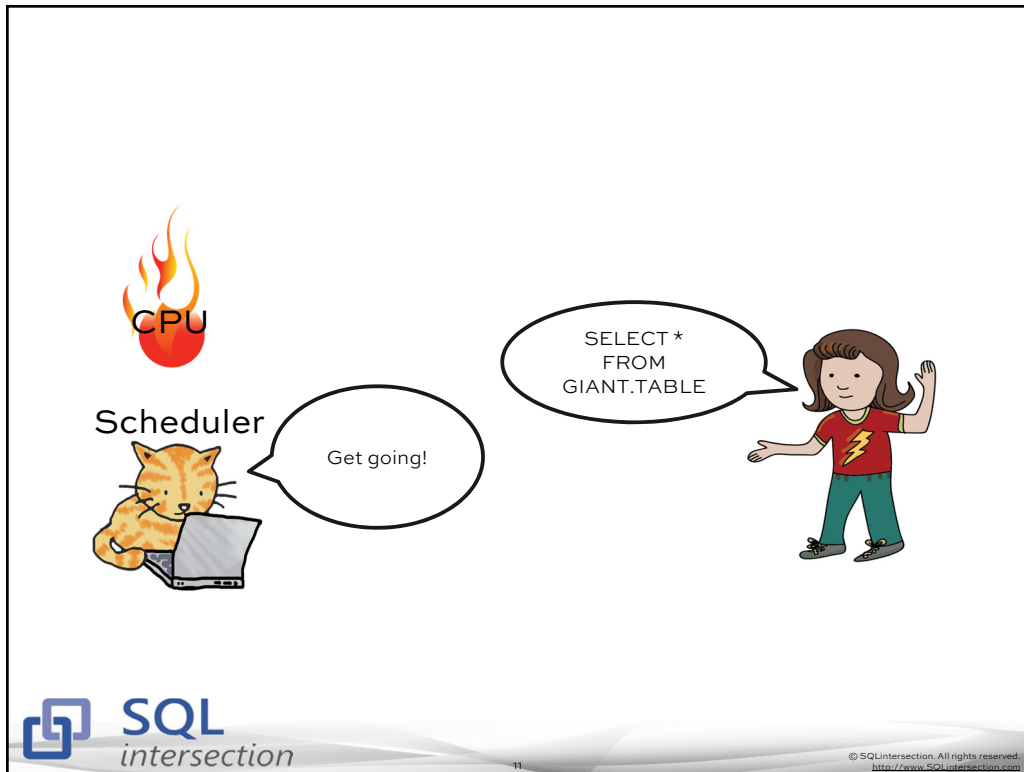
SQL Server tracks:

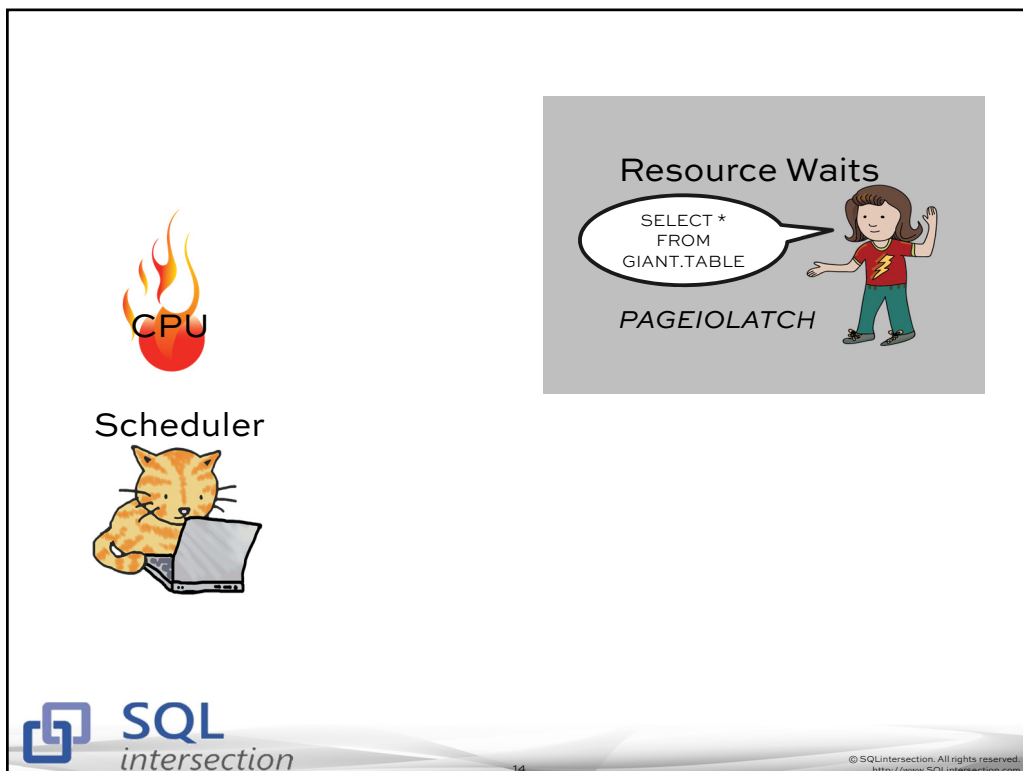
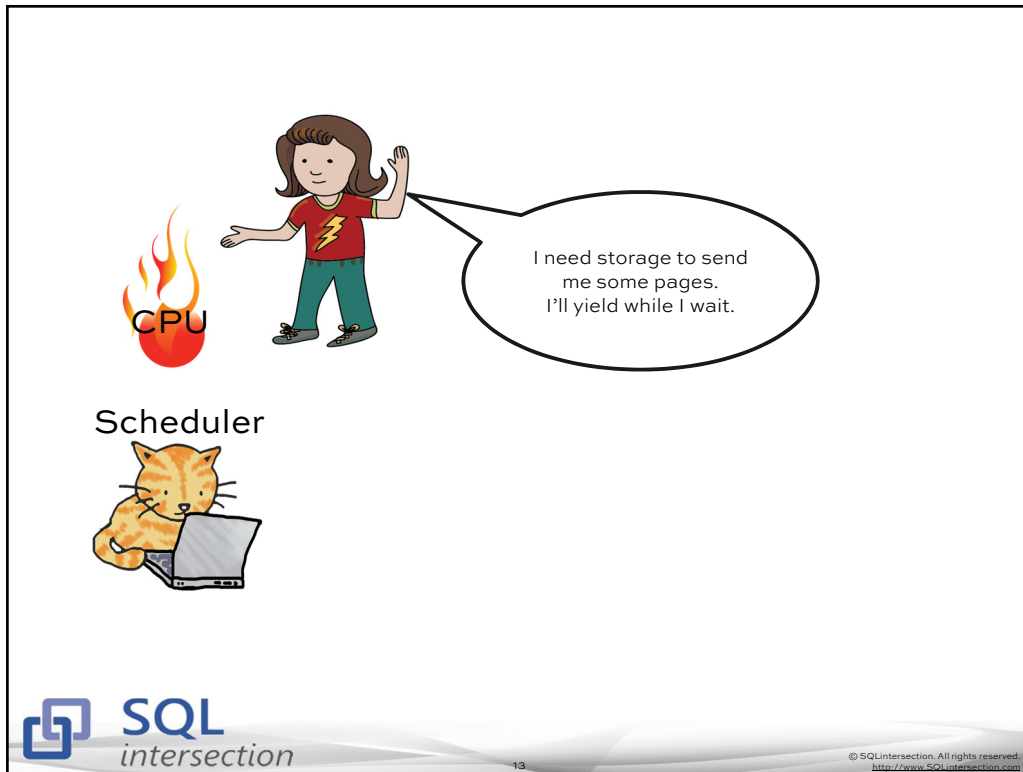
- What queries have run
- What execution plans they used
- How long queries were waiting
- What they were waiting for

Cat pictures will make this clearer

Let's look at an example







A new query...



Scheduler



Resource Waits

SELECT *
FROM
GIANT.TABLE

PAGEIOLATCH



SELECT COL1
FROM
GIANT.TABLE
WHERE COL2='y'



Scheduler



New guy, it's
all you.

Resource Waits

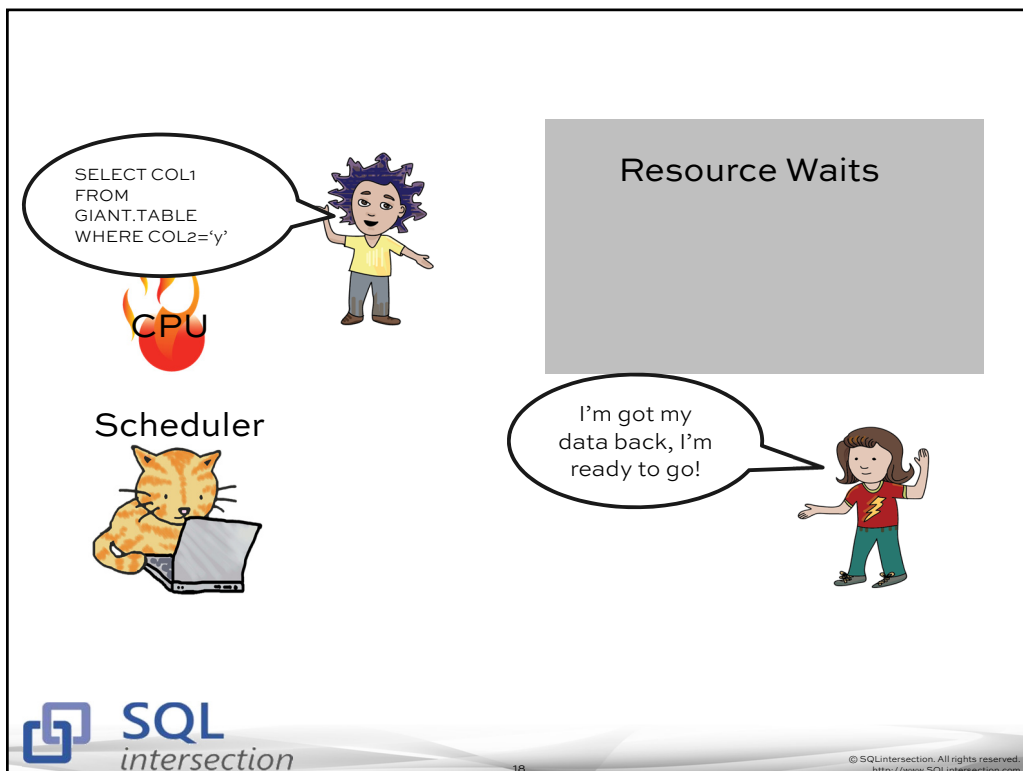
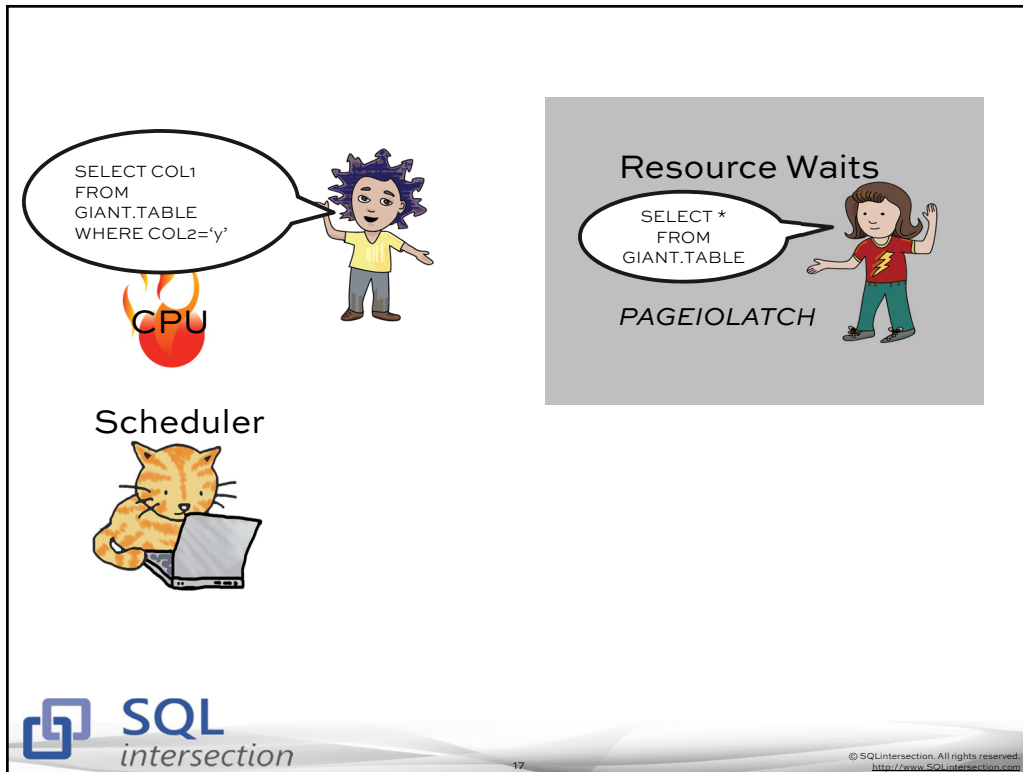
SELECT *
FROM
GIANT.TABLE

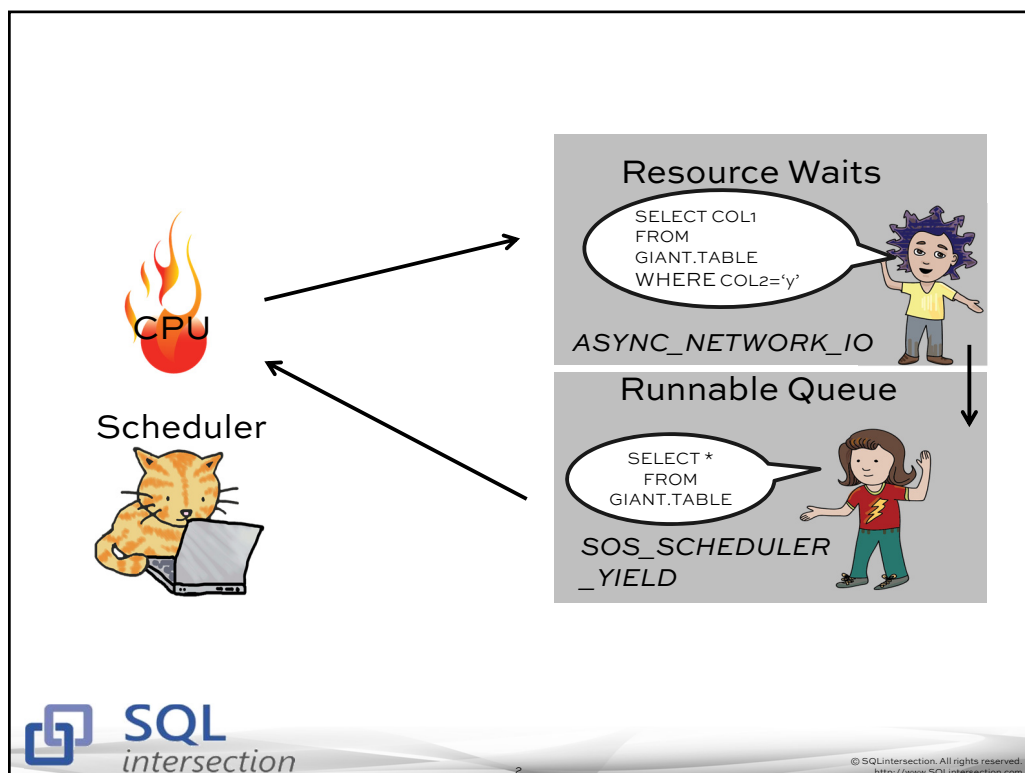
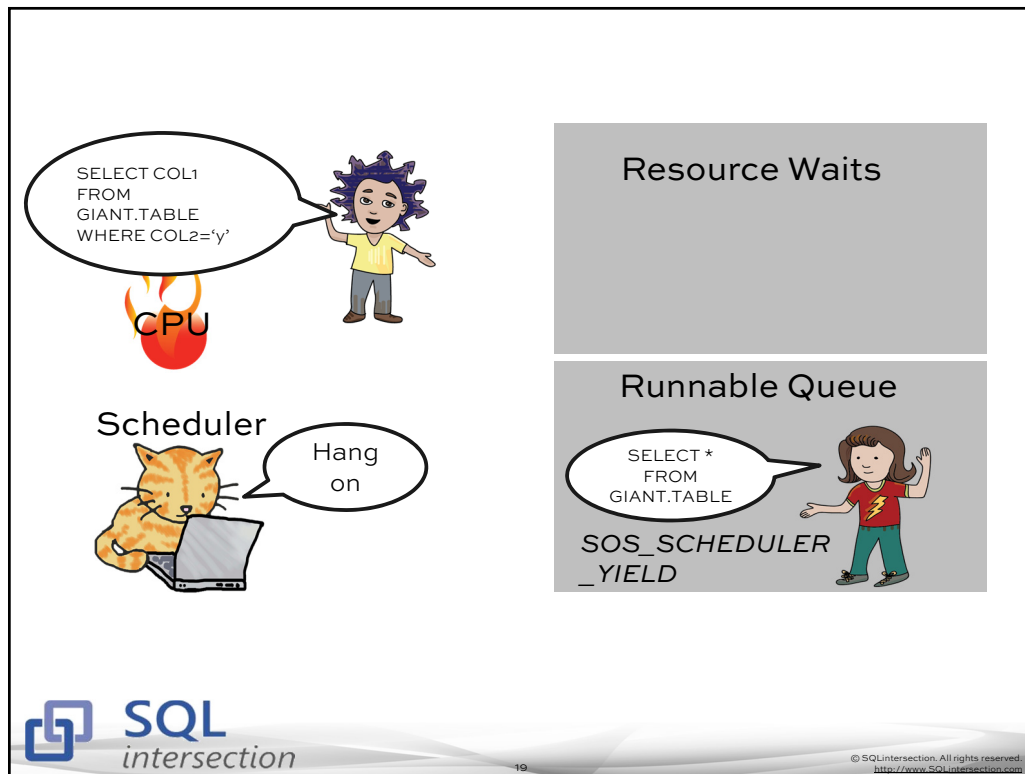
PAGEIOLATCH



SELECT COL1
FROM
GIANT.TABLE
WHERE COL2='y'







Waits identify SQL Server's bottlenecks

Is your server slowed down by....

- Lock waits?
- Storage waits?
- Delays by the application consuming data?
- CPU related waits?

Dynamic Management views:

- sys.dm_os_wait_stats ← Aggregate waits
- sys.dm_os_waiting_tasks ← Who is waiting on what (*right now*)

How to track wait stats and find top queries with free tools

Healthy vs. unhealthy monitoring

We're fans of:

- Tracking waits with a mature monitoring tool
- Taking a sample of waits and activity over a short duration with sp_AskBrent® (or your favorite script)
- Looking at queries running this instant and their related waits with Adam Machanic's sp_WhoIsActive
- Periodically reviewing top queries with sp_BlitzCache® (or your favorite script)

We're not big fans of:

- Automating frequent queries of DMVs for data you're not actively using
- The built-in Activity Monitor in SQL Server Management Studio

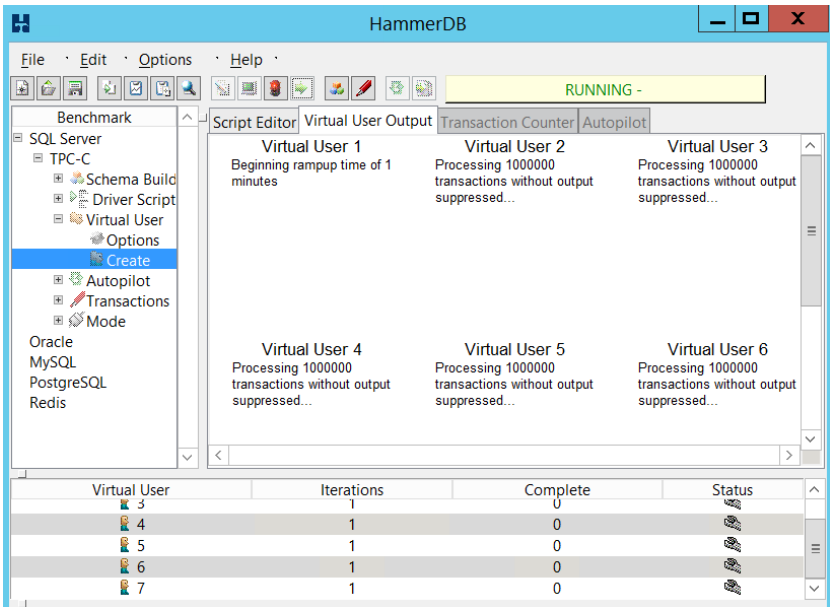


© SQLIntersection. All rights reserved.
<http://www.SQLIntersection.com>

Demo: The terrible tempdb workload



© SQLIntersection. All rights reserved.
<http://www.SQLIntersection.com>



The screenshot shows the HammerDB application window. The title bar is "HammerDB". The menu bar includes "File", "Edit", "Options", and "Help". The toolbar contains various icons. The main window is divided into several panes. On the left is a "Benchmark" tree view showing a hierarchy: "SQL Server" > "TPC-C" > "Schema Build" > "Driver Script" > "Virtual User" > "Options" > "Create" > "Autopilot" > "Transactions" > "Mode". Below this are "Oracle", "MySQL", "PostgreSQL", and "Redis". The main area is titled "Script Editor" and "Virtual User Output". It shows six virtual users: "Virtual User 1" (Beginning rampup time of 1 minutes), "Virtual User 2" (Processing 1000000 transactions without output suppressed...), "Virtual User 3" (Processing 1000000 transactions without output suppressed...), "Virtual User 4" (Processing 1000000 transactions without output suppressed...), "Virtual User 5" (Processing 1000000 transactions without output suppressed...), and "Virtual User 6" (Processing 1000000 transactions without output suppressed...). At the bottom is a table with columns: "Virtual User", "Iterations", "Complete", and "Status".

Virtual User	Iterations	Complete	Status
3	1	0	
4	1	0	
5	1	0	
6	1	0	
7	1	0	

SQL intersection

© SQLIntersection. All rights reserved.
<http://www.SQLIntersection.com>

1. Query waits since startup

```
/* Pseudo-code for readability */
/* The full query is in your course downloads */

SELECT TOP 25
  /* Wait Type, Metrics */
FROM sys.dm_os_wait_stats os
LEFT JOIN
  (VALUES('IGNORE_ME')) /*waits to ignore list here*/)
  AS iw (wait_type) on      os.wait_type=iw.wait_type
WHERE iw.wait_type is null
ORDER BY [Wait ms] DESC;
```

Waits since startup

Wait Type	Wait ms	% Wait Time	# Waits	Avg Wait ms	Sampled
CXPACKET	14,214,141	58.3	38,713,761	0.4	9/23/14 8:10 AM
PAGELATCH_UP	8,020,989	32.9	12,886,674	0.6	9/23/14 8:10 AM
PAGELATCH_SH	1,586,896	6.5	10,703,506	0.1	9/23/14 8:10 AM
LCK_M_X	243,721	1	11,212	21.7	9/23/14 8:10 AM
LCK_M_S	155,381	0.6	11,690	13.3	9/23/14 8:10 AM

Waits this instant: sp_WhoIsActive

```
EXEC dbo.sp_WhoIsActive;
GO
```

	dd hh:mm:ss.mss	session...	sql_text	login_na...	wait_info
1	00 00:20:12.073	55	<?query -- select cntr_value from sys.dm_os_perf...	sa	NULL
2	00 00:00:00.390	60	<?query -- SELECT TOP(5000) a.name, replicat...	sa	(199ms)LCK_M_X
3	00 00:00:00.373	61	<?query -- SELECT TOP(5000) a.name, replicat...	sa	(2ms)PAGELATCH_UP:tempdb:1(PFS)
4	00 00:00:00.373	59	<?query -- SELECT TOP(5000) a.name, replicat...	sa	(2ms)PAGELATCH_UP:tempdb:1(GAM)
5	00 00:00:00.373	56	<?query -- SELECT TOP(5000) a.name, replicat...	sa	(198ms)LCK_M_X
6	00 00:00:00.356	57	<?query -- SELECT TOP(5000) a.name, replicat...	sa	NULL
7	00 00:00:00.060	58	<?query -- SELECT TOP(5000) a.name, replicat...	sa	NULL

Waits over a sample period: sp_AskBrent

```
-- BrentOzar.com/AskBrent  
EXEC dbo.sp_AskBrent @ExpertMode=1, @Seconds=10;  
GO
```

100 % < <

Results Messages

	Priority	FindingsGroup	Finding	URL
1	0	sp_AskBrent (TM) v10 as of Jan 22 2014 12:00AM	From Brent Ozar Unlimited	http://www.BrentOzar.com/askbrent/
2	200	Wait Stats	CXPACKET	http://www.brentozar.com/sql/wait-stats/#CXPACKET
3	200	Wait Stats	PAGELATCH_SH	http://www.brentozar.com/sql/wait-stats/#PAGELATCH_SH
4	200	Wait Stats	PAGELATCH_UP	http://www.brentozar.com/sql/wait-stats/#PAGELATCH_UP
5	210	Query Stats	Most Resource-Intensive Queries	http://BrentOzar.com/go/topqueries
6	255	Thanks!	From Brent Ozar Unlimited	http://www.BrentOzar.com/askbrent/

< III

	Pattern	Sample Ended	Seconds Sam...	wait_type	Wait Time (Secon...	Signal Wait Time (Secon...	Percent Signal Wa
1	WAIT STATS	2014-09-23 08:20:31.230	10	CXPACKET	177.8	17.4	9.8
2	WAIT STATS	2014-09-23 08:20:31.230	10	PAGELATCH_UP	98.9	10.4	10.5
3	WAIT STATS	2014-09-23 08:20:31.230	10	PAGELATCH_SH	19.3	8.5	44.0
4	WAIT STATS	2014-09-23 08:20:31.230	10	LCK_M_X	4.5	0.0	0.0
5	WAIT STATS	2014-09-23 08:20:31.230	10	LCK_M_S	2.8	0.0	0.0
6	WAIT STATS	2014-09-23 08:20:31.230	10	LATCH_EX	1.6	0.7	43.8



© SQLIntersection. All rights reserved.
<http://www.SQLintersection.com>

Top queries

```
exec sp_BlitzCache;  
GO
```

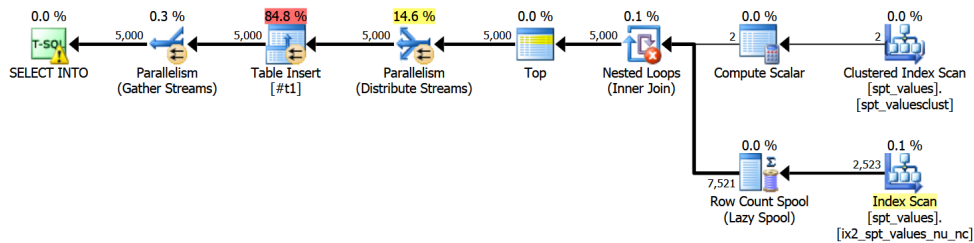
Database	Cost	Query Text	Query Type	Warnings	Executions / Minute
1	CriticalAppDB	NULL	CREATE PROCEDURE dbo.tem...	Procedure: temptable	No Join Predicate, Parallel, Plan... 514.3333
2	CriticalAppDB	9.84455	SELECT TOP(5000) a.name, ...	Statement	No Join Predicate, Parallel, Plan... 514.3333
3	CriticalAppDB	NULL	CREATE PROCEDURE dbo.tabl...	Procedure: tablecol...	Implicit Conversions 117.3333
4	CriticalAppDB	0.435988	SELECT TOP 1 OBJECT_NAM...	Statement	Implicit Conversions 117.3333
5	CriticalAppDB	NULL	CREATE PROCEDURE dbo.use...	Procedure: userlist	Plan Warnings, Implicit Convers... 126.6667
6	CriticalAppDB	0.0120253	SELECT count(*) FROM sys...	Statement	Plan Warnings, Implicit Convers... 126.6667



© SQLIntersection. All rights reserved.
<http://www.SQLintersection.com>

Top query's execution plan

Does it use tempdb?



Waits and top queries

Wait types:

- PAGELATCH_UP
- PAGELATCH_SH

sp_WhoIsActive confirms SGAM and PFS page types

We have only one tempdb data file and heavy use of tempdb

First actions:

1. We add more data files to tempdb
2. We make sure the data files are all equally sized

Wait stat comparison

```
EXEC dbo.sp_AskBrent @ExpertMode=1, @Seconds=10;  
GO
```

```
--First Sample  
--CXPACKET 190.9  
--PAGELATCH_UP 94.6  
--PAGELATCH_SH 16.2
```

	Pattern	Sample Ended	Seconds ...	wait_type	Wait Time (Secon...
1	WAIT STATS	2014-09-23 08:52:20.810	10	CXPACKET	239.5
2	WAIT STATS	2014-09-23 08:52:20.810	10	PAGELATCH_UP	44.1
3	WAIT STATS	2014-09-23 08:52:20.810	10	PAGELATCH_SH	12.7



© SQLIntersection. All rights reserved.
<http://www.SQLIntersection.com>

Top Waits

CXPACKET is even higher now

Remember that top query plan from sp_BlitzCache

- Parallel
- Est cost: 9.48

And why are we running these top queries so often?

- What is running them?
- Should it run at this frequency?

It's not uncommon to find this high execution rate in the real world

It might be your app, or it might be something else!



© SQLIntersection. All rights reserved.
<http://www.SQLIntersection.com>

Observations

CXPACKET went up

PAGELATCH_UP and PAGELATCH_SH went down
(but are still significant)

We start thinking about that top query

- Does it really need to go parallel?

We test it out with a MAXDOP 1 hint and it's just as fast



© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

We raise 'Cost Threshold for Parallelism'

```
EXEC sp_configure 'cost threshold  
for parallelism',50;  
GO  
  
SELECT *  
FROM sys.configurations  
WHERE value <> value_in_use;  
GO  
  
RECONFIGURE;  
GO
```



© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

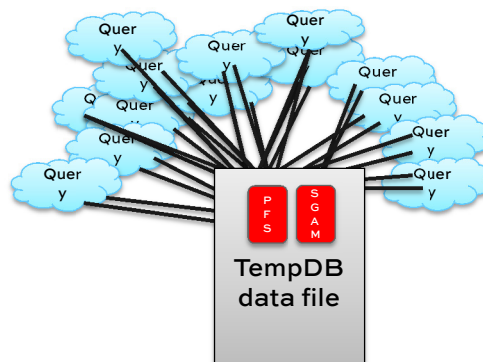
Wait stats after this change

```
EXEC dbo.sp_AskBrent @ExpertMode=1, @Seconds=10;  
GO
```

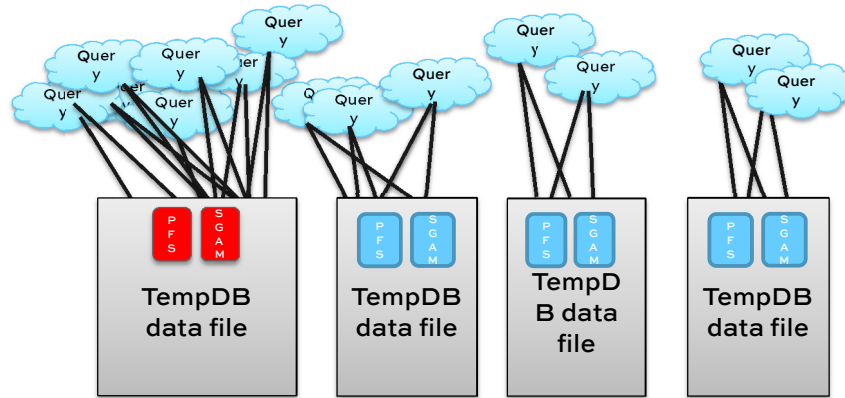
```
--First Sample  
--CXPACKET 190.9  
--PAGELATCH_UP 94.6  
--PAGELATCH_SH 16.2
```

100% - <					
Results Messages					
	Priority	FindingsGroup	Finding		URL
<	0	sp_AskBrent(TM) v10 as of Jan 23 2014 12:00AM - Free Brent Open Unlimited			http://www.BrentO...
	Pattern	Sample Ended	Seco...	wait_type	Wait Time (Seconds)
1	WAIT STATS	2014-09-23 09:01:41.183	10	PAGELATCH_UP	21.0
2	WAIT STATS	2014-09-23 09:01:41.183	10	SOS_SCHEDULER_YIELD	4.1
3	WAIT STATS	2014-09-23 09:01:41.183	10	PAGELATCH_SH	3.8

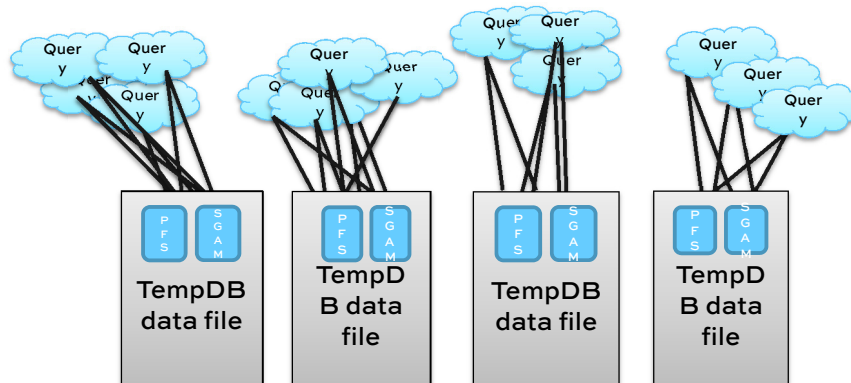
Allocating new objects: On tempdb data file



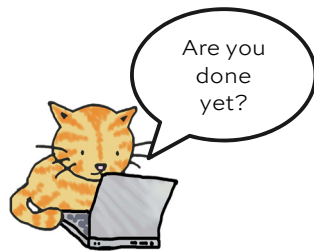
Four uneven tempdb data files



4 even tempdb data files



Parallelism takes work



CXPACKET



Done with
3991 of
3991 rows



Done with
4005 of
4005 rows



Done with
4200 of
5001 rows



Done with 2
of 2 rows

Parallel index scan

Ideally, work is distributed evenly.
That doesn't always happen.

Parallelism wasn't worth it here

Plan Tree										
Operation	Object	Est Cost	Est Subtree Cost	Actual Rows	Est Rows	Thread 0	Thread 1	Thread 2	Thread 3	Thread 4
SELECT INTO		100.0%	100.0%	5,000	5,000					
Parallelism (Gather Streams)		0.3%	100.0%	5,000	5,000	5,000				
Table Insert	[#12]	84.8%	99.7%	5,000	5,000	0	1,250	1,250	1,250	1,250
Parallelism (Distribute Streams)		14.6%	14.8%	5,000	5,000	0	1,250	1,250	1,250	1,250
Top		0.0%	0.3%	5,000	5,000	0	5,000			
Nested Loops (Inner Join)		0.1%	0.3%	5,000	5,000	0	5,000			
Compute Scalar		0.0%	0.0%	2	2					
Clustered Index Scan	[mssqlsystemresource].[sys].[spt_value...]	0.0%	0.0%	2	2	0	2			
Row Count Spool (Lazy Sp...)		0.0%	0.2%	5,000	7,521	0	5,000			
Index Scan	[mssqlsystemresource].[sys].[spt_value...]	0.1%	0.1%	2,523	2,523	0	2,523			

Plan tree viewed in SQL Sentry Plan Explorer.
Actual execution plan.

The terrible tempdb workload

Configuration	Top Wait Stats (30 second sample)	Avg Batch Requests / Min
1 tempdb file (default) Cost threshold 5 (default)	CXPACKET 506.2 PAGELATCH_UP 317.6 PAGELATCH_SH 64.4	642
1 tempdb file (default) Cost threshold 50	PAGELATCH_UP 68.0 SOS_SCHEDULER_YIELD 14.8 PAGELATCH_SH 9.0	1,631 WOW!
4 tempdb files cost threshold 50	PAGELATCH_UP 63.1 SOS_SCHEDULER_YIELD 18.7 PAGELATCH_SH 9.9	1,742
8 tempdb files cost threshold 50	PAGELATCH_UP 50.0 SOS_SCHEDULER_YIELD 21.9 PAGELATCH_SH 7.4	2,713
20 tempdb files cost threshold 50	PAGELATCH_UP 32.2 SOS_SCHEDULER_YIELD 27.8 PAGELATCH_SH 8.2	3,155 WOW!
	SOS_SCHEDULER_YIELD 34.3	



How did you know
it wasn't slow
storage?

- No PAGEIOLATCH_* waits
- Didn't see slow storage read and write times in sp_AskBrent®

How many tempdb files?

sp_Blitz® warns if you just have one, or if they're unevenly sized

You can check your count anytime:

- `exec tempdb..sp_helpfile`

Be proactive and don't just go with one tempdb data file– but don't go crazy.

For general guidelines: BrentOzar.com/go/setup



© SQLIntersection. All rights reserved.
<http://www.SQLintersection.com>

Danger...

Some free tools only show “resource” waits

Activity Monitor, I'm looking at you

I want to know about queries sitting, waiting in the runnable queue, too!

I also like knowing specific wait type names

Resource Waits

```
SELECT COL1  
FROM  
GIANT.TABLE  
WHERE COL2='y'
```



Runnable Queue

```
SELECT *  
FROM  
GIANT.TABLE
```



© SQLIntersection. All rights reserved.
<http://www.SQLintersection.com>

Activity monitor vs sp_AskBrent

Resource Waits				
Wait Category	Wait Time (ms/sec)	Recent Wait Time...	Average Waiter C...	Cumulative Wait Tim...
Buffer Latch	10414	10176	10.4	1267
Lock	382	458	0.4	61
Latch	99	93	0.1	12
Buffer I/O	0	0	0.0	0
Compilation	0	0	0.0	0
Logging	0	0	0.0	0
Memory	0	0	0.0	0
Network I/O	0	0	0.0	0
Other	0	0	0.0	0

Activity Monitor

Data File I/O				
Recent Expensive Queries				

```
SQLQuery2.sql - JUICE01.master (sa (52)) *  
exec sp_AskBrent @ExpertMode=1, @Seconds=10;  
GO
```

sp_AskBrent

Pattern	Sample Ended	Seconds Sample	wait_type	Wait Time (Seconds)	Signal Wait T
1 WAIT STATS	2014-09-23 09:19:26.293	10	CXPACKET	187.5	21.4
2 WAIT STATS	2014-09-23 09:19:26.293	10	PAGELATCH_UP	92.2	10.4
3 WAIT STATS	2014-09-23 09:19:26.293	10	PAGELATCH_SH	17.4	6.8

Analyzing wait statistics

Wait Type Decoder Ring

Avoid mistakes

Don't fixate on percentage wait time

Also look at:

- Total time waiting vs. sample interval
- Average wait time
- Number of waits

Don't neglect looking at the top queries running in this period: the wait stats are tied to those queries

Wait Type	What it means
CXPACKET	Parallel queries are occurring– which isn't necessarily a bad thing
SOS_SCHEDULER_YIELD	Waiting to get on a CPU to crunch some data (in the runnable queue)
PAGEIOLATCH_**	Waiting to pull data from storage into memory
WRITELOG	Flushing a write to the transaction log
ASYNC_NETWORK_IO	Sending data to the client (Network OR client app can be slow)
PAGELATCH_**	Protecting data in memory from inconsistencies (“Don't scribble on my page while I'm using it!”) – could be Tempdb page contention
LCK_**	Waiting to get a lock– and can't. Locks needed depends on isolation level, transaction status, and type of query (select vs modification)
LATCH_**	Contention over a page in memory. Often related to parallelism – but you have to check the subtype (“latch class”) to know for sure.

This isn't just about hardware.
Query and index patterns
cause waits, too.

Example pattern	What you see in sp_BlitzCache	Can drive up waits, including...
Implicit conversion	High avg logical reads High avg CPU	PAGEIOLATCH_** CXPACKET SOS_SCHEDULER_YIELD
Critical missing indexes		
Cursors / while loops	High executions/second	WRITELOG
“After” triggers	Execution plans for the triggers showing high execution/CPU/IO in the cache	LCK_** (which can lead to a big backlog of queries, which can lead to CXPACKET and PAGEIOLATCH_** waits)
App running giant reports	High avg duration High avg logical reads High avg rows	ASYNC_NETWORK_IO

What to do

Your Mission



© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

This requires practice

This is complex

- Lots of red herrings out there
- Easy to get distracted

But it really works

- We use these tools and this methodology in real-life consulting

The more you do this, the better you get at it!

A big part of this is clearing your assumptions about what the problems are, and what can and can't be changed



© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Get free tools

BrentOzar.com/go/Active

BrentOzar.com/Blitz

BrentOzar.com/AskBrent



© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Action Plan

1. **Measure your bottlenecks using wait stats**
 - Normal / low volume times
 - Peak volume times
2. **Analyze your top queries**
 - Look for relationships with your bottlenecks
 - Anti-patterns
 - Frequently run queries
 - Queries using lots of IO
3. **Keep the information on hand. When performance is a problem, take more samples and identify what is different**



© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>



BRENT OZAR
UNLIMITED®

Indexing to Fix Performance Problems

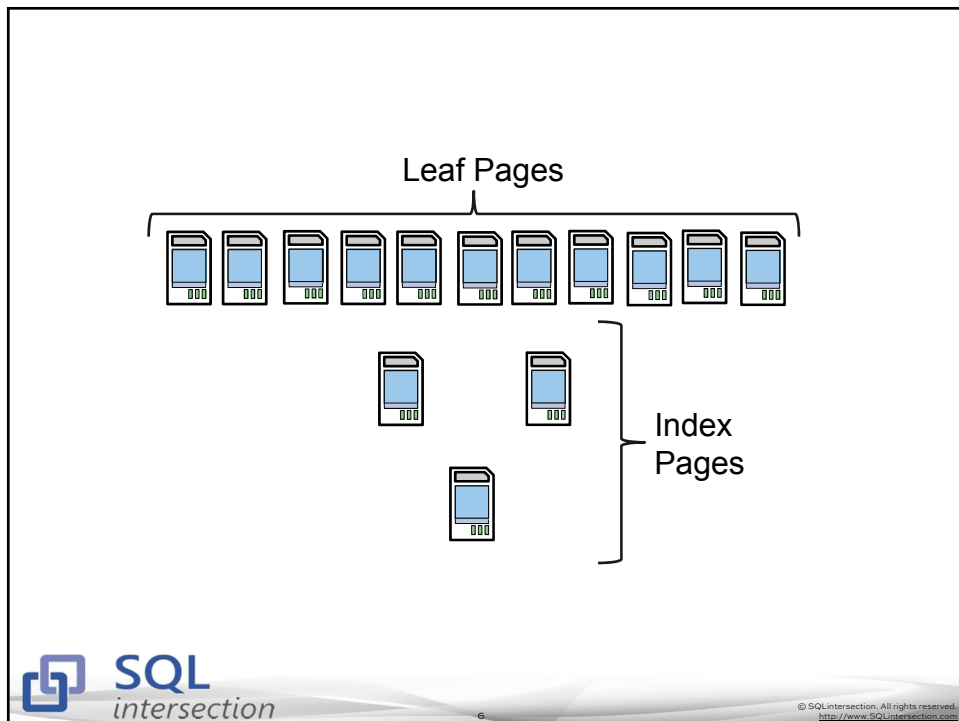
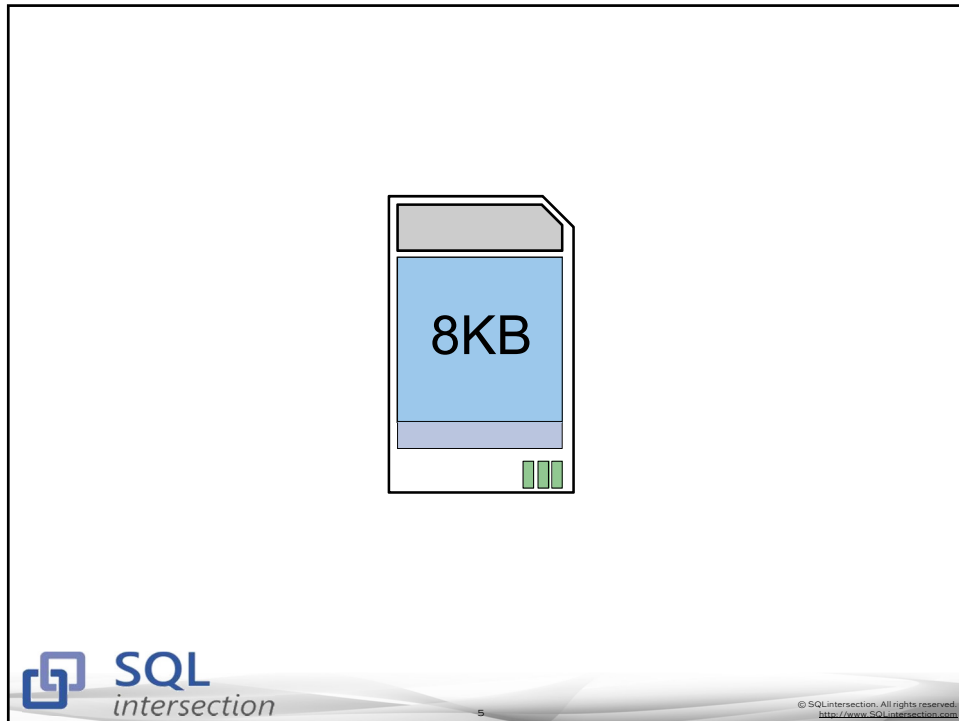
Confession: we haven't been
tuning indexes regularly

We're going to practice diagnosis

You'll learn:

- Free tools to make things easier
- How to use the tools to learn more
- How to plug the tools into a real world process
- Gotchas to look out for

Quick reference



Basic indexes

Clustered

- One per table
- This is the data itself (this IS the table)
 - Is ordered by the defined key
 - Also contains all columns in the table

NonClustered

- Many per table
- “Helper” copies of PART of the table
- Could be a “covering” index or could be used for lookups

Meet the Index Dynamic Management Views

(warning: there's a lot of them)

Basic index information

“What are the index properties?” (Name, ID, Fillfactor, Hypothetical, disabled, filters)

- sys.indexes

“What’s the index definition?” (key and included columns)

- sys.index_columns
- sys.columns

“How big is the index?” (Rows, size, compression)

- sys.partitions
- sys.dm_db_partition_stats



© SQLIntersection. All rights reserved.
<http://www.SQLIntersection.com>

“Usage Stats”

sys.dm_db_index_usage_stats

Shows the number of executions where a plan included an operator

- Does NOT show if the operator was used (or how often it was accessed)

Number and last date of reads

- Seeks
- Scans
- Lookups

Number and last date of last write

- Insert/update/deletes all called “updates”

Data is since startup OR since the last time the index was modified



© SQLIntersection. All rights reserved.
<http://www.SQLIntersection.com>

“Operational Stats”

sys.dm_db_index_operational_stats

Lower level, more transitory

Lock waits

- Page and row

Access counts

- Doesn't distinguish between full scans/range scans, or even range scans and seeks

Data is only persisted while an object's metadata is in memory

No good way when to tell it was last cleared



11

© SQLIntersection. All rights reserved.
<http://www.SQLIntersection.com>

“Missing index” DMVs

sys.dm_db_missing_index_details

sys.dm_db_missing_index_columns

sys.dm_db_missing_index_groups

sys.dm_db_missing_index_group_stats

They tell you:

- How many time an index would have been used
- Estimated improvement the index would give
- Estimated cost of queries that wanted it

This is incredibly useful information...

But it's just **raw information**



12

© SQLIntersection. All rights reserved.
<http://www.SQLIntersection.com>

Advice is complicated

Missing index DMV “requests”

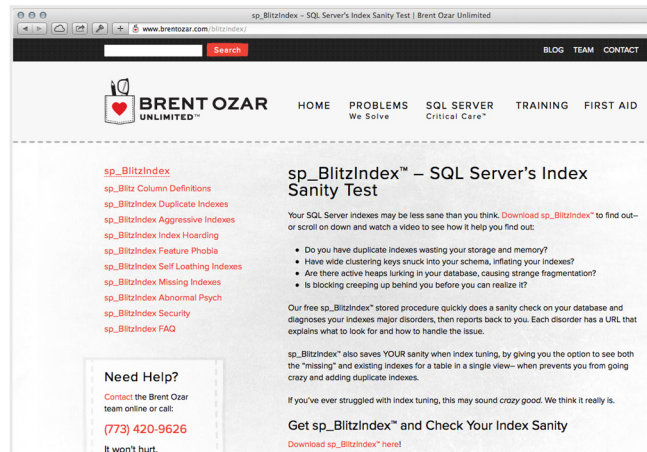
- Recommend super wide indexes
- Don’t de-duplicate requests
- Don’t get thrown for all types of queries
- Get cleared at tricky times

Database Tuning Advisor

- Needs a firm hand
- Shouldn’t be run against production
- Can recommend truly kooky things (including for clustered indexes)

sp_BlitzIndex® helps you see lots
information in one place

BrentOzar.com/BlitzIndex



Get details on a table

Examine individual tables with commands from the 'More Info' column:

```
EXEC dbo.sp_BlitzIndex  
    @DatabaseName='AdventureWorks',  
    @SchemaName='Person',  
    @TableName='Person';
```

This gives you:

- Index size, use, recent locking info
- Missing index requests
- Foreign key info
- Column list (and data types)

Demo 1: Meet the Index DMVs

Index DMVs

Remember:

- Scans aren't always bad
- Index usage stats stays around longer– but is less granular/precise
- Operational stats aren't perfect either. And ...

Look at the big picture

There's many free scripts and tools to look at these:
pick what works for you

Let's find out what's wrong.

How we diagnose

sp_BlitzIndex® (reads those DMVs!)

<http://www.brentozar.com/blitzindex>

Syntax to run in “diagnose” mode

```
EXEC dbo.sp_BlitzIndex  
@DatabaseName='AdventureWorks';
```


DEMO 2: Diagnose!

The top query plan

Query 2: Query cost (relative to the batch): 100%
SELECT TransactionType, SUM(Quantity) AS TotalQuantity, SUM(ActualCost) AS TotalCost, MAX(sh.ShipDate) AS LastShippedDate FROM [Production].[TransactionHistory] AS th
Missing Index (Impact 74.5132): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>] ON [Production].[TransactionHistory] ([TransactionID], [ProductID])



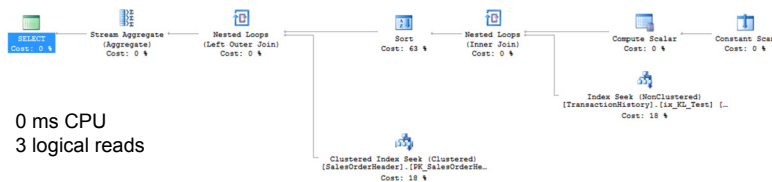
Create index...

```
CREATE INDEX ix_KL_Test
on Production.TransactionHistory
([TransactionDate], [ActualCost])
INCLUDE ([ReferenceOrderID],
        TransactionType], [Quantity])
WITH (ONLINE=ON, SORT_IN_TEMPDB=ON);
GO
```

Top query B&A



0 ms CPU
3 logical reads



**But wait, I made
one big mistake.
What was it?**

Indexing Gotchas

“Missing” missing index requests

Not every query generates a missing index request

- “Trivial” execution plans don’t generate requests
- Some “FULL” optimizations don’t either

If they are run very frequently, an index may be worthwhile

- You won’t see this in missing index recommendations!

To find these:

- Use the execution plan cache
 - Not perfect– memory pressure and RECOMPILE hints impact what you will see
- Run traces for frequently run queries

Missing Index Data Disappears Sometimes

You can't clear this manually without altering indexes on the table

Like usage data, this clears at SQL Server restart

Also cleared for an individual table when you make index changes, such as:

- Creating a NC index
- Dropping a NC index

Always capture usage and missing index info BEFORE each change!

Indexing Gotchas in SQL Server 2012 and onward

Index usage is really important, but make sure you understand when it was last cleared

- In SQL Server 2012+, this is currently cleared by any ALTER INDEX REBUILD commands
- This was not fixed in SP1
- There's no good way to tell when the usage was last cleared from DMVs, you can just see
 - When any metadata on the object was changed (sys.objects)
 - When SQL Server was restarted

2012+ gotchas continued

Readable secondary replicas in Availability Groups make indexes tricky to manage

- Reads are tracked on each instance
- Not aggregated centrally

Example problem: Is that index *really* unused?

Reducing change impact

If you have Enterprise Edition

- Use ONLINE keyword when creating or rebuilding nonclustered indexes to reduce blocking (unless blocking for the duration is OK)
- Dropping indexes still involves blocking locks.

Also plan how you will handle the rollback if needed

- Talk about this as part of the change process

3 Keys

Look out for big wins first

1. Missing indexes > 1 million (“magic number”)
2. Large duplicate indexes (“multiple personality”)
3. Large, active heaps



BRENT OZAR
UNLIMITED®

T-SQL Anti-patterns

Our Agenda

Low Functioning Functions

Nesting Habits of the Wild Query

Low Functioning Functions

Scalar Functions

A function that returns a **single** value

What could go wrong?

```
SELECT  h.SalesOrderID,  
        SUM(d.UnitPrice) AS SumUnitPrice  
FROM    Sales.Customer c  
        INNER JOIN Sales.SalesOrderHeader h  
            ON c.CustomerID = h.CustomerID  
        INNER JOIN Sales.SalesOrderDetail d  
            ON h.SalesOrderID = d.SalesOrderID  
WHERE   dbo.CheckPermissions(c.CustomerID) = 1  
GROUP BY h.SalesOrderID;
```

What could go wrong?

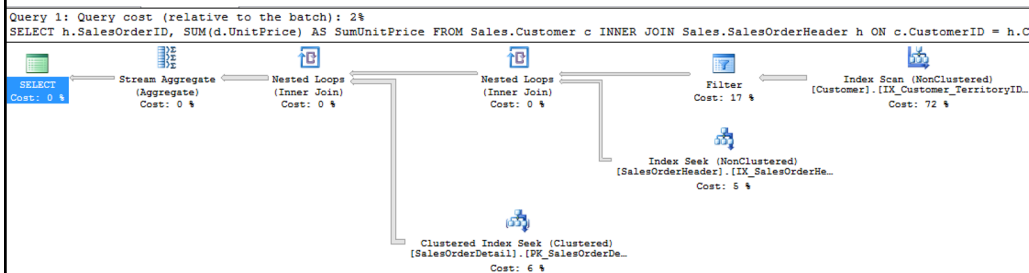
```
SELECT  h.SalesOrderID,  
        SUM(d.UnitPrice) AS SumUnitPrice  
FROM    Sales.Customer c  
        INNER JOIN Sales.SalesOrderHeader h  
        ON c.CustomerID = h.CustomerID  
        INNER JOIN Sales.SalesOrderDetail d  
        ON h.SalesOrderID = d.SalesOrderID  
WHERE   dbo.CheckPermissions(c.CustomerID) = 1  
GROUP BY h.SalesOrderID;
```



7

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

What could go wrong?



8

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

What could go wrong?

Table 'SalesOrderDetail'.
Scan count 861, logical reads 2765

Table 'SalesOrderHeader'.
Scan count 121, logical reads 243

Table 'Customer'.
Scan count 1, logical reads 37



© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

What could go wrong?

Table 'SalesOrderDetail'.
Scan count 861, logical reads 2765

Table 'SalesOrderHeader'.
Scan count 121, logical reads 243

Table 'Customer'.
Scan count 1, logical



Wait, what?

CheckPermissions forces row by row execution.

- Increases access to underlying tables.
- No easy way to tell without in-depth investigation.

This is going to be hard to find

- Average CPU per execution is 75 microseconds.
- 6 reads per execution.

Identifying the Problem: Interactive Means

Service Side Trace

- Requires the problem to be observed.
- This issue won't show up in most traces: short duration, low reads.

Extended Events

- Finer grained control.
- Same problems as a trace.

Identifying the Problem: Historical Means

Plan Cache

- Historical information.
- Lightweight querying.
- We don't have to prepare in advance to capture problems.

I like this approach

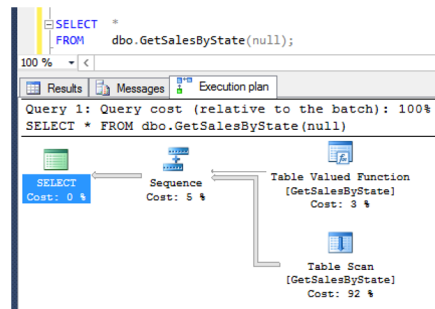
- It's lightweight.
- It lets me find problems I didn't think to watch for.

Demo!

What about TVFs?

A function
that returns
a **tabular** value

A TVF in Action!



A TVF in Action!

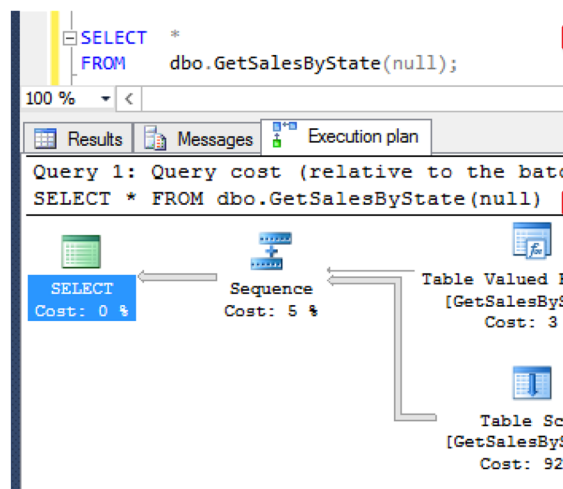


Table Scan	
Scan rows from a table.	
Physical Operation	Table Scan
Logical Operation	Table Scan
Actual Execution Mode	Row
Estimated Execution Mode	Row
Storage	RowStore
Actual Number of Rows	6669
Actual Number of Batches	0
Estimated I/O Cost	0.003125
Estimated Operator Cost	0.003392 (92%)
Estimated CPU Cost	0.000267
Estimated Subtree Cost	0.003392
Number of Executions	1
Estimated Number of Executions	1
Estimated Number of Rows	100
Estimated Row Size	87 B
Actual Rebinds	0
Actual Rewinds	0
Ordered	False
Node ID	3
Object	
[AdventureWorks2012_CS].[dbo].[GetSalesByState]	
Output List	
[AdventureWorks2012_CS].[dbo].[GetSalesByState].MonthFirstDate,	
[AdventureWorks2012_CS].[dbo].[GetSalesByState].StateProvinceName,	
[AdventureWorks2012_CS].[dbo].[GetSalesByState].TotalSales	

A TVF in Action! Now with different parameters

```
SELECT *  
FROM dbo.GetSalesByState('Washington')
```

Query 1: Query cost (relative to the batch): 100%
SELECT * FROM dbo.GetSalesByState('Washington')

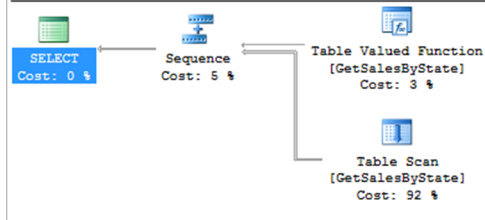


Table Scan	
Scan rows from a table.	
Physical Operation	Table Scan
Logical Operation	Table Scan
Actual Execution Mode	Row
Estimated Execution Mode	Row
Storage	RowStore
Actual Number of Rows	37
Actual Number of Batches	0
Estimated I/O Cost	0.003125
Estimated Operator Cost	0.003392 (92%)
Estimated CPU Cost	0.000267
Estimated Subtree Cost	0.003392
Number of Executions	1
Estimated Number of Executions	1
Estimated Number of Rows	100
Estimated Row Size	87 B
Actual Rebinds	0
Actual Rewinds	0
Ordered	False
Node ID	3
Object	
[AdventureWorks2012_CS].[dbo].[GetSalesByState]	
Output List	
[AdventureWorks2012_CS].[dbo].[GetSalesByState].MonthFirstDate, [AdventureWorks2012_CS].[dbo].[GetSalesByState].StateProvinceName, [AdventureWorks2012_CS].[dbo].[GetSalesByState].TotalSales	

TVFs Give Bad Estimates

Regardless of parameter, we get the same estimate.

If you only have a few rows, don't worry.

If you have a lot of rows:

- Dump rows into temporary tables
- Re-write the TVF to inline

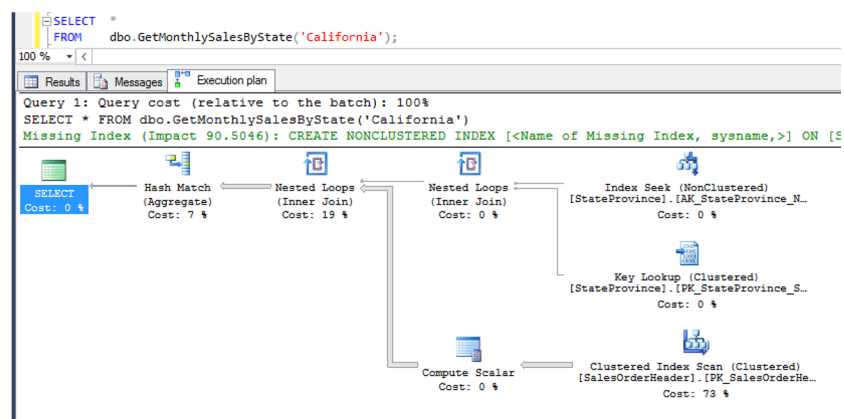
An Inline TVF

A single statement TVF can be inlined by the optimizer.

When inlined:

- The body of the TVF is re-written into the query.
- We get good estimates.
- Things can run faster.
- Everyone wins, yay!

An Inlined TVF



That's better

Nested Loops	
For each row in the top (outer) input, scan the bottom (inner) input, and output matching rows.	
Physical Operation	Nested Loops
Logical Operation	Inner Join
Actual Execution Mode	Row
Estimated Execution Mode	Row
Actual Number of Rows	6224
Actual Number of Batches	0
Estimated Operator Cost	0.1409626 (19%)
Estimated I/O Cost	0
Estimated CPU Cost	0.131524
Estimated Subtree Cost	0.69524
Estimated Number of Executions	1
Number of Executions	1
Estimated Number of Rows	3074.52
Estimated Row Size	77 B
Actual Rebinds	0
Actual Rewinds	0
Node ID	1
Predicate	
[AdventureWorks2012_CS].[Person].[StateProvince].	
[TerritoryID] as [sp].[TerritoryID]=	
[AdventureWorks2012_CS].[Sales].[SalesOrderHeader].	
[TerritoryID] as [soh].[TerritoryID]	
Output List	
[AdventureWorks2012_CS].[Sales].	
[SalesOrderHeader].SubTotal,	
[AdventureWorks2012_CS].[Person].	
[StateProvince].Name, Expr1003	

You can't even
count on the bad
TVF estimates to
be right.

Check this out!

A simple string splitting trick...

```
DECLARE @numbers AS VARCHAR(MAX) =
    '1,2,3,4,5,6';

DECLARE @words AS VARCHAR(MAX) =
    'one,two,three,four,five,six';

DECLARE @spanish AS VARCHAR(MAX) =
    'uno,dos,tres,quatro,cinco,seis';

SELECT  soh.PurchaseOrderNumber ,
        sod.CarrierTrackingNumber ,
        w.element AS EnglishQty ,
        s.element AS SpanishQty
FROM    Sales.SalesOrderHeader AS soh
        JOIN Sales.SalesOrderDetail AS sod ON soh.SalesOrderID =
        sod.SalesOrderID
        JOIN dbo.fn_split(@numbers) AS o ON sod.OrderQty = o.element
        JOIN dbo.fn_split(@words) AS w ON o.pos = w.pos
        JOIN dbo.fn_split(@spanish) AS s ON w.pos = s.pos ;
```



25

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

A Slight Estimation Problem

- It turns out that SQL Server doesn't always make great guesses.
- In this limited case, it doesn't matter. In big queries, this can create terrible execution plans

Table 'Worktable'. Scan count 0, logical reads 0, physical reads 0, read-ahead reads 0, bytes on disk 0
 Table 'Worktable'. Scan count 3, logical reads 346483, physical reads 0, read-ahead reads 0, bytes on disk 0
 Table 'nums'. Scan count 5, logical reads 14, physical reads 0, read-ahead reads 0, bytes on disk 0
 Table 'SalesOrderDetail'. Scan count 1, logical reads 1246, physical reads 0, read-ahead reads 0, bytes on disk 0

Nested Loops	
For each row in the top (outer) input, scan the bottom (inner) input, and output matching rows.	
Physical Operation	Nested Loops
Logical Operation	Inner Join
Actual Execution Mode	Row
Estimated Execution Mode	Row
Actual Number of Rows	6
Actual Number of Batches	0
Estimated Operator Cost	69690951.2 (92%)
Estimated I/O Cost	0
Estimated CPU Cost	31732900
Estimated Subtree Cost	72466000
Estimated Number of Executions	1
Estimated Number of Rows	1518320000000
Actual Rebinds	0
Actual Rewinds	0
Node ID	2

```

(1)=([AdventureWorks2012].[dbo].[nums].[n]-len(replace
(substring([@words],(1),[AdventureWorks2012].[dbo].[nums].[n]),',',CONVERT_IMPLICIT(varchar(max),'0')))+
(1))
Output List
[AdventureWorks2012].[dbo].[nums].n,
[AdventureWorks2012].[dbo].[nums].n,
[AdventureWorks2012].[dbo].[nums].n

```



26

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Moving Toward More Accurate Estimates

- Avoid joining to TVFs wherever possible
- Load data into temp tables
 - Better estimated and actual plans
 - Reduced CPU and I/O

```
CREATE TABLE #words
(
    pos INT ,
    ordinal VARCHAR(10) ,
    english VARCHAR(50) ,
    spanish VARCHAR(50)
);

-- Step one
INSERT INTO #words
( pos ,
  ordinal
)
SELECT pos ,
       element
FROM   dbo.fn_split(@numbers)

-- Step two
UPDATE #words
SET    english = element
FROM   fn_split(@words) AS w
WHERE  w.pos = #words.pos

-- Step three
UPDATE #words
SET    spanish = element
FROM   fn_split(@spanish) AS s
WHERE  s.pos = #words.pos ;
```

Functions in SQL Server

Pros:

- Encapsulate logic
- Re-usable

Cons:

- Provide inaccurate estimates
- Hide I/O requirements
- Obscure the cost of operations

The Catch All

Raise your hand if
you've seen a query
where *every*
parameter was
optional.

COALESCE

```
SELECT *  
FROM dbo.BestViewEver  
WHERE col_a = COALESCE(@a, col_a)  
      AND col_b = COALESCE(@b, col_b)  
      AND col_c = COALESCE(@c, col_c)
```

**But I'll just use an
OR, you say.**

IS NULL and OR

```
SELECT *  
FROM dbo.BestViewEver  
WHERE (@a IS NULL OR col_a = @a)  
      AND (@b IS NULL OR col_b = @b)  
      AND (@c IS NULL or col_c = @c)
```

It's all the same!

SQL Server is bad at short circuiting logical comparison.

- Both comparisons will be executed every time.
- No, you can't trick it.

The worst part:

SQL Server won't use indexes.

SARGability

The example is said to be non-SARG-able.

That's a fancy way of saying it won't use an index.

Basically, by hiding a column value we've messed with optimization.

Other losing patterns

WHERE LEFT(LastName,3) = ...

WHERE UPPER(string) = ...

WHERE LastName LIKE '%smith%'

WHERE YEAR(MyDateField) = '1973'

WHERE VARCHAR = @NVARCHAR

Learn more:

BrentOzar.com/go/sargable

What can we do?

Use dynamic SQL

Build a string with only the parameters we need.

Run that through `sp_executesql`

Yes, this makes development harder, but it makes SQL Server faster.

Re-write queries

Find creative ways to re-write queries.

Move functions to variables and parameters.

Find logical equivalents.

- `YEAR(col) = 1973` becomes
`col >= '1973-01-01 AND col < '1974-01-01'`

Nesting Habits of the Wild Query

Correlated Subqueries

What is a correlated subquery?

A subquery that uses values from an outer query.

Problems?

Extra reads and CPU time

- SQL Server may run the subquery once per row in the outer query.

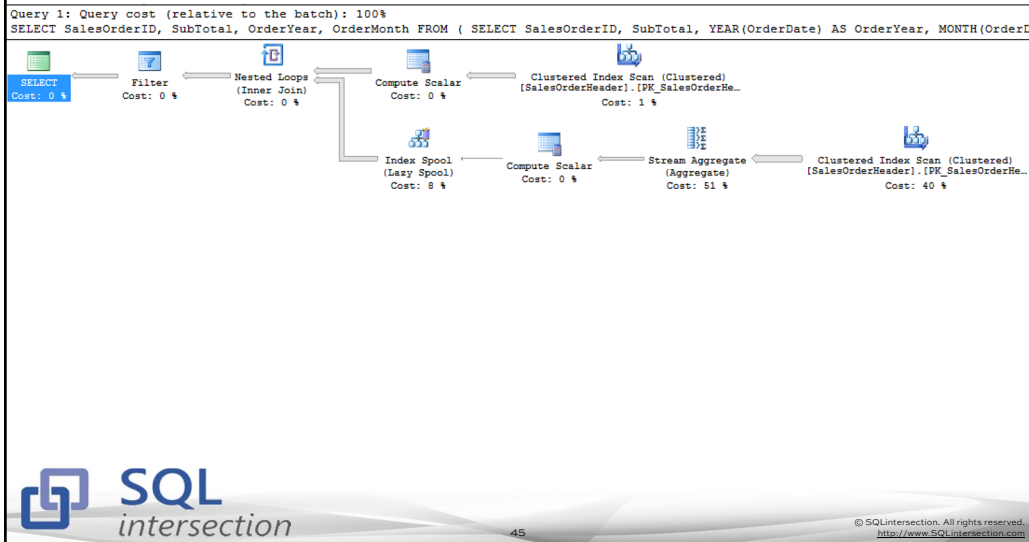
Sudden onset

- Great in development, terrible in production.
- Difficult to identify without careful observation.

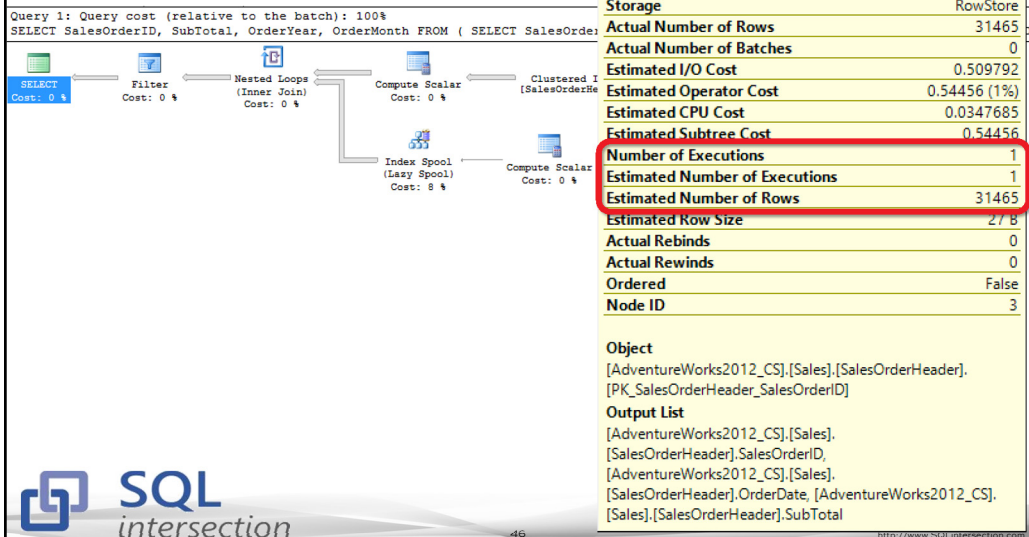
Our awful query

```
SELECT  SalesOrderID, SubTotal,
        OrderYear, OrderMonth
FROM    ( SELECT  SalesOrderID, SubTotal,
                YEAR(OrderDate) AS OrderYear,
                MONTH(OrderDate) AS OrderMonth
        FROM    Sales.SalesOrderHeader
        ) AS x
WHERE   x.SubTotal >
        ( SELECT  AVG(soh.SubTotal)
        FROM    Sales.SalesOrderHeader soh
        WHERE   YEAR(soh.OrderDate) =
                x.OrderYear
                AND MONTH(soh.OrderDate) =
                x.OrderMonth
        )
```

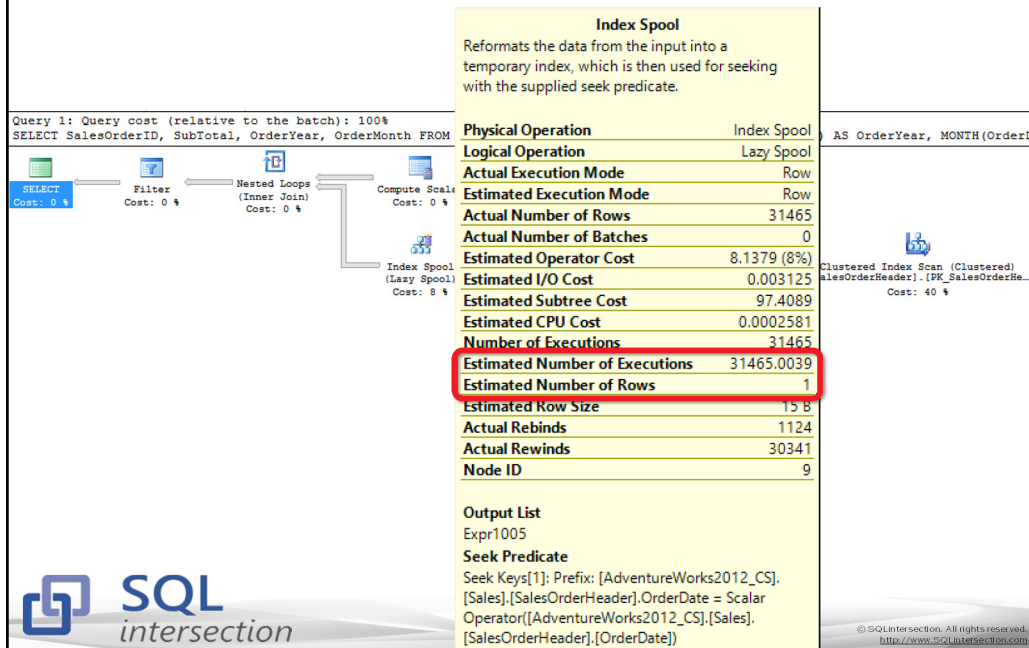
What's going on here?



Scan SalesOrderHeade



What's wrong here?



What's wrong here?

Table 'Worktable'.
Scan count 32589, logical reads 69792

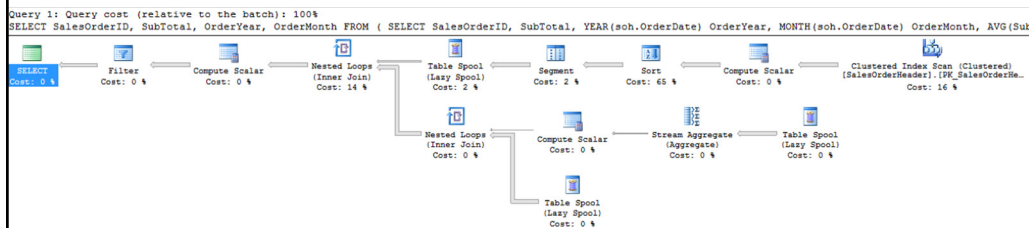
Table 'SalesOrderHeader'.
Scan count 1125, logical reads 775125

We read **6,759,336KB** to perform this query

A re-write!

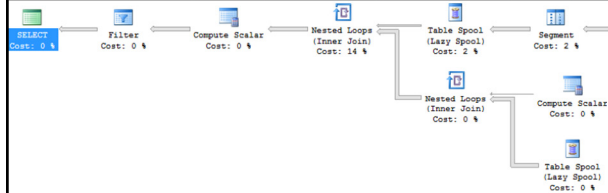
```
SELECT SalesOrderID, SubTotal,
       OrderYear, OrderMonth
FROM   (
        SELECT SalesOrderID,
               SubTotal,
               YEAR(soh.OrderDate) OrderYear,
               MONTH(soh.OrderDate) OrderMonth,
               AVG(SubTotal) OVER
               (PARTITION BY YEAR(soh.OrderDate),
                           MONTH(soh.OrderDate)
                ) AS AverageSale
        FROM   Sales.SalesOrderHeader soh
       ) AS x
WHERE  SubTotal > AverageSale ;
```

What happens now?



What happens now?

Query 1: Query cost (relative to the batch): 100%
SELECT SalesOrderID, SubTotal, OrderYear, OrderMonth FROM (SELECT SalesOrderID, SubTotal, YEAR



Clustered Index Scan (Clustered)	
Scanning a clustered index, entirely or only a range.	
Physical Operation	Clustered Index Scan
Logical Operation	Clustered Index Scan
Actual Execution Mode	Row
Estimated Execution Mode	Row
Storage	RowStore
Actual Number of Rows	31465
Actual Number of Batches	0
Estimated I/O Cost	0.509792
Estimated Operator Cost	0.54456 (16%)
Estimated CPU Cost	0.0347685
Estimated Subtree Cost	0.54456
Number of Executions	1
Estimated Number of Executions	1
Estimated Number of Rows	31465
Estimated Row Size	27 B
Actual Rebinds	0
Actual Rewinds	0
Ordered	False
Node ID	7
Object	
[AdventureWorks2012_CS].[Sales].[SalesOrderHeader].	
[PK_SalesOrderHeader_SalesOrderID] [soh]	
Output List	
[AdventureWorks2012_CS].[Sales].	
[SalesOrderHeader].SalesOrderID,	
[AdventureWorks2012_CS].[Sales].	
[SalesOrderHeader].OrderDate, [AdventureWorks2012_CS].	
[Sales].[SalesOrderHeader].SubTotal	

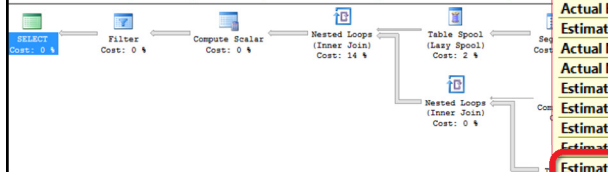


51

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Everything runs once!

Query 1: Query cost (relative to the batch): 100%
SELECT SalesOrderID, SubTotal, OrderYear, OrderMonth FROM (SELECT SalesOrderID, SubTo



Segment	
Segment.	
Physical Operation	Segment
Logical Operation	Segment
Actual Execution Mode	Row
Estimated Execution Mode	Row
Actual Number of Rows	31465
Actual Number of Batches	0
Estimated I/O Cost	0
Estimated Operator Cost	0.05939 (2%)
Estimated CPU Cost	0.0593869
Estimated Subtree Cost	2.77627
Estimated Number of Executions	1
Number of Executions	1
Estimated Number of Rows	31465
Estimated Row Size	35 B
Actual Rebinds	0
Actual Rewinds	0
Segment Column	Segment1006
Node ID	4
Output List	
[AdventureWorks2012_CS].[Sales].	
[SalesOrderHeader].SalesOrderID,	
[AdventureWorks2012_CS].[Sales].	
[SalesOrderHeader].OrderDate,	
[AdventureWorks2012_CS].[Sales].	
[SalesOrderHeader].SubTotal, Expr1001, Expr1002,	
Segment1006	
Group By	
Expr1001, Expr1002	



52

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

What's going on here?

Table 'Worktable'.

Scan count 3, logical reads 64141

Table 'SalesOrderHeader'.

Scan count 1, logical reads 689

We've reduced the amount of I/O to **518,640KB**

Simple Rewrites Save Time

By re-writing the query slightly differently:

- SQL Server produces a better plan.
- We read the table only once.
- The code is, arguably, more maintainable.

Most importantly:

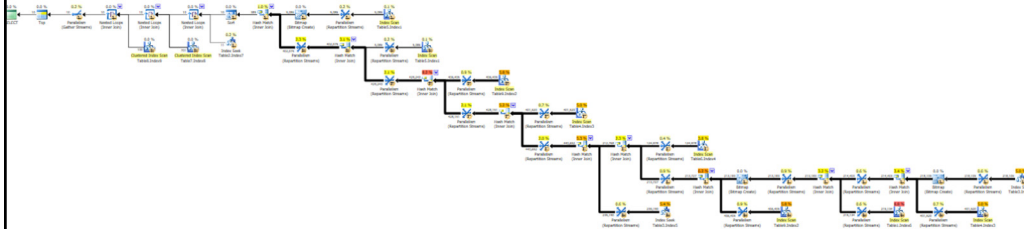
- We eliminate multiple executions of one code fragment.

Nested Views

It's so simple!

```
SELECT TOP 10 *  
FROM [dbo].[ILookReallySimple]  
WHERE Column10 = @Column10Value  
      OR Column11 = @Column11Value  
ORDER BY ADateColumn DESC
```

... What?



The problem with nested views

It's not directly a problem with nested views.

It's a problem of query complexity.

- 2 tables – 2 ways to join
- 3 tables – 9 ways to join
- 4 tables – 64 ways to join
- ...
- 10 tables – 1,000,000,000 ways to join

This is just the total number of possible join combinations and doesn't include optimizer smarts

At some point the optimizer will give up and return the best plan it has **right now**.

Finding Exhausted Optimizers

Optimization Level	FULL
Parameter List	Column14, Column13
Physical Operation	
QueryHash	0xF71AAAB5149FC515
QueryPlanHash	0x4D9C241222B6733D
Reason For Early Termination C	Time Out

Getting Rid of Nested Views

Break out the query.

Re-write using temp tables.

- Insert into temp table from view.
- Join between temp tables.
- May appear more complex, almost always faster.

Only use the tables you need.

- Track down the columns you need.
- Re-write your query to avoid views.
- May not always be possible, depending on permissions.

Why no working example?

We didn't produce an example in AdventureWorks that did the wrong thing.

In fact, in trying to "do the wrong thing" we made our queries faster.

Want to prove it out for yourself?

- Open up SSMS and get cracking!
- Look in your own code base.
- This is really time consuming if you're starting from scratch.

What Can We Do?

Patterns and Anti-Patterns

Understand the data access patterns of your code.

Write code for convenience.

Tune for performance.



BRENT OZAR
UNLIMITED®

Building a Health Check Report

When should you perform a health check?

As often as you want regular indicators of server health.

At least once a quarter.

- Regular indicator of system health.
- Spot hardware and software flaws before they stop the business cold.

Why run a health check?

You go to the doctor regularly:

- To make sure nothing is wrong.
- To get advice on your overall health.

Run a SQL Server health check for the same reasons!

It's an overview of system health.

It's your insurance policy

- Things can break at any minute.
- If you're running health checks and you know things are okay, a sudden failure really is sudden.



3

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Why check your ISV applications?

Point the finger of blame in the right direction.

Break problems into two sections:

- Infrastructure problems
- Software problems

Promotes better communication with the vendor.

- Focus on making the product better.
- Do it before it's a problem.
- Stop complaining, start helping.



4

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Think about your
last visit to the
doctor.

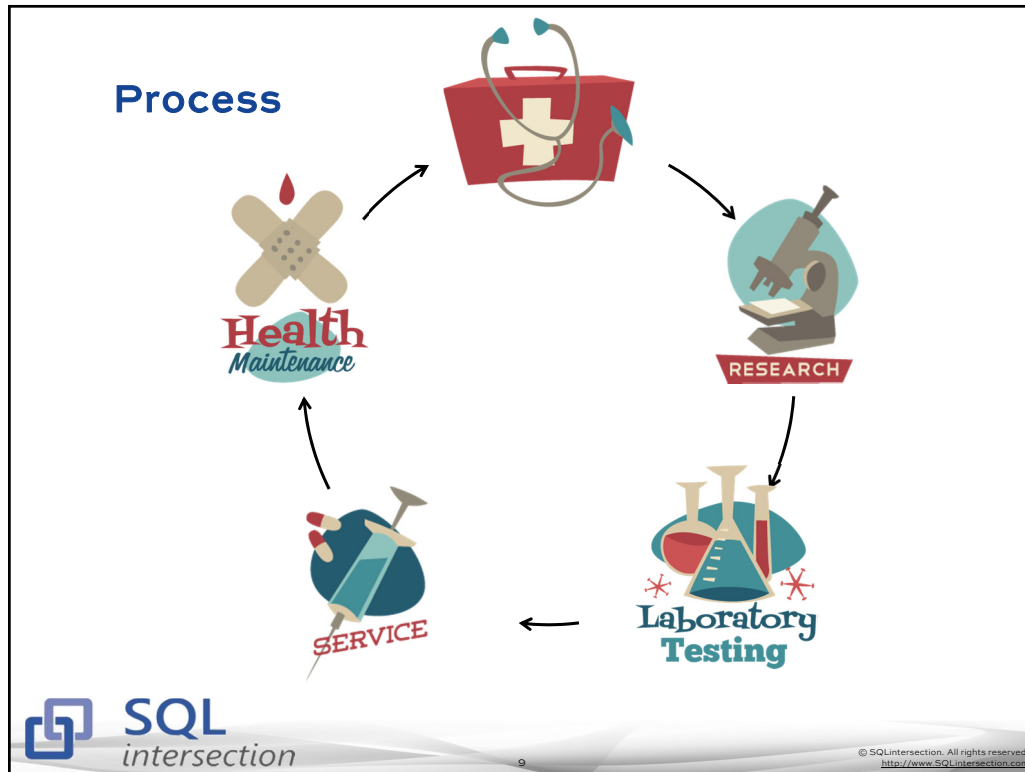
What kind of
process did they
follow?

A Visit to the Doctor

Does your doctor:

- A. Randomly poke you, look at various limbs/organs, ignore your symptoms, consult WebMD, and declare that you have bubonic plague and an obscure incurable cancer?
- B. Collect symptoms and data, perform research, recommend a course of treatment?

**SQL Server
is no different.**



What should your
health check
include?

SQL intersection

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

A Good Health Check Includes

1. List of symptoms
2. Evidence and vital signs
3. Details about recommended treatment
4. Prioritized action plan
5. A repeatable process

Symptoms



Collecting Symptoms

The goal of a health check is to fix users' problems.

Interview people to find out what's bothering them.

- Users
- Operations staff
- Business stakeholders

Get to the root of the problem quickly.



13

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

List of symptoms

Which application(s) are you using?

Which server(s) are involved?

Users – what are your current pain points?

Ops team – what are your current pain points?

Stakeholders – what are your current pain points?

If you want to get fancy, ask how this is preventing their success.



14

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Symptoms: A Word of Warning

Symptoms are what the users perceive to be wrong.

- Don't confuse symptoms for the problem.
- A numb left arm doesn't always mean heart attack.

Symptoms help us prioritize recommendations.

Symptoms: A Word of Warning

Symptoms are what the users perceive to be wrong.

- Don't confuse symptoms for the problem.
- A numb left arm doesn't always mean heart attack.

Symptoms help us prioritize recommendations.

- ETL performance is incredibly slow.
- No users are affected.
- Should we recommend tuning ETL processes as a high priority?

Symptoms guide findings **delivery**, not discovery.

Symptoms & Delivery

Match your findings with users' concerns.

Symptoms of a problem are just that: symptoms.

- A symptom is subjective.
- Usually cannot be measured directly.

Correlate your findings with the symptoms.

- “You saw a large number of time outs in the application. We can correlate that with these long running queries and associated wait stats.”

Allocate Responsibility

With ISV applications, two groups are responsible:

- The vendor.
- You.

Allocate responsibility appropriately.

- Poor hardware cannot perform well.
- Fast hardware, however, can hide ugly code.

Gathering Vital Signs



Typical Vital Signs

1. Configuration

- Windows settings
- SQL Server settings
- Large, system-wide improvements can happen here.
- Verify configuration settings with tools like sp_Blitz®

Typical Vital Signs

1. Configuration

2. Index information

- Missing indexes can hurt performance.
- Extra indexes can also hurt performance.
- Collect index usage data

Typical Vital Signs

1. Configuration
2. Index information
3. Poorly performing queries
 - Identify the sources of performance problems.
 - Examine plan cache metrics first – it's passive and low impact to production.
 - A slow report may be caused by frequent OLTP queries.



23

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Typical Vital Signs

1. Configuration
2. Index information
3. Poorly performing queries
4. Top wait statistics
 - Waits provide insight into why queries are slow.
 - Identify top waits over time and by average wait length.
 - Some waits may only occur during activity – sample regularly.



24

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Typical Vital Signs

1. Configuration
2. Index information
3. Poorly performing queries
4. Top wait statistics
5. Storage metrics
 - Find signs of storage contention.
 - Compare throughput to latency.

Typical Vital Signs

1. Configuration
2. Index information
3. Poorly performing queries
4. Top wait statistics
5. Storage metrics
6. Memory utilization
 - Is there enough memory to cache data?
 - Is memory being used by the plan cache?

Typical Vital Signs

1. Configuration*
2. Index information*
3. Poorly performing queries*
4. Top wait statistics
5. Storage metrics
6. Memory utilization
7. All kinds of other stuff

A * means we'll cover it today



27

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Our Tools

Configuration – sp_Blitz®

<http://brentozar.com/blitz/>

Indexing – sp_BlitzIndex®

<http://brentozar.com/blitzindex/>

Poorly performing queries

<http://brentozar.com/go/plancache>



28

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Let's grab vital signs!

Analyzing the Data



What story are we going to tell?

Understand success

If it's not related to pain, don't spend time on it.

- You don't have to document that 1AM ETL is slow if people are worried about mid-day inventory queries.
- You do have to document supporting evidence.

If it is related to success, make sure you include it.

- No backups, no DBCC checks – let someone know.

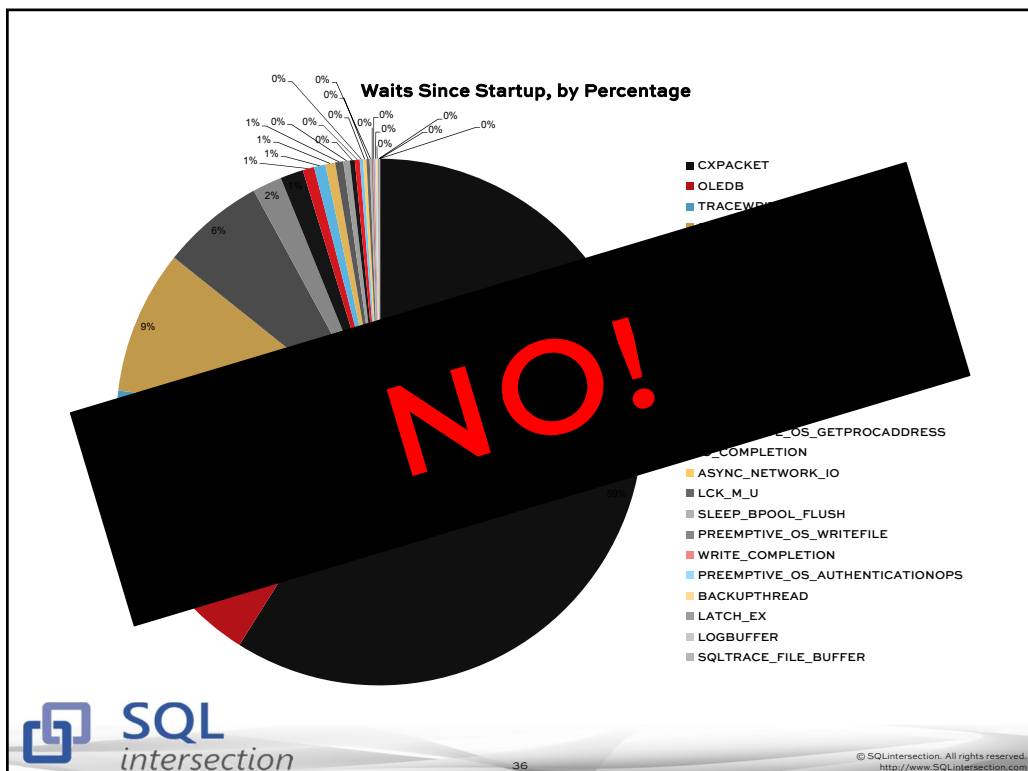
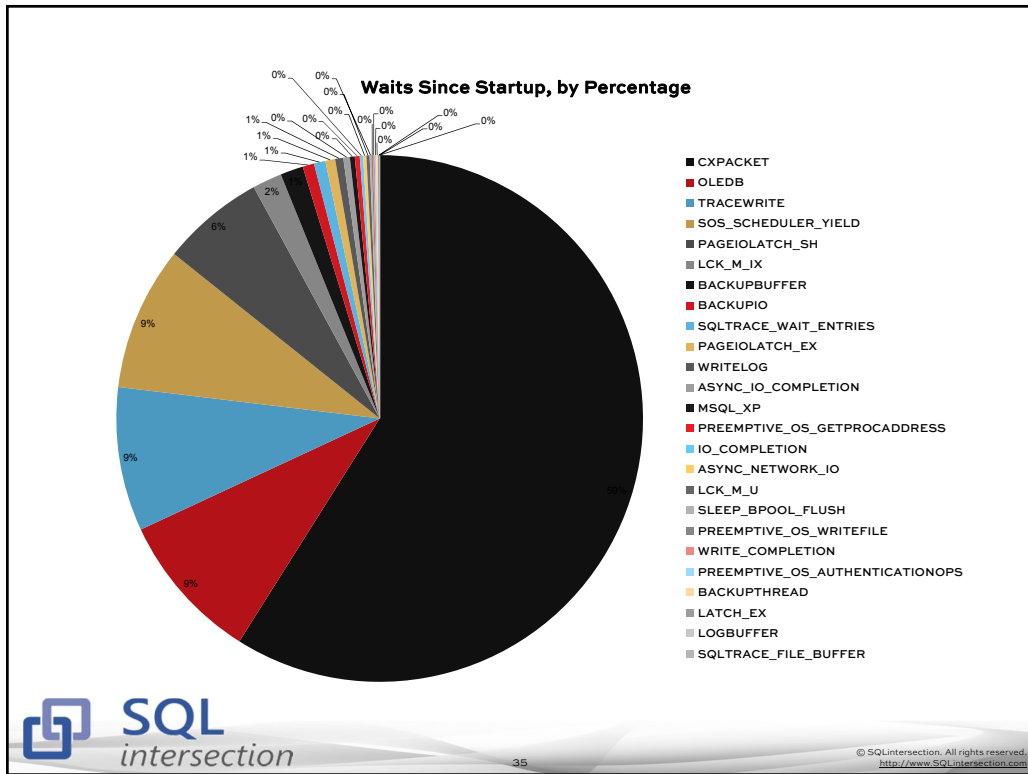
Correlate findings for long term success.

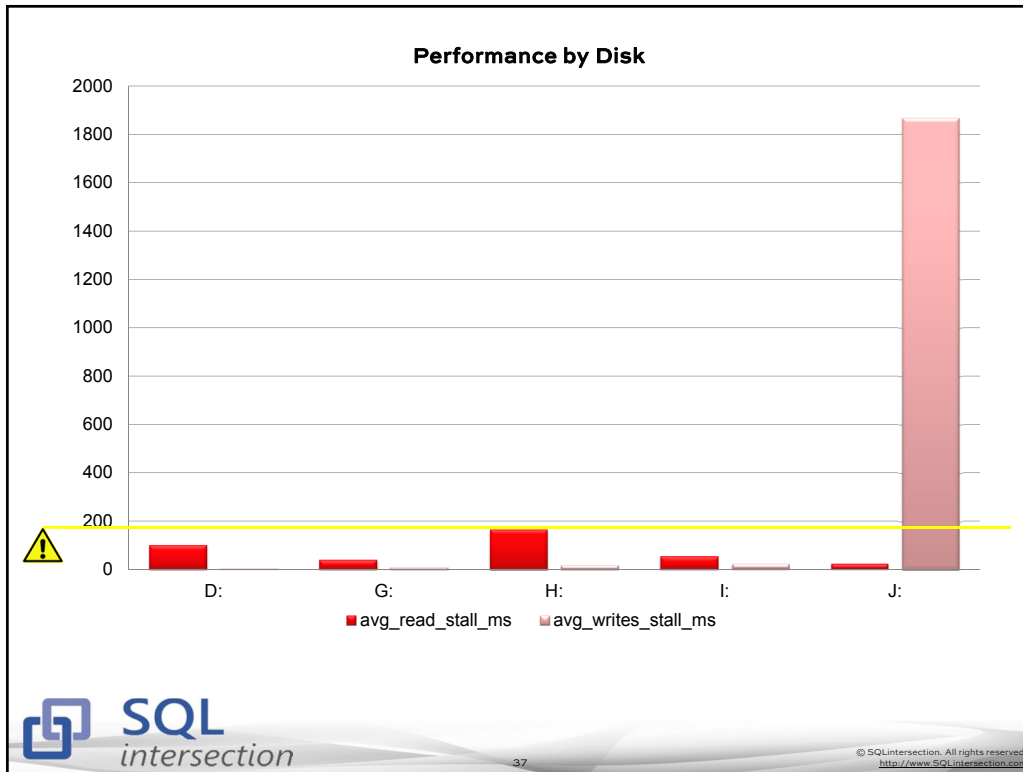
What should we show key stakeholders?

Focusing the Message

Narrow down your message to a few salient points.

Deliver the message based on evidence.





What do we learn?

Convey a clear message to business users & the ISV.

Use simple, understandable images to prove your points.

Isolate problems into clear responsibility areas:

- “Boss: We need more/better storage so reports will be faster and they don’t block business users.”
- “Vendor: We’d appreciate you supporting RCSI, readable secondaries, or a separate reporting schema.”

Solving the Problem



Detail Solutions

Create a list of possible solutions

- Match the solution to a symptom and pain.
- Detail a list of risks with the solution.
- Mitigate each risk.

People like options

- Even when the options are “get better or stay sick”.
- Provide a local option for every vendor fix.
Sometimes we can fix the vendor’s code problem with faster cores or more RAM.

“Stop doing that!”

Don’t just point, provide a road map.

Proposing solutions builds trust.

Describe anti-patterns that can get in the way.

- Is the ISV writing cursors?
- Is there no index maintenance?
- Is there too much index maintenance?

Make a map to **success**.
Point out failure pitfalls.

Create an action plan

Provide a solution that can be accomplished in stages.

- Some things will be clean up.
- Others will be small projects.
- There may even be large projects.

A prioritized plan makes implementation easier.

- Short, medium, long term.
- Time frame for each stage.

What's the vendor's plan?

Detail the problems you found.

- Be specific.
- Describe problem queries and provide steps to reproduce.

Be aware that your priority may not reflect the vendor's:

- Big fixes won't be fast.
- What seems trivial may be hard.

Do push for a roadmap, but know that specific dates may be hard to get.

No surprises

Everything in the action plan must be documented.

- Provide supporting evidence.
- If it doesn't relate to the pain points, throw it out.
- If it doesn't relate to success, throw it out.

Solutions should be reasonable.

- The solution shouldn't be "buy new servers and SAN" or "write a new app".
- Focus on measurable goals.

Meticulous Documentation?

Someone else already wrote an overview.

Provide a recommended reading/viewing list.

- Don't reinvent the wheel.
- Don't re-explain virtual log files.
- **Do** provide helpful, relevant links.

Highlight problems and point people to another explanation when you can't summarize quickly.

Staying in Shape



Success Criteria

How do you know you're getting better?

Define success criteria

Recommendations should include success metrics.

Examples:

- Reduction in PAGEIOLATCH_*
- I/O latency should be reduced
- Users will stop complaining

Problem Detection

Perform regular server reviews.

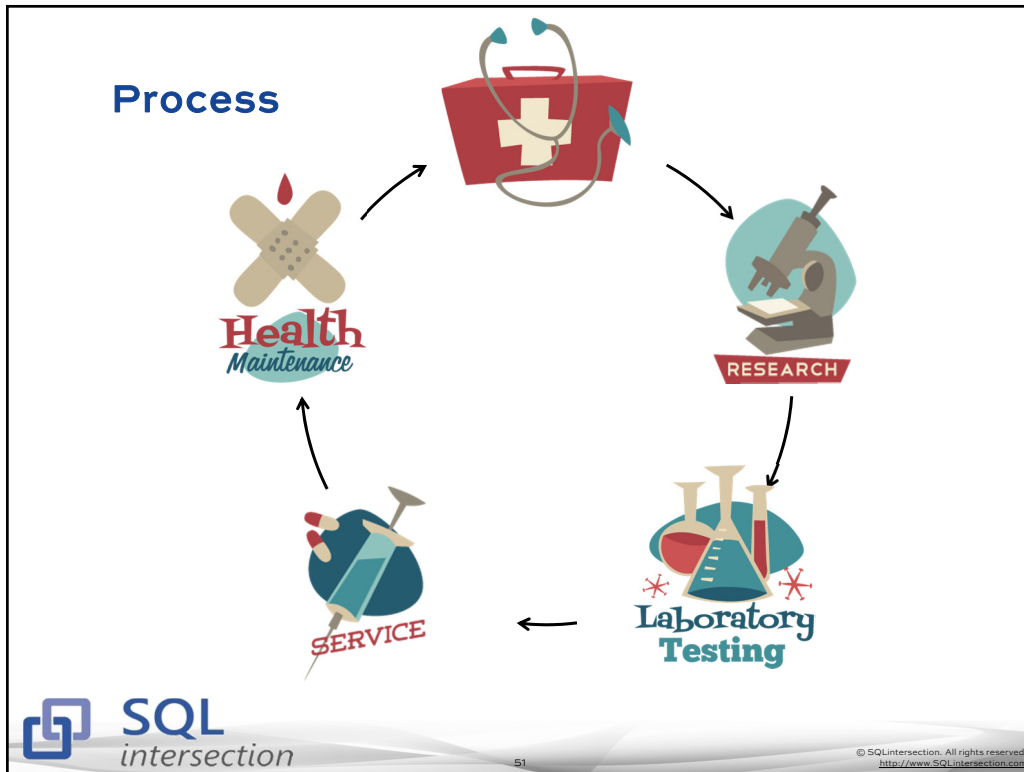
Use high level scripts to find major problems.

- sp_Blitz®
- sp_BlitzIndex®

Only dig deeper as needed.

- Look for problem indicators.
- When you find them, perform a deeper analysis.

SQL Server health is a cycle.



Ensure Database Health

1. Perform regular health checks
2. List of symptoms
3. Capture evidence and vital signs
4. Document recommended treatment
5. Create a prioritized action plan

SQL intersection

52

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Your vendor is a partner

Create a separate set of changes for them.

Share your own fixes.

- Your changes may help them improve other deployments.
- Plus it shows you're working with them.

Where possible, provide a local mitigation to ISV issues.



53

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Free Tools

<http://brentozar.com/blitz/>

<http://brentozar.com/blitzindex/>

<http://brentozar.com/go/plancache/>



54

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>



BRENT OZAR
UNLIMITED®

ExampleCo SQL Critical Care®

What Does Success Look Like To You?

Improved concurrency.

Ability to handle more users and spikes in load.

Long term stability.

Your Environment

EC2 instance

- 4 vCPUs
- 34GB memory

Modern software

- Windows Server 2012R2
- SQL Server 2014



57

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

General Health Issues

Some high level health items have not been set up:

- DBCC checks
- Corruption alerts
- High severity alerts

Mitigation:

- Use sp_Blitz® output or our setup guide to correct these configuration issues.
- See <http://brentozar.com/blitz> or <http://brentozar.com/go/setup> for more.



58

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

You aren't running backups!

You've never backed up any of these databases.

Run, don't walk, to <http://ola.hallengrent.com> and set up automated backups right now.

What are we waiting on?

Massive Parallelism

Many queries are using up a lot of CPU.

This is inhibiting concurrent querying.

The solution is to tune parallelism through:

- Effective indexes.
- Query tuning.
- Controlling parallelism with server level settings.



61

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

More CPU Contention

There are a lot of SOS_SCHEDULER_YIELD waits.

In effect, queries are fighting to get on CPUs.

Mitigations:

- Query re-writes to avoid common anti-patterns.
- Maybe some indexing – faster queries will use fewer resources.



62

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Signs that disks could be a problem

They're not the top problem now.

But read and write stalls are higher than expected.

Mitigations:

- Use the right indexes.
- Tune queries to avoid implicit conversions.
- Consider switching to SSDs
(this is the new default in AWS).

Wait Stats

Wait Type	% Total Wait Time	Number Waiting Tasks	Avg Wait Time (ms)
CXPACKET	79.6	2,003,890.0	4.10
SOS_SCHEDULER_YIELD	12.3	799,842.0	1.60
PAGEIOLATCH_SH	5.0	20,580.0	25.30
IO_COMPLETION	1.2	21,812.0	5.60

Query Problems

sp_BlitzCache™ found some problems

Implicit conversions

- SQL Server is converting values on the fly.
- This can lead to poor index use.

Parameter sniffing

- Big variance in query performance, this can be caused by data skew.

Missing Indexes

- Top slow queries have missing index requests.

Queries to tune

TopPeopleForTag

- 3 missing indexes
- Parameter sniffing
- Implicit conversions

CommentsByKeyword

- Parameter sniffing

Action Plan: Us

Short Term: 1 week

Start running DBCC checks.

Start running backups.

Implement high value missing indexes.

Fix basic setup

- <http://brentozar.com/go/setup>



69

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Medium Term: 1 month

After tuning indexes, examine wait stats.

- Tune for parallelism. If CXPACKET persists:
 - Increase cost threshold for parallelism.
 - Set MAXDOP appropriately (start with 2).



70

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Long Term: 1 Quarter

Evaluate disk latency:

- Consider optimizations provided by Amazon.
- Provisioned IOPS.
- EBS SSD volumes.

Action Items: VendCo

Vendor Action Items

Tune worst queries:

- TopPeopleForTag
- CommentsByKeyword

Evaluate a separate reporting system.

- Consider a data warehouse system.
- Consider supporting scale out reads with AlwaysOn AGs

Consider adding the missing indexes outlined in the short term section.



BRENT OZAR
UNLIMITED®

How to Recommend Fast, Reliable Server Setups

Sounds so easy, right? “Buy the big ones.” But no...

The big picture

1. What SQL Server edition should you recommend?
2. What hardware does the business’s RPO/RTO goal require?
3. Why shared storage is usually slower
4. Your solid state storage choices
5. The no-brainer answer for Standard Edition
6. Enterprise Edition’s socket choices

What SQL Server edition do you need?



© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

3

Picking your SQL Server edition

	Standard	Enterprise
Cost per core	\$2k	\$7k
Cost for 2-socket, hex-core	\$24k	\$84k
Cost for 4-socket, hex-core	--	\$168k
Max logical processors (cores)	16	OS max
Max memory used by SQL Server	64GB	OS max
Backup compression	Yes	Yes
Data compression	No	Yes
Transparent data encryption	No	Yes
Partitioned tables	No	Yes
Faster storage access for big data	No	Yes

All prices USD.



© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

4

Factor in the hardware costs

	SQL 2012 Standard	SQL 2012 Enterprise crippled	SQL 2012 Enterprise
Cores	12 (2x6)	12 (2x6)	12 (2x6)
Memory	64GB	64GB	768GB
Cool Stuff	No	Yes	Yes
Hardware w/TempDB SSDs	\$7k	\$7k	\$30k
Licensing	\$24k	\$84k	\$84k
Total Cost	\$31k	\$91k	\$114k

Signs you need Enterprise Edition

Your app requires table partitioning*

You have to offload reads to a second server*

Over 500GB of live, queried data (not logs/archives)

* - Don't be so sure.

Books Online “Reading Pages”

The storage engine uses prefetching to speed base table lookups from nonclustered indexes. The leaf rows of a nonclustered index contain pointers to the data rows that contain each specific key value. As the storage engine reads through the leaf pages of the nonclustered index, it also starts scheduling asynchronous reads for the data rows whose pointers have already been retrieved. This allows the storage engine to retrieve data rows from the underlying table before it has completed the scan of the nonclustered index. Prefetching is used regardless of whether the table has a clustered index. SQL Server Enterprise uses more prefetching than other editions of SQL Server, allowing more pages to be read ahead. The level of prefetching is not configurable in any edition. For more information about nonclustered indexes, see [Nonclustered Index Structures](#).

Advanced Scanning

In SQL Server Enterprise, the advanced scan feature allows multiple tasks to share full table scans. If the execution plan of a Transact-SQL statement requires a scan of the data pages in a table and the Database Engine detects that the table is already being scanned for another execution plan, the Database Engine joins the second scan to the first, at the current location of the second scan. The Database Engine reads each page one time and passes the rows from each page to both execution plans. This continues until the end of the table is reached.



7

© SQLIntersection. All rights reserved.
<http://www.SQLIntersection.com>

Merry-Go-Round Scans in Enterprise

tblSales – 1,000GB

My Query



8

© SQLIntersection. All rights reserved.
<http://www.SQLIntersection.com>

Merry-Go-Round Scans in Enterprise

tblSales – 1,000GB

My Query



© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

9

Merry-Go-Round Scans in Enterprise

tblSales – 1,000GB

My Query

Your Query



© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

10

Merry-Go-Round Scans in Enterprise

tblSales – 1,000GB

My Query

Your Query

Merry-Go-Round Scans in Enterprise

tblSales – 1,000GB

My Query

Your Query

Merry-Go-Round Scans in Enterprise

tblSales – 1,000GB

Continued

Your Query

What hardware does
the business's
RPO/RTO goal
require?

BrentOzar.com/go/fail

High Availability	Disaster Recovery	"Oops" Deletions																																													
																																															
For isolated server failures: Windows crash, RAID controller failure, SQL or Windows patch fails, C drive full, bad memory chip, wrong box unplugged	For widespread outages: Data center power or network outage, domain controller failure, SAN failure, fire, quake, zombies in the data center	For human T-SQL error: DELETE without a where clause, bug in stored procedure for updates, end user needing a restore due to human error																																													
RPO: Max acceptable data loss: <table border="1"> <thead> <tr> <th>Biz Goal</th> <th>IT Goal</th> <th>Current</th> </tr> </thead> <tbody> <tr><td></td><td></td><td>Zero</td></tr> <tr><td></td><td></td><td>1 minute</td></tr> <tr><td></td><td></td><td>1 hour</td></tr> <tr><td></td><td></td><td>1 day</td></tr> </tbody> </table>	Biz Goal	IT Goal	Current			Zero			1 minute			1 hour			1 day	RPO: Max acceptable data loss: <table border="1"> <thead> <tr> <th>Biz Goal</th> <th>IT Goal</th> <th>Current</th> </tr> </thead> <tbody> <tr><td></td><td></td><td>Zero</td></tr> <tr><td></td><td></td><td>1 minute</td></tr> <tr><td></td><td></td><td>1 hour</td></tr> <tr><td></td><td></td><td>1 day</td></tr> </tbody> </table>	Biz Goal	IT Goal	Current			Zero			1 minute			1 hour			1 day	RPO: Max acceptable data loss: <table border="1"> <thead> <tr> <th>Biz Goal</th> <th>IT Goal</th> <th>Current</th> </tr> </thead> <tbody> <tr><td></td><td></td><td>Zero</td></tr> <tr><td></td><td></td><td>1 minute</td></tr> <tr><td></td><td></td><td>1 hour</td></tr> <tr><td></td><td></td><td>1 day</td></tr> </tbody> </table>	Biz Goal	IT Goal	Current			Zero			1 minute			1 hour			1 day
Biz Goal	IT Goal	Current																																													
		Zero																																													
		1 minute																																													
		1 hour																																													
		1 day																																													
Biz Goal	IT Goal	Current																																													
		Zero																																													
		1 minute																																													
		1 hour																																													
		1 day																																													
Biz Goal	IT Goal	Current																																													
		Zero																																													
		1 minute																																													
		1 hour																																													
		1 day																																													
RTO: Max acceptable downtime: <table border="1"> <thead> <tr> <th>Biz Goal</th> <th>IT Goal</th> <th>Current</th> </tr> </thead> <tbody> <tr><td></td><td></td><td>Zero</td></tr> <tr><td></td><td></td><td>1 minute</td></tr> <tr><td></td><td></td><td>1 hour</td></tr> <tr><td></td><td></td><td>1 day</td></tr> </tbody> </table>	Biz Goal	IT Goal	Current			Zero			1 minute			1 hour			1 day	RTO: Max acceptable downtime: <table border="1"> <thead> <tr> <th>Biz Goal</th> <th>IT Goal</th> <th>Current</th> </tr> </thead> <tbody> <tr><td></td><td></td><td>Zero</td></tr> <tr><td></td><td></td><td>1 minute</td></tr> <tr><td></td><td></td><td>1 hour</td></tr> <tr><td></td><td></td><td>1 day</td></tr> </tbody> </table>	Biz Goal	IT Goal	Current			Zero			1 minute			1 hour			1 day	RTO: Max acceptable downtime: <table border="1"> <thead> <tr> <th>Biz Goal</th> <th>IT Goal</th> <th>Current</th> </tr> </thead> <tbody> <tr><td></td><td></td><td>Zero</td></tr> <tr><td></td><td></td><td>1 minute</td></tr> <tr><td></td><td></td><td>1 hour</td></tr> <tr><td></td><td></td><td>1 day</td></tr> </tbody> </table>	Biz Goal	IT Goal	Current			Zero			1 minute			1 hour			1 day
Biz Goal	IT Goal	Current																																													
		Zero																																													
		1 minute																																													
		1 hour																																													
		1 day																																													
Biz Goal	IT Goal	Current																																													
		Zero																																													
		1 minute																																													
		1 hour																																													
		1 day																																													
Biz Goal	IT Goal	Current																																													
		Zero																																													
		1 minute																																													
		1 hour																																													
		1 day																																													

BrentOzar.com/go/fail

Recovery Time Objective (RTO) - Downtime

		Zero-1 Sec	1 Minute	1 Hour	1 Day
Recovery Point Objective (RPO) - Data Loss	Zero-1 Sec	>\$1,000,000 Multiple active servers with bidirectional replication (usually requires code change)	\$100k-\$500k Clustering w/SAN, synch AlwaysOn Availability Groups (EE), synch database mirroring	\$100k-\$250k Synch SAN replication, synch VM replication	
	1 Minute			\$50k-\$250k Async AlwaysOn Avail Groups (EE), async DB mirroring (EE)	\$5k-\$100k Log shipping, async SAN replication, async VM replication
	1 Hour			\$5k-\$100k Log shipping, async SAN replication, async VM	
	1 Day				

Comparison of SQL Server Availability Features											
	>\$1mm	\$100k-\$500k			\$100k-\$250k		\$50k-\$250k		\$5k-\$100k		
	Replication	Failover Clustered Instances (FCI)	AlwaysOn Availability Groups - Synch	Database Mirroring - Synch	SAN Replication - Synch	VM Replication - Synch	AlwaysOn Availability Groups - Asynch	Database Mirroring - Asynch	Log Shipping	SAN Replication - Asynch	VM Replication - Asynch
Recovery Point Objective (Data Loss)											
Data loss possible	Yes	No*	No*	No*	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Typical RPO goal	Zero	Zero	Zero	Zero	Zero	Zero	1 Minute	1 Minute	1 Hour	1 Hour	1 Hour
Recovery Time Objective (Downtime)											
Failover automatic or manual	Automatic	Automatic	Automatic	Automatic	Optional	Optional	Manual	Manual	Manual	Manual	Manual
Can be failed over by the DBA alone	Yes	Yes	Yes	Yes	No	No	Yes	Yes	Yes	No	No
Easy to fail back and forth for patching	Some	Yes	Yes	Yes	No	No	Yes	Yes	Some	No	No
Multiple live writable servers	Optional	No	No	No	No	No	No	No	No	No	No
Fails over Agent jobs, SQL logins	No	Yes	No	No	Optional	Yes	No	No	No	Optional	Yes
3rd party apps fail over easily	No	Yes	Yes	Some	Yes	Yes	Yes	Some	No	Yes	Yes
Typical RTO goal	Zero	1 Minute	1 Minute	1 Minute	1 Hour	1 Hour	1 Hour	1 Hour	1 Day	1 Day	1 Day
Groups of Databases											
Are the same point in time	No	Yes	No*	No	Yes	Yes	No	No	No	No	No
Fail over together	No	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Hardware & Software Requirements											
SQL Server minimum version supported	Apps vary	Any	2012	2005	Any	Any	2012	2005	Any	Any	Any
SQL Server minimum edition required	Apps vary	Any	Enterprise	Standard	Any	Any	Enterprise	Enterprise	Any	Any	Any
May require application changes	Yes	No	No	No	No	No	No	No	No	No	No
Req storage for multiple copies of DBs	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Requires shared storage (SAN)	No	Yes	No	No	Yes	Yes	No	No	No	Yes	Yes
Requires virtualization	No	No	No	No	No	Yes	No	No	No	No	Yes
Can use local solid state storage	Yes	No	Yes	Yes	No	No	Yes	Yes	Yes	No	No
Performance											
Has noticeable performance impact	Maybe	No	Yes	Yes	Maybe	Maybe	Minimal	Minimal	No	Maybe	Maybe
Can offload backups, DBCCs, reports	Yes	No	Yes	No	No	No	Yes	No	Some	No	No



© 2014 Brent Ozar Unlimited. All rights reserved. Reproduction prohibited without the express written consent of Brent Ozar Unlimited.
Learn more at www.BrentOzar.com/go/fail. Your mileage may vary. Professional driver on closed course. Your eyesight is great.



17

© SQLIntersection. All rights reserved.
<http://www.SQLIntersection.com>

Comparison of SQL Server Availability Features											
	>\$1mm	\$100k-\$500k			\$100k-\$250k		\$50k-\$250k		\$5k-\$100k		
	Replication	Failover Clustered Instances (FCI)	AlwaysOn Availability Groups - Synch	Database Mirroring - Synch	SAN Replication - Synch	VM Replication - Synch	AlwaysOn Availability Groups - Asynch	Database Mirroring - Asynch	Log Shipping	SAN Replication - Asynch	VM Replication - Asynch
Recovery Point Objective (Data Loss)											
Data loss possible	Yes	No*	No*	No*	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Typical RPO goal	Zero	Zero	Zero	Zero	Zero	Zero	1 Minute	1 Minute	1 Hour	1 Hour	1 Hour
Recovery Time Objective (Downtime)											
Failover automatic or manual	Automatic	Automatic	Automatic	Automatic	Optional	Optional	Manual	Manual	Manual	Manual	Manual
Can be failed over by the DBA alone	Yes	Yes	Yes	Yes	No	No	Yes	Yes	Yes	No	No
Easy to fail back and forth for patching	Some	Yes	Yes	Yes	No	No	Yes	Yes	Some	No	No
Multiple live writable servers	Optional	No	No	No	No	No	No	No	No	No	No
Fails over Agent jobs, SQL logins	No	Yes	No	No	Optional	Yes	No	No	No	Optional	Yes
3rd party apps fail over easily	No	Yes	Yes	Some	Yes	Yes	Yes	Some	No	Yes	Yes
Typical RTO goal	Zero	1 Minute	1 Minute	1 Minute	1 Hour	1 Hour	1 Hour	1 Hour	1 Day	1 Day	1 Day
Groups of Databases											
Are the same point in time	No	Yes	No*	No	Yes	Yes	No	No	No	No	No
Fail over together	No	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Hardware & Software Requirements											
SQL Server minimum version supported	Apps vary	Any	2012	2005	Any	Any	2012	2005	Any	Any	Any
SQL Server minimum edition required	Apps vary	Any	Enterprise	Standard	Any	Any	Enterprise	Enterprise	Any	Any	Any
May require application changes	Yes	No	No	No	No	No	No	No	No	No	No
Req storage for multiple copies of DBs	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Requires shared storage (SAN)	No	Yes	No	No	Yes	Yes	No	No	No	Yes	Yes
Requires virtualization	No	No	No	No	No	Yes	No	No	No	No	Yes
Can use local solid state storage	Yes	No	Yes	Yes	No	No	Yes	Yes	Yes	No	No
Performance											
Has noticeable performance impact	Maybe	No	Yes	Yes	Maybe	Maybe	Minimal	Minimal	No	Maybe	Maybe
Can offload backups, DBCCs, reports	Yes	No	Yes	No	No	No	Yes	No	Some	No	No



© 2014 Brent Ozar Unlimited. All rights reserved. Reproduction prohibited without the express written consent of Brent Ozar Unlimited.
Learn more at www.BrentOzar.com/go/fail. Your mileage may vary. Professional driver on closed course. Your eyesight is great.



18

© SQLIntersection. All rights reserved.
<http://www.SQLIntersection.com>

May require application changes	Yes	No	No	
Req storage for multiple copies of DBs	Yes	No	Yes	
Requires shared storage (SAN)	No	Yes	No	
Requires virtualization	No	No	No	
Can use local solid state storage	Yes	No	Yes	

 SQL
intersection

19

© SQLIntersection. All rights reserved.
<http://www.SQLIntersection.com>

**Your budget
is limited.**

Don't get SAN storage, switch gear,
and a SAN admin unless you have to.

 SQL
intersection

20

© SQLIntersection. All rights reserved.
<http://www.SQLIntersection.com>

Or do you just want a SAN?

Already have a SAN that performs very well

Plan to move the database between hardware soon

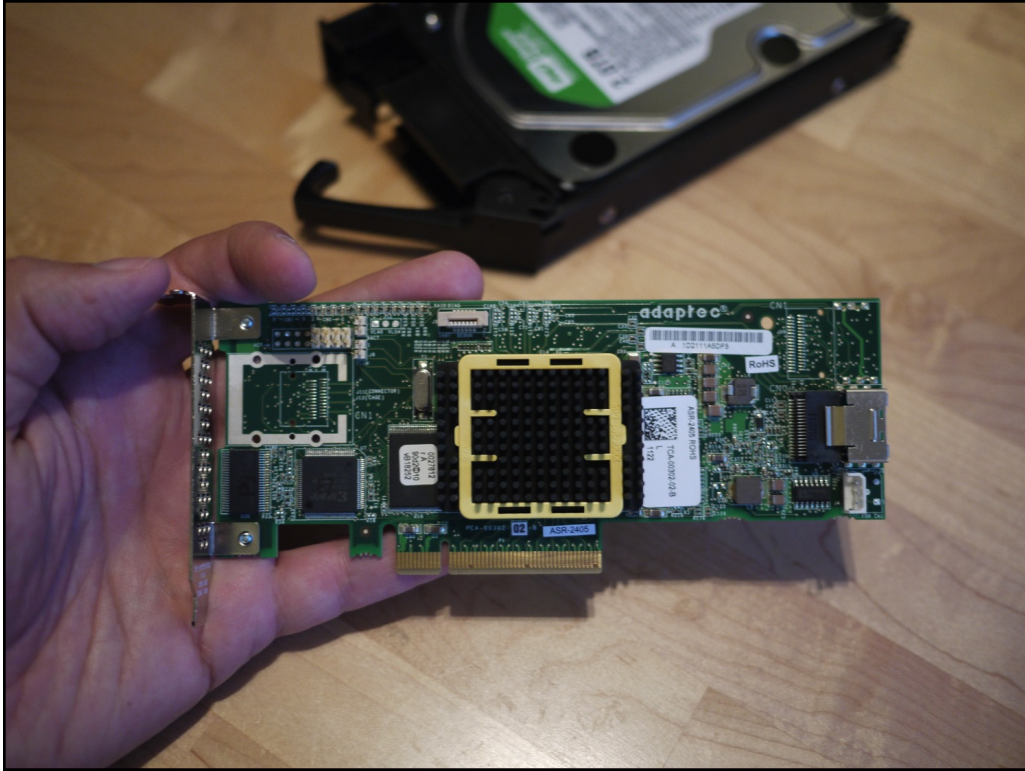
Data too large to fit in internal drives
(which is pretty unlikely these days, except blades)

Otherwise, local solid state is a shortcut to speed,
so your hardware choice should have local drive bays.

Why shared storage is usually slower

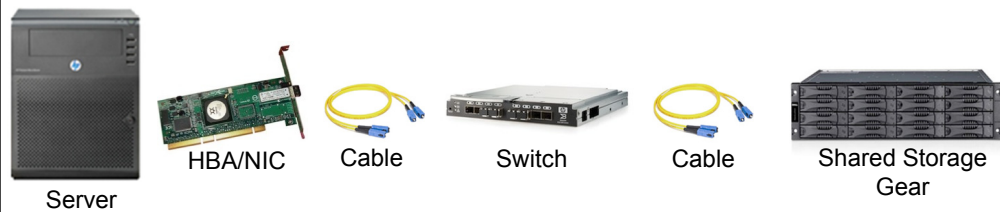




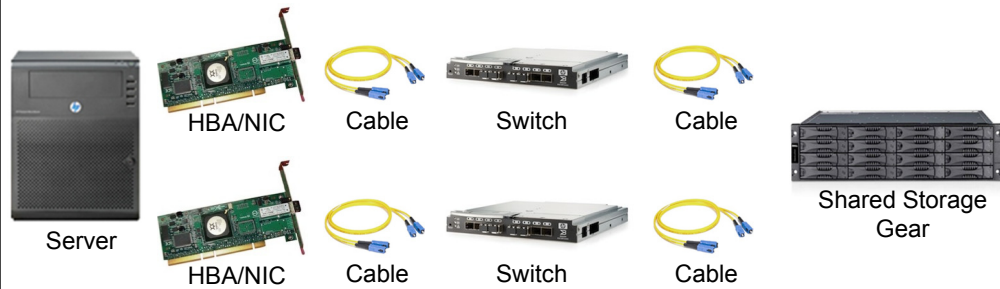




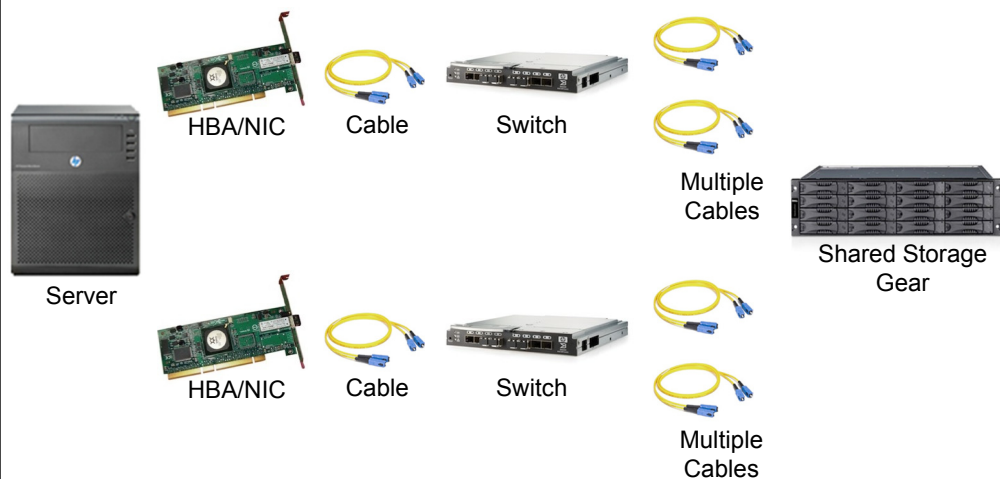
Simple SAN path



Two storage fabrics

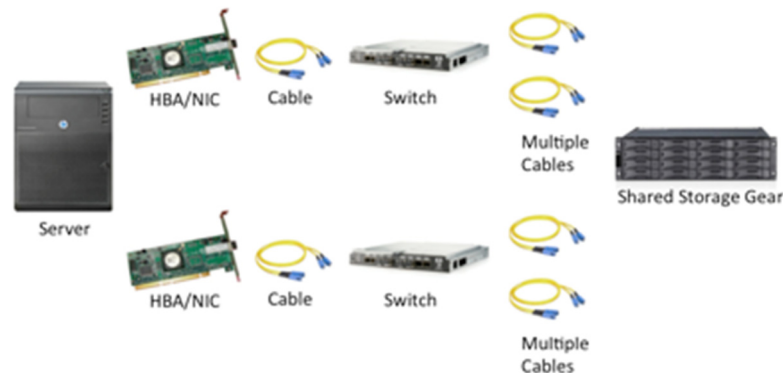


More paths, more throughput?



Active/Active Multipathing

Multiple active paths are actively carrying data, but they may only be carrying data for one drive letter, or they may only be active for writes or for reads.



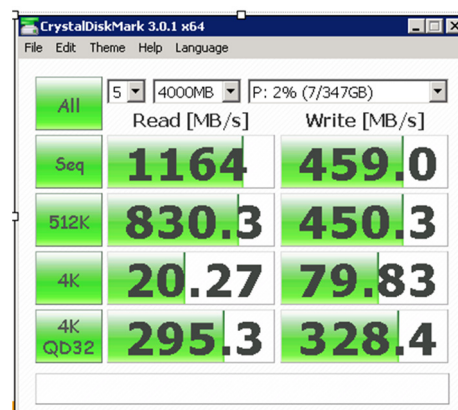
CrystalDiskMark

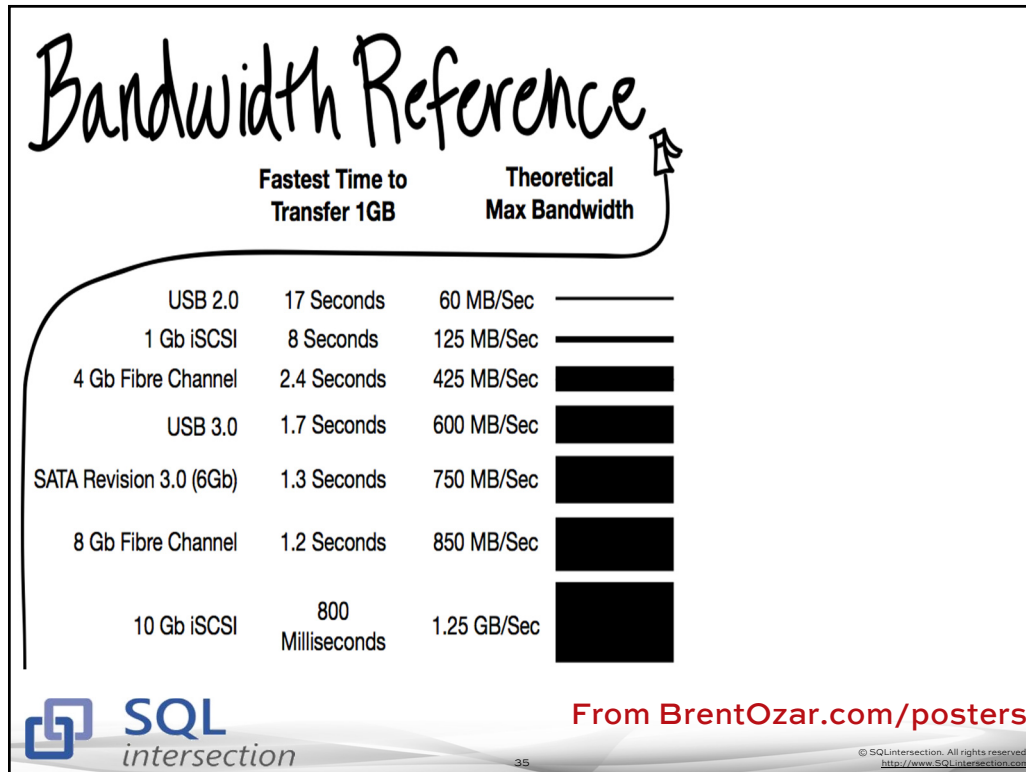
Free tool with instructions:
BrentOzar.com/go/cdm

Pick 5 tests, 4000MB test
file, drive letter

Sequential:
roughly akin to backups,
large table scans

4K QD32:
vaguely similar to active
OLTP server or TempDB





Your goals

Aim for 200MB per second of read throughput, per core of SQL Server

Example: 8 cores need at least 1,600MB/sec reads

Got less? Don't waste money on SQL licensing.

If you quickly and consistently hit a bottleneck lower than that, check your SAN fabric's throughput

Try using multiple paths, active/active multipathing

Easier way: cache the data in RAM, or ...

Storing the data locally on SSDs

“But SSDs aren’t reliable!”

News flash: neither are hard drives.

Mitigations:

- Backups
- DBCCs
- 2 hot spare drives (hey, we’ve got all those bays in the server chassis, might as well use them)
- Xanax

Dell PowerEdge 720XD



Top version: holds 26 2.5" drives (2 are on the back)

Bottom version: old 3.5" drives, holds 12



39

© SQLIntersection. All rights reserved.
<http://www.SQLIntersection.com>



HP DL380e G8

Similar, has models for different drive sizes.

You want 2.5" drive bays.



40

© SQLIntersection. All rights reserved.
<http://www.SQLIntersection.com>

Lots of drive bays for SSDs

Option 1: vendor-approved, enterprise-class drives.

- Vendor supports them, replaces when they break
- Well-tested – but this also means old and slow, because vendors don't test the latest and greatest
- Redonkulously expensive: 4TB capacity \$25K

Option B: off-the-shelf consumer drives

- You support them when they break (but they're dirt cheap, so buy spares)
- Redonkulously fast, cheap: 9TB capacity \$9K

The right hardware for Standard Edition

The Standard Standard

2-socket, 4-6 cores (get fastest cores possible)

128GB-192GB memory (SQL will use 64GB-128GB)

2 local SSDs in a mirrored pair for TempDB

Same spec kit whether you're doing cluster, non-clustered with a SAN, or non-clustered with local disks

Example: Dell PowerEdge 720 with 3.5GHz processors and 128GB memory: \$7K



43

© SQLIntersection. All rights reserved.
<http://www.SQLIntersection.com>

On Enterprise, how many CPU sockets do you need?



44

© SQLIntersection. All rights reserved.
<http://www.SQLIntersection.com>

Do you even need sockets?

	Standard Edition	Enterprise Edition
Physical	2 quad-core processors Licensing: \$14,344 Hardware: \$10,775 Total: \$25,119	2 quad-core processors Licensing: \$54,992 Hardware: \$23,206 Total: \$78,198
Virtual	Guest with 4 cores Licensing: \$7,172 Hardware: FREE* Total: Unknown	Host with 2 quad-core processors Licensing: \$54,992 Hardware: \$23,206 Total: \$78,198

Source: BrentOzar.com/go/smallest

Physical servers

Summary				
	Sockets	2	4	8
	Memory	768GB	3TB	4TB
	PCI Express slots	6	9	16
	Drive bays	26	10	8
Fastest low-core, low-licensing				
	Total speed, GHz	28	82	89
	Total cost	\$ 80,489	\$ 335,430	\$ 688,939
Most cores available				
	Total speed, GHz	65	168	192
	Total cost	\$ 192,613	\$ 569,547	\$ 965,267

# of CPU Sockets	2	4	8
Models	Dell R720		
	HP DL380 G8	HP DL580 G8	HP DL980 G7
Memory	768GB	3TB	4TB
PCI Express slots	6	9	16
Drive bays	26 Dell, 25 HP	10	8
Rack space required	2U	4U	8U
Fastest low-core, low-licensing			
Processor speed, GHz	3.5	3.4	1.86
# of cores	4	6	6
Total speed, GHz	28	82	89
Boost, theoretical		291%	319%
Cache, MB	15	38	18
Cache per core	3.8	6.3	3.0
Model	E5-2637 v2	E7-8893 v2	E7-4807
Hardware cost	\$ 25,497	\$ 170,454	\$ 358,987
SQL Server licensing	\$ 54,992	\$ 164,976	\$ 329,952
Total cost	\$ 80,489	\$ 335,430	\$ 688,939

# of CPU Sockets	2	4	8
Models	Dell R720		
	HP DL380 G8	HP DL580 G8	HP DL980 G7
Memory	768GB	3TB	4TB
PCI Express slots	6	9	16
Drive bays	26 Dell, 25 HP	10	8
Rack space required	2U	4U	8U
Most cores available			
Processor speed, GHz	2.7	2.8	2.4
# of cores	12	15	10
Total speed, GHz	65	168	192
Boost, theoretical		259%	296%
Cache, MB	30	38	30
Cache per core	2.5	2.5	3.0
Model	E5-2697 v2	E7-4890 v2	E7-4870
Hardware cost	\$ 27,637	\$ 157,107	\$ 415,347
SQL Server licensing	\$ 164,976	\$ 412,440	\$ 549,920
Total cost	\$ 192,613	\$ 569,547	\$ 965,267

Physical servers

Summary				
Sockets	2	4	8	
Memory	768GB	3TB	4TB	
PCI Express slots	6	9	16	
Drive bays	26	10	8	
Fastest low-core, low-licensing				
Total speed, GHz	28	82	89	
Total cost	\$ 80,489	\$ 335,430	\$ 688,939	
Most cores available				
Total speed, GHz	65	168	192	
Total cost	\$ 192,613	\$ 569,547	\$ 965,267	



49

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Boiling it down

Need <768GB RAM and <82GHz CPU? 2-socket.
The 25-26 local drive bays give you great options.

Need 1-3TB RAM and 8-10 PCI slots? 4-socket.
The March CPU updates beat 8-socket boxes.

Need over 3TB RAM or over 10 PCI slots?
8-socket, but start making plans to split this monster.



50

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Recap



51

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

The top 5 things we covered

1. Low RPO/RTO goals? Standard Edition VM.
2. If your RPO/RTO doesn't require shared storage, try local SSDs.
3. If you have to have a SAN, pathing is incredibly important. Get lots of fast paths.
4. CrystalDiskMark test for 200MB/sec/core.
5. In mid-2014, 2-sockets are an amazing deal.

Learn more: BrentOzar.com/go/faster



52

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>



BRENT OZAR
UNLIMITED®

ISV Best Practices Starter Kit

The Care and Feeding Of a Database Application

	Software Vendor	Customer
Design tables		
Write queries		
Deploy changes		
Tune queries		
Tune indexes		
Monitor performance		
Troubleshoot outages		
Install & configure SQL Server		
Buy & test hardware & storage		
Design & test HA/DR		

The Care and Feeding Of a Database Application

	Software Vendor	Customer
Design tables	✓	
Write queries	✓	
Deploy changes	✓	
Tune queries		
Tune indexes		
Monitor performance		
Troubleshoot outages		✓
Install & configure SQL Server		✓
Buy & test hardware & storage		✓
Design & test HA/DR		✓



3

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

The Care and Feeding Of a Database Application

	Software Vendor	Customer
Design tables	✓	
Write queries	✓	
Deploy changes	✓	
Tune queries		
Tune indexes		
Monitor performance		
Troubleshoot outages		✓
Install & configure SQL Server		✓
Buy & test hardware & storage		✓
Design & test HA/DR		✓



4

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Design and test a HA/DR solution

High Availability	Disaster Recovery	"Oops" Deletions																																																																																										
 For isolated server failures: Windows crash, RAID controller failure, SQL or Windows patch fails, C drive full, bad memory chip, wrong box unplugged RPO: Max acceptable data loss: <table border="1"> <thead> <tr> <th>Biz Goal</th> <th>IT Goal</th> <th>Current</th> </tr> </thead> <tbody> <tr><td></td><td></td><td>Zero</td></tr> <tr><td></td><td></td><td>1 minute</td></tr> <tr><td></td><td></td><td>1 hour</td></tr> <tr><td></td><td></td><td>1 day</td></tr> </tbody> </table> RTO: Max acceptable downtime: <table border="1"> <thead> <tr> <th>Biz Goal</th> <th>IT Goal</th> <th>Current</th> </tr> </thead> <tbody> <tr><td></td><td></td><td>Zero</td></tr> <tr><td></td><td></td><td>1 minute</td></tr> <tr><td></td><td></td><td>1 hour</td></tr> <tr><td></td><td></td><td>1 day</td></tr> </tbody> </table>	Biz Goal	IT Goal	Current			Zero			1 minute			1 hour			1 day	Biz Goal	IT Goal	Current			Zero			1 minute			1 hour			1 day	 For widespread outages: Data center power or network outage, domain controller failure, SAN failure, fire, quake, zombies in the data center RPO: Max acceptable data loss: <table border="1"> <thead> <tr> <th>Biz Goal</th> <th>IT Goal</th> <th>Current</th> </tr> </thead> <tbody> <tr><td></td><td></td><td>Zero</td></tr> <tr><td></td><td></td><td>1 minute</td></tr> <tr><td></td><td></td><td>1 hour</td></tr> <tr><td></td><td></td><td>1 day</td></tr> </tbody> </table> RTO: Max acceptable downtime: <table border="1"> <thead> <tr> <th>Biz Goal</th> <th>IT Goal</th> <th>Current</th> </tr> </thead> <tbody> <tr><td></td><td></td><td>Zero</td></tr> <tr><td></td><td></td><td>1 minute</td></tr> <tr><td></td><td></td><td>1 hour</td></tr> <tr><td></td><td></td><td>1 day</td></tr> </tbody> </table>	Biz Goal	IT Goal	Current			Zero			1 minute			1 hour			1 day	Biz Goal	IT Goal	Current			Zero			1 minute			1 hour			1 day	 For human T-SQL error: DELETE without a where clause, bug in stored procedure for updates, end user needing a restore due to human error RPO: Max acceptable data loss: <table border="1"> <thead> <tr> <th>Biz Goal</th> <th>IT Goal</th> <th>Current</th> </tr> </thead> <tbody> <tr><td></td><td></td><td>Zero</td></tr> <tr><td></td><td></td><td>1 minute</td></tr> <tr><td></td><td></td><td>1 hour</td></tr> <tr><td></td><td></td><td>1 day</td></tr> </tbody> </table> RTO: Max acceptable downtime: <table border="1"> <thead> <tr> <th>Biz Goal</th> <th>IT Goal</th> <th>Current</th> </tr> </thead> <tbody> <tr><td></td><td></td><td>Zero</td></tr> <tr><td></td><td></td><td>1 minute</td></tr> <tr><td></td><td></td><td>1 hour</td></tr> <tr><td></td><td></td><td>1 day</td></tr> </tbody> </table>	Biz Goal	IT Goal	Current			Zero			1 minute			1 hour			1 day	Biz Goal	IT Goal	Current			Zero			1 minute			1 hour			1 day
Biz Goal	IT Goal	Current																																																																																										
		Zero																																																																																										
		1 minute																																																																																										
		1 hour																																																																																										
		1 day																																																																																										
Biz Goal	IT Goal	Current																																																																																										
		Zero																																																																																										
		1 minute																																																																																										
		1 hour																																																																																										
		1 day																																																																																										
Biz Goal	IT Goal	Current																																																																																										
		Zero																																																																																										
		1 minute																																																																																										
		1 hour																																																																																										
		1 day																																																																																										
Biz Goal	IT Goal	Current																																																																																										
		Zero																																																																																										
		1 minute																																																																																										
		1 hour																																																																																										
		1 day																																																																																										
Biz Goal	IT Goal	Current																																																																																										
		Zero																																																																																										
		1 minute																																																																																										
		1 hour																																																																																										
		1 day																																																																																										
Biz Goal	IT Goal	Current																																																																																										
		Zero																																																																																										
		1 minute																																																																																										
		1 hour																																																																																										
		1 day																																																																																										



BRENT OZAR UNLIMITED © 2014 Brent Ozar Unlimited. All rights reserved. Reproduction prohibited without the express written consent of Brent Ozar Unlimited. Learn more at www.BrentOzar.com/go/fail. Your mileage may vary. Professional driver on closed course. Your eyesight is great.

Recovery Time Objective (RTO) - Downtime				
Recovery Point Objective (RPO) - Data Loss		Zero-1 Sec	1 Minute	1 Hour
	Zero-1 Sec	>\$1,000,000 Multiple active servers with bidirectional replication (usually requires code change)	\$100k-\$500k Clustering w/SAN, synch AlwaysOn Availability Groups (EE), synch database mirroring	\$100k-\$250k Synch SAN replication, synch VM replication
	1 Minute			\$50k-\$250k Async AlwaysOn Avail Groups (EE), async DB mirroring (EE)
	1 Hour			\$5k-\$100k Log shipping, async SAN replication, async VM
	1 Day			

©2014 Brent Ozar Unlimited. All rights reserved. Reproduction prohibited without the express written consent of Brent Ozar Unlimited.
Learn more at www.BrentOzar.com/go/fail - except in Nebraska, as Steve Ballmer explains: BrentOzar.com/go/nebraska

 **BRENT OZAR**
UNLIMITED

Comparison of SQL Server Availability Features											
	>\$1mm	\$100k-\$500k			\$100k-\$250k		\$50k-\$250k		\$5k-\$100k		
	Replication	Failover Clustered Instances (FCI)	AlwaysOn Availability Groups - Synch	Database Mirroring - Synch	SAN Replication - Synch	VM Replication - Synch	AlwaysOn Availability Groups - Async	Database Mirroring - Async	Log Shipping	SAN Replication - Async	VM Replication - Async
Recovery Point Objective (Data Loss)											
Data loss possible	Yes	No*	No*	No*	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Typical RPO goal	Zero	Zero	Zero	Zero	Zero	Zero	1 Minute	1 Minute	1 Hour	1 Hour	1 Hour
Recovery Time Objective (Downtime)											
Failover automatic or manual	Automatic	Automatic	Automatic	Automatic	Optional	Optional	Manual	Manual	Manual	Manual	Manual
Can be failed over by the DBA alone	Yes	Yes	Yes	Yes	No	No	Yes	Yes	Yes	No	No
Easy to fail back and forth for patching	Some	Yes	Yes	Yes	No	No	Yes	Yes	Some	No	No
Multiple live writeable servers	Optional	No	No	No	No	No	No	No	No	No	No
Fails over Agent jobs, SQL logins	No	Yes	No	No	Optional	Yes	No	No	No	Optional	Yes
3rd party apps fail over easily	No	Yes	Yes	Some	Yes	Yes	Yes	Some	No	Yes	Yes
Typical RTO goal	Zero	1 Minute	1 Minute	1 Minute	1 Hour	1 Hour	1 Hour	1 Hour	1 Day	1 Day	1 Day
Groups of Databases											
Are the same point in time	No	Yes	No*	No	Yes	Yes	No	No	No	No	No
Fail over together	No	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Hardware & Software Requirements											
SQL Server minimum version supported	Apps vary	Any	2012	2005	Any	Any	2012	2005	Any	Any	Any
SQL Server minimum edition required	Apps vary	Any	Enterprise	Standard	Any	Any	Enterprise	Enterprise	Any	Any	Any
May require application changes	Yes	No	No	No	No	No	No	No	No	No	No
Req storage for multiple copies of DBs	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Requires shared storage (SAN)	No	Yes	No	No	Yes	Yes	No	No	No	Yes	Yes
Requires virtualization	No	No	No	No	No	Yes	No	No	No	No	Yes
Can use local solid state storage	Yes	No	Yes	Yes	No	No	Yes	Yes	Yes	No	No
Performance											
Has noticeable performance impact	Maybe	No	Yes	Yes	Maybe	Maybe	Minimal	Minimal	No	Maybe	Maybe
Can offload backups, DBCCs, reports	Yes	No	Yes	No	No	No	Yes	No	Some	No	No

 **BRENT OZAR**
UNLIMITED

© 2014 Brent Ozar Unlimited. All rights reserved. Reproduction prohibited without the express written consent of Brent Ozar Unlimited.
Learn more at www.BrentOzar.com/go/fail. Your mission requires professional disaster-aligned expertise. Your survival is at stake.

High Availability gotchas for ISVs

- 0/0 – bidirectional transactional replication usually means data model changes and conflict resolution
- Failover Clustered Instances – everything fails over, including the server name. Totally transparent to your app – but things are down for a min.
- Database Mirroring – individual databases fail over, but not logins, Agent jobs, groups of databases, or the server name.
- AlwaysOn Availability Groups – like clustering, the name and groups of databases fail over together. Like mirroring, jobs/logins don't.



9

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

How to phrase it with clients

Your tolerance for data loss and downtime is up to your business goals and your budget.

Here's a helpful spreadsheet to help you guesstimate the costs of your goals: <http://BrentOzar.com/go/rpowsheet>

We have customers using (failover clustering, database mirroring, and duct tape) successfully. They typically have (two DBAs) on staff to manage it.



10

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

The Care and Feeding Of a Database Application

	Software Vendor	Customer
Design tables	✓	
Write queries	✓	
Deploy changes	✓	
Tune queries		
Tune indexes		
Monitor performance		
Troubleshoot outages		✓
Install & configure SQL Server		✓
Buy & test hardware & storage		✓
Design & test HA/DR		✓

Install and configure SQL Server

BrentOzar.com/go/setup

- Free 20-page e-book with the basics
- This way it comes from independent experts
- Updated with new SQL versions, guidance
- Make a list of additional changes you recommend as an addendum



BRENT OZAR
UNLIMITED

SQL Server Setup Checklist

Copyright Brent Ozar Unlimited® 2014

Last Updated: August 2, 2014



© Copyright 2014 Brent Ozar Unlimited®
Page 1 of 20

help@brentozar.com
<http://BrentOzar.com/go/setup>

The Care and Feeding Of a Database Application

	Software Vendor	Customer
Design tables	✓	
Write queries	✓	
Deploy changes	✓	
Tune queries		
Tune indexes		
Monitor performance		
Troubleshoot outages		✓
Install & configure SQL Server		✓
Buy & test hardware & storage		✓
Design & test HA/DR		✓



14

© SQLIntersection. All rights reserved.
<http://www.SQLIntersection.com>

Troubleshoot outages

Define outage

- One query is responding slowly
- One table is locked
- The database is offline, but the SQL Server is up
- The SQL Server service is down, but Windows is up
- Windows is not responding to pings
- The hardware cannot power on

Troubleshooting a performance emergency

- These steps are for your clients and your first tier support desk.
- Rule out server-level problems with `sp_AskBrent` –
<http://BrentOzar.com/askbrent>
- Find session-level problems with `sp_WhoIsActive @get_plans = 1`
<http://BrentOzar.com/go/active>
 - Look for long-running queries or blockers
 - Find out where they're coming from (app server/workstation name, app name)
 - Save the execution plans and queries for later troubleshooting



19

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Troubleshooting a corruption emergency

- If the SQL Server is online, but it's reporting corruption or an individual database shows as offline...
- Restore the most recent backups onto a different server.
- Have the client call Microsoft Support at 1-800-642-7676 for \$500.
 - They work the problem with you until it's fixed
 - Clearly state how many users are affected
 - Don't be afraid to ask for escalation
 - Be willing to stay on the phone with them continuously until it's fixed
- Fixing it yourself? <http://BrentOzar.com/go/corruption>
(and no, don't call us.)



20

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Troubleshooting an outage

- When the SQL Server isn't responding or multiple databases are down:
<http://BrentOzar.com/responder>
- Also has performance outage troubleshooting steps
- Print it out, fill it out as you go, and save it for the post-mortem

SQL Server First Responder Kit

This is a living document. Make it your own!

Instance Name: _____

Triaged By: _____

Date and Time: _____

1) Can you connect ?

Connect to SQL Server and run:

```
SELECT name, user_access_desc, state_desc, log_reuse_wait_desc
```

```
FROM sys.databases;
```

Did you connect successfully? _____

Anything notable in returned columns? _____

Problems connecting? _____

Try the dedicated admin connection: <http://BrentOzar.com/go/DAG>

2) Who's running?

Gather key pieces of information using Adam Machanic's sp_whoisactive:

```
EXEC dbo.sp_whoisactive;
```

Did you run this successfully? _____

How many rows did it return? _____

Was blocking present? (blocking_session_id column) _____

More info: <http://BrentOzar.com/go/active>

3) What does Brent say?

Check activity over a short sample using sp_AskBrent@:

```
EXEC dbo.sp_AskBrent @ExpertMode=1, @Seconds=10;
```

Did you run this successfully? _____

Did it find running queries? _____

Did it make a diagnosis? _____

More info: <http://BrentOzar.com/AskBrent>

4) Identify root cause of blocking (if present)

If there's blocking, save off the sql_text from the root query of the blocking. Get a screenshot that records the runtime, host name, and other relevant information from sp_whoisactive.

If you didn't catch it in time, consider setting up alerts or the Blocked Process Report. More info at: <http://BrentOzar.com/go/lockthis>

This document originally came from: <http://BrentOzar.com/Responder>



21

© SQLIntersection. All rights reserved.
<http://www.SQLIntersection.com>

The Care and Feeding Of a Database Application

	Software Vendor	Customer
Design tables	✓	
Write queries	✓	
Deploy changes	✓	
Tune queries		
Tune indexes		
Monitor performance		
Troubleshoot outages		✓
Install & configure SQL Server		✓
Buy & test hardware & storage		✓
Design & test HA/DR		✓



22

© SQLIntersection. All rights reserved.
<http://www.SQLIntersection.com>

Availability monitoring

Complexity	SQL Server Response Test	SQL Server Health	Windows Health
Low	Connection Test	Alerts for High Severity Errors	Ping Response
	Log in as application user	Notifications for SQL Agent job failures	Read recent errors and warnings in system log
	Measure time to run a query	Alerts for specialized components	Expand to application and security logs
High	Measure time to run a meaningful application query	Event notifications for availability-impacting events	Filter and add extra searches for informational events

What's most critical for your customers to know?

Complexity	SQL Server Response Test	SQL Server Health	Windows Health
Low	Connection Test	Alerts for High Severity Errors	Ping Response
	Log in as application user	Notifications for SQL Agent job failures	Read recent errors and warnings in system log
	Measure time to run a query	Alerts for specialized components	Expand to application and security logs
High	Measure time to run a meaningful application query	Event notifications for availability-impacting events	Filter and add extra searches for informational events

What's most meaningful for you?

Complexity	SQL Server Response Test	SQL Server Health	Windows Health
Low	Connection Test	Alerts for High Severity Errors	Ping Response
	Log in as application user	Notifications for SQL Agent job failures	Read recent errors and warnings in system log
	Measure time to run a query	Alerts for specialized components	Expand to application and security logs
High	Measure time to run a meaningful application query	Event notifications for availability-impacting events	Filter and add extra searches for informational events

Performance monitoring and trending

Comple xity	SQL Server Query Performance	SQL Server General Performance	Windows Performance
Low	Query snapshots	Basic SQL counters	Basic counters (Processor, Memory, Storage)
	Sample statement execution data from DMVs	Trend SQL waits, file waits, using DMVs	Counters from related systems (hypervisor host, apps, etc.)
	Measure workloads with server side trace or extended events	Graph trends in index usage	Correlate server side traces with perfmon data
High	Graph execution time of meaningful application query	Analyze trends of data on a weekly or monthly basis	Analyze trends of data on a weekly or monthly basis

What do customers typically monitor?

Comple xity	SQL Server Query Performance	SQL Server General Performance	Windows Performance
Low	Query snapshots	Basic SQL counters	Basic counters (Processor, Memory, Storage)
	Sample statement execution data from DMVs	Trend SQL waits, file waits, using DMVs	Counters from related systems (hypervisor host, apps, etc.)
	Measure workloads with server side trace or extended events	Graph trends in index usage	Correlate server side traces with perfmon data
High	Graph execution time of meaningful application query	Analyze trends of data on a weekly or monthly basis	Analyze trends of data on a weekly or monthly basis



29

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

What's the most meaningful for you?

Comple xity	SQL Server Query Performance	SQL Server General Performance	Windows Performance
Low	Query snapshots	Basic SQL counters	Basic counters (Processor, Memory, Storage)
	Sample statement execution data from DMVs	Trend SQL waits, file waits, using DMVs	Counters from related systems (hypervisor host, apps, etc.)
	Measure workloads with server side trace or extended events	Graph trends in index usage	Correlate server side traces with perfmon data
High	Graph execution time of meaningful application query	Analyze trends of data on a weekly or monthly basis	Analyze trends of data on a weekly or monthly basis



30

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

The solution: split monitoring.

Your customers need monitoring for:

- **Availability:** ping times, connection tests, errors in event logs
- **Performance:** Perfmon counters from BrentOzar.com/go/perfmon
- **Any mainstream tool can accomplish this goal, typically \$500-\$1k per server**
- **At your largest and most critical clients, you need monitoring for:**
 - Average execution time of your most meaningful application queries
 - List of top 10 resource-consuming queries per week
 - Index utilization – sp_BlitzIndex® copy/pasted into Excel



31

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Your performance monitoring needs

At your largest and most critical clients, you need monitoring for:

- Average execution time of your most meaningful application queries
- List of top 10 resource-consuming queries per week
- Index utilization – sp_BlitzIndex® copy/pasted into Excel

Ways to gather it:

- Build telemetry into your application
- Have your support staff or the customer's IT staff run sp_BlitzCache®, sp_BlitzIndex® and export the data to spreadsheet/file
- SQL Sentry Performance Advisor
- Dell (Quest) Spotlight on SQL Server Enterprise
- Idera SQL Diagnostic Manager



32

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

The Care and Feeding Of a Database Application

	Software Vendor	Customer
Design tables	✓	
Write queries	✓	
Deploy changes	✓	
Tune queries		
Tune indexes		
Monitor performance		
Troubleshoot outages		✓
Install & configure SQL Server		✓
Buy & test hardware & storage		✓
Design & test HA/DR		✓

Tune indexes

Two options to tune indexes

The sealed-box approach used by MS SharePoint, Dynamics:

- Under no circumstances is a customer allowed to touch the database.
- Change indexes, views, stored procs, etc, and you lose support.
- When the customer experiences performance problems, they contact the vendor, and the vendor gets remote access to the SQL Server.
- The vendor may make short-term recommendations for the customer, but usually any changes are built into the next version of the software, which comes out later.

The customer-tuned approach used by SAP, Relativity:

- Vendors ship a set of starter indexes.
- Customer DBA teams create, alter, and disable indexes to reflect how the application is used in their environment.



35

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

The sealed box approach

- Your support team needs a “gold standard” database from each version
- Ideally, they also need a way to detect whether the customer’s database has drifted from the gold standard version
 - sp_BlitzIndex® can be a simple sanity check – look for heaps, too many indexes
- They need a list of harmless changes, versus showstoppers:
 - Added fields to tables
 - Dropped clustered indexes
 - Dropped/disabled nonclustered indexes
 - Added indexes
 - Added indexed views
- They need an escalation process – and typically this involves sales



36

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

The customer-tuned approach

- Your packaged app should aim for ≤ 5 nonclustered indexes per table
- Your deployment scripts need to gracefully handle indexes that don't exist, or extra indexes
- It's okay if customers disable indexes: encourage them to drop rather than disabling
- Your support team still needs a "gold standard" database and drift detection tools because critical indexes may have gone missing
 - sp_BlitzIndex® looking for high-value missing indexes is a short-term fix
- Point customers to index training resources:
SQL Server Central's Stairway to SQL Server Indexes:
<http://www.sqlservercentral.com/stairway/72399/>



37

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

The Care and Feeding Of a Database Application

	Software Vendor	Customer
Design tables	✓	
Write queries	✓	
Deploy changes	✓	
Tune queries		
Tune indexes		
Monitor performance		
Troubleshoot outages		✓
Install & configure SQL Server		✓
Buy & test hardware & storage		✓
Design & test HA/DR		✓



38

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Tune indexes

You can't tune what you can't see.

- **The customer has to give you remote access to clear, unbiased data.**
- **Example: SQL Sentry Performance Advisor's cloud-based reports**
- **On a monthly/weekly basis:**
 - The customer builds a list of the top resource-intensive queries (using sp_BlitzCache or your monitoring tool)
 - Whoever's managing indexes tries to mitigate the problems with indexing first
 - The list is then handed off to the vendor to improve it with code
 - The vendor communicates which ones will be fixed in a future version of code, or which ones are expected to be at that level of resource usage

Document your company's defaults.

	Software Vendor	Customer
Design tables	✓	
Write queries	✓	
Deploy changes	✓	
Tune queries		
Tune indexes		
Monitor performance		
Troubleshoot outages		✓
Install & configure SQL Server		✓
Buy & test hardware & storage		✓
Design & test HA/DR		✓



41

© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Questions?



Don't forget to complete an online evaluation on EventBoard!

Session: POSTCON05

Developer's Guide to Tuning Somebody Else's SQL Server

Your evaluation helps organizers build better conferences
and helps speakers improve their sessions.



Thank you!