



BRENT OZAR
UNLIMITED®

Indexing to Fix Performance Problems

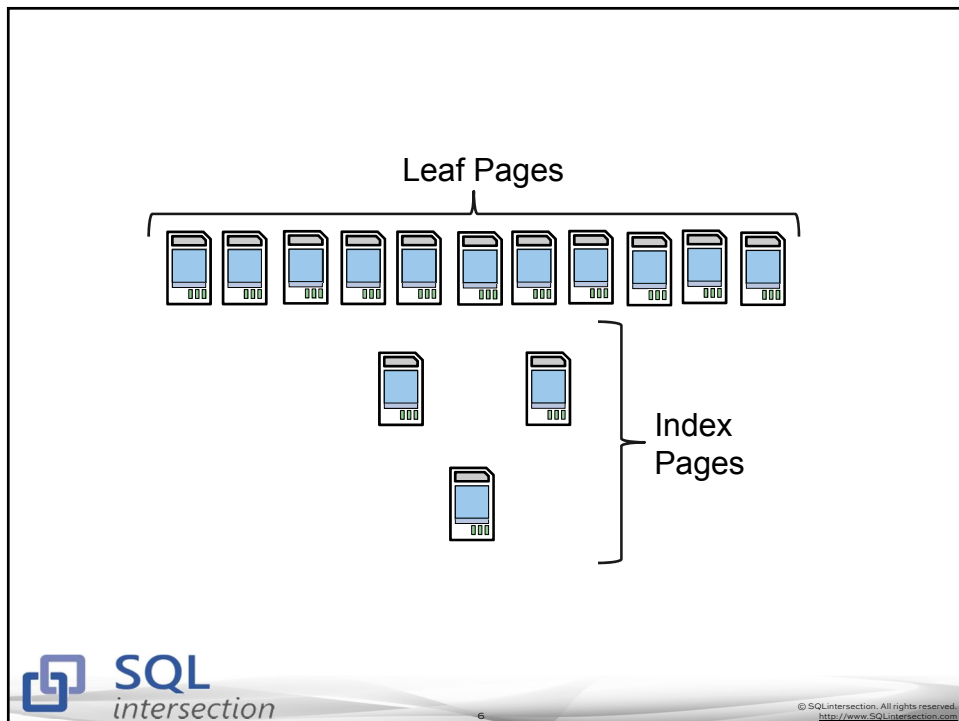
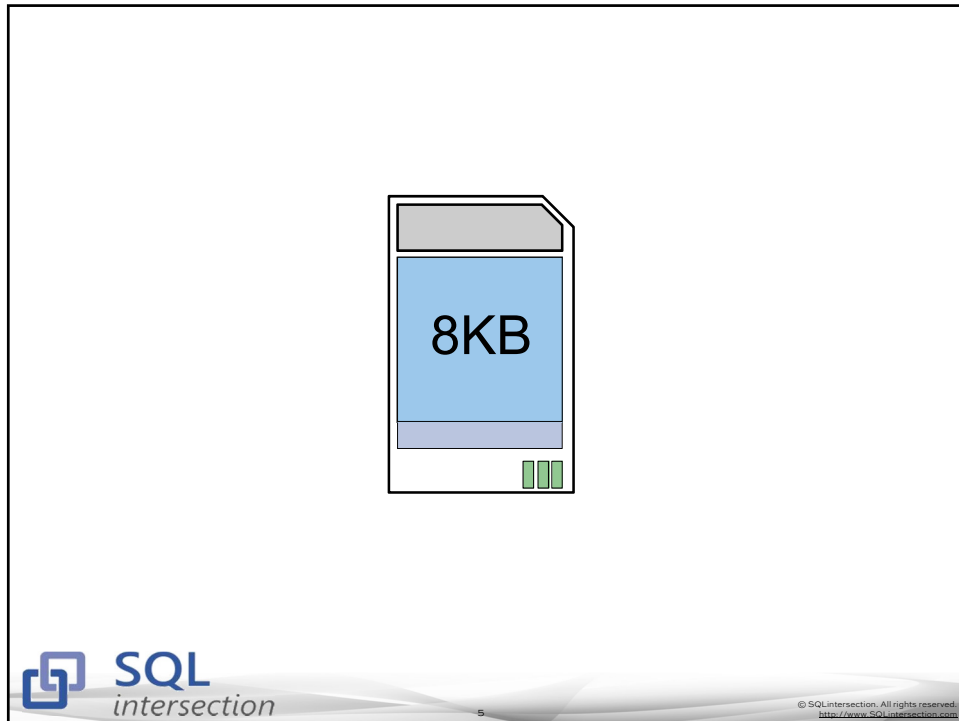
Confession: we haven't been
tuning indexes regularly

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

“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

“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

“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**

Advice is complicated

Missing index DMV “requests”

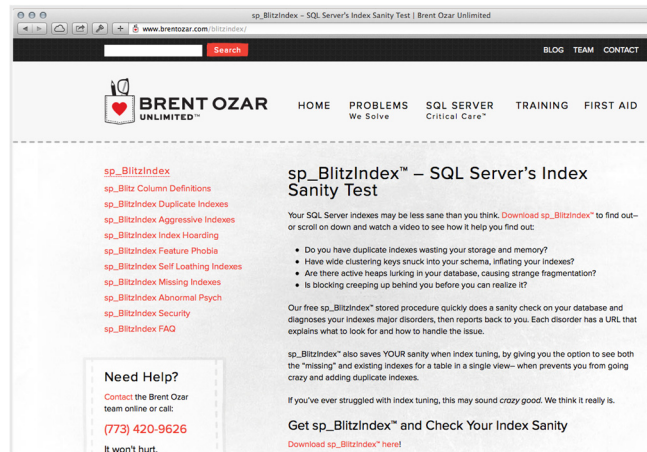
- Recommend super wide indexes
- Don’t de-duplicate requests
- Don’t get thrown for all types of queries
- Get cleared at tricky times

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>

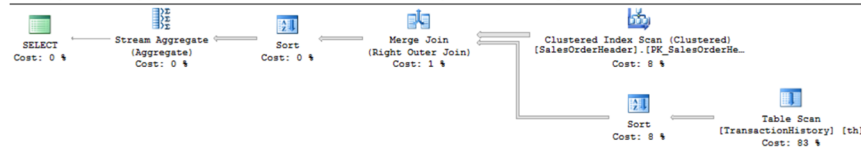
Syntax to run in “diagnose” mode

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

DEMO 2: Diagnose!

The top query plan

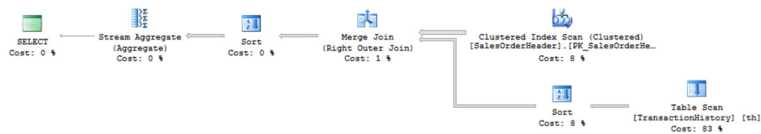
```
Query 2: Query cost (relative to the batch): 100%
SELECT TransactionType, SUM(Quantity) AS TotalQuantity, SUM(ActualCost) AS TotalCost, MAX(sh.ShipDate) AS LastShippedDate FROM [Production].[TransactionHistory]
Missing index (Impact 74.5132): CREATE NONCLUSTERED INDEX [Name of Missing Index, sysname.>]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



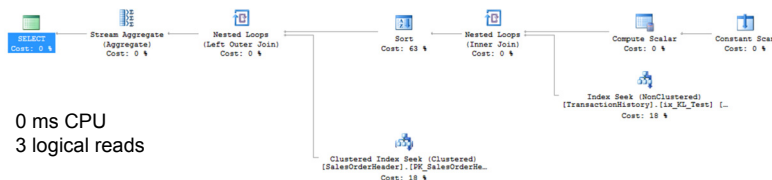
Create index...

```
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