BRENT OZAR
UNLIMITED®

The D.E.A.T.H. Method: Adding Indexes with the DMVs

2.1 p1

© Brent Ozar Unlimited®. All rights reserved.

In the last lab, we jumped to T.

Just once	**Dedupe** – reduce overlapping indexes **Eliminate** – unused indexes
Weekly for 1 month	**Add** – badly needed missing indexes
Do this only AFTER the easy stuff above	**Tune** – indexes for specific queries **Heaps** – usually need clustered indexes

2.1 p2

© Brent Ozar Unlimited®. All rights reserved.

The A part relies on Clippy.

SQL Server gives us missing indexes in:

- sys.dm_db_missing_index_details
- Query plans and the plan cache

He doesn't care about:

- The size of the index
- The index's overhead on D/U/I operations
- Your other existing indexes
- A lot of operations in the plan (GROUP BY)

© Brent Ozar Unlimited®. All rights reserved.

Clippy's order of fields

1. Equality searches in the query
   (ordered by the field order in the table)
2. Inequality searches
   (ordered by the field order in the table)
3. Includes
   (which might actually need to be sorted)

2.1 p4

© Brent Ozar Unlimited®. All rights reserved.

Time to level up.

In Fundamentals of Index Tuning,
I gave you queries,
and had you come up with Clippy's field ordering.

Here in Mastering Index Tuning,
I'm going to give you Clippy's suggestion,
and I want you to reverse engineer the query.
(There are multiple answers for each index.)

2.1 p5

© Brent Ozar Unlimited®. All rights reserved.

Quiz 1:
1 query at a time

2.1 p6

© Brent Ozar Unlimited®. All rights reserved.

Missing index suggestion #1

```
Missing Index (Impact 99.992): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>] ON [dbo].[Users] ([DisplayName],[Location])
```

2.1 p7

© Brent Ozar Unlimited®. All rights reserved.

Missing index suggestion #1

```
SELECT *
  FROM dbo.Users
  WHERE DisplayName = 'Brent Ozar'
    AND Location = 'San Diego, CA, USA';
GO
```

50 %

Messages | Execution plan

```
Query 1: Query cost (relative to the batch): 100%
SELECT * FROM dbo.Users WHERE DisplayName = 'Brent Ozar' AND Location = 'San Diego, CA, USA'
Missing Index (Impact 99.992): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>] ON [dbo].[Users] ([DisplayName],[Location])
```

SELECT
Cost: 0 %

Clustered Index Scan (Clustered)
[Users].[PK_Users_Id]
Cost: 100 %

In this case, key order really doesn't matter. (Even a single-column index on either field would work.)

2.1 p8

© Brent Ozar Unlimited®. All rights reserved.

Missing index suggestion #2

```
Missing Index (Impact 99.599): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>] ON [dbo].[Users] ([Location],[DisplayName]) INCLUDE ([AboutMe],[Age],[C
```


```
<MissingIndexes>
  <MissingIndexGroup Impact="99.599">
    <MissingIndex Database="[StackOverflow]" Schema="[dbo]" Table="[Users]">
      <ColumnGroup Usage="EQUALITY">
        <Column Name="[Location]" ColumnId="9" />
      </ColumnGroup>
      <ColumnGroup Usage="INEQUALITY">
        <Column Name="[DisplayName]" ColumnId="5" />
      </ColumnGroup>
      <ColumnGroup Usage="INCLUDE">
        <Column Name="[AboutMe]" ColumnId="2" />
        <Column Name="[Age]" ColumnId="3" />
        <Column Name="[CreationDate]" ColumnId="4" />
        <Column Name="[DownVotes]" ColumnId="6" />
        <Column Name="[EmailHash]" ColumnId="7" />
        <Column Name="[LastAccessDate]" ColumnId="8" />
        <Column Name="[Reputation]" ColumnId="10" />
        <Column Name="[UpVotes]" ColumnId="11" />
        <Column Name="[Views]" ColumnId="12" />
        <Column Name="[WebsiteUrl]" ColumnId="13" />
        <Column Name="[AccountId]" ColumnId="14" />
```


2.1 p9

© Brent Ozar Unlimited®. All rights reserved.

Missing index suggestion #2

```
SELECT *
  FROM dbo.Users
  WHERE DisplayName <> 'Brent Ozar'
    AND Location = 'San Diego, CA, USA';
GO
```

```
Messages   Execution plan
Query 1: Query cost (relative to the batch): 100%
SELECT * FROM dbo.Users WHERE DisplayName <> 'Brent Ozar' AND Location = 'San Diego, CA, USA'
Missing Index (Impact 99.599): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>] ON [dbo].[Users] ([Location],[DisplayName]) INCLUDE ([AboutMe],[Age],[C
SELECT
Cost: 0 %
Clustered Index Scan (Clustered)
[Users].[PK_Users_Id]
Cost: 100 %
```

Clippy puts equality fields first, then inequalities, all sorted by their field order in the table.

2.1 p10

© Brent Ozar Unlimited®. All rights reserved.

Missing index suggestion #3

```
Missing Index (Impact 59.5936): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>] ON [dbo].[Users] ([DisplayName]) INCLUDE ([Location])
```


© Brent Ozar Unlimited®. All rights reserved.

Missing index suggestion #3

```
SELECT Location
  FROM dbo.Users
  WHERE DisplayName LIKE 'B%';
GO
```

50 %

Messages | Execution plan

```
Query 1: Query cost (relative to the batch): 100%
SELECT Location FROM dbo.Users WHERE DisplayName LIKE 'B%'
Missing Index (Impact 59.5936): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>] ON [dbo].[Users] ([DisplayName]) INCLUDE ([Location])
```

SELECT
Cost: 0 %

Clustered Index Scan (Clustered)
[Users].[PK_Users_Id]
Cost: 100 %

In this case, the Location doesn't need to be sorted.

2.1 p12

© Brent Ozar Unlimited®. All rights reserved.

But this will also produce it...

```
SELECT Location
    FROM dbo.Users
    WHERE DisplayName LIKE 'B%'
    ORDER BY Location;
```

50 %

Messages | Execution plan

```
Query 1: Query cost (relative to the batch): 100%
SELECT Location FROM dbo.Users WHERE DisplayName LIKE 'B%' ORDER BY Location
Missing Index (Impact 46.6992): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>] ON [dbo].[Users] ([DisplayName]) INCLUDE ([Location])
```

Remember, Clippy has some blind spots like GROUP BY and ORDER BY. Generally, if you see a single-column index with includes, he probably needs one of the includes as a key, too.

2.1 p13

© Brent Ozar Unlimited®. All rights reserved.

Quiz 2:
2 queries at a time

2.1 p14

© Brent Ozar Unlimited®. All rights reserved.

Now it gets harder.

```
Missing Index (Impact 99.9968): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>] ON [dbo].[Users] ([LastAccessDate],[WebsiteUrl])
```

```
Missing Index (Impact 99.9956): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>] ON [dbo].[Users] ([DownVotes],[WebsiteUrl])
```

1. What queries might have built these hints?
2. Can one index satisfy both queries? Why?

2.1 p15

© Brent Ozar Unlimited®. All rights reserved.

```xml
<MissingIndexes>
  <MissingIndexGroup Impact="99.9968">
    <MissingIndex Database="[StackOverflow]" Schema="[dbo]" Table="[Users]">
      <ColumnGroup Usage="EQUALITY">
        <Column Name="[LastAccessDate]" ColumnId="8" />
        <Column Name="[WebsiteUrl]" ColumnId="13" />
      </ColumnGroup>
    </MissingIndex>
  </MissingIndexGroup>
</MissingIndexes>

<MissingIndexes>
  <MissingIndexGroup Impact="99.9956">
    <MissingIndex Database="[StackOverflow]" Schema="[dbo]" Table="[Users]">
      <ColumnGroup Usage="EQUALITY">
        <Column Name="[DownVotes]" ColumnId="6" />
        <Column Name="[WebsiteUrl]" ColumnId="13" />
      </ColumnGroup>
    </MissingIndex>
  </MissingIndexGroup>
</MissingIndexes>
```

2.1 p16

© Brent Ozar Unlimited®. All rights reserved.

```
SELECT Id
  FROM dbo.Users
  WHERE LastAccessDate = GETDATE()
    AND WebsiteUrl = 'https://www.BrentOzar.com';

SELECT Id
  FROM dbo.Users
  WHERE DownVotes = 0
    AND WebsiteUrl = 'https://www.BrentOzar.com';
```

150 %

Messages | Execution plan

```
Query 1: Query cost (relative to the batch): 50%
SELECT Id FROM dbo.Users WHERE LastAccessDate = GETDATE() AND WebsiteUrl = 'https://www.BrentOzar.com'
Missing Index (Impact 99.9968): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>] ON [dbo].[Users] ([LastAccessDate],[WebsiteUrl])
```

SELECT Cost: 0 % ← Clustered Index Scan (Clustered) [Users].[PK_Users_Id] Cost: 100 %

```
Query 2: Query cost (relative to the batch): 50%
; SELECT Id FROM dbo.Users WHERE DownVotes = 0 AND WebsiteUrl = 'https://www.BrentOzar.com'
Missing Index (Impact 99.9956): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>] ON [dbo].[Users] ([DownVotes],[WebsiteUrl])
```

SELECT Cost: 0 % ← Clustered Index Scan (Clustered) [Users].[PK_Users_Id] Cost: 100 %

2.1 p17

© Brent Ozar Unlimited®. All rights reserved.

Can one index help both?

```sql
SELECT Id
  FROM dbo.Users
  WHERE LastAccessDate = GETDATE()
    AND WebsiteUrl = 'https://www.BrentOzar.com';

SELECT Id
  FROM dbo.Users
  WHERE DownVotes = 0
    AND WebsiteUrl = 'https://www.BrentOzar.com';
```

Sure, as long as it leads with WebsiteUrl.

Any of these would help:

- (WebsiteUrl, DownVotes, LastAccessDate)
- (WebsiteUrl, LastAccessDate, DownVotes)
- (WebsiteUrl, DownVotes) INCLUDE (LastAccessDate)
- (WebsiteUrl, LastAccessDate) INCLUDE (DownVotes)
- Maybe even: (WebsiteUrl) INCLUDE (DownVotes, LastAccessDate)

2.1 p18

© Brent Ozar Unlimited®. All rights reserved.

Perfect is the enemy of good.

Voltaire on index design

2.1 p19

© Brent Ozar Unlimited®. All rights reserved.

Our job in index tuning

Look at the recommended indexes by:

- Equality fields
- Inequality fields
- Includes (which sometimes need to be promoted)

Look for things they have in common

Make guess based on our experience
with the queries and the data

Revisit it later in the T part of D.E.A.T.H.

2.1 p20

© Brent Ozar Unlimited®. All rights reserved.

Quiz 3:
3 queries at a time

2.1 p21

© Brent Ozar Unlimited®. All rights reserved.

We've been looking at plans...

```
Missing Index (Impact 99.9977): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>] ON [dbo].[Votes] ([PostId]) INCLUDE ([VoteTypeId])

Missing Index (Impact 99.9970): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>] ON [dbo].[Votes] ([PostId]) INCLUDE ([UserId],[Bounty

Missing Index (Impact 99.9996): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>] ON [dbo].[Votes] ([PostId],[VoteTypeId]) INCLUDE ([Us
```

2.1 p22

© Brent Ozar Unlimited®. All rights reserved.

But let's switch to sp_BlitzIndex

sp_BlitzIndex

sults | Messages

Priority	Finding	Details: schema.table.index(indexid)	Definition: [Property] ColumnName [datatype maxbytes]	Usage
1	sp_BlitzIndex(TM) v7.5 - April 27, 2019: Da...	http://FirstResponderKit.org	Server: SQL2017LAB1 Days Uptime: 0.25	
50	Indexophobia: High value missing index	[StackOverflow].[dbo].[Votes] Ext. benefit per day: 1,045,472	EQUALITY: [PostId] INEQUALITY: [VoteTypeId] INCLUDES: [UserId], [BountyAmount], [CreationDate]	3 uses; Impact: 100.0%; Avg query cost: 871.2268
50	Indexophobia: High value missing index	[StackOverflow].[dbo].[Votes] Ext. benefit per day: 900,733	EQUALITY: [PostId] INCLUDES: [VoteTypeId]	3 uses; Impact: 100.0%; Avg query cost: 750.6115
50	Indexophobia: High value missing index	[StackOverflow].[dbo].[Votes] Ext. benefit per day: 900,733	EQUALITY: [PostId] INCLUDES: [UserId], [BountyAmount], [VoteTypeId], [CreationDate]	3 uses; Impact: 100.0%; Avg query cost: 750.6114

1. What queries might have built these hints?
2. What indexes would you create to help as many of the queries as practical?

2.1 p23

© Brent Ozar Unlimited®. All rights reserved.

Guessing at it

EQUALITY: [PostId] INEQUALITY: [VoteTypeId] INCLUDES: [UserId], [BountyAmount], [CreationDate]
EQUALITY: [PostId] INCLUDES: [VoteTypeId]
EQUALITY: [PostId] INCLUDES: [UserId], [BountyAmount], [VoteTypeId], [CreationDate]

PostId, VoteTypeId would satisfy the top two

Depending on the uniqueness of PostId, it might even satisfy the 3rd one (but not cover it)

2.1 p24

© Brent Ozar Unlimited®. All rights reserved.

How'd your indexes do with these?

```
SELECT VoteTypeId, COUNT(*) AS TotalVotes
  FROM dbo.Votes
  WHERE PostId = 12345
  GROUP BY VoteTypeId;

SELECT *
  FROM dbo.Votes
  WHERE PostId = 12345
  ORDER BY CreationDate;

SELECT *
  FROM dbo.Votes
  WHERE PostId = 12345
    AND VoteTypeId IN (2, 3);
```


© Brent Ozar Unlimited®. All rights reserved.

What everyone wants to know:

“But how do I find the real queries?”

2.1 p26

© Brent Ozar Unlimited®. All rights reserved.

I don't usually do this in the A part.

Just once	**Dedupe** – reduce overlapping indexes **Eliminate** – unused indexes
Weekly for 1 month	**Add** – badly needed missing indexes
Do this only AFTER the easy stuff above	**Tune** – indexes for specific queries **Heaps** – usually need clustered indexes

2.1 p27

© Brent Ozar Unlimited®. All rights reserved.

Fastest: SQL Server 2019 method

SQL Server 2019 has a new undocumented DMV:
sys.dm_db_missing_index_group_stats_query.

sp_BlitzIndex Nov 2020 version added support, but it's still very experimental.

sp_BlitzIndex @TableName = 'mytable'
Look in the far right of the missing indexes section

sp_BlitzIndex @Mode = 3
(lists missing index requests)

© Brent Ozar Unlimited®. All rights reserved.

Fast, but less accurate: top reads

sp_BlitzCache @SortOrder = 'reads',
@DatabaseName = 'mydb'

- Lists the top resource-intensive queries
- They probably need an index or two
- Probably not the specific index you're looking for
- Doesn't catch queries that originate in other dbs, queries that aren't in the plan cache now

© Brent Ozar Unlimited®. All rights reserved.

Slow: query the entire cache

sp_BlitzCache queries just top readers, so it's fast.

This query checks the entire plan cache, looking at the XML to see if there's a missing index on a specific table you're looking for: http://www.sqlnuggets.com/blog/sql-scripts-find-queries-that-have-missing-index-requests/

Doesn't catch queries that originate in other dbs, queries that aren't in the plan cache now

2.1 p30

© Brent Ozar Unlimited®. All rights reserved.

Slow: Query Store

Query Store is just like a persisted version of the plan cache, and you can query the plans in it.

This query finds ALL missing index requests, so it's going to be slow and unfiltered:
https://www.scarydba.com/2019/03/11/missing-indexes-in-the-query-store/

Drawbacks:

- Requires Query Store (which has many drawbacks)
- Only catches plans that made it to Query Store

© Brent Ozar Unlimited®. All rights reserved.

Recap

2.1 p32

© Brent Ozar Unlimited®. All rights reserved.

Field order guidelines (not rules)

Fields you use the most often should go first

When doing range scans, selectivity matters

Compromise involves:

- Prioritizing reads vs writes
- Prioritizing which queries need to be fastest
- Caching data in the application
- Spending more money on hardware

In the A phase, I don't usually check queries.

2.1 p33

© Brent Ozar Unlimited®. All rights reserved.