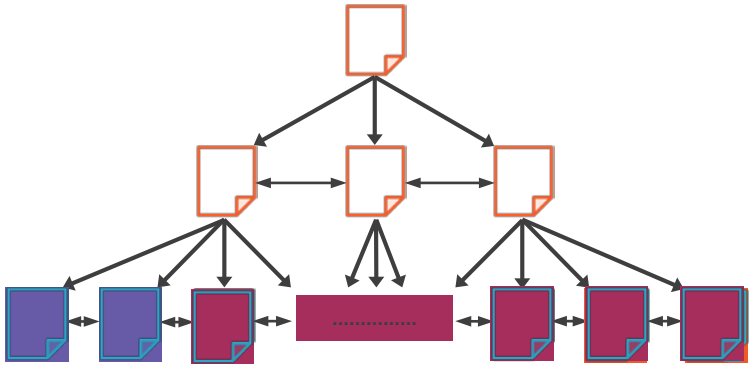
# Understanding Filtered Indexes

**Kimberly L. Tripp**
OWNER/PRESIDENT – SQLSKILLS.COM
@kimberlyltripp www.sqlskills.com/blogs/kimberly

# Module Overview

- Filtered indexes and requirements
- Filtered index use cases
- Filtered indexes and interval subsumption
- Filtered indexes and plan caching
- Filtered index strategies

# What is a Filtered Index?



**Traditionally, row-based index has one entry for every table row (in index order) in the leaf level**

**The B-tree is used to navigate to the leaf level**

**What if you're always using an index for just one portion of data**

- WHERE Active = 1
- WHERE Status > 8
- WHERE SalesPerson IS NOT NULL
- WHERE OrderDate BETWEEN ...

**The point: covering ALL of the data is expensive!**

# Filtered Indexes

Only applies to nonclustered indexes

Overall size might be significantly lower than an unfiltered index

Require consistent session settings at many levels of creation and use

Require code changes for some predicates and to use in stored procs / sp_executesql

Maintenance costs lower as only DML that affects rows in index causes index changes

DTA can suggest filtered indexes, missing index DMVs do not

# Session Settings

**Session settings control behavior, and the result of some computations**

**Data in these persisted structures must be consistent**

**Session settings that must be on:**

- ANSI_NULLS
- ANSI_WARNING
- QUOTED_IDENTIFIER
- CONCAT_NULL_YIELDS_NULL
- ANSI_PADDING
- ARITHABORT

**Session setting that must be off:**

- NUMERIC_ROUNDABORT

**Msg 1934, Level 16, State 1, Line 1**

CREATE INDEX failed because the following SET options have incorrect settings: 'QUOTED_IDENTIFIER'. Verify that SET options are correct for use with indexed views and/or indexes on computed columns and/or filtered indexes and/or query notifications and/or XML data type methods and/or spatial index operations.

# Client Consistency

**Consistency at creation**

Table and index must be created with correct session options

**Consistency with table changes**

All DML statements must use correct session settings

**Consistency with query access**

All SELECT statements must use correct session settings

# Use Case: Imagine “Types” of Data

**For example, insurance types, document types, customer types**

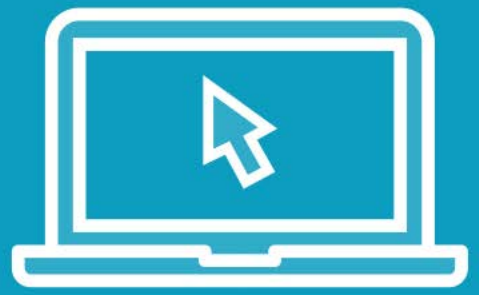
**Imagine there are 9 “statuses” of employee records (distribution can be even or NOT – doesn’t matter)**

- When you look at status = 1, you’re only interested in c6, c8, c4, c7
- When you look at status = 2, you’re only interested in c12, c16, c14, c6
- When you look at status = 3, you’re only interested in c2, c3, c4, c6

**FILTER each index on status with ONLY relevant included columns for a much smaller index size (matching only number of rows for filter)**

**Each index is only $^{1}/_{n}$ of the table; even with n specialized indexes storage is only what one unfiltered index would have been!**

# Demo



## Filtered indexes: EmployeeCaseStudy

# Cover More!

They're small so consider adding more columns to support additional queries!

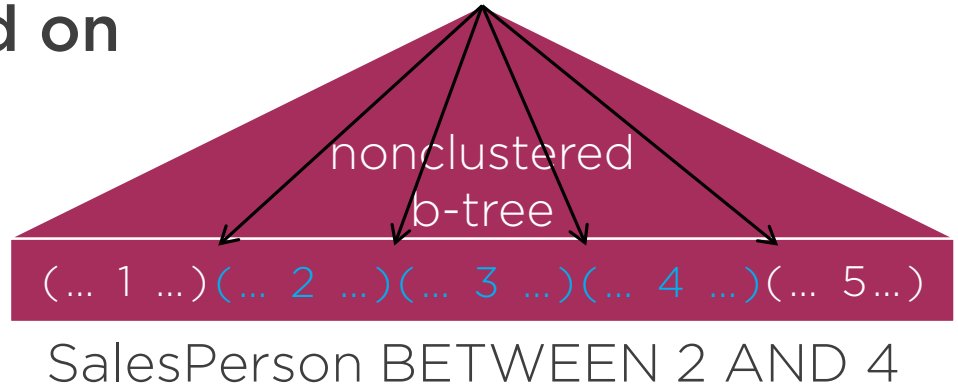
More effective for covering when sets are clearly defined

If equality-based sets consider NOT including that column (unless query needs it)

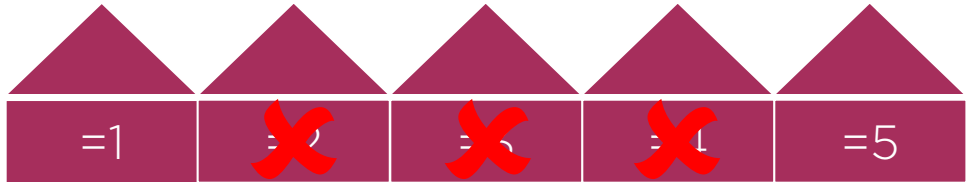
If NOT equality-based sets consider including column as part of key for effective seeking

# Why Not One Per Set Always?

**Leaf level of a regular nonclustered on (SalesPerson) INCLUDE (c6, c8)**




**Leaf level of individual filtered indexes by SalesPerson INCLUDE (C6, c8)**

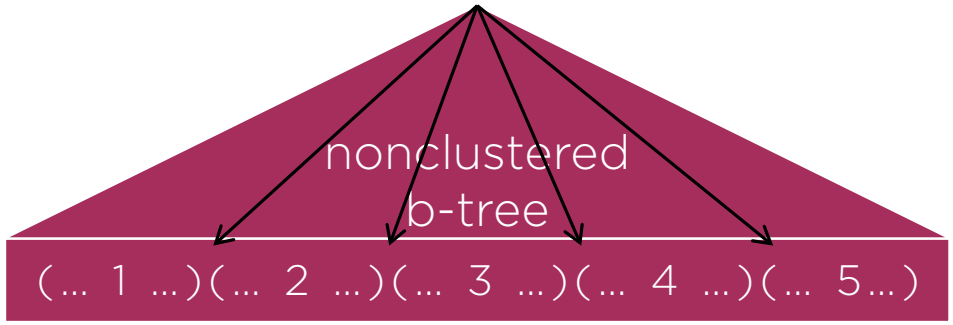



**Size difference negligible (but don't need to include filter column)**

- Predicate uses are MORE limited
- Covering choices are WAY MORE flexible

# When Is One Per Set Acceptable?

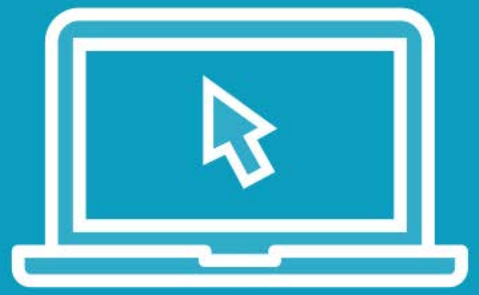
**Leaf level of a regular nonclustered on (SalesPerson) INCLUDE (c6, c8) requires same scolumns for all salespeople**




**What if you want different columns per set?**

- **SalesPerson = 1 (c16, c22)**
- **SalesPerson = 2 (c2, c8)**
- **Etc...**

# Demo



## Filtered indexes and interval subsumption

# Plan Caching

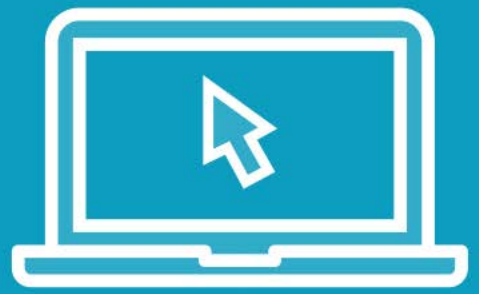
Stored procedures and sp_executesql statements compile and save an optimized plan for execution in the cache

- Not a permanent structure
- Plan is based on parameters used at time of plan caching (when there isn't already a plan in cache for that object)

Depending on how the criteria needing filtered index is specified, may or may not be able to leverage a filtered index without changing code

- Literals CAN use a filtered index
- Parameters CANNOT use filtered index unless OPTION (RECOMPILE) specified
- Variables CANNOT use filtered index unless OPTION (RECOMPILE) specified

# Demo



## Filtered indexes and plan caching

# What Filter?

sys.indexes

**Has two interesting columns:**

- has_filter: if index has a filter predicate
- filter_definition: expression for filter definition

# What Filter?

sp_helpindex

**Doesn't show included columns or filtered indexes**

- Use my tweaked "sp_helpindex" to get better information and determine if one index really is redundant/duplicate:
- https://www.sqlskills.com/BLOGS/KIMBERLY/category/sp_helpindex-rewrites

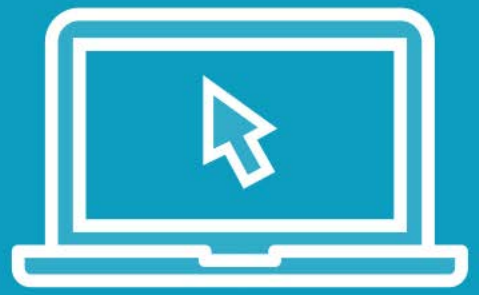
# What Filter?

DBCC
SHOW_STATISTICS

**Shows filter expression that defines subset over which statistics are computed**

- More accurate over the filtered set

# Demo



## Examining the filter predicate

# Maintenance

Statistics more accurate (when created) as they describe fewer rows

**More accurate after created or updated**

**Not auto-updated until threshold is met**

- Versions prior to SQL Server 2016, it's minimum of 500 + 20% of the TABLE
- Using trace flag 2371 OR in SQL Server 2016, it's "dynamic", happens much earlier than 20% for tables >25K rows

**Do not rely on "auto update statistics"**

**Better to have automated maintenance to keep statistics up-to-date**

# Maintenance

Index maintenance is required

**ALTER INDEX ... REBUILD/REORGANIZE work with filtered indexes**

**Online index operations work with filtered indexes**

Don’t go wild with filtered indexes – powerful but definitely specific uses!

Test, test, test!

# What We Covered

- Filtered indexes and requirements
- Filtered index use cases
- Filtered indexes and interval subsumption
- Filtered indexes and plan caching
- Filtered index strategies