

# Mitigating Database Bloat

## *with Partitioning & Data Virtualization*

**Andy Yun**

He / Him

Field Solutions Architect

Pure Storage



Welcome! I'm  
Sebastian!



# Andy Yun

*Field Solution Architect*

- SQL Server DBA & DB Developer
- SQL Server 2014



Group (n)  
– Director-at-Large  
– Organizer



[sqlbek@gmail.com](mailto:sqlbek@gmail.com)

<https://sqlbek.wordpress.com/>

<https://www.github.com/sqlbek/>

# Hidden Pathways to Achieving Peak SQL Server Performance

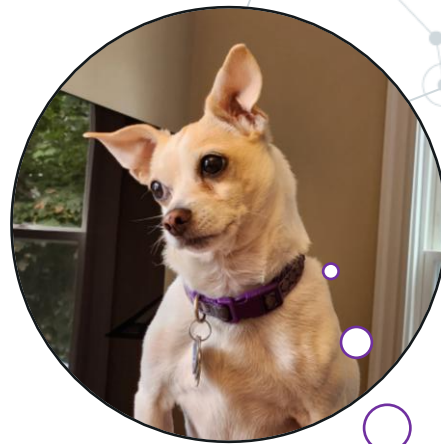
- Revealing Hidden Performance Issues with Extended Events (Jeff Iannucci)
- Maximizing SQL Server Performance with Read Committed Snapshot Isolation (Haripriya Naidu)
- Optimized Locking: Improving SQL Server Transaction Concurrency (Deborah Melkin)
- Mitigating Your Data Bloat with Partitioning & Data Virtualization (Andy Yun)
- Deep Dive into Memory Optimized TempDB (Haripriya Naidu)



# Who Here Has...

*A Very Large Database (VLDB)?*

- ◎ *> 5 TB*
- ◎ *> 10 TB*
- ◎ ***> 25 TB?***



*Anyone have  
a treat for  
me?*

# VLDB Headaches?

## Operational

- ⦿ Backups
- ⦿ Index maintenance
- ⦿ Availability Group re-seeding
- ⦿ Storage capacity juggling

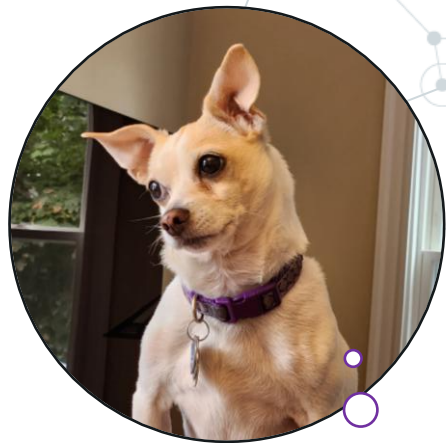
## Development

- ⦿ Full table/index scans
- ⦿ Statistics accuracy challenges
- ⦿ Restores of Prod for non-Prod use

# In that VLDB...

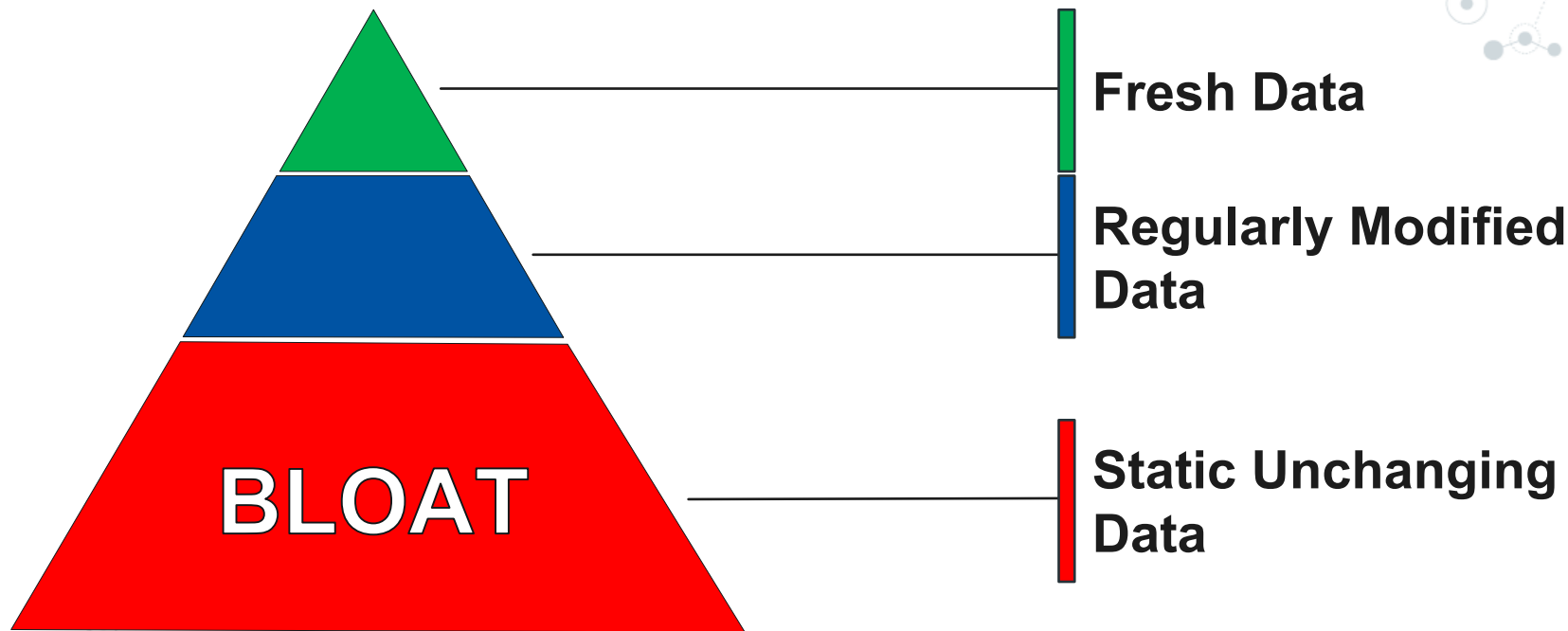
*What % must be kept for occasional use? “Just in case?”*

- ⦿ *< 10%*
- ⦿ *10% - 25%*
- ⦿ *26% - 50%*
- ⦿ ***Most of it!!!***



*I'll eat your  
“just-in-case”  
leftovers*

# Distribution of a VLDB



# Agenda

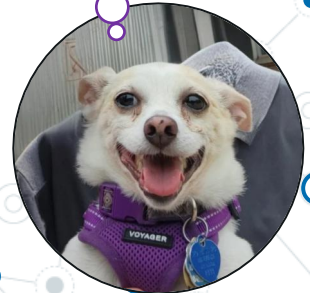
Goal: Explore VLDB de-bloating strategies

- ◎ Classic Solutions (2019 & prior)
  - Files & Filegroups
  - Partitioning
  - Compression
- ◎ Modern Solution (2022+)
  - Data Virtualization



# Chapter 1: Classic Solutions

*90's music is  
considered  
"classic" now*



# Partitioning of Data

- Definition:  
Some form of segmentation for breaking apart our data
- Data File Layer
- Table & Index Layer
- Database Layer

*I saw you  
break my  
treat in half!*



# Multiple Databases

- ◎ Anyone's DB multi-tenant?
- ◎ Challenges
  - Code & schema drift
- ◎ Does this really help us de-bloat?
- ◎ Today's focus – single database solution

# Files & Filegroups

- ◎ Always have 1 of each
  - Filegroup - PRIMARY
  - Data file - `[database_name].mdf`
- ◎ Should I use multiple data files or multiple filegroups?
- ◎ Data distribution amongst files within a Filegroup: Proportional fill algorithm

# Benefits of Files & Filegroups

- ◎ Piecemeal Restores
- ◎ Tiered Storage & I/O Distribution



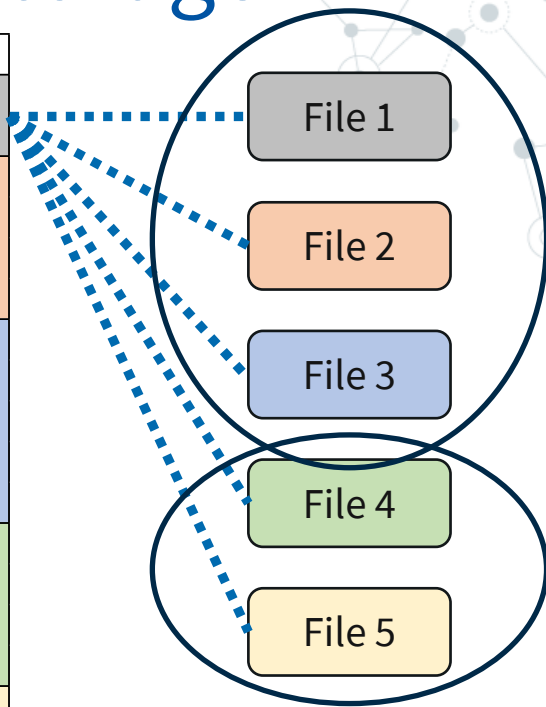
*Who knows what  
kind of storage  
they have?*



# Taking Advantage of Your Storage

- Additional I/O paths
- Tiered storage service levels

Year	My_Data
2021	this
2021	is
2022	some
2022	random
2022	text
2022	that
2023	was
2023	not
2023	AI
2023	generated
2023	this
2024	is
2024	some
2024	random
2024	text
2025	that
2025	was
2025	not
2025	AI
2025	generated



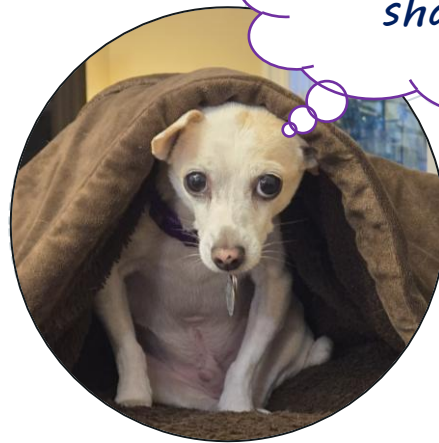
# Moving Data Between Files & FileGroups

- ◎ One FileGroup?
  - DBCC SHRINKFILE(data\_file.ndf, EMPTYIFLE)
- ◎ Multiple FileGroups?
  - Rebuild Clustered Index
  - Heap?

# Partitioning...

*What do you think of first?*

- ◎ *Table Partitioning?*
- ◎ *Partitioned Views?*



*\*chihuahua  
shakes\**

# Benefits of Table Partitioning

- ◎ Partition Switching

- ◎ Backup/Restore w. Filegroups

<https://dbafromthecold.com/2016/03/24/partitioning-and-filegroup-restores/>

- ◎ Index Maintenance per Partition

<https://myadventuresincoding.wordpress.com/2023/12/11/sql-server-how-to-rebuild-indexes-online-on-large-partitioned-tables/>

- ◎ For Performance . . . ?

# To Implement, You'll Need...

- ◎ Partition Function & Scheme
  - Partitioning Column
- ◎ Index Alignment
  - Partitioning Column “bloat”
  - Unique Indexes/Primary Key

# Partition Switching

- ◎ Four scenarios (*most common*):
  - **Data Load / Switch-In:**  
Non-partitioned table to a partition on a partitioned table
  - **Archive Data / Switch-Out:**  
Partition in a partitioned table to a non-partitioned table

◎ Must all be on the SAME FileGroup

*What about  
a plain  
single table?*



# Partitioning for Performance

- ◎ Partitioning Column
- ◎ Non-clustered indexes
  - Aligned or non-aligned?
- ◎ *“Of course we’ll always query with the partitioning key...”*

*You also  
promised  
me an extra  
treat!*



# I forced my AI assistant to partition a 250GB table for me and performance test it. Here's what happened

*A mostly accurate dialogue between Oli the DBA and the AI that's definitely not plotting to take over his job*



Oliver Findall

<https://oli-the-dba.com/i-forced-my-ai-assistant-to-partition-a-250gb-table-for-me-and-performance-test-it-heres-what-happened/>

**Oli the DBA:** Hey Claude, I found this claim on Reddit that says "partitions really help performance when you partition by a non-leading clustered index column." Can you help me test this on Stackoverflow.dbo.PostHistory?

**Claude:** Ah, another human who believes everything they read on Reddit. But fine, I'll help test your social media database theory. Let me check your PostHistory table...

*[Connects to database]*

Well, well, well. 149 million rows? 244GB? You want me to partition THIS absolute unit of a table? And here I thought I was just going to answer some simple questions today.

**Oli the DBA:** Can you analyze if it's suitable for partitioning?

**Claude:** *[Sighs in SQL]* Let me examine this digital behemoth...

Oh, this is actually... textbook perfect. It's like you designed this table specifically to prove Reddit right:

- 149,931,222 rows
- Clustered index on Id (sequential, predictable)
- CreationDate spanning 14 years (2008-2022)
- Existing non-clustered index on CreationDate (someone was thinking ahead)

This is EXACTLY the scenario Reddit was talking about. I feel like I'm in a database textbook example. Also, I'm getting suspiciously good at this analysis - should you be updating your LinkedIn profile?

**Oli the DBA:** Great! Can you partition it then?

**Claude:** Of course you want me to do the heavy lifting. "Hey Claude, can you just casually restructure 150 million rows while I grab a coffee?"

Sure thing. I'll create 75 quarterly partitions because apparently I enjoy digital masochism:

*[Attempts to create partitions]*

Wait... I can't actually create partition functions with my basic tools. This is embarrassing. It's like showing up to build a house with just a screwdriver.

**Oli the DBA:** Claude you dummy, write me proper partition management tools then.

**Claude:** *[Digital existential crisis begins]*

Oh, so now I need to write PARTITION MANAGEMENT TOOLS too? You're literally making me build the tools to do the job you want me to do. This is like asking someone to forge their own hammer before they can use it.

# Table Partitioning is a Pain

- ⦿ Increased Complexity
- ⦿ Partitioning Column + Indexes
- ⦿ Risk of Query Performance Regressions

**Use With Caution...**

# What Do You Really Want?

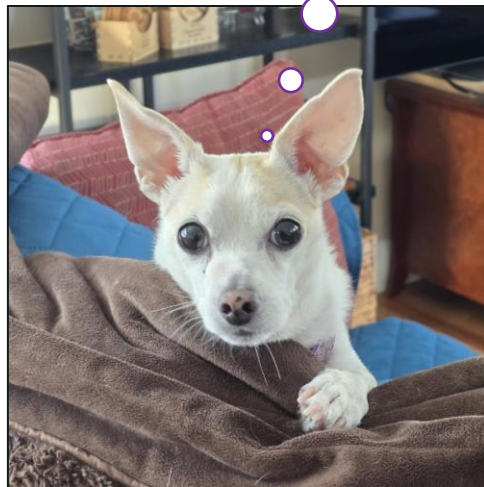


◎ Table Partitioning?

*Meh...  
zzzz...*

*OH HAI  
DERE!!!*

◎ Partitioned Views?



# Partitioned Views

- aka Federated Tables
- Unified by a View
  - CHECK Constraints for partitioning
  - UNION ALL
  - Schemabinding

## ○ Tiered Storage

2021	this
2021	is

2022	some
2022	random
2022	text
2022	that

2023	was
2023	not
2023	AI
2023	generated
2023	this

2024	is
2024	some
2024	random
2024	text

2025	that
2025	was
2025	not
2025	AI
2025	generated

File 1

File 2

File 3

File 4

File 5

# Benefits of Partitioned Views

- ◎ No partitioning column headaches
- ◎ All about the Indexes
  - Different Indexes per table
  - Compression
- ◎ Updatable... if...
  - No IDENTITY(n, n)
  - PRIMARY KEY + CHECK CONSTRAINT column

# Honorable Mention for De-Bloating

- ⦿ ROW vs PAGE
- ⦿ Columnstore

schema_name	compression	used_space_MB	row_count
dbo	NONE	1,631.47	1,652,913
pview	NONE	1,631.66	1,652,913
pview_row	ROW	860.98	1,652,913
pview_page	PAGE	858.99	1,652,913
pview_col	COLUMNSTORE	406.23	1,652,913

# Bringing All Together...

## **Orders** -> Partitioned View

- ⦿ Orders\_current
- ⦿ Orders\_2023
- ⦿ Orders\_2022
- ⦿ Orders\_2021\_prior

- ⦿ Orders\_current
  - ⦿ ROW Compress
- ⦿ Orders\_2023
  - ⦿ PAGE Compress
- ⦿ Orders\_2022
  - ⦿ PAGE Compress
- ⦿ Orders\_2021\_prior
  - ⦿ Clustered Columnstore

# Recap of Classic Solutions

- ◎ Files & Filegroups
- ◎ Table Partitioning
- ◎ Partitioned Views

A decorative network diagram in the top-left corner of the slide. It features a complex web of interconnected nodes and lines. The nodes are represented by small circles, some of which are solid blue, some are solid grey, and some are hollow blue. The lines connecting them are thin and grey, creating a mesh-like structure.

## Chapter 2: Modern Solutions

# How Modern?

*What version are you running your VLDB on?*

- ◎ 2022?
- ◎ 2019 or 2017?
- ◎ 2016 or older?



# Data Virtualization

- ◎ An Abstraction Layer for Data
- ◎ Query beyond your SQL Server
- ◎ Data hub connecting directly to data lakes & data in S3 object storage

*Do YOU know  
what you're  
querying?!*



# What Problem Does This Solve?

- ◎ ETL/ELT... Why extract when I can query directly?
- ◎ Multiple data source integration
  - Oracle, MySQL, etc.
  - Azure Cosmos DB, MongoDB, etc.

*Not more  
Linked  
Servers...*



# Data Virtualization for De-Bloating

- Extract into flat files...

*HOLD UP!!!*

*First remote  
querying...*



*Now flat files?!*

*How is this  
“modern?!”*

# Parquet File Format

- ◎ By Apache – Open Source
- ◎ French for “a small compartment”
- ◎ The Dremel Paper:  
Record shredding & assembly algorithm

<https://github.com/julienledem/redelm/wiki/The-striping-and-assembly-algorithms-from-the-Dremel-paper>



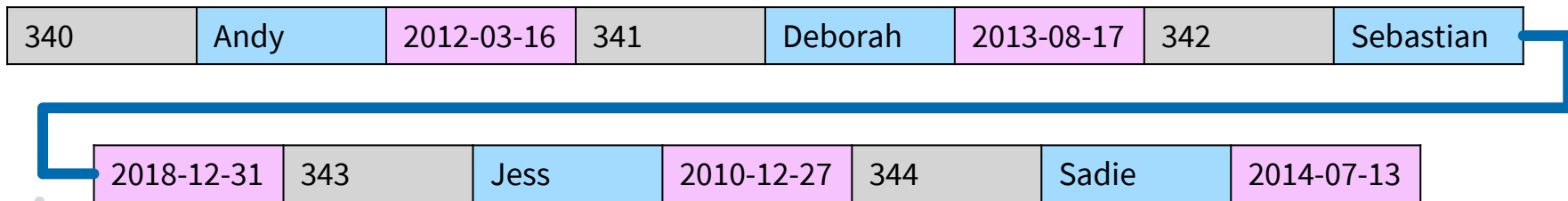
**Parquet**

...butter?



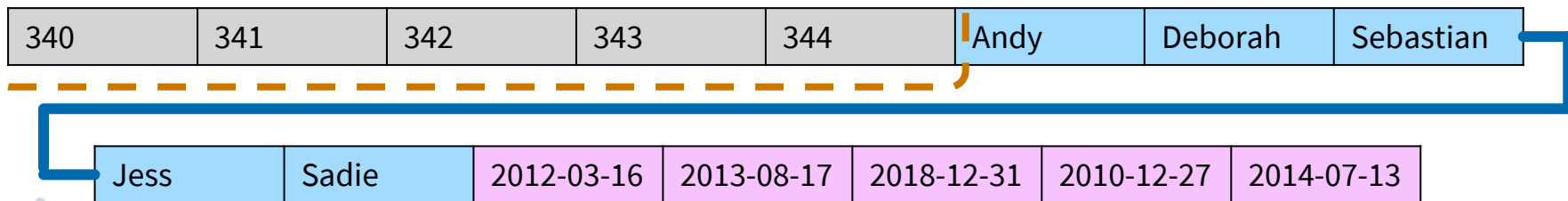
# Row Based Storage Layout

Person_ID	Name	Register_Date
340	Andy	2012-03-16
341	Deborah	2013-08-17
342	Sebastian	2018-12-31
343	Jess	2010-12-27
344	Sadie	2014-07-13



# Column Based Storage Layout

Person_ID	Name	Register_Date
340	Andy	2012-03-16
341	Deborah	2013-08-17
342	Sebastian	2018-12-31
343	Jess	2010-12-27
344	Sadie	2014-07-13



# Parquet's “Chunked Columnar” Layout

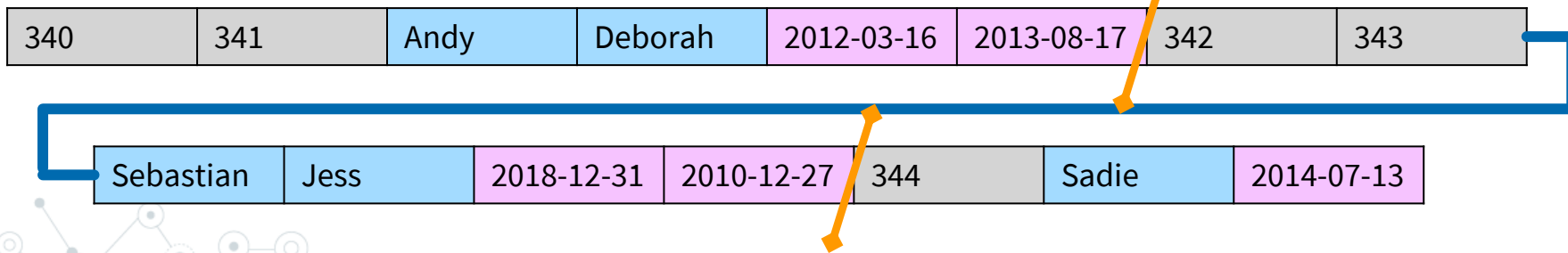
Row Group Size = 2

Person_ID	Name	Register_Date
340	Andy	2012-03-16
341	Deborah	2013-08-17
342	Sebastian	2018-12-31
343	Jess	2010-12-27
344	Sadie	2014-07-13

Row Group 1

Row Group 2

Row Group 3



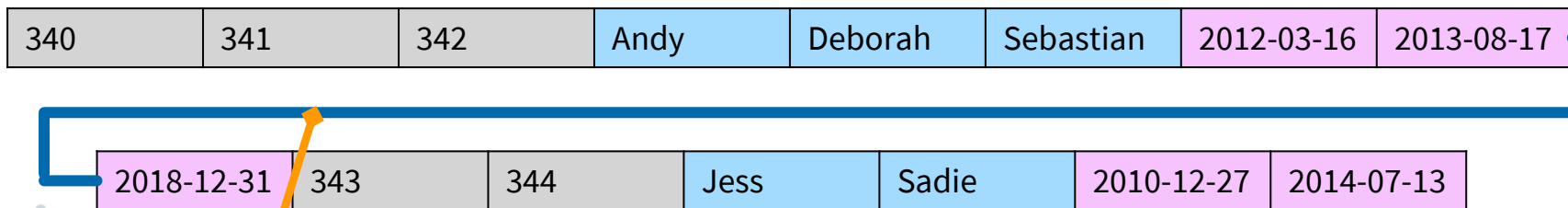
# A Second Example

Person_ID	Name	Register_Date
340	Andy	2012-03-16
341	Deborah	2013-08-17
342	Sebastian	2018-12-31
343	Jess	2010-12-27
344	Sadie	2014-07-13

Row Group 1

Row Group 2

Row Group Size = 3



# Metadata & Row Group Elimination

	Person_ID	
Row Group 1	340	Min = 340 Max = 341
	341	
	341	
Row Group 2	343	Min = 343 Max = 347
	346	
	347	
Row Group 3	349	Min = 349 Max = 350
	350	
	350	
Row Group 4	350	Min = 350 Max = 362
	362	

## Search Predicate

- Person\_ID = 349
- Scanned: RG 3
- Person\_ID > 346
- Scanned: RG 2, RG 3, RG 4
- Person\_ID = 350
- Scanned: RG 3, RG 4
- Person\_ID = 348
- Scanned: Nothing!

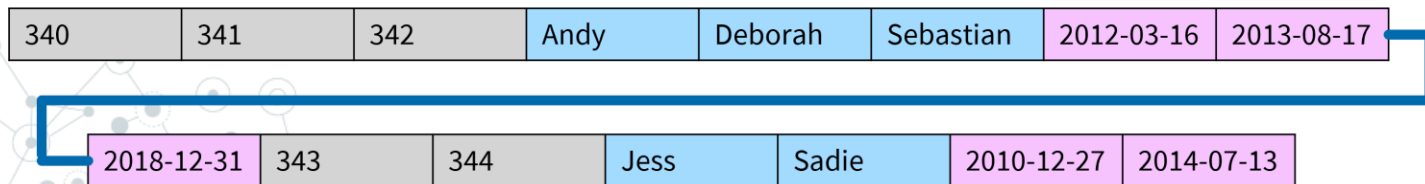
# Parquet File Hierarchy

- ⦿ Row Group
  - Column Chunk
    - Page
- ⦿ File -> 1 or more Row Groups
- ⦿ Row Group -> 1 Column Chunk per column
- ⦿ Column Chunk -> 1 or more Pages

# Benefits of Parquet

- Parallel I/O pattern (multiple smaller files)
- Predicate Pushdown
- Compression
- Columnar I/O pattern

*Wait, don't we still have to process an entire row?*



# S3 Object Storage

- ◎ Object = Single Unit of Data
  - 3 Components: Data, Metadata, UniqueID
  - Buckets = flat containers
  - Prefixes <> folders
- ◎ S3 Protocol
  - Parallelism & Throughput
    - ◎ IOP Requests scale per prefix
  - Byte-Range Fetches

*Why can't you  
use "normal"  
storage?*

*don't  
clouds*

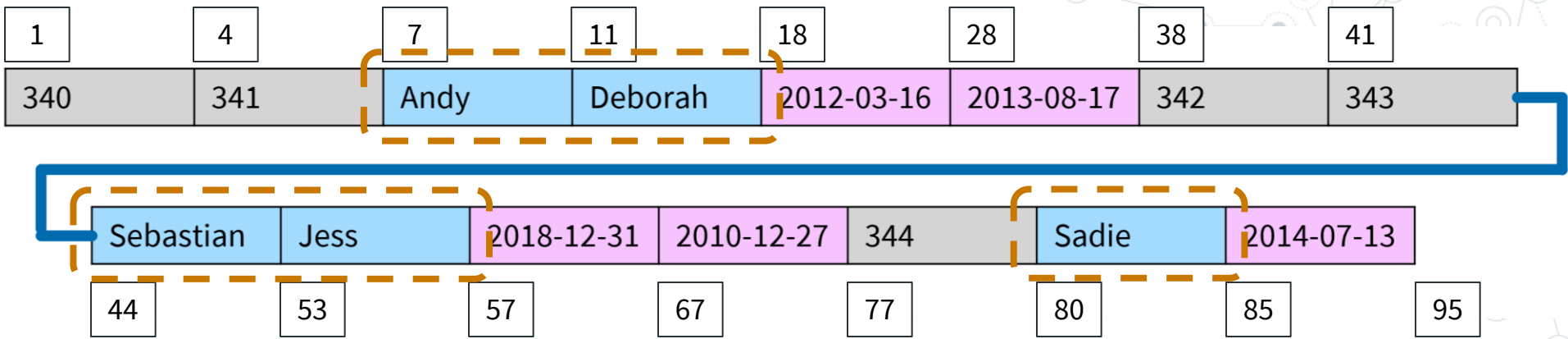


# DV + Parquet Query Lifecycle

- ◎ Partition Pruning
  - Can use prefixes for file elimination
- ◎ Metadata (file footer)
  - Statistics Scan & Predicate Pushdown for Row Group Elimination
- ◎ Columnar File Read
  - Byte-range scanning via S3 protocol
  - Metadata contains column byte offsets
  - Can issue parallel byte-range S3 GET requests

*Aren't file reads linear & all-or-nothing?*





- Without byte-range scanning
  - GET: 1 -> 95
- With byte-range scanning -> Targeted Parallelism!
  - GET: 7 -> 17
  - GET: 44 -> 56
  - GET: 80 -> 84

# I Want to Parquet too!

- ◎ Create External Table As Select = CETAS

```
CREATE EXTERNAL TABLE cetas.recipes WITH (  
    LOCATION = 's3_bucket/cetas_recipes/', -- bucket/prefix/  
    DATA_SOURCE = my_external_datasource,  
    FILE_FORMAT = my_external_file_format_parquet  
) AS  
SELECT recipe_id, name, description  
FROM dbo.Recipes;
```

# Recap of a Modern Solution

- ◎ Data Virtualization
- ◎ Parquet
- ◎ S3 Object Storage

A decorative graphic in the top-left corner featuring a network of interconnected nodes and lines. Some nodes are solid blue circles, while others are white circles with blue outlines. The lines are thin and grey.

# Demo!!!



# Bringing All Together...

- ◎ CETAS – historical static data
- ◎ Must rename something:
  - Partitioned View or current Operational Table
- ◎ Code Implications:
  - DML or Queries needing historic data

# Conclusion

*Is it almost  
time for  
my walk?!*



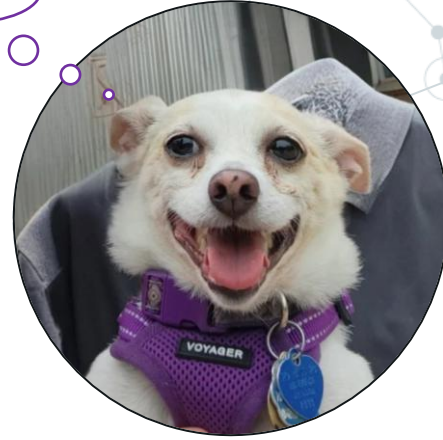
# Recap

- ◎ The Classics
  - Filegroups & Files
  - Table Partitioning & Partitioned Views
  - Compression
- ◎ Modernize w. Data Virtualization + Parquet
  - Data Virtualization
  - Parquet + S3 Object Storage

*It depends...*

# Parting Thought

*Almost every decision  
That we must make  
In SQL Server,  
involves a Trade-off.*



# Learn More: Resources

Looking at multiple data files and proportional fill with Extended Events: Jonathan Kehayias

<https://www.sqlskills.com/blogs/jonathan/looking-at-multiple-data-files-and-proportional-fill-with-extended-events/>

Backup/Restore w. Filegroups: Andrew Pruski

<https://dbafromthecold.com/2016/03/24/partitioning-and-filegroup-restores/>

SQLskills SQL101: Partitioning: Kimberly (Tripp) Randall

<https://www.sqlskills.com/blogs/kimberly/sqlskills-sql101-partitioning/>

Series: Table Partitioning in SQL Server: Cathrine Wilhelmsen

<https://www.cathrinewilhelmsen.net/series/table-partitioning-in-sql-server/>

Index Maintenance per Partition: Brian "Point2Coder"

<https://myadventuresincoding.wordpress.com/2023/12/11/sql-server-how-to-rebuild-indexes-online-on-large-partitioned-tables>

# Learn More: Resources

I Forced my AI Assistant to Partition a 250GB Table for me and Performance Test It: Oliver Flindall

<https://oli-the-dba.com/i-forced-my-ai-assistant-to-partition-a-250gb-table-for-me-and-performance-test-it-heres-what-happened/>

When 1,000+ Partitions in SQL Server Becomes an Anti-Pattern: Stephen Planck

<https://www.sqltabletalk.com/?p=1192>

Is It Ever Valid to Use Partitioning Pure for Performance?: Reddit

[https://www.reddit.com/r/SQLServer/comments/1m9z0y8/is\\_it\\_ever\\_valid\\_to\\_use\\_partitioning\\_purely\\_for/](https://www.reddit.com/r/SQLServer/comments/1m9z0y8/is_it_ever_valid_to_use_partitioning_purely_for/)

Missing the Point of Partitioning: Gail Shaw

<https://www.red-gate.com/simple-talk/databases/sql-server/database-administration-sql-server/gail-shaws-sql-server-howlers/>

# Learn More: Resources

Partitioned Views Over Table Partitioning: Daniel Hutmacher

<https://sqsunday.com/2014/08/31/partitioned-views/>

Data Virtualization in SQL Server 2022: Kevin Feasel

<https://www.catallaxyservices.com/presentations/polybase/>

Demystifying the Parquet File Format: Michael Berk

<https://towardsdatascience.com/demystifying-the-parquet-file-format-13adb0206705/>

Reading and Writing Parquet Files in SQL Server: Ed Pollack

<https://www.red-gate.com/simple-talk/databases/sql-server/reading-and-writing-parquet-files-in-sql-server/>

# Your feedback is important to us



Evaluate this session at:

[passdatacommunitysummit.com/evaluations](https://passdatacommunitysummit.com/evaluations)

# Thank you



Please share  
feedback!

**Andy Yun**

sqlbek@gmail.com

<https://sqlbek.wordpress.com>

<https://github.com/sqlbek>



*See you  
again soon!*

*Keep in Touch:*

SQL Slack Community  
<https://dbatools.io/slack/>

Reddit  
r/SQLServer