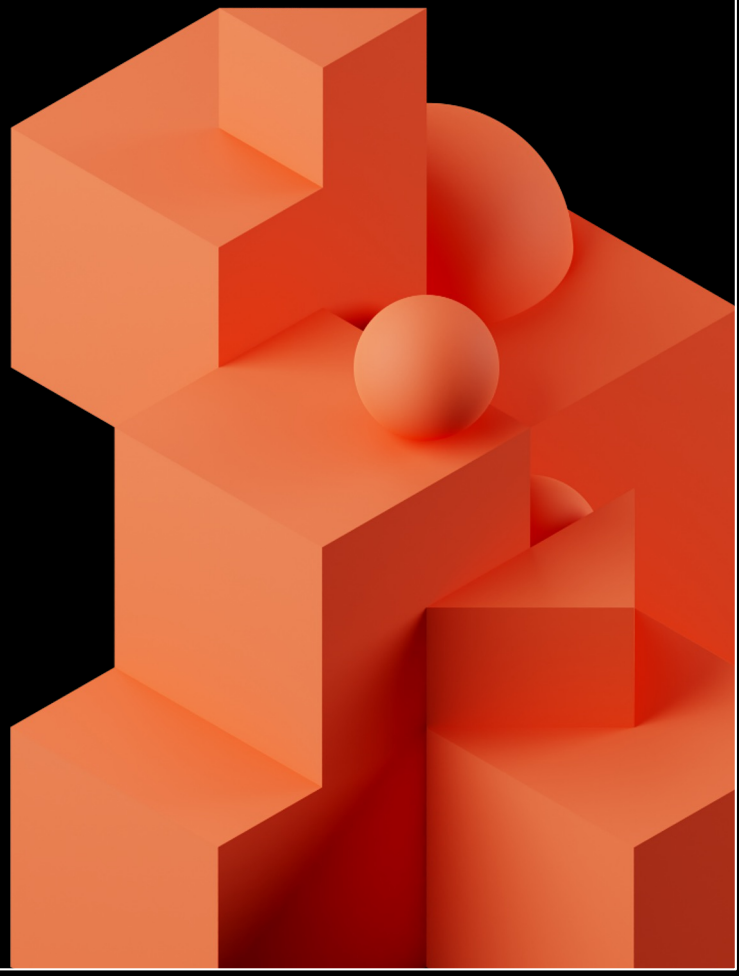
DATA+AI SUMMIT
BY databricks

Streaming Schema Drift Discovery, and Controlled Mitigation

Presented by: Alexander Vanadio



Databricks
2023

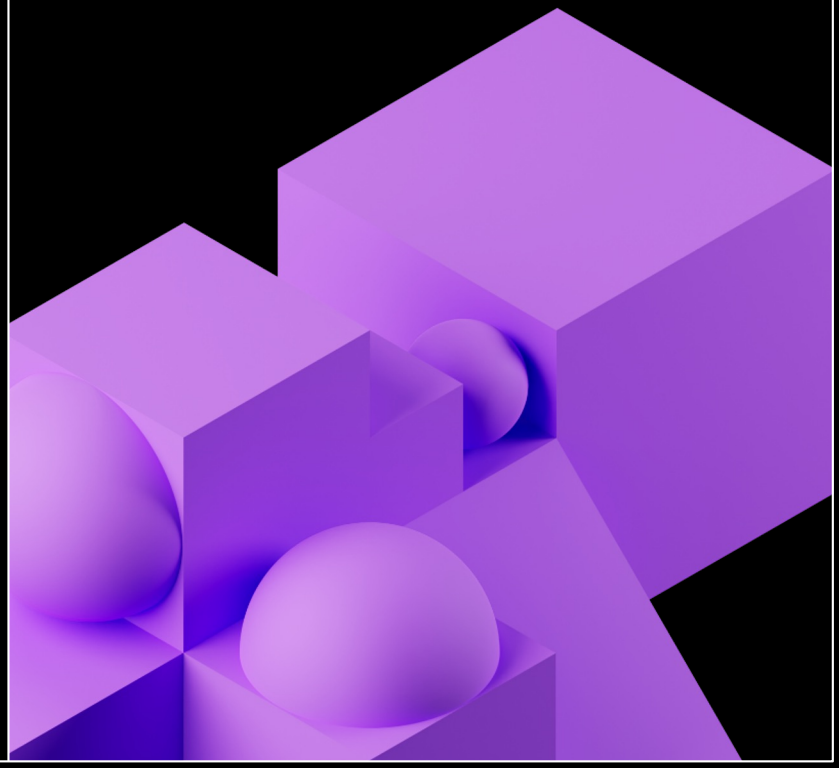
Prologue

About me

- I have been doing some combination of Software/Data/ML Engineering for about 15 years
- Consulted for a Fortune 100 company for a few years to help with their PB scale big data problems
- Big fan of computer science, music, and memes

“Alex, I have a time sensitive query that I need to run against [a Delta table], but don’t see some fields that should be there. What’s going on!?”

Schema Drift



Schema Drift

Moving the goals

- The data you ingested into your Delta Table today, might be different tomorrow
- Some keys might be so sparse that you didn’t even know they were there
- Vendors and upstream data providers usually aren’t going coordinate with you before they change something

Data Providers Are Ready

“Can’t we just set `.option(mergeSchema, True)` on our stream?”


MERGESCHEMA
TRUE TABLE
ALL YOUR
UNEXPECTED
COLUMNS
imgflip.com

Merge Schema

== True

- We’re putting a lot of trust in our data providers and removing human gates
- A table with 10 columns today could have 100 tomorrow
- This is especially bad when incoming data contains dynamically generated keys

“How about schema evolution within Auto Loader?”

Now we’re onto something

Auto Loader Schema Evolution

4 Unique Strategies

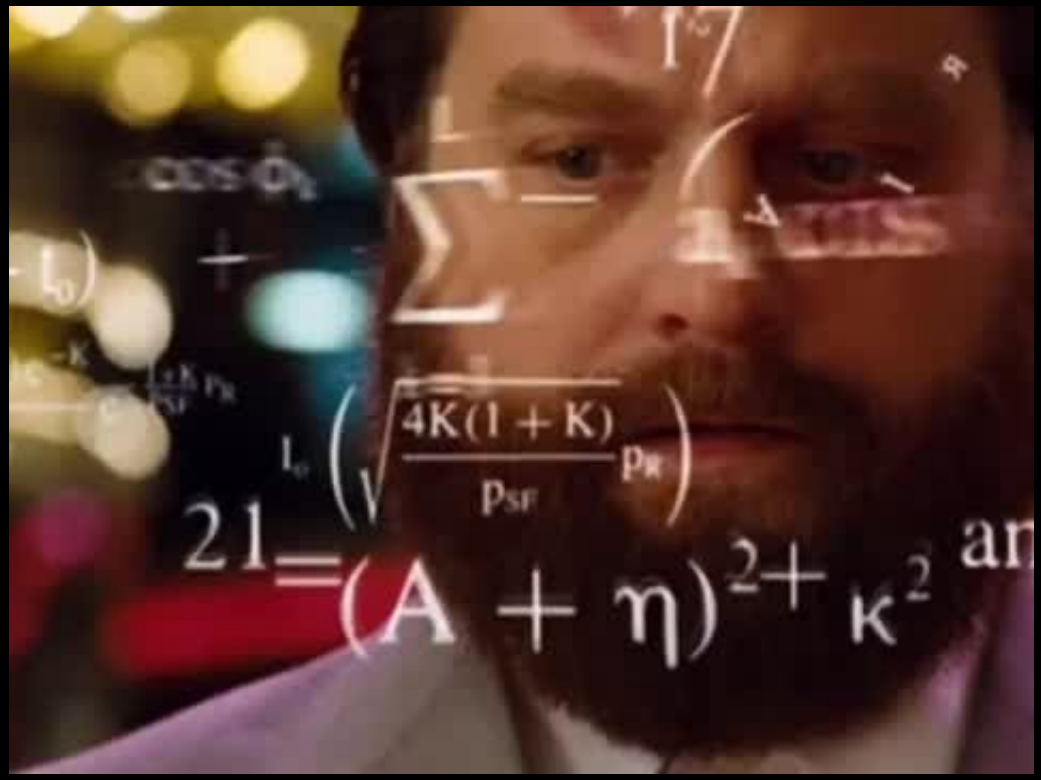
- What if we want to utilize the `badRecordsPath` option?
- What if we don't want to add ***all*** new columns to our table?
- What if we don't want to stop the stream (prioritize low-latency and availability)

Auto Loader Schema Evolution

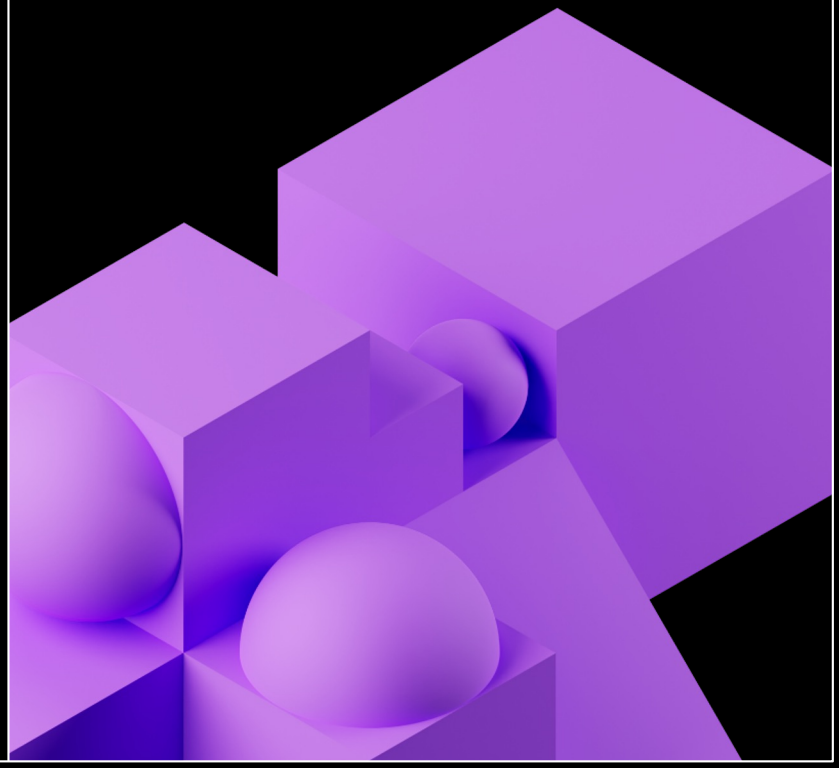
Selecting the `rescue` strategy

- Incoming data that doesn't match the internal schema file, ends up in a new String column called `_rescued_data`
- This is actually serialized JSON data (e.g. `'{"key": "value"}'`)

Can we use this for drift detection?...



Drift Detection



Quantifying Drift

Leveraging the `_rescued_data` column

- Every single record that comes in has well-formed JSON describing drift, but it's not useful as is
- What if we deserialized that data, transform it using a vectorized UDF, and write it to a new table?

Quantifying Drift – The Drift Table

Column	Type	Comment
eventDate	date	The date of the originating event (UTC)
window	struct	Drift window (aggregation)
window.start	timestamp	
window.end	timestamp	
tokens	array<string>	Dot delimited value captures. `abc.def` represents: `{"abc": { "def": "value" }}`
keys	array<string>	The **leaf** node captures AKA the inner most key. `key` could be `{"key": "value"}` or `{"nested": { "key": "value" }}`
avgNewKeysPerRecord	double	
maxNewKeysPerRecord	int	
dateSource	string	human readable upstream data source
tableName	string	originating Delta table
totalNewKeys	int	

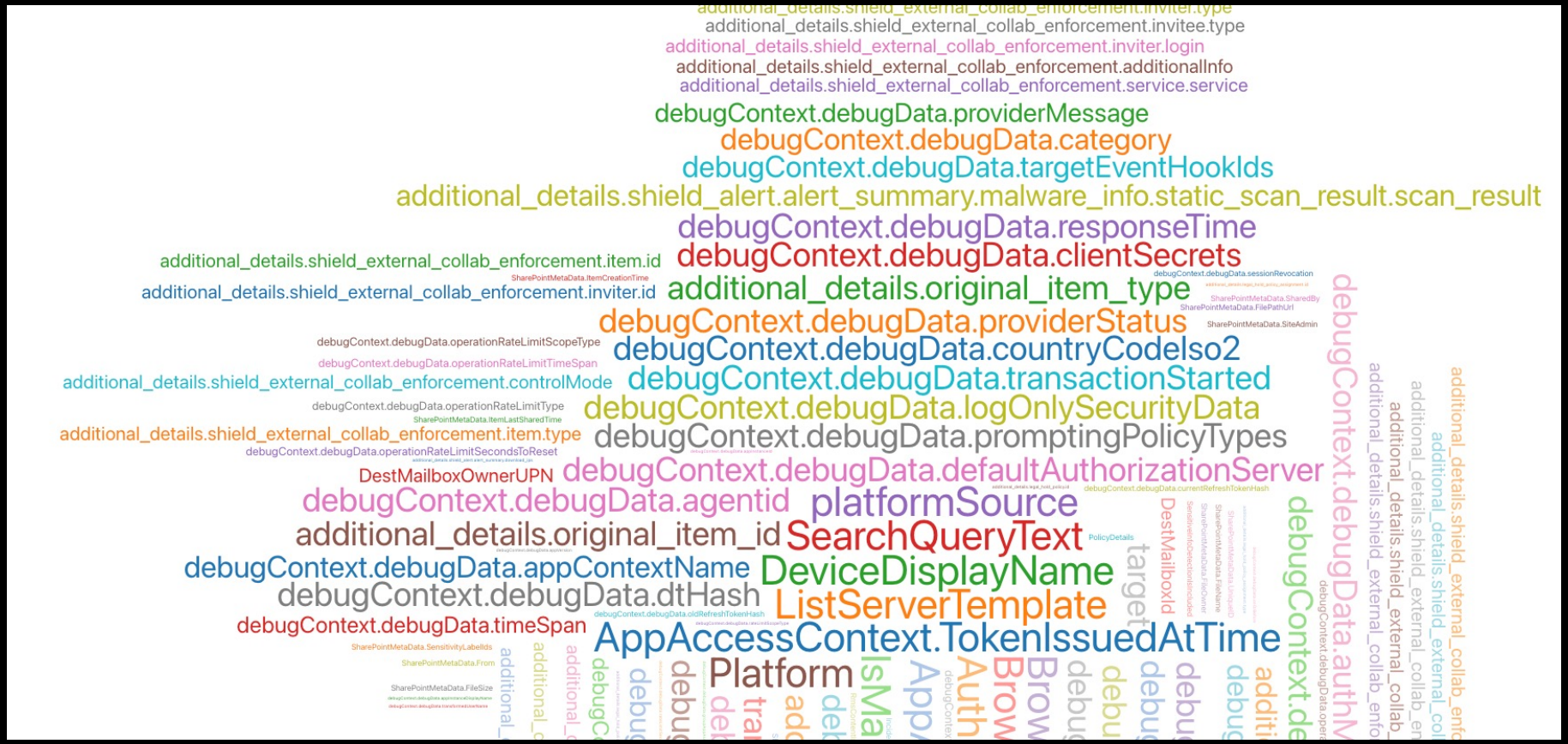
“Do I have to manually query that table? ☹”

Drift Detection - Dashboard

Visualizing Drift

- SQL Workspace is a pretty awesome place to make visualizations and dashboards
- Let's create some visualizations and add it to the pre-existing Ingest Dashboard
- It's possible to create alerts based on SQL queries

Drift – Word Cloud Across N Data Sources



Drift – Tabular

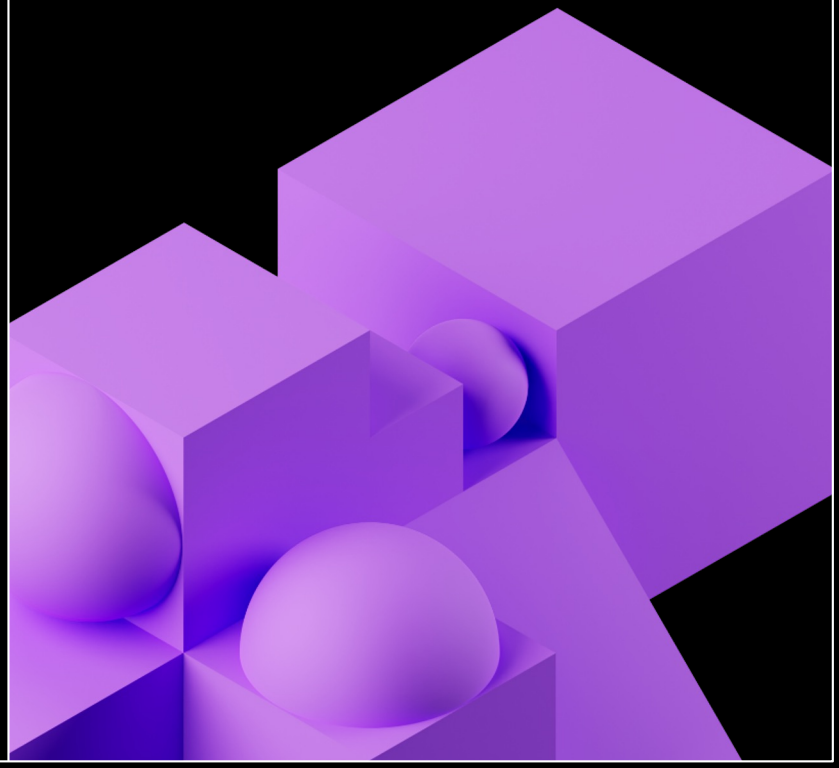
Results - Rescued Keys - Last X Hours

dataSource	tableName	token	windowCount	key
o365	cta.bronze_o365_sharepoint	BrowserVersion	12	BrowserVersion
o365	cta.bronze_o365_sharepoint	IsManagedDevice	12	IsManagedDevice
o365	cta.bronze_o365_sharepoint	platformSource	12	platformSource
o365	cta.bronze_o365_sharepoint	AppAccessContext.UniqueTokenId	12	UniqueTokenId
o365	cta.bronze_o365_sharepoint	DeviceDisplayName	12	DeviceDisplayName
o365	cta.bronze_o365_sharepoint	BrowserName	12	BrowserName
o365	cta.bronze_o365_sharepoint	AuthenticationType	12	AuthenticationType
o365	cta.bronze_o365_sharepoint	ListServerTemplate	12	ListServerTemplate
o365	cta.bronze_o365_sharepoint	AppAccessContext.TokenIssuedAtTime	12	TokenIssuedAtTime
o365	cta.bronze_o365_sharepoint	SearchQueryText	12	SearchQueryText
o365	cta.bronze_o365_sharepoint	Platform	12	Platform
o365	cta.bronze_o365_aad	platformSource	12	platformSource

Platform

a minute ago

Controlled Mitigation



Controlled Mitigation

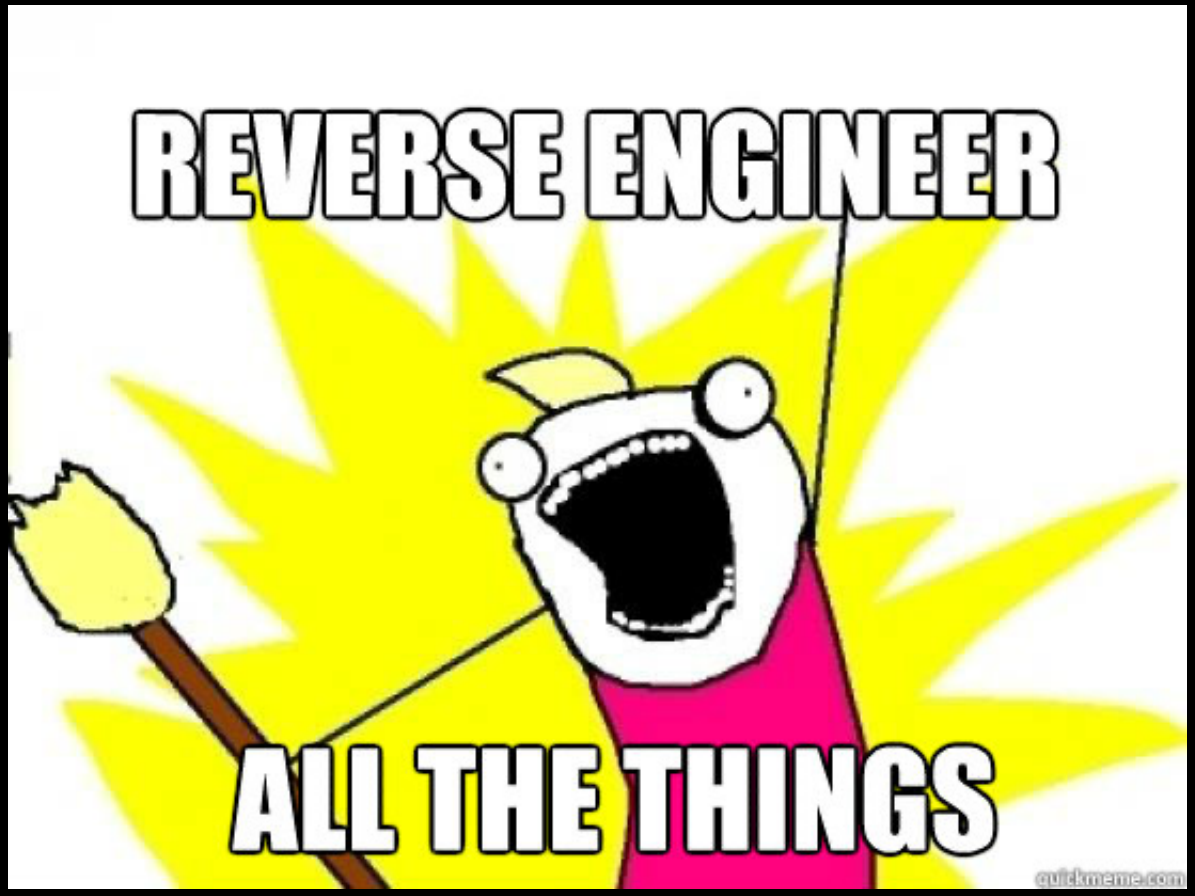
Selectively Promoting Rescued Keys

- We've been monitoring the drift discovery dashboard and think some of these keys are actually useful
- We also see some that definitely are not (e.g. `debugContext.*`)
- Now what?

Back to Auto Loader

Schema Management

- During schema inference a schema definition file is written, and then later referenced for all incoming data
- We can’t promote any rescued keys to our Delta table because this will always block us


REVERSE ENGINEER
ALL THE THINGS
quickmeme.com

Back to Auto Loader

Looking at the Schema File

- It turns out that the schema file is *mostly* a stringified `StructType`
- It should be possible to read this file, unmarshal the `StructType`, apply changes, then overwrite the file

Controlled Mitigation

Writing a Library to Modify The Schema File

- Let's write a simple python library with an api like `add_field(field: StructField)`
- This lets us represent arbitrarily complex data structures
- When we're done adding fields, the api should write all of the changes to the schema file

Controlled Mitigation

Operation Impact

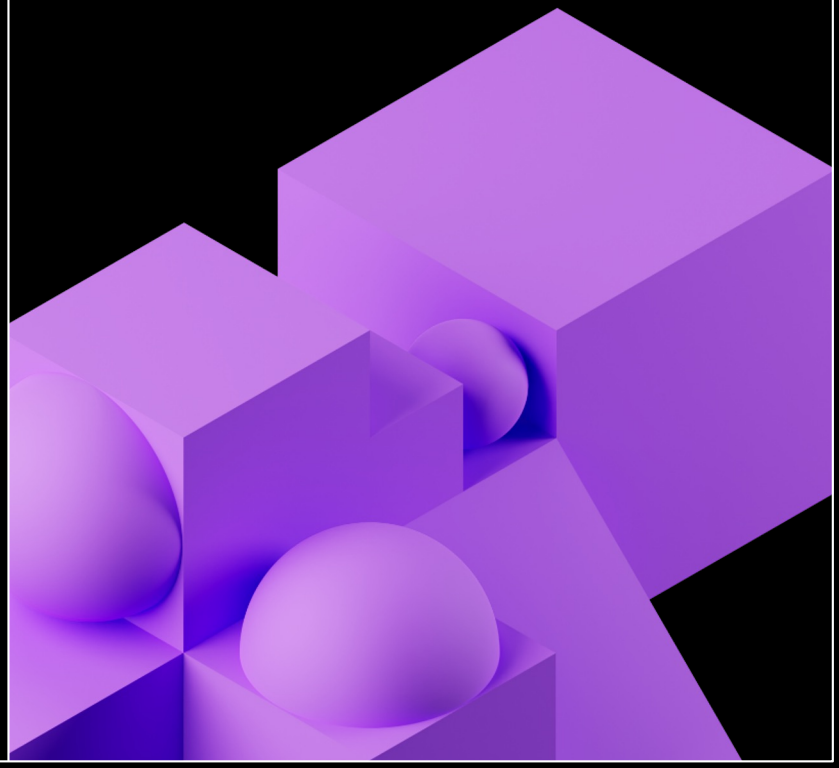
- The streaming job has been continually running
- When the evolution notebook is ready to go, then the streaming job shuts down
- The notebook takes a minute or so to run, then the stream is resumed

Merge Schema

== True (Again)

- We will be utilizing `mergeSchema` now that it only occurs on our terms
- Auto Loader schema evolution is the gatekeeper, and we only augment it when necessary

Conclusion




Only
Merge Schema
Auto Loader Rescue
+
Elbow Grease
+
Merge Schema
imgflip.com

Conclusion

Summary

- We can achieve schema drift detection across all Auto Loader streaming data sources that support rescue
- We can create visualizations, dashboards, and even alerts to help understand and assess drift

Conclusion

Summary Continued

- We can selectively decide what new columns to promote to our Delta sinks
- We can minimize streaming job downtime to a few minutes, during the evolution process

Conclusion

Limitations

- Specifically regarding modifying the schema file, Databricks can alter their internal process in a way that is not backwards compatible
- It’s wise to test this out for new DBR’s before jumping in

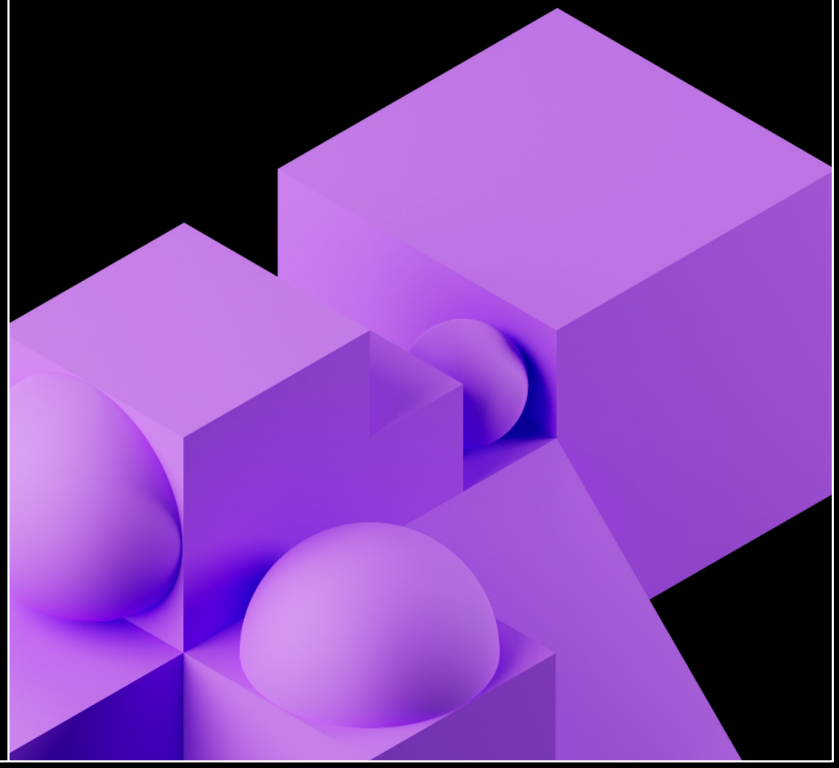
Conclusion

Blog

- All of this code, and a more detailed technical writeup, will be available in our upcoming Databricks blog!

Thanks!

You can reach out to avanadio@gmail.com



Q&A