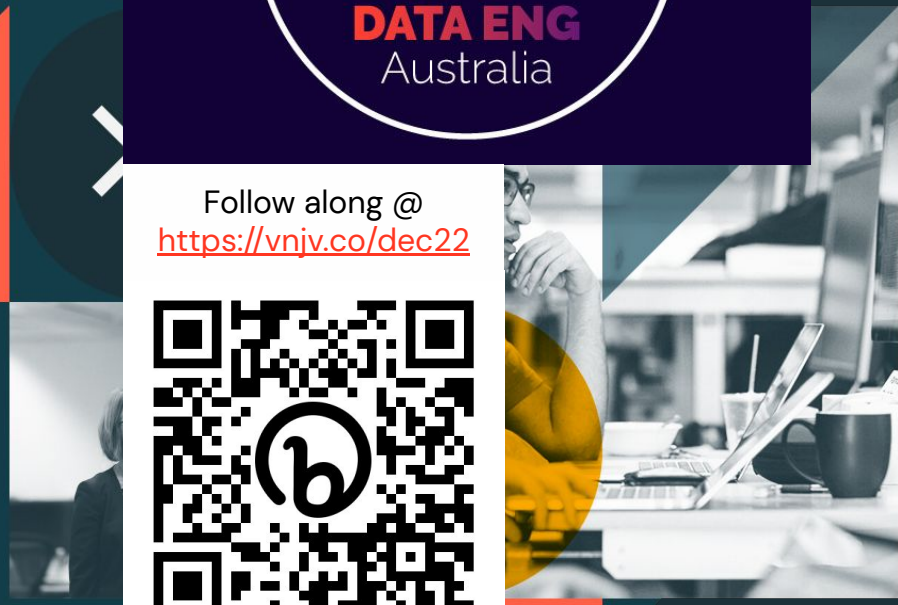
databricks

DELTA LAKE

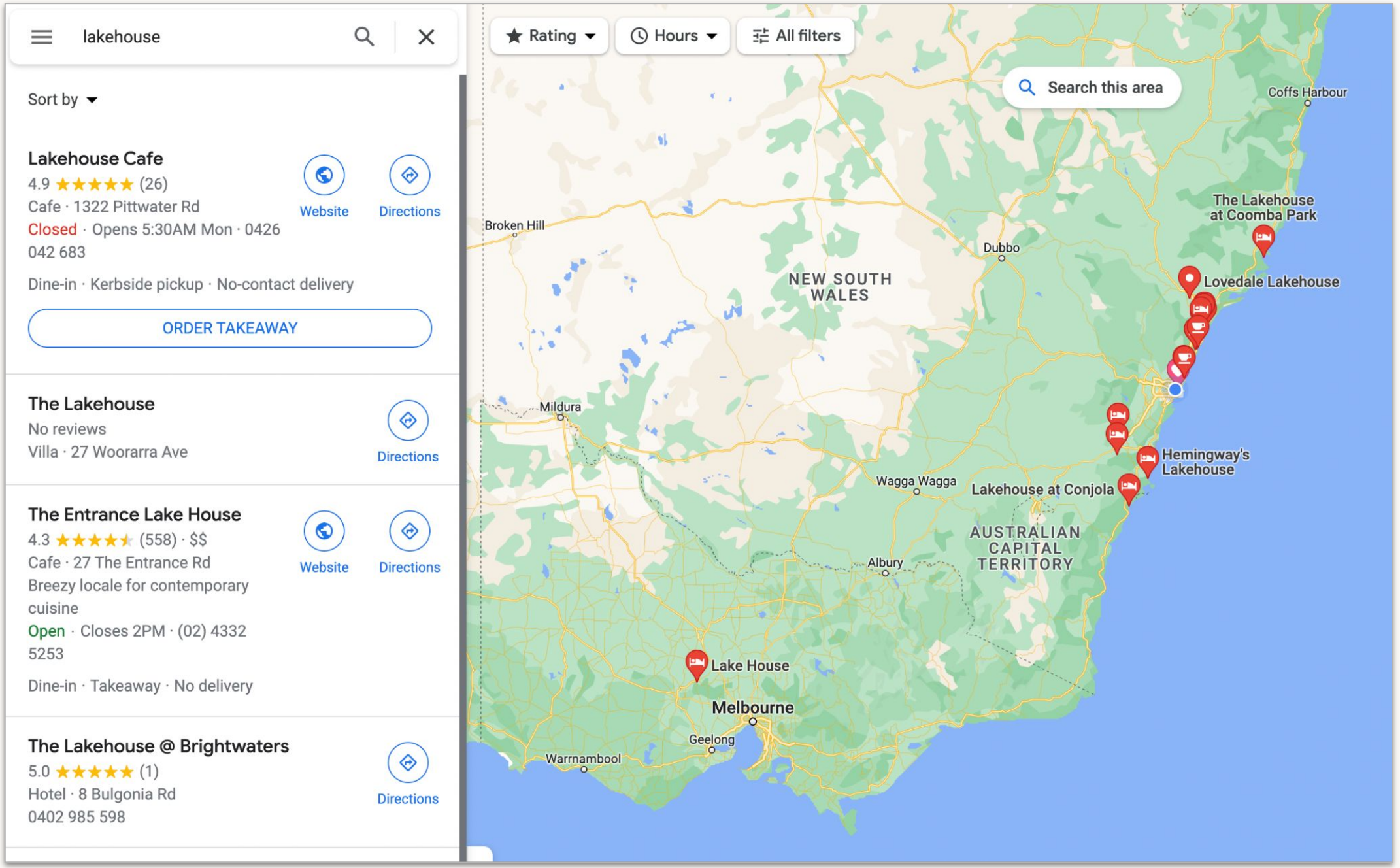
Delta Lake: Turn your Data Lake into a Lakehouse

DATA ENG Australia

Follow along @ https://vnjv.co/dec22



Vinoaj (Vinny) Vijeyakumaar
Senior Solutions Architect
vinny@databricks.com | /vinoaj



> whoami

Vinoaj (**Vinny**) Vijeyakumaar

Senior Solutions Architect

Databricks; Google Cloud; Google Analytics

17+ years in cloud & data space

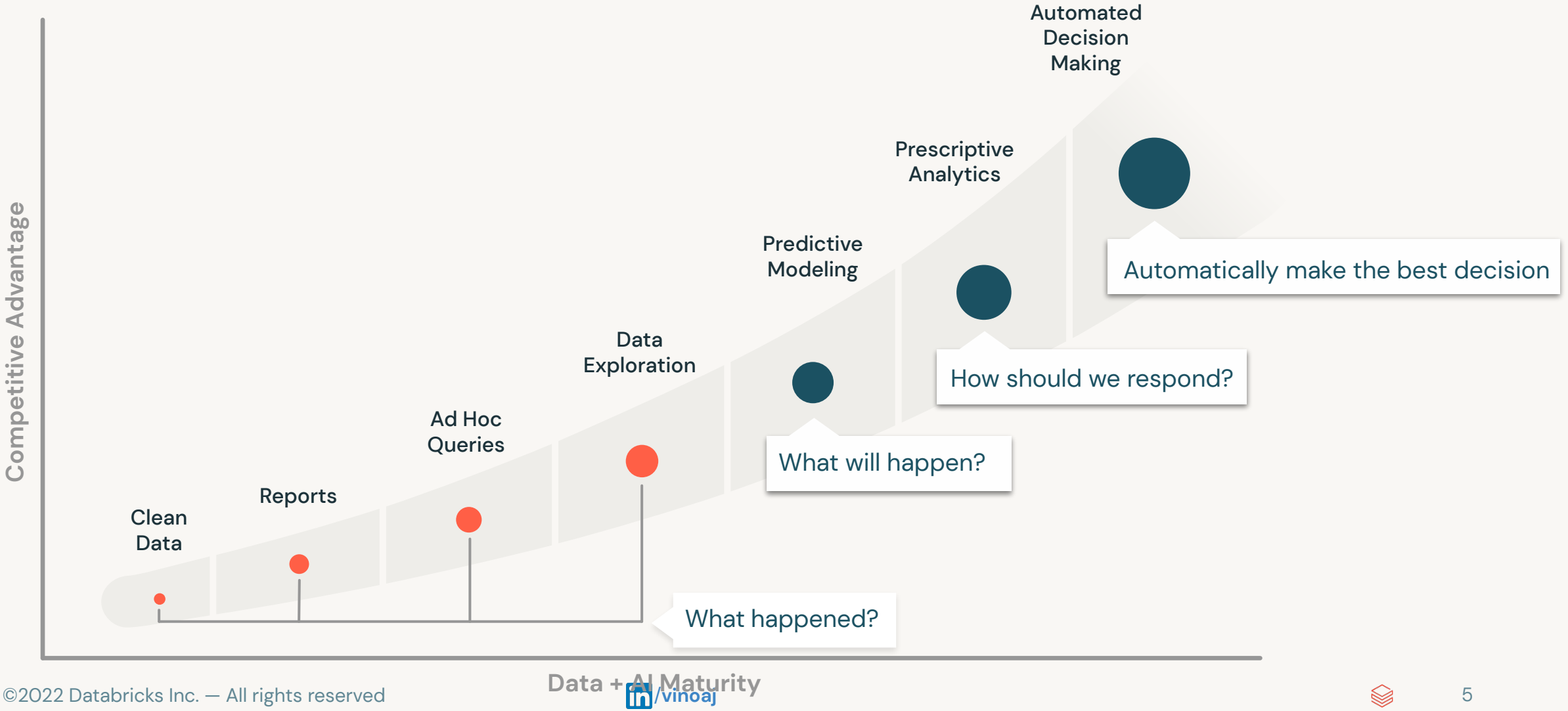
https://www.linkedin.com/in/vinoaj/

What has driven the need for Lakehouse?

Follow along @ https://vnjv.co/dec22

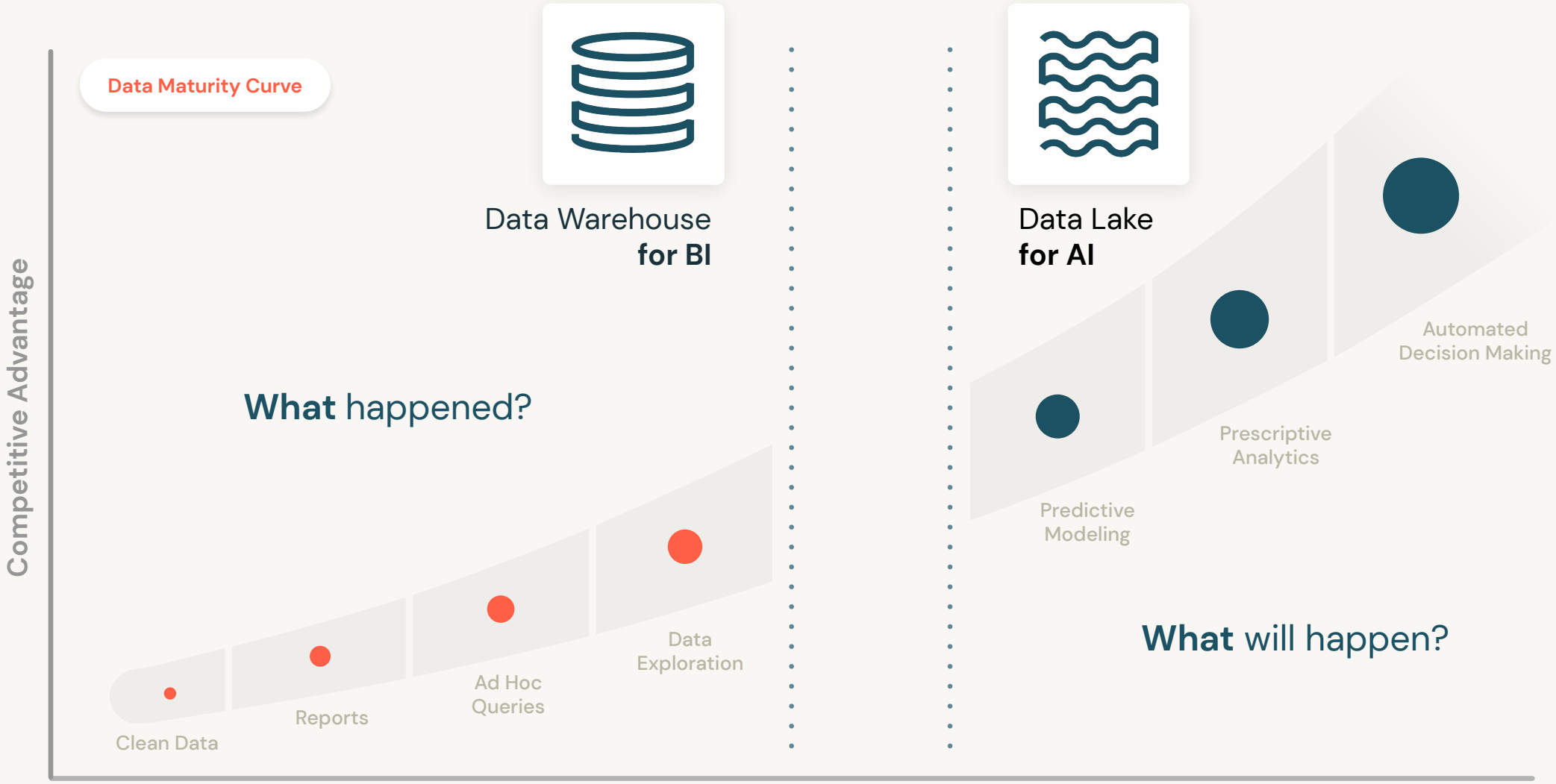
Tech leaders are to the right of the Data Maturity Curve

From hindsight to foresight

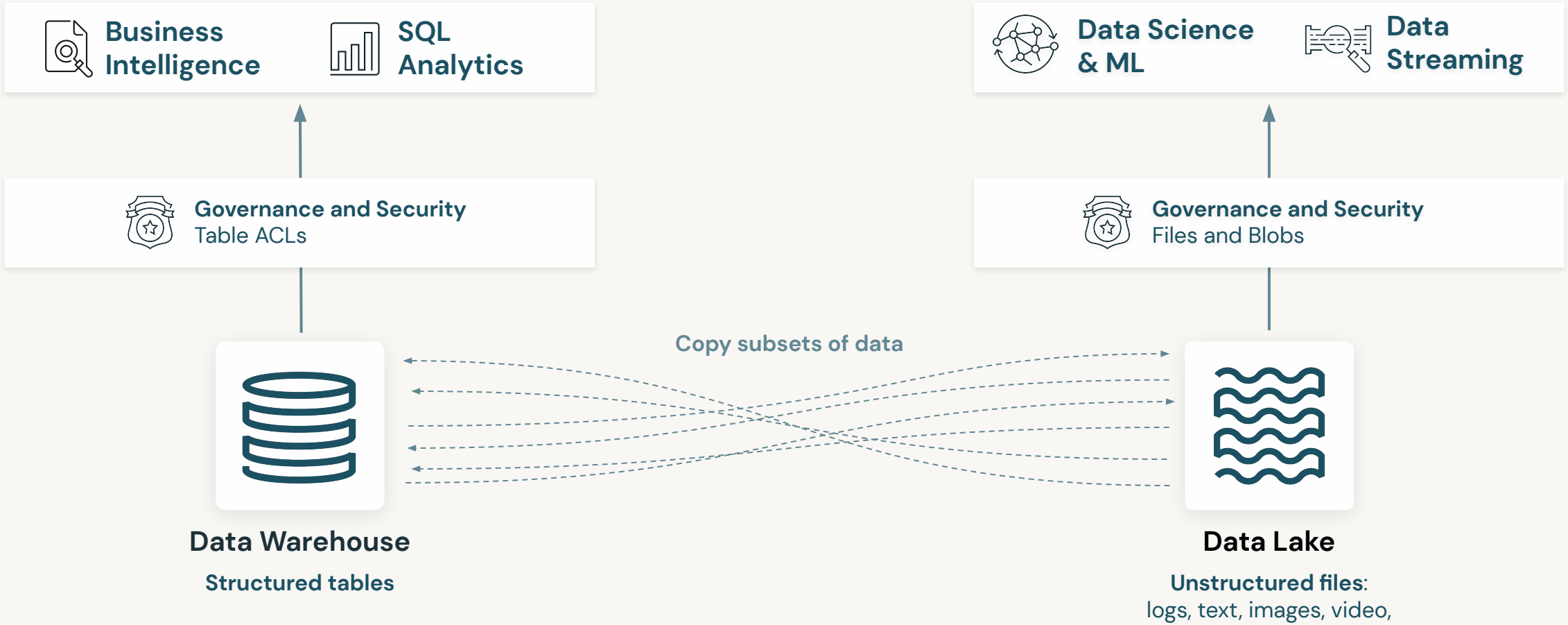


/vinoaj

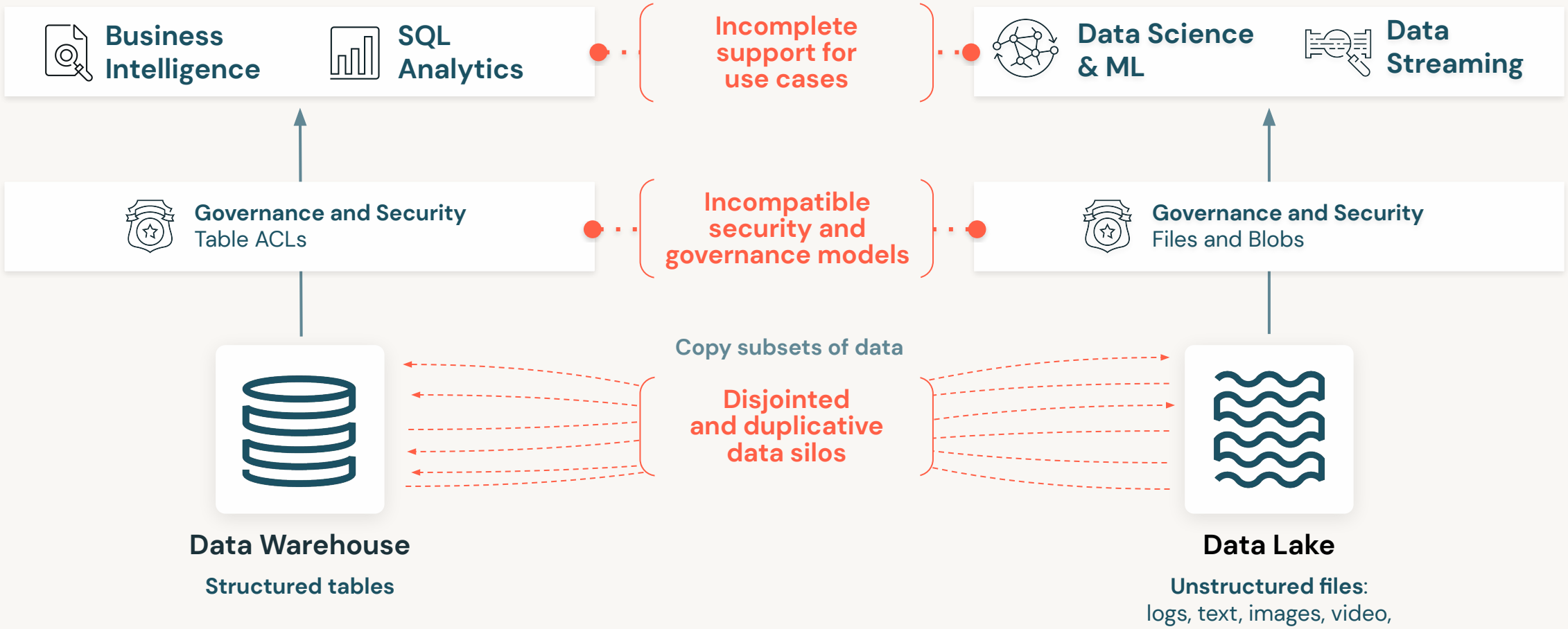
Realizing this requires two disparate, incompatible data platforms



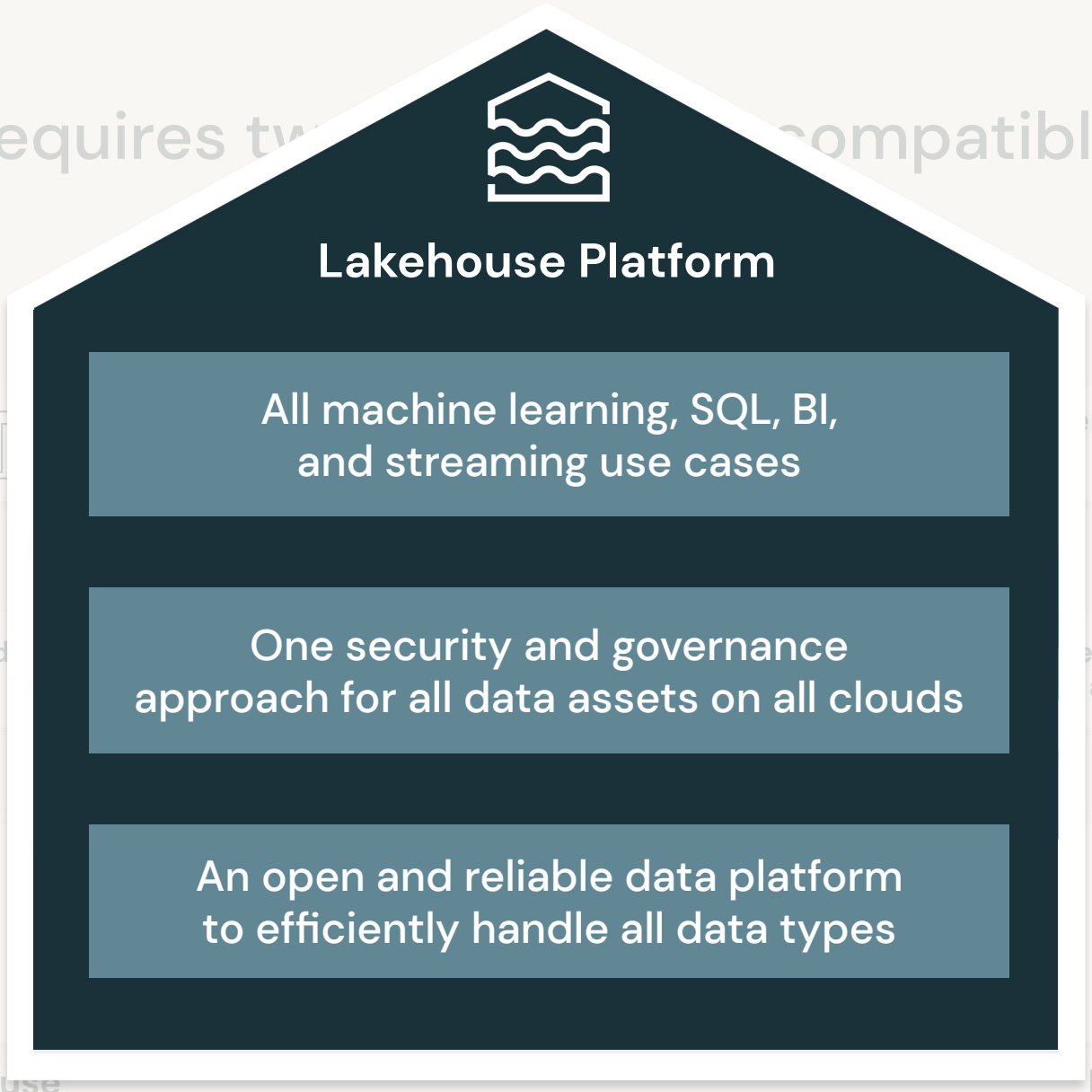
Realizing this requires two disparate, incompatible data platforms



Realizing this requires two disparate, incompatible data platforms



Realizing this requires tw compatible data platforms



Incomplete support for use cases

Incompatible security and governance models

Disjointed and duplicative data silos

Data Streaming

Data Warehouse

Structured tables

Data Lake

Unstructured files: logs, text, images, video,

Delta Lake makes the Lakehouse possible

Follow along @
https://vnjv.co/dec22

Integrations

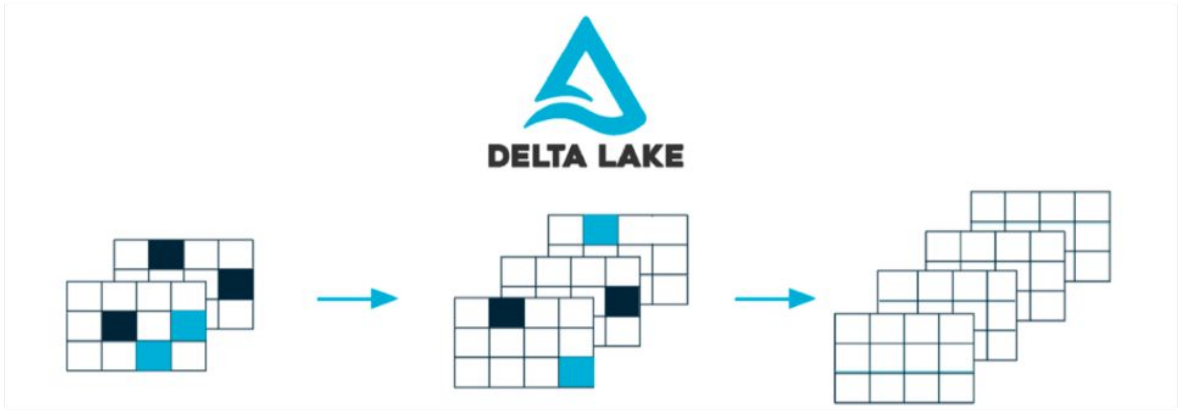
Spark, presto, Flink, trino, HIVE, kafka, amazon REDSHIFT, snowflake, Azure Synapse, DataHub, R, Python, Java

Coming Soon: amazon ATHENA, Google BigQuery, PULSAR

Streaming

Batch

DELTA LAKE



Ingestion Tables (Bronze) → Refined Tables (Silver) → Feature/Agg Data Store (Gold)

Analytics and Machine Learning

Your Existing Data Lake

Google Cloud, hadoop HDFS, Azure Data Lake Storage, amazon S3, IBM Cloud, ORACLE CLOUD

Community-driven!

/vinoaj

20 Jun 2019

liwensun

v0.2.0

ae3daa8

Compare

Delta Lake 0.2.0

We are delighted to announce the availability of Delta Lake 0.2.0!

To try out Delta Lake 0.2.0, please follow the Delta Lake Quickstart.

This release introduces two main features:

- **Cloud storage support** - In addition to HDFS, you can now configure Delta Lake to read and write data on cloud storage services such as Amazon S3 (issue #39) and Azure Blob Storage (issue #40). See here for configuration instructions.
- **Improved concurrency** (issue #69) - Delta Lake now allows concurrent *append-only* writes while still ensuring serializability. To be considered as append-only, a writer must be only adding new data without reading or modifying existing data in any way. See here for more details.

We have also greatly expanded the test coverage as part of this release.

Delta Lake 2.0 – All of Delta is now open

Unlock the power of Delta Lake

ACID Transactions

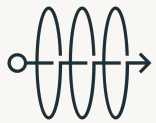
Scalable Metadata

Time Travel

Open Source

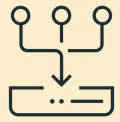
OPTIMIZE

OPTIMIZE ZORDER



Change data feed

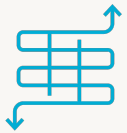
Generated column support w/ partitioning



Clones

Coming Soon!

Unified Batch/Streaming

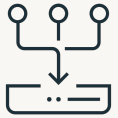


Schema Evolution /Enforcement

Audit History

DML Operations

Table Restore



S3 Multi-cluster writes

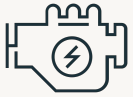
Data Skipping via Column Stats

Identity Columns

Iceberg to Delta converter

Compaction

MERGE Enhancements



Stream Enhancements

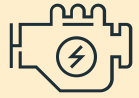
Simplified Logstore

Generated Columns

Multi-part checkpoint writes

Column Mapping

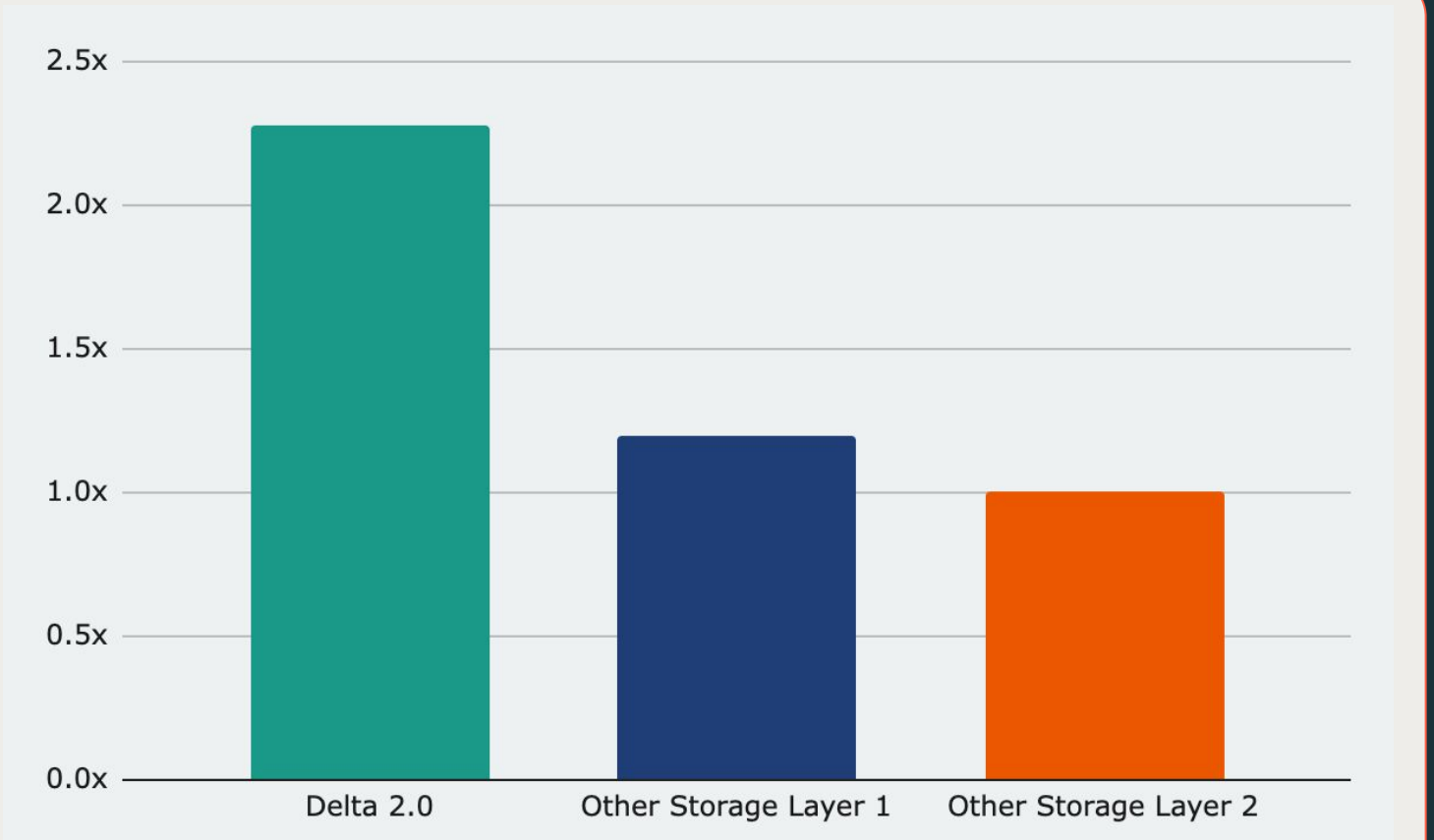
Subqueries in deletes and updates



Fast metadata only deletes

Delta Lake performance

TPC-DS Benchmark Comparison (Higher is better)



Delta Lake is

1.9x faster than Storage Format 1

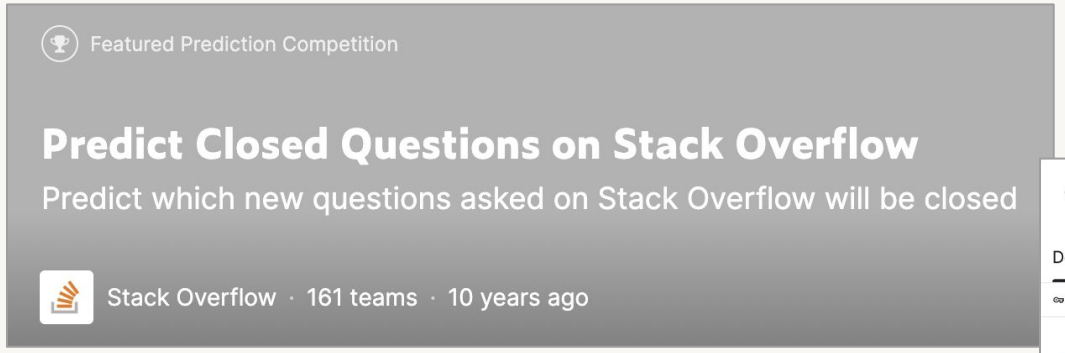
2.3x faster than Storage Format 2

Delta Lake: My Favourite Bits

(that fit into 10 minutes)

Follow along @
https://vnjv.co/dec22

Sample Data: Kaggle Stack Overflow Competition



Predict Closed Questions on Stack Overflow

Follow along @
https://vnjv.co/dec22

train.csv (3.73 GB)

Detail Compact Column 10 of 15 columns

PostId	PostCreationDate	OwnerUserId	OwnerCreationDate	ReputationAtPostCre...	OwnerUndeletedAns...	Title	BodyMarkdov
4 … 11.8m	1Aug08 … 1Aug12	1 … 1.57m	1Aug08 … 1Aug12	-55 … 461k	0 … 21.5k	3363198 unique values	33702 unique va
4	07/31/2008 21:42:52	8	07/31/2008 21:33:24	1	0	Decimal vs Double?	I'm new to C# want to use a trackbar for forms opacity is my code de tr...
6	07/31/2008 22:08:08	9	07/31/2008 21:35:26	1	0	Percentage width child in absolutely positioned parent doesn't work in IE7	I've got an absolutely positioned di containing se children, one which is a relatively po
8	07/31/2008 23:33:19	9	07/31/2008 21:35:26	16	1	Tools for porting J# code to C#	Are there any conversion to porting Visua code to C#?
9	07/31/2008 23:40:59	1	07/31/2008 14:22:31	1	1	How do I calculate someone's age in c#?	Given a DateT representing birthday, how calculate som age?
9610539	03/07/2012 23:07:09	1021610	10/31/2011 08:26:49	29	0	retrieve data from NSUserDefaults to TableView	I save values labels throug NSUserDefault (IBAction) saveDat...
39	08/01/2008 12:43:11	33	08/01/2008 12:32:10	1	0	Reliable Timer in a Console Application	I am aware in net there are timer types ([http://msdn. ft.com/en- us/magazine/c

Let's create our first Delta Table

```
filename = "train.csv"
filepath =
    f"/Users/{USERNAME}/custom_demos/input_files/{KAGGLE_COMPETITION}/{filename}"
print(f"Loading contents of: {filepath}")

data_csv = (spark.read
                .option("header", "true")
                .option("multiLine", "true")
                .option("escape", '"') # Handle double quotes in multiline entries
                .csv(filepath))

(data_csv.write
   .format("delta")
   .mode("overwrite")
   # All Delta Lake files (i.e. data and log files) for this table will be stored here
   .save(f"/Users/{USERNAME}/custom_demos/lakehouse/stackoverflow_train00"))
```

```
(spark.read.format("delta")
    .load(f"/Users/{USERNAME}/custom_demos/lakehouse/stackoverflow_train00")
    .display())
```

Table +

	PostId	PostCreationDate	OwnerUserId	OwnerCreationDate	ReputationAtPostCreation	OwnerUndeletedAnswerCountAtPostTime	Title
1	4	07/31/2008 21:42:52	8	07/31/2008 21:33:24	1	0	Decimal vs Double?
2	6	07/31/2008 22:08:08	9	07/31/2008 21:35:26	1	0	Percentage width child in absolutely positioned pare
3	8	07/31/2008 23:33:19	9	07/31/2008 21:35:26	16	1	Tools for porting J# code to C#
4	9	07/31/2008 23:40:59	1	07/31/2008 14:22:31	1	1	How do I calculate someone's age in c#?
5	9610539	03/07/2012 23:07:09	1021610	10/31/2011 08:26:49	29	0	retrieve data from NSUserDefaults to TableView
6	39	08/01/2008 12:43:11	33	08/01/2008 12:32:10	1	0	Reliable Timer in a Console Application

Truncated results, showing first 1,000 rows. | 3.23 seconds runtime

Create a Delta Lake table using SQL

```sql
CREATE OR REPLACE TEMPORARY VIEW tvw_train
USING CSV -- Databricks-specific operator
OPTIONS(
  path "/Users/vinny.vijeyakumaar@databricks.com/custom_demos/input_files/predict-closed-questions-on-stack-overflow/train.csv",
  header "true", multiLine "true", escape '"'
);

-- Create the Delta table
CREATE OR REPLACE TABLE vinny_vijeyakumaar.stackoverflow_train01
USING DELTA
PARTITIONED BY (PostCreationDate)
TBLPROPERTIES(
  delta.enableChangeDataFeed = true,
  -- autoOptimize.* available in Databricks; P1 priority on current Delta Lake OSS roadmap
  delta.autoOptimize.optimizeWrite = true,
  delta.autoOptimize.autoCompact = true
)
-- (Optional) All Delta Lake files (i.e. data and log files) for this table will be stored here
-- If not specified, location will be managed by your active metastore
LOCATION "/Users/vinny.vijeyakumaar@databricks.com/custom_demos/lakehouse/stackoverflow_train01"
AS
  SELECT
    PostId
    , DATE(TO_TIMESTAMP(PostCreationDate, "MM/dd/yyyy HH:mm:ss")) AS PostCreationDate
    , * EXCEPT (PostId, PostCreationDate)
  FROM tvw_train;

-- Cleanup
DROP VIEW IF EXISTS tvw_train;

-- Show us the data
SELECT *
FROM vinny_vijeyakumaar.stackoverflow_train01;
```

Follow along @ https://vnjv.co/dec22

Let's take a closer look

```
CREATE OR REPLACE TABLE vinny_vijeyakumaar.stackoverflow_train01
USING DELTA
PARTITIONED BY (PostCreationDate)
TBLPROPERTIES(
  delta.enableChangeDataFeed = true,
  -- autoOptimize.* available in Databricks; P1 priority on current Delta Lake OSS roadmap
  delta.autoOptimize.optimizeWrite = true,
  delta.autoOptimize.autoCompact = true
)
-- (Optional) All Delta Lake files (i.e. data and log files) for this table will be stored here
-- If not specified, location will be managed by your active metastore
LOCATION "/Users/vinny.vijeyakumaar@databricks.com/custom_demos/lakehouse/stackoverflow_train01"
AS
  SELECT
```

```
SELECT *
FROM vinny_vijeyakumaar.stackoverflow_train01;
```

Table + Give feedback

	PostId	PostCreationDate	OwnerUserId	OwnerCreationDate	ReputationAtPostCreation	OwnerUndeletedAnswerCountAtPostTime	Title
1	8704	2008-08-12	1042	08/11/2008 18:02:48	1	0	Final managed exception handler in a mixed na
2	8702	2008-08-12	1097	08/12/2008 11:40:09	1	0	What is the IDE for classic ASP and VBScript?
3	9009	2008-08-12	238	08/03/2008 21:42:37	94	13	Perl conditional operator is giving the wrong re
4	8728	2008-08-12	1008	08/11/2008 12:53:05	21	2	Delphi MDI Application and the titlebar of the M
5	8726	2008-08-12	572	08/06/2008 20:56:54	115	11	Checking FTP status codes with a PHP script.

Delta Lake 2.0 – All of Delta is now open

Unlock the power of Delta Lake

ACID Transactions

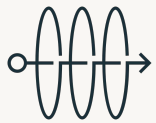
Scalable Metadata

Time Travel

Open Source

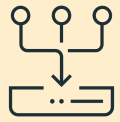
OPTIMIZE

OPTIMIZE ZORDER



Change data feed

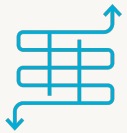
Generated column support w/ partitioning



Clones

Coming Soon!

Unified Batch/Streaming

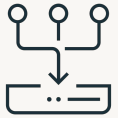


Schema Evolution /Enforcement

Audit History

DML Operations

Table Restore



S3 Multi-cluster writes

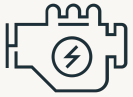
Data Skipping via Column Stats

Identity Columns

Iceberg to Delta converter

Compaction

MERGE Enhancements



Stream Enhancements

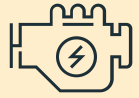
Simplified Logstore

Generated Columns

Multi-part checkpoint writes

Column Mapping

Subqueries in deletes and updates



Fast metadata only deletes

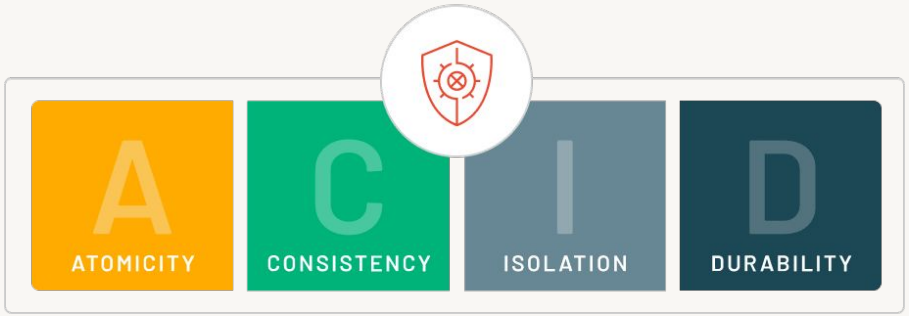
Delta Lake 2.0 – All of Delta is now open

Unlock the power of Delta Lake

ACID Transactions

Scalable Metadata

OPTIMIZE ZORDER



Data Skipping via Column Stats

Delta Lake File Structure

```
1  %fs ls /Users/vinny.vijeyakumaar@databricks.com/custom_demos/lakehouse/stackoverflow_train01/PostCreationDate=2008-07-31/
```

Python

Table + Give feedback

	path	name	size	modificationTime
1	dbfs:/Users/vinny.vijeyakumaar@databricks.com/custom_demos/lakehouse/stackoverflow_train01/PostCreationDate=2008-07-31/part-00000-28f21626-97da-4160-87e8-11cc19d91da3.c000.snappy.parquet	part-00000-28f21626-97da-4160-87e8-11cc19d91da3.c000.snappy.parquet	5614	1663966928000

Follow along @
https://vnjv.co/dec22

Delta Lake File Structure

```
1  %fs ls /Users/vinny.vijeyakumaar@databricks.com/custom_demos/lakehouse/stackoverflow_train01/_delta_log/
```

Table +

	path	name	size	modificationTime
2	dbfs:/Users/vinny.vijeyakumaar@databricks.com/custom_demos/lakehouse/stackoverflow_train01/_delta_log/.s3-optimization-1	.s3-optimization-1	0	1663889875000
3	dbfs:/Users/vinny.vijeyakumaar@databricks.com/custom_demos/lakehouse/stackoverflow_train01/_delta_log/.s3-optimization-2	.s3-optimization-2	0	1663889875000
4	dbfs:/Users/vinny.vijeyakumaar@databricks.com/custom_demos/lakehouse/stackoverflow_train01/_delta_log/00000000000000000000.crc	00000000000000000000.crc	3159	1663889875000
5	dbfs:/Users/vinny.vijeyakumaar@databricks.com/custom_demos/lakehouse/stackoverflow_train01/_delta_log/00000000000000000000.json	00000000000000000000.json	4129	1663889874000
6	dbfs:/Users/vinny.vijeyakumaar@databricks.com/custom_demos/lakehouse/stackoverflow_train01/_delta_log/00000000000000000001.crc	00000000000000000001.crc	3177	1663967267000
7	dbfs:/Users/vinny.vijeyakumaar@databricks.com/custom_demos/lakehouse/stackoverflow_train01/_delta_log/00000000000000000001.json	00000000000000000001.json	2445162	1663967264000

Let's generate more transactions

```
import random

for _ in range(250):
  [x, y] = random.sample(range(100000,999999), 2)
  spark.sql(f"""
    INSERT INTO vinny_vijeyakumaar.stackoverflow_train01 VALUES
      ({x}, DATE("2007-08-01"), "", {x}, "", "", "", "", "", "", "", "", "", "", "", ""),
      ({y}, DATE("2007-08-01"), "", {y}, "", "", "", "", "", "", "", "", "", "", "", "")
  """)
```

A new log per successful transaction set

Displaying contents of /Users/vinny.vijeyakumaar@databricks.com/custom_demos/lakehouse/stackoverflow_train01//_delta_log/

Table +

	path	name	size	modificationTime
515	dbfs:/Users/vinny.vijeyakumaar@databricks.com/custom_demos/lakehouse/stackoverflow_train01/_delta_log/00000000000000000254.json	00000000000000000254.json	1869	1663983013000
516	dbfs:/Users/vinny.vijeyakumaar@databricks.com/custom_demos/lakehouse/stackoverflow_train01/_delta_log/00000000000000000255.crc	00000000000000000255.crc	3182	1663983017000
517	dbfs:/Users/vinny.vijeyakumaar@databricks.com/custom_demos/lakehouse/stackoverflow_train01/_delta_log/00000000000000000255.json	00000000000000000255.json	1869	1663983016000
518	dbfs:/Users/vinny.vijeyakumaar@databricks.com/custom_demos/lakehouse/stackoverflow_train01/_delta_log/00000000000000000256.crc	00000000000000000256.crc	3182	1663983019000
519	dbfs:/Users/vinny.vijeyakumaar@databricks.com/custom_demos/lakehouse/stackoverflow_train01/_delta_log/00000000000000000256.json	00000000000000000256.json	1869	1663983018000
520	dbfs:/Users/vinny.vijeyakumaar@databricks.com/custom_demos/lakehouse/stackoverflow_train01/_delta_log/_last_checkpoint	_last_checkpoint	6854	1663982904000

Peeking inside a log file


```
{
    "protocol": {
        "minReaderVersion": 1,
        "minWriterVersion": 2
    }
}
{
    "metaData": {
        "id": "00a4f716-7cf3-4c2a-8804-16bcb4ead707",
        "format": {
            "provider": "parquet",
            "options": {}
        },
        "schemaString": "{\"type\":\"struct\",\"fields\":
[{\"name\":\"PostId\",\"type\":\"string\",\"nullable\":true,\"metadata\":{}},
{\"name\":\"PostCreationDate\",\"type\":\"date\",\"nullable\":true,\"metadata\":{}},
{\"name\":\"OwnerUserId\",\"type\":\"string\",\"nullable\":true,\"metadata\":{}},
{\"name\":\"OwnerCreationDate\",\"type\":\"string\",\"nullable\":true,\"metadata\":{}},
{\"name\":\"ReputationAtPostCreation\",\"type\":\"string\",\"nullable\":true,\"metadata\":{}},
{\"name\":\"OwnerUndeletedAnswerCountAtPostTime\",\"type\":\"string\",\"nullable\":true,\"metadata\":
{}},{\"name\":\"Title\",\"type\":\"string\",\"nullable\":true,\"metadata\":{}},
{\"name\":\"BodyMarkdown\",\"type\":\"string\",\"nullable\":true,\"metadata\":{}},
{\"name\":\"Tag1\",\"type\":\"string\",\"nullable\":true,\"metadata\":{}},
{\"name\":\"Tag2\",\"type\":\"string\",\"nullable\":true,\"metadata\":{}},
{\"name\":\"Tag3\",\"type\":\"string\",\"nullable\":true,\"metadata\":{}},
{\"name\":\"Tag4\",\"type\":\"string\",\"nullable\":true,\"metadata\":{}},
{\"name\":\"Tag5\",\"type\":\"string\",\"nullable\":true,\"metadata\":{}},
{\"name\":\"PostClosedDate\",\"type\":\"string\",\"nullable\":true,\"metadata\":{}},
{\"name\":\"OpenStatus\",\"type\":\"string\",\"nullable\":true,\"metadata\":{}}]}",
        "partitionColumns": [
            "PostCreationDate"
        ],
        "configuration": {
            "delta.autoOptimize.autoCompact": "true",
            "delta.autoOptimize.optimizeWrite": "true"
        },
        "createdTime": 1663966838668
    }
}
```


Follow along @
https://vnjv.co/dec22

Log Contents


```
{
    "add": {
        "path": "PostCreationDate=2008-08-03/part-00000-ddf05b95-9e25-4558-b3d7-
96475df8362d.c000.snappy.parquet",
        "partitionValues": {
            "PostCreationDate": "2008-08-03"
        },
        "size": 22643,
        "modificationTime": 1663966929000,
        "dataChange": true,
        "stats": "{\"numRecords\":39,\"minValues\":
{\"PostId\":\"588\",\"OwnerUserId\":\"101\",\"OwnerCreationDate\":\"08/01/2008
12:01:23\",\"ReputationAtPostCreation\":\"1\",\"OwnerUndeletedAnswerCountAtPostTime\":\"0\",\"Title\":\
".NET Testing Framework Advice\",\"BodyMarkdown\":\"Alright, this is a rather odd
qu\",\"Tag1\":\".net\",\"Tag2\":\".net\",\"Tag3\":\"api\",\"Tag4\":\"2008\",\"Tag5\":\"asp.net\",\"Post
ClosedDate\":\"06/19/2012 17:52:00\",\"OpenStatus\":\"not constructive\"},\"maxValues\":
{\"PostId\":\"888\",\"OwnerUserId\":\"93\",\"OwnerCreationDate\":\"08/03/2008
21:42:37\",\"ReputationAtPostCreation\":\"56\",\"OwnerUndeletedAnswerCountAtPostTime\":\"8\",\"Title\":
\"cx_Oracle - what is the best way\ufffd\",\"BodyMarkdown\":\"[Django][1] view points to a
fun\ufffd\",\"Tag1\":\"windows\",\"Tag2\":\"windows\",\"Tag3\":\"xslt\",\"Tag4\":\"vb.net\",\"Tag5\":\"
zend\",\"PostClosedDate\":\"06/19/2012 17:52:00\",\"OpenStatus\":\"open\"},\"nullCount\":
{\"PostId\":0,\"OwnerUserId\":0,\"OwnerCreationDate\":0,\"ReputationAtPostCreation\":0,\"OwnerUndeleted
AnswerCountAtPostTime\":0,\"Title\":0,\"BodyMarkdown\":0,\"Tag1\":0,\"Tag2\":3,\"Tag3\":12,\"Tag4\":26,
\"Tag5\":34,\"PostClosedDate\":38,\"OpenStatus\":0}}",
        "tags": {
            "INSERTION_TIME": "1663966928000003",
            "MIN_INSERTION_TIME": "1663966928000003",
            "MAX_INSERTION_TIME": "1663966928000003",
            "OPTIMIZE_TARGET_SIZE": "268435456"
        }
    }
}
```

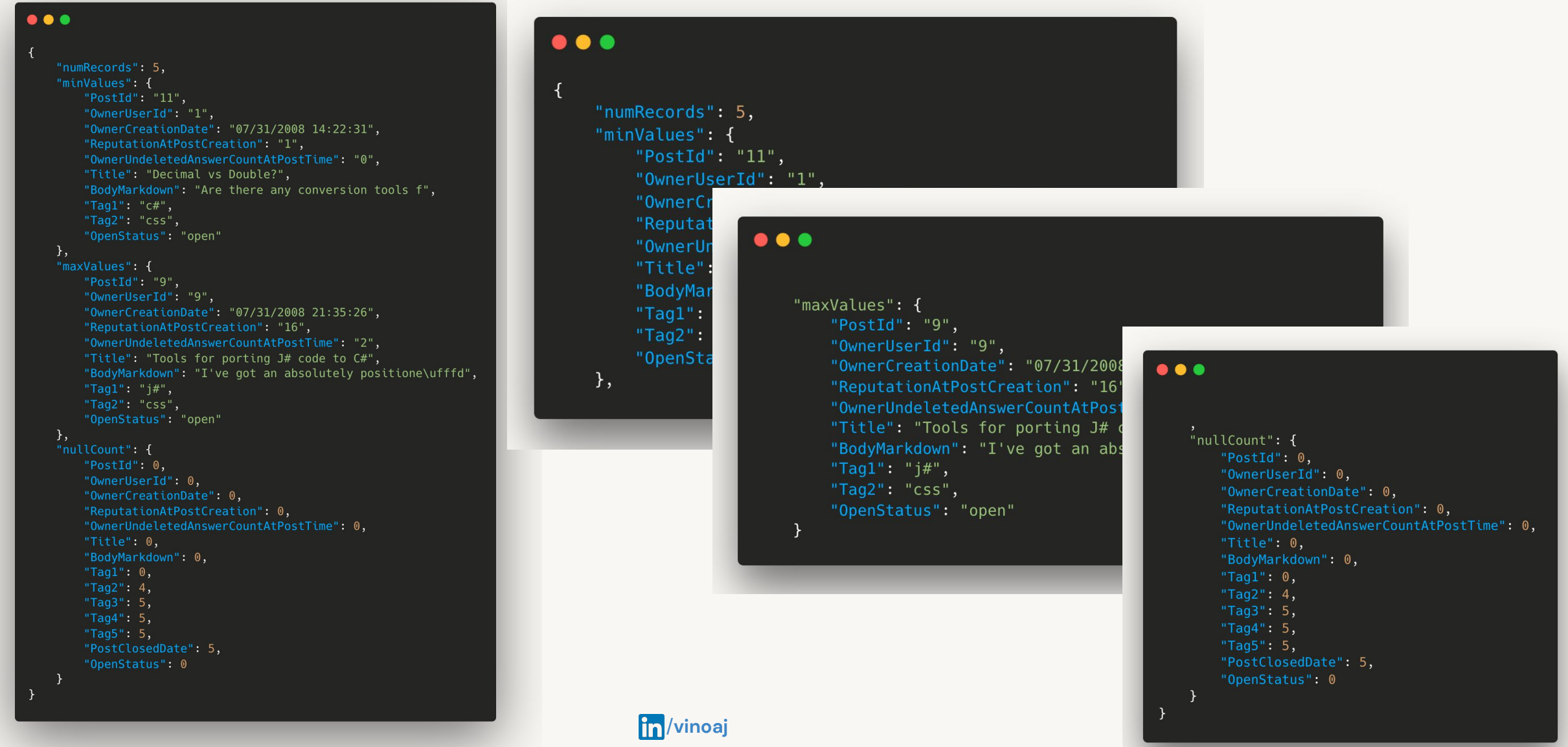

We know this is a reliable, fully-written out file, as the outcome of a successful DML transaction

Column stats are useful during query planning time to determine which files can be skipped (skipped files → less I/O, less memory required, reduced chance of disk spill)

Log entry after a MERGE INTO … command


```
{
    "remove": {
        "path": "PostCreationDate=2007-08-01/part-00000-f982e89f-4264-43bb-a043-
f1ba0a27492c.c000.snappy.parquet",
        "deletionTimestamp": 1664314049556,
        "dataChange": true,
        "extendedFileMetadata": true,
        "partitionValues": {
            "PostCreationDate": "2007-08-01"
        },
        "size": 3376,
        "tags": {
            "INSERTION_TIME": "1664314022000000",
            "MIN_INSERTION_TIME": "1664314022000000",
            "MAX_INSERTION_TIME": "1664314022000000",
            "OPTIMIZE_TARGET_SIZE": "268435456"
        }
    }
}
{
    "add": {
        "path": "PostCreationDate=2007-08-01/part-00000-5d440efc-3877-4fa4-8486-
6725dc987151.c000.snappy.parquet",
        "partitionValues": {
            "PostCreationDate": "2007-08-01"
        },
        "size": 3550,
        "modificationTime": 1664314050000,
        "dataChange": true,
        "stats": "{\"numRecords\":4,\"minValues\":
{\"PostId\":\"1078360\",\"OwnerUserId\":\"\",\"OwnerCreationDate\":\"1078360\",\"ReputationAtPostCreation
\":\"\",\"OwnerUndeletedAnswerCountAtPostTime\":\"\",\"Title\":\"\",\"BodyMarkdown\":\"\",\"Tag1\":\"\",\
"Tag2\":\"\",\"Tag3\":\"\",\"Tag4\":\"\",\"Tag5\":\"\",\"PostClosedDate\":\"\",\"OpenStatus\":\"\"},\"max
Values\":
{\"PostId\":\"999999\",\"OwnerUserId\":\"\",\"OwnerCreationDate\":\"999999\",\"ReputationAtPostCreation\"
:\"\",\"OwnerUndeletedAnswerCountAtPostTime\":\"\",\"Title\":\"\",\"BodyMarkdown\":\"\",\"Tag1\":\"\",\"T
```

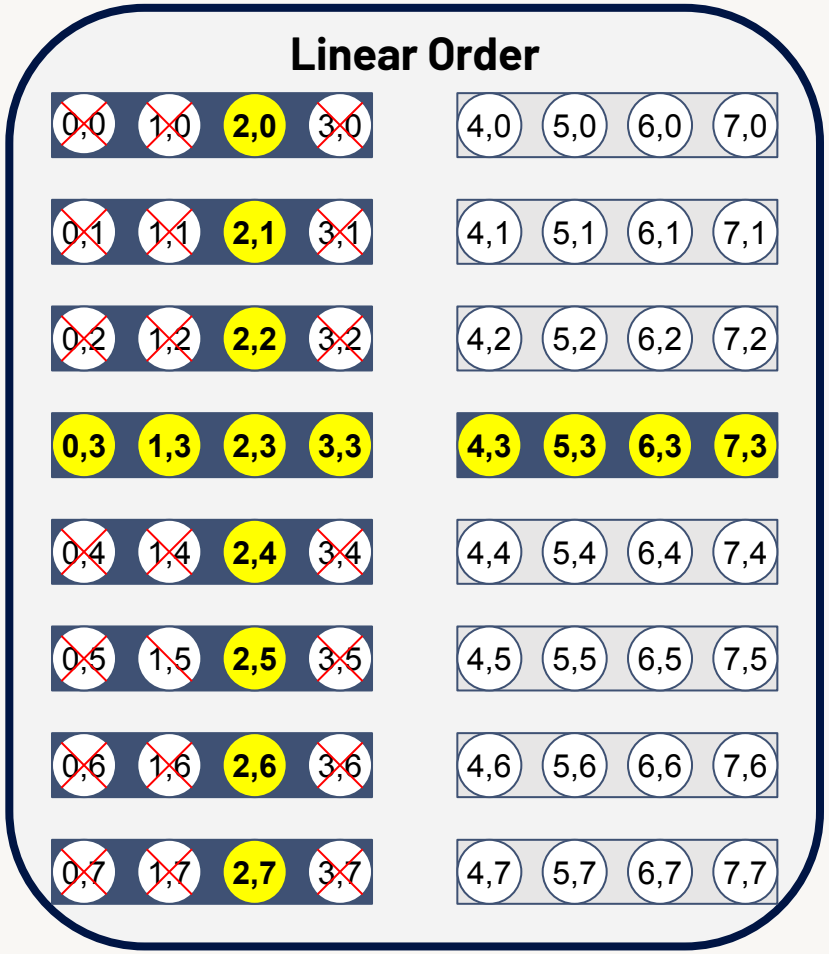
Unpacking "stats"

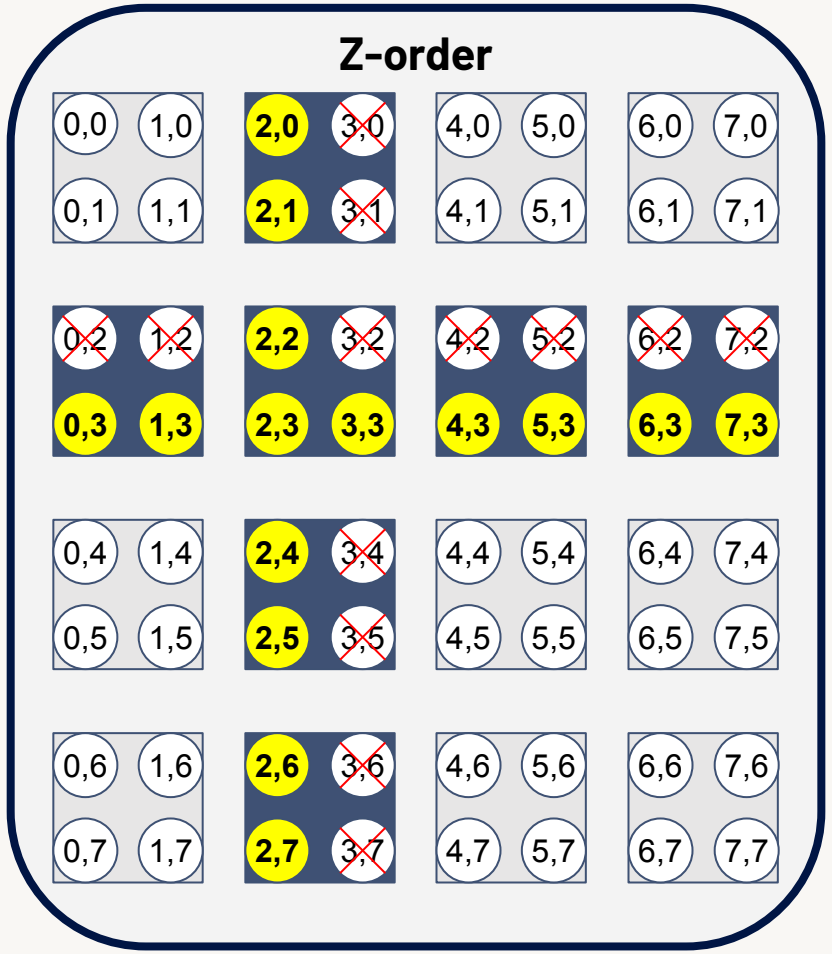


/vinoaj

ZORDER BY (col1, …)

Read in each part-file and test for X = 2 or Y = 3



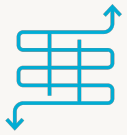


9 files scanned in total 👎
21 false positives 👎

7 files scanned in total 👍
13 false positives 👍

Delta Lake 2.0 – All of Delta is now open

Unlock the power of Delta Lake



Schema Evolution
/Enforcement

OOTB Schema Enforcement

```
1  %sql
2  INSERT INTO vinny_vijeyakumaar.stackoverflow_train01 VALUES
3    ("abc", "2007-08-01", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "extraColumn"),
4    ("xyz", DATE("2007-08-01"), "", 999999, "", "", "", "", "", "", "", "", "", "", "", "", "")
```

```
⊞AnalysisException: expected 17 columns but found 16 columns in row 1; line 1 pos 53
```

OOTB Schema Enforcement

```
1  %sql
2  INSERT INTO vinny_vijeyakumaar.stackoverflow_train01 VALUES
3    ("abc", "2007-08-01", "", "", "", "", "", "", "", "", "", "", "", "", "", "", ""),
4    ("xyz", DATE("2007-08-01"), "", 999999, "", "", "", "", "", "", "", "", "", "", "", "", "")
```

```
⊞AnalysisException: incompatible types found in column col2 for inline table; line 1 pos 53
```

Adding Constraints

```
ALTER TABLE vinny_vijeyakumaar.stackoverflow_train01
  ADD CONSTRAINT post_id_not_null CHECK (PostId IS NOT NULL);

ALTER TABLE vinny_vijeyakumaar.stackoverflow_train01
  ADD CONSTRAINT post_creation_date_valid CHECK (
    PostCreationDate BETWEEN "2007-08-01" AND "2012-09-01");
```

Constraint Enforcement

```
1  %sql
2  INSERT INTO vinny_vijeyakumaar.stackoverflow_train01 VALUES
3    ("xyz", DATE("2022-09-29"), "", 999999, "", "", "", "", "", "", "", "", "", "", "")
```

▸ (1) Spark Jobs

⊞com.databricks.sql.transaction.tahoe.schema.DeltaInvariantViolationException: CHECK constraint post_creation_date_valid ((PostCreationDate >= '2007-08-01') AND (PostCreationDate <= '2012-09-01')) violated by row with values:

Follow along @
https://vnjv.co/dec22

Delta Lake 2.0 – All of Delta is now open

Unlock the power of Delta Lake

Time Travel

Audit History

Table Restore


WHERE WERE GOING WE
DONT NEED ... ROADS

```
DESCRIBE HISTORY vinny_vijeyakumaar.stackoverflow_train01
LIMIT 10 -- get the last 10 operations only
```

Table + Give feedback

	version	timestamp	userId	userName	operation	operationParameters
1	274	2022-09-28T02:03:37.000+0000	5211215277230706	vinny.vijeyakumaar@databricks.com	MERGE	{"predicate": "(target.PostId = source "notMatchedPredicates": "[{\"actionTyp
2	273	2022-09-27T23:25:04.000+0000	5211215277230706	vinny.vijeyakumaar@databricks.com	WRITE	{"mode": "Append", "partitionBy": "[]
3	272	202				
4	271	202				
5	270	202				
6	269	202				
7	268	202				
8	267	202				
9	266	202				

clusterId	readVersion	isolationLevel	isBlindAppend	operationMetrics
0923-094955-git4n0b4	273	WriteSerializable	false	{"numTargetRowsCopied": "196", "numTargetRowsDeleted": "0", "numTargetFil "5529", "numTargetRowsInserted": "0", "scanTimeMs": "2848", "numTargetRows "200", "numTargetChangeFilesAdded": "1", "numSourceRows": "4", "numTargetF "1087"}
0923-094955-git4n0b4	272	WriteSerializable	true	{"numFiles": "1", "numOutputRows": "2", "numOutputBytes": "3228"}
0923-094955-git4n0b4	271	WriteSerializable	true	{"numFiles": "1", "numOutputRows": "2", "numOutputBytes": "3228"}
0923-094955-git4n0b4	270	WriteSerializable	true	{"numFiles": "1", "numOutputRows": "2", "numOutputBytes": "3226"}
0923-094955-git4n0b4	269	WriteSerializable	true	{"numFiles": "1", "numOutputRows": "2", "numOutputBytes": "3228"}
0923-094955-git4n0b4	268	WriteSerializable	true	{"numFiles": "1", "numOutputRows": "2", "numOutputBytes": "3228"}
0923-094955-git4n0b4	267	WriteSerializable	false	{"numRemovedFiles": "1", "numCopiedRows": "85", "numAddedChangeFiles": " "scanTimeMs": "1610", "numAddedFiles": "1", "numUpdatedRows": "2", "rewrite
0923-094955-git4n0b4	266	WriteSerializable	false	{"numRemovedFiles": "1", "numCopiedRows": "93", "numAddedChangeFiles": " "scanTimeMs": "1784", "numAddedFiles": "1", "numUpdatedRows": "2", "rewrite
0923-094955-git4n0b4	265	WriteSerializable	true	{"numFiles": "1", "numOutputRows": "2", "numOutputBytes": "3228"}

Time Travel by version number or timestamp

```sql
WITH rec_count_00 AS (
  SELECT COUNT(*) AS n FROM vinny_vijeyakumaar.stackoverflow_train01
  VERSION AS OF 1
)
, rec_count_01 AS (
  SELECT COUNT(*) AS n FROM vinny_vijeyakumaar.stackoverflow_train01
  TIMESTAMP AS OF (CURRENT_TIMESTAMP() - INTERVAL 240 MINUTES)
)
, rec_count_now AS (
  SELECT COUNT(*) AS n FROM vinny_vijeyakumaar.stackoverflow_train01
)
SELECT r0.n AS start, r1.n AS 240minsAgo, rn.n AS now
FROM rec_count_00 r0, rec_count_01 r1, rec_count_now rn
```

Follow along @ https://vnjv.co/dec22

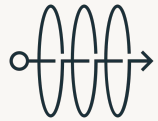
start	240minsAgo	now
3352998	3353020	3353280

Restore to a previous state

```
-- Restore table to previous state
RESTORE TABLE vinny_vijeyakumaar.stackoverflow_train01
TO VERSION AS OF 12
-- TO TIMESTAMP AS OF "2022-09-28 10:00:00"
```

Delta Lake 2.0 – All of Delta is now open

Unlock the power of Delta Lake



Change data feed

Audit History

Track changes to your data with Change Data Feed (CDF)

```
1  %sql
2  SELECT * FROM TABLE_CHANGES("vinny_vijeyakumaar.stackoverflow_train01", 270)
3  WHERE PostId = "13211"
```

SQL

▸ (3) Spark Jobs

Table + Give feedback

	Tag5	PostClosedDate	OpenStatus	_change_type	_commit_version	_commit_timestamp
1	null	11/21/2011 07:45:23	not constructive	update_postimage	289	2022-09-25T01:24:59.000+0000
2	systems	11/21/2011 07:45:23	not constructive	update_preimage	289	2022-09-25T01:24:59.000+0000

Propagate changes downstream

```sql
INSERT INTO target_schema.target_table
SELECT * EXCEPT (_change_type, _commit_version, _commit_timestamp)
-- Let's assume our orchestration tool knows that the last update happened at 2022-09-29 00:00:00
FROM TABLE_CHANGES("vinny_vijeyakumaar.stackoverflow_train01", "2022-09-29 00:00:00")
WHERE _change_type IN ("insert", "update_postimage");

DELETE FROM target_schema.target_table
WHERE PostId IN (
  SELECT PostId
  -- Let's assume our orchestration tool knows that the last update happened at 2022-09-29 00:00:00
  FROM TABLE_CHANGES("vinny_vijeyakumaar.stackoverflow_train01", "2022-09-29 00:00:00")
  WHERE _change_type IN ("delete")
);
```

Delta Lake 2.0 – All of Delta is now open

Unlock the power of Delta Lake

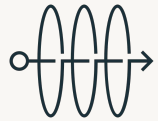
ACID Transactions

Scalable Metadata

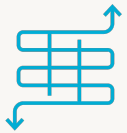
Time Travel

Open Source

OPTIMIZE



Change data feed



Schema Evolution /Enforcement

Audit History

DML Operations

Data Skipping via Column Stats

Delta Lake 2.0 – All of Delta is now open

Unlock the power of Delta Lake

ACID Transactions

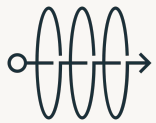
Scalable Metadata

Time Travel

Open Source

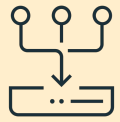
OPTIMIZE

OPTIMIZE ZORDER



Change data feed

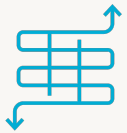
Generated column support w/ partitioning



Clones

Coming Soon!

Unified Batch/Streaming

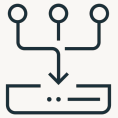


Schema Evolution /Enforcement

Audit History

DML Operations

Table Restore



S3 Multi-cluster writes

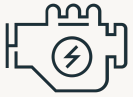
Data Skipping via Column Stats

Identity Columns

Iceberg to Delta converter

Compaction

MERGE Enhancements



Stream Enhancements

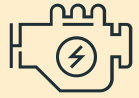
Simplified Logstore

Generated Columns

Multi-part checkpoint writes

Column Mapping

Subqueries in deletes and updates

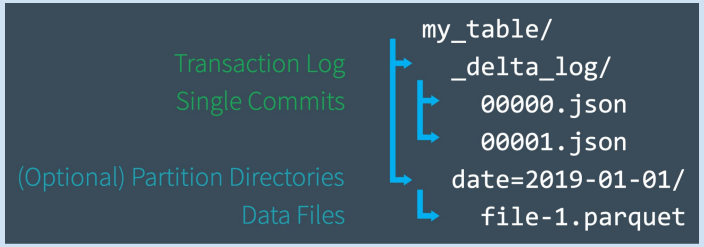


Fast metadata only deletes

Deep-dive Resources

Follow along @
https://vnjv.co/dec22

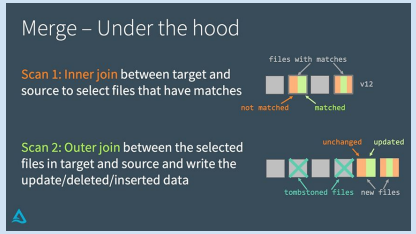
Diving Into … series



Link

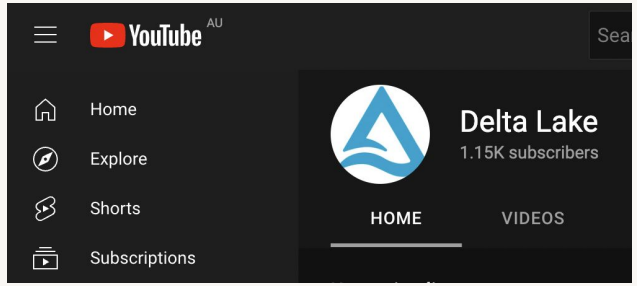

```
# Add the mergeSchema option
loans.write.format("delta") \
           .option("mergeSchema", "true") \
           .mode("append") \
           .save(DELTALAKE_SILVER_PATH)
```

Link



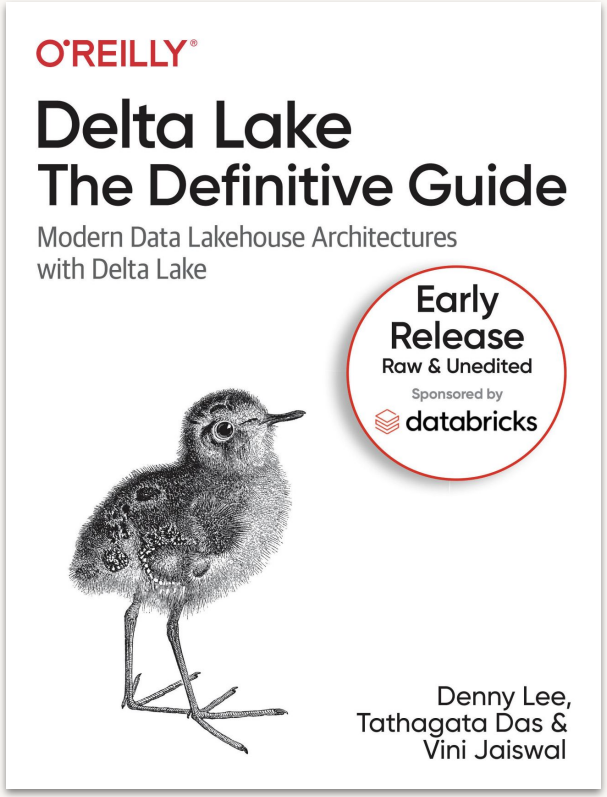
Link

Follow along @ https://vnjv.co/dec22

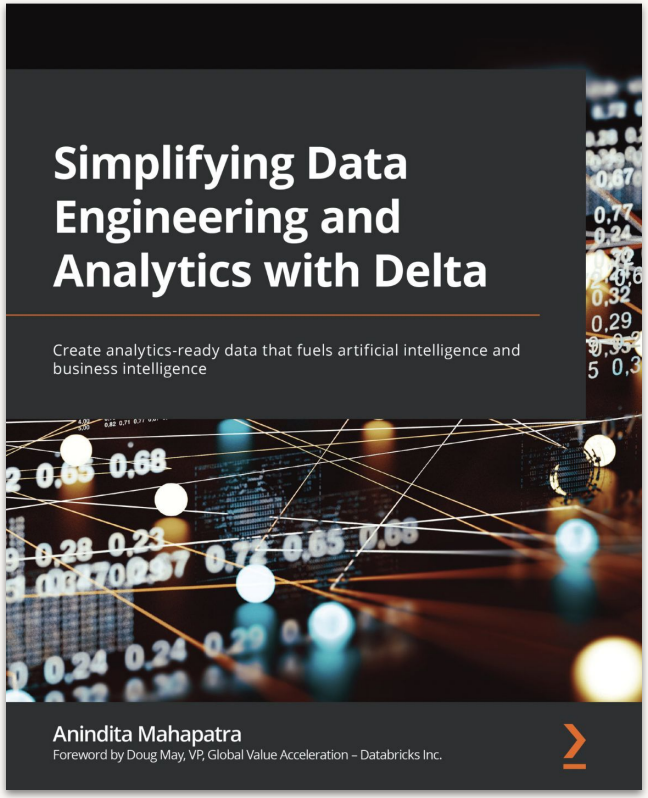


YouTube Channel: Deep Dives, AMAs, Community Office Hours, Tutorials, Tech Talks

/vinoaj



[Free preview!]



[Amazon]

Delta Lake: High-Performance ACID Table Storage over Cloud Object Stores

Michael Armbrust, Tathagata Das, Liwen Sun, Burak Yavuz, Shixiong Zhu, Mukul Murthy, Joseph Torres, Herman van Hovell, Adrian Ionescu, Alicja Łuszczak, Michał Świtakowski, Michał Szafrański, Xiao Li, Takuya Ueshin, Mostafa Mokhtar, Peter Boncz[1], Ali Ghodsi[2], Sameer Paranjpye, Pieter Senster, Reynold Xin, Matei Zaharia[3]
Databricks, [1]CWI, [2]UC Berkeley, [3]Stanford University
delta-paper-authors@databricks.com

ABSTRACT

Cloud object stores such as Amazon S3 are some of the largest and most cost-effective storage systems on the planet, making them an attractive target to store large data warehouses and data lakes. Unfortunately, their implementation as key-value stores makes it difficult to achieve ACID transactions and high performance: metadata operations such as listing objects are expensive, and consistency guarantees are limited. In this paper, we present Delta Lake, an open source ACID table storage layer over cloud object stores initially developed at Databricks. Delta Lake uses a transaction log that is compacted into Apache Parquet format to provide ACID properties, time travel, and significantly faster metadata operations for large tabular datasets (e.g., the ability to quickly search billions of table partitions for those relevant to a query). It also leverages this design to provide high-level features such as automatic data layout optimization, upserts, caching, and audit logs. Delta Lake tables can be accessed from Apache Spark, Hive, Presto, Redshift and other systems. Delta Lake is deployed at thousands of Databricks customers that process exabytes of data per day, with the largest instances managing exabyte-scale datasets and billions of objects.

PVLDB Reference Format:
Armbrust et al. Delta Lake: High-Performance ACID Table Storage over Cloud Object Stores. *PVLDB*, 13(12): 3411-3424, 2020.
DOI: https://doi.org/10.14778/3415478.3415560

1. INTRODUCTION

Cloud object stores such as Amazon S3 [4] and Azure Blob Storage [17] have become some of the largest and most widely used storage systems on the planet, holding exabytes of data for millions of customers [46]. Apart from the traditional advantages of clouds services, such as pay-as-you-go billing, economies of scale, and expert management [15], cloud object stores are especially attractive because they allow users to scale computing and storage resources separately: for example, a user can store a petabyte of data but only run a cluster to execute a query over it for a few hours.

As a result, many organizations now use cloud object stores to manage large structured datasets in data warehouses and data lakes.

Proceedings of the VLDB Endowment, Vol. 13, No. 12
ISSN 2150-8097.
DOI: https://doi.org/10.14778/3415478.3415560

The major open source "big data" systems, including Apache Spark, Hive and Presto [45, 52, 42], support reading and writing to cloud object stores using file formats such as Apache Parquet and ORC [13, 12]. Commercial services including AWS Athena, Google BigQuery and Redshift Spectrum [1, 29, 39] can also query directly against these systems and these open file formats.

Unfortunately, although many systems support reading and writing to cloud object stores, achieving *performant* and *mutable* table storage over these systems is challenging, making it difficult to implement data warehousing capabilities over them. Unlike distributed filesystems such as HDFS [5], or custom storage engines in a DBMS, most cloud object stores are merely key-value stores, with no cross-key consistency guarantees. Their performance characteristics also differ greatly from distributed filesystems and require special care.

The most common way to store relational datasets in cloud object stores is using columnar file formats such as Parquet and ORC, where each table is stored as a set of objects (Parquet or ORC "files"), possibly clustered into "partitions" by some fields (e.g., a separate set of objects for each date) [45]. This approach can offer acceptable performance for scan workloads as long as the object files are moderately large. However, it creates both *correctness* and *performance* challenges for more complex workloads. First, because multi-object updates are not atomic, there is no isolation between queries: for example, if a query needs to update multiple objects in the table (e.g., remove the records about one user across all the table's Parquet files), readers will see partial updates as the query updates each object individually. Rolling back writes is also difficult: if an update query crashes, the table is in a corrupted state. Second, for large tables with millions of objects, metadata operations are expensive. For example, Parquet files include footers with min/max statistics that can be used to skip reading them in selective queries. Reading such a footer on HDFS might take a few milliseconds, but the latency of cloud object stores is so much higher that these data skipping checks can take longer than the actual query.

In our experience working with cloud customers, these consistency and performance issues create major challenges for enterprise data teams. Most enterprise datasets are continuously updated, so they require a solution for atomic writes; most datasets about users require table-wide updates to implement privacy policies such as GDPR compliance [27]; and even purely internal datasets may require updates to repair incorrect data, incorporate late records, etc. Anecdotally, in the first few years of Databricks' cloud service (2014–2016), around half the support escalations we received were due to data corruption, consistency or performance issues due to cloud storage strategies (e.g., undoing the effect of a crashed update job, or improving the performance of a query that reads tens of thousands of objects).

[VLDB Paper]

Join the community today!

delta.io

go.delta.io/github

go.delta.io/slack

go.delta.io/twitter

Delta Lake Roadmap

Delta Lake project's roadmap is publicly available in this GitHub issue

Key upcoming features

Support for Spark 3.3

Extended Flink Support

Extended Rust support

VISIT: https://delta.io/roadmap/

Join us!

DATA+AI
WORLD TOUR 2022

Destination Lakehouse

IN PERSON + VIRTUAL

SYDNEY | 18 NOV
VIRTUAL | 1 DEC

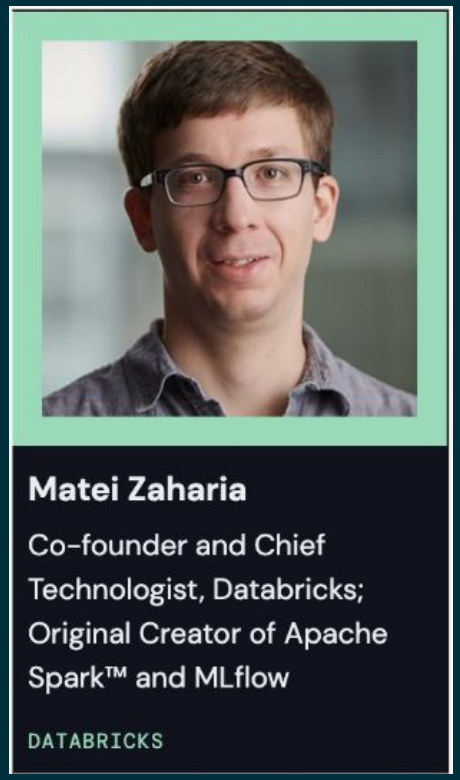
Free to attend

ORGANIZED BY databricks

DATA+AI WORLD TOUR 2022 | Destination Lakehouse

ORGANIZED BY databricks

Coming The Star Sydney 17 & 18 Nov



- SKILL-UP with complimentary training and onsite certification
- EXPLORE the technical and business value of the lakehouse with industry experts featuring Databricks Co-founder, Matei Zaharia
- DEEP DIVE into the latest innovations on the Databricks Lakehouse Platform through practical use cases and real customer scenarios shared by **Atlassian, Alinta Energy, AEMO, Sportsbet**, and many others

Free to attend

/vinoaj

databricks

DELTA LAKE

Delta Lake: Turn your Data Lake into a Lakehouse

Vinoaj (Vinny) Vijeyakumaar
Senior Solutions Architect
vinny@databricks.com | /vinoaj