

Open in app ↗



Search

Write



Unity Catalog — The Information Schema



Ryan Chynoweth

5 min read · Sep 1, 2022



51



1



Introduction

Unity Catalog is a product that enables data access control, data access audit, data lineage, and data discovery. The product is set at the account level and allows administrators to easily manage multiple Databricks workspaces with a single user interface. Now that Unity Catalog is generally available, the following features are also available:

- Support for streaming workloads
- SQL UDFs
- Information schema

In this article we will take a quick look at the information schema and some of the analytics that can be done using it. This likely seems like a boring topic as many data solutions have information schemas, however, Unity

Catalog introduces it for the first time in Databricks. First, we will do a quick overview of the schema then we will provide a handful of example queries. For all code discussed in this blog please reference this [Databricks notebook](#).

For reference please note the hierarchy of objects in Unity Catalog:

- **Metastore:** top-level container for metadata and exposes a three-level namespace (`catalog.schema.table`) to organize your data.
- **Catalog:** the first object layer in the metastore. A metastore can have many catalogs.
- **Schema:** the second layer in the metastore. A catalog can have many schemas.
- **Table:** the last layer in the namespace. Tables can be external or managed and a schema can have many tables.

Overview

The information schema is provided with each catalog in the metastore, with exception to the `hive_metastore` catalog. Each `information_schema` provided will contain information for the catalog that it is associated with.

There is one exception, the `system` catalog returns data pertaining to objects across all catalogs within the metastore. Note that by default the user will only be able to view data on tables that the user is privileged to interact with.

[Databricks documentation](#) provides descriptions for most of the tables available in the `information_schema`. However, a subset of the tables (e.g. the tables related to routines) have yet to be implemented and are reserved for future use.

The following is a full list of datasets available through `information_schema`:

- `catalog_privileges`
- `catalogs`
- `check_constraints`
- `columns`
- `information_schema_catalog_name`
- `referential_constraints`
- `schema_privileges`
- `schemata`
- `table_privileges`
- `tables`
- `views`
- `constraint_table_usage`
- `key_column_usage`
- `routines`
- `routine_privileges`
- `routine_parameters`
- `routine_columns`
- `constraint_column_usage`
- `table_constraints`

Similar to other data analytics solutions, the information schema simply allows organizations to capture and store metadata related to their data assets.

Examples

Columns with a Substring

Ever want to find a column but not sure which table it could reside in? With information schema you can leverage the `columns` table to scan all columns for substrings. Note that there is a function `ilike` that could be used for non-case sensitive string comparison.

```
-- Find all columns in a schema with substring values
-- this is a sql variable with column sub string value
SET var.col_string = datetime ;
-- use the sql variable in a like statement
SELECT * FROM columns WHERE column_name LIKE '%${var.col_string}%'
```

Analyzing Table Relationships

Analytic solutions will often implement referential integrity using primary and foreign key relationships between tables. This solution implements additional data quality checks and connections between different datasets. If you were interested in representing these connections using a table or other visualizations you could use the following query:

```
CREATE OR REPLACE VIEW table_relationships_view
AS
SELECT DISTINCT ptc.table_catalog as p_table_catalog -- table 1
, ptc.table_schema as p_table_schema
, ptc.table_name as p_table_name
```

```

, stc.table_catalog as s_table_catalog -- table 2
, stc.table_schema as s_table_schema
, stc.table_name as s_table_name
, rc.constraint_name -- foreign key
, pccu.column_name as constraint_column_name
, rc.unique_constraint_name -- primary key
, sccu.column_name as unique_constraint_column_name

FROM referential_constraints rc
INNER JOIN table_constraints ptc on ptc.constraint_name =
rc.constraint_name
INNER JOIN table_constraints stc on stc.constraint_name =
rc.unique_constraint_name
INNER JOIN constraint_column_usage pccu on pccu.constraint_name =
rc.constraint_name
INNER JOIN constraint_column_usage sccu on sccu.constraint_name =
rc.unique_constraint_name
;

SELECT * FROM table_relationships_view;

```

Using a little bit of Python code, I can then visualize this dataset as a network with the following code:

```

import matplotlib.pyplot as plt
import subprocess

def install(name):
    subprocess.call(['pip', 'install', name])

install('networkx')

import networkx as nx

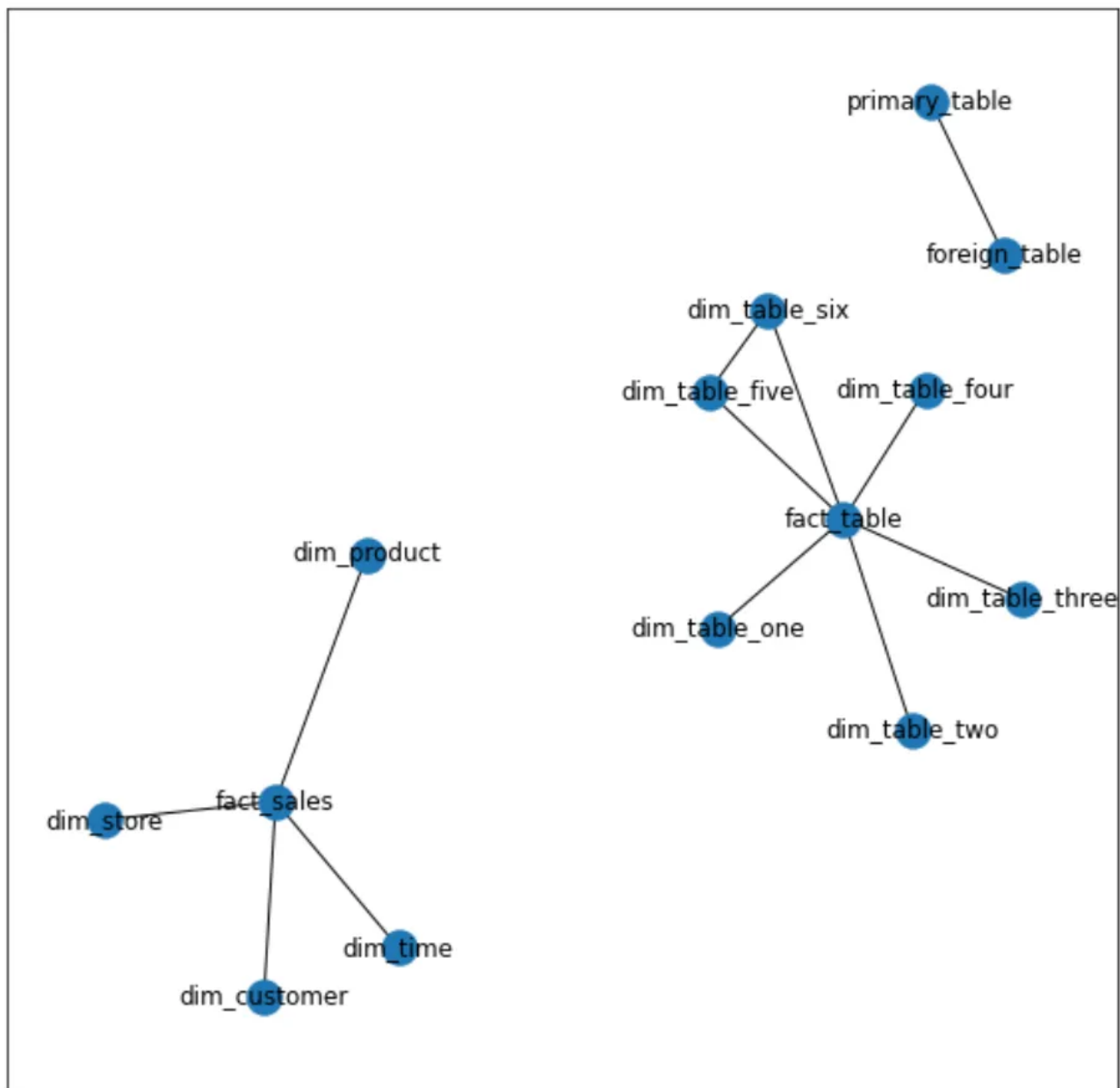
df = spark.sql("SELECT * FROM
rac_demo_catalog.rac_demo_db.table_relationships_view")
pdf = df.toPandas()

G = nx.from_pandas_edgelist(pdf,
                           source = 'p_table_name',
                           target = 's_table_name'
                           )

plt.figure(3,figsize=(10,10))
pos=nx.spring_layout(G,k=0.5)
nx.draw_networkx(G,pos)

```

Which outputs the image below:



Sample Table Relationship Network

Column Counts by Table

Do you have any tables that are ridiculously wide that you want to breakdown into multiple, more narrow, tables? Identify the tables that are the widest using the following:

```
SELECT table_catalog, table_name, table_schema, count(1) as
num_columns

FROM system.information_schema.columns

GROUP BY table_name, table_schema, table_catalog

ORDER BY 4 desc
```

Table Count by Schema

Which schema has the most tables? Oftentimes, more schema objects means more management and maintenance requirements. Seeing which schemas require more time from your data engineers is important to understand the return of investment by providing that data to end users.

```
SELECT table_catalog, table_schema, count(1) as num_tables
FROM system.information_schema.tables
-- WHERE table_schema = 'my_schema'
GROUP BY table_catalog, table_schema
ORDER BY 3 DESC;
```

Table Analysis

Let's assume you are interested in understanding how many of your tables have composite keys.

```
SELECT tc.constraint_catalog
, tc.constraint_schema
, tc.constraint_name
, tc.constraint_type
, tc.table_catalog
, tc.table_schema
, tc.table_name
, count(kcu.column_name) as key_column_count
```

```
FROM table_constraints tc

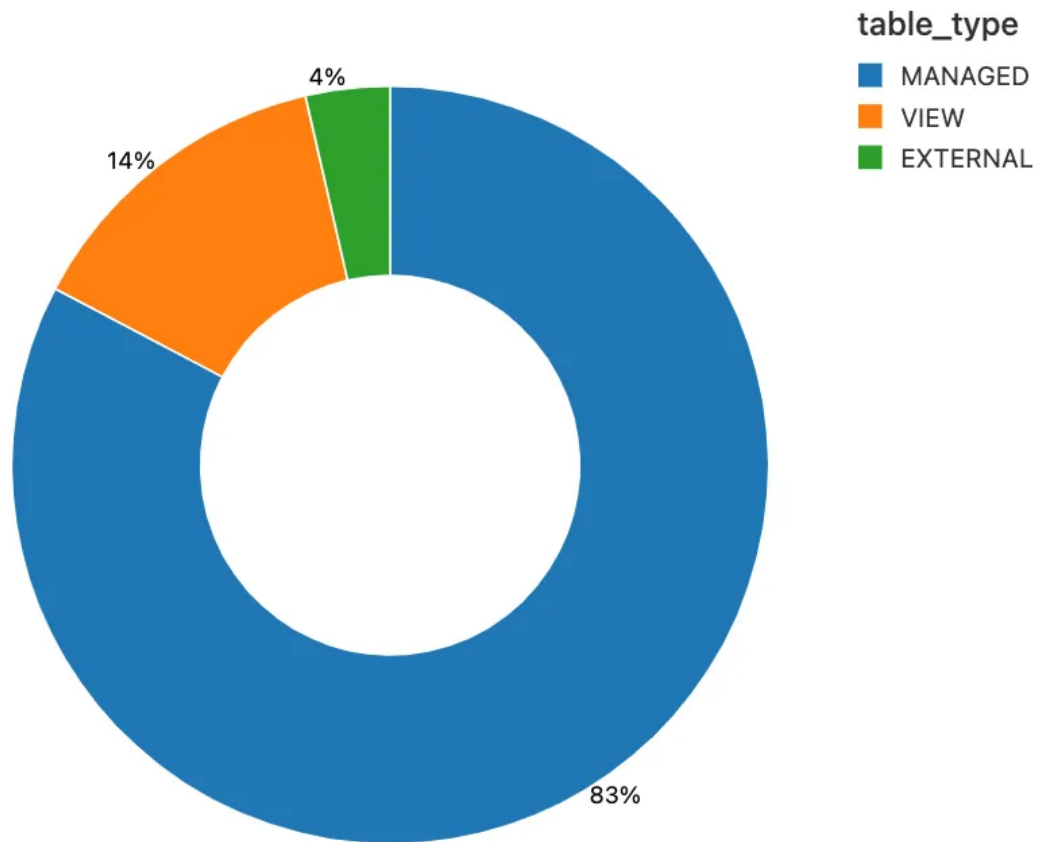
INNER JOIN key_column_usage kcu on tc.constraint_catalog =
kcu.constraint_catalog
        AND tc.constraint_schema = kcu.constraint_schema
        AND tc.constraint_name = kcu.constraint_name

GROUP BY tc.constraint_catalog
, tc.constraint_schema
, tc.constraint_name
, tc.constraint_type
, tc.table_catalog
, tc.table_schema
, tc.table_name

ORDER BY 8 desc
```

What is the breakdown of types of tables?

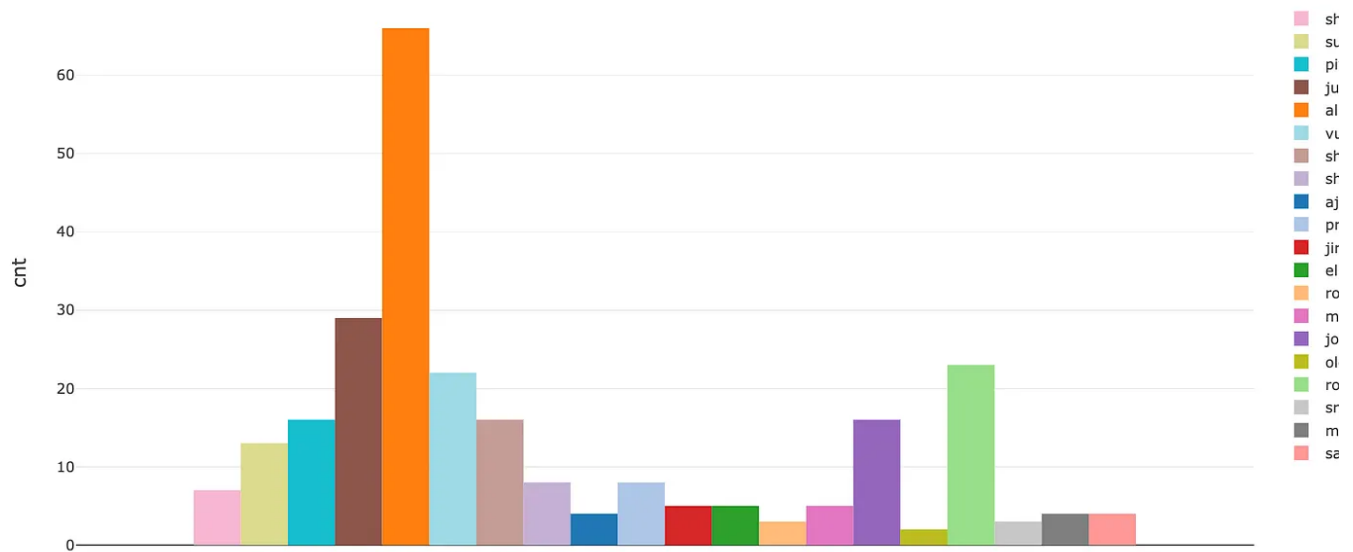
```
SELECT table_catalog, table_schema, table_name, table_type, 1 as cnt
FROM tables
WHERE table_schema != 'information_schema'
```

Pie Chart of the Previous Query

Or wanting to see who is creating tables:

```
SELECT table_catalog, table_schema, table_name, created_by, 1 as cnt
FROM tables
WHERE table_schema != 'information_schema'
```



Bar chart showing tables created by principal

Conclusion

There is a quick overview of `information_schema` in Databricks using Unity Catalog. Let me know of what other information you are interested in finding or any cool data points that you discover on your own!

Disclaimer: these are my own thoughts and opinions and not a reflection of my employer

[Unity Catalog](#)
[Databricks](#)
[Information Schema](#)



Written by Ryan Chynoweth

[Edit profile](#)

312 Followers

Senior Solutions Architect Databricks — anything shared is my own thoughts and opinions

More from Ryan Chynoweth



Open standard for secure data sharing

Industry's first open protocol for secure data sharing, making it easy for organizations regardless of which computing platform they use.

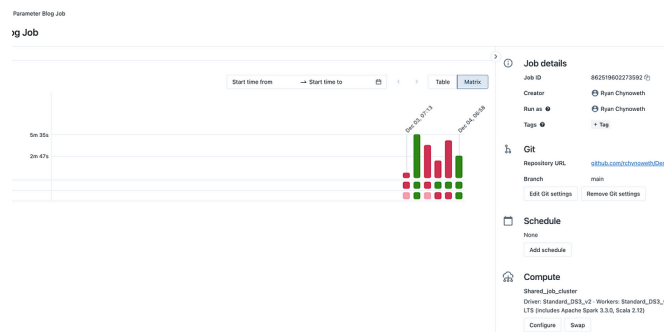


Ryan Chynoweth

Delta Sharing: An Implementation Guide for Multi-Cloud Architecture

Introduction

8 min read · 3 days ago

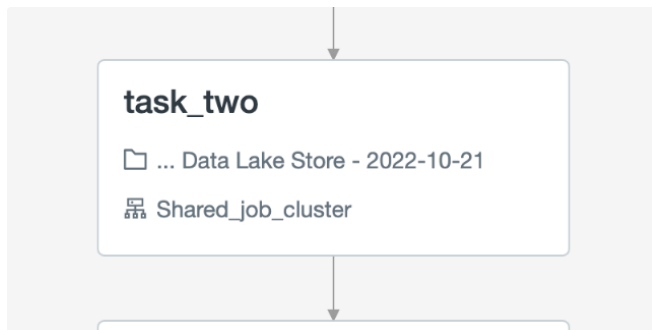


Ryan Chynoweth

Task Parameters and Values in Databricks Workflows

Databricks provides a set of powerful and dynamic orchestration capabilities that are...

11 min read · Dec 7, 2022



 Ryan Chynoweth

Converting Stored Procedures to Databricks

Special thanks to co-author Kyle Hale, Sr. Specialist Solutions Architect at Databricks.

14 min read · Dec 29, 2022



116



5



...



 Ryan Chynoweth

Recursive CTE on Databricks

Introduction

3 min read · Apr 20, 2022



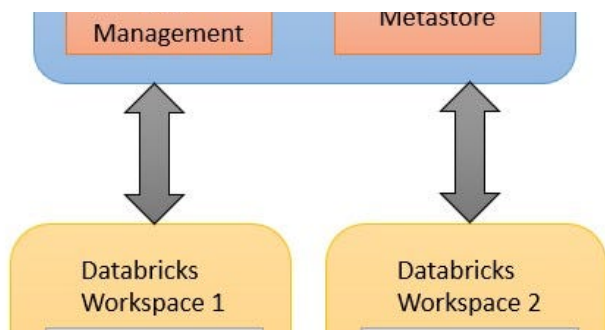
33



...

See all from Ryan Chynoweth

Recommended from Medium





Oindrila Chakraborty

Introduction to “Unity Catalog” in Databricks

What is “Unity Catalog”?

10 min read · Aug 14, 2023



18



2



...



SIRIGIRI HARI KRISHNA

1.0 Unity Catalog

Unity Catalog by Databricks provides a unified governance solution for managing...

2 min read · Nov 28, 2023



1



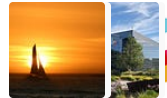
...

Lists



Staff Picks

547 stories · 597 saves



Stories to Help You Level-Up at Work

19 stories · 395 saves



Self-Improvement 101

20 stories · 1146 saves



Productivity 101

20 stories · 1047 saves



Nnaemezue Obi-Eyisi

Unveiling the Secrets: External Tables vs. External Volumes in...

While reviewing the Databricks documentation about Unity Catalog, I came...

🌟 · 7 min read · Sep 25, 2023



Daan Rademaker

Do-it-yourself, building your own Databricks Docker Container

In my previous LinkedIn article, I aimed to persuade you of the numerous advantages o...

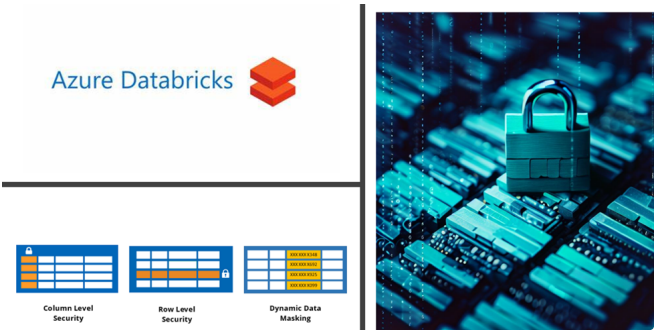
7 min read · Oct 16, 2023


 39  1

 26 



 Samarendra Panda


Dynamic Row Level Filtering and Column Level Masking in Azure...

Background

5 min read · Sep 23, 2023

 29 

 Matt Bradley

Azure Data Factory tips for running Databricks jobs

Following on from an earlier blog around using spot instances with Azure Data...

4 min read · Nov 2, 2023

 14 

See more recommendations