



March 2 - 3, 2022 – Subsurface LIVE Winter 2022

# Powering a Data Mesh with Dynamic GraphQL Schema Generation

Matthew Topol, Principal Software Architect  
Steve Perkins, Lead Software Engineer  
Content Engineering

The screenshot displays the FactSet financial data interface. At the top, there are navigation tabs: Home, Today's Top News, Markets, My Models & Lists, Internal Research, and Advisor Research. Below this is the 'MARKET WATCH' section, which is divided into 'Markets' and 'Financials'. The 'Markets' section shows a table of market indices with columns for Symbol, Last, Chg, % Chg, and YTD %... The 'Financials' section shows a table of financial metrics with columns for Symbol, Last, Chg, % Chg, and YTD %... Below the market watch is the 'QUOTE' section for Apple Inc. (AAPL:USA), showing the current price of 410.99 and various market statistics. At the bottom is the 'NEWS' section, which shows a list of news headlines with columns for Date/Time, Identifier, and Headline.

Symbol	Last	Chg	% Chg	YTD %...
S&P500	3,232.22	14.00	0.43%	0.04%
DJI	28,313.05	-223.92	-0.79%	-7.89%
COMP	10,615.95	27.63	0.26%	16.31%
TSEC-TSE	16,096.77	202.52	1.24%	5.67%
UKX.GB...	5,989.99	141.47	2.31%	20.58%
PXL-ENX	4,783.69	69.25	1.43%	19.98%
DAX-XEX	12,113.36	-66.29	-0.54%	-7.06%
US10Y...	0.55	0.00	0.44%	-71.23%
CL00	40.06	0.14	0.35%	-34.39%
GC00	1,986.80	20.00	1.02%	30.44%
USDCAL...	1.134	0.00	0.22%	3.07%
EURUS...	1.16	0.00	0.37%	5.22%
USDMX...	22.23	0.20	0.93%	17.48%

Symbol	Last	Chg	% Chg	YTD %...
GOOGL	48.81	-0.76	-1.56%	-37.81%
AFL	35.96	0.07	0.20%	33.16%
APO	45.79	-1.75	-3.82%	2.26%
BAM.A.T...	35.88	-2.07	-5.89%	-15.85%
BK	35.72	0.14	0.39%	29.03%
BLK	65.71	0.32	0.49%	12.94%
RY-TSE	92.31	0.93	1.00%	10.16%
BMO-TSE	73.26	1.11	1.49%	27.21%
BNS-TSE	54.99	0.80	1.43%	25.04%
BX	53.09	1.30	2.49%	5.20%
CACC	463.08	20.93	4.32%	4.89%
CB	126.10	3.90	3.00%	18.99%
CM-TSE	92.51	1.08	1.15%	14.39%
CSW	13.12	-0.01	-0.08%	-38.95%
EFC	11.72	0.30	2.50%	36.06%
ESGR	167.22	5.26	3.05%	19.16%
SNEX	51.65	1.76	3.30%	5.78%

Vol	Time	Open	Prev Close	Last Exch	Cvol	30D Avg	Block Qty	Block Vol	Bid	Ask	52 Week High	52 Week Low	Price Year End	YTD %Chg	High	Low	EPS	EPS FY1	PE	PE FY1	Book Val	P/B V
3,184	1:26:55 PM	411.54	384.76	ADF	50,491,679	34,677,740	32	2,695,198	410.94	410.99	412.22	192.58	293.65	-0.01	412.22	403.30	1.00	39.96%	10.16%	39.96%	4.12	4.03

Date/Time	Identifier	Headline
1:20 pm	2222-SA, AAPL...	Twitter - @ChannelNewsAsia: Apple tops Saudi Aramco as most valuable publicly listed company <a href="https://t.co/3p0y3p...">https://t.co/3p0y3p...</a>
1:18 pm	AAPL-US, AAPL...	DJ Apple Stock Hits New Highs: A \$2 Trillion Market Cap Is Within Reach. - Barrons.com
1:16 pm	AAPL-US, FB-U	Street Takeaways - Alphabet Q2 Earnings (\$1462.93, -75.44)
1:13 pm	3711-TW, AAPL...	Twitter - @DigtimesDotCom: ASE reportedly grants major SIP orders for new Apple Watch #accelerometer #Apple #A...
1:12 pm	AAPL-US, AMZ...	Wall Street: debate, reggie Nasdaq
12:59 pm	AAPL-US	Twitter - @dagensindustri: Efter rapporteringen - Apple återigen världens största börsbolag <a href="https://t.co/5lyXPF11S">https://t.co/5lyXPF11S</a>
12:54 pm	NVDA-US, NVD...	DJ Nvidia Should Follow Apple's Stock-Split Idea - Market Talk
12:54 pm	AAPL-US, NVD...	DJ Nvidia Should Follow Apple's Stock-Split Idea - Market Talk
12:49 pm	AAPL-US, AMZ...	Midday Report: Wall Street Lower as Apple, Facebook Hit Record Highs

## Agenda

- Why would you need this? (What were we trying to solve?)
- What about Federation?
- The Tech We're Using
- Flexibility Through Metadata
- Benefits of Dynamic Schema Generation
- Future Optimizations and Tooling
- Q&A

## Why would you need this?

### Consistent New Data Points

The data points that product developers want to expose is continuously expanding. Onboarding new data points needs to be quick and painless.

### Multiple Internal Consumers

Not only do different internal consumers exist for this data, but their query patterns also vary. Fetching the data needs to be straightforward and not require large changes on the consumer side. Data also needs to be consistent across all consumers



### Multiple Different Data Stores

The data that we need to expose is stored in multiple different technologies (postgres, MSSQL, parquet) and can not currently be migrated to something consistent.

### Non-Standard Schemas

The schemas for the various data stores are non-consistent. To ease data fetches for consumers, these should be standardized.

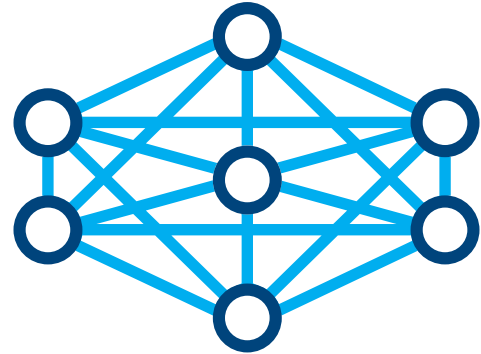
## Additional Considerations

- Abstraction from data source
  - Hide the data mesh complexity
- Data consistency and timeliness
- Cross dataset queries
- Performance
  - Query performance
  - Data transfer performance



## Why didn't we create a Federated GraphQL Service? (It's all the rage right now)

- A Federated Schema != A Standard Schema
- Maintaining multiple different services amongst various teams proved difficult over time
- Ensuring conformance updates were implemented across multiple services was not reliable
- Single-entry-point infrastructure allows us to streamline automation, improve development time, and quality assurance
  - Easier to enforce conformance to standards
- Push joins down to query engine
  - More performant





## Primary Tech We're Using



### Dremio

Our source agnostic query engine. Dremio allows us to pull in data from multiple different data stores and fetch data in a performant manner



### GraphQL

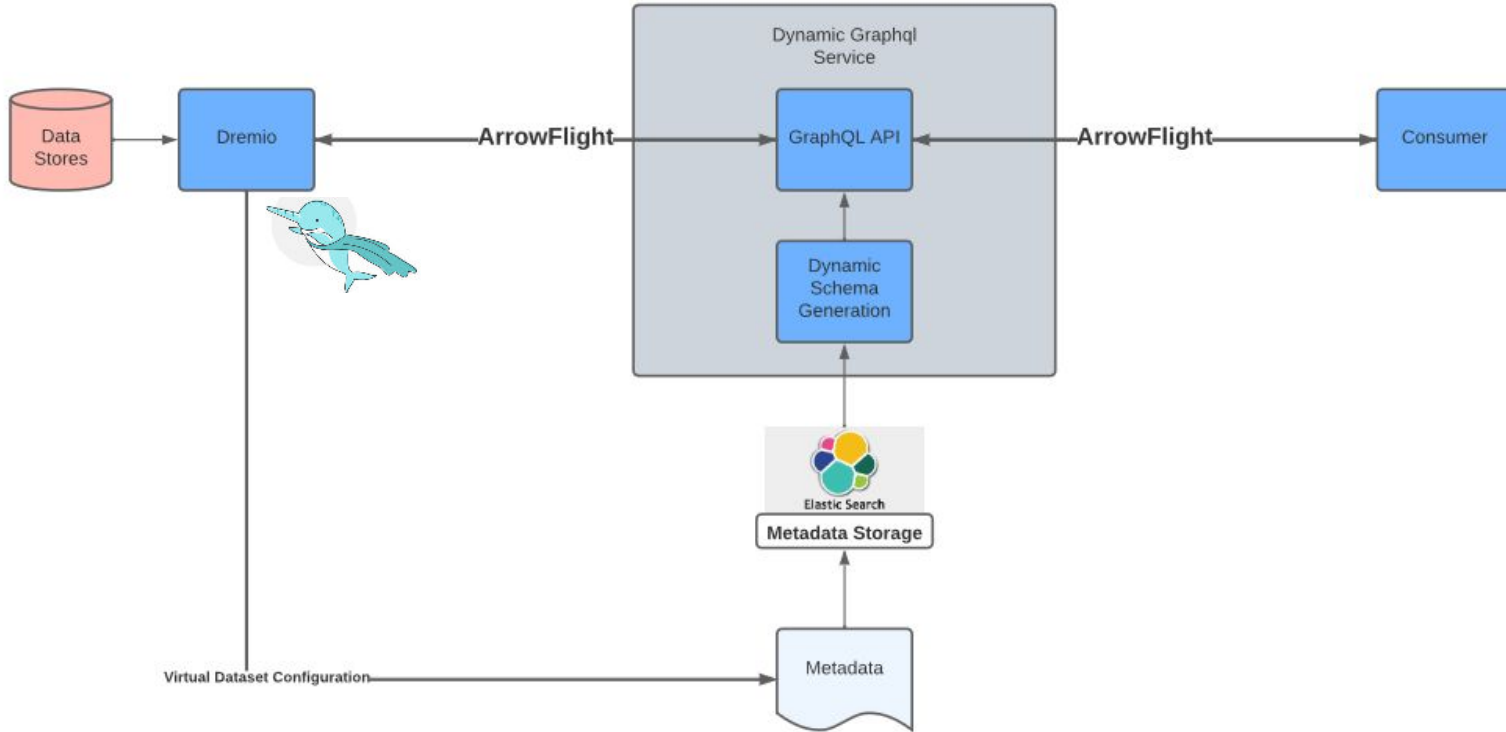
Our flexible schema interface. GraphQL allows us to have a deterministic schema that makes fetching data straightforward and extendable.



### Apache Arrow

Our communication format. In conjunction with Arrow Flight RPC, this communication protocol bolsters round trip performance with zero serialization/deserialization

# High Level Architecture



## Additional Technologies We're Using

- Programming language
  - Golang
- Communication framework
  - Arrow Flight RPC over gRPC
- Cluster Deployment
  - Kubernetes
- Golang Libraries
  - graphql-go-tools - [github.com/jensneuse/graphql-go-tools](https://github.com/jensneuse/graphql-go-tools)
  - Apache Arrow - [github.com/apache/arrow/go/v7/arrow](https://github.com/apache/arrow/go/v7/arrow)



# kubernetes



## Flexibility through Metadata

- Currently our metadata is driven by a fairly simple YAML configuration
  - Split into Entities and Data Items
- Multiple levels of automated schema validation to ensure expectations are met
  - Metadata schema defined using JSON Schema which can be applied to YAML
- Metadata can be owned and manipulated by content teams to provide a self-service API
  - Faster time-to-market for exposing new data!
- Schema generation is flexible enough to ignore unrecognized attributes until they are leveraged
- Deterministic schema generation leads to easily deterministic Query Generation!

Metadata Drives All

## Metadata Example

### Entity

```
entity: Bank
keyField: bank_entity_id
displayTable: bus.banks.banks_entity_display
filterTable: bus.banks.banks_entity_display
caccess: 00FFN,00FFG,OR
periodTypes:
  - name: Annual
    value: A
  - name: Quarterly
    value: Q
  - name: Semi-Annual
    value: S
pivot:
  Deal: bank_entity_id
rel:
  hasMany:
    - Deal
```

### Data Item

```
caccess: 00FFN,00FFG,OR
dataType: FLOAT
entity: Bank
id: venture_capital_revenue
name: Venture Capital Revenue
order: 11
periodData:
  keyField: entity_id
  table: bus.banks.bank_financials_loans_leases
  types:
    - ANN
    - QUARTER
    - SEMI
preferredDecimal: 2
query:
  queryType: FLOAT
result:
  sortable: true
scaling: 1e+06
shortDescription: Venture Capital Revenue
subGroup: Non-Interest Income
type:
  name: item
  parent: group.regulatory_income
unitLabel: USD
```

## Meshing Around With Different Schemas

🔍 Search the docs ...

QUERIES

- bankCounts(...): [bankCounts](#)
- banks(...): [\[Bank\]](#)
- bankAggs(...): [\[BankAggs\]](#)
- toptenants(...): [\[Toptenant\]](#)
- toptenantAggs(...): [\[ToptenantAggs\]](#)
- dealCounts(...): [dealCounts](#)
- deals(...): [\[Deal\]](#)
- dealAggs(...): [\[DealAggs\]](#)
- buyers(...): [\[Buyer\]](#)

```

deals(
  limit: Int
  offset: Int
  where: DealFilterInput
  sort_order: [DealOrderList]
): [Deal]
    
```

TYPE DETAILS

```

type Deal {
  deal_sub_type_description: String
  factset_industry_description: String
  profile: String
  deal_type_description: String
  factset_sector_description: String
  transaction_value: Float
  status: String
  open_date: Date
  ff_emp_num: Int
    
```

**where: DealFilterInput**

TYPE DETAILS

```

type DealFilterInput {
  deal: DealInput
  AND: [DealFilterInput]
  OR: [DealFilterInput]
  NOT: DealFilterInput
  bank: BankInput
  buyer: BigIntFilterInput
}
    
```

**deal: DealInput**

TYPE DETAILS

```

type DealInput {
  category: IntFilterInput
  deal_type_identifier: StringFilterInput
  deal_sub_type_identifier: StringFilterInput
  business_desc: StringFilterInput
  factset_sector: IntFilterInput
  factset_industry: IntFilterInput
  region_code: StringFilterInput
  joined_country_code: StringFilterInput
  joined_state_code: StringFilterInput
  transaction_value: FloatFilterInput
    
```

**status: StringFilterInput**

Filter By Deal Status

TYPE DETAILS

```

type StringFilterInput {
  ne: String
  eq: String
  le: String
  lt: String
  ge: String
  gt: String
  contains: String
  notContains: String
  oneOf: [String]
}
    
```

## Produced GraphQL Schema:

**QUERIES**

- [bankCounts\(...\): bankCounts](#)
- [banks\(...\): \[Bank\]](#)
- [bankAggs\(...\): \[BankAggs\]](#)
- [toptenants\(...\): \[Toptenant\]](#)
- [toptenantAggs\(...\): \[ToptenantAggs\]](#)
- [dealCounts\(...\): dealCounts](#)
- [deals\(...\): \[Deal\]](#)
- [dealAggs\(...\): \[DealAggs\]](#)
- [buyers\(...\): \[Buyer\]](#)

**banks(**  
**limit: Int**  
**offset: Int**  
**where: BankFilterInput**  
**sort\_order: [BankOrderList]**  
**): [Bank]**

TYPE DETAILS

```

type Bank {
  state_name: String
  status_code: String
  num_full_time_emp(...): Float
  bank_name: String
  bank_type_description: String
  ticker: String
  bank_sub_type_description: String
  bank_entity_id: String
  ff_peer_group_detail: String
  ff_int_exp_tot(...): Float

```

**where: BankFilterInput**

TYPE DETAILS

```

type BankFilterInput {
  bank: BankInput
  AND: [BankFilterInput]
  OR: [BankFilterInput]
  NOT: BankFilterInput
  deal: DealInput
}

```

**deal: DealInput**

TYPE DETAILS

```

type DealInput {
  category: IntFilterInput
  deal_type_identifier: StringFilterInput
  deal_sub_type_identifier: StringFilterInput
  business_desc: StringFilterInput
  factset_sector: IntFilterInput
  factset_industry: IntFilterInput
  region_code: StringFilterInput
  joined_country_code: StringFilterInput
  joined_state_code: StringFilterInput
  transaction_value: FloatFilterInput
  status: StringFilterInput

```

**open\_date: DateFilterInput**

Filter By Announcement Date

TYPE DETAILS

```

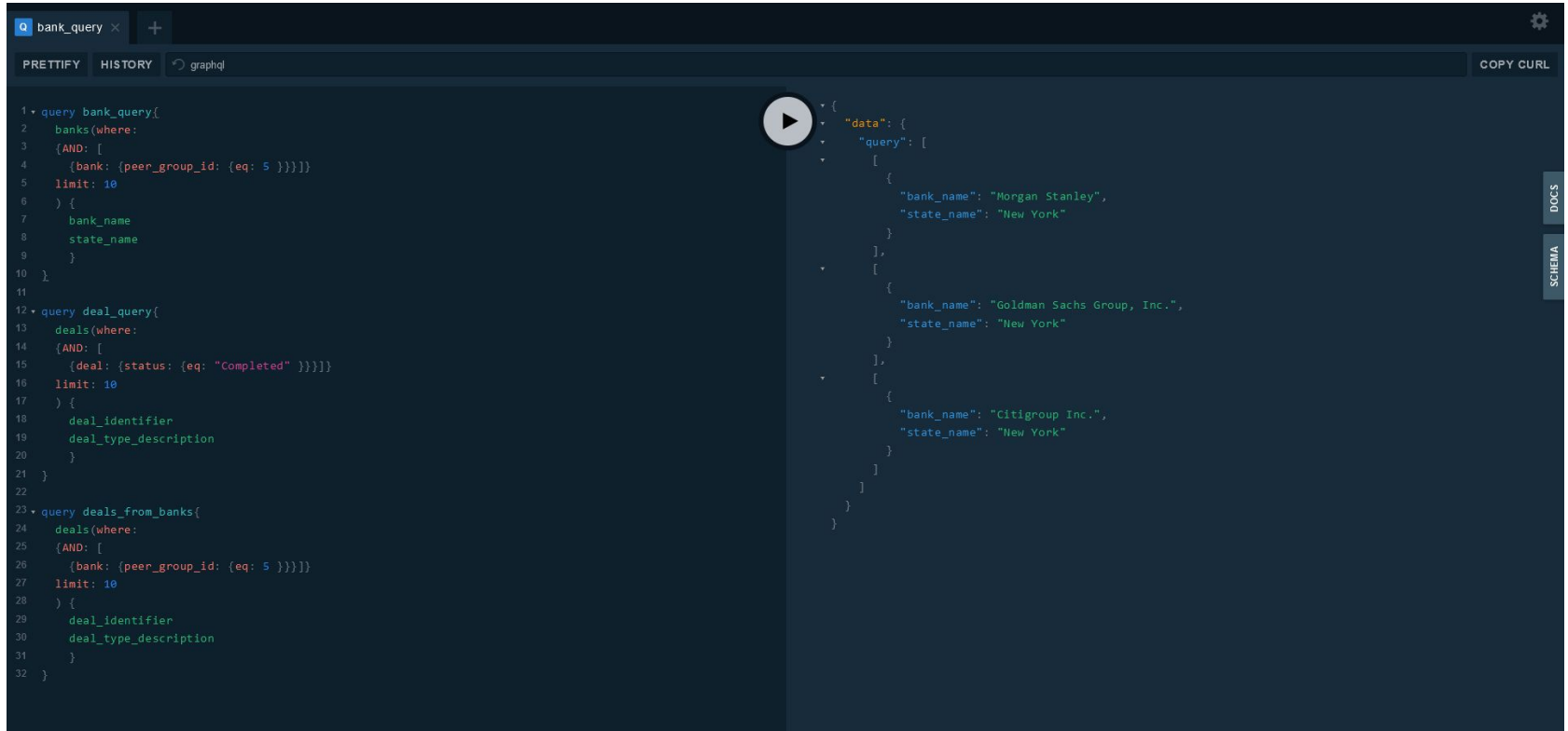
type DateFilterInput {
  ne: Date
  eq: Date
  le: Date
  lt: Date
  ge: Date
  gt: Date
  between: [Date]
  oneOf: [Date]
}

```

Public

Copyright © 2022 FactSet Research Systems Inc. All rights reserved. Confidential: Do not forward. 12

# One Query Pattern To Rule Them All



The image shows a GraphQL IDE interface with a dark theme. On the left, three queries are listed with line numbers 1 through 32. The first query, 'bank\_query', filters banks by peer\_group\_id: {eq: 5} and returns bank\_name and state\_name. The second query, 'deal\_query', filters deals by status: {eq: 'Completed'} and returns deal\_identifier and deal\_type\_description. The third query, 'deals\_from\_banks', filters deals by peer\_group\_id: {eq: 5} and returns deal\_identifier and deal\_type\_description. On the right, the JSON response for the first query is shown, containing an array of three bank objects from New York: Morgan Stanley, Goldman Sachs Group, Inc., and Citigroup Inc. A play button icon is visible in the center of the IDE. The top right corner has a settings gear icon and a 'COPY CURL' button. On the right edge, there are vertical tabs for 'DOCS' and 'SCHEMA'.

```
1 query bank_query{
2   banks(where:
3     {AND: [
4       {bank: {peer_group_id: {eq: 5 }}}}]
5     limit: 10
6   ){
7     bank_name
8     state_name
9   }
10 }
11
12 query deal_query{
13   deals(where:
14     {AND: [
15       {deal: {status: {eq: "Completed" }}}}]
16     limit: 10
17   ){
18     deal_identifier
19     deal_type_description
20   }
21 }
22
23 query deals_from_banks{
24   deals(where:
25     {AND: [
26       {bank: {peer_group_id: {eq: 5 }}}}]
27     limit: 10
28   ){
29     deal_identifier
30     deal_type_description
31   }
32 }
```

```
{
  "data": {
    "query": [
      [
        {
          "bank_name": "Morgan Stanley",
          "state_name": "New York"
        }
      ],
      [
        {
          "bank_name": "Goldman Sachs Group, Inc.",
          "state_name": "New York"
        }
      ],
      [
        {
          "bank_name": "Citigroup Inc.",
          "state_name": "New York"
        }
      ]
    ]
  }
}
```

## Benefits of Dynamic Schema Generation

- Faster time to market for new data points
- Ease of automated extension
  - Define relationships in metadata for easy one-to-many / many-to-many query generation
- Data source abstraction
  - Consumers write GraphQL, not SQL
- Reduced engineering effort necessary for maintenance
- Consistent query patterns allow for templated query generation





## The Query Template

```
SELECT <display_columns>
FROM <display_table> AS t1
LEFT JOIN (
    SELECT <dimension_1_display_columns>
    FROM <dimension_1_display_table>
    WHERE <dimension_1_display_filters>) AS t2 ON <dimension_1_join_fields>
LEFT JOIN ( ... )AS t2 ON <dimension_n_join_fields>
WHERE <display_key> IN (
    SELECT
        DISTINCT <display_key>
    FROM <filter_table> AS t1
    WHERE
        <filter_criteria> AND
        <display_key> IN (
            SELECT DISTINCT <display_key> FROM <dimension_table_1> AS t1 WHERE <dimension_1_filter_criteria>) AND
        <display_key> IN ( ... ))
ORDER BY LOWER(<display_key>) ASC
LIMIT 1000
```

## The Query

```
SELECT "bank_entity_id", "ticker", "bank_name", "bank_sub_type_description",...
FROM bus.banks.banks_entity_display AS t1
LEFT JOIN (...) AS t2 ON (t1."bank_entity_id" = t2."entity_id")
LEFT JOIN (...) AS t3 ON (t1."bank_entity_id" = t3."entity_id")
WHERE t1."bank_entity_id" IN (
  SELECT
    DISTINCT t1."bank_entity_id"
  FROM bus.banks.banks_entity_display AS t1
  WHERE
    ((t1."bank_sub_type" = '0|1' OR t1."bank_type" = '0' OR t1."bank_type" = '1') AND t1."bank_entity_id" IN (
      SELECT DISTINCT "entity_id" FROM bus.banks.bank_financials_loans_leases AS t1 WHERE
        (t1."venture_capital_revenue" >= 10 AND period_type = 'A' AND fiscal_period = 20211231)
    )
  AND t1."status_code" IN ('Active', 'Extinct')) ORDER BY LOWER("bank_entity_id") ASC LIMIT 1000
```

# Dynamic Schemas -> Deterministic Queries

Jobs » 1de16258-f936-aed3-d268-dac831540000
Overview SQL Raw profile

### Summary

Status: COMPLETED

Query Type: Arrow Flight Client (execute prepared statement)

Start Time: 03/01/2022 17:26:46

Duration: <1s

Wait on Client: <1s

User: svc-dremio-screening

Queue: High Cost User Queries

Input: 2.57 MB / 54.474K Records

Output: 134.88 KB / 1K Records

---

**Total Execution Time** <1s (100%)

Metadata Retrieval <1s (36.15%)

Queued <1s (4.23%)

Execution Planning <1s (2.29%)

Starting <1s (0.04%)

Running <1s (41.31%)

Need a hand?

Click "Download Profile" to download the new profile.

Download Profile

### Submitted SQL

```

1 SELECT "bank_entity_id", "ticker", "bank_name", "bank_sub_type_description", "state_name", "assets_20211231_a", "ff_roe_20211231_a", "ff_roa_20211231_a",
2 "net_interest_margin_20211231_a", "npas_20211231_a", "com_eq_t1_cap_ratio_20211231_a", "equity_assets_20211231_a", "loans_deposits_20211231_a"
3 FROM bus.banks.banks_entity_display AS t1
4 LEFT JOIN (
5 SELECT "entity_id", "assets" AS "assets_20211231_a", "net_interest_margin" AS "net_interest_margin_20211231_a", "npas" AS "npas_20211231_a", "com_eq_t1_cap_ratio" AS
6 "com_eq_t1_cap_ratio_20211231_a", "equity_assets" AS "equity_assets_20211231_a", "loans_deposits" AS "loans_deposits_20211231_a"
7 FROM bus.banks.bank_financials_loans_leases
8 WHERE
9   fiscal_period = 20211231 AND period_type = 'A'
10 ) AS t2 ON (t1."bank_entity_id" = t2."entity_id")
                    
```

### Queried Datasets

- banks\_entity\_display (bus.banks.banks\_entity\_display)
- bank\_financials\_loans\_leases (bus.banks.bank\_financials\_loans\_leases)
- bank\_industry\_fundamentals (bus.banks.bank\_industry\_fundamentals)

### Scans

- Raw Reflection (banks\_entity\_display)
- Raw Reflection (bank\_financials\_loans\_leases)
- Raw Reflection (bank\_industry\_fundamentals)

### Acceleration

Query was Accelerated

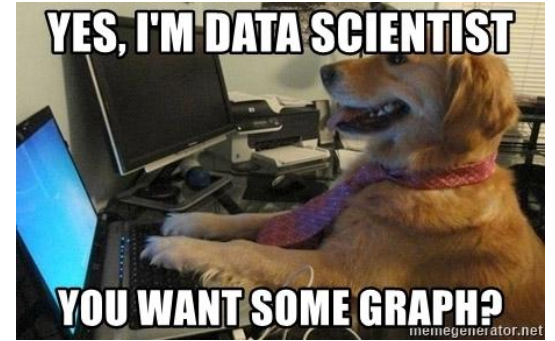
Reflections Used

- Raw Reflection (bus.banks.bank\_financials\_loans\_leases) - Age: 17 hours
- Raw Reflection (bus.banks.banks\_entity\_display) - Age: 17 hours
- Raw Reflection (bus.banks.bank\_industry\_fundamentals) - Age: 5 hours

Reflections Not Used

## Future Optimizations and Tooling

- Improve self-service tooling for adding new metadata
  - Web Platform for internal content teams to use
- More automated metadata generation from datasets
- Higher level GraphQL query planning for utilizing column-caches when available for super-low-latency queries
  - Provide query handling for subset of patterns which can't be served with Dremio SQL
- Computational capabilities in addition to fetches worked into dynamic schema and query generation



Questions?

