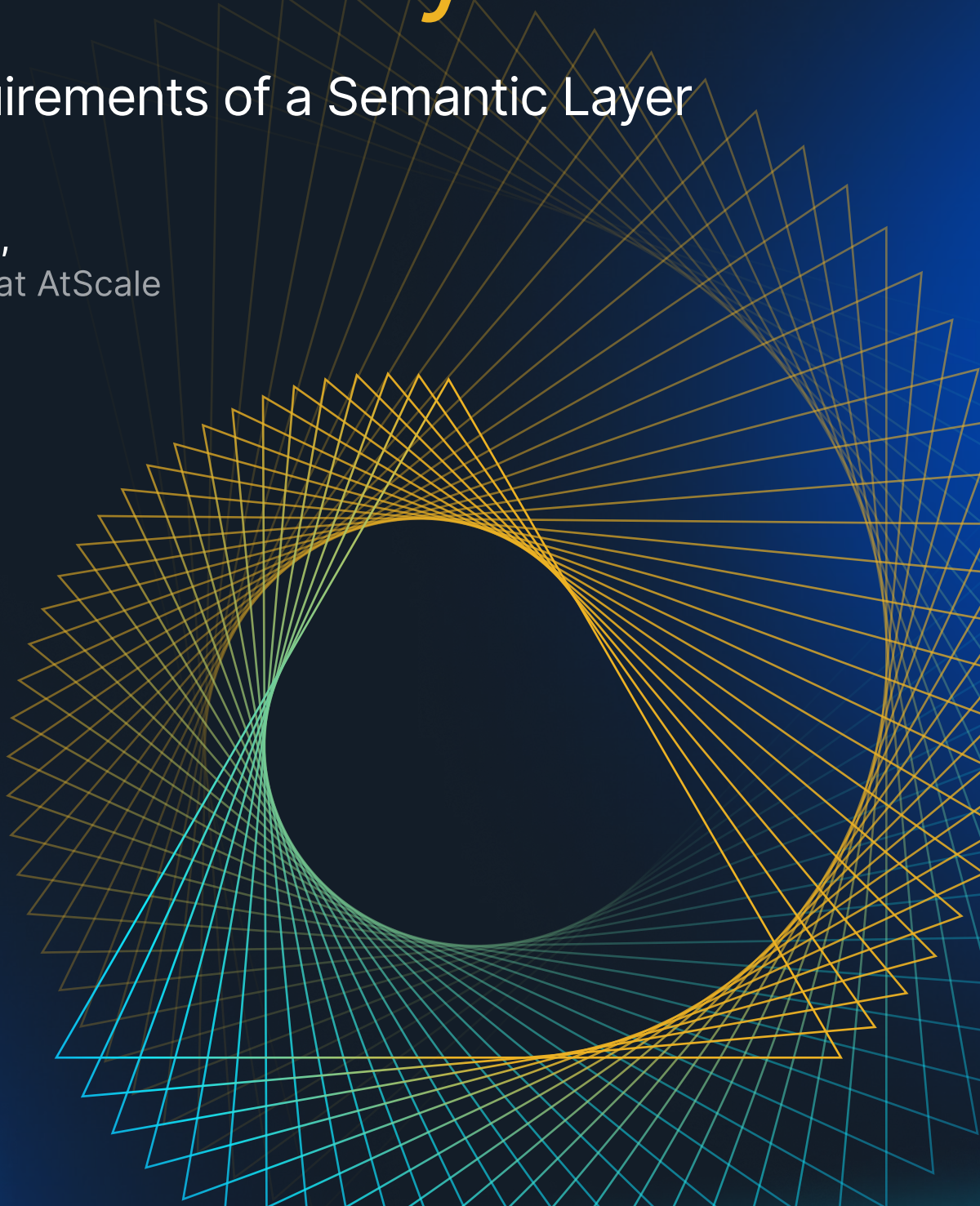
ATSCALE

WHITEPAPER

The Semantics of a Semantic Layer

The 7 Requirements of a Semantic Layer

David P. Mariani,
CTO & Founder at AtScale



Introduction

I co-founded AtScale to focus on the challenges of supporting a large number of data analysts working on disparate sets of data managed in a massive lake. We borrowed the term **“semantic layer”** from the folks at Business Objects who originally coined it in the 1990s. The term was actually over 20 years old when we adopted it.

So what is a semantic layer exactly? If you Google the term, the following definition will pop up, which is a pretty darn good definition in my opinion (Google’s highlighted words, not mine):

> A semantic layer is **a business representation of corporate data that helps end users access data autonomously using common business terms**. A semantic layer maps complex data into familiar business terms such as product, customer, or revenue to offer a unified, consolidated view of data across the organization.
>
> https://en.wikipedia.org › wiki › Semantic_layer
>
> Semantic layer - Wikipedia

Wikipedia defines a semantic layer as a business representation of data that allows end users to access data autonomously. Everyone can agree that a business-friendly view of data that provides users with self-service access to analytics is desirable — true data democratization. It’s easy to see why it is fundamental to scaling data and analytics.

The challenge is actually implementing a semantic layer in a way that just works.

We began building the AtScale semantic layer after working on big data from the trenches. We had to deal with the basic challenges of data scalability, query performance, metrics sprawl, complicated data pipelines and shadow business intelligence (BI). While the challenges seemed obvious to us, most of the industry was preoccupied with shifting data gravity to the cloud. With cloud data re-platforming in full swing, we are finally seeing attention turning to the last mile of enterprise analytics with the semantic layer topic surging in popularity.

The Semantics of a Semantic Layer

Cloud giants like Google and Snowflake, the unicorns like dbt Labs and a host of venture-backed startups are now talking about this critical **new layer** in the data and analytics stack. Some call it a **"metrics layer"**, or a **"metrics hub"** or **"headless BI"**, but most call it a **"semantic layer"**. I can't tell you how happy it makes me that the industry is finally recognizing the importance of the semantic layer in a modern, cloud-first analytics stack. I couldn't agree more that a logical, business-friendly view of data is what's needed to make analytics accessible to everyone, not just data engineers and SQL jockeys.

While it might be just a matter of semantics, I prefer **"semantic layer**" over **"metrics layer"**, **"metrics hub"**, or **"headless BI"**. I think that the term "semantic layer" best describes this business-friendly data interface because it covers all types of data and use cases.

For example, the terms "metric store" and "metric layer" ignore the concept of "dimensions" altogether. Take a look at just about every BI tool on the market (i.e Tableau, Looker, Power BI) and they all include measures (or metrics) *and* dimensions in their interfaces. Metrics *measure* something but dimensions (i.e. "product", "time" and "location") *categorize* data by grouping or aggregating metrics. So, terms using "metric" are confusing and don't map to how these layers will be consumed.

The term "headless BI" is also problematic because it only covers business intelligence use cases. A universal data layer is useful to more than just business analysts and BI. Data scientists need to access a consistent business-friendly interface to data for building and training their models. Furthermore, application developers who are building data-driven applications also need interfaces to data. As such, the term "headless BI" is inadequate because it only covers a single use case: business intelligence.

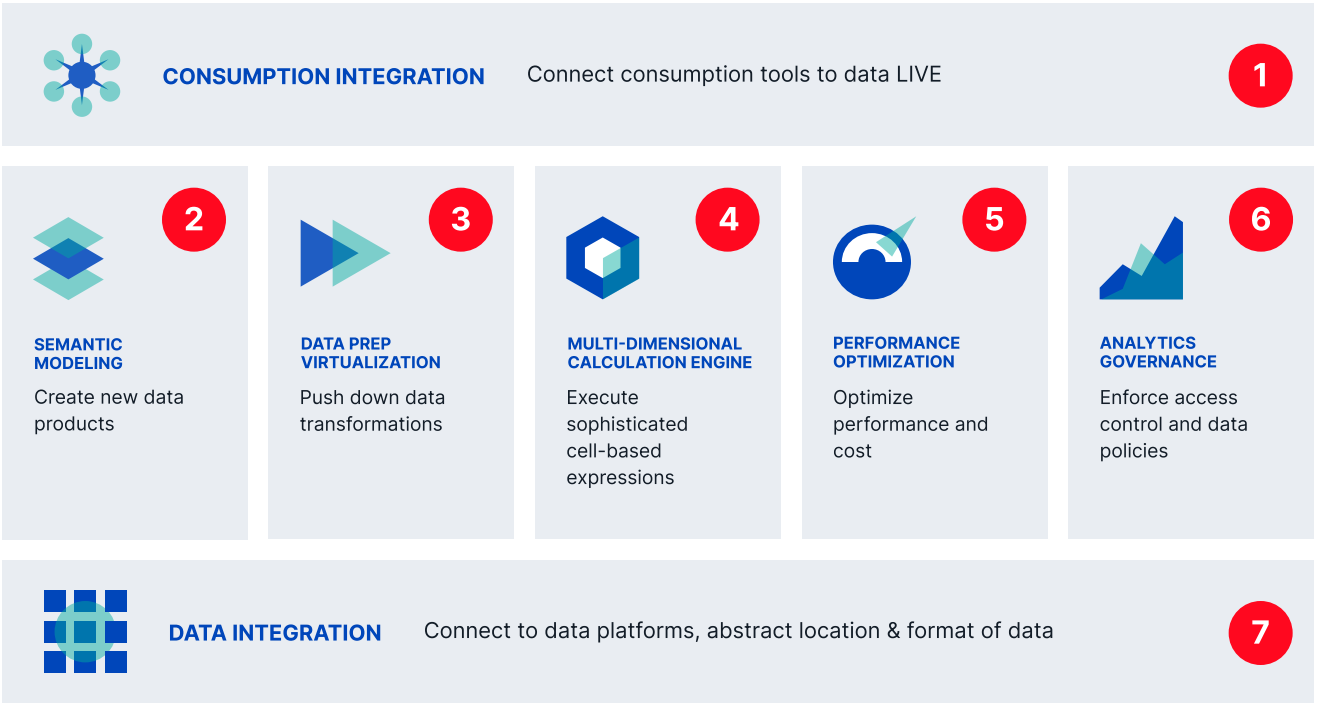
More Than Just Metrics

There's a reason why independent semantic layers have taken time to come to market — building a semantic layer is hard. Yes, a semantic layer serves as a common metrics store or single source of truth, but there's much more to it than that. For a semantic layer to be viable, it needs to:

1. Support any query tool, interface or protocol with a live connection to data
2. Express the most complex business logic (serve as a digital twin) using a semantic model
3. Deliver queries in under 2 seconds
4. Govern access to data for every query
5. Connect to any backend data store

The 7 Requirements of a Semantic Layer

The following diagram illustrates the core capabilities of a semantic layer:



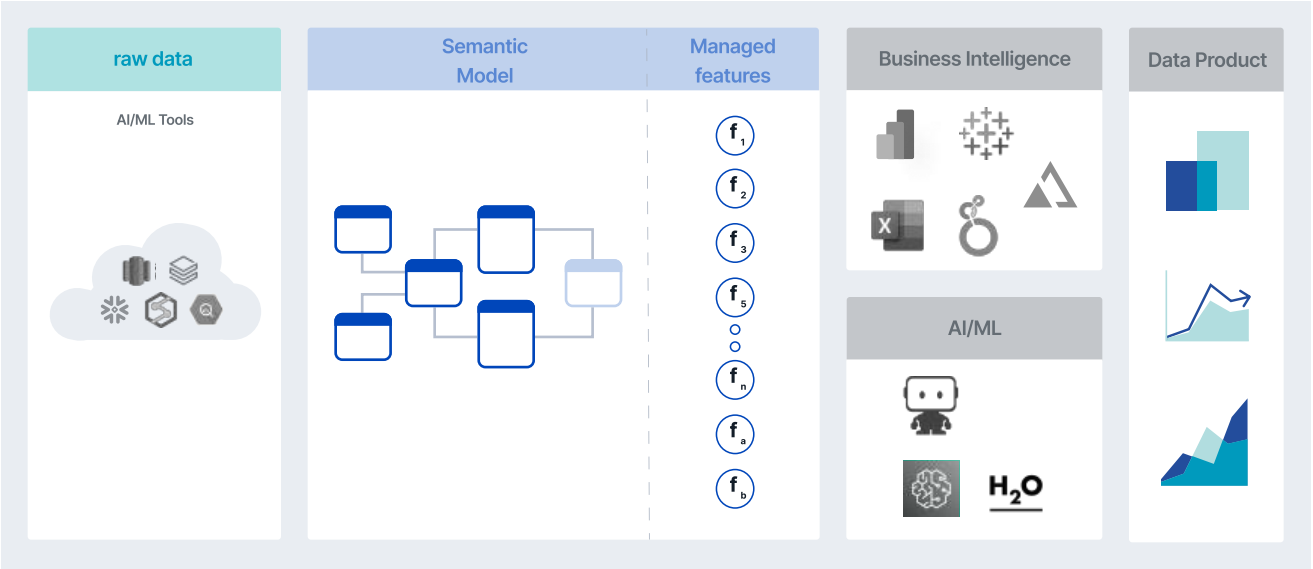
Chapter 1: Consumption Integration

For a semantic layer to be truly universal, it needs to support “live” query connections for all user personas and for all popular query tools and programming interfaces.

More Than Just BI

A universal data layer is useful to more than just the business analyst and business intelligence personas. A semantic layer must also serve the needs of the **data scientist** and **application developer**.

Let’s start with the data science persona. Like business analysts, data scientists also need access to consistent, business-friendly data for building and training their machine learning models. In addition to the ability to *read* (or consume) the semantic layer, data scientists also need to *write* their predictions and features back to the semantic layer. By supporting both reading and writing, the semantic layer and underlying semantic model becomes the bridge that spans the traditional business analytics and data science silos. The image below illustrates how the semantic layer unifies the workflows of the business analyst and data science personas:



In addition to business analysts and data scientists, application developers need simple interfaces into data to build data-driven applications.

By addressing all three personas, a semantic layer can deliver all four flavors of analytics, from *descriptive* and *diagnostic* (business analysts) to *predictive* (data scientist) to prescriptive (data scientist, application developer), becoming the unifying thread underpinning a full range of analysis and personas.

Key Takeaway: A semantic layer must support multiple consumer personas, including business analysts, data scientists and application developers, to deliver the full spectrum of data access and analysis.

More Than Just SQL

SQL was a godsend to database programmers because it became a standard for structured data access for a variety of data platforms. Since a semantic layer exists to provide data access to everyone, not just programmers and data engineers, SQL-only access limits users to tools that speak SQL or those who can write SQL.

While most tools do speak SQL, some tools like Excel (the most popular BI tool on the planet) and Power BI, don't play nice with SQL. Rather, these tools prefer to speak in their native, dimensional dialects using MDX (Excel) and DAX (Power BI). Data scientists prefer to speak to their data using Python and data frames while application developers may prefer using REST, JDBC or ODBC interfaces.

For example, the following queries all answer the same question "how many water bottles did I sell by state in the US?":

SQL (from Tableau)

```
SELECT `Internet Sales`.`CountryCity` AS `countrycity`,
  `Internet Sales`.`Product Name` AS `product_name`,
  `Internet Sales`.`State` AS `state`,
  SUM(`Internet Sales`.`orderquantity1`) AS `sum_orderquantity1_ok`
FROM `sales insights - snowflake`.`internet sales` `Internet Sales`
WHERE ((`Internet Sales`.`CountryCity` = 'United States') AND (`Internet Sales`.`Product
Name` = 'Water Bottle - 30 oz.'))
GROUP BY 1,
  2,
  3
```

MDX (From Excel)

```
SELECT
  NON EMPTY Hierarchize(
    DrilldownMember(
      { { DrilldownLevel(
        { [Geography Dimension].[Geography City].[All] },,,
        INCLUDE_CALC_MEMBERS
      ) } },
      { [Geography Dimension].[Geography City].[CountryCity].& [United States] },,,
      INCLUDE_CALC_MEMBERS
    )
  ) DIMENSION PROPERTIES PARENT_UNIQUE_NAME,
  HIERARCHY_UNIQUE_NAME ON COLUMNS
FROM
  (
    SELECT
      (
        { [Geography Dimension].[Geography City].[CountryCity].& [United States] }
      ) ON COLUMNS
    FROM
      [Internet Sales]
  )
WHERE
  (
    [Product Dimension].[Product Dimension].[Product Line].& [S ].& [S ] & [28].& [477],
    [Measures].[orderquantity1]
  ) CELL PROPERTIES VALUE,
  FORMAT_STRING,
  LANGUAGE,
  BACK_COLOR,
  FORE_COLOR,
  FONT_FLAGS
```

DAX (From Power BI)

```
EVALUATE TOPN(
  1001,
  CALCULATETABLE(
    ADDCOLUMNS(
      KEEPFILTERS(
        ADDCOLUMNS(
          KEEPFILTERS(
            FILTER(
              KEEPFILTERS(
                SUMMARIZE(
                  VALUES('Geography Dimension'),
                    'Geography Dimension' [City.Key0],
                    'Geography Dimension' [City.Key1],
                    'Geography Dimension' [City]
                )
              ),
              NOT(ISBLANK('CubeMeasures' [orderquantity1]))
            )
          ),
          "orderquantity1_City_Key0",
          'CubeMeasures' [orderquantity1]
        )
      ),
      "orderquantity1",
      [orderquantity1_City_Key0]
    ),
    KEEPFILTERS(
      FILTER(
        KEEPFILTERS(
          VALUES('Geography Dimension' [CountryCity.Key0])
        ),
        'Geography Dimension' [CountryCity.Key0] = "United States"
```

```
      )
    )
  ),
  [orderquantity1_City_Key0],
  0,
  'Geography Dimension' [City],
  1,
  'Geography Dimension' [City.Key0],
  1,
  'Geography Dimension' [City.Key1],
  1
)
ORDER BY
  [orderquantity1_City_Key0] DESC,
  'Geography Dimension' [City],
  'Geography Dimension' [City.Key0],
  'Geography Dimension' [City.Key1]
```

As you can see, even though the questions and answers are the same, these tools produce wildly different queries in their native dialect. A semantic layer should handle all these dialects (and more), deliver the same sub-second query performance, apply the same governance filters and, of course, return the same results. For a semantic layer to be universal, it must bring data to its consumers and that means speaking the native language of the end users' tool of choice, whether that be a business analyst, data scientist or application developer.

Key Takeaway: A semantic layer must support multiple inbound languages to support a wide range of data consumers using their preferred protocols. Semantic layer solutions that only support SQL or Javascript are unsuitable to serve as endpoints for a variety of popular consumption tools.

Zero Footprint

A semantic layer can't deliver on its full potential if it's not accessible and usable by everyone. In order to reach the largest number of users, a semantic layer shouldn't require additional client-side software to make it work. This is harder than it seems because custom drivers or plug-ins are usually required to make query tools and applications work with most data platforms.

A well-designed semantic layer will leverage each query tool's built-in connectivity for accessing the semantic layer. For example, AtScale uses the built-in SQL Server Analysis Services (SSAS) drivers in Excel and Power BI to connect to the AtScale semantic layer. This means that anyone with Excel can connect "live" to the AtScale semantic layer without any added software requirements

Key Takeaway: A semantic layer should not require the IT team to install additional client software on query consumers' machines.

Data for Everyone

Besides serving as a metrics hub, a semantic layer provides a business-friendly interface to data for all consumer personas, tools and dialects. A semantic layer truly democratizes data access by turning everyone into a data-driven decision-maker. In my next post, part three of seven, we'll dive into the semantic layer's **semantic data model** for mapping your digital assets to the business.

Chapter 2: Semantic Modeling

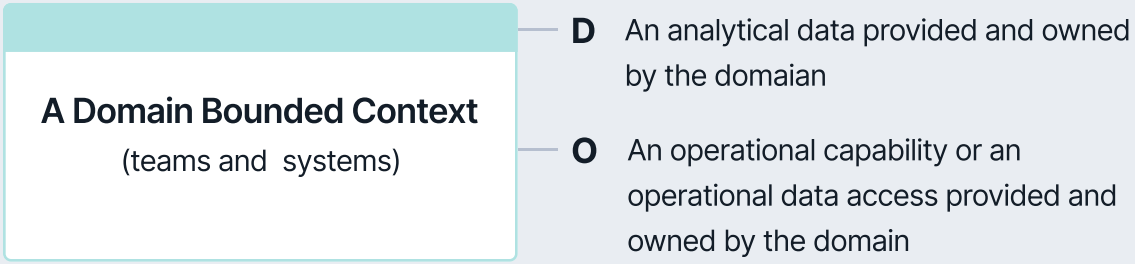
For a semantic layer to function, it must map the physical data objects to the logical business constructs, creating a digital twin of the business while serving as the graph-based query planner and optimizer.

Write Once, Re-Use Many

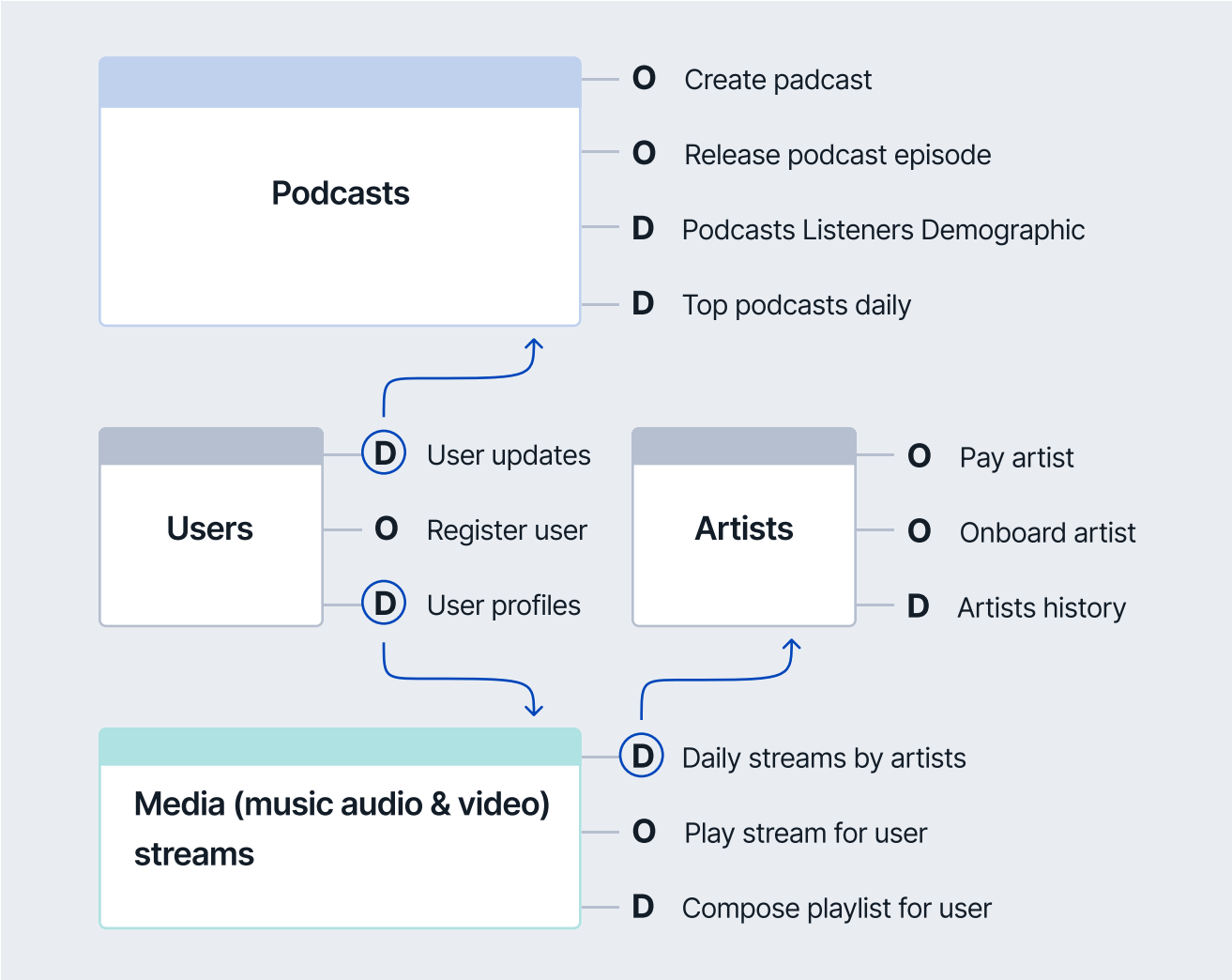
The days of centrally managed, monolithic data pipelines are over. Data moves and changes too fast for a single team to keep up with the demands of the business. At the other end of the spectrum, business users creating their own data pipelines, also proved problematic. After all, business users aren't data engineers, but they are business domain experts.

New approaches, like the data mesh or a hub and spoke model, seek to create a modern, distributed architecture for analytical data management that alleviates the traditional bottlenecks while putting business definitions in the hands of business domain experts.

The illustrations below show an example of how domain-oriented data ownership works using a data mesh approach to analytics data management.



Source: Data Mesh Principles and Logical Architecture, Zhamak Dehghani



Source: Data Mesh Principles and Logical Architecture, Zhamak Dehghani

The semantic data model is a key component for delivering this decentralized strategy for analytics data management because it gives the domain experts the environment to create their data products and share them across the organization. With a semantic data modeling platform that promotes object-oriented model definitions, reusability and sharing, business domain experts or data engineers can free themselves of manual data engineering tasks and combine models and components to create new analytics products. To support sharing and reuse, a semantic data platform must support the role-based security and sharing of model components for creating data products from multiple data domains.

Key Takeaway: A semantic data model must support a hub and spoke style for creating data products using an object-oriented data modeling language that allows subject matter experts to own and share data model components across teams.

Server-side Data Blending

Data was already siloed before the cloud revolution, but with the proliferation of cloud data lakes, cloud data warehouses and SaaS applications, data lives in more places than ever before. A web of proprietary and incompatible data APIs makes the matter even worse for users looking to create data products.

A semantic layer platform can break down these data silos and abstract away the location and format of the data. At the same time, modelers can create mashups of the data that span multiple locales and platforms to create new, composite data products that can serve as the digital twin of the business. By creating these logical views server-side instead of in the consumption tools, these blended data views can be shared across multiple user personas, whether it's business analysts, data scientists, or application developers

Key Takeaway: A semantic data model must break down data silos by blending data sources on the service side to create rich, composite views across multiple business domains.

Graphical & CI/CD Friendly

The people who understand their business domain and the data that feeds it are usually the best authors of the semantic data model. By leveraging their knowledge of the physical (how the data is stored and structured) and the logical (business terms and calculations), they can create data products that are consumable by a wide range of downstream users.

The modeler persona is not always a business analyst, though. Sometimes, a data engineer is the best person to own a particular data domain and thereby own the semantic data model. In order to support both types of model authors, business users and data engineers, a data modeling platform must support both visual and code-based model definitions. In order to support a decentralized analytics data management style, CI/CD is critical to supporting a scalable workflow to enable multiple data domain owners and model contributors.

Key Takeaway: A semantic later platform should support both code-based and graphical data modeling to allow engineers and non-engineers alike to build and collaborate on data models and data modeling components

The Power of the Semantic Model

Besides serving as a metrics hub, a semantic layer — powered by a semantic data model — creates a digital twin of your business. By harnessing the power of the subject matter expert, the semantic data model serves as the rosetta stone for the enterprise, providing a business-friendly access layer to data for everyone.

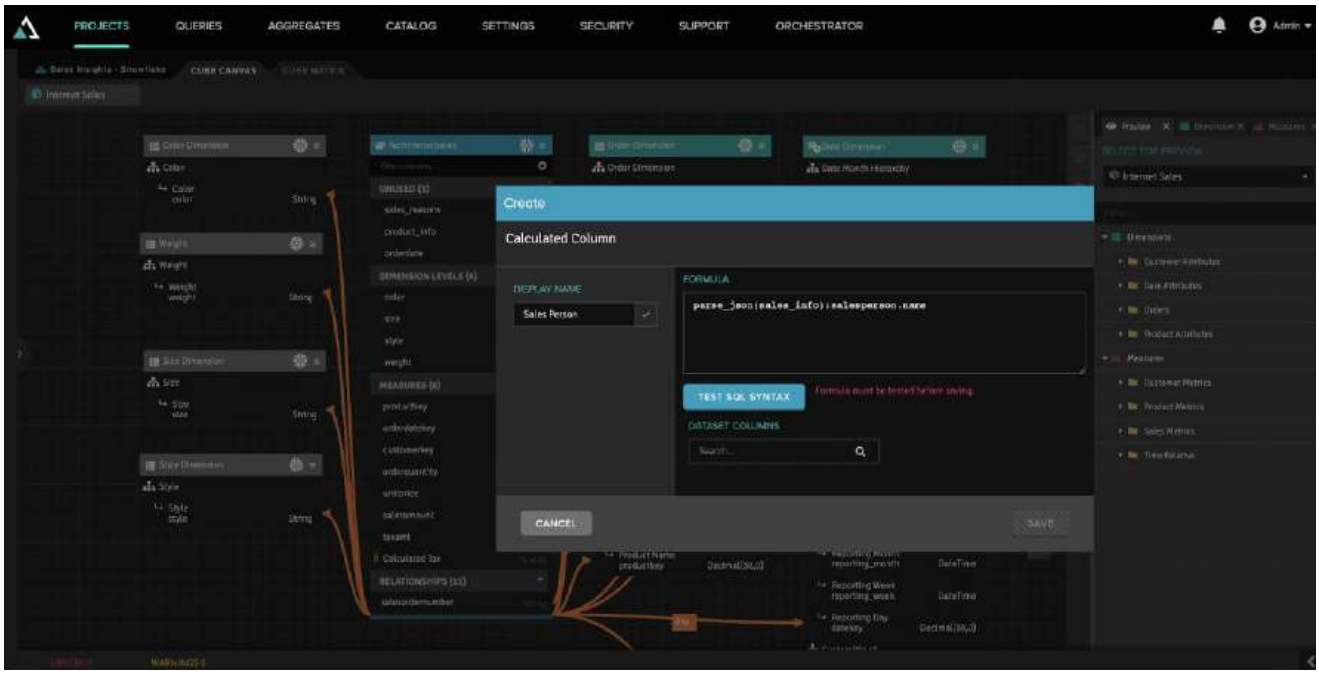
Chapter 3: Data Prep Virtualization

Data transformations are a necessary requirement for a semantic layer. A semantic layer platform should support virtualized calculations for expressing business logic and be capable of generating multi-pass SQL queries to handle calculations that require different levels of granularity like ratios and weighted averages.

Openness

Data platforms, especially cloud data platforms, have come a long way from their origins as relational databases. Modern cloud data platforms allow customers to use custom SQL to store and access nested, semi-structured data and extend their platforms' functionality with user-defined and custom aggregation functions. With a near constant flow of new, powerful functionality, it's critically important that a semantic layer platform supports native data platform dialects for defining data transformation rules, calculations and expressions. By passing these expressions down to the underlying data platforms for execution, customers can enjoy the data platforms' full range of capabilities and scale their data transformations by keeping them close to the data, server-side.

In the example below, we are using the native Snowflake PARSE_JSON function to find the "Sales Person" from a column containing JSON data about a sale called "SALES_INFO":



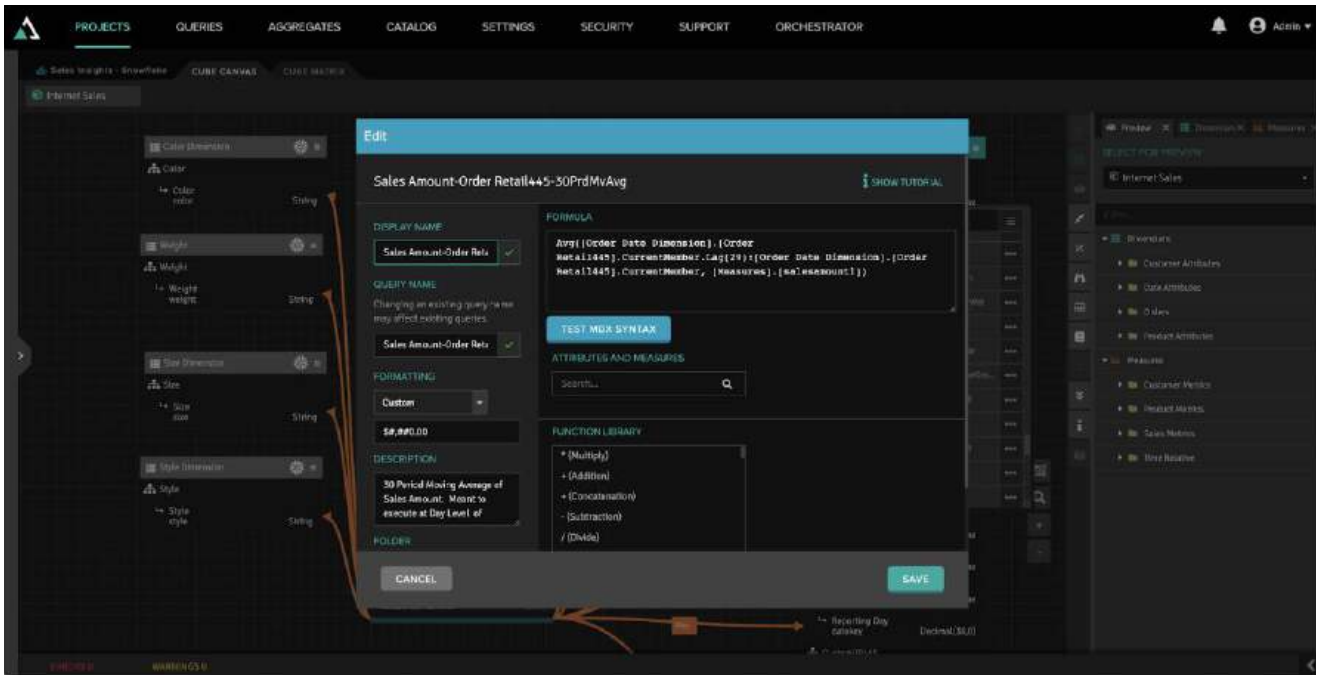
By taking advantage of the data platform's native functionality, data modelers can leverage the syntax of their underlying data platform without needing to learn another language.

Key Takeaway: A semantic data model must support data transformation expressions in the semantic data model using the native platform's SQL dialect.

Multipass

Sometimes a single-pass formula is not expressive enough to support calculations that combine data at different levels of granularity. For example, a weighted average calculation like "Average Interest Rate" or a ratio calculation like "Sales per Order" require data to be combined for a numerator and a denominator. In order to calculate these types of expressions, the semantic layer platform needs the ability to aggregate data first and then perform the final calculation on the aggregated results, requiring ordered operations and multi-pass queries.

The Multidimensional Expressions language, or MDX, is ideally suited for supporting these types of expressions. In addition to supporting calculations at different levels of granularity, MDX is perfect for creating time-relative and cell-based expressions like the following formula that calculates a “30 day moving average of Sales” using a Retail 445 calendar:



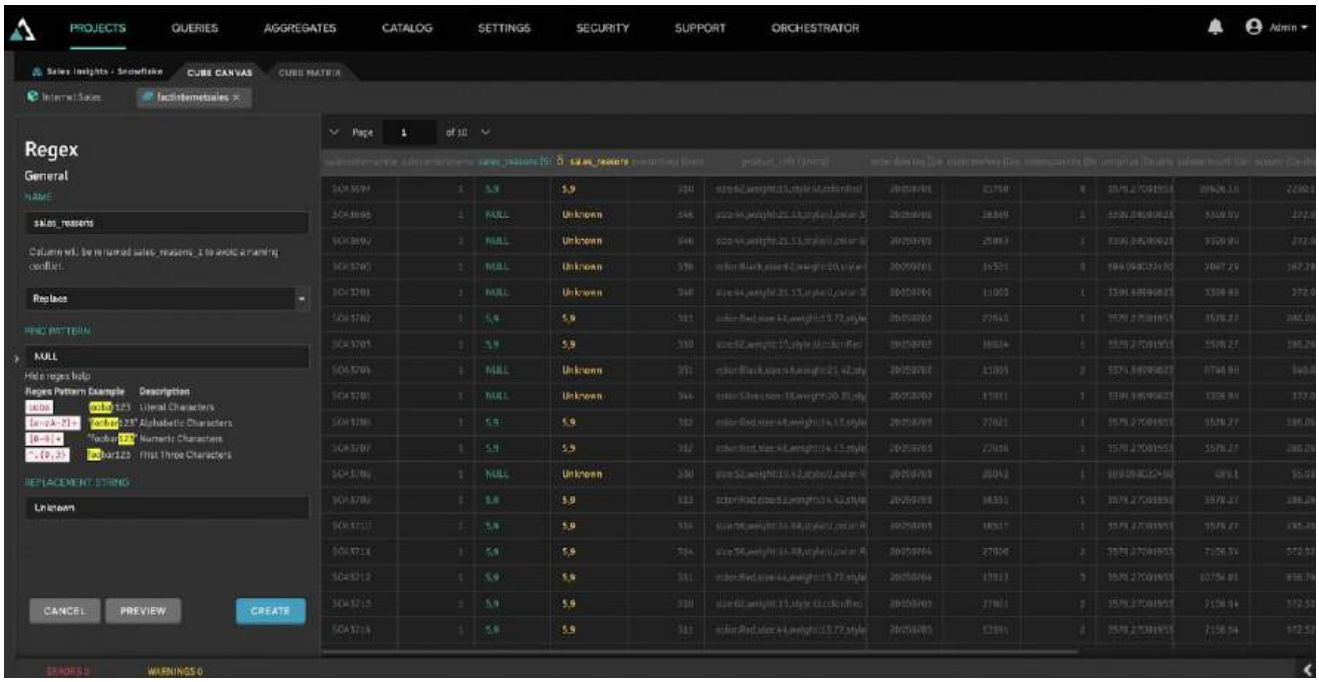
Key Takeaway: A semantic data model must support the ability to perform pre-query & post-query calculations for handling calculations that summarize data at different levels of granularity.

Virtualized

Data wrangling tools and ETL/ELT platforms are familiar to most data engineers. These tools are meant to move data from point A to point B while transforming data for the purposes of cleansing, correcting, combining or just calculating new values. For most use cases, creating new tables or files with transformed data is overkill and adds complexity to data pipelines.

Data virtualization can automate most data transformation tasks without data movement and with the added benefit of leveraging the power of the data platform for performing these transformations. By pushing calculations down to the underlying data platform without physical data movement, subject matter experts can create flexible, documented data transformations without writing complex code.

For example, in the image below, we are “cleaning” the “sales_reasons” field by replacing the NULL values with an “Unknown” string by creating a new virtual column.



Key Takeaway: A semantic later platform should support inline data transformations using direct queries without data movement or creating copies of data.

The Power of Data Prep Virtualization

Besides serving as a metrics hub, a semantic layer is a central repository for business logic and calculations. By supporting complex, multipass data transformation expressions, the semantic data model can take the place of physical ETL data pipelines. By harnessing the power of the subject matter expert, the semantic layer platform can become the digital twin of the business while avoiding dependencies on data engineers and SQL experts.

Chapter 4: Multi-dimensional Calculation Engine

For a semantic layer to function, it must translate the inbound, logical queries coming from the data consumers into physical SQL queries in the dialects of the underlying data platforms.

Cell-Based, Multi-dimensional

The semantic layer data model is ideally suited to define the digital version of the business, but it needs to be capable of expressing a wide variety of business concepts in a variety of contexts.

Tabular and Multidimensional

There are two types of semantic layers, or models, to consider: a tabular semantic layer and a multidimensional semantic layer.

The tabular or relational model was popularized by modeling gurus like EF Codd and Ralph Kimbal in the 70s and 80s. These modeling techniques rely on concepts like fact and dimension tables and are meant to make a relational database or data warehouse easier to query. A tabular view of the data is useful since it presents data as a simple, flatten view of data in rows and columns. This format is ideally suited for SQL-based use cases and tooling using common protocols like JDBC and ODBC.

The multidimensional data model goes one step further, though. By defining relationships and aggregation rules, the multidimensional semantic model adds a business friendly context and makes hand writing SQL either unnecessary or substantially more simplistic. For the widest range of uses and consumption styles, a multidimensional semantic layer offers more power in an easier to use package because it combines business-friendly metadata and data in one interface.

Dimensional Expressions

The relational database, with its row-oriented architecture, is well-suited to store transactional data, but relational databases are not well-suited to express business logic, since the business thinks in multiple dimensions or cells, instead of rows. It's not an accident that the spreadsheet interface like Excel became the dominant tool for modeling business because it is designed for multidimensional analysis. The spreadsheet's cell-based calculation architecture provides a full range of expressiveness that a row-based engine lacks. As such, the semantic layer must be capable of modeling data in cells, not rows.

The multidimensional expression language, or MDX, has hundreds of functions and operators to express a wide variety of cell-based calculations for expressing complex business logic. Let's take a simple example of defining a metric for a 30 day moving average of sales. While this seems like a simple metric, it gets complicated when you wish to use the metric in queries with a variety of time periods (i.e. 30 day moving average by week, month, quarter, year, etc.).

Using a semantic layer with support for MDX, we can define this metric once using the following expression:

```
WITH MEMBER [30 Day Moving Avg] AS
AVG([Order Date Dimension].[Order Retail445].CurrentMember.LAG(30) : [Order Date
Dimension].[Order Retail445].CURRENTMEMBER, [Measures].[Sales])
```

The above expression packs a lot of power in a compact package. In this example, we are using an AVG expression to create an average of the [Sales] metric, a LAG function to specific the lookback period (30 days), and a range expression using a ":" operator and the CURRENTMEMBER expression to dynamically calculate a range of values given a query context.

We can then use this common expression in a variety of queries without regard to its context. In the following example, we computed a 30 day moving average of sales by quarter for All-Purpose Bike Stands:

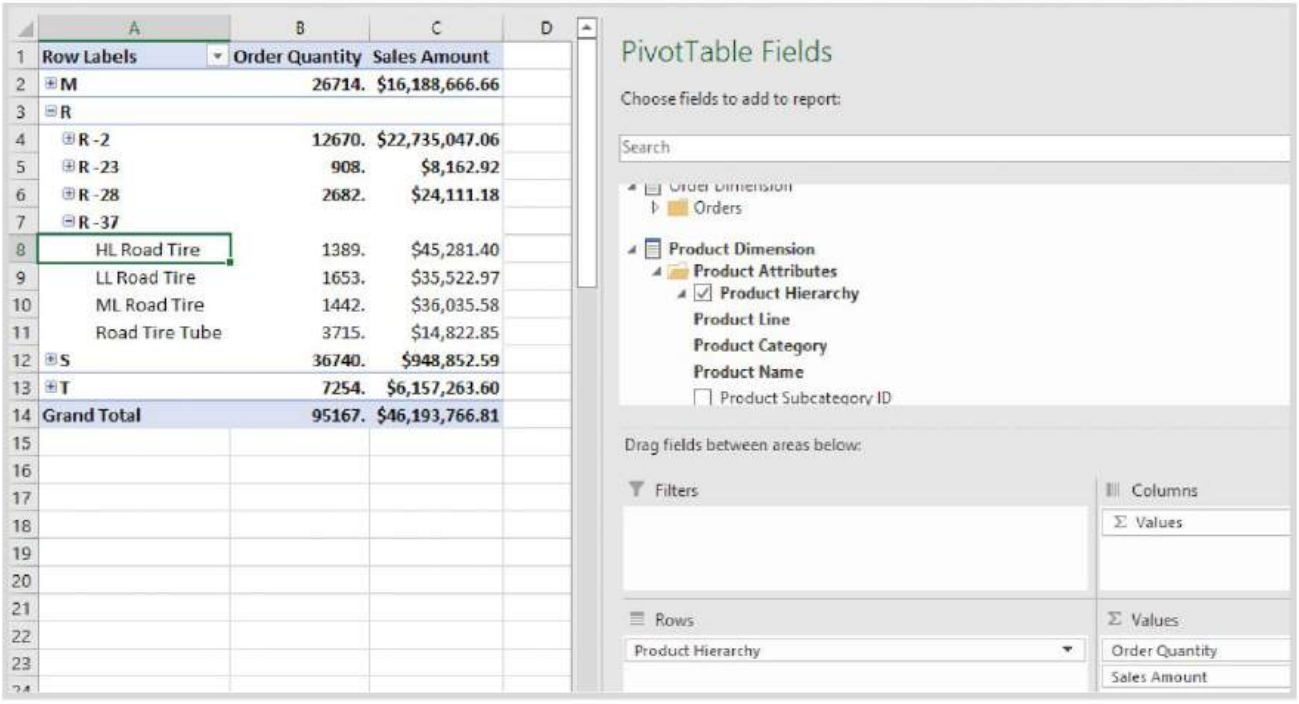
```
SELECT [Order Querter], AVG([30 Day Moving Avg])
FROM [Internet Sales Model]
WHERE ([Product Name] = 'All-Purpose Bike Stand')
GROUP BY 1
```

By simply changing [Order Quarter] to [Order Week] in the above query, the MDX expression would render the correct moving average automatically. Furthermore, changing the filter (All-Purpose Bike Stands) also returns the correct answer without further model modification.

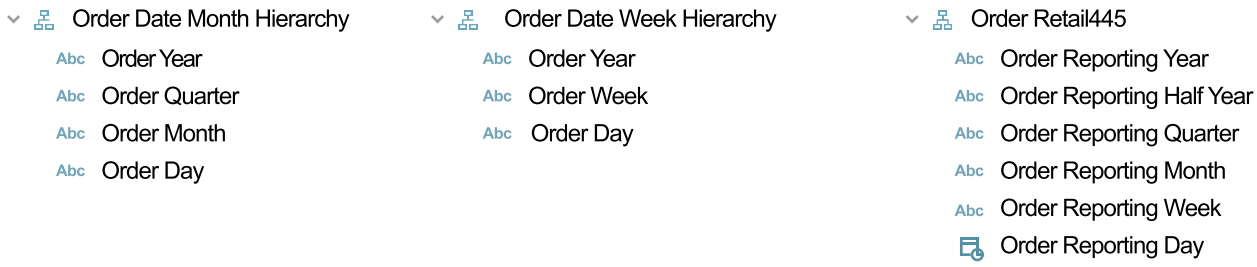
Without a cell-based computation language, a SQL-based semantic layer would require hand coding each query variation and require a separate query definition for each unit of time. Doing so would require loads of custom SQL code that lacks reusability and is tough to maintain, making a SQL-based semantic layer engine a poor choice for expressing even simple business constructs.

Hierarchies

The inventors of OLAP introduced the world to hierarchies in the 80s and it's been a core feature for data visualization ever since. Hierarchies allow users to drill into data along a defined path, moving from less to more detail intuitively. For example, the product hierarchy below allows Excel Pivot Table users to choose their level of product detail and drill down for more granularity with a simple mouse click.



Hierarchies provide more value than just making data visualization more intuitive. Hierarchies also provide easy-to-use alternative representations of the same base data. For example, the date dimension in the model below (as viewed through Tableau) allows users to group order data by Year-Quarter-Month-Day, by Year-Quarter-Week-Day or by reporting period using a 4-4-5 calendar popular with the retail and manufacturing industries.



Without the ability to model hierarchies, end users lose critical functionality when using tools like Tableau, Power BI and Excel which support hierarchical visualizations.

Key Takeaway: A semantic layer must be backed by a multidimensional, cell-based engine to express complex business logic. Semantic layer solutions that use SQL-based calculation engines cannot express business constructs in a variety of contexts. A semantic layer must also support hierarchies to allow for intuitive drill paths and level relationships. Semantic layer solutions that only support dimensions and metrics do not provide an intuitive data navigation experience for end users.

"Anything by Anything"

Trying to guess what questions end users may ask of data is an exercise in futility. The needs of the business often outpace the ability for data teams to respond. It's imperative, therefore, that a semantic layer be as flexible as possible in answering queries.

Recently, at the Coalesce 2021 Conference, Drew Banin, co-founder of dbt Labs, delivered an excellent presentation called "The Metric System". Drew did a great job explaining the value of the semantic layer and he lays out the dbt Labs approach to delivering one.

In his presentation, Drew introduced the following example of a metric model:

```
# models/marts/product/schema.yml

version: 2

models:
 - name: dim_customers
    ...

metrics:
  - name: new_customers
     label: New Customers
     model: dim_customers
     description: "The number of paid customers who are using the product"

     type: count
     sql: user_id # superflous here, but shown as an example

     timestamp: signup_date
     time_grains: [day, week, month]

     dimensions:
         - plan
         - country

     filters:
         - field: is_paying
           value: true

     meta: {}
```

I know this is meant to be a simple, illustrative example, but it (and similar metric layer solutions) demonstrates some serious shortcomings.

In the metric model example above, you will notice the following "filters" directive:

```
filters:
  - field: is_ paying
    value: true
```

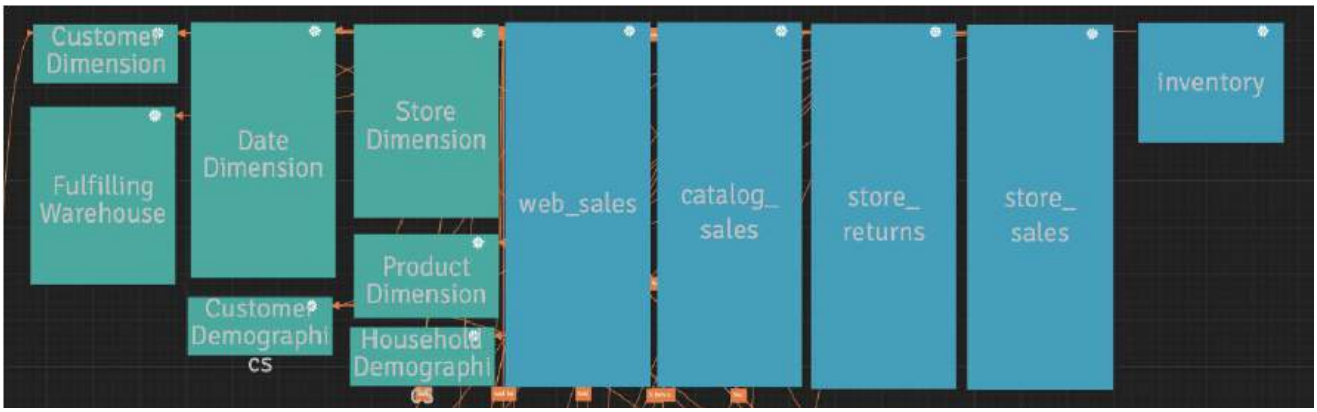
This directive is problematic because this type of filter belongs in the user's dashboard or report query, not in the definition of the model. As a result, this filter drastically reduces the value of this model since this directive means only "paying" customers can ever be analyzed. The following construct is also problematic:

```
dimensions:
  - plan
  - country
```

This directive limits the model to analysis by plan and country. While this is an illustrative example and more dimensions can be added to the model, ultimately this markup language is too simplistic.

A better approach is to use a multidimensional model that allows for “anything by anything” queries by supporting relationships between several entities (i.e. customer by location) with many-to-many, many-to-one and semi-additive relationships for expressing business rules.

For example, the multidimensional model below, based on an excerpt of the TPC-DS benchmark model, allows users to report sales and returns by source, by store, product, date, warehouse, customer and customer demographics in any combination desired.



The relationships between these business entities is crucial and SQL alone is too cumbersome for expressing these relationships.

Key Takeaway: A semantic layer must support a wide range of query patterns that are not constrained to a single view of data or subset of dimensions or metrics. Semantic layer solutions that define slices of data are not flexible enough to support a wide range of use cases and will force end users to wait for model owners to introduce new model views.

A Multidimensional Calculation Engine Is Critical

Besides serving as a metrics hub, a semantic layer —powered by a multi-dimensional calculation engine — creates a digital twin of your business. In order to express the complexities of business processes, the semantic layer platform must support dimensional expressions, hierarchies and entity relationships defined in a semantic model and backed by a graph-based query planner.

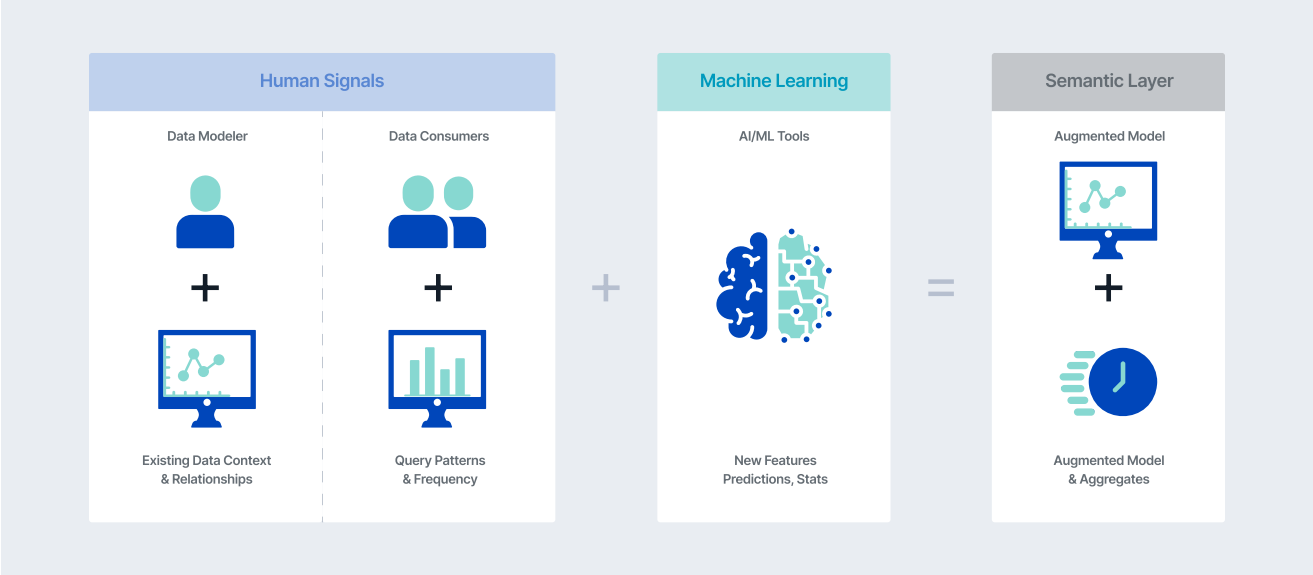
Chapter 5: Performance Optimization

For a semantic layer to function, it's critical to deliver a live, interactive query experience to discourage users from bypassing the semantic layer by moving data into external, ungoverned caching layers.

Autonomous & Adaptive

For a semantic layer to be useful, it must be the source of truth for all queries, which means data needs to be queried "live" regardless of where the data lives. Creating copies of data to improve performance introduces data latency and inconsistency and thereby undermines the core values of a semantic layer.

It follows, then, that a semantic layer needs to deliver extract-level performance against the cloud data platforms natively. As such, the semantic layer must include automated performance management to deliver queries at "speed of thought" with a live connection to cloud data platforms. Since data is always growing and evolving and user query behavior is far from predictable, attempts to manually tune queries is futile.



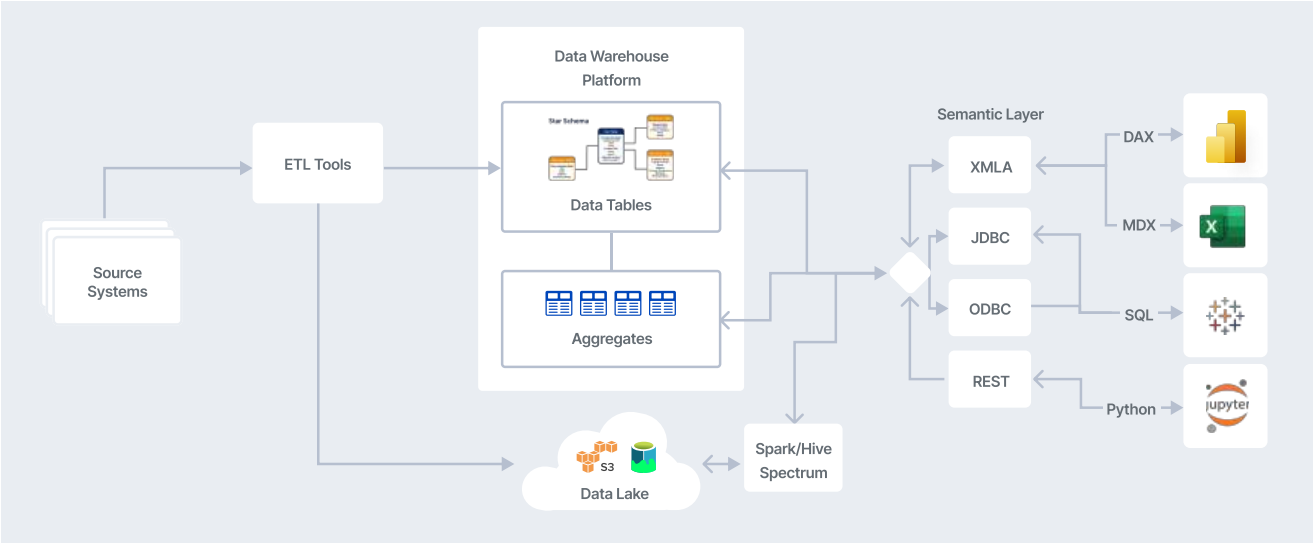
As illustrated above, the semantic layer platform must be capable of rewriting queries and creating aggregates on-demand using the data model, end user query patterns, data statistics and machine learning to automatically manage performance for every query.

Key Takeaway: A semantic layer must autonomously tune query performance to support interactive, live connections to data platforms. Semantic layer solutions that do not automatically manage query performance are unsuitable for supporting direct (live) queries.

In Situ: No Data Movement Required

When working with cloud data platforms, most data analysts using BI tools like Tableau and Power BI create data extracts (i.e. Hyper) or import data into their BI tools before creating their dashboards and reports. It's not their desire to add another data management task to their workflow, but end users must create these data copies in order to get the performance and interactivity they need when querying cloud data platforms. Besides the extra work involved, creating data copies outside of the data platform adds costs and introduces data inconsistency and security risks.

To avoid these pitfalls, a semantic layer should improve performance without moving data outside of the native cloud data platform and avoid creating and managing a separate query acceleration infrastructure for accelerating performance.



As illustrated above, ideally, the semantic layer should create aggregate tables, or materialized views, on-demand using a machine learning model that is informed by user query patterns and data statistics. By rewriting queries to target smaller aggregates instead of raw data, an optimized semantic layer can also substantially reduce your cloud data platform's operating costs.

To see just how much a semantic layer can accelerate queries and reduce costs, check out AtScale's TPC-DS 10TB Benchmark Reports.

Key Takeaway: A semantic layer must deliver query performance at "speed of thought" with a live connection to data platforms without the need to create tool-specific extracts or imports or moving data to separate caching subsystems.

Powerful Autonomous Performance Management

Besides serving as a metrics hub, a semantic layer — powered by autonomous performance management — provides a live, interactive connection to your data without data movement. By avoiding external caching layers, data stays where it landed and queries scale with your cloud data platform. By avoiding redundant data scans, an optimized semantic layer can also substantially reduce your data platform's operating costs.

Chapter 6: Analytics Governance

While there are platforms and tools focused on just data governance, it's impossible to truly secure data without integrating governance rules with the semantic layer's logical view of data. While the semantic layer must respect and integrate with the underlying data platforms' native, physical security policies, it must extend those policies to the derived business calculations and constructs defined in the semantic layer data model.

Who's Who in the Zoo?

Governance and security must start with the user's identity. Without knowing who is running a query, it's not feasible to apply data access policies to users or groups. No governance controls can work without user identity management. As a result, table stakes for any semantic layer is to support semantic layer integration with enterprise directory services like Active Directory (AD), LDAP, Okta, and more.

With enterprise directory integration, the semantic layer can identify a user and then run queries "as that user" against the native data platform. By doing so, the semantic layer will respect the underlying data platform's physical security policies — something not possible if the semantic layer uses a proxy or service account to run queries instead. The semantic layer must also synchronize its users and groups with the directory service users and groups to avoid a duplicative, shadow governance infrastructure when applying data access policies.

This requires a semantic layer to support deep integration with IT services and data platforms — nothing else will do.

Key Takeaway: A semantic layer must support deep integration with IT identity management services and respect underlying data platform security policies by running queries with the user's account.

Realtime Enforcement

Enterprises are in constant motion, adding new employees, new business groups and new data sets all the time. For a semantic layer to serve as the governance control plane for the enterprise, it must react to these changes in real time.

Starting with user and group management, a semantic layer must seamlessly integrate and sync with enterprise identity management services. As new employees join, depart or change groups, the semantic layer's governance policies will instantly reflect those changes.

With policies defined in the semantic layer and user identity up to date, the semantic layer platform can then intercept consumer queries and rewrite them to enforce governance policies in real time. Since the semantic layer can connect to any consumer, data platform or user persona, it can deliver comprehensive and consistent coverage with confidence.

Key Takeaway: A semantic layer must enforce data governance in real time for every query in order to provide comprehensive coverage and respond to frequently changing policies.

Who Sees What

With the ability to identify end users by name, the semantic layer can apply the following governance functions that are critical to securing data access in a semantic layer:

1. Row-level security
2. Column-level security
3. Object-level security

The first critical governance feature for a semantic layer is row-level security, or the ability to apply a filter (or WHERE clause) to each outbound query to select only the rows of data that users should see. For example, row-level security can be used to automatically restrict data for a sales team in the West so they only see their data and not the data from other regions. As long as the salesperson is mapped to the Western Region's group, the semantic layer can automatically generate the proper WHERE clause to restrict data access to just the West region's data for that user.

The second critical governance feature for a semantic layer is column-level security, or the ability to hide sensitive data columns or mask their contents. For example, a column-level security rule can hide or mask personally identifiable information (PII) fields for the marketing team while making them visible for the HR team.

Column	Rule	What HR Sees	What Marketing Sees
Social security number	Visible to HR, masked for everyone else	123-45-6789	XXX-XX-6789
Customer Address	Visible to HR, hidden for everyone else	123 Anywhere St.	Not Visible

By dynamically adjusting the view of the semantic layer based on user access rules, the semantic layer can be defined for everyone, but appear customized to users and groups.

The final key governance function is object-level security. This layer of governance allows for users and groups to own and share modeling components (i.e. conformed dimensions, hierarchies, calculations, models and much more). This functionality is critical to supporting the concept of a data mesh, a popular topic of discussion in the data and analytics community. The key principles for a data mesh architecture are supporting domain ownership of data objects and a decentralized system of data stewardship. Without modeling object-level security, backed by RBAC, achieving the vision of federating the creation and management of data products just isn't feasible.

Key Takeaway: A semantic layer must apply query governance with dynamic filtering, column-level security and object-level security based on the query user's identity. Semantic layer solutions that lack row, column and modeling object controls are not suitable for use cases where data access restrictions are required.

Data and Analytics Governance Together

Besides serving as a metrics hub, a semantic layer must apply data access controls and governance policies to every query using the query user's identity. By avoiding duplicative governance tools and shadow user management, enterprises can apply both physical and logical data access policies to ensure that data is only visible to those who have authorized access. With the trust that data is secure, organizations can confidently share data more broadly both internally and externally.

Chapter 7: Data Integration

By hiding the format, location and complexity of data, a semantic layer provides a business-friendly view of data for everyone, not just data engineers and SQL jockeys. Delivering a logical view of data on a variety of data platforms has its challenges, though. In this post, we'll drill down on some important considerations when evaluating a semantic layer's ability to integrate disparate data sources.

Speaking Their Language

In addition to having various levels of support for SQL and SQL extensions, different data platforms have different performance characteristics and optimization controls. In order to avoid data movement and all its adverse effects, it's imperative that the semantic layer generate platform-optimized queries and push down those queries to the underlying data platform. Lowest common denominator approaches that generate and execute simple, vanilla SQL and perform aggregations and calculations locally, just can't scale and don't allow customers to take advantage of platform-specific features.

A scalable semantic layer platform must integrate seamlessly with data platforms and must:

1. Generate and execute multi-pass SQL and push down queries to the underlying data platform
2. Leverage platform-specific optimizations, including partitioning, clustering and DDL hints
3. Avoid a separate compute infrastructure to process query results
4. Allow native SQL in calculations to leverage platform-specific and user-defined functions

With tight, platform-specific integrations, a semantic layer will generate optimized queries to deliver consistent performance and lower costs.

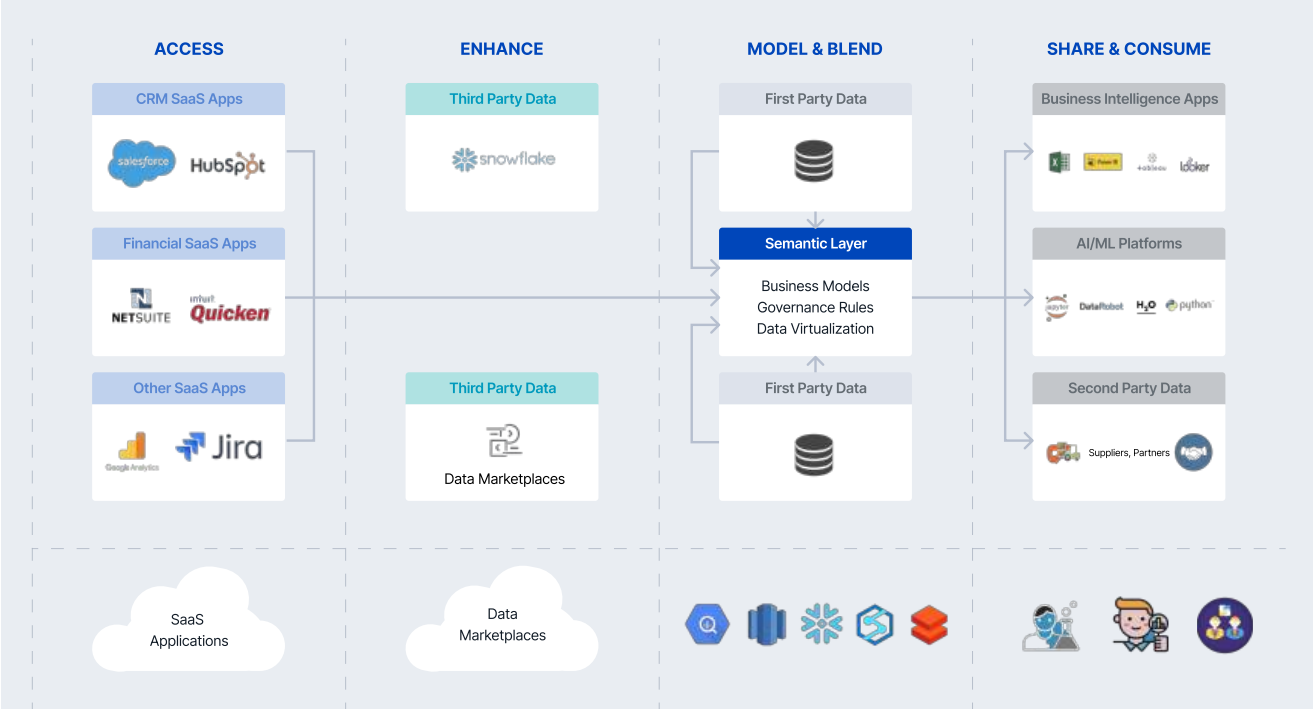
Key Takeaway: A semantic layer must work with a variety of data platforms equally well by supporting native platform dialects and optimizations.

Breaking Down Silos

It seems like just about every five years we see a new data platform technology or trend become all the rage. If your organization has been around long enough, you probably have one of everything and a proliferation of cloud-based applications with your precious data locked behind their proprietary APIs. A semantic layer with data virtualization future-proofs your data platform technology choices by creating an abstraction layer between your data and the tools that interact with it. Besides hiding the complexity of each data platform and preventing vendor lock-in, a semantic layer minimizes or eliminates the cost of migration to new data platforms in the future by using data virtualization as its core mechanism for querying the underlying data.

A semantic layer with query federation goes even further to break down data silos than just virtualizing data access. As I discussed in my earlier blog, The Semantic Layer. Back to the Future Part 3: The Semantic Model, a semantic layer's data model can blend data from multiple sources to create new, composite views of data for modeling complex business processes.

The illustration below shows how a semantic layer can blend data from multiple sources, including SaaS applications, third party data from exchanges and first party data:



Key Takeaway: A semantic layer must support data blending across multiple data platforms and data sources and minimize data movement for federated queries by leveraging localized aggregates and query push down.

More Than Rows and Columns

In today's cloud data platform world, a database table is no longer just made up of simple rows and columns. Modern data platforms now have advanced support for non-scalar data types like JSON and XML. Google BigQuery goes even further by allowing for nested and repeating fields, which effectively allows the embedding of tables within tables. For example, a customer table may take the form of:

1. id
2. first_name
3. last_name
4. dob (date of birth)
5. addresses (a nested and repeated field)
   - addresses.status (current or previous)
   - addresses.address
   - addresses.city
   - addresses.state
   - addresses.zip
   - addresses.numberOfYears (years at the address

It's easy to see how powerful these constructs can be for compressing information and reducing table joins, something cloud data platforms are not good at. However, while these new constructs make data loading easier and querying faster, they add additional complexity for the user when writing queries. Besides having to understand these more complex table constructs, each data platform has its own proprietary syntax for unnesting these data types.

A semantic layer should hide this additional complexity from the end user and abstract away the dialect differences of each platform. In this way, data warehouse architects can take advantage of these powerful new "schema on read" design principles without creating an additional burden on their end users.

Key Takeaway: A semantic layer must support modern data platform features and constructs to support analytics on unstructured and semi-structured data.

Data Integration for Data Democratization

Besides serving as a metrics hub, a semantic layer must hide the complexity of the data stored in various data platforms. A well designed semantic layer will scale with your data growth (and your data platforms) by pushing down queries and leveraging each platform's dialect and optimizations features. By eliminating data copies and data movement, a semantic layer can make data instantly accessible to everyone.

Final Thoughts

As you can see, building a semantic layer platform is not simply a matter of defining metrics with a cool new markup language. For a semantic layer to be practical and usable, it needs to:

1. Be capable of expressing your **most complex** business constructs
2. Be able to **perform better** than your underlying data platforms
3. Be able to **connect live** to all your data platforms
4. Be able to **connect to all** your data consumption tools
5. Be able to **govern every query** at the user level
6. Be able to **scale to everyone** in your business

If any of these requirements is missing, a semantic layer is unusable. In other words, it's binary — it either works 100% or it doesn't work at all. Therein lies the challenge for anyone building a universal semantic layer from scratch. It's not good enough to deliver an MVP that sort of works and can be enhanced as you go. The MVP is not an MVP — it just needs to work completely on day one or no one will bother using it.

The team at AtScale has spent more than a decade working to deliver the vision of a universal semantic layer and making it work for real, demanding customers. A universal semantic layer has become a critical component in the modern data and analytics stack. We cannot be more pleased to see our industry partners (and competitors) agree.



Dave Mariani, Chief Technology Officer, Founder, AtScale

Dave is the founder of AtScale and is the Chief Strategy Officer. Prior to AtScale, he was VP of Engineering at Klout & at Yahoo! where he built the world's largest multi-dimensional cube for BI on Hadoop. Mariani is a Big Data visionary & serial entrepreneur.

Brand and product names mentioned in this paper are trademarks or registered trademarks of AtScale and other companies.