SDF

Semantic Data Fabric*

SDF Labs
Wolfram Schulte
wolfram@sdf.com

PNW PLSE 2024 Workshop

---

* Restricted to using SDF as a static analyzer; other uses are discussed on sdf.com

What is a Data Fabric?

Architecture for managing data in the data warehouse

Elements of the architecture are…

- **Processors**: SQL, Python, etc
- **Dependencies**: Data,Time,Control
- **Data**: Tables, Streams, KeyValue Stores, Files, etc

Semantic Data Fabric

A Data Fabric that understands warehouse E2E

We need semantics for …

- **Security**: Who can access which data for what purpose
- **Privacy**: How is data being stored/exposed/processed/deleted for what purpose
- **Analytics**: What does that data mean,e.g. metrics, domains, provenance
- **Efficiency**: How can the data be processed more efficiently, e.g. materializations
- **Timeliness**: How recent is the data and when is the next data available

Static analysis can help with all of this, in particular privacy, analytics and efficiency!

Challenge to Understand a Data Warehouse?

SQL is the Lingua Franca of the warehouse, SQL is declarative

However …

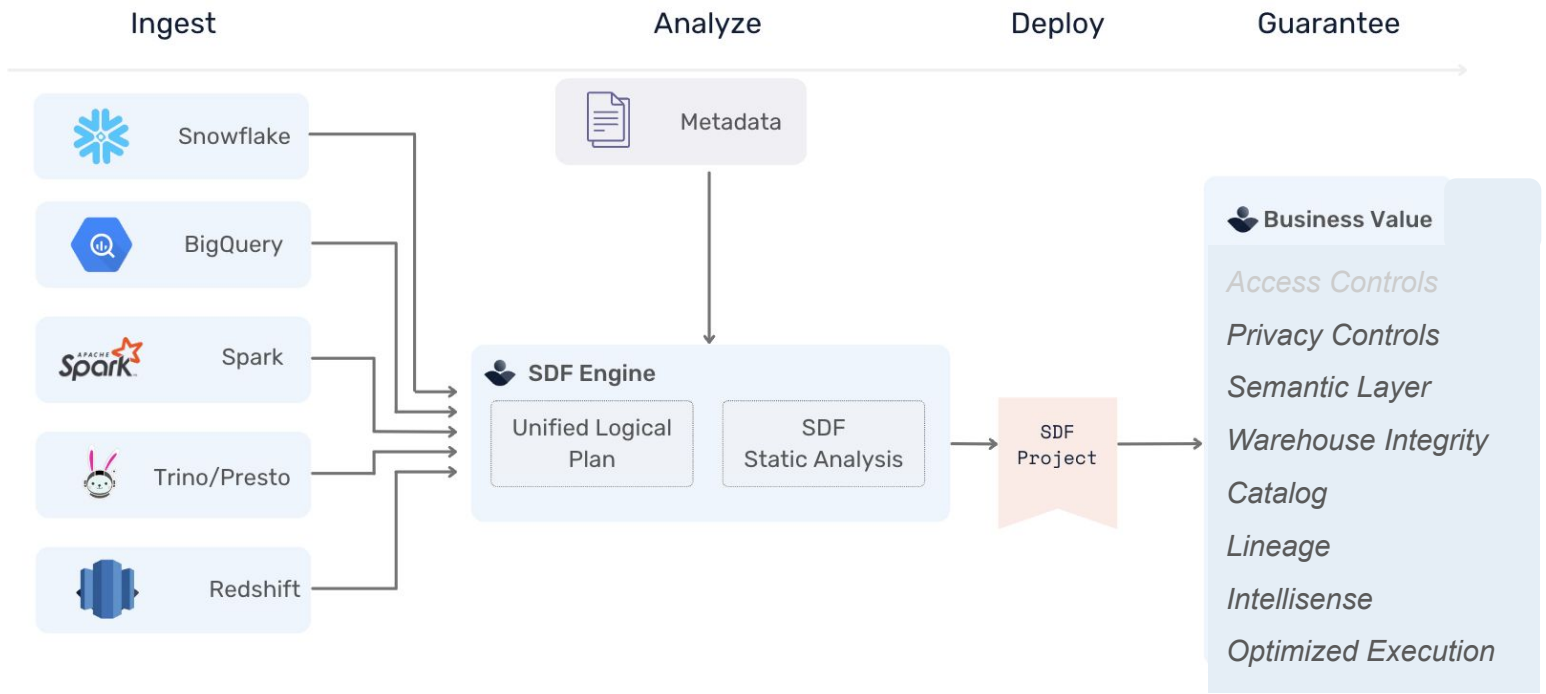
- SQL is **no general purpose** PL – Missing procedural and data abstraction, etc.
- SQL analysis is **non trivial** – Single queries with upto 200k LOC, and upto 20k fields
- SQL dialects **vary dramatically** – No agreed upon standard

So warehouse often use many processors/dependencies/data-representations

SDF is …

- A multi-dialect **SQL compiler and build system**
  - Analyzes warehouse E2E, so all of its SQL and all its dependencies; generates executable logical plan
- An extended **type system using information flow analysis** (IFA)
  - Supports semantic based column level lineage, classifier definition and propagation; IFA works on logical plan
- User authorable **checks and reports**
  - Reports and verifies user defined code properties and assertions, expressed as SQL; computed via IFA at compile time and accessible via compiler generated information schema
- Enriched **developer experience** with context and AI
  - Intellisense, and classifier, report and code check drafting using AI

SDF Compiler Architecture



SDF Classifiers and Assertions

To enrich SQL semantics and enforce constraints SDF uses 4 ingredients

User:

- **Define Classifiers**, e.g. label sets and propagation rules via metadata
- **Attach Classifiers** to root tables or columns and control their propagation via metadata
- **Write Assertions** in SQL against the SDF information schema to report/assert expected/unexpected behavior

Compiler:

- **Analyzes all SQL, propagates all classifiers** and **evaluates all assertions** at compile time.

Example: Marketing email lists should NOT expose users that have NOT given their consent

Define Classifier

```
classifier:
  name: consent_status
  labels: [accepted, denied]
  scope: [column, table]
  propagate: track
```

```
classifier:
  name: purpose
  labels: [marketing]
  scope: [table]
  propagate: none
```

Attach Classifier

```
table:
  name: user_consent
  columns:
  - name: user_id
  - name: status
    classifiers:
      [consent_status.*]
```

```
table:
  name: email_list
  classifiers:
    [purpose.marketing]
```

Compile-time Code Assertion

check that no such table exists

```
select
  table_id
from
  sdf.information_schema.tables
where
  array_contains(
    classifiers,
    'consent_status.denied')
and
  array_contains(
    classifiers,
   'purpose.marketing');
```

Constant effort for definition, attachment, and assertion.

Propagation, ***analysis & enforcement*** *for any scale* ***for free****!*

SDF Summary

SDF *is a command line tool and cloud service that*
statically analyzes Warehouses E2E
providing meaning and enforcements via classifiers and rules

to ensure and provide

- Privacy
- Analytics
- Integrity
- Efficiency
- Catalog & Lineage
- Intellisense & AI drafting

More at sdf.com