dbt next

(6 aug 2020)

New in Marian Anderson (v0.18.0)

- Advanced model selectors
- “Slim CI”: Deferred runs of changed models *only*
- More extensible framework for cross-database macros

Check out:

- changelog
- migration guide
- milestone

As dbt projects get bigger...

```
$ dbt run

$ dbt run -m tag:just_the_ones_i_want
```

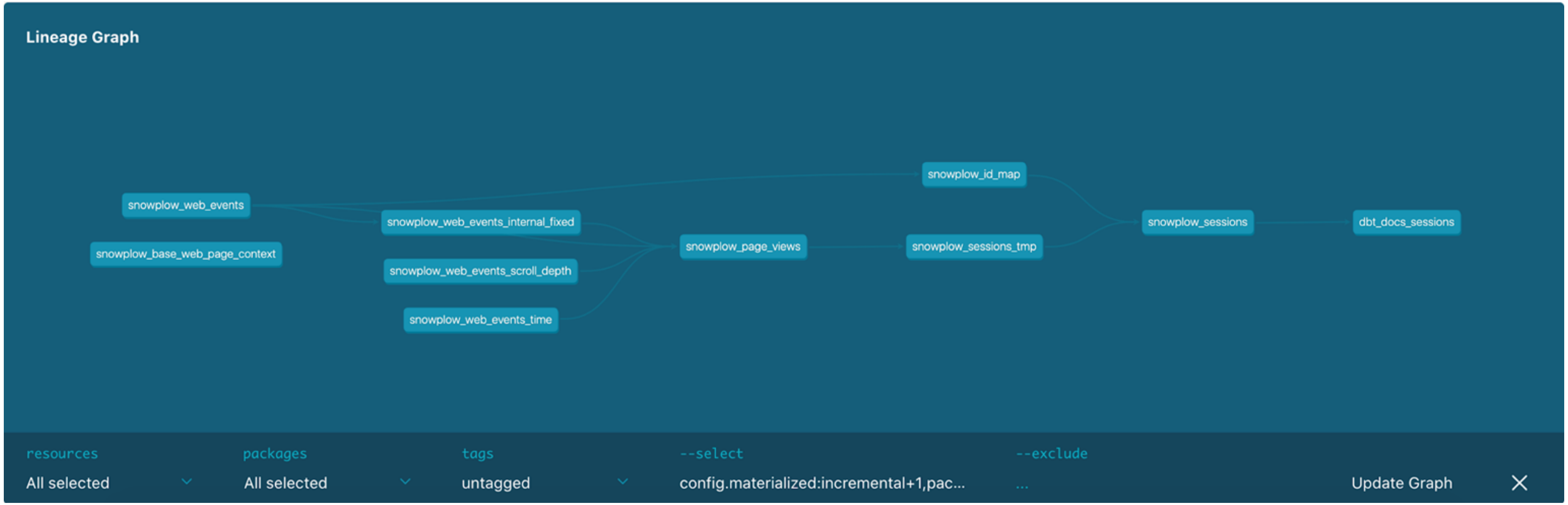
We need more power!

New node selection

- methods: config, test_type, test_name, package
- intersections (“this AND that”)
- nth-degree parent/child
- version-controlled YML selectors

New node selection

```
# list all my incremental models
$ dbt ls -m config.materialized:incremental

# run only incremental models defined in the snowplow package
$ dbt run -m config.materialized:incremental,package:snowplow

# run only incremental models defined in the snowplow package, and
their immediate offspring
$ dbt run -m config.materialized:incremental+1,package:snowplow+1

# execute my “severe” tests downstream of a source
$ dbt test -m source:stripe+ --exclude config.severity:warn
```



resources
All selected

packages
All selected

tags
untagged

--select
config.materialized:incremental+1,pac...

--exclude
...

“Slim CI”

dbt Cloud can “build on PR,” via GitHub integration, into a scratch schema.

What if it could run *only* the models it has to?

- How do we know which models *changed?* **Compare**
- What about their *parents?* **Defer**

```
$ dbt run -m state:modified+ --defer --state path/to/artifacts
```

Cross-database functionality

These all do the same thing:

```
-- postgres
extract(epoch from timestamp_a - timestamp_b)/3600

-- redshift
datediff(hour, timestamp_a, timestamp_b)

-- bigquery
timestamp_diff(timestamp_b, timestamp_a, hour)
```

“Dispatch” macros

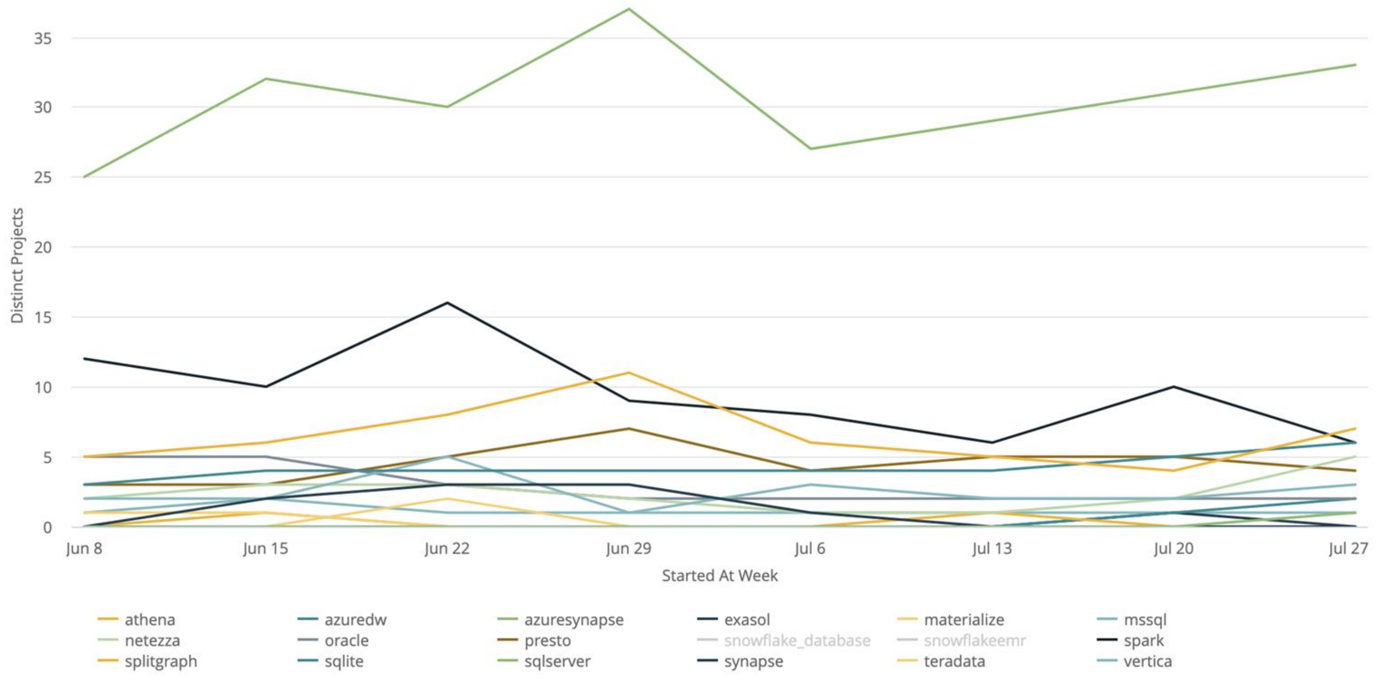
```
{% macro datediff(first_date, second_date, datepart) %}
         {{ adapter.dispatch(macro_name = 'datediff', packages = … )
                  (first_date, second_date, datepart) }}
{% endmacro %}

{% macro postgres__datediff(first_date, second_date, datepart) %} … {% endmacro %}

{% macro redshift__datediff(first_date, second_date, datepart) %} … {% endmacro %}

{% macro bigquery__datediff(first_date, second_date, datepart) %} … {% endmacro %}
```

And someone else can define:

```
{% macro spark__datediff(first_date, second_date, datepart) %} … {% endmacro %}
```

Why now?

- Road to [v1.0.0]
- Plugins!



Adapters as plugins

Since Stephen Girard (v0.13.0, March 2019), dbt has supported external adapter plugins as a way to port its functionality to new databases.

Core plugins	Public plugins		Private plugins
Fishtown	Fishtown	Community	???
Snowflake BigQuery Postgres Redshift	Spark Presto	SQLServer, MSSQL, Azure DW, Synapse Athena Oracle Exasol	Netezza Vertica IBM DB2 Hive ...

What does that get us?

- dbt (and its viewpoint) at more & more organizations
- More robust open-source community
- Functionality we could never have on the “core four” analytical databases. For instance, genuinely different tooling for real-time or operational analytics:
    - Spark structured streaming
    - Materialize.io

dbt plugins → dbt Cloud: Q4 2020