DWH
Data Vault - 2

otus.ru

Проверить, идет ли запись

Меня хорошо видно && слышно?

Тема вебинара

Data Vault - 2

Андрей Тюрин

DWH Analyst

- 4 года опыта Аналитиком хранилищ данных
- Сейчас работаю в экосистеме услуг для бизнеса
- Ранее : на проектах крупных банков

LinkedIn

Правила вебинара

Активно
участвуем

Задаем вопрос
в чат или голосом

Вопросы вижу в чате,
могу ответить не сразу

Условные обозначения

Индивидуально

Время, необходимое на активность

Пишем в чат

Говорим голосом

Документ

Ответьте себе или задайте вопрос

Маршрут вебинара

- Знакомство
- Business Vault с dbt
- Кодогенерация с dbt (повторение)
- Оркестрация процесса кодогенерации
- Рефлексия

Цели вебинара

К концу занятия вы сможете

1. Погрузиться в подход к организации детального слоя Data Vault 2.0
2. Рассмотреть пример построения DWH на DV 2.0.

Смысл

Зачем вам это уметь

1. Научиться применять подход Data Vault для моделирования детального слоя DWH

Требования при проектировании DWH

Требования к самому хранилищу:

- Гетерогенность источников
- Поддержка историчности
- Гибкость модели данных
- Скорость обновления
- Устойчивость к объёму

Типовые кейсы:

- Добавление новых данных (источники) в существующую модель
- Отработка изменений на источниках
- Создание пользовательских представлений и их сопровождение
- …

Практика: dbt-vault с Greenplum

https://github.com/kzzzr/dbtvault_greenplum_demo/tree/master

Data Vault 2.0 + Greenplum + dbtVault assignment

Business Vault

Business Data Vault

В чем недостатки классического Data Vault?

Business Vault

- Transactional Links - AutomateDV
- As of Date Tables - AutomateDV
- Point In Time (PIT) tables - AutomateDV
- Bridge Tables - AutomateDV
- PREDEFINED DERIVATIONS

Transactional Links

Помимо самой связи двух хабов содержат информацию о событии.

Например: покупки, авиабилеты или электронные письма

Отличия от обычных link:

- есть набор полей **payload** с данными (информацией, описывающей событие, например, покупку)
- **Effective From**

Payload не является обязательным. Почему?

Transactional Links

Payload не является обязательным. Почему?

Вместо хранения данных в самом Transactional Link можно добавить для него один или несколько сателитов.

Transactional Links

Payload не является обязательным. Почему?

Вместо хранения данных в самом Transactional Link можно добавить для него один или несколько сателитов.

```
{{ automate_dv.t_link(src_pk=src_pk, src_fk=src_fk, src_payload=src_payload,
                      src_eff=src_eff, src_ldts=src_ldts, src_source=src_source,
                      source_model=source_model)}}
```

Transactional Links

Как вид материализации лучше выбрать?

Transactional Links

Как вид материализации лучше выбрать?

Рекомендуемая материализация для Transactional Links - incremental, поскольку мы загружаем и добавляем новые записи в существующий набор данных.

Transactional Links

```
{{ config(materialized='incremental') }}
{%- set yaml_metadata -%}
source_model: 'v_stg_transactions'
src_pk: 'TRANSACTION_HK'
src_fk:
    - 'CUSTOMER_HK'
    - 'ORDER_HK'
src_payload:
    - 'TRANSACTION_NUMBER'
    - 'TRANSACTION_DATE'
    - 'TYPE'
    - 'AMOUNT'
src_eff: 'EFFECTIVE_FROM'
src_ldts: 'LOAD_DATETIME'
src_source: 'RECORD_SOURCE'
{%- endset -%}
```

```
{% set metadata_dict = fromyaml(yaml_metadata) %}
{{ automate_dv.t_link(src_pk=metadata_dict["src_pk"],
                      src_fk=metadata_dict["src_fk"],
                      src_payload=metadata_dict["src_payload"],
                      src_eff=metadata_dict["src_eff"],
                      src_ldts=metadata_dict["src_ldts"],
                      src_source=metadata_dict["src_source"],

source_model=metadata_dict["source_model"]) }}
```

As of Date Tables

Таблица **As of Date** содержит один столбец дат, используемый для построения истории в таких таблицах, как PIT и Bridges.

```
{{ config(materialized='table') }}

{%- set datepart = "day" -%}
{%- set start_date = "TO_DATE('2021/01/01', 'yyyy/mm/dd')" -%}
{%- set end_date = "TO_DATE('2021/04/01', 'yyyy/mm/dd')" -%}

WITH as_of_date AS (
    {{ dbt_utils.date_spine(datepart=datepart,
                            start_date=start_date,
                            end_date=end_date) }}
)

SELECT DATE_{{datepart}} as AS_OF_DATE FROM as_of_date
```

Point-in-time - PIT

- делим сателитов по частоте обновления
- грузим данные независимо
- как получать актуальные данные?

Point-in-time - PIT

- сделать JOIN
- создать несколько вложенных запросов (к каждому сателлиту содержащему информацию) с выбором максимальной даты обновления MAX(Дата обновления)

запросы очень сильно разрастутся!

Point-in-time - PIT

```
SELECT H.HUB_EMPLOYEE_KEY, H.EMPLOYEE_ID, S.FIRST_NAME, S.LAST_NAME,
S.HIRE_DATE, C.SALARY, …
FROM HUB_EMPLOYEE H
JOIN SAT_EMPLOYEE_NAME NM
ON H. HUB_EMPLOYEE_KEY = NM. HUB_EMPLOYEE_KEY
JOIN SAT_EMPLOYEE_COMPENSATION COMP
ON H. HUB_EMPLOYEE_KEY = COMP. HUB_EMPLOYEE_KEY
JOIN SAT_EMPLOYEE_ADDRESS ADDR
ON H. HUB_EMPLOYEE_KEY = ADDR. HUB_EMPLOYEE_KEY
JOIN SAT_EMPLOYEE_CONTACTS CNTC
ON H. HUB_EMPLOYEE_KEY = CNTC. HUB_EMPLOYEE_KEY
JOIN EMPLOYEE_PIT P
ON H. HUB_EMPLOYEE_KEY = P. HUB_EMPLOYEE_KEY
AND P.PIT_LOAD_DTS = '15-FEB-2015'
AND P.NAME_LOAD_DTS = NM.SAT_LOAD_DTS
AND P.ADDRESS_LOAD_DTS = ADDR.SAT_LOAD_DTS
AND P.CONTACT_LOAD_DTS = CNTC.SAT_LOAD_DTS
AND P.COMENSATION_LOAD_DTS = COMP.SAT_LOAD_DTS)
```

Point-in-time - PIT

Бизнес ключ	Дата загрузки в PIT	Дата загрузки (изменения) данных сателлита ИМЯ	Дата загрузки (изменения) данных сателлита АДРЕС	Дата загрузки (изменения) данных сателлита ТЕЛЕФОН
00096998_F	01.01.2020	02.12.2018	NULL	NULL
00096998_F	02.01.2020	02.12.2018	02.01.2020	NULL
00096998_F	03.01.2020	02.12.2018	03.01.2020	03.01.2020
00096998_F	04.01.2020	02.12.2018	03.01.2020	03.01.2020

- делим сателитов по частоте обновления
- грузим данные независимо
- как получать актуальные данные?

Point In Time (PIT) tables

Рекомендуется использовать таблицу PIT при ссылке как минимум на два сателита и особенно, когда сателиты имеют разную скорость обновления.

```
{{ automate_dv.pit(source_model=source_model, src_pk=src_pk,
                   as_of_dates_table=as_of_dates_table,
                   satellites=satellites,
                   stage_tables_ldts=stage_tables_ldts,
                   src_ldts=src_ldts) }}
```

Bridge Tables

Bridge-таблицы — это вспомогательные таблиц, являющиеся частью Business Vault.

Как и в случае с таблицами PIT, их цель — **повысить производительность запросов** к хранилищу необработанных данных за счет сокращения количества необходимых join-ов для таких запросов до простых эквивалентных соединений.

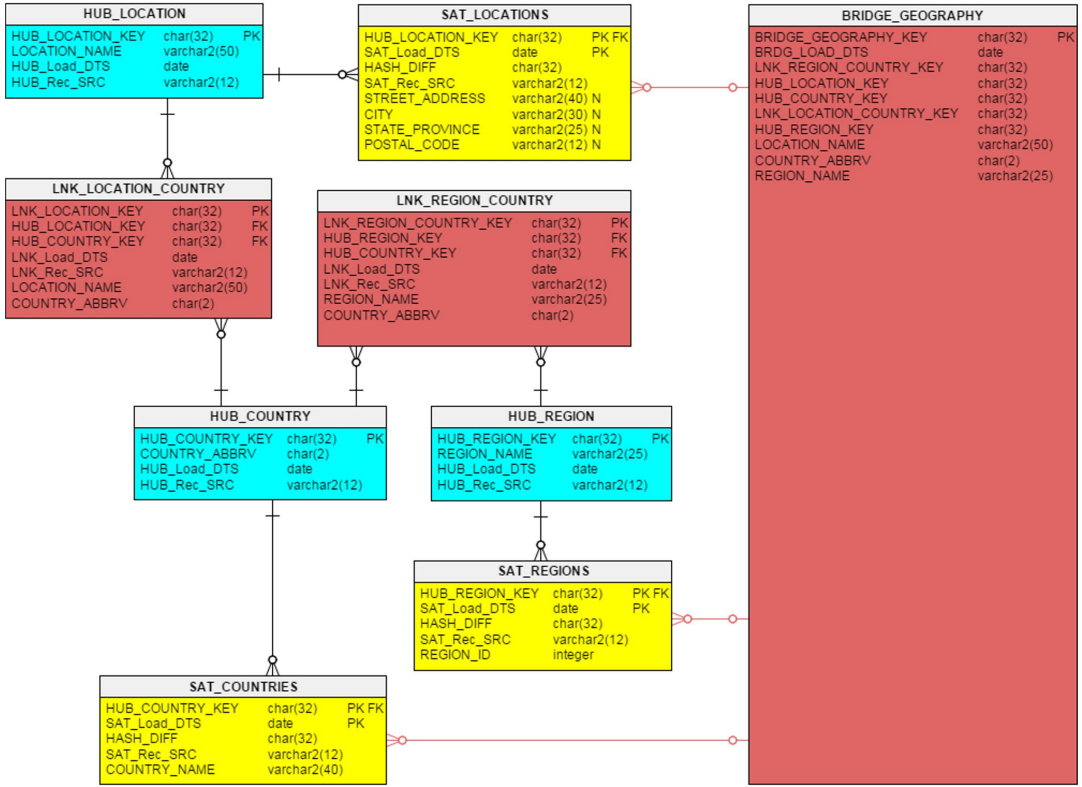
Таблица Bridge охватывает Hub и одну или несколько связанных Link.

```
{{ automate_dv.bridge(source_model=source_model, src_pk=src_pk,
                      src_ldts=src_ldts,
                      bridge_walk=bridge_walk,
                      as_of_dates_table=as_of_dates_table,
                      stage_tables_ldts=stage_tables_ldts) }}
```

Bridge table

Проблема:
в процессе получения данных находящихся в сателлитах принадлежащим разным хабам, потребуется произвести JOIN не только самих **сателлитов**, но и **линков** связывающих хабы.

Bridge table



Bridge table

```
SELECT BRDG.LOCATION_NAME, S1.STREET_ADDRESS, S1.CITY, S1.STATE_PROVINCE,
S1.POSTAL_CODE, S2.COUNTRY_NAME, BRDG.REGION_NAME, S3.REGION_ID
FROM BRIDGE_GEOGRAPHY BRDG
JOIN SAT_LOCATIONS S1
ON BRDG.HUB_LOCATION_KEY = S1.HUB_LOCATION_KEY
JOIN SAT_COUNTRIES S2
ON BRDG.HUB_COUNTRY_KEY = S2.HUB_COUNTRY_KEY
JOIN SAT_REGIONS S3
ON BRDG.HUB_REGION_KEY = S3.HUB_REGION_KEY
```

содержит:

- все необходимы ключи для всех сателлитов, которые часто используются в запросах.
- при необходимости хешированные бизнес ключи могут дополняться ключами в текстовом виде, если наименования ключей нужны для анализа.

PREDEFINED DERIVATIONS

таблицы содержащие предварительно рассчитанные показатели: еще один сателлит определенного хаба.

- Он, как и обычный сателлит содержит бизнес ключ и дату формирования записи в сателлите.
- Состав атрибутов такого «специализированного» сателлита определяется бизнес пользователями на основе наиболее востребованных, предварительно рассчитанных показателей.

PREDEFINED DERIVATIONS

таблицы содержащие предварительно рассчитанные показатели: еще один сателлит определенного хаба.

- Он, как и обычный сателлит содержит бизнес ключ и дату формирования записи в сателлите.
- Состав атрибутов такого «специализированного» сателлита определяется бизнес пользователями на основе наиболее востребованных, предварительно рассчитанных показателей.

! Обычно PREDEFINED DERIVATIONS включают в состав PIT таблицы этого же хаба, тогда можно без труда получить срезы данных на конкретно выбранную дату

Основы кодогенерации

Адаптеры для различных СУБД

adapter.dispatch

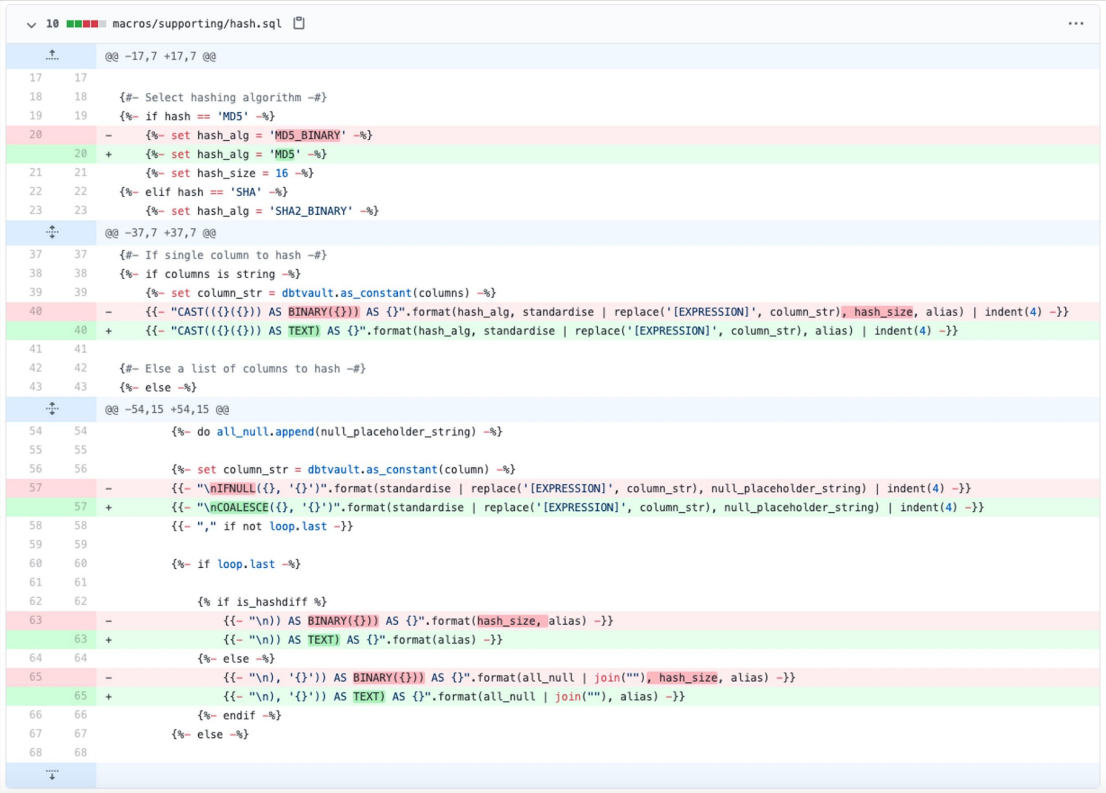
Every macro in dbtvault first calls adapter.dispatch to find platform specific implementations of the macro to execute.

Here is an example:

hub.sql

```
1  {%- macro hub(src_pk, src_nk, src_ldts, src_source, source_model) -%}
2
3  {{- adapter.dispatch('hub', packages = dbtvault.get_dbtvault_namespaces())(src
4                                                                              src
5                                                                              sou
6
7  {%- endmacro -%}
```

Snowflake > Postgres (Greenplum)



10 macros/supporting/hash.sql

```
@@ -17,7 +17,7 @@
17 17
18 18   {#- Select hashing algorithm -#}
19 19   {%- if hash == 'MD5' -%}
20    -     {%- set hash_alg = 'MD5_BINARY' -%}
   20 +     {%- set hash_alg = 'MD5' -%}
21 21       {%- set hash_size = 16 -%}
22 22   {%- elif hash == 'SHA' -%}
23 23       {%- set hash_alg = 'SHA2_BINARY' -%}
@@ -37,7 +37,7 @@
37 37   {#- If single column to hash -#}
38 38   {%- if columns is string -%}
39 39       {%- set column_str = dbtvault.as_constant(columns) -%}
40    -     {{- "CAST(({}({})) AS BINARY({}))".format(hash_alg, standardise | replace('[EXPRESSION]', column_str), hash_size, alias) | indent(4) -}}
   40 +     {{- "CAST(({}({})) AS TEXT) AS {}".format(hash_alg, standardise | replace('[EXPRESSION]', column_str), alias) | indent(4) -}}
41 41
42 42   {#- Else a list of columns to hash -#}
43 43   {%- else -%}
@@ -54,15 +54,15 @@
54 54           {%- do all_null.append(null_placeholder_string) -%}
55 55
56 56           {%- set column_str = dbtvault.as_constant(column) -%}
57    -         {{- "\nIFNULL({}, '{}')".format(standardise | replace('[EXPRESSION]', column_str), null_placeholder_string) | indent(4) -}}
   57 +         {{- "\nCOALESCE({}, '{}')".format(standardise | replace('[EXPRESSION]', column_str), null_placeholder_string) | indent(4) -}}
58 58           {{- "," if not loop.last -}}
59 59
60 60           {%- if loop.last -%}
61 61
62 62               {% if is_hashdiff %}
63    -                 {{- "\n)) AS BINARY({}))".format(hash_size, alias) -}}
   63 +                 {{- "\n)) AS TEXT) AS {}".format(alias) -}}
64 64               {%- else -%}
65    -                 {{- "\n), '{}')) AS BINARY({}))".format(all_null | join(""), hash_size, alias) -}}
   65 +                 {{- "\n), '{}')) AS TEXT) AS {}".format(all_null | join(""), alias) -}}
66 66               {%- endif -%}
67 67           {%- else -%}
68 68
```

Можно ли использовать automate-dv для создания Data Vault в Clickhouse?

automate-dv Platform Support

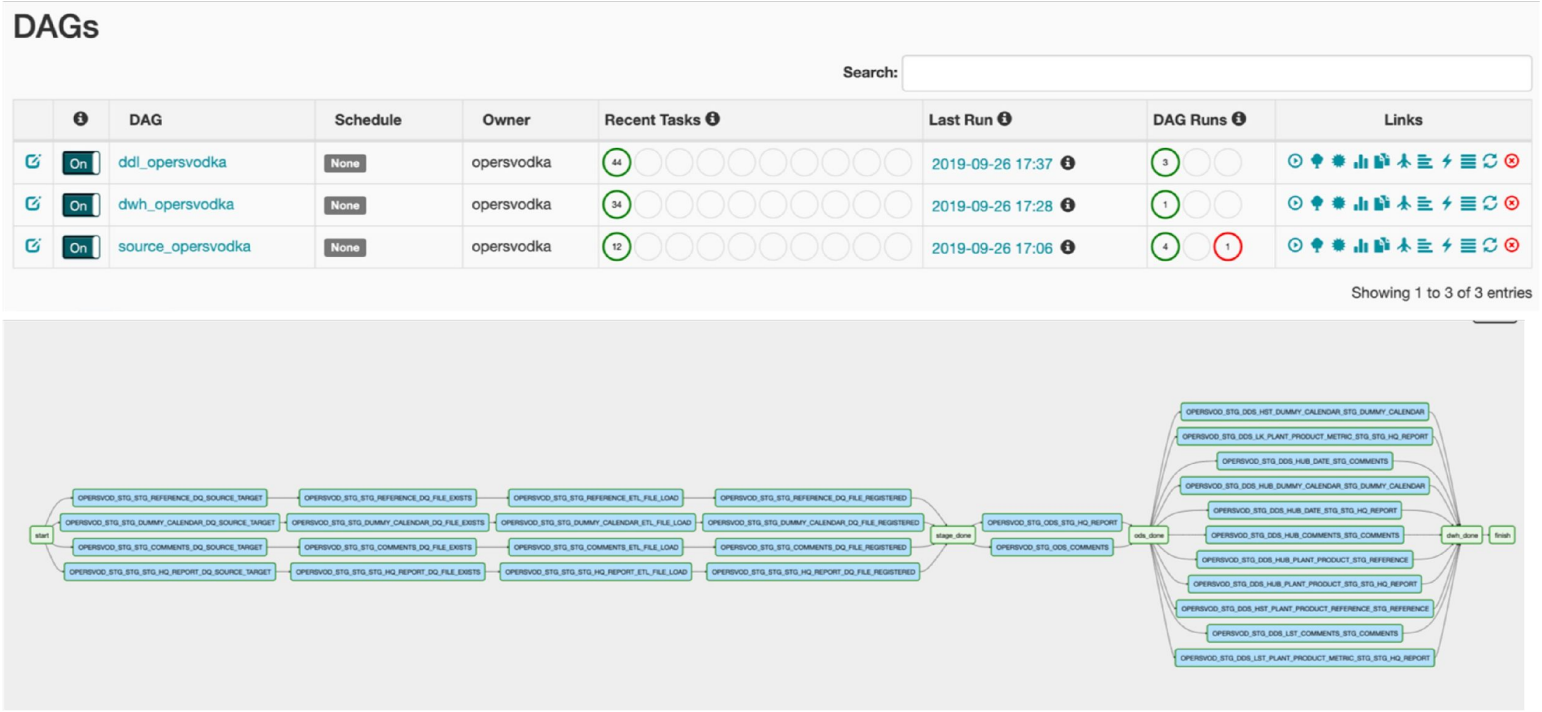
Macro/Template	Snowflake	Google BigQuery	MS SQL Server	Databricks	Postgres	Redshif
hash	✓	✓	✓	✓	✓	⊖
stage	✓	✓	✓	✓	✓	⊖
hub	✓	✓	✓	✓	✓	⊖
link	✓	✓	✓	✓	✓	⊖
sat	✓	✓	✓	✓	✓	⊖
t_link	✓	✓	✓	✓	✓	⊖
eff_sat	✓	✓	✓	✓	✓	⊖
ma_sat	✓	✓	✓	✓	✓	⊖
xts	✓	✓	✓	✓	✓	⊖
pit	✓	✓	✓	✓	✓	⊖
bridge	✓	✓	✓	✓	✓	⊖

https://automate-dv.readthedocs.io/en/latest/macros/#platform-support

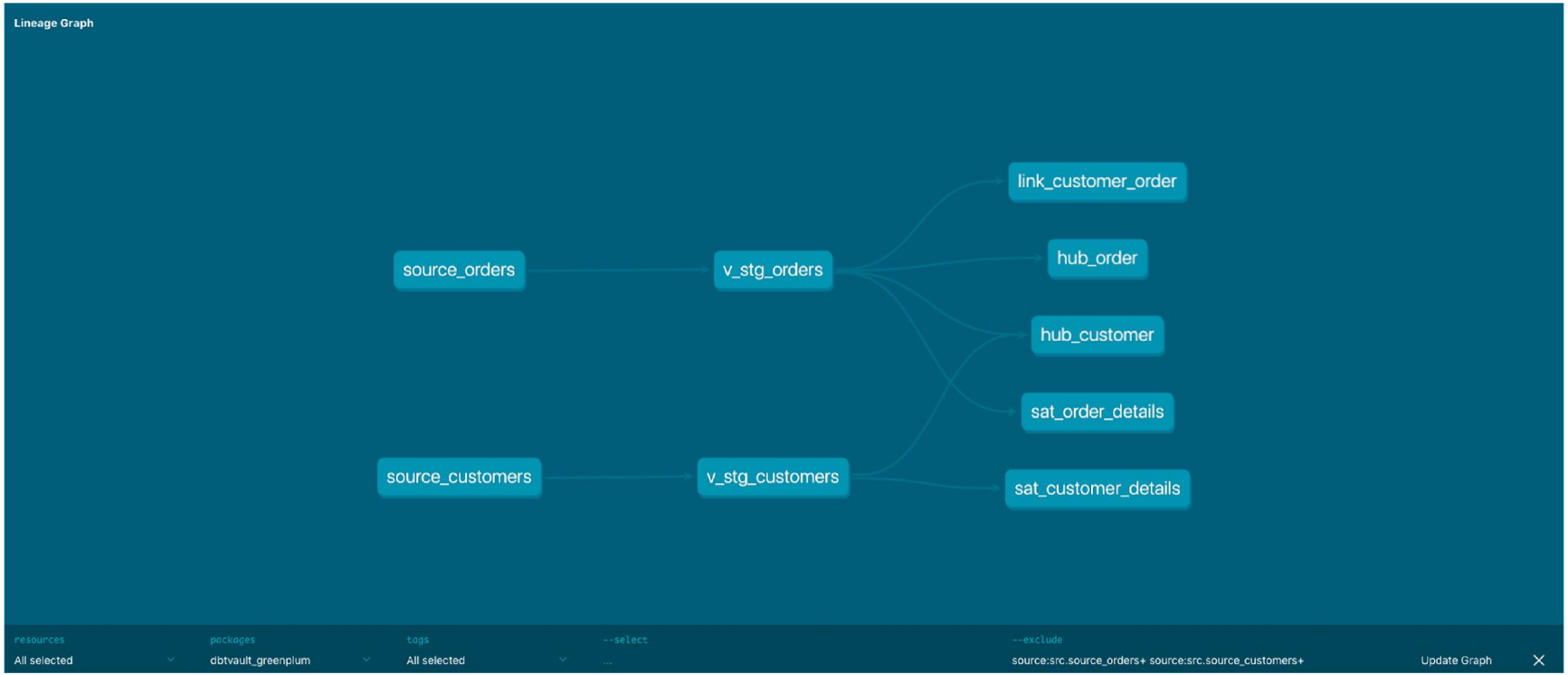
Оркестрация кодогенерации

Как управлять порядком загрузок и зависимостями?

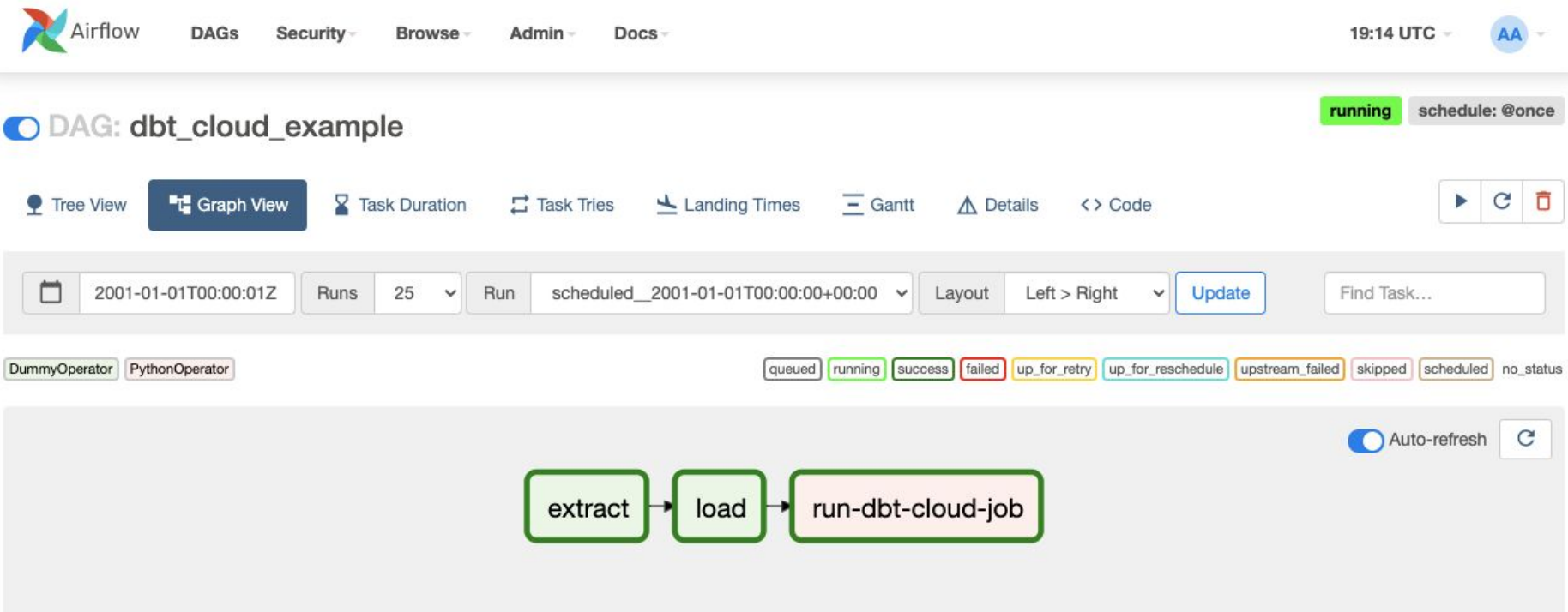
DAGs: Управление цепочками и зависимостями



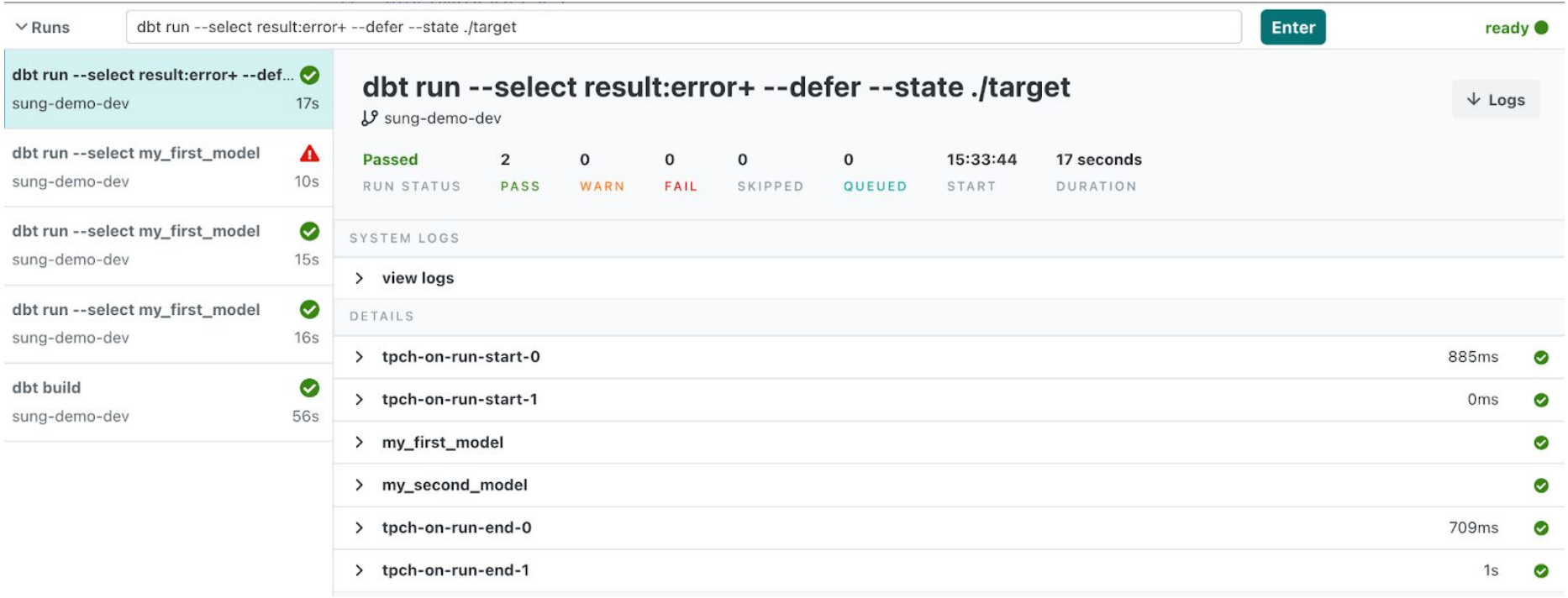
Data Build Tool DAG



Airflow + dbt cloud

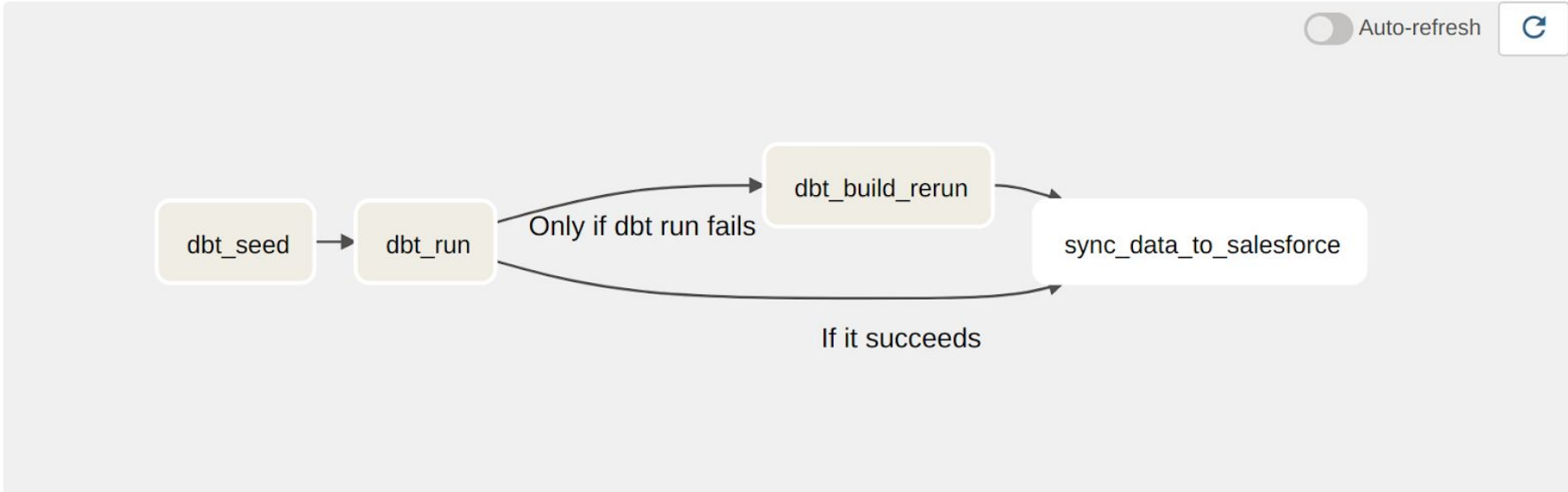


Airflow + dbt cloud



BashOperator
HightouchTriggerSyncOperator

queued running success failed up_for_retry up_for_reschedule upstream_failed skipped scheduled deferred
no_status



Example

```
from airflow import DAG
from airflow.operators.bash_operator import BashOperator
from datetime import datetime, timedelta
default_args = {
    'owner': 'airflow',
    'depends_on_past': False,
    'start_date': datetime(2023, 5, 1),
    'retries': 1,
    'retry_delay': timedelta(minutes=5),
}
# Create the DAG with the specified schedule interval
dag = DAG('dbt_dag', default_args=default_args, schedule_interval=timedelta(days=1))
# Define the dbt run command as a BashOperator
run_dbt_model = BashOperator(
    task_id='run_dbt_model',
    bash_command='dbt run',
    dag=dag
)
```

Example: dbt DAG with parameters

```
default_args = {
    'owner': 'airflow',
    'depends_on_past': False,
    'start_date': datetime(2023, 5, 1),
    'retries': 1,
    'retry_delay': timedelta(minutes=5),
}
dag = DAG('dbt_dag_with_params', default_args=default_args, schedule_interval=timedelta(days=1))
# Define the dbt run command with a parameter named "dataset"
run_dbt_model_with_params = BashOperator(
    task_id='run_dbt_model_with_params',
    bash_command='dbt run --var dataset=my_dataset',
    dag=dag
)
```

Рефлексия

Рефлексия

С какими впечатлениями уходите с вебинара?

Как будете применять на практике то,
что узнали на вебинаре?

Заполните, пожалуйста, опрос о занятии по ссылке в чате

Спасибо за внимание!

Приходите на следующие вебинары

Андрей Тюрин

DWH Analyst

- 4 года опыта Аналитиком хранилищ данных
- Сейчас работаю в транспортной отрасли
- Ранее : на проектах крупных банков

LinkedIn