tech

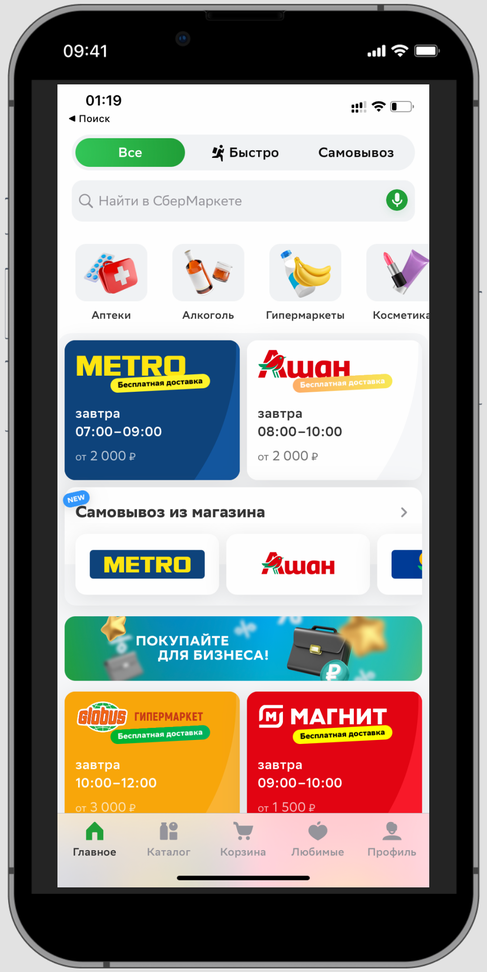
Визуализация для ELT-процессов в DWH

Как мы готовили у себя DBT и какие ошибки допустили.

14.09.2023

Содержание

СберМаркет – сервис доставки продуктов и товаров



> 32k

Магазинов

160

Ретейлеров

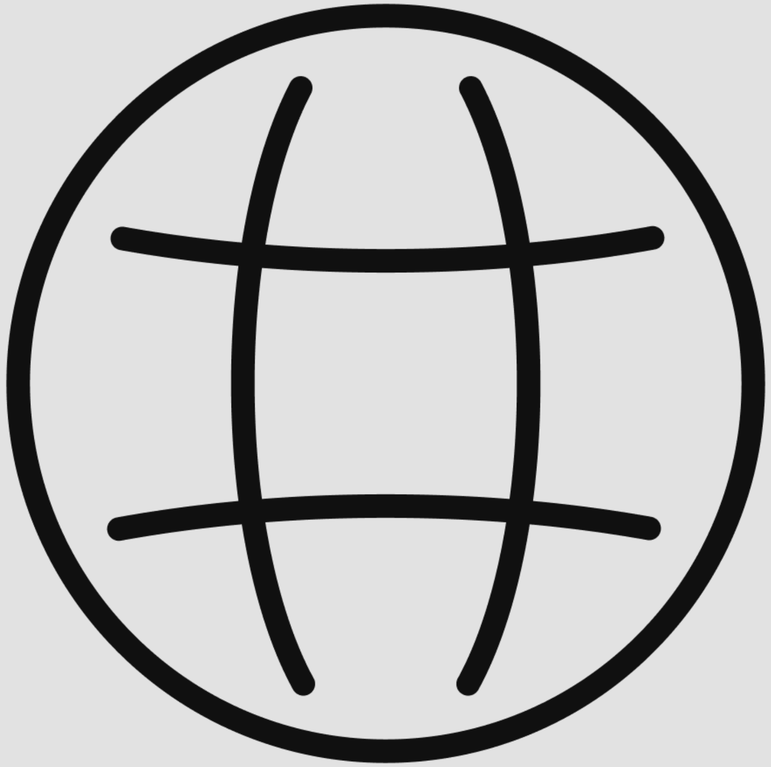
> 160

Городов

> 200kk

Количество SKU
(товарных позиций)

О данных



> 200Тб

DataLake

> 150

Пользователей GP

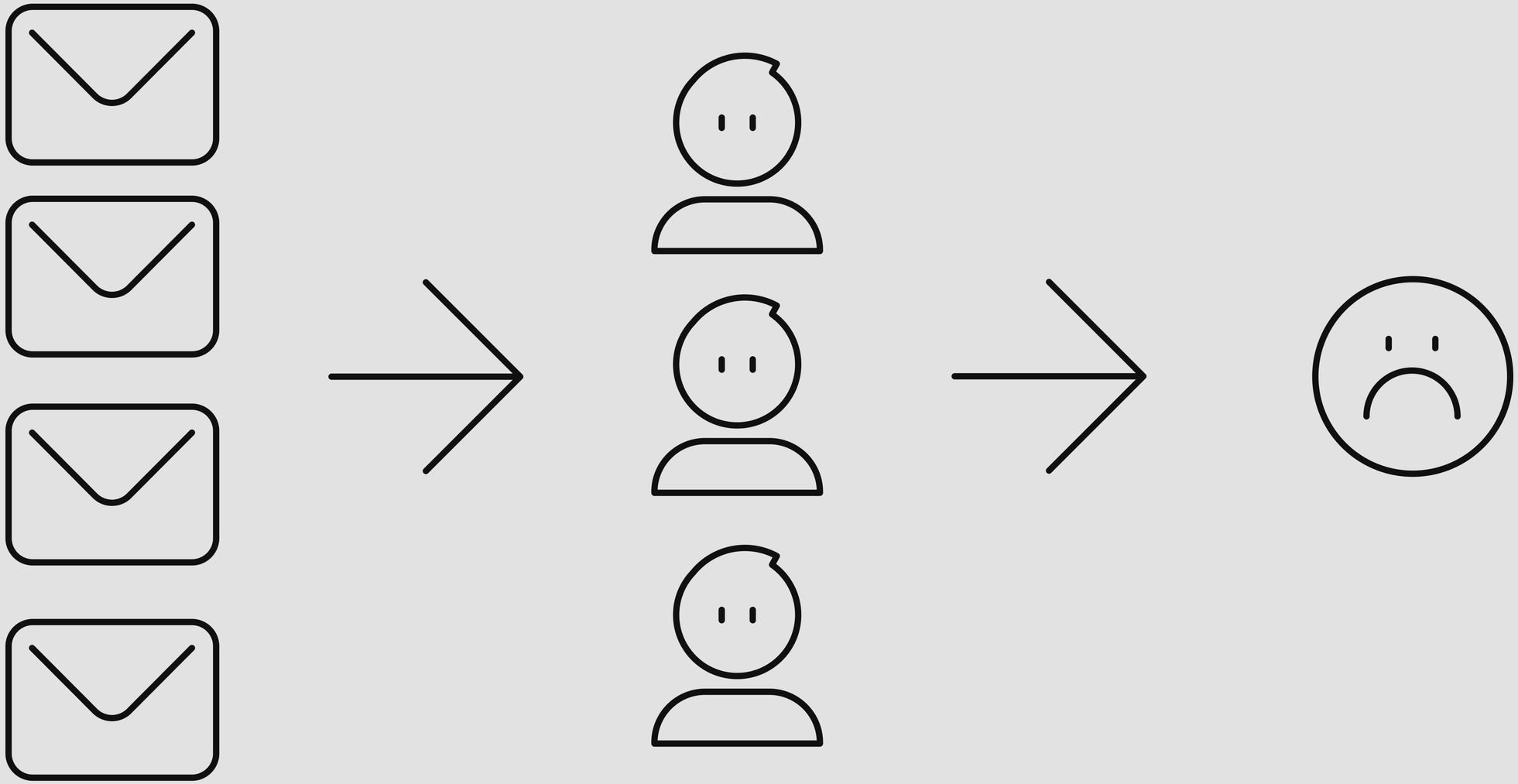
> 10Тб

Хранилище

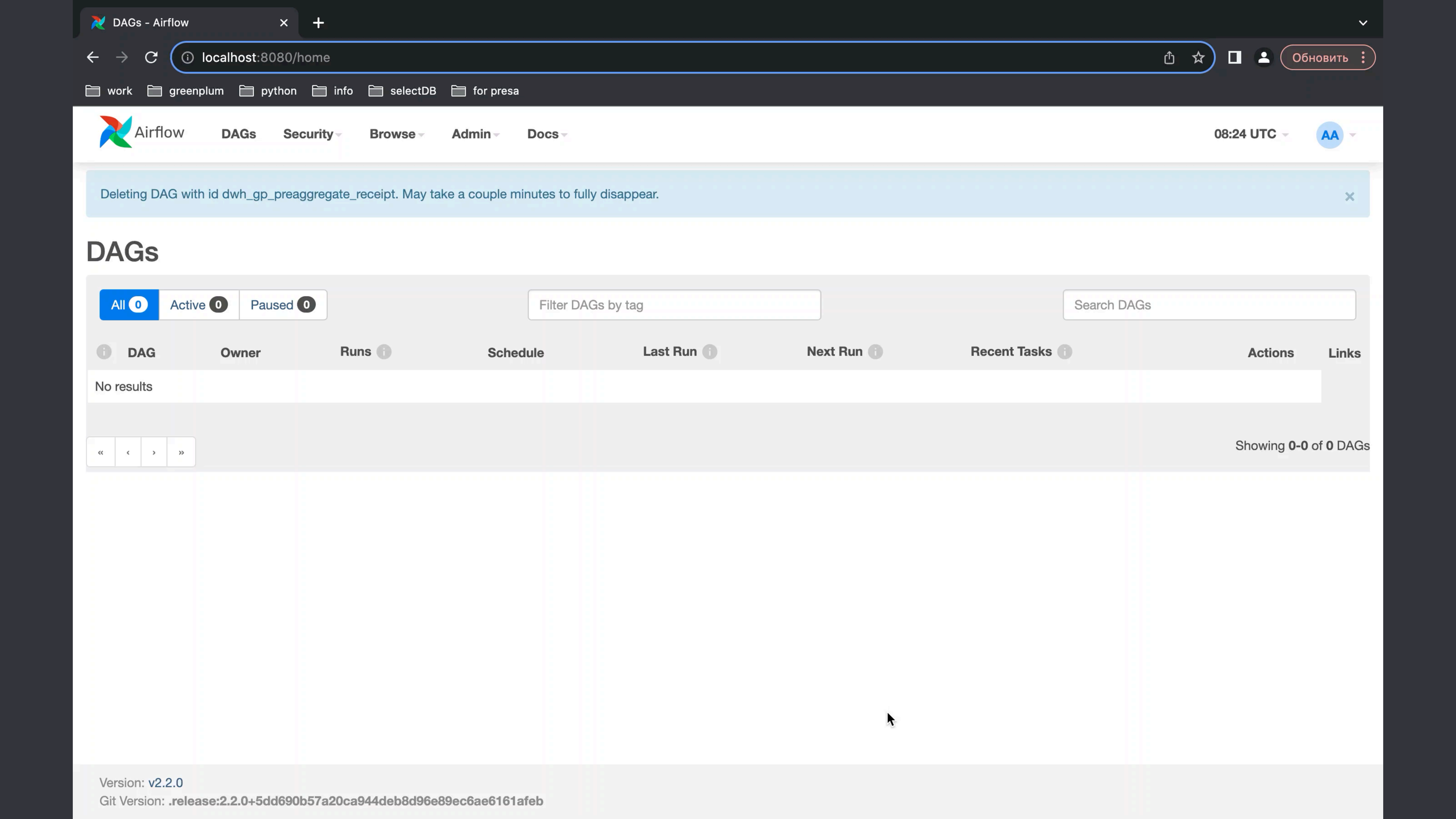
> 100

Витрин

Если бы инженер сам доставлял витрину



Предоставить инструмент

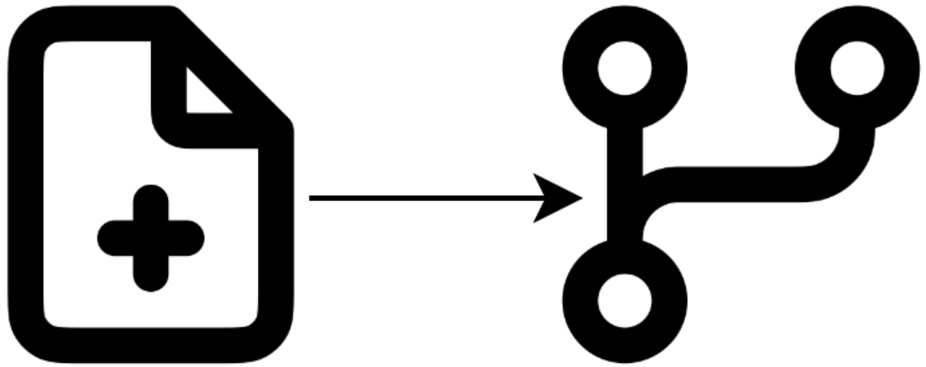

DAGs - Airflow
localhost:8080/home
Обновить
work
greenplum
python
info
selectDB
for presa
Airflow
DAGs
Security
Browse
Admin
Docs
08:24 UTC
AA
Deleting DAG with id dwh_gp_preaggregate_receipt. May take a couple minutes to fully disappear.
DAGs
All 0
Active 0
Paused 0
Filter DAGs by tag
Search DAGs
DAG
Owner
Runs
Schedule
Last Run
Next Run
Recent Tasks
Actions
Links
No results
Showing 0-0 of 0 DAGs
Version: v2.2.0
Git Version: .release:2.2.0+5dd690b57a20ca944deb8d96e89ec6ae6161afeb

Границы применения

Не подойдет для

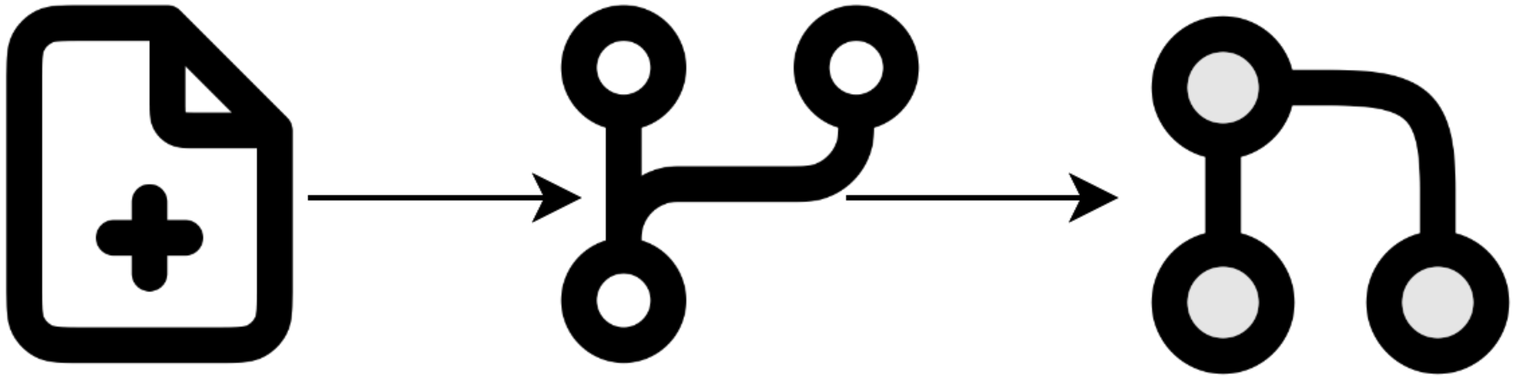
- Наполнение сырых данных
- Не хватает SQL

Наше решение до DBT

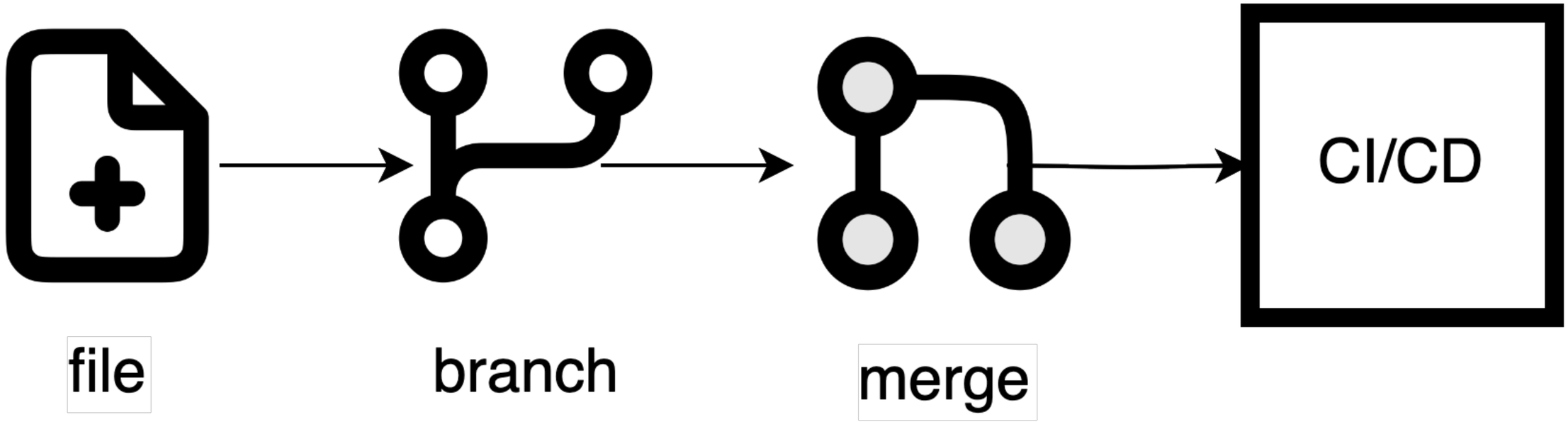


file branch

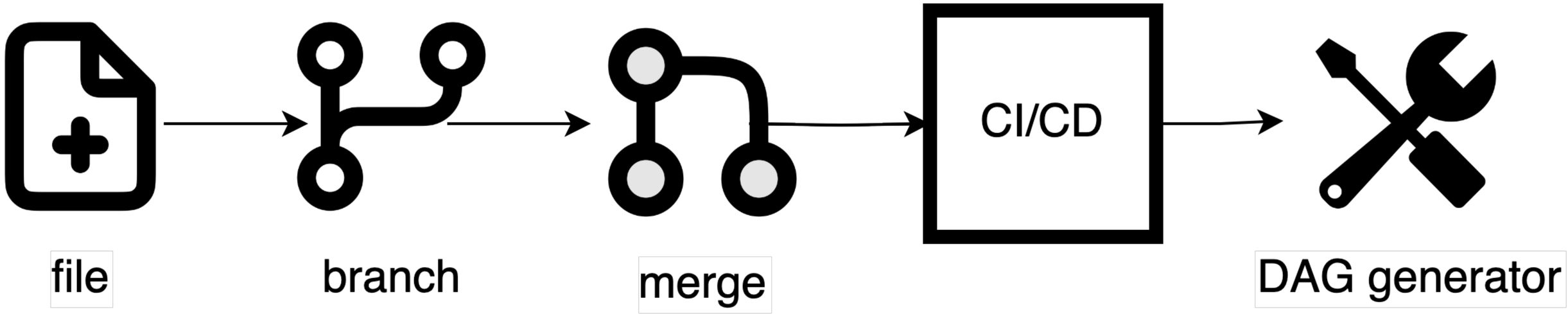
Наше решение до DBT



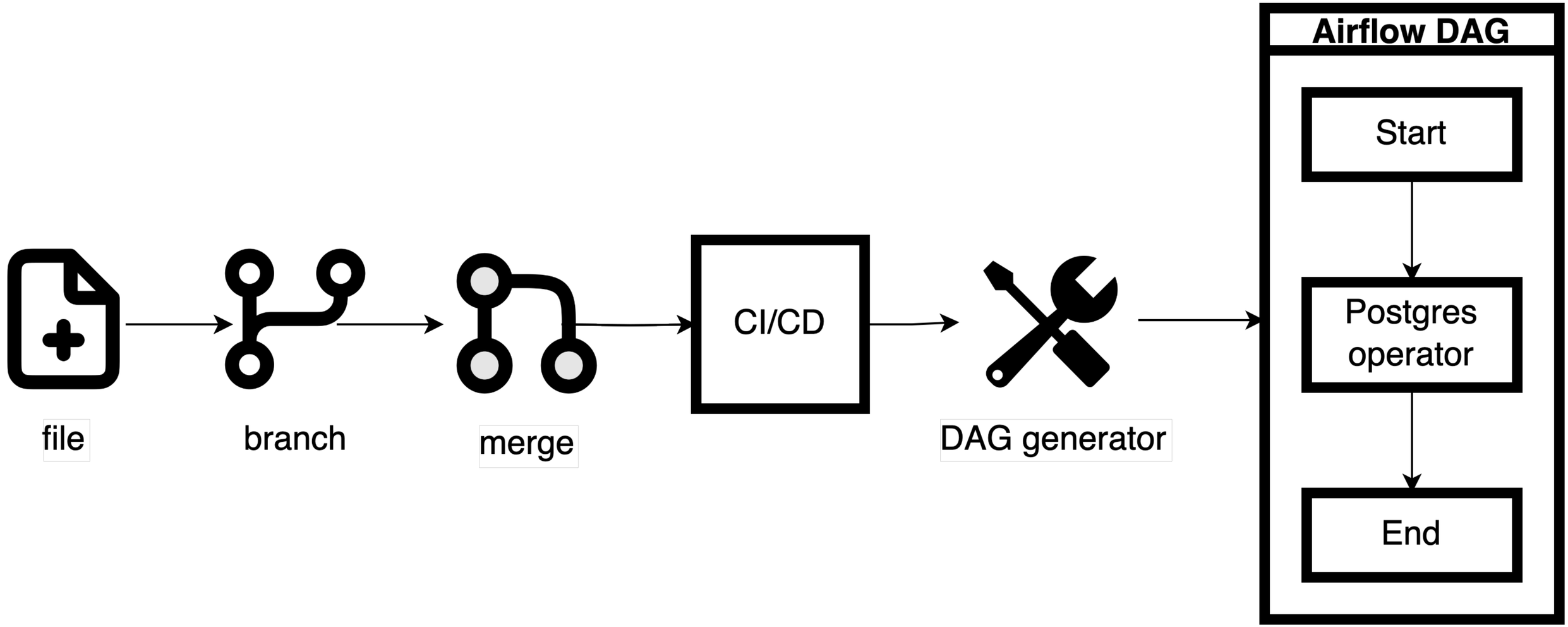
Наше решение до DBT



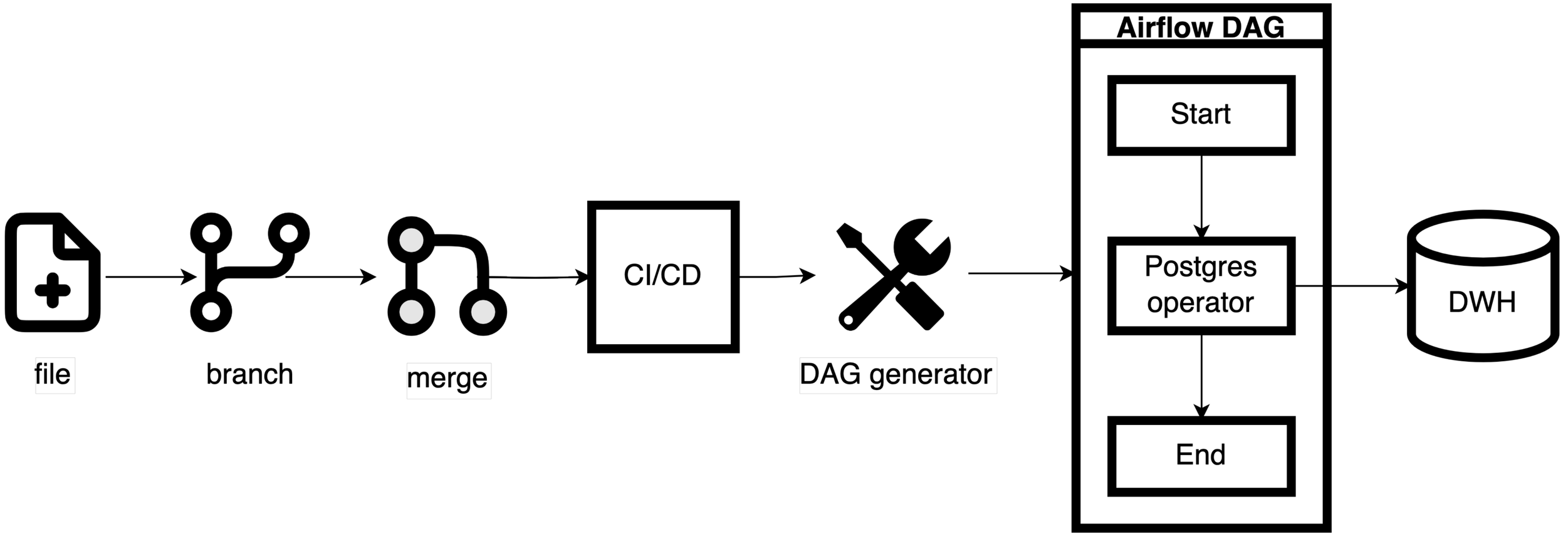
Наше решение до DBT

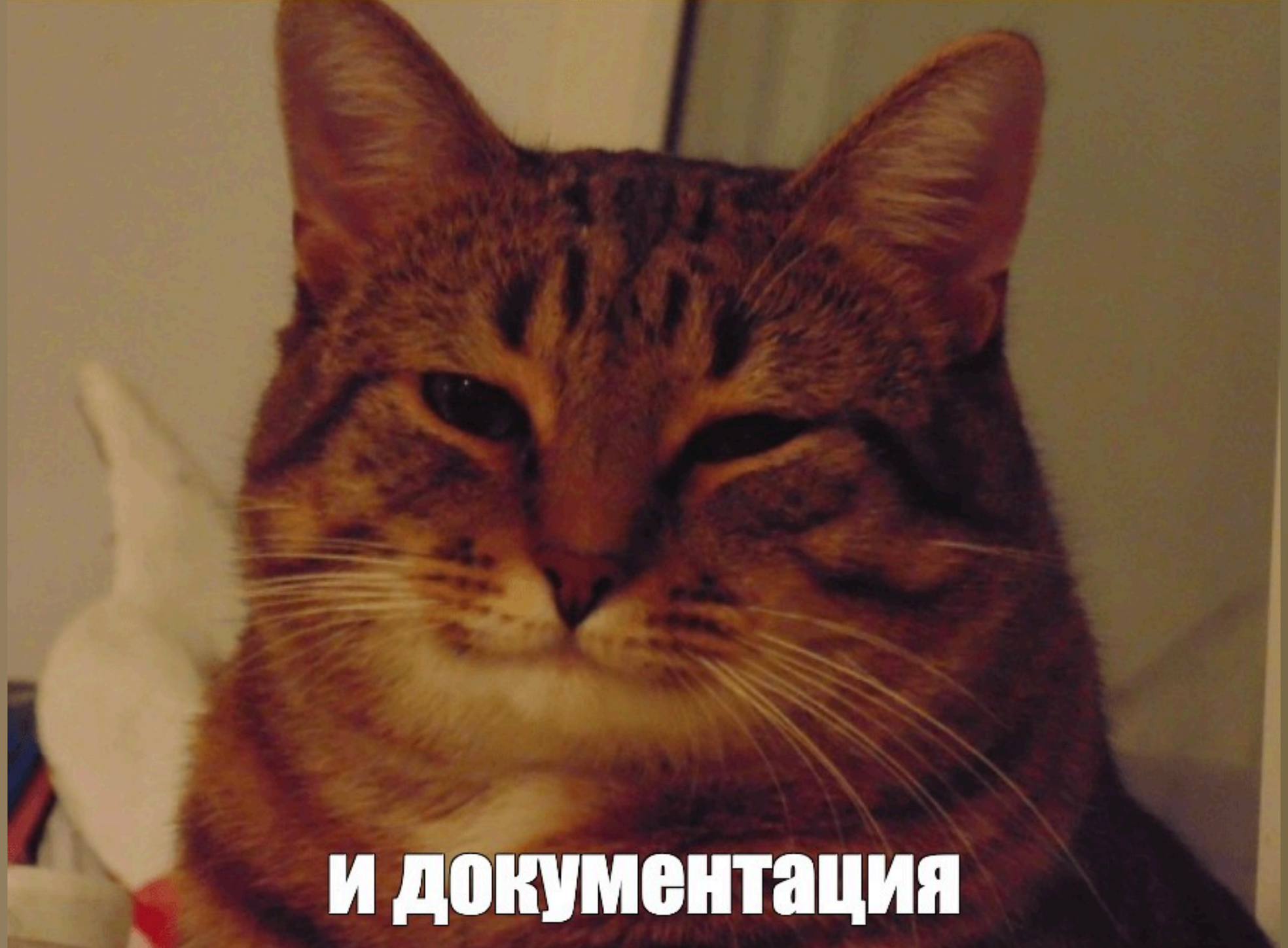


Наше решение до DBT



Наше решение до DBT




и документация


взять готовое
писать
самим

Data build tool

- DBT — это инструмент командной строки с открытым исходным кодом. Призван помочь более эффективно преобразовывать данные в хранилищах данных. В акрониме ELT (Extract, Load, Transform) DBT занимает этап T.
- Целью dbt является предоставление аналитикам возможности работать как инженеры-программисты

Чем привлек DBT

- Распространенность, развитие, обширное сообщества

Чем привлек DBT

- Распространенность, развитие, обширное сообщества
- Конфигурация на .yaml

Чем привлек DBT

- Распространенность, развитие, обширное сообщества
- Конфигурация на .yaml
- Возможность добавления тестов

Чем привлек DBT

- Распространенность, развитие, обширное сообщества
- Конфигурация на .yaml
- Возможность добавления тестов
- Логика не в ХП

Чем привлек DBT

- Распространенность, развитие, обширное сообщества
- Конфигурация на .yaml
- Возможность добавления тестов
- Логика не в ХП
- Метаданные

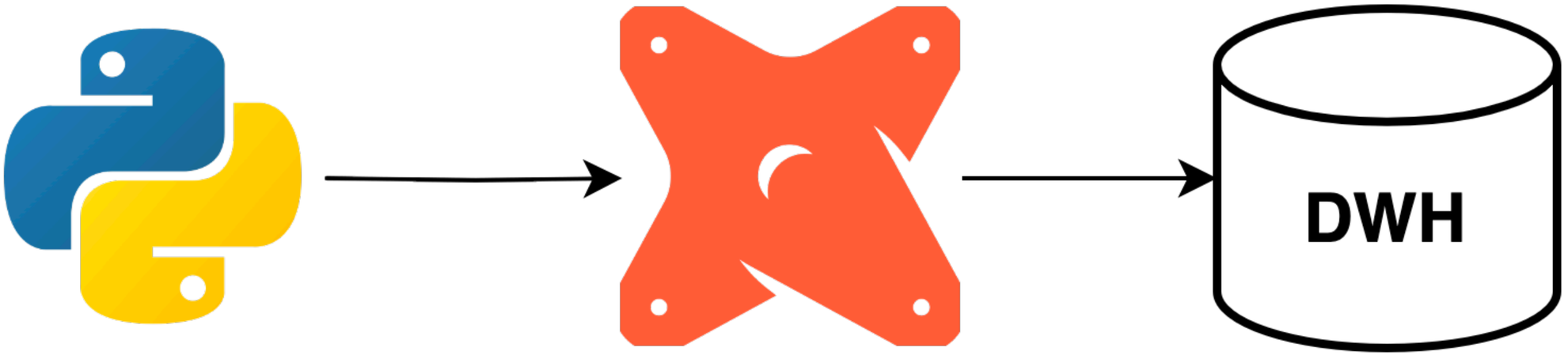
Чем привлек DBT

- Распространенность, развитие, обширное сообщества
- Конфигурация на .yaml
- Возможность добавления тестов
- Логика не в ХП
- Метаданные
- Графы зависимостей

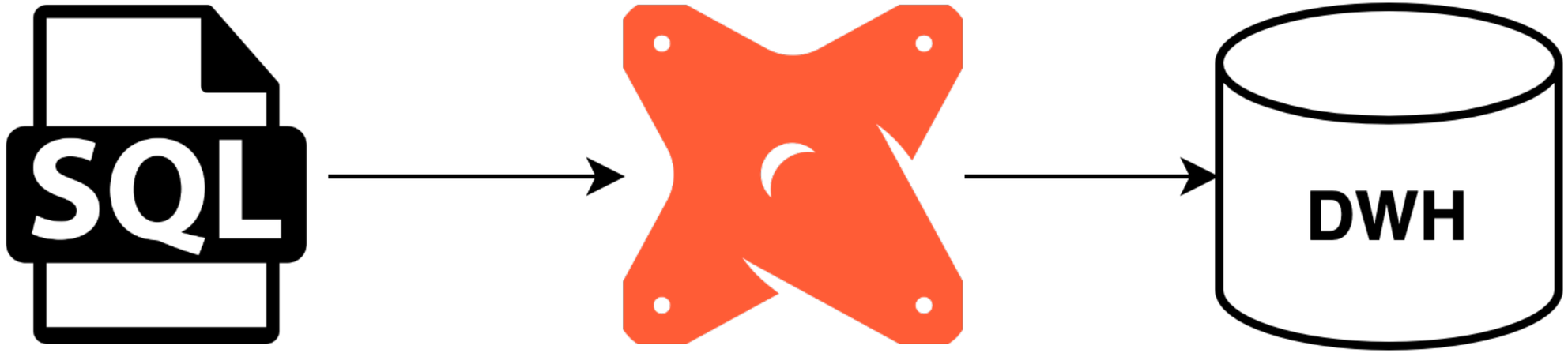
Чем привлек DBT

- Распространенность, развитие, обширное сообщества
- Конфигурация на .yaml
- Возможность добавления тестов
- Логика не в ХП
- Метаданные
- Графы зависимостей
- Визуализация и документация

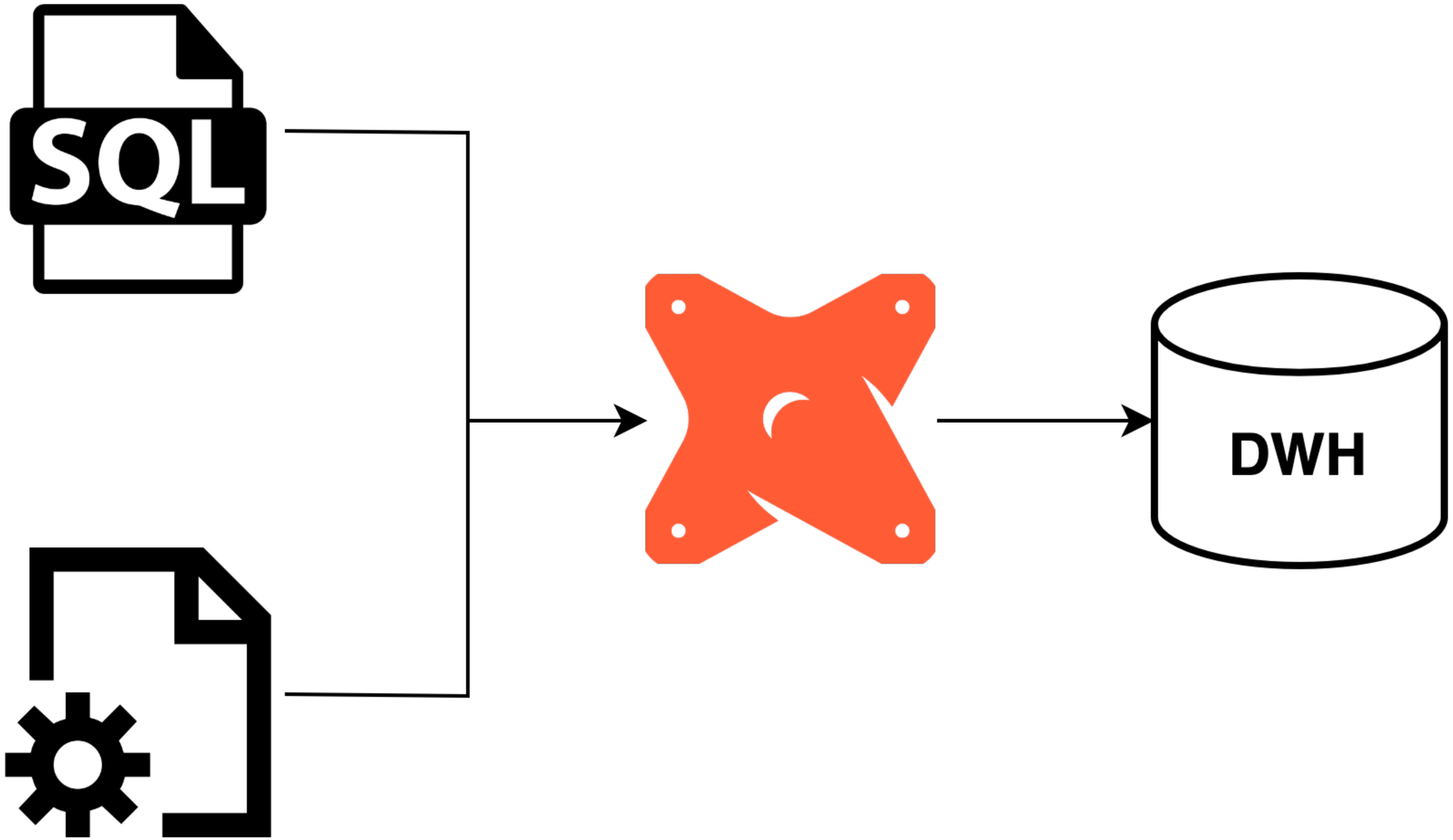
Как DBT работает



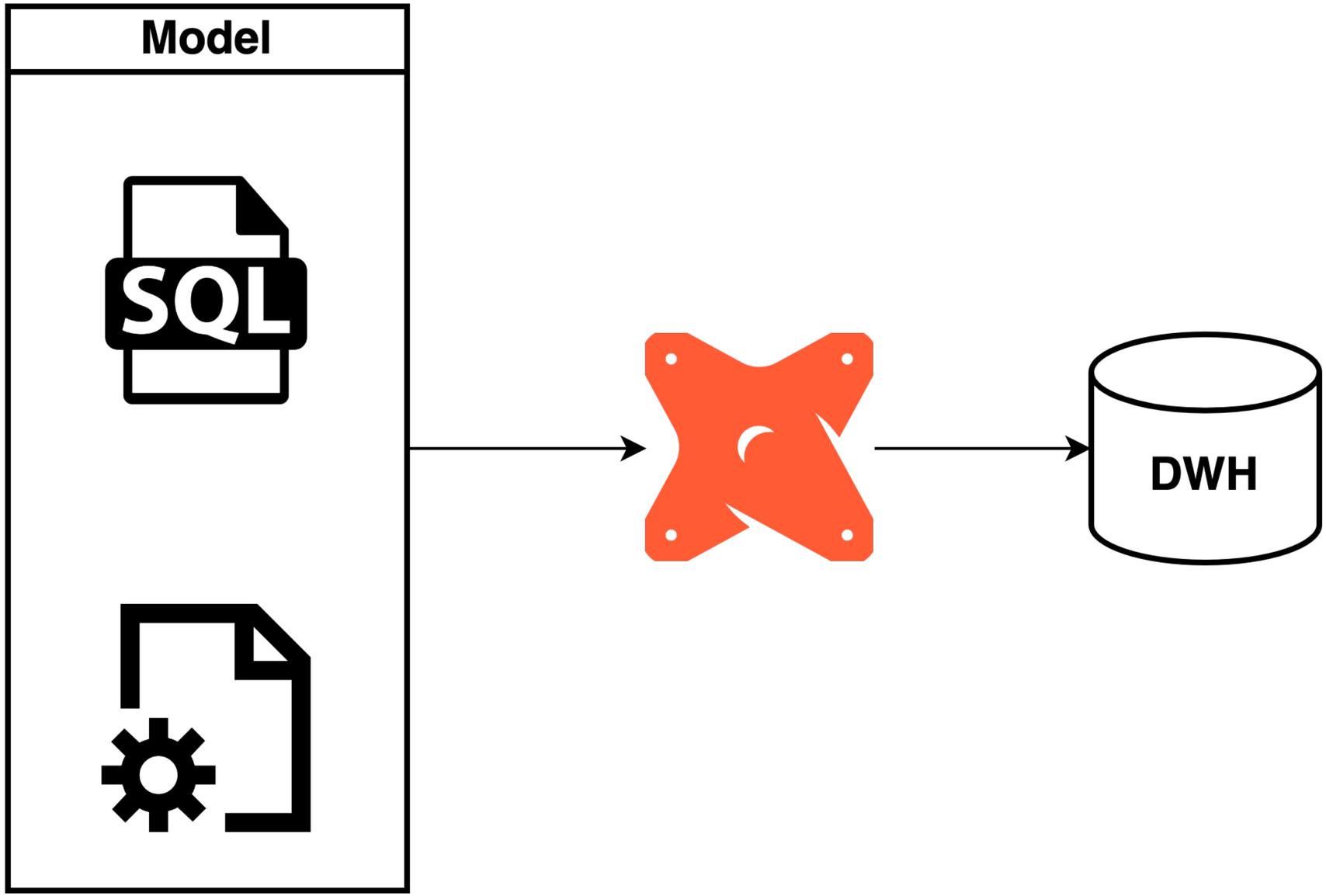
Как DBT работает



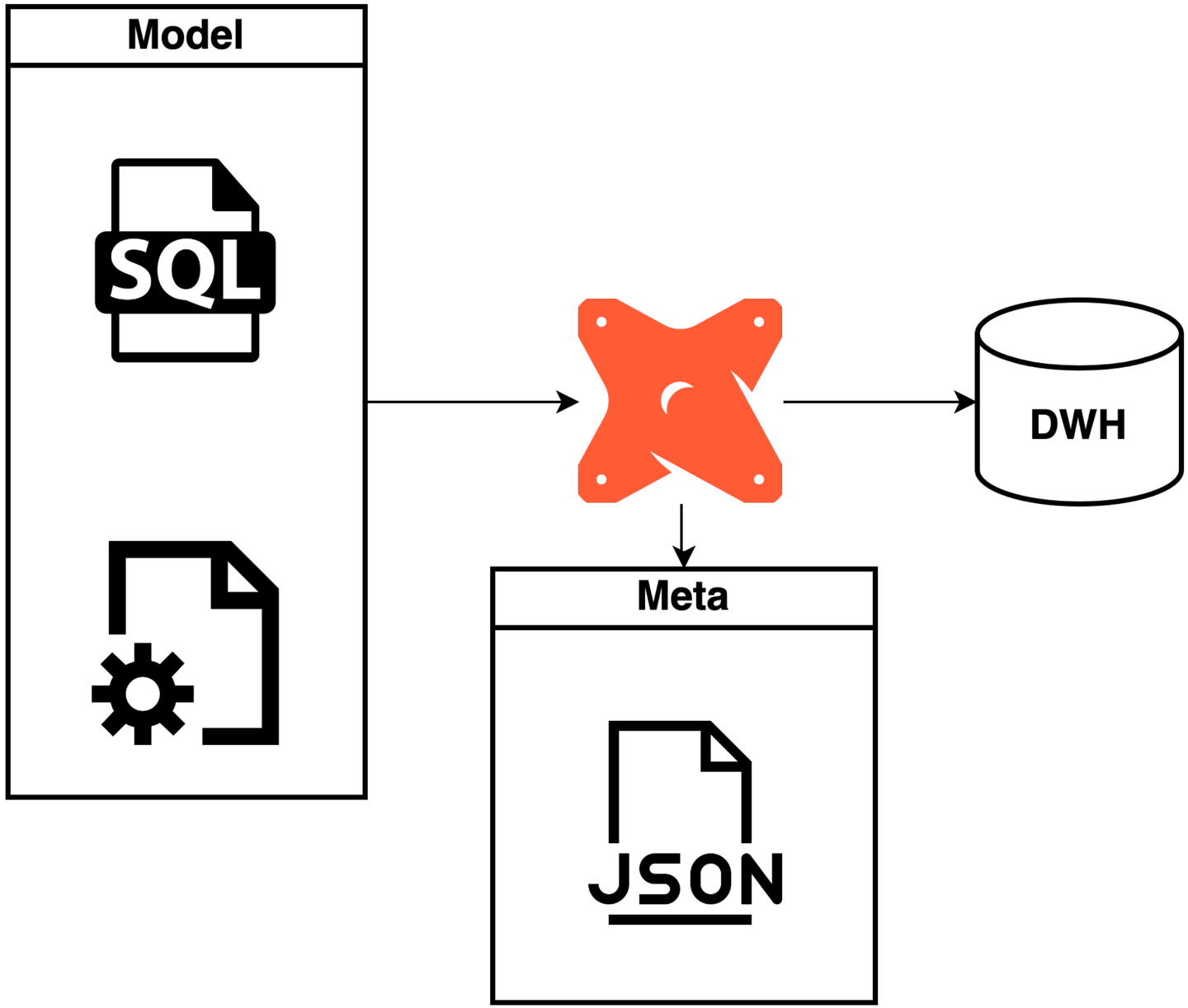
Как DBT работает



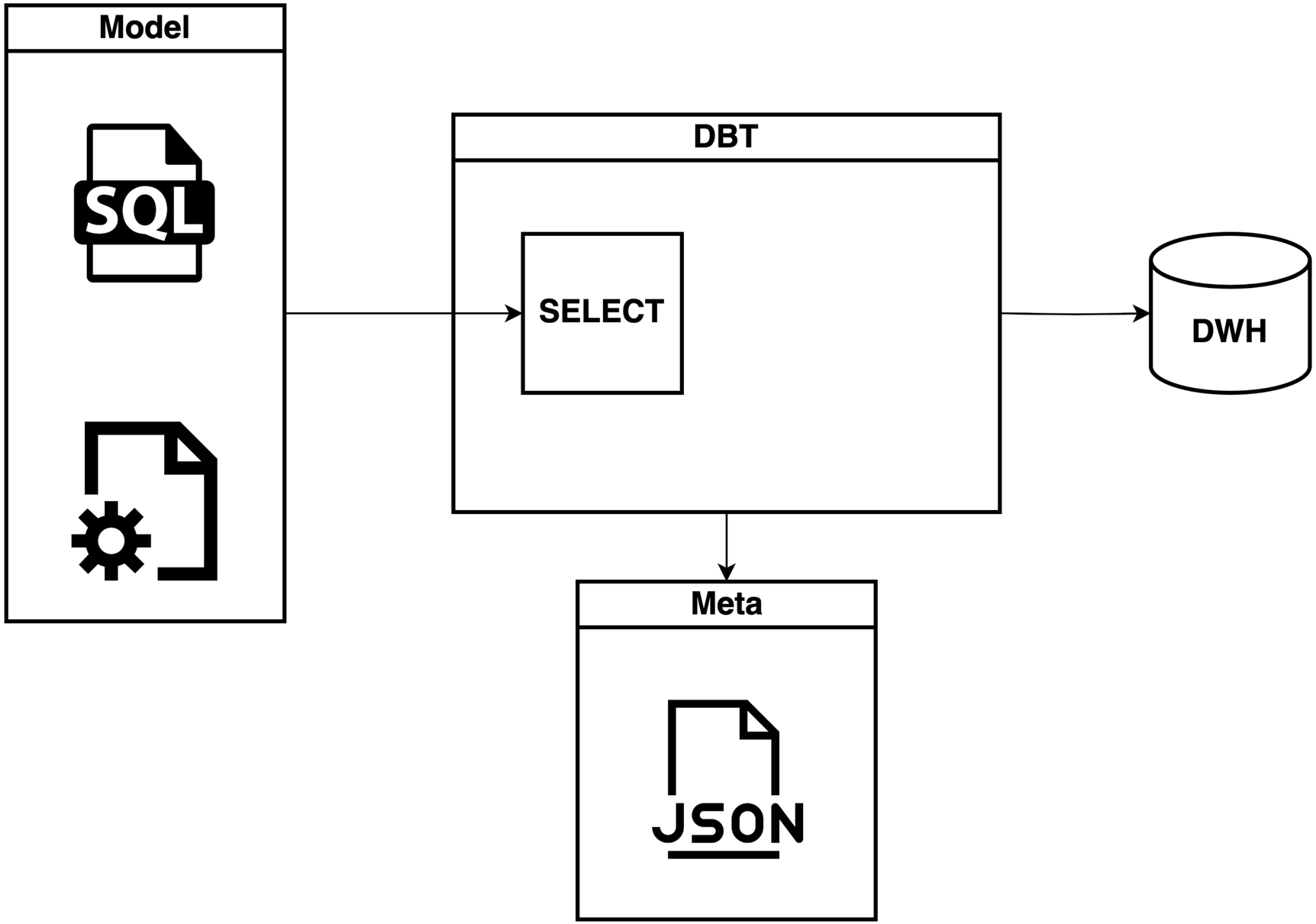
Как DBT работает



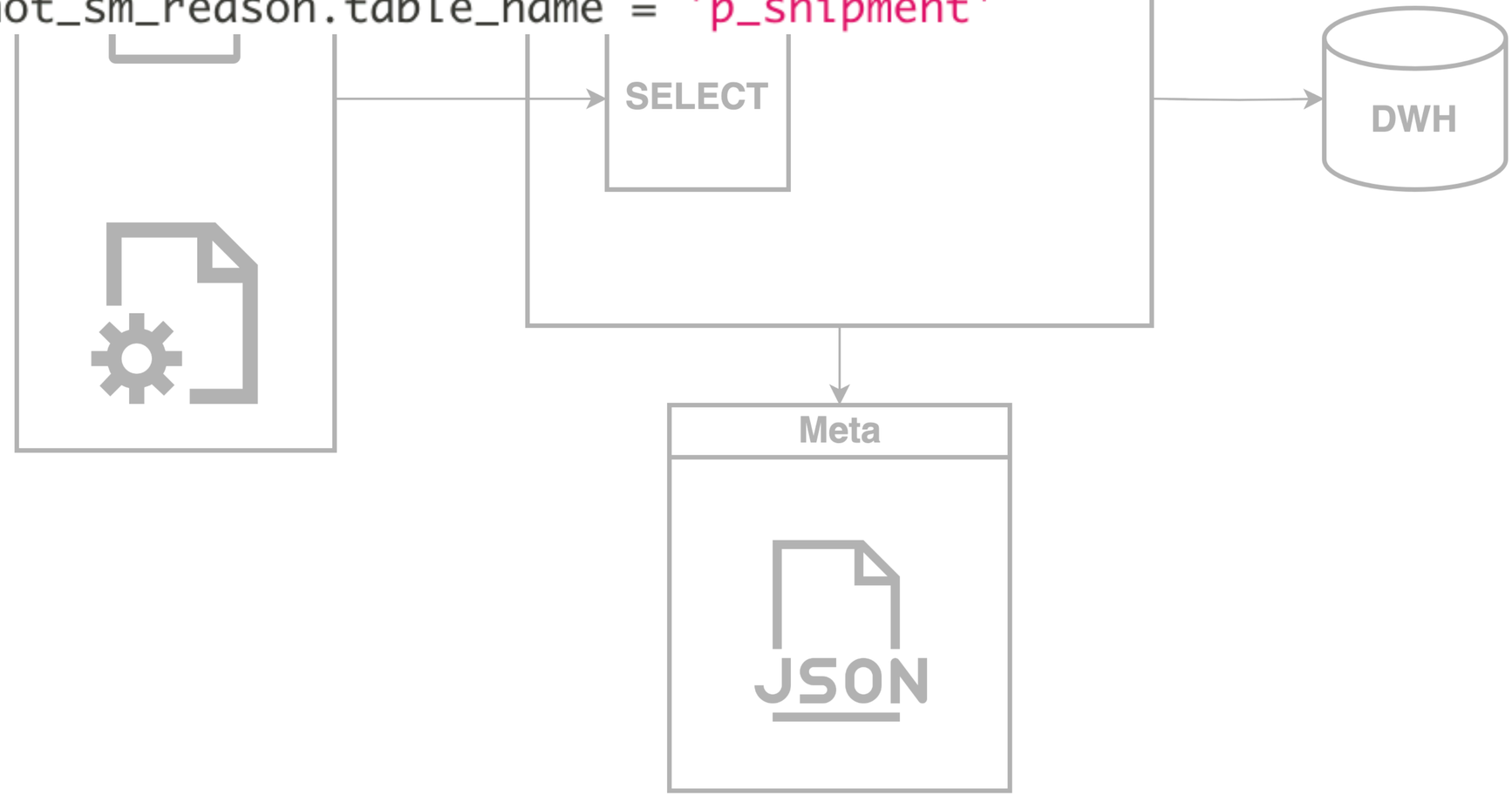
Как DBT работает



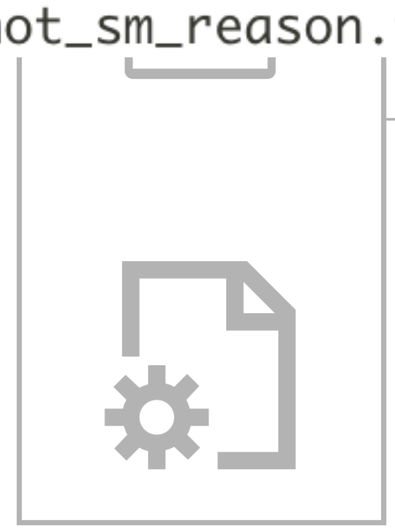
Как DBT работает



```
join {{ ref( 's_delivery_window') }} pwin
     on sdwch.previous_delivery_window_id = pwin.
     {{ actual_rows_by_source('pwin', 'INST') }}
left join {{ source('meta', 'all_layers_params')
     on not_sm_reason.layer_cd = 'dds'
     and not_sm_reason.table_name = 'p_shipment'
```



```
join {{ ref( 's_delivery_window') }} pwin
     on sdwch.previous_delivery_window_id = pwin.
     {{ actual_rows_by_source('pwin', 'INST') }}
left join {{ source('meta', 'all_layers_params')
      on not_sm_reason.layer_cd = 'dds'
      and not_sm_reason.table_name = 'p_shipment'
```



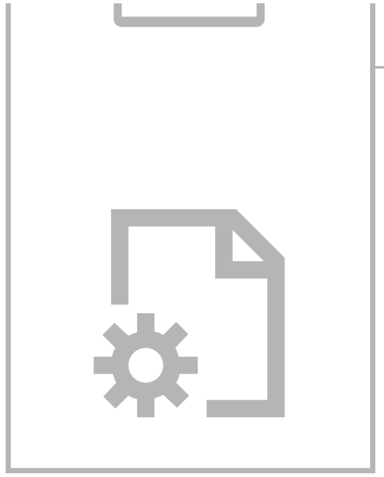
SELECT

DWH

```
  join "dwh"."dds"."s_delivery_window" pwin
       on sdwch.previous_delivery_window_id = pwin.d

  and pwin.mt_src_cd = 'INST'
  and not pwin.mt_deleted_flg
  and pwin.mt_active_flg

  left join "dwh"."meta"."all_layers_params" not_sm_
       on not_sm_reason.layer_cd = 'dds'
       and not_sm_reason.table_name = 'p_shipment'
```

tech

```
join {{ ref( 's_delivery_window') }} pwin
     on sdwch.previous_delivery_window_id = pwin.
     {{ actual_rows_by_source('pwin', 'INST') }}
left join {{ source('meta', 'all_layers_params')
     on not_sm_reason.layer_cd = 'dds'
     and not_sm_reason.table_name = 'p_shipment'
```



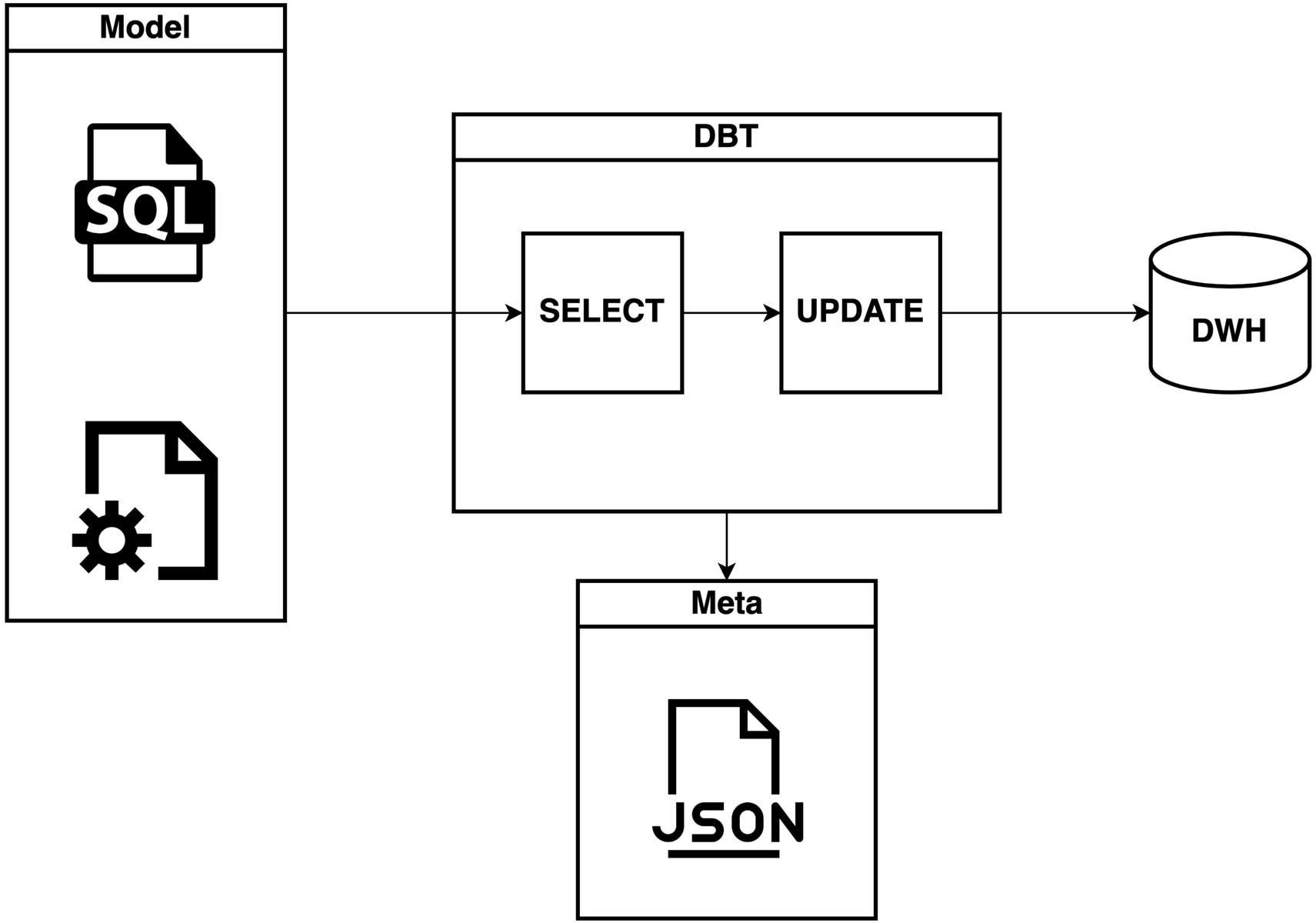
SELECT

DWH

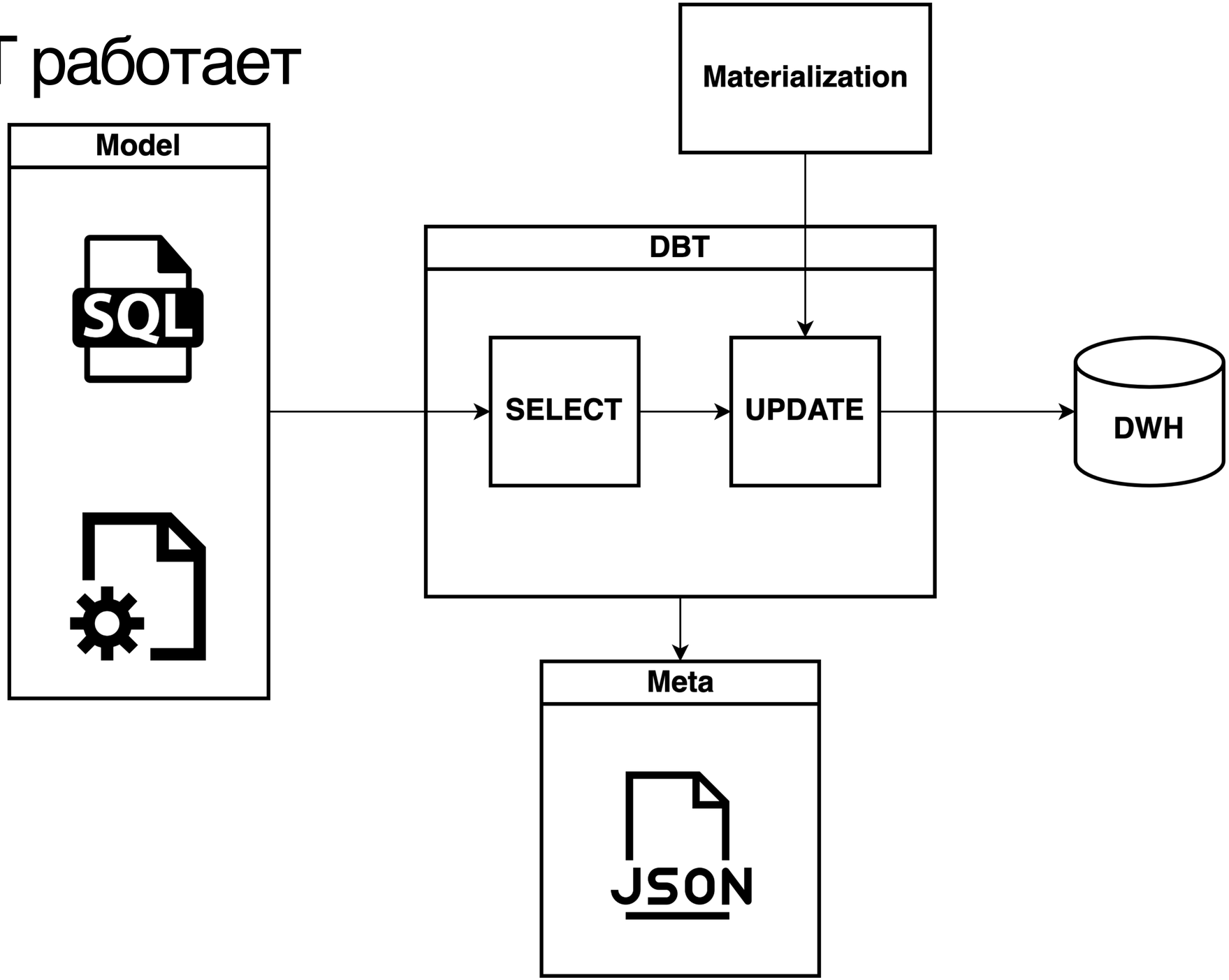
```
join "dwh"."dds"."s_delivery_window" pwin
     on sdwch.previous_delivery_window_id = pwin.d

and pwin.mt_src_cd = 'INST'
and not pwin.mt_deleted_flg
and pwin.mt_active_flg

left join "dwh"."meta"."all_layers_params" not_sm_
     on not_sm_reason.layer_cd = 'dds'
     and not_sm_reason.table_name = 'p_shipment'
```

tech

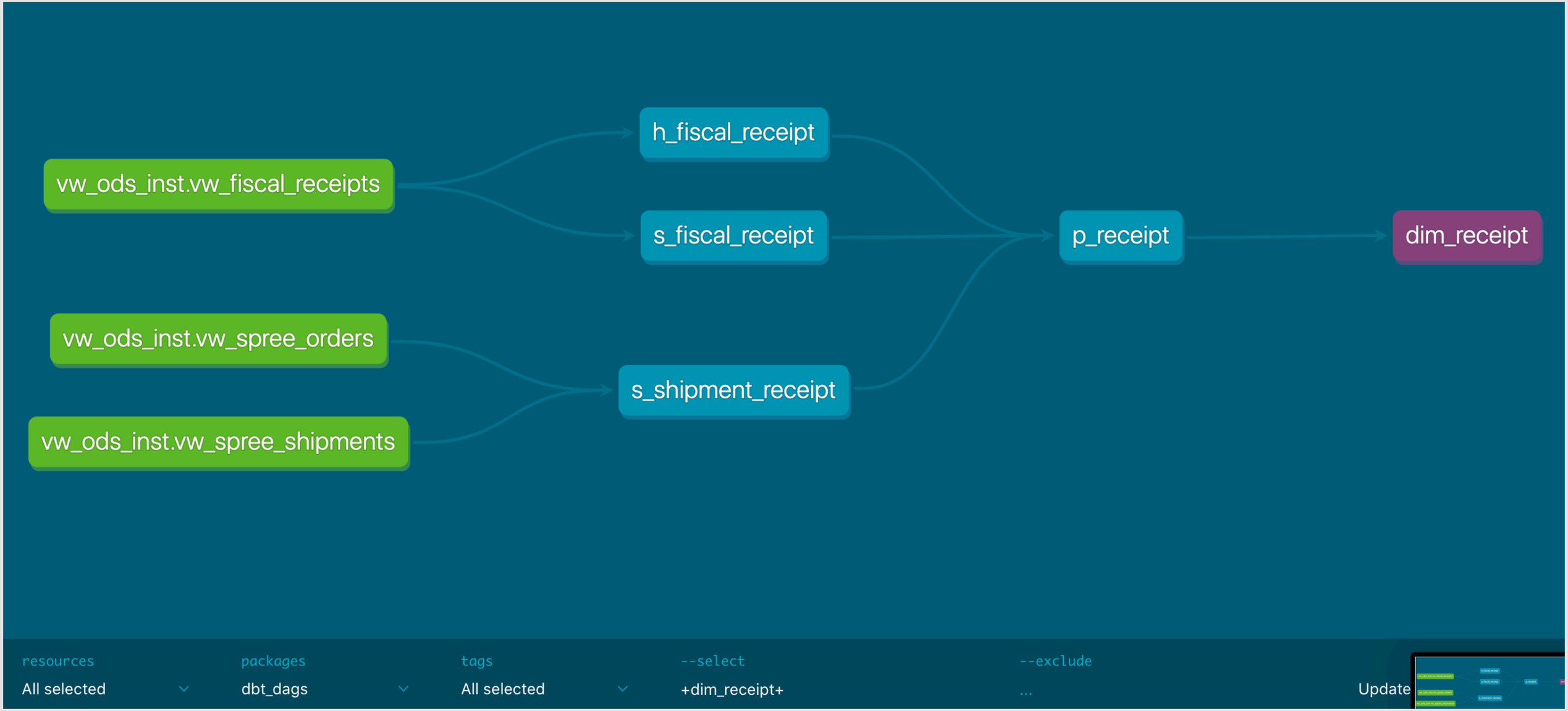
Как DBT работает



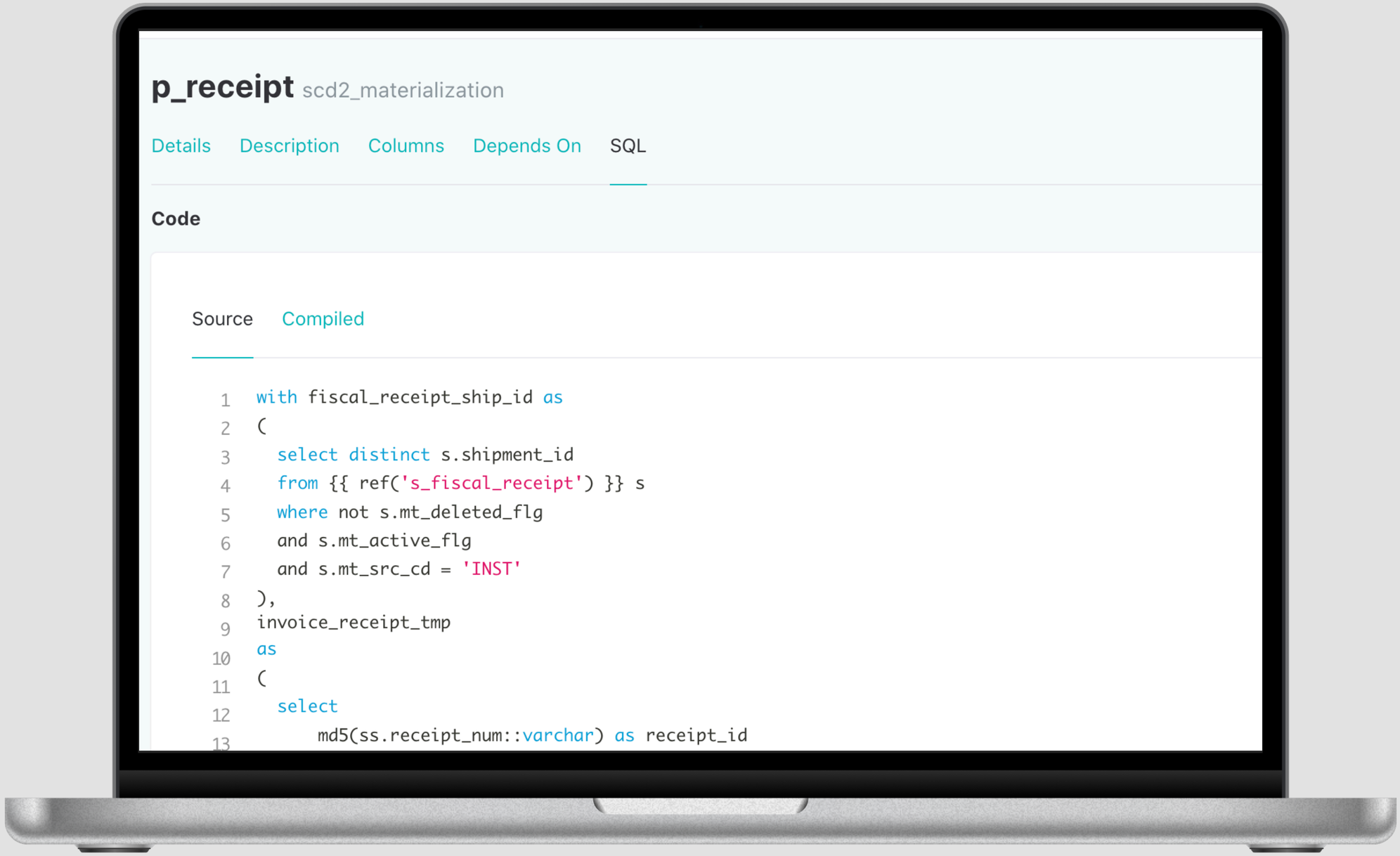
Как DBT работает



Граф зависимостей



Документация




aph
vw_ods_inst.vw_spree_promotion_actions
d_platform_brand
cert_orders_tmp
s_delivery_window
vw_ods_inst.vw_spree_retailers
d_api_client
retailer_group_tmp
gmv_for_ladder_tmp
ladder_commission_
h_shipment_inst
vw_ods_ncloud.vw_stores_fee
ladder_tmp
h_shipment_shop
vw_ods_inst.vw_spree_shipping_categories
d_delivery_category
l_adjustment_x_promo_action
ods_cpl.spasibo_transactions
h_store_inst
l_adjustment_x_shipment
p_shipment_num_items_cancel_replace_tmp
meta.all_layers_params
p_shipment
h_store_shop
meta.d_partner_subscription_packet_kind
d_partner_subscription_packet_kind
h_fiscal_receipt
h_shipment
s_fiscal_receipt
d_pay_state
s_shipment_receipt
h_retailer
vw_ods_inst.vw_tenants
h_store
p_shipment_delivery_nominal_cost_tmp
packages
tags
--select
--exclude
dbt_dags
All selected
++
...
Update

Раскатка и основные команды DBT

- Установка:

  pip install dbt

Раскатка и основные команды DBT

- Установка:

  pip install dbt

- Запуск проекта:

  dbt init

Раскатка и основные команды DBT

- Установка:

  pip install dbt
- Запуск проекта:

  dbt init
- Выполнение моделей:

  dbt run
- Выполнить конкретную модель:

  dbt run -m

Раскатка и основные команды DBT

- Установка:

  pip install dbt
- Запуск проекта:

  dbt init
- Выполнение моделей:

  dbt run
- Выполнить конкретную модель:

  dbt run -m
- Проверка:

  dbt parse
- Компиляция:

  dbt compile

Раскатка и основные команды DBT

- Установка:

  pip install dbt
- Запуск проекта:

  dbt init
- Генерация документации:

  dbt docs generate
- Запуск веб-клиента с докой:

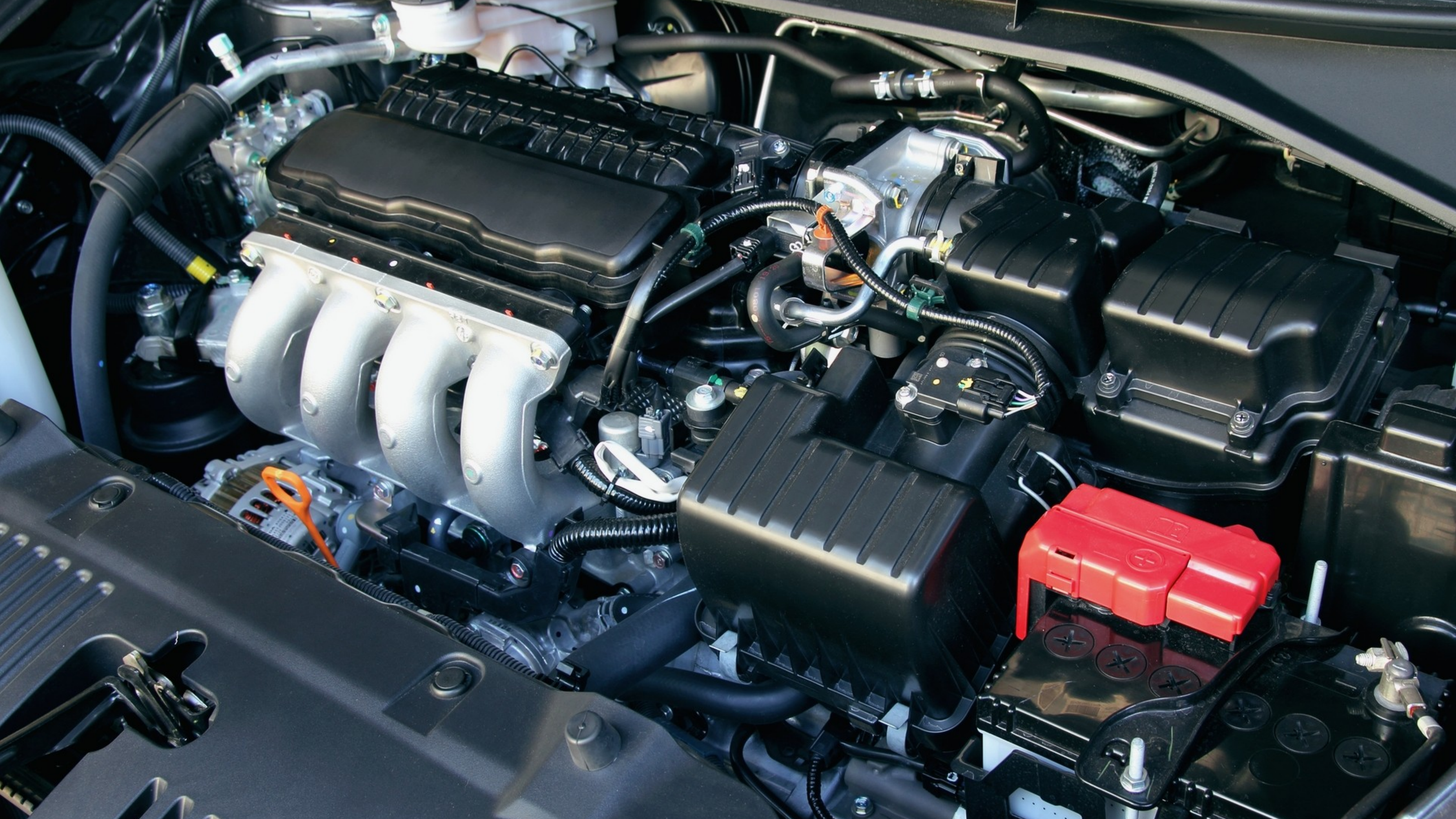
  dbt docs serve
- Выполнение моделей:

  dbt run
- Выполнить конкретную модель:

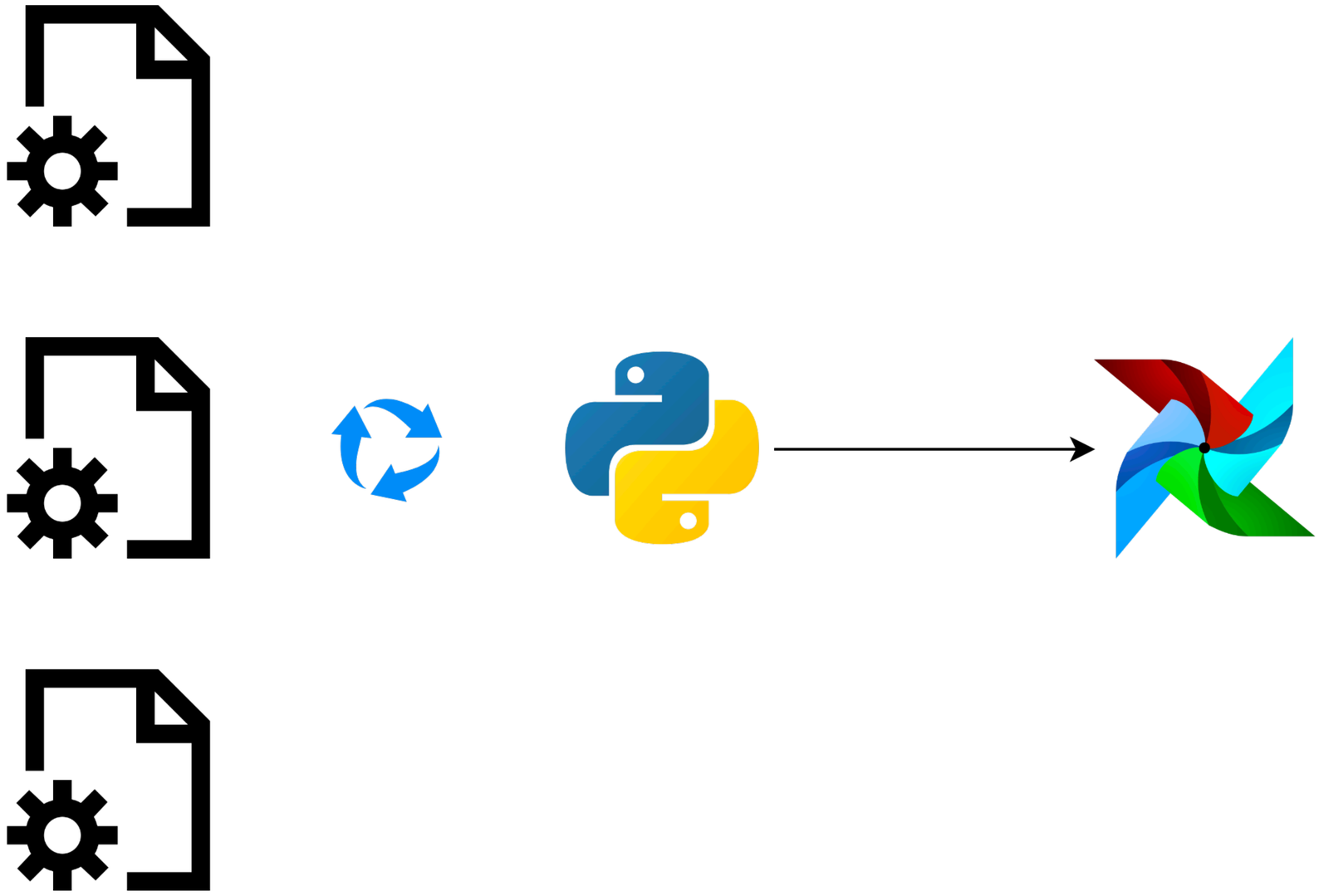
  dbt run -m
- Проверка:

  dbt parse
- Компиляция:

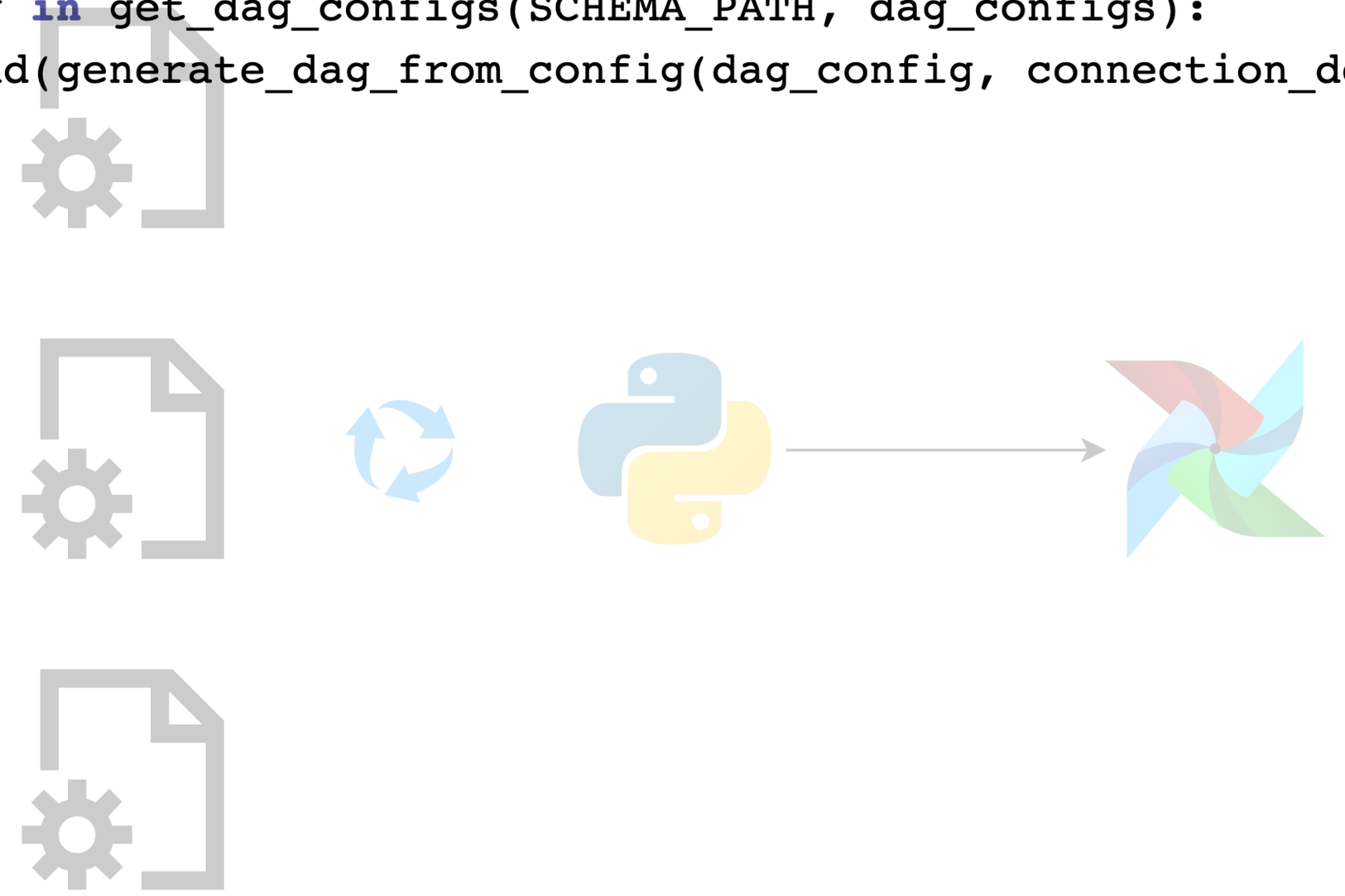
  dbt compile



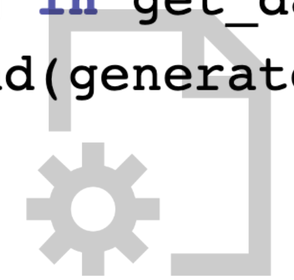
Даг-генератор

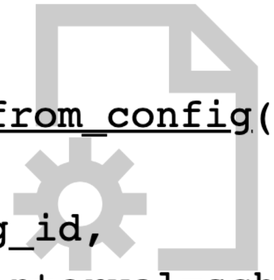


```
for dag_config in get_dag_configs(SCHEMA_PATH, dag_configs):
    dags.append(generate_dag_from_config(dag_config, connection_details))
return dags
```

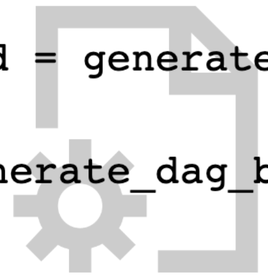


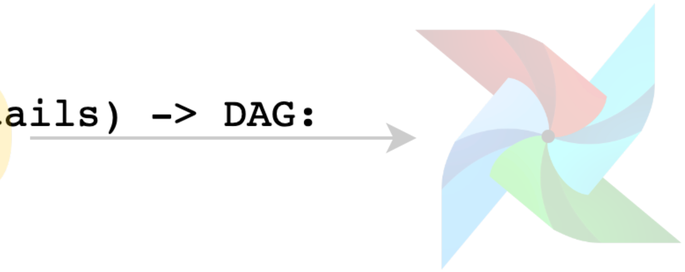
```
for dag_config in get_dag_configs(SCHEMA_PATH, dag_configs):
    dags.append(generate_dag_from_config(dag_config, connection_details))
return dags
```



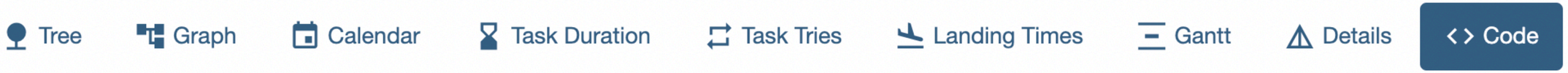


```
def generate_dag_from_config(dag_config, connection_details) -> DAG:
    with DAG(
        dag_id=dag_id,
        schedule_interval=schedule,
        ...
    ) as dag:
        dag.doc_md = generate_doc_md_from_config(dag_id, dag_config.get('models'))

        return generate_dag_branches(dag, main_configs, models, connection_details, process_type)
```





Даг-генератор




```python
from airflow import DAG
from [redacted] import generate_dags_from_configs

dags = generate_dags_from_configs('[redacted]')

for dag in dags:
    globals()[dag.dag_id] = dag
```

Выбор оператора для DBT

```
def run_dbt():
    import subprocess
    subprocess.run(['dbt', 'run'])


run_dbt_operator = PythonOperator(
    task_id='run_dbt',
    python_callable=run_dbt,
    dag=dag
)
```

Выбор оператора для DBT

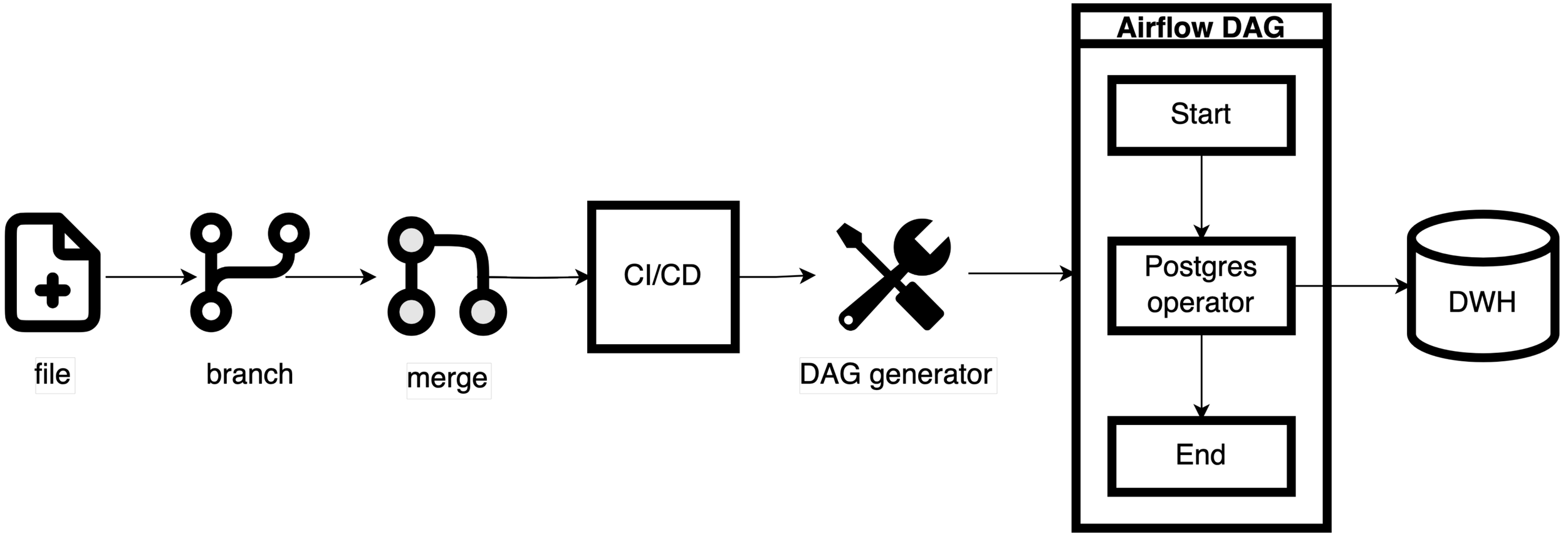
```
return DbtRunOperator(
    task_id=f'dbt_run_{model_name}',
    profiles_dir=DBT_DIR,
    dir=DBT_DIR,
    select=f'{" ".join(temp_models)} {model_name}',
    vars={
        "dag_id": dag_id,
        "process_id": process_id,
        "process_type": process_type,
        "load_dttm": load_dttm,
        "ext_vars": ext_vars,
        **conn_params_dict
    },
    on_failure_callback=callback_error_table(process
)
```

Выбор оператора для DBT

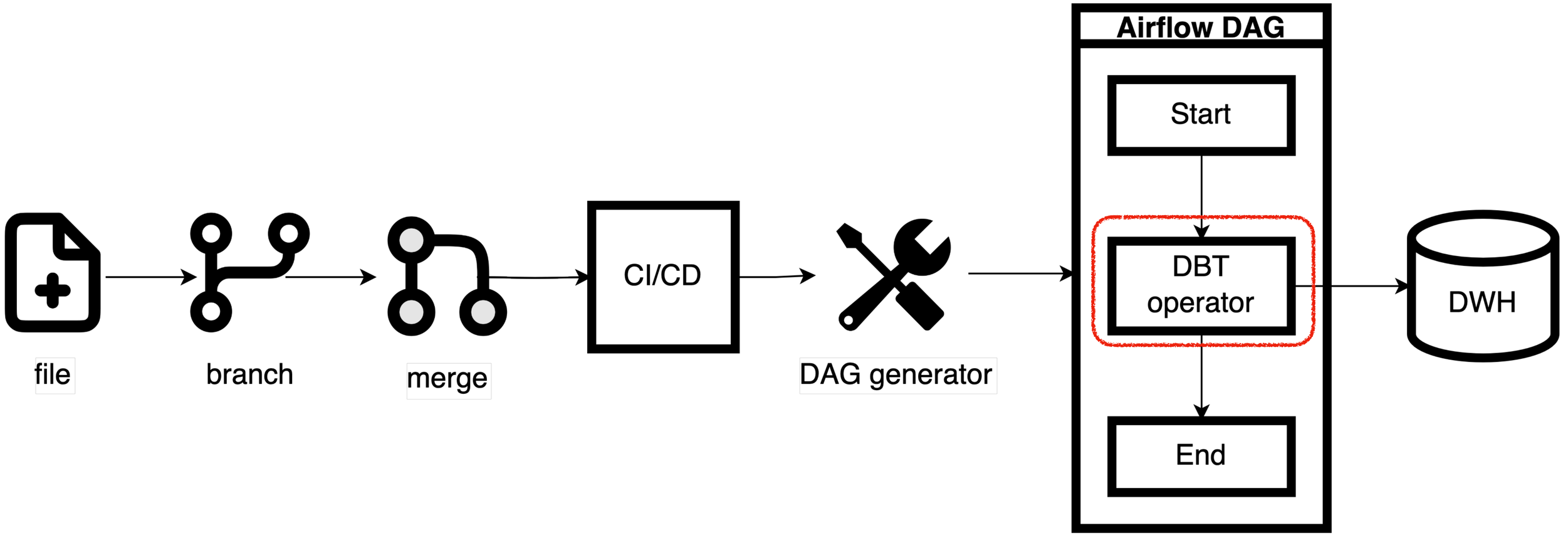
```
return DbtRunOperator(
    task_id=f'dbt_run_{model_name}',
    profiles_dir=DBT_DIR,
    dir=DBT_DIR,
    select=                                    _name}',
    vars={
        "da
        "pr
        "pr
        "lo
        "ext_vars" : ext_vars,
        **conn_params_dict
    },
    on_failure_callback=callback_error_table(process
)
```

```
dbt run
    --profiles-dir [redacted]
    --vars {"dag_id": "[redacted]
    --select [redacted]
```

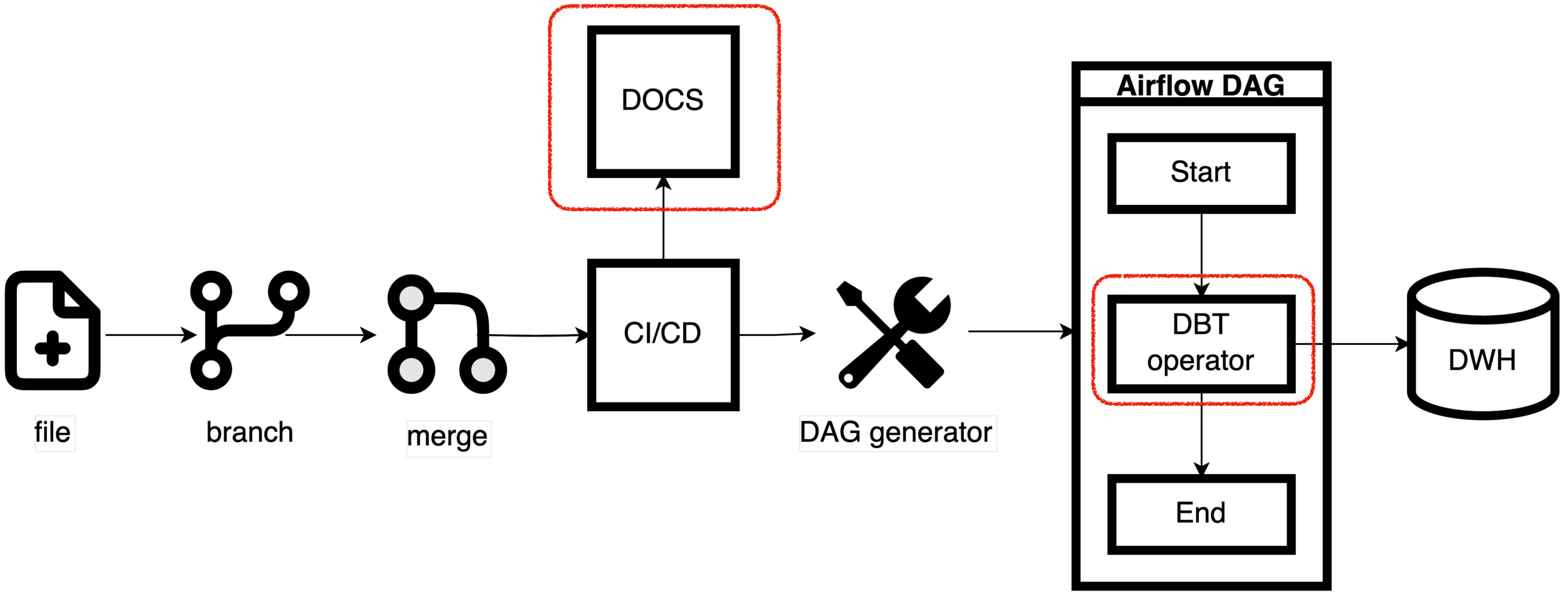
Наше решение до DBT



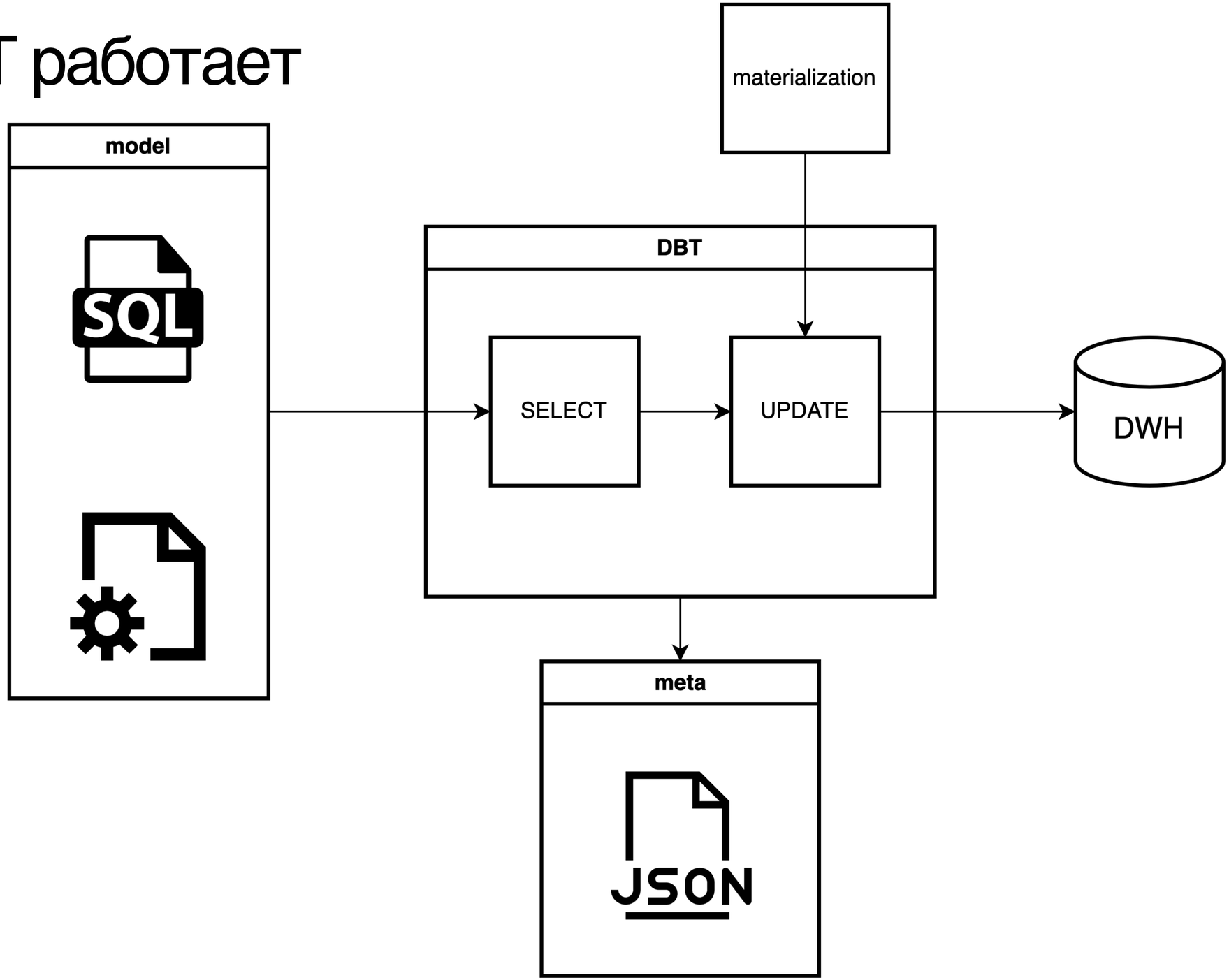
Наше решение с DBT



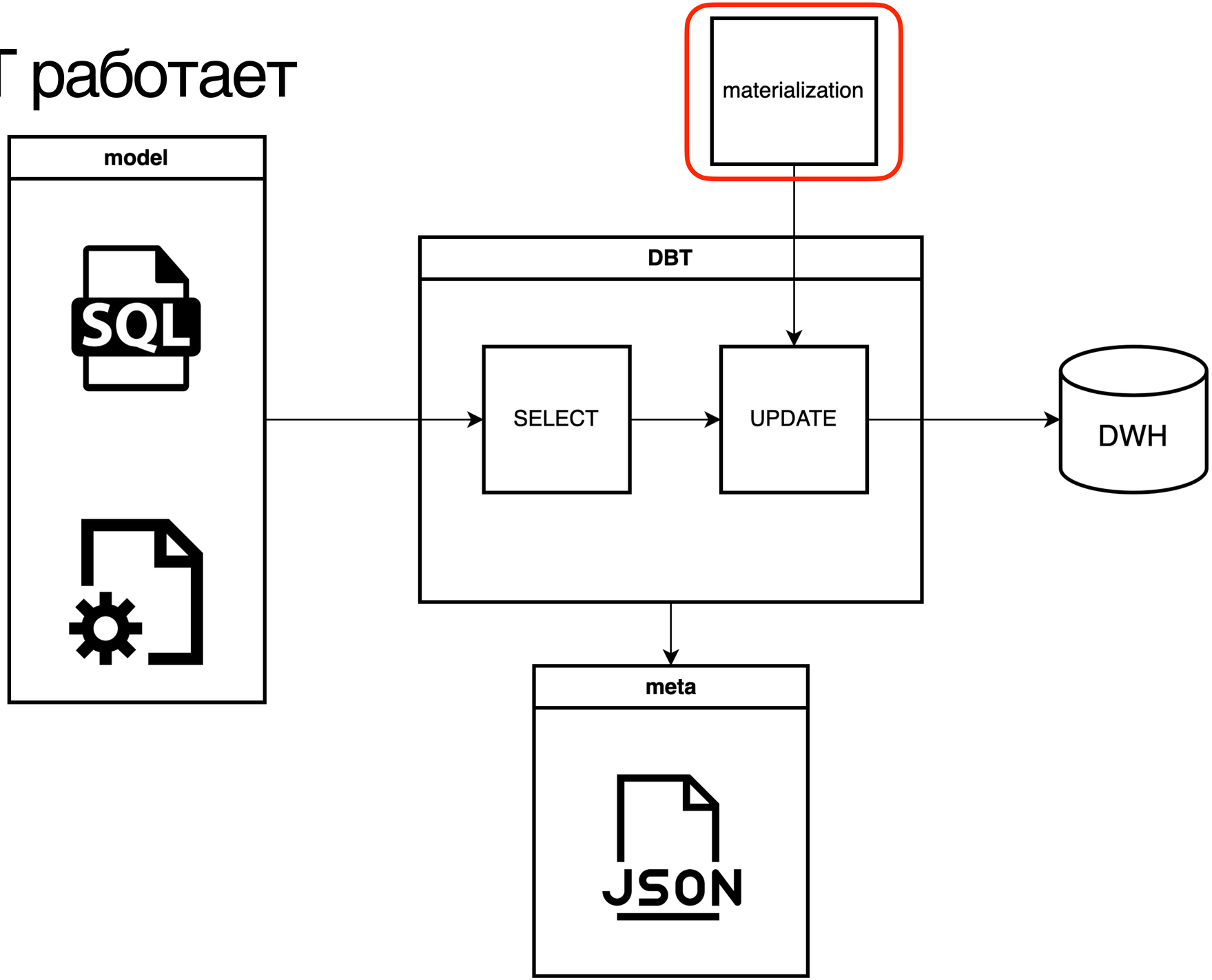
Наше решение с DBT



Как DBT работает



Как DBT работает

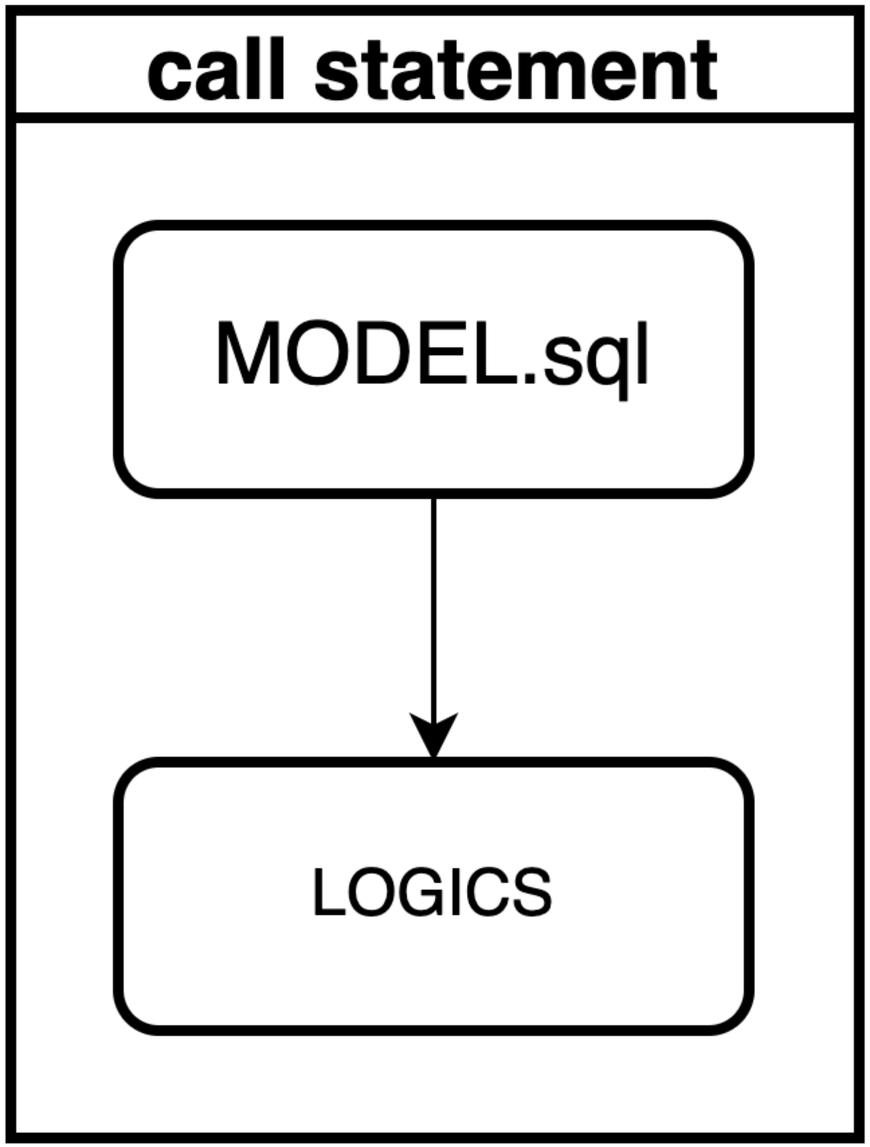


```
{% materialization demo_materialization, default -%}
    {{ run_hooks(pre_hooks, inside_transaction=False) }}

    {% call statement("main") -%}
        insert into {{ target_relation }} (
            {{ columns_name }},
            {{ meta_columns_name }}
        )
        select
            {{ cast_columns }},
            {{ meta_columns }}
        from (sql) as source;
    {%- endcall %}

    {{ run_hooks(post_hooks, inside_transaction=True) }}

    {{ adapter.commit() }}
{%- endmaterialization %}
```

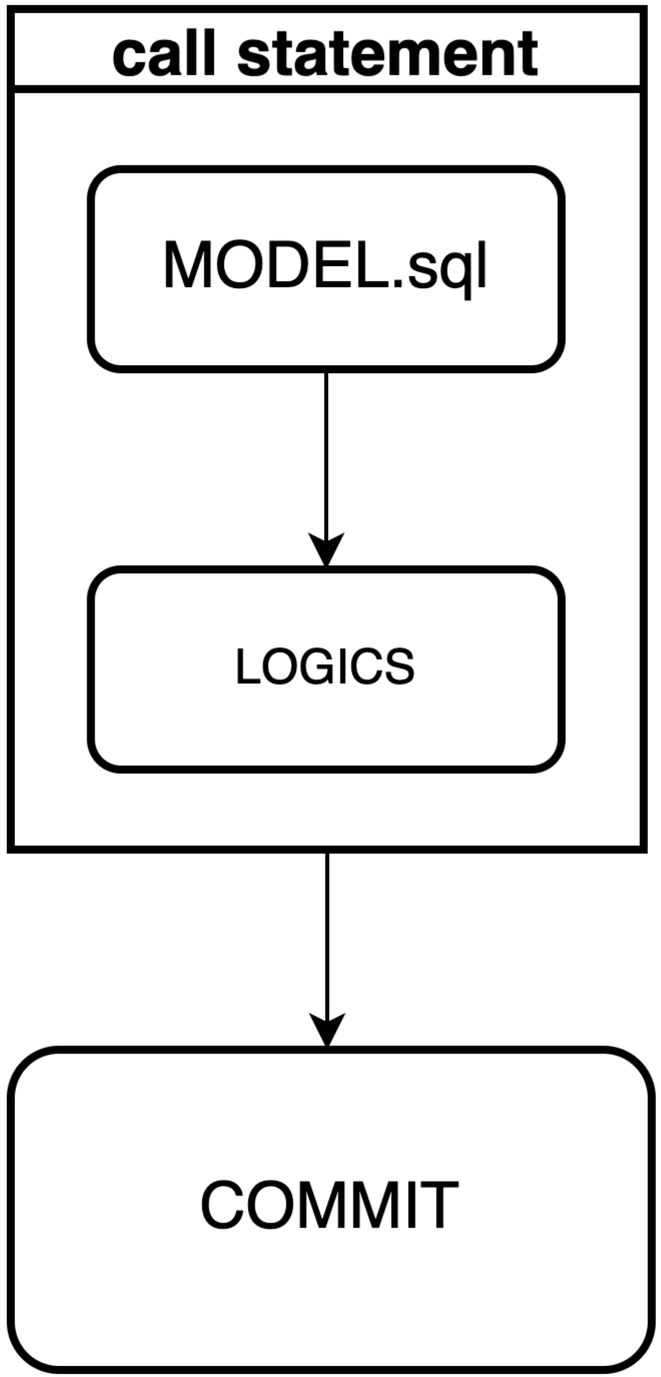
```
{% materialization demo_materialization, default -%}

        from (sql) as source;

{%- endmaterialization %}
```

MODEL.sql

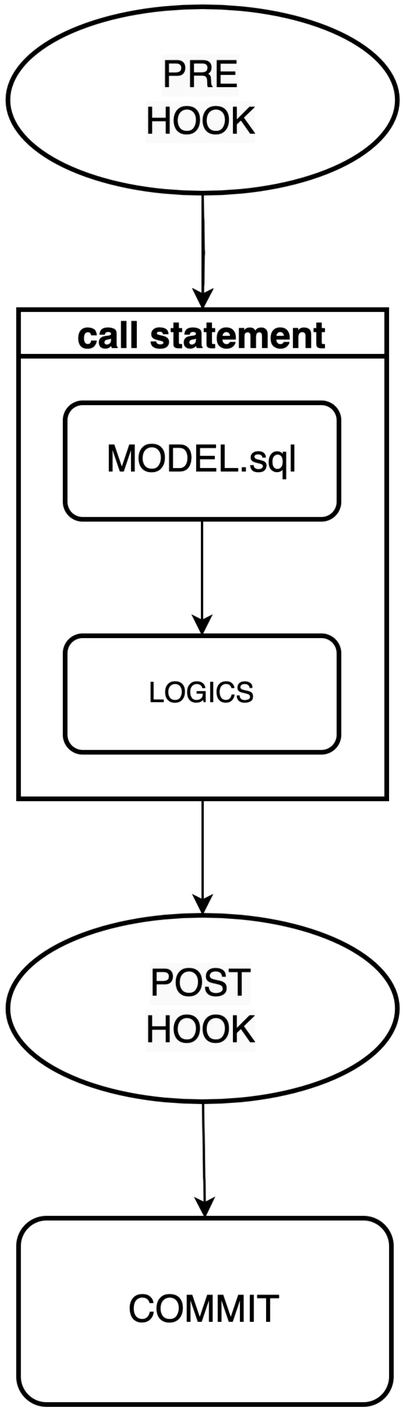
```
{% materialization demo_materialization, default -%}


    {% call statement("main") -%}
        insert into {{ target_relation }} (
            {{ columns_name }},
            {{ meta_columns_name }}
        )
        select
            {{ cast_columns }},
            {{ meta_columns }}
        from (sql) as source;
    {%- endcall %}



{%- endmaterialization %}
```



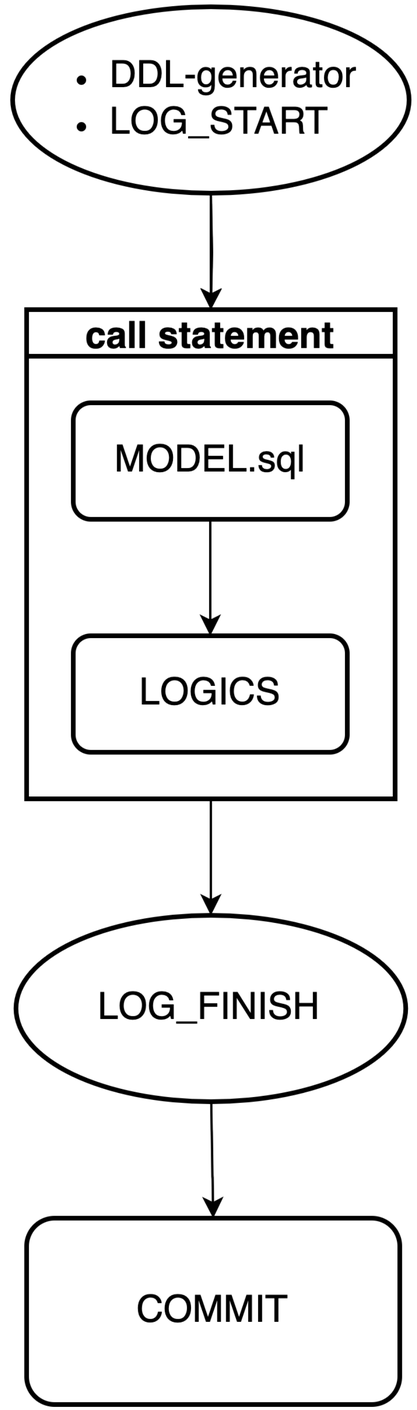
```
{% materialization demo_materialization, default -%}


    {% call statement("main") -%}
        insert into {{ target_relation }} (
            {{ columns_name }},
            {{ meta_columns_name }}
        )
        select
            {{ cast_columns }},
            {{ meta_columns }}
        from (sql) as source;
    {%- endcall %}



    {{ adapter.commit() }}
{%- endmaterialization %}
```



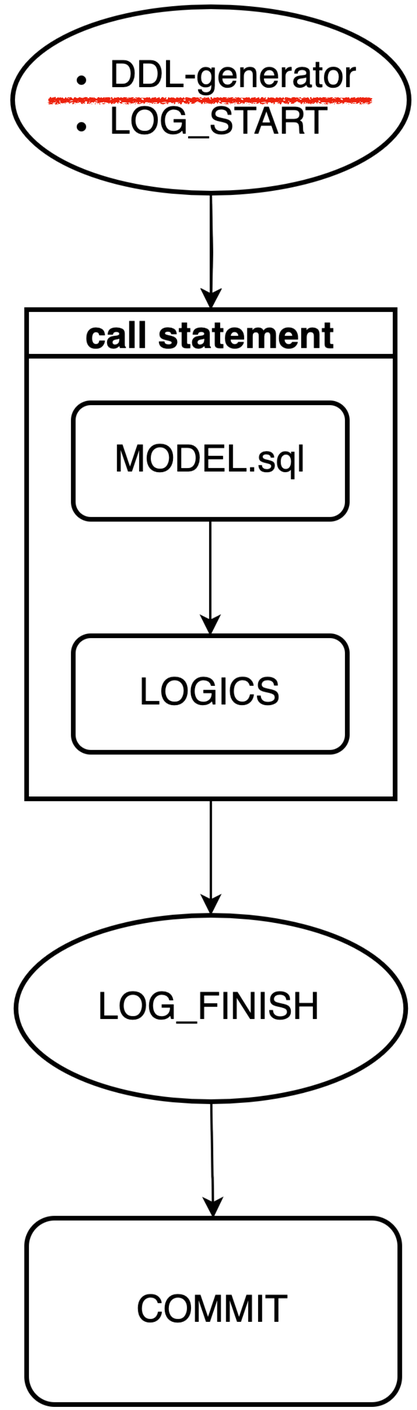
```
{% materialization demo_materialization, default -%}
    {{ run_hooks(pre_hooks, inside_transaction=False) }}

    {% call statement("main") -%}
        insert into {{ target_relation }} (
            {{ columns_name }},
            {{ meta_columns_name }}
        )
        select
            {{ cast_columns }},
            {{ meta_columns }}
        from (sql) as source;
    {%- endcall %}

    {{ run_hooks(post_hooks, inside_transaction=True) }}

    {{ adapter.commit() }}
{%- endmaterialization %}
```



```
{% materialization demo_materialization, default -%}
    {{ run_hooks(pre_hooks, inside_transaction=False) }}

    {% call statement("main") -%}
        insert into {{ target_relation }} (
            {{ columns_name }},
            {{ meta_columns_name }}
        )
        select
            {{ cast_columns }},
            {{ meta_columns }}
        from (sql) as source;
    {%- endcall %}

    {{ run_hooks(post_hooks, inside_transaction=True) }}

    {{ adapter.commit() }}
{%- endmaterialization %}
```



```
{% materialization demo_materialization, default -%}
    {{ run_hooks(pre_hooks, inside_transaction=False) }}

    {% call statement("main") -%}
        insert into {{ target_relation }} (
            {{ columns_name }},
            {{ meta_columns_name }}
        )
        select
            {{ cast_columns }},
            {{ meta_columns }}
        from (sql) as source;
    {%- endcall %}

    {{ run_hooks(post_hooks, inside_transaction=True) }}

    {{ adapter.commit() }}
{%- endmaterialization %}
```



DDL генератор

```
{% macro create_table_ddl(relation) %}
    CREATE TABLE {{ relation }} (
      {%- for col in columns() %}{% if not loop.first %}, {% endif %}
      {{ col }}{% endfor %}
    )
    {{ storage_parameters() }}
    {{ distribution() }}
    {{ partition() }}
    ;

    {%- for comment in comments(relation) %}
    {{ comment }}{% endfor %}

    {{ grant_table(relation) }}
{% endmacro %}
```

DDL генератор

```
{% macro create_table_ddl(relation) %}
    CREATE TABLE {{ relation }} (
      {%- for col in columns() %}{% if not loop.first %}, {% endif %}
      {{ col }}{% endfor %}
    )
    {{ storage_parameters() }}
    {{ distribution() }}
    {{ partition() }}
    ;

    {%- for comment in comments(relation) %}
    {{ comment }}{% endfor %}

    {{ grant_table(relation) }}
{% endmacro %}
```

Промежуточные итоги

ДБТ установили.

Генератор airflow написали. Даги создаются по конфигам.

Материализации написали.

Генератор таблиц.

Логирование.

Бери и пользуйся?

— Мне надоело, у нее были слишком большие запросы.
— Например какие?
— Ну например update instance inner join (select **group.id** as group_id, (select **message.id** from message inner join thread on **thread.id** = message.thread_id where location_id = @location_id and language_id = @language_id and concat(group_key, '.') like concat(group.`key`, '.%') order by message.created desc limit 1) as last_message_id, (select count(*) from thread where location_id = @location_id and language_id = @language_id and concat(group_key, '.') like concat(group.`key`, '.%')) as thread_count, (select if(sum(thread.message_count) is null, 0, sum(thread.message_count)) from thread where location_id = @location_id and language_id = @language_id and concat(group_key, '.') like concat(group.`key`, '.%')) as message_count from group where @channel_key like concat(`key`, '.%')) as statistics on statistics.group_id =

tech

tech

Временные таблицы

```
on-run-end:
  - "{{ drop_temp_tables(results) }}"

models:
  dbt_dags:
    scripts:
      tmp:
        +schema: tmp
        +materialized: temp_table_materialization
```

Временные таблицы

```
models:
  - name: p_shipment
    config:
      temp_models:
        - p_shipment_num_items_cancel_replace_tmp

  - name: p_shipment_num_items_cancel_replace_tmp
    docs:
      show: false
    config:
      meta:
        temporary: true
        distribution: shipment_id
```


Mobil 1
BRAKES ON
POWERED by HONDA
COURTAULDS
1
SHOWA

Зависимости дагов airflow

```
sensor:
  required_tables:
    detail:
      - 'dds.d_api_client'
      - 'dds.d_city'
      - 'dds.d_delivery_category'
      - 'dds.d_delivery_method_kind'
      - 'dds.d_delivery_window_change_reason'
      - 'dds.d_loyalty_transaction_kind'
      - 'dds.d_loyalty_transaction_state'
```

Как долго

- MVP (генератор + одна модель) - 1 мес
- Все модели на DBT - 6 мес
- К текущему виду - 3 мес
- Дальше - ?

Выводы

1. Легко начать
2. Гибко можно настроить под свои алгоритмы
3. Управление возможно по-модельное
4. Экономится время доставки витрин
5. Наглядные графы зависимостей

Спасибо за внимание

tech