

Проект: student_dwh

Выполнил(а): Бурлов Василий Тимофеевич

Группа: БД-251м

Граф зависимостей:



Код модели stg_orders.sql:

```
-- models/staging/stg_orders.sql
-- Эта модель читает данные из исходной таблицы stg.orders,
-- приводит их к нужным типам и исправляет ошибку с почтовым кодом.
-- Все последующие модели будут ссылаться на эту, а не на исходную таблицу.
```

SELECT

```
-- Приводим все к нижнему регистру для консистентности в dbt
"order_id",
("order_date")::date as order_date,
("ship_date")::date as ship_date,
"ship_mode",
"customer_id",
"customer_name",
"segment",
"country",
"city",
"state",
-- Исправляем проблему с Burlington прямо здесь, один раз и навсегда
CASE
  WHEN "city" = 'Burlington' AND "postal_code" IS NULL THEN '05401'
```

```

        ELSE "postal_code"
    END as postal_code,
    "region",
    "product_id",
    "category",
    "subcategory" as sub_category, -- переименовываем для соответствия
    "product_name",
    "sales",
    "quantity",
    "discount",
    "profit"
FROM {{ source('stg', 'orders') }}

```

Код модели sales_fact.sql:

```

-- Создает таблицу фактов, объединяя все измерения
SELECT
    -- Суррогатные ключи из измерений
    cd.cust_id,
    pd.prod_id,
    sd.ship_id,
    gd.geo_id,
    -- Ключи для календаря
    to_char(o.order_date, 'yyyymmdd')::int AS order_date_id,
    to_char(o.ship_date, 'yyyymmdd')::int AS ship_date_id,
    -- Бизнес-ключ и метрики
    o.order_id,
    o.sales,
    o.profit,
    o.quantity,
    o.discount
FROM {{ ref('stg_orders') }} AS o
LEFT JOIN {{ ref('customer_dim') }} AS cd ON o.customer_id = cd.customer_id
LEFT JOIN {{ ref('product_dim') }} AS pd ON o.product_id = pd.product_id
LEFT JOIN {{ ref('shipping_dim') }} AS sd ON o.ship_mode = sd.ship_mode
LEFT JOIN {{ ref('geo_dim') }} AS gd ON o.postal_code = gd.postal_code AND o.city =
gd.city AND o.state = gd.state

```

Код моей индивидуальной mart-модели (mart_quarterly_performance - вариант 5):

```
-- models/marts/mart_quarterly_performance.sql
with orders as (
  select *
  from {{ ref('stg_orders') }}
)

select
  date_trunc('quarter', order_date) as quarter,
  sum(sales) as total_revenue,
  sum(profit) as total_profit
from orders
group by 1
order by 1
```

Файл schema.yml с тестами для всех моделей:

```
# Путь к файлу: models/marts/schema.yml
```

```
version: 2
```

```
models:
```

```
- name: shipping_dim
```

```
  columns:
```

```
    - name: ship_id
```

```
    tests:
```

```
      - unique
```

```
      - not_null
```

```
- name: customer_dim
```

```
  columns:
```

```
    - name: cust_id
```

```
    tests:
```

```
      - unique
```

```
      - not_null
```

```
- name: geo_dim
```

```
  columns:
```

```
    - name: geo_id
```

```
    tests:
```

```
      - unique
```

```
      - not_null
```

```
- name: product_dim
```

```
columns:
  - name: prod_id
    tests:
      - unique
      - not_null

- name: sales_fact
  columns:
    - name: cust_id
      tests:
        - relationships:
            arguments:
              to: ref('customer_dim')
              field: cust_id

- name: mart_quarterly_performance
  description: "Суммарная выручка и прибыль по кварталам"
  columns:
    - name: quarter
      tests:
        - not_null
        - unique
    - name: total_revenue
      tests:
        - not_null
    - name: total_profit
      tests:
        - not_null
```

Скриншот успешного выполнения dbt run и dbt test для проекта student_dwh:

```

(dbt-env) C:\Users\User\Downloads\Магистратура\учеба\Платформы Data Engineering\pde_magistr\student_dwh>dbt run
14:30:37 Running with dbt=1.10.13
14:30:38 Registered adapter: postgres=1.9.1
14:30:40 Found 10 models, 17 data tests, 1 source, 448 macros
14:30:40
14:30:40 Concurrency: 1 threads (target='dev')
14:30:40
14:30:41 1 of 10 START sql table model dw_student.calendar_dim ..... [RUN]
14:30:42 1 of 10 OK created sql table model dw_student.calendar_dim ..... [SELECT 7670 in 1.40s]
14:30:43 2 of 10 START sql table model dw_student.my_first_dbt_model ..... [RUN]
14:30:43 2 of 10 OK created sql table model dw_student.my_first_dbt_model ..... [SELECT 2 in 0.66s]
14:30:43 3 of 10 START sql view model stg.stg_orders ..... [RUN]
14:30:44 3 of 10 OK created sql view model stg.stg_orders ..... [CREATE VIEW in 0.37s]
14:30:44 4 of 10 START sql view model dw_student.my_second_dbt_model ..... [RUN]
14:30:44 4 of 10 OK created sql view model dw_student.my_second_dbt_model ..... [CREATE VIEW in 0.29s]
14:30:44 5 of 10 START sql table model dw_student.customer_dim ..... [RUN]
14:30:44 5 of 10 OK created sql table model dw_student.customer_dim ..... [SELECT 793 in 0.50s]
14:30:44 6 of 10 START sql table model dw_student.geo_dim ..... [RUN]
14:30:45 6 of 10 OK created sql table model dw_student.geo_dim ..... [SELECT 632 in 0.49s]
14:30:45 7 of 10 START sql table model dw_student.mart_quarterly_performance ..... [RUN]
14:30:45 7 of 10 OK created sql table model dw_student.mart_quarterly_performance ..... [SELECT 16 in 0.41s]
14:30:45 8 of 10 START sql table model dw_student.product_dim ..... [RUN]
14:30:46 8 of 10 OK created sql table model dw_student.product_dim ..... [SELECT 4344 in 0.35s]
14:30:46 9 of 10 START sql table model dw_student.shipping_dim ..... [RUN]
14:30:46 9 of 10 OK created sql table model dw_student.shipping_dim ..... [SELECT 4 in 0.29s]
14:30:46 10 of 10 START sql table model dw_student.sales_fact ..... [RUN]
14:30:47 10 of 10 OK created sql table model dw_student.sales_fact ..... [SELECT 25667 in 0.68s]
14:30:47
14:30:47 Finished running 8 table models, 2 view models in 0 hours 0 minutes and 6.71 seconds (6.71s).
14:30:47
14:30:47 Completed successfully
14:30:47
14:30:47 Done. PASS=10 WARN=0 ERROR=0 SKIP=0 NO-OP=0 TOTAL=10

```

```

(dbt-env) C:\Users\User\Downloads\Магистратура\учеба\Платформы Data Engineering\pde_magistr\student_dwh>dbt test
14:33:52 Running with dbt=1.10.13
14:33:52 Registered adapter: postgres=1.9.1
14:33:53 Found 10 models, 17 data tests, 1 source, 448 macros
14:33:53
14:33:53 Concurrency: 1 threads (target='dev')
14:33:53
14:33:54 1 of 17 START test not_null_customer_dim_cust_id ..... [RUN]
14:33:54 1 of 17 PASS not_null_customer_dim_cust_id ..... [PASS in 0.23s]
14:33:54 2 of 17 START test not_null_geo_dim_geo_id ..... [RUN]
14:33:54 2 of 17 PASS not_null_geo_dim_geo_id ..... [PASS in 0.12s]
14:33:54 3 of 17 START test not_null_mart_quarterly_performance_quarter ..... [RUN]
14:33:54 3 of 17 PASS not_null_mart_quarterly_performance_quarter ..... [PASS in 0.23s]
14:33:54 4 of 17 START test not_null_mart_quarterly_performance_total_profit ..... [RUN]
14:33:55 4 of 17 PASS not_null_mart_quarterly_performance_total_profit ..... [PASS in 0.12s]
14:33:55 5 of 17 START test not_null_mart_quarterly_performance_total_revenue ..... [RUN]
14:33:55 5 of 17 PASS not_null_mart_quarterly_performance_total_revenue ..... [PASS in 0.13s]
14:33:55 6 of 17 START test not_null_my_first_dbt_model_id ..... [RUN]
14:33:55 6 of 17 FAIL 1 not_null_my_first_dbt_model_id ..... [FAIL 1 in 0.11s]
14:33:55 7 of 17 START test not_null_my_second_dbt_model_id ..... [RUN]
14:33:55 7 of 17 PASS not_null_my_second_dbt_model_id ..... [PASS in 0.11s]
14:33:55 8 of 17 START test not_null_product_dim_prod_id ..... [RUN]
14:33:55 8 of 17 PASS not_null_product_dim_prod_id ..... [PASS in 0.11s]
14:33:55 9 of 17 START test not_null_shipping_dim_ship_id ..... [RUN]
14:33:55 9 of 17 PASS not_null_shipping_dim_ship_id ..... [PASS in 0.14s]
14:33:55 10 of 17 START test relationships_sales_fact_cust_id_cust_id_ref_customer_dim ..... [RUN]
14:33:55 10 of 17 PASS relationships_sales_fact_cust_id_cust_id_ref_customer_dim ..... [PASS in 0.15s]
14:33:55 11 of 17 START test unique_customer_dim_cust_id ..... [RUN]
14:33:56 11 of 17 PASS unique_customer_dim_cust_id ..... [PASS in 0.14s]
14:33:56 12 of 17 START test unique_geo_dim_geo_id ..... [RUN]
14:33:56 12 of 17 PASS unique_geo_dim_geo_id ..... [PASS in 0.12s]
14:33:56 13 of 17 START test unique_mart_quarterly_performance_quarter ..... [RUN]
14:33:56 13 of 17 PASS unique_mart_quarterly_performance_quarter ..... [PASS in 0.11s]
14:33:56 14 of 17 START test unique_my_first_dbt_model_id ..... [RUN]
14:33:56 14 of 17 PASS unique_my_first_dbt_model_id ..... [PASS in 0.11s]
14:33:56 15 of 17 START test unique_my_second_dbt_model_id ..... [RUN]
14:33:56 15 of 17 PASS unique_my_second_dbt_model_id ..... [PASS in 0.16s]
14:33:56 16 of 17 START test unique_product_dim_prod_id ..... [RUN]
14:33:56 16 of 17 PASS unique_product_dim_prod_id ..... [PASS in 0.17s]
14:33:56 17 of 17 START test unique_shipping_dim_ship_id ..... [RUN]
14:33:56 17 of 17 PASS unique_shipping_dim_ship_id ..... [PASS in 0.12s]
14:33:56
14:33:56 Finished running 17 data tests in 0 hours 0 minutes and 3.08 seconds (3.08s).
14:33:57
14:33:57 Completed with 1 error, 0 partial successes, and 0 warnings:
14:33:57
14:33:57 Failure in test not_null_my_first_dbt_model_id (models\example\schema.yml)
14:33:57 Got 1 result, configured to fail if != 0
14:33:57
14:33:57 compiled code at target\compiled\student_dwh\models\example\schema.yml\not_null_my_first_dbt_model_id.sql
14:33:57
14:33:57 Done. PASS=16 WARN=0 ERROR=1 SKIP=0 NO-OP=0 TOTAL=17

```

Скриншот с данными из моей индивидуальной mart-модели (DBEaver):

The screenshot shows the DBEaver 25.2.2 interface. The left sidebar displays a project tree with a database named 'superstore' and a schema 'dw_student'. The main window shows a SQL query: `SELECT * FROM dw_student.mart_quarterly_performance;`. The results are displayed in a table with columns: 'quarter', 'total_revenue', and 'total_profit'. The table contains 12 rows of data, representing quarterly performance from 2016 to 2018. An 'Error Log' window at the bottom shows several error messages related to preference category paths.

quarter	total_revenue	total_profit
2016-01-01 00:00:00.000 +0300	74 447,796	3 811,229
2016-04-01 00:00:00.000 +0300	86 538,7596	11 204,0692
2016-07-01 00:00:00.000 +0300	143 633,2123	12 804,7218
2016-10-01 00:00:00.000 +0300	179 627,7302	21 723,9541
2017-01-01 00:00:00.000 +0300	68 851,7386	9 264,9416
2017-04-01 00:00:00.000 +0300	89 124,187	12 190,8234
2017-07-01 00:00:00.000 +0300	130 259,5752	16 853,6194
2017-10-01 00:00:00.000 +0300	182 297,0082	23 309,1203
2018-01-01 00:00:00.000 +0300	93 237,181	11 441,3708
2018-04-01 00:00:00.000 +0300	136 082,301	16 390,3394
2018-07-01 00:00:00.000 +0300	143 787,3622	15 823,6048
2018-10-01 00:00:00.000 +0300	236 098,7538	38 139,8593

Выводы:

Использование dbt для реализации DWH имеет несколько ключевых преимуществ по сравнению с ручным написанием DDL/DML скриптов:

Структурированность и повторяемость: dbt позволяет организовать проект в виде моделей, источников и макросов, что упрощает поддержку и масштабирование.

Автоматизация тестирования данных: встроенные тесты (unique, not_null, relationships) помогают проверять качество данных на каждом этапе, уменьшая риск ошибок.

Документирование и lineage: dbt автоматически генерирует документацию и граф зависимостей моделей, что облегчает понимание потока данных и коммуникацию с командой.

Модульность: модели можно легко комбинировать для разных аналитических задач, без дублирования SQL-кода.

Ускорение разработки: разработка через dbt снижает количество ручной работы, ускоряет развертывание новых моделей и улучшает контроль версий с помощью Git.

В целом, dbt делает процесс построения DWH более прозрачным, управляемым и надежным по сравнению с ручным созданием таблиц и написанием SQL-скриптов.