



Есть ли жизнь после dbt?

Попова Александра
Positive Technologies



О спикере

- 10+ in data
- 5 with dbt
- Ex: fin-tech, retail-tech, med-tech
- Now: infosec-tech

План



- Почему dbt захватил мир?
- Где есть нюансы?
- Альтернативы?
- Стоит ли мигрировать с dbt?
- Как выбрать инструмент под задачи?

Data Build Tool



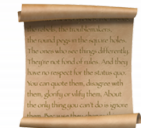
2016



13.7k



440



Apache 2.0

Почему DBT?



- Transformation-as-Code
- DAGs
- Tests
- Docs
- Python adapters
- dbt-packages, ecosys etc



Что делает DBT?

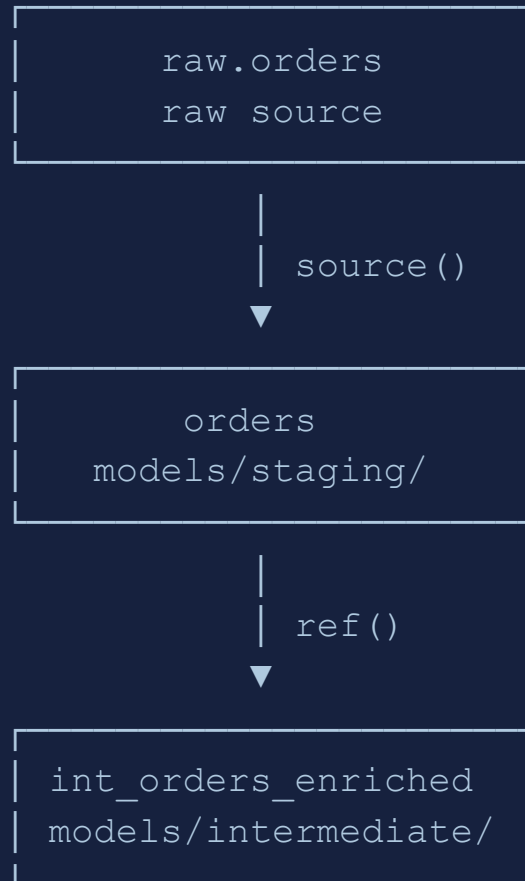


```
{{
config(
materialized= 'incremental',
incremental_strategy='delete+insert',
unique_key=['order_id'],
partition_by='toYYYYMM(order_dt)'
)
}}

SELECT
order_id, order_dt, status,
{{ revenue_expr('amount') }} AS revenue
FROM {{ ref('orders') }}
{% if is_incremental() %}
WHERE order_dt >= {{ get_max_cursor(...) }}
{% endif %}
```

-> RENDER

Что делает DBT?



-> DAG

-> EXECUTION

DBT артефакты



JSON	Что содержит
manifest.json	Полный DAG: модели, тесты, сиды, источники, зависимости, конфиги, скомпилированный
catalog.json	Метаданные схемы из самого СУБД (колонки, типы)
run_results.json	Результаты последнего запуска/теста: статус, время выполнения, ошибки
sources.json	Результаты freshness-проверок источников



Но где нюанс?

DBT-проект. 2021.



```
1 lcc_horns_and_hooves/
2 |— dbt_project.yml
3 |— packages.yml          # dbt_utils
4 |— models/               [247]
5 |   |— stg/              [96]
6 |       |— stg_orders.sql
7 |       |— stg_customers.sql
8 |       |— stg_payments.sql    # ... +93
9 |   |— dds/              [38]
10 |   |— marts/            [113]
11 |       |— finance/      [41]
12 |       |— marketing/    [72]
13 |   |— macros/          [4]
14 |       |— generate_schema_name.sql
15 |       |— cents_to_dollars.sql
16 |       |— date_spine.sql
17 |       |— limit_data_in_dev.sql
18 |   |— tests/            [3]
19 |       |— assert_positive_revenue.sql
20 |       |— assert_unique_order_id.sql
21 |       |— assert_no_future_dates.sql
22 |   |— seeds/            [2]
23 |       |— country_codes.csv
24 |       |— currency_rates.csv
25 |   |— snapshots/        [1]
26 |       |— customers_snapshot.sql
```

DBT-проект. 2026.



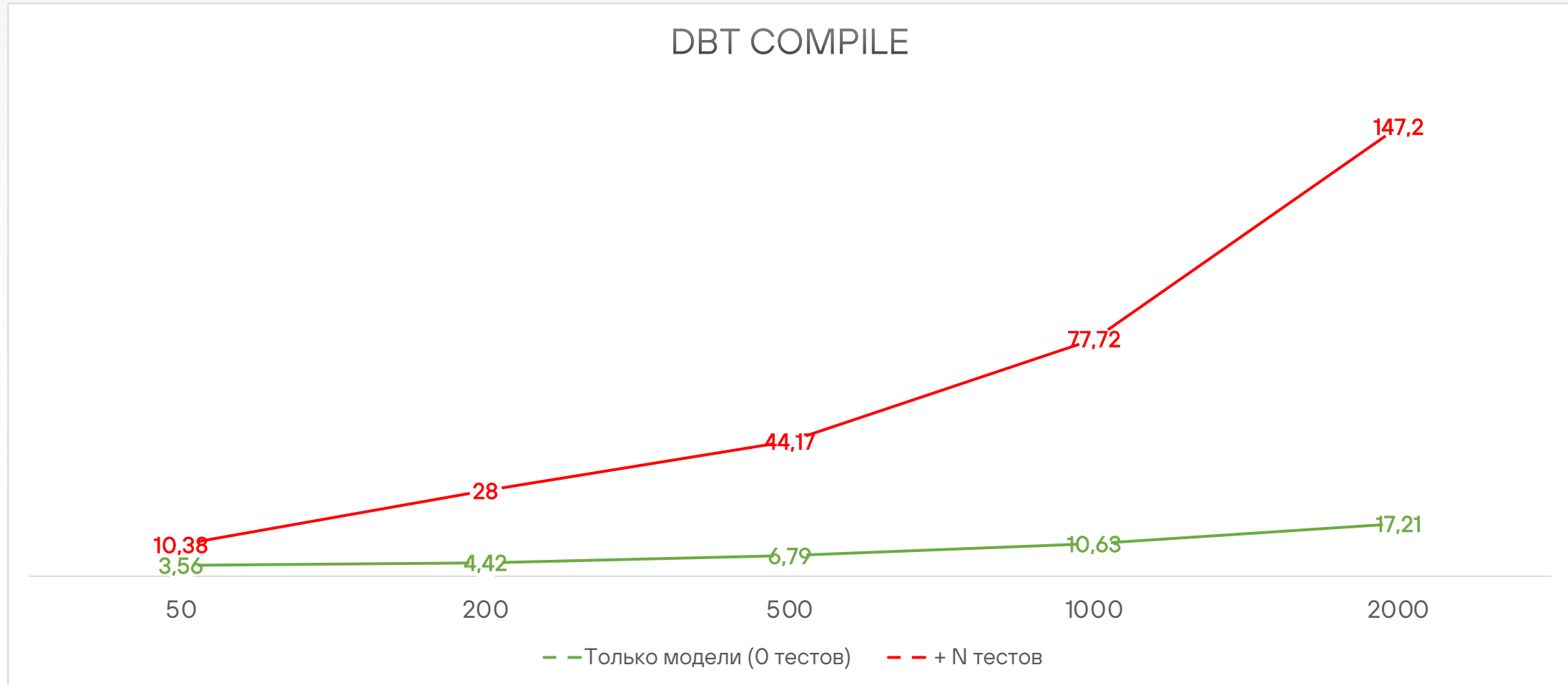
```
1  ch_lcc_horns_and_hooves/
2  ├── dbt_project.yml
3  ├── packages.yml           # dbt_utils, codegen, dbt_expectations, elementary ... +8
4  ├── dbt_packages/         # ~3 000 файлов
5  ├── models/                [6 656]
6  │   ├── stg/               [800]
7  │   │   └── ...             # 40+ источников
8  │   ├── dds/               [800]
9  │   ├── marts/             [2 340]
10 │   │   ├── finance/       [410]
11 │   │   ├── marketing/     [380]
12 │   │   ├── it/            [512]
13 │   │   ├── sales/         [340]
14 │   │   ├── ops/           [298]
15 │   │   ├── legacy_do_not_touch/ [400] # нельзя трогать!
16 │   │   └── ...             # ещё 8 доменов
17 │   └── deprecated_v1/     [376] # "снести после Q3 2027"
18 ├── macros/                [187]
19 │   ├── materializations/   [6]
20 │   │   ├── custom_incremental.sql
21 │   │   └── custom_snapshot_scd2.sql
22 │   ├── finance/            [24]
23 │   ├── marketing/          [18]
24 │   ├── get_fiscal_period.sql # автора нет 1,5 года
25 │   └── ...                 # ещё 130 макросов
26 ├── tests/                 [1 340]
27 │   ├── generic/            [420]
28 │   └── singular/           [920]
29 ├── seeds/                 [64]
30 ├── snapshots/             [58]
31 └── analyses/               [22]
```

DBT-проект. 2026.



```
1 trino_lcc_horns_and_hooves/
2 |— dbt_project.yml
3 |— packages.yml           # dbt_utils, codegen, dbt_expectations, elementary ... +8
4 |— dbt_packages/         # ~3 000 файлов
5 |— models/               [6 656]
6 |   |— stg/              [800]
7 |   |   |— ...          # 40+ источников
8 |   |— dds/              [800]
9 |   |— marts/            [2 340]
10 |       |— finance/     [410]
11 |       |— marketing/   [380]
12 |       |— it/          [512]
13 |       |— sales/       [340]
14 |       |— ops/         [298]
15 |       |— ...          # ещё 8 доменов
16 |       |— deprecated_v1/ [376] # "снести после Q3 2027"
17 |— macros/              [187]
18 |   |— materializations/ [6]
19 |       |— custom_incremental.sql
20 |       |— custom_snapshot_scd2.sql
21 |   |— finance/         [24]
22 |   |— marketing/       [18]
23 |   |— ...              # ещё 130 макросов
24 |— tests/               [1 340]
25 |   |— generic/         [420]
26 |   |— singular/       [920]
27 |— seeds/               [64]
28 |— snapshots/          [58]
29 |— analyses/            [22]
30
```

DBT-проект. 2026.



```

-- macros/current_timestamp.sql
{% macro current_timestamp() %}
    {{ return(adapter.dispatch('current_timestamp')()) }}
{% endmacro %}

{% macro default__current_timestamp() %}
    now()
{% endmacro %}

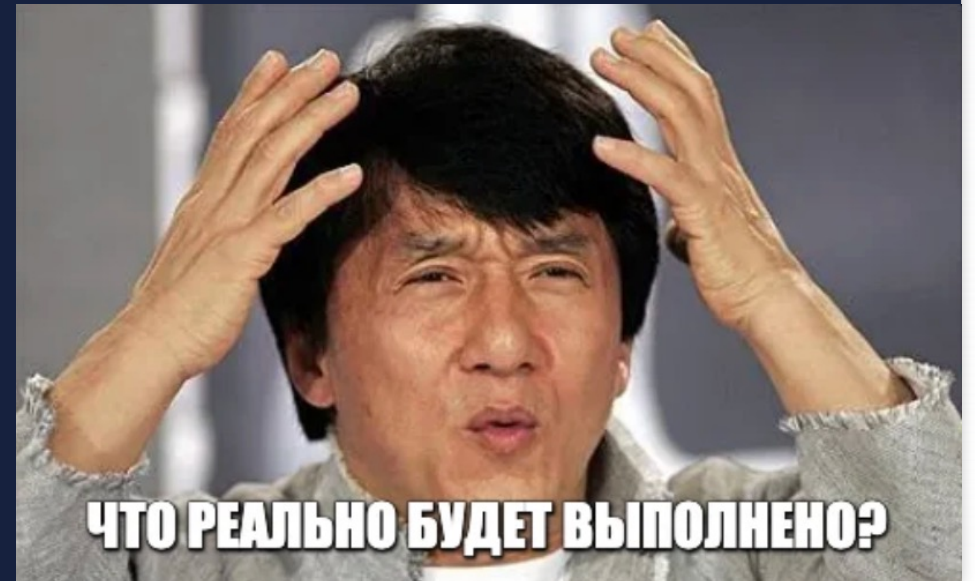
{% macro snowflake__current_timestamp() %}
    current_timestamp()
{% endmacro %}

{% macro bigquery__current_timestamp() %}
    CURRENT_TIMESTAMP()
{% endmacro %}
```



```
{{
config(
materialized= 'incremental',
incremental_strategy='delete+insert',
unique_key=['order_id'],
partition_by='toYYYYMM(order_dt)'
)
}}

SELECT
order_id, order_dt, status,
{{ revenue_expr('amount') }} AS revenue,
{{ current_timestamp() }}
FROM {{ ref('orders') }}
{% if is_incremental() %}
WHERE order_dt >= {{ get_max_cursor(...) }}
{% endif %}
```



Text ≠ semantics



```
SELECT *  
FROM {{ ref('orders') }}  
{% if is_incremental() %}  
WHERE updated_dt >= {{ get_max_cursor(updated_dt, this) }}  
{% endif %}
```

Text ≠ semantics



```
-- full-refresh
```

```
SELECT * FROM raw.orders
```

Text ≠ semantics



```
-- incremental
```

```
SELECT *
```

```
    FROM raw.orders
```

```
    WHERE updated_at >= (SELECT MAX(updated_at)
```

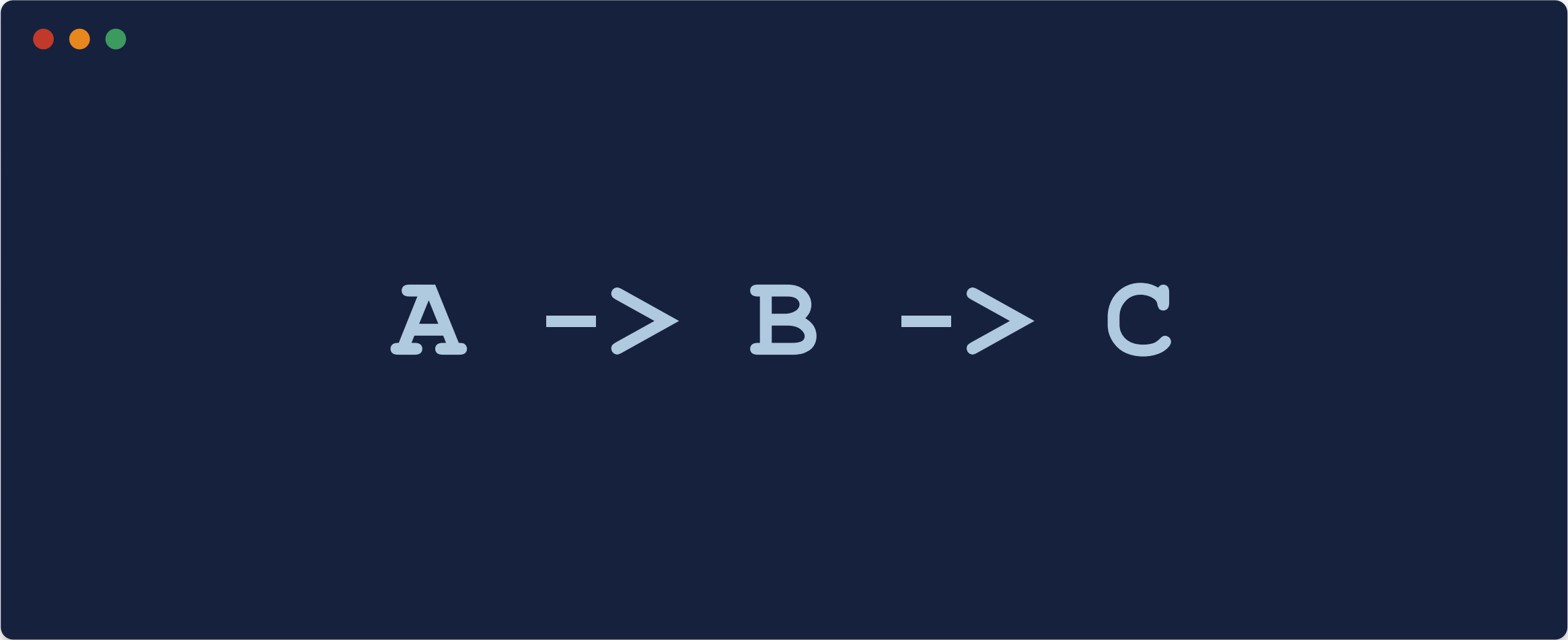
```
FROM stg.orders)
```

Text ≠ semantics



➤ Строковое представление модели одинаковое, но исполняемый SQL разный.

DAG ≠ impact graph

A dark blue terminal window with three colored window control buttons (red, orange, green) in the top-left corner. The text 'A -> B -> C' is displayed in the center in a light blue monospace font.

A -> B -> C

DAG ≠ impact graph



A.customer_id



B.customer_id



C.customer_id

A.amount



C.revenue

DAG $\neq$ impact graph



- Нужно ли перезапускать модель C?
- Нужен backfill?
- Нужен полный rebuild?

max(updated_at)



```
--- stg_orders.sql
{% if is_incremental() %}
WHERE updated_at > (
    SELECT MAX(updated_at)
    FROM {{ this }}
)
{% endif %}
```

max(updated_at)



```
--- fct_daily_revenue.sql
{% if is_incremental() %}
WHERE order_dt > (
    SELECT MAX(order_dt)
    FROM {{ this }}
)
{% endif %}
```

max(updated_at)



```
|-----|-----|-----|--...--|
| order_id | order_dt | updated_at |--...--|
|-----|-----|-----|--...--|
| 12345678 | 2026-09-12 | 2026-09-21 |--...--|
|-----|-----|-----|--...--|
```

max(updated_at) ≠ processed intervals



09-17 ✓ stg_orders - ok, fct_daily_revenue - ok

09-18 ✓ stg_orders - ok, fct_daily_revenue - ok

09-19 X - выходной, шедюлер отдыхал - **missing**

09-20 X - выходной, шедюлер отдыхал - **missing**

09-21 ✓ stg_orders - ok, fct_daily_revenue - not ok

Стоимость dev env



PROD

|— schema A - 2TB

| |— tab_a

| |— tab_b

|— schema B - 4TB

| |— tab_c

| |— tab_d

|— schema C - 1TB

| |— tab_e

Стоимость dev env



Вариант 1 – Общий TEST / DEV

|— schema A

| |— tab_a

| |— tab_b

|— schema B

| |— tab_c

| |— tab_d

|— schema C

| |— tab_e

Стоимость dev env



Вариант 1 – Общий TEST / DEV, **НО**

|— schema A

| |— tab_a <-- сейчас работает Маша, но **CI job #4821**

| |— tab_b

|— schema B

| |— tab_c <-- с утра перетёр Саша --full-refresh

| |— tab_d

|— schema C

| |— tab_e

Стоимость dev env



Вариант 2 – у каждого свой dev

```
|— dev_аророва
|   |— schema_a_tab_a
|   |— schema_a_tab_b
|   |— schema_b_tab_c
|— dev_еропов
|   |— schema_a_tab_a
|   |— schema_b_tab_c
|   |— schema_b_tab_d
```

Стоимость dev env



Вариант 2 – У каждого свой dev

```
{% if target.name == 'dev' %}
```

```
WHERE updated_at >= current_date - 7
```

```
-- или LIMIT 1000
```

```
{% endif %}
```

+

```
{% macro generate_schema_name(custom_schema_name, node) -%}
```

```
    {{ generate_schema_name_for_env(custom_schema_name, node) }}
```

```
{%- endmacro %}
```

Стоимость dev env



время создания одной dev-схемы
= полный физический пересчёт,
каждый раз



Откликается?





**А есть ли
альтернативы?**

2026: карта рынка



rendered SQL

независимый

dbt-core v1

2016

текстовый · независимый

SQL-comprehension

SQLMesh

2023

comprehension · независимый*

Coalesce

2020

текстовый · платформенный

Dataform

2018

текстовый · платформенный

Snowflake DT

2024

comprehension · платформенный

Bruin

2024

текстовый · платформенный

Matillion

2015

текстовый · платформенный

Databricks LF

2025

comprehension · платформенный

платформенный / vendor-managed control plane

*gossip time



03.09.2025



Fivetran покупает
Tobiko Data
(SQLMesh)

13.10.2025



Fivetran объявляет
слияние с dbt Labs

25.03.2026



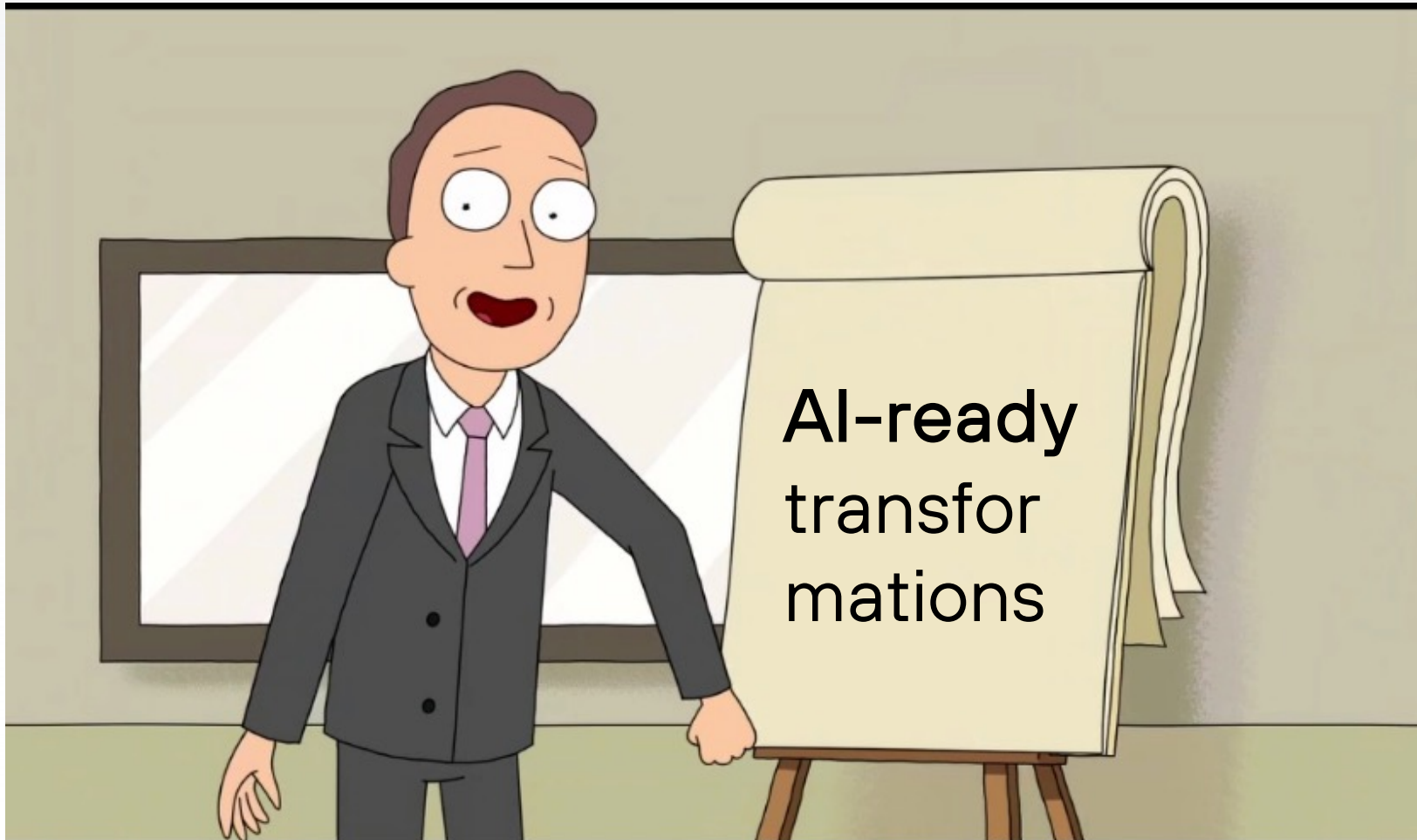
SQLMesh передан
в Linux Foundation

01.06.2026



Слияние Fivetran +
dbt Labs закрыто

2026: тренд рынка



2026: тренд рынка



AI-ready transformations:

- SQL
- Semantic Layer
- Contracts
- Lineage
- State
- Safe execution



**Попробуем
сравнить**

Критерии сравнения - 1

Архитектура



- Тип трансформации
- Open source / лицензия
- State / инкрементальность
- Multi-engine
- Тестирование / data quality
- CI/CD и git-интеграция
- Control plane / хостинг

Критерии сравнения - 2

Стоимость, экосистема



- Порог входа / Скорость обучения
- Стоимость
- Семантический слой
- Экосистема / Коммьюнити
- Python-модели
- Dev-окружения
- AI-ready

Сравнение инструментов (1/2)



Критерий	dbt v1	SQLMesh	Coalesce	Dataform	Bruin	Matillion	Snowflake DT	Databricks LF
Тип трансформации	Текст. SQL (Jinja)	SQL comprehension (AST)	Metadata-driven	Текст. SQL (SQLX)	SQL+Python (CLI)	Visual+SQL+Python	Декларативный SQL	Декл. SQL/Python
Open source / лицензия	Apache 2.0	Apache 2.0	Нет, free≠OSS	Компилятор OSS/GCP	OSS core+cloud	Нет, enterprise	Нет, платформа	Нет, платформа
Control plane / хостинг	Self-hosted	Self-hosted	Vendor-managed	Managed GCP	Self-hosted/cloud	Managed/self-host	Внутри Snowflake	Внутри Databricks
Оркестрация	✗	✗	✓	✓	✓	✓	✓	✓

Сравнение инструментов (1/2)



Критерий	dbt v1	SQLMesh	Coalesce	Dataform	Bruin	Matillion	Snowflake DT	Databricks LF
State / инкрементальность	state:modified есть	Нативно	Через metadata	State – when(incremental())	Через DQ-проверки	Платформенная логика	Авто платформой	Авто платформой
Multi-engine	Да, десятки адаптеров	Да (SQLGlot)	Нет	Нет, только BigQuery	Да	Да	только Snowflake	только Databricks
Тестирование / DQ	Generic+custom тесты	Unit tests+audits	Node-level checks	Assertions	Встроено в пайплайн	Validation-компоненты	Только constraints	Expectations
CI/CD и git-интеграция	Git, +- diff	Git, явный plan-diff	Git, diff	Git+release configs	Git, без diff	Git, визуальный	Нет своего	Repos/Asset Bundles

Сравнение инструментов (2/2)



Критерий	dbt v1	SQLMesh	Coalesce	Dataform	Bruin	Matillion	Snowflake DT	Databricks LF
Порог входа / Скорость обучения	Средняя	Средняя	Низкая (low-code)	Средняя (похож на dbt)	Средняя	Низкая (no-code)	Низкая	Низкая
Стоимость / модель цены	Бесплатно+о пц. cloud	Бесплатно+о пц. cloud	Seat/consumption	Бесплатно, если уже в BigQuery	Бесплатно+о пц. cloud	Seat+consumption	Часть Snowflake	Часть Databricks
Семантический слой	✓ в cloud	✗	✓	✗	✓	✗	✓	✓
Экосистема/Коммьюнити	Огромная (dbt Hub)	Молодая, растёт	Закрытая	Небольшая	Молодая, растёт	Закрытая	Закрытая	Растёт с маркетплейсом

Сравнение инструментов (2/2)



Критерий	dbt v1	SQLMesh	Coalesce	Dataform	Bruin	Matillion	Snowflake DT	Databricks LF
Python-модели	Да, но только на Snowflake/BigQuery/Databricks	✓	Ограниченно	Нет	✓	✓	Нет	✓
Zero-copy / dev-окружения	✗	✓	✗	✗	✗	✗	Платформенный clone	Unity Catalog clone
AI-ready / LLM-контекст	Структур. метаданные; MCP ✓	Структур. метаданные; sqlmesh-mcp ⚠	AI-генерация nodes MCP ✓	Интеграция с Gemini MCP ✓	Agent-friendly CLI MCP ✓	Designer AI встроен (enterprise) MCP ✓	AI — через Cortex MCP ✓	Интеграция с Mosaic AI MCP ✓



**Кто главный
конкурент?**

2026: карта рынка



rendered SQL

независимый

dbt-core v1

2016

текстовый · независимый

SQL-comprehension

SQLMesh

2023

comprehension · независимый*

dbt v2

2026

comprehension · независимый

Coalesce

2020

текстовый · платформенный

Dataform

2018

текстовый · платформенный

Snowflake DT

2024

comprehension · платформенный

Bruin

2024

текстовый · платформенный

Matillion

2015

текстовый · платформенный

Databricks LF

2025

comprehension · платформенный

платформенный / vendor-managed control plane



dbt v2 🎉

Rust + DataFusion = 🥰

dbt v2



14.01.2025



dbt Labs приобретает
SDF Labs

28.05.2025



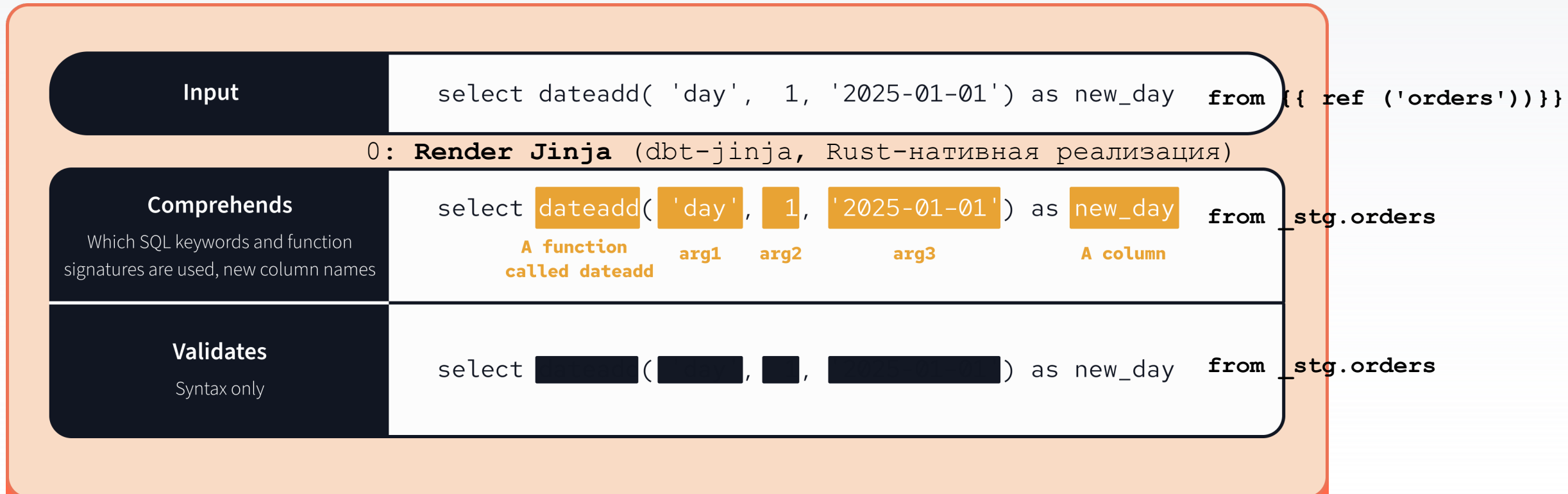
dbt Fusion Engine beta


16.09.2026



dbt v2 GA 

Как dbt v2 видит модель?



Как dbt v2 видит модель?



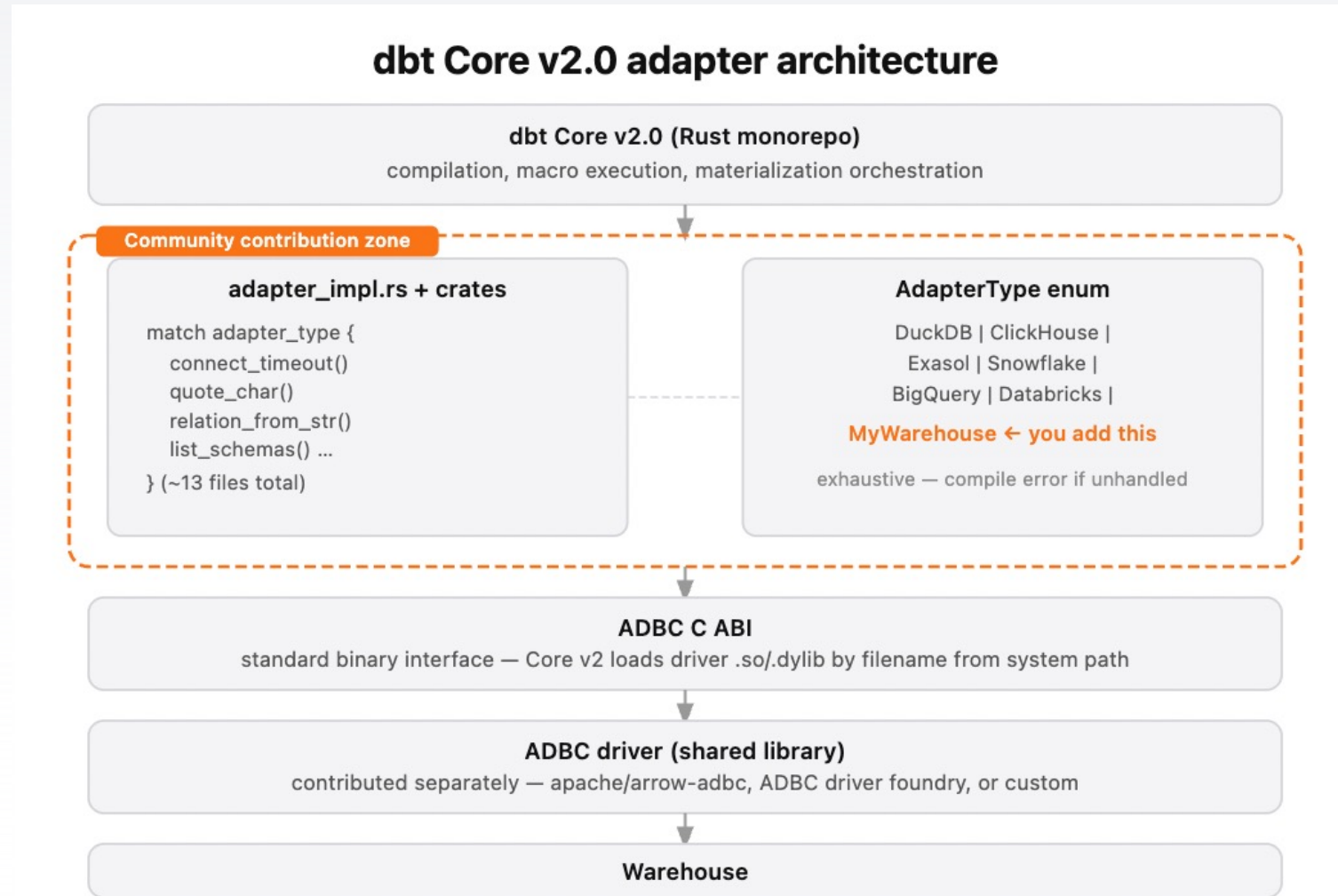
Input	<code>select dateadd('day', 1, '2025-01-01') as new_day</code>
Comprehends Level 1 + which function signatures are allowed, new column datatypes	<code>select dateadd('day', 1, '2025-01-01') as new_day</code> Snowflake's dateadd datepart int date_or_time_expr A date/timestamp/time column
Validates Level 1 + function names, argument count and data types	<code>select dateadd(datepart, int, date_or_time_expr) as new_day</code>

dbt v1 vs v2

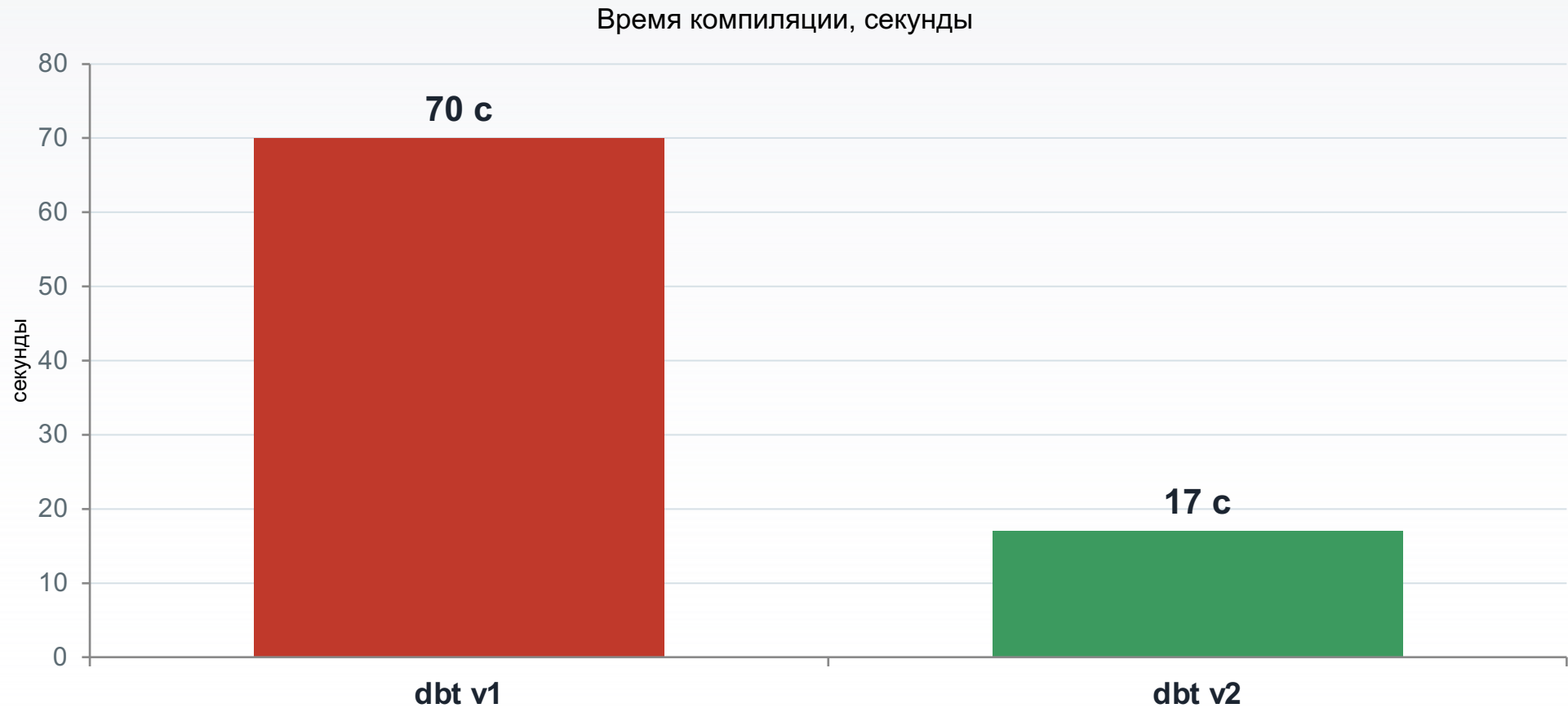


Критерий	v1	v2
Движок (runtime)	Python, Jinja-рендеринг	Rust-движок
SQL-обработка	Текстовый шаблон → rendered SQL, без понимания структуры	AST → DataFusion LP → типы/сигнатуры
Column-lineage	Нет	Да
Артефакты	JSON	Parquet + DuckDB для работы с метой
Язык адаптеров	Python	Rust*
Поддерживаемые адаптеры на GA	Десятки	BigQuery, Databricks, DuckDB, Redshift, Snowflake — GA; ClickHouse и Spark — в бете

*Rust Apapters



dbt v1 vs v2



<https://docs.getdbt.com/blog/dbt-v2-is-ga>



**Стоит ли мигрировать с dbt
прямо сейчас?**

А что именно болит?



- Боль организационная: naming, owners, слои?
- Число разработчиков? Стоимость dev-окружений?
- CI занимает десятки минут?
- Jinja стал вторым языком?
- Backfills – регулярная операционная проблема?
- Не мигрируем из-за хайпа технологии



Как выбрать инструмент под свои задачи?

Дерево принятия решений



Есть проблема с dbt?

нет -> оставаться на dbt

да

Проблема в долгом compile?

да -> dbt v2

нет

Проблема в повторном compute + backfill + dev-env?

да -> пилотировать SQLMesh

нет

Нужна портируемость?

да -> dbt v2 / SQLMesh / Bruin

нет -> оценить warehouse-native / vendor-managed:

- Уже в Snowflake, нужен low-code -> Coalesce
- Уже на BigQuery/GCP -> Dataform
- Весь pipeline (EL+T) в одном CLI -> Bruin
- Нужна enterprise no-code платформа -> Matillion
- Данные уже в Snowflake, отд. инструмент не нужен -> Snowflake Dynamic Tables
- Данные уже в Databricks -> Databricks LakeFlow

Триггер миграции



```
cost of staying > cost of changing
```

Есть ли жизнь после dbt?



➤ Да

➤ вместе с dbt v2 :)



Спасибо!
Вопросы?