



Data Productivity at Scale

April 18, 2024

About Me

- Co-Founder & Chief Architect at Tobiko Data
- Leading the development of SQLMesh
- Over 10 years of experience in data / ML infra
- Previously Netflix, Apple





Why am I here?

How do we test our pipeline changes today?



Why am I here?

Environments!

Why am I here?

- Creating environments for data is cumbersome



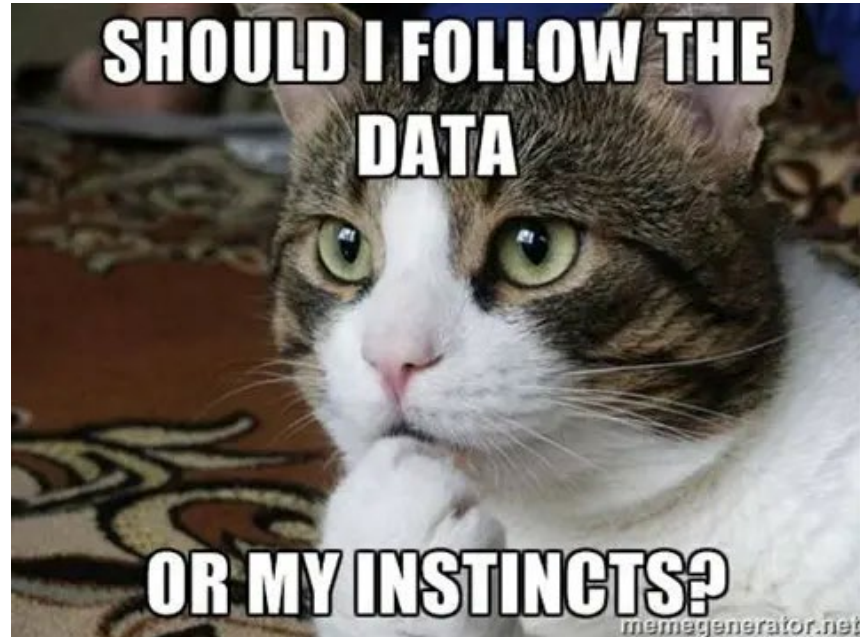
Why am I here?

- Populating new environments with data is inefficient



Why am I here?

- Development iterations are slow



Why am I here?

- Development iterations are slow

THE DATA CANT BE WRONG



IFYOURUNTHEJOBWITHNODATA

Why am I here?

- Production deployments are ad-hoc and error-prone



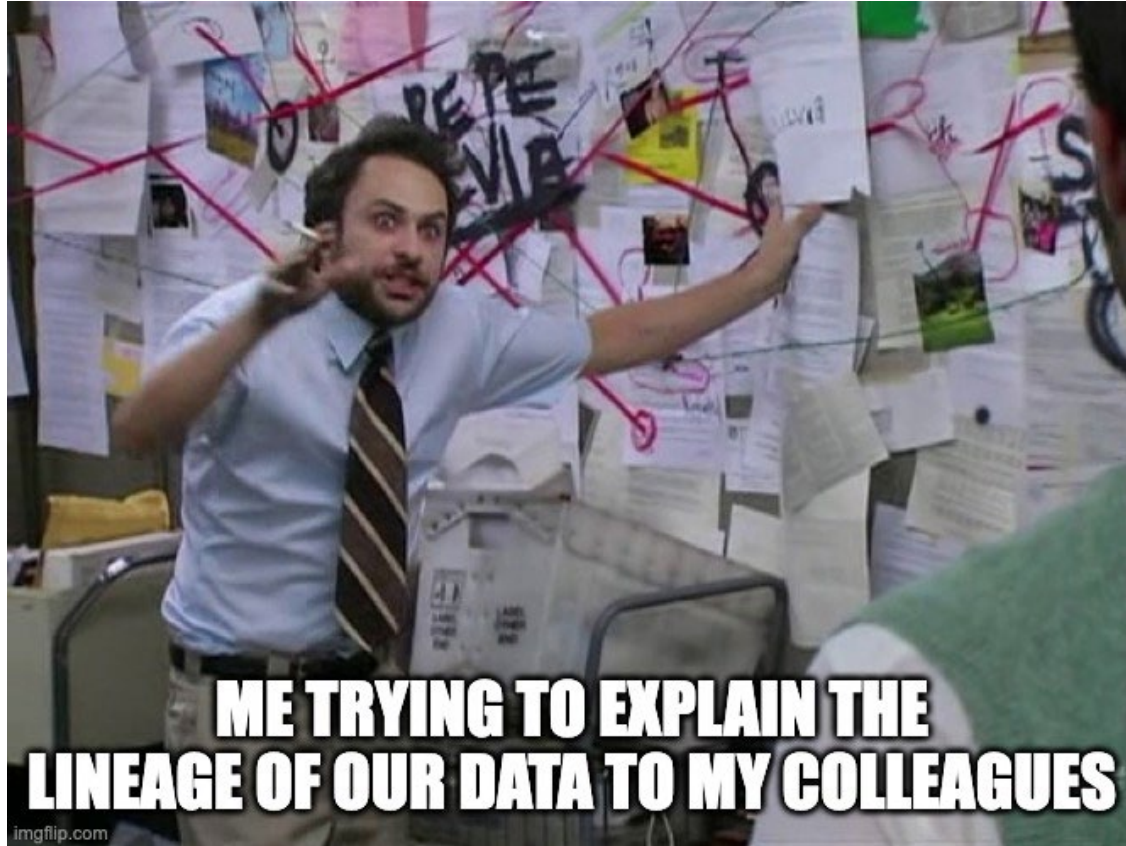
Why am I here?



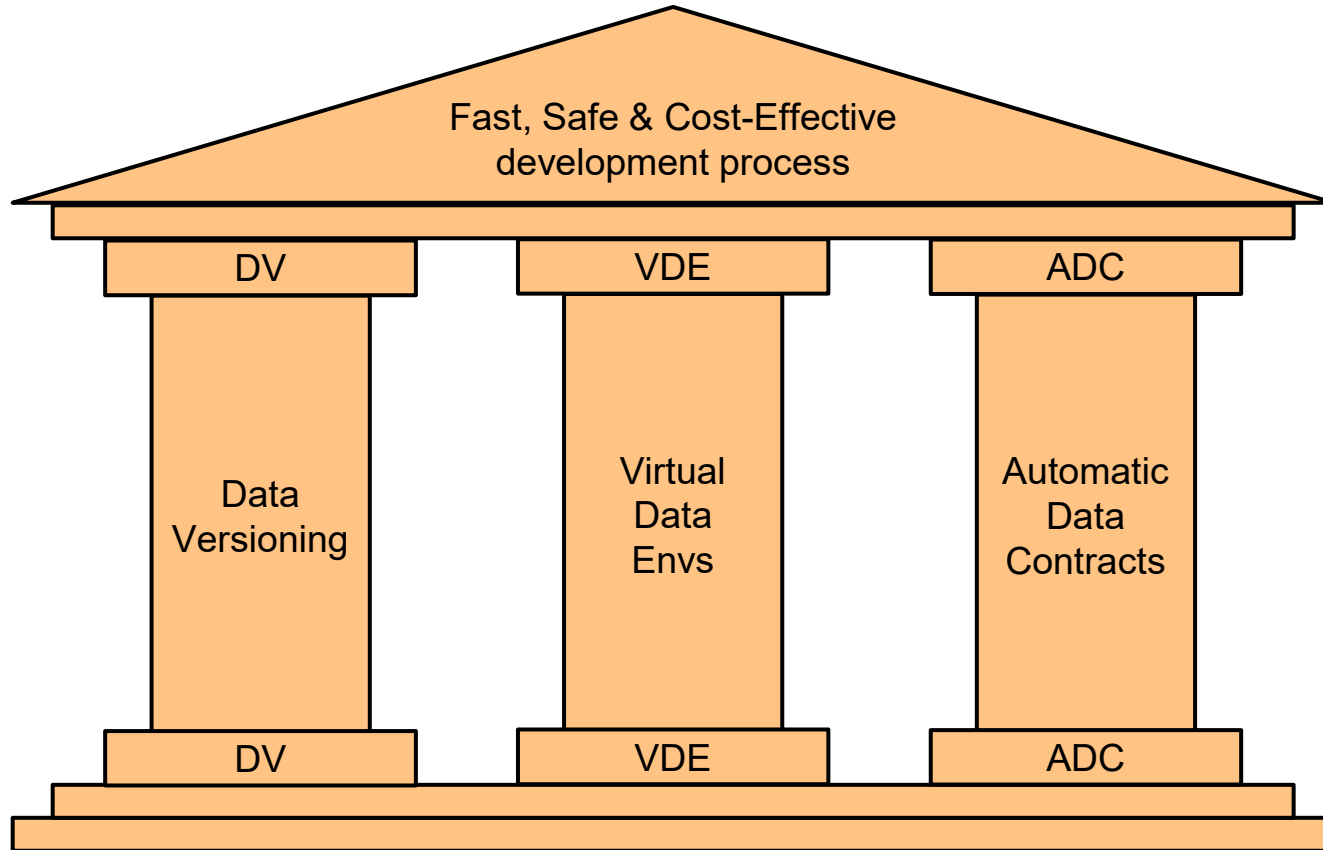
Scale



Scale Organizational Complexity



Three Pillars of Data Productivity



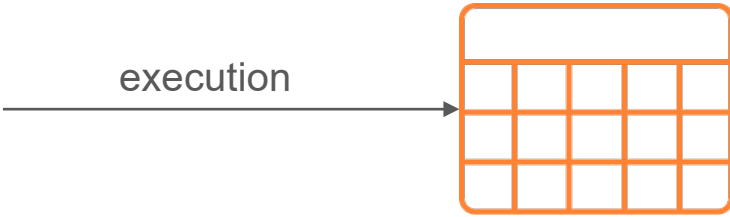
Model: T in ETL

```
SELECT a, b FROM source WHERE c > 0
```

Model

```
SELECT a, b  
FROM source  
WHERE c > 0
```

execution

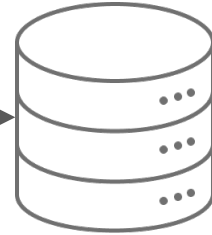


Model Snapshot

```
SELECT a, b  
FROM source  
WHERE c > 0
```

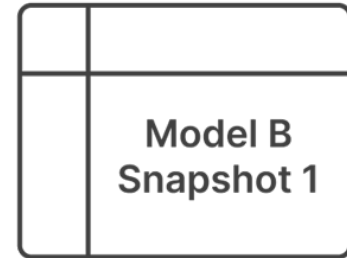
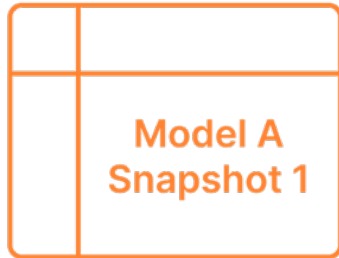
Model Fingerprint
(Hash)

Model
Snapshot



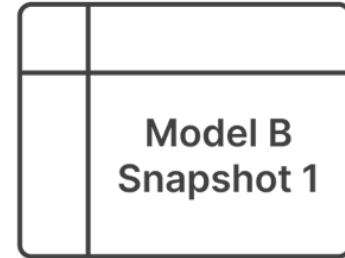
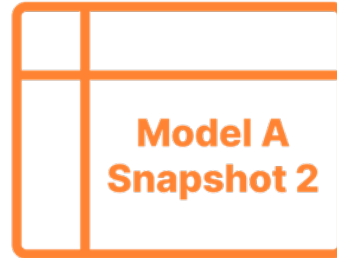
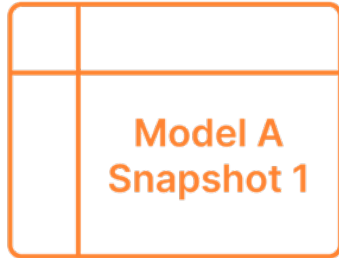
Snapshots in the Data Warehouse

Data Warehouse

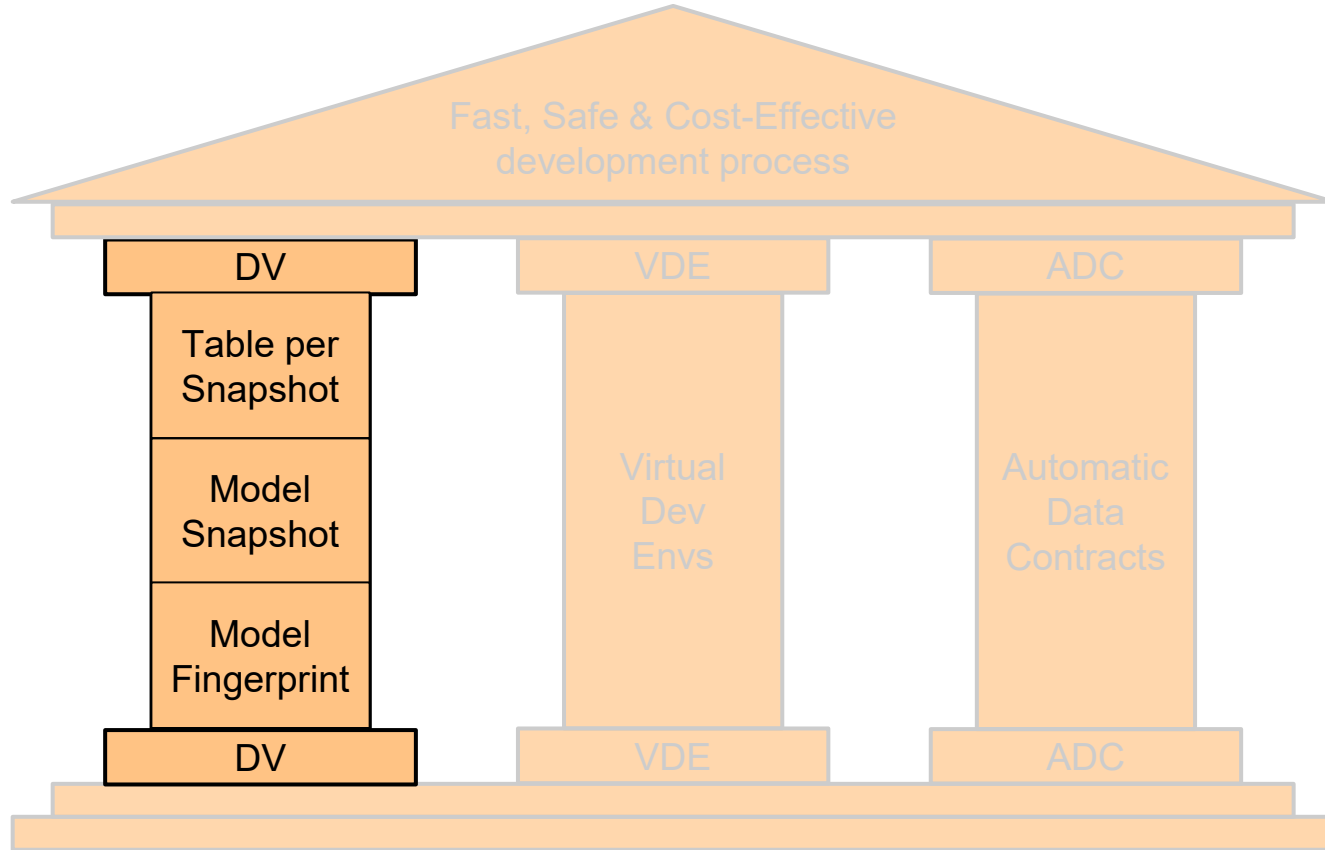


Snapshots in the Data Warehouse

Data Warehouse



Pillar of Data Versioning



Virtual Data Environments

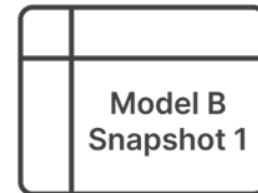
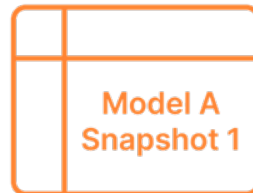
Virtual Layer

Physical Layer

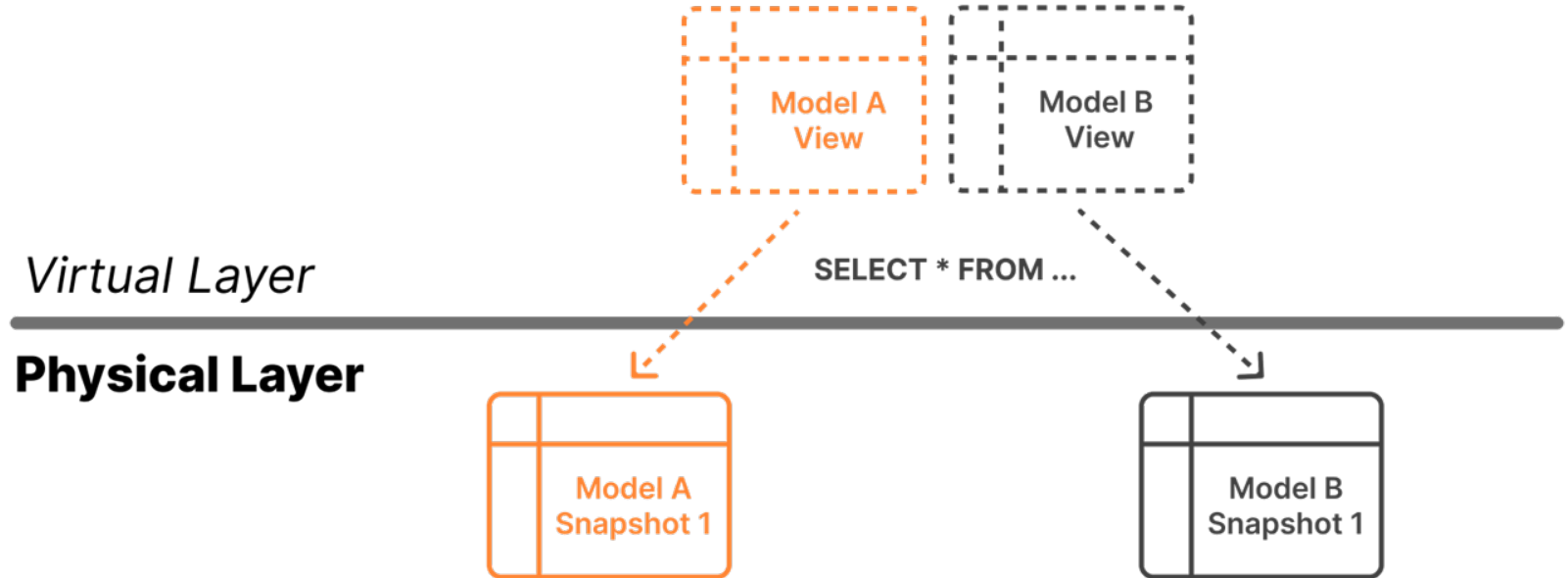
Virtual Data Environments

Virtual Layer

Physical Layer

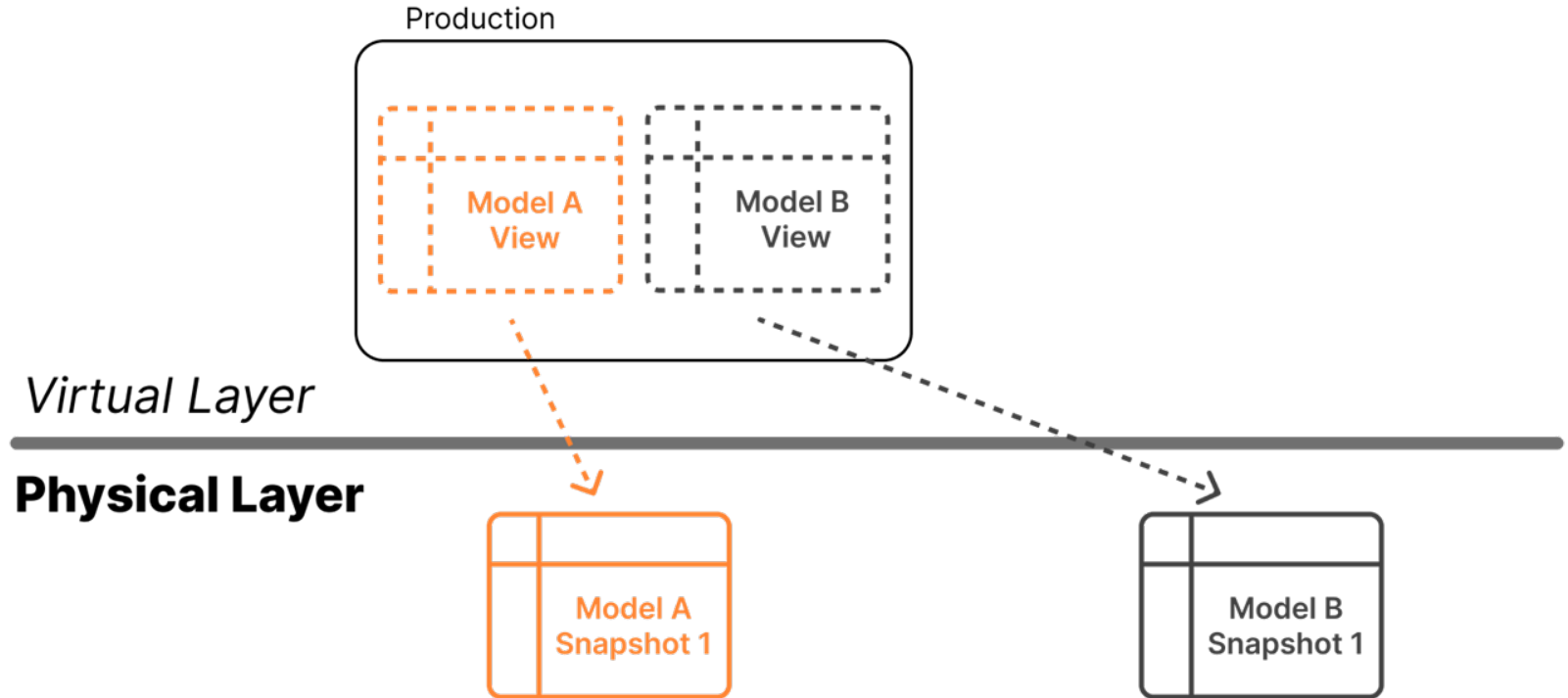


Virtual Data Environments



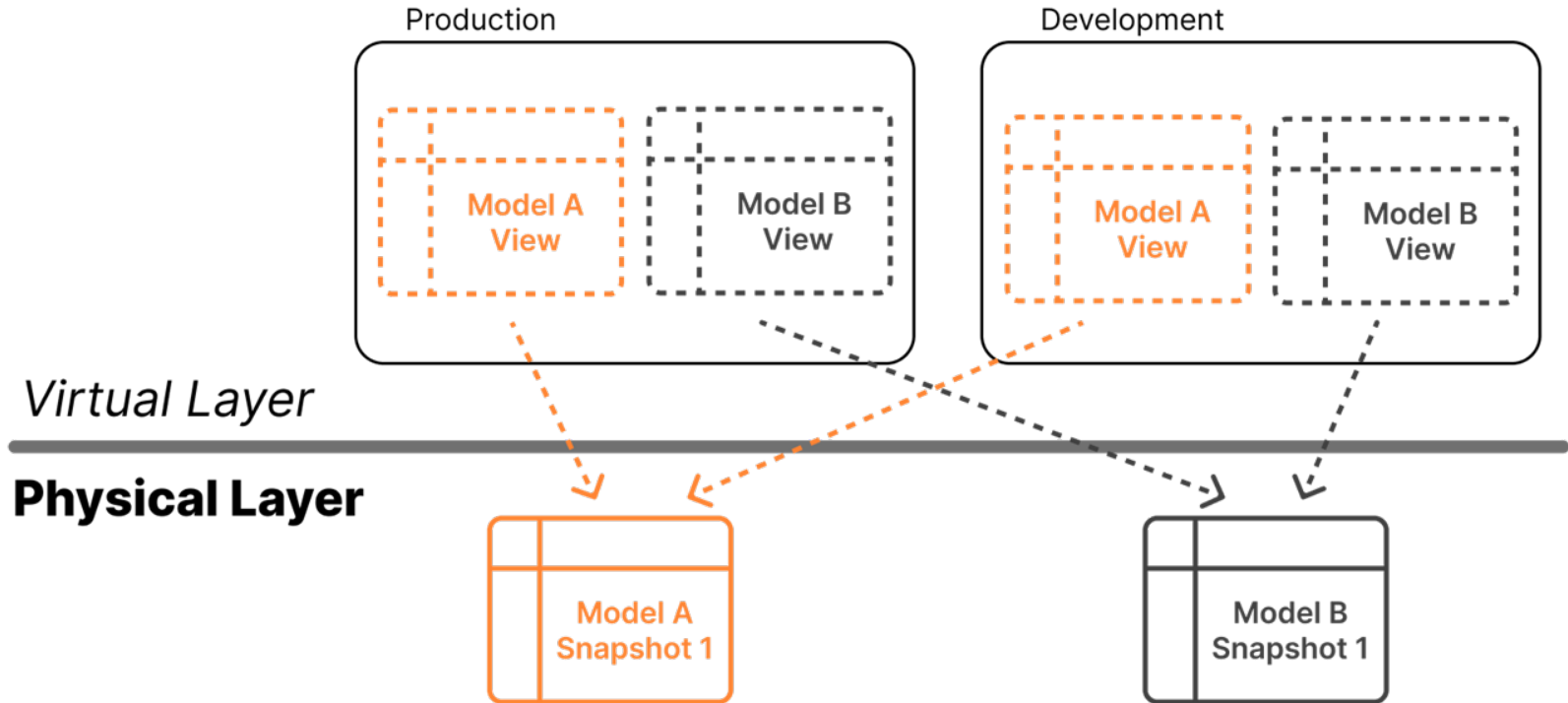
Virtual Data Environments: Flow

1. Starting production environment



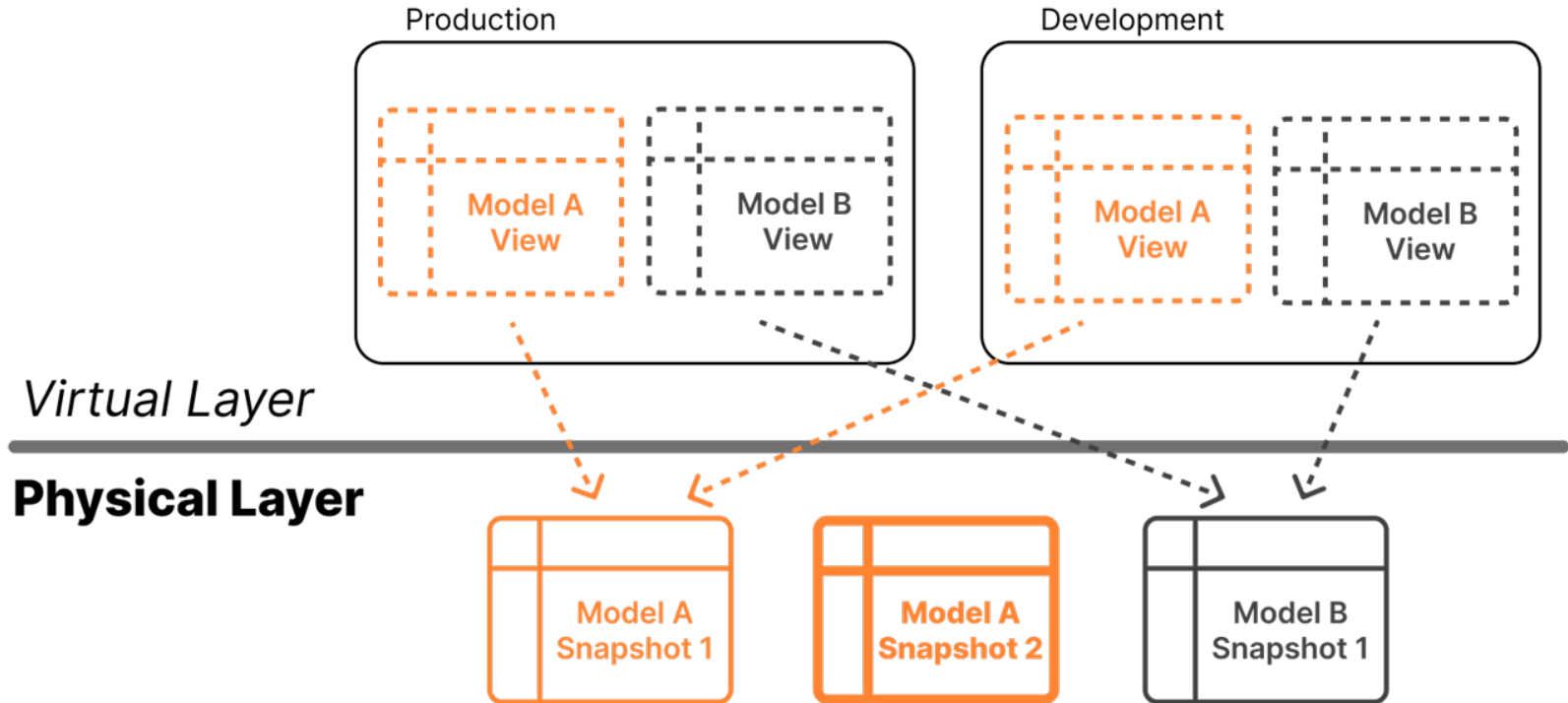
Virtual Data Environments: Flow

2. Create development as a mirror of production



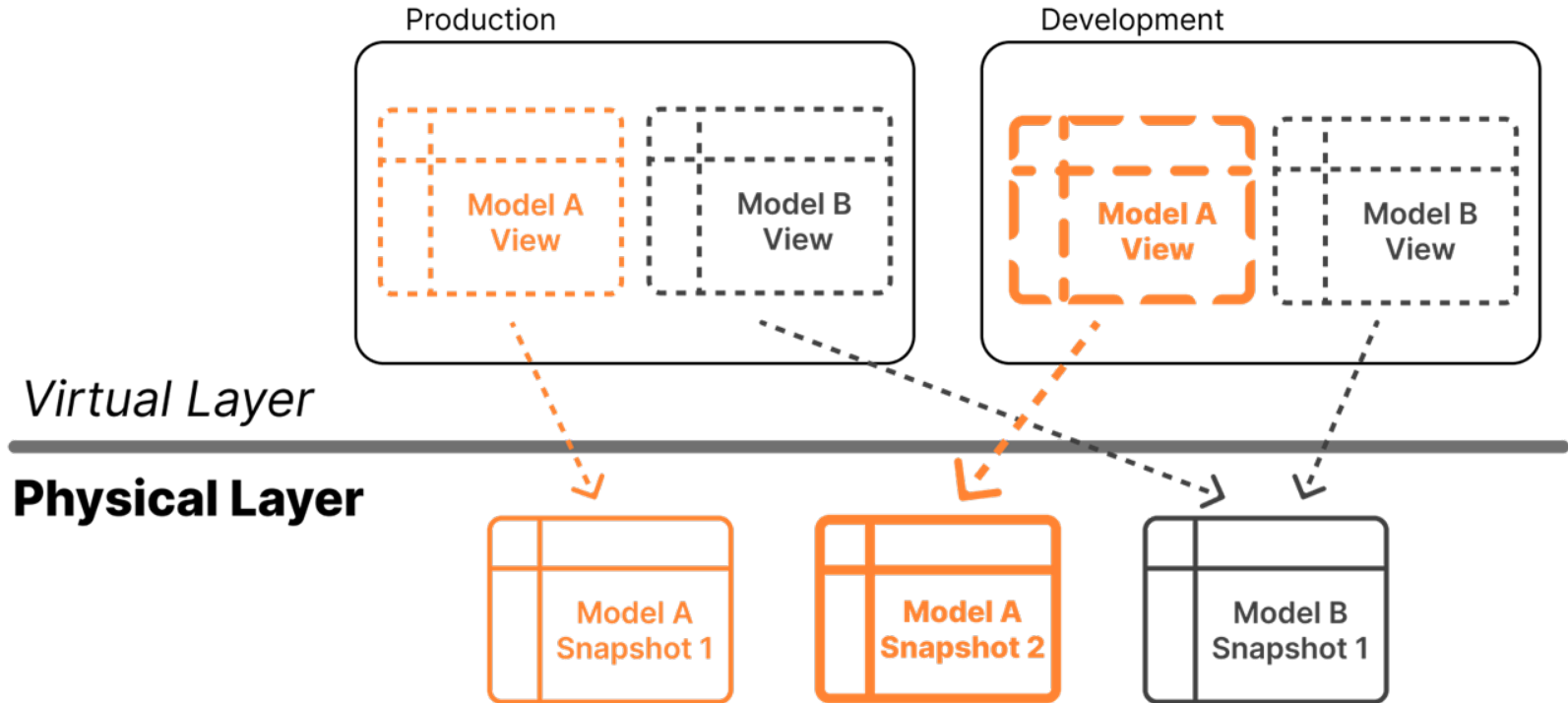
Virtual Data Environments: Flow

3. Make a change to Model A



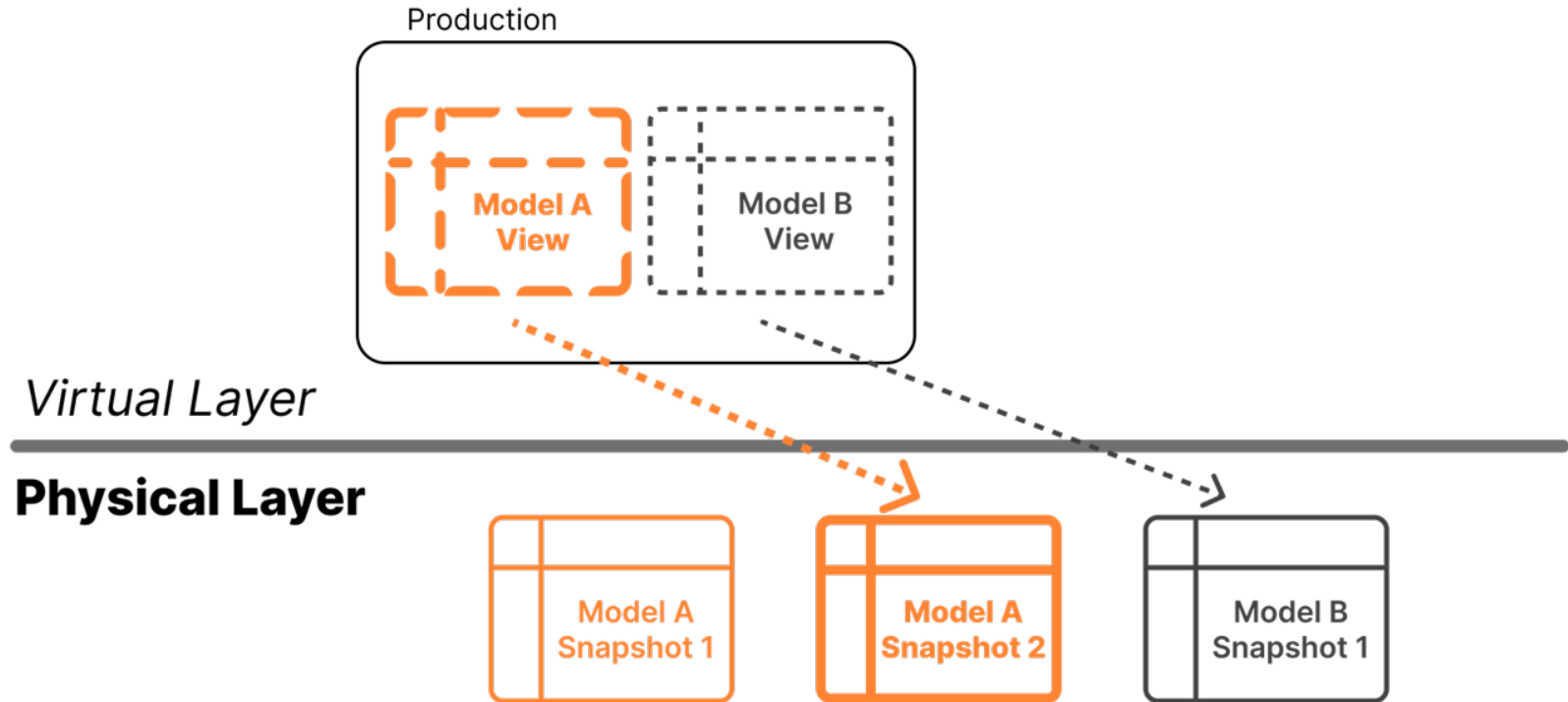
Virtual Data Environments: Flow

3. Make a change to Model A



Virtual Data Environments: Flow

4. Deploy model to production



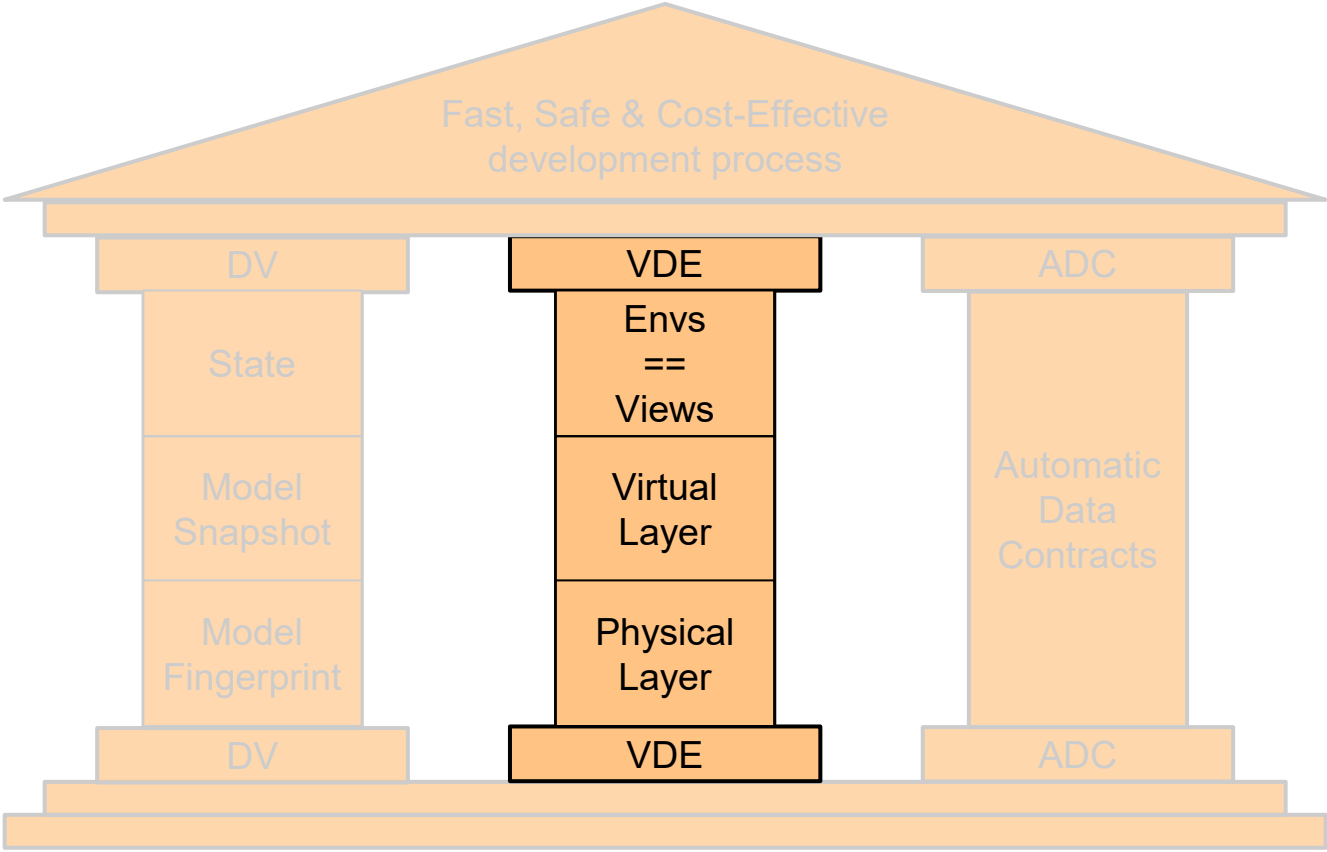
It's nice and all, but what if...

- The table is too large and costly to rebuild
- The model logic is not idempotent
- The historical data is not available upstream

It's nice and all, but what if...



Pillar of Virtual Data Environments

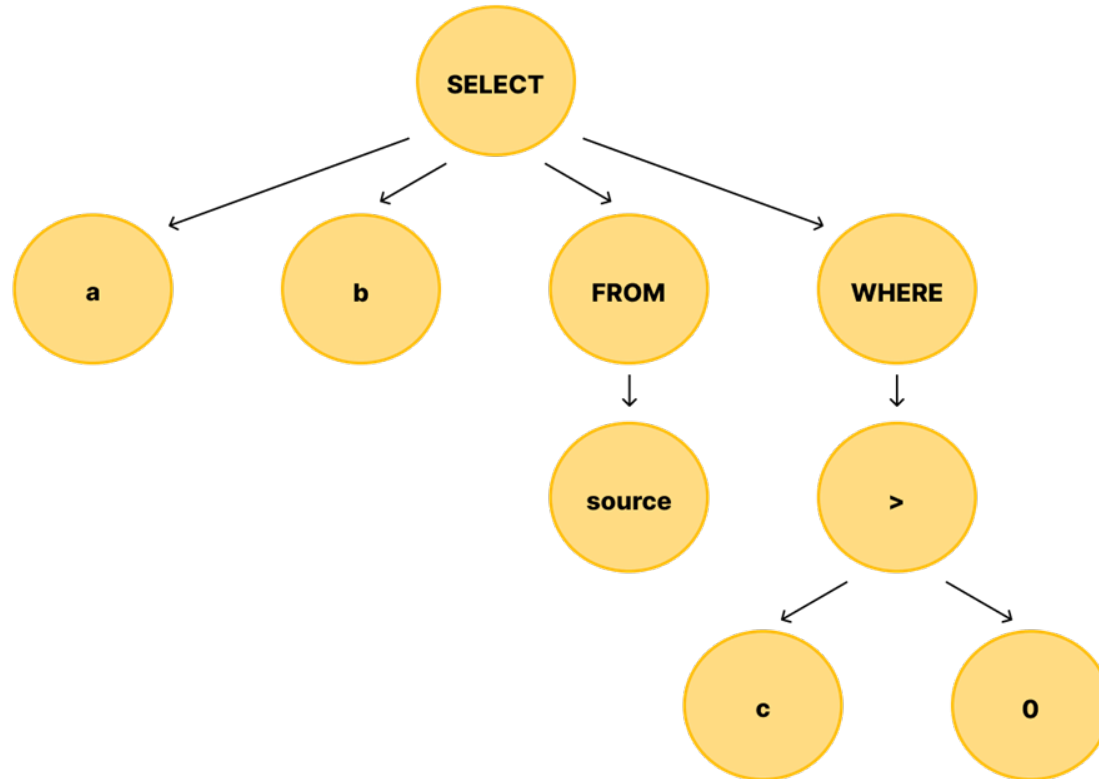


Semantic Understanding

```
SELECT a, b FROM source WHERE c > 0;
```

Semantic Understanding

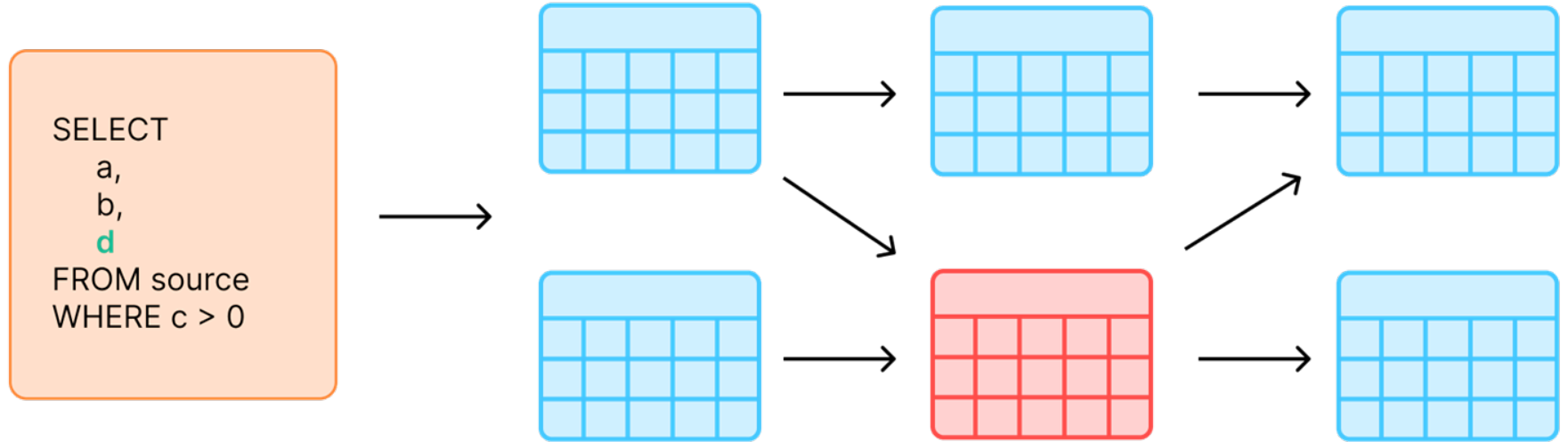
SELECT a, b **FROM** source **WHERE** c > 0;



Semantic Understanding

```
SELECT a, b, d FROM source WHERE c > 0
```

Non-breaking Change

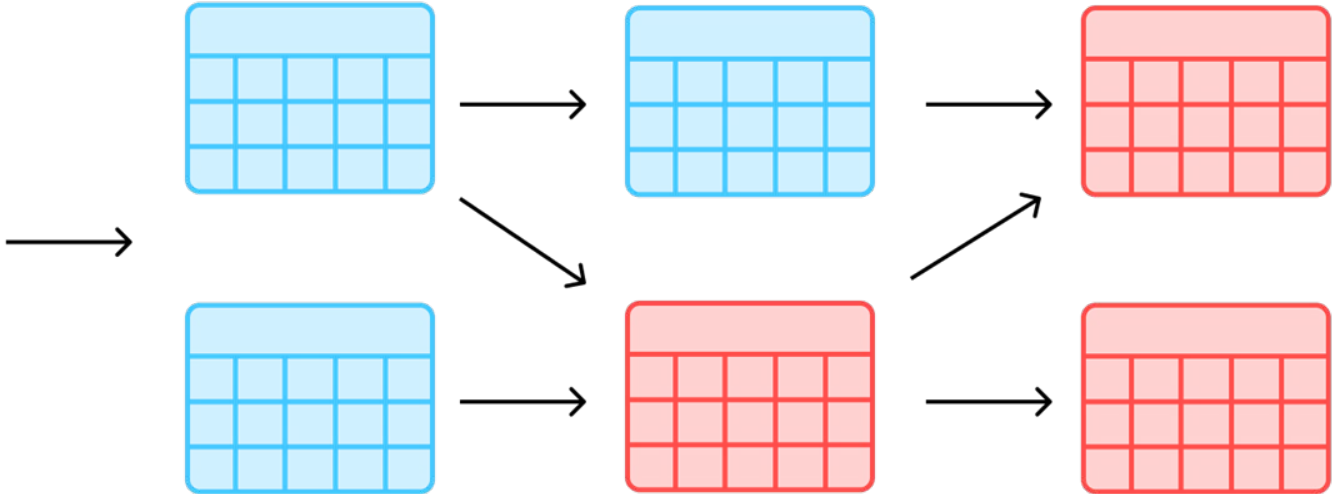


Breaking Change

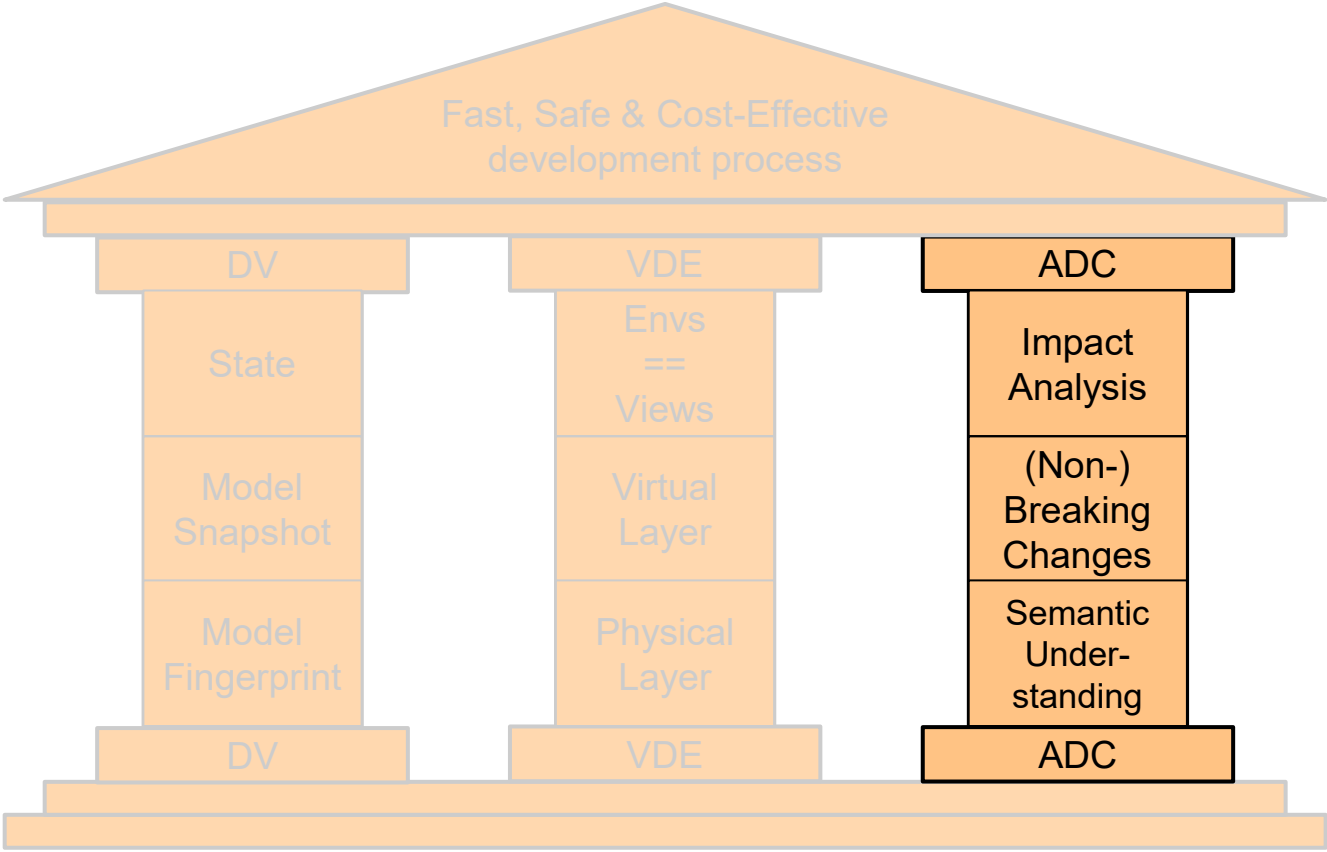
```
SELECT a, b FROM source WHERE c > 100
```

Breaking Change

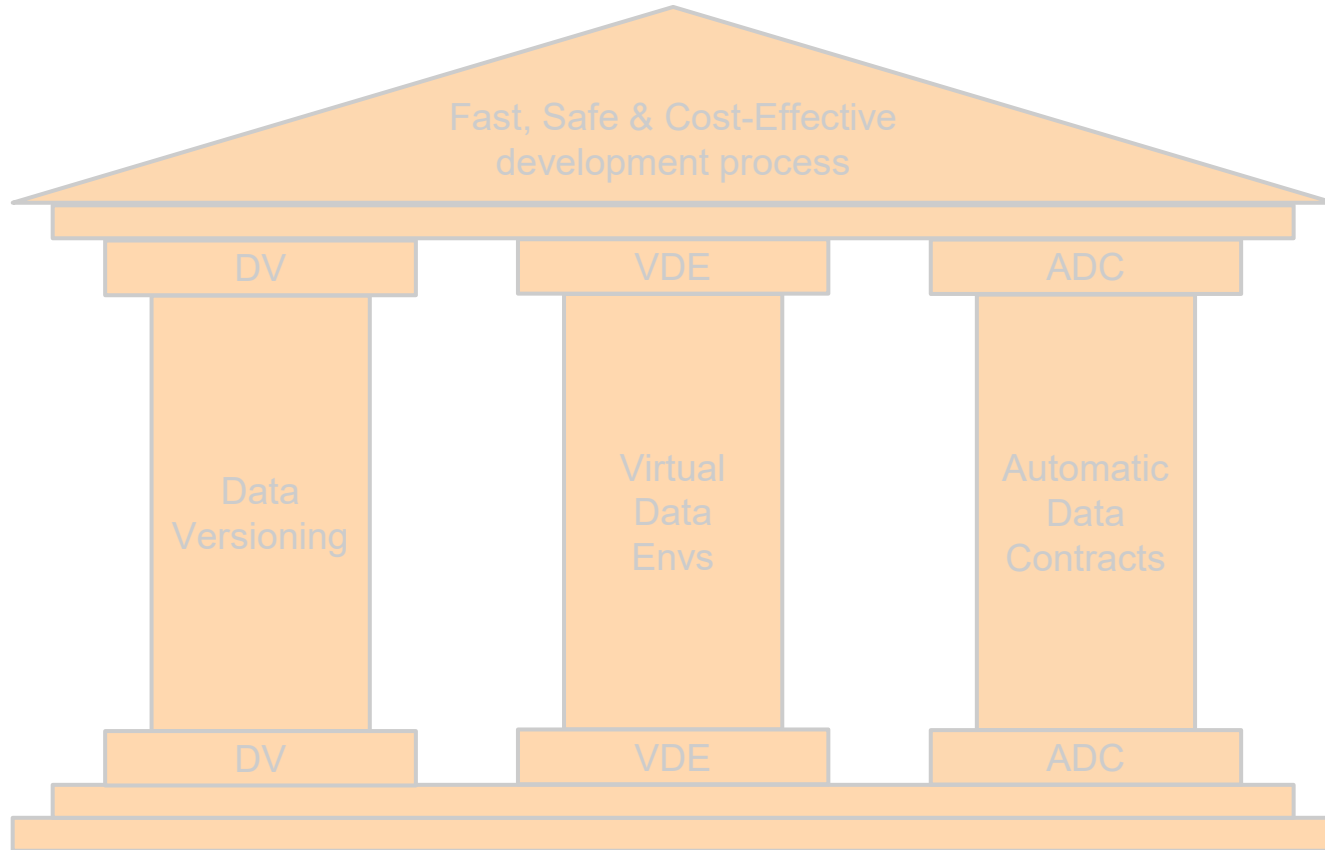
```
SELECT  
  a,  
  b,  
FROM source  
WHERE c > 100
```



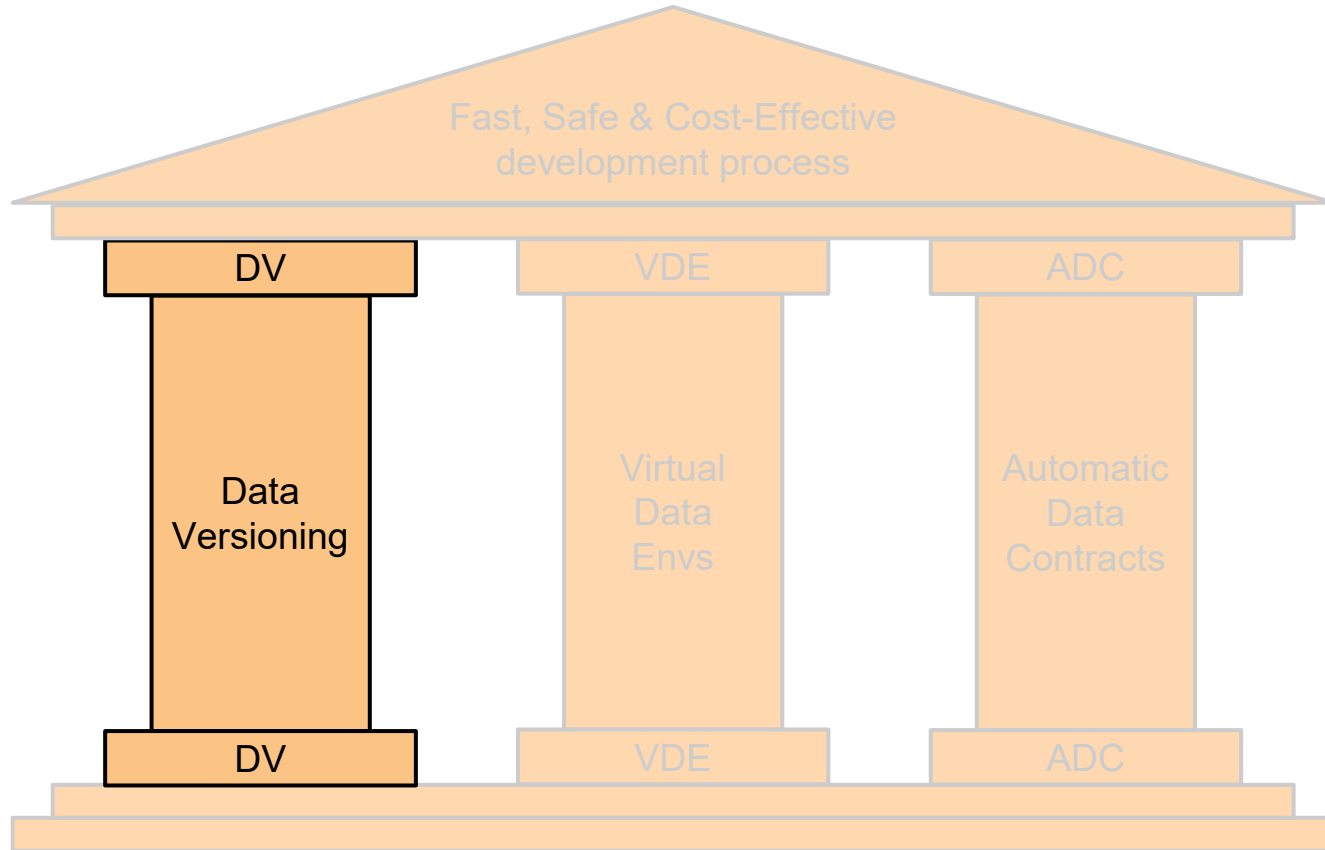
Pillar of Automatic Data Contracts



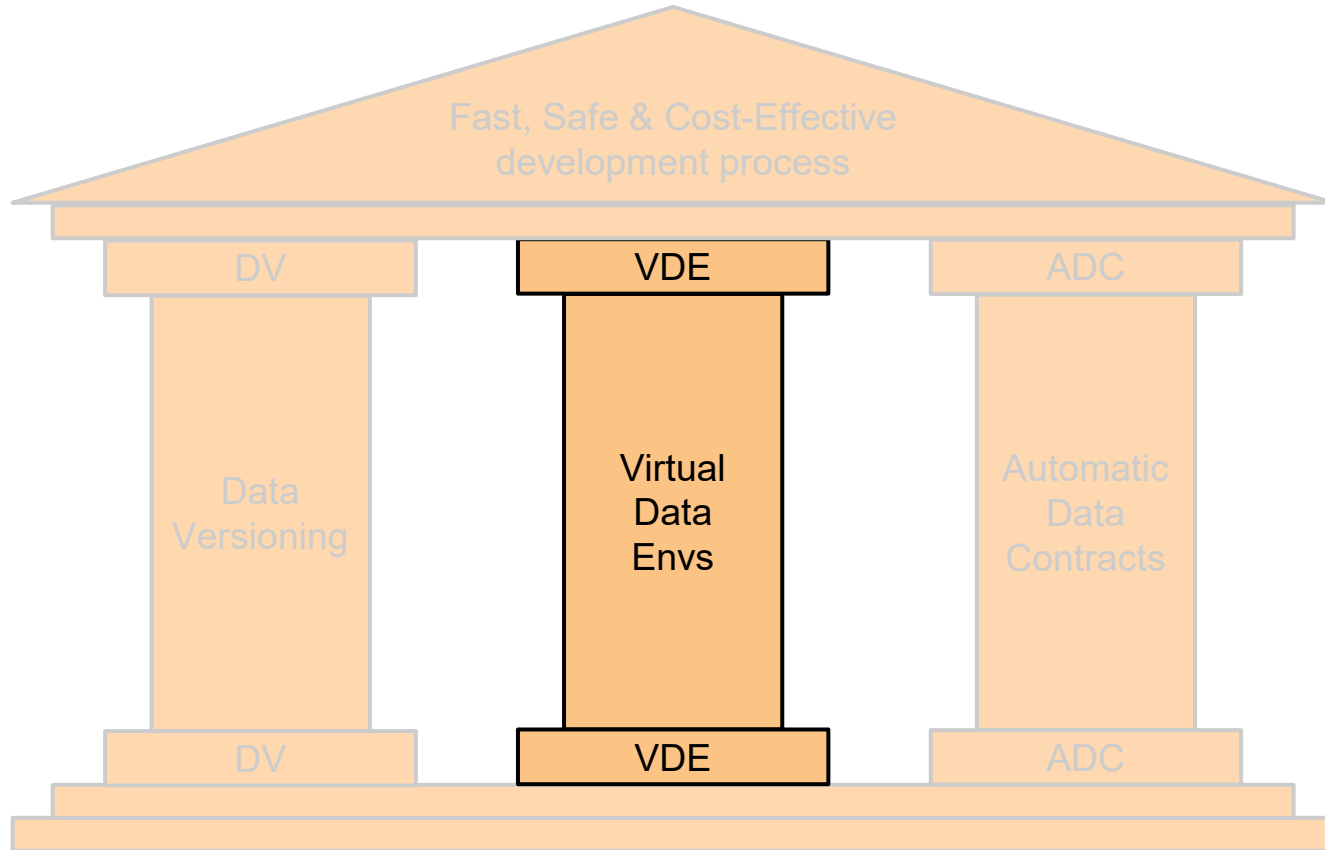
Three Pillars of Data Productivity



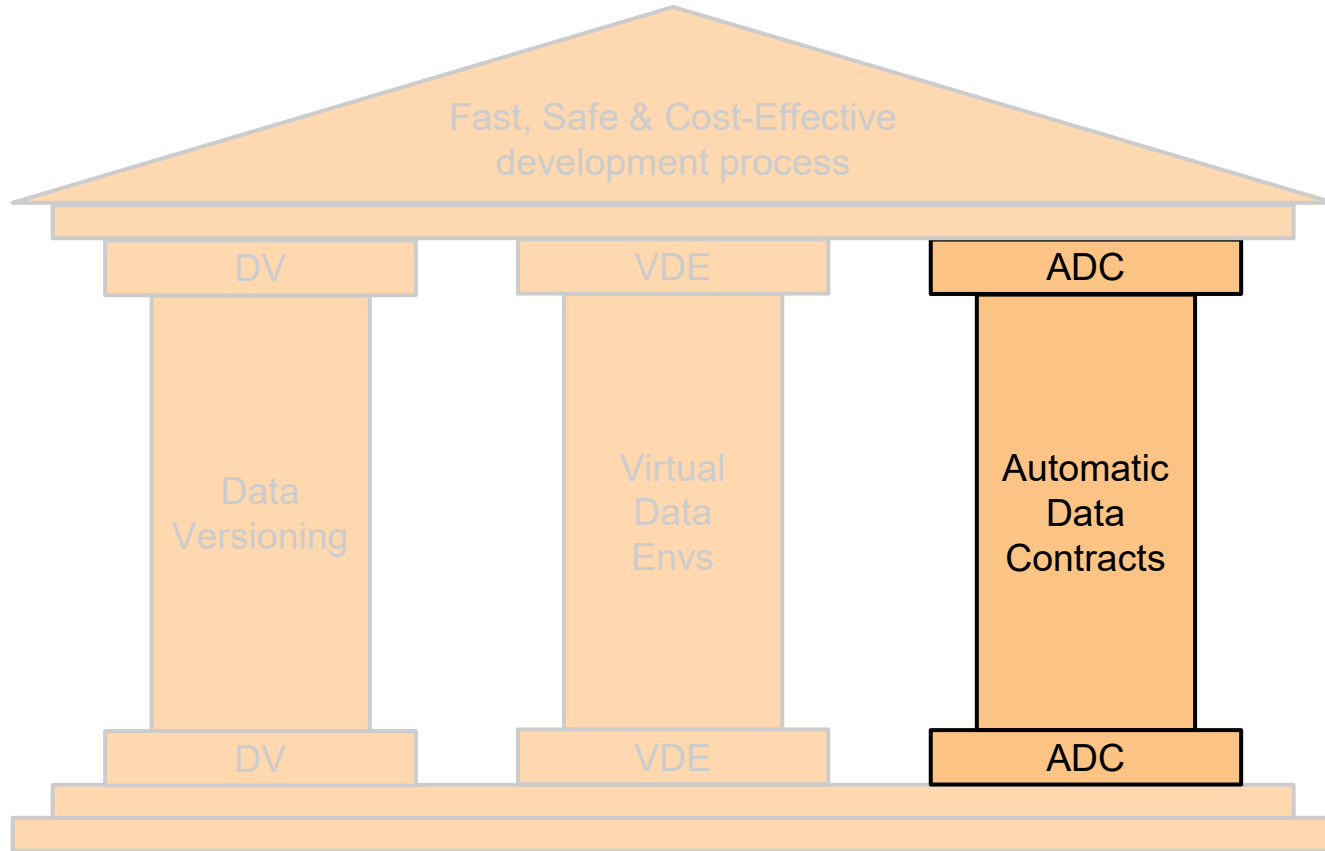
Three Pillars of Data Productivity



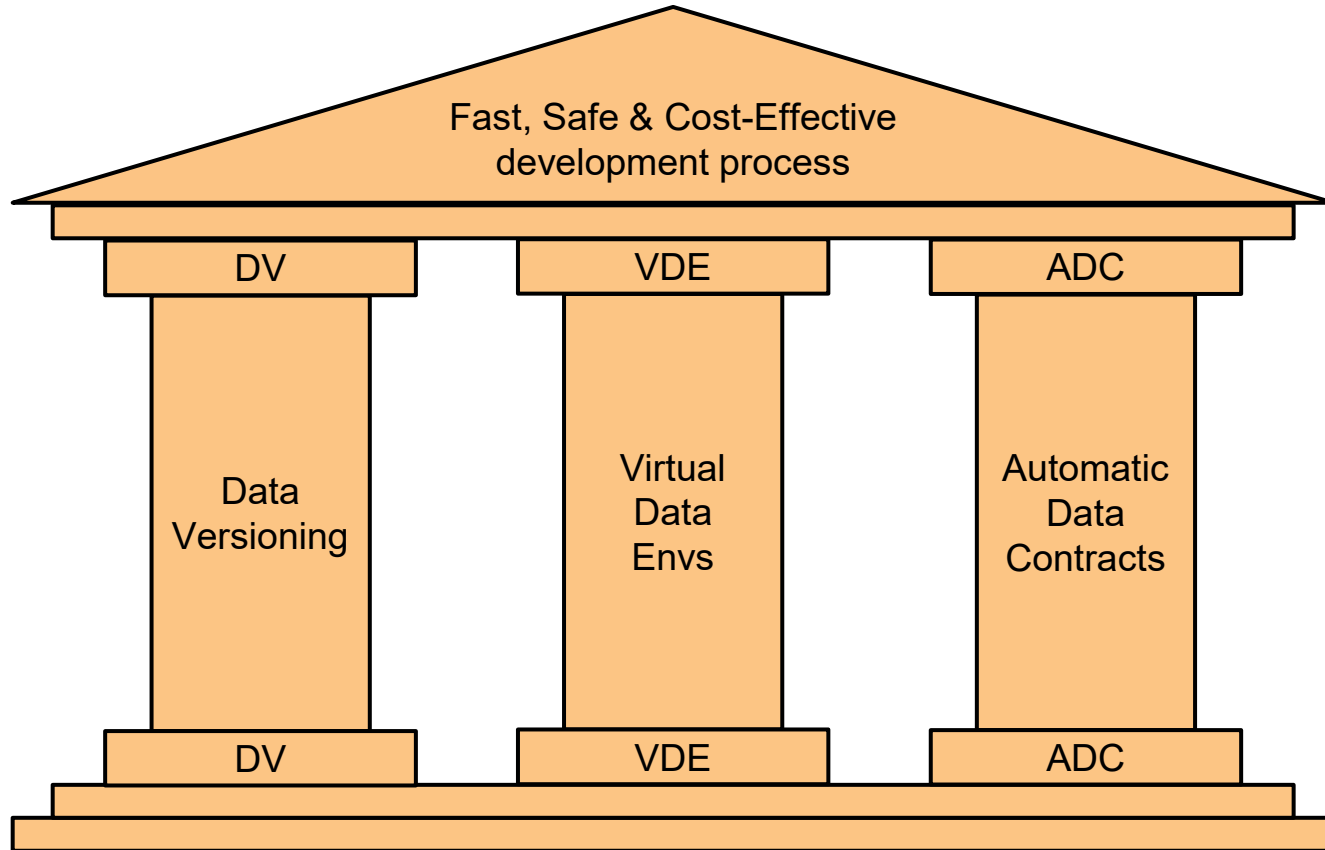
Three Pillars of Data Productivity



Three Pillars of Data Productivity



Three Pillars of Data Productivity



Putting Ideas into Practice with SQLMesh

About SQLMesh

- Open-source data transformation and modeling tool
- Pure SQL, no Jinja
- Batteries included: interval tracking, unit tests, CI/CD, etc
- Backwards compatible with dbt
- Brings **Three Pillars of Data Productivity** to life

Plan

Environment: prod

No Changes

No Errors

< >

+ title_performance.sql

```
1 MODEL (  
2   name sushiflix.title_performance,  
3   kind INCREMENTAL_BY_TIME_RANGE (  
4     time_column ds  
5   ),  
6   owner 'iaroslav',  
7   cron '@daily',  
8 );  
9  
10 SELECT  
11   ds,  
12   title_id,  
13   AVG(duration_ms) AS average_session_duration_ms,  
14   SUM(finished) / COUNT(*) AS finished_ratio,  
15 FROM  
16   sushiflix.playbacks  
17 WHERE  
18   ds BETWEEN @start_ds AND @end_ds  
19 GROUP BY  
20   ds,  
21   title_id  
22
```

Saved: ● Formatted: ● Language: SQL Dialect: duckdb SQLMesh Type: model

Lineage

sushiflix.title_performance

All: 3 [Reset](#) Sources: 1 Upstream/Downstream: 2

Find

Show

SEED sushiflix.playbacks 7	
finished	BIGINT
Filter items	
playback_id	BIGINT
user_id	BIGINT
title_id	BIGINT
duration_ms	BIGINT
liked	BIGINT

SQL sushiflix.title_performance 4	
finished_ratio	DOUBLE
ds	TEXT
title_id	BIGINT
average_session_duration_ms	DOUBLE

SQL sushiflix.top_titles 2	
finished_ratio	DOUBLE
title_id	BIGINT

Plan Environment: prod No Changes No Errors

title_performance.sql

```
1 MODEL (  
2   name sushiflix.title_performance,  
3   kind INCREMENTAL_BY_TIME_RANGE (  
4     time_column ds  
5   ),  
6   owner 'iaroslav',  
7   cron '@daily',  
8 );  
9  
10 SELECT  
11   ds,  
12   title_id,  
13   AVG(duration_ms) AS average_session_duration_ms,  
14   SUM(finished) / COUNT(*) AS finished_ratio,  
15 FROM  
16   sushiflix.playbacks  
17 WHERE  
18   ds BETWEEN @start_ds AND @end_ds  
19 GROUP BY  
20   ds,  
21   title_id  
22
```

prod (remote)
Production Environment

dev

Saved: Formatted: Language: SQL Dialect: duckdb SQLMesh Type: model

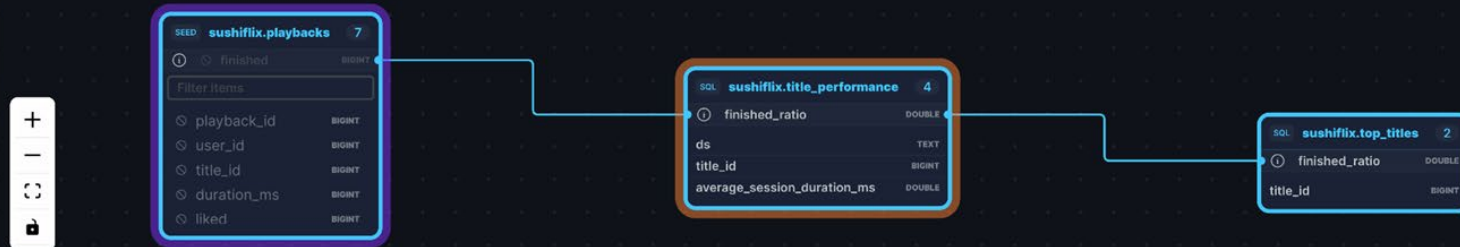
Lineage





sushiflix.title_performance

All: 3 Sources: 1 Upstream/Downstream: 2

Find



Show



Plan  Environment: dev  Changes   No Errors

```
1 MODEL (  
2   name sushiflix.title_performance,  
3   kind INCREMENTAL_BY_TIME_RANGE (  
4     time_column ds  
5   ),  
6   owner 'iaroslav',  
7   cron '@daily'  
8 );  
9  
10 SELECT  
11   ds,  
12   title_id,  
13   AVG(duration_ms) AS average_session_duration_ms,  
14   SUM(finished) / COUNT(*) AS finished_ratio,  
15   COUNT_IF(finished = 1 AND liked = 1) / SUM(finished) AS finished_and_liked_ratio  
16 FROM sushiflix.playbacks  
17 WHERE  
18   ds BETWEEN @start_ds AND @end_ds  
19 GROUP BY  
20   ds,  
21   title_id
```

Directly Modified
↻ sushiflix_dev.title_performance


Saved:  Formatted:  Language: SQL Dialect: duckdb SQLMesh Type: model

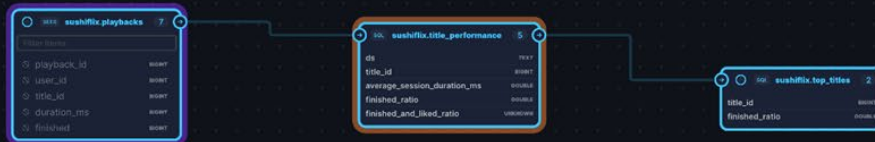
Lineage

sushiflix.title_performance

All: 3 Sources: 1 Upstream/Downstream: 2

Find

Show 



Plan Environment: dev Changes 1 1 No Errors

```
1 MODEL (  
2   name sushiflix.title_performance,  
3   kind INCREMENTAL_BY_TIME_RANGE (  
4     time_column ds  
5   ),  
6   owner 'iaroslav',  
7   cron '@daily'  
8 );  
9  
10 SELECT  
11   ds,  
12   title_id,  
13   AVG(duration_ms) AS average_session_duration_ms,  
14   SUM(finished) / COUNT(*) AS finished_ratio,  
15   COUNT_IF(finished = 1 AND liked = 1) / SUM(finished) AS finished_and_liked_ratio  
16 FROM sushiflix.playbacks  
17 WHERE  
18   ds BETWEEN @start_ds AND @end_ds  
19 GROUP BY  
20   ds,  
21   title_id
```

Indirectly Modified
sushiflix_dev.top_titles

Saved: Formatted Language: SQL Dialect: duckdb SQLMesh Type: model

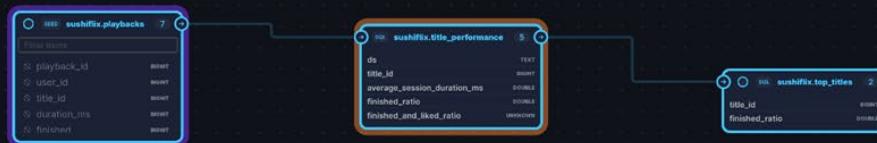
Lineage

sushiflix.title_performance

All: 3 Sources: 1 Upstream/Downstream: 2

Find

Show





< >

Plan

Environment: dev



Changes

1

1

No Errors



Start Date (UTC)

2023-12-13

The start datetime of the interval

End Date (UTC)

2022-12-13

The end datetime of the interval



Additional Options



✓ Tests Skipped

● Changes

Modified Directly

sushiflix__dev.title_performance

Non-Breaking Change



Modified Indirectly

sushiflix__dev.top_titles

● Backfills

Models 1

sushiflix__dev.title_performance

Apply Changes And Backfill

Start Over

Go Back



< >

Additional Options

Plan

Environment: dev

Changes

1

1

No Errors



✓ Tests Skipped

Changes

Modified Directly

sushiflix_dev.title_performance

└ sushiflix_dev.top_titles

Non-Breaking Change

- Breaking Change
It will rebuild all models
- Non-Breaking Change
It will exclude all indirect models caused by this change
- Forward-Only Change
The change requires no rebuilding

```
---  
+++  
  
@@ -11,7 +11,8 @@  
  
    ds,  
    title_id,  
    AVG(duration_ms) AS average_session_duration_ms,  
-   SUM(finished) / COUNT(*) AS finished_ratio  
+   SUM(finished) / COUNT(*) AS finished_ratio,  
+   COUNT_IF(finished = 1 AND liked = 1) / SUM(finished) AS finished_and_liked_ratio  
FROM sushiflix.playbacks  
WHERE  
    ds BETWEEN @start_ds AND @end_ds
```

Apply Changes And Backfill

Start Over

Go Back

Start Date (UTC)

2023-12-13

The start datetime of the interval

End Date (UTC)

2022-12-13

The end datetime of the interval

Additional Options

✓ Tests Skipped

+ Changes

- Backfills

Models 1

↻ sushiflix_dev.title_performance



Apply Changes And Backfill

Start Over

Go Back

Plan

Environment: dev

No Changes

No Errors

2023-12-13

2022-12-13

The start datetime of the interval

The end datetime of the interval

Additional Options (Read Only)

✓ No Tests

+ Changes

+ Backfills

Evaluation started at 2024-05-28 23:01:43

+ Snapshot Tables Created

✓ No Models To Restate

- Backfilled

Target Environment	dev	1 of 1 task		1 of 1 batch		100%
<hr/>						
2024-05-27	-	2024-05-27	sushiflix__dev.title_performance	1 of 1 batch		0:01
<hr/>						

+ Environment Promoted

Evaluation stopped at 2024-05-28 23:01:43

Go Back

Plan Environment: dev Changes 1 2 No Errors

title_performance.sql

```
1 MODEL (  
2   name sushiflix.title_performance,  
3   kind INCREMENTAL_BY_TIME_RANGE (  
4     time_column ds  
5   ),  
6   owner 'iaroslav',  
7   cron '@daily'  
8 );  
9  
10 SELECT  
11   ds,  
12   title_id,  
13   AVG(duration_ms) AS average_session_duration_ms,  
14   SUM(finished) / COUNT_IF(duration_ms > 0) AS finished_ratio  
15 FROM sushiflix.playbacks  
16 WHERE  
17   ds BETWEEN @start_ds AND @end_ds  
18 GROUP BY  
19   ds,  
20   title_id
```

Directly Modified
sushiflix_dev.title_performance

Save

Saved: Formatted Language: SQL Dialect: duckdb SQLMesh Type: model

Lineage

sushiflix.title_performance

All: 3

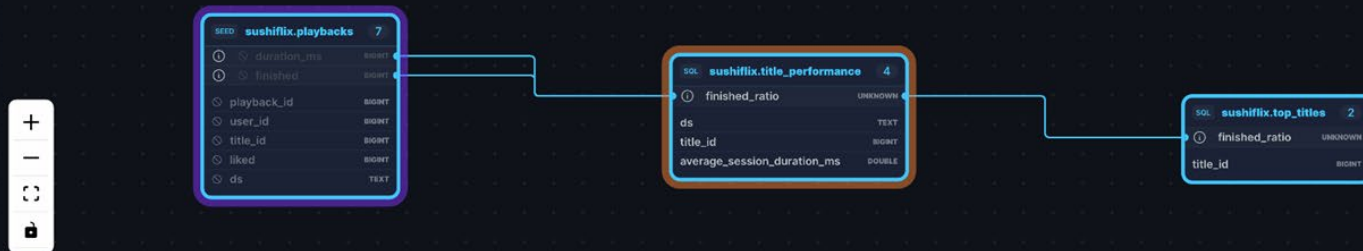
Reset

Sources: 1

Upstream/Downstream: 2

Find

Show





< >

Plan

Environment: dev

Changes 1 2

No Errors



Start Date (UTC)

2023-12-13

The start datetime of the interval

End Date (UTC)

2022-12-13

The end datetime of the interval

Additional Options



✓ Tests Skipped

Changes

Modified Directly

sushiflix_dev.title_performance

Breaking Change

Modified Indirectly

sushiflix_dev.top_titles

Backfills

Models 2

sushiflix_dev.title_performance

sushiflix_dev.top_titles

Apply Changes And Backfill

Start Over

Go Back

Plan

Environment: dev

Changes 1 2

No Errors

< >

Start Date (UTC)

2023-12-13

The start datetime of the interval

End Date (UTC)

2022-12-13

The end datetime of the interval

Additional Options

✓ Tests Skipped

⊕ Changes

⊖ Backfills

Models 2

- sushiflix_dev.title_performance
- sushiflix_dev.top_titles



Apply Changes And Backfill

Start Over

Go Back

Plan

Environment: dev

No Changes

No Errors

The start datetime of the interval

The end datetime of the interval

Additional Options (Read Only)

✓ No Tests

+ Changes

+ Backfills

Evaluation started at 2024-05-29 01:51:58

+ Snapshot Tables Created

✓ No Models To Restate

- Backfilled

Target Environment	dev	2 of 2 tasks		2 of 2 batches		100%
2024-05-27	-	2024-05-27		sushiflix_dev.top_titles		1 of 1 batch 0:01
2024-05-27	-	2024-05-27		sushiflix_dev.title_performance		1 of 1 batch 0:01

+ Environment Promoted

Evaluation stopped at 2024-05-29 01:51:58

Go Back

Plan Environment: dev No Changes No Errors

title_performance.sql

```
1 MODEL (  
2   name sushiflix.title_performance,  
3   kind INCREMENTAL_BY_TIME_RANGE (  
4     time_column ds  
5   ),  
6   owner 'iaroslav',  
7   cron '@daily'  
8 );  
9  
10 SELECT  
11   ds,  
12   title_id,  
13   AVG(duration_ms) AS average_session_duration_ms,  
14   SUM(finished) / COUNT_IF(duration_ms > 0) AS finished_ratio  
15 FROM sushiflix.playbacks  
16 WHERE  
17   ds BETWEEN @start_ds AND @end_ds  
18 GROUP BY  
19   ds,  
20   title_id
```

dev (remote)
prod (remote) ★
Production Environment
Environment Add

Saved: Formatted: Language: SQL Dialect: duckdb SQLMesh Type: model

Lineage

sushiflix.title_performance

All: 3 Sources: 1 Upstream/Downstream: 2

Find

Show






Table Name	Columns
sushiflix.playbacks	playback_id, user_id, title_id, duration_ms, finished

Table Name	Columns
sushiflix.title_performances	ds, title_id, average_session_duration_ms, finished_ratio

Table Name	Columns
sushiflix.top_titles	title_id, finished_ratio


< >

Plan  Environment: prod  Changes  No Errors

Prod Environment

Start Date (UTC) End Date (UTC)



The start datetime of the interval The end datetime of the interval

Additional Options (Read Only) 


✓ Tests Skipped

− Changes

Modified Directly

 sushiflix.title_performance Breaking Change 

Modified Indirectly

 sushiflix.top_titles

✓ No Backfills

+ Virtual Update

Apply Virtual Update

Start Over Go Back



< >

Plan

Environment: prod ▾

Changes

No Errors



Prod Environment



Start Date (UTC)

2023-12-13

The start datetime of the interval

End Date (UTC)

2022-12-13

The end datetime of the interval



Additional Options (Read Only)

 Tests Skipped Changes No Backfills Virtual Update

Apply Virtual Update

Start Over

Go Back

Plan Environment: prod No Changes No Errors

Prod Environment

Start Date (UTC)	End Date (UTC)
<input type="text" value="2023-12-13"/>	<input type="text" value="2023-12-13"/>
<small>The start datetime of the interval</small>	<small>The end datetime of the interval</small>

Additional Options (Read Only)

- ✓ No Tests
- ➕ Changes
- ✓ No Backfills
- ➕ Virtual Update Completed

Evaluation started at 2024-05-29 01:54:52

- ➕ Snapshot Tables Created
- ✓ No Models To Restate
- ➕ Environment Promoted

Evaluation stopped at 2024-05-29 01:54:52

[Go Back](#)



Join us on Slack!

<https://tobikodata.com/slack>

Thank you!

