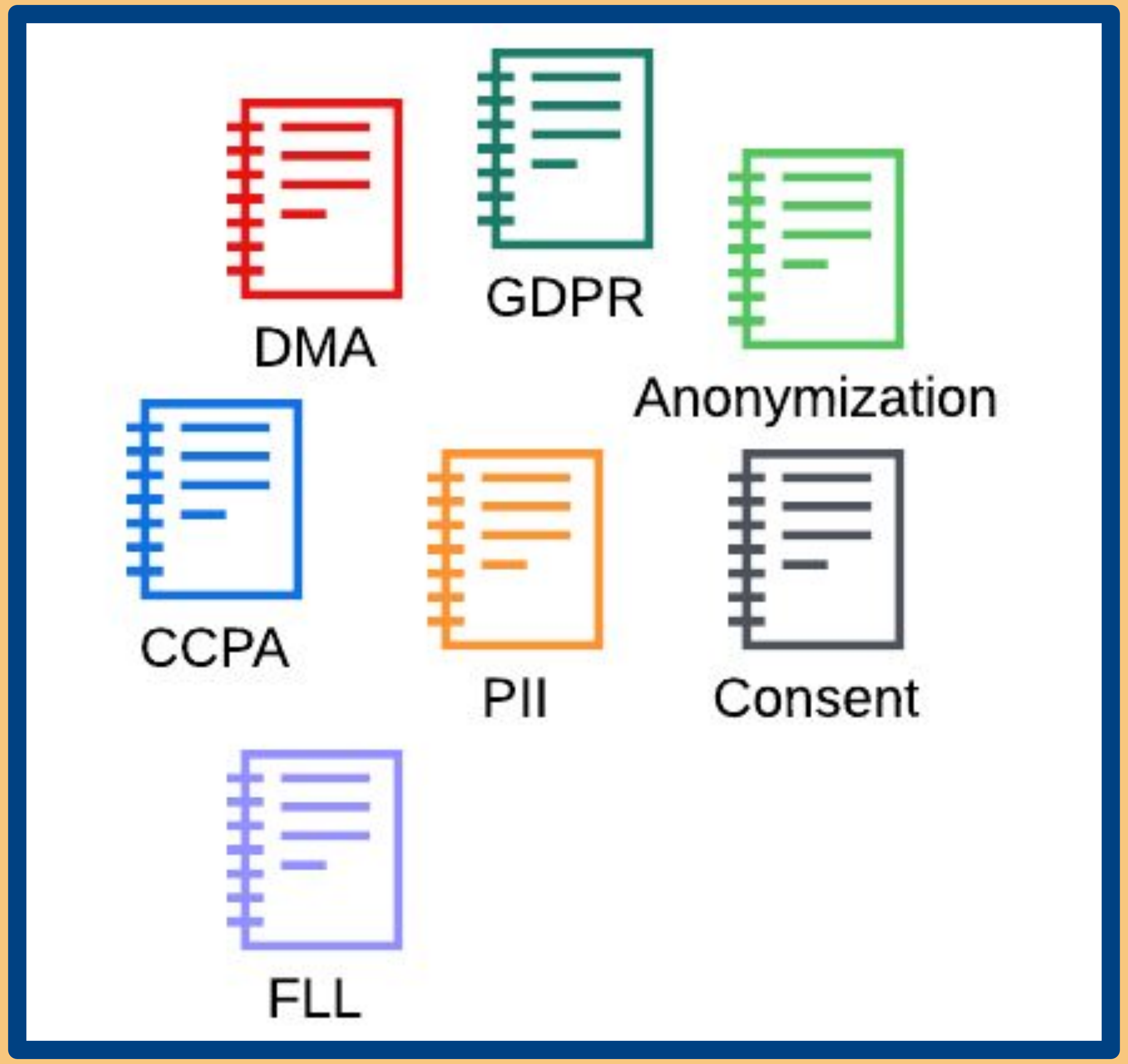
LinkedIn

Hassle-free Dynamic Policy Enforcement in Trino

Ramanathan Ramu
Pratham Desai

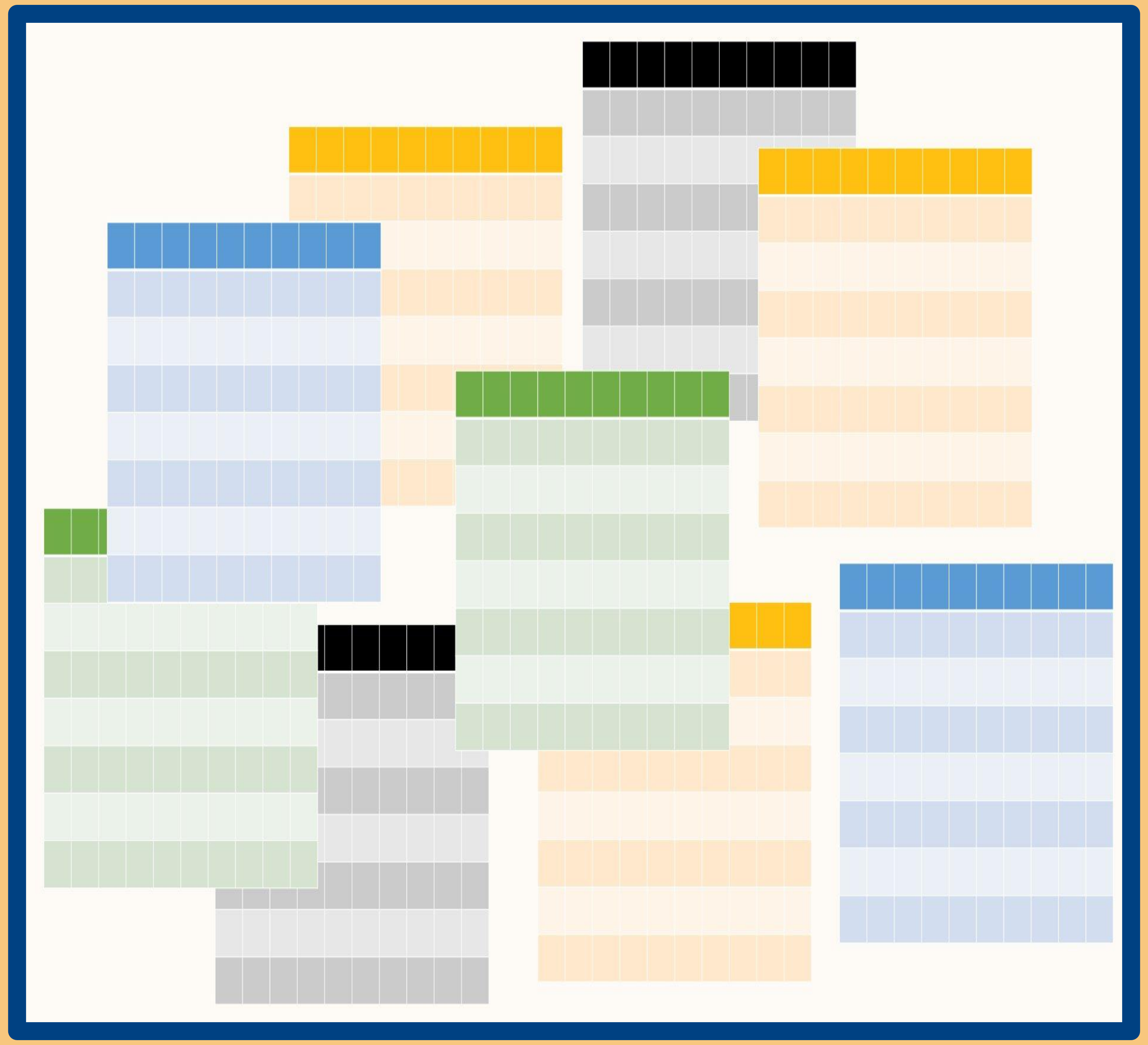
Policy Enforcement

Motivation



Too Many Policies!

+



Too Much Data!

⇒

Overwhelmed Data Engineers!

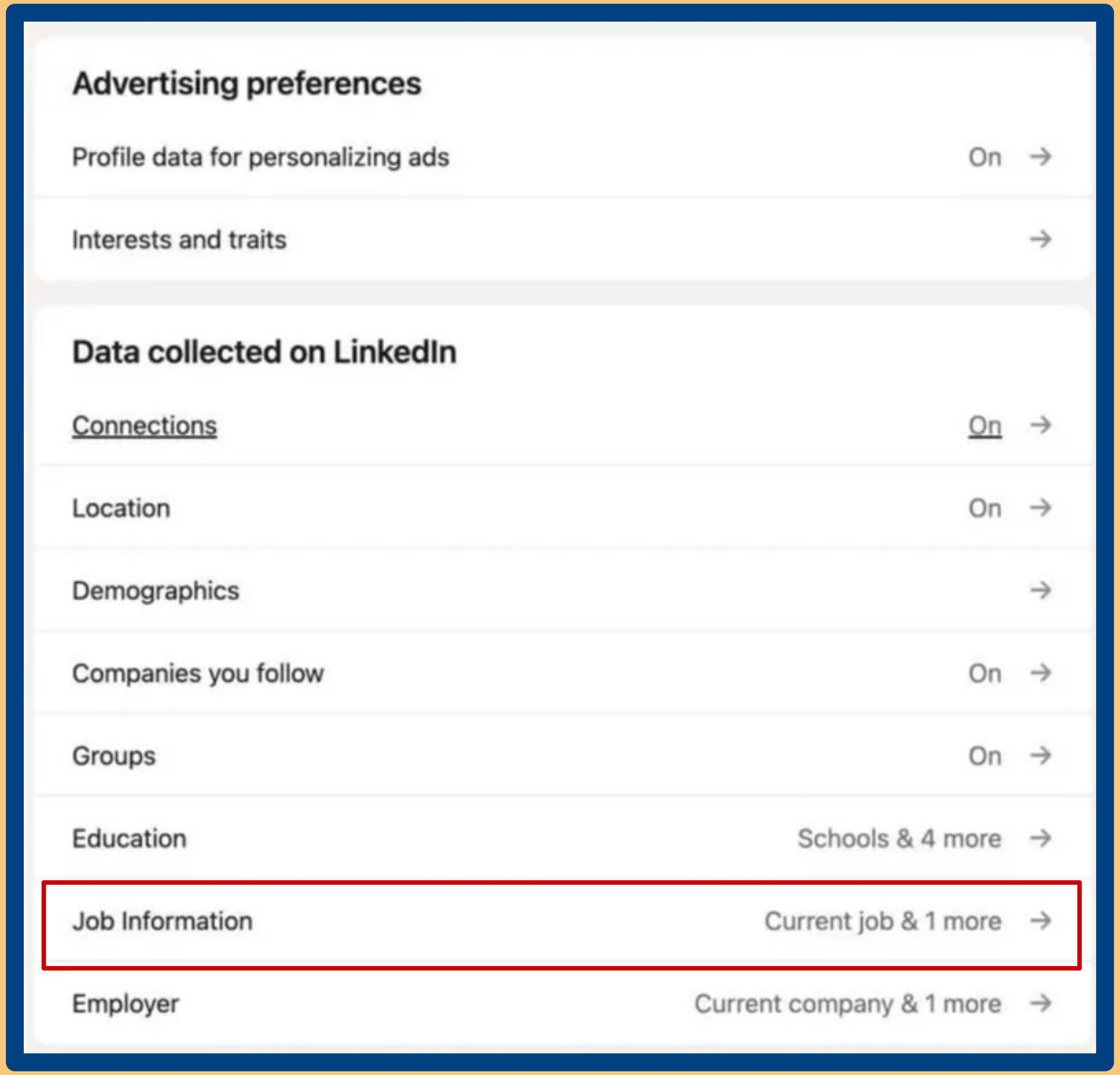
Policy Enforcement @ LinkedIn

Member Preferences

Data Collection & Labeling

Purpose Limitation

Member Preferences : Example

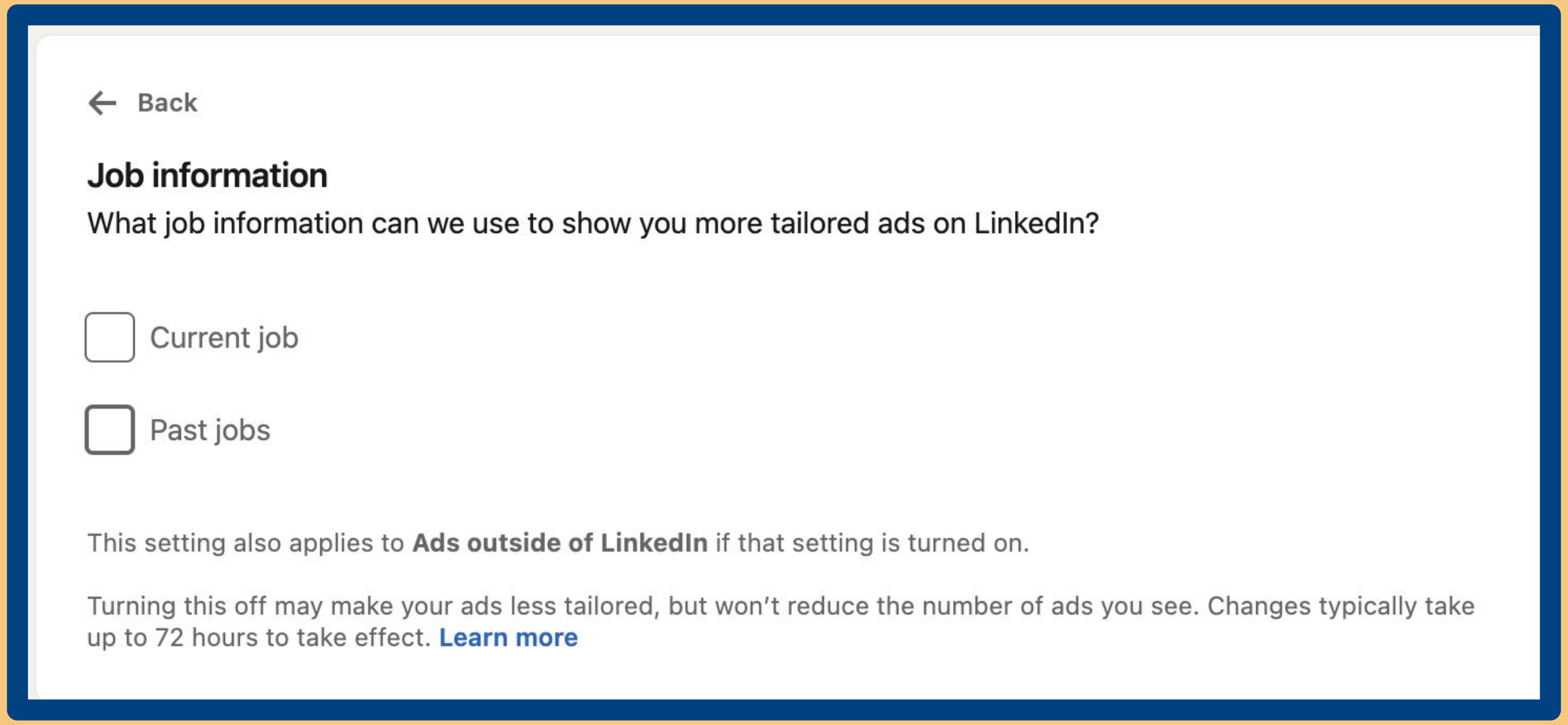


Advertising preferences

Profile data for personalizing ads	On	→
Interests and traits		→

Data collected on LinkedIn

Connections	On	→
Location	On	→
Demographics		→
Companies you follow	On	→
Groups	On	→
Education	Schools & 4 more	→
Job Information	Current job & 1 more	→
Employer	Current company & 1 more	→



← Back

Job information

What job information can we use to show you more tailored ads on LinkedIn?

- [ ] Current job
- [ ] Past jobs

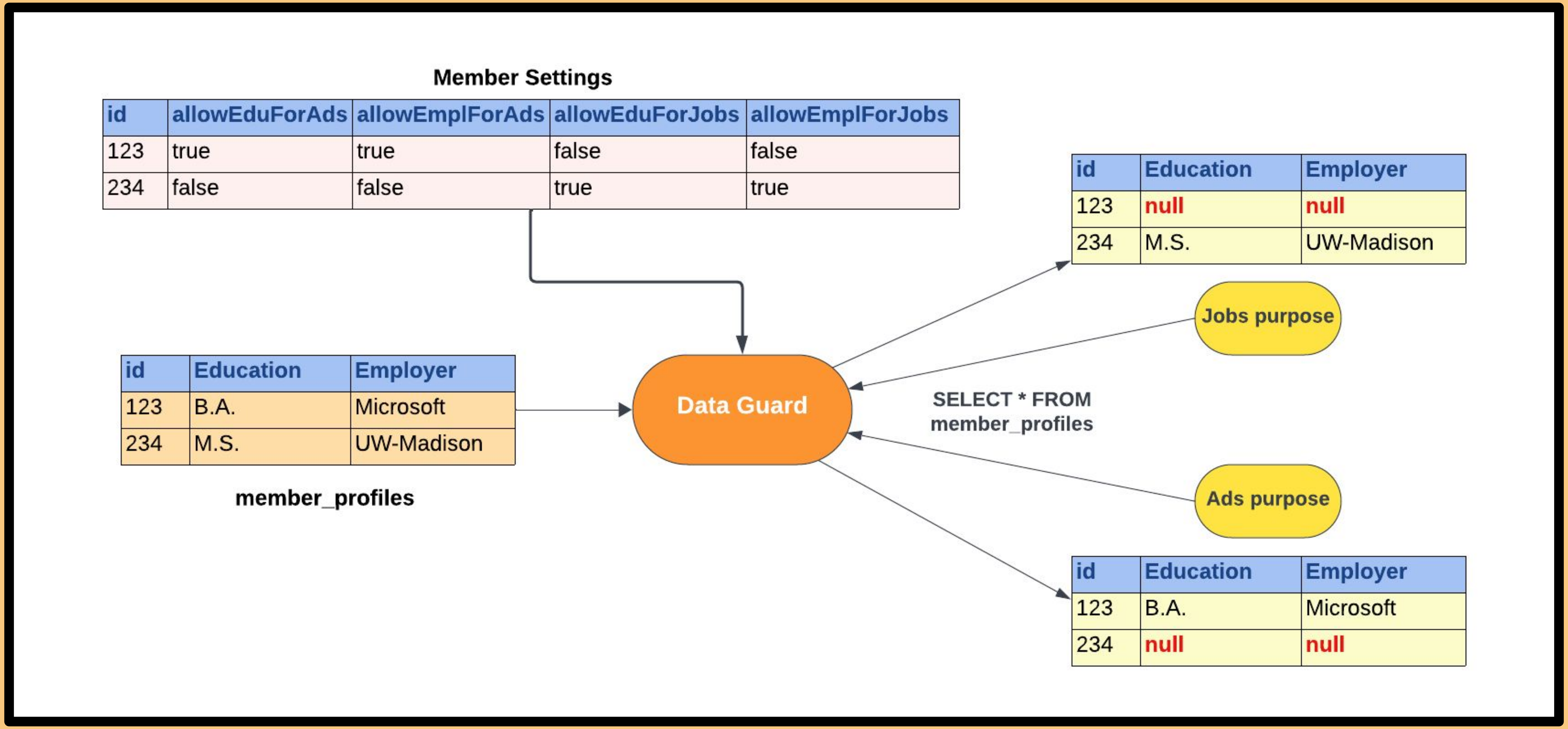
This setting also applies to **Ads outside of LinkedIn** if that setting is turned on.

Turning this off may make your ads less tailored, but won't reduce the number of ads you see. Changes typically take up to 72 hours to take effect. **Learn more**

Data Guard:
How does it work?

Policy Enforcement @ LinkedIn: Data Guard

- Data access through query engines (Trino, Spark) needs to be masked based on
  - Purpose of access
  - Data labels
  - Auxiliary data like member "preferences"



Member Settings

id	allowEduForAds	allowEmplForAds	allowEduForJobs	allowEmplForJobs
123	true	true	false	false
234	false	false	true	true

id	Education	Employer
123	B.A.	Microsoft
234	M.S.	UW-Madison

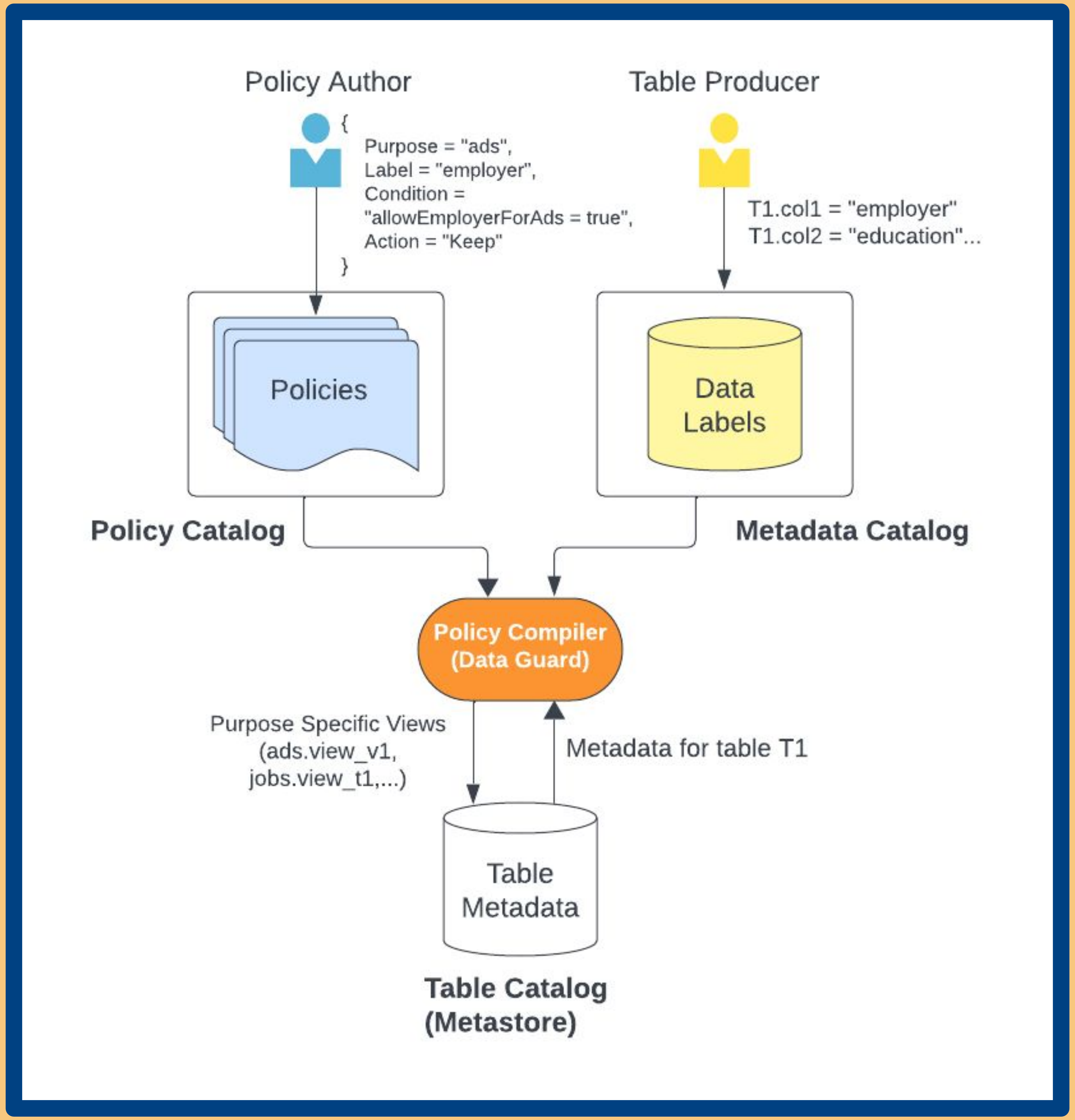
member_profiles

id	Education	Employer
123	null	null
234	M.S.	UW-Madison

id	Education	Employer
123	B.A.	Microsoft
234	null	null

Data Guard : Data Masking Views

- *Views* are the interface for policy enforcement
- Data Guard programmatically creates purpose-specific data-masking views on tables
- Views are accessible through query engines like Trino and Spark
- Data Guard compiles the View SQL on a table using the metadata catalog for “data labels” and policy catalog for “policies”
- Views are refreshed periodically with changes in policies and data labels



Data Masking Views : Example

member_profiles table

id	col1	col2
123	B.A.	Microsoft
234	M.S.	UW-Madison

Labels for member_profiles

column_name	label
id	*dataSubjectId*
col1	*education*
col2	*employer*

member_settings

id	allowEduForAds
123	true
234	false

Data labeled as "education" should not be used for "Ads" purpose if the user has not consented to it

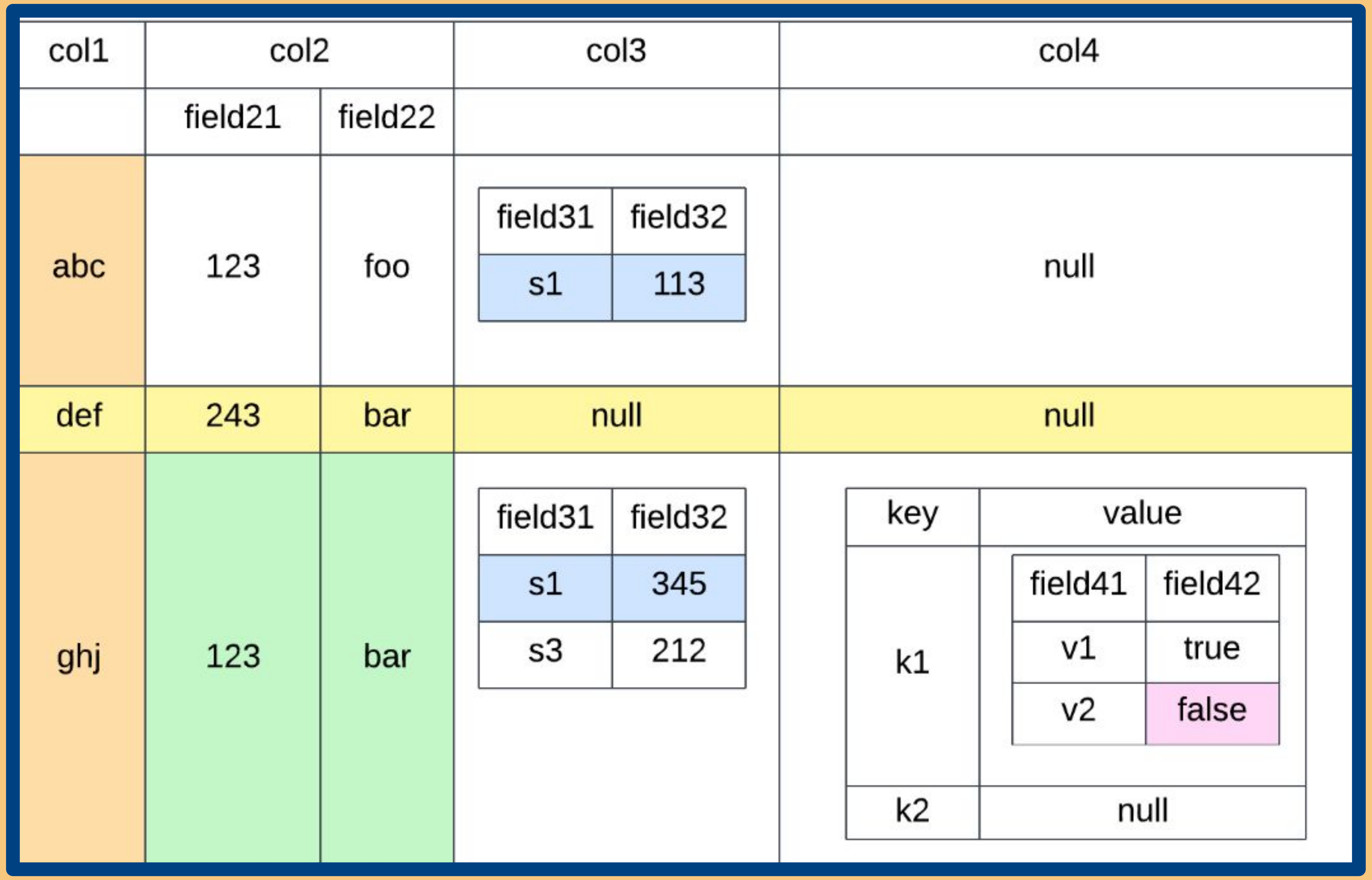
Data Guard View SQL for "Ads" Purpose

```
SELECT
    T1.id id,
    T1.col1 col1,
    CASE
      WHEN T2.allowEduForAds = true THEN T1.col2
      ELSE NULL
    END col2
FROM
     member_profiles T1 JOIN member_settings T2
ON T1.id = T2.id
```

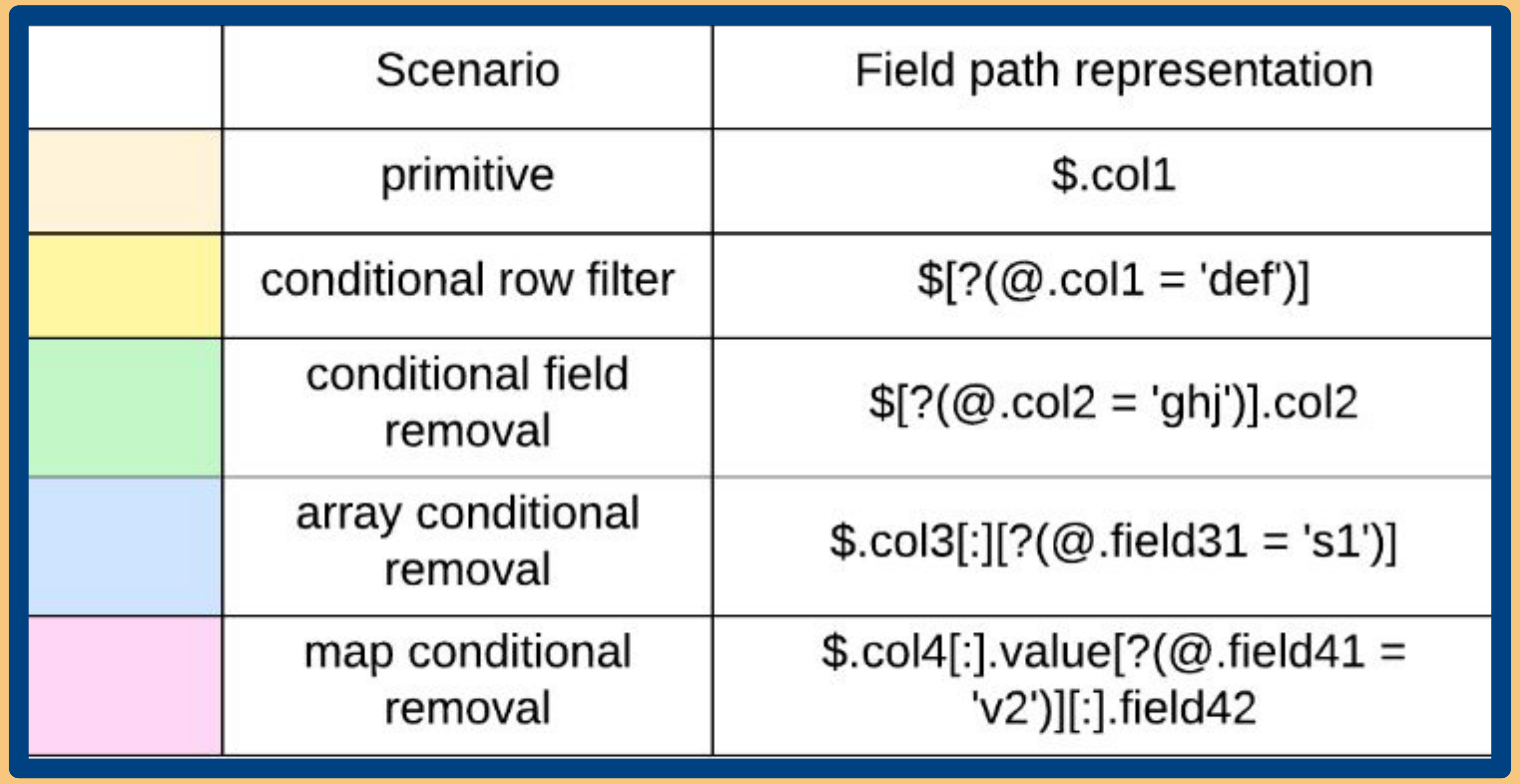
Schema preserving

Consent check

Views : Masking Granularity



col1	col2		col3	col4
	field21	field22		
abc	123	foo	field31 | field32 s1 | 113	null
def	243	bar	null	null
ghj	123	bar	field31 | field32 s1 | 345 s3 | 212	key | value k1 | field41 | field42 v1 | true v2 | false k2 | null



	Scenario	Field path representation
🟧	primitive	$.col1
🟨	conditional row filter	$[?(@.col1 = 'def')]
🟩	conditional field removal	$[?(@.col2 = 'ghj')].col2
🟦	array conditional removal	$.col3[:][?(@.field31 = 's1')]
🟪	map conditional removal	$.col4[:].value[?(@.field41 = 'v2')][:].field42

Using Views

Expressive - Express multiple policies with projections, filters, joins, UDFs.

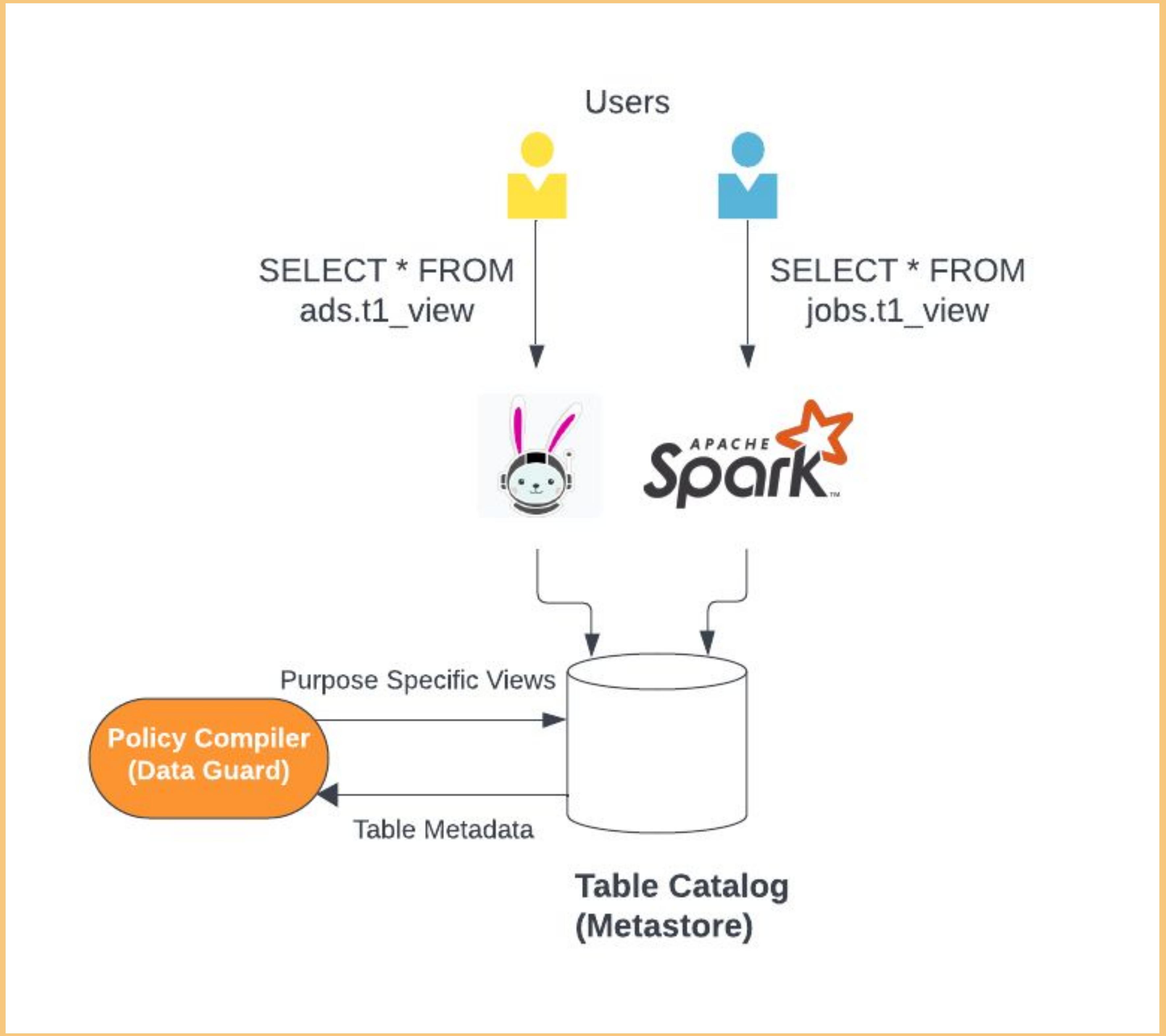
Portable - Executable on multiple engines

Modular - Can be drop-in replacement to underlying data

Agile - Roll-out new views, version, rollback to previous views

Data Guard : View Usage

- Users can access purpose-specific views through Trino and Spark


Users
SELECT * FROM ads.t1_view
SELECT * FROM jobs.t1_view
APACHE Spark
Purpose Specific Views
Policy Compiler (Data Guard)
Table Metadata
Table Catalog (Metastore)

- Next : How to roll this out to make workloads compliant?

Data Guard:
How is it rolled out?

How to roll out views?

Large Scale Migration?

- Force apps/users to apply the policy by explicitly adopting Data Guard views

Expensive & Slow

```
Select
    T1.a,
    T2.b,
    MAX(T3.c)  AS  max_c
FROM
    T1
INNER  JOIN
    T2  ON  T1.a  =  T2.c
LEFT  OUTER  JOIN
    T3  ON  T2.b  =  T3.a
GROUP  BY
    T1.a,  T2.b
ORDER  BY
    T1.a  DESC,  T2.b  ASC;
```

```
Select
    T1_DMView1.a,
    T2_DMView1.b,
    MAX(T3_DMView1.c)  AS  max_c
FROM
    T1_DMView1
INNER  JOIN
    T2_DMView1  ON  T1_DMView1.a  =  T2_DMView1.c
LEFT  OUTER  JOIN
    T3_DMView1  ON  T2_DMView1.b  =  T3_DMView1.a
GROUP  BY
    T1_DMView1.a,  T2_DMView1.b
ORDER  BY
    T1_DMView1.a  DESC,  T2_DMView1.b  ASC;
```

Expose implementation details

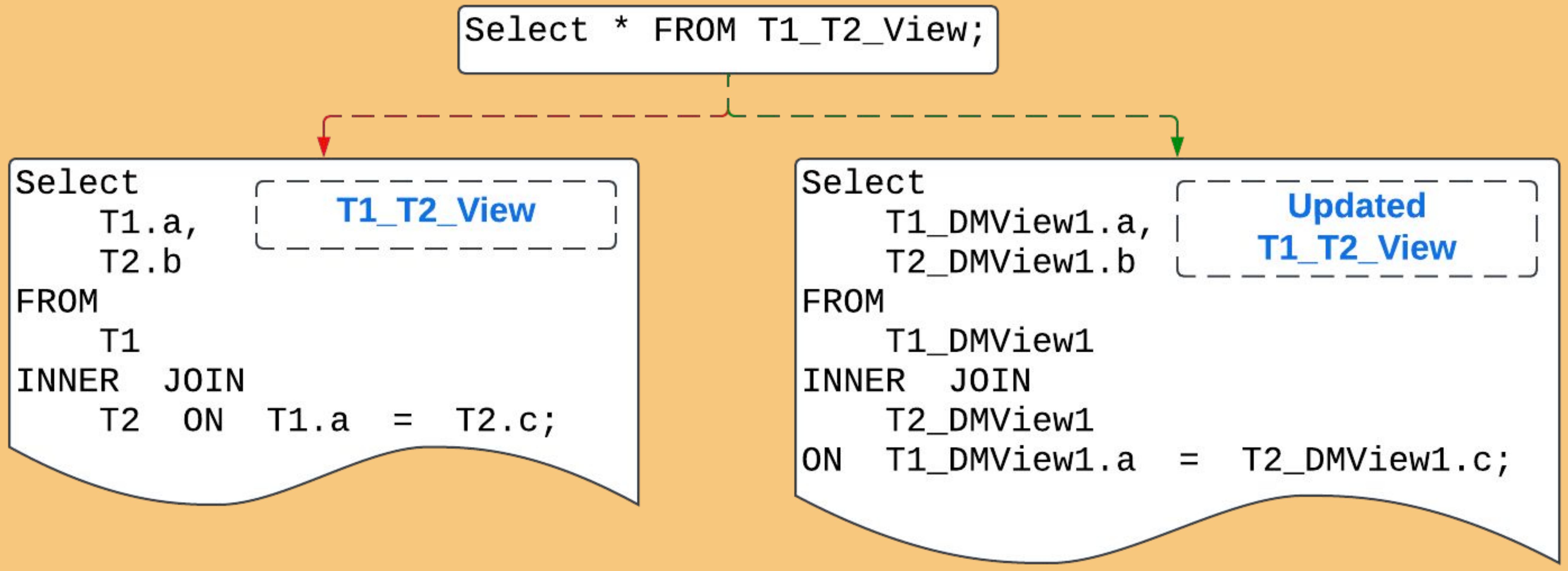
```
Select
    *
FROM
    member_profiles;
```

```
Select
    *
FROM
    ads.member_profiles_redact_pi;
```

How to roll out views?

Large Scale Migration?

- Force apps/users to apply the policy by explicitly adopting Data Guard views



```
Select * FROM T1_T2_View;
```

```
Select
    T1.a,
    T2.b
FROM
    T1
INNER  JOIN
    T2  ON  T1.a  =  T2.c;
```

```
Select
    T1_DMView1.a,
    T2_DMView1.b
FROM
    T1_DMView1
INNER  JOIN
    T2_DMView1
ON  T1_DMView1.a  =  T2_DMView1.c;
```

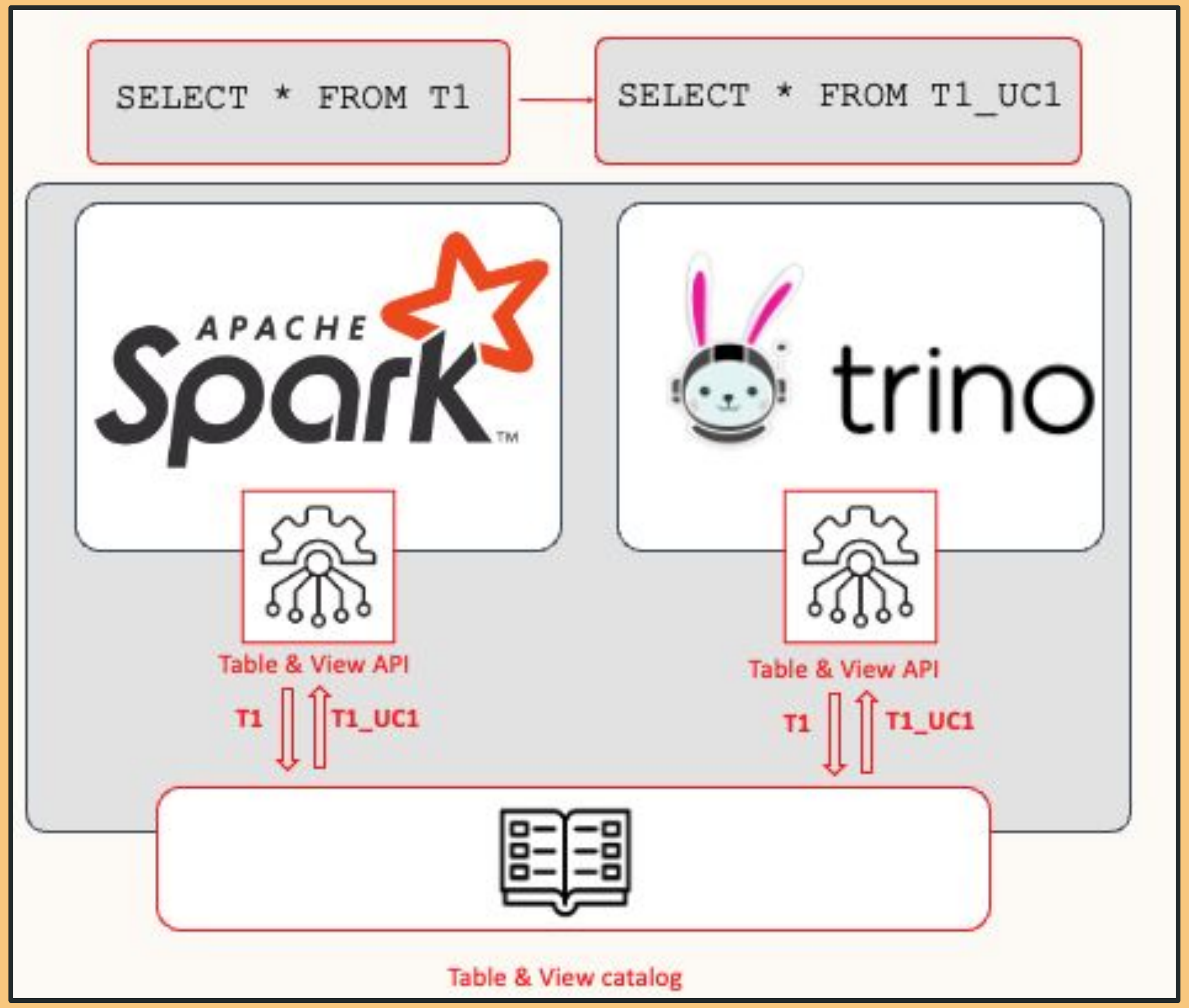
Existing Views have to be updated

Recreate/Update Existing Views

Error prone

Solution → ViewShift

Dynamically route tables to views at runtime!



Transparently replaces table access with views

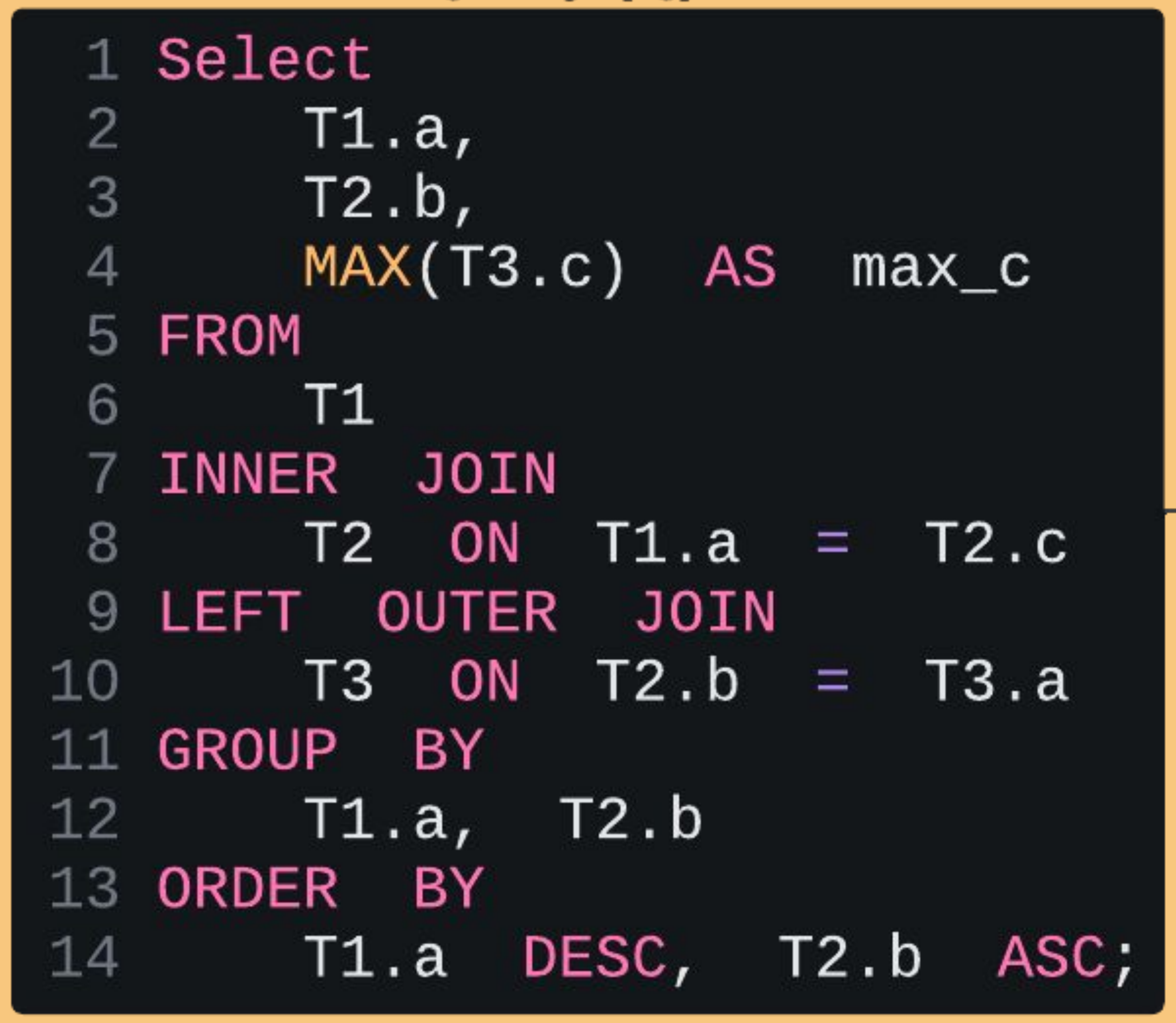
Familiar dataset names
Does Not expose policy details

Works for future regulation

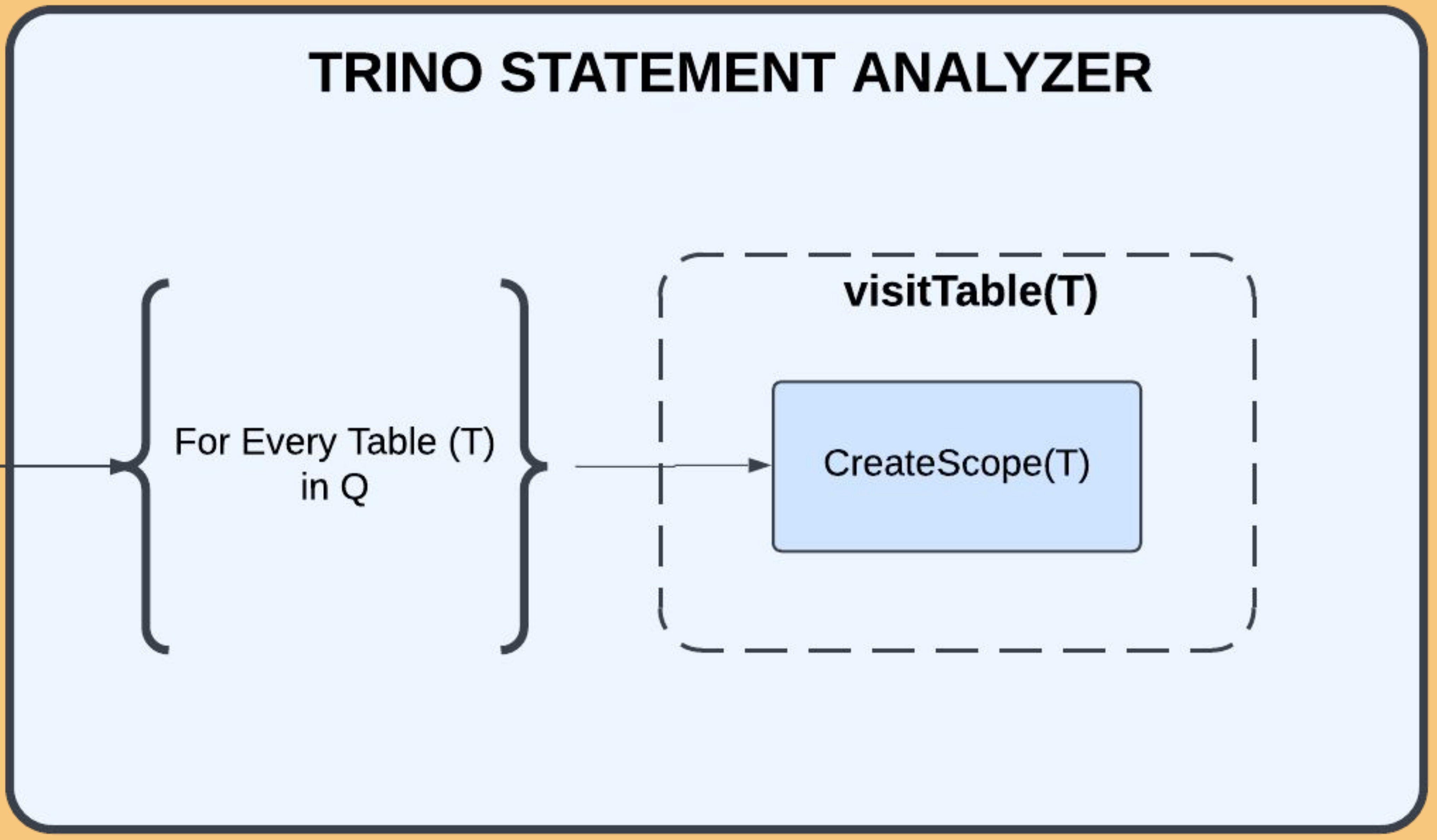
Cross engine compatibility

Trino without ViewShift

Query (Q)



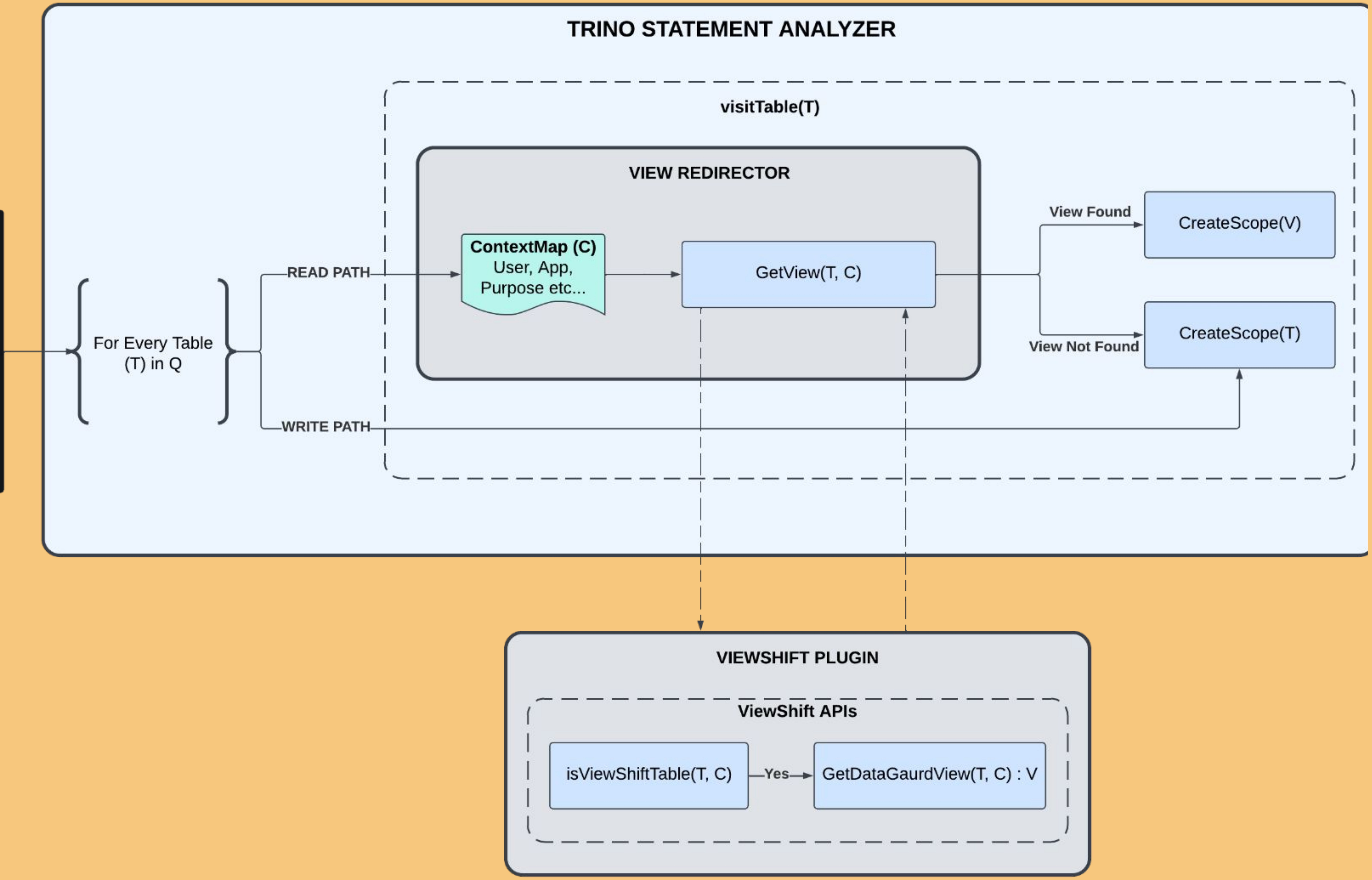
```
Select
    T1.a,
    T2.b,
    MAX(T3.c)  AS  max_c
FROM
    T1
INNER  JOIN
    T2  ON  T1.a  =  T2.c
LEFT  OUTER  JOIN
    T3  ON  T2.b  =  T3.a
GROUP  BY
    T1.a,  T2.b
ORDER  BY
    T1.a  DESC,  T2.b  ASC;
```



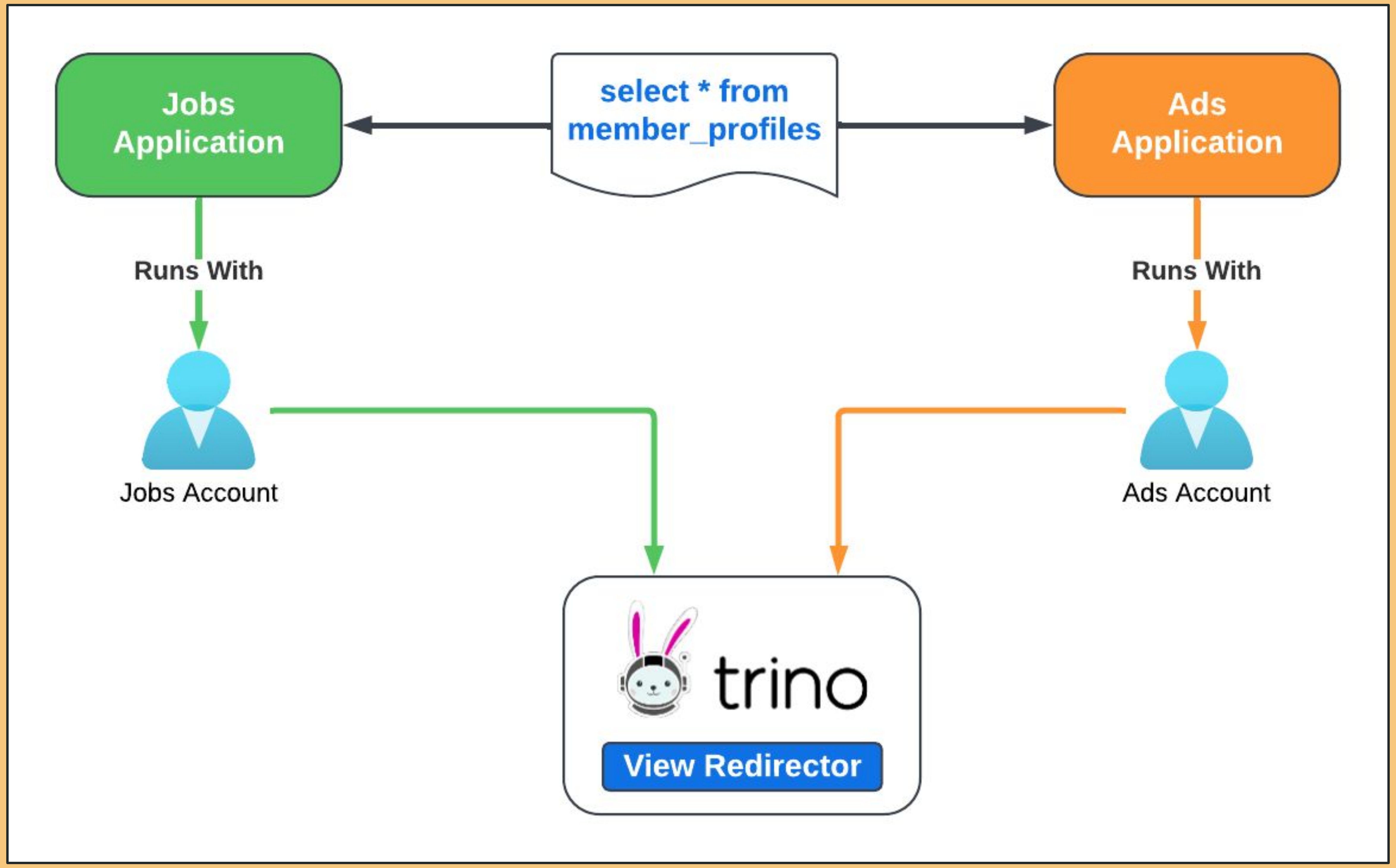
Trino with ViewShift

Query (Q)

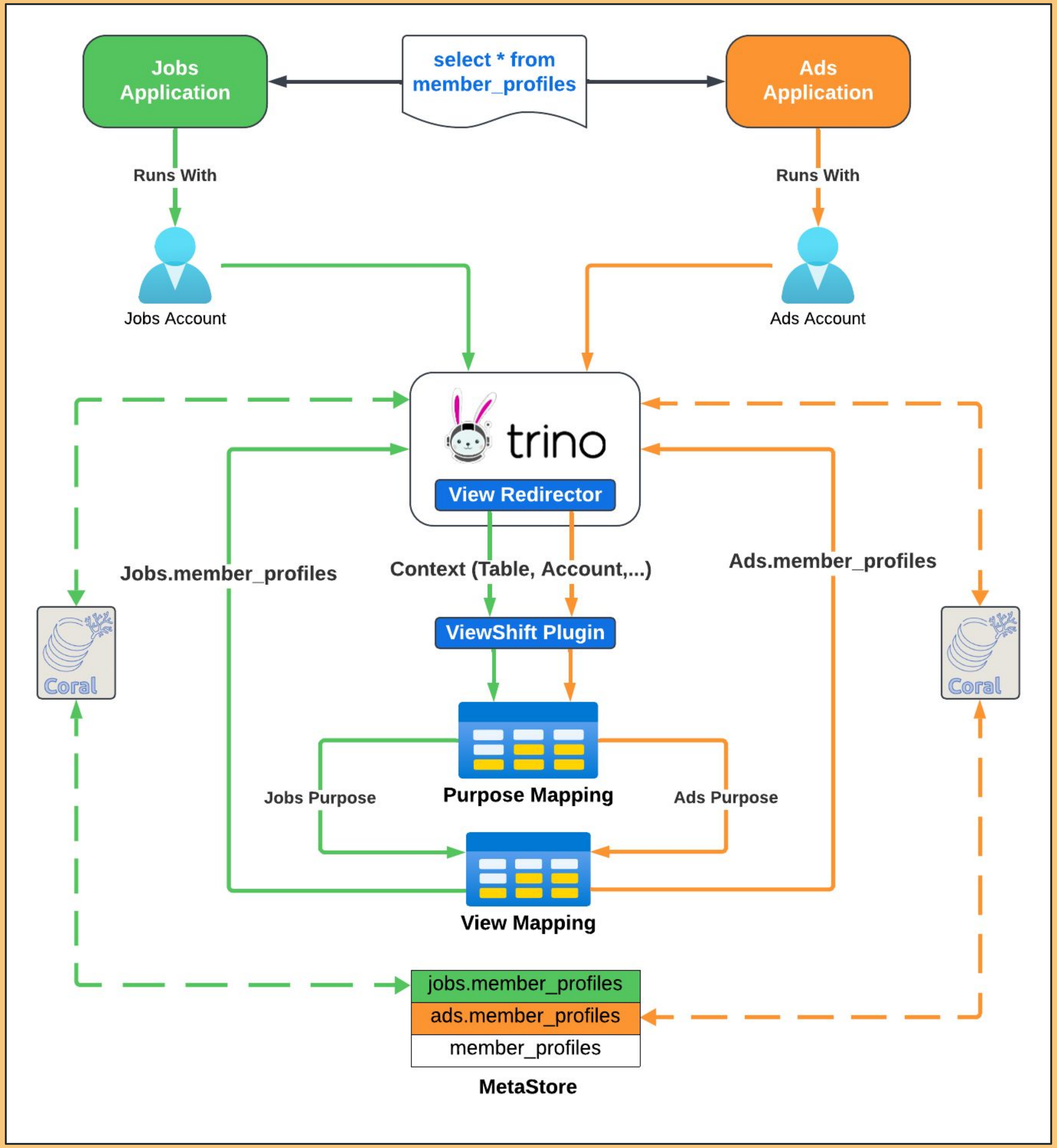
```
Select
    T1.a,
    T2.b,
    MAX(T3.c)  AS  max_c
FROM
    T1
INNER  JOIN
    T2  ON  T1.a  =  T2.c
LEFT  OUTER  JOIN
    T3  ON  T2.b  =  T3.a
GROUP  BY
    T1.a,  T2.b
ORDER  BY
    T1.a  DESC,  T2.b  ASC;
```



Example : Trino Query With ViewShift



Example : Trino Query With ViewShift



Other Approaches

- *Row Filter/Column Masking*
- *View + Table Redirection*

Future Work

- ***Open Source***
  - *Goal : More Generic and OSS friendly*
  - *Approach : Extend Table Redirection With ViewShift APIs*

Thank you