TECHNISCHE
UNIVERSITÄT
DARMSTADT

Towards Sloppy SQL Queries Over Tabular Data Lakes

Jan-Micha Bodensohn, Jakob Steinke and Carsten Binnig

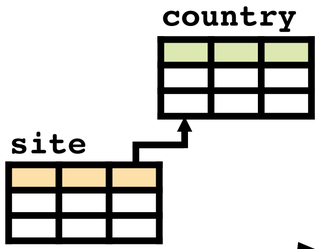
Technical University of Darmstadt
Data and AI Systems Lab

Databases vs. Data Lakes

Relational Databases

✅ **Querying is easy**

```
SELECT country.name, SUM(site.revenue)
FROM country
JOIN site ON country.id = site.cid
WHERE country.currency = 'EUR'
GROUP BY country.name
ORDER BY SUM(site.revenue)
```



Data Integration Overhead:
- Schema alignment
- Duplicate detection
- Data cleaning

❌ **Data ingestion is hard**

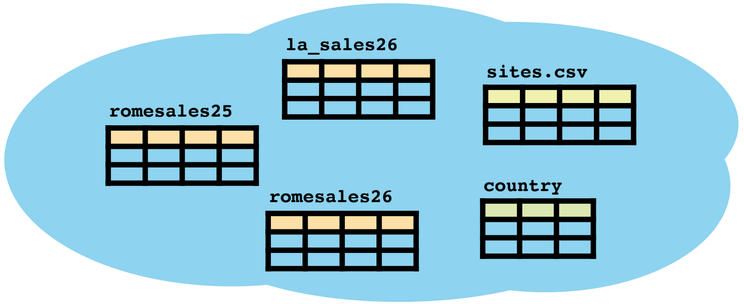
Data Lakes

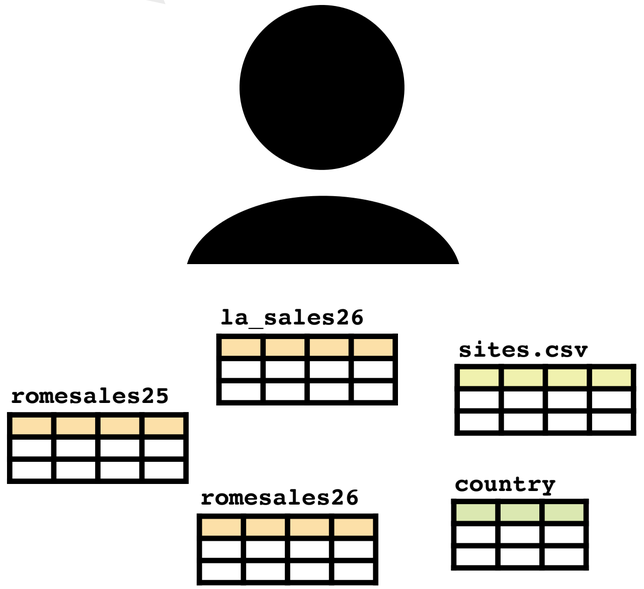
❌ **Querying is hard**

???

What data is in the data lake?

How is it organized in tables?





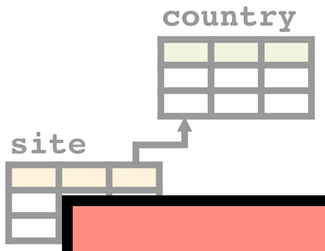
✅ **Data ingestion is easy**

Databases vs. Data Lakes

Relational Databases

✅ **Querying is easy**

```
SELECT country.name, SUM(site.revenue)
FROM country
JOIN site ON country.id = site.cid
WHERE country.currency = 'EUR'
GROUP BY country.name
ORDER BY SUM(site.revenue)
```



Data Integration C
- Schema alignm
- Duplicate detec
- Data cleaning

❌ **Data ingestion is hard**

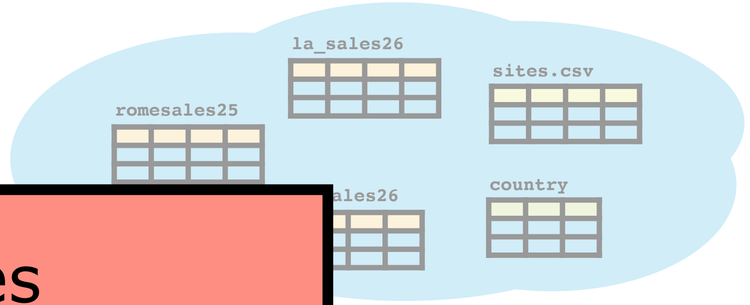
Data Lakes

❌ **Querying is hard**

???

What data is in the data lake?

How is it organized in tables?

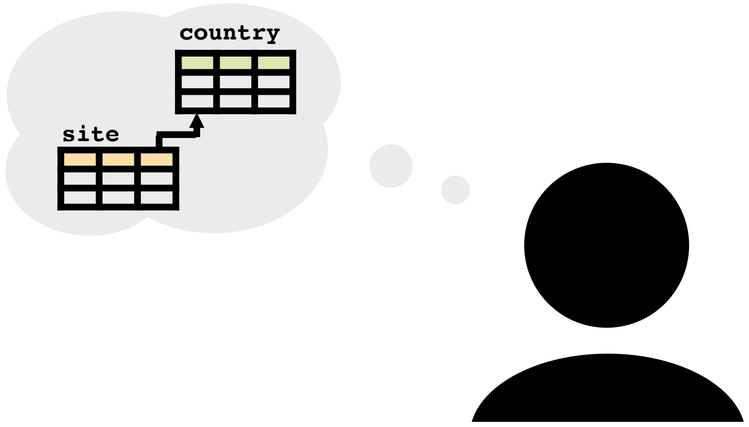


✅ **Data ingestion is easy**

How can we make data lakes
as easy to query as relational databases?

Querying Data Lakes with Sloppy SQL

User writes sloppy SQL query by assuming an idealized schema for the query




```
SELECT country.name, SUM(site.revenue)
FROM country
JOIN site ON country.id = site.cid
WHERE country.currency = 'EUR'
GROUP BY country.name
ORDER BY SUM(site.revenue)
```

Why SQL (and not natural language)?

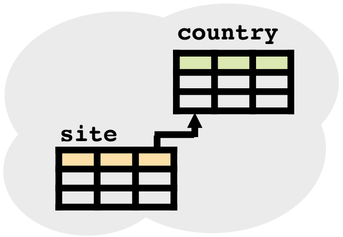
Natural language is **ambiguous** + Data lakes are **messy** = ☹

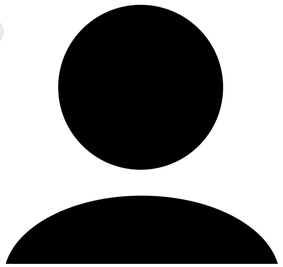
SQL enables users to **specify precisely** what they want.

What is the revenue of sites in Euro countries grouped by country name, ordered by revenue, excluding any countries without sites, and any sites where revenue is unknown?

Querying Data Lakes with Sloppy SQL

User writes sloppy SQL query by assuming an idealized schema for the query

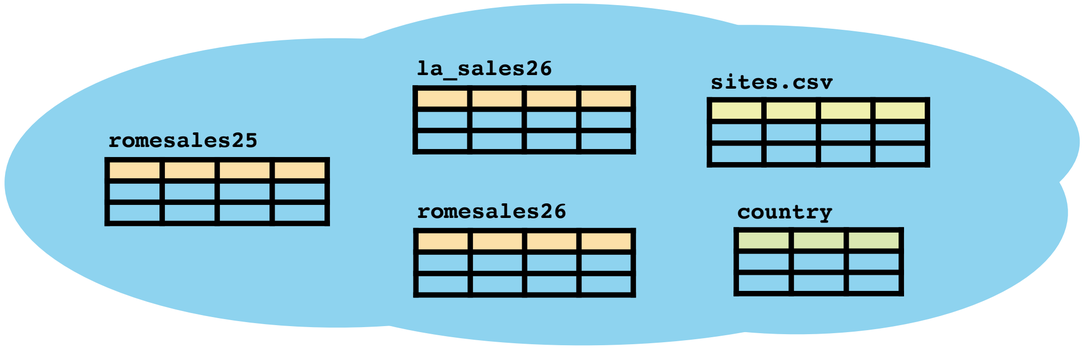




```
SELECT country.name, SUM(site.revenue)
FROM country
JOIN site ON country.id = site.cid
WHERE country.currency = 'EUR'
GROUP BY country.name
ORDER BY SUM(site.revenue)
```

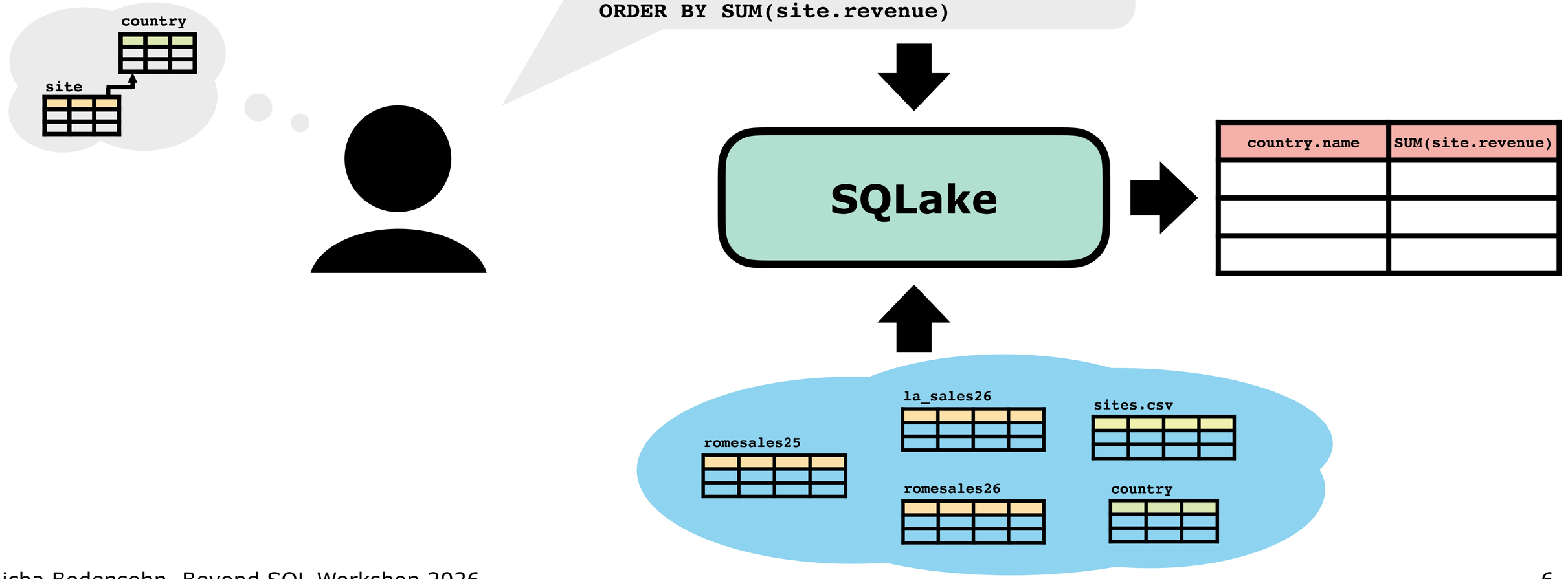
1. Schema Mismatch
- Revenue scattered across sales tables
- Additional join between sites & sales
- Incorrect table/attribute names

2. Data & Computation Mismatch
- Site revenue not in data, must be computed
- Noise: EUR vs. €, data types, etc.



Executing Sloppy SQL with SQLake

User writes sloppy SQL query by assuming an idealized schema for the query



Executing Sloppy SQL with SQLake

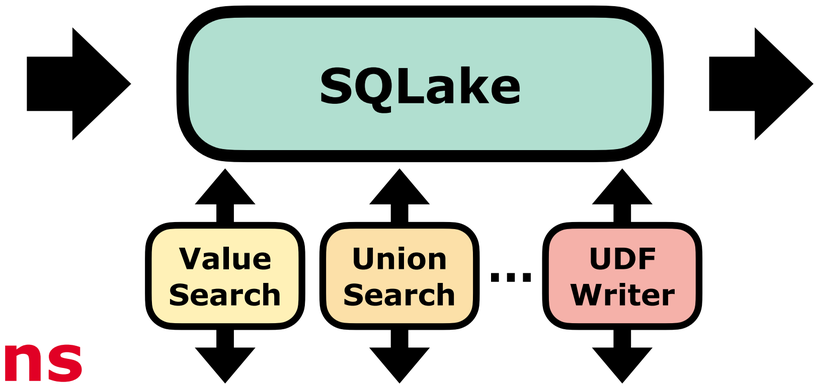
Key idea: Rewrite user query to run on data lake tables

Instead of preparing the data for the query, we adapt the query to the data!

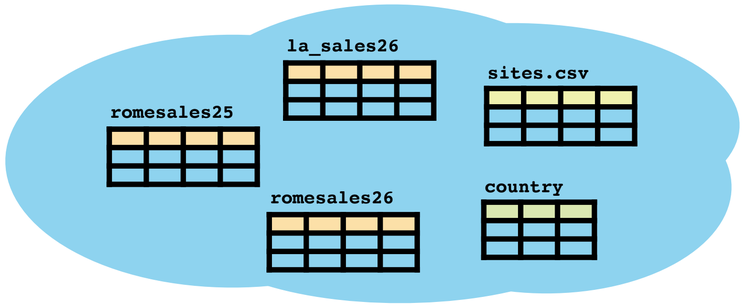
→ **Semantic Query Rewriting**

- Still captures user intent
- Executable on data lake tables

```
SELECT country.name, SUM(site.revenue)
FROM country
JOIN site ON country.id = site.cid
WHERE country.currency = 'EUR'
GROUP BY country.name
ORDER BY SUM(site.revenue)
```



Data Lake Actions

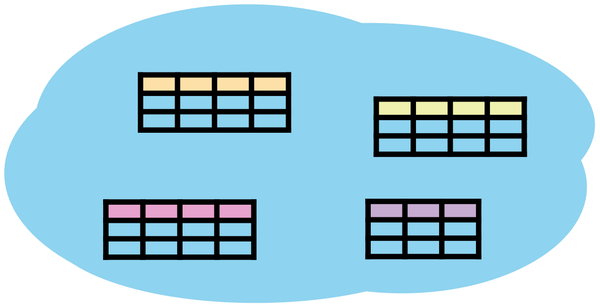



```
SELECT country.name, SUM(sales.revenue)
FROM country
JOIN site ON country.id = site.cid
JOIN (
    SELECT rev AS revenue, site_id AS site
    FROM rome_sales25
    UNION ALL
    SELECT revenue AS revenue, site AS site
    FROM rome_sales26
    UNION ALL
    …
) AS sales ON site.id = sales.site
WHERE country.currency = 'EUR'
GROUP BY country.name
ORDER BY SUM(sales.revenue)
```

Example Walkthrough

```
SELECT country.name, SUM(site.revenue)
FROM country
JOIN site ON country.id = site.cid
WHERE country.currency = 'EUR'
GROUP BY country.name
ORDER BY SUM(site.revenue)
```

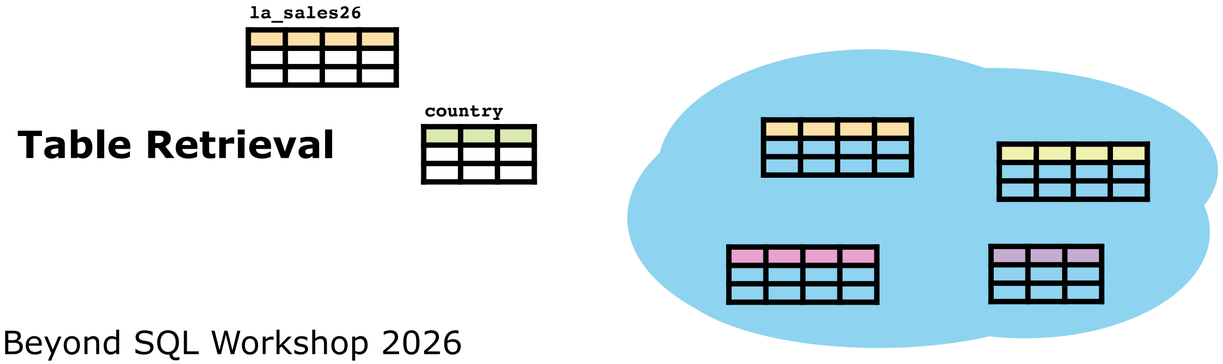
Sloppy SQL Query



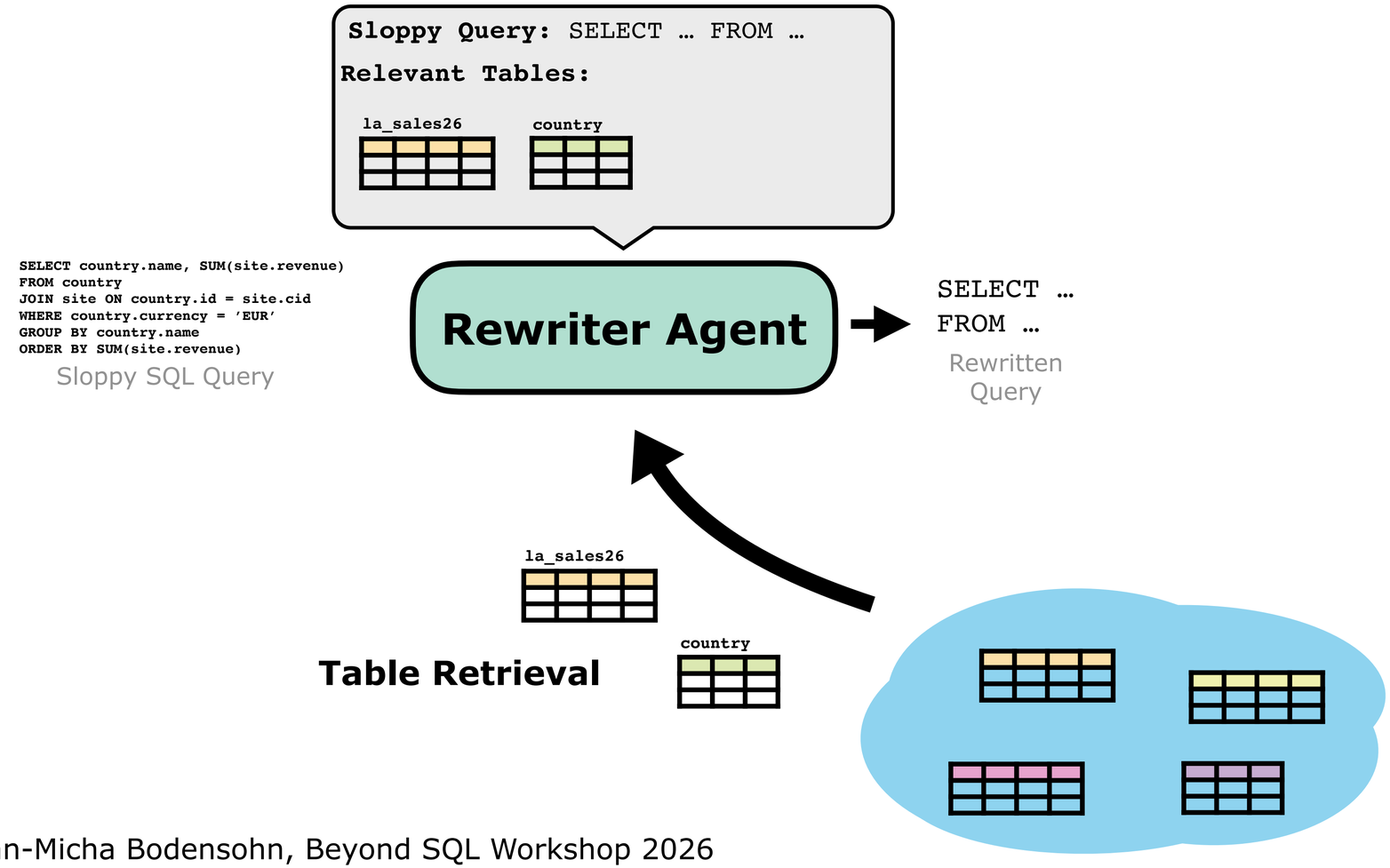
Step 1: Table Retrieval

```
SELECT country.name, SUM(site.revenue)
FROM country
JOIN site ON country.id = site.cid
WHERE country.currency = 'EUR'
GROUP BY country.name
ORDER BY SUM(site.revenue)
```

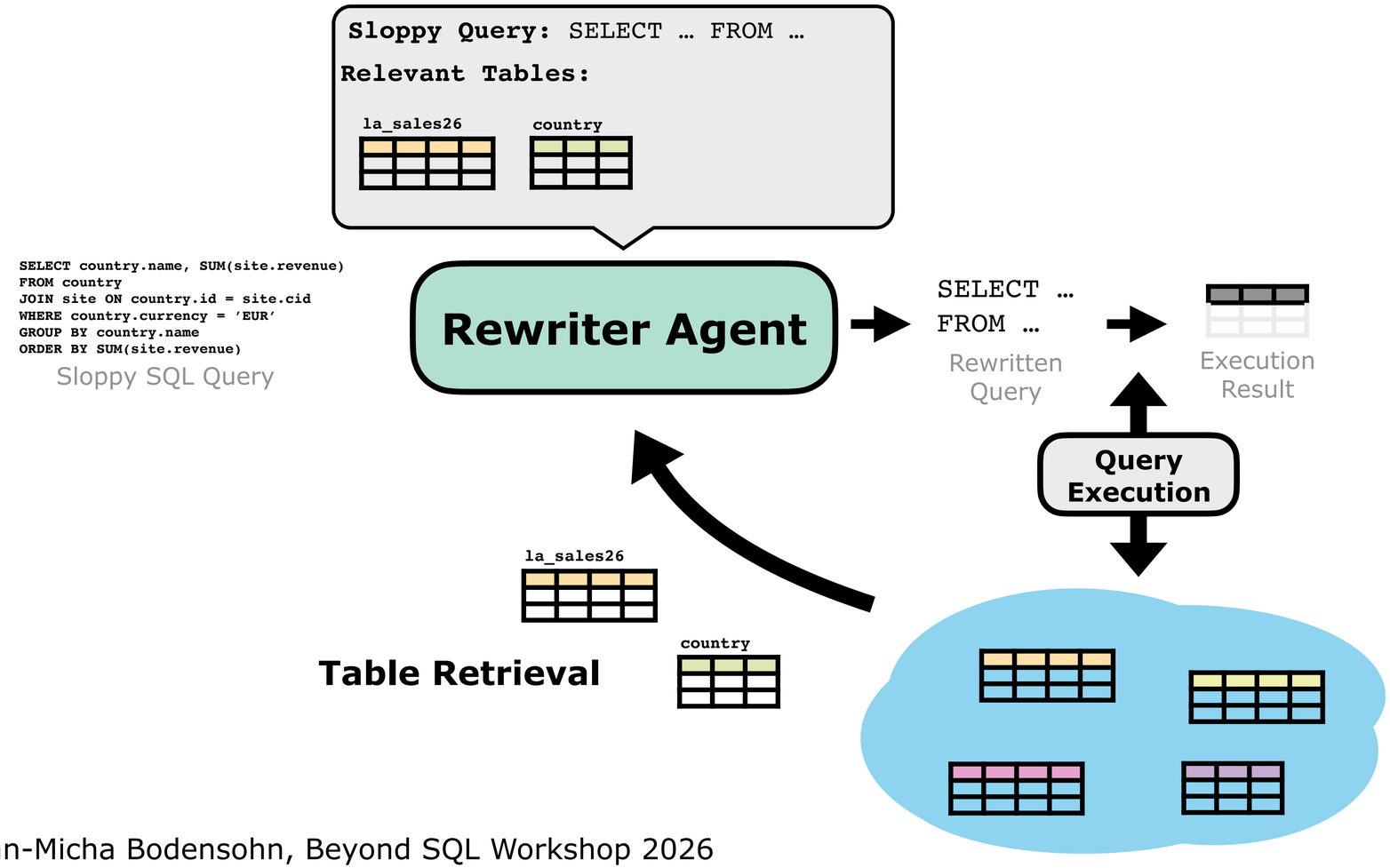
Sloppy SQL Query



Step 2: Rewriter Agent

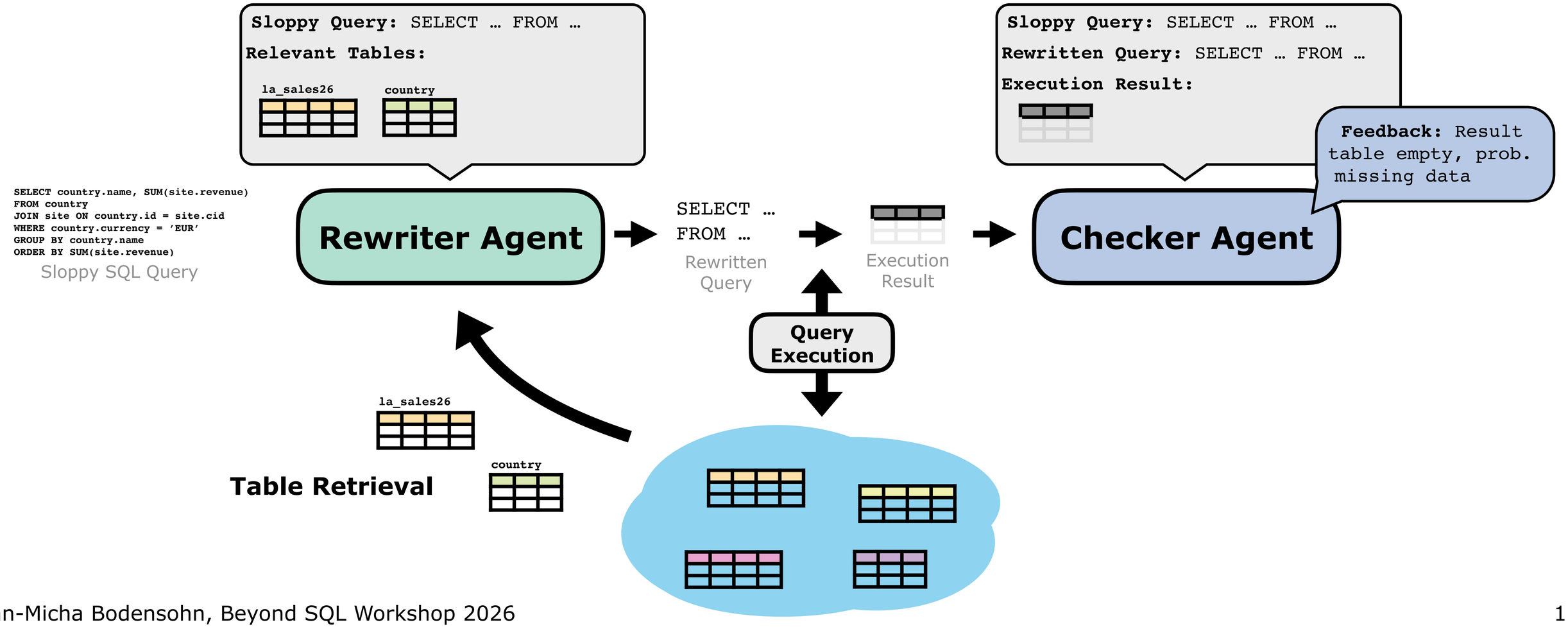


Step 3: Query Execution

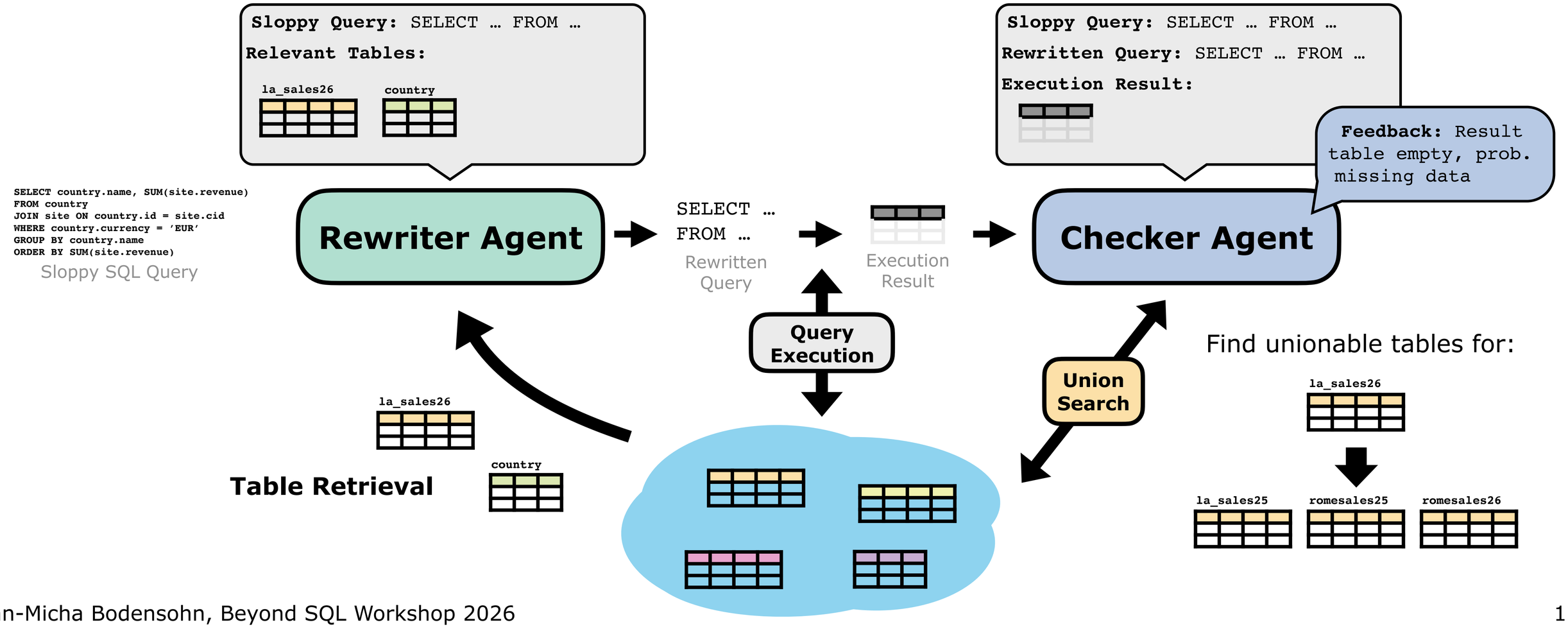


Empty execution result!

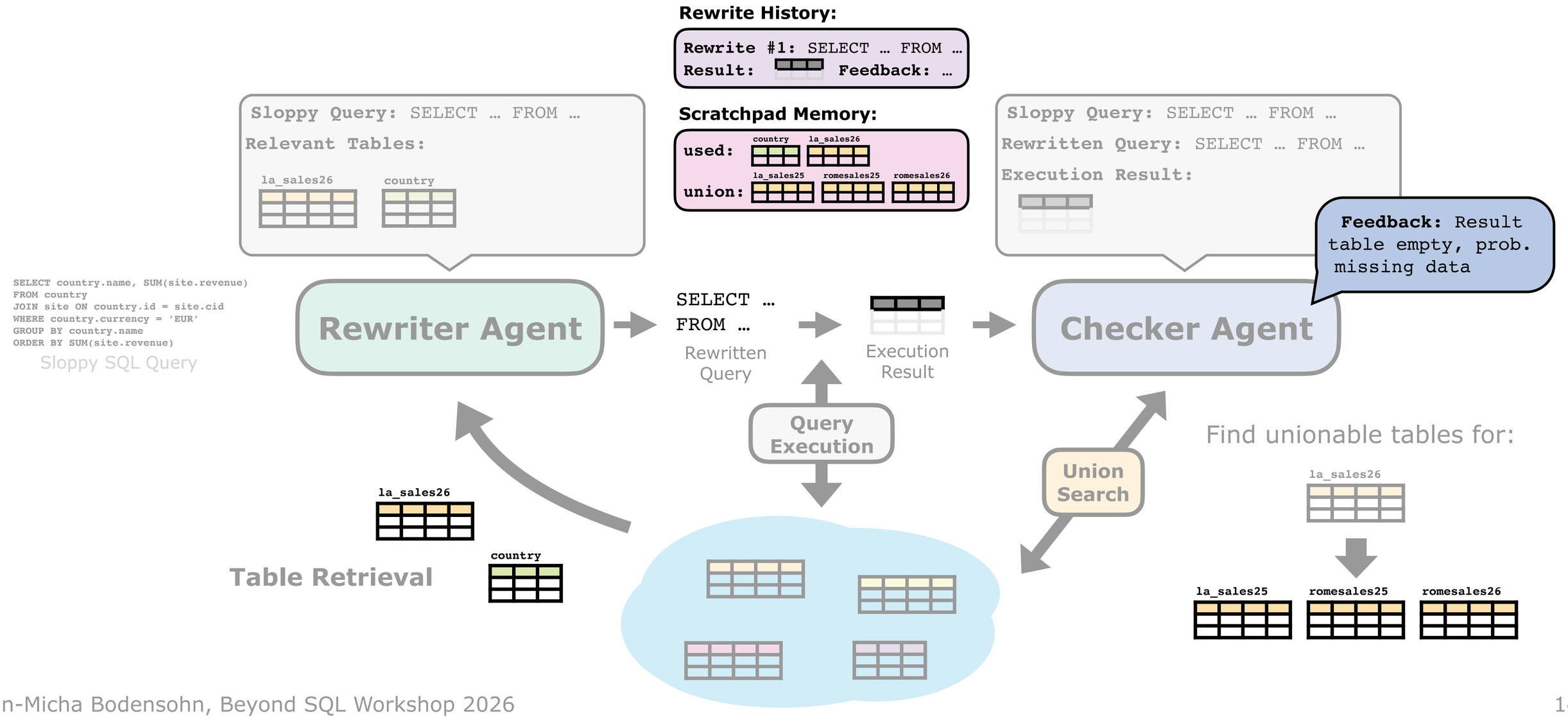
Step 4: Checker Agent



Union Search Action



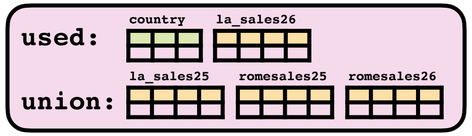
Context Management



Start again

Rewrite History:

Scratchpad Memory:

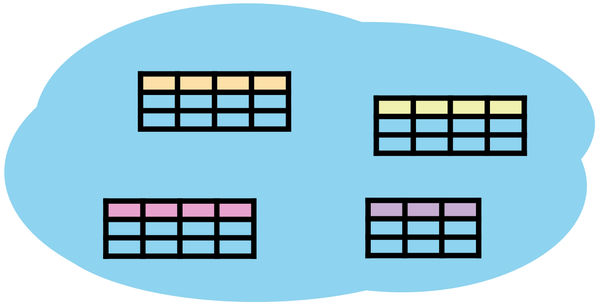



```
SELECT country.name, SUM(site.revenue)
FROM country
JOIN site ON country.id = site.cid
WHERE country.currency = 'EUR'
GROUP BY country.name
ORDER BY SUM(site.revenue)
```

Sloppy SQL Query

Rewriter Agent

Checker Agent



Start again: Table Retrieval

Rewrite History:

Rewrite #1: SELECT … FROM …
Result: **Feedback:** …

Scratchpad Memory:

used: country la_sales26

union: la_sales25 romesales25 romesales26

```
SELECT country.name, SUM(site.revenue)
FROM country
JOIN site ON country.id = site.cid
WHERE country.currency = 'EUR'
GROUP BY country.name
ORDER BY SUM(site.revenue)
```

Sloppy SQL Query

Rewriter Agent

Checker Agent

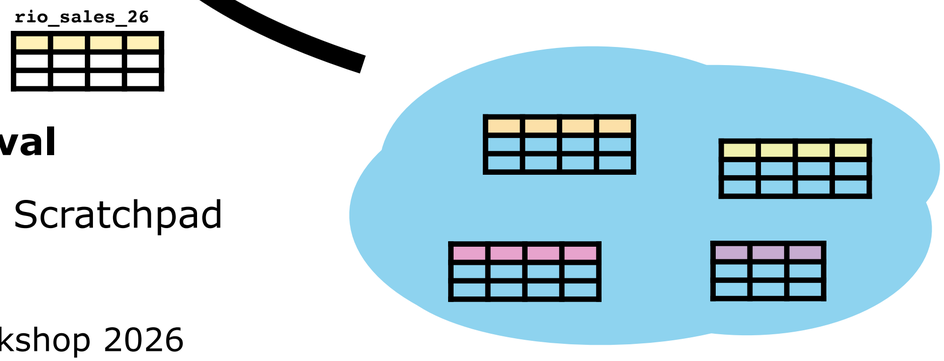
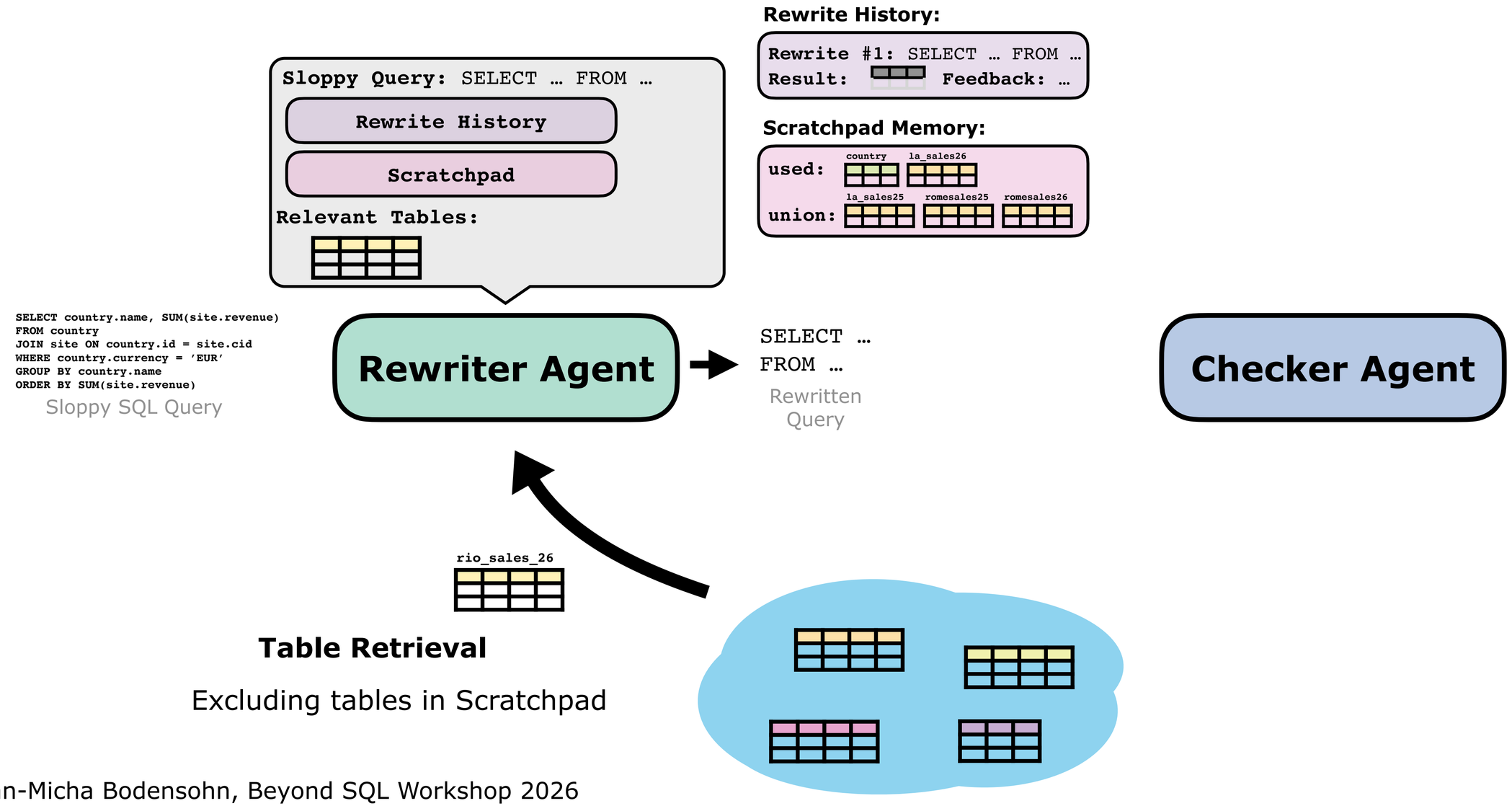


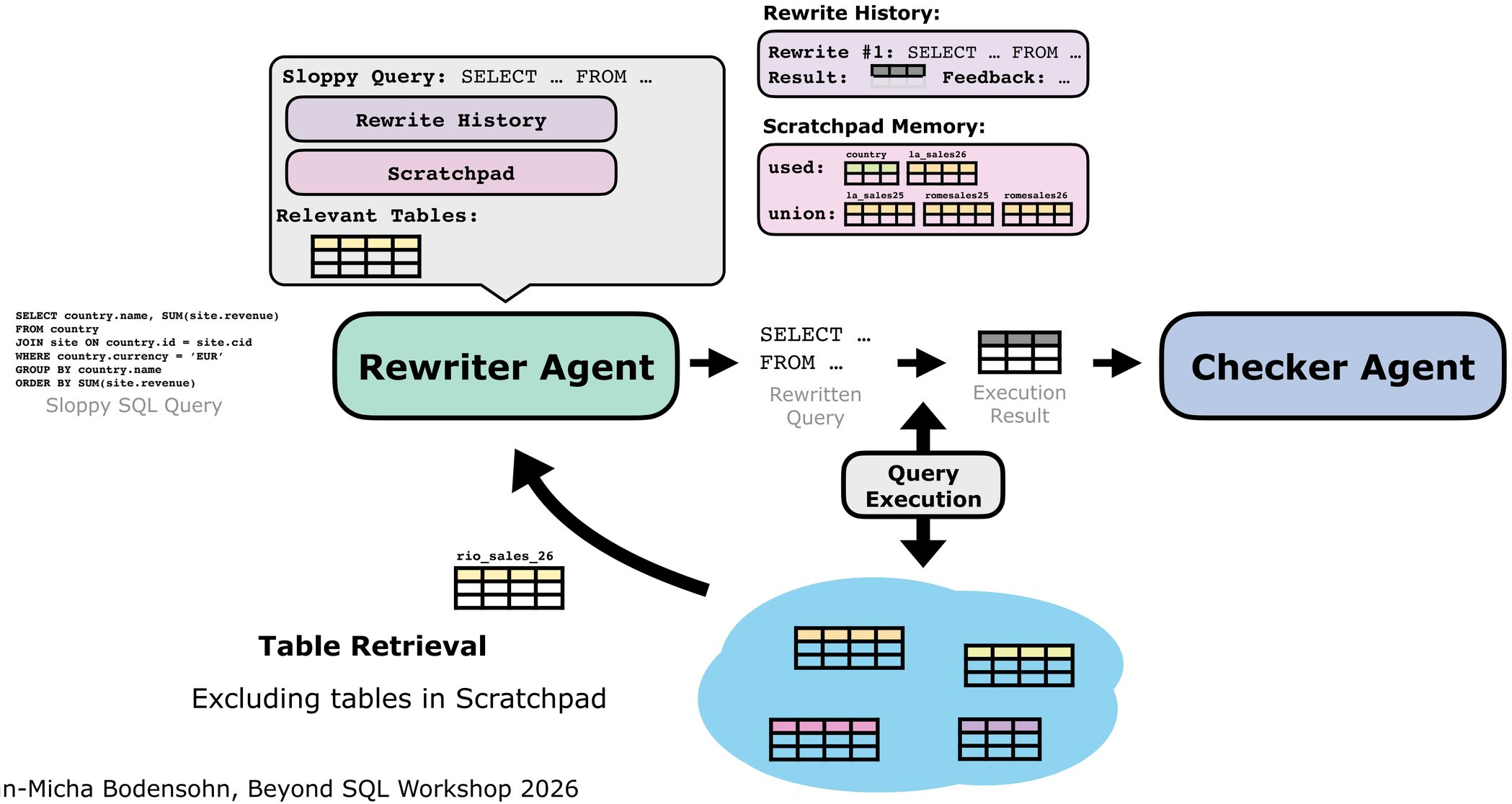
Table Retrieval

Excluding tables in Scratchpad

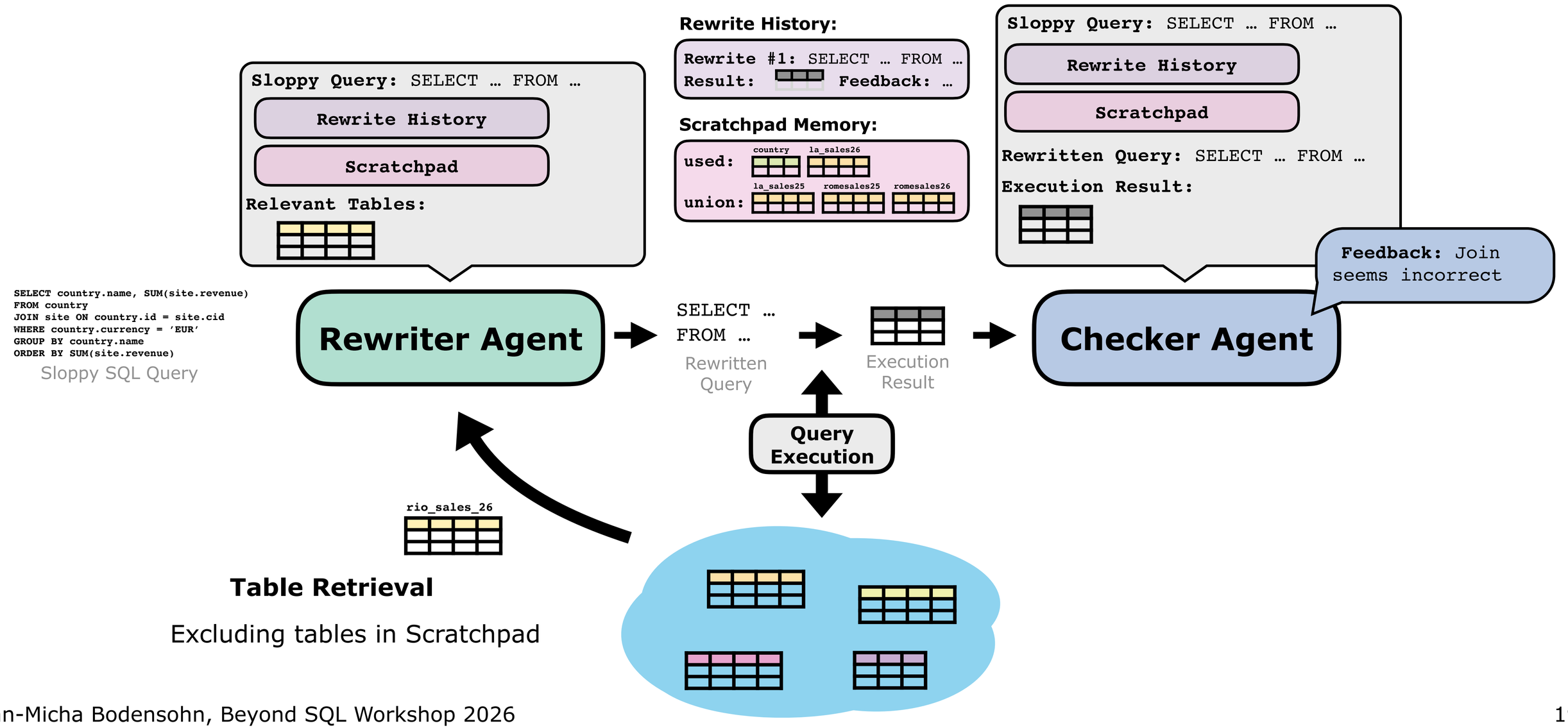
Step 2: Rewriter Agent



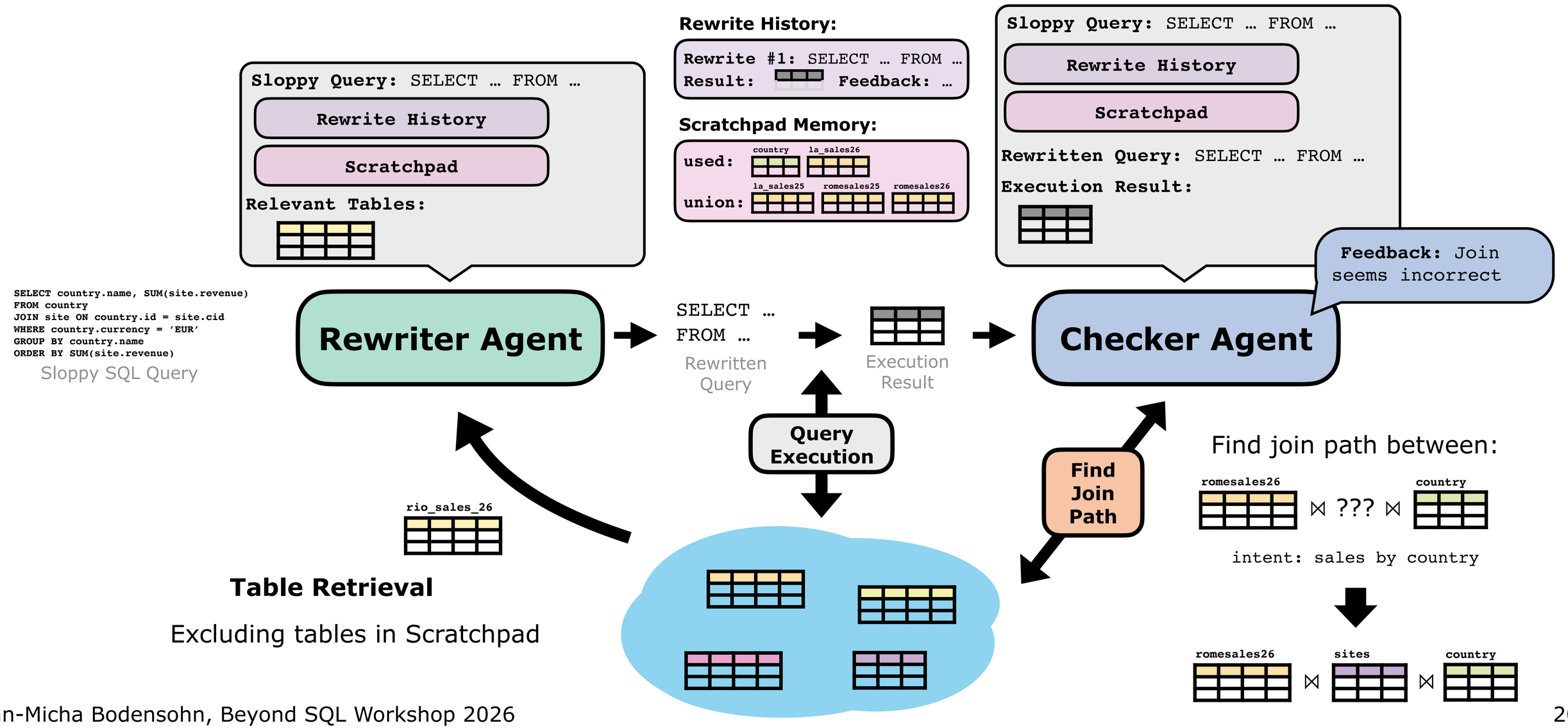
Step 3: Query Execution



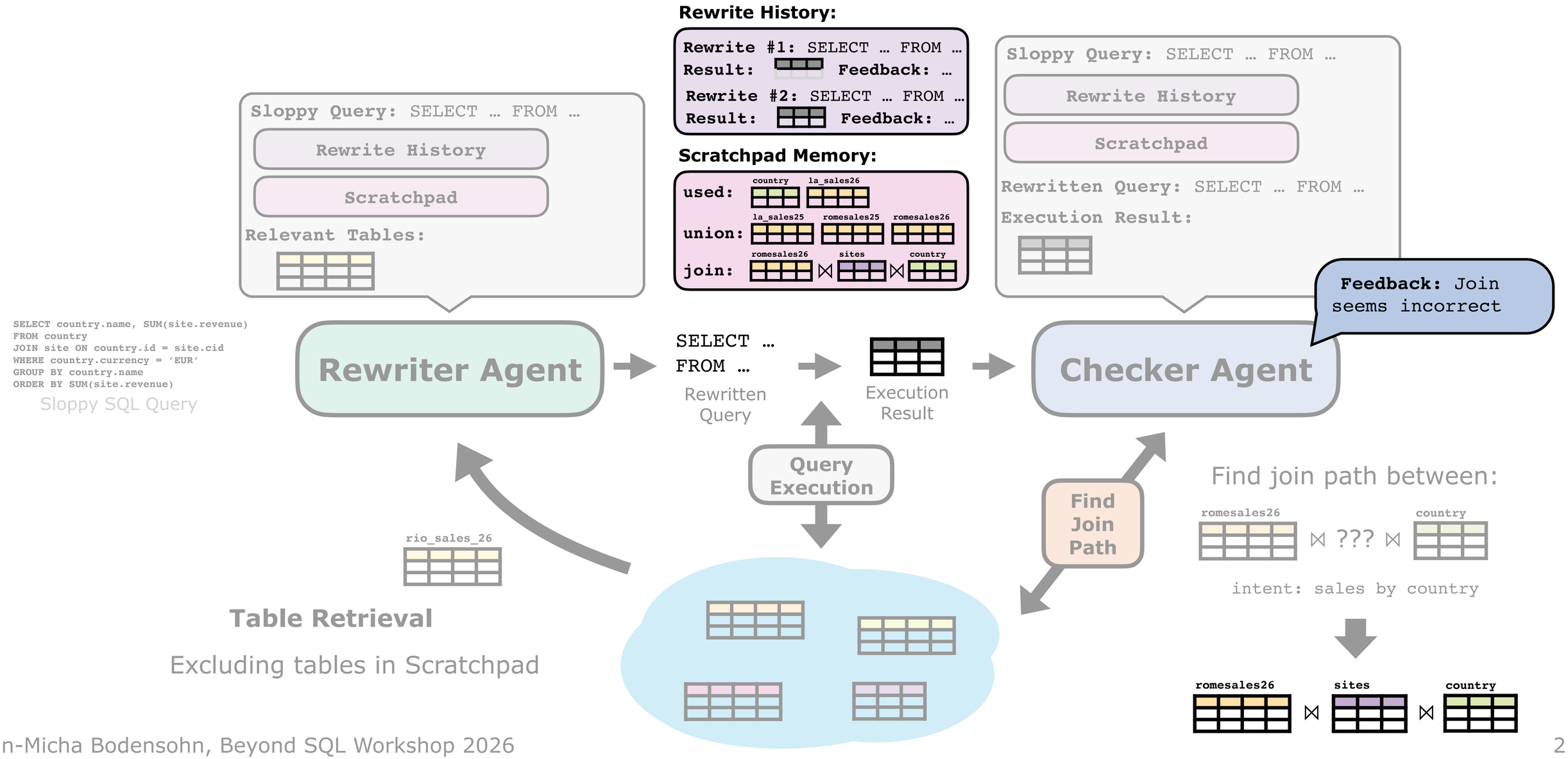
Step 4: Checker Agent



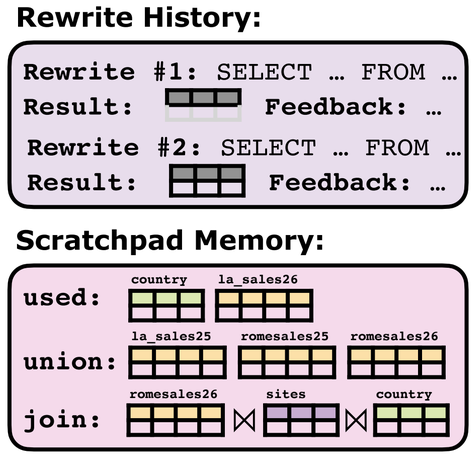
Find Join Path Action



Context Management



Start again

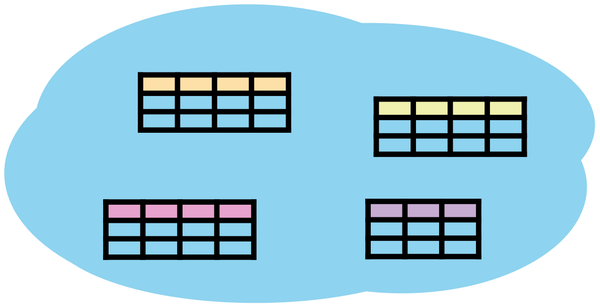



```
SELECT country.name, SUM(site.revenue)
FROM country
JOIN site ON country.id = site.cid
WHERE country.currency = 'EUR'
GROUP BY country.name
ORDER BY SUM(site.revenue)
```

Sloppy SQL Query

Rewriter Agent

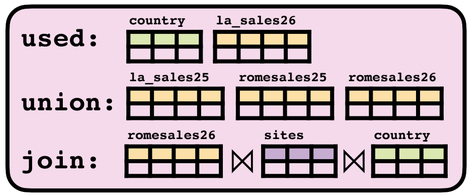
Checker Agent



Start again: Table Retrieval

Rewrite History:

Scratchpad Memory:




```
SELECT country.name, SUM(site.revenue)
FROM country
JOIN site ON country.id = site.cid
WHERE country.currency = 'EUR'
GROUP BY country.name
ORDER BY SUM(site.revenue)
```

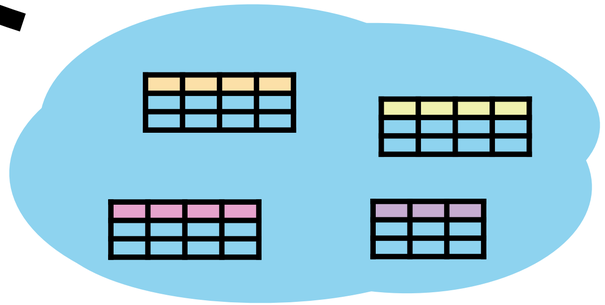
Sloppy SQL Query

Rewriter Agent

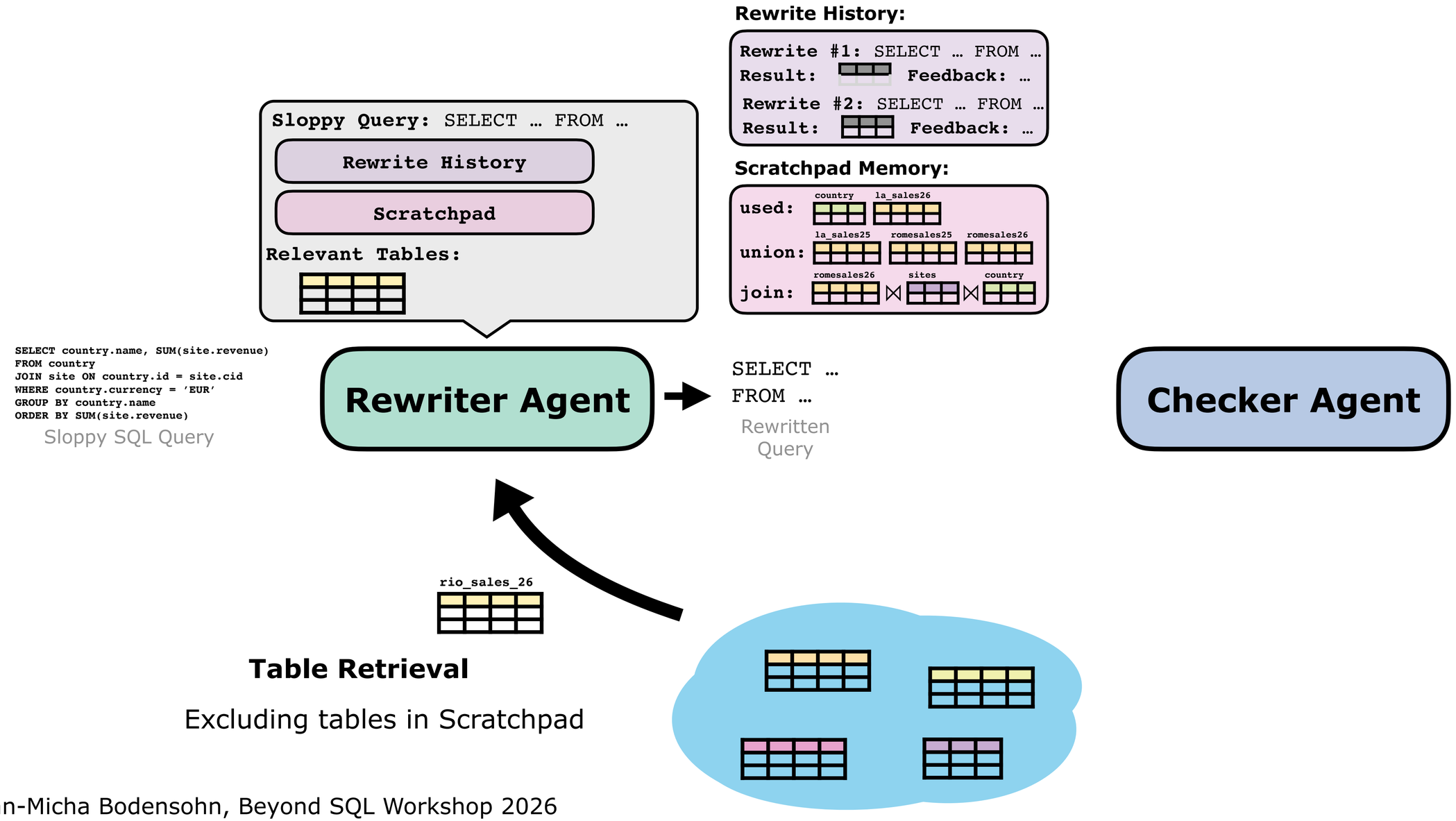
Checker Agent

Table Retrieval

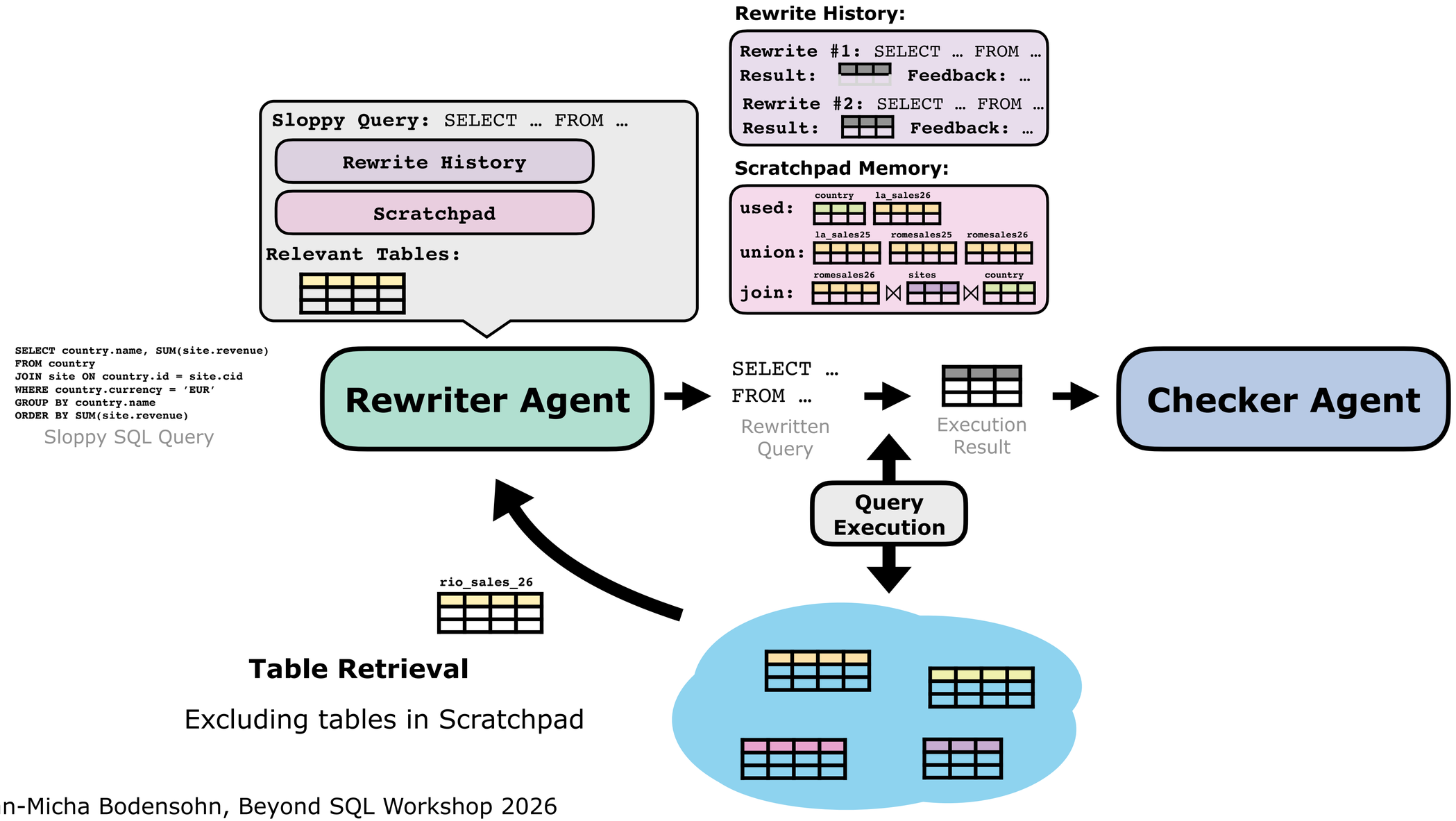
Excluding tables in Scratchpad



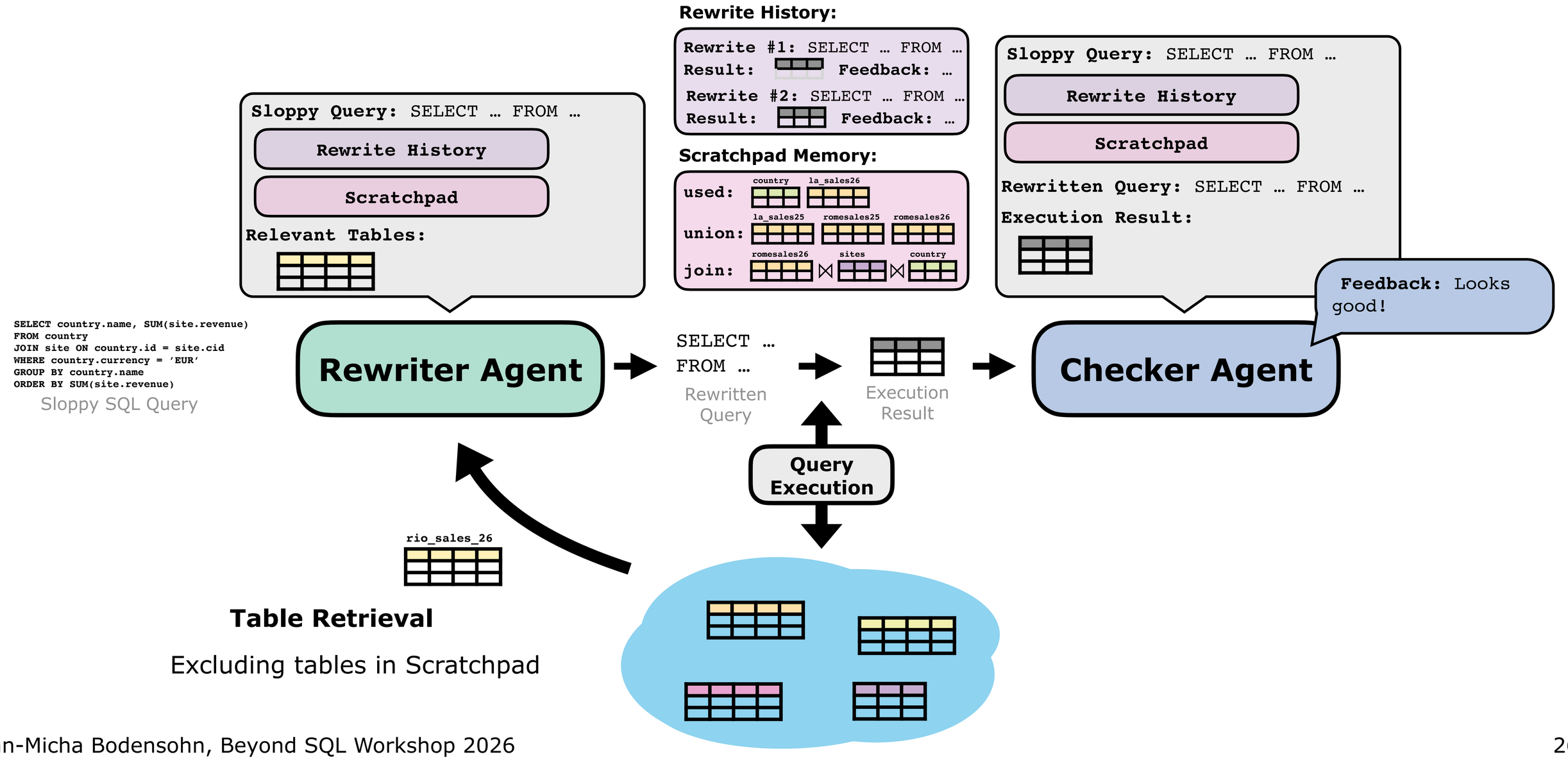
Step 2: Rewriter Agent



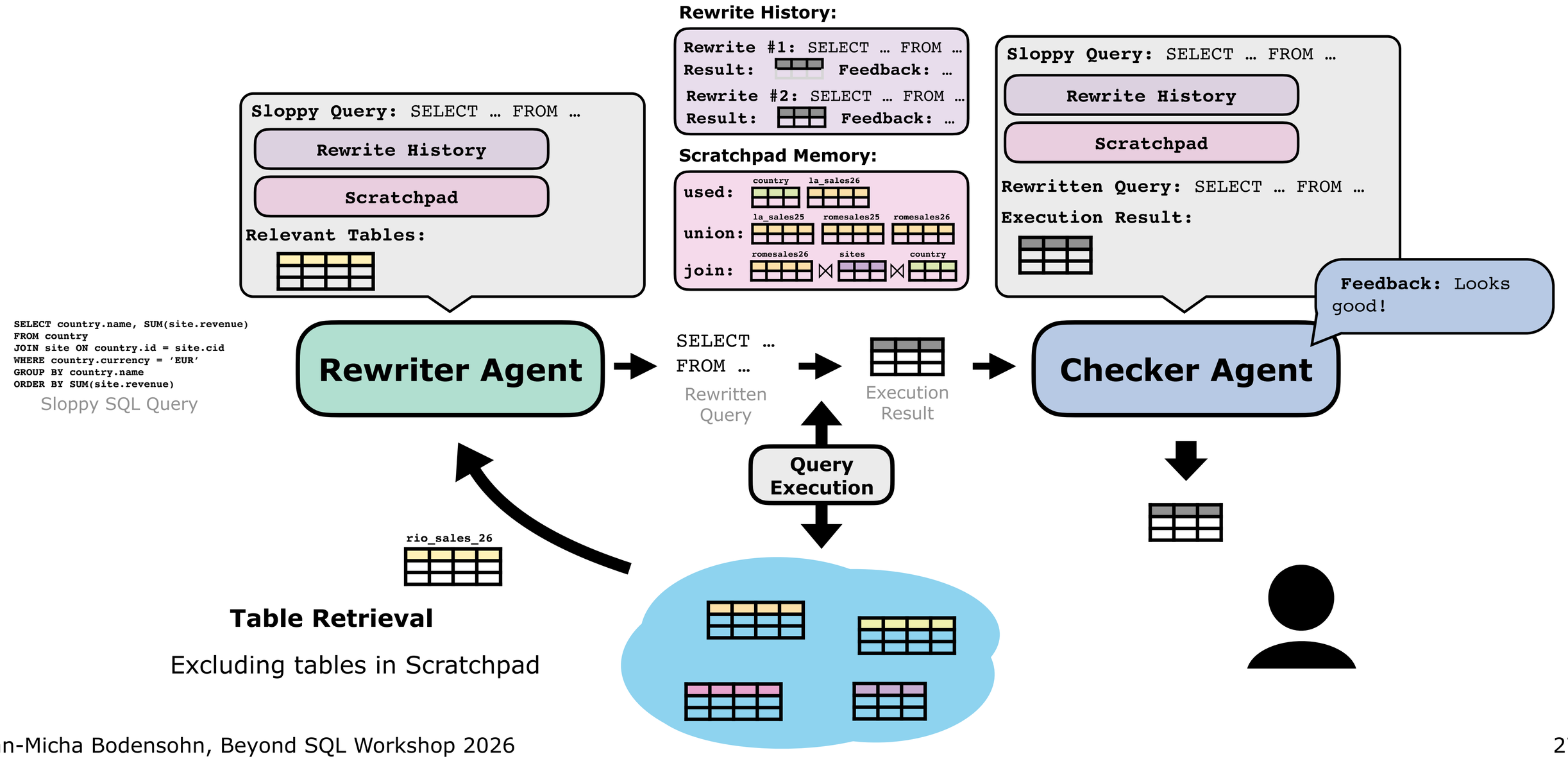
Step 3: Query Execution



Step 4: Checker Agent



Result Returned to User



Recap: Overview of SQLake

Semantic Query Rewriting in Two-Agent Feedback Loop

- **Rewriter** rewrites SQL query based on retrieved tables
- **Checker** analyzes execution result, provides feedback, calls data lake actions, returns result table

Scratchpad Memory & Rewrite History

→ Data-aware context management

Data Lake Actions:

Evaluation Setup

Sloppy SQL queries over tabular data lakes

New problem setting → no benchmarks

Created small dataset of **97 queries** over data lake[1] with **83 tables**

Included difficulties:

- Misnamed tables/attributes
- Schema mismatches requiring JOINS/UNIONS
- Incorrect filter conditions
- Formatting issues

```
SELECT name, country
FROM all_operators
WHERE …
ORDER BY …
```

simple sloppy query

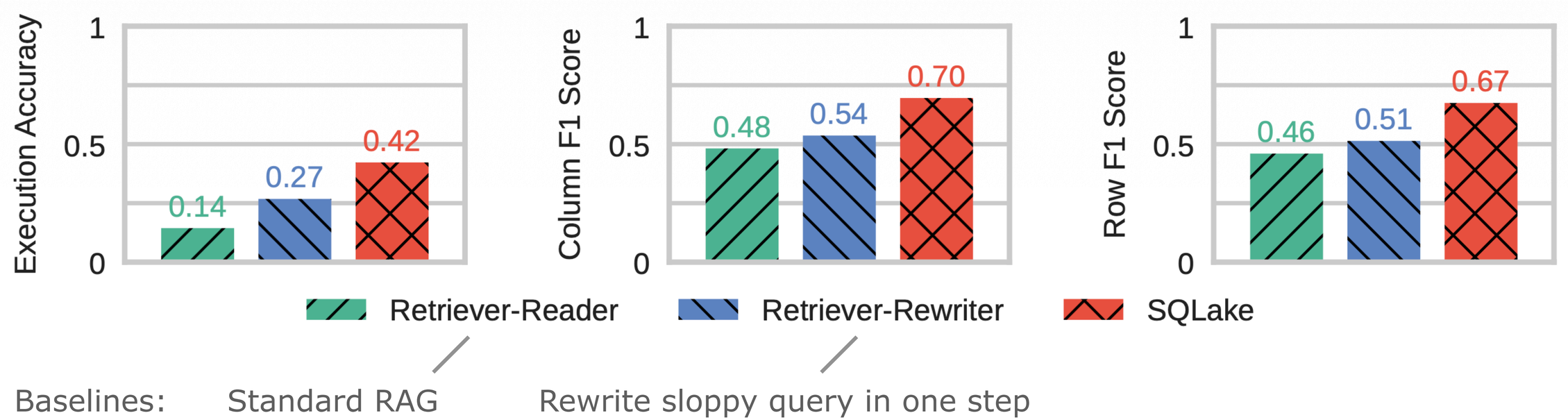
```
SELECT
   os.operator AS NAME,
   cs.country AS country
FROM
  (SELECT …
   UNION SELECT …) AS os
JOIN … ON …
WHERE …
GROUP BY …
ORDER BY …
```

complex golden query

[1]*Based on WikiDBs (Vogel et al. 2024)*

Initial Results

Measure partial correctness



Baselines: Standard RAG — Rewrite sloppy query in one step

👍 SQLake outperforms both baselines

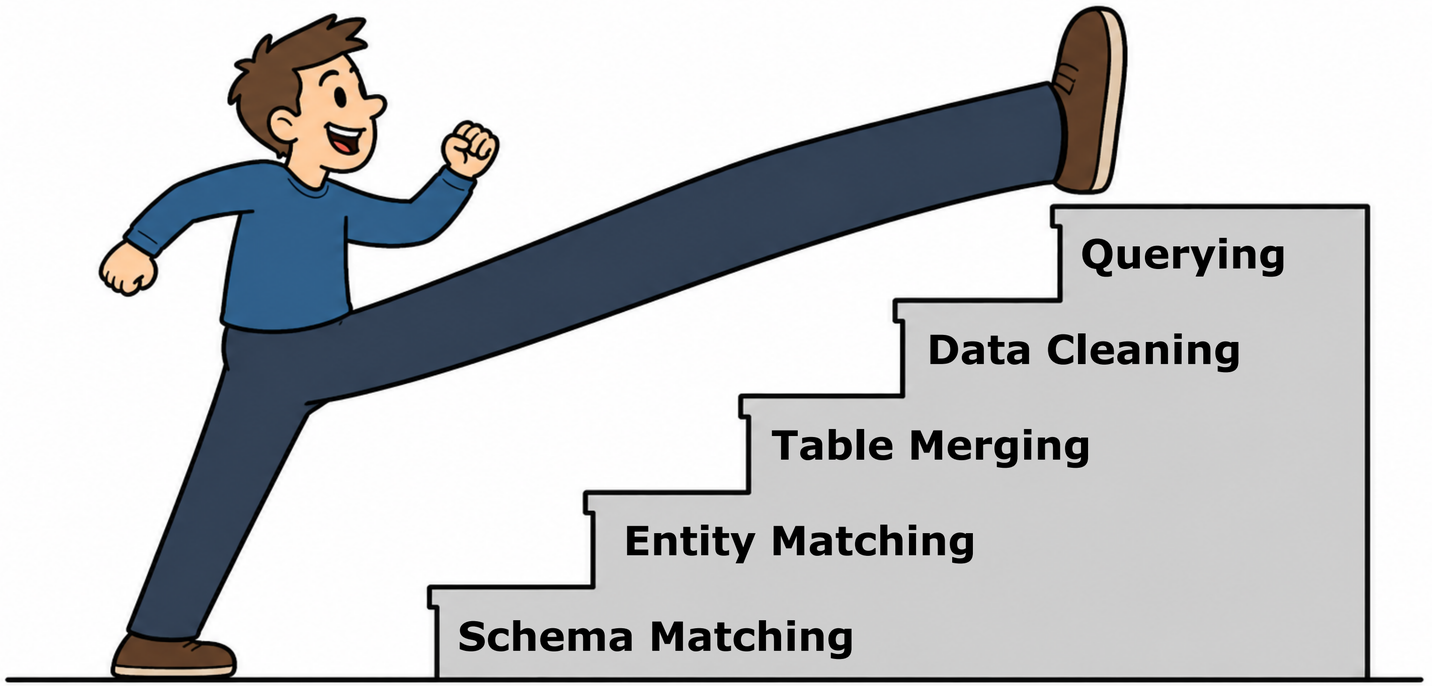
👎 Still lots of room for improvement

Experiments use GPT-5.4 with medium reasoning effort

Limitations & Future Work

Evaluating query execution over data lakes is super hard!

Because it means solving data engineering end-to-end.



Many individual steps are now (almost) solved.

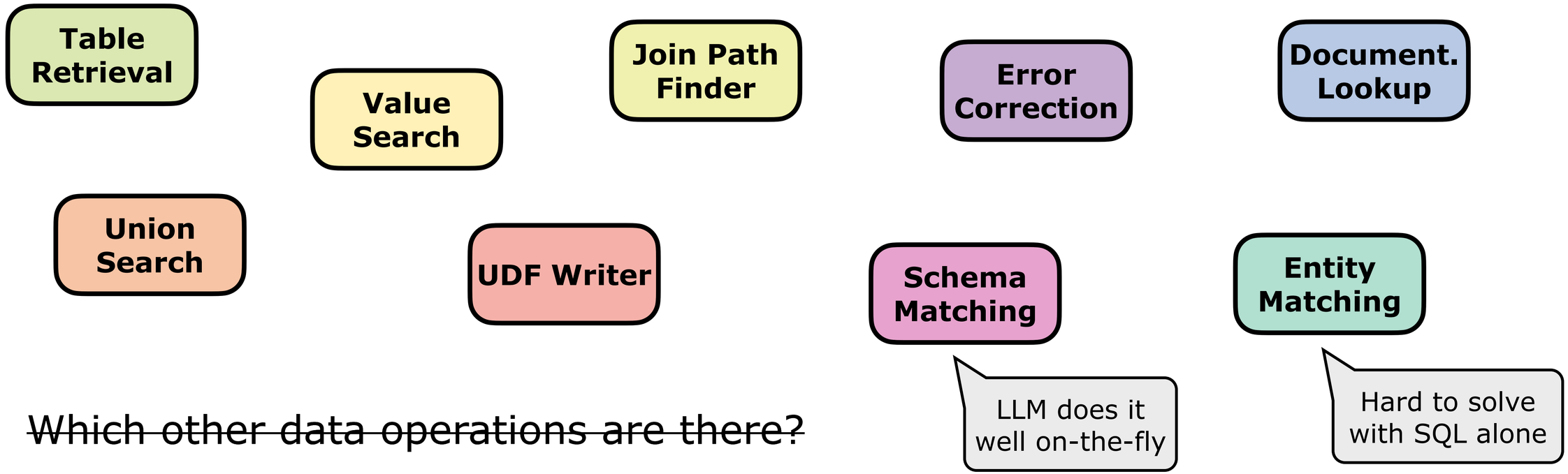
End-to-end data engineering brings new challenges:

- How to specify all the details?
- How to evaluate correctness?

Image from ChatGPT

Outlook: Algebra of Data Operations

Data Lake Actions: Overcome LLM limitations on data lakes



~~Which other data operations are there?~~

Which data operations do we need?

TECHNISCHE UNIVERSITÄT DARMSTADT

Towards Executing Sloppy SQL Queries Over Tabular Data Lakes

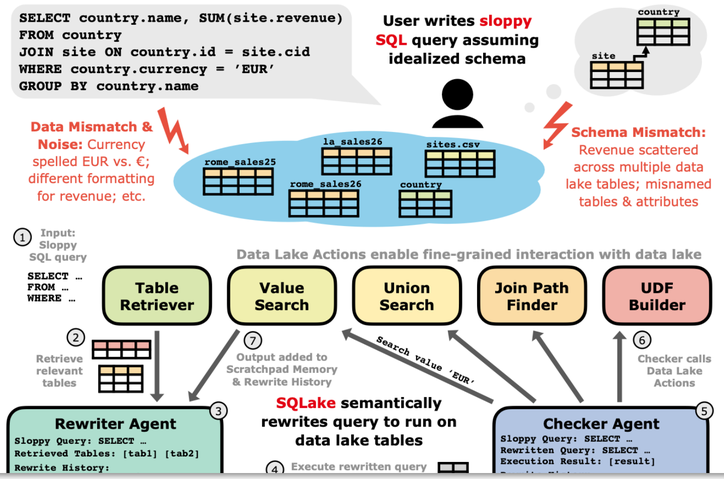
Jan-Micha Bodensohn*
Technical University of Darmstadt
jan-micha.bodensohn@tu-darmstadt.de
Darmstadt, Germany

Jakob Steinke*
Technical University of Darmstadt
jakob.m.steinke@gmail.com
Darmstadt, Germany

Carsten Binnig
Technical University of Darmstadt & hessian.ai & DFKI
carsten.binnig@tu-darmstadt.de
Darmstadt, Germany

***Abstract*—Data lakes store raw tabular data without enforcing a consistent schema, making data ingestion easy but exploration and querying difficult. Users must often combine multiple tools to find the required tables and prepare them for each query. To address this gap, we tackle a new problem setting where users directly query data lakes by writing so-called sloppy SQL queries, which simply assume an idealized schema for each query. We further present SQLake, a multi-agent framework which executes such sloppy queries by semantically rewriting them for the data lake, thus eliminating any data preparation overhead. SQLake uses a set of Data Lake Actions that enable the agents to interact with the data lake, including specialized retrievers, a Join Path Finder, and a UDF Builder to handle noise. Our initial evaluation shows the promise of this approach, bringing us closer to our goal of making data lakes as easy to query as relational databases.**

***Index Terms*—table, data lake, sql, query, llm, agent**



Thanks for your attention!