Deal with it! Towards Self-Organizing Data Schemas from Semi-Structured Inserts

TECHNISCHE UNIVERSITÄT DARMSTADT

dfki ai Deutsches Forschungszentrum für Künstliche Intelligenz
German Research Center for Artificial Intelligence

Benjamin Hättasch, Leon Krüger and Carsten Binnig

AIDM 2026

DFKI & Systems Group, TU Darmstadt

benjamin.haettasch@dfki.de

How can we deal with scenarios where the **structure** of the data **is unknown** or **changes over time**?

IT – Open a Ticket

E-Mail:

Description:

IT – Open a Ticket

E-Mail:

Category:

URL:

Error Code:

Add another row

New kinds of information

IT – Open a Ticket

Category:

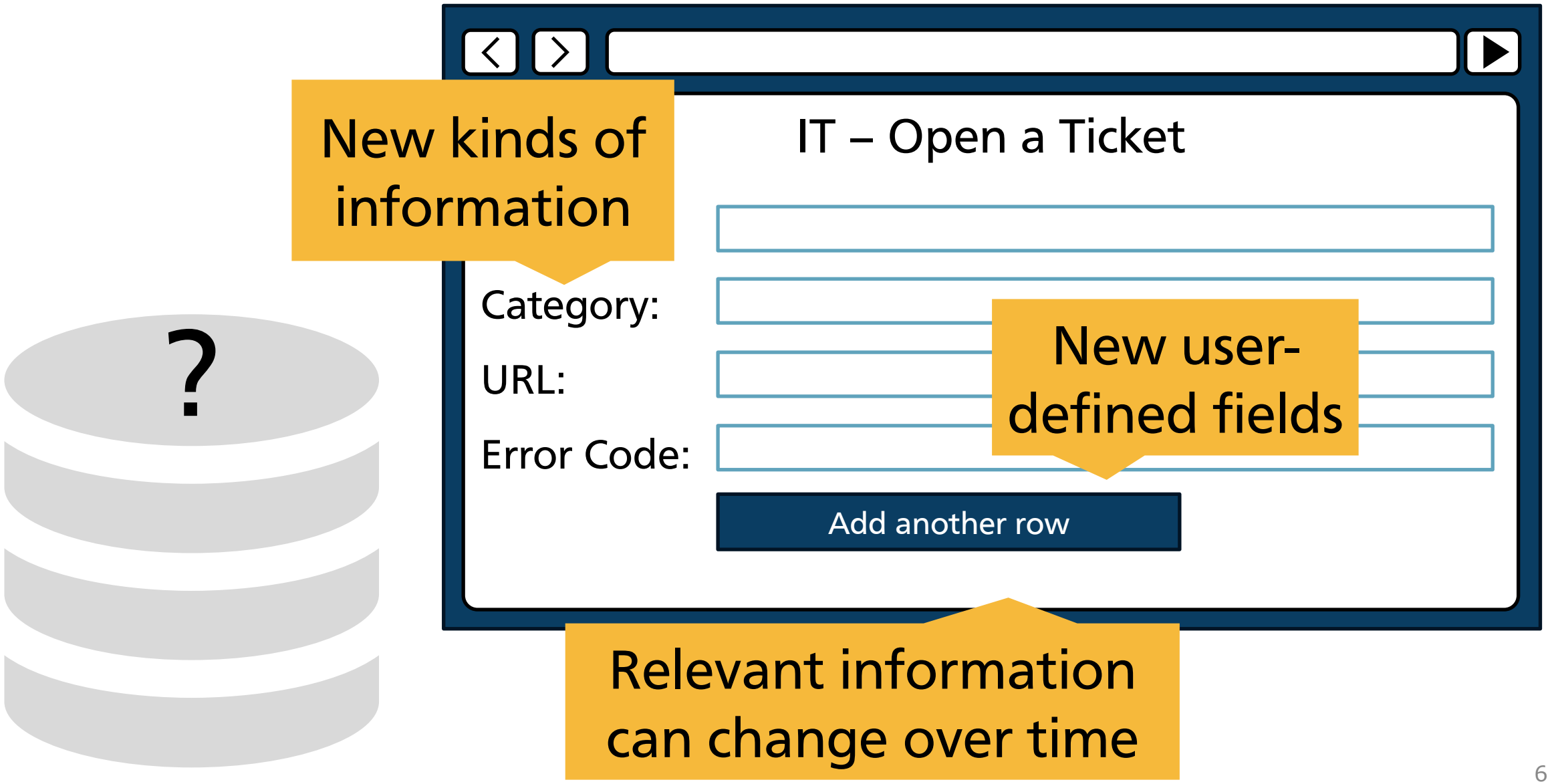
URL:

Error Code:

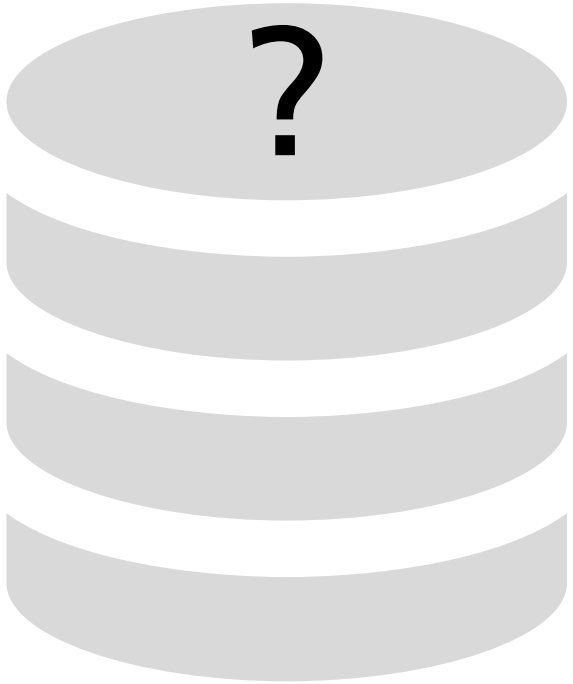
New user-defined fields

Add another row

Relevant information can change over time

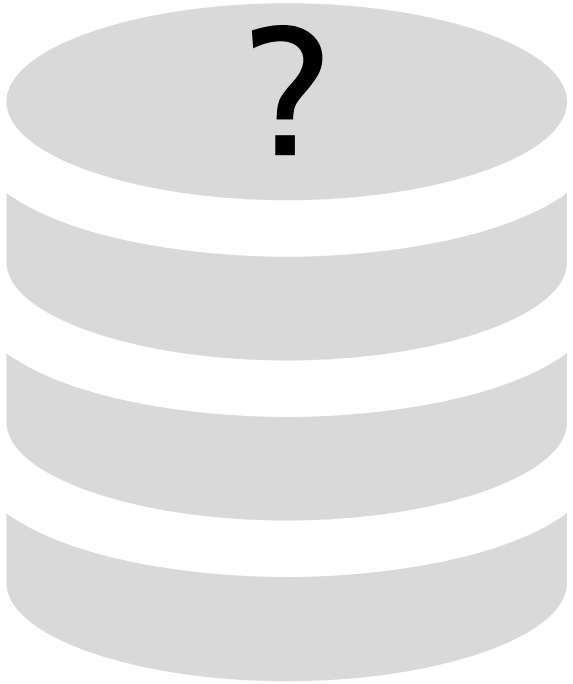

New kinds of information
IT – Open a Ticket
Category:
URL:
Error Code:
New user-defined fields
Add another row
?
Relevant information can change over time

KV-Store?



➔ Supports any data

➔ Requires schema checking on read

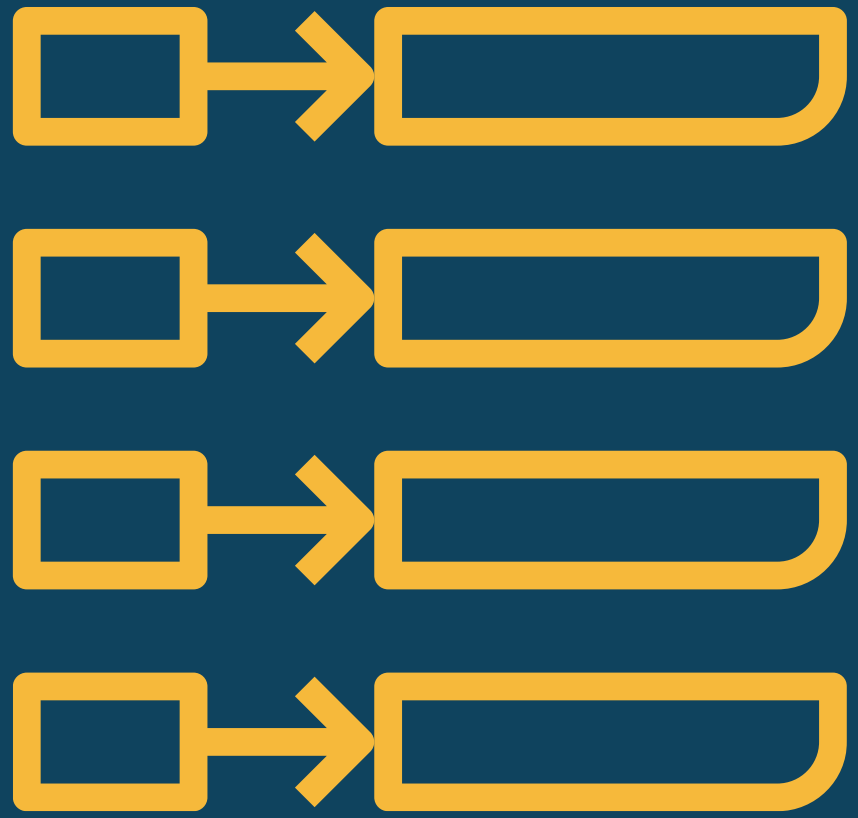


Relational DB

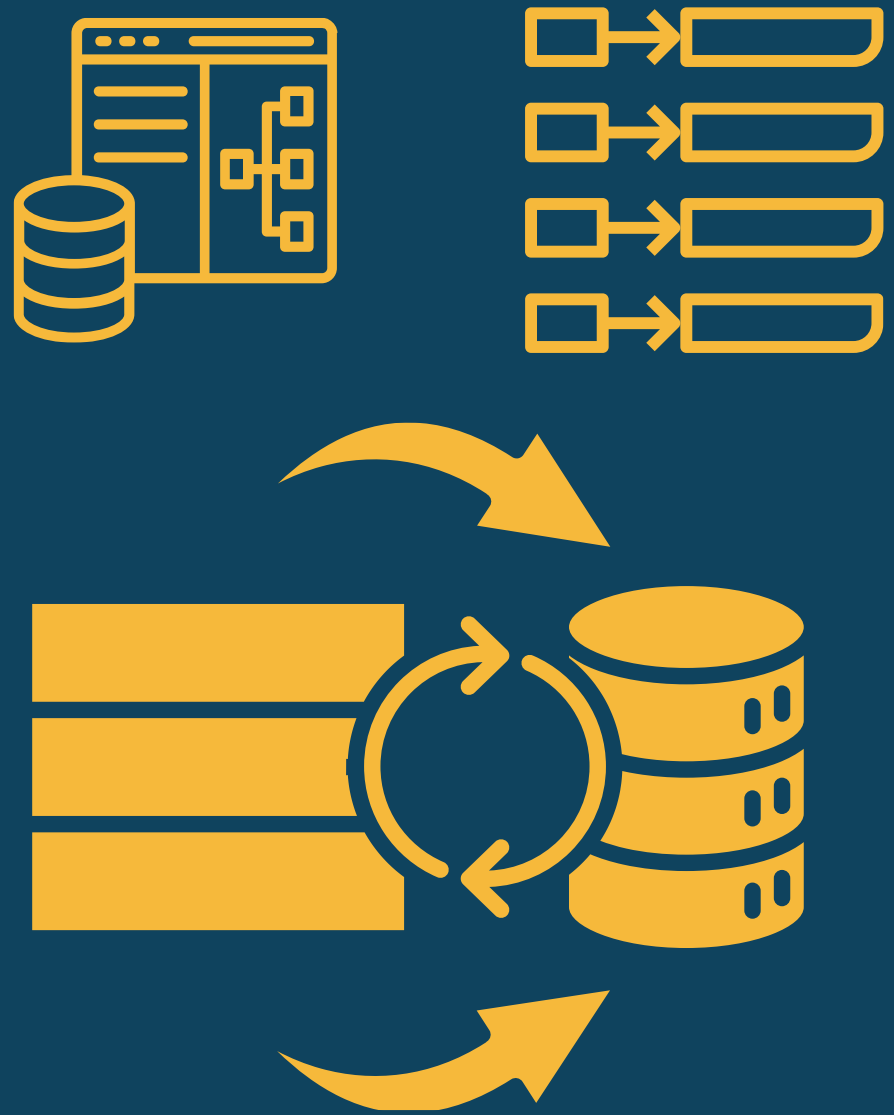
- ➔ Easy to navigate and browse
- ➔ Manual schema design (overhead)
- ➔ Cannot adjust to changes

Browsable Schema

Support any information



Icons:
RELATIONAL DATABASE by ProSymbols, Key-Value Database by Becris & automatic backup by Sinta Maulana, all from thenounproject.com (CC- BY 3.0)

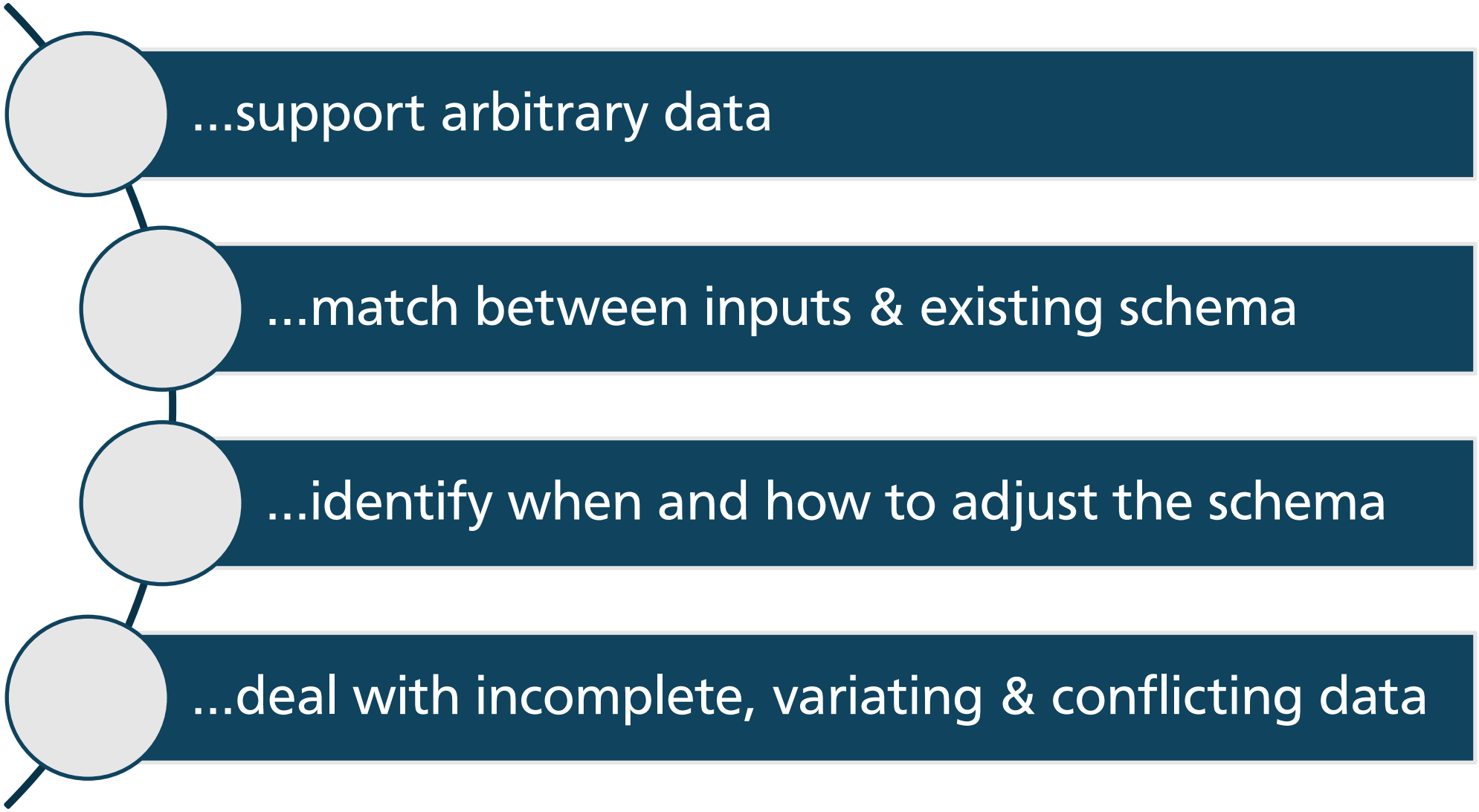


Self-Organizing Schemas

Icons:
RELATIONAL DATABASE by ProSymbols, Key-Value Database by Becris & automatic backup by Sinta Maulana, all from thenounproject.com (CC- BY 3.0)

What makes building self-organizing schemas difficult?

The DB needs to …



We propose a first step towards this:

JUSTINE: **JUST IN**sert **E**ngine

Takes arbitrary insert queries

and

finds ways to place them in a relational database,

adjusting the schema where needed.

Scenarios: How could those inserts look like?

Complete query, matching schema:

```
INSERT INTO broken_printers
(room, timestamp) VALUES (…)
```

Broken Printers	
Room	**Report date**
S2|02 D108	2026-05-31 14:34
…	…

Synonyms or related concepts used:

```
INSERT INTO paper_jam
(location, timestamp) VALUES (…)
```

Match between query and schema

Table or column information missing:

```
INSERT INTO broken_printers VALUES (…)
INSERT (room, timestamp) VALUES (…)
```

By allowing users to omit table and/or column names, we truly relieve them from the need to think about the schema.

Scenarios: How could those inserts look like?

Additional columns not yet in schema:

```
INSERT (page, date, URL, error_code) VALUES (…)
```

Derive/choose column names and extend existing table

WebApp Errors			
View	**Timestamp**	**URL**	**Error**
5	2025-03-02	http://,,,	
…	…	…	

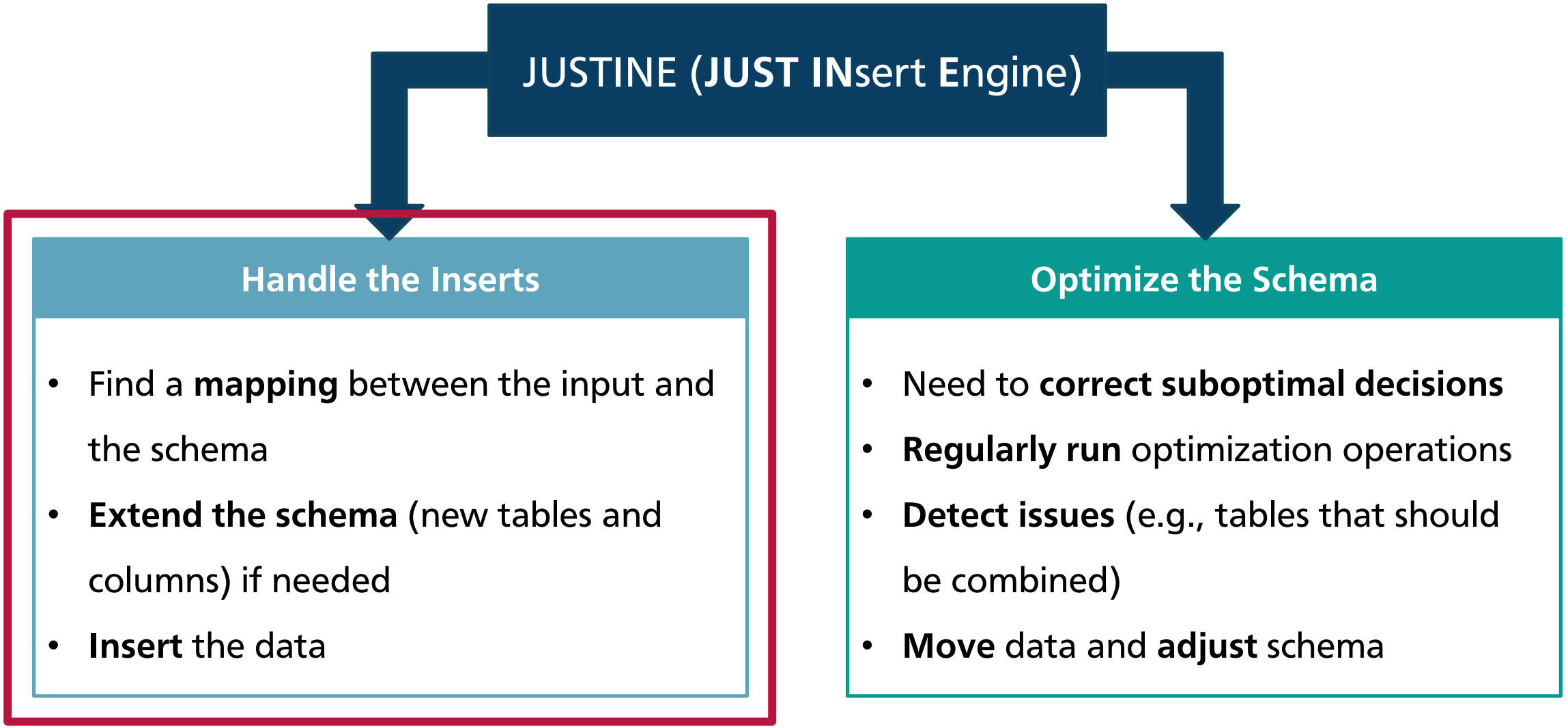
Entirely new data:

```
INSERT (sender, subject, did_click_on_link, timestamp) VALUES (…)
```

Derive design (including names) for a new table and add it to the schema

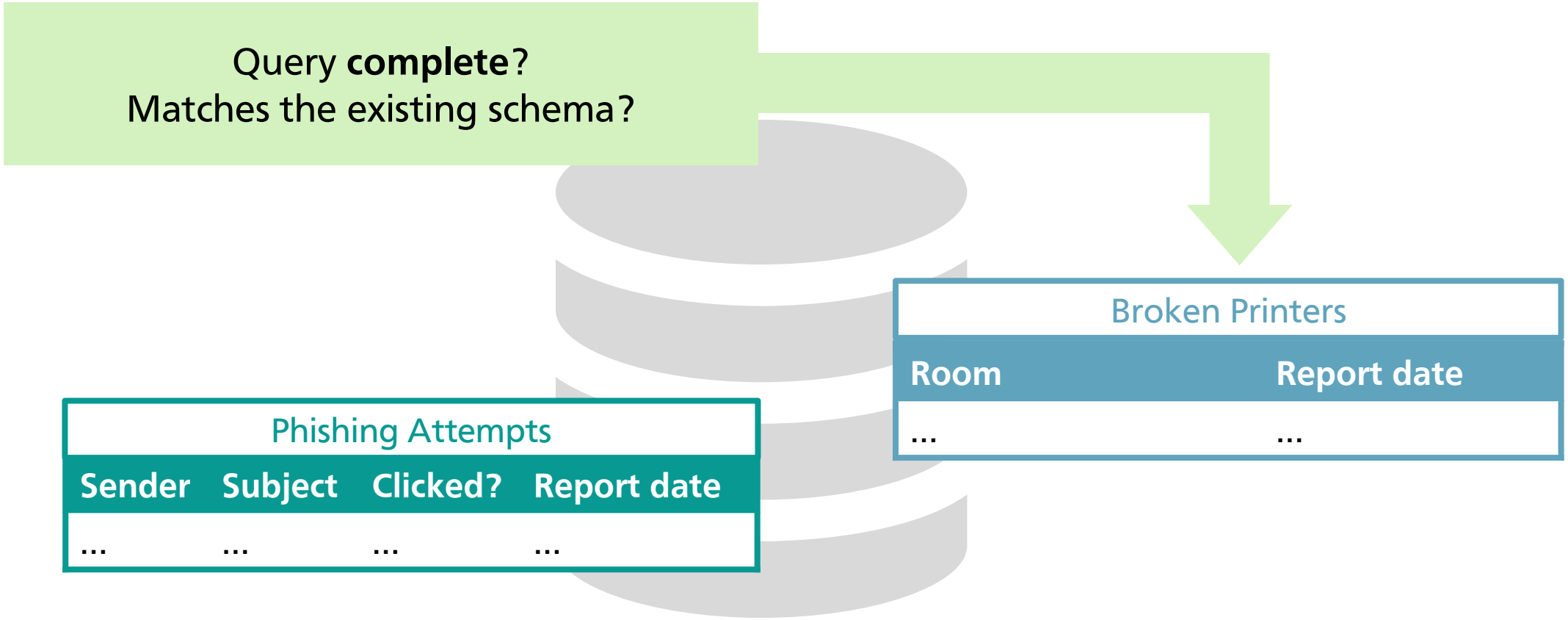
Phishing Attempts			
Sender	**Subject**	**Clicked?**	**Report date**
…	…	…	…

JUSTINE consists of two central components:



How inserts are handled:

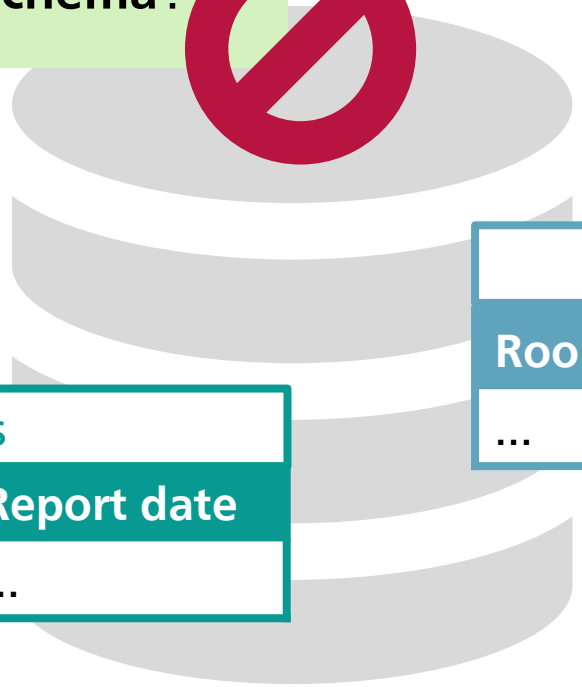
```
INSERT INTO broken_printers (room, timestamp)
VALUES (S2|02 D108, Sq, 2026-05-31 14:34);
```



How inserts are handled:

```
INSERT (location, timestamp)
VALUES (S2|02 D108, 2026-05-31 14:34);
```

Query **incomplete** or
not **matching the** existing **schema**?



Broken Printers

Room	Report date
…	…

Phishing Attempts

Sender	Subject	Clicked?	Report date
…	…	…	…

How inserts are handled:

Phishing Attempts			
Sender	**Subject**	**Clicked?**	**Report date**
…	…	…	…

Broken Printers	
Room	**Report date**
…	…

```
INSERT (location, timestamp) VALUES (S2|02 D108, 2026-05-31 14:34);
```

1) Phishing Attempts

2) Broken Printers

3) Paper Jams (NEW)

1) Find n = 3 possible tables

How inserts are handled:

Phishing Attempts			
Sender	**Subject**	**Clicked?**	**Report date**
…	…	…	…

Broken Printers	
Room	**Report date**
…	…

```
INSERT (location, timestamp) VALUES (S2|02 D108, 2026-05-31 14:34);
```

1) Phishing Attempts

Location → Location (NEW); Timestamp → Report Date

2) Broken Printers

Location → Room; Timestamp → Report Date

3) Paper Jams (NEW)

Place → Location (NEW); Timestamp → Date (NEW)

2) For each possible table determine column mapping

How inserts are handled:

Phishing Attempts			
Sender	**Subject**	**Clicked?**	**Report date**
…	…	…	…

Broken Printers	
Room	**Report date**
…	…

```
INSERT (location, timestamp) VALUES (S2|02 D108, 2026-05-31 14:34);
```

1) Phishing Attempts ✗

Location → Location (NEW); Timestamp → Report Date | NULL: 3, NEW: 1

2) Broken Printers ✓

Location → Room; Timestamp → Report Date | NULL: 0, NEW: 0

3) Paper Jams (NEW) ✗

Place → Location (NEW); Timestamp → Date (NEW) | NULL: 0, NEW: 2

3) Use heuristics to determine which candidate is best

How inserts are handled:

Phishing Attempts			
Sender	**Subject**	**Clicked?**	**Report date**
…	…	…	…

Broken Printers	
Room	**Report date**
…	…

```
INSERT (location, timestamp) VALUES (S2|02 D108, 2026-05-31 14:34);
```

1) Phishing Attempts ✗

Location → Location (NEW); Timestamp → Report Date | NULL: 3, NEW: 1

2) Broken Printers ✓

Location → Room; Timestamp → Report Date | NULL: 0, NEW: 0

3) Paper Jams (NEW) ✗

Place → Location (NEW); Timestamp → Date (NEW) | NULL: 0, NEW: 2

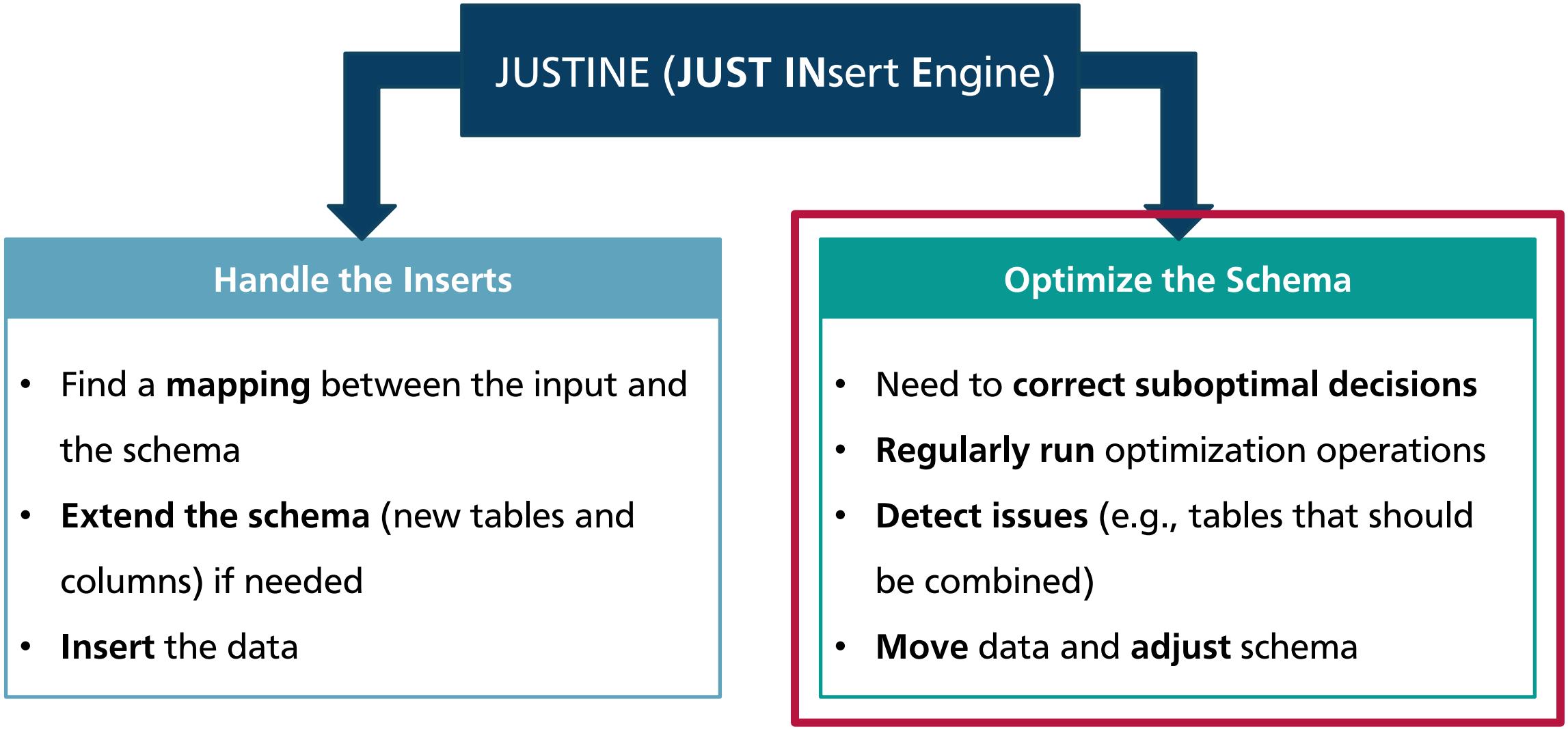
4) Choose that candidate, update the schema if needed, and insert the data

We use LLMs to find the mappings:

- Two Llama 3 models (8B parameters)
- Finetuned for table and column prediction
- On a dataset curated from
  - 52 DBs from BIRD
  - 151 DBs from SPIDER
  - 99 DBs from WikiDBs
  - ➔ Wide range of naming conventions

- By removing parts of the information or replacing it with synonyms
- Training cost reduction via QLoRA
- CSV-formatting for table representation in prompts
- Augmented with (fuzzy) string matching (➔ better quality & lower inference costs)

JUSTINE consists of two central components:



Optimize the schema

- Insertion flow just extends the schema
- How can we revert suboptimal decisions?
- Regularly run operations for schema and data placement optimization

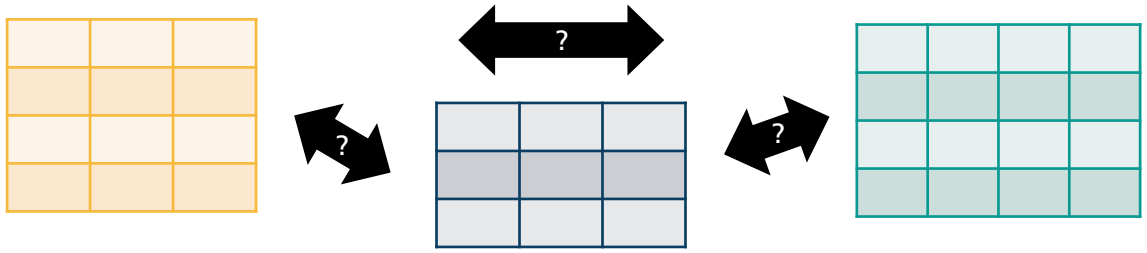
- Typical Issue: Wording differs too strongly

  ➔ Contents of the same kind are placed in different tables

  ➔ Tables need to be combined

Combine tables through structural & semantic matching

1) Compute similarity of table pairs



1) How well do the number of columns match?

2) How well do the table and column names match?
➔ Use distances of embeddings of the names

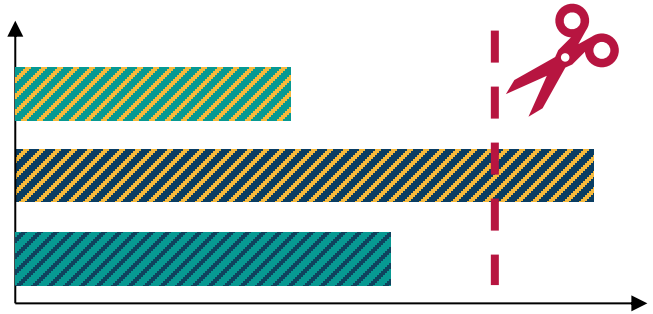
3) How well do the contents of the tables match?
➔ Use distances of embeddings over the instances

# of candidates

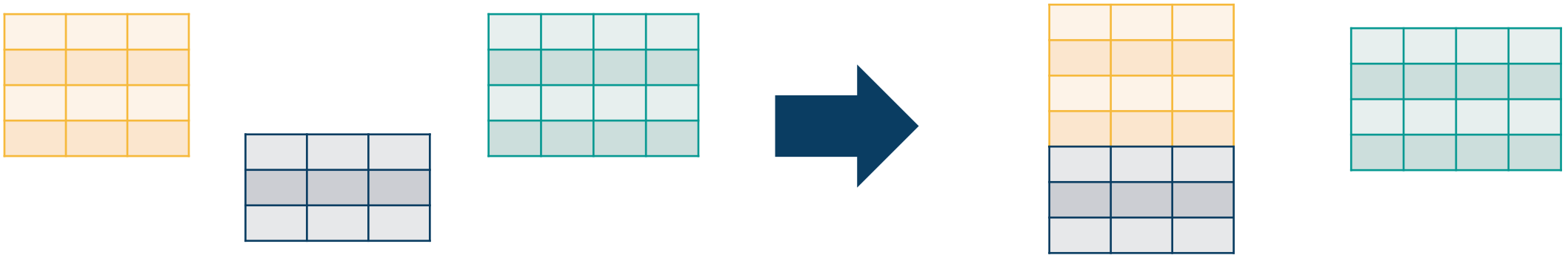
computation effort per comparison

Combine tables through structural & semantic matching

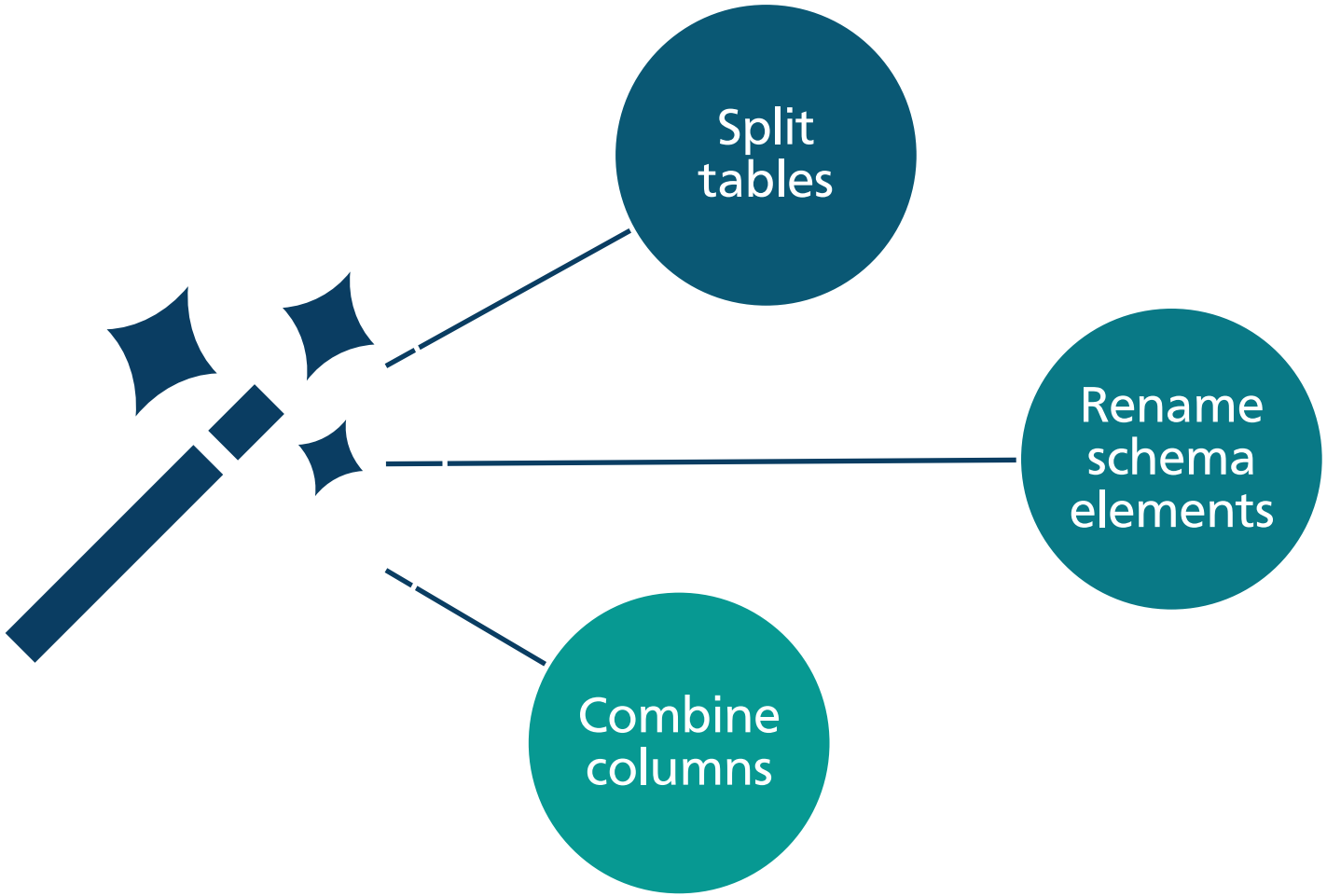
2) Only consider candidates above a given threshold



3) Combine



What could be other optimization operations?



How can we measure what a good schema is?

gold

movies

title	director	release
ET	Spielberg	11.06.1982
It	Muschietti	08.09.2017
Up	Docter	29.05.2009

directors

name	dob
Spielberg	18.12.1946
Muschietti	26.08.1973
Tarantino	27.03.1963

result

movies

name	director	release
ET	Spielberg	11.06.1982
It	Muschietti	08.09.2017
Spielberg	NULL	18.12.1946
Tarantino	NULL	27.03.1963
Up	Docter	29.05.2009

directors

l_name	birthdate
Muschietti	26.08.1973

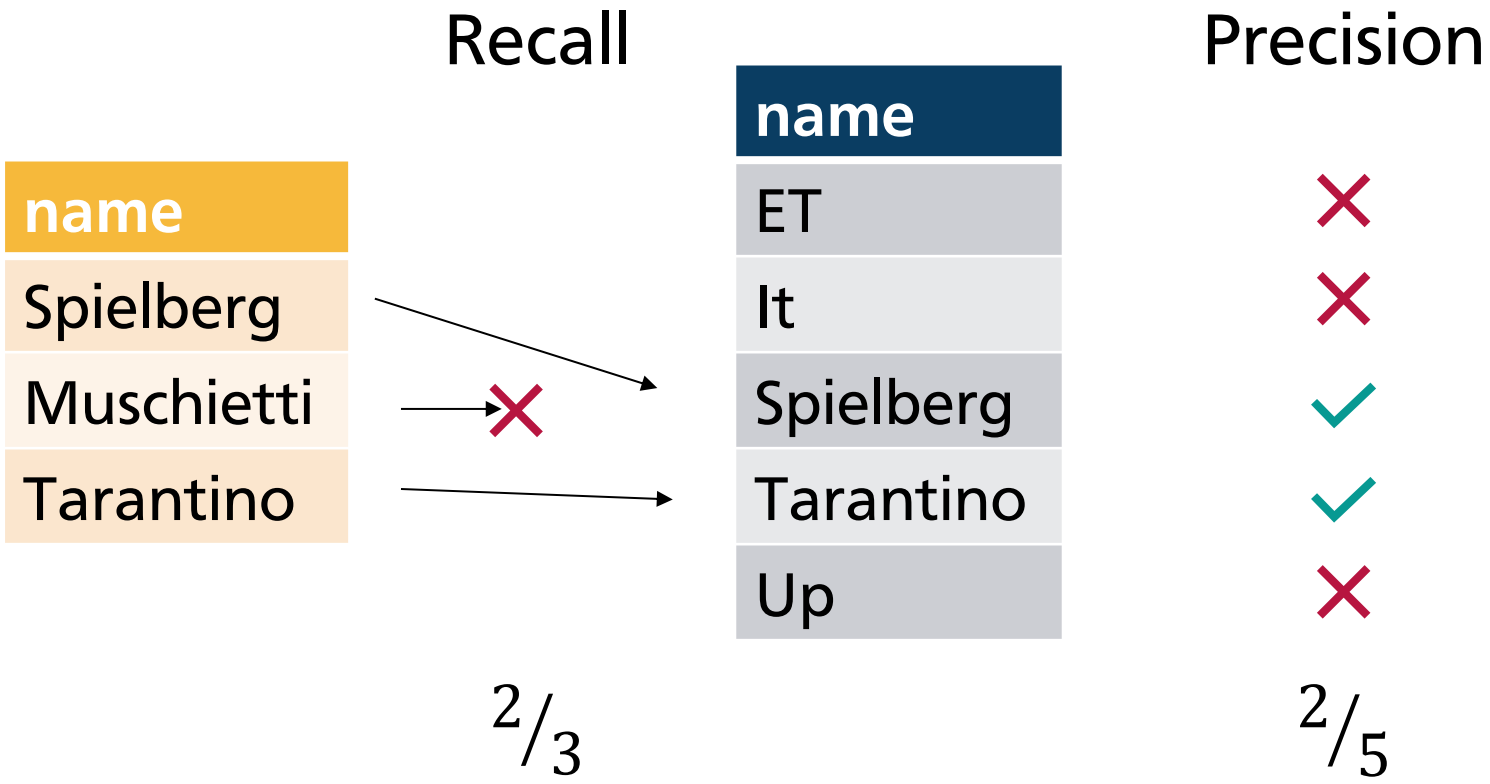
wide

movies and directors

m_name	director	release	d_name	birthdate
ET	Spielberg	11.06.1982	NULL	NULL
It	Muschietti	08.09.2017	NULL	NULL
NULL	NULL	NULL	Spielberg	18.12.1946
NULL	NULL	NULL	Tarantino	27.03.1963
Up	Docter	29.05.2009	NULL	NULL
NULL	NULL	NULL	Muschietti	26.08.1973

➔ Transfer F1 score to whole DBs

Database F1 score: Transfering F1 to whole databases

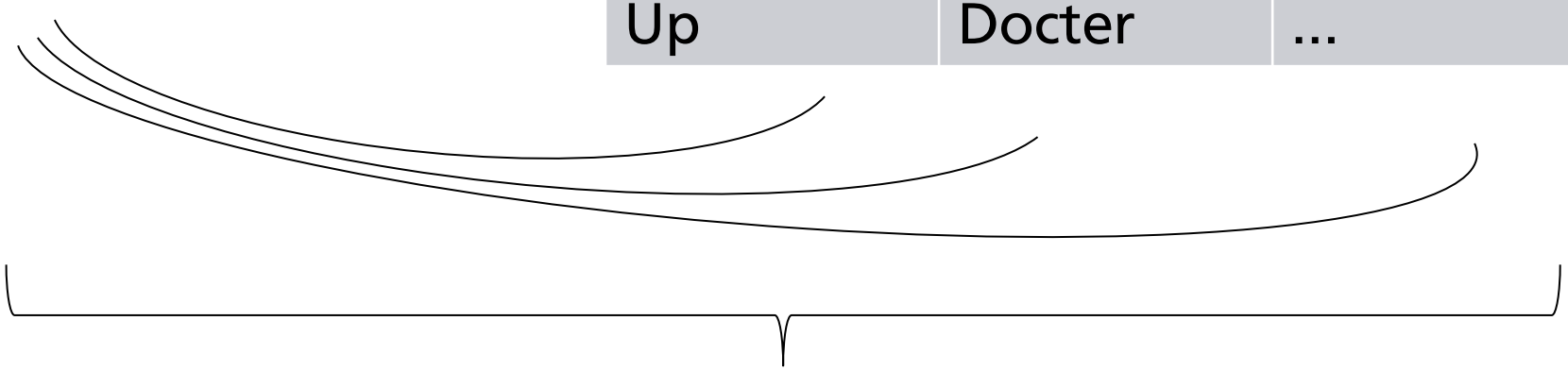


$$F_1 = \frac{2 \cdot p \cdot r}{p + r} = \frac{2 \cdot 0.66 \cdot 0.4}{0.66 + 0.4} = 0.498$$

Database F1 score: Transfering F1 to whole databases

name
Spielberg
Muschietti
Tarantino

name	director	release
ET	Spielberg	...
It	Muschietti	...
Spielberg	NULL	...
Tarantino	NULL	...
Up	Docter	...

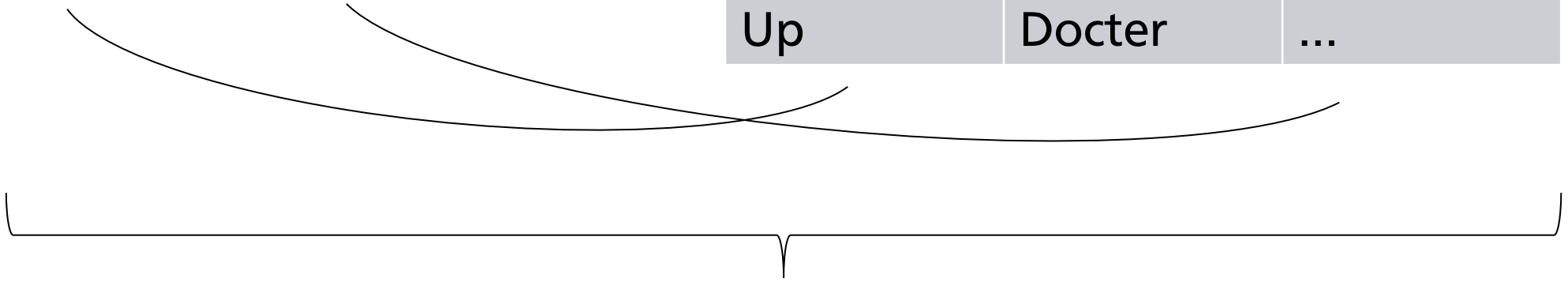


Determine the best match and use their score as F1 score for the gold column

Database F1 score: Transfering F1 to whole databases

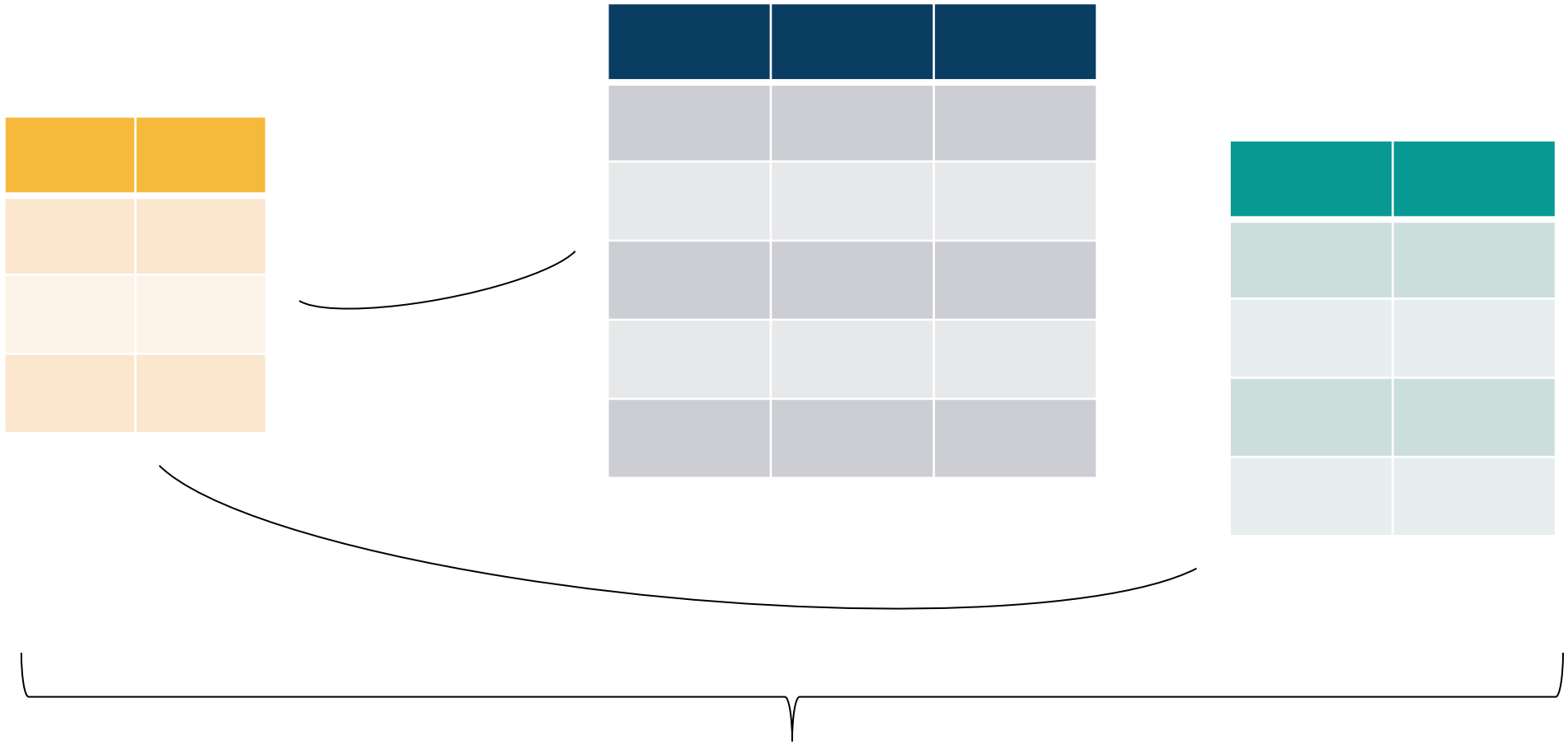
name	dob
Spielberg	...
Muschietti	...
Tarantino	...

name	director	release
ET	Spielberg	...
It	Muschietti	...
Spielberg	NULL	...
Tarantino	NULL	...
Up	Docter	...



Compute table to table F1 by averaging over the scores of corresponding columns

Database F1 score: Transfering F1 to whole databases

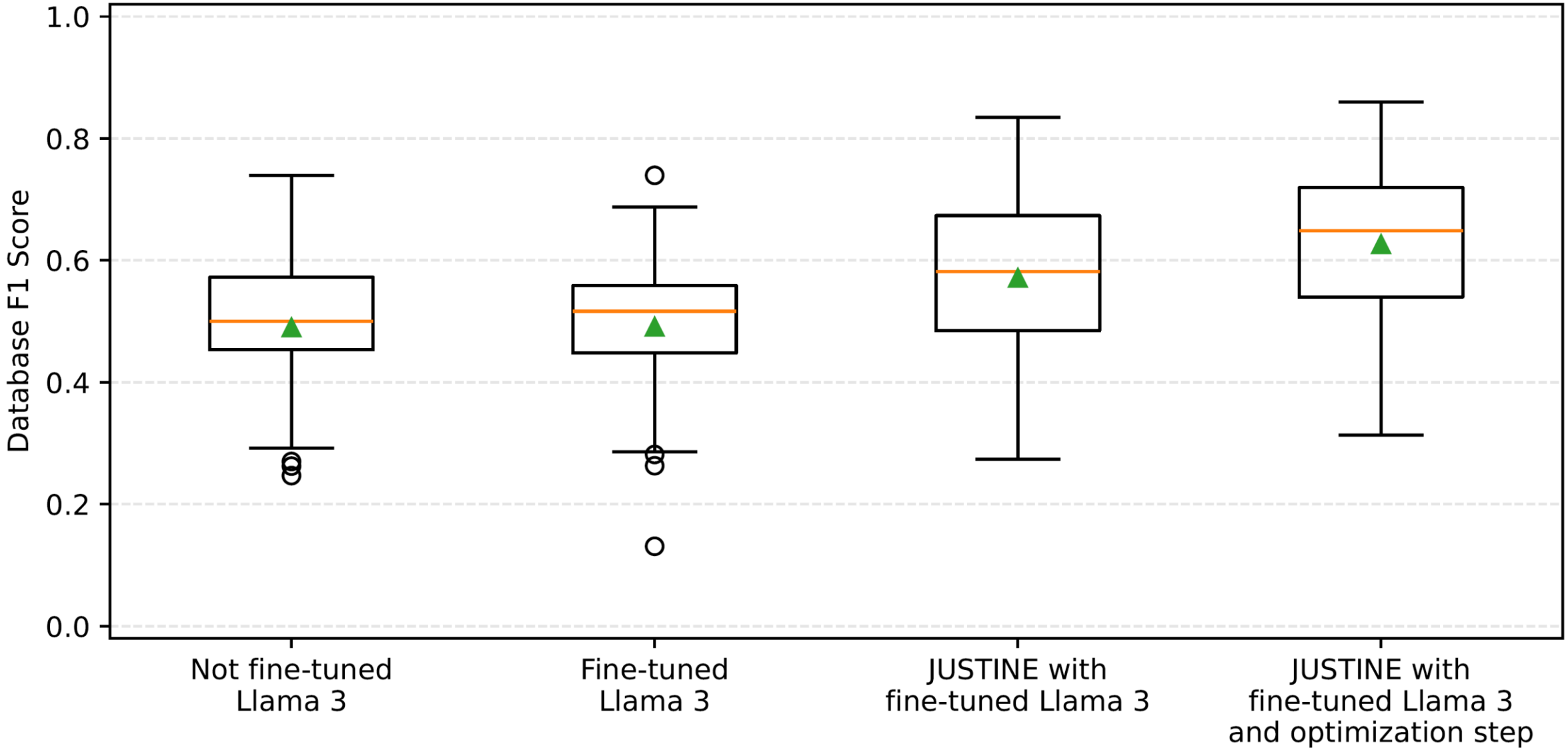


Compute overall F1 score as the average
over the best table matches

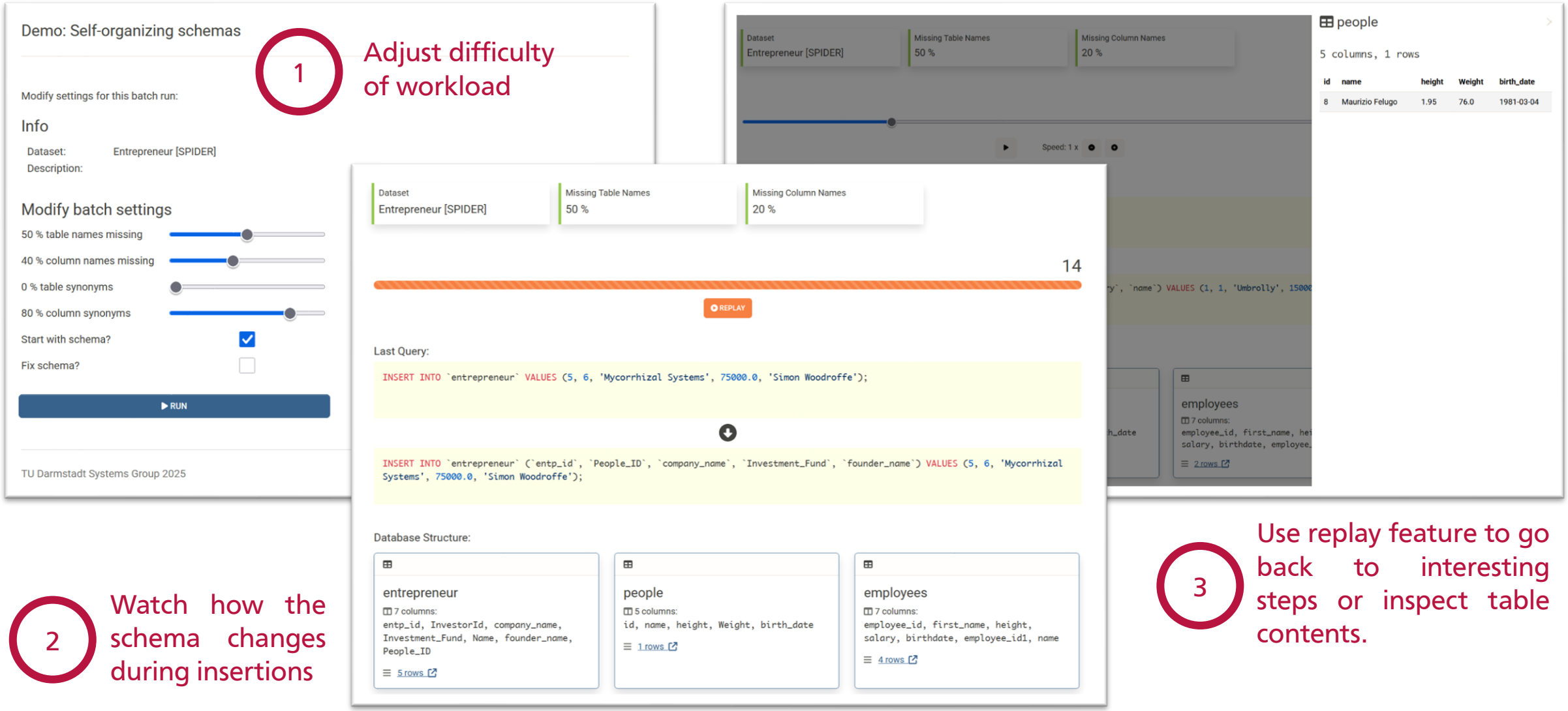
Evaluation: Setup & Datasets

- **30** different insertion **workloads**
  - based on the DBs from **BIRD, Spider & WikiDBs**
  - 10 each
- Modifications:
  - 50 % table **names removed**
  - 25 % **synonyms** for table names
  - 20 % column names removed
  - 30 % column synonyms
- Parameters:
  - 3 candidate mappings
  - One is always **fuzzy string matching**, the other two **LLM-based**

Evaluation: Initial Results



We integrated our approach into a demo:



1 Adjust difficulty of workload

2 Watch how the schema changes during insertions

3 Use replay feature to go back to interesting steps or inspect table contents.

This is just the start – what could be next steps?

Batch Insertion
- Infer placement from multiple instances at once

Optimizing the schema
- Rename
- Split
- Combine columns

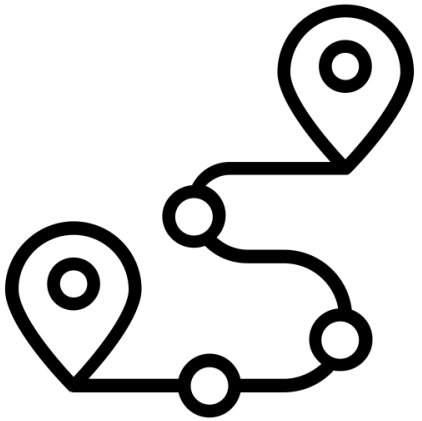
Protecting the schema
- Fixate parts of the schema
- Learn from UDFs

Beyond direct inserts
- Normalization
- Split data into multiple tables
- Adjust values

Leverage reads
- Adjust schema based on SELECT queries

Improve usage of LLMs
- Test other models
- Evaluate other prompts (e.g., in combination with batch inserts)

Leverage Domain Knowledge
- Provide additional sources (e.g., technical documentation) to the LLM



Our approach enables self-organizing data schemas based on the INSERTs:

Flow:

- Infer the correct tables and columns from incomplete queries
- Adjust the schema if needed
- Regularly run optimization operations

- Quality can be measured using DB F1 score & sparsity
- First steps towards a new class of data storage
- Code and demo available: link.tuda.systems/JUSTINE

Thanks for your attention. What are your questions?