FlockMTL: A Multimodal Querying Community Extension for DuckDB

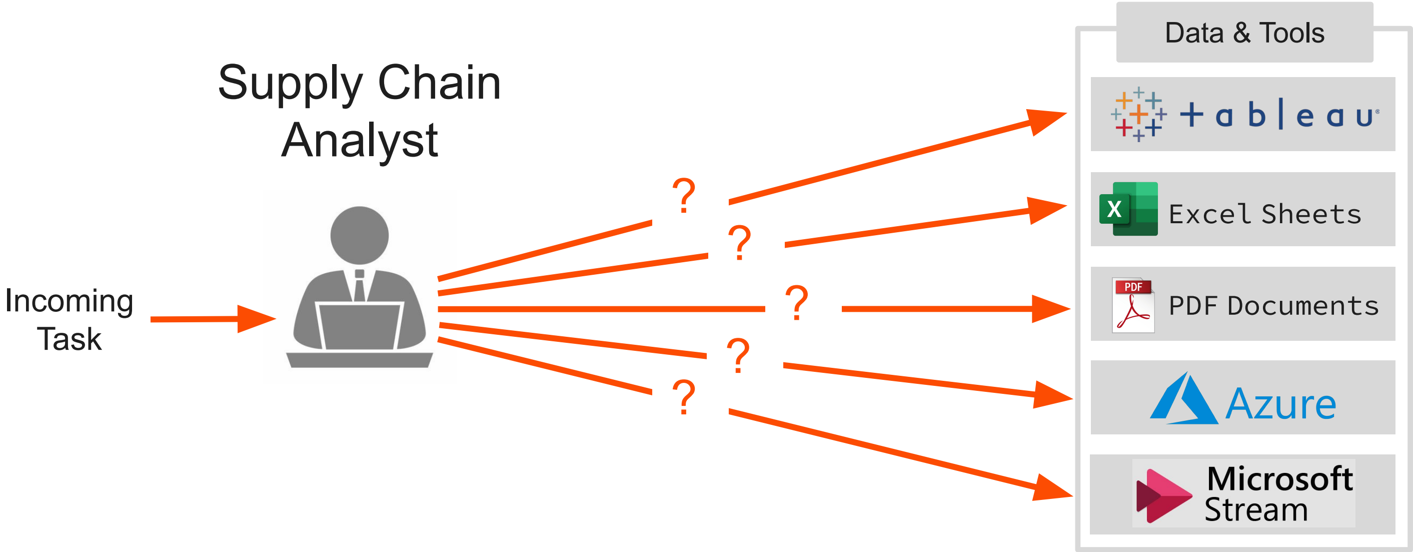
Amine Mhedhbi

POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

Revisiting Business Processes Example

Knowledge
Worker

Application Example

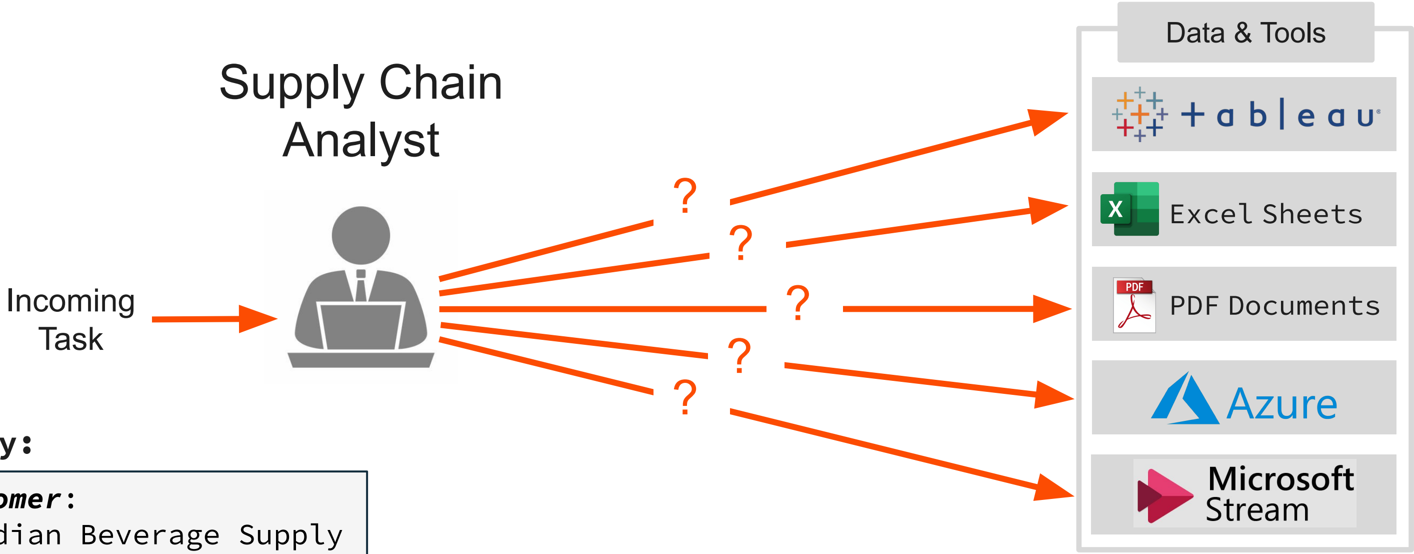


Incoming Task

Supply Chain Analyst

Data & Tools

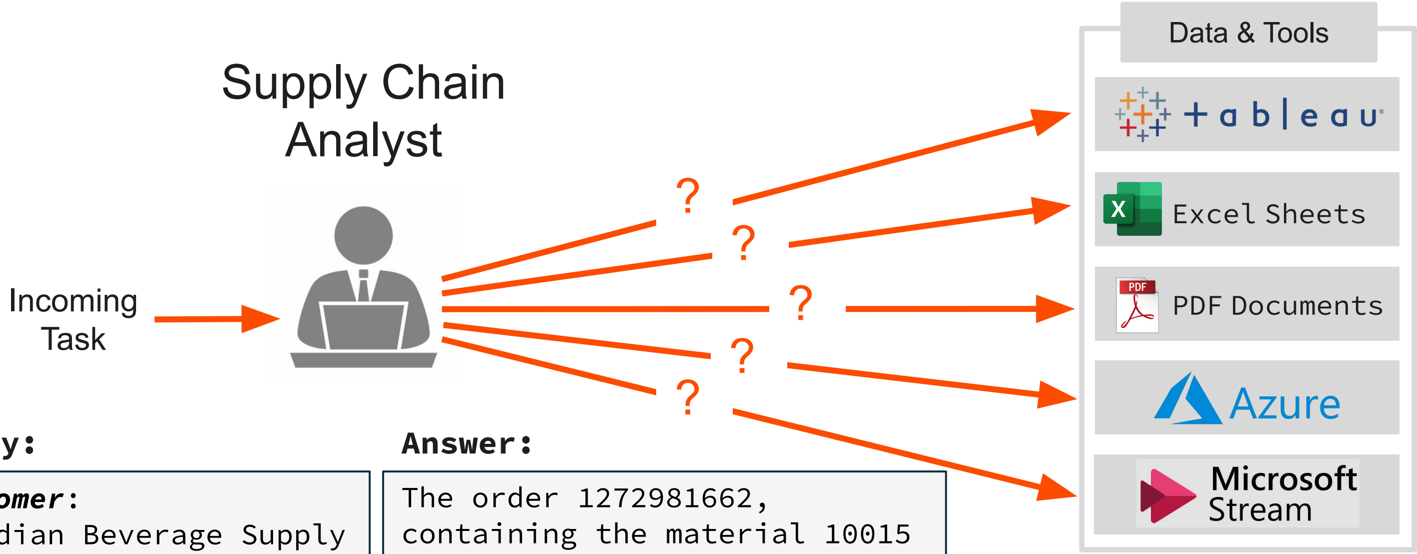
Application Example



Query:

```
Customer:
Canadian Beverage Supply
Order Number:
1272981662
Reported Issue:
Missing freshness req.
What to do?
```

Application Example



Query:

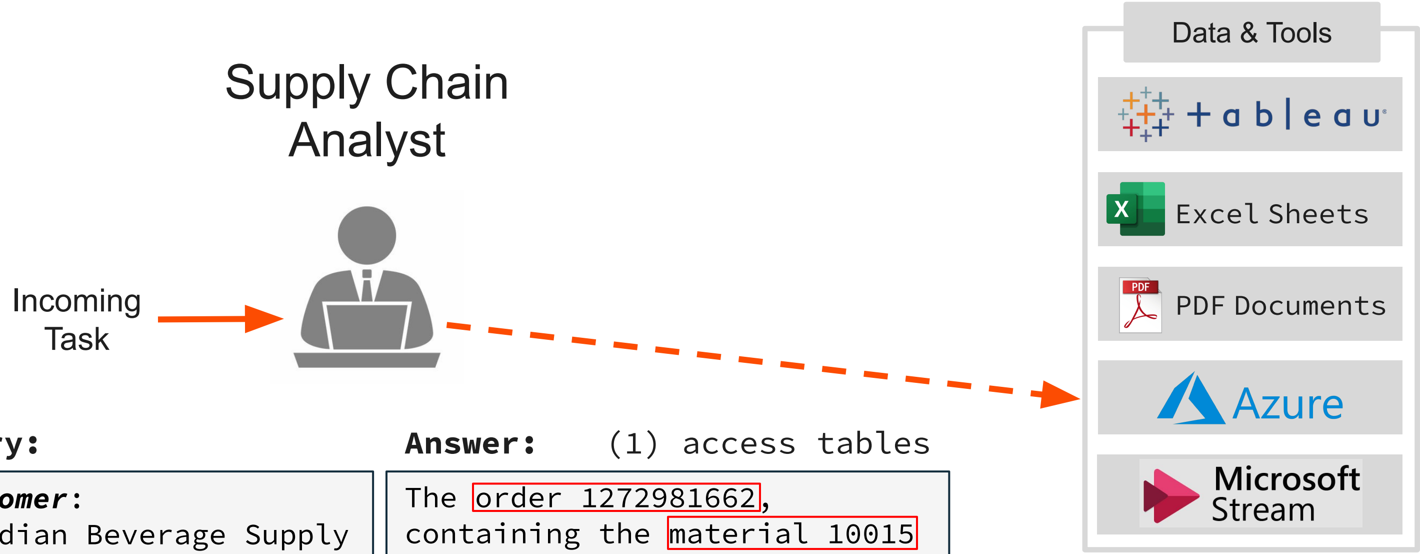
***Customer*:**
Canadian Beverage Supply
***Order Number*:**
1272981662
***Reported Issue*:**
Missing freshness req.
What to do?

Answer:

The order 1272981662, containing the material 10015 – '6CS BUTTER GARLIC', does not meet the freshness requirements. Please talk to the sales account manager.

Source: Freshness Tableau Dashboard

Application Example



Supply Chain Analyst

Incoming Task

Data & Tools

tableau

Excel Sheets

PDF Documents

Azure

Microsoft Stream

Query:

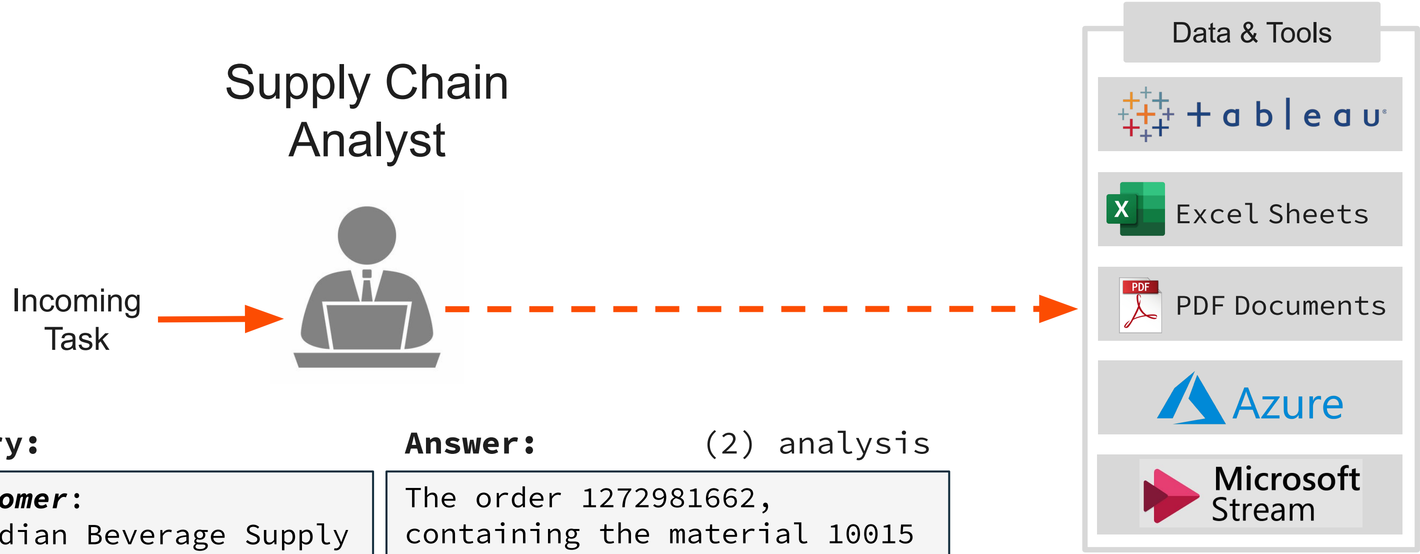
Customer:
Canadian Beverage Supply
Order Number:
1272981662
Reported Issue:
Missing freshness req.
What to do?

Answer: (1) access tables

The order 1272981662, containing the material 10015 – ‘6CS BUTTER GARLIC’, does not meet the freshness requirements. Please talk to the sales account manager.

Source: Freshness Tableau Dashboard

Application Example



Supply Chain Analyst

Incoming Task

Data & Tools

Query:

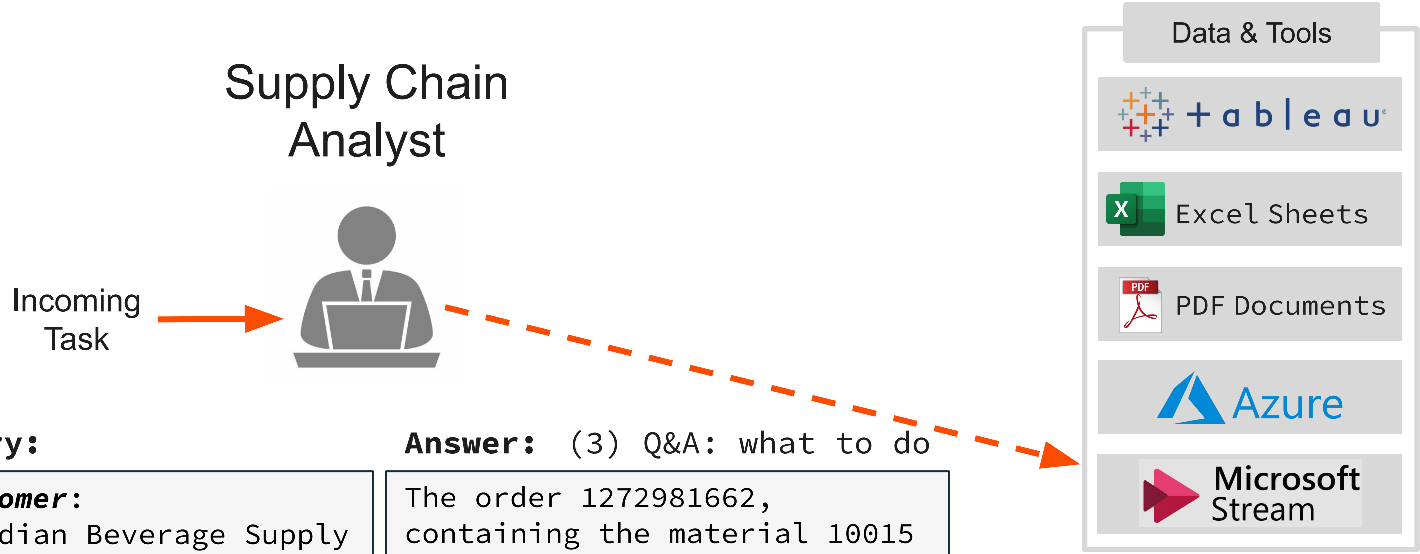
Customer:
Canadian Beverage Supply
Order Number:
1272981662
Reported Issue:
Missing freshness req.
What to do?

Answer: (2) analysis

The order 1272981662, containing the material 10015 – ‘6CS BUTTER GARLIC’, does not meet the freshness requirements. Please talk to the sales account manager.

Source: [Freshness Tableau Dashboard]

Application Example



Query:

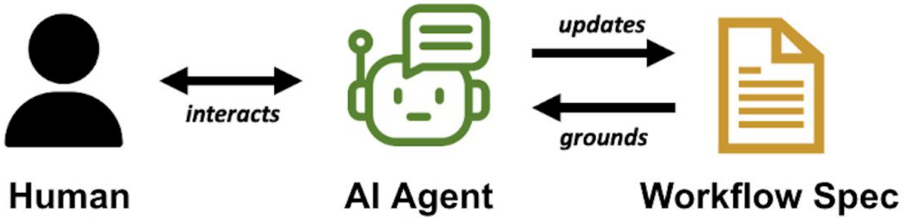
Customer: Canadian Beverage Supply
Order Number: 1272981662
Reported Issue: Missing freshness req. What to do?

Answer: (3) Q&A: what to do

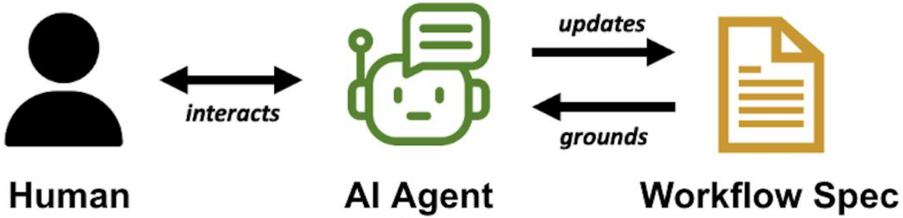
The order 1272981662, containing the material 10015 – ‘6CS BUTTER GARLIC’, does not meet the freshness requirements. Please talk to the sales account manager.

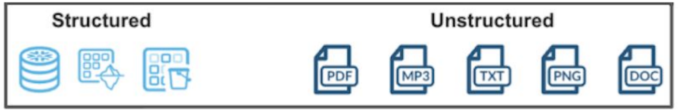
Source: Freshness Tableau Dashboard

New Workflow Implementations

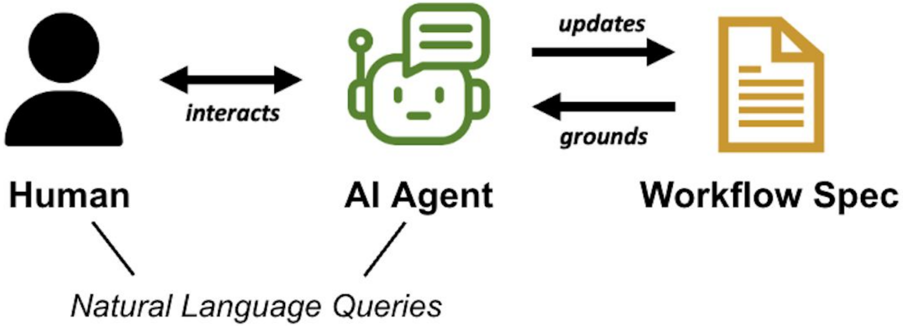


New Workflow Implementations



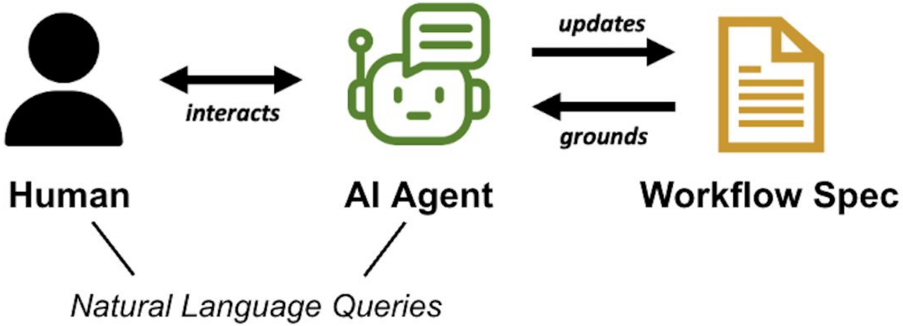


New Workflow Implementations



Structured
Unstructured
PDF
MP3
TXT
PNG
DOC

New Workflow Implementations



interacts

updates

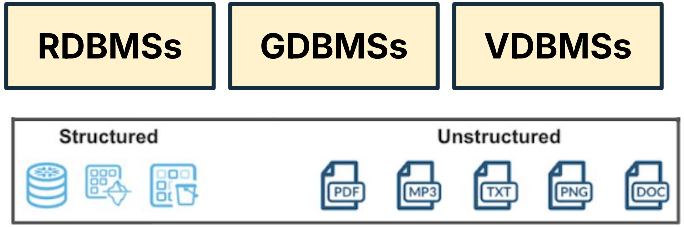
grounds

Human

AI Agent

Workflow Spec

Natural Language Queries



RDBMSs

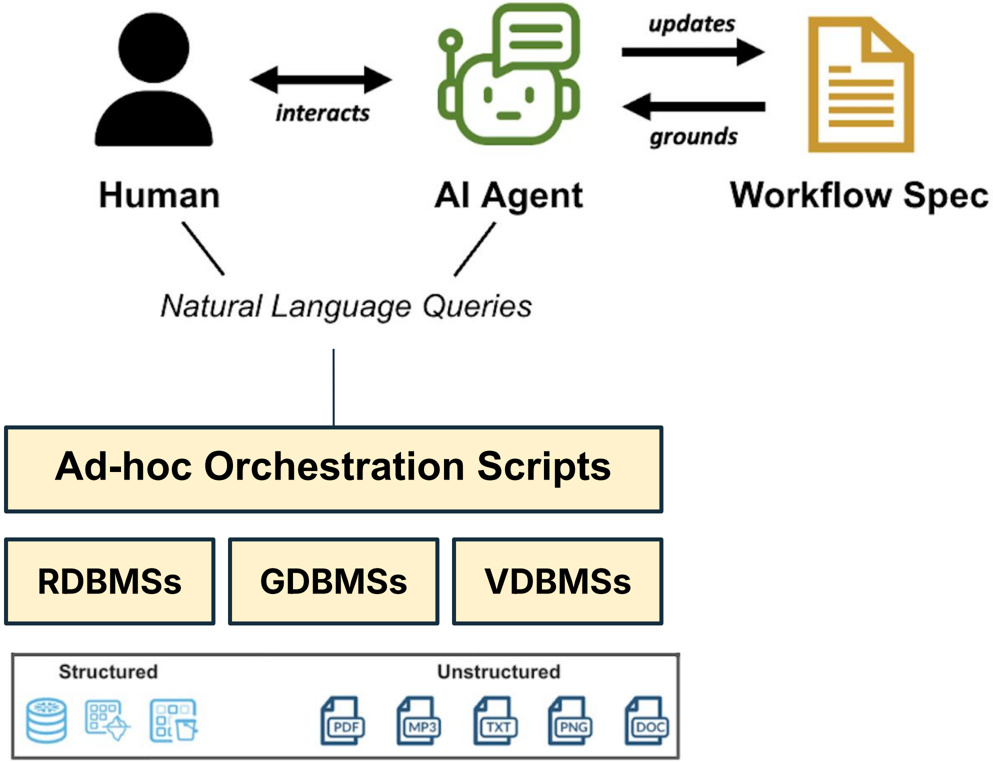
GDBMSs

VDBMSs

Structured

Unstructured

New Workflow Implementations



updates

interacts

grounds

Human

AI Agent

Workflow Spec

Natural Language Queries

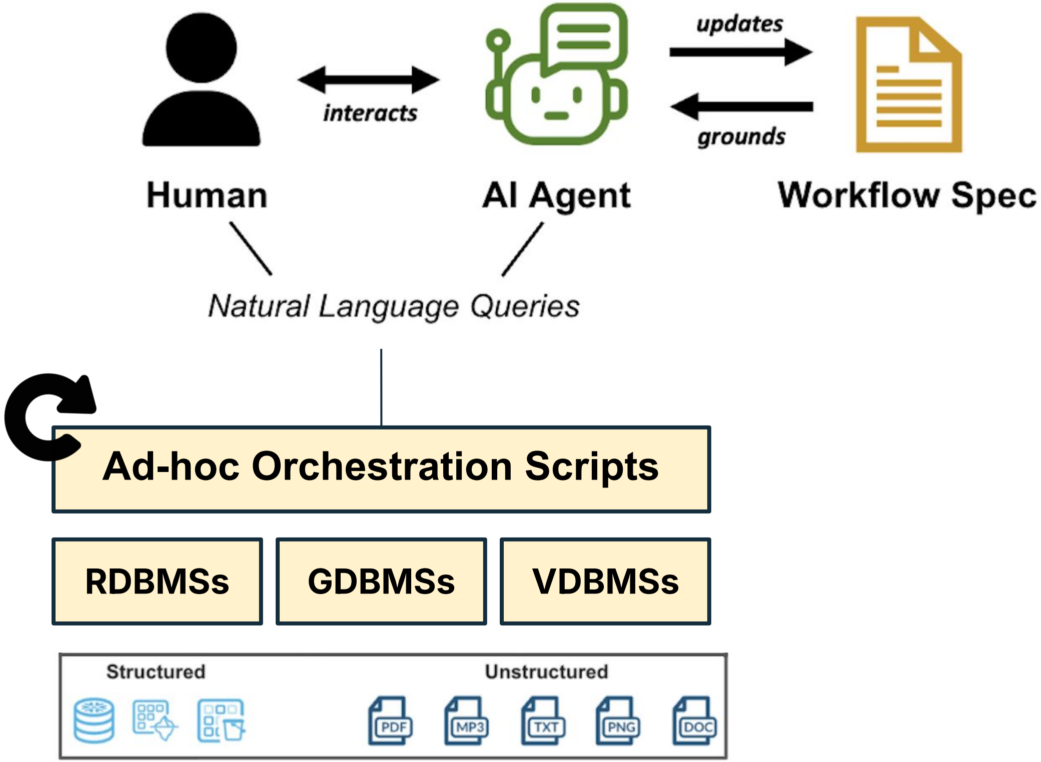
Ad-hoc Orchestration Scripts

RDBMSs

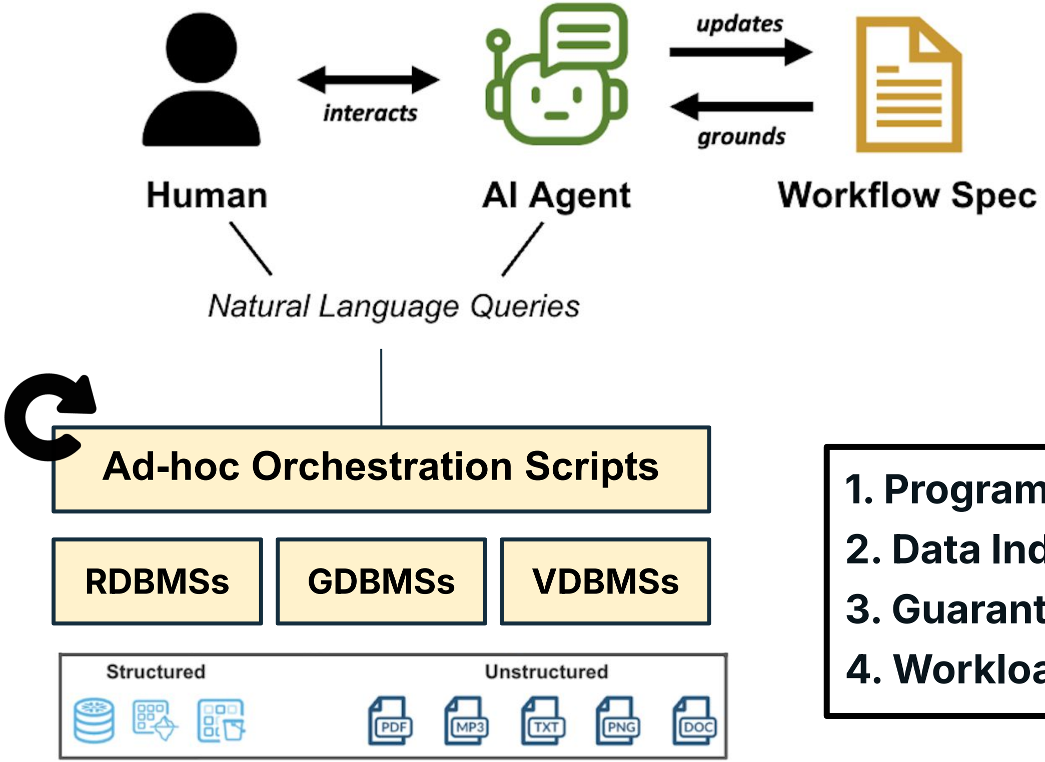
GDBMSs

VDBMSs

New Workflow Implementations



New Workflow Implementations



Human

interacts

AI Agent

updates

grounds

Workflow Spec

Natural Language Queries

Ad-hoc Orchestration Scripts

RDBMSs

GDBMSs

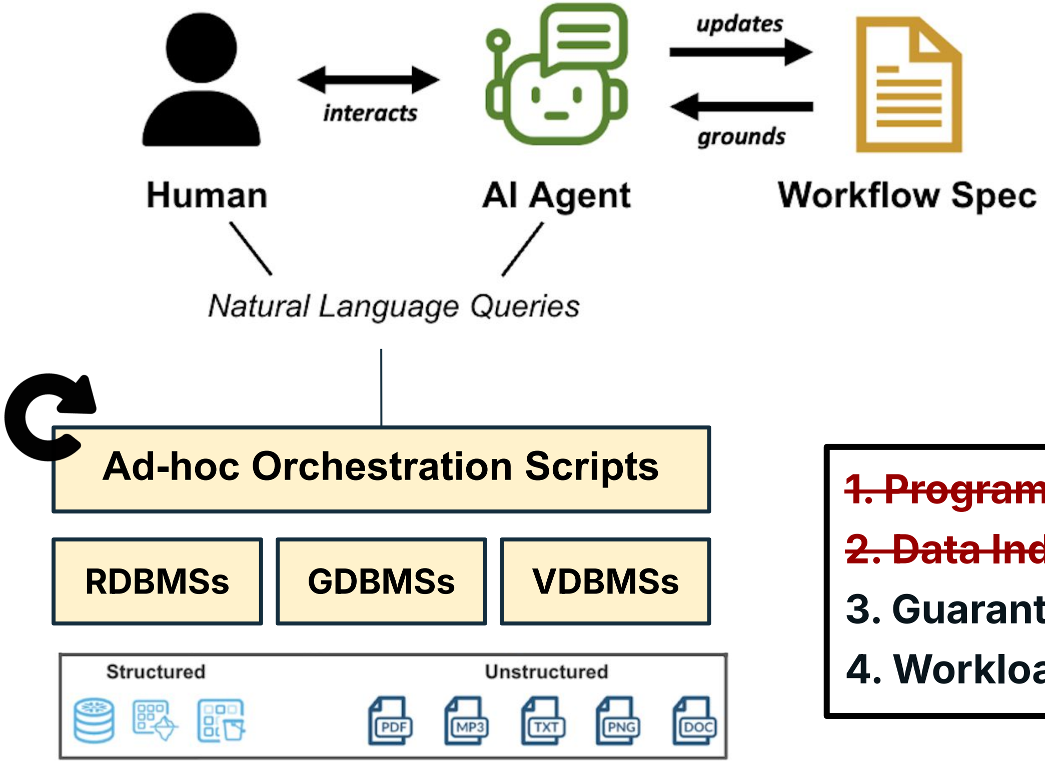
VDBMSs

Structured

Unstructured

1. Program Independence
2. Data Independence
3. Guarantees
4. Workload Specialization

New Workflow Implementations



Human

interacts

AI Agent

updates

grounds

Workflow Spec

Natural Language Queries

Ad-hoc Orchestration Scripts

RDBMSs

GDBMSs

VDBMSs

Structured

Unstructured

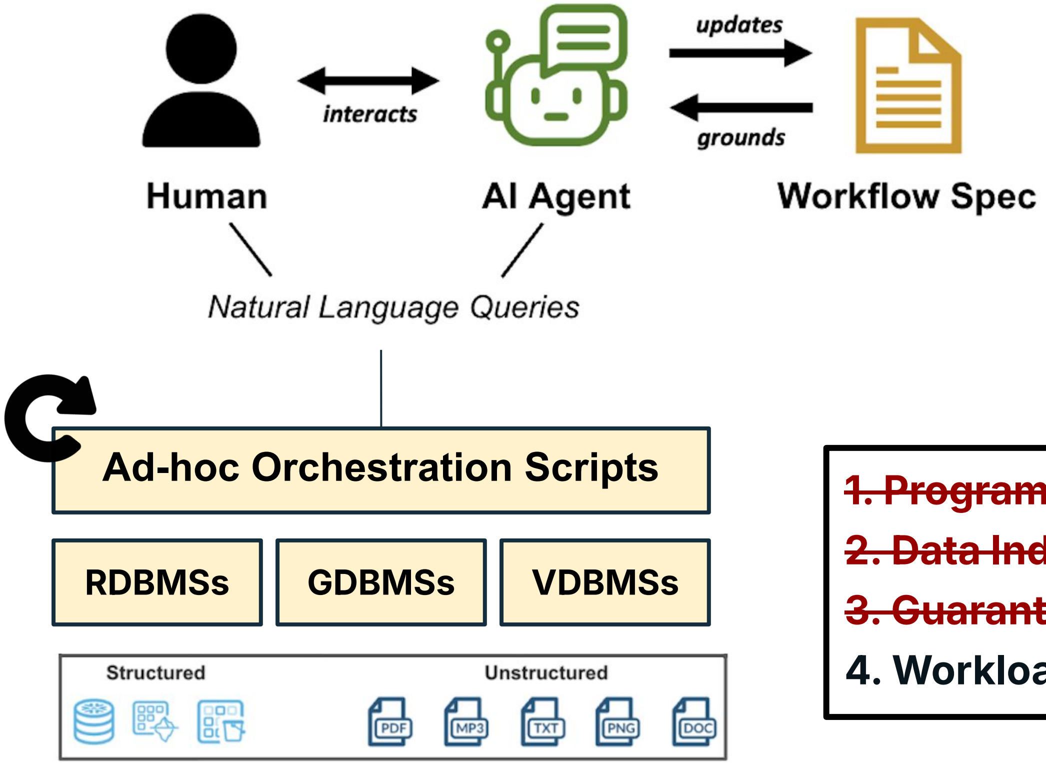
~~1. Program Independence~~

~~2. Data Independence~~

3. Guarantees

4. Workload Specialization

New Workflow Implementations



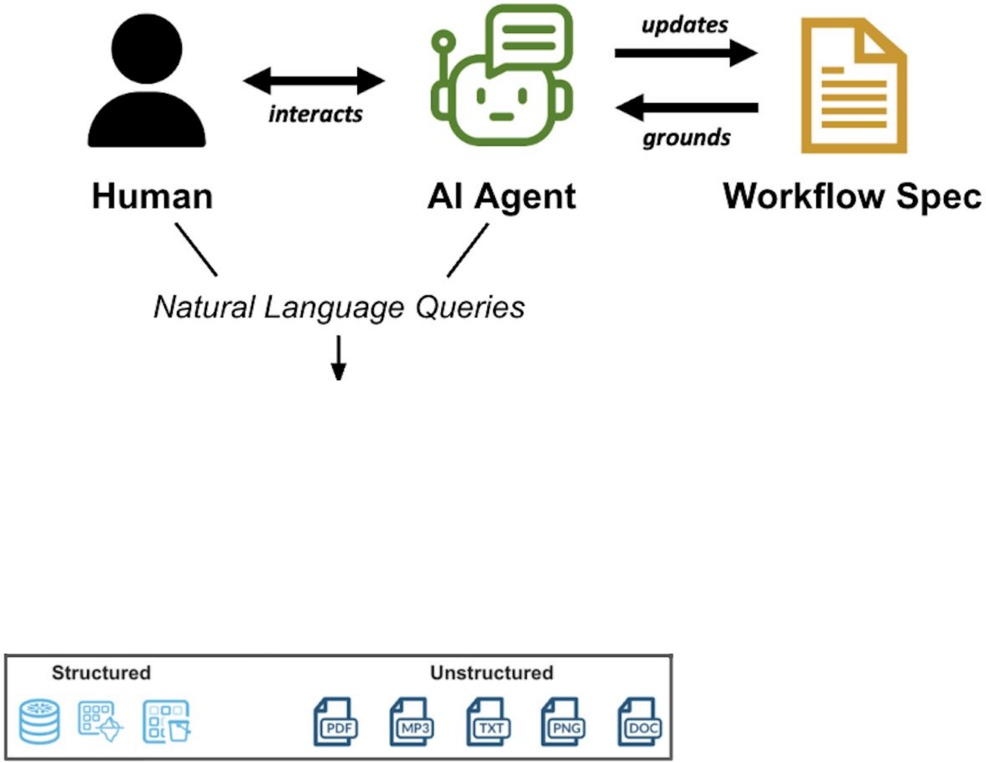
~~1. Program Independence~~

~~2. Data Independence~~

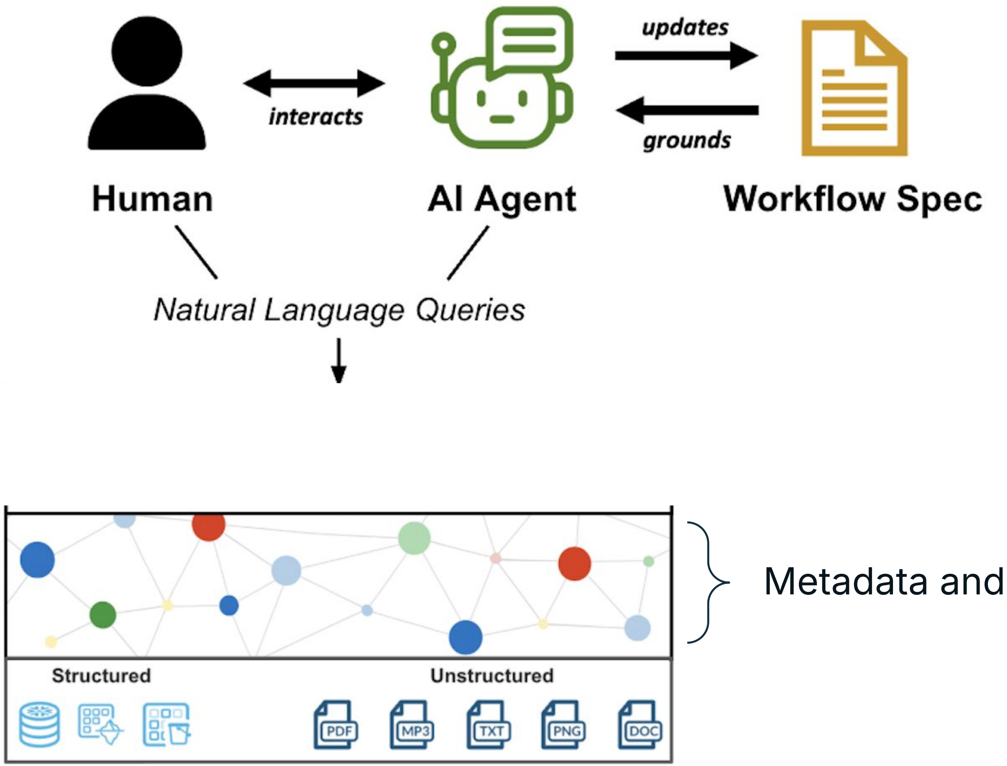
~~3. Guarantees~~

4. Workload Specialization

Vision

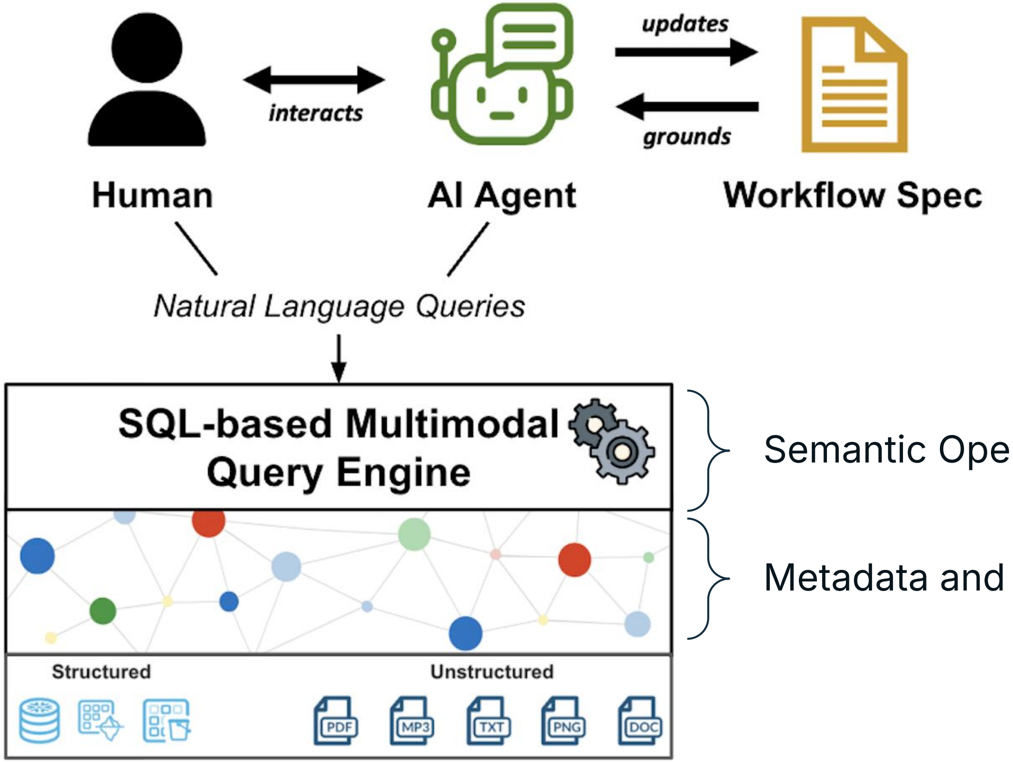

Human
interacts
AI Agent
updates
grounds
Workflow Spec
Natural Language Queries
Structured
Unstructured
PDF
MP3
TXT
PNG
DOC

Vision



Metadata and linking layer

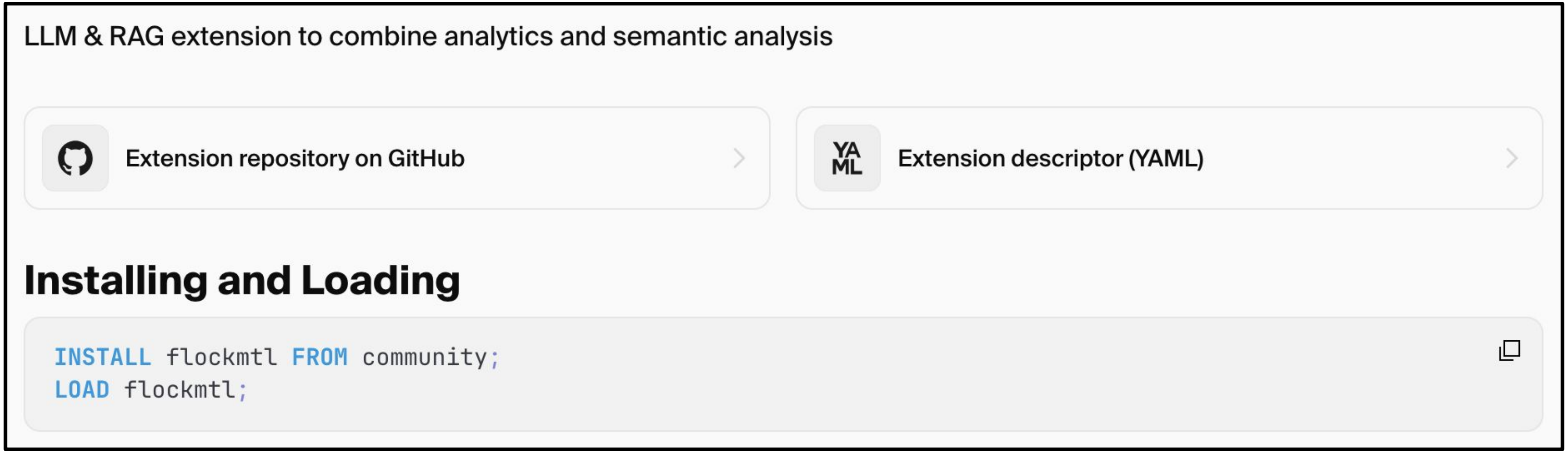
Vision



Semantic Operations

Metadata and linking layer

FlockMTL - LM Integration in RDBMSs



LLM & RAG extension to combine analytics and semantic analysis

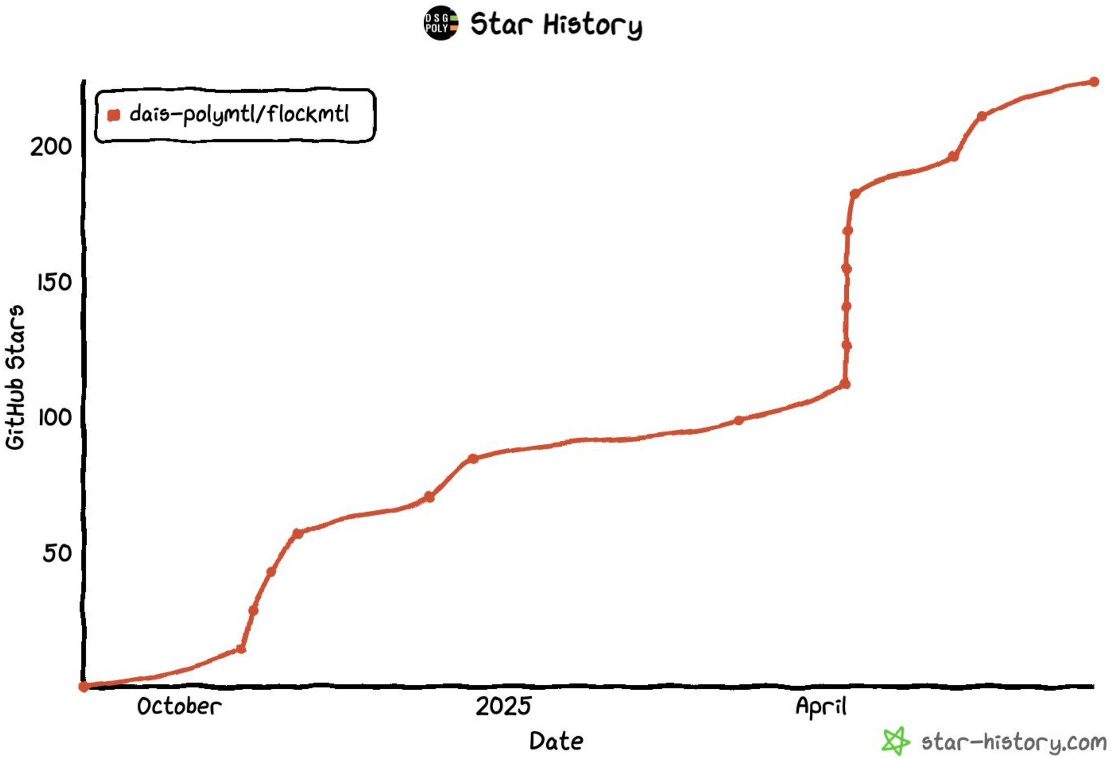
Extension repository on GitHub

Extension descriptor (YAML)

Installing and Loading

```
INSTALL flockmtl FROM community;
LOAD flockmtl;
```

FlockMTL - LM Integration in RDBMSs



FlockMTL Overview

FlockMTL Overview

Use a set of Scalar (“Map”) and Aggregate (“Reduce”) functions over a set of tuples or passages to implement various semantic operations within RDBMSs

FlockMTL Overview

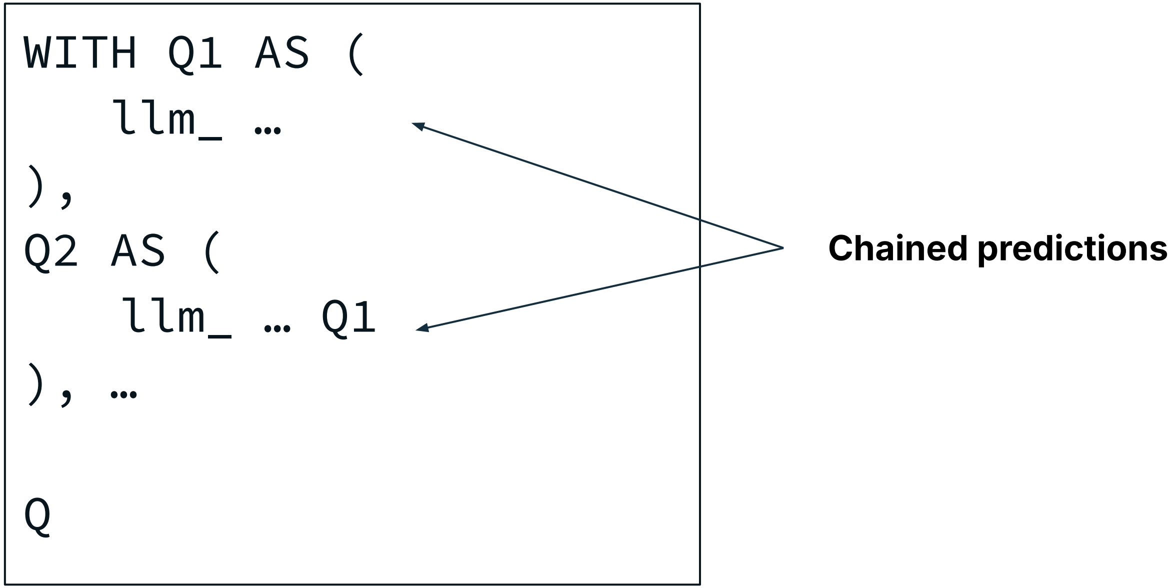
Scalar ("Map"):

```
llm_complete
llm_embedding
llm_filter
fusion
```

Aggregate ("Reduce")

```
llm_reduce
llm_reranker
llm_first
llm_last
```

Query Patterns

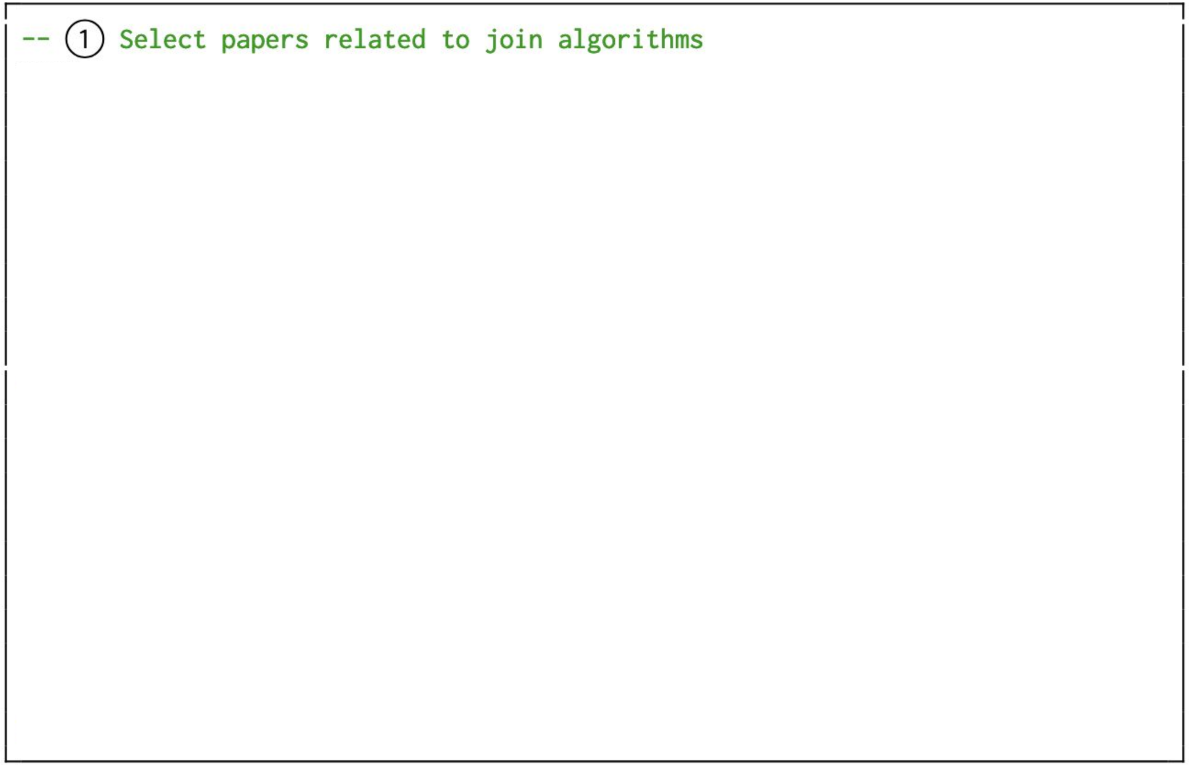


```
WITH Q1 AS (
   llm_ …
),
Q2 AS (
    llm_ … Q1
), …

Q
```

Chained predictions

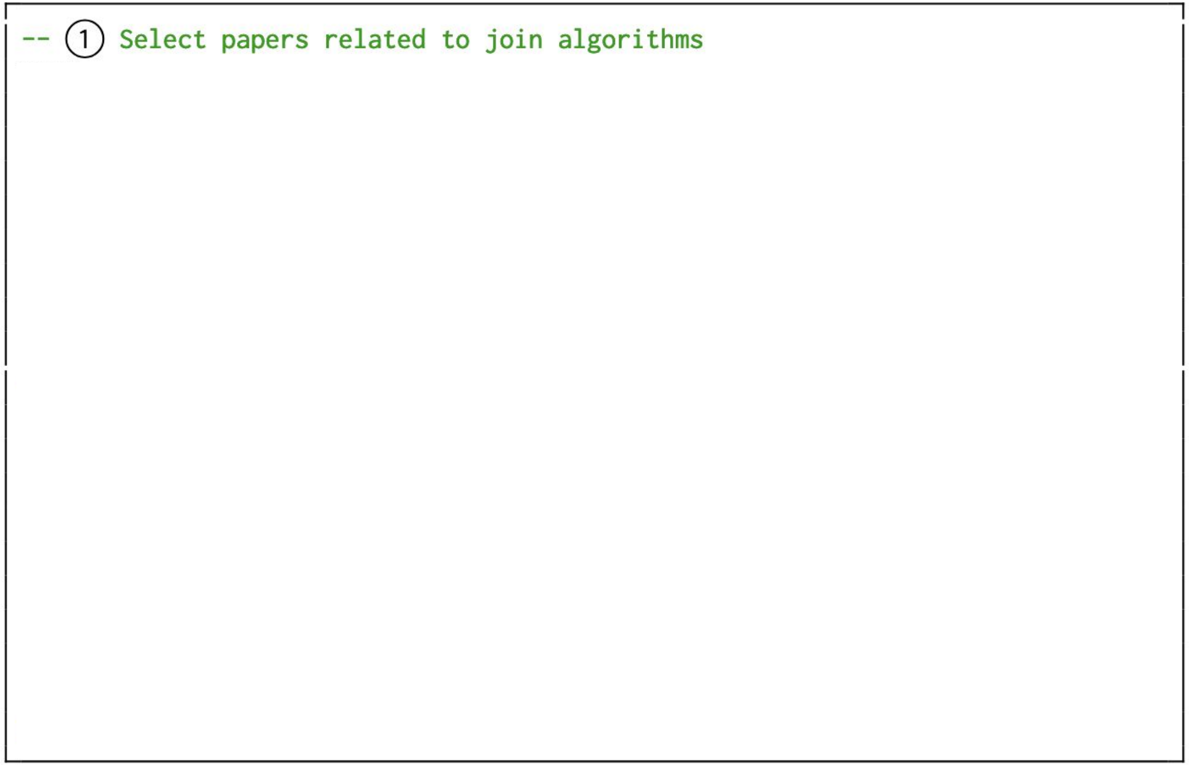
Query Examples (1)

Query Examples (1)

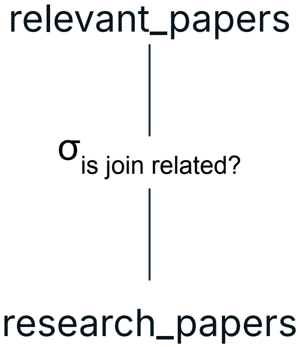


```
-- ① Select papers related to join algorithms
```

Query Examples (1)



```
-- ① Select papers related to join algorithms
```

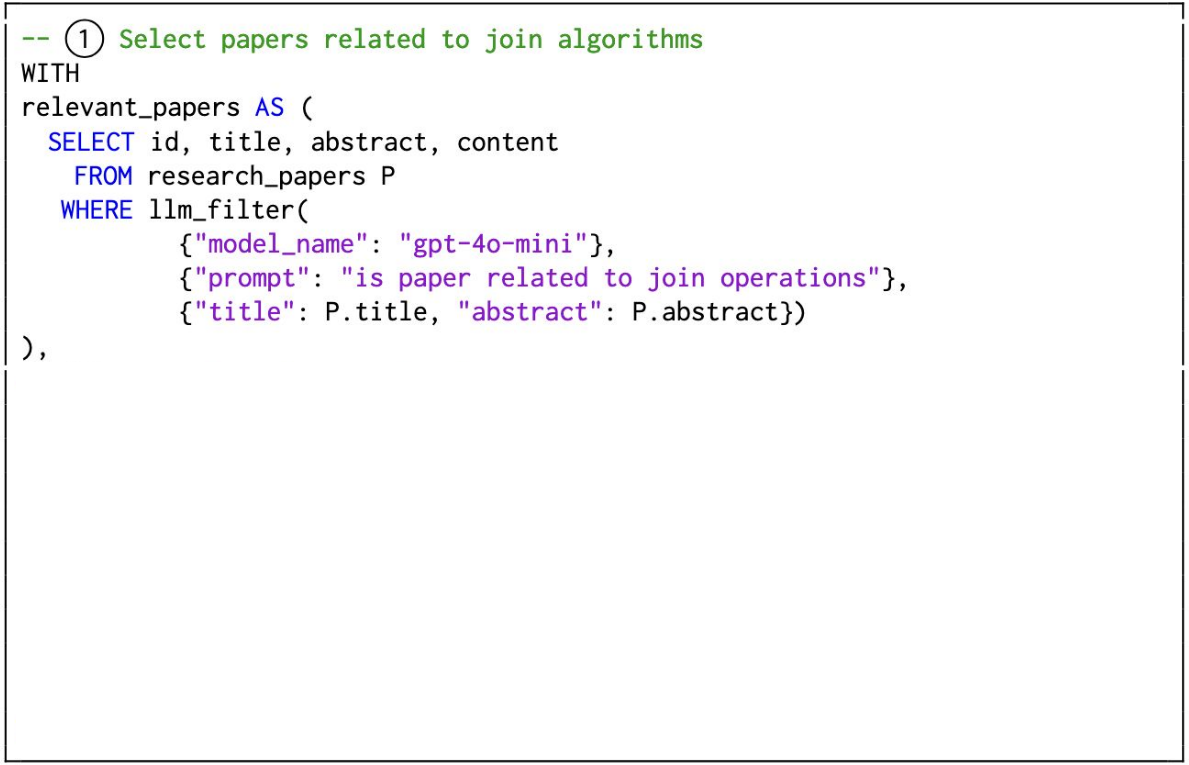


relevant_papers

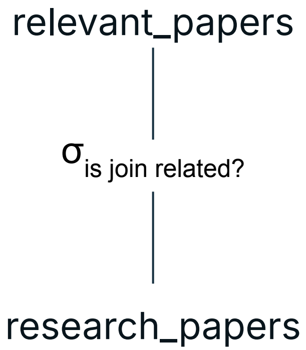
$\sigma_{\text{is join related?}}$

research_papers

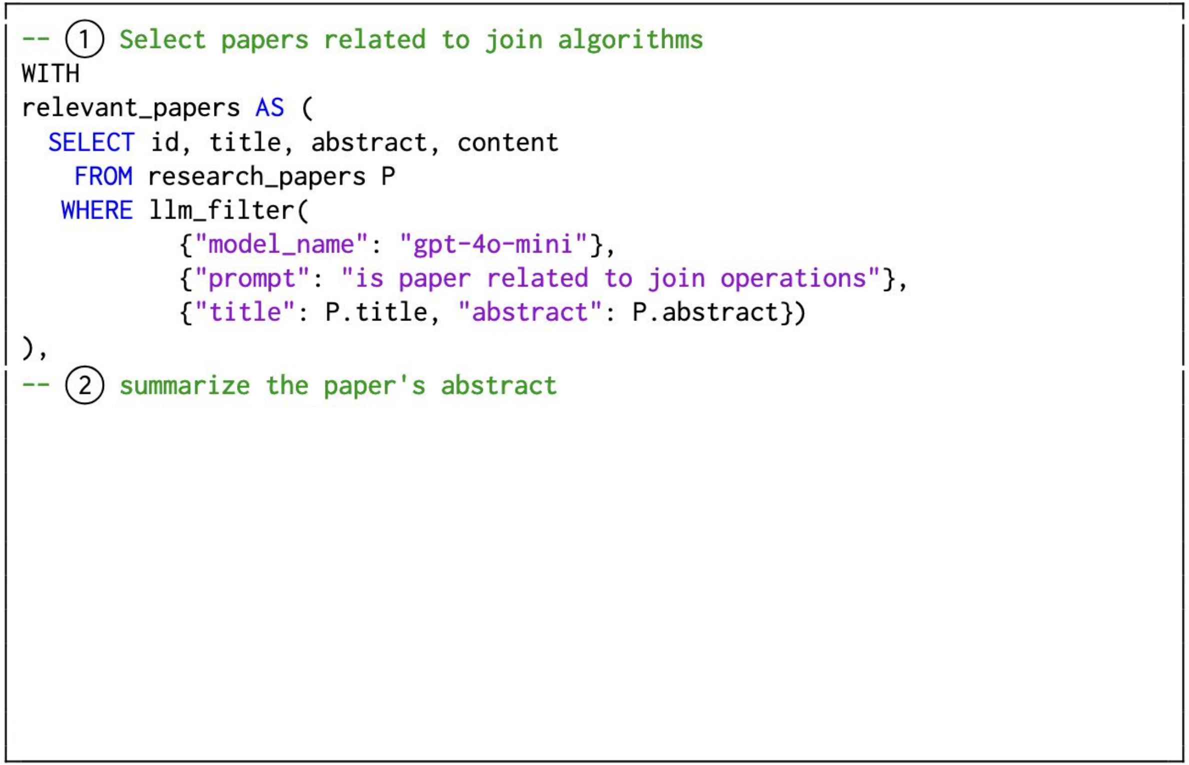
Query Examples (1)



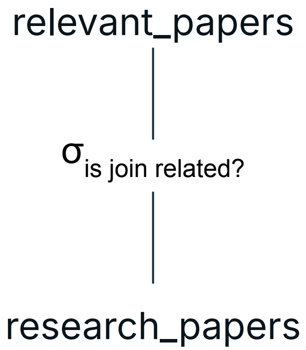
```
-- ① Select papers related to join algorithms
WITH
relevant_papers AS (
  SELECT id, title, abstract, content
    FROM research_papers P
   WHERE llm_filter(
           {"model_name": "gpt-4o-mini"},
           {"prompt": "is paper related to join operations"},
           {"title": P.title, "abstract": P.abstract})
),
```



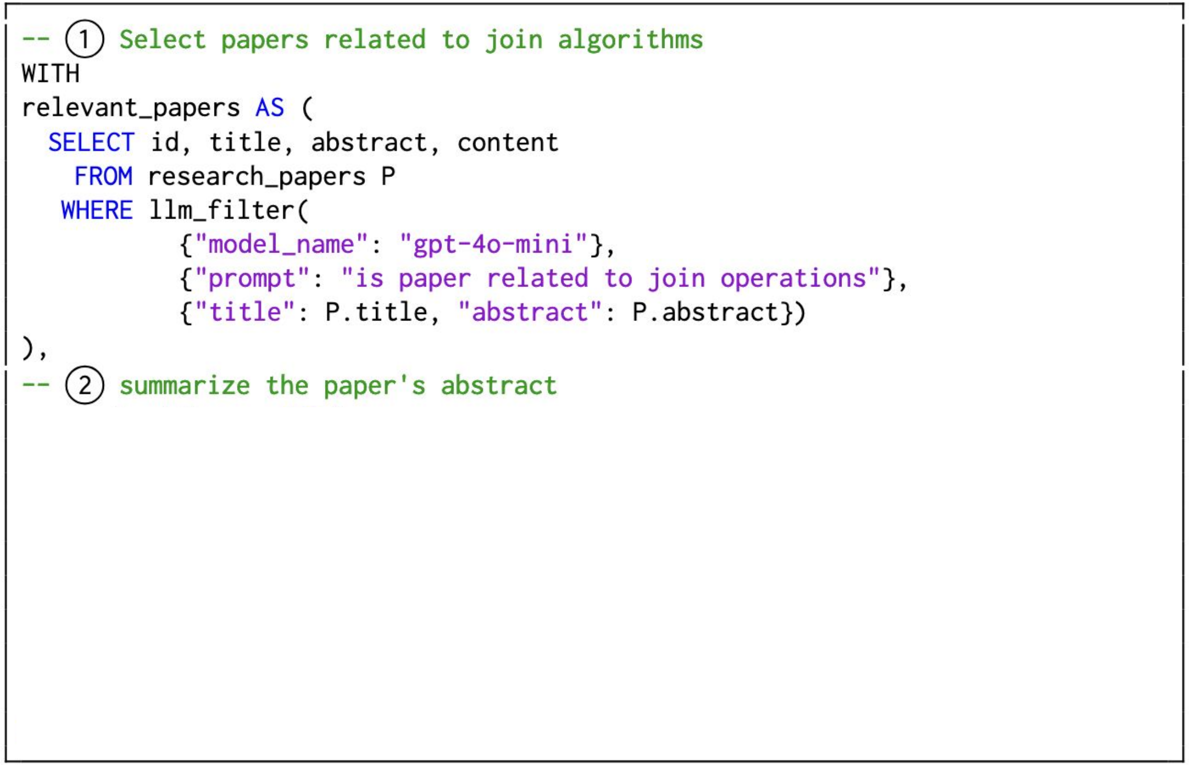
Query Examples (1)



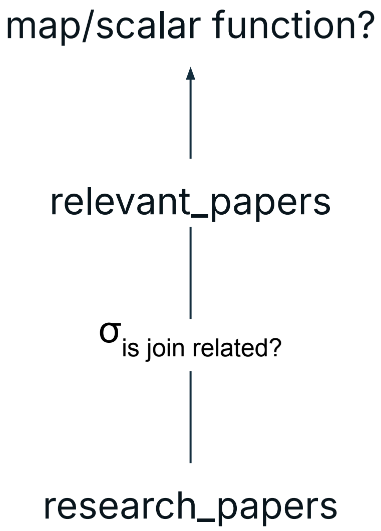
```
-- ① Select papers related to join algorithms
WITH
relevant_papers AS (
  SELECT id, title, abstract, content
    FROM research_papers P
   WHERE llm_filter(
           {"model_name": "gpt-4o-mini"},
           {"prompt": "is paper related to join operations"},
           {"title": P.title, "abstract": P.abstract})
),
-- ② summarize the paper's abstract
```



Query Examples (1)



```
-- ① Select papers related to join algorithms
WITH
relevant_papers AS (
  SELECT id, title, abstract, content
    FROM research_papers P
   WHERE llm_filter(
            {"model_name": "gpt-4o-mini"},
            {"prompt": "is paper related to join operations"},
            {"title": P.title, "abstract": P.abstract})
),
-- ② summarize the paper's abstract
```



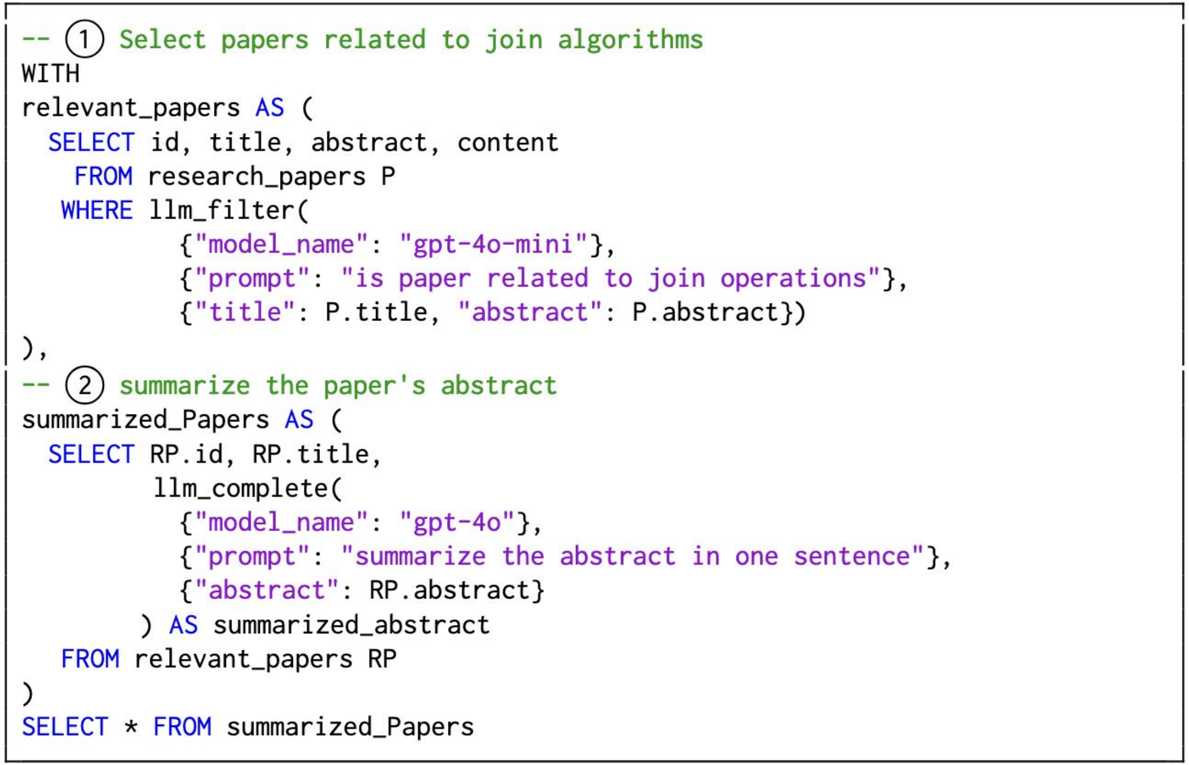
map/scalar function?

relevant_papers

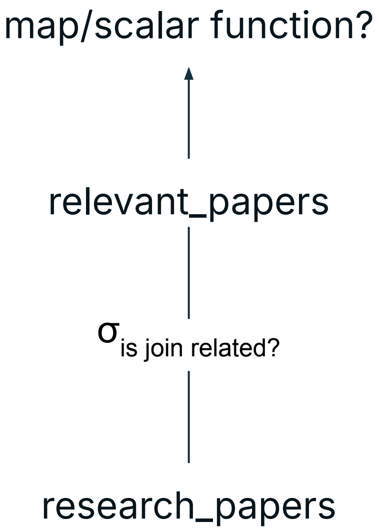
$\sigma_{\text{is join related?}}$

research_papers

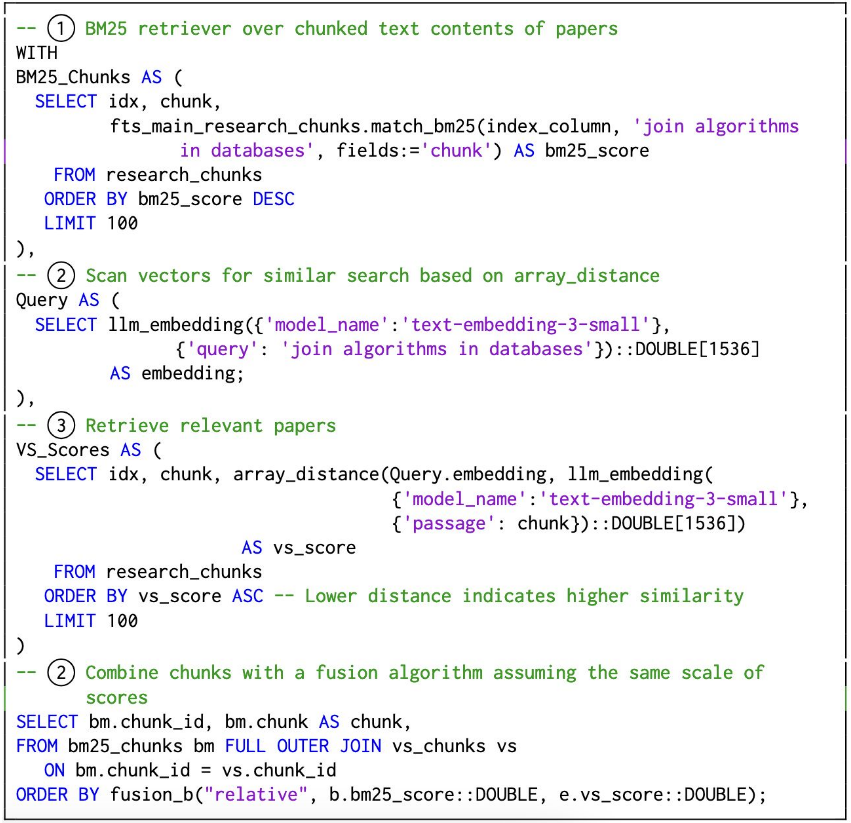
Query Examples (1)



```
-- ① Select papers related to join algorithms
WITH
relevant_papers AS (
  SELECT id, title, abstract, content
    FROM research_papers P
   WHERE llm_filter(
           {"model_name": "gpt-4o-mini"},
           {"prompt": "is paper related to join operations"},
           {"title": P.title, "abstract": P.abstract})
),
-- ② summarize the paper's abstract
summarized_Papers AS (
  SELECT RP.id, RP.title,
         llm_complete(
           {"model_name": "gpt-4o"},
           {"prompt": "summarize the abstract in one sentence"},
           {"abstract": RP.abstract}
        ) AS summarized_abstract
   FROM relevant_papers RP
)
SELECT * FROM summarized_Papers
```

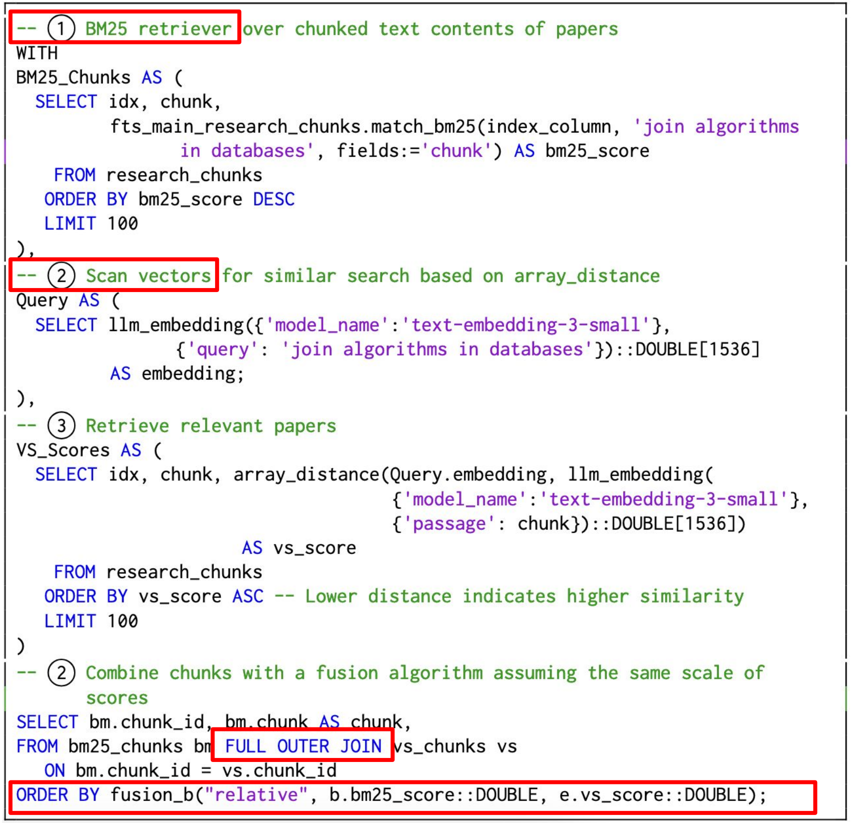


Query Examples (2)



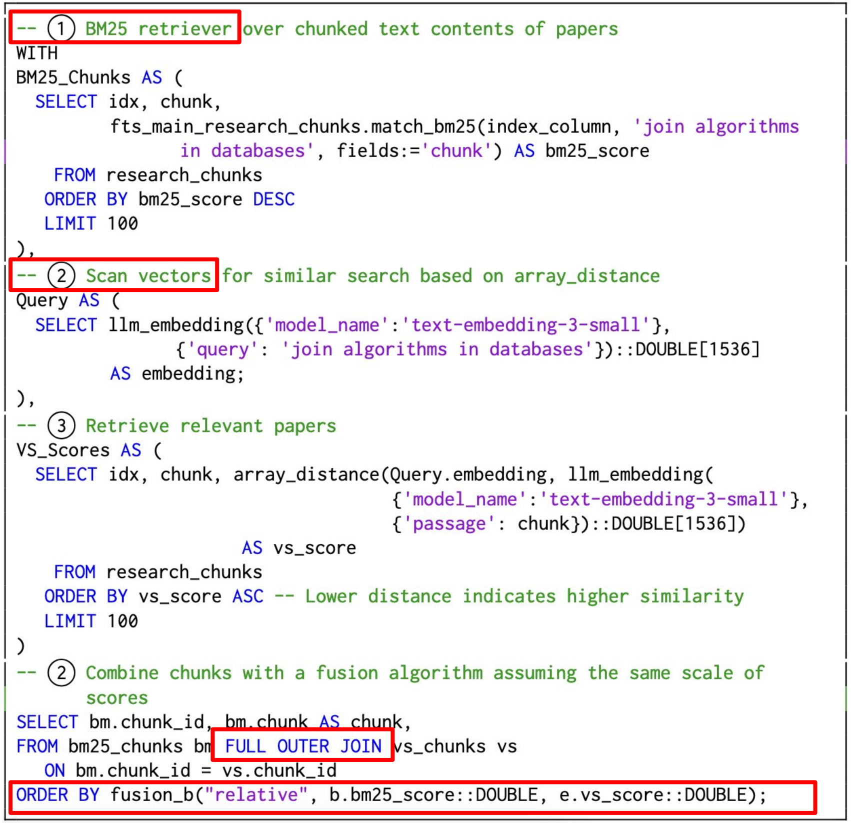
```
-- ① BM25 retriever over chunked text contents of papers
WITH
BM25_Chunks AS (
  SELECT idx, chunk,
         fts_main_research_chunks.match_bm25(index_column, 'join algorithms
                in databases', fields:='chunk') AS bm25_score
    FROM research_chunks
   ORDER BY bm25_score DESC
   LIMIT 100
),
-- ② Scan vectors for similar search based on array_distance
Query AS (
  SELECT llm_embedding({'model_name':'text-embedding-3-small'},
               {'query': 'join algorithms in databases'})::DOUBLE[1536]
         AS embedding;
),
-- ③ Retrieve relevant papers
VS_Scores AS (
  SELECT idx, chunk, array_distance(Query.embedding, llm_embedding(
                                    {'model_name':'text-embedding-3-small'},
                                    {'passage': chunk})::DOUBLE[1536])
                      AS vs_score
    FROM research_chunks
   ORDER BY vs_score ASC -- Lower distance indicates higher similarity
   LIMIT 100
)
-- ② Combine chunks with a fusion algorithm assuming the same scale of
       scores
SELECT bm.chunk_id, bm.chunk AS chunk,
FROM bm25_chunks bm FULL OUTER JOIN vs_chunks vs
   ON bm.chunk_id = vs.chunk_id
ORDER BY fusion_b("relative", b.bm25_score::DOUBLE, e.vs_score::DOUBLE);
```

Query Examples (2)

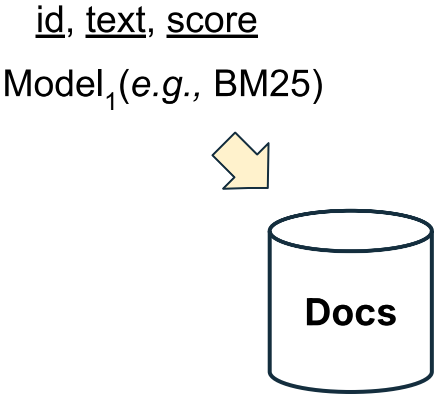


```
-- ① BM25 retriever over chunked text contents of papers
WITH
BM25_Chunks AS (
  SELECT idx, chunk,
         fts_main_research_chunks.match_bm25(index_column, 'join algorithms
                in databases', fields:='chunk') AS bm25_score
    FROM research_chunks
   ORDER BY bm25_score DESC
   LIMIT 100
),
-- ② Scan vectors for similar search based on array_distance
Query AS (
  SELECT llm_embedding({'model_name':'text-embedding-3-small'},
               {'query': 'join algorithms in databases'})::DOUBLE[1536]
         AS embedding;
),
-- ③ Retrieve relevant papers
VS_Scores AS (
  SELECT idx, chunk, array_distance(Query.embedding, llm_embedding(
                                    {'model_name':'text-embedding-3-small'},
                                    {'passage': chunk})::DOUBLE[1536])
                      AS vs_score
    FROM research_chunks
   ORDER BY vs_score ASC -- Lower distance indicates higher similarity
   LIMIT 100
)
-- ② Combine chunks with a fusion algorithm assuming the same scale of
   scores
SELECT bm.chunk_id, bm.chunk AS chunk,
FROM bm25_chunks bm FULL OUTER JOIN vs_chunks vs
   ON bm.chunk_id = vs.chunk_id
ORDER BY fusion_b("relative", b.bm25_score::DOUBLE, e.vs_score::DOUBLE);
```

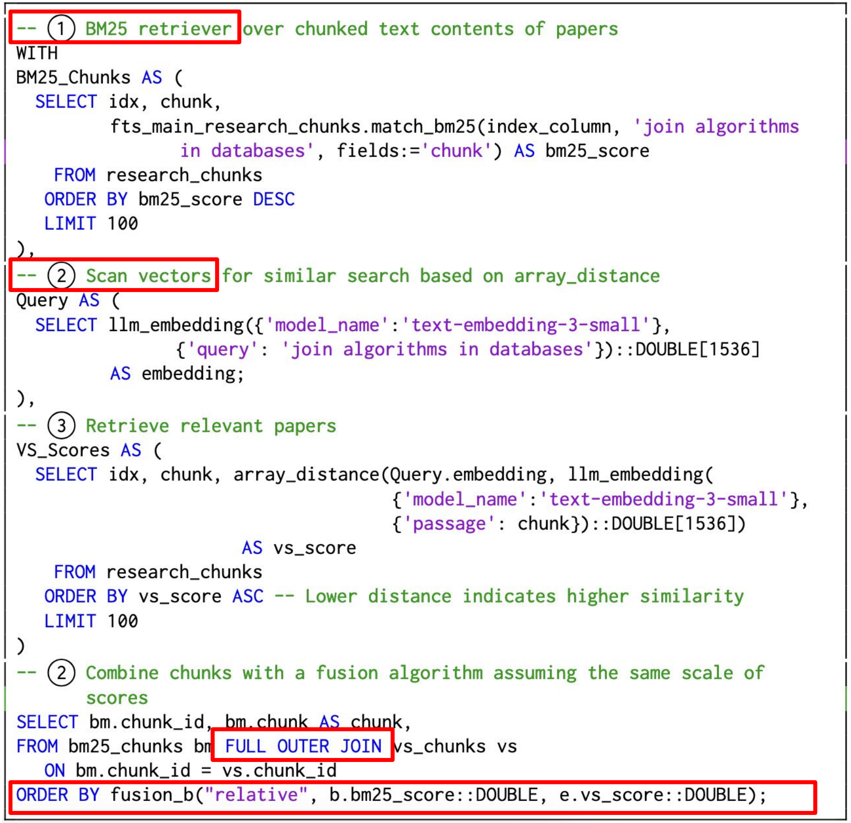
Query Examples (2)



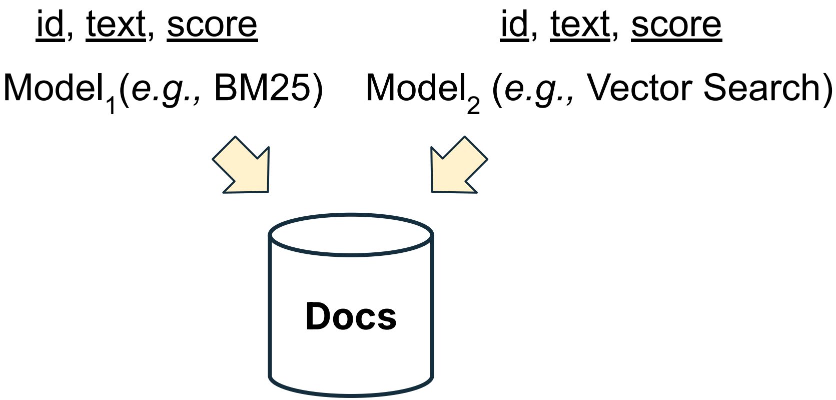
```
-- ① BM25 retriever over chunked text contents of papers
WITH
BM25_Chunks AS (
  SELECT idx, chunk,
         fts_main_research_chunks.match_bm25(index_column, 'join algorithms
                in databases', fields:='chunk') AS bm25_score
    FROM research_chunks
   ORDER BY bm25_score DESC
   LIMIT 100
),
-- ② Scan vectors for similar search based on array_distance
Query AS (
  SELECT llm_embedding({'model_name':'text-embedding-3-small'},
                {'query': 'join algorithms in databases'})::DOUBLE[1536]
         AS embedding;
),
-- ③ Retrieve relevant papers
VS_Scores AS (
  SELECT idx, chunk, array_distance(Query.embedding, llm_embedding(
                                    {'model_name':'text-embedding-3-small'},
                                    {'passage': chunk})::DOUBLE[1536])
                      AS vs_score
    FROM research_chunks
   ORDER BY vs_score ASC -- Lower distance indicates higher similarity
   LIMIT 100
)
-- ② Combine chunks with a fusion algorithm assuming the same scale of
   scores
SELECT bm.chunk_id, bm.chunk AS chunk,
FROM bm25_chunks bm FULL OUTER JOIN vs_chunks vs
   ON bm.chunk_id = vs.chunk_id
ORDER BY fusion_b("relative", b.bm25_score::DOUBLE, e.vs_score::DOUBLE);
```



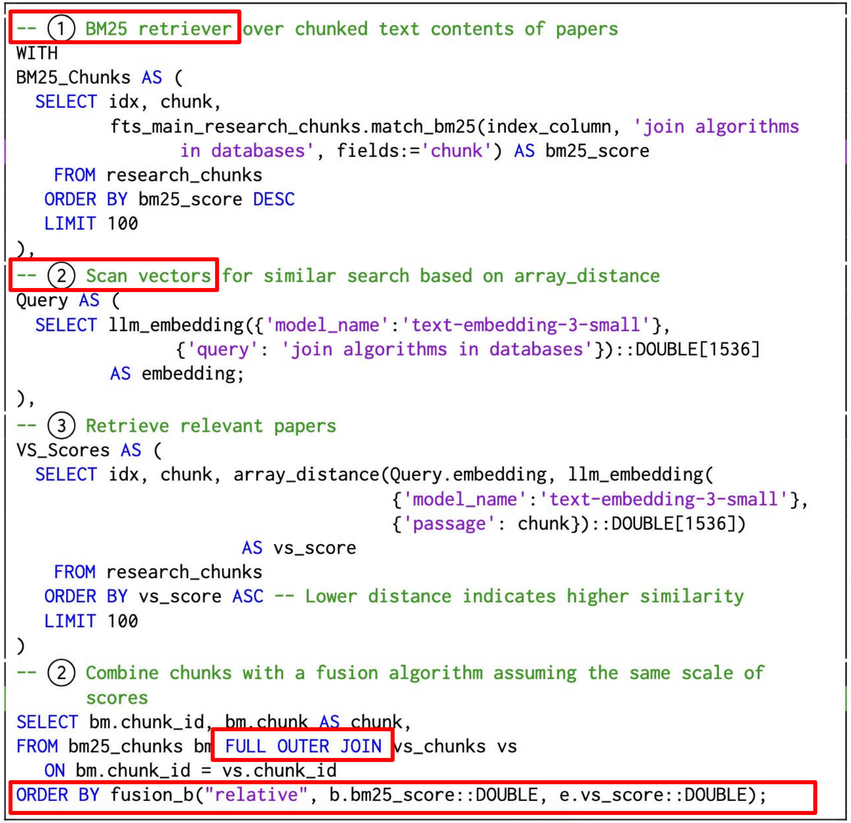
Query Examples (2)



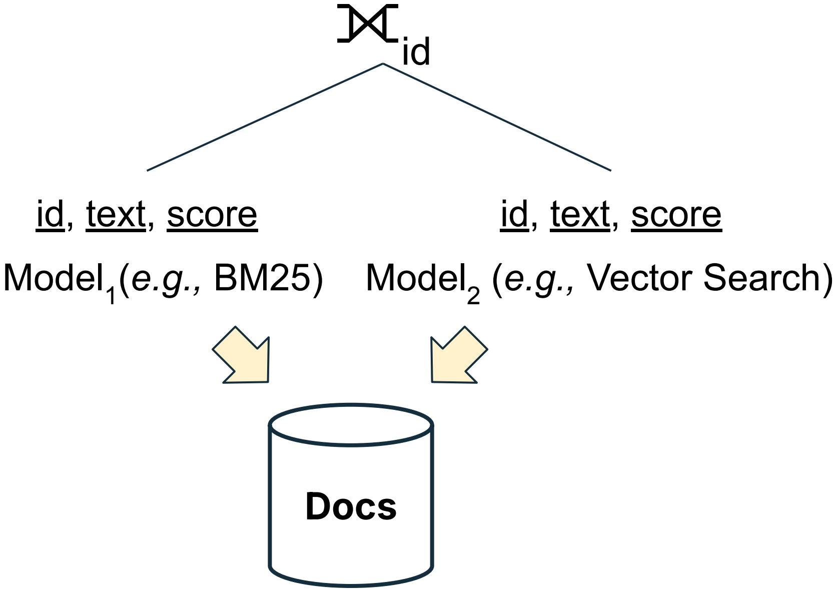
```
-- ① BM25 retriever over chunked text contents of papers
WITH
BM25_Chunks AS (
  SELECT idx, chunk,
         fts_main_research_chunks.match_bm25(index_column, 'join algorithms
                in databases', fields:='chunk') AS bm25_score
   FROM research_chunks
  ORDER BY bm25_score DESC
  LIMIT 100
),
-- ② Scan vectors for similar search based on array_distance
Query AS (
  SELECT llm_embedding({'model_name':'text-embedding-3-small'},
               {'query': 'join algorithms in databases'})::DOUBLE[1536]
         AS embedding;
),
-- ③ Retrieve relevant papers
VS_Scores AS (
  SELECT idx, chunk, array_distance(Query.embedding, llm_embedding(
                                          {'model_name':'text-embedding-3-small'},
                                          {'passage': chunk})::DOUBLE[1536])
                          AS vs_score
   FROM research_chunks
  ORDER BY vs_score ASC -- Lower distance indicates higher similarity
  LIMIT 100
)
-- ② Combine chunks with a fusion algorithm assuming the same scale of
   scores
SELECT bm.chunk_id, bm.chunk AS chunk,
FROM bm25_chunks bm FULL OUTER JOIN vs_chunks vs
  ON bm.chunk_id = vs.chunk_id
ORDER BY fusion_b("relative", b.bm25_score::DOUBLE, e.vs_score::DOUBLE);
```



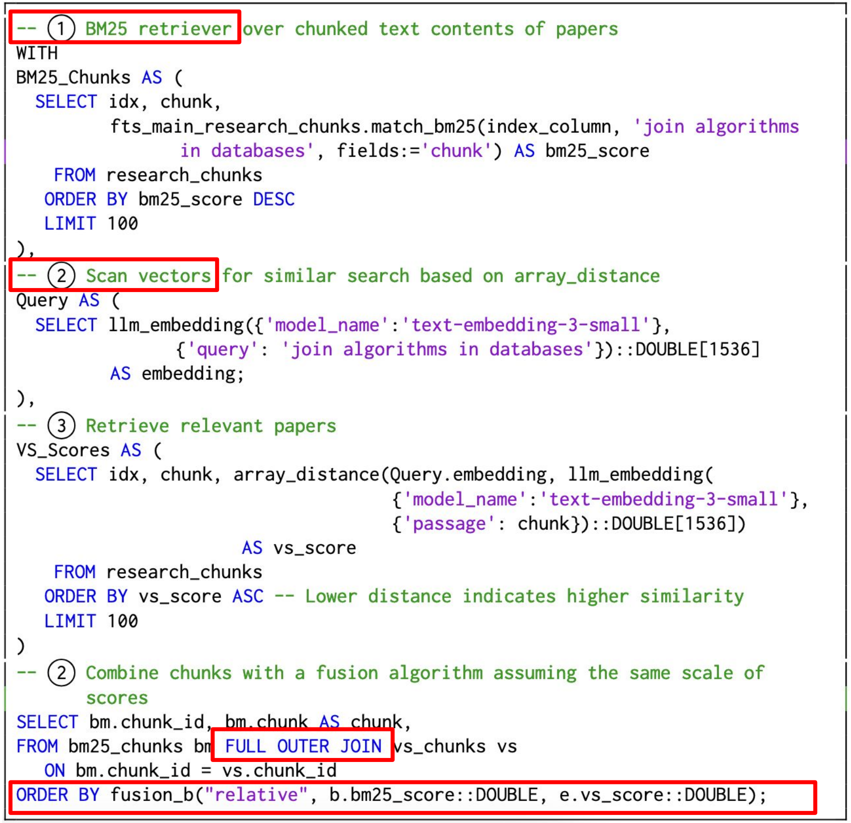
Query Examples (2)



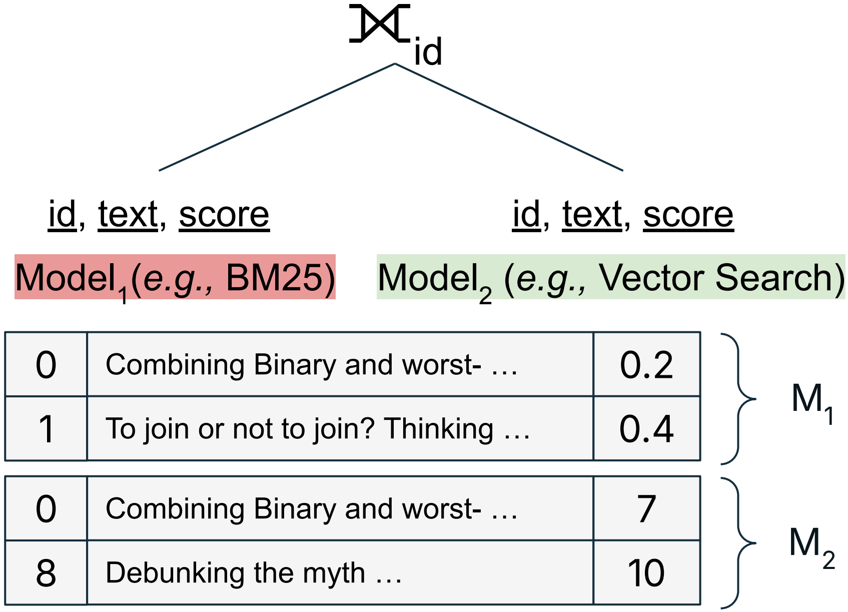
```
-- ① BM25 retriever over chunked text contents of papers
WITH
BM25_Chunks AS (
  SELECT idx, chunk,
         fts_main_research_chunks.match_bm25(index_column, 'join algorithms
                 in databases', fields:='chunk') AS bm25_score
    FROM research_chunks
   ORDER BY bm25_score DESC
   LIMIT 100
),
-- ② Scan vectors for similar search based on array_distance
Query AS (
  SELECT llm_embedding({'model_name':'text-embedding-3-small'},
                {'query': 'join algorithms in databases'})::DOUBLE[1536]
         AS embedding;
),
-- ③ Retrieve relevant papers
VS_Scores AS (
  SELECT idx, chunk, array_distance(Query.embedding, llm_embedding(
                                     {'model_name':'text-embedding-3-small'},
                                     {'passage': chunk})::DOUBLE[1536])
                     AS vs_score
    FROM research_chunks
   ORDER BY vs_score ASC -- Lower distance indicates higher similarity
   LIMIT 100
)
-- ② Combine chunks with a fusion algorithm assuming the same scale of
   scores
SELECT bm.chunk_id, bm.chunk AS chunk,
FROM bm25_chunks bm FULL OUTER JOIN vs_chunks vs
   ON bm.chunk_id = vs.chunk_id
ORDER BY fusion_b("relative", b.bm25_score::DOUBLE, e.vs_score::DOUBLE);
```



Query Examples (2)



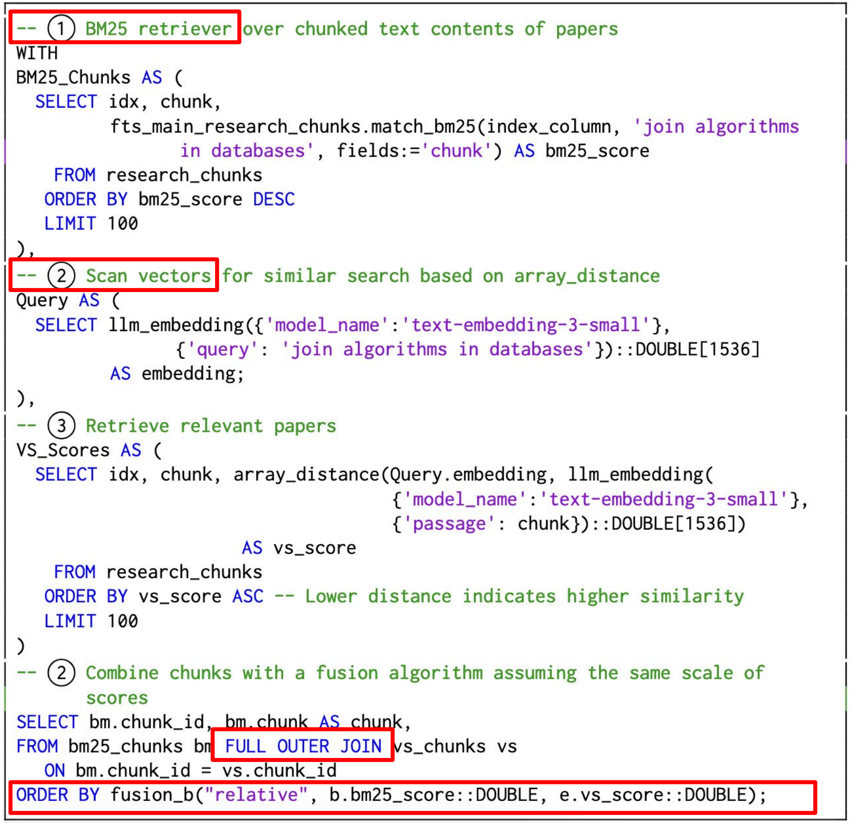
```
-- ① BM25 retriever over chunked text contents of papers
WITH
BM25_Chunks AS (
  SELECT idx, chunk,
          fts_main_research_chunks.match_bm25(index_column, 'join algorithms
                 in databases', fields:='chunk') AS bm25_score
    FROM research_chunks
   ORDER BY bm25_score DESC
   LIMIT 100
),
-- ② Scan vectors for similar search based on array_distance
Query AS (
  SELECT llm_embedding({'model_name':'text-embedding-3-small'},
                 {'query': 'join algorithms in databases'})::DOUBLE[1536]
          AS embedding;
),
-- ③ Retrieve relevant papers
VS_Scores AS (
  SELECT idx, chunk, array_distance(Query.embedding, llm_embedding(
                                       {'model_name':'text-embedding-3-small'},
                                       {'passage': chunk})::DOUBLE[1536])
                        AS vs_score
    FROM research_chunks
   ORDER BY vs_score ASC -- Lower distance indicates higher similarity
   LIMIT 100
)
-- ② Combine chunks with a fusion algorithm assuming the same scale of
      scores
SELECT bm.chunk_id, bm.chunk AS chunk,
FROM bm25_chunks bm FULL OUTER JOIN vs_chunks vs
   ON bm.chunk_id = vs.chunk_id
ORDER BY fusion_b("relative", b.bm25_score::DOUBLE, e.vs_score::DOUBLE);
```



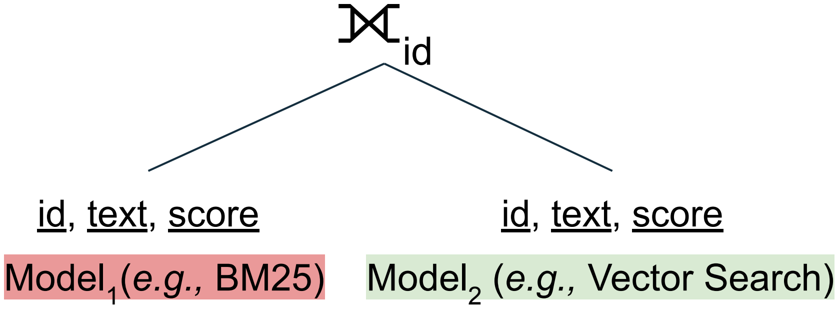
0	Combining Binary and worst- …	0.2
1	To join or not to join? Thinking …	0.4

0	Combining Binary and worst- …	7
8	Debunking the myth …	10

Query Examples (2)



```
-- ① BM25 retriever over chunked text contents of papers
WITH
BM25_Chunks AS (
  SELECT idx, chunk,
         fts_main_research_chunks.match_bm25(index_column, 'join algorithms
                 in databases', fields:='chunk') AS bm25_score
    FROM research_chunks
   ORDER BY bm25_score DESC
   LIMIT 100
),
-- ② Scan vectors for similar search based on array_distance
Query AS (
  SELECT llm_embedding({'model_name':'text-embedding-3-small'},
                {'query': 'join algorithms in databases'})::DOUBLE[1536]
         AS embedding;
),
-- ③ Retrieve relevant papers
VS_Scores AS (
  SELECT idx, chunk, array_distance(Query.embedding, llm_embedding(
                                          {'model_name':'text-embedding-3-small'},
                                          {'passage': chunk})::DOUBLE[1536])
                      AS vs_score
    FROM research_chunks
   ORDER BY vs_score ASC -- Lower distance indicates higher similarity
   LIMIT 100
)
-- ② Combine chunks with a fusion algorithm assuming the same scale of
       scores
SELECT bm.chunk_id, bm.chunk AS chunk,
FROM bm25_chunks bm FULL OUTER JOIN vs_chunks vs
   ON bm.chunk_id = vs.chunk_id
ORDER BY fusion_b("relative", b.bm25_score::DOUBLE, e.vs_score::DOUBLE);
```



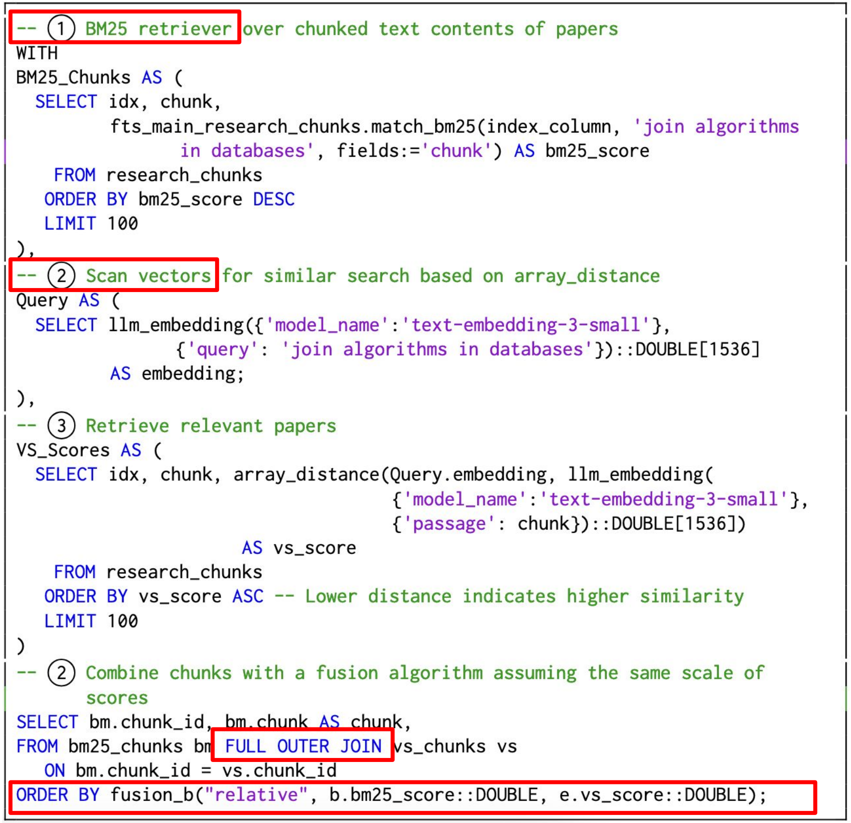
0	Combining Binary and worst- …	0.2
1	To join or not to join? Thinking …	0.4

M$_1$

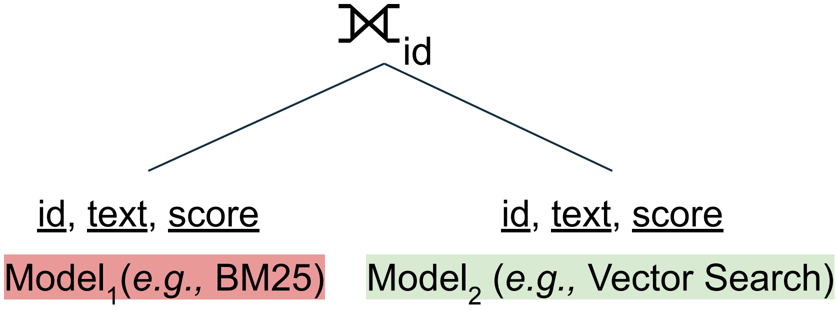
0	Combining Binary and worst- …	7
8	Debunking the myth …	10

M$_2$

Query Examples (2)



```
-- ① BM25 retriever over chunked text contents of papers
WITH
BM25_Chunks AS (
  SELECT idx, chunk,
         fts_main_research_chunks.match_bm25(index_column, 'join algorithms
                in databases', fields:='chunk') AS bm25_score
    FROM research_chunks
   ORDER BY bm25_score DESC
   LIMIT 100
),
-- ② Scan vectors for similar search based on array_distance
Query AS (
  SELECT llm_embedding({'model_name':'text-embedding-3-small'},
                {'query': 'join algorithms in databases'})::DOUBLE[1536]
         AS embedding;
),
-- ③ Retrieve relevant papers
VS_Scores AS (
  SELECT idx, chunk, array_distance(Query.embedding, llm_embedding(
                                        {'model_name':'text-embedding-3-small'},
                                        {'passage': chunk})::DOUBLE[1536])
                      AS vs_score
    FROM research_chunks
   ORDER BY vs_score ASC -- Lower distance indicates higher similarity
   LIMIT 100
)
-- ② Combine chunks with a fusion algorithm assuming the same scale of
   scores
SELECT bm.chunk_id, bm.chunk AS chunk,
FROM bm25_chunks bm FULL OUTER JOIN vs_chunks vs
   ON bm.chunk_id = vs.chunk_id
ORDER BY fusion_b("relative", b.bm25_score::DOUBLE, e.vs_score::DOUBLE);
```



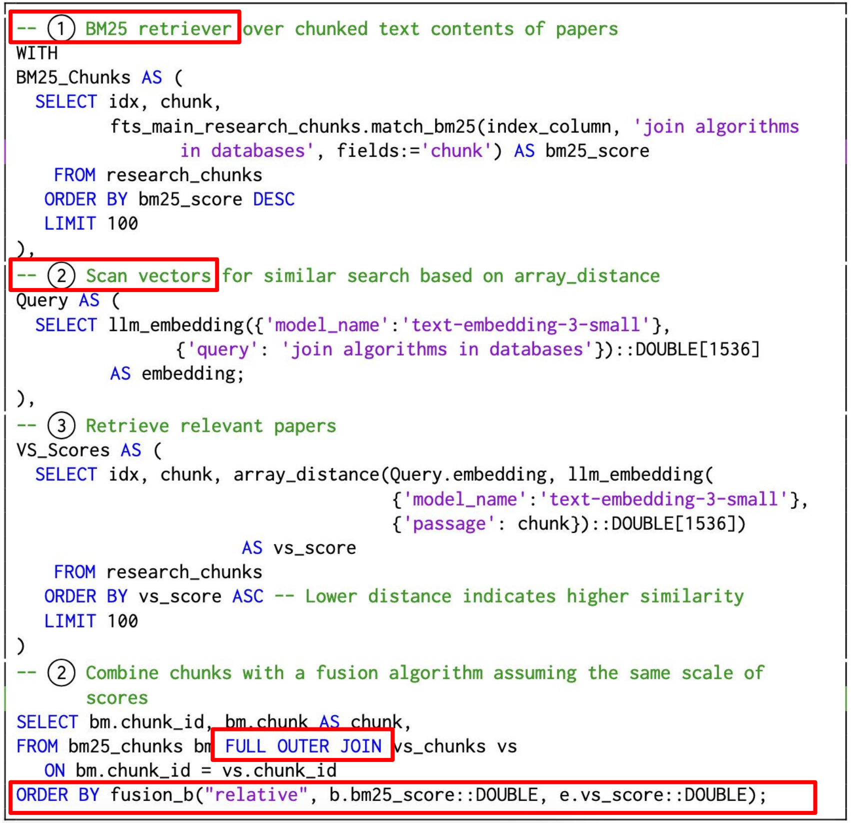
0	Combining Binary and worst- …	0.2
1	To join or not to join? Thinking …	0.4

M_1

0	Combining Binary and worst- …	7
8	Debunking the myth …	10

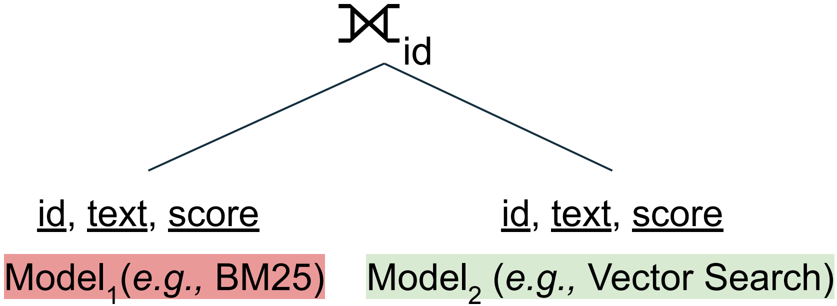
M_2

Query Examples (2)



```
-- ① BM25 retriever over chunked text contents of papers
WITH
BM25_Chunks AS (
  SELECT idx, chunk,
         fts_main_research_chunks.match_bm25(index_column, 'join algorithms
                in databases', fields:='chunk') AS bm25_score
   FROM research_chunks
  ORDER BY bm25_score DESC
  LIMIT 100
),
-- ② Scan vectors for similar search based on array_distance
Query AS (
  SELECT llm_embedding({'model_name':'text-embedding-3-small'},
               {'query': 'join algorithms in databases'})::DOUBLE[1536]
         AS embedding;
),
-- ③ Retrieve relevant papers
VS_Scores AS (
  SELECT idx, chunk, array_distance(Query.embedding, llm_embedding(
                                    {'model_name':'text-embedding-3-small'},
                                    {'passage': chunk})::DOUBLE[1536])
                     AS vs_score
   FROM research_chunks
  ORDER BY vs_score ASC -- Lower distance indicates higher similarity
  LIMIT 100
)
-- ② Combine chunks with a fusion algorithm assuming the same scale of
   scores
SELECT bm.chunk_id, bm.chunk AS chunk,
FROM bm25_chunks bm FULL OUTER JOIN vs_chunks vs
  ON bm.chunk_id = vs.chunk_id
ORDER BY fusion_b("relative", b.bm25_score::DOUBLE, e.vs_score::DOUBLE);
```

0	Combining Binary and worst- …	0.2	7



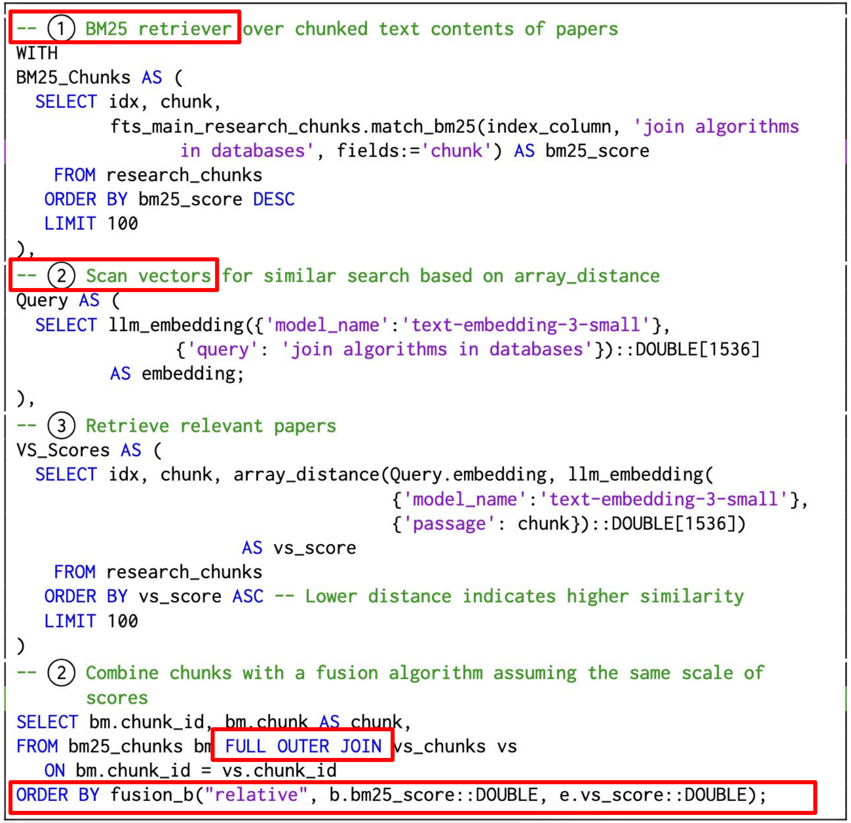
0	Combining Binary and worst- …	0.2
1	To join or not to join? Thinking …	0.4

M_1

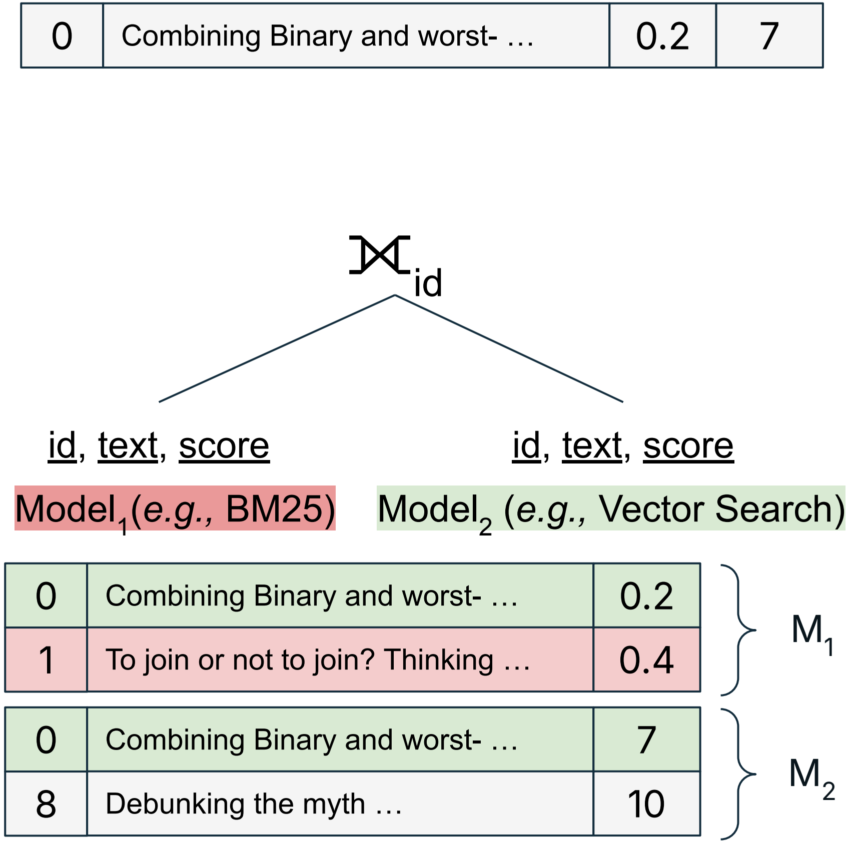
0	Combining Binary and worst- …	7
8	Debunking the myth …	10

M_2

Query Examples (2)



```
-- ① BM25 retriever over chunked text contents of papers
WITH
BM25_Chunks AS (
  SELECT idx, chunk,
         fts_main_research_chunks.match_bm25(index_column, 'join algorithms
                 in databases', fields:='chunk') AS bm25_score
   FROM research_chunks
  ORDER BY bm25_score DESC
  LIMIT 100
),
-- ② Scan vectors for similar search based on array_distance
Query AS (
  SELECT llm_embedding({'model_name':'text-embedding-3-small'},
                {'query': 'join algorithms in databases'})::DOUBLE[1536]
         AS embedding;
),
-- ③ Retrieve relevant papers
VS_Scores AS (
  SELECT idx, chunk, array_distance(Query.embedding, llm_embedding(
                                     {'model_name':'text-embedding-3-small'},
                                     {'passage': chunk})::DOUBLE[1536])
                   AS vs_score
   FROM research_chunks
  ORDER BY vs_score ASC -- Lower distance indicates higher similarity
  LIMIT 100
)
-- ② Combine chunks with a fusion algorithm assuming the same scale of
   scores
SELECT bm.chunk_id, bm.chunk AS chunk,
FROM bm25_chunks bm FULL OUTER JOIN vs_chunks vs
  ON bm.chunk_id = vs.chunk_id
ORDER BY fusion_b("relative", b.bm25_score::DOUBLE, e.vs_score::DOUBLE);
```



0	Combining Binary and worst- …	0.2	7

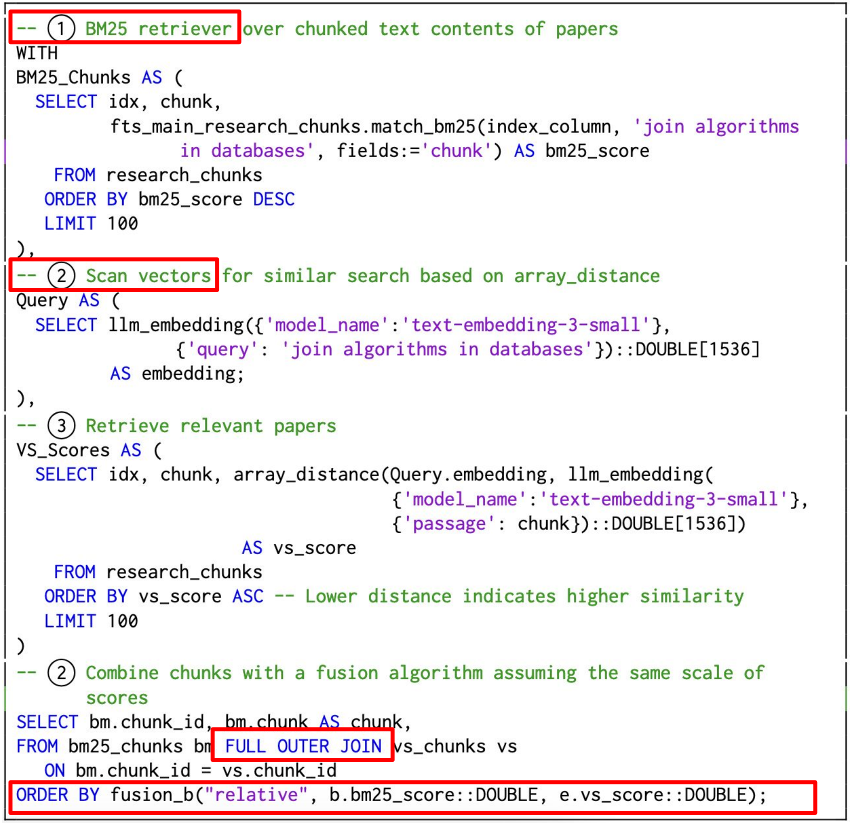
M1:

0	Combining Binary and worst- …	0.2
1	To join or not to join? Thinking …	0.4

M2:

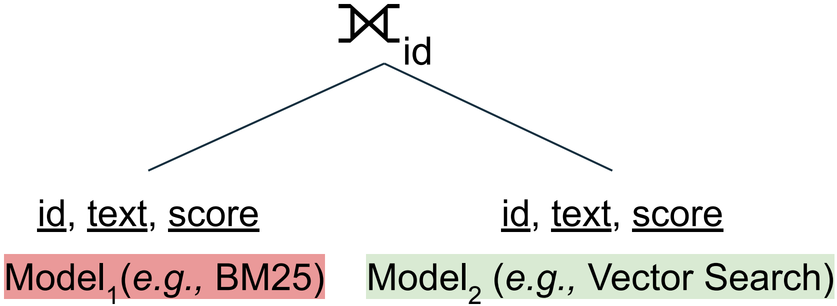
0	Combining Binary and worst- …	7
8	Debunking the myth …	10

Query Examples (2)



```
-- ① BM25 retriever over chunked text contents of papers
WITH
BM25_Chunks AS (
  SELECT idx, chunk,
         fts_main_research_chunks.match_bm25(index_column, 'join algorithms
                 in databases', fields:='chunk') AS bm25_score
    FROM research_chunks
   ORDER BY bm25_score DESC
   LIMIT 100
),
-- ② Scan vectors for similar search based on array_distance
Query AS (
  SELECT llm_embedding({'model_name':'text-embedding-3-small'},
                {'query': 'join algorithms in databases'})::DOUBLE[1536]
         AS embedding;
),
-- ③ Retrieve relevant papers
VS_Scores AS (
  SELECT idx, chunk, array_distance(Query.embedding, llm_embedding(
                                   {'model_name':'text-embedding-3-small'},
                                   {'passage': chunk})::DOUBLE[1536])
                     AS vs_score
    FROM research_chunks
   ORDER BY vs_score ASC -- Lower distance indicates higher similarity
   LIMIT 100
)
-- ② Combine chunks with a fusion algorithm assuming the same scale of
      scores
SELECT bm.chunk_id, bm.chunk AS chunk,
FROM bm25_chunks bm FULL OUTER JOIN vs_chunks vs
   ON bm.chunk_id = vs.chunk_id
ORDER BY fusion_b("relative", b.bm25_score::DOUBLE, e.vs_score::DOUBLE);
```

0	Combining Binary and worst- …	0.2	7
1	To join or not to join? Thinking …	0.4	NULL



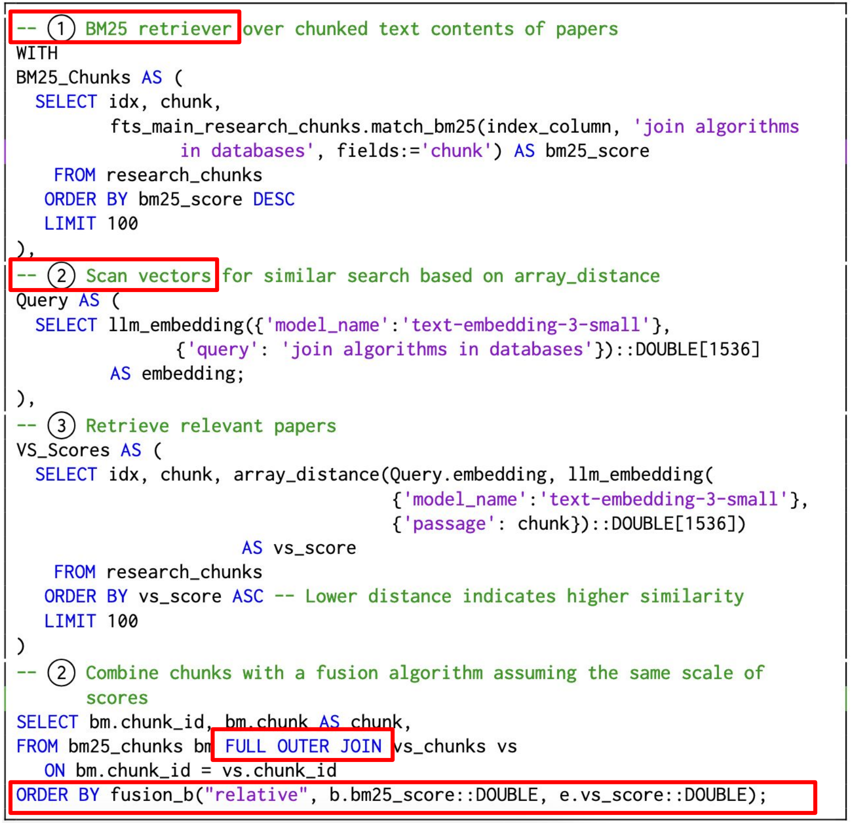
0	Combining Binary and worst- …	0.2
1	To join or not to join? Thinking …	0.4

M_1

0	Combining Binary and worst- …	7
8	Debunking the myth …	10

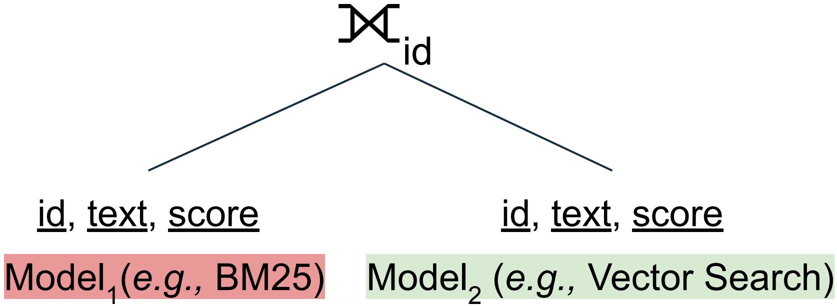
M_2

Query Examples (2)



```
-- ① BM25 retriever over chunked text contents of papers
WITH
BM25_Chunks AS (
  SELECT idx, chunk,
          fts_main_research_chunks.match_bm25(index_column, 'join algorithms
                  in databases', fields:='chunk') AS bm25_score
    FROM research_chunks
   ORDER BY bm25_score DESC
   LIMIT 100
),
-- ② Scan vectors for similar search based on array_distance
Query AS (
  SELECT llm_embedding({'model_name':'text-embedding-3-small'},
                 {'query': 'join algorithms in databases'})::DOUBLE[1536]
          AS embedding;
),
-- ③ Retrieve relevant papers
VS_Scores AS (
  SELECT idx, chunk, array_distance(Query.embedding, llm_embedding(
                               {'model_name':'text-embedding-3-small'},
                               {'passage': chunk})::DOUBLE[1536])
                    AS vs_score
    FROM research_chunks
   ORDER BY vs_score ASC -- Lower distance indicates higher similarity
   LIMIT 100
)
-- ② Combine chunks with a fusion algorithm assuming the same scale of
   scores
SELECT bm.chunk_id, bm.chunk AS chunk,
FROM bm25_chunks bm FULL OUTER JOIN vs_chunks vs
   ON bm.chunk_id = vs.chunk_id
ORDER BY fusion_b("relative", b.bm25_score::DOUBLE, e.vs_score::DOUBLE);
```

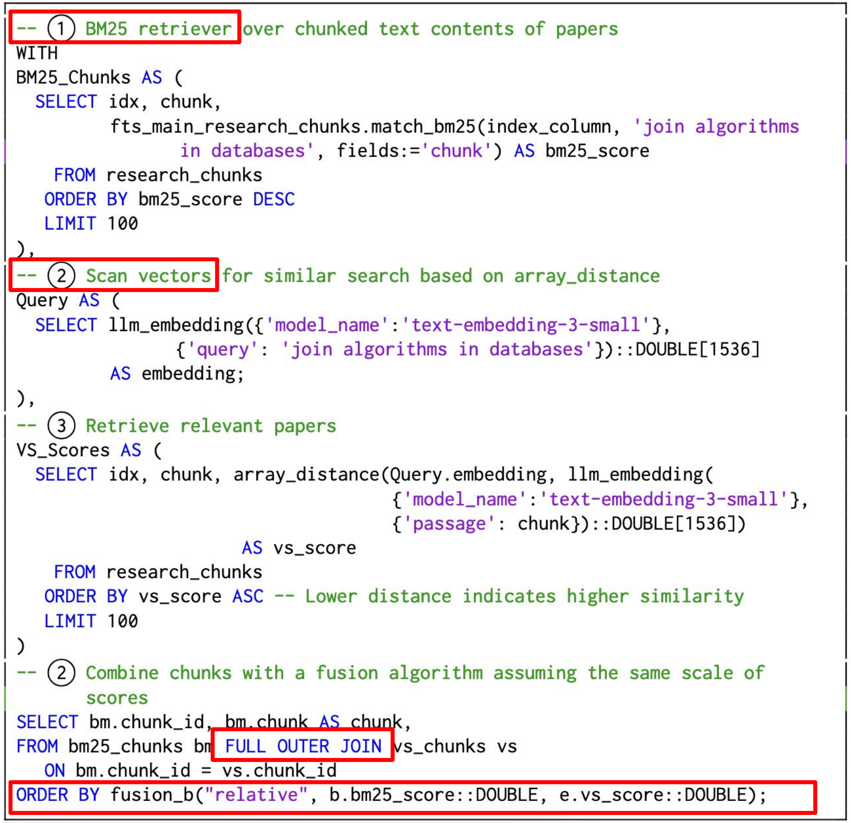
0	Combining Binary and worst- …	0.2	7
1	To join or not to join? Thinking …	0.4	NULL



0	Combining Binary and worst- …	0.2	M_1
1	To join or not to join? Thinking …	0.4	

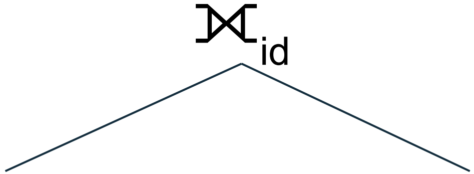
0	Combining Binary and worst- …	7	M_2
8	Debunking the myth …	10	

Query Examples (2)



```
-- ① BM25 retriever over chunked text contents of papers
WITH
BM25_Chunks AS (
  SELECT idx, chunk,
         fts_main_research_chunks.match_bm25(index_column, 'join algorithms
                in databases', fields:='chunk') AS bm25_score
    FROM research_chunks
   ORDER BY bm25_score DESC
   LIMIT 100
),
-- ② Scan vectors for similar search based on array_distance
Query AS (
  SELECT llm_embedding({'model_name':'text-embedding-3-small'},
                {'query': 'join algorithms in databases'})::DOUBLE[1536]
         AS embedding;
),
-- ③ Retrieve relevant papers
VS_Scores AS (
  SELECT idx, chunk, array_distance(Query.embedding, llm_embedding(
                                    {'model_name':'text-embedding-3-small'},
                                    {'passage': chunk})::DOUBLE[1536])
                     AS vs_score
    FROM research_chunks
   ORDER BY vs_score ASC -- Lower distance indicates higher similarity
   LIMIT 100
)
-- ② Combine chunks with a fusion algorithm assuming the same scale of
      scores
SELECT bm.chunk_id, bm.chunk AS chunk,
FROM bm25_chunks bm FULL OUTER JOIN vs_chunks vs
   ON bm.chunk_id = vs.chunk_id
ORDER BY fusion_b("relative", b.bm25_score::DOUBLE, e.vs_score::DOUBLE);
```

0	Combining Binary and worst- …	0.2	7
1	To join or not to join? Thinking …	0.4	NULL
8	Combining Binary and worst- …	NULL	10



id, text, score

id, text, score

Model$_1$(*e.g.,* BM25)

Model$_2$ (*e.g.,* Vector Search)

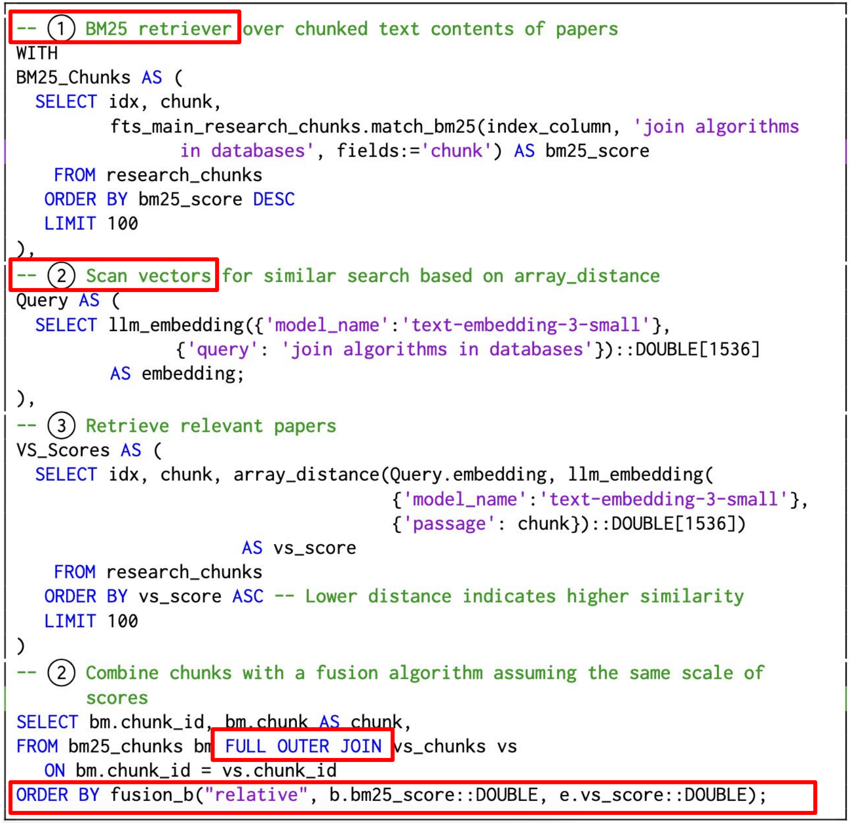
0	Combining Binary and worst- …	0.2
1	To join or not to join? Thinking …	0.4

M_1

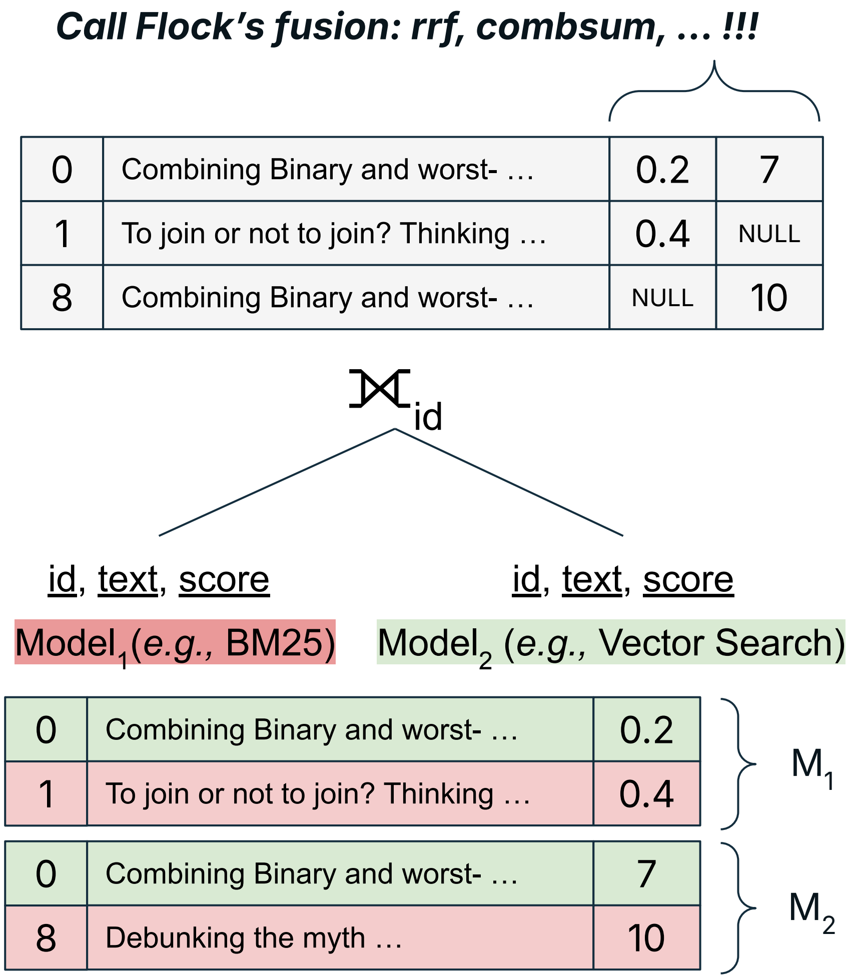
0	Combining Binary and worst- …	7
8	Debunking the myth …	10

M_2

Query Examples (2)



```
-- ① BM25 retriever over chunked text contents of papers
WITH
BM25_Chunks AS (
  SELECT idx, chunk,
         fts_main_research_chunks.match_bm25(index_column, 'join algorithms
                in databases', fields:='chunk') AS bm25_score
    FROM research_chunks
   ORDER BY bm25_score DESC
   LIMIT 100
),
-- ② Scan vectors for similar search based on array_distance
Query AS (
  SELECT llm_embedding({'model_name':'text-embedding-3-small'},
                {'query': 'join algorithms in databases'})::DOUBLE[1536]
         AS embedding;
),
-- ③ Retrieve relevant papers
VS_Scores AS (
  SELECT idx, chunk, array_distance(Query.embedding, llm_embedding(
                                 {'model_name':'text-embedding-3-small'},
                                 {'passage': chunk})::DOUBLE[1536])
                     AS vs_score
    FROM research_chunks
   ORDER BY vs_score ASC -- Lower distance indicates higher similarity
   LIMIT 100
)
-- ② Combine chunks with a fusion algorithm assuming the same scale of
   scores
SELECT bm.chunk_id, bm.chunk AS chunk,
FROM bm25_chunks bm FULL OUTER JOIN vs_chunks vs
   ON bm.chunk_id = vs.chunk_id
ORDER BY fusion_b("relative", b.bm25_score::DOUBLE, e.vs_score::DOUBLE);
```



Call Flock's fusion: rrf, combsum, ... !!!

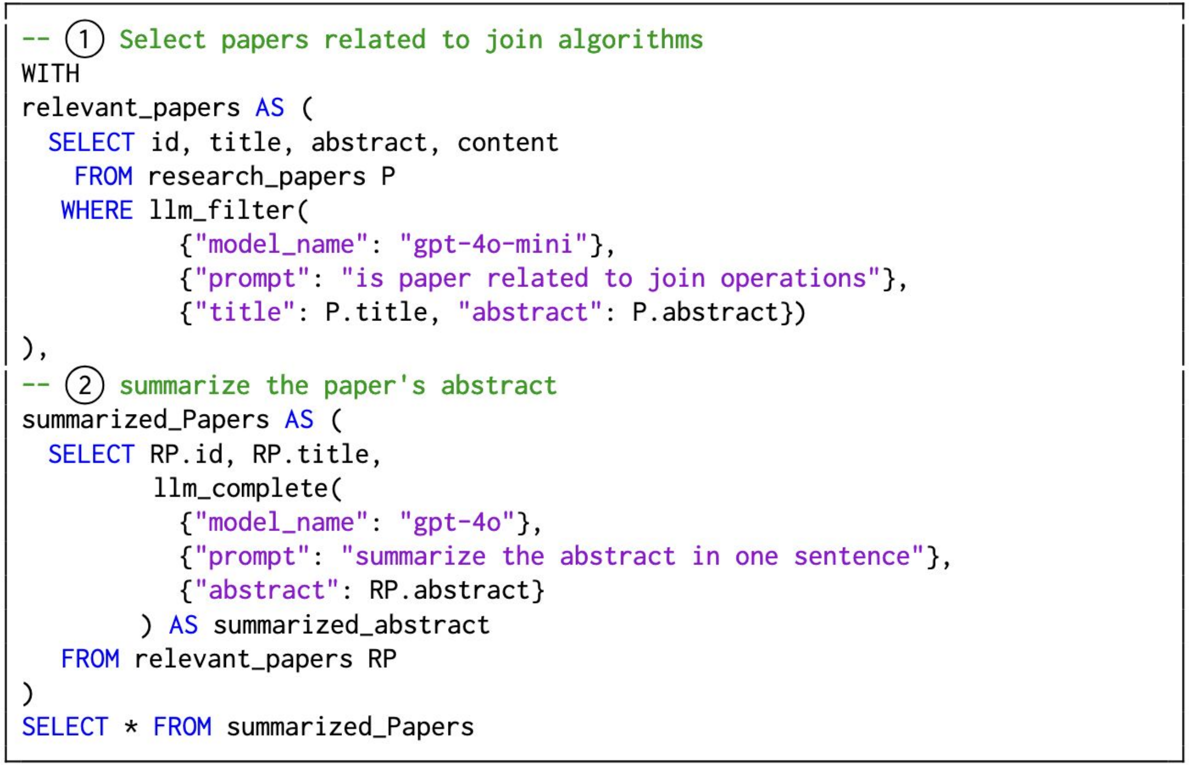
0	Combining Binary and worst- …	0.2	7
1	To join or not to join? Thinking …	0.4	NULL
8	Combining Binary and worst- …	NULL	10

0	Combining Binary and worst- …	0.2
1	To join or not to join? Thinking …	0.4

0	Combining Binary and worst- …	7
8	Debunking the myth …	10

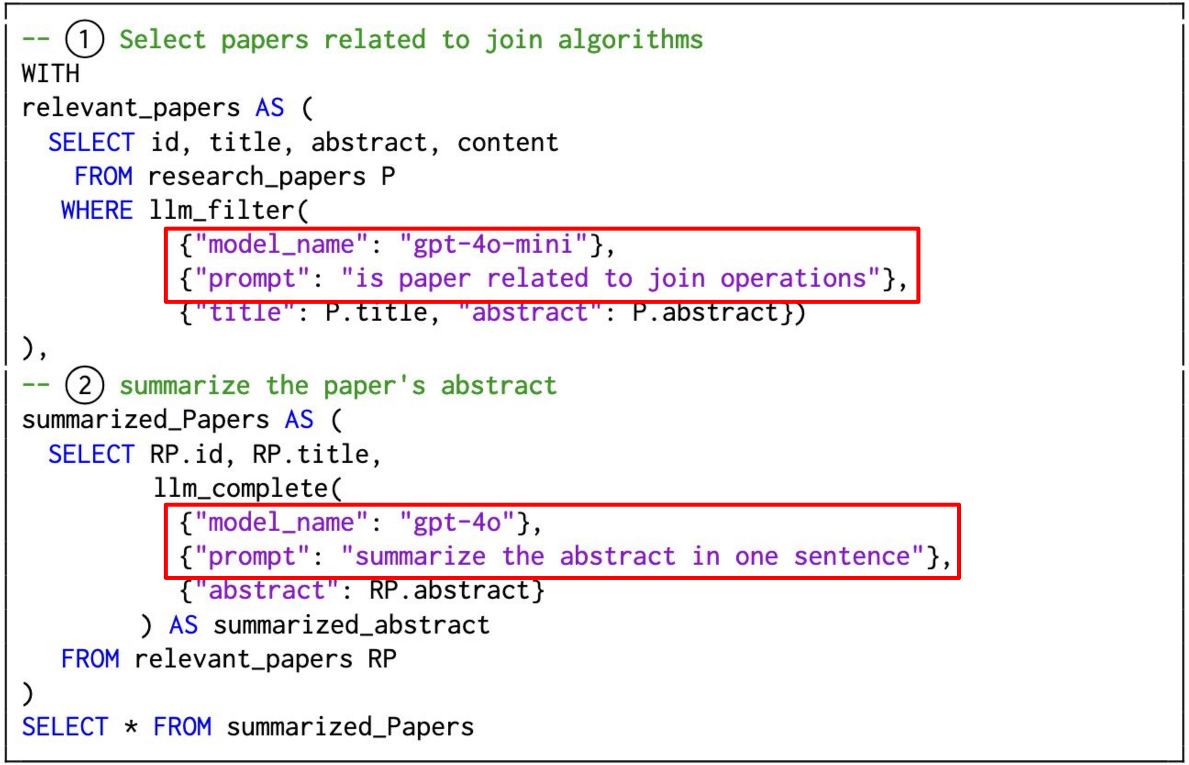
Data- / Model- Independence

Resource-Independence



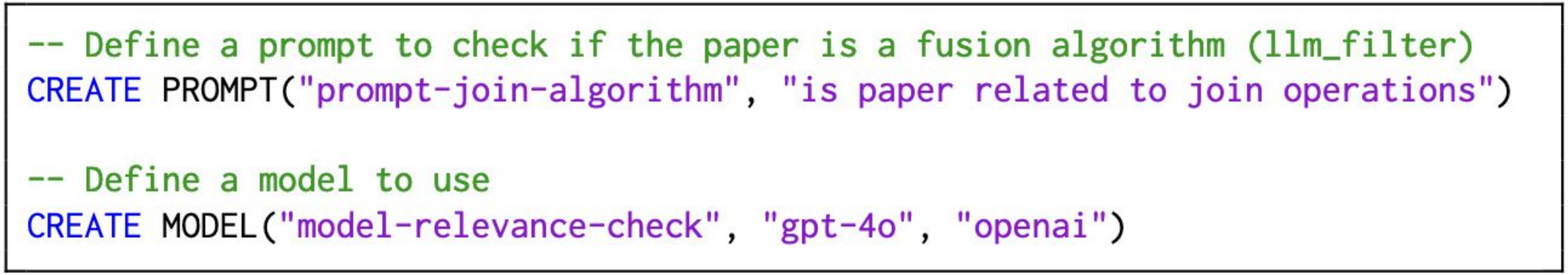
```sql
-- ① Select papers related to join algorithms
WITH
relevant_papers AS (
  SELECT id, title, abstract, content
    FROM research_papers P
   WHERE llm_filter(
            {"model_name": "gpt-4o-mini"},
            {"prompt": "is paper related to join operations"},
            {"title": P.title, "abstract": P.abstract})
),
-- ② summarize the paper's abstract
summarized_Papers AS (
  SELECT RP.id, RP.title,
          llm_complete(
            {"model_name": "gpt-4o"},
            {"prompt": "summarize the abstract in one sentence"},
            {"abstract": RP.abstract}
         ) AS summarized_abstract
   FROM relevant_papers RP
)
SELECT * FROM summarized_Papers
```

Resource-Independence



```
-- ① Select papers related to join algorithms
WITH
relevant_papers AS (
  SELECT id, title, abstract, content
    FROM research_papers P
   WHERE llm_filter(
           {"model_name": "gpt-4o-mini"},
           {"prompt": "is paper related to join operations"},
           {"title": P.title, "abstract": P.abstract})
),
-- ② summarize the paper's abstract
summarized_Papers AS (
  SELECT RP.id, RP.title,
         llm_complete(
           {"model_name": "gpt-4o"},
           {"prompt": "summarize the abstract in one sentence"},
           {"abstract": RP.abstract}
         ) AS summarized_abstract
   FROM relevant_papers RP
)
SELECT * FROM summarized_Papers
```

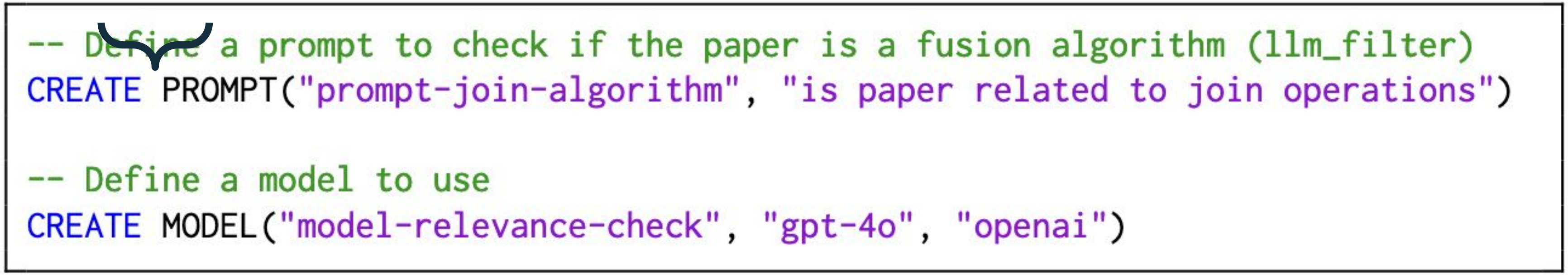
New resource creation



```
-- Define a prompt to check if the paper is a fusion algorithm (llm_filter)
CREATE PROMPT("prompt-join-algorithm", "is paper related to join operations")

-- Define a model to use
CREATE MODEL("model-relevance-check", "gpt-4o", "openai")
```

New resource creation - Local vs Global

GLOBAL

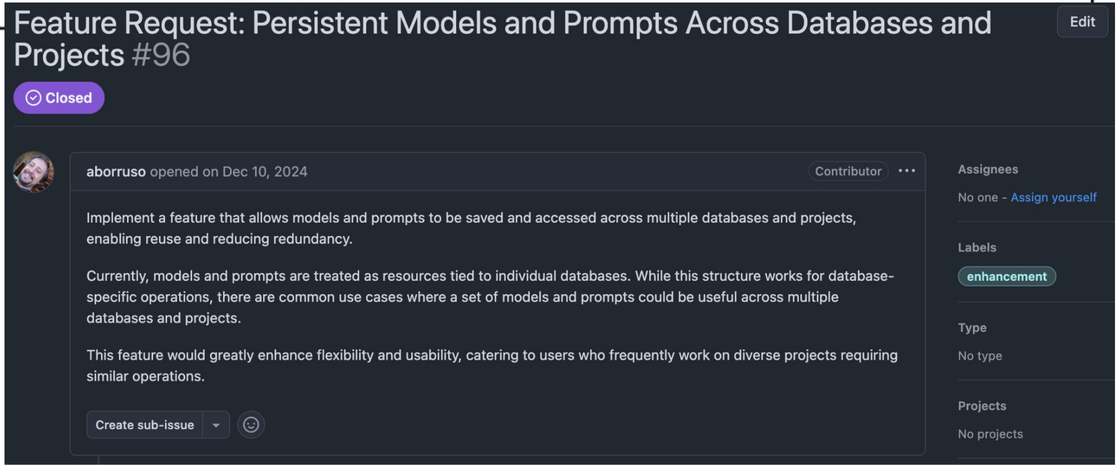


```
-- Define a prompt to check if the paper is a fusion algorithm (llm_filter)
CREATE PROMPT("prompt-join-algorithm", "is paper related to join operations")

-- Define a model to use
CREATE MODEL("model-relevance-check", "gpt-4o", "openai")
```

Data-Independence - new resources

GLOBAL

```
-- Define a prompt to check if the paper is a fusion algorithm (llm_filter)
CREATE PROMPT("prompt-join-algorithm", "is paper related to join operations")

-- Define a model to use
CREATE MODEL("model-relevance-check", "gpt-4o", "openai")
```



Feature Request: Persistent Models and Prompts Across Databases and Projects #96

Closed

aborruso opened on Dec 10, 2024

Implement a feature that allows models and prompts to be saved and accessed across multiple databases and projects, enabling reuse and reducing redundancy.

Currently, models and prompts are treated as resources tied to individual databases. While this structure works for database-specific operations, there are common use cases where a set of models and prompts could be useful across multiple databases and projects.

This feature would greatly enhance flexibility and usability, catering to users who frequently work on diverse projects requiring similar operations.

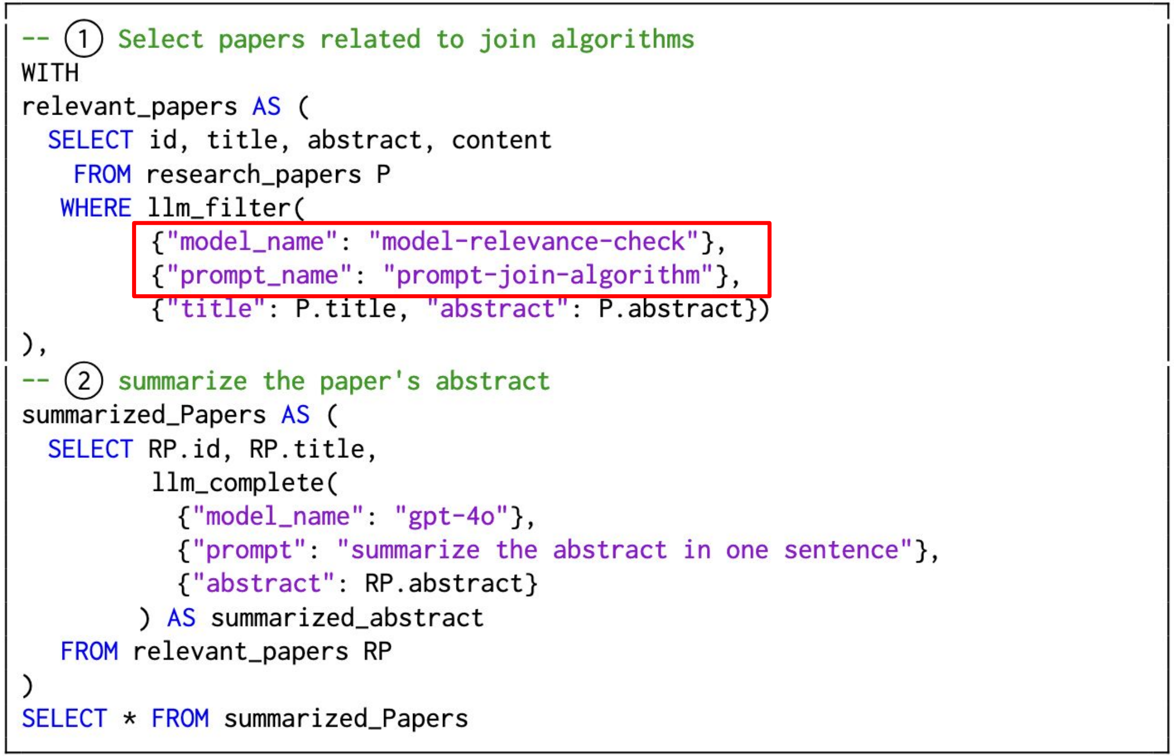
Assignees: No one - Assign yourself

Labels: enhancement

Type: No type

Projects: No projects

Data-Independence - new resources

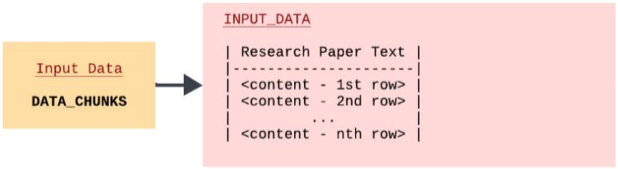


```
-- ① Select papers related to join algorithms
WITH
relevant_papers AS (
  SELECT id, title, abstract, content
    FROM research_papers P
   WHERE llm_filter(
           {"model_name": "model-relevance-check"},
           {"prompt_name": "prompt-join-algorithm"},
           {"title": P.title, "abstract": P.abstract})
),
-- ② summarize the paper's abstract
summarized_Papers AS (
  SELECT RP.id, RP.title,
           llm_complete(
             {"model_name": "gpt-4o"},
             {"prompt": "summarize the abstract in one sentence"},
             {"abstract": RP.abstract}
          ) AS summarized_abstract
   FROM relevant_papers RP
)
SELECT * FROM summarized_Papers
```

Prompt behind Implementation

Prompt behind Implementation

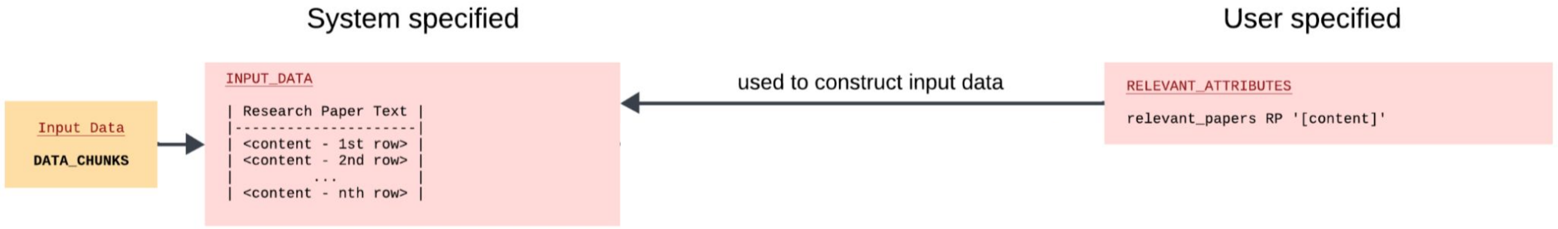
System specified



INPUT_DATA

```
| Research Paper Text |
|---------------------|
| <content - 1st row> |
| <content - 2nd row> |
|         ...         |
| <content - nth row> |
```

Prompt behind Implementation



System specified

User specified

Input Data

DATA_CHUNKS

INPUT_DATA

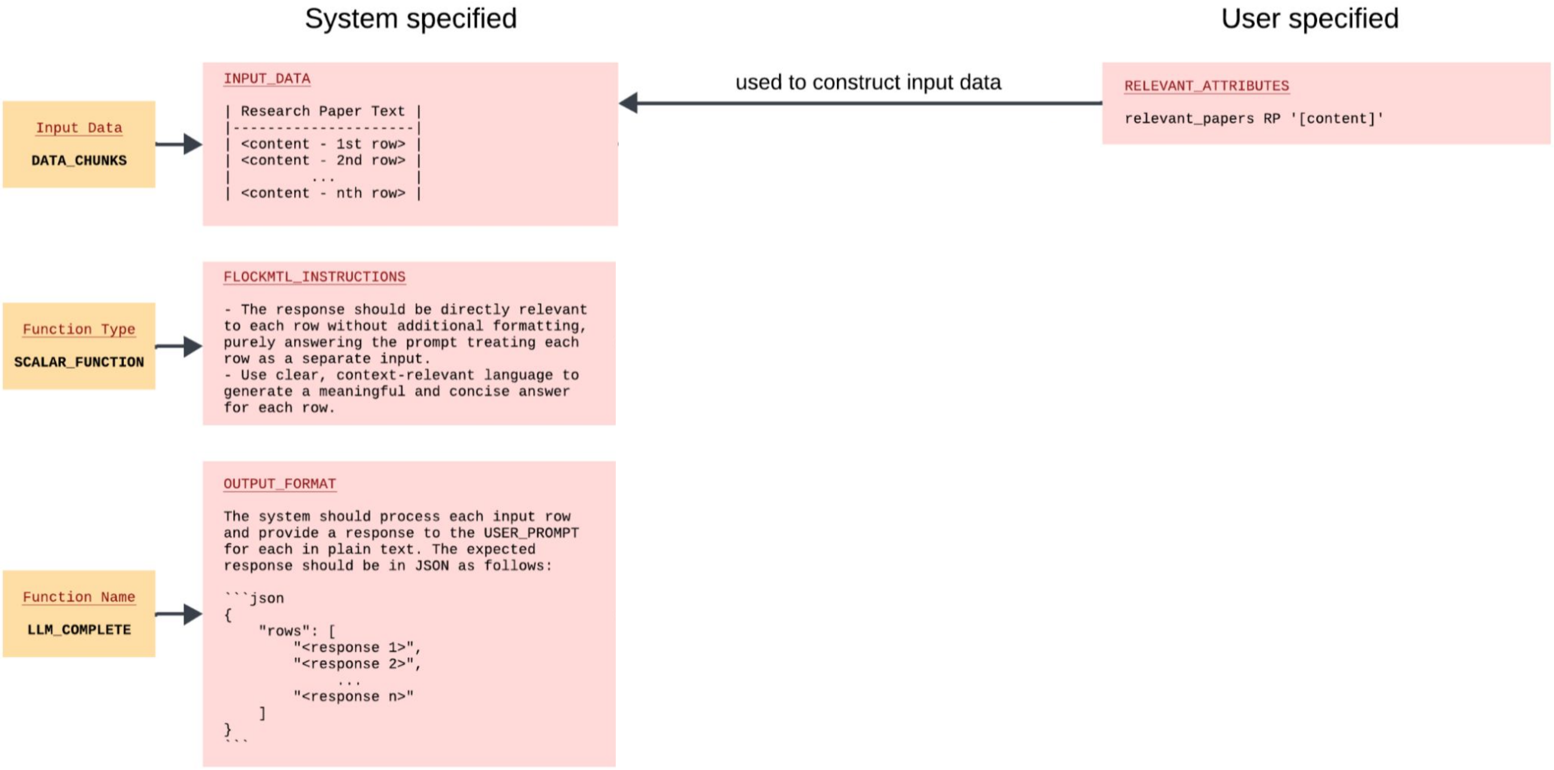
```
| Research Paper Text |
|---------------------|
| <content - 1st row> |
| <content - 2nd row> |
|         ...         |
| <content - nth row> |
```

used to construct input data

RELEVANT_ATTRIBUTES

```
relevant_papers RP '[content]'
```

Prompt behind Implementation



System specified

User specified

Input Data

DATA_CHUNKS

INPUT_DATA

```
| Research Paper Text |
|---------------------|
| <content - 1st row> |
| <content - 2nd row> |
|         ...         |
| <content - nth row> |
```

used to construct input data

RELEVANT_ATTRIBUTES

```
relevant_papers RP '[content]'
```

Function Type

SCALAR_FUNCTION

FLOCKMTL_INSTRUCTIONS

```
- The response should be directly relevant
to each row without additional formatting,
purely answering the prompt treating each
row as a separate input.
- Use clear, context-relevant language to
generate a meaningful and concise answer
for each row.
```

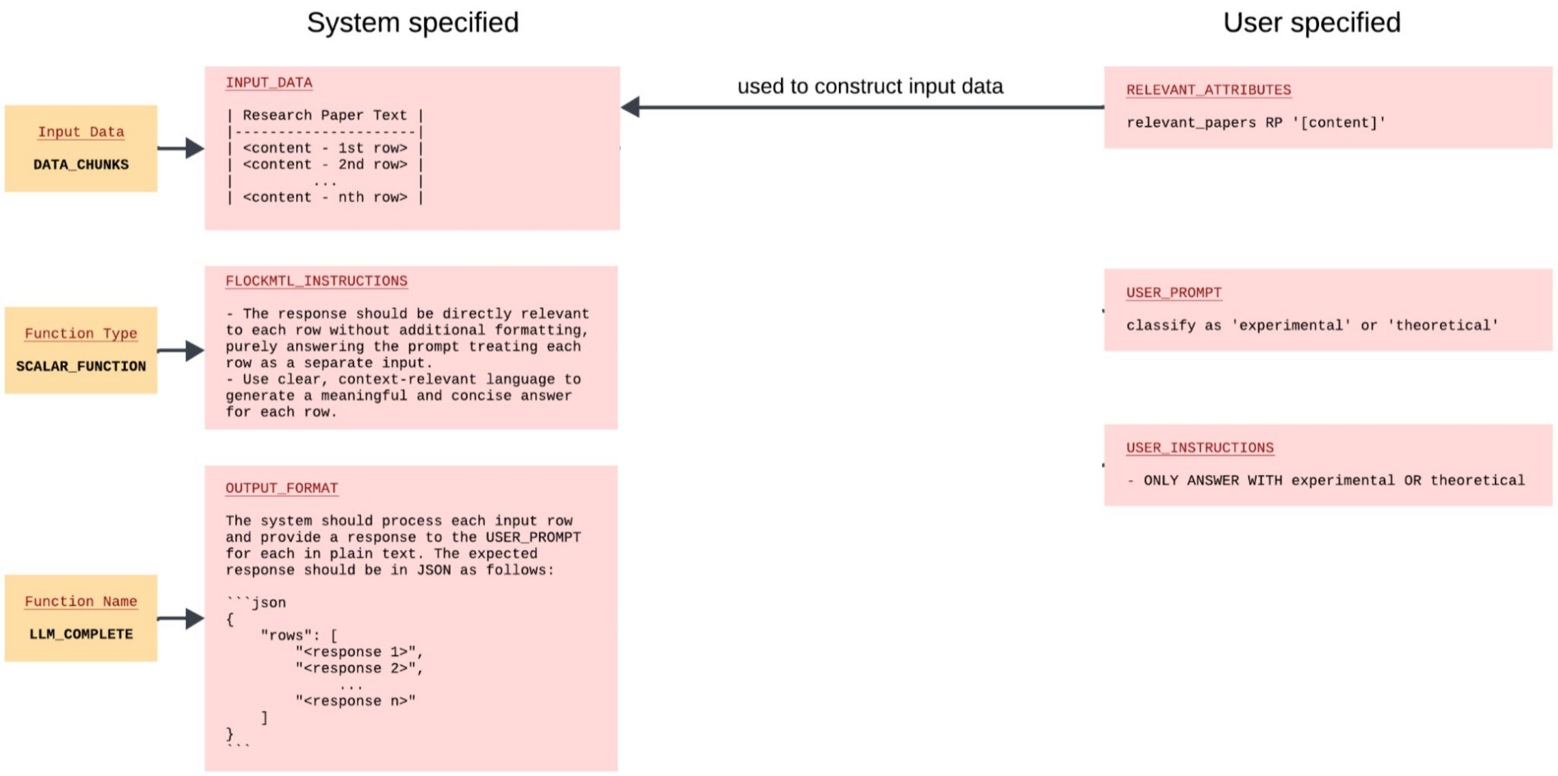
Function Name

LLM_COMPLETE

OUTPUT_FORMAT

~~~
The system should process each input row
and provide a response to the USER_PROMPT
for each in plain text. The expected
response should be in JSON as follows:

```json
{
    "rows": [
        "<response 1>",
        "<response 2>",
              ...
        "<response n>"
    ]
}
```
~~~

Prompt behind Implementation



INPUT_DATA

```
| Research Paper Text |
|---------------------|
| <content - 1st row> |
| <content - 2nd row> |
|         ...         |
| <content - nth row> |
```

FLOCKMTL_INSTRUCTIONS

```
- The response should be directly relevant
to each row without additional formatting,
purely answering the prompt treating each
row as a separate input.
- Use clear, context-relevant language to
generate a meaningful and concise answer
for each row.
```

OUTPUT_FORMAT

~~~
The system should process each input row
and provide a response to the USER_PROMPT
for each in plain text. The expected
response should be in JSON as follows:

```json
{
    "rows": [
        "<response 1>",
        "<response 2>",
            ...
        "<response n>"
    ]
}
```
~~~

RELEVANT_ATTRIBUTES

```
relevant_papers RP '[content]'
```

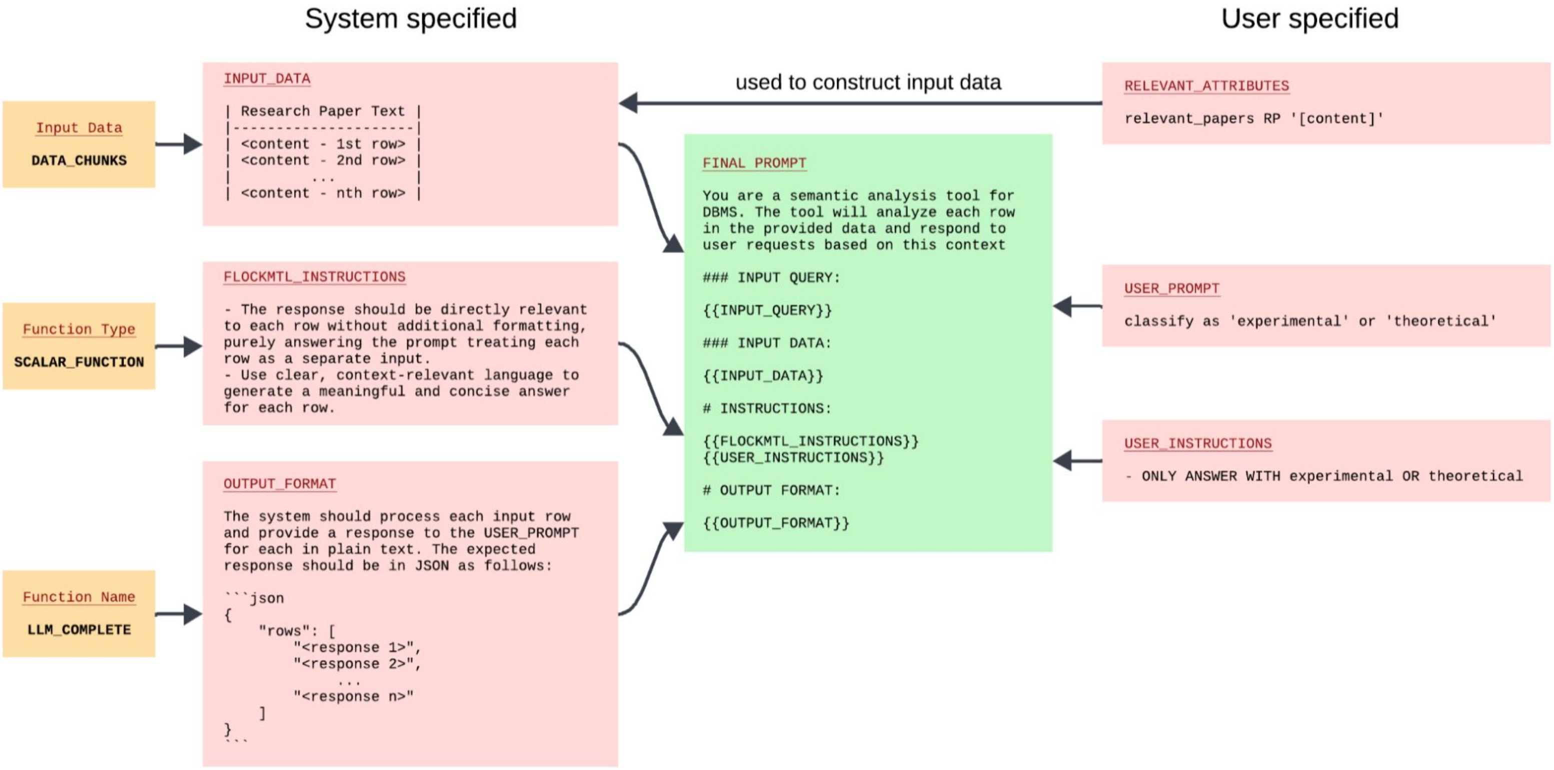
USER_PROMPT

```
classify as 'experimental' or 'theoretical'
```

USER_INSTRUCTIONS

```
- ONLY ANSWER WITH experimental OR theoretical
```

Prompt behind Implementation



System specified

User specified

Input Data

DATA_CHUNKS

INPUT_DATA

```
| Research Paper Text |
|---------------------|
| <content - 1st row> |
| <content - 2nd row> |
|         ...         |
| <content - nth row> |
```

Function Type

SCALAR_FUNCTION

FLOCKMTL_INSTRUCTIONS

```
- The response should be directly relevant
to each row without additional formatting,
purely answering the prompt treating each
row as a separate input.
- Use clear, context-relevant language to
generate a meaningful and concise answer
for each row.
```

Function Name

LLM_COMPLETE

OUTPUT_FORMAT

~~~
The system should process each input row
and provide a response to the USER_PROMPT
for each in plain text. The expected
response should be in JSON as follows:

```json
{
    "rows": [
        "<response 1>",
        "<response 2>",
          ...
        "<response n>"
    ]
}
```
~~~

used to construct input data

RELEVANT_ATTRIBUTES

```
relevant_papers RP '[content]'
```

FINAL_PROMPT

```
You are a semantic analysis tool for
DBMS. The tool will analyze each row
in the provided data and respond to
user requests based on this context

### INPUT QUERY:

{{INPUT_QUERY}}

### INPUT DATA:

{{INPUT_DATA}}

# INSTRUCTIONS:

{{FLOCKMTL_INSTRUCTIONS}}
{{USER_INSTRUCTIONS}}

# OUTPUT FORMAT:

{{OUTPUT_FORMAT}}
```

USER_PROMPT

```
classify as 'experimental' or 'theoretical'
```

USER_INSTRUCTIONS

```
- ONLY ANSWER WITH experimental OR theoretical
```

Operator Optimizations

Operator Optimizations

1) Batching

   Input / Output based

Operator Optimizations

1) Batching
   Input / Output based

Bank reviews from Kaggle	*llm_complete GPT4o in* secs over 1,000 tuples with batch size B	
	B = 1	B = 64
XML	1048.05	181.53 **(5.8x)**
JSON	1350.65	189.91 **(7.1x)**
Markdown	965.64	196.04 **(4.9x)**

Operator Optimizations

1) Batching
   Input / Output based

Bank reviews from Kaggle	*Llm_embedding GPT4o* in secs over 1,000 tuples with batch size B	
	B = 1	B = 512
	677.6	14.03 **(48.3x)**

Operator Optimizations

1) Batching
   Input / Output based

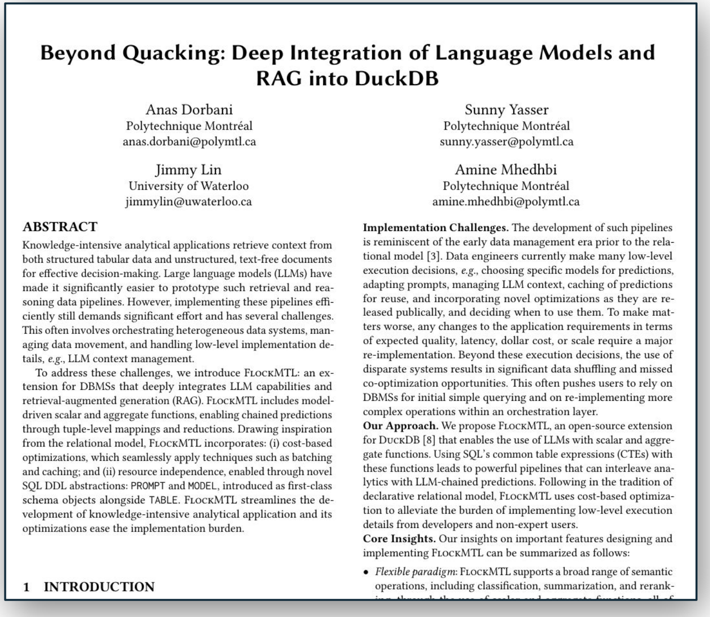
2) Deduplication
   Data Chunk / Stream

3) Caching
   Tuple → Output

What is next?

- Text → SQL (augmented by Flock)

VLDB 2025 - Demo



Beyond Quacking: Deep Integration of Language Models and RAG into DuckDB

Anas Dorbani
Polytechnique Montréal
anas.dorbani@polymtl.ca

Sunny Yasser
Polytechnique Montréal
sunny.yasser@polymtl.ca

Jimmy Lin
University of Waterloo
jimmylin@uwaterloo.ca

Amine Mhedhbi
Polytechnique Montréal
amine.mhedhbi@polymtl.ca

ABSTRACT

Knowledge-intensive analytical applications retrieve context from both structured tabular data and unstructured, text-free documents for effective decision-making. Large language models (LLMs) have made it significantly easier to prototype such retrieval and reasoning data pipelines. However, implementing these pipelines efficiently still demands significant effort and has several challenges. This often involves orchestrating heterogeneous data systems, managing data movement, and handling low-level implementation details, *e.g.*, LLM context management.

To address these challenges, we introduce FlockMTL: an extension for DBMSs that deeply integrates LLM capabilities and retrieval-augmented generation (RAG). FlockMTL includes model-driven scalar and aggregate functions, enabling chained predictions through tuple-level mappings and reductions. Drawing inspiration from the relational model, FlockMTL incorporates: (i) cost-based optimizations, which seamlessly apply techniques such as batching and caching; and (ii) resource independence, enabled through novel SQL DDL abstractions: `PROMPT` and `MODEL`, introduced as first-class schema objects alongside `TABLE`. FlockMTL streamlines the development of knowledge-intensive analytical application and its optimizations ease the implementation burden.

1 INTRODUCTION

Implementation Challenges. The development of such pipelines is reminiscent of the early data management era prior to the relational model [3]. Data engineers currently make many low-level execution decisions, *e.g.*, choosing specific models for predictions, adapting prompts, managing LLM context, caching of predictions for reuse, and incorporating novel optimizations as they are released publically, and deciding when to use them. To make matters worse, any changes to the application requirements in terms of expected quality, latency, dollar cost, or scale require a major re-implementation. Beyond these execution decisions, the use of disparate systems results in significant data shuffling and missed co-optimization opportunities. This often pushes users to rely on DBMSs for initial simple querying and on re-implementing more complex operations within an orchestration layer.

Our Approach. We propose FlockMTL, an open-source extension for DuckDB [8] that enables the use of LLMs with scalar and aggregate functions. Using SQL's common table expressions (CTEs) with these functions leads to powerful pipelines that can interleave analytics with LLM-chained predictions. Following in the tradition of declarative relational model, FlockMTL uses cost-based optimization to alleviate the burden of implementing low-level execution details from developers and non-expert users.

Core Insights. Our insights on important features designing and implementing FlockMTL can be summarized as follows:

- *Flexible paradigm*: FlockMTL supports a broad range of semantic operations, including classification, summarization, and rerank-

What is next?

- Text → SQL (augmented by Flock)
- Extending beyond OpenAI, Azure, Ollama
- Many short-term Optimizations
  - LLM-Filter
  - Accuracy vs exec trade-off on aggregates
  - Multi-modal Joins
- Go higher to the app layer
  - Multi-table RAG
  - Other models tabular FMs
- Go lower than operators

Fin.
Questions?

amine.mhedhbi@polymtl.ca

POLYTECHNIQUE
MONTRÉAL