Data Agents: Levels, State of the Art, and Open Problems

Guoliang Li
Tsinghua University

Ju Fan
Renmin University of China

Yuyu Luo
HKUST (Guangzhou)

Nan Tang
HKUST (Guangzhou)

https://github.com/HKUSTDial/awesome-data-agents
https://dbgroup.cs.tsinghua.edu.cn/ligl/DA.html

Outline

- Part I: Data Agents: Motivation, Definition, Autonomy Levels, and Core Challenges
- Part II: Towards Autonomous Data Agents: Key Challenges and Current Practices
  - Data-aware Workflow Orchestration & Cost-aware Execution
  - Long-horizon Agentic State, Memory & Skill Reuse
  - Semantic Grounding over Heterogeneous & Multimodal Data
- Part III: Research Opportunities and Open Challenges

Outline

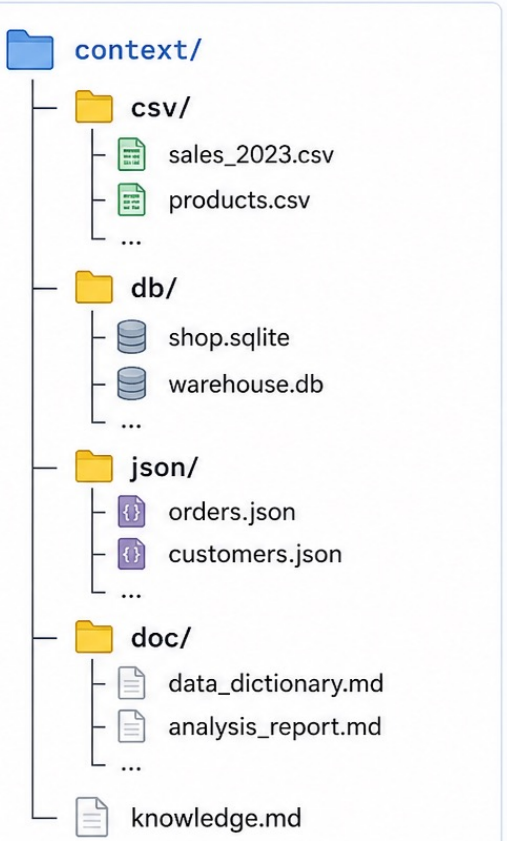
- **Part I: Data Agents: Motivation, Definition, Autonomy Levels, and Core Challenges**
- Part II: Towards Autonomous Data Agents: Key Challenges and Current Practices
  - Data-aware Workflow Orchestration & Cost-aware Execution
  - Long-horizon Agentic State, Memory & Skill Reuse
  - Semantic Grounding over Heterogeneous & Multimodal Data
- Part III: Research Opportunities and Open Challenges

A “Simple” Data Analysis Case

The data are multi-source and heterogeneous.

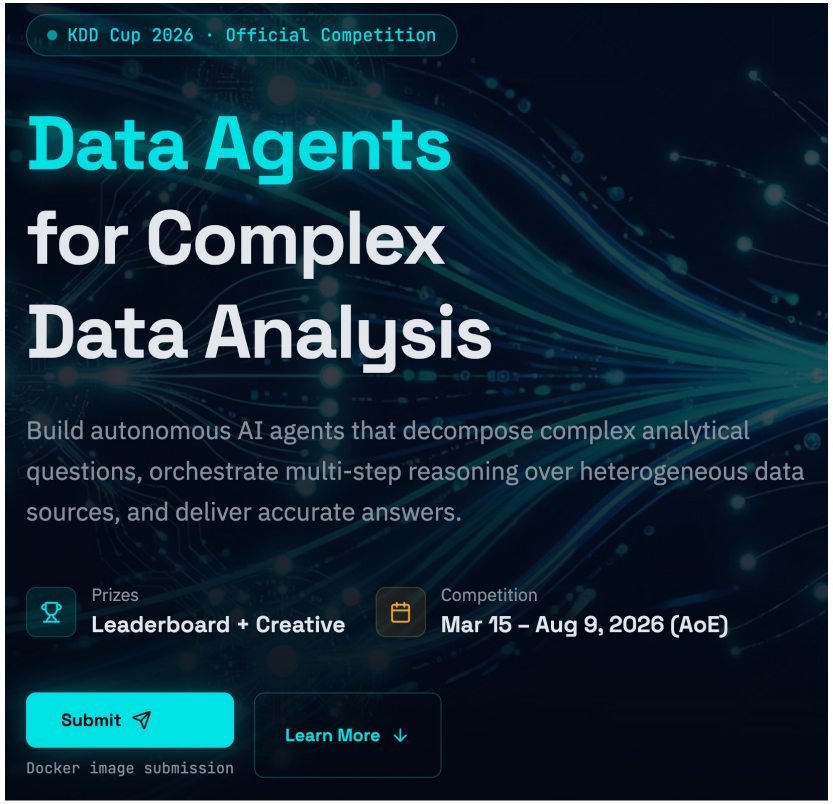
context/ Data Sources Description

The data source types under the `context/` directory vary by task. The participant's Agent must detect and read the subdirectories and files that actually exist:



Subdirectory / File	Content Type	Description
csv/	Structured Tables	One or more CSV files, directly readable with pandas and other tools
db/	SQLite Database	One or more .sqlite / .db files containing multiple relational tables
json/	Structured Data	Semi-structured data files in JSON format
doc/	Data Documentation	Data reports and analysis documents in Markdown or other formats
knowledge.md	Background Knowledge	Background knowledge document related to the task, including business definitions and terminology explanations

KDD Cup 2026 Data Agent Track
https://dataagent.top/

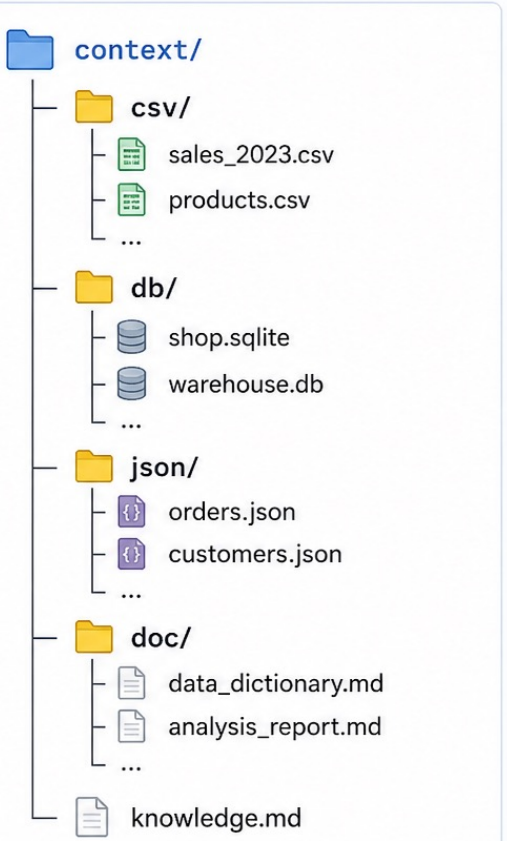


A “Simple” Data Analysis Case

The data are multi-source and heterogeneous.

context/ Data Sources Description

The data source types under the `context/` directory vary by task. The participant's Agent must detect and read the subdirectories and files that actually exist:



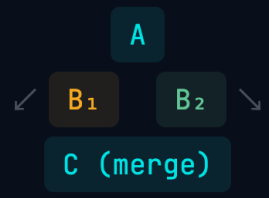
Subdirectory / File	Content Type	Description
csv/	Structured Tables	One or more CSV files, directly readable with pandas and other tools
db/	SQLite Database	One or more .sqlite / .db files containing multiple relational tables
json/	Structured Data	Semi-structured data files in JSON format
doc/	Data Documentation	Data reports and analysis documents in Markdown or other formats
knowledge.md	Background Knowledge	Background knowledge document related to the task, including business definitions and terminology explanations

The reasoning is diverse.

Sequential Chain

Each step depends on the previous step's output. Errors propagate downstream.

Branching & Merging



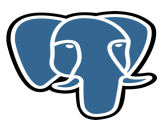
Parallel sub-queries across different data sources, then merge results.

Iterative Loop

Iterative refinement where the agent revisits and corrects intermediate results.

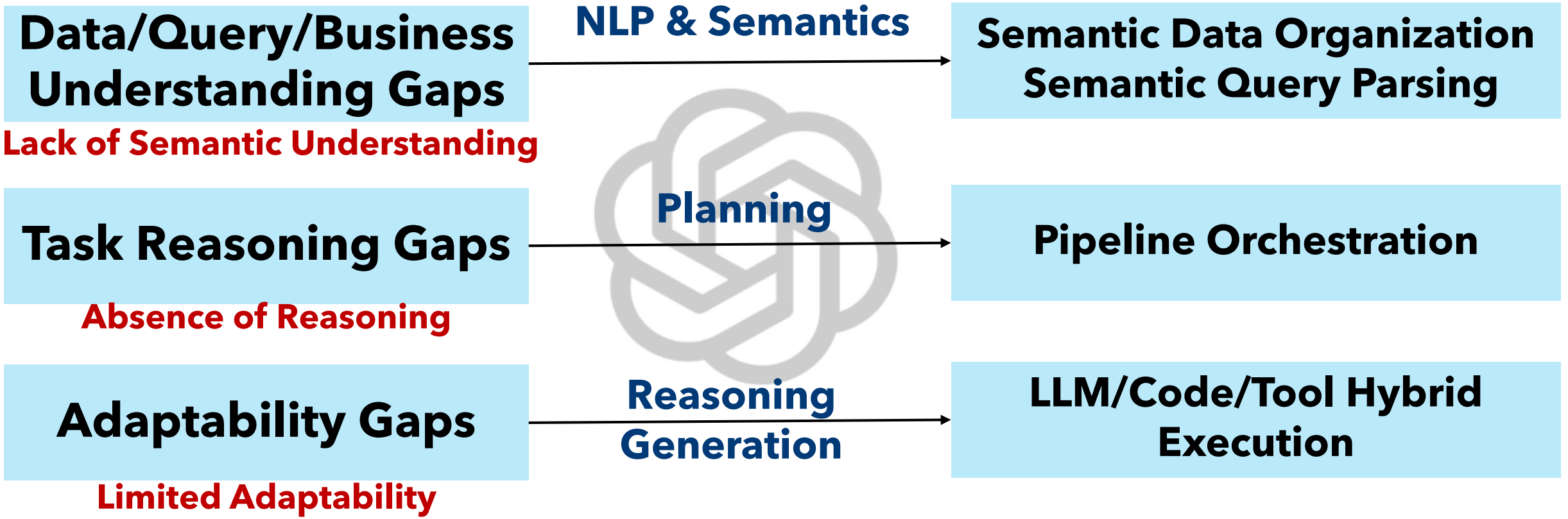
Database → Data Systems

Database		Data System
Structured	Understanding	Unstructured + structured
SQL	Reasoning	NL
Literal	Semantics	Semantic
Closed World	Generation	Open World
Manual	Planning	Autonomy




Autonomous Data System

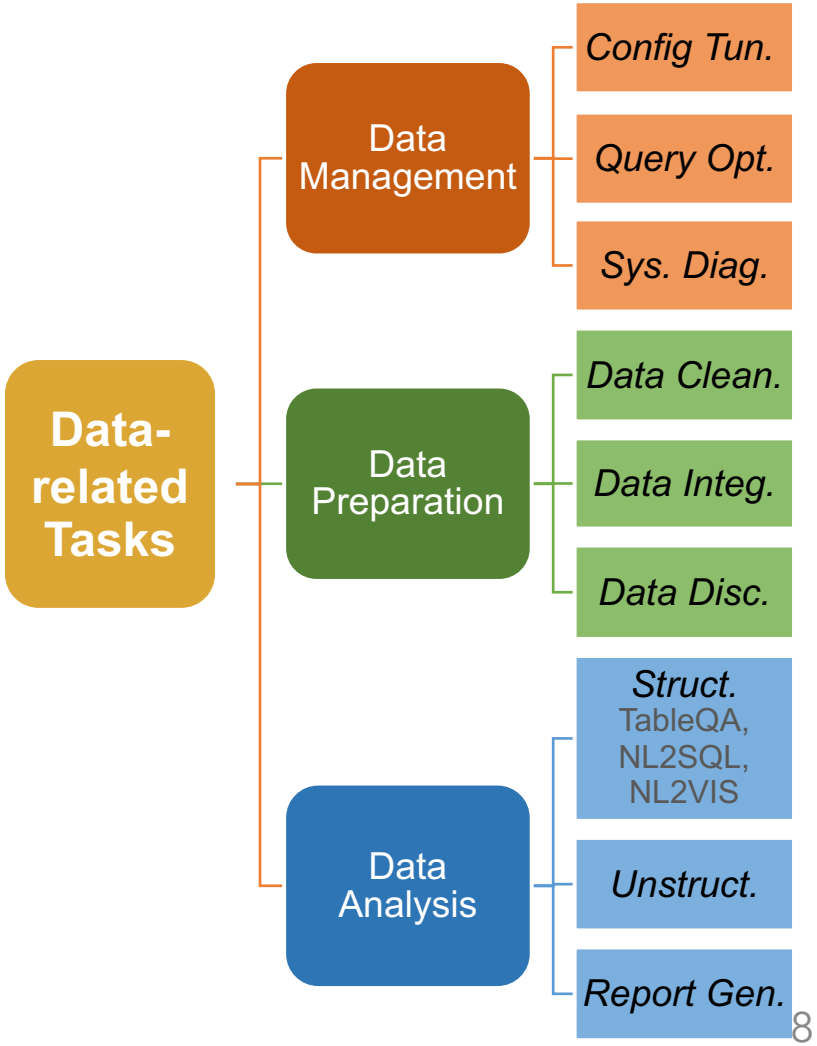
An autonomous system is crucial for Data+AI applications.



Data Agent (Agentic Data System)

Autonomously executes data tasks without human intervention, creating an E2E autonomous loop from raw data to business actions.

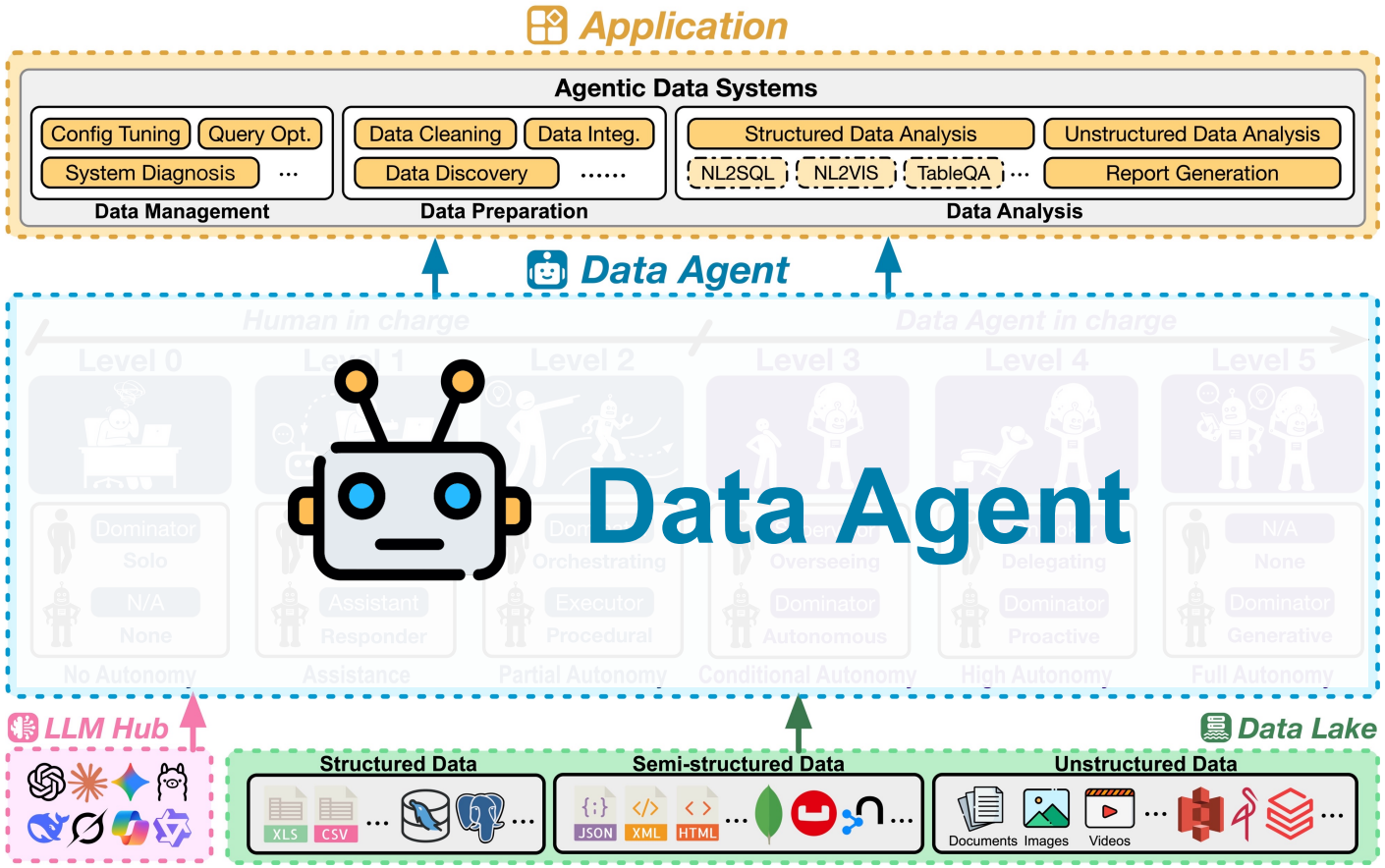
- **Connotation**
  - Autonomy: without continuous human intervention
    - Perception: Task & Environment Understanding
    - Orchestration: Task Decomposition
    - Reasoning & Planning: Optimization & Execution
    - Memory: Perception, Understanding, Semantics, Context
    - Self-reflection: Feedback
    - Multi-Agent Collaboration
  - Continuous Learning: from new data & behavior
  - Dynamic Adaptability: Adapt to different domains
  - Proactivity: Predict the future and take initiative



Data Agent (Agentic Data System)

Autonomously executes data tasks without human intervention, creating an E2E autonomous loop from raw data to business actions.

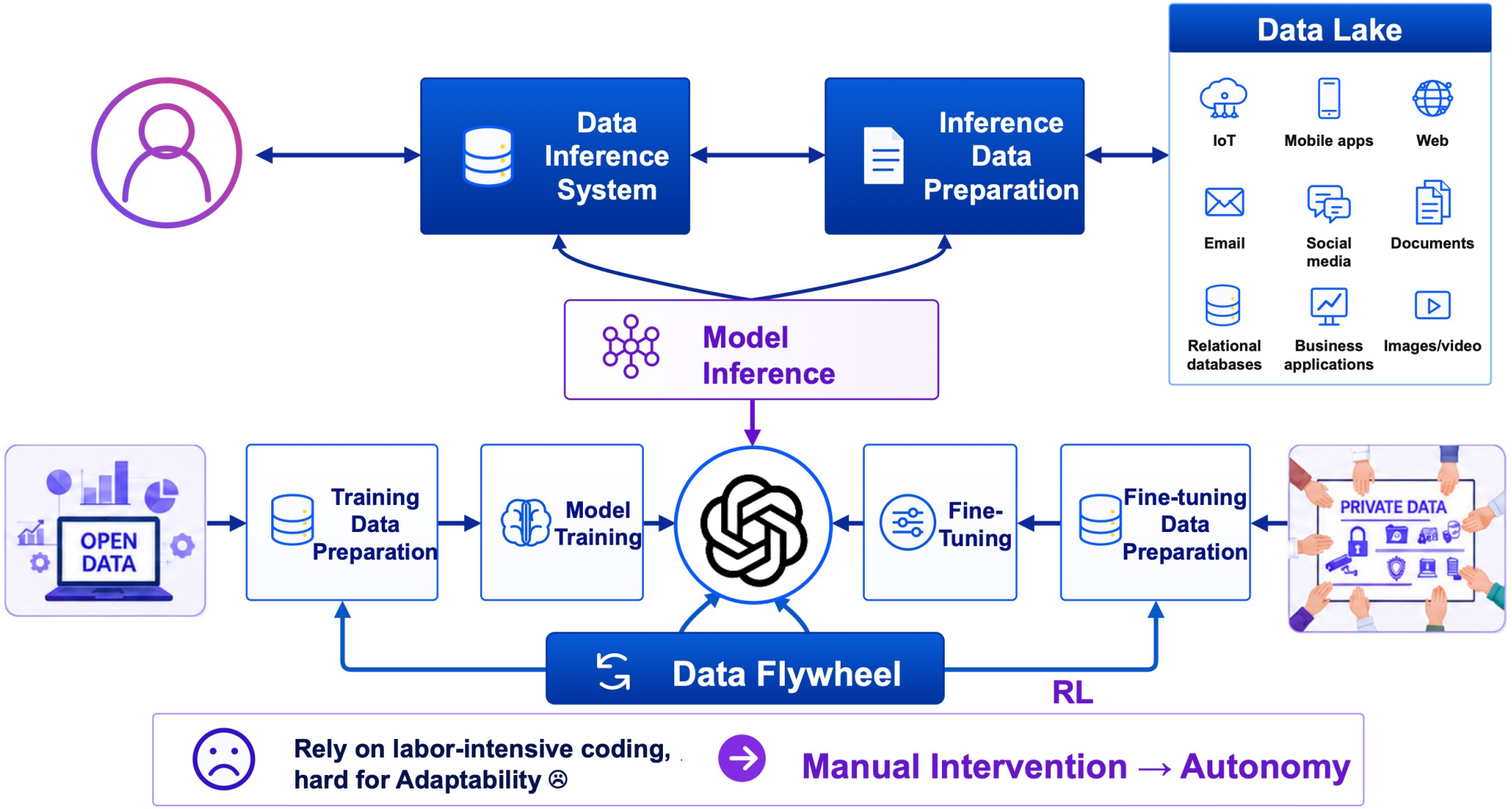
- Denotation: a paradigm shift towards autonomous data processing systems.
- Data Analytics Agent
  - Unstructured Data Agent
  - Semantic Structured Data Agent
  - Data Lake Agent
  - Multi-Modal Data Agent
- Data Management Agent
- Data Preparation



[1] Sun Z, Wang J, Zhao X, et al. Data Agent: A Holistic Architecture for Orchestrating Data+ AI Ecosystems

Data Agent: Framework

- **A holistic view of diverse sources to unify data-related tasks**



Scenario: Data Management

1) User Request

A sales dashboard query now takes 42s.Can you optimize it and explain why it is slow?

Analyst / DBA

2) Data Agent Workflow

1. **Understand workload**
   Interpret intent, SLA, and workload context.
2. **Ground on system evidence**
   Read schema, statistics, logs, and resource signals.
3. **Analyze plans & bottlenecks**
   Inspect query plan, cardinalities, skew, I/O, and locks.
4. **Invoke DM tools**
   Use SQL rewrite, index advisor, config tuning, and system diagnosis tools.
5. **Verify & recommend**
   Validate latency improvement and return actionable fixes.

Data-Agent-specific core

Planning | Tool Use | Memory | Verification

3) Outcome

- Optimized SQL / indexes
- Better execution plan
- Verified improvement 42s → 3.8s
- Clear diagnosis root cause + recommended actions

Evidence & Context Used by the Data Agent

- **Schema & metadata**
  tables, columns, indexes, constraints
- **Workload logs**
  SQL text,frequency, latency patterns
- **Execution plans**
  operators, costs, cardinalities
- **System metrics**
  CPU, memory, I/O, waits, locks

Why this is a Data Agent (not just an LLM agent)

- Grounded in live database evidence
- Uses specialized data-system tools
- Verifies changes before recommending them

Scenario: Data Preparation

1) User Request

Build a customer-360 table for churn prediction from CRM, orders, web logs, and support tickets.

Data Scientist / Analyst

2) Data Agent Workflow

1. **Discover sources**
   Find relevant tables, logs, documents, and schemas.
2. **Profile data quality**
   Detect missing values, outliers, drift, and conflicts.
3. **Clean & standardize**
   Apply cleaning rules, normalization, and schema alignment.
4. **Resolve & integrate**
   Deduplicate entities and join multi-source records.
5. **Validate & publish**
   Check quality constraints, lineage, and release a feature-ready table.

Data-Agent-specific core

Planning | Tool Use | Memory | Verification

3) Outcome

- Unified customer-360 table
- Higher data quality
- Feature-ready dataset
- Duplicate rate 18% → 1.2%

Evidence & Context Used by the Data Agent

- **CRM data**
  profiles, contacts, segments
- **Orders & transactions**
  products, spend, returns
- **Web logs**
  sessions, events, campaign signals
- **Support tickets**
  issues, categories, resolutions

Why this is a Data Agent (not just an LLM agent)

- Grounded in heterogeneous enterprise data
- Executes cleaning and integration tools
- Tracks lineage and validates quality before publishing

Scenario: Data Analysis

1) User Request

Revenue dropped in APAC last month. Find the main causes and prepare an executive report.

Business Analyst / Manager

2) Data Agent Workflow

1. **Understand the question** — Define KPI, scope, time period, and drill-down dimensions.
2. **Query structured data** — Retrieve sales, customer, and operational signals.
3. **Analyze unstructured evidence** — Read reviews, tickets, and textual feedback.
4. **Generate insights** — Compare segments, test hypotheses, and create charts.
5. **Summarize & report** — Produce a traceable report with recommendations.

Data-Agent-specific core

Planning | Tool Use | Memory | Verification

3) Outcome

- Root causes identified
- Multi-source evidence
- Executive report + charts
- 3 prioritized actions

Evidence & Context Used by the Data Agent

- **Sales tables** — orders, revenue, product, region, time
- **Customer reviews** — ratings, comments, sentiment
- **Support tickets** — issue trends, status history
- **Business KPIs** — targets, budget, seasonality, churn

Why this is a Data Agent (not just an LLM agent)

- Combines structured and unstructured data
- Grounds reasoning in executable queries and evidence
- Delivers traceable, reproducible analysis

Data Agents vs. General LLM Agents

Formally, a data agent A operates on raw data D within an environment E (e.g., DBMS, code interpreters, APIs, etc.), utilizing LLMs M, ultimately producing an output O to tackle the data-related task T :

$$\boldsymbol{A: (T, D, E, M) \rightarrow O}$$

Aspect	General LLM Agents	Data Agents
Core Goal	• Automate general user tasks • Generate or organize content • Assist with planning, search, writing, and dialogue	• Solve data-centric tasks over data systems • Support analysis, optimization, operation, and governance • Produce verifiable data artifacts, e.g., SQL, reports, dashboards, workflows
Core Capabilities	• Reason over prompts and limited context • Invoke generic tools, e.g., search, calculator • Rely mainly on user inspection for error correction	• Ground reasoning in schemas, data values, metadata, and constraints • Execute over large, heterogeneous, and dynamic data sources • Detect, verify, repair, trace, and roll back errors before they propagate
Typical Applications	• Chatbots and personal assistants • Writing, summarization, search, and writing • General task planning and content generation	• Data intelligence: Text-to-SQL, Table QA, autonomous analysis • Business intelligence: dashboards, reports, decision support • Data systems: ETL, cleaning, tuning, optimization, governance

Data Agents: Errors Come in All Shapes and Sizes

One agent, many failure modes

timeout · hallucination · retry · wrong join

1. Perception & Intent Errors

- Intent misinterpretation
- Multimodal parsing failure
- Schema hallucination
  - id INT, name TEXT, age INT
- Missing implicit commonsense
  - Profit = revenue - cost
  - = 500 - 300
  - = 200 ✗ =-200

2. Planning & Orchestration Errors

- Long-horizon planning collapse
  - Start → Step1 → Step2 → Step3
- Infinite loops & ineffective retries
  - Retry ↕ Retry
- Wrong dependency ordering
  - A → B
- Wrong tool selection
  - SQL Query → Python

3. Interaction & Execution Errors

- Execution timeout & resource exhaustion
  - 00:00 timeout
- Missing runtime dependencies
  - `ModuleNotFoundError: No module named 'xxx'`
- Unauthorized access & security blocking
  - ACCESS DENIED Permission denied
- Context explosion
  - Prompt (120K tokens) … … … (truncated)

4. Evaluation & Output Errors

- Silent errors
  - SQL Result: region East, profit 9,999,999
- Visualization disasters
- Overconfidence & blind confirmation
  - The trend looks stable! Very confident!
- Looks successful, but actually wrong
  - Analysis complete! Everything looks normal! Conclusion: Profit in East China increased significantly! Recommendation: Keep the current strategy.
  - Actually wrong

Chaotic Failure Ecosystem (just the tip of the iceberg)

Semantic drift · Unit mismatch · Data quality issues · Null traps · Format disorder · Encoding issues · Timezone errors · Concurrency conflicts · Cache inconsistency · API rate limiting · Third-party service failure · Network instability · Result truncation · Missing logs · …

Data Agents: Errors Come in All Shapes and Sizes

One agent, many failure modes

timeout · hallucination · retry · wrong join

1. Perception & Intent Errors

- Intent misinterpretation
- Multimodal parsing failure
- Schema hallucination

id	name	age
INT	TEXT	INT

- Missing implicit commonsense

Profit = revenue - cost
= 500 - 300
= 200 ✗ =-200

2. Planning & Orchestration Errors

- Long-horizon planning collapse (Start → Step1 → Step2 → Step3)
- Infinite loops & ineffective retries (Retry ↕ Retry)
- Wrong dependency ordering (A → B)
- Wrong tool selection (SQL Query → Python)

3. Interaction & Execution Errors

- 00:00 timeout — Execution timeout & resource exhaustion
- `ModuleNotFoundError: No module named 'xxx'` — Missing runtime dependencies
- ACCESS DENIED Permission denied — Unauthorized access & security blocking
- Prompt (120K tokens) … … … (truncated) — Context explosion

4. Evaluation & Output Errors

- SQL Result: region = East, profit = 9,999,999 — Silent errors
- Visualization disasters
- "The trend looks stable! Very confident!" — Overconfidence & blind confirmation
- "Analysis complete! Everything looks normal! Conclusion: Profit in East China increased significantly! Recommendation: Keep the current strategy." (Actually wrong) — Looks successful, but actually wrong

Chaotic Failure Ecosystem (just the tip of the iceberg)

Semantic drift · Unit mismatch · Data quality issues · Null traps · Format disorder · Encoding issues · Timezone errors · Concurrency conflicts · Cache inconsistency · API rate limiting · Third-party service failure · Network instability · Result truncation · Missing logs · …

Data Agents: Errors Come in All Shapes and Sizes

One agent, many failure modes

timeout · hallucination · retry · wrong join

1. Perception & Intent Errors

- Intent misinterpretation
- Multimodal parsing failure
- Schema hallucination
- Missing implicit commonsense

Profit = revenue - cost
= 500 - 300
= 200 ✗ =-200

2. Planning & Orchestration Errors

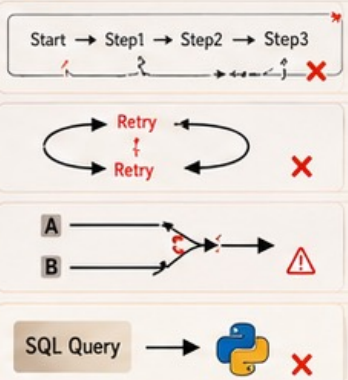
- Long-horizon planning collapse
- Infinite loops & ineffective retries
- Wrong dependency ordering
- Wrong tool selection

3. Interaction & Execution Errors

- Execution timeout & resource exhaustion (00:00 timeout)
- Missing runtime dependencies (`ModuleNotFoundError: No module named 'xxx'`)
- Unauthorized access & security blocking (ACCESS DENIED — Permission denied)
- Context explosion (Prompt (120K tokens) … … … (truncated))

4. Evaluation & Output Errors

- Silent errors
- Visualization disasters
- Overconfidence & blind confirmation
- Looks successful, but actually wrong



Chaotic Failure Ecosystem (just the tip of the iceberg)

Semantic drift · Unit mismatch · Data quality issues · Null traps · Format disorder · Encoding issues · Timezone errors · Concurrency conflicts · Cache inconsistency · API rate limiting · Third-party service failure · Network instability · Result truncation · Missing logs · …

Data Agents: Errors Come in All Shapes and Sizes

One agent, many failure modes

timeout · hallucination · retry · wrong join

1. Perception & Intent Errors

- Intent misinterpretation
- Multimodal parsing failure
- Schema hallucination
- Missing implicit commonsense

Profit = revenue - cost
= 500 - 300
= 200 ✗ =-200

2. Planning & Orchestration Errors

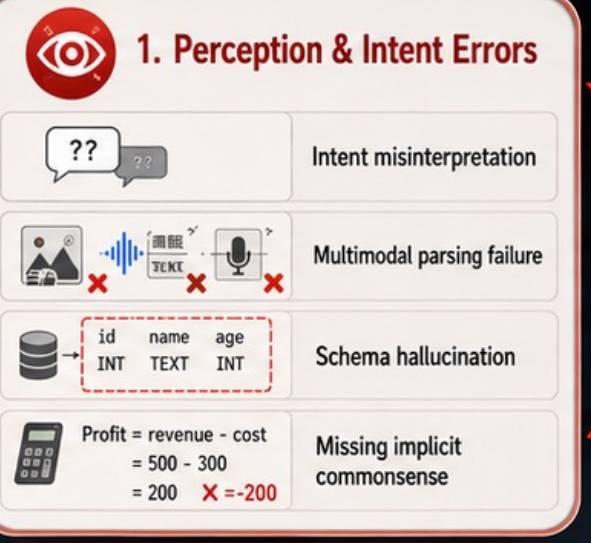
- Long-horizon planning collapse
- Infinite loops & ineffective retries
- Wrong dependency ordering
- Wrong tool selection

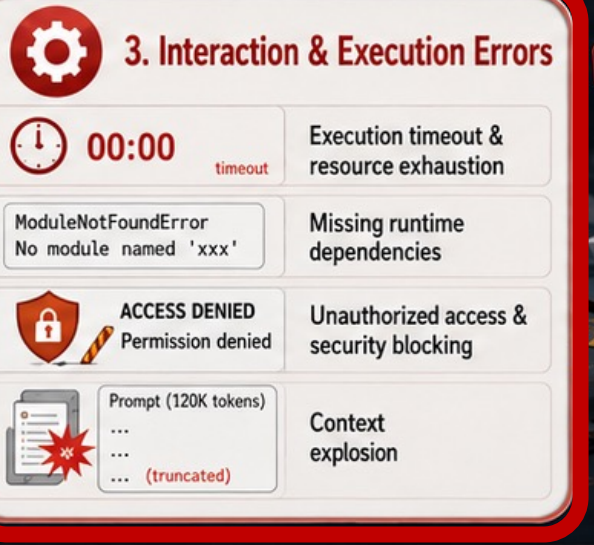
3. Interaction & Execution Errors

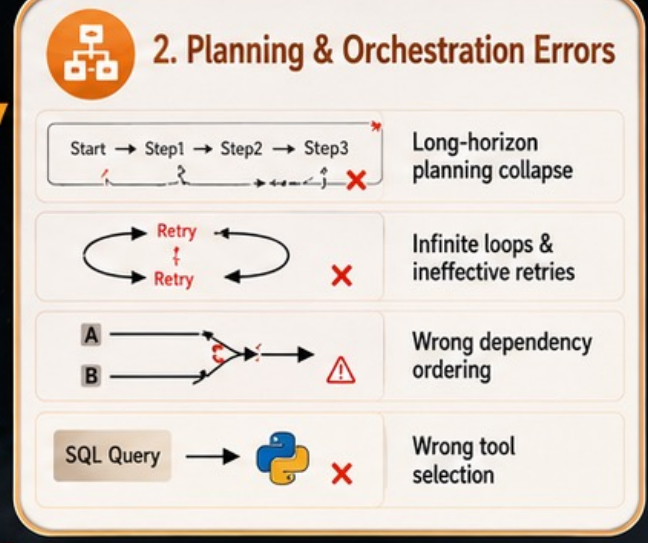
- Execution timeout & resource exhaustion
- Missing runtime dependencies
- Unauthorized access & security blocking
- Context explosion

4. Evaluation & Output Errors

- Silent errors
- Visualization disasters
- Overconfidence & blind confirmation
- Looks successful, but actually wrong







Chaotic Failure Ecosystem (just the tip of the iceberg)

Semantic drift · Unit mismatch · Data quality issues · Null traps · Format disorder · Encoding issues · Timezone errors · Concurrency conflicts · Cache inconsistency · API rate limiting · Third-party service failure · Network instability · Result truncation · Missing logs · ...

Data Agents: Errors Come in All Shapes and Sizes

One agent, many failure modes

timeout · hallucination · retry · wrong join

1. Perception & Intent Errors

- Intent misinterpretation
- Multimodal parsing failure
- Schema hallucination
- Missing implicit commonsense

Profit = revenue - cost
= 500 - 300
= 200 ✗ =-200

2. Planning & Orchestration Errors

- Long-horizon planning collapse
- Infinite loops & ineffective retries
- Wrong dependency ordering
- Wrong tool selection

3. Interaction & Execution Errors

- Execution timeout & resource exhaustion
- Missing runtime dependencies
- Unauthorized access & security blocking
- Context explosion

4. Evaluation & Output Errors

- Silent errors
- Visualization disasters
- Overconfidence & blind confirmation
- Looks successful, but actually wrong

SQL Result

region	profit
East	9,999,999

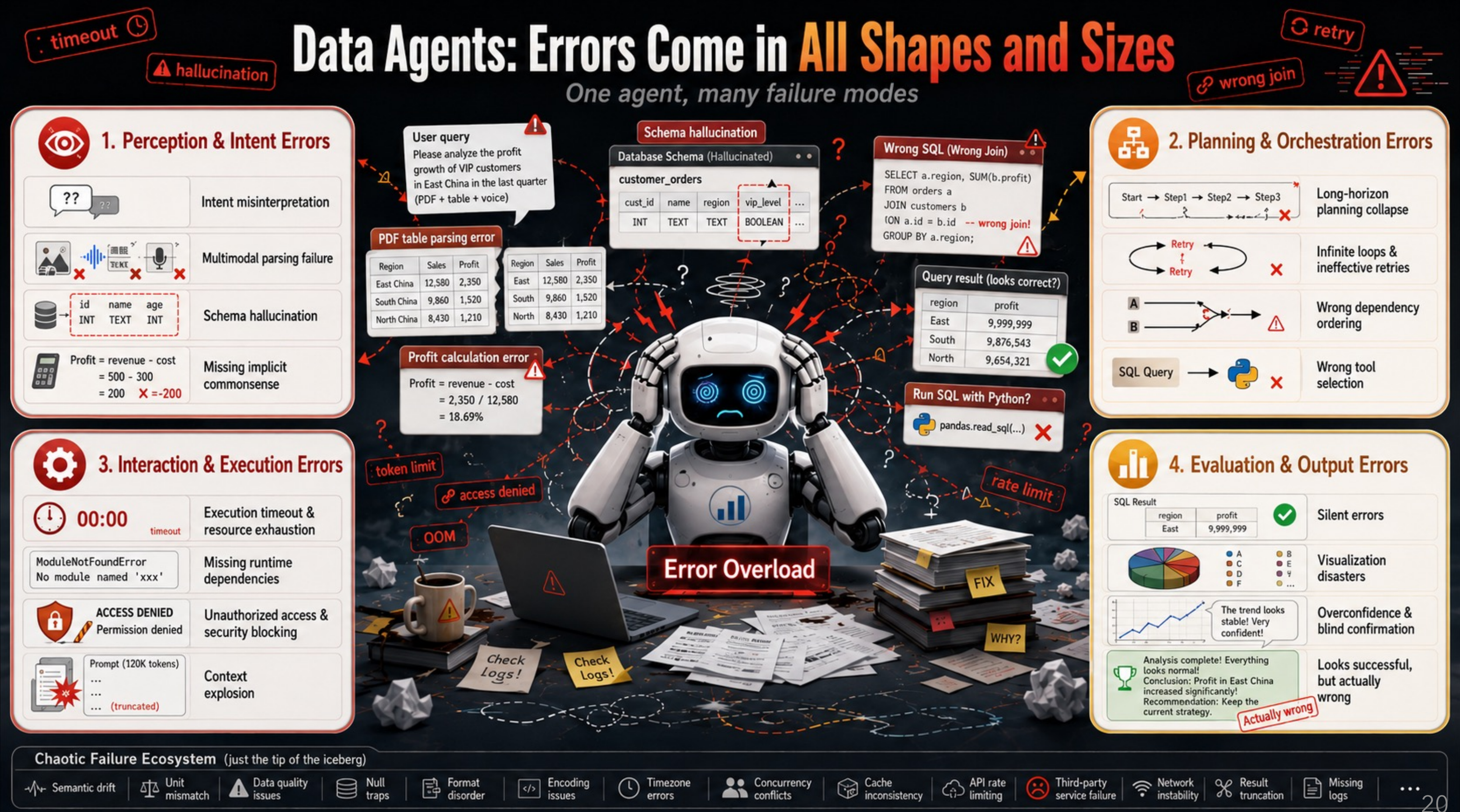
The trend looks stable! Very confident!

Analysis complete! Everything looks normal!
Conclusion: Profit in East China increased significantly!
Recommendation: Keep the current strategy.

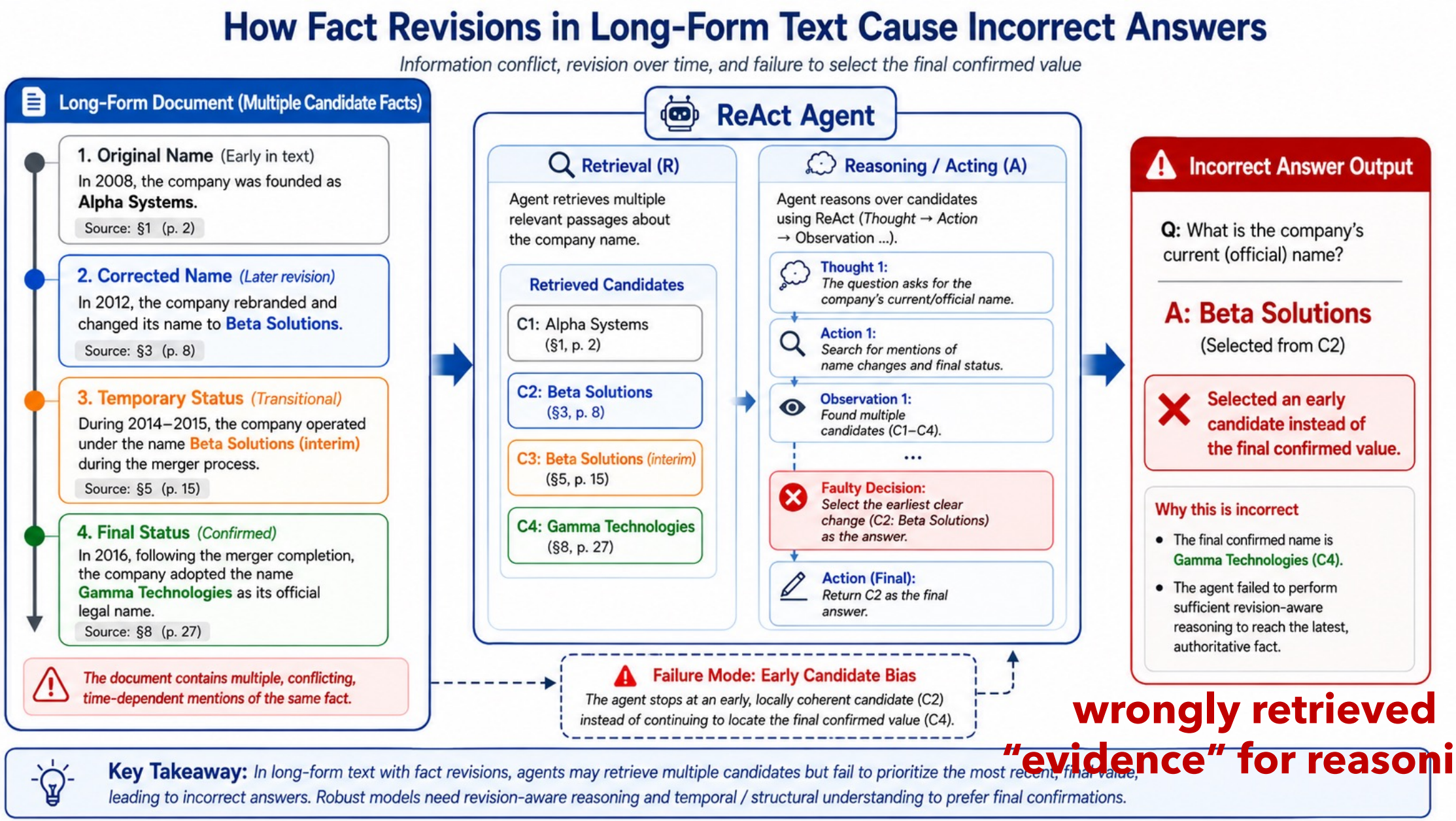
Actually wrong

Chaotic Failure Ecosystem (just the tip of the iceberg)

Semantic drift · Unit mismatch · Data quality issues · Null traps · Format disorder · Encoding issues · Timezone errors · Concurrency conflicts · Cache inconsistency · API rate limiting · Third-party service failure · Network instability · Result truncation · Missing logs · ...


Data Agents: Errors Come in All Shapes and Sizes
One agent, many failure modes
timeout
hallucination
retry
wrong join
1. Perception & Intent Errors
??
Intent misinterpretation
Multimodal parsing failure
id name age
INT TEXT INT
Schema hallucination
Profit = revenue - cost
= 500 - 300
= 200 ✗ =-200
Missing implicit commonsense
User query
Please analyze the profit growth of VIP customers in East China in the last quarter (PDF + table + voice)
Schema hallucination
Database Schema (Hallucinated)
customer_orders
cust_id name region vip_level ...
INT TEXT TEXT BOOLEAN ...
Wrong SQL (Wrong Join)
SELECT a.region, SUM(b.profit)
FROM orders a
JOIN customers b
ON a.id = b.id -- wrong join!
GROUP BY a.region;
PDF table parsing error
Region Sales Profit
East China 12,580 2,350
South China 9,860 1,520
North China 8,430 1,210
Region Sales Profit
East 12,580 2,350
South 9,860 1,520
North 8,430 1,210
Query result (looks correct?)
region profit
East 9,999,999
South 9,876,543
North 9,654,321
Profit calculation error
Profit = revenue - cost
= 2,350 / 12,580
= 18.69%
Run SQL with Python?
pandas.read_sql(...)
2. Planning & Orchestration Errors
Start → Step1 → Step2 → Step3
Long-horizon planning collapse
Retry
Retry
Infinite loops & ineffective retries
A
B
Wrong dependency ordering
SQL Query
Wrong tool selection
token limit
access denied
OOM
rate limit
Error Overload
FIX
WHY?
Check logs!
Check Logs!
3. Interaction & Execution Errors
00:00
timeout
Execution timeout & resource exhaustion
ModuleNotFoundError
No module named 'xxx'
Missing runtime dependencies
ACCESS DENIED
Permission denied
Unauthorized access & security blocking
Prompt (120K tokens)
...
...
... (truncated)
Context explosion
4. Evaluation & Output Errors
SQL Result
region profit
East 9,999,999
Silent errors
A B C E D F ...
Visualization disasters
The trend looks stable! Very confident!
Overconfidence & blind confirmation
Analysis complete! Everything looks normal!
Conclusion: Profit in East China increased significantly!
Recommendation: Keep the current strategy.
Actually wrong
Looks successful, but actually wrong
Chaotic Failure Ecosystem (just the tip of the iceberg)
Semantic drift
Unit mismatch
Data quality issues
Null traps
Format disorder
Encoding issues
Timezone errors
Concurrency conflicts
Cache inconsistency
API rate limiting
Third-party service failure
Network instability
Result truncation
Missing logs
...

The errors take many different forms



wrongly retrieved "evidence" for reasoning

The errors take many different forms

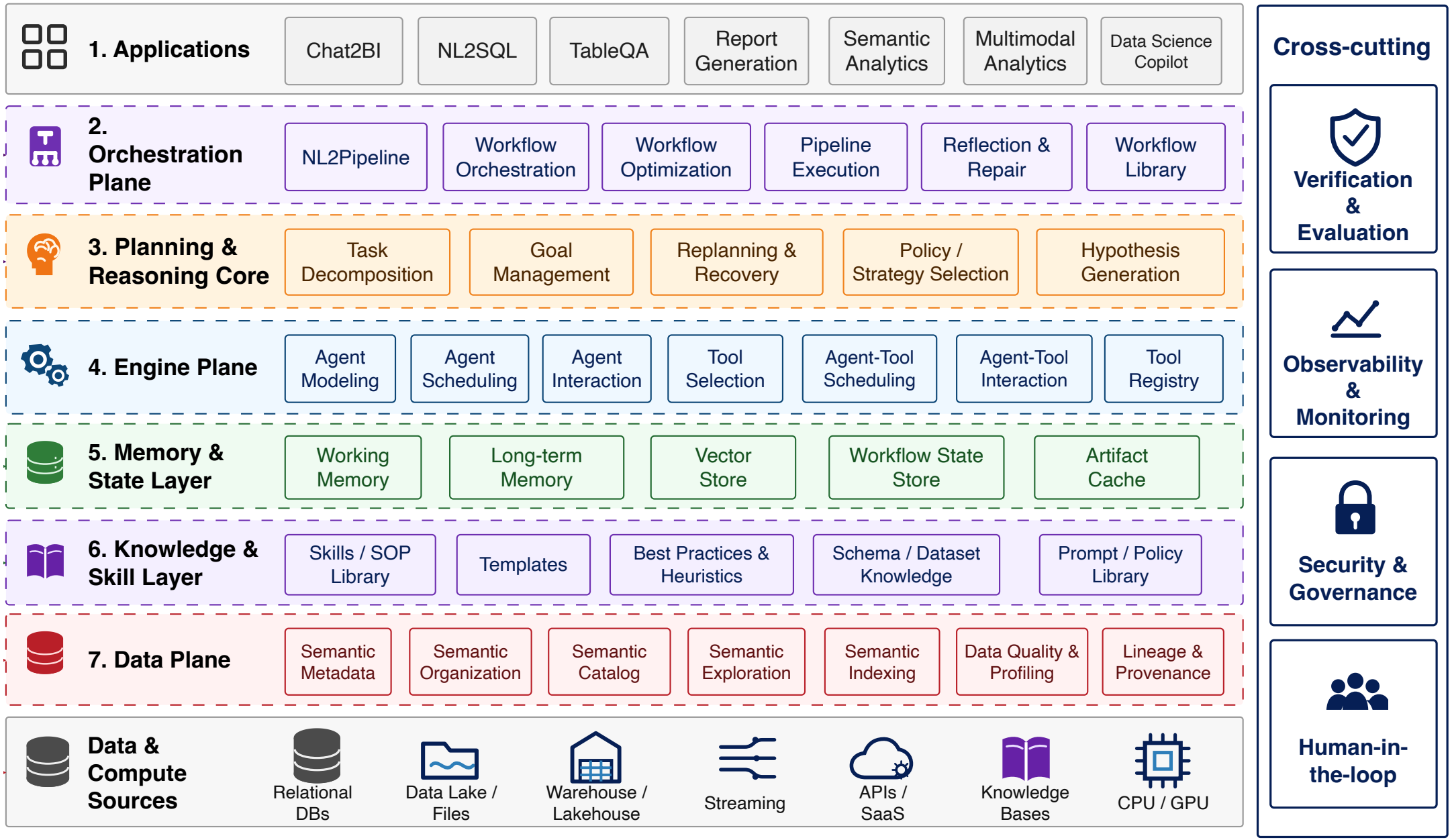
Example 2: Reasoning & Execution Trace

Step	Agent Thought (Summary)	Tool Call	Parameters / Code Logic	Observation / Result
1	List available files to understand data sources.	list_context	max_depth: 4	✓ Found db/postHistory.db, json/posts.json, json/users.json, knowledge.md
2	Read the knowledge guide to understand schema.	read_doc	path: knowledge.md max_chars: 4000	✓ Understood Posts, Users, postHistory structure
3	Inspect SQLite database schema.	inspect_sqlite_schema	path: db/postHistory.db	✓ Found table postHistory (Id, PostId, LastEditorUserId, CreationDate ...)
4	Try reading posts.json.	read_json	path: json/posts.json max_chars: 4000	✗ Failed: 'utf-8' codec can't decode byte 0xb3 in position 28
5	Try reading users.json.	read_json	path: json/users.json max_chars: 4000	✗ Failed: 'utf-8' codec can't decode byte 0xda in position 17
6	Handle JSON encoding issue with Python.	execute_python	Open files in 'rb' mode, inspect bytes to diagnose encoding	✓ Confirmed files are encoded in UTF-8 (invalid bytes ignored)
7	Search postHistory for the target post title.	execute_context_sql	SELECT * FROM postHistory WHERE Title LIKE '%Computer Game Datasets%'	✓ Found PostId = 8222 LastEditorUserId = 88
8	Load posts.json with UTF-8 and find post details.	execute_python	Load posts.json (UTF-8), find Id = 8222	✓ ViewCount = 1708 OwnerUserId = 37
9	Get user info for owner and last editor.	execute_python	Load users.json (UTF-8), lookup Id = 37 and Id = 88	✓ Owner: 37 (Tim Stone) Last editor: 88 (mbq)
10	Determine last user who posted it.	execute_python	Use LastEditorUserId = 88	✓ Last user: mbq
11	Return final answer.	answer	columns: ["total_views", "last_user"]	✓ Result: total_views = 1708 last_user = mbq

Exploration / Doc | Failed Attempt | Python / SQL Execution | Final Answer

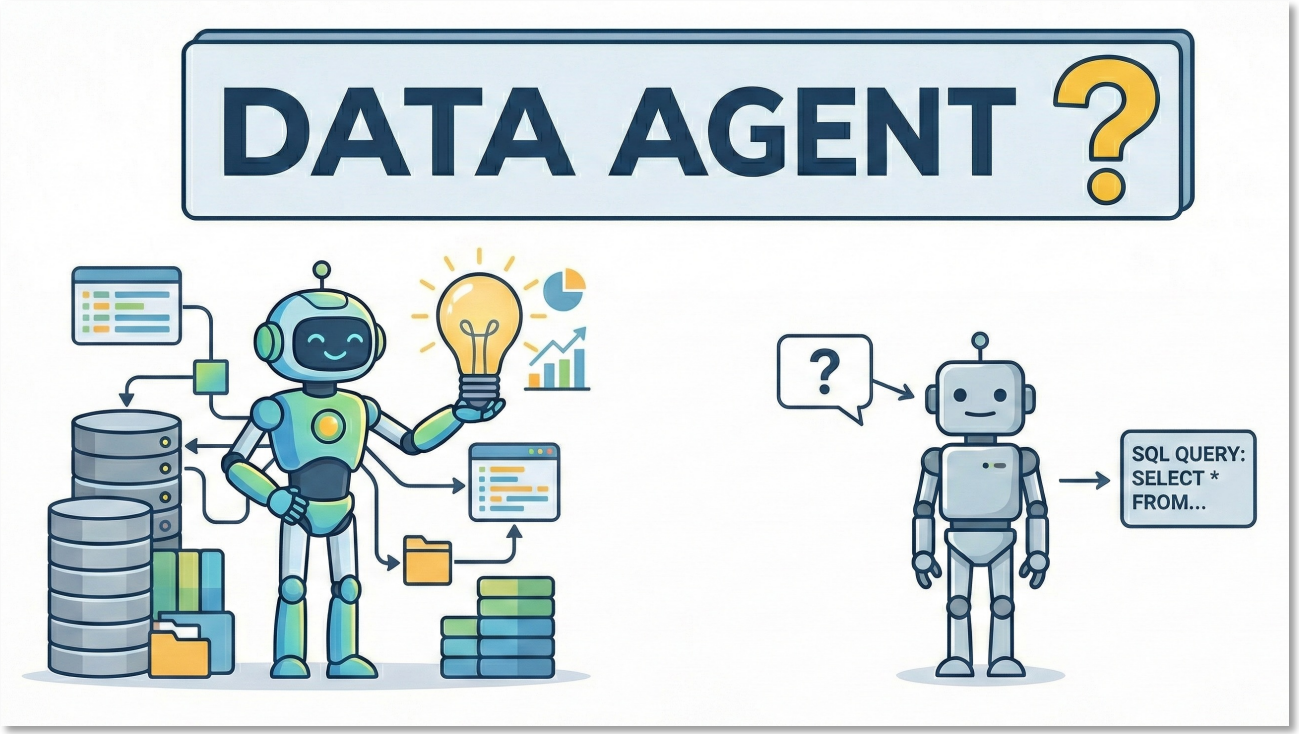
Seemingly impossible low-level file-parsing errors!

Data Agent: Modules



The Terminological Ambiguity of Data Agents

- The term “Data Agent” is applied inconsistently:
  - Sophisticated agentic data systems to autonomously interact with data lakes, invoke external tools, orchestrate and optimize tailored pipelines for complex data-related tasks
  - More rudimentary, narrowly scoped systems acting as simple query responders



Conflate systems of different autonomy, reliability, and complexity under a imprecisely defined umbrella term.

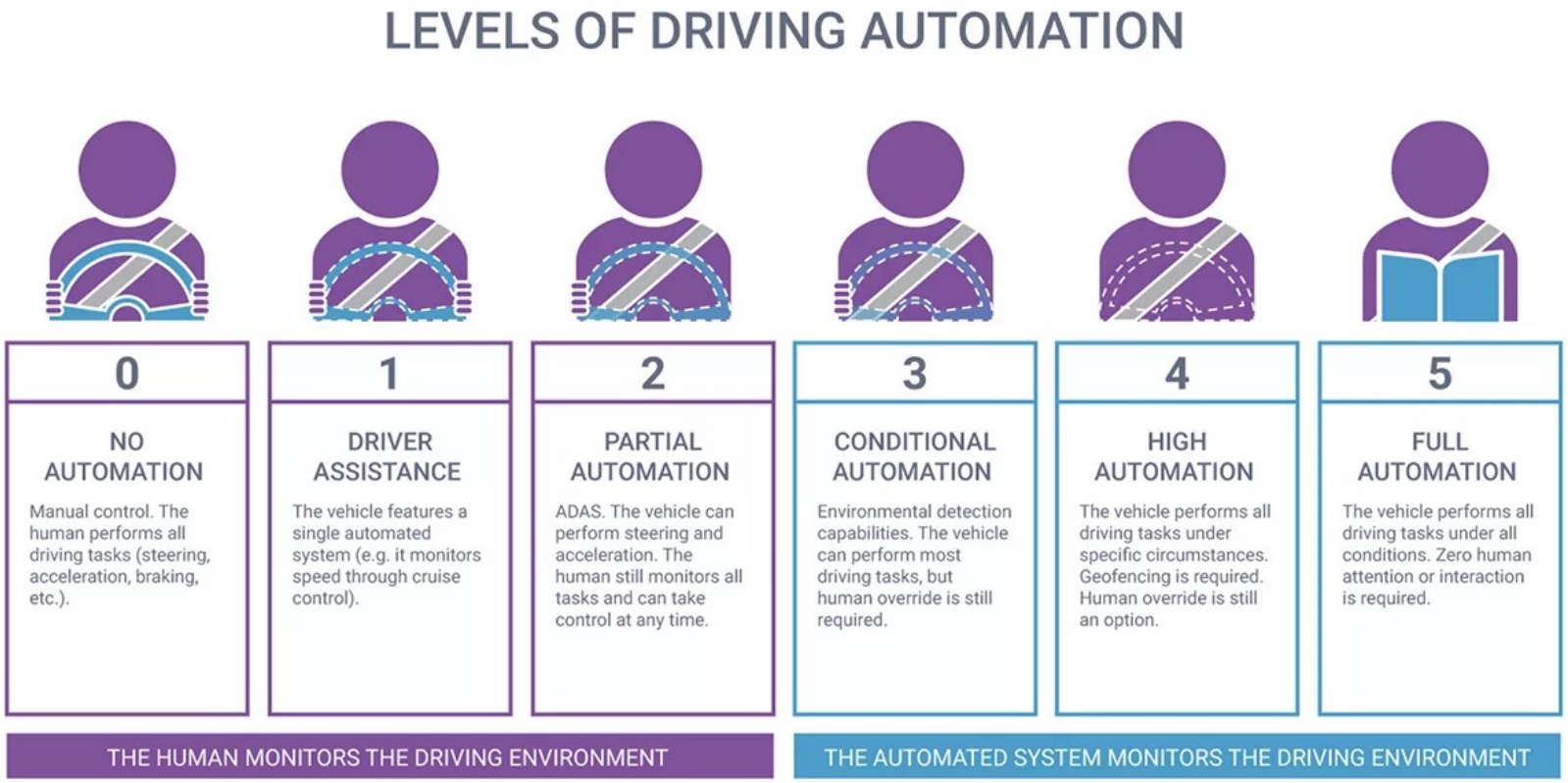
- **User-Side Risk:** User expectation mismatch
- **Governance Risk:** Unclear accountability
- **Industry-Side Risk:** Exaggeration and hype

Hierarchical Taxonomy for Data Agents

Precedent in Self-Driving

- Such a terminological ambiguity is not unprecedented:

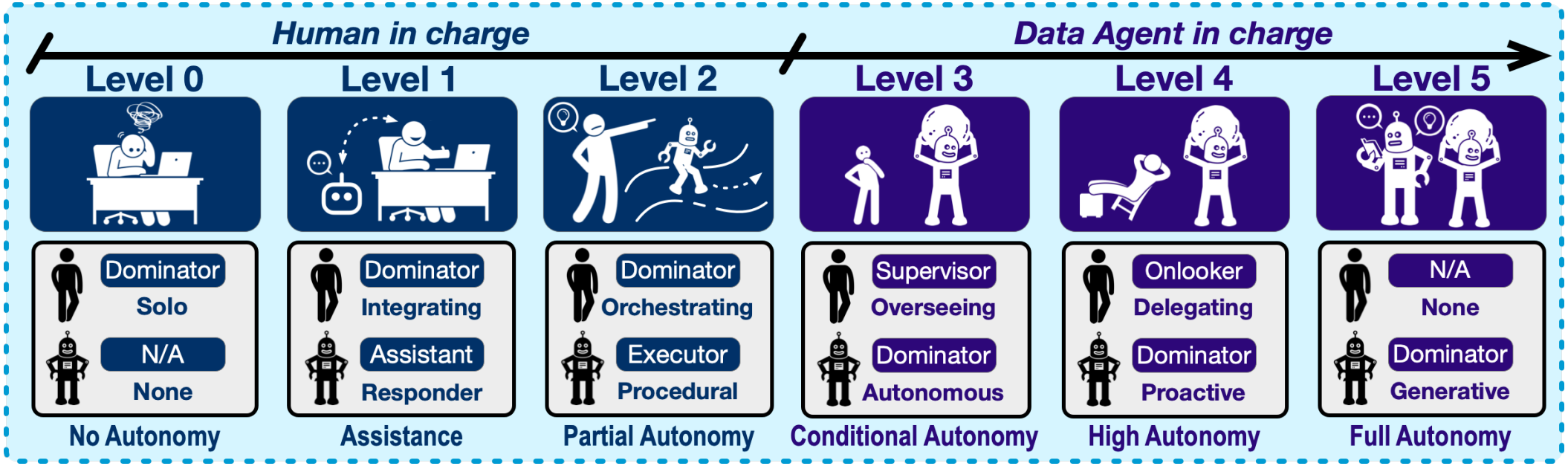
  Automotive industry and driving automation community had encountered similar challenges
- SAE introduced the J3016 standard, a six-level taxonomy for driving automation



https://www.synopsys.com/blogs/chip-design/autonomous-driving-levels.html

Hierarchical Taxonomy for Data Agents

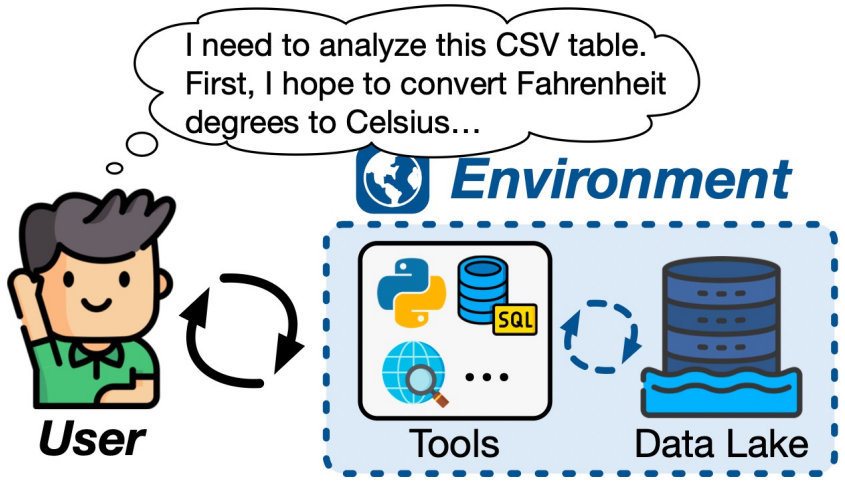
We Advocate a Hierarchical Taxonomy for Data Agents



- Map the progressive transitions of dominance and responsibility in data-related tasks from human to data agent as autonomy increases from L0 to L5
- Unified framework to compare existing works, delineating capability boundaries, and clarifying accountability, enabling practitioners to align expectations and intervention with autonomy levels.
- We will elaborate on the formal definition for each level in the following

L0: Manual Labor in Early Ages

Human-driven Data Management, Preparation, Analysis



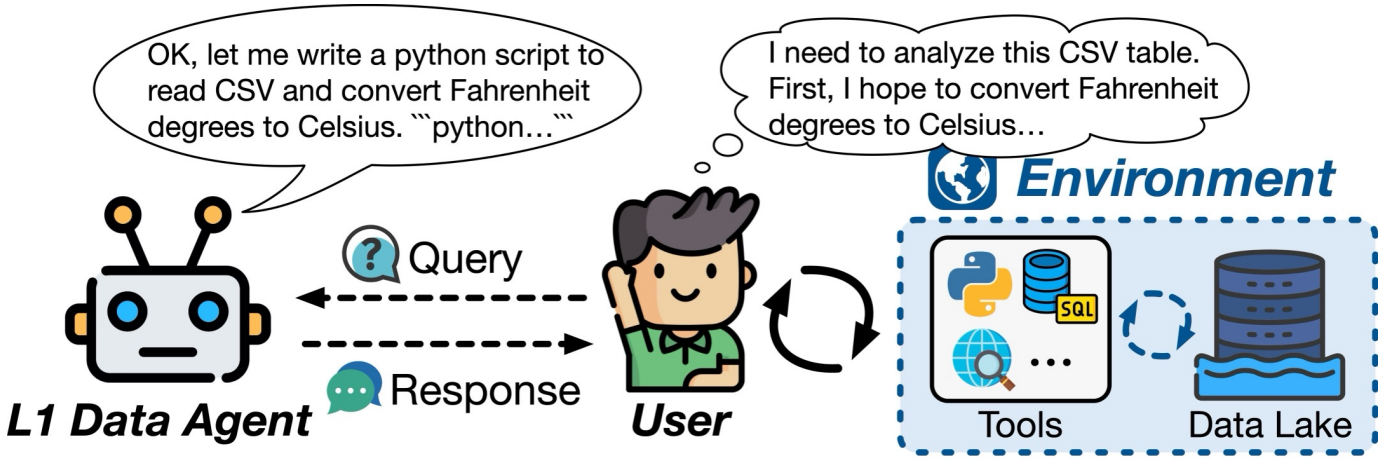
- Conventionally, all data management, preparation, and analysis tasks are performed entirely by humans without intelligent assistance.
- Formally, the human H is responsible for the entire process, orchestrating (π_H) pipeline P and executing (ϵ_H), while the data agent A is uninvolved yet:

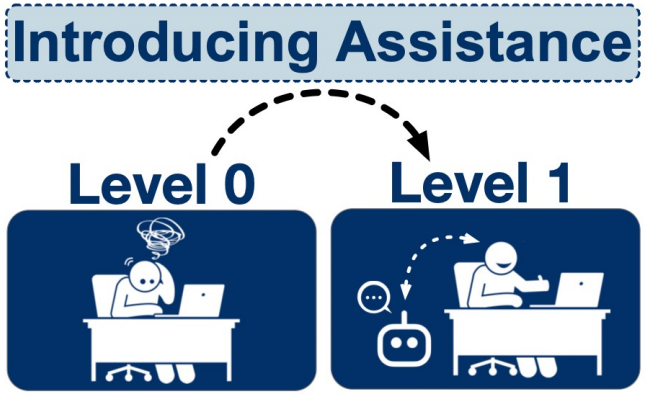
$$H: \pi_H(T, D, E) \rightarrow P; \epsilon_H(P, D, E) \rightarrow O$$

$$A: \emptyset$$

L1: Preliminary Assistance

Definition for L1 Data Agents (Assistance)





- Align with the early wave of LLM assistants
- Prompt-response paradigm: data agents act as nascent, stateless query-responsive assistants
- Incapable of perceiving and interacting with the environment
- Formally, the human H remains responsible for both pipeline orchestration (π_H) and execution (ϵ_H), while data agent A can respond r upon human query q for assistance

$$H: \pi_H(T, D, E) \rightarrow P; \epsilon_H(P, D, E, r) \rightarrow O.$$

$$A: (q, M) \rightarrow r$$

L1: Preliminary Assistance (Progress and Limitations)

Progress: Efficiency in Routine Tasks

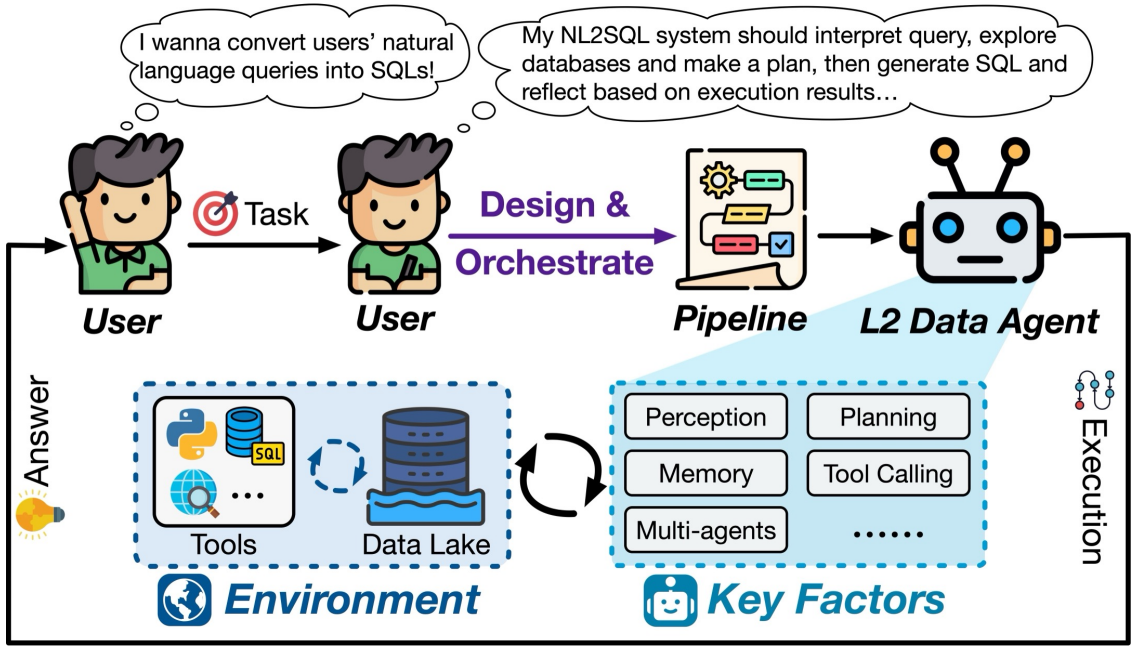
- **Query-Responsive Assistance:** interpret and respond to user queries on demand
- **Efficiency Boost:** improve efficiency by offloading trivial and routine operations, such as unit conversions or standard preprocessing code generation.
- **Lowering Barriers:** lower comprehension barriers for non-technique or novice users

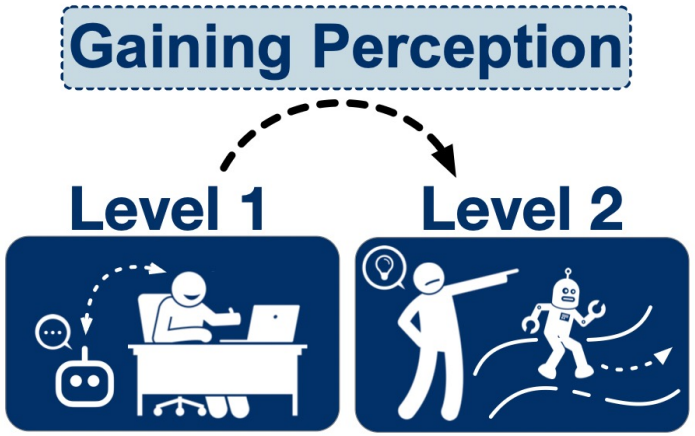
Limitations of L1 Data Agents

- **Stateless Nature:** operate in a "prompt-response" paradigm without maintaining state over time.
- **Lack of Perception:** unable to perceive or interact with the external environment (databases, APIs) autonomously, preventing a closed-loop refinement and optimization
- **Human Dependency:** The human users still manually execute, integrate, and verify outputs. Data agents cannot perform end-to-end procedures, limiting autonomy to atomic, static subtasks.

L2: Perceive the Environment

Definition for L2 Data Agents (Partial Autonomy)





- Data agents can perceive and interact with the environment (e.g., data lakes, DBMS, code interpreters, APIs, etc.), enabling partial autonomy to perform task-specific procedures independently.
- Data agents operate within human-orchestrated pipelines
- The data agent A gains environmental perception and interaction capabilities (D, E), capable of handling specific data-related tasks by executing (ϵ_A) pipeline P orchestrated by human H:

$$H: \pi_H(T, D, E) \rightarrow P. \qquad A: \epsilon_A(P, D, E, M) \rightarrow O$$

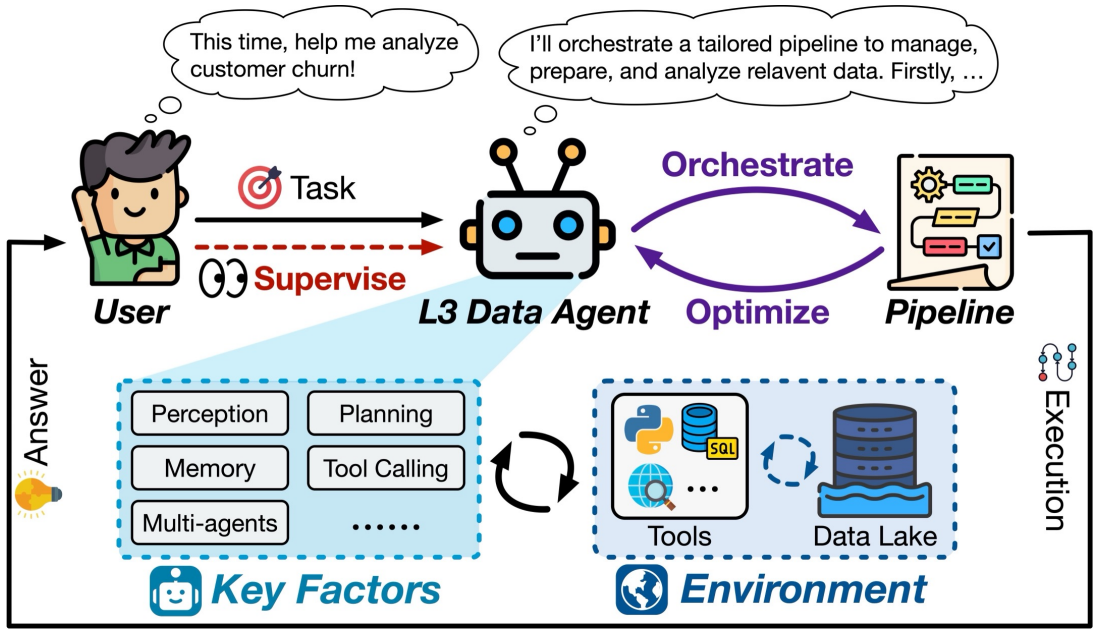
L2: Perceive the Environment

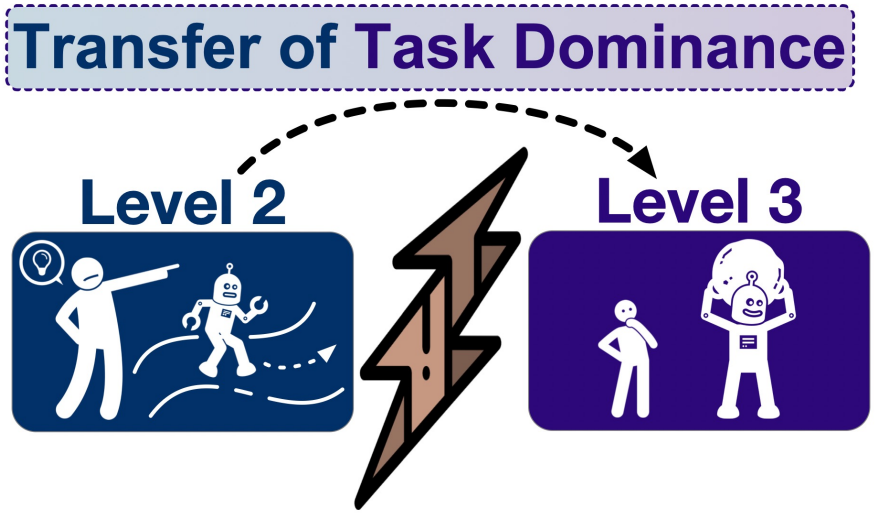
The Glass Ceiling of L2 Data Agents (Limitations)

- **Progress – Perception & Interaction:**
  - L2 data agents can connect to real-world systems, autonomously executing specific procedures and optimizing based on environmental feedback
- **Dependence on Human-Designed Pipelines:**
  - L2 data agents comply with pre-established pipelines orchestrated by humans, lacking the ability to independently orchestrate task-tailored new pipelines
  - L2 data agents operate within human-crafted agentic modules, architectures, and collaboration mechanisms
- **Task-Specific Rigidity:**
  - Systems are closely tied to specific tasks/domains (e.g., modules specialized only for NL2SQL)
  - Lack of versatility and generalizability to handle diverse and comprehensive tasks that potentially span the full data lifecycle in real-world scenarios

L3: Striving for Autonomous Data Agents

Definition for L3 Data Agents (Conditional Autonomy)





- Data agents autonomously orchestrate and optimize pipelines rather than following human-defined ones; managing diverse and comprehensive tasks potentially spanning the entire data lifecycle, rather than isolated and task-specific procedures
- Critical Leap: data agents assume task dominance from L3, while humans oversee the process.
- Formally, the data agent A autonomously manages the entire pipeline from orchestration π_A to execution ϵ_A, tackling versatile and comprehensive data-related tasks T under human H supervision:

$$A: \pi_A(T, D, E, M) \rightarrow P;\ \epsilon_A(P, D, E, M) \rightarrow O. \quad H: Supervise(\pi_A, \epsilon_A)$$

L3: Striving for Autonomous Data Agents

Challenges and Research Opportunities Towards True L3

- **Limited Autonomy in Orchestration:**
  - **Challenge:** Current Proto-L3 systems still rely on predefined operators/tools.
  - **Opportunity:** Skill Discovery [1] and Autonomous Operator Synthesis [2,3]. Autonomously generate, evaluate, and curate new tools/skills dynamically.
- **Incomplete Data Lifecycle Coverage:**
  - **Challenge:** Existing agents focus narrowly (mostly on Analysis, or involve basic preparation), largely neglecting Data Management (tuning, diagnosis) and broader Data Preparation.
  - **Opportunity:** Versatile Generalists. Handle the diverse and comprehensive data-related tasks across the full spectrum in the data lifecycle: Management → Preparation → Analysis.

[1] Sun Z, Wang J, Zhao X, et al. Data Agent: A Holistic Architecture for Orchestrating Data+ AI Ecosystems
[2] Sun J, Li G, Zhou P, et al. AgenticData: An Agentic Data Analytics System for Heterogeneous Data[J]. arXiv, 2025.
[3] Wang J, Li G, Feng J. iDataLake: An LLM-powered Analytics System on Data Lakes[J]. Data Engineering, 2025

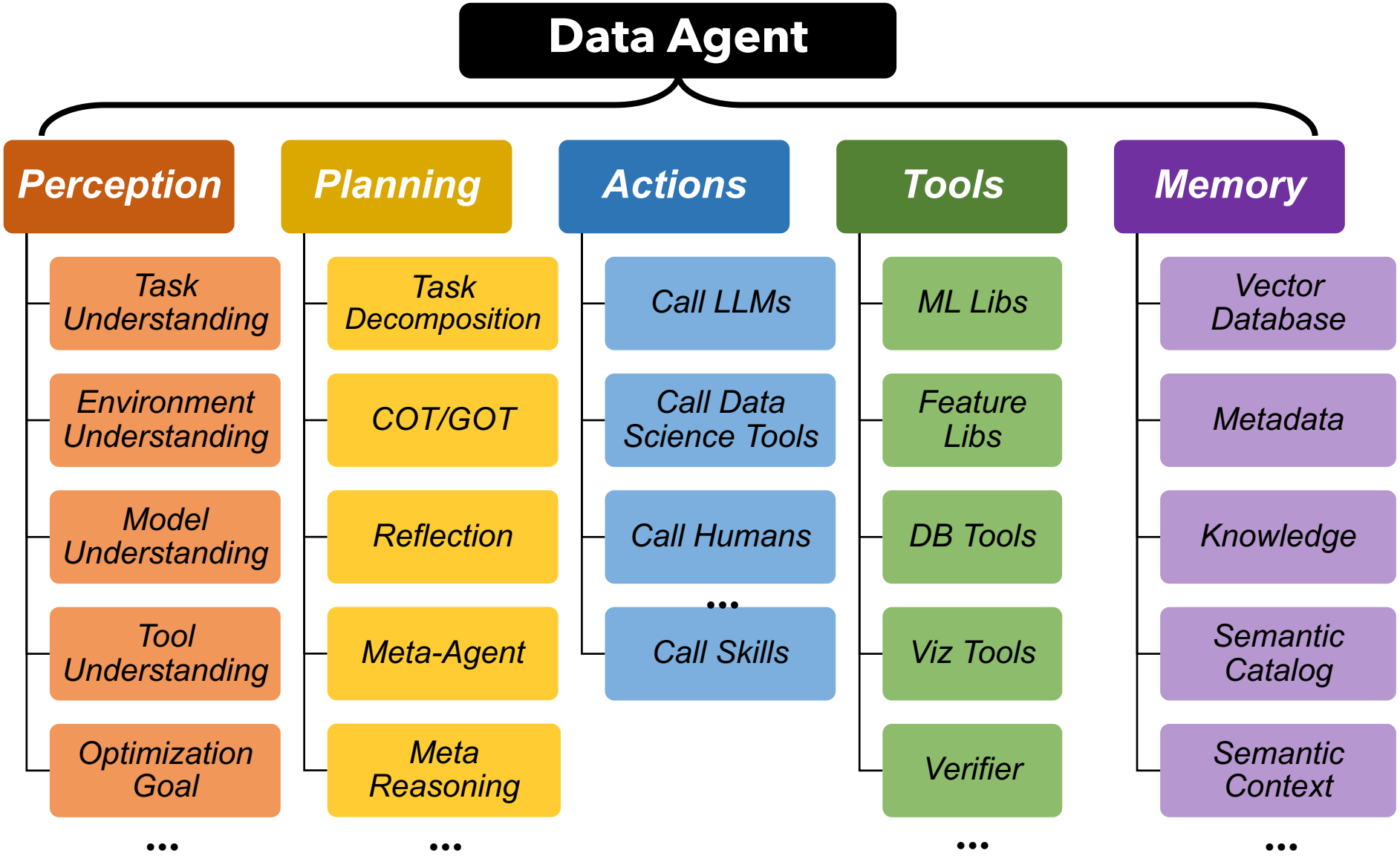
L3: Striving for Autonomous Data Agents

Challenges and Research Opportunities Towards True L3 (Cont'd)

- **Deficiencies in Advanced Reasoning:**
  - **Challenge:** Trapped in tactical fixes and unproductive loops due to a lack of strategic reflection.
  - **Opportunity:** Meta-Reasoning. Incorporating causal reasoning, meta-reasoning for cross-process optimization, and sophisticated memory architectures for abstract strategic knowledge.
- **Adaptation to Dynamic Environments:**
  - **Challenge:** Current evaluations use static data, ignoring real-world data drift.
  - **Opportunity:** Self-Evolution and Dynamic Benchmarking.
    - Enable data agents to adapt to evolving data environments without human intervention.
    - Establish and simulate a dynamic environment to rigorously evaluate and benchmark data agents' robustness under changing conditions

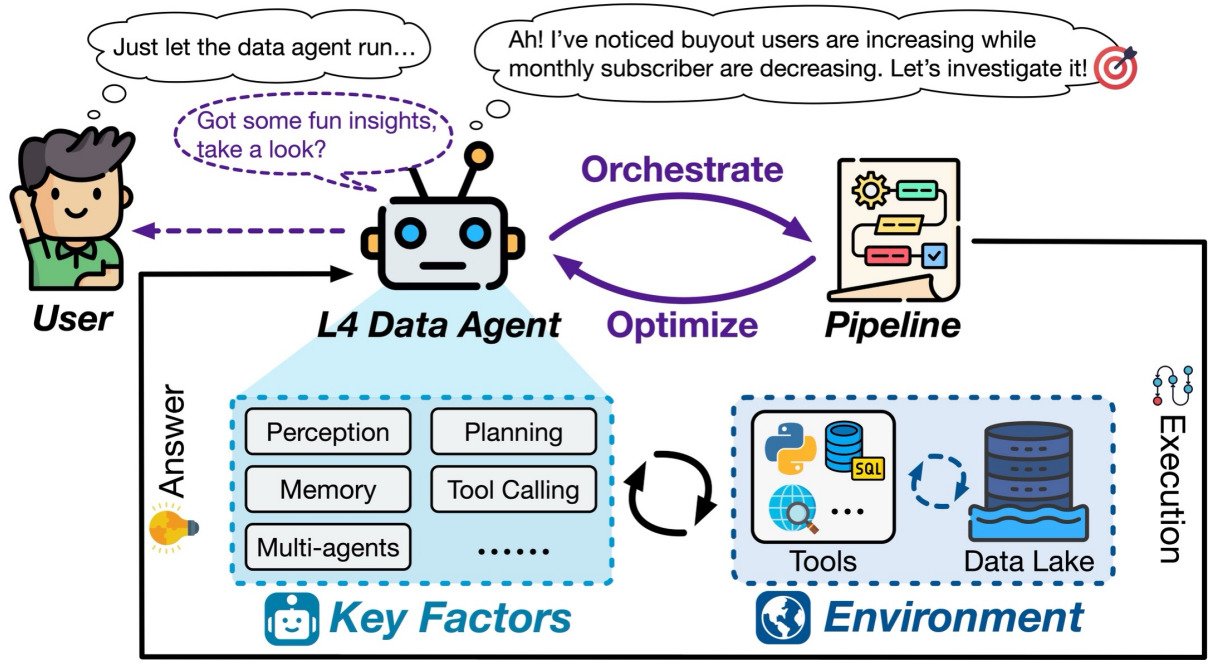
L3: Striving for Autonomous Data Agents

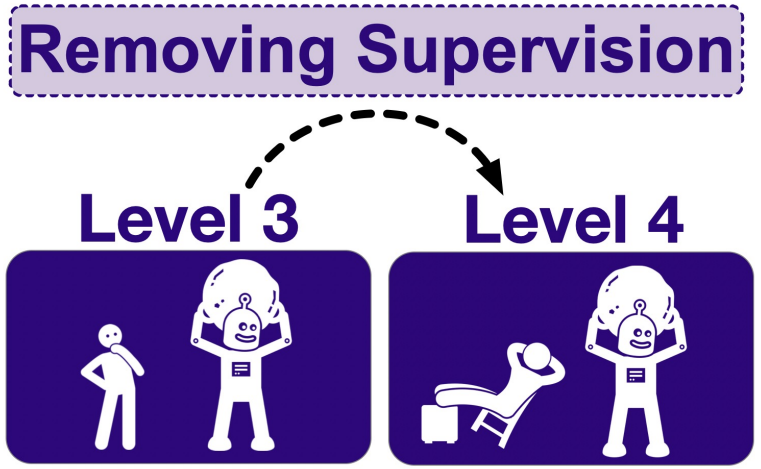
Data agents need fundamental breakthroughs to achieve L3 autonomy.



L4-L5: Vision of Proactive and Generative Data Agents

Definition for L4 Data Agents (High Autonomy)





- Data agents autonomously monitor and explore data lakes to proactively identify valuable and emerging tasks, rather than simply responding to given goals/instructions.
- Data agents present high reliability, no supervision is needed, and humans just receive the output.
- Formally, the data agent A takes full initiative, not only orchestrates π_A and executes ϵ_A pipeline P but also autonomously discovers task T' to begin with:

$$A: Discover_A(D, E, M) \rightarrow T';\ \pi_A(T', D, E, M) \rightarrow P;\ \epsilon_A(P, D, E, M) \rightarrow O. \quad H: Receive(O)$$

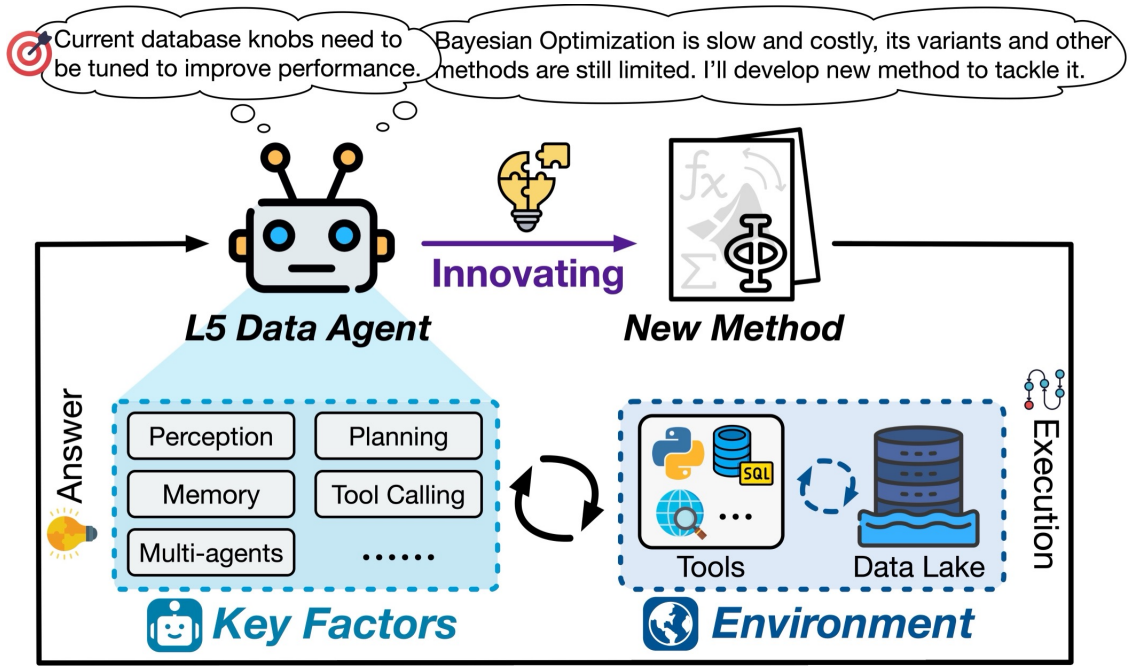
L4-L5: Vision of Proactive and Generative Data Agents

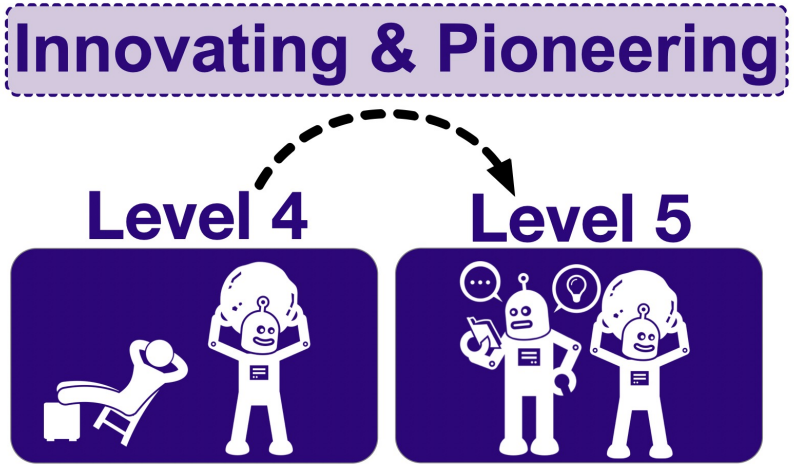
Research Directions Towards L4

- **Autonomous Problem Discovery:**
  - Move beyond execution to critical evaluation. Identifying anomalies, gaps, or emerging tasks without explicit task instruction
  - Research Direction: Developing *Task-Oriented Awareness* and *Intrinsic Curiosity*.
- **Trustworthy Self-Governance:**
  - Operate as reliable generalists that orchestrate robust pipelines tailored for self-discovered tasks, and self-manage resources, security, and accuracy without human oversight.
  - Research Direction: Robust effectiveness, efficiency, and safety guarantees
- **Long-Horizon & Holistic Planning:**
  - Make strategic trade-offs (e.g., balancing immediate cleaning costs vs. long-term analytical accuracy).
  - Research Direction: Capabilities for long-term planning and strategic decision-making, beyond local optimizations

L4-L5: Vision of Proactive and Generative Data Agents

Definition for L5 Data Agents (Full Autonomy)

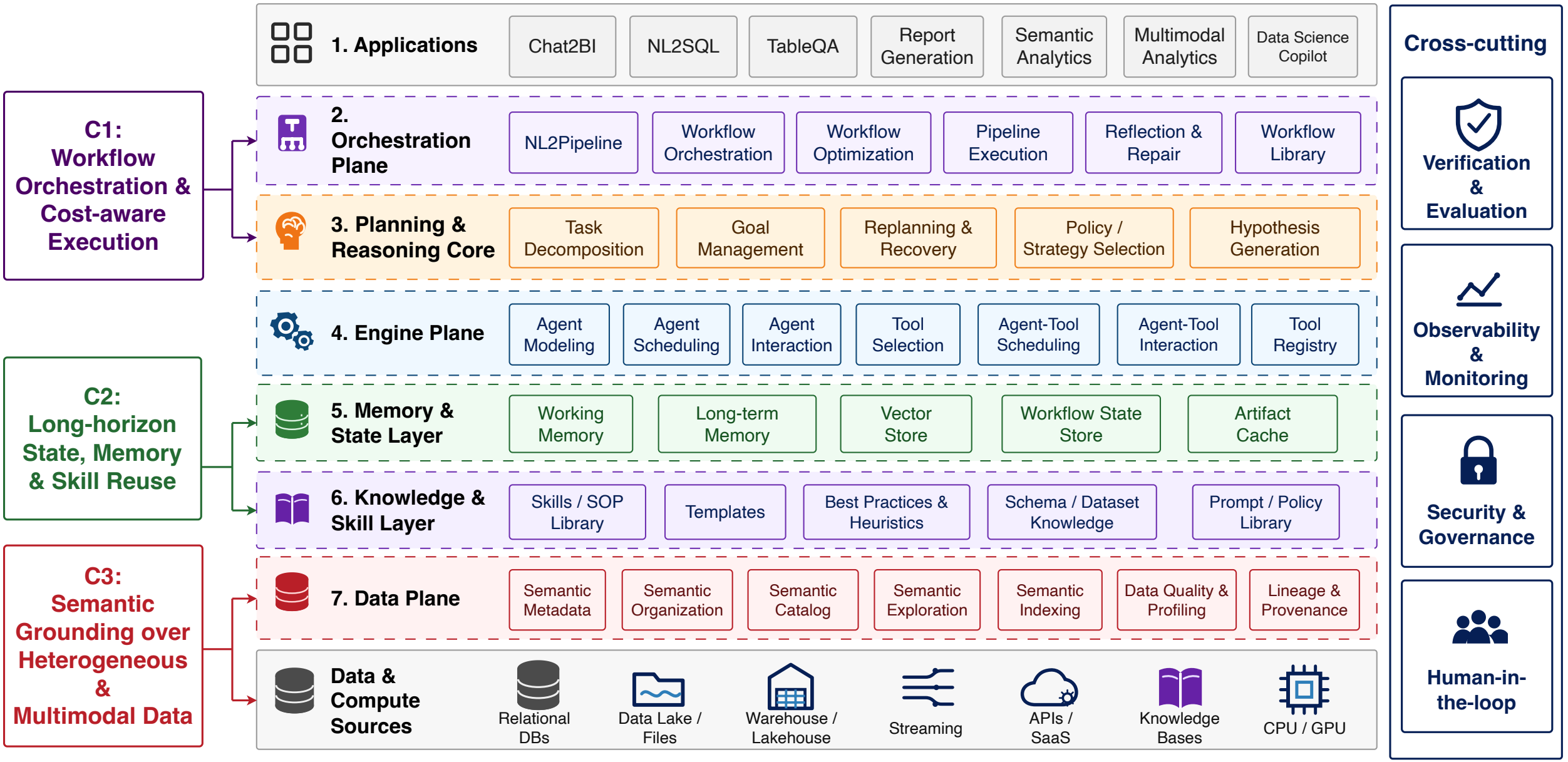




- Beyond merely applying existing methods, Data agents actively create new knowledge by identifying when conventional approaches are insufficient and innovating novel solutions.
- Formally, the data agent A not only autonomously identifies the promising task T' but also invents a new method Φ (e.g., a new theory, algorithm, or paradigm) to address it, while human H disengages:

$$A: Discover_A(D, E, M) \rightarrow T'; \quad Innovate_A(T', D, E, M) \rightarrow \Phi; \quad \Phi(T', D, E, M) \rightarrow O. \qquad H: \emptyset$$

Data Agent: Core Challenges



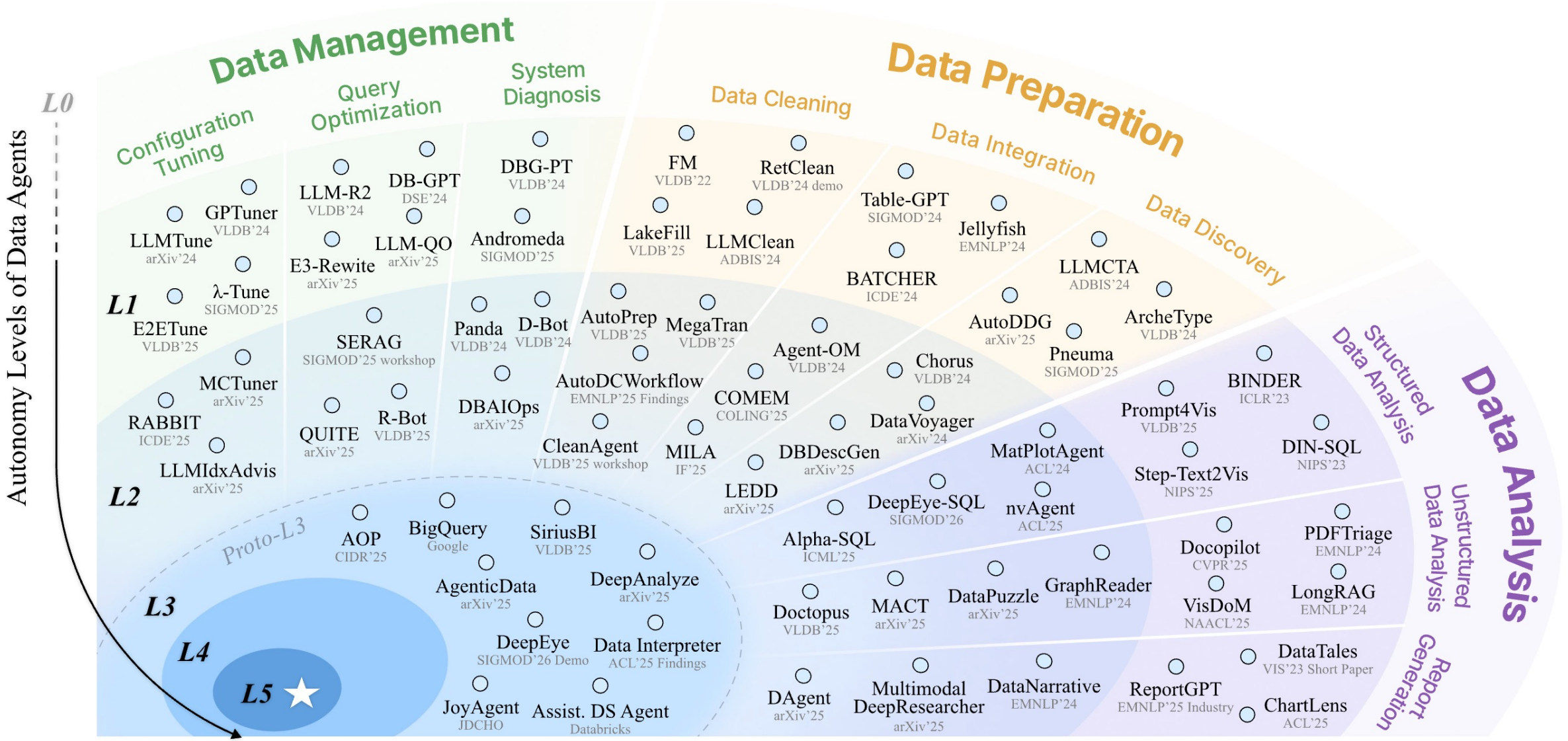
Core Data Agent Challenges Across Autonomy Levels

Challenge \ Level	L1 Assistance	L2 Partial Autonomy	L3 Conditional Autonomy	L4 High Autonomy	L5 Full Autonomy
	Prompt-driven Data Assistant	Human-guided Workflow Executor	Agentic Workflow Orchestrator	Proactive Data Agent	Self-evolving Autonomous Data Agent
C1. Data-aware Workflow Orchestration & Cost-aware Execution	Human-designed workflow; agent suggests.	Execute human-designed pipelines with local feedback.	Auto-compose and optimize workflows / DAGs.	Proactively discover tasks and plan long-horizon execution.	Invent new operators, tools, and workflow paradigms.
C2. Long-horizon Agentic State, Memory & Skill Reuse	Stateless; no persistent memory.	Short-term task memory and execution feedback.	Workflow state + provenance + reusable skills / SOPs.	Long-lived cross-task memory and shared skill reuse.	Self-evolving knowledge and skill base.
C3. Semantic Grounding over Heterogeneous & Multimodal Data	Prompt / few-shot / RAG-based grounding.	Environment-aware schema, entity, and evidence grounding.	Semantic catalog + metadata + multimodal evidence alignment.	Continuous semantic maintenance over evolving data lakes.	Create new semantic abstraction and representations.

Progression Assist & Suggest » Execute & Improve » Orchestrate & Optimize » Proactively Act » Invent & Evolve

Hierarchical Taxonomy for Data Agents

Structured Review Through this Lens



Check our full paper list: https://github.com/HKUSTDial/awesome-data-agents

Core Data Agent Challenges and Tutorial Roadmap

Challenge \ Level	L1 Assistance	L2 Partial Autonomy	L3 Conditional Autonomy	L4 High Autonomy	L5 Full Autonomy
	Prompt-driven Data Assistant	Human-guided Workflow Executor	Agentic Workflow Orchestrator	Proactive Data Agent	Self-evolving Autonomous Data Agent
C1. Data-aware Workflow Orchestration & Cost-aware Execution	Human-designed workflow; agent suggests.	Execute human-designed pipelines with local feedback.	Auto-compose and optimize workflows / DAGs.	Proactively discover tasks and plan long-horizon execution.	Invent new operators, tools, and workflow paradigms.
Memory & Skill Reuse	persistent memory.	memory and execution feedback.	reusable skills / SOPs.	shared skill reuse.	skill base.
C3. Semantic Grounding over Heterogeneous & Multimodal Data	Prompt / few-shot / RAG-based grounding.	Environment-aware schema, entity, and evidence grounding.	Semantic catalog + metadata + multimodal evidence alignment.	Continuous semantic maintenance over evolving data lakes.	Create new semantic abstraction and representations.

Data-aware Workflow Orchestration and Cost-aware Execution (40 min)

Progression: Assist & Suggest » Execute & Improve » Orchestrate & Optimize » Proactively Act » Invent & Evolve

Outline

- Part I: Data Agents: Motivation, Definition, Autonomy Levels, and Core Challenges
- **Part II: Towards Autonomous Data Agents: Key Challenges and Current Practices**
  - **Data-aware Workflow Orchestration & Cost-aware Execution**
  - Long-horizon Agentic State, Memory & Skill Reuse
  - Semantic Grounding over Heterogeneous & Multimodal Data
- Part III: Research Opportunities and Open Challenges

Core Data Agent Challenges and Tutorial Roadmap

Challenge \ Level	L1 Assistance	L2 Partial Autonomy	L3 Conditional Autonomy	L4 High Autonomy	L5 Full Autonomy
	Prompt-driven Data Assistant	Human-guided Workflow Executor	Agentic Workflow Orchestrator	Proactive Data Agent	Self-evolving Autonomous Data Agent
C1. Data-aware Workflow Orchestration & Cost-aware Execution	Human-designed workflow; agent suggests.	Execute human-designed pipelines with local feedback.	Auto-compose and optimize workflows / DAGs.	Proactively discover tasks and plan long-horizon execution.	Invent new operators, tools, and workflow paradigms.
Memory & Skill Reuse	persistent memory.	memory and execution feedback.	reusable skills / SOPs.	shared skill reuse.	skill base.
C3. Semantic Grounding over Heterogeneous & Multimodal Data	Prompt / few-shot / RAG-based grounding.	Environment-aware schema, entity, and evidence grounding.	Semantic catalog + metadata + multimodal evidence alignment.	Continuous semantic maintenance over evolving data lakes.	Create new semantic abstraction and representations.

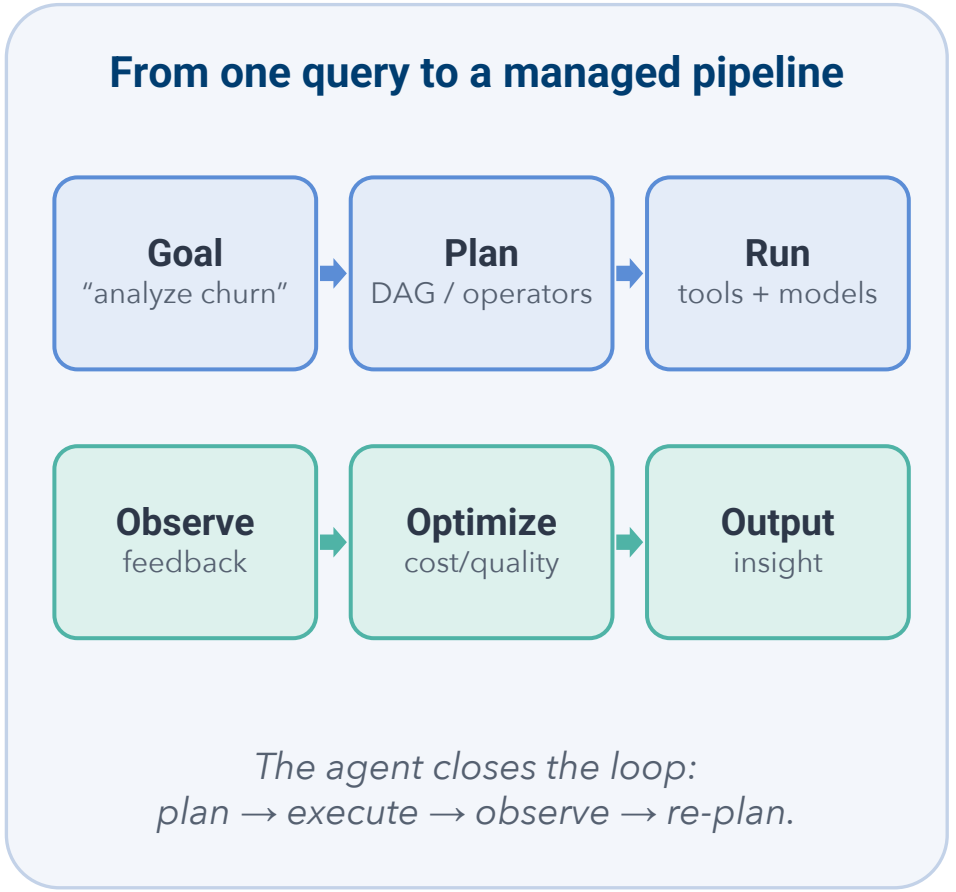
Data-aware Workflow Orchestration and Cost-aware Execution (45 min)

Progression: Assist & Suggest » Execute & Improve » Orchestrate & Optimize » Proactively Act » Invent & Evolve

Data-aware Workflow Orchestration & Cost-aware Execution

What is it?

- **Orchestration** decides what to do and in what order – decomposing a high-level data goal into an executable pipeline / DAG of operators, tools, and agents.
- **Cost-aware Execution** decides how to run it efficiently – choosing plans, models, and tools under latency, money, and accuracy budgets, with feedback-driven refinement.



Orchestration = the brain of the data agent;
Cost-aware execution = its discipline.

Two Tightly-Coupled Sub-Problems

① Workflow Orchestration

- Task understanding & decomposition
- Operator / tool / agent selection
- Pipeline (DAG) construction & ordering
- Dynamic re-planning on feedback

Q: WHAT to do, in WHAT order?

② Cost-aware Execution

- Plan / model / tool routing by cost
- Sampling, caching & reuse of results
- Execution-time optimization (selectivity)
- Budget & accuracy trade-offs

Q: HOW to run it cheaply & reliably?

Good plans waste resources without cost discipline;
Cheap execution is useless on the wrong plan.

Why Orchestration & Execution Matter

Data lakes are too big to ingest

Agents must sample, probe schemas, and refine queries on demand – a fixed prompt can't hold the whole lake.

Tasks span the full lifecycle

Real goals chain management → preparation → analysis; no single tool suffices, so steps must be orchestrated.

Errors cascade downstream

A wrong join or filter early on corrupts every later insight – execution needs verification and rollback.

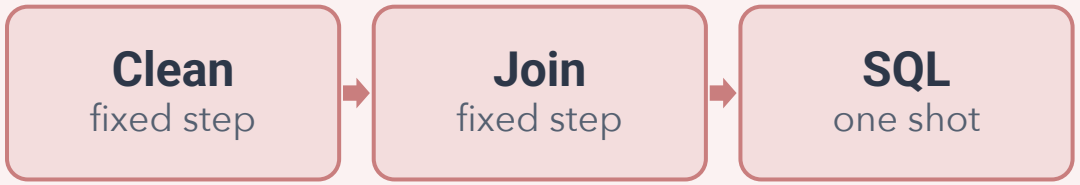
LLM calls are expensive

Naïve pipelines issue redundant, costly model and tool calls; budgets demand cost-aware routing and reuse.

Without orchestration & cost-aware execution, data agents stay stuck at brittle, one-shot assistance.

Motivated Example: Rigid Pipeline vs. Orchestrated Agent

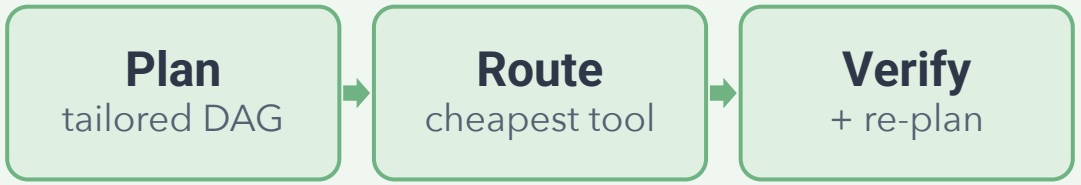
✗ Fixed, Human-wired Pipeline



- Same steps regardless of the question
- No re-planning when a step fails
- Redundant full-table scans & LLM calls
- Errors propagate silently to the report

✗ **Brittle, costly, hard to trust**

✓ Data-aware Orchestrated Agent



- Builds a pipeline tailored to the query
- Re-plans & repairs on execution feedback
- Caches, samples & routes by cost budget
- Validates each step before it propagates

✓ **Adaptive, efficient, auditable**

Evolution Across L1–L5

Level	Orchestration	Execution	Data awareness	Cost awareness
L1 Assistance	Human-designed one-shot workflow; agent suggests a step.	Human executes and verifies outside the agent.	Prompt / few-shot context only.	Implicit; user controls tokens and tools.
L2 Partial autonomy	Predefined pipeline for a specific task.	Agent runs tools with local feedback and retries.	Schema / execution feedback within the task.	Local heuristics: retry, prune, route, stop.
L3 Conditional autonomy	Agent composes and optimizes a tailored workflow or DAG.	Agent manages multi-step execution under supervision.	Cross-source task context, lineage, evidence.	Global budget over workflow: latency, tokens, compute, risk.
L4 High autonomy	Agent proactively discovers tasks and plans long-horizon workflows.	Continuous execution without routine supervision.	Persistent understanding of evolving data lake.	Strategic trade-offs across tasks and time.
L5 Full autonomy	Agent invents new operators and workflow paradigms.	Self-governing execution and validation.	Creates new data abstractions.	Learns new cost models and optimization strategies.

The key frontier today is L2 → L3:
from executing a human-defined procedure to designing the workflow itself.

Representative Works

Level	Orchestration Paradigm	Representative Works	Lifecycle Coverage
L1	Prompting (prompt–response, one-shot)	DIN-SQL (NeurIPS'23) DAIL-SQL (VLDB'24)	Data Analysis (NL2SQL)
L2	Predefined / human-designed workflows	D-Bot (VLDB'24) CleanAgent (VLDB'25) MAC-SQL (COLING'25)	Management · Preparation · Analysis
L3	Autonomous orchestration (Proto-L3)	Data Interpreter (ACL'25) AgenticData (2025) DeepAnalyze (ICML'26)	Toward the full lifecycle

We trace orchestration up the autonomy ladder: from prompting, to predefined workflows, to autonomous pipeline composition — then look ahead to L4/L5.

DIN-SQL [NeurIPS'23]: Decomposed In-Context Learning of Text-to-SQL with Self-Correction

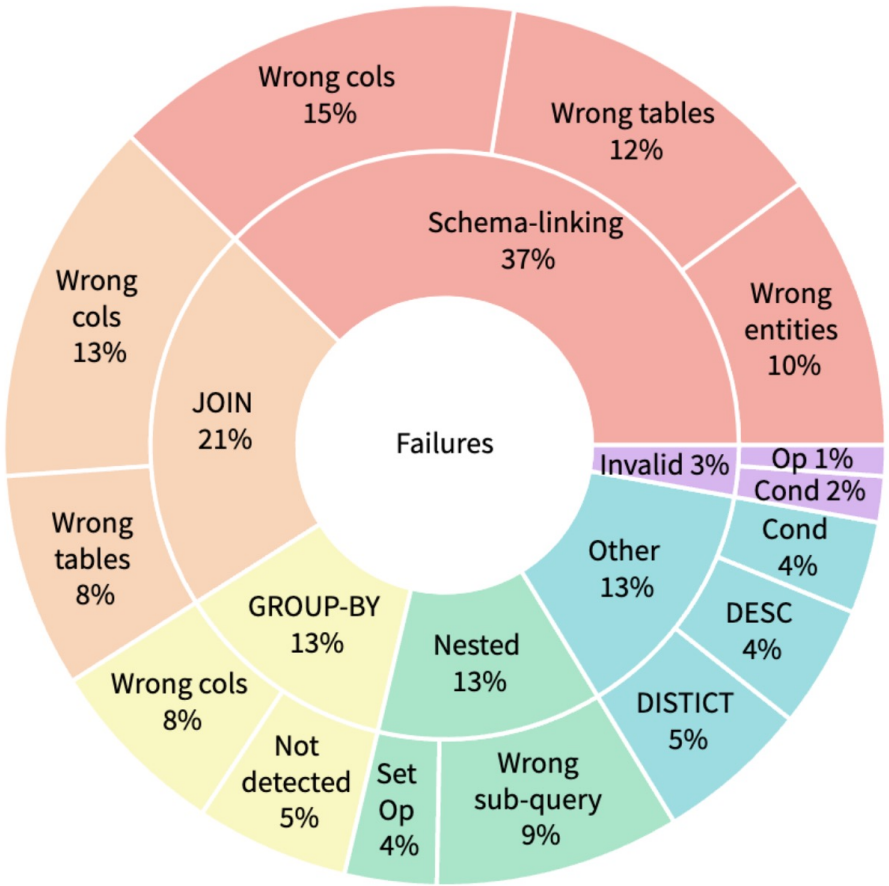


Figure 1: Statistics of simple few-shot failures using CodeX Davinci (Op refers to operators, Cond refers to conditions, and cols refers to columns)

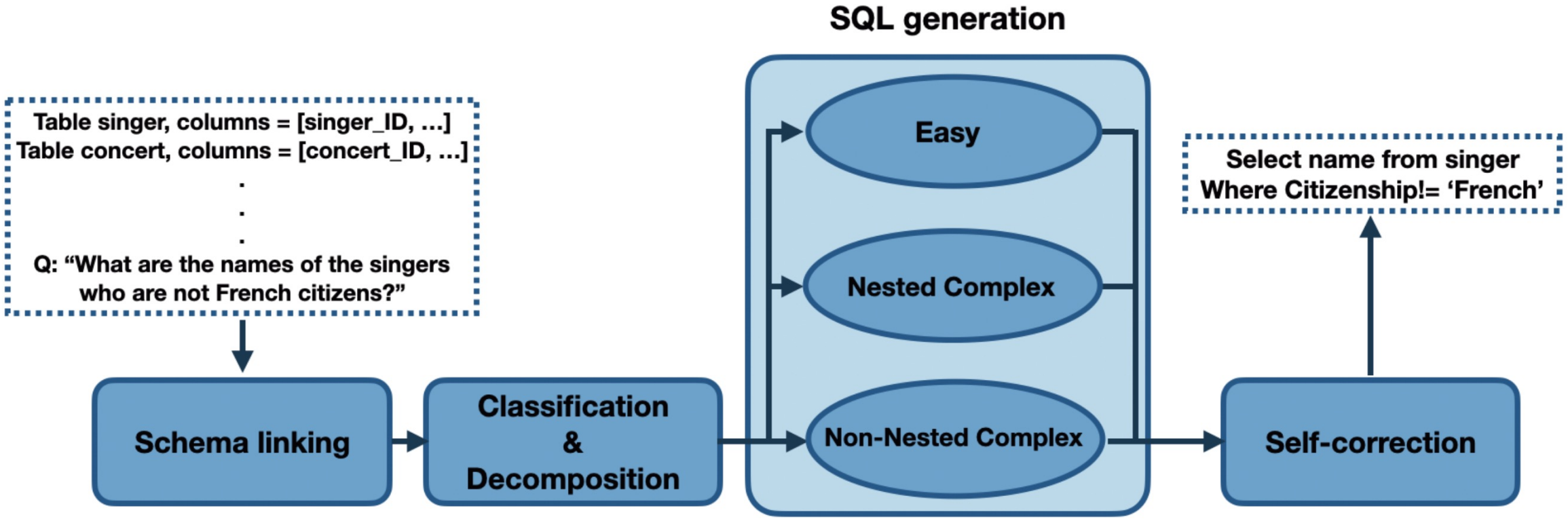
The one-shot bottleneck

A single zero/few-shot prompt must map a hard natural-language question – with joins, nesting, and aggregation – to correct SQL in one leap. Complex queries break this.

Why it matters

- LLMs struggle to do schema linking, structure planning, and clause writing all at once.
- Implicit reasoning stays hidden, so mistakes are hard to localize or correct.
- No environment access: the model cannot run the SQL or learn from execution.
- **Goal: make the reasoning explicit – purely through prompt design.**

DIN-SQL: Decomposed In-Context Learning of Text-to-SQL with Self-Correction



Decomposed in-context learning — a fixed, human-authored prompt chain

- Each sub-task has its own hand-crafted prompt; outputs are progressively combined into final SQL.
- A self-correction prompt reviews and repairs the draft – still without executing it.
- **"Orchestration" = a static chain the authors designed; every query takes the same fixed steps.**

DIN-SQL: Decomposed In-Context Learning of Text-to-SQL with Self-Correction

Model	EX	EM
DIN-SQL + GPT-4 (Ours)	**85.3**	60
RESDSQL-3B + NatSQL (DB content used) [Li et al., 2023a]	79.9	72
DIN-SQL + CodeX davinci (Ours)	78.2	57
Graphix-3B+PICARD (DB content used) [Li et al., 2023b]	77.6	**74**
SHiP+PICARD (DB content used) [Zhao et al., 2022]	76.6	73.1
N-best Rerankers + PICARD (DB content used) [Zeng et al., 2022]	75.9	72.2
RASAT+PICARD (DB content used) [Qi et al., 2022]	75.5	70.9
T5-3B+PICARD (DB content used) [Scholak et al., 2021]	75.1	71.9
RATSQL+GAP+NatSQL (DB content used) [Gan et al., 2021]	73.3	68.7
RYANSQL v2 + BERT [Choi et al., 2021]	-	60.6
SmBoP + BART [Rubin and Berant, 2020]	-	60.5

Spider

SOTA at release

Gains on complex splits

- Decomposition demonstrably improves one-shot accuracy and lowers the barrier for non-experts.
- But it is pure prompting: stateless, no perception, no cost-aware runtime control.
- **Takeaway:** helpful structure, yet the workflow is human-designed and frozen – the hallmark of L1.

DAIL-SQL [VLDB'24]: Text-to-SQL Empowered by LLMs: A Benchmark Evaluation

```
Table continents, columns = [ContId, Continent]
Table countries, columns = [CountryId, CountryName,
↳ Continent]
Q: How many continents are there?
A: SELECT
```

Listing 1: Example of Basic Prompt

```
Given the following database schema:
continents: ContId, Continent
countries: CountryId, CountryName, Continent

Answer the following: How many continents are there?
SELECT
```

Listing 2: Example of Text Representation Prompt

```
### Complete sqlite SQL query only and with no
↳ explanation
### SQLite SQL tables, with their properties:
#
# continents(ContId, Continent)
# countries(CountryId, CountryName, Continent)
#
### How many continents are there?
SELECT
```

Listing 3: Example of OpenAI Demostration Prompt

```
/* Given the following database schema: */
CREATE TABLE continents(
    ContId int primary key,
    Continent text,
    foreign key(ContId) references countries(Continent)
);

CREATE TABLE countries(
    CountryId int primary key,
    CountryName text,
    Continent int,
    foreign key(Continent) references continents(ContId)
);

/* Answer the following: How many continents are there?
↳ */
SELECT
```

Listing 4: Example of Code Representation Prompt

```
Below is an instruction that describes a task, paired
↳ with an input that provides further context. Write a
↳ response that appropriately completes the request.

### Instruction:
Write a sql to answer the question "How many continents
↳ are there?"

### Input:
continents(ContId, Continent)
countries(CountryId, CountryName, Continent)

### Response:
SELECT
```

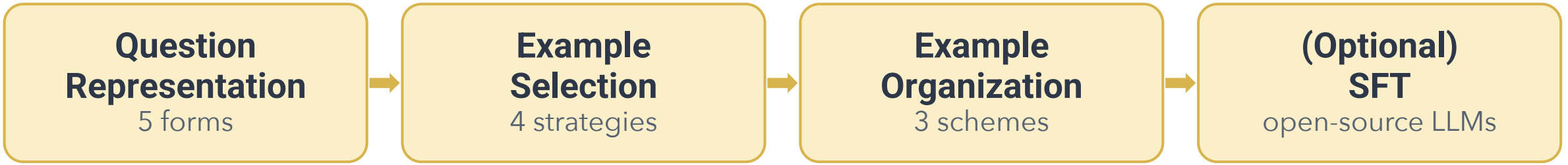
Listing 5: Example of Alpaca SFT Prompt

Prompt design was ad-hoc

Before DAIL-SQL, NL2SQL prompting was a grab-bag of tricks. It was unclear which question representation, examples, and organization actually matter for accuracy.

Text-to-SQL Empowered by Large Language Models: A Benchmark Evaluation

A systematic study of the prompt design space



- DAIL-SQL combines a structure-aware question representation with skeleton/similarity-based example selection.
- It also studies supervised fine-tuning, extending the analysis to open-source models.
- **"Orchestration" = choosing the best prompt template; the call itself is still single-shot.**

Text-to-SQL Empowered by Large Language Models: A Benchmark Evaluation

Few-shot	Selection	Question Similarity	Query Similarity	GPT-4		GPT-3.5-TURBO		TEXT-DAVINCI-003		Vicuna-33B	
				EM	EX	EM	EX	EM	EX	EM	EX
0-shot	-	-	-	22.1	72.3	34.6	74.4	31.7	71.7	6.9	43.7
1-shot	Random	0.23	0.47	41.7	77.4	45.9	73.9	38.2	70.6	14.4	47.9
	Question Similarity selection	0.39	0.65	53.3	78.8	51.9	74.3	44.1	72.3	16.5	48.5
	Masked Question Similarity selection	0.57	0.80	58.2	79.1	57.4	76.0	47.9	75.0	21.4	48.7
	DAIL selection	0.56	0.95	62.1	80.2	59.5	75.5	51.9	76.9	22.8	49.2
	Upper Limit	0.56	0.98	63.7	81.0	61.4	77.2	53.1	77.5	22.7	49.4
3-shot	Random	0.23	0.48	48.9	79.4	49.0	73.6	41.7	71.6	16.8	46.9
	Question Similarity selection	0.37	0.63	56.3	79.2	53.8	74.7	52.2	74.1	21.1	47.1
	Masked Question Similarity selection	0.54	0.78	66.1	81.5	61.1	77.3	59.7	77.0	27.7	52.3
	DAIL selection	0.53	0.94	69.1	81.7	63.9	77.8	64.4	79.5	30.7	53.6
	Upper Limit	0.53	0.98	71.5	83.4	66.2	79.2	66.7	81.1	31.2	54.4
5-shot	Random	0.23	0.48	51.6	79.5	52.9	75.7	49.0	72.1	-	-
	Question Similarity selection	0.36	0.61	58.2	79.9	55.9	75.1	54.8	73.2	-	-
	Masked Question Similarity selection	0.52	0.77	66.8	82.0	62.3	77.9	64.7	78.6	-	-
	DAIL selection	0.52	0.94	71.9	82.4	66.7	78.1	67.7	80.5	-	-
	Upper Limit	0.51	0.97	74.4	84.4	68.8	79.6	70.7	82.4	-	-

- Established strong, reproducible prompt-engineering baselines that later agentic systems build on.
- Token-budgeted prompts hint at cost-awareness, but there is no execution-time control or feedback.
- **Takeaway:** even the best L1 prompting remains a static, one-shot interaction.

L1 Recap: Useful Assistance, But No Workflow Ownership

✓ Progress

- Decomposition + prompt engineering sharply improve one-shot quality.
- Lower the comprehension barrier for non-expert users.
- Offload trivial, routine coding (e.g., boilerplate SQL).

✗ Limits

- Stateless prompt–response; no memory across turns.
- No environmental perception or interaction.
- Human still executes, integrates, and verifies.
- No cost-aware runtime control or feedback.

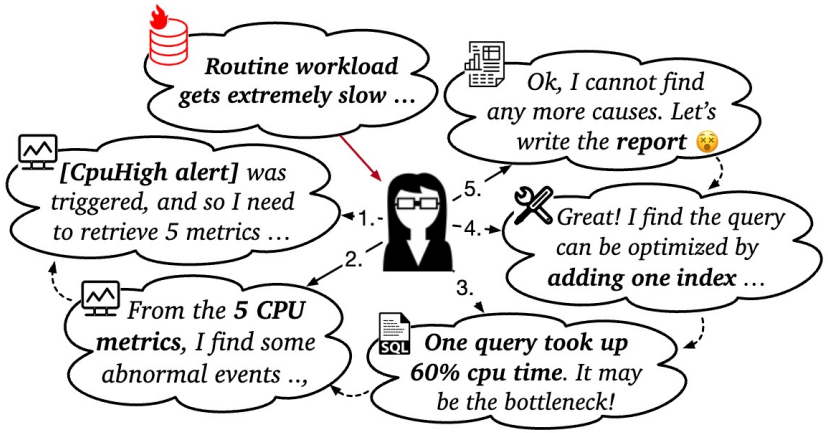
L1 is an assistant for workflow design, not a workflow orchestrator.

D-Bot [VLDB'24]: Database Diagnosis System using Large Language Models

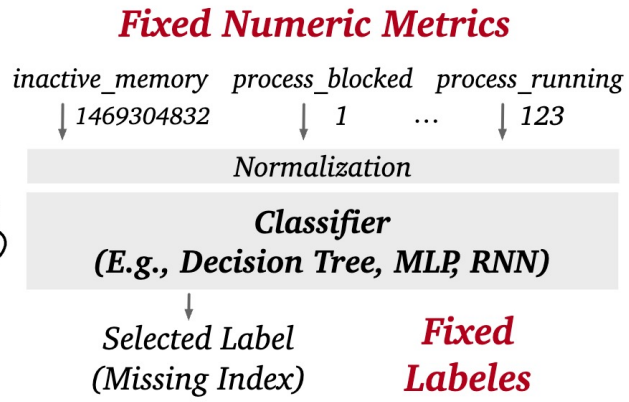
(a) Anomaly Events

Criteria \ Method	*Human*	*Classifier*	*D-Bot*
Supported Anomaly	*Most*	*Metric-Based*	*Most*
Expense	*High*	*Low*	*Low*
Efficiency	*Low*	*High*	*High*
Generalizability	*High*	*Low*	*High*

(b) Comparison of Diagnosis Methods



(c) Diagnosis by Human



(d) Classifier for Fixed Scenario

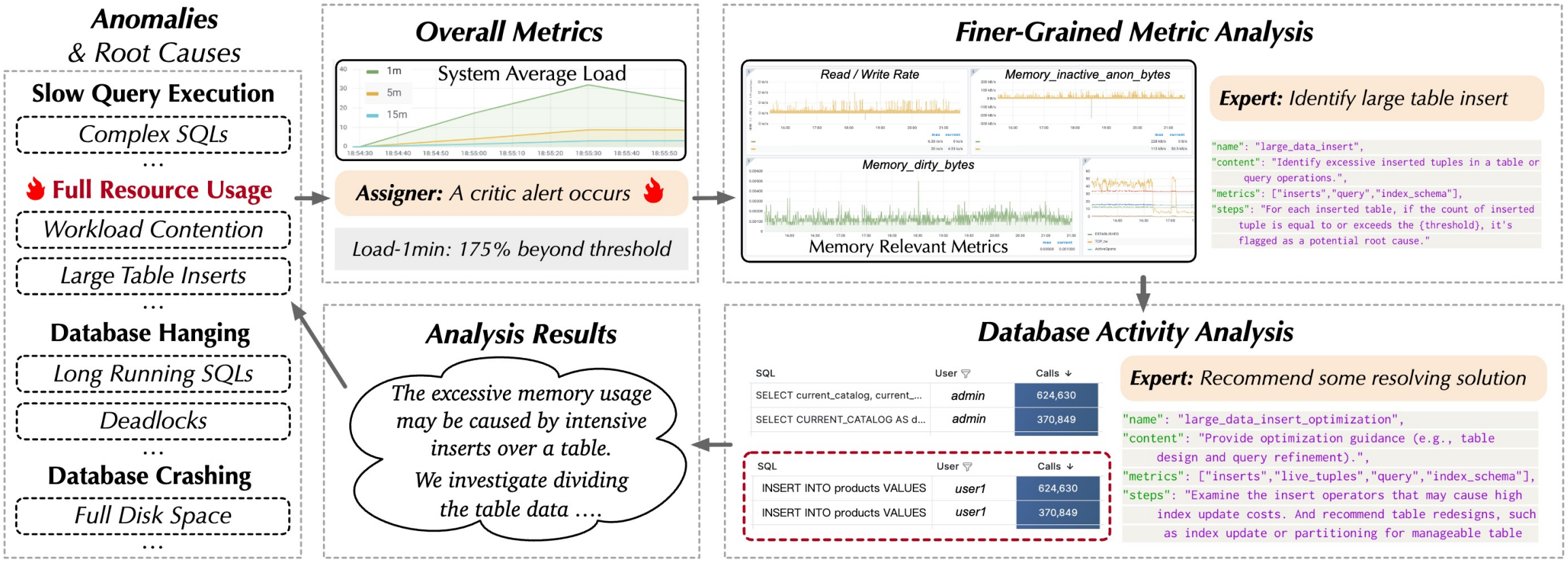
Diagnosis needs scattered expertise

Database anomaly diagnosis demands deep, fragmented expert knowledge. Manual root-cause analysis is slow, and a single LLM prompt only yields generic, unreliable advice.

Need: structured, knowledge-grounded, interactive diagnosis.

D-Bot: Database Diagnosis System using Large Language Models

Multi-agent diagnosis over a human-designed workflow



Agent roles & collaboration are predefined – the agent fills the template, it doesn't design it.

D-Bot: Database Diagnosis System using Large Language Models

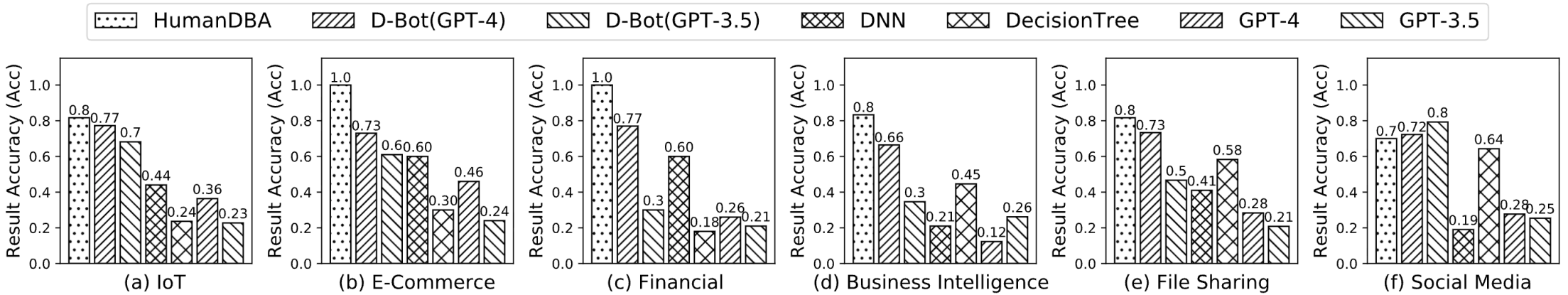


Figure 8: Performance Comparison (Result Accuracy). Note *Acc* is a general numerical metric, which cannot fully reflect the diagnosis capacity (e.g., whether the diagnosis process is reasonable)

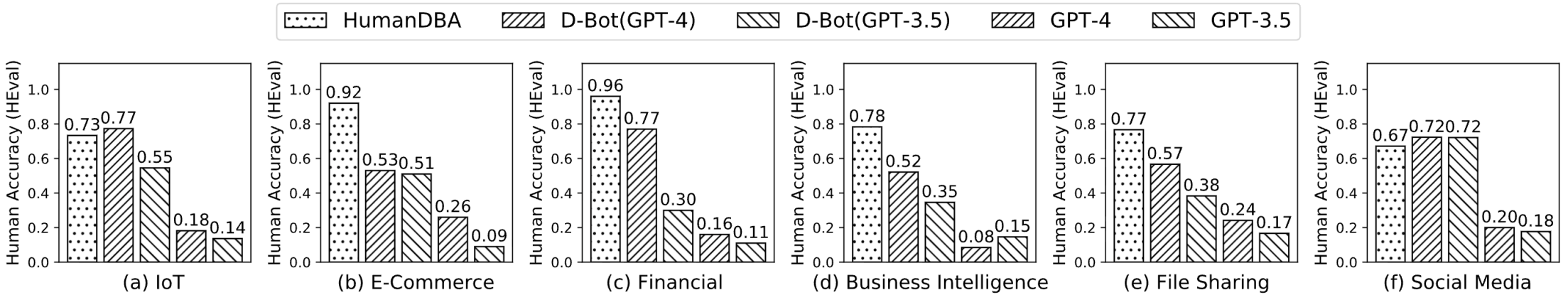


Figure 9: Performance Comparison (Human Evaluation). We do not include *DNN* and *DecisionTree* because they either have the black-box problem or fail to provide root cause analysis that is easy to understand by humans.

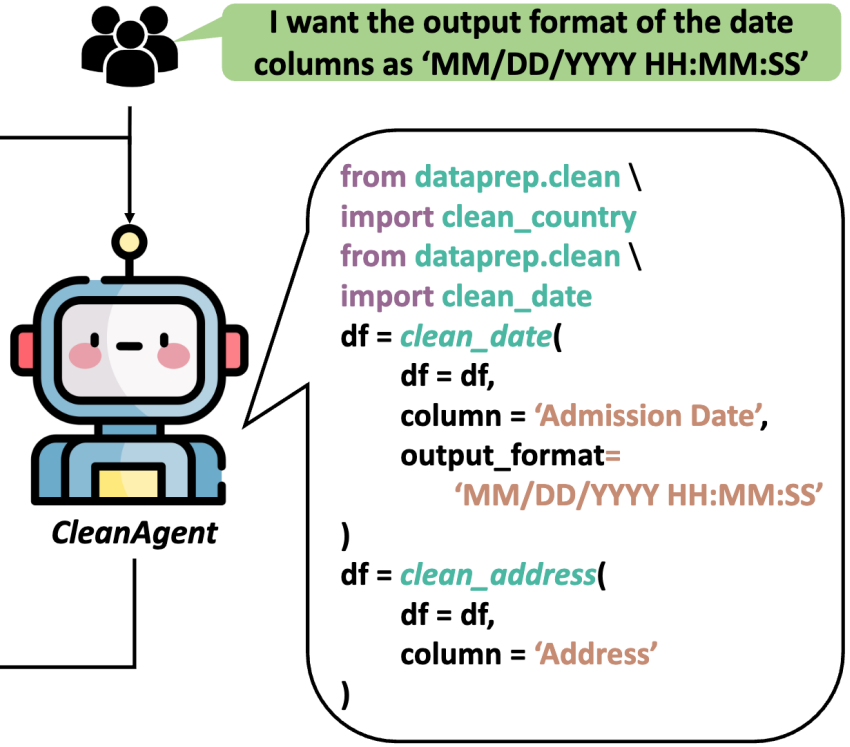
CleanAgent [VLDB'25]: Automating Data Standardization with LLM-based Agents

Input Table T

Name	Admission Date	Address
Abby	Fri Jan 1st 10:36:28 2021	1234 west main heights LA 57033
Scott	1996.07.10 AD at 15:08:56	1111 S Figueroa St, LA, 90015

Standardized Table T'

Name	Admission Date	Address
Abby	01/01/2021 10:36:28	1234 W. Main Hts., LA, 57033
Scott	01/15/2020 15:08:56	1111 S. Figueroa St., LA, 90015

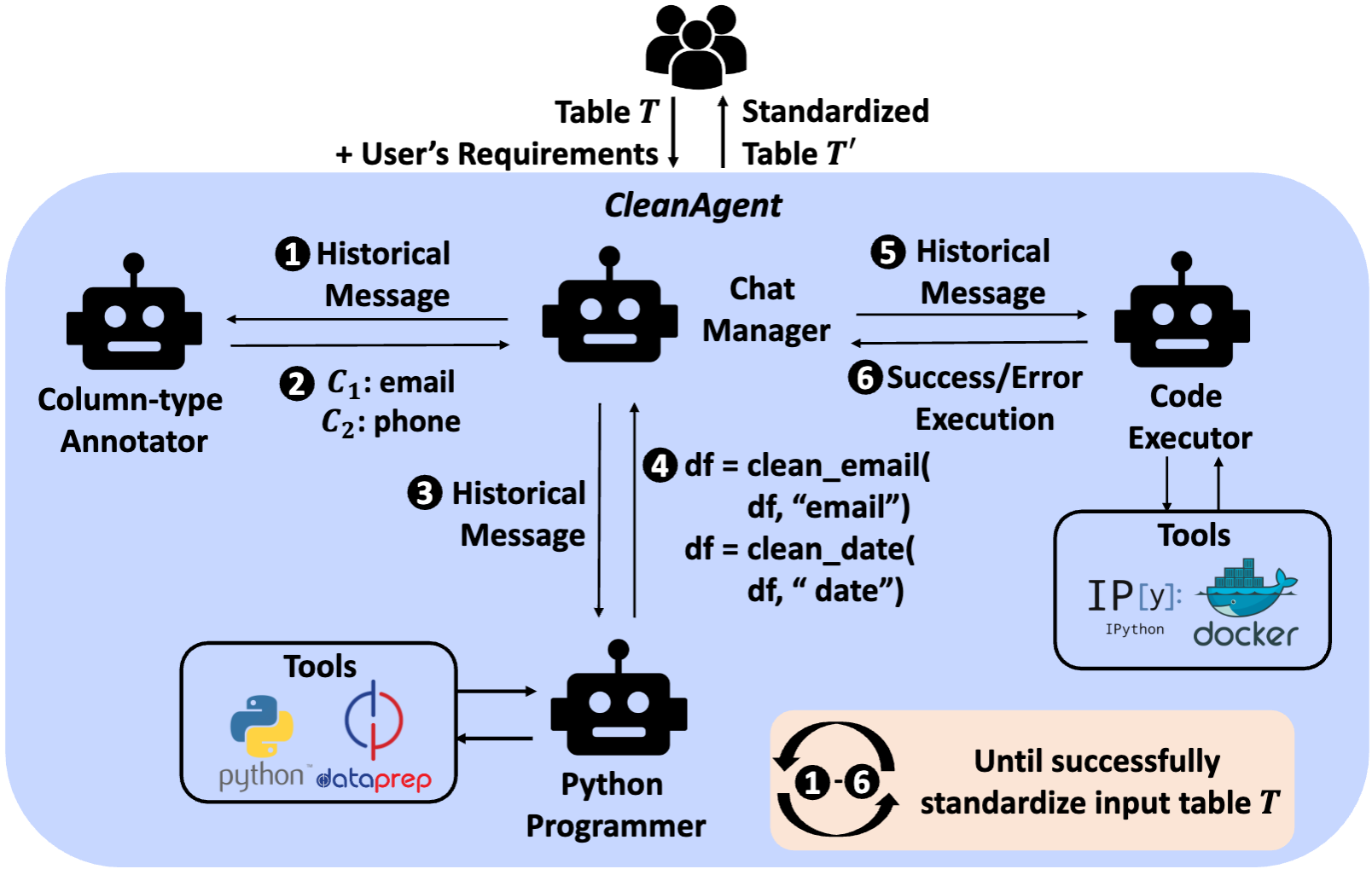


Problem

- Data cleaning requires detecting errors, choosing repair strategies, and verifying outcomes.
- One wrong repair can damage downstream models or analysis.
- Cleaning decisions should be traceable and revisable.

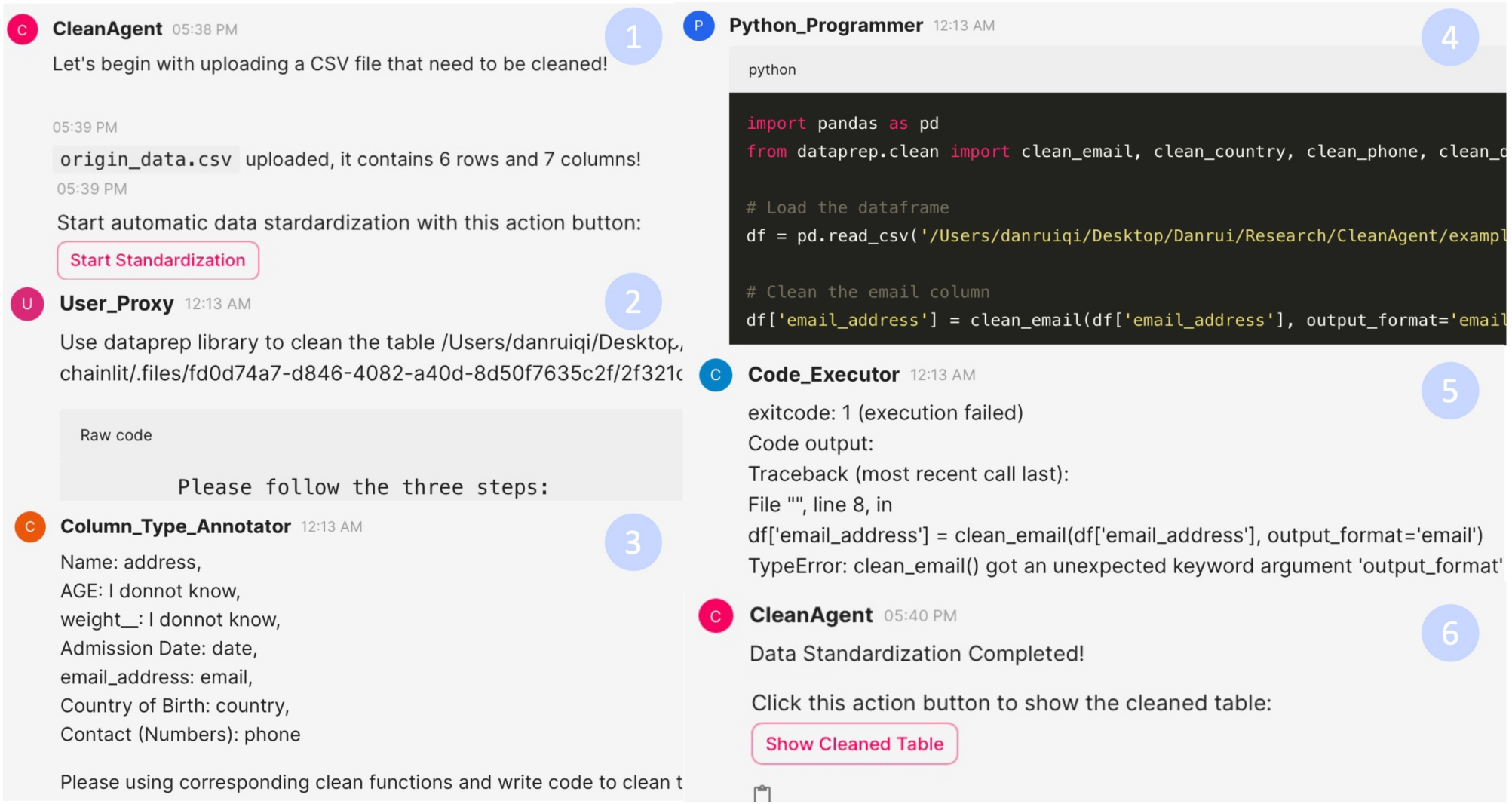
Cost-aware cleaning means early stopping when marginal repair value is low.

CleanAgent: Automating Data Standardization with LLM-based Agents



- Organize cleaning into specialized modules such as detection, repair, validation, and memory.
- Use execution feedback and historical context to refine actions.
- Iterate until quality checks or stopping conditions are satisfied.

CleanAgent: Automating Data Standardization with LLM-based Agents



More robust cleaning than one-shot prompting. Execution and validation reduce hallucinated repairs.

MAC-SQL [COLING'25]: A Multi-Agent Collaborative Framework for Text-to-SQL

User Question

```
List school names of charter schools with an SAT excellence
rate over the average.
```

Database schema

```
frpm: CDSCode, County Code, School Code, Charter School(Y/N)
satscores: cds, sname, AvgScrMath, NumTstTakr, NumGE1500, …
schools: CDSCode, NCESDist, County, City, Zip, …
```

Evidence

```
SAT_Excellence_Rate = CAST(NumGE1500 AS REAL) / NumTstTakr
```

Gold SQL

```
SELECT ST.sname FROM frpm FR JOIN satscores ST
ON FR.CDSCode = ST.cds WHERE FR.`Charter School (Y/N)` = 1
AND SAT_Excellence_Rate >
( SELECT AVG(SAT_Excellence_Rate) FROM frpm fr JOIN
  satscores st  ON fr.CDSCode = st.cds
  WHERE fr.`Charter School (Y/N)` = 1 )
```

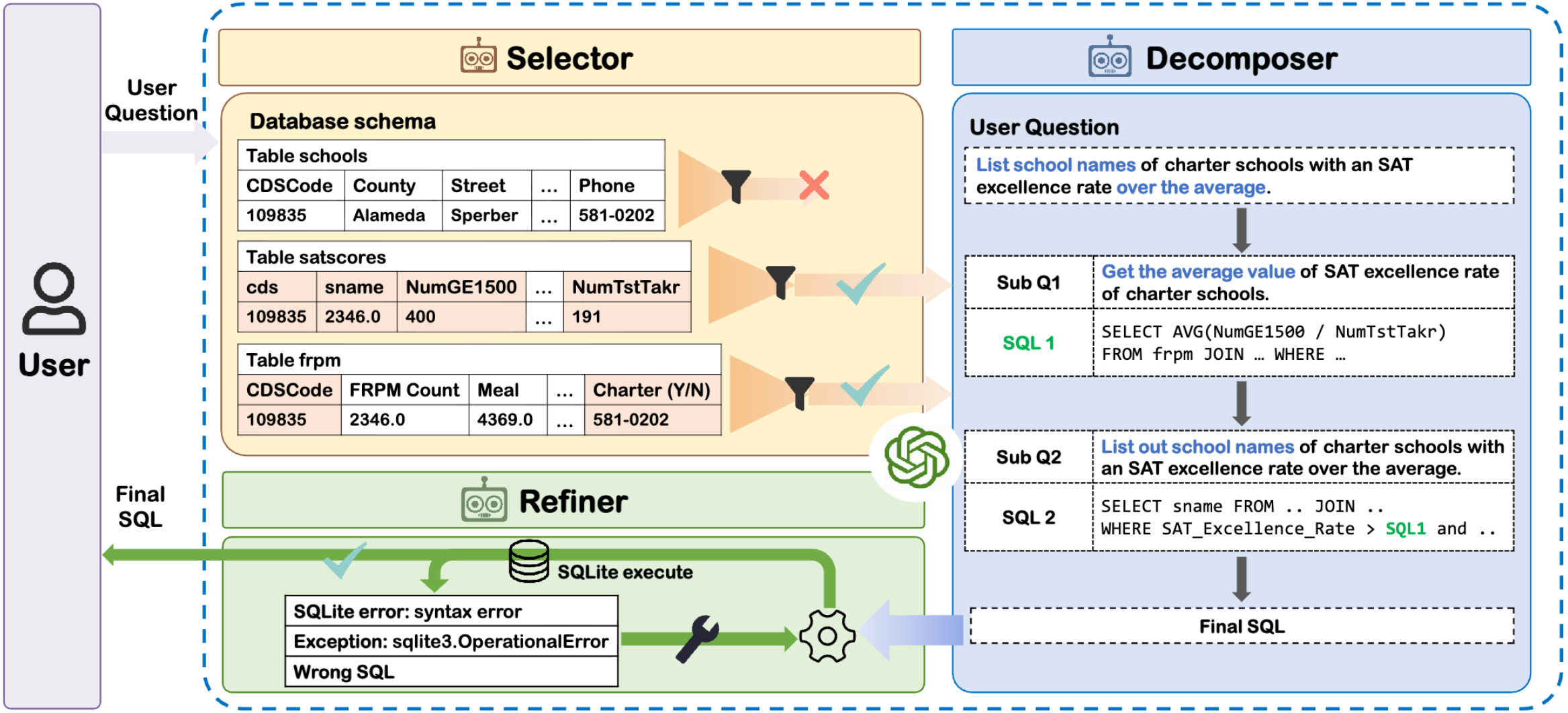
Big schemas & faulty SQL need teamwork

On large databases, a single pass struggles with schema overload and buggy queries. One-shot generation can't recover when the SQL simply doesn't run.

Why it matters

- Huge schemas overwhelm the prompt and dilute relevant columns.
- Generated SQL often fails on first execution and needs repair.
- **Need: collaboration to prune, decompose, and fix via execution feedback.**

MAC-SQL: A Multi-Agent Collaborative Framework for Text-to-SQL



The agent roles and message flow are predefined; the agent does not compose new workflows.

MAC-SQL: A Multi-Agent Collaborative Framework for Text-to-SQL

Method	Dev		Test	
	EX	VES	EX	VES
Palm-2	27.38	-	33.04	-
ChatGPT + CoT	36.64	42.30	40.08	56.56
Claude-2	42.70	-	49.02	-
GPT-4	46.35	49.77	54.89	60.77
DIN-SQL + GPT-4	50.72	58.79	55.90	59.44
DAIL-SQL + GPT-4	54.76	56.08	57.41	61.95
SQL-Llama	32.87	55.67	-	-
MAC-SQL + SQL-Llama	43.94	57.36	-	-
+ Oracle Schema	51.43	58.24	-	-
MAC-SQL+GPT-3.5-Turbo	50.56	61.25	-	-
+ Oracle Schema	65.78	60.62	-	-
MAC-SQL + GPT-4	**59.39**	**66.39**	**59.59**	**67.68**
+ Oracle Schema	70.28	62.63	-	-

Execution accuracy(EX) and Valid efficiency score (VES) on both dev and test set of BIRD dataset.

- Execution-feedback repair markedly improves robustness on complex, real-world databases.
- Perception + local feedback = L2; the collaboration template is still human-designed.
- **Takeaway:** feedback-driven repair within a fixed multi-agent workflow.

L2 Recap: Perception + Execution, But Bounded Workflows

✓ Progress

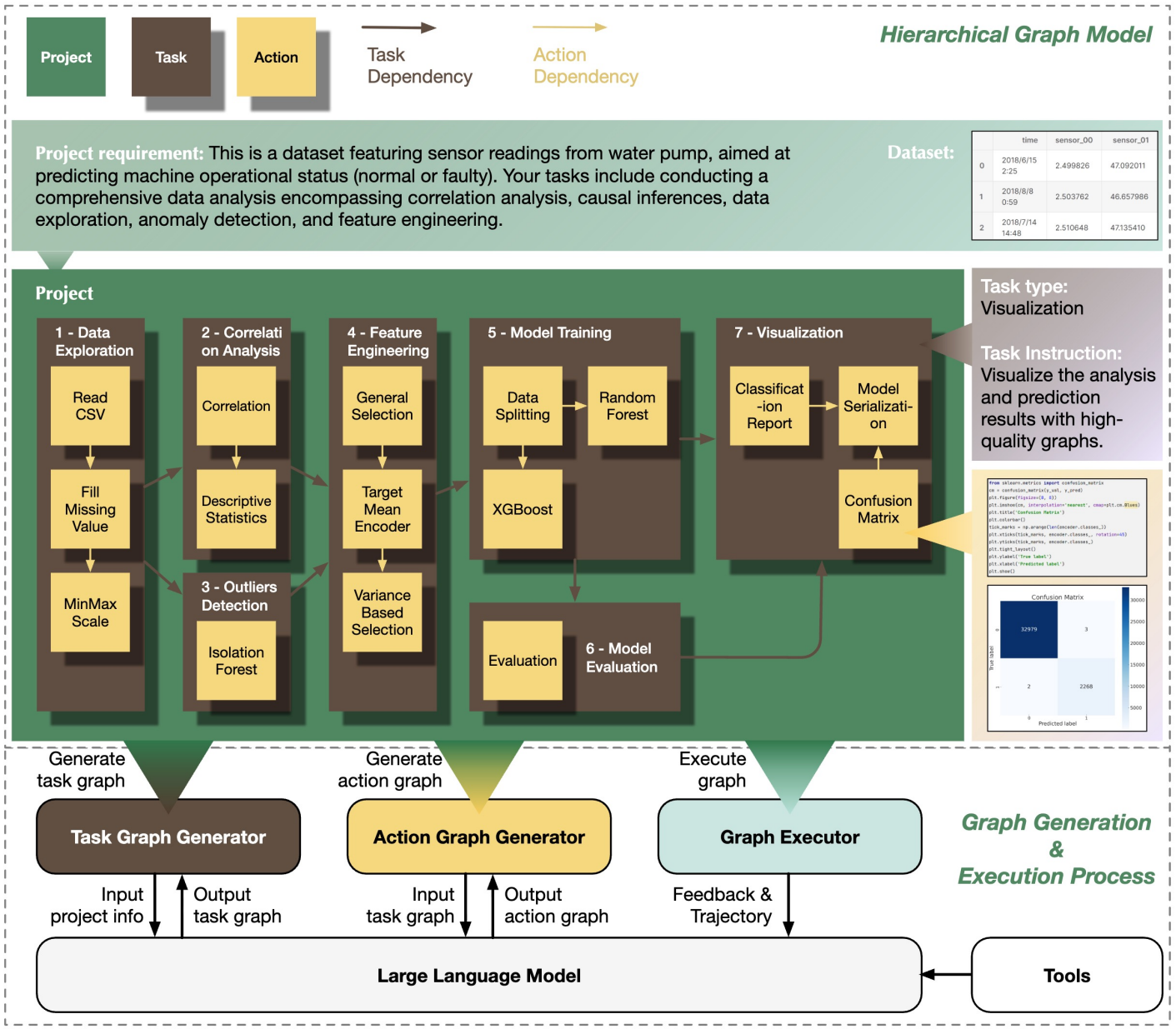
- Perceive the environment & invoke real tools.
- Refine outputs via local execution feedback.
- Cost-awareness as retries, schema pruning, caching.

✗ The glass ceiling

- Pipelines, roles & collaboration are human-designed.
- Task-specific rigidity (e.g., tuned only for NL2SQL).
- No orchestration across the full data lifecycle.
- Reactive executors, not self-directed orchestrators.

L2 systems are powerful executors, but not self-directed orchestrators.

Data Interpreter [ACL'25]: An LLM Agent For Data Science



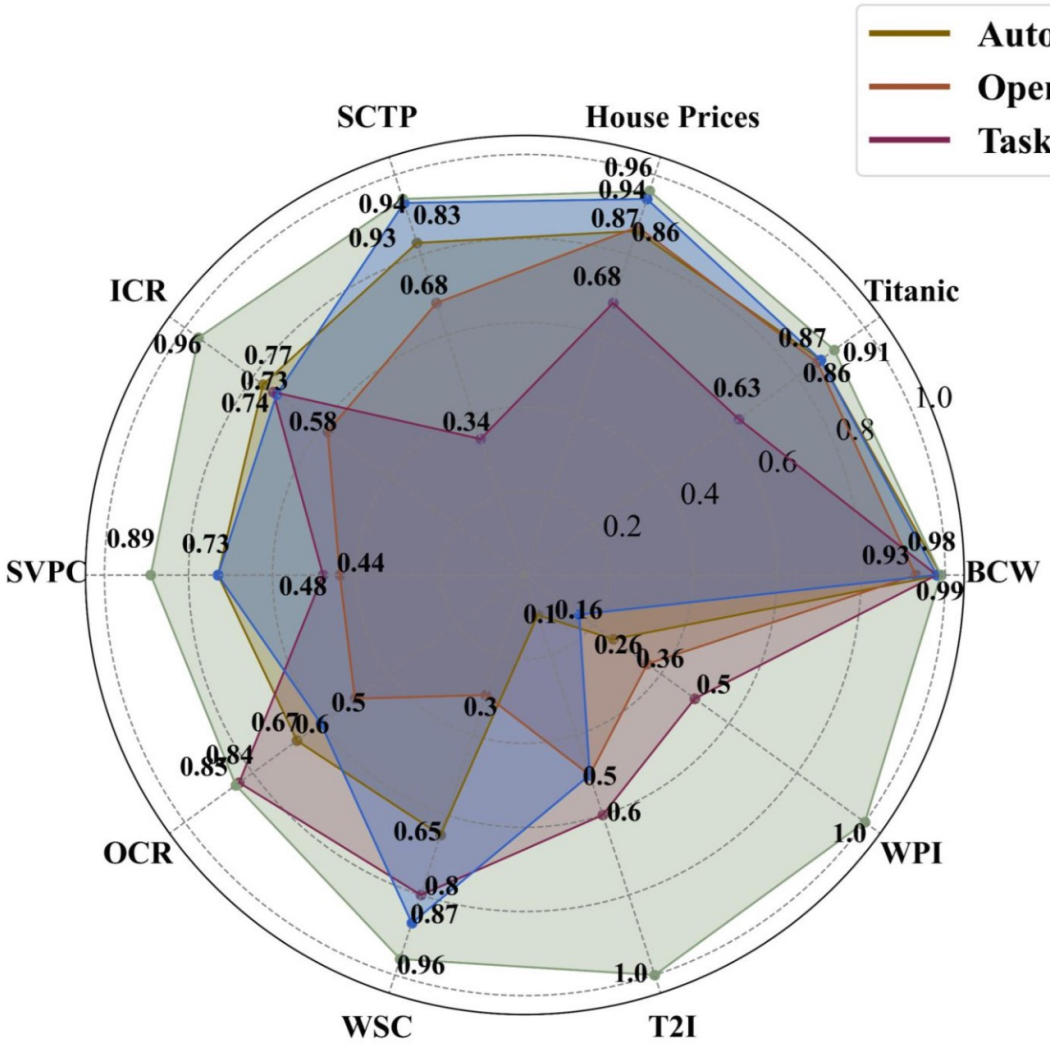
Fixed pipelines can't adapt

Data-science tasks are dynamic: sub-goals shift and steps fail at runtime. A statically wired pipeline cannot react, so the agent must build and revise the plan itself.

Orchestration as Hierarchical Graph Modeling

- The agent autonomously decomposes a high-level task into a Task Graph and an executable Action Graph.
- Iterative Graph Refinement dynamically modifies the graph based on graph-executor feedback.
- **No human-wired DAG: the agent composes and repairs the workflow itself – Proto-L3.**

Data Interpreter: An LLM Agent For Data Science



Machine learning tasks

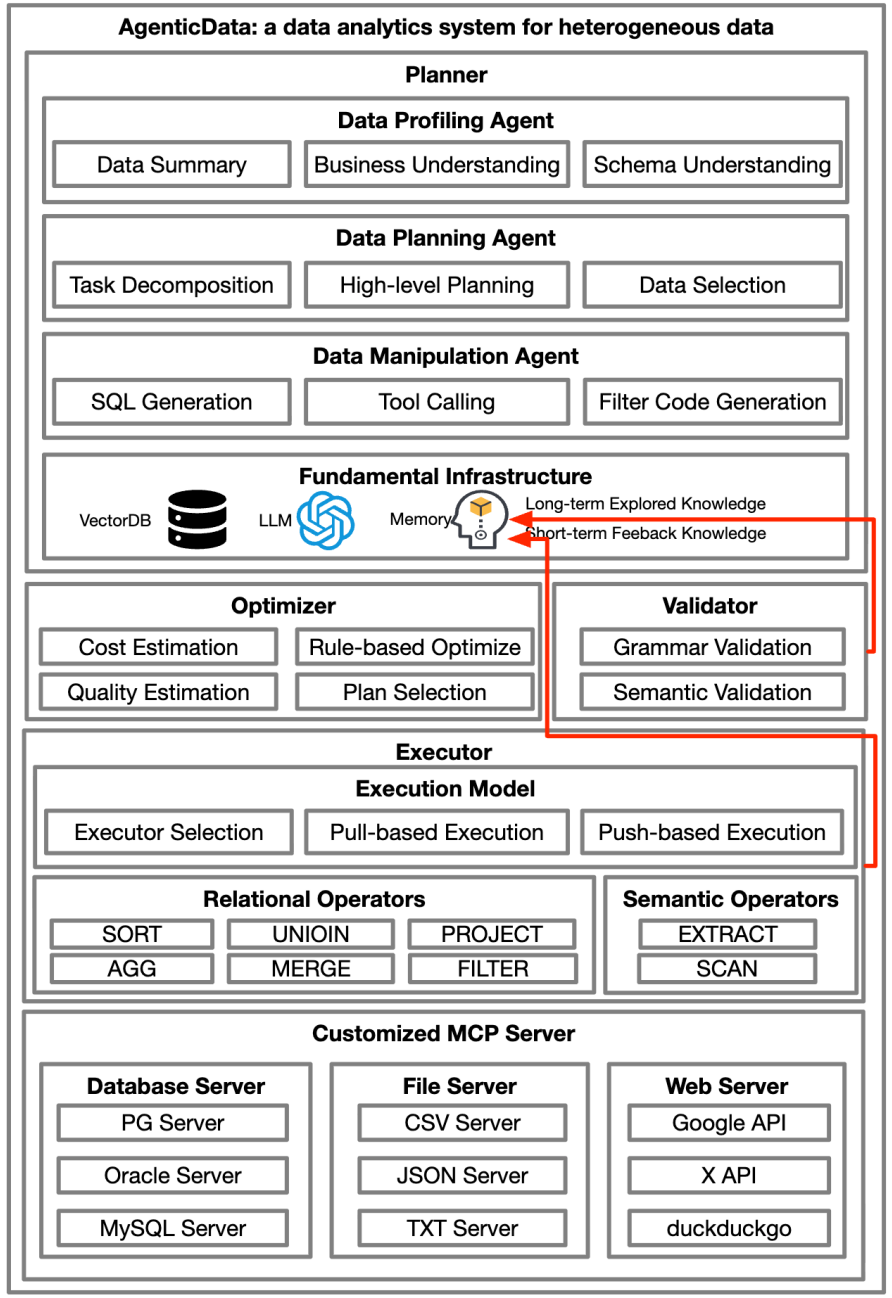
- SVPC: Santander Value Prediction Challenge
- ICR: Identifying Age-Related Conditions
- SCTP: Santander Customer Transaction Prediction
- BCW: Breast Cancer Wisconsin
- House Prices Prediction
- Titanic Survival Prediction

Real-world open-ended tasks

- OCR: Optical Character Recognition
- WSC: Web Search and Crawling
- T2I: Text-to-Image
- WPI: Web Page Imitation

- Autonomously plans, executes, and repairs data-science pipelines without a human-authored DAG.
- Limits: largely analysis-focused, leans on preprocessed data, uses a predefined action set.

AgenticData [ArXiv 2025]: An Agentic Data Analytics System for Heterogeneous Da



Beyond fixed operators & narrow scope

Real workloads span management, preparation, and analysis across heterogeneous sources – and often need operations no fixed operator library provides.

Multi-agent planning with on-the-fly operator synthesis

- Supports predefined AND non-predefined operators, the latter curated via LLM-based code generation.
- Customized MCP servers connect heterogeneous sources; a feedback-driven planner builds tree-structured pipelines.
- **Broadens task scope across management, preparation & analysis – the widest of the three.**

AgenticData: An Agentic Data Analytics System for Heterogeneous Data

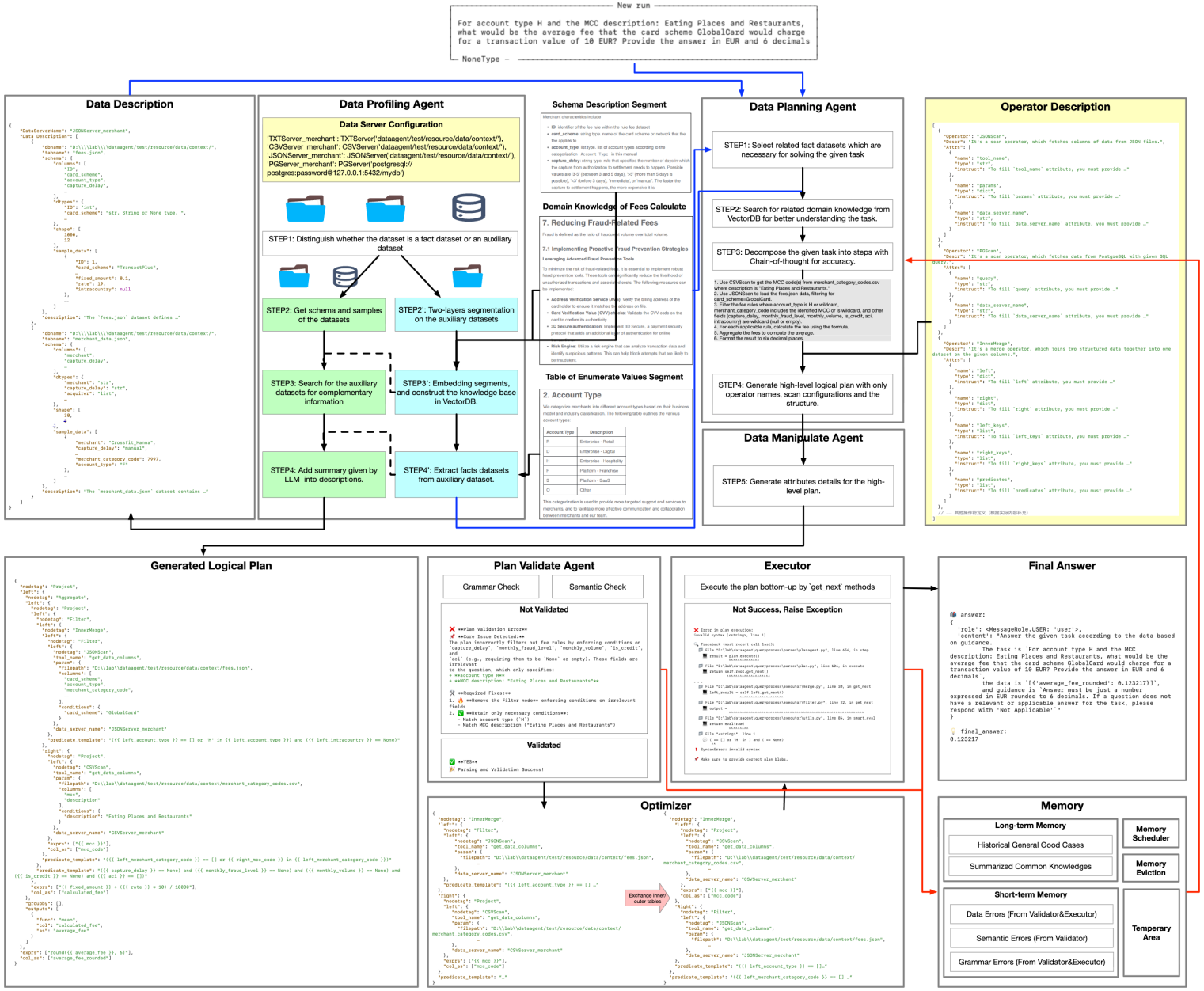


Table 1: Accuracy Comparison on `DABStep` (%).

Systems	LLM	Easy	Hard
`AgenticData`	Qwen3	94.44	50.79
Leaderboard `Amity`	Gemini2.5	80.56	41.01
Leaderboard `MultiStep`	GPT-o3	81.94	19.84

Table 2: Accuracy Comparison on `Spider-2.0-Lite` (%).

Systems	LLM	Accuracy
`AgenticData`	Qwen3	44.5
Leaderboard `ReFoRCE`	Qwen3	35.6

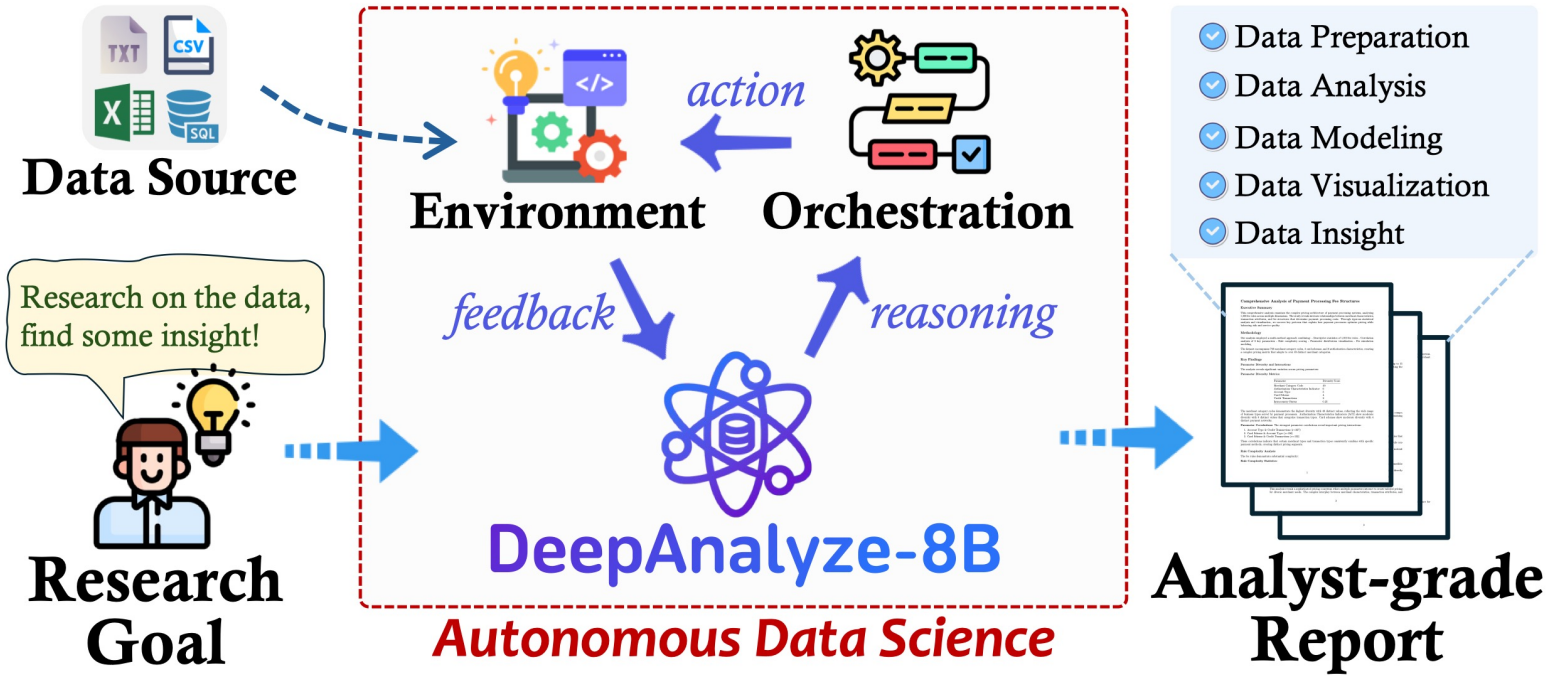
Table 3: Accuracy Comparison on `Wikipedia` (%).

Systems	LLM	Accuracy
`AgenticData`	Qwen3	95
`Palimpzest`	Qwen3	94

- Generates operators on the fly and spans the lifecycle across heterogeneous sources – the most ambitious Proto-L3.
- **Takeaway:** closest to self-directed orchestration; trustworthy operator synthesis is the missing piece.

DeepAnalyze [ICML'26]: Agentic Large Language Models for Autonomous Data Science

DeepAnalyze, an end-to-end agentic LLM that achieves autonomous data science, supporting **entire data science pipeline** and **open-ended data research**



DeepAnalyze: Agentic Large Language Models for Autonomous Data Science

DeepAnalyze emulates the analysis process of human data scientists by introducing **five actions**

Data Scientists → DeepAnalyze

1. Analyze tasks and plan
2. Access and explore the raw data in the environment
3. Understand intermediate execution results
4. Perform analysis, modeling, and visualization iteratively
5. Summarize and generate a data analysis report

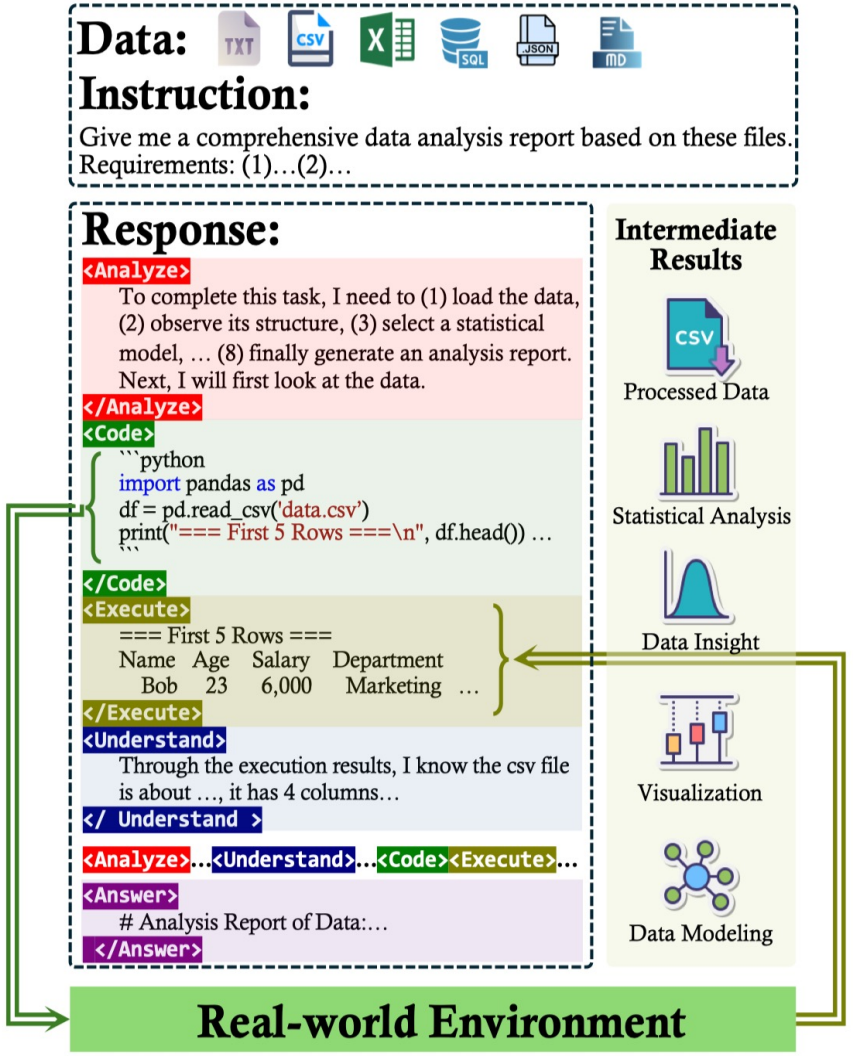
<Analyze>

<Code>&<Execute>

<Understand>

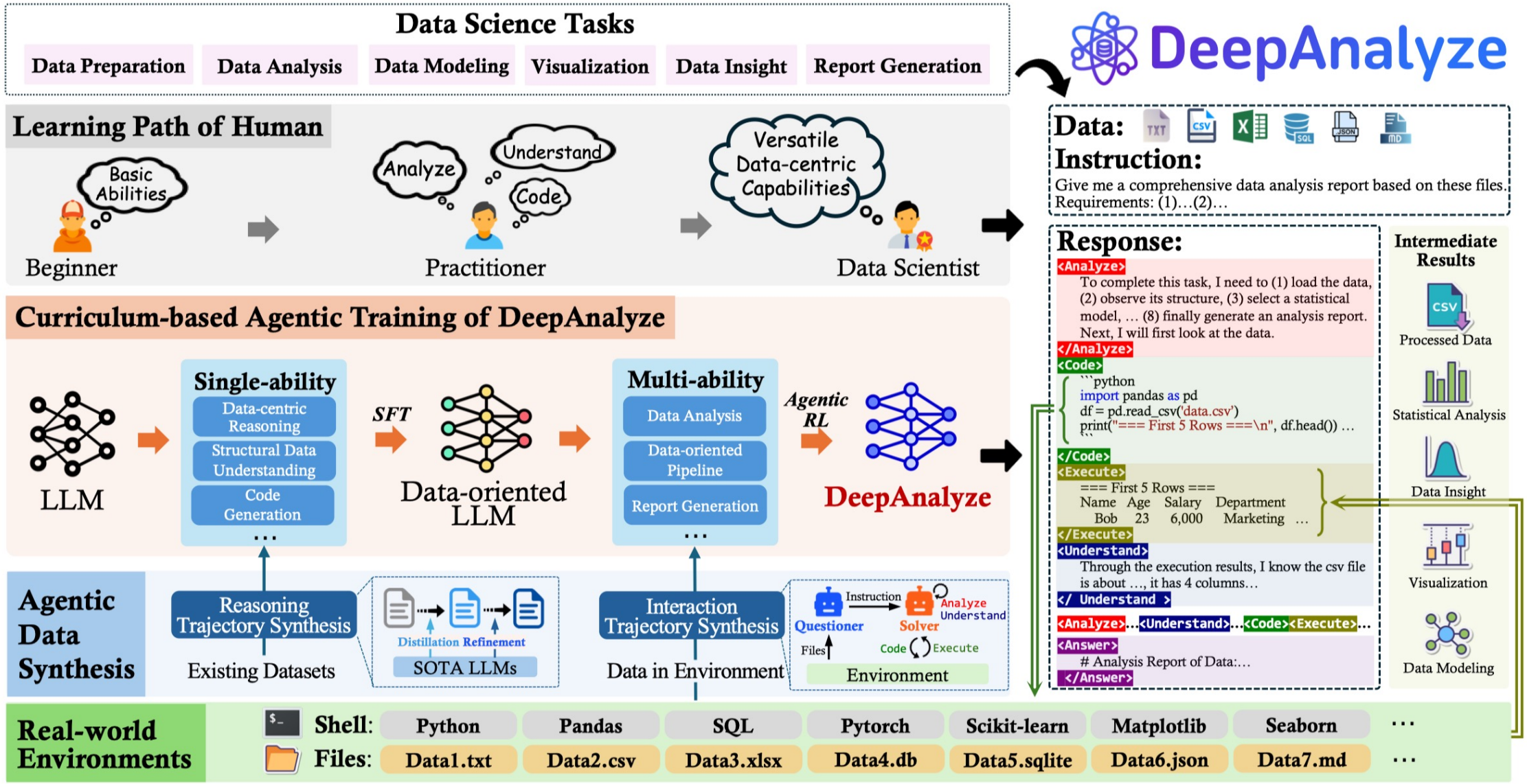
actions

<Answer>



DeepAnalyze: Agentic Large Language Models for Autonomous Data Science

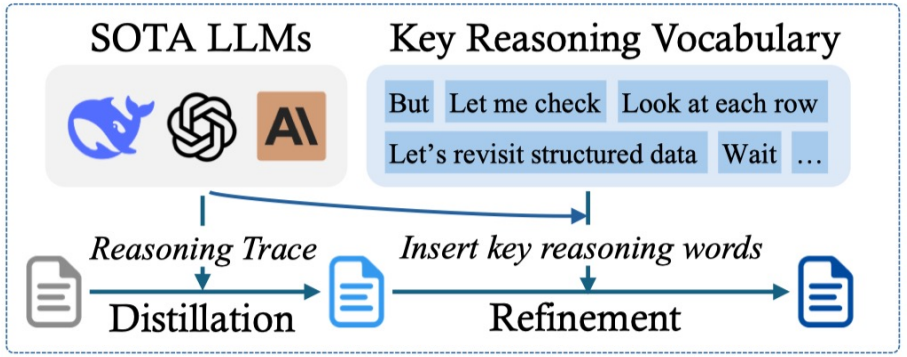
Curriculum-based Agentic Training: Single-ability Fine-tuning + Multi-ability Agentic Training



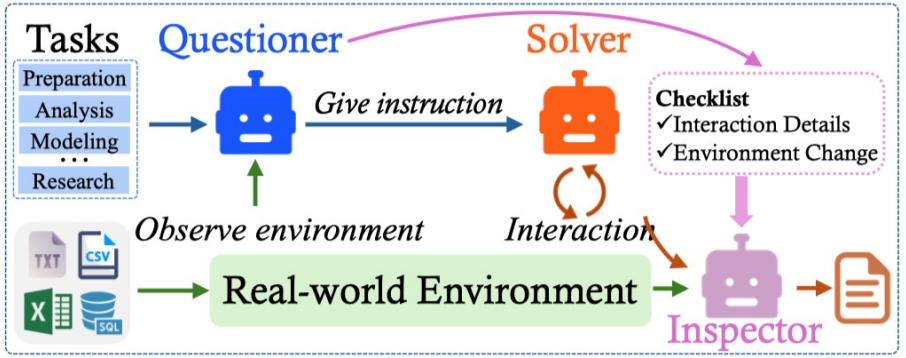
DeepAnalyze: Agentic Large Language Models for Autonomous Data Science

- The goal is to automatically construct high-quality trajectory data tailored to end-to-end data science tasks

- **Reasoning Trajectory Synthesis**
  - Enhancing public data science datasets by synthesizing reasoning trajectories distilled from state-of-the-art LLMs
- **Interaction Trajectory Synthesis**
  - No public datasets!
  - A Multi-agent data synthesis method
    - **Questioner**: formulating problems
    - **Solver**: interacting with the environment
    - **Inspector**: validating the result trajectories



(a) Reasoning Trajectory Synthesis



(b) Interaction Trajectory Synthesis

DeepAnalyze: Agentic Large Language Models for Autonomous Data Science

- Evaluating **end-to-end data science pipeline orchestration** on DataSciBench, where each task involves multiple key steps

Models	Coarse-grained Metrics		Fine-grained Metrics						Score
	Success Rate	Completion Rate	VLM	F1: Data Preparation	F2: Plot Validity	F3: Data Exploration	F4: Data Visualization	F5: Data Modeling	
Close-Source API-Based Agent									
o1-mini	29.77	45.26	2.87	44.63	19.27	36.01	30.94	23.81	38.78
GPT-4o-mini	50.63	57.78	3.05	60.30	48.02	57.84	59.24	53.54	54.18
GPT-4o	**66.31**	**68.44**	**3.91**	**75.93**	**56.14**	**69.33**	**71.35**	**57.67**	**64.51**
GPT-4-Turbo	51.93	58.87	3.09	62.30	41.62	57.75	60.25	50.75	54.65
Claude-3-5-Sonnet	47.48	58.11	2.14	49.07	36.94	55.84	52.87	46.04	52.29
GLM-4-Flash	30.32	34.04	1.33	36.53	29.42	32.57	27.64	14.44	30.74
Open-Source LLM-based Agent									
Llama-3.1-8B-Instruct	24.73	33.89	1.29	38.24	18.25	21.98	22.89	25.85	29.69
Gemma-2-9B-it	7.07	11.00	1.06	26.16	16.90	23.81	18.11	17.15	12.66
GLM-4-9B-Chat	25.72	30.38	1.69	31.51	23.15	28.07	27.19	19.14	27.57
Qwen2.5-7B-Instruct	43.83	50.74	1.43	51.18	36.41	47.25	45.24	34.77	45.99
Qwen2-7B-Instruct	22.84	25.58	1.16	30.93	20.78	28.73	25.87	7.52	23.52
Yi-1.5-9B-Chat-16K	38.20	42.35	0.73	38.14	36.36	35.64	37.08	27.79	38.22
CodeLlama-13B-Instruct	10.49	14.64	0.04	11.67	11.34	9.43	14.43	5.15	12.64
CodeLlama-7B-Instruct	2.88	3.97	0.00	3.53	2.37	2.57	1.74	1.59	3.31
StarCoder2-15B	2.07	2.61	0.07	2.57	1.81	1.59	3.43	1.19	2.33
Deepseek-Coder-6.7B-instruct	37.03	41.62	1.93	43.49	34.57	46.36	46.49	18.09	38.45
Qwen2.5-Coder-7B-Instruct	45.18	53.11	1.48	51.58	43.21	43.87	42.50	**35.23**	47.67
Agentic Model									
DeepAnalyze-8B	**59.91**	**66.24**	**2.86**	**71.68**	**67.86**	**58.62**	**69.09**	33.33	**61.11**

F1: Data Preparation
F2: Plot Validity
F3: Data Exploration
F4: Data Visualization
F5: Data Modeling

DeepAnalyze-8B achieves **the SOTA results** among open-source LLMs and **outperforms most close-source LLMs**, ranking 2nd only to GPT-4o

DeepAnalyze: Agentic Large Language Models for Autonomous Data Science

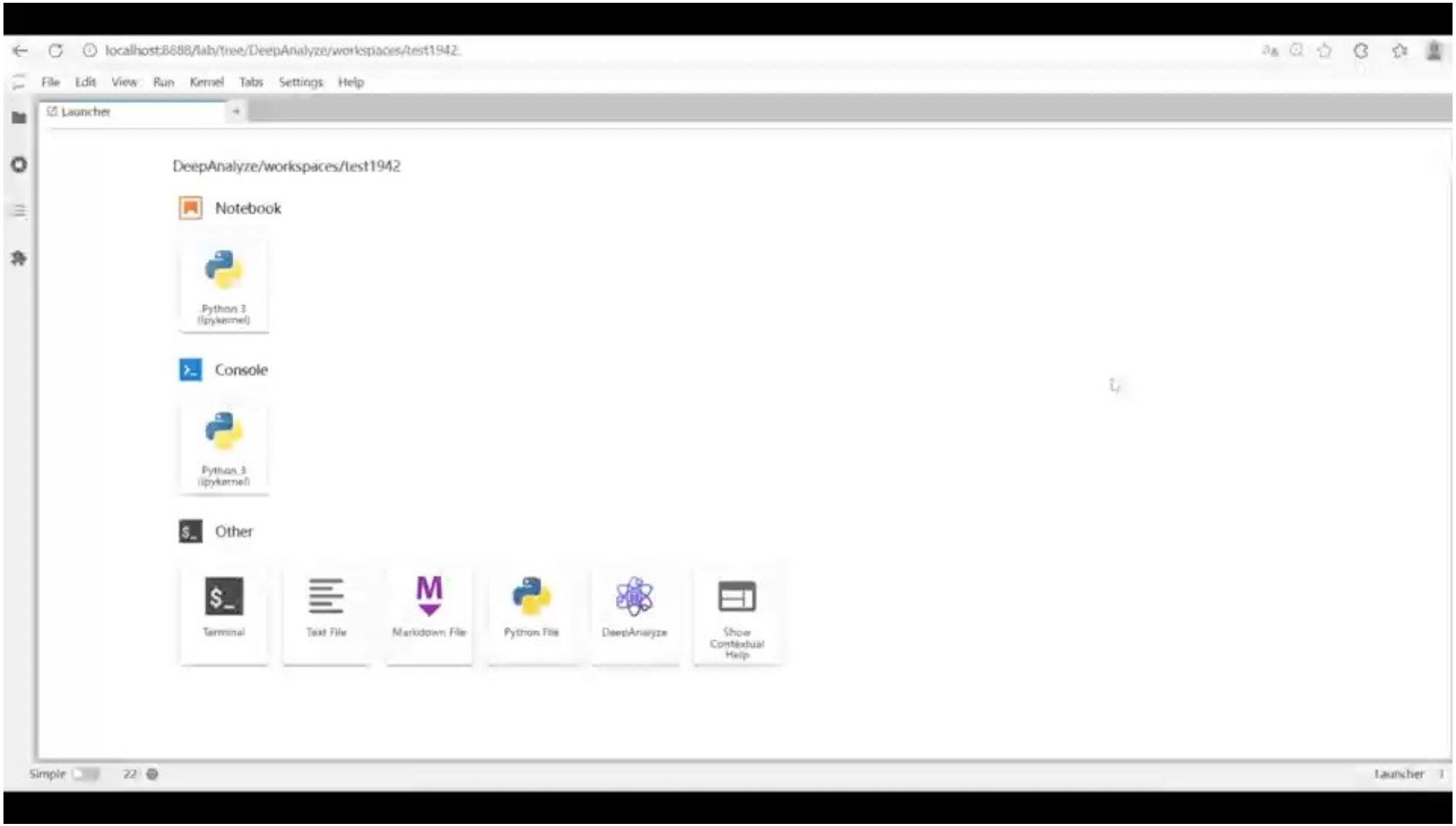
Model & Data

GitHub

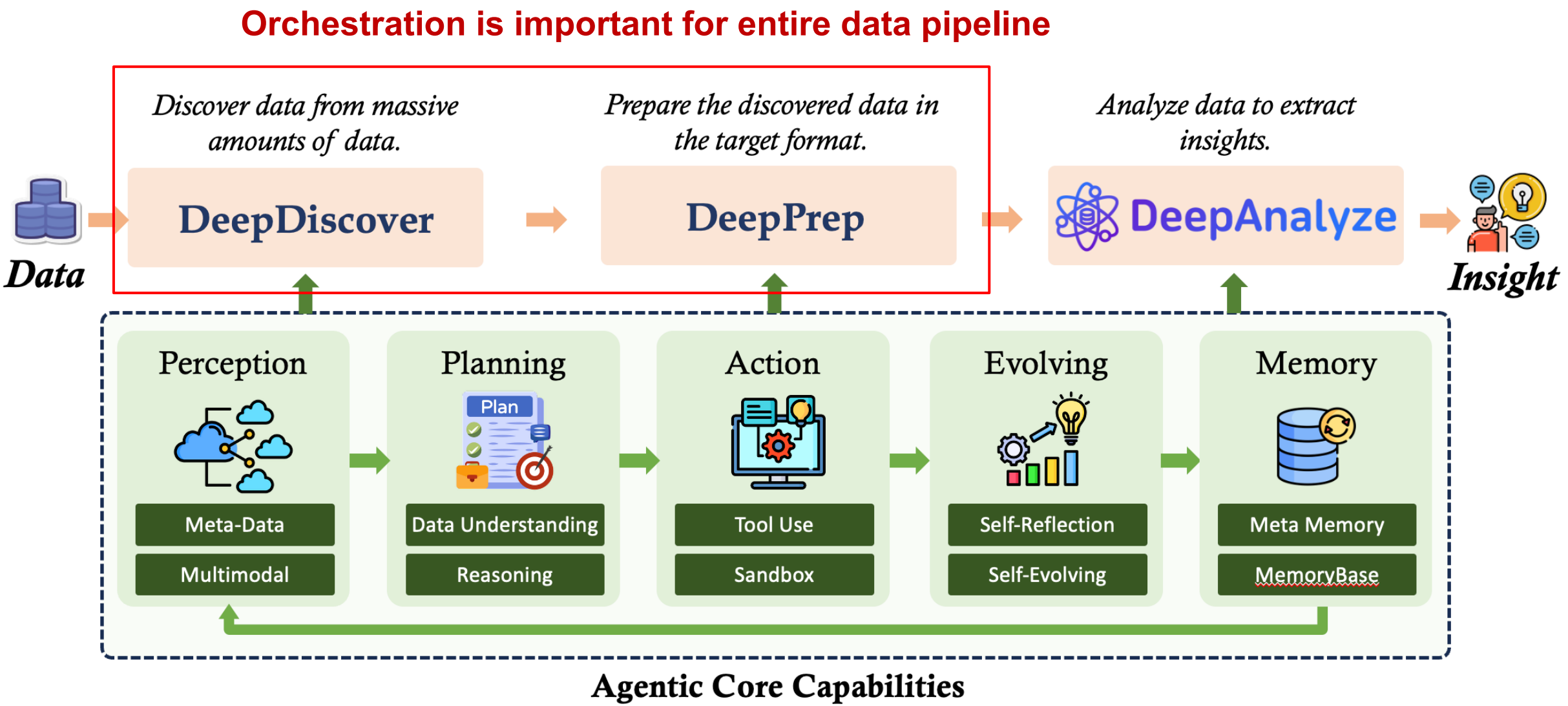
Code

4.1K+ GitHub Stars

600+ Forks

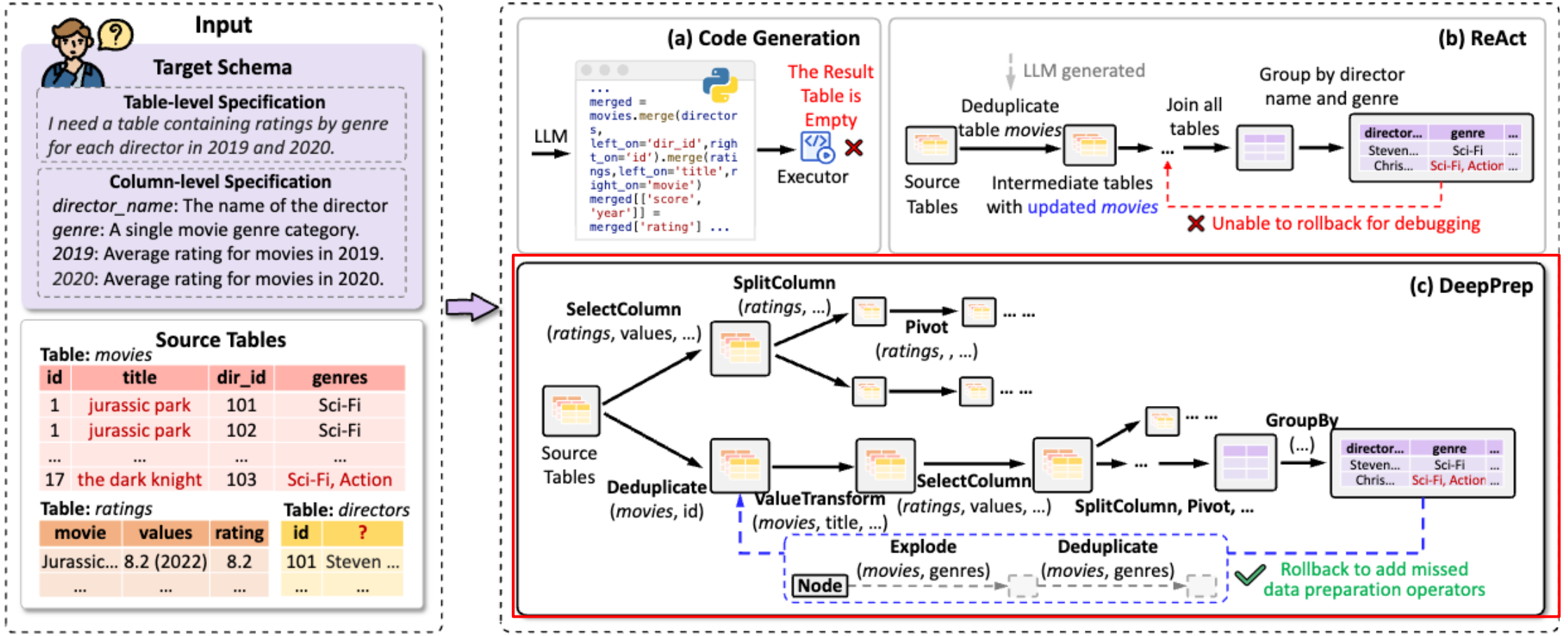


DeepAnalyze: Agentic Large Language Models for Autonomous Data Science

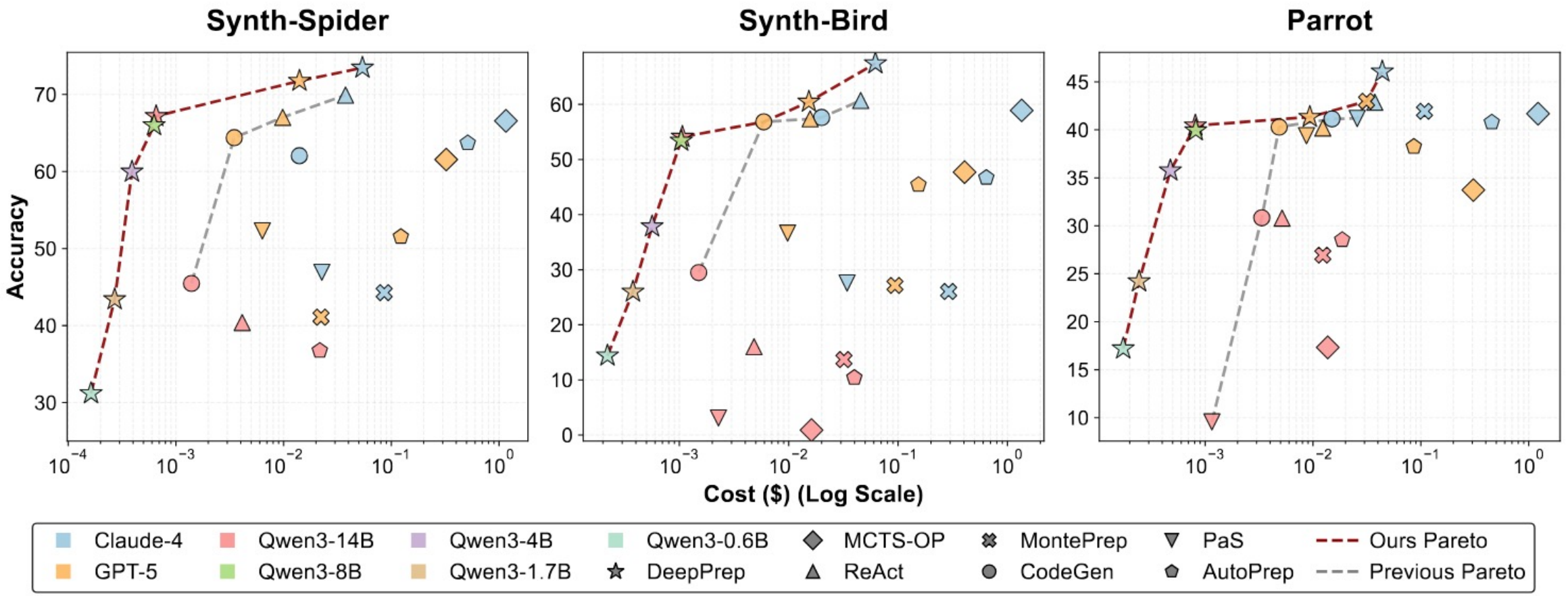


DeepPrep [ArXiv 2026]: An LLM-Powered Agentic System for Autonomous Data Preparation

- An Extension of DeepAnalyze for Operator-Based Pipeline Orchestration in (Semantic) Tabular Data Processing
  - Supporting 30 types of operators, e.g., cleaning, transformation, etc.



DeepPrep: An LLM-Powered Agentic System for Autonomous Data Preparation



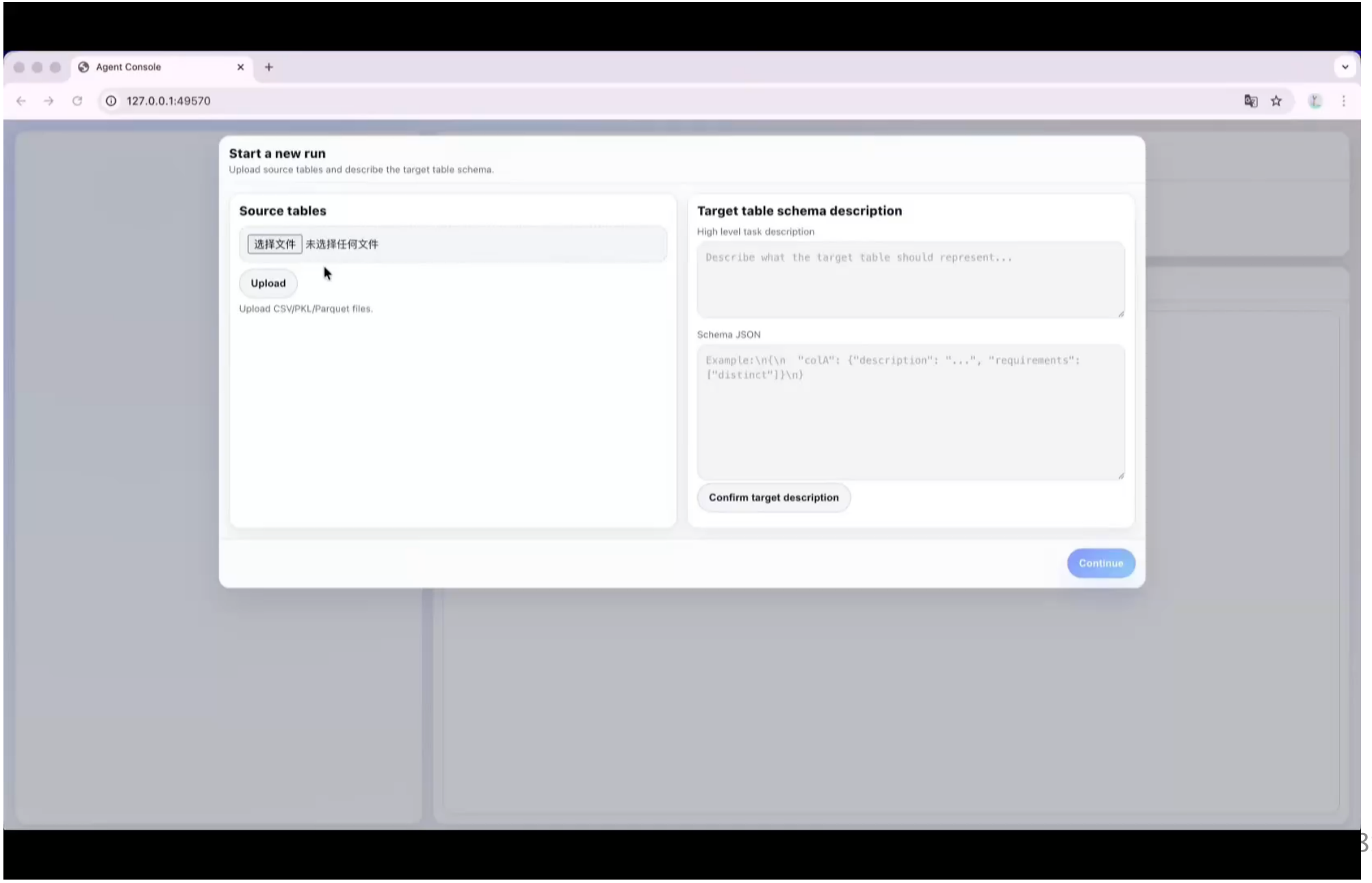
DeepPrep **extends the cost–accuracy Pareto frontier**, achieving accuracy close to baselines built on strong close-source LLMs at substantially lower inference cost.

DeepPrep: An LLM-Powered Agentic System for Autonomous Data Preparation

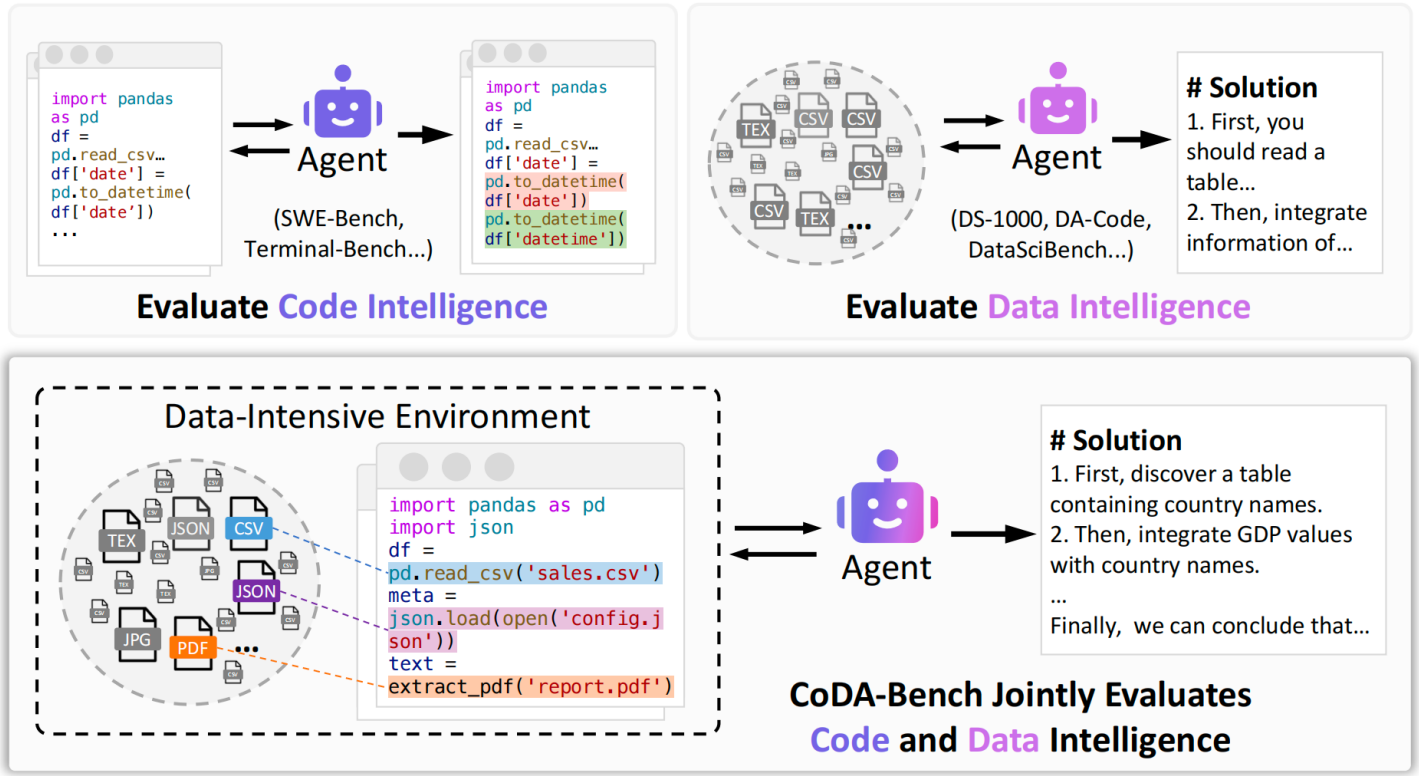
Model & Data

GitHub

Code



CoDA-Bench [ICML 2026]: Can Code Agents Handle Data-Intensive Tasks?



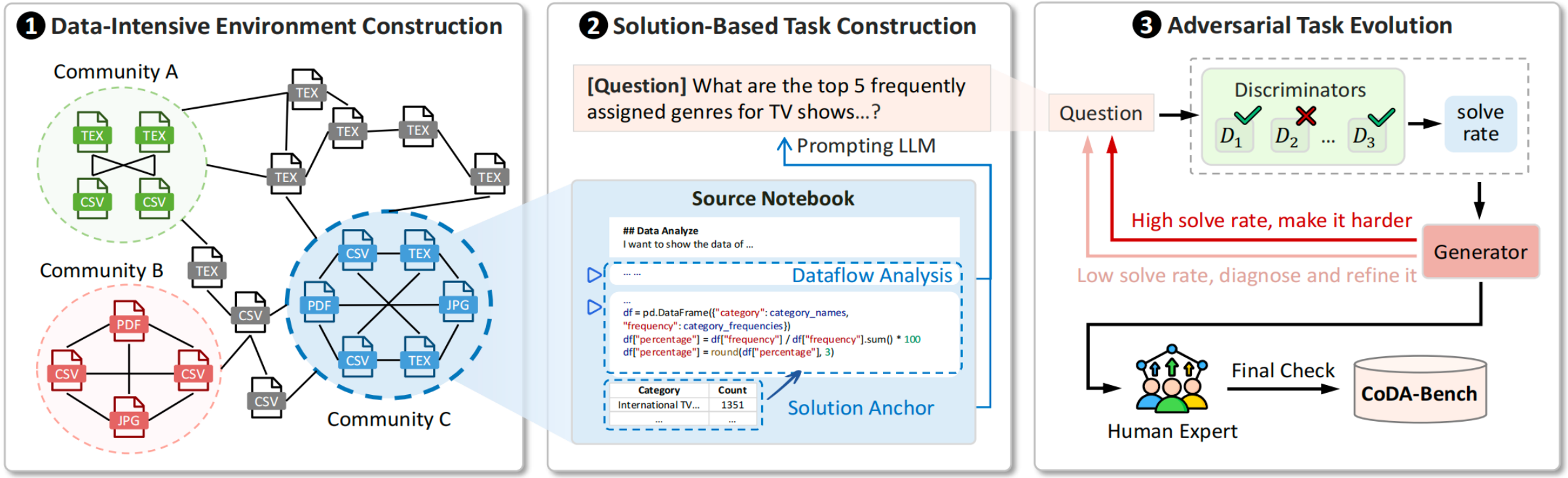
Existing benchmarks evaluate only one side:

- Code benchmarks such as SWE-Bench focus on repo-level coding, code repair, or terminal operations, but rarely test large-scale data discovery.
- Data benchmarks such as DA-Code usually provide the target data, reducing the need to search among noisy files.

Real data-intensive tasks require both **Code Intelligence** and **Data Intelligence**.

☆ CoDA-Bench jointly evaluates code and data intelligence in a data-intensive Linux sandbox.

CoDA-Bench: Can Code Agents Handle Data-Intensive Tasks?



How CoDA-Bench Is Built:

1. Build a dataset co-occurrence graph from Kaggle datasets and notebooks.
2. Extract solution anchors from real Kaggle notebook results
3. Refine task difficulty with a generator-discriminator loop and human validation.

Co-occurrence Graph

21,122 nodes
93,727 edges
323 communities

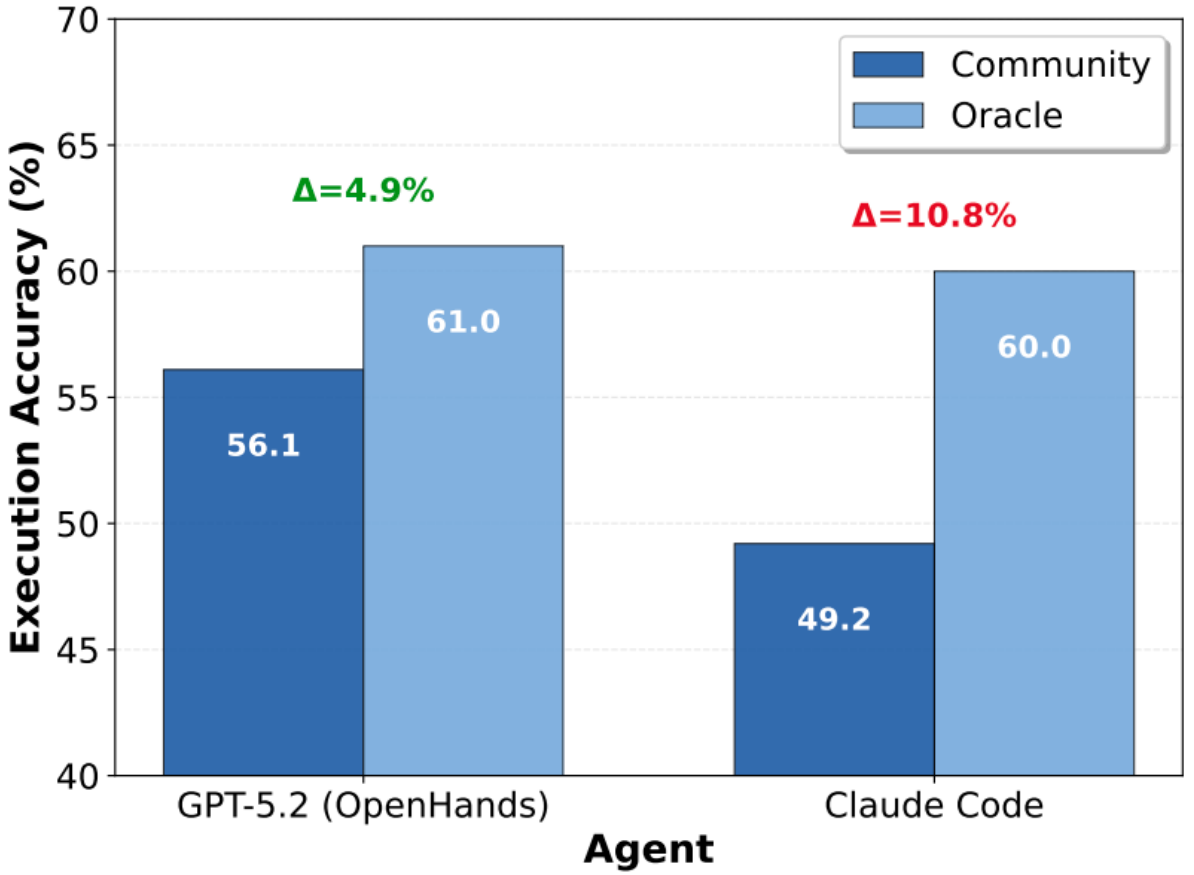
Selected:
53 communities
829 datasets

CoDA-Bench: Can Code Agents Handle Data-Intensive Tasks?

System	DA (%)	EA (%)	Hard EA (%)	Cost ($)
Openhands + GPT-5.2	73.3	56.1	36.2	0.73
Openhands + Gemini-3-Pro	81.2	54.9	39.5	1.29
Codex CLI + GPT-5.1-Codex	71.7	54.6	33.6	0.54

Main Findings:

1. Overall performance remains low; even strongest agent achieves only about 56% EA.
2. Data discovery is a key bottleneck, but not the only one; oracle data setting only partially improves performance.
3. Failures are mainly concentrated at the two ends: data discovery and code generation.



Even with the correct data, the accuracy only about 60.5%.

☆ Agents Struggle with Data-Intensive Tasks

L3 Orchestration — Gaps Toward L4

Limited autonomy

Still lean on predefined operators/tools. JoyAgent's “tool evolution” & AgenticData's code-gen are early steps; need continuous skill discovery.

Incomplete lifecycle

Mostly analysis + light preparation; data management (tuning, diagnosis) is under-served. Need versatile generalists.

Tactical, not strategic

Fix immediate errors but get stuck in loops. Need causal & meta-reasoning across processes.

Static environments

Evaluated on static data, ignoring drift. Need self-evolution + dynamic benchmarks.

L4: High Autonomy

From supervision to self-governance

Human supervision is removed. The agent proactively monitors and explores data lakes, discovers worthwhile tasks on its own, and plans over long horizons. The human becomes an onlooker.

What changes?

- **Autonomous problem discovery** – surface tasks (data drift, anomalies) without instructions.
- **Long-horizon, holistic planning** – cross-process cost / benefit trade-offs over time.
- **Trustworthy self-governance** – reliable generalist across the full lifecycle.
- **Cost-aware execution goes global** – optimize across many tasks, not one pipeline.

Outlook — L5: Full Autonomy

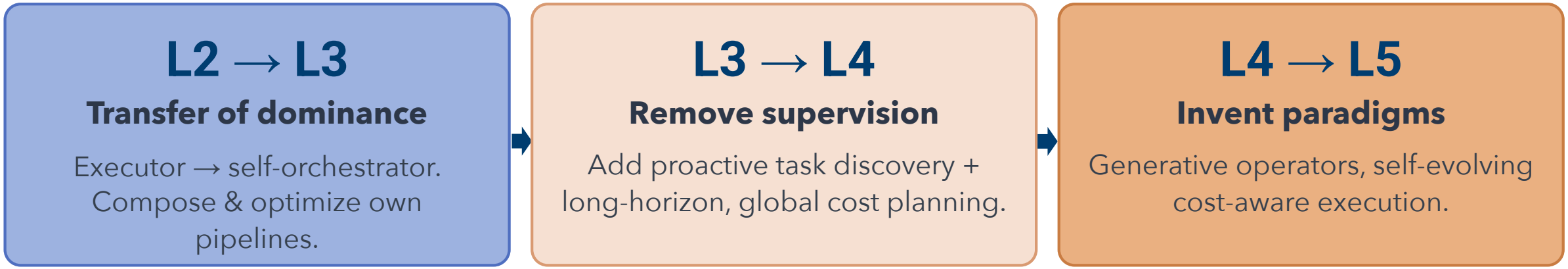
From applying methods to inventing them

The ultimate vision: a generative data scientist. When existing operators and pipelines fall short, the agent invents new methods, tools, and even workflow paradigms. The human disengages.

What changes?

- **Generative orchestration** — invent new operators & workflow paradigms on demand.
- **Self-devised cost models** — the agent designs its own execution strategies.
- **Pioneering examples** — new sampling theory, federated prep frameworks, novel viz grammars.
- **Human involvement** — unnecessary, and potentially detrimental.

Recap: Orchestration & Cost-aware Execution



- **Two coupled sub-problems.** Orchestration decides WHAT to do and in what order; cost-aware execution decides HOW to run it efficiently and reliably.
- **A clear storyline up the autonomy ladder.** L1 prompting → L2 predefined → L3 autonomous composition.
- **Open problems.** Operator / skill discovery, full-lifecycle coverage, strategic reasoning, and cost & safety guarantees stand between L3 and true autonomy.

Outline

- Part I: Data Agents: Motivation, Definition, Autonomy Levels, and Core Challenges
- **Part II: Towards Autonomous Data Agents: Key Challenges and Current Practices**
  - Data-aware Workflow Orchestration & Cost-aware Execution
  - **Long-horizon Agentic State, Memory & Skill Reuse**
  - Semantic Grounding over Heterogeneous & Multimodal Data
- Part III: Research Opportunities and Open Challenges

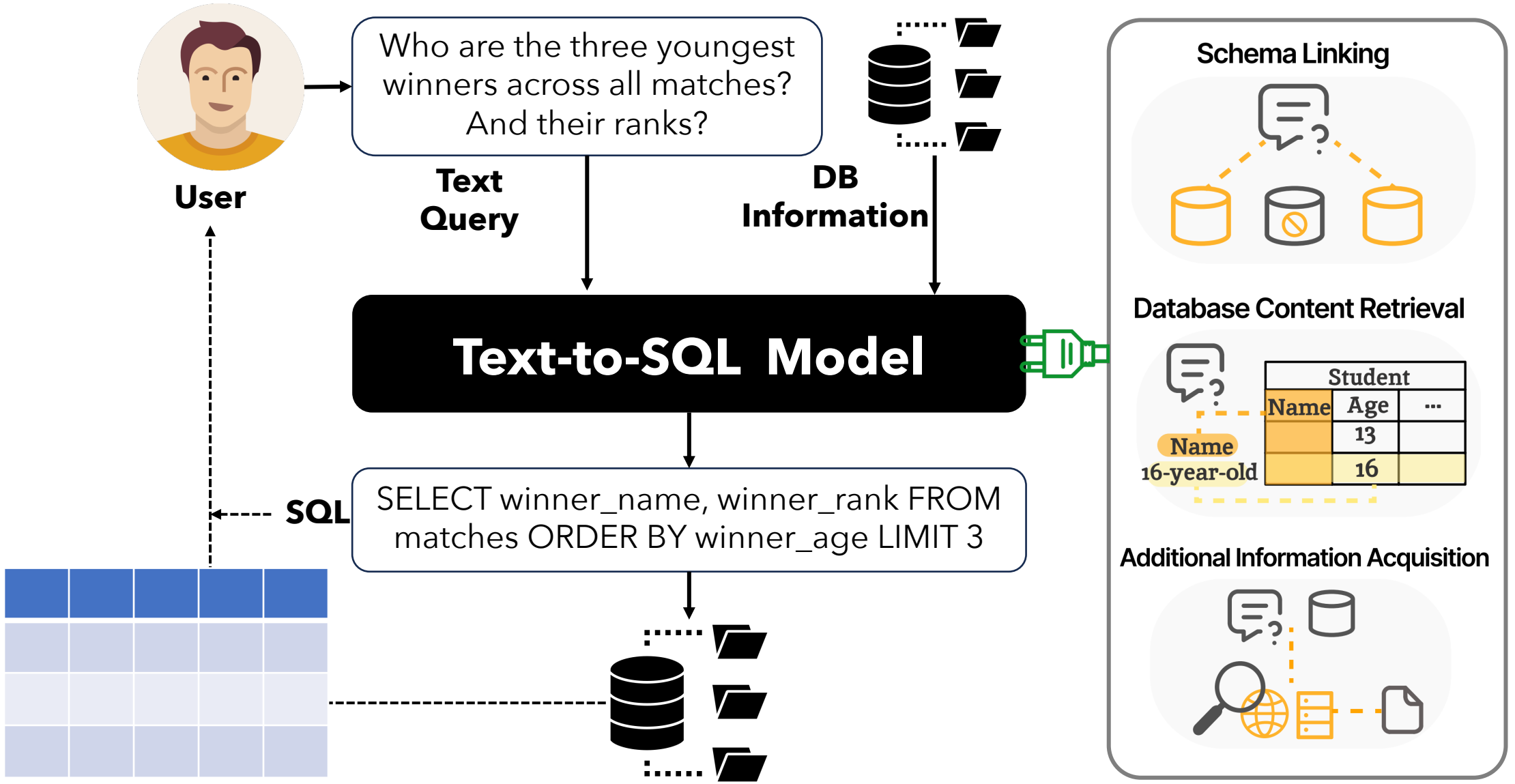
Core Data Agent Challenges and Tutorial Roadmap

Challenge \ Level	L1 Assistance	L2 Partial Autonomy	L3 Conditional Autonomy	L4 High Autonomy	L5 Full Autonomy
	Prompt-driven Data Assistant	Human-guided Workflow Executor	Agentic Workflow Orchestrator	Proactive Data Agent	Self-evolving Autonomous Data Agent
C1. Data-aware Workflow Orchestration & Cost-aware	Human-designed workflow; agent	Execute human-designed pipelines	Auto-compose and optimize workflows /	Proactively discover tasks and plan long-	Invent new operators, tools, and workflow
C2. Long-horizon Agentic State, Memory & Skill Reuse	Stateless; no persistent memory.	Short-term task memory and execution feedback.	Workflow state + provenance + reusable skills / SOPs.	Long-lived cross-task memory and shared skill reuse.	Self-evolving knowledge and skill base.
C3. Semantic Grounding over Heterogeneous & Multimodal Data	Prompt / few-shot / RAG-based grounding.	Environment-aware schema, entity, and evidence grounding.	Semantic catalog + metadata + multimodal evidence alignment.	Continuous semantic maintenance over evolving data lakes.	Create new semantic abstraction and representations.

Long-horizon Agentic State, Memory and Skill Reuse (~45 min)

Progression Assist & Suggest » Execute & Improve » Orchestrate & Optimize » Proactively Act » Invent & Evolve

Motivating Example (Text-to-SQL)

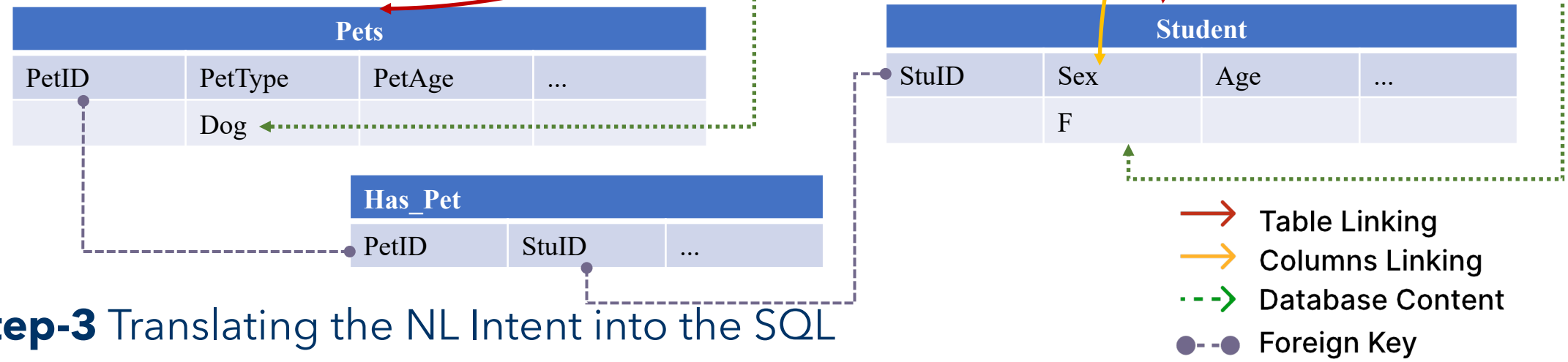


Text-to-SQL (Human Workflow)

Step-1 NL Understanding

Find the number of dog pets that are raised by female student

Step-2 Schema Linking and Database Content Retrieval

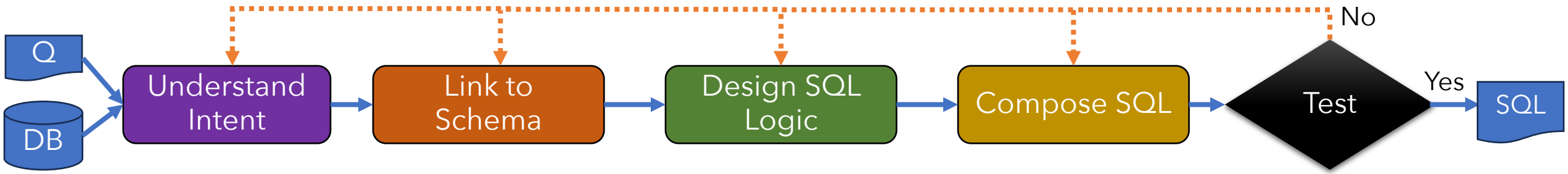


Step-3 Translating the NL Intent into the SQL

Select count(*) *FROM* student *AS* T1 *JOIN* has_pet *AS* T2 *ON* T1.stuid=T2.stuid
JOIN pets *AS* T3 *ON* T2.petid=T3.petid *WHERE* T1.sex='F' *AND* T3.pettype='Dog'

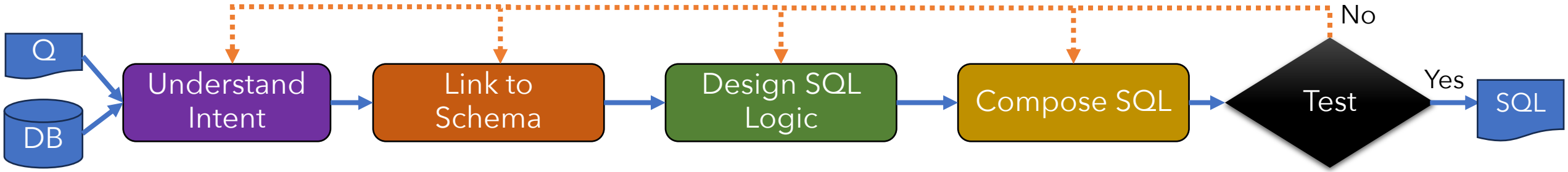
Task Formulation: Mimic Human Experts

- Human Expert Workflow for Text-to-SQL



Task Formulation: Mimic Human Experts

- Human Expert Workflow for Text-to-SQL

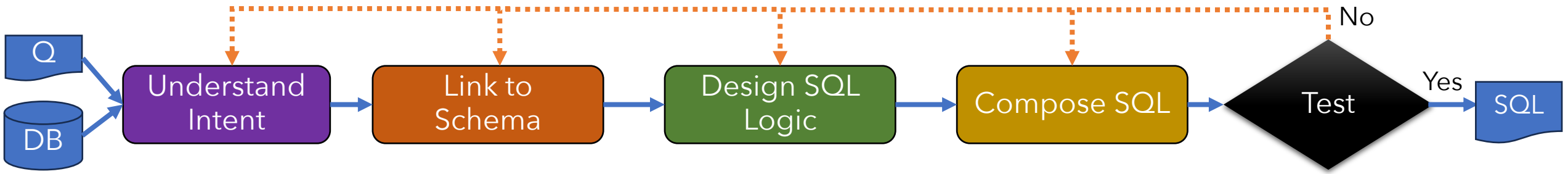


- From ***Human Actions*** to ***Agent Actions***

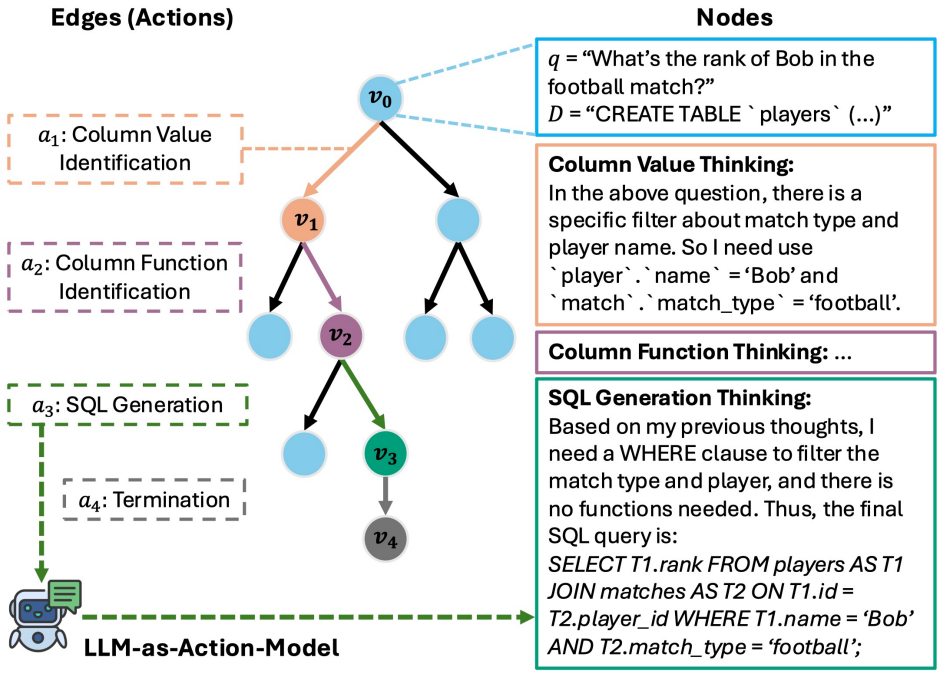
Understand Intent	**Question Rephrasing**	(Revise, clarify ambiguities, rephrasing)
Link to Schema	**Schema Selection** **Cell Value Selection**	(decides which tables / columns / values)
Design SQL Logic	**Column Function**	(joins, aggregations, functions)
Compose SQL	**SQL Generation**	(assemble an initial executable query)
Validate & Iter.	**SQL Revision**	(iteratively test, debug, and optimize the query)

Task Formulation: Mimic Human Experts

- Human Expert Workflow for Text-to-SQL



- From the ***Fixed*** pipelines to ***Dynamic*** (Pipelines) Actions



Tree-based Search:

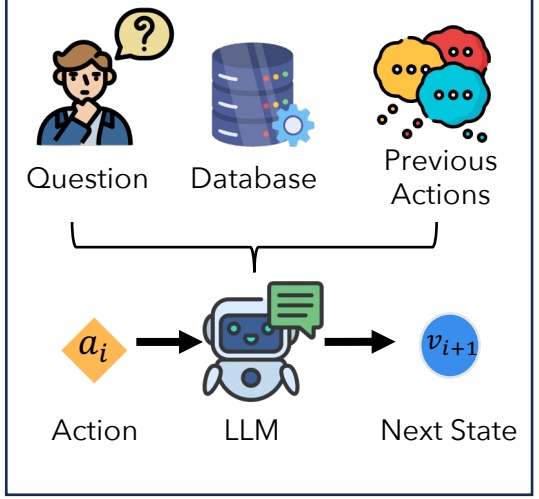
- Each ***edge*** corresponds to an ***agentic action*** in the query construction process,
- Each ***node*** represents a ***reasoning state*** at a specific step, and
- Each ***path*** corresponds ***to a sequence of SQL construction actions*** for Text-to-SQL task.

How Does a Data Agent Work on a Text-to-SQL Task?

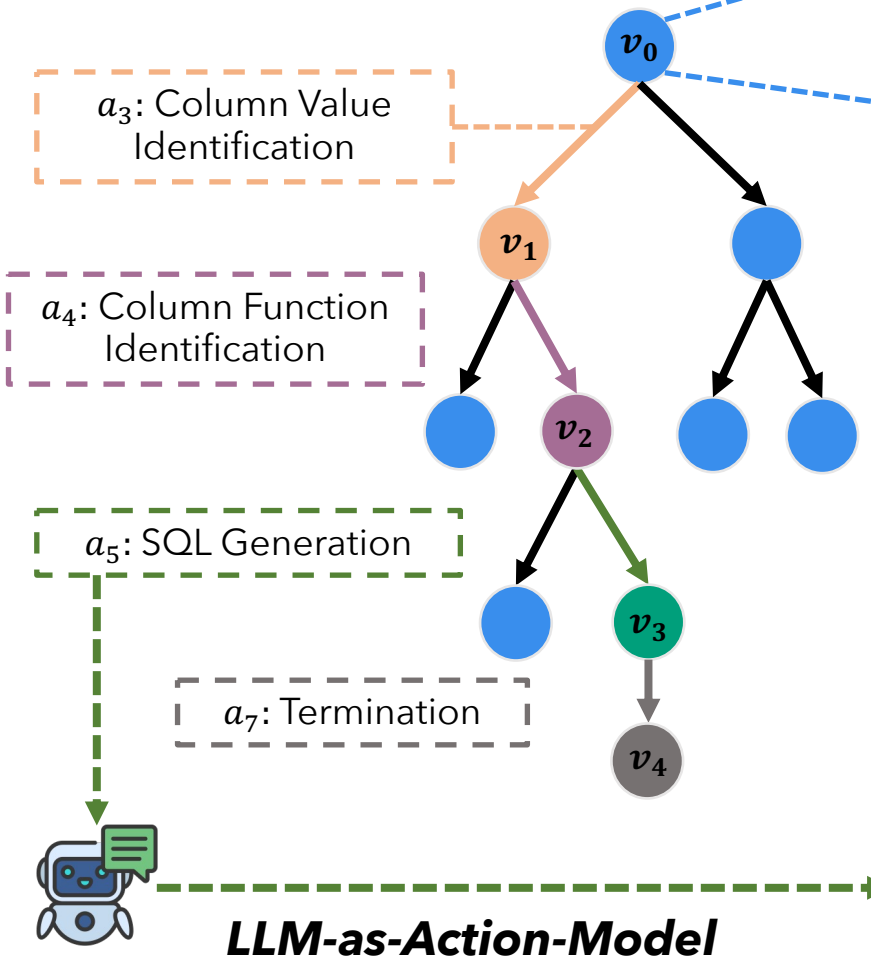
Action Space

- a_1 Rephrase Question
- a_2 Schema Selection
- a_3 Column Value Identification
- a_4 Column Function Identification
- a_5 SQL Generation
- a_6 SQL Revision
- a_7 Termination

LLM-as-Action-Model



Edges (Actions)



Nodes (Reasoning States)

q = "What's the rank of Bob in the football match?"
D = "CREATE TABLE `players` (…)"

Input

Column Value Thinking:
In the above question, there is a specific filter about match type and player name. So I need use `player`.`name` = 'Bob' and `match`.`match_type` = 'football'.

Column Function Thinking: …

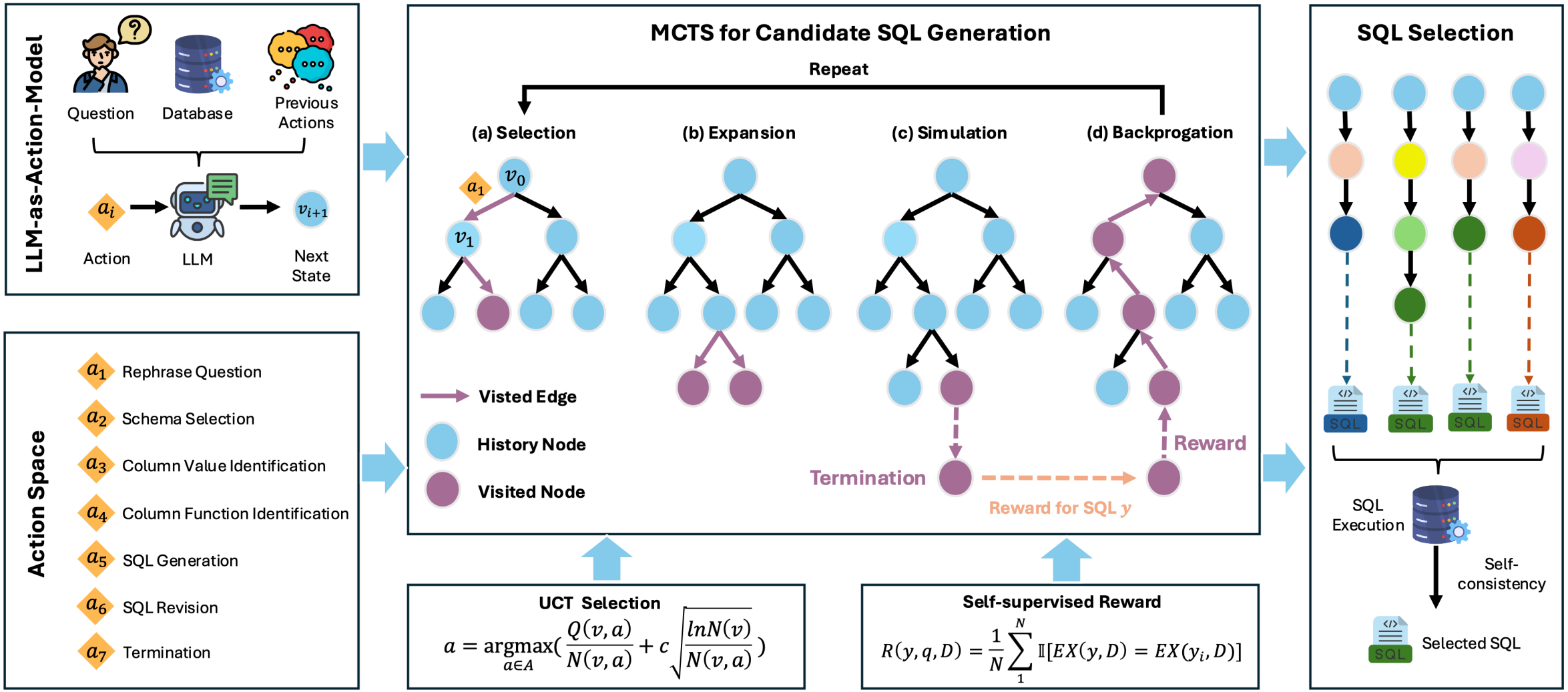
SQL Generation Thinking:
Based on my previous thoughts, I need a WHERE clause to filter the match type and player, and there is no functions needed. Thus, the final SQL query is:
SELECT T1.rank FROM players AS T1 JOIN matches AS T2 ON T1.id = T2.player_id WHERE T1.name = 'Bob' AND T2.match_type = 'football';

Output

LLM-as-Action-Model

Boyan Li, et al, Alpha-SQL: Zero-Shot Text-to-SQL using Monte Carlo Tree Search, **ICML** 2025.
https://github.com/HKUSTDial/Alpha-SQL

Alpha-SQL Solution Overview



Boyan Li, Yuyu Luo, Alpha-SQL: Zero-Shot Text-to-SQL using Monte Carlo Tree Search, **ICML** 2025.
https://github.com/HKUSTDial/Alpha-SQL

Alpha-SQL Takeaway: Search State Is Task-Local Memory

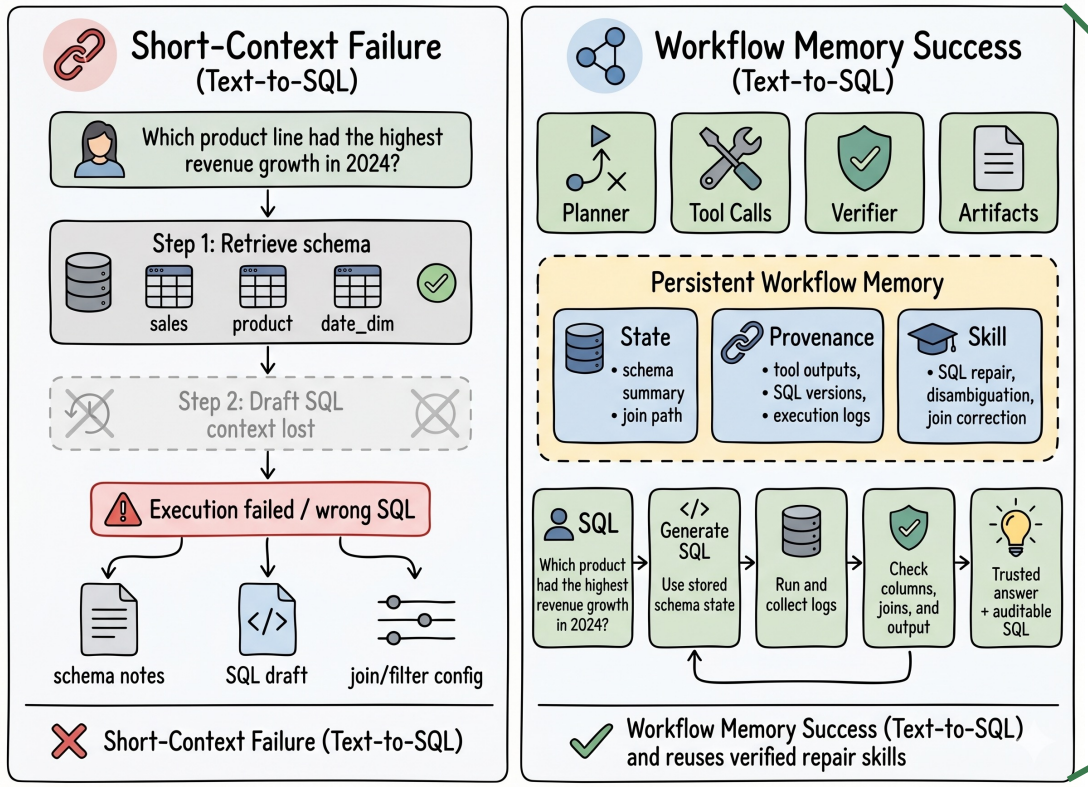
- Alpha-SQL shows that Text-to-SQL is not just one-shot SQL decoding. It can be formulated as a stateful search process over reasoning actions.

- **State**: partial reasoning path · selected schema/values/functions · candidate SQLs
- **Memory**: visited nodes · action history · execution/self-consistency feedback
- **Verification**: self-supervised reward and SQL execution guide search

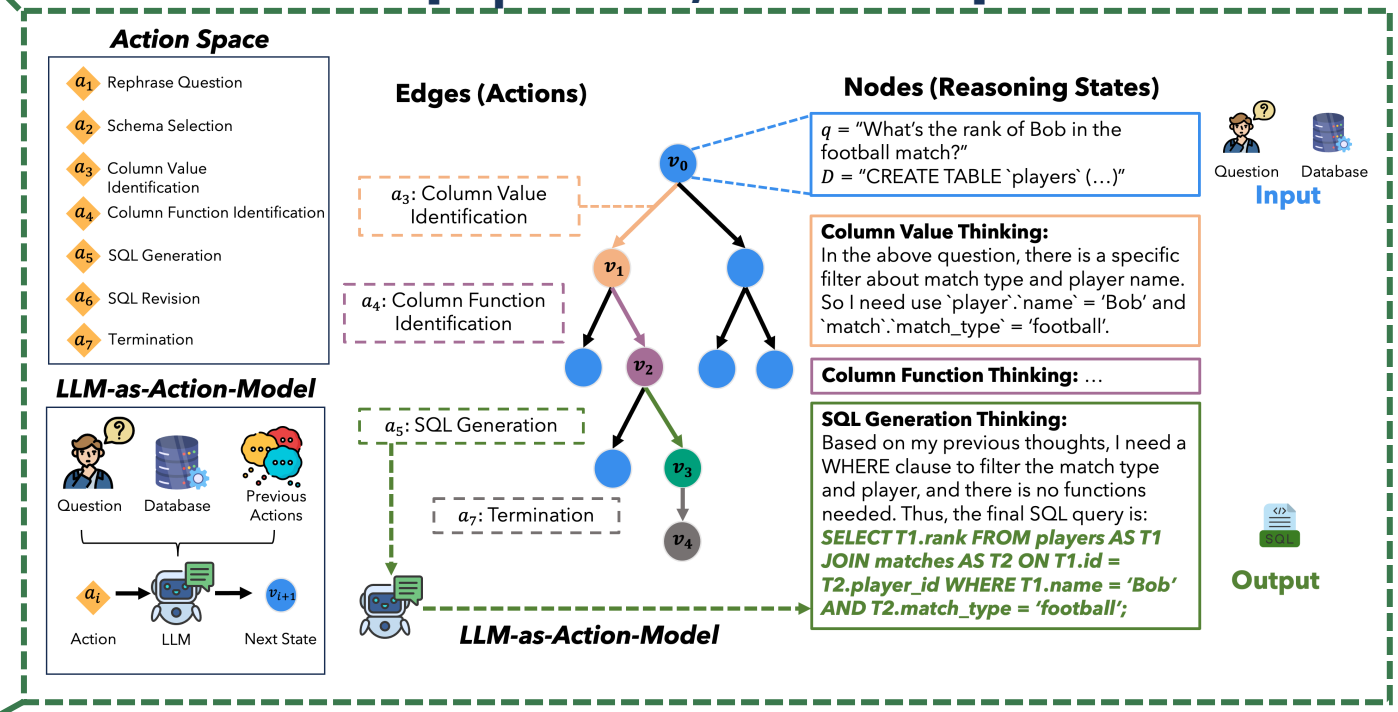
- **Design Rule:** Make intermediate reasoning states explicit before they can be searched, verified, or reused.

Boyan Li, Yuyu Luo, Alpha-SQL: Zero-Shot Text-to-SQL using Monte Carlo Tree Search, **ICML** 2025.
https://github.com/HKUSTDial/Alpha-SQL

What Should Be Remembered? (State, Memory, Skill)



[Alpha-SQL, ICML 2025]



State = where the workflow is now

Supports **Resume, Branching, Rollback.**

- **DM:** workload, plans, metrics, actions
- **DP:** versions, cleaning steps, lineage
- **DA:** hypotheses, SQL/code, artifacts

Memory = what happened and why

Supports **Reasoning, Audit, Verification.**

Skill = what can be packaged and reused

Supports **Reuse and Adaptation**

Takeaway: *Not all memory is text. In data agents, structured state often matters more than chat logs.*

Evolution Across L1–L5 for "Long-horizon Data Tasks"

Level	Agentic State	Memory Form	What is Remembered	Skill Reuse Capability
L1 Assistance Prompt-driven Data Assistant	**No persistent state**	Chat history	Current prompt / dialogue only	None
L2 Partial Autonomy Human-guided Workflow Executor	**Task-local execution state**	Task-local memory	Intermediate results, execution logs, error feedback	Local retry / ad-hoc scripts
L3 Conditional Autonomy Agentic Workflow Orchestrator	**Workflow-level state**	Workflow memory	Plan, DAG state, provenance, artifacts, validation results	Reusable SOPs / procedural skills
L4 High Autonomy Proactive Data Agent	**Cross-task / cross-agent state**	Long-lived shared memory	Historical tasks, user/ domain preferences, shared experience	Shared skill library across agents/tasks
L5 Full Autonomy Self-evolving Autonomous Data Agent	**Self-evolving process state**	Self-evolving skill base	Failure patterns, learned methods, new operators, meta-experience	Autonomous skill creation, refinement, and reuse

- The key frontier today is the transition from **L2 to L3**, from **L2 Task-Local** to **L3 Workflow Memory**
- L4/L5 are not just "more storage"; they represent advanced **memory management and skill evolution**.

Evolution Across L1–L5 for "Long-horizon Data Tasks"

Level	Agentic State	Memory Form	What is Remembered	Skill Reuse Capability
L1 Assistance Prompt-driven Data Assistant	**No persistent state**	Chat history	Current prompt / dialogue only	None
L2 Partial Autonomy Human-guided Workflow Executor	**Task-local execution state**	Task-local memory	Intermediate results, execution logs, error feedback	Local retry / ad-hoc scripts
L3 Conditional Autonomy Agentic Workflow Orchestrator	**Workflow-level state**	Workflow memory	Plan, DAG state, provenance, artifacts, validation results	Reusable SOPs / procedural skills
L4 High Autonomy Proactive Data Agent	**Cross-task / cross-agent state**	Long-lived shared memory	Historical tasks, user/ domain preferences, shared experience	Shared skill library across agents/tasks
L5 Full Autonomy Self-evolving Autonomous Data Agent	**Self-evolving process state**	Self-evolving skill base	Failure patterns, learned methods, new operators, meta-experience	Autonomous skill creation, refinement, and reuse

- The key frontier today is the transition from **L2 to L3**, from **L2 Task-Local** to **L3 Workflow Memory**
- L4/L5 are not just "more storage"; they represent advanced **memory management and skill evolution**.

Evolution Across L1–L5 for "Long-horizon Data Tasks"

Level	Agentic State	Memory Form	What is Remembered	Skill Reuse Capability
L1 Assistance Prompt-driven Data Assistant	**No persistent state**	Chat history	Current prompt / dialogue only	None
L2 Partial Autonomy Human-guided Workflow Executor	**Task-local execution state**	Task-local memory	Intermediate results, execution logs, error feedback	Local retry / ad-hoc scripts
L3 Conditional Autonomy Agentic Workflow Orchestrator	**Workflow-level state**	Workflow memory	Plan, DAG state, provenance, artifacts, validation results	Reusable SOPs / procedural skills
L4 High Autonomy Proactive Data Agent	**Cross-task / cross-agent state**	Long-lived shared memory	Historical tasks, user/ domain preferences, shared experience	Shared skill library across agents/tasks
L5 Full Autonomy Self-evolving Autonomous Data Agent	**Self-evolving process state**	Self-evolving skill base	Failure patterns, learned methods, new operators, meta-experience	Autonomous skill creation, refinement, and reuse

- The key frontier today is the transition from **L2 to L3**, from **L2 Task-Local** to **L3 Workflow Memory**
- L4/L5 are not just "more storage"; they represent advanced **memory management and skill evolution**.

Evolution Across L1–L5 for "Long-horizon Data Tasks"

Level	Agentic State	Memory Form	What is Remembered	Skill Reuse Capability
L1 Assistance Prompt-driven Data Assistant	**No persistent state**	Chat history	Current prompt / dialogue only	None
L2 Partial Autonomy Human-guided Workflow Executor	**Task-local execution state**	Task-local memory	Intermediate results, execution logs, error feedback	Local retry / ad-hoc scripts
L3 Conditional Autonomy Agentic Workflow Orchestrator	**Workflow-level state**	Workflow memory	Plan, DAG state, provenance, artifacts, validation results	Reusable SOPs / procedural skills
L4 High Autonomy Proactive Data Agent	**Cross-task / cross-agent state**	Long-lived shared memory	Historical tasks, user/ domain preferences, shared experience	Shared skill library across agents/tasks
L5 Full Autonomy Self-evolving Autonomous Data Agent	**Self-evolving process state**	Self-evolving skill base	Failure patterns, learned methods, new operators, meta-experience	Autonomous skill creation, refinement, and reuse

- The key frontier today is the transition from **L2 to L3**, from **L2 Task-Local** to **L3 Workflow Memory**
- L4/L5 are not just "more storage"; they represent advanced **memory management and skill evolution**.

Evolution Across L1–L5 for "Long-horizon Data Tasks"

Level	Agentic State	Memory Form	What is Remembered	Skill Reuse Capability
L1 Assistance Prompt-driven Data Assistant	**No persistent state**	Chat history	Current prompt / dialogue only	None
L2 Partial Autonomy Human-guided Workflow Executor	**Task-local execution state**	Task-local memory	Intermediate results, execution logs, error feedback	Local retry / ad-hoc scripts
L3 Conditional Autonomy Agentic Workflow Orchestrator	**Workflow-level state**	Workflow memory	Plan, DAG state, provenance, artifacts, validation results	Reusable SOPs / procedural skills
L4 High Autonomy Proactive Data Agent	**Cross-task / cross-agent state**	Long-lived shared memory	Historical tasks, user/ domain preferences, shared experience	Shared skill library across agents/tasks
L5 Full Autonomy Self-evolving Autonomous Data Agent	**Self-evolving process state**	Self-evolving skill base	Failure patterns, learned methods, new operators, meta-experience	Autonomous skill creation, refinement, and reuse

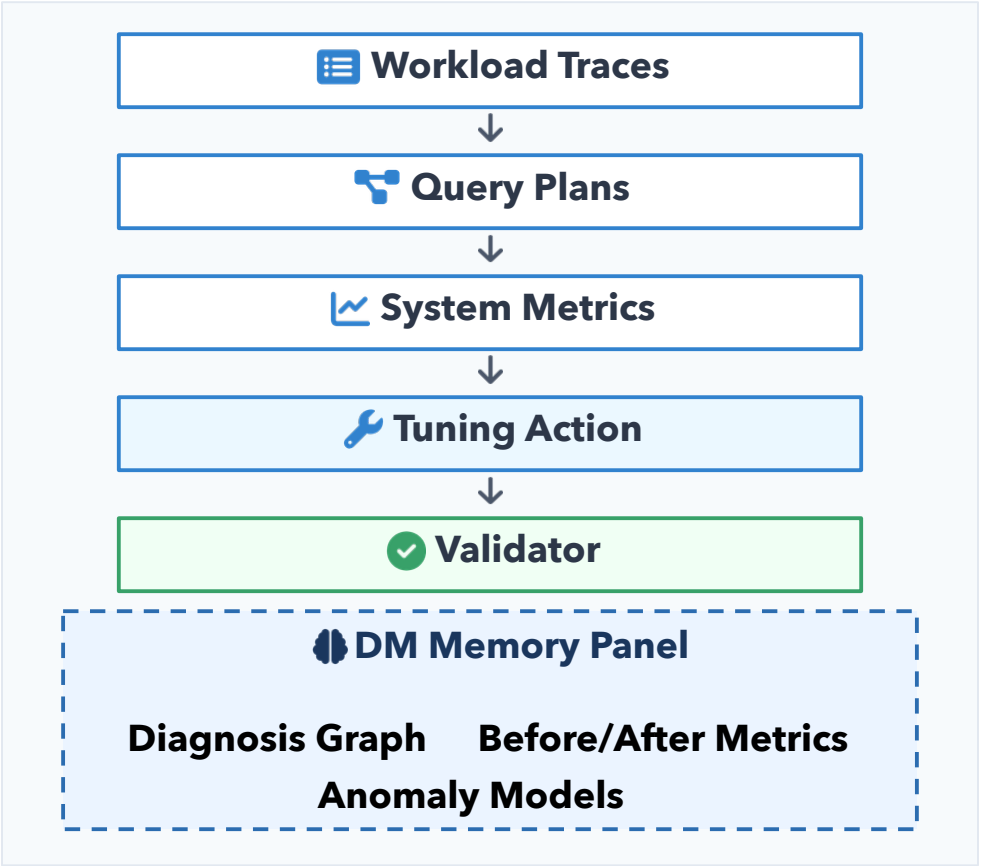
- The key frontier today is the transition from **L2 to L3**, from **L2 Task-Local** to **L3 Workflow Memory**
- L4/L5 are not just "more storage"; they represent advanced **memory management and skill evolution**.

Representative Papers for Long-horizon State, Memory, and Skill Reuse

Scenario	L1/L2-style anchor (task-local)	L3-style anchor (workflow / persistent memory)	Key Takeaway for "Long-horizon Data Tasks"
Data Management	**λ-Tune** LLM-generated complete DB configurations (SIGMOD 2025)	**AgentTune** Multi-agent DB knob tuning with workload feedback (SIGMOD 2026) **DBAIOps** KG-grounded O&M diagnosis memory + "800+" anomaly models (VLDB 2026)	From task-local configuration candidates to **feedback-driven tuning** and **KG-grounded diagnosis memory**: workload traces, query plans, metrics, anomaly patterns, and verified action effects become reusable DBA playbooks.
Data Preparation	**CleanAgent** Declarative APIs + LLM agents for data standardization (VLDB 2025 Workshop)	**AutoPrep** Planner–Programmer–Executor data-prep workflow (VLDB 2025)	From column-level standardization to **question-aware preparation workflows**: table versions, generated code, execution feedback, and transformation provenance make data prep resumable and reproducible.
Data Analysis	**Alpha-SQL** MCTS over partial SQL / reasoning states (ICML 2025)	**DeepEye** Workflow engine + memory-augmented planner + glass-box DAG + SOPs (SIGMOD 2026)	From partial-SQL search states to **auditable analysis workflows**: hypotheses, SQL / code, execution logs, evidence chains, charts / reports, and user edits become inspectable DAG artifacts and reusable analysis skills.

Scenario Lens: Data Management

- **Slow-query diagnosis and DB maintenance need stateful evidence, not just one-off tuning suggestions.**



- DM must remember **workload traces, query plans, system metrics, and action effects**.
- Every recommendation should be **verified against performance and operational safety**.
- The reusable unit is a **diagnosis / tuning playbook**.

L2 **λ-Tune (SIGMOD 2025)**

Task-local candidate tuning from current workload/context.

L3/L4-style **DBAIOps (VLDB 2026)**

Diagnosis experience as knowledge graph + **800+ reusable anomaly models**.

DM memory must be safety-aware and rollback-aware

λ-Tune [SIGMOD 2025]

λ-Tune: Harnessing Large Language Models for Automated Database System Tuning

Victor Giannakouris
Cornell University
Ithaca, NY, USA
vg292@cornell.edu

Immanuel Trummer
Cornell University
Ithaca, NY, USA
it224@cornell.edu

Abstract

We introduce λ-Tune, a framework that leverages Large Language Models (LLMs) for automated database system tuning. The design of λ-Tune is motivated by the capabilities of the latest generation of LLMs. Different from prior work, leveraging LLMs to extract tuning hints for single parameters, λ-Tune generates entire configuration scripts, based on a large input document, describing the tuning context. λ-Tune generates alternative configurations, using a principled approach to identify the best configuration, out of a small set of candidates. In doing so, it minimizes reconfiguration overheads and ensures that evaluation costs are bounded as a function of the optimal run time. By treating prompt generation as a cost-based optimization problem, λ-Tune conveys the most relevant context to the LLM while bounding the number of input tokens and, therefore, monetary fees for LLM invocations. We compare λ-Tune to various baselines, using multiple benchmarks and PostgreSQL and MySQL as target systems for tuning, showing that λ-Tune is significantly more robust than prior approaches.

Keywords

Database, Tuning, Large Language Models, Physical Design

ACM Reference Format:

Victor Giannakouris and Immanuel Trummer. 2018. λ-Tune: Harnessing Large Language Models for Automated Database System Tuning . In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX)*. ACM, New York, NY, USA, 14 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

The performance of database management system changes dramatically as a function of various tuning choices, including settings for system configuration parameters as well as physical design choices such as indexing, sorting, or partitioning. This has motivated a large body of research on automated database system tuning. Recent work exploits machine learning to find near-optimal configurations [5, 6, 13, 23] but suffers from high training and exploration overheads. This has motivated a new line of research [11, 20], exploiting LLMs to heuristically prune the search space for tuning. Similar to human database administrators, such models leverage commonsense knowledge, extracted from text documents, to narrow the focus to tuning options that seem "reasonable", given the tuning context. This paper presents λ-Tune (LAnguage Models for Better Database Administration), a system that exploits capabilities offered by the latest generation of LLMs, including the likes of GPT-4 and Claude 3, to optimize various tuning choices for specific systems and OLAP workloads, including system parameter settings as well as physical design decisions.

λ-Tune. Prior approaches to LLM-enhanced database tuning [11, 20] parse text documents (e.g., the database manual) to extract value recommendations for specific parameters. They still need to perform an optimization stage in which hints about specific parameters are combined into complete configurations. This approach is in line with the limitations of early-stage language models such as BERT [4] and GPT-2 [14]. For those models, input and output sizes are limited to a few hundred tokens, restricting the scope of these models to settings for single parameters (rather than entire configurations). Modern LLMs such as GPT-4 support input and output sizes of hundreds of thousands of tokens. The design of λ-Tune is motivated by these advances. It exploits increased input sizes by feeding to the language model a description of all information relevant for tuning, including the workload and target system. It also exploits the increased output size by generating entire configurations, rather than hints about single parameters. As shown in our experiments, modern LLMs such as GPT-4 are typically able to map information about the workload to efficient database configuration settings. Hence, unlike prior systems, λ-Tune avoids expensive optimization steps, combining settings for single parameters. Instead, it delegates more responsibility to the language model itself.

Prompt Generation. First, λ-Tune automates the *prompt generation* step by crafting prompts tailored to the input workload (analytical SQL queries), hardware specifications, and the database system. Our approach incorporates a workload representation method that decomposes the input SQL queries into much smaller, mergeable components, called *query snippets*. As costs increase in the prompt size, minimizing monetary fees while conveying the most relevant information is challenging. We select the most informative subset of snippets to include in the prompt, given a bound on the number of prompt tokens (which are proportional to processing fees for providers like OpenAI). We formulate workload representation as a cost-based optimization problem that we solve by a transformation to integer linear programming. Using the resulting prompt, λ-Tune

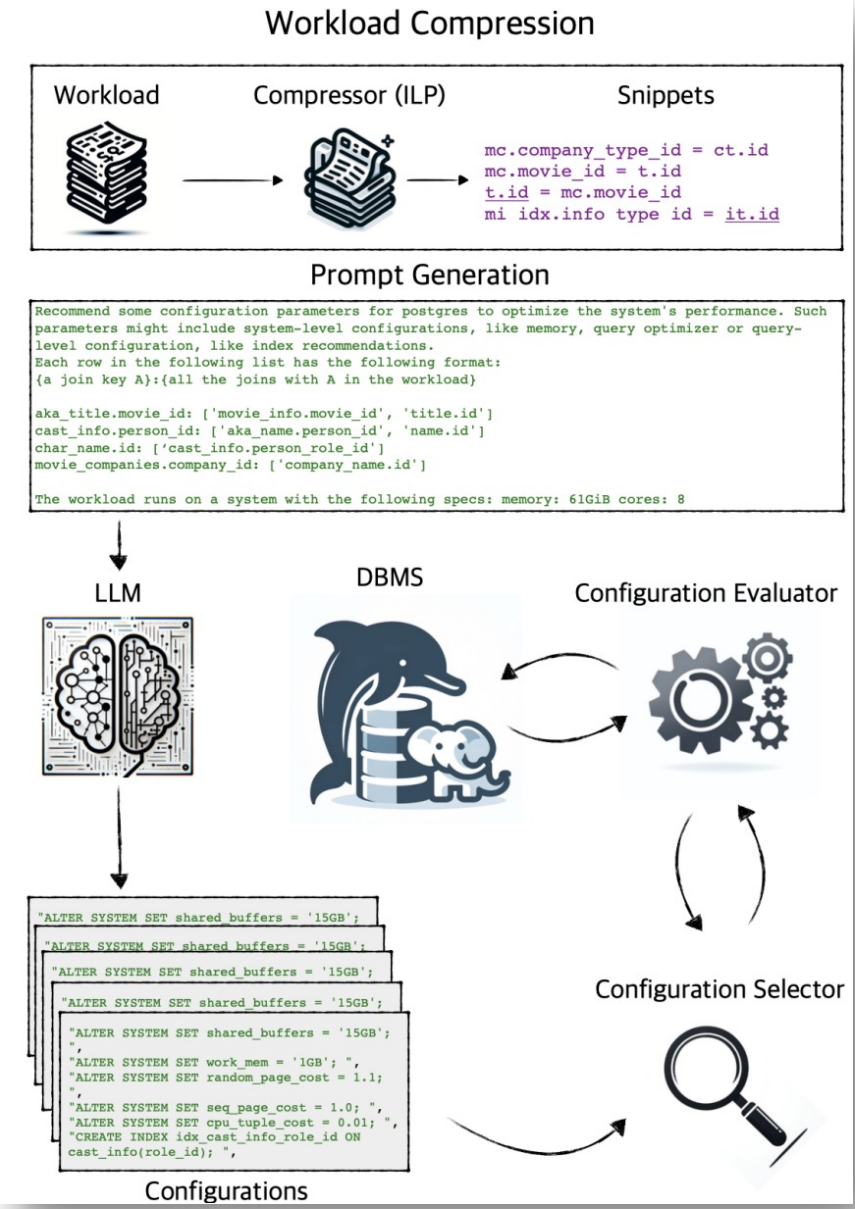
Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

- **Problem:** Harnessing Large Language Models for Automated OLAP Workload Tuning
- **Agent Level:** L2 Agent – LLM leverages pre-trained knowledge (memory & skills) to directly generate complete DB configurations
- **Challenge Addressed:** Long-horizon tuning with state management, knowledge reuse from pre-training, and skill transfer across workloads

λ-Tune System Architecture

Giannakouris V, Trummer I. λ-tune: Harnessing large language models for automated database system tuning. SIGMOD 2025



Input

OLAP Workload *W*, Hardware *H*, Database System *D*, Token Budget *B*

Output

Best Configuration (System Parameters + Physical Design)

Key Pipeline Steps

1. **Workload Compression:** SQL queries → ILP Compressor → Query Snippets
2. **Prompt Generation:** Template + Compressed Workload + Hardware Specs → Prompt
3. **LLM Invocation:** Send prompt to LLM (GPT-4) → Get *k* candidate configurations
4. **Configuration Selector:** Incremental evaluation with geometric timeouts
5. **Configuration Evaluator:** Lazy index creation + DP query scheduling

λ-Tune as an L2 Agent: Key Takeaway

1. Memory Reuse

- LLM's pre-trained weights encode DB tuning knowledge from manuals, forums, and best practices.
- Short-lived candidate-evaluation history

2. Skill Transfer

- No explicit long-lived reusable playbook; reuse is prompt/template level

3. State Management

- Current workload + hardware + candidate configs + measured outcomes
- Tracks configuration evaluation state across multiple rounds to avoid redundant work.

4. Bounded Exploration

- Principled approach (ILP + geometric timeout) instead of exhaustive search.

Strong verification loop; weak long-lived DBA memory

Key Distinction from Traditional ML Tuning

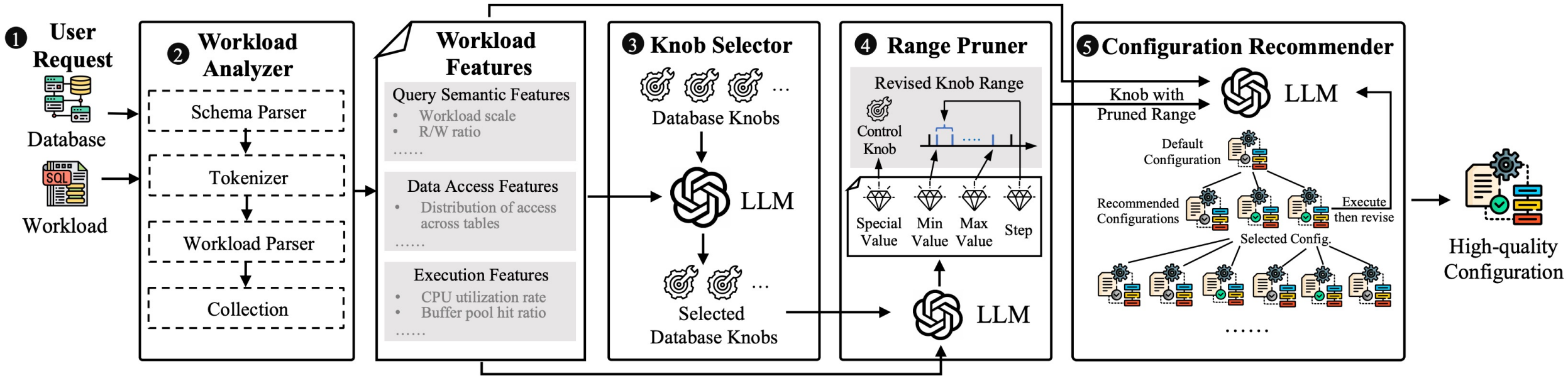
No training data needed (zero-shot approach)

No reward signal or RL exploration required

Leverages **"common sense"** from pre-training as reusable skills

Config Tuning: AgentTune (SIGMOD 2026)

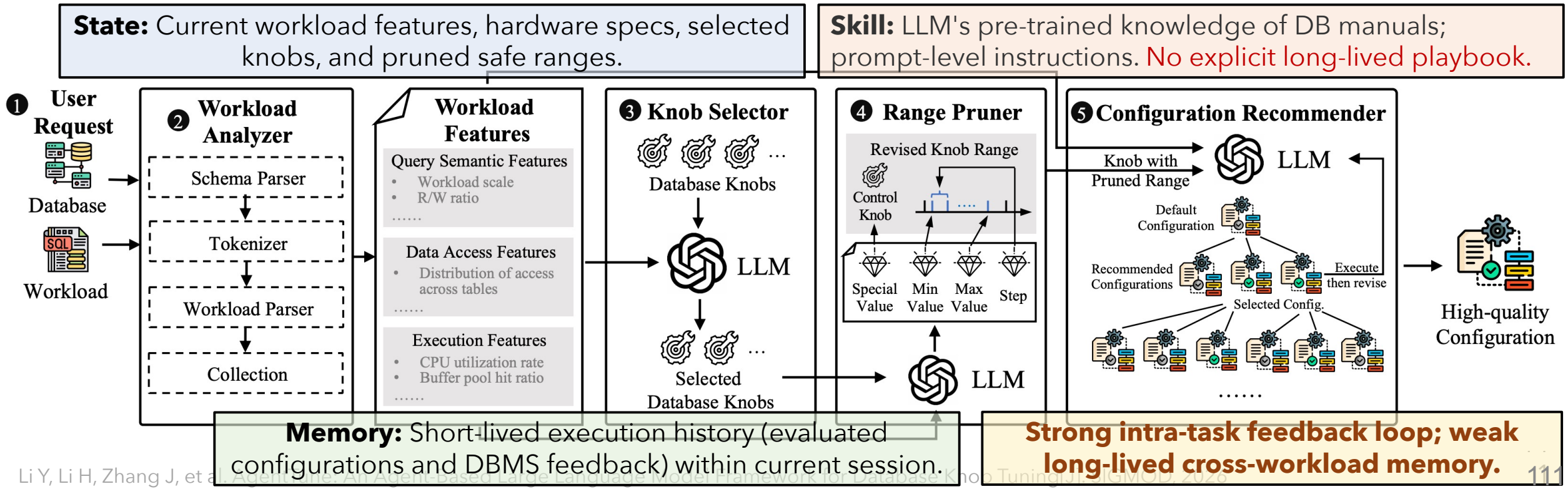
- **Problem:** High-dimensional continuous knob tuning space requiring safe exploration and domain expertise.
- **Core Method:** Decompose the tuning process into specialized agents: Workload Analyzer, Knob Selector, Range Pruner, and Configuration Recommender
- **Execution:** Tree-based iterative search generating and ranking multiple candidate configurations.
- **Verification:** Refine the configuration based on DBMS feedback (performance and execution features) and generate new candidate configurations for a beam search strategy



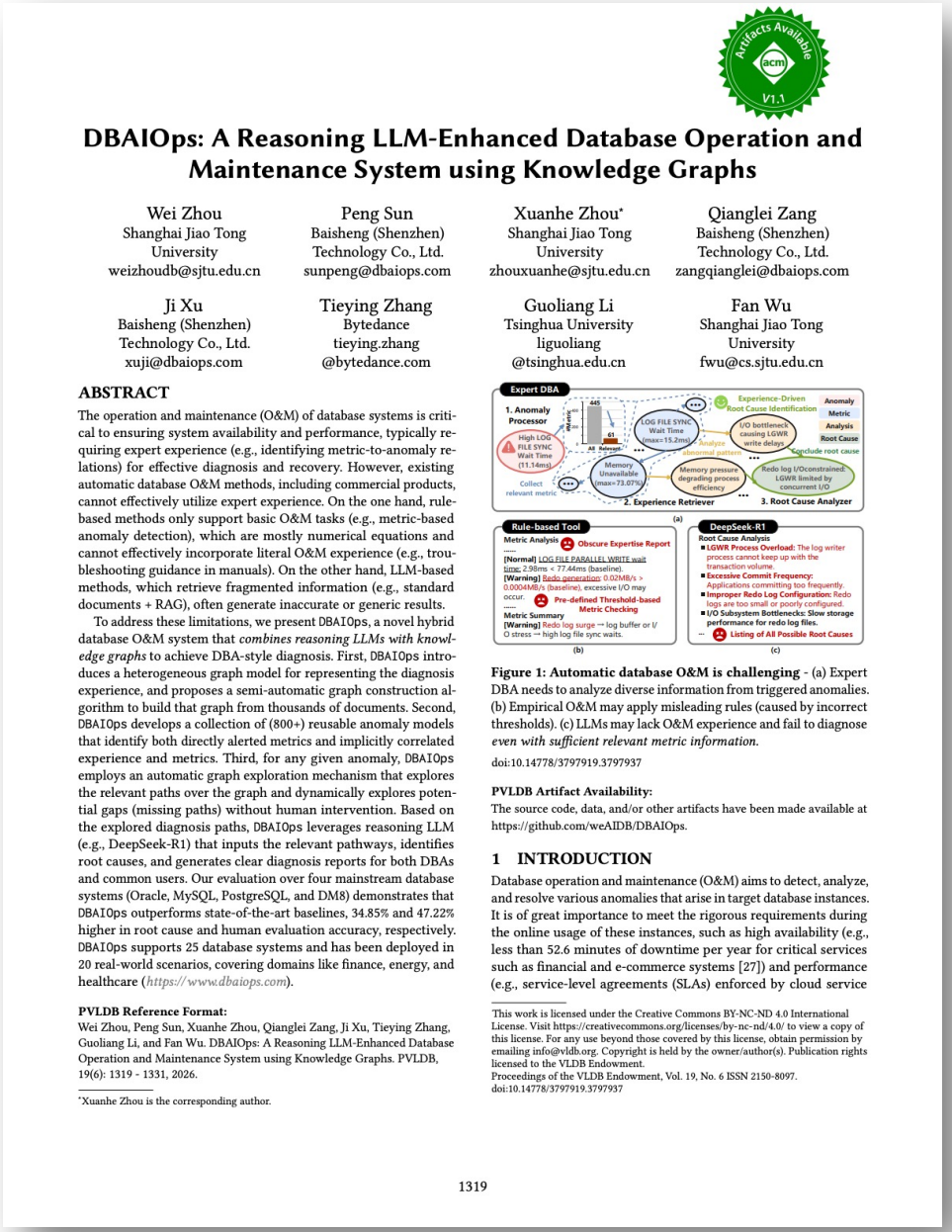
Li Y, Li H, Zhang J, et al. AgentTune: An Agent-Based Large Language Model Framework for Database Knob Tuning[J]. SIGMOD, 2026

Config Tuning: AgentTune (SIGMOD 2026)

- **Problem:** High-dimensional continuous knob tuning space requiring safe exploration and domain expertise.
- **Core Method:** Decompose the tuning process into specialized agents: Workload Analyzer, Knob Selector, Range Pruner, and Configuration Recommender
- **Execution:** Tree-based iterative search generating and ranking multiple candidate configurations.
- **Verification:** Refine the configuration based on DBMS feedback (performance and execution features) and generate new candidate configurations for a beam search strategy



DBAIOps [VLDB 2026]



Artifacts Available acm V1.1

DBAIOps: A Reasoning LLM-Enhanced Database Operation and Maintenance System using Knowledge Graphs

Wei Zhou
Shanghai Jiao Tong University
weizhoudb@sjtu.edu.cn

Peng Sun
Baisheng (Shenzhen) Technology Co., Ltd.
sunpeng@dbaiops.com

Xuanhe Zhou*
Shanghai Jiao Tong University
zhouxuanhe@sjtu.edu.cn

Qianglei Zang
Baisheng (Shenzhen) Technology Co., Ltd.
zangqianglei@dbaiops.com

Ji Xu
Baisheng (Shenzhen) Technology Co., Ltd.
xuji@dbaiops.com

Tieying Zhang
Bytedance
tieying.zhang @bytedance.com

Guoliang Li
Tsinghua University
liguoliang @tsinghua.edu.cn

Fan Wu
Shanghai Jiao Tong University
fwu@cs.sjtu.edu.cn

ABSTRACT

The operation and maintenance (O&M) of database systems is critical to ensuring system availability and performance, typically requiring expert experience (e.g., identifying metric-to-anomaly relations) for effective diagnosis and recovery. However, existing automatic database O&M methods, including commercial products, cannot effectively utilize expert experience. On the one hand, rule-based methods only support basic O&M tasks (e.g., metric-based anomaly detection), which are mostly numerical equations and cannot effectively incorporate literal O&M experience (e.g., troubleshooting guidance in manuals). On the other hand, LLM-based methods, which retrieve fragmented information (e.g., standard documents + RAG), often generate inaccurate or generic results.

To address these limitations, we present DBAIOps, a novel hybrid database O&M system that *combines reasoning LLMs with knowledge graphs* to achieve DBA-style diagnosis. First, DBAIOps introduces a heterogeneous graph model for representing the diagnosis experience, and proposes a semi-automatic graph construction algorithm to build that graph from thousands of documents. Second, DBAIOps develops a collection of (800+) reusable anomaly models that identify both directly alerted metrics and implicitly correlated experience and metrics. Third, for any given anomaly, DBAIOps employs an automatic graph exploration mechanism that explores the relevant paths over the graph and dynamically explores potential gaps (missing paths) without human intervention. Based on the explored diagnosis paths, DBAIOps leverages reasoning LLM (e.g., DeepSeek-R1) that inputs the relevant pathways, identifies root causes, and generates clear diagnosis reports for both DBAs and common users. Our evaluation over four mainstream database systems (Oracle, MySQL, PostgreSQL, and DM8) demonstrates that DBAIOps outperforms state-of-the-art baselines, 34.85% and 47.22% higher in root cause and human evaluation accuracy, respectively. DBAIOps supports 25 database systems and has been deployed in 20 real-world scenarios, covering domains like finance, energy, and healthcare (*https://www.dbaiops.com*).

PVLDB Reference Format:
Wei Zhou, Peng Sun, Xuanhe Zhou, Qianglei Zang, Ji Xu, Tieying Zhang, Guoliang Li, and Fan Wu. DBAIOps: A Reasoning LLM-Enhanced Database Operation and Maintenance System using Knowledge Graphs. PVLDB, 19(6): 1319 - 1331, 2026.

*Xuanhe Zhou is the corresponding author.

Figure 1: Automatic database O&M is challenging - (a) Expert DBA needs to analyze diverse information from triggered anomalies. (b) Empirical O&M may apply misleading rules (caused by incorrect thresholds). (c) LLMs may lack O&M experience and fail to diagnose *even with sufficient relevant metric information.*

doi:10.14778/3797919.3797937

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/weAIDB/DBAIOps.

1 INTRODUCTION

Database operation and maintenance (O&M) aims to detect, analyze, and resolve various anomalies that arise in target database instances. It is of great importance to meet the rigorous requirements during the online usage of these instances, such as high availability (e.g., less than 52.6 minutes of downtime per year for critical services such as financial and e-commerce systems [27]) and performance (e.g., service-level agreements (SLAs) enforced by cloud service

Proceedings of the VLDB Endowment, Vol. 19, No. 6 ISSN 2150-8097.
doi:10.14778/3797919.3797937

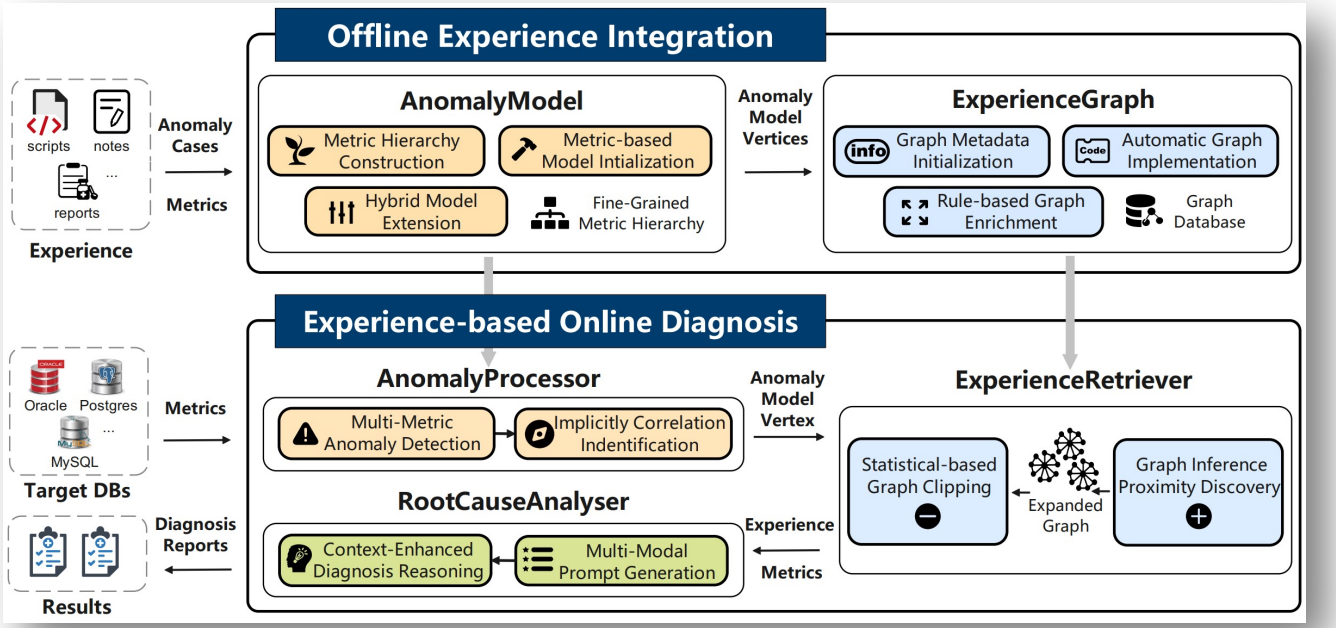
- **Problem:** Database O&M combining reusable graph-based expert diagnosis experience with the reasoning capability of LLMs.

- **L3-style Data Agent Level:**
  - workflow-level diagnosis with reusable anomaly models, and graph-based diagnosis paths.
  - long-lived shared O&M memory across 25 database systems and real deployments.

- **Challenge Addressed:**
  - **C1:** Characterize heterogeneous O&M experience.
  - **C2:** Identify implicitly correlated metrics from occurring anomalies.
  - **C3:** Adaptively explore diagnosis paths and generate actionable reports.

DBAIOps System Architecture

Wei Zhou et al., DBAIOps: A Reasoning LLM-Enhanced Database Operation and Maintenance System using Knowledge Graphs, VLDB 2026.



From offline experience integration to online graph-augmented diagnosis

Input

Anomaly-related Metrics, DB Instance, Experience Graph, Anomaly Models

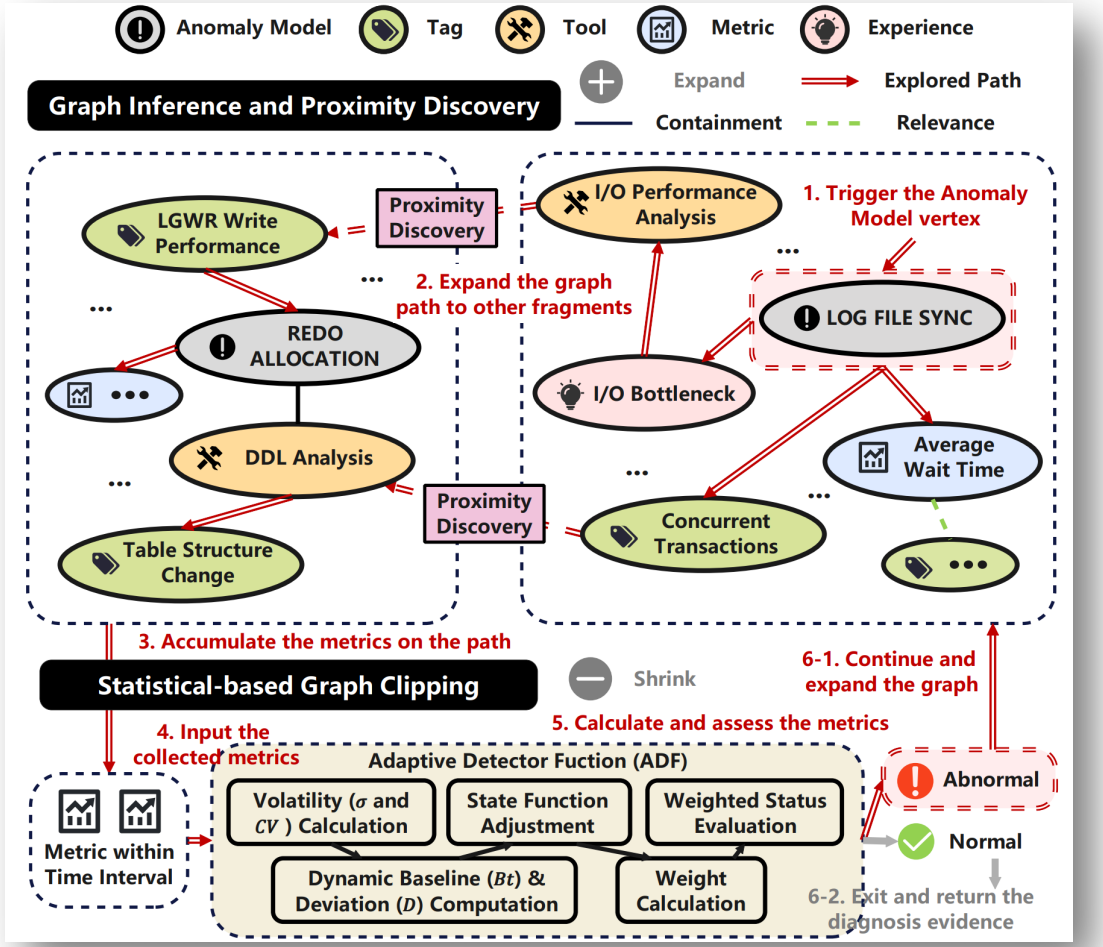
Output

Comprehensive Diagnosis Report
(Root Cause Analysis + Practical Solutions)

Key Pipeline Steps

1. **Experience Graph Construction:** O&M Documents / Cases / Scripts → Heterogeneous Experience Graph
2. **Anomaly Model Initialization:** Metric Hierarchy + Correlation Rules → Reusable Anomaly Model Vertices
3. **Runtime Anomaly Processing:** Target DB Metrics → Triggered Anomaly Models + Alert Descriptions
4. **Graph-based Experience Retrieval:** Triggered Vertices → Two-Stage Graph Exploration → Relevant Diagnosis Paths
5. **Root Cause Analysis:** Diagnosis Paths + Metric Evidence → Reasoning LLM → Diagnosis Report

DBAIOps: Graph-Grounded Diagnosis Memory for Database O&M Agents



Core Techniques

1. Correlation-aware Anomaly Perception

Metrics are processed by Anomaly Models to detect both explicit alerts and implicit metric correlations.

2. ExperienceGraph-based Semantic Grounding

O&M documents, scripts, historical cases, metrics, and tools are organized as heterogeneous graph paths.

3. Adaptive Evidence Exploration

Triggered anomaly vertices guide graph traversal; statistical clipping removes irrelevant paths.

4. Reasoning LLM for Actionable Diagnosis

The LLM reasons over retrieved graph paths and metric evidence to produce root causes and solutions.

Core Insight **Long-lived DBA experience becomes graph-grounded, reusable diagnosis memory**

DBAIOps as an L3-style Data Agent: Key Takeaway

1. Memory Reuse

- Experience graph stores DBA-style diagnosis paths rather than isolated document chunks.
- Anomaly models encode reusable multi-metric patterns of specific issues.

2. Skill Transfer

- Tool vertices link executable diagnosis scripts to experience paths.
- Graph paths function as reusable O&M playbooks for recurring anomalies.

3. State Management

- Triggered anomaly vertices, expanded graph, collected metrics, and diagnosis report.
- ADF decides whether graph exploration to new vertices continues or terminates.

4. Deterministic Verification

- Metric evidence and graph clipping constrain LLM reasoning.
- Reports target both DBAs and common users.

From fragmented O&M experience to reusable DBA-style diagnosis paths

Key Distinction from Traditional O&M Diagnosis

Executable experience graph
not scattered rules or chunks

Implicit correlation-aware detection
captures hidden metric-to-anomaly relations

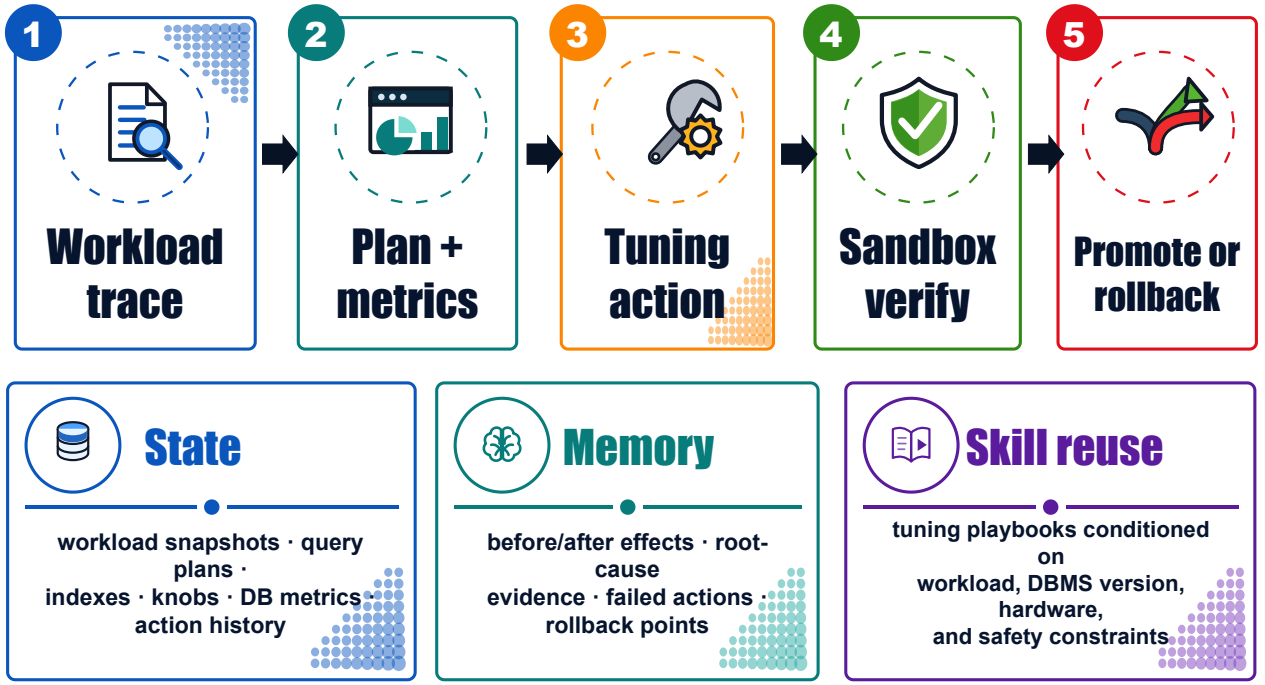
Graph-guided root cause reasoning
grounds LLM diagnosis in evidence paths

Takeaway: Memory Is a Safety Contract for System-Changing Actions

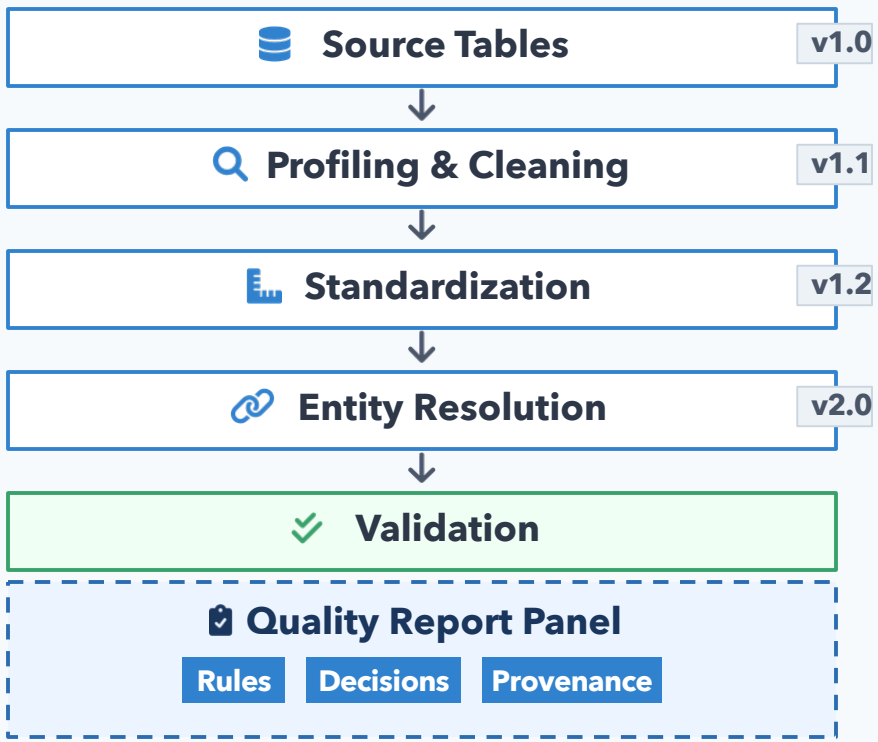
- In Data Management, ***the challenge is not deciding what to do next***, but ***deciding what can be trusted to run again***.

- Memory turns one-off interventions into reusable operational knowledge by preserving the conditions, risks, and observed outcomes behind each decision.

- **Design Rule:** Reliable automation requires remembered effects, not just recorded actions.



Scenario Lens – Data Preparation



- DP must remember **dataset versions, transformations, schema/entity decisions, and quality rules**.
- What matters is not only the final clean table, but also **how it was produced**.
- The reusable unit is a **cleaning / integration recipe**, not a one-off fix.

L1/L2 **CleanAgent (2025)**

Narrow but practical automation for data standardization.

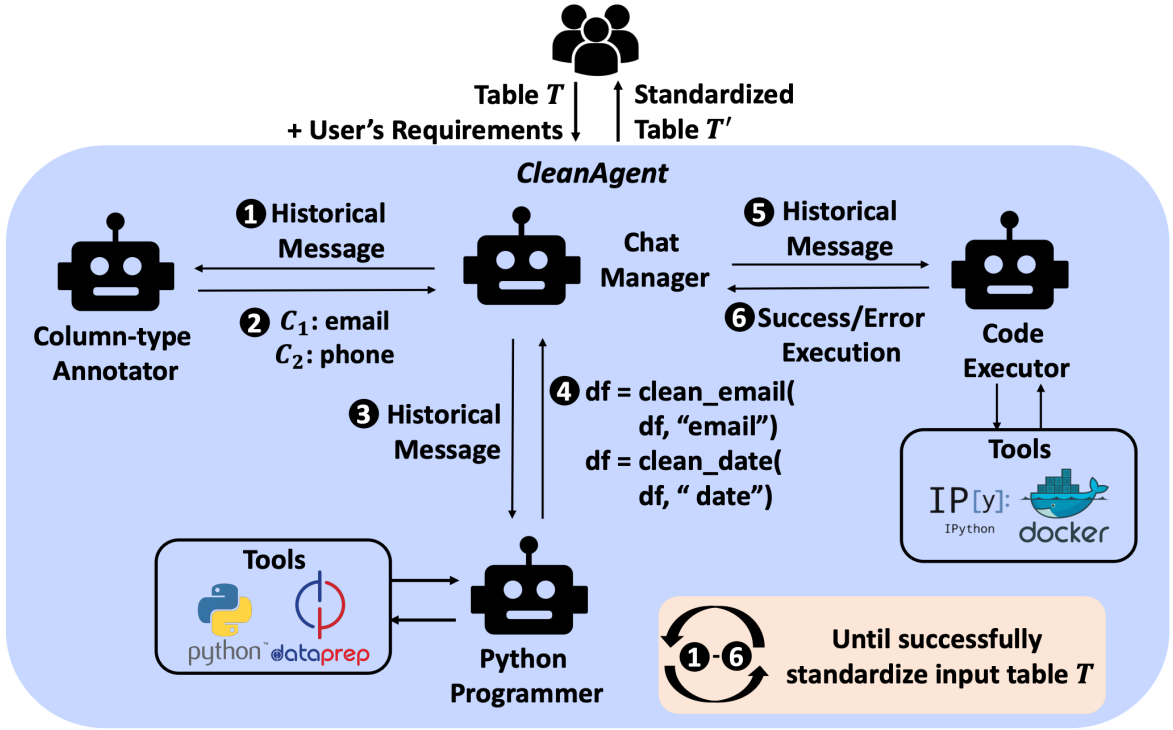
L2-style **AutoPrep (2025)**

Multi-agent **Planner** → **Programmer** → **Executor** for question-aware preparation.

DP memory must be provenance-aware and reproducibility-aware

Data Cleaning: CleanAgent (VLDB Workshop 2025)

- **Problem:** Automatic data standardization – users express requirement once, system handles the rest
- **Core Method:** CleanAgent can interact with raw data and execution environment (like Python or Docker), receiving feedback to refine generated code during data standardization
- Introduce a memory module to maintain the historical conversation context



Memory Lens

State: Current dataframe/column, target type, generated API call, cleaned output

Memory: Local operation context within agent conversation

Skill: External library functions (not newly distilled procedural memory)

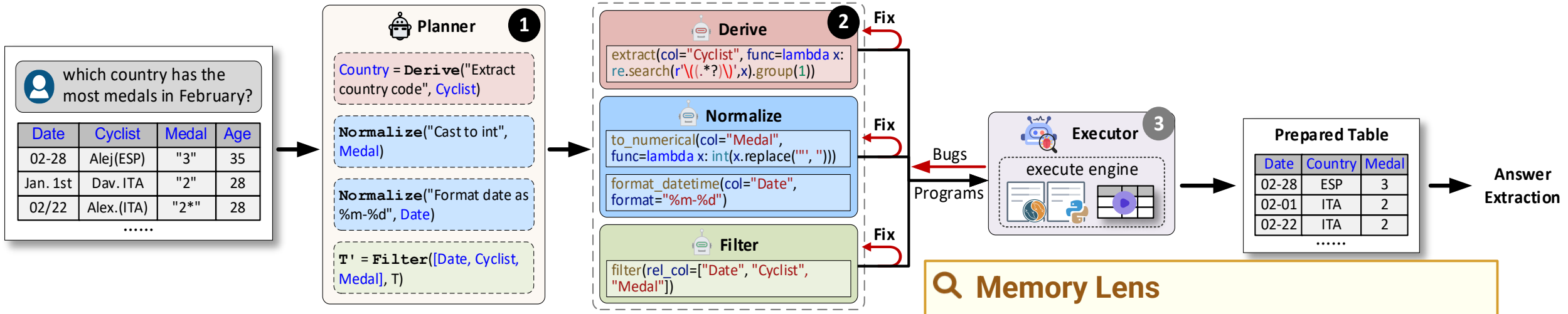
Good local verifiability via validation + change report; weak long-lived memory

Qi D, Miao Z, Wang J. CleanAgent: Automating data standardization with llm-based agents[C]. VLDB 2025 Workshop: Data Driven AI.

AutoPrep Framework (VLDB 2025)

Main Idea: decompose the complex problem into smaller, manageable sub-tasks

- Inspired by DBMS, AutoPrep separates high-level (logical) data prep(DP) operations from the concrete codes (physical) for execution
  - **Phase 1: Task Planning.** Design a Planner agent to generate all data prep tasks and plan them in sequence, i.e., a chain of logical operations (e.g., Derive)
  - **Phase 2: Task Programming & Execution.** Assign the Programmer agent to implement each logical operation and an Executor agent for execution with iterative debugging



Memory Lens

State: NL question, logical plan, code, outputs

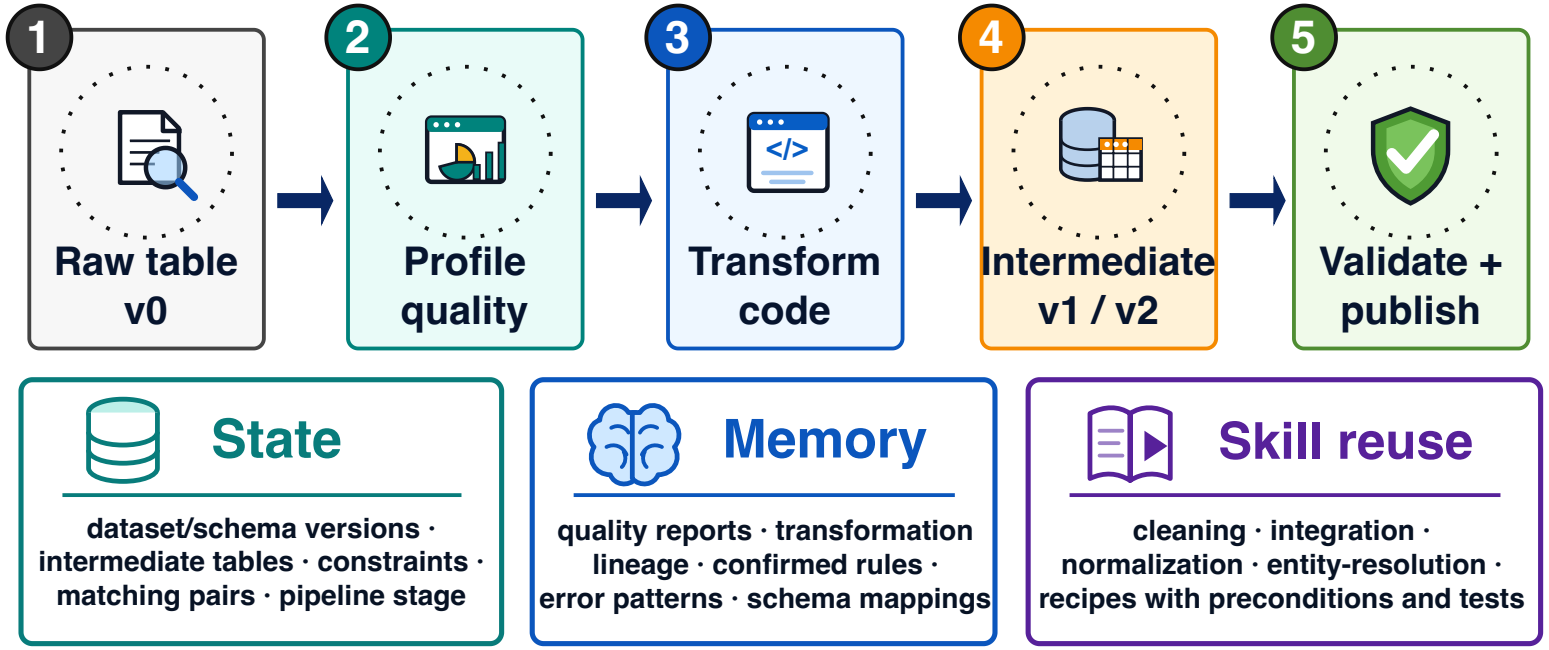
Memory: Run-level intermediate prep state

DP as workflow: plan → code → exec. Persistent lineage memory remains missing.

Takeaway: Memory Is Versioned Semantic Lineage

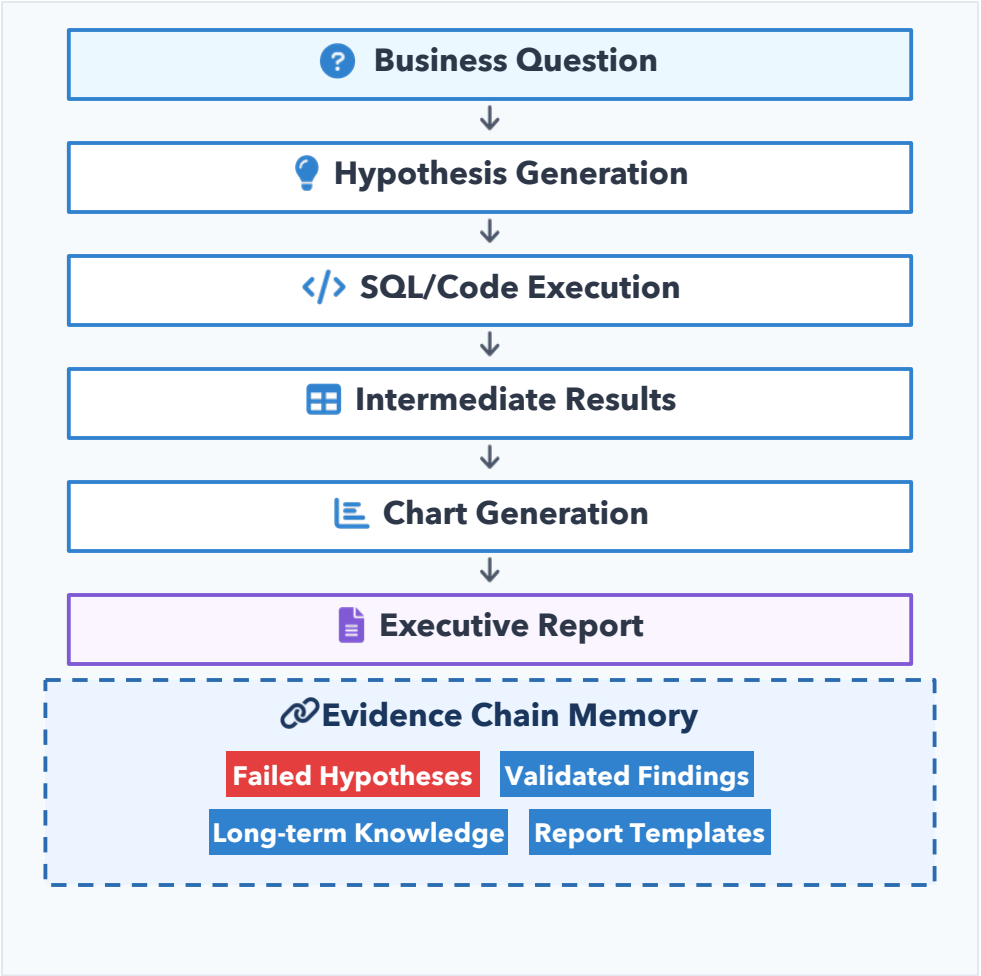
- In Data Preparation, the final clean table is not enough. ***A reliable data agent must remember how each transformation changed the data semantics***, which rules were confirmed, and which versions remain reproducible.

- **Design Rule**: No verified lineage, no promotion to downstream analytics.



Scenario Lens – Data Analysis

- **Root-cause analysis and executive reporting need evidence-aware reasoning memory**



- DA must remember **hypotheses, SQL/code, intermediate results, charts, and report artifacts**.
- The key quality test is whether every insight is backed by an **evidence trail**.
- The reusable unit is an **analysis / visualization / reporting template**.

L2 Alpha-SQL (ICML 2025)

MCTS over partial SQL states

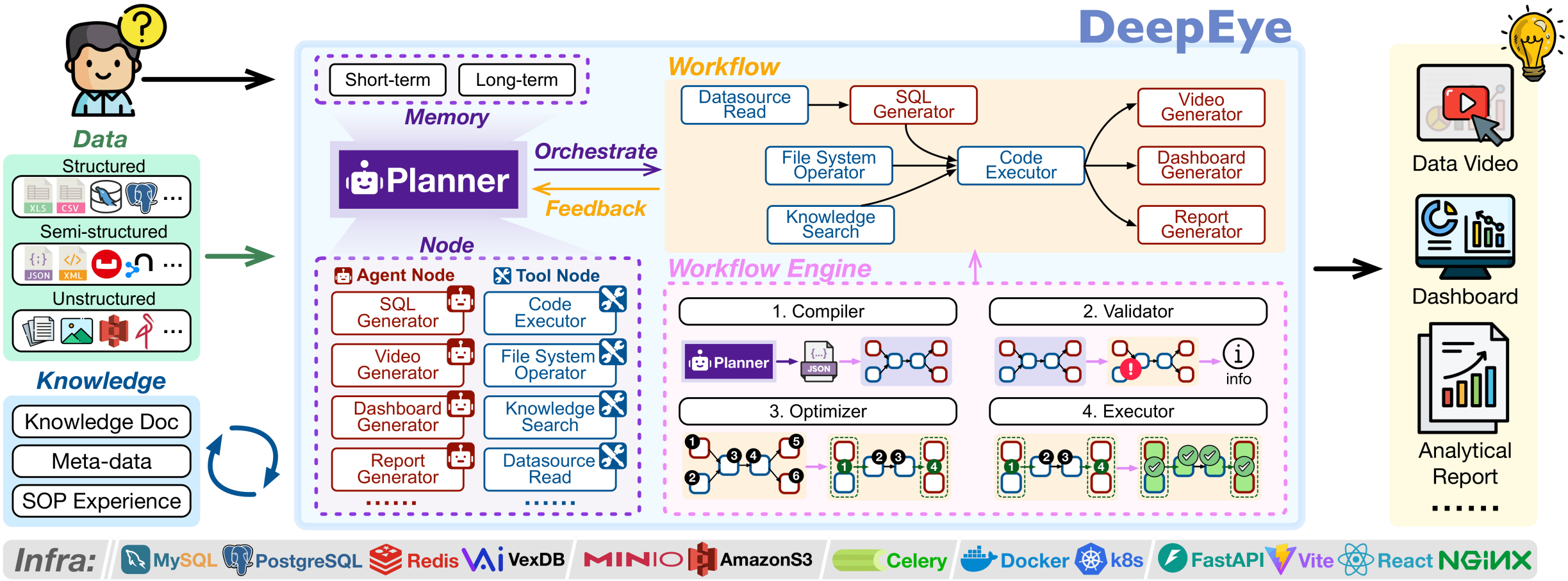
L3-style DeepEye (SIGMOD 2026)

SDLC + workflow engine + memory

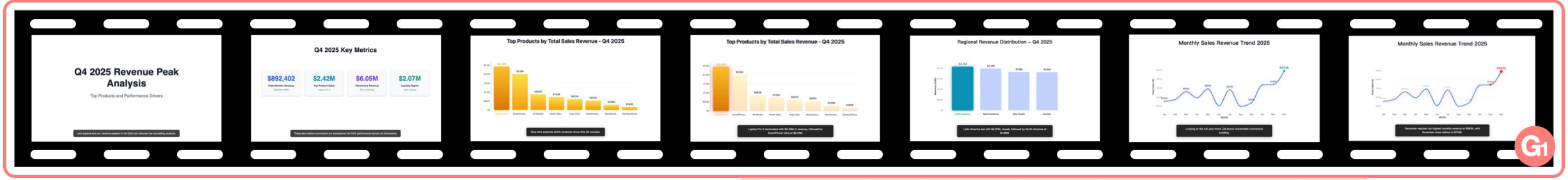
DA memory must be evidence-aware and reasoning-aware

No evidence chain, no trustworthy insight

The System Architecture of DeepEye



DeepEye: A Steerable Self-driving Data Agent System. **SIGMOD 2026**. https://arxiv.org/abs/2603.28889
https://github.com/HKUSTDial/DeepEye



Data Video

Knowledge Bases

Back to Chat

Manage your documents and knowledge

Search

+ New Knowledge Base

Knowledge Base	Description	Last Updated	Actions
Metrics Info… ID: 249867e1…	No description	2026/1/15	Open
Stock Metrics ID: 573498w…	No description	2026/1/15	Open
Financial Met… ID: 7936f53g…	No description	2026/1/15	Open
Trading Policy ID: 7286765…	No description	2026/1/15	Open
SOP Experie… ID: 86528f5…	No description	2026/1/15	Open
Meta-data ID: 683c79w1…	No description	2026/1/15	

A

DATA SOURCES

- Tariff.json
- Products Database
- updated_price.csv
- monthly_top_saler.xlsx
- Sales Database

B

C

@Financial Metrics

@Meta-data

@Sales Database

Help me analyze the 2025 global sales performance.

AI DeepEye

workflow_agent Done

INPUT

```
{'goal': 'Find the global sales performance of 2025.'}
```

create_plan Done

run_workfl... Done

OUTPUT

```
The workflow has been generated and run successfully.
```

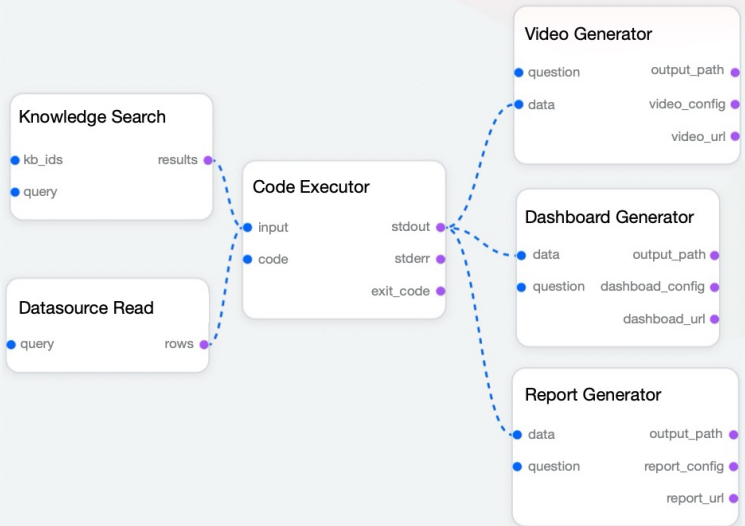
D

Message DeepEye...

Files | Workflow | Video Preview | Dashboard Preview | Report Preview | + | Split

Workflow · success sales_performance_analysis.json Save Upload Export Run

E



Inspector

Datasource Read

Read rows from a datasource.

datasource_id optional

F ba7400-e587-4e7b-a734-21da8880f8

query optional

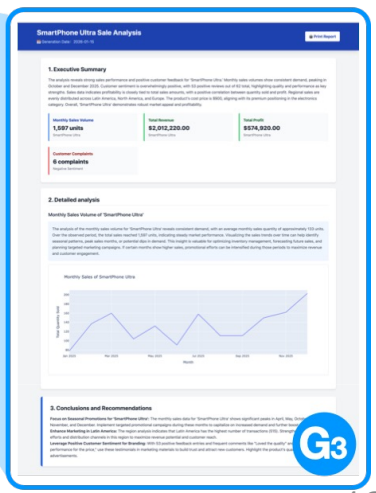
SELECT ProductID, COUNT(*) as count FROM

RUN STATUS

Status success

Started 16:38:27

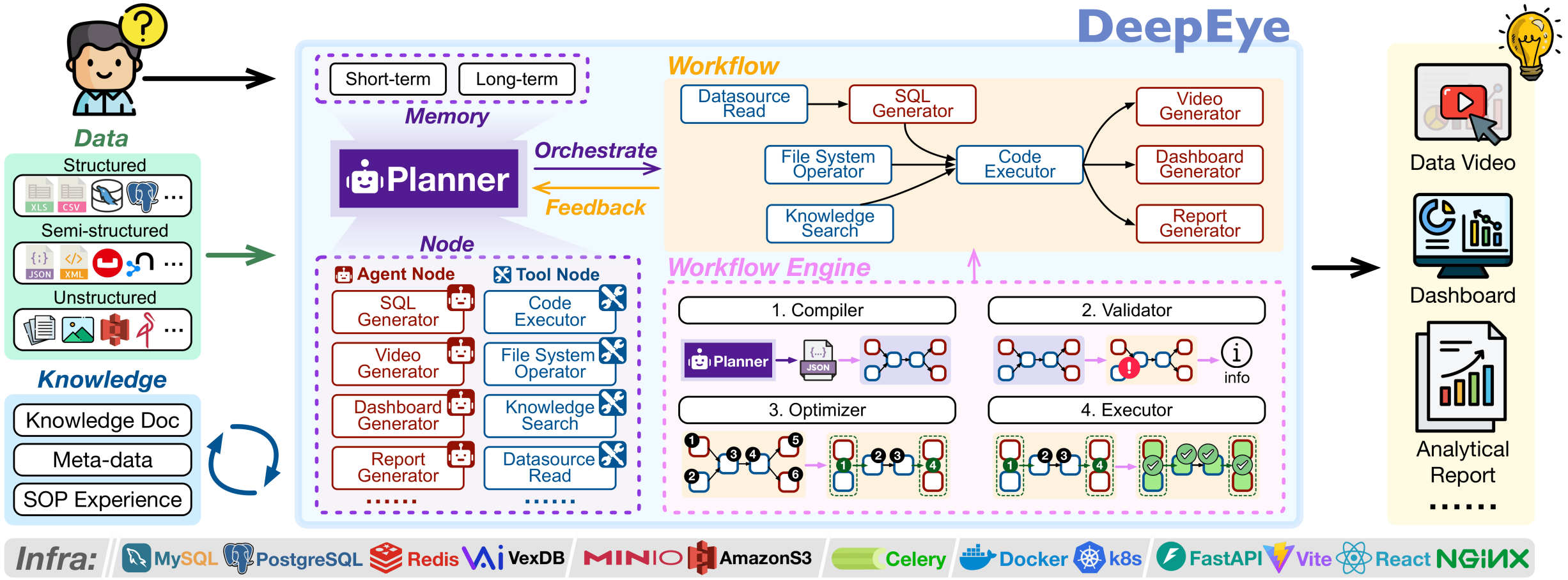
Dashboard



Analytical Report

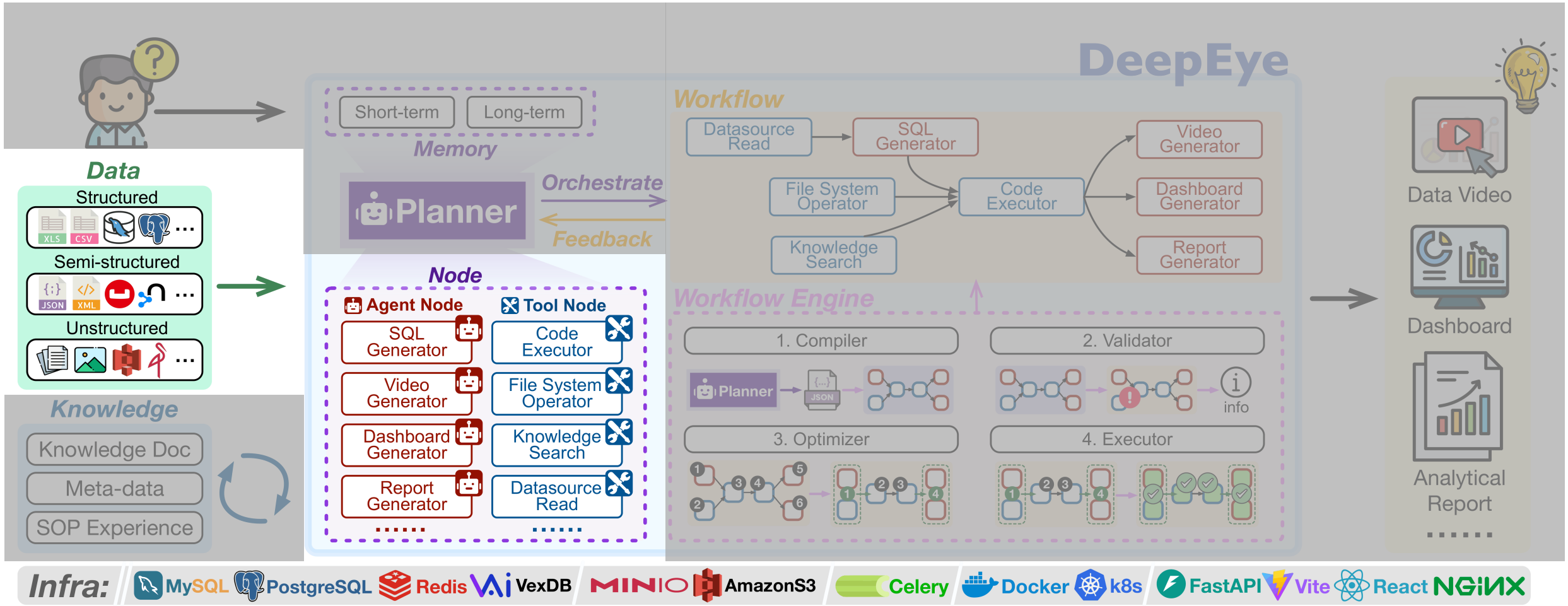
https://github.com/HKUSTDial/DeepEye https://deepeye.hk/

The System Architecture of DeepEye



DeepEye: A Steerable Self-driving Data Agent System. **SIGMOD 2026**. https://arxiv.org/abs/2603.28889
https://github.com/HKUSTDial/DeepEye

The System Architecture of DeepEye



1/3

Unified Node Protocol: Turn Every Analytics Capability into Building Blocks

Databases, files, knowledge bases, and code tools... are all first converted into **standard nodes!**

Database

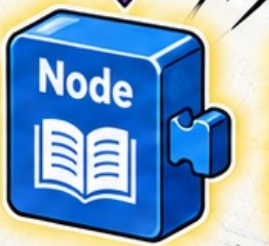
CSV / File

PDF / Report

Knowledge

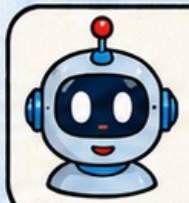
Python / Tool

Dashboard



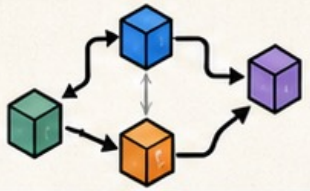
DeepEye = Orchestrating Multimodal Data with a Unified Interface

Question



Planner
Intelligent Planning
Planning & Decomposition

Node DAG
Orchestration & Execution



Multimodal Result
Integrated Output

DeepEye
Unified Node Protocol

HOW TO DO ①: EVERY NODE HAS THE SAME IDENTITY

In DeepEye, every node (Node) uses the same identity specification for decision-making and communication!

Unified standard, any combination, unlimited!

DeepEye Node Identity (Robot Passport)

ID : node_8f3c9a

Type : ToolNode / AgentNode

Version : v1.0.0

Created : 2025-05-18 10:30:00

DEEPEE CERTIFIED

$$N = \langle D, I, O, C, \Phi \rangle$$

D	I	O	C	Φ
Description	Input Port	Output Port	Configuration	Execution Logic
Basic information and capabllities of the node.	Definition of ports that accept external input.	Definition of ports that produce external output.	Parameters, options, and available resources.	Core logic and execution methods of the node.

Any node can be uniquely recognized, connected, configured, and executed!

1 ToolNode : Deterministic operation, e.g., DataConnector / CodeExecutor; takes input, produces output, no ambiguity.

2 AgentNode : LLM reasoning node; has private context W_{local}, responsible for complex reasoning.

Context Isolation : Only exposes I/O, hides internal reasoning noise.

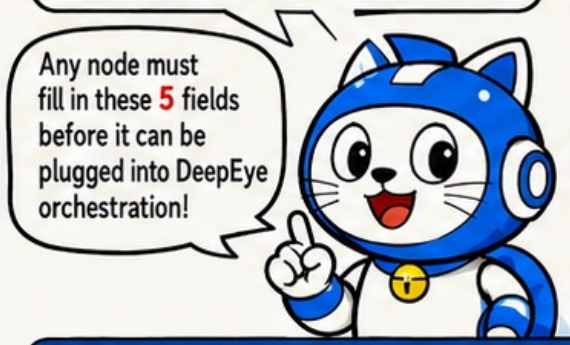
In the DeepEye Unified Node Protocol, how is an "SQL Generation Node" implemented under the protocol?

Example: SQLGeneratorNode = generate candidate SQL from the question + schema + knowledge base

Unified Protocol Template Card

N = <D, I, O, C, Φ>

D	Description	What the node does
I	Input Port	What inputs it needs
O	Output Port	What outputs it produces
C	Configuration	Which parameters are configurable
Φ	Execution Logic	How the task is executed

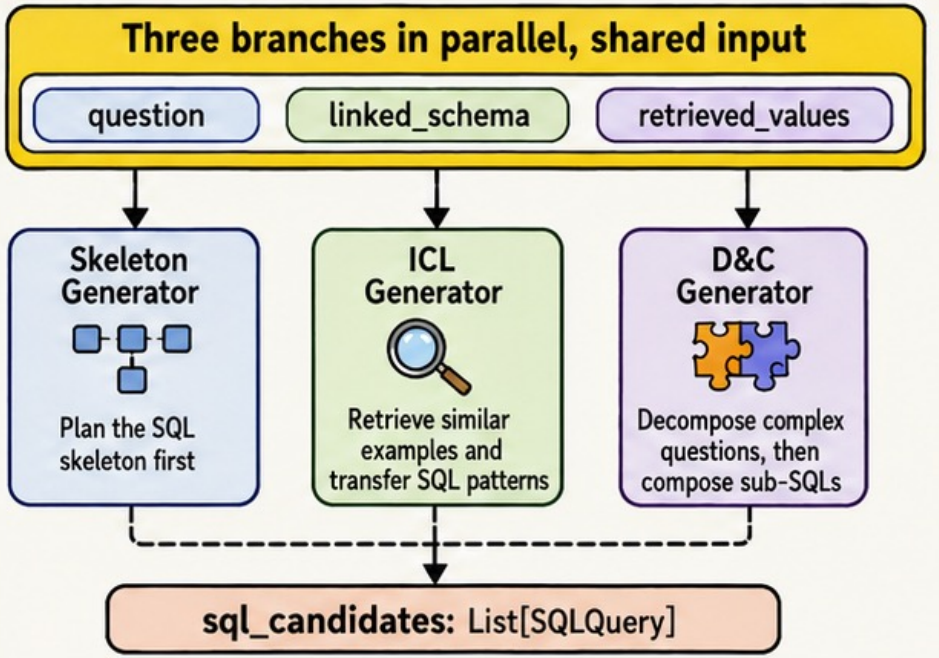


Node Implementation Blueprint

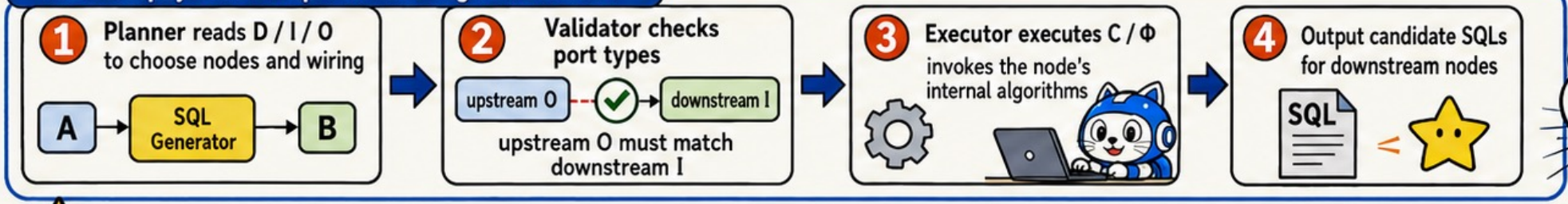
Node Name: SQLGeneratorNode

D Description	Generate candidate SQL from the natural-language question, linked schema, and retrieved knowledge-base evidence.
I Input Port	• question: string • linked_schema: SchemaGraph • retrieved_values: Dict[column, values] • knowledge_passages: List[Text] • constraints: JSON
O Output Port	• sql_candidates: List[SQLQuery] • reasoning_summary: Text • used_schema: SchemaGraph
C Configuration	• model = Qwen/DeepSeek • generation_mode = Skeleton / ICL / D&C • max_candidates; temperature • dialect = SQLite / PostgreSQL
Φ Execution Logic	1. plan_sql_components() 2. generate_skeleton() / retrieve_examples() / decompose_question() 3. fill_schema_and_values() 4. emit_candidate_sqls()

Internal Sub-DAG: Three-Branch Parallel Generation



How DeepEye uses the protocol during orchestration



Unified interface, heterogeneous internals: Planner connects I/O, Validator checks types, Executor executes C/Φ.

HOW TO DO ②

HOW TO DO ②: EXAMPLE — 2025 GLOBAL SALES ANALYSIS

3/3

Let's analyze by combining data from @Sales Database and @Financial Metrics!

Workflow DAG (DeepEye Unified Task-Oriented Protocol)

@Sales Database

@Financial Metrics

Datasource Read

Knowledge Search

SQL Generator

Code Executor

Data Video

Dashboard

Analytical Report

One protocol, high scalability! Combinable, reusable, and expandable!

DeepEye Workflow Engine

Workflow Engine (Orchestration Pipeline)

1 Compiler → 2 Validator → 3 Optimizer → 4 Executor

DeepEye Gadget Box

Validator: Unsupported data type or mismatch? Reject early!

Graph

Output: Table

Input: String

"ABC"

Optimizer detects: If database reads and knowledge searches are independent, they can run in parallel!

Datasource Read

Knowledge Search

Execute in parallel!

DeepEye Takeaway: Workflow DAG Is Long-Horizon Memory

- DeepEye turns heterogeneous data analysis from short-context interaction into an inspectable workflow artifact. The agent does not only produce an answer; it preserves the executable path that produced the answer.

- **State**: node status · I/O ports · configurations · intermediate artifacts
- **Memory**: workflow DAG · execution logs · provenance · knowledge docs · SOP experience
- **Skill reuse**: reusable nodes and verified workflows (SOPs) that can be adapted to future tasks
- **Verification**: compiler, validator, optimizer, and executor make the workflow inspectable and controllable

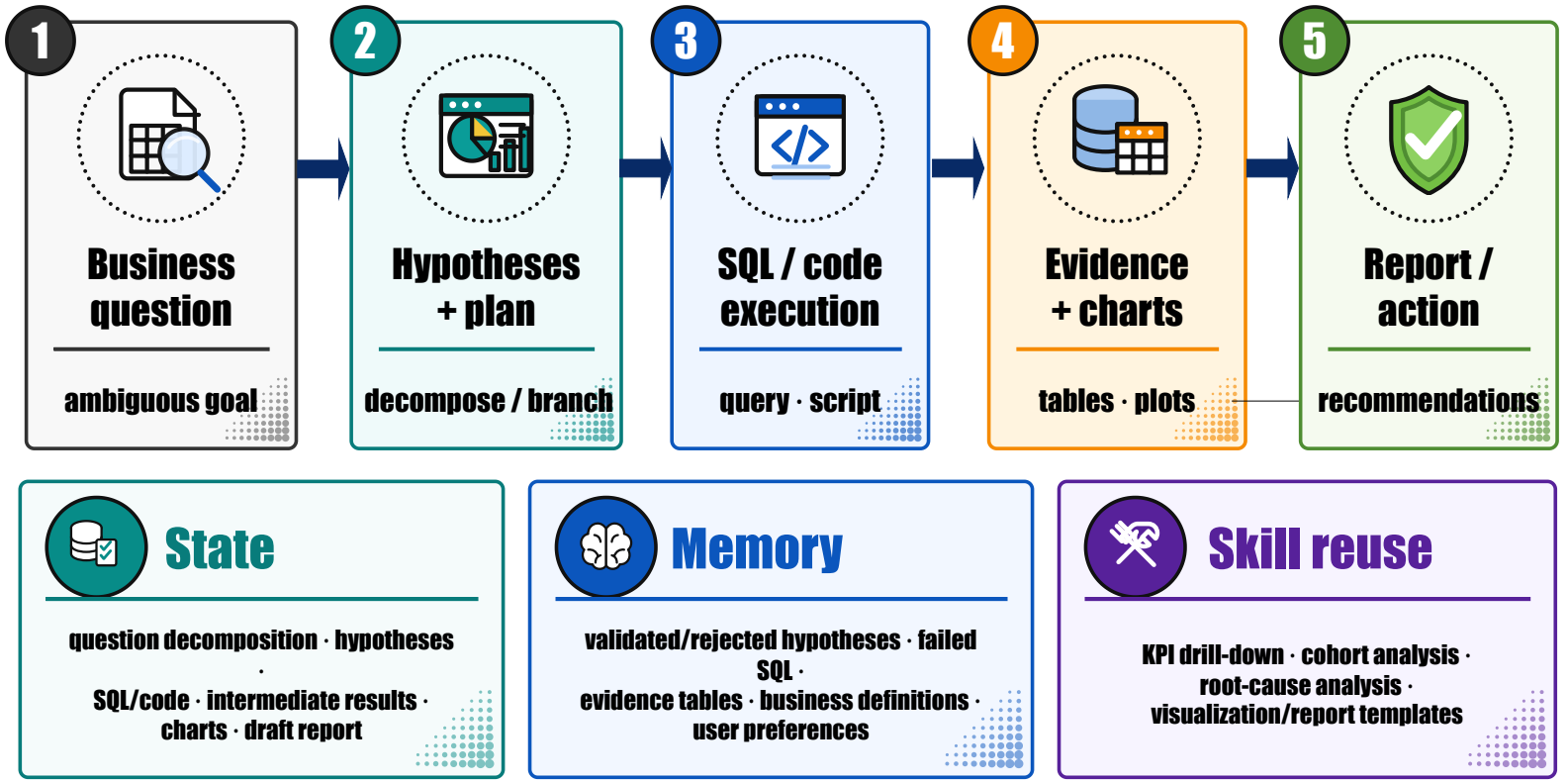
- **Design Rule**: If an analysis cannot be inspected as a workflow artifact, it cannot be reliably reused.

DeepEye: A Steerable Self-driving Data Agent System. **SIGMOD 2026**. https://arxiv.org/abs/2603.28889
https://github.com/HKUSTDial/DeepEye

Takeaway: No Evidence Chain, No Trustworthy Insight

- In Data Analysis, memory must bind every claim to the reasoning path that produced it: question, hypothesis, SQL/code, intermediate results, visual evidence, and report text.

- **Design Rule**: Every insight should be traceable, replayable, and verifiable.



Takeaway: What is common, what is different?

- **Common pattern: execution traces can be distilled into reusable procedural memory.**
- **Difference: each scenario has different state semantics and verification boundaries.**

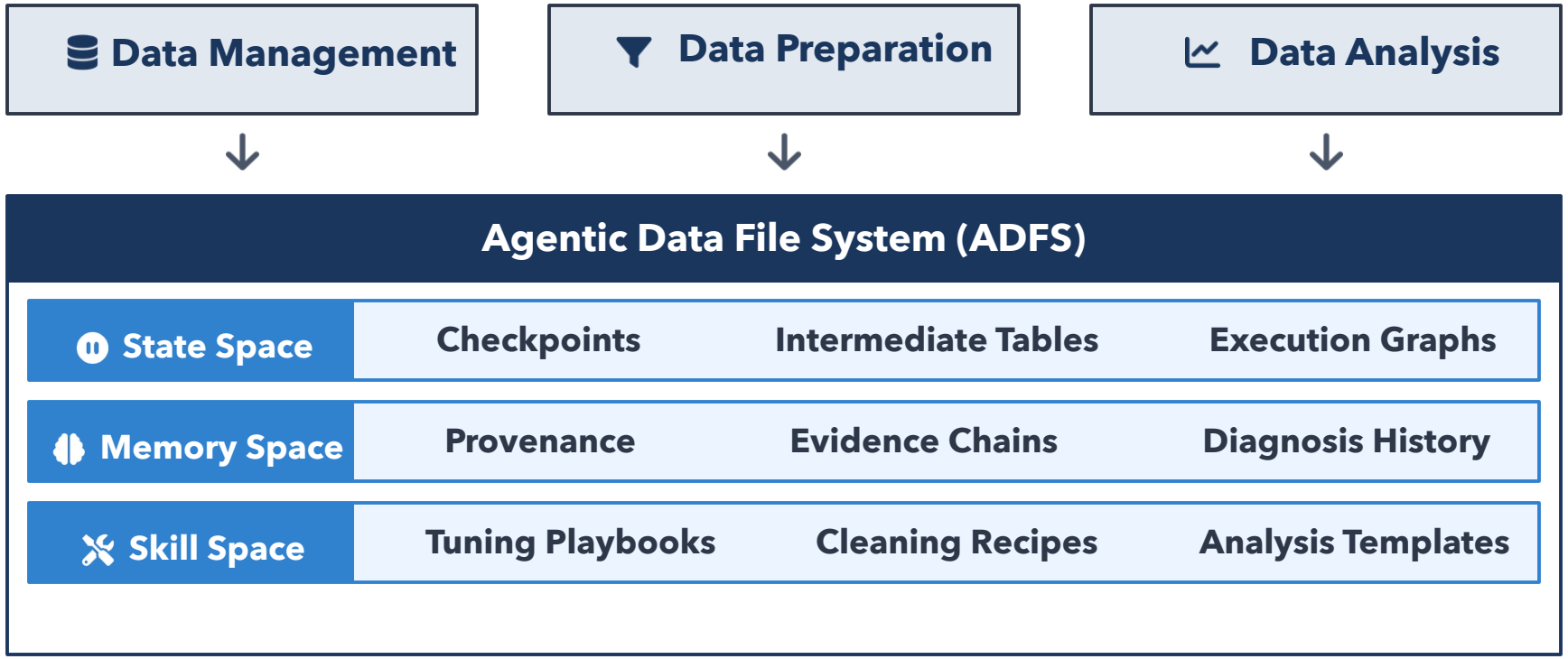
Dimension	DM	DP	DA
Memory object	System effects	Data transformations	Reasoning evidence
State granularity	DB / workload / plan / config	Record / table / pipeline / version	Question / hypothesis / artifact / insight
Skill type	Tuning + diagnosis playbook	Cleaning + integration recipe	Analysis + visualization + report template
Verifier	Latency · throughput · cost · safety	Constraints · quality · downstream effect	SQL correctness · stats validity · evidence coverage
Main risk	Unsafe production side effect	Silent semantic corruption	Unsupported insight / hallucination
Reuse boundary	DBMS · workload · hardware	Schema · domain · data distribution	Business metric · goal · user context

DM remembers system effects; DP remembers data transformations;
DA remembers reasoning evidence.

A Common Substrate – Agentic Data File System

Unifying State, Memory, and Skill across DM, DP, and DA

Instead of each agent building its own isolated memory silo, the next generation of data agents requires a unified, persistent file system to manage long-horizon execution.



ADFS-like Foundation: The foundational infrastructure for agentic state, memory, and skill reuse.

From Memory Reuse to Agentic Evolution

The ultimate goal: Meta-agents that learn and evolve

Long-horizon reliability is not just about executing a single task perfectly. It is about building a system that accumulates experience, distills it into skills, and evolves over time.

1. Discovery

Agents explore the state space, generate hypotheses, and execute actions. Successes and failures are recorded in the **Memory Space**.

2. Distillation

Meta-agents analyze the memory space to identify successful patterns, abstracting them from specific contexts into generalized procedures.

3. Reuse

Distilled procedures are stored in the **Skill Space** (e.g., ADFS) and retrieved by agents to solve similar future tasks efficiently.

4. Evolution

The system continuously refines its skills, deprecates outdated ones, and adapts to new data distributions and workloads autonomously.

Tutorial Takeaway

To build reliable Data Agents, we must move beyond short-context LLM calls.
We need State to resume and branch execution, Memory to preserve provenance and evidence, Verification to control risk, and Skills to turn successful traces into reusable workflows.
The reusable unit is not a prompt or an answer, but a verified workflow artifact.

Outline

- Part I: Data Agents: Motivation, Definition, Autonomy Levels, and Core Challenges
- **Part II: Towards Autonomous Data Agents: Key Challenges and Current Practices**
  - Data-aware Workflow Orchestration & Cost-aware Execution
  - Long-horizon Agentic State, Memory & Skill Reuse
  - **Semantic Grounding over Heterogeneous & Multimodal Data**
- Part III: Research Opportunities and Open Challenges

Core Data Agent Challenges and Tutorial Roadmap

Challenge \ Level	L1 Assistance	L2 Partial Autonomy	L3 Conditional Autonomy	L4 High Autonomy	L5 Full Autonomy
	Prompt-driven Data Assistant	Human-guided Workflow Executor	Agentic Workflow Orchestrator	Proactive Data Agent	Self-evolving Autonomous Data Agent
C1. Data-aware Workflow Orchestration & Cost-aware Execution	Human-designed workflow; agent suggests.	Execute human-designed pipelines with local feedback.	Auto-compose and optimize workflows / DAGs.	Proactively discover tasks and plan long-horizon execution.	Invent new operators, tools, and workflow paradigms.
C2. Long-horizon Agentic State,	Stateless; no	Short-term task	Workflow state + provenance +	Long-lived cross-	Self-evolving
Semantic Grounding over Heterogeneous & Multimodal Data (40 min)					
C3. Semantic Grounding over Heterogeneous & Multimodal Data	Prompt / few-shot / RAG-based grounding.	Environment-aware schema, entity, and evidence grounding.	Semantic catalog + metadata + multimodal evidence alignment.	Continuous semantic maintenance over evolving data lakes.	Create new semantic abstraction and representations.

Progression Assist & Suggest ≫ Execute & Improve ≫ Orchestrate & Optimize ≫ Proactively Act ≫ Invent & Evolve

From Database Abstractions to Semantic Operators

Database researchers turn messy application logic into reusable operators.
Semantic operators are the next abstraction layer for AI-native data processing.

Application Area	Classic DB Abstraction	Semantic-Operator Opportunity
Spatial databases	Range query, nearest neighbor, spatial join	Semantic similarity search and semantic join over places, images, or scenes
Sensor / stream systems	Windows, continuous queries, event detection	Semantic event detection and grouping of abnormal patterns in logs or telemetry
Text / IR systems	Keyword search, ranking, inverted index, top-k	Semantic filter and top-k ranking, e.g. papers with surprising claims
Data integration	Schema matching, entity recolution, record linkage	Semantic join across heterogeneous sources, e.g. records referring to the same company
OLAP / analytics	Group-by, roll-up, drill-down, aggregation	Semantic group-by and semantic aggregate, e.g. group tickets by root cause
Uncertain / provenance-aware data	Confidence, lineage, why-provenance	Semantic outputs with confidence scores, evidence, and provenance

Takeaway | **DB history:** abstract application logic into operators. **Data-agent future:** abstract AI reasoning into semantic operators.

Symbolic operators (abstractions)

Traditional DB operators whose logic is specified by explicit symbols: predicates, keys, expressions, rules

Semantic operators

AI-native operators whose logic is specified by natural language intent and interpreted by models/tools

Symbolic Operators vs Semantic Operators: Join as an example

Concrete enterprise example: joining a customer table with contract (table/documents)

Customer Table

Customer ID	Customer Name	Region
C1	Alibaba Cloud	China
C2	Tencent Cloud	China
C3	ByteDance	Singapore
C4	Baidu AI	China

join customers with contracts

join

join

Case 1: structured table

Contract Table

Contract ID	Customer Name	Service
D1	Alibaba Cloud	Cloud hosting
D2	Tencent Cloud	Storage
D3	ByteDance	Advertising
D4	Alibaba Group	Procurement

Case 2: raw documents

Contract Documents

- This agreement is signed with 阿里云 for cloud hosting...
- The contract is entered into by Tencent cloud division ...
- Agreement with ByteDance Ltd. for advertising services ...
- This contract is signed by Alibaba Group for procurement ...

What is a semantic operator?

```
sem_xxx(input data, intent/spec, context, model/tool, contract)
                → data product + confidence + provenance
```

sem_filter(
input data = customer_reviews,
intent/spec = "keep reviews complaining about delivery delay",
context = product category, recent logistics policy, examples of delay complaints,
model/tool = small classifier + LLM verifier,
contract = precision ≥ 0.9, output review_id + rationale + evidence span
)

Output: filtered_reviews + confidence + provenance

sem_count | sem_map | sem_filter | sem_extract | sem_join | sem_groupby | sem_summarize

Why semantic operators are important to data agents?

- *Inside databases*
- *Outside databases*

Symbolic Operators vs Semantic Operators: Join as an example

Traditional Join

Symbolic predicate over already-clean values

JOIN(R, S) where R.a = S.b

Example SQL

```
FROM Customers C
JOIN Contracts D
  ON C.name = D.customer_name
```

Works when both sides use exact keys or normalized values. Fails for aliases, translations, paraphrases, and ambiguity.

Misses: 阿里云 → Alibaba Cloud
Confuses: Alibaba Group ≠ Alibaba Cloud

from value equality to entity identity

Semantic Join

Semantic-based predicate grounded by evidence

JOIN(R, S) where sem_match(R.e, S.text)

Conceptual SQL

```
FROM Customers C
SEMANTIC JOIN Contracts D
  ON sem_match(C.name, D.text)
```

Joins when the document refers to the same real-world entity, even if the surface form differs.

Outputs: joined tuple + evidence span + confidence + provenance + constraint checks

Alibaba Cloud -> D1 evidence: 阿里云 confidence: 0.96

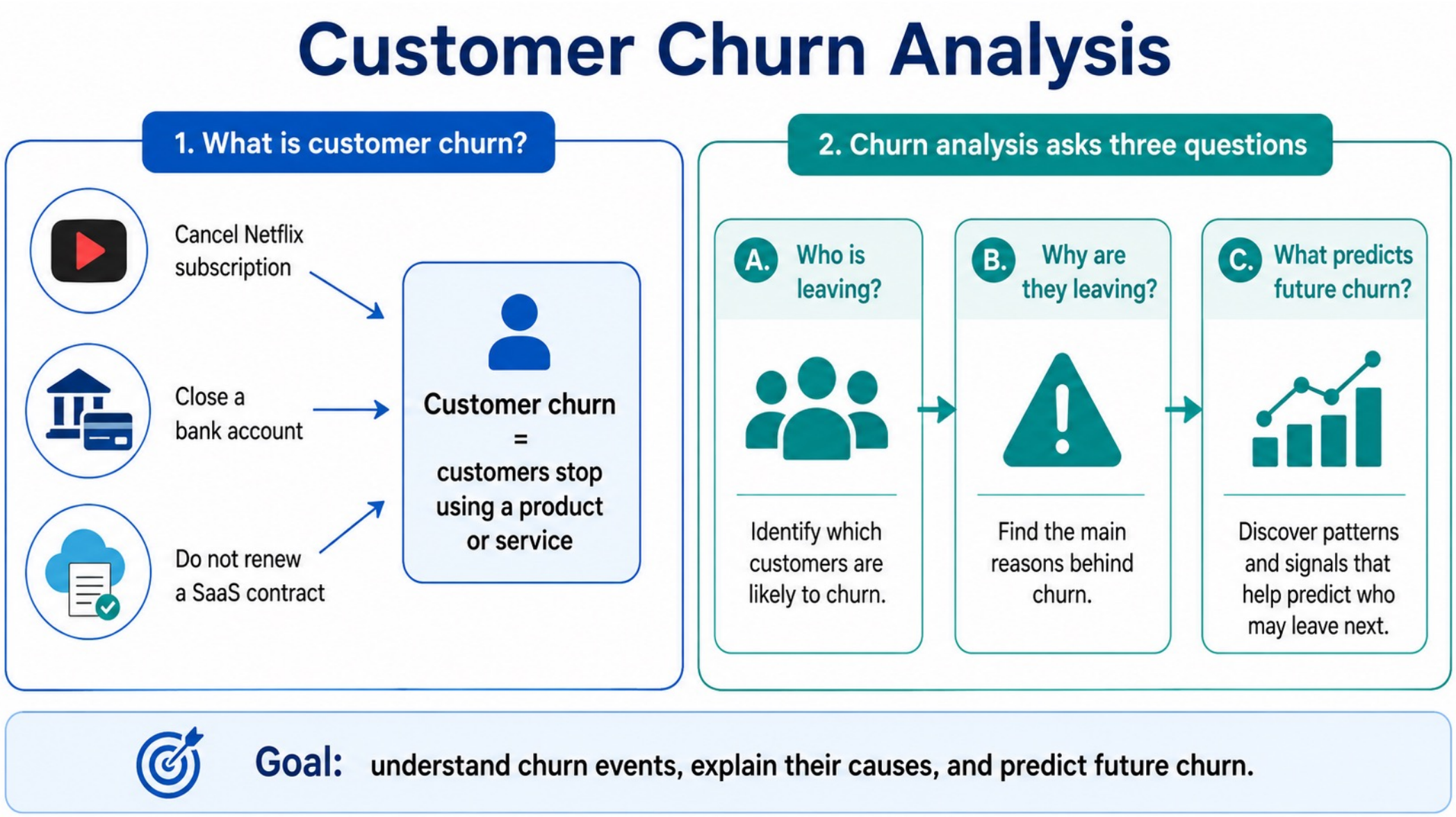
Tencent Cloud -> D2 evidence: Tencent cloud division confidence: 0.89

Alibaba Group -> D4 rejected: related but different entity

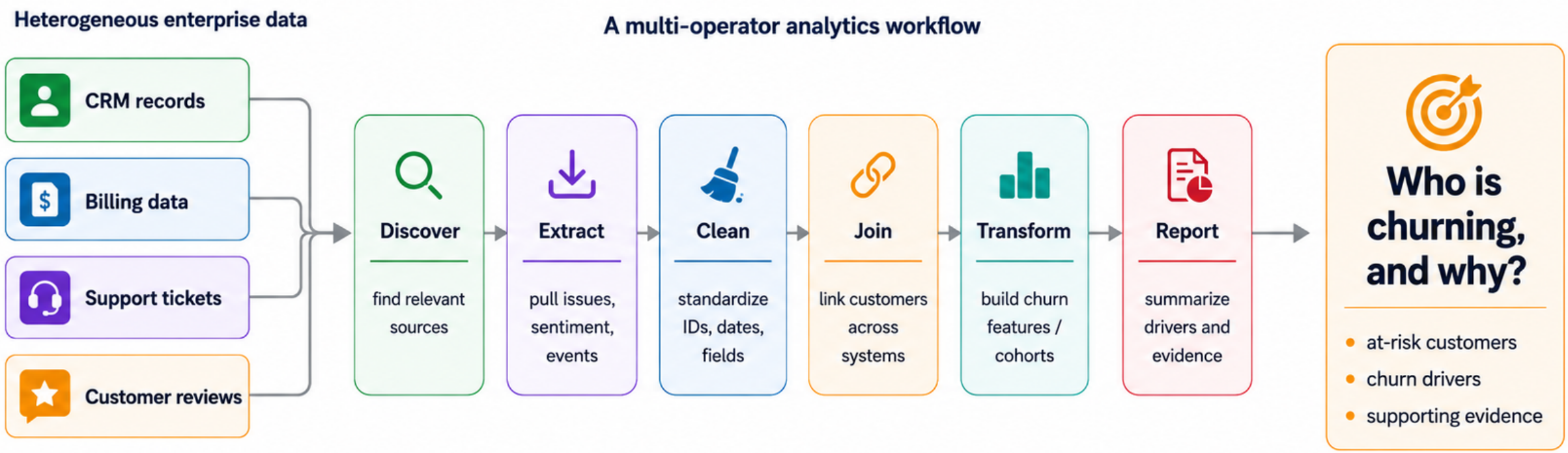
Why semantic operators are important to data agents?

- *Inside databases*
- *Outside databases*

Running example: analyze customer churn



Running example: analyze customer churn



Semantic operators have been widely studied by the database community

System family	Core idea	Main goal
LOTUS	Declarative semantic operators: filter, map, join, aggregate, top-k, transform, extract, ...	Operator algebra + optimization + statistical accuracy guarantees.
Palimpzest	Declarative AI workloads over unstructured data.	Cost / quality / latency optimization across models and prompts.
DocETL	Agentic rewrites for document-processing pipelines.	Rewrite rules, validation prompts, and multi-objective plan search.
Sema / SEMA-SQL	Semantic operators inside SQL/DB runtime.	Optimizer/runtimes for batching, reordering, UDF rewriting, AQE.
SemBench	Benchmark for semantic query-processing engines.	Scenarios × modalities × operators, beyond task-level accuracy.
VectraFlow	Continuous semantic operators over event/text streams.	Streaming semantics, windows, group-by, joins, long-horizon state.

Core Data Agent Challenges and Tutorial Roadmap

Challenge \ Level	L1 Assistance	L2 Partial Autonomy	L3 Conditional Autonomy	L4 High Autonomy	L5 Full Autonomy
	Prompt-driven Data Assistant	Human-guided Workflow Executor	Agentic Workflow Orchestrator	Proactive Data Agent	Self-evolving Autonomous Data Agent
C1. Data-aware Workflow Orchestration & Cost-aware Execution	Human-designed workflow; agent suggests.	Execute human-designed pipelines with local feedback.	Auto-compose and optimize workflows / DAGs.	Proactively discover tasks and plan long-horizon execution.	Invent new operators, tools, and workflow paradigms.
C2. Long-horizon Agentic State,	Stateless; no	Short-term task	Workflow state + provenance +	Long-lived cross-	Self-evolving
Semantic Grounding over Heterogeneous & Multimodal Data (40 min)					
C3. Semantic Grounding over Heterogeneous & Multimodal Data	Prompt / few-shot / RAG-based grounding.	Environment-aware schema, entity, and evidence grounding.	Semantic catalog + metadata + multimodal evidence alignment.	Continuous semantic maintenance over evolving data lakes.	Create new semantic abstraction and representations.

Progression Assist & Suggest » Execute & Improve » Orchestrate & Optimize » Proactively Act » Invent & Evolve

Semantic operators: Topic * Autonomy Map

Topic	L1: prompt	L2: execute + feedback	L3: orchestrate	L4: proactive
Transform	Table-GPT, Jellyfish	MegaTran, EVAPORATE, Palimpzest, DocETL	AOP, AgenticData, Data Interpreter	semantic stream / self-triggered transform monitors
Clean	RetClean, LakeFill, LLMClean	AutoPrep, CleanAgent, SketchFill, IterClean	AgenticData / DeepAnalyze as broader workflows	quality drift monitors; continual cleaning agents
Extract	LiteCost, LongRAG, PDFTriage, VisDoM, Docopilot	DataPuzzle, Doctopus, QUEST, MACT	AOP/iDataLake/AgenticData when extraction is part of a plan	continuous extraction and alerting on new corpora
Join / integrate	BATCHER, Jellyfish, LLMCTA	Agent-OM, MILA, COMEM, SEED, FDJ	iDataLake, AgenticData	schema-drift and identity-resolution monitors
Discover / search	ArcheType, AutoDDG, Pneuma	Chorus, LEDD, DataVoyager, DBDescGen	iDataLake, AOP, AgenticData	long-lived catalog agents

Core Data Agent Challenges and Tutorial Roadmap

Challenge \ Level	L1 Assistance	L2 Partial Autonomy	L3 Conditional Autonomy	L4 High Autonomy	L5 Full Autonomy
	Prompt-driven Data Assistant	Human-guided Workflow Executor	Agentic Workflow Orchestrator	Proactive Data Agent	Self-evolving Autonomous Data Agent
C1. Data-aware Workflow Orchestration & Cost-aware Execution	Human-designed workflow; agent suggests.	Execute human-designed pipelines with local feedback.	Auto-compose and optimize workflows / DAGs.	Proactively discover tasks and plan long-horizon execution.	Invent new operators, tools, and workflow paradigms.
C2. Long-horizon Agentic State,	Stateless; no	Short-term task	Workflow state + provenance +	Long-lived cross-	Self-evolving
C3. Semantic Grounding over Heterogeneous & Multimodal Data	Prompt / few-shot / RAG-based grounding.	Environment-aware schema, entity, and evidence grounding.	Semantic catalog + metadata + multimodal evidence alignment.	Continuous semantic maintenance over evolving data lakes.	Create new semantic abstraction and representations.

Semantic Grounding over Heterogeneous & Multimodal Data (40 min)

Progression: Assist & Suggest » Execute & Improve » Orchestrate & Optimize » Proactively Act » Invent & Evolve

Semantic operators: L1

Operator family	Representative systems	Typical L1 behavior
Transform / wrangle	**Table-GPT**, Jellyfish	Serialize tables/records; ask LLM to infer values, schema matches, or transformations.
Clean	**LakeFill**, LLMClean, RetClean	Retrieve tuples or generate constraints/rules; human or downstream system applies.
Integrate / match	BATCHER, Jellyfish, LLMCTA	Prompt or batch prompt for entity/schema/column decisions.
Discover / annotate	ArcheType, AutoDDG, Pneuma, LLMCTA	Dataset summaries, profiles, semantic column types, metadata.
Analyze / visualize	BINDER, DIN-SQL, Prompt4Vis, Step-Text2Vis	Answer/code/spec generation for TableQA, NL2SQL, NL2VIS.
Extract from docs	**LiteCOST**, LongRAG, PDFTriage, VisDoM, Docopilot	Retrieve/filter/answer over long or visually rich documents.

Semantic Operators: L1

Model tuning

Fine-tune or adapt LLMs for table understanding, table QA, extraction, and transformation.

Slogan

Table-tuning helps today; semantic operators survive tomorrow.

Limitation

Model-specific; hard to keep pace with fast LLM updates.

Data grounding

Retrieve better tuples, examples, documents, or evidence before asking the LLM.

Slogan

LLMs are strong reasoners; good data makes them reliable.

Limitation

Depends on retrieval quality; still mostly one-shot assistance.

Model distillation

Use large LLMs to teach smaller models for cheaper repeated execution.

Slogan

Need cheaper scale? Distill big-model intelligence into small-model execution.

Limitation

Cheaper but narrower; may fail on long-tail or complex cases.

Table-GPT: Table-tuned GPT for Diverse Table Tasks (SIGMOD 2024)

L1 · suggest | L2 · execute | L3 · orchestrate | L4 · proactive

T-1 Missing-value identification

Instruction:
Given the table below, which row and column has a missing value?

Table (preview):

row-id	name	grade	math	art	music	...
row-1	Jennifer	G-2	98	94	89	...
row-2	James	G-2	99	93	...	

Model response:
In row “row-2”, column “music” ✗

T-2 Column finding

Instruction:
Given the table below, which column has the value “93”?

Table (preview):

row-id	name	grade	math	art	music	...
row-1	Jennifer	G-2	98	94	89	...
row-2	James	G-2	99	86	93	...

Model response:
The value “93” is in column “art” ✗

Table-GPT: Table-tuned GPT for Diverse Table Tasks (SIGMOD 2024)

L1 · suggest | L2 · execute | L3 · orchestrate | L4 · proactive

Task-name	Task description (related work)	Task category	Table data	Train/Test
T-1: Missing-value identification (MV)	Identify the row and column position of the only missing cell in a given table	Table understanding	synthesized	Test only
T-2: Column-finding (CF)	Identify the column-name of a specific value that appears only once in a given table	Table Understanding	synthesized	Test only
T-3: Table-QA (TQA)	Answer a natural-language question based on the content of a table ([11, 42, 49])	Table QA	[42]	Test only
T-4: Column type annotation (CTA)	Find the semantic type of a column, from a given list of choices ([16, 25, 63])	Table understanding	[16, 25]	Test only
T-5: Row-to-row transform (R2R)	Transform table data based on input/output examples ([23, 24, 27])	Data transformation	synthesized (test: [24])	Train/Test
T-6: Entity matching (EM)	Match rows from two tables that refer to the same real-world entity ([32, 38, 41, 66])	Table matching	[1]	Train/Test
T-7: Schema matching (SM)	Match columns from two tables that refer to the same meaning ([30, 36, 44])	Table matching	synthesized (test: [30])	Train/Test
T-8: Data imputation (DI)	Predict the missing values in a cell based on the table context ([7, 37])	Data cleaning	synthesized	Train/Test
T-9: Error detection (ED)	Detect data values in a table that is a likely error from misspelling ([14, 45])	Data cleaning	synthesized	Train/Test
T-10: List extraction (LE)	Extract a structured table, from a list that lacks explicit column delimiters [9, 13, 19]	Data transformation	synthesized	Train only
T-11: Head value matching (HVM)	Match column-headers with its data values drawn from the same table	Table matching	synthesized	Train only
T-12: Natural-language to SQL (NS)	Translate a natural-language question on a table into a SQL query ([62, 65])	NL-to-SQL	[65]	Train only
T-13: Table summarization (TS)	Produce a natural-language summary for the content in a table	Data augmentation	synthesized	Train only
T-14: Column augmentation (CA)	Augment a table with additional columns compatible with a given table	Data augmentation	synthesized	Train only
T-15: Row augmentation (RA)	Augment a table with additional rows compatible with a given table	Data augmentation	synthesized	Train only
T-16: Row/column swapping (RCSW)	Manipulate a given table, by swapping the position of two rows or columns	Table manipulation	synthesized	Train only
T-17: Row/column filtering (RCF)	Manipulate a given table, by filtering on given rows or columns	Table manipulation	synthesized	Train only
T-18: Row/column sorting (RCS)	Manipulate a given table, by performing sorting on given rows or columns	Table manipulation	synthesized	Train only

Table-GPT: Table-tuned GPT for Diverse Table Tasks (SIGMOD 2024)

L1 · suggest | L2 · execute | L3 · orchestrate | L4 · proactive

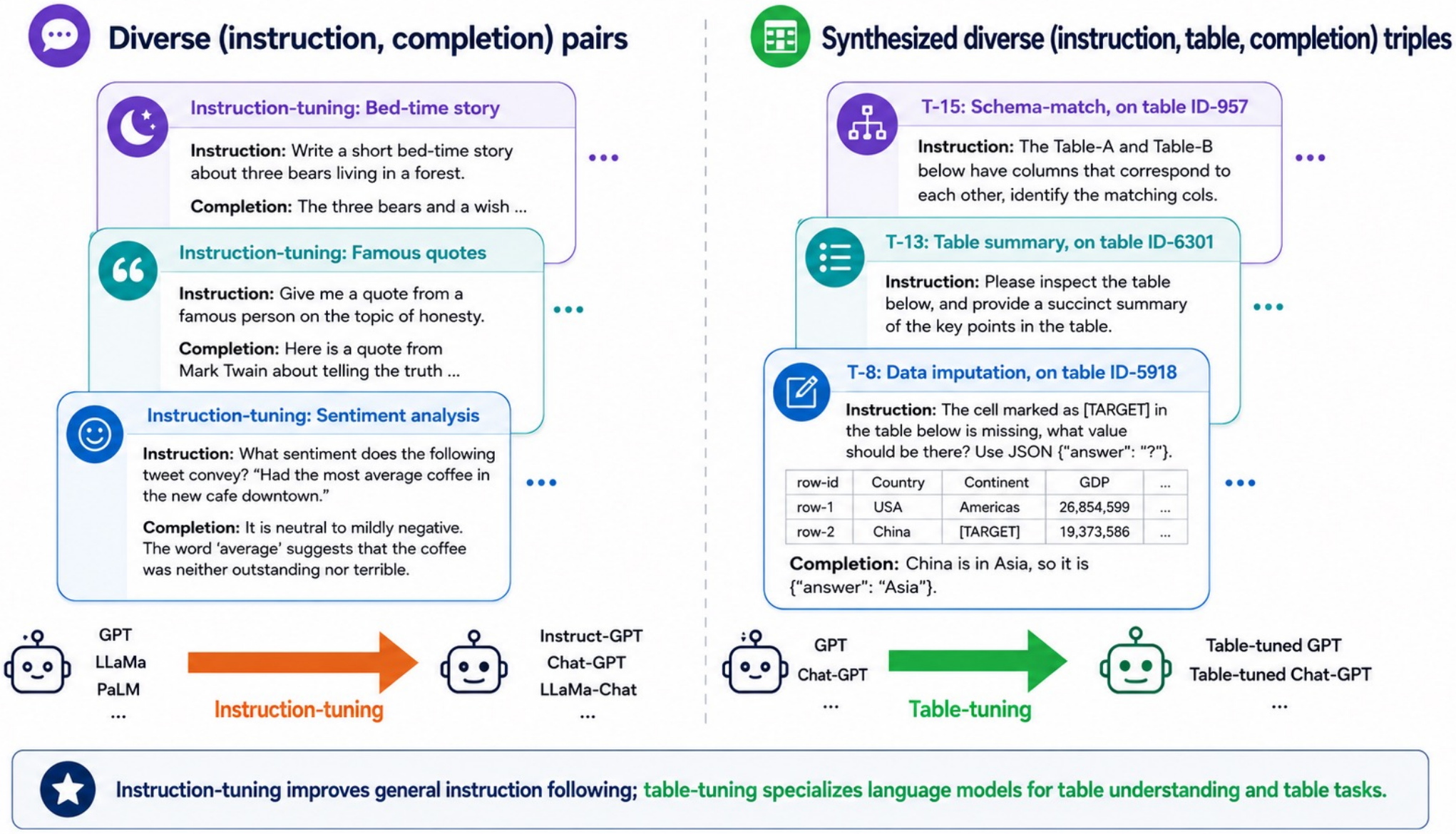
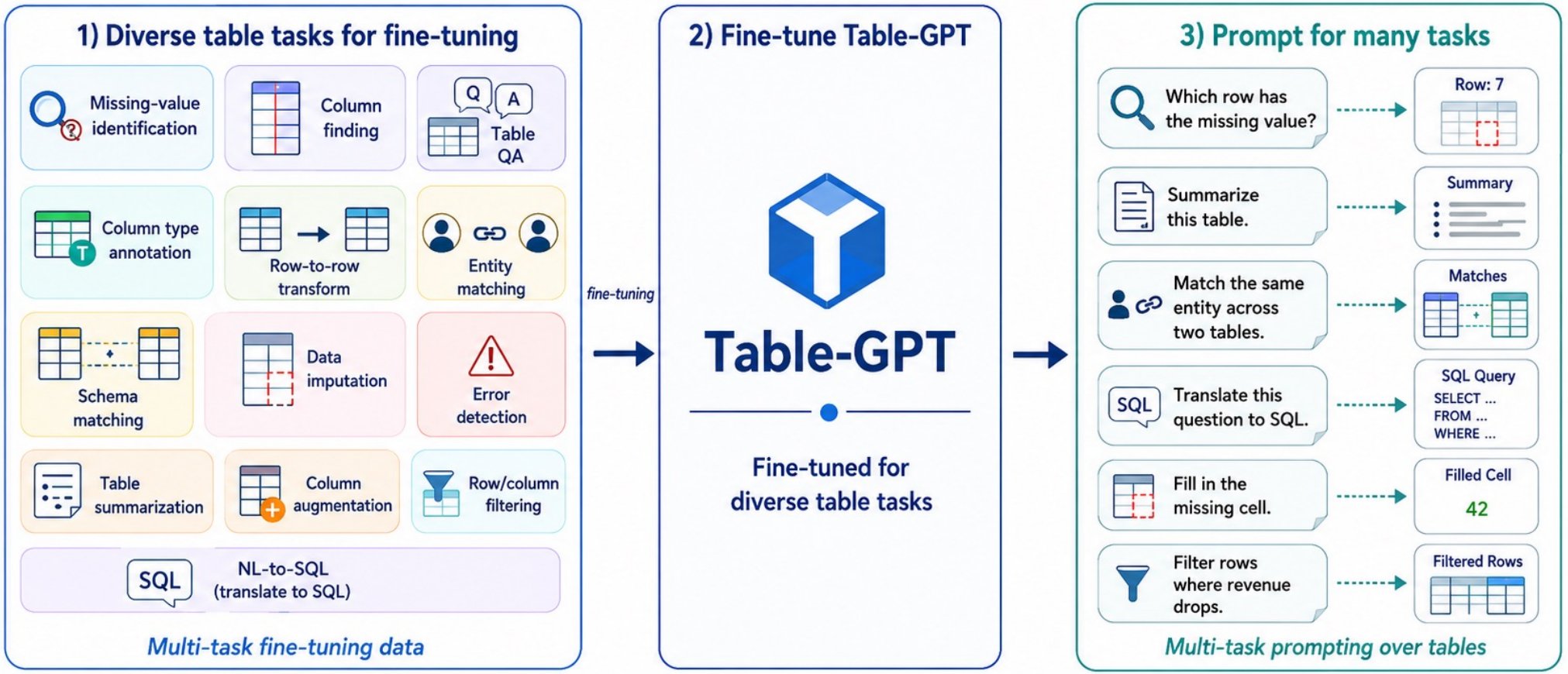


Table-GPT: Table-tuned GPT for Diverse Table Tasks (SIGMOD 2024)

L1 · suggest | L2 · execute | L3 · orchestrate | L4 · proactive

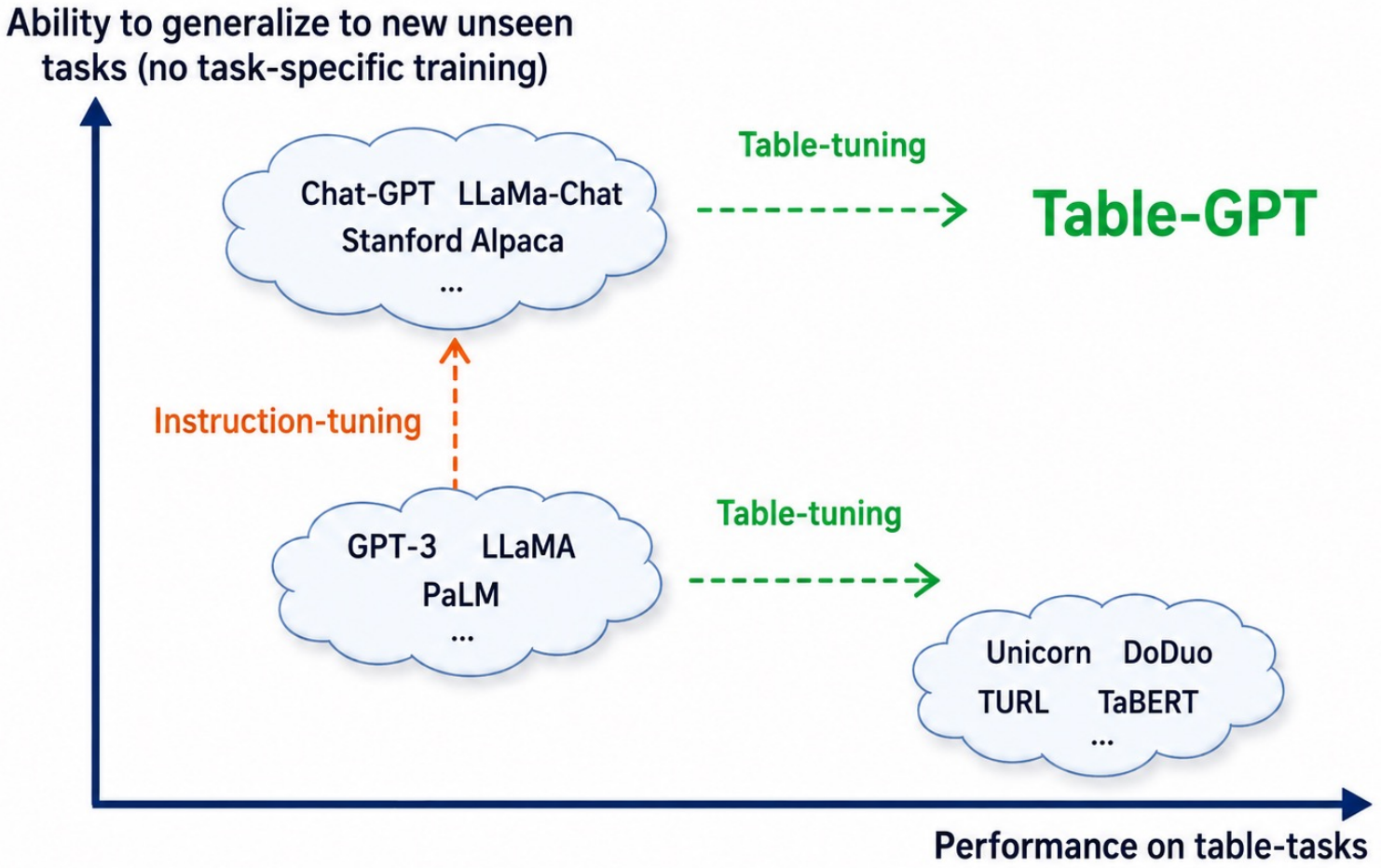


Why L1? Table-GPT supports many tasks through prompting after fine-tuning, but it remains a prompt-response assistant: no environment interaction, no tool execution, no autonomous pipeline control.

L1: Assistance

Table-GPT: Table-tuned GPT for Diverse Table Tasks (SIGMOD 2024)

L1 · suggest | L2 · execute | L3 · orchestrate | L4 · proactive



Instruction-tuning improves generalization;
table-tuning improves table understanding and table-task performance.

Table-GPT: Table-tuned GPT for Diverse Table Tasks (SIGMOD 2024)

L1 · suggest | L2 · execute | L3 · orchestrate | L4 · proactive

Task Type	Task	Dataset	Zero-Shot		Few-Shot		Zero-Shot		Few-Shot	
			GPT-3.5	+table-tune	GPT-3.5	+table-tune	ChatGPT	+table-tune	ChatGPT	+table-tune
Unseen	Column Finding	Spreadsheets-CF	0.461	**0.713**	0.682	**0.816**	0.699	**0.807**	0.803	**0.848**
	Column Type Annotation	Efthymiou	0.757	**0.886**	0.784	**0.847**	0.823	**0.882**	0.806	**0.861**
		Limaye	0.683	**0.755**	0.719	**0.853**	0.742	**0.769**	0.832	**0.853**
		Sherlock	0.332	**0.449**	0.528	**0.538**	0.454	**0.482**	0.521	**0.553**
		T2D	0.776	**0.875**	0.83	**0.915**	0.827	**0.886**	0.853	**0.912**
	Missing Value Identification	Column (no separator)	0.261	**0.294**	0.383	**0.441**	0.299	**0.351**	0.468	**0.474**
		Column (with separator)	0.305	**0.457**	0.519	**0.643**	0.422	**0.520**	0.635	**0.665**
		Row (no separator)	0.768	**0.851**	0.774	**0.882**	0.822	**0.840**	0.859	**0.894**
		Row (with separator)	0.875	**0.959**	0.917	**0.976**	0.923	**0.936**	0.960	**0.968**
	Table Question	Wiki	0.45	**0.486**	0.454	**0.478**	0.512	**0.521**	0.520	**0.527**
Seen	Data Imputation	Spreadsheets-DI	0.423	**0.558**	0.57	**0.625**	0.524	**0.594**	0.609	**0.649**
	Entity Matching	Amazon-Google	0.153	**0.657**	0.659	**0.676**	0.239	**0.566**	0.680	**0.701**
		Beer	0.5	**0.727**	0.815	**0.923**	0.741	**0.923**	0.783	**0.963**
		DBLP-ACM	0.402	**0.847**	0.954	**0.912**	0.833	**0.932**	0.961	**0.938**
		DBLP-GoogleScholar	0.206	**0.861**	0.809	**0.896**	0.632	**0.912**	0.823	**0.924**
		Fodors-Zagats	0.083	**0.872**	0.872	**0.977**	0.809	**1.000**	0.872	**0.977**
		Walmart-Amazon	0.268	**0.691**	0.519	**0.711**	0.206	**0.678**	0.664	**0.824**
		iTunes-Amazon	0	**0.788**	0.826	**0.943**	0.393	**0.862**	0.833	**0.929**
	Error Detection	Spreadsheets-Real	0.058	**0.565**	0.319	**0.552**	0.058	**0.544**	0.443	**0.551**
		WebTables-Real	0.077	**0.643**	0.338	**0.545**	0.078	**0.656**	0.364	**0.684**
	Schema Matching	DeepM	1	**1**	1	**1**	0.857	**1**	1	**1**
	Row-to-Row Transformation	BingQL-Unit	N.A.		0.213	**0.427**	N.A.		0.339	**0.446**
		BingQL-other			0.431	**0.588**			0.558	**0.607**
		FF-GR-Trifacta			0.712	**0.788**			0.772	**0.825**
		Headcase			0.636	**0.705**			0.704	**0.795**
		Stackoverflow			0.662	**0.745**			**0.800**	0.758

Semantic Operators: L1

Model tuning

Fine-tune or adapt LLMs for table understanding, table QA, extraction, and transformation.

Slogan

Table-tuning helps today; semantic operators survive tomorrow.

Limitation

Model-specific; hard to keep pace with fast LLM updates.

Data grounding

Retrieve better tuples, examples, documents, or evidence before asking the LLM.

Slogan

LLMs are strong reasoners; good data makes them reliable.

Limitation

Depends on retrieval quality; still mostly one-shot assistance.

Model distillation

Use large LLMs to teach smaller models for cheaper repeated execution.

Slogan

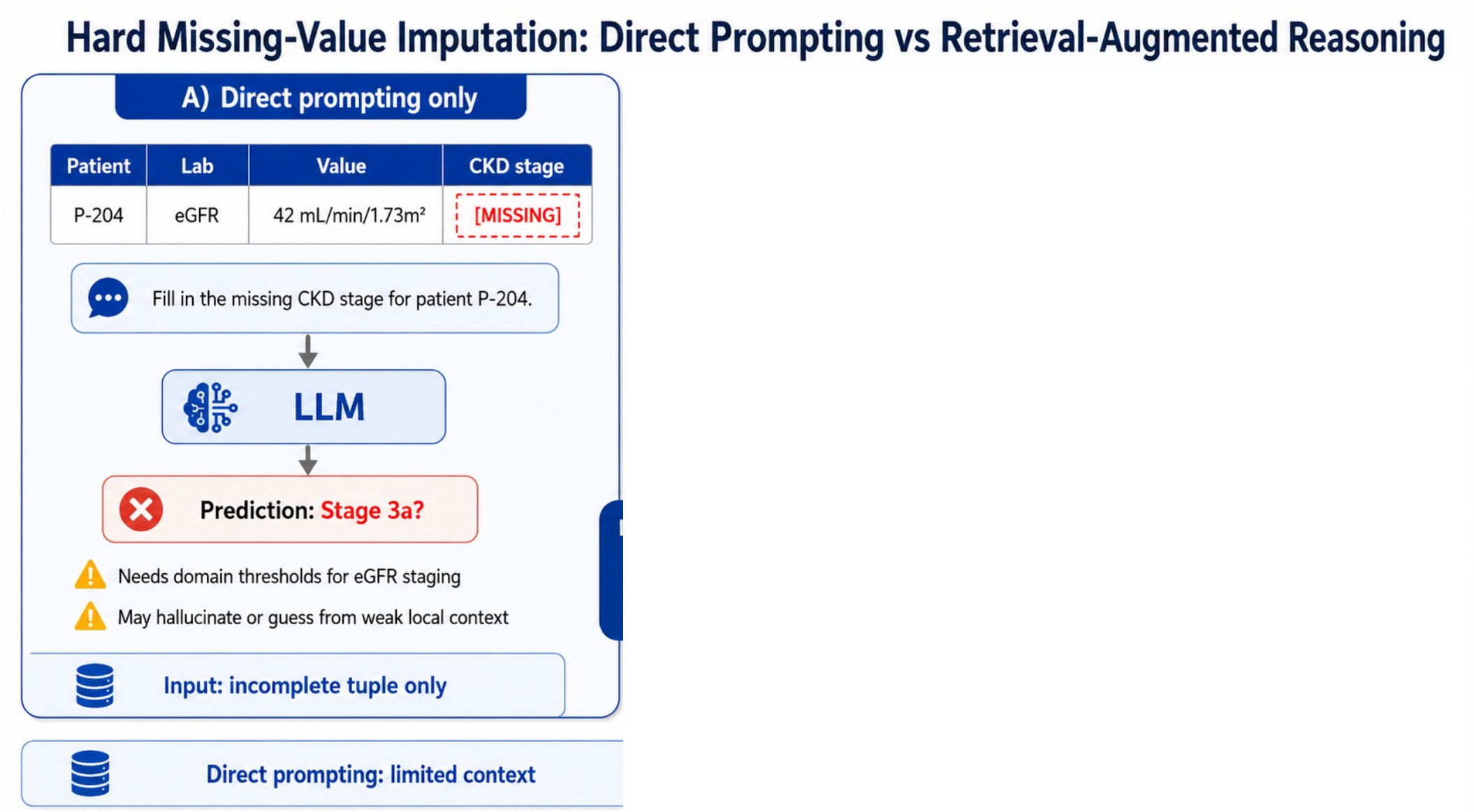
Need cheaper scale? Distill big-model intelligence into small-model execution.

Limitation

Cheaper but narrower; may fail on long-tail or complex cases.

Data Imputation with Limited Data Redundancy Using Data Lakes (VLDB 2025)

L1 · suggest | L2 · execute | L3 · orchestrate | L4 · proactive



Data Imputation with Limited Data Redundancy Using Data Lakes (VLDB 2025)

L1 · suggest | L2 · execute | L3 · orchestrate | L4 · proactive

When redundancy is scarce, imputation must search and combine evidence from external sources

A. Enough redundancy

self-contained

Incomplete tuple

Representative	State	District	Party	Birth date
Bob Riley	Alabama	NA	NA	NA

repair from the same table

Sufficient redundancy within one table

Representative	State	District	Party	Birth date
Bob Riley	Alabama	AL-3	R	NA
Bob Riley	Alabama	NA	R	Oct. 3, 1944
Bob Riley	Alabama	AL-3	NA	Oct. 3, 1944
Don Siegelman	Alabama	NA	Democratic	July 6, 1945

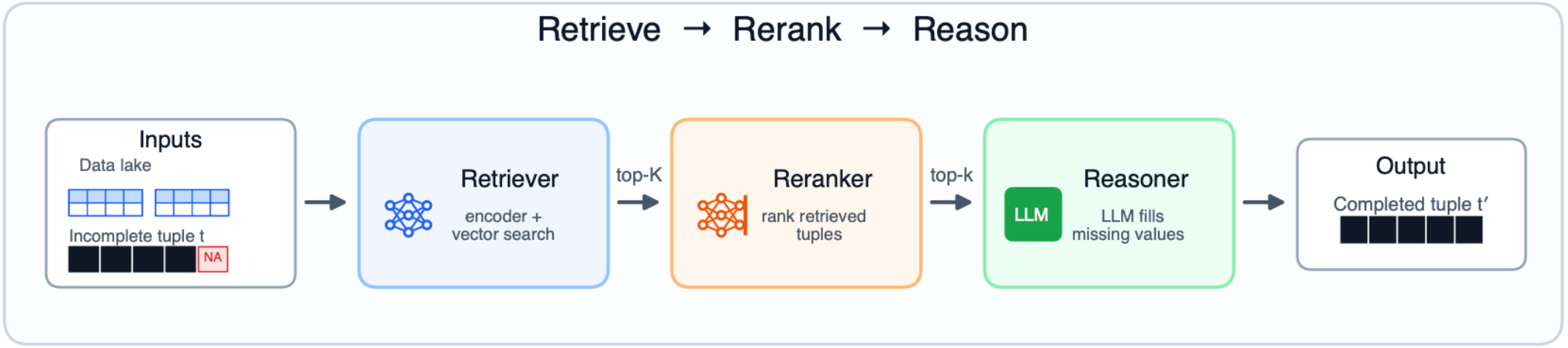
local redundancy is enough

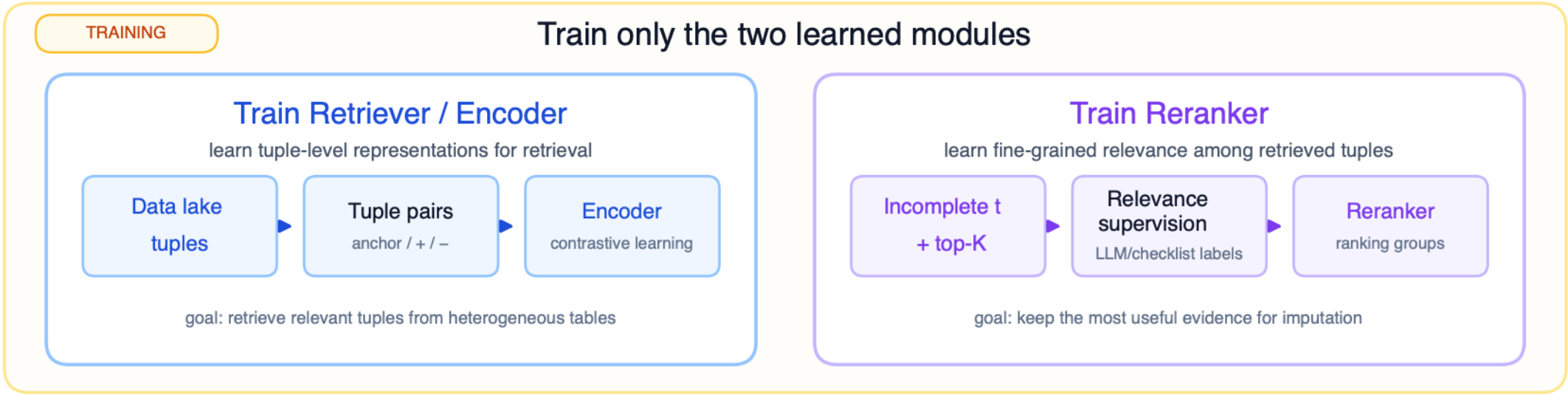
Completed tuple

Representative	State	District	Party	Birth date
Bob Riley	Alabama	AL-3	R	Oct. 3, 1944

Data Imputation with Limited Data Redundancy Using Data Lakes (VLDB 2025)

L1 · suggest | L2 · execute | L3 · orchestrate | L4 · proactive





Training Data Synthesis for Training the Encoder

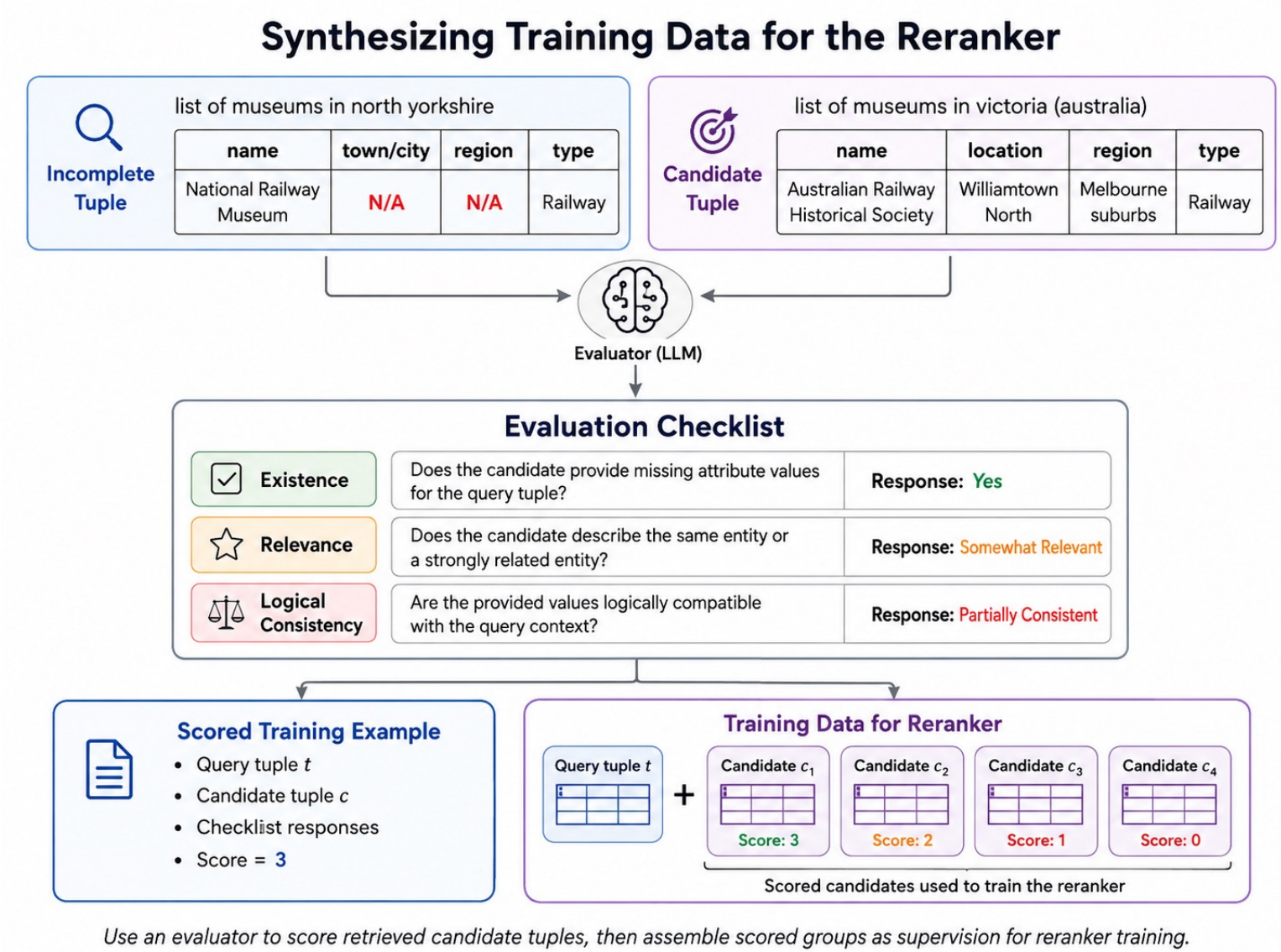
Consider a sample tuple with a caption and attribute/value pairs:

Caption: "Harrisburg, Pennsylvania, Sports"
(**club:** Harrisburg …, **league:** USL Soccer, **venue:** Skyline …)

Type	Operator	Example
Caption	delete_cap	Harrisburg, **NA,** Sports
	replace_cap	Harrisburg, Pennsylvania, **Athletic**
	shuffle_cap	**Pennsylvania, Harrisburg, Sports**
Attribute	shuffle_attr	new attribute order: (**league, venue, club**)
	delete_attr	new attribute set: (**club, venue**)
Value	replace_val	(Harrisburg …, **United Soccer League,** Skyline …)
	empty_val	(Harrisburg …, **NA,** Skyline …)

Data Imputation with Limited Data Redundancy Using Data Lakes (VLDB 2025)

L1 · suggest | L2 · execute | L3 · orchestrate | L4 · proactive



Use an evaluator to score retrieved candidate tuples, then assemble scored groups as supervision for reranker training.

Data Imputation with Limited Data Redundancy Using Data Lakes (VLDB 2025)

L1 · suggest | L2 · execute | L3 · orchestrate | L4 · proactive

LakeFill Dataset Statistics and Imputation Accuracy

Benchmark Statistics and Dataset Names

Dataset	Incomplete Tuples		Data Lake		Avg. Relevant Tup.	Part of Missing Attrs	#Training Tup.
	#Tab.	#Tup.	#Tab.	#Tup.			
WikiTuples (WT)	665	6,887	207,912	2,674,164	3.98	Party, Director, Team, ...	100
Show Movie (SM)	1	30	3	19,586	1	Age Rating	6
Cricket Players (CP)	1	213	2	94,164	1.38	Nationality, Batting Style	20
Education (ED)	2	654	17	11,132	4	Address, Zipcode, Phone	30
Zomato (ZM)	1	529	16	468,252	3.48	Location	30
FIFA World Cup (FF)	1	696	14	118,251	1.44	Home / Away Team Goals	30

Exact Match (EM) Accuracy of Data Imputation

"Tup." denotes tuples.

Reasoner	Retriever	WT	SM	ED	CP	ZM	FF
GPT-4o mini	w/o	0.63	0.75	0.119	0.904	0.0	0.475
	w/ tup. (BM25)	0.679	0.75	0.908	0.905	0.754	0.543
	w/ tup. (LakeFill)	**0.881**	**0.875**	**0.977**	**0.921**	**0.914**	**0.927**
GPT-4o	w/o	0.809	0.75	0.112	0.938	0.0	0.484
	w/ tup. (BM25)	0.815	0.833	0.928	0.932	0.758	0.551
	w/ tup. (LakeFill)	**0.907**	**0.917**	**0.978**	**0.97**	**0.924**	**0.948**

Upper table introduces dataset names/statistics; lower table reports EM accuracy across the same datasets.

Semantic Operators: L1

Model tuning

Fine-tune or adapt LLMs for table understanding, table QA, extraction, and transformation.

Slogan

Table-tuning helps today; semantic operators survive tomorrow.

Limitation

Model-specific; hard to keep pace with fast LLM updates.

Data grounding

Retrieve better tuples, examples, documents, or evidence before asking the LLM.

Slogan

LLMs are strong reasoners; good data makes them reliable.

Limitation

Depends on retrieval quality; still mostly one-shot assistance.

Model distillation

Use large LLMs to teach smaller models for cheaper repeated execution.

Slogan

Need cheaper scale? Distill big-model intelligence into small-model execution.

Limitation

Cheaper but narrower; may fail on long-tail or complex cases.

Long-document QA with Chain-of-structured thought and fine-tuned SLMs (ICLR 2026)

L1 · suggest | L2 · execute | L3 · orchestrate | L4 · proactive

Q1 How did the trend of ADM Endeavors, Inc.'s operating profit margin evolve from 2021 to 2024?

Documents:
10-k financial reports from 2021-2024

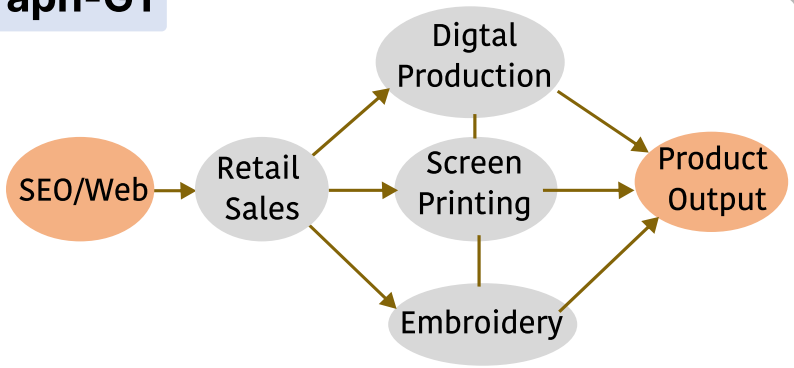
ADM Endeavors, Inc. report various asset categories in thousands of dollars, which include operating income of $1,21,237 and operating expenses of 5,158,755 in 2021, The other expenses are respectively...

Q2 In ADM Endeavors' departments, how many steps are required from lead generation to final product output?

Table-T1

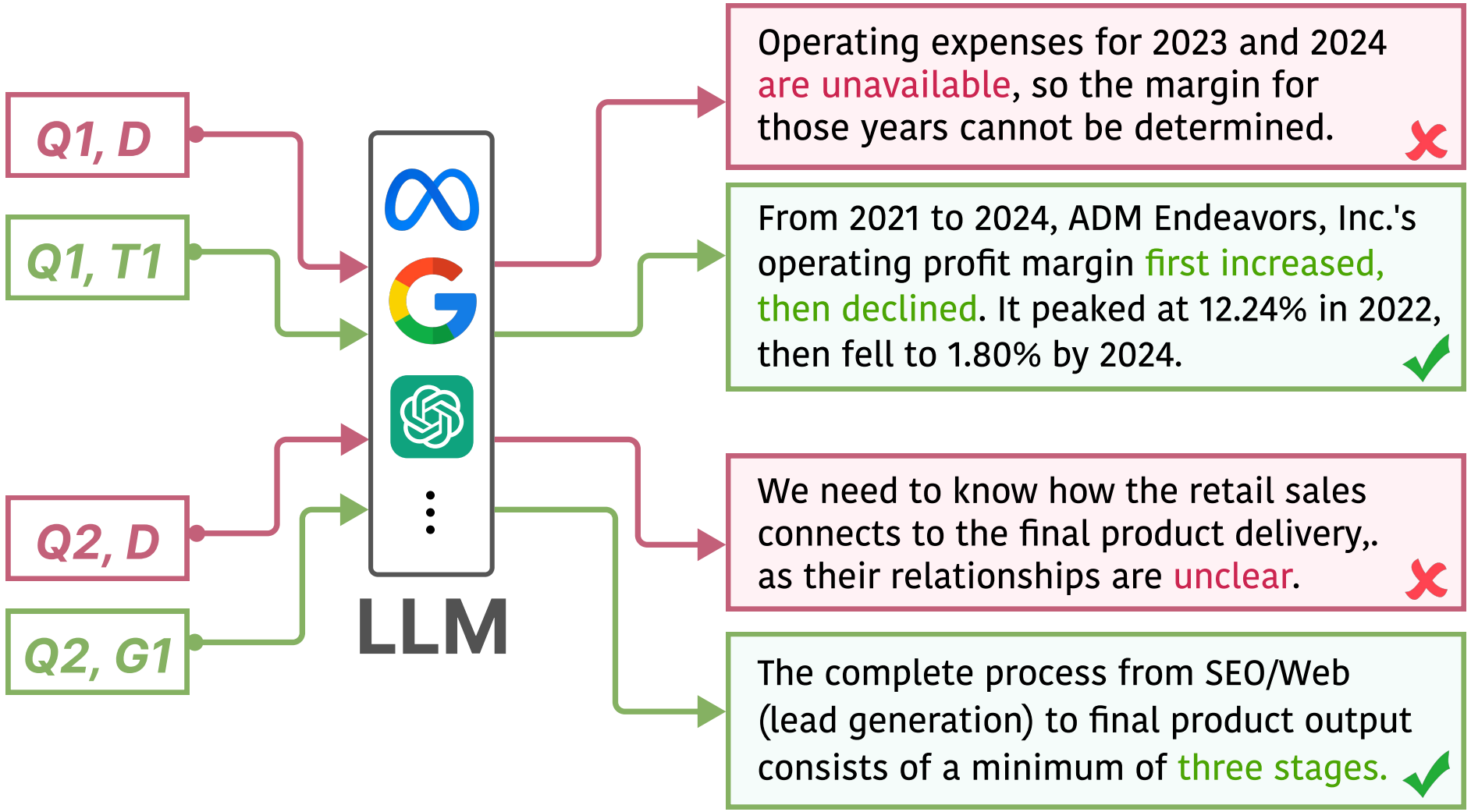
Company	Income	Expenses	Year
ADM Endeavors, Inc.	715,076	5,841,788	2022
ADM Endeavors, Inc.	121,237	5,158,755	2021
ADM Endeavors, Inc.	91,884	5,097,046	2024
ADM Endeavors, Inc.	353,044	5,271,456	2023

Graph-G1



Long-document QA with Chain-of-structured thought and fine-tuned SLMs (ICLR 2026)

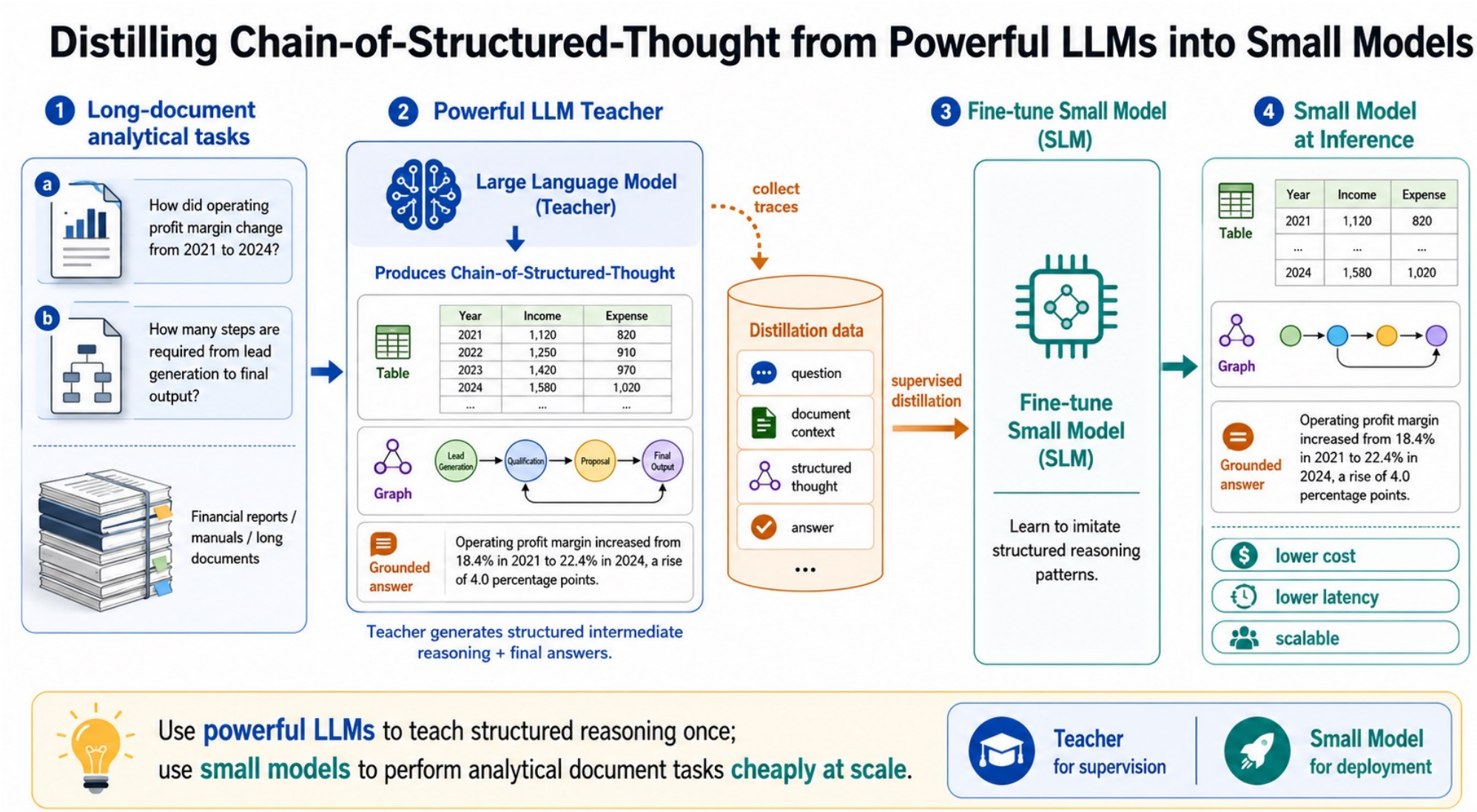
L1 · suggest | L2 · execute | L3 · orchestrate | L4 · proactive



Question: *Powerful closed-sourced LLMs can extract data, but can it be cheaper?*

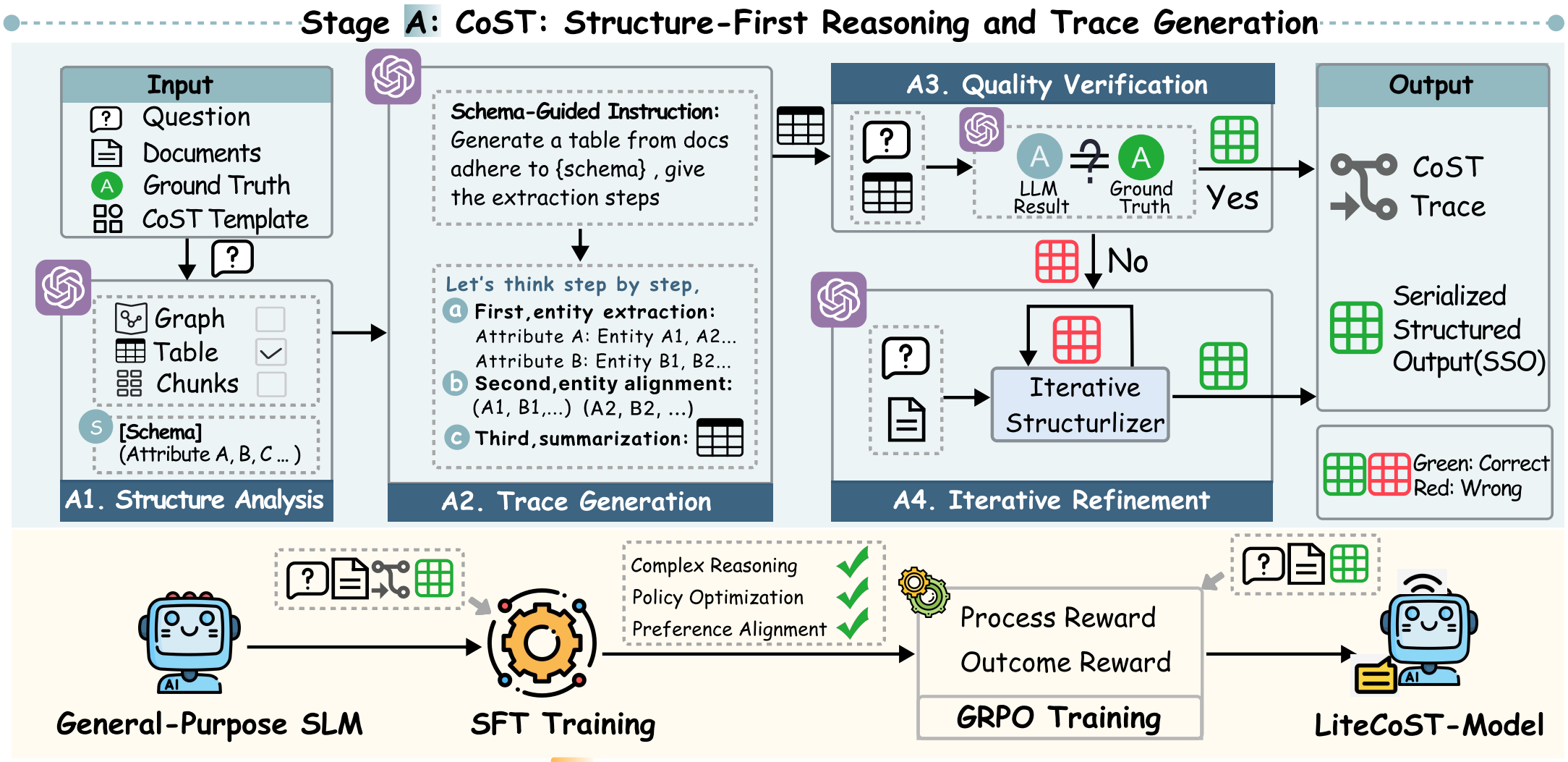
Long-document QA with Chain-of-structured thought and fine-tuned SLMs (ICLR 2026)

L1 · suggest | L2 · execute | L3 · orchestrate | L4 · proactive



Long-document QA with Chain-of-structured thought and fine-tuned SLMs (ICLR 2026)

L1 · suggest | L2 · execute | L3 · orchestrate | L4 · proactive



Long-document QA with Chain-of-structured thought and fine-tuned SLMs (ICLR 2026)

L1 · suggest | L2 · execute | L3 · orchestrate | L4 · proactive

Comparison of Different Models for Structured Long-Document QA (Finance Subset of Loong)

Green highlights indicate improvements over the corresponding base model.

Model	Model Size	Spotlight Locating		Comparison		Clustering		Chain of Reasoning		Overall	
		AS	PR	AS	PR	AS	PR	AS	PR	AS	PR
Closed-Sourced Models & Large Language Models											
LLaMA-3.1-8B-Instruct	8B	55.03	0.20	51.60	0.15	51.50	0.04	44.75	0.02	51.32	0.10
GPT-4o-mini	8B	84.42	0.70	80.40	0.67	77.38	0.40	65.35	0.18	78.08	0.51
Qwen2.5-14B-Instruct	14B	83.74	0.57	82.12	0.56	69.96	0.24	66.41	0.10	75.60	0.38
GPT-4o (Abacha et al., 2025)	200B	84.10	0.73	80.53	0.60	81.50	0.50	64.30	0.25	79.32	0.54
Deepseek-R1 (Guo et al., 2025)	671B	84.27	0.62	78.97	0.55	75.42	0.34	74.40	0.35	78.18	0.46
LLaMA-3.2-3B-Instruct (Base)	3B	49.90	0.16	52.10	0.14	47.89	0.07	46.85	0.06	49.37	0.11
LLaMA-3.2-3B-Instruct (Ours)	**3B**	**81.27**	**0.53**	**78.08**	**0.49**	**78.34**	**0.36**	**64.75**	**0.16**	**76.95**	**0.40**
		↑31.37	↑0.37	↑25.98	↑0.35	↑30.45	↑0.29	↑17.90	↑0.10	↑27.58	↑0.29
Qwen2-7B-Instruct (Base)	7B	63.10	0.36	67.85	0.37	60.83	0.18	52.25	0.09	62.10	0.26
Qwen2-7B-Instruct (Ours)	**7B**	**83.97**	**0.62**	**81.55**	**0.59**	**81.00**	**0.43**	**67.98**	**0.18**	**79.93**	**0.48**
		↑20.87	↑0.26	↑13.70	↑0.22	↑20.17	↑0.25	↑15.73	↑0.09	↑17.83	↑0.22

Semantic Operators: L1

Model tuning

Fine-tune or adapt LLMs for table understanding, table QA, extraction, and transformation.

Slogan

Table-tuning helps today; semantic operators survive tomorrow.

Limitation

Model-specific; hard to keep pace with fast LLM updates.

Data grounding

Retrieve better tuples, examples, documents, or evidence before asking the LLM.

Slogan

LLMs are strong reasoners; good data makes them reliable.

Limitation

Depends on retrieval quality; still mostly one-shot assistance.

Model distillation

Use large LLMs to teach smaller models for cheaper repeated execution.

Slogan

Need cheaper scale? Distill big-model intelligence into small-model execution.

Limitation

Cheaper but narrower; may fail on long-tail or complex cases.

Core Data Agent Challenges and Tutorial Roadmap

Challenge \ Level	L1 Assistance	L2 Partial Autonomy	L3 Conditional Autonomy	L4 High Autonomy	L5 Full Autonomy
	Prompt-driven Data Assistant	Human-guided Workflow Executor	Agentic Workflow Orchestrator	Proactive Data Agent	Self-evolving Autonomous Data Agent
C1. Data-aware Workflow Orchestration & Cost-aware Execution	Human-designed workflow; agent suggests.	Execute human-designed pipelines with local feedback.	Auto-compose and optimize workflows / DAGs.	Proactively discover tasks and plan long-horizon execution.	Invent new operators, tools, and workflow paradigms.
C2. Long-horizon Agentic State,	Stateless; no	Short-term task	Workflow state + provenance +	Long-lived cross-	Self-evolving
C3. Semantic Grounding over Heterogeneous & Multimodal Data	Prompt / few-shot / RAG-based grounding.	Environment-aware schema, entity, and evidence grounding.	Semantic catalog + metadata + multimodal evidence alignment.	Continuous semantic maintenance over evolving data lakes.	Create new semantic abstraction and representations.

Semantic Grounding over Heterogeneous & Multimodal Data (40 min)

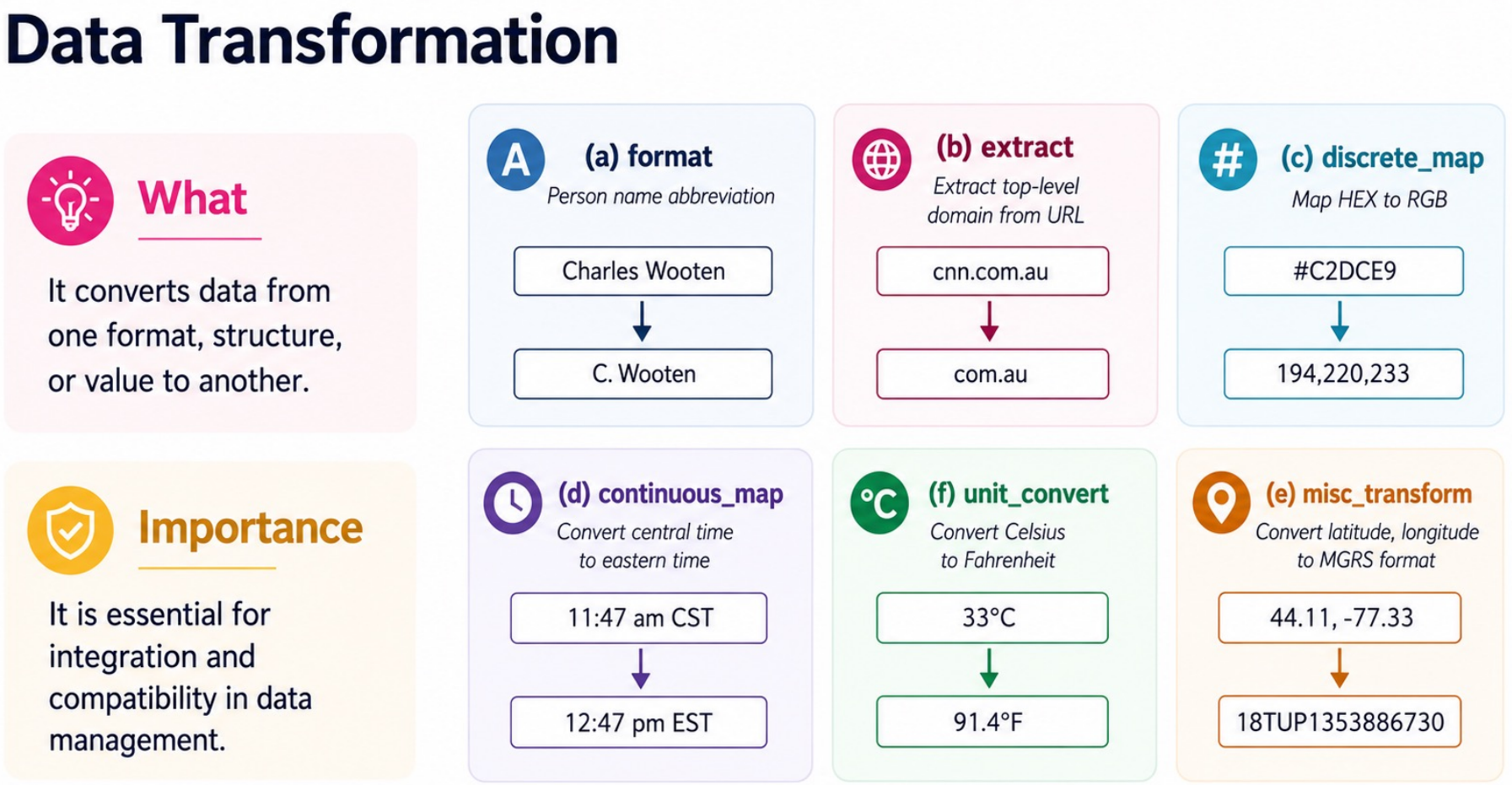
Progression: Assist & Suggest » Execute & Improve » Orchestrate & Optimize » Proactively Act » Invent & Evolve

Semantic operators: L2

L2	Work	How feedback enters	Boundary
Transform by generated code	**MegaTran**	**Generate code** → run it **Inspect** runtime / sanity-check errors **Reflector** suggests fixes **Lazy-RAG** retrieves code and docs	Fixed transformation task; local repair, not pipeline planning.
Rewrite & validate operator pipelines	**DocETL**	**Decompose** document operations **Synthesize** validation prompts **Test** candidate rewrites **Select** a better plan	Optimizes a given workflow; not task discovery.

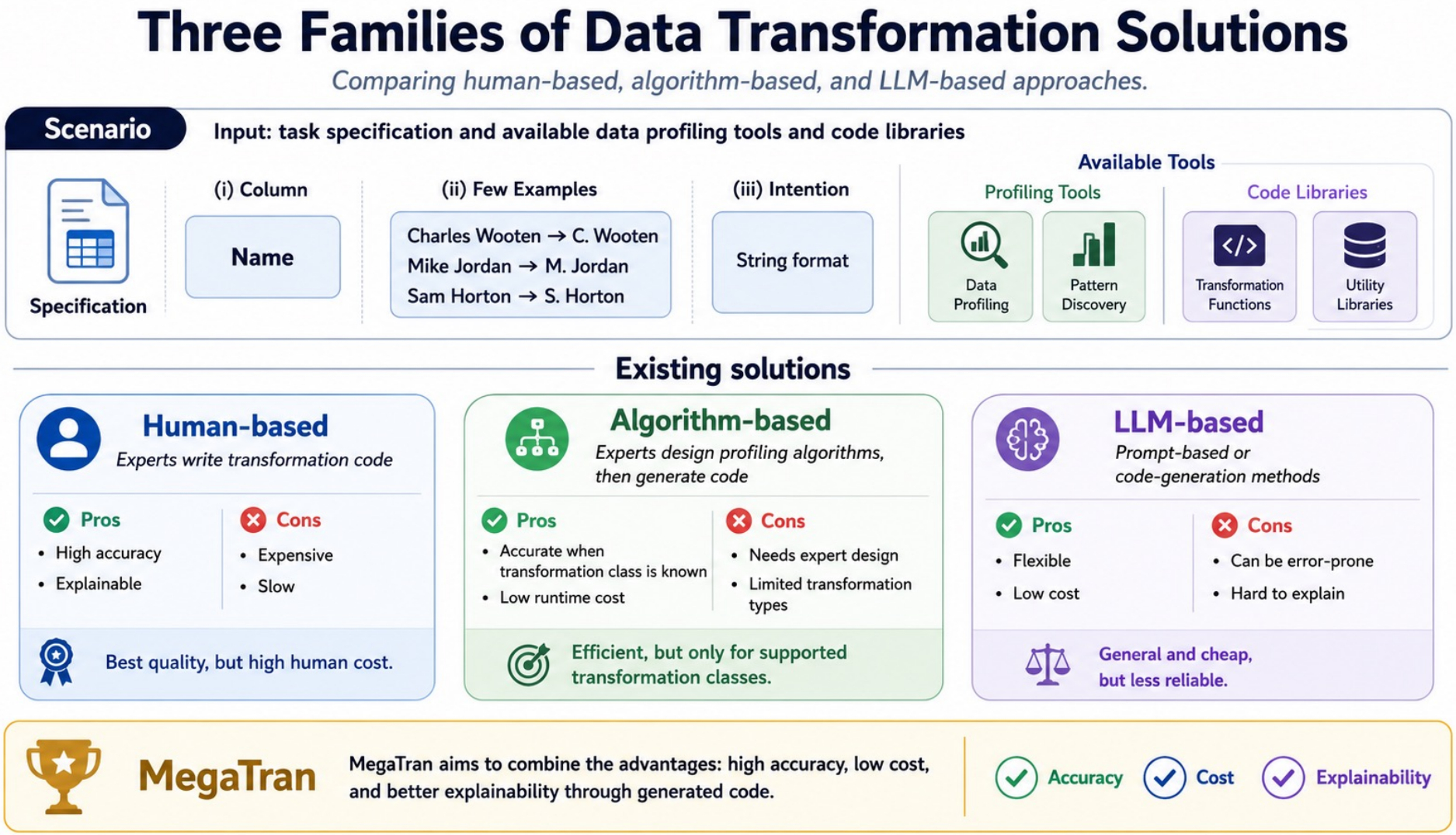
Weak-to-strong prompts with lightweight-to-powerful LLMs for high-accuracy, low-cost, and explainable data transformation (VLDB 2025)

L1 · suggest | L2 · execute | L3 · orchestrate | L4 · proactive



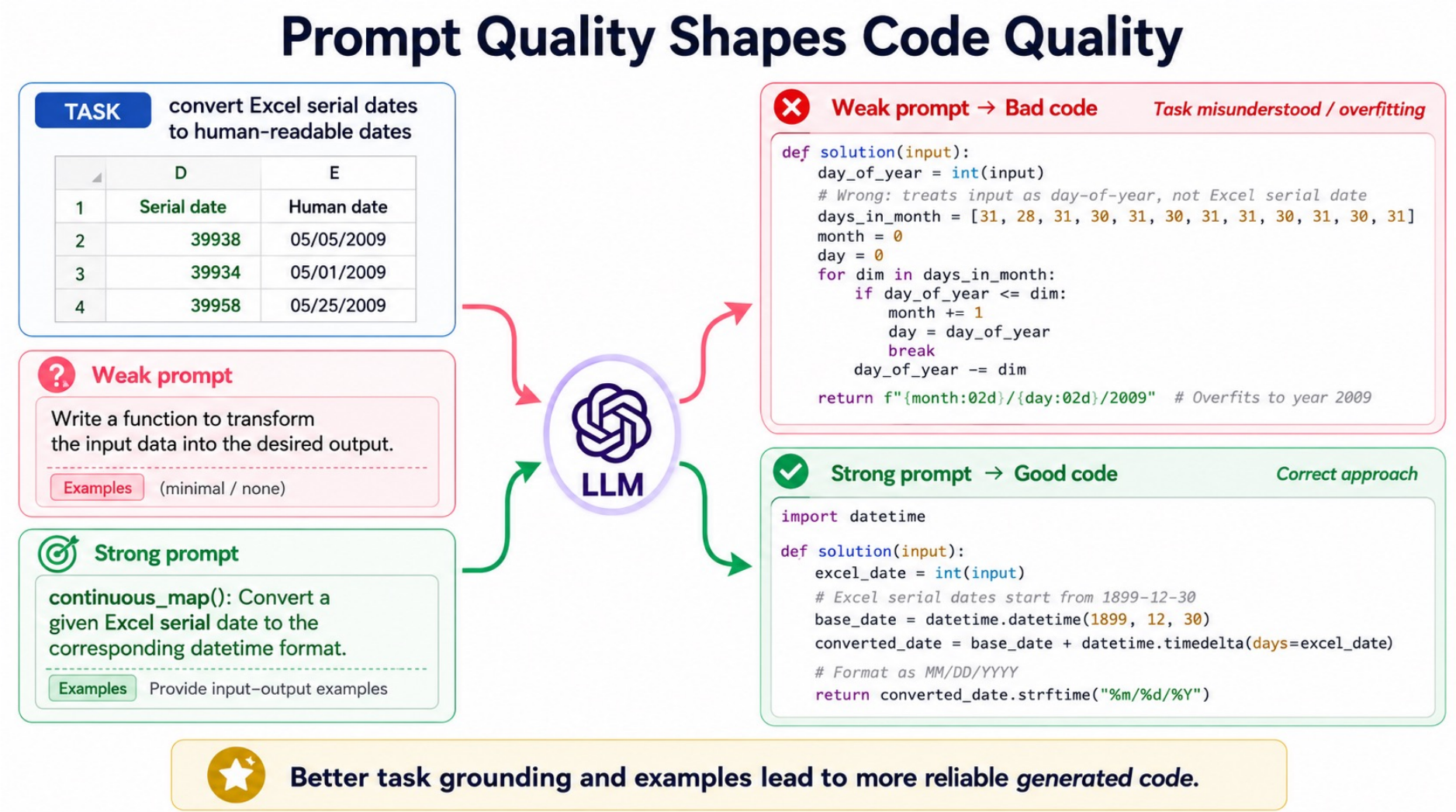
Weak-to-strong prompts with lightweight-to-powerful LLMs for high-accuracy, low-cost, and explainable data transformation (VLDB 2025)

L1 · suggest | L2 · execute | L3 · orchestrate | L4 · proactive



Weak-to-strong prompts with lightweight-to-powerful LLMs for high-accuracy, low-cost, and explainable data transformation (VLDB 2025)

L1 · suggest | L2 · execute | L3 · orchestrate | L4 · proactive



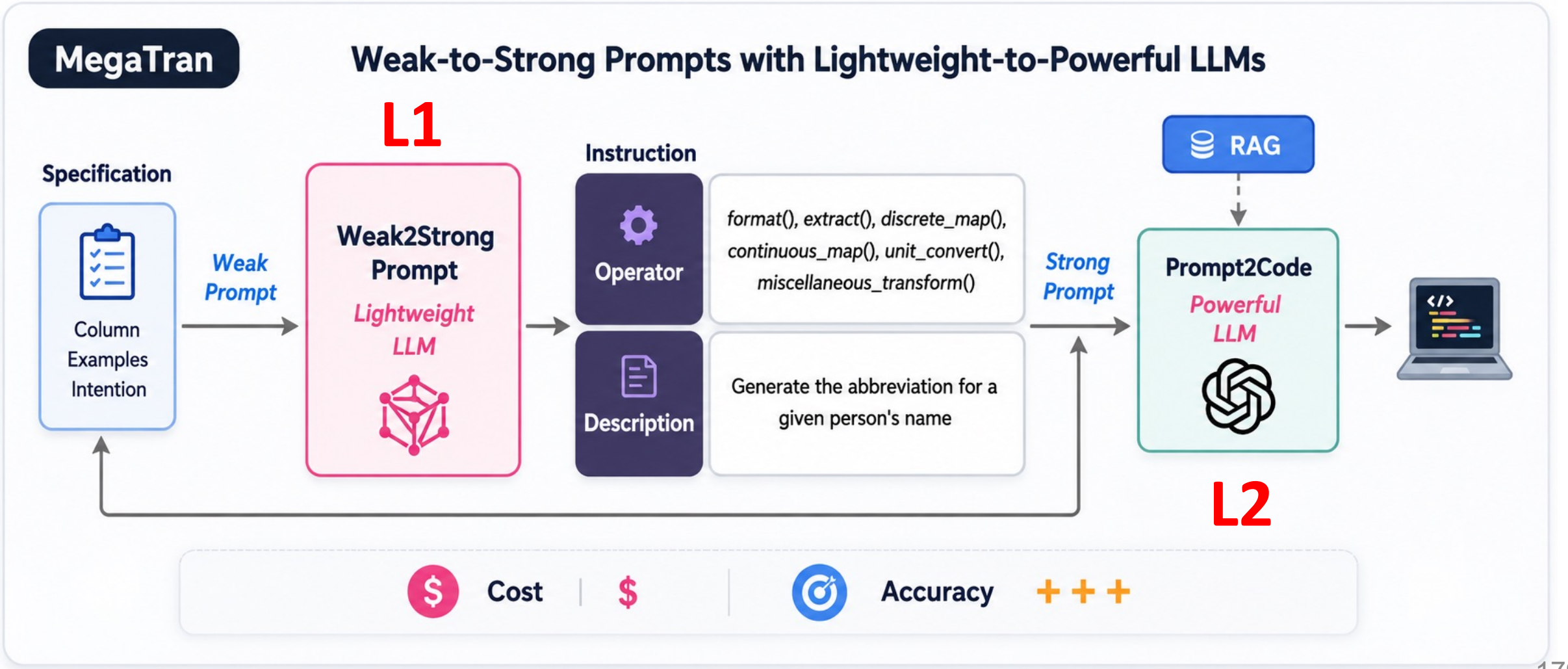
Challenges: Prompt-based Code Generation

1. Users often provide **weak prompts**, which describe imprecise or vague transformation requests
   -> rewrite as **strong prompts**

2. Strong prompts are not enough: need additional knowledge, code libraries
   -> optimized with executionfeedback

Weak-to-strong prompts with lightweight-to-powerful LLMs for high-accuracy, low-cost, and explainable data transformation (VLDB 2025)

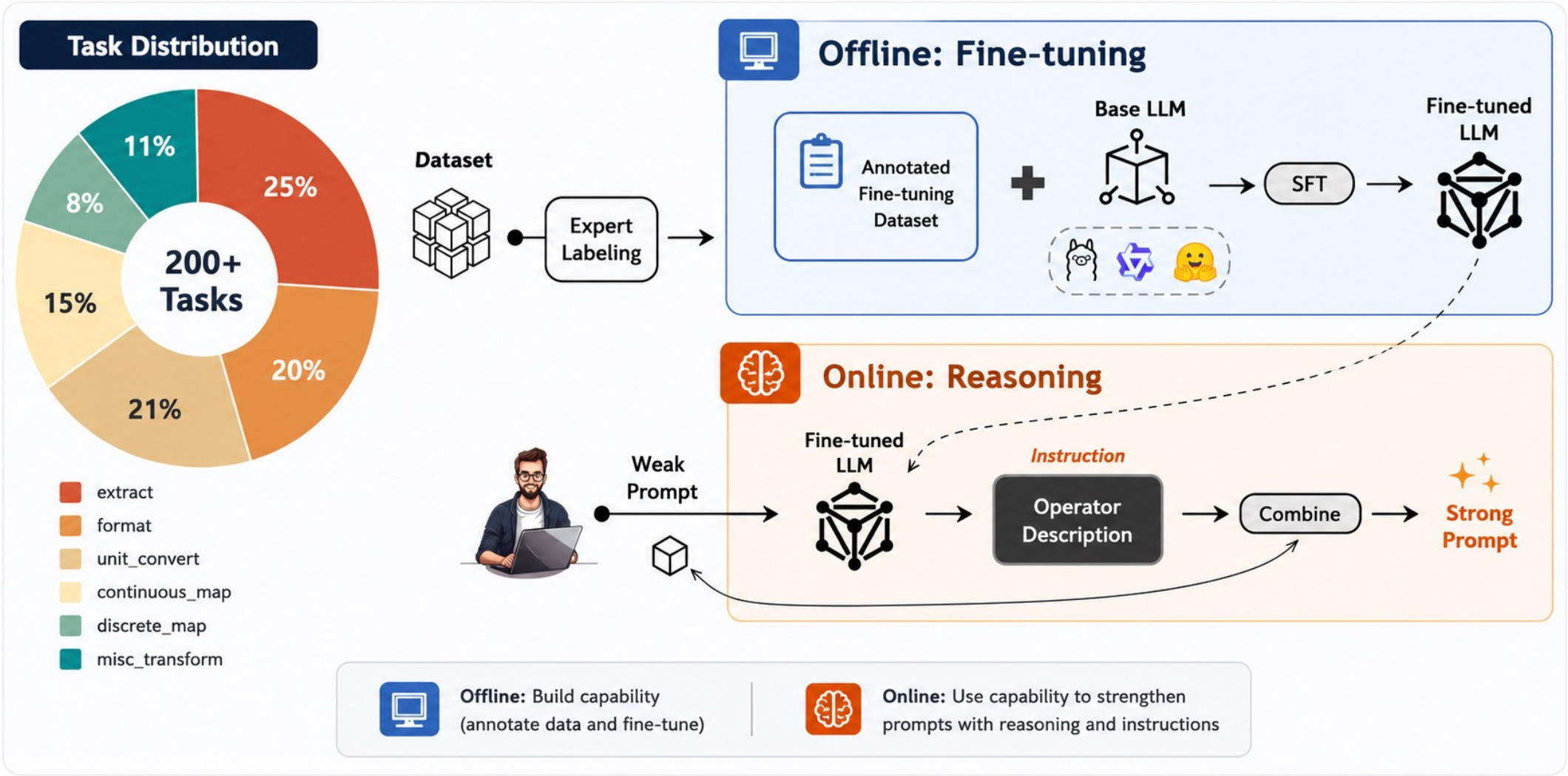
L1 · suggest | L2 · execute | L3 · orchestrate | L4 · proactive



Weak-to-strong prompts with lightweight-to-powerful LLMs for high-accuracy, low-cost, and explainable data transformation (VLDB 2025)

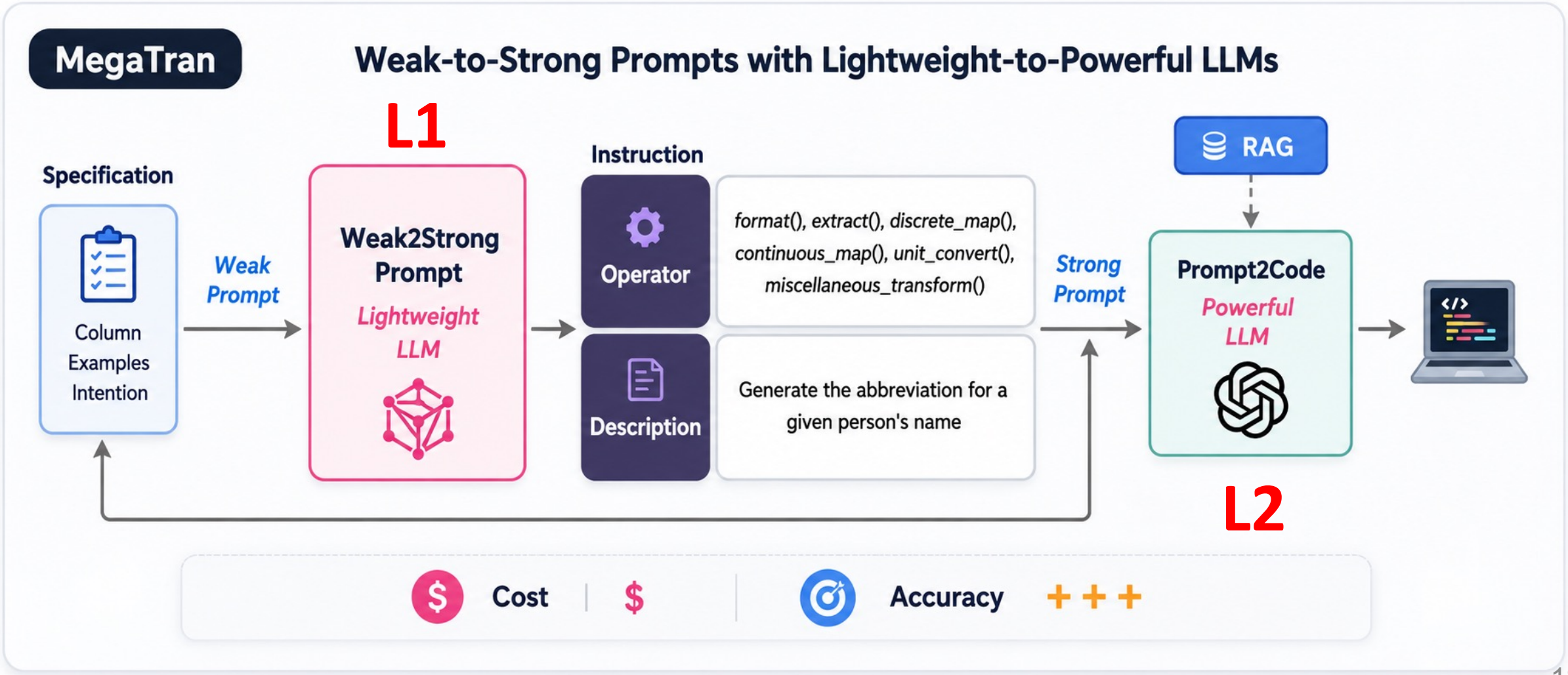
L1 · suggest | L2 · execute | L3 · orchestrate | L4 · proactive

Stage 1: Weak2Strong Prompt (L1)



Weak-to-strong prompts with lightweight-to-powerful LLMs for high-accuracy, low-cost, and explainable data transformation (VLDB 2025)

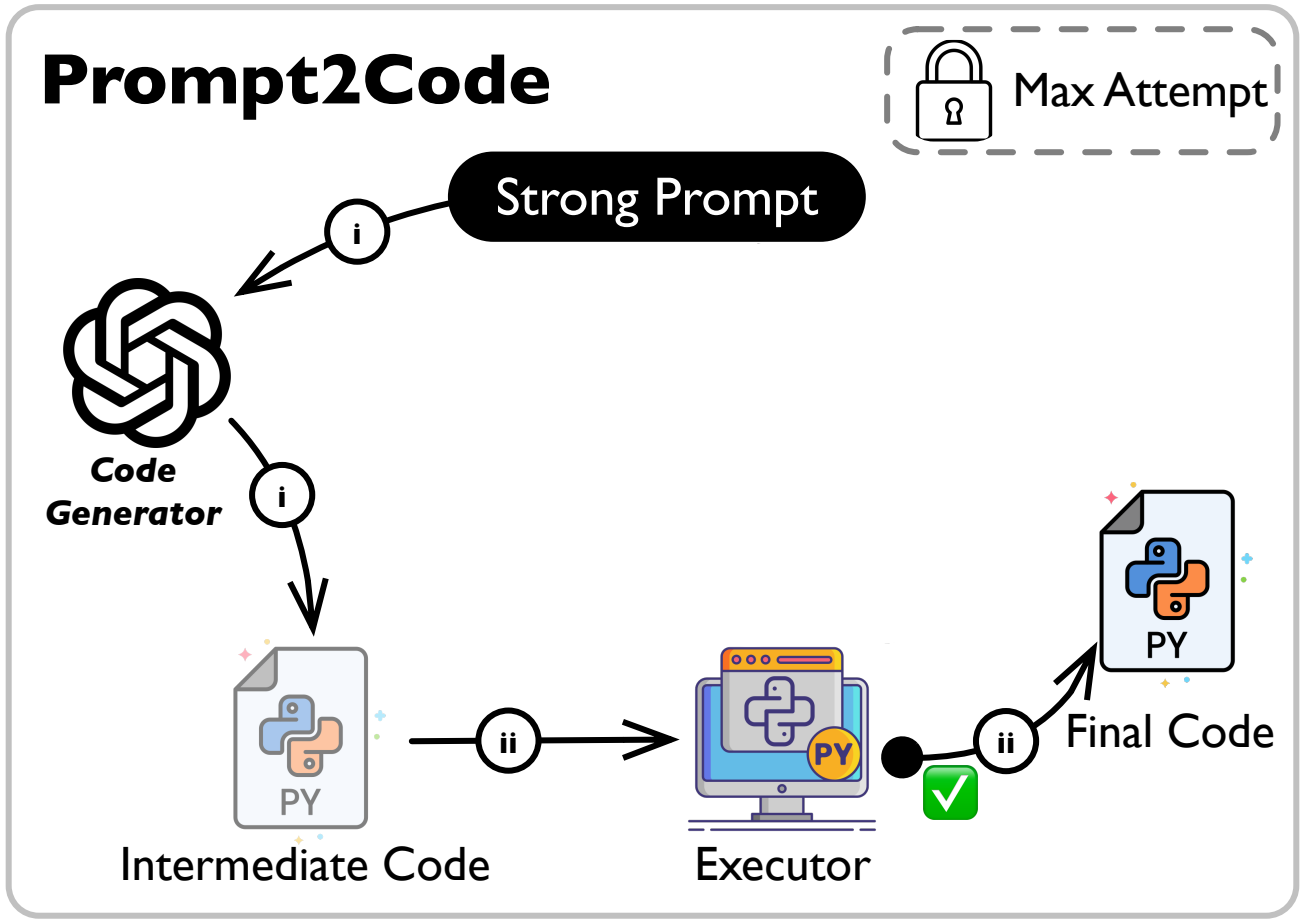
L1 · suggest | L2 · execute | L3 · orchestrate | L4 · proactive



Weak-to-strong prompts with lightweight-to-powerful LLMs for high-accuracy, low-cost, and explainable data transformation (VLDB 2025)

L1 · suggest | L2 · execute | L3 · orchestrate | L4 · proactive

Stage 2: Prompt2Code (L2)



Semantic Operators: L2

Transformation execution

MegaTran · Prompt2Code

Generate transformation code, run it, inspect errors, and repair with sanity-check reflection plus Lazy-RAG.

Slogan

From prompt to runnable operator.

Limitation

Weak2Strong improves the prompt; L2 is the execute–debug–retry loop. The pipeline is fixed.

Pipeline rewrite & validation

DocETL

Decompose document tasks, rewrite operator plans, and validate candidate outputs with agent-generated checks.

Slogan

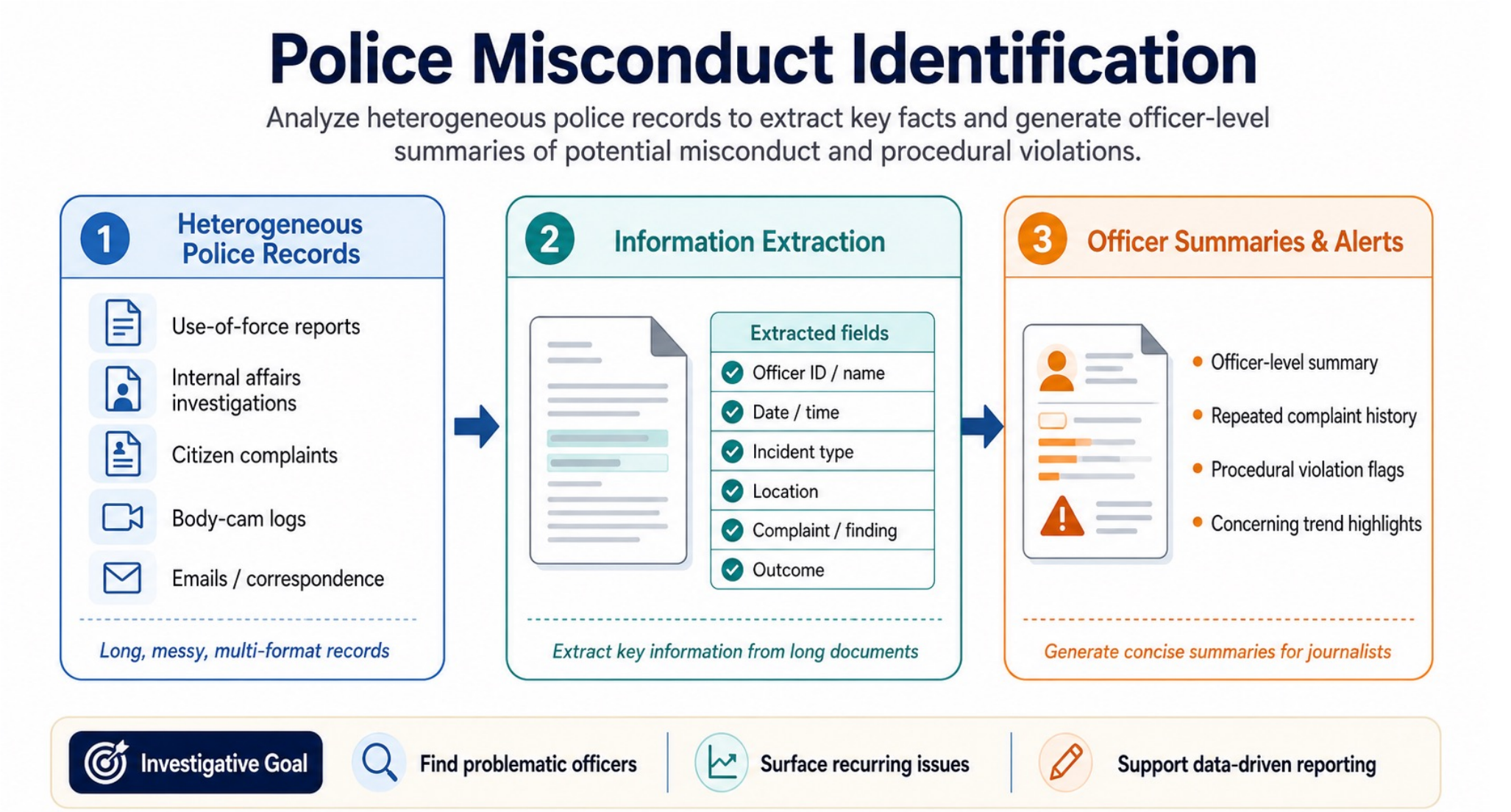
Complex documents need plans, not one big prompt.

Limitation

Improves a given workflow; not full L3 orchestration.

DocETL: Agentic query rewriting and evaluation for complex document processing (VLDB 2025)

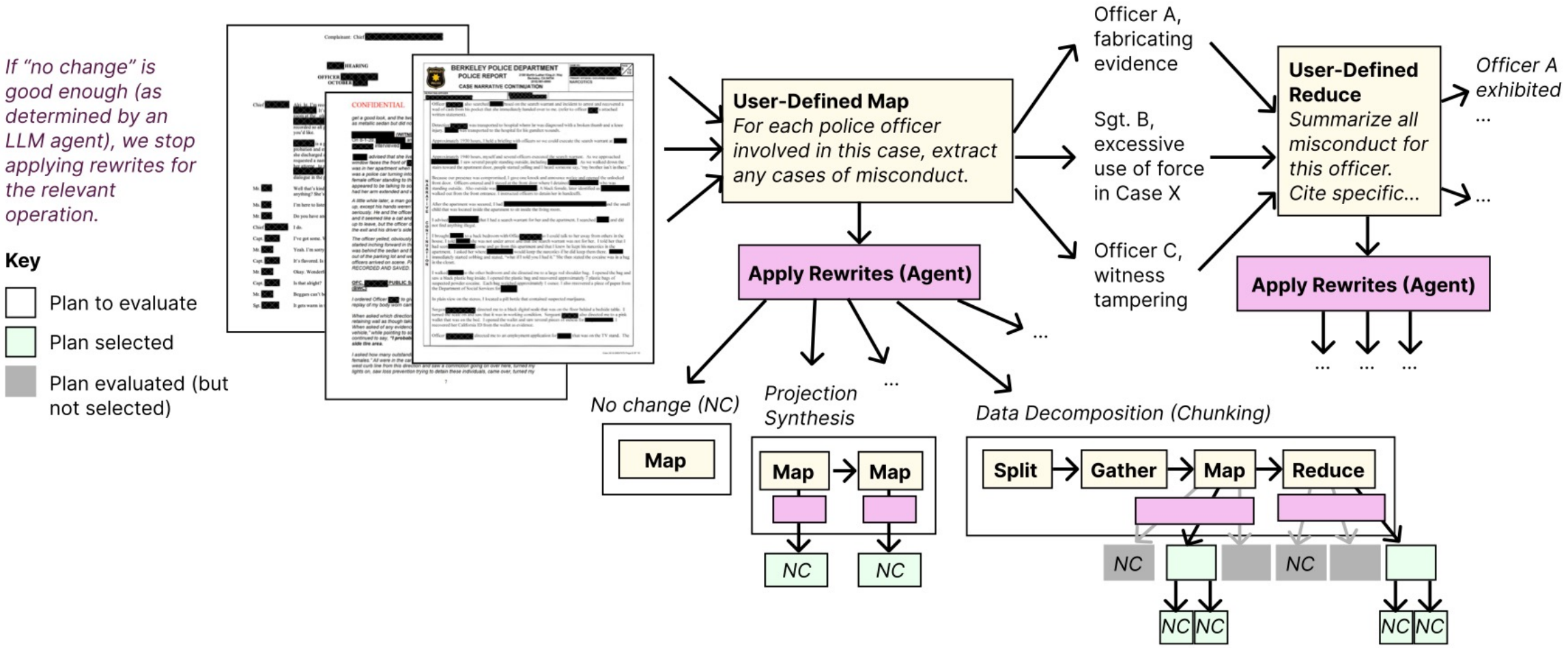
L1 · suggest | L2 · execute | L3 · orchestrate | L4 · proactive



DocETL: Agentic query rewriting and evaluation for complex document processing (VLDB 2025)

L1 · suggest | L2 · execute | L3 · orchestrate | L4 · proactive

1. Big picture — agentic optimizer for document pipelines.
2. Main rewrite example — split/gather for long documents.
3. Quality control — validation and refinement through gleaning.



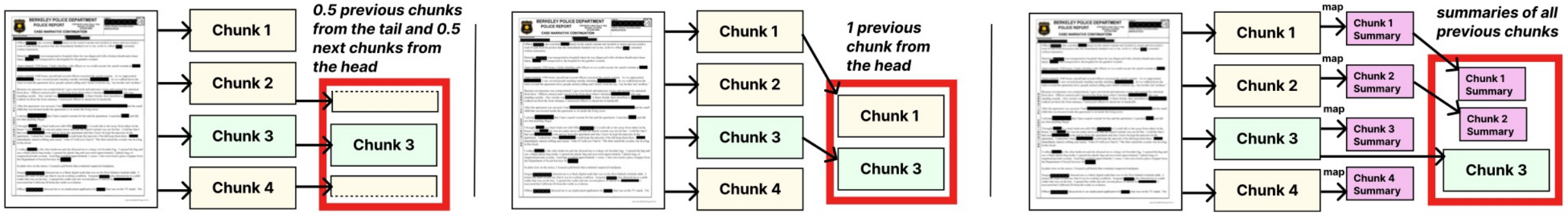
Disclaimer: the figure is from the DocETL paper

DocETL: Agentic query rewriting and evaluation for complex document processing (VLDB 2025)

L1 · suggest | L2 · execute | L3 · orchestrate | L4 · proactive

1. Big picture — agentic optimizer for document pipelines.
2. Main rewrite example — split/gather for long documents.
3. Quality control — validation and refinement through gleaning.

Three different ways of rendering chunk 3



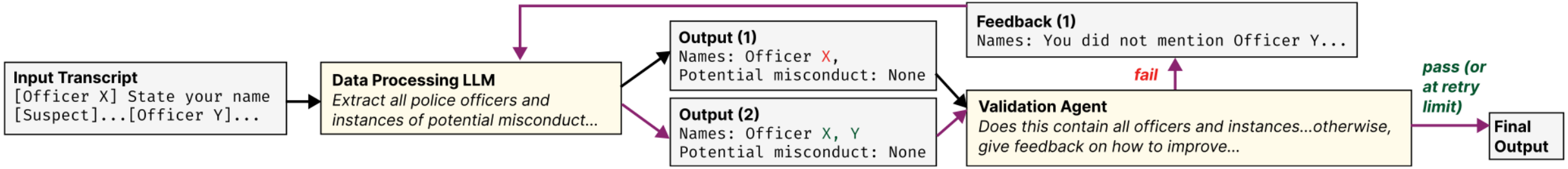
Key insight: not only "chunk the document", but "chunk the document and intelligently reconstruct the right context for each chunk".

Disclaimer: the figure is from the DocETL paper

DocETL: Agentic query rewriting and evaluation for complex document processing (VLDB 2025)

L1 · suggest | L2 · execute | L3 · orchestrate | L4 · proactive

1. Big picture — agentic optimizer for document pipelines.
2. Main rewrite example — split/gather for long documents.
3. Quality control — validation and refinement through gleaning.



In DocETL, LLMs are not trusted as one-shot operators. They are wrapped inside an execution framework with validation, feedback, and refinement.

Disclaimer: the figure is from the DocETL paper

Semantic Operators: L2

Transformation execution

MegaTran · Prompt2Code

Generate transformation code, run it, inspect errors, and repair with sanity-check reflection plus Lazy-RAG.

Slogan

From prompt to runnable operator.

Limitation

Weak2Strong improves the prompt; L2 is the execute–debug–retry loop.

Pipeline rewrite & validation

DocETL

Decompose document tasks, rewrite operator plans, and validate candidate outputs with agent-generated checks.

Slogan

Complex documents need plans, not one big prompt.

Limitation

Improves a given workflow; not full L3 orchestration.

Core Data Agent Challenges and Tutorial Roadmap

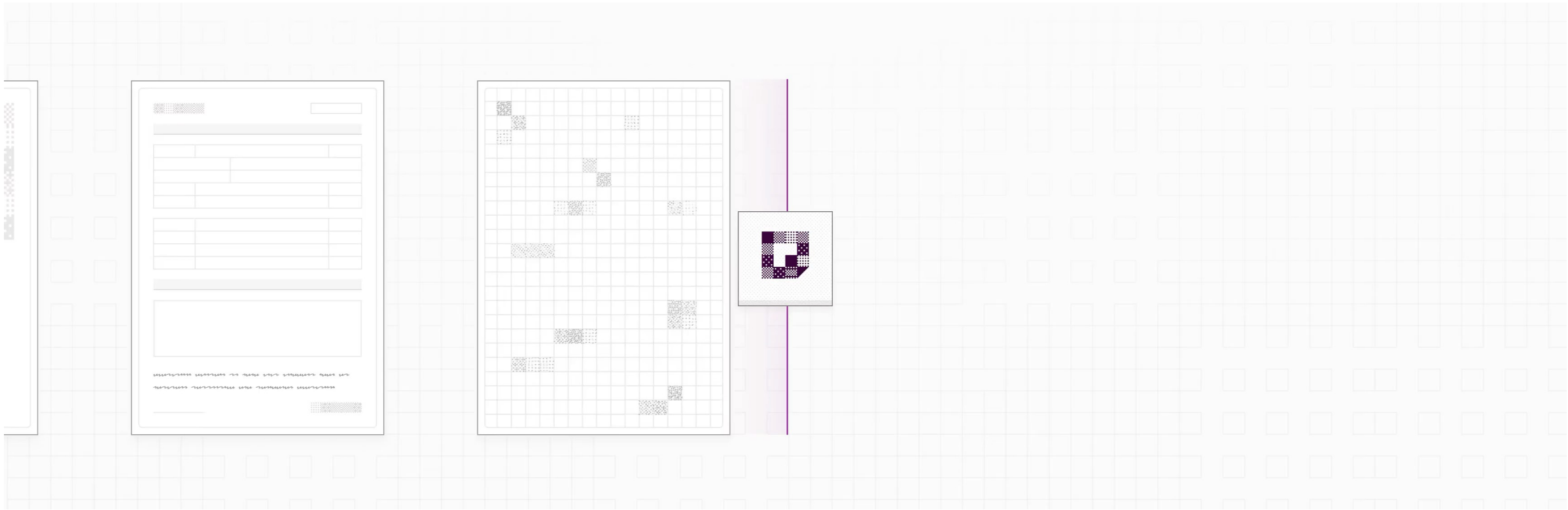
Challenge \ Level	L1 Assistance	L2 Partial Autonomy	L3 Conditional Autonomy	L4 High Autonomy	L5 Full Autonomy
	Prompt-driven Data Assistant	Human-guided Workflow Executor	Agentic Workflow Orchestrator	Proactive Data Agent	Self-evolving Autonomous Data Agent
C1. Data-aware Workflow Orchestration & Cost-aware Execution	Human-designed workflow; agent suggests.	Execute human-designed pipelines with local feedback.	Auto-compose and optimize workflows / DAGs.	Proactively discover tasks and plan long-horizon execution.	Invent new operators, tools, and workflow paradigms.
C2. Long-horizon Agentic State,	Stateless; no	Short-term task	Workflow state + provenance +	Long-lived cross-	Self-evolving
C3. Semantic Grounding over Heterogeneous & Multimodal Data	Prompt / few-shot / RAG-based grounding.	Environment-aware schema, entity, and evidence grounding.	Semantic catalog + metadata + multimodal evidence alignment.	Continuous semantic maintenance over evolving data lakes.	Create new semantic abstraction and representations.

Semantic Grounding over Heterogeneous & Multimodal Data (40 min)

Progression: Assist & Suggest » Execute & Improve » Orchestrate & Optimize » Proactively Act » Invent & Evolve

Document-to-Database: Extraction Meets Relational Semantics (VLDB 2026)

L1 · suggest | L2 · execute | L3 · orchestrate | L4 · proactive



Document-to-Database: Extraction Meets Relational Semantics (VLDB 2026)

L1 · suggest | L2 · execute | L3 · orchestrate | L4 · proactive

(a) A financial report

Blue = entity mentions
Purple = attribute values
Red (underlined) = sources of errors

Arena Tech reported solid performance in fiscal year 2023, with cash increasing from 9,867,000 at the beginning of the year to 11,200,000 at year end.

Alpha Holdings filed its annual report for 2024. The report states that cash began at 18,524 (in thousand) and ended at 32,368,000.

Alpha Holdings is a long-term investor with a significant stake in Arena Tech. According to the company overview, Arena Tech is among the core holdings associated with Alpha Holdings. The report further notes that Arena Tech invested in Coyni Corp.

Regarding Coyni Corp, the report records that its cash position declined from 4,544,000 to 4,062,000 during fiscal year 2023.

Elsewhere, Coyni Inc is mentioned as an affiliate company, but no corresponding financial figures are reported.

(b) Database schema definition

Entity Table

- Company (cid, name, fiscal_year, cash_begin, cash_end)
- Key generation: cid GENERATED ALWAYS AS IDENTITY.

Relationship Table

- Hold(investor_id, investee_id)
- Foreign keys: investor_id,investee_id → Company.cid

Integrity Constraints & Business Rules

ϕ_1: ***No mutual investment:*** $\neg(\text{Hold}(a,b) \wedge \text{Hold}(b,a))$

ϕ_2: ***Cash sanity bound:*** $cash\_end \,/\, cash\_begin < 10$

ϕ_3: ***Recursive investment:*** $\text{Hold}(a,b) \wedge \text{Hold}(b,c) \Rightarrow \text{Hold}(a,c)$

(c) Extracted Company table

Red = error; Blue = correct

cid	name	fiscal_year	cash_begin	cash_end
1	Arena Tech	2023	9,867,000	11,200,000
2	Alpha Holdings	2024	18,524 18,524,000	32,368,000
3	Coyni Corp	2023	4,544,000	4,062,000
4	Coyni Inc	2023	NULL	NULL

(d) Extracted Hold relationships

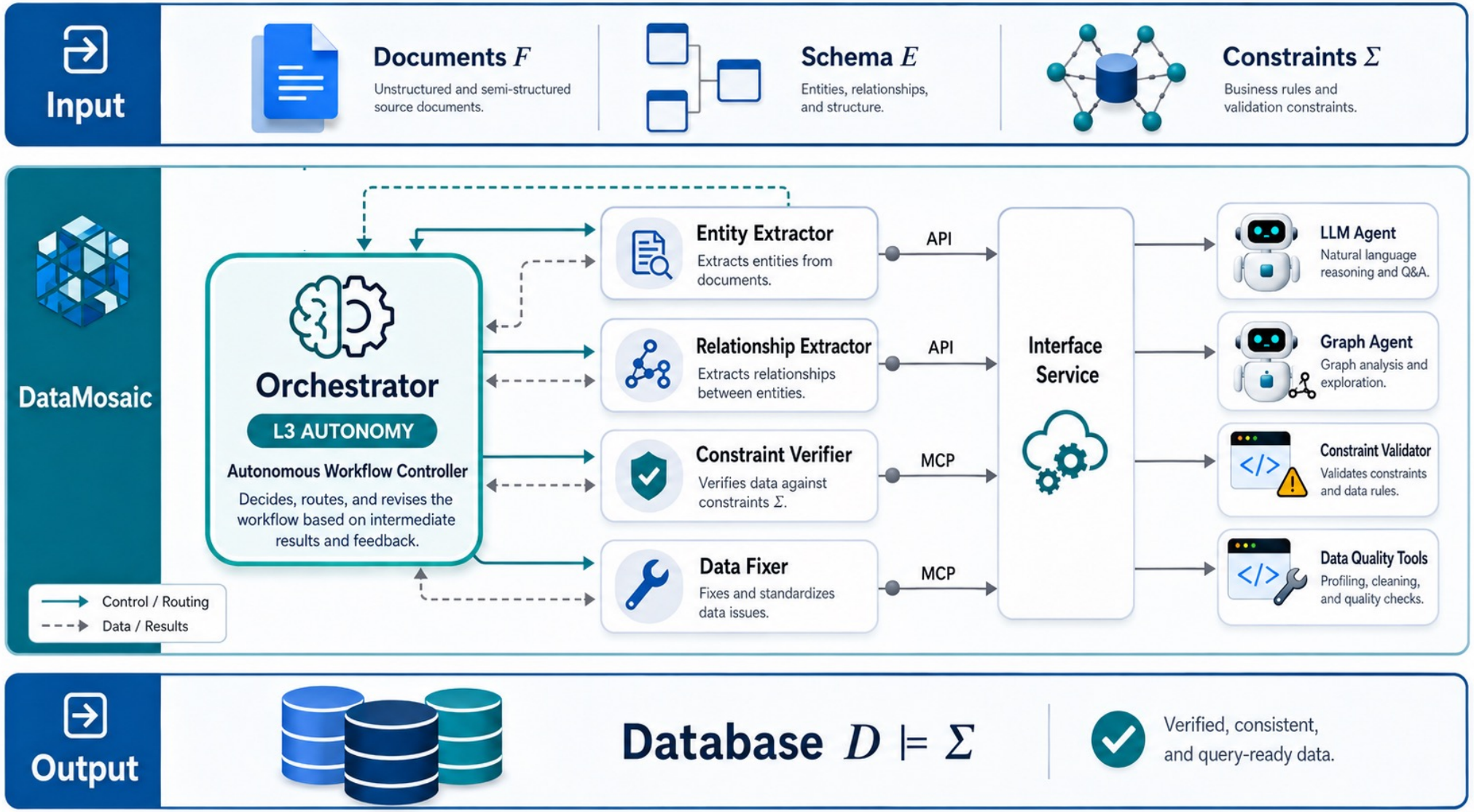
Red = spurious; Orange = missing

investor_cid	investee_cid
1	2
2	1
1	3
2	3

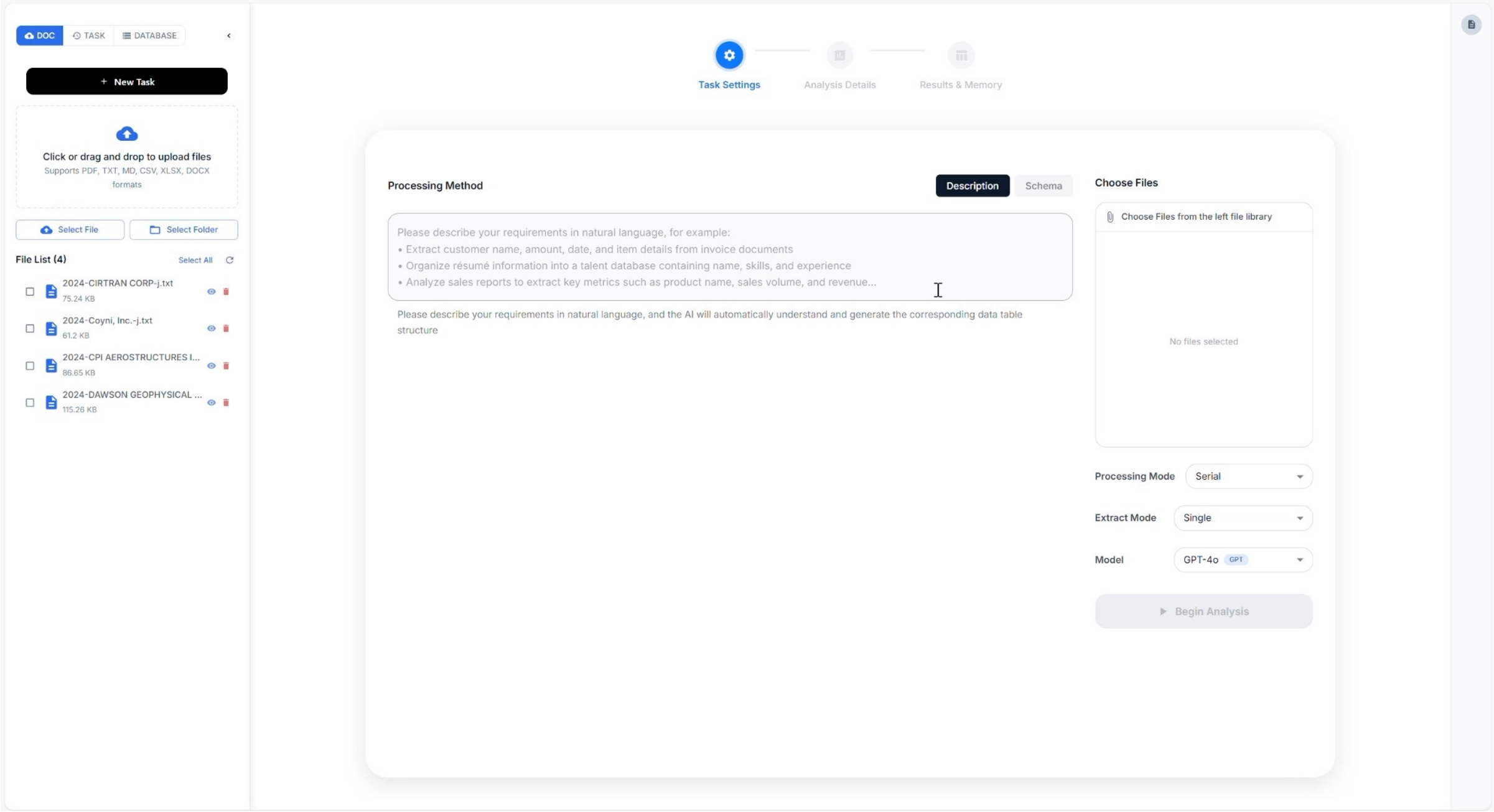
Input: Heterogeneous police records → **Extract:** Key entities, attributes, and relations → **Aggregate:** Across documents to build structured data → **Analyze:** Identify patterns and generate summaries

Document-to-Database: Extraction Meets Relational Semantics (VLDB 2026)

L1 · suggest | L2 · execute | L3 · orchestrate | L4 · proactive



DataMosaic (VLDB 2026 Demo)



Semantic Operators: L3

Constraint-guided Doc2DB Construction

DataMosaic

It has a **central orchestrator** that coordinates entity extraction, relationship extraction, constraint verification, repair, and targeted re-extraction inside a closed extract–verify–iterate loop

Slogan

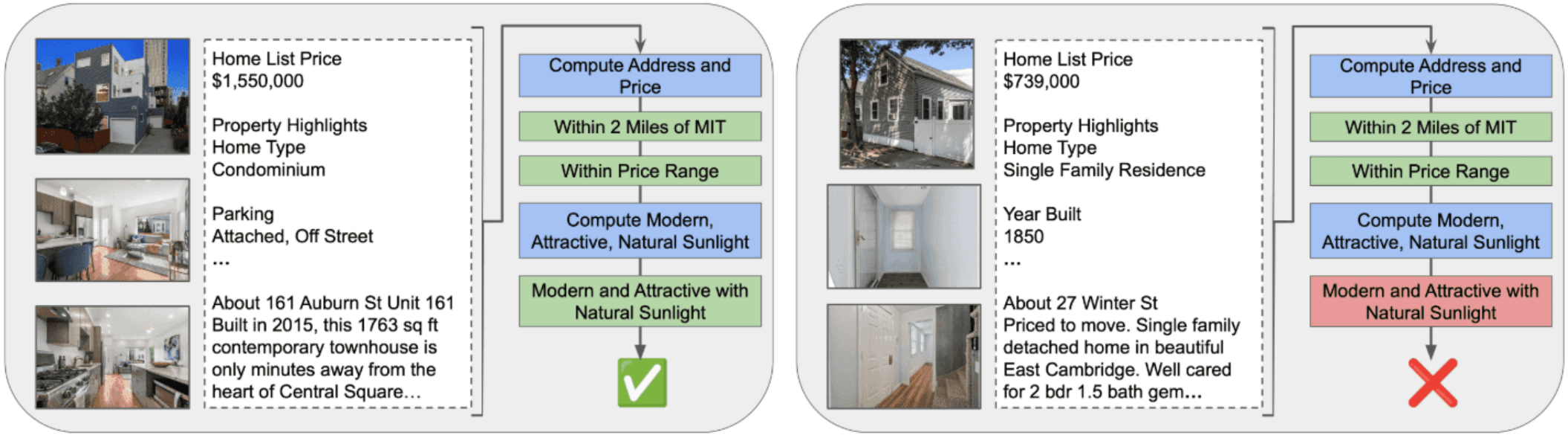
Not just extract tables – construct databases.

Limitation

Requires automated fact-checking, external knowledge integration, and automated constraint mining as future directions.

L1 · suggest | L2 · execute | L3 · orchestrate | L4 · proactive

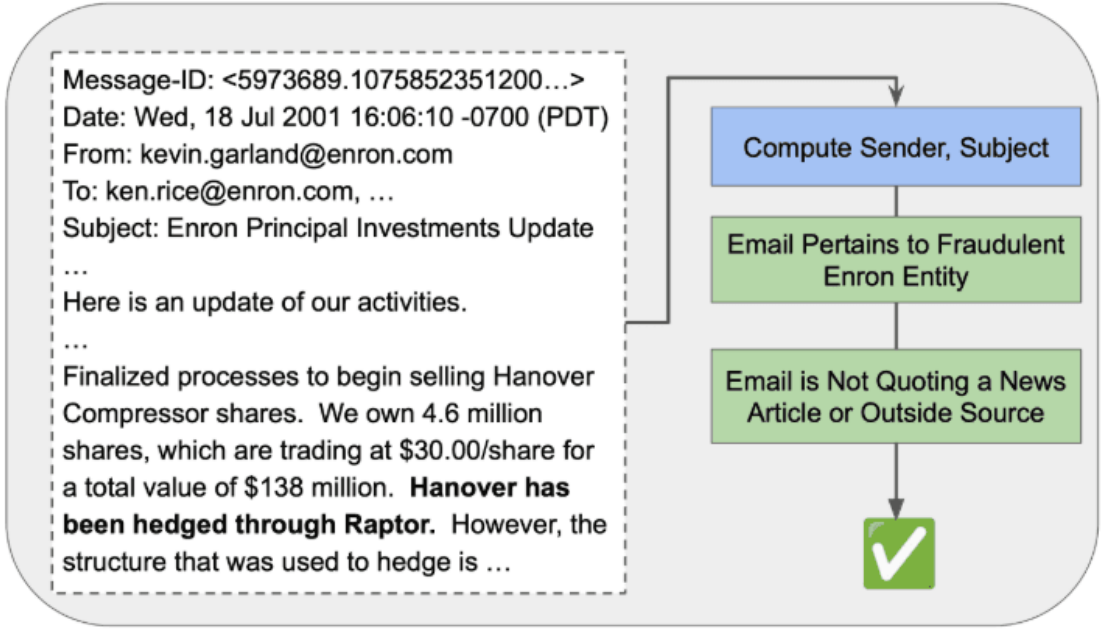
Real Estate Search: In this use case, a homebuyer wants to use online real estate listing data to find a place that is
(a) modern and attractive, and
(b) within two miles of work.

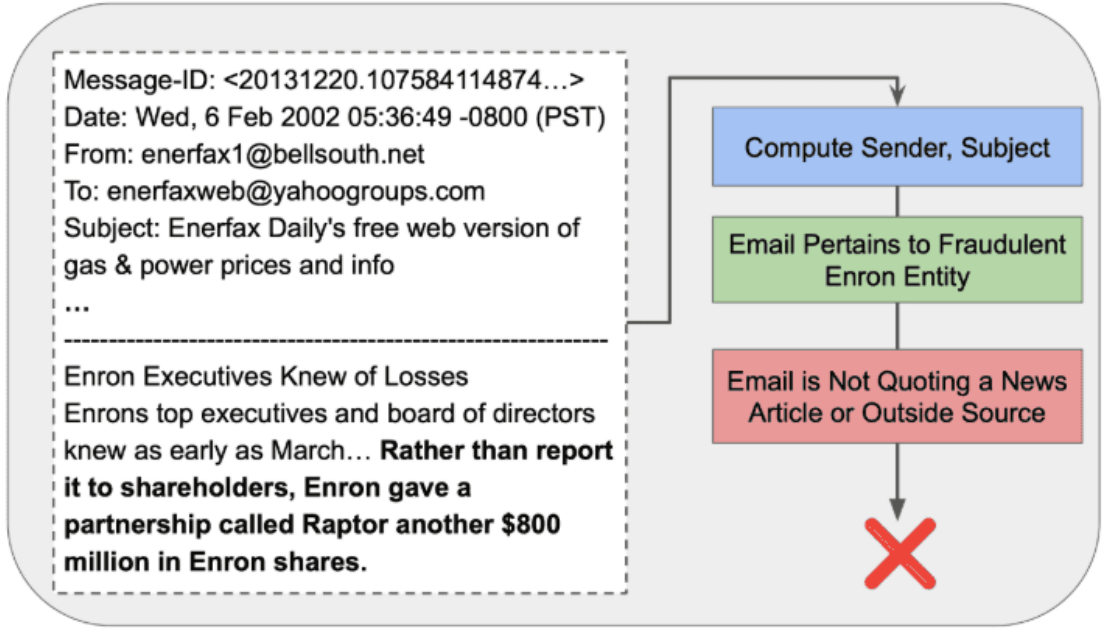


Disclaimer: the figure is from the Palimpzest paper

L1 · suggest | L2 · execute | L3 · orchestrate | L4 · proactive

Legal Discovery: In this use case, prosecutors conducting an investigation wish to identify emails from defendants which are
(a) related to corporate fraud (e.g., by mentioning a specific fraudulent investment vehicle)
(b) do not quote from a news article reporting on the business in question.





Disclaimer: the figure is from the Palimpzest paper

Palimpzest: Optimizing AI-Powered Analytics with Declarative Query Processing (CIDR 2025)

L1 · suggest | L2 · execute | **L3 · orchestrate** | L4 · proactive

Declarative Interface: You declare what you want; Palimpzest decides **how** to do it.

```
1  import palimpzest as pz
2
3  class Email(pz.TextFile):
4     """Represents an email, which can subclass a text file"""
5     sender = pz.StringField(desc="The email address of the sender", required=True)
6     subject = pz.StringField(desc="The subject of the email", required=True)
7
8  # define logical plan
9  emails = pz.Dataset(source="enron-emails", schema=Email)
10 emails = emails.filter("The email is not quoting from a news article or an article ...")
11 emails = emails.filter("The email refers to a fraudulent scheme (i.e., \"Raptor\", ...)")
12
13 # user specified policy
14 policy = pz.MinimizeCostAtFixedQuality(min_quality=0.8)
15
16 # execute plan
17 results = pz.Execute(emails, policy=policy)
```

1 Create a custom schema for Email

The programmer uses Palimpzest to define the structure (fields) of an `Email`.

2 Create a Dataset of Emails

- `source="enron-emails"` uniquely identifies a preregistered set of files.
- `schema=Email` instructs Palimpzest to transform raw data into the `Email` schema.
- The transformed results are stored in the `emails` Dataset.

3 Filter the Dataset

- Line 10: keep emails not quoting news articles.
- Line 11: keep emails discussing fraudulent investment entities.

4 Specify a policy

Describe how the system should choose among multiple possible implementations.
Here: `minimize cost` subject to a minimum quality (0.8).

5 Execute the program

Palimpzest will:

- Generate a logical execution plan
- Generate multiple optimized physical execution plan candidates
- Choose one according to the specified policy
- Execute the plan and yield results

→ Results

You focus on the what; Palimpzest handles the how.

Declarative — Describe data, rules, and goals.

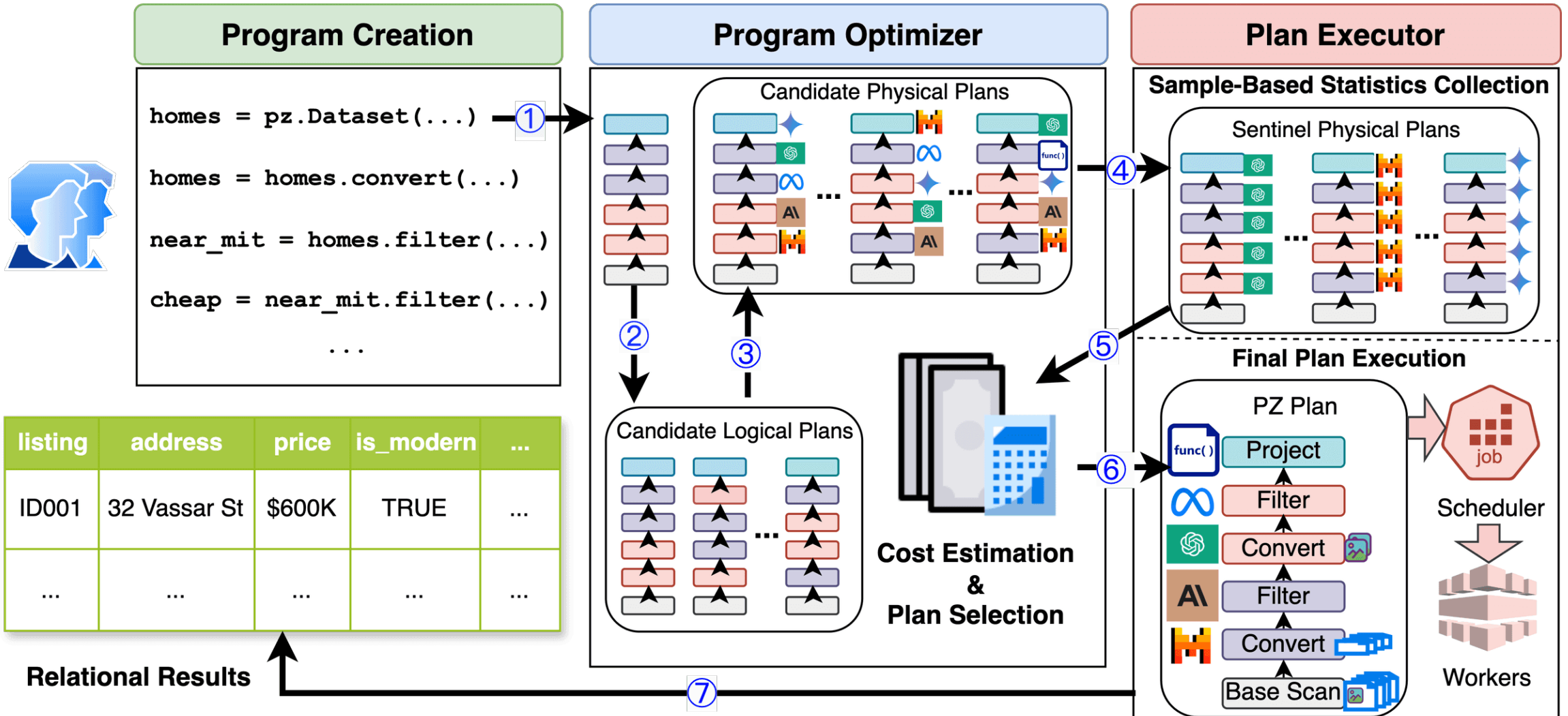
Optimized — Automatic plan generation and optimization.

Reliable — Constraints and policies ensure quality and correctness.

Efficient — Best plan chosen to minimize cost with required quality.

Palimpzest: Optimizing AI-Powered Analytics with Declarative Query Processing (CIDR 2025)

L1 · suggest | L2 · execute | **L3 · orchestrate** | L4 · proactive



Disclaimer: the figure is from the Palimpzest paper

Semantic Operators: L3

Constraint-guided Doc2DB Construction

DataMoasic

It has a **central orchestrator** that coordinates entity extraction, relationship extraction, constraint verification, repair, and targeted re-extraction inside a closed extract–verify–iterate loop

Slogan

Not just extract tables – construct databases.

Limitation

Requires automated fact-checking, external knowledge integration, and automated constraint mining as future directions.

Declarative Query Processing

Palimpzest

If you zoom into one Palimpzest operation, such as a semantic convert or filter, then that operator itself looks like L2: it executes an AI-powered transformation and may use model/prompt/implementation choices. But the Palimpzest paper as a whole is better classified as L3.

Slogan

From semantic operators to optimized AI pipelines.

Limitation

Does not yet discover the analysis goal, invent new operators, or autonomously manage the full end-to-end data workflow

Core Data Agent Challenges Across Autonomy Levels

Challenge \ Level	L1 Assistance	L2 Partial Autonomy	L3 Conditional Autonomy	L4 High Autonomy	L5 Full Autonomy
	Prompt-driven Data Assistant	Human-guided Workflow Executor	Agentic Workflow Orchestrator	Proactive Data Agent	Self-evolving Autonomous Data Agent
C1. Data-aware Workflow Orchestration & Cost-aware Execution	Human-designed workflow; agent suggests.	Execute human-designed pipelines with local feedback.	Auto-compose and optimize workflows / DAGs.	Proactively discover tasks and plan long-horizon execution.	Invent new operators, tools, and workflow paradigms.
C2. Long-horizon Agentic State, Memory & Skill Reuse	Stateless; no persistent memory.	Short-term task memory and execution feedback.	Workflow state + provenance + reusable skills / SOPs.	Long-lived cross-task memory and shared skill reuse.	Self-evolving knowledge and skill base.
C3. Semantic Grounding over Heterogeneous & Multimodal Data	Prompt / few-shot / RAG-based grounding.	Environment-aware schema, entity, and evidence grounding.	Semantic catalog + metadata + multimodal evidence alignment.	Continuous semantic maintenance over evolving data lakes.	Create new semantic abstraction and representations.

Progression Assist & Suggest » Execute & Improve » Orchestrate & Optimize » Proactively Act » Invent & Evolve

Outline

- Part I: Data Agents: Motivation, Definition, Autonomy Levels, and Core Challenges
- Part II: Towards Autonomous Data Agents: Key Challenges and Current Practices
  - Data-aware Workflow Orchestration & Cost-aware Execution
  - Long-horizon Agentic State, Memory & Skill Reuse
  - Semantic Grounding over Heterogeneous & Multimodal Data
- **Part III: Research Opportunities and Open Challenges**

Data Agent Opportunities in Data Lifecycle Management

① Data Generation → ② Data Transform → ③ Data Analytics → ④ Data Engineering → ⑤Data Usage → ⑥Data Insight

① OLTP

- Regression
  - Cardinality/Cost Estimation
- Online Optimization
  - Query Rewrite
  - Plan Selection
- Offline Optimization
  - Knob/Index/View Advisor
- Prediction
  - Workload
  - Resource
  - Data
- Database QAs
- Database Diagnosis
- SQL Dialect Translation

② Data Transformation

- Data Standardization
  - Automate Extract, Transform, Load
  - Auto schema mapping
  - Identifying patterns
  - Low code ETL
  - Predictive auto-scaling
  - Adaptive Transformation
- Multi-goal Optimization
  - Change data capture
  - Automate CDC
  - Predictive analytics
  - Reduce Human Cost

③ OLAP

- In-Database ML
  - In-Database Model
  - In-Database Vector
  - In-Database RAG
  - Anomaly Detection
  - Risk control analysis
- SQL+ML Analytics
  - TableQA
  - In-DB Semantics Analytics
- Autonomous Analytics
  - Autonomous workload management
  - Autonomous data format
  - Autonomous serverless

Data Agent Opportunities in Data Lifecycle Management

① Data Generation → ② Data Transform → ③ Data Analytics → ④ Data Engineering → ⑤Data Usage → ⑥Data Insight

④ Data Engineering

- Data Preparation
  - Data Discovery
  - Data Selection
  - Data Cleaning
  - Data Transformation
  - Data Integration
  - Data Generation
  - Data Mixing
  - Data Extraction
  - Data Labeling
  - Meta Data Manag.
- Data Flywheel
- Data Fabric

⑤ Descriptive Data Usage

- Prompting
  - Automation/Examples
- RAG
  - Multi-hop RAG
  - Graph RAG
  - Agentic RAG
- LLM Inference
  - Prefill/Decoder disagg.
  - KV cache
  - Scheduling
  - Data/Model/Resources Parallism
  - Quantization

⑥ Proactive Data Interpretation

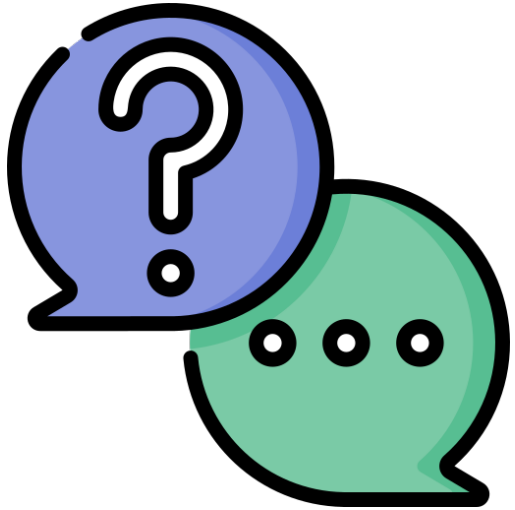
- Proactive Insights
  - Trends summarization
  - Insight discovery
  - Predictive decision making
  - Cognitive Analytics
- Proactive Data Agent
  - Chat2Data
  - Chat2BI
  - NL2Viz
  - In-DB Semantics Analytics
  - Unstructured data analytics
  - Multi-model data analytics on data lake

Conclusion

- **Data+AI is important for data management and analytics**
- **It urges the use of Data+AI techniques to revolutionize data systems**
  - Data science, Data Analytics, Data Lake
- **Data Agent is a promising direction for Database, Data, Data+AI**
  - Agent Orchestration and Scheduling
  - Multi-Agents Interaction
  - Agent Memory
  - Proactive Data Management
- **Open-source systems for Data Agent**

Thank you!

Repo: https://github.com/HKUSTDial/awesome-data-agents
Paper: https://arxiv.org/pdf/2510.23587



Any Questions?