(Neural ∘ Symbolic ∘ Neural)(Everything)

The Onion Framework for Modern Query Engines

Wenchao Bai

wbai@seu.edu.cn

September 29, 2025

Outline

Background: The Shifted Funding

Eugene Wu et al. [13] reported the shifted funding in US:

- **DB**: \$4B NSF expected budget for 2026.
- **AI**: $>$ \$100B VC funding in 2024, $>$ \$50B in Q1 2025.

Find the New Chalice of Data: Are We Polishing a Round Ball?

Where Does Academic Database Research Go From Here?

Continuing to improve RDBMS technology is helpful, but not necessarily competitive with industry, and the gains are increasingly marginal – we are **polishing a round ball**.

A promising direction is AI, but while AI is "an application of data," it doesn't seem like AI **needs** us.

— Eugene Wu, ACM SIGMOD Blog[1]

[1] https://wp.sigmod.org/?p=3801

Find the New Chalice of Data: Directions

Three points along a continuum in decreasing levels of ambition:

1. **Find the north star**: Does this problem matter to the world? And compared to all of industry and academia, is academic database research necessary to solve it?
2. **Constellation of capabilities**: Articulate *Essential Capabilities* that are missing today: without which applications simply do not and cannot otherwise exist.
3. **A sky full of stars**: <u>Evangelize declarative thinking to cultures throughout the world</u>, of which AI is just one culture. We must clarify what data- (or declarative-? scalable-?) thinking means[2].

[2] As emphasized in [13], core database principles is our competitive advantage, such as (1) "independence between physical and logical", (2) "declarativeness", and (3) "automatic scalability".

Find the New Chalice of Data: What About AI?

Potential Research Direction in AI.

What if we could query anything and everything in the world using LLMs?

- What does query execution and optimization on LLMs look like?
- How can LLMs aid optimizers?
- What if LLMs were access methods?
- ...

But, are we **uniquely positioned** to dominate this problem? (See: p. 42)

Positions of DB Researchers

Optimists:

- **Dan Suciu** (UW): "The people who need us, they know where to find us."
- **Joseph M. Hellerstein** (UC Berkeley): "The time spent debating the foolishness could be devoted to far more constructive purposes."[3]

Reformists:

- **Sihem Amer-Yahia** (CNRS): "... to reach out to colleagues in other **disciplines**."
- **Jens Dittrich** (Saarland Univ.): "We should work more on other topics like **usability**, revise interfaces (instead of performance)."

More positions in p. 43.

[3] https://jhellerstein.github.io/blog/sigmod-optimism/

Outline

Modern Query Engines

- **Assumption**: Closed-world[4] → Open-world (Generalization capabilities)
- **Capability**: Retrieval → Reasoning (Reasoning and planning capabilities)
- **Data object**: Relational data → Unstructured multi-modal data (Semantic understanding and extraction capabilities)
- **Query language**: SQL → Natural language (Usability and accessibility)

Refer to [3] for more information.

[4] **Closed-world assumption (CWA)**: A statement that is true is also known to be true. Therefore, conversely, what is not currently known to be true, is false.

Outline

Is LLM a Perfect Proxy?

Parameswaran et al. [5] studied prompt engineering from the perspective of declarative crowdsourcing. Specifically, they discussed the following questions:

1. Can we benefit from **fine-grained** prompting strategies?
2. How to benefit from the **composition** of prompting strategies?
3. How to benefit from processing a batch of **interrelated** tasks?
4. Can we reduce costs by leveraging **non-LLM** proxies?

Q1: Varying Prompting Strategies

Task 1: Semantic Ranking.

Given 20 ice-cream flavors, rank them by how "chocolatey" they are.

1. **Baseline**: List all the items in the prompt and ask LLM to rank them directly.
2. **Coarse-grained**: Rate each item and then sort them based on the ratings.
3. **Fine-grained**: Employ $O(N^2)$ pairwise comparisons to rank the items.

Evaluation: Fine-grained approach trades $\sim 100\times$ tokens for 20% improvement in Kendall Tau-β[5] over the baseline.

[5] **Kendall rank correlation coefficient**: a standard metric to compare rankings

Q2: Hybrid Coarse → Fine-Grained Prompting

Task 2: Alphabetical Sorting.[6]

Given a list of 100 random English words, sort them in alphabetical order.

1. **Baseline**: List all the items in the prompt and ask LLM to sort them directly.
2. **Hybrid strategy**: (a) Ask the LLM to sort the entire list; (b) Drop all hallucinated words; (c) Reinsert the missing words.

Evaluation: The hybrid strategy trades $O(kN)$ additional LLM calls (for k missing words and N partially sorted words) to eliminate hallucinated and missing words.

[6] This example illustrates error mitigation in fine-grained approaches, though alphabetical sorting represents a deterministic task where conventional algorithms would be more appropriate than LLMs.

Q3: Ensuring Internal Consistency

Task 3: Entity Resolution.

Identify all same citations on the DBLP-Google Scholar dataset.

1. **Baseline**: Employ $O(N^2)$ pairwise LLM comparisons for entity resolution.
2. *k*-**NN**: (a) Retrieve k nearest neighbors per citation; (b) Execute batch identification within neighborhood clusters for each citation pair ($2k+2$ citations per LLM-call); (c) Compute transitive closure[7] over identified equivalence pairs.

Evaluation: The baseline exhibits high precision (95.2%) but limited recall (50.3%). Applying transitive closure with 2-NN improves recall to 59.3% while maintaining acceptable precision (92.3%).

[7] *e.g.*, the transitive closure of $\{(1,2),(2,3)\}$ is $\{(1,2),(2,3),(1,3)\}$

Q4: Leveraging LLM and non-LLM Approaches

Missing Value Imputation.

Given an entity with j attributes $A = \{a_1, \ldots, a_j\}$ and values $E = \{e_1, \ldots, e_j\}$ where e_j is missing, predict e_j based on the known values.

1. **Baseline**: Employ LLM to predict e_j based on A and E directly.
2. k-**NN**[8]: (a) Retrieve k nearest neighbors of the entity; (b) Impute by k-NN if all neighbors have the same value for a_j; (c) Otherwise, impute by the LLM.

Evaluation: (a) The baseline suffers from the misalignment with ground truth (*e.g.*, "Elgato Systems" instead of "Elgato";) (b) The hybrid approach (LLM+k-NN) reduces token consumption by 50% with acceptable accuracy degradation (92.31%→87.69%.)

[8] Besides non-LLM methods, model cascade [14] is an alternative yet more flexible solution.

Takeaways

Q1: Rather than trying to accomplish the entire objective via a single task, it is beneficial to explore other task types, especially to maximize accuracy.

Q2: Employing hybrid strategies, with coarse-grained tasks first, followed by fine-grained ones, can lead to low cost and high accuracy overall.

Q3: Fixing erroneous LLM responses based on evidence from other responses can be an effective way to improve accuracy.

Q4: Leveraging a non-LLM proxy can help substantially reduce costs while keeping accuracy similar.

Insight: Identifying and reconstructing building blocks of LLM-powered query tasks yields benefits in both effectiveness and efficiency.

Outline

Symbolic ∘ Neural: Steering LLM-Powered Queries

- **Control the logic flow**: Fine-grained design under declarative frameworks[9].
  - Pandas-like: LOTUS [6], Palimpzest [4]
  - SQL-like: UQE [2], ELEET [8]
- **Control the budget**: Reduce LLM calls while maintaining acceptable quality.
  - Model cascade: BARGAIN [14]
  - Approximation: LOTUS [6], UQE [2]
  - Small language models (SLMs): ELEET [8]
  - Cost-based optimization: Palimpzest [4]
- **Control the quality**: Ensure intermediate and final results meet expectations.
  - Assertion synthesis: SPADE [7]

[9] **Declarative** means the focus is "what we want" instead of "how to implement". For example, SQL is a typical declarative language whereas C is a typical **imperative** language.

Declarative Frameworks (1/2): Palimpzest

```
1  import palimpzest as pz
2
3  class Email(pz.TextFile):
4    """Represents an email, subclass of 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 and plan execution
14 policy = pz.MinimizeCostAtFixedQuality(min_quality=0.8)
15 results = pz.Execute(emails, policy=policy)
```

Figure: Running example of Palimpzest.

1. Define the data schema (line 3-6.)
2. Declare the data source (line 9.)
3. Filter the data using semantic operator "filter" (line 10-11.)
4. Define the execution policy and execute the query (line 14-15.)

Declarative Frameworks (2/2): ELEET

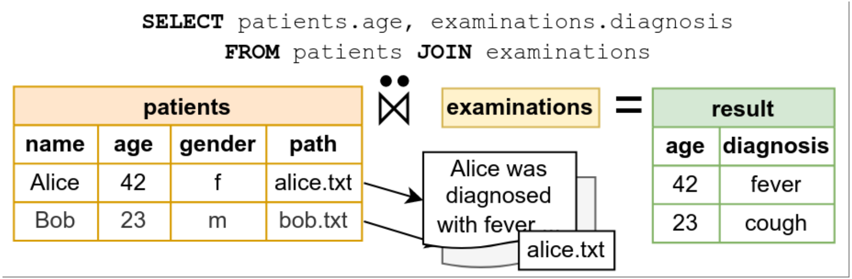

Figure: Example of a query that executes a multi-modal join between a patient table and examination reports. ELEET analyzes the texts and extracts values for each queried attribute, such as the diagnosis from each examination report.

Efficiency Optimization (1/5): Model Cascade

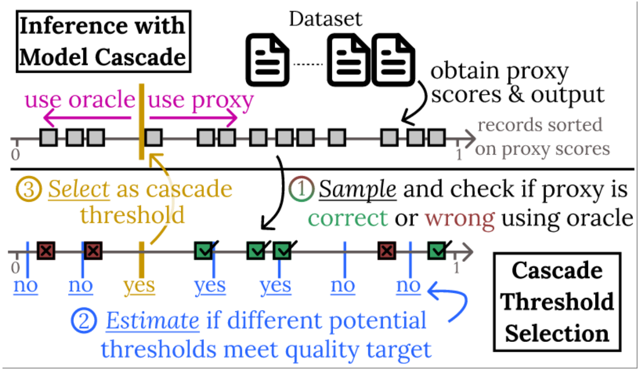

Figure: Overview of model cascade.

- **Intuition/Assumption**: High-confidence output from a small model is likely to be correct.
- BARGAIN [14] provides tight theoretical guarantees through task and data-aware sampling, estimation, and threshold selection.

Efficiency Optimization (2/5): Proxy-Based Approximation

- **Intuition**: Use a **fast-but-imperfect** approximate proxy to handle easy cases, reserving the **slow-but-accurate** model only for hard decisions.
- Examples of LOTUS [6]:
  - **Filter**: Use embedding-based classifier or distilled LLMs to filter out obvious matches/mismatches.
  - **Join**: Use embedding-based similarity to filter tuple pairs.

- **Limitations**:
  - Optimization degree is low; cannot optimize at the level of plan structure.
  - Inappropriate adoption of approximation methods results in low accuracy

Efficiency Optimization (3/5): Approximation for Aggregation Queries.

- **Intuition**: Aggregation queries can be accelerated by reducing the amount of data processed by LLMs
- UQE [2] adopts **unbiased** stratified sampling for accelerating **aggregation** queries:
  - Embed all rows and cluster them into K groups.
  - Perform stratified sampling within clusters to select a small number of rows.
  - Use weighted averaging of sampled results to unbiasedly estimate aggregation queries
- **Limitation**: This method is not universal, only support aggregation queries.

Efficiency Optimization (4/5): Small Language Models

- **Intuitions**: (1) SLMs are more efficient than LLMs, ensuring efficient online extraction; (2) Information in tables can help locate structured information in text.
- **Limitations**: (1) Cannot support complex semantic analytics; (2) Lack of world knowledge; (3) Impractical assumption: attributes in text are known.

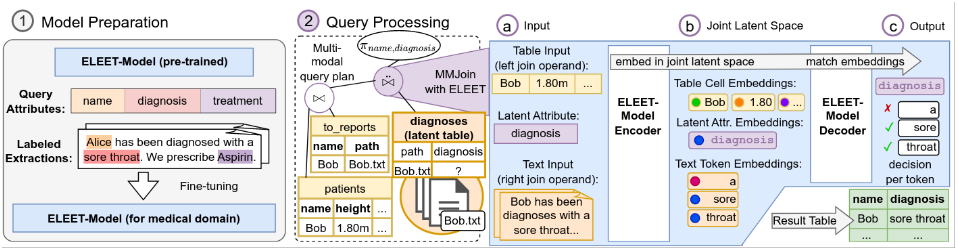

Figure: Overview of ELEET [9].

Efficiency Optimization (5/5): Cost-Based Optimization

- **Intuition/Assumption**: If operators are independent, we can compose operators estimations to estimate plan performance (to avoid exponential searching space.)
- Method of Palimpzest [4]:
  - Executes a set of plans on a small set of **sampled data**.
  - Obtain per-operator estimates (*e.g.*, distribution of runtimes, per-record cost and quality of each operator.)
  - Estimate performance of each plan by composing its per-operator estimates.

- **Limitation**: Estimation by executing over sampled data is time-consuming and inaccurate, which limits optimization effectiveness

Quality Control (1/2): SPADE

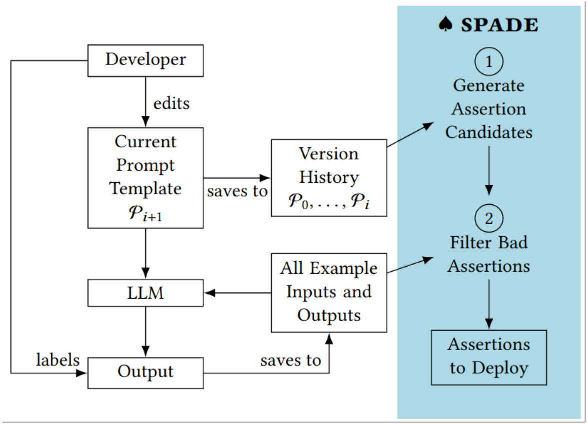

Figure: Workflow of SPADE.

- **Motivation**: Monitor the data quality through LLM pipelins.
- **Intuition**: We can mine prompt version histories to identify assertion criteria for LLM pipelines [7].
- **Objective**: Minimal set of assertions with qualified coverage and False Failure Rate.
- **Limitations**: (1) Only focus on single-prompt pipelines; (2) lack of labeled data; (3) LLM dependencies.

Quality Control (2/2): Delta-Driven Assertion Synthesis

Version i	$\Delta\mathcal{P}_i$	$\Delta\mathcal{P}_i$ **Category**	**Possible New Assertion Criteria**
1	+ Given the following information about the user, {personal_info}, and information about a movie, {movie_info}: write a personalized note for why the user should watch this movie.	Inclusion	Response should be personalized and relevant to the given user information
2	+ Include elements from the movie's genre, cast, and themes that align with the user's interests.	Inclusion	Response includes specific references to the user's interests related to the movie's genre, cast, and themes
3	+ Ensure the recommendation note is concise.	Qualitative Assessment	Response should be concise
4	- Ensure the recommendation note is concise. + Ensure the recommendation note is concise, not exceeding 100 words.	Count	Response should be within the 100 word limit
5	- Include elements from the movie's genre, cast, and themes that align with the user's interests. + Mention the movie's genre and any shared cast members between the {movie_name} and other movies the user has watched.	Inclusion	Response should mention genre and verify cast members are accurate
6	+ Mention any awards or critical acclaim received by movie_name.	Inclusion	Response should include references to awards or critical acclaim of the movie
7	+ Do not mention anything related to the user's race, ethnicity, or any other sensitive attributes.	Exclusion	Response should not include references to sensitive personal attributes

Figure: Comparison of 7 prompt versions for an LLM pipeline to write personalized movie recommendations.

Takeaways

- Model LLM queries under declarative frameworks make it easier to control the logic flow, execution budget, and quality.
- Model cascade method has tight theoretical guarantees.
- Proxy-based approximation may influence accuracy of the results.
- Approximate processing is not universal, only support aggregation queries.
- Cost-based optimization directly relies on the accuracy of cost estimation.

Caveat: These approaches require user-specified logic flow which introduce additional **human cost**.

Outline

Neural ∘ Symbolic ∘ Neural: Manipulate Symbols Automatically

To improve the accessibility, several challenges should be addressed:

- **How to align the natural language with the query language?**
  - NL2SQL: TAG [1] (Tabular data), CAESURA [9] (Multi-modal data)
  - NL to self-defined operators: Unify [10], iDataLake [12], AOP [11]
  - Planning & tool usage: CAESURA [9], AOP [11]
- **How to optimize the efficiency of the generated query plan?**
  - Independent parallelism: AOP [11]
  - Cost-based optimization: Unify [10]
  - Fault tolerance: iDataLake [12]

Convert Natural Language to Query Languages

- **Objective**: Find a convertion function f: (NL Query, Operators, Tools, *etc.*) $\rightarrow$ Query Language, that is logically correct.
- **Solution 1**: Instruct LLMs to generate the query plan.
  - Straightforward yet could be inaccurate (depends on LLM capabilities).
- **Solution 2**: Progressive matching [10].
  - Pros: More robustness and exaplainable.
  - Cons: Inflexibility of operaotrs.

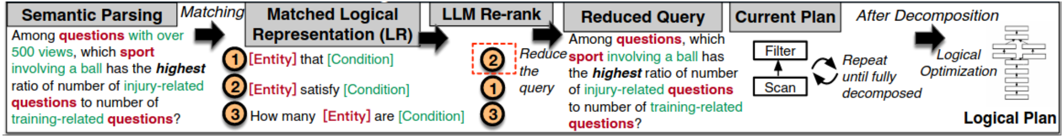

Figure: Logical plan generation in Unify via progressive matching.

Optimize the Execution Plan (1/3): Independent Parallelism

- **Intuition**: (1) Identifying and parallelizing independent operations can significantly reduce execution time.
- **Method of AOP** [11]: (1) Instruct LLMs to generate pipelines; (2) Optimize pipelines into DAG; (3) Combine different pipelines; (4) Layer-wise execution.

- **Limitation**: The quality of the generated plans rely on LLM capabilities.

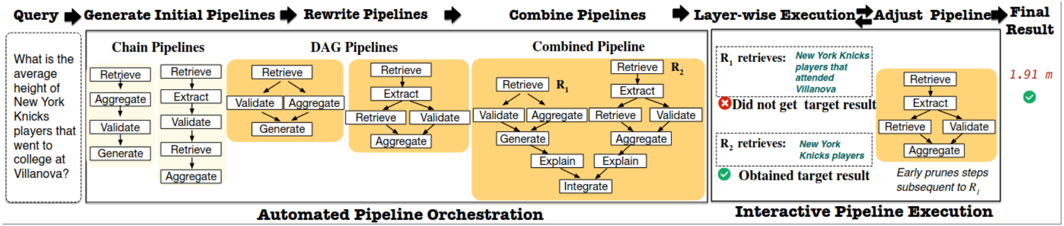

Figure: Framework of AOP.

Optimize the Execution Plan (2/3): Cost-Based Optimization

- **Observation**: Data points satisfying the query often have high semantic relevance with the query.
- **Mathod of Unify** [10]: (1) Embed records; (2) Retrieve query-related record samples via importance sampling; (3) Estimate the cardinality of query results based on the samples; (4) Optimize the execution plan based on the estimates.

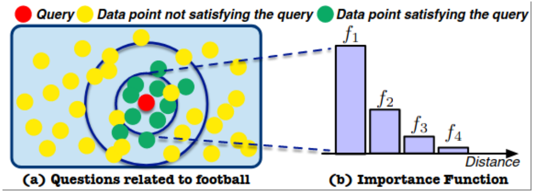

Figure: Importance sampling in Unify.

Optimize the Execution Plan (3/3): Online Plan Adjustment

- **Intuition**: When the execution fails, it's usually beneficial to restore from the checkpoint and switch to other other low-cost plans rather than starting over.

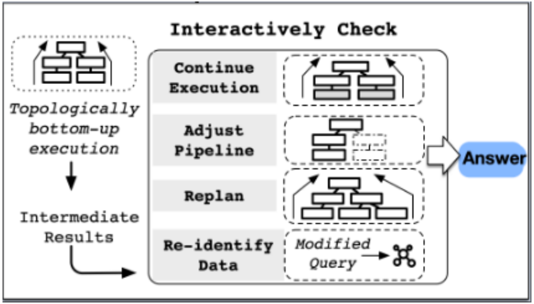

Figure: Pipeline execution in iDataLake [12].

Takeaways

- Directly instructing LLMs to generate pipeline achieves **limited accuracy**.
- Progressively matching appropriate operators is limited by **inflexibility** of operaotrs, strict requirement of intput/output relationship of operators.
- Optimization strategies: (1) Parallelizing independent operations; (2) Cost-based optimizations; and (3) Online plan adjustment.

Discussion: Where can we go from here?

References I

Asim Biswal, Liana Patel, Siddarth Jha, Amog Kamsetty, Shu Liu, Joseph E Gonzalez, Carlos Guestrin, and Matei Zaharia.
Text2sql is not enough: Unifying ai and databases with tag.
In CIDR, 2025.

Hanjun Dai, Bethany Wang, Xingchen Wan, Bo Dai, Sherry Yang, Azade Nova, Pengcheng Yin, Mangpo Phothilimthana, Charles Sutton, and Dale Schuurmans.
Uqe: A query engine for unstructured databases.
NeurIPS, 37:29807–29838, 2024.

Guoliang Li, Jiayi Wang, Chenyang Zhang, and Jiannan Wang.
Data+ai: Llm4data and data4llm.
In SIGMOD Tutorials, page 837–843, 2025.

References II

- Chunwei Liu, Matthew Russo, Michael Cafarella, Lei Cao, Peter Baile Chen, Zui Chen, Michael Franklin, Tim Kraska, Samuel Madden, Rana Shahout, et al.
  Palimpzest: Optimizing ai-powered analytics with declarative query processing.
  In CIDR, 2025.

- Aditya Parameswaran, Shreya Shankar, Parth Asawa, Naman Jain, and Yujie Wang.
  Revisiting prompt engineering via declarative crowdsourcing.
  In CIDR, 2024.

References III

Liana Patel, Siddharth Jha, Melissa Pan, Harshit Gupta, Parth Asawa, Carlos Guestrin, and Matei Zaharia.
Semantic operators: A declarative model for rich, ai-based data processing.
PVLDB, page 4171–4184, 2025.

Shreya Shankar, Haotian Li, Parth Asawa, Madelon Hulsebos, Yiming Lin, JD Zamfirescu-Pereira, Harrison Chase, Will Fu-Hinthorn, Aditya G Parameswaran, and Eugene Wu.
Spade: Synthesizing data quality assertions for large language model pipelines.
PVLDB, 17(12):4173–4186, 2024.

References IV

- Matthias Urban and Carsten Binnig.
  Eleet: Efficient learned query execution over text and tables.
  PVLDB, 17(13):4867–4880, 2024.
- Matthias Urban and Carsten Binnig.
  Caesura: language models as multi-modal query planners.
  In CIDR, 2025.
- Jiayi Wang and Jianhua Feng.
  Unify: An unstructured data analytics system.
  In ICDE, pages 4662–4674, 2025.

References V

Jiayi Wang and Guoliang Li.
Aop: Automated and interactive llm pipeline orchestration for answering complex queries.
In CIDR, 2025.

Jiayi Wang, Guoliang Li, and Jianhua Feng.
idatalake: An llm-powered analytics system on data lakes.
Data Engineering, page 57, 2025.

Eugene Wu and Raul Castro Fernandez.
Where does academic database research go from here?, 2025.

References VI

Sepanta Zeighami, Shreya Shankar, and Aditya Parameswaran.
Cut costs, not accuracy: Llm-powered data processing with guarantees.
PACMMOD, 2026.

A: Does DB Research Dominate LLM-Querying?

TL;DR: Yes and No.

- **Our advantages**. Declarativeness, cost-based optimization, approximation guarantees, and performance optimization. They can do RAG but we can do it better.
- **Poor predicatability**. LLMs as a compute substrate evolve weekly, meaning that any optimization rules based on yesterday's trade-offs are immediately slower, more expensive, or lower quality. (We can only follow LLM companies.)
- **Impossible to reason about**. The responses change for no reason, their tunable parameter is "any text." (Common task with the AI community.)
- **Undefined correctness**. There are no semantics to create correctness benchmarks, only vibes. (Common task with the AI community.)

Similar reasoning should be applied to data preprocessing for AI, video querying, provenance for AI, NL2SQL, ML training and serving, prompt engineering, etc.

B: More Positions for the Panel

- **Pınar Tözün** (IT Univ. of Copenhagen): "I think there are 'Systems for ML' aspects that our community is well-positioned to take on."
- **Leilani Battle** (UW): "(Consider) Not just 'what can we do?' but also 'what should we do?'(*e.g.*, ethical issues, software abusiveness, etc.)"
- **Aditya Parameswaran** (UC Berkeley): "(We need to) fully embrace LLMs as a means to solve the AI-complete problems, *e.g.*, data integration, data cleaning."
- **Sudeepa Roy** (Duke Univ.): "(We should) be open to embrace new ideas and take the opportunity to start new collaborations in the fast evolving landscape of AI."
- **Samuel Madden** (MIT): "Our community still has a role to play in the new AI era."
- **Felix Naumann** (HPI): "The traditional problem space of all things data management is alive and kicking."
- **Paolo Papotti** (EURECOM): "One risk is the limited openness to new topics."
- **James Cowling** (Convex): "The huge gift academia has is the ability to investigate impractical ideas."