

FinDS 2026

AI in Data Lakes

Rihan Hai

Self-introduction

Data systems for

AI

Diffusion models



Aditya Shankar

Research-12

Day: Tuesday
Time: 3:30 PM - 5:00 PM

LLMs

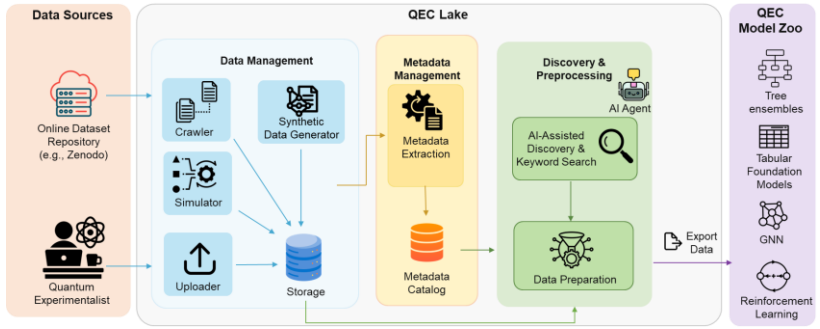


Wenbo Sun

Research-35

Day: Thursday
Time: 11:00 AM - 12:30 PM

Quantum



Why this matters in finance



Data sources

Transactions
Market feeds
Customer profiles
Documents and reports
Logs, alerts, graph signals

AI tasks

Explain risk alerts
Detect fraud
Forecast liquidity
Impute missing signals
Generate synthetic scenarios

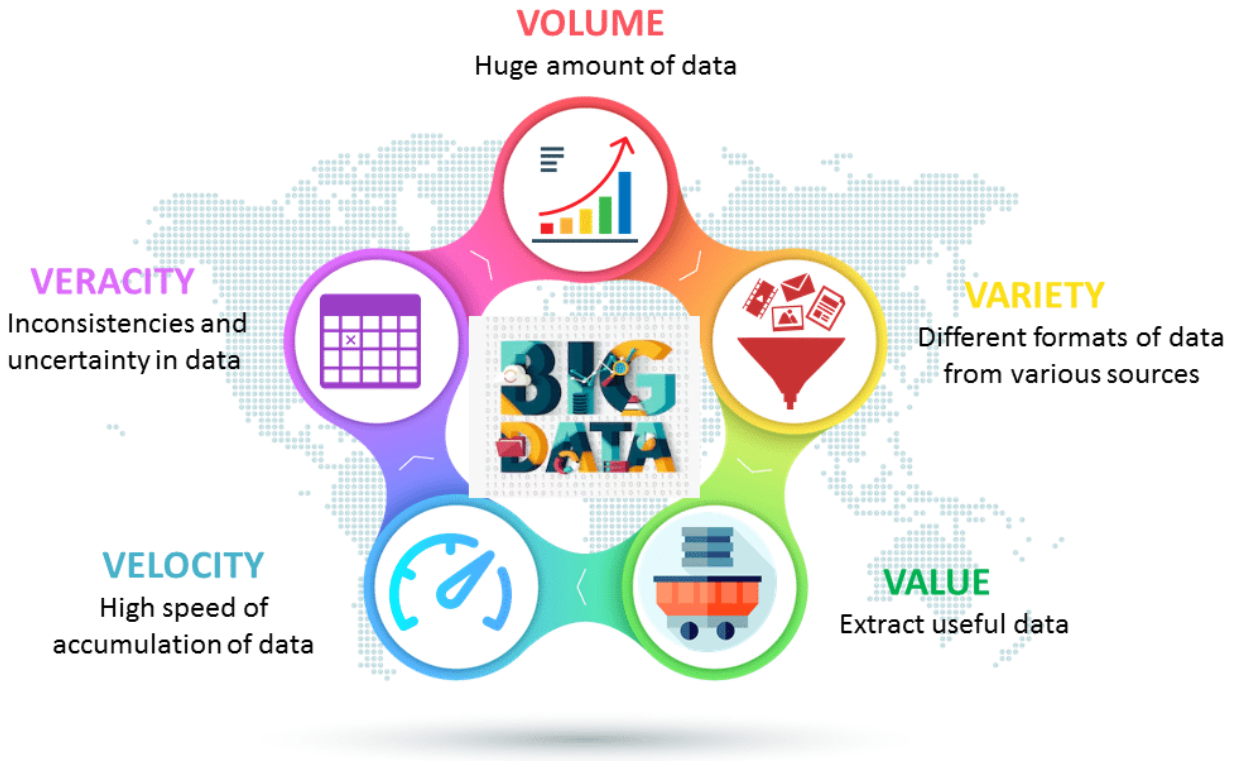
System constraints

Privacy
Low-latency inference
Audit
Data quality
Regulatory control

Part I

Data lakes: from storage to AI platform

The key concepts of Big Data



https://iclerisy.com/what-is-big-data/

Data lake Definition

A data lake is a flexible, scalable storage and management system where raw data from heterogeneous sources can be **ingested in original format** and **queried on the fly**.

IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 35, NO. 12, DECEMBER 2023 12571

Data Lakes: A Survey of Functions and Systems

Rihan Hai, *Member, IEEE*, Christos Koutras, Christoph Quix, and Matthias Jarke, *Senior Member, IEEE*

(Survey Paper)

Data warehouse vs. data lakes

Criteria	**Data Warehouses**	**Data Lakes**
Data ingestion	ETL	Load-as-is
Data format	Structured	Heterogeneous (structured, semi-structured, and unstructured)
Data storage	Relational databases	Hadoop, Relational databases, NoSQL data stores, etc
Data access	SQL queries (OLTP, OLAP)	Different query languages (e.g., SQL, Cypher), programming languages (e.g., Java, Python, R)

A brief history of data lakes

- 2010–2013: Beginnings

If you think of a datamart as a store of bottled water – cleansed and packaged and structured for easy consumption – the data lake is a large body of water in a more natural state. The contents of the data lake stream in from a source to fill the lake, and various users of the lake can come to examine, dive in, or take samples.

-- Pentaho CTO James Dixon



A brief history of data lakes

- 2014–2015: Criticisms and further development

A data swamp if

- No metadata
- No data governance



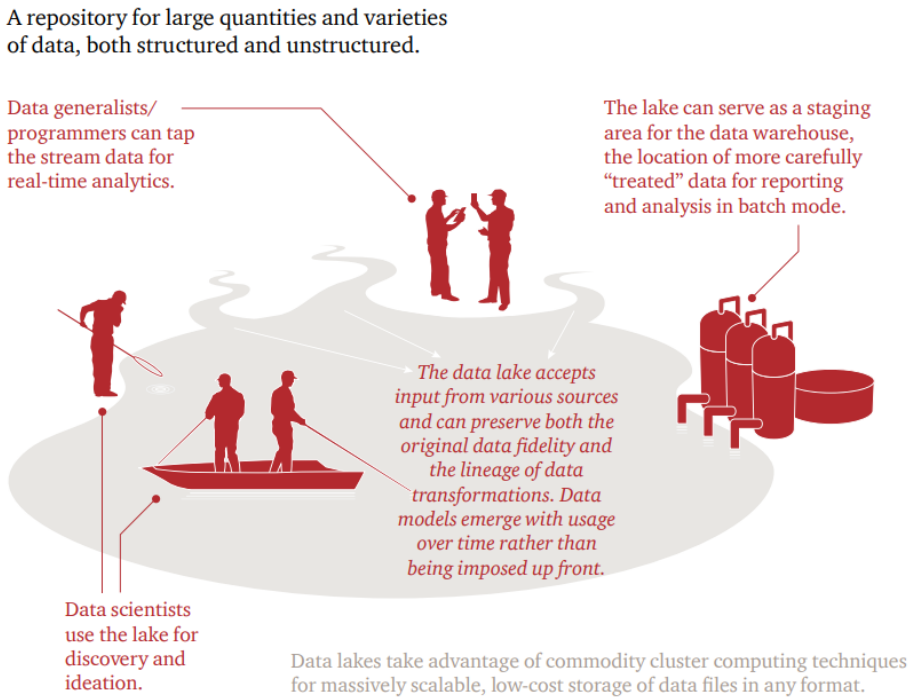
Data Lakes Revisited

- Data lake should also be equipped with metadata and governance
- Data lake could enable ad-hoc data analytics

Data Lakes Revisited | James Dixon's Blog (wordpress.com)

A brief history of data lakes

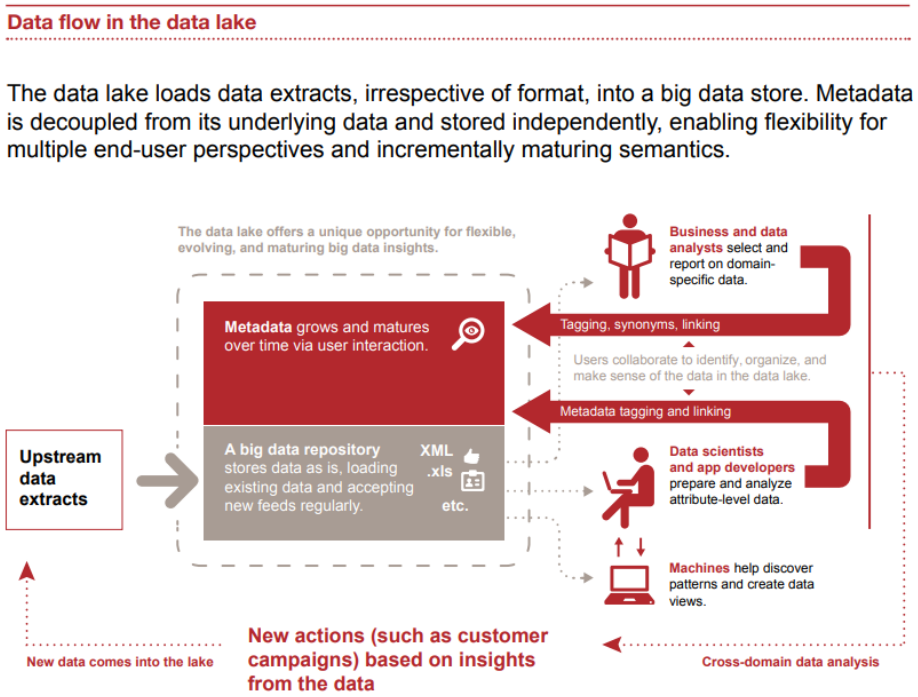
- 2014–2015: Criticisms and further development



B. Stein and A. Morrison. The enterprise data lake: Better integration and deeper analytics. PwC Technology Forecast: Rethinking integration, 1(1- 9):18, 2014

A brief history of data lakes

- 2014–2015: Criticisms and further development



B. Stein and A. Morrison. The enterprise data lake: Better integration and deeper analytics. PwC Technology Forecast: Rethinking integration, 1(1- 9):18, 2014

A brief history of data lakes

- 2016–present: Prosperity and diversity
  - Rising number of Vendors



  - Intensively studied research topic



Data Lake Management: Challenges and Opportunities

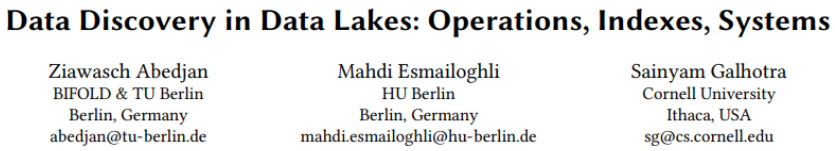
Fatemeh Nargesian
University of Toronto
fnargesian@cs.toronto.edu

Erkang Zhu
University of Toronto
ekzhu@cs.toronto.edu

Renée J. Miller
Northeastern University
miller@northeastern.edu

Ken Q. Pu
UOIT
ken.pu@uoit.ca

Patricia C. Arocena
TD Bank Group
prg@cs.toronto.edu



Data Discovery in Data Lakes: Operations, Indexes, Systems

Ziawasch Abedjan
BIFOLD & TU Berlin
Berlin, Germany
abedjan@tu-berlin.de

Mahdi Esmailoghli
HU Berlin
Berlin, Germany
mahdi.esmailoghli@hu-berlin.de

Sainyam Galhotra
Cornell University
Ithaca, USA
sg@cs.cornell.edu

A brief history of data lakes

- 2016–present: Prosperity and diversity

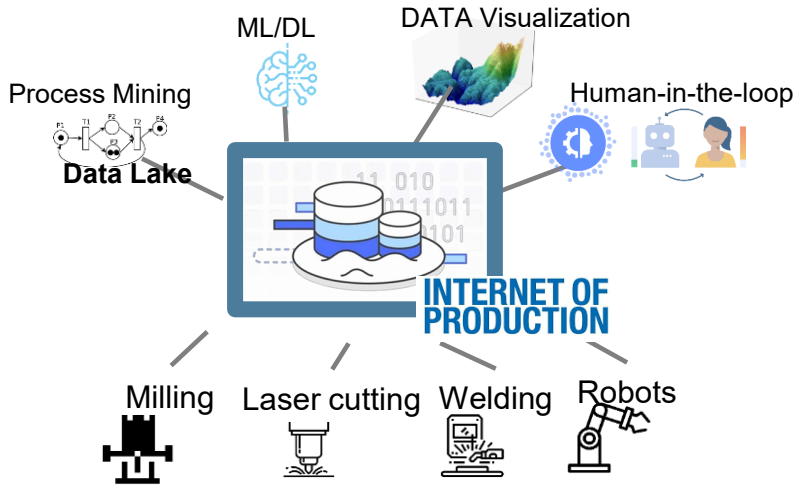
Q: How big is a data lake for various applications?

Data lake for COVID



A public data lake for analysis of COVID-19 data
https://aws.amazon.com/blogs/big-data/a-public-data-lake-for-analysis-of-covid-19-data/

Data lake for Industrial 4.0

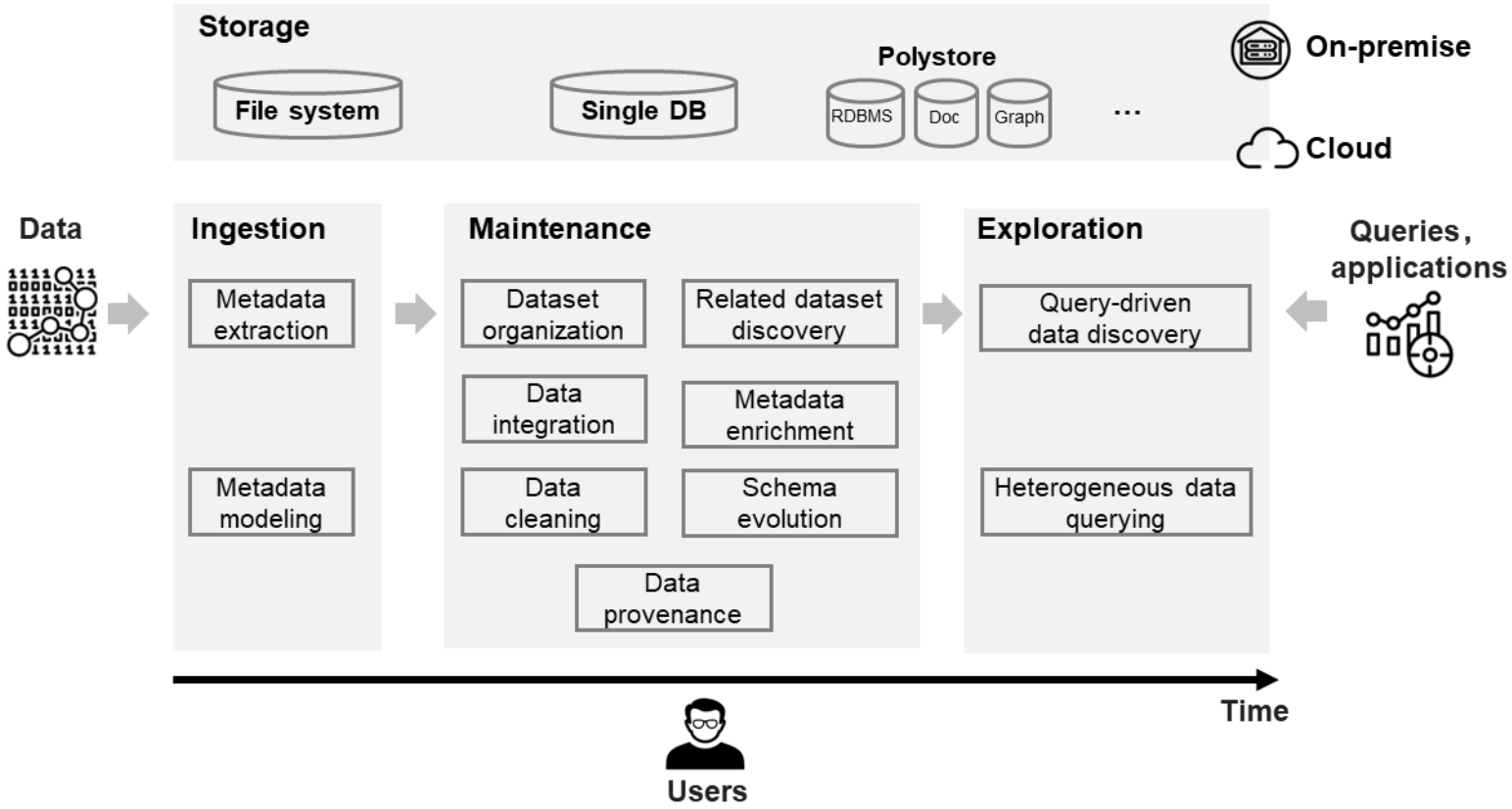


Google's ultimate data lake



Source: SIGMOD2020 Keynote

Function-based architecture



*Rihan Hai, Christos Koutras, Christoph Quix, and Matthias Jarke. Data lakes: A survey of functions and systems. **TKDE**, 35(12):12571–12590, 2023.*

Data lake systems overview

Tier	Functions	Systems
Ingestion	Metadata extraction	GEMMS [112]
		DATAMARAN [49]
		Skluma [129]
	Metadata modeling	GEMMS [61], [112]
		HANDLE [40]
		Data vault [54], [101]
		Diamantiniet al. [31], [32], [33]
		Aurum [45]
		Sawadogoet et al. [122]
Maintenance	Dataset organization	GOODS [64], [65]
		DS-Prox [4], [5], [6]
		KAYAK [85], [86]
		Nargesian et al. [98]
		Ronin [105]
		Juneau [143]
	Related dataset discovery	Aurum [45]
		Brackenbury et.al. [14]
		JOSIE [145]
		D^3L [13]
		Juneau [70], [142], [143]
		PEXESO [37]
		RNLIM [116]
		DLN [11]
	Data integration	Constance [58], [59], [60], [62]
	Metadata enrichment	CoreDB [8], [9]
		D^4 [104]
		DomainNet [79]
		Constance [61]
		GOODS [64], [65]
	Data cleaning	CLAMS [44]
		Constance [61]
		Song et al. [130]
	Schema evolution	Klettkeet et al. [75]
	Data provenance	IBM tool [134]
		Suriarachchi et al. [132]
		GOODS [64], [65]
		CoreDB [8], [9]
		Juneau [70], [142], [143]
Exploration	Query-driven data discovery	JOSIE [145]
		D^3L [13]
		Juneau [70], [142], [143]
		Aurum [45]
	Heterogeneous data querying	Constance [58], [62]
		CoreDB [8], [9]
		Ontario [41], [74]
		Squerall [89]

Rihan Hai, Christos Koutras, Christoph Quix, and Matthias Jarke. Data lakes: A survey of functions and systems. ***TKDE****, 35(12):12571–12590, 2023.*

Future directions from the survey

The survey already points to the gap between data lakes and AI workloads.

Training-data heterogeneity

Features, labels, logs, tables, text, and streams live in different forms.

In-lake machine learning

Move computation closer to data instead of repeatedly exporting data.

ML workflow optimization

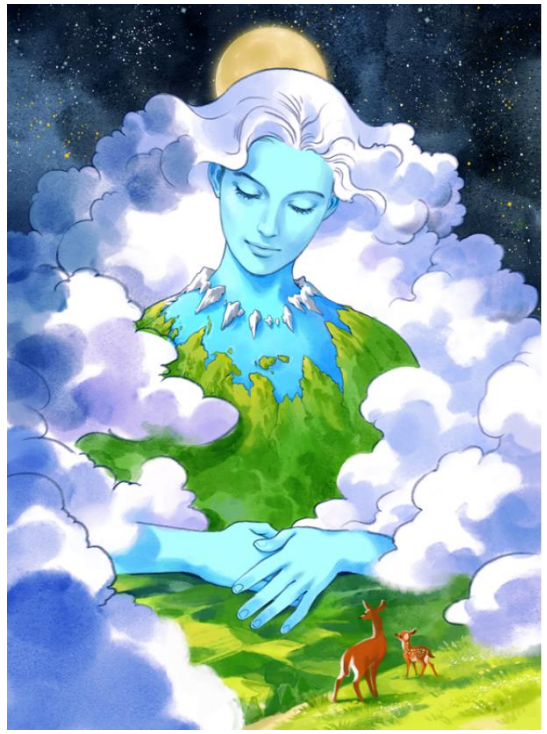
Optimize preprocessing, feature discovery, inference, and reuse.

ML-driven metadata

Use models to improve catalogs, discovery, lineage, and quality signals.

Today’s talk focuses on two concrete answers: in-lake LLM inference and conditional generative data operators.

In-lake machine learning



Source: https://www.behance.net/gallery/38395965/Mother-Earth

Amalur model lake



Data Lake

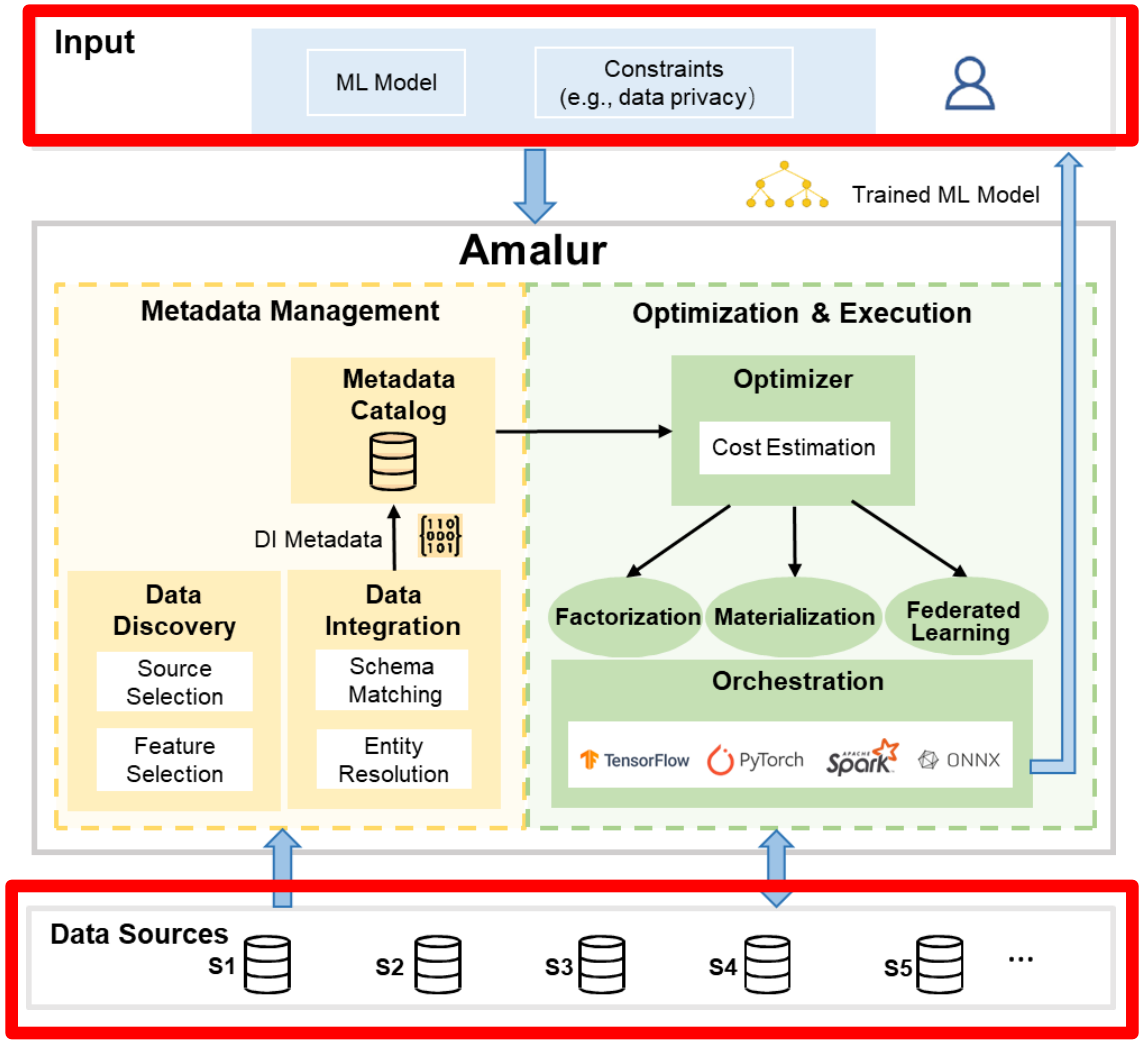


Model Zoo

*Rihan Hai, Christos Koutras, Andra Ionescu, Ziyu Li, Wenbo Sun, Jessie van Schijndel, Yan Kang, and Asterios Katsifodimos. Amalur: Data integration meets machine learning. In **ICDE**, pages 3729–3739, 2023.*

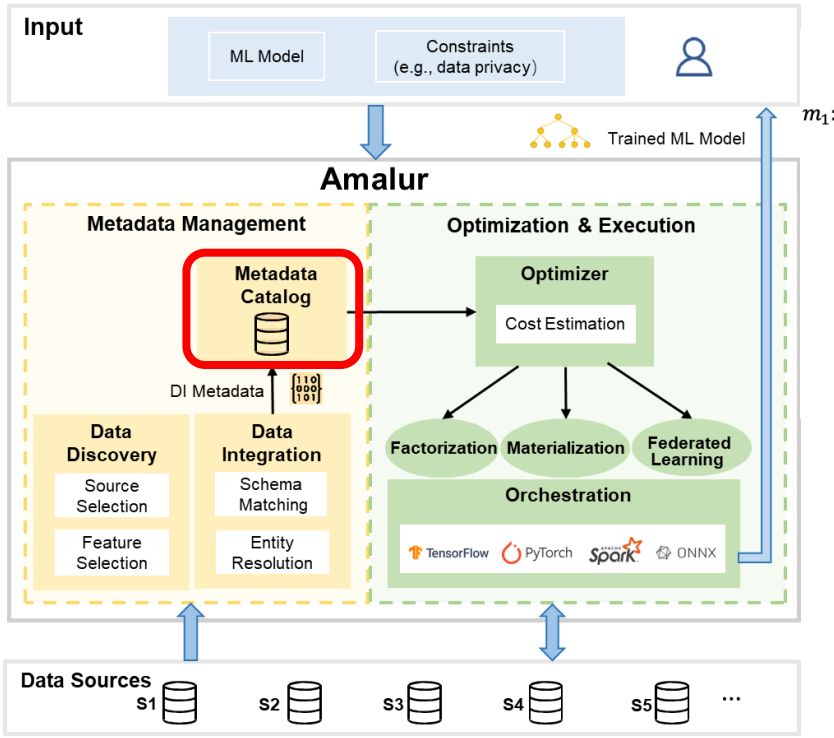
*Ziyu Li, Wenbo Sun, Danning Zhan, Yan Kang, Lydia Chen, Alessandro Bozzon, and Rihan Hai. Amalur: The convergence of data integration and machine learning. **TKDE**, pages 1–14, 2024.*

Amalur: Model Lake



*Rihan Hai, Christos Koutras, Andra Ionescu, Ziyu Li, Wenbo Sun, Jessie van Schijndel, Yan Kang, and Asterios Katsifodimos. Amalur: Data integration meets machine learning. In **ICDE**, pages 3729–3739, 2023.*

Amalur: Model Lake



Column matching (schema mapping)

$$m_1: \forall m, n, a, hr, o, dd\ (S_1(m, n, a, hr) \wedge S_2(m, n, a, o, dd) \longrightarrow T(m, a, hr, o))$$

$$M_1 \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \quad M_2 \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

(a) Mapping matrix

Row matching (entity resolution)

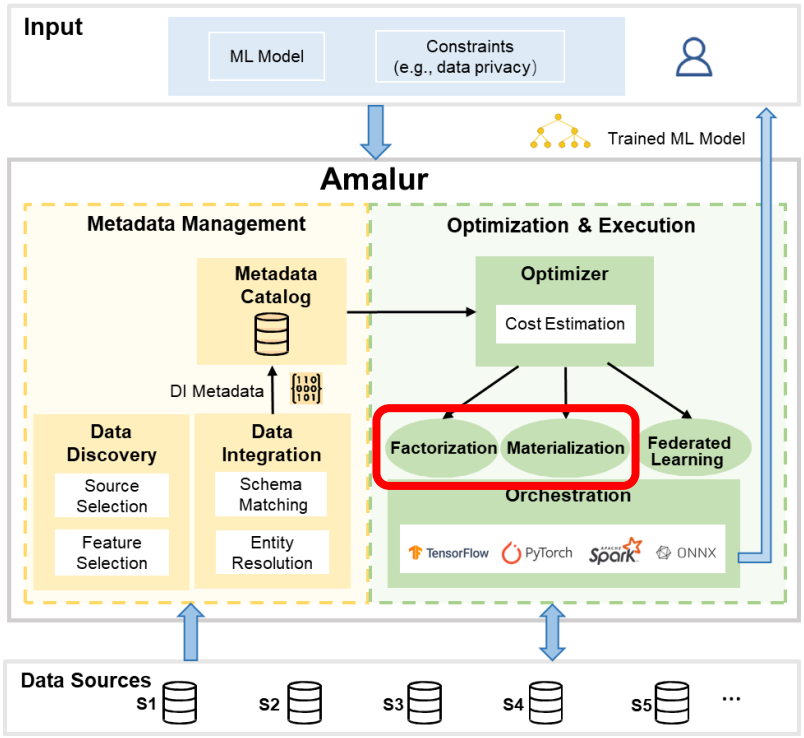
S_1	3	1	Jane	37	70	
S_2	2	1	Jane	37	92	11/5/21

$$I_1 \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad I_2 \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

(b) Indicator matrix

Representing DI metadata as tensors

Amalur: Model Lake

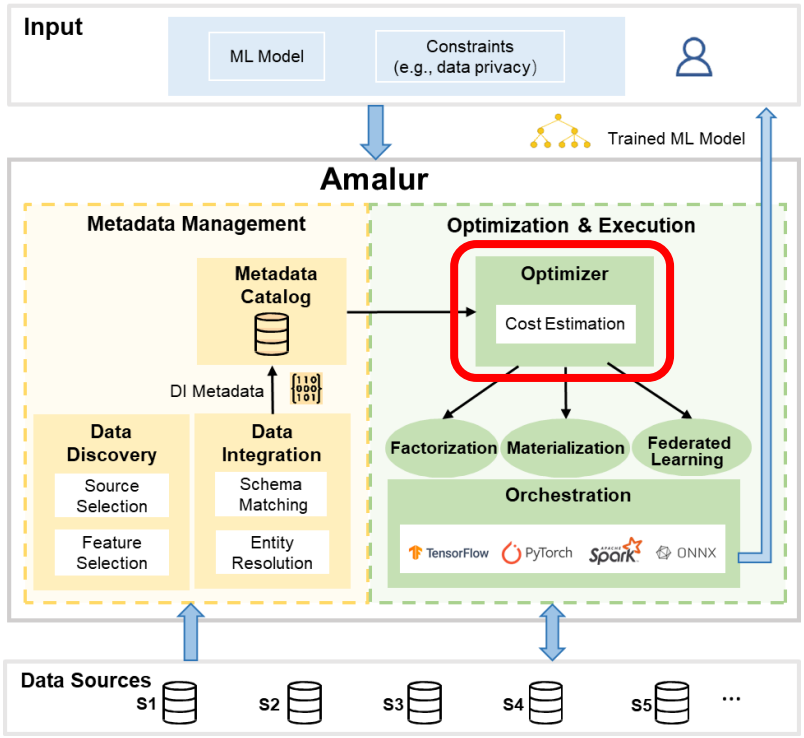


$$\text{Materialize: } T \rightarrow I_1 D_1 M_1^T + (I_2 D_2 M_2^T \circ R_2)$$

Factorize: $TX \rightarrow T_1 X + (T_2 \circ R_2) X$

Materialize: $T \rightarrow I_1 D_1 M_1^T + (I_2 D_2 M_2^T \circ R_2)$

Amalur: Model Lake



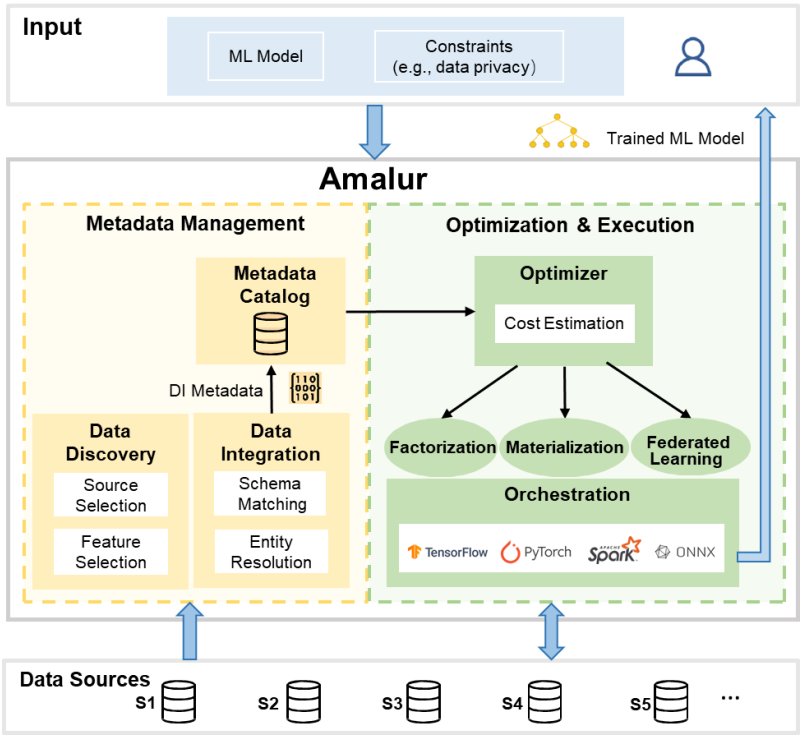
$$\text{Materialize: } T \rightarrow I_1 D_1 M_1^T + (I_2 D_2 M_2^T \circ R_2)$$

Factorize: $TX \rightarrow T_1 X + (T_2 \circ R_2) X$

Materialize: $T \rightarrow I_1 D_1 M_1^T + (I_2 D_2 M_2^T \circ R_2)$

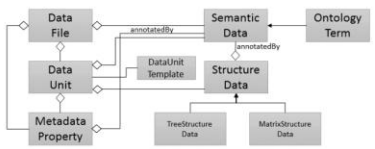
The whole picture

Model Lake Amalur

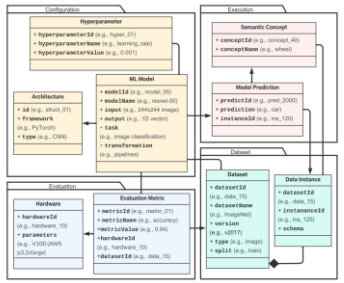


Metadata Representations

Data



ML model



Conceptual

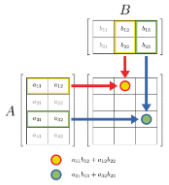
FO logic

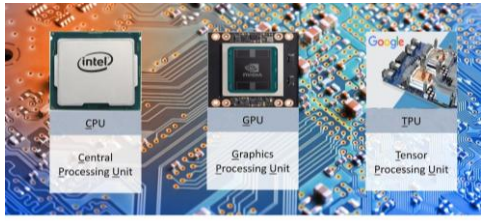
$$m_1 : \forall s, n, se, a, p, d \ (S_1(s, n, se, a) \wedge S_2(s, n, se, p, d) \rightarrow T(s, se, a, p))$$

Logical

Matrices

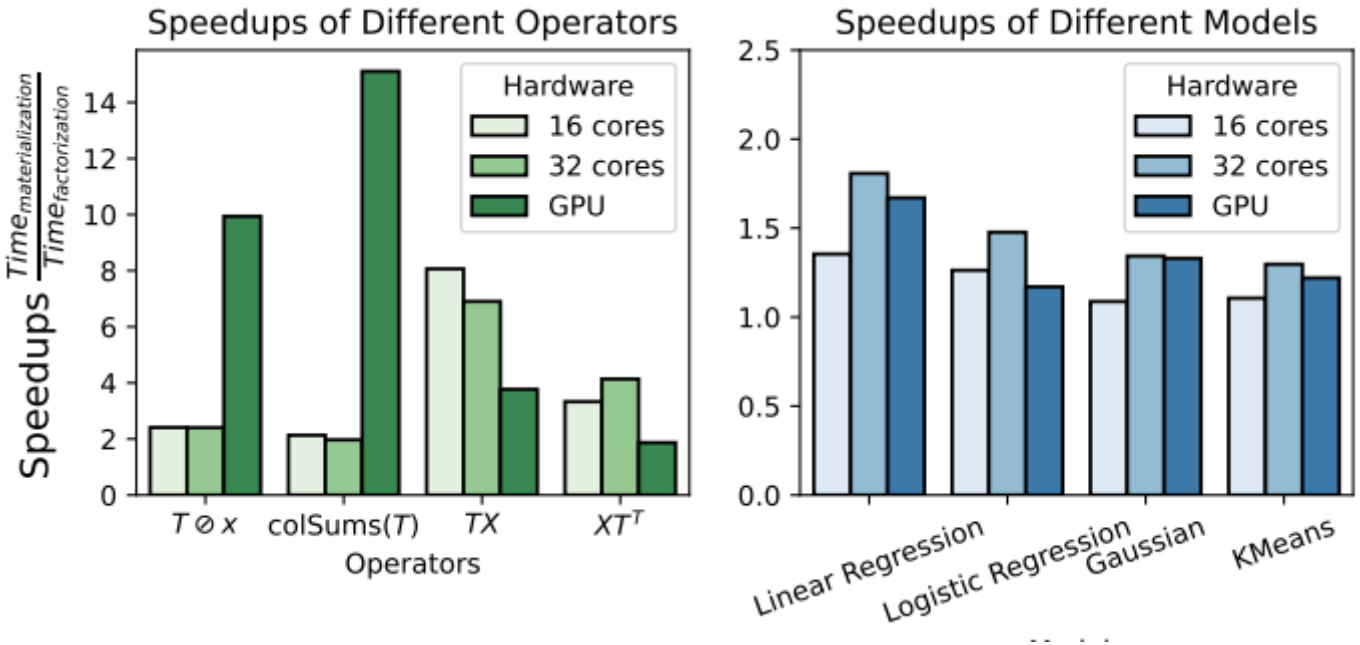
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$





Physical

Does it work?



Sun, Wenbo, Asterios Katsifodimos, and Rihan Hai. "Accelerating machine learning queries with linear algebra query processing." Proceedings of the 35th International Conference on Scientific and Statistical Database Management. 2023.

Does it work?

Associativity of matrix multiplication

$$\begin{aligned} \text{predictions} &= TL \\ &= (I_1 BM_1 + I_2 CM_2 + I_3 DM_3)L \\ &= I_1 \underline{BM_1 L} + I_2 \underline{CM_2 L} + I_3 \underline{DM_3 L} \end{aligned}$$

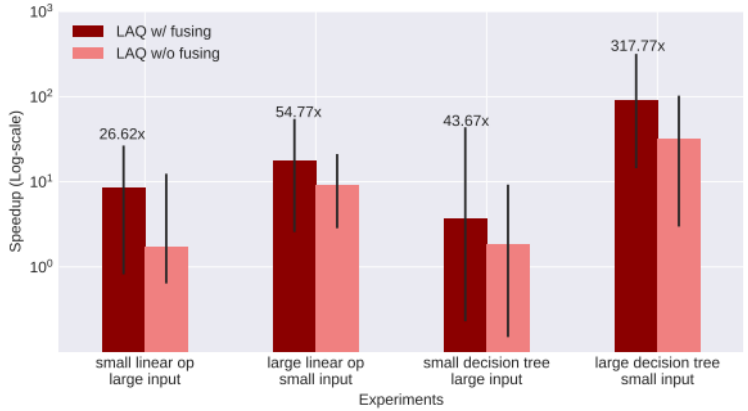


Figure 1: Speedups of our operator fusion method in four experimental predictive pipelines. The baseline is cuDF without operator fusion. The maximum attainable speedup is 317.77x.

Sun, Wenbo, Asterios Katsifodimos, and Rihan Hai. "Accelerating machine learning queries with linear algebra query processing." Proceedings of the 35th International Conference on Scientific and Statistical Database Management. 2023.

Research I

TranSQL+: LLM inference compiled into SQL

A template-based compiler that serves modern LLMs inside relational database engines on low-resource hardware.



Wenbo Sun

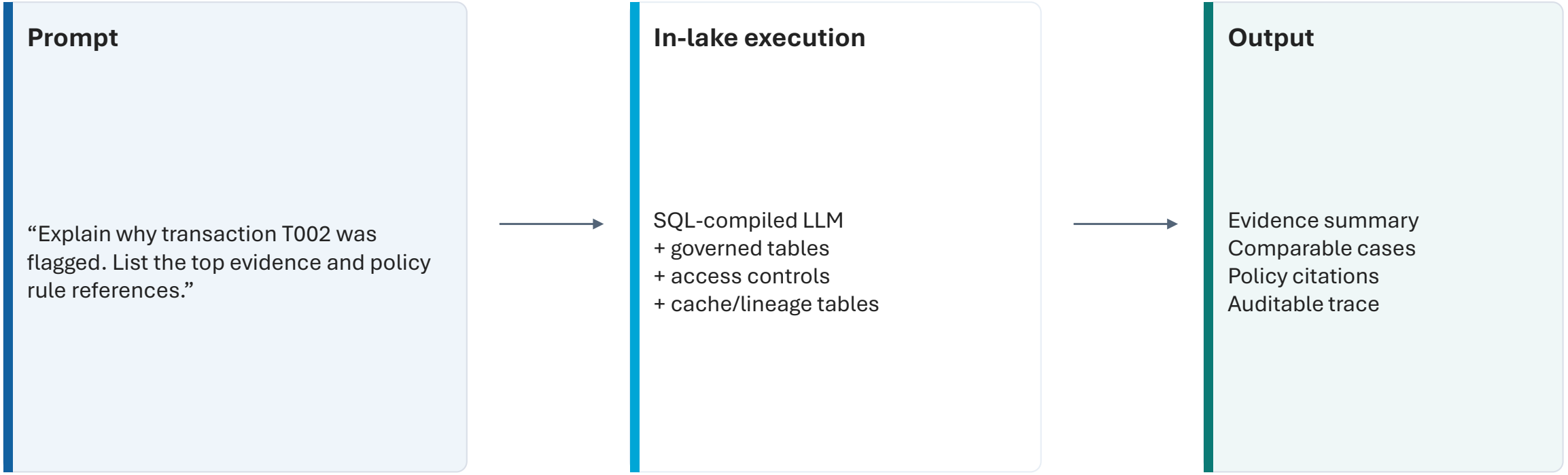
Research-35

Day: Thursday

Time: 11:00 AM - 12:30 PM

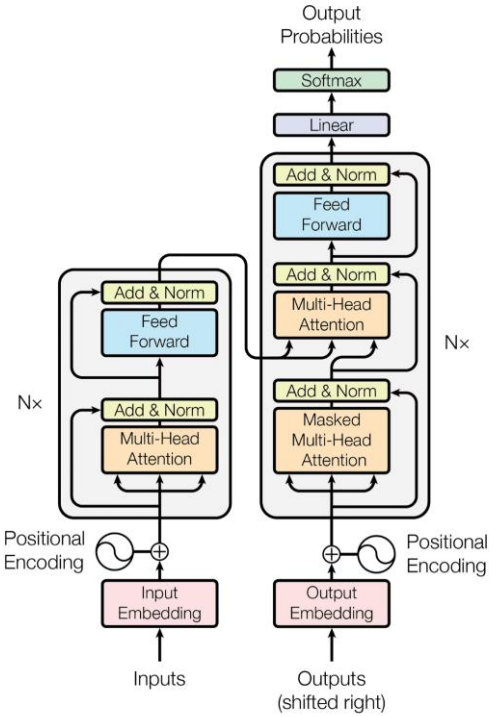
Finance example: local compliance copilot

An in-lake LLM can answer over sensitive data without exporting raw records.



The research opportunity is to combine model serving with database governance, provenance, and query optimization.

Transformer Architecture: Overview



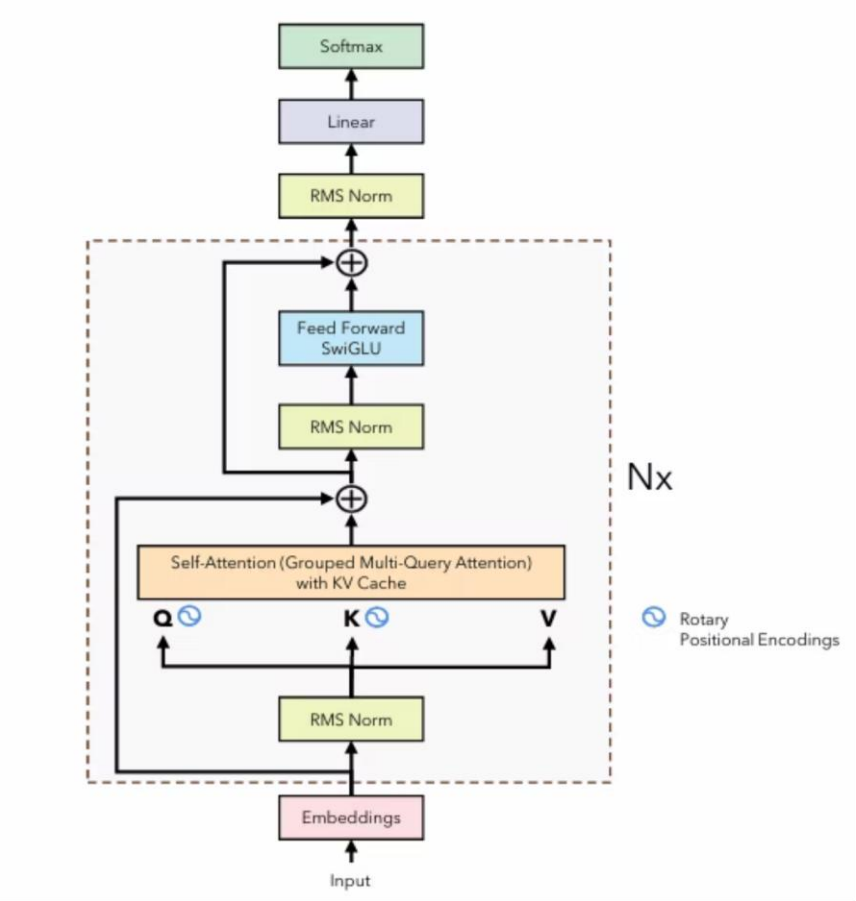
Encoder Stack:

- N identical layers (N=6 in original paper)
- Each layer has **two sub-layers**:
  - Multi-head **self-attention** mechanism
  - Position-wise **fully connected** feed-forward network
- **Residual connection** around each sub-layer
- **Layer normalization** after each residual connection

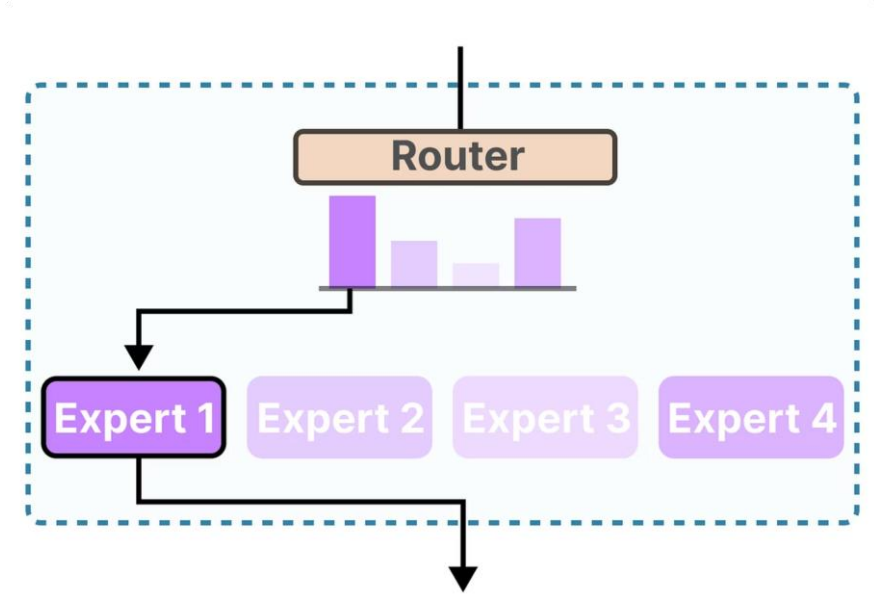
Decoder Stack:

- N identical layers (N=6 in original paper)
- Each layer has **three** sub-layers:
  - **Masked** multi-head self-attention
  - Multi-head attention over encoder output
  - Position-wise feed-forward network
- Residual connections and layer normalization

LLM Architecture



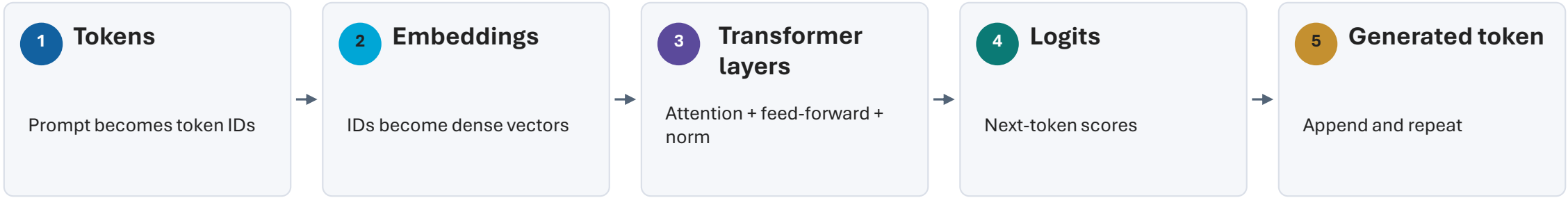
LLaMA



Mixture-of-Experts

Our intuition: LLM inference is a repeated dataflow program

A transformer layer is mostly linear algebra plus element-wise operations.



System view

The model is a graph of operators:
MatMul, Softmax, Norm, activation, reshape.

Database view

Those operators can be expressed using joins, aggregations, projections, and materialized state.

Why SQL can be an AI runtime

- The database does not “understand language”; it executes the numeric program produced by the model
- LLM operators have relational counterparts

Neural operator	Relational expression	Why it matters
Matrix multiplication	JOIN + SUM + GROUP BY	core cost in attention and FFN
Element-wise functions	projection expressions	activation, scaling, residuals
Softmax / normalization	group-wise aggregate + projection	attention weights and RMSNorm
KV cache/residuals	materialized tables	reuse during decoding
MoE expert selection	indexed lookup	read only activated experts

Why SQL can be an AI runtime

- The database does not “understand language”; it executes the numeric program produced by the model
- LLM operators have relational counterparts

Table 1. Templates for primary operator categories and examples.

Operator Category	Operator	Template	SQL Query
Matrix multiplication	AB	$R_3 \leftarrow \Pi_{\mathcal{G},\mathcal{S},\mathrm{DOT}(c_1,c_2)}\left(R_1 \bowtie_{\mathcal{S}} R_2\right)$ $\Pi_{\mathcal{G},\mathrm{SUM}(c_3)}\left(\gamma_{\mathcal{G}}(R_3)\right)$	SELECT A.m, B.q, SUM(DOT(A.chunk, B.chunk)) FROM A JOIN B ON A.n = B.p GROUP BY A.m, B.q
Element-wise function	Sigmoid(A)	$\Pi_{\mathcal{F},\, f(c_1)}\left(R_1\right)$	SELECT A.m, A.n, 1/(1+exp(-A.chunk)) FROM A
Element-wise arithmetic	$A + B$	$\Pi_{\mathcal{S},\, f(c_1,c_2)}\left(R_1 \bowtie_{\mathcal{S}} R_2\right)$	SELECT A.m, A.n, A.chunk + B.chunk FROM A JOIN B ON A.m = B.p AND A.n = B.q
Reshape	A.flatten()	$\Pi_{\mathcal{G},f(\mathcal{S}),\, c_1}\left(R_1\right)$	SELECT A.m*M+A.n, A.chunk FROM A
Normalization	Softmax(A)	$R2 \leftarrow \Pi_{\mathcal{F},\mathcal{G},\, \mathrm{agg}(f(c_1))}\left(R_1\right)$ $R3 \leftarrow \Pi_{\mathcal{G},\mathrm{agg}(c_2)}\gamma_{\mathcal{G}}\left(R_2\right)$ $\Pi_{\mathcal{F},\mathcal{G},g(c_2,c_3)}\left(R_2 \bowtie_{\mathcal{S}} R_3\right)$	WITH exp_sum AS (SELECT A.m, SUM(SUM(exp(A.chunk))) AS summation FROM A GROUP BY A.m) SELECT A.m, A.n, exp(A.chunk)/summation FROM A JOIN exp_sum ON A.m = exp_sum.m

Template-based SQL Code Generation

A neural operator acting on p operands is represented as:

$$\mathrm{OP}_{\mathrm{attr}}\Big(\{R_1, \ldots, R_p\}, \mathcal{F}, \mathcal{S}, \mathcal{G}\Big),$$

Where:

- $\mathcal{F}$ denotes the set of free dimensions retained in the output (grouping or projection columns)
- $\mathcal{S}$ encodes the mappings of shared dimensions between operands (translated to SQL join keys)
- $\mathcal{G}$ represents the dimensions used in GROUP BY clauses
- attr indicates operator-specific attributes that guide the projection logic

Toy example: matrix multiplication as SQL




```
SELECT
  A.row_id, B.col_id,
  SUM(A.value * B.value) AS out_value
FROM A
JOIN B ON A.k = B.k
GROUP BY A.row_id, B.col_id;
```

Why this matters

Once linear algebra is expressed as relational queries, decades of database optimization become available: joins, aggregates, vectorized execution, caching, and out-of-core processing.

Why use templates?

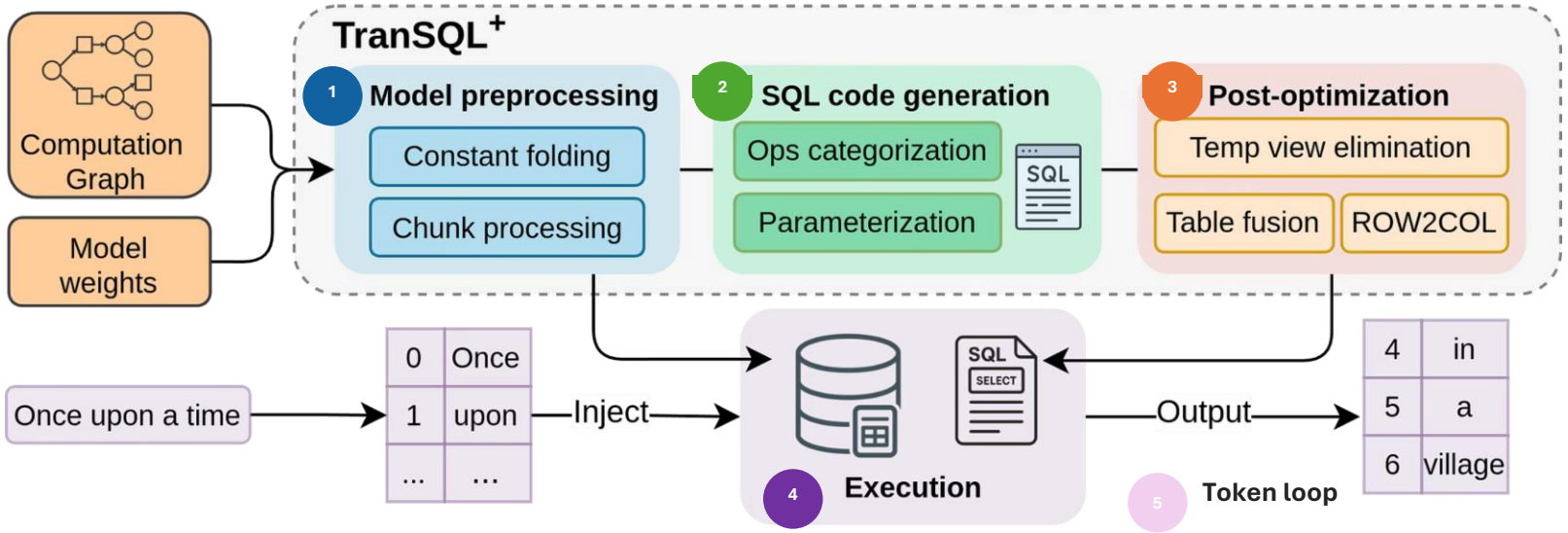
Operator Category	Operator	Template	SQL Query
Matrix multiplication	AB	$R_3 \leftarrow \Pi_{\mathcal{G},\mathcal{S},\mathrm{DOT}(\mathbf{c}_1,\mathbf{c}_2)}(R_1 \bowtie_{\mathcal{S}} R_2)$ $\Pi_{\mathcal{G},\mathrm{SUM}(\mathbf{c}_3)}(\gamma_{\mathcal{G}}(R_3))$	SELECT A.m, B.q, SUM(DOT(A.chunk, B.chunk)) FROM A JOIN B ON A.n = B.p GROUP BY A.m, B.q
Element-wise function	Sigmoid(A)	$\Pi_{\mathcal{F},\, f(\mathbf{c}_1)}(R_1)$	SELECT A.m, A.n, 1/(1+exp(-A.chunk)) FROM A
Element-wise arithmetic	$A + B$	$\Pi_{\mathcal{S},\, f(\mathbf{c}_1,\mathbf{c}_2)}(R_1 \bowtie_{\mathcal{S}} R_2)$	SELECT A.m, A.n, A.chunk + B.chunk FROM A JOIN B ON A.m = B.p AND A.n = B.q
Reshape	A.flatten()	$\Pi_{\mathcal{G},f(\mathcal{S}),\, \mathbf{c}_1}(R_1)$	SELECT A.m*M+A.n, A.chunk FROM A
Normalization	Softmax(A)	$R2 \leftarrow \Pi_{\mathcal{F},\mathcal{G},\, \mathrm{agg}(f(\mathbf{c}_1))}(R_1)$ $R3 \leftarrow \Pi_{\mathcal{G},\mathrm{agg}(\mathbf{c}_2)}\gamma_{\mathcal{G}}(R_2)$ $\Pi_{\mathcal{F},\mathcal{G},g(\mathbf{c}_2,\mathbf{c}_3)}(R_2 \bowtie_{\mathcal{S}} R_3)$	WITH exp_sum AS (SELECT A.m, SUM(SUM(exp(A.chunk))) AS summation FROM A GROUP BY A.m) SELECT A.m, A.n, exp(A.chunk)/summation FROM A JOIN exp_sum ON A.m = exp_sum.m

These templates cover the core neural operators used in transformer-based LLMs like Llama3 and DeepSeek MoE.

Why templates?

- **Extensibility**: easy to add new operators, simply add templates
- Examples: transformer components, convolution kernels

An overview of the workflow of TranSQL+



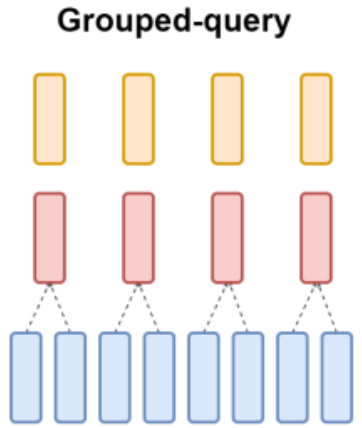
What is compiled?

The forward-pass computation graph: MatMul, Softmax, RMSNorm, activation, reshape, cache updates.

What is not required?

No custom inference engine inside the DB. The generated program is SQL.

LLaMA-3 8B



Model dimension d_k= 4096
Number of attention heads = **32**
So, per head dimension = **4096 ÷ 32 = 128**

For **keys (K)**:

$$W_K = [W_{K,1} W_{K,2} \ \ldots W_{K,32}]$$

Where each $W_{K,h} \in \mathcal{R}^{4096x128}$

For **values (V)**:

$$W_V = [W_{V,1} W_{V,2} \ \ldots W_{V,32}]$$

Where each $W_{V,h} \in \mathcal{R}^{4096x128}$

Model Preprocessing

1. Chunk-Based Representation

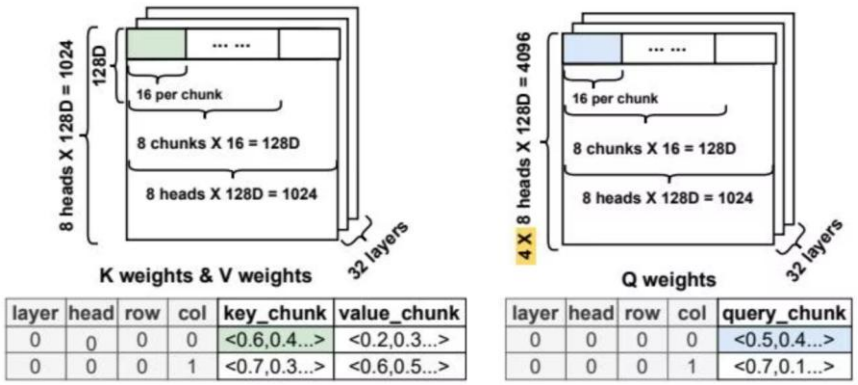
To execute neural operators via SQL, model weight matrices are converted into relational tables using a chunked format:

- Matrices are split into smaller blocks, indexed by row and column column
- Each row is split into chunks of fixed size
- For a matrix $W \in \mathbb{R}^{m \times n}$, each row is represented as:

$$\mathbf{w}_i = \left[\mathbf{w}_i^{(0)}, \mathbf{w}_i^{(1)}, \ldots, \mathbf{w}_i^{(C)} \right]$$

In the database table, each row corresponds to:

$$(i, c, \mathbf{w}_i^{(c)})$$



layer	head	row	col	key_chunk	value_chunk
0	0	0	0	<0.6,0.4...>	<0.2,0.3...>
0	0	0	1	<0.7,0.3...>	<0.6,0.5...>

layer	head	row	col	query_chunk
0	0	0	0	<0.5,0.4...>
0	0	0	1	<0.7,0.1...>

Figure 4: Illustration of slicing K, V, Q weights into chunks.

This chunking strategy provides flexibility by allowing trade-offs between the between the number of rows and the workload per thread when performing performing vector operations.

Model Preprocessing

2 Constant Folding

Precompute constant tensors and any downstream linear or scalar operations

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Benefit: avoid redundant computation at inference time

Example: attention scaling factor $\frac{1}{\sqrt{d_k}}$

Optimization 1: materialize only what must be reused

Problem: Naïvely translating each operator creates too many temporary views.

Before

Every operator becomes a temporary view.
Heavy I/O and repeated scans.

After

Merge non-critical operators into CTE blocks.
Materialize only reusable state.

38 → 7

materialized intermediates per Llama3-8B layer

Critical nodes

KV vectors for decoding and embeddings for residual connections are reused, so they remain materialized.

Optimization 2: table fusion for Q/K/V

Problem: Some projections share the same input; computing them separately wastes scans



Database interpretation

Fuse weight tables with a projection-type flag. One scan computes Q, K, and V in a single SQL query, reducing I/O and exposing better parallelism.

Optimization 3: ROW2COL pivoting

Change the physical representation to fit vectorized database execution

Row chunks

row	chunk	vector
0	0	[...]
0	1	[...]
1	0	[...]
1	1	[...]

Column slices

row	chunk0	chunk1
0	[...]	[...]
1	[...]	[...]
2	[...]	[...]

Effect

Lower join cardinality
Better SIMD locality
More parallel slices

Key idea

A database runtime is sensitive not only to the formula, but also to the physical layout of intermediate tensors.

Prefill and decoding latency

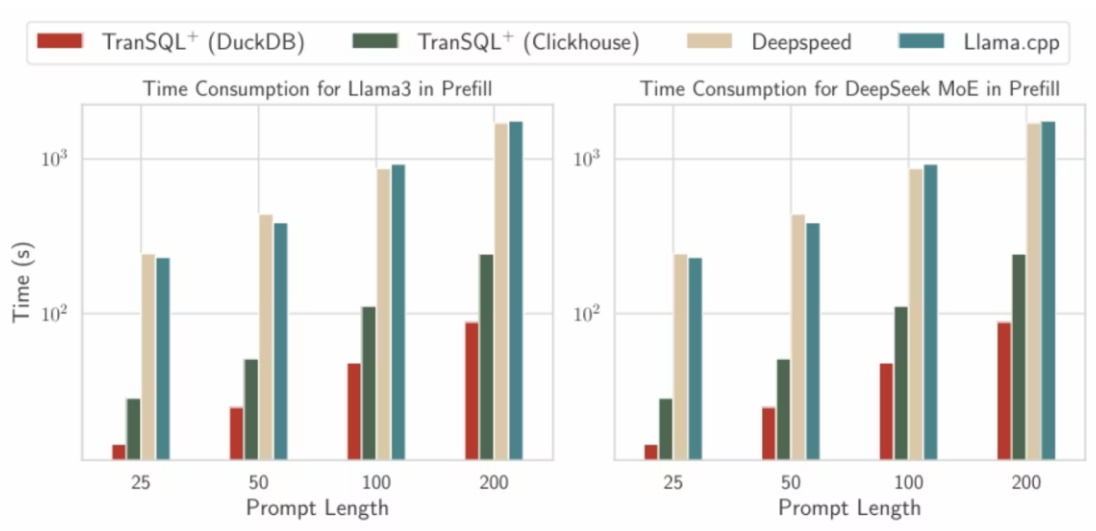


Figure 6: Prefilllatency with varying prompt length. TranSQL+ gains up to **20x speedups** compared to Deep speed and Llama.cpp.

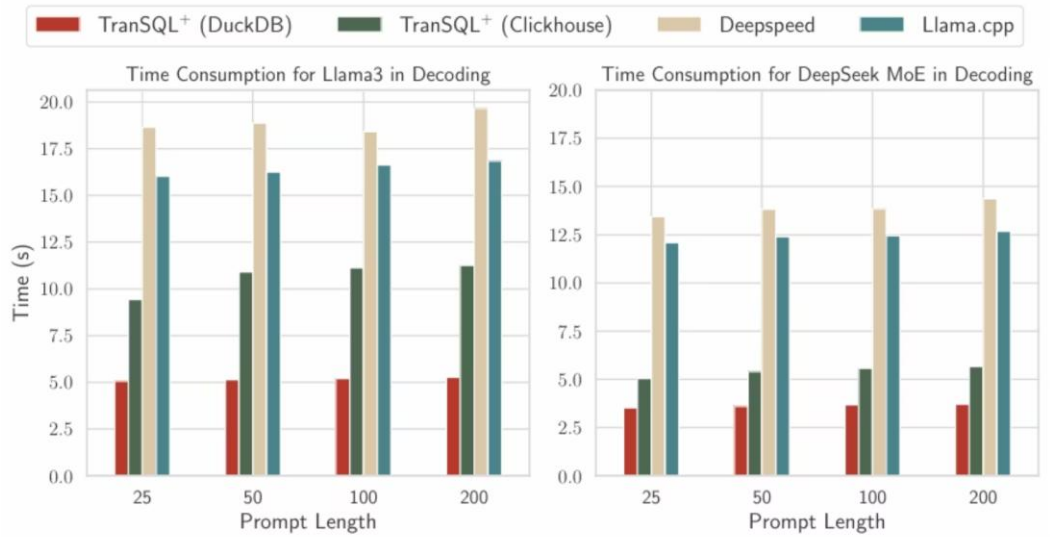


Figure7: Decoding latency with varying prompt length.TranSQL+gains up to length.TranSQL+gains up to **4x speedups** compared to Deepspeed and and Llama.cpp.

TranSQL+ outperforms existing inference frameworks on resource-constrained hardware.

Impact of Post-optimizations

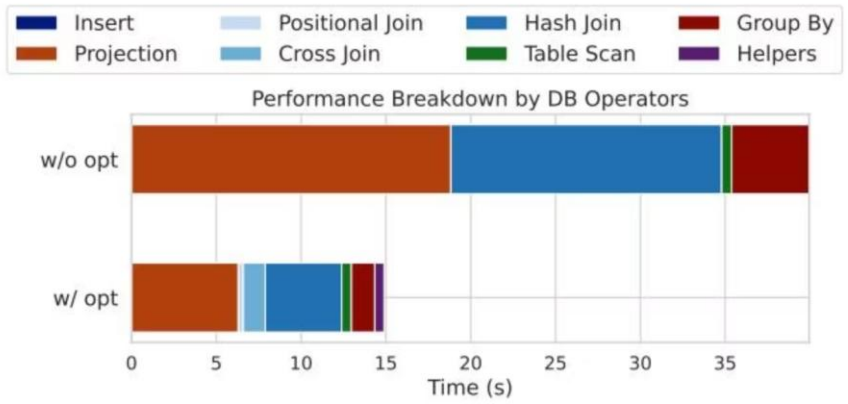


Figure 9: Join time drops by over 60% because the optimizations reduce reduce intermediate cardinality.

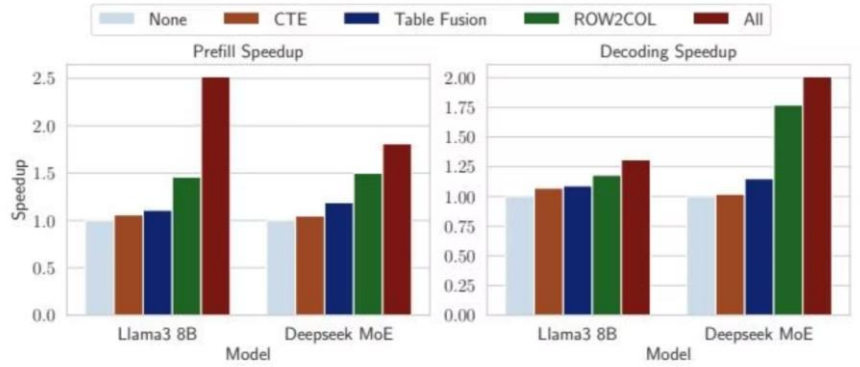


Figure 10: Ablation study for three post optimizations.

- The post-optimization strategy reduces total execution time by **64.8%** compared to unoptimized SQL queries. queries.
- Among the three optimizations, **ROW2COL** has the most dramatic effect by exploiting columnar storage and SIMD-parallel and SIMD-parallel operations.

Influence of Memory Capacity

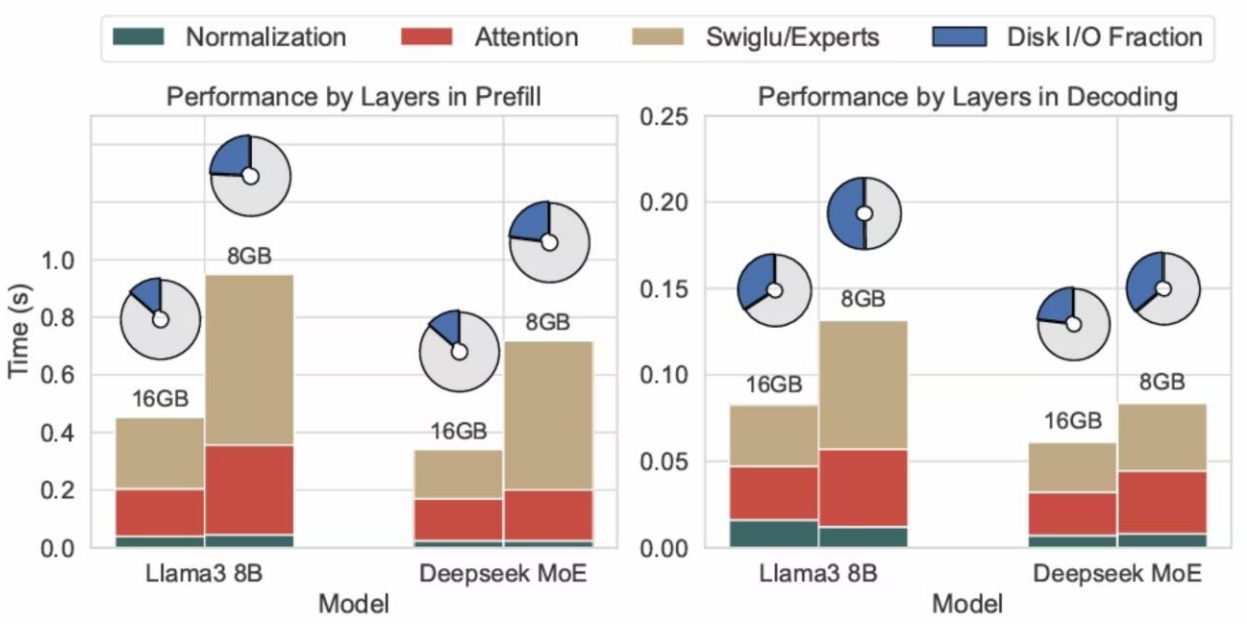


Figure 12: Layer-wise runtime and disk-I/O fraction for 25-token prompts under varying memory. Prefill performance degrades with less memory (nearly 2×disk I/O). MoE decoding scales better since only a subset of experts is active, reducing I/O.

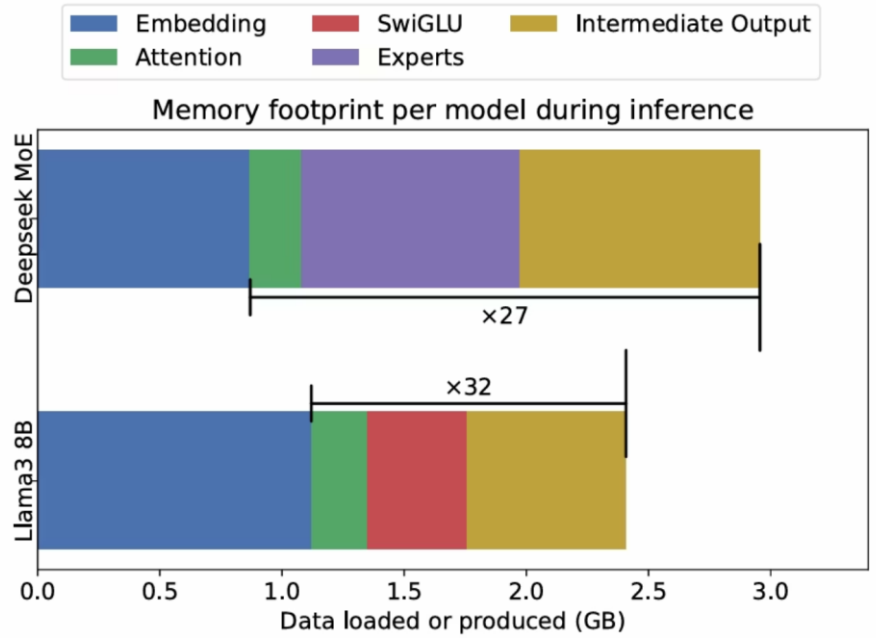


Figure 13: Prefill memory usage with 50-token input.

- Memory capacity is critical for efficient LLM inference on low-resource hardware.
- Adequate memory reduces latency in both prefill and decoding by minimizing data movement.

What the experiments show

20×

lower prefill latency vs CPU baselines

4×

faster decoding in low-memory settings

64.8%

less total execution time after optimizations

MoE

benefits from indexed expert access

Scope

Best fit: CPU-only, limited memory, privacy-sensitive, local deployment.

Limitation

A high-end GPU remains faster. The point is portability and controlled execution.

TranSQL+ is not a GPU replacement; it targets CPU-only, memory-constrained settings.

Summary: Key Idea & Contributions

Core Idea: Translate the LLM computation graph into pure SQL, execute in a relational DBd

C1

Template-based SQL Generator

First template-based SQL code generator to serve serve modern LLMs inside DB engines.

C2

Optimization Techniques

ROW2COL pivot + table fusion reduce intermediate intermediate cardinality and boost vectorized execution.

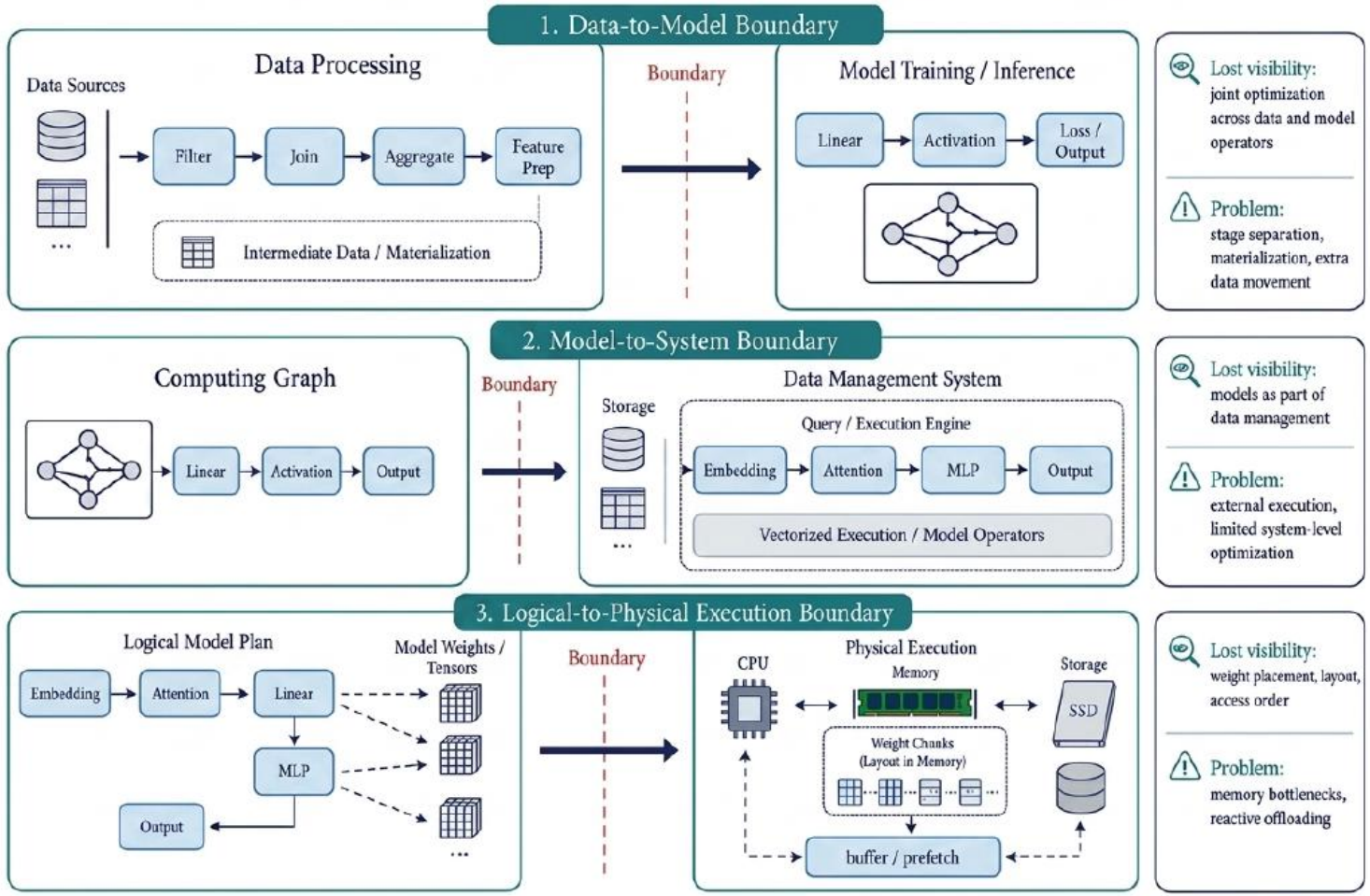
C3

Performance Gains

On CPU-only, low-memory hardware: **up to 20× 20× lower prefill latency**, **up to 4× faster decoding decoding** vs DeepSpeed & Llama.cpp.

Fragmentation in Modern ML Execution

Where execution–relevant information becomes hidden across boundaries



Part III

Diffusion models as data operators

The second direction: use generative models to repair, forecast, synthesize, and stress-test lake data under conditions.

Running example: fraud and risk analytics

tx_id	amount	merchant	region	device_risk	label
T001	€42.10	grocery	NL	0.12	normal
T002	€4,870	electronics	DE	?	review
T003	€213.40	travel	FR	0.71	fraud?

Analyst question

“Why was T002 flagged, and what similar cases should we inspect?”

Data-quality task

Device risk is missing; impute it using context and observations.

Scenario task

Generate plausible stress cases without exposing real customer data.

Finance example: policy-valid synthetic records

Synthetic data is only useful if it is both realistic and condition-compliant.

Inference-time constraints

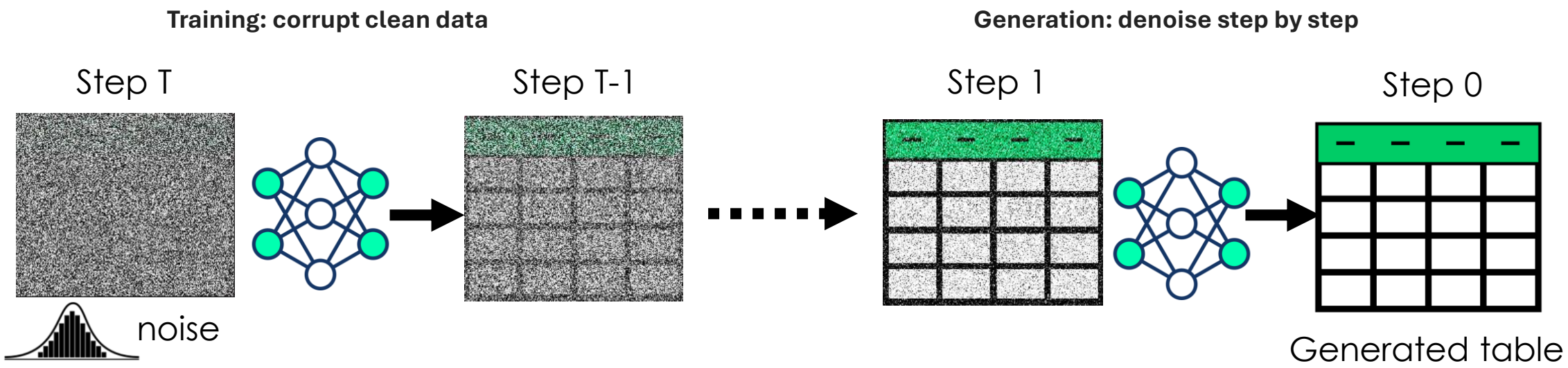
age ≥ 18
balance ≥ 0
region = EU
credit_limit ≤ policy threshold
merchant_category ∈ allowed set
missing KYC field must be imputed

age	balance	region	risk	credit_limit
37	€1,240	EU	0.18	€3,000
23	€420	EU	0.36	€1,200
61	€9,880	EU	0.07	€4,000

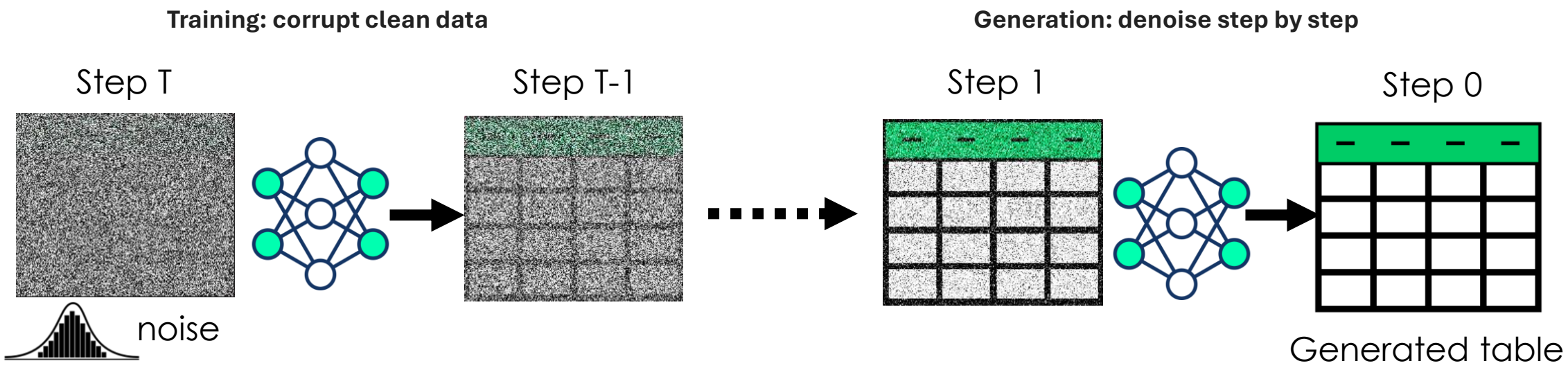
Use cases

Testing fraud pipelines, enriching rare-but-important cases, privacy-preserving analytics, and fairness checks across underrepresented segments.

Tabular Diffusion in 90 seconds



Tabular Diffusion in 90 seconds



Data-lake interpretation

For structured data, “denoising” can mean filling missing values, generating plausible future time windows, or synthesizing realistic rows under constraints.

Conditional diffusion: generate with steering

The practical value comes from controlling the generated data.

Metadata condition

customer segment = SME
region = EU
merchant category = travel
month = December

Observation condition

known values at some timestamps
partial transaction history
missing values to impute

Constraint condition

age ≥ 18
balance ≥ 0
risk score below policy threshold
region = allowed jurisdiction

The central challenge: conditions are often decided at inference time, not fixed during training.

Why financial time series are hard

Financial sequences are contextual, incomplete, multi-granular, and often urgent.



Context matters

Seasonality, holidays, product type, region, customer segment, merchant category.

Observations are partial

Missing market ticks, sparse fraud signals, delayed labels, sensor or logging gaps.

Generation must be fast

Scenario generation and imputation often sit inside interactive analytics loops.

This motivates WaveStitch: conditional generation that is flexible and parallelizable.

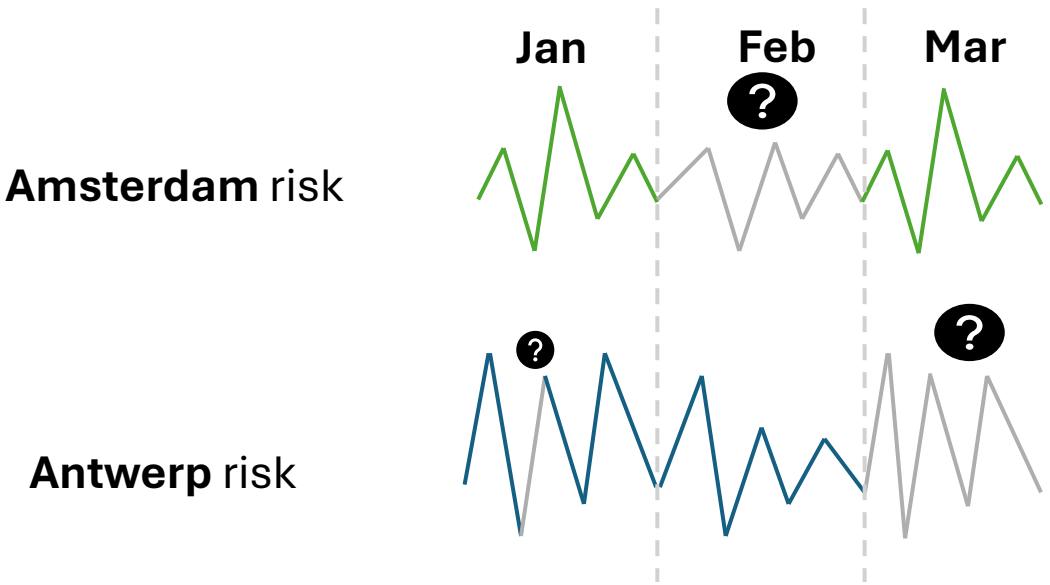
Research II

WaveStitch: fast conditional time-series generation

Generate, impute, and forecast time series using metadata and arbitrary partial observations.

Setting

- Time series comes with context
  - e.g. month, city, year, days..
- There may be gaps

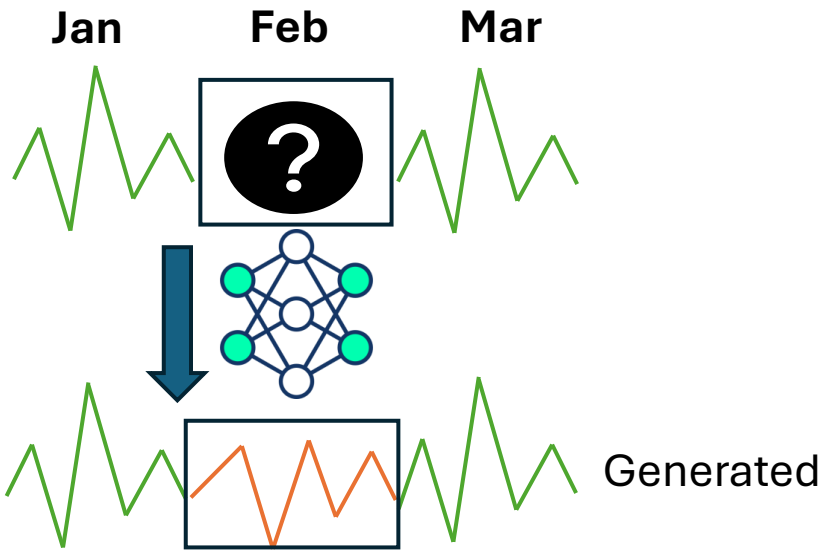


City	Month	Day	Risk
Ams.	Jan	1 2 .. 31	0.5 0.2 .. ..
Ams.	Feb	1 2 .. 28	?
Ams.	Mar	1 2 .. 31	..
Ant.	Jan	1 2 .. 31	..
Ant.	Feb	1 2 .. 28	..
Ant.	Mar	1 2 .. 31	?

Objective

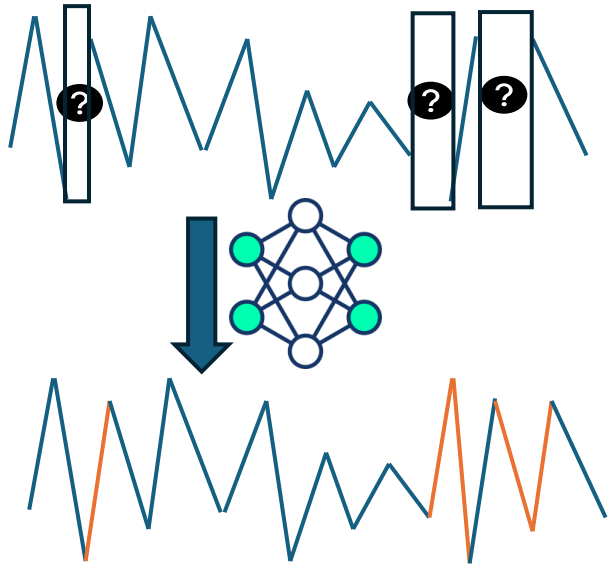
- **Simulate, forecast, or impute** at different granularities:

1 Coarse-grained patterns



e.g. Amsterdam, February

2 Fine-grained patterns



e.g. Antwerp, Jan., 18th, 21st, 5th etc.

Challenge 1: Dual-sourced Conditions

- Generated data is influeced by two sources:

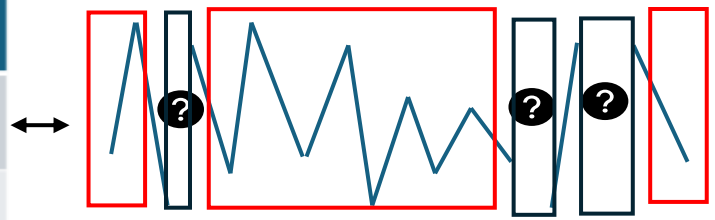
1 Context (metadata)

City	Month	Day	Temp.
Ams.	Jan	1 2 .. 31	5.5 6.2 .. ..
Ams.	Feb	1 2 .. 28	?
Ams.	Mar	1 2 .. 31	..
Ant.	Jan	1 2 31	..
Ant.	Feb	1 2 .. 28	..
Ant.	Mar	1 2 .. 31	?

e.g.
City=Antwerp
Month=Jan,
Day=2

2 Existing time series observations

City	Month	Day	Temp.
Ams.	Jan	1 2 .. 31	5.5 6.2 ?
Ams.	Feb	1 2 .. 28	8.1 6.2 .. ..



e.g. Neighbouring temperature values

- *Both can help, so how to utilise both soures?*

Challenge 2: Inference-time flexibility

- Observations can be at arbitrary positions
- Training-time methods generate fixed patterns/ratios



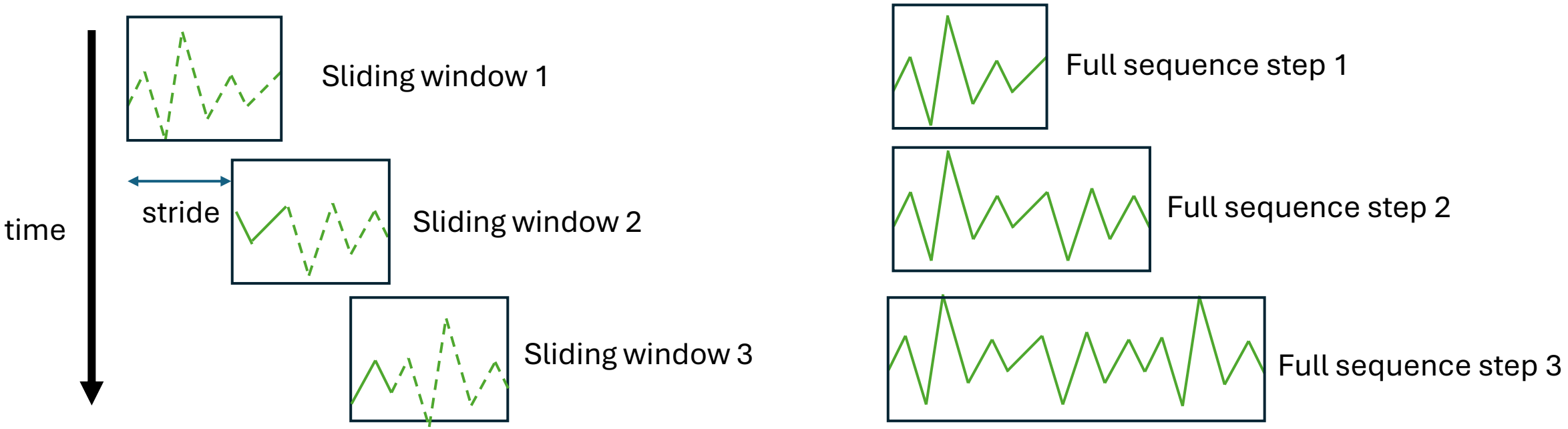
- Inference can have different patterns



- *How to generalise at inference-time?*

Challenge 3: Generation Speed

- Time series generation is naturally autoregressive

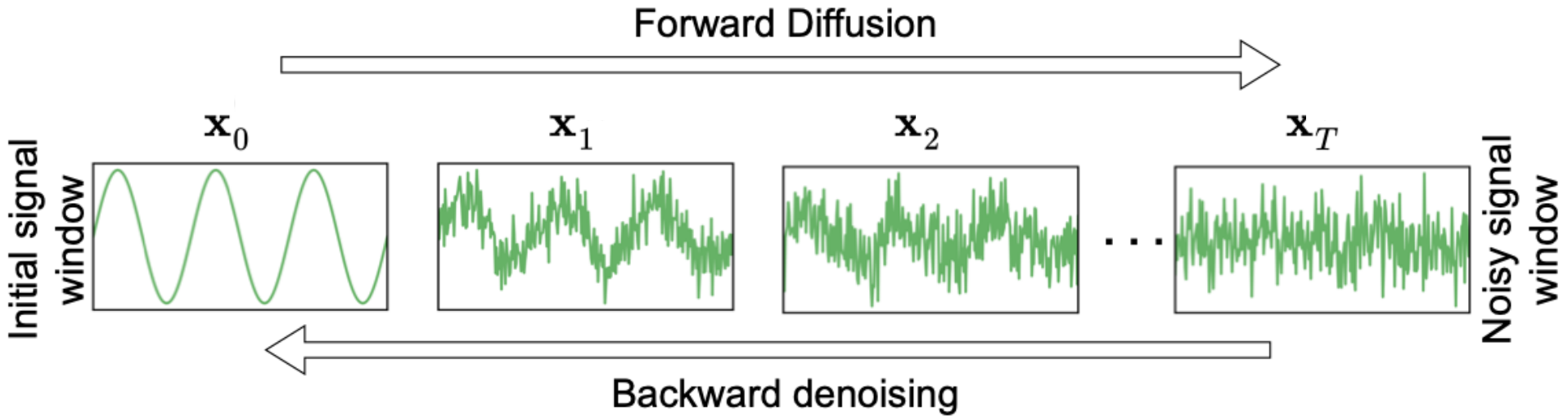


- *Autoregression is slow, can we parallelise?*

WaveStitch: Model choice

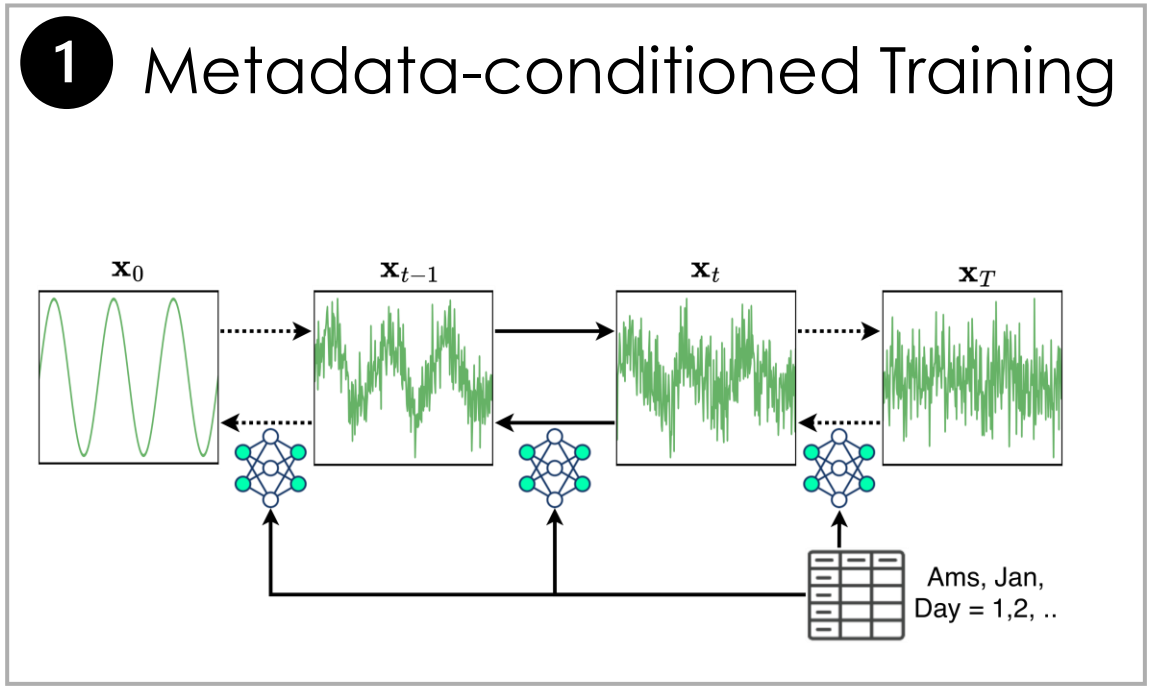
- *Diffusion* model for fast time series generation

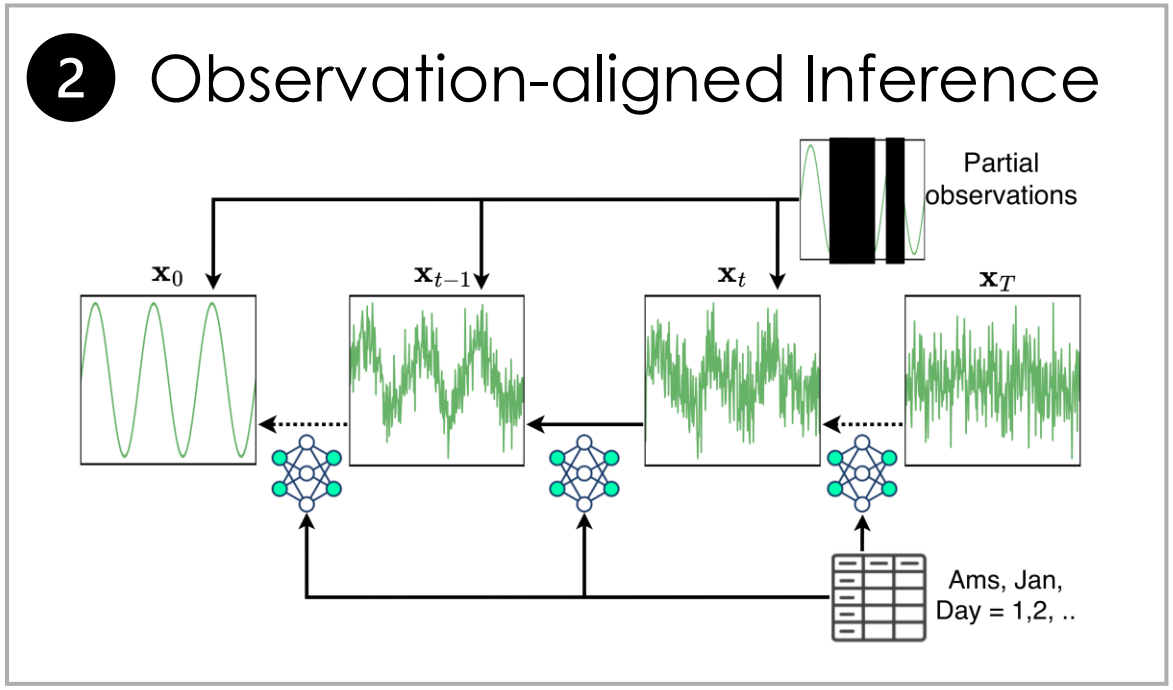
- Markovian noising-denoising process



Dual-conditioning design

- Metadata handled at **training**
- Alignment with observed values at **inference**

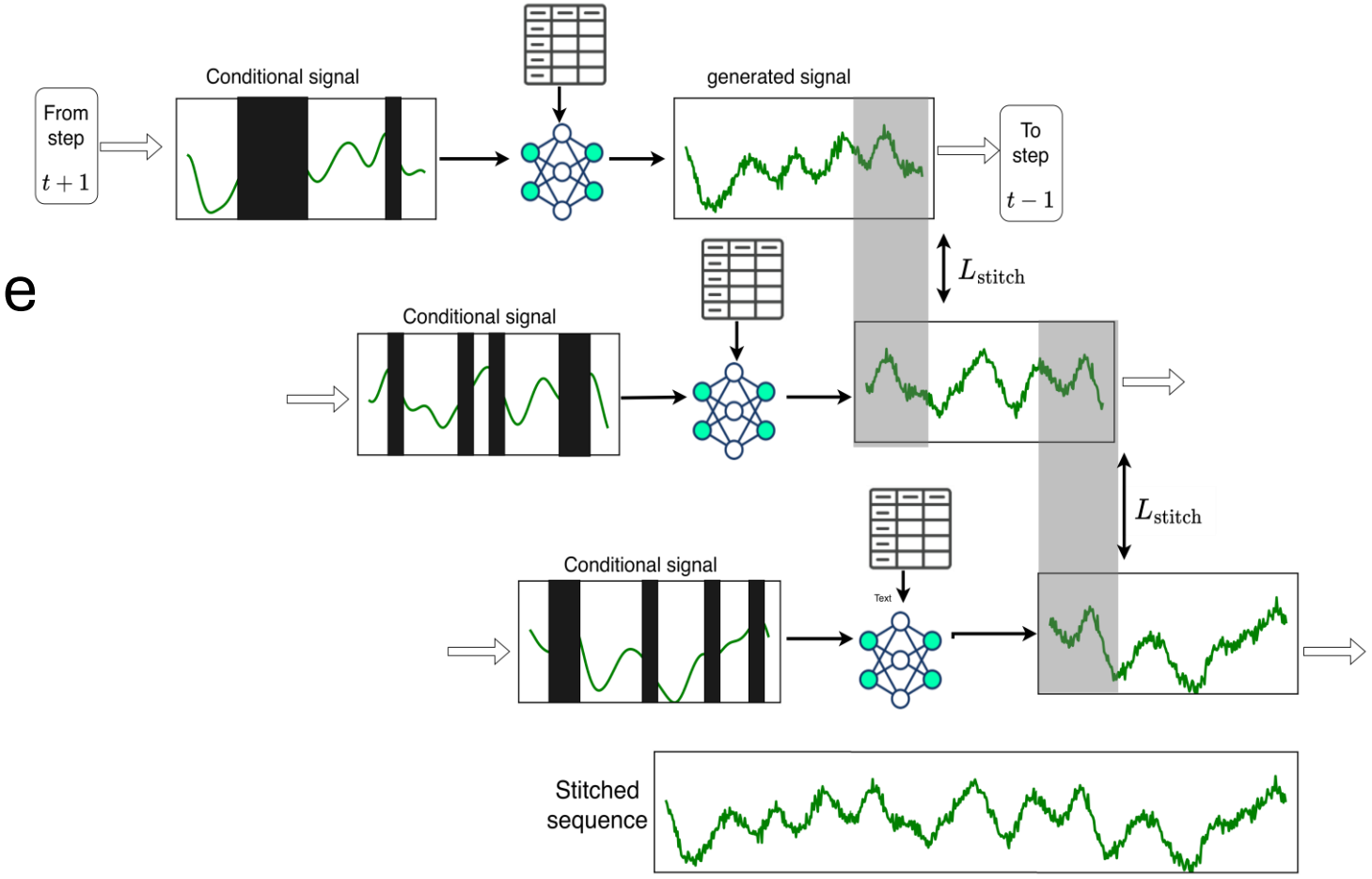




Pipelined Sequence Generation

- Generate overlapping windows in parallel
- Enforce overlap coherence with **stitch loss**

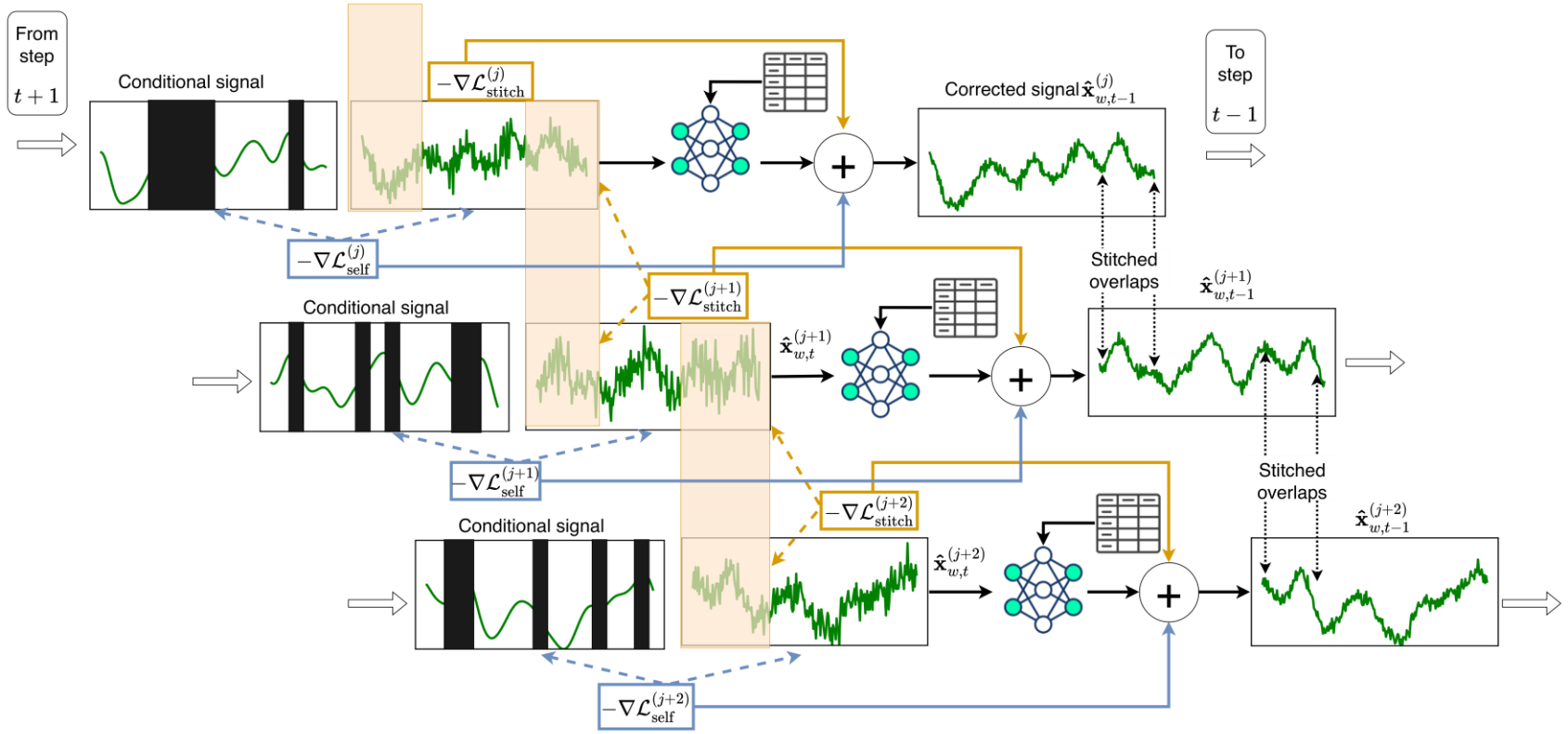
$$\nabla_x L_{\text{stitch}}$$



WaveStitch: Final design

- Combine **self-guidance** and **stitch** corrections

$$x := x - (\nabla_x L_{\text{stitch}} + \nabla_x L_{\text{self}})$$



WaveStitch takeaway

A sequence need not be generated sequentially.



10×

lower MSE compared with state-of-the-art methods in reported evaluations

460×

faster generation than autoregressive methods in reported evaluations

parallel

windows are generated concurrently and stitched coherently

For data lakes

A generative operator can impute, synthesize, and forecast time-series data while respecting available context.

For finance

Useful for missing market ticks, payment-volume scenario generation, liquidity stress paths, and privacy-preserving synthetic signals.

Research III

Harpoon: controllable tabular diffusion

The tabular counterpart: one diffusion model, many inference-time conditional-generation tasks.



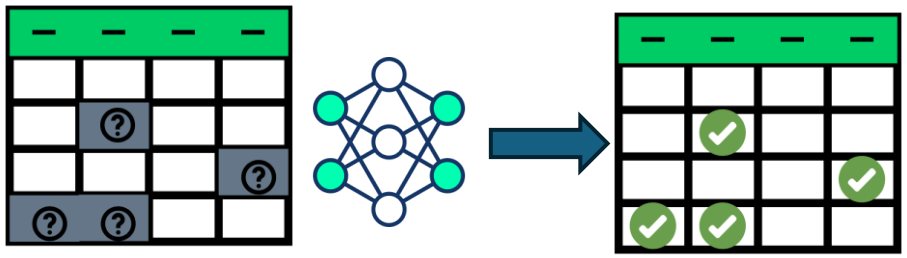
ICLR
International Conference On Learning Representations

2026

Conditional Tabular Diffusion

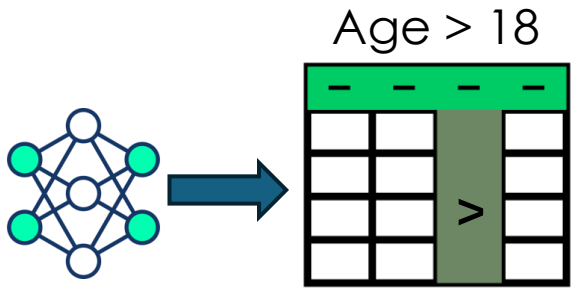
- Control the generative process

1 Imputation



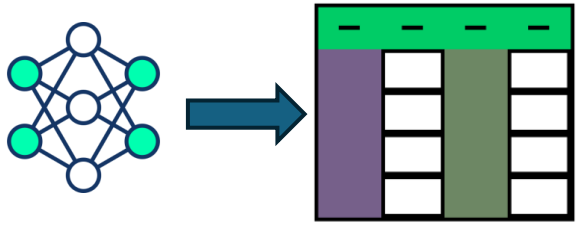
Imputation given partial observations

2 Inequalities



Inequality constraints: >, <, >=, <=

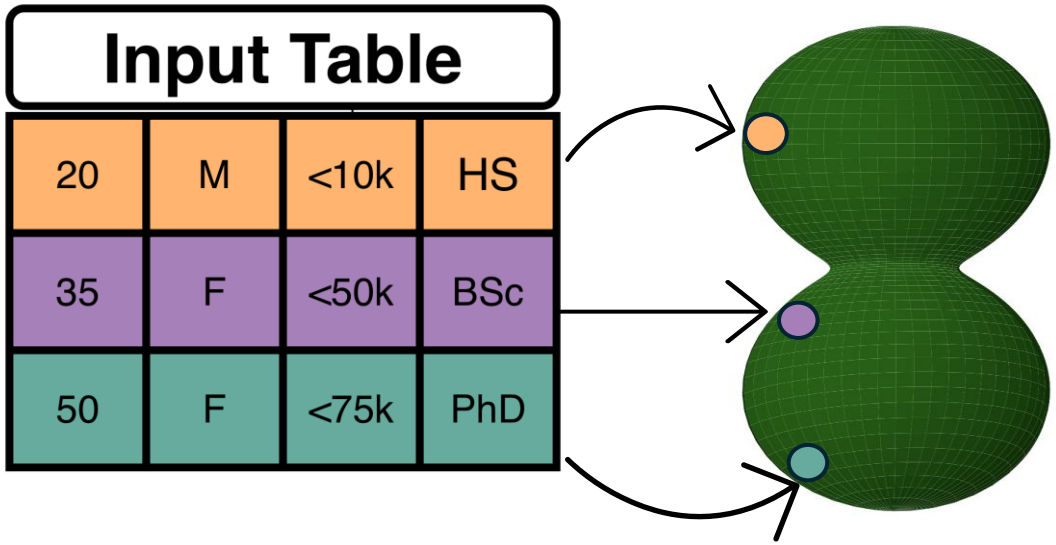
3 Combinations



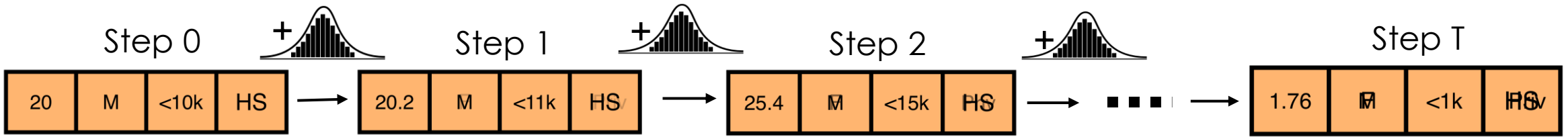
e.g. Color='Blue' **AND** Age > 18

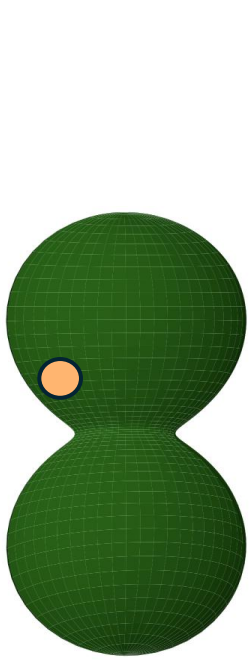
The Manifold Hypothesis

- Tabular data points assumed to lie on low-dim **manifold**

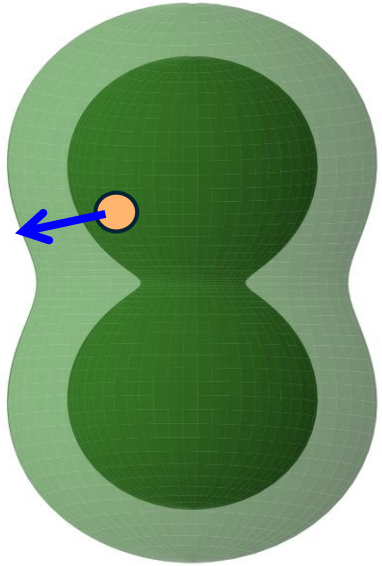


Geometric view of Tabular Diffusion

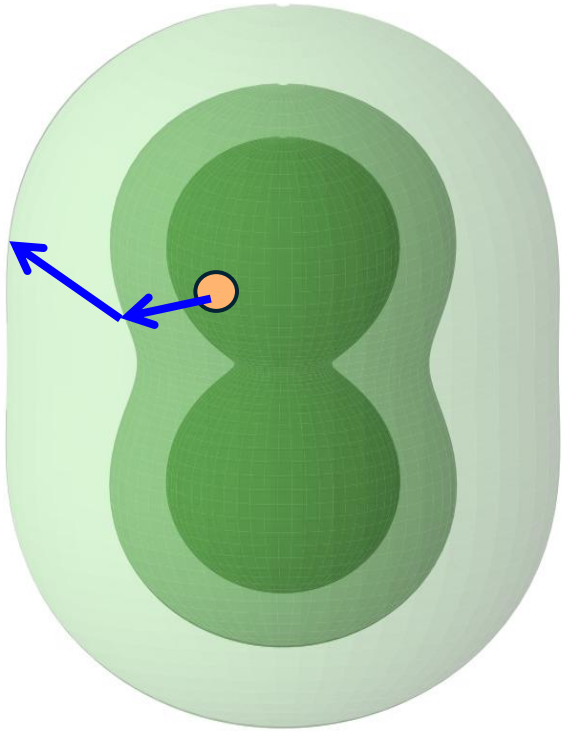




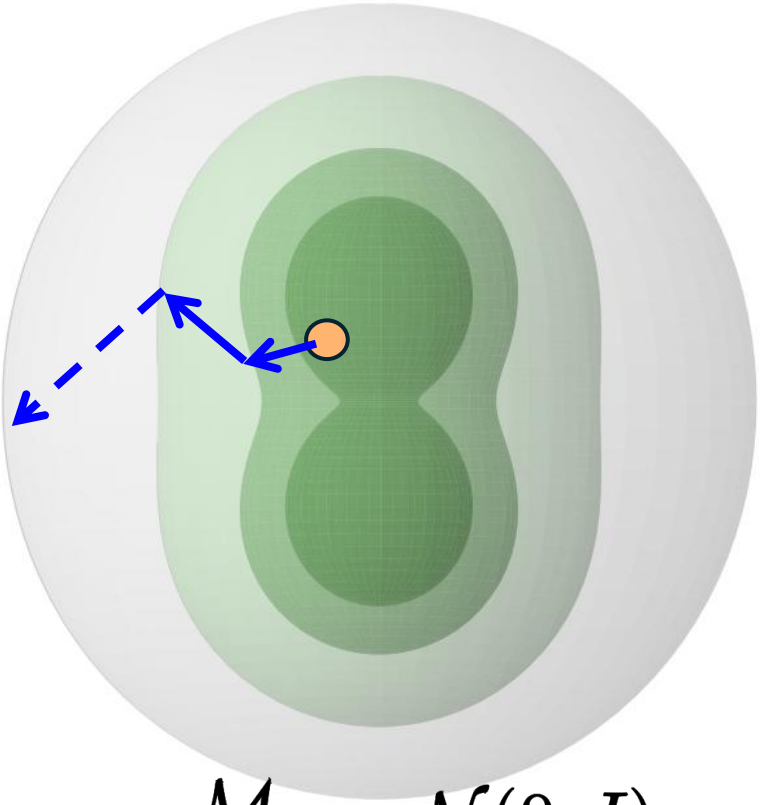
$\mathcal{M}_0$



$\mathcal{M}_0 \rightarrow \mathcal{M}_1$



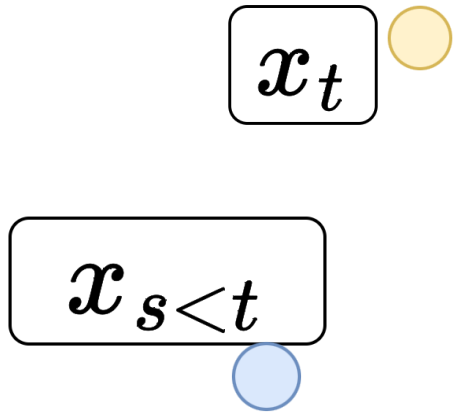
$\mathcal{M}_1 \rightarrow \mathcal{M}_2$



$\mathcal{M}_T \sim \mathcal{N}(0, I)$

Limiting behaviour of model estimates

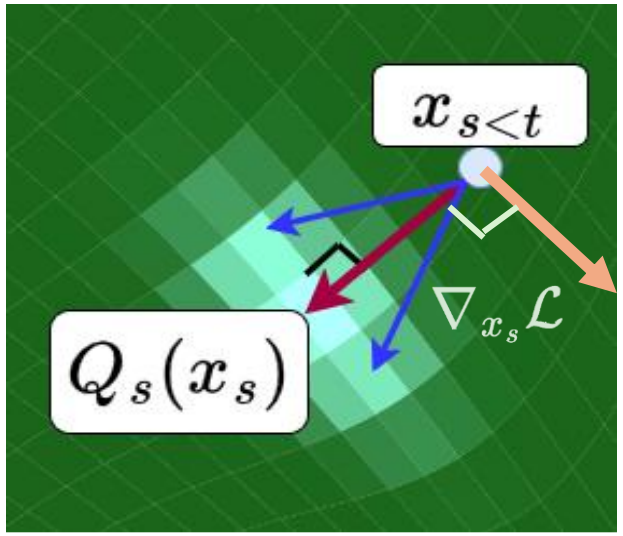
- Theorem 3.1 (Informal): "*At small noise levels, models converge to the* ***orthogonal projection*** *onto the manifold*"

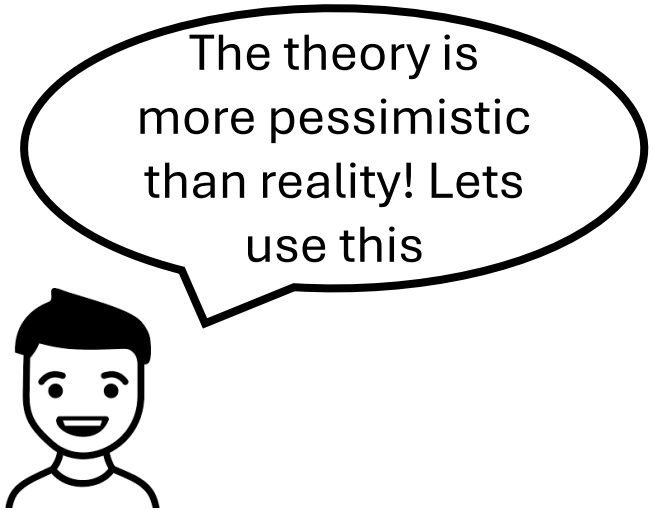


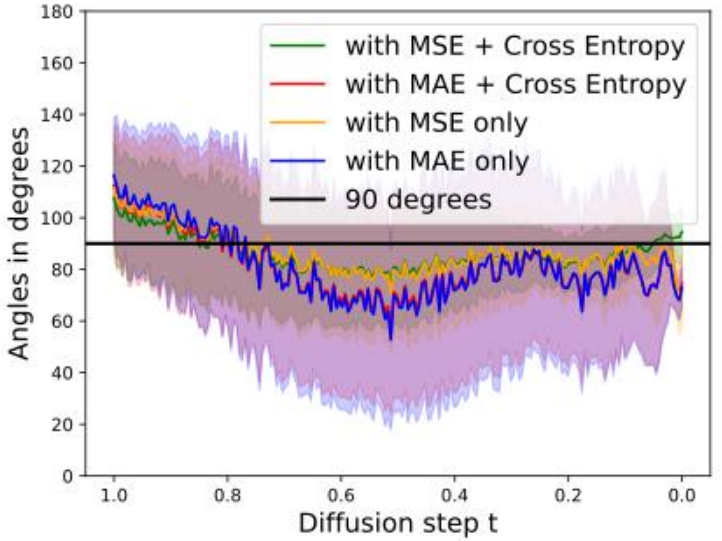
Limiting behaviour of gradients

- Theorem 3.2 (Informal): *"When the model projects orthogonally onto the manifold, the gradients from **any** differentiable loss w.r.t become tangential to the manifold."*

x_t







Conclusion

AI-native data lakes need both in-lake inference and controlled data generation.

Direction A — In-lake LLM inference

Compile LLM computation graphs into SQL.
Execute directly in relational engines.
Use database optimization for out-of-core, CPU-only settings.

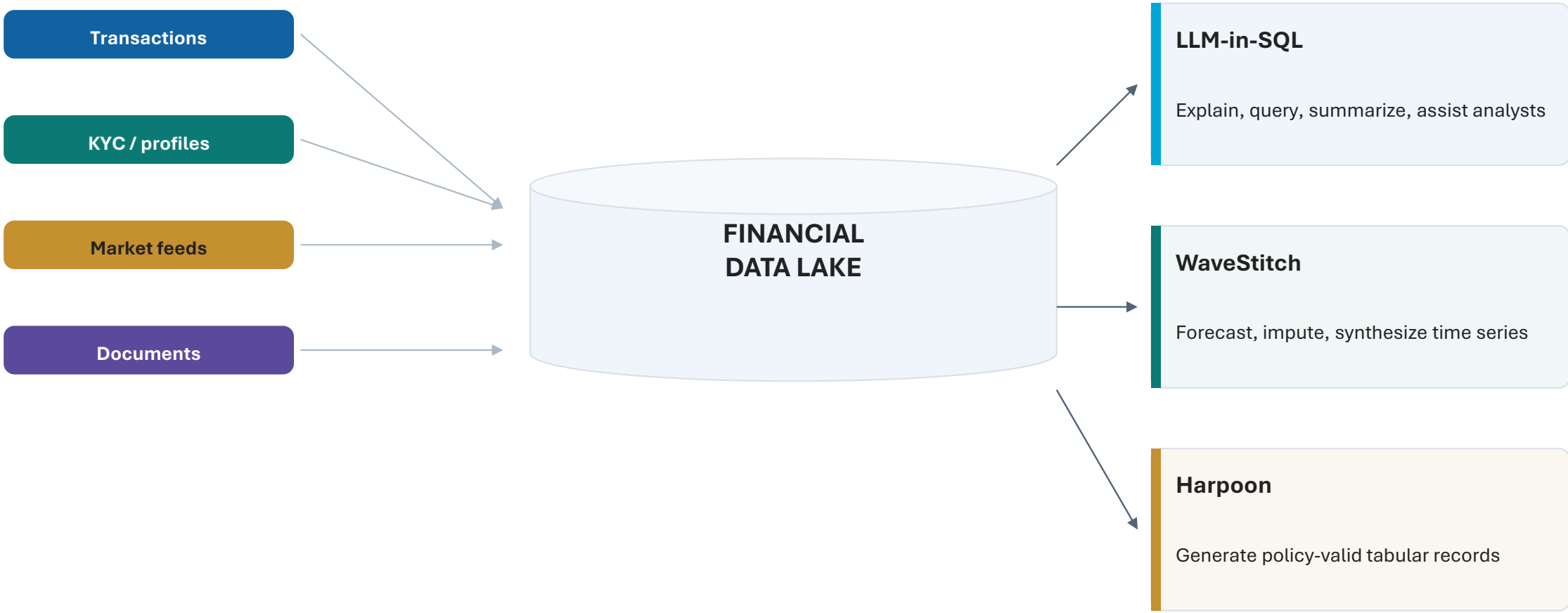
Direction B — Conditional generation

Use diffusion models as data operators.
Impute, forecast, synthesize, and stress-test under metadata, observations, and constraints.

Both directions reduce unnecessary data movement.

Together: AI operators inside a financial data lake

LLMs and diffusion models play complementary roles.



Governance layer: catalog · access control · lineage · quality checks · audit logs

Takeaways

The next data lake is an AI execution and governance platform.

1. Data lakes are evolving

From passive storage to active AI platforms with metadata, governance, and model-aware workflows.

2. LLM inference can be a DB workload

TranSQL+ compiles neural operators into SQL and uses relational engines for controlled local inference.

3. Diffusion models can be data operators

WaveStitch and Harpoon generate, impute, forecast, and synthesize data under conditions.

4. Finance is the forcing function

Privacy, latency, auditability, data quality, and control are not add-ons; they are design requirements.

The lake should execute, constrain, monitor, and govern AI over data.