Toward Temporal Attribution Analytics in Dataflows

Chrysanthi Kosyfaki
Hong Kong University of Science and Technology
Hong Kong SAR, China
ckosyfaki@cse.ust.hk

Ruiyuan Zhang
Hong Kong Generative AI R&D Center
Hong Kong SAR, China
zry@hkgai.org

Nikos Mamoulis
University of Ioannina
Ioannina, Greece
nikos@cs.uoi.gr

Xiaofang Zhou
Hong Kong University of Science and Technology
Hong Kong SAR, China
zxf@cse.ust.hk

ABSTRACT

Data provenance (the process of determining the origin and derivation of data outputs) has applications across multiple domains including explaining database query results and auditing scientific workflows. Despite decades of research, provenance tracing remains challenging due to its high computational cost and storage requirements. In streaming systems such as Apache Flink, fine-grained provenance graphs can grow super-linearly with data volume, posing significant scalability challenges. We define temporal attribution, a new lightweight form of provenance, appropriate for certain tasks, such as monitoring dependencies between system components over time quantitatively. Temporal attribution enables time-focused analysis that does not require fine-grained, tuple-level dependency meta-data. Inspired by volume-based provenance tracking in Temporal Interaction Networks (TINs), we demonstrate TINs' applicability in succinctly modeling quantified data exchanges between dataflow operators in stream data processing systems and in processing workflows, in general, over time. We classify data into discrete and liquid types, define five temporal provenance query types, and propose a state-based indexing approach. Our vision outlines research directions toward making this new form of temporal attribution a practical tool for large-scale dataflow analytics.

PVLDB Reference Format:
Chrysanthi Kosyfaki, Ruiyuan Zhang, Nikos Mamoulis, and Xiaofang Zhou. Toward Temporal Attribution Analytics in Dataflows. PVLDB, 19(): XXX-XXX, 2026.
doi:XX.XX/XXX.XX

1 INTRODUCTION

Data provenance (also known as data lineage) refers to the process of identifying the origin and transformations of data throughout its lifecycle [5, 6, 10, 11]. It is a fundamental concept in modern data management, enabling transparency, trust, and accountability in data-driven systems [23, 39]. In relational databases [12, 26, 33, 59, 60, 62], provenance can explain the results of complex SQL queries, supportiing query debugging [19, 41, 45, 46, 67, 73, 78, 83], view maintenance, and fine-grained access control. In distributed and streaming systems [16, 22, 48–50], it helps model and trace large-scale streaming dataflows for fault recovery and performance optimization. Its importance extends to multiple domains: in financial networks, provenance helps detect illicit activities and trace suspicious transactions; in cybersecurity, it identifies malicious behaviors linked to IP addresses; in healthcare, it ensures compliance and reproducibility by tracking the sources of clinical data; in AI, it can validate outputs generated by large language models (LLMs) and audit workflows in scientific experiments.

Fine-grained provenance, the most detailed form of data provenance, identifies which source tuples contributed to a given output of a workflow (and how) [13, 21, 22, 24, 28, 42, 45, 47–50, 56, 58, 69, 70, 74, 75, 79]. Despite decades of research, efficiently tracking and storing fine-grained provenance information remains a major challenge, due to its high computational cost and storage overhead. For instance, consider a provenance graph that captures record-level lineage in a modern distributed streaming system such as Apache Flink or Spark. Unlike the static job graph, the provenance graph expands as data flows through the system. Its size can grow super-linearly with data volume especially for operations like joins and aggregations [29]. This growth introduces severe scalability challenges in memory/storage and network traffic; the latency of lineage tracking can be too high in environments processing millions of events per second. Overall, fine-grained provenance is space demanding, expensive to track and could be overwhelming to users, especially in workflows that process huge data volumes and need to be monitored for long time periods. In addition, tuple-level provenance may also be the wrong unit of information when comparing provenance between outputs that are produced at different times; the tuples that contributed to these outputs could be totally different, so the only way to compare them is based on the sources (or paths) and the volume of data from them that affected the outputs. At the other extreme, coarse-grained lineage systems (i.e., table-level or partition-level lineage used for data governance) operate at far too coarse a granularity [30, 55]. They track which datasets feed which datasets across an organization, but they do not quantize the contribution of a dataset to a given output nor do they consider the data processing workflow.

Motivated by the above, we propose the new concept of *temporal attribution*, which computes *how much* each of the dataflow sources contributed to a specific output. In its generalized form, attribution quantizes the contribution of sources to *groups of output tuples*. By comparing this contribution over time, we can identify changes in

Proceedings of the VLDB Endowment, Vol. 19, No. 10 ISSN 2150-8097.
doi:XX.XX/XXX.XX

system behavior and monitor load distribution between operators and processing paths. Attribution has already been considered as a tool for interpreting neural network results [65, 80]; we are the first to suggest its use for the outputs of streaming dataflows. Previous work on *temporal provenance* [9, 17, 57, 72, 82] attaches timestamps to either coarse-grained or fine-grained provenance information, capturing when derivations occurred. This allows time-focused provenance representation and tracking, e.g., "which sources contributed to this output between 2PM and 3PM?", however, it does not address temporal attribution (e.g., how much each source contributed).

To facilitate temporal attribution, inspired by previous work on network flow analytics in Temporal Interaction Networks (TINs) [34–36], we propose modeling workflows as TINs, where nodes exchange (and possibly transform) data over time. Since our focus is quantitative provenance (i.e., attribution), we discretize time, and consider the workflow operators as processing groups of tuples in regular time intervals (e.g., processing is summarized for every period of 1 second), instead of monitoring the processing of individual tuples at a fine time granularity. This abstraction of the dataflow as a TIN facilitates the temporal attribution tracking that we are targeting and reduces the necessary meta-data required for it. Modeling the dataflow as a TIN achieves state-based compression at its core: instead of recording per-tuple dependencies, we do this between temporal states that aggregate data over time windows. This exploits the observation that, in many streaming workloads, large data volumes are transferred without materially changing the aggregate attribution structure (e.g., during window accumulation). By representing such periods with a small number of states, we achieve substantial compression while still preserving sufficient information to answer temporal attribution queries, without reconstructing fine-grained histories. Attribution is especially appropriate in cases where data are *liquid*, i.e., become indistinguishable when merged or split. Such data are natural in some applications (e.g., financial workflows where nodes exchange liquid assets), while data anonymization may also be enforced in distributed environments where data privacy is a concern [51, 54, 76, 77]. **Our contributions include:** ❶ demonstrating how TINs provide a model for dataflows that facilitates temporal attribution across applications that process data in a workflow; ❷ classifying data into discrete/identifiable types (where transferred data maintain identity) versus liquid types (where quantities become indistinguishable after merging); ❸ formalizing five temporal attribution query types: backward attribution (where-from), forward attribution (where-to), temporal lineage (when-contributed), flow lineage (how-much-through), and versioning attribution (how-changed) that leverage TINs' temporal structure; and ❹ proposing temporal attribution indexing based on vertex state sequences that enables efficient query evaluation without reconstructing entire interaction histories. By doing so, this agenda aims to help researchers develop scalable solutions for efficiently tracking temporal attribution.

Roadmap: The rest of the paper is organized as follows: Section 2 provides an overview of data provenance and TINs. Section 3 discusses TINs and demostrates their application to modeling dataflows. Section 4 analyzes different data classes and their impact on temporal attribution tracking. Section 5 presents our temporal attribution indexing and queries. Section 6 reviews related work. In Section 7 we propose a research agenda and Section 8 concludes the paper.

2 DATA PROVENANCE AND TINS

This section provides the necessary background. We begin with the classic provenance models and their limitations in dynamic settings, then present the Temporal Interaction Network formalism from prior work that we build upon.

2.1 Data Provenance

Data provenance captures the origin and derivation of data. The seminal work by Buneman et al. [10, 11, 26] defined three core provenance semantics for relational databases: *Where-provenance* identifies which input tuples contributed to an output tuple. For a query result, where-provenance returns the set of source tuples that appear in at least one derivation of the result. *Why-provenance* identifies which input tuples are necessary for an output. It returns the minimal sets of source tuples sufficient to derive the output, corresponding to witness bases. *How-provenance* provides an algebraic expression showing how output values depend on input values. Green et al. [25] formalized the concept of *provenance semirings*, which annotate tuples with polynomial expressions tracking their derivation.

These models were designed for *snapshot queries* over static databases, where provenance can be computed by inspecting the query plan and tracing tuple dependencies. However, they face fundamental challenges in *continuous* systems. First, traditional provenance does not capture *when* data contributions occurred; e.g., in a streaming system, where-provenance cannot distinguish between contributions at different time periods. Second, they do not handle operators that maintain state across multiple inputs, such as windowed aggregations or stateful joins common in stream processing. Third, quantities are not tracked. In systems where data represents quantities (event counts, monetary values, network volumes), provenance must track *how much* each source contributed, not just *whether* it contributed. The above limitations can be addressed by fine-grained provenance mechanisms for stream data processing systems, like Ariadne [22] and Genealog [48]; however, at the cost of tracking tuple-level dependencies between produced outputs throughout the workflow.

2.2 Temporal Interaction Networks

Temporal Interaction Networks (TINs) are a graph-based formalism for modeling time-varying data flows [34, 36]. A TIN is a triple $G = (V, E, R)$ where V is a set of vertices, $E \subseteq V \times V$ is a set of directed edges, and R is a set of interactions. Each interaction $r \in R$ is a quadruple (r_s, r_d, r_t, r_q) with source vertex r_s, destination vertex r_d, such that $(r_s, r_d) \in E$, timestamp r_t, and transferred quantity $r_q \in \mathbb{R}^+$. Each vertex maintains a time-varying buffer $B_v(t)$ representing the accumulated quantity in v at time t. Interactions increase destination buffers and decrease source buffers, enabling TINs to model both transient flows and accumulation.

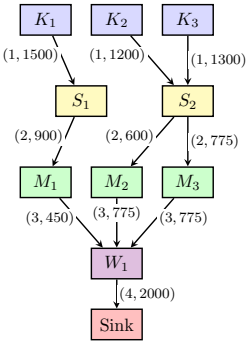
TINs differ from traditional temporal graphs in three key aspects: ❶ *explicit quantity tracking*: each interaction transfers a specific quantity, it is not just a binary connection or an event occurrence; ❷ *buffer state*: vertices maintain accumulated quantities over time,

capturing stateful behavior; ❸ *flow semantics*: interactions represent data transfers, where a quantity leaves one vertex and arrives at another.

These properties make TINs well-suited for modeling dataflows for temporal attribution tracking. The temporal dimension captures *when* data flows occurred, the quantity dimension captures *how much* data flowed, and the buffer mechanism captures stateful operators common in streaming systems. Ref. [34] shows how to propagate meta-data in TINs for provenance tracking of *liquid* data (e.g., money) in financial exchange networks, under certain models governing data selection from buffers for propagation. In the following sections, we demonstrate how to leverage TINs for efficient temporal attribution representation and tracking in data stream systems.

3 TINS FOR TEMPORAL ATTRIBUTION

TINs can be used to model dataflows in a way that facilitates temporal attribution. Specifically, we model each dataflow operator as a node in a TIN and summarize processing in regular time intervals (e.g., every second). We consider aggregate data transfers between connected operators in consecutive time windows and each such aggregate transfer is represented as a TIN interaction (src, dst, t, q), where q is the number of tuples (i.e., aggregated quantity) exchanged. Operators are classified based on how they transform incoming interactions into outgoing ones: (i) stateless operators (e.g., map/filter) apply a transformation function that preserves or scales quantities (e.g., filtering reduces q); (ii) partitioning operators (e.g., keyBy/shuffle) split incoming quantities (groups of tuples) to multiple outgoing interactions whose quantities sum to the input; and (iii) stateful operators (e.g., windows/joins) accumulate incoming quantities within time windows and emit new quantities when each window triggers, reflecting aggregated outputs. Attribution meta-data follow these transformations by propagating and aggregating quantities across interactions and by tracking dependencies between them, thus, forming an *attribution graph*.



(a) TIN

Time	Interaction	Type
$t = 1$	$(K_1, S_1, 1500)$ $(K_2, S_2, 1200)$ $(K_3, S_2, 1300)$	Kafka sources
$t = 2$	$(S_1, M_1, 900)$ $(S_2, M_2, 600)$ $(S_2, M_3, 775)$	Round-robin
$t = 3$	$(M_1, W_1, 450)$ $(M_2, W_1, 775)$ $(M_3, W_1, 775)$	KeyBy shuffle
$t = 4$	$(W_1, \text{Sink}, 2000)$	To sink

(b) Interactions

Figure 1: TIN-based provenance framework.

As an example of how to model a dataflow as a TIN, consider a Flink job, illustrated in Figure 1(a) that processes user activity and transactions streams from an e-commerce platform to compute the total number of items purchased per product over tumbling windows. Nodes inlcude Kafka sources (K_1, K_2, K_3), source operators (S_1, S_2), map operators (M_1, M_2, M_3), a window operator (W_1), and a sink. The system ingests three event streams from Kafka partitions K_1, K_2, K_3: ViewEvent: (userID, productID, timestamp) capturing product page visits, AddToCartEvent: (userID, productID, quantity, timestamp) capturing cart additions, and PurchaseEvent: (userID, productID, quantity, price, timestamp) capturing completed purchases. Source operators S_1, S_2 ingest and unify these streams, after which map operators transform events into quantitative records: M_1 filters view events and emits non-contributing signals, while M_2 and M_3 map add-to-cart and purchase events, respectively, into (productID, quantity) pairs. These records are then partitioned by key and routed to a window operator W_1, which aggregates the total quantity per product within each time window and emits the result to the sink.

Figure 1(b) shows TIN interactions ordered by discretized/abstracted time. At time window $t = 1$, Kafka sources ingest events: $(K_1, S_1, 1500)$, $(K_2, S_2, 1200)$, $(K_3, S_2, 1300)$. At $t = 2$, source operators apply round-robin partitioning: $(S_1, M_1, 900)$, $(S_2, M_2, 600)$, $(S_2, M_3, 775)$. At $t = 3$, map operators perform KeyBy shuffle to the window: $(M_1, W_1, 450)$, $(M_2, W_1, 775)$, $(M_3, W_1, 775)$. At $t = 4$, the window fires and sends aggregated results to the sink: $(W_1, \text{Sink}, 2000)$.

During the time interval $[3, 4)$, W_1 receives 2000 tuples from M_1, M_2, M_3. Instead of storing individual provenance back-links for all these tuples we store three attribution back-links to the interactions of the three mappers: $(M_1, 3, 450)$, $(M_2, 3, 775)$, and $(M_3, 3, 775)$. This *state-based compression* is particularly effective for windowed aggregations where thousands of events arrive between window boundaries but produce only a few state transitions. For operator-internal logic (e.g., auditing which specific tuples were removed by filtering or identifying the exact set of records merged during an aggregation), traditional fine-grained provenance remains necessary. Our model does not track the identity of individual records within the dataflow, but captures (i) the resulting changes in data volume and flow rates and (ii) aggregate dependencies between states of (consecutive) operators. This is done by linking consecutive interactions at different nodes if they correspond to the same groups of tuples. For example, $(K_1, S_1, 1500)$ is linked to $(S_1, M_1, 900)$.

TINs vs. Traditional Provenance Graphs. When the goal is temporal attribution tracking, TINs present key advantages in modeling dataflow graphs (at a coarse time granularity). First, although quantitative information can be aggregated from contribution graphs used by fine-grained provenance systems such as Genealog [48], this bears substantial computational cost. Second, due to the super-linear space complexity of contribution graphs, we cannot afford to keep them for a long time, for the sake of temporal quantitative dependency analytics over historical data. Even with state-of-the-art optimizations designed to minimize per-tuple metadata [49], the number of nodes becomes too high to store, making long-term monitoring unfeasible. Lazy fine-grained provenance schemes [22, 74] have reduced cost, but also require partially replaying the execution of the stream. TINs encode time in each coarse-level interaction, with our vision focusing on enabling direct index-based retrieval. Second, traditional fine-grained provenance graphs do not scale with data volume; for instance, a Flink job processing 1 million events per second would generate 3.6 billion provenance nodes per hour at the ingestion layer alone. Even with state-of-the-art

optimizations designed to minimize per-tuple metadata [49], the number of nodes becomes too high to store, making long-term monitoring unfeasible. Second, traditional fine-grained provenance graphs do not scale with data volume; for instance, a Flink job processing 1 million events per second would generate 3.6 billion provenance nodes per hour at the ingestion layer alone. Even with state-of-the-art optimizations designed to minimize per-tuple metadata [49], the number of nodes becomes too high to store, making long-term monitoring unfeasible. Second, traditional fine-grained provenance graphs do not scale with data volume; for instance, a Flink job processing 1 million events per second would generate 3.6 billion provenance nodes per hour at the ingestion layer alone. Even with state-of-the-art optimizations designed to minimize per-tuple metadata [49], the number of nodes becomes too high to store, making long-term monitoring unfeasible. TINs compress flows into temporal states: instead of numerous of individual tuples, we store aggregated interactions. Third, TINs explicitly capture quantities, enabling attribution-based analysis. In the Flink example, we can use the group-to-group quantity links to answer "How much data flowed from K_2 to W_1?". When detailed contribution graphs [48] are used, this requires traversing numerous paths.

Limitations of State Compression. While state-based compression may achieve dramatic space reduction for typical workloads, it becomes less effective when an operator produces quantities (groups of tuples) too often, or the groups are small by nature. Hence, worst-case scenarios happen when operator states change frequently (for example, the state changes for each incoming tuple as in sliding windows) or when states (groups) contain very few tuples. In general, compression introduces a trade-off: while quantitative contributions are preserved, fine-grained interaction-level details (e.g., individual events, ordering within a window, or per-record lineage) are not retained. As a result, fine-grained provenance that requires such information (e.g., reconstructing the exact sequence of events leading to a specific output, or distinguishing between multiple interaction paths that contribute identically at the aggregate level) cannot be served by the TIN representation and compressed states alone. Finally, the accuracy of temporal attribution queries presented in Section 5.1 relies on the time granularity of the aggregated quantities propagated through the dataflow modeled as a TIN. For temporal queries that ask for information at a finer granularity, the compressed information naturally can only give approximate estimates, in the same manner as histograms in selectivity estimation.

4 DISCRETE VS. LIQUID DATA

Data classes may affect the options for temporal attribution and the dependencies between states. *Discrete data* (data items with unique identities) allows straightforward path-based provenance, while *liquid data* (indistinguishable quantities that merge and split) add challenges to temporal attribution mechanisms.

Discrete, Identifiable Data. It consists of individual items (e.g., tuples) that maintain a unique identity throughout their lifecycle. While these items are transferred and transformed between workflow nodes, it is possible to track tuple-level dependencies, as streaming provenance systems do [22, 48, 49]. Hence, fine-grained provenance mechanisms can be used for temporal attribution, however, at a higher cost compared to modeling the dataflow as a TIN that captures transferred quantities instead of individual items.

Liquid Data. It refers to quantities that are not identifiable and cannot be tracked after transformations such as splitting, merging, etc. In streaming dataflows, liquid data arises when numeric values are extracted from tuples and merged into aggregate totals. After being redistributed across operators, these values become *un-named* quantities within a single sum; the resulting output obscures the origin of each unit, mirroring the ambiguity of merged funds in financial networks. In liquid data, because the origin of an individual output item becomes ambiguous and quantitative lineage becomes possible only for quantities (groups) of output items. As an analogy, an amount of money, originating from one account, can be split across several transactions, merged with other funds, and eventually appear in multiple destinations. In scenarios where unidentifiable tuples are propagated and transformed fine-grained provenance is not even possible (e.g., when explicit links between items at distributed dataflows cannot be established, due to security or privacy constraints [51, 54, 76, 77]), so lineage can only be tracked quantitatively, by means of temporal attribution.

When modeling a dataflow as a TIN, we may think groups of transferred data as liquid because we track aggregated counts rather than individual items. While each item may have a unique ID in the actual system, the TIN models the dataflow at the flow aggregation level, as explained in the previous section. Attribution meta-data do not differ between identifiable and liquid data, as long as the operators can identify how many tuples from a given input at time t contributed to how many tuples to a specific output at time $t + 1$. If liquid data consist of anonymized items, this is possible for all operators. In case where this is not possible (i.e., inputs are merged into indivisible quantities before the operator can process them), contribution propagation assumptions should be applied as in [34].

5 TEMPORAL PROVENANCE INDEXING

After modeling the dataflow as a TIN, to support diverse temporal attribution tasks, we envisage indexing methods that capture both the state evolution of dataflow vertices and attribution information on them over time. The key insight behind our approach is that each vertex in a TIN goes through a sequence of states, where each state corresponds to a time interval during which the vertex's quantities and their dependencies remain unchanged. By temporally indexing these states, we can answer temporal attribution questions without replaying the dataflow.

State-based Representation. Each vertex maintains a sequence of states characterized by time intervals, quantities, and attribution information. New states are created at each time window or when internal operations occur (e.g., windows in stateful operators fire), naturally compressing temporal evolution by grouping periods with identical buffer states.

Compression. The state-based representation provides substantial compression compared to storing fine-grained contribution graphs at the tuple level. The key insight is that we create new states only when quantities at operators (and their dependencies) change.

Attribution Computation. Each state maintains attribution information that captures how much each of the states at previous operators contributes to the current quantity at the state. Attribution can be computed by BFS along these state-to-state links. We may also explicitly propagate attribution information at states as

tuples of the form (origin, time, quantity), indicating that a specific quantity originated from a particular source vertex at a given time (or time interval). This allows us to know the origins/sources of quantities back through the network and their attribution in a compact representation. When a vertex transfers quantities outward, we propagate and update the corresponding attribution data.

Index Structure. The temporal attribution index (TAI) organizes states chronologically for each vertex, enabling efficient time-based retrieval of their buffered quantities and attribution data. For a given vertex v and time t, we can locate the relevant state using a B-tree over the temporal sequence of states, as they are naturally ordered by time. For example, to query W_1's attribution at $t = 3.5$, we perform search over its state sequence and retrieve state s_2, which covers the interval $[3, 4)$. The index should support different query types (e.g., "How much did each source contribute at time t?" or "How did attribution evolve between t_1 and t_2?"). Temporal attribution queries use TAI to find the state(s) of the target vertex that satisfy the temporal query predicate and then (temporally) join them with the states of adjacent vertices along the dataflow. Index maintenance occurs incrementally as new TIN interactions take place. When an interaction modifies a vertex's buffer, we close the current state and create a new one, updating both the buffer quantity and the attribution information based on the interaction.

5.1 What Can We Ask?

We formalize five provenance query types that leverage temporal states in TINs and illustrate them using the Flink pipeline from Figure 1. This query set is not meant to be complete; our goal is to introduce some tools that can facilitate attribution-based analysis.

Q1: Backward Attribution (Where-From). Given sink node s_i and time t, return all ⟨source, time, quantity⟩ tuples showing which origins (and how much) contributed to d's state at time t.

Example: "At $t = 4$, W_1 has 2000 events. What is the attribution of sources to them?" Trace backward: $M_1 \rightarrow 450$, $M_2 \rightarrow 775$, $M_3 \rightarrow 775$ (at $t = 3$). Recursively: $S_1 \rightarrow M_1$ (900), $S_2 \rightarrow M_2$ (600), $S_2 \rightarrow M_3$ (775) at $t = 2$, and ultimately K_1, K_2, K_3 at $t = 1$.

Q2: Forward Attribution (Where-To). Return all downstream destinations that were affected by the input s at time t, and how much was each of them affected.

Example: "At $t = 1$, K_1 ingests 1500 events. Where do they go?" All events affect the Sink via path: $K_1 \rightarrow S_1 \rightarrow M_1 \rightarrow W_1 \rightarrow$ Sink. Forward provenance supports impact analysis.

Q3: Temporal Lineage (When-Contributed). Given vertex v and time window $[t_1, t_2]$, return all sources whose contributions arrived at v during that period.

Example: "Which sources contributed to W_1 between $t = 2$ and $t = 3$?" At $t = 3$: M_1 (450), M_2 (775), M_3 (775).

Q4: Flow Lineage (How-Much-Through). Given source s, sink s_i, and intermediary v, compute the quantity that flowed from s to d via v during time $[t_1, t_2]$.

Example: "How much K_1 data reached W_1 via M_1 in $[3, 4]$?" Path: $K_1 \rightarrow S_1$ (1500), $S_1 \rightarrow M_1$ (900), $M_1 \rightarrow W_1$ (450). Answer: 450.

Q5: Versioning Attribution (How-Changed). Given vertex v and times t_1, t_2, where $t_1 < t_2$, compute the change in attribution data of v from t_1 to t_2.

Example: "How did W_1's attribution change from $t = 3$ to $t = 4$?" At $t = 3$: q: 2000, attribution:$\{M_1 : 450, M_2 : 775, M_3 : 775\}$. At $t = 4$: q: 0, attribution:$\emptyset$. Delta: all sources depleted (window fired).

All five queries support tracing through recursive temporal state lookups. Unlike applying expensive fine-grained provenance mechanisms, modeling the dataflow as a TIN, keeping attribution data with each state, and temporal indexing of states can facilitate fast retrieval of attribution information.

6 RELATED WORK

Temporal attribution computation is closely related to data provenance. Data provenance is well-studied in the research community [4, 7, 8, 14, 15, 18, 22, 27, 31, 32, 34, 38, 43, 53, 57, 63, 64, 66, 71, 77]. We briefly survey related work and position our contributions.

Database Provenance. Buneman et al. [10, 11] introduced where/why provenance; Green et al. [25, 26] developed the semiring framework. Surveys [20, 60, 62] provide comprehensive overviews. ProvSQL [61, 68] implements provenance tracking in PostgreSQL. While these approaches excel at static query provenance, they do not address continuous data flows or provide temporal indexing mechanisms.

Streaming Provenance. Ariadne [22] introduces operator instrumentation for fine-grained provenance tracking in data stream processing systems, extending the Borealis stream processing engine [1]. An independent provenance-focused work, also named Ariadne [52], presents a system for declaratively customizing provenance capture and querying in large-scale graph analytics on vertex-centric graph processing platforms. Genealog [48] provides fine-grained backward provenance for data streaming by annotating every output tuple with a constant-size provenance token that encodes which source tuples contributed to it. Ananke [49] extends Genealog to deliver forward provenance; a continuously maintained graph that records, for each source tuple, which downstream outputs it has contributed to so far and whether the source tuple can still influence future outputs. [50] is the first framework for why-not provenance in data streaming, explaining why a particular output was produced or why an expected output was absent by identifying the minimum subset of source data responsible. [74] suggests a lazy provenance mechanism based on checkpointing operator state; when provenance is requested, the dataflow is replayed from the nearest checkpoint, avoiding the throughput degradation of eager approaches. All of these systems track *per-tuple provenance*: given an output tuple, they identify the set of source tuples that produced it. On the other hand, our vision is to add temporal attribution functionality to streaming systems, where the objective is to measures the contribution of each source to a given output at a coarse level (e.g., all tuples generated within a time window). For this, the dataflow is modeled as a TIN where quantities (e.g., groups of tuples) flow together with lightweight attribution meta-data. Additionally, none of these systems defines our temporal query types (backward/forward how-much, versioning, flow lineage, path-based provenance) and maintains a respective temporal index.

Provenance data compression. Compression has been used within provenance frameworks. For example, Genealog [48] replaces explicit lineage graphs with compact per-tuple annotations. Other approaches [3, 40] generate concise representations by trading accuracy for compactness (e.g., via pattern-based or top-k summaries).

In distributed settings, provenance compression techniques [16] reduce storage and communication costs by compacting provenance graphs across nodes. Differently from all these approaches, our proposal incorporates compression directly into the data model via temporal state aggregation, representing flows and attribution at a coarser level while still supporting temporal attribution queries without reconstructing fine-grained lineage.

Temporal Provenance. TAP/DTaP [81, 82] use distributed Datalog to capture temporal provenance for distributed protocol debugging. Zeno [72] diagnoses performance problems using temporal provenance. We differ fundamentally by: (1) applying TINs to structurally model temporal provenance with explicit quantity tracking in data management contexts, (2) distinguishing discrete vs. liquid data classes that require different provenance semantics, (3) achieving compression via state-based indexing that groups consecutive time periods with identical buffer states, and (4) providing five temporal query types (backward, forward, temporal lineage, flow lineage, versioning) that leverage TINs' temporal structure.

Graph Provenance. Several efforts [2, 37, 44] model provenance using graph structures but typically represent static relationships. Prior work on TINs [34–36] examines flow computation and provenance tracking for liquid data in TINs; we extend this with the definition of temporal attribution, state-based compression, provenance tracking in stream networks, indexing for provenance, and support for a wide range of temporal attribution queries.

Our approach differs from all prior work by combining three elements: ❶ explicit attribution tracking in dataflows and temporal graphs; ❷ state-based compression achieving huge storage reductions for windowed workloads; and ❸ a query model supporting five temporal provenance query types (backward, forward, temporal lineage, flow lineage, versioning). To our knowledge, no existing system provides all these capabilities together.

7 THE PATH FORWARD

Our vision of TIN-based temporal attribution opens several research directions. We organize our agenda around three core challenges.

***Attribution Query Optimization challenge*:** Temporal attribution queries (Section 5.1) can be modeled as database queries over temporal tables that store attribution meta-data for each state. To obtain the attribution graph for a particular state, we should perform temporal joins with the states of previous nodes. Hence, a query optimization challenge arises.

Research Questions: ❶ What attribution-specific statistics (state transition frequency, state dependencies, temporal correlation) can be used for query planning? ❷ When can optimizers rewrite recursive provenance queries to skip deep traversal? For instance, a backward provenance query with depth limit k only requires the most recent k state transitions, avoiding historical states. ❸ How should systems handle skewed provenance distributions where some vertices have hundreds of contributing sources?

Approach: Represent state-based indices as relational tables, enabling engines like DuckDB to leverage vectorized execution and compression. Design tiered storage (hot states in memory, warm on SSD, cold in object storage) balancing latency and cost.

***Adaptive Compression and Unified Models challenge*:** State-based compression effectiveness varies dramatically across workloads; windowed streaming achieves million-fold compression while high-frequency trading sees diminished benefits.

Research Questions: ❶ How does query performance vary with time granularity used in compression? ❷ Can we predict optimal compression granularity for a given workflow and attribution requirements? ❸ When should systems transition between fine-grained provenance and temporal attribution?

Approach: Develop learned compression policies that observe workload characteristics (query temporal resolution, update rates, attribution analysis requirements) and dynamically adjust state granularity per vertex. Design unified models supporting both fine-grained provenance and attribution, enabling automatic conversion at operator boundaries.

***Distributed Attribution Indexing challenge*:** In distributed workflows that process big data tasks, centralized storage and indexing of temporal attribution data may not be feasible.

Research Questions: ❶ What partitioning strategies (by vertex ID, time ranges, or query patterns) minimize cross-machine temporal attribution queries? Time-based partitioning enables efficient temporal queries but splits high-degree vertices across partitions, while vertex-based partitioning co-locates attribution data for neighboring nodes, but complicates time-range queries. Can hybrid (vertex, time_bucket) partitioning balance both? ❷ What consistency models balance staleness versus overhead? ❸ How can vector clocks maintain causal ordering across replicas?

Approach: Extend states with causal metadata, design gossip protocols for asynchronous synchronization, and develop partition-aware query planners.

8 CONCLUSIONS

Fine-grained provenance mechanisms for streaming systems face scalability challenges as contribution graphs grow superlinearly with data volume. When it comes to analyzing historical data, the problem escalates, as we need to maintain fine-grained provenance meta-data over a long history. To alleviate this, we define the concept of temporal attribution for modeling the impact of inputs (sources) to the output in dataflows at a coarser, quantitative level compared to tuple-level provenance. We suggest Temporal Interaction Networks (TINs) as a way to model data flow at a coarser, compressed level over time. At the same time, we suggest the propagation of temporal attribution meta-data together with the flow capturing structural and temporal aspects for richer analysis. Through examples in streaming workflows, we classify data into identity-preserving and liquid types, define five temporal query types, and propose a state-based index for compressed querying. Our vision opens research directions in query optimization for temporal attribution analytics, adaptive compression, unified models for diverse data, and distributed indexing and advances temporal provenance from a debugging tool to a broad data management primitive.

ACKNOWLEDGMENTS

This work was supported by Hong Kong Research Grants Council (grant T43-513/23-N), HKUST-WeBank Joint Laboratory (grantWEB24EG01).

REFERENCES

[1] Daniel J. Abadi, Yanif Ahmad, Magdalena Balazinska, Ugur Çetintemel, Mitch Cherniack, Jeong-Hyon Hwang, Wolfgang Lindner, Anurag Maskey, Alex Rasin, Esther Ryvkina, Nesime Tatbul, Ying Xing, and Stanley B. Zdonik. 2005. The Design of the Borealis Stream Processing Engine. In *Second Biennial Conference on Innovative Data Systems Research, CIDR, Asilomar, CA, USA, January 4-7, Online Proceedings*. www.cidrdb.org, 277–289.

[2] Umut Acar, Peter Buneman, James Cheney, Jan Van den Bussche, Natalia Kwasnikowska, and Stijn Vansummeren. 2010. A graph model of data and workflow provenance.

[3] Eleanor Ainy, Pierre Bourhis, Susan B Davidson, Daniel Deutch, and Tova Milo. 2015. Approximated summarization of data provenance. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. 483–492.

[4] Daniel Alabi, Sainyam Galhotra, Shagufta Mehnaz, Zeyu Song, and Eugene Wu. 2025. Privacy and Security in Distributed Data Markets. In *Companion of the International Conference on Management of Data*. 775–787.

[5] Abdullah Hamed Almuntashiri, Luis-Daniel Ibàñez, and Adriane Chapman. 2024. LLMs for the post-hoc creation of provenance. In *2024 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. IEEE, 562–566.

[6] Abdullah Hamed Almuntashiri, Luis-Daniel Ibáñez, and Adriane Chapman. 2025. Using LLMs to infer provenance information. In *Proceedings of the ProvenanceWeek 2025*. 1–10.

[7] Mohamed Jehad Baeth and Mehmet S Aktas. 2019. Detecting misinformation in social networks using provenance data. *Concurrency and Computation: Practice and Experience* 31, 3 (2019), e4793.

[8] Geoffrey Barbier, Zhuo Feng, and Pritam Gundecha. 2013. *Provenance data in social media*. Morgan & Claypool Publishers.

[9] Seyed-Mehdi-Reza Beheshti, Hamid Reza Motahari-Nezhad, and Boualem Benatallah. 2012. Temporal provenance model (TPM): model and query language. *arXiv preprint arXiv:1211.5009* (2012).

[10] Peter Buneman, Sanjeev Khanna, and Wang Chiew Tan. 2001. Why and Where: A Characterization of Data Provenance. In *Database Theory - ICDT, 8th International Conference, London, UK, January 4-6, Proceedings (Lecture Notes in Computer Science)*, Vol. 1973. Springer, 316–330.

[11] Peter Buneman, Sanjeev Khanna, and Wang Chiew Tan. 2002. On Propagation of Deletions and Annotations Through Views. In *Proceedings of the Twenty-first ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 3-5, Madison, Wisconsin, USA*. ACM, 150–158.

[12] Peter Buneman and Wang-Chiew Tan. 2007. Provenance in databases. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*. 1171–1173.

[13] Adriane Chapman, Luca Lauro, Paolo Missier, and Riccardo Torlone. 2024. Supporting better insights of data science pipelines with fine-grained provenance. *ACM Transactions on Database Systems* 49, 2 (2024), 1–42.

[14] Adriane Chapman, Paolo Missier, Giulia Simonelli, and Riccardo Torlone. 2020. Capturing and querying fine-grained provenance of preprocessing pipelines in data science. *Proceedings of the VLDB Endowment* 14, 4 (2020), 507–520.

[15] Adriane P Chapman, Hosagrahar V Jagadish, and Prakash Ramanan. 2008. Efficient provenance storage. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. 993–1006.

[16] Chen Chen, Harshal Tushar Lehri, Lay Kuan Loh, Anupam Alur, Limin Jia, Boon Thau Loo, and Wenchao Zhou. 2017. Distributed provenance compression. In *Proceedings of the ACM International Conference on Management of Data*. 203–218.

[17] Peng Chen, Beth Plale, and Mehmet S Aktas. 2012. Temporal representation for scientific data provenance. In *2012 IEEE 8th International Conference on E-Science*. IEEE, 1–8.

[18] Susan B Davidson, Tova Milo, and Sudeepa Roy. 2013. A propagation model for provenance views of public/private workflows. In *Proceedings of the 16th International Conference on Database Theory*. 165–176.

[19] Daniel de Oliveira, Flavio Costa, Vítor Silva, Kary ACS Ocaña, and Marta Mattoso. 2014. Debugging Scientific Workflows with Provenance: Achievements and Lessons Learned.. In *SBBD*. 67–76.

[20] Boris Glavic. 2021. Data provenance: origins, applications, algorithms, and models. *Foundations and Trends in Databases* 9, 3-4 (2021), 209–441.

[21] Boris Glavic, Kyumars Sheykh Esmaili, Peter M. Fischer, and Nesime Tatbul. 2011. The Case for Fine-Grained Stream Provenance. In *Proceedings BTW- Workshops und Studierendenprogramm, 1. März , Kaiserslautern, Germany*. 58–61.

[22] Boris Glavic, Kyumars Sheykh Esmaili, Peter Michael Fischer, and Nesime Tatbul. 2013. Ariadne: managing fine-grained provenance on data streams. In *The 7th ACM International Conference on Distributed Event-Based Systems, DEBS '13, Arlington, TX, USA - June 29 - July 03, 2013*. ACM, 39–50.

[23] Todd J Green, Zachary G Ives, Grigoris Karvounarakis, and Val Tannen. 2010. Provenance in ORCHESTRA. (2010).

[24] Todd J. Green, Grigoris Karvounarakis, Zachary G. Ives, and Val Tannen. 2007. Update Exchange with Mappings and Provenance. In *Proceedings of the 33rd International Conference on Very Large Data Bases, University of Vienna, Austria, September 23-27*. ACM, 675–686.

[25] Todd J Green, Grigoris Karvounarakis, and Val Tannen. 2007. Provenance semirings. In *Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. 31–40.

[26] Todd J Green and Val Tannen. 2017. The semiring framework for database provenance. In *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*. 93–99.

[27] Pritam Gundecha, Zhuo Feng, and Huan Liu. 2013. Seeking provenance of information using social media. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. 1691–1696.

[28] Mohammad Rezwanul Huq, Andreas Wombacher, and Peter M. G. Apers. 2010. Facilitating fine grained data provenance using temporal data model. In *Proceedings of the 7th Workshop on Data Management for Sensor Networks, in conjunction with VLDB, DMSN, Singapore, September 13 (ACM International Conference Proceeding Series)*. ACM, 8–13.

[29] Matteo Interlandi, Kshitij Shah, Sai Deep Tetali, Muhammad Ali Gulzar, Seunghyun Yoo, Miryung Kim, Todd Millstein, and Tyson Condie. 2015. Titian: Data provenance support in spark. In *Proceedings of the VLDB Endowment International Conference on Very Large Data Bases*, Vol. 9. 216.

[30] Gabriela Jacques-Silva, Evangelia Kalyvianaki, Katriel Cohn-Gordon, Adham Meguid, Huy Nguyen, Danny Ben-David, Carl Nayak, Varun Saravagi, George Stasa, Ioannis Papagiannis, et al. 2025. Unified lineage system: Tracking data provenance at scale. In *Companion of the International Conference on Management of Data*. 457–470.

[31] Marco Johns, Lena Baum, and Fabian Prasser. 2025. Tracking provenance in clinical data warehouses for quality management. *International Journal of Medical Informatics* 193 (2025), 105690.

[32] Grigoris Karvounarakis, Zachary G Ives, and Val Tannen. 2010. Querying data provenance. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*. 951–962.

[33] Anastasios Kementsietsidis and Min Wang. 2009. Provenance query evaluation: what's so special about it?. In *Proceedings of the 18th ACM conference on Information and knowledge management*. 681–690.

[34] Chrysanthi Kosyfaki and Nikos Mamoulis. 2022. Provenance in Temporal Interaction Networks. In *38th IEEE International Conference on Data Engineering, ICDE, Kuala Lumpur, Malaysia, May 9-12*. IEEE, 2277–2290.

[35] Chrysanthi Kosyfaki, Nikos Mamoulis, Evaggelia Pitoura, and Panayiotis Tsaparas. 2019. Flow Motifs in Interaction Networks. In *Advances in Database Technology - 22nd International Conference on Extending Database Technology, EDBT, Lisbon, Portugal, March 26-29*. OpenProceedings.org, 241–252.

[36] Chrysanthi Kosyfaki, Nikos Mamoulis, Evaggelia Pitoura, and Panayiotis Tsaparas. 2021. Flow Computation in Temporal Interaction Networks. In *37th IEEE International Conference on Data Engineering, ICDE, Chania, Greece, April 19-22*. IEEE, 660–671.

[37] Rohit Kumar and Toon Calders. 2017. Information propagation in interaction networks. In *Advances in Database Technology, EDBT 2017: Proceedings of the 20th International Conference on Extending Database Technology Venice, Italy, March 2124*. 270–281.

[38] Samuele Langhi, Angela Bonifati, and Riccardo Tommasini. 2025. Evaluating continuous queries with inconsistency annotations. *Proceedings of the VLDB Endowment* 18, 5 (2025), 1321–1334.

[39] Kisung Lee, Raghu Ganti, Mudhakar Srivatsa, and Prasant Mohapatra. 2013. Spatio-temporal provenance: Identifying location information from unstructured text. In *International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*. IEEE, 499–504.

[40] Seokki Lee, Xing Niu, Bertram Ludäscher, and Boris Glavic. 2017. Integrating approximate summarization with provenance capture. In *9th USENIX Workshop on the Theory and Practice of Provenance (TaPP)*.

[41] Brandon Lucia and Luis Ceze. 2015. Data provenance tracking for concurrent programs. In *2015 IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*. IEEE, 146–156.

[42] Paolo Missier, Norman W. Paton, and Khalid Belhajjame. 2010. Fine-grained and efficient lineage querying of collection-based workflow provenance. In *EDBT, 13th International Conference on Extending Database Technology, Lausanne, Switzerland, March 22-26, Proceedings (ACM International Conference Proceeding Series)*. ACM, 299–310.

[43] Haneen Mohammed and Eugene Wu. 2025. Lineage Capture Trade-offs: A Case Study in DuckDB. In *Proceedings of the ProvenanceWeek 2025*. 32–36.

[44] Luc Moreau, Ben Clifford, Juliana Freire, Joe Futrelle, Yolanda Gil, Paul Groth, Natalia Kwasnikowska, Simon Miles, Paolo Missier, Jim Myers, et al. 2011. The open provenance model core specification (v1. 1). *Future generation computer systems* 27, 6 (2011), 743–756.

[45] Tobias Müller and Pascal Engel. 2022. How, Where, and Why Data Provenance Improves Query Debugging: A Visual Demonstration of Fine-Grained Provenance Analysis for SQL. In *38th IEEE International Conference on Data Engineering, ICDE, Kuala Lumpur, Malaysia, May 9-12*. IEEE, 3178–3181.

[46] Xing Niu, Bahareh Sadat Arab, Seokki Lee, Su Feng, Xun Zou, Dieter Gawlick,

Vasudha Krishnaswamy, Zhen Hua Liu, and Boris Glavic. 2017. Debugging transactions and tracking their provenance with reenactment. *arXiv preprint arXiv:1707.09930* (2017).

[47] Xing Niu, Raghav Kapoor, Boris Glavic, Dieter Gawlick, Zhen Hua Liu, and Venkatesh Radhakrishnan. 2017. Provenance-Aware Query Optimization. In *33rd IEEE International Conference on Data Engineering, ICDE, San Diego, CA, USA, April 19-22*. IEEE Computer Society, 473–484.

[48] Dimitris Palyvos-Giannas, Vincenzo Gulisano, and Marina Papatriantafilou. 2018. Genealog: Fine-grained data streaming provenance at the edge. In *Proceedings of the 19th International Middleware Conference*. 227–238.

[49] Dimitris Palyvos-Giannas, Bastian Havers, Marina Papatriantafilou, and Vincenzo Gulisano. 2020. Ananke: A Streaming Framework for Live Forward Provenance. *Proc. VLDB Endow.* 14, 3 (2020), 391–403.

[50] Dimitris Palyvos-Giannas, Katerina Tzompanaki, Marina Papatriantafilou, and Vincenzo Gulisano. 2022. Erebus: Explaining the Outputs of Data Streaming Queries. *Proc. VLDB Endow.* 16, 2 (2022), 230–242.

[51] Bofeng Pan, Natalia Stakhanova, and Suprio Ray. 2023. Data provenance in security and privacy. *Comput. Surveys* 55, 14s (2023), 1–35.

[52] Vicky Papavasileiou, Ken Yocum, and Alin Deutsch. 2019. Ariadne: Online provenance for big graph analytics. In *Proceedings of the 2019 International Conference on Management of Data*. 521–536.

[53] Beatriz Pérez, Julio Rubio, and Carlos Sáenz-Adán. 2018. A systematic review of provenance systems. *Knowledge and Information Systems* 57, 3 (2018), 495–543.

[54] Eugenia Politou, Efthimios Alepis, Maria Virvou, and Constantinos Patsakis. 2022. *Privacy and data protection challenges in the distributed era*. Vol. 26. Springer.

[55] Fotis Psallidas, Ashvin Agrawal, Chandru Sugunan, Khaled Ibrahim, Konstantinos Karanasos, Jesús Camacho-Rodríguez, Avrilia Floratou, Carlo Curino, and Raghu Ramakrishnan. 2023. OneProvenance: Efficient Extraction of Dynamic Coarse-Grained Provenance from Database Query Event Logs. *Proceedings of the VLDB Endowment* 16, 12 (2023), 3662–3675.

[56] Fotis Psallidas and Eugene Wu. 2018. Smoke: Fine-grained Lineage at Interactive Speed. *Proc. VLDB Endow.* 11, 6 (2018), 719–732.

[57] Jakub Reha, Giulio Lovisotto, Michele Russo, Alessio Gravina, and Claas Grohnfeldt. 2023. Anomaly detection in continuous-time temporal provenance graphs. In *Temporal Graph Learning Workshop@ NeurIPS 2023*.

[58] Pingcheng Ruan, Gang Chen, Tien Tuan Anh Dinh, Qian Lin, Beng Chin Ooi, and Meihui Zhang. 2019. Fine-Grained, Secure and Efficient Data Provenance for Blockchain. *Proc. VLDB Endow.* 12, 9 (2019), 975–988.

[59] Aryak Sen, Silviu Maniu, and Pierre Senellart. 2025. ProvSQL: A General System for Keeping Track of the Provenance and Probability of Data. *arXiv preprint arXiv:2504.12058* (2025).

[60] Pierre Senellart. 2019. Provenance in databases: Principles and applications. In *Reasoning Web. Explainable Artificial Intelligence: 15th International Summer School 2019, Bolzano, Italy, September 20–24, 2019, Tutorial Lectures*. Springer, 104–109.

[61] Pierre Senellart, Louis Jachiet, Silviu Maniu, and Yann Ramusat. 2018. ProvSQL: Provenance and probability management in PostgreSQL. *Proceedings of the VLDB Endowment (PVLDB)* 11, 12 (2018), 2034–2037.

[62] Wang Chiew Tan et al. 2007. Provenance in databases: Past, current, and future. *IEEE Data Eng. Bull.* 30, 4 (2007), 3–12.

[63] Io Taxidou. 2018. *Information diffusion and provenance in social media*. Ph.D. Dissertation. Dissertation, Universität Freiburg.

[64] Io Taxidou, Tom De Nies, Ruben Verborgh, Peter M Fischer, Erik Mannens, and Rik Van de Walle. 2015. Modeling information diffusion in social media as provenance with W3C PROV. In *Proceedings of the 24th international conference on world wide web*. 819–824.

[65] Chase Walker and Rickard Ewetz. 2025. Explaining the Reasoning of Large Language Models Using Attribution Graphs. arXiv:2512.15663 [cs.AI] https://arxiv.org/abs/2512.15663

[66] Xiaolan Wang, Alexandra Meliou, and Eugene Wu. 2017. QFix: Diagnosing errors through query histories. In *Proceedings of the ACM International Conference on Management of Data*. 1369–1384.

[67] Michael Whittaker, Cristina Teodoropol, Peter Alvaro, and Joseph M Hellerstein. 2018. Debugging distributed systems with why-across-time provenance. In *Proceedings of the ACM symposium on cloud computing*. 333–346.

[68] Albert Ariel Widiaatmaja, Belkis Djeffal, Ashish Dandekar, and Pierre Senellart. 2025. Demonstration of ProvSQL Update Provenance through Temporal Databases. In *Proceedings of the ProvenanceWeek 2025*. 71–76.

[69] Jennifer Widom. 2005. Trio: A System for Integrated Management of Data, Accuracy, and Lineage. In *Second Biennial Conference on Innovative Data Systems Research, CIDR, Asilomar, CA, USA, January 4-7, Online Proceedings*. www.cidrdb.org, 262–276.

[70] Allison Woodruff and Michael Stonebraker. 1997. Supporting Fine-grained Data Lineage in a Database Visualization Environment. In *Proceedings of the Thirteenth International Conference on Data Engineering, April 7-11, Birmingham, UK*. IEEE Computer Society, 91–102.

[71] Yinjun Wu, Abdussalam Alawini, Daniel Deutch, Tova Milo, and Susan Davidson. 2019. ProvCite: provenance-based data citation. *Proceedings of the VLDB Endowment* 12, 7 (2019), 738–751.

[72] Yang Wu, Ang Chen, and Linh Thi Xuan Phan. 2019. Zeno: Diagnosing performance problems with temporal provenance. In *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19)*. 395–420.

[73] Yang Wu, Mingchen Zhao, Andreas Haeberlen, Wenchao Zhou, and Boon Thau Loo. 2014. Diagnosing missing events in distributed systems with negative provenance. *ACM SIGCOMM Computer Communication Review* 44, 4 (2014), 383–394.

[74] Masaya Yamada, Hiroyuki Kitagawa, Salman Ahmed Shaikh, Toshiyuki Amagasa, and Akiyoshi Matono. 2025. LPStream: Fine-grained Lazy Provenance for Stream Processing. *Proc. ACM Manag. Data* 3, 4 (2025), 254:1–254:25.

[75] Alexander Yao. 2022. Interactive Query Explanations Using Fine Grained Provenance. In *SIGMOD: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17*. ACM, 2536–2538.

[76] Bing Zhang, Vadym Doroshenko, Peter Kairouz, Thomas Steinke, Abhradeep Thakurta, Ziyin Ma, Eidan Cohen, Himani Apte, and Jodi Spacek. 2023. Differentially private stream processing at scale. *arXiv preprint arXiv:2303.18086* (2023).

[77] Yuankai Zhang, Adam O'Neill, Micah Sherr, and Wenchao Zhou. 2017. Privacy-preserving network provenance. *Proceedings of the VLDB Endowment* 10, 11 (2017), 1550–1561.

[78] David Zhao, Pavle Subotić, and Bernhard Scholz. 2020. Debugging large-scale datalog: A scalable provenance evaluation strategy. *ACM Transactions on Programming Languages and Systems (TOPLAS)* 42, 2 (2020), 1–35.

[79] Nan Zheng and Zack Ives. 2020. Compact, Tamper-Resistant Archival of Fine-Grained Provenance. *Proc. VLDB Endow.* 14, 4 (2020), 485–497.

[80] Xiangwei Zheng, Lifeng Zhang, Chunyan Xu, Xuanchi Chen, and Zhen Cui. 2024. An attribution graph-based interpretable method for CNNs. *Neural Networks* 179 (2024), 106597. https://doi.org/10.1016/J.NEUNET.2024.106597

[81] Wenchao Zhou, Ling Ding, Andreas Haeberlen, Zachary Ives, and Boon Thau Loo. 2011. {TAP}: Time-aware Provenance for Distributed Systems. In *3rd USENIX Workshop on the Theory and Practice of Provenance (TaPP 11)*.

[82] Wenchao Zhou, Suyog Mapara, Yiqing Ren, Yang Li, Andreas Haeberlen, Zachary Ives, Boon Thau Loo, and Micah Sherr. 2012. Distributed time-aware provenance. *Proceedings of the VLDB Endowment* 6, 2 (2012), 49–60.

[83] Michael Zipperle, Florian Gottwalt, Elizabeth Chang, and Tharam Dillon. 2022. Provenance-based intrusion detection systems: A survey. *Comput. Surveys* 55, 7 (2022), 1–36.