Reliable Context

RELIABLE CONTEXT / RESEARCH & PRACTICAL GUIDANCE

Query Provenance: Which Source Rows Justify a Derived Value?

> Reliable Context is a kernel of truth: a system of record for organizations that fuse private and public information and must defend every value they act on. One canonical record per real thing, a cited source and an observation date behind every value, unknowns declared rather than left blank, and any past state of the record reproducible.

Published by Reliable Context · 2026-10-27 · query provenance · data provenance · why provenance · where provenance · how provenance · provenance semirings · data lineage vs data provenance · row-level provenance · prov-o · sql witness sets

Original article: https://reliablecontext.com/articles/query-provenance-source-rows-derived-value



In this article

1. Executive Summary
2. Introduction and Background
3. Key Distinctions in Query Provenance
4. The Reference Asset: A Toy Join, Projection and SUM
5. Aggregates: Why Tuple-Only Annotation Does Not Explain the Number
6. Mapping the Example to PROV-O
7. Implementation Considerations and Process Changes
8. Limits: Unsupported Operators, Redactions and Unresolved Source Conflicts
9. Data Analysis and Evidence
10. Implications and Future Directions
11. Frequently Asked Questions (FAQs)
12. Conclusion

Executive Summary

A derived value in a canonical record is defensible only if a reviewer can name the input rows, and ideally the input cells, that produced it. A dataset-level lineage arrow does not meet that bar. Data lineage tools describe how data flows through systems (datahub.com), and the OpenLineage specification models Jobs, Runs and Datasets rather than rows (openlineage.io). Row-level query provenance answers a different question: which source tuples witnessed an output tuple (homepages.inf.ed.ac.uk). This report turns that distinction into a black-box acceptance test that a context engineer can run without access to any vendor internals.

The vocabulary comes from the database literature. Why-provenance identifies the source data that influenced the existence of an output (link.springer.com). Where-provenance identifies the location a value was copied from (homepages.inf.ed.ac.uk). How-provenance describes how the source tuples witness the output tuple (homepages.inf.ed.ac.uk). **Green, Karvounarakis and Tannen (2007)** unified these for positive relational algebra using commutative semirings (web.cs.ucdavis.edu), and, in the same paper, extended the approach toward recursive Datalog (web.cs.ucdavis.edu). For aggregates, **Amsterdamer, Deutch and Tannen (2011)** showed that tuple-only annotation is inapplicable and that individual values inside tuples must carry provenance too (arxiv.org).

The reusable asset is a fully specified toy example: four private order rows joined to four public registry rows, with a DISTINCT projection, a row-level projection and a SUM. The EMEA region has two alternative witnesses, {o1, r1} and {o2, r2}; the EMEA total of **200** needs both (120 + 80). A decoy order worth 999 (**79.98%** of the 1,249 total order amount) has no join partner and contributes nothing, yet a dataset-level arrow would include it. The witness sets were derived by hand, by SQLite, by an independent Python evaluator and by exhaustive enumeration of all **255** non-empty input subsets; all four agree. To reproduce the enumeration, rerun the query on each of the 255 non-empty subsets of the eight input rows and compare each result with the full-input result. The evaluator code itself is not published here.

Standards help with record-keeping but do not compute witnesses. W3C PROV-O, a Recommendation dated **30 April 2013** (www.w3.org), defines Entity, Activity, usage and derivation, but the data model states that PROV does not specify the conditions under which derivations exist (www.w3.org). Tool support is bounded: ProvSQL, the PostgreSQL extension, labels where-provenance an experimental feature (provsql.org) and states that an aggregate value has no source location because it is computed rather than copied (provsql.org). The acceptance test therefore combines removal, perturbation, a three-way evidence check (where, why, how) and a preserved run record, and it names unsupported operators, redactions and unresolved source conflicts as explicit limits.

Introduction and Background

Context engineers who maintain canonical records face a recurring acceptance decision: a field displays a number or label that was calculated from several inputs, some private and some public, and a reviewer asks what justifies it. The usual answer is a citation to a dataset or a pipeline step. That answer is a lineage arrow. It states that table A and table B fed table C. It does not state which rows of A and B the displayed cell depends on, and it cannot distinguish a row that mattered from a row that sat in the same table.

The terminology is contested in practice. One vendor explainer published in June 2026 defines data lineage as tracking how data flows through systems (datahub.com) and data provenance as the historical record of data (datahub.com), and frames the two as differing in purpose, one operational and the other about governance (datahub.com). The glossary of the National Network of Libraries of Medicine notes that data provenance is sometimes called data lineage (www.nnlm.gov). The research literature draws sharper lines. Cui and Widom's lineage definition identified the exact set of base data items that produced a view data item (cs.brown.edu), while Cheney, Chiticariu and Tan (2009) showed that a lineage set is not as precise as one may like (homepages.inf.ed.ac.uk). Query provenance is the family of techniques that resolve this ambiguity at the level of individual input tuples and cells.

This report treats the problem as one of evidence rather than tooling: given a displayed value, a query and a source snapshot, what must be shown for the value to count as justified? The method is black-box. It requires only the ability to run the query on modified copies of the inputs and to inspect the query text.

Disclosure: the publisher, Reliable Context, builds a system of record, a tool that works at the level of individual values. It is complementary to query-level witness analysis, and this report does not claim that a system of record computes SQL witness sets. The test below is how a reader can check any system, including that one, against the input-level standard.

Scope and sources. The formal claims rest on primary papers, the W3C PROV family and tool documentation retrieved in October 2026. Search-volume figures for the topic are advertising estimates and are not used as evidence. All worked numbers were recomputed locally.

Key Distinctions in Query Provenance

Defining the objects: output value, source cell, contributing row, witness set, transformation

Five terms recur in the acceptance test, and each is used here in a fixed sense.

- **Output value:** one cell of a query result, addressed by result row and column, for example the SUM cell of the EMEA row.
- **Source cell:** one cell of one input row, addressed by relation, row identifier and column, for example orders.amount of row o1.
- **Contributing row:** an input row that appears in at least one witness of the output.
- **Witness set:** a subset of the input records sufficient for an output record to appear (homepages.inf.ed.ac.uk). A minimal witness has no removable member. One output can have several alternative witnesses (homepages.inf.ed.ac.uk).
- **Transformation:** the query text and version that maps inputs to the output, because provenance is sensitive to how the query is written. Where-provenance in particular is not invariant under equivalent queries (homepages.inf.ed.ac.uk).

Where-provenance versus why-provenance

Buneman, Khanna and Tan (2001) distinguished two questions. Why-provenance concerns the source data that had some influence on the existence of the data (link.springer.com). Where-provenance concerns where a piece of data is copied from (homepages.inf.ed.ac.uk). The survey by Cheney and colleagues adds the unit of analysis: why-provenance relates source and output tuples, whereas where-provenance relates source and output locations, meaning cells (homepages.inf.ed.ac.uk). In the toy example below, the label EMEA in the output has two candidate copy locations (the region cells of r1 and r2), while the existence of that output row is explained by two alternative witnesses.

Müller and colleagues (2018) generalise where-provenance for SQL: it includes cells copied verbatim but also cells used to compute a value (arxiv.org). A reviewer who asks for "the source cells" of a computed number is therefore asking for more than a copy pointer.

How-provenance and provenance semirings explained

Why-provenance reports which tuples contribute but not how. Green, Karvounarakis and Tannen observed that one needs to know not just which input tuples contribute but also how they contribute (web.cs.ucdavis.edu), and said the resulting annotation could justifiably be called how-provenance (web.cs.ucdavis.edu). Their construction annotates each tuple with an element of a commutative semiring and defines a generalised positive algebra on those annotated relations (web.cs.ucdavis.edu). The rules are compact: for projection and union the annotations of the same tuple are added, and for join they are multiplied (web.cs.ucdavis.edu). The most general annotation is a polynomial with integer coefficients over input tuple identifiers (web.cs.ucdavis.edu), and any other annotation semantics follows from it through a semiring homomorphism (web.cs.ucdavis.edu). The lineage of a tuple is the set of all contributing input tuples (web.cs.ucdavis.edu), which is why lineage is strictly less informative than the polynomial: Cheney and colleagues note that why-

provenance cannot show that one source tuple contributes twice, while the polynomial can (homepages.inf.ed.ac.uk), and that one can derive the why-provenance of an output tuple from the polynomial (homepages.inf.ed.ac.uk).

For the toy example the polynomial of the EMEA row is o1·r1 + o2·r2. Addition marks alternative derivations; multiplication marks inputs that must co-occur. The set of minimal witnesses is read off the monomials: {o1, r1} and {o2, r2}.

Row-level and cell-level granularity

Herschel, Diestelkaemper and Ben Lahmar (2017) place data provenance at the finest level of a type hierarchy, one that can track the processing of individual data items (www.ipvs.uni-stuttgart.de), and describe a spectrum whose two ends are coarse-grained and fine-grained provenance (www.ipvs.uni-stuttgart.de). Row granularity is what the majority of existing work provides (arxiv.org). The acceptance test uses rows first and cells second.

Data lineage versus data provenance, restated for acceptance

The practical difference is the unit of evidence. Lineage systems document datasets, jobs and, increasingly, columns. **OpenLineage** describes itself as an open framework for data lineage collection and analysis (openlineage.io) and defines a Dataset as an abstract representation of data (openlineage.io). Its column-level facet gives fine grained information on datasets' dependencies (openlineage.io), but it operates on columns, not on the rows that held the values on a particular run. A catalog-level fact such as "column region depends on column registry.region" is true of every output row, including rows with no support. The remainder of this report shows why that is insufficient for a single displayed cell.

The Reference Asset: A Toy Join, Projection and SUM

Inputs and queries

The relations below are fully specified. Each row carries an identifier (o1 to o4, r1 to r4) so that witness sets can be written down. The orders relation stands in for a private source and the registry relation for a public one; the logic does not depend on that split.

```
CREATE TABLE orders  (rid TEXT, order_id INT, org_id TEXT, amount INT);
INSERT INTO orders  VALUES ('o1',1,'A',120), ('o2',2,'B',80),
                           ('o3',3,'C',50),  ('o4',4,'Z',999);
CREATE TABLE registry (rid TEXT, org_id TEXT, name TEXT, region TEXT);
INSERT INTO registry VALUES ('r1','A','Acme','EMEA'), ('r2','B','Bolt','EMEA'),
                            ('r3','C','Cirrus','APAC'), ('r4','D','Delta','EMEA');

Q1: SELECT DISTINCT r.region
    FROM orders o JOIN registry r ON o.org_id = r.org_id;
Q2: SELECT r.region, SUM(o.amount)
    FROM orders o JOIN registry r ON o.org_id = r.org_id GROUP BY r.region;
Q3: SELECT r.name, r.region
    FROM orders o JOIN registry r ON o.org_id = r.org_id;
```

The join keeps three order-registry pairs: (o1, r1), (o2, r2) and (o3, r3). Order o4 carries org_id Z, which has no registry row, and registry row r4 (org D) has no order. Inner join semantics mean that the joined table has a row for each pair satisfying the condition (www.postgresql.org), so o4 and r4 never enter the join result. The outputs are Q1 = {APAC, EMEA}; Q2 = {(APAC, 50), (EMEA, 200)}; Q3 = {(Acme, EMEA), (Bolt, EMEA), (Cirrus, APAC)}.

The witness matrix

Table 1 below lists, for each output, the minimal witnesses, the how-polynomial, and the source cells at three roles: value cells (their numbers enter the output), key cells (join and group conditions that were inspected) and label cells (the origin of copied text). The sets were derived by hand from the join pairs and then checked by exhaustive enumeration of all 255 non-empty subsets of the eight input rows.

Output	Minimal witnesses (why)	How-polynomial	Source cells
Q1 (EMEA)	{o1, r1}, {o2, r2}	o1·r1 + o2·r2	Label: registry.region of r1 and r2 (two candidate copy locations). Key: org_id of o1, r1, o2, r2.
Q1 (APAC)	{o3, r3}	o3·r3	Label: registry.region of r3. Key: org_id of o3, r3.
Q2 (EMEA, 200)	Both rows of each witness are needed together: {o1, r1, o2, r2}	(o1·r1)⊗120 + (o2·r2)⊗80	Value: orders.amount of o1 (120) and o2 (80). Key: org_id of o1, r1, o2, r2. Group label: registry.region of r1, r2.
Q2 (APAC, 50)	{o3, r3}	(o3·r3)⊗50	Value: orders.amount of o3 (50). Key: org_id of o3, r3. Group label: registry.region of r3.
Q3 (Acme, EMEA)	{o1, r1}	o1·r1	Label: registry.name and registry.region of r1. Key: org_id of o1, r1.
Q3 (Bolt, EMEA)	{o2, r2}	o2·r2	Label: registry.name and registry.region of r2. Key: org_id of o2, r2.
Q3 (Cirrus, APAC)	{o3, r3}	o3·r3	Label: registry.name and registry.region of r3. Key: org_id of o3, r3.

Three features stand out. First, the DISTINCT projection in Q1 is where two witnesses compete: either {o1, r1} or {o2, r2} makes EMEA appear. Second, in Q2 the same pairs are not alternatives for the number: the SUM cell depends on both together. Third, the cells that matter inside a contributing row are a minority. The Acme row r1 contributes its org_id and region to the SUM, but its name cell is irrelevant to Q2 and relevant to Q3. Row-level witnesses therefore over-approximate cell-level ones, which is the reason the acceptance test treats cells as a second pass.

The perturbation results

Table 2 below records what happens when inputs are removed or changed. Each row was computed in SQLite and recomputed with a separate Python evaluator that implements the join and grouping directly; the two agreed on every row.

Perturbation of the inputs	Q1 result	Q2 result	Reading
None (baseline)	APAC, EMEA	APAC 50, EMEA 200	Reference.
Delete o1	APAC, EMEA	APAC 50, EMEA 80	EMEA survives in Q1 through {o2, r2}. The SUM drops by 120.
Delete o2	APAC, EMEA	APAC 50, EMEA 120	The SUM drops by 80.
Delete r1	APAC, EMEA	APAC 50, EMEA 80	The registry row is as necessary to the SUM as the order row.
Delete o1 and o2 (one member of each witness)	APAC	APAC 50	EMEA disappears: the deleted set hits every witness.
Delete o1 and r2	APAC	APAC 50	Same.
Delete o4 or r4	APAC, EMEA	APAC 50, EMEA 200	No change: neither row is in any witness.
Set o1.amount from 120 to 130	APAC, EMEA	APAC 50, EMEA 210	The value cell passes through with weight 1.
Set o4.amount from 999 to 1	APAC, EMEA	APAC 50, EMEA 200	No change: an unwitnessed cell cannot move the output.
Set r1.region from EMEA to APAC	APAC, EMEA	APAC 170, EMEA 80	The group label cell controls which group receives 120.
Set o1.org_id from A to Z	APAC, EMEA	APAC 50, EMEA 80	The join-key cell is inspected: breaking it removes the pair.
Set r1.name from Acme to Acme2	APAC, EMEA	APAC 50, EMEA 200	Q3 changes to (Acme2, EMEA); Q2 does not.

Two lessons follow. A single-row removal test cannot certify a contributing row when an alternative witness exists; it must be applied to sets that hit every minimal witness. And a perturbation test certifies a cell only for the change tested, which is why the protocol pairs it with a structural witness.

Aggregates: Why Tuple-Only Annotation Does Not Explain the Number

The semiring polynomial of the EMEA row in Q2 is the same as that of EMEA in Q1: o1·r1 + o2·r2. Read alone, it says that EMEA exists if either pair exists. That is true of the group, but it is silent on the number. Deleting o1 changes the total from 200 to 80, and the polynomial gives no way to predict that.

Amsterdamer, Deutch and Tannen (2011) address this gap. They state that aggregate queries pose novel challenges, rendering the semiring approach inapplicable (arxiv.org), and propose annotating not just tuples but also the individual values within tuples (arxiv.org). Their device is the tensor product (arxiv.org). For the toy SUM, the EMEA cell is the formal expression (o1·r1)⊗120 + (o2·r2)⊗80, which is the form shown in *Table 1*.

Setting the token o1 to zero removes the first term and leaves 80, which matches the deletion row in *Table 2*. The authors relate this algebraic treatment of deletions to the counting algorithm for view maintenance (arxiv.org).

Two limits from the same paper bear on acceptance:

- **Size.** The construction keeps the representation size polynomial in the size of the input database (arxiv.org), and the authors argue that tuple-level annotation alone cannot represent aggregation with a feasible amount of annotation.
- **Negation.** Relational difference is encoded through nested aggregation, and the authors caution that the resulting semantics is probably not the last word on difference of annotated relations (arxiv.org).

PostgreSQL's documentation adds a practical detail for any audit of a SUM: it computes the sum of the non-null input values (www.postgresql.org), and the sum of no rows returns null, not zero (www.postgresql.org). A NULL amount therefore contributes no numerical weight yet can still be a witnessed row, and a group with no contributing rows yields NULL rather than 0. A reviewer should confirm which of those two cases a displayed zero or blank represents.

How tools handle this varies. Perm, an early SQL rewriting system, defines the provenance of a query computing the sum of an attribute for all tuples of an input relation as every input tuple (permdbms.sourceforge.net), and rewrites aggregation by joining its results with the rewritten input (permdbms.sourceforge.net). ProvSQL, by contrast, documents that aggregation is treated as a scalar multiplication of provenance values (provsql.org), in line with the semimodule construction.

Mapping the Example to PROV-O

W3C PROV-O is a Recommendation dated 30 April 2013 (www.w3.org). It is a vocabulary for recording the history of a record, and an OWL2 ontology that maps the PROV data model to RDF (www.w3.org). For the toy example, four of its terms carry the weight.

- **Entity:** a physical, digital, conceptual, or other kind of thing (www.w3.org). Each input row, each result cell and the source snapshot can be entities.
- **Activity:** something that occurs over a period of time (www.w3.org). The execution of Q2 against a snapshot is an activity.
- **used:** usage is the beginning of utilizing an entity by an activity (www.w3.org). The query run used the snapshot and the rows read.
- **wasDerivedFrom:** a derivation is a transformation of an entity into another (www.w3.org). The EMEA total derives from o1 and o2, with the join keys and group label cells recorded as additional inputs where the reviewer needs them.

A responsible agent is also recordable. An agent is something that bears some form of responsibility (www.w3.org), and an activity association is an assignment of responsibility to an agent for an activity (www.w3.org). Generation, the completion of production of a new entity by an activity (www.w3.org), links the result to the run. A minimal Turtle sketch of the Q2 EMEA cell follows.

```
ex:emea_total  a prov:Entity ;
    prov:wasGeneratedBy ex:run_q2_2026_10_20 ;
    prov:wasDerivedFrom ex:o1, ex:o2, ex:r1, ex:r2 .
ex:run_q2_2026_10_20  a prov:Activity ;
    prov:used ex:orders_snapshot, ex:registry_snapshot ;
    prov:wasAssociatedWith ex:analyst_agent .
```

What PROV-O does not do is equally important. The data model defines provenance as information about entities, activities and people involved in producing a piece of data (www.w3.org), that can be used to form assessments about its quality, reliability or trustworthiness (www.w3.org). It records; it does not rule on truth. It also does not compute the derivation edges: PROV does not attempt to specify the conditions under which derivations exist (www.w3.org), so a derivation is considered to have been determined by unspecified means (www.w3.org). Whether ex:o4 is absent from the derivation list is a fact about the query, to be established by a witness computation or the perturbation tests, not by the vocabulary. A cloud-database article mentions PROV-O only in a bullet about a separate metadata dataset describing such things as where the data came from, who created it, or when it was created (aws.amazon.com), which is the coarse end of the spectrum.

Implementation Considerations and Process Changes

The acceptance protocol

The protocol below accepts or rejects one displayed derived value. Its output is the audit trail for a derived data value: a record that another reviewer can replay. It needs the query text, a frozen copy of the inputs and the ability to run the query again.

- **Step 1, freeze.** Preserve the query text and version, the source snapshot identifiers, the execution time and the responsible agent. These four items are what PROV-O's activity, used and association terms exist to hold.
- **Step 2, enumerate candidates.** List the input rows that the join and filter conditions admit. For the toy query this is three of four orders and three of four registry rows.
- **Step 3, state witnesses.** For each output, write the minimal witness sets. Two outputs with the same set of rows but different roles, such as Q1 and Q2 EMEA, must be reported separately, since the number needs a conjunction of witnesses.
- **Step 4, removal test.** Delete a set of rows that hits every minimal witness and confirm that the output disappears or changes as predicted. Delete a row outside the witness sets and confirm that nothing changes.
- **Step 5, perturbation test.** Change one input cell and confirm the output moves by the predicted amount. For SUM, record each input's numerical contribution and confirm that the contributions add to the total.
- **Step 6, evidence kind.** Record which of where, why and how evidence supports the cell: a copy location for labels, a witness set for existence, a formal expression for computed numbers.

- **Step 7, source quotation.** For each contributing source cell, attach the verbatim source text and observation date. Query provenance stops at the input row; whether that row is correct is a separate question.

Decision table: audit question to minimum evidence

Table 3 below maps the question a reviewer asks to the least evidence that answers it. The mapping is the author's synthesis of the definitions above, not a standard.

Audit question	Minimum evidence	Why a weaker artifact fails
Which pipelines or datasets touched this value?	Dataset lineage	Sufficient only when the reviewer accepts every row of every upstream dataset as potentially relevant.
Which input rows does the displayed value depend on?	Input rows: the minimal witness sets from the query run	A dataset arrow cannot exclude row o4, which carries 999 of the 1,249 total order amount.
Which input cells does it depend on?	Source cells at three roles: value, key and label	Row-level evidence over-approximates: r1.name is in a contributing row but irrelevant to the SUM.
How were the inputs combined?	Transformation: query text, version and a how-polynomial or tensor expression	A witness set cannot show that two pairs must co-occur for the SUM.
Is the input itself right?	Source quotation with observation date and the source's own identifier	Provenance proves dependence, not accuracy.
Can the value be reproduced later?	Query version, snapshot, time, agent	Without a frozen snapshot, a re-run may diverge from the displayed value.

Dataset lineage is cheapest and weakest; only the source quotation speaks to correctness.

Limits: Unsupported Operators, Redactions and Unresolved Source Conflicts

What the formal results do and do not cover

The semiring result applies to positive relational algebra, extended to recursive Datalog queries (web.cs.ucdavis.edu), but Green and colleagues left negation open, noting that it seems to require axiomatizing an additional operation (web.cs.ucdavis.edu). The toy example uses only inner join, projection, DISTINCT, GROUP BY and SUM, so it sits partly inside that fragment and partly in the aggregate extension. Constructs such as EXCEPT, NOT EXISTS, outer joins, window functions, correlated subqueries and user-defined functions are outside the positive-algebra result, and a claim that all SQL shares the result would be unsupported. The Perm authors list queries with multiple set-difference operations and subqueries in negated or disjunctive expressions as an open problem (permdbms.sourceforge.net).

Tool documentation as of October 2026

ProvSQL describes itself as a PostgreSQL extension that transparently tracks the provenance of query results (provsql.org) and represents provenance as a circuit of gates (provsql.org). PGXN lists release **1.13.0**, dated **17 October 2026** (pgxn.org), and its release notes call it a hardening and performance release (github.com). Where-provenance in ProvSQL tracks, for each value in a query result, the specific cell (provsql.org), is disabled by default (provsql.org) and is labelled an experimental feature (provsql.org). That label is the documented basis for the statement that not every SQL construct is supported under where-provenance. The same page says an aggregate value is computed rather than copied, so it has no location (provsql.org). Outside where-provenance mode, the evaluation function sr_why returns the set of witnesses that support the result (provsql.org). A HAVING condition is not exploded because it is already the provenance of the group (provsql.org). The documentation also states that EXCEPT ALL and INTERSECT ALL over tracked relations are refused (provsql.org). A 2018 demonstration paper cites the earlier result that where-provenance cannot be captured by semiring provenance (vldb.org), which is why tools expose both modes rather than one.

The Tübingen group's demonstration targets a feature-rich SQL dialect with correlated subqueries and a focus on bag semantics (db.cs.uni-tuebingen.de). A reviewer should ask which contract a tool implements, since reporting the particular set of input cells used differs from the minimal-witness definition in *Table 1*.

Redactions and unresolved conflicts

Two non-algebraic limits remain.

- **Redactions:** if a contributing source cell is withheld from the reviewer, the witness can be named but the source quotation cannot be shown, and the evidence must be recorded as blocked with a reason rather than silently omitted.
- **Unresolved source conflicts:** if two registry rows disagreed about org A's region, the query would join both and Q2 would produce a number that mixes two claims.

Truth-discovery research notes that information about one object from different sources may conflict due to errors and missing records (arxiv.org). Query provenance names which rows were used; it does not adjudicate between them.

When a catalog or lineage product is the right tool

Table 4 below states the fit between needs and tool classes. It is a description of categories and documented behavior, not a ranking.

Need	Tool class	Documented behavior or fit
Estate-wide map of datasets, jobs and columns	Lineage and catalog products	OpenLineage defines metadata for running jobs and their corresponding events (openlineage.io). DataHub's guide describes column-level lineage that tracks changes and movements for each specific data column (docs.datahub.com). Google documents that lineage records metadata about data movement but doesn't capture the data itself (docs.cloud.google.com).

Need	Tool class	Documented behavior or fit
Which input rows and cells justify one SQL result	Query-provenance engines or the black-box protocol above	ProvSQL for PostgreSQL; research systems such as Perm and the Tübingen tools. Coverage is bounded as in the previous subsections.
A portable record of the run (snapshot, time, agent)	PROV-O or an equivalent run record	Records entities, activities and derivations; does not compute them (www.w3.org).
One canonical record per real thing, with a verbatim cited source and observation date behind each value and its chain of custody exportable	A system of record (the publisher's product category; see the disclosure above)	Relevant to the source-quotation row of *Table 3* only to the extent a given system implements it. It is deliberately not a data catalog. Whether a given derived field has input-level justification is checked with the protocol above.

A lineage or catalog product is the right choice when the question is "what depends on this table", for impact analysis or estate inventory. Microsoft Purview's classic catalog documentation, for example, says its lineage granularity can vary based on the data systems supported (learn.microsoft.com); dbt notes that column-level lineage does not reflect other usage like joins and filters (docs.getdbt.com) and relies on SQL parsing (docs.getdbt.com); Google documents that column-level lineage is not collected if a job creates more than 1,500 column-level links (docs.cloud.google.com). These scope statements explain why a column arrow cannot answer which orders produced a particular total.

Data Analysis and Evidence

Worked numbers from the toy example

The toy example yields the following quantities, each recomputed by two independent evaluators and an exhaustive subset search.

- **Rows involved.** The eight input rows split into six that appear in some witness (o1, o2, o3, r1, r2, r3) and two that appear in none (o4, r4). The EMEA SUM cell depends on **4 of 8 rows (50%)**.
- **SUM decomposition.** EMEA **200** = 120 (o1, 60%) + 80 (o2, 40%). APAC 50 = 50 (o3, 100%). The output total of 250 is 20.0% of the 1,249 total order amount.
- **Decoy weight.** Order o4 holds 999 of 1,249, or **79.98%** of the order amount, and contributes 0 to any output.
- **Deletion sensitivity.** Deleting any single row that belongs to an EMEA witness changes the EMEA SUM (by minus 120 or minus 80). Deleting any one non-witness row changes nothing. For Q1 no single deletion changes EMEA, but two deletions (one per witness) remove it.

Published measurements on cost and scope

The burden of fine-grained capture is quantified in the literature, with methods that differ.

- **Perm (2009):** overhead for provenance-computing TPC-H queries in a range of factor 3 to 30 (permdbms.sourceforge.net) on test databases of 10MB, 100MB and 1GB (permdbms.sourceforge.net); for one query the number of result tuples increased by a factor of approximately **15000** (permdbms.sourceforge.net).
- **Smoke (2018):** at most **22%** overhead across four TPC-H queries at scale factor 1 (arxiv.org).
- **ProvSQL (2025):** scale factors 1 to 10, with databases of size 1 to 10 GB (arxiv.org), and a high overhead on two queries that nevertheless remains under a factor of **4** (arxiv.org).

The systems differ in definition and workload, so the figures are not a ranking; capture is feasible but not free.

Computational complexity bounds what any checker can promise. Deletion propagation is in general NP-complete for conjunctive queries containing projections and joins (arxiv.org), so no checker can promise a minimal repair for arbitrary queries.

Conflicts and missing provenance at scale

The scale of source disagreement motivates the "unresolved conflict" limit.

- **Stock and flight sources:** for **70%** of data items more than one value is provided (arxiv.org), and only **70%** of correct values are provided by the majority of the sources (arxiv.org).
- **Web-scale extraction:** extraction errors are far more prevalent than source errors (arxiv.org), which implies a second provenance layer, the extractor, that a row-level witness over the extracted table does not itself record.
- **Wikidata referencing:** Curotto and Hogan's August 2020 snapshot found 771 million statements (**68.56%**) referenced to external sources, 68 million (6.02%) referenced to Wikipedia, and **286 million** (25.42%) without reference (www.aidanhogan.com).

No cross-industry survey of the share of derived values lacking input-level provenance was found, and none is asserted here.

Implications and Future Directions

For context engineers, the main implication is a change of evidence standard. A derived field in a canonical record should not be accepted on the strength of a pipeline diagram. It should carry, or be able to regenerate on demand, the witness rows for each displayed cell, the query version and snapshot, and the source quotation behind each witnessed input. The cost literature suggests, though no study tests it directly, that generating witnesses on demand for a reviewed subset may be practical even where capture on every query is not, since overheads range from under a factor of 4 even on ProvSQL's two highest-overhead evaluated queries (arxiv.org) to a factor of **3 to 30** for Perm's (permdbms.sourceforge.net).

For tool buyers, the documented limits are the useful part of the documentation. Where-provenance in ProvSQL is described as experimental (provsql.org), and column-level lineage in the catalog products cited above is scoped by parsing and by supported systems (docs.getdbt.com) (learn.microsoft.com). Procurement should ask for the output of the toy example, in particular the Q2 EMEA cell and the exclusion of o4, before accepting a claim of fine-grained provenance.

For researchers, three gaps are visible. First, no public benchmark pairs witness-set correctness with verbatim source quotations; the performance papers measure capture overhead rather than reviewer-facing evidence. Second, the aggregate and negation cases remain the least settled. Third, extraction pipelines add a layer that SQL provenance does not see. Combining row witnesses, extractor identity and source quotations in a single exportable chain is an open design problem.

Frequently Asked Questions (FAQs)

What is query provenance, in one sentence?

Query provenance is the record of which input data a query result depends on, at the level of tuples or cells, in the senses of why, where and how (db.cs.uni-tuebingen.de).

How do you trace a derived value to its source rows?

Write the query, freeze the snapshot, enumerate the join pairs, state the minimal witness sets, then confirm them by deleting a set that hits every witness and by changing one cell at a time, as in *Table 2*. The how-polynomial or tensor expression records the combination rule.

Why-provenance versus where-provenance: which one is needed?

Existence questions ("why is this row here?") need why-provenance, a witness set. Copy questions ("which cell did this label come from?") need where-provenance. A computed number needs both plus the combination rule, because an aggregate value is computed rather than copied (provsql.org).

What are provenance semirings, in plain terms?

They are annotations on tuples that combine by addition for alternatives and multiplication for joint use (web.cs.ucdavis.edu).

Is data provenance the same as data lineage?

Not in the sense used here. Provenance names the origin and history of data (csrc.nist.gov), lineage tools document flows between datasets and columns, and one vendor describes provenance as the trust layer underneath lineage (datahub.com). For acceptance, the question is whether the artifact names rows.

Conclusion

A dataset-level lineage arrow documents that data moved; it does not show that a displayed cell is supported. The input-level standard is the witness: the set of rows, and where needed cells, that jointly produce the value, with a rule for combining them. The toy example makes the difference concrete. Two alternative witnesses explain why EMEA appears, both pairs are needed for the total of 200, and a decoy order holding 79.98% of the order amount contributes nothing. A reviewer who sees only the lineage arrow cannot tell these cases apart.

The formal basis is well established for positive relational queries and, with extra structure, for aggregates. It is not established for every SQL construct, and tool documentation reflects that, including the experimental status and stated gaps of current implementations. Standards such as PROV-O supply the vocabulary to preserve the query version, snapshot, time and agent, but they do not compute the derivations. The practical result is a short protocol: freeze, enumerate, state witnesses, remove, perturb, record the evidence kind and attach the source quotation.

Two things stay outside the method by design. Provenance shows dependence, not accuracy, so the verbatim source and observation date behind each witnessed input remain necessary. And where sources disagree, the method exposes the conflict instead of resolving it. Teams can adopt the toy example as a regression fixture for any system that claims fine-grained provenance.

Source links

Links cited in this article, in order of first appearance. Full addresses are provided for readers using extracted text.

1. https://datahub.com/blog/data-lineage-vs-data-provenance
2. https://openlineage.io/docs/spec/object-model
3. https://homepages.inf.ed.ac.uk/jcheney/publications/provdbsurvey.pdf
4. https://link.springer.com/chapter/10.1007/3-540-44503-X_20
5. https://web.cs.ucdavis.edu/~green/papers/pods07.pdf
6. https://arxiv.org/abs/1101.1110
7. https://www.w3.org/TR/prov-o
8. https://www.w3.org/TR/prov-dm
9. https://provsql.org/docs/user/where-provenance.html
10. https://www.nnlm.gov/guides/data-glossary/data-provenance
11. https://cs.brown.edu/courses/cs227/archives/2008/Papers/Lineage/trace.pdf
12. https://arxiv.org/pdf/1805.11517
13. https://www.ipvs.uni-stuttgart.de/departments/de/resources/papers/provenance_survey_online.pdf
14. https://openlineage.io/docs
15. https://openlineage.io/docs/spec/facets/dataset-facets/column_lineage_facet
16. https://www.postgresql.org/docs/current/queries-table-expressions.html
17. https://arxiv.org/pdf/1101.1110
18. https://www.postgresql.org/docs/current/functions-aggregate.html
19. https://permdbms.sourceforge.net/pdfs/GA09.pdf
20. https://provsql.org/docs/user/aggregation.html
21. https://www.w3.org/TR/prov-overview
22. https://aws.amazon.com/blogs/database/benefitting-from-sparql-1-1-federated-queries-with-amazon-neptune
23. https://provsql.org/docs/user/introduction.html
24. https://pgxn.org/dist/provsql

25. https://github.com/PierreSenellart/provsql/releases
26. https://provsql.org/docs/user/semirings.html
27. https://provsql.org/docs/user/querying.html
28. https://vldb.org/pvldb/vol11/p2034-senellart.pdf
29. https://db.cs.uni-tuebingen.de/publications/2022/how-where-and-why-data-provenance-improves-query-debugging--a-visual-demonstration-of-fine-grained-provenance-analysis-for-sql
30. https://arxiv.org/pdf/1505.02463
31. https://docs.datahub.com/docs/features/feature-guides/lineage
32. https://docs.cloud.google.com/dataplex/docs/about-data-lineage
33. https://learn.microsoft.com/en-us/purview/concept-data-lineage
34. https://docs.getdbt.com/docs/explore/column-level-lineage
35. https://arxiv.org/pdf/1801.07237
36. https://arxiv.org/pdf/2504.12058
37. https://arxiv.org/pdf/1507.00674
38. https://arxiv.org/pdf/1503.00303
39. https://arxiv.org/pdf/1502.03519
40. https://www.aidanhogan.com/docs/wikidata-references.pdf
41. https://csrc.nist.gov/glossary/term/provenance

THE PUBLISHER

About Reliable Context

Reliable Context is a system of record for organizations that fuse private and public information and must defend every value they act on.

The guarantees

One canonical record per real thing: duplicates resolve by rule, ambiguity is shown rather than hidden, and related organizations relate without ever being merged. A verbatim, cited source and an observation date sit behind every value. Unknowns are declared as not found, blocked or not applicable with a reason, never left as an empty cell. Any past state of the record is reproducible, together with what changed since.

Where it runs

It runs inside the customer's perimeter, one database per organization. Permitted sources, model routing, reader clearance and retention are set as the customer's own policies, and every value is exportable with its chain of custody for an auditor or regulator.

What it is not

It is not a vector database or a RAG framework, not a data catalog, MDM or data-quality suite, not a CRM or data-enrichment vendor, and not a chatbot or AI assistant. Where a reader needs one of those, the honest answer is to say so and point them there.

Visit

See Reliable Context.

Reliable Context is in production use in two sectors; more than six thousand of its identity decisions were made by rule and none has been reversed. Articles published here explain general practice, cite their primary and academic sources, and name the jurisdiction of any regulatory rule they describe. They are not legal, regulatory or compliance advice for a specific organization, and requirements change.

Website: https://reliablecontext.com

This article is educational. Verify consequential medical, legal, financial, regulatory and technical decisions with appropriate professionals and the cited primary sources. Company services are described separately from the article's research findings.