Multi-Level Aggregation and Summarisation of Data Provenance

Paulo Pintor[1], Rogério L. Costa[2], José Moreira[1]
University of Aveiro – IEETA (1), Polytechnic of Leiria - CIIC (2)
Contact: paulopintor@ua.pt

Abstract

Data Provenance focuses on the annotation of database query results with information such as the data sources, why a row participates in the resultset, and how it was obtained. The size of such annotations can significantly grow as queried data size increases. This work presents a method for summarising provenance annotations in database queries, with a focus on aggregations.

Motivation / Objectives

- A single result may have a provenance polynomial like: $t_1 + (t_2 . (t_3 + t_4)) + ((t_5 . t_6). (t_7 + t_8 + t_9) + ... + (t_{n-2} . (t_{n-1} + t_n))$ - N terms, nested structure and many variable occurrences

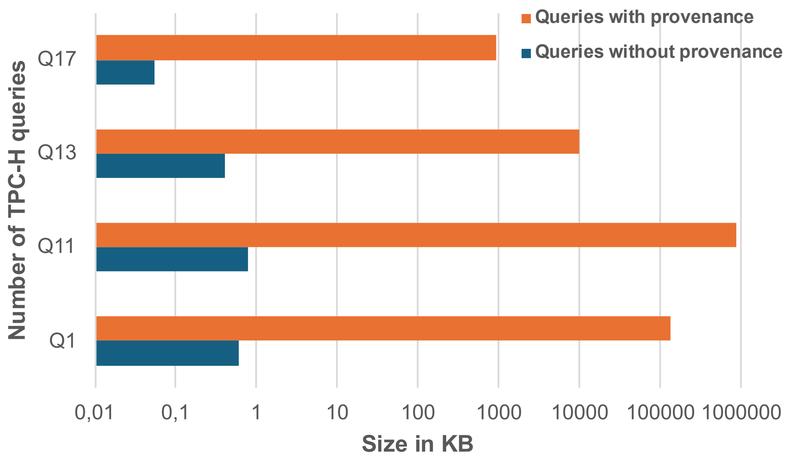

Figure 1: Some TPC-H queries and their size in KB with and without Provenance

- This complexity leads to:
  - High computational overhead;
  - Difficult interpretation for users;
- **This work proposes a multi-level summarisation method focused on aggregations, aimed at:**
  - **Reducing provenance size**
  - **Preserving the structure of provenance polynomials**
  - **Supporting varying levels of detail**

Method

- Automatic query rewriting to generate multi-level summarisation with varying levels of detail;
- Identify fixed attributes (columns on the original query) and free attributes (columns not present in the query from all the tables involved);
- Chose the best combination between fixed and free attributes, to each level until get the original result;
- Can also handle subqueries.
- Provenance annotations follow the semiring formalism for SPJU queries [2], extended to aggregations using the approach in [1];

A simple example:

- Table Orders (Key, Priority, Clerk, Cust, Comment, TotalPrice);
- Original Query: $\gamma_{\text{priority; count(key) -> count}}$ (Orders)

Priority	Count	Provenance Polynomial
p1	6	$\delta(t1 + t2 + t3 + t10 + t11 + t12)$
p2	14	$\delta(t4 + t5 + t6 + t7 + t8 + t9 + t13 + t14 + t15 + t16 + t17 + t18 + t19 + t20)$

Table 1: An hypothetical result of Original Query

- From the free attributes we selected two columns *Cust* and *Clerk;*
- Query Level 1: $\gamma_{\text{priority, cust, clerk; count(key) -> count}}$ (Orders);

Priority	Cust	Clerk	Count	Provenance Polynomial	Var
p1	c1	k1	3	$\delta(t1 + t2 + t3)$	v1
p2	c1	k2	6	$\delta(t4 + t5 + t6 + t7 + t8 + t9)$	v2
p1	c2	k1	2	$\delta(t10 + t11)$	v3
p1	c3	k1	1	$\delta(t12)$	v4
p2	c2	k3	3	$\delta(t14 + t13 + t15)$	v5
p2	c3	k2	4	$\delta(t16 + t17 + t18 + t19)$	v6
p2	c3	k2	1	$\delta(t20)$	v7

Table 2: Result of Query Level 1 (nomenclature in [1] simplified)

- Column ***Var*** represents the provenance polynomials identifier;
- The identifiers enable retrieval at varying levels of detail;
- For the next level we selected *Clerk;*
- Query Level 2: $\gamma_{\text{priority, clerk; sum(count) -> count}}$ (Q1);
- For the next levels we need to use the operator "sum" to keep the original query results.

Priority	Clerk	Count	Provenance Polynomial	Var
p1	k1	6	δ(v1 +v2 +v3)	vc1
p2	k2	11	δ(v2 + v6 +v7)	vc2
p2	k3	3	δ(v5)	vc3

Table 3: Result of Query Level 2

- Query Level 3: $\gamma_{\text{priority, sum(count) -> count}}$ (Q2);

Priority	Count	Provenance Polynomial
p1	6	δ(vc1)
p2	14	δ(vc2 + vc3)

Table 4: Result of Query Level 3

An example with TPC-H (query 10 adapted):

- $\pi_{c\_cols,\ n\_name,\ revenue, o\_custkey,\ o\_orderkey}$ ($\sigma_{\text{condition}}$ (Customer x Orders x Lineitem x Nation))
- where c_cols are the columns from Customer, $revenue$ = $l\_extendedprice$ x (1−$l\_discount$), and $condition$ denotes the join and selection conditions in the query

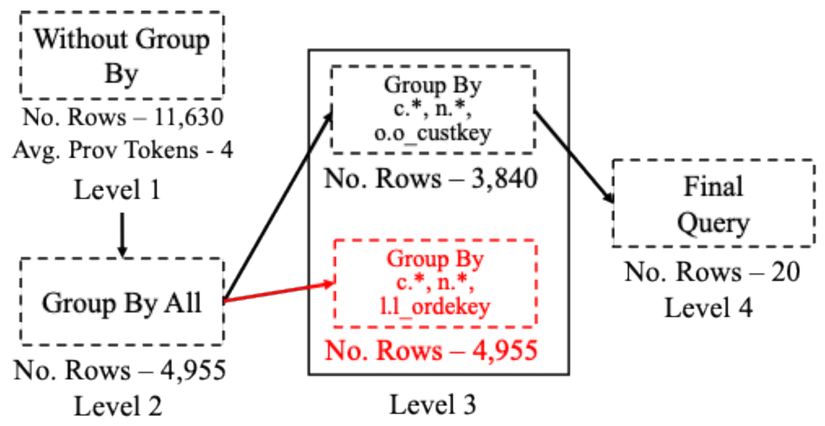

Figure 1: Summarisation path for Q10

- We used ProvSQL [3] as the framework to test our method;
- Using our method with 4 levels, the size of the provenance polynomials in the TPC-H example was reduced by approximately 40%.

Highlights and next steps

- The method provides versatile summarisation of provenance polynomials, allowing aggregation to be tailored through any combination of free variables.
- Runs during query execution without post-processing;
- The method is agnostic to the framework used to generate provenance polynomials.
- Shows good results in reducing provenance polynomials

Next steps include:

- Extending support to nested queries and limited free-attribute cases;
- Developing visualisation and query tagging to enhance provenance understanding and filtering.

References

[1] Yael Amsterdamer, Daniel Deutch, and Val Tannen. 2011. Provenance for Aggregate Queries. Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (2011), 153–164. doi:10.1145/1989284.1989302

[2] 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 (PODS '07). ACM, 31–40.

[3] Pierre Senellart, Louis Jachiet, Silviu Maniu, and Yann Ramusat. 2018. ProvSQL: provenance and probability management in postgreSQL. Proc. VLDB Endow. 11,12 (Aug. 2018), 2034–2037. doi:10.14778/3229863.3236253

ACKNOWLEDGEMENTS

This work is partially funded by National Funds through FCT, under the Scientific Employment Stimulus: Institutional Call - CEECIN - ST/00051/2018, and projects UIDB/04524/2020 and UID/00127.