SQL Query Visualization Using Data Provenance

Robert Ward, Charlie Summers, Haneen Mohammed, Professor Eugene Wu

Context: Database Queries

Datasets and SQL Query Input

SQL is a language that uses **Queries** to interact with and manipulate databases

- Declarative
- high-level

```
SELECT p.plant, sum(f.sweet + f.sour + f.bitter) as
total_flavor
FROM flavor_profile AS f
JOIN plant_info AS p ON f.name = p.name
GROUP BY p.plant
```

Query Execution

A Query engine converts it to a **Physical Plan** which is executed on the data.

- Restructured
- Engine-specific
- Lower-level

```
HASH_GROUP_BY_4 | #0\sum(#1)
      PROJECTION_3 | plant \+(+(sweet, sour), bitter)
            HASH_JOIN_2 | INNER\name=name
                  PANDAS_SCAN_0 | name\sweet\sour\bitter
                  PANDAS_SCAN_1 | name\plant
```

It's difficult to assess a query's execution process

Using Provenance for Visualization

Data Provenance

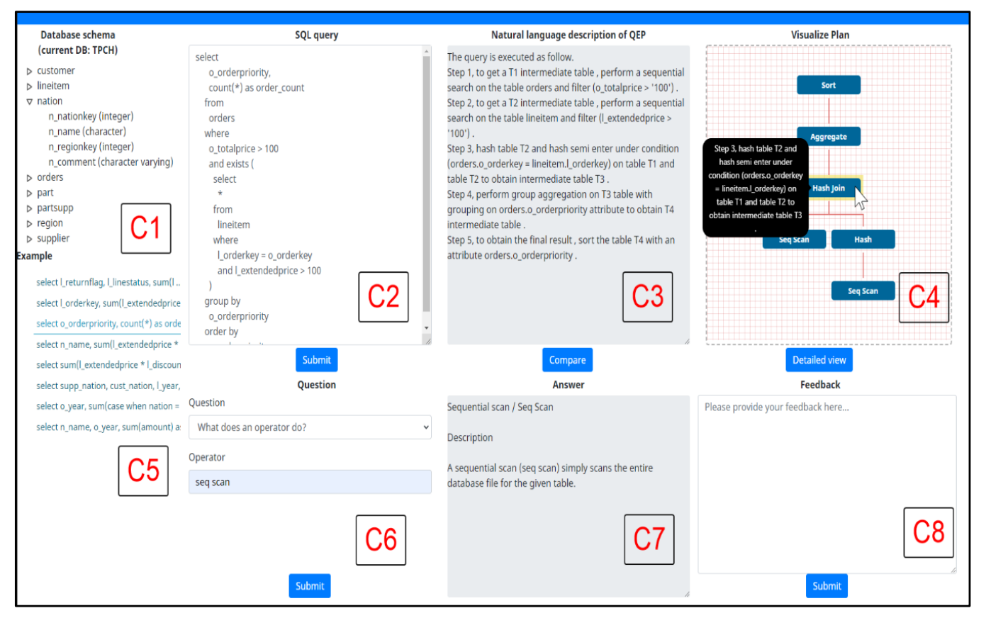
Data Provenance is metadata describing the origin of data values and how it was processed throughout the execution.

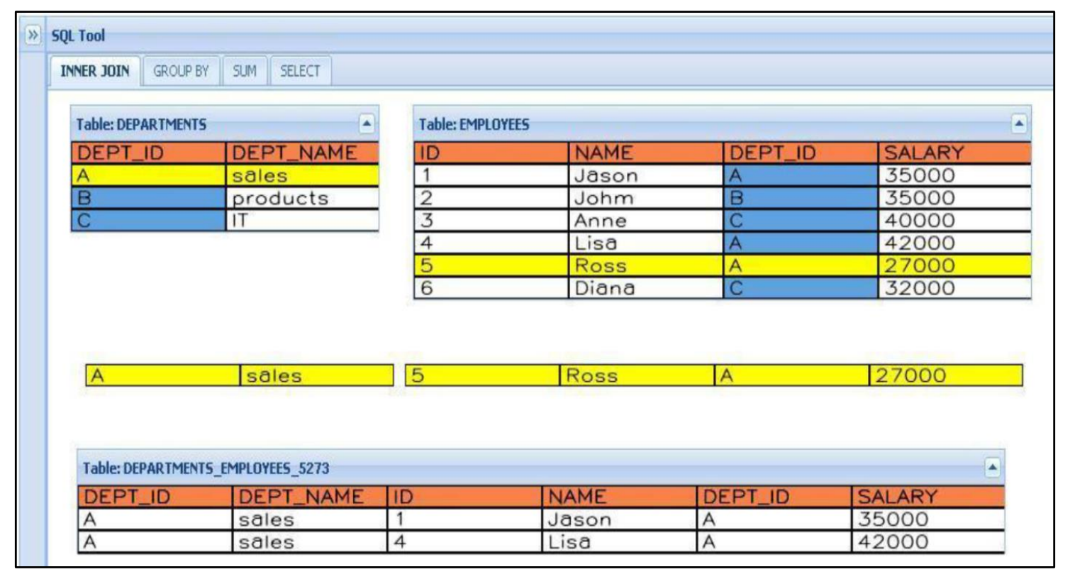
Once provenance is captured, **How should we convey it?**

Related Implementations

LANTERN [1] – Physical Plan Execution

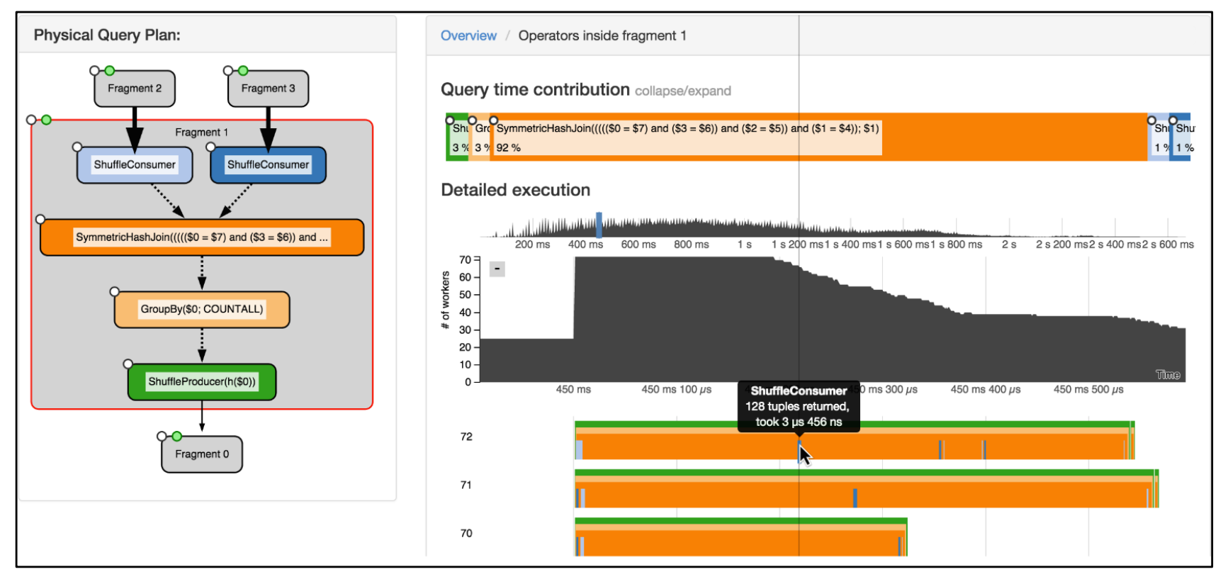
- Natural Language descriptions
- Physical operator explanations
- Can arrange to full sequence steps





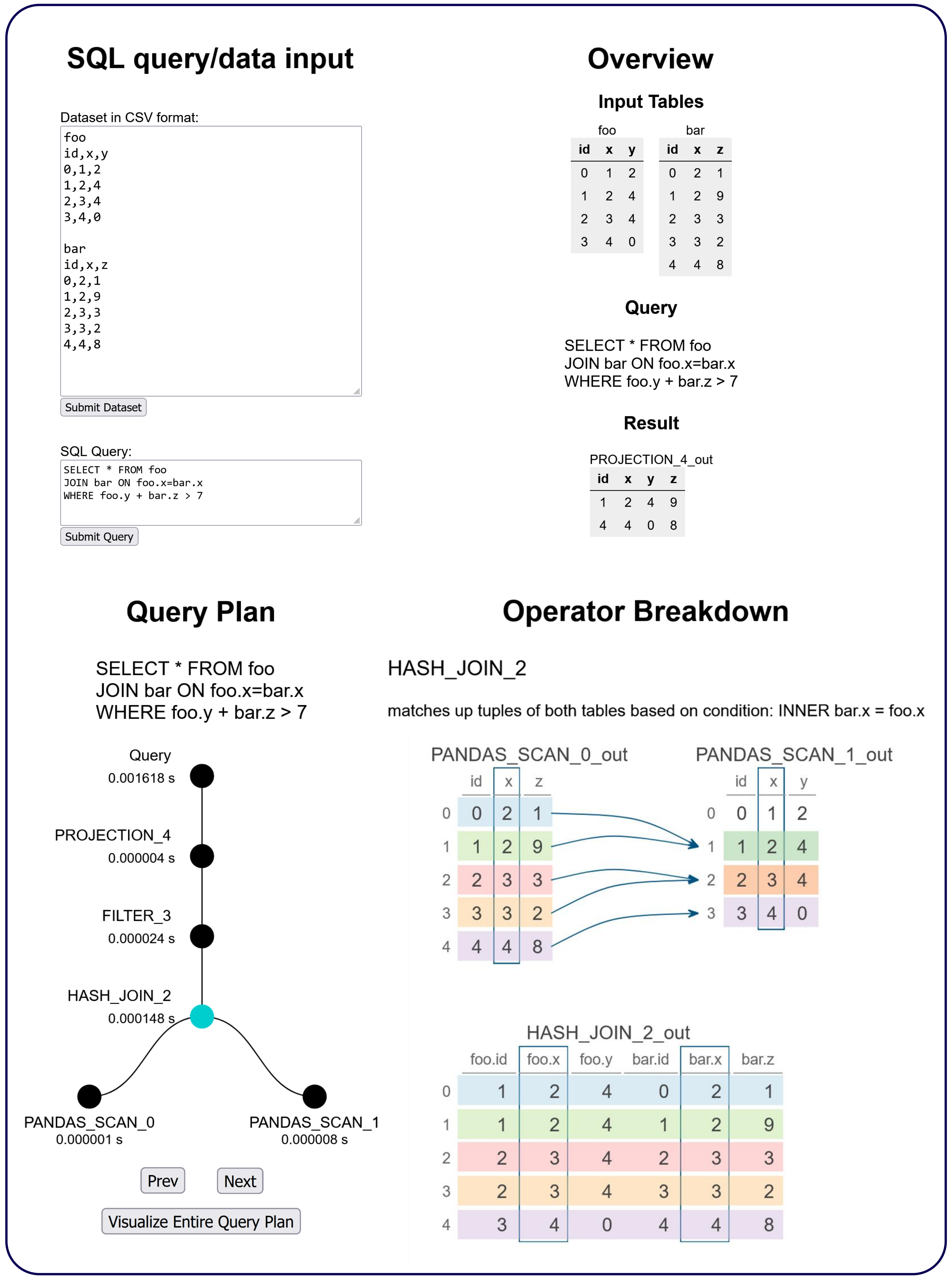
SAVI [2] –Query Syntax Breakdown

- Abstracted to query-level operations
- Generates intermediate datasets
- Animates transformations



Perfopticon [3] – Distributed Database Performance Analysis

- Physical plan overview
- Performance details by operator



SQL query/data input

Dataset in CSV format:

```
foo
id,x,y
0,1,2
1,2,4
2,3,4
3,4,0

bar
id,x,z
0,2,1
1,2,9
2,3,3
3,3,2
4,4,8
```

SQL Query:

```
SELECT * FROM foo
JOIN bar ON foo.x=bar.x
WHERE foo.y + bar.z > 7
```

Overview

Input Tables

foo

id	x	y
0	1	2
1	2	4
2	3	4
3	4	0

bar

id	x	z
0	2	1
1	2	9
2	3	3
3	3	2
4	4	8

Query

SELECT * FROM foo
JOIN bar ON foo.x=bar.x
WHERE foo.y + bar.z > 7

Result

PROJECTION_4_out

id	x	y	z
1	2	4	9
4	4	0	8

Query Plan

SELECT * FROM foo
JOIN bar ON foo.x=bar.x
WHERE foo.y + bar.z > 7

Operator Breakdown

HASH_JOIN_2

matches up tuples of both tables based on condition: INNER bar.x = foo.x

PANDAS_SCAN_0_out

	id	x	z
0	0	2	1
1	1	2	9
2	2	3	3
3	3	3	2
4	4	4	8

PANDAS_SCAN_1_out

	id	x	y
0	0	1	2
1	1	2	4
2	2	3	4
3	3	4	0

HASH_JOIN_2_out

	foo.id	foo.x	foo.y	bar.id	bar.x	bar.z
0	1	2	4	0	2	1
1	1	2	4	1	2	9
2	2	3	4	2	3	3
3	2	3	4	3	3	2
4	3	4	0	4	4	8

SQL Query Execution Visualizer

Considerations

- Be able to follow along the entire query execution
- Show how values are derived and processed
- Allow to focus on a singular operation
- Visuals should resemble actual operator behavior

Implementation

Query Plan Tree Diagram

Query Plan displayed as interactive diagram, acts as point of reference for entire query

Operator Transformations

Visualizations for each physical operator, with unique annotations for each operator *type*

Intermediate Datasets

All visualizations display inputs (output of the previous operators) and how it is transformed by this operator to the output for the next operator

Operator Steps

Can view operators in isolation, or all at once arranged in post-order to show entire sequence

References

1. Chen, P., Li, H., Bhowmick, S. S., Joty, S. R., & Wang, W. (2022, June). LANTERN: Boredom-conscious Natural Language Description Generation of Query Execution Plans for Database Education. In Proceedings of the 2022 International Conference on Management of Data (pp. 2413-2416).
2. Cembalo, M., De Santis, A., & Ferraro Petrillo, U. (2011, October). SAVI: a new system for advanced SQL visualization. In Proceedings of the 2011 conference on Information technology education (pp. 165-170).
3. Moritz, D., Halperin, D., Howe, B., & Heer, J. (2015, June). Perfopticon: Visual query analysis for distributed databases. In Computer Graphics Forum (Vol. 34, No. 3, pp. 71-80).