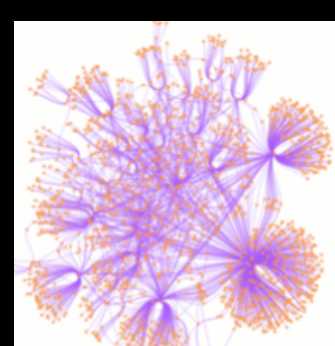
Graph-based Exploration of Non-graph Datasets

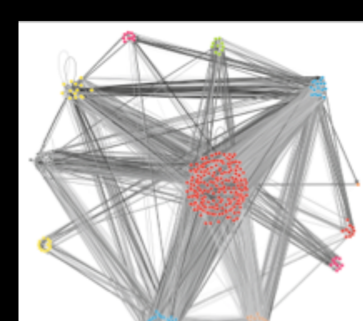
Udayan Khurana
IBM Research
ukhurana@us.ibm.com

work with Deepak Turaga, Srinivasan Parthasarathy

Graph Analytics

- Graphs capture interconnections or interactions.
- Graphs are omnipresent - social media, emails, news stories, financial records, system logs, biological structures, …
- Network analysis provide useful insights
  - Node centrality metrics tell relative importance of entities
  - Global network metrics reflect collective behavior.
- Last few years have seen a rapid growth graph database management systems and graph analysis software.



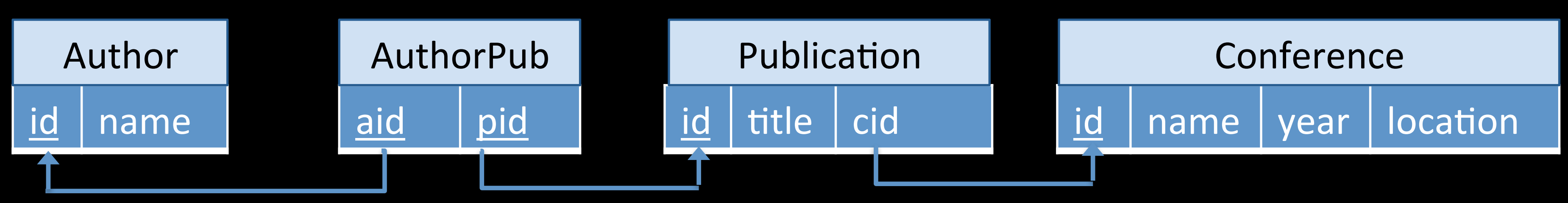


Non-graph Datasets

- Majority of world's data is present in non-graph formats
  - Relational Databases, JSON, XML, CSV, Plain Text, …
- Extracting graphs from a dataset is hard:
  - Identifying the appropriate graph
  - Writing code to perform the extraction
  - Time, skill and effort from an analyst or data scientist

Manual Graph Extraction

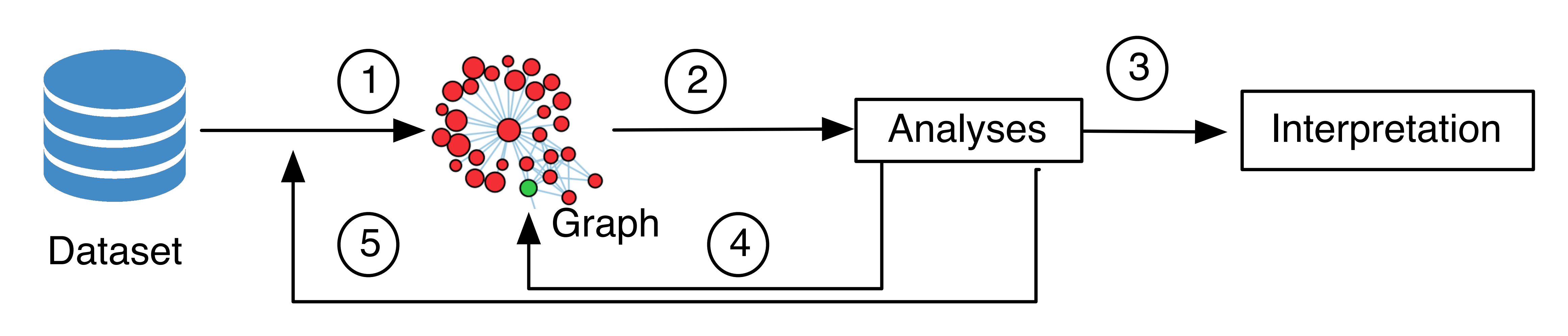
- The process of manual graph extraction is based on visual inspection and understanding of the data/domain.
- However, it is (a) time consuming; (b) Requires coding/query writing; (c) Possible to miss out on useful graphs
- E.g., In DBLP database:



- Example of an extraction query (Co-authorship graph):

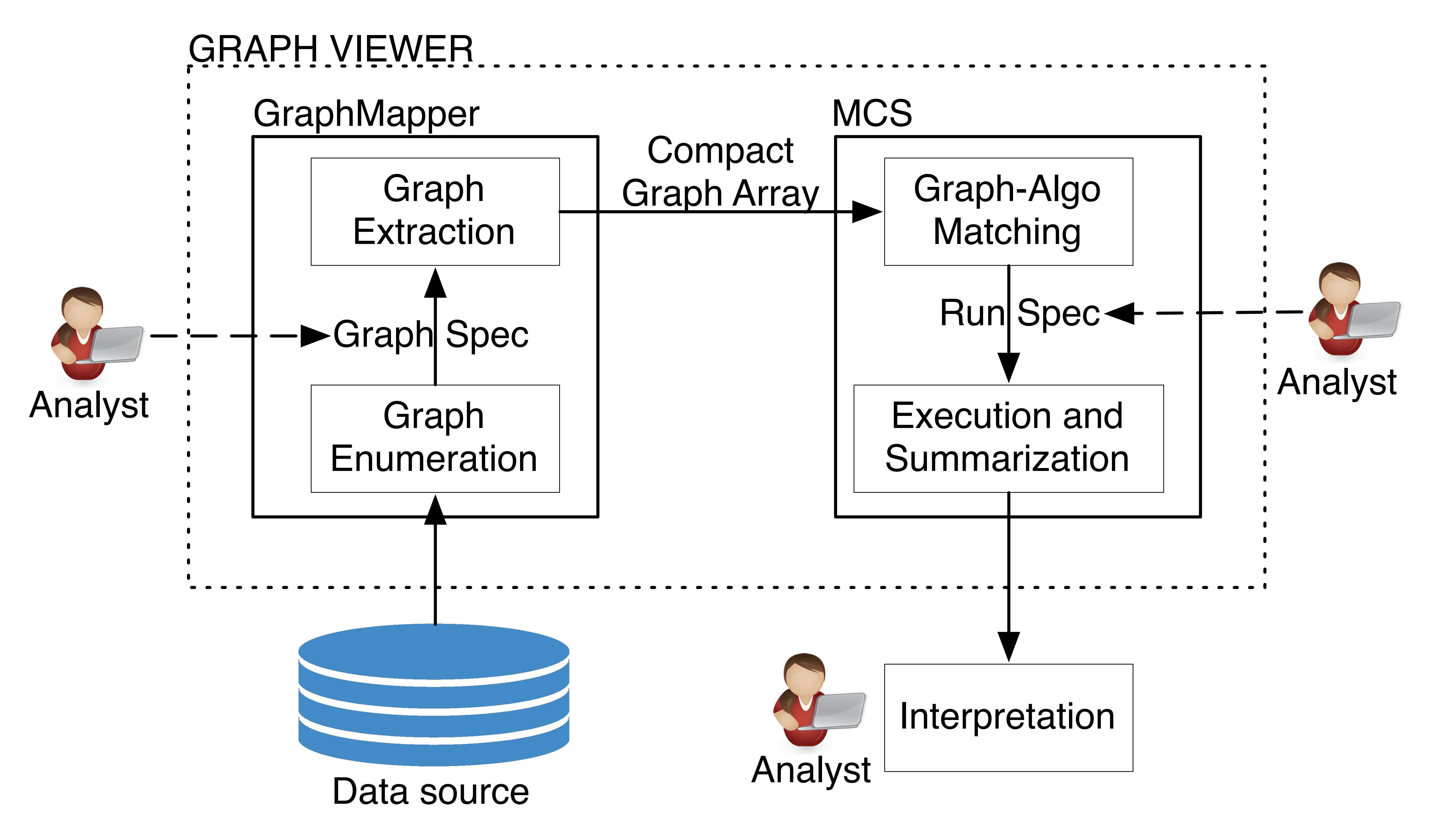
```
Nodes: Select Author.id from Author;
Edges: Select A1.id, A2.id from Author A1,
Author A2, AuthPub AP1, AuthPub AP2 WHERE
AP1.aid = A1.aid AND AP2.aid = A2.aid AND
AP1.pid = AP2.pid;
```

Graph Analytics Methodology



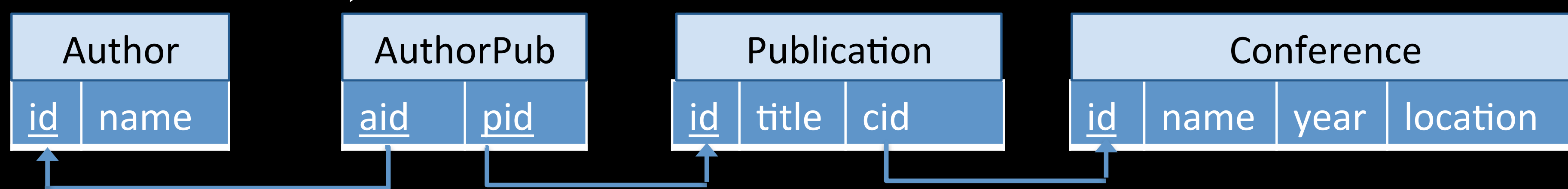
- Graph Analytics is performed in an iterative manner:
  - (1) graph extraction from the dataset
  - (2) running an analysis algorithm on the graph
  - (3) interpretation of results
  - (4) running a different algorithm on the same graph
  - (5) finding another graph from the dataset.

GraphViewer: Overview

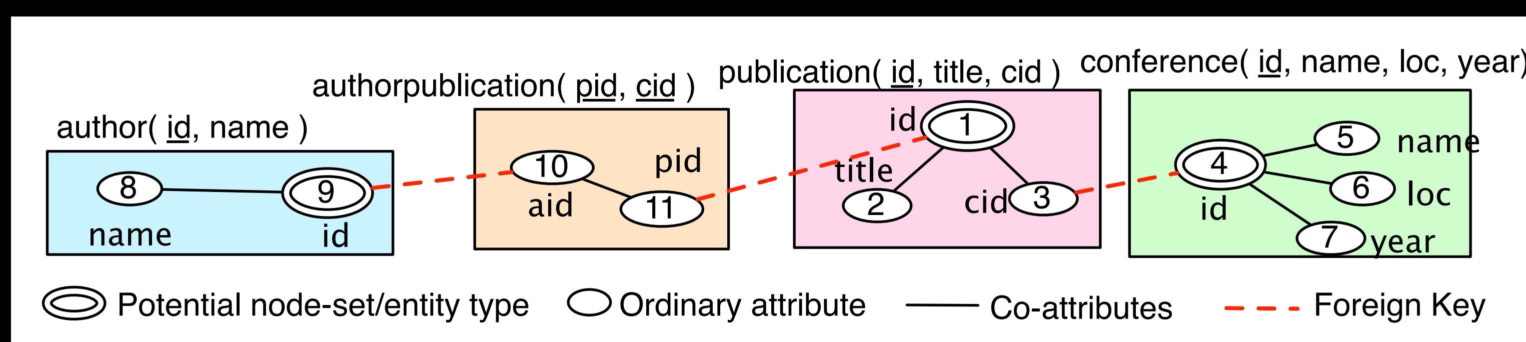


Alternative Graph Extraction Approach: Enumeration

- We propose an automated graph enumeration approach.
- Given a schema,



- We construct a *Schema Graph*



- Any path starting and ending at a terminal node describes a graph definition.
    - E.g., 9 -> 10 -> 11 -> 10 -> 9 corresponds to the co-author graph described earlier.

Extracting Graphs from Non-structured Data

- Unstructured data such as text doesn't provide a schema
- Approach: Use Natural Language Processing to annotate:
  - Entities: Persons, Locations, Organizations, etc.
  - Relationships: Subject-Object-Predicate, Co-occurence
- For semi-structured data such as JSON, XML, CSV, we
  - Adopt dual approaches - structured and unstructured
  - Try to introduce structure as well as treat it as text

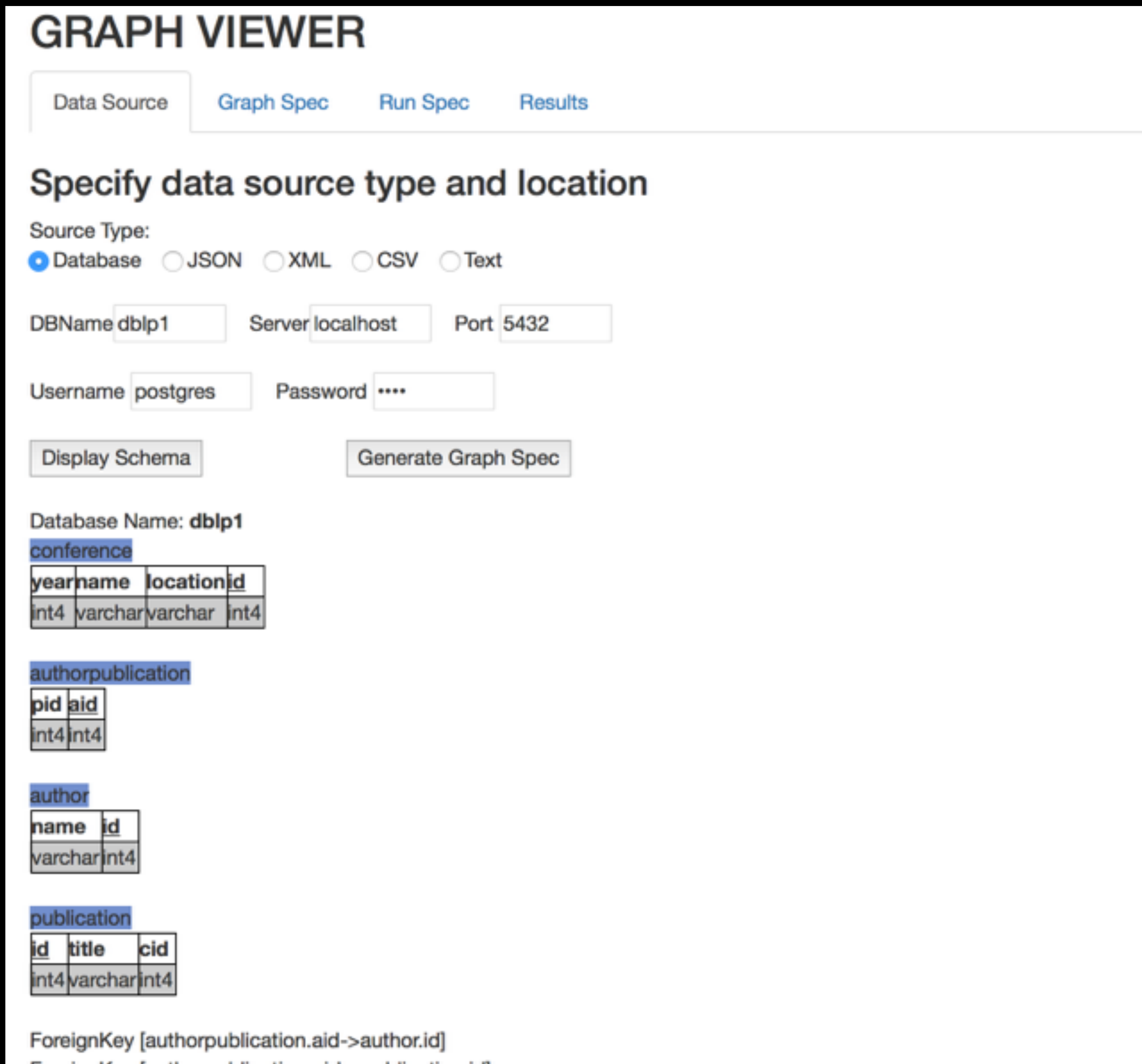
Metric Computation and Summarization

- Graph enumeration produces multiple graphs,
  - $S_G = \{G_1, G_2 . . . \}$
- These are matched with different algorithms,
  - $S_A = \{A_1, A_2 . . . \}$
- We run $O(|S_A| x |S_G|)$ graph metric algorithms
  - Produces $O(|S_A| x |S_G| x |E|)$ results where E is the set of all entities
- The user can arrange the results in different ways
  - Aggregate by entity, entity-type, graph or algo.
  - Filter, Sort and Join to perform data exploration.
  - Potentially use data cubes.

Demo

- View this demo at VLDB 2016 at the following times (Graph and Semistructured Data):
  - Tuesday 4:00-5:30 PM (Maple - 3a)
  - Wednesday 11:15-12:45 (Maple - 3b)
- http://localhost:8080/GraphViewer

Demo: Data Source



Demo: Graph Specification

GRAPH VIEWER

Data Source | Graph Spec | Run Spec | Results

Node Specification:

NS-ID	Node Set	Node Count	Node Properties
N1	publication.id	1677	publication.titlepublication.cid,
N2	conference.id	27	conference.yearconference.nameconference.location,
N3	author.id	2576	author.name,

Graph Specification:

GID	Nodesets	Relationships	Est. Edges	Graph Summary	Graph Layout
G1	N1	Path=publication.id,publication.title,	1677	Graph Summary	View Graph
G2	N1	Path=publication.id,publication.cid,	195007	Graph Summary	View Graph
G3	N2	Path=conference.id,conference.year,	75	Graph Summary	View Graph
G4	N2	Path=conference.id,conference.name,	249	Graph Summary	View Graph
G5	N2	Path=conference.id,conference.location,	147	Graph Summary	View Graph
G6	N3	Path=author.id,author.name,	2578	Graph Summary	View Graph

Add Graph Definition | Edit Graph Criteria | Create Run Spec

Clear Graph Spec

Demo: Run Specification

GRAPH VIEWER

Data Source | Graph Spec | Run Spec | Results

Run Specification:

		Graph	Algo	Max Count
Remove	Edit	G1	LocalClusteringCoefficient	10
Remove	Edit	G1	BetweennessCentrality	10
Remove	Edit	G1	PageRank	10
Remove	Edit	G1	ClosenessCentrality	10
Remove	Edit	G2	LocalClusteringCoefficient	10
Remove	Edit	G2	BetweennessCentrality	10
Remove	Edit	G2	PageRank	10
Remove	Edit	G2	ClosenessCentrality	10
Remove	Edit	G3	LocalClusteringCoefficient	10
Remove	Edit	G3	BetweennessCentrality	10
Remove	Edit	G3	PageRank	10
Remove	Edit	G3	ClosenessCentrality	10
Remove	Edit	G4	LocalClusteringCoefficient	10
Remove	Edit	G4	BetweennessCentrality	10
Remove	Edit	G4	PageRank	10

Demo: Exploring Results

GRAPH VIEWER

Data Source | Graph Spec | Run Spec | Results

Browse Results

Run Complete

Results are stored in the following schema. Please write a SQL query or populate one by clicking an option below.

Results(entityID, entityType, graph, algo, score)

Entity Average (Decreasing) | EntityType Average (Decreasing) | Entity Score (Decreasing) | Graph Average (Decr

```
SELECT entityid, AVG(score) FROM vals GROUP BY entityid order by avg(score) desc;
```

Limit 20 | Query Results DB

entityid	avg
N2.91	1.00000000000000000000
N2.33	1.00000000000000000000
N3.1744	1.00000000000000000000
N3.2447	1.00000000000000000000
N2.86	1.00000000000000000000
N2.89	1.00000000000000000000
N2.85	1.00000000000000000000
N2.35	1.00000000000000000000

System Aspects

- Extensibility:
  - New data types; new analytics; different aggregators
- Storage efficiency:
  - Shared nodes and edges in multiple fetched graphs present redundancy.
  - We store the graphs efficiently in an overlaid manner
- Runtime computational sharing:
  - Multi-query optimization in relational databases eliminate computational redundancy
  - Sharing text parsing across multiple graphs

Contributions

- We propose a new methodology for performing graph-based analysis on of non-graph datasets:
  - Through automated graph enumeration
  - Through automated code assembly and deployment
  - Combines an analyst’s input with automated specs
- It assists an analyst in graph-based analysis by:
  - Reducing time, skill and effort
- It is efficient in exploring different kinds of datasets through graph-based analytics

Research Problems

- Efficiently find graphs, subgraphs or nodes that satisfy a certain criteria:
  - For instance, “list graphs with density > d”.
  - Can we do it without extracting all possible graphs?
- Predict the more useful graph/analytics for a given task:
  - Use training examples to learn correlations.
  - Avoid generating all graphs.
- System efficiency in extraction and execution of:
  - Multiple graphs and multiple algorithms.