The Quest for Schemas in Graph Databases

Angela Bonifati\textsuperscript{1}, Stefania Dumbrava\textsuperscript{2,3}, Emile Martinez\textsuperscript{4}, Nicolas Mir\textsuperscript{2}

\textsuperscript{1}Lyon 1 University & LIRIS CNRS, France
\textsuperscript{2}ENSIIE & \textsuperscript{3}Institut Polytechnique de Paris, France
\textsuperscript{4}ENS de Lyon
Property Graph Schemas: State of Affairs

- Ongoing PG Schema standardisation process (ISO SC32/WG3) in collaboration with PGSWG\(^a\).
- Early proposal for a concise DDL for Cypher with Neo4j folks along with mechanisms for schema validation and evolution\(^b\).
- While waiting for a standard PG schema, we need mechanisms for schema discovery from property graph instances (focus of my talk).

\(^a\)https://ldbcouncil.org/gql-community/pgswg/
Schema Discovery

Interconnected Data:

- ubiquitous (Semantic Web, social networks, scientific repositories,...), heterogeneous & semi-structured.

Graph Databases:

- NoSQL store for efficiently storing & processing graph-shaped data.
- No a priori schema constraints $\rightarrow$ error-prone data integration
- Underlying property graph model
  (labeled multigraph with key/value lists attached to nodes & edges)
  $\rightarrow$ rich formalism amenable to schema discovery
• Existing schema inference mechanisms are basic:
  • no hierarchies,
  • no complex types.

• Recent work on schema inference using MapReduce (MRSchema): a
  • only considers either node labels or node properties → trade-off
  • property co-occurrence information loss (label-oriented approach)
    vs. extraneous type inference (property-oriented approach).

---

Overview of DiscoPG’s Algorithms

**Static Case**: discover the schema of a static graph dataset $G$.

- **GMM–S**: novel *hierarchical clustering algorithm*.
  - Based on fitting a Gaussian Mixture Model (GMM).
  - Accounts for both node label & property information.

**Dynamic Case**: update the schema of $G$ upon modifications.

- **I–GMM–D**: incremental approach; reuses GMM–S’s clustering.
- **GMM–D**: recomputation approach; memoization-based GMM–S.
A property graph $\mathcal{G}$ is a tuple $(\mathcal{V}, \mathcal{E}, \rho, \lambda, \sigma)$, where:

- $\mathcal{V}$ and $\mathcal{E}$: disjoint finite sets of vertices, and edges,
- $\rho : \mathcal{E} \to (\mathcal{V} \times \mathcal{V})$: associates each edge with a pair of nodes,
- $\lambda : (\mathcal{V} \cup \mathcal{E}) \to \mathcal{P}(\mathcal{L})$: associates a vertex/edge with a set of labels,
- $\sigma : (\mathcal{V} \cup \mathcal{E}) \times \mathcal{K} \to \mathcal{P}(\mathcal{N})$: associates vertex/edges with properties and, for each property, assigning a set of values from $\mathcal{D}$.

---

Property Graph Schemas

**Base Types (\(BT\)):** set of element types \((L, K, O, E_b)\), where:

- \(L \in \mathcal{L}\): set of labels,
- \(K \in \mathcal{K}\): set of property names,
- \(O \subseteq K\): subset of optional property names,
- \(E_b \subset BT\): set of element types \(b\) extends.

**Example:**

```json
{'Post': {
    'creationDate': '2015-06-24T12:50:35.556+01:002',
    'locationIP': 42, 'browser': 'Chrome',
    'length': 10, 'language': 'latin',
    'content': 'Lorem ipsum'}}
```

LDBC Post node instance

**Base type:** \((\{Post\}, K, \{language, content\}, \emptyset)\),

where \(K = \{creationDate, locationIP, browser, length\}\).
Figure 1: LDBC Property Graph
GMMSchema Methodology

**Figure 2: System Workflow**

**Idea:**

- Gaussian Mixture Model (GMM)\(^a\) to discover hierarchical node types.
- for every node label, run GMM algorithm to fit a mixture of normal distributions & use the resulting model for clustering.
- re-iterate procedure in each sub-cluster.

---

GMMSchema Base Algorithm (GMM-S)

- Collect node labels $\mathcal{L}_G$ & their number of occurrences.
- For each label $L \in \mathcal{L}_G$ (in descending frequency order), iteratively process the set $C$ of all nodes with label $L$.
- Reference Base Type ($b_{\text{ref}}$): most general type for $C$ built at each step from all of its node labels accounts for the most frequent properties.
- Feature vector: constructed from the similarity scores of all nodes in $C$ w.r.t $b_{\text{ref}}$ & used to fit a GMM model.
- EM algorithm: parameter estimation for Gaussian mixture discovered node types.
- Hierarchical clustering ($C_{\mathcal{H}}$): update $b_{\text{ref}}$ with overlapping properties, record $C$ sub-clusters & recursive call in each.
Illustrating the discovery of the sub-types for Post-labeled nodes:

- **Parent Node Base Type**: \[ b = (\{Post\}, K, \emptyset, \emptyset) \], where \( K = \{creationDate, locationIP, browser, length\} \).
- Run GMM; the new reference nodes are:
  \[ b_1 = (\{Post\}, K, \{language, content\}, \{b\}) \] and 
  \[ b_2 = (\{Post\}, K, \{imageFile\}, \{b\}) \]
- Repeating the procedure in each sub-cluster does not infer new types, as all nodes in each share the same properties. → **new discovered sub-types**: Post1 and Post2.
Figure 3: LDBC Property Graph GMMSchema
Experimental Evaluation: Schema Quality (I/II)

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Nodes</th>
<th>Edges</th>
<th>Node Labels</th>
<th>Edge Labels</th>
<th>Unlabeled</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDBC</td>
<td>1577397</td>
<td>8179418</td>
<td>7</td>
<td>14</td>
<td>0</td>
</tr>
<tr>
<td>Mb6</td>
<td>486267</td>
<td>961571</td>
<td>10</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>Fib25</td>
<td>802473</td>
<td>1625428</td>
<td>10</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>Covid19</td>
<td>36025729</td>
<td>59768373</td>
<td>121</td>
<td>168</td>
<td>474</td>
</tr>
</tbody>
</table>

**Figure 4:** Dataset characteristics prior to schema discovery.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Node Types</th>
<th>Edge Types</th>
<th>Subtype Edges</th>
<th>Hierarchy Depth</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDBC</td>
<td>17</td>
<td>36</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>Mb6</td>
<td>19</td>
<td>27</td>
<td>14</td>
<td>4</td>
</tr>
<tr>
<td>Fib25</td>
<td>26</td>
<td>106</td>
<td>21</td>
<td>6</td>
</tr>
</tbody>
</table>

**Figure 5:** Dataset characteristics with GMMSchema discovery.
Experimental Evaluation: Schema Quality (II/II)

- 2-3 discovered types/label & 3 orders of magnitude more edge types.
- MRSchema infers up to 3 times more node types, up to 3 orders of magnitude more edge types, up to 7 orders of magnitude more subtype edges (for mb6) → up to double the hierarchy depth.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Rand Index</th>
<th>AMI</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDBC</td>
<td>0.96</td>
<td>0.91</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>Mb6</td>
<td>0.79</td>
<td>0.49</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>Fib25</td>
<td>0.75</td>
<td>0.41</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>Covid19</td>
<td>0.94</td>
<td>0.71</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
</tr>
</tbody>
</table>

**Figure 6:** GMMSchema clustering quality estimates.

- However, most MRSchema inferred nodes are spurious.
- GMMSchema: perfect accuracy by also leveraging node labels.
Experimental Evaluation: GMMSchema Runtime

Figure 7: GMMSchema vs. MRSchema total avg. runtimes.

(a) LDBC, Fib25, Mb6

(b) Covid19

→ GMMSchema speeds-up schema discovery:
  \( \times 5 \) (for Mb6) & \( \times 8 \) (for LDBC and Fib25).
Inputs:

- Discovered schema for $G$, as computed by GMM-S.
- Graph updates $\Delta$: set of nodes inserted into $G$.

For each node in $\Delta$:

- Compute its similarity score w.r.t every reference base type corresponding to the clustering $C_H$.
- Assign it to the cluster maximizing this similarity.

Performance only depends $|\Delta|$ & $|C_H|$:

$\rightarrow$ multiple efficient iterations

$\rightarrow$ highly robust in practice (maintains schema quality).
GMM-D

Inputs:

- Discovered schema for \( G \), as computed by GMM-S.
- Graph updates \( \Delta \): set of nodes inserted into \( G \).

Process the *updated graph* using GMM-S, optimized to:

- track the sub-clusters unchanged by the classification step.
  (no nodes assigned, due to reference base type dissimilarity)
- memoize & avoid recursive calls in these sub-clusters.

W.r.t I-GMM-D:

\[\uparrow \text{convergence}, \uparrow \text{iteration-wise runtime}, \downarrow \text{robustness}.\]

Trade-off: performance vs. quality
DiscoPG System\(^1\) – Workflow Diagram

\(^1\)Accepted in VLDB 2022 (demo track)
Conclusions

- DiscoPG: first schema discovery approach for property graphs (accounting for both node labels & properties).
- addresses previous limitations (incomplete/spurious node inference) while showing superior accuracy & performance.
- promise of employing statistical methods for schema discovery.
- extensibility to future standard PG schema languages
Perspectives

- integrating topological information (graph embeddings),
- extension to streaming graphs,
- discovery of property graph constraints (PG-Keys, ...)

PG-Keys: Keys for Property Graphs. [SIGMOD 2021]a

Thank you!

aJoint work with the Property Schema Group.
References


