Aim of the Talk

- Enhance the Graph Model (GM) for data modelling and answer the following questions:
  - Is the GM suitable for data schemas?
  - Which enhancements to the GM are needed?
  - Is it better matching the way we communicate reality?
  - What is the semantic expressiveness of the GM?
  - Is there support for multiple abstraction levels?

- Contents
  - Present the GM with some enhancements for our purpose
    - Formally compact, yet sufficient for the target aim
  - Apply and compare the GM to prevailing data models
    - Show and discuss the results (benefits and pitfalls)
Advocates of the GM like Robinson et al. recommend to use specification by example which builds on real objects as the following (p. 42):

The problem with this is that we cannot exemplify all situations.

The object “Review” depends on the existence of a “User” and a “Performance”

We cannot know if Billy is allowed to have multiple reviews (on the same performance?)

In order to express this semantics it is necessary to abstract and specify integrity constraints.

This means we have to deal with abstract things (like a generic Person) and not only with real objects (like Billy).

A mathematical (directed) Graph \( G = (V, E) \) is defined as

- a set of Vertices \( V \) and
- a set of Edges \( E \) connecting 2 (ordered) vertices \((u, v)\), with \( u, v \in V \).

The vertices can be numbered for identification and the edges may have „weight“ for calculating the cost of a path.

Shortcomings for data modelling:

1. Two modelling elements are not sufficient to express data structures
   - e.g. even the relational model has 3 modelling elements
   - We want to distinguish different association types, e.g. inheritance, aggregation

2. The Graph Model is originally instance based
   - If we apply the GM on the Schema level, how can we ensure integrity constraints e.g. capture the multiplicity of an association?
Solving Shortcoming (1): Labeled Property Graph

- **Use 4 Model elements to capture more semantics**
  - Nodes (Vertices) ≈ objects
  - Lines (Edges) either directed or undirected ≈ related objects
  - Properties (of vertices and/or edges) ≈ detail information as key-value pairs
  - Labels (of vertices) group nodes ≈ type/class name

- **Definition: Labeled Property Graph (LPG)**
  - A (Labeled) Property Graph PG = (V, E, P, L) is a Graph where any x ∈ V ∪ E can
    - have a subset P_x ⊆ P of properties (e.g. key-value pairs) attached to x.
  - Nodes v and Edges e can have labels L_v, L_e.
  - Labels serve on the meta-level (e.g. type)

- **Labeled Property Graph (LPG) Example** → see Spyratos et al. 2)

![Labeled Property Graph Example Diagram]

Solving Shortcoming (2): Graph-based Data Model (GDM)

- **J. Hidders proposed a GDM for Schema Graphs** 3) based on the Graph-Oriented Object Database 4) (GOOD) model.
  - The idea is to use nodes for all meta-data (i.e. attributes & classes, )
  - I believe, this makes the model too large and confusing

- **We use the LPG as basis**
  - Model elements (viewed as meta-data)
    - Nodes (Vertices) ≈ class/type (variable)
    - Lines (Edges) either directed or undirected ≈ association class/type
    - Properties (of vertices or edges) ≈ as property name:domain pairs
    - Labels (of vertices) classify nodes ≈ class/type name
  - Add cardinality to the edges → see Angles 5)
    - Use UML like notation to specify the multiplicity of an association
    - Special types of association like generalization, aggregation, etc. may be expressed as labels to an edge.

![Graph-based Data Model (GDM) Example Diagram]
Example (1/4)

EGM for tabular/relational data structures

**Source structure**
- Table T(key:id, c2:num, c3:string)

**corresponding Graph Schema**
- Table T(key:id, c2:num, c3:string)
- Join Table JT

**Foreign key**
- T1
  - k1: id
  - FK(k1)
  - col1
- T2
  - k2: id
  - col2

**Join Table**
- JT
  - FK(k1)
  - FK(k2)

Example (2/4)

EGM for a hierarchical structure (e.g. XML)

**XML document / schema**
```
<bookstore>
  <category>
    <book>
      <title>example from w3schools.com</title>
      <author>...</author>
      <year>...</year>
      <price>...</price>
    </book>
  </category>
</bookstore>
```

**corresponding Graph Schema**
- bookstore:
  - book
    - category:
      - title:
      - author:
      - year:
      - price:

- XML attribute
- XML element
- Label price can't hold value
- The order of the elements cannot be expressed
- or more compact: all book data in one node.
- distinguish between XML attribute and element only by naming convention
Example (3/4)

**EGM for an object-oriented structure**

- **Class**
  - Cname
  - Attributes
  - Methods

- **Association**
  - \(<\text{cat}>\>
  - \(C1 \rightarrow_{m..n} C2\)

- **Aggregation**
  - \(C1 \rightarrow 4 C2\)

- **Generalization**
  - \(C1 \leftarrow 4 C2\)

- **Association class**
  - \(C1 \rightarrow \text{attrib.} \rightarrow C2\)

Example (4/4)

**EGM compared to the Entity-Relationship Model (ERM)**

- **ERM**
  - \(E1 \rightarrow_{n} R1 \rightarrow_{m} E2\)

- **EGM**
  - \(E1 \rightarrow_{0..n} R3 \rightarrow_{0..m} E2\)

For higher degrees of relationships a hypergraph model would be needed.
### Answering the Questions

<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Is the GM suitable for data schemas?</strong></td>
<td>Yes, if the model is enhanced with properties, labels and edge cardinality</td>
</tr>
<tr>
<td><strong>Is it better matching the way we communicate reality?</strong></td>
<td>No, the models considered in the examples all basically rely on objects/entities/elements and associations/relationships.</td>
</tr>
<tr>
<td><strong>What is the semantic expressiveness of the GM?</strong></td>
<td>The EGM has less modelling power than XML schema and UML class diagrams, but more than the RM. It is comparable to the ERM</td>
</tr>
<tr>
<td><strong>Is there support for multiple abstraction levels?</strong></td>
<td>Not by the model itself, responsibility of the designer</td>
</tr>
<tr>
<td><strong>Consequences of using the GM vs. other data models?</strong></td>
<td>In general there is no real benefit as the modelling decisions remain the same except if the target database is a Graph Database (no semantic mismatch) or link analysis is important</td>
</tr>
</tbody>
</table>

### Lessons learned

<table>
<thead>
<tr>
<th><strong>Use the EGM on the meta-level</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Model entities/classes as nodes</td>
<td></td>
</tr>
<tr>
<td>Use labels for class names</td>
<td></td>
</tr>
<tr>
<td>Model detail information (attributes) as properties</td>
<td></td>
</tr>
<tr>
<td>It is a modelling decision whether to model a data element as property or as node (compactness vs. precision)</td>
<td></td>
</tr>
<tr>
<td>Model associations as edges and add properties if needed</td>
<td></td>
</tr>
<tr>
<td>Use labels as association types (is-a, aggregate, etc.)</td>
<td></td>
</tr>
<tr>
<td>Add cardinalities to the association type</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>In real world scenarios the GM tends to become large and confusing</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Suppress properties in the diagram</td>
<td></td>
</tr>
<tr>
<td>Use higher abstraction level aggregates like category, stereotype, component, etc. to provide an overview model</td>
<td></td>
</tr>
<tr>
<td>Model partial structures separately</td>
<td></td>
</tr>
</tbody>
</table>
References

1) I. Robinson et al.: Graph Databases, 2nd ed., O'Reilly Media, 2015


3) J. Hidders: “Typing Graph-Manipulation Operations”, Proc. 9th International Conference on Database Theory (ICDT), 2003, pp. 391-406


Discussion