Model-Independent Design of Knowledge Graphs

Luigi Bellomarini (1)
Andrea Gentili (1)
Eleonora Laurenza (1)
Emanuel Sallinger (2,3)

The opinions expressed and conclusions drawn are those of the authors and do not necessarily reflect the views of the Bank of Italy.
How the design story began

- Reveal power
- finding controllers
- studying the structure of Italian market
- studying dispersion of control
- global shareholding analysis
- intercept and prevent hostile takeovers

- AI-aided banking supervision
- detect collusion and do forensics
- support anti-money laundering
- detecting ultimate beneficial owners
- analyze qualifying holdings
- evaluate risks
- guarantee compliance
- perform enhanced due diligence
- know real cash flows

* And much more
  - evaluate and guarantee anonymity of microdata
  - model propagations (e.g., of shocks)

KGs in Economics & Finance
Knowledge Graphs \text{KG19, or building a financial KG}

\[ \sum \text{Business}(x) \rightarrow \text{Controls}(x, x) \quad (1) \]
\[ \text{Controls}(x, y), \text{Owns}(y, z, w), v = \text{sum}(w, \langle y \rangle), v > 0.5 \rightarrow \text{Controls}(x, z) \quad (2) \]
A Real Case, with Very Large Graphs

Ownership graph with European graph:

- 10 million individuals
- 30 million ownerships
- 20 million roles (e.g. CEO)
- 200k company events (e.g. M&A)

Plus, derived knowledge:

- company control
- relevant influence
- family links

- 264 million nodes
- 660 million edges
- 1+ billion properties
1. how to design, understand and communicate a KG of a complex domain?
2. how to make the deployed KG schema mirror the conceptual specification?
3. how to express and apply complex business behaviour in the KG?

in the presence of heterogeneity?
• The goal of the **KG designer** (knowledge engineer, etc.) is to **design a KG schema**
• A **KG schema** is a representation of the domain of interest in terms of its main concepts, their attributes, the connections between them, etc.

• **Model**: the set of constructs that can be used to **define database schema**
  • in our context, the **schema of the extensional component of the KG**.

• Many different schemas (**schema heterogeneity**)  
  • the goal of design (matter of art)

• Many different models (**model heterogeneity**)  
  • different sets of constructs that can be used  
  • design-oriented models are typically conceptual  
  • conceptual models are typically coupled to specific logical models
Model Heterogeneity in the relational world (in theory)

Conceptual, design-oriented model (e.g., Entity-Relationship)

Logical model (e.g., relational)

Physical model (e.g., relational)
Model Heterogeneity in the relational world (in practice)

Conceptual, design-oriented model
(e.g., Entity-Relationship)

Logical model
(e.g., relational)

Physical model
(e.g., Oracle, DB2, Postgres, MySQL, SQLite, …)

Atzeni, Cappellari, Torlone, Bernstein, Gianforme
Model-Independent Schema Translation, VLDBJ 2008
Model Heterogeneity in the KG world (in theory)

Conceptual, design-oriented model

Logical model

- Proposals for the extensional component
  - PG-Schema [Angela’s keynote]

- No broadly adopted methodology for the intensional component

Directed edge-labelled graphs

Property graphs

Physical model

(e.g., graph DBs)

- Some overlap of conceptual and logical models
Model Heterogeneity

Conceptual, design-oriented model

Logical model
(e.g., property graph)

PG node labs, w/ multi-lab, w/ edge labs

REL, with parent-child tables

PG node labs, node multi-lab, edge labs

Logical model
(e.g., property graph)

RDF

PG node labs, node multi-lab, w/ edge labs

Document model
(nested entities)

Physical model
(e.g., Oracle, DB2, Postgres, MySQL, SQLite, … Neo4J, OrientDB, Neptune, AllegroGraph, Triple Stores…)

PG node labs, w/ multi-lab, w/ edge labs

RDF

PG node labs, node multi-lab, edge labs

Document model
(nested entities)
Language Heterogeneity in the KG world

Conceptual, design-oriented model

No languages at this level

Logical model
(e.g., property graph)

- RPQs, 2RPQs, UC2RPQs, …
- RDF/SPARQL
- PGQL
- SQL
- Logic-based languages (e.g., Datalog)
- …

Physical model
(e.g., Oracle, DB2, Postgres, MySQL, SQLite, … Neo4J, OrientDB, Neptune, AllegroGraph, …)

- Almost one language per system
- and systems with no dedicated languages
or, Model-Independent Schema and Data Translation (by Atzeni, Bellomarini et al.) reloaded for KGs

Meta-model

It contains the foundational **meta-constructs**

- **MM_Entity**: abstract domain entity
- **MM_Link**: connection between entities
- **MM_Property**: their properties
Our Proposal: A Meta-level Approach

Supermodel

The most **comprehensive** set of general constructs that represent graph-like elements at a conceptual level

- SM_Node
- SM_Edge
- SM_Attribute
- SM_Type
- …
Super-Model

- specialises MM_Entity
- specialises MM_Link
- specialises MM_Property
Visual Design and Rendering Function

<table>
<thead>
<tr>
<th>Super-construct</th>
<th>Attributes</th>
<th>Grapheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>SM_NODE</td>
<td>isTENsional = false,</td>
<td>name</td>
</tr>
<tr>
<td></td>
<td>name from SM_TYPE</td>
<td></td>
</tr>
<tr>
<td>SM_NODE</td>
<td>isTENsional = true,</td>
<td>name</td>
</tr>
<tr>
<td></td>
<td>name from SM_TYPE</td>
<td></td>
</tr>
<tr>
<td>SM_EDGE</td>
<td>isTENsional = false, name from</td>
<td>name</td>
</tr>
<tr>
<td></td>
<td>SM_TYPE, c\textsubscript{max},</td>
<td></td>
</tr>
<tr>
<td></td>
<td>c\textsubscript{min}, c\textsubscript{max} from isOpt and isFun</td>
<td></td>
</tr>
<tr>
<td>SM_EDGE</td>
<td>isTENsional = true, name from</td>
<td>name</td>
</tr>
<tr>
<td></td>
<td>SM_TYPE, c\textsubscript{min},</td>
<td></td>
</tr>
<tr>
<td></td>
<td>c\textsubscript{min}, c\textsubscript{max} from isOpt and isFun</td>
<td></td>
</tr>
</tbody>
</table>

A portion of a KG from the Bank of Italy

**Design Patterns**, guidelines, best practices

*Understand and communicate*
The Tooling: **Clickable Graph Schema**

![Clickable Graph Schema](image-url)
An instance of the super-model

- represents a specific domain
- a model-independent representation of a schema

Rendering function

- maps each super-construct into a grapheme
- allows multiple representations
Physical persons and legal persons can have their place for residence. They can hold stakes in a company shareholding capital. Moreover, both have a fiscal code, which uniquely identifies them in the national system.

I will introduce a **SM_Generalization**, where a Person generalizes and collects the common features of PhysicalPerson and LegalPerson. As every person can be in exactly one of those two categories, the generalization will be **disjoint** and **total**.
A Translation Mechanism

1. **visual design**
   - Visual Graph Schema Language (VGSL)

2. **serialization**
   - Graph Schema
   - Graph Schema Language (GSL)

3. **loading into super-model**
   - Super-schema

4. **candidate mapping selection**
   - Metalog
   - M
   - REL
   - PG
   - RDF

5. **implementation strategy selection**
   - $\mathcal{M}(M)$
   - OrientDB
   - Neo4J
   - ... (other graph databases)

6. **deploy & enforce**
   - target system
   - model
   - schema
The designer writes a Metalog program $\Sigma$ based on which the visual graph schema is generated.

The intensional component involves creating a target system $\mathcal{M}(M)$ from the super-model $\mathcal{M}(M)^{-1}$, applying instances to it, and then generating views for input/output.

Mathematically, the business rule can be represented as:

$$(x : \text{Business}) \rightarrow \exists c (x)[c : \text{CONTROLS}](x)$$

$$(x : \text{Business})[c : \text{CONTROLS}](x)[z : \text{Business}][\text{OWNS} ; \text{percentage} : w](y : \text{Business}),$$

$$v = \text{sum}(w, (z)), v > 0.5 \rightarrow \exists c (x)[c : \text{CONTROLS}](y)$$
The Metalog Language

Metalog

\[(x : \text{Business}) \rightarrow \exists c \ (x)[c : \text{CONTROLS}] (x)\]

\[(x : \text{Business})[: \text{CONTROLS}] (z : \text{Business})[: \text{OWNS}; \text{percentage} : w] (y : \text{Business}),
    \ v = \text{sum}(w, \langle z \rangle), \ v > 0.5 \rightarrow \exists c \ (x)[c : \text{CONTROLS}] (y)\]

Datalog+/- (Warded Datalog+/-)

\[\text{Business}(x) \rightarrow \text{Controls}(x, x)\]

\[\text{Controls}(x, y), \text{Owns}(y, z, w), \ v = \text{sum}(w, \langle y \rangle), \ v > 0.5 \rightarrow \text{Controls}(x, z)\]

Translation

Applies to

Vadalog System

Gottlob, Pieris 2015
Beyond SPARQL under OWL 2 QL
Entailment Regime: Rules to the Rescue, IJCAI 2015

Bellomarini, Gottlob, Sallinger
Datalog-based Reasoning for KGs, VLDB 2017

graph-based data dictionary
1. how to design, understand and communicate a KG of a complex domain?
2. how to make the deployed KG schema mirror the conceptual specification?
3. how to express and apply complex business behaviour in the KG?

- A model-independent design methodology
  - a design methodology and a set of tools
  - model independence and model awareness
  - model-drivenness
  - conceptual ergonomics

- Seamless connection between
  - design methodology and visual representation
  - conceptual schema, logical schemas, and physical schemas

- Model-independent description of business behaviours
  - declarative, tractable, semantically sound, plain, expressive, … language
  - system-independent execution
  - at conceptual level

with heterogeneity
Model-Independent Design of Knowledge Graphs