


Property Graph Schema Working Group

Juan Sequeda (data.world) on behalf of the entire
Working Group

Background

- LDBC Graph Query Language Taskforce
- Industry and Academic unite: G-CORE (SIGMOD2018)
- GQL Manifesto
- Property Graphs need a schema (Position Statement at W3C Graph Data Workshop)



The smartest person in the
room is never as smart as
all the people in the room.

John C. Maxwell

(In)formal Property Graph Schema Working Group

- Started Q4 2018
- Informal group, following the Existing Languages Group
- Imagine if you get a group of smart folks from industry and academia to work together and come up with an ideal schema for property graphs?

- Goal: Provide a **recommendation** to the GQL standards committee on what a core property graph schema should be: GS-CORE

- **We are NOT making a standard.**

What have we done up to now?

- Q4 2018 - Q1 2019
 - Academic Survey
 - Industry Survey
 - Use Case and Requirements
- Position statement presented at the W3C Graph Workshop March 2019
- F2F meeting in Berlin March 2019
 - Came up with the first list of requirements that should be considered to be part of gs-core
 - Discussed the UC&R
- Q2 2019
 - Discussion on defining the property graph model
 - Multiple graphs

F2F Meeting Yesterday (July 4)

Came to an agreement of the list of the type of features that should be in gs-core!

GS-CORE:

- Defining Nodes and Edges with Labels and Types, and Properties
- Mandatory Properties
- Local Key Constraints for Vertices
- Local Key Constraints for Edges
- Inheritance
- Property Types

The Property Graph Model

Overlapping graphs within a database

Provisional formalisation of the Berlin model

Amsterdam Face-to-Face Meeting

The Property Graph Model

Overlapping graphs within a database

Provisional formalisation of the Berlin model

Amsterdam Face-to-Face Meeting

Agreed Basic features of the Property Graph model

- ▶ Discussion on Basecamp 3
- ▶ Main decisions:
 - ▶ Properties are identified by their name.
 - ▶ Elements can have zero or more labels.
 - ▶ No meta-properties.
 - ▶ No edges between edges or properties.
 - ▶ Undirected edges.

Postulated sets for Property Graphs

- ▶ A countably infinite set \mathcal{L} of element labels.
- ▶ A countably infinite set \mathcal{K} of property names.
- ▶ A countably infinite set \mathcal{V} of property values.
- ▶ **Note:** We explicitly allow that the sets \mathcal{L} , \mathcal{K} and \mathcal{V} overlap.
- ▶ A countably infinite set of abstract identifiers \mathcal{I} **partitioned** into vertex identifiers \mathcal{I}_v and edge identifiers \mathcal{I}_e .

Property graphs

Definition (Property graph)

A *property graph* is defined as $(V, E, \lambda, \rho, \pi)$ where

- ▶ $V \subseteq \mathcal{I}_v$ is a set of vertex identifiers,
- ▶ $E \subseteq \mathcal{I}_e$ is a set of edge identifiers partitioned into a set of *directed-edge* identifiers E^d and *undirected-edge* identifiers E^u ,
- ▶ $\lambda : (V \cup E) \rightarrow 2^{\mathcal{L}}$ is a total function that maps all vertex identifiers and edge identifiers to a finite set of labels,
- ▶ ρ is the union of ρ^d and ρ^u where
 - ▶ $\rho^d : E^d \rightarrow (V \times V)$ is a total function that maps directed-edge identifiers to a pair of vertex identifiers and
 - ▶ $\rho^u : E^u \rightarrow \left(\binom{V}{2} \cup \binom{V}{1}\right)$ is a total function that maps undirected-edge identifiers to a set of one or two vertex identifiers,
- ▶ $\pi : (V \cup E) \rightarrow (\mathcal{K} \rightarrow \mathcal{V})$, a total function that maps edge/vertex identifiers to a finite partial function that maps property names to property values.

The Property Graph Model

Overlapping graphs within a database

Provisional formalisation of the Berlin model

Amsterdam Face-to-Face Meeting

The discussion concerning overlap

- ▶ **Central question:**

- ▶ *Can element identifiers appear in multiple graphs within the same database?*

- ▶ **Subquestions:**

- ▶ *What does this mean for the user?*
 - ▶ The same business entity? A derived business entity?
- ▶ *How does this come about?*
 - ▶ Views? Updates? Transformations? Versions?
- ▶ *Which consistency rules?*
 - ▶ Edge identifiers must in all graphs have same associated head and tail?
 - ▶ Element identifiers must in all graphs have the same properties? Or only be consistent?

Unresolved

The Property Graph Model

Overlapping graphs within a database

Provisional formalisation of the Berlin model

Amsterdam Face-to-Face Meeting

Additional postulated sets for graph schemas

- ▶ A set \mathcal{T}_p of property type names.
- ▶ For each property type name $\tau \in \mathcal{T}_p$ there is a set $[[\tau]] \subseteq \mathcal{V}$ containing all primitive values of type τ .

Core graph schemas

Definition (Core schema)

We define a *core schema* as (L_S, ρ_S, π_S) where

- ▶ $L_S \subseteq \mathcal{L}$ is a finite set of labels that is partitioned into L_S^V , the set of *vertex labels*, L_S^d , the set of *directed-edge labels* and L_S^u , the set of *undirected-edge labels*;
- ▶ ρ_S is the union of ρ_S^d and ρ_S^u where
 - ▶ $\rho_S^d : L_S^d \rightarrow (L_S^V \times L_S^V)$ is a total function that maps all directed-edge labels to an ordered pair of vertex labels, and
 - ▶ $\rho_S^u : L_S^u \rightarrow \left(\binom{L_S^V}{1} \cup \binom{L_S^V}{2} \right)$ is a total function that maps all undirected-edge labels to a set of one or two vertex labels,
- ▶ $\pi_S : L_S \rightarrow (\mathcal{K} \rightarrow \mathcal{T}_p)$ is a total function that maps element labels to a finite partial function that maps property names to property types.

The semantics of core graph schemas

Introduction

- ▶ We define the semantics by defining when a property graph *satisfies* a core schema.
- ▶ We distinguish a weaker and a stronger notion of satisfaction that correspond to the schema being *open* and *closed*:
 - ▶ **Weak satisfaction:** All elements and properties must satisfy the typing requirements that follow from their labels and names, but it is allowed that
 - (1) some vertices and edges are without labels,
 - (2) the graph uses labels that are not mentioned by the schema and
 - (3) some elements have properties that are not justified by any of their labels.
 - ▶ **Strong satisfaction** Here (1), (2) and (3) are not allowed to occur.

The semantics of core graph schemas

Weak satisfaction

Notation: For an ordered pair y we let $y.1$ and $y.2$ denote the first and second component of y .

Definition (Weak satisfaction)

We say that a property graph $G = (V, E, \lambda, \rho, \pi)$ *weakly satisfies* core schema $S = (L_s, \rho_s, \pi_s)$ if all the following hold.

1. For every element $a \in V \cup E$, label $\ell \in \lambda(a)$, pair $(p, w) \in \pi(a)$ and pair $(p, \tau) \in \pi_s(\ell)$, it holds that $w \in \llbracket \tau \rrbracket$.
2. For every edge $e \in E$ and label $\ell \in \lambda(e)$ such that $\ell \in L_s^d$,
 - ▶ $e \in E^d$ and
 - ▶ $\rho_s^d(\ell).1 \in \lambda(\rho^d(e).1)$ and $\rho_s^d(\ell).2 \in \lambda(\rho^d(e).2)$.
3. For every edge $e \in E$ and label $\ell \in \lambda(e)$ such that $\ell \in L_s^u$,
 - ▶ $e \in E^u$ and
 - ▶ $\rho_s^u(\ell) \subseteq \lambda(v)$ for all $v \in \rho^u(e)$.

The semantics of core graph schemas

Strong satisfaction

Definition (Strong satisfaction)

We say that a property graph $G = (V, E, \lambda, \rho, \pi)$ *strongly satisfies* a core schema $S = (L_S, \rho_S, \pi_S)$ if all the following hold.

1. The graph G weakly satisfies the schema S .
2. For every element $a \in V \cup E$ it holds that $\lambda(a) \neq \emptyset$ and $\lambda(a) \subseteq L_S$.
3. For every element $a \in V \cup E$ and pair $(p, w) \in \pi(a)$ there is a pair $(p, \tau) \in \pi_S(\ell)$ for some label $\ell \in \lambda(a)$.

The Property Graph Model

Overlapping graphs within a database

Provisional formalisation of the Berlin model

Amsterdam Face-to-Face Meeting

Amsterdam Face-to-Face Meeting

- ▶ Current proposal for formal model was presented.
 - ▶ Plus list of candidate schema constraints
- ▶ Votes were taken on which types of constraints should be considered first.
- ▶ It was suggested that schemas should have a graphical representation.
- ▶ Neo's alternative type-based proposal was presented.
 - ▶ It addresses a lack in expressive power concerning unlabelled elements and typing based on properties rather than labels.
- ▶ The current proposal will be extended by
 - ▶ adding *type names* and allowing them in all places where labels are expected
 - ▶ but without explicitly labelling elements with them.

Next steps

- ▶ Producing a summary for coming WG3 meeting on Sept. 23.
- ▶ Three groups are to be formed for the following topics:
 1. Adding type names and defining a graphical representation for schemas.
 2. Mandatory property constraints and key constraints.
 3. Inheritance.

Note: *property types* are assumed to touch all groups.

Thank You