PG-FD: Mapping Graph FD to PG-Schema

Maude Manouvrier   Khalid Belhajjame

PSL Research University, Université Paris-Dauphine, LAMSADE UMR CNRS 7243, France

ADBIS 2024

Outline

1 Introduction

2 Preliminaries

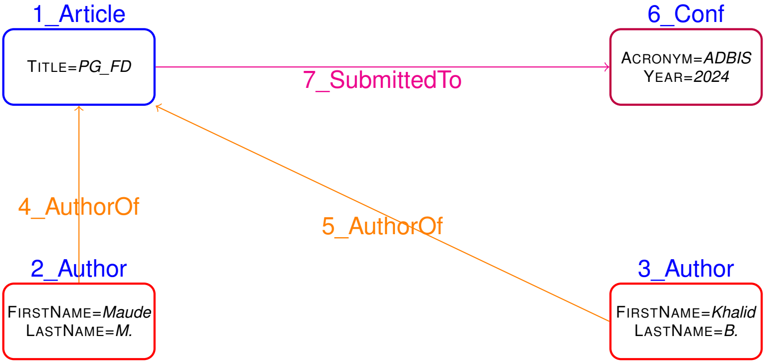
3 Related work

4 Mapping Graph Dependencies to PG-Schema

5 Conclusion

A Property Graph Example

;



$V = \{1, 2, 3, 6\}$ $\quad E = \{4, 5, 7\}$,
$\lambda(1) = Article$, $\lambda(2) = \lambda(3) = Author$, $\lambda(4) = \lambda(5) = AuthorOf$ ...
$\eta(4) = (1, 2)$, $\eta(5) = (1, 3)$ and $\eta(7) = (1, 6)$
$v(1, Title) = PG\_FD$, $v(6, Acronym) = ADBIS$, $v(6, Year) = 2024$,
...

SQL/PGQ, GQL, PG-Keys and PG-Schema

- Property Graph Queries (SQL/PGQ[1]) added to the SQL standard SQL:2023 or ISO/IEC 9075:2023
- GQL[2] (Graph Query Language) officially published as an ISO/IEC standard in April 2024[3]
- Linked Data Benchmark Council (LDBC) Property Graph Schema Working Group defining
  - PG-Keys : Keys for Property Graph (SIGMOD 2021 [Angles et al., 2021])
  - PG-Schema : Schema for Property Graph (SIGMOD 2023 [Angles et al., 2023])

[1] https://www.iso.org/standard/79473.html
[2] https://www.gqlstandards.org/home
[3] https://www.iso.org/standard/76120.html

Integrity constraints and dependencies for graph

Several approaches particularly interested in integrity constraints and dependencies for graphs:

Acronym	Definition	References
gFD	Graph-tailored functional dependency	[Skavantzos et al., 2023]
GED	Graph Entity Dependency	[Fan et al., 2017, 2019]
GD	Graph Dependency	[Zheng et al., 2023]
...	...	...

Our objective

Mapping for translating a notable subset of the identified constraint types into the future property graph schema standard PG-Schema

Outline

1 Introduction

2 Preliminaries

3 Related work

4 Mapping Graph Dependencies to PG-Schema

5 Conclusion

Functional Dependency (FD)

Relational Functional Dependency (RFD): $X \rightarrow Y$

- defined on a relational schema R specifying the "scope" of the FD
- with attribute sets X and Y
- meaning that: $\forall$ instance r of R and $\forall$ t_1 and $t_2 \in r$

$$t_1.X = t_2.X \Longrightarrow t_1.Y = t_2.Y$$

Example : *address* $\rightarrow$ *region*

Graph Dependency

Functional dependency:

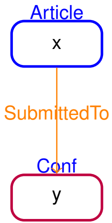
- Fundamental for many areas of data management, such as integrity maintenance, query optimization, database design, and data cleaning
- Scope of RFD: intrinsically delimited by the relation to which the attributes participating in the FD belong

Graph dependency: need for information about the scope of the dependency delimiting the sub-graphs in which the dependency is valid

Graph Pattern:

A directed graph $Q[\bar{x}] = (V_Q, E_Q, L_Q)$

- V_Q (E_Q, respectively): a finite set of pattern nodes (edges, respectively);
- L_Q: a function that assigns a label to each node $u \in V_Q$ (edge $e \in E_Q$, respectively); and
- $\bar{x}$: a list of distinct variables, each denoting a node in V_Q.



A Graph Pattern Example

Outline

1 Introduction

2 Preliminaries

3 Related work

4 Mapping Graph Dependencies to PG-Schema

5 Conclusion

Graph Dependency related work

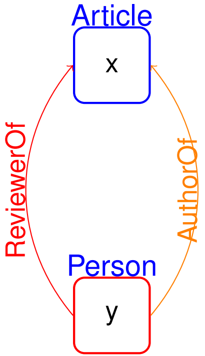
- **Pattern-based graph dependencies**: *Graph Entity Dependencies* (GED) of [Fan et al., 2017, 2019]
  - Subsuming Functional Dependencies (FD) and Conditional Functional Dependencies (CFD) of [Fan et al., 2008]
  - Widely extended to temporal or probabilitic dependency for example
- **Existence-based functional dependency**: limiting the graph objects on which the graph dependencies hold by existence conditions
  - *graph-tailored Functional Dependency* (*gFD*), defined in [Skavantzos et al., 2023]
  - Graph Dependency (GD) of [Zheng et al., 2023]

Graph Entity Dependencies (GED) [Fan et al., 2017, 2019]

Graph Entity Dependencies (GED): $Q[\bar{x}](X \to Y)$

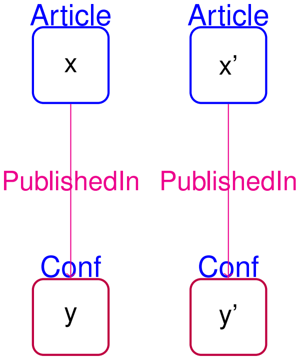
- $Q[\bar{x}]$: a graph pattern
- $X \to Y$: a FD to be applied to entities identified by Q
- X and Y: two (possibly empty) sets of literals of $\bar{x}$
  - a constant literal $x.A = c$ with $A \in \mathcal{K}$ an attribute of $x \in \bar{x}$ and $c \in \mathcal{N}$ a constant, meaning $v(x, A) = c$;
  - a boolean constant *False*, meaning that pattern $Q[\bar{x}]$ is "illegal";
  - a variable literal $x.A = y.B$ with $A, B \in \mathcal{K}$, attributes of the respective entities $x, y \in \bar{x}$, meaning $v(x, A) = v(y, B)$;
  - an id literal $id(x) = id(y)$, $x, y \in \bar{x}$ and $id()$ denoting the vertex or edge identities, and pattern $Q[\bar{x}]$ being composed of two similar sub-patterns.

An "illegal" pattern GED



$$\varphi_1 = \mathcal{Q}_1[x, y](\emptyset \rightarrow \mathit{False})$$

A vertex identity GED



$\mathcal{Q}_2[x, x', y, y']$

$\varphi_2 = \mathcal{Q}_2[x, x', y, y'](X_2 \rightarrow Y_2)$
with $X_2 = \{x.title = x'.title,\ y.id = y'.id\}$
and $Y_2 = \{x.id = x'.id\}$

graph-tailored Functional Dependency (gFD)

[Skavantzos et al., 2023]

graph-tailored Functional Dependency (gFD): $L : P : X \rightarrow Y$

- $L \subseteq \mathcal{L}$ with $\mathcal{L}$ a finite set of labels
- $X, Y \subseteq P \subseteq \mathcal{K}$ with $\mathcal{K}$ a set of property keys

$L : P : X \rightarrow Y$ is satisfied iff

- there are no vertices $v_1, v_2 \in V$ such that $v_1 \neq v_2$,
- for all $A \in P$, $v(v_1, A)$ and $v(v_2, A)$ are defined,
- for all $A \in X$, $v(v_1, A) = v(v_2, A)$
- and for some $A \in Y$, $v(v_1, A) \neq v(v_2, A)$.

Example:

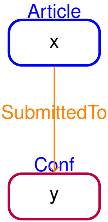
$$Conf : \{Acronym, ConfName\} : ConfName \rightarrow Acronym$$

Graph Dependency (GD) [Zheng et al., 2023]

GD: $Q[\bar{x}](X \rightarrow Y)$

- $Q[\bar{x}]$: a graph pattern
- $X \rightarrow Y$: a FD to be applied to entities identified by Q with
  - X: an existing condition, such as $\exists o \in V$ or $\exists o \in E$, associated with predicates such as $o.label = \ell$ with $\ell \in \mathcal{L}$, or $o.A = c$ with c a constant and $A \in \mathcal{K}$, or $o.A = x.A$ with $x \in \bar{x}$
  - Y: an ASCII art notation of Cypher [Francis et al., 2018] to support connection between nodes defined in X and nodes in $\bar{x}$, e.g. `(x)->(y)` or `(x)-[e]->(y)`

A GD example



$Q[\bar{x}]\ X \rightarrow Y$,
with X: $\exists$ edge $e \in E$, $\lambda(e) = \mathit{SubmittedTo}$
and Y: `(x)-[e]->(y)`

PG-Schema [Angles et al., 2023]

Constraint in PG-Schema

FOR $p(x)$ <qualifier> $q(x, \bar{y})$

- $p(x)$ and $q(x, \bar{y})$: scope and the descriptor
- <qualifier>: $\forall$ output x of $p(x)$
  - MANDATORY: at least 1 tuple $\bar{y}$ that satisfies $q(x, \bar{y})$
  - SINGLETON: at most 1 tuple $\bar{y}$ that satisfies $q(x, \bar{y})$
  - EXCLUSIVE: no $\bar{y}$ should be shared by 2 different values of x
  - IDENTIFIER $\equiv$ EXCLUSIVE MANDATORY SINGLETON
  - COUNT LB..UB OF: $| \, q(x, \bar{y}) \, | \in [LB, UB]$
    COUNT 0 OF: $q(x, \bar{y}) = \emptyset$

Example:

```
FOR x:Article MANDATORY e,y
 WITHIN (x)-[e:AuthorOf]->(y:Author)
```

Outline

1 Introduction

2 Preliminaries

3 Related work

4 Mapping Graph Dependencies to PG-Schema

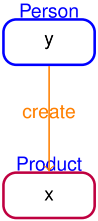
5 Conclusion

Rules to translate GED into PG-Schema

1. When X and Y consist of constant literals:
   `FOR (x:L) WHERE X MANDATORY Y WITHIN Q`
2. When X is $\emptyset$ and Y consists of variable literals:
   `FOR (x:L) MANDATORY Y WITHIN Q`
3. When X consists of variable literals and Y consists of id literals:
   `FOR (x:L) IDENTIFIER left side of variable literals X WITHIN Q'`
   with `Q'` a sub-pattern of `Q`
4. When X is $\emptyset$ and Y is *False*: `FOR (x:L) COUNT 0 OF Q`

When X and Y consist of constant literals

GED example from [Fan et al., 2017, 2019]:



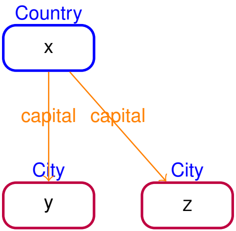
$Q[x, y]\ X \to Y$
with $X = \{x.type = "videogame"\}$
and $Y = \{y.type = "programmer"\}$

Translation into PG-Schema:

```
FOR (x:Product) WHERE x.type = "video game"
MANDATORY y.type = "programmer"
WITHIN (y:Person)-[:create]->(x)
```

When X is $\emptyset$ and Y consists of variable literals

GED example from [Fan et al., 2017, 2019]:

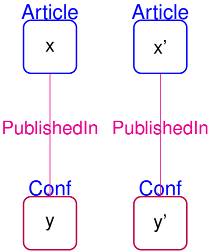


$$Q[x, y, z]\ X \rightarrow Y$$
with $X = \emptyset$ and $Y = \{y.name = z.name\}$

Translation into PG-Schema:

```
FOR (x:Country) MANDATORY y.name = z.name
WITHIN (y:city)<-[:capital]-(x)-[:capital]->(z:city)
```

When X consists of variable literals and Y consists of id literals



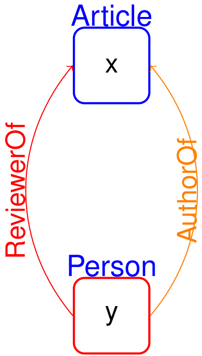
$\mathcal{Q}_2[x, x', y, y']$

$\varphi_2 = \mathcal{Q}_2[x, x', y, y'](X_2 \rightarrow Y_2)$
with $X_2 = \{x.title = x'.title,\ y.id = y'.id\}$
and $Y_2 = \{x.id = x'.id\}$

Translation into PG-Schema:

```
FOR (x:Article) IDENTIFIER x.title, y.id
WITHIN (x)-[:PublishedIn]->(y:Conf)
```

When X is $\emptyset$ and Y is *False*



$$\varphi_1 = \mathcal{Q}_1[x, y](\emptyset \rightarrow \mathit{False})$$

Translation into PG-Schema:

```
FOR (x:Article) COUNT 0 OF
(y:Person)-[:AuthorOf]->(x)<-[:ReviewerOf]-(y:Person)
```

Rule to translate gFD into PG-Schema

gFD: $\{L_1, ..., L_m\} : \{P_1, ..., P_n\} : \{X_1, ..., X_k\} \rightarrow \{Y_1, ..., Y_j\}$
where $\{L_1, ..., L_m\} \subseteq \mathcal{L}$ and
$\{X_1, ..., X_k\}, \{Y_1, ..., Y_j\} \subseteq \{P_1, ..., P_n\} \subseteq \mathcal{K}$

Translation into PG-Schema:

```
FOR x.Y1, ..., x.Yk WITHIN x:L1 ...  :Lm
WHERE x.X1 IS NOT NULL AND ...  AND x.Xk IS NOT NULL
EXCLUSIVE MANDATORY x.X1, ..., x.Xj
```

Example: $Conf : \{Acronym, ConfName\} : ConfName \rightarrow Acronym$

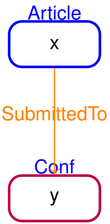
Translation into PG-Schema:

```
FOR x.Acronym WITHIN x:Conf
WHERE x.ConfName IS NOT NULL AND x.Acronym IS NOT NULL
EXCLUSIVE MANDATORY x.ConfName
```

Rules to translate GD into PG-Schema

1. When X is $\exists e \in E, \lambda(e) = L$ and Y is `(x)-[e]->(y)`:
   ```
   FOR (x:x.label) MANDATORY e,y
   WITHIN (x)-[e:L]->(y:y.label)
   ```
2. When X is $\exists x \in V, \lambda(x) = L$ and Y is `(x)->(y)`:
   ```
   FOR (x:L) MANDATORY e,y
   WITHIN (x)-[e]->(y:y.label)
   ```

Translation of a GD example into PG-Schema



$Q[\bar{x}]\ X \to Y$,
with X is $\exists$ edge $e \in E$, $\lambda(e) = SubmittedTo$
and Y is `(x)-[e]->(y)`

Translation into PG-Schema:

```
FOR (x:Article) MANDATORY e,y
WITHIN (x)-[e:SubmittedTo]->(y:Conf)
```

Rules to translate RFD to PG-Schema

RFD: $X \rightarrow Y$, defined on a relation schema $R(\mathcal{K}_R)$, with $\mathcal{K}_R$ the attribute set containing sets $X = \{X_1, X_2, .., X_n\}$ and $Y = \{Y_1, Y_2, .., Y_m\}$

Translation into PG-Schema:

```
FOR x.Y1, ...  Y.Ym WITHIN (x:R)
EXCLUSIVE MANDATORY x.X1, ..., X.Xn
```

Example: *address* $\rightarrow$ *region*

Translation into PG-Schema:

```
FOR x.region WITHIN (x:R)
EXCLUSIVE MANDATORY x.address
```

No loss of information

- $\mathcal{M}$: a graph data model encompassing various existing proposals ranging from relational to RDF and property graph models
- *dep*: a data dependency expressed within $\mathcal{M}$
- PG-Schema considers graph data expressed using property graphs

An instance $\mathcal{I}$ of a data model $\mathcal{M}$ can be translated to a property-graph instance $\mathcal{I}^{\mathcal{PG}}$ without any loss of information

Computability, semantics preservation and information preservation

properties of our mapping:

- **Computability**: mapping rules can be implemented as an algorithm – see `https://github.com/MaudeManouvrier/PG-FD` ;
- **Information preserving**: the PG-Schema dependency obtained through the translation preserves the entire semantics of the original dependency ;
- **Semantics preservation**:

$$\forall \mathcal{I} \in \textit{instances}(\mathcal{M})\ \mathcal{I} \models \textit{dep} \implies \mathcal{I}^{PG} \models \textit{dep}^{PS} \quad (1)$$
$$\mathcal{I} \not\models \textit{dep} \implies \mathcal{I}^{PG} \not\models \textit{dep}^{PS} \quad (2)$$

with *dep* a dependency expressed in a model $\mathcal{M}$ into a dependency $\textit{dep}^{PS}$ compliant with PG-Schema

Outline

1 Introduction

2 Preliminaries

3 Related work

4 Mapping Graph Dependencies to PG-Schema

5 Conclusion

Conclusion and Future Work

- Definition of mapping rules allowing to translate graph dependencies, focusing on prominent ones, into the PG-Schema [Angles et al., 2023]
- Presentation of proof of the soundless of the proposed translation and a proof-of-concept implementation prototype *PG-FD*[4]
- Future Work: extend our solution to cater for other forms of graph-based dependencies and experiment our prototype with graph dependencies discovered from DBPedia or YAGO.

[4] https://github.com/MaudeManouvrier/PG-FD

References (1/2)

- Angles, R., Bonifati, A., Dumbrava, S.et al. (2021)
  PG-Keys: Keys for property graphs
  SIGMOD https://dl.acm.org/doi/pdf/10.1145/3448016.3457561

- Angles, R., Bonifati, A., Dumbrava, et al. (2023)
  PG-Schema: Schemas for property graphs
  ACM Management of Data https://dl.acm.org/doi/pdf/10.1145/3589778

- Fan, W., Geerts, F., Jia, X. et al. (2008)
  Conditional functional dependencies for capturing data inconsistencies
  TODS https://core.ac.uk/download/pdf/28962894.pdf

- Fan, W., Lu, P. (2017, 2019)
  Dependencies for graphs
  PODS (2017) and TODS (2019)
  https://www.pure.ed.ac.uk/ws/portalfiles/portal/44159778/pods17.pdf

References (2/2)

- Francis, N. Green, A., Guagliardo, P. et al. (2018)
  Cypher: An evolving query language for property graphs
  SIGMOD https://www.pure.ed.ac.uk/ws/portalfiles/portal/56321692/cypher_sigmod18_crc_1.pdf
- Skavantzos, P., Zhao, K., Link, S. (2023)
  Uniqueness constraints on property graphs
  CAiSE https://researchspace.auckland.ac.nz/bitstream/handle/2292/61545/eUCs-revised.pdf?sequence=1
- Skavantzos, P., Link, S. (2023)
  Normalizing Property Graphs
  VLDB Endowment https://www.vldb.org/pvldb/vol16/p3031-link.pdf
- Zheng, X., Dasgupta, S., Gupta, A. (2023)
  P2KG: declarative construction and quality evaluation of knowledge graph from polystores
  ADBIS https://escholarship.org/content/qt30h5c8jj/qt30h5c8jj.pdf