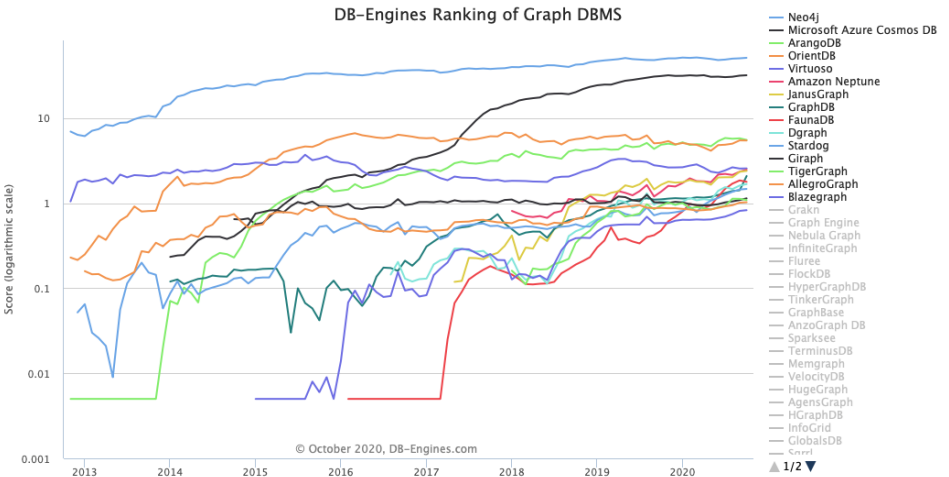
Neo4j Keys

Entity Integrity on Property Graphs

Sebastian Link
The University of Auckland, New Zealand

ER 2020
Vienna, Austria

Popularity of Neo4j



https://db-engines.com/en/ranking_trend/graph+dbms

Popularity of Keys

- Relational data:
  - Claudio L. Lucchesi, Sylvia L. Osborn: Candidate Keys for Relations. J. Comput. Syst. Sci. 17(2): 270-279 (1978)
- Incomplete data (Codd keys, Key sets, Embedded Uniques, Possible and strongly possible and certain keys):
  - B. Thalheim: On Semantic Issues of Keys in Relational Databases with Null Values. J. Inf. Process. Cybern. 25(1/2): 11-20 (1989)
  - Z. Wei, U. Leck, S. Link: Discovery of Embedded Uniqueness Constraints. Proc. VLDB Endow. 12(13): 2339-2352 (2019)
  - H. Koehler, U. Leck, S. Link, X. Zhou: Possible and certain keys for SQL. VLDB J. 25(4): 571-596 (2016)
- Uncertain data (probabilistic keys, possibilistic keys):
  - P. Brown, S. Link: Probabilistic Keys. IEEE Trans. Knowl. Data Eng. 29(3): 670-682 (2017)
  - N. Balamuralikrishna, Y. Jiang, H. Koehler, U. Leck, S. Link, H. Prade: Possibilistic keys. Fuzzy Sets Syst. 376: 1-36 (2019)
- XML data:
  - P. Buneman, S.B. Davidson, W. Fan, C.S. Hara, W.C. Tan: Keys for XML. Comput. Networks 39(5): 473-487 (2002)
  - S. Hartmann, S. Link: Efficient reasoning about a robust XML key fragment. ACM Trans. Database Syst. 34(2): 10:1-10:33 (2009)
  - M. Karlinger, M.W. Vincent, M. Schrefl: Keys in XML: Capturing Identification and Uniqueness. WISE 2009: 563-571
- RDF data:
  - G. Lausen: Relational Databases in RDF: Keys and Foreign Keys. SWDB-ODBIS 2007: 43-56
- Ontologies/Description logics:
  - D. Toman, G.E. Weddell: On Keys and Functional Dependencies as First-Class Citizens in Description Logics. J. Autom. Reason. 40(2-3): 117-132 (2008)
- Graph data:
  - W. Fan, Z. Fan, C. Tian, X.L. Dong: Keys for Graphs. Proc. VLDB Endow. 8(12): 1590-1601 (2015)
  - J. Pokorny, M. Valenta, J. Kovacic: Integrity constraints in graph databases. ANT/SEIT 2017: 975-981

Popularity of Keys

- Relational data:
  - Claudio L. Lucchesi, Sylvia L. Osborn: Candidate Keys for Relations. J. Comput. Syst. Sci. 17(2): 270-279 (1978)
- Incomplete data (Codd keys, Key sets, Embedded Uniques, Possible and strongly possible and certain keys):
  - B. Thalheim: On Semantic Issues of Keys in Relational Databases with Null Values. J. Inf. Process. Cybern. 25(1/2): 11-20 (1989)
  - Z. Wei, U. Leck, S. Link: Discovery of Embedded Uniqueness Constraints. Proc. VLDB Endow. 12(13): 2339-2352 (2019)
  - H. Koehler, U. Leck, S. Link, X. Zhou: Possible and certain keys for SQL. VLDB J. 25(4): 571-596 (2016)
- Uncertain data (probabilistic keys, possibilistic keys):
  - P. Brown, S. Link: Probabilistic Keys. IEEE Trans. Knowl. Data Eng. 29(3): 670-682 (2017)
  - N. Balamuralikrishna, Y. Jiang, H. Koehler, U. Leck, S. Link, H. Prade: Possibilistic keys. Fuzzy Sets Syst. 376: 1-36 (2019)
- XML data:
  - P. Buneman, S.B. Davidson, W. Fan, C.S. Hara, W.C. Tan: Keys for XML. Comput. Networks 39(5): 473-487 (2002)
  - S. Hartmann, S. Link: Efficient reasoning about a robust XML key fragment. ACM Trans. Database Syst. 34(2): 10:1-10:33 (2009)
  - M. Karlinger, M.W. Vincent, M. Schrefl: Keys in XML: Capturing Identification and Uniqueness. WISE 2009: 563-571
- RDF data:
  - G. Lausen: Relational Databases in RDF: Keys and Foreign Keys. SWDB-ODBIS 2007: 43-56
- Ontologies/Description logics:
  - D. Toman, G.E. Weddell: On Keys and Functional Dependencies as First-Class Citizens in Description Logics. J. Autom. Reason. 40(2-3): 117-132 (2008)
- Graph data:
  - W. Fan, Z. Fan, C. Tian, X.L. Dong: Keys for Graphs. Proc. VLDB Endow. 8(12): 1590-1601 (2015)
  - J. Pokorny, M. Valenta, J. Kovacic: Integrity constraints in graph databases. ANT/SEIT 2017: 975-981

entity integrity, referential integrity, database design, query processing, etc.

Overview

- Introduce a notion of keys
  - compliant with the one used by Neo4j
  - extended to make the most of the multi-label property graph model of Neo4j
- Use Cases
- Characterize the implication problem
  - axiomatically
  - algorithmically

Overview

- Introduce a notion of keys
  - compliant with the one used by Neo4j
  - extended to make the most of the multi-label property graph model of Neo4j
- Use Cases
- Characterize the implication problem
  - axiomatically
  - algorithmically

Entity Integrity on (Neo4j) Property Graphs

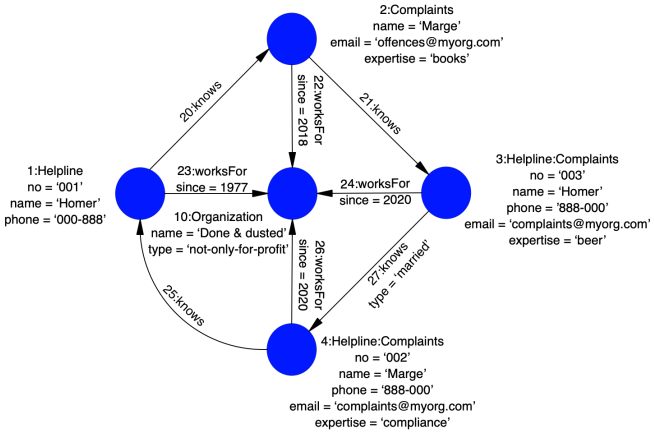
The (Neo4j) Property Graph Model

Disjoint Sets:

- $\mathcal{O}$ set of objects
- $\mathcal{L}$ finite set of labels
- $\mathcal{K}$ set of property attributes
- $\mathcal{N}$ set of values

Property Graph $G = (V, E, \eta, \lambda, \nu)$

- $V \subseteq \mathcal{O}$, eg. $V = \{1, 2, 3, 4, 10\}$
- $E \subseteq \mathcal{O}$, eg. $E = \{20, \ldots, 27\}$
- $\eta : E \to V \times V$, eg. $20 \mapsto (1, 2)$
- $\lambda : V \cup E \to \mathcal{P}(\mathcal{L})$,
  eg. $3 \mapsto \{:\text{Helpline}, :\text{Complaints}\}$
- $\nu : (V \cup E) \times \mathcal{K} \to \mathcal{N}$,
  eg. $(23, \text{since}) \mapsto 1977$



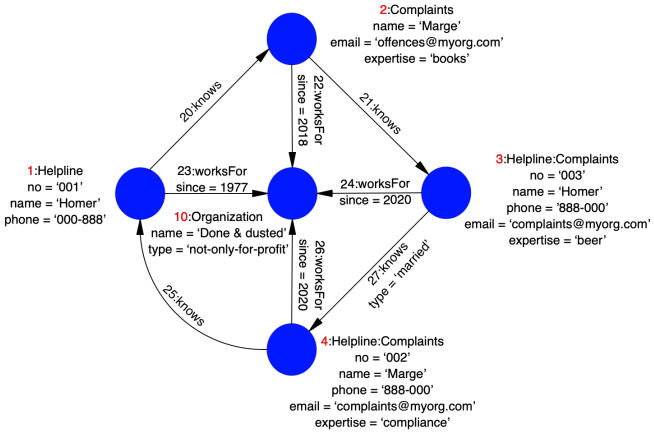
The (Neo4j) Property Graph Model

Disjoint Sets:

- $\mathcal{O}$ set of objects
- $\mathcal{L}$ finite set of labels
- $\mathcal{K}$ set of property attributes
- $\mathcal{N}$ set of values

Property Graph $G = (V, E, \eta, \lambda, \nu)$

- $V \subseteq \mathcal{O}$, eg. $V = \{1, 2, 3, 4, 10\}$
- $E \subseteq \mathcal{O}$, eg. $E = \{20, \ldots, 27\}$
- $\eta : E \to V \times V$, eg. $20 \mapsto (1, 2)$
- $\lambda : V \cup E \to \mathcal{P}(\mathcal{L})$,
  eg. $3 \mapsto \{:\text{Helpline}, :\text{Complaints}\}$
- $\nu : (V \cup E) \times \mathcal{K} \to \mathcal{N}$,
  eg. $(23, \text{since}) \mapsto 1977$



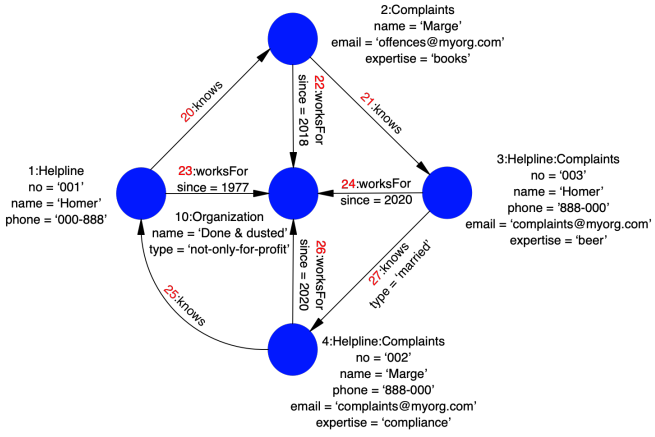
The (Neo4j) Property Graph Model

Disjoint Sets:

- $\mathcal{O}$ set of objects
- $\mathcal{L}$ finite set of labels
- $\mathcal{K}$ set of property attributes
- $\mathcal{N}$ set of values

Property Graph $G = (V, E, \eta, \lambda, \nu)$

- $V \subseteq \mathcal{O}$, eg. $V = \{1, 2, 3, 4, 10\}$
- $E \subseteq \mathcal{O}$, eg. $E = \{20, \ldots, 27\}$
- $\eta : E \to V \times V$, eg. $20 \mapsto (1, 2)$
- $\lambda : V \cup E \to \mathcal{P}(\mathcal{L})$,
  eg. $3 \mapsto \{:\text{Helpline}, :\text{Complaints}\}$
- $\nu : (V \cup E) \times \mathcal{K} \to \mathcal{N}$,
  eg. $(23, \text{since}) \mapsto 1977$



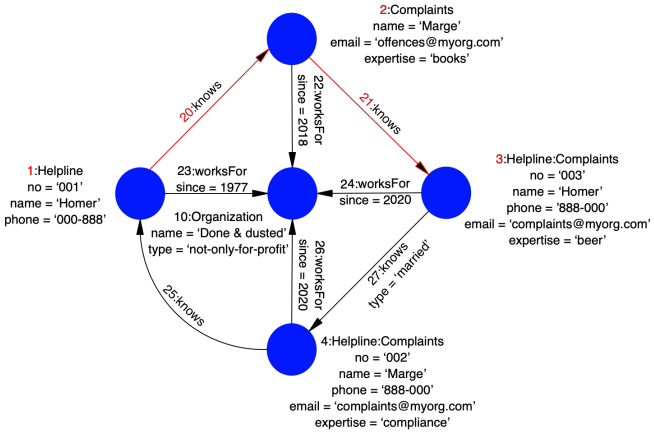
The (Neo4j) Property Graph Model

Disjoint Sets:

- $\mathcal{O}$ set of objects
- $\mathcal{L}$ finite set of labels
- $\mathcal{K}$ set of property attributes
- $\mathcal{N}$ set of values

Property Graph $G = (V, E, \eta, \lambda, \nu)$

- $V \subseteq \mathcal{O}$, eg. $V = \{1, 2, 3, 4, 10\}$
- $E \subseteq \mathcal{O}$, eg. $E = \{20, \ldots, 27\}$
- $\eta : E \to V \times V$, eg. $20 \mapsto (1, 2)$
- $\lambda : V \cup E \to \mathcal{P}(\mathcal{L})$,
  eg. $3 \mapsto \{:\text{Helpline}, :\text{Complaints}\}$
- $\nu : (V \cup E) \times \mathcal{K} \to \mathcal{N}$,
  eg. $(23, \text{since}) \mapsto 1977$



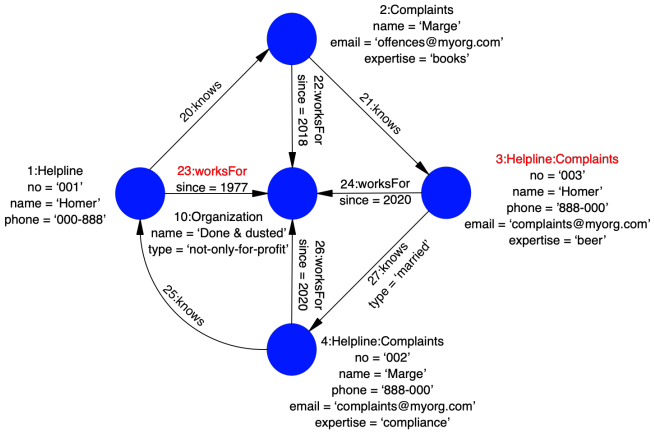
The (Neo4j) Property Graph Model

Disjoint Sets:

- $\mathcal{O}$ set of objects
- $\mathcal{L}$ finite set of labels
- $\mathcal{K}$ set of property attributes
- $\mathcal{N}$ set of values

Property Graph $G = (V, E, \eta, \lambda, \nu)$

- $V \subseteq \mathcal{O}$, eg. $V = \{1, 2, 3, 4, 10\}$
- $E \subseteq \mathcal{O}$, eg. $E = \{20, \ldots, 27\}$
- $\eta : E \to V \times V$, eg. $20 \mapsto (1, 2)$
- $\lambda : V \cup E \to \mathcal{P}(\mathcal{L})$, eg. $3 \mapsto \{:\text{Helpline}, :\text{Complaints}\}$
- $\nu : (V \cup E) \times \mathcal{K} \to \mathcal{N}$, eg. $(23, \text{since}) \mapsto 1977$



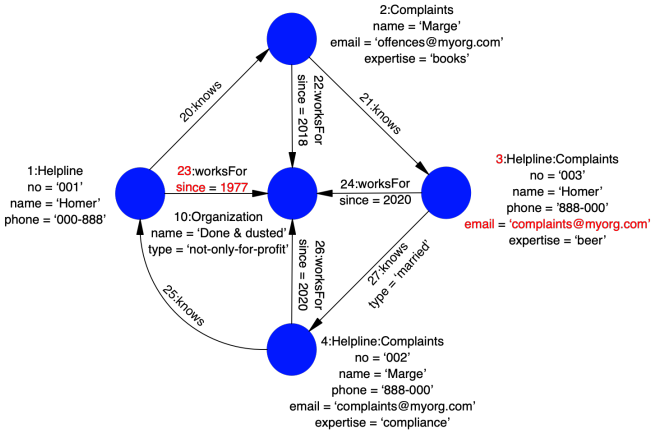
The (Neo4j) Property Graph Model

Disjoint Sets:

- $\mathcal{O}$ set of objects
- $\mathcal{L}$ finite set of labels
- $\mathcal{K}$ set of property attributes
- $\mathcal{N}$ set of values

Property Graph $G = (V, E, \eta, \lambda, \nu)$

- $V \subseteq \mathcal{O}$, eg. $V = \{1, 2, 3, 4, 10\}$
- $E \subseteq \mathcal{O}$, eg. $E = \{20, \ldots, 27\}$
- $\eta : E \to V \times V$, eg. $20 \mapsto (1, 2)$
- $\lambda : V \cup E \to \mathcal{P}(\mathcal{L})$,
  eg. $3 \mapsto \{:\text{Helpline}, :\text{Complaints}\}$
- $\nu : (V \cup E) \times \mathcal{K} \to \mathcal{N}$,
  eg. $(23, \text{since}) \mapsto 1977$



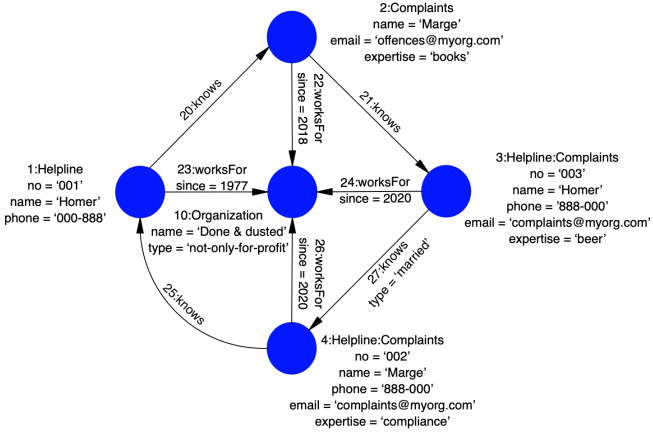
The (Neo4j) Property Graph Model

Disjoint Sets:

- $\mathcal{O}$ set of objects
- $\mathcal{L}$ finite set of labels
- $\mathcal{K}$ set of property attributes
- $\mathcal{N}$ set of values

Property Graph $G = (V, E, \eta, \lambda, \nu)$

- $V \subseteq \mathcal{O}$, eg. $V = \{1, 2, 3, 4, 10\}$
- $E \subseteq \mathcal{O}$, eg. $E = \{20, \ldots, 27\}$
- $\eta : E \to V \times V$, eg. $20 \mapsto (1, 2)$
- $\lambda : V \cup E \to \mathcal{P}(\mathcal{L})$,
  eg. $3 \mapsto \{:\text{Helpline}, :\text{Complaints}\}$
- $\nu : (V \cup E) \times \mathcal{K} \to \mathcal{N}$,
  eg. $(23, \text{since}) \mapsto 1977$



Definition of Neo4j Keys

Syntax

- Given are finite sets $\mathcal{L}$ of labels and $\mathcal{K}$ of property attributes
- A *key* is an expression $\mathfrak{L} : \mathfrak{K}$ where $\mathfrak{L} \subseteq \mathcal{L}$ and $\mathfrak{K} \subseteq \mathcal{K}$
- If $\mathfrak{L}$ is a singleton, $\mathfrak{L} : \mathfrak{K}$ is a *single-labelled* key, and otherwise *multi-labelled*.

Definition of Neo4j Keys

Syntax

- Given are finite sets $\mathcal{L}$ of labels and $\mathcal{K}$ of property attributes
- A *key* is an expression $\mathfrak{L} : \mathfrak{K}$ where $\mathfrak{L} \subseteq \mathcal{L}$ and $\mathfrak{K} \subseteq \mathcal{K}$
- If $\mathfrak{L}$ is a singleton, $\mathfrak{L} : \mathfrak{K}$ is a *single-labelled* key, and otherwise *multi-labelled*.

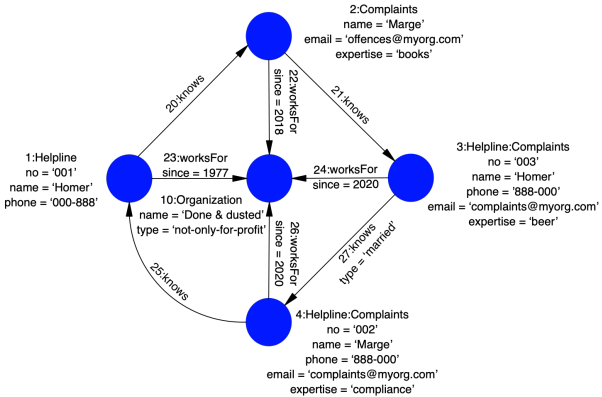
Semantics

- Given is a property graph $G = (V, E, \eta, \lambda, \nu)$ over $\mathcal{O}$, $\mathcal{L}$, $\mathcal{K}$, and $\mathcal{N}$
- G *satisfies* the key $\mathfrak{L} : \mathfrak{K}$ over $\mathcal{L}$ and $\mathcal{K}$, denoted by $\models_G \mathfrak{L} : \mathfrak{K}$, if and only if
  1. (Completeness) $V_{\mathfrak{L}} := \{v \in V \mid \mathfrak{L} \subseteq \lambda(v)\}$ is $\mathfrak{K}$-complete, that is, for all $v \in V_{\mathfrak{L}}$ and for all $A \in \mathfrak{K}$, $\nu(v, A)$ is defined
  2. (Uniqueness) For all $v_1, v_2 \in V_{\mathfrak{L}}$, if $v_1 \neq v_2$, then there is some $A \in \mathfrak{K}$ such that $\nu(v_1, A) \neq \nu(v_2, A)$

Examples

Keys satisfied by property graph

- $Helpline{:}\{no\}$
- $Helpline{:}\{name, phone\}$,
- $Complaints{:}\{name, email\}$
- $\{Helpline, Complaints\}{:}\{no, email\}$



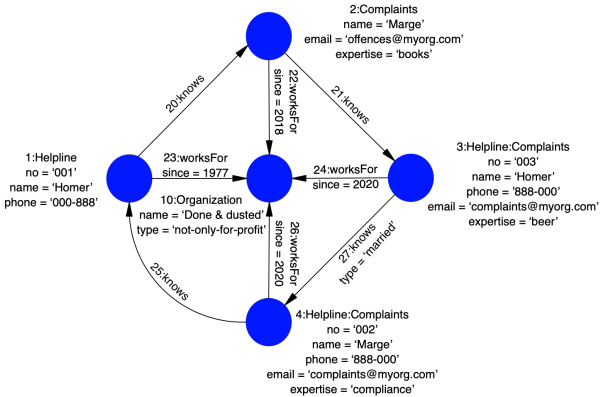
Examples

Keys satisfied by property graph

- $Helpline{:}\{no\}$
- $Helpline{:}\{name,phone\}$,
- $Complaints{:}\{name,email\}$
- $\{Helpline,Complaints\}{:}\{no,email\}$

Keys violated by property graph

- $Helpline{:}\{no,expertise\}$
- $Complaints{:}\{name\}$
- $\{Helpline,Complaints\}{:}\{phone,email\}$



Use Case 1: Updates under the key *Helpline:{name,phone}*

No node created due to violation of key

```
CREATE (h:Helpline {no:004, name:'Bart', expertise:'pranks'})
```

Use Case 1: Updates under the key *Helpline:{name,phone}*

No node created due to violation of key

```
CREATE (h:Helpline {no:004, name:'Bart', expertise:'pranks'})
```

No property removed due to violation of key

```
MATCH (h:Helpline {name:'Homer', phone:'000-888'}) REMOVE h.phone
```

Use Case 1: Updates under the key *Helpline:{name,phone}*

No node created due to violation of key

```
CREATE (h:Helpline {no:004, name:'Bart', expertise:'pranks'})
```

No property removed due to violation of key

```
MATCH (h:Helpline {name:'Homer', phone:'000-888'}) REMOVE h.phone
```

No update of values due to violation of key

```
MATCH (h:Helpline {name:  'Homer', phone:'000-888'})
SET h.phone = '888-000'
RETURN h.name, h.phone
```

Use Case 2: Physical query optimization

```
MATCH (h:Helpline {name:'Homer',phone:'000-888'})
RETURN h.no, h.name
```

- will do a NodeByLabelScan, due to node label *Helpline*
- query would still take long on realistic property graphs

Use Case 2: Physical query optimization

```
MATCH (h:Helpline {name:'Homer',phone:'000-888'})
RETURN h.no, h.name
```

- will do a NodeByLabelScan, due to node label *Helpline*
- query would still take long on realistic property graphs

```
CREATE INDEX index_Helpline FOR (h:Helpline)
ON (h.name, h.phone)
```

- query optimizer will take advantage of the index
- a NodeUniqueIndexSeek search is very efficient

Use Case 3: Logical query optimization

```
MATCH (s:Helpline:Complaints)
RETURN DISTINCT s.no, s.email
```

- Removal of duplicates expensive
- *Helpline*:{*no*} can avoid DISTINCT
- {*Helpline*,*Complaints*}:{*no*,*email*} may use multi-label index

Use Case 3: Logical query optimization

```
MATCH (s:Helpline:Complaints)
RETURN DISTINCT s.no, s.email
```

- Removal of duplicates expensive
- *Helpline*:{*no*} can avoid DISTINCT
- {*Helpline*,*Complaints*}:{*no*,*email*} may use multi-label index

No support of multi-label keys by Neo4j (yet)

Axiomatic Characterization

$$\frac{\mathfrak{L} : \mathfrak{K}}{\mathfrak{L} \cup \mathfrak{L}' : \mathfrak{K}}\ \text{(label-extension)} \qquad \frac{\mathfrak{L} : \mathfrak{K}_1 \quad \mathfrak{L} : \mathfrak{K}_2 \cup \mathfrak{K}_3}{\mathfrak{L} : \mathfrak{K}_1 \cup \mathfrak{K}_2}\ \text{(attribute-extension)}$$

Axiomatic Characterization

$$\frac{\mathfrak{L} : \mathfrak{K}}{\mathfrak{L} \cup \mathfrak{L}' : \mathfrak{K}}\ \text{(label-extension)} \qquad \frac{\mathfrak{L} : \mathfrak{K}_1 \quad \mathfrak{L} : \mathfrak{K}_2 \cup \mathfrak{K}_3}{\mathfrak{L} : \mathfrak{K}_1 \cup \mathfrak{K}_2}\ \text{(attribute-extension)}$$

Theorem

Label-extension and attribute-extension form a sound and complete set of inference rules for the implication of Neo4j keys

Axiomatic Characterization

$$\frac{\mathfrak{L} : \mathfrak{K}}{\mathfrak{L} \cup \mathfrak{L}' : \mathfrak{K}}\ \text{(label-extension)} \qquad \frac{\mathfrak{L} : \mathfrak{K}_1 \quad \mathfrak{L} : \mathfrak{K}_2 \cup \mathfrak{K}_3}{\mathfrak{L} : \mathfrak{K}_1 \cup \mathfrak{K}_2}\ \text{(attribute-extension)}$$

Theorem

Label-extension and attribute-extension form a sound and complete set of inference rules for the implication of Neo4j keys

Example ($\Sigma = \{$*Helpline:*$\{$*no*$\}$, *Helpline:*$\{$*name,phone*$\}$, *Complaints:*$\{$*name,email*$\}\}$)

Helpline:$\{$*name,phone*$\}$ *Complaints:*$\{$*name,email*$\}$

Axiomatic Characterization

$$\frac{\mathfrak{L} : \mathfrak{K}}{\mathfrak{L} \cup \mathfrak{L}' : \mathfrak{K}}\ \text{(label-extension)} \qquad \frac{\mathfrak{L} : \mathfrak{K}_1 \quad \mathfrak{L} : \mathfrak{K}_2 \cup \mathfrak{K}_3}{\mathfrak{L} : \mathfrak{K}_1 \cup \mathfrak{K}_2}\ \text{(attribute-extension)}$$

Theorem

Label-extension and attribute-extension form a sound and complete set of inference rules for the implication of Neo4j keys

Example ($\Sigma = \{$*Helpline*:{*no*}, *Helpline*:{*name*,*phone*}, *Complaints*:{*name*,*email*}$\}$)

$$\frac{\textit{Helpline}:\{\textit{name},\textit{phone}\}}{\{\textit{Helpline},\textit{Complaints}\}:\{\textit{name},\textit{phone}\}} \qquad \textit{Complaints}:\{\textit{name},\textit{email}\}$$

Axiomatic Characterization

$$\frac{\mathfrak{L} : \mathfrak{K}}{\mathfrak{L} \cup \mathfrak{L}' : \mathfrak{K}}\ \text{(label-extension)} \qquad \frac{\mathfrak{L} : \mathfrak{K}_1 \quad \mathfrak{L} : \mathfrak{K}_2 \cup \mathfrak{K}_3}{\mathfrak{L} : \mathfrak{K}_1 \cup \mathfrak{K}_2}\ \text{(attribute-extension)}$$

Theorem

Label-extension and attribute-extension form a sound and complete set of inference rules for the implication of Neo4j keys

Example ($\Sigma = \{\mathit{Helpline}{:}\{\mathit{no}\}, \mathit{Helpline}{:}\{\mathit{name},\mathit{phone}\}, \mathit{Complaints}{:}\{\mathit{name},\mathit{email}\}\}$)

$$\frac{\mathit{Helpline}{:}\{\mathit{name},\mathit{phone}\}}{\{\mathit{Helpline},\mathit{Complaints}\}{:}\{\mathit{name},\mathit{phone}\}} \qquad \frac{\mathit{Complaints}{:}\{\mathit{name},\mathit{email}\}}{\{\mathit{Helpline},\mathit{Complaints}\}{:}\{\mathit{name},\mathit{email}\}}$$

Axiomatic Characterization

$$\frac{\mathfrak{L} : \mathfrak{K}}{\mathfrak{L} \cup \mathfrak{L}' : \mathfrak{K}}\ \text{(label-extension)} \qquad \frac{\mathfrak{L} : \mathfrak{K}_1 \quad \mathfrak{L} : \mathfrak{K}_2 \cup \mathfrak{K}_3}{\mathfrak{L} : \mathfrak{K}_1 \cup \mathfrak{K}_2}\ \text{(attribute-extension)}$$

Theorem

Label-extension and attribute-extension form a sound and complete set of inference rules for the implication of Neo4j keys

Example ($\Sigma = \{\mathit{Helpline}{:}\{\mathit{no}\}, \mathit{Helpline}{:}\{\mathit{name},\mathit{phone}\}, \mathit{Complaints}{:}\{\mathit{name},\mathit{email}\}\}$)

$$\frac{\dfrac{\mathit{Helpline}{:}\{\mathit{name},\mathit{phone}\}}{\{\mathit{Helpline},\mathit{Complaints}\}{:}\{\mathit{name},\mathit{phone}\}} \qquad \dfrac{\mathit{Complaints}{:}\{\mathit{name},\mathit{email}\}}{\{\mathit{Helpline},\mathit{Complaints}\}{:}\{\mathit{name},\mathit{email}\}}}{\{\mathit{Helpline},\mathit{Complaints}\} : \{\mathit{name}, \mathit{phone}, \mathit{email}\}}$$

Deciding Implication

1: **procedure** IMPLY($\Sigma, \mathfrak{L} : \mathfrak{K}$) **return** *TRUE*, if $\Sigma \models \mathfrak{L} : \mathfrak{K}$, and *FALSE*, otherwise
2: `Unique` $\leftarrow$ *FALSE*
3: `Exists` $\leftarrow$ *FALSE*
4: $\mathfrak{K}_{\mathfrak{L},\Sigma} \leftarrow \emptyset$
5: **for all** $\mathfrak{L}' : \mathfrak{K}' \in \Sigma$ **do**
6: **if** $\mathfrak{L}' \subseteq \mathfrak{L}$ **then** ▷ Properties of such keys must exist
7: $\mathfrak{K}_{\mathfrak{L},\Sigma} \leftarrow \mathfrak{K}_{\mathfrak{L},\Sigma} \cup \mathfrak{K}'$ ▷ Include properties of the key
8: **if** (NOT(`Unique`) AND $\mathfrak{K}' \subseteq \mathfrak{K}$) **then**
9: `Unique` $\leftarrow$ *TRUE* ▷ Found input key that makes $\mathfrak{L} : \mathfrak{K}$ unique
10: **if** $\mathfrak{K} \subseteq \mathfrak{K}_{\mathfrak{L},\Sigma}$ **then return** `Exists` $\leftarrow$ *TRUE* ▷ All required properties exist
11: **if** (`Exists` AND `Unique`) **then return** *TRUE* ▷ Properties exist and are unique
12: **else return** *FALSE*

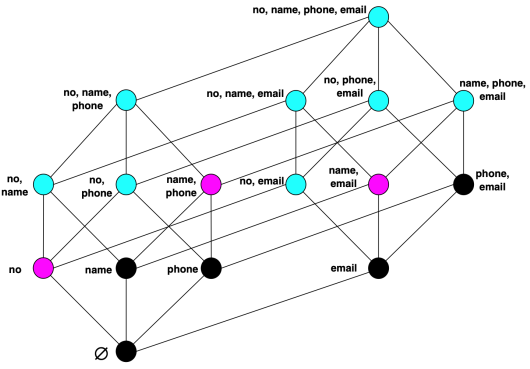
Deciding Implication

1: **procedure** IMPLY($\Sigma, \mathfrak{L} : \mathfrak{K}$) **return** *TRUE*, if $\Sigma \models \mathfrak{L} : \mathfrak{K}$, and *FALSE*, otherwise
2: `Unique` $\leftarrow$ *FALSE*
3: `Exists` $\leftarrow$ *FALSE*
4: $\mathfrak{K}_{\mathfrak{L},\Sigma} \leftarrow \emptyset$
5: **for all** $\mathfrak{L}' : \mathfrak{K}' \in \Sigma$ **do**
6: **if** $\mathfrak{L}' \subseteq \mathfrak{L}$ **then** ▷ Properties of such keys must exist
7: $\mathfrak{K}_{\mathfrak{L},\Sigma} \leftarrow \mathfrak{K}_{\mathfrak{L},\Sigma} \cup \mathfrak{K}'$ ▷ Include properties of the key
8: **if** (NOT(`Unique`) AND $\mathfrak{K}' \subseteq \mathfrak{K}$) **then**
9: `Unique` $\leftarrow$ *TRUE* ▷ Found input key that makes $\mathfrak{L} : \mathfrak{K}$ unique
10: **if** $\mathfrak{K} \subseteq \mathfrak{K}_{\mathfrak{L},\Sigma}$ **then return** `Exists` $\leftarrow$ *TRUE* ▷ All required properties exist
11: **if** (`Exists` AND `Unique`) **then return** *TRUE* ▷ Properties exist and are unique
12: **else return** *FALSE*

Theorem

Let $\Sigma \cup \{\mathfrak{L} : \mathfrak{K}\}$ denote a set of Neo4j keys over $\mathcal{L}$ and $\mathcal{K}$. Then $\Sigma \models \mathfrak{L} : \mathfrak{K}$ if and only if $\mathfrak{K} \subseteq \mathfrak{K}_{\mathfrak{L},\Sigma}$ and there is some $\mathfrak{L}' : \mathfrak{K}' \in \Sigma$ where $\mathfrak{L}' \subseteq \mathfrak{L}$ and $\mathfrak{K}' \subseteq \mathfrak{K}$. The algorithm IMPLY$(\Sigma, \mathfrak{L} : \mathfrak{K})$ *decides the implication problem for the class of Neo4j keys in $\mathcal{O}(||\Sigma \cup \{\mathfrak{L} : \mathfrak{K}\}||)$ time.*

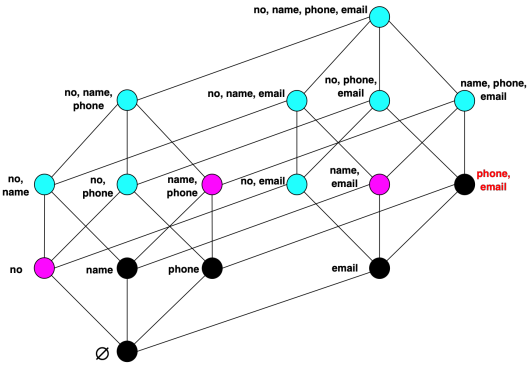
Example for Implication



:Helpline, :Complaints

Powerset lattice of Neo4j keys Implied by the keys marked in magenta: implied non-given keys marked in cyan, and non-implied keys marked in black

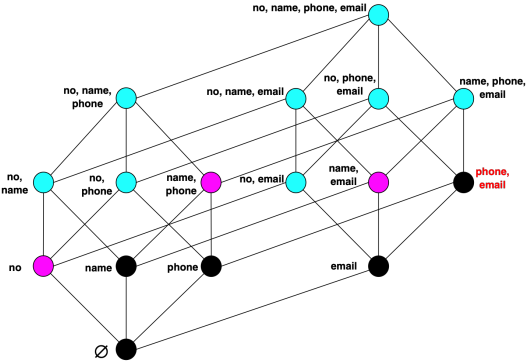
Example for Implication



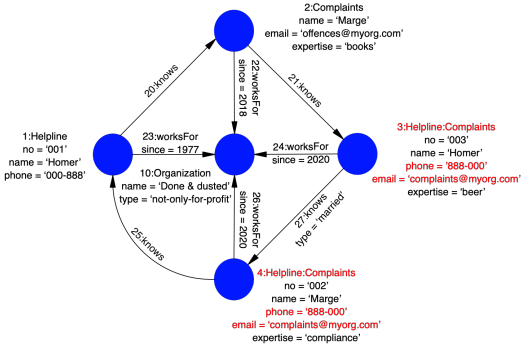
:Helpline, :Complaints

Powerset lattice of Neo4j keys Implied by the keys marked in magenta: implied non-given keys marked in cyan, and non-implied keys marked in black

Example for Implication



:Helpline, :Complaints



Powerset lattice of Neo4j keys Implied by the keys marked in magenta: implied non-given keys marked in cyan, and non-implied keys marked in black

Summary

- Keys in Neo4j require completeness and uniqueness, as do candidate keys in SQL
- Neo4j is a multi-label graph model but only supports single-label keys
- Introduced a notion of Neo4j keys that
  - subsumes the single-label keys currently supported by Neo4j
  - permits the specification of multi-label keys
- Illustrated the importance of Neo4j keys through several use cases, including
  - updates
  - indexing
  - physical and logical query optimization
- Characterized the implication problem
  - axiomatically
  - algorithmically
- Future work
  - Combinatorics, sampling, discovery
  - Smarter notions of keys
  - Other constraints

Contact details

- Professor Sebastian Link
  - School of Computer Science
    The University of Auckland
  - s.link@auckland.ac.nz
  - http://www.science.auckland.ac.nz/people/profile/s-link
- Associate Dean International Science
  - auckland.ac.nz/en/science.html
- Director of Data Science in the Home of *R*
  - auckland.ac.nz/en/study/study-options/find-a-study-option/data-science.html
  - r-project.org