SPARKLing Constraints for RDF

Georg Lausen, Michael Meier, Michael Schmidt

Institut für Informatik
Albert-Ludwigs-Universität Freiburg

RDF and Relational Databases

RDF

- *Resource Description Framework* RDF is the basis for building the semantic web.

  Using RDF, any kind of information can be represented by a set of triples, where each triple states a subject-property-object relationship.

  Triples $(a, b, c) \Longrightarrow$ RDF graph; edge $a \xrightarrow{b} c$.

Relational Databases

- Today, most of the data on the web resides in relational databases.

  Exporting data from relational databases to the semantic web using RDF basically means to map the relational data into an RDF graph.

From Relational Data to RDF

When mapping, what shall we do with the constraints?

Typically, keys and foreign keys are no longer explicit in an RDF graph.

Problems when constraints are lost

- A user builds her own knowledge base by integrating several RDF graphs found on the internet.
- If an exported RDF graph has to be imported in a relational database at another place.
- When updates on a materialized RDF graph have to be performed then key and foreign key properties have to be checked.
- Optimization of queries.

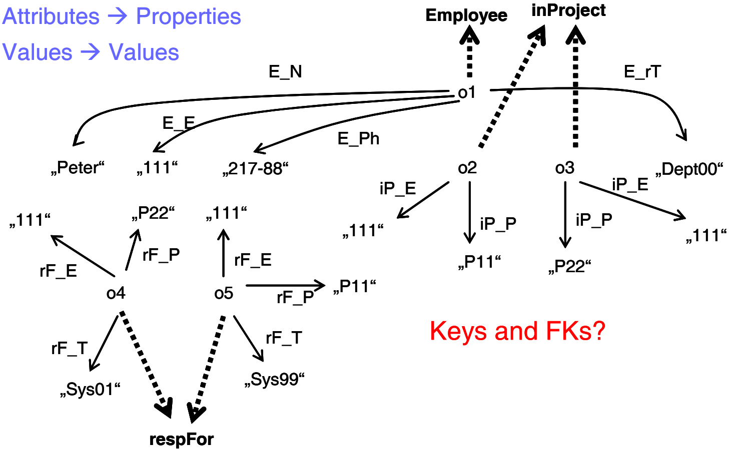
Tupel-based Mapping

Relations → Classes

Tuples → Resource Identifier (IRI / URI)

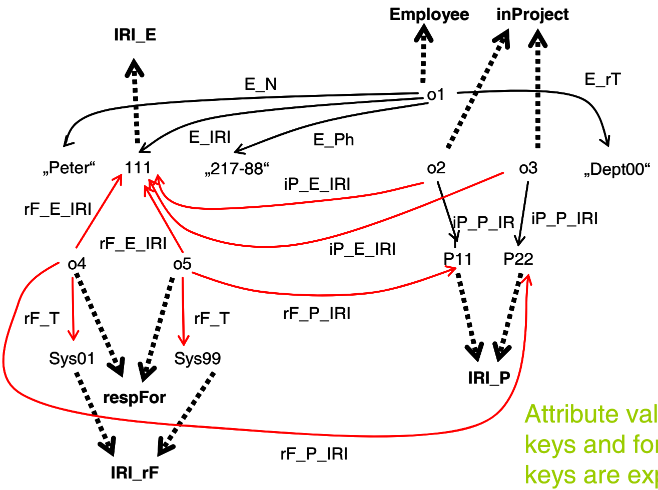
Attributes → Properties

Values → Values



Keys and FKs?

Key and Foreign Key values preserving mapping



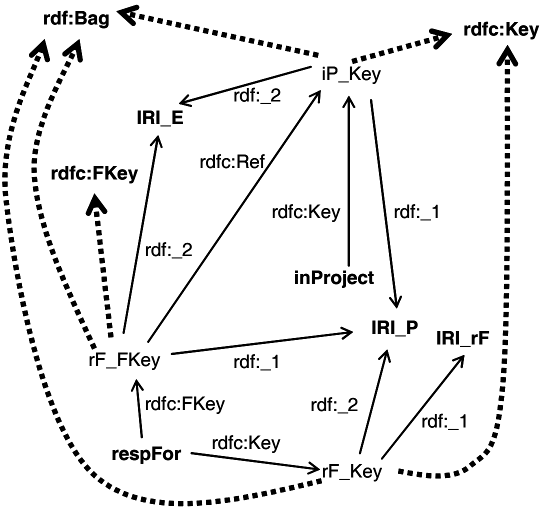
Attribute values of keys and foreign keys are explicitly represented by IRIs

Explicit statement of keys and foreign keys

- Keys and foreign keys are stated inside an RDF graph.
- Extend the RDF vocabulary by a new namespace which prefix `rdfc`.
  - Classes `rdfc:Key` and `rdfc:FKey`. Instances represent keys and foreign keys.
  - Properties `rdfc:Key` and `rdfc:FKey` to associate with each class its key and foreign keys.
  - Property `rdfc:Ref` to link a foreign key to the key of the respective parent class.

very similar to the SQL-approach

Key and Foreign Key statements



Formal definitions

RDF Vocabulary

- An RDF *vocabulary* $\mathcal{V} = (N_C, N_P)$, where N_C is a finite set of *classes* and N_P is a finite set of *properties*.
- An *interpretation* $\mathcal{I}$ of $\mathcal{V}$, $\mathcal{I} = (\Delta_I, \Delta_D, .^{I_C}, .^{I_P})$ is given as
  - Δ_I is a possibly infinite, nonempty set, called *object domain*,
  - Δ_D is a possibly infinite, nonempty set, called the *data domain*, which we assume to be disjoint from Δ_I, i.e. $\Delta_I \cap \Delta_D = \emptyset$,
  - $.^{I_C}$ is the *class interpretation function* assigning to each class $A \in N_C$ a finite subset $A^{I_C} \subseteq \Delta_I$,
  - $.^{I_P}$ is the *property interpretation function* assigning to each property $Q \in N_P$ a finite subset $Q^{I_P} \subseteq \Delta_I \times (\Delta_I \cup \Delta_D)$.

Constraints

Key and Foreign Key

Let $\mathcal{V} = (N_C, N_P)$ be a vocabulary and $\mathcal{I} = (\Delta_I, \Delta_D, \cdot^{I_C}, \cdot^{I_P})$ an interpretation of $\mathcal{V}$.

- $\mathcal{I}$ satisfies $Key(C, [Q_1, \ldots, Q_n])$,

$$\mathcal{I} \models Key(C, Q_1, \ldots, Q_n),$$

  if, whenever $\exists o_1, o_2 \in C^{I_C}$, $\exists v_i \in \Delta_I \cup \Delta_D, 1 \leq i \leq n$, such that $(o_1, v_i), (o_2, v_i) \in Q_i^{I_P}$, then $o_1 = o_2$.
- $\mathcal{I}$ satisfies $FK(C, [Q_1, \ldots, Q_n], C', [Q'_1, \ldots, Q'_n])$,

$$\mathcal{I} \models FK(C, [Q_1, \ldots, Q_n], C', [Q'_1, \ldots, Q'_n]),$$

  if, whenever $o_1 \in C^{I_C}$, then $\exists o_2 \in C'^{I_C}$ such that $(o_1, v_i) \in Q_i^{I_P}$ implies $(o_2, v_i) \in Q_i'^{I_P}, 1 \leq i \leq n$.

RDFS Constraints SubC, SubP, PropD, PropR

Let be given a vocabulary $\mathcal{V} = (N_C, N_P)$ of RDF and a corresponding *interpretation* $\mathcal{I} = (\Delta_I, \Delta_D, .^{I_C}, .^{I_P})$. Let $C, D \in N_C$ and $R, S \in N_P$. Let ϕ be one of the constraints mentioned above.

$\mathcal{I}$ satisfies ϕ, $\mathcal{I} \models \phi$, if depending on ϕ there holds:

$$\begin{array}{ll} SubC(C, D): & C^{I_C} \subseteq D^{I_C}, \\ SubP(R, S): & R^{I_P} \subseteq S^{I_P}, \\ PropD(R, C): & \{x \mid \exists y : (x, y) \in R^{I_P}\} \subseteq C^{I_C}, \\ PropR(R, C): & \{y \mid \exists x : (x, y) \in R^{I_P}\} \subseteq C^{I_C}. \end{array}$$

Cardinalities

Let $n \geq 0$ and $C \in N_C$, $R \in N_P$.

$\mathcal{I}$ satisfies ψ, $\mathcal{I} \models \psi$, if there holds:

$$\begin{aligned} Min(C, n, R) &: \ \{x \mid \#\{y \mid (x, y) \in R^{I_P}\} \geq n\} \supseteq C^{I_C} \\ Max(C, n, R) &: \ \{x \mid \#\{y \mid (x, y) \in R^{I_P}\} \leq n\} \supseteq C^{I_C}. \end{aligned}$$

Subproperty-Chain

Let $\circ$ denote the composition of binary relations.

Let $\phi = \mathit{SubPChain}(C, R_1, \ldots, R_n, S)$. $\mathcal{I}$ satisfies ϕ, $\mathcal{I} \models \phi$, if there holds:

$$\{(x, y) \mid (x, y) \in R_1^{I_P} \circ \ldots \circ R_n^{I_P}, x \in C^{I_C}\} \subseteq \{(x, y) \mid (x, y) \in S^{I_P}, x \in C^{I_C}\}.$$

Anti-Key

- $\mathcal{I}$ satisfies $AntiKey(C, [Q_1 \ldots Q_n])$, $\mathcal{I} \models AntiKey(C, [Q_1 \ldots Q_n])$, if $\exists o_1, o_2 \in C^{I_C}$, $o_1 \neq o_2$, $\exists v_i \in \Delta_I \cup \Delta_D, 1 \leq i \leq n$, such that $(o_1, v_i), (o_2, v_i) \in Q_i^{I_P}$.

Checking Constraints

```
ASK {
  aConstraint expressed as a SPARQL query.
}
```

A constraint is violated, whenever `ASK` returns `true`.

Checking Key Constraints `Key(C,[P1,...,Pn])`

```
ASK {
  ?x rdf:type C.
  ?y rdf:type C.
  ?x p1 ?p1; ...; pn ?pn.
  ?y p1 ?p1; ...; pn ?pn.
  FILTER (?x!=?y)
}
```

Checking Foreign Key Constraints FK(C,[P1,...,Pn],D,[Q1,...,Qn])

```
ASK {
  ?x rdf:type C; p1 ?p1; ...; pn ?pn.
  OPTIONAL {
    ?y rdf:type D; q1 ?p1; ...; qn ?pn.
  } FILTER (!bound(?y))
}
```

Checking Cardinality: `Max(C,n,P)`

$\texttt{allDist([?p1,...,?pn])} \overset{def}{:=} \bigwedge_{1 \leq i \leq n}(\bigwedge_{i < j \leq n} \texttt{?pi!=?pj})$

```
ASK {
  ?x rdf:type C.
  ?x p ?p1; ...; p ?pn+1.
  FILTER (allDist([?p1,...,?pn+1]))
}
```

Checking Cardinality: Min(C,n,P)

$$\texttt{allDist([?p1,...,?pn])} \overset{def}{\coloneqq} \bigwedge_{1\leq i\leq n}(\bigwedge_{i<j\leq n}\texttt{?pi!=?pj})$$

```
ASK {
  ?x rdf:type C.
  OPTIONAL {
    ?y rdf:type C.
    ?y p ?p1; ...; p ?pn.
    FILTER (allDist(?p1,...,?pn) && ?x=?y)
  } FILTER (!bound(?y))
}
```

Checking SubProperty-Chain Constraints SubPChain(C,P1,...,Pn,Q)

```
ASK {
  ?x  rdf:class  C;  p1 ?p1.
  ?p1 p2 ?p2. .... ?pn-1  pn ?pn.
  OPTIONAL  {  ?x q ?q. FILTER (?pn=?q)  }
  FILTER (!bound(?q))
}
```

Checking Anti-key Constraints `AntiKey(C,[P1,...,Pn])`

```
ASK {
  ?x rdf:type C.
  ?y rdf:type C.
  ?x p1 ?p1; ...; pn ?pn.
  ?y p1 ?p1; ...; pn ?pn.
  FILTER (?x!=?y)
}
```

Anti-key constraints are violated, if `ASK` returns `false`.

Complexity of Constraint Checking

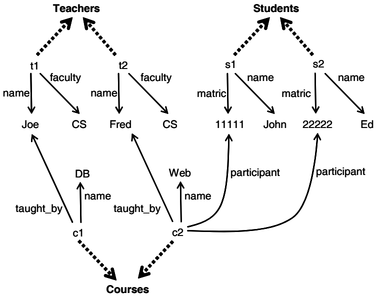
Complexity of SPARQL (J.Perez, M.Arenas and Gutierrez; 2006):

- combined complexity: PSPACE-complete.
- data complexity: LOGSPACE, resp. PTIME.

Exploiting Constraints: when to ignore OPTIONAL

```
SELECT ?student, ?teachername          ⟹
  WHERE {
  ?course  rdf:type    Courses.
  ?course  name         ?coursename.
  ?student rdf:type    Students.
  ?course  participant ?student.
  OPTIONAL {
    ?course  taught_by ?teachername.
    ?teacher rdf:type Teachers.
    ?teacher name      ?teachername. } }
```

```
SELECT ?student, ?teachername
  WHERE {
  ?course  rdf:type    Courses.
  ?course  name         ?coursename.
  ?student rdf:type    Students.
  ?course  participant ?student.
  ?course  taught_by ?teachername.
  ?teacher rdf:type Teachers.
  ?teacher name      ?teachername. } }
```



```
SELECT ?student, ?teachername
  WHERE {
  ?course  rdf:type   Courses.
  ?course  name        ?coursename.
  ?student rdf:type   Students.
  ?course  participant ?student.
  OPTIONAL {
    ?course  taught_by ?teachername.
    ?teacher rdf:type Teachers.
    ?teacher name      ?teachername. } }
```

$\Longrightarrow$

```
SELECT ?student, ?teachername
  WHERE {
  ?course  rdf:type   Courses.
  ?course  name        ?coursename.
  ?student rdf:type   Students.
  ?course  participant ?student.
  ?course  taught_by ?teachername.
  ?teacher rdf:type Teachers.
  ?teacher name      ?teachername. } }
```

Constraints

`taught_by`:

- primary key of `Courses`, and
- foreign key with respect to `name` of `Teachers`.

OPTIONAL can be ignored

Facts	`edge(a,rdf:type,Courses)`
	`edge(a,name,b)`
	`edge(c,rdf:type,Students)`
	`edge(a,participant,c)`

Query	∃ `X,Y edge(a,taught_by,X), edge(Y,type,Teachers), edge(Y,name,X).`
Constraints	`(K1) taught_by is key of Courses and thus total.` `(FK) taught_by is foreign key of Courses with respect to name of Teachers.`
`(K1)` `(FK)`	∃ `Y edge(a,taught_by,d), edge(Y,type,Teachers), edge(Y,name,d).` `edge(a,taught_by,d), edge(e,type,Teachers), edge(e,name,d).`
Answer	`edge(a,taught_by,d), edge(e,type,Teachers), edge(e,name,d).`

Satisfiability

Decidability

- Let $\mathcal{R}$ be a (relational) schema, Σ a set of keys and foreign keys over $\mathcal{R}$, and φ a key over $\mathcal{R}$. It is known that the implication-problem of φ from Σ is undecidable.
- This means, whenever we allow a set of constraints being formed out of key, foreign key and anti-key constraints, satisfiability is undecidable.

Theorem

Let $\mathcal{V}$ be a RDF vocabulary, $\mathcal{C}$ be a set of constraints over $\mathcal{V}$ containing arbitrary constraints, however no anti-key constraints. Testing satisfiability of $\mathcal{V}$ with respect to $\mathcal{C}$ is undecidable.

$\mathcal{ALCHIQ}$

The satisfiability of RDF vocabularies, where subclass, subproperty, property domain and range, min-cardinality, max-cardinality, unary foreign key and unary key constraints are allowed, can be decided using a reduction to the description logic $\mathcal{ALCHIQ}$. For $\mathcal{ALCHIQ}$ it is known that satisfiability can be decided in exponential time.

Our framework	$\mathcal{ALCHIQ}$ **construct**
$SubC(C, D)$	$C \sqsubseteq D$
$SubP(R, S)$	$R \sqsubseteq S$
$PropD(R, C)$	$\exists R.\top \sqsubseteq C$
$PropR(R, C)$	$\exists R^{-}.\top \sqsubseteq C$
$Min(C, n, R)$	$C \sqsubseteq \geq nR$
$Max(C, n, R)$	$C \sqsubseteq \leq nR$
$FK(C, [R], D, [S])$	$\exists R^{-}.C \sqsubseteq \exists S^{-}.D$
$Key(C, [R])$	$\exists R^{-}.C \equiv \leq 1R^{-}.C$

Outlook

... $\mathcal{SEMANTIC\ PROCESSING\ OF\ SPARQL}$.