Ontology-Based Data Access via Query Rewriting: Practice

Roman Kontchakov

Department of Computer Science and Inf. Systems, Birkbeck College, London

http://www.dcs.bbk.ac.uk/~roman

Example: Who Works with Professors?

$\mathcal{T}$: $\mathsf{RA} \sqsubseteq \exists \mathsf{worksOn}.\mathsf{Project}$ $\qquad \mathsf{worksOn}^{-} \sqsubseteq \mathsf{involves}$

$\mathsf{Project} \sqsubseteq \exists \mathsf{isManagedBy}.\mathsf{Professor}$ $\qquad \mathsf{isManagedBy} \sqsubseteq \mathsf{involves}$

$\mathcal{A}$: $\mathsf{RA}(\mathsf{stas})$, $\mathsf{worksOn}(\mathsf{stas}, \mathsf{itract})$, $\mathsf{Project}(\mathsf{itract})$, $\mathsf{Lecturer}(\mathsf{igor})$, $\mathsf{worksOn}(\mathsf{igor}, \mathsf{itract})$

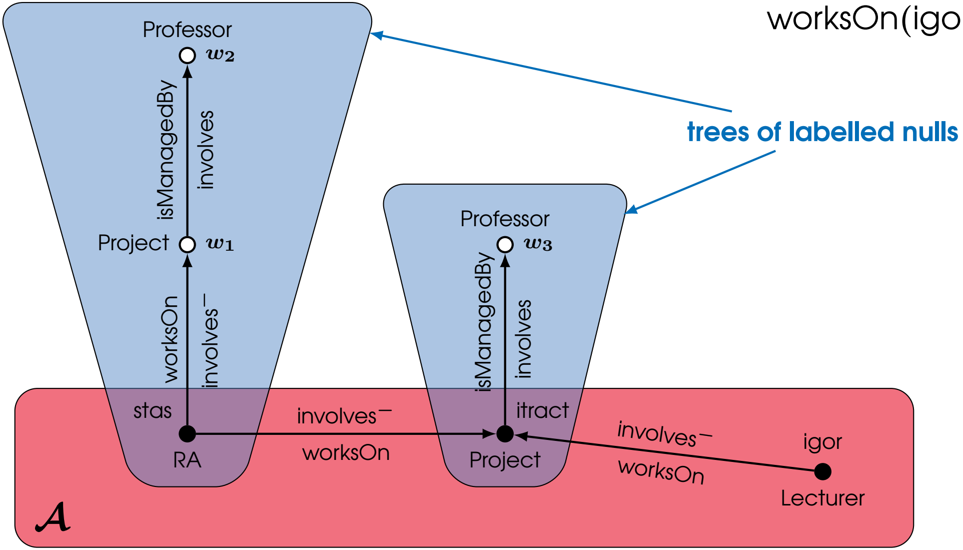
Example: Who Works with Professors?

$\mathcal{T}$: $\mathsf{RA} \sqsubseteq \exists \mathsf{worksOn}.\mathsf{Project}$ $\qquad$ $\mathsf{worksOn}^{-} \sqsubseteq \mathsf{involves}$

$\mathsf{Project} \sqsubseteq \exists \mathsf{isManagedBy}.\mathsf{Professor}$ $\qquad$ $\mathsf{isManagedBy} \sqsubseteq \mathsf{involves}$

$\mathcal{A}$: $\mathsf{RA}(\mathsf{stas}),\ \ \mathsf{worksOn}(\mathsf{stas}, \mathsf{itract}),\ \ \mathsf{Project}(\mathsf{itract}),\ \ \mathsf{Lecturer}(\mathsf{igor}),$

$\mathsf{worksOn}(\mathsf{igor}, \mathsf{itract})$



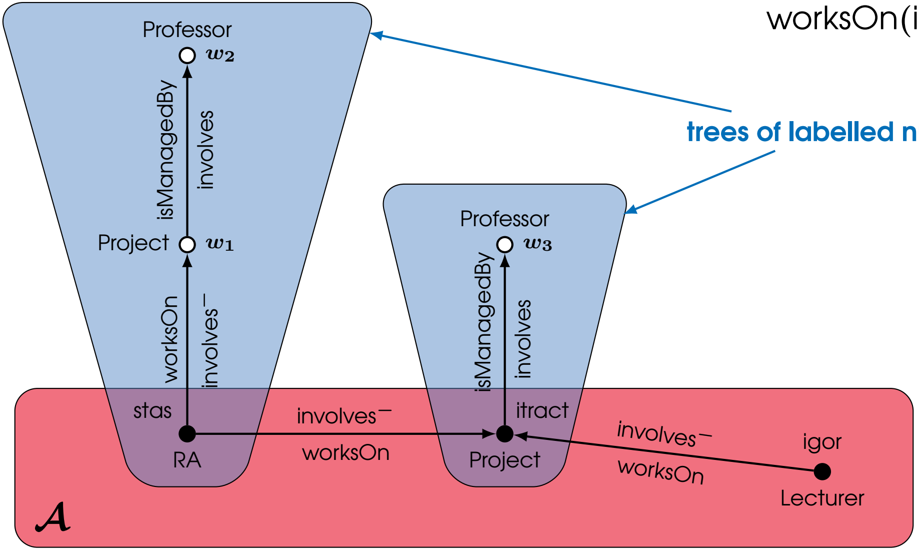
trees of labelled nulls

Example: Who Works with Professors?

$\mathcal{T}$: RA $\sqsubseteq$ $\exists$worksOn.Project   worksOn$^-$ $\sqsubseteq$ involves

Project $\sqsubseteq$ $\exists$isManagedBy.Professor   isManagedBy $\sqsubseteq$ involves

$\mathcal{A}$: RA(stas), worksOn(stas, itract), Project(itract), Lecturer(igor), worksOn(igor, itract)



CQ: $q(x) = \exists y, z\,(\text{worksOn}(x, y) \land \text{involves}(y, z) \land \text{Professor}(z))$

Who Works with Professors: Tree-Witness Query Rewriting

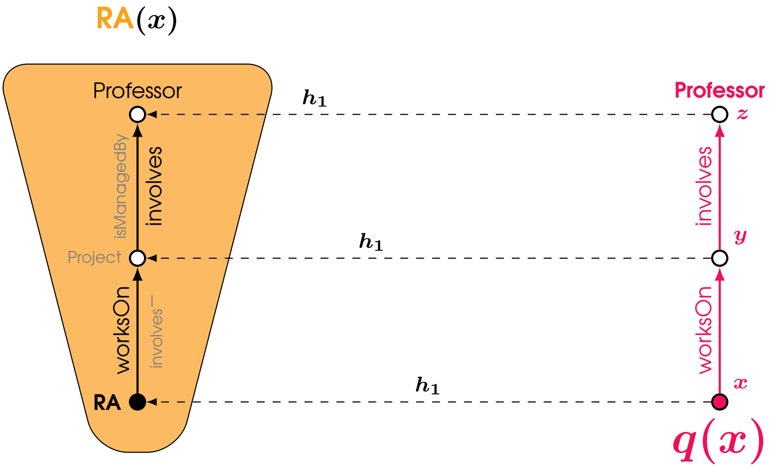
a **tree witness** ≈ a fragment of the query such that

- only `boundary' (join) variables may be answer variables
- the fragment is embeddable into a tree of labelled nulls

Who Works with Professors: Tree-Witness Query Rewriting

a **tree witness** ≈ a fragment of the query such that

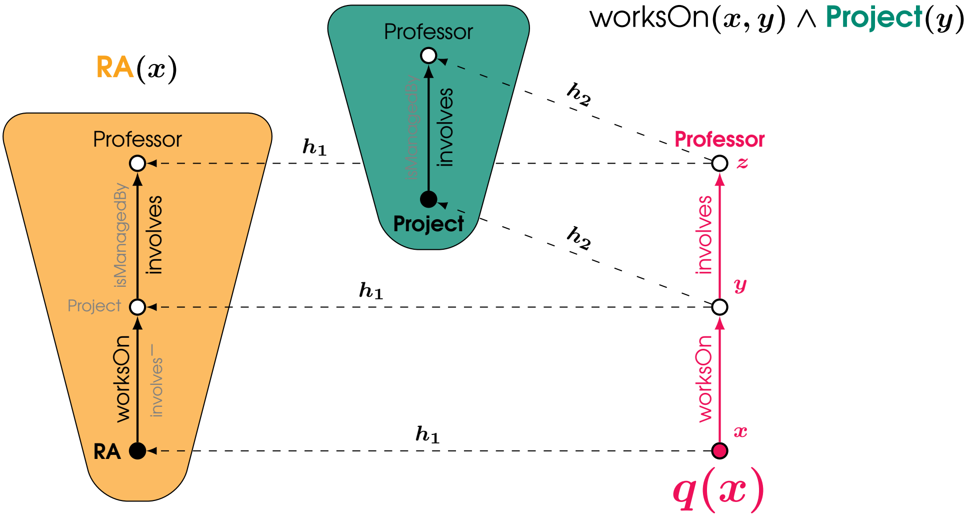
- only 'boundary' (join) variables may be answer variables
- the fragment is embeddable into a tree of labelled nulls



Who Works with Professors: Tree-Witness Query Rewriting

a **tree witness** ≈ a fragment of the query such that

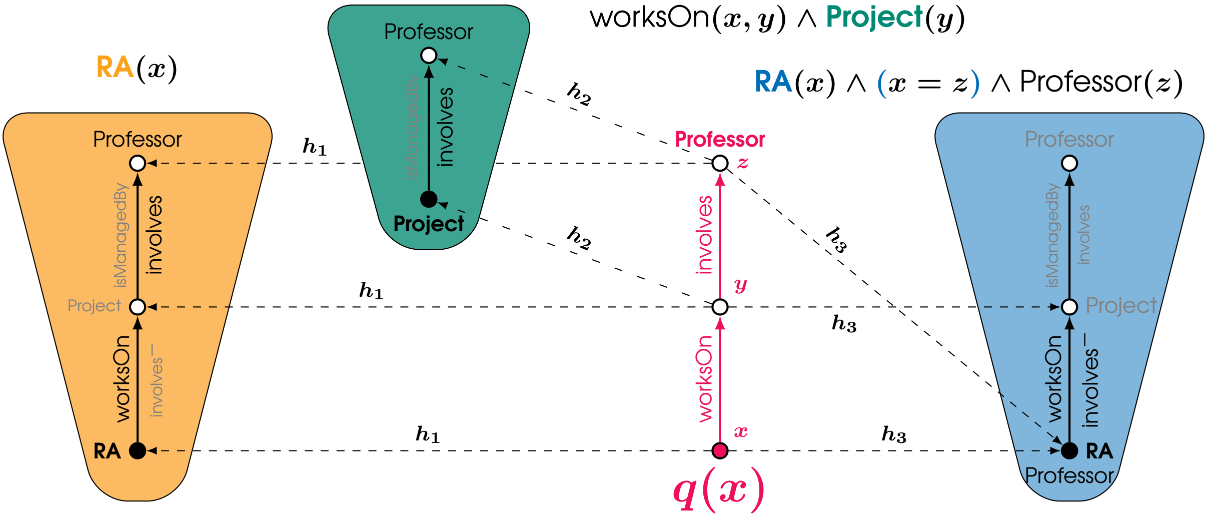
- only ‘boundary’ (join) variables may be answer variables
- the fragment is embeddable into a tree of labelled nulls



Who Works with Professors: Tree-Witness Query Rewriting

a **tree witness** $\approx$ a fragment of the query such that

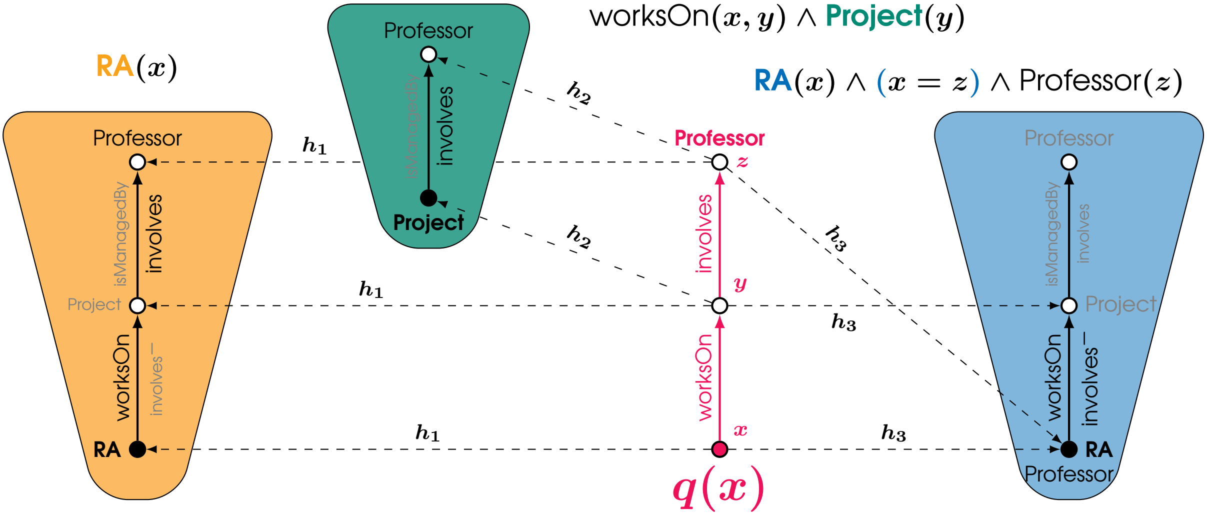
- only 'boundary' (join) variables may be answer variables
- the fragment is embeddable into a tree of labelled nulls



Who Works with Professors: Tree-Witness Query Rewriting

a **tree witness** $\approx$ a fragment of the query such that

- only 'boundary' (join) variables may be answer variables
- the fragment is embeddable into a tree of labelled nulls



PE-rewriting:

$$q'(x) \;=\; \exists y, z\,\Big[\mathsf{RA}(x) \;\lor\; (\mathsf{worksOn}(x,y) \land \mathsf{Project}(y)) \;\lor\; (\mathsf{RA}(x) \land (x = z) \land \mathsf{Professor}(z)) \;\lor\; (\mathsf{worksOn}(x,y) \land \mathsf{involves}(y,z) \land \mathsf{Professor}(z))\Big]$$

Tree Witness Rewriting in Practice

a **tree witness** ≈ a fragment of the query such that

- only ‘boundary’ (join) variables may be answer variables
- the fragment is embeddable into a tree of labelled nulls
- constants may be mapped only to the root of the tree

- University, Stockexchange, LUBM, Vicodi have **no** tree witnesses

➡ the rewriting = replacing each atoms with **disjunctions** of atoms
(SCQs, semi-conjunctive queries)

Tree Witness Rewriting in Practice

a **tree witness** ≈ a fragment of the query such that

- only 'boundary' (join) variables may be answer variables
- the fragment is embeddable into a tree of labelled nulls
- constants may be mapped only to the root of the tree

- University, Stockexchange, LUBM, Vicodi have **no** tree witnesses

➡ the rewriting = replacing each atoms with **disjunctions** of atoms
(SCQs, semi-conjunctive queries)

- LUBM-ex-20 have 0/1 tree witnesses, one query has 3 (from Vienna)
(but these 3 only simplify the rewriting due to CQ containment check)
- Adolena have 0/1 tree witnesses
- P5 have 1–**5** tree witnesses (but they are all nicely nested)

Tree Witness Rewriting in Practice

a **tree witness** ≈ a fragment of the query such that

- only ‘boundary’ (join) variables may be answer variables
- the fragment is embeddable into a tree of labelled nulls
- constants may be mapped only to the root of the tree

- University, Stockexchange, LUBM, Vicodi have **no** tree witnesses

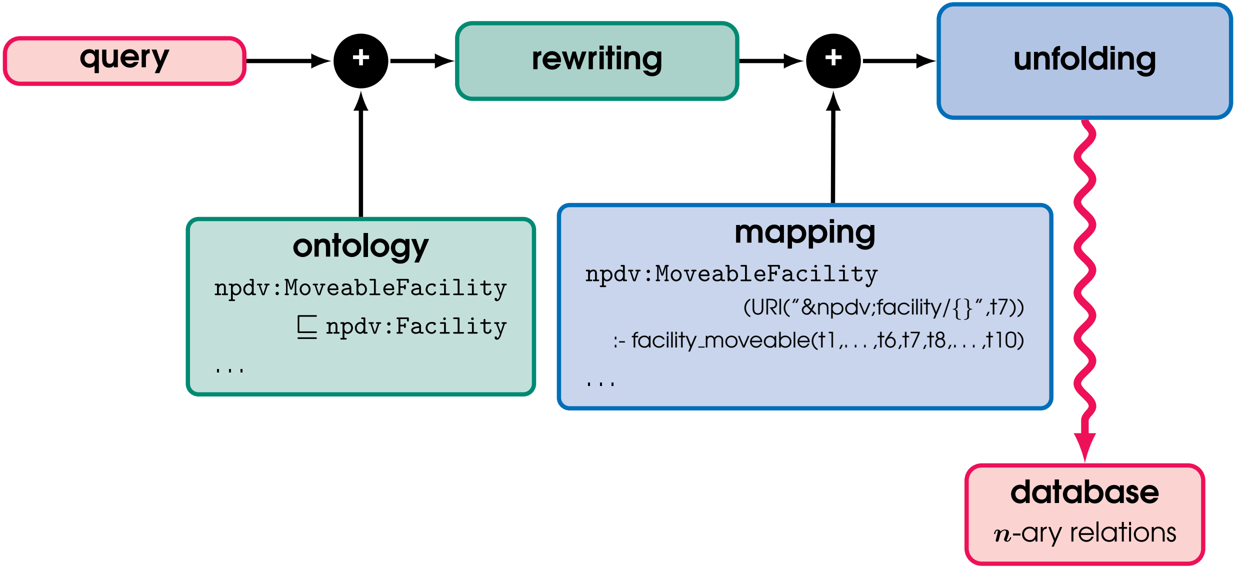
  ➡ the rewriting = replacing each atoms with **disjunctions** of atoms
  (SCQs, semi-conjunctive queries)

- LUBM-ex-20 have 0/1 tree witnesses, one query has 3 (from Vienna)
  (but these 3 only simplify the rewriting due to CQ containment check)
- Adolena have 0/1 tree witnesses
- P5 have 1–**5** tree witnesses (but they are all nicely nested)

entailment regime in SPARQL : **difficult** to write queries with non-answer variables

```
SELECT ?x WHERE {
   ?x a [a owl:Restriction; owl:onProperty :worksOn; owl:someValuesFrom
     [a owl:Restriction; owl:onProperty :involves; owl:someValuesFrom :Professor]
   ]
}
```

OBDA = Ontology + Mappings



query

+

rewriting

+

unfolding

ontology

npdv:MoveableFacility

⊑ npdv:Facility

...

mapping

npdv:MoveableFacility

(URI(“&npdv;facility/{}”,t7))

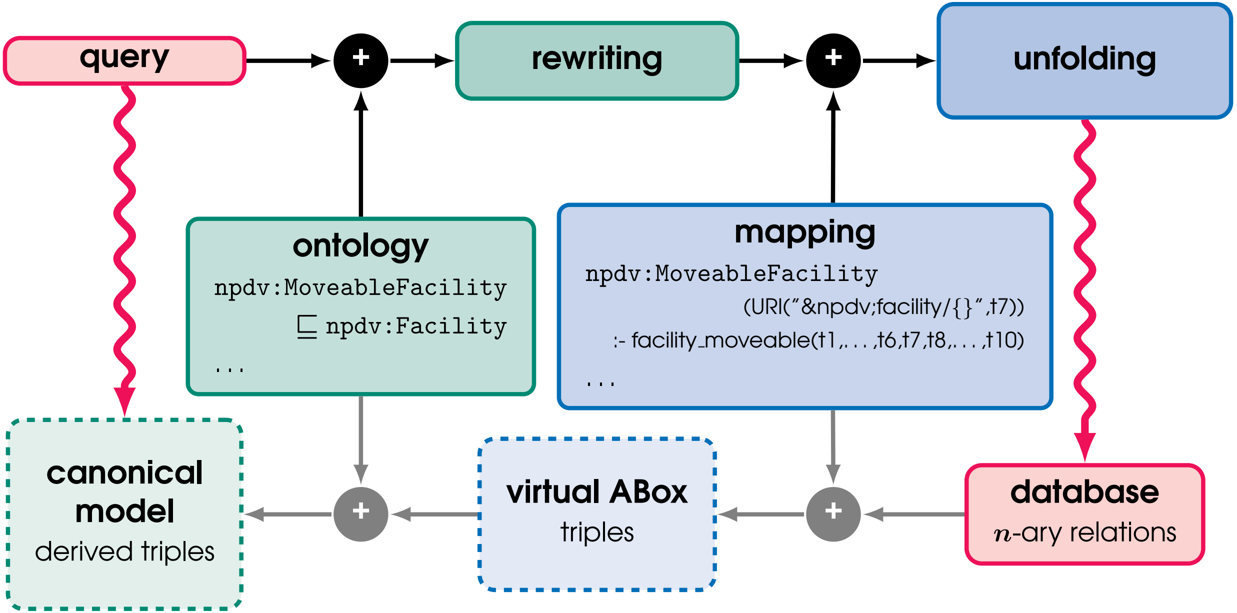
:- facility_moveable(t1,. . . ,t6,t7,t8,. . . ,t10)

...

database

n-ary relations

OBDA = Ontology + Mappings



query

rewriting

unfolding

ontology

```
npdv:MoveableFacility
          ⊑ npdv:Facility
...
```

mapping

```
npdv:MoveableFacility
```
(URI("&npdv;facility/{}",t7))

:- facility_moveable(t1,. . . ,t6,t7,t8,. . . ,t10)

. . .

canonical model

derived triples

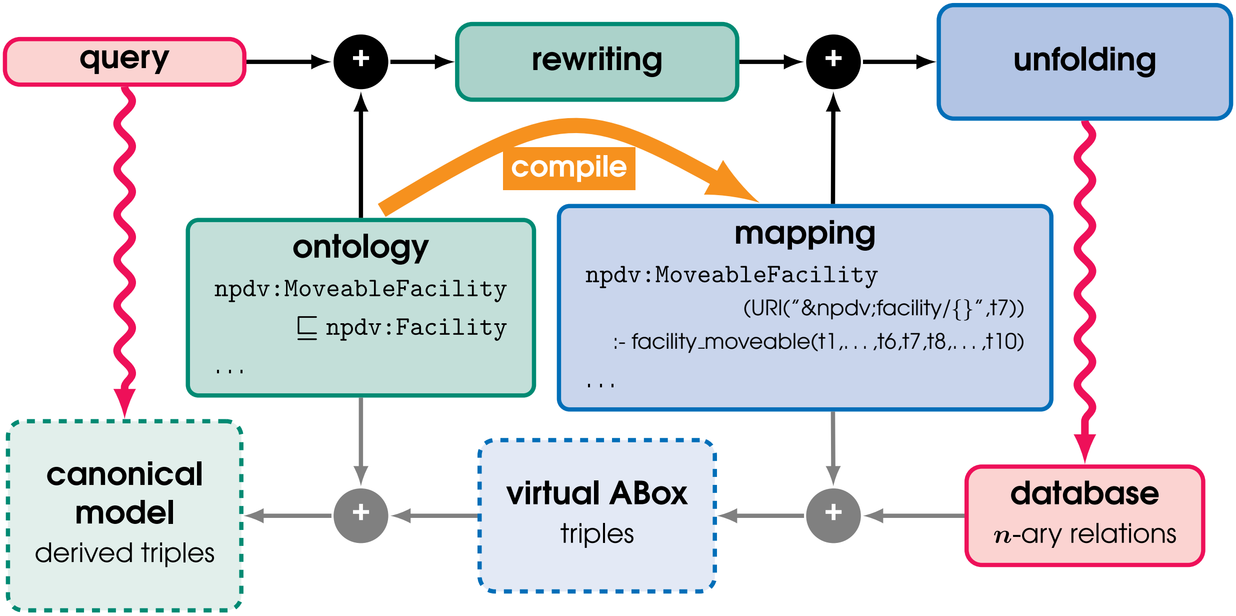
virtual ABox

triples

database

n-ary relations

OBDA = Ontology + Mappings



query

+

rewriting

+

unfolding

compile

ontology

npdv:MoveableFacility

$\sqsubseteq$ npdv:Facility

...

mapping

npdv:MoveableFacility

(URI("&npdv;facility/{}",t7))

:- facility_moveable(t1,...,t6,t7,t8,...,t10)

...

canonical model

derived triples

+

virtual ABox

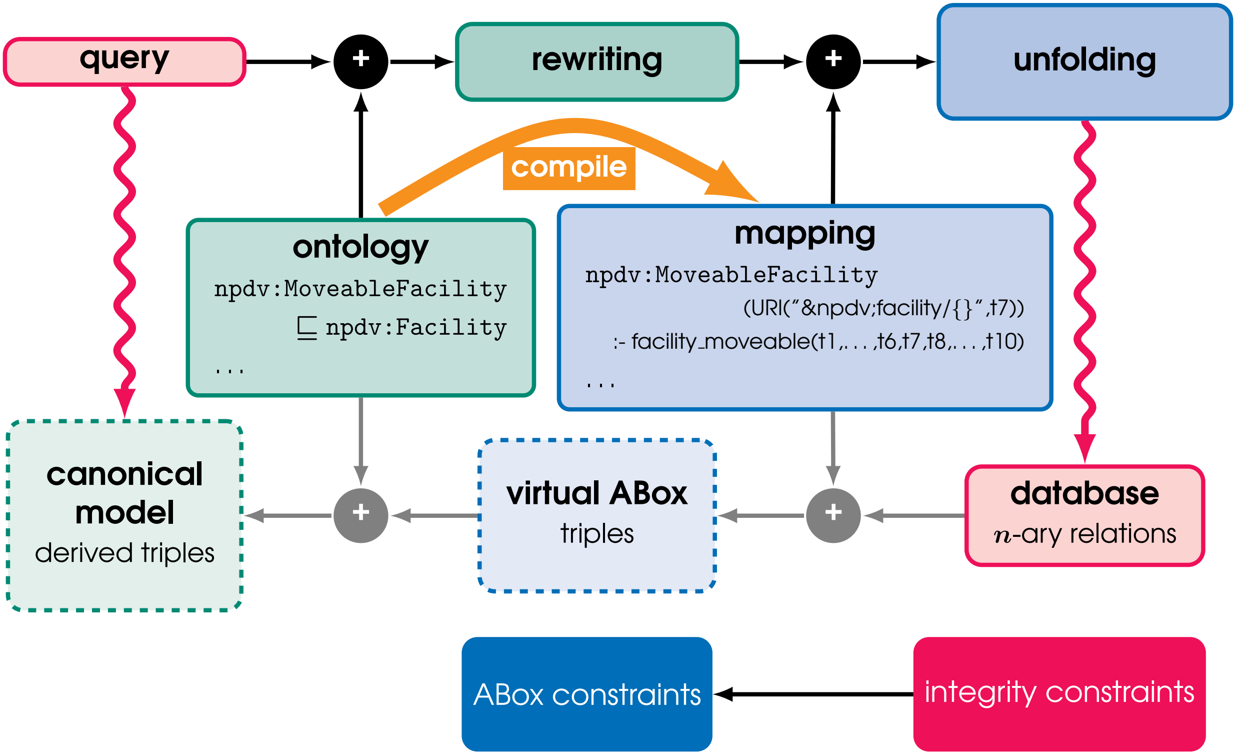
triples

+

database

n-ary relations

OBDA = Ontology + Mappings



query

rewriting

unfolding

compile

ontology

`npdv:MoveableFacility`

$\sqsubseteq$ `npdv:Facility`

...

mapping

`npdv:MoveableFacility`

(URI("&npdv;facility/{}",t7))

:- facility_moveable(t1,...,t6,t7,t8,...,t10)

...

canonical model

derived triples

virtual ABox

triples

database

n-ary relations

ABox constraints

integrity constraints

rewritings over **complete** ABoxes

Compiling the Ontology into Mappings: Class Inclusions

```
npdv:MoveableFacility(URI("&npdv;facility/{}",t7))
      :- facility_moveable(t1,...,t6,t7,t8,...,t10)

npdv:FixedFacility(URI("&npdv;facility/{}",t24))
      :- facility_fixed(t1,...,t23,t24,t25,t26)
```

Compiling the Ontology into Mappings: Class Inclusions

```
npdv:MoveableFacility(URI("&npdv;facility/{}",t7))
     :- facility_moveable(t1,...,t6,t7,t8,...,t10)

npdv:FixedFacility(URI("&npdv;facility/{}",t24))
     :- facility_fixed(t1,...,t23,t24,t25,t26)
```

⊑ npdv:Facility

Compiling the Ontology into Mappings: Class Inclusions

```
npdv:MoveableFacility(URI("&npdv;facility/{}",t7))
      :- facility_moveable(t1,...,t6,t7,t8,...,t10)

npdv:FixedFacility(URI("&npdv;facility/{}",t24))
      :- facility_fixed(t1,...,t23,t24,t25,t26)
```

$\sqsubseteq$ npdv:Facility

```
npdv:Facility(URI("&npdv;facility/{}",t))
      :- facility_moveable(t1,...,t6,t,t8,...,t10)
      :- facility_fixed(t1,...,t23,t,t25,t26)
```

Compiling the Ontology into Mappings: Class Inclusions

```
npdv:MoveableFacility(URI("&npdv;facility/{}",t7))
      :- facility_moveable(t1,...,t6,t7,t8,...,t10)
npdv:FixedFacility(URI("&npdv;facility/{}",t24))
      :- facility_fixed(t1,...,t23,t24,t25,t26)
```

⊑ `npdv:Facility`

↓

```
npdv:Facility(URI("&npdv;facility/{}",t))
      :- facility_moveable(t1,...,t6,t,t8,...,t10)
      :- facility_fixed(t1,...,t23,t,t25,t26)
      :- ...
```

NB: `npdv:Facility` has many other subclasses

T-mappings

Compiling the Ontology into Mappings: Domains and Ranges

$\texttt{npdv:currentResponsibleCompany}^{-}.\top \sqsubseteq \texttt{npdv:Company}$

```
npdv:currentResponsibleCompany rdfs:range npdv:Company .
```

Compiling the Ontology into Mappings: Domains and Ranges

$\texttt{npdv:currentResponsibleCompany}^{-}.\top \sqsubseteq \texttt{npdv:Company}$

```
npdv:currentResponsibleCompany rdfs:range npdv:Company .
```

+

```
npdv:currentResponsibleCompany
                 (URI("&npdv;facility/{}",t7),URI("&npdv;company/{}",t8))
     :- facility_moveable(t1,...,t6,t7,t8,t9,t10)
```

Compiling the Ontology into Mappings: Domains and Ranges

$\texttt{npdv:currentResponsibleCompany}^{-}.\top \sqsubseteq \texttt{npdv:Company}$

```
npdv:currentResponsibleCompany rdfs:range npdv:Company .
```

+

```
npdv:currentResponsibleCompany
                 (URI("&npdv;facility/{}",t7),URI("&npdv;company/{}",t8))
     :- facility_moveable(t1,...,t6,t7,t8,t9,t10)
```

↓

```
npdv:Company(URI("&npdv;company/{}",t8))
     :- facility_moveable(t1,...,t6,t7,t8,t9,t10)
```

Compiling the Ontology into Mappings: beyond OWL 2 QL

$\texttt{npdv:Wellbore} \sqcap \texttt{npdv:wellborePlotSymbol}.\texttt{171} \sqsubseteq \texttt{npdv:InjectionWellbore}$

Compiling the Ontology into Mappings: beyond OWL 2 QL

$\texttt{npdv:Wellbore} \sqcap \texttt{npdv:wellborePlotSymbol}.\texttt{171} \sqsubseteq \texttt{npdv:InjectionWellbore}$

+

```
npdv:Wellbore(URI("&npdv;wellbore/{}",t2))
    :- wellbore_npdid_overview(t1,t2,t3,...,t5)

npdv:wellborePlotSymbol(URI("&npdv;wellbore/{}",t54),xsd:integer(t29))
    :- wellbore_development_all(t1,...,t28,t29,t30,...,t53,t54,t55,...,t65)
```

Compiling the Ontology into Mappings: beyond OWL 2 QL

$\texttt{npdv:Wellbore} \sqcap \texttt{npdv:wellborePlotSymbol}.\texttt{171} \sqsubseteq \texttt{npdv:InjectionWellbore}$

+

```
npdv:Wellbore(URI("&npdv;wellbore/{}",t2))
    :- wellbore_npdid_overview(t1,t2,t3,...,t5)

npdv:wellborePlotSymbol(URI("&npdv;wellbore/{}",t54),xsd:integer(t29))
    :- wellbore_development_all(t1,...,t28,t29,t30,...,t53,t54,t55,...,t65)
```

↓

```
npdv:InjectionWellbore(URI("&npdv;wellbore/{}",t2))
    :- wellbore_npdid_overview(t1,t2,t3,...,t5),
       wellbore_development_all(t1′,...,t28′,171,t30′,...,t53′,t2,t55′,...,t65′)
```

Compiling the Ontology into Mappings: beyond OWL 2 QL

```
npdv:Wellbore ⊓ npdv:wellborePlotSymbol.171 ⊑ npdv:InjectionWellbore
```

+

```
npdv:Wellbore(URI("&npdv;wellbore/{}",t2))
   :- wellbore_npdid_overview(t1,t2,t3,...,t5)

npdv:wellborePlotSymbol(URI("&npdv;wellbore/{}",t54),xsd:integer(t29))
   :- wellbore_development_all(t1,...,t28,t29,t30,...,t53,t54,t55,...,t65)
```

↓

```
npdv:InjectionWellbore(URI("&npdv;wellbore/{}",t2))
   :- wellbore_npdid_overview(t1,t2,t3,...,t5),
      wellbore_development_all(t1′,...,t28′,171,t30′,...,t53′,t2,t55′,...,t65′)
```

NB: such GCIs are **ignored** by ontop now (but could be supported in the future)

Compiling the Ontology into Mappings: Limitations

- inclusions $\exists R.A \sqsubseteq B$ **cannot** be supported (if $A \neq \top$)

  because $\mathrm{AC}_0 \subsetneq$ NLogSpace

Compiling the Ontology into Mappings: Limitations

- inclusions $\exists R.A \sqsubseteq B$ **cannot** be supported (if $A \neq \top$)

  because $\mathrm{AC}_0 \subsetneq$ NLogSpace

- inclusions $A \sqsubseteq \exists R.B$ are supported but **cannot be compiled**

query rewriting is needed!

Compiling the Ontology into Mappings: Limitations

- inclusions $\exists R.A \sqsubseteq B$ **cannot** be supported (if $A \neq \top$)

  because $\mathrm{AC}_0 \subsetneq$ NLogSpace

- inclusions $A \sqsubseteq \exists R.B$ are supported but **cannot be compiled**

  query rewriting is needed!

- OWL 2 QL has **no** `owl:sameAs`...

Compiling the Ontology into Mappings: Limitations

- inclusions $\exists R.A \sqsubseteq B$ **cannot** be supported (if $A \neq \top$)

  because $\mathrm{AC}_0 \subsetneq$ NLogSpace

- inclusions $A \sqsubseteq \exists R.B$ are supported but **cannot be compiled**

  query rewriting is needed!

- OWL 2 QL has **no** `owl:sameAs`... but

```
owl:sameAs(URI("&npdv;company/{}",t5),URI("&brreg;{}",t2))
          :- company(t1,t2,t3,t4,t5,t6,...,t10)
```

could be supported because (a) IRI templates are incompatible and (b)

```
CREATE TABLE ‘company‘ (
   ...
   ‘cmpNpdidCompany‘ int(11) NOT NULL,
   ...
   PRIMARY KEY (‘cmpNpdidCompany‘)
);
```

Compiling the Ontology into Mappings: Limitations

- inclusions $\exists R.A \sqsubseteq B$ **cannot** be supported (if $A \neq \top$)

  because AC$_0$ $\subsetneq$ NLogSpace

- inclusions $A \sqsubseteq \exists R.B$ are supported but **cannot be compiled**

  query rewriting is needed!

- OWL 2 QL has **no** `owl:sameAs` . . . but

```
owl:sameAs(URI("&npdv;company/{}",t5),URI("&brreg;{}",t2))
         :- company(t1,t2,t3,t4,t5,t6,...,t10)
```

could be supported because (a) IRI templates are incompatible and (b)

```
CREATE TABLE ‘company‘ (
   ...
   ‘cmpNpdidCompany‘ int(11) NOT NULL,
   ...
   PRIMARY KEY (‘cmpNpdidCompany‘)
);
```

- functional properties **cannot** be supported because AC$_0$ $\subsetneq$ P

Database Constraints: Equality-Generating Dependencies

$$\forall x_1, \ldots, x_n \left(\varphi(x_1, \ldots, x_n) \rightarrow (x_i = x_j)\right)$$

Database Constraints: Equality-Generating Dependencies

$$\forall x_1, \ldots, x_n \, (\varphi(x_1, \ldots, x_n) \rightarrow (x_i = x_j))$$

```
CREATE TABLE ‘wellbore_core‘ (
  ‘wlbName‘ varchar(60),
  ‘wlbCoreNumber‘ int(11) NOT NULL,
  ...
  ‘wlbNpdidWellbore‘ int(11) NOT NULL,
  ...
  ‘wellbore_core_id‘ bigint(20) unsigned NOT NULL AUTO_INCREMENT,
  PRIMARY KEY (‘wellbore_core_id‘,‘wlbNpdidWellbore‘,‘wlbCoreNumber‘),



  UNIQUE KEY ‘wellbore_core_id‘ (‘wellbore_core_id‘),



  ...
);
```

Database Constraints: Equality-Generating Dependencies

$$\forall x_1,\ldots,x_n\,(\varphi(x_1,\ldots,x_n) \rightarrow (x_i = x_j))$$

```
CREATE TABLE ‘wellbore_core‘ (
  ‘wlbName‘ varchar(60),
  ‘wlbCoreNumber‘ int(11) NOT NULL,
  ...
  ‘wlbNpdidWellbore‘ int(11) NOT NULL,
  ...
  ‘wellbore_core_id‘ bigint(20) unsigned NOT NULL AUTO_INCREMENT,
  PRIMARY KEY (‘wellbore_core_id‘,‘wlbNpdidWellbore‘,‘wlbCoreNumber‘),
```

$$\forall \mathtt{t1},\ldots,\mathtt{t12},\mathtt{t1}',\ldots,\mathtt{t12}' \; \big(\mathtt{wellbore\_core}(\mathtt{t1},\ldots,\mathtt{t12}) \wedge \mathtt{wellbore\_core}(\mathtt{t1}',\ldots,\mathtt{t12}') \wedge (\mathtt{t12} = \mathtt{t12}') \wedge (\mathtt{t9} = \mathtt{t9}') \wedge (\mathtt{t2} = \mathtt{t2}') \rightarrow (\mathtt{t}i = \mathtt{t}i')\big)$$, for **all** i

```
  UNIQUE KEY ‘wellbore_core_id‘ (‘wellbore_core_id‘),




  ...
);
```

Database Constraints: Equality-Generating Dependencies

$$\forall x_1,\dots,x_n\,(\varphi(x_1,\dots,x_n) \rightarrow (x_i = x_j))$$

```
CREATE TABLE ‘wellbore_core‘ (
  ‘wlbName‘ varchar(60),
  ‘wlbCoreNumber‘ int(11) NOT NULL,
  ...
  ‘wlbNpdidWellbore‘ int(11) NOT NULL,
  ...
  ‘wellbore_core_id‘ bigint(20) unsigned NOT NULL AUTO_INCREMENT,
  PRIMARY KEY (‘wellbore_core_id‘,‘wlbNpdidWellbore‘,‘wlbCoreNumber‘),
```

$$\forall \mathtt{t1},\dots,\mathtt{t12},\mathtt{t1}',\dots,\mathtt{t12}' \big(\mathtt{wellbore\_core}(\mathtt{t1},\dots,\mathtt{t12}) \land \mathtt{wellbore\_core}(\mathtt{t1}',\dots,\mathtt{t12}') \land (\mathtt{t12} = \mathtt{t12}') \land (\mathtt{t9} = \mathtt{t9}') \land (\mathtt{t2} = \mathtt{t2}') \rightarrow (\mathtt{t}i = \mathtt{t}i')\big), \text{ for all } i$$

```
  UNIQUE KEY ‘wellbore_core_id‘ (‘wellbore_core_id‘),
```

$$\forall \mathtt{t1},\dots,\mathtt{t12},\mathtt{t1}',\dots,\mathtt{t12}' \big(\mathtt{wellbore\_core}(\mathtt{t1},\dots,\mathtt{t12}) \land \mathtt{wellbore\_core}(\mathtt{t1}',\dots,\mathtt{t12}') \land (\mathtt{t12} = \mathtt{t12}') \rightarrow (\mathtt{t}i = \mathtt{t}i')\big), \text{ for all } i$$

```
  ...
);
```

Database Constraints: Tuple-Generating Dependencies

$$\forall x_1, \ldots, x_n \left(\varphi(x_1, \ldots, x_n) \rightarrow \exists y_1, \ldots, y_m \psi(x_1, \ldots, x_n, y_1, \ldots, y_m)\right)$$

Database Constraints: Tuple-Generating Dependencies

$$\forall x_1, \ldots, x_n \left(\varphi(x_1, \ldots, x_n) \rightarrow \exists y_1, \ldots, y_m \psi(x_1, \ldots, x_n, y_1, \ldots, y_m)\right)$$

```
CREATE TABLE ‘licence_task‘ (
  ‘prlName‘ varchar(50) NOT NULL,
  ...
  ‘prlNpdidLicence‘ int(11) NOT NULL,
  ‘prlTaskID‘ int(11) NOT NULL,
  ‘prlTaskRefID‘ int(11) DEFAULT NULL,
  ...
  PRIMARY KEY (‘prlNpdidLicence‘,‘prlTaskID‘),
  CONSTRAINT ‘licence_task_ibfk_2‘ FOREIGN KEY (‘prlNpdidLicence‘)
                              REFERENCES ‘licence‘ (‘prlNpdidLicence‘),



  CONSTRAINT ‘licence_task_ibfk_3‘ FOREIGN KEY (‘prlTaskRefID‘, prlNpdidLicence)
                    REFERENCES ‘licence_task‘ (‘prlTaskID‘, prlNpdidLicence),
  ...
);
```

Database Constraints: Tuple-Generating Dependencies

$$\forall x_1, \ldots, x_n \, (\varphi(x_1, \ldots, x_n) \rightarrow \exists y_1, \ldots, y_m \psi(x_1, \ldots, x_n, y_1, \ldots, y_m))$$

```
CREATE TABLE ‘licence_task‘ (
  ‘prlName‘ varchar(50) NOT NULL,
  ...
  ‘prlNpdidLicence‘ int(11) NOT NULL,
  ‘prlTaskID‘ int(11) NOT NULL,
  ‘prlTaskRefID‘ int(11) DEFAULT NULL,
  ...
  PRIMARY KEY (‘prlNpdidLicence‘,‘prlTaskID‘),
  CONSTRAINT ‘licence_task_ibfk_2‘ FOREIGN KEY (‘prlNpdidLicence‘)
                              REFERENCES ‘licence‘ (‘prlNpdidLicence‘),
```

```
licence_task(t1,...,t12,p10,t14,...,t18)
          → ∃p1,..., p9, p11,...,p15 licence(p1,...,p9,p10,p11,...,p15)
```

```
  CONSTRAINT ‘licence_task_ibfk_3‘ FOREIGN KEY (‘prlTaskRefID‘, prlNpdidLicence)
                    REFERENCES ‘licence_task‘ (‘prlTaskID‘, prlNpdidLicence),
  ...
);
```

Database Constraints: Tuple-Generating Dependencies

$$\forall x_1, \ldots, x_n \left(\varphi(x_1, \ldots, x_n) \rightarrow \exists y_1, \ldots, y_m \psi(x_1, \ldots, x_n, y_1, \ldots, y_m)\right)$$

```
CREATE TABLE ‘licence_task‘ (
  ‘prlName‘ varchar(50) NOT NULL,
  ...
  ‘prlNpdidLicence‘ int(11) NOT NULL,
  ‘prlTaskID‘ int(11) NOT NULL,
  ‘prlTaskRefID‘ int(11) DEFAULT NULL,
  ...
  PRIMARY KEY (‘prlNpdidLicence‘,‘prlTaskID‘),
  CONSTRAINT ‘licence_task_ibfk_2‘ FOREIGN KEY (‘prlNpdidLicence‘)
                                       REFERENCES ‘licence‘ (‘prlNpdidLicence‘),
```

```
licence_task(t1,...,t12,p10,t14,...,t18)
        → ∃p1,..., p9, p11,...,p15 licence(p1,...,p9,p10,p11,...,p15)
```

```
  CONSTRAINT ‘licence_task_ibfk_3‘ FOREIGN KEY (‘prlTaskRefID‘, prlNpdidLicence)
                    REFERENCES ‘licence_task‘ (‘prlTaskID‘, prlNpdidLicence),
```

```
licence_task(t1,...,t12,p13,t14,p14,t16,...,t18)
        → ∃p1,..., p12, p15,...,p18
                         licence_task(p1,...,p12,p13,p14,p15,...,p18)
```

Using SQO to Optimise User-Defined Mappings

```
npdv:explorationWellboreForField(URI("&npdv;wellbore/{}",t70),URI("&npdv;field/{}",t88))
    :- wellbore_exploration_all(t1,...,t69,t70,t71,t88,t73,...,t79),
       field(t80,...,t87,t88,t89,...,t95)
```

Using SQO to Optimise User-Defined Mappings

```
npdv:explorationWellboreForField(URI("&npdv;wellbore/{}",t70),URI("&npdv;field/{}",t88))
    :- wellbore_exploration_all(t1,...,t69,t70,t71,t88,t73,...,t79),
       field(t80,...,t87,t88,t89,...,t95)
```

✚ **foreign key (FK):**

```
wellbore_exploration_all(t1,...,t69,t70,t71,p9,t73,...,t79)
      → ∃p1,...,p8,p10,...,p16 field(p1,...,p8,p9,p10,...,p16)
```

Using SQO to Optimise User-Defined Mappings

```
npdv:explorationWellboreForField(URI("&npdv;wellbore/{}",t70),URI("&npdv;field/{}",t88))
    :- wellbore_exploration_all(t1,...,t69,t70,t71,t88,t73,...,t79),
       field(t80,...,t87,t88,t89,...,t95)
```

+ **foreign key (FK):**

```
wellbore_exploration_all(t1,...,t69,t70,t71,p9,t73,...,t79)
    → ∃p1,...,p8,p10,...,p16 field(p1,...,p8,p9,p10,...,p16)
```

↓

```
npdv:explorationWellboreForField(URI("&npdv;wellbore/{}",t70),URI("&npdv;field/{}",t88))
    :- wellbore_exploration_all(t1,...,t69,t70,t71,t88,t73,...,t79)
```

Using SQO to Optimise User-Defined Mappings

```
npdv:explorationWellboreForField(URI("&npdv;wellbore/{}",t70),URI("&npdv;field/{}",t88))
    :- wellbore_exploration_all(t1,...,t69,t70,t71,t88,t73,...,t79),
       field(t80,...,t87,t88,t89,...,t95)
```

+ **foreign key (FK):**

```
wellbore_exploration_all(t1,...,t69,t70,t71,p9,t73,...,t79)
     → ∃p1,...,p8,p10,...,p16 field(p1,...,p8,p9,p10,...,p16)
```

↓

```
npdv:explorationWellboreForField(URI("&npdv;wellbore/{}",t70),URI("&npdv;field/{}",t88))
    :- wellbore_exploration_all(t1,...,t69,t70,t71,t88,t73,...,t79)
```

	one DB atom	two DB atoms	total
without SQO	**979**	**211**	1190
with FK-SQO	**1058**	**132**	
	+79		

Using OR to Optimise T-mappings

```
npdv:Survey(URI("&npdv;survey/{}",t1))
  :- seis_acquisition(t1,t2,...,t5,"Grunnundersøkelser",t7,...,t18)
  :- seis_acquisition(t1,t2,...,t5,"Ordinær seismisk undersøkelse",t7,...,t18)
  :- seis_acquisition(t1,t2,...,t5,"Traseundersøkelser",t7,...,t18)
  :- seis_acquisition(t1,t2,...,t5,"Borestedundersøkelse / sitesurvey",t7,...,t18)
  :- seis_acquisition(t1,t2,...,t5,"Annen undersøkelse",t7,...,t18)
  :- seis_acquisition(t1,t2,...,t5,"Havbunnseismisk undersøkelse",t7,...,t18)
  :- seis_acquisition(t1,t2,...,t5,"Elektromagnetisk undersøkelse",t7,...,t18)
```

Using OR to Optimise T-mappings

```
npdv:Survey(URI("&npdv;survey/{}",t1))
      :- seis_acquisition(t1,t2,...,t5,"Grunnundersøkelser",t npdv:GroundSurvey
      :- seis_acquisition(t1,t2,...,t5,"Ordinær seismisk  npdv:RegularSeismicSurvey
      :- seis_acquisition(t1,t2,...,t5,"Traseundersøkelser",t npdv:SeismicSurvey
      :- seis_acquisition(t1,t  ,t5,"Borestedundersøkelse / npdv:RouteSurvey ..,t18)
      :- seis_acquisition(t1,t2,...,t5,"Annen undersøkelse",t7 npdv:OtherSurvey
      :- seis_acquisition(t1,t2,...,t5,"Havbunnseismisk un npdv:SeabedSeismicSurvey
      :- seis_acquisition(t1,t2,...,t5,"Elektromagnetisk  npdv:ElectromagneticSurvey
```

Using OR to Optimise T-mappings

```
npdv:Survey(URI("&npdv;survey/{}",t1))
      :- seis_acquisition(t1,t2,...,t5,"Grunnundersøkelser",t npdv:GroundSurvey
      :- seis_acquisition(t1,t2,...,t5,"Ordinær seismisk  npdv:RegularSeismicSurvey
      :- seis_acquisition(t1,t2,...,t5,"Traseundersøkelser",t npdv:SeismicSurvey
      :- seis_acquisition(t1,t     ,t5,"Borestedundersøkelse / npdv:RouteSurvey ..,t18)
      :- seis_acquisition(t1,t2,...,t5,"Annen undersøkelse",t7 npdv:OtherSurvey
      :- seis_acquisition(t1,t2,...,t5,"Havbunnseismisk un npdv:SeabedSeismicSurvey
      :- seis_acquisition(t1,t2,...,t5,"Elektromagnetisk  npdv:ElectromagneticSurvey
```

```
npdv:Survey(URI("&npdv;survey/{}",t1))
     :- seis_acquisition(t1,t2,...,t5,t6,t7,...,t18),
        OR(EQ(t6,"Grunnundersøkelser"),
          OR(EQ(t6,"Ordinær seismisk undersøkelse"),
            OR(EQ(t6,"Traseundersøkelser"),
              OR(EQ(t6,"Borestedundersøkelse / sitesurvey"),
                OR(EQ(t6,"Annen undersøkelse"),
                  OR(EQ(t6,"Havbunnseismisk undersøkelse"),
                    EQ(t6,"Elektromagnetisk undersøkelse")))))))
```

Using FK-SQO to Optimise T-mappings

```
npdv:Company(URI("&npdv;company/{}",t5))
      :- company(t1,...,t4,t5,t6,...,t10)

npdv:Company(URI("&npdv;company/{}",t8))
      :- facility_moveable(t1,...,t7,t8,t9,t10)
```

Using FK-SQO to Optimise T-mappings

```
npdv:Company(URI("&npdv;company/{}",t5))
      :- company(t1,...,t4,t5,t6,...,t10)

npdv:Company(URI("&npdv;company/{}",t8))
      :- facility_moveable(t1,...,t7,t8,t9,t10)
```

✚ **foreign key (FK):**

```
facility_moveable(t1,...,t7,p5,t9,t10)
     → ∃p1,...,p4,p6,...,p10 company(p1,...,p4,p5,p6,...,p10)
```

Using FK-SQO to Optimise T-mappings

```
npdv:Company(URI("&npdv;company/{}",t5))
     :- company(t1,...,t4,t5,t6,...,t10)
npdv:Company(URI("&npdv;company/{}",t8))
     :- facility_moveable(t1,...,t7,t8,t9,t10)
```

+ **foreign key (FK):**

```
facility_moveable(t1,...,t7,p5,t9,t10)
     → ∃p1,...,p4,p6,...,p10 company(p1,...,p4,p5,p6,...,p10)
```

⇓

```
npdv:Company(URI("&npdv;company/{}",t5))
     :- company(t1,...,t4,t5,t6,...,t10)
```

Using FK-SQO to Optimise T-mappings

```
npdv:Company(URI("&npdv;company/{}",t5))
      :- company(t1,...,t4,t5,t6,...,t10)
npdv:Company(URI("&npdv;company/{}",t8))
      :- facility_moveable(t1,...,t7,t8,t9,t10)
```

```
npdv:currentResponsibleCompany rdfs:range npdv:Company
```

+ foreign key (FK):

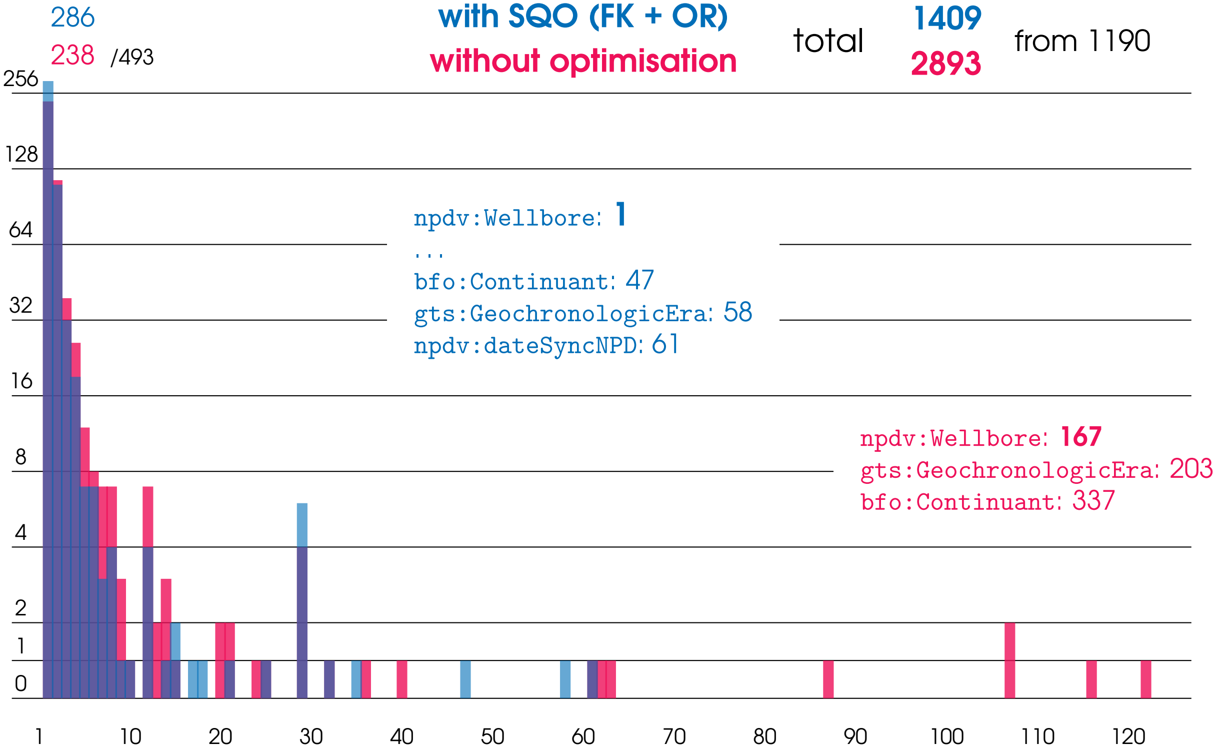
```
facility_moveable(t1,...,t7,p5,t9,t10)
     → ∃p1,...,p4,p6,...,p10 company(p1,...,p4,p5,p6,...,p10)
```

↓

```
npdv:Company(URI("&npdv;company/{}",t5))
      :- company(t1,...,t4,t5,t6,...,t10)
```

NB: this optimisation eliminates redundant rules that are introduced by 'applying' the ontology

Number of Rules in T-mappings: NPD FactPages



with SQO (FK + OR)

without optimisation

total 1409 / 2893 from 1190

286

238 /493

npdv:Wellbore: **1**

...

bfo:Continuant: 47

gts:GeochronologicEra: 58

npdv:dateSyncNPD: 61

npdv:Wellbore: **167**

gts:GeochronologicEra: 203

bfo:Continuant: 337

Using delRling in Unfolding

Q9:

```
?p a npdv:FieldYearlyProduction;
     npdv:producedOil ?oil.
```

Using deIRling in Unfolding

Q9:

```
?p a npdv:FieldYearlyProduction;
     npdv:producedOil ?oil.
```

+

```
npdv:FieldYearlyProduction(URI("&npdv;field/{}/production/{}",t9,t2))
    :- field_production_yearly(t1,t2,t3,...,t8,t9)

npdv:producedOil(URI("&npdv;ncs/production/{}",t1),xsd:decimal(t2))
    :- field_production_totalt_NCS_year(t1,t2,t3,...,t7)
npdv:producedOil(URI("&npdv;ncs/production/{}/{}",t1,t2),xsd:decimal(t3))
    :- field_production_totalt_NCS_month(t1,t2,t3,t4,...,t8)
npdv:producedOil(URI("&npdv;field/{}/production/{}",t9,t2),xsd:decimal(t3))
    :- field_production_yearly(t1,t2,t3,t4,...,t8,t9)
npdv:producedOil(URI("&npdv;field/{}/production/{}/{}",t10,t2,t3),xsd:decimal(t4))
    :- field_production_monthly(t1,t2,t3,t4,...,t9,t10)
```

Using deIRIing in Unfolding

Q9:

```
?p a npdv:FieldYearlyProduction;
     npdv:producedOil ?oil.
```

+

```
npdv:FieldYearlyProduction(URI("&npdv;field/{}/production/{}",t9,t2))
     :- field_production_yearly(t1,t2,t3,...,t8,t9)

npdv:producedOil(URI("&npdv;ncs/production/{}",t1),xsd:decimal(t2))
     :- field_production_totalt_NCS_year(t1,t2,t3,...,t7)
npdv:producedOil(URI("&npdv;ncs/production/{}/{}",t1,t2),xsd:decimal(t3))
     :- field_production_totalt_NCS_month(t1,t2,t3,t4,...,t8)
npdv:producedOil(URI("&npdv;field/{}/production/{}",t9,t2),xsd:decimal(t3))
     :- field_production_yearly(t1,t2,t3,t4,...,t8,t9)
npdv:producedOil(URI("&npdv;field/{}/production/{}/{}",t10,t2,t3),xsd:decimal(t4))
     :- field_production_monthly(t1,t2,t3,t4,...,t9,t10)
```

disjointness

```
Q9(URI("&npdv;ncs/production/{}/{}",t9,t2), t3′)
     :- field_production_yearly(t1,t2,t3,...,t8,t9),
        field_production_yearly(t1′,t2,t3′,t4′,...,t8′,t9)
```

Using Primary and Alternative Keys in Unfolding

```
Q9′(URI("&npdv;ncs/production/{}/{}",t9,t2), t3′, t4″)
      :- field_production_yearly(t1,t2,t3,...,t8,t9),
         field_production_yearly(t1′,t2,t3′,t4′,...,t8′,t9)
         field_production_yearly(t1″,t2,t3″,t4″,...,t8″,t9)
                                                                    + npdv:producedGas
```

Using Primary and Alternative Keys in Unfolding

```
Q9′(URI("&npdv;ncs/production/{}/{}",t9,t2), t3′, t4″)
      :- field_production_yearly(t1,t2,t3,...,t8,t9),
         field_production_yearly(t1′,t2,t3′,t4′,...,t8′,t9)
         field_production_yearly(t1″,t2,t3″,t4″,...,t8″,t9)
                                                          + npdv:producedGas
```

+

```
CREATE TABLE ‘field_production_yearly‘ (
  ‘prfInformationCarrier‘ varchar(40) NOT NULL,
  ‘prfYear‘ int(11) NOT NULL,
  ‘prfPrdOilNetMillSm3‘ decimal(13,6) NOT NULL,
  ‘prfPrdGasNetBillSm3‘ decimal(13,6) NOT NULL,
  ...
  ‘prfNpdidInformationCarrier‘ int(11) NOT NULL,
  PRIMARY KEY (‘prfNpdidInformationCarrier‘,‘prfYear‘)
);
```

Using Primary and Alternative Keys in Unfolding

```
Q9′(URI("&npdv;ncs/production/{}/{}",t9,t2), t3′, t4″)
      :- field_production_yearly(t1, t2 ,t3,...,t8, t9 ),
         field_production_yearly(t1′, t2 , t3′ ,t4′,...,t8′, t9 )
         field_production_yearly(t1″, t2 ,t3″, t4″ ,...,t8″, t9 )
```

+ npdv:producedGas

+

```
CREATE TABLE ‘field_production_yearly‘ (
  ‘prfInformationCarrier‘ varchar(40) NOT NULL,
  ‘prfYear‘ int(11) NOT NULL,
  ‘prfPrdOilNetMillSm3‘ decimal(13,6) NOT NULL,
  ‘prfPrdGasNetBillSm3‘ decimal(13,6) NOT NULL,
  ...
  ‘prfNpdidInformationCarrier‘ int(11) NOT NULL,
  PRIMARY KEY (‘prfNpdidInformationCarrier‘,‘prfYear‘)
);
```

↓

```
Q9′(URI("&npdv;ncs/production/{}/{}",t9,t2), t3 , t4 )
      :- field_production_yearly(t1, t2 , t3 , t4 ,t5,...,t8, t9 )
```

NB: this optimisation eliminates the redundant self-joins that are introduced by reification

Conclusions

- T-mappings compile (big parts of) OWL 2 QL ontologies into mappings
  (domain and range constraints, concept and role hierarchies)
- can be optimised **offline**
- give answers to all SPARQL **triple patterns**
  rewriting required for complex class expressions only (tree-shaped CQs)

Conclusions

- T-mappings compile (big parts of) OWL 2 QL ontologies into mappings
  (domain and range constraints, concept and role hierarchies)

- can be optimised **offline**

- give answers to all SPARQL **triple patterns**
  rewriting required for complex class expressions only (tree-shaped CQs)

- very few tree witnesses in real-world OBDA → polynomial-size rewritings

- integrity constraints and OR can significantly simplify T-mappings

Conclusions

- T-mappings compile (big parts of) OWL 2 QL ontologies into mappings (domain and range constraints, concept and role hierarchies)
- can be optimised **offline**
- give answers to all SPARQL **triple patterns**
  rewriting required for complex class expressions only (tree-shaped CQs)
- very few tree witnesses in real-world OBDA → polynomial-size rewritings
- integrity constraints and OR can significantly simplify T-mappings

→ efficient SQL queries over the data