# A Datalog+ RuleML 1.01 Architecture for Rule-Based Data Access in Ecosystem Research

**(Long version: cs.unb.ca/~boley/talks/RulesOBDA.pdf)**


Harold Boley (UNB, RuleML)

Rolf Grütter (WSL)

Gen Zou (UNB)

Tara Athan (RuleML)

Sophia Etzold (WSL)



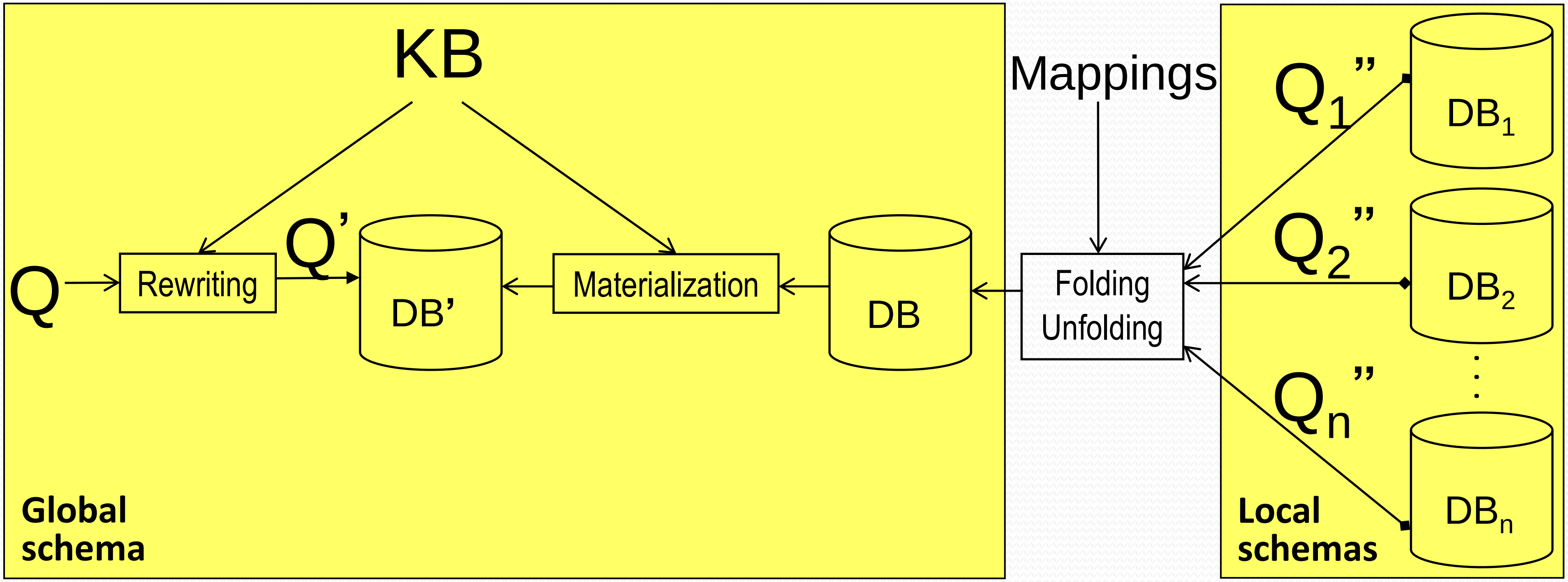
The 8th International Web Rule Symposium (RuleML 2014)

August 18-20, 2014, Prague, Czech Republic

# What is **K**nowledge-**B**ased **D**ata **A**ccess?

- KBDA applies AI / Semantic Technologies to databases
- Founded on **(logical) knowledge** representation for:
  - **Ontology**-Based Data Access (OBDA), e.g. data integration/federation and query optimization
  - **Rule**-Based Data Access (RBDA), e.g. Datalog / deductive databases and query answering
- **Knowledge Base** as generalized **global** schema for data in **local** (e.g., relational or graph) DBs
- KB module amplifies data storage & query execution of **distributed, heterogeneous** (No)SQL DBs
- Provides **multi-purpose** knowledge level for data

# Preview: Unified Architecture



# Why Knowledge-Based Data Access?

- Domain knowledge utilized to deal with data torrent
  - Domain experts conceptually *fold* data / *unfold* queries via **Mappings** defined <u>with</u> IT (SQL, SPARQL, …) experts
  - User concepts are captured in **Knowledge Base** for *domain-enriched* database materialization / querying <u>without</u> IT experts
  - **Engines** use KB to deduce answers implicit in DBs
  - **Analytics** enabled by queries exploring hypotheses
- KB as major organizational resource also for, e.g.:
  - Data validation (consistency, completeness, …)
  - Schema-level query answering (even without DBs)

# RBDA Realizes Uniform KBDA – 1 of 3: Queries as Rules

1. a) A conjunctive ***query*** is a special Datalog **rule** whose body can be *rewritten* (see 2.) and *unfolded* (see 3.), and whose head instantiates the distinguished answer variables of the body
   b) KBDA ontologies beyond RDF Schema (RDFS) often permit **Boolean** conjunctive queries corresponding to ***integrity* rules**
2. …
3. …

# RBDA Realizes Uniform KBDA – 2 of 3: KBs as Rules

1. …
2. KBDA ***KB*** supports, e.g., query *rewriting* through global-schema-level reasoning, including with RDFS **taxonomies** or Datalog **rule** axioms, and DL-Lite (OWL 2 QL) or (head-)existential **rules**; KBDA **rules** also permit Description Logic Programs (OWL 2 RL), Datalog$^{\pm}$, and Disjunctive Datalog. [Semantics of ***ontology languages*** customizable for expressivity and efficiency requirements by adding/deleting **rules** (SPIN)]
3. …

# RBDA Realizes Uniform KBDA – 3 of 3: Mappings as Rules

1. …
2. …
3. KBDA data integration is centered on Global-As-View (GAV) ***mappings***, which are Datalog **rules** for, e.g., *unfolding* each global head predicate to (a join, i.e. conjunction, of) local body predicates

# Example: Forest/Orchard Knowledge

# EntityWithTree KB: Named Root Class (1)

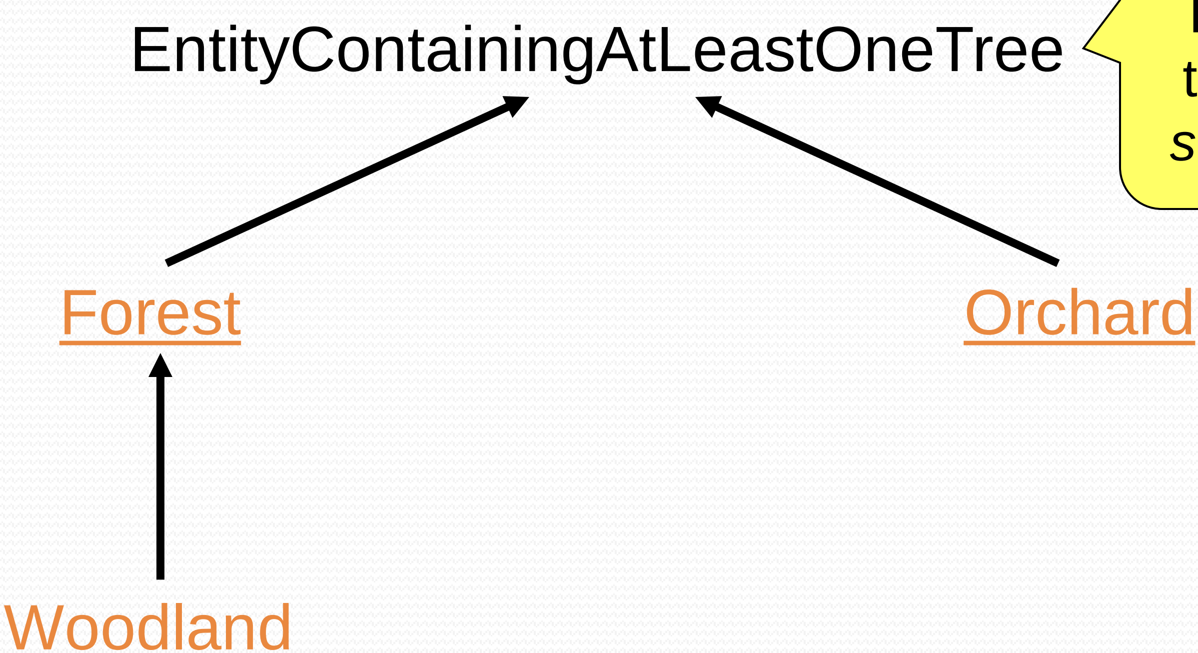
***Subsumption axioms (in higher-order rule syntax):***

EntityContainingAtLeastOneTree ← Forest.

EntityContainingAtLeastOneTree ← Orchard.

Forest ← Woodland.

"←" is taxonomy-style 'subsumes' infix



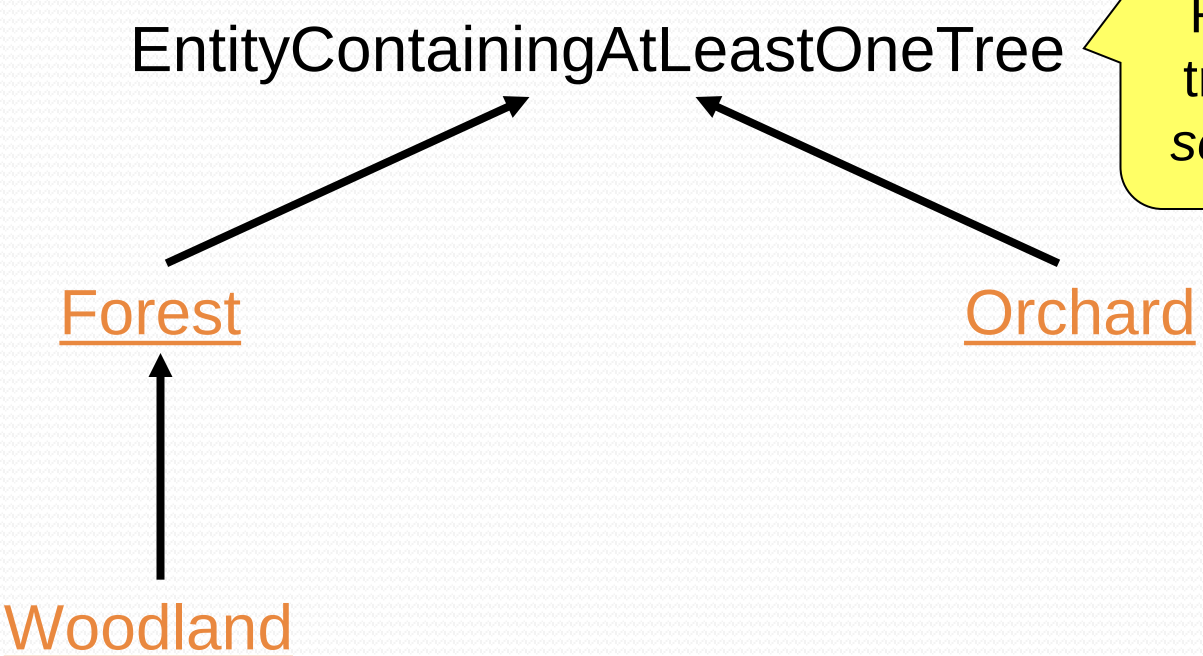

Root of taxonomy tree of tree-containing entities to *see the forest for the trees*

# EntityWithTree KB: Named Root Class (2)

***Subsumption axioms (in higher-order rule syntax):***

EntityContainingAtLeastOneTree :- Forest.
EntityContainingAtLeastOneTree :- Orchard.
Forest :- Woodland.

“ **:-** ” is rule-style ‘if’ infix




Root of taxonomy tree of tree-containing entities to *see the forest for the trees*

# EntityWithTree KB: Constructed Root Class

***Subsumption axioms (in higher-order rule syntax):***

∃contains.Tree :- Forest.

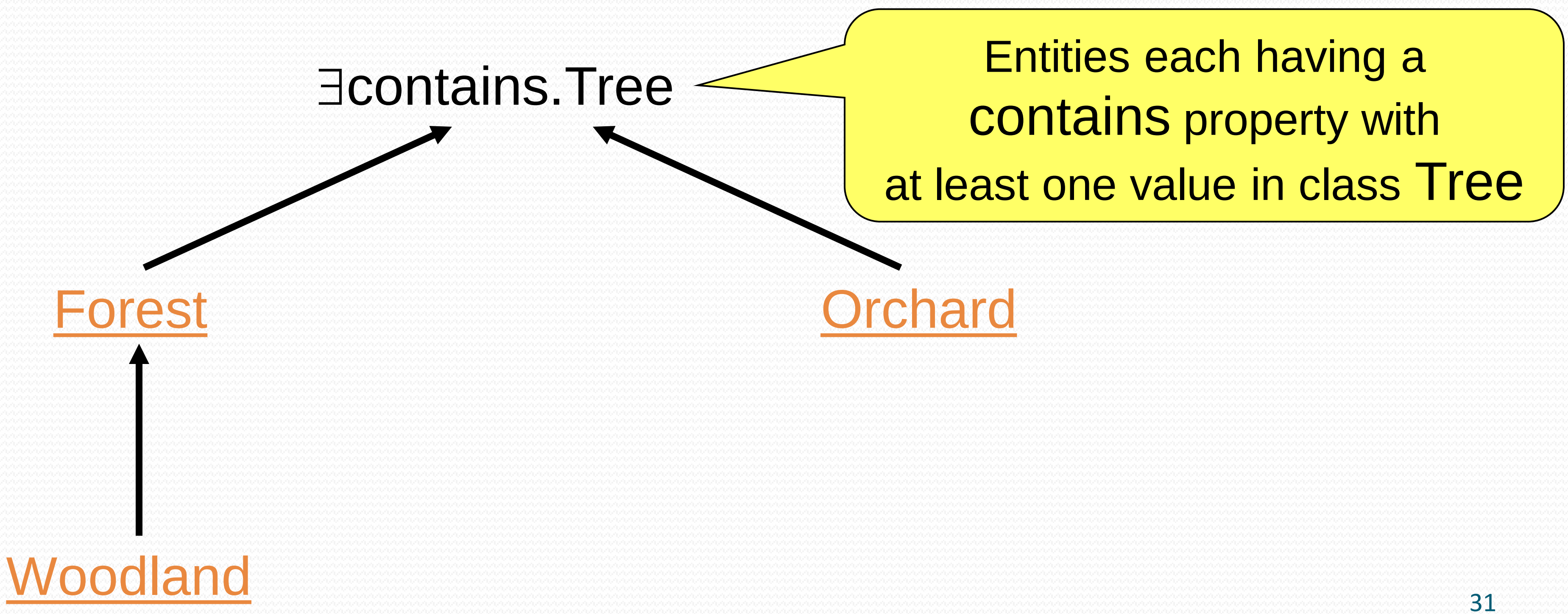
∃contains.Tree :- Orchard.

Forest :- Woodland.

Cf. ontology-style (description logic) axioms:

$\exists$contains.Tree $\sqsupseteq$ Forest

$\exists$contains.Tree $\sqsupseteq$ Orchard



# Three Dimensions of KBDA$_s$: R,Q,m

# RBDQ$_{mediator}$

R

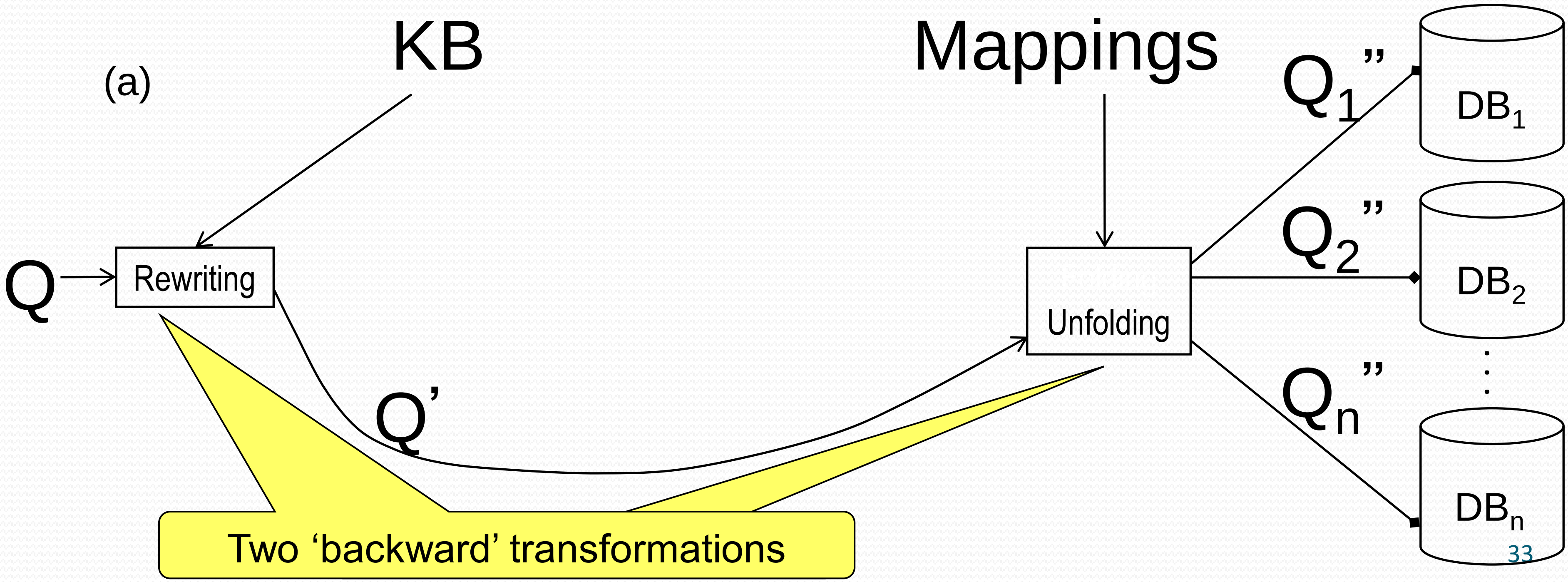
u

l

e -Based Data Querying

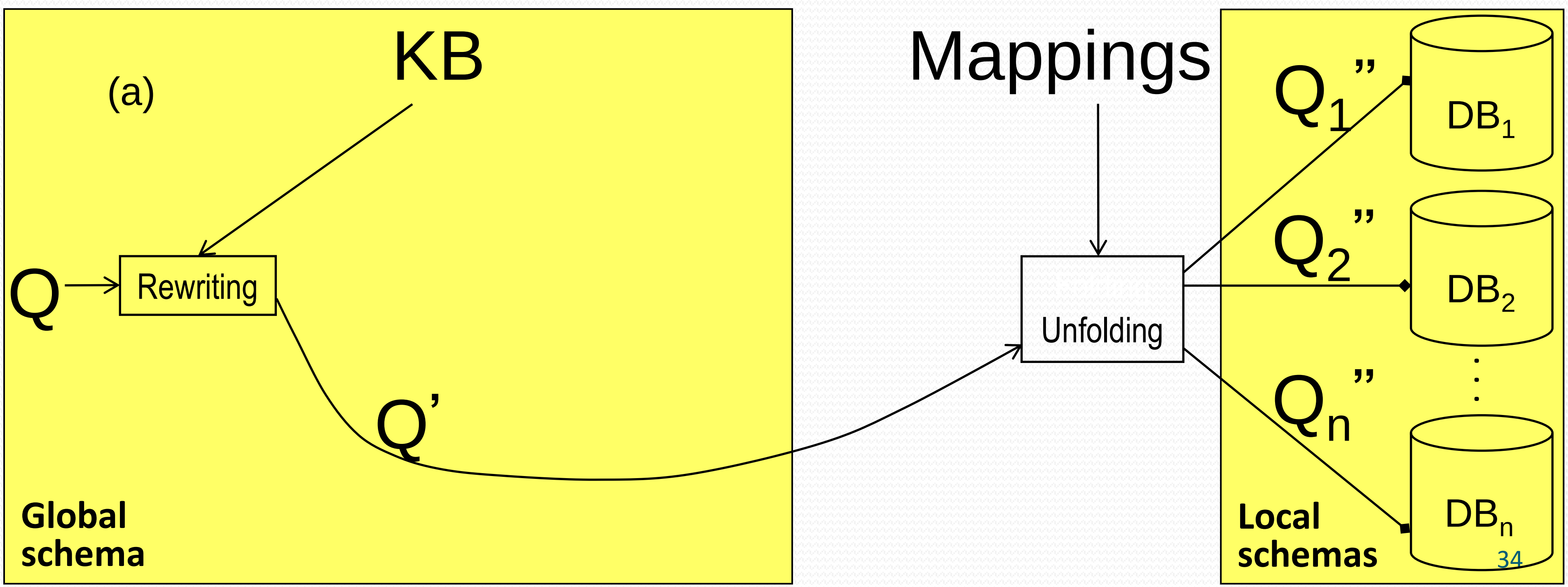
# Query Rewriting and Unfolding

- Mediator strategy uses:
  KB to *rewrite* Q to Q' and Mappings (**rules**) to *unfold* Q' to $Q_i$''
- KB can be **ontology**, e.g. in OWL 2 QL (DL-Lite), or **rules**
- Abstract (relational/graph/…) queries $Q_i$'' ♦-grounded (to SQL/SPARQL/…) for $DB_i$
- Each (relational/graph/…) database $DB_i$ left as original; answers at ♦



# Query Rewriting and Unfolding

- Mediator strategy uses:
  KB to *rewrite* Q to Q' and Mappings (**rules**) to *unfold* Q' to $Q_i$''
- KB can be **ontology**, e.g. in OWL 2 QL (DL-Lite), or **rules**
- Abstract (relational/graph/…) queries $Q_i$'' ♦-grounded (to SQL/SPARQL/…) for $DB_i$
- Each (relational/graph/…) database $DB_i$ left as original; answers at ♦



# Query Rewriting Use of KB

- In Information Retrieval:
  **Query expansion**
  Intuitive idea of **query rewriting**
  - With increased *recall*
  - Without loss of *precision*
- From Logic Programming (for Horn expressivity):
  Can use **resolution method**
  for KB-enrichment of a given (conjunctive) query
  with expanded (conjunctive) queries
  so that, for any DB,
  the answers to the enriched queries no longer using the KB
  are the same as the answers to the original query using the KB

# Ontology-to-Rule Clausification of EntityWithTree KB Omitting Orchard

***Description Logic subsumptions (as higher-order rules):***

∃contains.Tree :- Forest.

Forest :- Woodland.

Higher-order syntactic sugar for existential rule:
∃?y(contains(?x ?y) ∧ Tree(?y)) :- Forest(?x).

***Horn Logic rules (the first with conjunctive head):***

(contains(?x s(?x)) ∧ Tree(s(?x))) :- Forest(?x).

Forest(?x) :- Woodland(?x).

s is fresh Skolem function symbol

***Horn Logic rules (the first head split into two conjuncts):***

contains(?x s(?x)) :- Forest(?x).

Tree(s(?x)) :- Forest(?x).

Forest(?x) :- Woodland(?x).

# KB Rules Perform Rewriting of Given Query

**KB *with Horn rules* *(from above)* *for rewriting of query rules:***

contains(?x s(?x)) :- Forest(?x).

Tree(s(?x)) :- Forest(?x).

Forest(?x) :- Woodland(?x).

***Rewriting Datalog query rule to obtain extra query rules:***

**Q**: Given

**q(?z) :- contains(?z ?y) ∧ Tree(?y).**

q(?z):-Forest(?z)∧Tree(s(?z))

q(?z):-contains(?z s(?x))∧Forest(?x)

q(?z) :- Forest(?z) ∧ Forest(?z).

Expansion

**q(?z) :- Forest(?z).**

**q(?z) :- Woodland(?z).**

**Q’**: Given ∪ Expansion

# Query Unfolding Use of Mappings to Original Database Sources

- Datalog rules **bridging** between:
  - KB
  - Distributed DBs

Intuitive idea of **query unfolding**

- Use partial deduction-like unfolding (and simplification) of (conjunctive) KB queries to (conjunctive) *abstract DB queries*
  - Abstract relational queries grounded to SQL, abstract graph queries grounded to SPARQL, etc.
  - Lower-level optimization and execution by SQL, SPARQL, etc. engines
- Generated queries distributed over multiple DBs as indicated by "*source***.**" name prefixes

# Sample Mapping Rules to Three Local Data Sources

***Map KB predicates to locDB/regionDB tables for geo data:***

contains(?x ?y) :- locDB.cnt(?x ?kind ?y).
contains(?x ?y) :- regionDB.sub(?x ?r) ∧ locDB.cnt(?r ?kind ?y).

***Map KB predicates to locDB/ecoDB tables for forestry data:***

Tree(?t) :- locDB.cnt(?plot "tree" ?t).
Tree(?t) :- ecoDB.Plant(?plot "tree" ?size ?t).

Forest(?x) :- ecoDB.Habitat(?plot "forest" ?size ?x).

Woodland(?x) :- ecoDB.Habitat(?plot "wood" ?size ?x).

# Mapping Rules Perform Unfolding of Rewritten Queries

***Union of conjunctive queries as Datalog rules (rewritten):***

q(?z) :- contains(?z ?y) $\wedge$ Tree(?y).
q(?z) :- Forest(?z).
q(?z) :- Woodland(?z).

**Q'**: Given $\cup$ Expansion

***Unfolding above queries via mappings from previous slide:***

q(?z) :- locDB.cnt(?z ?kind ?y) $\wedge$ locDB.cnt(?plot "tree" ?y).

**q(?z) :- locDB.cnt(?z "tree" ?y).**

Simplification: Integrity rules

**q(?z) :- locDB.cnt(?z ?kind ?y) $\wedge$ ecoDB.Plant(?plot "tree" ?size ?y).**

q(?z) :- regionDB.sub(?z ?r) $\wedge$ locDB.cnt(?r ?kind ?y) $\wedge$ locDB.cnt(?plot "tree" ?y).

**q(?z) :- regionDB.sub(?z ?r) $\wedge$ locDB.cnt(?r "tree" ?y).**

**q(?z) :- regionDB.sub(?z ?r)$\wedge$locDB.cnt(?r ?kind ?y)$\wedge$ecoDB.Plant(?plot "tree" ?size ?y).**

**q(?z) :- ecoDB.Habitat(?plot "forest" ?size ?z).**

**q(?z) :- ecoDB.Habitat(?plot "wood" ?size ?z).**

$\mathbf{Q_i''}$: **Bold**-faced ($1 \le i \le 3$)

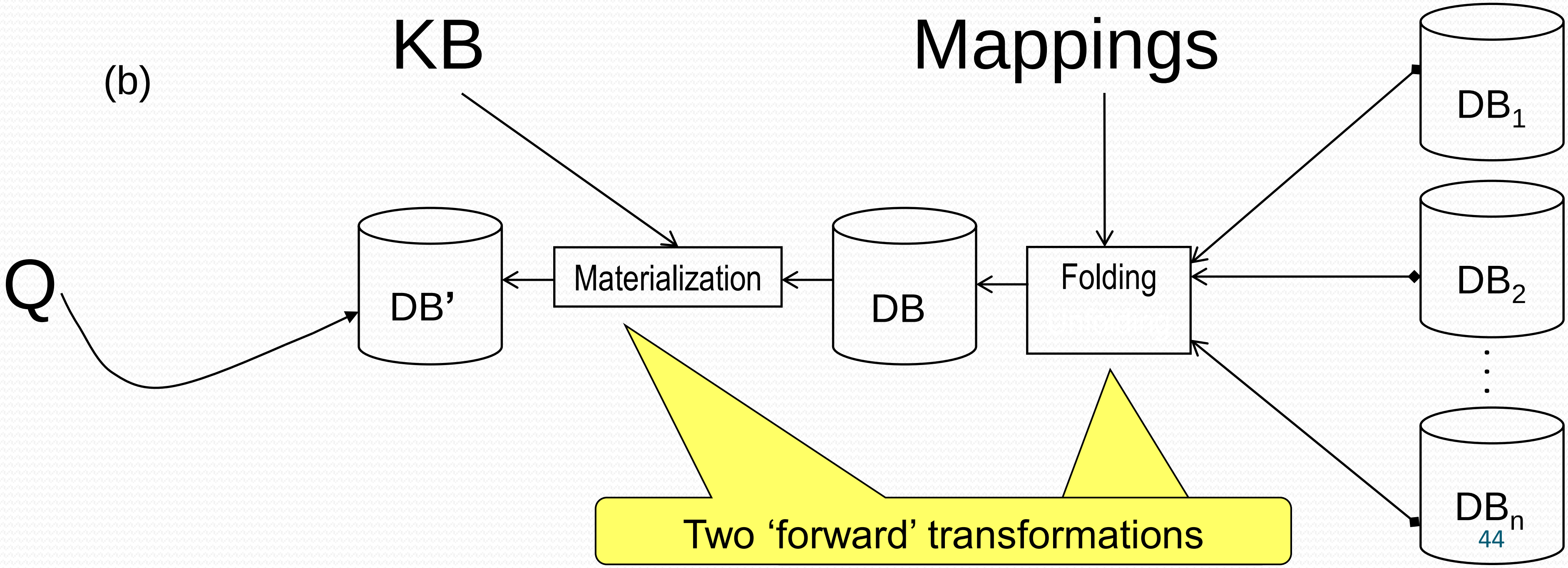
# Three Dimensions of $KBDA_s$: R,Q,w

RBDQ$_{warehouse}$

R
u
l
e
-Based Data Querying

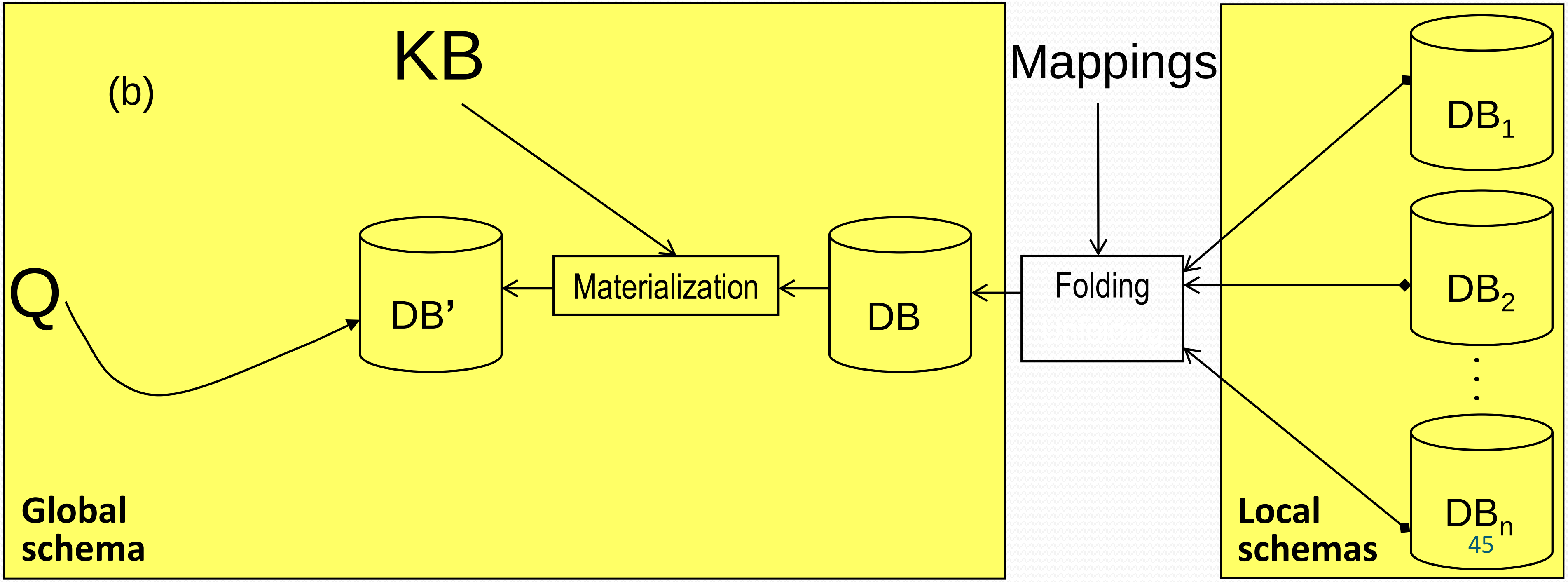
# Database Materialization after Folding

- Warehouse strategy uses:
  Mappings (same **rules** as for *unfolding*) to *fold* $DB_i$ to DB and KB to *materialize* Database DB to DB'
- KB can be **ontology**, e.g. in OWL 2 RL (DLP), or **rules**
- Query is left as original; answers at solid triangular arrow head



# Database Materialization after Folding

- Warehouse strategy uses:
  Mappings (same **rules** as for *unfolding*) to *fold* $DB_i$ to DB and KB to *materialize* Database DB to DB'
- KB can be **ontology**, e.g. in OWL 2 RL (DLP), or **rules**
- Query is left as original; answers at solid triangular arrow head

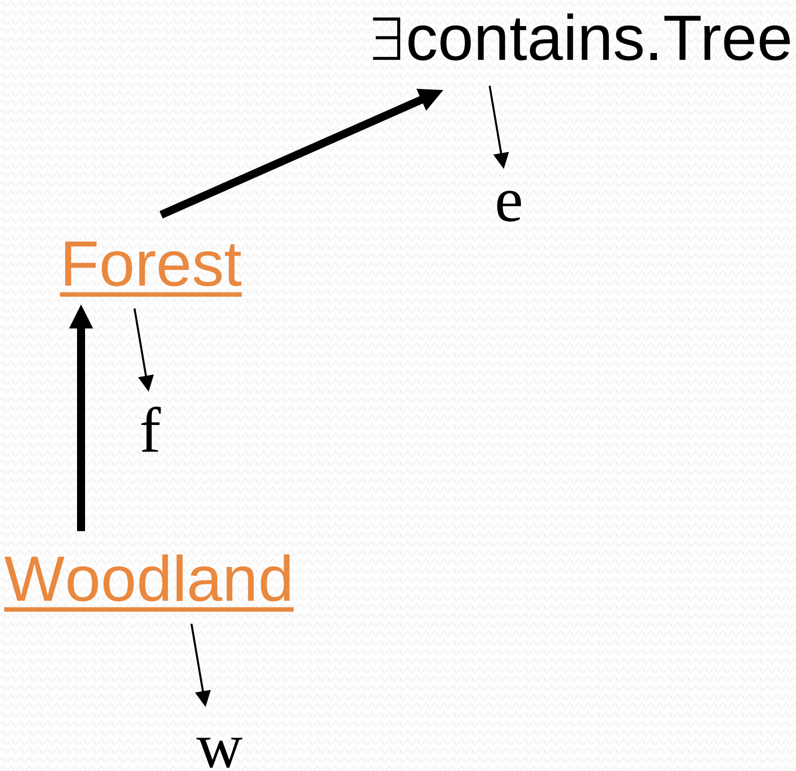


# EWT KB and DB: ‘Populated’ KB

***Subsumptions and Data (as higher-order rules and facts):***

∃contains.Tree :- Forest.
Forest :- Woodland.

∃contains.Tree(e).
Forest(f).
Woodland(w).



# Applying All KB Rules to All DB Facts (1)

***Subsumptions and Data (as higher-order rules and facts):***

∃contains.Tree :- Forest.

Forest :- Woodland.

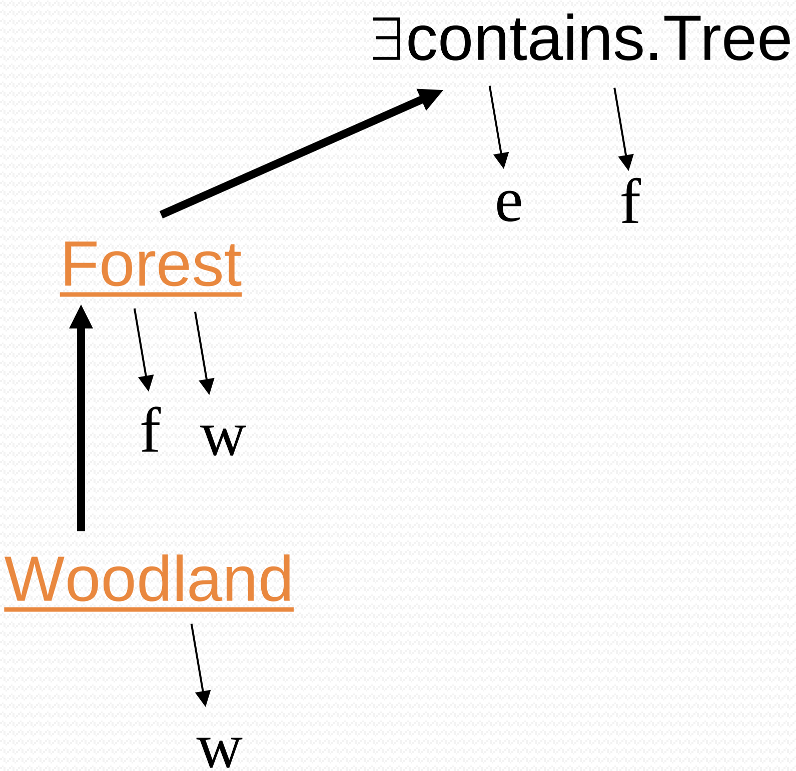
∃contains.Tree(e).

∃contains.Tree(f).

Forest(f).

Forest(w).

Woodland(w).

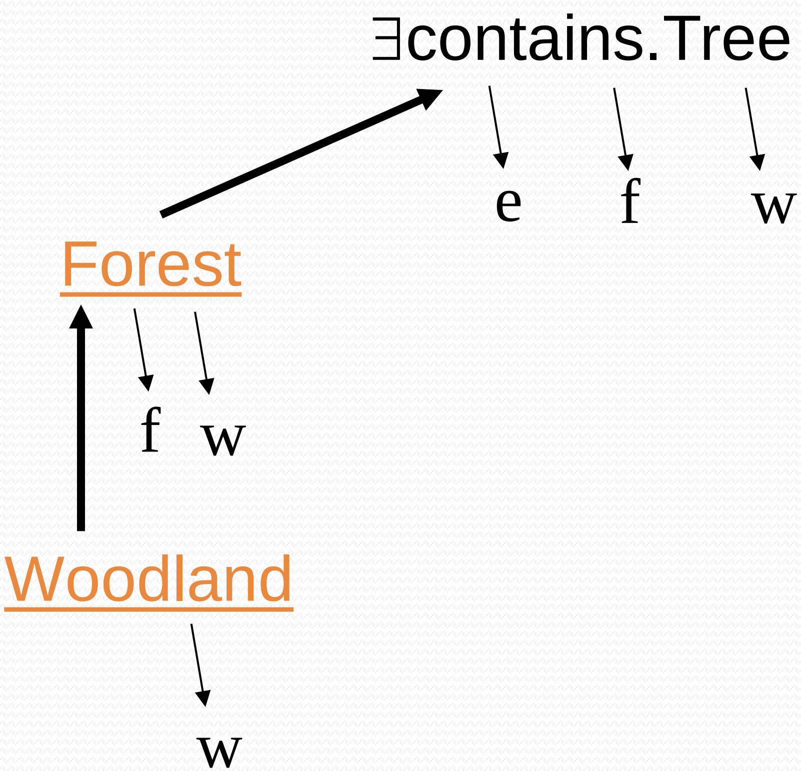


# Applying All KB Rules to All DB Facts (2)

***Subsumptions and Data (as higher-order rules and facts):***

∃contains.Tree :- Forest.

Forest :- Woodland.




∃contains.Tree(e).

∃contains.Tree(f).

∃contains.Tree(w).

Forest(f).

Forest(w).

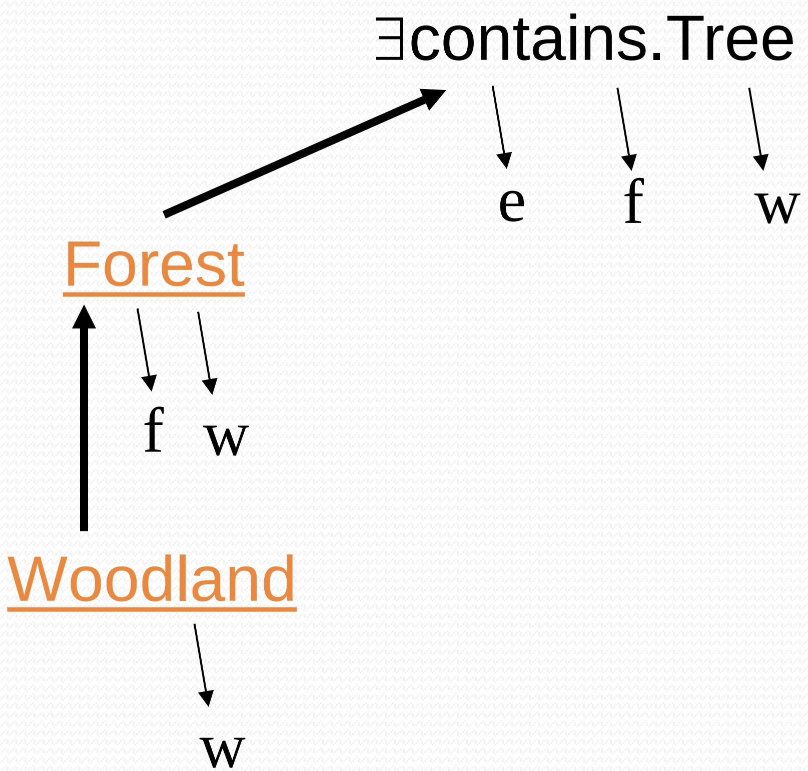
Woodland(w).

# Fixpoint with No New Rule-Derived Facts

***Subsumptions and Data (as higher-order rules and facts):***

∃contains.Tree :- Forest.

Forest :- Woodland.




∃contains.Tree(e).

∃contains.Tree(f).

∃contains.Tree(w).

Forest(f).

Forest(w).

Woodland(w).

# Three Dimensions of KBDA$_s$

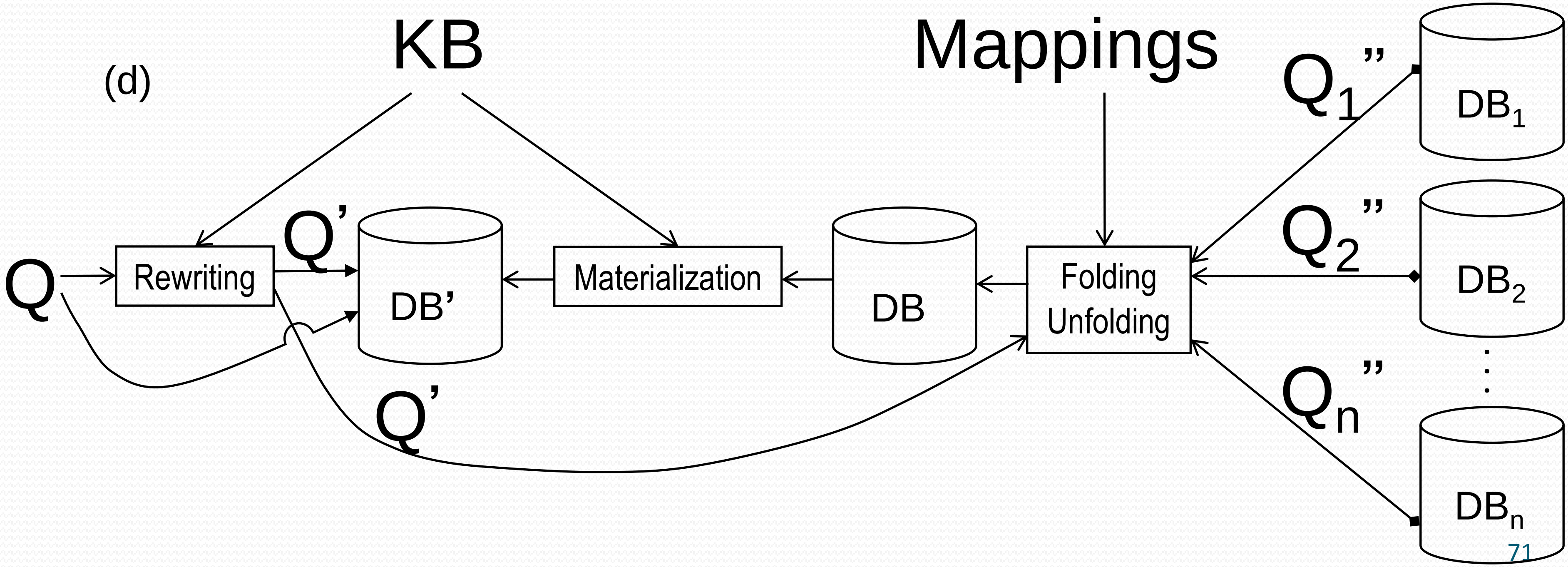
# KBDA$_{strategy}$

K
n
o
w
l
e
d
g
e

-Based Data Access

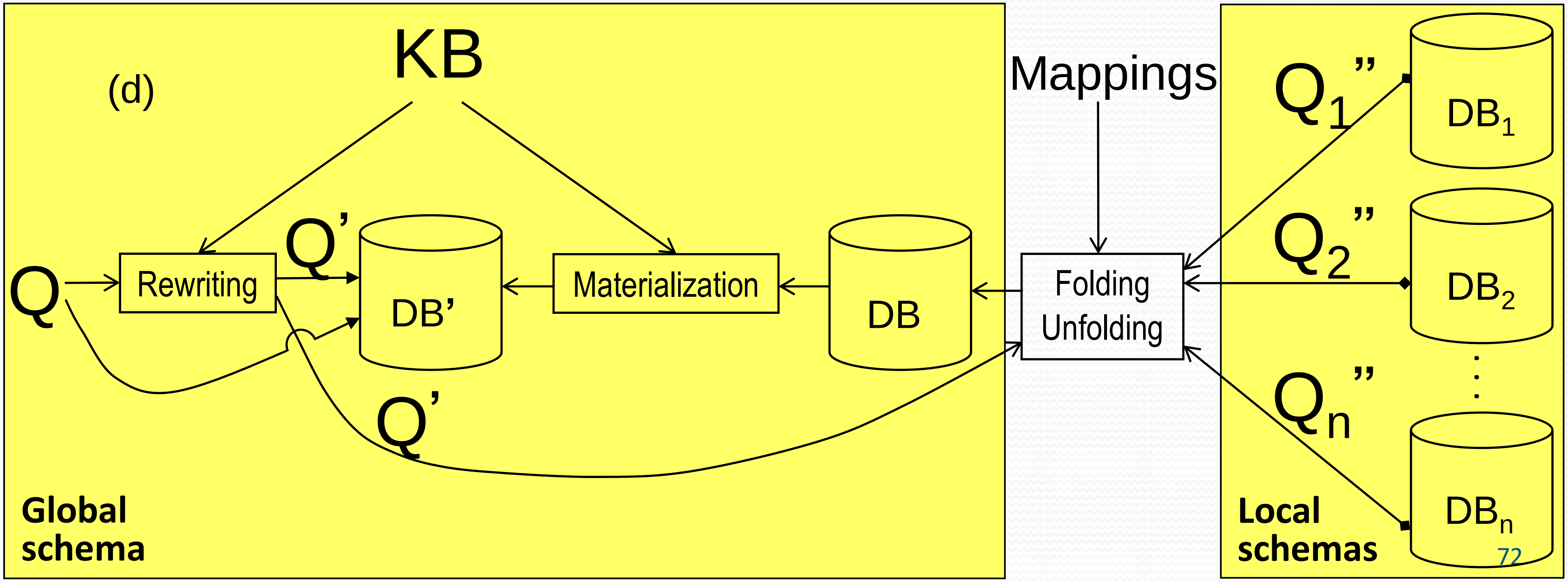
# Unified Architecture

- Combines strategies (a)-(c) of earlier slides
- Meets the needs of ΔForest case study



# Unified Architecture

- Combines strategies (a)-(c) of earlier slides
- Meets the needs of ΔForest case study



# RBDA$_s$-Style KBDA$_s$ Architecture: Expressivity of Rule Systems

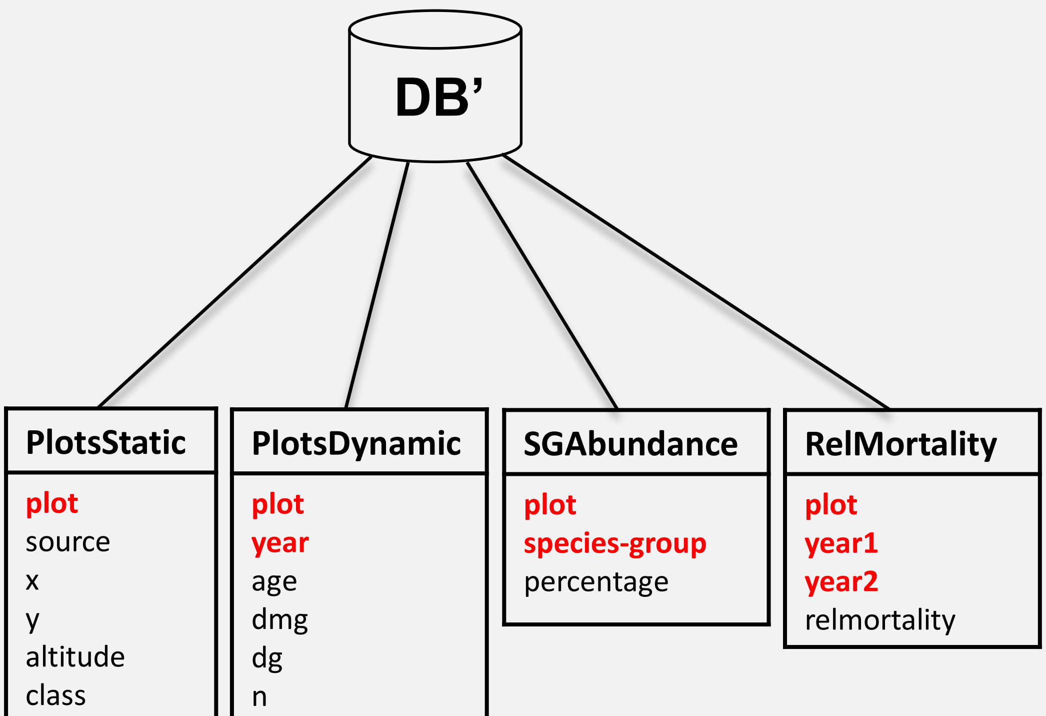
- The language of the global schema can be generalized from unary/binary (OBDA$_s$) to n-ary predicates (RBDA$_s$)
- When decidability of querying is not required, RBDA$_s$ expressivity can be extended from Datalog, Datalog$^{\pm}$, and description logic to Datalog$^{+}$, Horn logic, and FOL, as enabled by Deliberation RuleML 1.01,
  - Features customizable with the MYNG 1.01 GUI
- Moreover, Reaction RuleML 1.0 can express updates, as needed for KBDM$_s$ (Ontology-based Data Management)

# RBDA$_s$-Style KBDA$_s$ Architecture: Uniformity via Rule Systems

- Rule-based style of Unified Architecture (earlier slide)
- Presentation syntax (":-"), serialization ("<RuleML>"), and semantics approach (model theory) uniform from *queries (+ integrity constraints)* to *KBs* to *mappings* to *abstract DBs*
- Division of labor between KB rules and mapping rules can be modified *without crossing paradigm boundaries*
  - Allows KB- and mapping-directed normal forms
- Assumptions (unique-name and closed-world) of DBs accommodated by *default assumptions of rule systems*

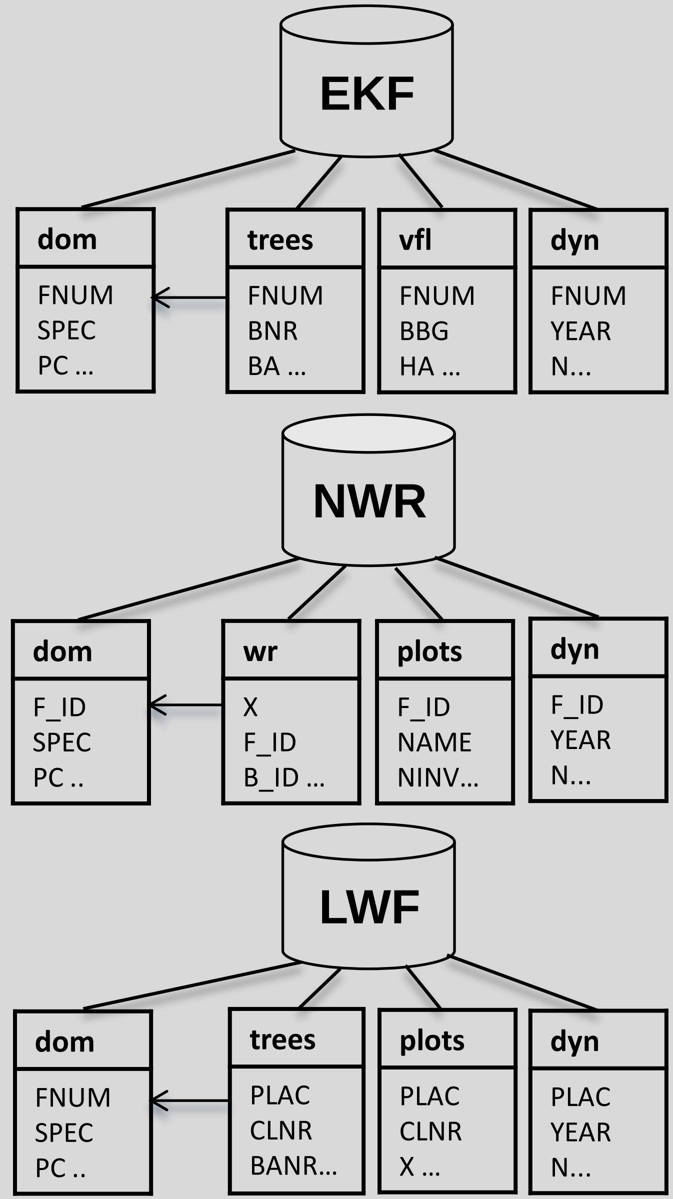
## Global schema

# ΔForest: Schemas for Architecture (a)-(d)




Keys – three composite – in **bold red**

## Local schemas



# Questions Addressed

1. Are there sufficiently many eligible plots in order to perform an analysis per main tree species?
2. Are there sufficiently many eligible plots in order to perform an analysis per main tree species **and** climatic region?
3. Which eligible plots represent pure tree stands and which eligible plots represent mixed tree stands?

# Query Rewriting

**q(?plot) :- .EligiblePlot(?plot)**
**.TreeStandKey(?id ?plot "oak")**
**.TreeStandAbundance(?id ?pct)**
**?pct >= 15.**

Query

Input

**Exists ?id (.TreeStandKey(?id ?plot ?sp) .TreeStandAbundance(?id ?pct)) :-**
**.TreeStandMerged(?plot ?sp ?pct).**

KB Rule

---

**q(?plot) :- .EligiblePlot(?plot)**
**.TreeStandMerged(?plot "oak" ?pct)**
**?pct >= 15.**

Rewritten

Output

# Query Unfolding

**q2(?plot) :- .TreeStandMerged(?plot "oak" ?pct)**
**?pct >= 15.**

Query 2

Input

**.TreeStandMerged(?plot "oak" ?pct) :-**
**EKF.dom(?plot "Quercus petraea" ?pct1)**
**EKF.dom(?plot "Quercus robur" ?pct2)**
**?pct = ?pct1 + ?pct2.**

Mapping Rule

---

**q2(?plot) :- EKF.dom(?plot "Quercus petraea" ?pct1)**
**EKF.dom(?plot "Quercus robur" ?pct2)**
**?pct1+?pct2 >= 15.**

Unfolded

Output

# Conclusions

- Ontology-Based Data Access (OBDA) founded on three kinds of rules: *Query rules* (including integrity rules), *KB rules* (for query rewriting and DB materialization), as well as *mapping rules* (for query unfolding and DB folding)
- OBDA complemented by Rule-Based Data Access (RBDA) and generalized to Knowledge-Based Data Access (KBDA)
- Specified an RBDA-uniform $KBDA_s$ architecture with unified mediator, warehouse, and bidirectional strategies
- RuleML used for XML-serialized rules, MYNG-customized rule expressivity, and platform-independent RBDA
- Introduced *ΔForest* specialization of RBDA architecture for statistical data analysis in ecosystem research at WSL

# Future Work (1)

- Translate simplified presentation syntax into released XML serialization of Deliberation RuleML 1.01 / MYNG 1.01
- Support implementations of specified architecture reusing (open source) KBDA technology (cf. RBDA wiki page)
- For high-precision RBDQ$_s$ language support, complement current techniques of Datalog$^+$ RuleML for Datalog$^\pm$ RuleML using context-sensitive/semantic validators for "-" constraints
- Evaluate (mediator/warehouse, relational/graph, …) trade-offs for KBDQ$_s$ in PSOA RuleML as executed in PSOATransRun
- Develop ΔForest study at WSL for extended and new data sources of big volume, variety, and velocity (e.g., about climate change)
- Augment geospatial KBDQ$_s$ mappings with Optique mapping (bootstrapping, repair, …) techniques

# Future Work (2)

- Compare engines for OBDQ$_s$ and RBDQ$_s$, including HYDRA and RDFox, w.r.t. expressivity and efficiency
- Adapt *Semantic Automated Discovery and Integration* (SADI) test cases for KBDQ$_s$ experiments in PSOA RuleML querying
- Evaluate A*bstract* L*ogic-based* A*rchitecture* S*torage systems* & K*nowledge base* A*nalysis* (ALASKA) for RBDA$_s$
- Extend the KBDA$_s$ architecture with semantic annotation rules for (Deep) Web data extraction (Deep Web Mediator)
- Use Grailog for KBDA$_s$ data and knowledge visualization
- Explore synergies between the logical KBDA$_s$ approach with statistical approaches, e.g. from Statistical Relational AI