Efficient XQuery Rewriting using Multiple Views

Ioana Manolescu [1] **Konstantinos Karanasos [1]**
Vasilis Vassalos [2] Spyros Zoupanos [1]

[1]Leo team, INRIA Saclay and Univ. Paris-Sud XI, France [2]AUEB, Greece

ICDE 2011, Hannover, Germany

Arpil 15, 2011

XML view-based rewriting

- XML: the standard model for data on the Web
- XQuery: de-facto query language for XML
- Materialized views crucial in query optimization

XML view-based rewriting

- XML: the standard model for data on the Web
- XQuery: de-facto query language for XML
- Materialized views crucial in query optimization

- Equivalent XQuery rewriting using multiple XQuery views

Steps of view-based query evaluation

- Filter out useless views
- Find all equivalent query rewritings
  - the problem we consider
- Choose and evaluate the most appropriate rewriting
  - query optimizer, execution engine
  - implemented in ViP2P (http://vip2p.saclay.inria.fr)

Target rewritings

- Equivalent: *the same results as the query*
- Complete: *no need of the base documents*
- Multiple views: *joining them a problem on its own*
- Minimal: *no redundant view instances*
- Generic logical XML algebra:
  easy to use by existing systems

Novelties of our approach

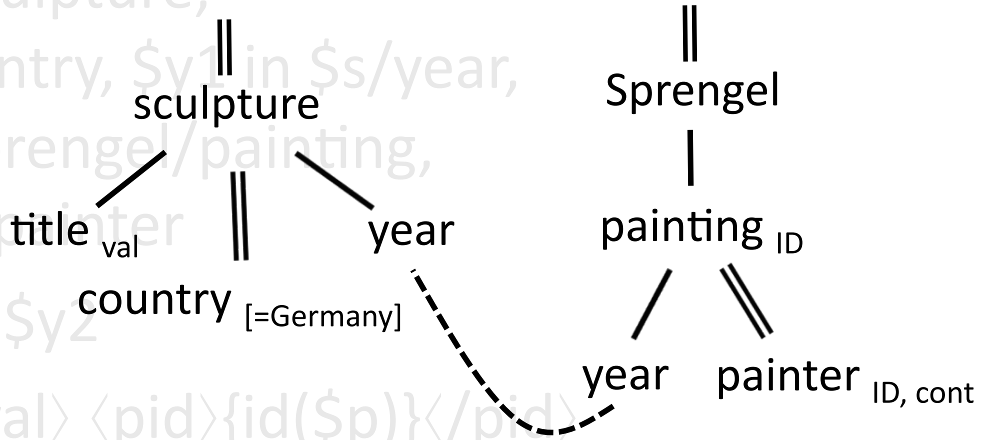
- Query/view language: powerful conjunctive XQuery dialect
  - for … where … return …
  - bindings from multiple documents
- Rewriting minimality built in the search process
- Efficient query rewriting algorithms
  - algebra based pruning
  - exploit XDBMS implementation artifacts when available (IDs, structural IDs)

Query and view language

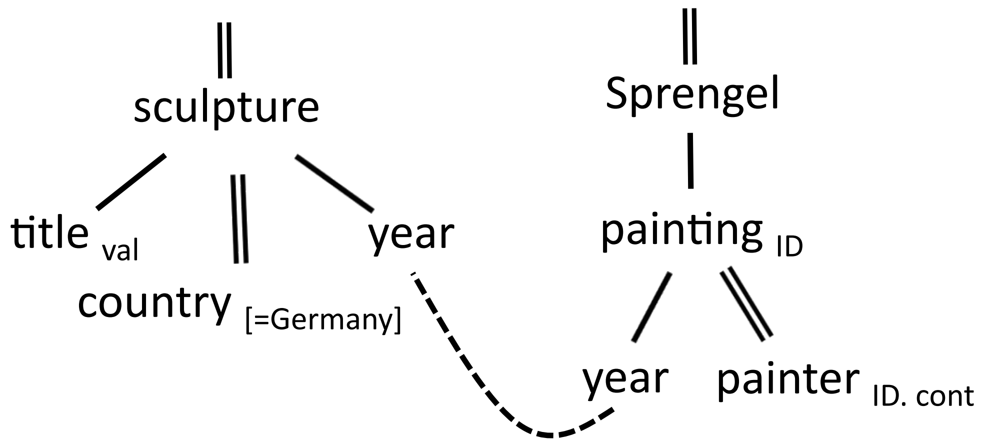
```
for  $s in doc(”sculptures”)//sculpture,
     $t in $s/title, $c in $s//country, $y1 in $s/year,
     $p in doc(”museums”)//Sprengel/painting,
     $y2 in $p/year, $pr in $p//painter
where $c=‘Germany’ and $y1=$y2
return ⟨v⟩ ⟨tval⟩{string($t)}⟨/cval⟩ ⟨pid⟩{id($p)}⟨/pid⟩
                ⟨prid⟩{id($pr)}⟨/prid⟩ ⟨prcont⟩{$pr}⟨/prcont⟩ ⟨/v⟩
```

Query and view language

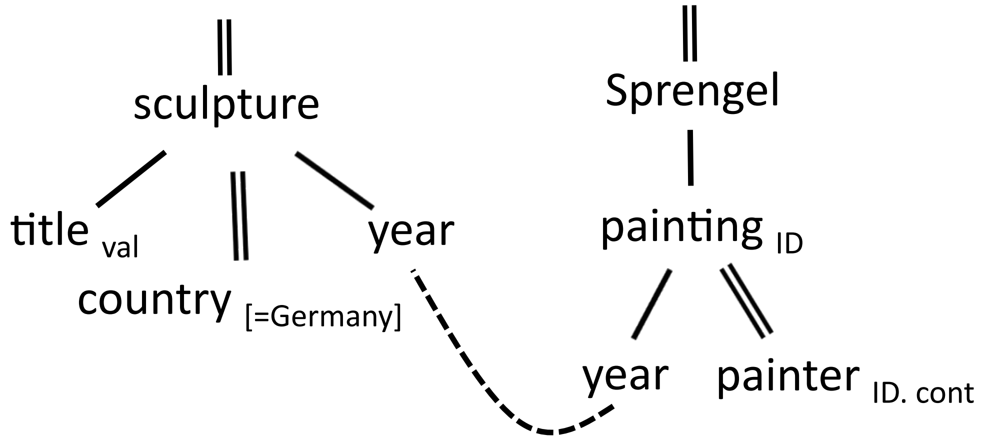
```
for  $s in doc(”sculptures”)//sculpture,
     $t in $s/title, $c in $s//country, $y1 in $s/year,
     $p in doc(”museums”)//Sprengel/painting,
     $y2 in $p/year, $pr in $p//painter
where $c=‘Germany’ and $y1=$y2
return 〈v〉 〈tval〉{string($t)}〈/cval〉 〈pid〉{id($p)}〈/pid〉
            〈prid〉{id($pr)}〈/prid〉 〈prcont〉{$pr}〈/prcont〉 〈/v〉
```



Query/view data



Query/view data

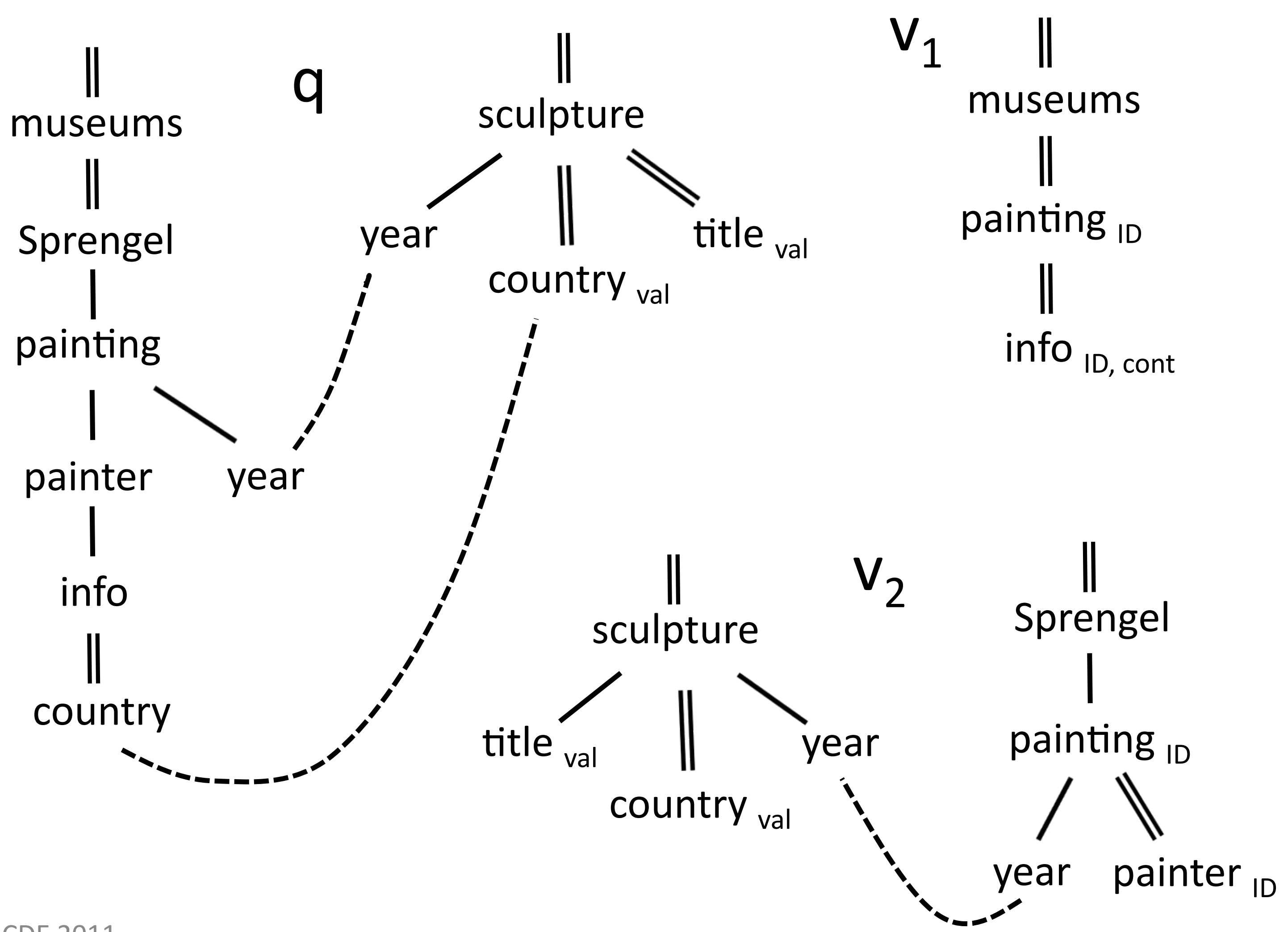


title.val	painting.ID	painter.ID	painter.cont
...	...	...	...

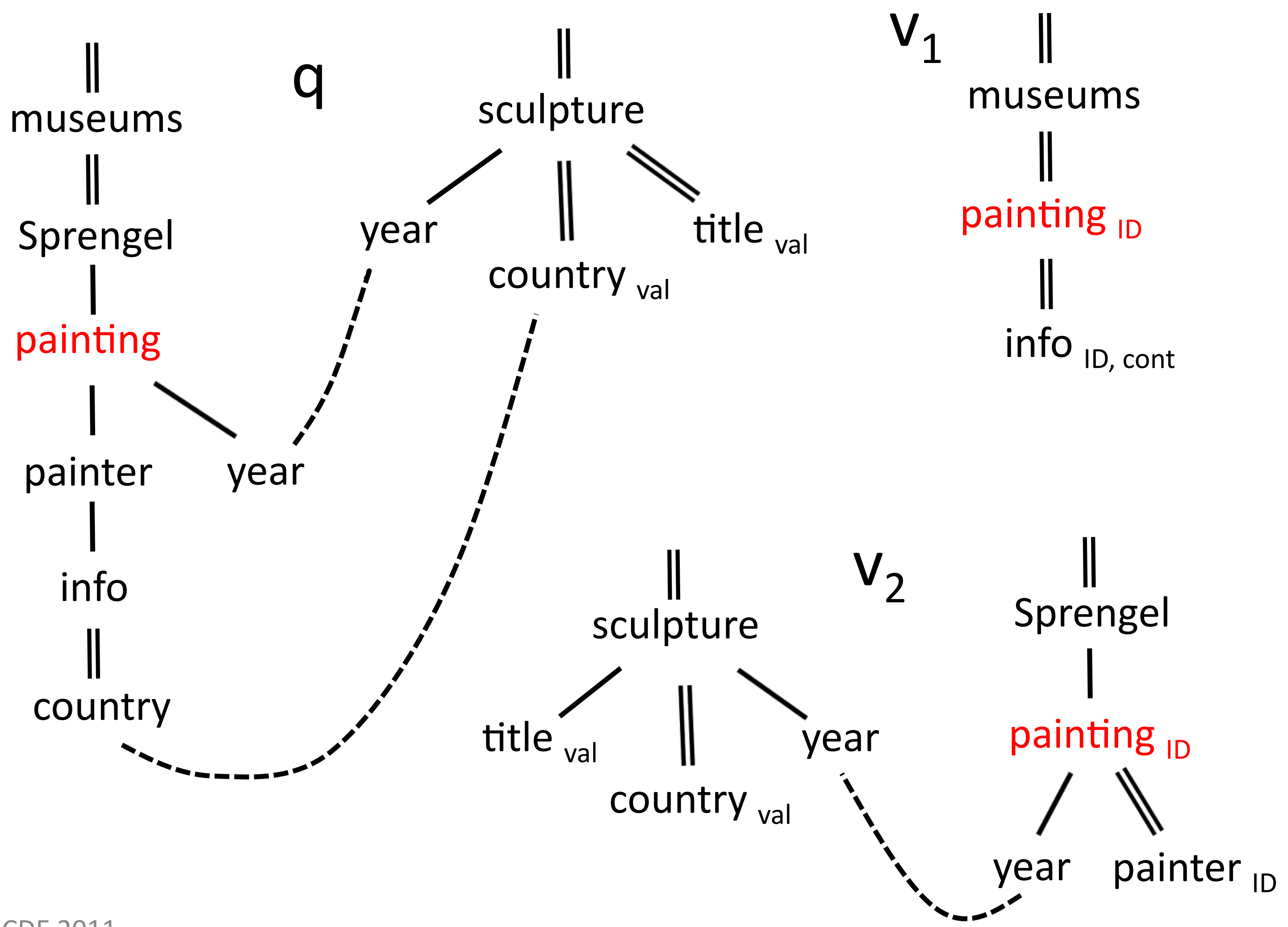
Outline

- Introduction
- ➢ **Motivating example**
- Tree pattern rewriting
- Joined pattern rewriting
- Experiments
- Conclusion

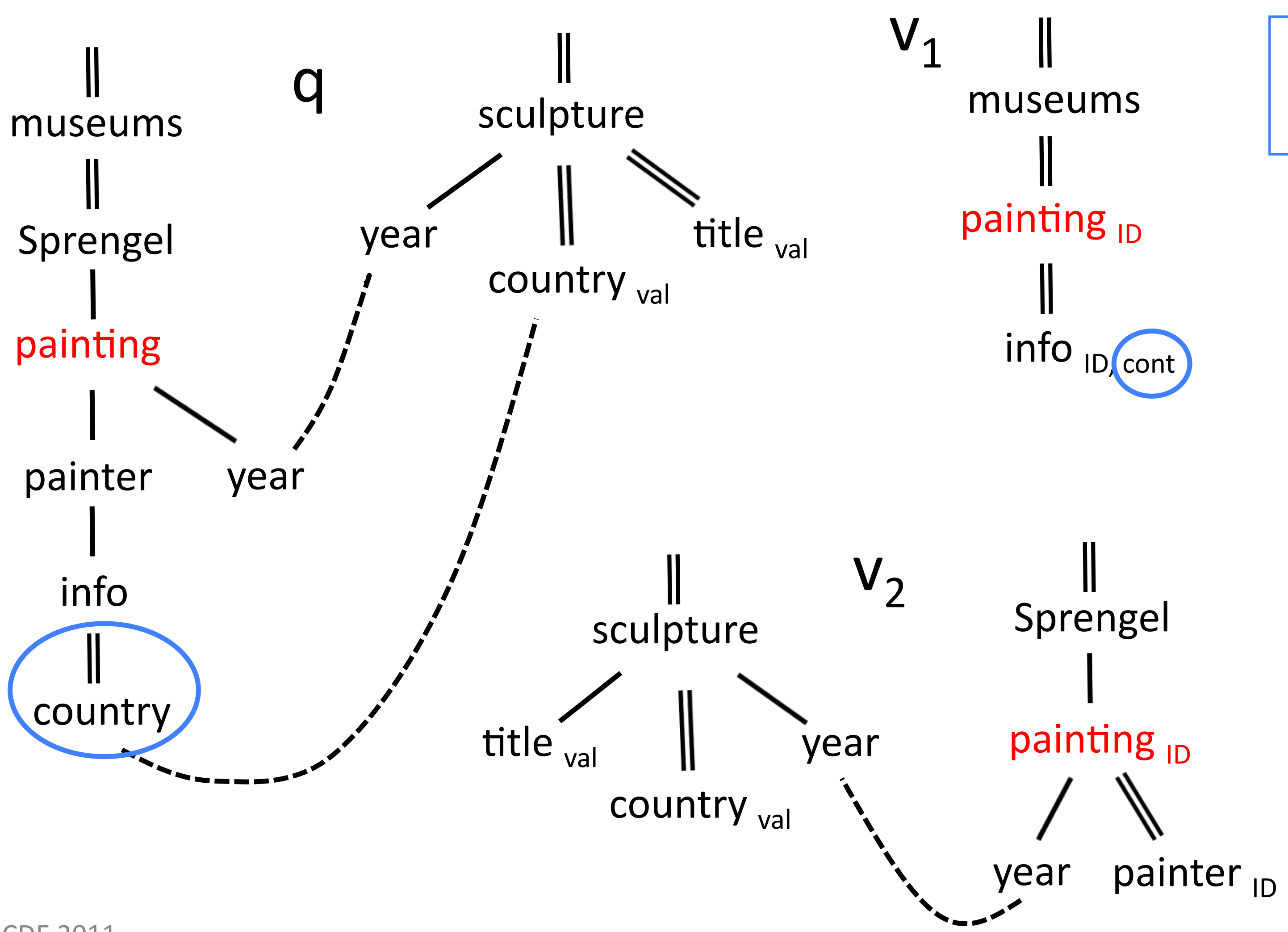
Motivating example



Motivating example

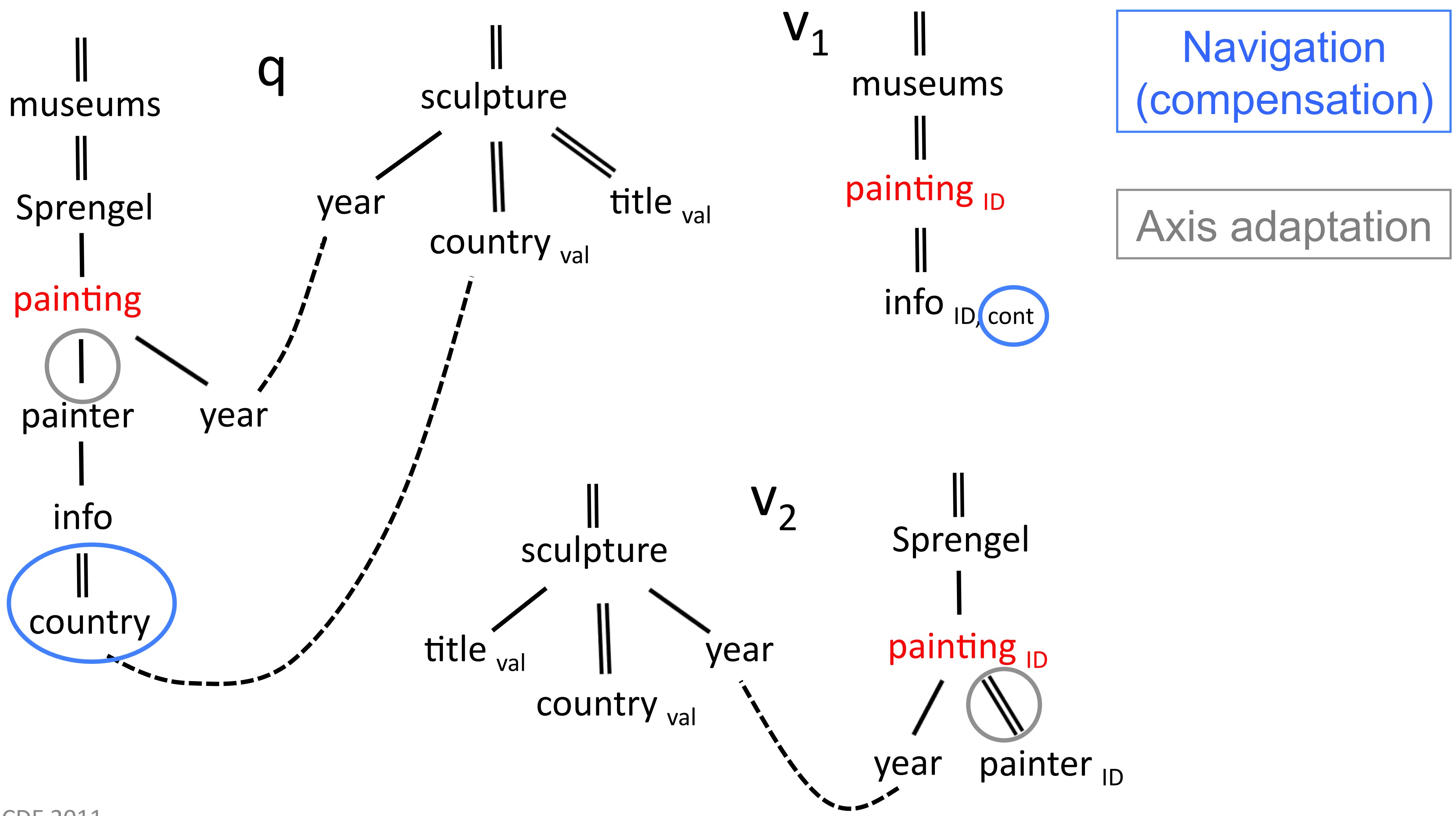


Motivating example

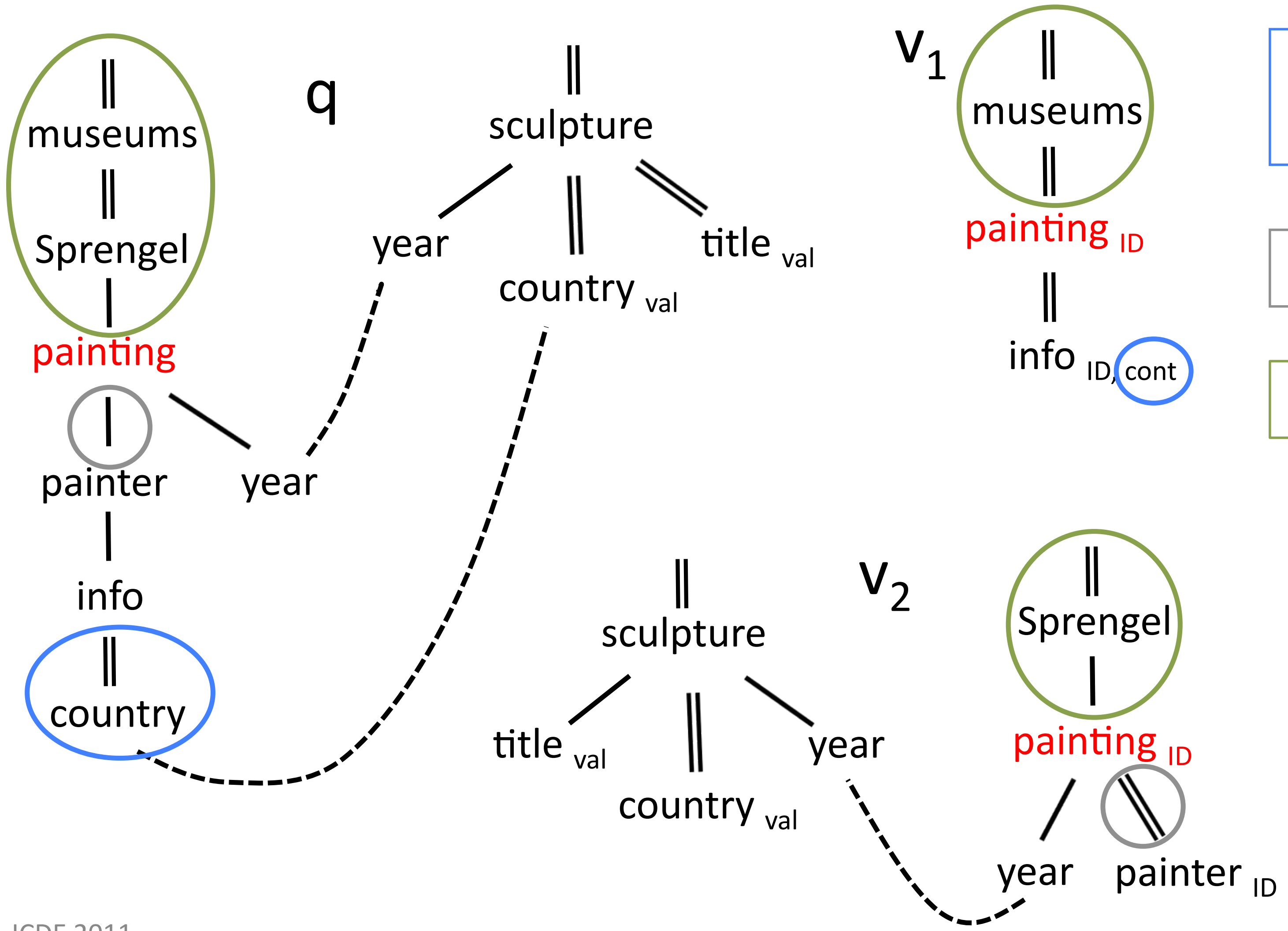


Navigation (compensation)

Motivating example



Motivating example

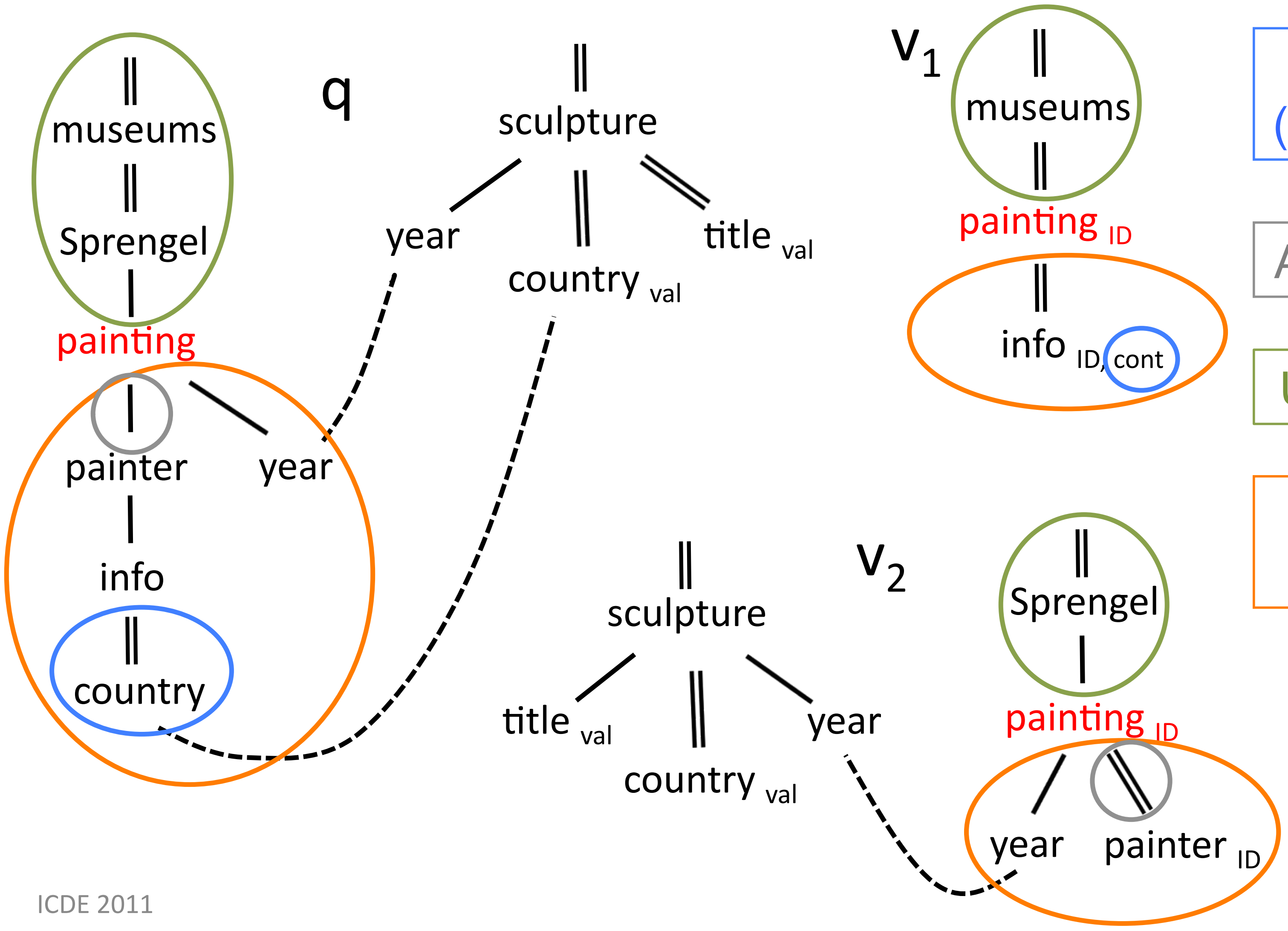


Navigation (compensation)

Axis adaptation

Upward joining

Motivating example



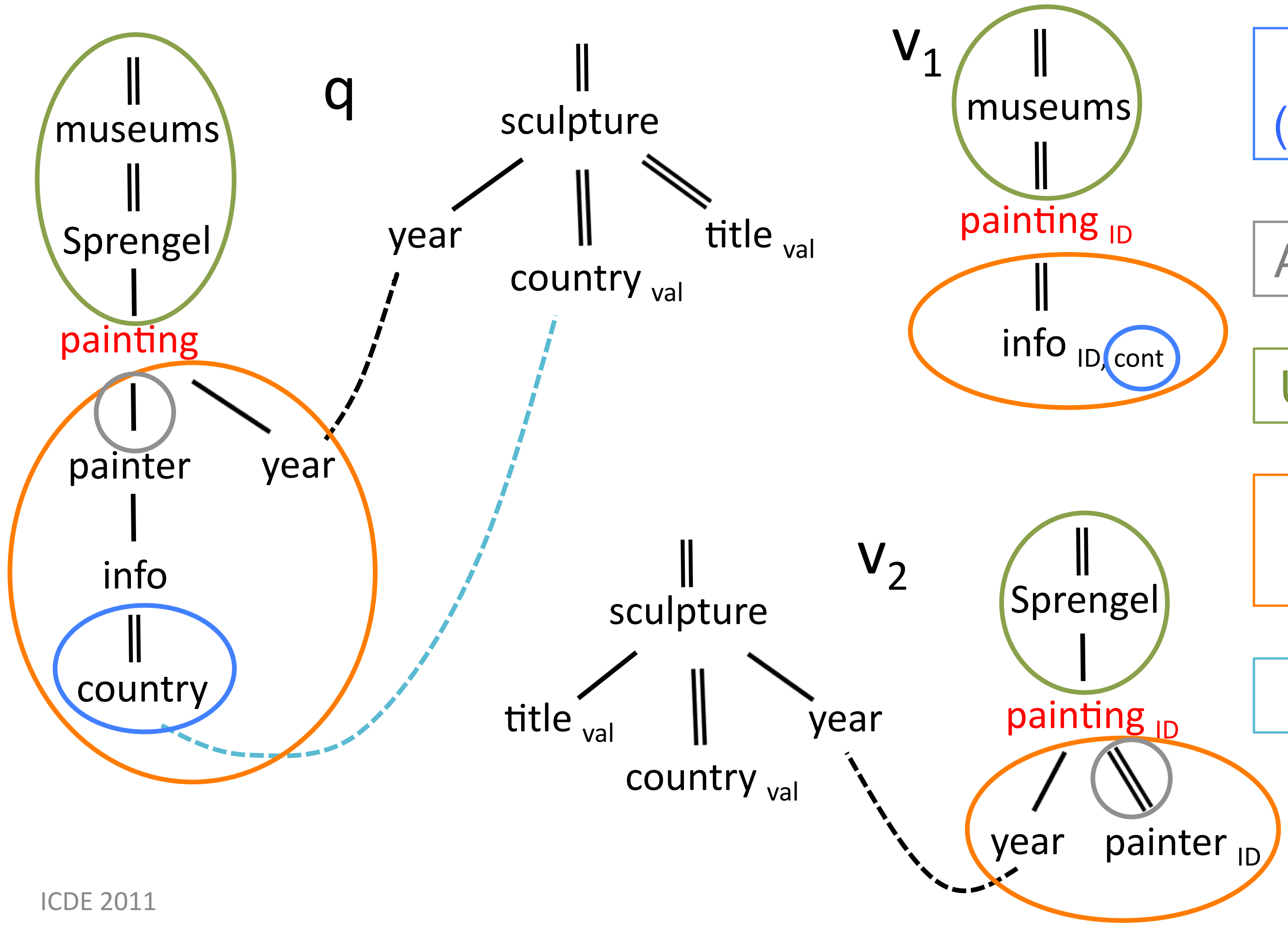
Navigation (compensation)

Axis adaptation

Upward joining

Downward joining

Motivating example



Navigation (compensation)

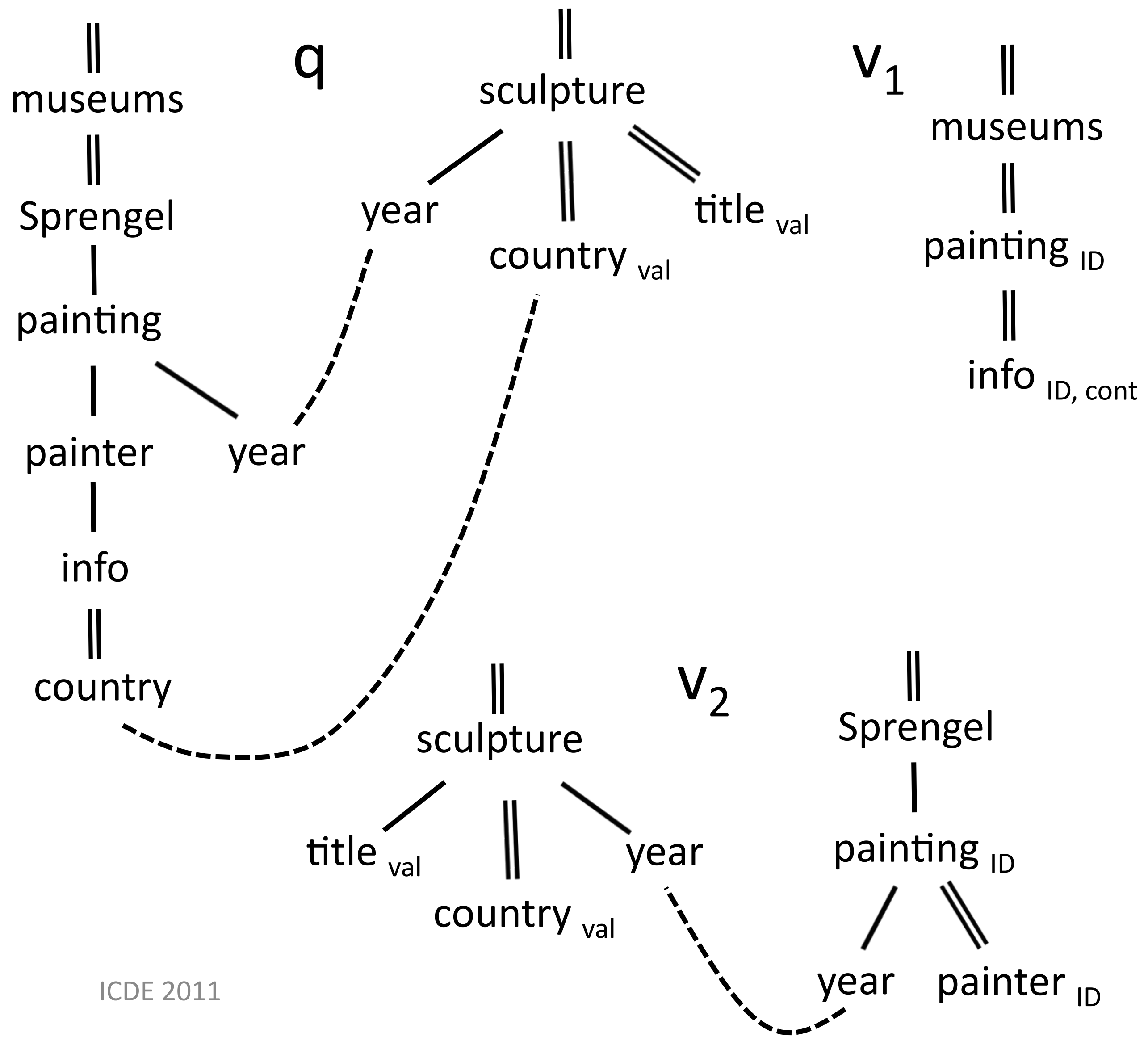
Axis adaptation

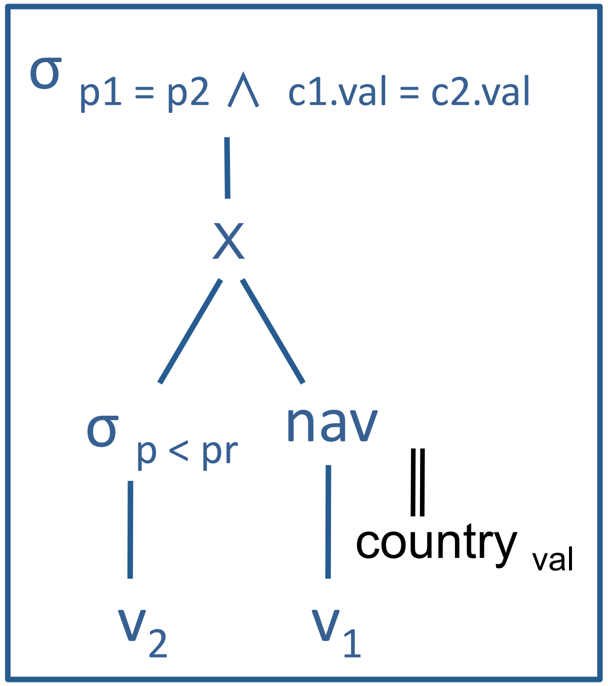
Upward joining

Downward joining

Value joins

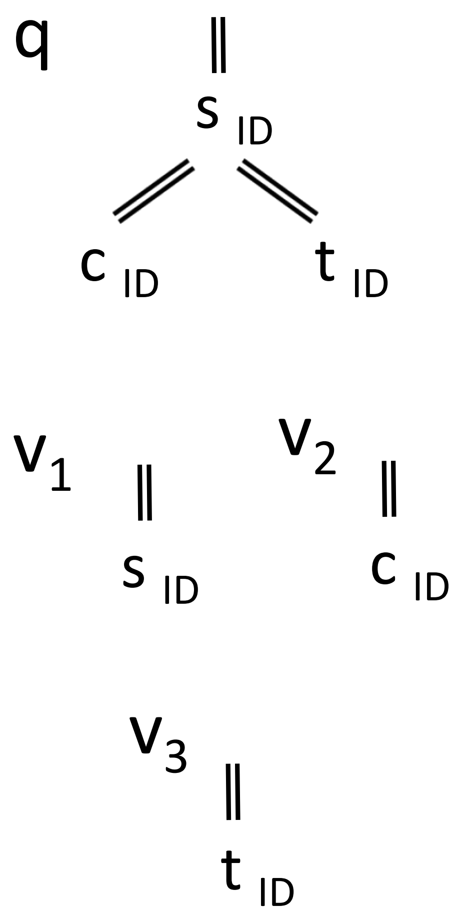
Rewriting (logical algebra)



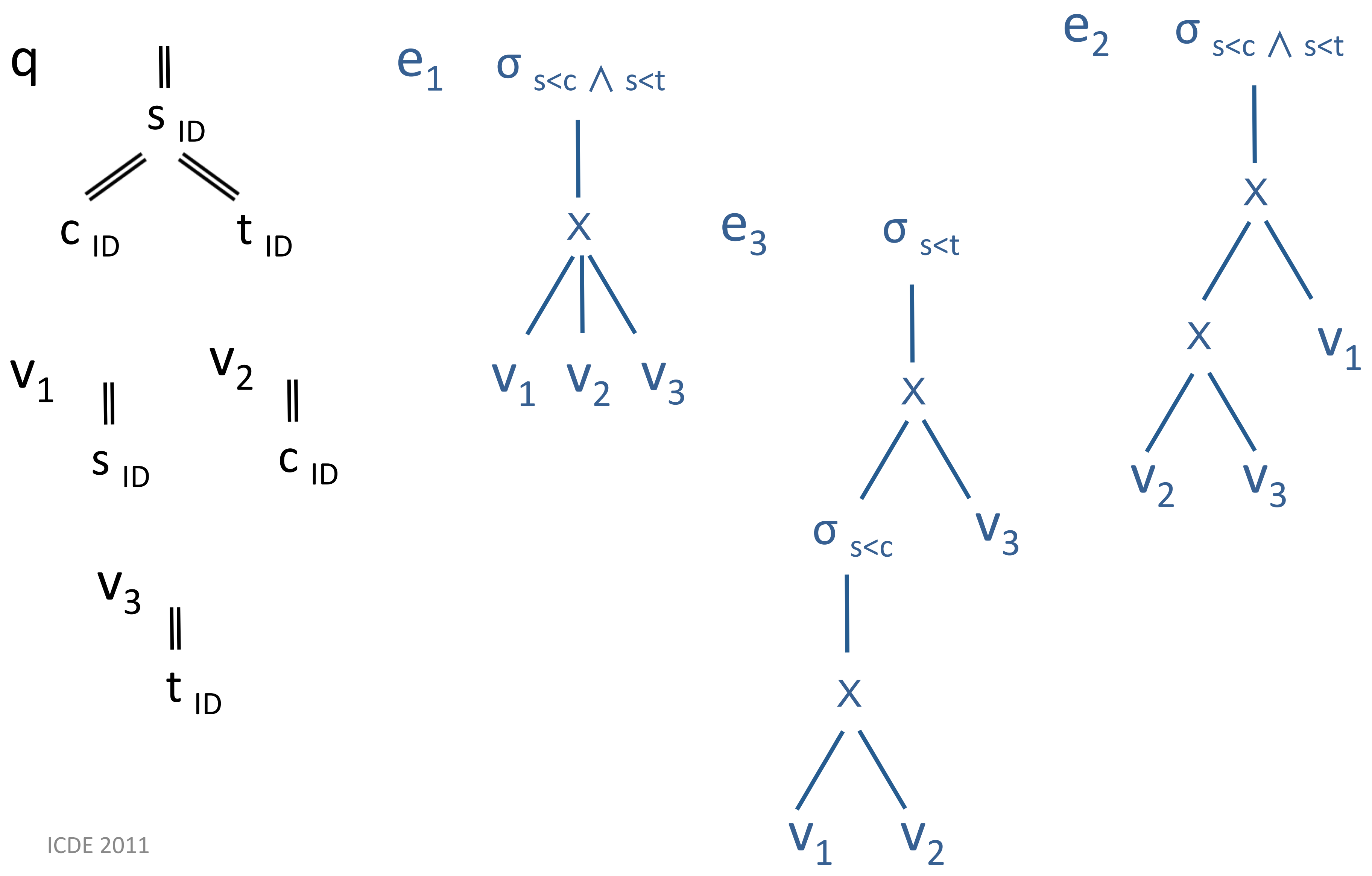


Outline

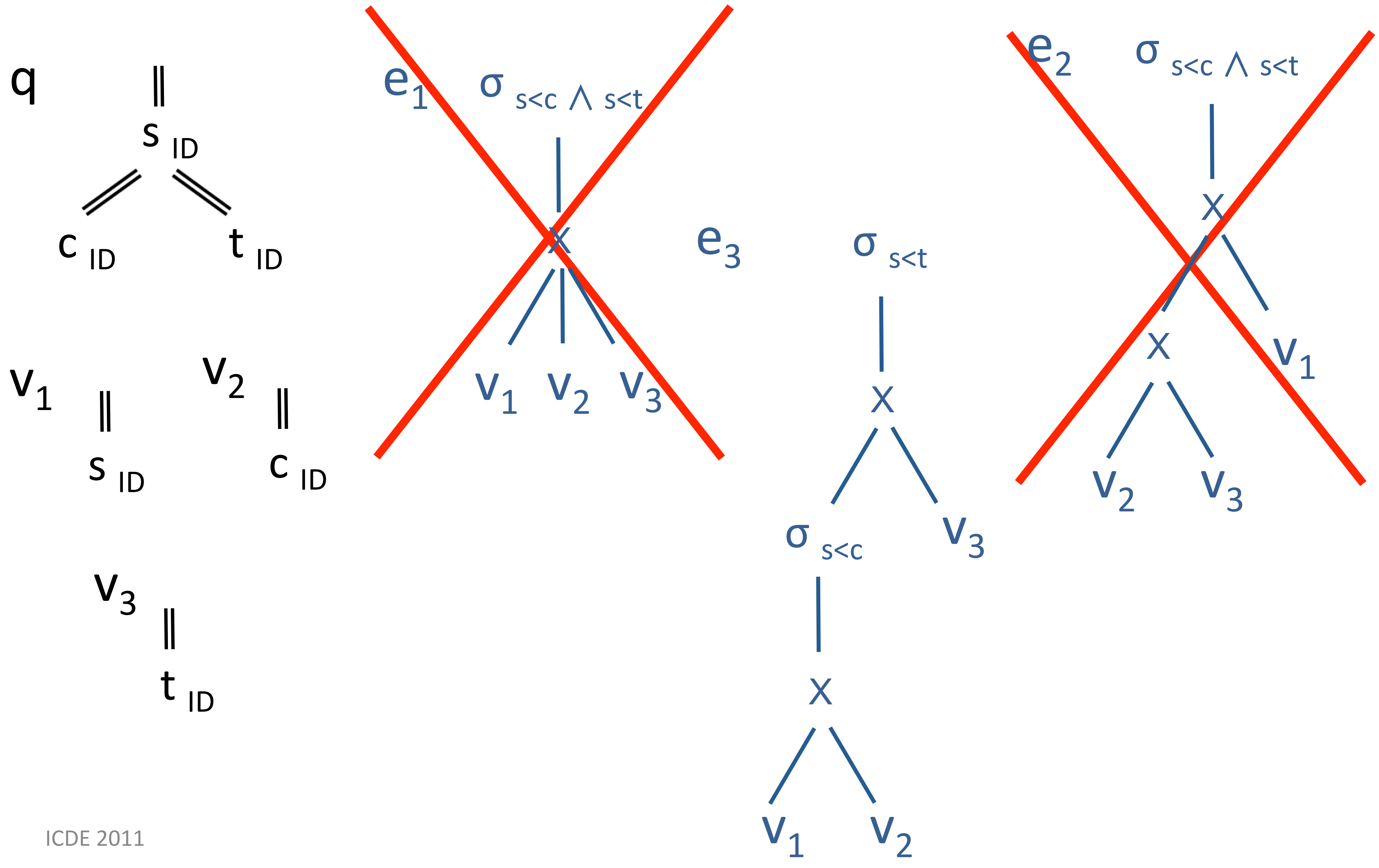
LDQT rewritings



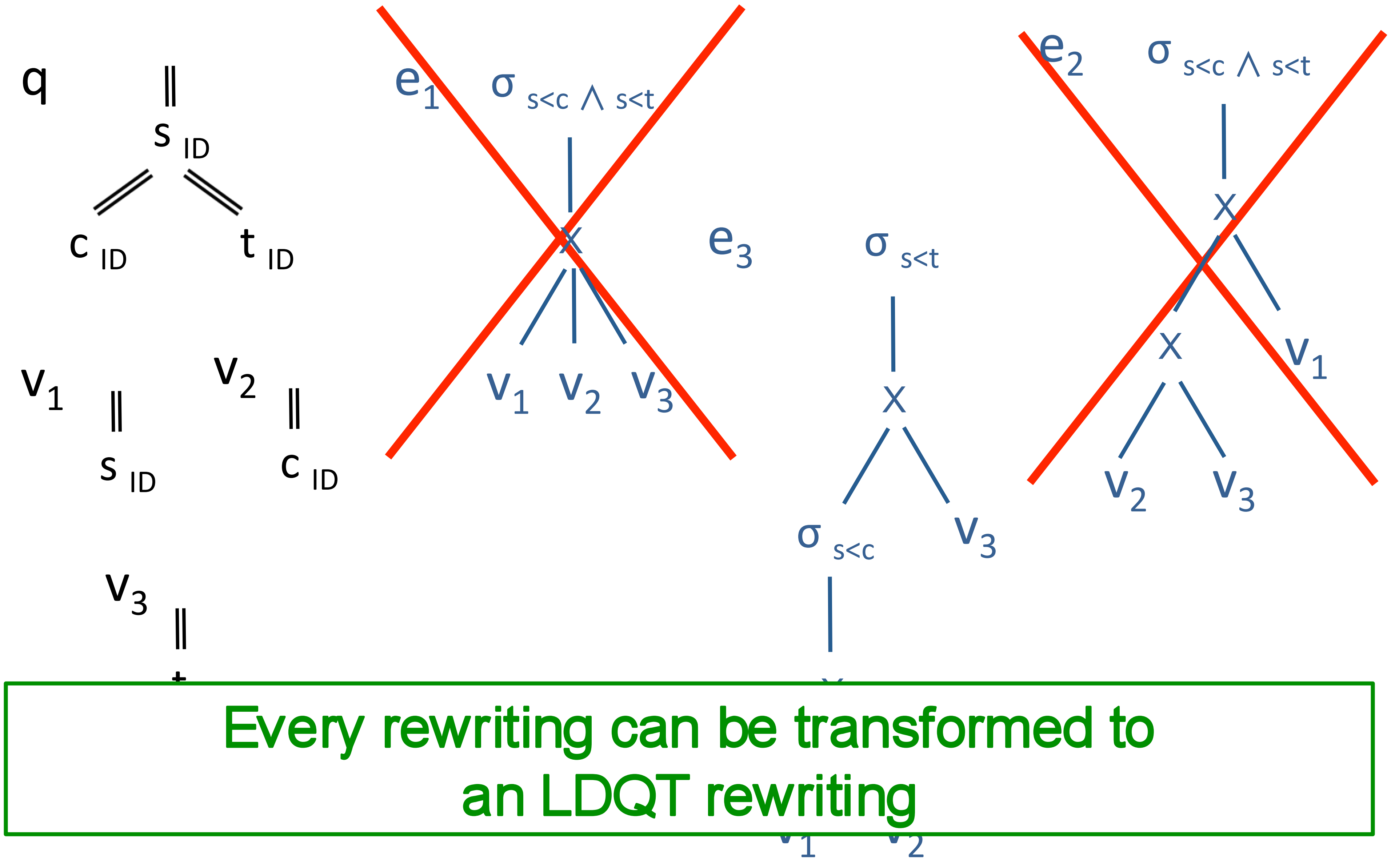
LDQT rewritings



LDQT rewritings



LDQT rewritings

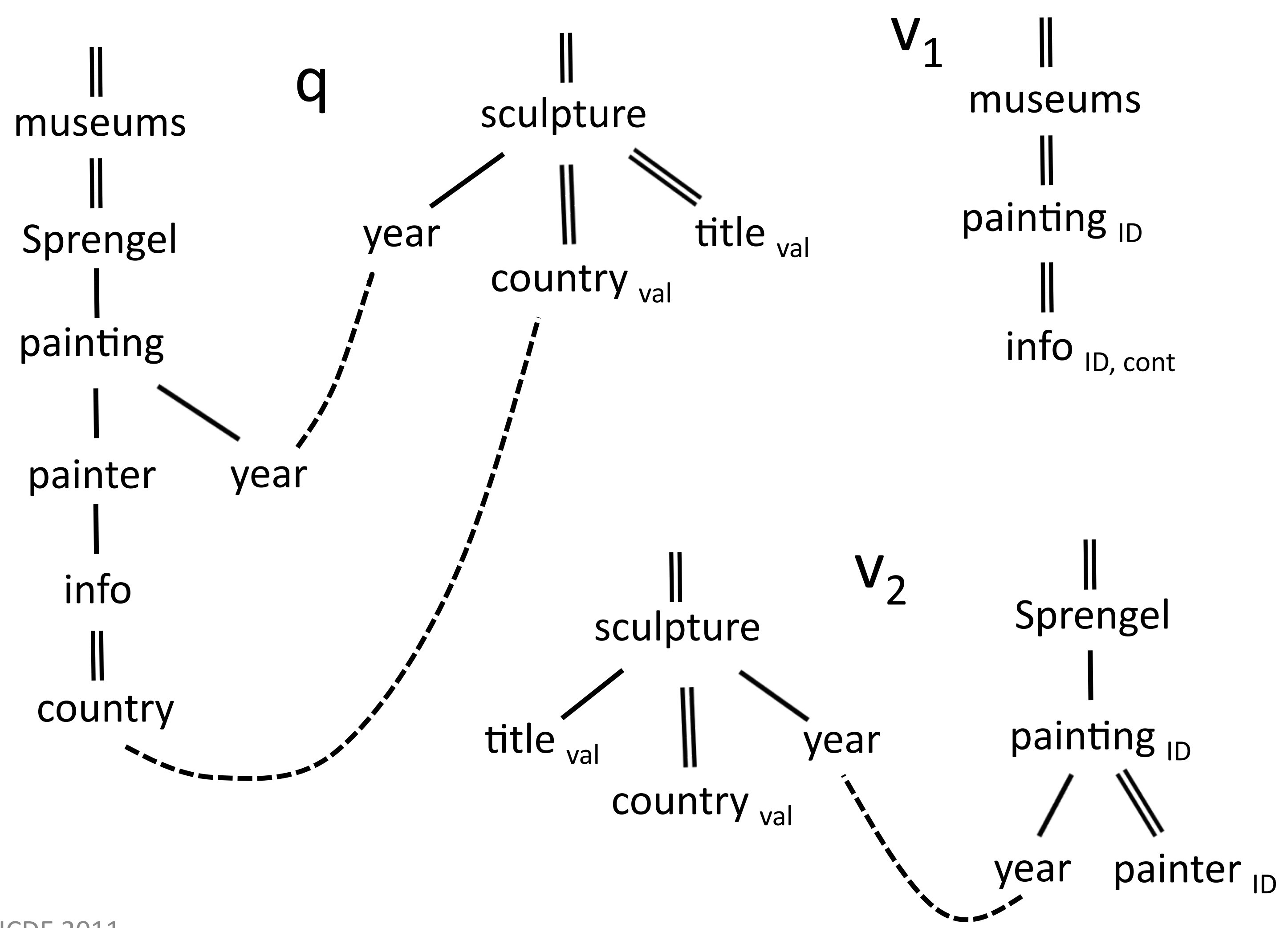


Every rewriting can be transformed to an LDQT rewriting

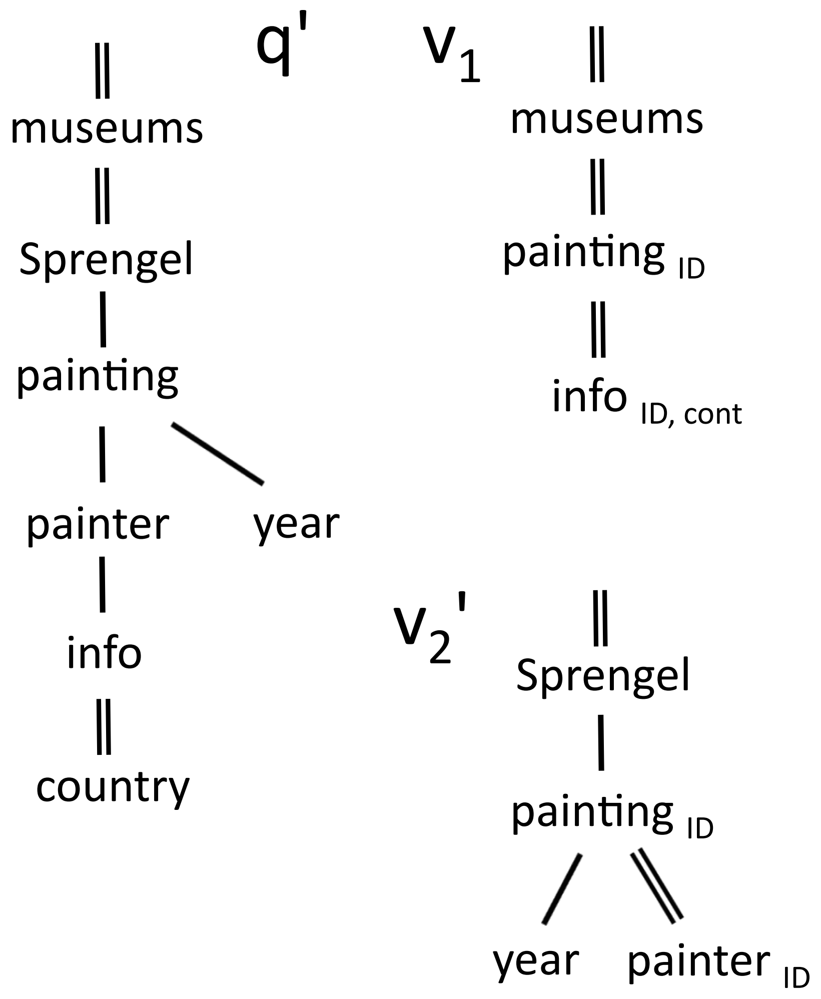
Rewriting algorithm

- Choose the useful (query embeddable) views
  - 1-view (partial) rewritings
- Adaptation of 1-view rewritings
  - navigation, axis adaptation, selection predicates
- Join partial rewritings to reach equivalent rewritings
- Sound and complete algorithm

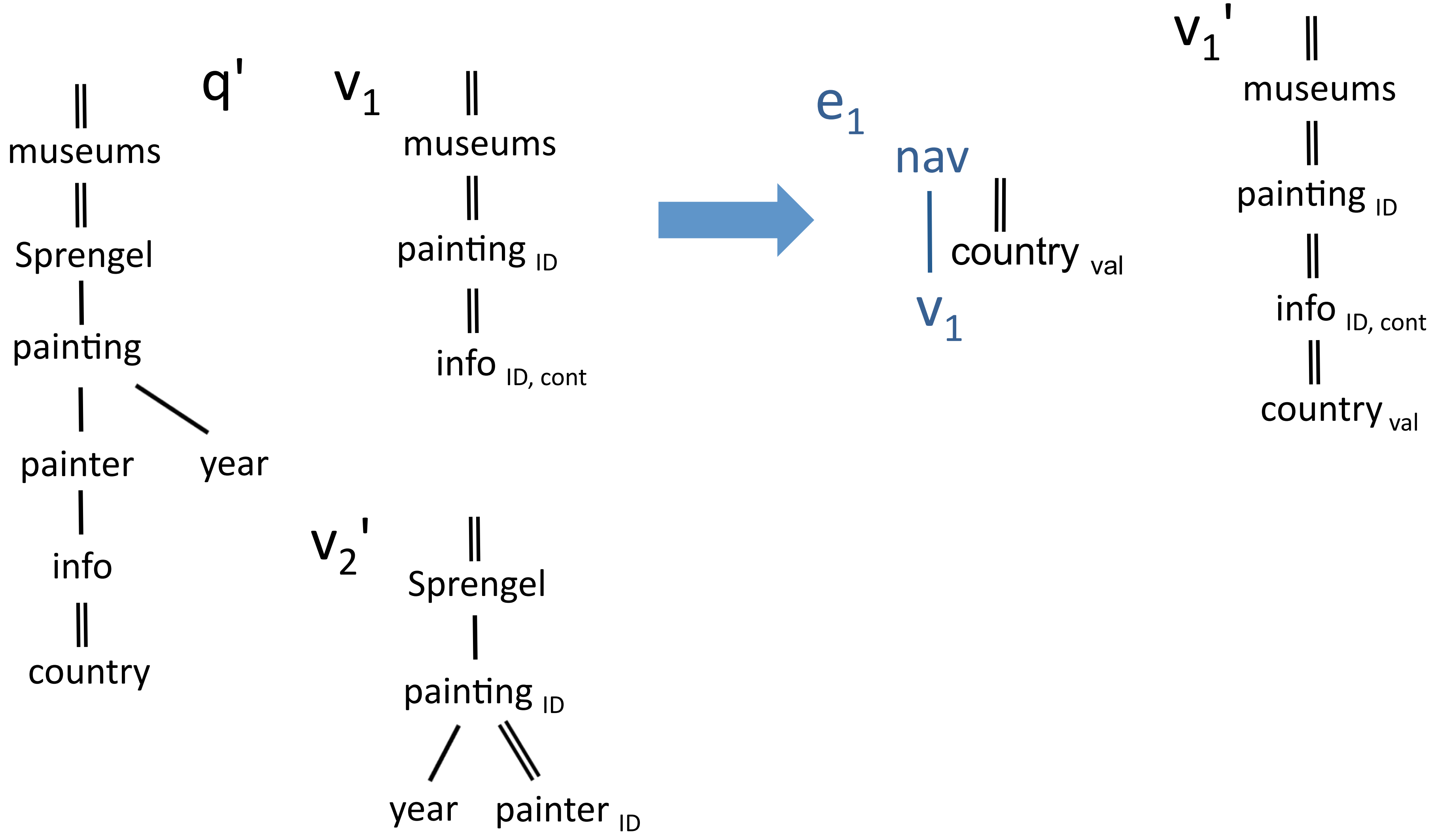
Motivating example



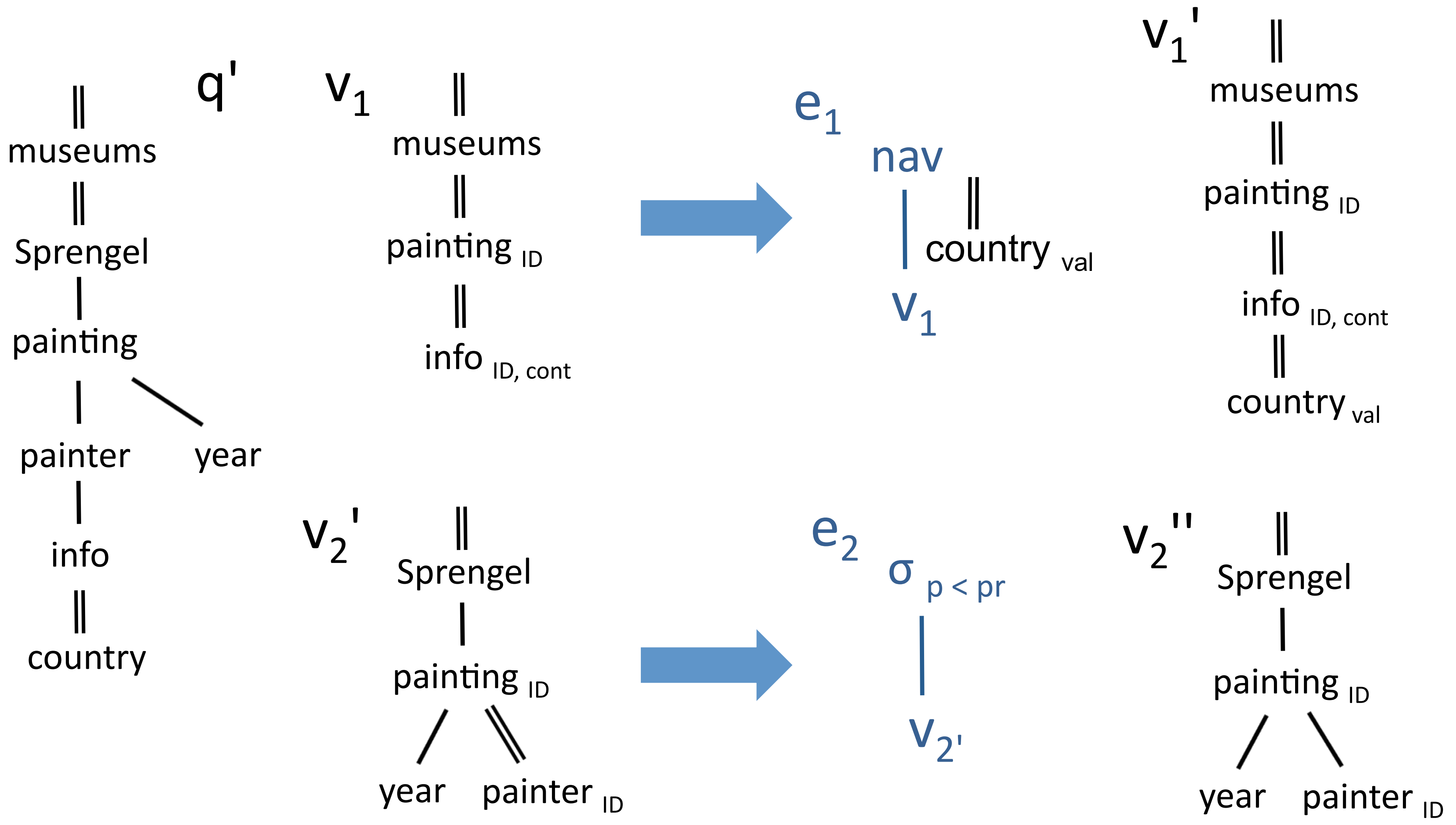
Motivating example



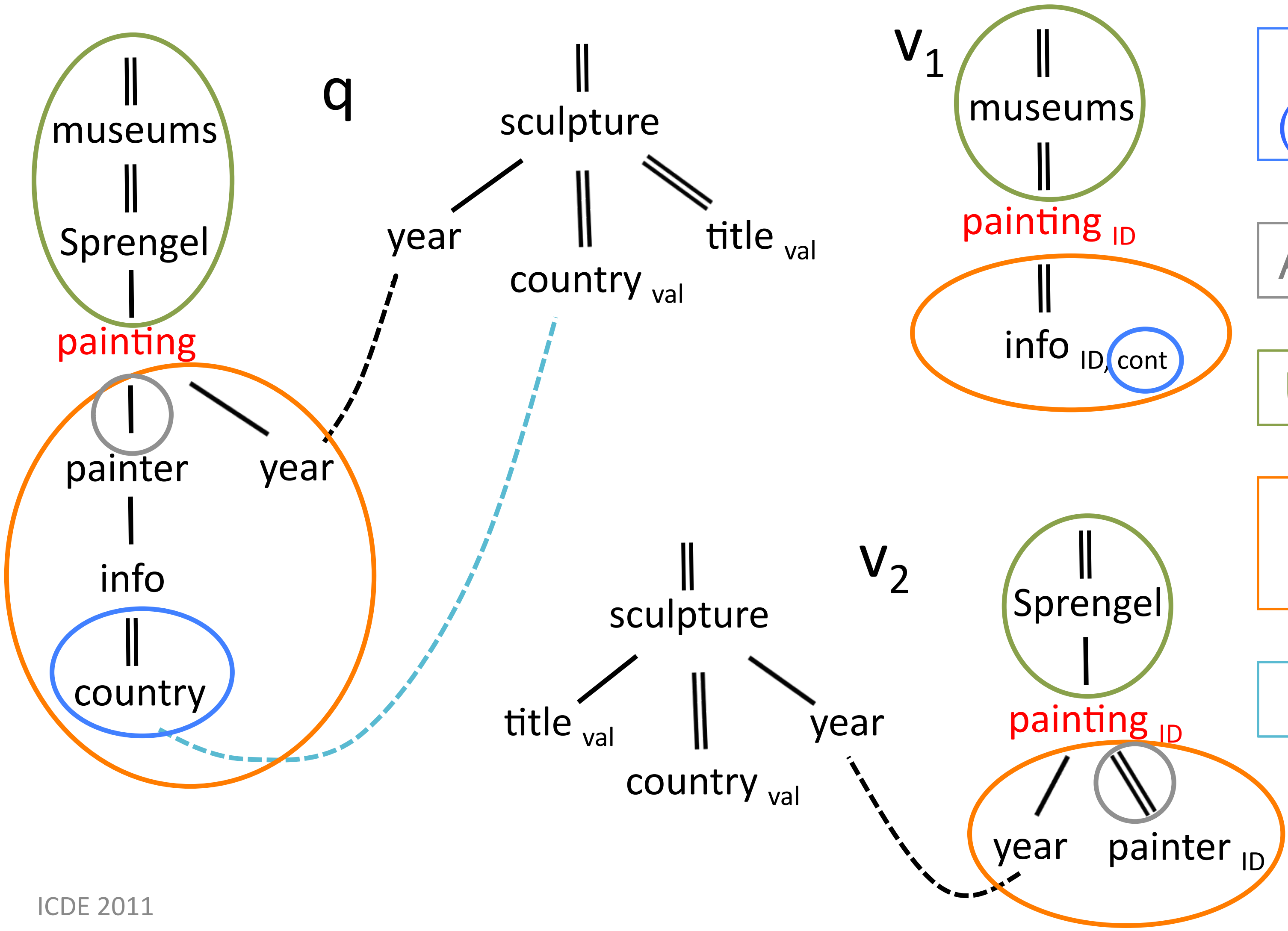
Navigation (compensation)



Axis adaptation



Motivating example



Navigation (compensation)

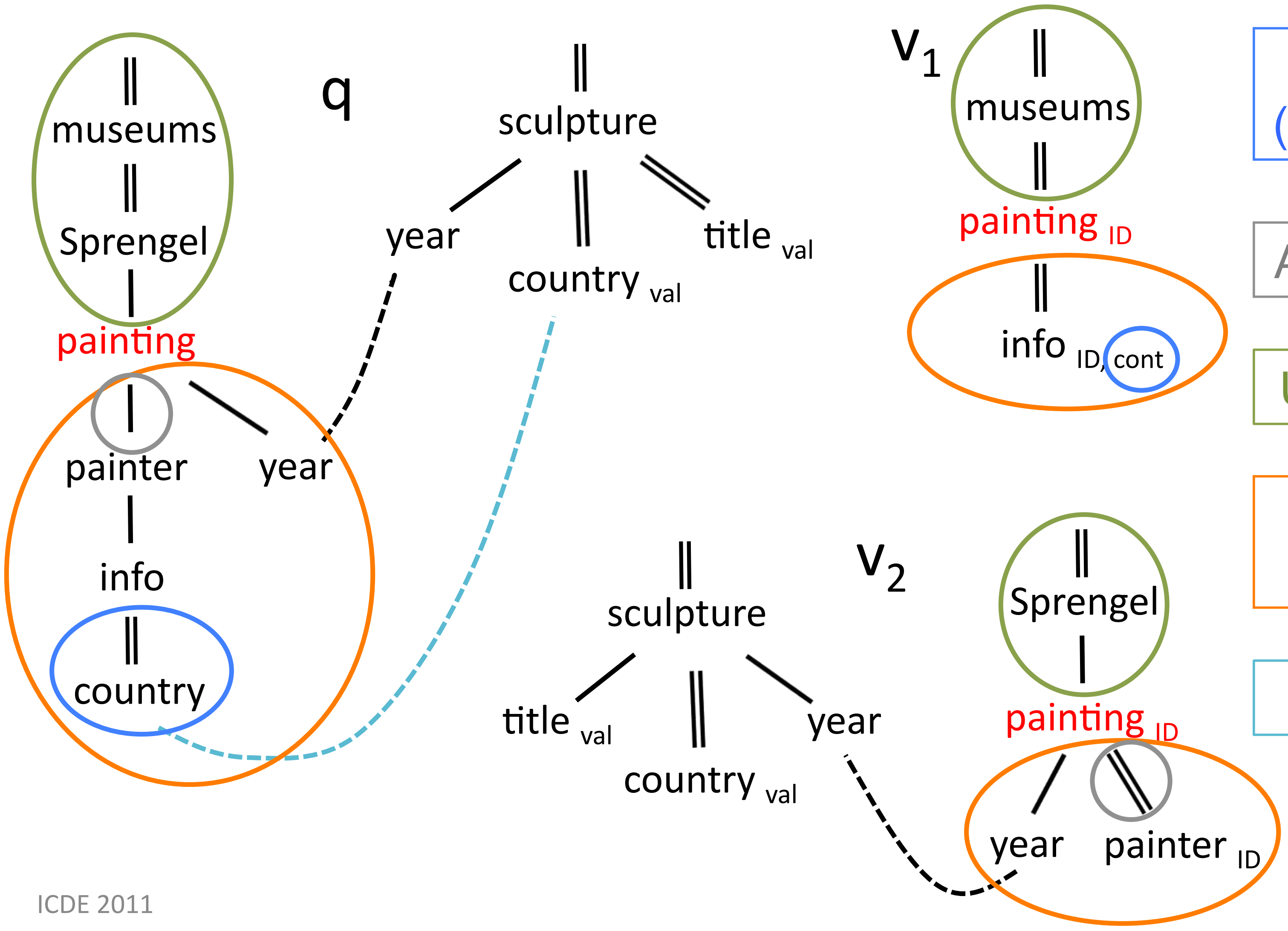
Axis adaptation

Upward joining

Downward joining

Value joins

Motivating example



Navigation (compensation) ✓

Axis adaptation ✓

Upward joining

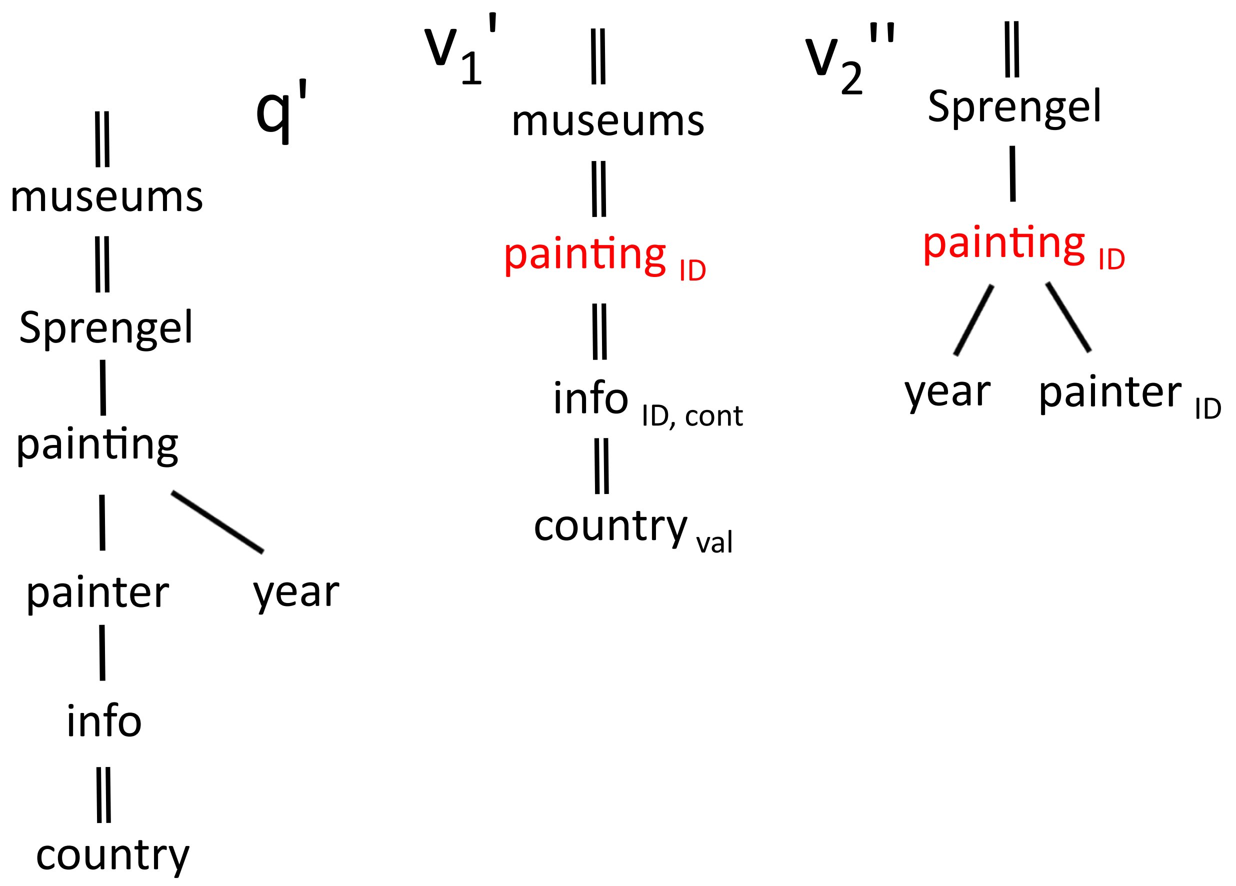
Downward joining

Value joins

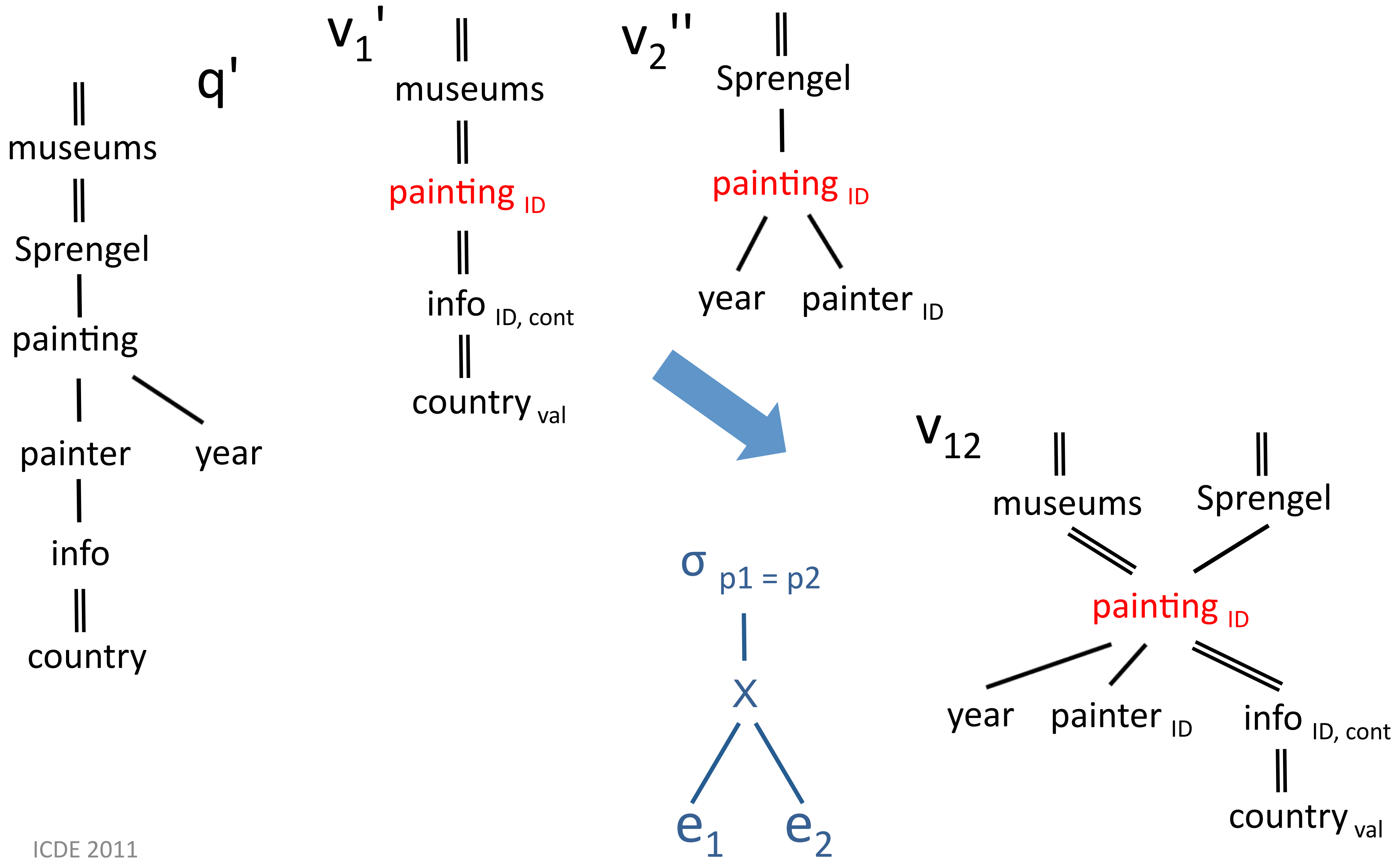
Query-driven join

- Given two partial rewritings for query *q*, join them into a new (partial) rewriting for *q*
- Equi- and structural-joins (if structural IDs)
- Zipping: the paths above joining nodes
- Merging: the forests below joining nodes
- Extra challenges due to multiple returning nodes

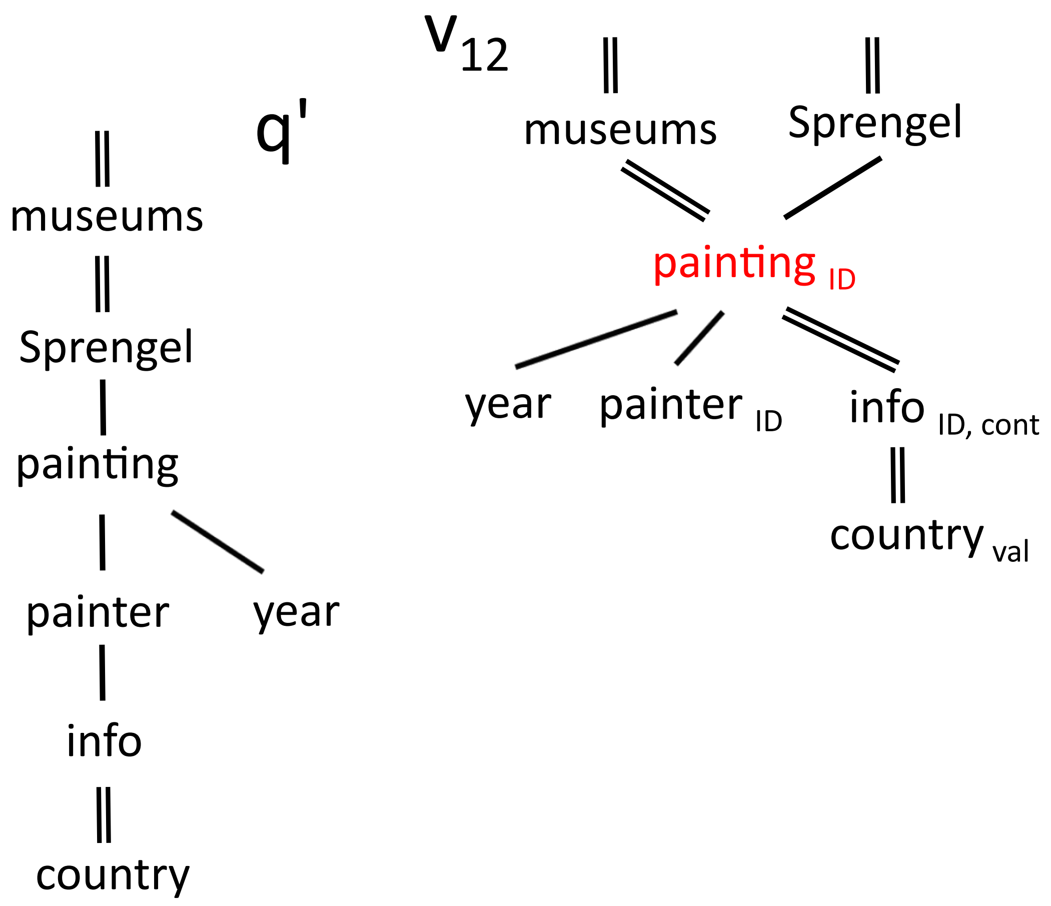
Tree pattern joining



Tree pattern joining

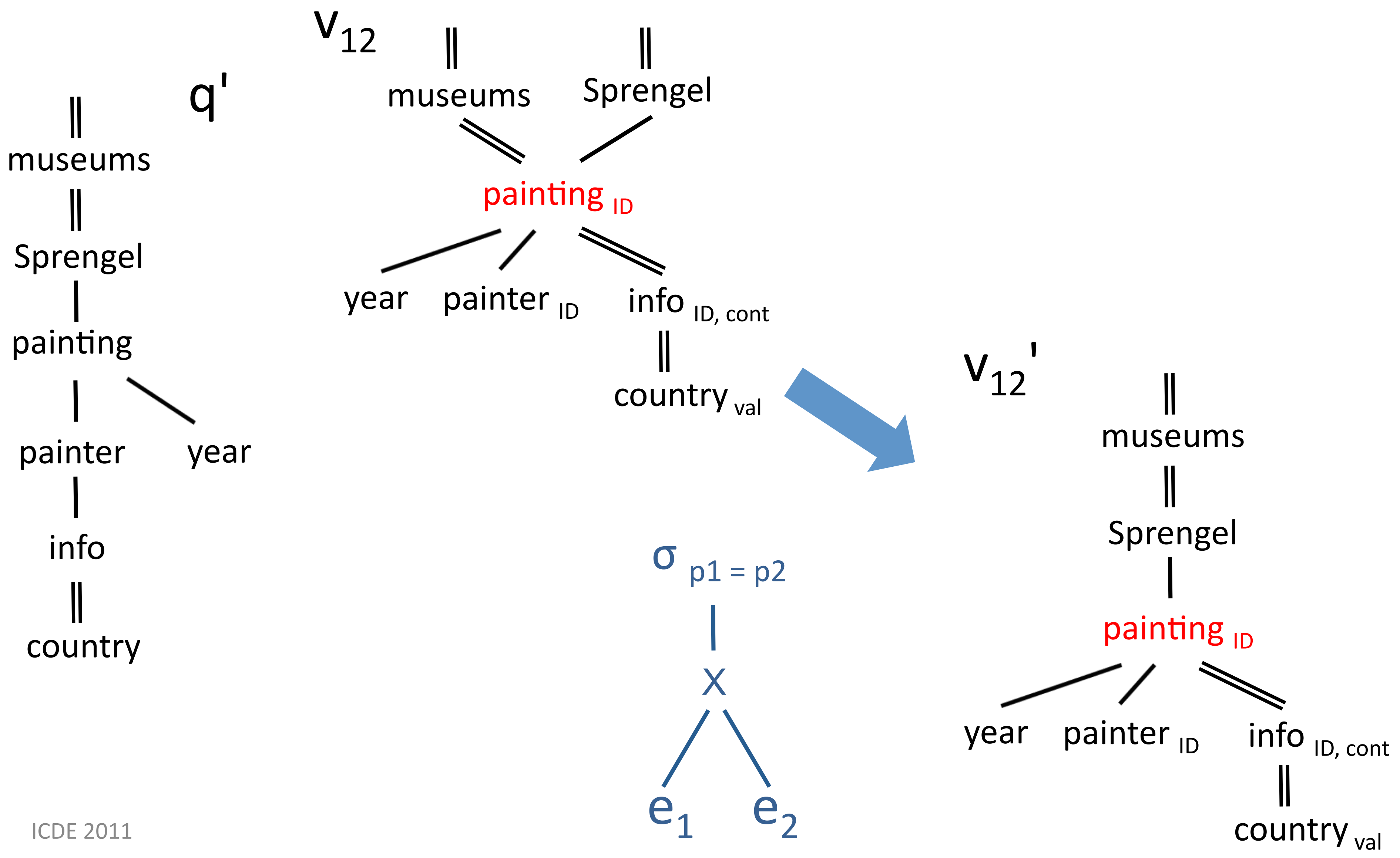


Zipping

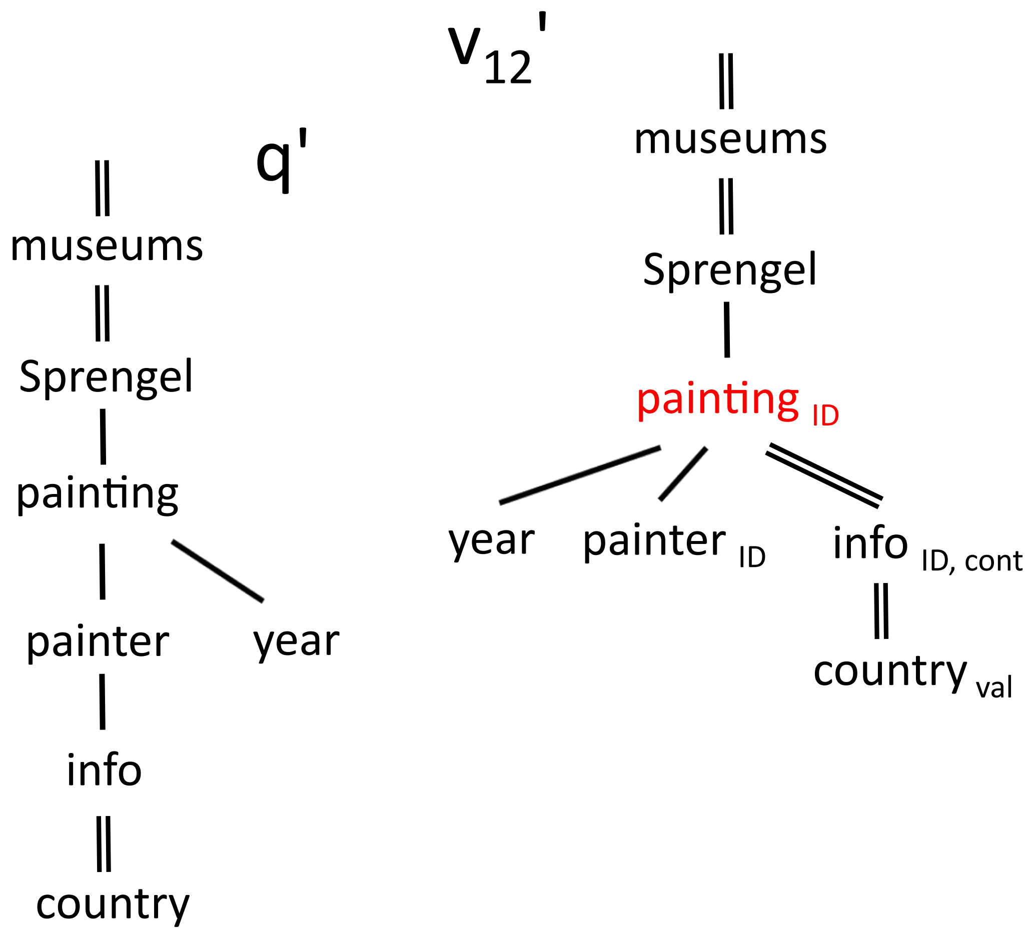


q'
museums
Sprengel
painting
painter
year
info
country
v_{12}
museums
Sprengel
painting ID
year
painter ID
info ID, cont
country val

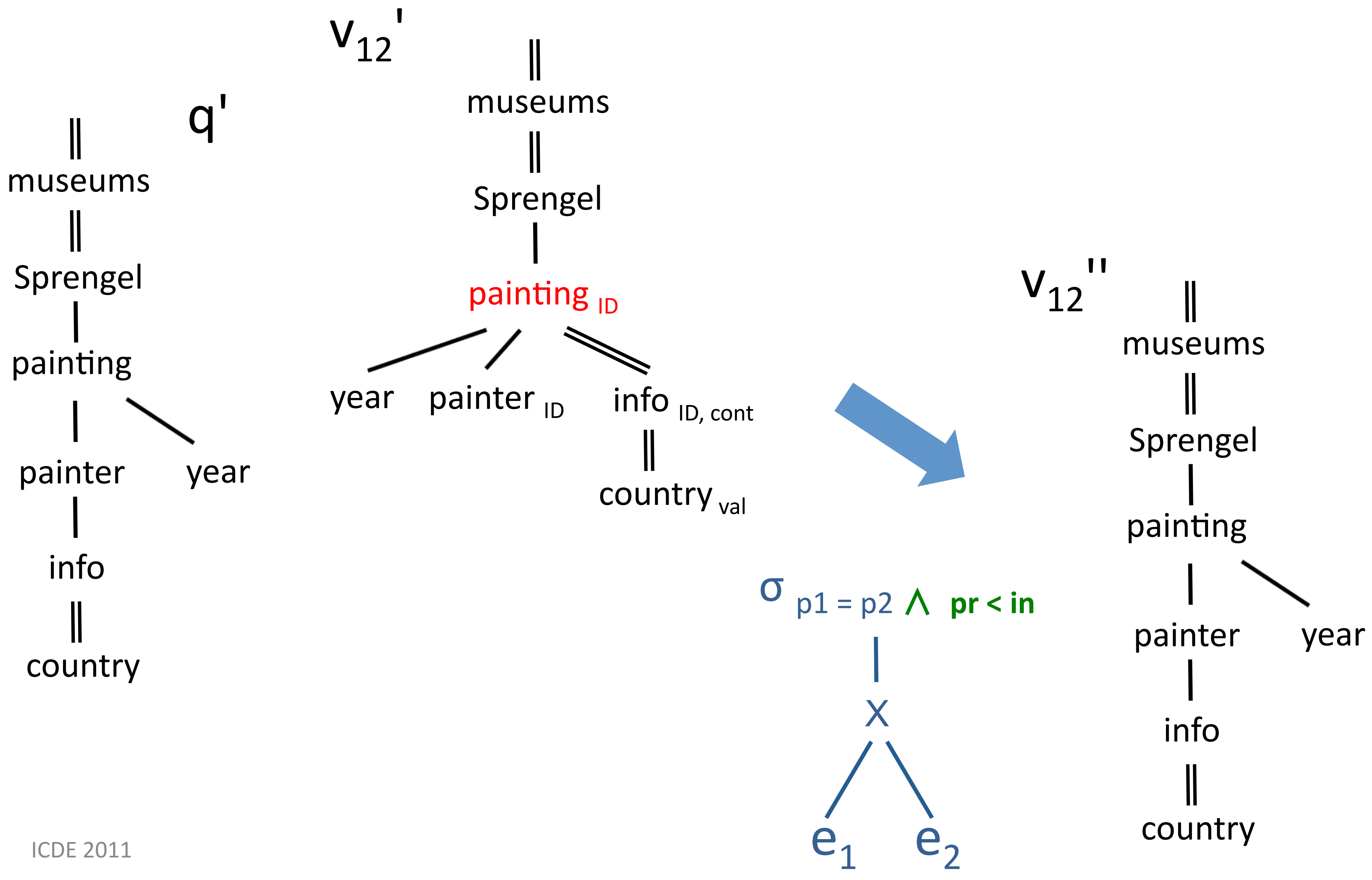
Zipping


q'
museums
Sprengel
painting
painter
year
info
country
v12
museums
Sprengel
painting ID
year
painter ID
info ID, cont
country val
v12'
museums
Sprengel
painting ID
year
painter ID
info ID, cont
country val
σ p1 = p2
X
e1
e2

Merging


q'
museums
Sprengel
painting
painter
year
info
country
v_{12}'
museums
Sprengel
painting ID
year
painter ID
info ID, cont
country val

Merging

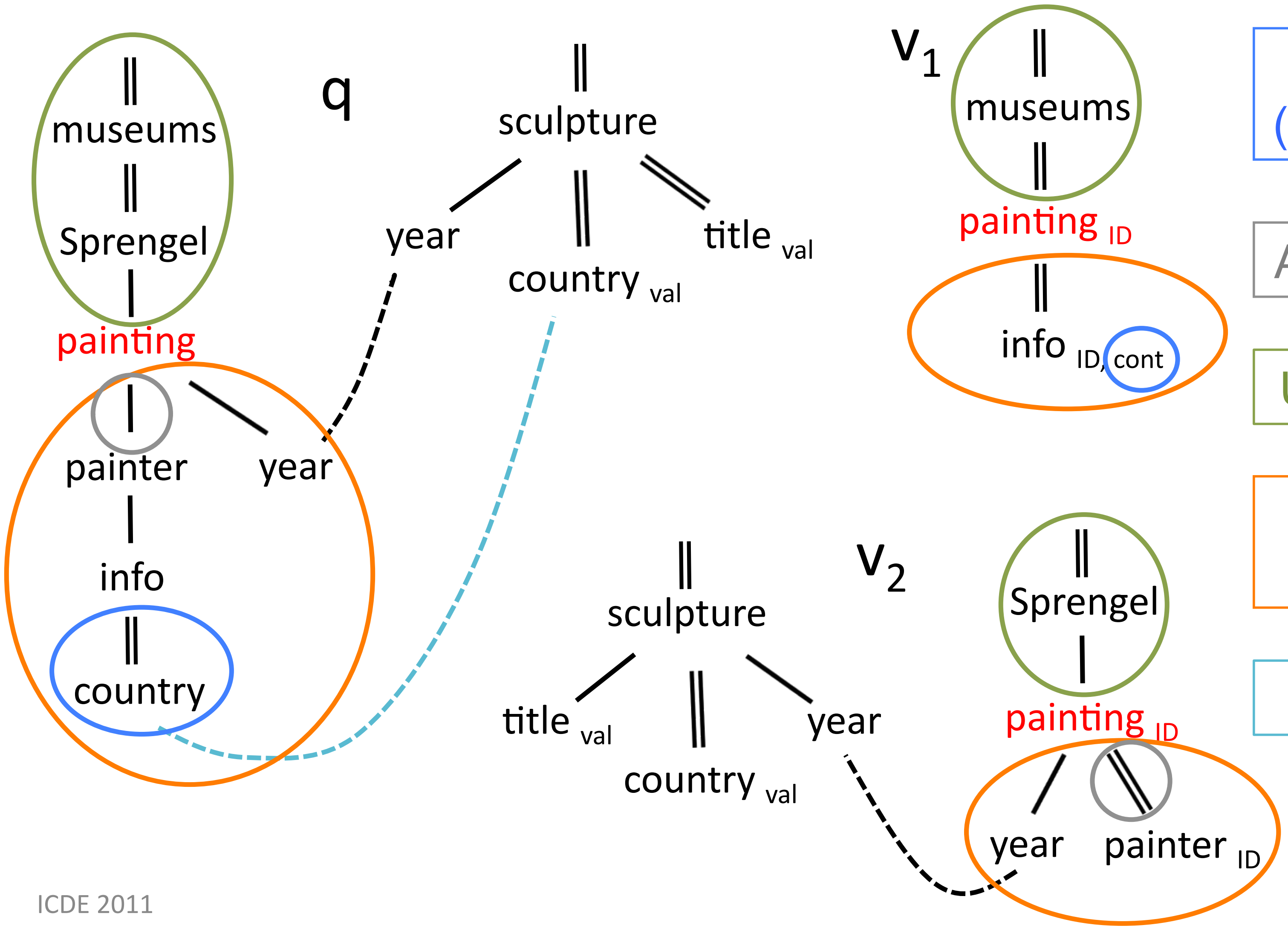


q'
museums
Sprengel
painting
painter
year
info
country
v_{12}'
museums
Sprengel
painting $_{ID}$
year
painter $_{ID}$
info $_{ID, cont}$
country $_{val}$
v_{12}''
museums
Sprengel
painting
painter
year
info
country
$\sigma_{p1 = p2 \wedge pr < in}$
X
e_1
e_2

Rewriting strategies

- Strategy: order of partial rewriting joins
- Naïve dynamic programming (NDP)
  - rewritings of *k* views only after all of (*k-1*) views
- Query-driven dynamic programming (QDP)
  - joins inspired by query nodes and edges
- Query-driven depth-first (QDF)
  - greedily cover the biggest part of the query
- Strategies features:
  - Output only minimal rewritings
  - NDP and QDP find min-size rewritings
  - QDF may reach a solution more quickly
  - upper bound on the number of joins in an equivalent rewriting

Motivating example



Navigation (compensation) ✓

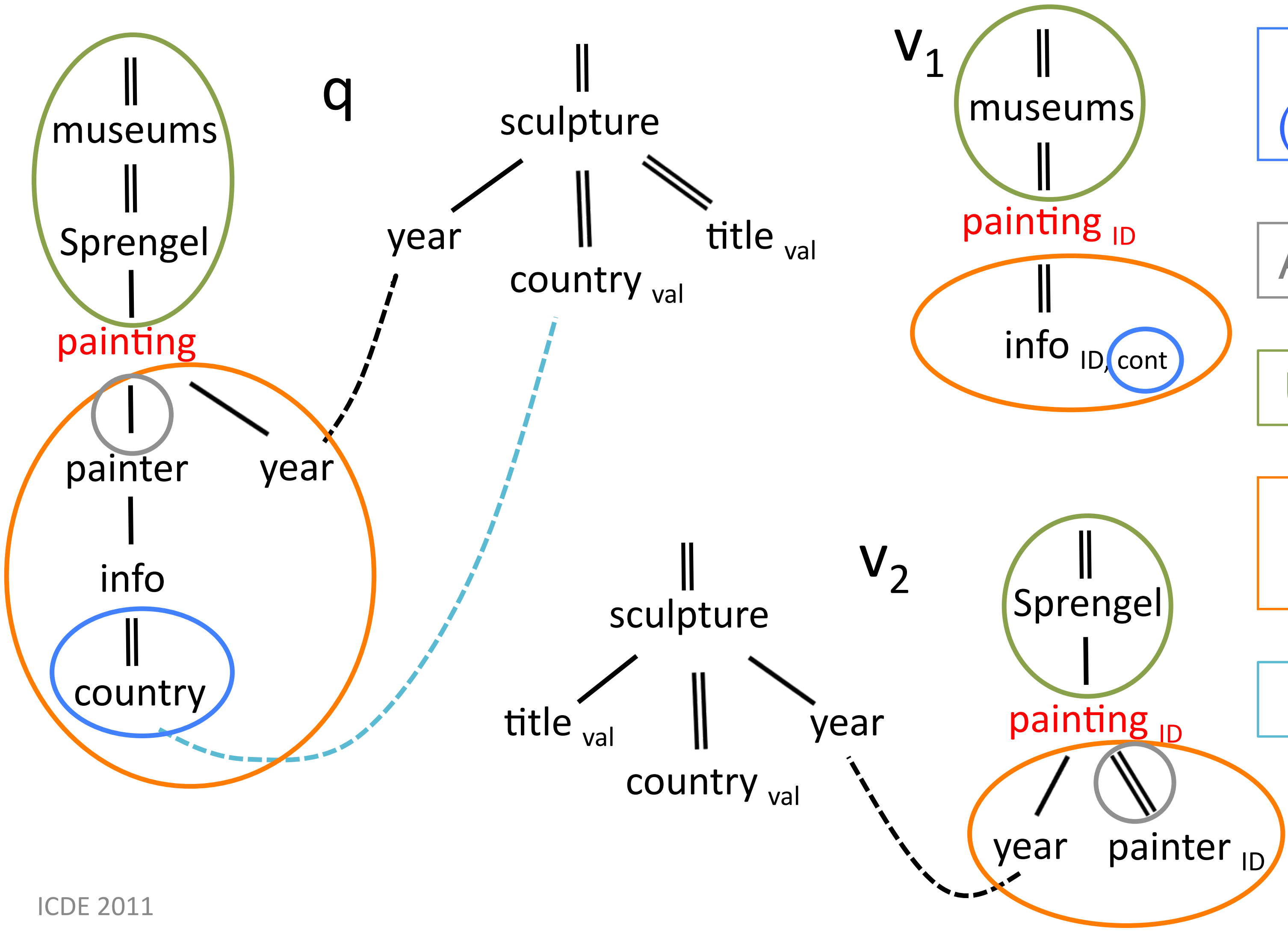
Axis adaptation ✓

Upward joining

Downward joining

Value joins

Motivating example



Navigation (compensation) ✓

Axis adaptation ✓

Upward joining ✓

Downward joining ✓

Value joins

Outline

- Introduction
- Motivating example
- Tree pattern rewriting
- **Joined pattern rewriting**
- Experiments
- Conclusion

Joined pattern rewriting

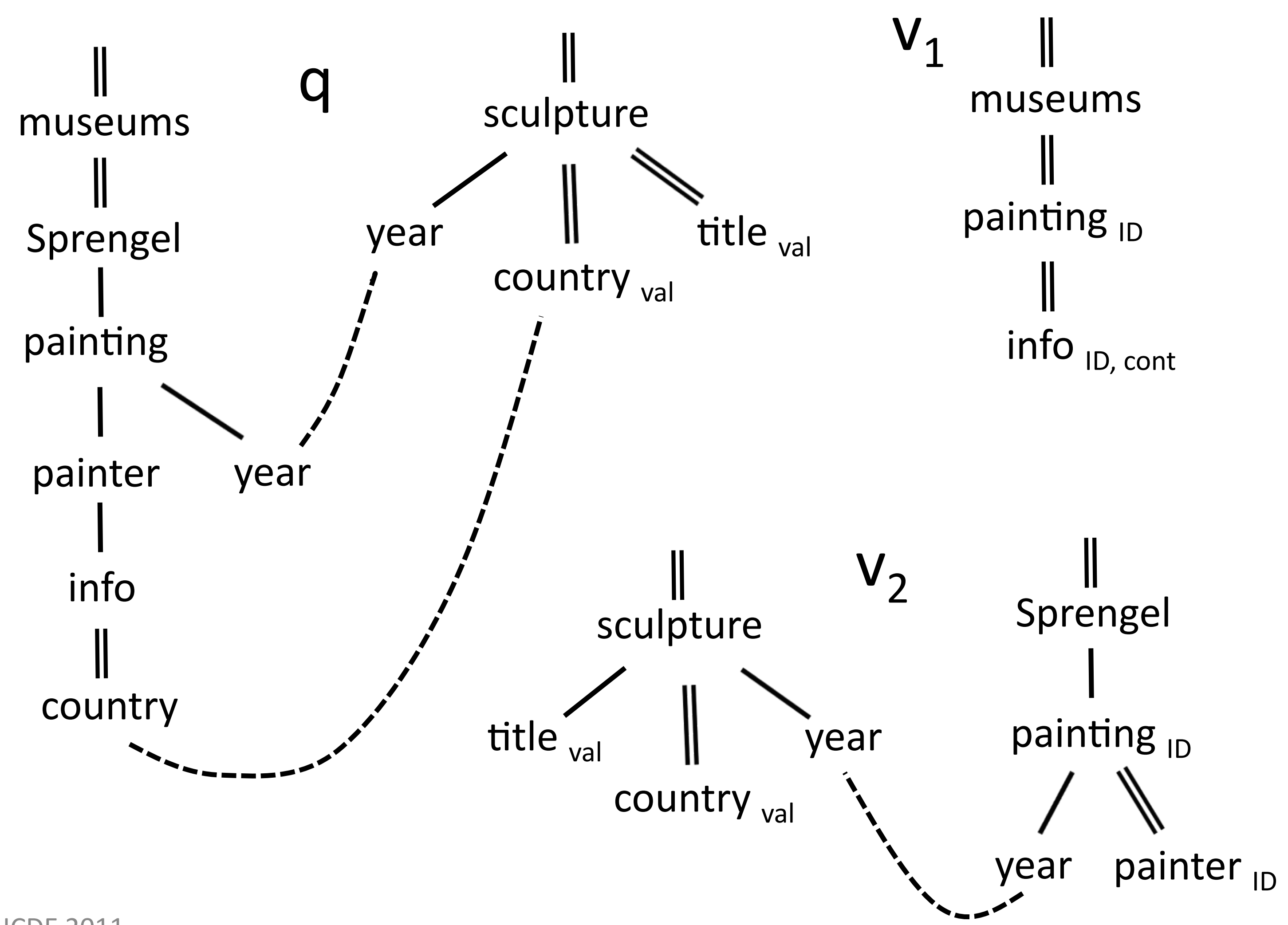
- Let *jq* a joined query and *JV* a joined pattern view set

Joined pattern queries	**Conjunctive queries**
tree pattern	query atom
value join edge	join predicate (shared variable)

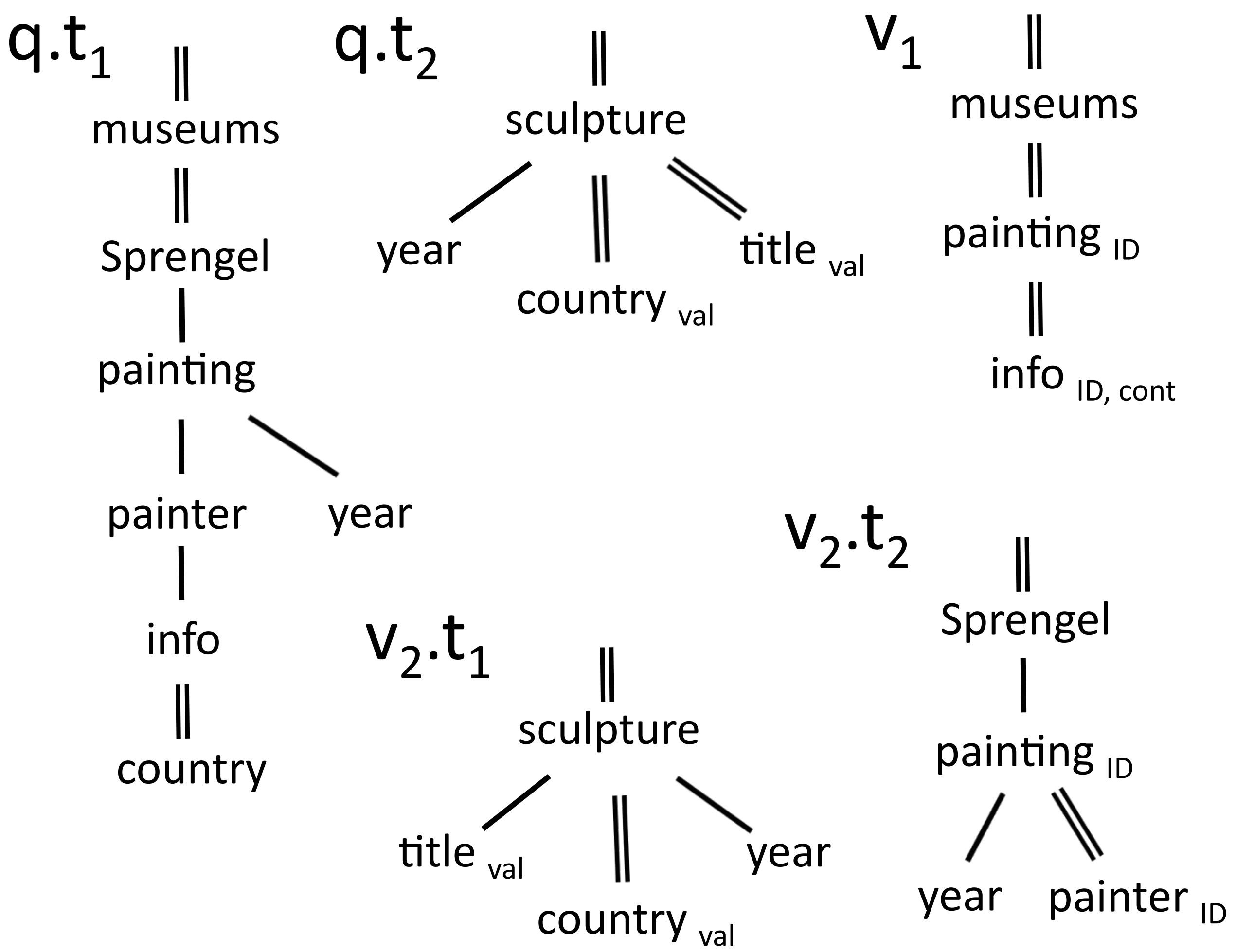
- Bucket-style algorithm:

  Rewrite each query tree pattern and combine the solutions.
- Correct and complete algorithm

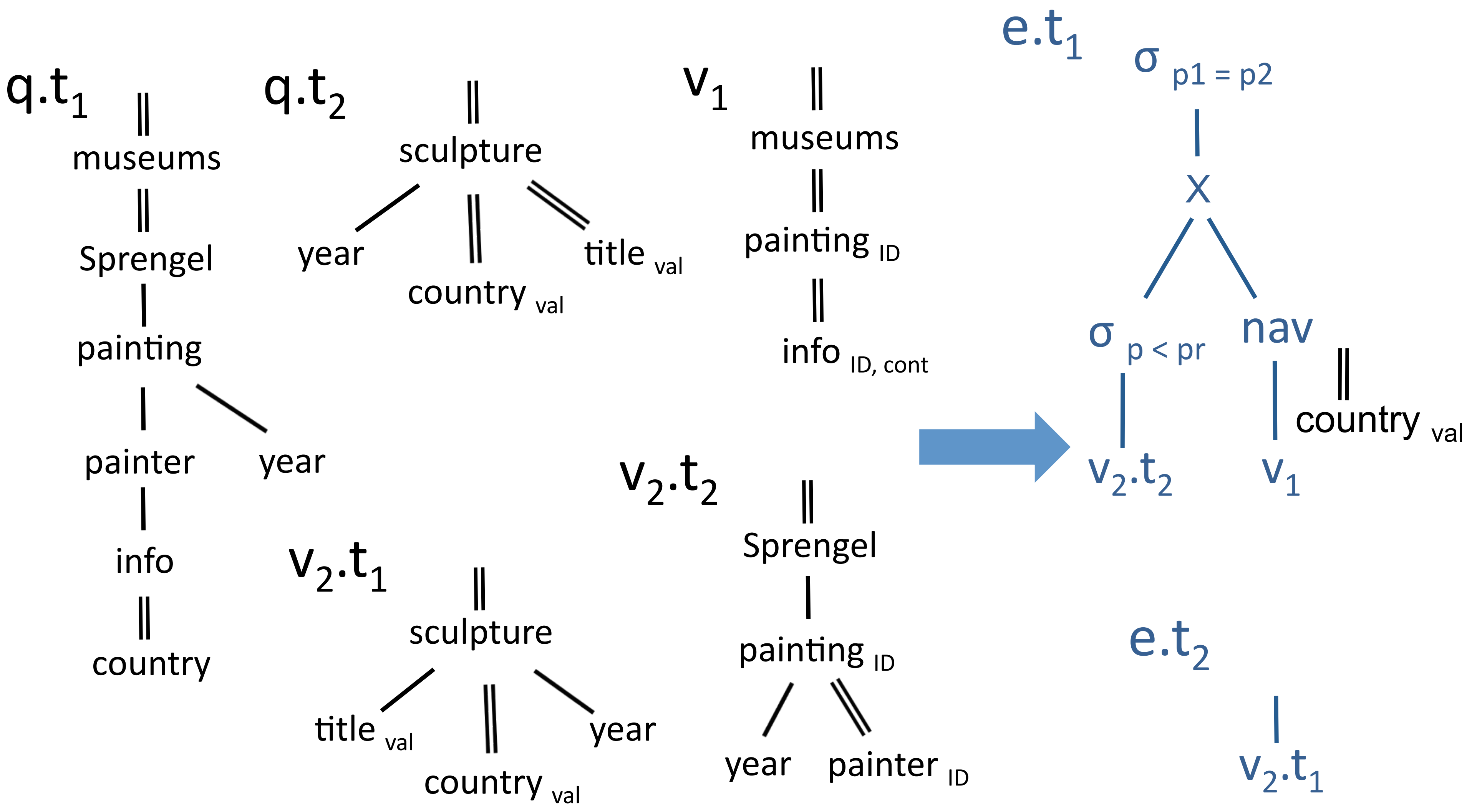
Joined pattern rewriting



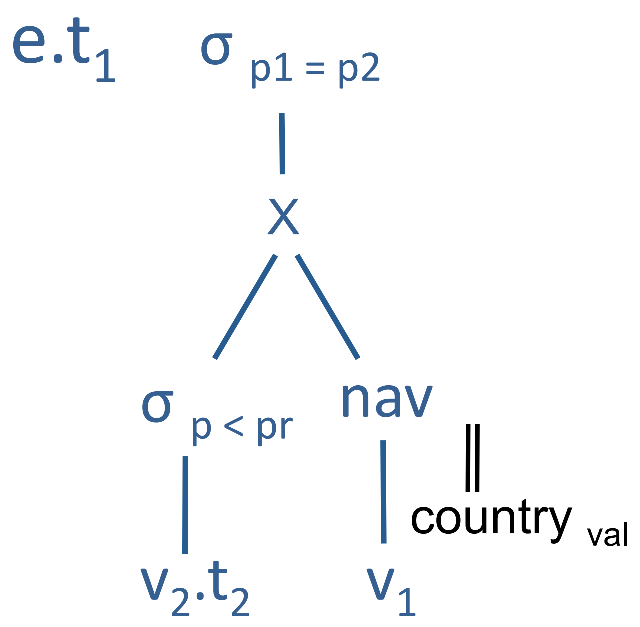
Joined pattern rewriting

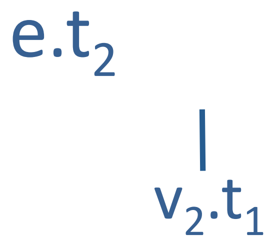


Joined pattern rewriting

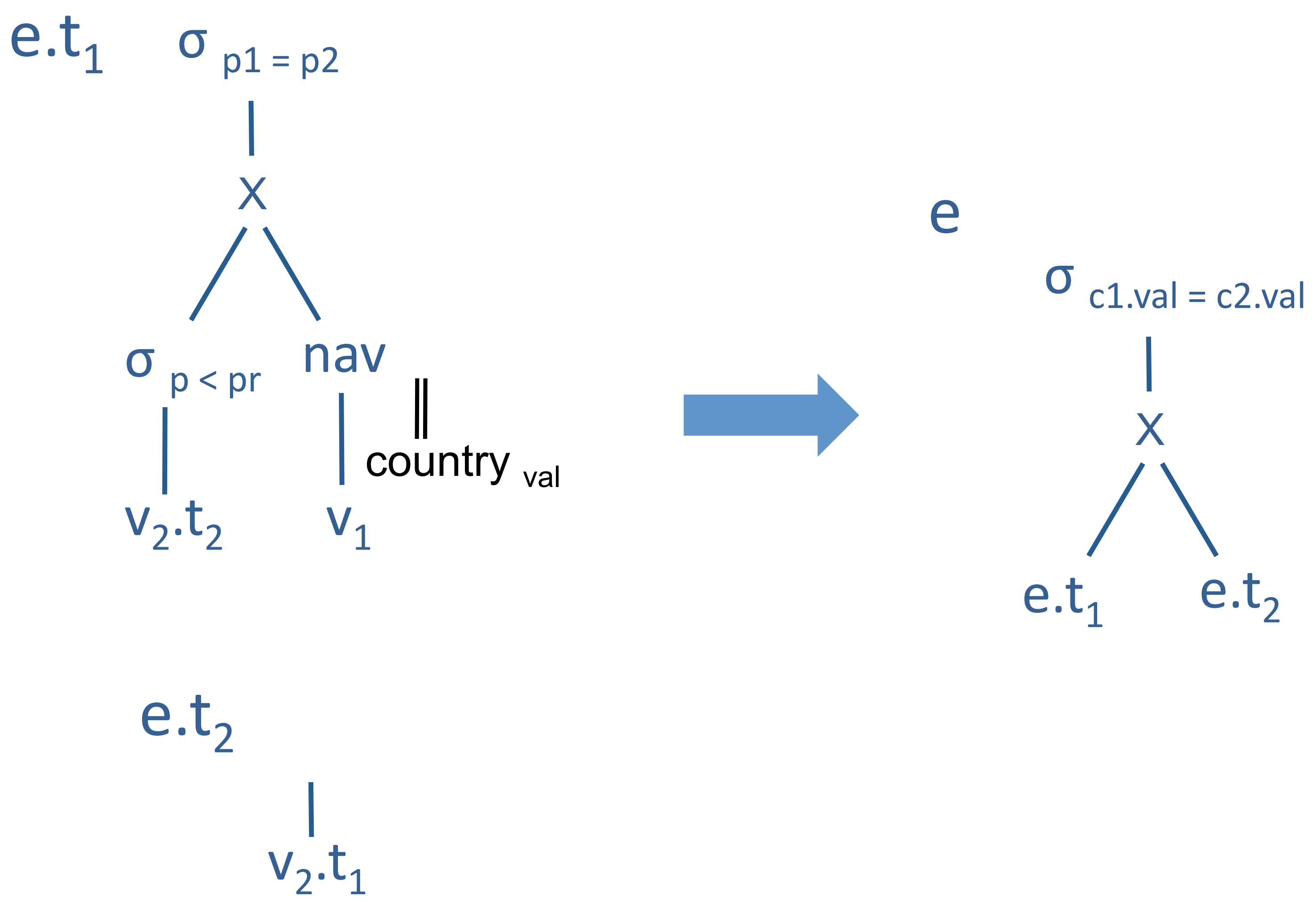


Joined pattern rewriting

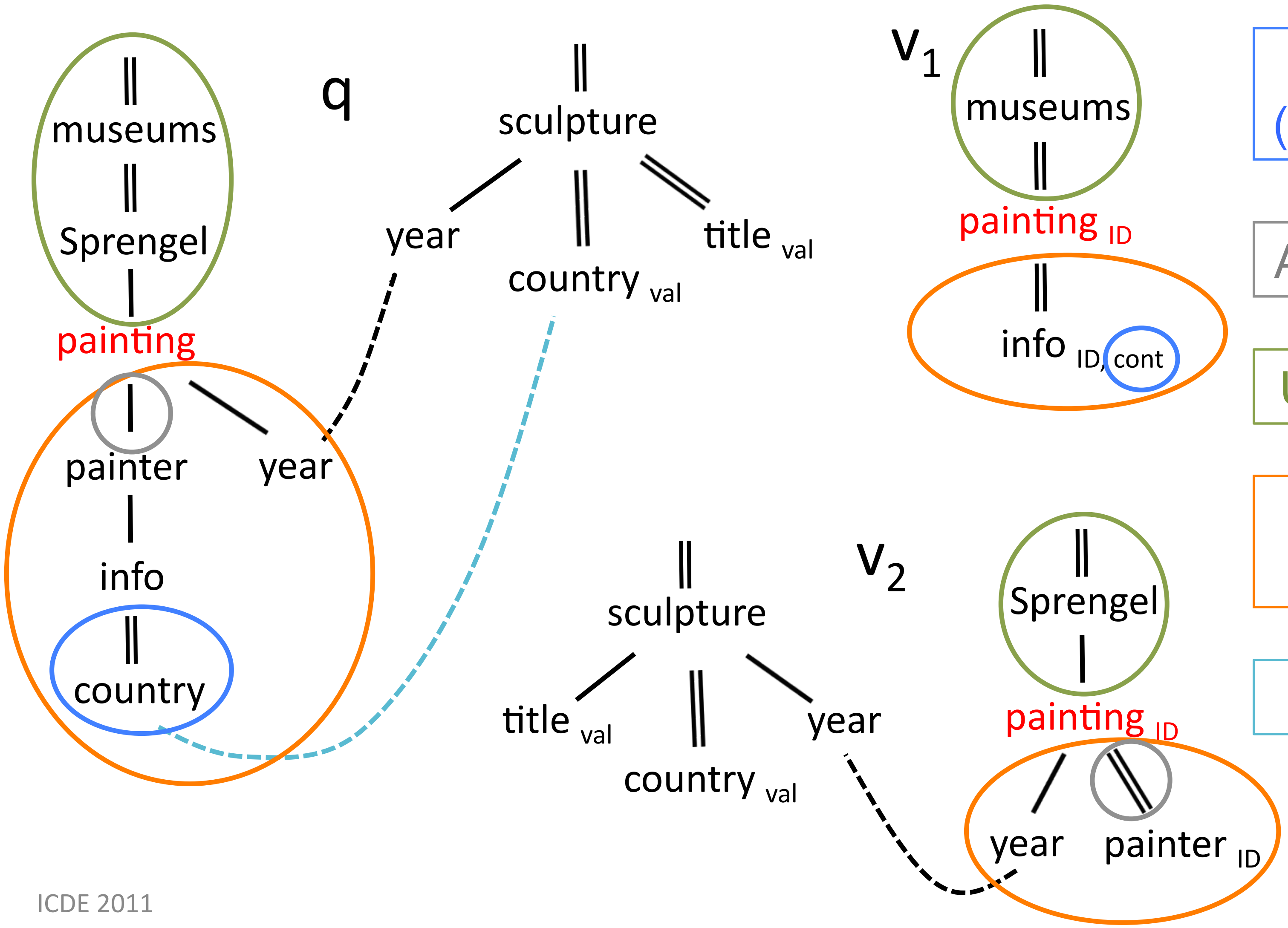




Joined pattern rewriting



Motivating example



Navigation (compensation) ✓

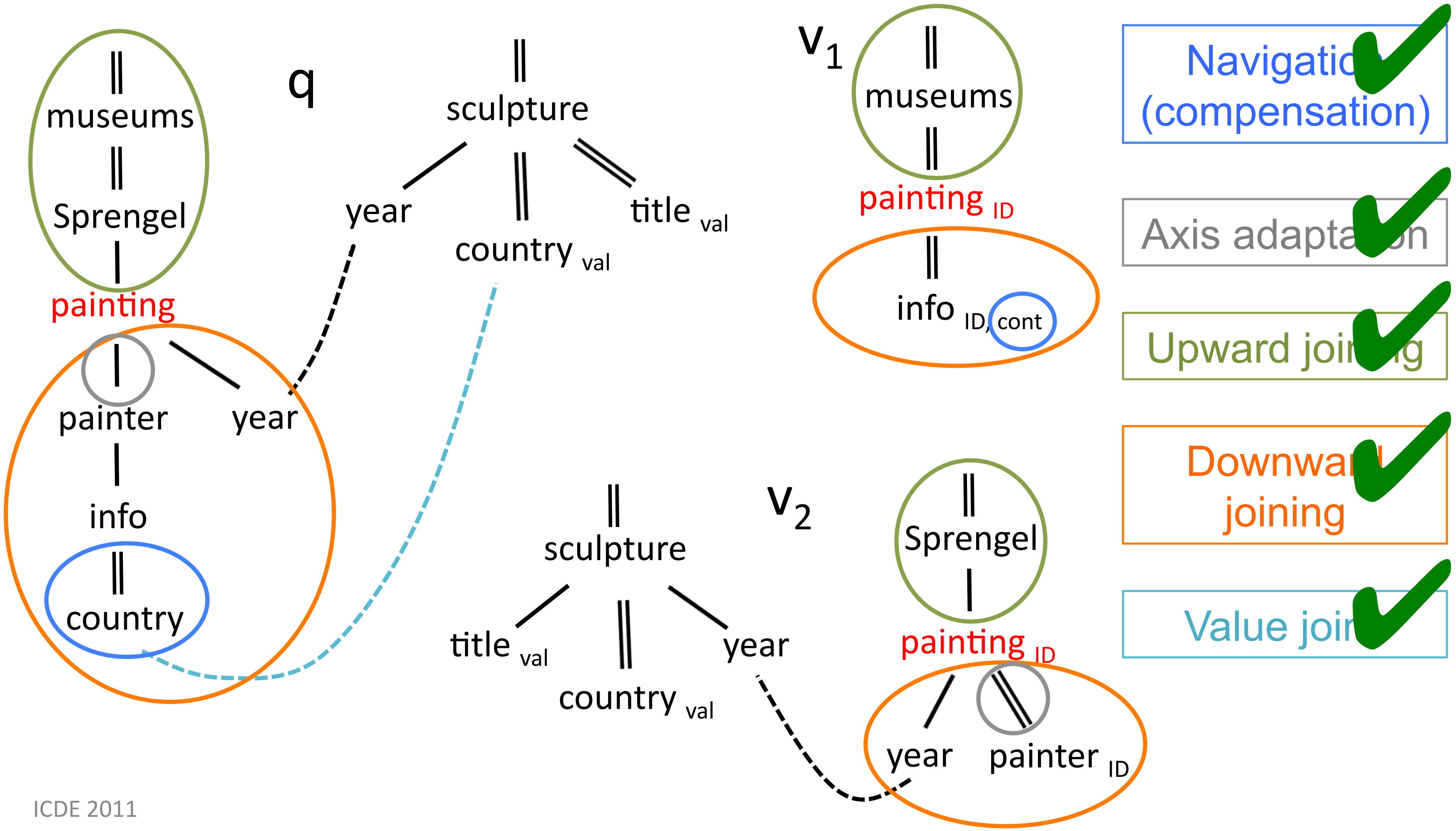
Axis adaptation ✓

Upward joining ✓

Downward joining ✓

Value joins

Motivating example



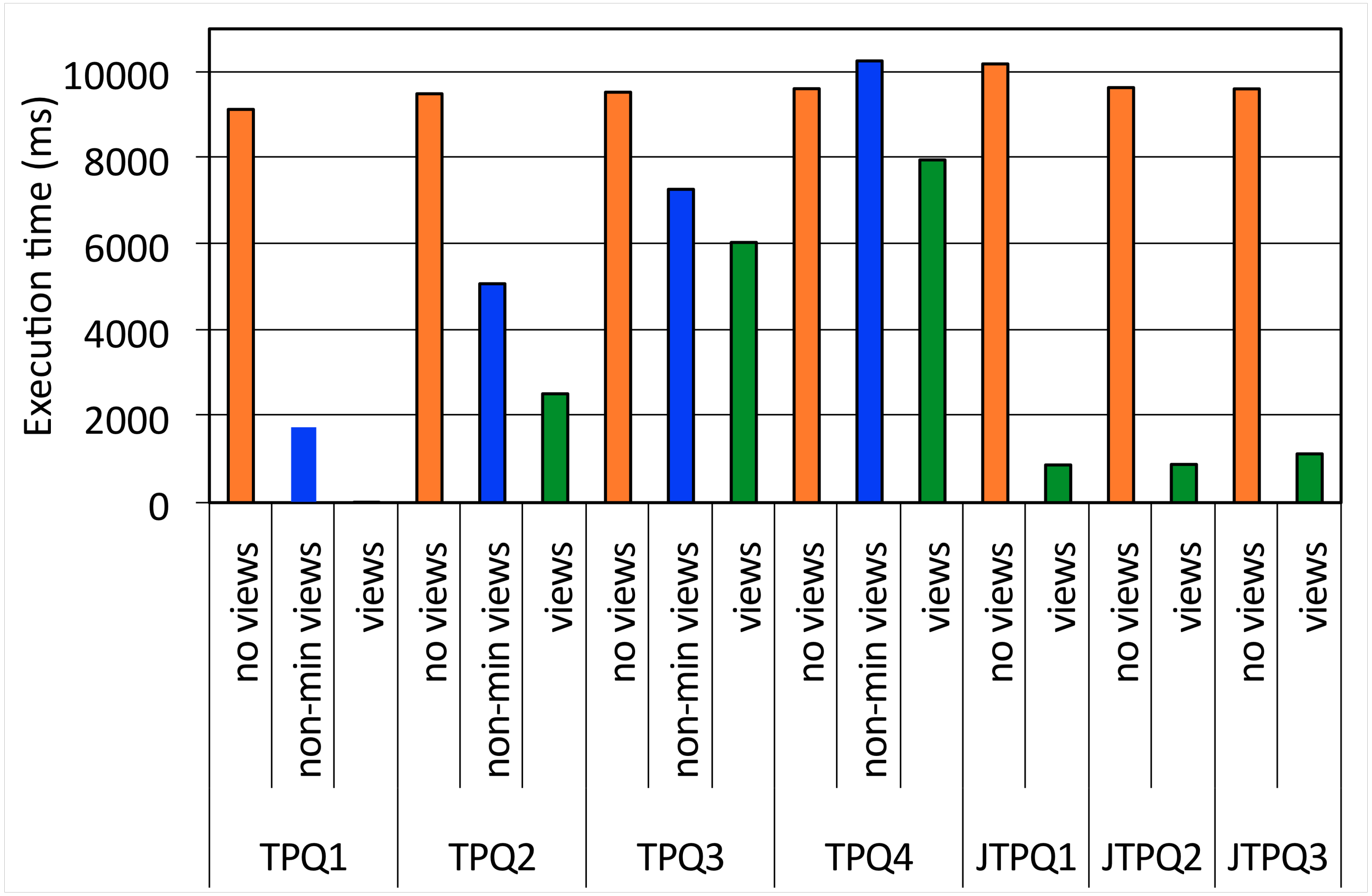
Outline

- Introduction
- Motivating example
- Tree pattern rewriting
- Joined pattern rewriting
- ➢ **Experiments**
- Conclusion

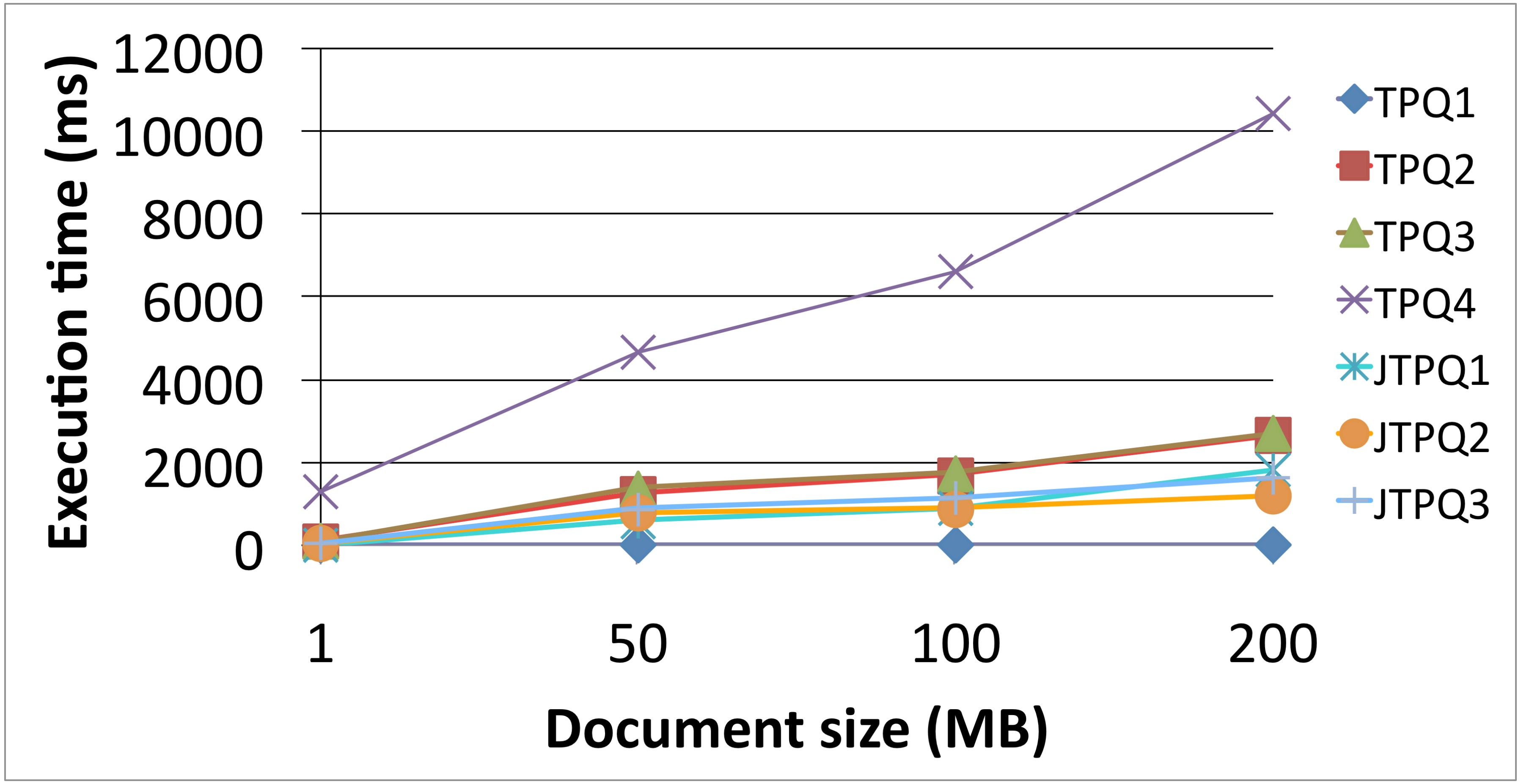
Experimental platform

- Fully implemented in Java 6
- BerkeleyDB v3.3.75 for materialized views
- Saxon v9.1 for navigation operator
- Execution engine of our ViP2P system
- XMark documents and queries

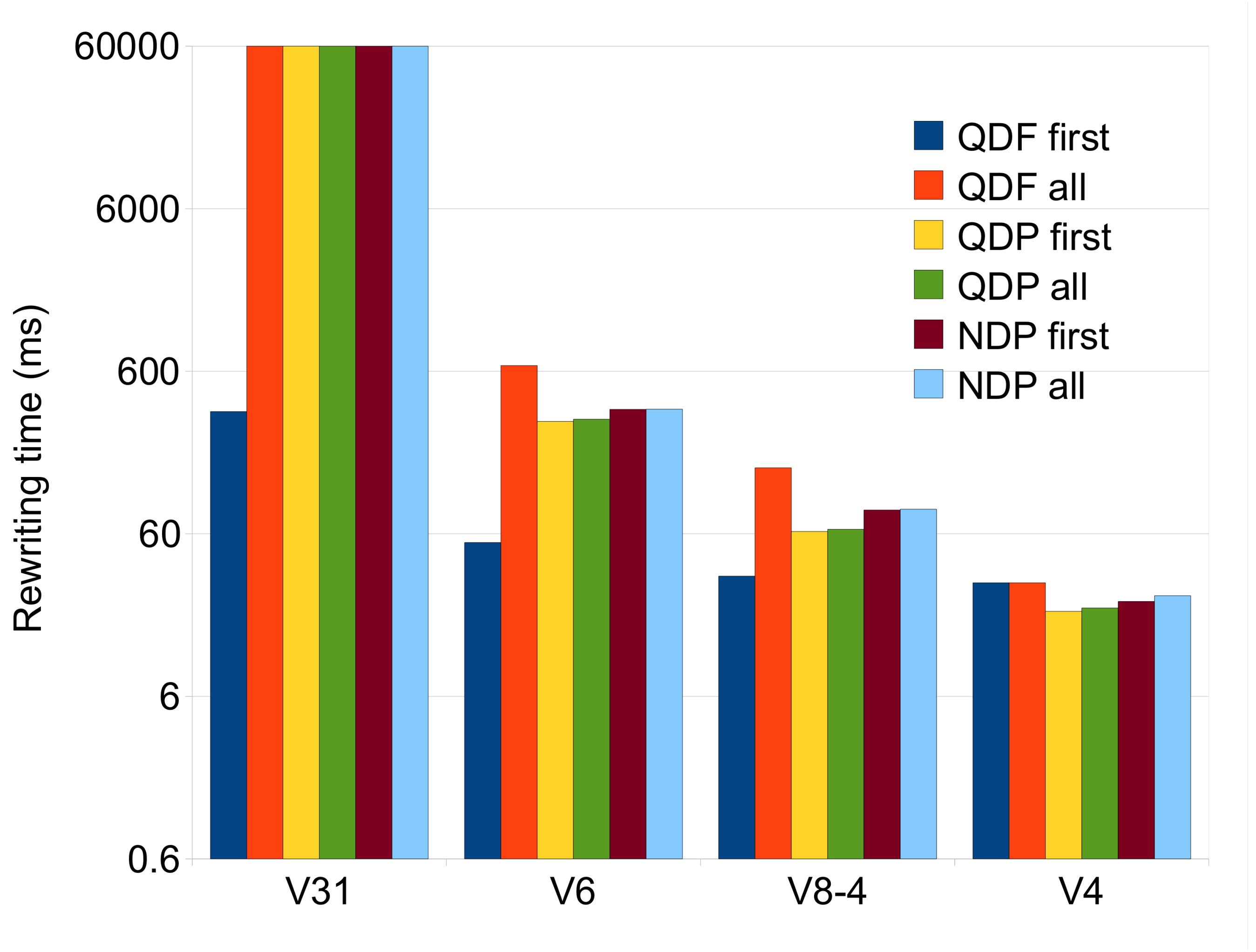
View-based query evaluation performance (100MB XMark doc)



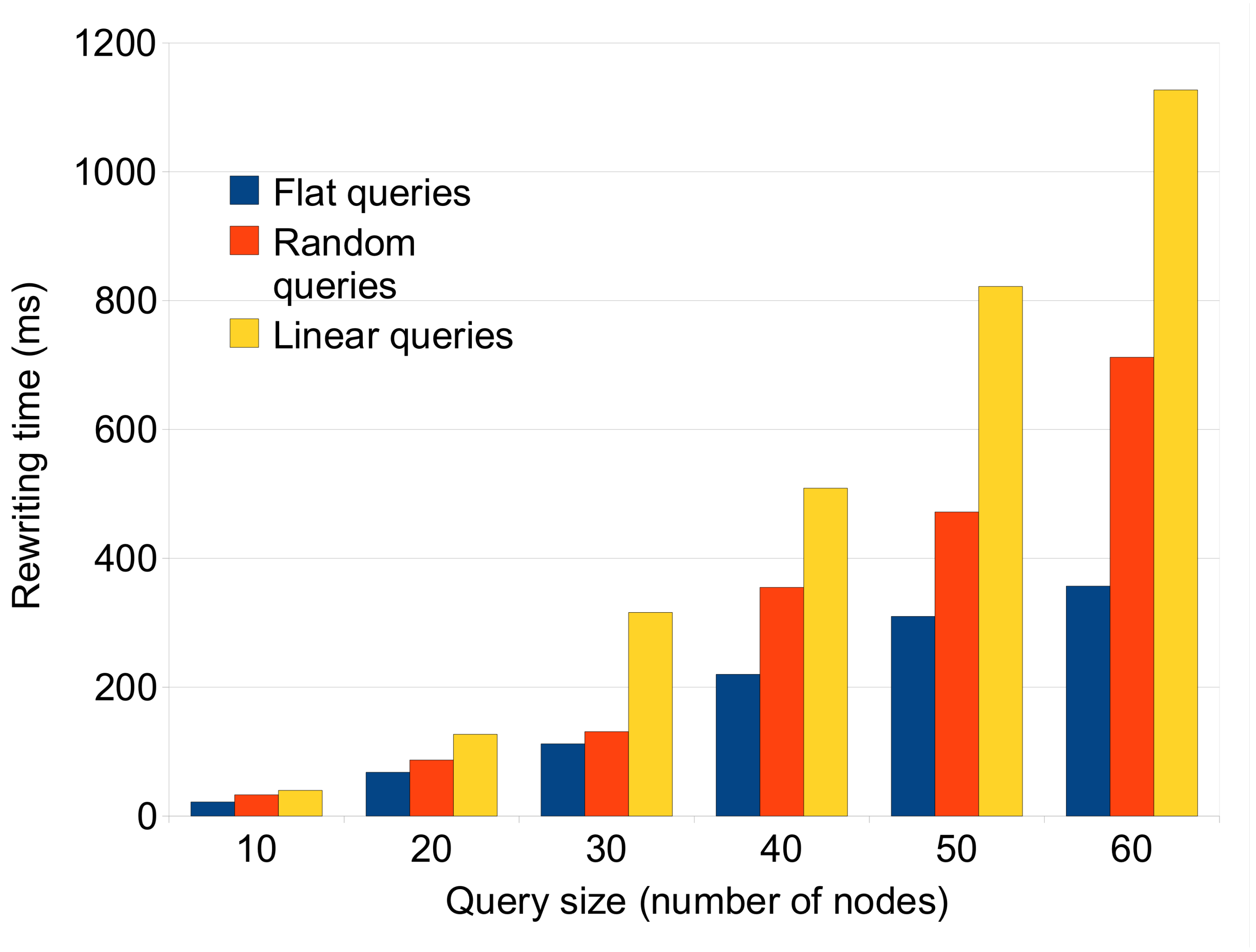
Scalability of view-based query evaluation



Rewriting strategies performance



Rewriting scalability (QDF strategy)



Outline

- Introduction
- Motivating example
- Tree pattern rewriting
- Joined pattern rewriting
- Experiments
- **Conclusion**

Conclusion

- XQuery view based rewriting
- Expressive query and view language
  - multiple returning nodes
  - value joins
- Equivalent, minimal and complete rewritings, expressed in a generic algebra
- Experimental evaluation
  - benefit of views
  - scalability of our approach

Related works

- B. Cautis, A. Deutsch, and N. Onose, "XPath rewriting using multiple views: Achieving completeness and efficiency," in *WebDB, 2008*
- N. Tang, J.X. Yu, M.T. Ozsu, B. Choi and K.-F. Wong, "Multiple materialized view selection for XPath query rewriting," in *ICDE, 2008*
- A. Arion, V. Benzaken, I. Manolescu and Y. Papakonstantinou, "Structured materialized views for XML queries," in *VLDB, 2007*
- A. Balmin, F. Ozcan, K. Beyer, R. Cochrane and H. Pirahesh, "A framework for using materialized XPath views in XML query processing," in *VLDB, 2004*

Thank you!