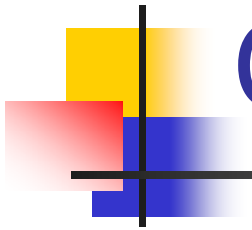
A Publish & Subscribe Architecture for Distributed Metadata Management

Markus Keidl[1] Alexander Kreutz[1] Alfons Kemper[1] Donald Kossmann[2]

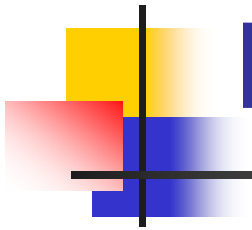
[1] Universität Passau
D-94030 Passau
<lastname>@db.fmi.uni-passau.de

[2] TU München
D-81667 München
kossmann@in.tum.de



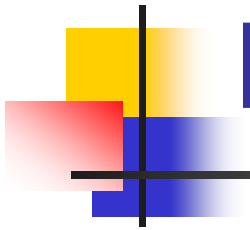
Outline

- Motivation
- The MDV system
- The publish & subscribe algorithm
- Conclusion



Motivation

- Resource management in ObjectGlobe
- Requirements:
  - Large number of clients
    $\Rightarrow$ 3-tier architecture
  - Information close to the clients
    $\Rightarrow$ caching
  - Up-to-date information

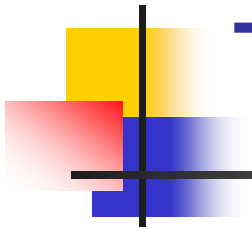


RDF and RDF Schema

- RDF = Resource Description Framework
  - W3C Recommendation
  - Defines resources, properties, and values
- RDF Schema
  - W3C Candidate Recommendation
  - Defines schema of metadata, similar to class hierarchy

RDF Example: doc.rdf

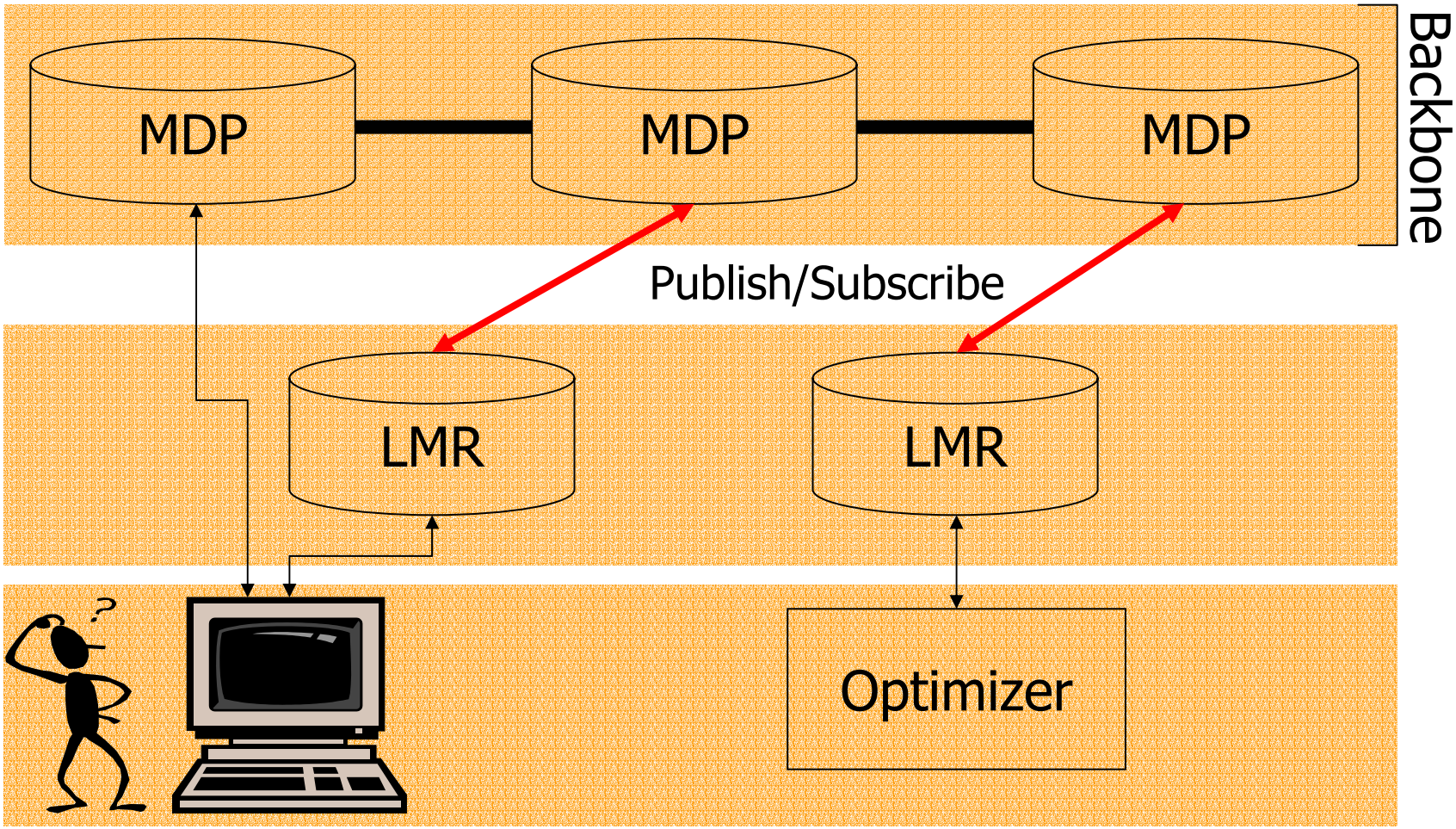
```
<CycleProvider rdf:ID="host1">
  <serverHost>pirates.uni-passau.de</>
  <serverPort>5874</serverPort>
  <serverInformation>
    <ServerInformation rdf:ID="info1"
        memory="92" cpu="600" />
  </serverInformation>
</CycleProvider>
```

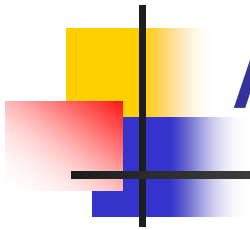


The MDV System

- Metadata: RDF and RDF Schema
- 3-tier Architecture:
  MDPs, LMRs, and MDV Clients
- Caching on local tier
- Up-to-date metadata by using a publish & subscribe mechanism

Architecture Overview

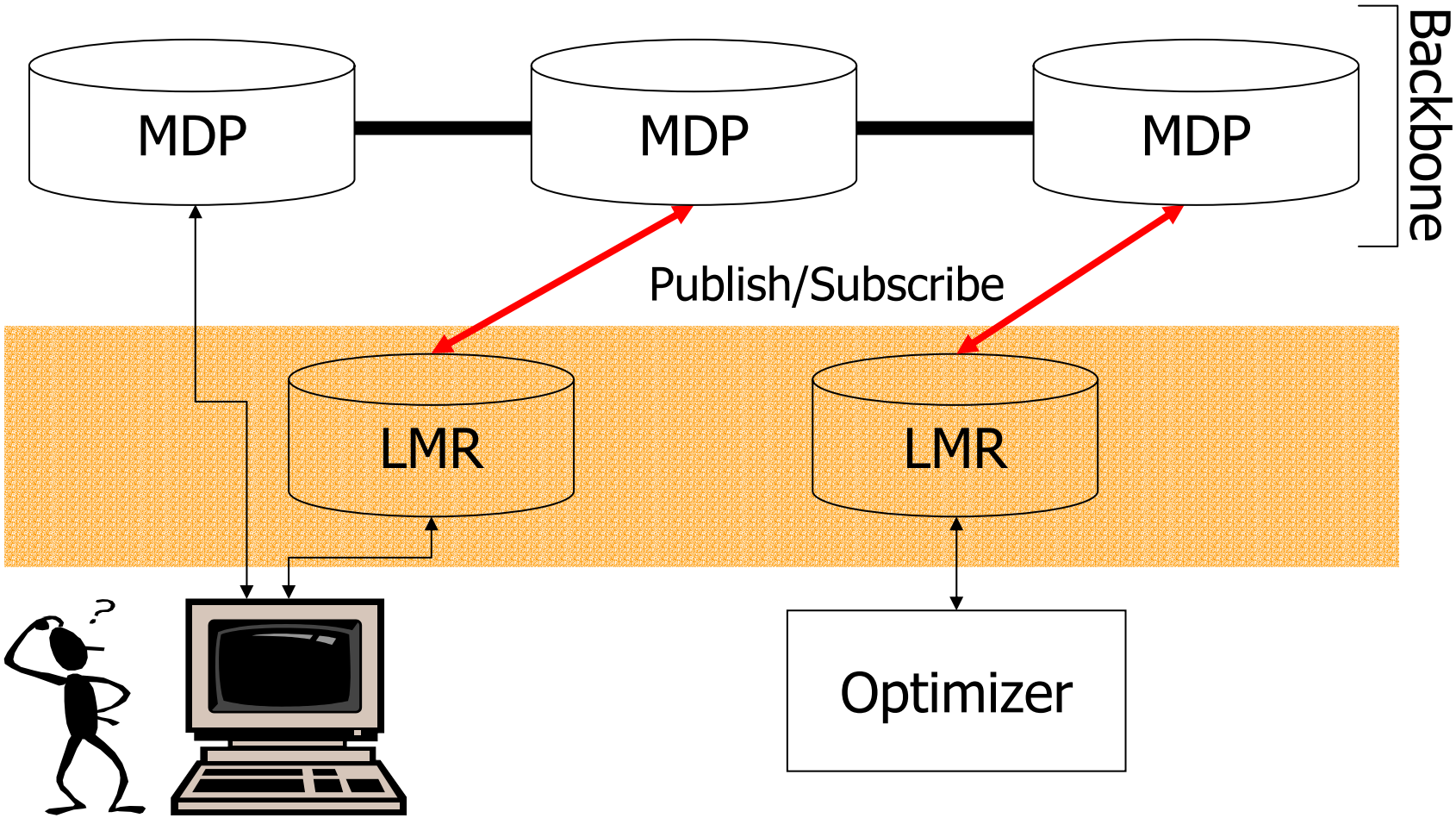


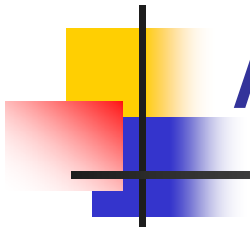


Architecture: MDPs

- MDP = Metadata Provider
- Backbone of MDPs
  - Sharing the same schema
  - Full Replication of metadata
- Metadata: globally and publicly available
- Registration, update, deletion of metadata

Architecture Overview

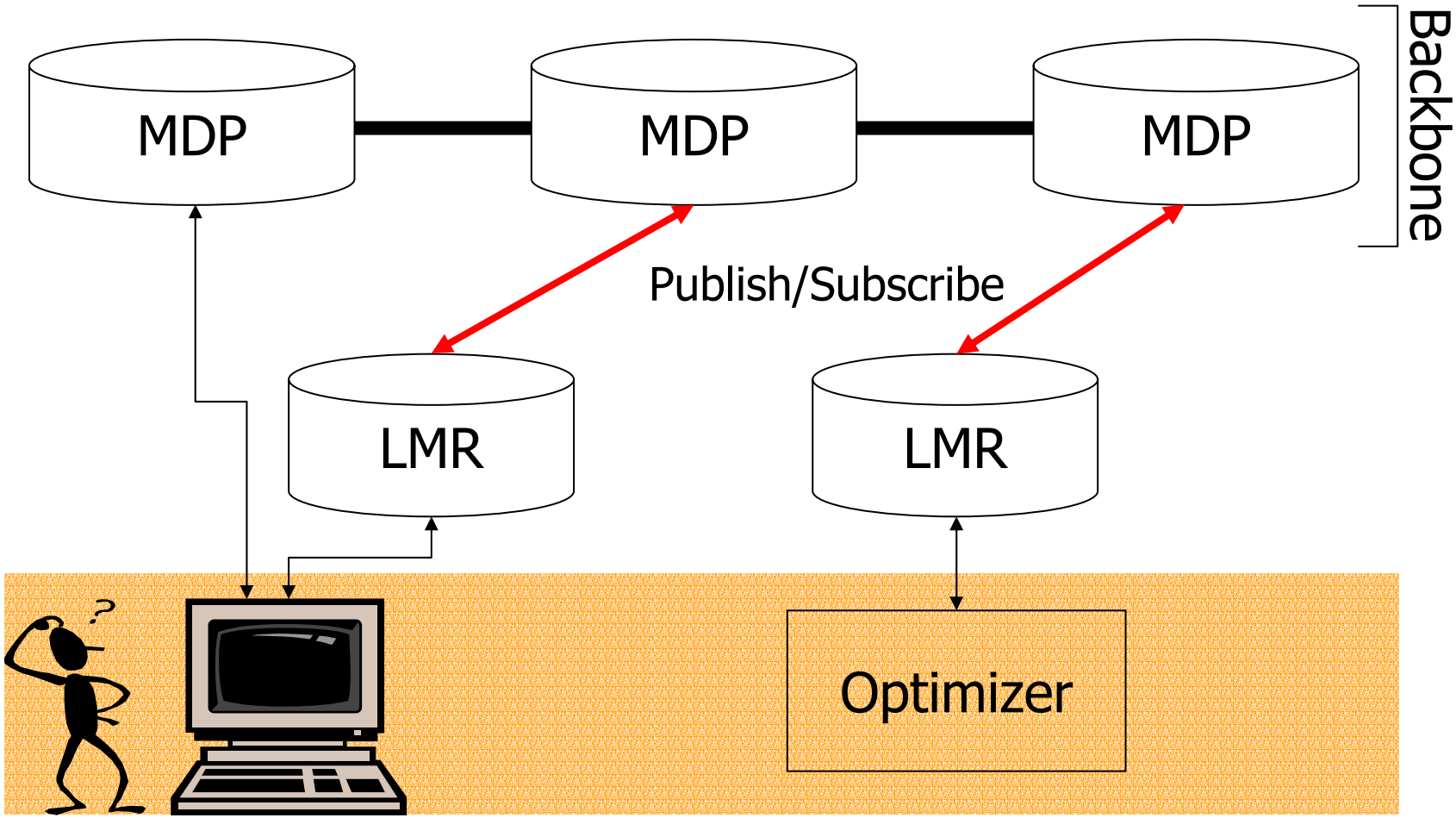




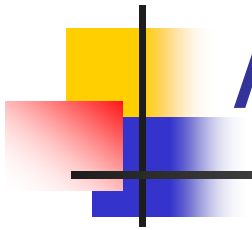
Architecture: LMRs

- LMR = Local Metadata Repository
- Metadata
  - Caching of global metadata
    $\Rightarrow$ publish & subscribe
  - Storing of local metadata
    $\Rightarrow$ only locally accessible, for sensitive data
- Query processing
  - Declarative language
  - Cached and local metadata

Architecture Overview



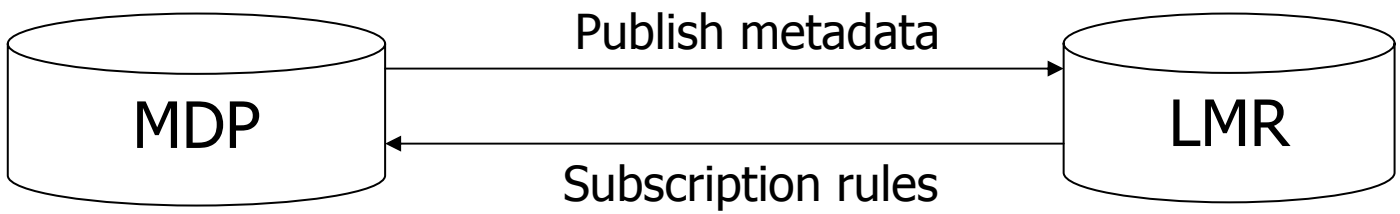
Architecture: MDV Clients



- Pose queries to LMRs
- Browse metadata at MDPs and LMRs
  $\Rightarrow$ determine metadata that should be cached
- Modify subscription rules of LMRs

The Publish & Subscribe System

- Based on subscription rules:
  - LMRs subscribe to metadata (at MDPs)
  - MDPs determine which metadata to publish (to LMRs)



- Insertion, update, or deletion of metadata $\Rightarrow$ Evaluation

Subscription Rule Language

- Operators: =, !=, <, <=, >, >=, **`contains`**
- Example:

```
search      CycleProvider c
register    c
where       c.serverHost contains 'uni-passau.de'
            and c.serverInformation.memory > 64
```

- Joins
- Input: document + complete database
- Publish: resources, not documents

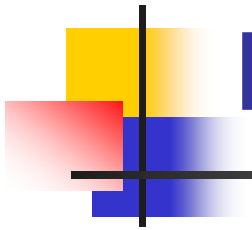
References

```
search      CycleProvider c
register    c
where       c.serverHost contains 'uni-passau.de'
            and c.serverInformation.memory > 64
```

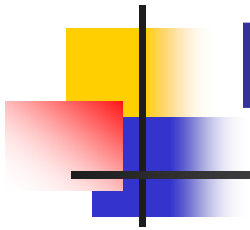
- Problem: only subscription of CycleProvider resources
- What's with ServerInformation resources?

```
<CycleProvider rdf:ID="host1">
  <serverHost>pirates.uni-passau.de</>
  <serverPort>5874</serverPort>
  <serverInformation>
    <ServerInformation rdf:ID="info1"
        memory="92" cpu="600" />
  </serverInformation>
</CycleProvider>
```

References - Solution



- Augmentation of RDF schema
  $\Rightarrow$ "user-defined" dangling references
- Strong references:
  - transmitted together with referencing resource
  - garbage collector deletes superfluous resources at LMR
- Weak references:
  - never transmitted with referencing resource



Publish & Subscribe Algorithm

- Problem: huge set of subscription rule
- Solution: index on complete set of rules
- Goal: evaluation of a subset of all subscription rules
- Based on standard RDBMS technology

Basic Approach

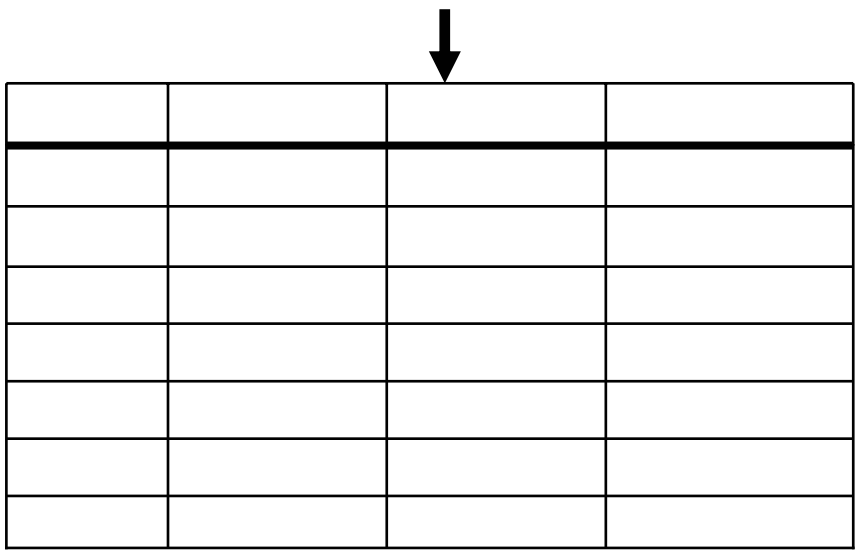
RDF Document

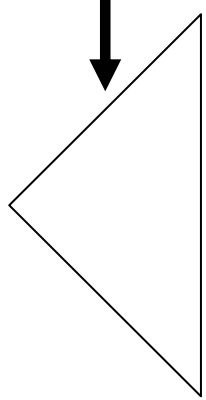
```
<CycleProvider rdf:ID="host1">
  <serverHost>pirates.uni-passau.de</>
  <serverPort>5874</serverPort>
  <serverInformation>
    <ServerInformation rdf:ID="info1"
          memory="92" cpu="600" />
  </serverInformation>
</CycleProvider>
```

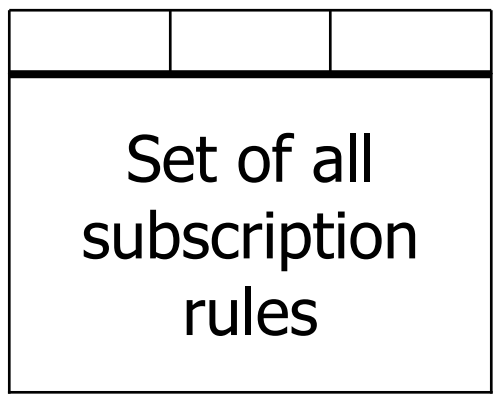
Subscription Rule

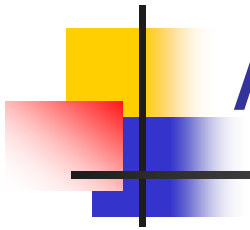
search CycleProvider c **register** c **where** c.serverHost **contains** 'uni-passau.de' **and** c.serverInformation.memory > 64

Decomposition









Advantages of the algorithm

- Based on standard RDBMS technology: robustness, scalability, and query abilities
- Usage of tables, SQL, indexes, optimizer, …
- Insertions, updates, and deletions
- Support of joins

The Filter Algorithm

- Decomposition of subscription rules
- Registration of new RDF document:
  - Decomposition of the RDF document
  - Execution of algorithm:
    $\Rightarrow$ Rules that match new metadata
    +
    new metadata
  - Rules $\Rightarrow$ LMRs
  - Notification of these LMRs

Details of the Algorithm

- Decomposition into atoms
  - RDF documents
  - Rules $\Rightarrow$ triggering and join rules
- Evaluation:
  - Determination of affected triggering rules
  - Iterative evaluation of join rules
    $\Rightarrow$ calculation of transitive closure

Decomposition of Documents

```
<CycleProvider rdf:ID="host1">
   <serverHost>pirates.uni-passau.de</>
   <serverPort>5874</serverPort>
   <serverInformation>
      <ServerInformation rdf:ID="info1"
         memory="92" cpu="600" />
  </serverInformation>
</CycleProvider>
```

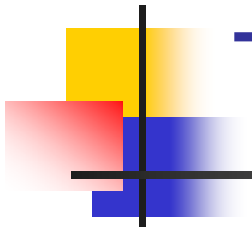


Table: FilterData

rid	class	property	value
doc.rdf#host1	CycleProvider	rdf#subject	doc.rdf#host1
doc.rdf#host1	CycleProvider	serverHost	pirates.uni-passau.de
doc.rdf#host1	CycleProvider	serverPort	5874
doc.rdf#host1	CycleProvider	serverInformation	doc.rdf#info1
doc.rdf#info1	ServerInformation	rdf#subject	doc.rdf#info1
doc.rdf#info1	ServerInformation	memory	92
doc.rdf#info1	ServerInformation	cpu	600

Details of the Algorithm

- Decomposition into atoms
  - RDF documents
  - Rules $\Rightarrow$ triggering and join rules
- Evaluation:
  - Determination of affected triggering rules
  - Iterative evaluation of join rules
    $\Rightarrow$ calculation of transitive closure

Normalization

- Path expressions are split up
- Search part contains all classes referenced by the rule
- Example:

```
search      CycleProvider c, ServerInformation s
register    c
where       c.serverHost contains 'uni-passau.de'
            and c.serverInformation = s
            and s.memory > 64
```

Rule Decomposition – Example

```
search      CycleProvider c, ServerInformation s
register    c
where       c.serverHost contains 'uni-passau.de'
            and c.serverInformation = s
            and s.memory > 64 and s.cpu > 500
```

RuleA **search** ServerInformation s **register** s **where** s.memory > 64

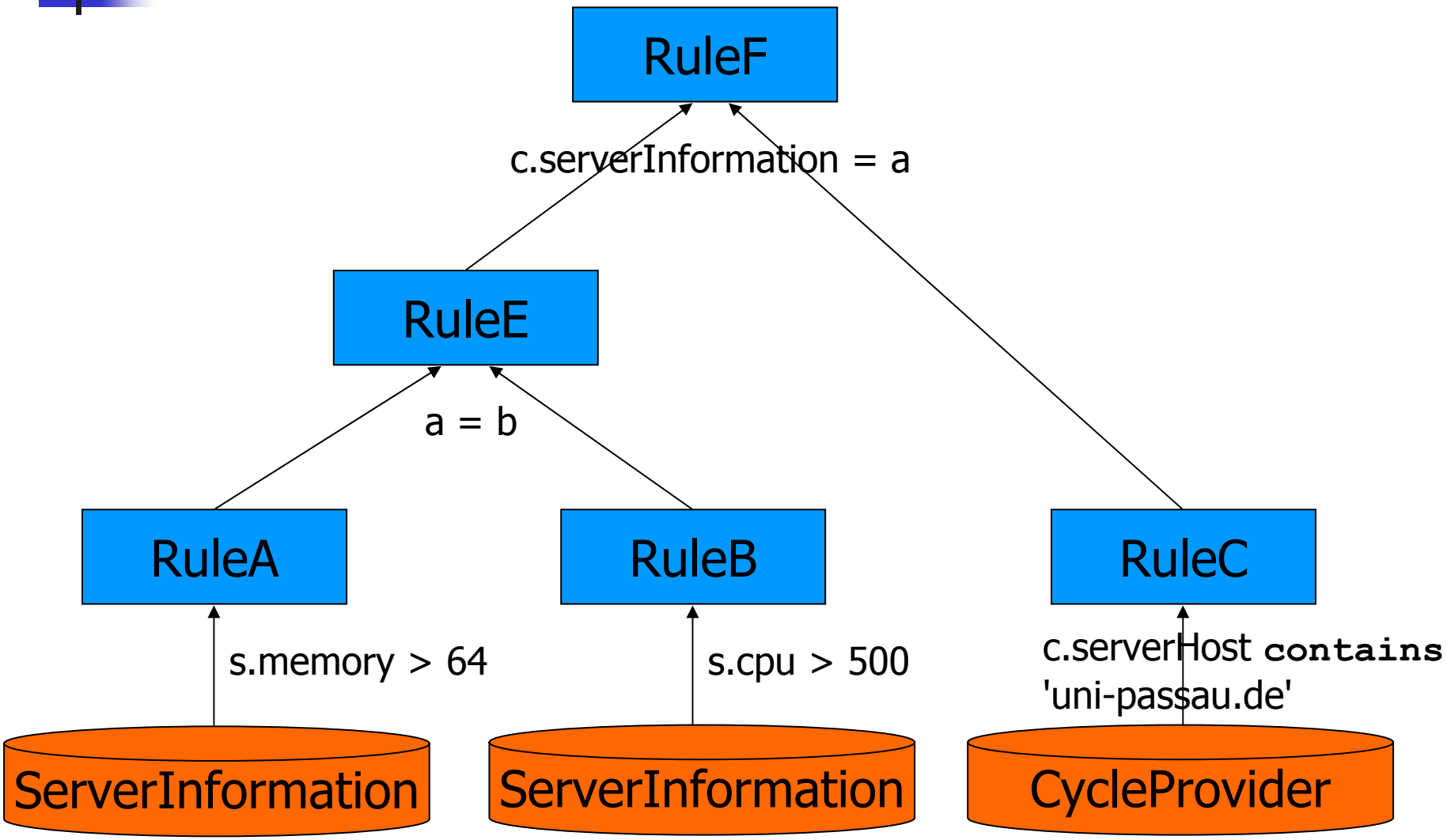
RuleB **search** ServerInformation s **register** s **where** s.cpu > 500

RuleC **search** CycleProvider c **register** c
where c.serverHost **contains** 'uni-passau.de'

RuleE **search** RuleA a, RuleB b **register** a **where** a = b

RuleF **search** RuleE a, RuleC c **register** c **where**
c.serverInformation = a

Dependency Tree



Details of the Algorithm

- Decomposition into atoms
  - RDF documents
  - Rules $\Rightarrow$ triggering and join rules
- Evaluation:
  - Determination of affected triggering rules
  - Iterative evaluation of join rules
    $\Rightarrow$ calculation of transitive closure

Filter Algorithm - Example

oid	class	property	value
doc.rdf#host1	CycleProvider	rdf#subject	doc.rdf#host1
doc.rdf#host1	CycleProvider	serverHost	pirates.uni-passau.de
doc.rdf#host1	CycleProvider	serverPort	5874
doc.rdf#host1	CycleProvider	serverInformation	doc.rdf#info1
doc.rdf#info1	ServerInformation	rdf#subject	doc.rdf#info1
doc.rdf#info1	ServerInformation	memory	92
doc.rdf#info1	ServerInformation	cpu	600

FilterData

rule_id	class	property	value
RuleA	ServerInformation	memory	64
RuleB	ServerInformation	cpu	500

FilterRulesGT

rule_id	class	property	value
RuleC	CycleProvider	serverHost	uni-passau.de

FilterRulesCON

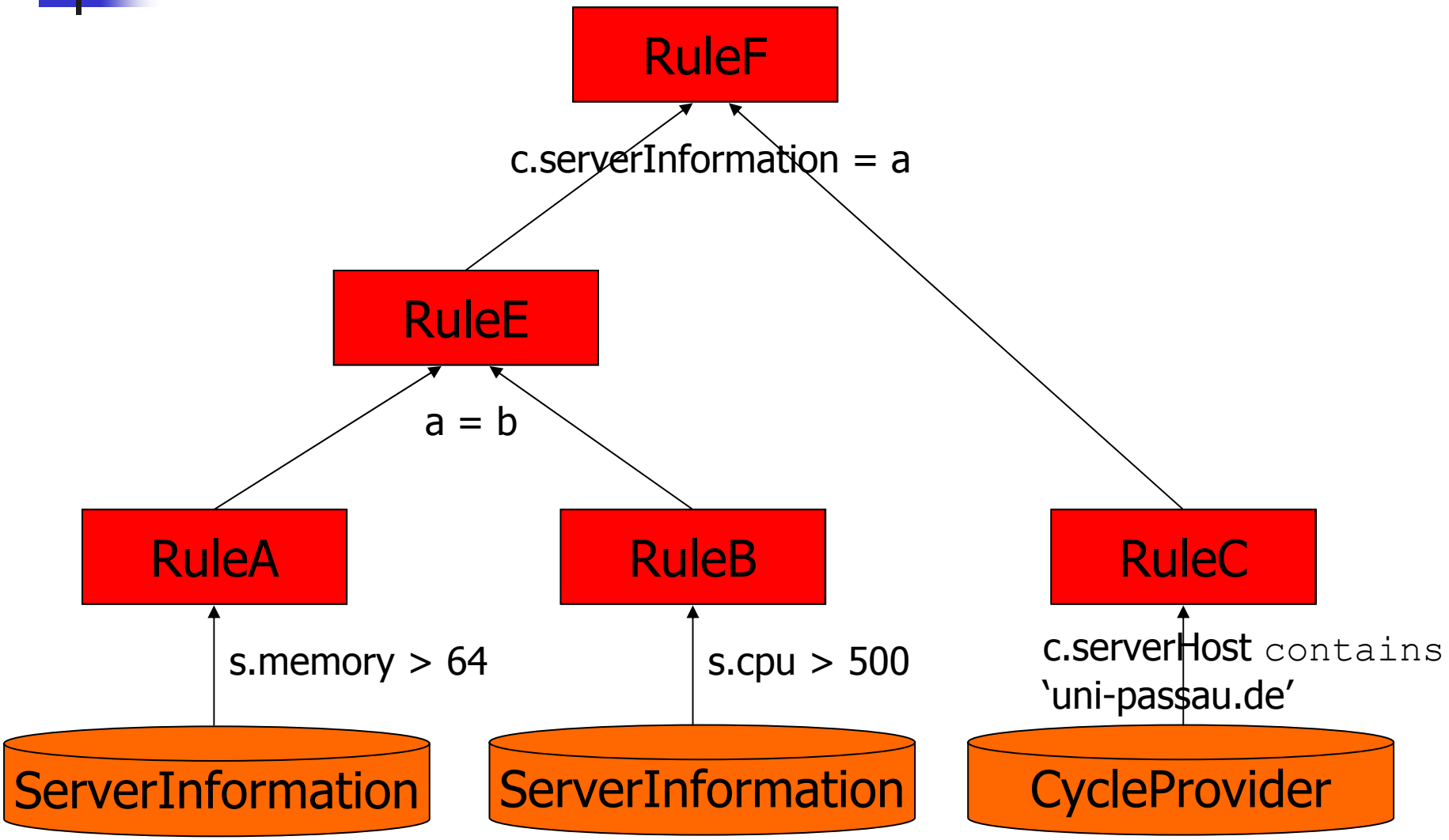
rid	rule_id
RuleA	doc.rdf#info1
RuleB	doc.rdf#info1
RuleC	doc.rdf#host1

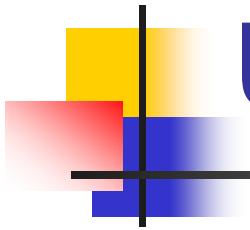
ResultObjects

Details of the Algorithm

- Decomposition into atoms
  - RDF documents
  - Rules $\Rightarrow$ triggering and join rules
- Evaluation:
  - Determination of affected triggering rules
  - Iterative evaluation of join rules
    $\Rightarrow$ calculation of transitive closure

Iterative Evaluation





Updates and Deletions

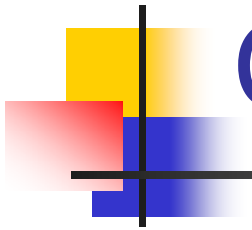
- Filter algorithm only works for new metadata
- Solution: execute algorithm 3 times

Related Work - 1

- **Metadata:**
  Equal Time For Data on the Internet with WebSemantics
  [Mihaila, Raschid, Tomasic; EDBT '98]
  MOCHA: A Self-Extensible Database Middleware System for Distributed Data Sources [Rodriguez-Martinez, Roussopoulos; SIGMOD '00]
  Universal Description, Discovery, and Integration (UDDI)
  [Ariba, Inc., IBM, Microsoft; http://www.uddi.org]
- **Publish/Subscribe:**
  Efficient Matching for Web-Based Publish/Subscribe Systems
  [Pereira, Fabret, Llirbat, Shasha; CoopIS '00]
  Matching Events in a Content-Based Subscription System
  [Aguilera, Strom, Sturman, Astley, Chandra; PODC '99]
  The SIFT Information Dissemination System
  [Yan, Garcia-Molina; TODS '99]
  Efficient Filtering of XML Documents for Selective Dissemination of Information [Altinel, Franklin; VLDB '00]

Related Work - 2

- **Continuous Queries:**
  NiagaraCQ: A Scalable Continuous Query System for Internet Databases
  [Chen, DeWitt, Tian, Wang; SIGMOD '00]
  Continual Queries for Internet Scale Event-Driven Information Delivery [Liu, Pu, Tang; IEEE TKDE '99]
- **Materialized Views and Semantic Caching:**
  Maintaining Views Incrementally
  [Gupta, Mumick, Subrahmanian; SIGMOD '93]
  Efficiently Updating Materialized Views
  [Blakeley, Larson, Tompa; SIGMOD '86]
  Semantic Data Caching and Replacement
  [Dar, Franklin, Jónsson, Srivastava, Tan; VLDB '96]



Conclusion

- The MDV System:
  MDPs, LMRs, and MDV Clients
- The Publish & Subscribe Algorithm:
  - Decomposition of documents and rules
  - Determination of affected triggering rules
  - Iterative evaluation of join rules