Active documents and Active XML

Serge Abiteboul

INRIA Saclay, Collège de France, ENS Cachan

Organization

Introduction

Modeling data intensive distributed systems

Query optimization in distributed systems

Monitoring in distributed systems

Task sequencing in distributed systems

Conclusion

Introduction

Context: Web data management

Scale: lots of servers, large volume of data

Servers are autonomous (heterogeneous also)

Data may be very dynamic, heavy update rates

Peers are possibly moving

The focus in this class

Relation	→	**Tree**
Centralized	→	**Distributed**
Precise data	→	Incomplete, probabilistic
Precise schemas	→	Ontologies

The lesson from the past

The success of the relational model with **2D-tables on local servers**

- A logic for defining tables
- An algebra for describing query plans over tables

We should do similarly for **trees in a distributed environment**

- A logic for defining distributed trees and data services
- An algebra for optimizing queries over trees/services

Roadmap

1. Modeling: the AXML model of active documents
   - **Views**: to capture intentional data
   - **Streams**: to capture exchanges of data and evolution

Key concept for Data management

Key concept for distribution and evolution

2. Optimization: an algebra for AXML
3. Monitoring: based on AXML documents
4. Task sequencing: A workflow based on AXML documents
   - In the spirit of business artifacts

Modeling data intensive distributed systems

Active XML

Active XML

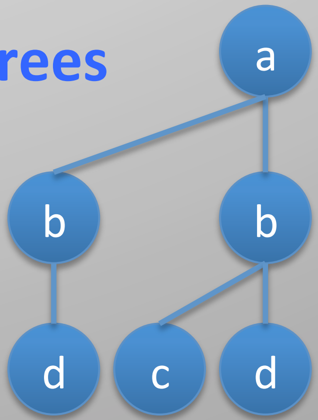
Based on Web standards:

XML + Web services + Xpath/Xquery

Idea: Exchange XML documents with embedded function calls

Active XML: Unordered, unranked, labeled , evolving trees

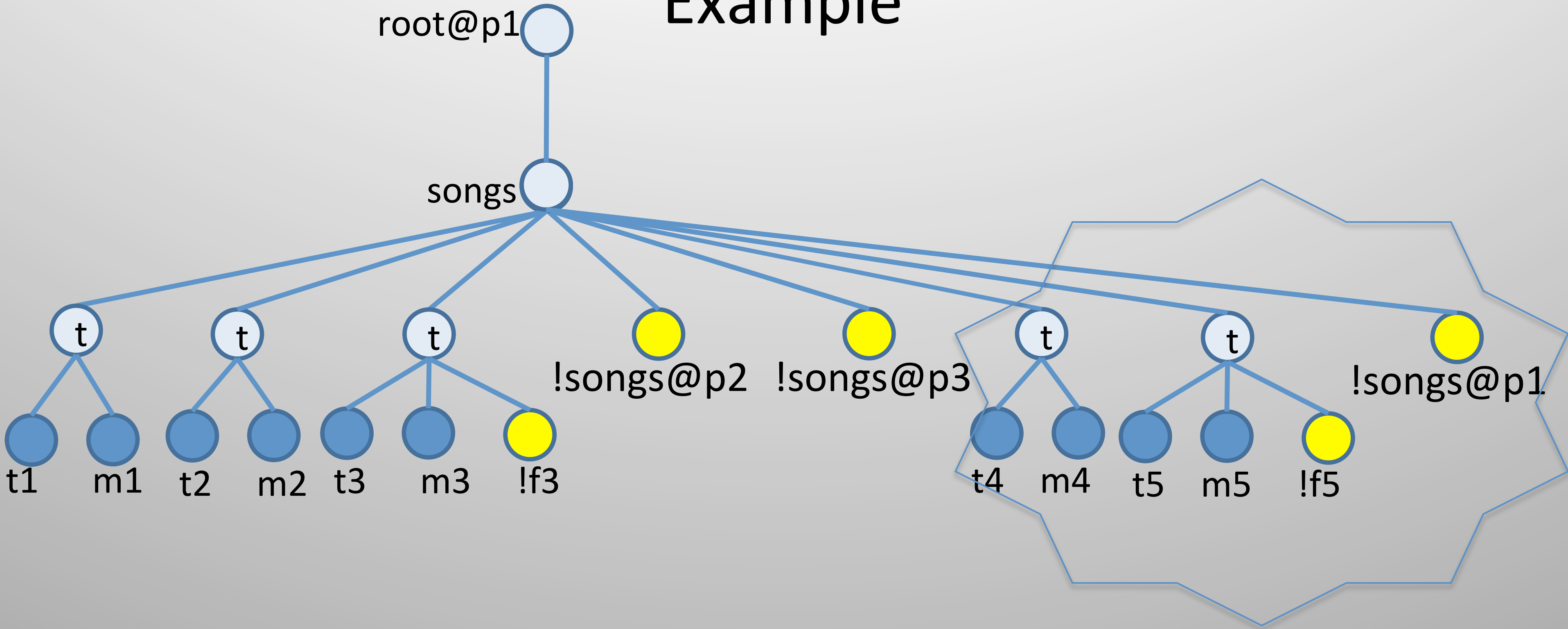
- Internal nodes are labeled by tags
- Leaves are labeled by tags, data, or **function symbols**
- Set semantics: No isomorphic sibling sub-trees



The functions are interpreted as calls to external services

- Embedding calls in data is an old idea in databases

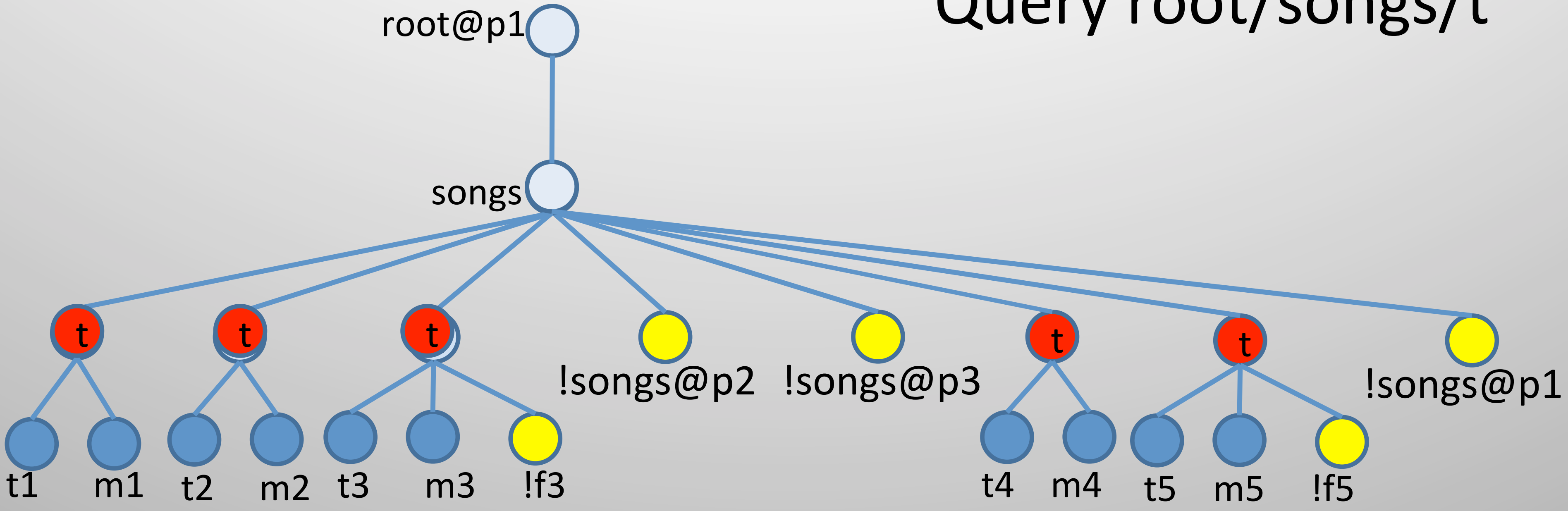
Example



Leads to evolving trees

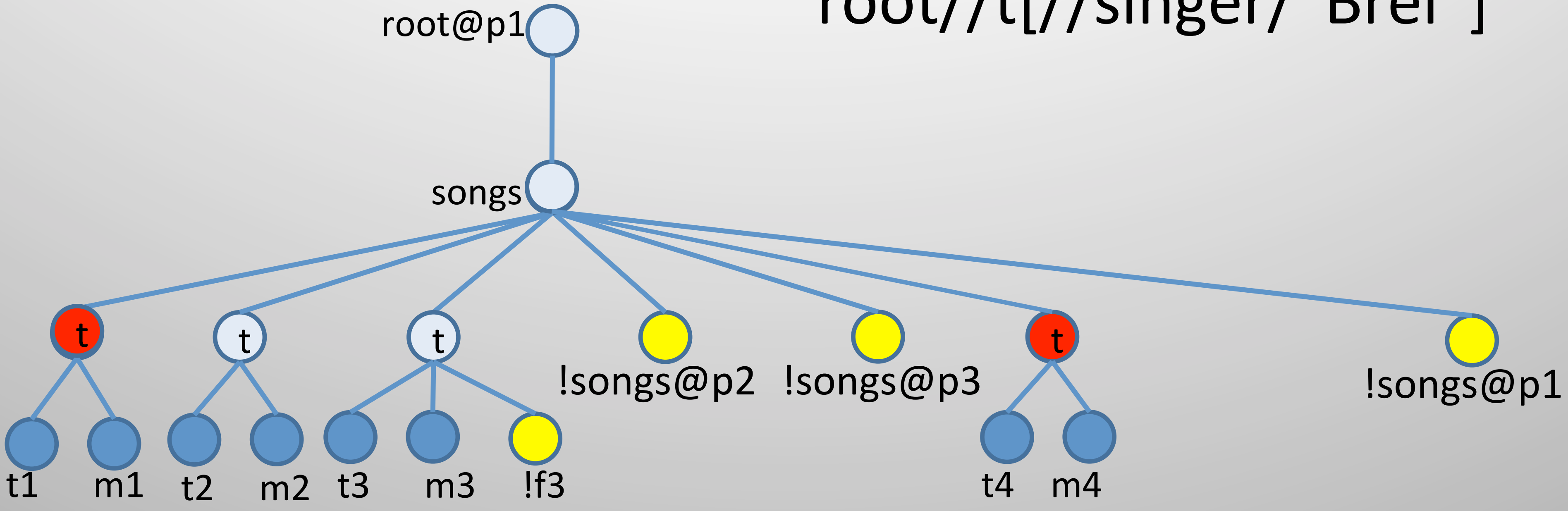
- Intentional data: get the data only when desired
- Dynamic data: If data sources change, the document changes
- Flexible data: adapt to the needs
- Function in push & pull mode

Query root/songs/t



Recursive calls

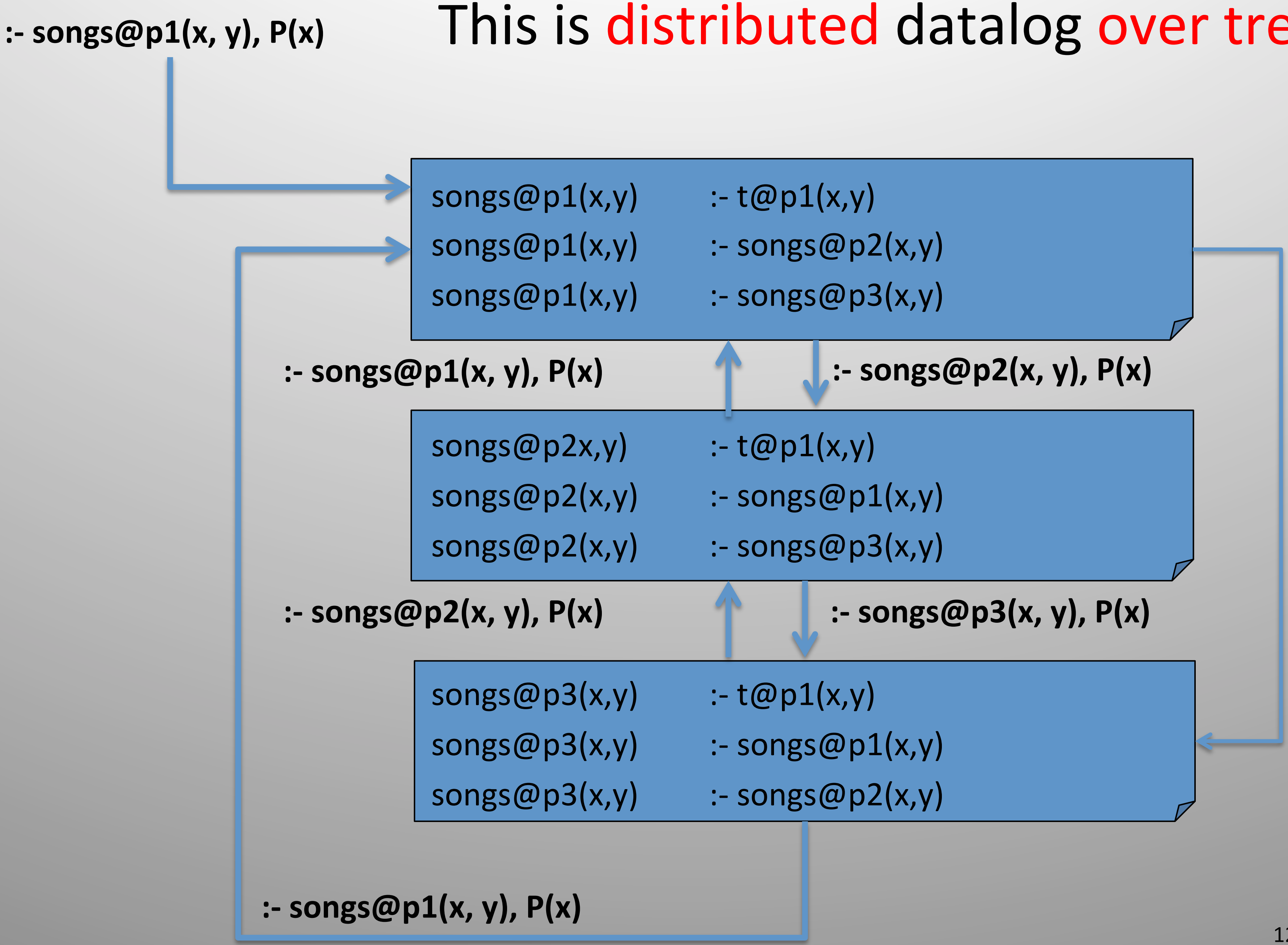
root//t[//singer/“Brel”]



Push queries to data sources

- !songs@p3: root//t[//singer/“Brel”]
- !songs@p2 root//t[//singer/“Brel”]
- !songs@p1: root//t[//singer/“Brel”]
- Distributed query/subquery (or Magic Set)

This is distributed datalog over trees



:- songs@p1(x, y), P(x)

songs@p1(x,y) :- t@p1(x,y)
songs@p1(x,y) :- songs@p2(x,y)
songs@p1(x,y) :- songs@p3(x,y)

:- songs@p1(x, y), P(x)

:- songs@p2(x, y), P(x)

songs@p2x,y) :- t@p1(x,y)
songs@p2(x,y) :- songs@p1(x,y)
songs@p2(x,y) :- songs@p3(x,y)

:- songs@p2(x, y), P(x)

:- songs@p3(x, y), P(x)

songs@p3(x,y) :- t@p1(x,y)
songs@p3(x,y) :- songs@p1(x,y)
songs@p3(x,y) :- songs@p2(x,y)

:- songs@p1(x, y), P(x)

Fun issues: The semantics of calls

When to activate the call?

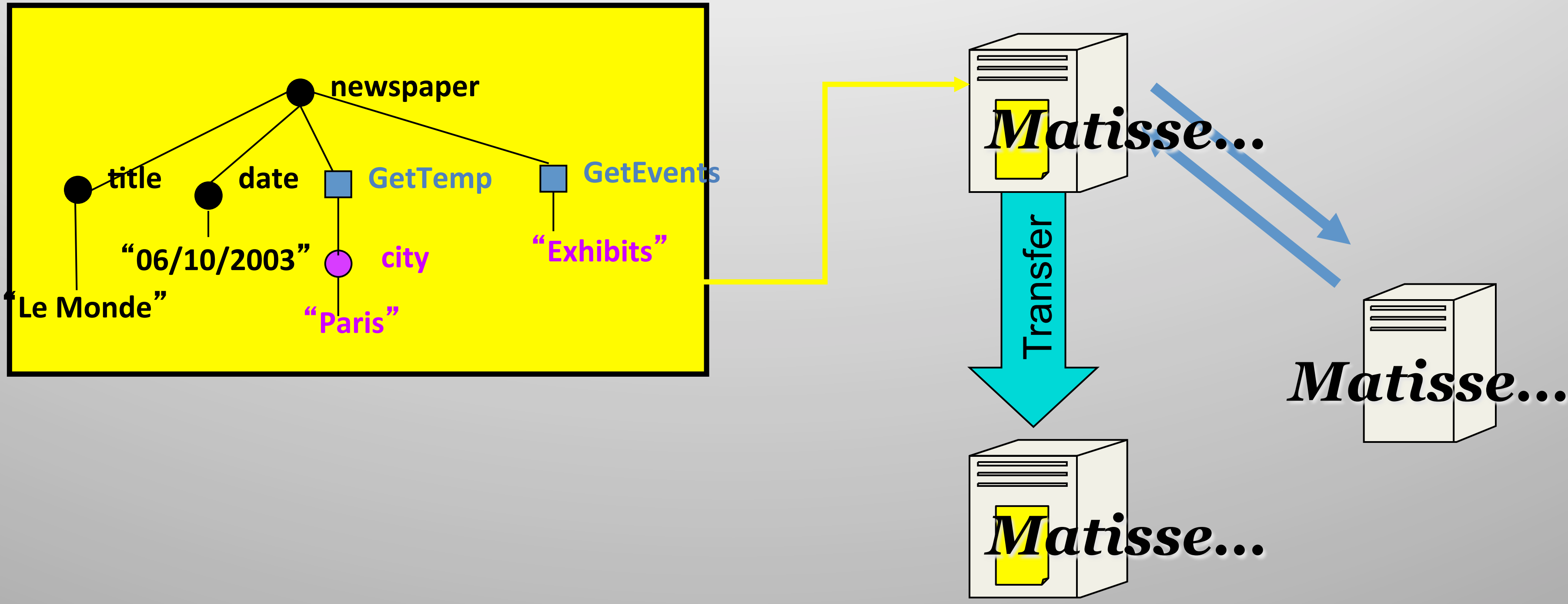
- Explicit pull mode: active databases
- Implicit pull mode: deductive databases
- Push mode: query subscription

What to do with its result?

How long is the returned data valid?

Sending an AXML documents: evaluate the service calls before sending or not?

Exchanging AXML data

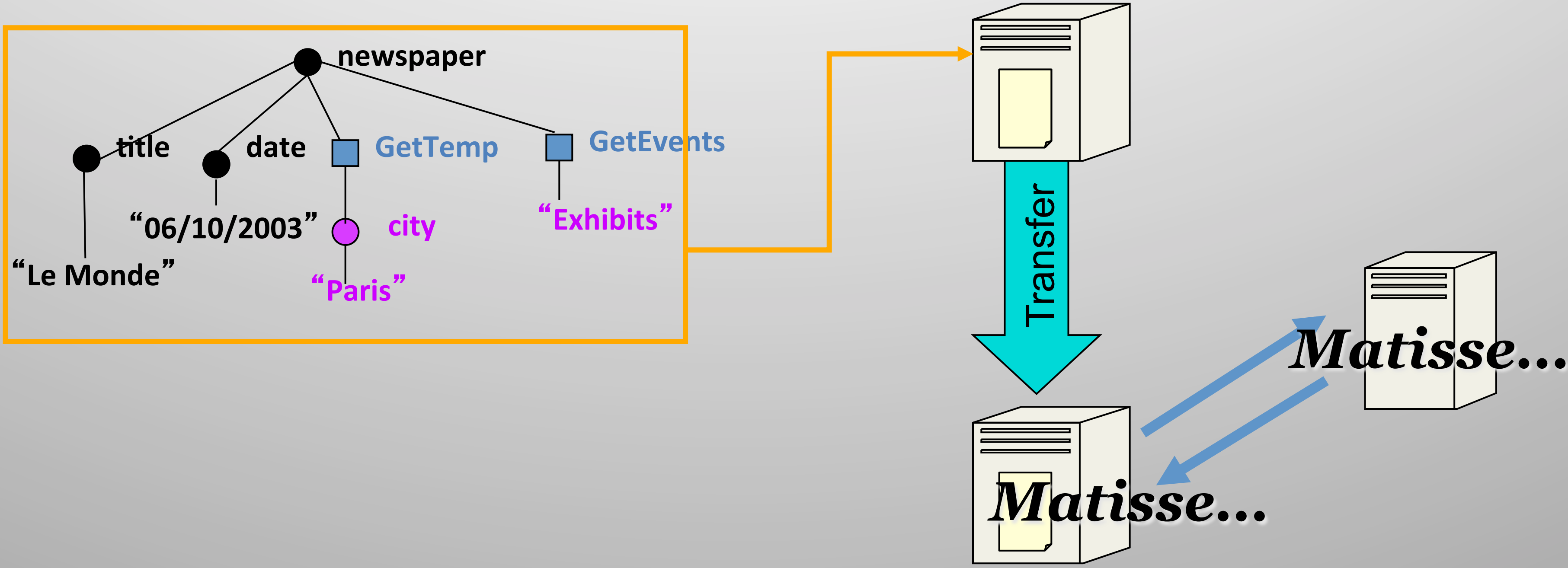


Web services exchange intentional documents

Materialization can be performed

- by the sender, before sending a document or
- by the receiver, after receiving it.

Exchanging AXML data



Web services exchange intentional documents

Materialization can be performed

- by the sender, before sending a document or
- by the receiver, after receiving it.

Some reasons for *not* materializing data before sending the document

Freshness

- The receiver will get up-to-date information when needed

Security

- Only the receiver has the credential to call the service
- One needs to record who is actually using the data

Performance

- To save on the bandwidth of the sender

To delegate work to someone else

How to specify it: casting based on types ☞ jewel section

Complex issues

Brings to a unique setting

distributed db
deductive db
active db
stream data
warehousing & mediation

This seems to us necessary for capturing all the facets of data management in distributed systems

This is unreasonable? Yes!

Query optimization in distributed systems

Active XML Algebra optimization

AXML system

A ***system*** = a set of peers

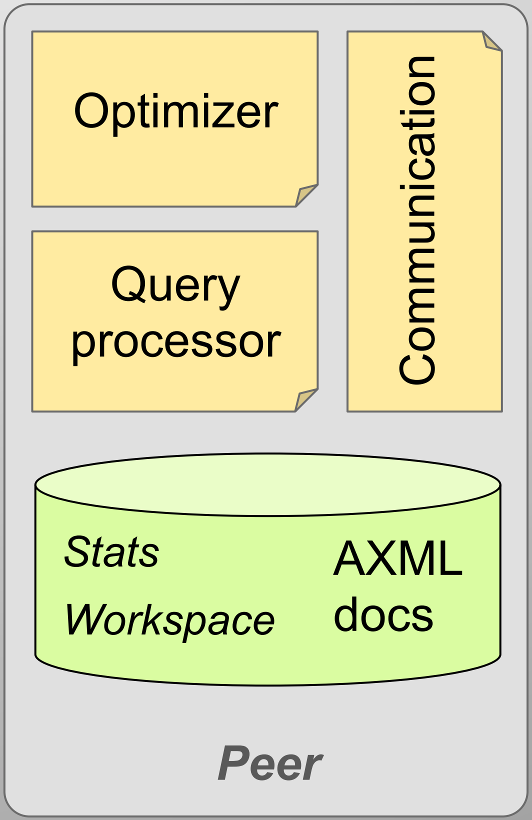
- Each peer provides storage and query processing
- Each peer hosts active documents

Extensional data

Intentional data (query calls in the document)

Problem:

Given a query q at some peer evaluate the answer to q with optimal response time

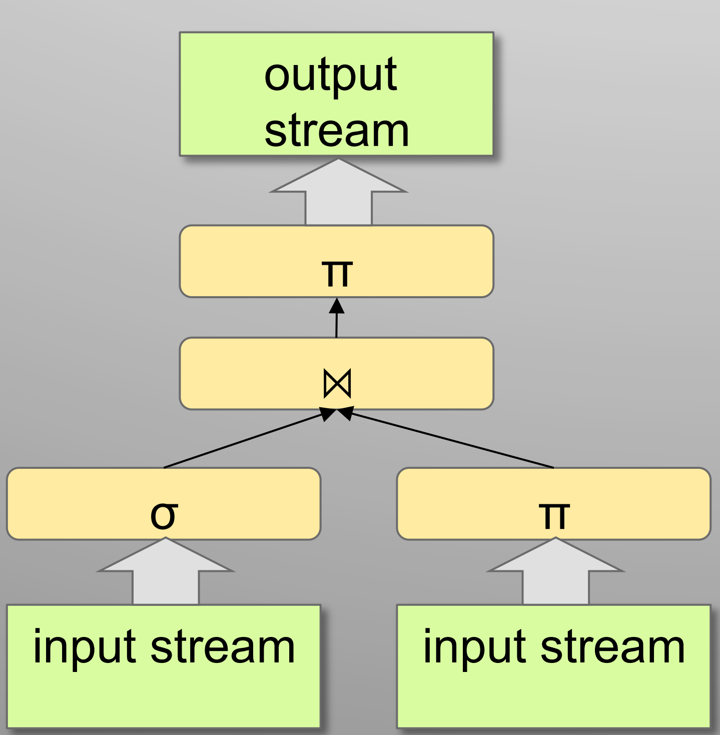


Local and global query processing

Local processing

☛ Input/output streams

Local query optimization

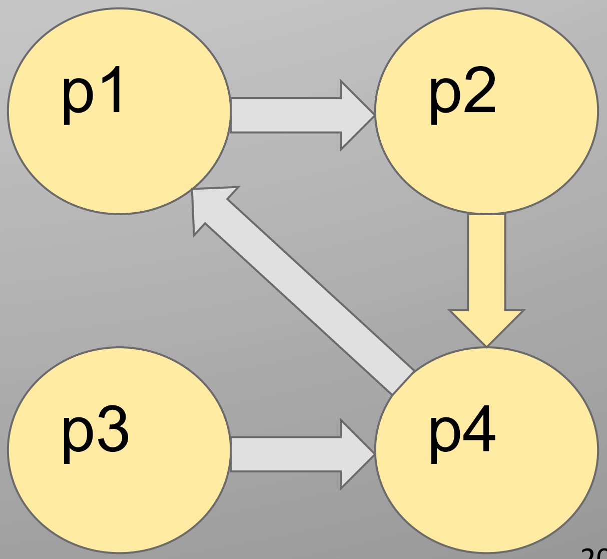


Global processing

☛ Streams for communications

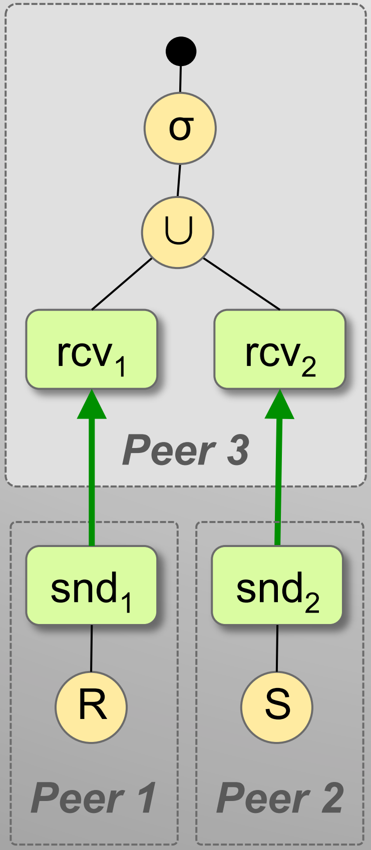
Global query optimization

☛ Delegate work to other peers

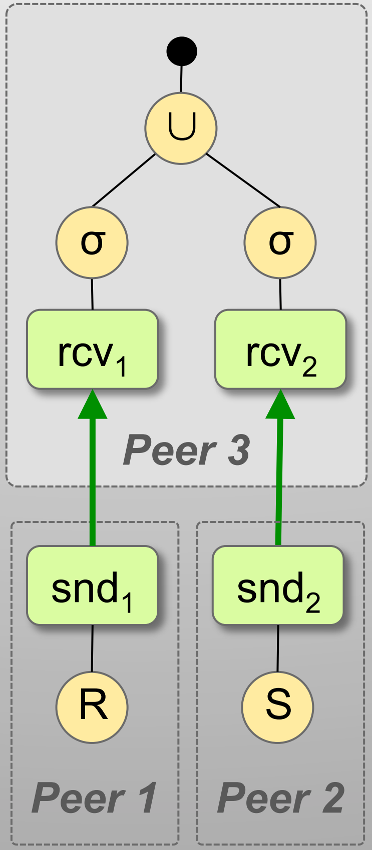


Example 1: Local and global optimization

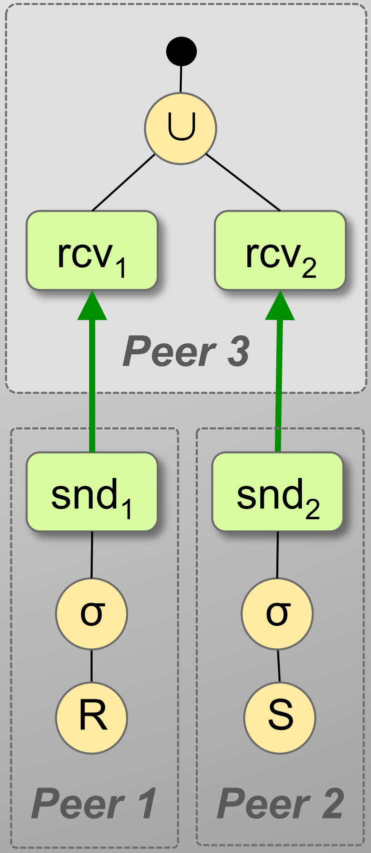
p3 asks for σ (R@p1 $\cup$ S@p2)



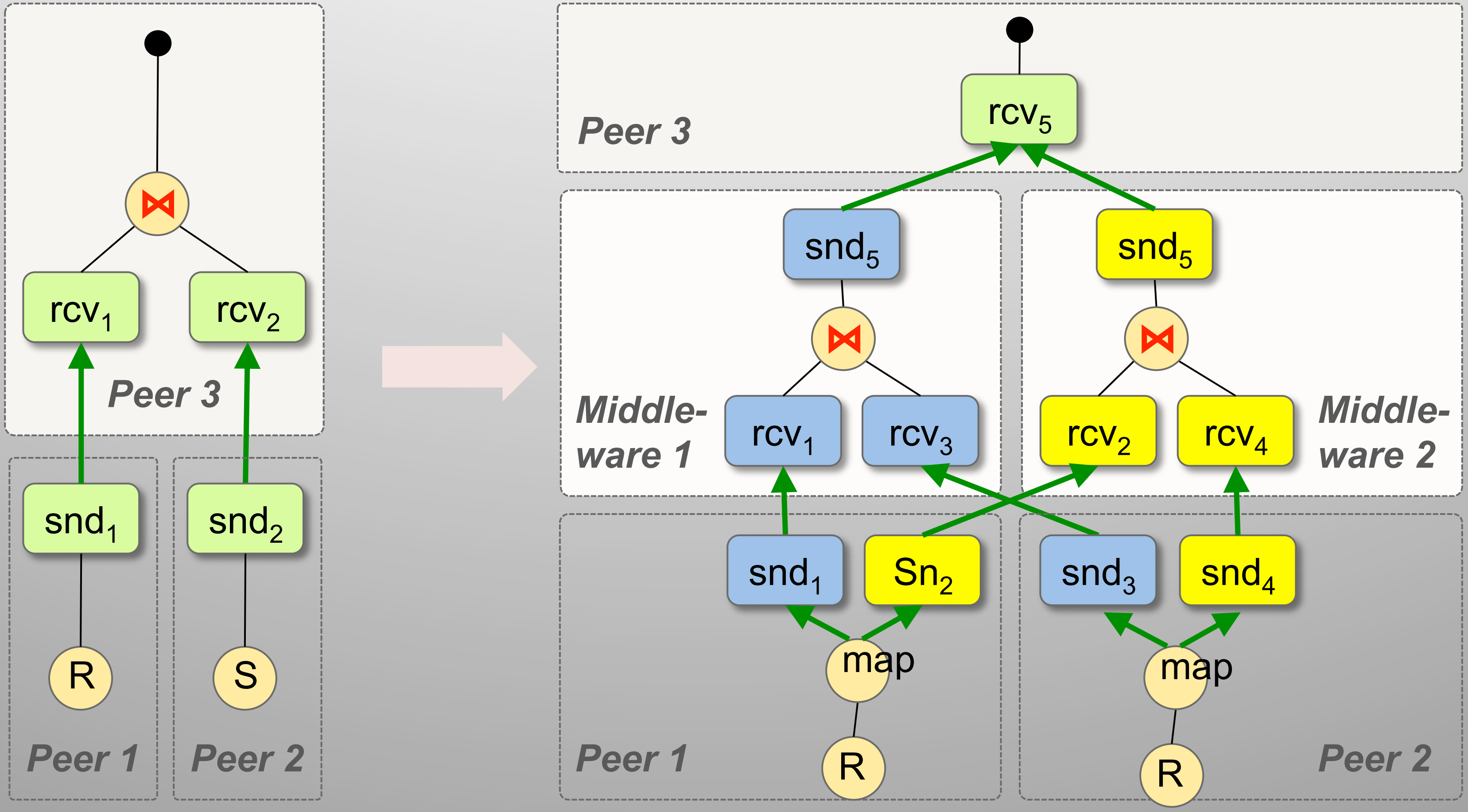
Local Rewriting: Selection & Union commute



Global Rewriting: Push selections to sources



Example 2: MapReduce



The Active XML algebra

Passive nodes

Annotated with labels

root a b

Query nodes

q

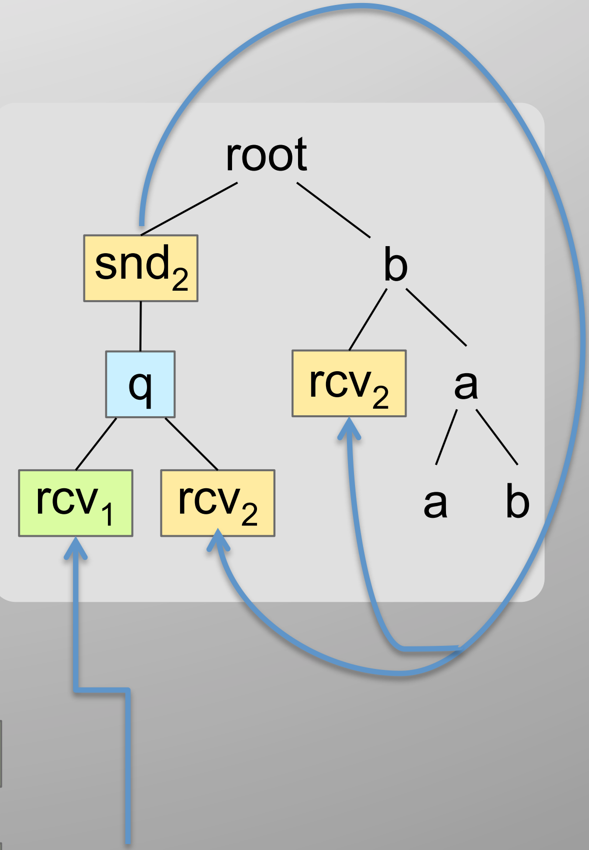
Annotated with queries

For instance Tree-Pattern-Queries

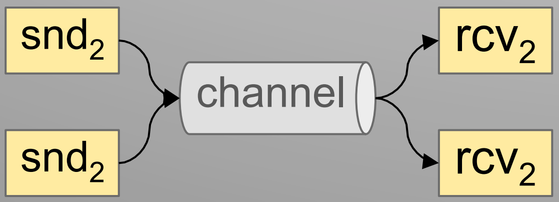
Send/Receive nodes

snd$_2$ rcv$_2$ rcv$_1$

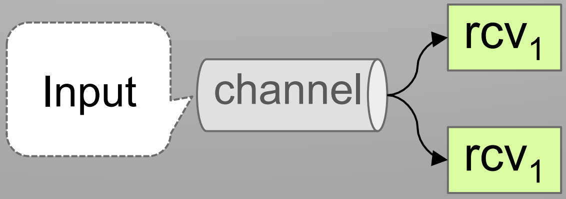
Annotated with channel ids



Internal channel



Input channel (no snd)



Evolution of a system

A system evolves by activating:

- a query node
- a send/receive node on an internal channel
- a receive node on an input channel

Equivalence problem for AXML systems

	No query	TPQ	TPQ with XPath joins	TPQ with joins	TPQ with constructor
No input	PTIME	PTIME	PTIME	Hard	Undecidable
Input	PTIME	Hard	Hard	?	Undecidable

Complexity increases with:

- richer query language
- the presence of input

Axiomatization of equivalence in absence of queries

Optimization

As usual

Use algebraic rewriting rules

Use simplistic estimators for query plans

Use heuristics to prune the search space

Examples of performance optimization techniques

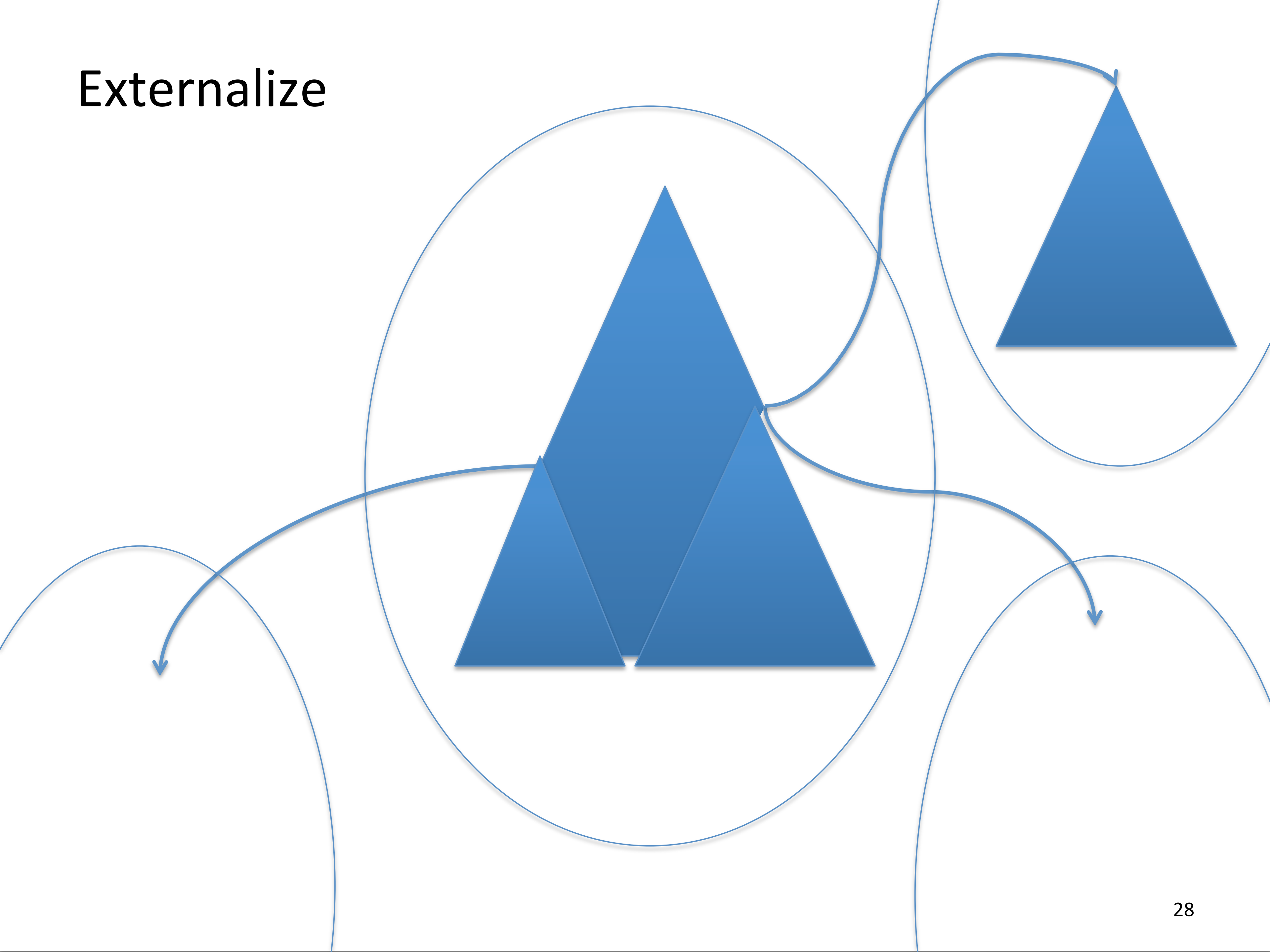
Externalize data in devices with limited capabilities

- Cell phone, tablets, home appliances…
- Limited storage space, computational power, network bandwidth

Replicate documents and services

- To allow for “local” computation
- To increase parallelism

Externalize



Monitoring in distributed systems

The Axlog system

Monitoring distributed systems

Distributed applications are often very dynamic

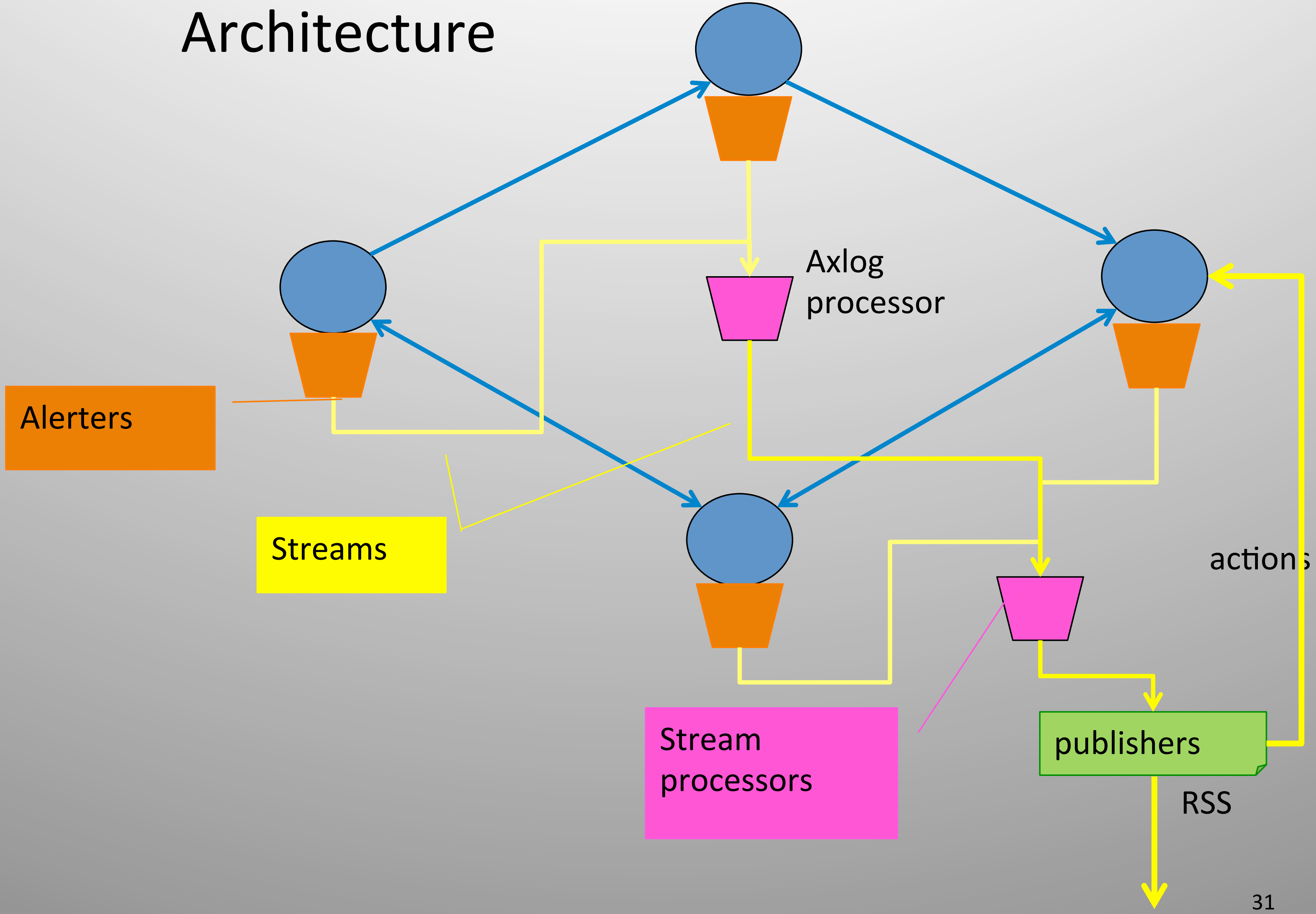
- Content change rapidly
- Intense communications
- Peers sometimes come and leave

Complex and hard to control such systems

- Many peers
- Peers are distributed & autonomous
- Peers are sometimes unreliable and selfish

Goal: **monitor** such systems

Architecture



Axlog principle = active document & query

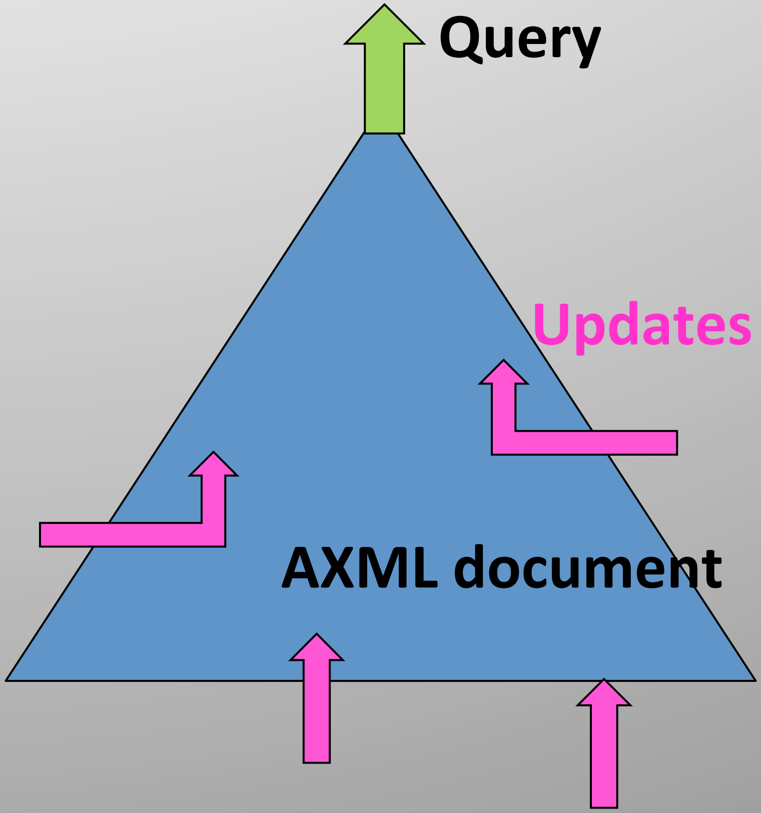
Incoming streams of updates

The outgoing stream is defined by a query Q (e.g. TPQ)

Each time an incoming message arrives, it modifies the document so possibly the query result

The output stream specifies how the view is modified

Incremental view maintenance



Axlog engine

Datalog is used to evaluate queries with benefit from

– Incremental view maintenance in datalog	Δ technique
– Query optimization in datalog	MagicSet
– Constraint query languages	CQL

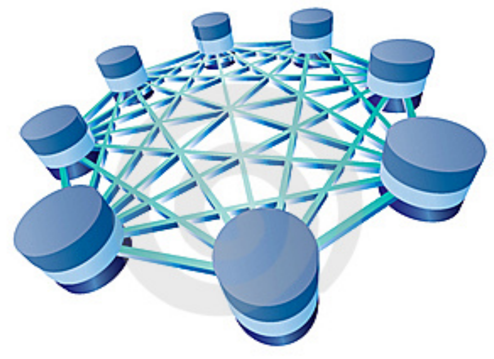
Specific techniques

- Push queries to the sources to avoid loading irrelevant data
- Use of FSA on XML inputs: YFilter

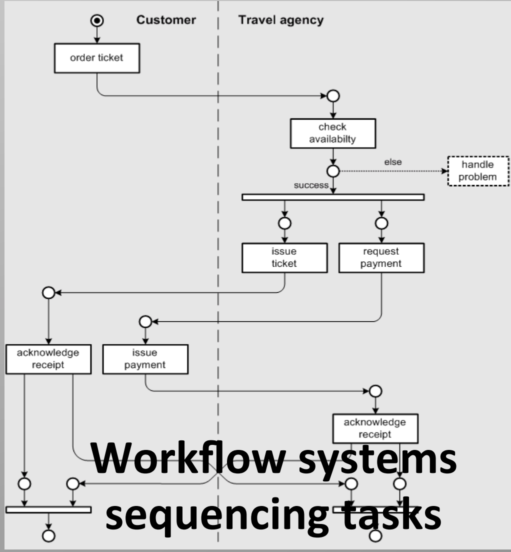
Task sequencing in distributed systems

Task sequencing and verification

- Task sequencing is a major difficulty for distributed systems
  - Difficulty to integrate **workflow** and **database** systems
- Verification of temporal properties is hard
  - Typically verification is harder than evaluation
    - Evaluating an FO query is ptime data complexity
    - Verifying that Q ⊆ Q' is undecidable
  - Verification will be the topic of the seminar by **Victor Vianu**

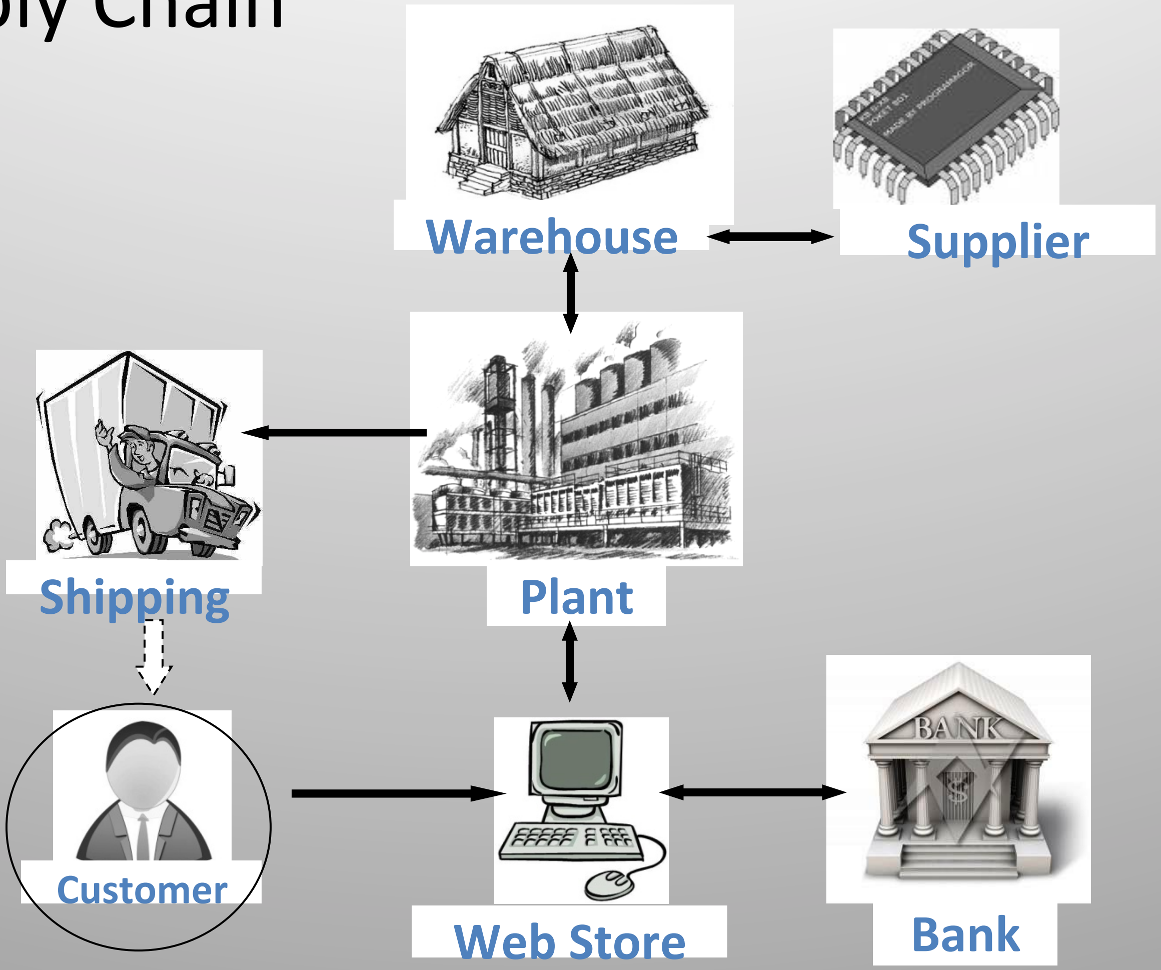


DBMSs exchanging data



Workflow systems sequencing tasks

Example:
Dell Supply Chain



AXML as *business artifacts*

Concept introduced by IBM

[Nigam & Caswell 03, Hull & Su 07]

Data-centric workflows

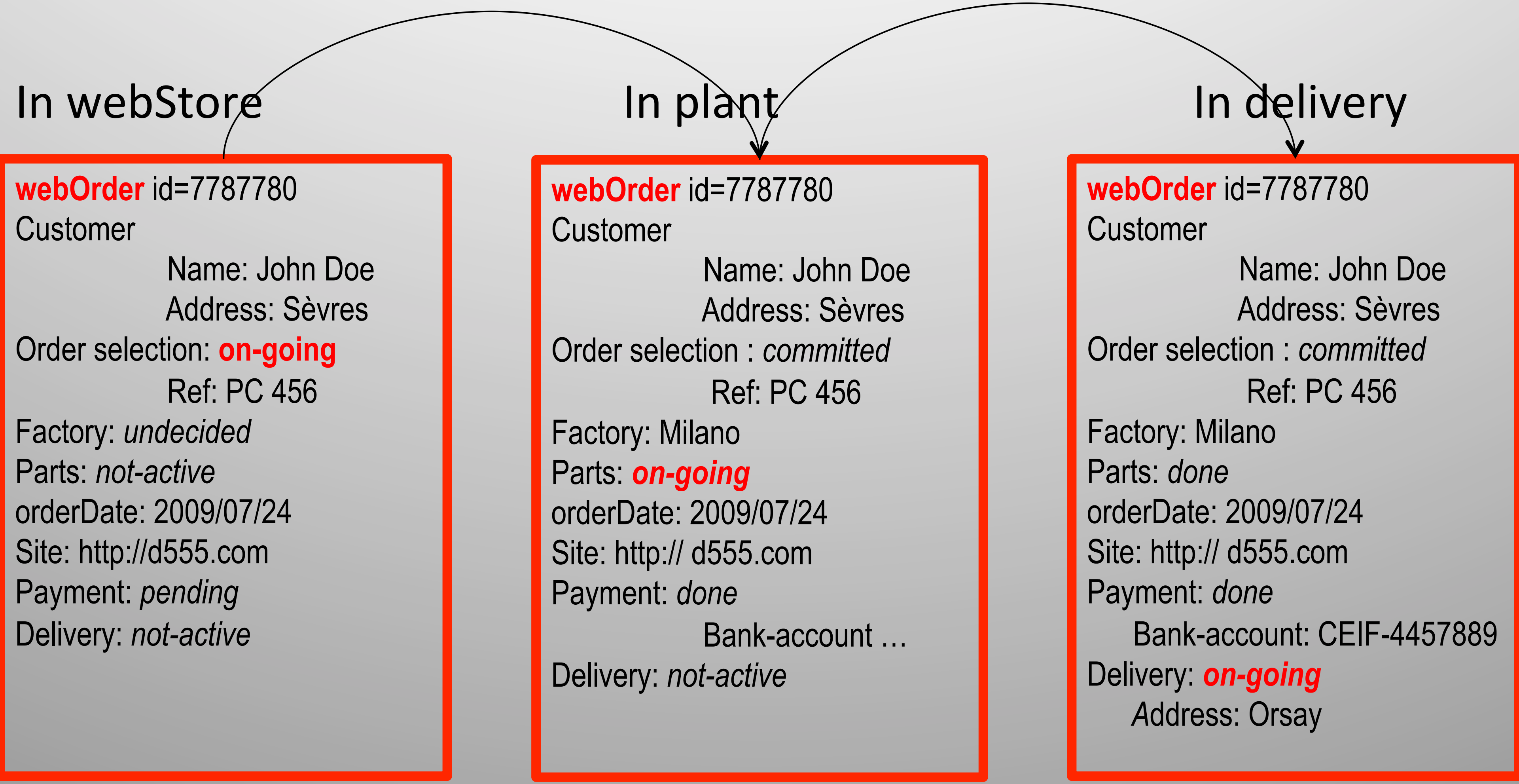
- A process is described by a document (possibly moving in the enterprise)
- The behavior of an artifact is specified by some constraints on its evolution

Vs. state-transition-based workflows

- Based on some form of state transition diagrams (BPEL, Petri,…)
- Mostly ignore data

webOrder id=7787780
Customer
    Name: John Doe
    Address: Sèvres
Product: *committed*
    Ref: PC 456
Factory: Milano
Parts: *waiting*
orderDate: 2009/07/24
Site: http:// d555.com
Payment: *done*
    Bank-account …
Delivery: *not-active*

Axml Artifacts move between peers



In webStore

webOrder id=7787780

Customer

Name: John Doe

Address: Sèvres

Order selection: **on-going**

Ref: PC 456

Factory: *undecided*

Parts: *not-active*

orderDate: 2009/07/24

Site: http://d555.com

Payment: *pending*

Delivery: *not-active*

In plant

webOrder id=7787780

Customer

Name: John Doe

Address: Sèvres

Order selection : *committed*

Ref: PC 456

Factory: Milano

Parts: ***on-going***

orderDate: 2009/07/24

Site: http:// d555.com

Payment: *done*

Bank-account …

Delivery: *not-active*

In delivery

webOrder id=7787780

Customer

Name: John Doe

Address: Sèvres

Order selection : *committed*

Ref: PC 456

Factory: Milano

Parts: *done*

orderDate: 2009/07/24

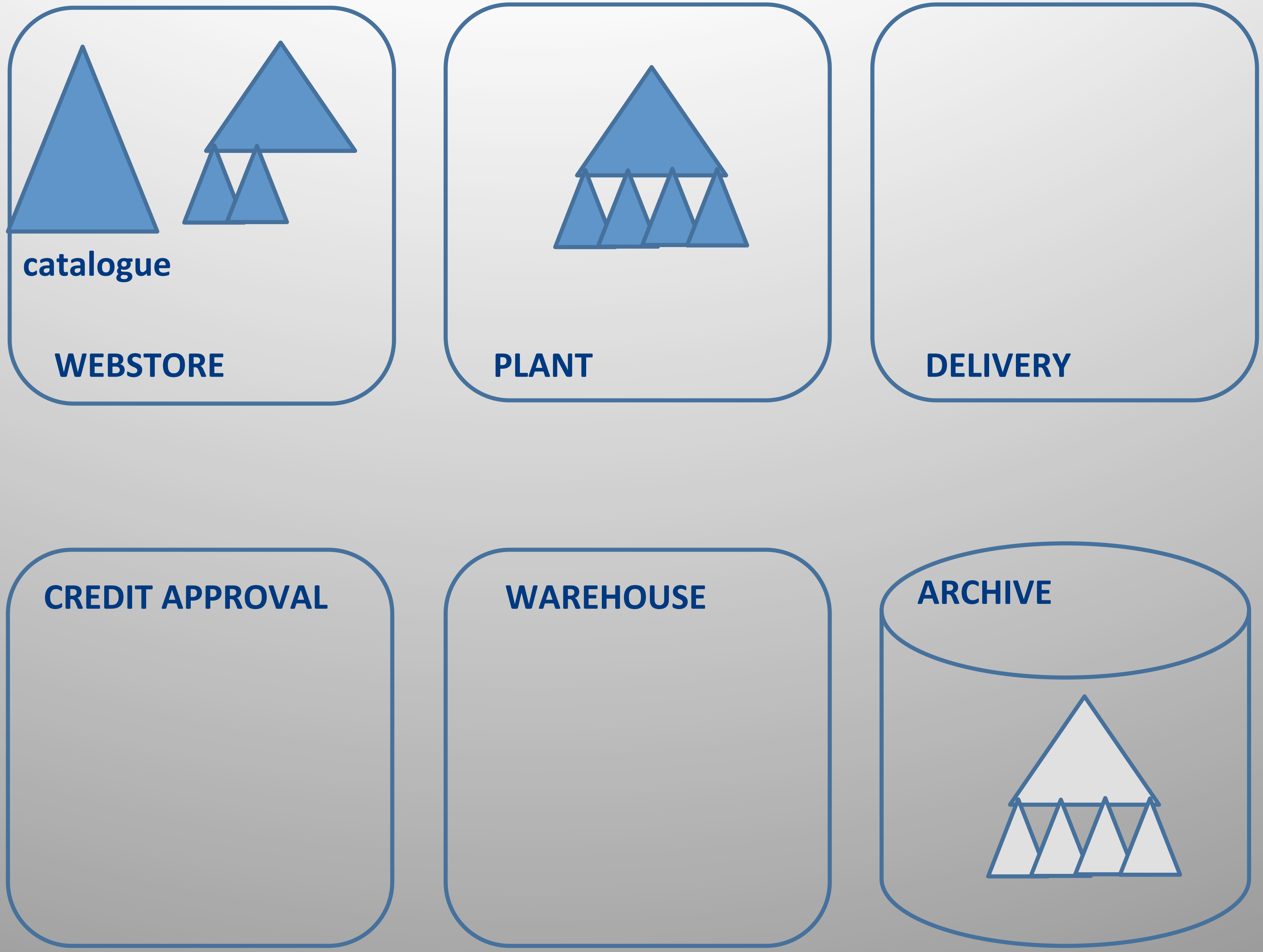
Site: http:// d555.com

Payment: *done*

Bank-account: CEIF-4457889

Delivery: ***on-going***

Address: Orsay



catalogue

WEBSTORE

PLANT

DELIVERY

CREDIT APPROVAL

WAREHOUSE

ARCHIVE

Sequencing of operations

Different ways of expressing sequencing of tasks

- Guards: preconditions for function calls
- Transition-based diagrams
- Formulas in temporal logic

Study how they can simulate each other using some "scratch paper"

A jewel of active documents

Casting document to a target type

The casting problem

Given

- An active document I
- The signature of the functions
- And a target type T

Which functions to call to be sure to reach T?

2-player game

- Juliet chooses which function to call
- Romeo chooses a value within the domain of the function

Juliet wins if she can reach a document in T

An abstraction: active context-free games

On words instead of trees

- Game (**Σ**,R,T)
  - **Σ** is a finite alphabet
  - R set of CF rules
  - T is a regular target language
- w is the start word

Output: true if Juliet has a winning strategy

Alternation of

∃ states (Juliet pick next function to call) and

∀ states (the adversary Romeo picks the answer)

Examples

- Winning
- Losing

a→abc*; b→(ba)*b; c→ab
Target abab(ab)*

- Start word aba
- Strategy
  - Call the second a
  - Call all the c’s
  - Obtain a word in Target

- Start word ab
- No strategy
  - Initially #(a) – #(b) = 0
  - If I call a or b, #(a) – #(b) < 0

Fun rewriting game

The problem is undecidable in general

Interesting decidable subcases

- MuschollSchwentickSegoufin
- Juliet has to traverse the string from left to right
- No recursion among function calls
- Function call are “linear”

Also in practice, very efficient casting based on unambiguous grammars

Conclusion

Some works around Axml

The Axml system – open-source (on server, on smartphone)

The useful: Replication and query optimization

How to evaluate a query efficiently by taking advantage of replication

The useful: Lazy query evaluation

How to evaluate a query without calling all embedded services

The fun: Casting problem

Which functions to call to “match” a target type

Active context-free games

The exotic

- Diagnosis of communication systems based on datalog optimization
- Access control
- Distributed design
- Probabilistic generation of documents

We will come back to distribution

Lesson 6: datalog - recursion is essential

Lesson 7: distributed data management in general

Lesson 8: distributed knowledge bases

Acknowledgements

With many colleagues, in particular:

- Tova Milo (Tel Aviv)
- Luc Segoufin (INRIA)
- Georg Gottlob (Oxford)
- Angela Bonifati (Lille)
- Balder ten Catte (UCSC)
- Victor Vianu (UCSD)
- Ioana Manolescu (INRIA)
- Alkis Polyzotis (UCSC)
- Marie-Christine Rousset (Grenoble)
- Yannis Katsis (UCSD)

And PhD students

- Omar Benjelloun (Google)
- Pierre Bourhis (INRIA)
- Marco Manna (Calabria)
- Zoe Abrams (Google)
- Bogdan Cautis (Telecom)
- Bogdan Marinoiu (SAP)
- Alban Galland (INRIA)
- Nicoleta Preda (Versailles)
- Emmanuel Taropa (Google)
- Spyros Zoupanos (Max-Planck-Institut)

And others

informatiques mathématiques

COLLÈGE DE FRANCE 1530

CACHAN

Static Analysis and Verification

Victor Vianu, U.C. San Diego

PhD from USC 1983

Sabbaticals INRIA, ENS Cachan, Ulm, Telecom

Interests: database theory, computational logic, Web data

Co-author of *Foundations of databases*

- Aka the Alice book

Vianu has served as

- General Chair of SIGMOD, PODS,
- Program Chair of PODS, ICDT

Editor-in-Chief of the J. ACM

ACM Fellow