

# Scalable View Expansion in a Peer Mediator System

Presented by: Timour Katchaounov  
UDBL, Uppsala University, Sweden

Authors: Timour Katchaounov, Vanja Josifovski, Tore Risch

# Outline

---

- **Introduction**

- Motivation
- Peer mediator architecture
- Summary of contributions
- Related work

- View expansion in peer mediators
- Experimental evaluation
- Conclusions

# Motivation: Internet-scale data integration

---

## Characteristics:

- Many autonomous data sources
  - distributed
  - preexisting
  - unaware of each other
- Data integration is *'manual'*
- Bottom-up design
- Examples:
  - scientific applications
  - distributed engineering

## Problems:

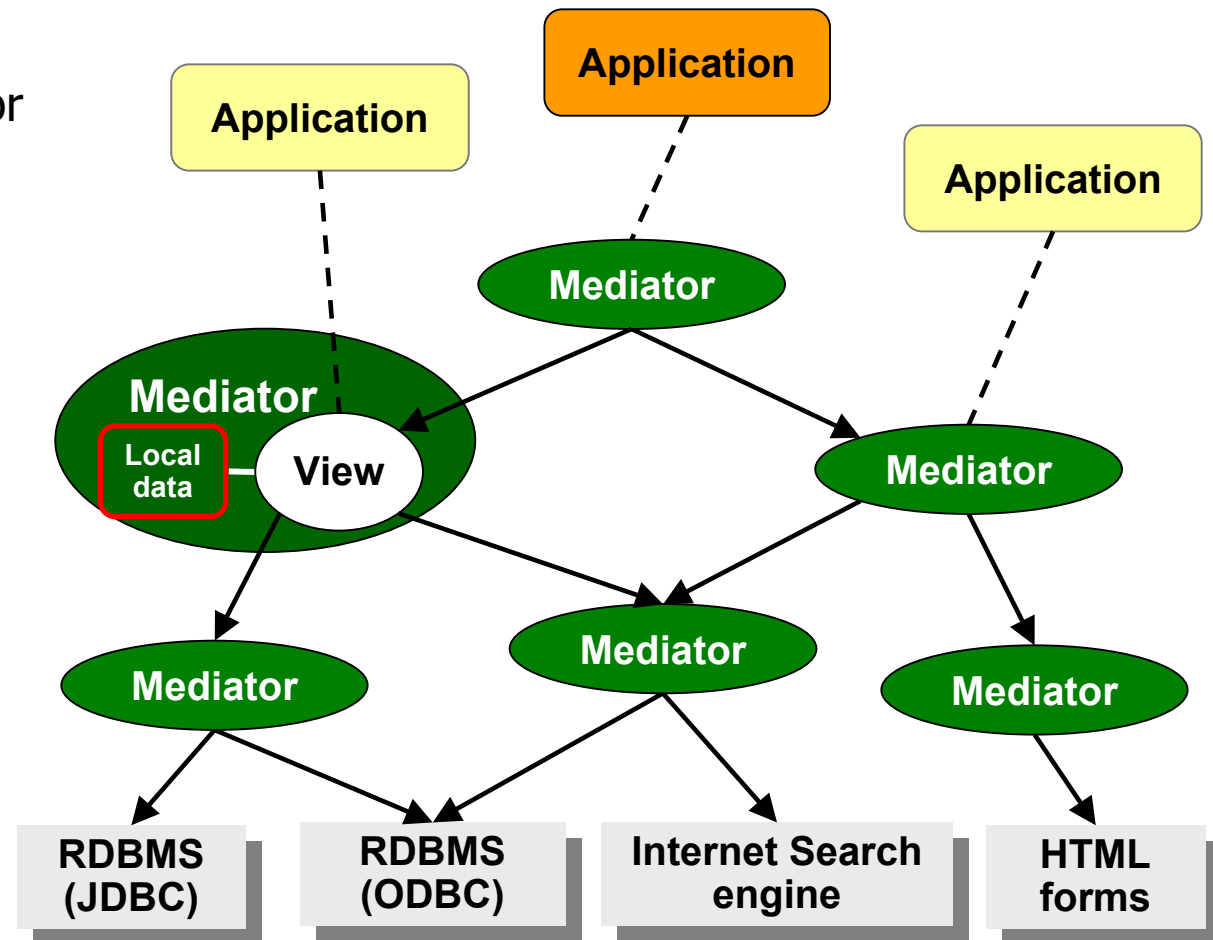
- Preserve autonomy
- Scalability of integration => reuse human effort
- System components evolve independently
- High overall scalable performance
- Others: security, query interfaces, etc.

# Peer mediator architecture

- Mediators are ORDBMS:
  - Autonomous
  - No central schema/coordinator
  - Cooperating
    - compile, cost, execute
  - Extensible

## => Peer architecture

- Logical composition of views
- *Multidatabase* views
- Logical view integration graphs:
  - nodes: mediator views
  - arcs: “defined in terms of”



# Summary of contributions

---

Mediator composition = view composition

Problem: ***free composition*** of mediators (views) = ***redundancy***

***How to process queries against composed multidatabase views?***

- Traditional approaches:
  - views as black boxes (no view expansion)
  - expand all views
- Our approach:  
generalize 1,2 => ***expand multidatabase views selectively***
- Investigate query compilation/execution tradeoffs
- Implementation - AMOS II mediator system

# Related work

---

- Mediators
  - centralized or fixed n-tier
  - full or no view expansion
- Distributed databases
  - tightly coupled
  - full view expansion
  - restricted view expansion:  
System R\*
- Web Services
  - focus on workflow composition
- P2P file sharing systems
  - large scale  
but
  - simple keyword searching
  - no view support

No performance studies of  
peer mediator/database architectures

# Outline

---

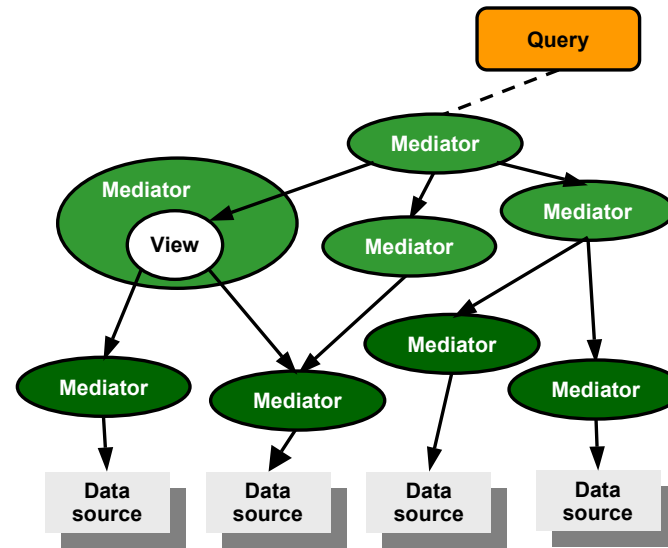
- Introduction
- **View expansion in peer mediators**
  - no view expansion (views are black boxes)
  - full view expansion (views are transparent)
  - selective view expansion
- Experimental evaluation
- Summary and conclusions

# View expansion in peer mediators

view = multidatabase view

Existing approaches:

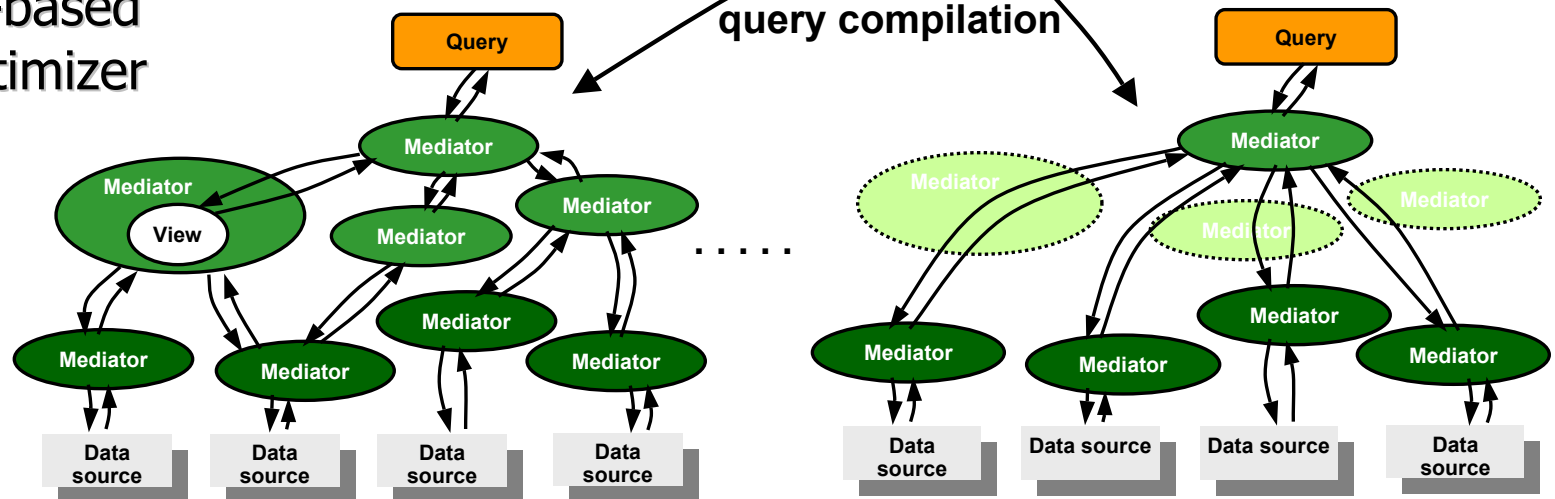
- Views are black boxes:
  - CORBA, Web services, Mediators
- Views are transparent:
  - DDBMS, Mediators
- Typically cost-based 'System-R' optimizer



one

**logical  
view  
integration  
graph**

query compilation



many

**distributed  
dataflow  
graphs**

# View expansion in peer mediators

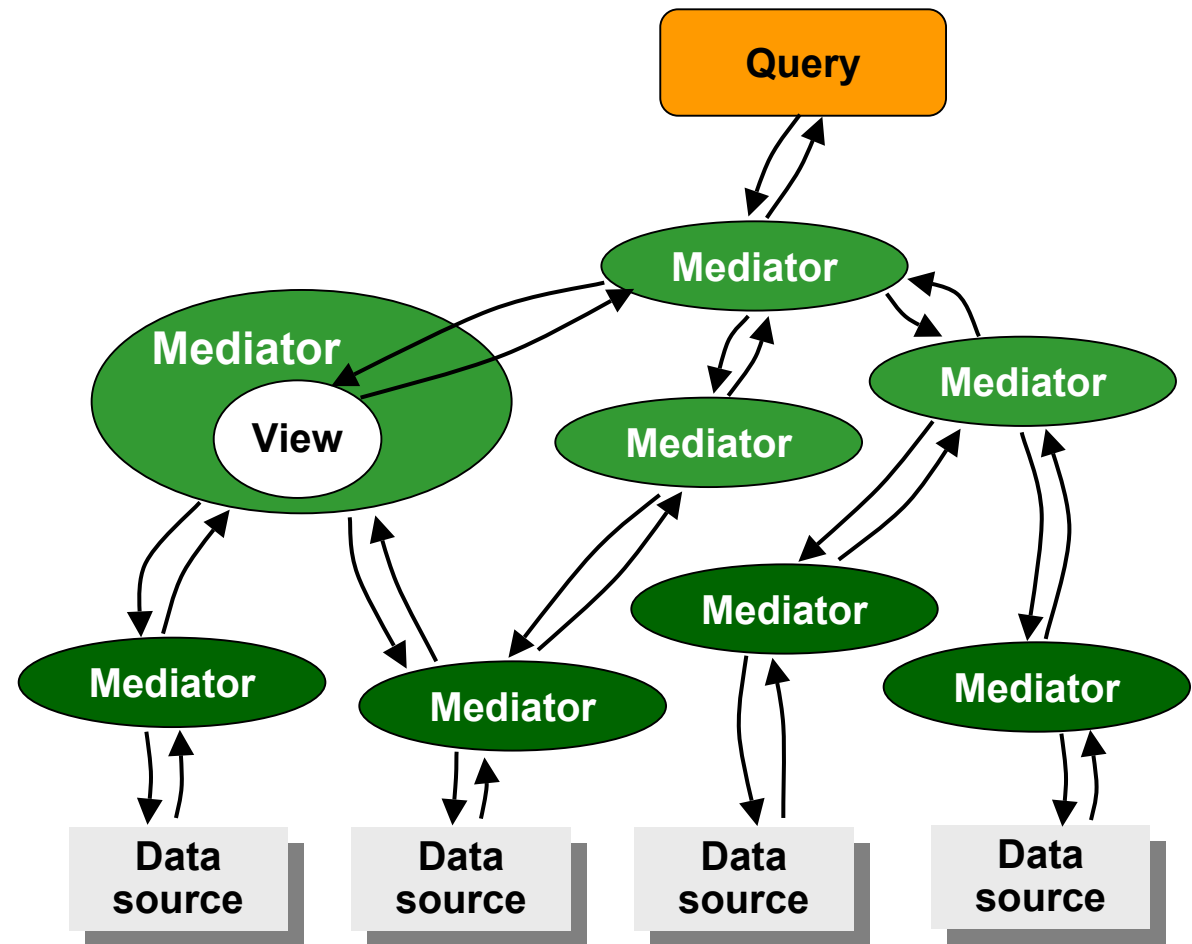
<1> Views are black boxes

Query compilation:

- Decompose queries
- 'Push' sub-queries to mediators/sources
- Optimize sub-queries with local views
- Recursive compilation

Pros and cons:

- + respect view autonomy
- + faster compilation (?)
- inefficient plans  
(redundancy, no indexes)



**Distributed dataflow graph**

# View expansion in peer mediators

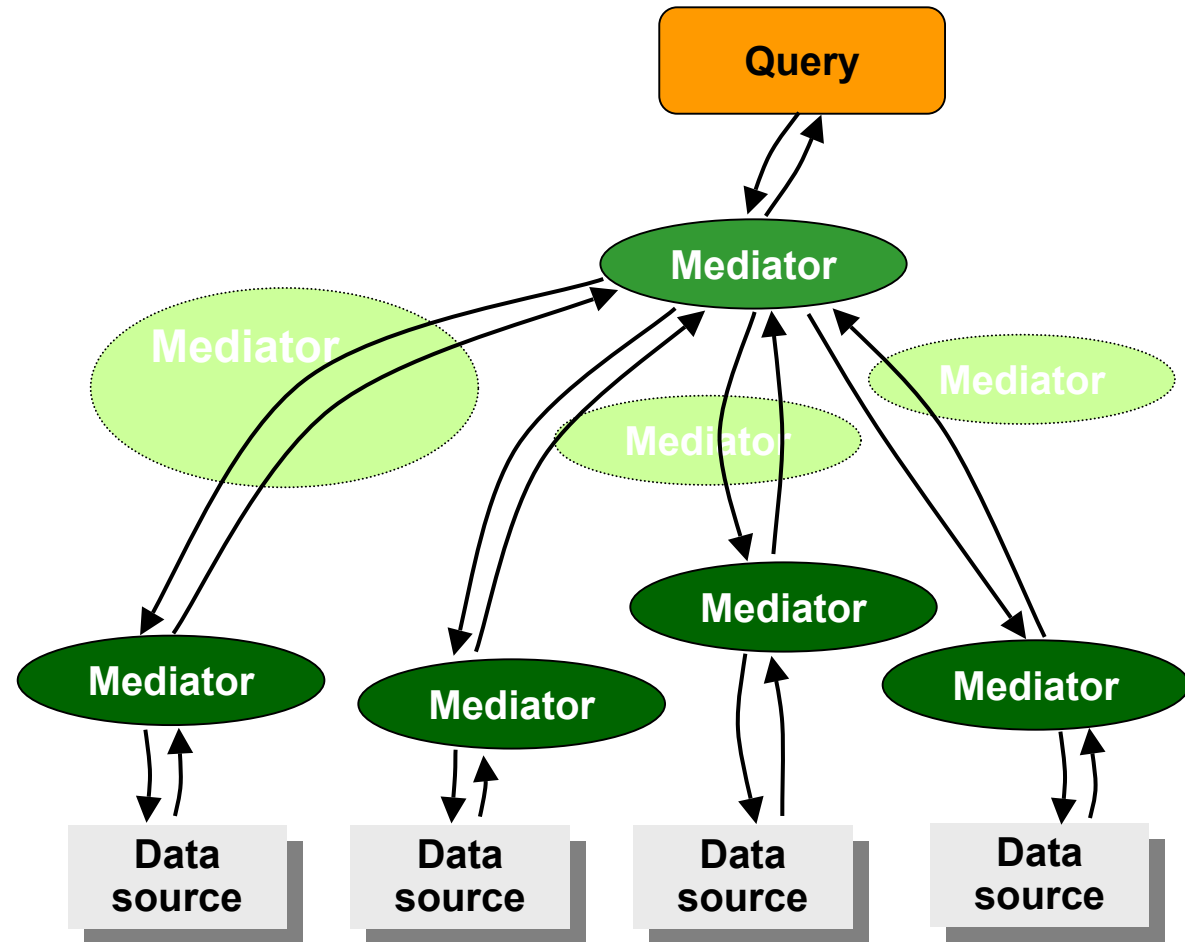
<2> Views are transparent

Query compilation:

- **Expand remote views**
- Decompose queries
- 'Push' sub-queries to mediators/sources
- Optimize sub-queries with local views
- Recursive compilation

Pros and cons:

- + remove composition overhead  
=> optimal query plans (?)
- high compilation cost
- violate view autonomy



**Distributed dataflow graph**

# View expansion in peer mediators

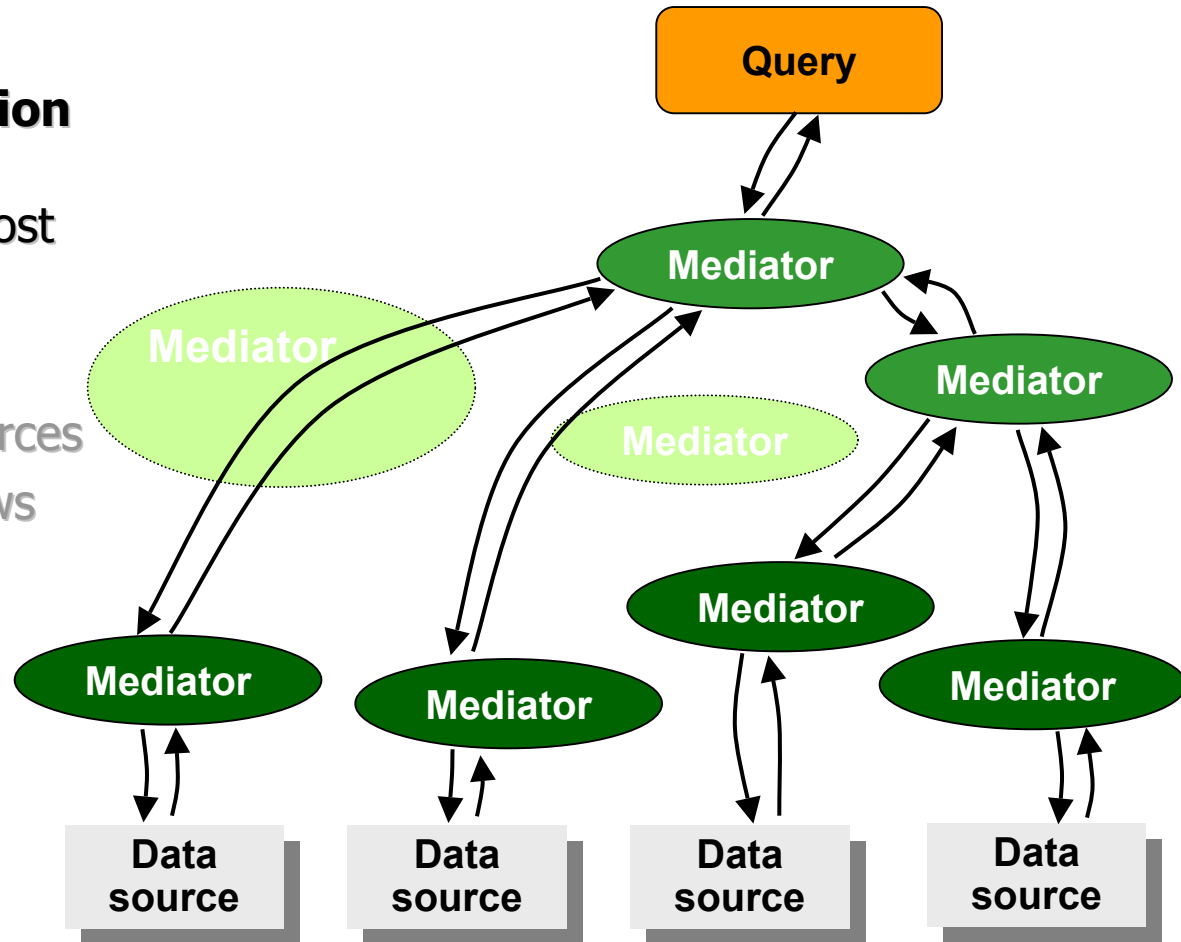
## <3> Selective view expansion

Query compilation:

- **Select remote views for expansion**  
*heuristic:*  
reduce compilation and execution cost
- **Expand selected views**
- Decompose queries
- 'Push' sub-queries to mediators/sources
- Optimize sub-queries with local views
- Recursive compilation

Resulting query plans:

- + acceptable compilation cost
- + remove most overhead =>  
acceptable query execution
- + respect view autonomy



**Distributed dataflow graph**

# Outline

---

- Introduction
- View expansion in peer mediators
- **Experimental evaluation**
  - Mediation scenario: bottom-up integration
  - Experimental results
    - query compilation
    - query execution
- Summary and conclusions

# Distributed mediation scenario

<1> data sources

- Many part suppliers
- Relational data sources

Mediator layers

1) Data source

S0

S1

S2

S3

...

S8

S9

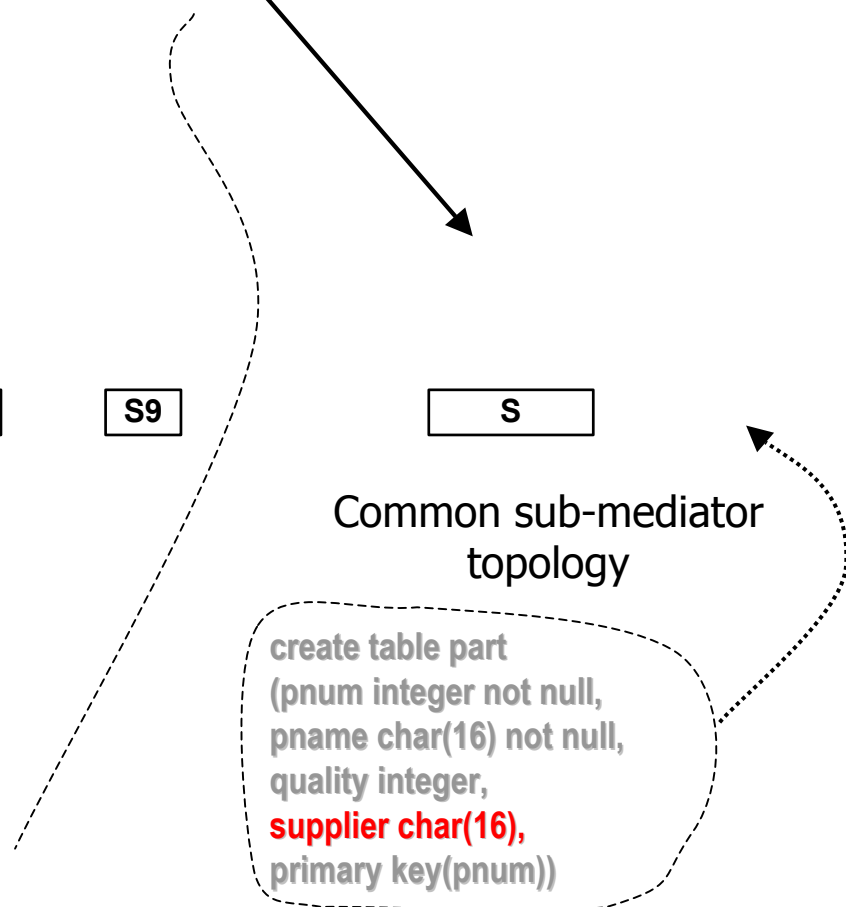
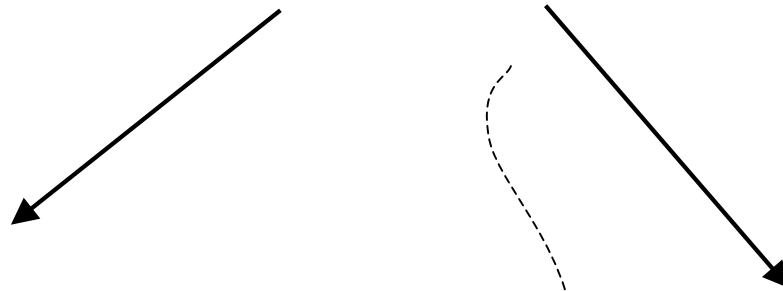
S

Tree topology

Common sub-mediator topology

create table part  
(pnum integer not null,  
pname char(16) not null,  
quality integer,  
primary key(pnum))

create table part  
(pnum integer not null,  
pname char(16) not null,  
quality integer,  
**supplier char(16),**  
primary key(pnum))



# Distributed mediation scenario

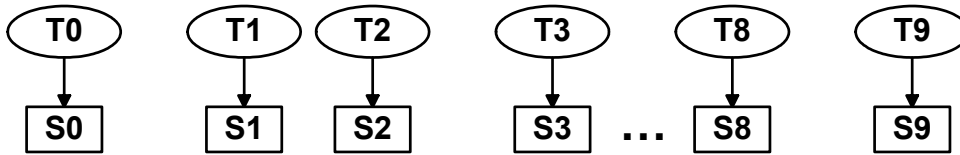
## <2> translators

Translator mediators:

- translate to the Common Data Model
- export views

Mediator layers

2) Translator



1) Data source

Tree topology

Common sub-mediator topology

create table part  
(pnum integer not null,  
pname char(16) not null,  
quality integer,  
primary key(pnum))

create table part  
(pnum integer not null,  
pname char(16) not null,  
quality integer,  
**supplier char(16),**  
primary key(pnum))

# Distributed mediation scenario

<3> integrators

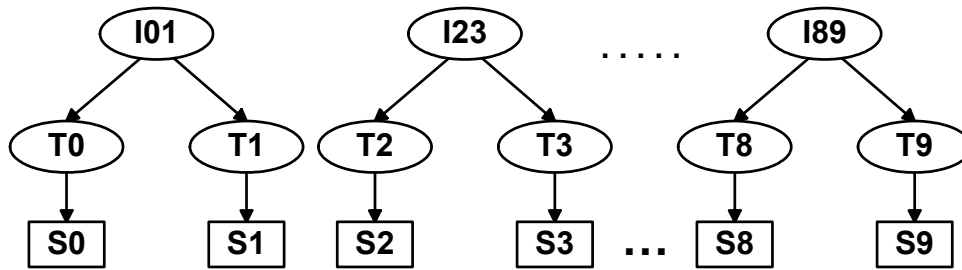
Integrator mediators:  
export logically equivalent views

Mediator layers

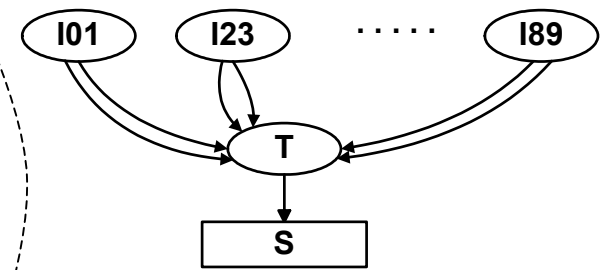
3) Integrator

2) Translator

1) Data source



**Tree topology**



**Common sub-mediator topology**

```
create view part@I01 as
select p0.pnum, p0.pname, combine_quality(p0.quality, p1.quality) as quality
/* TREE topology: */ from part@T0 p0, part@T1 p1
/* CSM topology: */ from part0@T p0, part1@T p1
where p0.pnum = p1.pnum;
```

# Distributed mediation scenario

## <4> client mediator

```
select p1.pname  
from part@l01 p1, part@l23 p2  
where p1.quality >= 7 and  
p2.quality >= 7 and  
p1.pnum = p2.pnum;
```

Test query:  
quality\_parts

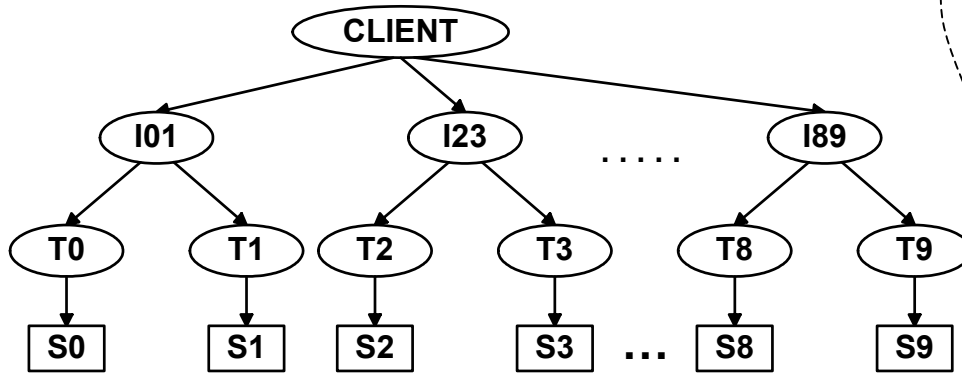
Mediator layers

4) Client

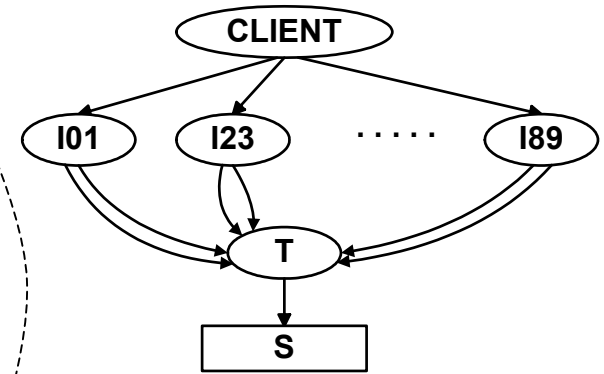
3) Integrator

2) Translator

1) Data source



Tree topology

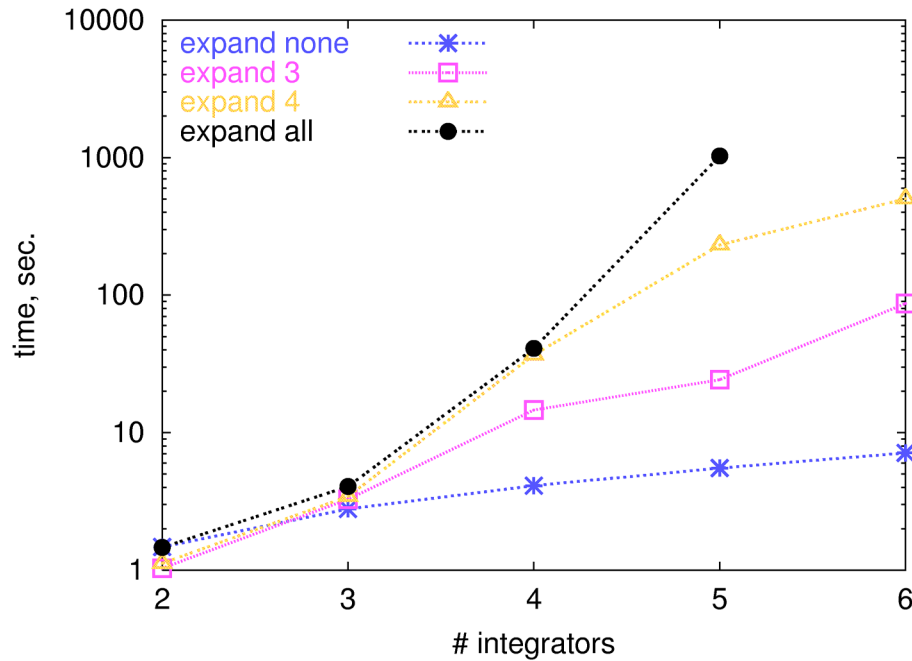


Common sub-mediator topology

# Experimental evaluation

## <1> query compilation

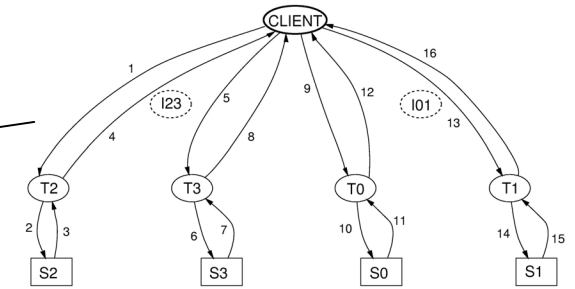
### TREE topology



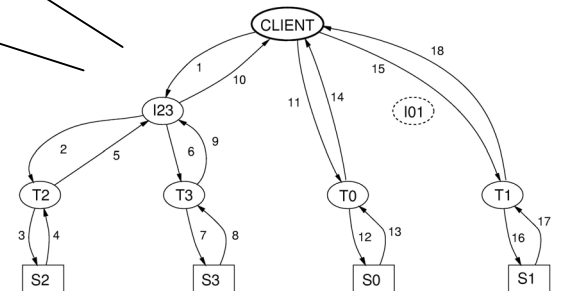
more expansions =>  
 more mediators =>  
 expensive compilation

Max. 10 - 15 mediators  
 with dynamic programming

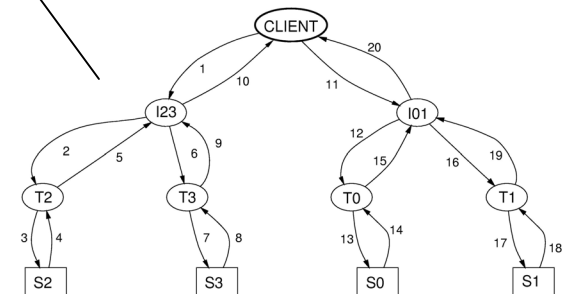
2 integrators



full view expansion



partial view expansion

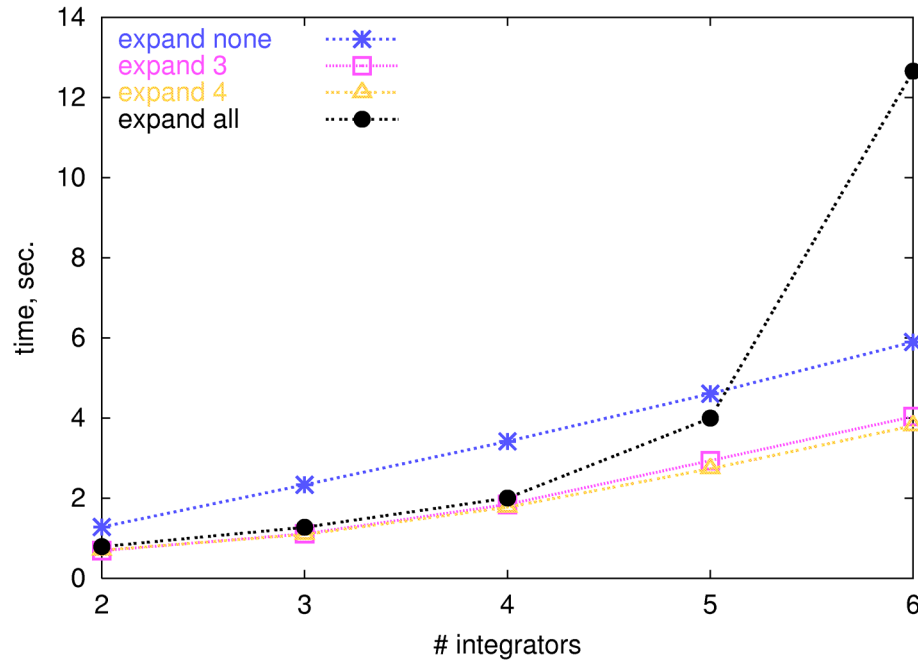


no view expansion

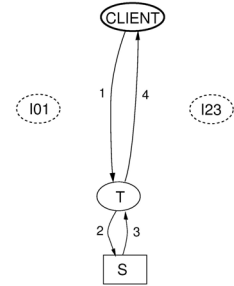
# Experimental evaluation

## <2> query compilation

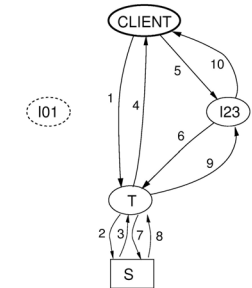
### Common sub-mediator topology



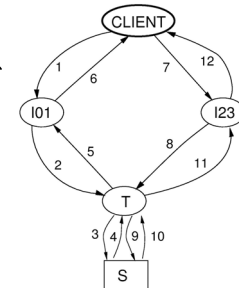
2 integrators



full view expansion



partial view expansion

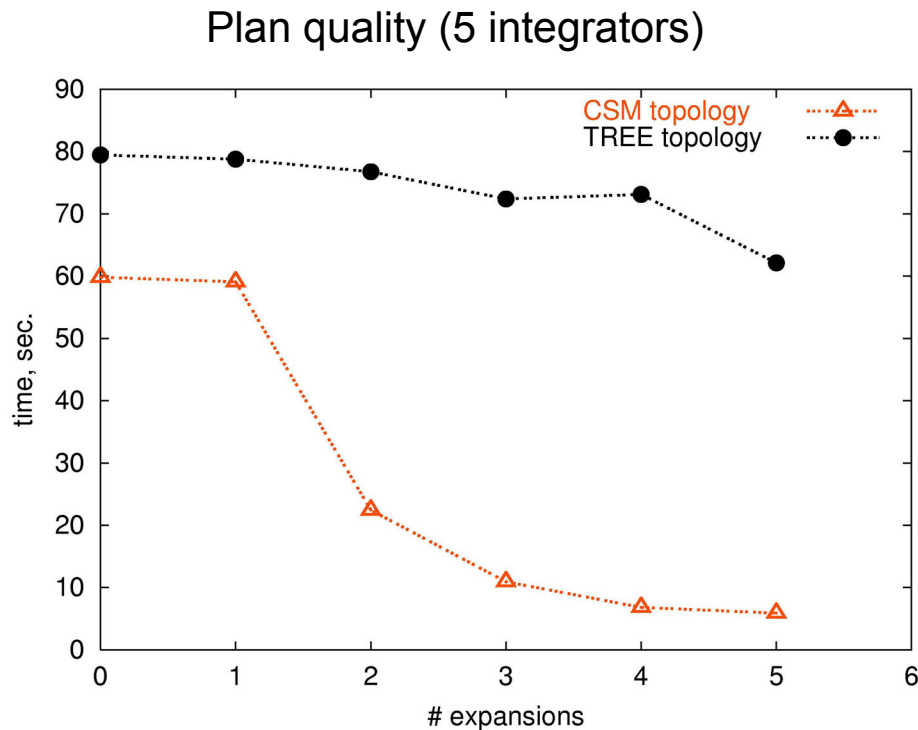


no view expansion

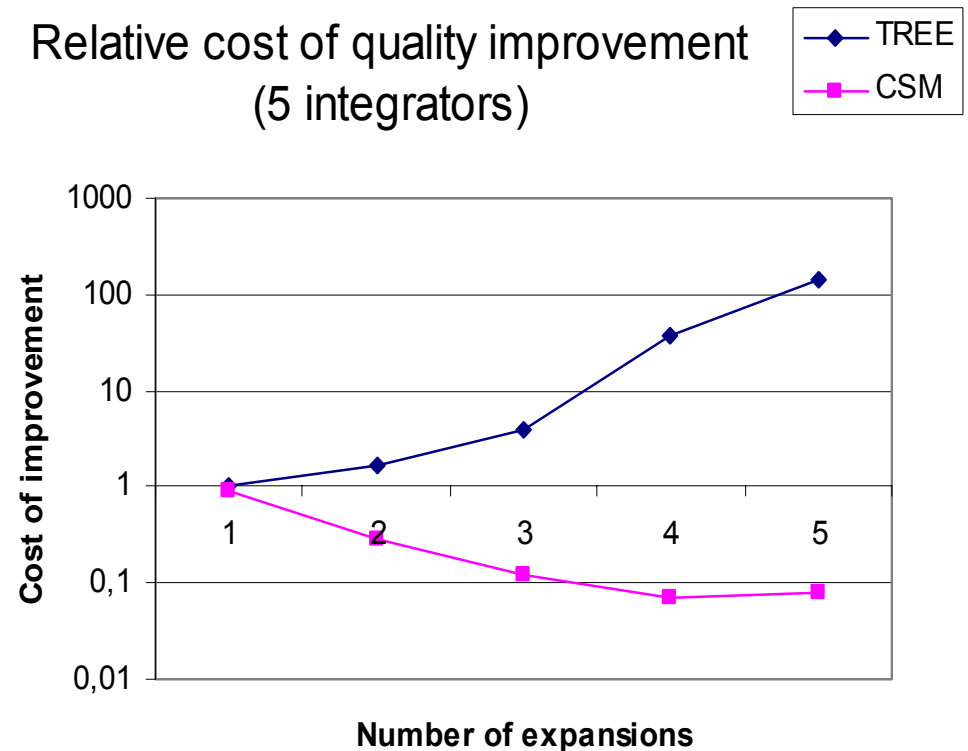
- more expansions => less mediators
- surprise: compilation time is reduced !!!
- reason: simplified queries (no redundancy)

# Experimental evaluation

## <3> execution plan quality



Relative cost of quality improvement  
(5 integrators)



- More expansions => better plan quality
- Logical topology matters:
  - plan quality improvement: TREE - 24%, CSM - 10 times
  - relative cost of improvement: TREE > 100 times, CSM < 1

# Outline

---

- Introduction
- View expansion in peer mediators
- Experimental evaluation
  - Mediation scenario
  - Experimental results
- **Summary and conclusions**

# Conclusions

---

- The reference approaches unsuitable for peer mediation systems
  - ‘black-box’:
    - TREE: low compilation cost, good quality
    - CSM: low compilation cost, poor quality
  - full expansion:
    - TREE: high compilation cost, small improvements
    - CSM: low compilation cost (!), big improvements
- Logical view integration topology is crucial
- Partial view expansion provides:
  - Sufficiently good quality
  - Low compilation cost
- Simple view expansion heuristic (DSVE):
  - If pure TREE topology - do not expand views
  - If pure Common Sub-Mediator - expand all views
  - If mixed topology - expand only common sub-views

# Summary

---

- Peer mediator architecture
    - efficient Internet-scale data integration
    - easy to tailor to existing infrastructure
    - mediators composed through multi-database views
  - Study of query processing for multi-database views
    - Query **optimization** cost matters
    - **Selective** expansion of views
    - Take into account logical integration topology
      - => utilize **knowledge** of view definitions
  - Prototype implementation based on AMOS II
- => AMOS II available at: <http://user.it.uu.se/~udbl/amos/>