Integrating Streams of Monitoring Data
Using Flexible View Hierarchies

Werner Nutt, Heriot-Watt University, Edinburgh

together with Andy Cooke, Alasdair Gray, Lisha Ma

CologNet Workshop Vienna, 23 August 2003

Overview

1. **Computational Grids** and Grid **Monitoring**
2. The **R-GMA** Grid Monitoring System
3. **Republishers:** A View Mechanism in R-GMA
4. Maintaining **Hierarchies** of Republishers

The Context of the Work: Grid Monitoring

A computational Grid

- is a collection of computing resources
  - *computers, storage devices, network bandwith*
  - *specialised equipment (supercomputers, databases)*
- accepts "jobs" in a job description language

A Grid Monitoring System answers the questions

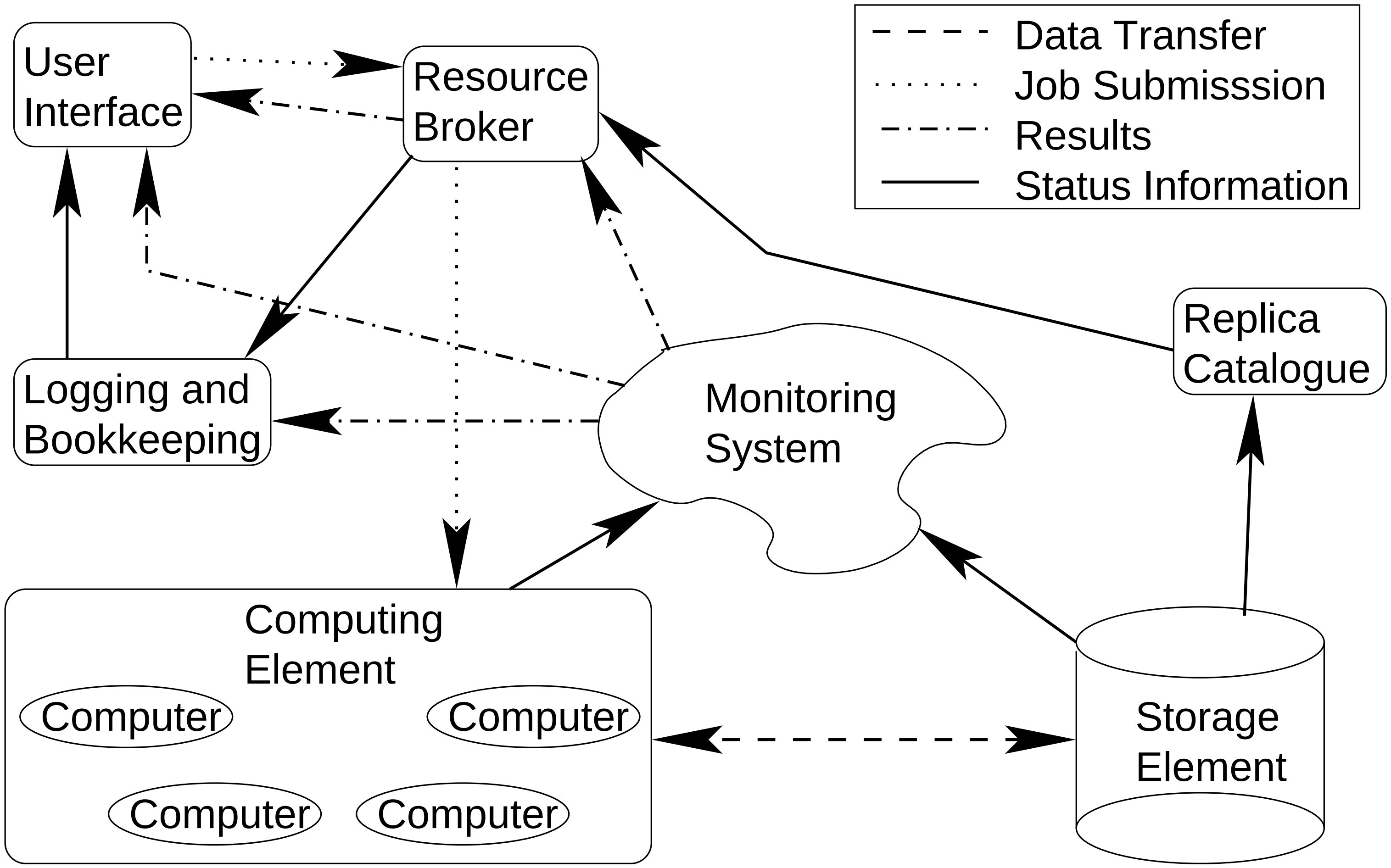
- What is the *current state* of the Grid?
- How did the Grid behave in the *past*?

R-GMA (= Relational Grid Monitoring Architecture)
is the monitoring system of the European DataGrid

Contributors to R-GMA

- Heriot-Watt, Edinburgh
  - Andrew Cooke, Alasdair Gray, Lisha Ma, Werner Nutt
- IBM-UK
  - James Magowan, Manfred Oevers, Paul Taylor
- Queen Mary, University of London
  - Roney Cordenonsi
- CCLRC/PPARC
  - Rob Byrom, Laurence Field, Steve Hicks, Manish Soni, Antony Wilson, Jason Leake
  - Linda Cornwall, Abdeslem Djaoui, Steve Fisher, Robin Middleton
- SZTAKI, Hungary
  - Peter Kacsuk, Norbert Podhorszki
- Trinity College Dublin
  - Brian Coghlan, Stuart Kenny, David O'Callaghan

Major Components of DataGrid



Monitoring Data Come in Two Kinds

A Grid monitoring system makes available two kinds of data

- static data "pools", e.g., databases on
  - *network topology*
  - *applications available (versions, licences, . . .)*
- "streams" of data, e.g.,
  - *sensor data (CPU load, network traffic, . . .)*

Data streams may give rise to data pools if they are archived

Today: *R-GMA is tailored towards streams, but not pools*

Examples of Monitoring Queries (1)

- **Resource Broker:**

  "Where is currently a computing element CE and a storage element SE such that

  - *user U is authorised to use CE and SE*
  - *CE has 5 CPUs available, each with at least 200 MB of memory*
  - *CE has software S1, S2, S3 installed*
  - *SE holds copies of files F1, F2*
  - *throughput between CE and SE is at least 500 Mbps?"*

"Latest-state query"

Examples of Monitoring Queries (2)

- **Visualisation Tool:**

  "What is the progress of the jobs of user U?
  How does their status change?"

  "Continuous query"

- **Network Administrator:**

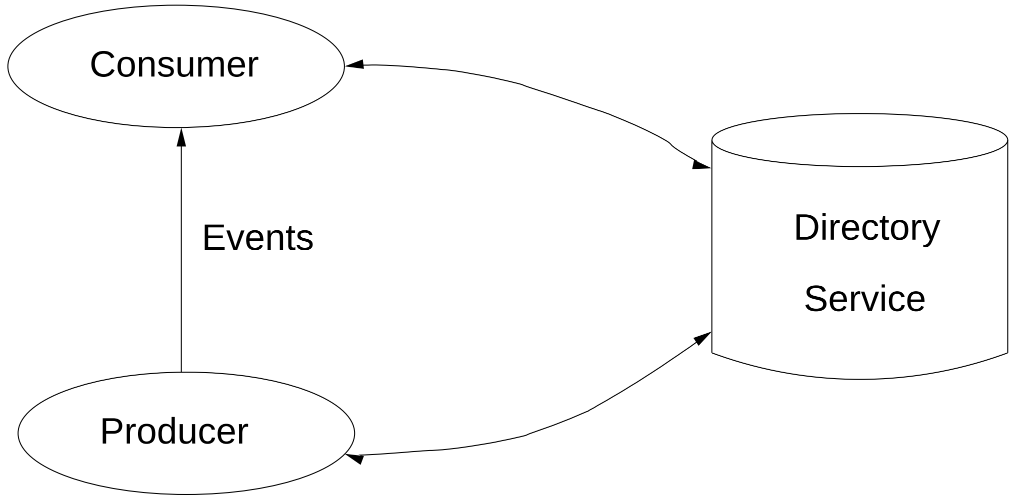
  "Between which nodes was yesterday
  the average transportation time for 1 MB packets
  higher than than 0.... seconds?"

  "History query"

Consumers and Producers

R-GMA provides an *infrastructure* that allows Grid components to play the *roles* of

- consumers of information
- producers of information



At the registry

- *consumers* register their *demand*
- *producers* register their *supply*

Consumers and producers are supported by "agents."

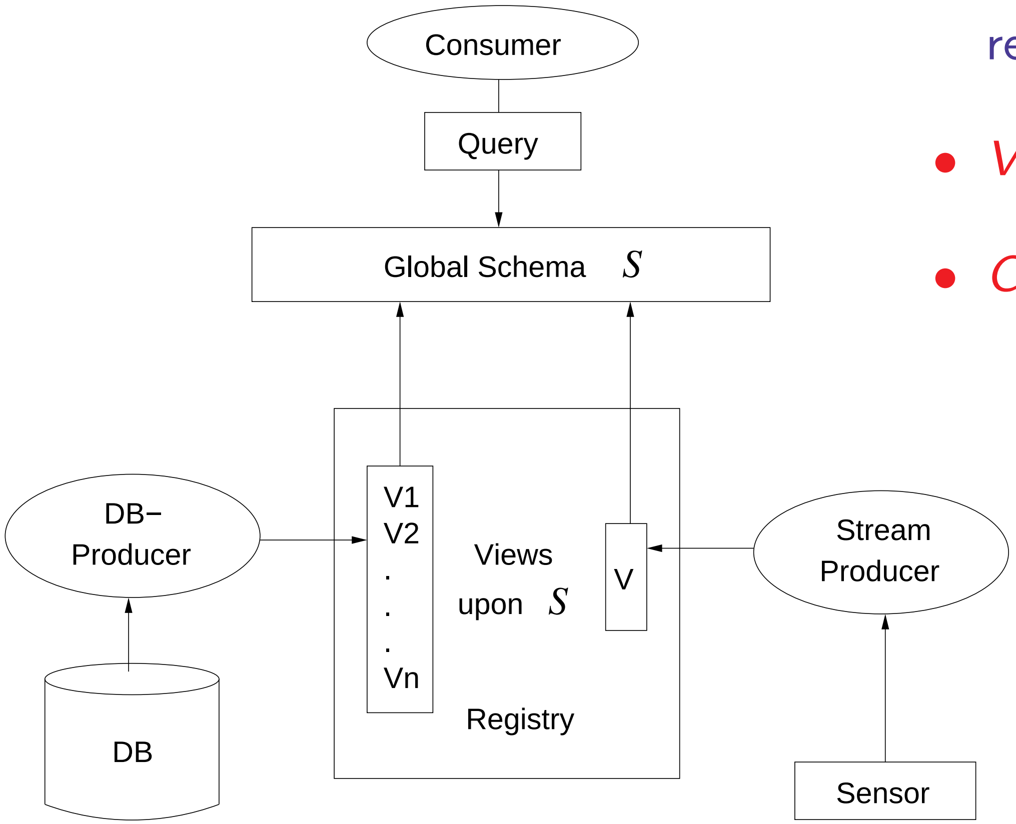
Questions about Consumers and Producers

- In which **language** do *producers* register their *supply* and *consumers* their *demand?*

- What is the **meaning** of a *registration?*

- How does a *consumer* **find** suitable *producers*? And how does a *producer* find suitable *consumers?*

R-GMA: A Virtual Monitoring Database



- *Language* of producers and consumers:
  relational queries (SQL)
- *Vocabulary:* Relations in a global schema
- *Consumer:* poses queries over
  global schema
- *Producer:*
  - has a type (*stream p.*, *database p.*)
  - publishes relations $R_1, \ldots, R_k$
  - for every R, registers a
    simple view V on the global schema

R-GMA's Components (So far . . .)

- **Schema:** fixes relations, attributes, *primary keys*
- **Producers:** publish local relations, described by views on schema
- **Consumers:** defined by query and type

  (continuous, latest state, history)
- **Agents:** make query plans, send and retrieve data
- **Registry:** finds suitable producers and consumers

$\rightsquigarrow$ *How can we answer latest-state and history queries?*

Republishers Publish Views Over Streams

A republisher

- consumes answers to a collection of *simple continuous queries*

  (selections currently)

- publishes
  - the answer stream: *"stream r."*
  - a db with latest answer for each key: *"latest-state r."*
  - a db with all answers: *"history r."*

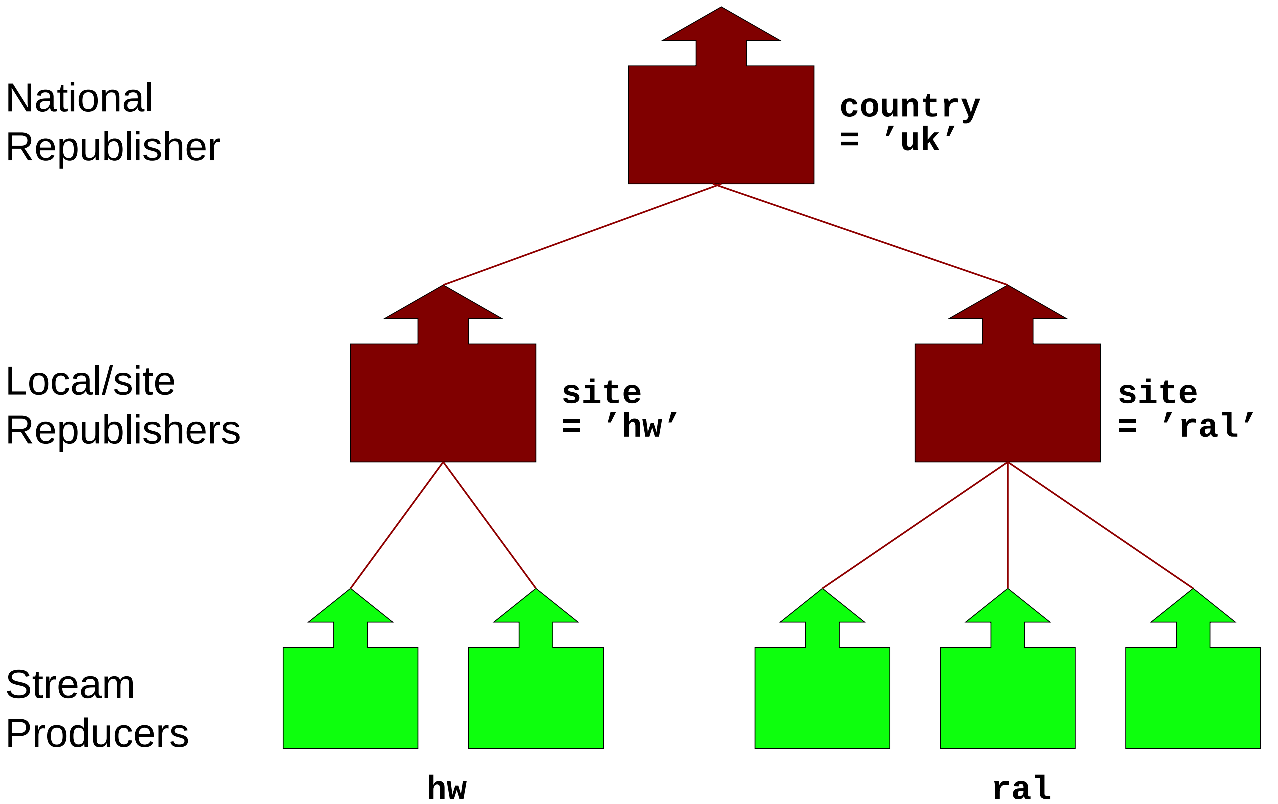
$\leadsto$ *Consumer agents have to choose the right republishers*

to answer complex queries

Stream Republishers Can Form Hierarchies

Stream relation

cpuLoad(<u>machineId</u>, <u>site</u>, <u>country</u>, load, timestamp)



Query Planning in the Presence of Republishers

- How can we plan the execution of a continuous query, i.e.,
  - *which publishers* (= producers or republishers) should we access?
  - *which query* should be posed over each publisher?

- How do we have to modify a plan
  - when a *new publisher* comes *online?*
  - when an *existing publisher* goes *offline?*

We study these questions for the simple case of selection queries

Stream Publishers and Their Relations

Notation:

- S stream *producer*
- R stream *republisher*
- P stream *publisher* (i.e., producer or republisher)

Assumption: every stream publisher publishes for a single relation
$\leadsto$ we identify the publisher and its relation

For each publisher P, the registry contains a *descriptive view*

$$\sigma_D(r)$$

which is a *selection* on some *global relation* r

Properties of Streams

A stream is

- **duplicate free** if it never outputs a tuple twice,
  taking into account the timestamps
- **weakly ordered** if the tuples belonging to one tuple of key values
  (= "channel") occur in the order of their timestamps
- **sound** *w.r.t. a view* $\sigma_D(r)$ if its tuples satisfy the view
- **complete** *w.r.t. a view* $\sigma_D(r)$ if it contains
  every tuple *in the system* that satisfies the view

Two streams are **disjoint** if they do not share any tuples

Properties of Stream Publishers

A publisher is

- **duplicate free**
- **weakly ordered**
- **sound** or **complete** w.r.t. a view

if its stream is

Two publishers are **disjoint** if their streams are

Stream Producers are Disjoint

Two views $\sigma_D(r)$ and $\sigma_{D'}(r)$ are **disjoint** if $D \wedge D'$ is *unsatisfiable*

Assumption: A stream *producer* can only register a view that is disjoint from all other producer views.

(This may not be achievable in practice,
but stream producers will usually be disjoint.)

Proposition. *If all stream producers are sound w.r.t. their views, then all stream producers are*

- *mutually* disjoint
- complete *w.r.t. their views* (because they are disjoint)

Query Plans

A **plan** for a query $Q = \sigma_C(r)$ is an expression

$$\sigma_{C_1}(P_1) \uplus \cdots \uplus \sigma_{C_m}(P_m)$$

where $P_1, \ldots, P_m$ are stream publishers, each described by a view $\sigma_{D_i}(r)$.

Note: A plan may give rise to duplicates.

Properties of Query Plans

A republisher for Q, realised by running a plan for Q, should be

- *sound* and *complete* w.r.t. Q
- *duplicate free*
- *weakly ordered*.

Question: *How can we recognise whether a plan has these properties?*

Characterisation: Soundness

Proposition. *Assuming soundness of the* P_i *w.r.t.* $\sigma_{D_i}(r)$, *the plan*

$$\sigma_{C_1}(P_1) \uplus \cdots \uplus \sigma_{C_m}(P_m)$$

is sound w.r.t. $\sigma_C(r)$ *if and only if*

$$C_i \wedge D_i \models C$$

for each $i \in 1..m$.

Note: A plan *remains sound* when new producers are added to the system.

Characterisation: Completeness

Proposition. *Assuming completeness of the P_i w.r.t. $\sigma_{D_i}(r)$, the plan*

$$\sigma_{C_1}(P_1) \uplus \cdots \uplus \sigma_{C_m}(P_m)$$

is complete w.r.t. $\sigma_C(r)$ if and only if we have

$$C \wedge E \models \bigvee_{i=1}^{m} C_i \wedge D_i$$

for every stream producer S with view $\sigma_E(r)$.

Note: A plan *may cease to be complete* when new producers are added.

Characterisation: Duplicate Freeness

Proposition. *Assuming the P_i are duplicate free and sound w.r.t. $\sigma_{D_i}(r)$, then the plan*

$$\sigma_{C_1}(P_1) \uplus \cdots \uplus \sigma_{C_m}(P_m)$$

is duplicate free *if and only if*

$$(C_i \wedge D_i) \wedge (C_j \wedge D_j)$$

are unsatisfiable for all $i, j \in 1..m$ with $i \neq j$.

Note: A plan *remains duplicate free* if new producers are added to the system.

Characterisation: Weak Order

We look for another property that ensures weak order:

If publisher P_i contributes to the plan **some** *values y for channel x, then it contributes* **all** *values y for channel x.*

We say the plan is **faithful** if this holds for all P_i.

Clearly: If the P_i are weakly ordered and our plan is faithful, then our plan is weakly ordered.

How can we check whether a plan is faithful?

Characterisation: Faithfulness

Notation: we write the *query condition* C as $\boxed{C(x, y)}$,
where $\boxed{\ x\ }$ stands for the *key attributes* of r,
and $\boxed{\ y\ }$ for the *non-key attributes.*

Similarly: $\boxed{C_i(x, y)}$ for C_i and $\boxed{D_i(x, y)}$ for D_i

Abbreviation: $\boxed{F_i(x, y)} \;=\; C_i(x, y) \wedge D_i(x, y)$

Proposition: *A plan is faithful if and only if we have*

$$\exists y.\, (C(x, y) \wedge F_i(x, y)) \quad \models \quad \forall y.\, (C(x, y) \;\rightarrow\; F_i(x, y))$$

Faithfulness for Decoupled Conditions

Faithfulness is easy to check if key conditions and value conditions are decoupled, i.e.

$$C(x, y) = C^k(x) \wedge C^v(y)$$

(and C_i, D_i, F_i analogously)

Proposition. *Suppose* $C(x, y) = C^k(x) \wedge C^v(y)$ *and* $F_i(x, y) = F_i^k(x) \wedge F_i^v(y)$*. Then*

$$\exists y. (C(x, y) \wedge F_i(x, y)) \models \forall y. (C(x, y) \to F_i(x, y))$$

holds if and only if one of the following holds:

1. $C^k(x) \wedge F_i^k(x)$ *is unsatisfiable;*
2. $C^v(y) \wedge F_i^v(y)$ *is unsatisfiable;*
3. $C^v(y) \models F_i^v(y)$*.*

Relevant Publishers

Question: When can a publisher for $V = \boxed{\sigma_{D^k \wedge D^v}(r)}$ potentially contribute to answering $Q = \boxed{\sigma_{C^k \wedge C^v}(r)}$?

Answer: If for *some channel* it delivers *all values,* i.e.,

1. $C^k \wedge D^k$ is satisfiable
2. $C^v \models D^v$.

Such a publisher is **relevant** for Q.

Note: For some simple conditions we can write SQL queries that retrieve all relevant publishers for Q.

Subsumption of Publishers

Consider: publishers P, P' with views $\boxed{\sigma_D(r)}$, $\boxed{\sigma_{D'}(r)}$

a query $Q = \boxed{\sigma_C(r)}$

Then

- P is *subsumed* by P' w.r.t. Q, written $P \preceq_Q P'$, if

$$C \land D \models D'$$

- P is *subsumed* by P', written $P \preceq P'$ if

$$D \models D'$$

Strict subsumption: $P \prec_Q P'$, $P \prec P'$

Query Planning

For a given query Q, R-GMA goes through *two phases:*

1. The **registry** retrieves those *relevant publishers* that are *maximal w.r.t.* "$\preceq_Q$"
2. The **consumer agent** chooses a *subset* of those publishers and poses *local queries*

How should the consumer agent choose the subset?

Maximal Relevant Publishers

Observation: On the maximal relevant publishers for Q,
the relation "$\preceq_Q$" is an equivalence relation.

The agent chooses *one* publisher from each class, which gives a sequence

$$\langle P_1, \ldots, P_m \rangle$$

(called a **supplier sequence**).

Question: Which publisher should the agent choose
from such an equivalence class?

We think: One that is *minimal* w.r.t. "$\preceq$"!

(... to avoid too high a load on the general republishers)

Canonical Plans

$\langle P_1, \ldots, P_m \rangle$ supplier sequence $\qquad$ P_i described by $\sigma_{D_i}(r)$

The **canonical plan** for the sequence is

$$\sigma_{C_1}(P_1) \uplus \cdots \uplus \sigma_{C_m}(P_m)$$

where

- $C_1 = C$
- $C_i = C \land \neg D_1 \land \cdots \land \neg D_{i-1}$.

Properties of Canonical Plans

Proposition. *If* $\langle P_1, \ldots, P_m \rangle$ *is a supplier sequence for query* Q*, then the canonical plan*

$$\sigma_{C_1}(P_1) \uplus \cdots \uplus \sigma_{C_m}(P_m)$$

is

- sound *and* complete *for* Q
- duplicate free
- weakly ordered.

Irreducible Plans

A plan is **irreducible** if none of its publishers can be dropped.

- Canonical plans are *irreducible*

  if conditions are expressible in Horn logic, e.g.,

  – conjunctions of "$attr\ \{=,<\}\ val$" or

  – conjunctions of "$attr\ \{=,>\}\ val$"

- Canonical plans are *not* always *irreducible* if conditions are

  – conjunctions of "$attr\ \{\leq,>\}\ val$"

  In that case, irreducibility is *NP-hard*.

Plan Adaptation

Plans **have to** be modified if

- *new producers* are registered, or
- *existing republishers* are dropped.

Plans **may be** modified if

- *new republishers* are registered, or
- *existing producers* are dropped.

All these tasks are straightforward with the theory presented.

Open Problems

- **Protocols** for the transition between plans
- **View languages** precise enough to describe producers
  (do we need joins between streams and static tables?)
- **Sound**, but **incomplete** subsumption algorithms
  for expressive conditions
- **Creating** good republishers for a given **query load**
  (both, stream republishers and archivers)