Carleton UNIVERSITY

Virtual Data Integration

Leopoldo Bertossi

Carleton University
School of Computer Science
Ottawa, Canada

www.scs.carleton.ca/~bertossi

bertossi@scs.carleton.ca

Chapter 1: Introduction and Issues

Data in Different Forms

There is a large and increasing number of data sources

People and companies need to integrate data and the systems that handle that data

- Data in DBMSs: relational, OO, XML, ...
- Legacy data in different formats and systems
- Text files repositories
- Spread sheets

- Data on the Web
  - Non- and semi-structured data in the WWW: Plain text files, HTML text files, native XML
  - Organizational databases
  - Libraries, catalogues, etc.
  - Data in research repositories: genome databases, scientific databases, environmental databases, etc.
  - Web services
  - Semantic Web
  - Knowledge-Based Systems
  - Ontologies: Structurally and semantically reach domain descriptions with associated data

Database systems have to inter-operate, cooperate, and coordinate with each other

Data has to be shared, exchanged, integrated, ...

- In particular, it has to be reconciliated

  Syntactically: compatible formats and data types

  Semantically: compatible meanings

Data has to be queried

What if the data is spread out in different sources?

Some Forms of Data Integration

There are different approaches and paradigms for data integration, some of them are

- Materialized: physical, integrated repository is created

  Data Warehouses: physical repositories of selected data extracted from a collection of DBs and other information sources

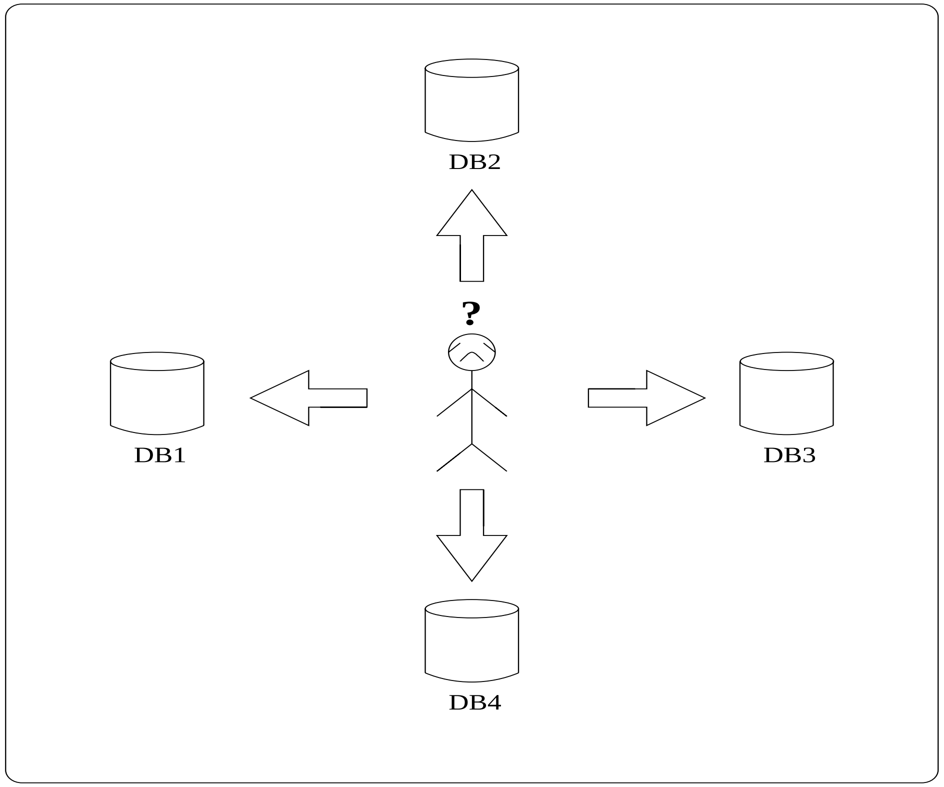
- Mediated: data stay at the sources, a virtual integration system is created $\Longleftarrow$

- Federated and cooperative: DBMSs are coordinated to collaborate

- Exchange: Data is exported from one system to another
- Peer-to-Peer data exchange: Many peers exchange data without a central control mechanism

  Data is passed from peer to peer upon request, as query answers

Mediator-Based Data Integration



How can users confront such a large and increasing number of information sources?

Interacting with each of the sources by means of queries?

- Considering all available sources?
- Selecting only those to be queried?
- Querying the relevant sources on an individual basis?
- Handcraft the combination of results from different sources?

A long, tedious, complex and error prone process ...

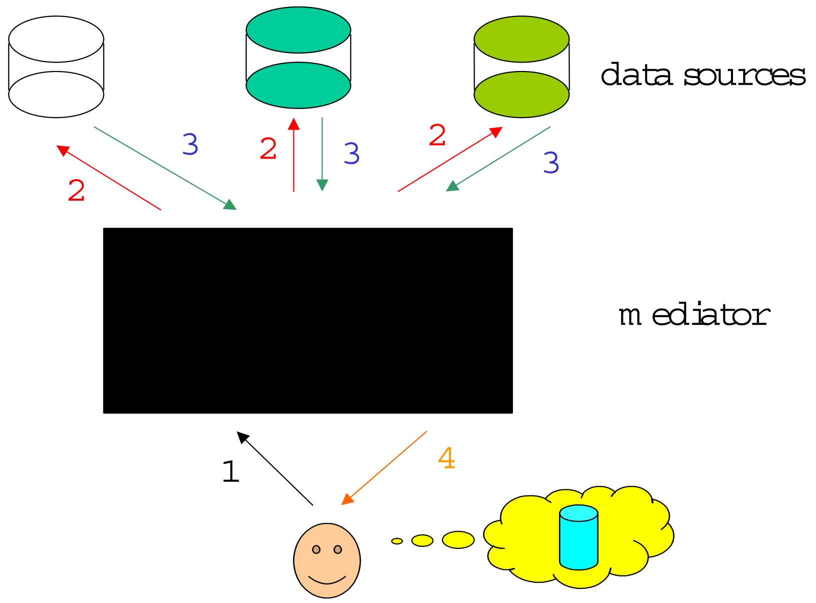
Independently of other “intrinsic” issues like:

- Incomplete data
- Redundant data
- Inconsistent data
- Diverse data formats
- Different forms of external data presentation
- Different and possibly limited query languages (if any)

  Common when interacting with the WWW: keyword search, fixed query patterns and templates, ...
- Restrictions on access to data

One way to go:

Use a mediator-based information integration system



A mediator is a software system that offers a common query interface to a set of heterogeneous information sources

We have:

- Collection of data sources that are independent and autonomous
- A virtual “database” is created which is accessed via the *mediator*
- A mediator that creates the illusion on users of being interacting with a real database
- Queries are posed and answered via the mediator

More precisely, the mediator:

- Accepts the participation of different data sources
- Contains information about the contents of the data sources
- Collects data from sources upon request; at query time
- Logically integrates the different data sources by means of a unifying, global or mediated schema
- Receives queries from users that are expressed in the language of the global schema
- In order to answer global queries, it sends appropriate queries to the sources
- Combines the answers received from the sources to build up the final answer to the user

Main Issues and Features Around VDISs

- Autonomy:

  Update operations on data sources via the mediator are not allowed

  Individual data source are updated in an independent and autonomous manner

  Sources do not necessarily cooperate with each other

  Data sources are mutually independent and may participate in different mediated systems at the same time

- Data location and flexibility:

  Data is kept in the local, individual sources, and extracted at the mediator's request

  System allows sources to get in and out

  Set and number of participating sources should be flexible and open

  Mediator has to know what kind of data is offered by the sources and how they relate to the unifying global schema

  A problem of describing data and specifying mappings between data schemas

  Description formalism should be expressive, computationally easy to use, and easy to maintain

- Data presentation:

  The interaction with the system is realized through queries (and answers to them)

  Data sources have their own schema, the virtual database has its own data presentation schema

  Database schemas are a special kind of metadata, i.e. data about data

  In this case, schemas say how the data is logically structured

  Establishing the correspondence between the schemas of the sources and the global schema is a form and instance of metadata management

  Metadata management appears in different forms in all the subjects around data integration and related subjects

- Problems with data:

  Global system is responsible for solving/addressing problems with data, like:

  Redundancy: to avoid unnecessary computations

  Complementarity: data of the same kind may be spread through different sources and has to be detected and combined

  Consistency: two sources, independently, may be consistent, but taken together, possibly not

  E.g. Same ID card number may be assigned to different people in different sources

Existing VDISs offer almost no support for consistency handling

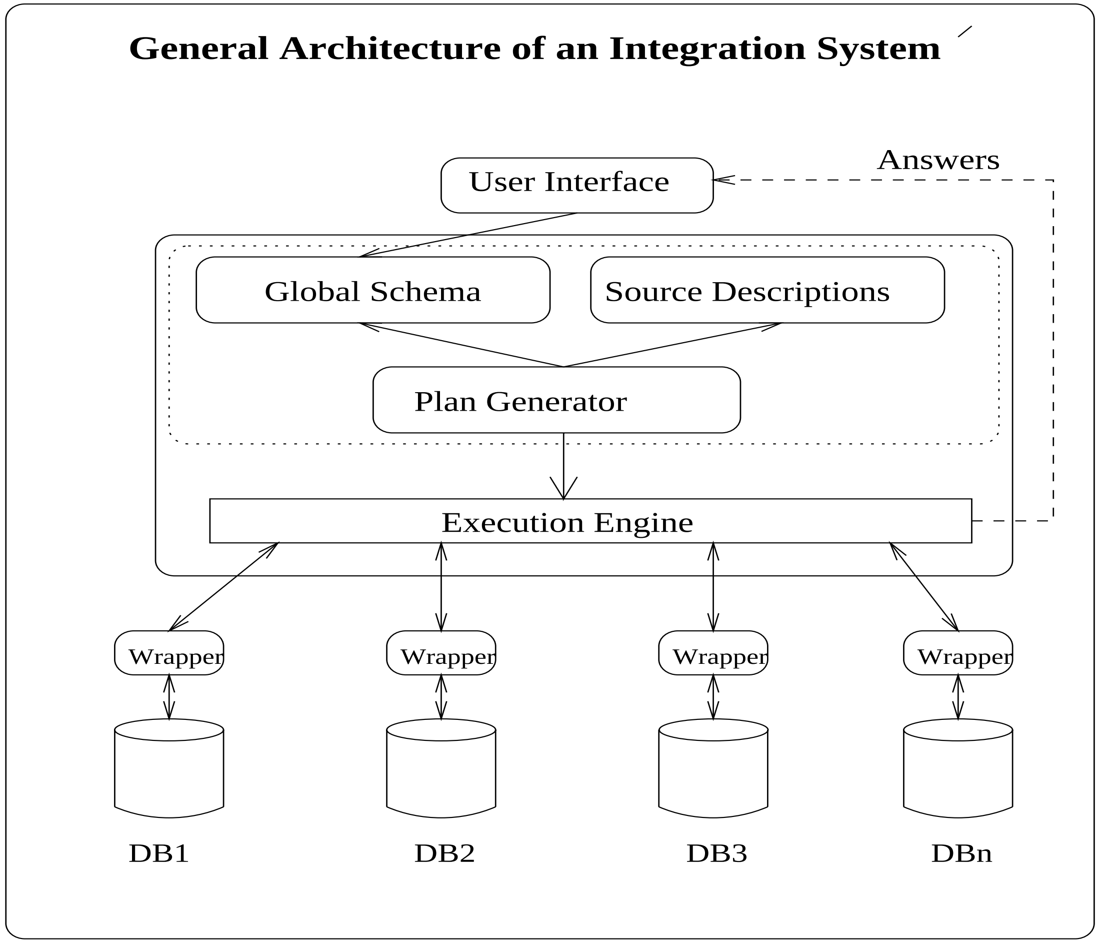
Many interesting research issues here ...

Even commercial DBMSs for stand alone databases (no integration) offer limited *general purpose support* in this direction

Consistency maintenance has to be achieved at the application level: rejection or compensation of updates via triggers or application programs

Chapter 2: Components of a Virtual Data Integration System

General Architecture



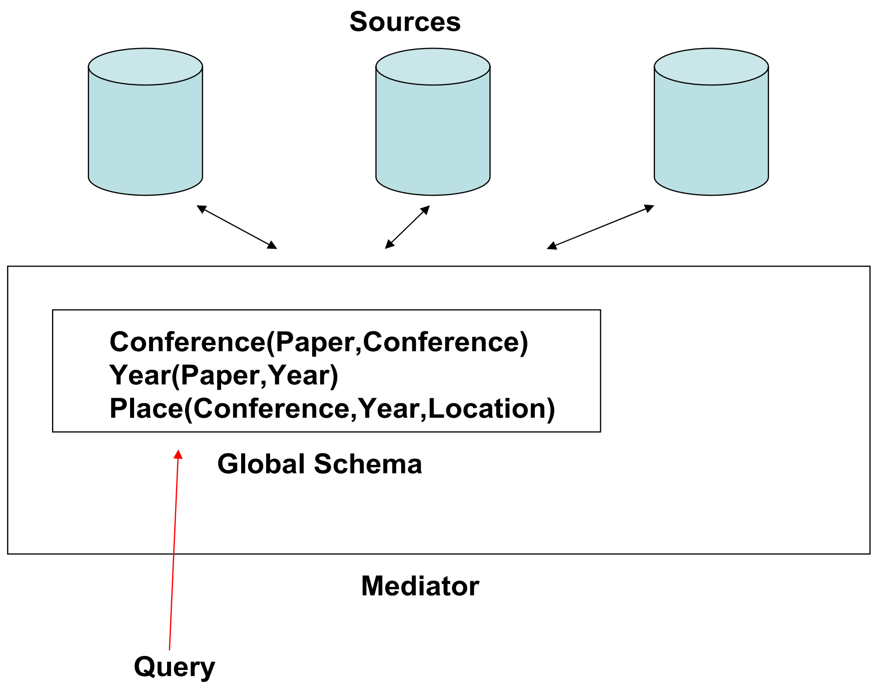
Main components of a mediator-based VDIS

Global or Mediated Schema:

- Schema used to present and export the data from the VDIS
- For example, if it is a relational schema, then it is a set of names for relations (tables), their attributes, etc.
- Application dependent
- Like in a normal, usual relational DB, from the user point of view
- Data is not stored in “tables” of the global schema, but in the sources
- The DB “instance” corresponding to the global schema is virtual

- User poses queries in terms of the relations in the global schema
- Relationship between the global schema (or DB) and the data sources (and their local schemas) is specified at the mediator level

Example: Global schema for a DB "containing" information about scientific publications:



$Conference(Paper, Conference), \quad Year(Paper, Year),$
$Place(Conference, Year, Location)$

User wants to know where conference PODS'89 was held

Query to global system: a simple selection

SELECT Location
FROM Place
WHERE conference = 'pods' AND year = '1989';

Or: $\Pi_L(\sigma_{C=pods,Y=1989}(\texttt{Place}(\texttt{C},\texttt{Y},\texttt{L})))$

Or using a rule based query language, like Datalog:

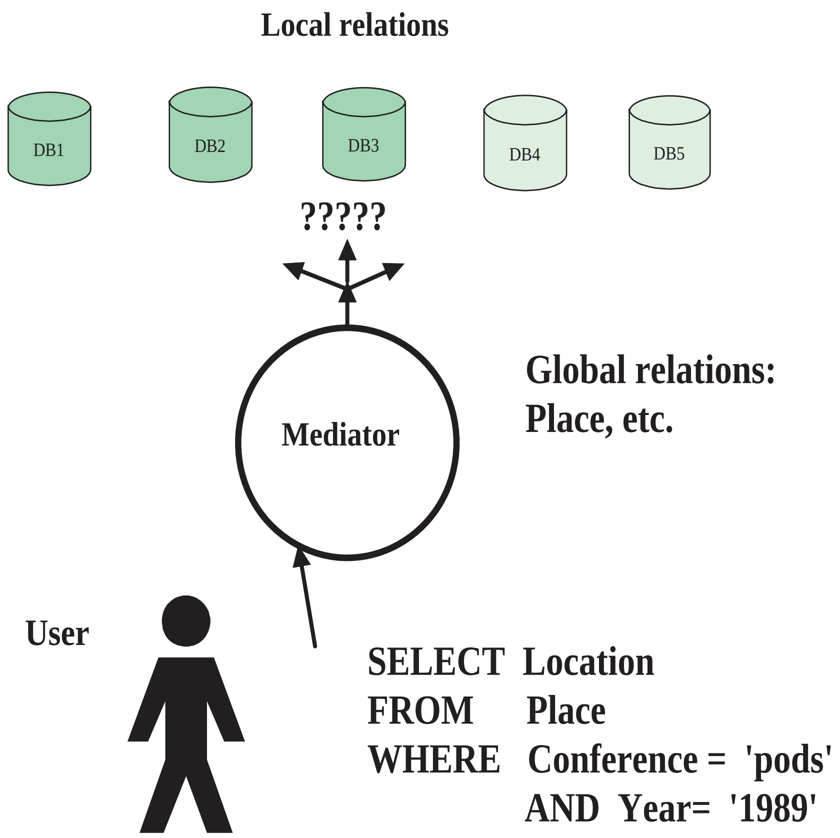
$$Q: \quad Ans(L) \leftarrow Place(pods, \mathit{1989}, L)$$

Predicate Ans contains the answers, that are obtained by computing the RHS (the body) of the rule

But data is not in table *Place*

A query plan has to be generated to:

- identify relevant data sources
- identify relevant data in them
- determine (sub- or local) queries to be sent to the sources
- combine the answer sets obtained from the sources into one final answer set



```
SELECT  Location
FROM     Place
WHERE   Conference =  'pods'
         AND  Year=  '1989'
```

Wrapper:

- Module that is responsible for wrapping a data source, so that it can interact with the rest of the VDIS
- To present/export the data in the source as needed by the mediator
- It presents the data source as a convenient database, with the right schema, structure, and data
- This presentation schema may be different from the real, internal one (if any)
- Data provided by the wrapper my be different from the real one in the local source
- Possibly several wrappers for a data source

- Wrapper may have to perform some preliminary transformation, cleaning, etc., before exporting the data to the VDIS
- Provides data as requested by the execution engine

We will not say much about wrappers

We will conceptualize each data source as an appropriate (relational) database, with the right schema and data

The wrapper will be implicit and taken for granted

Description of the Sources:

- Mediator needs to know what is available in the sources and how that data relates to the global schema
- How this data is described will heavily determine how to compute query plans
- The sources are described by means of a set of logical formulas

  Like the formulas used to express queries and define views in terms of base tables in a relational DB

  We use logical formulas, usually incarnated as SQL queries or SQL view definitions, or Datalog

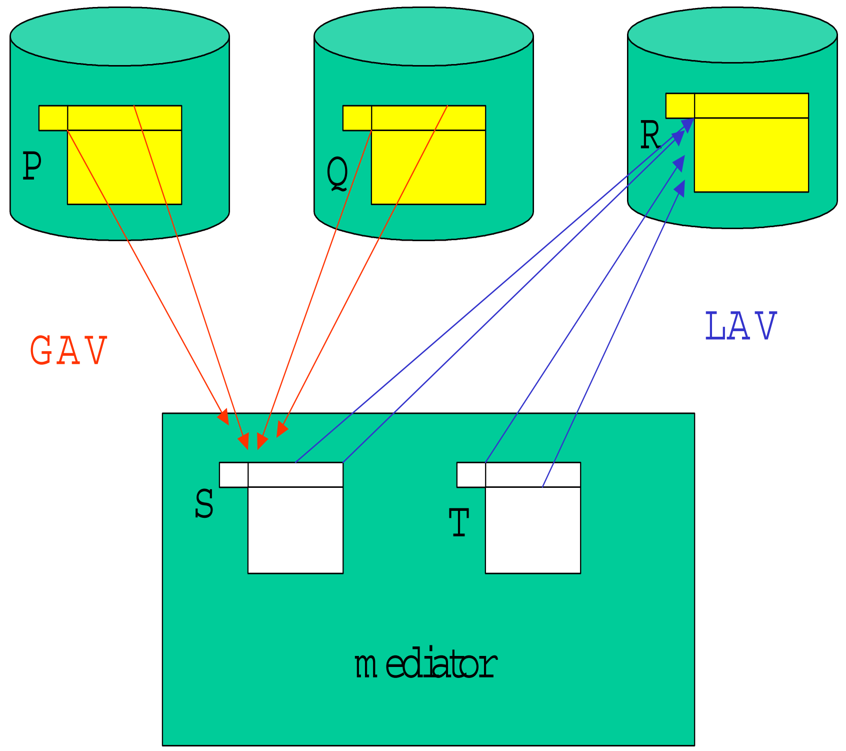
- Those formulas relate the global (or mediated) schema with the local schemas

  They define the mappings between the global schema and the local schemas

  How is this relationship described?

Main classical approaches:

- Global-as-View (GAV): Relations in the global schema are described as views over the tables in the local schemas
- Local-as-View (LAV): Relations in the local schemas (at the source level) are described as views over the global schema



LAV: Mediated schema may contain details than are not in a source

GAV: Sources may contain more (and unnecessary) details wrt the mediated schema

Also common, more recently:

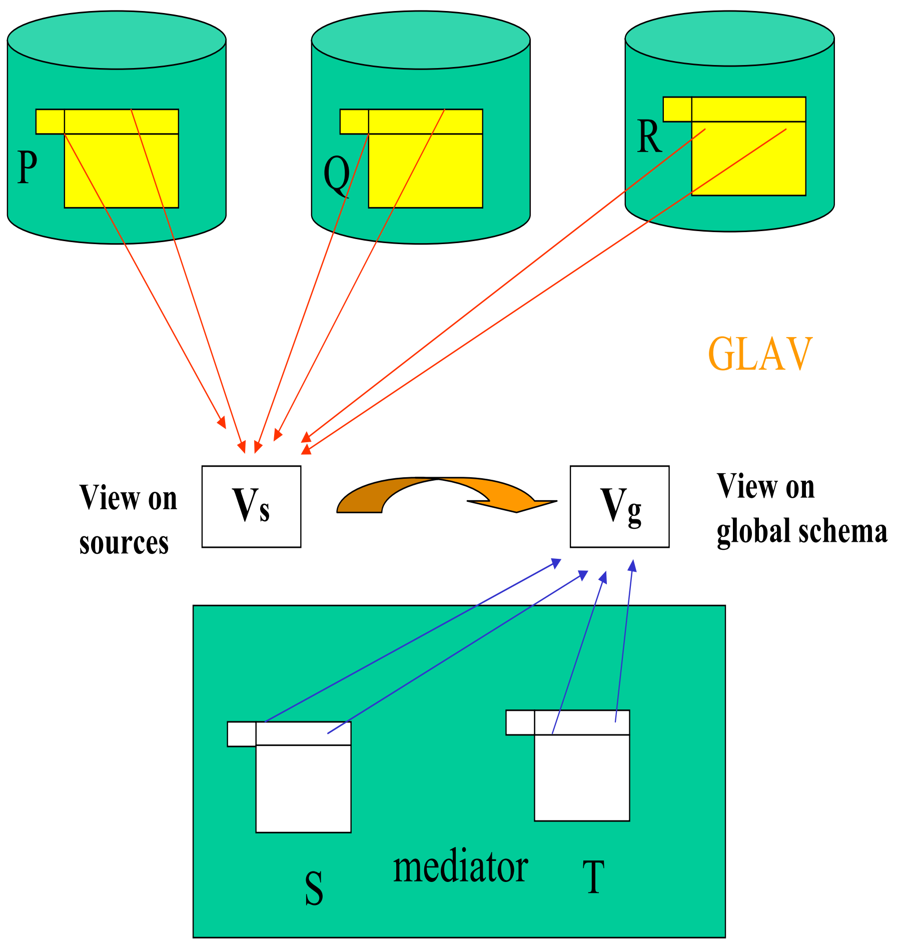
- Global-and-Local-as-View (GLAV): A correspondence between views on the global schema and views over the local schema are established and described

  Under certain assumptions on the sources, GAV and LAV can be obtained as special cases

  Gives more expressive power, more natural source descriptions, and more natural mediated schemas

M. Friedman, A. Levy, T. Millstein. Navigational Plans for Data Integration. Proc. AAAI'99.

The mappings (arrows in the pictures) require a language and a semantics ... Coming ...


P
Q
R
GLAV
View on sources
V_s
V_g
View on global schema
S
mediator
T

Plan Generator:

- It gets a user query in terms of global relations
- It uses the source descriptions and rewrites the query as a query plan

  Which involves a set of queries expressed in terms of local relations
- This is the most complex part
- Rewriting depends on mappings (LAV, GAV, or GLAV)
- Query plan includes a specification as to how to combine the results from the local sources

Execution Engine:

- Query plan is just that, a plan; is has to be executed
- Execution engine gets the query plan and distributes the sub-queries to the relevant sources (to their wrappers)
- Then gets the answers from the local sources and composes the answer for the user

  At this point the composer should solve inconsistencies, etc.
- Unless the plan generator is able to anticipate potential inconsistencies and "solve them in advance", when the plan is being generated ... we'll see ...

Description of Data Sources

Given as logical formulas in terms of relations in the global schema and the relations in the data sources (or their wrappers)

Since under any of the approaches mentioned above (GAV, LAV, GLAV) we are defining views, we will use a common language to define views in relational databases: Datalog formulas for view definitions

Example: A relational database with base relations

$Employee(Id, Name, Position), \quad Salary(Id, Amount)$

The definition of a view that gives the employee names and salaries

$$NameSal(x, y) \ \leftarrow \ Employee(u, x, v), Salary(u, y)$$

This Datalog rule says that, in order to compute the tuples in the relation on the LHS (the head), we have to go to the RHS (the body) and compute whatever is specified there

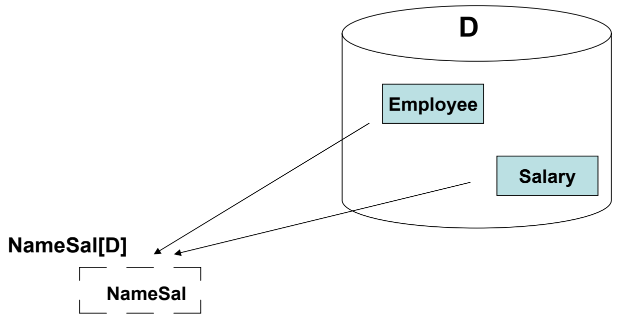
The values of variables in the body that do not appear in the head are projected out after computing a join via Id (variable u) of the two base relations

View is defined as a conjunctive query over the base relations

Beware: The semantics of Datalog rules does not follow classic predicate logic semantics

The tuples in the (usually virtual) extension of the view are only those that can be obtained by propagating body to head

Given an instance D for the base schema, we obtain a possibly virtual extension $NameSal[D]$ for the view



We will keep using the Datalog language for defining views, but being careful about their semantics in the context of virtual data integration

An Example of GAV

Relations in global schema are described as *views* of the relations in the (union of the) local schemas

Conceptually very natural: Usually views are virtual relations defined in terms of material relations (tables)

Since global relations are virtual and local sources are material, it is conceptually as usual ...

Example: We have data sources with different kinds of data about movies

We integrate those data sources into a mediated system

New sources may be available later, some may disappear, ...

As modelers of the integration system, we decide what common data interface or schema we want to offer to the outside world

Local sources:

- $DB_1(Title, Dir, Year)$
- $DB_2(Title, Dir, Year)$
- $DB_3(Title, Review)$

The first two are complementary, the third of a different, but related kind

A global relation "containing" movies and their years:

$$\begin{array}{lcl} \mathit{MovieYear}(\mathit{Title}, \mathit{Year}) & \leftarrow & DB_1(\mathit{Title}, \mathit{Dir}, \mathit{Year}) \\ \mathit{MovieYear}(\mathit{Title}, \mathit{Year}) & \leftarrow & DB_2(\mathit{Title}, \mathit{Dir}, \mathit{Year}) \end{array}$$

A disjunctive view (defined as a disjunction of conjunctive queries)

Defined by two Datalog rules

$\mathit{MovieYear}$ defined as the union of two projections, of DB_1 and DB_2 on attributes $\mathit{Title}, \mathit{Year}$, i.e.

$$\mathit{MovieYear} \;:=\; \Pi_{\mathit{Movie},\mathit{Year}}(DB_1) \cup \Pi_{\mathit{Movie},\mathit{Year}}(DB_2)$$

Another global relation containing movies, their directors and reviews:

$$MovieRev(Title, Dir, Review) \leftarrow DB_1(Title, Dir, Year), DB_3(Title, Review)$$

A view defined by, first, the join of DB_1 and DB_3 over attribute $Title$, and then a projection on $Title, Dir, Review$

```
SELECT  DB1.Title, Dir, Review
FROM     DB1, DB2
WHERE   DB1.Title = DB2.Title;
```

In relational algebra:

$$MovieRev(Title, Dir, Review) := \Pi_{Title, Dir, Review}(DB_1 \bowtie_{Title} DB_2)$$

$MovieRev(Title, Dir, Review) \leftarrow DB_1(Title, Dir, Year), DB_3(Title, Review)$

defines a conjunctive query, i.e. it is expressed in terms of

- conjunctions (or joins)
- projections

An important, useful, common, and well-studied class of queries

Using Datalog for view/query definitions makes use of recursion and syntactic processing of query/view definitions easier

And plan generation will rely on syntactic transformation of view definitions and queries ...

How to pose queries to the integration system?

In a language based on the mediated, global schema

Query: *Movies shown in year 2001, with their reviews?*

$$\begin{aligned} Ans(Title, Review) \leftarrow\ & MovieYear(Title, 2001), \\ & MovieRev(Title, Dir, Review) \end{aligned}$$

Query is expressed in terms of the global schema

There is no data in the global “DB”

The data has to be obtained from the sources

What are the intended, correct, expected answers to the query?

How can they be computed?

The first question should be answered by providing the semantics of a virtual data integration system under GAV

It is the semantics that determines the correct answers

This is coming

But, how could we proceed to obtain answers following our first natural impulse?

$$\mathit{Ans}(\mathit{Title}, \mathit{Review}) \leftarrow \underline{\mathit{MovieYear}(\mathit{Title}, 2001)}, \underline{\underline{\mathit{MovieRev}(\mathit{Title}, \mathit{Dir}, \mathit{Review})}}$$

The query can be rewritten in terms of the source relations

This is simple under GAV: rule unfolding

"Unfold" each global relation into its definition in terms of the local relations

$$\mathit{Ans}'(\mathit{Title}, \mathit{Review}) \leftarrow \underline{\mathit{DB}_1(\mathit{Title}, \mathit{Dir}, 2001)}, \underline{\underline{\mathit{DB}_1(\mathit{Title}, \mathit{Dir}, 2001), \mathit{DB}_3(\mathit{Title}, \mathit{Review})}}$$

$$\mathit{Ans}'(\mathit{Title}, \mathit{Review}) \leftarrow \underline{\mathit{DB}_2(\mathit{Title}, \mathit{Dir}, 2001)}, \underline{\underline{\mathit{DB}_1(\mathit{Title}, \mathit{Dir}, 2001), \mathit{DB}_3(\mathit{Title}, \mathit{Review})}}$$

These new queries do get answers directly from the sources
Final answer is the union of two answer sets, one for each rule

Under GAV it seems to be easy to obtain (correct?) answers to global queries

On the other side, if new sources join in the system or older leave it, definitions of global relations as views have to be rewritten

Not very flexible ...

Notice: There is a redundant condition (subgoal) in the RHS of the first rule; and the second rule is completely redundant

The plan generator (or execution engine) should be able to notice this, optimizing the query, before performing redundant computations

The second (sub)query is contained in the first one, i.e. for every database instance, the answer set for the second query is a subset of the answer set for the first one

Query containment is a key notion in databases

Tests for QC are used in different areas, e.g. query optimization, query answering using views

An Example of LAV

Each table in each local data source is described as a view in terms of global relations

Somehow unnatural:

- From the conceptual point of view
- From perspective of usual databases practice

  Here, views contain data, but “base tables”don’t

But this approach has some advantages

It also makes sense:

- A designer of a virtual, mediated system defines its own schema

  The way data will be offered to users

  Invites potential contributors of data to participate

  May not know who are or will be potential participants

  Or how their data is logically structured

- The latter have to describe their local relations in terms of the global relations (that are fixed)

  And pass the descriptions to the mediator

  Independently from other sources or contributors!

Example: Global schema offered by mediated system $\mathfrak{G}$:

$Movie(Title, Year, Director, Genre), AmerDir(Director),$
$Review(Title, Review)$

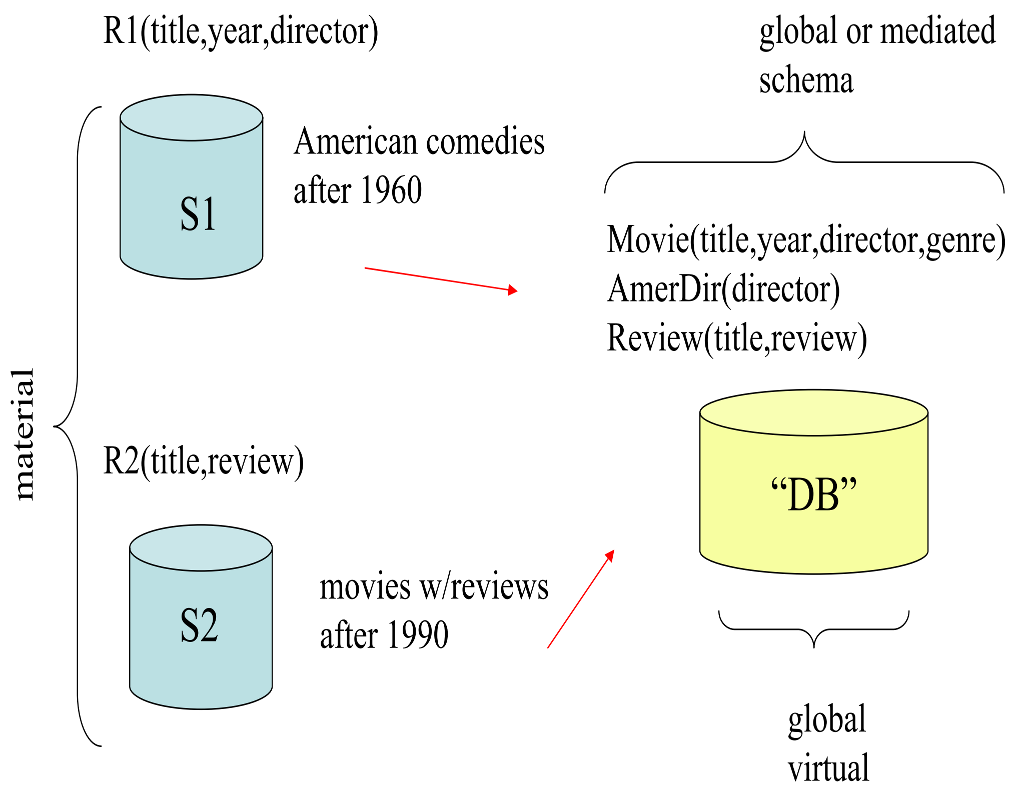
Sources S_1, S_2 are defined as views by means of conjunctive queries with built-ins:

$$\begin{aligned} S_1: \quad V_1(Title, \ Year, Director) \leftarrow & \\ & Movie(Title, Year, Director, Genre), \\ & AmerDir(Director), Genre = comedy, \\ & Year \geq 1960. \end{aligned}$$

S_1: Has a relation V_1 containing comedies, filmed after 1960, with American directors and their years

$$\begin{aligned} S_2: \quad V_2(\mathit{Title},\ \mathit{Review}) \leftarrow \; & \mathit{Movie}(\mathit{Title}, \mathit{Year}, \mathit{Director}, \mathit{Genre}), \\ & \mathit{Review}(\mathit{Title}, \mathit{Review}), \mathit{Year} \geq 1990. \end{aligned}$$

S_2: Has a relation V_2 containing movies filmed after 1990 with their reviews, but no directors



R1(title,year,director)
S1
American comedies after 1960
material
R2(title,review)
S2
movies w/reviews after 1990
global or mediated schema
Movie(title,year,director,genre)
AmerDir(director)
Review(title,review)
"DB"
global virtual

- Definition of each source does not depend on other sources!

  Sources can easily leave the system or join in

  Other sources' definitions are not affected

- From the perspective of S_2, there could be other sources contributing with information about comedies after 1990 with their reviews

  In this sense, information in S_2 could be "incomplete" wrt what $\mathfrak{G}$ "contains" (or might contain)

  So, S_2 could containing only a part of the information of the same kind in the global system

  This is the most common scenario: sources are incomplete

Query to $\mathfrak{G}$: *Comedies with their reviews produced since 1950?*

$$\begin{aligned} Ans(Title, Review) \leftarrow\ & Movie(Title, Year, Director, comedy), \\ & Review(Title, Review),\ Year \geq 1950. \end{aligned}$$

Query expressed in terms of mediated schema, as expected

Information is in the sources, now defined as views

Not possible to obtain answers by a simple, obvious or direct computation of the RHS of the query

No simple rule unfolding for the relations in the body: no definitions for them as in GAV

Plan generation to extract information from the sources is more complex than with GAV

A plan is a rewriting of the query as a set of queries to the sources and a prescription of how to combine their answers (which is needed here)

This is a query plan for our query: (we'll come back to this ...)

$$Ans'(Title, Review) \leftarrow V_1(Title, Year, Director), V_2(Title, Review)$$

Query is rewritten in terms of the views; and can be computed:

1. Extract values for $Title$ from V_1
2. Extract the tuples from V_2
3. At the mediator level, compute the join via $Title$

Due to the limited contents of the sources, we obtain comedies by American directors with their reviews filmed after 1990

This is the best or most we can get from the sources

The plan is maximally contained in the original global query

Something that can be made precise and established after defining the semantics of the system

Data Integration and Consistency

In the virtual approach to data integration, one usually assumes that certain ICs hold at the global level

However, there is no global IC maintenance mechanism

So, no guarantee that global ICs on the global schema hold

Actually, under the LAV approach they are sometimes used to generate a query plan to answer a global query

There are situations where without assuming and using those global ICs no query plans can be generated

Example: Global schema

$$Conferences(Paper, Conference)$$
$$Years(Paper, Year)$$
$$Locations(Conference, Year, Location)$$

Global functional dependencies (FDs):

$$Conferences\text{:}\quad Paper \longrightarrow Conference$$
$$Years\text{:}\quad Paper \longrightarrow Year$$
$$Locations\text{:}\quad Conference, Year \longrightarrow Location$$

Data sources as views of the global DB, i.e. LAV:

$$\begin{aligned} S_1(P, C, Y) &\longleftarrow \; Conferences(P, C),\, Years(P, Y) \\ S_2(P, L) &\longleftarrow \; Conferences(P, C),\, Years(P, Y),\, Locations(C, Y, L) \end{aligned}$$

S_1 contains papers (titles) with their conferences (names) and years

S_2 contains the papers and their locations (of presentation)

A global query about the location of PODS99

$$Ans(L) \leftarrow Locations(pods, 1999, L)$$

Answer cannot be obtained from a single data source, they need to be combined

A query plan to answer it:

$$Ans'(L) \leftarrow S_1(P, pods, 1999), S_2(P, L).$$

It prescribes:

- First use source S_1 to find some paper presented at PODS99
- Next, use source S_2 to find the location of the conference where the paper was presented

Plan is correct: every paper is presented at one conference and in one year only

Without the global FDs, there would be no way to answer this query using the given sources

But how can we be sure that such global ICs hold?

They are not maintained or checked at the global level and could be easily violated

Even when sources satisfy the natural local ICs related to the global FDs:

S_1	Paper	Conference	Year
	querying inconsistent databases	pods	1999
	workflow specifications	sigmod	1998
	...	...	...

This one looks O.K. (wrt $Paper \rightarrow Conference, Year$)

S_2	Paper	Location
	querying inconsistent databases	philadelphia
	querying inconsistent databases	schloss dagstuhl
	...	...

This one too, given that papers are presented at conferences, workshops, seminars, ... (no local IC)

Each local source seems to be consistent, but not necessarily the global system

The strategy captured by the plan does not determine a unique location for a paper presented at PODS99

We cannot be more precise at this stage about consistency issues of virtual data integration systems because we haven't presented the semantics of those systems

Since we cannot guarantee the consistency of the virtual system ...

An alternative approach: Consistent Query Answering

- Do not worry too much about the consistency of the data “contained” in the integration system (or stand alone DB)
- Better deal with the problem at query time
- When queries are posed to the integrated system, retrieve only those answers from the global database that are “consistent with” the global ICs

  What is a consistent answer to a query?
- The notion of consistency or ICs satisfaction is a holistic notion, the whole DB satisfies the ICs or not

  However, most likely most of the data in DB is still “consistent”

- When DB is queried, we want only the "consistent answers", a local notion ...
- We need a precise definition of what is a consistent answer to a query in an inconsistent DB
- We need to develop mechanisms for computing consistent answers, hopefully querying the only available, possibly inconsistent DB

Example: (informal and intuitive) Data sources

CUstudents	Number	Name
	101	john
	102	mary

OUstudents	Number	Name
	103	claire
	101	peter

Both sources satisfy the local FD: $Number \rightarrow Name$

Global relation: (defined using GAV)

$Students(x, y) \leftarrow CUstudents(x, y)$
$Students(x, y) \leftarrow OUstudents(x, y)$

The global data does not seem to satisfy the same FD, but now considered as a global IC: $Students\colon Number \longrightarrow Name$

Consistency of global system cannot be restored from the mediator

Alternative: consider the global FD when queries are answered

Obtain only the consistent answers

Which is the consistent data in an inconsistent database, in particular, query answers?

The data that is invariant under all "minimal ways" of restoring consistency

How to compute them? Different techniques

In several cases (of queries and ICs): FO rewriting of the query

Compiling ICs into the query, to enforce consistency

Make the answers respect the global ICs

Some global queries:

- $Ans(x, y) \leftarrow Students(x, y)$

  Unrestricted answers (no FD considered):

  $\{(101, john), (101, peter), (102, mary), (103, claire)\}$

  Consistent answers (FD considered):

  $\{(102, mary), (103, claire)\}$

- $Ans(x) \leftarrow Students(x, y)$

  Unrestricted answers: $\{(101), (102), (103)\}$

  Consistent answers: $\{(101), (102), (103)\}$

Bertossi, L. Consistent Query Answering in Databases. ACM Sigmod Record, June 2006, 35(2):68-76.

Bertossi, L. and Bravo, L. Consistent Query Answers in Virtual Data Integration Systems. In ‘Inconsistency Tolerance’, Springer LNCS 3300, 2004, pp. 42-83.

Chapter 3: Semantics of Virtual Data Integration Systems

Idea behind the Semantics of a VDIS

When we pose queries to a VDIS, we expect to receive answers

What answers are we talking about?

What are the correct answers?

It depends on the semantics of the system

So, what are the semantically correct answers?

Notice that there is no global instance and then no answer to a global query in the classical sense

We proceed by indicating what are the intended models of the system

More precisely, what are the intended global instances of the integration system

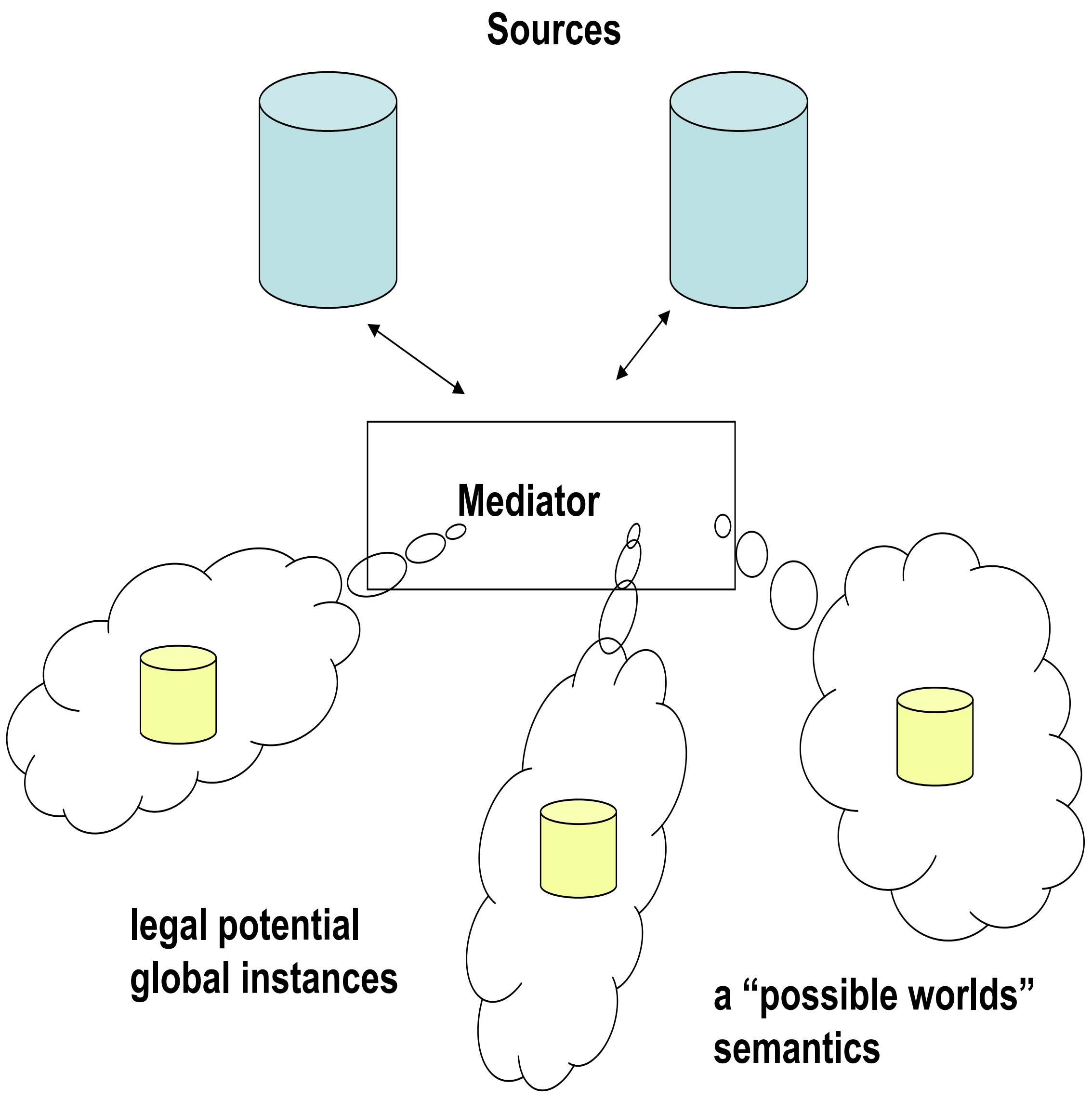
In general, we are not going to materialize those global instances

But the set of admissible, legal global instances will give a meaning to the system

Global instances (of the global schema) must satisfy the constraints and parameters of the problem, e.g.

- Source descriptions or mappings (e.g. LAV or GAV)
- Contents of the data sources
- Character of the sources, e.g. incomplete, complete, ... data
- Global ICs $\times$

We will assume there are no global integrity constraints


Sources
Mediator
legal potential global instances
a "possible worlds" semantics

Semantics under GAV

Example: Source relations:

$S_1(\mathit{Title}, \mathit{Dir}, \mathit{Year})$, $S_2(\mathit{Title}, \mathit{Dir}, \mathit{Year})$, $S_3(\mathit{Title}, \mathit{Review})$

We can see the collection of local relations as forming a single source schema $\mathcal{S}$

For each local relation $S \in \mathcal{S}$, we have an extension s

And the extensions s as forming a single source instance I

Global relations:

$$\begin{aligned}
\mathit{MovieYear}(\mathit{Title}, \mathit{Year}) &\leftarrow S_1(\mathit{Title}, \mathit{Dir}, \mathit{Year})\\
\mathit{MovieYear}(\mathit{Title}, \mathit{Year}) &\leftarrow S_2(\mathit{Title}, \mathit{Dir}, \mathit{Year})\\
\mathit{MovieRev}(\mathit{Title}, \mathit{Dir}, \mathit{Review}) &\leftarrow S_1(\mathit{Title}, \mathit{Dir}, \mathit{Year}),\\
&\qquad S_3(\mathit{Title}, \mathit{Review})
\end{aligned}$$

All this determines a global (mediated) integration system $\mathfrak{G}$, whith a global schema $\mathcal{G}$

The elements of $\mathcal{G}$ can be seen as views over $\mathcal{S}$

And each global relation $G \in \mathcal{G}$ gets an extension $G[I]$ by applying the view definition on instance I

This is as usual, propagating the data through Datalog rules in view definitions from body to head

Assume the sources are incomplete (aka. open, sound)

We also assume that the view definitions are given by positive, non-recursive Datalog rules, possibly with built-ins

A global instance D for schema $\mathcal{G}$ is legal if, for each global relation G, it holds: $G[D] \supseteq G[I]$

$G[D]$ is the extension of relation G in D

$Legal(\mathfrak{G})$ denotes the set of legal instances

Its elements are the “possible worlds”

What is true of the mediated system, at the global level, is what is true of all the possible worlds, i.e. in all the legal (global) instances

This applies in particular to query answers

If $Q(\bar{x})$ is a global query

$$\begin{aligned} Certain_{\mathfrak{G}}(Q) &= \{\bar{t} \mid \bar{t} \text{ is a usual answer to } Q \text{ in } D, \text{ for every legal instance } D\} \\ &= \bigcap_{D \in Legal(\mathfrak{G})} Q[D] \end{aligned}$$

The certain answers to the global query Q

These are the semantically correct answers

We have a model theoretic definition of correct answer?

How to compute them?

What is the connection (if any) with the intuitive method we saw before (unfolding)?

It can be easily proved (using the semantics of Datalog programs) that the certain answers coincide with those obtained as follows:

1. Take source instance I
2. Propagate its data through the view definitions
3. Obtain extensions g for each of the global relations G
4. The g's form a global instance $\bar{D}$, the so-called retrieved database

   The retrieved database is the global instance obtained by propagating the data at the sources through the rules from right to left
5. Pose Q to $\bar{D}$ as usual
6. The obtained answers are exactly the certain answers

   From this perspective of query answering, this particular (and legal) global instance gives the semantics to the integration system A generic legal instance ...

It can also be proved that the unfolding method returns the same answers

So, it provides correct query plans!

We obtain immediately that obtaining certain answers from a mediated integration system can be done in polynomial time in the size of the combined data sources:

1. Computing the retrieved database can be done in polynomial time
2. Querying the retrieved database too

Example: (cont.) Given the material sources

$S_1 = \{(aaa, peter, 1989), (abc, john, 1960), (cdde, mary, 1978)\}$
$S_2 = \{(assd, steve, 1997), (shhhh, alice, 1920)\}$
$S_3 = \{(shhhh, good), (abc, awful), (kkkk, excellent), (cdde, mediocre)\}$

The retrieved database is

$$\begin{aligned} MovieYear = \{&(aaa, 1989), (abc, 1960), (cdde, 1978), (assd, 1997),\\ &(shhhh, 1920)\}\\ MovieRev = \{&(abc, john, awful), (cdde, mary, mediocre)\}\end{aligned}$$

This an ordinary relational database

Given the global query

$$\begin{aligned} Q: \quad Ans(Dir, Year) \leftarrow\ & MovieRev(Title, Dir, Review),\\ & MovieYear(Title, Year)\end{aligned}$$

The correct answers are: $\{(john, 1960), (mary, 1978)\}$

Remark: The same semantics can be obtained in classical logical terms

Each of the view definitions

$G(\bar{x}) \leftarrow \varphi^1_{\mathcal{S}}(\bar{x}')$
$\dots$
$G(\bar{x}) \leftarrow \varphi^k_{\mathcal{S}}(\bar{x}')$ $\quad \bar{x} \subseteq \bar{x}'$

above, where G is a global relation and the formulas in the bodies are over schema $\mathcal{S}$ can be seen as a mapping

$$G(\bar{x}) \quad \mapsto \quad \Phi^G_{\mathcal{S}}(\bar{x})$$

where $\Phi_{\mathcal{S}}(\bar{x})$ is a conjunctive query (or disjunction thereof) over the source schema

It turns out that

$$\begin{aligned} Legal(\mathfrak{G}) = \{D \mid D \text{ is instance over } \mathcal{G} \text{ and,} \\ \text{for every } G \in \mathcal{G}: \\ D \cup I \ \models \ \forall \bar{x}(\Phi_{\mathcal{S}}^{G}(\bar{x}) \rightarrow G(\bar{x}))\} \end{aligned}$$

We have classical logical consequence from the theory consisting of (the logical reconstruction of) I plus D

Actually, "certain query answering" turns out to be logical consequence from a theory in FO logic: $\bar{t}$ is a certain answer to query $Q(\bar{x})$ iff

$$I \cup \{\forall \bar{x}(\Phi_{\mathcal{S}}^{G}(\bar{x}) \rightarrow G(\bar{x})) \mid G \in \mathcal{G}\} \ \models \ Q[\bar{t}]$$

(More precisely, here we should have instead of I, Ray Reiter's logical reconstruction of I as a logical theory; and also a restriction to Herbrand models)

Semantics under LAV

A virtual data integration system $\mathfrak{G}$ under LAV and open sources

$$\begin{array}{lclc} V_1(\bar{x}_1) & \leftarrow & \varphi_1(\bar{x}'_1) & \qquad v_1 \\ \dots & \dots & \dots & \\ V_n(\bar{x}_n) & \leftarrow & \varphi_n(\bar{x}'_n) & \qquad v_n \end{array}$$

Here, the V_is are the source relations, and each v_i is an extension (material data source) for relation (view) V_i, the given contents

The $\varphi_i(\bar{x}_i)$ are conjunctions of global database atoms (and possibly built-ins), i.e. conjunctive view definitions; $\bar{x}_i \subseteq \bar{x}'_i$

$\mathfrak{G}$ determines a set of legal global instances   Which ones?

Example: (views or sources on LHS, global relations on RHS)

$$\begin{aligned} \mathit{DirYears}(\mathit{Dir}, \mathit{Year}) &\leftarrow \mathit{MovieRev}(\mathit{Title}, \mathit{Dir}, \mathit{Review}),\\ &\ \mathit{MovieYear}(\mathit{Title}, \mathit{Year})\\ \mathit{Movies}(\mathit{Title}, \mathit{Dir}) &\leftarrow \mathit{MovieRev}(\mathit{Title}, \mathit{Dir}, \mathit{Review}) \end{aligned}$$

Given material extension for $\mathit{DirYears}$:

$$\{(peter, 1989), (john, 1960), (mary, 1978),\\ (steve, 1997), (alice, 1920)\}$$

Given material extension for Movies:

$$\{(aaa, peter), (abc, john), (cdde, mary),\\ (assd, steve), (shhhh, alice)\}$$

Let D be a concrete global instance

- Its underlying domain $\mathcal{U}$ contains all the constants in the sources and those appearing in the view definitions (and possibly others)
- $V_i[D]$ is the contents of the view V_i when its definition

  $$V_i(\bar{x}_i) \leftarrow \ \varphi_i(\bar{x}_i)$$

  is applied to D (the computed view on D)

Example: (cont.) Consider the global instance D_0 with

$$\begin{aligned} \mathit{MovieYear} &= \{(aaa, 1989), (abc, 1960), (cdde, 1978),\\ &\qquad (assd, 1997), (shhhh, 1920)\}\\ \mathit{MovieRev} &= \{(shhhh, alice, good), (abc, john, awful),\\ &\qquad (cdde, mary, mediocre)\}\end{aligned}$$

The view definitions evaluated on D_0 give:

- $\mathit{DirYears}[D_0] = \{(\mathit{john}, 1960), (\mathit{mary}, 1978), (\mathit{alice}, 1920)\}$
- $\mathit{Movies}[D_0] = \{(\mathit{shhhh}, \mathit{alice}), (\mathit{abc}, \mathit{john}), (\mathit{cdde}, \mathit{mary})\}$

We can see that the computed views on D_0 differ from the given material contents of the views

Actually, the computed views are both strictly contained in the original source extensions

We would expect for open sources, their material extensions to be contained in the computed views, i.e. the other way around

The chosen global instance D_0 is not one of the intended instances of the integration system ...

$$Legal(\mathfrak{G}) := \{ \text{ global } D \mid v_i \subseteqq V_i[D], \quad i = 1, \ldots, n\}$$

This is the set of intended global instances

Example: (cont.) Global instance D_0 is not legal, because

$$\{(peter, 1989), (john, 1960), (mary, 1978), (steve, 1997),$$
$$(alice, 1920)\} \not\subseteq DirYears[D_0]$$

$$\{(aaa, peter), (abc, john), (cdde, mary), (assd, steve),$$
$$(shhhh, alice)\} \not\subseteq Movies[D_0]$$

Legal instances give the semantics to the integration system

We may have several legal global instances

We can define, as before, the semantically correct answers from the integration system to a global query Q

But now, we may not have something like a single generic, representative global instance as in GAV with the retrieved DB

The certain answers to a global query are those that can be obtained from every legal global instance

$Certain_{\mathfrak{G}}(Q) := \{\bar{t} \mid \bar{t}$ is an answer to Q in D for all $D \in Legal(\mathfrak{G})\}$

Example: System $\mathfrak{G}_1$ under LAV with global relation $R(x,y)$ and open sources

$V_1(x,y) \leftarrow R(x,y)$ with $v_1 = \{(a,b),(c,d)\}$

$V_2(x,y) \leftarrow R(x,y)$ with $v_2 = \{(a,c),(d,e)\}$

Legal instance: $D_1 = \{R(a,b), R(c,d), R(a,c), R(d,e)\}$

- $v_1 \subseteq V_1[D_1] = \{(a,b),(c,d),(a,c),(d,e)\}$
- $v_2 \subseteq V_2[D_1] = \{(a,b),(c,d),(a,c),(d,e)\}$

All supersets of D_1 are also legal global instances; no subset of D_1 is legal, e.g.

$$\{R(a,b), R(c,d), R(a,c), R(d,e), R(c,e)\} \in \mathit{Legal}(\mathfrak{G}_1)$$

$$\{R(a,b), R(c,d), R(a,c)\} \notin \mathit{Legal}(\mathfrak{G}_1)$$

Consider now the global query

$$Q_1 : \quad Ans(x,y) \leftarrow R(x,y)$$

By direct inspection of the legal instances we get

$$Certain_{\mathfrak{G}_1}(Q_1) = \{(a,b),(c,d),(a,c),(d,e)\}$$

Notice $(c,e) \notin Certain_{\mathfrak{G}_1}(Q_1)$

$$Q_2 : \quad Ans(x) \leftarrow R(x,y)$$

$$Certain_{\mathfrak{G}_1}(Q_2) = \{(a),(c),(d)\}$$

We did not use any query plan to get them, only the semantics

Here, D_1 does act as a generic legal instance

Example: Global system $\mathfrak{G}_2$ with global relations P, Q, and sources

$$\begin{array}{lcll} V_1(x,y) & \leftarrow & P(x,z), Q(z,y); & v_1 = \{(a,b)\} \\ V_2(x,y) & \leftarrow & P(x,y); & v_2 = \{(a,c)\} \end{array}$$

$Legal(\mathfrak{G}_2)$?

- From definition of V_2 and the contents v_2, in any legal instance P is forced to contain (a,c)
- From definition of V_1 and the contents v_1, a legal global instance must contain at least a set of tuples of the form $\{P(a,e), Q(e,b)\}$

The legal instances of $\mathfrak{G}_2$ are all the supersets of instances of the form $\{P(a,c), P(a,z), Q(z,b) \mid z \in \mathcal{U}\}$

- $\{P(a,c), Q(c,b)\} \in Legal(\mathfrak{G}_2)$
- $\{P(a,c), P(a,e), Q(e,b)\} \in Legal(\mathfrak{G}_2)$
- $\{P(a,c), Q(c,b), P(e,e), Q(e,a), Q(d,d), Q(a,c)\} \in Legal(\mathfrak{G}_2)$
- $\{P(a,c), Q(e,b)\} \notin Legal(\mathfrak{G}_2)$

Here we have legal instances that are incomparable under set inclusion!

Global query: $Q_1: \ Ans(x,y) \leftarrow P(x,y)$

$$Certain_{\mathfrak{G}_2}(Q_1) = \{(a,c)\}$$

Global query: $Q_2: \ Ans(x) \leftarrow Q(x,y)$

$$Certain_{\mathfrak{G}_2}(Q_2) = \{\}$$

Global query: $Q_3: \ Ans(y) \leftarrow Q(x,y)$

$$Certain_{\mathfrak{G}_2}(Q_3) = \{(b)\}$$

Global query: $Q_4\colon\ Ans(x,y) \leftarrow P(x,y), not\ Q(x,y)$

Not a conjunctive query

In legal instances (under open sources) we can always add new tuples, so $not\ Q(x,y)$ can always be made false in some legal global instance: $Certain_{\mathfrak{G}_2}(Q_4) = \emptyset$

Notion of certain answer does not seem to be a sensible notion for non-monotone queries

Conjunctive queries (and others) are monotone: the set of answers may only grow if the database grows: for databases D_1, D_2

$D_1 \subseteq D_2 \implies$ Answers to Q in $D_1 \subseteq$ Answers to Q in D_2 (i.e. $Q[D_1] \subseteq Q[D_2]$)

General mechanisms for query planning available are for classes of monotone queries

More on Openess, etc.

The definition of legal instance we gave captures the fact that sources are open (aka. incomplete or sound) sources

There are also the notions (labels) of “closed” (complete) and “closed and open” (clopen, exact)

The notion of legal instance under LAV can be easily adapted to capture these other possible labels

Before, more intuition on these concepts

Example: A LAV integration system that integrates data sources about teams participating in the soccer world cup 2002

Global relation $Team(Country, Group)$

A first source of information contains the countries whose matches in the first round are shown on TV; defined by

$$ShownOnTV(x) \leftarrow Team(x, y)$$

It contains only a subset of the expected entries for relation $Team$:

$$ShownOnTV \subseteqq \Pi_{Country}(Team)$$

This source is open (or incomplete or sound)

A second source $Qual$ contains all the countries participating in the qualifying matches (before WC02), e.g. Germany, Canada, ...

$$Qual(x) \leftarrow Team(x, y)$$

Canada did not participate in the World Cup, but did participate in the qualifying matches, then

$$Qual \nsubseteq \Pi_{Country}(Team)$$

The source is not open, rather

$$Qual \supseteqq \Pi_{Country}(Team)$$

We say the source is closed (or complete)

A third source $First$ contains the countries participating in the first round: $First(x) \leftarrow Team(x, y)$

Now $First = \Pi_{Country}(Team)$, and the source is open and closed (or clopen or exact)

More generally, in an integration system, different sources can have different labels

E.g. a LAV system $\mathfrak{G}$ can be something like

$$\begin{array}{lcllll} V_1(\bar{x}_1) & \leftarrow & \varphi_1(\bar{x}_1'); & v_1 = \{\ldots\}; & \text{open} \\ V_2(\bar{x}_2) & \leftarrow & \varphi_2(\bar{x}_2'); & v_2 = \{\ldots\}; & \text{open} \\ V_3(\bar{x}_3) & \leftarrow & \varphi_3(\bar{x}_3'); & v_3 = \{\ldots\}; & \text{closed} \\ V_4(\bar{x}_4) & \leftarrow & \varphi_4(\bar{x}_4'); & v_4 = \{\ldots\}; & \text{clopen} \end{array}$$

The notion of legal instance has to respect the labels:

$$\mathit{Legal}(\mathfrak{G}) := \{ \text{ global } D \mid \begin{array}{l} v_1 \subseteq V_1[D], \\ v_2 \subseteq V_2[D], \\ v_3 \supseteq V_3[D], \\ v_4 = V_4[D]\} \end{array}$$

It is easier to specify the conditions (assumptions) on the source under LAV than under GAV, because the sources are directly specified in the system

Reference:

G. Grahne, A. Mendelzon. Tableaux Techniques for Querying Information Sources through Global Schemas, ICDT 99

Some Remarks on GLAV

Remember that in this case we have mappings between local views and global views

For motivation, remember that so far the mappings have been as follows (we express mappings in FOL, not Datalog, for uniformity):

1. GAV, open sources; G global relation

   $G(\bar{x}) \quad \mapsto \quad \Phi_{\mathcal{S}}^{G}(\bar{x})$ (formula on local schema $\mathcal{S}$)

   Legal D: $D \cup I \quad \models \quad \forall \bar{x}(\Phi_{\mathcal{S}}^{G}(\bar{x}) \rightarrow G(\bar{x}))$

   Or $\Phi_{\mathcal{S}}^{G}[I] \subseteq G[D]$

2. LAV, open sources; S local relation

   $S(\bar{x}) \quad \mapsto \quad \Phi^S_{\mathcal{G}}(\bar{x})$ (formula on global schema $\mathcal{G}$)

   Legal D: $D \cup I \ \models \ \forall \bar{x}(S(\bar{x}) \rightarrow \Phi^S_{\mathcal{G}}(\bar{x}))$

   Or $S[I] \subseteq \Phi^S_{\mathcal{G}}[D]$

Under GLAV, open sources:

view over local schema $\mathcal{S} \ \mapsto$ view over global schema $\mathcal{G}$

That is: $\Phi_{\mathcal{S}}(\bar{x}) \ \mapsto \ \Phi_{\mathcal{G}}(\bar{x})$ ($\bar{x}$ are the free variables)

Legal D: $D \cup I \ \models \ \forall \bar{x}(\Phi_{\mathcal{S}}(\bar{x}) \rightarrow \Phi_{\mathcal{G}}(\bar{x}))$ (i.e. $\Phi_{\mathcal{S}}[I] \subseteq \Phi_{\mathcal{G}}[D]$)

In general, the formulas $\Phi_{\mathcal{S}}(\bar{x})$ and $\Phi_{\mathcal{G}}(\bar{x})$ are conjunctive; so LAV is a special case

$\Phi_{\mathcal{S}}(\bar{x})$ could also be a the query predicate $Ans(\bar{x})$ for a Datalog program over the local schema $\mathcal{S}$

Example: Source relations: $DirYears(Dir, Year)$,
$Movies(Title, Dir)$

Global relations: $MovieRev(Title, Dir, Review)$,
$MovieYear(Title, Year)$

Before (LAV):

$DirYears(Dir, Year) \leftarrow \quad MovieRev(Title, Dir, Review), MovieYear(Title, Year)$

$Movies(Title, Dir) \leftarrow \quad MovieRev(Title, Dir, Review)$

Now, we could have a view over local schema:

$$Directors(Dir) \leftarrow DirYears(Dir, Year), Movies(Title, Dir)$$

And the mapping: (*)

$Directors(Dir) \mapsto \exists Title, Review \;\; MovieRev(Title, Dir, Review)$

Equivalently: (view over global schema at the RHS)

$$\exists \mathit{Year}, \mathit{Title}\ (\mathit{DirYears}(\mathit{Dir}, \mathit{Year}) \wedge \mathit{Movies}(\mathit{Title}, \mathit{Dir})) \mapsto$$

$$\exists \mathit{Title}, \mathit{Review}\ \mathit{MovieRev}(\mathit{Title}, \mathit{Dir}, \mathit{Review})$$

I.e. $\Phi_{\mathcal{S}}(\mathit{Dir}) \mapsto \Phi_{\mathcal{G}}(\mathit{Dir})$

We do not have to define every source attribute in (*)

This may me more natural

In the following, until further notice, open sources only ...

The question now is whether a query plan to answer a global query is capable of retrieving all the certain answers

Now query plans can be assessed against a precise semantics

Chapter 4: Query Answering

Query Plans

We have seen that generation of query plans is simple under GAV

Also that the naive algorithm for query answering is correct: it obtains all and only certain answers to global queries

We will concentrate on the LAV approach with open sources

Given a global query Q posed in terms of the global schema, we need to go to the sources to find the local data to return global answers to the user

How?? We need a plan for evaluating Q

Query Q has to be "rewritten" in terms of the views, i.e. as a set of queries expressed in terms of the relations in the sources

Some rewriting algorithms:

- Bucket [Halevy et al.]
- Inverse Rules [Duschka, Genesereth, Halevy]
- MiniCon [Pottinger, Halevy]
- Deductive [Grant, Minker]
- ...

They are designed to handle conjunctive global queries (and minor extensions)

We will concentrate on the “Inverse Rules Algorithm” (IRA) due to its conceptual interest, and to show some general issues

Inverse Rules Algorithm

Given: Set of rules describing the source relations as (non recursive Datalog) views of the global schema

Input: A global query expressed in Datalog (may be recursive, but no negation)

Output: A new Datalog program expressed in terms of the source relations

Source relations described by conjunctive rules (queries):

$$S(\bar{x}) \leftarrow P_1(\bar{x_1}), \ldots, P_n(\bar{x_n})$$

S a source relation; P_is global relations, no built-ins

Example: V_1, V_2: local relations

$R_1, R_2, R_3, R4, R_5, R_6, R_7$: global relations

Source descriptions $\mathcal{V}$:

S_1: $V_1(x, z) \leftarrow R_1(x, y), R_2(y, z)$

S_2: $V_2(x, y) \leftarrow R_3(x, y)$

IRA obtains from these descriptions, "inverse rules" that describe the global relations

Idea: V_2 is incomplete, i.e. its contents is a subset of the "contents" of the (legal) global relation R_3

That is, V_2 "$\subseteq$" R_3, i.e. V_2 "$\Rightarrow$" R_3

More precisely, we invert the rule in the description of V_2:

$$R_3(x, y) \leftarrow V_2(x, y)$$

A rule describing R_3!!

> We could obtain other rules describing R_3, e.g. if we had a third source description $V_3(x, y) \leftarrow R_3(x, y)$, we would get the inverse rule $R_3(x, y) \leftarrow V_3(x, y)$; and we take the union
>
> Notice that both V_2, V_3 are incomplete, they have only part of the data in R_3

What about inverting the rule for V_1:

$$R_1(x,y), R_2(y,z) \leftarrow V_1(x,z) \quad ???$$

What kind of rule (head) is this? Maybe the conjunction ...

$R_1(x,y) \leftarrow V_1(x,z)$

$R_2(y,z) \leftarrow V_1(x,z)$

We lost the shared variable y (the join), the two occurrences of y are independent now

Furthermore: what does y in the head mean??

y does not appear in the bodies, so no condition on y ...

Any value for y? Not the idea ...

Those rules are not safe

Better:

$V_1(x,z) \leftarrow R_1(x,y), R_2(y,z)$ is logically equivalent to

$V_1(x,z) \leftarrow \exists y\ (R_1(x,y)\ \wedge\ R_2(y,z))$

Inverting the rule, we obtain:

$$\exists y (R_1(x,y)\ \wedge\ R_2(y,z)) \leftarrow V_1(x,z)$$

There is an implicit universal quantification on x, z

$$\forall x \forall z (\exists y (R_1(x,y)\ \wedge\ R_2(y,z)) \leftarrow V_1(x,z))$$

Each value for y possibly depends on the values for x, z, i.e. y is a function of x, z

To keep this dependence, replace y by a "function symbol" $f(x,z)$

f is a so-called "Skolem" function

We may need more of them, to capture other dependencies between variables

One for each rule and existential variable (projection) appearing in it

$(R_1(x, f(x,z)) \ \wedge \ R_2(f(x,z), z)) \leftarrow V_1(x,z)$

and then

$$R_1(x, f(x,z)) \leftarrow V_1(x,z)$$

$$R_2(f(x,z), z)) \leftarrow V_1(x,z)$$

Finally, we have obtained the following inverse rules $\mathcal{V}^{-1}$:

$$\begin{array}{rcl} R_1(x, f(x,z)) & \leftarrow & V_1(x,z) \\ R_2(f(x,z), z) & \leftarrow & V_1(x,z) \\ R_3(x,y) & \leftarrow & V_2(x,y) \end{array}$$

The global relations are described as views of the local relations!

Notice: not exactly a Datalog program, it contains functions ...

In order to answer global queries we can append (always) the (same) inverse rules to any Datalog global query

We can use the inverse rules to compute global queries ...

For example, the query Q

$$\begin{array}{rcl} Ans(x,z) & \leftarrow & R_1(x,y), R_2(y,z), R_4(x) \\ R_4(x) & \leftarrow & R_3(x,y) \\ R_4(x) & \leftarrow & R_7(x) \\ R_7(x) & \leftarrow & R_1(x,y), R_6(x,y) \end{array}$$

A Datalog query in terms of the "base" global relations R_1, R_2, R_3

We have descriptions for those three global relations; the inverse rules ...

The goal R_6 cannot be computed, there is no description for it, in particular, it does not appear in any source description, its contents is empty

Then, R_7 cannot be evaluated either (also empty contents); delete that rule; we are left with

$$\begin{array}{rcl} Ans(x,z) & \leftarrow & R_1(x,y), R_2(y,z), R_4(x) \\ R_4(x) & \leftarrow & R_3(x,y) \\ R_4(x) & \leftarrow & R_7(x) \end{array}$$

For the same reason, the second rule for R_4 cannot be evaluated; delete it

$$\begin{array}{rcl} Ans(x,z) & \leftarrow & R_1(x,y), R_2(y,z), R_4(x) \\ R_4(x) & \leftarrow & R_3(x,y) \end{array}$$

We obtain a pruned query Q^-

Then the plan $Plan(Q)$ returned by the IRA is $Q^- \cup \mathcal{V}^{-1}$:

$$\begin{array}{rcl} Ans(x,z) & \leftarrow & R_1(x,y), R_2(y,z), R_4(x) \\ R_4(x) & \leftarrow & R_3(x,y) \\ R_1(x, f(x,z)) & \leftarrow & V_1(x,z) \\ R_2(f(x,z),z) & \leftarrow & V_1(x,z) \\ R_3(x,y) & \leftarrow & V_2(x,y) \end{array}$$

A "Datalog" program with functions ...

This is "the best we have for answering the original query"

We will get answers to Q, and "the most" that could be computed

These claims require a precise formulation and proof ...

This Datalog query can be evaluated, e.g. bottom-up, from concrete source contents, e.g. assume that the source relations are:

$V_1 = \{(a,b),(a,a),(c,a),(b,a)\}$

$V_2 = \{(a,c),(a,a),(c,d),(b,b)\}$

$$\begin{array}{rcl} Ans(x,z) & \leftarrow & R_1(x,y), R_2(y,z), R_4(x) \qquad (1) \\ R_4(x) & \leftarrow & R_3(x,y) \qquad (2) \\ R_1(x, f_1(x,z)) & \leftarrow & V_1(x,z) \qquad (3) \\ R_2(f_1(x,z), z) & \leftarrow & V_1(x,z) \qquad (4) \\ R_3(x,y) & \leftarrow & V_2(x,y) \qquad (5) \end{array}$$

- Using rules (5), (3), (4) we get

  $R_3 = \{(a,c),(a,a),(c,d),(b,b)\}$

  $R_1 = \{(a,f(a,b)),(a,f(a,a)),(c,f(c,a)),(b,f(b,a))\}$

  $R_2 = \{(f(a,b),b),(f(a,a),a),(f(c,a),a),(f(b,a),a)\}$

- Now rule (2)

  $R_4 = \{(a), (c), (b)\}$

- Finally, rule (1)

  $\Pi_{x,z}(R_1 \bowtie R_2) = \{(a,b), (a,a), (c,a), (b,a)\}$

  We keep only those tuples in this relation such that the first argument appears in R_4; all of them in this case

  Finally, $Ans = \{(a,b), (a,a), (c,a), (b,a)\}$

Remarks:

- We process the functions symbolically

  They did not appear in the final answer set

  This may not be always the case

  If tuples with function symbols appear in the final answer set, we filter them out ...

- It may be necessary to introduce more than one function symbol

- We may get several rules describing the same global relation, in this case, that relation is described by a disjunctive query (view)

We illustrate all these issues with an example

Example: Source descriptions $\mathcal{V}$

$$\begin{aligned} V_1(x) &\leftarrow R(x,y), G(y,z) \\ V_2(x,y) &\leftarrow R(z,x), U(x,y) \end{aligned}$$

Inverse rules:

$$\begin{aligned} R(x, f_1(x)) &\leftarrow V_1(x) \\ G(f_1(x), f_2(x)) &\leftarrow V_1(x) \\ R(f_3(x,y), x) &\leftarrow V_2(x,y) \\ U(x,y) &\leftarrow V_2(x,y) \end{aligned}$$

Source contents:

$$\begin{aligned} V_1 &= \{(a), (d)\} \\ V_2 &= \{(a,c), (c,d), (b,a)\} \end{aligned}$$

Global query: $Ans(x) \leftarrow R(x,y)$

$$R = \{(a, f_1(a)), (d, f_1(d))\} \cup$$
$$\{(f_3(a,c), a), (f_3(c,d), c), (f_3(b,a), b)\}$$

$$\Pi_x(R) = \{(a), (d)\} \cup \{(f_3(a,c)), (f_3(c,d)), (f_3(b,a))\}$$

$$Ans = \{(a), (d)\}$$

The definition of R by the two rules

$$\begin{array}{rcl} R(x, f_1(x)) & \leftarrow & V_1(x) \\ R(f_3(x,y), x) & \leftarrow & V_2(x,y) \end{array}$$

reflects the fact that each of V_1, V_2 contains only part of the data of its kind, i.e. that the source relations are incomplete wrt to the same kind of data in the system

Several questions:

- Are the answers obtained with IRA any good?

  In what sense?

- Is there a better plan?

  Or better mechanisms for generating plans?

  Better in what sense?

- What is the most information we can get from such a system?

- What is the "data contained in the system", i.e. in the global relations if they were to be materialized?

  There must be some sort of data, otherwise how can we be obtaining answers to global queries?

All these questions have to do with the semantics of a virtual data integration system we introduced before

Some of them have been answered already, and for answering the others we also have the basic semantic framework

Properties of the IRA

The query plan obtained may not be exactly a Datalog program, due to the auxiliary function symbols

The same inverse rules can be used with any global query; we compute them once

The query plan can be evaluated in a bottom-up manner and always has a unique fix point

The query plan can be constructed in polynomial time in the size of the original query and the source descriptions

The resulting plan can be evaluated in polynomial time in the size of the underlying data sources (polynomial time in data complexity)

The plan obtained is the best we can get under the circumstances, i.e. given the query, the sources, and their descriptions

More precisely, the query plan is maximally contained in the original query Q

There is no (other) query plan that retrieves a proper superset of certain answers to Q from the integration system

We'll make this precise ...

Maximal Containment

Consider a query plan P (like the Datalog program above) for answering the global query Q using the views in $\mathcal{V}$

How can we compare the original query, that is expressed in terms of global predicates, with the query plan, that contains view predicates?

Let's compare them in the language of the mediated schema

The expansion P^{exp} of P is obtained from P by replacing the view predicates in P by their definitions

(possibly using fresh variables for existential variables in the views)

In this way we obtain a new program with global predicates only

Example: (continued) To answer query Q we had the plan

$$\begin{array}{rcl}
Ans(x,z) & \leftarrow & R_1(x,y), R_2(y,z), R_4(x) \\
R_4(x) & \leftarrow & R_3(x,y) \\
R_1(x, f(x,z)) & \leftarrow & V_1(x,z) \\
R_2(f(x,z), z) & \leftarrow & V_1(x,z) \\
R_3(x,y) & \leftarrow & V_2(x,y)
\end{array}$$

The query Q did not contain the views V_1, V_2

We can eliminate the views using their descriptions

We obtain a plan in terms of the global relations

$$\begin{array}{rcl}
Ans'(x,z) & \leftarrow & R_1(x,y), R_2(y,z), R_4(x) \\
R_4(x) & \leftarrow & R_3(x,y) \\
R_1(x, f_1(x,z)) & \leftarrow & R_1(x,y), R_2(y,z) \\
R_2(f_1(x,z), z) & \leftarrow & R_1(x,y), R_2(y,z) \\
R_3(x,y) & \leftarrow & R_3(x,y)
\end{array}$$

This one is expressed in terms of the global relations R_1, R_2, R_3 only (and views defined on them)

It can be compared with the original query Q

By definition, the query plan P using views is maximally contained in a query Q if

1. $P^{exp} \subseteq Q$

   I.e. for every global database instance D, $P^{exp}[D] \subseteq Q[D]$

2. for every query plan P' such that $(P')^{exp} \subseteq Q$, it holds $P' \subseteq P$

Here, $P^{exp} \subseteq Q$ (or $P' \subseteq P$) is the usual and central notion of query containment

That is, $P^{exp} \subseteq Q$ means that the extension of the query predicate in P^{exp} is included in the extension of the query predicate in Q for every database instance over the global schema

We are using the classical and important notion of query containment

E.g. for the queries

$Q_1: \quad Ans(x,y) \leftarrow R(x,y)$ and

$Q_2: \quad Ans(x,y) \leftarrow R(x,y), S(x,y)$,

it holds $Q_2 \subseteq Q_1$

because for every possible contents for the relations R, S, the answers to Q_2 are contained among the answers to Q_1

So, we want P^{exp} to be maximally contained in Q

What about the functions introduced by the former?

Denote by $P\downarrow$ the "pruned" plan P that computes the answer set for P, but with all the tuples containing Skolem functions deleted at the end

Theorem: For every global Datalog program Q and every set of conjunctive source descriptions $\mathcal{V}$, the query plan $(Q^{-}, \mathcal{V}^{-1})\downarrow$ is maximally contained in Q

Even more, as a consequence, under LAV with open sources, conjunctive view definitions, and conjunctive queries, IRA returns all and only the certain answers to conjunctive queries

I.e. under these conditions, IRA is a sound and complete mechanism for query answering

Complexity?

Under the same conditions as above, for conjunctive queries Q without built-ins, the data complexity of deciding is a tuple is a certain answer is polynomial

That is, for every fixed set of view definitions $\mathcal{V}$ and fixed $Q(\bar{x})$, the problem:

$$\{\ (I, \bar{t}) \mid \bar{t} \in \mathit{Certain}_{\mathfrak{G}}(Q)\}$$

can be decided in polynomial time in the size of I

Here:

- I is the collection of data source instances
- $\mathfrak{G}$ is the VDIS formed by $\mathcal{V}$ and I

How?

Use the IRA for computing the certain answers to Q and check

With slightly more expressive queries, e.g. conjunctive with built-in $\neq$, the problem becomes $coNP$-complete
(even with purely conjunctive view definitions)

Under GAV it is still polynomial: compute the retrieved database and pose an ordinary query ...

Some references:

Serge Abiteboul, Oliver M. Duschka: Complexity of Answering Queries Using Materialized Views. PODS 1998: 254-263

Oliver M. Duschka, Michael R. Genesereth, Alon Y. Levy: Recursive Query Plans for Data Integration. J. Log. Program. 43(1): 49-73 (2000)

Maurizio Lenzerini: Data Integration: A Theoretical Perspective. PODS 2002: 233-246

The Need for Recursion

Given a conjunctive or Datalog query Q, IRA will produce a query plan that is a Datalog program with functions

If Q is conjunctive or non-recursive Datalog, IRA produces a non-recursive Datalog query plan

If Q is a recursive Datalog query, IRA produces a recursive Datalog query plan

So, for conjunctive queries we do not need recursion?

In some cases yes; and we need to modify the plan if the query plan is to be maximally contained in the original query

At least this is the case when there are some restrictions on the query patterns or query bindings

For example, when the data sources may be accessed only with particular patterns or bindings

Example: (of query bindings) We may have a table of employees

$$\mathit{Emp}(\mathit{SecretCode}, \mathit{Name}, \mathit{Salary})$$

For privacy reasons, we are not allowed to pose queries of the form

$$\mathit{Ans}(y, z) \leftarrow \mathit{Emp}(x, y, z)$$

But a query like this could be allowed

$$\mathit{Ans}(z) \leftarrow \mathit{Emp}(\mathit{xi56rf}, \mathit{john}, z)$$

I.e. the first two arguments of Emp must have bindings (or have to be bound), and only the last one can be free

These restrictions can be expressed using binding patterns in query atoms

For example, for table $Grades^{bf}(Student, Grade)$ (b indicates that the variable has to be "bound", f that the variable can be "free")

Meaning that

- Not allowed to ask "give me all the students with their grades", i.e. $Grades(x, y)$?
- Allowed to ask for grades of specific students, e.g. "give me John's grades", i.e. $Grades(john, y)$?

Typically web pages can be queried with fixed patters, e.g. a (key)word to be typed in a slot window

Our data sources could have pattern restrictions on queries that can receive

Those restrictions have to be taken into account by the query planner ...

Example: Three global relations

- $AAAIpapers(x)$ "contains" papers presented at the AAAI conference
- $Cites(x, y)$ contains papers citing other papers; papers presented anywhere
- $AwardPaper(x)$ contains award winning papers (presented anywhere)

Three open data sources

- One containing papers presented at the AAAI conference, no access restrictions

$$AaaiDB^{f}(x) \leftarrow AAAIpapers(x)$$

- Another containing papers and their citations, but can be accessed providing the title of the citing paper

$$CitationDB^{bf}(x, y) \leftarrow Cites(x, y)$$

- Finally, one containing award winning papers, that can be accessed to check specific papers

$$AwardDB^{b}(x) \leftarrow AwardPaper(x)$$

Global query: "Give me all award winning papers", i.e.

$$Q: \quad Ans(x) \leftarrow AwardPaper(x)$$

$AwardPaper$ cannot be used directly, because no data there

Going to $AwardDB$ that allows only ground queries does not help much ...

Need to connect with to $AAAIpapers$ and also to $Cites$ if we want to find as many papers as possible

A query plan Q':

$$Ans'(x) \leftarrow AaaiDB(x), AwardDB(x)$$

$$\begin{aligned} Ans'(x) \leftarrow\ & AaaiDB(V), CitationDB(V, x_1), \ldots, CitationDB(x_m, x), \\ & AwardDB(x) \end{aligned}$$

No plan that fixes the length m of the chain of citations will be maximally contained in the original query

Instead, the following recursive Datalog query plan Q'' will do …

$$\begin{array}{rcl} papers(x) & \leftarrow & AaaiDB(x) \\ papers(x) & \leftarrow & papers(y), CitationDB(y, x) \\ Ans(x) & \leftarrow & papers(x), AwardDB(x) \end{array}$$

(Collect all possible papers and check if they are award winners)

Reference:

O. Duschka, M. Genesereth, A. Levy. Recursive Query Plans for Data Integration. J. Logic Programming, 2000

Chapter 5: Related Subjects

Query Answering Using Views

In LAV, GAV, and GLAV, generation of query plans becomes an instance of a more general and traditional problem: query answering using views, in particular, query rewriting using views

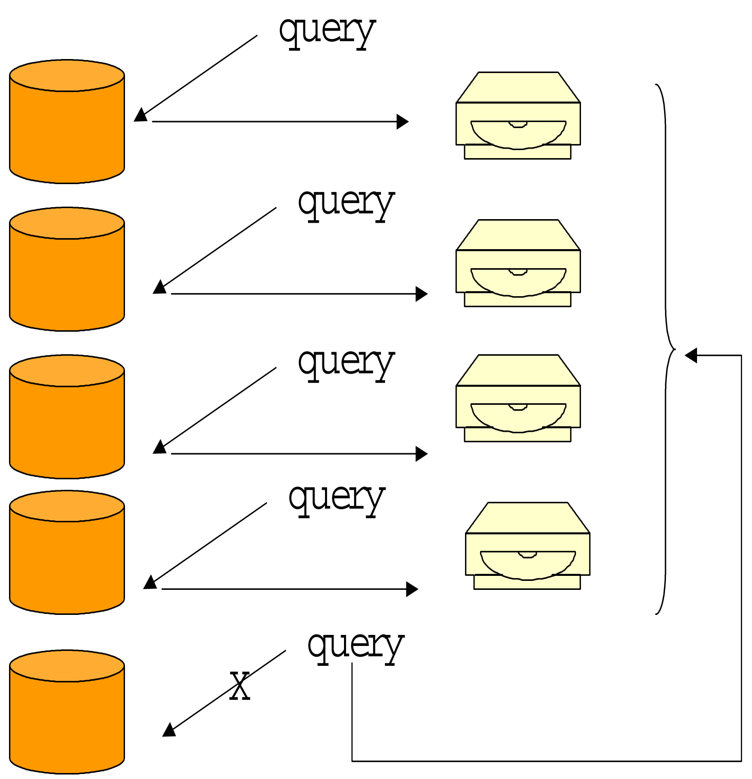
- Given is a collection of views (queries) $V_1, \ldots, V_n$

  Whose contents (answers) have already been computed

- New query Q arrives

  Instead of computing its answers directly, try to use the answers to (contents of) $V_1, \ldots, V_n$

Every query can be seen as a view, but usually they are not defined as such in the DBMS, because they will not be of any interest later on



Instead, views will be used along the session or across sessions

They are usually kept virtual, and are recomputed when needed (if updates on base tables are executed and whole recomputation of views is easy)

However, it may be useful to materialize the contents of views; the same with answers to queries, so the view contents and query answers are cached

We do this if computing those answers has been expensive and the information obtained has been detected as potentially useful by being related to answers to future queries

Query answering using view has obvious applications in VDI, but also in query optimization (stand alone databases), dataware-houes (that can be conceived as collections of materialized views), etc.

When a new query arrives, one could try to take advantage of those precomputed, cached results ... How?

An obvious problem consists in characterizing and determining how much of the real answer we get by using the precomputed views only? What is the maximum we can get?

Query containment is again a key notion/technique in this context

References:

A. Levy, A. Mendelzon, D. Srivastava and Y. Sagiv. Answering queries using views. PODS 95

A. Halevy. Answering Queries using Views: a Survey. VLDB Journal, 2006

Stay in touch ...

www.scs.carleton.ca/~bertossi

bertossi@scs.carleton.ca