Consistent Answers from Integrated Data Sources

Leopoldo Bertossi

Carleton University
School of Computer Science
bertossi@scs.carleton.ca
www.scs.carleton.ca/~bertossi

Join work with:

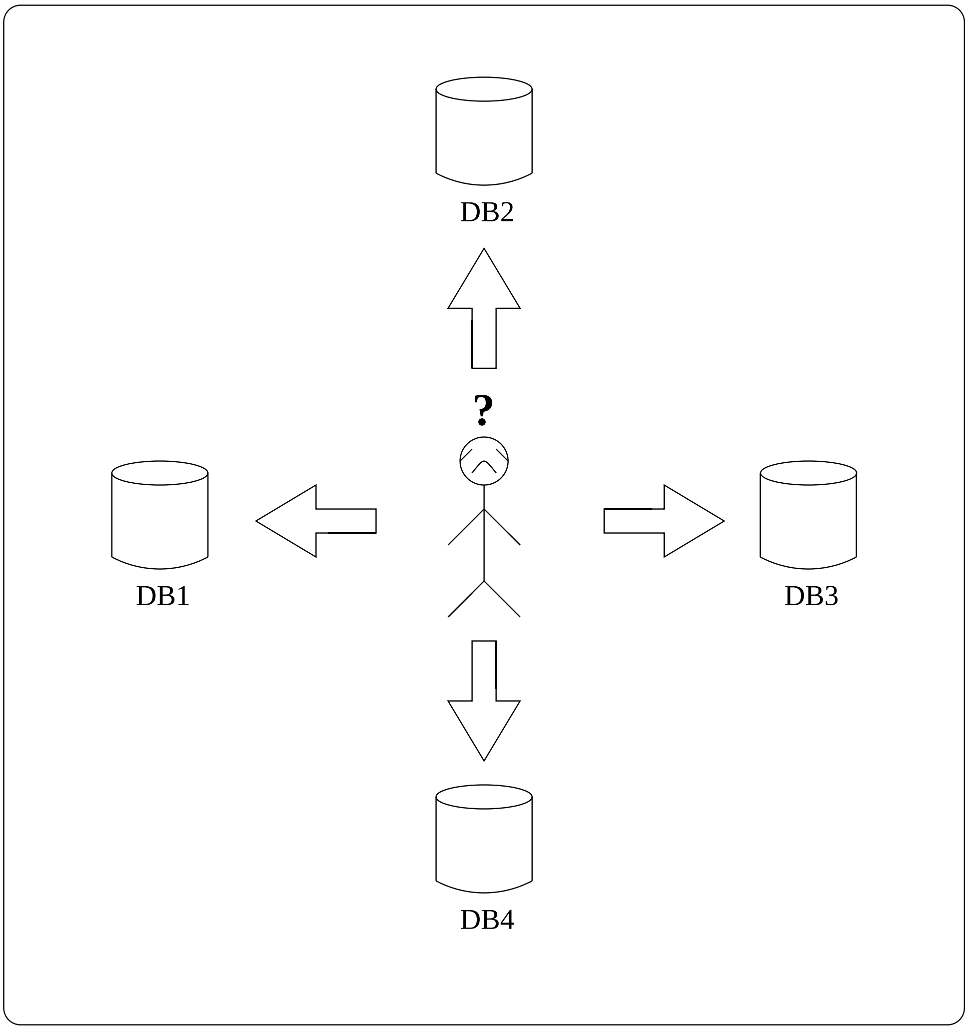
Jan Chomicki (University at Buffalo)

Alvaro Cortes (Catholic University of Chile)

Claudio Gutierrez (University of Chile)

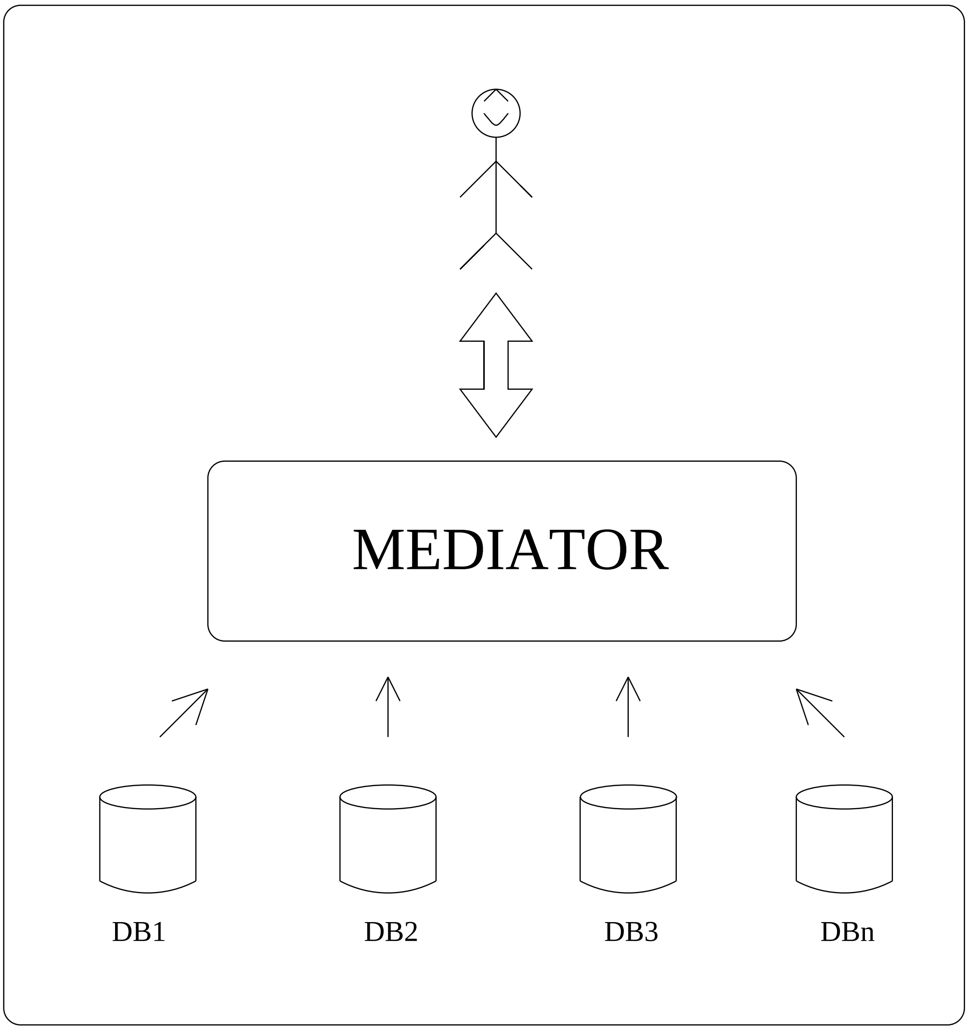
Virtual Integration of Data Sources

We need to integrate an increasing number of different data sources



One approach consists in integrating the sources by means of a "mediator"

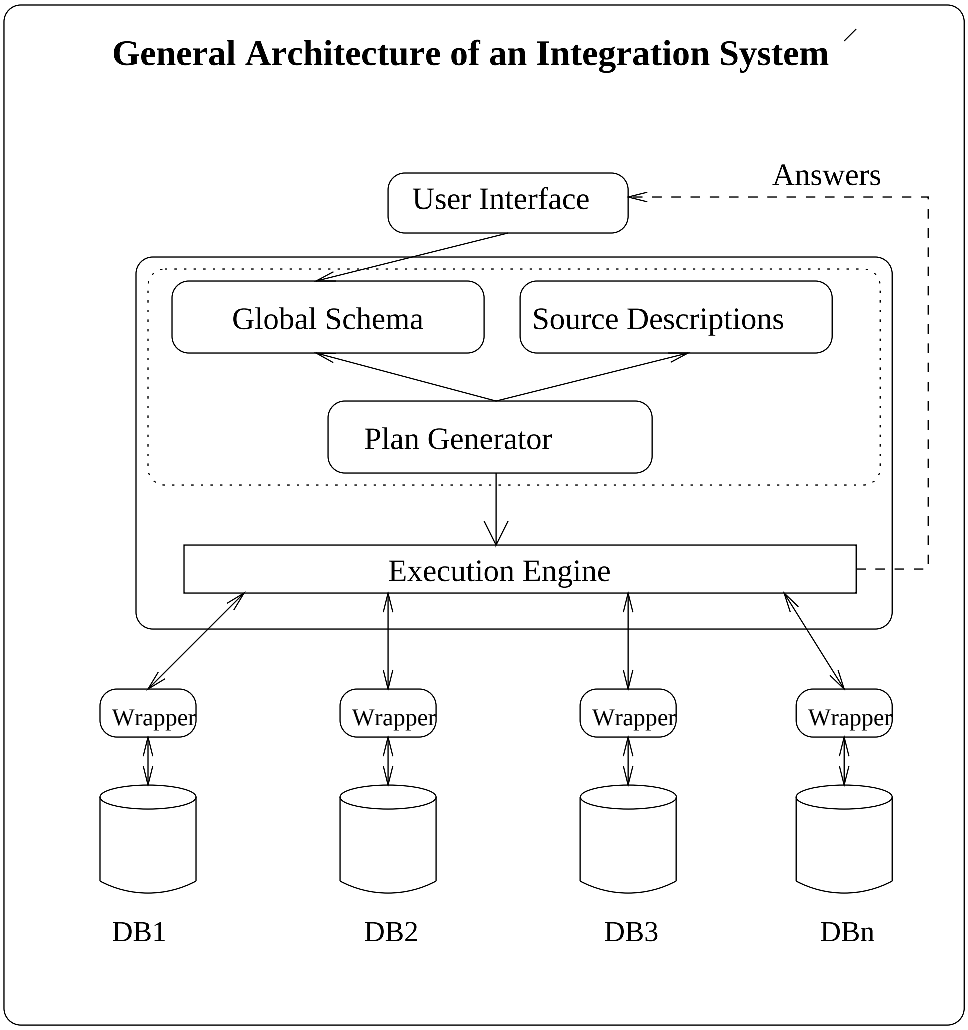
Interaction with a mediated integrated information system



Software system for an architecture that offers a common interface to a set of possibly heterogeneous information sources

- Data is kept in the sources
- Data sources are independent and do not cooperate with each other
- The set and number of participating sources may be flexible and open

Main components of a mediation based integration system



Global or Mediated Schema:

- Set of names for relations (tables) and their attributes
- 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 data sources
- The "DB" corresponding to the global schema is virtual
- User poses queries in terms of the relations in the global schema

- The query planner should output a new query, expressed in terms of the tables in the sources

- Subqueries have to be send to the sources and their results combined

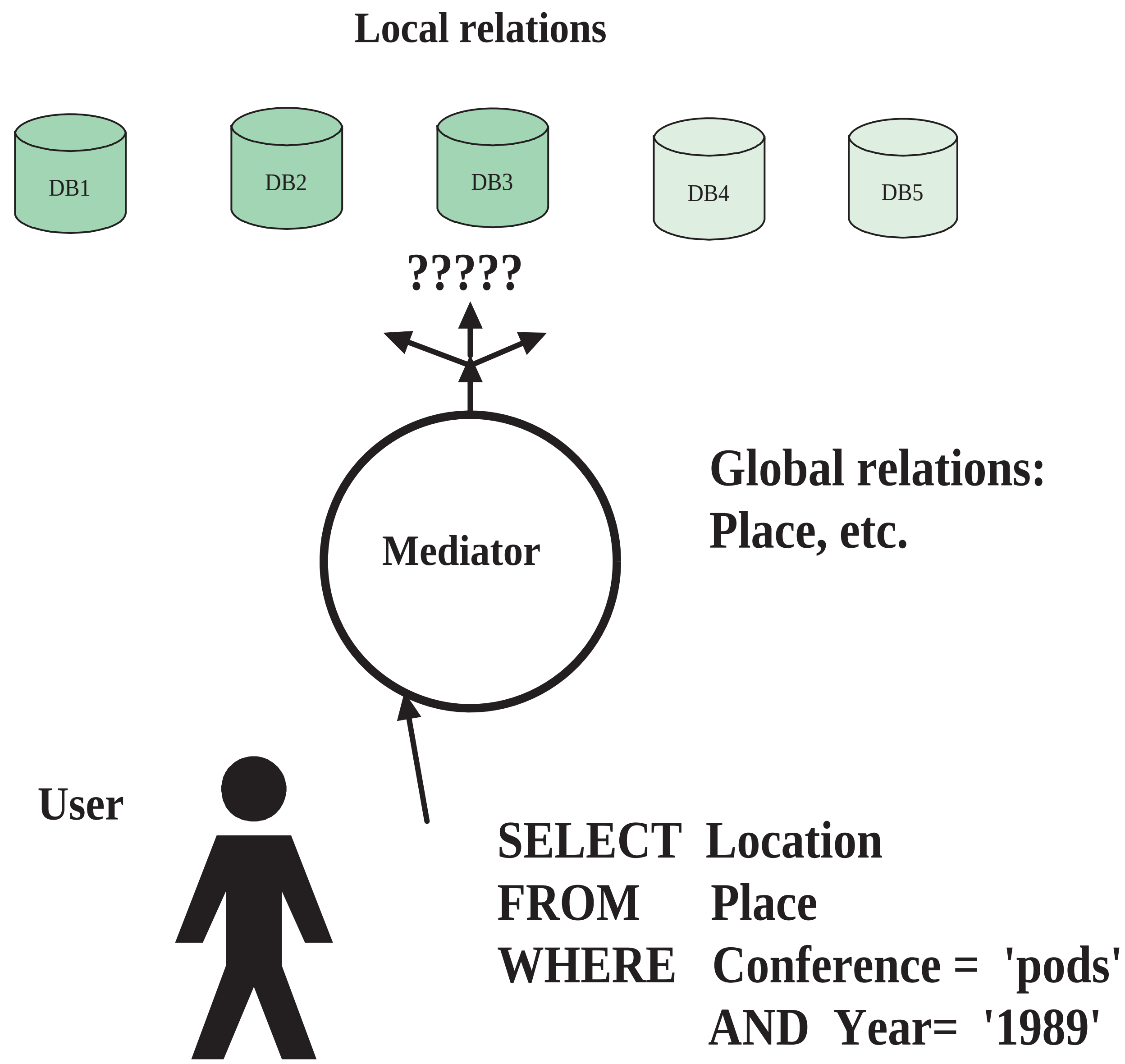
Example: Consider global schema G for a DB "containing" information about movies:

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

Query posed to G: "Comedies with their reviews produced since 1950?"

$$Ans(Title, Review) \leftarrow$$
$$Movie(Title, Year, Director, comedy),$$
$$Review(Title, Review), Year \geq 1950.$$

Data is not in tables $Movie$, $Review$!!



Query plan is a new query identifying: relevant data sources, relevant data in them, and the way the results have to be combined

The LAV Approach to Integration

- The mediator needs to know source contents
- The sources are described by means of a set of formulas that relate the global with the local schemata

  They define the mappings between the global schema and the local schemata
- One of the approaches to describe this relationship is
  - **Local-as-View (LAV)**: Each relation in a local schema (data source) is described as a view of the global schema
- Somehow unusual: now views contain data, but "base tables" don't

Example: Sources for global schema G:

$$S_1\colon V_1(Title, Year, Director) \leftarrow$$
$$Movie(Title, Year, Director, Genre),$$
$$American(Director), Year \geq 1960,$$
$$Genre = comedy.$$

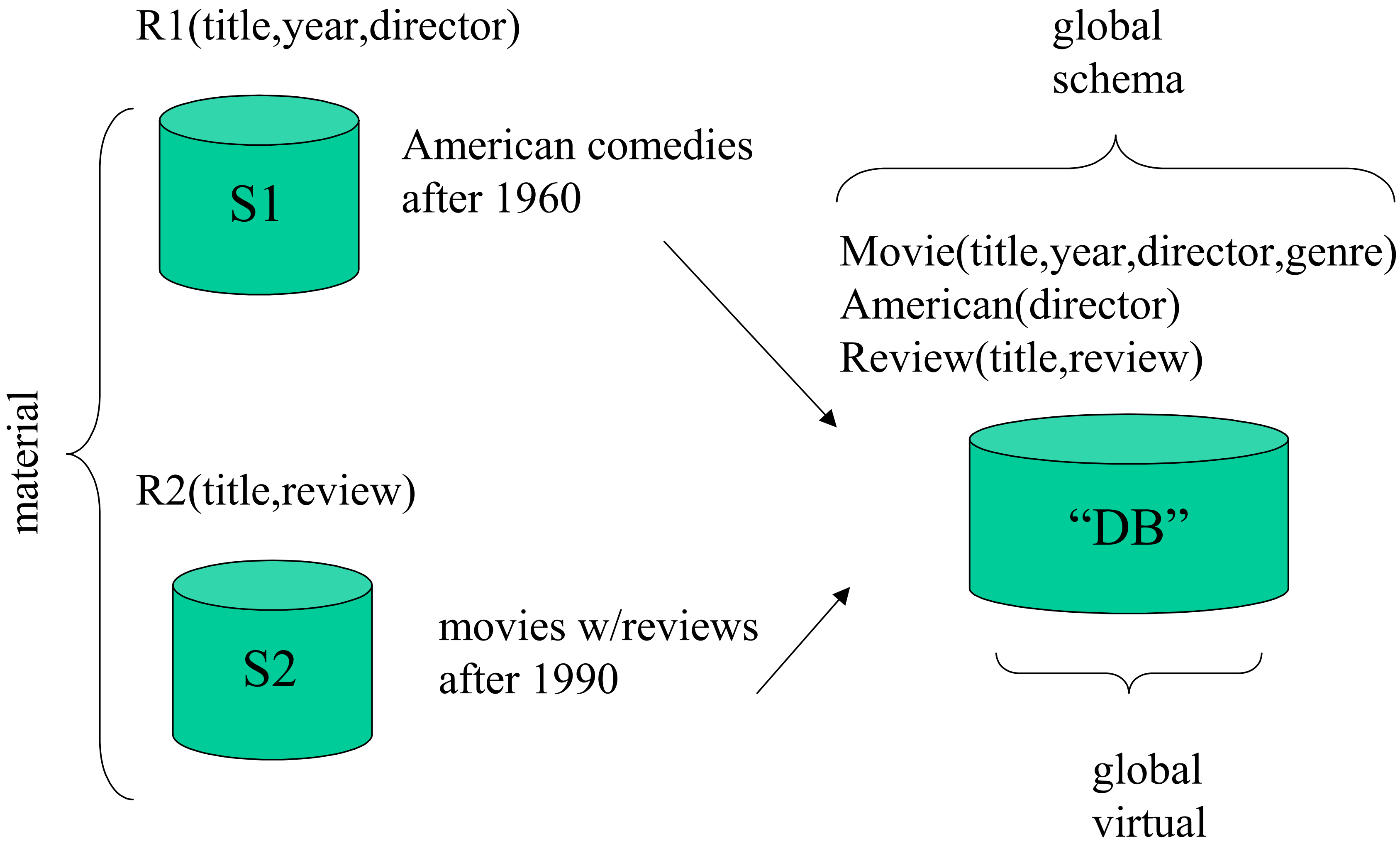
S_1 contains comedies, not older than 1960, with American directors and their years

$$S_2\colon V_2(Title, Review) \leftarrow$$
$$Movie(Title, Year, Director, Genre),$$
$$Year \geq 1990, Review(Title, Review).$$

S_2 contains movies not older than 1990 with their reviews, but not their directors

Definition of sources are independent!

There could be sources other than S_2 with comedies after 1990 with reviews, then S_2 could be an "incomplete" source


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

Again:

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

Information is in the sources, now, views ...

A query plan for our query:

$$\begin{aligned} Ans'(Title, Review) \leftarrow \\ & V_1(Title, Year, Director), \\ & V_2(Title, Review). \end{aligned}$$

Query was *rewritten* in terms of the views and can be computed

How to come up with such a plan?

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

Advantages of LAV:

- More flexibility to add new sources to the integration system
- A new source is just a new view definition
- Other sources do not need to be considered at this level, no interaction with other sources
- Easier to specify constraints on the **individual** sources
- Only the plan generator has to be aware of the new source
- In general, plan generation is non trivial, actually computationally complex

For the same reasons: it is easy to get into inconsistecies at the **global** level

Virtual Integration and Consistency

The sources, independently, may be consistent, but taken together, possibly not

What to do if we ask about a person's ID card number and we get two different numbers, each coming from a different source?

The mediator should be responsible of solving problems of: redundancy, complementarity, and consistency of data "in" the integrated system

Hopefully anticipating potential inconsistencies and "solving them in advance", when plans are generated

Existing systems can handle consistency issues in a very limited and ad hoc way

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

Actually, in the LAV approach they are **used** in the generation of the query plan

There are situations where without ICs no query plan can be generated

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

Problem: Retrieve only those answers to queries posed to the global system that are "consistent with" the global ICs

The notion of consistency or ICs satisfaction is a holistic notion, the whole DB satisfies the ICs or not

So, what are the individual pieces of data that are "consistent" and in what sense?

Consistent Answers in Relational DBs

For different reasons a -now material- relational DB may not satisfy certain intended integrity constraints

Not all data participates in the violation of the ICs

The inconsistent database can still give us "consistent" answers to certain queries!

It seems necessary to:

- Give a precise definition of consistent answer to a query in an inconsistent database
- Find mechanisms for obtaining such consistent information from the inconsistent database

Example: A database instance r

$Employee$	$Name$	$Salary$
	$J.Page$	5,000
	$J.Page$	8,000
	$V.Smith$	3,000
	$M.Stowe$	7,000

FD: $Name \rightarrow Salary$

r violates FD, by the tuples with $J.Page$ in $Name$

There are two possible ways to repair the database in a minimal way if only deletions/insertions of whole tuples are allowed

r_1

$Employee$	$Name$	$Salary$
	$J.Page$	5,000
	$V.Smith$	3,000
	$M.Stowe$	7,000

r_2

$Employee$	$Name$	$Salary$
	$J.Page$	8,000
	$V.Smith$	3,000
	$M.Stowe$	7,000

Certain information persists in both repairs, e.g. $(M.Stowe, 7,000)$

It does not participate in the violation of the FD

Some information does not persist in all repairs, e.g. $(J.Page, 8,000)$

It does participate in the violation of FD

Basic Notions: Fixed DB schema and domain; a set of ICs, IC

Definition: (Arenas, Bertossi, Chomicki; PODS99)

A *repair* of a database instance r is a database instance r'

- over the same schema and domain
- satisfies IC
- differs from r by a minimal set of changes (insertions/deletions of tuples), wrt set inclusion

Definition: (Arenas, Bertossi, Chomicki, PODS99)

A tuple $\bar{t}$ is a **consistent answer** to query $Q(\bar{x})$ in r iff $\bar{t}$ is an answer to query $Q(\bar{x})$ in every repair r' of r

$$r \models KQ[\bar{t}] \iff \text{for every repair } r' \text{ of } r: \; r' \models Q[\bar{t}]$$

Example: The two repairs r_1, $r2$ of r

$Employee$	$Name$	$Salary$
	$J.Page$	5,000
	$V.Smith$	3,000
	$M.Stowe$	7,000

$Employee$	$Name$	$Salary$
	$J.Page$	8,000
	$V.Smith$	3,000
	$M.Stowe$	7,000

$$r \models K\ Employee(M.Stowe,\ 7{,}000)$$

$$r \models K\ (Employee(J.Page,\ 5{,}000) \vee Employee(J.Page,\ 8{,}000))$$

$$r \models K\ \exists X\, Employee(J.Page, X)$$

$$r \not\models K\ Employee(J.Page,\ 8{,}000)$$

$$r \not\models K\ \exists Y\, Employee(Y,\ 5{,}000)$$

The notion of of consistent answer in an inconsistent database is model-theoretic

It appeals to all repairs of the original instance

We want to **compute** consistent answers

But not by computing all possible repairs and detencting answers in common!!

Rather working with the original instance alone

A possible approach: Query transformation!!

- Transform the query and keep the database instance

- In principle applicable to first-order queries and ICs

- Qualify the query with conditions derived from its interaction with ICs

  - To locally satisfy the ICs

  - To discriminate between tuples in the answer set

  - Inspired by "semantic query optimization"

Consistent answers to $Q(\bar{x})$ in r??

Rewrite query: $Q(\bar{x}) \longmapsto T(Q(\bar{x}))$

Retrieve from r the (ordinary) answers to $T(Q(\bar{x}))$

Example: A *FD*

$$\forall XYZ\,(\neg Employee(X,Y)\ \vee\ \neg Employee(X,Z)$$
$$\vee\ \ Y = Z)$$

Query Q to r: $\quad Employee(X,Y)?$

The consistent answers – $(V.Smith$, 3,000), $(M.Stowe$, 7,000) (and nothing else) – can be obtained with transformed query

$$T(Q(X,Y)) := Employee(X,Y)\ \wedge$$
$$\forall Z\,(\neg Employee(X,Z) \vee Y = Z)$$

... those tuples (X,Y) for which X does not have and associated Z different from Y ...

The *residue*

$$\forall Z\,(\neg Employee(X,Z)\ \vee\ Y = Z)$$

can be automatically obtained from Q and FD

In general, T is an iterative operator

There are conditions on queries and ICs for soundness, completeness, and termination of T

Natural and useful syntactical classes of ICs satisfy them

However the approach based on (finite) query rewriting works well for non-existentially quantified conjunctive queries

Other methodologies based on logic programming specification of database repairs can be used for general first-order queries

In the rest we concentrate on query transformation

Consistent Answers & Integrated Sources

Back to our problem:

- A global integration system $\mathcal{G}$ mediated according to the LAV approach
- A set of global integrity constraints IC

  We assume IC is *generic*: for every ground literal L, $IC \not\models L$
- A global query Q posed to $\mathcal{G}$
- Want the answers to Q that consistent wrt IC

Apply previous approaches, developed for instances of stand alone relational databases?

Several interesting issues appear:

- We have a virtual global DB: Where and what are the "instances"?
- What does it mean that $\mathcal{G}$ (or a "repair" of it) satisfies IC?
- What is a repair of the global, virtual database?
- What is a consistent answer then?
- Computation of consistent answers?

We need some preliminary notions

A *view* V is defined by means of a formula

$$V(\bar{X}) \leftarrow \varphi_V(\bar{X}),$$

where φ_V is a conjunction of atoms

Example: Source S_1 seen as the view

$$\begin{aligned} V_1(Title, Year, Director) \leftarrow & \\ & Movie(Title, Year, Director, Genre), \\ & American(Director), Year \geq 1960, \\ & Genre = comedy \end{aligned}$$

Given a database instance D, $\varphi_V(D)$ is the extension of V obtained by applying φ_V to D

If V is given an extension v (a data source), v may be incomplete, i.e. $v \subsetneqq \varphi_V(D)$

We say that v is *open* wrt D if $v \subseteq \varphi_V(D)$

An *open global system* $\mathcal{G}$ is determined by a finite set of view definitions plus their pre-given extensions

$$\begin{array}{lcll} V_1(\bar{X}_1) & \leftarrow & \varphi_{V_1}(\bar{X}_1) & \quad\quad v_1 \\ \cdots & \cdots & \cdots & \quad\quad \cdots \\ V_n(\bar{X}_n) & \leftarrow & \varphi_{V_n}(\bar{X}_n) & \quad\quad v_n \end{array}$$

This determines a global schema $\mathcal{R}$: the predicates in the bodies and a domain containing the constants appearing in the definitions and the v_i's

The legal instances must be open:

$$Linst(\mathcal{G}) = \{D \text{ instance over } \mathcal{R} \mid v_i \subseteq \varphi_{V_i}(D), \\ i = 1, \ldots, n\}$$

An open global system may admit more than one legal instance!!

Example: Global system $\mathcal{G}$ with relation $R(X,Y)$ and sources

$$V_1(X,Y) \leftarrow R(X,Y) \;\; \text{with} \;\; v_1 = \{(a,b),(c,d)\}$$
$$V_2(X,Y) \leftarrow R(X,Y) \;\; \text{with} \;\; v_2 = \{(a,c),(d,e)\}$$

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

- $v_1 \subseteq \varphi_1(D) = \{(a,b),(c,d),(a,c),(d,e)\}$
- $v_2 \subseteq \varphi_2(D) = \{(a,b),(c,d),(a,c),(d,e)\}$

All supersets of D are also legal global instances; no subset of D is legal

What are the answers to a query posed to *system* $\mathcal{G}$? There are several global instances ...

The *certain answers* are those that can be obtained as answers from every legal instance

Given: Open global system $\mathcal{G}$, a global query $Q(\bar{X})$

A tuple $\bar{t}$ is a *certain answer* to Q in $\mathcal{G}$ if for every global instance $D \in \ Linst(\mathfrak{G})$, it holds $D \models Q[\bar{t}]$

The global query $Ans(X,Y) \leftarrow R(X,Y)$ to the *system* $\mathcal{G}$ gets $(a,b),(c,d),(a,c),(d,e)$ as certain answers

Local FDs $V_1\colon\ X \rightarrow Y$, $V_2\colon\ X \rightarrow Y$ are satisfied in the sources

Apparently not the the global FD $R\colon X \rightarrow Y$ in $\mathcal{G}$ (what does this mean???)

Only $(c,d),(d,e)$ “should” be consistent (certain) answers

Instances that do not contain “unnecessary” data wrt the source contents should determine if the global system is consistent wrt global ICs

An open global system is *consistent* wrt to global ICs if every *minimal* instance satisfies the ICs

A *minimal* global instance if is a legal instance that is not properly contained in any other legal instance!

$Mininst(\mathcal{G})$:= set of minimal instances of $\mathcal{G}$

Example: (continued) $\mathcal{G}$ is inconsistent: its only minimal instance

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

does not satisfy FD $R\colon X \to Y$

A global system can be inconsistent, but still have legal instances

Considering minimal instances is like imposing a closed-world assumption on the global instances associated to open local sources

The minimal answers answers to a query are those that can be contained from every minimal instance

$$Certain_{\mathcal{G}}(Q) \subseteqq Minimal_{\mathcal{G}}(Q)$$

For monotone queries "minimal" and "certain" answers coincide

With negation in queries, possibly not

At some point, our global queries will contain negation ...

Example: $\mathcal{G} = \{S_1, S_2\}$, with

$S_1 = \langle V_1(X,Y) \leftarrow P(X,Z),\ Q(Z,Y), \{(a,b)\}\rangle$

$S_2 = \langle V_2(X,Y) \leftarrow P(X,Y), \{(a,c)\}\rangle$

Elements of $Mininst(\mathcal{G})$:

$D_z = \{P(a,z), Q(z,b), P(a,c)\}, \quad z \in \mathcal{D}$

FD $P(X,Y)$: $X \rightarrow Y$ violated exactly in those minimal legal instances D_z for which $z \neq c$

The global system is inconsistent

Query Q: $Ans(X,Y) \leftarrow \neg P(X,Y)$

$P(b,a)$ not present in any minimal instance: $(b,a) \in Minimal_{\mathcal{G}}(Q)$

$P(b,a)$ is present in some minimal instances: $(b,a) \notin Certain_{\mathcal{G}}(Q)$

Intuitively, a consistent answer is invariant under sensible restorations of consistency

A repair of a global system is a consistent **instance** that minimally differs from a minimal instance

Difference wrt to inclusion of sets of whole inserted/deleted tuples

Inspired by [Arenas,Bertossi,Chomicki; PODS99]

$Repairs(\mathcal{G}) :=$ set of repairs of $\mathcal{G}$

If $\mathcal{G}$ is consistent, then

$$Repairs(\mathcal{G}) = Mininst(\mathcal{G})$$

Example: Global system $\mathcal{G}_2$ with

$S_1 = \langle V_1(X,Y) \leftarrow R(X,Y), \{(a,b_1)\}\rangle$

$S_2 = \langle V_2(X,Y) \leftarrow R(X,Y), \{(a,b_2)\}\rangle$

$FD \;\; R(X,Y)$: $X \rightarrow Y$

$Mininst(\mathcal{G}_2) = \{D\}$ with

$D = \{R(a,b_1), R(a,b_2)\} \;\not\models\; FD$

$Repairs(\mathcal{G}_2) = \{D^1, D^2\}$ with

$D^1 = \{R(a,b_1)\}$ and $D^2 = \{R(a,b_2)\}$

They minimally differ from D and satisfy FD

$$D^1, D^2 \notin Linst(\mathcal{G}_2)\text{: } \begin{array}{l} v_2 = \{(a,b_2)\} \nsubseteq \varphi_2(D^1) \\ v_1 = \{(a,b_1)\} \nsubseteq \varphi_1(D^2) \end{array}$$

We are not requiring that sources are open wrt to repairs

A repair – still a global instance – may not be a legal instance

Without this, a global system might not be repairable

Repairs are used as an auxiliary concept, to define consistent answers

A tuple $\bar{t}$ is a *consistent answer* to query Q wrt IC if for every $D \in Repair(\mathcal{G})$: $D \models Q[\bar{t}]$

$$Consis_{\mathcal{G}}^{IC}(Q) \ := \ \text{set of consistent answers to } Q \text{ in } \mathcal{G} \text{ wrt } IC$$

Example: (continued) For system $\mathcal{G}_2$

- a is a consistent answer to query $Q_1(X)$: $\exists Y\ R(X,Y)$

- Query $Q_2(X,Y)$: $R(X,Y)$ does not have consistent answers

For an open global system $\mathcal{G}$, global integrity constraints IC, and a global query Q it holds:

(a) $Consis_{\mathcal{G}}^{IC}(Q) \subseteqq Minimal_{\mathcal{G}}(Q)$

(b) If $\mathcal{G}$ is consistent wrt IC, then

$$Consis_{\mathcal{G}}^{IC}(Q) = Minimal_{\mathcal{G}}(Q)$$

Computing Consistent Answers

Framework:

- Virtual integration of open data sources
- LAV approach
- Queries are conjunctions of literals
- Global integrity constraints are universal

Approach:

- Apply methodologies for obtaining consistent answers from stand alone relational databases [ABC; PODS99]

  Compile the ICs into the query, rewriting the original query into a new query

- Pose and answer the new query as usual in integration systems

- Answers obtained are the consistent answers

Given a query Q, the rewritten query is $T^{\omega}(Q)$

$T^{\omega}(Q)$ is obtained iteratively applying the operator T introduced in [ABC; PODS99]

T transforms a query by appending a $residue$ to each database literal appearing in it

The residue of a database literal forces the local satisfaction of the ICs for the tuples satisfying the literal

Residues can be obtained by resolution between the table and the ICs

The methodology based on the T operator works well for queries that are conjunctions of literals

For the usual universal ICs found in database praxis, e.g. functional dependencies, a fixpoint can be reached in a finite number of steps

Example: IC consists of $(P(X) \vee \neg Q(X))$
and $(R(X) \vee \neg P(X) \vee \neg Q(X))$

Query $Q(X)$?

Residue of $Q(X)$ wrt the first IC is $P(X)$
(if $Q(X)$ is true, then $P(X)$ has to be true if the IC must be true)

Residue of $Q(X)$ wrt the second IC is

$$(R(X) \vee \neg P(X))$$

$$T^1(Q(X)) = Q(X) \wedge (R(X) \vee \neg P(X)) \wedge P(X)$$

$$\begin{aligned}
T^2(Q(X)) &= T(T^1(Q(X))) \\
&= Q(X) \wedge (T(R(X)) \vee T(\neg P(X))) \\
&\quad \wedge T(P(X)) \\
&= Q(X) \wedge \\
&\quad (R(X) \vee (\neg P(X) \wedge \neg Q(X))) \\
&\quad \wedge P(X) \wedge (R(X) \vee \neg Q(X))
\end{aligned}$$

$$\begin{aligned}
T^3(Q(X)) &= Q(X) \wedge (R(X) \vee (\neg P(X) \wedge \\
&\quad T(\neg Q(X)))) \wedge P(X) \wedge \\
&\quad (T(R(X)) \vee T(\neg Q(X))) \\
&= T^2(Q(X))
\end{aligned}$$

A fixpoint! $\qquad T^{\omega}(Q(X)) = T^2(Q(X))$

A high level description of the algorithm for obtaining consistent answers to global queries posed to an integration system:

Algorithm for CQA
Input: a global query Q that is a conjunction of literals

1. Rewrite $Q(\bar{X})$ into the first-order query $T^{\omega}(Q(\bar{X}))$ using IC
2. Using a standard methodology, transform $T^{\omega}(Q(\bar{X}))$ into a recursion-free Datalog$^{\neg}$ program $\Pi(T^{\omega}(Q))$, possibly containing built-ins
3. Find a query plan, $Plan(\Pi(T^{\omega}(Q)))$ to answer $\Pi(T^{\omega}(Q))$ seen as a query to the global system
4. Evaluate the query plan on the view extensions of $\mathcal{G}$ to compute an answer set

Program $\Pi(T^\omega(Q))$ may contain negation

$\Pi(T^\omega(Q))$ does not contain recursion

We need a methodology for generating query plans for queries of this kind

$Plan(\Pi(T^\omega(Q)))$ is a new Datalog$^\neg$ query program that is run on top of the data sources that, together, act as the extensional database

Example: Global system $\mathcal{G}_3$

$S_1 = \langle V_1(X,Y) \leftarrow R(X,Y), \{(a,b)\}\rangle$
$S_2 = \langle V_2(X,Y) \leftarrow R(X,Y), \{(a,c),(c,d)\}\rangle$

FD $R(X,Y)$: $X \rightarrow Y$

Global query $Q(X,Y)$: $R(X,Y)$

$Mininst(\mathcal{G}_3) = \{D\}$ with

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

D violates FD through tuples (a,b), (a,c)

It has two repairs: $D^1 = \{R(a,b), R(c,d)\}$
$D^2 = \{R(a,c), R(c,d)\}$

The only consistent answer to the query is the tuple (c,d)

What about the Algorithm for CQA?

FD: $(\neg R(X,Y) \vee \neg R(X,Z) \vee Y = Z)$

Step 1: Rewriting $T^\omega(Q(X,Y))$ of the query

$$R(X,Y) \wedge \neg \exists Z (R(X,Z) \wedge Z \neq Y)$$

Step 2: Translate the query into a Datalog$^\neg$ program $\Pi(T^\omega(Q(X,Y)))$:

$$\begin{aligned} Ans(X,Y) &\leftarrow R(X,Y),\ not\ S(X,Y) \\ S(X,Y) &\leftarrow R(X,Z), Y \neq Z \end{aligned}$$

Step 3: We need a query plan to answer this query program

It contains negation ...

Plans for Queries with Negation

Inverse-Rules Algorithm: query plans to answer global Datalog queries accessing the sources [Duschka,Genesereth,Levy, JLP00]

Plan is *maximally contained* in the the original query and retrieves all its certain answers

IRA allows Datalog queries without negation

We need plans for global queries containing negation

More specifically, for recursion-free Datalog$^{\neg}$ queries with built-ins

We can extend IRA to these programs

Q: recursion-free Datalog$^\neg$ query in terms of global and defined relations

$IRA^\neg$: works as IRA, but when a rule in Q contains a negated literal in the body

$$S(\bar{X}) \leftarrow \ldots, L_1(\bar{X}), \neg G(\bar{X}), L_n(\bar{X}), \ldots$$

it is checked if G can be eventually evaluated in terms of the global relations appearing in the source descriptions

... because those are the global relations that can be evaluated using the inverse rules

If not, then goal G is eliminated, obtaining

$$S(\bar{X}) \leftarrow \ldots, L_1(\bar{X}), L_n(\bar{X}), \ldots$$

Example: Global system $\mathcal{G}_3 = \{S_1, S_2\}$

S_1: $\langle V_1(X,Z) \leftarrow R_1(X,Y), R_2(Y,Z), \{(a,b)\}\rangle$
S_2: $\langle V_2(X,Y) \leftarrow R_3(X,Y), \{(c,d)\}\rangle$

Global query Q: (e.g. obtained after T^ω)

$$\begin{aligned} Ans(X,Z) &\leftarrow R_1(X,Y), R_2(Y,Z), \\ &\quad\; not\ R_4(X,Y) \\ R_4(X,Y) &\leftarrow R_3(X,Y),\ not\ R_5(X,Y) \\ R_7(X) &\leftarrow R_1(X,Y), R_6(X,Y) \end{aligned}$$

IRA obtains from descriptions, inverses rules for "base" global relations:

$$\begin{aligned} R_1(X, f_1(X,Z)) &\leftarrow V_1(X,Z) \\ R_2(f(X,Z), Z) &\leftarrow V_1(X,Z) \\ R_3(X,Y) &\leftarrow V_2(X,Y) \end{aligned}$$

(f: auxiliary function symbol)

Now prune from the query those rules and negated goals that cannot be computed in terms of R_1, R_2, R_3

$$\begin{aligned} Ans(X,Z) &\leftarrow R_1(X,Y), R_2(Y,Z), \\ &\quad\ not\ R_4(X,Y) \\ R_4(X,Y) &\leftarrow R_3(X,Y). \end{aligned}$$

Literal $not\ R_5(X,Y)$ from second query rule; and third query rule were eliminated

We are left with the query plan

$$\begin{aligned} Ans(X,Z) &\leftarrow R_1(X,Y), R_2(Y,Z), \\ &\quad\ not\ R_4(X,Y) \\ R_4(X,Y) &\leftarrow R_3(X,Y) \\ R_1(X, f_1(X,Z)) &\leftarrow V_1(X,Z) \\ R_2(f_1(X,Z),Z) &\leftarrow V_1(X,Z) \\ R_3(X,Y) &\leftarrow V_2(X,Y) \end{aligned}$$

The query plan can be evaluated in a bottom-up manner starting from the extensions:

$v_1 = \{(a, b)\}$ for source V_1

$v_2 = \{(c, d)\}$ for source V_2

$Ans(a, b)$ is obtained as final answer to the global query Q

Tuples containing symbol f are filtered out from the answer set

$- - - - - - \cdots - - - - - -$

For our query programs Q and the corresponding *Plan* obtained with $IRA^{\neg}$, we can prove:

- *Plan* is maximally contained in Q
- *Plan* can be evaluated bottom-up from the source extensions
- *Plan* retrieves exactly $Minimal_{\mathcal{G}}(Q)$

CQA Algorithm Revisited

Example: (continued) Query program obtained after applying T^ω to query Q: $R(X,Y)$ to $\mathcal{G}_3$

$$\begin{aligned} Ans(X,Y) &\leftarrow R(X,Y),\ not\ S(X,Y) \\ S(X,Y) &\leftarrow R(X,Z),\ Y \neq Z \end{aligned}$$

Following *Plan* is obtained with $IRA^{\neg}$:

$$\begin{aligned} Ans'(X,Y) &\leftarrow R(X,Y),\ not\ S(X,Y) \\ S(X,Y) &\leftarrow R(X,Z), Y \neq Z \\ R(X,Y) &\leftarrow V_1(X,Y) \\ R(X,Y) &\leftarrow V_2(X,Y) \end{aligned}$$

The only answer is $Ans'(c,d)$, the only consistent answer

For the CQA Algorithm, it is possible to prove:

- For a conjunctive global query Q

$$Consistent_{\mathcal{G}}(Q) = Minimal(T^{\omega}(Q))$$

- If $T^{\omega}(Q)$ is monotone, then

$$Consistent_{\mathcal{G}}(Q) = Certain_{\mathcal{G}}(T^{\omega}(Q))$$

- $Plan(\Pi(T^{\omega}(Q))$ retrieves exactly $Consis_{\mathcal{G}}(Q)$

Ongoing and Future Work

We have concentrated on:

- Open sources

  What about other kinds? Closed? Clopen?

  The more general approach presented in [Grahne, Mendelzon;ICDT99] could be used

- LAV approach

  What about GAV and hybrid approaches?

  Some work done by [Lembo,Lenzerini,Rosati;KRDB02]

- Non existential conjunctive queries

  What about more general queries?

  Take advantage of methodologies based on logic programming with answer set semantics for CQA

  They work for general FO queries for stand alone databases

  [Arenas, Bertossi, Chomicki; FQAS00]
  [Barcelo, Bertossi; NMR02]
  [Greco, Greco, Zumpano; ICLP01]

- Universal ICs

  What about referential ICs?

  [Barcelo, Bertossi; PADL03]