Consistent Query Answering in Databases

Prof. Leopoldo Bertossi

Carleton University
School of Computer Science
Ottawa, Canada

bertossi@scs.carleton.ca
www.scs.carleton.ca/~bertossi

Faculty Fellow, IBM Center for Advanced Studies, Toronto Lab.

CanDB 04, Toronto, August 2004.

The Problem

For several reasons a database may become inconsistent with respect to a given set of integrity constraints

- The DBMS that does not have a mechanism to maintain by itself certain class of ICs

  (and no other user or application dependent maintenance mechanisms have been created)

- Data of different sources are being integrated, either virtually or under a materialized approach

  Even if the independent data sources are consistent with respect to certain ICs, the global integrated system might be inconsistent with respect to other global ICs

- New constraints are to be imposed on a pre-existing database, i.e., legacy data

- Soft or user constraints, to be considered only when queries are answered, but without being enforced by the system

It can be difficult, impossible or undesirable to repair the database in order to restore consistency

- Expensive process
- Useful data may be lost
- Not clear how to restore the consistency
- No permission to make the necessary changes

We have to live with inconsistent data ...

Characterizing Semantically Correct Data

Possibly most of the data is still “consistent” and can be retrieved when queries are posed to the database

[Arenas,Bertossi,Chomicki. PODS99]: Consistent data is characterized as the data that is invariant under all minimal restorations of consistency; i.e.

As data that is present in all minimally repaired versions of the original instance: the repairs

A consistent answer to a query can be obtained as a standard answer to the query from *every possible* repair

How to obtain consistent answers to queries?

A Vision

Next DBMSs should provide more flexible, poweful, and user friendly mechanisms for dealing with semantic constraints

ICs could be another input to query answering process taken into account as answers to the query are computed

A query expressed in an enhanced version of SQL

```
SELECT       Name, Salary            (⋆)
FROM         Employee
WHERE        Position = 'manager'
CONSIST/W    FD: Name -> Salary;
```

Where the FD may not be maintained by the DBMS

With DB:

Employee	Name	Salary	Position
	John	*55,000*	*manager*
	Peter	*50,000*	*manager*
	John	*60,000*	*manager*
	Ken	*40,000*	*secretary*

Repairs:

Employee1	Name	Salary	Position
	John	*55,000*	*manager*
	Peter	*50,000*	*manager*
	Ken	*40,000*	*secretary*

and

Employee2	Name	Salary	Position
	Peter	*50,000*	*manager*
	John	*60,000*	*manager*
	Ken	*40,000*	*secretary*

Answers returned from DB to query $(\star)$ should be those consistent with FD: only the tuple *(Peter, 50,000)*

It is the only tuple that is an (usual) answer in both repaired instances to query $(\star)$ (without the consistency clause in the last line)

With the same DB, query

```
SELECT        Name
From          Employee
WHERE         Position = 'manager'
CONSIST/W     FD: Name -> Salary;
```

has as (consistent) answers: *(John)* and *(Peter)*

Computing consistent query answers is different from data cleaning!

In consistent query answering (CQA) we see (some of) the ICs as constraints on query answers rather than on database states

What about computing CQA?

Query $(\star)$ can be transformed into a standard SQL query to be posed to the original database

```
SELECT      Name, Salary
FROM        Employee
WHERE       Position = 'manager'
            AND NOT EXISTS (
   SELECT *
   FROM   Employee E
   WHERE  E.Name = Name AND
          E.Salary <> Salary);
```

(retrieves employees with their salaries for which there is no other employee with the same name, but different salary)

Standard answers to this standard query from the original database are the consistent answers to query $(\star)$

No repair is needed to answer this query!

First-order query rewriting-based methodology like this provably works only for restricted classes of queries and ICs

For more expressive FO queries and ICs, the query has to be rewritten using a more expressible query languages, e.g. disjunctive logic programs with stable model semantics

Stable models of the program are in one-to-one correspondence with the repairs of the database

Complete computation of them has to be avoided or minimized

Many interesting research issues around optimization of logic programs and their evaluation/implementation!

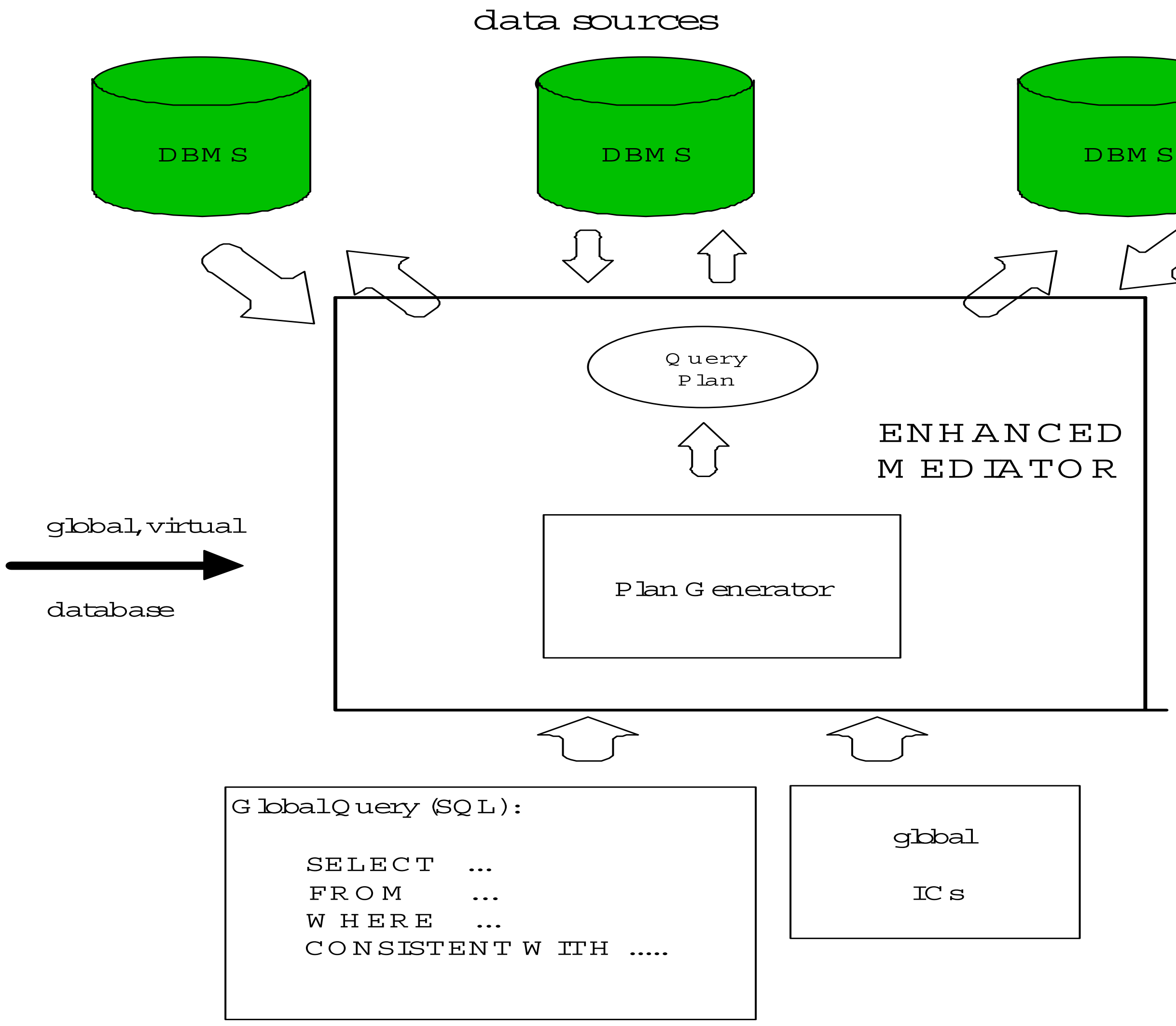
Beyond Single Databases

Concepts and techniques for CQA have been applied in

- Obtaining answers from virtual data integration systems that are consistent with global ICs

  Data sources are independent, queries are posed via a mediator, global ICs are not necessarily maintained

  Answer set programming used to compute CQA ...


data sources
DBMS
DBMS
DBMS
Query Plan
ENHANCED MEDIATOR
Plan Generator
global, virtual
database
Global Query (SQL):
SELECT ...
FROM ...
WHERE ...
CONSISTENT WITH .....
global
ICs

- Query answering in peer-to-peer data exchange systems

  No central data repository; no centralized management; data resides at peers' sites

  Peers exchange data at query answering time according to certain data exchange constraints or data exchange mappings

  Queries are posed to a peer, who, in order to answer the query, imports other peers' data or filters/adjusts its own data

  Trust relationships between peers may influence this process

Databases and Intelligent Information Systems

DBIIS Group Members

Director: Prof. Leo Bertossi

PhD students:

- Loreto Bravo
- Monica Caniupan
- Natalia Villanueva
- Mauricio Vines
- Rasha Tawhid

MSc students:

- Mehdi Kazemi

Some Material

- Bertossi, L. and Chomicki, J. “Query Answering in Inconsistent Databases”. Chapter ’Logics for Emerging Applications of Databases’, J. Chomicki, G. Saake and R. van der Meyden (eds.), Springer, 2003.
- Bertossi, L. and Bravo, L. “Consistent Query Answers in Virtual Data Integration Systems”. To appear as chapter in ‘Inconsistency Tolerance in Database, Knowledgebase and Software Systems’, Springer.
- Bertossi, L. and Bravo, L. “Query Answering in Peer-to-Peer Data Exchange Systems”. Proc. International Workshop on Peer-to-Peer Computing & DataBases (P2P&DB 2004) (collocated with EDBT 04).

- Caniupan, M. “Handling Inconsistencies in Data Warehouses”. In ‘Current Trends in Database Technology’, EDBT 04 Workshops, Springer, LNCS 3268, 2004.
- L. Bertossi. “Consistent Query Answering in Databases”. Tutorial at the Italian Conference on Databases (SEBD 04), Sardinia, June 2004.

  http://www.scs.carleton.ca/~bertossi/talks/talk2SEBD04.pdf

Appendix

Example: Full inclusion dependency $IC\colon\ \forall \bar{x}(P(\bar{x}) \rightarrow Q(\bar{x}))$

Inconsistent instance $r = \{P(\bar{c}), P(\bar{d}), Q(\bar{d}), Q(\bar{e})\}$

The programs use annotation constants in an extra attribute in the database relations

Annotation	Atom	The tuple $P(\bar{a})$ is ...
$\mathbf{t_d}$	$P(\bar{a}, \mathbf{t_d})$	a fact of the database
$\mathbf{f_d}$	$P(\bar{a}, \mathbf{f_d})$	a fact not in the database
$\mathbf{t_a}$	$P(\bar{a}, \mathbf{t_a})$	advised to be made true
$\mathbf{f_a}$	$P(\bar{a}, \mathbf{f_a})$	advised to be made false
$\mathbf{t}^\star$	$P(\bar{a}, \mathbf{t}^\star)$	true or becomes true
$\mathbf{f}^\star$	$P(\bar{a}, \mathbf{f}^\star)$	false or becomes false
$\mathbf{t}^{\star\star}$	$P(\bar{a}, \mathbf{t}^{\star\star})$	true in the repair
$\mathbf{f}^{\star\star}$	$P(\bar{a}, \mathbf{f}^{\star\star})$	false in the repair

Repair program $\Pi(r, IC)$:

1. The original data:

$$P(\bar{c}, \mathbf{t_d}) \leftarrow$$
$$P(\bar{d}, \mathbf{t_d}) \leftarrow$$
$$Q(\bar{d}, \mathbf{t_d}) \leftarrow$$
$$Q(\bar{e}, \mathbf{t_d}) \leftarrow$$

2. Whatever was true (false) or becomes true (false), gets annotated with $\mathbf{t}^\star$ $(\mathbf{f}^\star)$:

$$P(\bar{x}, \mathbf{t}^\star) \leftarrow P(\bar{x}, \mathbf{t_d})$$
$$P(\bar{x}, \mathbf{t}^\star) \leftarrow P(\bar{x}, \mathbf{t_a})$$
$$P(\bar{x}, \mathbf{f}^\star) \leftarrow \ \mathit{not}\ P(\bar{x}, \mathbf{t_d})$$
$$P(\bar{x}, \mathbf{f}^\star) \leftarrow P(\bar{x}, \mathbf{f_a})$$

... the same for Q ...

3. There may be interacting ICs (not here), and the repair process may take several steps, changes could trigger other changes

   We need annotation constants for the local changes $(\mathbf{t_a}, \mathbf{f_a})$, but also annotations $(\mathbf{t}^\star, \mathbf{f}^\star)$ to provide feedback to the rules that produce local repair steps

   $$P(\bar{x}, \mathbf{f_a}) \ \vee \ Q(\bar{x}, \mathbf{t_a}) \ \leftarrow \ P(\bar{x}, \mathbf{t}^\star), Q(\bar{x}, \mathbf{f}^\star)$$

   One rule per IC; that says how to repair the IC in case of a violation

   Passing to annotations $\mathbf{t}^\star$ and $\mathbf{f}^\star$ allows to keep repairing the DB wrt to all the ICs until the process stabilizes

4. Repairs must be coherent: use denial constraints at the program level to prune undesirable models

$$\leftarrow P(\bar{x}, \mathbf{t_a}), P(\bar{x}, \mathbf{f_a})$$
$$\leftarrow Q(\bar{x}, \mathbf{t_a}), Q(\bar{x}, \mathbf{f_a})$$

5. Annotations constants $\mathbf{t}^{\star\star}$ and $\mathbf{f}^{\star\star}$ are used to read off the literals that are inside (outside) a repair

$$P(\bar{x}, \mathbf{t}^{\star\star}) \leftarrow P(\bar{x}, \mathbf{t_a})$$
$$P(\bar{x}, \mathbf{t}^{\star\star}) \leftarrow P(\bar{x}, \mathbf{t_d}),\ \ not\ P(\bar{x}, \mathbf{f_a})$$
$$P(\bar{x}, \mathbf{f}^{\star\star}) \leftarrow P(\bar{x}, \mathbf{f_a})$$
$$P(\bar{x}, \mathbf{f}^{\star\star}) \leftarrow\ \ not\ P(\bar{x}, \mathbf{t_d}),\ \ not\ P(\bar{x}, \mathbf{t_a}).\ \ \text{... etc.}$$

The program has two stable models (and two repairs):

$$\{P(\bar{c},\mathbf{t_d}), ..., P(\bar{c},\mathbf{t}^\star), Q(\bar{c},\mathbf{f}^\star), Q(\bar{c},\mathbf{t_a}), P(\bar{c},\mathbf{t}^{\star\star}), Q(\bar{c},\mathbf{t}^\star), Q(\bar{c},\mathbf{t}^{\star\star}), ...\} \quad \equiv \quad \{P(\bar{c}), Q(\bar{c}), P(\bar{d}), Q(\bar{d}), Q(\bar{e})\}$$

... insert $Q(\bar{c})$!!

$$\{P(\bar{c},\mathbf{t_d}), ..., P(\bar{c},\mathbf{t}^\star), P(\bar{c},\mathbf{f}^\star), Q(\bar{c},\mathbf{f}^\star), P(\bar{c},\mathbf{f}^{\star\star}), Q(\bar{c},\mathbf{f}^{\star\star}), P(\bar{c},\mathbf{f_a}), ...\} \quad \equiv \quad \{P(\bar{d}), Q(\bar{d}), Q(\bar{e})\}$$

... delete $P(\bar{c})$!!

To obtain consistent answers to a FO SQL query:

1. Transform or provide the query as a logic program (this is standard methodology)

2. Run the query program together with the specification program

   ... under the skeptical or cautious stable model semantics that sanctions as true of the programs what is true of all the stable models

Consistent answers to query $P(\bar{x}) \wedge \neg Q(\bar{x})$?

Run repair program $\Pi(r, IC)$ together with query program

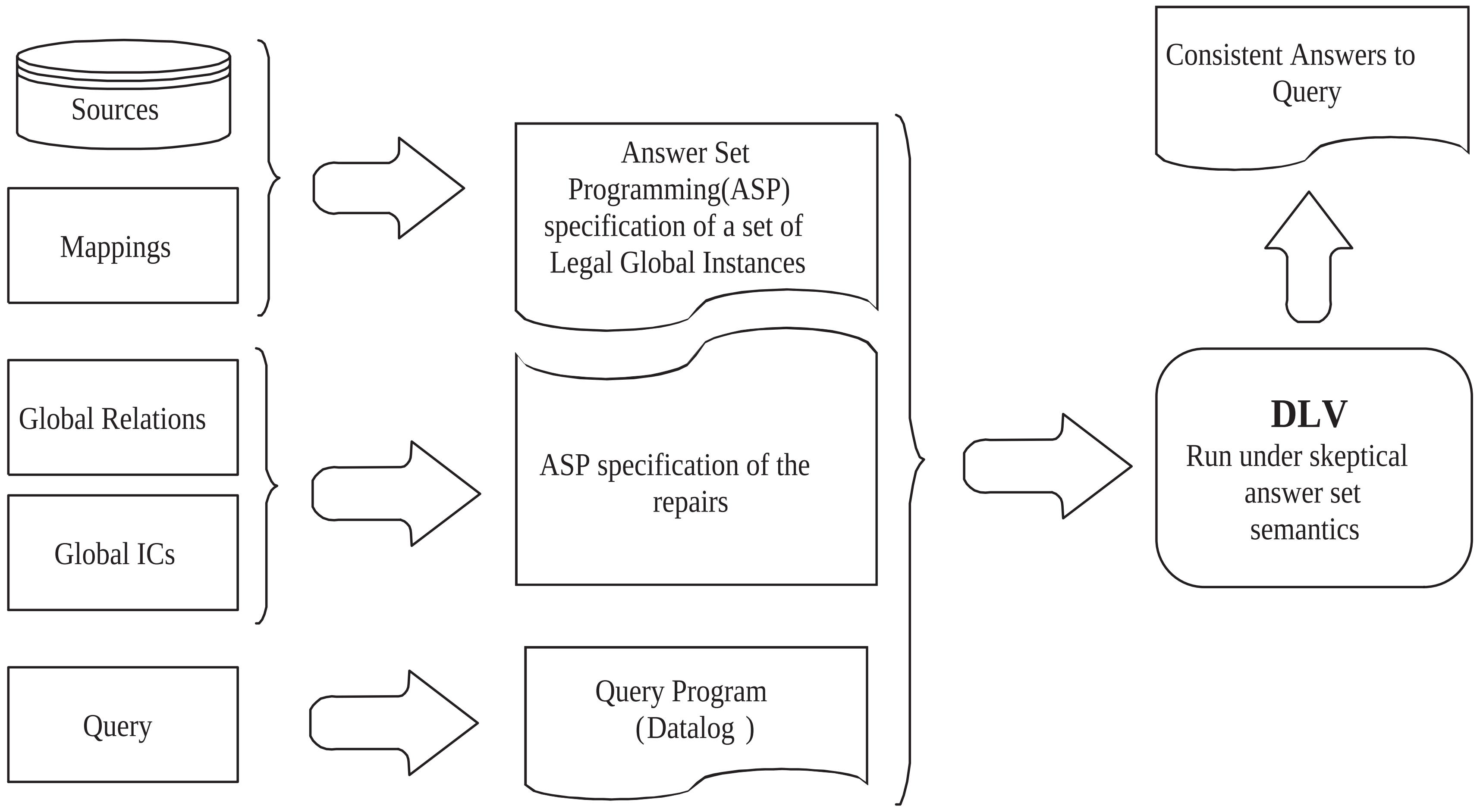
$$Ans(\bar{x}) \leftarrow P(\bar{x}, \mathbf{t}^{\star\star}), Q(\bar{x}, \mathbf{f}^{\star\star})$$

The two previous stable models become extended with ground Ans atoms

None of them in the intersection of the two models

In consequence, under the skeptical SMS, $Ans = \emptyset$, i.e. no consistent answers, as expected ...

Virtual Data Integration


Sources
Mappings
Global Relations
Global ICs
Query
Answer Set Programming(ASP) specification of a set of Legal Global Instances
ASP specification of the repairs
Query Program (Datalog)
DLV
Run under skeptical answer set semantics
Consistent Answers to Query