Consistent Query Answers in Inconsistent Databases

Jan Chomicki

University at Buffalo

Inconsistent Databases

Database instance D:

- a finite first-order structure
- the information about the world

Inconsistent Databases

Database instance D:

- a finite first-order structure
- the information about the world

Integrity constraints IC:

- first-order logic formulas
- the properties of the world

Database instance D:

- a finite first-order structure
- the information about the world

Integrity constraints IC:

- first-order logic formulas
- the properties of the world

Satisfaction of constraints: $D \models IC$

Formula satisfaction in a first-order structure.

Inconsistent Databases

Database instance D:

- a finite first-order structure
- the information about the world

Integrity constraints IC:

- first-order logic formulas
- the properties of the world

Satisfaction of constraints: $D \models IC$

Formula satisfaction in a first-order structure.

Inconsistent database: $D \not\models IC$

Inconsistent Databases

Database instance D:

- a finite first-order structure
- the information about the world

Integrity constraints IC:

- first-order logic formulas
- the properties of the world

Satisfaction of constraints: $D \models IC$

Formula satisfaction in a first-order structure.

Inconsistent database: $D \not\models IC$

Name	City	Salary
Gates	Redmond	20M
Gates	Redmond	30M
Grove	Santa Clara	10M

Name $\rightarrow$ City Salary

Whence Inconsistency?

Sources of inconsistency:

- integration of independent data sources with overlapping data
- time lag of updates (eventual consistency)
- unenforced integrity constraints
- dataspace systems,...

Whence Inconsistency?

Sources of inconsistency:

- integration of independent data sources with overlapping data
- time lag of updates (eventual consistency)
- unenforced integrity constraints
- dataspace systems,...

Eliminating inconsistency?

- not enough information, time, or money
- difficult, impossible or undesirable
- unnecessary: queries may be insensitive to inconsistency

Ignoring Inconsistency

Query results not reliable.

Ignoring Inconsistency

Query results not reliable.

Name	City	Salary
Gates	Redmond	20M
Gates	Redmond	30M
Grove	Santa Clara	10M

Name $\rightarrow$ City Salary

Ignoring Inconsistency

Query results not reliable.

Name	City	Salary
Gates	Redmond	20M
Gates	Redmond	30M
Grove	Santa Clara	10M

Name $\rightarrow$ City Salary

SELECT Name
FROM Employee
WHERE Salary $\leq$ 25M

$\longrightarrow$

Ignoring Inconsistency

Query results not reliable.

Name	City	Salary
Gates	Redmond	20M
Gates	Redmond	30M
Grove	Santa Clara	10M

Name $\rightarrow$ City Salary

SELECT Name
FROM Employee
WHERE Salary $\leq$ 25M

$\longrightarrow$

Name
Gates
Grove

Horizontal Decomposition

Decomposition into two relations:

- violators
- the rest

[Paredaens, De Bra: 1981–83]

Horizontal Decomposition

Decomposition into two relations:

- violators
- the rest

[Paredaens, De Bra: 1981–83]

Name	City	Salary
Gates	Redmond	20M
Gates	Redmond	30M
Grove	Santa Clara	10M

Name $\rightarrow$ City Salary

Horizontal Decomposition

Decomposition into two relations:

- violators
- the rest

[Paredaens, De Bra: 1981–83]

Name	City	Salary
Gates	Redmond	20M
Gates	Redmond	30M
Grove	Santa Clara	10M

Name → City Salary

Grove	Santa Clara	10M

Name → City Salary

Gates	Redmond	20M
Gates	Redmond	30M

Name → City Salary

Exceptions to Constraints

Weakening the contraints:

- functional dependencies $\rightarrow$ denial constraints

[Borgida: TODS'85]

Exceptions to Constraints

Weakening the contraints:

- functional dependencies → denial constraints

[Borgida: TODS'85]

Name	City	Salary
Gates	Redmond	20M
Gates	Redmond	30M
Grove	Santa Clara	10M

Name → City Salary

Exceptions to Constraints

Weakening the contraints:

- functional dependencies → denial constraints

[Borgida: TODS'85]

Name	City	Salary
Gates	Redmond	20M
Gates	Redmond	30M
Grove	Santa Clara	10M

Name → City Salary

→

Name	City	Salary
Gates	Redmond	20M
Gates	Redmond	30M
Grove	Santa Clara	10M

Name → City Salary

except Name='Gates'

The Impact of Inconsistency on Queries

Traditional view

- query results defined irrespective of integrity constraints
- query evaluation may be optimized in the presence of integrity constraints (semantic query optimization)

The Impact of Inconsistency on Queries

Traditional view

- query results defined irrespective of integrity constraints
- query evaluation may be optimized in the presence of integrity constraints (semantic query optimization)

"Post-modernist" view

- inconsistency reflects uncertainty
- query results may depend on integrity constraint satisfaction
- inconsistency may be eliminated or tolerated

Database Repairs

Restoring consistency:

- insertion, deletion, update
- minimal change?

Restoring consistency:

- insertion, deletion, update
- minimal change?

Name	City	Salary
Gates	Redmond	20M
Gates	Redmond	30M
Grove	Santa Clara	10M

Name $\rightarrow$ City Salary

Database Repairs

Restoring consistency:

- insertion, deletion, update
- minimal change?

Name	City	Salary
Gates	Redmond	20M
Gates	Redmond	30M
Grove	Santa Clara	10M

Name $\rightarrow$ City Salary

Name	City	Salary
Gates	Redmond	30M
Grove	Santa Clara	10M

Name $\rightarrow$ City Salary

Name	City	Salary
Gates	Redmond	20M
Grove	Santa Clara	10M

Name $\rightarrow$ City Salary

Consistent Query Answering

Consistent query answer:

Query answer obtained in every repair.

[Arenas,Bertossi,Ch.: PODS'99]

Consistent Query Answering

Consistent query answer:

Query answer obtained in every repair.

[Arenas,Bertossi,Ch.: PODS'99]

Name	City	Salary
Gates	Redmond	20M
Gates	Redmond	30M
Grove	Santa Clara	10M

Name $\rightarrow$ City Salary

Consistent Query Answering

Consistent query answer:

Query answer obtained in every repair.

[Arenas,Bertossi,Ch.: PODS'99]

Name	City	Salary
Gates	Redmond	20M
Gates	Redmond	30M
Grove	Santa Clara	10M

Name $\rightarrow$ City Salary

SELECT Name
FROM Employee
WHERE Salary $\leq$ 25M

$\longrightarrow$

Name
Grove

Consistent query answer:

Query answer obtained in every repair.

[Arenas,Bertossi,Ch.: PODS'99]

Name	City	Salary
Gates	Redmond	20M
Gates	Redmond	30M
Grove	Santa Clara	10M

Name $\rightarrow$ City Salary

SELECT Name
FROM Employee
WHERE Salary $\geq$ 10M

Name
Gates
Grove

Formal definition

What constitutes reliable (consistent) information in an inconsistent database.

Research Goals

Formal definition

What constitutes reliable (consistent) information in an inconsistent database.

Algorithms

How to compute consistent information.

Research Goals

Formal definition

What constitutes reliable (consistent) information in an inconsistent database.

Algorithms

How to compute consistent information.

Computational complexity analysis

- tractable vs. intractable classes of queries and integrity constraints
- tradeoffs: complexity vs. expressiveness.

Research Goals

Formal definition

What constitutes reliable (consistent) information in an inconsistent database.

Algorithms

How to compute consistent information.

Computational complexity analysis

- tractable vs. intractable classes of queries and integrity constraints
- tradeoffs: complexity vs. expressiveness.

Implementation

- preferably using DBMS technology.

Research Goals

Formal definition

What constitutes reliable (consistent) information in an inconsistent database.

Algorithms

How to compute consistent information.

Computational complexity analysis

- tractable vs. intractable classes of queries and integrity constraints
- tradeoffs: complexity vs. expressiveness.

Implementation

- preferably using DBMS technology.

Applications

???

Repair D' of a database D w.r.t. the integrity constraints IC:

- D': over the same schema as D
- $D' \models IC$
- symmetric difference between D and D' is minimal.

Repair D' of a database D w.r.t. the integrity constraints IC:

- D': over the same schema as D
- $D' \models IC$
- symmetric difference between D and D' is minimal.

Consistent query answer to a query Q in D w.r.t. IC:

- an element of the result of Q in every repair of D w.r.t. IC.

Repair D' of a database D w.r.t. the integrity constraints IC:

- D': over the same schema as D
- $D' \models IC$
- symmetric difference between D and D' is minimal.

Consistent query answer to a query Q in D w.r.t. IC:

- an element of the result of Q in every repair of D w.r.t. IC.

Another incarnation of the idea of sure query answers [Lipski: TODS'79].

A Logical Aside

Belief revision

- semantically: repairing $\equiv$ revising the database with integrity constraints
- consistent query answers $\equiv$ counterfactual inference.

Logical inconsistency

- inconsistent database: database facts together with integrity constraints form an inconsistent set of formulas
- trivialization of reasoning does not occur because constraints are not used in relational query evaluation.

Exponentially many repairs

Example relation $R(A, B)$

- violates the dependency $A \rightarrow B$
- has 2^n repairs.

A	B
a_1	b_1
a_1	c_1
a_2	b_2
a_2	c_2
...	
a_n	b_n
a_n	c_n

$A \rightarrow B$

Exponentially many repairs

Example relation $R(A, B)$

- violates the dependency $A \rightarrow B$
- has 2^n repairs.

A	B
a_1	b_1
a_1	c_1
a_2	b_2
a_2	c_2
…	
a_n	b_n
a_n	c_n

$A \rightarrow B$

It is impractical to apply the definition of CQA directly.

Computing Consistent Query Answers

Query Rewriting

Given a query Q and a set of integrity constraints IC, build a query Q^{IC} such that for every database instance D

> *the set of answers to* Q^{IC} *in* D = *the set of consistent answers to* Q *in* D *w.r.t.* IC.

Computing Consistent Query Answers

Query Rewriting

Given a query Q and a set of integrity constraints IC, build a query Q^{IC} such that for every database instance D

> *the set of answers to* Q^{IC} *in* D = *the set of consistent answers to* Q *in* D *w.r.t.* IC.

Representing all repairs

Given IC and D:

1. build a space-efficient representation of all repairs of D w.r.t. IC
2. use this representation to answer (many) queries.

Computing Consistent Query Answers

Query Rewriting

Given a query Q and a set of integrity constraints IC, build a query Q^{IC} such that for every database instance D

> *the set of answers to* Q^{IC} *in* D = *the set of consistent answers to* Q *in* D *w.r.t.* IC.

Representing all repairs

Given IC and D:

1. build a space-efficient representation of all repairs of D w.r.t. IC
2. use this representation to answer (many) queries.

Logic programs

Given IC, D and Q:

1. build a logic program $P_{IC,D}$ whose models are the repairs of D w.r.t. IC
2. build a logic program P_Q expressing Q
3. use a logic programming system that computes the query atoms present in all models of $P_{IC,D} \cup P_Q$.

Constraint classes

Universal constraints

$\forall.\ \neg A_1 \vee \cdots \vee \neg A_n \vee B_1 \vee \cdots \vee B_m$

Constraint classes

Universal constraints

$\forall.\ \neg A_1 \vee \cdots \vee \neg A_n \vee B_1 \vee \cdots \vee B_m$

Example

$\forall.\ \neg Par(x) \vee Ma(x) \vee Fa(x)$

Constraint classes

Universal constraints

$\forall.\ \neg A_1 \vee \cdots \vee \neg A_n \vee B_1 \vee \cdots \vee B_m$

Denial constraints

$\forall.\ \neg A_1 \vee \cdots \vee \neg A_n$

Example

$\forall.\ \neg Par(x) \vee Ma(x) \vee Fa(x)$

Constraint classes

Universal constraints

$\forall.\ \neg A_1 \vee \cdots \vee \neg A_n \vee B_1 \vee \cdots \vee B_m$

Example

$\forall.\ \neg Par(x) \vee Ma(x) \vee Fa(x)$

Denial constraints

$\forall.\ \neg A_1 \vee \cdots \vee \neg A_n$

Example

$\forall.\ \neg M(n, s, m) \vee \neg M(m, t, w) \vee s \leq t$

Constraint classes

Universal constraints

$\forall.\ \neg A_1 \vee \cdots \vee \neg A_n \vee B_1 \vee \cdots \vee B_m$

Denial constraints

$\forall.\ \neg A_1 \vee \cdots \vee \neg A_n$

Functional dependencies

$X \rightarrow Y$:

- a key dependency in F if X is a key
- a primary-key dependency: only one key exists

Example

$\forall.\ \neg Par(x) \vee Ma(x) \vee Fa(x)$

Example

$\forall.\ \neg M(n, s, m) \vee \neg M(m, t, w) \vee s \leq t$

Constraint classes

Universal constraints

$\forall.\ \neg A_1 \vee \cdots \vee \neg A_n \vee B_1 \vee \cdots \vee B_m$

Denial constraints

$\forall.\ \neg A_1 \vee \cdots \vee \neg A_n$

Functional dependencies

$X \rightarrow Y$:

- a key dependency in F if X is a key
- a primary-key dependency: only one key exists

Example

$\forall.\ \neg Par(x) \vee Ma(x) \vee Fa(x)$

Example

$\forall.\ \neg M(n, s, m) \vee \neg M(m, t, w) \vee s \leq t$

Example primary-key dependency

Name $\rightarrow$ Address Salary

Constraint classes

Universal constraints

$\forall.\ \neg A_1 \vee \cdots \vee \neg A_n \vee B_1 \vee \cdots \vee B_m$

Denial constraints

$\forall.\ \neg A_1 \vee \cdots \vee \neg A_n$

Functional dependencies

$X \rightarrow Y$:

- a key dependency in F if X is a key
- a primary-key dependency: only one key exists

Inclusion dependencies

$R[X] \subseteq S[Y]$:

- a foreign key constraint if Y is a key of S

Example

$\forall.\ \neg Par(x) \vee Ma(x) \vee Fa(x)$

Example

$\forall.\ \neg M(n, s, m) \vee \neg M(m, t, w) \vee s \leq t$

Example primary-key dependency

Name $\rightarrow$ Address Salary

Constraint classes

Universal constraints

$\forall.\ \neg A_1 \vee \cdots \vee \neg A_n \vee B_1 \vee \cdots \vee B_m$

Example

$\forall.\ \neg Par(x) \vee Ma(x) \vee Fa(x)$

Denial constraints

$\forall.\ \neg A_1 \vee \cdots \vee \neg A_n$

Example

$\forall.\ \neg M(n, s, m) \vee \neg M(m, t, w) \vee s \leq t$

Functional dependencies

$X \rightarrow Y$:

- a key dependency in F if X is a key
- a primary-key dependency: only one key exists

Example primary-key dependency

Name $\rightarrow$ Address Salary

Inclusion dependencies

$R[X] \subseteq S[Y]$:

- a foreign key constraint if Y is a key of S

Example foreign key constraint

$M[Manager] \subseteq M[Name]$

Building queries that compute CQAs

- relational calculus (algebra) $\rightsquigarrow$ relational calculus (algebra)
- SQL $\rightsquigarrow$ SQL
- leads to PTIME data complexity

Building queries that compute CQAs

- relational calculus (algebra) $\leadsto$ relational calculus (algebra)
- SQL $\leadsto$ SQL
- leads to PTIME data complexity

Query

$Emp(x, y, z)$

Building queries that compute CQAs

- relational calculus (algebra) $\rightsquigarrow$ relational calculus (algebra)
- SQL $\rightsquigarrow$ SQL
- leads to PTIME data complexity

Query

$Emp(x, y, z)$

Integrity constraint

$\forall\ x, y, z, y', z'.\ \neg Emp(x, y, z) \lor \neg Emp(x, y', z') \lor z = z'$

Building queries that compute CQAs

- relational calculus (algebra) $\rightsquigarrow$ relational calculus (algebra)
- SQL $\rightsquigarrow$ SQL
- leads to PTIME data complexity

Query

$Emp(x, y, z)$

Integrity constraint

$\forall\ x, y, z, y', z'.\ \neg Emp(x, y, z) \vee \neg Emp(x, y', z') \vee z = z'$

Building queries that compute CQAs

- relational calculus (algebra) $\rightsquigarrow$ relational calculus (algebra)
- SQL $\rightsquigarrow$ SQL
- leads to PTIME data complexity

Query

$Emp(x, y, z)$

Integrity constraint

$\forall\ x, y, z, y', z'.\ \neg Emp(x, y, z) \vee \neg Emp(x, y', z') \vee z = z'$

Rewritten query

$Emp(x, y, z) \wedge \forall\ y', z'.\ \neg Emp(x, y', z') \vee z = z'$

The Scope of Query Rewriting

[Arenas, Bertossi, Ch.: PODS'99]

- Queries: conjunctions of literals (relational algebra: $\sigma, \times, -$)
- Integrity constraints: binary universal

The Scope of Query Rewriting

[Arenas, Bertossi, Ch.: PODS'99]

- Queries: conjunctions of literals (relational algebra: $\sigma, \times, -$)
- Integrity constraints: binary universal

[Fuxman, Miller: ICDT'05]

- Queries: C_{forest}
  - a class of conjunctive queries ($\pi, \sigma, \times$)
  - no non-key or non-full joins
  - no repeated relation symbols
  - no built-ins
- Integrity constraints: primary key functional dependencies

SQL query

```
SELECT Name FROM Emp
WHERE Salary ≥ 10K
```

SQL query

```
SELECT Name FROM Emp
WHERE Salary ≥ 10K
```

SQL rewritten query

```
SELECT e1.Name FROM Emp e1
WHERE e1.Salary ≥ 10K AND NOT EXISTS
       (SELECT * FROM EMPLOYEE e2
       WHERE e2.Name = e1.Name AND e2.Salary < 10K)
```

SQL query

```
SELECT Name FROM Emp
WHERE Salary ≥ 10K
```

SQL rewritten query

```
SELECT e1.Name FROM Emp e1
WHERE e1.Salary ≥ 10K AND NOT EXISTS
       (SELECT * FROM EMPLOYEE e2
       WHERE e2.Name = e1.Name AND e2.Salary < 10K)
```

[Fuxman, Fazli, Miller: SIGMOD'05]

- ConQuer: a system for computing CQAs
- conjunctive (C_{forest}) and aggregation SQL queries
- databases can be annotated with consistency indicators
- tested on TPC-H queries and medium-size databases

Conflict Hypergraph

Vertices

Tuples in the database.

(Gates, Redmond, 20M)

(Grove, Santa Clara, 10M)

(Gates, Redmond, 30M)

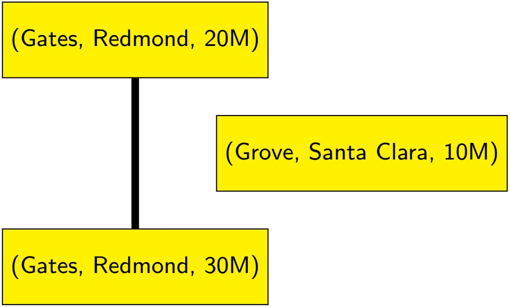
Conflict Hypergraph

Vertices

Tuples in the database.

Edges

Minimal sets of tuples violating a constraint.



Conflict Hypergraph

Vertices

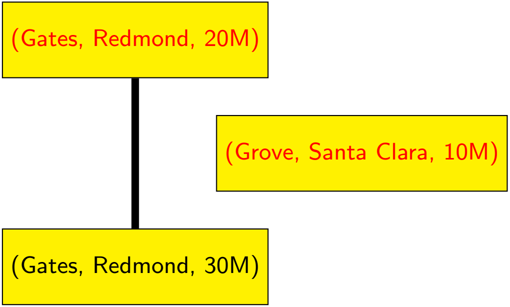
Tuples in the database.

Edges

Minimal sets of tuples violating a constraint.

Repairs

Maximal independent sets in the conflict graph.



Conflict Hypergraph

Vertices

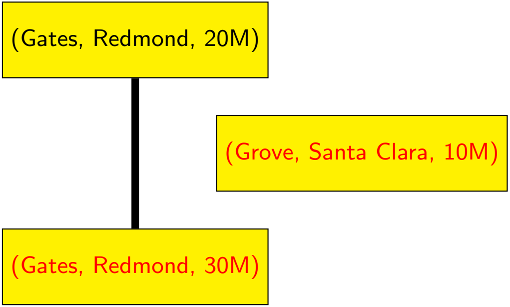
Tuples in the database.

Edges

Minimal sets of tuples violating a constraint.

Repairs

Maximal independent sets in the conflict graph.



Computing CQAs Using Conflict Hypergraphs

Algorithm HProver

INPUT: query Φ a disjunction of ground atoms, conflict hypergraph G
OUTPUT: is Φ false in some repair of D w.r.t. IC?
ALGORITHM:

1. $\neg\Phi = P_1(t_1) \wedge \cdots \wedge P_m(t_m) \wedge \neg P_{m+1}(t_{m+1}) \wedge \cdots \wedge \neg P_n(t_n)$
2. find a consistent set of facts S such that
    - $S \supseteq \{P_1(t_1), \ldots, P_m(t_m)\}$
    - for every fact $A \in \{P_{m+1}(t_{m+1}), \ldots, P_n(t_n)\}$: $A \notin D$ or there is an edge $E = \{A, B_1, \ldots, B_m\}$ in G and $S \supseteq \{B_1, \ldots, B_m\}$.

Computing CQAs Using Conflict Hypergraphs

Algorithm HProver

INPUT: query Φ a disjunction of ground atoms, conflict hypergraph G
OUTPUT: is Φ false in some repair of D w.r.t. IC?
ALGORITHM:

1. $\neg\Phi = P_1(t_1) \wedge \cdots \wedge P_m(t_m) \wedge \neg P_{m+1}(t_{m+1}) \wedge \cdots \wedge \neg P_n(t_n)$
2. find a consistent set of facts S such that
   - $S \supseteq \{P_1(t_1), \ldots, P_m(t_m)\}$
   - for every fact $A \in \{P_{m+1}(t_{m+1}), \ldots, P_n(t_n)\}$: $A \notin D$ or there is an edge $E = \{A, B_1, \ldots, B_m\}$ in G and $S \supseteq \{B_1, \ldots, B_m\}$.

[Ch., Marcinkowski, Staworko: CIKM'04]

- Hippo: a system for computing CQAs in PTIME
- quantifier-free queries and denial constraints
- only edges of the conflict hypergraph are kept in main memory
- optimization can eliminate many (sometimes all) database accesses in HProver
- tested for medium-size synthetic databases

Specifying repairs as answer sets of logic programs

- [Arenas, Bertossi, Ch.: FQAS'00, TPLP'03]
- [Greco, Greco, Zumpano: LPAR'00, TKDE'03]
- [Calì, Lembo, Rosati: IJCAI'03]

Specifying repairs as answer sets of logic programs

- [Arenas, Bertossi, Ch.: FQAS'00, TPLP'03]
- [Greco, Greco, Zumpano: LPAR'00, TKDE'03]
- [Calì, Lembo, Rosati: IJCAI'03]

Example

$$emp(x, y, z) \leftarrow emp_D(x, y, z), not\ dubious\_emp(x, y, z).$$
$$dubious\_emp(x, y, z) \leftarrow emp_D(x, y, z), emp(x, y', z'), y \neq y'.$$
$$dubious\_emp(x, y, z) \leftarrow emp_D(x, y, z), emp(x, y', z'), z \neq z'.$$

Specifying repairs as answer sets of logic programs

- [Arenas, Bertossi, Ch.: FQAS'00, TPLP'03]
- [Greco, Greco, Zumpano: LPAR'00, TKDE'03]
- [Calì, Lembo, Rosati: IJCAI'03]

Example

$emp(x, y, z) \leftarrow emp_D(x, y, z), not\ \ dubious\_emp(x, y, z).$
$dubious\_emp(x, y, z) \leftarrow emp_D(x, y, z), emp(x, y', z'), y \neq y'.$
$dubious\_emp(x, y, z) \leftarrow emp_D(x, y, z), emp(x, y', z'), z \neq z'.$

Answer sets

- $\{emp(Gates, Redmond, 20M), emp(Grove, SantaClara, 10M), \ldots\}$
- $\{emp(Gates, Redmond, 30M), emp(Grove, SantaClara, 10M), \ldots\}$

Logic Programs

- disjunction and classical negation
- checking whether an atom is in all answer sets is Π_2^p-complete
- `dlv`, `smodels`, ...

Logic Programs for computing CQAs

Logic Programs

- disjunction and classical negation
- checking whether an atom is in all answer sets is Π_2^p-complete
- `dlv`, `smodels`, ...

Scope

- arbitrary first-order queries
- universal constraints
- approach unlikely to yield tractable cases

Logic Programs for computing CQAs

Logic Programs

- disjunction and classical negation
- checking whether an atom is in all answer sets is Π_2^p-complete
- `dlv`, `smodels`, ...

Scope

- arbitrary first-order queries
- universal constraints
- approach unlikely to yield tractable cases

INFOMIX [Eiter et al.: ICLP'03]

- combines CQA with data integration (GAV)
- uses `dlv` for repair computations
- optimization techniques: localization, factorization
- tested on small-to-medium-size legacy databases

Co-NP-completeness of CQA

Theorem (Ch., Marcinkowski: Inf. Comp.'05)

For primary-key functional dependencies and conjunctive queries, consistent query answering is data-complete for co-NP.

Co-NP-completeness of CQA

Theorem (Ch., Marcinkowski: Inf. Comp.'05)

For primary-key functional dependencies and conjunctive queries, consistent query answering is data-complete for co-NP.

Proof.

Membership: V is a repair iff $V \models IC$ and $W \not\models IC$ if $W = V \cup M$.
Co-NP-hardness: reduction from MONOTONE 3-SAT.

1. Positive clauses $\beta_1 = \phi_1 \wedge \cdots \wedge \phi_m$, negative clauses $\beta_2 = \psi_{m+1} \wedge \cdots \wedge \psi_l$.
2. Database D contains two binary relations $R(A, B)$ and $S(A, B)$:
   - $R(i, p)$ if variable p occurs in ϕ_i, $i = 1, \ldots, m$.
   - $S(i, p)$ if variable p occurs in ψ_i, $i = m + 1, \ldots, l$.
3. A is the primary key of both R and S.
4. Query $Q \equiv \exists x, y, z.\ \big(R(x, y) \wedge S(z, y)\big)$.
5. There is an assignment which satisfies $\beta_1 \wedge \beta_2$ iff there exists a repair in which Q is false.

□

Co-NP-completeness of CQA

Theorem (Ch., Marcinkowski: Inf. Comp.'05)

For primary-key functional dependencies and conjunctive queries, consistent query answering is data-complete for co-NP.

Proof.

Membership: V is a repair iff $V \models IC$ and $W \not\models IC$ if $W = V \cup M$.
Co-NP-hardness: reduction from MONOTONE 3-SAT.

1. Positive clauses $\beta_1 = \phi_1 \wedge \cdots \wedge \phi_m$, negative clauses $\beta_2 = \psi_{m+1} \wedge \cdots \wedge \psi_l$.
2. Database D contains two binary relations $R(A, B)$ and $S(A, B)$:
   - $R(i, p)$ if variable p occurs in ϕ_i, $i = 1, \ldots, m$.
   - $S(i, p)$ if variable p occurs in ψ_i, $i = m + 1, \ldots, l$.
3. A is the primary key of both R and S.
4. Query $Q \equiv \exists x, y, z. \; (R(x, y) \wedge S(z, y))$.
5. There is an assignment which satisfies $\beta_1 \wedge \beta_2$ iff there exists a repair in which Q is false.

□

Q does not belong to C_{forest}.

Data complexity of CQA

	Primary keys	*Arbitrary keys*	*Denial*	*Universal*
$\sigma, \times, -$				
$\sigma, \times, -, \cup$				
σ, π				
$\sigma, \pi, \times$				
$\sigma, \pi, \times, -, \cup$				

Data complexity of CQA

	Primary keys	*Arbitrary keys*	*Denial*	*Universal*
$\sigma, \times, -$	PTIME	PTIME		PTIME: binary
$\sigma, \times, -, \cup$				
σ, π				
$\sigma, \pi, \times$				
$\sigma, \pi, \times, -, \cup$				

- [Arenas, Bertossi, Ch.: PODS'99]

	Primary keys	*Arbitrary keys*	*Denial*	*Universal*
$\sigma, \times, -$	PTIME	PTIME	PTIME	PTIME: binary
$\sigma, \times, -, \cup$	PTIME	PTIME	PTIME	
σ, π	PTIME	co-NPC	co-NPC	
$\sigma, \pi, \times$	co-NPC	co-NPC	co-NPC	
$\sigma, \pi, \times, -, \cup$	co-NPC	co-NPC	co-NPC	

- [Arenas, Bertossi, Ch.: PODS'99]
- [Ch., Marcinkowski: Inf.Comp.'05]

Data complexity of CQA

	Primary keys	*Arbitrary keys*	*Denial*	*Universal*
$\sigma, \times, -$	PTIME	PTIME	PTIME	PTIME: binary
$\sigma, \times, -, \cup$	PTIME	PTIME	PTIME	
σ, π	PTIME	co-NPC	co-NPC	
$\sigma, \pi, \times$	co-NPC PTIME: C_{forest}	co-NPC	co-NPC	
$\sigma, \pi, \times, -, \cup$	co-NPC	co-NPC	co-NPC	

- [Arenas, Bertossi, Ch.: PODS'99]
- [Ch., Marcinkowski: Inf.Comp.'05]
- [Fuxman, Miller: ICDT'05]

Data complexity of CQA

	Primary keys	*Arbitrary keys*	*Denial*	*Universal*
$\sigma, \times, -$	PTIME	PTIME	PTIME	PTIME: binary Π_2^p-complete
$\sigma, \times, -, \cup$	PTIME	PTIME	PTIME	Π_2^p-complete
σ, π	PTIME	co-NPC	co-NPC	Π_2^p-complete
$\sigma, \pi, \times$	co-NPC PTIME: C_{forest}	co-NPC	co-NPC	Π_2^p-complete
$\sigma, \pi, \times, -, \cup$	co-NPC	co-NPC	co-NPC	Π_2^p-complete

- [Arenas, Bertossi, Ch.: PODS'99]
- [Ch., Marcinkowski: Inf.Comp.'05]
- [Fuxman, Miller: ICDT'05]
- [Staworko, Ph.D.]

Tuple-based repairs

- asymmetric treatment of insertion and deletion:
  - repairs by minimal deletions only [Ch., Marcinkowski: Inf.Comp.'05]: data possibly incorrect but complete
  - repairs by minimal deletions and arbitrary insertions [Calì, Lembo, Rosati: PODS'03]: data possibly incorrect and incomplete
- minimal cardinality changes [Lopatenko, Bertossi: ICDT'07]

Tuple-based repairs

- asymmetric treatment of insertion and deletion:
  - repairs by minimal deletions only [Ch., Marcinkowski: Inf.Comp.'05]: data possibly incorrect but complete
  - repairs by minimal deletions and arbitrary insertions [Calì, Lembo, Rosati: PODS'03]: data possibly incorrect and incomplete
- minimal cardinality changes [Lopatenko, Bertossi: ICDT'07]

Attribute-based repairs

- (A) ground and non-ground repairs [Wijsen: TODS'05]
- (B) project-join repairs [Wijsen: FQAS'06]
- (C) repairs minimizing Euclidean distance [Bertossi et al.: DBPL'05]
- (D) repairs of minimum cost [Bohannon et al.: SIGMOD'05].

Tuple-based repairs

- asymmetric treatment of insertion and deletion:
  - repairs by minimal deletions only [Ch., Marcinkowski: Inf.Comp.'05]: data possibly incorrect but complete
  - repairs by minimal deletions and arbitrary insertions [Calì, Lembo, Rosati: PODS'03]: data possibly incorrect and incomplete
- minimal cardinality changes [Lopatenko, Bertossi: ICDT'07]

Attribute-based repairs

- (A) ground and non-ground repairs [Wijsen: TODS'05]
- (B) project-join repairs [Wijsen: FQAS'06]
- (C) repairs minimizing Euclidean distance [Bertossi et al.: DBPL'05]
- (D) repairs of minimum cost [Bohannon et al.: SIGMOD'05].

Computational complexity

- (A) and (B): similar to tuple based repairs
- (C) and (D): checking existence of a repair of cost $< K$ NP-complete.

The Need for Attribute-based Repairing

Tuple-based repairing leads to information loss.

The Need for Attribute-based Repairing

Tuple-based repairing leads to information loss.

EmpDept

Name	*Dept*	*Location*
John	Sales	Buffalo
Mary	Sales	Toronto

Name $\rightarrow$ Dept

Dept $\rightarrow$ City

The Need for Attribute-based Repairing

Tuple-based repairing leads to information loss.

EmpDept

Name	*Dept*	*Location*
John	Sales	Buffalo
Mary	Sales	Toronto

Name $\rightarrow$ Dept

Dept $\rightarrow$ City

Name	*Dept*	*Location*
John	Sales	Buffalo

Name $\rightarrow$ Dept

Dept $\rightarrow$ City

Name	*Dept*	*Location*
Mary	Sales	Toronto

Name $\rightarrow$ Dept

Dept $\rightarrow$ City

Attribute-based Repairs through Tuple-based Repairs

Repair a lossless join decomposition.

The decomposition:

$$\pi_{Name,Dept}(EmpDept) \bowtie \pi_{Dept,Location}(EmpDept)$$

Attribute-based Repairs through Tuple-based Repairs

Repair a lossless join decomposition.

The decomposition:

$$\pi_{Name,Dept}(EmpDept) \bowtie \pi_{Dept,Location}(EmpDept)$$

Name	*Dept*	*Location*
John	Sales	Buffalo
John	Sales	Toronto
Mary	Sales	Buffalo
Mary	Sales	Toronto

Name $\rightarrow$ Dept

Dept $\rightarrow$ City

Attribute-based Repairs through Tuple-based Repairs

Repair a lossless join decomposition.

The decomposition:

$$\pi_{Name,Dept}(EmpDept) \bowtie \pi_{Dept,Location}(EmpDept)$$

Name	*Dept*	*Location*
John	Sales	Buffalo
John	Sales	Toronto
Mary	Sales	Buffalo
Mary	Sales	Toronto

Name $\rightarrow$ Dept

Dept $\rightarrow$ City

Name	*Dept*	*Location*
John	Sales	Buffalo
Mary	Sales	Buffalo

Name $\rightarrow$ Dept

Dept $\rightarrow$ City

Name	*Dept*	*Location*
John	Sales	Toronto
Mary	Sales	Toronto

Name $\rightarrow$ Dept

Dept $\rightarrow$ City

Probabilistic framework for “dirty” databases

[Andritsos, Fuxman, Miller: ICDE’06]

- potential duplicates identified and grouped into clusters
- worlds ≈ repairs: one tuple from each cluster
- world probability: product of tuple probabilities
- clean answers: in the query result in some (supporting) world
- clean answer probability: sum of the probabilities of supporting worlds
  - consistent answer: clean answer with probability 1

Probabilistic framework for "dirty" databases

[Andritsos, Fuxman, Miller: ICDE'06]

- potential duplicates identified and grouped into clusters
- worlds ≈ repairs: one tuple from each cluster
- world probability: product of tuple probabilities
- clean answers: in the query result in some (supporting) world
- clean answer probability: sum of the probabilities of supporting worlds
  - consistent answer: clean answer with probability 1

Salaries with probabilities

EmpProb

Name	*Salary*	Prob
Gates	20M	0.7
Gates	30M	0.3
Grove	10M	0.5
Grove	20M	0.5

Name → Salary

Computing Clean Answers

SQL query

```
SELECT Name
FROM EmpProb e
WHERE e.Salary > 15M
```

Computing Clean Answers

SQL query

```
SELECT Name
FROM EmpProb e
WHERE e.Salary > 15M
```

SQL rewritten query

```
SELECT e.Name,SUM(e.Prob)
FROM EmpProb e
WHERE e.Salary > 15M
GROUP BY e.Name
```

Computing Clean Answers

SQL query

```
SELECT Name
FROM EmpProb e
WHERE e.Salary > 15M
```

SQL rewritten query

```
SELECT e.Name,SUM(e.Prob)
FROM EmpProb e
WHERE e.Salary > 15M
GROUP BY e.Name
```

EmpProb

Name	*Salary*	Prob
Gates	20M	0.7
Gates	30M	0.3
Grove	10M	0.5
Grove	20M	0.5

Name $\rightarrow$ Salary

Computing Clean Answers

SQL query

```
SELECT Name
FROM EmpProb e
WHERE e.Salary > 15M
```

SQL rewritten query

```
SELECT e.Name,SUM(e.Prob)
FROM EmpProb e
WHERE e.Salary > 15M
GROUP BY e.Name
```

EmpProb

Name	*Salary*	Prob
Gates	20M	0.7
Gates	30M	0.3
Grove	10M	0.5
Grove	20M	0.5

Name → Salary

```
SELECT e.Name,SUM(e.Prob)
FROM EmpProb e
WHERE e.Salary > 15M
GROUP BY e.Name
```

→

Computing Clean Answers

SQL query

```
SELECT Name
FROM EmpProb e
WHERE e.Salary > 15M
```

SQL rewritten query

```
SELECT e.Name,SUM(e.Prob)
FROM EmpProb e
WHERE e.Salary > 15M
GROUP BY e.Name
```

EmpProb

Name	*Salary*	Prob
Gates	20M	0.7
Gates	30M	0.3
Grove	10M	0.5
Grove	20M	0.5

Name $\rightarrow$ Salary

```
SELECT e.Name,SUM(e.Prob)
FROM EmpProb e
WHERE e.Salary > 15M
GROUP BY e.Name
```

Name	Prob
Gates	1
Grove	0.5

Consistent Query Answering: Looking Back

PODS'99, June 1999

- Arenas, Bertossi, Ch.: "Consistent Query Answers in Inconsistent Databases."

Consistent Query Answering: Looking Back

PODS'99, June 1999

- Arenas, Bertossi, Ch.: "Consistent Query Answers in Inconsistent Databases."

Other concurrent events:

Consistent Query Answering: Looking Back

PODS'99, June 1999

- Arenas, Bertossi, Ch.: "Consistent Query Answers in Inconsistent Databases."

Other concurrent events:

Consistent Query Answering: Looking Back

PODS'99, June 1999

- Arenas, Bertossi, Ch.: "Consistent Query Answers in Inconsistent Databases."

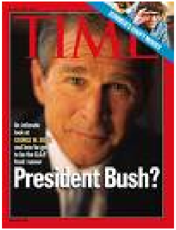
Other concurrent events:

Consistent Query Answering: Looking Back

PODS'99, June 1999

- Arenas, Bertossi, Ch.: "Consistent Query Answers in Inconsistent Databases."

Other concurrent events:



Taking Stock: Good News

Technology

- practical methods for CQA for a subset of SQL:
  - restricted conjunctive/aggregation queries, primary/foreign-key constraints
  - quantifier-free queries/denial constraints
  - LP-based approaches for expressive query/constraint languages
- implemented in prototype systems
- tested on medium-size databases

Taking Stock: Good News

Technology

- practical methods for CQA for a subset of SQL:
  - restricted conjunctive/aggregation queries, primary/foreign-key constraints
  - quantifier-free queries/denial constraints
  - LP-based approaches for expressive query/constraint languages
- implemented in prototype systems
- tested on medium-size databases

The CQA Community

- over 30 active researchers
- up to 100 publications (since 1999)
- outreach to the AI community (qualified success)

Taking Stock: Initial Progress

Taking Stock: Initial Progress

“Blending in” CQA

- data integration: tension between repairing and satisfying source-to-target dependencies
- peer-to-peer: how to isolate an inconsistent peer?

Taking Stock: Initial Progress

“Blending in” CQA

- data integration: tension between repairing and satisfying source-to-target dependencies
- peer-to-peer: how to isolate an inconsistent peer?

Extensions

- nulls:
  - repairs with nulls?
  - clean semantics vs. SQL conformance
- priorities:
  - preferred repairs
  - application: conflict resolution
- XML
  - notions of integrity constraint and repair
  - repair minimality based on tree edit distance?
- aggregate constraints

Taking Stock: Largely Open Issues

Taking Stock: Largely Open Issues

Applications

- no deployed applications
- repairing vs. CQA: data and query characteristics
- heuristics for CQA and repairing

Taking Stock: Largely Open Issues

Applications

- no deployed applications
- repairing vs. CQA: data and query characteristics
- heuristics for CQA and repairing

Consolidation

- taming the semantic explosion
- general first-order definability of CQA
- CQA and data cleaning
- CQA and schema matching/mapping

Taking Stock: Largely Open Issues

Applications

- no deployed applications
- repairing vs. CQA: data and query characteristics
- heuristics for CQA and repairing

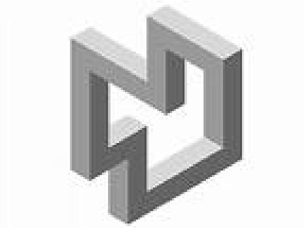
Consolidation

- taming the semantic explosion
- general first-order definability of CQA
- CQA and data cleaning
- CQA and schema matching/mapping

Foundations

- defining measures of consistency
- more refined complexity analysis

Taking Stock: Largely Open Issues



Applications

- no deployed applications
- repairing vs. CQA: data and query characteristics
- heuristics for CQA and repairing

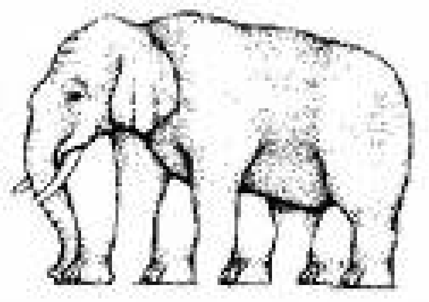
Consolidation

- taming the semantic explosion
- general first-order definability of CQA
- CQA and data cleaning
- CQA and schema matching/mapping

Foundations

- defining measures of consistency
- more refined complexity analysis

Inconsistent elephant (by Oscar Reutersvärd)



Selected overview papers

L. Bertossi, J. Chomicki, Query Answering in Inconsistent Databases. In *Logics for Emerging Applications of Databases*, J. Chomicki, R. van der Meyden, G. Saake [eds.], Springer-Verlag, 2003.

J. Chomicki and J. Marcinkowski, On the Computational Complexity of Minimal-Change Integrity Maintenance in Relational Databases. In *Inconsistency Tolerance*, L. Bertossi, A. Hunter, T. Schaub, editors, Springer-Verlag, 2004.

L. Bertossi, Consistent Query Answering in Databases. SIGMOD Record, June 2006.