Consistent Query Answering in Relational Databases. . . with Preferences

Sławek Staworko[1]
(joint work with Jan Chomicki[2] and Jerzy Marcinkowski[3])

[1]LINKS, INRIA Lille, CNRS, University of Lille, France

[2]University at Buffalo, NY, USA

[3]University of Wrocław, Poland

Montpellier

July 20, 2014

LINKS

- research group INRIA Lille, Université de Lille 1 & 3
- techniques of trees automata (XML), logic, formal languages
- linked (open) data = graphs
- focus on handling heterogeneous databases
- inference of queries, schemas, schema mappings

Motivation

Traditional Databases

Database instance D:

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

Integrity constraints Σ

- first-order logic formulas
- express the properties/rules of the world

Consistent database

- Formula satisfaction in a first-order structure $D \models \Sigma$
- RDBMS enforces consistency

Traditional database

Muppet

Name	Role	DoB
Kermit	Manager	14.03.1965
Miss Piggy	Diva	21.06.1976
T. Statler	Old Man	12.04.1946

Data integration (a novel database application)

Muppet (CBS)

Name	Role	DoB
Kermit	Manager	14.03.1965
Miss Piggy	Diva	21.06.1976
T. Statler	Old Man	12.04.1946

Muppet (Vanity Fair)

Name	Role	DoB
Kermit	Manager	14.03.1965
Miss Piggy	Diva	01.04.1950
T. Statler	Old Man	18.06.1942

Muppet (Federated Database)

Name	Role	DoB
Kermit	Manager	14.03.1965
Miss Piggy	Diva	21.06.1976
Miss Piggy	Diva	01.04.1950
T. Statler	Old Man	12.04.1946
T. Statler	Old Man	18.06.1942

Inconsistency

Source of Inconsistency

- integration of independent data sources with overlapping data
- time lag of updates (eventual consistency)
- unenforced integrity constraints (denormalized DBs)

Eliminating inconsistency?

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

Living with inconsistency?

- ignoring inconsistency
- modifying the schema
- adding exceptions to constraints
- redefining query answers

Ignorantia Beatitudo Est?

A (young) woman of taste doesn't look at the price!

Muppet

Name	Role	DoB
Kermit	Manager	14.03.1965
Miss Piggy	Diva	21.06.1976
Miss Piggy	Diva	01.04.1950
T. Statler	Old Man	12.04.1946
T. Statler	Old Man	18.06.1942

Who's eligible for senior discount?

$Q(x) = \exists y, z.\ Muppet(x, y, z) \wedge z \leq 9.11.1950$

Standard answer semantics is (in)consistency oblivious

$\{$Miss Piggy, T. Statler$\}$

Impact of Inconsistency on Queries

Traditional view

- query results are defined irrespective of integrity constraints
- integrity constraints may be used to optimize the query

Our view

- inconsistency leads to uncertainty (possible worlds)
- integrity constraints guide the user when formulating a query
- query results may depend on satisfaction of integrity constraints
- inconsistency may be eliminated (repairing) or tolerated (consistent query answering)

Basic Notions

Restoring Consistency: Two operations

$$R[A, B] \subseteq P[A, B]$$

r:

A	B
1	2

p:

A	B
1	3

Restoring Consistency: Two operations

$$R[A, B] \subseteq P[A, B]$$

r:

A	B
1	2

p:

A	B
1	3

Delete a tuple

r:

A	B

p:

A	B
1	3

Insert a tuple

r:

A	B
1	2

p:

A	B
1	3
1	2

Repairs

Repair

A consistent instance obtained by performing a minimal set of operations.

Name	Role	DoB
Kermit	Manager	14.03.1965
Miss Piggy	Diva	21.06.1976
Miss Piggy	Diva	01.04.1950
T. Statler	Old Man	12.04.1946
T. Statler	Old Man	18.06.1942

r_1:

Name	Role	DoB
Kermit	Manager	14.03.1965
Miss Piggy	Diva	01.04.1950
T. Statler	Old Man	18.06.1942

r_2:

Name	Role	DoB
Kermit	Manager	14.03.1965
Miss Piggy	Diva	21.06.1976
T. Statler	Old Man	18.06.1942

r_3:

Name	Role	DoB
Kermit	Manager	14.03.1965
Miss Piggy	Diva	21.06.1976
T. Statler	Old Man	12.04.1946

r_4:

Name	Role	DoB
Kermit	Manager	14.03.1965
Miss Piggy	Diva	01.04.1950
T. Statler	Old Man	12.04.1946

Consistent Query Answers

Consistent Query Answer

Query answer present in every repair.

Who's eligible for senior discount?

$Q(x) = \exists y, z.\ Muppet(x, y, z) \wedge z \leq 9.11.1950$

Consistent Answers to $Q(x)$

- T. Statler is a consistent answer to $Q(x)$
- Miss Piggy is not a consistent answer to $Q(x)$ because of r_2 and r_3

CQA scientifically proven to make you feel much younger !

Naïve Data Cleansing

How about removing all conflicting data?

Name	Role	DoB
Kermit	Manager	14.03.1965
Miss Piggy	Diva	21.06.1976
Miss Piggy	Diva	01.04.1950
T. Statler	Old Man	12.04.1946
T. Statler	Old Man	18.06.1942

$\longrightarrow$ r^o:

Name	Role	DoB
Kermit	Manager	14.03.1965

What?!? NO DISCOUNT ?!?
I want to speak to the manager!!!
This is outrageous!!!
The whole world shall hear about this!!!

$Q(x) = \exists y, z.\ Muppet(x, y, z) \wedge z \leq 9.11.1950$
The set of answers to $Q(x)$ in r_0 is empty

Radical approaches lead to information loss.

Computing Consistent Query Answers

Warning: Exponentially Many Repairs

A	B
1	0
1	1
⋮	
n	0
n	1

There are 2^n repairs of this instance w.r.t. the FD $A \to B$.

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 Σ, build a query Q^Σ such that

$$\textit{answers to } Q^\Sigma \textit{ in } D = \textit{consistent answers to } Q \textit{ in } D \textit{ w.r.t. } \Sigma$$

for every database D.

Representing all repairs

Given a database D and a set of integrity constraints Σ

1. build a compact representation of all repairs of D w.r.t. Σ
2. use it to compute the consistent answers

Logic programs

Given a database D, a set of integrity constraints Σ, and a query Q

1. build a logic program $P_{\Sigma,D}$ whose models represent repairs of D w.r.t. Σ
2. build a logic program P_Q expressing Q
3. use a LP system (Smodels, dlv) with cautious evaluation semantics to find answers present in all repairs.

Query Rewriting Example

Database Schema

$Muppet(Name, Role, DoB)$ with $Muppet : Name \rightarrow Role\ DoB$

Query

$\exists y, z.\ Muppet(x, y, z) \land z \leq 9.11.1950$

Integrity constraint $Muppet : Name \rightarrow Role\ DoB$

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

Rewritten query

$\exists y, z.\ Muppet(x, y, z) \land z \leq 9.11.1950 \land \nexists x', y'.\ Muppet(x, y', z') \land z' > 9.11.1950$

Milestones in Query Rewriting

- Arenas, Bertrossi, Chomicki [ABC99]
  - binary universal constraints (includes FDs and full INDs)
  - quantifier-free conjunctive queries
- Fuxman, Miler [FM07]
  - primary key dependencies
  - a class of conjunctive queries C_{forest}
    - no cycles (join graph is a forest)
    - no non-key or non-full joins
    - no repeated relation symbols
    - no built-ins
- Wijsen [Wij10]
  - primary key dependencies
  - a class of conjunctive queries C_{rooted}
    - semantic definition
    - syntactic (effective) characterization that is:
    - based on a notion of an attack graph
    - sound for conjunctive queries without self-join
    - complete for acyclic conjunctive queries without self-join

Rewriting SQL Queries

SQL query

```
SELECT Name FROM Muppet
WHERE DoB ≤ '9.11.1950'
```

SQL rewritten query

```
SELECT m1.Name FROM Muppet m1
WHERE m1.DoB ≤ '9.11.1950' AND NOT EXISTS
       (SELECT * FROM Muppet m2
       WHERE m2.Name = m1.Name AND m2.DoB > '9.11.1950')
```

Together, we shall CONQUER the universe !

(Fuxman, Fazli, Miller [FFM05])

- 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

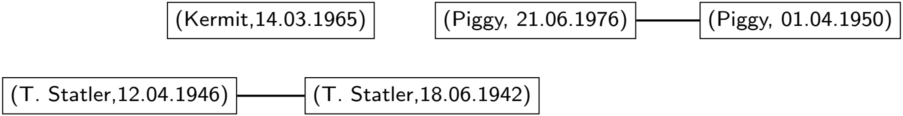
Conflict Hypergraph

Conflict Graph (Arenas et al. [ABC+03b])

Vertex tuple in the database

Edge two conflicting tuples

Repair is a maximal independent set



Extensions

- Conflict Hypergraph for denial constraints: hyperedges span sets of tuples (Chomicki, Marcinkowski)[CM05]
- Extended Conflict Hypergraph for universal constraints: hyperedges may contain tuples to be added (S., Chomicki [SC10])

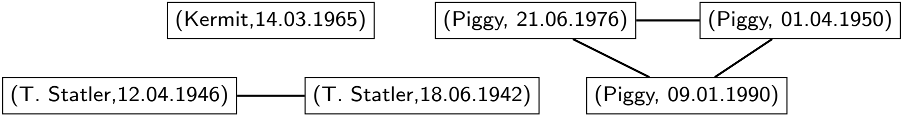
Conflict Hypergraph

Conflict Graph (Arenas et al. [ABC$^+$03b])

Vertex tuple in the database

Edge two conflicting tuples

Repair is a maximal independent set



Extensions

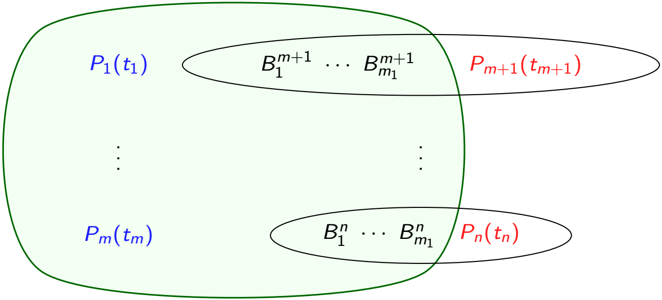
- Conflict Hypergraph for denial constraints: hyperedges span sets of tuples (Chomicki, Marcinkowski)[CM05]
- Extended Conflict Hypergraph for universal constraints: hyperedges may contain tuples to be added (S., Chomicki [SC10])

Computing CQAs Using Conflict Hypergraphs

Algorithm HProver

Input: Φ a disjunction of ground literals, conflict hypergraph G of I w.r.t. Σ
Output: NO if Φ is false in some repair of D w.r.t. Σ?

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 supports all positive facts i.e., $S \supseteq \{P_1(t_1), \ldots, P_m(t_m)\}$
   - S blocks all negative fact i.e., for every $A \in \{P_{m+1}(t_{m+1}), \ldots, P_n(t_n)\} \setminus D$ there is an edge $\{A, B_1, \ldots, B_m\}$ in G such that $S \supseteq \{B_1, \ldots, B_m\}$.



Computing CQA using Conflict Hypergraphs (cont.)

Quantifier-free CNF query Ψ

1. compute a superset A of consistent answers (with an envelope expression)
2. ground the query with a candidate tuple $t \in A$ and convert to CNF
$$\Psi(t) = \Phi_1 \wedge \ldots \wedge \Phi_k$$
3. if for some Φ_i HProver returns NO then discard t
4. otherwise, t is a consistent answer to the query

I'm a powerful beast too !

(Chomicki, Marcinkowski, S. [CMS04])

- Hippo: a system for computing CQAs in PTIME
- quantifier-free queries and denial constraints
- only edges of the conflict hypergraph held in memory
- tested for medium-size synthetic databases

Logic Programs for computing CQAs

Logic Programs [ABC03a, GGZ03, CLR03]

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

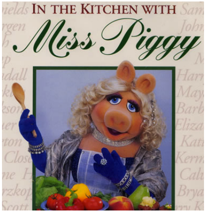
Scope

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

Guess what's in my MIX !

INFOMIX (Eiter et al. [EFGL03, EFGL08])

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



Summary of Complexity Results

What's so (coNP-)hard about it?

$$\varphi = (x_1 \vee \neg x_2 \vee x_4) \wedge (x_2 \vee \neg x_4 \vee x_3) \wedge (\neg x_3 \vee x_4 \vee \neg x_1)$$

Reduction

R :

A	B	$A \to B$
1	0	$x_1 = \mathit{false}$
1	1	$x_1 = \mathit{true}$
⋮		⋮
5	0	$x_5 = \mathit{false}$
5	1	$x_5 = \mathit{true}$

Falsifying valuation for each clause

P :

A_1	B_1	A_2	B_2	A_3	B_3
1	0	2	1	4	0
2	0	4	1	3	0
3	1	4	0	1	1

- repairs correspond to all valuations of variables
- we want all valuations to fail to satisfy φ i.e. there always should be one clause whose none of literals isn't satisfied.

$$Q = \exists x_1, y_1, x_2, y_2, x_3, y_3.\ P(x_1, y_1, x_2, y_2, y_3) \wedge R(x_1, y_1) \wedge R(x_2, y_2) \wedge R(x_3, y_3)$$

Claim

True is the consistent answer to Q iff $\varphi \notin 3SAT$

Constraint classes

Universal constraints
$\forall.\ A_1 \wedge \cdots \wedge A_n \Rightarrow B_1 \vee \cdots \vee B_m$

Example
$\forall.\ Par(x, y) \Rightarrow Ma(x, y) \vee Fa(x, y)$

Tuple-generating dependencies
$\forall.\ A_1 \wedge \cdots \wedge A_n \Rightarrow \exists.\ B$

Example full TGD
$\forall.\ Ma(x, y) \wedge Ma(x, z) \Rightarrow Sib(y, z)$

Denial constraints
$\forall.\ \neg(A_1 \wedge \cdots \wedge A_n)$

Example
$\forall.\ \neg(M(n, s, m) \wedge M(m, t, w) \wedge s > t)$

Functional dependencies
$X \rightarrow Y$

- key dependency: $Y = U$

Example primary-key dependency
Name $\rightarrow$ Address Salary

Inclusion dependencies
$R[X] \subseteq S[Y]$

- a foreign key constraint: key Y

Example foreign key constraint
$M[Manager] \subseteq M[Name]$

Data complexity of CQA

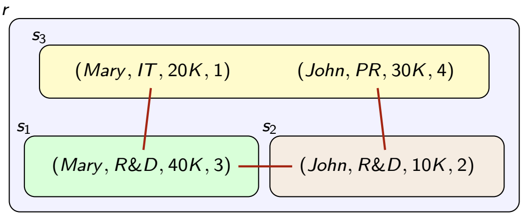
- PTIME for $\{\sigma, \times, \setminus\}$-queries and binary universal constraints (FD + full IND) [ABC99]
- PTIME for $\{\sigma, \times, \setminus, \cup\}$-queries and denial constraints [CM05]
- PTIME for $\{\pi, \sigma\}$-queries and primary keys [CM05]
- coNP-complete for $\{\pi, \sigma, \times\}$-queries and primary keys, and $\{\pi, \sigma\}$-queries and FDs [CM05]
- undecidable for arbitrary functional and inclusion dependencies [CLR03]
- Π_p^2-complete for arbitrary sets of functional and inclusion dependencies (repairs obtained by deletions only) [CM05]
- PTIME for $\{\pi, \sigma, \times\}$-queries in C_{forest} and primary keys [FM07]
- PTIME for quantifier-free queries and acyclic full TGDs, join dependencies, and denial constraints [SC10]
- Π_2^p-complete for universal constraints [SC10]

For more, see surveys

- Chomicki, ICDT'07 [Cho07]
- Bertossi, SIGMOD Record'06 [Ber06]
- Fan, Encyclopedia of Database Systems [Fan09]
- Ioannou and S., DEIS (Dagstuhl 2010) [IS13]

Adding Preferences to the Framework of CQA

Motivation



Schema

$Mgr(Name, Dept, Salary, Reports)$
$Key_1 : Name \quad Key_2 : Dept$

Q_1: John earns more than Mary?

?- Mgr(John, _, s_1, _), Mgr(Mary, _, s_2, _), $s_1 > s_2$.
$r \models Q_1$, but is Q_1 really *true*?

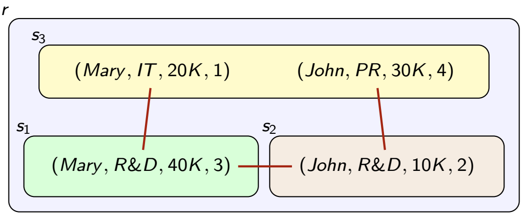
Consistent Query Answers

Repairs:
$r_1 = \{(Mary, R\&D, 40K, 3), (John, PR, 30K, 4)\}$
$r_2 = \{(Mary, IT, 20K, 1), (John, R\&D, 10K, 2)\}$
$r_3 = \{(Mary, IT, 20K, 1), (John, PR, 30K, 4)\}$
Q_1 is not consistently true in r!

What if ???

The user knows:
"s_1, s_2 better than s_3"

Motivation (cont.)



Schema

$Mgr(Name, Dept, Salary, Reports)$
$Key_1 : Name$ $Key_2 : Dept$

Data cleaning

- s_1,s_2 more reliable than s_3.
- the clean database:

$$r' = \left\{ \begin{array}{c} (Mary, R\&D, 40K, 3), \\ (John, R\&D, 10K, 2) \end{array} \right\}$$

- r' is inconsistent.

Preferred Repairs and CQA

Preferred repairs (maximizing reliablility):
$r_1 = \{(Mary, R\&D, 40K, 3), (John, PR, 30K, 4)\}$
$r_2 = \{(Mary, IT, 20K, 1), (John, R\&D, 10K, 2)\}$
~~$r_3 = \{(Mary, IT, 20K, 1), (John, PR, 30K, 4)\}$~~

Q_2: Mary earns more for less?

?- $\mathtt{Mgr(Mary, \_, s_1, r_1), Mgr(John, \_, s_2, r_2), s_1 > s_2, r_1 < r_2.}$

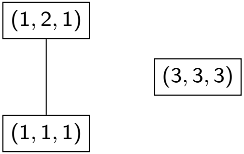
Repairs and Consistent Query Answers

Conflict graph:

- vertices = all tuples
- edges connect conflicting tuples

$R : A \to B$

A	B	C
1	1	1
1	2	1
3	3	3



Repair:

- a maximal consistent subset of the database
- Rep – all repairs of the database
- Rep = MIS

$$r_1 = \{(1,2,1),(3,3,3)\}$$
$$r_2 = \{(1,1,1),(3,3,3)\}$$

Consistent Query Answers:

answers present in every repair.

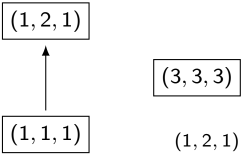
Priorities, Preferences, and Cleaning

Priority $\succ$

- an acyclic orientation of the conflict graph
- $\succ$ is called total when all edges are oriented

Preferred CQA

- A family of preferred repairs is a function that specifies how a given priority is used to define preferred repairs.
- $\mathcal{A}$-$Rep(\succ)$, $\mathcal{B}$-$Rep(\succ)$, ... different families of preferred repairs w.r.t. $\succ$ are possible
- $\mathcal{X}$-preferred consistent answers w.r.t. $\succ$ are the answers present in every $\mathcal{X}$-preferred repair w.r.t $\succ$



$(1,2,1) \succ (1,1,1)$

$\omega_{\succ}(r) = \{t \in r | \neg\exists t' \in r. t' \succ t\}$

Database cleaning with a total $\succ$

- $r' := \varnothing$
- **while** $\omega_{\succ}(r) \neq \varnothing$ **do**
  1. **choose any** $x \in \omega_{\succ}(r)$
  2. add x to r'
  3. remove x from r with neighbors
- **return** r'

Basic Characterization of Preferred Repairs

($\mathcal{P}$1) Non-emptiness

$$\mathcal{X}\text{-}\mathit{Rep}(\succ) \neq \varnothing$$

($\mathcal{P}$2) Monotonicity

$$\begin{gathered}\succ_1 \subseteq \succ_2 \\ \Downarrow \\ \mathcal{X}\text{-}\mathit{Rep}(\succ_2) \subseteq \mathcal{X}\text{-}\mathit{Rep}(\succ_1)\end{gathered}$$

($\mathcal{P}$3) Non-discrimination

$$\mathcal{X}\text{-}\mathit{Rep}(\varnothing) = \mathit{Rep}$$

($\mathcal{P}$4) Categoricity

$\succ$ is total $\Rightarrow |\mathcal{X}\text{-}\mathit{Rep}(\succ)| = 1$

Basic Characterization of Preferred Repairs

($\mathcal{P}$1) Non-emptiness

$$\mathcal{X}\text{-}\mathit{Rep}(\succ) \neq \varnothing$$

($\mathcal{P}$2) Monotonicity

$$\succ_1 \subseteq \succ_2 \\ \Downarrow \\ \mathcal{X}\text{-}\mathit{Rep}(\succ_2) \subseteq \mathcal{X}\text{-}\mathit{Rep}(\succ_1)$$

($\mathcal{P}$3) Non-discrimination

$$\mathcal{X}\text{-}\mathit{Rep}(\varnothing) = \mathit{Rep}$$

($\mathcal{P}$4) Categoricity

$\succ$ is total $\Rightarrow |\mathcal{X}\text{-}\mathit{Rep}(\succ)| = 1$

Trivial family $\mathcal{T}\text{-}\mathit{Rep}(\succ)$:

1° if $\succ$ is total then return the clean database

2° otherwise return *Rep*

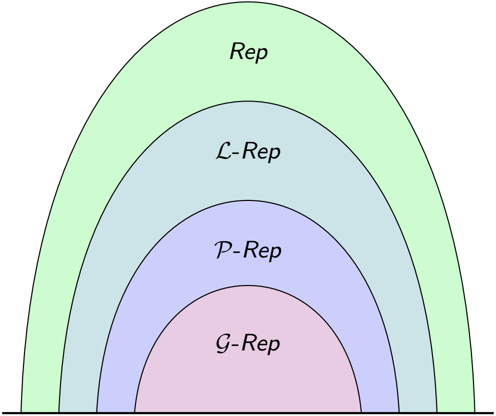
$\mathcal{T}\text{-}\mathit{Rep}$ satisfies $\mathcal{P}1 - \mathcal{P}4$.

Optimal Use of Priorities

Complexity:
(of CQA)

Priority enforcement:

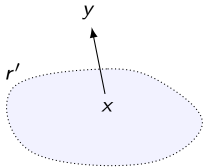
Complexity	Priority enforcement
PTIME	None
co-NP-c	Local (single tuple)
co-NP-c	Pareto
Π_2^p-c	Global (tuple sets)



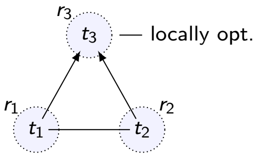
$\mathcal{L}$-*Rep*: Locally-Optimal Repairs

r' is *locally optimal* iff

no tuple $x \in r'$ can be replaced with a tuple y such that: $y \succ x$.
(and the result is consistent)

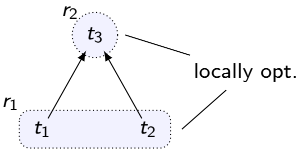


$\mathcal{L}$-*Rep* satisfies $\mathcal{P}1 - \mathcal{P}3$



Key dep.

$\mathcal{L}$-*Rep* is not categorical (not $\mathcal{P}4$)

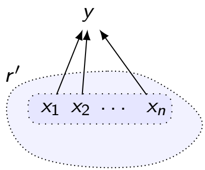


Non-key FD

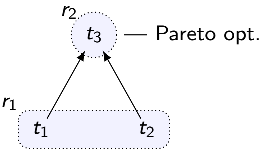
$\mathcal{P}$-*Rep*: Pareto-Optimal Repairs

r' is *Pareto optimal* iff

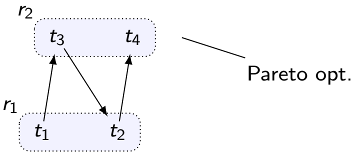
no set $X \subseteq r'$ can be replaced with a tuple y such that:
$$\forall x \in X.\ y \succ x.$$



$\mathcal{P}$-*Rep* satisfies $\mathcal{P}1 - \mathcal{P}4$



Non-key FD

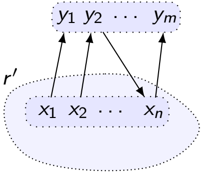


Many FDs

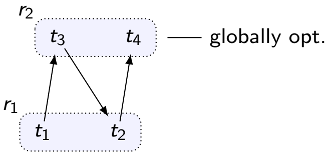
$\mathcal{G}$-*Rep*: Globally-Optimal Repairs

r' is *globally optimal* iff

no set $X \subseteq r'$ can be replaced with a set Y such that:
$\forall x \in X. \exists y \in Y. y \succ x$.



$\mathcal{G}$-*Rep* satisfies $\mathcal{P}1 - \mathcal{P}4$



Many FDs

Alternative characterization

$$\mathcal{G}\text{-}Rep = \ll\text{-maximal repairs}$$
$$r_1 \ll r_2 \Leftrightarrow \forall x \in r_1 \setminus r_2. \exists y \in r_2 \setminus r_1. y \succ x.$$

Computational (data-)complexity

	Repair Check	Consistent Answers to	
		$\{\forall, \exists\}$-free queries	conjunctive queries
Rep	PTIME	PTIME	co-NP-complete
$\mathcal{L}$-*Rep*	PTIME	co-NP-complete	co-NP-complete
$\mathcal{P}$-*Rep*	PTIME	co-NP-complete	co-NP-complete
$\mathcal{G}$-*Rep*	co-NP-complete	Π_p^2-complete	Π_p^2-complete

$\mathcal{L}$-*Rep*, $\mathcal{P}$-*Rep*, and $\mathcal{G}$-*Rep*

For one FD computing consistent answers to $\{\exists, \forall\}$-free queries is PTIME.

Computing preferred CQA with any family of Pareto-optimal repairs satisfying $\mathcal{P}1$ and $\mathcal{P}2$ is co-NP-hard.
(one atom and 2 FDs)

$\mathcal{C}$-*Rep*: Common-optimal repairs

Desired properties:

- *optimality* to enforce priority use
- *monotonicity* ($\mathcal{P}2$) to prevent groundless elimination of repairs
- *non-emptiness* ($\mathcal{P}1$)

$\mathcal{C}$-*Rep* - repairs *common* for all families of (globally) optimal repairs satisfying $\mathcal{P}1$ and $\mathcal{P}2$

- $\mathcal{C}$-*Rep* satisfies $\mathcal{P}1 - \mathcal{P}4$
- $\mathcal{C}\text{-}Rep \subseteq \mathcal{G}\text{-}Rep$
- $\mathcal{C}\text{-}Rep = \mathcal{G}\text{-}Rep$ for priorities that cannot be extended to a cyclic orientation.
- Repair check: PTIME; CQA: co-NP-c

Database cleaning

- $r' := \varnothing$
- **while** $\omega_{\succ}(r) \neq \varnothing$ **do**
  1. **choose any** $x \in \omega_{\succ}(r)$
  2. add x to r'
  3. remove x from r with neighbors
- **return** r'

Alternative characterization

$r' \in \mathcal{C}\text{-}Rep(\succ)$ iff r' can be a result of cleaning the database with $\succ$.

Conclusions

	Repair Check	Consistent Answers to		Possible Applications
		$\{\forall, \exists\}$-free queries	conj. queries	
Rep	PTIME	PTIME	co-NP-c	no priorities given
$\mathcal{L}$-*Rep*	PTIME	co-NP-c		key (no duplicates)
$\mathcal{P}$-*Rep*	PTIME	co-NP-c		one FD (duplicates)
$\mathcal{G}$-*Rep*	co-NP-c	Π_p^2-c		many FDs with
$\mathcal{C}$-*Rep*	PTIME	co-NP-c		mutual conflicts

Related Work

S. Flesca, S. Greco, and E. Zumpano. *Active Integrity Constraints*.

$$S_{\succ}(r') = \{(x,y) \in \succ \mid x \in r'\}$$
$$\mathcal{S}\text{-}Rep(\succ) = \{r' \in Rep \mid S_{\succ}(r') \text{ is maximal}\}$$

- CQA: Π_3^p-complete
- satisfies $\mathcal{P}1$ and $\mathcal{P}3$
- handles cyclic $\succ$, but then
- violates $\mathcal{P}2$ and $\mathcal{P}4$

G. Greco and D. Lembo *Data Integration with Preferences among Sources*.

- repairing a relation by removing tuples has to be *justified* by removing *similar* tuples from other relations.
- satisfies $\mathcal{P}2$, but not $\mathcal{P}1$ (non-emptiness)
- *weakened* framework satisfies $\mathcal{P}1$ but $\mathcal{P}2$ is lost.

M. Arenas, L. Bertossi, and J. Chomicki.
Consistent query answers in inconsistent databases.
In *ACM Symposium on Principles of Database Systems (PODS)*, pages 68–79, 1999.

M. Arenas, L. Bertossi, and J. Chomicki.
Answer sets for consistent query answering in inconsistent databases.
Theory and Practice of Logic Programming, 3(4–5):393–424, 2003.

M. Arenas, L. Bertossi, J. Chomicki, X. He, V. Raghavan, and J. Spinrad.
Scalar aggregation in inconsistent databases.
Theoretical Computer Science (TCS), 296(3):405–434, 2003.

L. Bertossi.
Consistent query answering in databases.
SIGMOD Record, 35(2):68–76, June 2006.

J. Chomicki.
Consistent query answering: Five easy pieces.
In *International Conference on Database Theory (ICDT)*, pages 1–17, 2007.

A. Cali, D. Lembo, and R. Rosati.
On the decidability and complexity of query answering over inconsistent and incomplete databases.

In *ACM Symposium on Principles of Database Systems (PODS)*, pages 260–271, 2003.

J. Chomicki and J. Marcinkowski.
Minimal-change integrity maintenance using tuple deletions.
Information and Computation, 197(1–2):90–121, February 2005.

J. Chomicki, J. Marcinkowski, and S. Staworko.
Computing consistent query answers using conflict hypergraphs.
In *International Conference on Information and Knowledge Management (CIKM)*, pages 417–426. ACM Press, November 2004.

T. Eiter, M. Fink, G. Greco, and D. Lembo.
Efficient evaluation of logic programs for querying data integration systems.
In *International Conference on Logic Programming (ICLP)*, pages 163–177, 2003.

T. Eiter, M. Fink, G. Greco, and D. Lembo.
Repair localization for query answering from inconsistent databases.
ACM Transactions on Database Systems (TODS), 33(2), 2008.

W. Fan.
Constraint-driven database repair.
In *Encyclopedia of Database Systems*, pages 458–463. Springer US, 2009.

A. Fuxman, E. Fazli, and R. J. Miller.
Conquer: Efficient management of inconsistent databases.

In *ACM SIGMOD International Conference on Management of Data*, pages 155–166, 2005.

A. Fuxman and R. J. Miller.
First-order query rewriting for inconsistent databases.
Journal of Computer and System Sciences, 73(4):610–635, 2007.

G. Greco, S. Greco, and E. Zumpano.
A logical framework for querying and repairing inconsistent databases.
IEEE Transactions on Knowledge and Data Engineering, 15(6):1389–1408, 2003.

E. Ioannou and S. Staworko.
Management of inconsistencies in data integration.
In *Data Exchange, Information, and Streams*, pages 217–225, 2013.

S. Staworko and J. Chomicki.
Consistent query answers in the presence of universal constraints.
Information Systems, 35(1):1–22, 2010.

J. Wijsen.
On the first-order expressibility of computing certain answers to conjunctive queries over uncertain databases.
In *ACM Symposium on Principles of Database Systems (PODS)*, pages 179–190, 2010.