Consistent Query Answering

Jan Chomicki
University at Buffalo

October 2017

Table of contents

Database instance D:

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

I

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

Integrity constraints (dependencies)

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

I

Satisfaction of constraints: $D \models IC$

Formula satisfaction in a first-order structure.

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

I

Satisfaction of constraints: $D \models IC$

Formula satisfaction in a first-order structure.

Consistent database: $D \models IC$

Name	City	Salary
Gates	Redmond	30M
Musk	Palo Alto	10M

Name $\rightarrow$ City Salary

Integrity constraints (dependencies)

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

I

Satisfaction of constraints: $D \models IC$

Formula satisfaction in a first-order structure.

Consistent database: $D \models IC$

Name	City	Salary
Gates	Redmond	30M
Musk	Palo Alto	10M

Name $\rightarrow$ City Salary

Inconsistent database: $D \not\models IC$

Name	City	Salary
Gates	Redmond	20M
Gates	Redmond	30M
Musk	Palo Alto	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

Whence Inconsistency?

Sources of inconsistency:

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

Eliminating inconsistency?

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

Whence Inconsistency?

Sources of inconsistency:

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

Eliminating inconsistency?

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

Living with inconsistency?

- ignoring inconsistency
- redefining query answers

Whence Inconsistency?

Sources of inconsistency:

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

Eliminating inconsistency?

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

Living with inconsistency?

- ignoring inconsistency
- redefining query answers

▸ CQA

Eliminating inconsistency

Dropping conflicting tuples

- information is lost

Eliminating inconsistency

Dropping conflicting tuples

- information is lost

Name	City	Salary
Gates	Redmond	20M
Gates	Redmond	30M
Musk	Palo Alto	10M

Name $\rightarrow$ City Salary

Eliminating inconsistency

Dropping conflicting tuples

- information is lost

Name	City	Salary
Gates	Redmond	20M
Gates	Redmond	30M
Musk	Palo Alto	10M

Name $\rightarrow$ City Salary

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

$\longrightarrow$

Name
Musk

Dropping conflicting tuples

- information is lost

Name	City	Salary
Gates	Redmond	20M
Gates	Redmond	30M
Musk	Palo Alto	10M

Name $\rightarrow$ City Salary

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

Name
Musk

Ignoring Inconsistency

Ignoring Inconsistency

Name	City	Salary
Gates	Redmond	20M
Gates	Redmond	30M
Musk	Palo Alto	10M

Name $\rightarrow$ City Salary

Ignoring Inconsistency

Name	City	Salary
Gates	Redmond	20M
Gates	Redmond	30M
Musk	Palo Alto	10M

Name $\rightarrow$ City Salary

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

$\longrightarrow$

Name	City	Salary
Gates	Redmond	20M
Gates	Redmond	30M
Musk	Palo Alto	10M

Name $\rightarrow$ City Salary

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

$\longrightarrow$

Name
Gates
Musk

Ignoring Inconsistency

Name	City	Salary
Gates	Redmond	20M
Gates	Redmond	30M
Musk	Palo Alto	10M

Name $\rightarrow$ City Salary

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

Name
Gates
Musk

Query results not reliable.

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)

Our view

- inconsistency leads to uncertainty
- query results may depend on integrity constraint satisfaction
- inconsistency may be eliminated (repairing) or tolerated (consistent query answering)

Database Repairs

Restoring consistency

- insertion, deletion
- minimal change

Restoring consistency

- insertion, deletion
- minimal change

Name	City	Salary
Gates	Redmond	20M
Gates	Redmond	30M
Musk	Palo Alto	10M

Name $\rightarrow$ City Salary

Restoring consistency

- insertion, deletion
- minimal change

Name	City	Salary
Gates	Redmond	20M
Gates	Redmond	30M
Musk	Palo Alto	10M

Name $\rightarrow$ City Salary

Name	City	Salary
Gates	Redmond	30M
Musk	Palo Alto	10M

Name $\rightarrow$ City Salary

Name	City	Salary
Gates	Redmond	20M
Musk	Palo Alto	10M

Name $\rightarrow$ City Salary

Consistent query answer

- query answer obtained in every repair
- database not changed

(Arenas, Bertossi, Ch. [ABC99])

Consistent query answer

- query answer obtained in every repair
- database not changed

(Arenas, Bertossi, Ch. [ABC99])

Name	City	Salary
Gates	Redmond	20M
Gates	Redmond	30M
Musk	Palo Alto	10M

Name $\rightarrow$ City Salary

Consistent query answer

- query answer obtained in every repair
- database not changed

(Arenas, Bertossi, Ch. [ABC99])

Name	City	Salary
Gates	Redmond	20M
Gates	Redmond	30M
Musk	Palo Alto	10M

Name $\rightarrow$ City Salary

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

$\longrightarrow$

Name
Musk

Consistent query answer

- query answer obtained in every repair
- database not changed

(Arenas, Bertossi, Ch. [ABC99])

Name	City	Salary
Gates	Redmond	20M
Gates	Redmond	30M
Musk	Palo Alto	10M

Name $\rightarrow$ City Salary

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

$\longrightarrow$

Name
Gates
Musk

Research Goals

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

- data cleaning

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.

Basic Notions

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/certain query answers (Lipski [Jr.79]).

A Logical Aside

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.

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.*

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

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

Universal constraints

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

Example

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

Constraint classes

Universal constraints

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

Tuple-generating dependencies

$\forall.\ A_1 \wedge \cdots \wedge A_n \Rightarrow B$

Example

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

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

$\forall.\ A_1 \wedge \cdots \wedge A_n \Rightarrow B$

Example

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

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

$\forall.\ A_1 \wedge \cdots \wedge A_n \Rightarrow B$

Example

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

Denial constraints

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

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

$\forall.\ A_1 \wedge \cdots \wedge A_n \Rightarrow B$

Example

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

Denial constraints

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

Example

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

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

$\forall.\ A_1 \wedge \cdots \wedge A_n \Rightarrow B$

Example

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

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: $Y = U$

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

$\forall.\ A_1 \wedge \cdots \wedge A_n \Rightarrow B$

Example

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

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

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

$\forall.\ A_1 \wedge \cdots \wedge A_n \Rightarrow B$

Example

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

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

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

$\forall.\ A_1 \wedge \cdots \wedge A_n \Rightarrow B$

Example

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

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

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

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

$\forall.\ A_1 \wedge \cdots \wedge A_n \Rightarrow B$

Example

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

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

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

Query Rewriting

Building queries that compute CQAs

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

Query Rewriting

Building queries that compute CQAs

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

Query

$Emp(x, y, z)$

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

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

Query Rewriting

Building queries that compute CQAs

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

Query

$Emp(x, y, z)$

Integrity constraint

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

Rewritten query

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

The Scope of Query Rewriting

(Arenas, Bertossi, Ch. [ABC99])

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

The Scope of Query Rewriting

(Arenas, Bertossi, Ch. [ABC99])

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

(Fuxman, Miller [FM07])

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

SQL Rewriting

SQL query

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

SQL Rewriting

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 Rewriting

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 [FM05])

- 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)

(Musk, Palo Alto, 10M)

(Gates, Redmond, 30M)

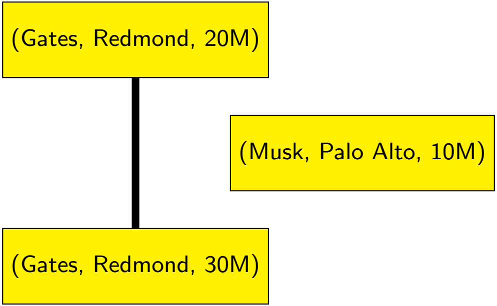
Conflict Hypergraph

Vertices

Tuples in the database.

Edges

Minimal sets of tuples violating a constraint.



Vertices

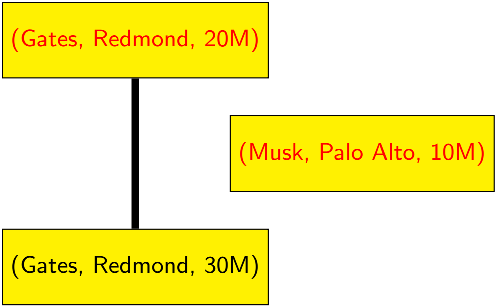
Tuples in the database.

Edges

Minimal sets of tuples violating a constraint.

Repairs

Maximal independent sets in the conflict graph.



Conflict Hypergraph

Vertices

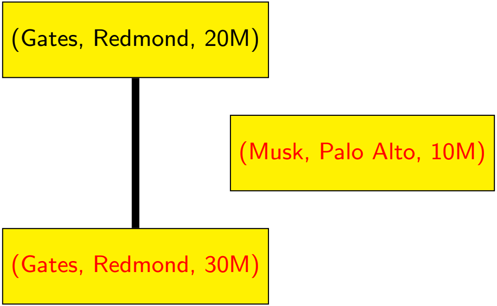
Tuples in the database.

Edges

Minimal sets of tuples violating a constraint.

Repairs

Maximal independent sets in the conflict graph.



Conflict Hypergraph

Vertices

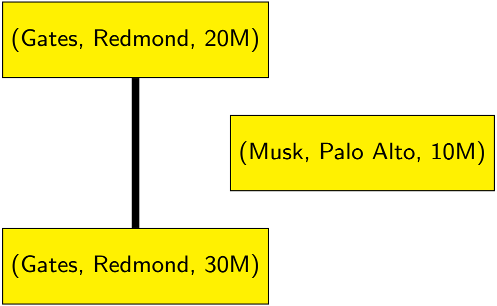
Tuples in the database.

Edges

Minimal sets of tuples violating a constraint.

Repairs

Maximal independent sets in the conflict graph.



Representation applicable only to denial constraints.

Algorithm HProver

INPUT: query Φ a disjunction of ground literals, 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 literals, 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 [CMS04])

- 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. [ABC03])
- (Greco, Greco, Zumpano [GGZ03])
- (Calì, Lembo, Rosati [CLR03b])

Specifying repairs as answer sets of logic programs

- (Arenas, Bertossi, Ch. [ABC03])
- (Greco, Greco, Zumpano [GGZ03])
- (Calì, Lembo, Rosati [CLR03b])

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. [ABC03])
- (Greco, Greco, Zumpano [GGZ03])
- (Calì, Lembo, Rosati [CLR03b])

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

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

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`, ...

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 and 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 and universal constraints
- approach unlikely to yield tractable cases

INFOMIX (Eiter et al. [EFGL03])

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

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 and universal constraints
- approach unlikely to yield tractable cases

INFOMIX (Eiter et al. [EFGL03])

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

complexity

Co-NP-completeness of CQA

Theorem (Ch., Marcinkowski [CM05a])

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 [CM05a])

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 $Q \equiv \exists x, y, z. \left(R(x, y) \wedge S(z, y)\right)$.
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 [CM05a])

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.

□

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. [ABC99])

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	co-NPC	co-NPC	
$\sigma, \pi, \times, -, \cup$	co-NPC	co-NPC	co-NPC	

- (Arenas, Bertossi, Ch. [ABC99])
- (Ch., Marcinkowski [CM05a])

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. [ABC99])
- (Ch., Marcinkowski [CM05a])
- (Fuxman, Miller [FM07])

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. [ABC99])
- (Ch., Marcinkowski [CM05a])
- (Fuxman, Miller [FM07])
- (Staworko, Ph.D., 2007), (Staworko, Ch., 2008):
  - quantifier-free queries
  - co-NPC for full TGDs and denial constraints
  - PTIME for acyclic full TGDs, join dependencies and denial constraints

Last

Dichotomy

Complexity of self-join-free conjunctive queries (Koutris, Wijsen [KW17])

- it can be decided whether or not CQA can be computed by a first-order query (and if so the corresponding SQL query is easily computable)
- computing CQA is either in PTIME or co-NP complete (and it can be decided which case applies)

The Explosion of Semantics

Tuple-based repairs

- asymmetric treatment of insertion and deletion:
  - repairs by minimal deletions only (Ch., Marcinkowski [CM05a]): data possibly incorrect but complete
  - repairs by minimal deletions and arbitrary insertions (Calì, Lembo, Rosati [CLR03a]): data possibly incorrect and incomplete
- minimal cardinality changes (Lopatenko, Bertossi [LB07]), more...

The Explosion of Semantics

Tuple-based repairs

- asymmetric treatment of insertion and deletion:
  - repairs by minimal deletions only (Ch., Marcinkowski [CM05a]): data possibly incorrect but complete
  - repairs by minimal deletions and arbitrary insertions (Calì, Lembo, Rosati [CLR03a]): data possibly incorrect and incomplete
- minimal cardinality changes (Lopatenko, Bertossi [LB07]), more...

Attribute-based repairs

- repairs of minimum cost (Bohannon et al. [BFFR05])
- checking existence of a repair of cost $< K$ NP-complete.

▸ Clean ▸ Last

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 → Dept

Dept → City

Name	*Dept*	*Location*
John	Sales	Buffalo

Name → Dept

Dept → City

Name	*Dept*	*Location*
Mary	Sales	Toronto

Name → Dept

Dept → City

Repair the lossless join decomposition:

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

Repair the lossless join 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 (Wijsen [Wij06])

Repair the lossless join 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 [AFM06])

- potential duplicates identified and grouped into clusters
- worlds $\approx$ 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 [AFM06])

- potential duplicates identified and grouped into clusters
- worlds $\approx$ 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
Musk	10M	0.5
Musk	20M	0.5

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
```

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
Musk	10M	0.5
Musk	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
Musk	10M	0.5
Musk	20M	0.5

Name → Salary

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

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
Musk	10M	0.5
Musk	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
Musk	0.5

Good News

Technology

- practical methods for CQA for subsets of SQL:
  - restricted conjunctive/aggregation queries, primary/foreign-key constraints
  - quantifier-free queries, denial constraints/acyclic TGDs/JDs
  - LP-based approaches for expressive query/constraint languages
- (slow) emergence of generic techniques
- implemented in prototype systems
- tested on medium-size databases

Good News

Technology

- practical methods for CQA for subsets of SQL:
  - restricted conjunctive/aggregation queries, primary/foreign-key constraints
  - quantifier-free queries, denial constraints/acyclic TGDs/JDs
  - LP-based approaches for expressive query/constraint languages
- (slow) emergence of generic techniques
- implemented in prototype systems
- tested on medium-size databases

The CQA Community

- over 30 active researchers
- [ABC99] has over 800 citations
- 10-15 doctoral dissertations in Europe and North America
- 2007 SIGMOD Doctoral Dissertation Award (Ariel Fuxman)
- overview papers [BC03, Ber06, Cho07, CM05b]

Taking stock

Taking stock

Topics of recent interest

- CQA under prioritized repairs
- CQA and knowledgebases
- CQA and temporal databases
- repairing: algorithms, heuristics
- repairing the database and the schema
- CQA and data exchange

Taking stock

Topics of recent interest

- CQA under prioritized repairs
- CQA and knowledgebases
- CQA and temporal databases
- repairing: algorithms, heuristics
- repairing the database and the schema
- CQA and data exchange

Open issues

- inconsistency and incompleteness
- CQA and schema matching/mapping

Taking stock

Topics of recent interest

- CQA under prioritized repairs
- CQA and knowledgebases
- CQA and temporal databases
- repairing: algorithms, heuristics
- repairing the database and the schema
- CQA and data exchange

Open issues

- inconsistency and incompleteness
- CQA and schema matching/mapping

Acknowledgments

Marcelo Arenas
Alessandro Artale
Leo Bertossi
Loreto Bravo
Andrea Calì
Àlvaro Cortés
Thomas Eiter
Wenfei Fan
Enrico Franconi
Ariel Fuxman
Gianluigi Greco
Sergio Greco
Claudio Gutierrez
Roger He
Phokion Kolaitis
Domenico Lembo
Maurizio Lenzerini
Jerzy Marcinkowski
Renée Miller
Cristian Molinaro
Vijay Raghavan
Riccardo Rosati
Gunter Saake
Jeremy Spinrad
Sławek Staworko
David Toman
Jef Wijsen

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.

P. Andritsos, A. Fuxman, and R. Miller.
Clean Answers over Dirty Databases.
In *IEEE International Conference on Data Engineering (ICDE)*, 2006.

L. Bertossi and J. Chomicki.
Query Answering in Inconsistent Databases.
In J. Chomicki, R. van der Meyden, and G. Saake, editors, *Logics for Emerging Applications of Databases*, pages 43–83. Springer-Verlag, 2003.

L. Bertossi.
Consistent Query Answering in Databases.
SIGMOD Record, 35(2), June 2006.

P. Bohannon, M. Flaster, W. Fan, and R. Rastogi.
A Cost-Based Model and Effective Heuristic for Repairing Constraints by Value Modification.

In *ACM SIGMOD International Conference on Management of Data*, pages 143–154, 2005.

J. Chomicki.
Consistent Query Answering: Five Easy Pieces.
In *International Conference on Database Theory (ICDT)*, pages 1–17. Springer, LNCS 4353, 2007.
Keynote talk.

A. Calì, 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.

A. Calì, D. Lembo, and R. Rosati.
Query Rewriting and Answering under Constraints in Data Integration Systems.
In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 16–21, 2003.

J. Chomicki and J. Marcinkowski.
Minimal-Change Integrity Maintenance Using Tuple Deletions.
Information and Computation, 197(1-2):90–121, 2005.

J. Chomicki and J. Marcinkowski.

On the Computational Complexity of Minimal-Change Integrity Maintenance in Relational Databases.
In L. Bertossi, A. Hunter, and T. Schaub, editors, *Inconsistency Tolerance*, pages 119–150. Springer-Verlag, 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, 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.

A. Fuxman 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.

W. Lipski Jr.
On Semantic Issues Connected with Incomplete Information Databases.
ACM Transactions on Database Systems, 4(3):262–296, 1979.

Paraschos Koutris and Jef Wijsen.
Consistent query answering for self-join-free conjunctive queries under primary key constraints.
ACM Trans. Database Syst., 42(2):9:1–9:45, 2017.

A. Lopatenko and L. Bertossi.
Complexity of Consistent Query Answering in Databases under Cardinality-Based and Incremental Repair Semantics.
In *International Conference on Database Theory (ICDT)*, pages 179–193. Springer, LNCS 4353, 2007.

J. Wijsen.
Project-Join Repair: An Approach to Consistent Query Answering Under Functional Dependencies.
In *International Conference on Flexible Query Answering Systems (FQAS)*, 2006.