The Complexity of Query Answering in Inconsistent Databases

Phokion G. Kolaitis

UC Santa Cruz and IBM Research - Almaden

Syntax and Semantics - INI
April 2012
In 1970, Edgar (Tedd) F. Codd introduced the relational data model.

Since that time, there has been a continuous and extensive interaction between logic and databases.

Two main uses of logic in databases:

- Logic is used as a database query language.
- Logic is used to specify integrity constraints in databases.
The Relational Data Model

- **Relational Database**
  - Collection \((R_1, \ldots, R_m)\) of finite relations (tables).
  - Relational structure \(A = (A, R_1, \ldots, R_m)\).
    In relational databases, the universe is not made explicit. Typically, one works with the active domain of the database.

- **Relational Query Languages**
  - **Relational Algebra:** Operations \(\pi, \sigma, \times, \cup, \setminus\)
  - **Relational Calculus:** (Safe) First-Order Logic
  - **SQL:** The standard commercial database query language based on relational algebra and relational calculus.
Conjunctive Queries

Definition

A conjunctive query is a query specified by a first-order formula of the form

$$\exists y_1 \cdots \exists y_m \varphi(x_1, \ldots, x_n, y_1, \ldots, y_m),$$

where $\varphi(x_1, \ldots, x_n, y_1, \ldots, y_m)$ is a conjunction of atoms.

Example

- Path-of-Length-3($x_1, x_2$):
  $$\exists y_1 \exists y_2 (E(x_1, y_1) \land E(y_1, y_2) \land E(y_2, x_2))$$

- SAME-MANAGER($x_1, x_2$):
  $$\exists y (\text{MANAGES}(y, x_1) \land \text{MANAGES}(y, x_2)).$$
Conjunctive Queries

Fact:

- Conjunctive queries are among the most frequently asked queries against databases.
- SQL provides direct support for expressing conjunctive queries via the SELECT ... FROM ... WHERE ... construct.

Example

- Given relation MANAGER(manager,employee)
- SQL expression for SAME-MANAGER:
  
  ```sql
  SELECT R.employee, S.employee
  FROM MANAGES AS R, MANAGES AS S,
  WHERE R.manager = S.manager
  ```
Integrity Constraints in Relational Databases

Extensive study of various types of integrity constraints in relational databases during the 1970s and early 1980s:

- Key constraints and functional dependencies
- Inclusion dependencies, join dependencies, multi-valued dependencies, ...

Eventually, it was realized that all these different types of dependencies can be specified in fragments of first-order logic.
Two Unifying Classes of Integrity Constraints

Definition

- **Equality-generating dependency (egd):**
  \[ \forall x (\phi(x) \rightarrow x_i = x_j), \]
  where \( \phi(x) \) is a conjunction of atoms.

  **Special Cases:** Key constraints, functional dependencies.

- **Tuple-generating dependency (tgd):**
  \[ \forall x (\phi(x) \rightarrow \exists y \psi(x, y)), \]
  where \( \phi(x) \) is a conjunction of atoms with vars. in \( x \), and \( \psi(x, y) \) is a conjunction of atoms with vars. in \( x \) and \( y \).

  **Special Cases:** LAV constraints and GAV constraints.

Note: Extensive study of egds and tgds in the context of data integration and data exchange during the past decade.
Example

- \( \forall x, y, z, w (R(x, y, z) \land R(x, y, w) \rightarrow z = w) \)
  - A key constraint written as an egd.

- \( \forall x, y (P(x, y) \rightarrow \exists z Q(z, x)) \)
  - An inclusion dependency written as a tgd.

- \( \forall x, y, z (T(x, y) \land T(y, z) \rightarrow T(x, z)) \)
  - Transitivity is specified by a tgd

- \( \forall x, y (E(x, y) \rightarrow \exists z (F(x, z) \land F(z, y))) \)
  - A tgd that “transforms” edges to paths of length 2
LAV and GAV Constraints

**Definition**

- **LAV (Local-As-View) Constraint**: A tgd of the form
  \[ \forall x (R(x) \rightarrow \exists y \psi(x, y)) \]
  where \( R(x) \) is a single atom.

- **GAV (Global-As-View) Constraint**: A tgd of the form
  \[ \forall x (\phi(x) \rightarrow T(x')) \]
  where \( T(x') \) is a single atom and \( x' \subseteq x \).
Examples of LAV and GAV Constraints

Example

(dropping universal quantifiers)

- LAV and GAV constraint:
  \[(\text{SIBLING}(x, y) \rightarrow \text{SIBLING}(y, x))\]

- LAV and inclusion dependency:
  \[(\text{PERSON}(x) \rightarrow \exists z \text{MOTHER}(z, x))\]

- LAV but not an inclusion dependency:
  \[(\text{SIBLING}(x, y) \rightarrow \exists z (\text{MOTHER}(z, x) \land \text{MOTHER}(z, y)))\]

- GAV constraint:
  \[(\text{MOTHER}(z, x) \land \text{MOTHER}(z, y) \rightarrow \text{SIBLING}(x, y))\]
In designing databases, one specifies a schema $S$ and a set of integrity constraints $\Sigma$ on $S$.

An inconsistent database is a database $I$ that does not satisfy $\Sigma$.

Inconsistent databases arise in a variety of contexts and for different reasons:

- For lack of support of particular integrity constraints.
- In data integration of heterogeneous data obeying different integrity constraints.
- In data warehousing and in Extract-Transform-Load (ETL) applications, where data has to be “cleaned” before it can be processed.
Coping with Inconsistent Databases

Two different approaches:

- **Data Cleaning**: Based on heuristics or specific domain knowledge, the inconsistent database is transformed to a consistent one by modifying (adding, deleting, updating) tuples in relations.
  - This is the prevailing approach in industry.
  - More engineering than science as quite often arbitrary choices have to be made.
Coping with Inconsistent Databases

Two different approaches:

- **Data Cleaning**: Based on heuristics or specific domain knowledge, the inconsistent database is transformed to a consistent one by modifying (adding, deleting, updating) tuples in relations.
  - This is the prevailing approach in industry.
  - More engineering than science as quite often arbitrary choices have to be made.

- **Database Repairs**: A framework for coping with inconsistent databases in a principled way and without “cleaning” dirty data first.
Definition (Arenas, Bertossi, Chomicki – 1999)

\[ \Sigma \] a set of integrity constraints and \( I \) an inconsistent database. A database \( J \) is a \textit{repair} of \( I \) w.r.t. \( \Sigma \) if

- \( J \) is a consistent database (i.e., \( J \models \Sigma \));
- \( J \) differs from \( I \) in a \textit{minimal} way.
Definition (Arenas, Bertossi, Chomicki – 1999)

$\Sigma$ a set of integrity constraints and $I$ an inconsistent database. A database $J$ is a repair of $I$ w.r.t. $\Sigma$ if

- $J$ is a consistent database (i.e., $J \models \Sigma$);
- $J$ differs from $I$ in a minimal way.

Fact

Several different types of repairs have been considered:

- Set-based repairs;
- Cardinality-based repairs;
- Attribute-based repairs.
Set-Based Repairs

**Definition**

\( \Sigma \) a set of integrity constraints and \( I \) an inconsistent database.

- \( J \) is a \( \oplus\)-repair of \( I \) w.r.t. \( \Sigma \) if \( J \models \Sigma \) and there is no \( J' \) such that \( J' \models \Sigma \) and \( I \oplus J' \subset I \oplus J \).

- \( J \) is a **subset-repair** of \( I \) w.r.t. \( \Sigma \) if \( J \) is a \( \oplus\)-repair of \( I \) such that \( J \subseteq I \).
  
  In other words, \( J \subset I \), \( J \models \Sigma \), and there is no \( J' \) such that \( J' \models \Sigma \) and \( J \subset J' \subset I \).

- \( J \) is a **superset-repair** of \( I \) w.r.t. \( \Sigma \) if \( J \) is a \( \oplus\)-repair of \( I \) such that \( I \subseteq J \).
  
  In other words, \( I \subset J \), \( J \models \Sigma \), and there is no \( J' \) such that \( J' \models \Sigma \) and \( I \subset J' \subset J \).
Example

Relation schema $R$, instance $I = \{ R(a, b), R(a, c), R(b, c) \}$

- $\Sigma = \{ \forall x \forall y \forall z ((R(x, y) \land R(x, z) \rightarrow y = z) \}$

$I$ has two $\oplus$-repairs (and subset repairs) w.r.t. $\Sigma$:

- $J_1 = \{ R(a, b), R(b, c) \}$
- $J_2 = \{ R(a, c), R(b, c) \}$. 

$\Sigma' = \{ \forall x \forall y (R(x, y) \rightarrow R(y, x)) \}$

$I$ has eight $\oplus$-repairs w.r.t. $\Sigma'$:

- $J_1 = \emptyset$ (also a subset repair)
- $J_2 = \{ R(a, b), R(b, a) \}$ (neither a subset, nor a superset repair)
- $J_3 = \{ R(a, b), R(b, a), R(a, c), R(c, a), R(b, c), R(c, b) \}$ (also a superset repair).

... Exponentially many repairs, in general.
Example

Relation schema $R$, instance $I = \{ R(a, b), R(a, c), R(b, c) \}$

- $\Sigma = \{ \forall x \forall y \forall z ((R(x, y) \land R(x, z) \rightarrow y = z) \}$
  - $I$ has two $\oplus$-repairs (and subset repairs) w.r.t. $\Sigma$:
    - $J_1 = \{ R(a, b), R(b, c) \}$
    - $J_2 = \{ R(a, c), R(b, c) \}$.

- $\Sigma' = \{ \forall x \forall y (R(x, y) \rightarrow R(y, x)) \}$
  - $I$ has eight $\oplus$-repairs w.r.t. $\Sigma'$:
    - $J_1 = \emptyset$ (also a subset repair)
    - $J_2 = \{ R(a, b), R(b, a) \}$
      (neither a subset, nor a superset repair)
    - $J_3 = \{ R(a, b), R(b, a), R(a, c), R(c, a), R(b, c), R(c, b) \}$
      (also a superset repair).
    - $\ldots$

Exponentially many repairs, in general.
Query Answering over Inconsistent Databases

Definition (Arenas, Bertossi, Chomicki)

Let $\Sigma$ be a set of integrity constraints, $q$ a query, $I$ an instance, and $\ast \in \{\oplus, \text{subset}, \text{superset}\}$. The $\ast$-consistent answers of $q$ on $I$ w.r.t. $\Sigma$ is the set

$$\ast\text{-Con}(q, I, \Sigma) = \bigcap \{ q(J) : J \text{ is a } \ast\text{-repair of } I \text{ w.r.t. } \Sigma \}.$$  

Note

- The motivation comes from the semantics of queries in the context of incomplete information and possible worlds.
- The $\ast$-consistent answers of $q$ in $I$ are theCertain answers of $q$ on $I$, when the set of all possible worlds is the set of all $\ast$-repairs of $I$ w.r.t. $\Sigma$.  

Example (Revisited)

Relation schema $R$, instance $I = \{ R(a, b), R(a, c), R(b, c) \}$,

$$\Sigma = \{ \forall x \forall y \forall z ((R(x, y) \land R(x, z) \rightarrow y = z) \}$$

Recall that $I$ has two $\oplus$-repairs (and subset repairs) w.r.t. $\Sigma$:

$$J_1 = \{ R(a, b), R(b, c) \} \text{ and } J_2 = \{ R(a, c), R(b, c) \}.$$ 

- If $q(x)$ is the query $\exists y E(x, y)$, then
  $$\oplus\text{-Con}(q, I, \Sigma) = \{ a, b \}.$$ 

- If $q(x)$ is the query $\exists z E(z, x)$, then
  $$\oplus\text{-Con}(q, I, \Sigma) = \{ c \}.$$
Consistent Query Answering (CQA)

Main themes in investigation of CQA so far.

- **Complexity of CQA for conjunctive queries:**
  From polynomial-time computability to undecidability.

- **Prototype Systems:**
  - Hippo (Chomicki, Marcinkowski, Staworko - 2004)
  - ConQuer (Fuxman - 2007)
  - ConEx (Caniupan, Bertossi - 2010).

**Note**

For an overview, see the monograph *Database Repairing and Consistent Query Answering* by L. Bertossi, 2011.
**Data Complexity of CQA**

**Definition**

- Every fixed set $\Sigma$ of constraints and every fixed conjunctive query $q$ give rise to the following computational problem:
  
  **Input:** Instance $I$.
  **Output:** $\ast$-Con$(q, I, \Sigma)$, where $\ast \in \{\oplus, \text{subset}, \text{superset}\}$.

- Let $\mathcal{L}$ be a class of constraints and $\mathcal{C}$ a complexity class.

  1. We say that the data complexity of $\ast$-CQA w.r.t. $\mathcal{L}$ is in $\mathcal{C}$ if $\ast$-Con$(q, I, \Sigma)$ is in $\mathcal{C}$, for every finite subset $\Sigma$ of $\mathcal{L}$ and every conjunctive query $q$.

  2. We say that the data complexity of $\ast$-CQA w.r.t. $\mathcal{L}$ is $\mathcal{C}$-complete if it is in $\mathcal{C}$ and there are a finite subset $\Sigma$ of $\mathcal{L}$ and a conjunctive query $q$ such that $\ast$-Con$(q, I, \Sigma)$ is $\mathcal{C}$-complete.
## Data Complexity of CQA for Conjunctive Queries

<table>
<thead>
<tr>
<th>Constraints</th>
<th>Superset-CQA</th>
<th>Subset-CQA</th>
<th>⊕-CQA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keys/egds</td>
<td>coNP-comp.</td>
<td>coNP-comp</td>
<td>coNP-comp.</td>
</tr>
<tr>
<td>LAV</td>
<td>PTIME</td>
<td>PTIME</td>
<td>PTIME</td>
</tr>
<tr>
<td>GAV†</td>
<td>PTIME</td>
<td>coNP-comp.</td>
<td>coNP-comp.</td>
</tr>
<tr>
<td>Weakly acyclic sets of tgds†</td>
<td>PTIME</td>
<td>Π₂-comp.</td>
<td>Π₂-comp.</td>
</tr>
<tr>
<td>Arbitrary tgds†</td>
<td>Undecidable</td>
<td>Π₂-comp.</td>
<td>Undecidable</td>
</tr>
</tbody>
</table>

### Note:
- † indicates that the upper bound holds also in the presence of egds.
- Entries in blue indicate that this is a result in a recent paper by ten Cate, Fontaine, K ... 2012.
The Complexity of CQA

- The preceding results give a fairly comprehensive picture of the complexity of CQA for conjunctive queries and w.r.t. various types of constraints.

- However, these results fall short from yielding a complete classification of the complexity of CQA.

- The ultimate goal of this investigation is to prove or disprove that dichotomy and/or trichotomy theorems hold for the complexity of CQA.
Ladner’s Theorem and Dichotomies

Theorem (Ladner 1975): If $P \neq NP$, then there is a decision problem $Q$ such that

1. $Q$ is in NP, but not in P.
2. $Q$ is not NP-complete.

Definition

We say that a *dichotomy theorem* holds for a collection $\mathcal{F}$ of decision problems in NP if for every problem $Q$ in $\mathcal{F}$, one of the following holds:

1. $Q$ is in P.
2. $Q$ is NP-complete.
Dichotomy Theorems and Conjectures

Dichotomy theorems tend to be rare and difficult to establish.

- Schaefer’s Dichotomy Theorem for Boolean Satisfiability (1978).
- Fortune-Hopcroft-Wyllie Dichotomy Theorem for the Directed Graph Homeomorphism Problem (1980).

...
Dichotomy Conjecture for CQA and Key Constraints

Conjecture

If $\Sigma$ is a set of key constraints and $q$ is a conjunctive query, then one of the following holds:

1. subset-$\text{Con}(q, I, \Sigma)$ is in P.
2. subset-$\text{Con}(q, I, \Sigma)$ is coNP-complete.

Note:

- To date, little progress has been made towards resolving this dichotomy conjecture.
- Conditions sufficient for tractability and also conditions sufficient for intractability have found.
- However, there is still a big gap remaining to be filled.
Binary relations $R$ and $S$ having the first attribute as key, i.e.,

$$\Sigma = \{ R(u, v) \land R(u, w) \rightarrow v = w, \ S(u, v) \land S(u, w) \rightarrow v = w \}.$$

- Let $q_1$ be the Boolean query $\exists x, y, z (R(x, y) \land S(y, z))$.
- Let $q_2$ be the Boolean query $\exists x, y (R(x, y) \land S(y, x))$.
- Let $q_3$ be the Boolean query $\exists x, y, z (R(x, y) \land S(z, y))$.

**Question:**
What can we say about subset-\(\text{Con}(q_i, I, \Sigma)\), where $i = 1, 2, 3$?
Binary relations $R$ and $S$ having the first attribute as key, i.e.,

$$\Sigma = \{ R(u, v) \land R(u, w) \rightarrow v = w, \ S(u, v) \land S(u, w) \rightarrow v = w \}.$$ 

- Let $q_1$ be the query $\exists x, y, z (R(x, y) \land S(y, z))$.
  Then subset-Con($q_1, I, \Sigma$) is in P; in fact, subset-Con($q_1, I, \Sigma$) is first-order rewritable.

- Let $q_2$ be the query $\exists x, y (R(x, y) \land S(y, x))$.
  Then subset-Con($q_2, I, \Sigma$) is in P; however, subset-Con($q_2, I, \Sigma$) is not first-order rewritable.

- Let $q_3$ be the query $\exists x, y, z (R(x, y) \land S(z, y))$.
  Then subset-Con($q_3, I, \Sigma$) is coNP-complete.
Dichotomy Conjecture for CQA and Key Constraints

A very modest first step towards the Dichotomy Conjecture.

Theorem (K ... and Pema - 2012). Let $q$ be a conjunctive query with exactly two atoms involving different relations $R$ and $S$, and let $\Sigma$ be a set consisting of a key constraint on $R$ and a key constraint on $S$. Then one of the following holds:

1. $\text{subset-Con}(q, I, \Sigma)$ is in $P$.
2. $\text{subset-Con}(q, I, \Sigma)$ is coNP-complete.

Moreover, there is an effective criterion for determining which of the two holds.

Note:
The tractable case uses Minty’s polynomial-time algorithm for the maximal independent set on claw-free graphs.
Prove or disprove the following Trichotomy Conjecture.

Conjecture

Let \( \Sigma \) be a weakly acyclic set of tgds and let \( q \) be a conjunctive query. Then one of the following holds:

1. \( \oplus\text{-Con}(q, I, \Sigma) \) is in \( P \).
2. \( \oplus\text{-Con}(q, I, \Sigma) \) is coNP-complete.
3. \( \oplus\text{-Con}(q, I, \Sigma) \) is \( \Pi^p_2 \)-complete.