First-Order Under-Approximations of Consistent Query Answers

DBDBD 2015, Amsterdam

Floris Geerts Fabian Pijcke Jef Wijsen

Dept. of Computer Science — University of Mons
Dept. of Mathematics and Computer Science — University of Antwerp

Uncertain Database

Definition (Uncertain Database and Repair)

An **uncertain database** is a database in which primary keys can be violated.
A **repair** of an uncertain database is any maximal consistent subset.

Example

ManagedBy

Dept	*Mgr*	*Budget*
CIA	Barack	60M
MI6	James	15M

WorksFor

Agent	*Dept*
James	CIA
James	MI6

The uncertainty about James' department gives rise to two repairs:
one with *WorksFor*(James, CIA), another with *WorksFor*(James, MI6).

Certain Query Answering

Definition

The **certain answer** to a query q on an uncertain database $\mathbf{db}$ is defined by:

$$\bigcap\{q(\mathbf{rep}) \mid \mathbf{rep} \text{ is a repair of } \mathbf{db}\}.$$

Intuitively, an answer is certain if it holds true in every repair.

We write $\lfloor q \rfloor$ for the query that takes in an uncertain database $\mathbf{db}$, and returns the certain answer, i.e.,

$$\lfloor q \rfloor(\mathbf{db}) := \bigcap\{q(\mathbf{rep}) \mid \mathbf{rep} \text{ is a repair of } \mathbf{db}\}.$$

Certain Query Answering: Example

- Let **db** be the following uncertain database:

ManagedBy

Dept	*Mgr*	*Budget*
CIA	Barack	60M
MI6	James	15M

WorksFor

Agent	*Dept*
James	CIA
James	MI6

- Let
  - $\mathbf{rep}_1$ be the repair with *WorksFor*(James, CIA), and
  - $\mathbf{rep}_2$ be the repair with *WorksFor*(James, MI6).

Certain Query Answering: Example

- Let **db** be the following uncertain database:

ManagedBy	*Dept*	*Mgr*	*Budget*
	CIA	Barack	60M
	MI6	James	15M

WorksFor	*Agent*	*Dept*
	James	CIA
	James	MI6

- Let
  - $\mathbf{rep}_1$ be the repair with *WorksFor*(James, CIA), and
  - $\mathbf{rep}_2$ be the repair with *WorksFor*(James, MI6).
- Let q_0 be the query *"Which departments are self-managed, i.e., managed by one of its agents?"*

$$q_0 = \{d \mid \exists m \exists b \,(ManagedBy(\underline{d}, m, b) \wedge WorksFor(\underline{m}, d))\}.$$

$$\begin{aligned} \lfloor q_0 \rfloor (\mathbf{db}) &= q_0(\mathbf{rep}_1) \cap q_0(\mathbf{rep}_2) \\ &= \{\} \cap \{\text{MI6}\} \\ &= \{\} \end{aligned}$$

Data Complexity

- The focus of this paper is on computing certain answers to self-join-free conjunctive queries q, for which three possibilities can occur:

Data Complexity

- The focus of this paper is on computing certain answers to self-join-free conjunctive queries q, for which three possibilities can occur:
  - A $\lfloor q \rfloor$ can be expressed in relational calculus (the "ideal" case);

Data Complexity

- The focus of this paper is on computing certain answers to self-join-free conjunctive queries q, for which three possibilities can occur:
  - A $\lfloor q \rfloor$ can be expressed in relational calculus (the "ideal" case);
  - B $\lfloor q \rfloor$ **cannot** be expressed in relational calculus, but can be computed by a polynomial-time algorithm; or

Data Complexity

- The focus of this paper is on computing certain answers to self-join-free conjunctive queries q, for which three possibilities can occur:
  - A $\lfloor q \rfloor$ can be expressed in relational calculus (the "ideal" case);
  - B $\lfloor q \rfloor$ **cannot** be expressed in relational calculus, but can be computed by a polynomial-time algorithm; or
  - C $\lfloor q \rfloor$ cannot even be computed by a polynomial-time algorithm (unless $\mathbf{P} = \mathbf{NP}$).

Data Complexity

- The focus of this paper is on computing certain answers to self-join-free conjunctive queries q, for which three possibilities can occur:
  - A $\lfloor q \rfloor$ can be expressed in relational calculus (the "ideal" case);
  - B $\lfloor q \rfloor$ **cannot** be expressed in relational calculus, but can be computed by a polynomial-time algorithm; or
  - C $\lfloor q \rfloor$ cannot even be computed by a polynomial-time algorithm (unless $\mathbf{P} = \mathbf{NP}$).
- Recall: a self-join-free conjunctive query q is a relational calculus query of the form:

$$\{\vec{x} \mid \exists \vec{y} \, (R_1(\vec{z}_1) \wedge \cdots \wedge R_\ell(\vec{z}_\ell))\},$$

in which $i \neq j$ implies $R_i \neq R_j$.

Examples

Case A: $\lfloor q \rfloor$ in relational calculus *"Who is the manager of CIA?"*:

$$q_0 = \{m \mid \exists b\,(ManagedBy(\underline{\mathrm{CIA}}, m, b))\}.$$

$\lfloor q_0 \rfloor$ **can** be expressed in relational calculus, as follows:

$$\begin{aligned}\lfloor q_0 \rfloor = \{m \mid\ & \exists b\,(ManagedBy(\underline{\mathrm{CIA}}, m, b) \\ & \wedge \forall m' \forall b'\,(ManagedBy(\underline{\mathrm{CIA}}, m', b') \rightarrow m' = m))\}\end{aligned}$$

Examples

Case A: $\lfloor q \rfloor$ in relational calculus *"Who is the manager of CIA?"*:

$$q_0 = \{m \mid \exists b\,(ManagedBy(\underline{\mathrm{CIA}}, m, b))\}.$$

$\lfloor q_0 \rfloor$ **can** be expressed in relational calculus, as follows:

$$\begin{aligned}\lfloor q_0 \rfloor = \{m \mid\ & \exists b\,(ManagedBy(\underline{\mathrm{CIA}}, m, b) \\ & \wedge \forall m' \forall b'\,(ManagedBy(\underline{\mathrm{CIA}}, m', b') \rightarrow m' = m))\}\end{aligned}$$

Case B: $\lfloor q \rfloor$ in **P**, but not expressible in relational calculus *"Get budgets of self-managed departments"*:

$$q_0 = \{b \mid \exists d \exists m\,(ManagedBy(\underline{d}, m, b) \wedge WorksFor(\underline{m}, d))\}.$$

Examples

Case A: $\lfloor q \rfloor$ in relational calculus *"Who is the manager of CIA?"*:

$$q_0 = \{m \mid \exists b\,(ManagedBy(\underline{\mathrm{CIA}}, m, b))\}.$$

$\lfloor q_0 \rfloor$ **can** be expressed in relational calculus, as follows:

$$\begin{aligned}\lfloor q_0 \rfloor = \{m \mid\ & \exists b\,(ManagedBy(\underline{\mathrm{CIA}}, m, b) \\ & \wedge \forall m' \forall b'\,(ManagedBy(\underline{\mathrm{CIA}}, m', b') \rightarrow m' = m))\}\end{aligned}$$

Case B: $\lfloor q \rfloor$ in **P**, but not expressible in relational calculus *"Get budgets of self-managed departments"*:

$$q_0 = \{b \mid \exists d \exists m\,(ManagedBy(\underline{d}, m, b) \wedge WorksFor(\underline{m}, d))\}.$$

Case C: $\lfloor q \rfloor$ is **coNP**-hard Example in the paper.

Research Question

- Since RDBMSs cope well with relational calculus (in the form of SQL), it is easy to handle the case where $\lfloor q \rfloor$ is expressible in relational calculus (case A).
- But what if $\lfloor q \rfloor$ is not expressible in relational calculus (cases B and C)?

Research Question

- Since RDBMSs cope well with relational calculus (in the form of SQL), it is easy to handle the case where $\lfloor q \rfloor$ is expressible in relational calculus (case A).
- But what if $\lfloor q \rfloor$ is not expressible in relational calculus (cases B and C)?
- Find a relational calculus query φ (the greater with respect to $\subseteq$, the better) such that

  Under-Approximation: $\varphi \subseteq \lfloor q \rfloor$; and

  First-Order Postprocessig: φ is a first-order combination (using $\wedge$, $\vee$, $\neg$, $\exists$, $\forall$) of queries of the form $\lfloor q_i \rfloor$, where q_i is self-join-free conjunctive and $\lfloor q_i \rfloor$ can be expressed in relational calculus (as in case A).

  Such query φ is called a strategy for $\lfloor q \rfloor$.

Practical Setting

1. Restricted query interface to an inconsistent database **db**:

   You can only ask self-join-free conjunctive queries q!

2. Moreover, the interface only returns consistent answers computable in relational calculus:

   If $\lfloor q \rfloor$ cannot be expressed in relational calculus, then your query q is rejected;
   otherwise the answer $\lfloor q \rfloor(\mathbf{db})$ will be returned.

3. Assume that your query q is rejected. How will you proceed?

   Find queries $q_1, \ldots, q_\ell$, each accepted by the interface, and a relational calculus query φ such that
   $\varphi(\lfloor q_1 \rfloor(\mathbf{db}), \ldots, \lfloor q_\ell \rfloor(\mathbf{db}))$ is a "large" subset of $\lfloor q \rfloor(\mathbf{db})$.

Intuitively, the strategy φ does some first-order postprocessing on answers obtained from the interface.

Optimality of Strategies

Let q be a self-join-free conjunctive query q such that $\lfloor q \rfloor$ is not expressible in relational calculus.

- Obviously, there exists no strategy φ such that $\varphi \equiv \lfloor q \rfloor$, because φ is a relational calculus query, but $\lfloor q \rfloor$ cannot be expressed in relational calculus.
- Obviously, strategies are **closed under union**: if φ_1 and φ_2 are strategies, then $\varphi_1 \cup \varphi_2$ is a strategy.
  If neither of φ_1 or φ_2 is contained in the other, then $\varphi_1 \cup \varphi_2$ is a better strategy than φ_1 (and than φ_2).
- A strategy φ for $\lfloor q \rfloor$ is called optimal if for every other strategy φ', we have $\varphi' \subseteq \varphi \subseteq \lfloor q \rfloor$.

Example

1. *"Get budgets of self-managed departments"*:

$$q_0 = \{b \mid \exists d \exists m \, (\mathit{ManagedBy}(\underline{d}, m, b) \land \mathit{WorksFor}(\underline{m}, d))\}.$$

$\lfloor q_0 \rfloor$ **cannot** be expressed in relational calculus!!!

2. *"Get budgets of self-managed departments managed by Barack (or James)"*:

$$\begin{aligned} q_1 &= \{b \mid \exists d \, (\mathit{ManagedBy}(\underline{d}, \text{'Barack'}, b) \land \mathit{WorksFor}(\underline{\text{'Barack'}}, d))\} \\ q_2 &= \{b \mid \exists d \, (\mathit{ManagedBy}(\underline{d}, \text{'James'}, b) \land \mathit{WorksFor}(\underline{\text{'James'}}, d))\} \end{aligned}$$

$\lfloor q_1 \rfloor$ and $\lfloor q_2 \rfloor$ **can** be expressed in relational calculus!!!

3. Then, the following query is a strategy for $\lfloor q_0 \rfloor$:

$$\lfloor q_1 \rfloor \cup \lfloor q_2 \rfloor \, .$$

This strategy is **not optimal** (since we can add a query for, e.g., 'Sherlock').

Example (Continued)

$q_0 =$ *"Get budgets of self-managed departments"*:

1. *"Get self-managed departments together with their budget"*:

$$q_3 = \{d, b \mid \exists m \,(ManagedBy(\underline{d}, m, b) \wedge WorksFor(\underline{m}, d))\}.$$

"Get manager and budget of self-managed departments":

$$q_4 = \{m, b \mid \exists d \,(ManagedBy(\underline{d}, m, b) \wedge WorksFor(\underline{m}, d))\}.$$

$\lfloor q_3 \rfloor$ and $\lfloor q_4 \rfloor$ **can** be expressed in relational calculus!!!

2. Then, the following query is a strategy for $\lfloor q_0 \rfloor$:

$$\exists d \,(\, \lfloor q_3(d, b) \rfloor \,) \cup \exists m \,(\, \lfloor q_4(m, b) \rfloor \,)\,.$$

This strategy is strictly better than the strategy on the previous slide (it does not rely on constants like Barack, James, Sherlock).

Contribution

1. We show how to build, given a self-join-free conjunctive query q, a strategy φ for $\lfloor q \rfloor$ of the **syntactic form**

$$\exists \vec{x}_1 \left(\lfloor q_1 \rfloor \right) \cup \cdots \cup \exists \vec{x}_\ell \left(\lfloor q_\ell \rfloor \right).$$

That is, postprocessing is limited to $\exists$ and $\cup$.

Contribution

1. We show how to build, given a self-join-free conjunctive query q, a strategy φ for $\lfloor q \rfloor$ of the **syntactic form**

$$\exists \vec{x}_1 \, (\, \lfloor q_1 \rfloor \,) \cup \cdots \cup \exists \vec{x}_\ell \, (\, \lfloor q_\ell \rfloor \,) \, .$$

   That is, postprocessing is limited to $\exists$ and $\cup$.

2. We show that our strategy is optimal in some weak sense: for every other strategy φ' **of the same syntactic form**, we have $\varphi' \subseteq \varphi \subseteq \lfloor q \rfloor$.

Contribution

1. We show how to build, given a self-join-free conjunctive query q, a strategy φ for $\lfloor q \rfloor$ of the **syntactic form**

$$\exists \vec{x}_1 \left(\lfloor q_1 \rfloor \right) \cup \cdots \cup \exists \vec{x}_\ell \left(\lfloor q_\ell \rfloor \right).$$

   That is, postprocessing is limited to $\exists$ and $\cup$.

2. We show that our strategy is optimal in some weak sense: for every other strategy φ' **of the same syntactic form**, we have $\varphi' \subseteq \varphi \subseteq \lfloor q \rfloor$.

3. Open question: Is it possible to improve strategies by using negation in postprocessing?

Conclusion

1. We have proposed a new framework for divulging an inconsistent database to end users, which adopts two postulates:

Conclusion

1. We have proposed a new framework for divulging an inconsistent database to end users, which adopts two postulates:
   - never divulge inconsistencies to end users; and

Conclusion

1. We have proposed a new framework for divulging an inconsistent database to end users, which adopts two postulates:
   - never divulge inconsistencies to end users; and
   - the data complexity of queries must remain tractable (and even within relational calculus in this paper).

Conclusion

1. We have proposed a new framework for divulging an inconsistent database to end users, which adopts two postulates:
   - never divulge inconsistencies to end users; and
   - the data complexity of queries must remain tractable (and even within relational calculus in this paper).
2. The notion of strategy captures how end users can obtain certain answers under such access postulates.

Conclusion

1. We have proposed a new framework for divulging an inconsistent database to end users, which adopts two postulates:
   - never divulge inconsistencies to end users; and
   - the data complexity of queries must remain tractable (and even within relational calculus in this paper).
2. The notion of strategy captures how end users can obtain certain answers under such access postulates.
3. We show how to build strategies of a syntactically restricted form.

Conclusion

1. We have proposed a new framework for divulging an inconsistent database to end users, which adopts two postulates:
   - never divulge inconsistencies to end users; and
   - the data complexity of queries must remain tractable (and even within relational calculus in this paper).
2. The notion of strategy captures how end users can obtain certain answers under such access postulates.
3. We show how to build strategies of a syntactically restricted form.
4. Challenging open question: Is it possible to find better strategies of a more general syntactic form?