# Consistent Query Answering via SAT Solving

Phokion G. Kolaitis

UC Santa Cruz and IBM Research

*joint work with*

Akhil A. Dixit

Google

# Roadmap

- Relational databases, conjunctive queries, integrity constraints
- Inconsistent databases, repairs, consistent answers
- Complexity of consistent answers to conjunctive queries
- Aggregation queries, range consistent answers
- CAvSAT: Consistent query answering via SAT solving
- Experimental evaluation of CAvSAT

# The Relational Data Model

- Relational Database
  - Collection $\mathcal{I} = (R_1, \ldots, R_m)$ of finite relations (tables).
  - Relational structure $\mathbf{A} = (A, R_1, \ldots, R_m)$.

    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 $\cup$, $\setminus$, $\times$, $\pi$, $\sigma$
  - Relational Calculus: (Safe) First-Order Logic
  - SQL: The industry-standard query language based on relational algebra and relational calculus.

# Conjunctive Queries

## Definition

A conjunctive query (CQ) is specified by a FO-formula

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

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

## Example

- $\textsc{Path-of-Length-3}(x_1, x_2)$:
  $$\exists y_1 \exists y_2 (E(x_1, y_1) \wedge E(y_1, y_2) \wedge E(y_2, x_2))$$
- $\textsc{Taught-By}(x_1, x_2)$:
  $$\exists y (\textsc{Enrolls}(x_1, y) \wedge \textsc{Teaches}(x_2, y)).$$

# Conjunctive Queries

## Fact

- CQs are among the most frequently asked queries.
- SQL provides direct support for expressing CQs via the `SELECT` ... `FROM` ... `WHERE` ... construct.

## Example

- ENROLLS(student,course), TEACHES(professor,course)

  TAUGHT-BY($x_1, x_2$): $\exists y($ Enrolls$(x_1, y) \wedge$ Teaches$(x_2, y))$

- SQL expression for TAUGHT-BY:

```
SELECT ENROLLS.student, TEACHES.professor
FROM   ENROLLS, TEACHES
WHERE  ENROLLS.course = TEACHES.course
```

# Boolean Conjunctive Queries

**Definition**

A Boolean CQ is a CQ with no free variables:

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

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

**Example**

- $\exists x, y, z(E(x, y) \wedge E(y, z) \wedge E(z, x))$
  (“there is a triangle”)
- $\exists x, y, z(R(x, y) \wedge T(x, z))$
  ("there is a node that has an $R$-neighbor and a $T$-neighbor")

# The Conjunctive Query Evaluation Problem

Definition [CONJUNCTIVE QUERY EVALUATION - CQE]
Given a database $\mathcal{I}$ and a Boolean CQ $q$, does $\mathcal{I} \models q$?
(i.e., is $q$ true on $\mathcal{I}$?)

# The Conjunctive Query Evaluation Problem

Definition [CONJUNCTIVE QUERY EVALUATION - CQE]
Given a database $\mathcal{I}$ and a Boolean CQ $q$, does $\mathcal{I} \models q$?
(i.e., is $q$ true on $\mathcal{I}$?)

Fact SAT is a special case of CQE.

Example The following statements are equivalent:

1. $(P \vee Q \vee T) \wedge (\neg P \vee Q \vee T) \wedge (\neg P \vee \neg Q \vee T)$ is satisfiable.
2. $\mathcal{I} \models \exists x, y, z(R_0(x, y, z) \wedge R_1(x, y, z) \wedge R_2(x, y, z))$, where
   - $\mathcal{I} = (R_0, R_1, R_2)$,
   - $R_0 = \{(0, 1)\}^3 \setminus \{(0, 0, 0)\}$,
   - $R_1 = \{(0, 1)\}^3 \setminus \{(1, 0, 0)\}$,
   - $R_2 = \{(0, 1)\}^3 \setminus \{(1, 1, 0)\}$.

# The Difference between SAT and CQE

Data Complexity: In practice, the query is typically fixed, only the database varies.

- If $q$ is a Boolean CQ, then CQE($q$) asks:
  Given a database $\mathcal{I}$, does $\mathcal{I} \models q$?
- Fact: CQE($q$) is in $\mathrm{L}$, for every Boolean CQ $q$.
- The Data Complexity of CQE is in $\mathrm{L}$.

Combined Complexity: In SAT (viewed as a CQE problem), both the query and the database vary.

- The Combined Complexity of CQE is $\mathrm{NP}$-complete.

# Integrity Constraints in Databases

**Definition** $R$ a relation schema, $X$ and $Y$ sets of attributes

- **Functional Dependency** $R : X \rightarrow Y$
  If two tuples in $R$ agree on $X$, then they agree on $Y$.
- **Key Constraint** $R : X \rightarrow Y$, where $Y = Attr(R) \setminus X$.

**Example** $R(A, B, C, D)$

- **Functional Dependency** $R : A, B \rightarrow D$:

$$\forall a, b, c, c', d, d' (R(a, b, c, d) \wedge R(a, b, c', d') \rightarrow d = d')$$

- **Key Constraint** $R : A, B \rightarrow C, D$:

$$\forall a, b, c, c', d, d' (R(a, b, c, d) \wedge R(a, b, c', d') \rightarrow c = c' \wedge d = d')$$

# Inconsistent Databases

- When designing databases, a schema **S** and a set $\Sigma$ of integrity constraints on **S** are specified.
- An inconsistent database is a database $\mathcal{I}$ that does not satisfy $\Sigma$.
- Inconsistent databases arise in a variety of contexts and for different reasons, including:
  - Lack of support of particular integrity constraints.
  - Integration of heterogeneous data residing in different sources and obeying different integrity constraints.

Question: How to cope with inconsistent databases?

# Two Approaches for Coping with Inconsistency

- Data Cleaning: Based on heuristics or specific domain knowledge, the inconsistent database is transformed to a consistent one by modifying tuples in relations.
  - Data cleaning is the main approach in industry.
  - More engineering than science due to arbitrary choices.

# Two Approaches for Coping with Inconsistency

- Data Cleaning: Based on heuristics or specific domain knowledge, the inconsistent database is transformed to a consistent one by modifying tuples in relations.
  - Data cleaning is the main approach in industry.
  - More engineering than science due to arbitrary choices.

- Database Repairs: A framework for coping with inconsistent databases without "cleaning" dirty data first.
  - Extensive study in academia.
  - A more principled approach.

# Database Repairs

### Definition (Arenas, Bertossi, Chomicki – 1999)

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

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

# Database Repairs

## Definition (Arenas, Bertossi, Chomicki – 1999)

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

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

## Definition

$\Sigma$ a set of integrity constraints and $\mathcal{I}$ an inconsistent database.
A database $\mathcal{J}$ is a *subset-repair* of $\mathcal{I}$ w.r.t. $\Sigma$ if

- $\mathcal{J} \subset \mathcal{I}$
- $\mathcal{J} \models \Sigma$ (i.e., $\mathcal{J}$ is consistent)
- there is no $\mathcal{J}'$ such that $\mathcal{J}' \models \Sigma$ and $\mathcal{J} \subset \mathcal{J}' \subset \mathcal{I}$.

Note: From now on, the term repair means subset repair.

# Example of Repairs

- Schema consists of a binary relation symbol $R$.
- Key constraint
  $\Sigma = \{\forall x \forall y \forall ((R(x,y) \wedge R(x,z) \rightarrow y = z)\}$
- Database
  $\mathcal{I} = \{R(a_1, b_1), R(a_1, b_2), R(a_2, b_1), R(a_2, b_2)\}$
- Repairs
  $\mathcal{I}$ has four (subset) repairs w.r.t. $\Sigma$:
  - $\mathcal{J}_1 = \{R(a_1, b_1), R(a_2, b_1)\}$
  - $\mathcal{J}_2 = \{R(a_1, b_1), R(a_2, b_2)\}$
  - $\mathcal{J}_3 = \{R(a_1, b_2), R(a_2, b_1)\}$
  - $\mathcal{J}_4 = \{R(a_1, b_2), R(a_2, b_2)\}$.

  Exponentially many repairs, in general.

# Consistent Query Answering (CQA)

## Definition (Arenas, Bertossi, Chomicki - 1999)

$\Sigma$ a set of integrity constraints, $q$ a query, and $I$ a database.

The *consistent answers of $q$ on $\mathcal{I}$ w.r.t. $\Sigma$* is the set

$$\textsc{Cons}(q, \mathcal{I}, \Sigma) = \bigcap \{q(\mathcal{J}) : \mathcal{J} \text{ is a repair of } \mathcal{I} \text{ w.r.t. } \Sigma\}.$$

Note:

- The motivation comes from the semantics of queries in the context of incomplete information and possible worlds.
- A consistent answers is guaranteed to be found in the evaluation of the query $q$ on *every* repair of the inconsistent database $\mathcal{I}$.

# Consistent Query Answering

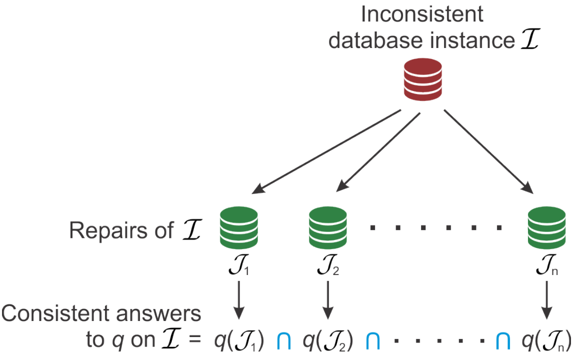



Figure: Consistent Answers

# Example of Consistent Query Answering

- $\Sigma = \{\forall x \forall y \forall z ((R(x,y) \wedge R(x,z) \rightarrow y = z)\}$
- $\mathcal{I} = \{R(a_1,b_1), R(a_1,b_2), R(a_2,b_1), R(a_2,b_2)\}$
- Recall that $\mathcal{I}$ has four repairs w.r.t. $\Sigma$:
  - $\mathcal{J}_1 = \{R(a_1,b_1), R(a_2,b_1)\}$, $\mathcal{J}_2 = \{R(a_1,b_1), R(a_2,b_2)\}$
  - $\mathcal{J}_3 = \{R(a_1,b_2), R(a_2,b_1)\}$, $\mathcal{J}_4 = \{R(a_1,b_2), R(a_2,b_2)\}$.
- If $q(x)$ is the query $\exists y R(x,y)$, then

$$\textsc{Cons}(q,\mathcal{I},\Sigma) = \{a_1, a_2\}.$$

- If $q(x)$ is the query $\exists z R(z,x)$, then

$$\textsc{Cons}(q,\mathcal{I},\Sigma) = \emptyset.$$

# Overview of Research on Database Repairs

Main themes explored so far:

- Complexity of Consistent Query Answering
- Prototype Systems for Consistent Query Answering

# Complexity of CQA: A "Simple" Case Study

Assume that

- $\Sigma$ is a set of key constraints with one key per relation.
- $q$ is a Boolean conjunctive query (no free variables).

Definition: $\textsc{Certainty}(q, \Sigma)$ is the following decision problem:
Given a database $\mathcal{I}$, is $\textsc{Cons}(q, \mathcal{I}, \Sigma)$ true?
(i.e., is $q$ true on every repair $\mathcal{J}$ of $\mathcal{I}$?)

Question: What is the complexity of $\textsc{Certainty}(q, \Sigma)$?

Easy Fact: $\textsc{Certainty}(q, \Sigma)$ is in $\mathrm{coNP}$.

Reason: Repair-checking is in $\mathrm{P}$.

# Complexity of CQA: An Illustration

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 PATH be the Boolean query $\exists x, y, z(R(x,y) \land S(y,z))$.
- Let CYCLE be the Boolean query $\exists x, y(R(x,y) \land S(y,x))$.
- Let SINK be the Boolean query $\exists x, y, z(R(x,y) \land S(z,y))$.

Question:
What can we say about CERTAINTY$(q, \Sigma)$, where $q$ is one of these three queries?

# Complexity of CQA: An Illustration

- Let PATH be the query $\exists x, y, z(R(x, y) \wedge S(y, z))$.

  CERTAINTY(PATH, $\Sigma$) is in P; in fact, it is FO-rewritable as

  $\exists x, y, z(R(x, y) \wedge S(y, z) \wedge \forall y'(R(x, y') \rightarrow \exists z' S(y', z')))$.

  (Fuxman and Miller - 2007)

- Let CYCLE be the query $\exists x, y(R(x, y) \wedge S(y, x))$.

  CERTAINTY(CYCLE, $\Sigma$) is in P, but it is not FO-rewritable.

  (Wijsen - 2010)

- Let SINK be the query $\exists x, y, z(R(x, y) \wedge S(z, y))$.

  CERTAINTY(SINK, $\Sigma$) is coNP-complete.

  (Fuxman and Miller - 2007)

# Classifying the Complexity of CQA

### Conjecture (Trichotomy Conjecture for CERTAINTY$(q, \Sigma)$)

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

- CERTAINTY$(q, \Sigma)$ is FO-rewritable.
- CERTAINTY$(q, \Sigma)$ is in $\mathrm{P}$, but is not FO-rewritable.
- CERTAINTY$(q, \Sigma)$ is $\mathrm{coNP}$-complete.

Moreover, this trichotomy is decidable in polynomial time.

# Progress towards the Trichotomy Conjecture

- In 2015, Koutris and Wijsen proved the conjecture for Boolean conjunctive queries with no self-joins, i.e., no relation symbol occurs more than once in the query.

Key Notion: The attack graph associated with $\Sigma$ and $q$.

- The nodes of the attack graph are the atoms of $q$.
- The edges of the attack graph are determined by the functional dependencies on the variables of an atom that are implied by the keys of the other atoms.

# Progress towards the Trichotomy Conjecture

### Theorem (Koutris and Wijsen - 2015)

Let $\Sigma$ be a set of key constraints with one key per relation and let $q$ is a Boolean self-join free conjunctive query.

- If the attack graph is acyclic, then CERTAINTY$(q, \Sigma)$ is FO-rewritable.
- If the attack graph contains a cycle, but no strong cycle, then CERTAINTY$(q, \Sigma)$ is in $\mathrm{P}$, but it is not FO-rewritable.
- If the attack graph contains a strong cycle, then CERTAINTY$(q, \Sigma)$ is $\mathrm{coNP}$-complete.

Moreover, these conditions can be checked in quadratic time.

# Theory and Practice

- The framework of repairs and consistent query answering is a principled approach to coping with inconsistency in databases.
- Extensive study of the complexity of repair checking and consistent query answering during the past twenty years.
- This research, however, has not penetrated the industry.
- One of the reasons for this gap between theory and practice is that industrial-strength CQA-systems have yet to be developed.

# Earlier Prototype Consistent Query Answering Systems

| System | Constraints | Queries | Method |
|---|---|---|---|
| Hippo | Universal | Projection-free with $\cup$ and $\setminus$ | Direct Algorithm |
| ConQuer | Key | Aggregation queries in $C_{aggforest}$ | SQL-Rewriting |
| ConsEx | Universal$^{+}$ | Datalog with $\neg$ | Answer Set Programming |
| EQUIP | Key | Conjunctive | Reduction to ILP |

- Hippo (Chomicki, Marcinkowski, Staworko - 2004)
- ConQuer (Fuxman - 2007)
- ConsEx (Caniupan, Bertossi - 2010)
- EQUIP (K . . ., Pema, Tan - 2013)

# A New Consistent Query Answering System

CAvSAT: Consistent Query Answering via SAT Solving

- CAvSAT can handle denial constraints.
- CAvSAT can handle unions of conjunctive queries and aggregation queries whose underlying query is a union of conjunctive queries.
- CAvSAT deploys reductions to SAT and to optimization variants of SAT.
- Developed by Akhil A. Dixit in his 2021 PhD Dissertation at UCSC.

# Denial Constraints

**Definition**

A denial constraint is a FO-formula of the form

$$\forall \mathbf{x} \neg \psi(\mathbf{x}),$$

where $\psi(\mathbf{x})$ is a conjunction of atoms and of built-in predicates $=, \neq, \leq, \leq$.

**Example**

- Every functional dependency (hence, every key) is a denial constraint.

$$\forall a, b, c, c', d, d' (R(a, b, c, d) \wedge R(a, b, c', d') \rightarrow d = d')$$

$$\forall a, b, c, c', d, d' \neg (R(a, b, c, d) \wedge R(a, b, c', d') \wedge d \neq d')$$

- Every disjointness constraint is a denial constraint.

$$\forall \mathbf{x} \neg (R(\mathbf{x}) \wedge S(\mathbf{x}))$$

# Aggregation Queries

**Definition** An aggregation query is a query of the form

$$\texttt{SELECT}\ Z, f(A)\ \texttt{FROM}\ R(U, Z, A)\ \texttt{GROUP BY}\ Z, \text{where}$$

- $f(A)$ is one of the aggregation operators SUM($A$), COUNT($A$), COUNT(*), MIN($A$), MAX($A$), and AVG($A$);
- $R(U, Z, A)$ is a conjunctive query or a union of conjunctive queries.

**Example**

- Relation ACCOUNTS(accid, type, city, bal)
- Aggregation query

SELECT city, SUM(bal) FROM ACCOUNTS GROUP BY city

**Note**

Aggregation queries are the most frequently asked database queries.

# Range Consistent Answers

Question: What is the semantics of an aggregation query over an inconsistent database?

Definition: Let $\mathcal{I}$ be a database and let $Q$ be an aggregation query

$$\texttt{SELECT } Z, f(A) \texttt{ FROM } R(U, Z, A) \texttt{ GROUP BY } Z.$$

A tuple $(T, [glb, lub])$ is a range consistent answer to $Q$ on $\mathcal{I}$ if

- For every repair $\mathcal{J}$ of $\mathcal{I}$, there exists $d$ s.t. $(T, d) \in Q(\mathcal{J})$ and $glb \leq d \leq lub$
- For some repair $\mathcal{J}'$ of $\mathcal{I}$, we have that $(T, glb) \in Q(\mathcal{J}')$
- For some repair $\mathcal{J}''$ of $\mathcal{I}$, we have that $(T, lub) \in Q(\mathcal{J}'')$.

Arenas, Bertossi, Chomicki – 2003, Fuxman, Fazli, Miller – 2005)

# Example of Range Consistent Answers

- Constraints: Set $\Sigma$ of two key constraints

  ACCOUNTS: accid $\rightarrow$ type, city, bal    CUSTACC: accid $\rightarrow$ cid

- Database: $\mathcal{I}$

| ACCOUNTS | | | |
|---|---|---|---|
| accid | type | city | bal |
| A1 | Checking | LA | 900 |
| A2 | Checking | LA | 1000 |
| A3 | Saving | SJ | 1200 |
| A3 | Saving | SF | -100 |
| A4 | Saving | SJ | 300 |

| CUSTACC | |
|---|---|
| cid | accid |
| C1 | A1 |
| C2 | A2 |
| C2 | A3 |
| C3 | A4 |

- Aggregation Query: $Q$

  SELECT SUM(ACCOUNTS.bal) FROM ACCOUNTS, CUSTACC
  WHERE ACCOUNTS.accid $=$ CUSTACC.accid AND CUSTACC.CID $=$ 'C2'

- Range Consistent Answers: $\text{CONS}(Q, \mathcal{I}, \Sigma) = \{$ [900, 2200] $\}$

# CQA Systems for Aggregation Queries

- ConQuer is the only earlier CQA system supporting aggregation queries.

  Fuxman, Fazli, Miller – 2005, Fuxman – 2007

- However, ConQuer can only handle aggregation queries

  $$\texttt{SELECT}\ Z, f(A)\ \texttt{FROM}\ R(U, Z, A)\ \texttt{GROUP BY}\ Z,$$

  where the underlying query $R(U, Z, A)$ is a conjunctive query in a class, called $C_{forest}$, of FO-rewritable queries.

- The range consistent answers of such aggregation queries are SQL-rewritable.

# CQA Systems for Aggregation Queries

- ConQuer is the only earlier CQA system supporting aggregation queries.

  Fuxman, Fazli, Miller – 2005, Fuxman – 2007

- However, ConQuer can only handle aggregation queries

  $$\texttt{SELECT}\ Z, f(A)\ \texttt{FROM}\ R(U, Z, A)\ \texttt{GROUP BY}\ Z,$$

  where the underlying query $R(U, Z, A)$ is a conjunctive query in a class, called $C_{forest}$, of FO-rewritable queries.

- The range consistent answers of such aggregation queries are SQL-rewritable.

Fact: The range consistent answers to an aggregation query can be NP-hard, even when the underlying query has SQL-rewritable consistent answers.

# NP-Hardness of Range Semantics

Theorem: Let $Q$ be the aggregation query

$$\texttt{SELECT SUM}(A)\ \texttt{FROM}\ q(A),$$

where $q(A)$ is the conjunctive query

$$\exists x \exists y (R_1(\underline{x}, \text{'red'}) \wedge R_2(\underline{y}, \text{'blue'}) \wedge R_3(\underline{x, \text{'red'}, y, \text{'blue'}}, A))$$

with the underlined attributes as the keys. Then the following statements hold.

- CONS($q$) is FO-rewritable (hence, it is SQL-rewritable).
- CONS($Q$) is NP-hard.

Proof Hint: Polynomial-time reduction from MAXIMUM CUT to CONS($Q$).

# Consistent Query Answering Via SAT Solving

CAvSAT: Consistent Query Answering via SAT Solving

- CAvSAT can handle unions of conjunctive queries and aggregation queries with `SUM`($A$), `COUNT`($A$), `COUNT`(*), `MIN`($A$), `MAX`($A$), whose underlying query is a union of conjunctive queries.
- CAvSAT deploys reductions to SAT and to optimization variants of SAT.

# Basic Notions and Terminology

Definition: Let $\mathcal{I}$ be a database.

- $R(a_1, \ldots a_n)$ is a fact of $\mathcal{I}$ if $(a_1, \ldots, a_n)$ is a tuple in the relation $R$ of $\mathcal{I}$.
- Two facts $R(a_1, \ldots a_n)$ and $R(a'_1, \ldots, a'_n)$ of a relation $R$ of $\mathcal{I}$ are key-equal if they agree on the key attributes of $R$.
- A key-equal group of facts of $\mathcal{I}$ is an equivalence class of the key equal equivalence relation on a relation $R$ of $\mathcal{I}$.
- Let $q$ be a Boolean query on $\mathcal{I}$. A sub-database $\mathcal{S}$ of $\mathcal{I}$ is a minimal witness to $q$ on $\mathcal{I}$ if $\mathcal{S} \models q$, but for every $\mathcal{S}' \subset \mathcal{S}$, we have that $\mathcal{S}' \not\models q$.

Example: Assume that $\mathcal{I}$ consists of the facts $R(\underline{a}, c)$, $R(\underline{a}, d)$, $S(\underline{b}, c)$.

- The facts $R(\underline{a}, c)$ and $R(\underline{a}, d)$ are key-equal.
- The facts $R(\underline{a}, c)$, and $S(\underline{b}, c)$ form a minimal witness to the query

$$\exists x, y, z(R(x, y) \wedge S(z, y)).$$

# Warmup: Reducing CQA to UNSAT for Boolean Conjunctive Queries

Fix a set $\Sigma$ of key constraints and a Boolean conjunctive query $q$

Problem: Given a database $\mathcal{I}$, compute $\textsc{Certainty}(q, \mathcal{I}, \Sigma)$

Reduction:

- Given database $\mathcal{I}$, let

$$\begin{aligned} \mathcal{G} &= \text{the set of key-equal groups of facts of } \mathcal{I} \\ \mathcal{W} &= \text{the set of minimal witnesses to the query } q \text{ on } \mathcal{I}. \end{aligned}$$

- For every fact $f_i$ of $\mathcal{I}$, introduce a Boolean variable $x_i$ .
- For every key-equal group $G_j \in \mathcal{G}$, let $\alpha_j = \bigvee_{f_i \in G_j} x_i$
- For every minimal witness $W_k \in \mathcal{W}$, let $\beta_k = \bigvee_{f_i \in W_k} \neg x_i$
- Construct the CNF formula $\varphi = (\bigwedge_j \alpha_j) \wedge (\bigwedge_k \beta_k)$

Fact: The formula $\varphi$ is satisfiable if and only if $\textsc{Certainty}(q, \mathcal{I}, \Sigma) =$ *false*.

# Reducing Range CQA to Weighted Partial MaxSAT

Fix a set $\Sigma$ of key constraints and an aggregation query $Q$

$$\texttt{SELECT COUNT}(A)\ \texttt{FROM}\ T(U, A),$$

where $T(U, A)$ is a self-join free conjunctive query.

Problem: Given a database $\mathcal{I}$, compute Range $\textsc{Cons}(Q, \mathcal{I}, \Sigma)$

# Reducing Range CQA to Weighted Partial MaxSAT

Fix a set $\Sigma$ of key constraints and an aggregation query $Q$

$$\texttt{SELECT COUNT}(A)\ \texttt{FROM}\ T(U, A),$$

where $T(U, A)$ is a self-join free conjunctive query.

Problem: Given a database $\mathcal{I}$, compute Range $\textsc{Cons}(Q, \mathcal{I}, \Sigma)$

Definition: Weighted Partial MaxSAT ( WPMaxSAT)

Given a CNF-formula $\psi$ in which

- some clauses are assigned infinite weight (hard clauses) and
- the rest are assigned positive weights (soft clauses),

find an assignment that

- satisfies all hard clauses and
- maximizes the sum of weights of the satisfied soft clauses.

# Reduction of Range CQA to Weighted Partial Max SAT

- Given database $\mathcal{I}$, let

$$\begin{aligned} \mathcal{G} &= \text{the set of key-equal groups of facts of } \mathcal{I} \\ \mathcal{W} &= \text{the set of minimal witnesses to } q(A) := \exists U\ T(U, A) \text{ on } \mathcal{I}. \end{aligned}$$

- For every $G_j \in \mathcal{G}$, do the following:
  - Construct a hard clause $\alpha_j = \bigvee_{f_i \in G_j} x_i$
  - For every pair $(f_m, f_n)$ of facts in $G_j$ with $m \neq n$, construct a hard clause $\alpha_j^{mn} = (\neg x_m \vee \neg x_n)$
- For each $W_j \in \mathcal{W}$, construct a weighted soft clause $\beta_j = \left(\bigvee_{f_i \in W_j} \neg x_i, 1\right)$
- Let $\psi = \left(\bigwedge_{j=1}^{|\mathcal{G}|} \alpha_j\right) \wedge \left(\bigwedge_{j=1}^{|\mathcal{G}|} \left(\bigwedge_{f_m, f_n \in \mathcal{G}_j} \alpha_j^{mn}\right)\right) \wedge \left(\bigwedge_{j=1}^{|\mathcal{W}|} \beta_j\right)$.

Fact: In a max assignment of the WPMaxSAT instance $\psi$, the sum of weights of the falsified clauses is the glb-answer in $\textsc{Cons}(Q, \mathcal{I}, \Sigma)$. Similarly, for min assignments and lub-answers.

# Modifications to Handle Self-Joins

- SQL uses bag (multiset) semantics.
- If there are no self-joins, it suffices to consider minimal witnesses.
- If there are self-joins, we need to consider witnesses with multiplicities

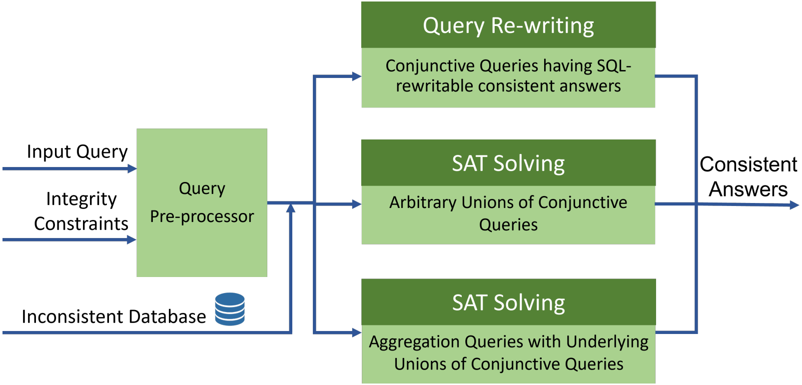
Example Let $Q$ be the query

$$\texttt{SELECT COUNT(*) FROM}\ \mathrm{T(X,Y)},$$

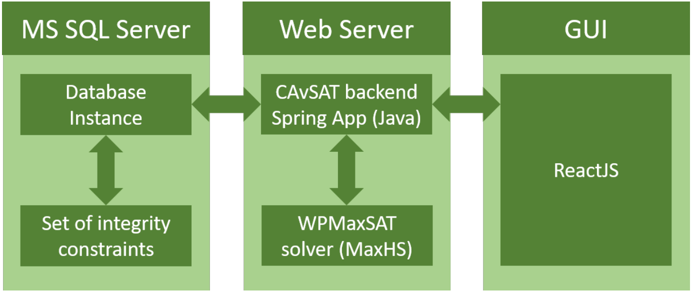
where $T(X, Y) := \exists Z(R(X, Y) \wedge R(X, Z))$, and let $\mathcal{I} = \{R(1, 1), R(1, 2)\}$.

- Witness $\{R(1, 1)\}$, assignment $(X/1, Y/1, Z/1) \hookrightarrow$ tuple $T(1, 1)$.
- Witness $\{R(1, 2)\}$, assignment $(X/1, Y/2, Z/2) \hookrightarrow$ tuple $T(1, 2)$.
- Witness $\{R(1, 1), R(1, 2)\}$
  - assignment $(X/1, Y/1, Z/2) \hookrightarrow$ tuple $T(1, 1)$.
  - assignment $(X/1, Y/2, Z/1) \hookrightarrow$ tuple $T(1, 2)$.
- Bag of Witnesses
  $\mathcal{W} = \{\{R(1, 1)\} : 1, \{R(1, 2\} : 1, \{R(1, 1), R(1, 2)\} : 2\}$.

# Architecture Overview of CAvSAT



# Implementation Overview of CAvSAT




Code is open-sourced at https://github.com/uccross/cavsat

# Experimental Evaluation

- Standard TPC-H databases and TPC-H aggregation queries
- Aggregation queries with and without grouping
- Comparison of CAvSAT vs. ConQuer SQL-rewriting
- Scalability experiments by varying database size and inconsistency percentage
- Real-world Medigap database with denial constraints

Note:

- TPC-H is a decision support benchmark.
- Medigap is a public database about Medicare supplement insurance.

# Experiments with Aggregation Queries Without Grouping

- TPC-H databases generated using the `DBGen`-tool (10% inconsistency, 1GB repair)
- One key constraint per relation
- TPC-H inspired aggregation queries without grouping

# Experiments with Aggregation Queries Without Grouping

- TPC-H databases generated using the `DBGen`-tool (10% inconsistency, 1GB repair)
- One key constraint per relation
- TPC-H inspired aggregation queries without grouping

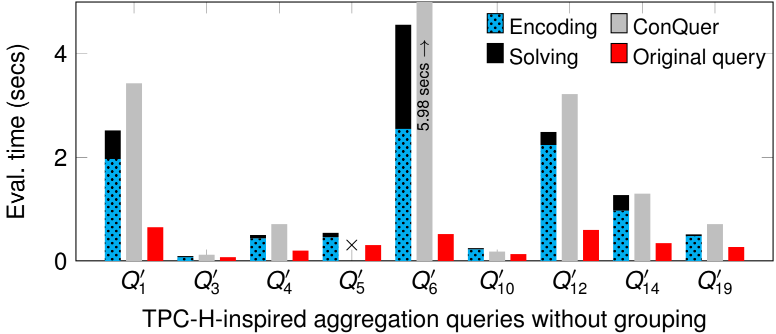
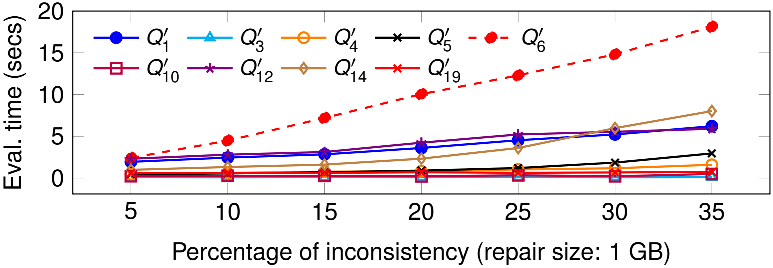
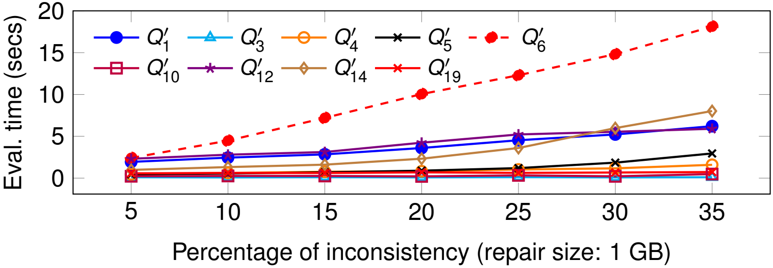



Figure: CAvSAT vs. ConQuer on TPC-H data generated using the `DBGen` tool

# Scalability of CAvSAT: Aggregation Queries Without Grouping



# Scalability of CAvSAT: Aggregation Queries Without Grouping



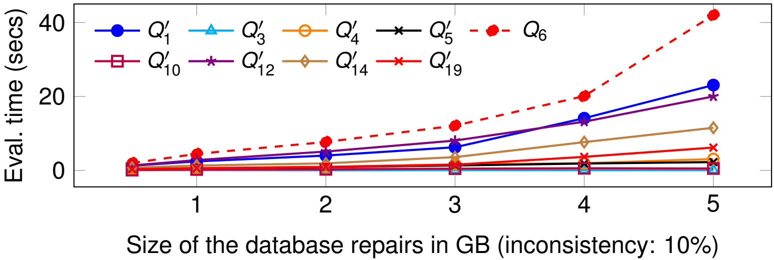




Figure: Evaluation time of CAvSAT with varying inconsistency and database sizes

# Experiments with Real-world Database and Queries

- Medigap: real-world database about Medicare supplement insurance
- Two functional dependencies, one denial constraint
  (5% existing inconsistency)
- Twelve natural aggregation queries

# Experiments with Real-world Database and Queries

- Medigap: real-world database about Medicare supplement insurance
- Two functional dependencies, one denial constraint (5% existing inconsistency)
- Twelve natural aggregation queries

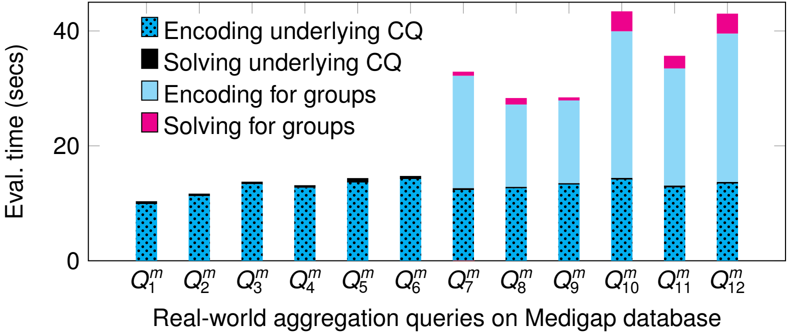



Figure: Performance of CAvSAT on a real-world database

# Concluding Remarks

Summary:

- CAvSAT: A SAT-based CQA system that beats or performs as well as the earlier ones and supports aggregation queries.
- Natural reductions to compute the range consistent answers to aggregation queries with COUNT(*), COUNT($A$), SUM($A$), MIN($A$), MAX($A$).

Open Problems:

- Find "good" reductions from the range consistent answers to aggregation queries with AVG($A$) to SAT and its variants.
- Prove dichotomy theorems for richer classes of queries.
- Develop a methodology for comparing data cleaning to consistent query answering.

# References

Papers:


- Akhil A. Dixit, Phokion G. Kolaitis: A SAT-Based System for Consistent Query Answering. SAT 2019: 117-135.
- Akhil A. Dixit, Phokion G. Kolaitis: CAvSAT: Answering Aggregation Queries over Inconsistent Databases via SAT Solving. SIGMOD Conference 2021 (Demo Track): 2701-2705.
- Akhil A. Dixit, Phokion G. Kolaitis: Consistent Answers of Aggregation Queries via SAT. ICDE 2022: 924-936.


Dissertation:


- Akhil A. Dixit: Answering Queries Over Inconsistent Databases Using SAT Solvers. University of California, Santa Cruz, USA, 2021