Tuple Inconsistency Measures: Towards Explaining Query Answers

Yurun Gu[1] , **Badran Raddaoui**[1] , **Yue Ma**[2] , **Aikaterini Tzompanaki**[3] and **Nicole Bidoit**[2]
[1]SAMOVAR, Télécom SudParis, Institut Polytechnique de Paris, Palaiseau, France
[2]LISN, Université Paris-Saclay, Gif-sur-Yvette, France
[3]ETIS, CY Cergy-Paris University, ENSEA, CNRS, UMR8051
{yurun.gu, badran.raddaoui}@telecom-sudparis.eu, {ma, nicole.bidoit}@lisn.fr,
aikaterini.tzompanaki@cyu.fr

Abstract

In this paper, we introduce novel tuple inconsistency measures that quantify the extent to which individual tuples contribute to violations of denial constraints in inconsistent databases. Then, we formally show that one of the proposed measures fully satisfies a set of well-motivated axiomatic properties. As an application, we lift tuple inconsistency to the level of query answers, obtaining a principled framework for measuring and explaining the inconsistency inherent in query results. We further investigate the computational complexity of tuple and answer inconsistency measures under denial constraints, identifying tractable cases that make their application to real-world data feasible. Finally, our experimental study on varying dataset and inconsistency settings demonstrates that, in comparison to existing methods, the proposed measures yield more fine-grained and informative assessments of inconsistency in databases.

1 Introduction

In real-world applications, databases often violate semantic requirements, especially under rapid data growth and integration. These requirements are formalized as integrity constraints (e.g., functional dependencies, denial constraints) that encode domain knowledge. Constraint violations lead to data inconsistencies, yet fully repairing a database is often impractical due to the exponential number of possible repairs. To assess the severity of violations or guide repair selection, quantitative metrics known as *inconsistency measures* are employed [Livshits *et al.*, 2021b; Issa *et al.*, 2020].

Inconsistency measures, recently studied in the database literature, can be broadly categorized into three classes. The first quantifies conflicts of an entire *database* [Parisi and Grant, 2020; Livshits *et al.*, 2021b; Parisi and Grant, 2023], providing a global data quality assessment that distinguishes data sources, but cannot capture how individual tuples contribute to overall inconsistency. The second, *tuple measures*, evaluates each tuple's participation in constraint violations [Issa *et al.*, 2020; Livshits and Kimelfeld, 2022] and is particularly valuable for tasks such as tuple ranking and data cleaning. The third class, *query answer measures*, quantifies the inconsistency associated with individual query answers [Issa *et al.*, 2020]. The first two classes of inconsistency measures have been extensively studied in knowledge representation [Hunter and Konieczny, 2010; Mu, 2018; Raddaoui *et al.*, 2024; Kuhlmann *et al.*, 2025; Straßer *et al.*, 2025], and the third is specific for databases and applications, e.g., [Corea and Thimm, 2020; Ribeiro and Thimm, 2021; Corea *et al.*, 2024].

This paper focuses on the second and third aforementioned classes. Our goal is to systematically analyze and compare different inconsistency measures, both theoretically and empirically. Based on existing and new desiderata, we show that current measures are either too coarse-grained or fail to meet some intuitive logical properties. To address these shortcomings, we propose a class of fine-grained inconsistency measures and ultimately identify a measure that satisfies all the axiomatic properties. Moreover, our answer inconsistency measures investigate how much query results inherit conflicts from the underlying database, which complements the only existing proposal in this category [Issa *et al.*, 2020].

Our contributions are fourfold: We propose three novel tuple inconsistency measures and prove that one of them, $\mathsf{IM}_{\mathsf{PIM}}$, satisfies all properties expected from a reasonable measure (**Section 3**). Building on these tuple measures, we develop an extensible framework to quantify the inconsistency of query answers using different answer explainability methods. We further show that a subset of these measures satisfies the proposed criteria for answer-level inconsistency (**Section 4**). We present a data complexity analysis of both tuple and answer measures, identifying tractable cases that facilitate practical deployment (**Section 5**). We perform a comparative experimental evaluation to show the feasibility of our tuple and answer inconsistency measures. In particular, we show that $\mathsf{IM}_{\mathsf{PIM}}$ and $\mathsf{ICQA}^{\mathsf{IM}}_{\mathsf{shap}}$ provide useful discrimination for tuples and answers, respectively, w.r.t. their inconsistency levels, while exhibiting promising scalability (**Section 6**).

2 Preliminaries

A database **schema** $\mathcal{S}$ defines the structural design of a database and it specifies a finite set of *relation schemas*. Each **relation schema** R is associated with a type $type(R)$, a finite set of attributes $\{R.A_1, \ldots, R.A_n\}$, where each attribute is prefixed by the relation name. The *arity* of a relation R is its

type cardinality. A **tuple** t is a sequence of *attribute-value pairs* $(A_1 : a_1, \ldots, A_n : a_n)$.

A **database instance** (or database) $\mathcal{I}$ over a schema $\mathcal{S}$ maps each relation $R \in \mathcal{S}$ to a finite set of tuples of $type(R)$, denoted $\mathcal{I}|_R$. We treat $\mathcal{I}$ as a set of tuples with possibly different types. We assume all attributes share a common domain dom, a countably infinite set of constants. The *active domain* of $\mathcal{I}$, denoted $dom(\mathcal{I})$, is the set of constants appearing in $\mathcal{I}$. A valuation ν over $\mathcal{I}$ for a vector/tuple of variables $\vec{x} = (x_1, \ldots, x_n)$ is a tuple $(a_1, \ldots, a_n)$ such that each $\nu(x_i) = a_i \in dom(\mathcal{I})$. We use $R(\vec{x})$ to denote a relational atom, where $\vec{x}$ matches the arity of R.

A *condition* is a comparison $f(\vec{x}) \star g(\vec{x})$, where f and g are functions (constant and string function included), and $\star$ is a classical comparator, for instance among $=, \neq, \leq, \geq, <, >$. Conditions are used in *denial constraints* as follows:

Definition 1. *A* denial constraint δ *has the form* $\forall \vec{x}\, \neg[R_1(\vec{u_1}) \wedge \cdots \wedge R_k(\vec{u_k}) \wedge \phi(\vec{x})]$ *where* $\vec{u_i}$ $(1 \leq i \leq k)$ *is a tuple of variables,* $\vec{x}$ *captures all the variables appearing in* $\vec{u_1}, \ldots, \vec{u_k}$*, and* $\phi(\vec{x}) = C_1 \wedge \ldots \wedge C_l$ $(l \geq 1)$ *is a conjunction of conditions over* $\vec{x}$*. Moreover, for any* i,j *s.t.* $1 \leq i \neq j \leq k$*, the tuples* $\vec{u_i}$ *and* $\vec{u_j}$ *do not share variables. We write* Δ *for a set of denial constraints.*

Given a database $\mathcal{I}$ and a set of denial constraints Δ, *conjunctive queries* can be used to check the consistency of $\mathcal{I}$ w.r.t. Δ, and more generally for posing queries over $\mathcal{I}$.

Definition 2. *A conjunctive query* $q(\vec{y})$ *is defined as an expression of the form* $R_1(\vec{u_1}) \wedge \cdots \wedge R_k(\vec{u_k}) \wedge \phi(\vec{x})$ *subject to the same conditions as in Definition 1, where* $\vec{y}$ *is a subvector of variables in* $\vec{x}$*. The result* $q(\vec{y})$ *over an instance* $\mathcal{I}$*, i.e., the set of query answers, is:* $q(\vec{y})(\mathcal{I}) = \{\nu(\vec{y}) \mid \forall i \in [1,k],\ \nu(u_i) \in R_i,$ *and* $\forall j \in [1,l], C_j(\nu(\vec{x}))$ *is true*$\}$ *for valuations* ν *of* $\vec{x}$ *over* $\mathcal{I}$ *and conditions* C_j *in* ϕ*.*

When used for finding conflicting tuples, Definition 2 is derived by negating the expression in Definition 1. For simplicity, we denote by $q(\mathcal{I})$ the set of answers returned by the query $q(\vec{y})$ when evaluated over the instance $\mathcal{I}$. The query $q(\vec{y})$ is called a *boolean query* when $\vec{y}$ is empty. Given a denial constraint $\delta \in \Delta$, we write q^δ for the boolean query associated with δ in the obvious manner. $\mathcal{I}$ satisfies δ, denoted $\mathcal{I} \models \delta$, if $q^\delta(\mathcal{I})$ is empty. We also write $\mathcal{I} \models \Delta$ if $\mathcal{I}$ satisfies all constraints in Δ. We now introduce two key concepts for reasoning about inconsistencies in Symbolic AI: *minimal inconsistent sets* and *minimal correction sets* [Reiter, 1987].

Definition 3. *A subset* $\mathcal{I}' \subseteq \mathcal{I}$ *is a* minimal inconsistent set *for* $\mathcal{I}$ *w.r.t.* $\delta \in \Delta$ *if* $\mathcal{I}' \not\models \delta$ *and* $\forall\, \mathcal{I}'' \subset \mathcal{I}', \mathcal{I}'' \models \delta$*. A subset* $\mathcal{I}' \subseteq \mathcal{I}$ *is a* minimal inconsistent set *for* $\mathcal{I}$ *w.r.t.* Δ *if* $\mathcal{I}' \not\models \Delta$ *and* $\forall \mathcal{I}'' \subset \mathcal{I}', \mathcal{I}'' \models \Delta$*.*

Definition 4. *A subset* $\mathcal{I}' \subseteq \mathcal{I}$ *is a* minimal correction set *for* $\mathcal{I}$ *and* Δ *if* $\mathcal{I} \setminus \mathcal{I}' \models \Delta$ *and* $\forall t \in \mathcal{I}'$*,* $(\mathcal{I} \setminus \mathcal{I}') \cup \{t\} \not\models \Delta$*.*

The set of all minimal correction sets for $\mathcal{I}$ w.r.t. Δ is denoted by $\mathsf{MIC}(\mathcal{I}, \Delta)$. The set of all minimal inconsistent sets for $\mathcal{I}$ w.r.t. δ (resp. Δ) is denoted by $\mathsf{MIS}(\mathcal{I}, \delta)$ (resp. $\mathsf{MIS}(\mathcal{I}, \Delta)$). We write $\biguplus_{\delta \in \Delta} \mathsf{MIS}(\mathcal{I}, \delta)$ for the multiset union of all minimal inconsistent subsets for $\mathcal{I}$ w.r.t. $\delta \in \Delta$. Given a tuple $t \in \mathcal{I}$, we define $\mathsf{MIS}(t, \mathcal{I}, \Delta) = \{M \in \mathsf{MIS}(\mathcal{I}, \Delta) \mid t \in M\}$, and $\mathsf{MIC}(t, \mathcal{I}, \Delta) = \{M \in \mathsf{MIC}(\mathcal{I}, \Delta) \mid t \in M\}$.

A tuple $t \in \mathcal{I}$ is called *free* w.r.t. Δ if t does not belong to any set $M \in \mathsf{MIS}(\mathcal{I}, \delta)$ for any $\delta \in \Delta$. We write $\mathsf{Free}(\mathcal{I}, \Delta)$ for the set of free tuples in $\mathcal{I}$ given Δ.

Example 1. *Consider a database with relations* R_1, R_2, R_3 *containing PhD candidate information as given in Figure 1. Tuples are assigned unique identifiers, i.e.,* $t_1, \ldots, t_{15}$*. We consider the following denial constraints:*

$$\delta_1 : \neg[R_1(x_1, y, z) \wedge R_2(x_2, v, u) \wedge x_1 = x_2 \wedge z \geq u]$$
$$\delta_2 : \neg[R_1(x_1, y, z) \wedge R_2(x_2, v, u) \wedge x_1 = x_2 \wedge u - z < 3 \wedge z > 2020]$$
$$\delta_3 : \neg[R_1(x_1, y, z) \wedge R_2(x_2, w, r) \wedge R_3(x_3, v, u) \wedge x_1 = x_2 \wedge x_1 = x_3 \wedge y > v \wedge r - u < 1]$$

Intuitively, the constraint δ_1 *ensures that graduation must occur after admission;* δ_2 *imposes a condition on students admitted after 2020, preventing them from graduating within three year; and* δ_3 *disallows graduating the same year as candidacy when the qualification grade is lower than the admission grade. We have* $\mathsf{MIS}(\mathcal{I}, \Delta) = \{\{t_1, t_6\},\ \{t_2, t_6\},\ \{t_5, t_9\},\ \{t_5, t_{10}\},\ \{t_4, t_8, t_{13}\}\}$*. Observe that* $\{t_5, t_{10}, t_{14}\} \notin \mathsf{MIS}(\mathcal{I}, \Delta)$*, but* $\{t_5, t_{10}, t_{14}\} \in \biguplus_{\delta \in \Delta} \mathsf{MIS}(\mathcal{I}, \delta)$ *since it belongs to* $\mathsf{MIS}(\mathcal{I}, \delta_3)$*, while its subset* $\{t_5, t_{10}\}$ *already violates* δ_2*. This shows that* $\mathsf{MIS}(\mathcal{I}, \Delta)$ *is not always* $\biguplus_{\delta \in \Delta} \mathsf{MIS}(\mathcal{I}, \delta)$ *(see Proposition 1).*

Proposition 1. $\mathsf{MIS}(\mathcal{I}, \Delta) \subseteq \biguplus_{\delta \in \Delta} \mathsf{MIS}(\mathcal{I}, \delta)$.

Given a query $q(\vec{y})$, [Buneman *et al.*, 2001] introduced the concept of *why-provenance* to formally explain the answers of $q(\vec{y})$ by tracing the lineage of the results.

Definition 5. *The* why-provenance *of an answer tuple* $\mathsf{ans} \in q(\mathcal{I})$*, denoted by* $\mathsf{WhyP}(\mathcal{I}, q, \mathsf{ans})$*, is the set of minimal (w.r.t. set inclusion) subsets* $\mathcal{I}' \subseteq \mathcal{I}$ *such that* $\mathsf{ans} \in q(\mathcal{I}')$*.*

Example 2 (Example 1 contd.)**.** *Consider the query* $q_{ex}(x) : R_1(x, y, z) \wedge R_2(x, y_1, z_1) \wedge R_3(x, y_2, z_2)$*. The last table in Table 1 lists the query result with its provenance.*

3 Tuple Inconsistency Measures

Tuple inconsistency measures quantify the degree to which an individual tuple contributes to denial constraint violations in a database instance. Existing approaches generally distinguish between absolute and relative measures [Besnard and Grant, 2020]; in this work we focus on absolute measures.

Let $\mathbb{I}$ denote the set of all database instances over a given schema $\mathcal{S}$, and define $\underline{\mathbb{I}} = \{(t, \mathcal{I}, \Delta) \mid \mathcal{I} \in \mathbb{I}, t \in \mathcal{I}\}$.

Definition 6. *A* tuple inconsistency measure *is a function* IM *on* $\mathbb{I}$ *that associates a real value to each tuple* $t \in \mathcal{I}$ *w.r.t. a set of denial constraints* Δ*, i.e.,* $\mathsf{IM} : \underline{\mathbb{I}} \to \mathbb{R}_0^+$.

Intuitively, the more a tuple t participates in violations of denial constraints, the higher the value $\mathsf{IM}(t, \mathcal{I}, \Delta)$ it receives. Usually, we expect IM to satisfy a few basic properties: $\mathsf{IM}(t, \mathcal{I}, \Delta) = 0$ if $\mathcal{I} \models \Delta$ or if $t \in \mathsf{Free}(\mathcal{I}, \Delta)$, and that IM is always finite in all cases, i.e., $\mathsf{IM}(t, \mathcal{I}, \Delta) < \infty$. While these properties (also known as rationality postulates) are typically easy to satisfy, other intuitive criteria become challenging. We *adapt to the database setting* the following four important desiderata from [Raddaoui *et al.*, 2024]:

Free Tuple Independence (FI): If $t' \in \mathsf{Free}(\mathcal{I} \cup \{t'\}, \Delta)$, for any $t \in \mathcal{I}$ it holds that $\mathsf{IM}(t, \mathcal{I}, \Delta) = \mathsf{IM}(t, \mathcal{I} \cup \{t'\}, \Delta)$.

R_1: Admission

	StudID	Grade	Year
t_1	1	90	2018
t_2	1	85	2019
t_3	2	92	2021
t_4	3	90	2020
t_5	4	92	2021

R_2: Defense

	StudID	Grade	Year
t_6	1	90	2017
t_7	2	95	2024
t_8	3	89	2023
t_9	4	89	2023
t_{10}	4	82	2023

R_3: Qualification

	StudID	Grade	Year
t_{11}	1	90	2016
t_{12}	2	92	2024
t_{13}	3	87	2023
t_{14}	4	60	2023
t_{15}	4	91	2023

Answers (StudID)	Provenance
1	$\{\{t_1,t_6,t_{11}\},\{t_2,t_6,t_{11}\}\}$
2	$\{\{t_3,t_7,t_{12}\}\}$
3	$\{\{t_4,t_8,t_{13}\}\}$
4	$\{\{t_5,t_9,t_{14}\},\{t_5,t_9,t_{15}\},$ $\{t_5,t_{10},t_{14}\},\{t_5,t_{10},t_{15}\}\}$

Table 1: PhD Candidate database and query answer provenance of q_{ex} from the running example.

That is, adding a free tuple to the database instance does not change the inconsistency score of existing tuples.

Positivity (PO): If $t \in \mathsf{MIS}(\mathcal{I},\delta)$ for some $\delta \in \Delta$, then $\mathsf{IM}(t,\mathcal{I},\Delta) > 0$. This ensures that tuples involved in constraint violations receive a positive inconsistency score, in contrast to free tuples, whose inconsistency measure is zero.

Centrality Conflict (CC): If $t, t' \in \mathcal{I}$ such that $\mathcal{I} \setminus \{t\} \models \Delta$ and $\mathcal{I} \setminus \{t'\} \not\models \Delta$, then $\mathsf{IM}(t,\mathcal{I},\Delta) > \mathsf{IM}(t',\mathcal{I},\Delta)$. If removing t resolves all inconsistencies but removing t' does not, then t should have a higher inconsistency value. This property distinguishes tuples whose removal suffices to restore consistency from those whose removal does not.

Data Monotonicity (DM): For any $t \in \mathcal{I}' \subseteq \mathcal{I}$, it holds that $\mathsf{IM}(t,\mathcal{I}',\Delta) \leq \mathsf{IM}(t,\mathcal{I},\Delta)$. Adding tuples to a database cannot reduce the inconsistency value of a tuple. The rationale is that violations can only increase when database grows; hence, IM should be monotone w.r.t. instance extension.

Now, we show that these properties pose substantial challenges for the tuple measure $\mathsf{IM}_{\mathsf{CBM}}$ [Issa *et al.*, 2020] as well as for several measures adapted from Symbolic AI [Hunter and Konieczny, 2010; Ribeiro and Thimm, 2021; Raddaoui *et al.*, 2024].

Definition 7 (**$\mathsf{IM}_{\mathsf{CBM}}$**[Issa *et al.*, 2020])**.** *The Constraint-based Bag Semantics tuple inconsistency measure ($\mathsf{IM}_{\mathsf{CBM}}$) of a tuple $t \in \mathcal{I}$ is defined as $\mathsf{IM}_{\mathsf{CBM}}(t,\mathcal{I},\Delta) = \sum_{\delta\in\Delta} \mathbb{1}_{t\in\bigcup \mathsf{MIS}(\mathcal{I},\delta)}$, where $\mathbb{1}_A$ returns 1 if the logical assertion A holds, and 0 otherwise.*

Proposition 2. $\mathsf{IM}_{\mathsf{CBM}}$ *satisfies all the properties except* CC.

Definition 8 (KB Tuple Measure)**.** *The knowledge base (*KB*) tuple inconsistency measures of a tuple $t \in \mathcal{I}$ are defined as:*

- $\mathsf{IM}_d(t,\mathcal{I},\Delta) = 0$ *if* $\mathsf{MIS}(t,\mathcal{I},\Delta) = \emptyset$*; and* 1 *otherwise.*
- $\mathsf{IM}_{\mathsf{dr}}(t,\mathcal{I},\Delta) = \max\{1/|M| \mid M \in \mathsf{MIC}(t,\mathcal{I},\Delta)\}$ *if* $\mathsf{MIC}(t,\mathcal{I},\Delta) \neq \emptyset$*; and* 0 *otherwise.*
- $\mathsf{IM}_{\#}(t,\mathcal{I},\Delta) = |\mathsf{MIS}(t,\mathcal{I},\Delta)|$
- $\mathsf{IM}_P(t,\mathcal{I},\Delta) = |\bigcup_{M\in\mathsf{MIS}(t,\mathcal{I},\Delta)} M|$
- $\mathsf{IM}_{\mathsf{MIS}}(t,\mathcal{I},\Delta) = \sum_{M\in\mathsf{MIS}(t,\mathcal{I},\Delta)} \frac{1}{|M|}$

Proposition 3. *All* KB *tuple measures satisfy* FI*, but not* PO.

Next, we extend the Shapley tuple measure, originally proposed by [Livshits and Kimelfeld, 2022] in the context of functional dependencies, to the setting of denial constraints.

Definition 9. *The Shapley tuple inconsistency measure of a tuple $t \in \mathcal{I}$ is defined as follows:* $\mathsf{IM}^h_{\mathsf{shap}}(t,\mathcal{I},\Delta) =$

$$\sum_{\mathcal{I}'\subseteq(\mathcal{I}\setminus\{t\})} [\frac{|\mathcal{I}'|!\cdot(|\mathcal{I}|-|\mathcal{I}'|-1)!}{|\mathcal{I}|!} \times (h(\mathcal{I}'\cup\{t\},\Delta) - h(\mathcal{I}',\Delta))],$$

where $h(\mathcal{I},\Delta)$ is an inconsistency measure for the database $\mathcal{I}$ w.r.t. Δ. We consider two variants of h: $h_{\mathsf{MIS}}(\mathcal{I},\Delta) = |\mathsf{MIS}(\mathcal{I},\Delta)|$ and $h_P(\mathcal{I},\Delta) = |\cup_{M\in\mathsf{MIS}(\mathcal{I},\Delta)} M|$, resulting in two Shapley tuple measures, denoted $\mathsf{IM}^{\mathsf{MIS}}_{\mathsf{shap}}$ and $\mathsf{IM}^{P}_{\mathsf{shap}}$.

Note that [Livshits and Kimelfeld, 2022] showed that these two choices of h yield tractable measures. The same tractability result also holds for denial constraints (see Section 5).

Proposition 4. $\mathsf{IM}^{\mathsf{MIS}}_{\mathsf{shap}}$ *and* $\mathsf{IM}^{P}_{\mathsf{shap}}$ *satisfy* CC *and* DM*, but violate* PO *and* FI.

Violating some of these four desiderata may lead to counter-intuitive behavior, as illustrated in the next example.

Example 3 (Example 1 contd.)**.** *Given $\mathcal{I} = \{R_1, R_2, R_3\}$, one can observe that $\mathcal{I}$ is inconsistent w.r.t. δ_i $(1 \leq i \leq 3)$ with $\mathsf{MIS}(\mathcal{I},\delta_1) = \{\{t_1,t_6\},\{t_2,t_6\}\}$, $\mathsf{MIS}(\mathcal{I},\delta_2) = \{\{t_5,t_9\},\{t_5,t_{10}\}\}$, and $\mathsf{MIS}(\mathcal{I},\delta_3) = \{\{t_4,t_8,t_{13}\}, \{t_5,t_9,t_{14}\}, \{t_5,t_9,t_{15}\}, \{t_5,t_{10},t_{14}\}, \{t_5,t_{10},t_{15}\}\}$. We have $\mathsf{IM}_{\mathsf{CBM}}(t_5,\mathcal{I},\delta_2) = \mathsf{IM}_{\mathsf{CBM}}(t_9,\mathcal{I},\delta_2) = \mathsf{IM}_{\mathsf{CBM}}(t_{10},\mathcal{I},\delta_2) = 2$, indicating that t_5, t_9, and t_{10} have the same contribution to the violation of δ_2 according to $\mathsf{IM}_{\mathsf{CBM}}$. However, removing t_5 makes $\mathcal{I}$ consistent with δ_2, whereas removing t_9 (or t_{10}) does not. The same situation arises for t_1, t_2, and t_6 w.r.t. δ_1, illustrating the impact of the failure of* CC *for $\mathsf{IM}_{\mathsf{CBM}}$. Moreover, we have $\mathsf{IM}(t_{14},\mathcal{I},\Delta) = \mathsf{IM}(t_{15},\mathcal{I},\Delta) = 0$ for any measure* IM *defined in Definitions 8 and 9, in violation of* PO.

In what follows, we introduce new tuple inconsistency measures to investigate whether the four postulates are compatible, i.e., jointly satisfiable by a tuple measure.

3.1 Fine-Grained Tuple Inconsistency Measures

To address the limitations of existing tuple inconsistency measures, we introduce a novel family of measures that employ the *internal structure* of denial constraints to more accurately assess each tuple's contribution to constraint violations.

Condition-based Inconsistency Measure

Our first tuple inconsistency measure is designed to capture both the frequency with which a tuple appears in minimal inconsistent subsets of the database and the number of distinct constraint conditions it violates within each such subset.

Definition 10 ($\mathsf{IM}_{\mathsf{CIM}}$**).** *The* condition-based inconsistency measure *of t w.r.t. Δ is defined as:*

$$\mathsf{IM}_{\mathsf{CIM}}(t,\mathcal{I},\Delta) = \sum_{\delta\in\Delta} \sum_{M\in\mathsf{MIS}(\mathcal{I},\delta)} \sum_{1\leq i\leq l} \mathbb{1}_{t\in\bigcup \mathsf{MIS}(M,\delta_{C_i})},$$

where $\delta_{C_i} = \forall\vec{x}\,\neg[R_1(\vec{u_1})\wedge\cdots\wedge R_k(\vec{u_l})\wedge C_i(\vec{x})]$ for a given $\delta = \forall\vec{x}\,\neg[R_1(\vec{u_1})\wedge\cdots\wedge R_k(\vec{u_l})\wedge C_1(\vec{x})\wedge\cdots\wedge C_l(\vec{x})]$.

Before evaluating $\mathsf{IM}_{\mathsf{CIM}}$, we require every denial constraint to be in a *reduced form*, obtained by grouping the conditions of a constraint into one condition for "equal" variables (e.g., $x_1{=}x_2 \wedge x_2{=}x_3$ is rewritten as $x_1{=}x_2{=}x_3$), to ensure

Tuple	$\mathsf{IM}_{\mathsf{CBM}}$	$\mathsf{IM}_{\mathsf{CIM}}$	$\mathsf{IM}_{\mathsf{RIM}}$	$\mathsf{IM}_{\mathsf{PIM}}$
t_1	1	2	$1/4 \times 2 = 1/2$	$1/2$
t_2	1	2	$1/4 \times 2 = 1/2$	$1/2$
t_4	1	2	$3/20 \times 5 = 3/4$	$1/3$
t_5	2	14	$1/2 \times 2 + 3/20 \times 5 = 7/4$	$7/3$
t_6	1	4	$1/2 \times 2 = 1$	1
t_8	1	2	$3/20 \times 5 = 3/4$	$1/3$
t_9	2	6	$1/4 \times 2 + 1/10 \times 5 = 1$	$7/6$
t_{10}	2	6	$1/4 \times 2 + 1/10 \times 5 = 1$	$7/6$
t_{13}	1	3	$3/20 \times 5 = 3/4$	$1/3$
t_{14}	1	6	$1/10 \times 5 = 1/2$	$2/3$
t_{15}	1	6	$1/10 \times 5 = 1/2$	$2/3$

Table 2: Comparison of non-zero tuple inconsistency measures.

that syntactically different but logically equivalent constraints induce the same $\mathsf{IM}_{\mathsf{CIM}}$ value.

Example 4 shows that $\mathsf{IM}_{\mathsf{CIM}}$ effectively addresses the issue highlighted in Example 3. However, this observation does not hold in general, as shown by Proposition 5.

Example 4 (Examples 1 and 3 contd.)**.** *Let us focus on Column 2 and 3 of Table 2. Tuple t_5 receives the highest $\mathsf{IM}_{\mathsf{CIM}}$ score 14, as it is involved in multiple violations of both δ_2 and δ_3, and it contributes to a greater number of violated conditions than any other tuple. In contrast, t_9 and t_{10} make weaker overall contributions to inconsistency, despite their participation in violation of both δ_2 and δ_3. This behavior cannot be captured by $\mathsf{IM}_{\mathsf{CBM}}$, since $\mathsf{IM}_{\mathsf{CBM}}(t_5, \mathcal{I}, \Delta) = \mathsf{IM}_{\mathsf{CBM}}(t_9, \mathcal{I}, \Delta) = \mathsf{IM}_{\mathsf{CBM}}(t_{10}, \mathcal{I}, \Delta) = 2$.*

Proposition 5. $\mathsf{IM}_{\mathsf{CIM}}$ *satisfies* FI, PO *and* DM*, but fails* CC.

Responsibility-based Inconsistency Measure

Proposition 5 shows that condition-level granularity is insufficient to distinguish tuples whose removal restores consistency from those that do not, as required by the CC property. To address this limitation, we introduce a novel measure inspired by causality theory [Meliou *et al.*, 2010], enabling a more precise individual tuple inconsistency quantification.

Given a denial constraint δ and its corresponding boolean query q^δ, we consider the *responsibility measure* as defined in the context of query explainability [Meliou *et al.*, 2010]. We first recall some additional concepts. A tuple $t \in \mathcal{I}$ is a *counterfactual cause* for an answer ans to a query q if $\mathsf{ans} \in q(\mathcal{I})$ and $\mathsf{ans} \notin q(\mathcal{I} \setminus \{t\})$. A set $\Gamma \subseteq \mathcal{I}$ is a *contingency set* for t if t becomes a counterfactual cause for ans in $\mathcal{I} \setminus \Gamma$. Using these notions, we define the responsibility of a tuple $t \in \mathcal{I}$ for violating δ as: $\rho(t, \delta) = (1 + \min_{\Gamma(t,q^\delta)} |\Gamma(t, q^\delta)|)^{-1}$, where $\Gamma(t, q^\delta)$ ranges over all contingency sets for which t is a counterfactual cause of the *true* answer to q^δ. Notice that, if a tuple t does not violate a constraint δ, there is no contingency set for t, and we set $\rho(t, \delta) = 0$.

Using the notion of responsibility, we now define our tuple inconsistency measure that accounts for both how frequently a tuple is involved in denial constraint violations and the strength of its contribution to each such violation.

Definition 11 ($\mathsf{IM}_{\mathsf{RIM}}$)**.** *The* responsibility-based inconsistency measure *of t w.r.t. Δ is defined as:* $\mathsf{IM}_{\mathsf{RIM}}(t, \mathcal{I}, \Delta) = \sum_{\delta \in \Delta} \rho'(t, \delta) \times |\mathsf{MIS}(\mathcal{I}, \delta)|$ *where* $\rho'(t, \delta) = \rho(t, \delta) / \sum_{t' \in \bigcup \mathsf{MIS}(\mathcal{I}, \delta)} \rho(t', \delta)$.

	$\mathsf{IM}_{\mathsf{PIM}}$	$\mathsf{IM}_{\mathsf{CIM}}$	$\mathsf{IM}_{\mathsf{RIM}}$	$\mathsf{IM}_{\mathsf{CBM}}$	$\mathsf{IM}^{\mathsf{MIS}}_{\mathsf{shap}}$ $\mathsf{IM}^{P}_{\mathsf{shap}}$	IM_P IM_d	$\mathsf{IM}_{\mathsf{dr}}$	$\mathsf{IM}_{\mathsf{MIS}}$ $\mathsf{IM}_{\#}$
FI	√	√	√	√	⊗	√	√	√
PO	√	√	√	√	⊗	⊗	⊗	⊗
CC	√	⊗	√	⊗	√	⊗	√	√
DM	√	√	⊗	√	√	√	⊗	√

Table 3: Properties of tuple inconsistency measures.

Example 5 (Example 1 contd.)**.** *As illustrated in Table 2, the measure $\mathsf{IM}_{\mathsf{CIM}}$ assigns identical inconsistency values to tuples t_2 and t_4, i.e., $\mathsf{IM}_{\mathsf{CIM}}(t_2, \mathcal{I}, \Delta) = \mathsf{IM}_{\mathsf{CIM}}(t_4, \mathcal{I}, \Delta) = 2$. However, by $\mathsf{IM}_{\mathsf{RIM}}$ we have $\mathsf{IM}_{\mathsf{RIM}}(t_2, \mathcal{I}, \Delta) = 1/2 < 3/4 = \mathsf{IM}_{\mathsf{RIM}}(t_4, \mathcal{I}, \Delta)$, thereby providing a more nuanced and informative differentiation.*

Proposition 6. $\mathsf{IM}_{\mathsf{RIM}}$ *satisfies* FI, PO*, and* CC*, but fails* DM.

Proposition 6 highlights a key advantage of $\mathsf{IM}_{\mathsf{RIM}}$ over $\mathsf{IM}_{\mathsf{CBM}}$ and $\mathsf{IM}_{\mathsf{CIM}}$ measures, namely that it satisfies CC. However, it still fails to satisfy DM.

Provenance-based Inconsistency Measure

In this section, we present a tuple measure that satisfies all the properties studied in this paper (see Table 3). The intuition behind this new measure is that it quantifies the violation of a tuple with each constraint δ by using the sizes of minimal inconsistent sets containing it. More formally:

Definition 12 ($\mathsf{IM}_{\mathsf{PIM}}$)**.** *The* provenance-based inconsistency measure *of t w.r.t. Δ is defined as:* $\mathsf{IM}_{\mathsf{PIM}}(t, \mathcal{I}, \Delta) = \sum_{M \in \biguplus_{\delta \in \Delta} \mathsf{MIS}(\mathcal{I}, \delta)} \frac{1}{|M|} \cdot \mathbb{1}_{t \in M}$.

Example 6. *By Table 2, $\mathsf{IM}_{\mathsf{PIM}}(t_2, \mathcal{I}, \Delta) = 1/2$ and $\mathsf{IM}_{\mathsf{PIM}}(t_{14}, \mathcal{I}, \Delta) = 2/3$, even though these two tuples receive the same $\mathsf{IM}_{\mathsf{RIM}}$ value. Specifically, t_2 violates δ_1 and t_{14} violates δ_3; however, t_2 appears in only one set in $\mathsf{MIS}(\mathcal{I}, \delta_1)$, whereas t_{14} occurs in two sets in $\mathsf{MIS}(\mathcal{I}, \delta_3)$. This difference explains the discrepancy in their $\mathsf{IM}_{\mathsf{PIM}}$ values. Tuple t_5, which violates two constraints and occurs in 6 sets in $\biguplus_{\delta \in \Delta} \mathsf{MIS}(\mathcal{I}, \delta)$, receives the highest inconsistency value.*

It is important to highlight the key distinction between $\mathsf{IM}_{\mathsf{PIM}}$ and $\mathsf{IM}_{\mathsf{MIS}}$ (Definition 8) is that $\mathsf{IM}_{\mathsf{MIS}}$ overlooks certain "iceberg" violations that cannot be detected in $\mathsf{MIS}(\mathcal{I}, \Delta)$ (see Example 1). This limitation prevents $\mathsf{IM}_{\mathsf{MIS}}$ from ensuring PO, a requirement that $\mathsf{IM}_{\mathsf{PIM}}$ does meet.

Proposition 7. $\mathsf{IM}_{\mathsf{PIM}}$ *satisfies* FI, PO, CC*, and* DM.

4 Application: Explaining Query Answers

Building on tuple inconsistency measures, we propose a formal framework that quantifies the **In**Consistency of a **Q**uery **A**nswer (ICQA) by systematically aggregating the inconsistencies of its contributing tuples. This aggregation is parameterized by a tuple inconsistency measure, and (ii) an answer explainability method, for a versatile family of instantiations. Our framework generalizes the only existing proposal for quantifying inconsistency of query answers [Issa *et al.*, 2020].

Given a conjunctive query q and $\mathsf{ans} \in q(\mathcal{I})$ over a database $\mathcal{I}$ subject to a set of denial constraints Δ, our generic inconsistency measurement for query answers is denoted by

$\mathsf{ICQA}^{\mathsf{IM}}_{\mathsf{X}}(q, \mathsf{ans}, \mathcal{I}, \Delta)$, where X specifies the answer explanation method and IM denotes the tuple inconsistency measure. Below, we define several instantiations of this framework. For ease of notation, we denote $\mathsf{ICQA}^{\mathsf{IM}}_{\mathsf{X}}(q, \mathsf{ans}, \mathcal{I}, \Delta)$ simply as $\mathsf{ICQA}^{\mathsf{IM}}_{\mathsf{X}}$ whenever the context allows. We start by our first instantiation of ICQA, based on provenance.

Definition 13. *The* provenance based ICQA *for an answer* ans *to a query q over the database $\mathcal{I}$, denoted* $\mathsf{ICQA}^{\mathsf{IM}}_{\mathsf{Prov}}$*, is:* $\mathsf{ICQA}^{\mathsf{IM}}_{\mathsf{Prov}} = \sum_{M \in \mathsf{WhyP}(\mathcal{I}, q, \mathsf{ans})} \sum_{t \in M} \mathsf{IM}(t, \mathcal{I}, \Delta)$.

In Definition 13, the outer summation over $M \in \mathsf{WhyP}(\mathcal{I}, q, \mathsf{ans})$ explicitly aggregates across all the derivations of the answer ans. Using summation rather than a max-operator ensures that the measure reflects the cumulative inconsistency contributions of all supporting derivations, instead of being dominated by a single most inconsistent derivation. In this way, $\mathsf{ICQA}^{\mathsf{IM}}_{\mathsf{Prov}}$ captures a more complete provenance picture and enables finer discrimination among answers. Note that the inconsistency measure proposed by [Issa *et al.*, 2020] is a special case of Definition 13 by instantiating the tuple measure with $\mathsf{IM}_{\mathsf{CBM}}$, as detailed below.

Proposition 8. *Let* CBM *be the answer measure defined in [Issa* et al.*, 2020] (Definition 4). It holds* $\mathsf{CBM} = \mathsf{ICQA}^{\mathsf{IM}_{\mathsf{CBM}}}_{\mathsf{Prov}}$.

Shapley values [Shapley, 1953] are useful in various settings, including knowledge representation and databases [Hunter and Konieczny, 2010; Livshits and Kimelfeld, 2022]. As a second instantiation of our ICQA framework, we adapt Shapley values for inconsistent query answers.

Definition 14. *The* Shapley based ICQA *for an answer* ans *to a query q over the database $\mathcal{I}$, denoted* $\mathsf{ICQA}^{\mathsf{IM}}_{\mathsf{shap}}$*, is:*

$$\mathsf{ICQA}^{\mathsf{IM}}_{\mathsf{shap}} = \frac{1}{|\mathcal{I}|!} \sum_{t \in \mathcal{I}} \mathsf{IM}(t, \mathcal{I}, \Delta) \left(\sum_{\mathcal{I}' \in \pi_{\mathcal{I} \setminus \{t\}}} \nabla v \right)$$

where $\pi_{\mathcal{I}\setminus\{t\}}$ denotes all permutations of tuples in $\mathcal{I} \setminus \{t\}$, the marginal $\nabla v = v(\mathcal{I}' \cup \{t\}) - v(\mathcal{I}')$ with the wealth function $v(\cdot)$ returns 1 if $\mathsf{ans} \in q(\cdot)$*, and 0 otherwise.*

Notably, unlike the formulation of [Livshits *et al.*, 2021a], our value function $v(\cdot)$ is defined w.r.t. each individual answer rather than the complete query result set.

The final ICQA instantiation combines responsibility with tuple measures to quantify answer inconsistency.

Definition 15. *The* responsibility based ICQA *for an answer* ans *to a query q over the database $\mathcal{I}$, denoted* $\mathsf{ICQA}^{\mathsf{IM}}_{\mathsf{Resp}}$*, is:* $\mathsf{ICQA}^{\mathsf{IM}}_{\mathsf{Resp}} = \sum_{t \in \bigcup \mathsf{WhyP}(\mathcal{I}, q, \mathsf{ans})} \rho(t) \cdot \mathsf{IM}(t, \mathcal{I}, \Delta)$, *where* $\rho(t) = 1/(1+\min_{\Gamma_t} |\Gamma_t|)$ *with Γ_t the contingency set of t.*

To assess the suitability of measures for quantifying query-answer inconsistency, we introduce the following properties.

Blamelessness: $\mathsf{ICQA}^{\mathsf{IM}}_{\mathsf{X}}(q, \mathsf{ans}, \mathcal{I}, \Delta) = 0$ if and only if $\bigcup_{M \in \mathsf{WhyP}(q, \mathsf{ans}, \mathcal{I})} M \cap \bigcup_{M' \in \biguplus_{\delta \in \Delta} \mathsf{MIS}(\mathcal{I}, \delta)} M' = \emptyset$. Only answers with violation-free provenance are blameless (i.e., score 0). Otherwise, they have a positive $\mathsf{ICQA}^{\mathsf{IM}}_{\mathsf{X}}$ score.

Additivity: For two queries $q_1(\vec{y}), q_2(\vec{y})$ of the same arity, if $\mathsf{ans} \in q_1(\mathcal{I}) \cap q_2(\mathcal{I})$, then $\mathsf{ICQA}^{\mathsf{IM}}_{\mathsf{X}}(q_1 \vee q_2, \mathsf{ans}, \mathcal{I}, \Delta) \leq \mathsf{ICQA}^{\mathsf{IM}}_{\mathsf{X}}(q_1, \mathsf{ans}, \mathcal{I}, \Delta) + \mathsf{ICQA}^{\mathsf{IM}}_{\mathsf{X}}(q_2, \mathsf{ans}, \mathcal{I}, \Delta)$. Combining queries should not increase the inconsistency of an answer beyond the sum of individual inconsistencies. Indeed, the provenance of $q_1 \vee q_2$ is the union of the provenances of q_1 and q_2, introducing no additional sources of inconsistency.

In particular, additivity ensures that answers with a score 0 for both queries retain a score 0 under their disjunction.

Monotonicity: For any $\mathcal{I}' \subseteq \mathcal{I}$, $\mathsf{ICQA}^{\mathsf{IM}}_{\mathsf{X}}(q, \mathsf{ans}, \mathcal{I}', \Delta) \leq \mathsf{ICQA}^{\mathsf{IM}}_{\mathsf{X}}(q, \mathsf{ans}, \mathcal{I}, \Delta)$. Intuitively, monotonicity ensures that an answer cannot become more inconsistent after removing tuples from $\mathcal{I}$. This is desired because inconsistency stems from constraint violations, and removing data from the database can only eliminate or reduce such violations.

Proposition 9. $\mathsf{ICQA}^{\mathsf{IM}}_{\mathsf{X}}$ *satisfies Blamelessness for any explainability method* X *and tuple inconsistency measure* IM*. Moreover,* $\mathsf{ICQA}^{\mathsf{IM}}_{\mathsf{Prov}}$ *satisfies Additivity for any* IM*, and Monotonicity if* IM *satisfies Data Monotonicity.*

5 Computational Complexity Analysis

We study the following complexity problems, given $k \geq 0$:

TUPLE-MEASURE: is $\mathsf{IM}(t, \mathcal{I}, \Delta) > k$?

ANSWER-MEASURE: is $\mathsf{ICQA}^{\mathsf{IM}}_{\mathsf{X}}(q, \mathsf{ans}, \mathcal{I}, \Delta) > k$?

For each denial constraint $\delta \in \Delta$, we assume that $\mathsf{MIS}(\mathcal{I}, \delta)$ is given; it can be computed efficiently by evaluating the conjunctive query q^δ associated to δ.

To facilitate our analysis, we introduce the following terms:
(I) *MisConstant*: $|\mathsf{MIS}(\mathcal{I}, \Delta)| < c$ for some constant c,
(II) *FixedSize-Disj*: for each $M_i \neq M_j \in \mathsf{MIS}(\mathcal{I}, \Delta)$, $|M_i| = |M_j|$ and $|M_i \cap M_j| \leq 1$.

5.1 Complexity of Tuple Measures

[Livshits and Kimelfeld, 2022] prove that $\mathsf{IM}^{P}_{\mathsf{shap}}$ and $\mathsf{IM}^{\mathsf{MIS}}_{\mathsf{shap}}$ are in PTime for functional dependencies. Our next result extends this to denial constraints for which either MisConstant or FixedSize-Disj holds. Indeed, if Δ contains only functional dependencies, then $\mathsf{MIS}(\mathcal{I}, \Delta)$ satisfies FixedSize-Disj as $|M_i| = 2$, which implies $|M_i \cap M_j| \leq 1$.

Proposition 10. $\mathsf{IM}^{P}_{\mathsf{shap}}$ *and* $\mathsf{IM}^{\mathsf{MIS}}_{\mathsf{shap}}$ *are* **PTime** *if* $\mathsf{MIS}(\mathcal{I}, \Delta)$ *satisfies MisConstant. Moreover,* $\mathsf{IM}^{\mathsf{MIS}}_{\mathsf{shap}}$ *is* **PTime** *if FixedSize-Disj holds.*

The case of $\mathsf{IM}^{P}_{\mathsf{shap}}$ follows analogously to [Livshits and Kimelfeld, 2022]. For $\mathsf{IM}^{\mathsf{MIS}}_{\mathsf{shap}}$, the minimal inconsistent subsets contributing to its value are of size two for functional dependencies, whereas they can be of arbitrary size for denial constraints. We can prove that deciding such subsets remains polynomial under MisConstant and FixedSize-Disj.

Proposition 11. IM *is in* **PTime** *for* $\mathsf{IM} \in \{\mathsf{IM}_d, \mathsf{IM}_{\#}, \mathsf{IM}_P, \mathsf{IM}_{\mathsf{MIS}}\}$*, and* **coNP-Complete** *for* $\mathsf{IM}_{\mathsf{dr}}$ *unless* $|\mathsf{MIC}(\mathcal{I}, \Delta)| < c$ *for a constant c.*

Given $\mathsf{MIS}(\mathcal{I}, \Delta)$, $\mathsf{IM}_d, \mathsf{IM}_{\#}, \mathsf{IM}_P$ and $\mathsf{IM}_{\mathsf{MIS}}$ are computed by scanning the set MIS. For $\mathsf{IM}_{\mathsf{dr}}$, MIC corresponds to minimal hitting sets of $\mathsf{MIS}(t, \mathcal{I}, \Delta)$; deciding the existence of a bounded one is NP-complete, yielding coNP-completeness. If $|\mathsf{MIC}|$ is bounded, enumeration is polynomial.

Proposition 12. $\mathsf{IM}_{\mathsf{CIM}}$ *and* $\mathsf{IM}_{\mathsf{PIM}}$ *are* **PTime***, and* $\mathsf{IM}_{\mathsf{RIM}}$ *is* **NP-hard**.

$\mathsf{IM}_{\mathsf{CIM}}$ and $\mathsf{IM}_{\mathsf{PIM}}$ are computable by polynomial scans on violation sets. NP-hardness of $\mathsf{IM}_{\mathsf{RIM}}$ follows from the causal responsibility for conjunctive queries [Meliou *et al.*, 2010].

5.2 Complexity of Answer Measures

Recall that an ICQA measure depends on the tuple measure and on the aggregation method of each tuple contribution to the query answer. For conjunctive queries with equality comparisons only, the following tractability result holds.

Proposition 13. $\mathsf{ICQA}^{\mathsf{IM}}_{\mathsf{Prov}}$ *has the same complexity as* IM. *In particular,* $\mathsf{ICQA}^{y}_{\mathsf{Prov}}$ *are in* **PTime** *for* $y \in \{\mathsf{IM}^{\mathsf{MIS}}_{\mathsf{shap}}, \mathsf{IM}^{P}_{\mathsf{shap}}\}$.

$\mathsf{ICQA}^{\mathsf{IM}}_{\mathsf{Prov}}$ aggregates tuple scores over $\mathsf{WhyP}(\mathcal{I}, q, \mathsf{ans})$, which is PTime computable. Since aggregation is linear, the complexity matches that of IM.

Proposition 14. *For a hierarchical boolean conjunctive query* without self-joins *(see [Livshits* et al.*, 2021a]), the complexity of* $\mathsf{ICQA}^{\mathsf{IM}}_{\mathsf{shap}}$ *matches that of the underlying measure* IM. *Otherwise, it is* $\mathbf{FP}^{\#\mathbf{P}}$**-complete**.

The computation reduces to the Shapley value of a Boolean query q' induced by ans. By [Livshits and Kimelfeld, 2022], this is PTime for hierarchical queries without self-joins and $\mathbf{FP}^{\#\mathbf{P}}$-complete otherwise; the overall complexity follows.

Proposition 15. $\mathsf{ICQA}^{\mathsf{IM}}_{\mathsf{Resp}}$ *is* **NP-hard** *in general, regardless* IM. *It becomes* **PTime** *for linear queries when combined with tractable* IM *(see Propositions 10,11,12).*

This result follows from [Meliou *et al.*, 2010], where responsibility is NP-hard in general and tractable for linear queries; combining with tractable IM yields the result.

6 Experimental Evaluation

The experiments are designed to assess three aspects of the proposed inconsistency measures. First, we examine their ability to discriminate among tuples w.r.t. their involvement in violations under varying inconsistency ratios and different sets of constraints. Second, we analyze the propagation of tuple inconsistency to query answers under multiple aggregation methods, assessing whether the resulting ICQA scores remain meaningful and stable. Third, we investigate the computational efficiency and scalability of the different measures.

All experiments are conducted on macOS 15.6.1 (M1 Max, 32GB RAM) with Python 3.11.14, DuckDB 1.4.3 as a DBMS, and TPC-H (https://www.tpc.org/tpch/) dbgen v3.0.1 for datasets generation. The source code of our experiments is available at https://github.com/yrGnote/tuple-measure-icqa.

Datasets. First, we generate database instances with scale factors SF $\in \{0.01, 0.05, 0.1\}$, yielding 'clean' datasets with 86.8k, 432.8k and 866.6k tuples, respectively. Second, we define two sets of denial constraints over the TPC-H schema, namely $\Delta_1 = \{\delta_1, \delta_2\}$ and $\Delta_2 = \{\delta_1, \delta_2, \delta_3, \delta_4\}$, where:

$\delta_1 : \neg\big(L(k_o, k_p, k_s, d_{\text{ship}}, d_{\text{receipt}}, d_{\text{cmt}}, s_l) \wedge d_{\text{receipt}} < d_{\text{ship}}\big)$

$\delta_2 : \neg\big(L(k'_o, k_p, k_s, d_{\text{ship}}, d_{\text{receipt}}, d_{\text{commit}}, s_l) \wedge O(k_o, d_{\text{order}}, s_o) \wedge k_o = k'_o \wedge d_{\text{commit}} < d_{\text{order}}\big)$

$\delta_3 : \neg\big(O(k_o, d_{\text{order}}, s_o) \wedge L(k'_o, k_p, k_s, d_{\text{ship}}, d_{\text{recp}}, d_{\text{cmt}}, s_l) \wedge k_o = k'_o \wedge s_o = \text{'F'} \wedge s_l \neq \text{'F'}\big)$

$\delta_4 : \neg\big(L(k_o, k_p, k_s, d_{\text{ship}}, d_{\text{recpt}}, d_{\text{cmt}}, s_l) \wedge PS(k'_p, k'_s, q_{\text{avl}}) \wedge P(k''_p) \wedge k_p = k'_p = k''_p \wedge k_s = k'_s \wedge q_{\text{avl}} < 0\big)$

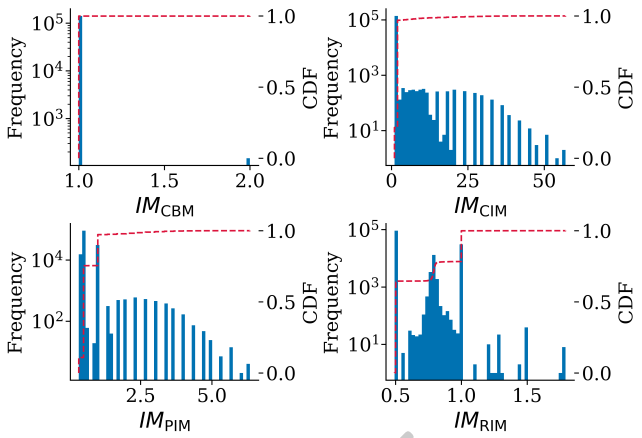

Figure 1: Distribution of tuple inconsistency measures.

Starting from the three clean TPC-H instances, we inject synthetic violations of Δ_1 and Δ_2, choosing to modify each time, a single attribute appearing in the condition of each δ as follows: for δ_1, we decrease the receipt date to enforce $d_{\text{receipt}} < d_{\text{ship}}$; for δ_2 we set $d_{\text{commit}} = d_{\text{order}} - 1$; for δ_3, we flip the lineitem status attribute s_l from F to O and for δ_4, we assign a negative available quantity q_{avl}. Violations are injected on a sampled subset of tuples with ratios in $\{0.01\%, 0.05\%, 0.1\%\}$, and sampling is repeated using 10 random seeds (affecting only tuple selection). Combining the 3 database scale factors (SF$\in\{0.01, 0.05, 0.1\}$), the 2 constraint subsets (Δ_1, Δ_2), the 3 inconsistency ratios ($0.01\%, 0.05\%, 0.1\%$), and the 10 random sampling seeds, we finally obtain *180 test databases*. We compute MIS for each δ by executing the SQL query q^{δ} (see Section 2).

6.1 Evaluation of Tuple Inconsistency Measures

We evaluate our three tuple inconsistency measures introduced in Section 3.1 and the competitor $\mathsf{IM}_{\mathsf{CBM}}$ over all 180 databases. While $\mathsf{IM}_{\mathsf{CIM}}$, $\mathsf{IM}_{\mathsf{PIM}}$, and $\mathsf{IM}_{\mathsf{CBM}}$ are obtained by direct computation over the set of MIS, for $\mathsf{IM}_{\mathsf{RIM}}$ the contingency set for each tuple t and each constraint equals to a minimum hitting set problem over the set of MIS that does not contain t. We use the CP-SAT solver for computing minimum hitting sets [Perron *et al.*, 2023]. Time-limited solving gives a feasible solution and thus an approximate $\mathsf{IM}_{\mathsf{RIM}}$ value.

Measure Value Distribution. Figure 1 compares the distributions of all non-zero scored tuples from all databases w.r.t. the different tuple measures. The values of $\mathsf{IM}_{\mathsf{CBM}}$ are highly concentrated at 1, with a small fraction at 2. In contrast, $\mathsf{IM}_{\mathsf{CIM}}$ has a distribution with high density near the origin with a wide dynamic range, resulting from multiplying the number of violated conditions with the number of MIS the tuple participates in. The measures $\mathsf{IM}_{\mathsf{PIM}}$ and $\mathsf{IM}_{\mathsf{RIM}}$ lie between $\mathsf{IM}_{\mathsf{CBM}}$ and $\mathsf{IM}_{\mathsf{CIM}}$: both retain variation in the scores (as opposed to $\mathsf{IM}_{\mathsf{CBM}}$), while avoiding the heavy heads of $\mathsf{IM}_{\mathsf{CIM}}$. Among the four measures, $\mathsf{IM}_{\mathsf{PIM}}$ shows the most balanced distributional footprint (frequency and CDF). $\mathsf{IM}_{\mathsf{RIM}}$ also has a smooth CDF, but is not as efficient as $\mathsf{IM}_{\mathsf{PIM}}$.

To further quantify a measure's discrimination ability, we

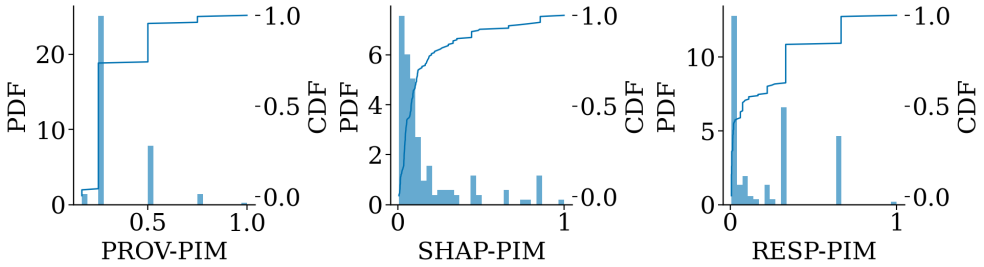

Figure 2: Distribution of $\mathsf{ICQA}^{\mathsf{PIM}}_{\mathsf{X}}$ with $\mathsf{X} \in \{\mathsf{Prov}, \mathsf{shap}, \mathsf{Resp}\}$.

compute its (Gini coef/std) over all non-zero tuples across 180 databases. The results are: $\mathsf{IM}_{\mathsf{CBM}}$ (0.001/0.03), $\mathsf{IM}_{\mathsf{CIM}}$ (0.25/2.81), $\mathsf{IM}_{\mathsf{RIM}}$ (0.16/0.21), and $\mathsf{IM}_{\mathsf{PIM}}$ (0.23/0.42). The measure $\mathsf{IM}_{\mathsf{CBM}}$ is not very informative, $\mathsf{IM}_{\mathsf{CIM}}$ is quite unstable, and $\mathsf{IM}_{\mathsf{RIM}}$ is conservative. Interestingly, $\mathsf{IM}_{\mathsf{PIM}}$ provides the best balance between discrimination and dispersion.

Ranking Comparison. We further compare the four tuple measures through their induced inconsistent tuple rankings. For this, we use the *Spearman's correlation* ρ between two rankings. On the database with the setting sf0.1-Δ_1-0.1%-seed01, $\mathsf{IM}_{\mathsf{PIM}}$ and $\mathsf{IM}_{\mathsf{RIM}}$ are almost rank-equivalent ($\rho \approx 0.99$), $\mathsf{IM}_{\mathsf{CBM}}$ is nearly independent from the other measures ($\rho \approx 0.07$), and $\mathsf{IM}_{\mathsf{CIM}}$ induces an almost reversed ordering w.r.t. $\mathsf{IM}_{\mathsf{PIM}}$ and $\mathsf{IM}_{\mathsf{RIM}}$ ($\rho \approx -0.98$). Only two tuples appear in all four rankings, suggesting that they are consistently identified as central contributors to inconsistency. The variation in the rankings of the other tuples reflects their different modes of involvement in inconsistencies.

Efficiency. Across the 180 databases, the average runtimes (std) are 0.0017s (0.0020) for $\mathsf{IM}_{\mathsf{CBM}}$, 0.0032s (0.0026) for $\mathsf{IM}_{\mathsf{CIM}}$, 0.0019s (0.0028) for $\mathsf{IM}_{\mathsf{PIM}}$, and 15.1653s (32.6909) for $\mathsf{IM}_{\mathsf{RIM}}$. $\mathsf{IM}_{\mathsf{CBM}}$, $\mathsf{IM}_{\mathsf{CIM}}$, and $\mathsf{IM}_{\mathsf{PIM}}$ are consistently fast and scalable, in contrast to $\mathsf{IM}_{\mathsf{RIM}}$. Overall, $\mathsf{IM}_{\mathsf{PIM}}$ has the best quality-versus-efficiency trade-off.

6.2 Evaluation of ICQA Measures

Our goal now is to assess whether the answer level ICQA scores constitute an indicator for distinguishing inconsistent answers, and how different aggregations and tuple measures jointly affect the distributional footprint, score resolution, and ranking of answers. Note that for Shapley-based ICQA, exact values are computed by enumerating all coalitions when $|\mathcal{I}'| \leq 12$; otherwise, they are approximated via Monte Carlo sampling with 10,000 permutations using a fixed seed.

We evaluate 5 SPJ queries over the TPC-H schema, with increasing structural complexity, ranging from single-table selections to multi-way joins involving up to 5 relations. For each query, 12 instantiated answer measures are computed over all 180 databases. Each instantiation yields a distribution over query answers, reflecting both the degree of inconsistency in the data and the selectivity of the measure.

Figure 2 presents normalized ICQA distributions (histograms and CDFs) for different aggregations combined with $\mathsf{IM}_{\mathsf{PIM}}$, our most successful tuple inconsistency measure, based also on the previous experiments. Similar patterns are observed for the other three tuple measures. We observe that the explainability methods Prov and Resp concentrate ICQA mass on a few discrete values, resulting in steep CDF jumps and poor ranking discrimination. In contrast, shap yields high density near the origin distributions but with gradual CDF growth, thereby enabling more fine-grained differentiation.

The total time for computing all four tuple inconsistency measures for all tuples is 2730.98 seconds. Given the values for the tuple measures, the total time for computing ICQA over all answers for the five queries is: 17.13s for $\mathsf{ICQA}_{\mathsf{Prov}}$, 25.12s for $\mathsf{ICQA}_{\mathsf{shap}}$, and 1118.77s for $\mathsf{ICQA}_{\mathsf{Resp}}$. Thus, $\mathsf{ICQA}_{\mathsf{Prov}}$ and $\mathsf{ICQA}_{\mathsf{shap}}$ are quite fast, given the precomputed tuple measures. Considering both efficiency and discrimination, shap is the best aggregation method for ICQA.

7 Related Work and Conclusions

For tuple measures, both the Shapley-based approach [Livshits *et al.*, 2021a] and the KB measures [Raddaoui *et al.*, 2024] compute a tuple's average marginal contribution to constraint violations, providing finer granularity than $\mathsf{IM}_{\mathsf{CBM}}$ [Issa *et al.*, 2020]. However, they still treat all tuples and constraints uniformly. Our approach advances this line of work by introducing provenance- and responsibility-based measures capturing nuanced violation weights at the tuple level, for more precise inconsistency scores. In particular, $\mathsf{IM}_{\mathsf{PIM}}$ satisfies all the desired properties, admits PTime computation, and shows promising empirical performance. Moreover, unlike [Ribeiro and Thimm, 2021], which reveals implicit inconsistencies via logical reasoning, our $\mathsf{IM}_{\mathsf{CIM}}$ measure focuses on *explicit* inconsistencies, a design that aligns with the absence of implicit derivations in typical databases.

For query answer measures, Shapley-based approaches have been applied to evaluate tuples' marginal contributions to query results, ranging from basic expectation of query answer counts [Reshef *et al.*, 2020; Livshits *et al.*, 2021a] to generalized weighted expected sums of minimal supports [Bienvenu *et al.*, 2025]. Moreover, [Meliou *et al.*, 2010] proposes to explain query (non)-answers based on *actual causes* and *responsibility* of tuples. In contrast, measuring *inconsistency* in an answer remains largely unaddressed. The closest approach ranks answers by inconsistency [Issa *et al.*, 2020], which we include as a specific instance of our ICQA framework. Notably, our framework yields tractable measures (e.g. $\mathsf{ICQA}^{\mathsf{PIM}}_{\mathsf{Prov}}$) that fulfill desirable criteria. Our experiments show that the $\mathsf{IM}_{\mathsf{PIM}}$ and $\mathsf{ICQA}^{\mathsf{IM}}_{\mathsf{shap}}$ measures provide useful discrimination for tuples and answers, respectively, while showing promising scalability.

Acknowledgments

This work is supported by the National French Agency, under the EXPIDA-ANR-22-CE23-0017 project.

References

[Besnard and Grant, 2020] Philippe Besnard and John Grant. Relative inconsistency measures. *Artificial Intelligence (AIJ)*, 280:103231, 2020.

[Bienvenu *et al.*, 2025] Meghyn Bienvenu, Diego Figueira, and Pierre Lafourcade. Shapley revisited: Tractable responsibility measures for query answers. *Proceedings of the ACM on Management of Data*, 3(2):1–26, 2025.

[Buneman *et al.*, 2001] Peter Buneman, Sanjeev Khanna, and Tan Wang-Chiew. Why and where: A characterization of data provenance. In *ICDT*, pages 316–330, 2001.

[Corea and Thimm, 2020] Carl Corea and Matthias Thimm. Towards inconsistency measurement in business rule bases. In *ECAI*, pages 704–711, 2020.

[Corea *et al.*, 2024] Carl Corea, Isabelle Kuhlmann, Matthias Thimm, and John Grant. Paraconsistent reasoning for inconsistency measurement in declarative process specifications. *Inf. Syst.*, 122:102347, 2024.

[Hunter and Konieczny, 2010] Anthony Hunter and Sébastien Konieczny. On the measure of conflicts: Shapley inconsistency values. *Artificial Intelligence*, 174(14):1007–1026, 2010.

[Issa *et al.*, 2020] Ousmane Issa, Angela Bonifati, and Farouk Toumani. Evaluating Top-k queries with inconsistency degrees. *PVLDB*, 13(12):2146–2158, 2020.

[Kuhlmann *et al.*, 2025] Isabelle Kuhlmann, Anna Gessler, Vivien Laszlo, and Matthias Thimm. Comparison of SAT-based and ASP-based algorithms for inconsistency measurement. *J. Artif. Intell. Res.*, 82:563–685, 2025.

[Livshits and Kimelfeld, 2022] Ester Livshits and Benny Kimelfeld. The shapley value of inconsistency measures for functional dependencies. *Logical Methods in Computer Science*, 18, 2022.

[Livshits *et al.*, 2021a] Ester Livshits, Leopoldo E. Bertossi, Benny Kimelfeld, and Moshe Sebag. The shapley value of tuples in query answering. *Log. Methods Comput. Sci.*, 17(3), 2021.

[Livshits *et al.*, 2021b] Ester Livshits, Rina Kochirgan, Segev Tsur, Ihab F. Ilyas, Benny Kimelfeld, and Sudeepa Roy. Properties of inconsistency measures for databases. In *SIGMOD*, page 1182–1194, 2021.

[Meliou *et al.*, 2010] Alexandra Meliou, Wolfgang Gatterbauer, Katherine F. Moore, and Dan Suciu. The complexity of causality and responsibility for query answers and non-answers. *CoRR*, abs/1009.2021, 2010.

[Mu, 2018] Kedian Mu. Measuring inconsistency with constraints for propositional knowledge bases. *Artificial Intelligence*, 259:52–90, 2018.

[Parisi and Grant, 2020] Francesco Parisi and John Grant. On measuring inconsistency in relational databases with denial constraints. In *ECAI*, pages 857–864, 2020.

[Parisi and Grant, 2023] Francesco Parisi and John Grant. On measuring inconsistency in definite and indefinite databases with denial constraints. *Artificial Intelligence*, 318:103884, 2023.

[Perron *et al.*, 2023] Laurent Perron, Frédéric Didier, and Steven Gay. The CP-SAT-LP solver (invited talk). In *CP*, pages 3:1–3:2, 2023.

[Raddaoui *et al.*, 2024] Badran Raddaoui, Christian Straßer, and Saïd Jabbour. Towards a principle-based framework for assessing the contribution of formulas on the conflicts of knowledge bases. In *IJCAI*, pages 3541–3548, 2024.

[Reiter, 1987] Raymond Reiter. A theory of diagnosis from first principles. *Artif. Intell.*, 32(1):57–95, 1987.

[Reshef *et al.*, 2020] Alon Reshef, Benny Kimelfeld, and Ester Livshits. The impact of negation on the complexity of the shapley value in conjunctive queries. In *PODS*, pages 285–297, 2020.

[Ribeiro and Thimm, 2021] Jandson S. Ribeiro and Matthias Thimm. Consolidation via tacit culpability measures: Between explicit and implicit degrees of culpability. In *KR*, pages 529–538, 2021.

[Shapley, 1953] Lloyd S Shapley. A value for n-person games. In *Contributions to the Theory of Games II*, pages 307–317. Princeton University Press, 1953.

[Straßer *et al.*, 2025] Christian Straßer, Badran Raddaoui, and Saïd Jabbour. Decomposing inconsistencies: Marginal contributions and pooling techniques. In *IJCAI*, pages 4687–4695, 2025.