# Summarizing Provenance of Aggregate Query Results in Relational Databases

Omar AlOmeir*, Eugenie Yujing Lai*, Mostafa Milani*, Rachel Pottinger*
*University of British Columbia
{oomeir, abyujing, mkmilani, rap}@cs.ubc.ca

***Abstract*—Data provenance is any information about the origin of a piece of data and the process that led to its creation. Most database provenance work has focused on creating models and semantics to query and generate this provenance information. While comprehensive, provenance information remains large and overwhelming, which can make it hard for data provenance systems to support data exploration.**

**We present a new approach to provenance exploration that builds on data summarization techniques. We contribute novel summarization schemes for the provenance of aggregation queries. We conduct thorough experiments to show the feasibility and relevance of our approaches.**

***Index Terms*—provenance, relational databases, summarization, visualization, usability**

## I. Introduction

Provenance has been a focus of research in a number of different areas. In the scientific work-flow context, provenance is used to show the processes that data went through. Thus, provenance work in work-flow management has included work on presenting provenance information to users. In contrast, in database research, most of the focus has been on finding the sources that contributed to the results of a query. This has included a lot of work on developing comprehensive models and representations of data provenance. However, there is little research that focuses on presenting database provenance information in a way that is not overwhelming to users. Using existing provenance database systems can be overwhelming without prior knowledge of provenance models or the data itself. Furthermore, provenance information can increase the data size exponentially. This can have serious implications on both storage and usability. We argue that data provenance needs to be summarized to support broad exploration.

This paper studies provenance summarization and exploration for aggregate query results in relational databases. We focus on aggregate queries because they are common in database applications and their provenance poses a greater challenge compared to other types of queries since it is usually large and difficult to explore. We use the following running example to explain the problem and our solution.

**Example 1.** Consider a user who starts by posing the following aggregate SQL query, $Q_1$, over *MoviesDirectors* in Table I that is a small subset of the IMDB dataset:[1]

$Q_1$ : SELECT Gender, SUM(Rev) AS SumRev
FROM MoviesDirectors GROUP BY Gender

[1]https://developer.imdb.com

The query asks the sum of the revenue of movies directed by directors of different genders. The result is in Table II. For answer $a_2$, highlighted in Table II, the provenance only contains 10 tuples with all the male directors and the movies that they directed. However, in practice, querying more than 4 million movies in the entire IMDB dataset would yield a much larger number of tuples, which would hinder free-form exploration. Thus, instead of the full provenance, the user would benefit from a summary. □

Our provenance summaries consist of rules from [1]. These rules were introduced for summarizing relational data and are applied in several frameworks for other purposes [2], [3], [4] (see Section VII for detail). Before we explain our provenance summaries, we briefly review these rules and the summaries in [1]. We refer to those summaries as *basic summaries*.

**Example 2.** A *summarization rule* over a relation is a tuple of the same schema with some wild-card values, $\star$s, that match any value in the relation and enable the rule to cover and summarize several tuples [1]. For example, rule $s_1$ in Table IV covers movies with rating 7 directed by male directors, i.e. $t_1, t_3, t_5, t_7, t_{11}$ in Table I. A *basic summary* of a relation is a set of $k$ rules, each of which summarizes some "interesting" aspects of the relation. Table IV is a basic summary of Table I with $k = 3$, where we omitted the attributes with $\star$s across different rules, e.g. *Title*, *Rev*, and *Director*. The *score* of a rule depends on its *weight* and *coverage*. The *weight* specifies how informative the rule is and is measured by the number of non-$\star$ values in the rule, whereas *coverage* is the number of tuples in the relation covered by the rule [1]. For example, the coverage and weight of $s_1$ are 5 and 2, and its score is $5 \times 2$. In Table IV, the coverage percentages are shown as an attribute. □

We will build on these basic summaries and the summarization rules in Example 2 and present two types of provenance summaries, *impact summaries* and *comparative summaries*, which we explain using the running example.

While basic summaries are effective in summarizing relational data in general, they are not as helpful in summarizing provenance of aggregate queries. Specifically, the score of a rule in a basic summary is independent of how much the tuples covered by the rule contribute to the aggregate result in the answer. We propose *impact summaries* to resolve this issue.

**Example 3.** Table V shows an impact summary of the

| ID | Title | Year | Genre | Rating | Rev (M$) | Director | Gender | Country |
|---|---|---|---|---|---|---|---|---|
| $t_1$ | Lincoln | 2012 | Drama | 7 | 275.3 | Steven Spielberg | Male | US |
| $t_2$ | Sicario | 2014 | Drama | 8 | 84.9 | Denis Villeneuve | Male | US |
| $t_3$ | Crimson Peak | 2014 | Horror | 7 | 74.7 | Guillermo del Toro | Male | Mexico |
| $t_4$ | Bonjour Anne | 2014 | Comedy | 5 | 8.9 | Eleanor Coppola | Female | US |
| $t_5$ | The Terminal | 2004 | Comedy | 7 | 219.4 | Steven Spielberg | Male | US |
| $t_6$ | MMFR | 2015 | Action | 8 | 378.9 | George Miller | Male | Australia |
| $t_7$ | Pacific Rim | 2013 | Action | 7 | 411.5 | Guillermo del Toro | Male | Mexico |
| $t_8$ | La La Land | 2015 | Drama | 8 | 446.1 | Damien Chazelle | Male | US |
| $t_9$ | Mamma Mia | 2008 | Comedy | 6 | 615.7 | Phyllida Lloyd | Female | US |
| $t_{10}$ | Rogue One | 2015 | Action | 8 | 1.056 B$ | Gareth Edwards | Male | UK |
| $t_{11}$ | Superbad | 2007 | Comedy | 7 | 169.9 | Greg Mottola | Male | US |
| $t_{12}$ | Annabelle | 2014 | Horror | 6 | 257.1 | John R. Leonetti | Male | US |

TABLE I: MoviesDirectors table

| ID | Gender | SumRev (M$) |
|---|---|---|
| $a_1$ | Female | 624.6 |
| $a_2$ | Male | 3.374 B$ |

TABLE II: Results of $Q_1$

| ID | Gender | Genre | Count |
|---|---|---|---|
| $a_3$ | Male | Drama | 3 |
| $a_4$ | Male | Horror | 2 |
| $a_5$ | Female | Comedy | 2 |
| $a_6$ | Male | Comedy | 2 |
| $a_7$ | Male | Action | 3 |
| $a_8$ | Male | Comedy | 2 |

TABLE III: Results of $Q_2$

| ID | Year | Genre | Rating | Gender | Country | Coverage |
|---|---|---|---|---|---|---|
| $s_1$ | ⋆ | ⋆ | 7 | Male | ⋆ | 41.67 |
| $s_2$ | ⋆ | Drama | ⋆ | Male | US | 25.00 |
| $s_2$ | 2014 | ⋆ | ⋆ | Male | ⋆ | 25.00 |

TABLE IV: A basic summary

| ID | Year | Genre | Rating | Gender | Country | Coverage | Impact |
|---|---|---|---|---|---|---|---|
| $s_4$ | ⋆ | Action | 8 | Male | ⋆ | 20.00 | 43.49 |
| $s_5$ | ⋆ | Comedy | 7 | Male | US | 20.00 | 11.53 |
| $s_6$ | 2015 | ⋆ | ⋆ | Male | ⋆ | 30.00 | 55.74 |

TABLE V: An impact summary

| ID | Rating | Director | Gender | Country | Cvg $a_3$ | Cvg $a_8$ |
|---|---|---|---|---|---|---|
| $c_1$ | ⋆ | Steven Spielberg | Male | US | 33.34 | 50.00 |
| $c_2$ | 7 | ⋆ | Male | US | 66.67 | 50.00 |

TABLE VI: A comparative summary

provenance of $a_2$ from Table II. Rules in this summary are preferred to the rules in the basic summary from Example 2 if we choose to consider the impact of movies' revenues on $a_2[SumRev]$. For example, $s_1$ covers more movies than $s_4$, though the movies in $s_4$ have higher revenue. This means $s_4$ covers movies with higher impact on the answer $a_2[SumRev]$ compared to the movies covered by $s_1$. In impact summaries, the score of a rule depends on its weight, coverage, and a new impact factor. The impact percentages of the rules in Table V are shown with a new attribute. Impact summaries contain rules that consider the impact of the provenance tuples on the aggregate query answer, e.g. $s_4$. They are different from basic summaries with rules that cover more tuples, e.g. $s_1$. □

We also introduce a second type of provenance summaries, *comparative summaries*, that highlight the similarities between the provenance of two answers to an aggregate query and apply the summarization rules.

**Example 4.** To explain comparative summaries, consider the following aggregate query, $Q_2$, that asks the number of directors from each gender and movie genre.

```
Q2 : SELECT Gender, Genre, COUNT(⋆) AS CountM
     FROM MoviesDirectors GROUP BY Gender, Genre
```

The result is shown in Table III. Consider two highlighted answer tuples $a_3$ and $a_8$, that specify the number of male directors of drama and comedy movies. Table VI is a comparative summary with rules that cover tuples from the provenance of both answers. The score of a rule in such summaries depends on how many tuples it summarizes from the provenance of both selected answers. Comparative summaries can help users find similar tuples that contributed to the query answers. In Table VI, the new attributes show the coverage percentages of the tuples from the provenance of $a_3$ and $a_8$, which helps the user quickly identify directors who contribute to both genres. For example, $c_1$ is a rule that shows *Steven Spielberg* has movies in both genres. Then the user could reason over if such directors should be counted twice, removed, or counted in only one genre. Comparative summaries differ from basic summaries as they summarize provenance tuples from two sets of tuples, unlike basic summaries that summarize a single set. Note that comparative summaries cannot be reduced to basic summaries by using the union of the two sets. In particular, if the two sets are not of similar size, the basic summaries will be dominated by rules that cover the larger set. □

To present provenance summaries, i.e. impact and comparative summaries, to the user we implement the interactive smart drill-down operator in [1] that allows the user to explore data by using data summarization. In our running example, the user can select an answer for impact summary, or a pair of answers for comparative summaries, and receive a summary. Then the user can continue to explore the provenance data by selecting one of the rules and apply the drill-down operator that shows *super-rules* with fewer ⋆s and reveal more detail of the provenance covered by the selected rule. Unlike [1] that applies drill-down for basic summaries and over one relation, we use them for provenance summaries that may involve multiple relations (cf. Section V for detail).

**Relationship with Other Work.** The most similar work to ours is [5] in which the provenance of an aggregate query is represented as a polynomial expression, i.e. how-provenance [6]. A provenance summary is a shorter polynomial expression that highlights the important aspects of the original provenance expression. The quality of a summary depends on general measures such as its size and its distance

with the original expression [5]. The algorithm for generating these summaries does not involve the user and fails to capture her interest [5]. Also, the summaries are generated independent of the impact of provenance tuples on the aggregate value in the query answer. One can also argue that the expressions in these summaries are hard to understand for non-expert users.

The impact summaries are related to the query explanation work [7], [8], [9]. In particular, we define the impact of summarization rules using *sensitivity analysis* [10], which is similar to the notions of *intervention* [8], [9] and *influence* [7] used in the query explanation literature. However, our work is fundamentally different from the query explanation work since our work focuses on data summarization and exploration, whereas their objective is to explain a query result, such as an unexpected answer or an outlier. More specifically, query explanation is more related to the impact evaluation in our summaries and does not consider the coverage of provenance data and the interactive exploration that are fundamental in having good provenance summaries.

**Contributions.** Our contributions are as follows:

- We define provenance summaries, impact summaries and comparative summaries, for general form of aggregate SQL queries. We formalize score functions for these summaries and present the summarization problems for finding the best provenance summaries (Section III).
- We present efficient algorithms for producing summaries with high scores (Section IV).
- We present novel interaction methods and visualization techniques in a user interface (Section V).
- We conduct thorough experiments and a user survey to show that our proposed solutions are feasible and useful (Section VI).

## II. Background

Our work builds on previous research in database provenance and data summarization, which we review in this section. We first present some preliminary notations. A relation $R$ with a set of attributes (relation schema) $\mathcal{R} = \{A_1, ..., A_m\}$ is a finite set of $m$-ary tuples $\{t_1, ..., t_n\}$. Each relation is represented as a table, each of which has columns and contains rows. We occasionally abuse notation by saying that tables contain tuples or have attributes. A database $D$ with database schema $\mathcal{S} = \{\mathcal{R}_1, ..., \mathcal{R}_N\}$ is a finite set of relations $R_1, ..., R_N$ with schemas $R_j \in \mathcal{R}_j$. We denote the value in the $i$-th position in tuple $t$ in $D$ as $t[A_i]$. $Q(D)$ is the set of answers to a query $Q$ over database $D$. For attribute $A_i \in \mathcal{R}$, $Dom(A_i)$ is the domain of values in $A_i$. Table VII summarizes our symbols and notation.

### A. Provenance

Data provenance is defined as provenance information related to the data itself. To show what we mean by data provenance in the relational context, we introduce a running example. Consider the following query:

$Q_3$ : SELECT d.Gender, m.Genre FROM
Movies m JOIN Directors d ON m.Director = d.Name

TABLE VII: Summary of symbols and notations

| Symbol | Description |
|---|---|
| $R, \mathcal{R}$ | relation and relational schema |
| $A, B, ..., G$ | relational attributes |
| $t, r$ | tuples |
| $a, b$ | tuples in query answer |
| $Q, Q(D)$ | database, query, query answer |
| $D, \mathcal{S}$ | database (instance), database schema |
| $s, c$ | rules ($c$ for comparative rules) |
| $S$ | set or list of rules |
| $Dom(A)$ | domain of attribute $A$ |
| $\star$ | wildcard character |
| $R^a$ | provenance of answer $a$ in $R$ |
| $Q^a$ | query that returns aggregate value in $a$ |
| $m_w$ | maximum rule weight |
| $m_D, m_q$ | number of attributes, number of group-by attributes |

Applying $Q_3$ to the data in Tables VIII-IX results in Table X.

| ID | Title | Year | Genre | Rating | Rev (M$) | Director |
|---|---|---|---|---|---|---|
| $r_1$ | Lincoln | 2012 | Drama | 7 | 275.3 | Steven Spielberg |
| $r_2$ | Sicario | 2015 | Drama | 8 | 84.9 | Denis Villeneuve |
| $r_3$ | Crimson Peak | 2015 | Horror | 7 | 74.7 | Guillermo del Toro |
| $r_4$ | Bonjour Anne | 2016 | Comedy | 5 | 8.9 | Eleanor Coppola |
| $r_5$ | The Terminal | 2004 | Comedy | 7 | 219.4 | Steven Spielberg |

TABLE VIII: Movies table

| ID | Name | Gender | Country |
|---|---|---|---|
| $r_6$ | Steven Spielberg | Male | US |
| $r_7$ | Denis Villeneuve | Male | Canada |
| $r_8$ | Guillermo del Toro | Male | Mexico |
| $r_9$ | Eleanor Coppola | Female | US |

TABLE IX: Directors table

| ID | Gender | Genre |
|---|---|---|
| $b_1$ | Male | Drama |
| $b_2$ | Male | Horror |
| $b_3$ | Female | Comedy |
| $b_4$ | Male | Comedy |

TABLE X: $Q_3$'s answer

The goal of data provenance is finding the source of a piece of data and the process it went through to make it into the results of a query. More precisely, given a database $D$, a query $Q$, and the result of applying the query to the database $Q(D)$, the provenance problem in databases is to find the sources in $D$ that contributed to $Q(D)$ [11].

Different notions of provenance, including *lineage*, *why-provenance*, *where-provenance*, and *how-provenance*, are proposed and studied for databases queries [12]. We review lineage, and why-provenance, which we use in this paper.

Intuitively, **lineage** lists the tuples that caused a tuple to appear in the answer of a query [13]. In contrast, **why-provenance** explains which tuples (witnesses) are necessary to make a tuple appear in the answer [11]. To continue with our running example, the lineage of $b_1$ is $\{r_1, r_6, r_2, r_7\}$. The why-provenance of $b_1$ in Table X is $r_1, r_6$ or $r_2, r_7$, either of which is sufficient to explain the presence of $b_1$ in the results.

### B. Data Summarization Rules

The work in [3], [1] introduces summarization rules to summarize "interesting" aspects of a table. In Section III we use them to summarize and explore the provenance of aggregate queries. Here we review some concepts from [1].

**Definition 1.** [Summarization rules] A summarization rule $s$ over a relation $R$ with schema $\mathcal{R} = \{A_1, ..., A_m\}$ is an $n$-ary tuple in which for every $A_i \in \mathcal{R}$, $s[A_i] \in \mathit{Dom}(A_i) \cup \{\star\}$ and $\star$ is a value not in $\mathit{Dom}(A_i)$. □

The value $\star$ is a wildcard that matches every attribute value and allows the rule to summarize multiple tuples.

**Example 5.** In Table IV, $s_1 = (\star, \star, \star, 7, \star, \star, \mathit{Male}, \star)$ is a summarization rule over *MoviesDirectors* that matches every movie with rating 7 that is directed by a male director. □

*For clarity, we include attribute names in the rules, e.g.* $s_1$ *in Example 5 is* $(\mathit{Title} : \star, \mathit{Year} : \star, \mathit{Genre} : \star, \mathit{Rating} : 7, \mathit{Rev} : \star, \mathit{Director} : \star, \mathit{Gender} : \mathit{Male}, \mathit{Country} : \star)$.

**Definition 2.** [Cover and count] Given a relation $R$, a summarization rule $s$ covers a tuple $r \in R$ denoted by $r \in s$ if for every $A_i$, $r[A_i] = s[A_i]$ or $s[A_i] = \star$. $\mathit{Cover}(s)$ is the set of tuples in $R$ that are covered by $s$ and $\mathit{Count}(s) = |\mathit{Cover}(s)|$ is the number of tuples covered by $s$. □

**Example 6.** In Example 5, $s_1$ covers $t_1, t_3, t_5, t_7, t_{11}$, $\mathit{Cover}(s_1) = \{t_1, t_3, t_5, t_7, t_{11}\}$ and $\mathit{Count}(s_1) = 5$. Rule $s_2 = (\mathit{Title} : \star, \mathit{Year} : \star, \mathit{Genre} : \mathit{Drama}, \mathit{Rating} : \star, \mathit{Rev} : \star, \mathit{Director} : \star, \mathit{Gender} : \mathit{Male}, \mathit{Country} : \mathit{US})$ in Table IV covers the drama movies directed by male directors that are born in US, $\mathit{Cover}(s_2) = \{t_1, t_2, t_8\}$ and $\mathit{Count}(s_2) = 3$. □

**Definition 3.** [Marginal cover and marginal count] For a list of rules $S = (s_1, s_2, ...)$ over relation $R$, $\mathit{MCover}(s_i, S)$ is the marginal cover of $s_i$ and is defined as the tuples that are covered by $s_i$ and not any $s_j \in S$ with $j < i$. $\mathit{MCount}(s_i, S) = |\mathit{MCover}(s_i, S)|$ is the marginal count of $s_i$. □

**Example 7.** Considering $S = (s_1, s_2)$, $\mathit{MCover}(s_1, S) = \{t_1, t_3, t_5, t_7, t_{11}\}, \mathit{MCover}(s_2, S) = \{t_2, t_8\}$, $\mathit{MCount}(s_1, S) = 5$, $\mathit{MCount}(s_2, S) = 2$. □

**Definition 4.** [Score function and marginal score] The score of a list of rules $S = (s_1, s_2, ...)$ is defined as follows:

$$\mathit{Score}(S) = \sum_{s_i \in S} \mathit{MCount}(s_i, S) \times \mathit{Weight}(s_i). \quad (1)$$

*Weight* is a monotone function that returns a non-negative real number. For $s_i \in S$, $\mathit{MCount}(s_i, S) \times \mathit{Weight}(s_i)$ is its marginal score. □

The weight function conveys how well a rule summarizes the values in a table. We use the common weight function that is introduced in [3]: the number of non-$\star$ values.

**Example 8.** For $s_1$ and $s_2$, $\mathit{Weight}(s_1) = 2$ and $\mathit{Weight}(s_2) = 3$. The score of $S = (s_1, s_2)$ is $5 \times 2 + 2 \times 3 = 16$ and the score of $S' = (s_2, s_1)$ is $3 \times 3 + 4 \times 2 = 17$. The marginal scores of $s_1$ and $s_2$ are subsequently 10 and 6 in $S$, and 9 and 8 in $S'$. □

**Definition 5.** [Summary] A summary over a relation $R$ is a set of summarization rules over $R$. The score of a summary is the maximum score between all the possible lists containing the rules in the summary. □

**Example 9.** In Example 8, $\{s_1, s_2\}$ is a summary with score $\max(\mathit{Score}((s_1, s_2)), \mathit{Score}((s_2, s_1))) = \max(16, 17)$. □

**Definition 6.** [The summarization problem] Given a relation $R$ and a fixed value $k$, the summarization problem is to find a summary $S$ with $|S| = k$ and maximum $\mathit{Score}(S)$. □

The summarization problem is NP-hard [1]. The authors in [1] present a greedy algorithm called **Best Rule Set (BRS)** that finds a sub-optimal set of rules efficiently. At a high level, BRS has $k$ steps. It starts with an empty rule set $S$ and at each step it adds the best rule that maximizes the score function. In order to find the rule $s$ to add in each step, the algorithm computes the impact of every possible rule on the score function. The algorithm prunes some of the rules without computing the score function. The authors also suggest a data sampling scheme for finding the best rule when only a limited number of tuples can be processed in memory. The approximation guarantee in the algorithm is based on the fact that the score function (Definition 4) is sub-modular [1, Lemma 3].

## III. Formalizing Provenance Summarization Problems

In this section, we define impact summaries and comparative summaries and we formalize the summarization problems for finding these summaries. The key challenge is to define new score functions that specify these provenance summaries while preserving the sub-modularity property for the efficient computation of them. Our new score function for impact summaries computes an accumulative impact of the provenance records on the query answer in Section III-B. We present an extension of the score function in Section III-B1 to solve a problem that occurs when summarizing the provenance of some aggregate queries, such as queries with MIN and MAX aggregate functions and queries with joins. We show how this extension works for queries with join in Section III-B2. We then define a score function for comparative summaries by extending the coverage of summarization rules to a pair-wise coverage of the provenance of two answers in Section III-C.

### A. Basics and Problem Setting

To define the provenance summarization problem, we assume a general aggregate SQL query $Q$ over a database schema $\mathcal{S}$ as in Listing 1.

$Q$: SELECT $G_1, ..., G_m, f(G^*)$ AS $G$ FROM table_references WHERE conditions GROUP BY $G_1, ..., G_m$

Listing 1: The general form of aggregate queries

In $Q$, $G_j, 1 \leq j \leq m$ and $G^*$ are attributes in the from clause and $f$ is an aggregate function, i.e. either a built-in function, such as COUNT and AVG, or a user-defined function. In our problem definition we assume no restriction on the from clause and the where clause in $Q$, e.g. table_references and condition may contain sub-queries over $\mathcal{S}$.

Consider a database $D$ of the schema $\mathcal{S}$. For $a \in Q(D)$, we define $Q^a$ in Listing 2 as a query that returns $a[G]$.

$Q^a$: SELECT $f(G^*)$ FROM table_references WHERE conditions AND $G_1 = a[G_1]$ AND ... $G_m = a[G_m]$

Listing 2: A query that returns the aggregate value in t

**Example 10.** For $Q_1$ in Example 2, $G_1 = Gender$, $G = SumRev$, $G^* = Rev$, and $f$ is the SUM aggregate function. For $a_2 \in Q_1(MoviesDirectors)$, $Q_1^{a_2}$ is the following query:

$Q_1^{a_2}$: SELECT SUM(Rev) FROM MoviesDirectors WHERE Gender=Male

It returns $a_2[SumRev] = 3.374$ *B$* which is the sum of the revenue of the movies directed by male directors. □

For $a \in Q(D)$ and $R \in D$, we use $R^a \subseteq R$ to refer to the lineage of $Q^a(D)$ in $R$. For example $MoviesDirectors^{a_2}$ for $a_2 \in Q_1(MoviesDirectors)$ contains all the tuples in *MoviesDirectors* excluding $t_4, t_9$.

### B. The Impact Summarization Problem

As motivated in Section I, an impact summary summarizes the provenance of a tuple in the result of an aggregate query. It does a better job compared to the basic summaries in Section II-B. To define impact summaries and the impact summarization problem, we specify the new score function, *IScore*. Similar to the score in Definition 4, we define *IScore* (impact score) for a list of summarization rules $S$.

Consider a tuple $a \in Q(D)$ and a relation $R \in D$ with schema $\mathcal{R}$ and the lineage $R^a \neq \emptyset$. Let $S$ be a set of rules over $\mathcal{R}$. The impact score of $S$ is the maximum impact score between every possible list that contains the rules in the set:

$$IScore^{a,R}(S) = \sum_{s_i \in S} Impact^{a,R}(s_i, S) \times Weight^a(s_i), \quad (2)$$

The attribute values in $a$ may also appear in $s_i$. For example in $s_1$ from Section I, $s_1[Gender] = a_2[Gender]$. Therefore the weight function $Weight^a(s_i)$ is the number of non-$\star$ values in $s_i$ that are not in $a$. Note that this weight function is different from *Weight* is Definition 4. For example, $Weight^{a_2}(s_1) = 1, Weight^{a_2}(s_4) = 2$ because each rule has two non-$\star$ values including *Male* that does not count in the weight function since it also appears in $a_2$.

The *IScore* function considers the impact of the provenance tuples in $R^a$ covered by rules in $S$ on tuple $a$ in the query result. This is noted by the superscript $a, R$ in $IScore^{a,R}$ and is applied by replacing $MCount(s_i, S)$ with $Impact^{a,R}(s_i, S)$,

$$Impact^{a,R}(s_i, S) = \sum_{t \in MCover(s_i, S)} |Q^a(D) - Q^a(D \setminus \{t\})|. \quad (3)$$

The impact factor in Equation 3 measures the impact of a rule $s_i$ using *sensitivity analysis*. It is a technique that measures the sensitivity of a query to a tuple or a set of tuples by comparing the answers to the query with and without the tuple(s) in the database [10], [7]. In Equation 3, we apply this analysis w.r.t. the tuples in the marginal cover set $MCover(S, s_i)$ and considering $Q^a$. We omit relation names, e.g. $R$ in Equations 3 and 2, in the superscripts when they are clear from context.

**Example 11.** Consider $Q_1$, $a_2 \in Q_1(MoviesDirectors)$ and summaries $S = \{s_1\}$ and $S' = \{s_4\}$ with $s_1$ and $s_4$ from Section I. Based on the score function in Definition 4, $Score(S) = 10$ and $Score(S') = 6$. The new scores based on Equation 2 are $IScore^{a_2}(S) = Impact^{a_2}(s_1) \times Weight^{a_2}(s_1) = (t_1[Rev] + t_3[Rev] + t_5[Rev] + t_7[Rev] + t_{11}[Rev]) \times 1 = 1,150.8$ and $IScore(S') = Impact^{a_2}(s_3) \times Weight^{a_2}(s_4) = (t_6[Rev] + t_{10}[Rev]) \times 1 = 1434.9$ because $s_1$ covers $t_1, t_3, t_5, t_7, t_{11}$ and $s_4$ matches $t_6, t_{10}$ from the provenance of $a_2$. Therefore, $S'$ is preferred over $S$ w.r.t. *IScore*. □

**Definition 7.** [Impact summary and the impact summarization problem] Given relation $R \in D$, query $Q$ over $D$, a tuple $a$ in the answer $Q(D)$, and a constant $k$, the impact summarization problem is to find a summary $S$ of size $k$ with maximum $IScore^a(S)$. $S$ is an impact summary. □

The impact summarization problem is NP-hard because it extends the summarization problem [1] (Section II), which has been proved to be NP-hard. The BRS baseline algorithm relies on sub-modularity of *Score* in Definition 4. In Section IV, we present the *IPS* algorithm for finding impact summaries. To guarantee the algorithm's approximation, we need to verify whether *IScore* is sub-modular.

**Theorem 1.** *IScore in Equation 2 is sub-modular.*

The proof of Theorem 1 is based on that the marginal impact of every tuple $a \in Q(D)$ in Equation 3, i.e. $|Q^a(D) - Q^a(D \setminus \{t\})|$, is independent of the selected rules in $S$ and it is only considered in the marginal score of one rule in $S$ (see Appendix B for the complete proof). As a result of this theorem, we can efficiently generate approximately optimal impact summaries w.r.t. our score function.

*1) Impact Summaries with Contingency:* We now explain a shortcoming of the impact definition in Equation 3 and we present our solution to extend the impact with contingency.

**Example 12.** Consider the following query that asks the maximum rating of movies from each genre.

$Q_4$ : SELECT Gendre, MAX(Rating) AS MaxR FROM MoviesDirectors GROUP BY Genre

According to Table I, $a = (Drama, 8)$ is an answer to $Q_4$ with provenance $R^a = \{t_2, t_8\}$, which contains all the drama movies. There is no meaningful impact summary for $R^a$ using the score function in Equation 2 and the impact value in Equation 3. This is because *IScore* is 0 for every summary. This happens because there are two tuples $t_2$ and $t_8$ with the maximum rating 8 and removing each one from *MoviesDirectors* does not change $a$. As a result, $Impact^a$ is 0 for every rule and score is 0 for every summary. □

Example 12 shows that the sensitivity analysis used in the definition of *Impact* in Equation 3 is not effective for certain queries, e.g. some queries with MIN or MAX aggregate functions. To solve this problem, we define a new *Impact* that is based on the notion of the contingency set. It is more

general and extends the *Impact* in Equation 3 to address the shortcoming in Example 12.

**Definition 8.** [Contingency Set] Consider a query $Q$ in Listing 1 over a database $D$, a tuple $a \in Q(D)$, the query $Q^a$ as defined in Listing 2, and $R^a$ (the provenance of $a$ in $R$). *The contingency set* for $t \in R^a$ denoted by $C^a(t)$ is a minimal set of tuples in $R^a$ such that $Q^a(D) \neq Q^a(D \setminus C^a(t) \cup \{t\})$ and $Q^a(D) = Q^a(D \setminus C^a(t)) = Q^a(D \setminus \{t\})$. □

Intuitively, the contingency set $C^a(t)$ is a minimal subset of $R^a$ that must be removed from $R$ before $Q^a$ becomes sensitive to $a$. A particular case is when $C^a(t) = \emptyset$ which means removing $a$ immediately changes $Q^a(D)$. The notion of contingency is first introduced in causal inference in AI to specify the degree of responsibility [14], [15]. In data management Meliu et al. use contingency set of a tuple to define its responsibility w.r.t. a query answer [16], [17]. Definition 8 is adapted from [17, Definition 2.1].

**Example 13.** In Example 12, $C^a(t_2) = \{t_8\}$ and $C^a(t_8) = \{t_2\}$. This is because removing either $t_2$ or $t_8$ does not change the answer but removing both does. □

We now define a new *Impact* by extending Equation 3 with consistency set:

$$\mathit{Impact}^a(s_i, S) = \sum_{t \in \mathit{MCover}(s_i, S)} \frac{|Q^a(D) - Q^a(D \setminus C^a(t) \cup \{t\})|}{|C^a(t)| + 1}. \quad (4)$$

In Equation 4, similar to Equation 3, the impact of $s_i$ is the sum of the impacts of tuples $a$ in the marginal cover of $s_i$. For every tuple $a$, its impact is the the total impact of its contingency set, i.e. $|Q^a(D) - Q^a(D \setminus C^a(t) \cup \{t\})|$ multiplied by $\frac{1}{|C^a(t)|+1}$. This ratio is called *the responsibility of a for the answer* $Q^a(D)$ [16].

**Example 14.** Consider summary $\{s\}$ for $a = (\mathit{Drama}, 8) \in Q_4(\mathit{MoviesDirectors})$ in Example 12 where $s = (\mathit{Title} : \star, \mathit{Year} : 2014, \mathit{Genre} : \mathit{Drama}, \mathit{Rating} : \star, \mathit{Rev} : \star, \mathit{Director} : \star, \mathit{Gender} : \mathit{Male}, \mathit{Country} : \star)$. The rule only covers $t_2$, which is the only drama movie from a male director released in 2014, i.e. $\mathit{MCover}(s, \{s\}) = \{t_2\}$. The score of $\{s\}$ is 0 if we use the impact function in Equation 3. According to Equation 4, the new impact value is $\mathit{IScore}^a(\{s\}) = \mathit{Impact}^a(s, \{s\}) \times \mathit{Weight}^a(s) = ((8-7) \times 1/2) \times 1 = 1$. This is because the contingency set of $t_2$ is $\{t_8\}$, the responsibility of $t_2$ is $1/2$ and removing $t_8$ and $t_2$ from *MoviesDirectors* changes the answer to $Q_4^a$ from 8 to 7. $\mathit{Weight}^a(s) = 2$ because $s_5$ has three non-$\star$ values including *Drama* from $a$. □

The new impact definition in Equation 4 is compatible with the impact in Equation 3 in the sense that the former reduces to the latter when the aggregate function is SUM or COUNT. This is because the contingency set of every tuple with a positive impact value according to Equation 3 will be empty, its responsibility is 1, and therefore Equation 4 reduces to Equation 3. The score function with the new impact value is sub-modular which can be proved similar to Theorem 1 (see Appendix B).

*2) Impact Summaries for Queries with Joins:* The class of queries we considered in Listing 1 are general and cover a wide range of aggregate queries, such as queries with joins and nested queries. Here, we showcase and discuss in detail how these summaries work for queries with joins.

Consider queries of the following general form that is defined over a database $D$ including relations $R$ and $T$ with schemas $\mathcal{R} = \{A_1, A_2, ...\}$ and $\mathcal{T} = \{B_1, B_2, ...\}$, respectively:

$Q$: SELECT $G_1, ..., G_m, f(G^*)$ AS $G$ FROM $R$ JOIN $T$ ON $A_i = B_j$ WHERE conditions GROUP BY $G_1, ..., G_m$

Listing 3: The general form of aggregate queries with join

In Listing 3, we assume $\{G_1, ..., G_m\} \subseteq \mathcal{R} \cup \mathcal{T}$, $G^* \in \mathcal{R}$ and $A_i$ is a foreign key and $B_j$ is the key attribute in $T$. Our discussion extends to queries with multiple joins and joins on multiple attributes.

For a tuple $a \in Q(D)$, we define $Q^a$ similar to Listing 2 but over $R$ and $T$. We also define $R^a$ and $T^a$ subsequently as the lineage (provenance) of $Q^a(D)$ in $R$ and $T$.

**Example 15.** Consider $Q_5$ over Movies and Directors in Tables VIII and IX that asks the sum of the revenues of the movies directed by directors of different genders.

$Q_5$: SELECT d.Gender, SUM(m.Rev) FROM Movies m JOIN Directors d ON m.Director = d.Name

For the answer $a = (\mathit{Male}, 654.3M\$)$, $Q_5^a$ is the following query that returns $654.3M\$$:

$Q_5^a$: SELECT SUM(Rev) FROM Movies m JOIN Directors d ON m.Director = d.Name WHERE d.Gender = Male

The provenance of $a$ contains its lineage tuples in both *Movies* and *Directors*: $\mathit{Movies}^a = \{r_1, r_2, r_3, r_5\}$ contains every movie directed by a male director, and $\mathit{Directors}^a = \{r_6, r_7, r_8\}$ are male directors with at least one movie. □

The impact summary of an answer $a$ to an ASPJ query $Q$ contains two sets of rules over $R$ and $T$, e.g. the impact summary of $a$ is two sets of rules over *Movies* and *Directors*. This makes these summaries different from the impact summaries over the join result, e.g. *MoviesDirectors*, since those could be dominated by important tuples in just one relation.

**Example 16.** Consider $S_M = \{s_m\}$ and $S_D = \{s_d\}$ as sets of rules over *Movies* and *Directors*. $s_m = (\mathit{Title} : \star, \mathit{Year} : \star, \mathit{Genre} : \star, \mathit{Rating} : 7, \mathit{Rev} : \star, \mathit{Director} : \star)$ is a rule that summarises movies with rating 7, i.e. $r_1, r_3, r_5$, and $s_d = (\mathit{Name} : \star, \mathit{Gender} : \mathit{Male}, \mathit{Country} : \mathit{US})$ covers male directors from the US, i.e. $r_6$. The impact of $s_m$ is $r_1[\mathit{Rev}] + r_3[\mathit{Rev}] + r_5[\mathit{Rev}] = 569.4$ and the score of $S_M$ is $\mathit{Impact}^a(s_m) \times \mathit{weight}^a(s_m) = 569.4 \times 1$. The impact of $s_d$ only depends on $r_6$, i.e. Steven Spielberg. Since he directed two movies in the *Movies* relation with revenues 275.3 and 219.4, the impact of $s_d$ that covers him is $275.3 + 219.4$ and that is also the score of $S_D$ since the weight of $s_d$ is 1. □

### C. The Comparative Summarization Problem

As motivated in Section I, comparative summaries best summarize the similarities between the sets of provenance $R^a$

and $R^{a'}$ for $a, a' \in Q(D)$. To define the comparative summarization problem, we present the following score function that measures how well rules $s_i \in S$ summarise $R^a$ and $R^{a'}$ (we omitted $R$ is the superscripts similar to Equation 3 and 2):

$$\textit{CScore}^{a,a'}(S) = \sum_{s_i \in S} \textit{MPCount}^{a,a'}(s_i, S) \times \textit{Weight}^{a,a'}(s_i), \quad (5)$$

In *CScore* (the comparative score), $\textit{MPCount}^{a,a'}(s_i, S)$ is the marginal number of tuple-pairs from $R^a$ and $R^{a'}$ that are covered by $s_i$. It is marginal because it does not count the pairs that are already covered by previous rules $s_j, j < i$. The weight function $\textit{Weight}^{a,a'}(s_i)$ returns the number of non-$\star$ values in $s_i$ that do not appear in $a$ and $a'$. Using the standard marginal count function for the provenance of two tuples $a$ and $a'$ where $R^a$ is much bigger than $R^{a'}$ would result in a summary where $R^a$ would dominate. *CScore* gives a summary that covers both $R^a$ and $R^{a'}$ in a balanced way.

**Example 17.** In our example, the comparative score of $\{s_2\}$ for summarizing the similarities between the provenance of $a_3, a_8$ in $Q_3$(*MoviesDirectors*) is $\textit{CScore}^{a_3,a_8}(\{s_2\}) = \textit{MPCount}^{a_3,a_8}(s_2, \{s_2\}) \times \textit{Weight}^{a_3,a_8}(s_2) = 0 \times 1$. The marginal pair count *MPCount* is 0 because $s_2$ does not cover any pair from the provenance of $a_3, a_8$. The weight function returns 1 since there is only one non-$\star$ value, i.e. 2014, that do not appear in $a_3, a_8$. For the set of rules $\{c_1\}$, the score is the following: $\textit{CScore}^{a_3,a_8}(\{c_1\}) = \textit{MPCount}^{a_3,a_8}(c_1, \{c_1\}) \times \textit{Weight}^{a_3,a_8}(c_1) = 1 \times 2$. *MPCount* is 1 since $c_1$ covers a pair of tuples $t_1, t_5$ from the provenance of $a_3$ and $a_8$ respectively. The weight of $c_1$ is 2 because it has 3 non-$\star$ values including *Male* that appears in $a_3$ and $a_8$ and does not count. □

**Definition 9.** [The comparative summarization problem] Given database $D$, relation $R$, query $Q$, tuples $a, a'$ in $Q(D)$, and a constant $k$, the comparative summarization problem is to find a summary $S$ of size $k$ with maximum $\textit{CScore}^{a,a'}(S)$. □

**Theorem 2.** *$\textit{CScore}^{a,a'}$ in Equation 5 is sub-modular.*

Theorem 2 holds because every pair of tuples in $R^a, R^{a'}$ are counted once in $\textit{MPCount}^{a,a'}$. As a result if $S \supseteq S'$ then $\textit{MPCount}^{a,a'}(c_i, S) \leq \textit{MPCount}^{a,a'}(c_i, S')$. The detailed proof of the theorem is in Appendix B. We present a greedy algorithm to find comparative summaries in Section IV. As a result of Theorem 2, we can claim the algorithm gives an approximately optimal set of rules w.r.t. our score function in Equation 5. We study extensions of comparative summaries considering the difference between the provenance tuples and with joins in Appendix C.

## IV. Summarization Algorithms

In this section, we present *the impact provenance summarization algorithm* (*IPS*) and *the comparative provenance summarization algorithm* (*CPS*) for finding impact summaries and comparative summaries. These algorithms extend the BRS algorithm in [1] to summarize the provenance of queries using the new score functions in Section III. In our description of *IPS* and *CPS*, we omit the optimization details of the BRS algorithm to focus on the extension for summarizing provenance, but we apply those optimizations in our experiments.

Algorithm 1 shows the detail of *IPS* that takes as input $D$, $R \in D$, $Q$, an answer $a \in Q(D)$, and value $k$ and computes an impact summary $S$ of $R^a$ with $k$ rules. First, it generates the provenance tuples $R^a \subseteq R$ using an existing provenance framework, such as Perm [18] (Line 1). Then it computes the impact of each tuple in $R^a$ on $Q(D)$ (Line 2). The algorithm uses $R^a$ and the impacts to find the $k$ best marginal rules by calling *BestMarginalRule* $k$ times and returns the results as $S$. The best marginal rule $s$ maximizes the score of $S \cup \{s\}$.

**Algorithm 1:** The *IPS* Algorithm

**Input:** Database $D$, relation $R$, query $Q$, $a \in Q(D)$, $k$.
**Output:** A set of rules $S$.

1 $S := \emptyset, R^a := \textit{Provenance}(D, R, Q, a)$
2 **foreach** $t$ **in** $R^a$ **do** $I(t) := |Q^a(D) - Q^a(D \setminus \{t\})|$;
3 **for** $i$ **from** 1 **to** $k$ **do**
4 $\quad s := \textit{BestMarginalRule}(R^a, a, S, I)$
5 $\quad S := S \cup \{s\}$
6 **return** $S$

Algorithm 2 shows the main steps of *BestMarginalRule*. To find the best marginal rule, $s$, the procedure maintains two sets of new and old candidate rules, $S_n$ and $S_o$ respectively. At the $j$-th iteration of the main loop, it generates every possible candidate rule of weight $j$, stores them in $S_n$, and computes their marginal scores in $M$. For $j > 1$, the procedure generates the new rules in $S_n$ using the rules with weight $j-1$ in $S_o$ (Line 4). For example, if $s_o = (A : \star, B : b)$ is in $S_o$, $s_n = (A : a, B : b)$ will be added to $S_n$. Here $s_n$ is a **super-rule** of $s_o$ and $s_o$ is a **sub-rule** of $s_n$.

Before computing the scores (Lines 11-14), the procedure prunes some candidate rules that cannot beat the best rule $s$ from the previous iterations without computing their scores (Lines 5-10). For each $s_n$, the procedure computes an upper bound $U_n$ using the scores of its sub-rules $s_o$ that are computed in the previous iteration. The value $(M(s_o)/\textit{Weight}^a(s_o)) \times |\mathcal{R}|$ in Line 8 is an upper bound for the marginal score of $s_n$, since the impact of $s_n$ is always less than $M(s_o)/\textit{Weight}^a(s_o)$, which is the impact of its sub-rule $s_o$, and the weight of $s_n$ can never exceed $|\mathcal{R}|$. The procedure removes the candidate $s_n$ from $S_n$ if its score's upper bound is still less than the current best marginal rule $s$. In line 10, if no candidate remains after pruning $S_n = \emptyset$, the procedure stops and returns the best marginal rule $s$.

The procedure computes the marginal score of candidate rule $s_n$ in $S_n$ in Line 14 by adding the impact of every tuple $t$ in $R^a$ to $M(s)$ if $t$ is not covered by any rule in $S$. At the end of each iteration in Line 15, the procedure checks for the best marginal rule $s$ in $S_n$ and then updates $S_o$ for next iteration. The procedure returns $s$ after checking every candidate rule.

Algorithm 3 details *CPS*, which takes $R$, $Q$, $k$, and $a_1, a_2 \in Q(D)$ and returns a comparative summary $S$ with $k$ rules. The

**Algorithm 2:** *BestMarginalRule*$(R^a, a, S, I)$

**Output:** A rule $s$ with maximum marginal score.

1 **for** $j$ **from** 1 **to** $|\mathcal{R}|$ **do**
2  **if** $j = 1$ **then** $S_n :=$ **all rules with weight** 1;
3  **else**
4  $S_n :=$ **all weight-**$j$ **super-rules of rules in** $S_o$
5  **foreach** $s_n \in S_n$ **do**
6  $U_n := \infty$
7  **foreach** $s_o \in S_o$ **i.e. a sub-rule of** $s_n$ **do**
8  $U_n := \min(U_n, \frac{M(s_o)}{\mathit{Weight}^a(s_o)} \times |\mathcal{R}|)$
9  **if** $U_n < M(s)$ **then** $S_n := S_n \setminus \{s_n\}$ **break**;
10  **if** $S_n = \emptyset$ **then break**;
11  **foreach** $s_n \in S_n$ **do** $M(s_n) := 0$ ;
12  **foreach** $t \in R^a$ **not covered by rules in** $S$ **do**
13  **foreach** $s_n \in S_n$ **that covers** $t$ **do**
14  $M(s_n) := M(s_n) + I(t) \times \mathit{Weight}^a(s_n)$
15  $s := \mathit{argmax}_{s_n \in S_n}(M(s_n)), S_o := S_n$
16 **return** $s$

algorithm first computes the provenance of $a_1, a_2$, then uses it to find the top $k$ marginal rules by calling *BestComparativeMRule* (Algorithm 4) $k$ times. This procedure takes the provenance sets $R^{a_1}$, $R^{a_2}$, and the set of rules $S$, and finds the rule that best summarizes $R^{a_1}$ and $R^{a_2}$. It computes the marginal score of all rules except those that are already in $S$ and returns the one with the highest score. However, to compute the marginal scores, it only adds to the score of a rule if it covers pairs of tuples $t_1, t_2$ from $R^{a_1}, R^{a_2}$ that are not covered by any rule in $S$ (Line 7).

**Algorithm 3:** The *CPS* Algorithm

**Input:** Database $D$, relation $R$, query $Q$, $a_1, a_2 \in Q(D)$, value $k$.

**Output:** A set of rules $S$.

1 $S := \emptyset$
2 $R^{a_1} :=$*Provenance*$(D, R, Q, a_1)$; $R^{a_2} :=$*Provenance*$(D, R, Q, a_2)$
3 **for** $i$ **from** 1 **to** $k$ **do**
4  $c :=$ *BestComparativeMRule*$(R^{a_1}, R^{a_2}, a_1, a_2, S)$
5  $S := S \cup \{c\}$
6 **return** $S$

*IPS* is different from the BRS algorithm (as detailed in [3]) since *IPS* generates provenance and considers the impact values while it computes the marginal scores. The BRS algorithm prunes candidate rules similar to *IPS* with the difference that *IPS* uses impact values that can help prune rules faster when the numerical attribute that is aggregated has skewed data. The *CPS* algorithm is also different from the BRS algorithm because *CPS* generates provenance and computes the score function considering pairs of tuples.

**Algorithm 4:** *BestComparativeMRule*$(R^{a_1}, R^{a_2}, a_1, a_2, S)$

**Output:** A rule $s$ with maximum marginal score.

1 $M_s := 0$
2 $S_n :=$ **all possible rules that are not in** $S$
3 **foreach** $c_n \in S_n$ **do**
4  $M := 0$
5  **foreach** $t_1 \in R^{a_1}, t_2 \in R^{a_2}$ **covered by** $c_n$ **do**
6  **if no rule in** $S$ **covers** $t_1$ **and** $t_2$ **then**
7  $M := M + \mathit{Weight}^{a_1,a_2}(c_n)$
8  **if** $M_s < M$ **then** $c := c_n, M_s := M$;
9 **return** $s$

### A. Computing Impacts and Contingency Sets

So far, in our problem definition in Section III and the summarization algorithms in Section IV, we considered the general class of aggregate SQL queries in Listing 1. However, in our experiments we focus on the sub-class of Aggregate-Select-Project-Join (ASPJ) queries with built-in aggregate functions for two reasons. First, they cover a wide range of queries that are used in practice, and second, we can compute the provenance summaries for these queries efficiently as we discuss in Section IV-B. Note that computing the impact of tuples on the general class of aggregate queries in Listing 1 is intractable since finding the contingency set for the provenance tuples is NP-hard in general [16]. In addition, computing the impact of the provenance tuples $t$ on a general query $Q$ with user-defined aggregate functions requires running $Q^a(D \setminus \{t\})$ for every tuple $t$ which can be costly. We can avoid this costly query execution for ASPJ queries as we will explain next.

In Line 2, *IPS* computes the impacts of provenance tuples $t$ in $R^a$ and stores them in $I$. For queries in our experiments, we can compute impacts of tuples without running queries. For example for an Aggregate-Select-Project (ASP) query $Q$ with SUM, the impact of $t \in R^a$ is $|t[G^*]|$, and it is $\frac{|t[G^*]|}{|R^a|}$ for queries with AVG. In Example 11, the impact of the movie $t_1$ on the sum of revenues in $Q_2^{a_2}$ is the movie's revenue, i.e. $t_1[\mathit{Rev}] = 275.3M\$$. For an ASP query with MIN or MAX, computing the impact in Equation 4 requires finding the contingency set $C^a(t)$. Although computing the contingency set is proved to be NP-hard for general SQL queries [16], we can efficiently compute it for queries of the general form in Listing 1. If $t[G^*] = a[G]$, which means $t[G^*]$ is the min or max value in the answer, the contingency set is the set of all tuples $t'$ such that $t'[G^*] = t[G^*]$. In Example 12, $a[\mathit{MaxR}] = t_2[\mathit{Rating}]$ and $C^a(t_2) = \{t_8\}$. The impact of $t$ is $\frac{|t[G^*]-m|}{|C^a(t)|+1}$ in which $m$ is the second min or max value in $R^a$. In Example 12, the impact of $t_2$ is $\frac{8-7}{2}$. If $t[G^*] \neq a[G]$, the impact of $t$ is 0, e.g. the impact of $t_1$ is 0 in Example 12.

For an ASPJ query of the general form in Listing 3, we compute the impact of $t \in R^a$ as we explained above for queries without joins. If $t \in T^a$, the impact of $t$ depends on values all the tuples $t' \in R^a$ that join with $t$, e.g. $t'[A_i] = t[B_j]$. We compute the impact of $t$ using $t'[G^*]$ for every $t'$. For

example, for ASPJ queries with the SUM aggregate function, the impact of $t$ is $\sum(|t'[G^*]|)$.

### B. Analysis of Summarization Algorithms

The cost of running *IPS* is divided into (a) generating $R^a$, (b) computing the impacts in $I$, and (c) finding the best marginal rules. We only analyse the cost of (b) and (c) since (a) depends on the provenance framework. For the queries in our experiments, the cost of (b) is $O(n)$ with $n = |R|$ because the impact of each tuple can be computed in $O(1)$ as noted in Section IV-A. The cost of (c) is at most $O(k \times n^2 \times 2^m)$ with $m = |\mathcal{R}|$ since there are at most $n^2 \times 2^m$ possible rules, and the cost of computing the marginal score of each rule is $n$. The rule pruning technique helps reduce the cost to $O(k \times n^2 \times m)$ (the detail of its analysis is in [3]). Therefore, the total cost of *IPS* is at most $O(k \times n^2 \times m)$. The cost of *CPS* only consists of the cost of generating $R^{a_1}, R^{a_2}$, which we omit, and the cost of finding the best marginal rules in Line 4 of Algorithm 4. That cost is $n^2$ since *BestComparativeMRule* iterates over pairs of tuples in Lines 5-7 of Algorithm 4. That means the total cost of *CPS* is $O(k \times n^3 \times m)$.

*IPS* and *CPS* only incur the approximation caused by the greedy rule selection. The score functions in both algorithms are monotone since the marginal scores are non-negative. They are also sub-modular as we proved in Theorems 1 and 2. As a result, the algorithms have the approximation ratio $\alpha = 1 - \frac{1}{e}$, i.e. the score of a result summary is greater than the optimal score multiplied by $\alpha$ [19].

## V. User Interface and Visualization

The user interface and visualizations provide users with all the facilities they need to explore the provenance summarization rules. Rules are presented to the user in a way that helps them uncover insights about the data.

The user starts by writing a query and seeing the results of the query. The user then clicks on one or more tuples and asks to see a summary of the provenance. We offer two different interfaces depending on the type of query or question: (a) Impact summaries if the user clicks on a single tuple; this includes impact summaries for the results of queries with joins. (b) Comparative summaries if a user clicks on multiple tuples.

For impact summaries, rules are presented as rows in a table (or multiple tables when the query involves multiple tables) with a quality measure for each rule. This quality measure is one of the following. **Score**: the score of a rule according to Equations 2 and Equations 5, which reflect the contribution of each rule to the summary's total score. **Coverage**: the fraction of provenance tuples that are covered by a rule. **Impact**: the impact of a rule normalized by the total possible impact. The impact and coverage values are not marginal and do not depend on the other rules in the summary.

For impact summaries for queries that involve multiple tables, the interface shows separate summaries for each table in the join. The user can click on a rule in one summary to see how it relates to the rules in the other summary. Figure 1 shows an example with three lines denoting different values for the relationship between the rules. The thicker the line the stronger the connection between the tuples.

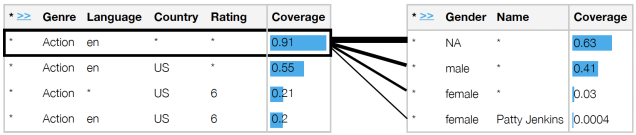



Fig. 1: User interface for summaries for queries with join.

For comparative summaries, we offer a similar interface. The difference is the horizontal bars show how balanced each rule is in covering the provenance of the two tuples.

In all our interfaces the rules offer the following interactions. The user can click on a rule to expand it and see a set of super-rules contained within a rule (see Section IV for the definition of super-rule). The user can also contract the list to hide any sets of super-rules. In a list of rules, there can be trivial attributes that have all their values as $\star$ in all rules. For example, the attribute Title in Figure 1. These attributes can all be collapsed into a single attribute named $\star$ to reduce clutter as seen in Figures 1

## VI. Experimental Evaluation

In this section, we have three objectives: (a) to evaluate the performance of our algorithms and show they can generate summaries in real-time (Section VI-B), (b) to compare provenance summaries with basic summaries and show provenance summaries have higher quality w.r.t. our new metrics (Section VI-C), (c) to show the relevance of impact and other related metrics such as diversity for provenance summarization using a user survey (Section VI-D).

### A. Experimental Setup

The performance of our summarization algorithms and their result summaries vary depending on the selected tuples in the query answer. Since there are no restrictions over these user selected tuples, for each experiment we produced summaries for a large number of randomly selected answers and reported the results as the distribution of those answers.

The results of our summarization algorithms also depend on the query. We ran our experiments for different queries. To analyse the algorithms on queries with different aggregate functions, we used the same queries and replaced the aggregate function. We reported the results for all functions only if there is significant difference between the results. We used *IAGG* and *CAGG* to refer to the *IPS* and *CPS* with AGG as the aggregate function, e.g. *ISUM* is *IPS* with SUM.

We implemented our algorithms in Python and ran the experiments on a machine with 3.3GHz Intel CPU and 16 GB RAM that uses PostgreSQL 9.4. The provenance generation component is Perm [18].

**Datasets** We used three datasets. Table XI gives the data characteristics, showing a range of data sizes w.r.t. the number

| | GLEI | IMDB | TPC-H |
|---|---|---|---|
| $\|D\|$ | 250k | 181k+323k | 60k - 3m |
| $N$ | 1 | 2 | 8 |
| $m_D$ | 12 | 15 | 61 |
| $q$ | 5 | 6 | 6 |

TABLE XI: Datasets characteristics.

of tuples ($|D|$), the number of tables ($N$), the total number of attributes in the tables ($m_D$), and the number of queries ($q$):

**Global Legal Entity Identifier (GLEIs).** This is a real world financial dataset collected from various sources for financial institutions [20] to create a single, universal identifier for entities that are involved in any financial transaction. We used this dataset for performance analysis (Exp-3 in Section VI-B). In those experiments, we ran the following query over a single table, GLEI, with varying number of attributes $G_1, G_2, ...$ and aggregate functions $f$:

SELECT $G_1, G_2, ..., f$(Dist) FROM GLEI GROUP BY $G_1, G_2, ...$

Dist is a numerical attribute describing the distance between each entity and the center of the city where it is located.

**IMDB.** We used a subset of the IMDB dataset with two tables, *Movies* and *Directors*. The movies table includes revenue for which we calculated the aggregate results. We used this dataset for performance analysis (Exp-2 and 4) and quality experiments (Exp-6 and 7). In those experiments, we ran the following query with different number of attributes and different aggregate functions.

SELECT $G_1, G_2, ..., f$(Rev) FROM Movies m, Directors d
WHERE m.Director = d.Name GROUP BY $G_1, G_2, ...$

**TPC-H.** As far as we know, there is no standard benchmark for provenance systems or summarization. We used TPC-H for performance experiments and to show how our methods handle queries with multiple types of aggregation and multiple complex joins.[2] We varied the database size as follows: 60k, 300k, 600k, and 3m. In the TPC-H experiments, we ran five queries, $Q_1, Q_3, Q_5, Q_6, Q_{10}$, as generated by the TPC-H tool. We chose those queries because they contain features covered by our summarization algorithm.

**Parameters** The default value of $k$, the size of provenance summaries, is 8, which we decided based on survey results in Section VI-D. The value $m_w$ specifies the maximum weight of rules and its default value is 4, higher values of $m_w$ make the rules too verbose for a summary of our datasets. The number of group-by attributes in the queries, $m_q$, has the default value of 1, we assume for users unfamiliar with a dataset they would start their analysis with a single group-by attribute query.

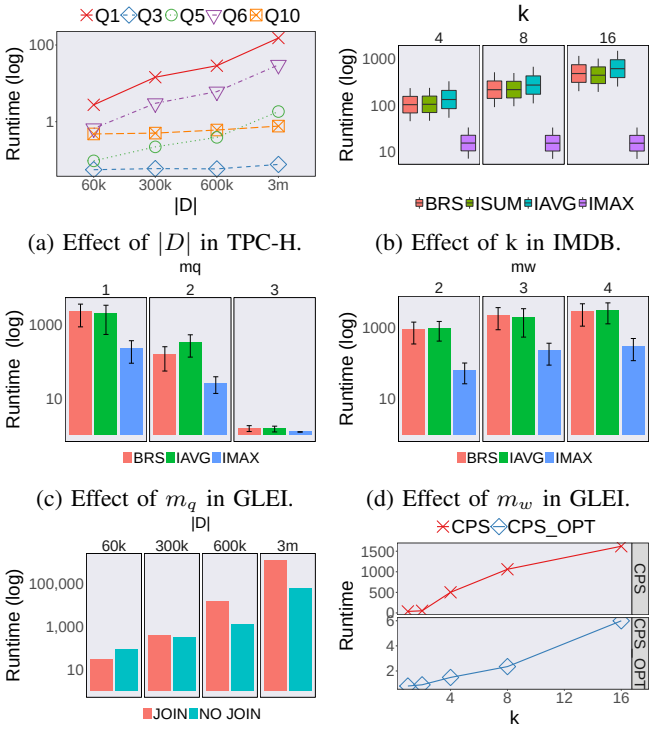
### B. Runtime Performance

To evaluate performance of *IPS*, we compared its runtime with BRS while changing $|D|$ and $k$ in Exp-1 and Exp-2, and $m_q$ and $m_w$ in Exp-3, respectively. We studied the runtime of *IPS* for queries with joins in Exp-4 and *CPS* in Exp-5.

[2]http://www.tcp.org/hspec.html

**Exp-1: Effects of $|D|$.** Figure 2a shows the effect of $|D|$ on the runtime of *IPS* for the queries in the TPC-H dataset. As we expected, the runtime increases for larger $|D|$. However, *IPS* scales differently for each query. For example, while the increase in the runtime of *IPS* for $Q_3, Q_{10}$ is hardly noticeable, there is a clear jump in the runtime of $Q_1$. This is because the provenance of $Q_1$ involves a portion of $D$ that increases as we increase $|D|$, while $Q_3$ and $Q_{10}$ access almost a fixed part of $D$. This experiment shows that *IPS* scales w.r.t. $|D|$ but its actual runtime highly depends on the query.

**Exp-2: Effects of $k$.** Figure 2b shows the effect of $k$ on the runtime of *IPS*. We see an increase in runtime as we increase $k$ for all algorithms. The increase in runtime is linear because the main loop of *IPS* (cf. Algorithm 2) runs $k$ times. We also observe that IMAX runs faster than the other algorithms since it only summarizes a small subset of the provenance set. ISUM, BRS, and IAVG perform similarly. IAVG performs slightly worse because of the extra computations.




(a) Effect of $|D|$ in TPC-H. (b) Effect of k in IMDB.

(c) Effect of $m_q$ in GLEI. (d) Effect of $m_w$ in GLEI.

(e) Effect of join in TPC-H. (f) Effect of opt. on CPS.

Fig. 2: Effect of different parameters on Runtime, in seconds, of *IPS*. CPS experiments used the IMDB dataset.

**Exp-3: Effects of $m_q$ and $m_w$.** According to Figure 2c, the runtime of *IPS* greatly decreases as the number of group-by attributes $m_q$ increases. In the experiments on the GLEI dataset, the average size of a provenance set is $9,000$ tuples, $4,500$ tuples, and 922 tuples for $m_q = 1, 2, 3$ respectively. This shows that the more variables used to aggregate the data, the faster the summaries can be produced. As expected, $m_w$ has the inverse effect with runtime increasing as $m_w$ increases. The main loop for Algorithm 2 runs $m_w$ times.

**Exp-4: Effects of join in queries.** We considered TPC-H queries that contain joins and generated the same queries without joins by materializing the join result. Figure 2e shows the runtime of *IPS* for all queries. *IPS* performs similarly for all queries when data size is small. The join queries slightly outperform those without join because the summaries for queries with join have smaller number of attributes as a the shared join attributes appear only once in the join result. However, we see a more pronounced effect of join for larger data sizes, because finding contingency sets, which is costly.

**Exp-5: CPS and optimized CPS.** As we discussed in Section IV-B, *CPS* is more costly compared to *IPS* since it compares pairs of tuples. Therefore, we implemented an optimization of *CPS* that prunes the rules with low coverage on either provenance sets using a coverage threshold. We evaluated the effect of this optimization in Figure 2f by comparing the runtime of CPS and optimized CPS using the IMDB dataset. The optimized version runs 250% faster because many tuples have a low coverage on at least one provenance set. While this is shown to be very effective for IMDB, it might not be as effective in other datasets.

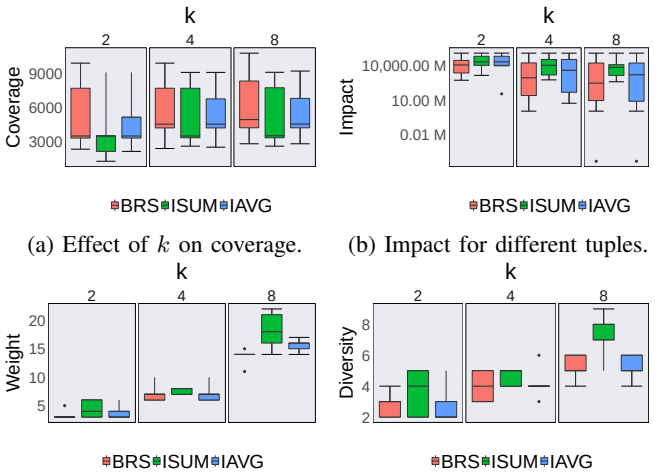
### C. Quality of Provenance Summaries

Measuring the quality of a summary of this type is a challenging task. We chose four objective metrics from and used them to evaluate summaries produced using the BRS algorithm and *IPS*. We also conducted a user survey to study users' expectations for such summaries to show the relevance of our metrics.

We define the following summary quality metrics for a summary: **Impact** is the distribution of impact values for rules in the summary. Sum and average of impact for a summary do not give the full picture. A good impact summary needs to have all rules of high impact. **Coverage** is the total number of provenance tuples that are covered by the rules in the summary. **Weight** is the total weight (the number of non-$\star$ values) of the rules in a summary. Higher weight represents summaries that are more descriptive. **Diversity** is the number of unique values for attributes present in the summary. Diversity is used in [2] to optimize a summary using a distance function. We used it as a quality measure where a more diverse summary would cover more attribute values. We argue that a single measure by itself is not sufficient to evaluate the quality of a summary. A high quality summary would have high impact rules, high coverage, high diversity, and its rules would have higher weights.

**Exp-6: Coverage and impact.** Figures 3a and 3b respectively show the coverage and the impact of the rules generated by different algorithms. Figure 3a shows that BRS generates rules with higher coverage, which was expected. Surprisingly, Figure 3b reveals that BRS also generates rules with high impact. This can be explained by looking at the distribution of the impact values, which shows a wide range of impact values in the basic summaries. This is because the basic summaries in BRS achieve high impact with a few high-coverage rules that have low weight and are not very informative. The rest of the rules in a basic summary do not have high impact. However, the impact summaries, e.g. ISUM, gain high impact with informative rules with high weights. This confirms that *IPS* finds rules with high impact that are also informative.

**Exp-7: Diversity and weight.** Figures 3c also supports the fact that *IPS* consistently picks higher weight rules than the BRS algorithm. ISUM generates rules with higher weights than IAVG due to the higher values of impact multiplied by weight. In terms of diversity, *IPS* outperforms the BRS algorithm (Figure 3d). With higher diversity values, we can say that the higher weight rules of *IPS* are also more informative.




(a) Effect of $k$ on coverage. (b) Impact for different tuples.

(c) Effect of $k$ on weight. (d) Effect of $k$ on diversity.

Fig. 3: The effect of $k$ on summaries' quality metrics. Results are reported from experiments on the IMDB data.

### D. User Survey

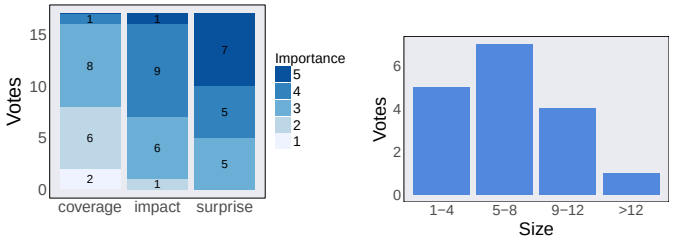
We conducted an online survey to ask computer science students about their expectations of a data summarization system. We designed our questionnaire to be generic, presenting a use-case scenario that general users would be familiar with. We tried to minimise bias by giving a very generic example of the IMDB dataset.

We presented participants with a scenario where a user posed a query to the system, looked at an aggregate answer, and received a short summary of the data that contributed to the answer. We asked users to evaluate this summary then asked them about their thoughts for what type of information should be included in such a summary. We asked users to rank: rules covering movies with high revenue, rules covering a lot of movies, and rules with surprising information using Likert scales from not important (1) to essential (5). These rules represent impact, coverage, and a surprise factor respectively. We also asked users what size of summary would be appropriate and asked them to justify their choices. We use those qualitative answers to filter the survey responses with low quality responses and as evidence of the quality metrics users prefer.

17 participants completed the survey. Results show participants favor high impact rules over rules with high coverage.

They also favor surprising rules the most (Figure 4a). In their evaluation of the summaries, most users picked the high impact, low coverage rules, and high weight rule as the most interesting. Results also show that participants favor summaries of small sizes, i.e. 5-8. Only one user picked the summaries of size $> 12$ as seen in Figure 4b.

Users wrote out qualitative answers to justify their choices. Notably, users commented that high impact rules were most interesting and that they preferred shorter summaries. We present some quotes from the survey participants: "Row 2 was most interesting showing that only $25\%$ of movies collected $78\%$ of revenue which is significant." "I care about ranking rules based on the impact factor." "More than 5 rows is not a summary".




(a) Ranked summary metrics. (b) Preferred summary size.

Fig. 4: Survey results show how users ranked the various metrics (5 essential - 1 not important) and summary sizes.

## VII. Related work

**Provenance Database Systems.** Data provenance is studied extensively in the database literature. While there are different notions of provenance in databases [11], e.g. lineage, why-provenance, how-provenance and where-provenance, in this work we started from lineage. Summarizing other types of provenance is one of our future research directions.

There are several frameworks for generating and managing provenance data in data management [13], [18]. Perm [18] is a common provenance framework that manages provenance information using easily optimizable SQL. The semantics of Perm are similar to why- and where-provenance. We use Perm to generate the provenance we summarize. More approaches are explored in data provenance surveys such as [12].

**Query Explanation.** In Section I, we noted that our work is built on the query explanation work to generate the impact of summaries [7], [8], [9]. Our work is different since it focuses on summarization rather than finding the best explanation. Here, we review the recent work on query explanation.

Scorpion [7] is a framework that helps users understand outliers in aggregate query answers. It provides explanations in the form of predicates over the attributes of tuples that contribute to the outliers. The framework uses the notion of influence to compute the effect of tuples the on query answer; this is similar to the sensitivity analysis in our solution. In [9], the authors present an approach for explaining answers to SQL queries based on the notion of intervention, i.e. removing of tuples from the database and measuring their effect on query answers. They focus on answering user questions in the form of $(Q, high)$ or $(Q, low)$, where a user seeks an explanation for an unexpected high or low result. The explanation is in the form of a set of predicates or conditions that specify the tuples that have major contribution to the answer to the query in the user's question. [8] extends [9] to explain more general questions and user questions using intervention. This approach is different in the kind of questions it answers and the explanations it provides.

**Summarization rules.** There are several frameworks that apply summarization rules similar to those we use: [2], [1], [3], [4]. However, their purposes are different from ours. The work in [1], [3] summarizes a relation and does not consider the query or its provenance. The interactive summarization framework in [2] applies these rules to summarize the top $k$ answers of an aggregate query but not their provenance. The data summarization technique in [4] solves a different problem: given a relation with an outcome attribute, it constructs a summary of the factors affecting the outcome attribute in a prediction task. Our approach is the only framework that extensively uses these informative and interpretable rules for provenance of queries.

## VIII. Conclusion

In this paper we looked at provenance and summarization research. We implemented new summarization techniques, impact summaries and comparative summaries. Our summaries are ideal for users with little knowledge of provenance semantics or the data itself. We validated our techniques with thorough experiments and a user survey to show they present summarized information that is relevant to users. We intend to extend our solution to support more general use cases and conduct a usability study to further validate our findings.

## References


[1] M. Joglekar, H. Garcia-Molina, and A. Parameswaran, "Interactive data exploration with smart drill-down," in *ICDE*, 2016, pp. 906–917.

[2] Y. Wen, X. Zhu, S. Roy, and J. Yang, "Interactive summarization and exploration of top aggregate query answers," *PVLDB*, vol. 11, no. 13, pp. 2196–2208, Sep. 2018.

[3] M. Joglekar, H. Garcia-Molina, and A. Parameswaran, "Smart drill-down: A new data exploration operator," *PVLDB*, vol. 8, no. 12, pp. 1928–1931, 2015.

[4] K. El Gebaly, P. Agrawal, L. Golab, F. Korn, and D. Srivastava, "Interpretable and informative explanations of outcomes," *PVLDB*, vol. 8, no. 1, pp. 61–72, 2014.

[5] E. Ainy, P. Bourhis, S. B. Davidson, D. Deutch, and T. Milo, "Approximated summarization of data provenance," in *CIKM*, 2015, pp. 483–492.

[6] Y. Amsterdamer, D. Deutch, and V. Tannen, "Provenance for aggregate queries," in *PODS*, 2011, p. 153–164.

[7] E. Wu and S. Madden, "Scorpion: Explaining away outliers in aggregate queries," *PVLDB*, vol. 6, no. 8, pp. 553–564, 2013.

[8] S. Roy, L. Orr, and D. Suciu, "Explaining query answers with explanation-ready databases," *PVLDB*, vol. 9, pp. 348–359, 2015.

[9] S. Roy and D. Suciu, "A formal approach to finding explanations for database queries," in *SIGMOD*, 2014, pp. 1579–1590.

[10] B. Kanagal, J. Li, and A. Deshpande, "Sensitivity analysis and explanations for robust query evaluation in probabilistic databases," in *SIGMOD*, 2011, pp. 841–852.

[11] P. Buneman, S. Khanna, and T. Wang-Chiew, "Why and where: A characterization of data provenance," in *ICDT*, 2001, pp. 316–330.

[12] J. Cheney, L. Chiticariu, and W.-C. Tan, *Provenance in databases: Why, how, and where*. Now Publishers Inc, 2009.
[13] Y. Cui and J. Widom, "Lineage tracing in a data warehousing system," in *ICDE*, 2000, pp. 683–684.
[14] H. Chockler and J. Y. Halpern, "Responsibility and blame: A structural-model approach," in *IJCAI*, 2003, pp. 147–153.
[15] T. Eiter and T. Lukasiewicz, "Complexity results for structure-based causality," *Artif. Intell.*, vol. 142, no. 1, pp. 53–89, Nov. 2002.
[16] A. Meliou, W. Gatterbauer, K. F. Moore, and D. Suciu, "The complexity of causality and responsibility for query answers and non-answers," *PVLDB*, vol. 4, no. 1, pp. 34–45, Oct. 2010.
[17] A. Meliou, W. Gatterbauer, S. Nath, and D. Suciu, "Tracing data errors with view-conditioned causality," in *SIGMOD*, 2011, pp. 505–516.
[18] B. Glavic, R. J. Miller, and G. Alonso, *Using SQL for Efficient Generation and Querying of Provenance Information*. Springer Berlin Heidelberg, 2013, pp. 291–320.
[19] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions–i," *Math. Program.*, vol. 14, no. 1, pp. 265–294, 1978.
[20] K. K. Chan and A. Milne, "The global legal entity identifier system: How can it deliver?" *J. of Risk and Financial Management*, vol. 12, no. 1, 2019.

## APPENDIX

### A. Alternative Score Functions

We present two alternative score functions, *agglomerative impact score* for impact summaries and *entropy-based score* for comparative summaries. While these score functions specify meaningful summaries that are different from the summaries in Section III, they are non-submodular functions and do not allow efficiently computing sub-optimal summaries.

*1) Accumulative Impact Summaries:* We present $AImpact^t$ (accumulative impact) as an alternative for $Impact^t$ in Equation 3 that computes the impact of a tuples covered by a rule as a group:

$$AImpact^t(s_i, S) = |Q^t(R) - Q^t(R \setminus MCover(s_i, S))|. \quad (6)$$

Using this impact function, we define a new score function in Equation 7:

$$AIScore^t(S) = \sum_{s_i \in S} AImpact^t(s_i, S) \times Weight^t(s_i), \quad (7)$$

The following example applies this new score function and compares it with the score function used in the impact summaries in Section III.

**Example 18.** Consider the following query on *MoviesDirectors* that returns two answers, $w_1 = (Female, 312.3\ M\$)$ and $w_2 = (Male, 337.3\ M\$)$:

*Q : SELECT Gender, AVG(Rev) AS AvgRev*
*FROM MoviesDirectors GROUP BY Gender*

Let assume the user requests an impact summary for $w_2$. Consider summaries $S = \{s_1\}$ and $S' = \{s_3\}$ with $s_1 = (Title : \star, Year : \star, Genre : \star, Rating : 7, Rev : \star, Director : \star, Gender : Male, Country : \star)$ and $s_3 = (Title : \star, Year : \star, Genre : Action, Rating : \star, Rev : \star, Director : \star, Gender : Male, Country : \star)$, we compute the score functions w.r.t. the score functions in Equations 2 and 7 as follows:

$$IScore^{w_2}(S) = Impact^{w_2}(s_1) \times Weight^{w_2}(s_1) = (|337.3 - 344.2| + |337.3 - 366.5| + |337.3 - 350.4| + |337.3 - 329.1| + |337.3 - 355.9|) \times 1 = 76.03.$$

$$IScore^{w_2}(S') = Impact^{w_2}(s_3) \times Weight^{w_2}(s_3) = (|337.3 - 332.7| + |337.3 - 329.1| + |337.3 - 257.5|) \times 1 = 92.7.$$

$$AIScore^{w_2}(S) = AImpact^{w_2}(s_1) \times Weight^{w_2}(s_1) = = |337.3 - 444.6| \times 1 = 107.2.$$

$$IScore^{w_2}(S') = Impact^{w_2}(s_3) \times Weight^{w_2}(s_3) = |337.3 - 218.2| \times 1 = 119.18$$

The rule $s_1$ covers $t_1, t_3, t_5, t_7, t_{11}$ and $s_3$ covers $t_6, t_7, t_{10}$. In both types of summaries, $S' = \{s_3\}$ is preferred to $S = \{s_1\}$ because it has a higher score. □

**Proposition 1.** *The score function AIScore in Equation 7 is sub-modular if the aggregate function in Q is monotone (e.g. COUNT, MIN, MAX) and it is not sub-modular if the aggregate function is non-monotone (e.g. SUM or AVG).* ■

*2) Entropy-based Comparative Summaries:* We presented comparative summaries in Section **??** based on a new score function that measures the balance between the tuples covered by a summarization rule from two provenance sets. Now, we suggest an alternative score function that gives a different measure of the balance between the covered tuples using *binary entropy function*:

$$ECScore^{t,t'}(S) = \sum_{s_i \in S} MCount^{t,t'}(s_i, S) \times Weight(s_i), \quad (8)$$

This comparative score function is different from the sore function in Definition 4 in its marginal count function. Unlike *MCount* that counts the number of remaining records covered by $s_i$, $MCount^{t,t'}$ counts records from the provenance of both $t$ and $t'$. We define $MCount^{t,t'}$ as follows:

$$(MCount^t(s_i, S) + MCount^{t'}(s_i, S)) \times Entropy^{t,t'}(s_i, S). \quad (9)$$

Here, $MCount^t(s_i, S) + MCount^t(s_i, S)$ is a marginal count of the tuples in the provenance of $t$ or $t'$ that are covered by $s_i$ ($MCount^t$ and $MCount^{t'}$ are defined in Definition 3). The binary entropy function $0 \leq Entropy^{t,t'}(s_i, S) \leq 1$ measures the balance between the coverage from the provenance of $t$ and $t'$. It is 0 if either $MCount^t(s_i, S) = 0$ or $MCount^t(s_i, S) = 0$ meaning all the covered provenance records are from one side, and it is 1 if exactly the same number of records are covered from the provenance of $t$ and $t'$. More precisely, the binary entropy function is defined as follows:

$$Entropy^{t,t'}(s_i, S) = H(p(s_i, S)^t) = H(p(s_i, S)^{t'}). \quad (10)$$

in which $H(p) = -p \times \log p - (1-p) \times \log(1-p)$ is the binary entropy function, $p(s_i, S)^t$ is defined as follows,

$$p(s_i, S)^t = \frac{\mathit{MCount}^t(s_i, S)}{\mathit{MCount}^t(s_i, S) + \mathit{MCount}^t(s_i, S)},$$

while $p(s_i, S)^{t'}$ is defined analogously. We define $\mathit{MCount}^{t,t'} = 0$ if $\mathit{MCount}^t(s_i, S) = 0$ and $\mathit{MCount}^t(s_i, S) = 0$ when $s_i$ has no marginal coverage from from $R_t \cup R_{t'}$.

**Example 19.** In our running example, considering $t = g_1, t' = g_4$ and $s_1, ..., s_4$, the score of $\{s_1, ..., s_4\}$ is 0 because each $s_i$ only covers records from the provenance of $g_4$. The score of a set of rules $\{s_5\}$ is 6 because $\mathit{MCount}^{t,t'}(s_5, \{s_5\}) = 2 \times H(0.5) = 2$ and $\mathit{Weight}(s_5) = 3$. □

**Proposition 2.** *The score function ECScore in Equation 8 is not sub-modular.* ■

The proof of this proposition is based on a similar counter example in the proof of Proposition 1.

*B. Proofs*

**Proof of Theorem 1.** Given a set of rules $S$ that summarize the provenance of $t \in Q(R)$, and a tuple $r \in R$ in the provenance of $Q^t(R)$, let $L(r, S)$ be the first rule in $S$ that covers $r$ (the rule that covers $r$ and has the maximum score). Let $\Delta^t(r) = |Q_t(R) - Q_t(R \backslash \{r\})|$ be the impact of $r$ on $Q_t(R)$. The score can be rewritten as the following:

$$\mathit{IScore}^t(S) = \sum_{r \in R} \mathit{Weight}^t(L(r, S)) \times \Delta^t(r). \tag{11}$$

Assuming a set of rules $S'$ such that $S \subsetneq S'$ and a rule $s \notin S'$, we prove $\mathit{IScore}^t$ is submodular by showing the following always holds: $\mathit{IScore}^t(S \cup \{s\}) - \mathit{IScore}^t(S) \geq \mathit{IScore}^t(S' \cup \{s\}) - \mathit{IScore}^t(S')$. The marginal score for $s$ w.r.t. $S$ and $S'$ can be written using Equation 11. For example for $S$, the marginal score $\mathit{IScore}^t(S \cup \{s\}) - \mathit{IScore}^t(S)$ can be written as the following:

$$\sum_{r \in R} [\mathit{Weight}^t(L(r, S \cup \{s\})) - \mathit{Weight}^t(L(r, S))] \times \Delta^t(r).$$

Since $\Delta(r)$ is a positive value that only depends on $r, Q^t$ and $R$, we can consider the following cases for every $r \in R$:

**Case 1.** $\mathit{Weight}^t(L(r, S' \cup \{s\})) - \mathit{Weight}^t(L(r, S')) > 0$ which means $r$ is in the marginal cover set of $s$ which has the highest weight between the rules that cover $r$ and $L(r, S \cup \{s\}) = s$. Since $S \subsetneq S'$, we can claim that $\mathit{Weight}^t(L(r, S \cup \{s\})) \geq \mathit{Weight}^t(L(r, S))$ and $L(r, S) = s$ which means

$$\mathit{Weight}^t(L(r, S \cup \{s\})) - \mathit{Weight}^t(L(r, S)) \geq \mathit{Weight}^t(L(r, S' \cup \{s\})) - \mathit{Weight}^t(L(r, S')).$$

**Case 2.** $\mathit{Weight}^t(L(r, S' \cup \{s\})) - \mathit{Weight}^t(L(r, S')) = 0$ and

$$\mathit{Weight}^t(L(r, S \cup \{s\})) - \mathit{Weight}^t(L(r, S)) \geq \mathit{Weight}^t(L(r, S' \cup \{s\})) - \mathit{Weight}^t(L(r, S'))$$

because $\mathit{Weight}^t(L(r, S \cup \{s\})) - \mathit{Weight}^t(L(r, S)) \geq 0$. This means for every $r \in R$

$$\mathit{Weight}^t(L(r, S \cup \{s\})) - \mathit{Weight}^t(L(r, S)) \geq \mathit{Weight}^t(L(r, S' \cup \{s\})) - \mathit{Weight}^t(L(r, S'))$$

which proves the theorem. This proof is based on the proof of [1, Lemma 3]. □

**Proof of Theorem 2.** Similar to the proof of Theorem 1, consider $S$, $t_1, t_2 \in Q(R)$, $r_1 \in R^{t_1}, r_2 \in R^{t_2}$ in the provenance of $t_1$ and $t_2$. Let $L(\langle r_1, r_2 \rangle, S)$ be the first rule in $S$ that covers $\langle r_1, r_2 \rangle$ (the rule that covers both $r_1$ and $r_2$ and has the maximum score). The *CScore* function can be rewritten as the following:

$$\mathit{CScore}^{t_1,t_2}(S) = \sum_{\langle r_1, r_2 \rangle \in R^{t_1} \times R^{t_2}} \mathit{Weight}^{t_1,t_2}(L(\langle r_1, r_2 \rangle, S)). \tag{12}$$

Assuming a set of rules $S'$ such that $S \subsetneq S'$ and a rule $s \notin S'$, we prove $\mathit{CScore}^{t_1,t_2}$ is submodular by showing the following always holds: $\mathit{CScore}^{t_1,t_2}(S \cup \{s\}) - \mathit{CScore}^{t_1,t_2}(S) \geq \mathit{CScore}^{t_1,t_2}(S' \cup \{s\}) - \mathit{CScore}^{t_1,t_2}(S')$. The marginal score for $s$ w.r.t. $S$ and $S'$ can be written using Equation 12. For example for $S$, the marginal score $\mathit{CScore}^{t_1,t_2}(S \cup \{s\}) - \mathit{CScore}^{t_1,t_2}(S)$ can be written as the following:

$$\sum_{\langle r_1, r_2 \rangle \in R^{t_1} \times R^{t_2}} [\mathit{Weight}^{t_1,t_2}(L(\langle r_1, r_2 \rangle, S \cup \{s\})) - \mathit{Weight}^{t_1,t_2}(L(\langle r_1, r_2 \rangle, S))].$$

Similar the proof of Theorem 1, we can consider two cases for every $\langle r_1, r_2 \rangle \in R^{t_1} \times R^{t_2}$:

**Case 1.** $\mathit{Weight}^{t_1,t_2}(L(\langle r_1, r_2 \rangle, S \cup \{s\})) - \mathit{Weight}^{t_1,t_2}(L(\langle r_1, r_2 \rangle, S)) > 0$ which means $s$ covers both $r_1, r_2$ and has the highest weight between the rules that cover them and $L(\langle r_1, r_2 \rangle, S \cup \{s\}) = s$. Since $S \subsetneq S'$, we can claim that $\mathit{Weight}^{t_1,t_2}(L(\langle r_1, r_2 \rangle, S \cup \{s\})) \geq \mathit{Weight}^{t_1,t_2}(L(\langle r_1, r_2 \rangle, S))$ and $L(\langle r_1, r_2 \rangle, S) = s$ which means

$$\mathit{Weight}^{t_1,t_2}(L(\langle r_1, r_2 \rangle, S \cup \{s\})) - \mathit{Weight}^{t_1,t_2}(L(\langle r_1, r_2 \rangle, S)) \geq \mathit{Weight}^{t_1,t_2}(L(\langle r_1, r_2 \rangle, S' \cup \{s\})) - \mathit{Weight}^{t_1,t_2}(L(\langle r_1, r_2 \rangle, S')).$$

**Case 2.** $\mathit{Weight}^{t_1,t_2}(L(\langle r_1, r_2 \rangle, S \cup \{s\})) - \mathit{Weight}^{t_1,t_2}(L(\langle r_1, r_2 \rangle, S)) = 0$ and

$$\textit{Weight}^{t_1,t_2}(L(\langle r_1, r_2\rangle, S \cup \{s\})) - \textit{Weight}^{t_1,t_2}(L(\langle r_1, r_2\rangle, S)) \geq$$
$$\textit{Weight}^{t_1,t_2}(L(\langle r_1, r_2\rangle, S' \cup \{s\})) - \textit{Weight}^{t_1,t_2}(L(\langle r_1, r_2\rangle, S')).$$

because $\textit{Weight}^{t_1,t_2}(L(\langle r_1, r_2\rangle, S \cup \{s\})) - \textit{Weight}^{t_1,t_2}(L(\langle r_1, r_2\rangle, S)) \geq 0$. This means for every $\langle r_1, r_2\rangle \in R^{t_1} \times R^{t_2}$,

$$\textit{Weight}^{t_1,t_2}(L(\langle r_1, r_2\rangle, S \cup \{s\})) - \textit{Weight}^{t_1,t_2}(L(\langle r_1, r_2\rangle, S)) \geq$$
$$\textit{Weight}^{t_1,t_2}(L(\langle r_1, r_2\rangle, S' \cup \{s\})) - \textit{Weight}^{t_1,t_2}(L(\langle r_1, r_2\rangle, S'))$$

which proves the theorem. □

**Proof of Proposition 1.** The proof of sub-modularity for COUNT is given in [1, Lemma 3] which can be extended to any monotone aggregate function. For sets of rules $S_1 \subseteq S_2$ that summarize relation $R$ and a new rule $s \notin S_1 \cup S_2$, the proof is based on the fact that the marginal score w.r.t. every tuple $t \in R$ after adding $s$ is greater for $S_1$ compared to $S_2$. For the second part of the proposition, we apply the following counter-example that shows the score function in (2) is not sub-modular when the aggregate function is SUM. Similar counter examples can be generated for any non-submodular function such as AVG.

Let $R = \{(a, b, -1), (c, b, 1)\}$ be a relation with attributes $A_1, A_2, A_3$. Consider $Q$ as the following query: SELECT SUM $(A_3)$ FROM $R$. Let $S_1 = \emptyset$ and $S_2 = \{(a, b, \star)\}$ be sets of summarization rules and $s = (\star, b, \star)$ be a summarization rule. We compute the following score values using the score function in (2) for the only tuple $t$ in $Q(R)$: $\textit{IScore}^t(S_1) = 0$, $\textit{IScore}^t(S_2) = 2$, $\textit{IScore}^t(S_1 \cup \{s\}) = 0$, $\textit{IScore}^t(S_2 \cup \{s\}) = 3$. Therefore, $\textit{IScore}^t(S_1 \cup \{s\}) - \textit{IScore}^t(S_1) = 0$ and $\textit{IScore}^t(S_2 \cup \{s\}) - \textit{IScore}^t(S_2) = 1$ which shows $S_1 \subseteq S_2$ does not imply $\textit{IScore}^t(S_2 \cup \{s\}) - \textit{IScore}^t(S_2) \leq \textit{IScore}^t(S_1 \cup \{s\}) - \textit{IScore}^t(S_1)$ and proves the claim. □

### C. Extensions

In this section, we briefly discuss two extensions: one for summarizing differences and one for handling joins in comparative summaries.

In Section III-C, we focused on comparative summaries for capturing the similarities between two provenance sets. The differences can also be summarised using a new score function similar to the function in Equation 5. The change is that the marginal pair count (*MPCount*) in Equation 5 will be replaced with a function that counts the number of tuples $s_i$ covers from $R^t$ and penalizes $s_i$ if it covers tuples from $R^{t'}$. The definition of such a score function is in A where we also prove it is sub-modular. The following example illustrates the problem of summarizing the differences in the provenance of two tuples:

**Example 20.** In Table III, consider the provenance of $a_3, a_5 \in Q_2(\textit{MoviesDirectors})$. A rule $s_5 = (\textit{Title} : \star, \textit{Year} : \star, \textit{Genre} : \textit{Comedy}, \textit{Rating} : 7, \textit{Rev} : \star, \textit{Director} : \star, \textit{Gender} : \textit{Male}, \textit{Country} : \textit{US})$ provides a comparative summary for the differences between the provenance of $c_1$ and $c_6$ since the rule summarizes $t_5, t_{11}$ in the provenance of $c_6$ but it does not cover any tuple from the provenance of $c_1$. □

In Appendix A , we present other score functions for comparative summaries using the notion of *binary entropy function* that measure the balance between the coverage of the provenance tuples. However, we show that those score functions do not preserve the sub-modularity property.

The comparative summarization problem can be extended for queries with join operators. For example, if the user selects two tuples $e_1$ and $e_2$, we can summarize similarities and differences between the provenance tuples in both *Movies* and *Directors*.

The main difference with the comparative summaries in Section III-C is that the provenance sets for two selected tuples might overlap and this has to be considered in the score function for such comparative summaries.