Provenance in Relational Databases: Usability and Applications

by

Omar AlOmeir

A thesis submitted in partial fulfilment of the requirements for the degree of

Doctor of Philosophy

in

The Faculty of Graduate and Postdoctoral Studies

(Computer Science)

The University of British Columbia

(Vancouver)

April 2023

© Omar AlOmeir, 2023
The following individuals certify that they have read, and recommend to the Faculty of Graduate and Postdoctoral Studies for acceptance, the thesis entitled:

Provenance in relational databases: Usability and applications

submitted by Omar AlOmeir in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science

Examining Committee:

Rachel Pottinger, Professor, Computer Science, UBC
Supervisor

Raymond Ng, Professor, Computer Science, UBC
Supervisory Committee Member

Terje Haukaas, Professor, Civil Engineering, UBC
University Examiner

Alan Wagner, Associate Professor, Computer Science, UBC
University Examiner

Additional Supervisory Committee Members:

Joanna McGrenere, Professor, Computer Science, UBC
Supervisory Committee Member
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 or any meaningful applications. This thesis is focused on facilitating the use of database provenance through visual interfaces, summarization techniques, and curation techniques for real world applications.

In the first part, we present visualization techniques for provenance information in relational databases. Our visualizations address every part of provenance information to facilitate user exploration. Through a user experiment, we show that our approach improves the accuracy and efficiency of performing exploration tasks.

The next part addresses the challenge of volume of provenance information. Specifically, in the case of aggregation queries. The volume increases with the size of the database and creates a “needle in a haystack problem”. We present novel summarization techniques that build on existing summarization literature. Our techniques work to support exploration for users who are not familiar with the data or its provenance.

The final part shows our use of our summarization techniques to address the problem of refining aggregate queries. Aggregate queries pose a chal-
lenge in that they present ambiguous results to inexperienced users. Query refinement can help users realize their query errors and help them fix them. Through user experiment, we present evidence of the usefulness, and usability of our methods.

Overall, the goal of this thesis is to facilitate the use of provenance information in relational databases. Through the use of novel techniques and user-centric evaluation, we present novel solutions and user interaction methods to enable new applications in this domain.
Lay summary

Users of data across all domains require the origin of data in order to gain trust and understanding of the underlying data. Furthermore, information about the origin can improve the process of gaining knowledge from data. Data provenance is the scientific name given to the process of finding the origin of a piece of data. With data provenance comes a large host of complex models and operations. Our goal in this thesis is to facilitate the use of this provenance information for relational database users. We do that through visualizations, summarization techniques, and through the application of our techniques to help users refine their database questions and answers. We conduct experiments with real users to show the validity of our methods.
Preface

This dissertation is the result of multiple collaborations. All work was completed in collaboration with my supervisor Dr. Rachel Pottinger, from the University of British Columbia, Canada. The primary investigator role implies: 1) identifying the research question, 2) designing and performing the research, 3) analysis of results, as well as 4) writing and preparing manuscripts and thesis chapters. Any work done by other team members is addressed for each chapter. Most of the work has been published in peer reviewed venues or is currently under submission. The break down of published content is as follows:

1. Chapter 3 is based on a publication at ICDE 2020. I was the primary investigator for this work. Eugenie helped with implementing the summarization method. Mostafa Milani helped with the write up and formalization. The study in this chapter was approved as minimal risk by the Behavioural Research Ethics Board at the University of British Columbia. UBC BREB number: H16-00797. The paper can found in [ALMP20b]:

2. Chapter 4 is based on a publication at ICDE 2021. An extended version of this work has been accepted in the IEEE Transactions on Knowledge and Data Engineering journal. I was the primary investigator for this work. Eugenie Lai worked on implementing the summarization methods and experiments. Mostafa Milani helped with formulating the problem, writing the manuscript, and came up with the ideas for summaries with joins. The survey conducted in this chapter was approved as minimal risk by the Behavioural Research Ethics Board at the University of British Columbia. UBC BREB number: H20-01634. The paper can be found in [ALMP21]:


3. Chapter 5 is being prepared for submission but is currently unpublished. It is joint work with Jianhao Cao and Rachel Pottinger. I was the lead investigator of this work. Jianhao Cao was in charge of implementing and evaluating the baseline methods. The study in this chapter was approved as minimal risk by the Behavioural Research Ethics Board at the University of British Columbia. UBC BREB number:H21-03344.
Table of Contents

Abstract ........................................................................ iii
Lay summary ................................................................. v
Preface ........................................................................ vi
Table of Contents .......................................................... viii
List of tables ................................................................... xii
List of Figures .................................................................. xiv
Acknowledgements ......................................................... xvii

1 Introduction ................................................................... 1
  1.1 Motivations ............................................................. 1
  1.2 Goals ...................................................................... 3
  1.3 Overview ............................................................... 4
    1.3.1 Visualizing provenance ........................................ 4
    1.3.2 Summarizing provenance ..................................... 4
    1.3.3 Query refinement using provenance and summary rules 5
  1.4 Contributions .......................................................... 6
  1.5 Outline .................................................................... 7

2 Background ................................................................... 8
# List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Summary of symbols and notations</td>
<td>9</td>
</tr>
<tr>
<td>2.2</td>
<td>Movies table</td>
<td>10</td>
</tr>
<tr>
<td>2.3</td>
<td>Directors table</td>
<td>10</td>
</tr>
<tr>
<td>2.4</td>
<td>( Q_1 )'s answer</td>
<td>10</td>
</tr>
<tr>
<td>3.1</td>
<td>Movies table</td>
<td>25</td>
</tr>
<tr>
<td>4.1</td>
<td>( Q_1 )'s result</td>
<td>39</td>
</tr>
<tr>
<td>4.2</td>
<td>( Q_2 )'s result</td>
<td>39</td>
</tr>
<tr>
<td>4.3</td>
<td>IMDB table</td>
<td>40</td>
</tr>
<tr>
<td>4.4</td>
<td>A basic summary</td>
<td>40</td>
</tr>
<tr>
<td>4.5</td>
<td>An impact summary</td>
<td>40</td>
</tr>
<tr>
<td>4.6</td>
<td>A comparative summary</td>
<td>40</td>
</tr>
<tr>
<td>4.7</td>
<td>Movies table</td>
<td>51</td>
</tr>
<tr>
<td>4.8</td>
<td>Directors table</td>
<td>51</td>
</tr>
<tr>
<td>4.9</td>
<td>Summary A</td>
<td>75</td>
</tr>
<tr>
<td>4.10</td>
<td>An impact summary from the IPS algorithm</td>
<td>78</td>
</tr>
<tr>
<td>4.11</td>
<td>A basic summary from the BRS algorithm</td>
<td>78</td>
</tr>
<tr>
<td>5.1</td>
<td>MoviesGenre sample table</td>
<td>88</td>
</tr>
<tr>
<td>5.2</td>
<td>Summary of symbols and notations for query refinement</td>
<td>97</td>
</tr>
<tr>
<td>5.3</td>
<td>Table shows the average time (in seconds) it took users to finish tasks using the two methods (ProvSum and PULearn)</td>
<td>120</td>
</tr>
<tr>
<td>B.1</td>
<td>Summary A</td>
<td>168</td>
</tr>
</tbody>
</table>
List of Figures

3.1 *Pastwatch* use case ..................................................... 23
3.2 *Pastwatch* interface. (A) Main interface with 1) Schema list, 2) Query input, and 3) Query results. (B) The provenance graph shows the history of data to the most recent results, including database operations (Union and Select in this case). (C) The overview visualization is: a) A linked view of a world map and bar chart. Showing geographic locations and a number of companies respectively. b) A treemap visualization is also available for datasets that do not contain geographic information. (D) The summarization rules as a horizontal treemap ........................................... 24
3.3 A screenshot of a table showing provenance information. The prov prefix tells the user they are looking at provenance information. The name of the table follows the prefix, and then the name of the attribute ................................................................. 31
3.4 Results for time (in seconds) broken down by task for each method. Error bars correspond to standard error. Case refers to task ................................................................. 33
3.5 Results for perceived difficulty (1 is very difficult - 5 very easy) broken down by task for each method. Error bars correspond to standard error. Case refers to task ................................................................. 33
4.1 User interface for impact summaries ..................................... 66
4.2 User interface for summaries for queries with join (NA refers to missing data) ................................................................. 66
4.3 User interface for comparative summaries. . . . . . . . . . . . . 67
4.4 Effect of different parameters on Runtime, in seconds, of IPS.
   CPS experiments used the IMDB dataset. . . . . . . . . . . . . 81
4.5 The quality of summaries w.r.t. coverage, weight, impact, and
   diversity for the IMDB data, and the impact of $k$ and $m_w$ (the
   maximum weight) on these quality metrics. . . . . . . . . . . . 82
4.6 The quality of summaries for five queries in the TPC-H dataset
   ($k = 4$ and $m_w = 3$). . . . . . . . . . . . . . . . . . . . . . . 83
4.7 a shows sampling performance. b & c show the effect of sam-
   pling on the quality of summaries. ($k = 4$, $m_w = 3$) . . . . . . 84
4.8 Survey results show how users ranked the various metrics (5
   essential – 1 not important) and summary sizes. . . . . . . . . 85
5.1 Overview of solution from the query step all the way to the
   recommended groups of tuples . . . . . . . . . . . . . . . . . . . 92
5.2 Query refinement framework architecture . . . . . . . . . . . . 93
5.3 t-SNE visualization of TFIDF representation of groups of rules
   in example 20 in 2D space . . . . . . . . . . . . . . . . . . . . . . 110
5.4 User interface . . . . . . . . . . . . . . . . . . . . . . . . . . 116
5.5 Users’ perceived usefulness for both types of refinement method
   (1: very useful - 5: Very useless). . . . . . . . . . . . . . . . 122
5.6 The time it took users to complete tasks for both groups of
   users (Group A started with the PULearn method, Group B
   started with the ProvSum method) . . . . . . . . . . . . . . . . 123
5.7 The time it took users to complete tasks using both methods. 124
5.8 Effect of $|D|$ in TPC-H. . . . . . . . . . . . . . . . . . . . . . . . 126
5.9 Method and time in IMDB. . . . . . . . . . . . . . . . . . . . . . 127
5.10 Breakdown of prediction/training for PULearn in IMDB dataset. 127
5.11 Effect of indexing in IMDB. . . . . . . . . . . . . . . . . . . . . 128
5.12 Preprocessing steps breakdown in IMDB. . . . . . . . . . . . . 128
5.13 Preprocessing steps breakdown in TPC-H. . . . . . . . . . . . . 129
| C.1 | Demographic questionnaire for the query refinement study | 173 |
| C.2 | Tasks interface for the query refinement study | 174 |
| C.3 | Exit questionnaire for the query refinement study | 174 |
| C.4 | Individual movie suggestions (PULearn) | 175 |
| C.5 | Rules suggestions (ProvSum) | 175 |
Acknowledgements

First of all, all praises go to Allah the most merciful for his guidance.

I would like to start by expressing my utmost gratitude to my supervisor, Dr. Rachel Pottinger for not only being an amazing supervisor but also a constant source of inspiration, and a role model in kindness, patience, and dedication to research. Her encouragement and insightful feedback have been extremely valuable to me. We have been through a lot in this PhD journey and she was nothing but supportive and inspirational.

I would also like to thank Dr. Mostafa Milani for all his support, inspiration, kindness, and guidance. I learned a lot from him.

I would also like to thank my Ph.D. committee members, Dr. Joanna McGrenere for all the meetings, advice, feedback, and support. Dr. Raymond Ng for his invaluable advice, kindness, and support. Thank you both for your time and effort.

I thank my collaborators, in no particular order, Eugenie Y. Lai, Jianhao Cao, and Zainab Zolaktaf. Thank you so much for all the excellent work and contributions. Thank you to all members of the db-talks group, and the DMM lab. I will miss both.

I thank my friends in the department. Jessica Wong, for being there from the beginning to the end. I also thank Laura Cang, Enrique Rosales, and Arthur Marques. Thank you for making this a fun experience.

I am incredibly grateful to my parents, who always believed in me. For all the sacrifices they made for me so that I can be who I am today. Thank
you to my father for believing in me and supporting me throughout. Thank you to my mother, my first teacher, my best teacher, I learned so much from you that I cannot possibly repay you. Thank you both.

I express my sincere and wholehearted thanks to my wife, my love, Abrar for believing in me even when I did not believe in myself. Without her by my side, I would not have been able to complete this work. I am thankful for her help, her insight, and her constant love and support. Thank you.

To my daughters Serene and Yasmeen, thank you for being there with me throughout all of this. Playing with you has always helped keep this journey fun, and enjoyable. I couldn’t have done it without you as my guiding light.

I am thankful to my siblings, Othman and Nora, for their love and support throughout the years I spent away from home.

Finally, I would like to thank the Saudi government and Prince Sultan University for their funding and financial support.
Chapter 1

Introduction

1.1 Motivations

The provenance of data refers to the ability to track the origin and evolution of a piece of data throughout its lifetime. Provenance has been a focus of research in several different areas. In the scientific workflow context, provenance is used to show the processes that data went through. This research on provenance work in workflow management has included work on visualizing and 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 very little work on presenting database provenance information in a way that does not overwhelm users. Using existing provenance database systems can be overwhelming without prior knowledge of data provenance models or the data itself. Furthermore, provenance information can increase data size exponentially. This can have serious implications on both storage and usability. We argue that data provenance needs to be visualized and in certain cases summarized to support and facilitate broad exploration. In our work, we propose and develop techniques as se-
rious steps toward making provenance data widely usable without the need for deep technical knowledge.

A formal definition of provenance in database research is as follows:

**Definition 1.1.** Given a database D, a query Q, and the results 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).

Provenance information is important for a number of applications. Our focus is on using provenance to support analysis and exploration tasks. However, to reach its potential, provenance information needs to be presented to the user in a way they can easily understand and explore.

We identify three main challenges:

1. Provenance information needs to be visualized in a way that facilitates exploration. Insightful visualizations of carefully curated provenance information enable users to use provenance information to explore the data. Visualizations should include query results, the processes that led to the creation of the query results, and a visual summary of provenance information.

2. Provenance of aggregation query results presents a “needle in a haystack problem”. The volume of provenance information for an aggregation result can be large and prohibitive for exploration. An aggregation query that counts the number of tuples in a relation returns a single number. The provenance of this single number can be the whole relation or a group of relations. This creates a barrier for users to look through provenance information and find meaningful information.

3. Provenance information is too complex to support applications. In certain cases, provenance information is not sufficient to solve a problem. Refinement of aggregate queries is such a problem. When a user issues an erroneous aggregate query, provenance plays a role in looking at the
origin of the answer but not finding the correct answer. We utilize the
techniques developed to tackle the first two challenges to solve this one.

Addressing these challenges should allow the user to freely explore data
and its related provenance without knowledge of the structure of provenance
information or the data itself. The last challenge creates a meaningful use
case for the techniques developed to tackle the first two.

1.2 Goals

The aim of this research is to design and implement a provenance explo-
ration system for relational databases and utilize and extend the techniques
developed for this system to handle aggregate query refinement. We define
a provenance exploration system as a system that presents provenance in-
formation without overwhelming size and complexity. The specific sub-goals
involved in achieving this are as follows:

- Sub-goal 1: Develop visualizations of all aspects of provenance in database
  systems. This includes Query results, the process that led to creating
  those query results, and a visual summary of provenance information.

- Sub-goal 2: Develop a new way to summarize provenance data for ag-
gregation query results. This method takes into account the unique
aspects of provenance. Our method integrates the aggregation func-
tion used and the impact of provenance tuples on the result into the
calculation of the score function used to evaluate the quality of the
summary. We extend this algorithm to support the following function-
alities: 1) The user can compare the provenance of different partitions
in the aggregation. Such comparisons can uncover interesting insights
into the data. 2) Define different impact functions for multiple aggre-
gation functions. 3) Compare the different semantics of the scoring
function on real-world datasets to see how they perform.
• Sub-goal 3: Develop a query refinement approach using provenance and summaries. This final task is unrelated to the provenance exploration system described previously. It is a related task as we use provenance information and the summarization techniques we have developed. The goal is to allow users to get feedback as to why their aggregation queries are not providing the answer that they expect. We present the user with possible changes to the query results if they incorporate new suggested tuples from the base data.

1.3 Overview

We define three main parts for this thesis that we briefly discuss below:

1.3.1 Visualizing provenance

Provenance information can be large and overwhelming to users. We present a set of criteria that any provenance exploration tool must have and introduce Pastwatch, a provenance exploration system that adheres to those criteria. We also address the issues associated with the provenance of aggregation queries, including creating a summarization method that makes the provenance of aggregation queries manageable for users. Finally, we conduct a quantitative user study to show statistically significant results that Pastwatch makes provenance information more efficient and easier to use than standard approaches.

1.3.2 Summarizing provenance

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, making 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, and techniques for the fast generation
of these summarization schemes. We present impact summaries that consider
the impact of specific groups of tuples on an aggregate value. We also intro-
duce comparative summaries that allow users to compare the provenance of
two aggregate values. We conduct thorough experiments and a user survey
to show the feasibility and relevance of our approaches.

1.3.3 Query refinement using provenance and summary
rules

Our primary motivation for this work is to help users perform aggregation
queries. Users use aggregation queries to perform many tasks: Exploring a
dataset, deriving summary statistics such as count, sum, and avg, or gen-
erating reports. The process is iterative and error-prone. We address the
version of this problem where the user knows the domain well enough to
know that the answer is wrong. In such a case, the user may want to modify
the aggregate query to receive a specific answer. Our technique supports the
user in such a scenario by presenting them with suggestions on how to alter
their query to receive their preferred answer. Our technique is based on the
provenance summaries we describe above. We conduct a user study to show
that our methods outperform comparable machine learning methods. We
also show that our approach gives meaningful suggestions for a broad class
of queries.
1.4 Contributions

We identify 3 main contributions of this thesis:

1. We present, to the best of our knowledge, the first system with visualization of provenance in a relational database system as part of our Pastwatch system. We identify the criteria for designing this system and implementing it. We also conduct a user study to evaluate the utility and usability of our system.

2. We introduce two novel techniques for summarizing the provenance of relational data. We based our techniques on the results of a user survey we conducted. We evaluate those summarization techniques based on criteria for the summary quality we identify in the literature and through our survey results.

3. We introduce a new approach to query refinement based on the use of provenance data and summary patterns. We demonstrate the utility of our approach through a user study. We also perform performance experiments to show the minimal cost incurred by our method.

Throughout our work, we have used user studies as a means of demonstrating the utility of our solutions. In the provenance visualization solution, we compared the visualization we developed with provenance information presented to the user in tables. We showed that the provenance visualizations allowed users to perform the tasks faster and more accurately.

In the provenance summarization work, our initial plan was to conduct a controlled lab experiment to determine some of the solution’s parameters (summary size, user preferred utility function, and presentation method). We opted for a remote user study instead for circumstances at the time (March 2020) did not allow for an in-person user study. It was also easier to acquire ethics approval for a remote survey than a remote study. The survey presented users with a scenario of a data table and asked them questions
about multiple proposed short summaries. We designed the survey to find evidence of what users value in such a summary. We used those metrics to evaluate our summaries in performance experiments.

In our final project, we conducted an in-person and remote user study to compare how users perform the outlined tasks using our approach compared to comparable machine learning techniques.

User studies have not historically been a focus in database research, which has a tendency to focus exclusively on performance rather than usability. While user studies have become more common over the last decade, they are certainly still quite rare. This is a promising development, although the use of user studies in database research still lacks maturity. The rigor and detail of such studies are not usually as well reported or as carefully scrutinized by reviewers as in fields such as human computer interaction. It is difficult to give a full picture of the field, but at the time of publication, our papers were among a few others with such evaluations. While not the main focus of this thesis, we hope the user experiments described here can help maintain the standards of the more rigorous work we found in the literature, and can be up to the standards set in other research fields.

1.5 Outline

This thesis is organized as follows. Chapter 2 presents background information that is relevant to all other chapters. Chapter 3 presents the problem of visualizing provenance information to facilitate user exploration. Chapter 4 presents the problem of summarizing provenance information. Chapter 5 addresses how to use provenance information and summary rules to refine aggregate queries. Finally, Chapter 6 concludes the thesis and gives directions for future work.
Chapter 2

Background

In this chapter, we present some background information that is important for understanding the other chapters. We start with important notation and basic definitions to establish consistent terminology throughout the thesis. In addition, we cover the basics of provenance in database management systems, including the different types and semantics, and the different implementations, along with some examples. We also discuss summary rules, their definition, and their different representations. Both of those topics are used in the following chapters so we decided to collect the information here before the reader gets into the main topics.

We begin with some notation: A relation $R$ with a set of attributes (relation schema) $R = \{A_1, ..., A_m\}$ 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 $S = \{R_1, ..., R_N\}$ is a finite set of relations $R_1, ..., R_N$. 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 $A_i \in R$, $Dom(A_i)$ is the domain of values in $A_i$. Table 2.1 summarizes our symbols and notation.
Table 2.1: Summary of symbols and notations

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R$</td>
<td>relation</td>
</tr>
<tr>
<td>$A, B, ..., G$</td>
<td>relational attributes</td>
</tr>
<tr>
<td>$t, r$</td>
<td>tuples</td>
</tr>
<tr>
<td>$a, b$</td>
<td>tuples in query answer</td>
</tr>
<tr>
<td>$Q, Q(D)$</td>
<td>database, query, query answer</td>
</tr>
<tr>
<td>$D, S$</td>
<td>database (instance), database schema</td>
</tr>
<tr>
<td>$s, c$</td>
<td>rules ($c$ for comparative rules)</td>
</tr>
<tr>
<td>$S$</td>
<td>set or list of rules</td>
</tr>
<tr>
<td>$Dom(A)$</td>
<td>domain of attribute $A$</td>
</tr>
<tr>
<td>$*$</td>
<td>wildcard character</td>
</tr>
<tr>
<td>$R^a$</td>
<td>provenance of answer $a$ in $R$</td>
</tr>
<tr>
<td>$Q^a$</td>
<td>query that returns aggregate value in $a$</td>
</tr>
<tr>
<td>$m_w$</td>
<td>maximum rule weight</td>
</tr>
<tr>
<td>$m_D, m_q$</td>
<td>number of attributes, number of group-by attributes</td>
</tr>
</tbody>
</table>

### 2.1 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:

$Q_1$: `SELECT d.Gender, m.Genre FROM Movies m, Directors d` 
`WHERE m.Director = d.Name`

Applying $Q_1$ to the data in Tables 2.2-2.3 results in Table 2.4.

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 $Q(D)$ [BKWC01].

Different notions of provenance, including lineage, why-provenance, where-provenance, and how-provenance, are proposed and studied for databases
queries [CCT09]. We review those provenance types with emphasis on lineage and why-provenance, which we use in this work.

**Lineage**

In addition to often being used as a synonym for provenance, lineage is one of the earliest types of provenance. Lineage lists the tuples that caused a tuple to appear in the answer of a query [CW00]. The lineage of \( b_1 \), the first tuple in the results, w.r.t. query \( Q_1 \) is simply the set of tuples that contribute to \( b_1 \). In this case, the lineage of \( b_1 \) is \( \{r_1, r_2, r_6, r_7\} \).

We give a formal definition of lineage, as it is the provenance type we use in our work. This definition is paraphrased from definition 2.1 in [CCT09] to fit our notation:

**Definition 2.1 (Lineage).** Given a query \( Q \) over relations \( R_1, \ldots, R_n \) in database \( D \), the lineage of a tuple \( t \in Q(D) \) is a sequence \( (R'_1, \ldots, R'_n) \) of subsets \( R'_i \subseteq R_i \). Such that:
1. $Q(R'_1, \ldots, R'_n) = \{t\}$.

2. For each $1 \leq i \leq n$ and for each $t_i \in R'_i$, we have $Op(R'_1, \ldots, R'_{i-1}, \{t_i\}, \ldots, R'_n) = \phi$.

3. $(R'_1, \ldots, R'_n)$ is maximal among subsets of $R_1, \ldots, R_n$ satisfying (1) and (2).

**Why-provenance**

Intuitively, why-provenance consists of witnesses, i.e. sets of tuples that are necessary to make a tuple appear in the answer [BKWC01]. To continue with our running example, the why-provenance of $b_1$ in Table 2.4 consists of two witnesses $\{r_1, r_6\}$ or $\{r_2, r_7\}$, either of which is sufficient to explain the presence of $b_1$ in the results. In other words, having this tuple in the results of $Q_3$ does not require the tuples in both witnesses; only one witness is needed.

**Where-Provenance**

Where-provenance answers the question of where in the input that piece of data is copied from [BKWC01]. In our example, the where-provenance of $b_1[Gender]$ in Table 2.4 would be $r_6[Gender], r_7[Gender]$ since data is copied from those attributes in tuples $r_6$ and $r_7$.

**How-provenance**

How-provenance answers the question of what process the piece of data went through to end up in the output. How-provenance represented by elements of a semiring of polynomials, as seen in [GKT07], can give a more detailed account of the transformation applied to the piece of data. In our example, the how-provenance of $b_1$ in Table 2.4 would be the polynomial $r_1.r_6 + r_2.r_7$. 
This represents that $r_1, r_2$ was joined with $r_6, r_7$, and subsequent results were combined to get the final result.

We do not use why, how, or where provenance in our work and refer the reader to [CCT09] for formal definitions.

Provenance for aggregate queries

For aggregate queries, the notion of lineage is still applicable, e.g. the lineage of an answer \textit{(Male, 654.3M$\$$)} to the aggregate query,

$Q_4$: \textbf{SELECT} d.Gender, \textbf{SUM}(m.Rev) \textbf{FROM}

Movies m \textbf{JOIN} Directors d \textbf{ON} m.Director = d.Name

contains $r_1, r_2, r_3, r_5$ from Movies (Table 2.2) and $r_6, r_7, r_8$ from Directors (Table 2.3), i.e. the male directors and their directed movies.

The authors in [ADT11] extend how-provenance to aggregate queries using semimodules instead of semirings, which allows bag and set semantics in aggregate queries, and enables tracing provenance information for values rather than tuples. For example, the provenance of \textit{(Male, 654.3M$\$$)} is the aggregate polynomial expression $r_1.r_6 \otimes 275.3$ \oplus $r_2.r_7 \otimes 84.9$ \oplus $r_3.r_8 \otimes 74.7$ \oplus $r_5.r_6 \otimes 219.4$. The relation \oplus in the expression is the operation of the monoid. It can be interpreted as sum, average, min, or max, depending on the aggregate function in the query, e.g., for $Q_4$, \oplus is interpreted as a sum. The tensor product \otimes in the semimodule maps to a symbolic expression that involves tensors, scalars, polynomials, and the addition operation of the semimodule., e.g., \otimes maps the pair of $r_5.r_6$ and 219.4$ to 219.4$ when $r_5$ and $r_6$ are respectively present in Movies and Directors. With the semimodule in this example, computing the provenance will result in the aggregate query answer 654.3M$\$$.

While this provenance representation is informative, it can easily become a large and unreadable expression even for fairly simple queries.

\textit{We use lineage as the notion of provenance in our provenance summaries. Therefore, by the provenance of a query, we refer to the set of tuples in the}
Query lineage.

Compared with the other types of provenance, lineage leaves out some important provenance information; however, its simplicity is advantageous in our solution as it makes the provenance summary easy to navigate and understand. Our solution can be extended to support other types of provenance where the representation is in the form of tables or sets of tuples. We leave such extensions as future work and focus mainly on lineage as we mainly explore usability. We also augment the provenance data with impact and coverage information. Coupled with the simplicity of lineage, this allows us to present users with the information they need in a concise, clear, and comprehensible representation.

2.2 Data Summarization Rules

The work in [JGMP15, JGP16] introduces summarization rules to summarize “interesting” aspects of a table. In Chapters 3 and 4, we use them to summarize and explore the provenance of aggregate queries. In Chapter 5, we use them to refine aggregate queries. Here we review some key concepts about these summarization rules.

Definition 2.2 (Summarization rules). A summarization rule \( s \) over a relation \( R \) with schema \( R = \{A_1, ..., A_m\} \) is an \( n \)-ary tuple in which for every \( A_i \in R \), \( s[A_i] \in \text{Dom}(A_i) \cup \{\star\} \) and \( \star \) is a value not in \( \text{Dom}(A_i) \).

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

**Example 1.** The rule \( s_1 = (\star, 2015, \star, 7, \star, \star) \) is a summarization rule over Table 2.2 that matches every movie with rating 7 made in the year 2015. This matches the tuple: \( r_3 \)

For clarity, we include attribute names in the rules, e.g. \( s_1 \) in Example 7 is \( (\text{Title} : \star, \text{Year} : 2015, \text{Genre} : \star, \text{Rating} : 7, \text{Rev} : \star, \text{Director} : \star) \).
Definition 2.3 (Sub-rule and super-rule). Consider rules $s_1$ and $s_2$ over a relation $R$ with schema $\mathcal{R} = \{A_1, ..., A_m\}$. Rule $s_1$ is a sub-rule of $s_2$, denoted by $s_1 \sqsubseteq s_2$, if and only if for every $A_i \in \mathcal{R}$, $s_1[A_i] = s_2[A_i]$ or $s_1[A_i] = \star$. If $s_1$ is a sub-rule of $s_2$, $s_2$ is a super-rule of $s_1$, which we denote by $s_2 \sqsupseteq s_1$.

Example 2. Rule $s_1$ is a sub-rule of $s' = (Title : \star, Year : 2015, Genre : Romance, Rating : 7, Rev : \star, Director : \star)$, and it is a super-rule of $s'' = (Title : \star, Year : \star, Genre : \star, Rating : 7, Rev : \star, Director : \star)$.

Definition 2.4 (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 $s[A_i] = \star$. $\text{Cover}(s)$ is the set of tuples in $R$ that are covered by $s$ and $\text{Count}(s) = |\text{Cover}(s)|$ is the number of tuples covered by $s$.

Example 3. In Example 1, $s_1$ covers $r_2, r_3$, $\text{Cover}(s_1) = \{r_2, r_3\}$ and $\text{Count}(s_1) = 2$. Rule $s_2 = (Title : \star, Year : \star, Genre : Drama, Rating : \star, Rev : \star, Director : \star)$ covers the drama movies, $\text{Cover}(s_2) = \{r_1, r_2, r_5\}$ and $\text{Count}(s_2) = 3$.

Definition 2.5 (Marginal cover and marginal count). For a list of rules $S = (s_1, s_2, ...)$ over relation $R$, $\text{MCover}(s_i, S)$ 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 $j < i$. $\text{MCount}(s_i, S) = |\text{MCover}(s_i, S)|$ is the marginal count of $s_i$.

Example 4. Considering $S = (s_1, s_2)$ that summarizes Table 2.2 with rules: $s_1 = (Title : \star, Year : \star, Genre : Drama, Rating : \star, Rev : \star, Director : \star)$, and $s_2 = (Title : \star, Year : \star, Genre : \star, Rating : 7, Rev : \star, Director : \star)$, $\text{MCover}(s_1, S) = \{r_1, r_2, r_5\}, \text{MCover}(s_2, S) = \{r_3\}, \text{MCount}(s_1, S) = 3, \text{MCount}(s_2, S) = 1$.

Definition 2.6 (Score function and marginal score). The score of a list of rules $S = (s_1, s_2, ...)$ is defined as follows:

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

(2.1)
Weight is a function that returns a non-negative real number. For $s_i \in S$, $MCount(s_i, S) \times Weight(s_i)$ 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 [JGMP15]: the number of non-$\star$ values. We prove our results for a general class of monotone weight functions as in Definition 2.7.

Example 5. For $s_1$ and $s_2$, $Weight(s_1) = 1$ and $Weight(s_2) = 1$. The score of $S = (s_1, s_2)$ is $3 \times 1 + 1 \times 1 = 4$ and the score of $S' = (s_2, s_1)$ is $2 \times 1 + 2 \times 1 = 4$. The marginal scores of $s_1$ and $s_2$ are subsequently 3 and 1 in $S$, and 2 and 2 in $S'$.

Definition 2.7 (Monotone weight functions). A weight function $Weight$, or $W$ for short, is monotone if for every pair of rules $s_1, s_2$ if $s_1 \sqsubseteq s_2$, then $W(s_1) \leq W(s_2)$.

Example 6. The weight function $W$ that returns the number of non-$\star$ values is monotone. In Example 2, $s_1 \sqsubseteq s'$ and $W(s_1) \leq W(s')$ as $W(s_1) = 2$ and $W(s') = 3$. Also $s'' \sqsubseteq s_1$ and $W(s'') \leq W(s_1)$ as $W(s'') = 1$.

Definition 2.8 (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.

For the rest of the thesis, we use sets of summarization rules rather than lists of summarization rules where the score of a set of rules is the maximum score between all the possible lists containing the rules in the set.

Example 7. In Example 3, $\{s_1, s_2\}$ is a summary with score $\max(\text{Score}((s_1, s_2)), \text{Score}((s_2, s_1))) = \max(4, 4)$.

Definition 2.9 (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 $\text{Score}(S)$. 
The summarization problem is NP-hard \cite{JGP16}. The authors in \cite{JGP16} 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 ruleset $S$, 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. This is done in several iterations over the tuples in the relation. The algorithm applies ideas from the Apriori Algorithm \cite{AS94} for frequent item-set mining to prune 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 \ref{def:score_function}) is sub-modular \cite{JGP16, Lemma 3}.

**Definition 2.10** (Sub-modular set functions). A set function $f$ is sub-modular iff for any sets of items $S_1, S_2$ and an item $s \not\in S_1 \cup S_2$, if $S_1 \subseteq S_2$ then $f(S_1 \cup \{s\}) - f(S_1) \geq f(S_2 \cup \{s\}) - f(S_2)$. 

\[ 16 \]
Chapter 3

Provenance visualization

3.1 Introduction

Data provenance is any information about the origin of a piece of data and the process that led to its creation. Provenance has been researched in a number of different areas. In the scientific workflow context, provenance is used to show the processes that data went through. Thus, provenance work in workflow management has included work on visualizing and 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 much work on developing comprehensive models and representations of data provenance. However, there is very little work on presenting database provenance information in a way that does not overwhelm users. Using existing database provenance systems can be overwhelming for those who do not have prior knowledge of data provenance models or the data itself. Furthermore, provenance information can increase the data size considerably. This can have serious implications on both storage and usability. We argue that data provenance needs to be visualized and sometimes summarized to support and facilitate broad exploration. In this chapter, we present Pastwatch, which is a first step toward making prove-
nance data widely usable without the need for deep technical knowledge.

We identify two main challenges:

1. Provenance information needs to be visualized in a way that facilitates exploration. Visualizations should include query results, the processes that led to the creation of the query results, and the source tables.

2. Provenance information for aggregate query results can be large and prohibitive for exploration. An aggregation query that counts the number of tuples in a relation returns a single number. The provenance of this single number can be the whole relation. This creates a barrier for users to look through provenance information and find meaningful information.

Pastwatch addresses these two challenges as we describe in the following subsection.

3.1.1 Contributions

In this chapter, we review provenance research and use our findings to create Pastwatch, a first step toward comprehensible database provenance. In particular, our contributions are:

- We present, to the best of our knowledge, the first system with visualization of provenance information in a relational database system.

- We survey provenance research and create a concise set of desirable features for a provenance exploration system. We identify the criteria for designing this system and implementing it.

- We validate Pastwatch with a user study that shows how it improves accuracy and efficiency.
3.2 System design principles

In this section, we lay out a set of design principles that should be followed by future database provenance exploration systems and are adhered to by our Pastwatch system.

In particular, in this chapter, we seek to solve the problem of provenance exploration, which is to present provenance information without overwhelming size and complexity. Provenance exploration in turn allows non-experts to understand the provenance of their relational data and query results. Insightful visualizations of carefully curated provenance information give users the ability to use provenance information to explore the data.

We define a data provenance exploration system to support data provenance exploration through two components: 1) A back-end DBMS with support for provenance, 2) A front-end user interface with visualizations. In this section, we characterize design principles for these two components which are considered in the design of Pastwatch.

3.2.1 Provenance System Principles

What sets an exploration system apart from traditional systems is that the traditional approaches rely on the user to query and explore the results via data manipulation languages. In traditional approaches, the models can be complex and some query languages can be unintuitive, making things difficult.

Several data provenance papers have looked at the desirable features of a data provenance system [GD07, SPG05]. Authors in such surveys list desirable features for provenance in data management systems, including user interfaces and visualizations. In this section, we combine these features and augment them with new features, based on recent literature, to detail the principles for a data provenance exploration system for which Pastwatch is a first attempt. Based on this background research, our principles for a data
provenance exploration system are as follows:

1. **Support multiple types of provenance.** As argued in [GD07], a comprehensive provenance exploration system should support multiple types of provenance. The user could need different semantics depending on the scenario, this includes access to different visualizations. In this work, we present visualizations that work for why provenance and lineage.

2. **The back-end system should allow for provenance data to be queried.** [GD07] Without this feature, it would be impossible to show provenance information to the user, let alone visualize it meaningfully.

3. **Provenance exploration systems should support provenance information at different granularities.** [DF08] Provenance exploration systems must support both tuple and table-level granularities in order to make decisions.

4. **Store provenance data in a way that lets provenance information be decoupled from the data.** Provenance and its data should be distinct enough so that the user can tell where it comes from and to which tuple the provenance belongs. Storage is key: The way provenance is stored should allow for querying and isolating of provenance information while retaining the link to the data.

5. **The model should be efficient and simple enough to query and visualize.** A provenance exploration system based on a relational DBMS allows for improved efficiency. This allows it to perform provenance operations with the good performance provided by years of optimization of SQL queries. It also supports SQL-like manipulation that can power visualizations.

6. **Make provenance simpler to use with different dissemination techniques.** As seen in workflow provenance systems, provenance ex-
ploration needs visualizations that give the user the ability to explore freely.

7. **Provenance of large size results of aggregation queries should be summarized.** Provenance of aggregation query results can be large and daunting. A summary of such provenance information can help users make the most of this information. This problem is quite challenging, as we discuss in detail in Section 3.3.2.

We designed *Pastwatch* in accordance with the above design principles. *Pastwatch* supports multiple types of provenance; the back-end supports both why- and where-provenance. The back end is a relational DBMS, which means it is efficient and simple to query, and provenance information can be queried separately. Finally, visualization and summarization of data and provenance are first-class components of *Pastwatch*.

### 3.2.2 Visualization principles

In section 3.2.1, we detail what the system can provide to the user of provenance information. In this section, we explore what the user can do when using this provenance information in the form of visualizations. The design of the visualization and exploration component is guided by the following set of visualization principles:

- **Overview first, zoom and filter, then details on demand** [Shn96]. This principle is essential for dealing with large-scale information. The main goal of our visualizations is to show the user a summarized view of the data (usually an aggregation). The user can browse through a custom overview by specifying an aggregation or filtering query. The user can then zoom in and look at specific tuples and look at their provenance information.
• **Recognition over recall** [Mun14]. Navigating through data tables and drilling down can confuse the user as to where they are in the system or the provenance of query results. Keeping track of where the user is via visual elements beats having to remember where they are. Hence we create a provenance graph to show a persistent overview that highlights which level the user is currently browsing.

• **Appropriate encoding for the underlying data.** We use a graph to show the provenance information sources and how they relate to the results. We use a horizontal treemap to represent the summarization rules. Using this visualization allows nesting rules within rules. The score of a rule is mapped to the area of the rectangle that contains the rule.

• **Multiple views are most effective when explicitly linked** [Rob07]. The user is given the option to utilize multiple views at once. Highlighting one element in a certain view (the bar chart for example) would highlight the same item on a different view (country on a map for example).

• **Aggregation and filtering to reduce the data.** Visual idioms have limitations regarding the number of items they can display [Mun14]. Reducing guarantees better scalability for large datasets and more efficient visualizations.

### 3.3 Pastwatch overview

Our goal in implementing *Pastwatch* was to provide users with all the facilities they need to use provenance information. *Pastwatch* has two main components:

1. A data and provenance visualization component.
2. A data summarization component that summarizes the provenance of tuples produced by aggregation queries. In this section, we go into detail on each component.

![Diagram of Pastwatch system](image)

**Figure 3.1**: An overview of *Pastwatch*. The system takes a query and returns the query results to the user, including visualization of provenance information. The user asks for the provenance of a tuple in the query results. The summarization component takes the provenance, summarizes it and returns a ranked list of rules of size $k$.

### 3.3.1 Visualization components

As seen in Figure 3.2, *Pastwatch* visualizes provenance information that is stored in relational tables side by side with their respective tuples. The data alone can be large and overwhelming, and the provenance information
Figure 3.2: *Pastwatch* interface. (A) Main interface with 1) Schema list, 2) Query input, and 3) Query results. (B) The provenance graph shows the history of data to the most recent results, including database operations (Union and Select in this case). (C) The overview visualization is: a) A linked view of a world map and bar chart. Showing geographic locations and a number of companies respectively. b) A treemap visualization is also available for datasets that do not contain geographic information. (D) The summarization rules as a horizontal treemap.
increases the size and complexity. The user needs the information to be divided into smaller subsets. To maximize comprehensibility, provenance information should be hidden until the user asks for it.

We created two components to browse the provenance information: the overview visualization and the provenance graph. The overview visualization summarizes the data and presents the user with a manageable overview. In this overview, the user can click on any data item to see its provenance. The provenance graph presents an interactive overview of the provenance of a piece of data. Furthermore, users can ask for the provenance information of a piece of aggregated data to be summarized as a set of summary rules. Summary rules are shown in Figure 3.2 where a horizontal treemap embeds the score of a rule as height and color.

### 3.3.2 Summarization of provenance of aggregate query results

**Data summarization rules**

<table>
<thead>
<tr>
<th>ID</th>
<th>Title</th>
<th>Director</th>
<th>Gender</th>
<th>Year</th>
<th>Genre</th>
<th>Rev (M$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>t₁</td>
<td>Lincoln</td>
<td>Steven Spielberg</td>
<td>M</td>
<td>2012</td>
<td>Drama</td>
<td>182</td>
</tr>
<tr>
<td>t₂</td>
<td>Bonjour</td>
<td>Anne Eleanor Coppola</td>
<td>F</td>
<td>2016</td>
<td>Comedy</td>
<td>13</td>
</tr>
<tr>
<td>t₃</td>
<td>Sicario</td>
<td>Denis Villeneuve</td>
<td>M</td>
<td>2015</td>
<td>Action</td>
<td>47</td>
</tr>
<tr>
<td>t₄</td>
<td>Mamma Mia</td>
<td>Phyllida Lloyd</td>
<td>F</td>
<td>2008</td>
<td>Comedy</td>
<td>144</td>
</tr>
<tr>
<td>t₅</td>
<td>Pitch Perfect 2</td>
<td>Elizabeth Banks</td>
<td>F</td>
<td>2015</td>
<td>Comedy</td>
<td>184</td>
</tr>
</tbody>
</table>

Table 3.1: Movies table

The work in [JGPT6] introduces summarization rules to summarize “interesting” aspects of a table. In this section, we use our version of such rules to summarize and explore the provenance of aggregate queries. The provenance of an aggregate query is all the tuples that contributed to the aggregate value. Therefore, it is usually a much larger set than the results
of an aggregation query.

**Definition 3.1 (Summarization rules).** A summarization rule $s$ over a relation $R$ with schema $\mathcal{R} = \{A_1, ..., A_n\}$ is an $n$-ary tuple in which for every $A_i \in \mathcal{R}$, $s[A_i] \in \text{Dom}(A_i) \cup \{\star\}$.

The value $\star$ is a wildcard that is not in $\text{Dom}(A_i)$. It matches every attribute value and allows the rule to summarize multiple tuples.

Example: $s_1 = (\star, \star, \star, 2015, \text{Action}, \star)$ is a summarization rule over Table 3.1 that matches every Action movie from 2015.

In [JGP16] the score of a list of rules $S = (s_1, s_2, ...)$ is defined as follows:

$$\text{Score}(S) = \sum_{s_i \in S} M\text{Count}(s_i, S) \times \text{Weight}(s_i).$$

(3.1)

where the score of a set of rules is the maximum score between all the possible lists containing the rules in the set. **Weight** is a monotone function that returns a non-negative real number. The weight function conveys how well a rule summarizes the values in a table. We use the common weight function from [JGP16], $w =$ the number of non-$\star$ values.

The summarization problem is defined as follows: Given a relation $R$ and a fixed value $k$, the summarization problem is to find a set of rules $S$ with $|S| = k$ and maximum $\text{Score}(S)$. It is an NP-hard problem [JGP16]. The authors of [JGP16] 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 ruleset $S$ 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. This is done in several iterations over the tuples in the relation. The algorithm applies ideas from the apriori algorithm [AS94] for frequent item-set mining to prune some of the rules without computing the score function. The approximation guarantee in the algorithm is based on the
fact that the Score function is a sub-modular set-function [JGP16, Lemma 3]. For functions other than count, the score function is not sub-modular. In the following section, we show our own definition of a score function that takes impact into consideration and is sub-modular for the avg and sum functions.

**Pastwatch summarization**

In Pastwatch, we present AScore for a set of rules as the maximum score between every possible list that contains the rules in the set. AScore generalizes Score by replacing MCount(s_i, S) with MAgg^t(s_i, S). Unlike Score, the AScore function considers the impact of the tuples covered by rules in S on the aggregate query result Q^t(R). To measure this impact, we use sensitivity analysis, 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 [KLD11]. This is noted by the superscript t in AScore^t. For query Q^t, we measure the impact of a rule s_i by sensitivity analysis w.r.t. the tuples in the marginal cover set MCover(S, s_i). The following function defines MAgg^t based on the sensitivity of Q^t:

\[
MAgg^t(s_i, S) = \sum_{r \in MCover(s_i, S)} |Q^t(R) - Q^t(R \setminus \{r\})|.
\] (3.2)

**Example:** The user asks a query for the average revenue of all movies in Table 3.1. The user proceeds to click on the tuple: \( t = (\text{Comedy}, 113.6 \ M\$) \), asking for a summary of the provenance of this tuple. The Why-provenance of \( t \) is three tuples: \( t_2, t_4, \) and \( t_5 \). In this example \( t_2 \) is an interesting tuple because \( t_2[\text{Rev}] = 13 \). This means \( t_2 \) has a considerable impact on the average revenue of comedy movies. In this case, we prefer rules that can single out and highlight this tuple. Say there is a list of rules \( S_1 \) with only one rule \( s_1 = (\ast, \ast, F, \ast, \text{Comedy}, \ast) \) that covers and explains all tuples \( t_2, t_4, t_5 \). AScore would assign a higher score to a set of rules \( S_2 = (s_2, s_1) \) with \( s_2 = (\ast, \text{Eleanor Coppola}, F, \ast, \text{Comedy}, \ast) \) that highlights the movie with the
highest impact on the average result.

\( AScore \) is sub-modular for aggregate functions Sum and Average which means we can use the greedy algorithm with the same guarantees.

We elaborate on and extend our work on aggregate summarization in Chapter 4.

### 3.4 User experiment

We performed a quantitative user study\(^1\) to validate the visualization and exploration components of *Pastwatch*. In this study, the users interacted with a visualization of the provenance of a dataset and its provenance metadata to answer a set of questions. The users also tried to answer similar questions using a web interface that presented the provenance in HTML tables in the same format of Perm [GMA13]. We purposefully did not include querying of provenance as an interaction paradigm in this user study. While querying provenance is mature enough to support any use cases, we felt our users did not have the expertise to write queries using a query language like SQL. Despite citing experience with data or data management systems, most of them were not comfortable enough to write SQL queries during a user study. The other reason is that *Pastwatch* was initially developed as part of the GLEI system. The user study described here was meant to assess the provenance component of the GLEI system which has a different group of users than a general provenance data management system. While users performed these tasks, we measured the time of completion and answer accuracy. The users also evaluated the difficulty of each task and offered subjective feedback at the conclusion of the study. We compiled the full materials used in the study in Appendix A.

\(^1\) The study was approved as minimal risk by the Behavioural Research Ethics Board at the University of British Columbia. UBC BREB number: H16-00797.
3.4.1 Hypotheses

Our hypotheses for this user study were:

- **H1**: The users will find it less difficult to answer questions using the Pastwatch interface than the table interface.

- **H2**: The users will complete the tasks faster using the Pastwatch interface than the table interface.

- **H3**: The users will answer the questions more accurately using Pastwatch than the table interface.

A description of the different interfaces and measures follows in the next few sections.

3.4.2 Participants and setup

All participants were graduate and undergraduate students from the Computer Science and Electrical and Computer Engineering departments at the University of British Columbia who had sufficient familiarity with data and relational tables. Familiarity is determined by taking an entry-level course in databases or equivalent. However, none of the participants had any prior experience with provenance information.

The dataset used in the study is a real-world financial dataset used in creating Global Legal Entity Identifiers (GLEIs) for financial institutions [LLP14] [CM19]. This is a real-world financial dataset collected from various sources for financial institutions to create a single, universal identifier for entities that are involved in any financial transaction. The data characteristics are as follows: The number of tuples: 250k each tuple corresponds to a real-world financial entity. The number of tables in the GLEI dataset: 1. The total number of attributes in the tables: 12. The number of source tables in the provenance: 6.
The study was conducted with 21 participants: 6 women, 14 men, and 1 participant who did not wish to disclose. Participants ranged from second-year undergrads to fourth-year Ph.D. students.

Study sessions were done with a single user at a time on a Macbook Pro 15” laptop. Participants who were not familiar with the Apple trackpad were offered a mouse.

3.4.3 Procedure

Each study session started with a survey to collect demographic information and assess participants’ familiarity with the study concepts. The users were shown a demo of the system component that produces the table format output. The demo shows an example which is a smaller-scale problem that resembles the four tasks the participants have to perform. Participants were also shown how to access the visualization component and where to input the answers to the tasks’ answers. The participants were also provided details on the dataset and what provenance means. The participants were then given instructions on how to perform their tasks with task questions and subjective questions. At the end of each session, participants were handed a questionnaire to evaluate the tool and provide comments.

The participants’ tasks consisted of two different sets of questions. In the first set, users were asked to locate a data item in the output and then locate its origin point. Users were asked to find a specific entity in the dataset using its unique identifier. They were then asked to locate the source of this entity using provenance information. In the second set, users were asked to choose any entity from a certain financial source and to find out how it has changed on update. Participants performed a task using Pastwatch or the table interface. 11 participants started with the visualization method, the other 10 started the first task with the table interface. The time of completion for each task and solution accuracy was measured. The full details of tasks and questionnaires are in the appendix in Chapter A.1.2.
Figure 3.3: A screenshot of a table showing provenance information. The prov prefix tells the user they are looking at provenance information. The name of the table follows the prefix, and then the name of the attribute.

Users did not write any queries to get the results, they simply clicked on a button to generate the query results. In the table method, all the information is presented in a single table. In the Pastwatch, users interacted with the world map and provenance graph to navigate the dataset and its provenance. Each session lasted about 30-40 minutes.

### 3.4.4 Experimental design and analysis

The study had one within-subject factor: the method used to find answers to the questions in each task and the following levels:

1. A table representation of the results of a provenance generating query
2. The Pastwatch interface. We refer to it as visualization.
The dependent measures were:

1. The accuracy of answers (there was a binary correct or incorrect answer to each question).

2. The time it took to complete a task in seconds. The timer starts as the user begins to explore the data and ends by the time they have answered the main questions. The follow-up questions about the difficulty and subjective feedback are not timed.

3. The subjective perceived difficulty of a task on Likert scale of 1-5. 1 is very difficult and 5 is very easy.

The tasks were counterbalanced across interfaces. Half the users saw the table interface first, and the other half saw the Pastwatch interface first. Because the data does not follow a normal distribution, we used the Wilcoxon signed rank test which tests the median difference between two sets of observations. [McD09]

3.4.5 Results and discussion

**Efficiency, Perceived difficulty, and Accuracy** We saw statistically significant effects of the different methods on the time it took to complete a task, the perceived difficulty, and the accuracy ($P < 0.01$). Participants spent less time completing tasks using Pastwatch ($M = 163$ seconds) than they did using the table interface ($M = 262$ seconds). On a scale of 1 being very difficult and 5 very easy, participants rated the task ($M = 3.95$) using Pastwatch and ($M = 2.42$) using the table interface. Participants, on average, were able to answer 92% of the questions correctly using Pastwatch compared to 82% using the table interface.

**Subjective feedback** More than 80% of users commented that they found Pastwatch easier to deal with and easier to figure out the provenance relationships between tables.
Figure 3.4: Results for time (in seconds) broken down by task for each method. Error bars correspond to standard error. Case refers to task.

Figure 3.5: Results for perceived difficulty (1 is very difficult - 5 very easy) broken down by task for each method. Error bars correspond to standard error. Case refers to task.
The most requested feature was a search box; that feature was disabled prior to the study so as not to bias the study toward the visualization tool. The browser search function also did not work with Pastwatch due to incompatibility with the scroll bar plug-in. This was left unfixed to keep the focus on general exploration using the visualization tool.

20 participants rated it as likely or very likely that they would use the provenance tool in their work to confirm the reliability or accuracy of a piece of data/dataset. Only one user was not interested in using the provenance tool at all listing the reason as “because I prefer to write queries rather than trusting my eyes.”

Other comments from the subjective feedback survey: “As a normal user; this tool would be useful in any context where I know it is possible that my data comes from several sources.” Another participant states: “First: it is a good tool to have. Second: in the case of troubleshooting it is very easy to go back to the source and check the accuracy of the information. I think it makes troubleshooting easier in general.”

Discussion Visualization of database provenance is not the end-all solution to all provenance information problems. However, it is a first step toward a comprehensive understanding of provenance. We use the table representation of provenance with some visual aid to give a fair baseline to the user. There are no other current approaches that can offer a comparable visualization. Work-flow provenance systems work with different semantics and other database approaches offer little in terms of user interface we could compare against. To make this a fair comparison, no other features such as search, filter, or coloring were enabled in the user study. Enabling such features Pastwatch would likely perform above the numbers seen in the results.
3.5 Related work

**Provenance Database Systems.** Data provenance is studied extensively in the database literature. One of the earliest applications of data provenance is lineage in data warehouses as seen in WHIPS [CW00]. It generates the provenance of views in a data warehousing environment and supports a manipulation language that extends SQL. The provenance information in WHIPS is the view definition and source tuples. DBNotes [CTV05,BCTV05] is a database annotation system that uses where-provenance. In DBNotes, the provenance information is simply user-made annotations propagated with the results. DBNotes does not support aggregation due to the semantics used and the way annotations are managed. Orchestra [KIT10] is a collaborative data-sharing system where users share data using schema mappings; it uses provenance semirings to represent how-provenance information. Perm [GMA13] generates and queries provenance information using easily optimizable SQL. The semantics of Perm is 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 [CCT09,SPG05].

Even though systems like WHIPS [CW00] and Perm [GMA13] offer extensible and simple SQL-like syntax, provenance information is still prohibitive to users and can easily be overwhelming for large datasets. The natural approach would be to visualize this information.

More recent systems include developing provenance systems as an extra layer above the DBMS, such as [AFG+18]. Other systems include more features to support provenance and probabilistic queries [SJMR18] or aim to improve the performance of provenance capture with optimization techniques [PW18].

**Provenance Visualizations.** A lot of work on user interaction with provenance information has been done in the field of scientific work-flow provenance [KOF+16, ZGSB04, CH06, BYB+13]. Such systems offer visual
graphs that represent the processes that produced or changed the data. Database provenance has different semantics that do not adhere to the same graph-like representation without some modification. We also offer a summary of provenance tuples as a means to make it easier for the user to understand provenance information.

**Provenance summarization.** The most similar work to our work in summarization is [ABD+15] in which the provenance of an aggregate query is represented as a polynomial expression, i.e. how-provenance [ADTH]. Our provenance summaries are shorter polynomial expressions that highlight 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 [ABD+15]. The algorithm for generating these summaries does not involve aggregated values within the data and thus can fail to capture the user’s interest [ABD+15]. We utilize quantitative attributes in the data, such as revenue, to make more meaningful summaries. We discuss more general summarization methods in the next chapter.

### 3.6 Conclusions and Future Work

In this chapter, we have looked extensively at provenance research. We have identified the most desirable features in a provenance system and used them to implement *Pastwatch*. We developed *Pastwatch* to support data exploration using provenance information. *Pastwatch* is ideal for broad exploration with little knowledge of provenance semantics or the data itself.

We used this system to develop a visualization component to visualize the provenance of query results and explore provenance information. We performed a user study to validate it. The results of our user study show that *Pastwatch* makes it easier for users to perform tasks with the data and to find answers faster. Our summarization approach for aggregation provenance makes it easier for users to find relevant tuples. We believe that a solution
that makes provenance information more manageable for users fills a gap in this area and is a valid contribution to this field of research. It is also a first step toward better integration of provenance information within data systems.

However, there is still a lot to be done in this field before a single approach is deemed complete. The size of the provenance data is still too large to handle, and scalability is a real concern. We feel that solutions in the vein of Pastwatch are a step in the right direction. In terms of data summarization, more research needs to be done to make sure summaries are comprehensive and cover all users’ needs. In the next chapter, we delve deeper into the summarization of provenance information and present different types of summaries to extend the work we started here.
Chapter 4

Provenance summarization

4.1 Introduction

As explained in Chapter 3, 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 without prior knowledge of provenance models or the data itself can be overwhelming. Furthermore, provenance information can increase the data size considerably. This can have serious implications on both storage and usability. We argue, as motivated in the previous chapter, that data provenance needs to be summarized to support broad exploration. We made a first attempt at summarizing and presenting provenance information in Chapter 3. In this chapter, we extend this work to produce more comprehensive summaries for different scenarios.

This chapter studies provenance summarization and exploration for aggregate query results in relational databases. We focus on aggregation 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. The following running example explains the problem and our idea of our solution.

Example 8. Consider the following aggregate SQL query, $Q_1$, over the IMDB
table in Table 4.3 that contains data about a few movies and their respective directors from the IMDB dataset.

\[ Q_1 : \text{SELECT Gender, SUM(Rev) AS SumRev} \]

\[ \text{FROM IMDB GROUP BY Gender} \]

The query asks for the sum of the revenues of the movies directed by directors of different genders. Table 4.1 shows \( Q_1 \)'s result. For answer \( a_2 \), highlighted in Table 4.1, the provenance contains all the male directors and their movies. In this example, the provenance consists of seven tuples in Table 4.3; however, in practice, querying millions of movies and directors in the 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.

A summary can provide information showing that the revenue is dominated by a small set of movies allowing the user to see that the data is badly skewed. A different type of summary could show that movies made by male and female directors have different revenue patterns. This shows the user that any analysis of the two groups needs to account for factors such as genre, duration, and production costs. We will show summaries that can aid users in those scenarios in Examples 10 and 11.

<table>
<thead>
<tr>
<th>ID</th>
<th>Gender</th>
<th>SumRev</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a_1 )</td>
<td>Female</td>
<td>624.6 M$</td>
</tr>
<tr>
<td>( a_2 )</td>
<td>Male</td>
<td>4.208 B$</td>
</tr>
</tbody>
</table>

Table 4.1: \( Q_1 \)'s result

<table>
<thead>
<tr>
<th>ID</th>
<th>Gender</th>
<th>Genre</th>
<th>CountM</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a_3 )</td>
<td>Male</td>
<td>Drama</td>
<td>2</td>
</tr>
<tr>
<td>( a_4 )</td>
<td>Female</td>
<td>Comedy</td>
<td>2</td>
</tr>
<tr>
<td>( a_5 )</td>
<td>Male</td>
<td>Action</td>
<td>3</td>
</tr>
<tr>
<td>( a_6 )</td>
<td>Male</td>
<td>Comedy</td>
<td>1</td>
</tr>
<tr>
<td>( a_7 )</td>
<td>Male</td>
<td>Horror</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 4.2: \( Q_2 \)'s result

Our provenance summaries consist of rules that summarize groups of tuples. These rules were introduced for summarizing relational data and are

\[ \text{https://developer.imdb.com} \]
<table>
<thead>
<tr>
<th>ID</th>
<th>Title</th>
<th>Year</th>
<th>Genre</th>
<th>Rating</th>
<th>Rev (M$)</th>
<th>Country</th>
<th>Duration</th>
<th>Director</th>
<th>Gender</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td>Lincoln</td>
<td>2012</td>
<td>Drama</td>
<td>8</td>
<td>275.3</td>
<td>US</td>
<td>120 min</td>
<td>Steven Spielberg</td>
<td>Male</td>
</tr>
<tr>
<td>t2</td>
<td>Sicario</td>
<td>2015</td>
<td>Drama</td>
<td>8</td>
<td>84.9</td>
<td>US</td>
<td>90 min</td>
<td>Denis Villeneuve</td>
<td>Male</td>
</tr>
<tr>
<td>t3</td>
<td>Crimson Peak</td>
<td>2014</td>
<td>Horror</td>
<td>7</td>
<td>74.7</td>
<td>Mexico</td>
<td>90 min</td>
<td>Guillermo del Toro</td>
<td>Male</td>
</tr>
<tr>
<td>t4</td>
<td>Bonjour Anne</td>
<td>2014</td>
<td>Comedy</td>
<td>5</td>
<td>8.9</td>
<td>US</td>
<td>90 min</td>
<td>Eleanor Coppola</td>
<td>Female</td>
</tr>
<tr>
<td>t5</td>
<td>The Terminal</td>
<td>2004</td>
<td>Comedy</td>
<td>7</td>
<td>219.4</td>
<td>US</td>
<td>90 min</td>
<td>Steven Spielberg</td>
<td>Male</td>
</tr>
<tr>
<td>t6</td>
<td>Pacific Rim</td>
<td>2013</td>
<td>Action</td>
<td>7</td>
<td>411.5</td>
<td>Mexico</td>
<td>120 min</td>
<td>Guillermo del Toro</td>
<td>Male</td>
</tr>
<tr>
<td>t7</td>
<td>Furious 7</td>
<td>2015</td>
<td>Action</td>
<td>8</td>
<td>1.516 B$</td>
<td>Australian</td>
<td>150 min</td>
<td>James Wan</td>
<td>Male</td>
</tr>
<tr>
<td>t8</td>
<td>Jurassic World</td>
<td>2015</td>
<td>Action</td>
<td>8</td>
<td>1.627 B$</td>
<td>US</td>
<td>150 min</td>
<td>Colin Trevorrow</td>
<td>Male</td>
</tr>
<tr>
<td>t9</td>
<td>Mamma Mia</td>
<td>2008</td>
<td>Comedy</td>
<td>6</td>
<td>615.7</td>
<td>US</td>
<td>120 min</td>
<td>Phyllida Lloyd</td>
<td>Female</td>
</tr>
</tbody>
</table>

### Table 4.3: IMDB table

<table>
<thead>
<tr>
<th>ID</th>
<th>Country</th>
<th>Duration</th>
<th>Gender</th>
<th>Cvg(%)</th>
<th>Imp(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1</td>
<td>US</td>
<td>*</td>
<td>Male</td>
<td>57</td>
<td>52</td>
</tr>
<tr>
<td>s2</td>
<td>*</td>
<td>90 min</td>
<td>Male</td>
<td>42</td>
<td>9</td>
</tr>
</tbody>
</table>

### Table 4.4: A basic summary

<table>
<thead>
<tr>
<th>ID</th>
<th>Country</th>
<th>Duration</th>
<th>Gender</th>
<th>Cvg(%)</th>
<th>Imp(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>s3</td>
<td>*</td>
<td>150 min</td>
<td>Male</td>
<td>28</td>
<td>74</td>
</tr>
<tr>
<td>s4</td>
<td>US</td>
<td>*</td>
<td>Male</td>
<td>57</td>
<td>52</td>
</tr>
</tbody>
</table>

### Table 4.5: An impact summary

<table>
<thead>
<tr>
<th>ID</th>
<th>Country</th>
<th>Duration</th>
<th>Genre</th>
<th>Cvg $a_4$</th>
<th>Cvg $a_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>c1</td>
<td>*</td>
<td>90 min</td>
<td>Comedy</td>
<td>50</td>
<td>100</td>
</tr>
<tr>
<td>c2</td>
<td>US</td>
<td>120 min</td>
<td>Comedy</td>
<td>50</td>
<td>0</td>
</tr>
</tbody>
</table>

### Table 4.6: A comparative summary
applied in several frameworks for other purposes [WZRY18, JGP16, JGMP15, EGAG+14] (see Section 4.2 for detail). Before we explain our provenance summaries, we briefly review these rules and the summaries they produce in [JGP16]. We refer to those summaries as basic summaries. We position our solution as different summaries that can help users in the scenarios mentioned in Example 8.

Example 9. A summarization rule over a relation is a tuple of the same schema with some wild-card values, ⋆s, that match any value in the relation and enable the rule to cover and summarize several tuples [JGP16]. For example, rule s_1 in Table 4.4 covers movies made in the US and directed by male directors. A basic summary of a relation is a set of k rules, each of which summarizes some “interesting” aspects of the relation. Table 4.4 is a basic summary of IMDB (Table 4.3) with k = 2, where we omitted the attributes with ⋆s across different rules, e.g. Title and Rev. 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-⋆ values in the rule, whereas coverage is the number of tuples in the relation covered by the rule [JGP16]. For example, the coverage and weight of s_1 are 4 (57% of the 7 movies directed by males) and 2, and its score is 4 \times 2. In Table 4.4, the coverage percentages are shown as values of a new attribute.

We will use the basic summaries and the summarization rules in Example 9 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 a table, they are not as helpful in summarizing the 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 10. Table 4.5 shows an impact summary of the provenance of a_2 from Table 4.1. Rules in this summary offer advantages over the rules in the
basic summary. Impact summaries contain rules, such as $s_3$, that consider the provenance tuples’ impact on the aggregate query answer. They differ from basic summaries with rules, such as $s_2$, that cover more tuples. $s_3$ covers fewer tuples than $s_2$, but the tuples covered by $s_3$ account for a much higher impact (74%) than the tuples covered by $s_2$ (9%) because long movies (150 minutes) include blockbusters (i.e., high revenue movies) such as *Jurassic World* and *Furious 7*.

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

$$Q_2: \text{SELECT Gender, Genre, COUNT(*) AS CountM}$$

$$\text{FROM IMDB GROUP BY Gender, Genre}$$

The query result is shown in Table 4.2. Consider two highlighted answer tuples $a_4$ and $a_6$, that specify the number of female directors and male directors of comedy movies. Table 4.6 is a comparative summary with a rule that covers tuples from the provenance of both answers. The score of a rule in such a summary 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 4.6, the new attributes show the coverage percentages of the tuples from the provenance of $a_4$ and $a_6$, which help the user quickly identify key differences between the two groups. For example, $c_1$ is a rule that shows that movies in the comedy genre that are of length 90 minutes are common for both male and female directors but make a larger percentage of the movies made by females. $c_2$ shows that female directors have longer movies that span about 120 minutes than male directors. These two rules highlight important differences between...
the 2 groups which can compel the user to dig deeper and analyze individual movies that may have a different sub-genre or may even be misclassified.

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. We present more examples that illustrate the usefulness of our summaries in our case study in Section 4.6.5.

Also note that while we apply comparative summaries for provenance summarization, in general, they can be used to summarize similarities and differences between two sets of tuples of the same schema and for other purposes, beyond provenance summarization.

To specify good impact summaries and comparative summaries, we define score functions by blending the factors in Examples 10 and 11, e.g. coverage, impact, and weight.

We present impact and comparative summaries to the user as interactive summarization rules that allow the user to explore the data. In our running example, the user can select an answer for which to create an impact summary or a pair of answers for comparative summaries. Then the user can continue to explore the provenance by selecting one of the rules and applying the drill-down operator that shows super-rules with fewer ⋆s and reveals more detail of the provenance covered by the selected rule. Unlike the solution in [JGP16], which produces summarization rules over a single relation, we use our approaches to produce summarization rules for provenance tuples that may involve multiple relations (cf. Section 4.3 for detail).

4.1.1 Contributions

Our contributions in this chapter are as follows:
• We define provenance summaries, impact summaries, and comparative summaries, for the 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 4.3).

• We present efficient algorithms for producing both impact and comparative summaries (Section 4.4).

• We present novel interaction methods and visualization techniques in a user interface for impact and comparative summaries (Section 4.5).

• We conduct performance experiments and a user survey to show that our proposed solutions perform well and provide useful summaries (Section 4.6).

### 4.2 Related work

**Provenance summarization.** The most similar work to our work in this chapter is [ABD+15] in which the provenance of an aggregate query is represented as a polynomial expression, i.e. how-provenance [ADT11]. Our provenance summaries are shorter polynomial expressions that highlight 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 [ABD+15]. The algorithm for generating these summaries does not involve aggregated values within the data and thus can fail to capture the user’s interest [ABD+15]. We utilize quantitative attributes in the data, such as revenue, to make more meaningful summaries. Our user study shows that users favor impact summaries over basic summaries.

**Provenance Database Systems.** There are several frameworks for generating and managing provenance data in data management [CW00, GMA13, AFG+18, PW18, SJMR18]. Perm [GMA13] is a common provenance framework that manages provenance information using easily optimizable SQL.

44
The semantics of Perm are similar to why- and where-provenance. We use Perm to generate the provenance we summarize. Even though systems like WHIPS [CW00] and Perm [GMA13] offer extensible and simple SQL-like syntax, provenance information is still prohibitive to users and can easily be overwhelming for large datasets. The natural approach would be to summarize this information. Our provenance summaries provide a tool for navigating and understanding provenance data and can be used in applications that require understanding the provenance of aggregate queries, including reproducibility [RDA+20], collaborative scientific research [JDB20, CLMT22, SdOVM18], query composition, and query debugging [ADG18, ME22].

**Query Explanation.** Query explanation is one of the main applications of our provenance summaries. It has been extensively studied with many techniques to generate explanations. Sensitivity-based techniques generate explanations using intervention and measuring the impact of data modification on query answers [WM13]. Complaint-based techniques start with a user complaint about query answers (e.g., why an aggregate value in the answer is higher than expected) and find records or predicates that explain the unexpected answer [RS14, ROS15]. Counterbalancing techniques explain outliers in aggregate query answers using related outliers in the opposite direction while providing counterbalance [MZGR19, MZL+19]. Recent techniques use repair models and anomalies [HW22], provenance augmented with context [LLM+22], human-in-the-loop and user interactions [MCB+20], and contrasts between aggregate query answers [AKS+18] to generate explanations.

Our application of summarization rules differs from the existing applications in [WM13, ROS15]. Those approaches use rules to explain outliers or high/low results so their summaries are more specific. This makes their techniques inapplicable to our problem without modification. Likewise, our general-purpose summaries would not work to explain specific problems with query results such as outliers. We use summary rules for summarizing the
provenance of aggregate queries and produce general summaries for provenance exploration. This requires new techniques for finding rules that consider the impact of the tuples covered by the rules on the aggregate result and finding interesting patterns that do not explain any specific patterns. We introduce impact as a new quality metric to find the best provenance summaries for aggregate queries. The impact of a rule depends on the impact of the tuples covered by the rule on the query answers. In our experiments, we use these metrics to compare our summaries with basic summaries, we do not conduct experiments against more specialized systems. While our technique focuses more on impact due to the new score function, it still works for high coverage due to the algorithm we use. Therefore, our approach and [JGMP15] are very closely related. Another reason we did not compare against the system in [ROS15] is their use of entropy to generate rules. As explained in Appendix B.4, entropy-based functions cannot be used with our algorithm out of the box to generate rules with any guarantees. If we were to repurpose the approach in [WM13] that uses sensitivity analysis to generate general rules like ours, their approach would produce similar results as those techniques are very similar to ours. Other factors for not comparing against such systems include time and availability of code. The implementation of [JGMP15] we used was our implementation made specifically for this project, which we extended and modified to develop our approach. It was simply not feasible to implement other systems within the time frame of the project.

**Statistical methods.** While we do not employ a statistical approach to finding summaries, our approach has some statistical properties and bears similarity to known statistical approaches. Our approach most of all resembles regression [Fre09]. Wherein the aim of regression is to find relationships between dependent and independent variables for the purpose of prediction, our aim is to find the highest impact tuples on an aggregate value using known values stored in the database. Other similarities appear in our sam-

46
pling approach, random sampling allows for the probabilistic choosing of tuples to summarize. We employ a sampling approach that prioritizes high-impact tuples that lowers the effect of probability and relies mostly on impact measures.

**Summarization Rules.** Summarization rules like those we use are applied in many applications, including summarizing query results [WZRY18], outlier detection and explanation [MZL+19, WM13], data quality analysis [GKKS10, WDM15], and feature selection [EGFG+18, EGAG+14] (see [DGS22] for a recent survey). Our purposes differ from these applications as we use the rules for summarizing the provenance of aggregate queries. Several quality metrics are used for finding summarization rules. **Coverage** measures the number of records summarized by the rules and **weight** specifies the amount of information provided by a rule and is often measured by the number of non-\(\ast\) values in a rule [JGMP15, JGP16]. **Diversity** is another metric that determines the uniqueness of the rules and their values in a summary [WZRY18]. We introduce impact as a new metric to find the best provenance summaries for aggregate queries. The impact of a rule depends on the impact of the tuples covered by the rule on the query answers. In our experiments, we use all these metrics to compare our summaries with basic summaries.

### 4.3 Formalizing Provenance Summarization Problems

In this section, we define impact summaries and comparative summaries, and we formalize the summarization problems for finding these summaries. A key challenge is to define new score functions that specify these provenance summaries while preserving the sub-modularity property needed to compute them efficiently. Our new score function for impact summaries computes an accumulative impact of the provenance records on the query answer in Section 4.3.2. We present an extension of the score function in Section 4.3.2 to
solve a problem that occurs when summarizing the provenance of some ag-
ggregate 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 4.3.3. 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 4.3.4.

4.3.1 Basics and Problem Setting

We assume an aggregate query \( Q \) over a database schema \( S \) as follows:

\[
Q: \text{SELECT } G_1, \ldots, G_m, f(G^*) \text{ AS } G \text{ FROM } R \\
\quad \text{WHERE } \text{conditions} \text{ GROUP BY } G_1, \ldots, G_m
\]

Listing 4.1: The general form of aggregate queries

In \( Q, \{G_1, \ldots, G_m, G^*\} \subseteq \mathcal{R} \), \( f \) is an aggregate function, such as COUNT and AVG, and \text{conditions} contains a conjunction of conditions on \( R \). \( G^* \) is the aggregated attribute (e.g. Revenue). We discuss more general queries
obtained by relaxing \( Q \) to aggregate-select-projection-join (ASPJ) queries in
Section 4.3.3.

Consider a database \( D \) of the schema \( S \). For \( a \in Q(D) \), we define \( Q^a \) in
Listing 4.2 as a query that returns \( a[G] \).

\[
Q^a: \text{SELECT } f(G^*) \text{ FROM } R \quad \text{WHERE} \\
\quad \text{conditions AND } G_1 = a[G_1] \quad \text{AND} \quad \ldots \quad G_m = a[G_m]
\]

Listing 4.2: A query that returns the aggregate value in \( t \)

Example 12. For \( Q_1 \) in Example 8, \( G_1 = Gender, G = SumRev, G^* = Rev \),
and \( f \) is SUM. For \( a_2 \in Q_1(IMDB) \), \( Q_1^{a_2} \) is the following query:

\[
\text{SELECT SUM(Rev) FROM IMDB WHERE Gender=Male}
\]

It returns \( a_2[SumRev] = 4.208B \$ \), which is the sum of the revenues of the movies directed by male directors.

48
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, $IMDB^{a_2}$ for $a_2 \in Q_1(IMDB)$ contains all the tuples in $IMDB$ excluding those movies that are made by non-male directors.

### 4.3.2 The Impact Summarization Problem

As motivated in Section 4.1, an impact summary summarizes the provenance of a tuple in the result of an aggregate query. It does a better job of showing the impact on aggregate values compared to the basic summaries in Section 2.2. Impact on a numeric value (e.g. Revenue) is a very useful tool for summarization. Moreover, basic summaries and their score function cannot address aggregate functions beyond Count, whereas the impact functions we present can cover Sum, AVG, MIN, and MAX. To define the impact summarization problem, we specify the new score function, $IScore$. Similar to the score in Definition 2.6, 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:

$$IScore^a(S) = \sum_{s_i \in S} Impact^a(s_i, S) \times Weight^a(s_i),$$  \hspace{1cm} (4.1)

$IScore$ 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 $IScore^a$ and is applied by replacing $MCount(s_i, S)$ with $Impact^a(s_i, S)$,

$$Impact^a(s_i, S) = \sum_{t \in MCover(s_i, S)} Impact^a(t).$$  \hspace{1cm} (4.2)
where \( \text{Impact}^a(t) = |Q^a(D) - Q^a(D \setminus \{t\})| \) is the impact of each tuple \( t \) on answer \( a \) 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 \cite{KLD11,WM13}. In Equation \ref{eq:impact}, we apply this analysis w.r.t. the tuples in the marginal cover set \( MCover(s_i, S) \) and consider \( Q^a \). We omit relation names, e.g. \( R \) in Equations \ref{eq:impact} and \ref{eq:iscore} in the superscripts when they are clear from context.

**Example 13.** Consider \( Q_1 \) and \( a_2 \in Q_1(IMDB) \) in Example\ref{ex:rule-marg-cov} and rule summaries \( S = \{s_2\} \) and \( S' = \{s_3\} \) with \( s_2, s_3 \) in Tables \ref{tab:rule-marg-cov} and \ref{tab:rule-marg-cov-2}. Based on the score function in Definition \ref{def:score}, \( \text{Score}(S) = MCount(s_2, S) \times \text{Weight}(s_2) = 3 \times 2 \) and \( \text{Score}(S') = MCount(s_3, S) \times \text{Weight}(s_3) = 2 \times 2 \). The new scores based on Equation \ref{eq:iscore} are \( \text{IScore}^{a_2}(S) = \text{Impact}^{a_2}(s_2, S) \times \text{Weight}(s_2) = 379 \text{ M$} \times 2 \) and \( \text{IScore}^{a_2}(S') = \text{Impact}^{a_2}(s_3, S) \times \text{Weight}(s_3) = 3.143 \text{ B$} \times 2 \). \( s_2 \) covers a big part of \( a_2 \)'s provenance while \( s_3 \) covers the most impactful parts of the provenance of \( a_2 \). Therefore, \( S' \) is preferred over \( S \) w.r.t. \( \text{IScore} \). \( \square \)

**Definition 4.1** (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 an optimization problem to find a summary \( S \) of size \( k \) with maximum \( \arg \max_S (\text{IScore}^a(S)) \). \( S \) is an impact summary.

The impact summarization problem is NP-hard because it extends the summarization problem \cite{JGP16} (Chapter \ref{chap:summarization} Section \ref{sec:summarization}), which has been proved to be NP-hard. The BRS baseline algorithm relies on the sub-modularity of \( \text{Score} \) in Definition \ref{def:score}. In Section \ref{sec:algo-imp} we present the IPS algorithm for finding impact summaries. To guarantee the algorithm’s approximation, we need to verify whether \( \text{IScore} \) is sub-modular.

**Theorem 1.** \( \text{IScore} \) in Equation \ref{eq:iscore} is sub-modular.
The proof of Theorem 1 is based on that the marginal impact of every tuple $a \in Q(D)$ in Equation 4.2, 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 B.2 for the complete proof). As a result of this theorem, we can efficiently generate approximately optimal impact summaries w.r.t. our score function.

In the rest, whenever it is clear from context, we omit the superscripts, e.g. we use $\text{IScore}$ instead of $\text{IScore}^a$.

**Impact Summaries with Contingency**

We now explain a shortcoming of the impact definition in Equation 4.2 and we present our solution to extend the impact with contingency.

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

Table 4.7: Movies table

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Gender</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_6$</td>
<td>Steven Spielberg</td>
<td>Male</td>
<td>US</td>
</tr>
<tr>
<td>$r_7$</td>
<td>Denis Villeneuve</td>
<td>Male</td>
<td>Canada</td>
</tr>
<tr>
<td>$r_8$</td>
<td>Guillermo del Toro</td>
<td>Male</td>
<td>Mexico</td>
</tr>
<tr>
<td>$r_9$</td>
<td>Eleanor Coppola</td>
<td>Female</td>
<td>US</td>
</tr>
</tbody>
</table>

Table 4.8: Directors table

**Example 14.** Consider the following query that asks the maximum rating of movies from each genre in Table 4.7.
\[ Q_5 : \text{SELECT} \quad \text{Gender}, \quad \text{MAX}(\text{Rating}) \quad \text{AS} \quad \text{MaxR} \]
\[ \text{FROM} \quad \text{Movies} \quad \text{GROUP BY} \quad \text{Genre} \]

\( a = (\text{Drama}, 8) \) is an answer to \( Q_5 \) with provenance which contains all the drama movies. There is no meaningful impact summary for \( R^a \) using the score function in Equation 4.1 and the impact value in Equation 4.2. This is because \( I\text{Score} \) is 0 for every summary. This happens because there are multiple tuples with the maximum rating 8 and removing each one from \( \text{MoviesDirectors} \) does not change \( a \). As a result, \( \text{Impact}^a \) is 0 for every rule and score is 0 for every summary. \( \square \)

Example 14 shows that the sensitivity analysis used in the definition of \( \text{Impact} \) in Equation 4.2 is not effective for certain queries, e.g. some queries with MIN or MAX aggregate functions. To solve this problem, we define a new \( \text{Impact}, \text{Impact}^a(t), \) that is based on the notion of the contingency set. \( \text{Impact}^a(t) \) is more general and extends the \( \text{Impact} \) in Equation 4.2 to address the shortcoming in Example 14.

**Definition 4.2 (Contingency Set).** Consider a query \( Q \) in Listing 4.1 over a database \( D \), a tuple \( a \in Q(D) \), the query \( Q^a \) as defined in Listing 4.2 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 \[ \text{CH03, EL02} \]. In data management, Meliu et al. use the contingency set of a tuple to define its responsibility w.r.t. a query answer \[ \text{MGMS10, MGNS11} \]. Definition 4.2 is adapted from \[ \text{MGNS11, Definition 2.1} \].
Example 15. In Example 14, \( C^a(r_1) = \{r_2\} \) and \( C^a(r_2) = \{r_1\} \). This is because removing either \( r_1 \) or \( r_2 \) does not change the answer but removing both does. 

We extend \( Impact^a(t) \) to capture contingency:

\[
Impact^a(t) = \frac{|Q^a(D) - Q^a(D \setminus C^a(t) \cup \{t\})|}{|C^a(t)| + 1}.
\] (4.3)

In Equation 4.3, the impact of each tuple is computed w.r.t. its contingency set \( C^a(t) \). The denominator \(|C^a(t)| + 1\) is called the responsibility of \( a \) for the answer \( Q^a(D) \) [MGMS10].

Example 16. Consider summary \( \{s\} \) for \( a = (\text{Drama}, 8) \in Q_5(\text{Movies}) \) in Example 14 where \( s = (\text{Title} : \star, \text{Year} : 2012, \text{Genre} : \text{Drama}, \text{Rating} : \star, \text{Rev} : \star, \text{Director} : \star) \). The rule only covers \( r_1 \), which is the only drama movie released in 2012, i.e. \( MCover(s, \{s\}) = \{r_1\} \). The score of \( \{s\} \) is 0 if we use the impact function in Equation 4.2. According to Equation 4.3, the new impact value is \( IScore^a(\{s\}) = Impact^a(s, \{s\}) \times Weight^a(s) = ((8 - 7) \times 1/2) \times 1 = 1 \). This is because the contingency set of \( r_1 \) is \( \{r_2\} \), the responsibility of \( r_1 \) is 1/2 and removing \( r_1 \) and \( r_2 \) from \( Movies \) changes the answer to \( Q^a_5 \) from 8 to 7. \( Weight^a(s) = 1 \) because \( s \) has two non-\( \star \) values including \( \text{Drama} \) from \( a \).

The new \( Impact^a(t) \) definition in Equation 4.3 is compatible with the impact in Equation 4.2 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 4.2 will be empty, its responsibility is 1, and therefore Equation 4.3 reduces to Equation 4.2. The score function with the new impact value is sub-modular which can be proved similar to Theorem 1 (see B.2).
The connection between impact summaries and provenance of aggregate queries:

The provenance of aggregate query in Listing 4.1 is of the form \( r_1 \otimes r_1[G^*] \oplus r_2 \otimes r_2[G^*] \oplus \ldots \oplus r_n \otimes r_n[G^*] \) where \( r_i \)'s are tuples in \( R \) and \( G^* \) is the aggregate result in the query. Basic summaries can summarize the tuples \( r_i \) in the provenance but do not represent the numerical values of \( r_i[G^*] \). The impact summaries include these numerical values in the impact of summarization rules to define their score and therefore represent entire provenance summaries of aggregate queries.

4.3.3 Impact Summaries for Queries with Joins

The class of queries we considered in Listing 4.1 does not allow joins, which we study in this section. Consider queries of the following form that is defined over a database \( D \) including relations \( R_1, R_2 \):

\[
Q: \text{SELECT } G_1, \ldots, G_m, f(G^*) \text{ AS } G \\
\text{FROM } R_1 \text{ AS rel}_1, R_2 \text{ AS rel}_2 \\
\text{WHERE conditions GROUP BY } G_1, \ldots, G_m
\]

Listing 4.3: The general form of aggregate queries with join

In Listing 4.3 we assume \( G_1, \ldots, G_m \) are attributed in either \( R_1 \) or \( R_2 \). Without loss of generality, we assume \( G^* \) is in \( R_1 \). Our discussion in this section easily extends to joins with multiple tables. We discuss the challenges of extending this general form to more complex queries, e.g. nested queries, in Section 4.2. Without loss of generality, we assume the join conditions are captured by the conditions in the where-clause.

For a tuple \( a \in Q(D) \), we define \( Q^a \) similar to Listing 4.2 but over \( R_i \). We also define \( R_i^a \) as the lineage (provenance) of \( Q^a(D) \) in \( R_i \), \( i \in \{1, 2\} \).

Example 17. Consider \( Q_6 \) over Movies and Directors in Tables 4.7 and 4.8 that asks the sum of the revenues of the movies directed by directors of different genders.
\( Q_6: \) **SELECT** \( d\).Gender, **SUM**(\( m\).Rev) **FROM**

Movies \( m \), Directors \( d \) **WHERE** \( m\).Director = \( d\).Name

For the answer \( a = (\text{Male}, 654.3 M\$) \), \( Q_6^a \) is the following query that returns 654.3M$:

\( Q_6^a: \) **SELECT** **SUM**(Rev) **FROM** Movies \( m \), Directors \( d \)

**WHERE** \( m\).Director = \( d\).Name **WHERE** \( d\).Gender = Male

The provenance of \( a = (\text{Male}, 654.3 M\$) \) contains its lineage tuples in both Movies and Directors: Movies\(^a\) = \{r_1, r_2, r_3, r_5\} contains every movie directed by a male director, and Directors\(^a\) = \{r_6, r_7, r_8\} are male directors with at least one movie.

The impact summaries in Section 4.3.2 summarize tuples in only one table. The impact summary for a join query of the general form in Listing 4.3 consists of two table summaries as defined next.

**Definition 4.3** (Provenance Summaries for Join Queries). Given a query \( Q \) of the form in Listing 4.3 over database \( D \), and an answer tuple \( a \in Q(D) \), an impact summary is a set \( S^a = \{(rel_1, S_1), (rel_2, S_2)\} \) in which \( rel_1 \) and \( rel_2 \) are table aliases for \( R_1 \) and \( R_2 \) in \( Q \) and \( S_1 \) and \( S_2 \) are table summaries on \( R_1 \) and \( R_2 \), respectively.

In the impact summary in Definition 4.3 we use aliases \( rel_1 \) and \( rel_2 \) to distinguish between the summaries over the tables in the query. It allows these summaries to represent queries with self-join and queries with joins over multiple tables.

To define the provenance summarization problem for queries with join, we need to define the score of a summary specifying what constitutes a good summary. Similar to the impact summaries in Section 4.3.2 we consider the coverage and the impact of the rules in the summaries. But additionally, we also take into account the relationship between the rules in the table summaries \( S_1 \) and \( S_2 \) to make sure they cover tuples that participate in the join operation.
The impact score of a summary $S$ for an answer $a$ to the join query is defined as $IScore^a(S) = \sum_{i \in \{1,2\}} IScore^a(S_i)$ where $IScore^a(S_i)$ is defined by Equations 4.1 and 4.2. To capture the relationship between the two table summaries $S_1$ and $S_2$, we define the impact of a tuple in Equation 4.2, i.e. $Impact^a(t)$, as follows. For $t_1 \in R^a_1$, the impact is defined by Equation 4.3 and considering the contingency set for $t_1$ if there is $t_2 \in R^a_2$ that is covered by $S_2$ and $t_1$ and $t_2$ satisfy the conditions in $Q$. Similarly, we define the impact of tuples in $R^a_2$.

Example 18. Consider $S_M = \{s_m\}$ and $S_D = \{s_d\}$ as sets of rules over Movies and Directors (Tables 4.8 and 4.7). $s_m = (\text{Title} : *, \text{Year} : *, \text{Genre} : *, \text{Rating} : 7, \text{Rev} : *, \text{Director} : *)$ is a rule that summarises movies with rating 7, i.e. $r_3, r_5$, and $s_d = (\text{Name} : *, \text{Gender} : \text{Male}, \text{Country} : \text{US})$ covers male directors from the US, i.e. $r_6$. The impact of $s_m$ is $r_3[\text{Rev}] + r_5[\text{Rev}] = 294.1$ and the score of $S_M$ is $Impact^a(s_m) \times weight^a(s_m) = 294.1 \times 1$. 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 the score of $S_D$ is $2 \times (275.3+219.4)$ since the weight of $s_d$ is 2.

4.3.4 The Comparative Summarization Problem

As motivated in Section 4.1, 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 4.2 and 4.1):

$$CScore^{a,a'}(S) = \sum_{s_i \in S} MPCount^{a,a'}(s_i, S) \times Weight^{a,a'}(s_i), \quad (4.4)$$
In $Cscore$ (the comparative score), $MPCount^{a,a'}(s_i, S)$ 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 $Weight^{a,a'}(s_i)$ 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 19.** In Example 11 the comparative score of $\{s_2\}$ for summarizing the similarities between the provenance of $a_4, a_6$ in $Q_2(IMDB)$ is $Cscore^{a_4,a_6}(\{s_2\}) = MPCount^{a_4,a_6}(s_2, \{s_2\}) \times Weight(s_2) = 0 \times 1$. The marginal pair count $MPCount$ is 0 because $s_2$ does not cover any pair from the provenance of $a_4, a_6$. For the set of rules $\{c_1\}$, the score is the following: $Cscore^{a_4,a_6}(\{c_1\}) = MPCount^{a_4,a_6}(c_1, \{c_1\}) \times Weight(c_1) = 1 \times 2$. $MPCount$ is 1 since $c_1$ covers a pair of tuples from the provenance of $a_4$ and $a_6$. The weight of $c_1$ is 2 because it has 2 non-$\star$ values.

**Definition 4.4** (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 $Cscore^{a,a'}(S)$.

**Theorem 2.** $Cscore^{a,a'}$ in Equation 4.4 is sub-modular.

Theorem 2 holds because every pair of tuples in $R^a, R^{a'}$ are counted once in $MPCount^{a,a'}$. As a result if $S \supseteq S'$ then $MPCount^{a,a'}(c_i, S) \leq MPCount^{a,a'}(c_i, S')$. The detailed proof of the theorem is in B.2. We present a greedy algorithm to find comparative summaries in Section 4.4. 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 4.4. We study extensions of comparative summaries considering the difference between the provenance tuples and with joins in B.3.
4.4 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 [JGP16] to summarize the provenance of queries using the new score functions in Section 4.3 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 [GMA13] (Line 1). Then it computes the impact of each tuple in $R^a$ on $Q(D)$, $I(t)$ is the function to calculate the impact of a tuple $t$ (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. $S := \emptyset$, $R^a := \text{Provenance}(D, R, Q, a)$
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. 
   5. $s := \text{BestMarginalRule}(R^a, a, S, I)$
   6. $S := S \cup \{s\}$
7. **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 $s_o = (A : \ast, B : b)$ 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 $(M(s_o)/\text{Weight}^d(s_o)) \times |R|$ in Line 8 is an upper bound for the marginal score of $s_n$, since the impact of $s_n$ is always less than $M(s_o)/\text{Weight}^d(s_o)$, which is the impact of its sub-rule $s_o$, and the weight of $s_n$ can never exceed $|R|$. 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 the 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 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
Algorithm 2: BestMarginalRule($R^a, a, S, I$)

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

1. for $j$ from 1 to $|R|$ 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.                 $U_n := \min(U_n, \frac{M(s_o)}{\text{Weight}^a(s_o)} \times |R|)$
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.                 $M(s_n) := M(s_n) + I(t) \times \text{Weight}^a(s_n)$
15.             $s := \arg\max_{s_n \in S_n} (M(s_n)), S_o := S_n$
16. return $s$

rule in $S$ (Line 7).

IPS differs the BRS algorithm (as detailed in [JGMP15]) since IPS generates provenance and considers the impact values while it computes the marginal scores. The BRS algorithm prunes candidate rules in a way that is 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.

4.4.1 Computing Impacts and Contingency Sets

So far, in our problem definition in Section 4.3 and the summarization algorithms earlier in this section, we considered the general class of aggregate
**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. $R^{a_1} := \text{Provenance}(D, R, Q, a_1); \quad R^{a_2} := \text{Provenance}(D, R, Q, a_2)$
3. for $i$ from 1 to $k$ do
4.   $c := \text{BestComparativeMRule}(R^{a_1}, R^{a_2}, a_1, a_2, S)$
5.   $S := S \cup \{c\}$
6. return $S$

**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. $S_n := \text{all possible rules that are not in } S$
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.       $M := M + \text{Weight}^{a_1,a_2}(c_n)$
8.     if $M_s < M$ then $c := c_n, M_s := M$
9. return $s$
SQL queries in Listing 4.1. 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. Second, we can compute the provenance summaries for these queries efficiently as we discuss in Section 4.4.2. Note that computing the impact of tuples on the general class of aggregate queries in Listing 4.1 is intractable since finding the contingency set for the provenance tuples is NP-hard in general [MGMS10]. In addition, computing the impact of the provenance tuples on a general query $Q$ with user-defined aggregate functions requires running, in the worst case, $Q^a(D \setminus D')$ for every subset $D'$ which can be costly. We can avoid this costly query execution for ASPJ queries as we 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 the 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 $|t[G^*]| / |R^a|$ for queries with AVG. In Example 13, the impact of the movie $t_1$ on the sum of revenues in $Q_2$ is the movie’s revenue, i.e. $t_1[Rev] = 275.3M\$. For an ASP query with MIN or MAX, computing the impact in Equation 4.3 requires finding the contingency set $C^a(t)$. Although computing the contingency set is proved to be NP-hard for general SQL queries [MGMS10], we can efficiently compute it for queries of the general form in Listing 4.1. 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 14, $a[MaxR] = t_2[Rating]$ and $C^a(t_2) = \{t_8\}$. The impact of $t$ is $|t[G^*] - m| / |C^a(t)| + 1$ in which $m$ is the second min or max value in $R^a$. In Example 14, 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 14. We note that computing the impact in $O(1)$ would require an additional data structure (e.g., a hash table) to find all rows with a particular value and assumes that there is some constant $k$ such that there are no more than $k$
rows with the min/max value.

For an ASPJ query of the general form in Listing 4.3, 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 \( \sum(|t'[G^*]|) \).

### 4.4.2 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 analyze 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 4.4.1. The cost of (c) is at most \( O(k \times n^2 \times 2^m) \) with \( m = |R| \) 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 [JGMP15]). 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 \( R^{a1}, R^{a2} \), which we omit, and the cost of finding the best marginal rules in Line 4 of Algorithm 4. That cost is \( n^2 \) since \textit{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 \( \alpha \) [NWF78].
4.4.3 A note about sampling

Sampling is the most common technique to optimize the performance of summarization algorithms. It is used in [JGP16] and [EGAG+14] to run the summarization algorithm on smaller samples instead of the full data set |D|. The simplest form is to draw a random sample every time the algorithm is run. The downside to random sampling is the loss of accuracy. [EGAG+14] find that their algorithm requires a very large sample size which is still prohibitive to real-time performance. The solution is a dynamic sampling solution with estimates on the drawn sample. For our purposes, for IPS, we require a sampling algorithm that estimates and maximizes the impact of rules in the sampled set. For CPS, we require a sampling method that maximizes coverage for the two sets (if looking for similarities) or differences between the two sets (if looking for differences).

The interactive sampling method in [JGP16] is more suited to their approach and the interaction method they employ. We propose a sampling method similar to the flashlight strategy described in [EGAG+14]. We start by running a CUBE query that takes in the top rules for impact. We filter out any low-impact rules and their associated tuples based on a threshold value set by the user. This gives us a decent estimate of the real impact values. We run our algorithm on this sample. We use the same strategy for drill-down operations and obtaining super-rules. We use this method for AVG and SUM, summaries for MAX and MIN is very fast operations without the need for sampling.

As we discussed in Section 4.4.2, CPS is more costly compared to IPS since it compares pairs of tuples. Therefore, we implemented a similar sampling optimization to IPS for CPS that prunes the rules with low coverage on both provenance sets using a coverage threshold.

Since we do not have access to the sampling method used in [JGP16], we perform our experiments without sampling to show how the two methods compare in terms of raw un-optimized performance.
Furthermore, [FGS17] shows that random samples at certain sizes only degrade the quality of the summary by a small margin. The gains in performance are worth the small loss in accuracy. We employ both impact and random sampling as possible optimization options.

4.5 User Interface and visualization

The user interface and visualizations provide users with all the necessary facilities 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 for 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 4.1 and Equations 4.4, 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 4.2 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.
Figure 4.1: User interface for impact summaries.

Figure 4.2: User interface for summaries for queries with join (NA refers to missing data).
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, e.g. in Figure 4.3 the first rule represents a large impact of the provenance of Action as it does in the provenance of Drama.

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 it (see Section 4.4 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, e.g. the attribute Title in Figure 4.2. These attributes can be collapsed into a single attribute named $\star$ as seen in Figures 4.2.

<table>
<thead>
<tr>
<th>Year</th>
<th>Genre</th>
<th>Rating</th>
<th>Language</th>
<th>Country</th>
<th>Coverage b1</th>
<th>Coverage b2</th>
</tr>
</thead>
<tbody>
<tr>
<td>2016</td>
<td>Action/DRama</td>
<td>$\star$</td>
<td>en</td>
<td>$\star$</td>
<td>0.94</td>
<td>0.88</td>
</tr>
<tr>
<td>2016</td>
<td>Action/DRama</td>
<td>$\star$</td>
<td>en</td>
<td>US</td>
<td>0.66</td>
<td>0.6</td>
</tr>
<tr>
<td>2016</td>
<td>Action/DRama</td>
<td>$\star$</td>
<td>en</td>
<td>US</td>
<td>0.63</td>
<td>0.5</td>
</tr>
<tr>
<td>2016</td>
<td>Action/DRama</td>
<td>6</td>
<td>en</td>
<td>$\star$</td>
<td>0.21</td>
<td>0.08</td>
</tr>
<tr>
<td>2016</td>
<td>Action/DRama</td>
<td>6</td>
<td>$\star$</td>
<td>$\star$</td>
<td>0.21</td>
<td>0.08</td>
</tr>
<tr>
<td>2016</td>
<td>Action/DRama</td>
<td>$\star$</td>
<td>en</td>
<td>UK</td>
<td>0.09</td>
<td>0.12</td>
</tr>
<tr>
<td>2016</td>
<td>Action/DRama</td>
<td>$\star$</td>
<td>en</td>
<td>UK</td>
<td>0.09</td>
<td>0.12</td>
</tr>
<tr>
<td>2016</td>
<td>Action/DRama</td>
<td>$\star$</td>
<td>en</td>
<td>Germany</td>
<td>0.05</td>
<td>0.01</td>
</tr>
</tbody>
</table>

Figure 4.3: User interface for comparative summaries.


4.6 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 4.6.2), (b) to compare provenance summaries with basic summaries and show provenance summaries have higher quality w.r.t. our new metrics (Section 4.6.3), (c) to show the relevance of impact and other related metrics such as diversity for provenance summarization using a user survey (Section 4.6.4).

4.6.1 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 selection of answers was done by a uniform random sampler.

The results of our summarization algorithms also depend on the query. We ran our experiments for different queries. To analyze 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 a significant difference between the results. We used the labels \textit{IAGG} and \textit{CAGG} to refer to the IPS and CPS with AGG as the aggregate function, e.g. \textit{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 \cite{GMA13}.

\textbf{Datasets} We used three datasets. For the GLEI, IMDB, and TPC-H datasets, the data characteristics are as follows: The number of tuples ($|D|$): 250k, 181k + 323k, and 60k-3m. The number of tables ($N$): 1, 2, and 8. The total number of attributes in the tables ($m_D$): 12, 15, and 61. The number of
queries \((q)\): 5, 6, 5.

**Global Legal Entity Identifier (GLEIs).** This is a real world financial dataset collected from various sources for financial institutions\(^\text{[CM19]}\) 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\(^\text{4.6.2}\)). In those experiments, we ran the following query over a single table, GLEI, with a varying number of attributes \(G_1, G_2, \ldots\) and aggregate functions \(f\):

\[
\text{SELECT } G_1, G_2, \ldots, f(\text{Dist}) \text{ FROM GLEI GROUP BY } G_1, G_2, \ldots
\]

\(\text{Dist}\) 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 a different number of attributes and different aggregate functions:

\[
\text{SELECT } G_1, G_2, \ldots, f(\text{Rev}) \text{ FROM Movies m, Directors d WHERE } m.\text{Director} = d.\text{Name} \text{ GROUP BY } G_1, G_2, \ldots
\]

**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

\(^2\text{http://www.tcp.org/lspec.html}\)
8, which we decided based on survey results in Section 4.6.4. 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 that users unfamiliar with a dataset would start their analysis with a single group-by attribute query.

### 4.6.2 Runtime Performance

To evaluate the 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.

**Exp-1: Effects of \(|D|\).** Figure 4.4a 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 depends on the query.

**Exp-2: Effects of \(k\).** Figure 4.4b 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.

**Exp-3: Effects of \(m_q\) and \(m_w\).** According to Figure 4.4c the runtime of IPS greatly decreases as the number of group-by attributes \(m_q\) increases.
the GLEI experiments, the average size of a provenance set is $9k$ tuples, $4.5k$ 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 4.4e shows the runtime of IPS for all queries. IPS performs similarly for all queries when the data size is small. The join queries slightly outperform those without join because the summaries for queries with join have a smaller number of attributes, and the shared join attributes appear only once in the join result. However, we see a larger effect of join for larger data sizes because finding contingency sets is costly.

**Exp-5: Effect of sampling on performance.** We evaluate the effect of sampling on performance by comparing various sample sizes with running the various algorithms without sampling. Figure 4.7a shows the results. We see a linear effect of sampling that is very pronounced. We study the effect of sampling on the quality of the rules in experiment 8.

We also evaluated the effect sampling in Figure 4.4f 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.

### 4.6.3 Quality of Provenance Summaries

Measuring the quality of a summary of this type is a challenging task. We chose four objective metrics to evaluate summaries produced using the BRS algorithm and IPS. We also later describe our user survey to study users’ expectations for such summaries to show the relevance of our metrics (Sec-
We define the following quality metrics for a summary:

- **Impact** is the distribution of impact values for rules in the summary. The sum and average impact for a summary do not give the full picture. A good impact summary needs to have all high-impact rules.

- **Coverage** is the total number of provenance tuples covered by the rules in the summary.

- **Weight** is the total weight (the number of non-\(\ast\) values) of the rules in a summary. A higher weight represents more descriptive summaries.

- **Diversity** is the number of unique values for attributes present in the summary. Diversity is used in [WZRY18] 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, and high diversity, and its rules would have higher weights.

**Exp-6: Coverage and impact.** Figures 4.5a and 4.5b respectively show the coverage and the impact of the rules generated by different algorithms for the IMDB dataset. Figures 4.6a and 4.6b show the values for the TPC-H queries. Figures 4.5a and 4.6a show that BRS generates rules with higher coverage, which was expected. Surprisingly, Figures 4.5b and 4.6b reveal 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. However, impact as a quality metric cannot be analyzed in isolation, we need to also look at weight. This is because the basic summaries in BRS achieve high impact with a few high-coverage rules that have low weight and are thus not very informative. The rest of the rules in a basic summary do not have a high impact. However, the
impact summaries, e.g. ISUM and IAVG, gain high impact with informative rules with high weights. This confirms that IPS finds rules with high impact that are also informative. For IMAX and IMIN, due to the distribution of the datasets in our experiments, IPS finds summaries that only contain a few (often one) rules with high impact and weight but low coverage and diversity.

For TPC-H, impact and coverage values for some queries are not very informative. Q1 and Q10 specifically always produce similar rules and thus the values for all variables do not change. This is due to the provenance of the query results being summarized which all contain the same categorical values. In the other queries we see a similar trend to IMDB experiments where coverage and impact are close.

**Exp-7: Diversity and weight.** Figure 4.5c 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 the weight. Regarding diversity, IPS outperforms the BRS algorithm (Figure 4.5d). With higher diversity values, we can say that the higher weight rules of IPS are also more informative. Figures 4.6d and 4.6c show the diversity and weight values for all TPC-H queries with similar results. We see higher weight and diversity for ISUM and IAVG for q3, q5, and q6. With similar coverage and impact, they produce more meaningful rules. q1 and q10, similar to the previous experiment, do not show any meaningful values.

**Exp-8: Effect of sampling on quality.** The plots in Figure 4.7 show the effect of sampling on all metrics: coverage, impact, weight, and diversity across datasets. We see a very small effect of sampling on weight and diversity as the summary rules for the smallest samples have similar weight and diversity values, we omit those figures to save space. Impact and coverage see slight changes with the larger sample sizes getting close to the values obtained with unsampled runs. This is because the largest values are present in all samples as those values dominate the rules being picked, and as a result,
the patterns are very similar across datasets. Those attribute values include binary values (male, female), as well as highly dominant values (country = "United States" in IMDB and GLEI). A good future direction could be to try to increase the diversity of rules by pruning out such values. Figures 4.7b and 4.7c show the distribution of different coverage and impact values from running multiple queries.

4.6.4 User Survey

We conducted an online survey to ask computer science students with entry-level knowledge of data and databases about their expectations of a data summarization system. We designed our questionnaire to be generic, presenting a use-case scenario that general users, with database entry-level course knowledge, would be familiar with. We tried to minimize bias by giving a very generic example of the IMDB dataset. We did not mention provenance or provenance summarization, as those are specialized topics that the typical computer science student is unlikely to be familiar with. Instead, we refer to the summary as a summary of tabular data. We also used neutral terms like “group of movies with high revenue” or “groups that contain a lot of movies”, and tried to avoid using terms like “coverage” and “impact”. We also asked users for qualitative feedback in the form of justification for their choices. We present the full survey materials in Appendix B.

We presented participants with a single 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. The scenario is as follows:

---

**Summarization scenario.**

A user accesses a dataset about movies and directors. The user asks the

---

3The study was approved as minimal risk by the Behavioural Research Ethics Board at the University of British Columbia. UBC BREB number: H20-01634.
system for the total revenue for action movies made in the year 2017. The user gets the result back: {Action; 2017; 7.3 Billion.} Now, given this result we would like to provide the user with a summary of movies made in 2017 that made this 7.3 billion figure. Here is an example of a summary that is of length 3. It contains 3 rows:

<table>
<thead>
<tr>
<th>Genre</th>
<th>Year</th>
<th>Country</th>
<th>Duration</th>
<th>Language</th>
<th>coverage</th>
<th>impact</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>2017</td>
<td>US</td>
<td>*</td>
<td>English</td>
<td>35%</td>
<td>85%</td>
</tr>
<tr>
<td>Action</td>
<td>2017</td>
<td>*</td>
<td>&gt; 120</td>
<td>*</td>
<td>25%</td>
<td>78%</td>
</tr>
<tr>
<td>Action</td>
<td>2015</td>
<td>*</td>
<td>&lt; 90</td>
<td>*</td>
<td>39%</td>
<td>21%</td>
</tr>
</tbody>
</table>

Table 4.9: Summary A

This summary is presented with real values for their respective columns and * value as do-not-care values filling in for any other value in the column. For example, row 1 comprises action movies from 2017 made in the US with all possible duration values in English. Those movies make up 35% of rows in the original database and have 85% of total revenue.

We asked users to evaluate this summary and then asked them about their thoughts on 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, in terms of the number of rules, would be appropriate and asked them to justify their choices. We used qualitative answers given by users throughout the survey to filter out any survey responses with low-quality responses. We were able to remove 2 responses where the qualitative answers were not understandable by the researchers. We also did not record any survey responses from participants who did not complete the full survey. Furthermore, we used qualitative responses as evidence of the quality metrics users prefer.
27 participants completed the survey. Results show participants favor high-impact rules over rules with high coverage. They also favor surprising rules the most (Figure 4.8a). In their evaluation of the summaries, most users picked the high impact, low coverage rules, and high weight rules 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, and users were given multiple size options above 12. 1 user picked other and wrote down “depends on the data”. Full results can be seen in Figure 4.8b.

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”. “I feel that in 5-8 rows you could cover all the important groups of action movies that are of interest”.

We acknowledge the limitations of a survey of this type. We took some measures to control the quality of responses: responses were inspected manually before analysis. Qualitative responses were inspected to make sure users understood the questions and provided relevant answers. We also asked for self-reporting of accuracy from users. We are confident this is a first step in evaluating the usefulness of our summaries and setting up future work.

4.6.5 Case Study

We present three examples to further explain how the provenance summaries are different from basic summaries and summarise provenance data more effectively.

Impact Summaries. We look at the quality of an impact summary for the provenance of Action movies from the following query:
SELECT Genre, SUM(Rev) FROM Movies GROUP BY Genre

We ran the query over Movies in the IMDB dataset with additional attributes, such as language and duration, compared in Table 2.2. Tables 4.10 and 4.11 subsequently show the impact summaries and basic summaries of size $k = 4$ generated by IPS and the BRS algorithm. As we explained in Exp-6, impact summaries usually have rules with higher weights compared to basic summaries because of using the larger impact values, that are multiplied by weight to compute the score of a rule. In this case, the impact summary includes $s_3$ with higher weight than $s_7$ despite $s_7$ having higher impact and coverage. This is because $s_3$’s marginal score, which is a function of impact and weight, is higher than $s_7$’s marginal score. Regarding coverage and impact, we see that basic summaries include rules that cover more tuples but do not have high impact, e.g. $s_6$ has high coverage 0.5 and low impact 0.03. This is unlike impact summaries that prioritize impact over coverage, e.g. $s_2$ in our impact summary has low coverage, 0.08, and high impact, 0.52. This is an informative rule that covers movies with a duration of 150 minutes; such movies account for a lot of recent blockbusters with huge revenue values such as *Star Wars: The Force Awakens*, *Jurassic World*, *Furious 7*, and *Avengers: Age of Ultron*. Such rules with high impact, high weight, and low coverage can provide the surprising information that users are looking for.

**Impact Summaries for Queries with Joins.** We look at the impact summaries for the following query:

```sql
SELECT m.Genre, AVG(m.Rev) FROM Movies m JOIN Directors d ON m.director=d.name GROUP BY m.Genre
```

We show how an impact summary containing separate rules over the movies and directors tables can provide new information compared to those over the join result, *MoviesDirectors*. We assume that the user asks for an impact summary for the answer containing action movies. In an impact summary
Table 4.10: An impact summary from the IPS algorithm

<table>
<thead>
<tr>
<th>ID</th>
<th>Genre</th>
<th>Lang</th>
<th>Country</th>
<th>Duration</th>
<th>Weight</th>
<th>Coverage</th>
<th>Impact</th>
</tr>
</thead>
<tbody>
<tr>
<td>s₁</td>
<td>Action</td>
<td>EN</td>
<td>US</td>
<td>⋆</td>
<td>2</td>
<td>0.45</td>
<td>0.63</td>
</tr>
<tr>
<td>s₂</td>
<td>Action</td>
<td>EN</td>
<td>⋆</td>
<td>150 min</td>
<td>2</td>
<td>0.08</td>
<td>0.52</td>
</tr>
<tr>
<td>s₃</td>
<td>Action</td>
<td>EN</td>
<td>US</td>
<td>120 min</td>
<td>3</td>
<td>0.22</td>
<td>0.29</td>
</tr>
<tr>
<td>s₄</td>
<td>Action</td>
<td>EN</td>
<td>⋆ ⋆</td>
<td>1</td>
<td>1</td>
<td>0.76</td>
<td>0.97</td>
</tr>
</tbody>
</table>

Table 4.11: A basic summary from the BRS algorithm

<table>
<thead>
<tr>
<th>ID</th>
<th>Genre</th>
<th>Lang</th>
<th>Country</th>
<th>Duration</th>
<th>Weight</th>
<th>Coverage</th>
<th>Impact</th>
</tr>
</thead>
<tbody>
<tr>
<td>s₅</td>
<td>Action</td>
<td>EN</td>
<td>US</td>
<td>⋆</td>
<td>2</td>
<td>0.45</td>
<td>0.63</td>
</tr>
<tr>
<td>s₆</td>
<td>Action</td>
<td>EN</td>
<td>⋆</td>
<td>90 min</td>
<td>2</td>
<td>0.5</td>
<td>0.03</td>
</tr>
<tr>
<td>s₇</td>
<td>Action</td>
<td>EN</td>
<td>⋆</td>
<td>120 min</td>
<td>2</td>
<td>0.37</td>
<td>0.42</td>
</tr>
<tr>
<td>s₈</td>
<td>Action</td>
<td>EN</td>
<td>⋆ ⋆</td>
<td>1</td>
<td>1</td>
<td>0.76</td>
<td>0.97</td>
</tr>
</tbody>
</table>

Comparative Summaries. We also look at an example to show the value of our comparative summaries and compare them with basic summaries. We consider two answers $t = (\text{Romance}, 167)$ and $t' = (\text{Foreign}, 39)$ to the following query:

```sql
SELECT Genre, COUNT(*) FROM Movies
WHERE Language = EN AND ReleaseYear = 2010
GROUP BY Genre
```

We compare two sets of rules: (a) a comparative summary obtained from $CPS$ and for the provenance of $t$ and $t'$, and (b) a basic summary obtained over $Movies\!Directors$, we do not see any insightful rules, and all rules revolve around common categorical attributes such as country, language, and director gender. However, looking at the individual contribution of each table in an impact summary over $Movies$ and $Directors$, we see an informative rule with ($Gender : \text{Female}$ and $Name : \text{Patty Jenkins}$) in the impact summary over $Directors$. This is because $Patty Jenkins$ made $Wonder Woman$, a movie that made a lot of money and has a substantial impact on the average revenue for female directors.
by running the BRS algorithm on $Movies^t \cup Movies^{t'}$.

In (a), we see interesting rules with values common in romance and foreign movies. For example, a common class of movies are those made in English speaking countries other than the US, e.g. the Philippines. The rules in (b) are dominated by the bigger provenance set, i.e. Romance movies, and always show the most occurring values in that set, e.g. movies that are made in the US, or with a runtime of 120 minutes. Our CPS algorithm shows that movies made in foreign countries such as the Philippines had markedly different characteristics from “standard” movies. The user may have wanted to make sure that such movies were only counted in one genre, i.e. foreign movies, instead of both. This type of analysis would not have been possible with the basic summaries produced by the BRS algorithm.

The basic summary is always dominated by the bigger set (Romance movies) and always shows the most occurring values in that set, such as: (Country: US), (Rating: 0.0), and (Runtime: 120). In the comparison situation in Section 4.3 the user is likely to be interested in exploring which answers are anomalous. For example, our CPS algorithm shows that movies made in foreign countries such as the Philippines had markedly different characteristics from “standard” movies. The user may have wanted to make sure that such movies were handled differently. This type of analysis would have been impossible with the BRS algorithm.

While this analysis is not conclusive, it clearly shows that different summaries cover different user needs depending on the use case. The basic summaries work best for coverage, while the impact summaries highlight high-impact tuples and the comparative summaries are more effective in showing overlooked interesting areas or tuples. As users seem to seek more informative and interesting summaries, a summarization system needs to be flexible to cover all users’ needs.
4.7 Conclusion and Future Work

In this chapter, we implemented new summarization techniques: impact summaries and comparative summaries. Our summaries work well for users without knowledge of provenance semantics or the data. We validated our techniques with thorough experiments and a user survey.

There are many viable avenues for future work. One direction is to support a bigger class of queries, such as nested queries. Another future work item is extending comparative summaries to more than two tuples. We could also explore other semantics of contribution, such as the Shapley value [Sha97]. Overall, there is still a lot to be done in this field. However, we believe impact and comparative summaries are a step in the right direction. We utilize those summaries in Chapter 5 to develop new techniques for query refinement. Such techniques aim to show that summaries are useful for more applications beyond improving the usability of provenance information and data exploration.
Figure 4.4: Effect of different parameters on Runtime, in seconds, of IPS. CPS experiments used the IMDB dataset.
<table>
<thead>
<tr>
<th>Coverage</th>
<th>Impact</th>
<th>Weight</th>
<th>Diversity</th>
</tr>
</thead>
<tbody>
<tr>
<td>BRS</td>
<td>ISUM</td>
<td>IAVG</td>
<td></td>
</tr>
<tr>
<td>k</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 4.5: The quality of summaries w.r.t. coverage, weight, impact, and diversity for the IMDB data, and the impact of $k$ and $m_w$ (the maximum weight) on these quality metrics.
Figure 4.6: The quality of summaries for five queries in the TPC-H dataset ($k = 4$ and $m_w = 3$).
Figure 4.7: a shows sampling performance. b & c show the effect of sampling on the quality of summaries. ($k = 4$, $m_w = 3$)
Figure 4.8: Survey results show how users ranked the various metrics (5 essential – 1 not important) and summary sizes.
Chapter 5

Aggregate query refinement using summary rules and provenance summaries

5.1 Introduction

Users perform aggregation queries for many purposes, including exploring a dataset, deriving numeric measures, and generating reports. Our focus is on the scenario of data exploration, where the user has limited knowledge and expertise in dealing with the dataset. The data exploration process for such a user is iterative and error-prone. The user starts the process by looking to form a picture of what the data looks like. This is usually done by asking aggregate queries. When users receive an aggregate answer they ask: “What is included in the result? What is not included?” While database provenance can provide an answer to the first question, it is not sufficient to answer the second one and is often verbose and complex enough to require further expertise. One variation of this problem is if the user knows the domain well enough to know that the answer is wrong. In this case, the user may want to modify the aggregate query to receive a specific answer.
We tackle this instance of the problem with a solution that shows the user suggestions to help the user find the specific answer they are looking for. Our solution addresses two main problems: relaxing and contracting queries. Intuitively, relaxing a query means expanding the query answer to include more tuples. Contracting a query means shrinking the query answer to include fewer tuples. We define the problem more formally later in this section.

There currently exist query refinement solutions such as [MK09], that can solve the relax/contract query problem. Current solutions focus mainly on cardinality constraints and the query itself. Those solutions are very query oriented, with minimal access to the database or underlying data. They also rely heavily on the existence of hierarchies for categorical variables. In terms of interaction, these solutions rely on the user to interact with the data to manually refine the query and reach a close enough answer. We argue that such solutions are not sufficient for the scenario we describe above. In our scenario, the user knows the answer is wrong but lacks the expertise to manually refine the query. Therefore, our solution is fully automated: the user submits a query and their feedback and we provide refinement suggestions. Our solution also helps users navigate the space of categorical variables without the need for input hierarchies, which may not be available. Example 20 shows in detail how this works.

Some promising related areas of research are in answer explanation [AKS+18], tuple recommendation [DP13], and query by example [FM19]. Such solutions access the data and use different utilities to help users in different applications. We use techniques similar to those used for query refinement. Accessing the data and finding interesting patterns to show the user can help them understand their query and query answers, and can lead to a fuller understanding of the refinement process. We differentiate our work from each of those domains in the related work section (Section 5.2).

We formally define the problem as follows:
Definition 5.1 (Aggregate query refinement problem). A user asks an aggregate query, \( Q \), to a database, \( D \), where the answer, \( Q(D) \), is equal to some single value, \( k \). Additionally, the user has an expected target value, \( k' \).

If \( k \neq k' \), the aggregate query refinement problem is to help the user find a query, \( Q'' \), where the answer to \( Q'' \), \( Q''(D) \) is as close to \( k' \) as possible.

In other words, the goal of aggregate query refinement is to help the user to recover the hidden query \( Q' \) where \( Q'(D) = k' \); we approximate this by finding \( Q'' \), which ideally is equal to \( Q' \).

In this chapter, we propose a framework that uses the data itself, without using query logs, hierarchies about the data, or user session information, as those are rarely available. Our framework uses provenance summaries of answers and non-answers as well as summary statistics to get the user as close to the desired answer as possible. We show that our framework is useful and produces meaningful recommendations through a user study. We also conduct thorough experiments to show that our system can work in real-time and is scalable.

We go into more detail on our framework in Section 5.4.

<table>
<thead>
<tr>
<th>ID</th>
<th>Title</th>
<th>Year</th>
<th>Genre(s)</th>
<th>Rating</th>
<th>Revenue</th>
<th>Runtime</th>
<th>Production Company</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Frozen</td>
<td>2013</td>
<td>Animation, Adventure</td>
<td>7</td>
<td>1.2B</td>
<td>102</td>
<td>Walt Disney Pictures</td>
</tr>
<tr>
<td>2</td>
<td>Monsters, Inc.</td>
<td>2001</td>
<td>Animation, Adventure</td>
<td>7</td>
<td>560M</td>
<td>96</td>
<td>Walt Disney Pictures</td>
</tr>
<tr>
<td>3</td>
<td>Aladdin</td>
<td>1992</td>
<td>Musical, Adventure</td>
<td>7</td>
<td>504M</td>
<td>90</td>
<td>Walt Disney Productions</td>
</tr>
<tr>
<td>4</td>
<td>Toy Story</td>
<td>1995</td>
<td>Comedy, Adventure</td>
<td>7</td>
<td>373M</td>
<td>89</td>
<td>Walt Disney Pictures</td>
</tr>
<tr>
<td>5</td>
<td>Toy Story 2</td>
<td>1999</td>
<td>Adventure, Family</td>
<td>7</td>
<td>497M</td>
<td>95</td>
<td>Walt Disney Pictures</td>
</tr>
<tr>
<td>6</td>
<td>Finding Nemo</td>
<td>2003</td>
<td>Family, Adventure</td>
<td>7</td>
<td>940M</td>
<td>100</td>
<td>Pixar Animation Studios</td>
</tr>
<tr>
<td>7</td>
<td>Up</td>
<td>2009</td>
<td>Adventure, Family, Animation</td>
<td>7</td>
<td>735M</td>
<td>96</td>
<td>Pixar Animation Studios</td>
</tr>
<tr>
<td>8</td>
<td>The Lion King</td>
<td>1994</td>
<td>Musical, Animation</td>
<td>7</td>
<td>788M</td>
<td>89</td>
<td>Walt Disney Productions</td>
</tr>
<tr>
<td>10</td>
<td>Guardians of the Galaxy</td>
<td>2014</td>
<td>Action, Sci-fi</td>
<td>7</td>
<td>773M</td>
<td>122</td>
<td>Marvel Studios</td>
</tr>
<tr>
<td>11</td>
<td>Dr. Strange</td>
<td>2016</td>
<td>Action, Adventure</td>
<td>7</td>
<td>677M</td>
<td>115</td>
<td>Marvel Studios</td>
</tr>
<tr>
<td>12</td>
<td>Rogue One: A Star Wars Story</td>
<td>2016</td>
<td>Action, Sci-fi</td>
<td>7</td>
<td>1.05B</td>
<td>133</td>
<td>Lucasfilm</td>
</tr>
</tbody>
</table>

Table 5.1: MoviesGenre sample table

Example 20. A user starts exploring a dataset of movies with information about movies and their respective genres (Table 5.1). For simplicity, we show
Table 5.1 as a joined table with the genre attribute denormalized. The user is looking for a set of highly rated, high-grossing, adventure movies made by Disney. The user tries to define the criteria for this task. Based on those criteria, the user asks a query $Q$ to get all movies in the Adventure genre, that have revenue greater than 300 million, and a user rating greater than 7. Finally, the query specifies that the movies are made by Walt Disney Studios or any variation of the studio name. $Q_1$ shows the SQL query:

$$Q_1 : \text{SELECT COUNT}(\ast) \text{ FROM movie m, genre g}$$

WHERE m.id = g.movie_id
AND m.revenue > 300m
AND m.rating > 7
AND g.genre = 'Adventure'
AND m.production_company LIKE 'Walt Disney%'

This query can be expressed as a summarization rule in the form:

$(\text{Title} : \ast, \text{Year} : \ast, \text{Genre} : \text{Adventure}, \text{Rating} : > 7, \text{Production company} : \text{Walt Disney\%}, \text{count} : 5, \text{Rev} : > 300m)$

The rule contains $\ast$ which is a wildcard variable that matches all values for an attribute. Other attributes contain values that match those specified in the query conditions. Summarization rules are explained in depth in Chapter 2.

There is a total of 5 movies ($k$) in Table 5.1 that match the criteria. The user expects a larger set of answers, so they examine the provenance of the result ($t_1 - t_5$ from Table 5.1). The user is surprised to see that some movies they know should be included are missing. So they submit feedback that the result size is low and provide an expected number of results, $k' > 5$.

The system goes through the table looking for groups of tuples in the data that meet the new cardinality constraint and meet the conditions in the query as much as possible. The system provides several suggestions, in the form of summarization rules that summarize such groups. The rules provide relaxation suggestions that contain the original query results and meet or get close to the cardinality constraints set by the user. In this case, it will return
the rules:

1. \( s_1 = \{ r_1 : \{ \text{Title} : *, \text{Year} : *, \text{Genre} : *, \text{Rating} : > 7, \text{Production company} : \text{WaltDisney\%}, \text{count} : 6, \text{Rev} : > 300 m \} \} \)

2. \( s_2 = \{ r_1 : \{ \text{Title} : *, \text{Year} : *, \text{Genre} : \text{Adventure}, \text{Rating} : > 7, \text{Production company} : \text{WaltDisney\%}, \text{count} : 5, \text{Rev} : > 300 m \}, r_2 : \{ \text{language} : \text{en}, \text{Title} : *, \text{Year} : *, \text{Genre} : \text{Adventure}, \text{Rating} : > 7, \text{Production company} : \text{PixarAnimationStudios}, \text{count} : 2, \text{Rev} : > 300 m \} \} \)

The first suggestion \( s_1 \) relaxes the genre attribute specified in the query, to include \( t_8 \) The Lion King in the results. \( k'' \) here is equal to 6. The second suggestion \( s_2 \) is a conjunction of 2 rules that adds movies made by Pixar Animation Studios to the results and increases the result size to \( k'' = 7 \). Both suggestions are made because the underlying tuples share common properties with those in the provenance set \( (t_1 - t_5) \). \( t_8 \) is made around the same time and has a similar rating and runtime. \( t_6 \) and \( t_7 \) are also made around the same time, with similar ratings, revenue, and runtime. The other movies are much longer, contain different genres (Action, Sci-Fi), and are more recent, thus they are deemed dissimilar to those in \( t_1 - t_5 \). It is worth noting that Disney bought Pixar in 2007 so the two are closely related. The example shows our system can detect similarities without augmenting it with more data, using simple descriptive attributes such as genre and duration.

Without our system, the user would have to manually explore the data. This would involve rewriting the query, relaxing, or removing conditions. The user would also have to inspect the data manually to see what is contained in the results. This is a costly and not necessarily intuitive process. Furthermore, without hierarchies or order for the categorical attributes, the user would have no idea where to go without knowledge of the attribute values. Moving from Action to Adventure or from a production company to
another that is related can be an ambiguous process without the aid of our solution.

With our approach, we define the problem of query relaxation/refinement under a user-provided cardinality constraint as finding a summary (candidate query) from the space of candidate summaries mined from the data (which are also candidate queries) that fulfill the cardinality constraint and is most similar to the user’s original query in terms of results. Figure 5.1 gives an overview of the solution.

As seen in Figure 5.2 which shows the details of our query refinement framework architecture, our system performs the following steps:

1. Reduce the number of attributes using correlation coefficients.

2. Produce rules using a CUBE query over all leftover attributes. Store these rules in a rules database. Both of these steps are performed as initial offline steps.

3. When the user submits a complaint, generate a list of candidate rules from the rule table using cardinality constraints and rule containment. This step prunes a large portion of non-answers.

4. Generate a summary as a set of rules for the results of \( Q(D) \).

5. Compare the summary of the provenance of answer \( Q(D) \) against the candidate rules of non-answers. We encode rulesets as TFIDF vectors that are then compared using a similarity metric. The end result is a ranked list of the most recommended non-answers that can fulfill the users’ cardinality constraint.

We explore the techniques we use further in Section 5.4.

5.1.1 Contributions

In summary, our contributions for this chapter are as follows:
Figure 5.1: Overview of solution from the query step all the way to the recommended groups of tuples
Figure 5.2: Query refinement framework architecture
1. We develop a new method for query refinement for the scenario where the user does not like their initial query result and wants to see a different answer (Section 5.4).

2. We conduct a user study to show that our method provides useful refinement suggestions to users (Section 5.5).

3. We perform performance experiments to show that our method is scalable and outperforms automated machine learning methods (Section 5.6).

5.2 Related work

Reverse engineering queries. Such works as [TCP14, TZES17] tackle the query reverse engineering problem or QRE. The problem is defined as finding a query, \( Q \), from a set of results that has an output equal to the results. This problem is relevant but not directly equivalent to the problem of refining query results. It is relevant because it takes a set of tuples where the query is unknown and produces a query. It is not equivalent because this set is assumed to be a complete result. Systems such as TALOS [TCP14] can provide explanations for missing answers and provide suggestions for refined queries. However, they require the user to provide a full set of such missing answers to get a new query, \( Q' \), that explains why the answers are missing. While this is indeed similar functionality, the requirement does not work for our scenario where users are not very familiar with the data and cannot provide the full set of results but a partial set at best.

Query by example. [TCP09] An alternative method to declarative query languages is query by example or QBE. Here the user must provide a subset of the answer to receive suggestions on queries that produce results that include the answer. While most commercial DBMSs provide such functionality, the results are often lacking or simplistic. More modern solutions such as [FM19]
aim to find the users’ intentions and produce more intuitive queries. While the description of QBE systems and query refinement systems sound similar, they are not quite the same. One element that is lacking is that QBE systems take in a small set of tuples that contain very limited attribute values. Large inputs would need to be filled in manually by the user and can be taxing. Whereas query refinement systems take in a query and full results. QBE systems also rely on the user having some sort of intuition on what answer they would like to see, which assumes some knowledge of the data.

**Interactive data exploration.** Such systems work on the idea of providing users with suggestions of individual tuples [SDP09, DP13] or interesting areas or groups of tuples [DPD16, CCD+13] to get them where they need to go as they explore a dataset. Such work is relevant but does not take in any constraints such as cardinality or relevance to a full set of results. Furthermore, the burden is often on the user to provide feedback on what they see as relevant. This user feedback can help build models that refine their suggestions the more they interact with the system. This line of work is directly relevant, but not specific enough for query refinement tasks.

**OLAP.** Papers in the online analytical processing space, such as [SAM98] provide ideas for automatic exploration of data. However, the techniques are better suited for the data warehousing context where data is cleaned, curated, and contains hierarchies and functional dependencies that are explicitly defined. The scalability of such approaches is also in question when applied to larger datasets with more dimensions.

**Missing answers.** [CJ09, HH10a, HCDN08] present the problem of explaining missing answers. This is a directly related topic to relaxing queries. Where query relaxation has the goal of finding tuples closely related to the query answer, explanations for a missing answer try to find the most probable explanations for why a tuple is missing from an answer set. The explanations are typically in the form of conditions to be fulfilled to add the missing tuple to the result. While this work contains a lot of useful techniques and
is closely related, it does not address the problem of having multiple tuples or a large result set and finding related query refinement candidates.

**Query refinement.** There have been multiple systems in the query refinement space that handle the same problem we defined in the introduction [MK08, MK09, VRRM16, ASZ14, BHT16], those systems perform very similar tasks to ours. All systems, however, focus mostly on the query and provide query-centric solutions. They make assumptions about the presence of hierarchies for categorical values. They also offer different interfaces for users to interactively refine their queries. Our solution makes different assumptions and provides fully automated suggestions that do not require user feedback. Our solution is also data-centric. More specifically, we propose an approach that uses the data without analyzing queries. We show the user a summary (candidate query) from the space of candidate queries mined from the data (which are summaries or sets of summaries) that fulfills the cardinality constraint and is similar to the user’s original query. Whereas our work is focused mostly on categorical variables (due to the nature of the rules we use), the other related works we cite have made tremendous progress in dealing with continuous attributes.

### 5.3 Problem Setting

In this section, we formalize the problem of query refinement and define the parameters for every part of it. We use slightly different symbols than those defined in Chapter 2 as shown in Table 5.2. We define the class of aggregate functions we handle to be Count, Sum, Min, and Max. We also define the problem in terms of attribute types: numeric or categorical. We use data-independent containment to find rules that cover query results. We also use bucketizing for numeric data. We discuss all of these concepts in detail in this section.
Table 5.2: Summary of symbols and notations for query refinement

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(R, \mathcal{R})</td>
<td>relation and relational schema</td>
</tr>
<tr>
<td>(\mathcal{R}_M)</td>
<td>Set of relations in database instance (D)</td>
</tr>
<tr>
<td>(A_1, A_2, \ldots, A_n)</td>
<td>relational attributes</td>
</tr>
<tr>
<td>(t)</td>
<td>tuples</td>
</tr>
<tr>
<td>(Q, Q(D))</td>
<td>query, query answer</td>
</tr>
<tr>
<td>(D, S)</td>
<td>database (instance), database schema</td>
</tr>
<tr>
<td>(r_1, r_2, \ldots, r_n)</td>
<td>rules</td>
</tr>
<tr>
<td>(S)</td>
<td>set or list of rules</td>
</tr>
<tr>
<td>(Dom(A))</td>
<td>domain of attribute (A)</td>
</tr>
<tr>
<td>(\ast)</td>
<td>wildcard character</td>
</tr>
<tr>
<td>(P(Q(D)))</td>
<td>provenance of results of Query (Q) in database (D)</td>
</tr>
<tr>
<td>(P(Q'(D)))</td>
<td>provenance of results of Query (Q') in database (D)</td>
</tr>
<tr>
<td>(k)</td>
<td>aggregate answer to query (Q)</td>
</tr>
<tr>
<td>(k')</td>
<td>expected answer and answer to target query (Q')</td>
</tr>
</tbody>
</table>

5.3.1 Preliminaries

We assume the user asks an aggregate query of the form:

\[ Q: \text{SELECT } A_S, f(A^*) \text{ AS } A \text{ FROM } R_1, R_2, \ldots \text{ WHERE conditions GROUP BY } A_G \]

Listing 5.1: The general form of aggregate query asked by the user

Where:

**Query relations** \(\mathcal{R}_M = \{R_1, R_2, \ldots, R_m\}\) is the set of relations in database \(D\)

**Query attributes** \(A_n = \{A_1, \ldots, A_n\}\) is the set of attributes of \(R_1, R_2, \ldots, R_m\)

**Group By** GROUP BY is an optional clause of the query, which uses standard GROUP BY semantics
**Group by attributes** if there is a GROUP BY clause, then $\mathcal{A}_G \subseteq \mathcal{A}_n$ is the set of attributes in that clause.

**Selected attributes** $\mathcal{A}_S$ is the selected attributes. If $Q$ contains a GROUP BY clause, then $\mathcal{A}_S \subseteq \mathcal{A}_G \subseteq \mathcal{A}_n$.

**Aggregated attributes** $\mathcal{A}^*$ is the aggregated attributes where $\mathcal{A}^* \subseteq \mathcal{A}_S$.

**Function** $f$ is an aggregate function, such as COUNT, which is applied to the aggregated attribute.

**Conditions** contain a conjunction or disjunction of conditions on the tables $\mathcal{R}_M$ as well as the join predicates.

**Example 21.** Continuing with Example 20, we have the aggregate query:

```sql
SELECT COUNT(*) FROM movie m, genre g
WHERE m.id = g.movie_id
AND m.revenue > 100m
AND m.rating > 7
AND g.genre = 'Adventure'
AND m.production_company LIKE 'Walt_Disney%'
```

Listing 5.2: Aggregate query from example 20

In this example, $f$ is the aggregate COUNT function. $\mathcal{A}^*$ is the wildcard $\star$. **conditions** contain a conjunction of 4 conditions. Query 5.2 returns $k = 5$ which is the number of movies that meet the conditions.

Given a query, $Q$ over a database, $D$, with schema, $\mathcal{R}$, we define the provenance of $Q$ as the set of tuples $P(Q(D))$ that contributed to the aggregate answer $Q(D)$ using lineage semantics (See Chapter 2 for provenance types details). We use $P(Q(D))$ to refer to the provenance of $Q(D)$ in $D$. For example, $P(Q(D))$ in Query 5.2 contains all the tuples in the MoviesGenre table that meet the conditions in $Q1$ ($t_1 - t_5$ in Table 5.1).
5.3.2 The scope of the query refinement solution

As motivated and defined in Section 5.1, Q is a query with results Q(D) such that k ∈ Q(D). The user submits feedback with k' which is the expected target value. The problem is to help the user find Q'' where Q''(D) is as close to k' as possible. In this section, we explain our assumptions and define the subset of queries we target with our approach.

For our study of this problem, we make an assumption that is common across many aggregate queries: the quantities that are being aggregated are non-negative. For example, movies do not earn negative money at the box office, and there are not a negative number of students who take classes. Under this assumption, we use P(Q(D)) to denote a set of tuples in the provenance of k ⊆ Q(D). We define expanding and contracting queries in terms of provenance:

**Definition 5.2** (Expanding the aggregate query). If $k < k'$, then the goal is to relax Q to get the answer $P(Q''(D))$, where $P(Q(D)) \subset P(Q''(D))$.

Example 20 shows an example of relaxing the query Q1.

**Definition 5.3** (Contracting the aggregate query). Else if $k > k'$, then the goal is to contract the query Q, where $P(Q''(D)) \subset P(Q(D))$.

We also restrict the class of aggregate functions we handle to: Count, Sum, Min, and Max. Count and Sum are both monotone functions that work similarly. For Min and Max, we utilize similar techniques to those used in Chapter 4. Summarizing the Max/Min function requires the same operations as finding a new Max/Min value set by the user. A natural extension to support the AVG function is to break down the average into Sum/Count. Other non-monotone functions require a more thorough investigation and are left as future work.

We also further restrict the class of queries we handle to exclude sub-queries. Sub-queries introduce complex concepts to provenance and summarization and we leave them as future work.
Rule containment

We use summary rules to represent the aggregation queries and to present suggestions to users. We use rules as introduced in Chapter 2 to model every query and every group of tuples in our solution. Representing queries and groups of tuples as rules enables our solution to find queries that contain other queries, or queries contained by others. It also allows us to find groups of tuples that fulfill refinement requirements. We use Definition 2.3 to define containment. A sub-rule contains super-rules, and super-rules are contained within sub-rules as per the definition.

Rule containment is a quick check over the rules’ attributes and values within. It works quickly and without checking the original data to determine groups of tuples that contain other groups of tuples. For example, the rule $R_1 : (language : en, Title : *, Year : *, Genre : Adventure, Rating : > 7, Productioncompany : PixarAnimationStudios)$ is contained within the rule $R_2 : (language : *, Title : *, Year : *, Genre : Adventure, Rating : > 7, Productioncompany : *)$. We say $R_1$ is contained in $R_2$ when $R_2$ is a super rule of $R_1$ as seen in Definition 2.3. It is the basis for our solution, which we go into further in Section 5.4.

We use the same format for the rules that we use in previous chapters. The only addition is that of representing simple disjunctive queries, which is a class of queries we did not handle in previous work. We expand the definition of rules to handle simple disjunctions for categorical values. For such queries, we use multiple rules to represent the disjunction.

Example 22. To give an example of what we mean by a simple disjunctive query, we show a query to select Action or Adventure movies. The query:

\[ Q^{\text{dis}}: \text{SELECT count(*) FROM moviesgenre} \]

\[ \text{WHERE genre in ('Action', 'Adventure')} \]

is represented by the rules: $\{r1 : (language : *, Title : *, Year : *, Genre : Action, Rating : *, Production company : *, count : 4, Rev : *)\}$,
5.3.4 Continuous Attributes

We treat continuous numeric attribute values as categorical attributes by defining buckets of intervals of values, e.g. $8 > Rating > 6$. We also define buckets of cumulative values, e.g. $Rating > 6$. The rules are created to facilitate rule containment and as such, they contain cumulative values like $> 100m$ or buckets such as $Rating > 6$ to $<= 8$. We initially bucketize numeric values based on 10 quantiles for the initial set of rules. If sufficient rules are found to fulfill the user constraint $k'$ we return those rules and the bucketization stops. We maintain access to the full values if we need to dynamically bucketize numeric values further. In cases where the limits of numeric buckets are too wide for relaxation or too small for contraction, we further bucketize the relevant buckets into 10 more quantiles. The process continues until the buckets provide values that can satisfy the user cardinality constraint. We find bucketizing continuous data to be the best, and fastest, approach to handling such attributes for the datasets we use. We characterize our datasets as consisting of mostly categorical attributes with a few continuous, numeric attributes that we use mainly as measure attributes (e.g. revenue in a movies dataset). There is research into finding patterns for numeric values [VGBS19] but it is not directly applicable to our problem and is currently outside of our scope.

5.3.5 Categorical attributes

While other query refinement solutions handle hierarchies, they do not propose any solution to handle categorical attributes that do not have hierarchies. Categorical attributes without hierarchies are the most common in our datasets. We believe users need to see other values of categorical attributes
to gain a full grasp of their data. In Example 20, the value of Pixar Animation Studios is completely hidden from the user because their query is asking for a specific categorical value (Disney Studios). Yet it is very likely to be the refinement they seek. This also comes into play in datasets that contain miscategorized information. Finding and recommending relevant categorical values is a key component of our solution.

We handle categorical attributes differently depending on their type. We define four different types of categorical attributes: 1) hierarchical attributes, 2) ordinal attributes, 3) semantically meaningful attributes, and 4) none of the above. We encode order for all attributes depending on their type.

For explicit hierarchies (e.g. City, State, Country) the traditional approach is to move up through the hierarchy to relax or move down to contract a query. However, hierarchies are not naturally available in our data and require user involvement to define them. In the presence of hierarchies, we use them as all other refinement systems.

For ordinal attributes, we encode order to be able to show users adjacent values. For attributes such as days of the week, the encoded order helps find closely related values such as the next day of the week to recommend more relevant suggestions. We encode order into ordinal attributes to facilitate suggesting relevant rules.

For semantically meaningful values, (e.g. movie genre) we calculate semantic distance based on WordNet [Mil95]. Alternatively, corpus-based measures can also serve this purpose. We use this measure to order these attributes based on conceptual closeness to each other. We can recommend more closely related movies to Q1 such as Action movies which can be more relevant to the user query. In Example 20, Action movies can be relevant to Q1 where the user is asking for Adventure movies. In cases where semantic similarity isn’t feasible, we can treat this class of attributes the same way we handle the next one.

For categorical attributes that do not fit any of the above, we use sum-
maries of associated attributes to encode order. For example, we can encode order to production companies in movies. Companies can be found to be more closely related if we examine movies, revenue, location, or actors involved. In Example 20, Pixar Animation Studios is very relevant to Q1 where the user is asking for Disney movies. By analyzing the properties of both production companies, we can encode order to the attribute values based on the similarity of common properties. Those properties are co-occurring attribute values such as genres, years, revenue patterns, and runtime. This analysis can be costly for attributes with many values and is done as an offline pre-processing step.

5.4 Solution Overview

We propose a framework that allows the user to explore interesting groups of tuples outside of the query answer set. Those groups correspond to query refinement candidates. Our system comprises the following components:

1. Rule generator to perform attribute reduction and rule generation and store rules in the rules database.

2. Provenance summary component to summarize the provenance of Q(D).

3. Candidate rule mining component to find candidate rules.

4. A candidate rule embedding and ranking component to compare the summary of the provenance of answer Q(D) against the candidate rules of non-answers.

The end result is a ranked list of the most recommended non-answers that can fulfill the users’ cardinality constraint.

Figure 5.2 shows the overall architecture of our solution. The user submits an aggregation query Q, they receive an unexpected result. The user then submits feedback on the expected value. For relaxing queries, we make the
assumption that the initial query $Q$ is partially correct. As in the final query result the user wants $Q'(D)$ containing the results they got from their initial query $Q$. The final result $Q'(D)$ comprises a collection of tuples with certain properties in common. Be it a categorical value, a numeric trend, or a correlation of values and trends. For contracting queries, the assumption is that the correct result is contained within the initial query result $Q(D)$. We produce a summary of the results that include common attribute values, numeric trends, and how those trends and values are correlated. We theorize that a refinement of the user’s query would include tuples that are consistent with those summary patterns. Those refinements are groups of tuples that when added to, or removed from, the result can fulfill the user’s constraints. Refinements can be one of two things: 1) a single rule that satisfies the cardinality constraint. 2) A conjunction of multiple rules that together cover enough tuples to meet the cardinality constraint.

Our framework comprises two parts: an offline and an online component. First is the rules generator component responsible for generating the rules table. This is done completely offline. In order to derive and recommend such patterns, we take the initial result $Q(D)$ as well as the feedback from the user $k'$ and run them through a number of different components. First, the provenance summary component produces a summary of the results $Q(D)$ with the highlights in terms of categorical properties and numeric trends. Second, is a candidate rule mining component to find candidate rules and tuples. Finally, we have a ranking mechanism that recommends these candidates in a list to be added to $Q(D)$ to produce the final answer $Q''(D)$. These components enable our approach to present real-time suggestions to users that can aid them in refining their uncertain queries. Mining useful rules alleviates the need for categorical hierarchies or multiple iterations.
5.4.1 Rule generator

The rule generator component produces comprehensive rules that cover the entire database \( D \). We simplify by joining the full database to produce a universal table and refer to it as \( D \). As a first step, we use correlation measures to reduce the number of attributes present in the table \( D \). Alternatively, we could use functional dependencies to achieve a similar effect. Without those, we rely on the more general correlation measures. The number of rules produced from a table is exponential in the number of attribute values. Thus, reducing the number of attributes involved in generating rules severely reduces the complexity of the operation. We use the Cramer’s V measure \(^{[AS79]}\) for categorical attributes and the Pearson correlation coefficient \(^{[BCHC09]}\) for numeric attributes.

After reducing the number of attributes, we run a CUBE query to generate rules that summarize the full table. CUBE queries produce aggregations for all attributes involved. Those aggregations include key statistics and measurements that cover the whole database.

The CUBE query is of the form:

\[
\text{CREATE MATERIALIZED VIEW rules\_cube AS}
\begin{align*}
\text{SELECT COALESCE}(A_1, '∗'), \\
\text{COALESCE}(A_2, '∗'), \\
\ldots \\
\text{COALESCE}(A_n, '∗'), \\
\text{count}(\ast) \text{ AS count}, \\
f_1(A^*) \text{ AS STAT} f_1^A, \\
\ldots \\
f_n(A^*) \text{ AS STAT} f_n^A, \\
\text{count}(\text{non NULL values}) \text{ AS weight}
\end{align*}
\text{FROM } R_1, R_2, \ldots, R_n
\text{WHERE conditions}
\]
GROUP BY CUBE($A_1, \ldots, A_n$);

Listing 5.3: The general form of cube query to generate rules

We further augment the list of rules by joining it with cumulative values of interest from the user query. Cumulative queries take on the form:

\[
\text{SELECT } A_1, \\
\text{count(*) AS count,} \\
f_1(G^*) \text{ AS } CUMULATIVE^{f_1^A}, \\
(\text{sum(count(*)) OVER (ORDER BY } A_1)) \text{ AS cumulative_count,} \\
(\text{sum(sum(A^*)) OVER (ORDER BY } A_1)) \text{ AS cumulative_sum} \\
\text{FROM R1} \\
\text{GROUP BY CUBE}(A_1); \\
\]

Listing 5.4: The general form of cumulative queries to generate cumulative values for attributes of interest

**Example 23.** [Cumulative value rules] A user asks a query that contains an inequality condition. The query is of the form:

\[Q: \text{SELECT count(*) FROM movie WHERE rating} > 7;\]

Typically, summarization rules contain attribute values and some aggregation statistics (Min, Max, Avg, Sum, Count). To find rules that refine queries with inequalities ($>$ or $<$), we must look up the cumulative sum of all ratings less than 7. This is an expensive query that requires a large operation over the whole table. Instead, we perform a cumulative query that adds up the cumulative sum of all revenue and counts for ratings. We also add the results of this query to the rules table. Furthermore, this step provides this information for any future queries that want to access cumulative information. The cumulative query to produce these rules for the query $Q$ is as follows:

\[
\text{SELECT rating,} \\
(\text{sum(count(*)) OVER (ORDER BY rating)) AS cumulative_count} \\
\]

106
FROM movie
GROUP BY rating;

With the CUBE query and cumulative queries, we obtain a set of rules that cover the entire database including summary statistics (count, min, max, sum, and average) in addition to cumulative values for attributes of interest (e.g. rating in movies). Those values enable the querying of rules that are directly related to the user query that can contain equality or inequality conditions.

5.4.2 Provenance summary

We use a summarization algorithm that produces a small number of rules \( k \) that summarize the most important properties of the result \( Q(D) \). We take the query result and summarize the provenance. Our summary algorithm can produce a summary based on the coverage for count queries or the impact of categorical values on numeric measure variables in the query as seen in Chapter 4.4.

We use sampling to reduce the time it takes to produce a summary. Our sampling method is similar to the one described in Section 4.4.3. We sample uniformly at random but also keep groups of tuples with a high likelihood of containing high coverage or high impact values. We resample for every new rule to cover as much of the data as possible.

5.4.3 Candidate rule mining

The search for candidate rules starts with the cardinality constraints set by the user. Any rules that do not meet the constraints are filtered out as an initial step. The rest of the rules that fulfill the constraint are added to a list of candidates \( C \). Combinations of rules that fulfill the constraint are
checked for data-independent containment. If data-dependent containment is needed, we go down to the tuple level to check for total marginal coverage. This is only done for rules that are not contained in higher-level rules that are already in $C$.

When individual rules in the rules database do not make up the cardinality constraint set by the user, we need to recommend multiple rules as a set of conjunctive rules that make up the refined answer. In this case, we reduce our problem to the subset-sum problem. In our problem, the total is $k'$ or the cardinality target which maps directly to the target sum $T$ in the subset-sum problem. The rules are our individual items are given a value of size (coverage for count) or impact (for sum, or average), and all are non-negative values. We want to greedily choose a group of non-overlapping rules that fit the total. We apply a greedy algorithm to solve this problem in polynomial time with a guaranteed $1 - \varepsilon$ approximation of the optimal solution. We leave the error threshold $\varepsilon$ up to the user to choose, it affects the result in terms of how close the algorithm gets to the cardinality constraint they set themselves. We use a modified algorithm from [IK75]. Our main modifications relate to pruning the candidates based on sub/super rules, and the values being marginal as in no overlap exists between candidate rules that are input for the algorithm. Algorithm 5 shows the greedy algorithm details.

Algorithm 5 is a greedy algorithm that solves the subset sum problem. The first step is to initialize an empty list. Then it proceeds to add rules one by one. Each rule is checked if it is contained by the previous rule. If not, the rule is added to the list if it does not exceed the size $T$. Rules are also trimmed if they are too close to other rules in terms of size; closeness is determined by the approximation factor $\varepsilon$ in $\varepsilon T/|C_n|$. This last trimming step prunes the list and guarantees that the sum does not exceed the target. The algorithm returns a set of rules with a total size close to the target $T$.

As an additional step, the rules in $C$ are augmented with super-rules on lower levels that are contained by other rules in $C$. The lower-level rules
Algorithm 5: Subset-sum greedy algorithm to find sets of rules that satisfy cardinality target

| **Input:** A set of rules \( C_n \) with resolved overlap, value \( T = \text{target cardinality} \), \( \varepsilon \) an approximation ratio  |
| **Output:** A group of rules \( C \) with total cardinality sum \( T \) |

1. \( C = \emptyset \)
2. \( S = \) all rules of size \( s < T \) sorted in ascending order
3. for \( i \) from 1 to \( |C_n| \) do
   4. foreach \( r \in C_n \) do
      5. if \( r \) is contained by any rule in \( C_n \): then remove \( r \), break;
      6. if \( y + \varepsilon.T/|C_n| < (\sum C) + r \leq T \) then \( y = r \), add \( r \) to \( C \);
7. return \( C_R \)

make up a summary of what’s contained in a rule in \( C \). The candidate rules and their summaries (super-rules) are then added to a final set of candidates \( C_f \) that is given to the next component to be embedded and compared.

### 5.4.4 Candidate rules embedding, and ranking

Candidate rules with attribute values that are close in order to the attribute values in the query are prioritized. For example, rules with the *Pixar* value are prioritized for a query like \( Q1 \) which contains the *Disney* value. e.g. Adventure, family, and animation genres are close together. The same thing applies for runtime values.

We embed the candidate rules with super rules that make up a summary of the candidate rules. We use Term Frequency Inverse Document Frequency [Sal91] vector embedding to facilitate comparison using soft cosine similarity metric. We chose TFIDF for its simplicity and faster performance than other methods, such as one-hot encoding. This metric also maintains enough semantic information for our purposes. Our approach for encoding groups of tuples is in the vein of similar work such as [CD03], with the exception of the absence of query workloads in our case. We treat each
rule and its corresponding super-rules as a single document. While TFIDF is more commonly used in natural language applications, we believe it is equally effective for our purpose. First of all, rules are composed mostly of categorical variables and those are not unique and can mimic words in documents. Unique values such as IDs and movie titles would not appear in summary rules and are replaced by the $*$ value. Second, rules contain numeric values that have been bucketized and thus can be treated the same way as categorical variables. Third, IDF takes care of common values that dominate summaries. Such values occur in most rules and are not distinct enough to use for ranking.

Example 24. Figure 5.3 shows a t-distributed stochastic neighbor embedding (t-SNE [VdMH08]) visualization of the encoded TFIDF vectors in 2-dimensional space. The visualization shows clearly how different attributes are encoded differently and clustered together. Furthermore, it shows how
similar values are clustered together, like Disney and Pixar for genre and
duration. Whereas equally similar values such as production_country are
equidistant from each other. This example demonstrates that an encoding
of TFIDF is sufficient to reach the suggestions highlighted in the result of
example [20]

After compiling the full list of candidates $C_f$, and generating the summary
of $P(Q(D))$, we rank the candidate rules and recommend the top rules to
the user. We use soft cosine similarity to rank the most likely candidates.
The model finds the top likely candidates in $C_f$ and ranks them according
to the degree of similarity to the provenance of results $P(Q(D))$.

5.5 User study

5.5.1 Experiment design

We designed a user study to show that our approach gives meaningful and
useful suggestions. We aimed to design tasks that tested the effectiveness and
meaningfulness of our recommendations against those made by the baseline
methods. We refer to our method as the provenance summary method (Prov-
Sum). This section aims to show that our method’s recommendations are
meaningful and that users prefer our method of interaction over those of the
baseline machine learning method.

Baseline methods For our baselines, we used out-of-the-box machine learn-
ing models that recommend users data that is relevant to their query results.
Using such baselines, we can show that such methods cannot learn the same
structures and thus cannot match the performance of our framework. The
most suitable methods involve classification, where the query result is the
target class, and the rest of the data contains both candidates for the target
class and irrelevant tuples.
This task description falls into the category of Positive Unlabeled Learning (PU learning) [BD20] where the tuples in the query result $Q(D)$ are labeled as positive. Everything else in the table is unlabeled. With a PU learning model, we can find labels for the unlabeled data (positive or negative) and show the user the positive results found by the classifier. To address the nature of the nontraditional tabular dataset that has positive and unlabeled records, Elkan and Noto proposed a method that leverages the estimated conditional probability of the positive instances being labeled to calibrate the prediction of a model trained on the labeled and unlabeled examples [EN08]. Following this approach, we use the pulearn Python package\footnote{https://pulearn.github.io/pulearn/} with a random forest underlying model to build the first baseline candidate (baseline 1) for our test scenarios; the bagging meta-algorithm in the random forests can mitigate the instability of classifiers caused by the PU Learning problems [MV14].

Additionally, we used the deep tabular data learning model TabNet [AP21] as another baseline candidate (baseline 2), which applies sequential attention for feature selection and mimics ensembling with its multi-step architecture. With TabNet, we built a classifier that treats the data as if it was traditionally labeled as positive and negative.

In order to decide whether to use baseline 1 or baseline 2 in the user study, we conducted an initial walk-through and 2 pilot studies. We concluded that the original PUlearn method [EN08] served as a better baseline and removed TabNet [AP21] from the user study. This is simply because the users in the pilot thought the output of PUlearn was more relevant and concise. TabNet was not designed for such a task and the output included a lot of irrelevant movies. In contrast, PUlearn produced a more concise list with more relevant tuples. Thus, since PUlearn would have dominated TabNet, we did not include it in the full user study.

We had the following goals for this experiment:
1. Establish that our method (provenance summary) provides meaningful query refinement recommendations to the user.

2. Find out which method (baseline or provenance summary) provides the best performance and is preferred by users.

Participants

In this study, we selected computer science students with experience using databases or tabular data. We define experience as an entry-level database course or equivalent knowledge. We recruited participants through the department of computer science public channels. We recruited 16 participants; we did not restrict recruiting based on other factors than experience with data. We offered users compensation in the form of 15 dollars.

Conditions

We had two conditions for the recommendation method: provenance summarization method (ProvSum), and baseline method (PULearn). As well as two conditions for the order of presentation: ProvSum first, and baseline first.

We varied the order of presentation of tasks by dividing users into two groups: A, and B. Tasks were the same across groups, but the order of presentation of recommendation method was varied by group.

We conducted the experiment with 16 participants and fully balanced the groups: A) 8 users started with the baseline method for the first task, and B) 8 users started with the ProvSum method for the first task.

Tasks

The users were presented with four different tasks. All tasks involved looking at a pre-written query, formulating a refined query, then evaluating if the
presented suggestions helped in formulating this refined query. A summary of tasks follows, the full details are in Appendix C.1.3.

1. Task 1: Select all Action movies that have total revenue > $100m, and have a vote average (rating) of >= 7. How would you go about expanding this query? Expand this query result by looking at suggestions made by our system, and choosing the most relevant ones.

2. Task 2: Select all Adventure movies with total revenue > $100m, and have a vote average (rating) >= 7. How would you go about expanding this query? Expand this query result by looking at suggestions made by our system, and choosing the most relevant ones.

3. Task 3: Select all movies with total revenue > $100m, that are made by male directors and released after the year 2015. How would you go about expanding this query? Expand this query result by looking at suggestions made by our system, and choosing the most relevant ones.

4. Task 4: Select all movies with total revenue > $1m, that are made by female directors and released after the year 2015. How would you go about expanding this query? Expand this query result by looking at suggestions made by our system, and choosing the most relevant ones.

We presented the users with preset queries and data from the IMDB dataset. This dataset included information about movies, genres, and directors. See Section 4.6.2 for a more thorough dataset description. Each user had to do the 4 tasks. Where they performed 2 tasks with each refinement method. For the tasks performed with the provenance summary method, users were presented with summaries of suggestions. Users could navigate the summaries and examine the details if they wished. For tasks performed with the baseline method, users were presented with a sample of individual movies to choose from. We aimed to limit the number of tuples shown so the mental load would be similar between the two methods. None of the
users were familiar with summary rules, so they needed time to process the new data format. 5-6 suggestions were shown in the provenance summary method, while about 10-15 individual movies were shown in the baseline method. Users were allowed to sort or filter the lists as they saw fit.

Instead of presenting all users with the 4 tasks using both interfaces for a total of 8 tasks, we opted for a design where each user would see task 1 with a certain method and then task 2 with a different method. We did the same for tasks 3 and 4. The two pairs of tasks (1, 2) and (3, 4) are very similar. We claim that tasks 1 and 2 are isomorphic ‘genre’ tasks. Whereas tasks 3 and 4 are isomorphic ‘director gender’ tasks. We make this claim because those tasks are very similar and ask the user to perform the same steps with slightly different data but similar structures. We expected that there would be strong learning effects, which we tried to avoid by using isomorphic tasks. In other words, we avoided users looking for the same recommendations they picked with the other method. The order of presentation of methods would ensure that the tasks are balanced and there is no bias against the method users see first in terms of time or preference.

**Design**

Our user study design was a two-factor design. The first factor was methods of recommending refinements (baseline, provenance summary; within-subject). The second factor was the order of presentation (baseline first, provenance summary first; between subject). We used a 2 x 2 design where the method of recommendation and the order of presentation were fully balanced.

**Procedure**

Figure 5.4 shows the user interface of our system. We started by asking the user to fill out a consent form. After that, each user took a demographic questionnaire on a laptop. In the questionnaire, we collected user data (age, and
experience with databases). We also assessed users’ familiarity with the domain (Movies dataset). We gave users a brief introduction to the data, tables, and user interface. We then briefed them about the tasks to be performed. Depending on which group they belonged to, the order of presentation of the refinement method was different, but the tasks remained consistent. Participants in group A experienced task 1 using the ProvSum method. Participants in group B experienced task 1 using the baseline method and so on.

To proceed, the user would sit in front of a laptop. We instructed the user to start a task. The moderator started the timer. The user performed the task as instructed. The moderator stopped the timer at the end of the task and asked the user to verbally evaluate satisfaction after the task and briefly justify their choices. After all tasks were complete, we asked users to fill in an exit questionnaire digitally on a laptop, see Appendix C.1.3 for details. The questionnaire has quantitative and qualitative questions that are subjective. We used a 1-5 Likert scale for quantitative questions. We also
included questions of the form “If no, please explain” or “Please, explain.” to gain some qualitative feedback.

In cases where the experiments had to be performed remotely, we had the system deployed on a server. Users were asked to perform the tasks through their browsers while sharing their screens to be observed by the moderator conducting the study. The questionnaires were filled out through a different tab in the browser. 10 experiments out of the total of 18 were conducted remotely. We did not observe any effect on the time to perform tasks or the quality of answers.

Apparatus

We used the following tools: a laptop with the software interface open in a browser tab, a timer, and input forms open in a different tab. For remote studies, we used Zoom to observe participants and asked them to share their screens. We deployed the system and allowed users to access it through a link.

Variables and hypotheses

- **Independent variables**: type of refinement method (baseline, provenance summary).

- **Dependent variables**: Time to perform a task, satisfaction (subjective 1-5 Likert scale), perceived usefulness and preference (subjective 1-5 Likert scale).

We measured each variable as follows:

- **Time**: Timer from start to finish of the task. The timer started when users hit start. We define the end of the task as the point when the user hit “submit result”.

- **Satisfaction**: 1-5 Likert scale question after each task.
- **Usefulness and preference**: Subjective quantitative (1-5 Likert scale) and qualitative questions on usefulness and preference in exit questionnaire.

**Hypotheses**: We tested the following hypotheses:

**Performance (Task completion time)**

- **H1**: Users will perform the tasks faster with the provenance summary method compared to the baseline method.

**Usefulness and satisfaction**

- **H2**: Users will rate the provenance summary method higher in terms of perceived usefulness compared to the baseline method.
- **H3**: Users will rate the provenance summary method higher in terms of perceived satisfaction compared to the baseline method.

**Limitations/Problems**

We present our limitations regarding the different categories of threats to validity:

**Internal validity**: We attempted to make the size of output for both methods equal. However, the Pu-learn methods generate more tuples than our method generates rules. In our pilot study, we limited the tuples to those with the highest accuracy. We also note that rules are harder to read and decipher and take longer for users. Therefore, we limited the ratio of tuples to rule to 3:1. We think this is fair as we observed users make quick decisions about tuples but take 2-3 times longer to make decisions about rules. Furthermore, satisfaction and preference can be affected by elements that we cannot measure or control.

**Statistical validity**: We believe we used the appropriate statistical tests. We also verified the assumptions of the tests before we ran them on our data to make sure they are appropriate.
External validity: The results should give a good representation of how users would interact with our system.

Ecological validity: Our tasks should be representative. We validated them with a small pilot study before proceeding with the full study.

5.5.2 User Study results

16 participants completed the study. Results show participants favor our method of interaction over presenting individual tuples. They also favor our recommendations over those presented by the baseline methods. In their evaluation of the suggestions of both methods, most users picked the most semantically similar tuples or rules. We noticed users gravitate towards Adventure movies when the query is about Action movies and vice versa. This shows that our intuition in ranking rules according to semantic similarity is sound.

Time. We saw statistically significant effects of the different methods on the time it took to complete a task, and the perceived usefulness of the methods. We report significance as \( P < 0.01 \). We ran an ANOVA test after running some checks to make sure it applies to our data. We ran a Bartlett test to check for homogeneity of variances, a Shapiro-Wilk normality test and drew q-q plots to check normality. The ANOVA test we ran took method \( x \) group as factors. Groups represent the order of presentation. The results of the test show a significant main effect of the method \( (F = 41.67, P < 0.001) \) on the time it takes to finish a task. We observed no significant effect of the order of presentation \( (F = 2.01, P > 0.1) \) and no interaction effect between the two factors \( (F = 2.88, P > 0.09) \). We also ran a separate test to make sure there was no significant effect between the different tasks \( (P > 0.01) \).

In general, it took less time to finish tasks with ProvSum than PU Learn, the breakdown for tasks and method statistics is in Table 5.3. It also took

\footnotetext{The study was approved as minimal risk by the Behavioural Research Ethics Board at the University of British Columbia. UBC BREB number:H21-03344.}
more time to finish tasks for people in group A (Mean = 100.5 seconds) than in group B (Mean = 83.25), but this was not statistically significant, according to our tests.

Table 5.3: Table shows the average time (in seconds) it took users to finish tasks using the two methods (ProvSum and PULearn).

<table>
<thead>
<tr>
<th>Task</th>
<th>ProvSum</th>
<th>PULearn</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td>54.7</td>
<td>116.9</td>
</tr>
<tr>
<td>t2</td>
<td>53.9</td>
<td>82.6</td>
</tr>
<tr>
<td>t3</td>
<td>53.7</td>
<td>183.7</td>
</tr>
<tr>
<td>t4</td>
<td>48.7</td>
<td>141.8</td>
</tr>
</tbody>
</table>

Figure 5.6 presents the full set of results for time for the two groups. We see the time it took users from both groups A and B to finish the tasks. We see group A took significantly longer to finish tasks with PULearn (t1 and t3) than tasks with ProvSum (t2 and t4). This effect is less pronounced for group B but still significant. Group B used ProvSum for t1 and t3 and used PULearn in t2 and t4. Figure 5.7 shows the breakdown of time by method, for ease of reading.

Usefulness. We collected usefulness measurements as a question at the end of experiments, asking participants to evaluate usefulness on a 1-5 Likert scale where 1 is very useful and 5 is very useless. We collected two groups of measurements: How users felt about ProvSum suggestions (mean = 1.3) and how they felt about PULearn suggestions (mean = 2.25). We ran a non-parametric test to compare the two measurements: the Wilcoxon Signed Ranks test. We found a significant difference between the perceived usefulness of the two methods (adjusted p=0.002). In Figure 5.5, we see that most users rated ProvSum as very useful, and only 1 user picked the neutral rating. The majority of users rated PULearn as useful or neutral, with only 1 user rating it as very useful. None of the users rated either as less than neutral (3).

Satisfaction. We collected satisfaction measurements as a question at the
end of every task. We asked users to evaluate their satisfaction on a 1-5 Likert scale. We ran a non-parametric test to compare the two measurements (satisfaction for group A and group B for each task). We ran the Wilcoxon Signed Ranks test. We did not find a significant difference between the two groups depending on which method was used to finish the task (adjusted p=0.9).

**Qualitative results.** Users wrote justifications for their choices and answered open-ended questions. Notably, users commented that our rules are optimal if the user is not familiar with the dataset. Individual movie recommendations only worked for those familiar with the data or the domain. Users with no familiarity with movies (at least 3 of our users said they were not familiar with American movies at all) had a hard time picking recommendations from a list of individual movies and much preferred the grouping format of our rules. Some quotes from our users: “individual movies would give more accurate answers but patterns are easier to use/interact with”, “The pattern tool gives very clear presentation. I think it would be very useful for such tasks.”, “It took time to find answers in the individual movies, time-consuming. The pattern is preferred.”.

**Discussion.** We saw a significant effect on both usefulness ($p < 0.002$) and time ($p < 0.001$). This leads us to believe that our hypotheses (H1 and H2) are supported. We did not find a significant effect on satisfaction to support our third hypothesis (H3) ($p > 0.5$). We suspect that satisfaction was high across the board because the question was asked after every task is done and did not account for how much effort it took to finish the task. Perhaps a question about difficulty would have served us better. Usefulness data was collected at the end of the experiment so users had an idea about both methods and could evaluate them with more complete knowledge. We did not test for accuracy or errors because the tasks are fairly open-ended and subjective. We did, however, inspect the responses by users and ask users to evaluate their choices. The qualitative answers give insight into how users
Figure 5.5: Users’ perceived usefulness for both types of refinement method (1: very useful - 5: Very useless).

see their choices and how they feel about the two methods. All users showed interest in our suggestions and none of them asked for different suggestions. They also commented that our suggestions work better for users unfamiliar with the data which is the intended goal of our approach.

This user study showed evidence that refinement suggestions are useful and usable. We also showed that our users prefer our method’s suggestions to those of comparable machine learning methods.

5.6 Performance evaluation

This section has three objectives: (a) to evaluate the performance of our provenance summarization method and show it can work in real-time; (b) to compare our method with machine learning methods and show provenance summaries outperform those baselines in terms of real-time performance; and
Figure 5.6: The time it took users to complete tasks for both groups of users (Group A started with the PULearn method, Group B started with the ProvSum method)
Figure 5.7: The time it took users to complete tasks using both methods.
(c) to conduct experiments to show how our system compares to a baseline in query refinement in terms of interaction and output.

5.6.1 Experimental Setup

We implemented our algorithms in Python and ran the experiments on a machine with a 3.4GHz Intel CPU and 32 GB RAM that uses PostgreSQL 10. The provenance generation component is Perm [GMA13].

We implemented our method and 2 baselines as described in the previous section: PULearn and TabNet. We found TabNet, in general, to be more expensive to train. Since our problem requires retraining with every new query, TabNet would take way too long (up to a day or so) for many experiments and including the results in the graphs would completely distort them. Hence we omit TabNet results where it becomes impossible to meaningfully plot them with the other methods.

datasets As far as we know, there is no standard benchmark for query refinement. We used two datasets to conduct our experiments: IMDB, and TPC-H. The data characteristics are as follows: The number of tuples (|D|): 181k + 323k (IMDB), and 60k-3m (TPC-H). The number of tables (N): 3 (IMDB) and 8 (TPC-H). The total number of attributes in the tables (mD): 15 (IMDB) and 61 (TPC-H). The number of queries (q): 4 (IMDB) and 5 (TPC-H).

IMDB. We used a subset of the IMDB dataset with 3 tables, Movies, Genre, and Directors. The movies table included revenue for which we calculated the aggregate results. We used this dataset for performance analysis. In those experiments, we ran the same queries that were run in the user study with a different number of attributes and different aggregate functions:

```
SELECT G1, G2, ..., f(Rev) FROM Movies m, Directors d
WHERE m.Director = d.Name GROUP BY G1, G2, ...
```

This is the same dataset we used in Chapter 4.
TPC-H. We used TPC-H for performance experiments and to show how our methods handle queries with multiple types of aggregation and multiple complex joins. We varied the database size as follows: 60k, 300k, 600k, 1.5m, 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 provenance summarization method.

5.6.2 Runtime Performance

Exp-1: Effects of $|D|$. Figure 5.8 shows clearly that our approach grows linearly with size while outperforming the machine learning methods. This

---

http://www.tcp.org/hspec.html
Figure 5.9: Method and time in IMDB.

Figure 5.10: Breakdown of prediction/training for PULearn in IMDB dataset.
Figure 5.11: Effect of indexing in IMDB.

Figure 5.12: Preprocessing steps breakdown in IMDB.
is mostly because training takes a long time even for smaller data sizes. Prediction, on the other hand, performs really well at small sizes. The problem is that machine learning methods need to train after every user query. The only exception is Q5 in TPC-H, where PUlearn outperforms and scales better than our method. This is due to the small size of the query results which minimizes the training time. Figure 5.9 shows the results for the IMDB dataset with the same trend we saw in the TPC-H data, except that now we see provsum outperforming the other methods. TabNet performs much worse but it executes here, whereas, for the TPC-H data, TabNet takes too long when data sizes exceed 600k tuples (over 1 hour).

**Exp-2: Effects of optimizations.** We do not address the effects of sampling on summarization methods because it is already addressed in the previous chapter. Summarization is also prohibitively expensive without sampling. The other major optimization is the indexing of the rules table. We build indexes that facilitate access to key attributes in the rules table. Figure 5.11 shows the effect on the IMDB dataset.

**Exp-3: Preprocessing costs.** Figures 5.12 and 5.13 show the preprocessing times. Preprocessing is not as costly as we initially expected. It is
an offline step performed once at the beginning of the process. The CUBE queries are surprisingly fast on modern hardware. The indexes take almost as long to build as the CUBE queries take to execute. The cumulative queries and joins to the rules table are quite fast, in comparison. For larger sizes (> 1.5m), the preprocessing is better left as an offline step but for smaller data, it can work as an online step and still achieve real-time performance (under 1 minute).

**Exp-4: Comparison to existing query refinement solutions.** Our approach is the only one as far as we can tell that is completely automated and does not require much user interaction. We compared our approach to an implementation of the query refinement system in [MK08, MK09] to see how many steps it would take a user to arrive at their best refinement when compared to ours. We ran the same query $Q_1$ to see how the system behaved. The lack of hierarchy limited our options for categorical variables; therefore, we set the same numeric values for $revenue > 300m$ and $rating > 7$. The system took our input query, as well as a cardinality constraint, $k$. The system gave us an estimate for the number of tuples at that level, asked us to refine one of the two attribute values (revenue or rating), and provided estimates at different levels. Proceeding with the query, we provided a smaller number for revenue and hit the target after 2 more refinements at $revenue > 95m$. We note that a user with very little knowledge of the data would find it difficult to navigate these suggestions. Whereas for our approach, the user would only have to submit the query with feedback and receive the same answer contained within a set of suggestions. The main shortcoming of our solution, in comparison, is that it relies heavily on the bucketization of numeric values, whereas the other approaches provide a more fleshed-out approach to refining numeric attributes. Both approaches can complement each other to achieve better summaries of categorical and numeric attributes.

**Discussion.** Machine learning models require very costly training after every query, which makes them ill-suited for this problem. However, PUlearn meth-
ods are still comparable despite some assumptions about the data. Those assumptions include: 1) labeled data being selected randomly, and 2) the two classes (positive and negative) being naturally separate. Those assumptions are not necessarily in line with the problem. The positive set isn’t necessarily representative and the 2 classes can be highly subjective. The approach is still useful despite those assumptions.

Another hurdle to using PUHearn methods is the encoding of categorical data. 1-hot encoding is prohibitively expensive for any reasonably sized dataset. In our experiments, 1-hot encoding became unusable at the second scale for TPC-H and wasn’t usable at all for the IMDB dataset. We opted for a simpler encoding of converting categorical values into numbers. This encoding adds order to non-ordinal values which could prove problematic for some classifiers but seemed to be the only reasonable solution in terms of scalability.

Our main goal with these experiments was to show the scalability and real-time performance of our method. As we can see with the TPC-H experiments, our query refinement approach works in real-time even at larger data sizes. Sampling and indexing contribute to this performance. Reducing the number of attributes also works to improve performance.

We also show briefly, in comparison to [MK09], that our approach works to give automated suggestions that the baseline here would take several steps to show to the user, if at all.

We set out to show the performance and scalability of our approach. Our experiments demonstrate that we can perform query refinement in real time for datasets or sizes up to $3n$ tuples. With our user study, we demonstrated that our refinement suggestions are useful and usable. We also showed that our users prefer them to the output of comparable machine learning methods.
5.7 Conclusion and future work

In this chapter, we presented a novel, automated, and data-centric approach to query refinement. This approach utilizes provenance summaries and summary rules to present users with suggestions that can help them refine their uncertain queries. With the performance experiments, we fully evaluated our solution and studied the effects of different factors. In the user study we conducted, we showed that users prefer our recommendations against those generated by machine learning methods. We also showed that our method gives useful and meaningful recommendations. Given the results of our experiments and user study, we can conclude that our approach to query refinement performs well and in real-time. There is more to be done to expand the class of queries as well as the possible recommendations to include joins and set operations. We leave those items as future work recommendations that we expand on in the following chapter.
Chapter 6

Conclusions and future work

The main focus of this thesis has been on examining data provenance in the context of relational database management systems from the perspective of potential users. The thesis does not present a single solution or system that addresses usability, instead, it works in three complementary different directions. The first direction is a visualization of relational provenance information in a specific context. The second direction is summarizing relational provenance information of aggregation queries for general applications to improve usability. The third direction had the goal of curating provenance information of aggregation queries to address a specific challenge, that of aggregate query refinement. All three directions address related issues with the usefulness and usability of provenance information in relational databases.

For the conclusion of this thesis, we will list a summary of contributions for each direction and give some brief future work recommendations. Furthermore, we will also include a discussion section to outline how this thesis tackled the challenges posed and where it fits within the literature.
6.1 Summary of chapters and future work

In Chapter 3, we went through the process of developing a system we name Pastwatch. We based our approach on certain principles from the literature on data provenance and visualization. We touched briefly on the summarization of provenance as a component of this system. We conducted a user study to take in the user perspective on how such a system compares to one with minimal effort to present provenance information. Our results show significant effects of visualization and curation techniques we employed in Pastwatch.

We left Chapter 3 with the following recommendations:

1. Summarization techniques need further exploration to develop summaries that help users in different scenarios. This is an item that we will cover in the next chapter.

2. For more general-purpose applications, provenance can be catered to the user with automated visualization suggestions such as those presented in [DD19].

In Chapter 4, we looked at provenance summarization for multiple purposes and devised different solutions for different scenarios. Impact summaries for users interested in certain numeric values. Comparative summaries for users interested in comparing two or more values in an aggregate query result. Join summaries for queries with join. As well as different scenarios depending on the different aggregate functions.

For future work, we recommend the following:

1. Support for nested queries with similar mechanics that we discussed for join queries. However, the scalability of such queries is a concern given the added complexity of sub-queries and joins.

2. We use contribution semantics for impact summaries. Other semantics such as [Sha97] are gaining traction in the field as a serious alternative.
3. Our focus was on optimizing sub-modular functions. Assuming a lack of sub-modularity, we briefly explored alternative score functions in [3]. Those functions would not work with the greedy algorithms we propose and as such, we left them as future work.

Last but not least, Chapter 5 presents an approach to query refinement for aggregation queries. This approach is data-centric in that it mines the data for useful refinement suggestions and does not rely on generating new queries. Instead, the suggestions are shown to the users in the form of groups of data.

There is more work to be done in this area:

1. In terms of expanding the class of queries. One direction is to expand the work to include sub-queries. A simple and natural extension is to allow sub-queries without aggregation. However, nested aggregation presents the extra complexity of dealing with multiple aggregates at multiple levels.

2. Another extension is giving the system the ability to suggest new joins. Techniques such as [ZNDM19] can be used to automate join discovery as a more advanced method than relying on database constraints.

Throughout the work in this thesis, we have used user-centric experiments (user studies in Chapters 3 and 5 and a user survey in Chapter 4). We used those approaches not only to validate our work and results but also as a means to inform our design and validate our evaluation metrics. We believe any claims made about usability should be informed by the users and this is the approach to research we took to inform our findings. We believe grounding our work in usability and following what we hope are best practices has helped us get closer to the goal of usable provenance information in this context.
6.2 Discussion and reflections

In Chapter 1 we outlined 3 challenges. To reiterate them briefly:

1. Provenance information needs to be visualized in a way that facilitates exploration. Insightful visualizations of all aspects of provenance are important.

2. The volume of provenance information for an aggregation result can be large and prohibitive for exploration.

3. When a user issues an erroneous aggregate query, provenance plays a role in looking at the origin of the answer but not finding the correct answer. We utilize the techniques developed to tackle the first two challenges to solve this one.

Throughout this thesis, we have described solutions that we believe contribute to addressing these challenges. To summarize how we tackled these challenges:

1. We presented a provenance visualization framework that includes all the elements of provenance in databases. Furthermore, we presented the results of a user study we conducted to show that users prefer curated and visualized provenance data and can use such data to perform tasks more efficiently.

2. We tackled the problem of aggregation provenance by presenting solutions to summarize this provenance. We present two different summarization methods with the main goal of supporting meaningful exploration of provenance information.

3. We aimed to solve the aggregate query refinement problem using provenance summaries and data summaries. In doing so, we posed a problem where provenance data is useful if not sufficient by itself. We used the techniques we developed in our work to handle this new problem.
Throughout these chapters, we presented user-centering methods to evaluate our solutions. We believe that in doing so, we can further emphasize the need for such methods in database research. In all our cases, we observed a relationship between performance experiments and user studies. While performance experiments are great for evaluating efficiency, the objective metrics often proposed within may not always align with those important to users. We hope our work on user experiments will motivate future research in usability and efficiency.

There is still a lot of work to be done in the area of provenance. We hope that this thesis presents ample work in addressing some of the major challenges. The future is clearly headed toward employing such techniques to explain the results of machine learning models. The challenges of visualization/summarization and improving usability are at the forefront of this direction. We believe our work can contribute to that, and we can further extend some of it in this direction.

We have explored the use of machine learning methods in our query refinement problem (Chapter 5) and found them lacking. Databases are a tricky field for machine learning or deep learning approaches to be sufficient on their own, without tremendous engineering effort to make them fit. The main challenge comes up when we have a lack of training data such as query logs, session logs, or workloads. We presented our work to fill a gap where such information may not be present or transferable from another domain.

As with any other work, there are limitations to what can be done when trying to work in two different directions. Doing deeply technical work while utilizing user studies means one of these aspects will not be fully developed or explored. We did not purposefully undermine one side for the other, we attempted to balance both aspects of this work. While not the main focus of this thesis, we hope the user experiments described here will help to build the standards for user studies in database research.
Bibliography


Appendix A

Provenance visualization

A.1 User study materials

The following are the user tasks and questionnaires given to users in paper form to perform the study.

A.1.1 Task Questionnaire

Case 1. Find the company with LEI “213800E546PSA5GCOD18” in Australia.

Where does this piece of data come from originally? From which table does it originate?

Have the column names been changed from the origin?

Has it been modified from its origin?

On a scale of 1 to 5 with 1 being very difficult and 5 being very easy, how easy was it to find the information on the data origin?

1. Very difficult
2. Difficult
3. Normal
4. Easy

5. Very easy

Why did you choose this rating?

**Case 2.** Find the company with LEI “549300S208H6WNCAFD40” in Canada.

Where does this piece of data come from originally? From which table does it originate?

Have the column names been changed from the origin?

Has it been modified from its origin?

On a scale of 1 to 5 with 1 being very difficult and 5 being very easy, how easy was it to find the information on the data origin?

1. Very difficult

2. Difficult

3. Normal

4. Easy

5. Very easy

Why did you choose this rating?

**Case 3.** Find a company in the United States that originated from the “London Stock Exchange”.

What is the LEI of this company?

Have the column names been changed from the origin?

Has it been modified from its origin?

On a scale of 1 to 5 with 1 being very difficult and 5 being very easy, how easy was it to find the information on the data origin?

1. Very difficult

154
2. Difficult

3. Normal

4. Easy

5. Very easy

Why did you choose this rating?

**Case 4.** Find a company in the Australia that originated from the "WM Datenservice".

- What is the LEI of this company?
- Have the column names been changed from the origin?
- Has it been modified from its origin?
- On a scale of 1 to 5 with 1 being very difficult and 5 being very easy, how easy was it to find the information on the data origin?

1. Very difficult

2. Difficult

3. Normal

4. Easy

5. Very easy

Why did you choose this rating?

**A.1.2 Exit Questions**

Is there any extra information you wish to see in this tool?

- Do you have any suggestions for improvement of interaction? If so, what are they?
On a scale from 1 to 5 with 1 being very likely and 5 being very unlikely, how likely are you to use provenance information to determine the reliability/accuracy of a piece of data?

1. Very likely
2. Likely
3. Neutral
4. Unlikely
5. Very unlikely

Why did you choose this rating?
Appendix B

Provenance summarization

B.1 Alternative Score Functions

We present two alternative score functions, *accumulative 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 4.3, they are non-submodular functions and do not allow efficiently computing sub-optimal summaries.

B.1.1 Accumulative Impact Summaries

We present $A\text{Impact}^t$ (accumulative impact) as an alternative for $\text{Impact}^t$ in Equation [4.2] that computes the impact of a tuples covered by a rule as a group:

$$A\text{Impact}^t(s_i, S) = |Q^t(R) - Q^t(R \setminus M\text{Cover}(s_i, S))|. \quad (B.1)$$

Using this impact function, we define a new score function in Equation [B.2]...
\[ AIScore^t(S) = \sum_{s_i \in S} AImpact^t(s_i, S) \times Weight^t(s_i), \quad (B.2) \]

The following example applies this new score function and compares it with the score function used in the impact summaries in Section 4.3.

**Example 25.** Consider the following query on MoviesDirectors that returns two answers, \( w_1 = (Female, 312.3 \text{ M}) \) and \( w_2 = (Male, 337.3 \text{ M}) \):

\[
Q: \text{SELECT} \ Gender, \ \text{AVG(Rev) AS AvgRev FROM MoviesDirectors GROUP BY Gender}
\]

Let assume the user requests an impact summary for \( w_2 \). Consider summaries \( S = \{s_1\} \) and \( S' = \{s_3\} \) with \( s_1 = (Title : \ast, Year : \ast, Genre : \ast, Rating : 7, Rev : \ast, Director : \ast, Gender : Male, Country : \ast) \) and \( s_3 = (Title : \ast, Year : \ast, Genre : Action, Rating : \ast, Rev : \ast, Director : \ast, Gender : Male, Country : \ast) \), we compute the score functions w.r.t. the score functions in Equations 4.1 and B.2 as follows:
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 B.2 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).

### B.1.2 Entropy-based Comparative Summaries

We presented comparative summaries in Section 4.3.4 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.
function that gives a different measure of the balance between the covered
tuples using binary entropy function:

\[
ECScore^{t,t'}(S) = \sum_{s_i \in S} MCount^{t,t'}(s_i, S) \times \text{Weight}(s_i), \quad (B.3)
\]

This comparative score function is different from the sore function in Def-
ition [2.6] 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:

\[
(MCount^t(s_i, S) + MCount^{t'}(s_i, S)) \times \text{Entropy}^{t,t'}(s_i, S). \quad (B.4)
\]

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 de-

defined in Definition [2.5]). The binary entropy function \( 0 \leq \text{Entropy}^{t,t'}(s_i, S) \leq 1 \) 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:

\[
\text{Entropy}^{t,t'}(s_i, S) = H(p(s_i, S)^t) = H(p(s_i, S)^{t'}). \quad (B.5)
\]

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,

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

**Example 26.** In our running example, considering \( t = g_1, t' = g_4 \) and \( s_1, \ldots, s_4 \), the score of \( \{s_1, \ldots, s_4\} \) 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 \( MCount^t_{t'}(s_5, \{s_5\}) = 2 \times H(0.5) = 2 \) and \( \text{Weight}(s_5) = 3 \).

**Proposition 2.** The score function \( ECScore \) in Equation B.3 is not submodular.

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

### B.2 Proofs

**Proof of Theorem.** 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 \( Q'(R) \), let \( L(r, S) \) be the first rule in \( S \) that covers \( r \) (the rule that covers \( r \) and has the maximum score). Let \( \Delta^t_r = |Q_t(R) - Q_t(R \setminus \{r\})| \) be the impact of \( r \) on \( Q_t(R) \). The score can be rewritten as the following:

\[
IScore^t_t(S) = \sum_{r \in R} \text{Weight}^t_t(L(r, S)) \times \Delta^t_t(r). \tag{B.6}
\]

Assuming a set of rules \( S' \) such that \( S \subsetneq S' \) and a rule \( s \notin S' \), we prove \( IScore^t \) is submodular by showing the following always holds: \( IScore^t_t(S \cup \{s\}) - IScore^t_t(S) \geq IScore^t_t(S' \cup \{s\}) - IScore^t_t(S') \). The marginal score for \( s \) w.r.t. \( S \) and \( S' \) can be written using Equation B.6. For example for \( S \), the marginal score \( IScore^t_t(S \cup \{s\}) - IScore^t_t(S) \) can be written as the following:
\[
\sum_{r \in R} [\text{Weight}^t(L(r, S \cup \{s\})) - \text{Weight}^t(L(r, S))] \times \Delta^t(r).
\]

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.** \(\text{Weight}^t(L(r, S' \cup \{s\})) - \text{Weight}^t(L(r, S')) > 0\) 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 \(\text{Weight}^t(L(r, S \cup \{s\})) \geq \text{Weight}^t(L(r, S))\) and \(L(r, S) = s\) which means

\[
\text{Weight}^t(L(r, S \cup \{s\})) - \text{Weight}^t(L(r, S)) \geq \text{Weight}^t(L(r, S' \cup \{s\})) - \text{Weight}^t(L(r, S')).
\]

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

\[
\text{Weight}^t(L(r, S \cup \{s\})) - \text{Weight}^t(L(r, S)) \geq \text{Weight}^t(L(r, S' \cup \{s\})) - \text{Weight}^t(L(r, S'))
\]

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

\[
\text{Weight}^t(L(r, S \cup \{s\})) - \text{Weight}^t(L(r, S)) \geq \text{Weight}^t(L(r, S' \cup \{s\})) - \text{Weight}^t(L(r, S')).
\]
which proves the theorem. This proof is based on the proof of [JGP16, 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 $C$Score function can be rewritten as the following:

$$C_{Score}^{t_1, t_2}(S) = \sum_{\langle r_1, r_2 \rangle \in R^{t_1} \times R^{t_2}} Weight^{t_1, t_2}(L(\langle r_1, r_2 \rangle, S)). \quad (B.7)$$

Assuming a set of rules $S'$ such that $S \subsetneq S'$ and a rule $s \notin S'$, we prove $C_{Score}^{t_1, t_2}$ is submodular by showing the following always holds: $C_{Score}^{t_1, t_2}(S \cup \{s\}) - C_{Score}^{t_1, t_2}(S) \geq C_{Score}^{t_1, t_2}(S' \cup \{s\}) - C_{Score}^{t_1, t_2}(S')$. The marginal score for $s$ w.r.t. $S$ and $S'$ can be written using Equation (B.7). For example for $S$, the marginal score $C_{Score}^{t_1, t_2}(S \cup \{s\}) - C_{Score}^{t_1, t_2}(S)$ can be written as the following:

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

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.** $Weight^{t_1, t_2}(L(\langle r_1, r_2 \rangle, S \cup \{s\})) - Weight^{t_1, t_2}(L(\langle r_1, r_2 \rangle, S)) > 0$ 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 $Weight^{t_1, t_2}(L(\langle r_1, r_2 \rangle, S \cup \{s\})) \geq Weight^{t_1, t_2}(L(\langle r_1, r_2 \rangle, S))$ and...
\[ L(⟨r_1, r_2⟩, S) = s \] which means

\[
\text{Weight}^{t_1,t_2}(L(⟨r_1, r_2⟩, S \cup \{s\})) - \text{Weight}^{t_1,t_2}(L(⟨r_1, r_2⟩, S)) \geq
\]

\[
\text{Weight}^{t_1,t_2}(L(⟨r_1, r_2⟩, S' \cup \{s\})) - \text{Weight}^{t_1,t_2}(L(⟨r_1, r_2⟩, S')).
\]

**Case 2.** \[ \text{Weight}^{t_1,t_2}(L(⟨r_1, r_2⟩, S \cup \{s\})) - \text{Weight}^{t_1,t_2}(L(⟨r_1, r_2⟩, S)) = 0 \]

and

\[
\text{Weight}^{t_1,t_2}(L(⟨r_1, r_2⟩, S \cup \{s\})) - \text{Weight}^{t_1,t_2}(L(⟨r_1, r_2⟩, S)) \geq
\]

\[
\text{Weight}^{t_1,t_2}(L(⟨r_1, r_2⟩, S' \cup \{s\})) - \text{Weight}^{t_1,t_2}(L(⟨r_1, r_2⟩, S')).
\]

because \[ \text{Weight}^{t_1,t_2}(L(⟨r_1, r_2⟩, S \cup \{s\})) - \text{Weight}^{t_1,t_2}(L(⟨r_1, r_2⟩, S)) \geq 0. \]

This means for every \( ⟨r_1, r_2⟩ \in R^{t_1} \times R^{t_2}, \)

\[
\text{Weight}^{t_1,t_2}(L(⟨r_1, r_2⟩, S \cup \{s\})) - \text{Weight}^{t_1,t_2}(L(⟨r_1, r_2⟩, S)) \geq
\]

\[
\text{Weight}^{t_1,t_2}(L(⟨r_1, r_2⟩, S' \cup \{s\})) - \text{Weight}^{t_1,t_2}(L(⟨r_1, r_2⟩, S')).
\]

which proves the theorem. \( \square \)

**Proof of Proposition 1.** The proof of sub-modularity for COUNT is given in [JGP16, 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 (4.1) 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: \( \text{SELECT SUM}(A_3) \text{ FROM } R \). Let \( S_1 = \emptyset \) and \( S_2 = \{(a, b, \ast)\} \) be sets of summarization rules and \( s = (\ast, b, \ast) \) be a summarization rule. We compute the following score values using the score function in (4.1) for the only tuple \( t \) in \( Q(R) \): \( IScore^t(S_1) = 0, IScore^t(S_2) = 2, IScore^t(S_1 \cup \{s\}) = 0, IScore^t(S_2 \cup \{s\}) = 3 \). Therefore, \( IScore^t(S_1 \cup \{s\}) - IScore^t(S_1) = 0 \) and \( IScore^t(S_2 \cup \{s\}) - IScore^t(S_2) = 1 \) which shows \( S_1 \subseteq S_2 \) does not imply \( IScore^t(S_2 \cup \{s\}) - IScore^t(S_2) \leq IScore^t(S_1 \cup \{s\}) - IScore^t(S_1) \) and proves the claim.

\[ \Box \]

\section*{B.3 Extensions}

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

In Section 4.3.4, 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 4.4. The change is that the marginal pair count (\( MPCount \)) in Equation 4.4 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 B.1 where we also prove it is sub-modular. The following example illustrates the problem of summarizing the differences in the provenance of two tuples:

\begin{example}
In Table 4.2, consider the provenance of \( a_4, a_6 \in Q_2(MoviesDirectors) \). A rule \( s_5 = (Title : \ast, Year : \ast, Genre : Comedy, Rating : 7, Rev : \ast, Director : \ast, Gender : Male, Country : US) \) provides a comparative summary for the differences between the provenance of \( a_4 \) and \( a_6 \) since the rule summarizes tuples in the provenance of \( a_6 \) but it does not cover any tuple from the provenance of \( a_4 \). \[ \Box \]
\end{example}
In [B.1] we presented 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 4.3.4 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.

### B.4 User survey materials

The following is a copy of the user survey questions. The survey itself was given using an online tool.

**Introduction.**

The purpose of this questionnaire is to study the user needs for a data summarization application. The goal is to find out which utility functions the users value the most as well as the preferred size of a summary of tabular data. Our target demographic are users who have had at least an undergraduate course in databases or equivalent external experience.

**Demographic information.**

**Age group:**
- 19-29
- 30-39
- 40-49
- More than 50

**Gender:**
- Male
Female
Other
Prefer not to say

**Education.** Highest level of education completed:
- High school
- BSc or equivalent
- MSc or equivalent
- PhD or equivalent
- Other:

**Experience with data.** How do you describe your experience working with data analysis or data management technologies?
- None at all
- A little
- A moderate amount
- A lot
- A great deal

**Tools.** What data technologies are you familiar with?
- SQL
- R
- Python
- Excel
- Other:

**Use.** How often do you use data related technologies (data analysis tools, spreadsheets, databases) in your work?
- Always
- Most of the time
- About half the time
- Sometimes
- Never

**Summarization scenario.**
A user accesses a dataset about movies and directors. The user asks the system for the total revenue for action movies made in the year 2017. The user gets the result back:

Action; 2017; 7.3 Billion. Now, given this result we would like to provide the user of a summary of movies made in 2017 that made this 7.3 billion figure. Here is an example of a summary that is of length 3. It contains 3 rows:

<table>
<thead>
<tr>
<th>Genre</th>
<th>Year</th>
<th>Country</th>
<th>Duration</th>
<th>Language</th>
<th>coverage</th>
<th>impact</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>2017</td>
<td>US</td>
<td>*</td>
<td>English</td>
<td>35%</td>
<td>85%</td>
</tr>
<tr>
<td>Action</td>
<td>2017</td>
<td>*</td>
<td>&gt; 120</td>
<td>*</td>
<td>25%</td>
<td>78%</td>
</tr>
<tr>
<td>Action</td>
<td>2015</td>
<td>*</td>
<td>&lt; 90</td>
<td>*</td>
<td>39%</td>
<td>21%</td>
</tr>
</tbody>
</table>

Table B.1: Summary A

This summary is presented with real values for their respective columns and * value as do-not-care values filling in for any other value in the column. For example, row 1 is action movies from 2017 made in the US with all possible duration values in English. Those movies make up 35% of rows in the original database and have 85% of total revenue.

Q1. Can you rank the rows in the summary from most interesting (1) to least interesting (3)?
   Row 1
   Row 2
   Row 3

Can you elaborate on why you picked this order?

Q2. Please rank the following statements according to their importance level when looking at a summary.
   The right length (number of rows) in a summary

   1. Very important
   2. important

168
3. Neutral
4. important
5. Not important at all

Showing groups of high grossing movies
1. Very important
2. important
3. Neutral
4. important
5. Not important at all

Showing groups that contain a lot of movies
1. Very important
2. important
3. Neutral
4. important
5. Not important at all

Showing a summary that covers the whole table
1. Very important
2. important
3. Neutral
4. important

5. Not important at all

Highlighting groups with surprising information

1. Very important
2. important
3. Neutral
4. important
5. Not important at all

Q3. The system will give you a summary overview of the data. This overview will give you an immediate look at the interesting aspects of the data. How long do you think this summary should be?

1-4 rows
5-8 rows
9-12 rows
13-16 rows
17 or more rows
Other:
Can you please elaborate on why you picked this length?

Closing.

Q1. How accurate do you feel you were about your answers?
Less than I would like
About right

Q2. Is there any other information that you would like to share regarding your needs for a data summarization system?
Appendix C

Aggregate query refinement

C.1 User study materials

C.1.1 Tasks

The following are the tasks given to the users:

**Task 1:** Query 1: “SELECT count(*) FROM movie, genre WHERE id = movie_id AND revenue > 100000000 AND vote_average >= 7 AND genre = 'Action'”

Explanation: Select all Action movies that have total revenue > $100m, and have a vote_average (rating) >= 7

The result is 91 movies. Assuming you want to expand the query to include more movies and get a bigger number, how would you go about it?

We have a list of suggestions by our system we would like you to see. Look at the suggestions and choose the most relevant tuples based on your examination of results and query. Feel free to ask questions and talk through the process. When you are finished, you can hit submit.

**Task 2:** Query 2: “SELECT count(*) FROM movie, genre WHERE id = movie_id AND revenue > 100000000 AND vote_average >= 7 AND genre
= 'Adventure'

Explanation: Select all Adventure movies that have total revenue > $100m, and have a vote_average (rating) >= 7

The result is 87 movies. Assuming you want to expand the query to include more movies and get a bigger number, how would you go about it?

We have a list of suggestions by our system we would like you to see. Look at the suggestions and choose the most relevant tuples based on your examination of results and query. Feel free to ask questions and talk through the process. When you are finished, you can hit submit.

Task 3: Query 3: “SELECT count(*) FROM movie m, director d
WHERE m.dir_id = d.id AND m.revenue > 100000000 AND d.gender = 2 AND release_date > 2015”

Explanation: Select all movies that have total revenue > $100m, that are made by male directors and released after the year 2015

The result is 70 movies. Assuming you want to expand the query to include more movies and get a bigger number, how would you go about it?

We have a list of suggestions by our system we would like you to see. Look at the suggestions and choose the most relevant tuples based on your examination of results and query. Feel free to ask questions and talk through the process. When you are finished, you can hit submit.

Task 4: Query 4: “SELECT count(*) FROM movie m, director d
WHERE m.dir_id = d.id AND m.revenue > 1000000 AND d.gender = 1 AND release_date > 2015”

Explanation: Select all movies that have total revenue > $1m, that are made by female directors and released after the year 2015

The result is 20 movies. Assuming you want to expand the query to include more movies and get a bigger number, how would you go about it?

We have a list of suggestions by our system we would like you to see.
Look at the suggestions and choose the most relevant tuples based on your examination of results and query. Feel free to ask questions and talk through the process. When you are finished, you can hit submit.

### C.1.2 Questionnaires and interfaces

Figure C.1 shows the demographic questions. Figure C.2 shows the tasks interface. Figure C.3 shows the exit questions.

### C.1.3 Suggestions

Figures C.4 and C.5 show the suggestions made within the interface. Tables C.1 and C.2 show the full list of suggestions for both methods.
Figure C.2: Tasks interface for the query refinement study

Figure C.3: Exit questionnaire for the query refinement study
Figure C.4: Individual movie suggestions (PULearn)

<table>
<thead>
<tr>
<th>Option</th>
<th>Id</th>
<th>Proba</th>
<th>Title</th>
<th>Language</th>
<th>Revenue</th>
<th>Genre</th>
<th>Release_date</th>
<th>Vote_average</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>281,957</td>
<td>0.83</td>
<td>The Revenant</td>
<td>en</td>
<td>532,950,503</td>
<td>Western</td>
<td>2015</td>
<td>7.3</td>
</tr>
<tr>
<td></td>
<td>296,098</td>
<td>1.1</td>
<td>Bridge of Spies</td>
<td>en</td>
<td>165,478,348</td>
<td>Thriller</td>
<td>2015</td>
<td>7.2</td>
</tr>
<tr>
<td></td>
<td>679</td>
<td>1.59</td>
<td>Aliens</td>
<td>en</td>
<td>183,316,455</td>
<td>Horror</td>
<td>1986</td>
<td>7.7</td>
</tr>
<tr>
<td></td>
<td>6844</td>
<td>0.58</td>
<td>The Ten Commandments</td>
<td>en</td>
<td>122,700,000</td>
<td>Drama</td>
<td>1956</td>
<td>7.4</td>
</tr>
<tr>
<td></td>
<td>37,799</td>
<td>0.89</td>
<td>The Social Network</td>
<td>en</td>
<td>224,920,315</td>
<td>Drama</td>
<td>2010</td>
<td>7.1</td>
</tr>
<tr>
<td></td>
<td>452</td>
<td>0.80</td>
<td>The Usual Suspects</td>
<td>en</td>
<td>190,092,277</td>
<td>Drama</td>
<td>2003</td>
<td>8.0</td>
</tr>
</tbody>
</table>

Figure C.5: Rules suggestions (ProvSum)
<table>
<thead>
<tr>
<th>Task ID</th>
<th>Title</th>
<th>Language</th>
<th>Rating</th>
<th>Director</th>
<th>Year</th>
<th>Genre</th>
<th>Revenue</th>
<th>Release Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Pitch Perfect 2</td>
<td>en</td>
<td>0.94</td>
<td>Leslye Headland</td>
<td>2015</td>
<td>Action, Comedy</td>
<td>3566225</td>
<td>2015-09-12</td>
</tr>
<tr>
<td>2</td>
<td>The To Do List</td>
<td>en</td>
<td>0.9</td>
<td>Michael Dowse</td>
<td>2015</td>
<td>Comedy, Drama</td>
<td>127380</td>
<td>2015-08-07</td>
</tr>
<tr>
<td>3</td>
<td>Pitch Perfect</td>
<td>en</td>
<td>0.81</td>
<td>Elizabeth银行</td>
<td>2015</td>
<td>Comedy, Drama</td>
<td>210860</td>
<td>2015-09-19</td>
</tr>
<tr>
<td>4</td>
<td>A United Kingdom</td>
<td>en</td>
<td>1.01</td>
<td>Mike Banky</td>
<td>2015</td>
<td>Thriller</td>
<td>474171806</td>
<td>2015-09-12</td>
</tr>
</tbody>
</table>

Table C.1: PULearn suggestions for each task
<table>
<thead>
<tr>
<th>Task</th>
<th>Language</th>
<th>Genre</th>
<th>Year</th>
<th>Country</th>
<th>Company</th>
<th>Rating</th>
<th>Gender</th>
<th>Runtime</th>
<th>Revenue</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>*</td>
<td>Action</td>
<td>&gt;2015</td>
<td></td>
<td></td>
<td>&gt;= 7</td>
<td>*</td>
<td>&gt;90m</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>1</td>
<td>*</td>
<td>Action &amp; Western</td>
<td>&gt;2015</td>
<td></td>
<td></td>
<td>&gt;= 7</td>
<td>*</td>
<td>&gt;100m</td>
<td>99</td>
<td>102</td>
</tr>
<tr>
<td>1</td>
<td>*</td>
<td>Action &amp; Horror</td>
<td>&gt;2015</td>
<td></td>
<td></td>
<td>&gt;= 7</td>
<td>*</td>
<td>&gt;100m</td>
<td>114</td>
<td>184</td>
</tr>
<tr>
<td>1</td>
<td>*</td>
<td>Action &amp; Adventure</td>
<td>&gt;2015</td>
<td></td>
<td></td>
<td>&gt;= 7</td>
<td>*</td>
<td>&gt;100m</td>
<td>100</td>
<td>181</td>
</tr>
<tr>
<td>1</td>
<td>*</td>
<td>Action &amp; War</td>
<td>&gt;2015</td>
<td></td>
<td></td>
<td>&gt;= 7</td>
<td>*</td>
<td>&gt;100m</td>
<td>115</td>
<td>114</td>
</tr>
<tr>
<td>1</td>
<td>*</td>
<td>Action &amp; Adventure</td>
<td>&gt;2015</td>
<td></td>
<td></td>
<td>&gt;= 7</td>
<td>*</td>
<td>&gt;100m</td>
<td>181</td>
<td>269</td>
</tr>
<tr>
<td>1</td>
<td>*</td>
<td>Action &amp; Adventure</td>
<td>&gt;2015</td>
<td></td>
<td></td>
<td>&gt;= 6</td>
<td>*</td>
<td>&gt;100m</td>
<td>59</td>
<td>94</td>
</tr>
<tr>
<td>1</td>
<td>*</td>
<td>Action &amp; Adventure</td>
<td>&gt;2015</td>
<td></td>
<td></td>
<td>&gt;= 6</td>
<td>*</td>
<td>&gt;50m</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>1</td>
<td>*</td>
<td>Action &amp; Adventure</td>
<td>&gt;2015</td>
<td></td>
<td></td>
<td>&gt;= 6</td>
<td>*</td>
<td>&gt;100m</td>
<td>14</td>
<td>424</td>
</tr>
<tr>
<td>1</td>
<td>*</td>
<td>Action &amp; Adventure</td>
<td>&gt;2015</td>
<td></td>
<td></td>
<td>&gt;= 6</td>
<td>*</td>
<td>&gt;100m</td>
<td>179</td>
<td>179</td>
</tr>
<tr>
<td>1</td>
<td>*</td>
<td>Action &amp; Adventure</td>
<td>&gt;2015</td>
<td></td>
<td></td>
<td>&gt;= 6</td>
<td>*</td>
<td>&gt;100m</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>1</td>
<td>*</td>
<td>Action &amp; Adventure</td>
<td>&gt;2015</td>
<td></td>
<td></td>
<td>&gt;= 6</td>
<td>*</td>
<td>&gt;1m</td>
<td>27</td>
<td>27</td>
</tr>
<tr>
<td>1</td>
<td>*</td>
<td>Action &amp; Adventure</td>
<td>&gt;2015</td>
<td></td>
<td></td>
<td>&gt;= 6</td>
<td>*</td>
<td>&gt;1m</td>
<td>45</td>
<td>45</td>
</tr>
<tr>
<td>1</td>
<td>*</td>
<td>Action &amp; Adventure</td>
<td>&gt;2015</td>
<td></td>
<td></td>
<td>&gt;= 6</td>
<td>*</td>
<td>&gt;1m</td>
<td>44</td>
<td>44</td>
</tr>
<tr>
<td>1</td>
<td>*</td>
<td>Action &amp; Adventure</td>
<td>&gt;2015</td>
<td></td>
<td></td>
<td>&gt;= 6</td>
<td>*</td>
<td>&gt;1m</td>
<td>391</td>
<td>391</td>
</tr>
<tr>
<td>1</td>
<td>*</td>
<td>Action &amp; Adventure</td>
<td>&gt;2015</td>
<td></td>
<td></td>
<td>&gt;= 6</td>
<td>*</td>
<td>&gt;1m</td>
<td>895</td>
<td>895</td>
</tr>
</tbody>
</table>

Table C.2: ProvSum suggestions for each task