# QUERY PROCESSING OVER INCOMPLETE AUTONOMOUS WEB DATABASES

by

Hemal Khatri

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

ARIZONA STATE UNIVERSITY

August 2006

QUERY PROCESSING OVER INCOMPLETE AUTONOMOUS WEB DATABASES

by

Hemal Khatri

has been approved

July 2006

APPROVED:

, Chair

Supervisory Committee

ACCEPTED:

Department Chair

Dean, Division of Graduate Studies

## ABSTRACT

Incompleteness due to missing attribute values (aka “null values”) is very common in autonomous web databases, on which user accesses are usually supported through mediators. Traditional query processing techniques that focus on the strict soundness of answer tuples often ignore tuples with critical missing attributes, even if they wind up being relevant to the user query. Ideally, the mediator is expected to retrieve such relevant uncertain answers and gauge their relevance by accessing their likelihood of being relevant answers to the query. The autonomous nature of the databases poses several challenges in realizing this idea. Such challenges include restricted access privileges, limited query patterns and cost sensitivity of database and network resource consumption in web environment. This thesis presents *QPIAD* – a framework for query processing over incomplete autonomous databases. *QPIAD* is able to retrieve relevant uncertain answers with high precision, high recall and manageable cost. Data integration over multiple autonomous data sources is an important task performed by a mediator. This thesis describes query rewriting techniques to perform data integration over multiple incomplete autonomous data sources on the web. Results of experimental evaluation on real-life databases demonstrate that our system retrieve relevant answers with high precision and manageable cost.

To my parents

## ACKNOWLEDGMENTS

I would be restating the obvious when I say Prof. Subbarao Kambhampati is a great advisor. I consider myself truly fortunate to have him as my advisor and would like to thank him sincerely for the opportunity to work under his guidance. I am deeply grateful to him for showing immense patience during the long and frustrating phase of research problem identification and the change of my research area from Automated Planning to Database/Information Integration. He has not only helped me in the development of ideas and solutions but also in careful presentation of them. I am also thankful to his constructive criticism on various drafts of research papers which helped me improve my writing skills.

I would also like to thank Prof. Yi Chen for helping me tremendously in making steady progress in my thesis work. I benefited significantly from her counsel and guidance in many important situations. I enjoyed the rewarding experience of having worked closely with her on interesting problems. I would also like to thank other members of my committee Prof. Chitta Baral and Prof. Huan Liu for providing helpful comments.

My special thanks to Jianchun Fan for being a great friend and collaborator. I will always remember the discussions we had on research problems during the database meetings. I would like to thank other member of DB-Yochan group Ullas Nambiar, Bhaumik Chokshi and Garrett Wolf for their valuable comments and support in my research. I also acknowledge all the AI Lab members who volunteered to take part in the user study and helped me find enough users to make the user study possible.

Last, but not the least, I thank my parents for motivating me constantly in my life and encouraging me to strive for higher goals in life.

TABLE OF CONTENTS

Page

LIST OF TABLES . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ix

LIST OF FIGURES . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . x

CHAPTER 1 INTRODUCTION . . . . . . . . . . . . . . . . . . . . . . . . . . 1

1.1. Query Processing over Incomplete Autonomous Databases . . . . . . . . . . 3

1.2. Data Integration over Incomplete Autonomous Databases . . . . . . . . . . 5

1.3. Thesis Contributions . . . . . . . . . . . . . . . . . . . . . . . . . . 7

1.4. Outline . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7

CHAPTER 2 CHALLENGES IN HANDLING INCOMPLETENESS OVER AUTONOMOUS DATABASES . . . . . . . . . . . . . . . . . . . . . . . . . . 9

CHAPTER 3 QPIAD QUERY PROCESSING OVER INCOMPLETE AUTONOMOUS DATABASES . . . . . . . . . . . . . . . . . . . . . . . . . . 13

3.1. Preliminaries and Overview of Solution . . . . . . . . . . . . . . . . . . 13

3.2. Retrieving Relevant Incomplete Information . . . . . . . . . . . . . . . . 17

3.2.1. Retrieving Relevant Uncertain Answers from Local Databases . . . . 18

3.2.2. Retrieving Relevant Uncertain Answers from Autonomous Databases 20

3.2.3. Algorithm for RRUA Approach . . . . . . . . . . . . . . . . . . . 23

3.3. Learning Attribute Correlations and Value Probability Distributions . . . . 26

3.3.1. Learning Attribute Correlations by Mining Approximate Functional Dependencies(AFDs) . . . . . . . . . . . . . . . . . . . . . . . 26

3.3.2. Learning Value Distributions using Classifiers . . . . . . . . . . . . 28

3.3.3. Combining AFDs and Classifiers . . . . . . . . . . . . . . . . . . 29

Page

3.3.4. Learning Value Distributions Online from Base Result Set . . . . . . 31

3.4. Empirical Evaluation for QPIAD . . . . . . . . . . . . . . . . . . . . . . . . . . . 33

3.4.1. Implementation and User Interface . . . . . . . . . . . . . . . . . . . . 33

3.4.2. Experimental Aims & Settings . . . . . . . . . . . . . . . . . . . . . . . 34

3.4.3. Accuracy of Classifiers . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 35

3.4.4. Comparing RRUA with AUA and RUA . . . . . . . . . . . . . . . . 38

3.4.5. Effect of Confidence Threshold on Precision . . . . . . . . . . . . . . 42

3.4.6. Ranking the Rewritten Queries . . . . . . . . . . . . . . . . . . . . . . 43

3.4.7. Robustness of the RRUA Approach . . . . . . . . . . . . . . . . . . . 44

3.4.8. Comparison of RRUA with O-RRUA . . . . . . . . . . . . . . . . . . 46

CHAPTER 4 DATA INTEGRATION OVER INCOMPLETE AUTONOMOUS DATABASES . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 48

4.1. Leveraging Correlations among Data Sources . . . . . . . . . . . . . . . . . 48

4.1.1. Motivation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 48

4.1.2. Retrieving Relevant Answers from Data Sources Not Supporting the Query Attribute . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 49

4.1.3. Empirical Evaluation of using Correlation Between Data Sources . . 52

4.2. Handling Joins over Incomplete Databases . . . . . . . . . . . . . . . . . . . 54

4.2.1. Motivation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 54

4.2.2. Issues in Handling Joins over Incomplete Databases . . . . . . . . . 56

4.2.3. Joins over Incomplete Databases . . . . . . . . . . . . . . . . . . . . . 57

4.2.4. Empirical Evaluation of Joins over Incomplete Databases . . . . . . 60

Page

CHAPTER 5 RELATED WORK . . . . . . . . . . . . . . . . . . . . . . . . . . . . 63

CHAPTER 6 DISCUSSION AND FUTURE WORK . . . . . . . . . . . . . . . . . . 66

6.1. Handling General Queries . . . . . . . . . . . . . . . . . . . . . . . . . . . . 66

6.2. Supporting Imprecise Queries over Incomplete Autonomous Databases . . . 67

CHAPTER 7 CONCLUSION . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 69

REFERENCES . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 71

# LIST OF TABLES

Table Page

1. Statistics on missing values in web databases . . . . . . . . . . . . . . . . . . 3
2. Fragment of a Car Database . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3
3. Fragment of a Car Database . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9
4. Domain size of attributes in the Cars database . . . . . . . . . . . . . . . . . 35
5. Domain size of attributes in the Census database . . . . . . . . . . . . . . . . 35
6. AFDs mined for the Cars database . . . . . . . . . . . . . . . . . . . . . . . . 36
7. Comparison of null value prediction accuracy across different AFD-enhanced classifiers . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 37
8. Test queries on the global schema on *body style* attribute . . . . . . . . . . 53
9. Fragment of a Car Database . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 68

## LIST OF FIGURES

Figure Page

1. Global schema and local schema of data sources . . . . . . . . . . . . . . . . 6
2. QPIAD System architecture. . . . . . . . . . . . . . . . . . . . . . . . . . . 14
3. Comparison of prediction accuracy across AFD-enhanced NBC classifier and bayes network classifier over different samples sizes of the Car database. . . 37
4. Precision-Recall curves for $\sigma_{body\ style=convt}(Cars)$. . . . . . . . . . . . . . . . 39
5. Precision-Recall curves for $\sigma_{education=bachelors}(Census)$. . . . . . . . . . . . 39
6. Precision curves for $\sigma_{body\ style=convt}(Cars)$. . . . . . . . . . . . . . . . . . . 41
7. Precision curves for $\sigma_{education=bachelors}(Census)$. . . . . . . . . . . . . . . . 41
8. Average Precision for various confidence thresholds(Cars). . . . . . . . . . . 42
9. Accumulated precision curve for 30 test queries on Car database. . . . . . . 43
10. Accumulated precision curve for 30 test queries on Census database. . . . . 44
11. Accumulated precision curve for $\sigma_{body\ style=convt}$ with different sample sizes on Car database. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 45
12. Accumulated precision curve for $\sigma_{workshop=private}$ with different sample sizes on Census database. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 45
13. Comparison of accumulated precision across $RRUA$ and $O{-}RRUA$ over 40 test queries on the Car database . . . . . . . . . . . . . . . . . . . . . . . 46
14. Comparison of accumulated precision across $RRUA$ and $O{-}RRUA$ zoomed over the top 10 rewritten queries . . . . . . . . . . . . . . . . . . . . . . . 47
15. Global schema and local schema of data sources . . . . . . . . . . . . . . . . 49
16. Precision curve for top $K$ tuples retrieved from *Yahoo! Autos* using AFDs learned from *Cars.om* . . . . . . . . . . . . . . . . . . . . . . . . . . . . 53

Figure Page

17. Precision curve for top $K$ tuples retrieved from *CarsDirect* using AFDs learned from *Cars.com* . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 54

18. Autonomous sources $S_1(Cars)$ and $S_2(Review)$ . . . . . . . . . . . . . . . . 55

19. Mediator view over sources $S_1$ and $S_2$ . . . . . . . . . . . . . . . . . . . . . . 55

20. Precision-Recall curve for the query $\sigma_{model=Civic}$ on global schema $R$ . . . . 61

21. Precision-Recall curve for the query $\sigma_{make=Audi}$ on global schema $R$ . . . . 62

22. Precision-Recall curve for the query $\sigma_{Rating=4}$ on global schema $R$ . . . . . 62

CHAPTER 1

# INTRODUCTION

With the advent of Web, information has been made more readily and easily available from a variety of data sources. For example, there are large number of web databases containing used cars or houses available for sale[7, 41, 34, 18, 3]. Data sources on the Web, unlike local relational databases, are evolving in an autonomous manner. Most web data sources provide a form based interface which allows a user to query the database and retrieve results matching the query constrained attributes they provided. Such databases on which the query processor only has read-only access without any capability to write or modify the databases are called as *autonomous databases.*

Data integration in autonomous web database scenarios has drawn much attention in recent years, as more and more data becomes accessible via web servers which are supported by back-end databases. A mediator provides a unified query interface as a global schema of the underlying databases. Queries on the global schema are then rewritten as queries over autonomous databases through their web interfaces. Current mediator systems [25, 22] return to user only *certain answers* that exactly satisfy all the user query predicates. Tuples that are otherwise highly relevant for the query will not be retrieved if they have null values on any of the query predicates. For example, in a used car trading application, if a user asks for convertible cars, all the returned answers must have the value "convt" for the attribute *body style*. Even though all Z4's are convertible, a BMW Z4 car which has a null value in its *body style* will not be returned. Unfortunately, such an approach is both inflexible and inadequate for querying many autonomous web databases many of which are inherently incomplete for the following reasons:

**Incomplete Entry:** Web databases are often input by lay individuals without any central curation. For example, websites such as *Cars.com* and *Yahoo! Autos*, obtain car information from individual car owners who may not provide complete information for their cars, resulting in a lot of missing values (aka "null" values) in the databases. In the example above, the owner of the BMW Z4 may have skipped filling the *body style* attribute assuming that it is obvious (just as the owner of an Accord may skip entering the *make* to be Honda). As a result, this car won't be retrieved by current mediators for queries on "*body style=convt*".[1]

**Extraction Inaccuracy:** Many web databases are being populated using automated information extraction techniques. As a result of the inherent imperfection of the extraction, these web databases may contain missing values.

**Schema Heterogeneity:** The global schema provided by a mediator may often contain attributes that do not appear in some local schemas. For example, a global schema for used car trading may have an attribute called *body style*, which is supported in *Cars.com*, but not in *Yahoo! Autos*. Given a query on the global schema that selects cars having *body style* equal to "convertible", approaches that only return certain answers won't be able to return car information from *Yahoo! Autos*.

Table 1 shows statistics on the percentage of incomplete tuples on two autonomous

[1]This type of incompleteness is expected to increase even more with services such as GoogleBase which provide users significant freedom in deciding which attributes to define and/or list.

| **Website** | **# of Attributes** | **Total tuples** | **Incomplete tuples** | **Body style** | **Engine** |
|---|---|---|---|---|---|
| www.AutoTrader.com | 13 | 25127 | 33.67% | 3.6% | 8.1% |
| www.CarsDirect.com | 14 | 32564 | 98.74% | 55.7% | 55.8% |

Table 1. Statistics on missing values in web databases

| **ID** | **Make** | **Model** | **Year** | **Body style** |
|---|---|---|---|---|
| 1 | Audi | A4 | 2001 | convt |
| 2 | BMW | Z4 | 2002 | convt |
| 3 | Porsche | Boxster | 2005 | convt |
| 4 | BMW | Z4 | 2003 | null |
| 5 | Honda | Civic | 2004 | null |
| 6 | Toyota | Camry | 2002 | sedan |

Table 2. Fragment of a Car Database

web databases. The statistics were computed from a probed sample. The table also gives statistics on the percentage of missing values for the *body style* and *engine* attributes. These statistics show that incompleteness can indeed creep into online databases.

## 1.1. Query Processing over Incomplete Autonomous Databases

When faced with such incomplete web databases, current mediators provide only certain answers thereby sacrificing recall. This is particularly problematic when the data sources have a significant fraction of incomplete tuples, and/or the user requires high recall (consider, for example, a law-enforcement scenario, where a potential crime suspect goes unidentified because of information that is fortuitously missing in the database).

A naïve approach for improving the recall would be to return, in addition to all the certain answers, all the tuples with missing values on the constrained attribute(s). For example, consider a selection query $Q'$ for cars having *body style=convt* on a fragment of a Car database as shown in Table 2. For the above query $Q'$, a mediator could return not only the tuples $t_1$, $t_2$, $t_3$ whose *body style* values are "convt" but also the tuples $t_4$ and $t_5$ whose *body style* values are missing(null). This approach of returning *all uncertain* *answers* referred to as AUA has two obvious drawbacks. First, it is infeasible to retrieve all tuples with nulls in some cases, as web databases usually do not allow the mediator to directly retrieve tuples with null values on specific attributes. Second, and perhaps more important, many tuples with missing values on constrained attributes are *irrelevant* to the query. Intuitively, not all the tuples that have a null in the attribute *body style* represent convertible cars! The AUA approach thus improves recall but suffers from low precision and high cost.

**Our Approach:** In this thesis, we focus on query processing techniques for incomplete autonomous databases, that not only return certain answers but also return, in a ranked fashion, tuples that have missing values and yet are highly relevant to queries. The goal of our query processing framework *QPIAD*[2] is to return query answers with good precision, recall and manageable cost.

Our approach starts by adapting standard techniques to predict missing values. Given techniques to predict values for null, one obvious way of exploiting them would be to store the predicted values in databases and then process queries on them directly. Indeed this is advocated by some existing research efforts [21, 26, 31, 37]. Unfortunately, such an approach is not appropriate for querying incomplete autonomous web databases. Since a mediator usually does not have update capabilities over the autonomous databases, it cannot directly replace the null values.

A more plausible solution is to first retrieve all the tuples with nulls on constrained attributes, predict missing values for them, and then decide relevant query answer set. However, this approach may be inappropriate as discussed before due to limited query access pattern of web databases and high network transmission costs.

[2]*QPIAD* is an acronym for Query(Q) Processing(P) over Incomplete(I) Autonomous(A) Databases(D) pronounced as *kyōōpēăd′*.

To handle this challenge, we propose online query rewriting techniques that not only retrieve certain answers to a query, but also highly relevant answers that have nulls on constrained attributes. These latter answers, termed as *relevant uncertain answers*, are retrieved without modifying the underlying data sources. When a query is submitted, the mediator first retrieves all the certain answers for the given user query. Then based on the certain answers and a set of mined attribute correlation rules, the mediator forms a group of rewritten queries to be sent to the data sources. The rewritten queries are then ranked based on their likelihood of bringing back relevant answers before being posed to the data sources. Using missing value prediction techniques in the context of such query rewriting approach poses some unique challenges such as how to generate and rank the rewritten queries. To handle these, we present a missing value prediction strategy that uses Approximate Functional Dependencies(AFDs) and Naïve Bayesian Classifiers(NBC). In the car database example, this analysis may allow *QPIAD* to identify that *model* determines *body style* and the fact that $Z4$ cars in the database often seem to be convertibles. *QPIAD* then rewrites the query on *body style=convt* into additional selection queries such as *model=*$Z4$ to retrieve relevant uncertain tuples such as $t_4$.

## 1.2. Data Integration over Incomplete Autonomous Databases

Data integration over autonomous web databases is an important task performed by mediators in web scenarios. In such scenarios, the mediator provides a global view over the local schema of underlying data sources.

The mediator supports queries on attributes in the global schema which are often not supported in the local schema of some individual data sources. For example, consider a global schema $GS_{UsedCars}$ supported by the mediator over the sources *Cars.com*[7] and *Ya-*

*hoo! Autos*[41] as shown in Figure 1. Since the form based interface of *Yahoo! Autos* doesn't support queries on *body style* attribute, the mediator cannot directly query the database in order to retrieve cars having a specific *body style*. Given a query $Q\colon \sigma_{body\ style=coupe}$ on global schema that selects cars of *body style* "coupe", approaches that only return certain answers won't be able to return car information from *Yahoo! Autos*. Hence many relevant cars from *Yahoo! Autos* wouldn't be shown to the user. In order to retrieve ranked rele-

| | |
|---|---|
| Mediator | $GS_{UsedCars}(Make, Model, Year, Price, Mileage, Location, Body\ style)$ |
| Cars.com | $LS_{Autotrader}(Make, Model, Year, Price,\ Mileage, Location, Body\ style)$ |
| Yahoo! Autos | $LS_{Cars}(Make, Model, Year, Price,\ Mileage, Location)$ |

Figure 1. Global schema and local schema of data sources

vant uncertain tuples from such data sources, we use query rewriting techniques based on attribute correlations and value distributions learned from other data sources supporting the query attribute in their local schema.

Many queries issued on the global schema of the mediator may require performing joins over individual data sources. For example, consider two data sources having local schema as $S_1(Make, Model, Year)$ and $S_2(Model, Ratings)$ and global schema provided by the mediator as $GS(Make, Model, Year, Ratings)$. Consider a user query asking for all cars made by $Honda$ along with their $ratings$. If both sources $S_1$ and $S_2$ are complete, then the mediator would first retrieve all tuples having $make{=}Honda$ from source $S_1$. Then it would retrieve tuples from source $S_2$ corresponding to the *models* of the tuples returned from source $S_1$. Finally, it would combine the tuples from the two sources using the join attribute *model* and return the results to the users. If the sources are incomplete, then above approach is not able to retrieve join results for car tuples having missing *make* in source $S_1$ but still might be relevant to the user query. Also, if the join attribute *model* is missing for some tuples having $make{=}Honda$ in source $S_1$, the mediator cannot join that tuple with a ratings tuple from source $S_2$. We present query rewriting techniques to perform joins over incomplete autonomous databases in order to present ranked uncertain join results for the user query.

## 1.3. Thesis Contributions

This thesis presents a framework for query processing over incomplete autonomous web databases under a variety of circumstances. The major contributions of this thesis are as follows:

- To the best of our knowledge, our framework is the first that retrieves relevant but uncertain answers with missing values on constrained attributes without modifying the underlying databases. Consequently, it is suitable for querying incomplete autonomous databases. The idea of using attribute correlations and predicted value distributions on missing values to rewrite queries is also a novel contribution of our work. Our experimental evaluations show that our system retrieves the most relevant information with highly manageable query evaluation cost.
- We also propose techniques to leverage correlations between data sources to retrieve relevant uncertain answers from data sources not supporting the query attributes. We also propose techniques to perform joins over incomplete autonomous databases.

## 1.4. Outline

The rest of the thesis is organized as follows. In the next chapter, we describe the challenges involved in supporting query processing under imprecision and incompleteness over autonomous web databases. In Chapter 3, we describe our query rewriting and value distribution learning techniques which are needed in order to support query processing over incomplete autonomous databases. In Chapter 4, we describe data integration over incomplete sources by supporting join queries over incomplete sources and by leveraging correlation among data sources to answer queries on attributes which are not supported by the local data source schema. Chapter 5 discusses the related work on handling incompleteness in databases. In Chapter 6, we present a discussion and describe future work. Finally, we conclude the thesis in Chapter 7.

CHAPTER 2

# CHALLENGES IN HANDLING INCOMPLETENESS OVER AUTONOMOUS DATABASES

In this chapter we describe the challenges involved in query processing under imprecision and incompleteness over autonomous web databases.

Consider a precise user query[1] $Q'$: $\sigma_{body\ style=convt}$ on the fragment of an online Car database shown in Table 3. Current query processing techniques only retrieve tuples that exactly satisfy the user query $Q'$. Using these techniques, only tuples $t_1$,$t_2$, and $t_3$ would be retrieved based on the user query. However, it is quite possible that the entities represented by incomplete tuples $t_4$ may indeed be convertible cars. Current query processing techniques fail to retrieve such tuples as a possible answer to the user query. We propose a query processor capable of retrieving such possibly relevant tuples and ranking them in the order

[1]A precise query is the one in which the user knows exactly what he is looking for.

| ID | Make | Model | Year | Body style |
|---|---|---|---|---|
| 1 | Audi | A4 | 2001 | convt |
| 2 | BMW | Z4 | 2002 | convt |
| 3 | Porsche | Boxster | 2005 | convt |
| 4 | BMW | Z4 | 2003 | null |
| 5 | Honda | Civic | 2004 | null |
| 6 | Toyota | Camry | 2002 | sedan |

Table 3. Fragment of a Car Database

of their likelihood of being a relevant answer to the user query.

There are several challenges involved in supporting query processing over an incomplete autonomous database:

1. We need to develop techniques for predicting null values for autonomous databases. This becomes more complicated in autonomous databases, as the mediator doesn't have access to the entire database and cannot write the predicted value back to the database.

2. Since the mediator does not have access to the entire autonomous database, it has to use a sample of the entire database for predicting missing values. The techniques for predicting missing values have to be robust so that they can be applied for tuples not present in the sample. In other words, using only attribute value correlations from the tuples in the sample would not be effective for predicting missing values for autonomous database.

3. Most autonomous databases do not support binding for null values in their web interfaces. Hence, we need to develop query rewriting techniques in order to retrieve tuples containing null values which might be relevant to a given user query.

4. The mediator has to employ appropriate learning techniques based on the sample database in order to effectively predict null values for possible relevant tuples corresponding to a query. In order to improve the classification accuracy, the mediator can use certain results(without any null values) retrieved during query time to predict missing values for other tuples containing null values.

5. In order to present ranked results to a user query, we need to develop techniques for ranking tuples containing missing values in terms of their relevance to the user query.

Specifically, we need to develop appropriate ranking criteria as well as techniques to assign ranks to tuples containing null values.

6. In order to achieve our goal of returning tuples with good precision, recall and manageable cost, the techniques developed should have high prediction accuracy and should minimize network traffic while retrieving possible relevant tuples containing null values.

7. Since the query processor presents tuples that do not correspond to exact answers to user query, it needs to provide explanations and justifications to gain the user's trust.

In addition to the above mentioned challenges, when it comes to supporting query processing over *multiple* incomplete autonomous databases, we need to address the following issues:

1. For data aggregation across multiple incomplete autonomous databases, the global schema may contain certain attributes which may not be supported by the local schema of some databases. When the query involves an attribute which is not supported by a data source, we need to retrieve ranked relevant results from such databases in addition to certain answers from other databases.

2. Because the mediator doesn't have access over the entire database, supporting joins over multiple heterogenous incomplete databases becomes particularly challenging. As the mediator doesn't have access over entire database, we need to develop query rewriting techniques to retrieve relevant tuples containing missing values from one database and then join them with relevant tuples containing missing values from another database.

3. Learning on a sample in order to predict null values can be done for each database at a local schema level or can be done at the global schema level by the mediator.

In the following chapters, we describe how our query processing framework addresses these challenges.

CHAPTER 3

# QPIAD QUERY PROCESSING OVER INCOMPLETE AUTONOMOUS DATABASES

In this chapter[1], we describe our techniques for supporting query processing over incomplete autonomous databases. We describe *QPIAD* – our framework for mediator query processing over incomplete autonomous databases.

## 3.1. Preliminaries and Overview of Solution

Our goal is to retrieve ranked relevant uncertain answers along with certain answers for selection queries on incomplete autonomous databases.

We will start with formal definitions of certain answers and uncertain answers with respect to selection queries.

**Definition 1 (Complete/Incomplete Tuples)** *Let $R(A_1, A_2, \cdots, A_n)$ be a database relation. A tuple $t \in R$ is said to be complete if it has non-null values for each of the attributes $A_i$; otherwise it is considered incomplete. A complete tuple $t$ is considered to belong to the set of completions of an incomplete tuple $\widehat{t}$ (denoted $\mathcal{C}(\widehat{t})$), if $t$ and $\widehat{t}$ agree on all the non-null attribute values.*

[1]The material presented in this chapter is based on the ideas developed by the author with Jianchun Fan.

The notion of completions brings forth the connection between incompleteness and uncertainty: an incomplete tuple can thus be seen as a disjunction of its possible completions. We go a step further and view the incomplete tuple as a *probability distribution* over its completions. The distribution can be interpreted as giving a quantitative estimate of the probability that the incomplete tuple corresponds to a specific completion in the real world.

Now consider a selection query $Q{:}\,\sigma_{A_m=v_m}$ $(1 \leq m \leq n)$ over $R$.

**Definition 2 (Certain/Uncertain Answers)** *A tuple $t$ is said to be a certain answer for the query $Q{:}\,\sigma_{A_m=v_m}$ if $t.A_m{=}v_m$. $t$ is said to be an uncertain answer for $Q$ where $t.A_m{=}null$ (where $t.A_m$ is the value of attribute $A_m$ in $t$).*

Notice that in the definition above, $t$ can be either a complete or incomplete tuple. An incomplete tuple can be a certain answer to the query, if its incompleteness is not on the attribute being selected by the query.

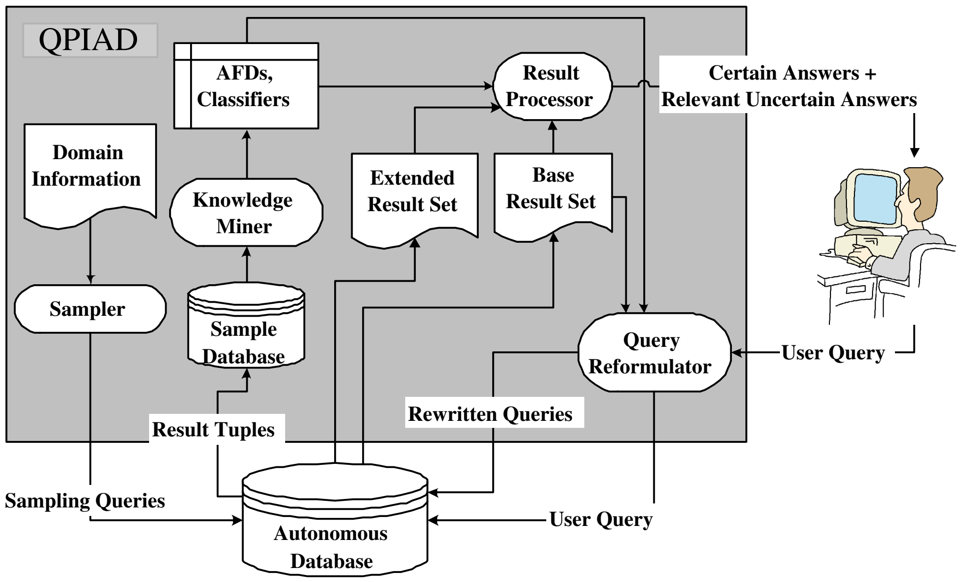

Figure 2. QPIAD System architecture.

Since we view the incomplete tuple as a probability distribution over its completions, it is possible for us to quantify the degree to which an uncertain answer is actually relevant to a query Q.

**Definition 3 (Degree of Relevance)** *The degree of relevance of a tuple* $t$ *to the query* $Q\colon \sigma_{A_m=v_m}$ *is defined to be the probability* $P(t.A_m=v_m)$.

Figure 2 shows our system architecture. In this framework, a user accesses autonomous databases through a mediator. When a user submits a query to the mediator, the query reformulator first directs the query to the autonomous databases and retrieves the set of all certain answers (called the *base result set*). In order to retrieve relevant uncertain answers, the mediator needs to issue additional queries taking into account the limited access patterns of the autonomous databases. We propose online query rewriting techniques to generate new queries based on the original query, the base result set, and a set of rules learned from a database sample. The goal of these new queries is to return an *extended result set*, which consists of highly relevant uncertain answers to the original query. Since these new queries are not all equally good in terms of retrieving relevant uncertain answers, they are ranked before being posed to the databases.

For example, consider a user query $Q\colon \sigma_{make=Honda}(cars)$, the mediator returns the following certain answers that satisfy the query predicates:

| **Make** | **Model** | **Year** | **Color** | ... |
|---|---|---|---|---|
| Honda | Accord | 1999 | red | $\cdots$ |
| Honda | Civic | 2000 | cactus | $\cdots$ |
| Honda | Accord | 2001 | silver | $\cdots$ |
| $\cdots$ | $\cdots$ | $\cdots$ | $\cdots$ | $\cdots$ |

From the table above, it is intuitive that if we generate a new query to retrieve tuples from the database having *model*=*Accord*, we are likely to get cars made by *Honda*, including those having missing value on the make attribute. We *capture this intuition by mining attribute correlations* from the data itself. One obvious type of attribute correlations is "*functional dependencies*". For example, the functional dependency $model \rightarrow make$ often holds in automobile data records. Given such a dependency, we can see that among all the tuples with missing *make* values, the ones with *model* value "*Accord*" (or other car models made by Honda) are very likely to be query answers and thus should be returned. There are two problems in adapting the method directly based on functional dependencies: (i) often there are not enough strong functional dependencies in the data and (ii) autonomous databases are unlikely to advertise the functional dependencies. The answer to both these problems involves *learning* approximate functional dependencies from a (probed) sample of the database.

**Definition 4 (Approximate Functional Dependency)** *$X \rightsquigarrow A$ over relation $R$ is an* approximate functional dependency(AFD) *if it holds on all but a small fraction of the tuples. The set of attributes $X$ is called the* determining set *of $A$ denoted by dtrSet(A).*

For example, an AFD $model \rightsquigarrow body\ style$ indicates that the value of a car's *model* attribute *sometimes* (but not always) determines the value of *body style* attribute. Note that AFDs can also describe *distributional* properties of the attribute values without explicit semantic significance. For example, AFD $(model, color) \rightsquigarrow year$ suggests that statistically some *year* value is more common than others for a given $(model, color)$ value pair.

While AFDs can help us rewrite a given query to retrieve uncertain answers, we still need a mechanism for ranking the rewritten queries as not all rewritten queries produce equally relevant uncertain answers. We can rank the rewritten queries if we have access to distribution of values for the missing attribute. For example, consider an AFD $model \rightsquigarrow body\ style$ and the original user query $\sigma_{body\ style=convt}(Cars)$ asking for all convertible(convt) cars in the database. In this case, the two rewritten queries $Q_1$: $\sigma_{model=Mustang}$ and $Q_2$: $\sigma_{model=Z4}$ are generated to retrieve tuples that have null values on the attribute *body style* but are likely to be relevant to the query. If we know $P(body\ style{=}convt \mid model{=}Mustang){=}0.4$ and $P(body\ style{=}convt \mid model{=}Z4){=}0.9$, then we can estimate that query $Q_2$ is likely to retrieve uncertain answers that are more relevant than those of $Q_1$. This type of relevance ranking is beneficial in mediated query processing over autonomous databases. When database or network resources are limited, the mediator can send only the most relevant rewritten queries to the autonomous databases and sacrifice least amount of relevant uncertain answers. It also allows the query processing procedure to adapt to different users' preferences in terms of precision and recall of the uncertain answers.

The correlations between attributes and value distributions are exploited in query rewriting to retrieve highly relevant uncertain answers. We reduce the problem of acquiring such value distributions to learning classifiers based on a sample of the autonomous database. We develop an AFD-enhanced Naïve Bayesian classifier learning method(where AFD plays a feature selection role for the classification task).

Our system mines AFDs and learns value distributions on a small portion of data from the autonomous database. The sampling module collects the sample data from the autonomous database, and the knowledge mining module learns AFDs and the AFD-enhanced classifiers from these samples.

## 3.2. Retrieving Relevant Incomplete Information

In this section we first introduce how to retrieve relevant uncertain answers from incomplete local databases, over which we have full control. We then discuss the limitations of these approaches in the context of autonomous databases. Finally, we will present a novel query rewriting technique that overcomes these limitations.

To evaluate the effectiveness of methods for retrieving relevant uncertain answers from incomplete databases, we use *precision* and *recall* as the measurements. Precision is defined as the ratio of the number of relevant tuples retrieved to the total number of tuples retrieved; and recall is defined as the ratio of the number of relevant tuples retrieved to the total number of relevant tuples in the database.

**3.2.1. Retrieving Relevant Uncertain Answers from Local Databases.** Typically, missing values in relational databases are interpreted in two typical ways. The first approach treats missing values as a special value, "null", which is not equal to any concrete value . In this case, evaluating a selection query with attributes bound to a concrete value does not return the tuples with null on the constrained attributes. For example, a selection query on a Car database $\sigma_{body\ style=convt}(Cars)$, retrieves all tuples having *convt* in the *body style* attribute. This approach gives answers with 100% precision but low recall, since all the relevant uncertain answers are ignored. We call this approach as returning *certain answers only (CAO).*

Alternatively, a null value can be interpreted as a universal value, i.e. the system assumes that without further information, a null possibly matches any concrete value. In this case, evaluating a selection query returns not only all the certain answers but also all the uncertain answers that have nulls on the constrained attributes. For example, for the same selection query as above $\sigma_{body\ style=convt}(Cars)$, it returns the results of the extended query $\sigma_{body\ style=convt\ \vee\ body\ style=null}$. This approach gives answers with 100% recall but low precision, since not all uncertain answers are relevant. We call this approach as returning *all uncertain answers(AUA)*.

When traditional relational databases evaluate selection queries, each tuple is considered as either a *correct* answer or an incorrect answer. In traditional databases, there is no gray area and thus no *uncertain* answers will be shown to the user. Naturally, when it comes to incomplete databases, such systems implicitly use the CAO approach. As shown in the example above, such systems can support the AUA approach by extending the selection predicates of the queries with a *null value binding*, such as *body style=null*.

We propose an approach that returns *relevant uncertain answers(RUA)* by interpreting missing values according to the inherent correlations among attributes and values. With a better estimation of missing values, this approach improves precision significantly when compared with AUA. In Section 3.3, we will present an AFD-enhanced classifier to explore the correlations among attributes and values. In this section, we focus on how to leverage the mined correlations to retrieve relevant uncertain answers.

We begin by retrieving all the certain and uncertain answers, as in AUA. The certain answers are then presented directly to the users as sound answers. Then for each uncertain answer, we apply the classification techniques (see Section 3.3) to assess its likelihood of being a relevant answer to the query.

Specifically, let $R(A_1, A_2, \cdots, A_n)$ be a database relation, and query $Q\colon \sigma_{A_m=v_m}$ $(1 \leq m \leq n)$ a selection query over $R$. Suppose $\hat{t}$ is an uncertain answer returned by AUA, with a null value on attribute $A_m$. Suppose we have an AFD $dtrSet(A_m) \rightsquigarrow A_m$, where $dtrSet(A_m)$, the determining set of $A_m$ is a set of attributes that determines $A_m$. In Section 3.3, we will discuss how to mine and choose AFDs in detail. Let $\hat{t}(dtrSet(A_m))$ be the projection of tuple $\hat{t}$ on the determining set. For tuple $\hat{t}$, our AFD-enhanced classifiers give the value probability distribution $P(A_m{=}v_m \mid \hat{t}(dtrSet(A_m)))$, which captures the likelihood of $\hat{t}$ being relevant to the query. Such probability is computed for each uncertain answer as its ranking score. RUA only returns the most relevant uncertain answers whose ranks are above a given threshold. These results are returned to the user following the set of certain answers.

Our experiments (see Section 3.4) show that RUA approach significantly improves the precision of the relevant uncertain answers that are returned when compared to AUA.

**3.2.2. Retrieving Relevant Uncertain Answers from Autonomous Databases.** Retrieving relevant uncertain answers is more challenging when it comes to autonomous web databases when compared to local databases. First, web databases usually do not support arbitrary selection query patterns as local databases do. For example, a website rarely allows users to submit queries such as "list all the cars that have a missing value for *body style* attribute". Therefore the mediator may not be able to directly retrieve uncertain answers, thus AUA and RUA approaches are not applicable in this scenario. Second, even if the web databases allow null value binding, AUA and RUA approaches tend to return too many irrelevant uncertain answers. In the sample query $\sigma_{body\ style=convt}(Cars)$, both AUA and RUA retrieve all the tuples with missing *body style* values, most of which may not be "convt" at all. Although RUA ranks uncertain answers and only returns the most relevant ones, significant database and network resources are wasted while processing and transmitting the irrelevant ones. Thus in a web environment, such approaches are not desirable.

To address the two challenges above, intuitively we would like to issue queries in an intelligent way, so that the query patterns are more likely to be supported by web databases, and only the most relevant uncertain answers are sent back to the mediator in the first place.

**Query Rewriting:** The goal of our query rewriting is to generate a set of rewritten queries to retrieve relevant uncertain answers. Let's consider the same user query $Q'$ asking for all convertible cars. We use the fragment of Car database shown in Table 2 to explain our approach. First, we issue the query $Q'$ to the autonomous database to retrieve all the certain answers which correspond to tuples $t_1$, $t_2$ and $t_3$ from Table 2. These certain answers form the *base result set* of $Q'$.

| **Make** | **Model** | **Year** | **Body style** |
|---|---|---|---|
| Audi | A4 | 2001 | convt |
| BMW | Z4 | 2002 | convt |
| Porsche | Boxster | 2005 | convt |

In this table, we only show the *make*, *model*, *year* and *body style* attributes since *body style* is the constrained attribute in the query and suppose we have an AFD *model* $\rightsquigarrow$ *body style*. Consider the first tuple $t_1{=}\langle Audi, A4, 2001, convt\rangle$ in the base result set. If there is a tuple $t_i$ in the database with the same value for *model* as $t_1$ but missing value for *body style*, then $t_i$.*body style* is likely to be *convt* according to the AFD. Therefore we consider $t_i$ as a relevant uncertain answer to the query $Q'$ and to retrieve $t_i$ from the database (which does not support null value binding), the mediator can issue another query $Q_1{:}\,\sigma_{model=A4}$. Similarly, we can issue queries $Q_2{:}\,\sigma_{model=Z4}$ and $Q_3{:}\,\sigma_{model=Boxster}$ to retrieve other relevant uncertain answers. Note that the generated queries will return highly relevant uncertain answers, such as $t_4$ from Table 2, as well as a few tuples whose *body style* value is neither *convt* nor *null*, as the AFD only holds approximately. Therefore, we would like to filter out those tuples from the answers which will be discussed in more detail later.

As illustrated above, this approach restricts the retrieval of uncertain answers to just the *relevant* tuples, it is named as returning *restricted relevant uncertain answers (RRUA)*. RRUA follows a two-step approach. First, the original query is sent to the database to retrieve the certain answers which are then returned to the user. Next, a group of rewritten queries are sent to the database to retrieve the relevant uncertain answers, which have the same values as some certain answers on the determining attribute sets of the constrained attribute according to AFDs.

RRUA approach has two advantages. First, it can be used to query autonomous databases which do not support null value binding. Second, RRUA is much more efficient as it only retrieves relevant uncertain answers rather than all uncertain answers thus requiring fewer tuples to be retrieved and transmitted when compared with AUA and RUA.

**Ranking Rewritten Queries:** In the query rewriting step of RRUA, we generate new queries according to the distinct value combination on the determining set of the constrained attribute in the base result set. However, these queries may not be equally good in terms of retrieving relevant uncertain answers. In the example above, based on three certain answers to the user query $Q'$: $\sigma_{body\ style=convt}(Cars)$, we generate three new queries: $Q_1$: $\sigma_{model=A4}$, $Q_2$: $\sigma_{model=Z4}$ and $Q_3$: $\sigma_{model=Boxster}$. Although all three queries retrieve uncertain answers that are likely to be more relevant to $Q'$ than a random tuple with missing *body style* value, they may not be equally good. For example, based on the value distribution in the sample database, we may find that a $Z4$ model car is more likely to be a *convertible* than a car with $A4$ model. As will be discussed in Section 3.3.2, we build AFD-enhanced classifiers which give the probability values $P(body\ style{=}convt|model{=}A4)$, $P(body\ style{=}convt|model{=}Z4)$ and $P(body\ style{=}convt|model{=}Boxster)$. Using these probability values, we rank the rewritten queries according to the relevance of their expected query results.

Ranking rewritten queries according to their answers' expected degrees of relevance brings forth two very appealing characteristics of the RRUA approach.

1. When database or network resources are limited, the mediator can choose the most relevant queries to be sent to the database (bringing in the most relevant answers) and sacrifice the least amount of relevant answers. This allows the mediator to retrieve relevant uncertain answers with highly manageable cost. For a given cost limit (defined in terms of the number of queries to be sent to a database, and correspondingly the number of answer tuples returned), RRUA is able to maximize the precision.

2. This adjustable mechanism allows the mediator to adapt to different users' preferences on precision and recall. For users who care more about the relevance or precision of the results, the mediator can send only the top ranked rewritten queries. For those who are more interested in recall, the mediator can send a larger portion of the rewritten queries and retrieve more relevant uncertain answers.

**3.2.3. Algorithm for RRUA Approach.** Algorithm 1 shows the details of the RRUA approach. Let $R(A_1, A_2, \cdots, A_n)$ be a database relation, and $dtrSet(A_m)$ be the determining set of an attribute $A_m$ $(1 \leq m \leq n)$, according to the best selected AFD (to be discussed in Section 3.3.3). RRUA processes a given selection query $Q : \sigma_{A_m = v_m}$ according to the following two steps.

1. Send $Q$ to the database and retrieve the base result set $RS(Q)$ as the certain answers of $Q$. Return $RS(Q)$ to the user.

2. Generate a set of new queries, rank them, and send the most relevant ones to the database to retrieve the extended result set $\widehat{RS}(Q)$ as relevant uncertain answers of

**Algorithm 1** Rewriting queries to retrieve relevant uncertain answers from autonomous database

```
Require: $R(A_1, A_2, \cdots, A_n)$: database relation; $A_m$: attribute of $R, (1 \leq m \leq n)$; $Q : \sigma_{A_m=v_m}$: selection query over $R$;
 1: send $Q$ to the database
 2: let $RS(Q)$= base result set returned for $Q$
 3: show $RS(Q)$ to the user
 4: for all $t_i \in \pi_{dtrSet(A_m)}(RS(Q))$ do
 5:    selection predicate $sp_{Q_i} = \emptyset$
 6:    for all $A_j \in dtrSet(A_m)$ do
 7:       $sp_{Q_i} = sp_{Q_i} +$ " $\wedge A_j = t_i(A_j)$"
 8:    $Q_i =$ "$\sigma_{sp_{Q_i}}$"
 9:    $P_{Q_i} = P(A_m = v_m|t_i)$
10: rank $\{Q_i|t_i \in \pi_{dtrSet(A_m)}(RS(Q))\}$ according to $P_{Q_i}$
11: pick top K queries
    $(1 \leq K \leq ||\pi_{dtrSet(A_m)}(RS(Q))||)$
12: let extended result set $\widehat{RS}(Q) = \emptyset$
13: for all $Q_i \in$ top K queries do
14:    send $Q_i$ to the database and retrieve result set $RS(Q_i)$
15:    append $RS(Q_i)$ to $\widehat{RS}(Q)$
16: for all $\widehat{t_r} \in \widehat{RS}(Q)$ do
17:    if $\widehat{t_r} \in RS(Q)$ or $\widehat{t_r}(A_m) \neq v_m$ then
18:       $\widehat{RS}(Q) = \widehat{RS}(Q) - \{t_r\}$
19: show $\widehat{RS}(Q)$ to the user
```

Q. This step contains the following tasks.

(a) *Generate rewritten queries.* Let $\pi_{dtrSet(A_m)}(RS(Q))$ be the projection of $RS(Q)$ onto $dtrSet(A_m)$. For each tuple $t_i$ in $\pi_{dtrSet(A_m)}(RS(Q))$, create a selection query $Q_i$ in the following way. For each attribute $A_j$ in $dtrSet(A_m)$, create a selection predicate $A_j{=}t_i(A_j)$. The selection predicates of $Q_i$ consist of the conjunction of all these predicates.

(b) *Rank rewritten queries.* Compute the conditional probability of $P_{Q_i}{=}P(A_m{=}v_m|t_i)$ for each $Q_i$. Rank all $Q_i$s according to their $P_{Q_i}$ values.

(c) *Retrieve extended result set.* Pick the top $K$ queries $\{Q_1, Q_2, \cdots, Q_K\}$ and issue them in the order of their ranks. Their result sets $RS(Q_1), RS(Q_2), \cdots, RS(Q_K)$ compose the extended result set $\widehat{RS}(Q)$. The results in $\widehat{RS}(Q)$ are ranked according to the ranks of the corresponding queries, i.e. the results in $RS(Q_1)$ are ranked higher than those in $RS(Q_2)$, and so on.[2]

(d) *Post-filtering.* Remove from $\widehat{RS}(Q)$ the tuples with $A_m \neq null$. Return the remaining tuples in $\widehat{RS}(Q)$ as the relevant uncertain answers of $Q$.

For each relevant uncertain answer $\widehat{t_r}$, the mediator can optionally return its relevance confidence, $P(\widehat{t_r}(A_m){=}v_m \mid \widehat{t_r}(dtrSet(A_m)))$. This allows the users to further filter the answers according to their own preferences on precision and recall.

**Multi-attribute Selection Queries:** Although we described the above algorithm in the context of single attribute selection queries, it is easy to see that this algorithm can be adapted to support multi-attribute selection queries by adding extra selection predicates. These selection predicates correspond to the user query constrained values except the attribute for which we are retrieving a relevant uncertain tuple having a null value. They are added while generating the rewritten queries in Step 2(a).

As discussed earlier, web databases usually support limited types of query patterns. In the case where null value binding is supported by the database (thus AUA, RUA and RRUA approaches are all applicable), RRUA approach can take advantage of the null value binding to avoid retrieving undesirable tuples. In the query rewriting step, each of the rewritten queries can have an extra selection predicate ($A_m{=}null$) in conjunction with other predicates. In this case, the post-filtering step can be avoided.

Admittedly, the queries issued by RRUA are likely to contain more predicates, and therefore are more complex than those of AUA and RUA. As a result the database query

[2] All results returned for a single query are ranked equally.

processing cost could increase. However, in a web environment, the overall processing cost is highly dominated by the data transmission latency. By significantly reducing the amount of irrelevant data transmitted, RRUA becomes more desirable in terms of overall cost.

## 3.3. Learning Attribute Correlations and Value Probability Distributions

As we have discussed, to retrieve uncertain answers in the order of their relevance, RRUA requires two types of information: (i) attribute correlations in order to generate rewritten queries (ii) value distributions in order to rank the rewritten queries. In this section, we present how each of these are learned. Our solution consists of two stages. First, the system mines the inherent correlations among database attributes represented as AFDs. Then it builds Naïve Bayes Classifiers based on the features selected by AFDs to compute probability distribution over the possible values of the missing attribute for a given tuple. We exploit AFDs for feature selection in our classifier as it has been shown that appropriate feature selection before classification can improve learning accuracy[6].

**3.3.1. Learning Attribute Correlations by Mining Approximate Functional Dependencies(AFDs).** In this section, we describe the method for mining AFDs from a (probed) sample of database. We also present our algorithm for pruning noisy AFDs in order to retain only the valuable ones for use in the query rewriting module. Recall that an AFD is a functional dependency if holds on all but a small fraction of tuples. Several error measurements for AFDs have been proposed [23, 12, 24]. Among them, the $g_3$ measure proposed by Kivinen and Mannila [23] is widely accepted. $g_3$ measure is defined as the ratio of the minimum number of tuples that need to be removed from relation $R$ to make $X \rightsquigarrow A$ a functional dependency over the total number of tuples in $R$. The *confidence* of an AFD

$X \rightsquigarrow A$ is defined as: $conf(X \rightsquigarrow A)$=$1 - g_3(X \rightsquigarrow A)$. Similarly, we define *approximate key(AKey)* as an attribute set $X \subset R$, such that $X$ is a key of all but a small fraction of tuples in $R$. We use TANE[20] algorithm to discover AFDs and AKeys whose confidence is above a threshold $\alpha$, which is set to 0.3 in our system. This ensures that we do not miss any significant AFDs.

**Pruning Noisy AFDs:** In most cases, AFDs with high confidence are desirable features for learning probability distributions for missing values. However, not all high confidence AFDs are useful for feature selection. To see this, we first observe that AFDs and AKeys act differently for missing value prediction. AKeys with high confidence are not useful because most of the AKey values are distinct. For example, consider a relation *car(VIN, model, make, color, year)*. After mining, we find that *VIN* is an AKey (in fact, a key) which determines all other attributes. Given a tuple $t$ with null value on *model*, its *VIN* is not helpful in estimating the missing *model* value, since there are no other tuples sharing $t$'s *VIN* value. Therefore, we use AFDs with high confidence as selecting features for classification as long as their determining sets are not AKeys.

**Algorithm 2** Pruning AFDs

```
Require: $A$: set of AFDs along with their confidence,$K$: set of AKeys along with their
    confidence
1: set $N = \emptyset$
2: for each AFD $A_i$ in $A$ do
3:   for each Approximate key $K_i$ in $K$ do
4:     if $K_i \subseteq dtrSet(A_i)$ and $(conf(A_i) - conf(K_i)) < \delta$ then
5:       Prune $A_i$
6:     else
7:       $N = N \cup A_i$
8: return N
```

Algorithm 2 describes how to prune AFDs that have high confidence AKey in the determining set. Note that AFDs with a superset of AKey attributes in the determining set should also be removed. For example, suppose we have an AFD $\{A_1, A_2\} \rightsquigarrow A_3$ with confidence 0.97, and an AKey $\{A_1\}$ with confidence 0.95. Since $\{A_1, A_2\}$ is a high confidence superset of AKey, where most of $\{A_1, A_2\}$ value pairs would be distinct, this AFD won't be useful in predicting the values for $A_3$ and needs to be pruned. The difference between the confidence level of an AFD and the corresponding AKey is considered in AFD pruning based on a threshold $\delta$(currently set at 0.3 in our system based on experimentation).

**3.3.2. Learning Value Distributions using Classifiers.** Given a tuple with a null value, we now need to estimate the probability of each possible value of the attribute containing the null. We reduce this problem to a classification problem using mined AFDs as a feature selection step. A classifier is a function $f$ that maps a given attribute vector $\vec{x}$ to a confidence that the vector belongs to a class - that is, $f(\vec{x}){=}confidence(class)$. The input of our classifier is a random sample $S$ of an autonomous database $R$ with attributes $A_1, A_2, \cdots, A_n$ and the mined AFDs. For a given attribute $A_m$, $(1 \leq m \leq n)$, we compute the probabilities for all possible class values of $A_m$, given all possible values of its determining set $dtrSet(A_m)$ in the corresponding AFDs.

We construct a naïve-Bayes classifiers (NBC) for each missing attribute. Let a value $v_i$ in the domain of the missing attribute $A_m$ represent a possible class for $A_m$. Let $\vec{x}$ denote the values of $A_m$'s determining set in a tuple with missing $A_m$ value. We use Bayes theorem to estimate the probabilities:

$$P\left(A_m{=}v_i|\vec{x}\right) = \frac{P\left(\vec{x}|A_m{=}v_i\right) P\left(A_m{=}v_i\right)}{P\left(\vec{x}\right)}$$

for all values $v_i$ in the domain. $P(\vec{x}|A_m{=}v_i)$ is often impractical to compute without simplifying assumptions. NBC assumes that for a given class, the features $X_1, \cdots, X_n$ are conditionally independent, and therefore simplifies the computation to:

$$P\left(\vec{x}|A_m{=}v_i\right) = \prod_i P\left(x_i|A_m{=}v_i\right)$$

Despite this strong simplification, NBC has been shown to be surprisingly effective[15]. In the actual implementation, we adopt the standard practice of using NBC with a variant of Laplacian smoothing called m-estimates[28] to smooth the estimates and to improve the accuracy.

**3.3.3. Combining AFDs and Classifiers.** In the preceding discussion, we glossed over the fact that there may be more than one AFD associated with an attribute. In other words, one attribute may have multiple determining set with different confidence levels. For example, we have the AFD *model* $\rightsquigarrow$ *make* with confidence 0.99. We also see that certain types of cars are made in certain countries, so we might have an AFD *country* $\rightsquigarrow$ *make* with some confidence value. As we use AFDs as a feature selection step for NBC, we experimented with several alternative approaches for combining AFDs and classifiers to learn the probability distribution of possible values for null.

**All Attributes:** In this “baseline” method, we do not consider feature selection based on AFD but instead use all the attributes[3] for learning the value distribution of possible values for null using the Naïve Bayes classifier.

**One AFD with highest Confidence(Best AFD):** In this method, we consider feature selection based on AFD. For each attribute, among the AFDs with this attribute as dependent attribute, we select the one with highest confidence. We use the determining attributes in the selected AFD for learning the probability distribution of null value for that attribute using NBC. If there are multiple AFDs with the same highest confidence value, we would

[3]We can also use this method for attributes for which AFDs do not exist.

select the AFD which has the least number of attributes as the *selectivity* of that AFD will be high when compared to the AFD which has more attributes. For example, if we are learning the probability distribution of the null value for $A_m$ and we have two AFDs with the same highest confidence: $\{A_1, A_2\} \rightsquigarrow A_m$ and $\{A_3, A_4, A_5\} \rightsquigarrow A_m$, we chose the first AFD as it has fewer attributes. In this method, there will be one classifier for each attribute which is based on the AFD selected. Based on the features selected by the AFD, we use a Naïve Bayes Classifier to compute the value distribution of the possible values for the null.

**Hybrid One-AFD:** Our experiments(see Section 3.4.3) indicate that using an AFD which has a very low confidence degrades the classification accuracy. So, if we are using just the best AFD, namely the one with the highest confidence, and the confidence value is quite low, then that AFD will not be effective in learning an accurate probability distribution for a null value. Hence, we do prune the AFDs based on the confidence measure. If the confidence of the best selected AFD is below a threshold[4], then we ignore that AFD for feature selection and instead add another AFD having the highest confidence with its determining set consisting of all other attributes except the attribute we are predicting. Thus, for such low confidence AFDs, we wind up using all the attributes as in the method "All Attributes" for learning the value distribution of the possible values for the null using a Naïve Bayes Classifier.

**Ensemble of Classifiers:** Each attribute may have multiple AFDs, all of which can be used for learning the probability distribution of null value for that attribute. So we combine the predictions of all AFDs by using an ensemble of classifiers as follows: We have a set of $k_m$ AFDs for each attribute $A_m$: $S_1 \rightsquigarrow A_m, S_2 \rightsquigarrow A_m, \cdots, S_{km} \rightsquigarrow A_m$. So we have $k_m$ $classifiers$ - $C_1, C_2, \cdots, C_{k_m}$ corresponding to the $k_m$ AFDs to compute the probability

[4]Currently the threshold is set at 0.5 based on experimentation

distribution of $A_m$ with each classifier $C_i$ using different set of features. Each classifier $C_i$ is trained independently on the sample training set and then is used to predict individually on the testing set. The accuracy that the classifier $C_i$ achieved on the testing test is chosen as the weight of the classifier $C_i$ denoted by $w_i$. The weight for each classifier is determined by performing multiple runs on different training data and testing data to get an average value of the weight $w_i$. While computing the value distribution for missing values using the ensemble classifier, we do a weighted voting of all the classifiers based on the learned weights for each classifier.

$$P(A_m{=}V_i){=}\sum_{i=1}^{k_m} w_i * P(A_m{=}V_i|S_i)$$

Our experiments described in Section 3.4.3 show that Hybrid One-AFD approach performs better than all other approaches in terms of classification accuracy.

**3.3.4. Learning Value Distributions Online from Base Result Set.** $RRUA$ approach learns value distributions on a probed sample of autonomous database. Any technique based on learning from a sample can be inaccurate if it encounters data previously unseen in the training sample. For the cars domain, the probed sample covered a major portion of the domain values for all attributes. However, in domains where the sample obtained through probing queries covers only a small fraction of the actual domain values, learning based on a sample would lead to an inaccurate estimation of value distributions. For example, consider a rare query $Q{:}\,\sigma_{model=viper}$ which corresponds to a car made by *Dodge* for which we have no data in our training sample. Hence, the rewritten query ranker module would not be able to rank the queries for bringing in possible relevant tuples having missing values on *model*. Hence, we propose an online learning algorithm $O{-}RRUA$ which ranks rewritten queries based on the value distribution learned from the base result set

**Algorithm 3** Rewriting queries online using $O{-}RRUA$ approach to retrieve relevant uncertain answers

```
Require: $R(A_1, A_2, \cdots, A_n)$: database relation; $A_m$: attribute of $R, (1 \leq m \leq n)$; $Q$ : $\sigma_{A_m=v_m}$: selection query over $R$;
 1: send $Q$ to the database;
 2: let $RS(Q) = base\ result\ set\ returned\ for\ Q$;
 3: learn AFD-enhanced NBC on the base result set $RS(Q)$;
 4: for all $t_i \in \pi_{dtrSet(A_m)}(RS(Q))$ do
 5:    selection predicate $sp_{Q_i} = \emptyset$;
 6:    for all $A_j \in dtrSet(A_m)$ do
 7:       $sp_{Q_i} = sp_{Q_i} +$ " $\wedge A_j = t_i(A_j)$";
 8:    $Q_i =$ "$\sigma_{sp_{Q_i}}$";
 9:    $P_{Q_i} = P(A_m = v_m|t_i)$;
10: rank $\{Q_i|t_i \in \pi_{dtrSet(A_m)}(RS(Q))\}$ according to $R_{Q_i}$;
11: pick $top\ K\ queries$
    $(1 \leq K \leq ||\pi_{dtrSet(A_m)}(RS(Q))||)$;
12: let extended result set $\widehat{RS}(Q) = \emptyset$;
13: for all $Q_i \in top\ K\ queries$ do
14:    send $Q_i$ to the database and retrieve result set $RS(Q_i)$;
15:    append $RS(Q_i)$ to $\widehat{RS}(Q)$;
16: for all $\widehat{t_r} \in \widehat{RS}(Q)$ do
17:    if $\widehat{t_r} \in RS(Q)$ or $\widehat{t_r}(A_m) \neq v_m$ then
18:       $\widehat{RS}(Q) = \widehat{RS}(Q) - \{t_r\}$;
19: show $\widehat{RS}(Q)$ to the user;
```

instead of using a NBC classifier on a probed sample. $O{-}RRUA$ uses a probed sample in order to learn only higher level attribute correlations in terms of AFDs which are less susceptible to sample biases. Hence, $O{-}RRUA$ is less prone to any biases that the sample may introduce.

We use value distributions which are learned online by applying NBC classifiers on the base result set to rank the rewritten queries so that they are issued in order of their likelihood of bringing highly relevant results. We propose an online learning algorithm described in Algorithm 3 that dynamically ranks the rewritten query based on the tuples in the base result set. $O{-}RRUA$ approach uses the same set of rewritten queries as $RRUA$, except that the ranking of queries is based on the value distributions learned from the base result set rather than from a probed sample. The algorithm is based on the heuristic that frequently appearing determining set of attributes are more likely to bring in possibly relevant tuples containing null on the query constrained attribute.

## 3.4. Empirical Evaluation for QPIAD

In this section, we describe implementation and an empirical evaluation of our system *QPIAD* for query processing over incomplete web databases.

**3.4.1. Implementation and User Interface.** *QPIAD* system is implemented in Java and has a form based query interface. The system returns each relevant uncertain answer to the user along with a *confidence* measure equal to the assessed degree of relevance. Although the relevance estimate could be biased by the imperfections of the learning method, its inclusion can provide useful guidance to the users, over and above the ranking. This allows users to further filter off uncertain answers based on her notion of what level of confidence would be acceptable. In the used car domain, due to large number of answers retrieved, the user may only be interested in top uncertain answers having high confidences. In other sensitive applications like an FBI agent querying a database in order to identify possible terrorist suspects, the agent may look for more uncertain results even with low confidence. *QPIAD* also can optionally "explain" its relevance assessment by providing snippets of its reasoning. In particular, it justifies the confidence associated with an answer by listing the AFD that was used in making the density assessment. In case of our running example, the uncertain answer $t_4$ for the query $Q'$ will be justified by showing the learned AFD $model \rightsquigarrow body\ style$.

**3.4.2. Experimental Aims & Settings.** Since the performance of the query rewriting and ranking technique relies on the quality of the classifiers, we first show that our AFD-enhanced classifiers give highly accurate classification. We will then evaluate our query rewriting and ranking techniques on incomplete databases. The evaluation spans three aspects (1) comparing the precision, recall and cost of AUA, RUA and RRUA in retrieving relevant uncertain answers (2) evaluating the precision obtained by returning uncertain answers at different confidence thresholds (3) evaluating the query ranking method in RRUA (4) testing the robustness of the RRUA approach and (5) evaluating the effectiveness of query rewriting in order to retrieve relevant answers from data sources not supporting the query attribute.

Two different databases are tested. One is the *Used car database* extracted from AutoTrader[3] which has around 100,000 tuples. The relation of this database is *Cars(make,model,year,price,milage,location,color,body style)*. The second database that we use is the *Census database* from UCI[38] data repository which has around 45,000 tuples. The stored relation of this database is *Census(age, workshop, weight, education, marital-status, occupation, relationship, race, sex, capital-gain, capital-loss, hours-per-week, native-country)*. The *AutoTrader*[41] database inherently has lot of cars having missing values for some attributes as described in Table 1. To evaluate the effectiveness of our techniques against a *ground truth*, we consider only tuples of the database which were complete and then artificially make the test database incomplete by randomly selecting 10% of records and making them incomplete. Each incomplete tuple has exactly one random attribute as null.[5] We use a similar approach to artificially introduce null values in the Census database.

We partitioned each database into two parts: a training set and a test set. To

[5]In order to show the effectiveness of our AFD-enhanced classifier, we only consider missing values on attributes for which AFDs exist.

simulate the relatively small percentage of the training data available to the mediators over autonomous databases, we experimented with different sizes of training set, varying from 3% to 20% of the entire databases. We also artificially insert an ID attribute into each tuple and keep a complete copy of each database so that we can compare the retrieved incomplete tuples against the ground truth to compute the accuracy for our methods.

**3.4.3. Accuracy of Classifiers.** Since we use classifiers as a basis for query rewriting and for ranking queries, we perform a baseline study on their accuracy. Specifically, for each tuple in the test set, we compute the probability distribution of possible values of a null, and choose the one with the maximum probability. Then we can compare the predicted value against the actual value of that attribute to see if the prediction is correct. The classification accuracy is defined as the proportion of the tuples in the test set that have their null values predicted correctly.

In this experiment, we use a training set whose size is 10% of the database to train the classifiers for each attribute, according to the methods discussed in Section 3.3.3. Our goals were to evaluate and compare the classification accuracy achieved by various methods of AFD-enhanced classifiers. We also compare the classification accuracy obtained by the AFD-enhanced classifier versus other alternatives such as Bayesian Network Classifiers.

| **Attribute** | **Make** | **Model** | **Year** | **Color** | **Body style** |
|---|---|---|---|---|---|
| Cars10(Mediator) | 51 | 568 | 28 | 600 | 23 |
| Cars100(Complete) | 100 | 1151 | 55 | 2427 | 28 |

Table 4. Domain size of attributes in the Cars database

| Attribute | Workshop | Education | Marital status | Relationship | Race | Sex | Capital Gain | Hours Week | Native Country |
|---|---|---|---|---|---|---|---|---|---|
| Census25 (Mediator) | 7 | 16 | 7 | 6 | 5 | 2 | 100 | 86 | 41 |
| Census100 (Complete) | 7 | 16 | 7 | 6 | 5 | 2 | 121 | 96 | 41 |

Table 5. Domain size of attributes in the Census database

| **AFD** | **Confidence** |
|---|---|
| $model \rightsquigarrow make$ | 0.99 |
| $(year, location) \rightsquigarrow make$ | 0.51 |
| $price \rightsquigarrow make$ | 0.43 |
| $(make, year, color) \rightsquigarrow model$ | 0.47 |
| $(make, location) \rightsquigarrow model$ | 0.46 |
| $(make, color) \rightsquigarrow model$ | 0.36 |
| $(make, location) \rightsquigarrow year$ | 0.54 |
| $price \rightsquigarrow year$ | 0.45 |
| $model \rightsquigarrow year$ | 0.4 |
| $(make, location) \rightsquigarrow color$ | 0.43 |
| $price \rightsquigarrow color$ | 0.37 |
| $model \rightsquigarrow body\ style$ | 0.9 |

Table 6. AFDs mined for the Cars database

In order to learn AFDs, we need to first collect a representative sample of the data stored in the sources. Since the sources are autonomous, this will involve *probing* the sources with a set of *probing queries*. Table 4 and 5 show the domain size of various attributes for the Cars and Census databases as seen by the mediator and in the complete database. Although the training set in mediator is only 10% of the complete database, it covers a significant part of the domain for each attribute. We mine AFDs from the training set in order to use them for null value prediction on the test set. Table 6 shows the AFDs mined for the Cars database. We tried using different percentages of training set 3%, 5%, 10%, 15%, 20% however the mined AFDs were the same with only a minor change in their confidence values. The best selected AFD for each attribute remained the same across different sample sizes. In addition, the classification accuracy obtained was roughly the same across the sample sizes. This shows that the value distribution learning approach is robust against different sample sizes. Later on in Section 3.4.7 we also show the robustness of our query rewriting and ranking approach $RRUA$ across different sample sizes.

For each database, the accuracy is measured over 5 runs using a different training set

| **Database** | **Best AFD** | **All Attributes** | **Ensemble** | **Hybrid One-AFD** |
|---|---|---|---|---|
| Cars | 43.06 | 28.22 | 34.3 | 43.86 |
| Census | 72 | 70.51 | 70.56 | 72 |

Table 7. Comparison of null value prediction accuracy across different AFD-enhanced classifiers

and test set for each run. Table 7 shows the average prediction accuracy of various AFD-enhanced classifiers for both databases. Considering the large domain sizes of attributes in the Cars database the classification accuracy obtained is quite reasonable, since a random guess would give much lower prediction accuracy.

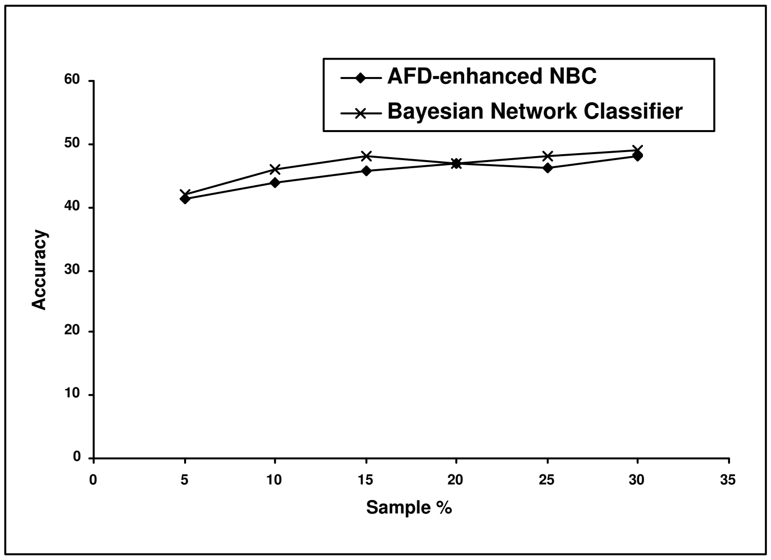

Figure 3. Comparison of prediction accuracy across AFD-enhanced NBC classifier and bayes network classifier over different samples sizes of the Car database.

We can also see in Table 7 that the Hybrid One-AFD approach performs better than both the method that uses all attributes and the method that uses an ensemble classifier. Note that in the Census database, the best selected AFD for each attribute has a confidence value above the threshold, therefore the performance of using the best AFD method is the same as that of using the Hybrid One-AFD method. As a result, we use the Hybrid One-

AFD method in our query rewriting implementation. We will also see that this helps us to keep query cost manageable during query rewriting phase as we issue much less number of queries by using just one AFD than using multiple AFDs.

We also compared the accuracy of our AFD-enhanced NBC classifier with two other approaches - one based on association rules[40] and the other that learns bayesian networks[17] from the data. Figure 3 shows comparison of the accuracy between the AFD-enhanced NBC classifier and bayes network. For various sample sizes, the AFD-enhanced NBC classifier performs on par with the bayesian network classifier while taking significantly less time to learn. The accuracy of our approach is competitive with the bayesian networks approach while being significantly better than the association rules. Learning time for bayesian networks is significantly higher compared to AFD-enhanced NBC classifier.

**3.4.4. Comparing RRUA with AUA and RUA.** To compare the effectiveness of retrieving relevant uncertain answers, we randomly formulate selection queries and retrieve uncertain answers from the test databases using AUA, RUA and RRUA. Recall that AUA approach presents all tuples containing missing values on the query constrained attribute without ranking them. RUA approach begins by retrieving all the certain and uncertain answers, as in AUA, then it rank uncertain answers according to the classification techniques described in Section 3.3. In contrast, RRUA uses query rewriting techniques to retrieve only relevant uncertain answers in a ranked order. For each uncertain answer retrieved, we verify whether or not it is indeed an answer to the user query by checking if it is also a result for the query when evaluated on the original complete copy of the databases. Since RUA and RRUA rank uncertain answers in terms of relevance, we use precision-recall curves to compare the relevance of the returned answers. Precision and recall are calculated

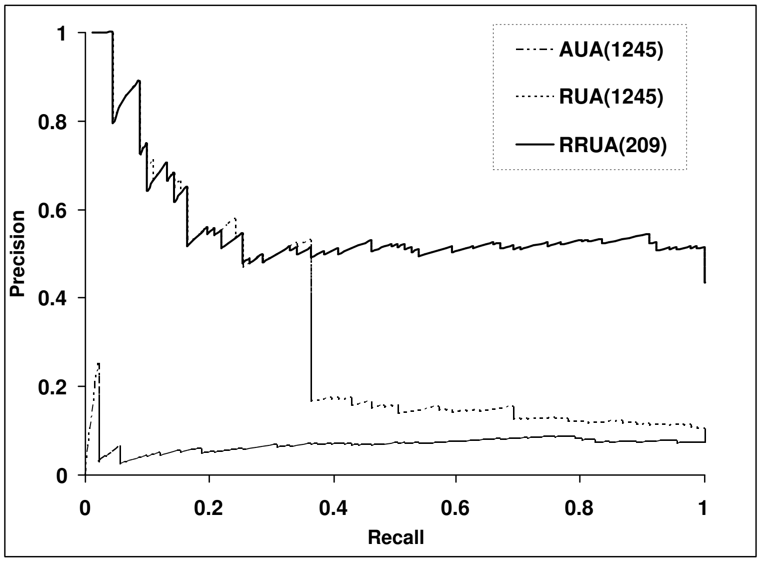

Figure 4. Precision-Recall curves for $\sigma_{body\ style=convt}(Cars)$.

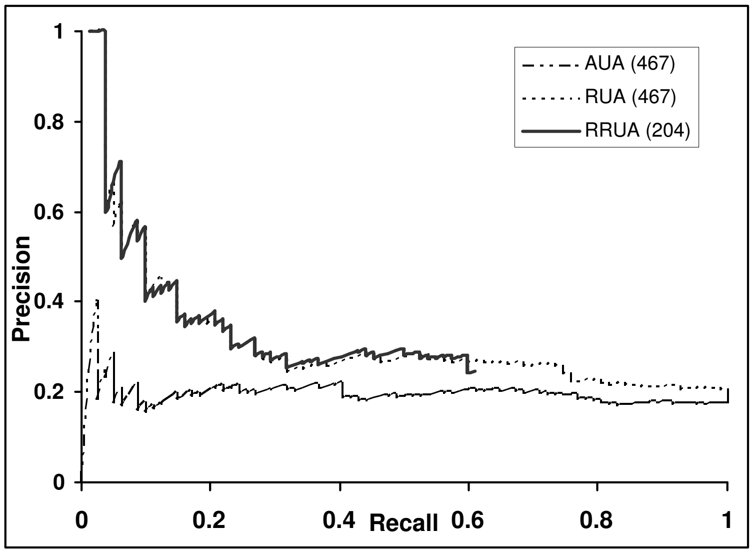

Figure 5. Precision-Recall curves for $\sigma_{education=bachelors}(Census)$.

as follows. At the time when the mediator sees the $K^{th}$ ($K$=1, 2, 3, $\cdots$) answer tuple,

$$precision = \frac{number\ of\ true\ relevant\ answers\ so\ far}{number\ of\ answers\ so\ far}$$

$$recall = \frac{number\ of\ true\ relevant\ answers\ so\ far}{total\ number\ of\ true\ relevant\ answers}$$

Since all three methods return certain answers, we focus on comparing the effectiveness of retrieving relevant uncertain answers. Here, the precision and recall are both calculated with respect to uncertain answers instead of the entire answer set. Figure 4 shows the precision and recall curves of a query on car database, and Figure 5 shows the curves of a query on census database.[6] In both figures, the numbers in the parenthesis in the legend are the number of uncertain answers retrieved by each method. It shows that both RUA and RRUA approach have significantly higher precision compared to AUA. The curves for RUA and RRUA are almost always overlapping for the Cars and Census database since RRUA returns a subset of the answers of RUA, and both of them use the same probability distribution to rank the query answers. For the query *body style=convt*, the curves for RRUA and RUA are overlapping in the initial portion. However, later on precision for RUA falls since it returns all uncertain answers and in the process returns large number of tuples which have a higher rank but are not really relevant to the query. In contrast, RRUA only retrieves few highly relevant tuples, hence the precision for the returned tuples remains high throughout. Note that RRUA does not reach 100% recall since it does not return all uncertain answers.

In terms of the cost, RRUA is the best as it avoids retrieving too many irrelevant tuples. The number of tuples processed and transmitted by RRUA approach is a small

[6]Note that to evaluate the full spectrum of precision-recall curve, we return all results for RUA without pruning them on a threshold on confidence in all the experiments. Similarly, RRUA sends all rewritten queries rather than few selected top ranked ones to databases.

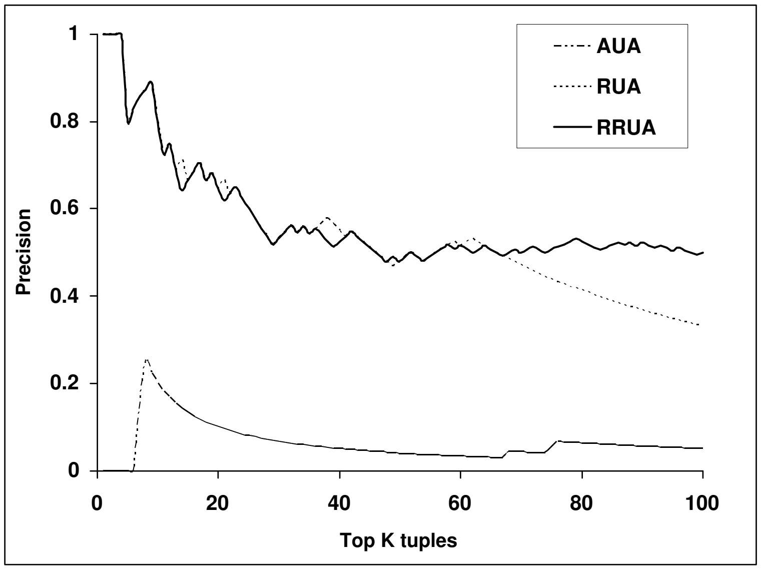

Figure 6. Precision curves for $\sigma_{body\ style=convt}(Cars)$.

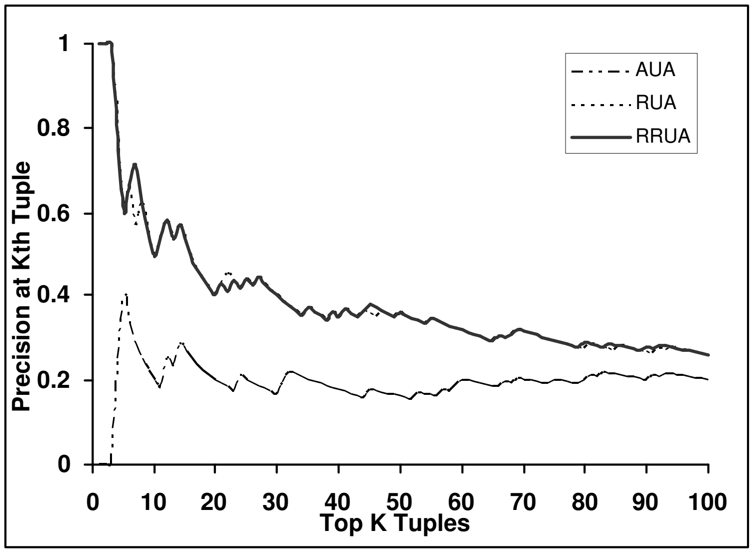

Figure 7. Precision curves for $\sigma_{education=bachelors}(Census)$.

fraction of that by AUA and RUA approaches. RRUA can further cut down the cost by only sending the top ranked rewritten queries to the database.

Though precision-recall curves capture the relevance of tuples in the full range of the uncertain answer set, they cannot reflect the "density" of the relevant answers along the time line. Thus, we also plot the precision of each method at the time when top $K(K{=}1, 2, \cdots, 100)$ answers are retrieved using the same experiment setting, as shown in Figure 6 and 7. Again RUA and RRUA are much better in retrieving relevant uncertain answers in top $K$ results which is critical in web scenarios. In Figure 6, the curve of AUA is not visible since it performs so bad that there are no relevant answers in the first 100 tuples returned.

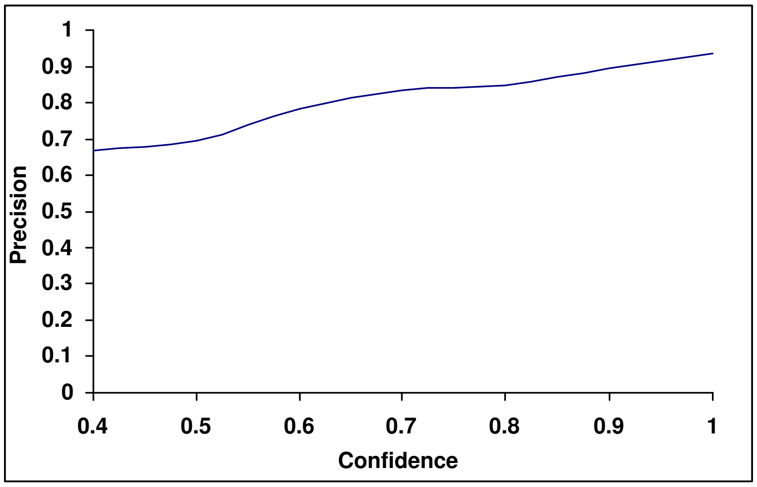

Figure 8. Average Precision for various confidence thresholds(Cars).

**3.4.5. Effect of Confidence Threshold on Precision.** *QPIAD* presents ranked relevant uncertain answers to users along with a confidence so that the users can use their own discretion to filter off answers with low confidence. We conducted experiments to evaluate how pruning answers based on a confidence threshold affects the precision of the results returned. Figure 8 shows the average precision obtained over 40 test queries on Cars database by pruning answers based on different confidence thresholds. It shows that the high confidence answers returned by *QPIAD* are most likely to be relevant answers.

**3.4.6. Ranking the Rewritten Queries.** To control query processing cost according to application requirements, users' preferences and real-time system workload, ranking the rewritten queries is critical in the RRUA approach. In this section, we verify whether the ranked rewritten queries actually bring uncertain answers in the descending order of relevance, measured by the precision of the returned answer set.

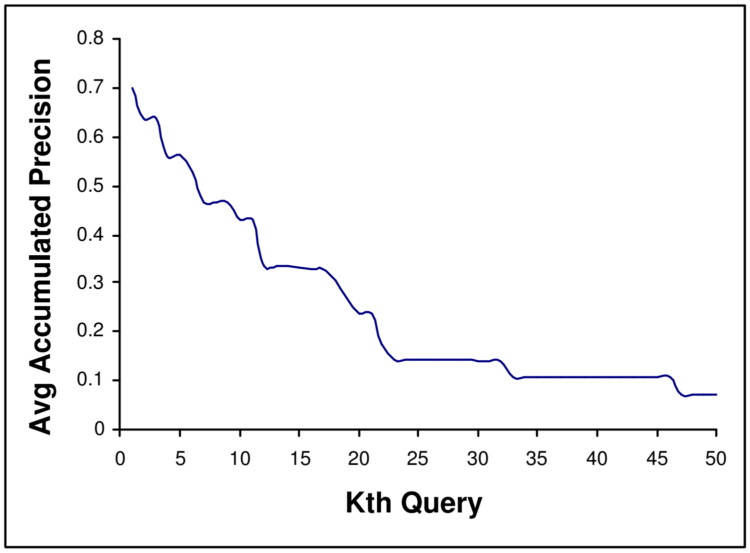

Figure 9. Accumulated precision curve for 30 test queries on Car database.

For this purpose, we experiment on the two test databases as follows. For each database, we choose 30 test queries, each of which is a selection query with a single "attribute=value" predicate. For each query, we use RRUA to generate a ranked list of rewritten queries $\{Q_1, Q_2, \cdots, Q_n\}$, which are issued to the database one by one in order. After the extended result set $\widehat{RS}(Q_m)$ of each query $Q_m$ is retrieved, we calculate the *accumulated*

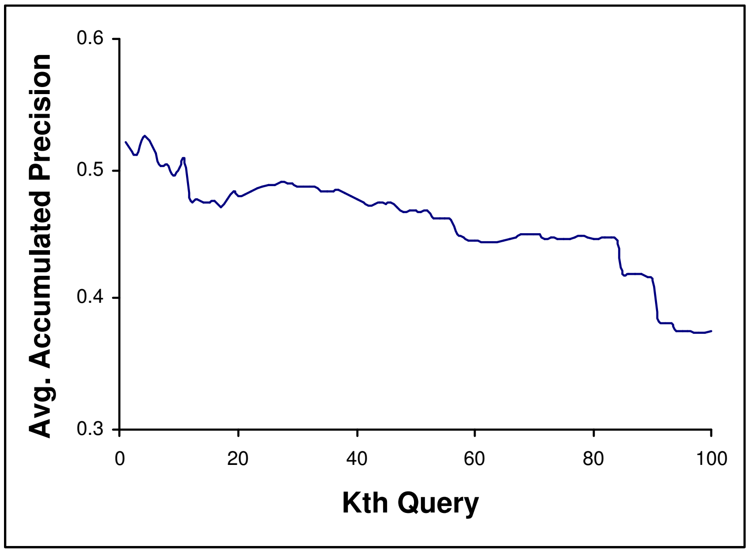

Figure 10. Accumulated precision curve for 30 test queries on Census database.

*precision* as follows:

$$\frac{\sum_{k=1}^{m} number\ of\ relevant\ answers\ in\ \widehat{RS}(Q_k)}{\sum_{k=1}^{m} number\ of\ answers\ in\ \widehat{RS}(Q_k)}$$

Figure 9 and 10 show the average accumulated precision of the 30 test queries on each database. We used the aggregate statistics over 30 queries to avoid biased information from a single query. As we can see, RRUA is able to generate a ranked list of rewritten queries, and the queries ranked higher tend to bring in uncertain answers that are more relevant. By choosing the top ranked rewritten queries to send to the database, the mediator can maximize the *precision* with any given cost restriction.

**3.4.7. Robustness of the RRUA Approach.** The performance of RRUA approach, in terms of precision and recall, relies on the quality of the AFDs and Naïve Bayesian classifiers learned by the knowledge mining module. In data integration systems, the availability of the sample training data from the autonomous data sources is restrictive. Usually

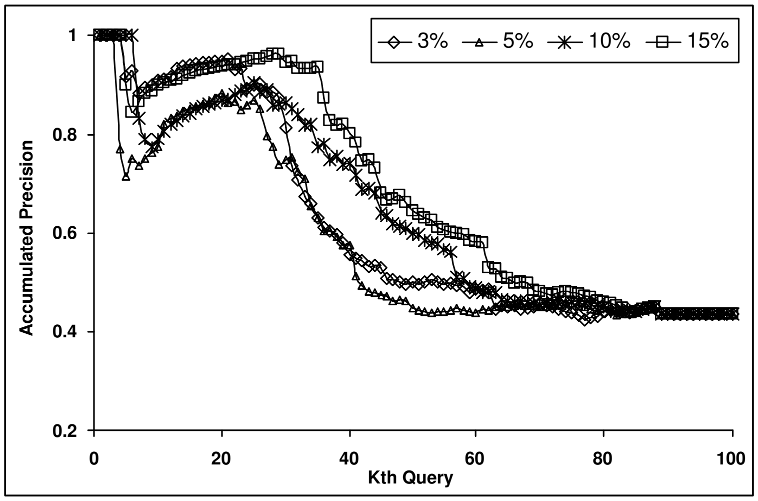

Figure 11. Accumulated precision curve for $\sigma_{body\ style=convt}$ with different sample sizes on Car database.

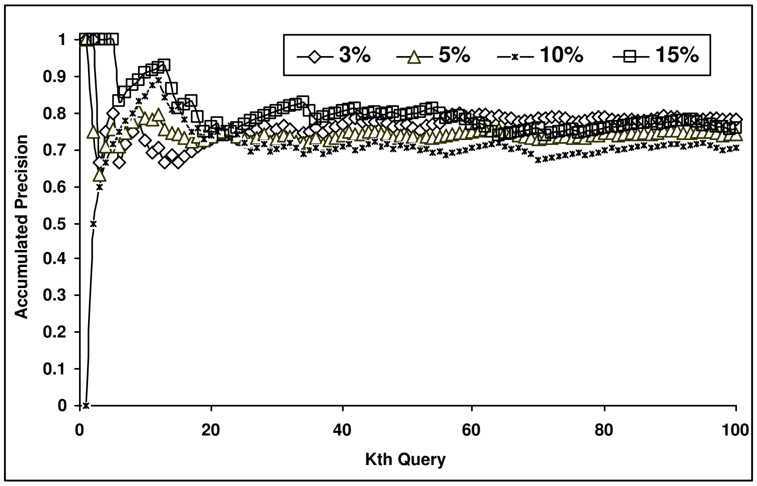

Figure 12. Accumulated precision curve for $\sigma_{workshop=private}$ with different sample sizes on Census database.

the mediator is only able to sample a rather small portion of the data from each source. Here we present the robustness of the RRUA approach in the face of limited sample data.

The performance of RRUA approach, in terms of precision and recall, relies on the quality of the AFDs and Naïve Bayesian Classifiers learned by the knowledge mining module. In data integration scenario, the availability of the sample training data from the autonomous data sources is restrictive. Here we present the robustness of the RRUA approach in the face of limited size of sample data.

Figure 11 and Figure 12 show the accumulated precision of a selection query on the Car and Census databases, using various sizes of sample data as training set. We see that the quality of the rewritten queries all fluctuate in a relatively narrow range and there is no significant drop of precision with the sharp decrease of sample size from 15% to 3%.

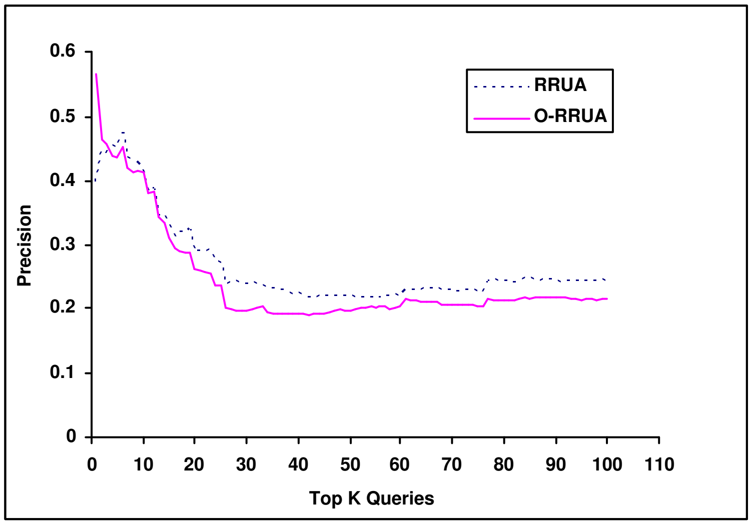

Figure 13. Comparison of accumulated precision across $RRUA$ and $O{-}RRUA$ over 40 test queries on the Car database

**3.4.8. Comparison of RRUA with O-RRUA.** Figure 13 compares the average accumulated precision over 40 test queries on the Cars database using the $RRUA$ and $O{-}RRUA$ approaches. The precision curves for both the approaches more or less overlap

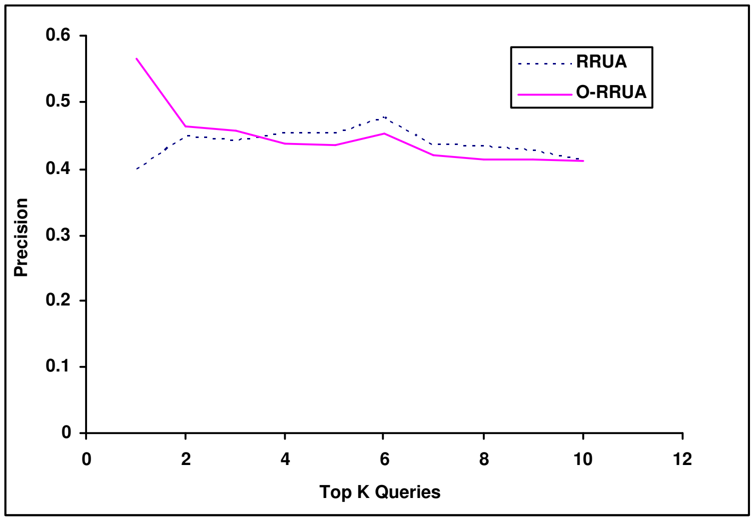

Figure 14. Comparison of accumulated precision across $RRUA$ and $O{-}RRUA$ zoomed over the top 10 rewritten queries

with $O{-}RRUA$ approach performing slightly better for the top 5 rewritten queries. Figure 14 zooms in to show the precision across these two approaches for top 10 rewritten queries. This shows that if the mediator were to issue only a few top queries due to limited network resources, using the online learning approach would result in higher precision. Since both $O{-}RRUA$ and $RRUA$ approach uses the same set of queries ranked in a different order, $RRUA$ approach slowly catches up with the $O{-}RRUA$ approach. Since both the approaches use attribute correlations learned using AFDs to retrieve only relevant results, both approaches retrieve answers with high precision. However, with no significant difference in precision between $RRUA$ and $O{-}RRUA$, the query time processing costs involved with $O{-}RRUA$ approach makes it less feasible for web scenarios. Hence, we use $RRUA$ approach for all our experiments.

CHAPTER 4

# DATA INTEGRATION OVER INCOMPLETE AUTONOMOUS DATABASES

In this chapter, we describe our approaches to perform data integration over incomplete autonomous databases in a mediator scenario. In many scenarios, the mediator supports queries on attributes in the global schema which are not supported in the local schema of some individual data sources. In this case, effectively all data sources not supporting the query attributes are ignored. We present query rewriting techniques to retrieve relevant uncertain tuples from such data sources based on attribute correlations and value distributions learned from other data sources. Many queries on the global schema of the mediator may require performing joins over individual data sources. We describe query rewriting techniques to perform joins over incomplete autonomous databases in order to present uncertain join results for the user query.

## 4.1. Leveraging Correlations among Data Sources

**4.1.1. Motivation.** The global schema supported by the mediator may often contains attributes which are not supported in some individual data sources. For example, consider a global schema $GS_{UsedCars}$ supported by the mediator over the sources *Cars.com*[7] and *Yahoo! Autos*[41] as shown in Figure 15. Since the form based interface of *Yahoo! Autos* doesn't support queries on *body style* attribute, the mediator cannot directly query the database in order to retrieve cars having a specific *body style*. Given a query $Q$: $\sigma_{body\ style=coupe}$ on global schema that selects cars of *body style* "coupe", approaches that only return certain answers won't be able to return car information from *Yahoo! Autos*. Hence many relevant cars from *Yahoo! Autos* wouldn't be shown to the user.

| | |
|---|---|
| Mediator | $GS_{UsedCars}(Make, Model, Year, Price, Mileage, Location, Body\ style)$ |
| Cars.com | $LS_{Autotrader}(Make, Model, Year, Price,\ Mileage, Location, Body\ style)$ |
| Yahoo! Autos | $LS_{Cars}(Make, Model, Year, Price,\ Mileage, Location)$ |

Figure 15. Global schema and local schema of data sources

We use AFDs and NBC classifiers which we learned on *Cars.com* to retrieve cars from *Yahoo! Autos* as possible ranked relevant answers to a query on *body style=coupe*. For example, consider the *Cars.com* database for which we have mined an AFD *model* $\rightsquigarrow$ *body style*. Note that this is an approximate functional dependency since a single model may have two different *body styles*. For example, the *model Civic* has both the *coupe* and *sedan body styles*. In order to retrieve relevant answers from the *Yahoo! Autos* database, the mediator issues rewritten queries to *Yahoo! Autos* based on the AFD and tuples retrieved from the base set results of *Cars.com*.

### 4.1.2. Retrieving Relevant Answers from Data Sources Not Supporting the Query Attribute.

Consider a mediator which performs data aggregation over autonomous sources $S_1, S_2, \cdots, S_n$. Let $LS_i$ denote the local schema for each source $S_i$ and $GS$ denote the global schema supported by the mediator over all the sources. Consider a query $Q$ on an attribute $A_m$ over the global schema $GS$. Let $S_s$ be the set of sources supporting attribute $A_m$ in their local schema $LS$. Let $\overline{S_s}$ be the set of sources not supporting attribute $A_m$ in their local schema $LS$. For each source $S_k \in \overline{S_s}$, we use AFDs for attribute $A_m$ from a *correlated source* $S_c \in S_s$ in order to retrieve relevant tuples from source $S_k$.

**Definition 5 (Correlated Source)** *For any autonomous data source $S_k$ not supporting the attribute $A_m$, we define a* correlated source $S_c$ *as any data source that satisfies the following: (i) $S_c$ supports attribute $A_m$ in its local schema (ii) $S_c$ has an AFD for $A_m$ (iii) $S_k$ supports the determining set of attributes in the AFD for $A_m$ mined from $S_c$.*

**Definition 6 (Maximum Correlated Data Source)** *From all sources correlated with a given source $S_k$, the source for which the AFD for $A_m$ has the highest confidence is called the maximum correlated source.*

For example, consider four autonomous sources $S_1$, $S_2$, $S_3$ and $S_4$ with local schema as $LS_1(E, F, G)$, $LS_2(A, B, C)$, $LS_3(C, E, F)$ and $LS_4(C, F, G)$ respectively. The global schema of the mediator is $GS(A, B, C, D, E, F, G)$. For queries on attribute $C$, in order to retrieve relevant tuples from source $S_1$ which doesn't support attribute $C$, we need to find correlated sources. Suppose sources $S_2$, $S_3$ and $S_4$ have AFDs $\{A, B\} \rightsquigarrow C(0.7)$, $\{E, F\} \rightsquigarrow C(0.8)$ and $\{F, G\} \rightsquigarrow C(0.6)$ respectively. Then only sources $S_3$ and $S_4$ are correlated with source $S_1$ because $S_1$ does not support attributes $A, B$ in its local schema. Among the two correlated sources, $S_3$ will be the maximum correlated source as the AFD for $C$ in $S_1$ has the highest confidence(0.8).

Algorithm 4 shows the details of the approach used to retrieve relevant tuples from the source not supporting the query attribute. Let $R(A_1, A_2, \cdots, A_n)$ be a database relation of the maximum correlated source $S_c$, and $dtrSet(A_m)$ be the determining set of an attribute $A_m$ $(1 \leq m \leq n)$, according to the best selected AFD (as discussed in Section 3.3.3). The algorithm processes a given selection query $Q : \sigma_{A_m=v_m}$ over the global schema $GS$ in the

**Algorithm 4** Rewriting queries to retrieve relevant uncertain answers from an autonomous data source not supporting the query attribute

```
Require: $R(A_1, A_2, \cdots, A_n)$: database relation of maximum correlated source $S_c$; $A_m$: attribute of query not supported in source $S_k$, $(1 \leq m \leq n)$; $Q : \sigma_{A_m=v_m}$: selection query over global schema $GS$;
 1: send $Q$ to the correlated data source $S_c$
 2: let $RS_{S_c}(Q)$= base result set returned for $Q$ on data source $S_c$
 3: for all $t_i \in \pi_{dtrSet(A_m)}(RS_{S_c}(Q))$ do
 4:    selection predicate $sp_{Q_i} = \emptyset$
 5:    for all $A_j \in dtrSet(A_m)$ do
 6:       $sp_{Q_i} = sp_{Q_i} +$ "$\wedge A_j = t_i(A_j)$"
 7:    $Q_i =$ "$\sigma_{sp_{Q_i}}$"
 8:    $P_{Q_i} = P(A_m = v_m | t_i)$
 9: rank $\{Q_i | t_i \in \pi_{dtrSet(A_m)}(RS_{S_c}(Q))\}$ according to $P_{Q_i}$
10: pick $top\ K\ queries$
    $(1 \leq K \leq ||\pi_{dtrSet(A_m)}(RS_{S_c}(Q))||)$
11: let result set for source $S_k$ be $\widehat{RS_{S_k}}(Q) = \emptyset$
12: for all $Q_i \in top\ K\ queries$ do
13:    send $Q_i$ to the data source $S_k$ and retrieve result set $RS_{S_k}(Q_i)$
14:    append $RS_{S_k}(Q_i)$ to $\widehat{RS_{S_k}}(Q)$
15: Return $\widehat{RS_{S_k}}(Q)$
```

following two steps.

1. Send $Q$ to the correlated data source $S_c$ and retrieve the base result set $RS_{S_c}(Q)$ as the answers of $Q$.

2. Generate a set of new queries, rank them, and send the most relevant ones to the data source $S_k$ not supporting the query attribute to retrieve the result set as relevant uncertain answers of Q. This step contains the following tasks.

   (a) *Generate rewritten queries.* Let $\pi_{dtrSet(A_m)}(RS_{S_c}(Q))$ be the projection of $RS_{S_c}(Q)$ onto $dtrSet(A_m)$. For each tuple $t_i$ in $\pi_{dtrSet(A_m)}(RS_{S_c}(Q))$, create a selection query $Q_i$ in the following way. For each attribute $A_j$ in $dtrSet(A_m)$, create a selection predicate $A_j$=$t_i(A_j)$. The selection predicates of $Q_i$ consist of the conjunction of all these predicates.

(b) *Rank rewritten queries.* Compute the conditional probability of $P_{Q_i}{=}P(A_m{=}v_m|t_i)$ for each $Q_i$. Rank all $Q_i$s according to their $P_{Q_i}$ values.

(c) *Retrieve extended result set.* Pick the top $K$ queries $\{Q_1, Q_2, \cdots, Q_K\}$ and issue them in the order of their ranks to data source $S_k$. Their result sets $RS_{S_k}(Q_1), RS_{S_k}(Q_2), \cdots, RS_{S_k}(Q_n)$ compose the result set $\widehat{RS_{S_k}}(Q)$. The results in $\widehat{RS_{S_k}}(Q)$ are ranked according to the ranks of the corresponding queries, i.e. the results in $RS_{S_k}(Q_1)$ are ranked higher than those in $RS_{S_k}(Q_2)$, and so on.[1]

### 4.1.3. Empirical Evaluation of using Correlation Between Data Sources.

In this section, we describe experiments that show the effectiveness of our approach of query rewriting using attribute correlations mined from one data source to retrieve the relevant uncertain answers from other data sources.

4.1.3.1. *Experimental Settings.* We consider a mediator supporting data aggregation over three data sources. The global schema supported by the mediator is $GS_{UsedCars}(Make, Model, Year, Price, Mileage, Location, Body\ \ style)$ The local schema of the individual source *Cars.com*[7] is $LS_{Cars}(Make, Model, Year, Price, Mileage, Location, Body\ \ style)$ and its form based interface supports queries on the *body style* attribute using Advanced Search[8]. The local schema of sources *Cars.com*[7] and *CarsDirect*[9] is $LS_{Yahoo,CarsDirect}(Make, Model, Year, Price, Mileage, Location)$ which do not support a query on *body style* in their form based interface. To evaluate the effectiveness of query rewriting approach in retrieving relevant answers for queries not supported by *Yahoo! Autos* and *CarsDirect*, we check the actual *body style* of the retrieved car tuple

[1]All results returned for a single query are ranked equally.

| **Query Number** | **Query** |
|---|---|
| 1 | body style=coupe |
| 2 | body style=sedan |
| 3 | body style=convt |
| 4 | body style=suv |
| 5 | body style=wagon |

Table 8. Test queries on the global schema on *body style* attribute

to determine whether the tuple was indeed a relevant answer to the original query. In order to retrieve relevant tuples from *Yahoo! Autos* and *CarsDirect*, we use 5 test queries on the attribute *body style* as shown in Table 8.

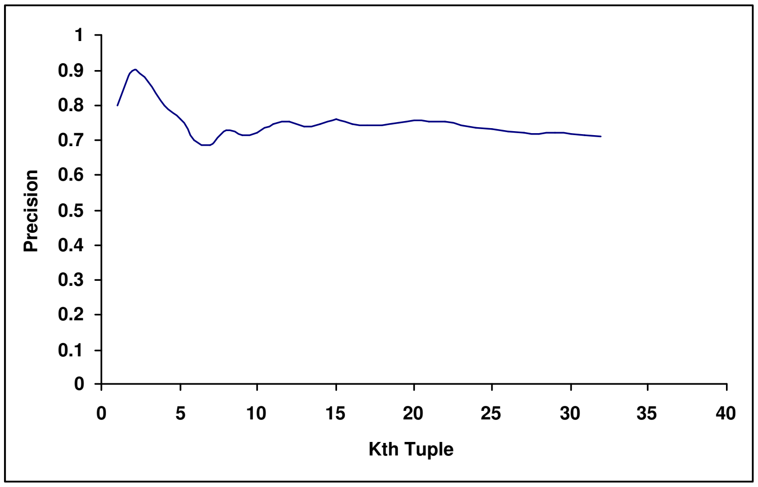

Figure 16. Precision curve for top $K$ tuples retrieved from *Yahoo! Autos* using AFDs learned from *Cars.om*

4.1.3.2. *Experimental Results.* We use AFDs and NBC classifiers learned from *Cars.com* to retrieve cars from *Yahoo! Autos* and *CarsDirect* as possible ranked relevant answers to a query on *body style*. We use the AFD *model* $\rightsquigarrow$ *body style* mined from a sample of *Cars.com* in order to retrieve tuples from *Yahoo! Autos* and *CarsDirect* using Algorithm 4. We measure the average precision for the top $K$ tuples retrieved from *Yahoo! Autos* and *CarsDirect* over the test queries. Figure 16 and 17 shows that our query rewriting technique is able to retrieve tuples with high precision from *Yahoo! Autos* and *CarsDirect*.

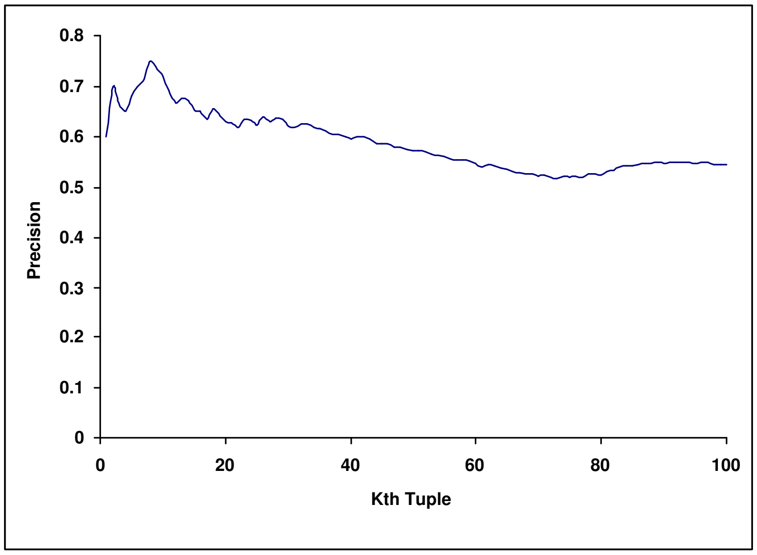

Figure 17. Precision curve for top $K$ tuples retrieved from *CarsDirect* using AFDs learned from *Cars.com*

This shows that our techniques of rewriting queries using AFDs and value distributions learned from correlated sources can be used to retrieve relevant answers from data sources not supporting query attribute.

## 4.2. Handling Joins over Incomplete Databases

Handling joins over autonomous databases is important for a mediator performing data integration over multiple heterogenous data sources. When the autonomous databases are complete, the mediator can use the PARTITION algorithm described in [42]. In this section, we describe our initial efforts to support join queries over incomplete autonomous databases under precise queries.

### 4.2.1. Motivation.

Consider two sources $S_1$ and $S_2$ providing information about used cars as shown in Figure 18 and a mediator that provides an integrated view as shown in Figure 19. Consider a user query asking for all cars made by *Honda* along with their

| Make | Model | Year | Price |
|---|---|---|---|
| Honda | Civic | 2000 | 10000 |
| Honda | Accord | 2004 | 20000 |
| Honda | null | 2000 | 15000 |
| null | Accord | 2002 | 18000 |
| Toyota | Camry | 2003 | 16000 |

| Model | Rating | Reliability |
|---|---|---|
| Civic | 5 | high |
| Corolla | 4 | medium |
| Accord | 4 | very high |
| Camry | 5 | high |
| A4 | 3 | low |

Figure 18. Autonomous sources $S_1(Cars)$ and $S_2(Review)$

| Mediator View: |
|---|
| $UsedCars(Make, Model, Year, Price, Rating, Reliability) :-$ $Cars(Make, Model, Year, Price),\ Review(Model, Rating, Reliability)$ |

Figure 19. Mediator view over sources $S_1$ and $S_2$

*ratings*. If both sources $S_1$ and $S_2$ are complete, then the mediator would first retrieve all tuples having *make=Honda* from source $S_1$. Then it would retrieve tuples from source $S_2$ corresponding to the *models* of the tuples returned from source $S_1$. Finally, it would combine the tuples from the two sources using the join attribute *model* and return the results to the users.

If the sources are incomplete, then above approach is not able to retrieve join results for car tuples having missing *make* in source $S_1$ but still might be relevant to the user query. Also, if the join attribute *model* is missing for some tuples having *make=Honda* in source $S_1$, the mediator cannot join that tuple with a ratings tuple from source $S_2$. We would like to present such *uncertain join results* to the users in a ranked order based on their likelihood of being a relevant answer to the user query. Since the data sources are autonomous, we need query rewriting techniques to retrieve tuples containing null values and to join tuples from the two sources in order to obtain uncertain join results. We also need ranking techniques to rank the join tuples based on their relevance to the user query.

**4.2.2. Issues in Handling Joins over Incomplete Databases.** Since incomplete databases are similar to probabilistic databases once the probabilities for the missing values are assessed, we first discuss the issues involved in performing joins over probabilistic databases. Consider an incomplete tuple with a missing *model* value from the Cars database and two complete tuples from the Review database as shown in the table below.

| **Make** | **Model** | **Year** | **Price** |
|---|---|---|---|
| Honda | null[0.6 Civic 0.4 Accord] | 2003 | 18000 |

| **Model** | **Ratings** |
|---|---|
| Civic | 5 |
| Accord | 4 |

| **Make** | **Model** | **Year** | **Price** | **Ratings** | **Probability** |
|---|---|---|---|---|---|
| Honda | Civic | 2003 | 18000 | 5 | 0.6 |
| Honda | Accord | 2003 | 18000 | 4 | 0.4 |

Given that the missing *model* value has 0.6 probability to be Civic and 0.4 Accord, the join of this incomplete tuple with a tuple from Review table on the *foreign key model* will lead to a disjunction between the tuples in the result set. Each tuple in the disjunction is associated with a probability corresponding to the probability value of the missing attribute. In web scenarios, users find it hard to read disjunctive tuples. Hence we perform an *approximation* on the set of disjunctive tuples and only show the tuple with the highest probability from a set of disjunctive tuples. We will use this approximation for performing joins over incomplete autonomous databases as described in the next section. Note that this approximation is performed at the end when an uncertain join result is to be presented to the user. In all intermediate join results we keep the disjunction over tuples without performing an approximation so as to get the correct maximum probability value. Current databases use RUA approach to perform joins over incomplete databases where first all incomplete tuples are retrieved from one relation and then joined with tuples from other relation. We describe a query rewriting approach to retrieve ranked uncertain join answers based on their likelihood of being a relevant answer to the query.

**4.2.3. Joins over Incomplete Databases.** To keep the discussion simple we consider single attribute selection queries over global schema $GS(A_1, A_2, \cdots, A_n)$ of the mediator performing data integration over an incomplete source $S_1$ and a complete source $S_2$. The relation of incomplete source $S_1$ is $LS_1(A_1, A_2, \cdots, A_{j-1}, A_j)$ and the relation of complete source $S_2$ is $LS_2(A_j, A_{j+1}, \cdots, A_n)$ with $A_j$ as the primary key. Consider a single attribute selection query[2] $Q{:}\, \sigma_{A_q=v_q}$ over global schema $GS$. Here, we have three possibilities for the selection query attribute. For each case, we describe how our RRUA approach retrieves highly relevant uncertain join results in a ranked order.

**Case 1**: $A_q{=}A_j$

In this case, the query attribute is the same as the join attribute. For example, consider a query $Q{:}\, \sigma_{model=Civic}(UsedCars)$ over global schema. We first retrieve complete tuples from source $S_1$ which satisfy the user query $Q$ which form the base set. We also retrieve tuples from $S_2$ that satisfy the query attribute value. Next, we combine the two results based on the join attribute $A_j$ to get certain join results. In order to present uncertain join results to the user we need to retrieve tuples from source $S_1$ containing missing values on the join attribute. We use our query rewriting techniques, as described in Section 3.2, to retrieve ranked results from source $S_1$. These incomplete tuples are then joined with tuples retrieved from source $S_2$ on attribute $A_j$ to get uncertain join results. The uncertain join results are ranked based on the rank of the rewritten queries that retrieved the incomplete tuple corresponding to source $S_1$ having missing value on the $A_j$.

[2]Note that here we only consider general selection queries projected on all attributes in global relation $GS$ as it is most likely to be in web scenarios.

**Algorithm 5** Algorithm to perform joins over incomplete autonomous databases

```
Require: $GS(A_1, A_2, \cdots, A_n)$: global schema supported by mediator; $LS_1(A_1, A_2, \cdots, A_{j-1}, A_j)$: relation of incomplete source $S_1$; $LS_2(A_j, A_{j+1}, \cdots, A_n)$: relation of complete source $S_2$ with primary key $A_j$; $A_j$: join attribute of source $S_1$ and $S_2$; $Q$:$\sigma_{A_q=v_q}$:selection query over global schema $R$;
 1: initialize uncertain join result set $JRS = \emptyset$
 2: if $A_j = A_q$ then
 3:   send $Q$ to the $S_1$ and $S_2$ to get base result sets $RS_1(Q)$ and $RS_2(Q)$ resp.
 4:   use Algorithm 1 to retrieve extended result set $ERS_1(Q)$ using base set $RS_1(Q)$
 5:   for all $t_k \in ERS_1(Q)$ do
 6:     use hash joins to join tuple $t_k$ with result set $RS_2(Q)$ on attribute $A_j$
 7:     rank each uncertain join tuple $t_j$: $R(t_j)$ = rank of rewritten query that retrieved $t_k$
 8:     add $t_j, R(t_j)$ to $JRS$
 9: else
10:   if $A_j \neq A_q$ and $A_q \in LS_1$ and $A_q \notin LS_2$ then
11:     send $Q$ to the data source $S_1$ to get base result set $RS_1(Q)$
12:     for all $t_i \in \pi_{dtrSet(A_j)}(RS_1(Q))$ having $t_i(A_j) = null$ do
13:       $P_{t_i} = MAX\{P(A_j = v_t|t_i); 1 \leq t \leq |A_j|\}$
14:       $V_{t_i} = \{v_t|P(A_j = v_t|t_i)\ is\ maximum\}$
15:       if $ERS_2(V_{t_i})$ does not exist then
16:         send query $Q$:$\sigma_{A_j=V_{t_i}}$ to data source $S_2$ to get the result set $ERS_2(V_{t_i})$
17:       use hash joins to join tuple $t_i$ with result set $ERS_2(V_{t_i})$ on attribute $A_j$
18:       rank each uncertain join tuple $t_j$:$R(t_j) = P_{t_i}$
19:       add $t_j, R(t_j)$ to $JRS$
20:     use Algorithm 1 to retrieve extended result set $ERS_1(Q)$ using base set $RS_1(Q)$
21:     for all $t_k \in ERS_1(Q)$ do
22:       if $ERS_2(t_k(A_j))$ does not exist then
23:         send query $Q$:$\sigma_{A_j=t_k(A_j)}$ to data source $S_2$ to get the result set $ERS_2(t_k(A_j))$
24:       use hash joins to join tuple $t_k$ with result set $ERS_2(t_k(A_j))$ on attribute $A_j$
25:       rank each uncertain join tuple $t_j$: $R(t_j)$ = rank of the rewritten query that retrieved $t_k$
26:       add $t_j, R(t_j)$ to $JRS$
27: else
28:   if $A_j \neq A_q$ and $A_q \in LS_2$ and $A_q \notin LS_1$ then
29:     send $Q$ to the data source $S_2$ to get base result set $RS_2(Q)$
30:     for all $V_i \in \pi_{(A_j)}(RS_2(Q))$ do
31:       if $ERS_1(V_i)$ does not exist then
32:         send query $Q'$:$\sigma_{A_j=V_i}$ to data source $S_1$ to get base result set $RS_1(Q')$)
33:       use Algorithm 1 to retrieve extended result set $ERS_1(V_i)$ using base set $RS_1(Q')$
34:       for all $t_k \in ERS_1(V_i)$ do
35:         rank each tuple $t_k$: $R(t_k)$ = Rank of rewritten query that retrieved $t_k$
36:         add $t_k, R(t_k)$ to $ERS_1$
37:     for all $t_k \in ERS_1$ do
38:       use hash joins to join $t_k$ with $RS_2(Q)$ on maximum probability value for $A_j$ in $t_k$
39:       rank each uncertain join tuple $t_j$: $R(t_j)$ = maximum rank of $R(t_k)$ among all queries that
          retrieved $t_k$
40:       add $t_j, R(t_j)$ to $JRS$
41: sort the uncertain join results in $JRS$ based on their rank and display them
```

**Case 2**: $A_q \neq A_j$ **and** $A_q \in LS_1$ **and** $A_q \notin LS_2$

In this case, the query attribute is not the join attribute and the query attribute belongs only to the relation $LS_1$ of source $S_1$. For example, consider a query $Q$: $\sigma_{make=Honda}(UsedCars)$ over the global schema. In order to present uncertain join results, we need to consider two types of tuples from the incomplete data source: (i) Tuples having a missing value on the query constrained attribute and (ii) Tuples from the base set having a missing value on the join attribute. We will use query rewriting techniques, discussed in Section 3.2, in order to retrieve tuples having missing value on the query constrained attribute. We then join these tuples with the tuples of source $S_2$ based on the join attribute. The uncertain join results are ranked based on the rank of the rewritten queries that retrieved them. For tuples in the base set having a missing value on the join attribute, we predict the value of null as the value having maximum probability using NBC classifiers as described in Section 3.3.2. We then use the predicted value to join it with the corresponding tuple(s) of source $S_2$. In this case, the uncertain joins results are then ranked based on the maximum probability value of the join attribute.

**Case 3**: $A_q \neq A_j$ **and** $A_q \in LS_2$ **and** $A_q \notin LS_1$

In this case, the query attribute is not the join attribute and the query attribute belongs only to the relation $LS_2$ of source $S_2$. For example, consider a query $Q$: $\sigma_{Rating=4}(UsedCars)$ over the global schema. First we issue the query $Q$: $\sigma_{A_q=v_q}$ to source $S_2$ to get the base set. We then retrieve values of join attribute $A_j$ for all the tuples in the base set. We need to join these tuples with the corresponding tuples of source $S_1$ on $A_j$. There may be tuples in the source $S_1$ having a missing value on the join attribute but would produce a relevant join result. We retrieve such tuples having a missing value on the join attribute using the query rewriting techniques discussed in Section 3.2. Notice that it is possible for an incomplete tuple to be present in extended results result set of more than one rewritten query. In this case, the rank of the tuple will be equal to the maximum of the ranks of the rewritten queries that retrieved it. The value having the maximum probability for null attribute $A_j$ will be used to join results with those from source $S_2$. We then present ranked uncertain join results in response to the original query on the global schema.

Algorithm 5 describes the details of our approach for performing joins over incomplete data sources. This algorithm can be extended to handle joins over two incomplete databases by using similar query rewriting techniques to retrieve uncertain relevant results from both the databases and then joining them based on the predicted join value. The rank of the uncertain join results would be the product of the rank of the individual tuples forming the join result from source $S_1$ and $S_2$.

**4.2.4. Empirical Evaluation of Joins over Incomplete Databases.** In this section, we show the experimental results which illustrate the effectiveness of our approach to perform joins over autonomous incomplete databases.

4.2.4.1. *Experimental Settings.* We consider a mediator supporting data aggregation over two data sources. The global schema supported by the mediator is $GS(Make, Model, Year, Price, Mileage, Location, Rating)$. We consider two individual sources:*AutoTrader* which gives a description of used cars and contains $100,000$ tuples as used in Section 3.4.2 and a synthetic database which gives user ratings for *models* of cars containing reviews for 600 models. The local schema of the individual source *AutoTrader*[3] is $LS_{AutoTrader}(Make, Model, Year, Price, Mileage, Location)$ and local schema of the synthetic source *Review* is $LS_{Review}(Model, Rating)$ with *model* as the primary key. We consider test queries of three types as described in Section 4.2.3.

4.2.4.2. *Experimental Results.* We evaluate the effectiveness of our approach by measuring precision-recall for the uncertain join answers for the test queries. We compare our RRUA based query rewriting approach which retrieves uncertain join results in a ranked order based on their relevance to the user query to the RUA approach which returns all uncertain joins results without query rewriting.

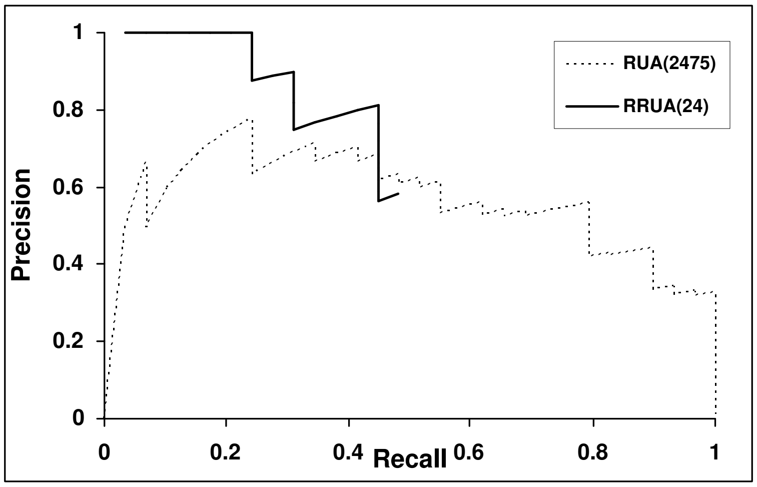

Figure 20. Precision-Recall curve for the query $\sigma_{model=Civic}$ on global schema $R$

Figures 20, 21 and 22 show the precision recall curves for uncertain join results over the incomplete databases for the test queries $\sigma_{model=Civic}$, $\sigma_{make=Audi}$ and $\sigma_{ratings=4}$ for RUA and RRUA approaches. These queries are issued on the global schema $GS$. The figures in the parentheses in the legend represent the number tuples retrieved by these approaches. As we can see from these figures that RRUA retrieves only a fraction of tuples retrieved by RUA with high precision. Thus our join algorithm based on RRUA is able to retrieve uncertain join results with high precision. Even though our approach cannot achieve 100% recall, the precision is reasonably high for the retrieved tuples and in web scenarios only top $K$ uncertain join answers are to be displayed to the user.

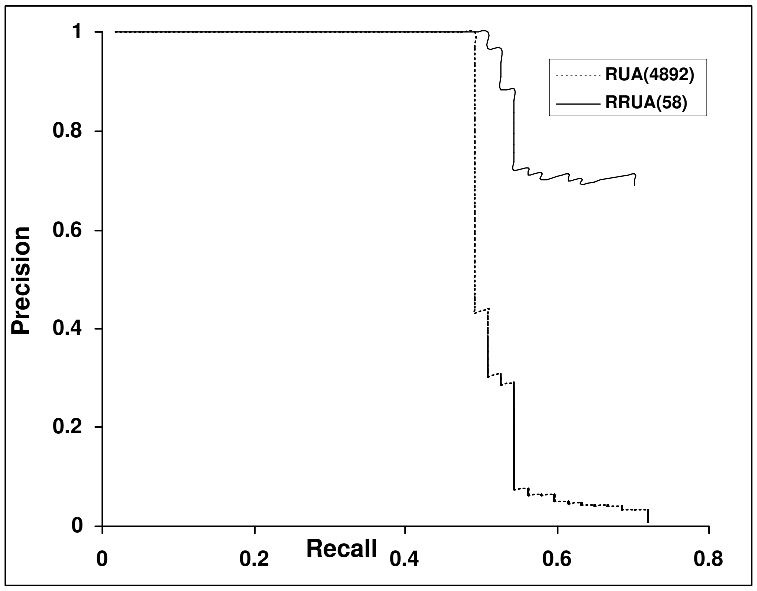

Figure 21. Precision-Recall curve for the query $\sigma_{make=Audi}$ on global schema $R$

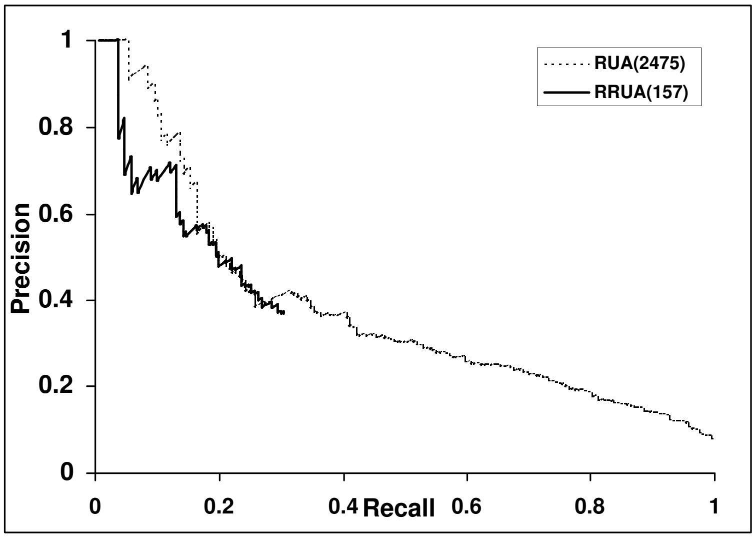

Figure 22. Precision-Recall curve for the query $\sigma_{Rating=4}$ on global schema $R$

CHAPTER 5

# RELATED WORK

**Querying incomplete databases:** Compared to previous work on querying incomplete databases, the critical novelty of our work is that our approach does not modify original data. It is therefore suitable for querying incomplete autonomous databases, where a mediator is not able to store the estimation of missing values in sources. Techniques have been studied to process queries on databases with null values [21]. Null values are typically represented in three different approaches:(i) Codd Tables where all the null values are treated as the same; (ii) V-tables which allow many different null values marked by variables; and (iii) Conditional tables which are V-tables with additional attributes for conditions. [26] proposed a query language on incomplete databases, where we can have a subset of the domain as an attribute value. [31, 11] discussed query evaluation on incomplete databases where attribute values can be intervals. In all the above approaches, data sources need to be modified, and standard relational data model and query languages need to be extended in order to handle incomplete data. In contrast, our work proposes online query rewriting techniques to query incomplete autonomous databases without modifying source data and data models.

**Probabilistic Databases:** Incomplete databases are similar to probabilistic databases once the probabilities for missing values are assessed. [37] gives an overview of querying probabilistic databases where each tuple is associated with an additional attribute describing the probability of its existence. Some recent work on the TRIO[36, 39] system deals with handling uncertainty over probabilistic relational databases. ConQuer[16, 2] system returns clean answers over inconsistent databases. Handling inconsistency in databases is a special case of handling missing values where the inconsistent attribute can only take values that lead to inconsistency rather than any possible value. These approaches assume the presence of probabilities for missing data. Since autonomous databases do not store or allow mediators to store probability distribution, our approach assesses these probabilities in order to issue rewritten queries to retrieve relevant answers. Thus, our query rewriting techniques could also be used by these systems if the databases are autonomous.

**Query relaxation:** Reformulating queries using database constraints for query optimization in distributed mediator systems has been described in [32]. There has been work on query relaxation over databases [30, 29] which focuses on how to relax input query constraints such that data that *partially* satisfies the query constraints is also returned. Our work is similar to such efforts in the spirit of retrieving relevant data even when it does not exactly satisfy user queries. However, we focus on retrieving data that has missing values in the query constrained attributes, yet is likely to be relevant to the original user query. Since we use an entirely new set of rewritten queries to retrieve uncertain answers, our query relaxation is value directed.

**Learning missing values:** There has been a large body of work on missing values imputation [14, 33, 35, 40, 4, 27]. Common imputation approaches include substituting missing data values by the mean, the most common value, default value of the attribute in question, or using k-Nearest Neighbour[4], association rules[40], conditioned sets[27] etc. Other approach used to estimate missing values is *parameter estimation.* Maximum likelihood procedures that use variants of the Expectation-Maximization algorithm[14, 33] can be used to estimate the parameters of a model defined for the complete data. In this thesis, we are interested not in the standard imputation problem but a variant that can be used in the context of query rewriting. In this context, it is important to have schema level dependencies between attributes as well as distribution information over missing values. We use AFDs for the former, and an AFD-enhanced Naïve Bayes Classifier for the later. We experimented with other methods including association rules and bayes network learning - but found them to be either significantly less accurate or significantly costlier to compute.

**Keyword search over relational databases:** In recent years, there has been a huge interest on ranking answers in the context of keyword queries over relational databases[1, 5, 19, 10, 13]. However, all these efforts assume that the databases are complete which may not be true in the context of web databases. Moreover, their focus is mainly on local databases over which the query processor has full control. In contrast, we consider incomplete autonomous web databases providing limited access to a mediator through a form based interface which proliferates the web these days.

CHAPTER 6

# DISCUSSION AND FUTURE WORK

In this chapter, we discuss future directions for the work presented in this thesis. We describe how to extend our current system handling only single attribute selection queries to general queries involving multi-attribute selection queries along with multiple nulls per tuple. We also describe the need to handle incompleteness and imprecision over incomplete autonomous databases.

## 6.1. Handling General Queries

To support queries with selections on multiple attributes, we need to address the following challenges. First, assessing the relevance function over all the subset of attributes can be expensive. This requires exploring appropriate relevance independence assumptions over attributes. Second, assessing the density function can be difficult since the values of multiple attributes can be missing and those attributes may interact with each other. For example, if a user is interested in searching cars where *model* is *Explorer* and *body style* is *SUV*. Since *body style* highly depends on model as we learned from the AFDs and Bayesian Nets, it is not straightforward to assess the density for a tuple that has missing values on both *model* and *body style*. We can explore two options to infer correlated attribute values and compare their performance. One is to use inferences of attribute causal relationship. For example, we predict the possible values of *model*, based on which we further predict the possible values of *body style*. However, the prediction accuracy degrades after each inference step. The option is to explore additional attribute causal relationships. For example, if the *engine* is $V8$, then the *body style* is likely to be $SUV$. The prediction accuracy depends on the availability of other causal relationship with high accuracy and the existence of relevant values.

In order to support selection queries having multiple attributes bound, we need to extend our query rewriting techniques in order to retrieve tuples containing nulls on different query constrained attributes. First, we retrieve tuples containing nulls on only one of the query attributes while satisfying all other query constrained attributes. Such tuples containing nulls on different attributes could be ranked based on the probability of null being equal to the query constrained attribute. Next, if sufficient number of results are not obtained, we retrieve all tuples with two query attributes having null values. All such tuples must be ranked lower than tuples having missing values on only one attribute of the query predicate. We continue in similar fashion until we have retrieved a sufficient number of uncertain answers or have considered all possible sizes of subsets of query constrained attributes having null values.

## 6.2. Supporting Imprecise Queries over Incomplete Autonomous Databases

More and more lay users are accessing databases on the web and often, these lay users are not able to articulate their queries in a precise manner. Instead, the *imprecise* queries they pose only approximately characterize the type of answer tuples they are looking

| **Id** | **Make** | **Model** | **Year** | **Color** | **Body style** |
|---|---|---|---|---|---|
| 1 | Honda | Civic | 2000 | red | sedan |
| 2 | Honda | Accord | 2004 | blue | coupe |
| 3 | Toyota | Camry | 2001 | silver | sedan |
| 4 | Honda | null | 2004 | black | coupe |
| 5 | BMW | 3-series | 2001 | blue | convt |
| 6 | Honda | Civic | 2004 | green | coupe |
| 7 | Honda | null | 2000 | white | sedan |
| 8 | Toyota | null | 1999 | beige | sedan |

Table 9. Fragment of a Car Database

for. For example, the query $Q$:CarDB($model \approx Civic$) is an imprecise query, the answers to which should have a value similar to *Civic* for the *model* attribute.

In both cases, there are some tuples which do not exactly satisfy the query constraints but nevertheless are likely to be relevant to the user.

**Example:** Consider a fragment of online Car database $DB_f$ as shown in Table 9. Suppose a user poses an imprecise query $Q'$: $\sigma_{model \approx Civic}$ on this table. Clearly tuples 1 and 6 are relevant as they are exact answers to the query. In addition, the tuples 4 and 7 might be relevant if there are reasons to believe that the missing value might be a Civic. Finally, tuples 2 and 3 might be relevant if Civic is considered similar enough to Accord and/or Camry.

Ideally, a query processor should be able to present such implicitly relevant results to the user, ranking them in terms of their expected relevance. Query processing in this context has commonalities with query processing in probabilistic databases and handling imprecise queries. However, to support query processing under imprecision and incompleteness brings forth many challenges the primary of which is to rank results in presence of both incomplete and similar tuples.

CHAPTER 7

# CONCLUSION

Incompleteness is inevitable in autonomous web databases. Retrieving highly relevant uncertain answers from such databases is challenging due to the restricted access privileges of mediator, the limited query patterns supported by autonomous databases, and the sensitivity of database and network workload in web environment. We developed a novel query rewriting technique that tackles these challenges. Our approach involves rewriting the user query based on the knowledge of database attribute correlations. The rewritten queries are then ranked by leveraging attribute value distributions according to their likelihood of retrieving relevant uncertain answers before they are posed to the databases. To support such query processing techniques, we developed methods to mine attribute correlations in the form of AFDs and the value distributions of AFD-enhanced classifiers, from a small sample of the database itself. Our primary technique RRUA, is able to retrieve relevant uncertain answers from databases that don't allow null value binding. It's query ranking procedure is able to provide a spectrum of cost-quality trade-offs. Our comprehensive experiments demonstrate the effectiveness of our query processing and knowledge mining techniques.

We also described experiments for performing data integration over multiple incomplete data sources by supporting join queries and leveraging the correlation among data sources to retrieve relevant tuples from a source not supporting the query attribute.

In summary, this thesis presented techniques to support mediated query processing over incomplete autonomous web databases.

# REFERENCES

[1] S. Agrawal, S. Chaudhuri, and G. Das. Dbxplorer: A system for keyword-based search over relational databases. In *ICDE*, pages 5–16, 2002.

[2] P. Andritsos, A. Fuxman, and R. J. Miller. Clean answers over dirty databases: A probabilistic approach. In *ICDE*, page 30, 2006.

[3] AutoTrader. http://www.autotrader.com, 2006.

[4] G. E. A. P. A. Batista and M. C. Monard. A study of k-nearest neighbour as an imputation method. In *HIS*, pages 251–260, 2002.

[5] G. Bhalotia, C. Nakhe, A. Hulgeri, S. Chakrabarti, and S. Sudarshan. Keyword searching and browsing in databases using BANKS. In *ICDE*, 2002.

[6] A. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97(1-2):245–271, 1997.

[7] Cars. http://www.cars.com, 2006.

[8] Cars.com Advanced Search. http://www.cars.com/go/search/advance_search.jsp, 2006.

[9] CarsDirect. http://www.carsdirect.com, 2006.

[10] S. Chaudhuri, G. Das, V. Hristidis, and G. Weikum. Probabilistic ranking of database query results. In *Proceedings 2004 VLDB Conference : The 30th International Conference on Very Large Databases (VLDB)*, pages 888–899, 2004.

[11] R. Cheng, D. V. Kalashnikov, and S. Prabhakar. Evaluating probabilistic queries over imprecise data. In *SIGMOD Conference*, 2003.

[12] M. M. Dalkilic and E. L. Roberston. Information dependencies. In *PODS Symposium*, pages 245–253, 2000.

[13] G. Das, V. Hristidis, N. Kapoor, and S. Sudarshan. Ordering the attributes of query results. In *SIGMOD Conference*, 2006.

[14] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via em algorithm. In *JRSS*, pages 1–38, 1977.

[15] A. Doan, P. Domingos, and A. Y. Halevy. Reconciling schemas of disparate data sources: A machine-learning approach. In *SIGMOD Conference*, 2001.

[16] A. Fuxman, E. Fazli, and R. J. Miller. Conquer: Efficient management of inconsistent databases. In *SIGMOD Conference*, pages 155–166, 2005.

[17] D. Heckerman. A tutorial on learning with bayesian networks, 1995.

[18] Househunt. http://www.househunt.com, 2006.

[19] V. Hristidis, L. Gravano, and Y. Papakonstantinou. Efficient ir-style keyword search over relational databases. In *VLDB*, pages 850–861, 2003.

[20] Y. Huhtala, J. Krkkinen, P. Porkka, and H. Toivonen. Efficient discovery of functional and approximate dependencies using partitions. In *ICDE Conference*, pages 392–401, 1998.

[21] T. Imieliski and J. Witold Lipski. Incomplete information in relational databases. *Journal of ACM*, 31(4):761–791, 1984.

[22] Z. G. Ives, A. Y. Halevy, and D. S. Weld. Adapting to source properties in processing data integration queries. In *SIGMOD Conference*, 2004.

[23] J. Kivinen and H. Mannila. Approximate dependency inference from relations. In *ICDT Conference*, pages 86–98, 1992.

[24] T. T. Lee. An information-theoretic analysis of relational databases - part i: Data dependencies and information metric. *IEEE Transactions on Software Engineering*, 13(10):1049–1061, 1987.

[25] D. Lembo, M. Lenzerini, and R. Rosati. Source inconsistency and incompleteness in data integration. In *KRDB Workshop*, 2002.

[26] W. Lipski. On semantic issues connected with incomplete information databases. *ACM TODS*, 4(3):262–296, 1979.

[27] S.-C. C. Mei-Ling Shyu, Indika Priyantha Kuruppu-Appuhamilage and L. Chang. Handling missing values via decomposition of the conditioned set. In *IEEE IRI Conference*, 2005.

[28] T. Mitchell. *Machine Learning.* McGraw Hill, 1997.

[29] I. Muslea and T. J. Lee. Online query relaxation via bayesian causal structures discovery. In *AAAI*, pages 831–836, 2005.

[30] U. Nambiar and S. Kambhampati. Answering imprecise queries over autonomous web databases. In *ICDE*, 2006.

[31] A. Ola and G. Özsoyoglu. Incomplete relational database models based on intervals. *IEEE Transactions on Knowledge and Data Engineering*, 5(2):293–308, 1993.

[32] L. Popa and V. Tannen. An equational chase for path-conjunctive queries, constraints, and views. In *ICDT*, 1999.

[33] M. Ramoni and P. Sebastiani. Robust learning with missing data. *Mach. Learn.*, 45(2):147–170, 2001.

[34] Realtors. http://www.realtor.com, 2006.

[35] D. B. R. Roderick J. A. Little. *Statistical Analysis with Missing Data,Second edition.* Wiley, 2002.

[36] A. D. Sarma, O. Benjelloun, A. Y. Halevy, and J. Widom. Working models for uncertain data. In *ICDE*, 2006.

[37] D. Suciu and N. Dalvi. Tutorial: Foundations of probabilistic answers to queries. In *SIGMOD Conference*, 2005.

[38] UCI Machine Learning Repository. http://www.ics.uci.edu/mlearn/mlrepository.html.

[39] J. Widom. Trio: A system for integrated management of data, accuracy, and lineage. In *CIDR*, pages 262–276, 2005.

[40] C.-H. Wu, C.-H. Wun, and H.-J. Chou. Using association rules for completing missing data. In *HIS Conference*, pages 236–241, 2004.

[41] Yahoo! Autos. http://autos.yahoo.com, 2006.

[42] R. Yerneni, C. Li, J. Ullman, and H. Garcia-Molina. Optimizing large join queries in mediation systems. *Lecture Notes in Computer Science*, 1540:348–364, 1999.