QPIAD: Query Processing over Incomplete Autonomous Databases

Hemal Khatri Jianchun Fan Yi Chen Subbarao Kambhampati
Department of Computer Science and Engineering
Arizona State University
{hemal.khatri,jcf,yi,rao}@asu.edu

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 a user query. Ideally we would like the mediator 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 the restricted access privileges, limited query patterns and sensitivity of database and network resource consumption in the web environment. We introduce a novel query rewriting and optimization framework that tackles these challenges. Our technique involves reformulating the user query based on approximate functional dependencies (AFDs) among the database attributes. The reformulated queries are aimed at retrieving the relevant uncertain answers in addition to the certain answers. Our query processing framework QPIAD is able to gauge the relevance of such reformulated queries to manage the cost of database query processing and answer transmission. To support this framework, we develop methods for mining attribute correlations (in terms of AFDs) and value distributions(using Naïve Bayes Classifiers). We present empirical studies to demonstrate that our approach is effective in retrieving relevant uncertain answers with high precision, high recall and manageable cost.

1 Introduction

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 [14, 11] return to user only *certain answers* that exactly satisfy all the user query predicates. Tuples that are otherwise highy 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*.

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

Table 1. Statistics on missing values in web databases

Table 1 shows statistics on the percentage of incomplete

[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.

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

tuples on two autonomous 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.

When faced with such incomplete 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.

QPIAD Approach: In this paper, 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 [10, 15, 18, 23]. 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 the relevant query answer set. We call this approach as returning *relevant uncertain answers(RUA)*. However, this approach may be inappropriate as discussed before due to the limited query access pattern of web databases and high network transmission costs.

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.

QPIAD also provides support for handling incompleteness caused by schema heterogeneity (see above). Specifically, when faced with data sources not supporting the query attribute (e.g. $body\ style$ attribute by *Yahoo! Autos*), QPIAD leverages the mined approximate functional dependencies from correlated data sources (such as *Cars.com*) to rewrite the query and retrieve relevant answers.

Contributions: To the best of our knowledge, *QPIAD* framework is the first that retrieves relevant but uncertain answers with missing values on constrained attributes without modifying underlying databases. Consequently it is suitable for querying incomplete autonomous databases. The idea of using attribute correlations and predicted distributions on missing values to rewrite queries is also a novel contribution of our work. Our framework also describes a novel application of leveraging attribute correlations among data sources in order to retrieve relevant answers from data sources not supporting the query attribute. Our experimental evaluation shows that *QPIAD* retrieves most relevant information while keeping the query processing costs low.

Organization: The rest of the paper is organized as follows. In the next section we describe related work on handling incomplete data. In Section 3, we give some preliminaries and describe the architecture of QPIAD. Section 4 discusses online query rewriting and ranking techniques

[2]QPIAD is an acronym for Query(Q) Processing(P) over Incomplete(I) Autonomous(A) Databases(D)

to retrieve relevant uncertain answers from incomplete autonomous databases and from data sources not supporting the query attribute in their local schema. Section 5 provides the details of computing attribute correlations and missing value distributions used in our query rewriting phase. A comprehensive empirical evaluation of our approach is presented in Section 6. We conclude the paper in Section 7.

2 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 [10]. 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. [15] proposed a query language on incomplete databases, where we can have a subset of the domain as an attribute value. [18, 4] 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. [23] 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[22, 24] system deals with handling uncertainty over probabilistic relational databases. ConQuer[7, 1] 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 [19]. There has been work on query relaxation over databases [17] 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 [5, 20, 21, 25, 2]. 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[2], association rules[25], etc. Other approach used to estimate missing values is *parameter estimation*. Maximum likelihood procedures that use variants of the Expectation-Maximization algorithm[5, 20] can be used to estimate the parameters of a model defined for the complete data. In this paper, 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.

3 Preliminaries and Architecture of QPIAD

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.*

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.

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: $\sigma_{A_m=v_m}$ is defined to be the probability $P(t.A_m=v_m)$.*

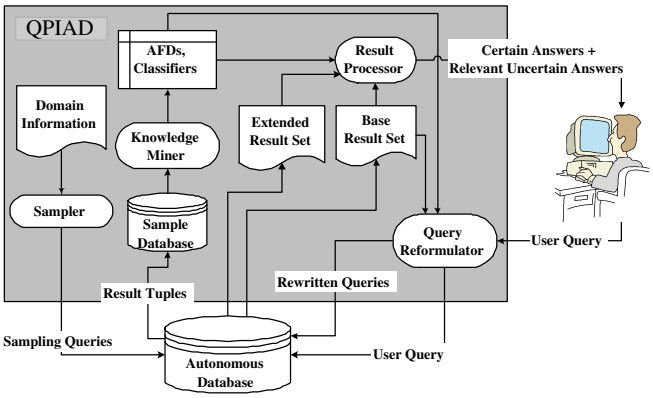


Figure 1. QPIAD System Architecture.

Conceptually, the operation of QPIAD can be understood in terms of (i) automatically assessing the probability distributions and (ii) actively using the learned distribution information to retrieve uncertain tuples that have a high degree of relevance to the query. Realizing this seemingly straightforward idea presents several challenges because of the autonomous nature of the databases. Figure 1 shows the *QPIAD* 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 attribute correlations learned from a database sample. The attribute correlations used to generate the rewritten queries are mined in terms of Approximate Functional Dependencies(AFDs). 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. The ranking of rewritten queries is based on the value distributions for the missing attribute. We reduce the problem of acquiring such value distributions to learning classifiers. We develop an AFD-enhanced Naïve Bayesian classifier learning method(where AFD plays a feature selection role for the classification task). As shown in Figure 1, *QPIAD* mines attribute correlations and learns value distributions on a small portion of data sampled from the autonomous database. The sampling module collects the sample data from the autonomous database using random probing queries, and the knowledge mining module learns AFDs and the AFD-enhanced classifiers from these samples.

4 Retrieving Relevant Incomplete Answers

In this section, we first briefly review the limitations of AUA and RUA approaches in the context of autonomous databases. Then, we will present a novel query rewriting technique that overcomes these limitations in order to retrieve relevant uncertain answers from incomplete autonomous databases. Finally, we describe how to extend this query rewriting framework in order to retrieve relevant answers from data sources not supporting the query attribute.

4.1 Retrieving Relevant Uncertain Answers from Autonomous Databases

Retrieving relevant uncertain answers is more challenging when it comes to autonomous web databases 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 Q': $\sigma_{body\ style=convt}$, both AUA and RUA retrieve all the tuples with missing $body\ style$ values, most of which may not be "*convt*". 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'. 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 $convt$. 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. There are two problems in adopting 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.

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: $\sigma_{model=A4}$ based on the *determining set* of the attribute $body\ style$. 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, 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 aims to restrict 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.

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 in the base result set based on the determining set of the constrained attribute. 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', 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 5.2, 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. The relevant uncertain answers retrieved by these queries need not be ranked again as they are implicitly ranked based on the rank of the rewritten query that retrieved them. Ranking rewritten queries according to their answers' expected degrees of relevance brings forth an appealing characteristic of the RRUA approach. 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. This adjustable mechanism allows the mediator to adapt to different users' preferences on precision and recall. For users who care more about the 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.

4.2 Algorithm for RRUA Approach

In this section, we describe the algorithmic details of the RRUA approach. Let $R(A_1, A_2, \cdots, A_n)$ be a database relation. Suppose $dtrSet(A_m)$ is the determining set of attribute A_m $(1 \leq m \leq n)$, according to the highest confidence AFD (to be discussed in Section 5.3). 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 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_is 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.[3]
   (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.

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

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).

4.3 Retrieving Relevant Answers from Data Sources Not Supporting the Query Attribute

We adapt the query rewriting techniques to a scenario where a mediator handles multiple sources in order to retrieve relevant answers from a data source not supporting the query attribute. The global schema supported by the mediator may often contain 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 *Yahoo! Autos* and *Cars.com* as shown in Figure 2. 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': $\sigma_{body\ style=convt}$ on the global schema, 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	$GS(Make, Model, Year, Price, Mileage, Location, Bodystyle)$
Cars.com	$LS(Make, Model, Year, Price, Mileage, Location, Bodystyle)$
Yahoo! Autos	$LS(Make, Model, Year, Price, Mileage, Location)$

Figure 2. 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 $body\ style$. For example, consider the *Cars.com* database for which we have mined an AFD $model \rightsquigarrow body\ style$. In order to retrieve relevant answers from the *Yahoo! Autos* database, the mediator issues rewritten queries to *Yahoo! Autos* based on this AFD and tuples retrieved from the base set results of *Cars.com*.

The algorithm used to retrieve relevant tuples from a source S_k not supporting the query attribute is similar to RRUA Algorithm described in Section 4.2 except that the base result set is retrieved from the *correlated source* S_c in Step 1.

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.*

From all sources correlated with a given source S_k, we use the source for which the AFD for A_m has the highest confidence. Then using AFDs and value distribution learned from S_c, ranked rewritten queries are generated in Step 2. These rewritten queries are then issued in the order of their ranks in order to retrieve relevant answers for the user query from source S_k.

5 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[3].

5.1 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. Recall that an AFD ϕ is a functional dependency that holds on all but a small fraction of tuples. According to [13], we define the *confidence* of an AFD ϕ on a relation R as: $conf(\phi) = 1 - g_3(\phi)$, where g_3 is the ratio of the minimum number of tuples that need to be removed from R to make ϕ a functional dependency on R. Similarly, we define *approximate key(AKey)* which is a key of all but a small fraction of tuples in R. We use TANE[9] algorithm to discover AFDs and AKeys whose confidence is above a threshold α, which is set to 0.3 in QPIAD to ensure that we do not miss any significant AFDs.

Pruning Noisy AFDs: In most cases, AFDs with high confidence are desirable for learning probability distributions for missing values. However, not all high confidence AFDs are useful for feature selection, such as high confidence AKeys whose values are distinct. For example, consider a relation *car(VIN, model, make)*. 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 AFDs with a superset of AKey attributes in the determining set are not useful features for classification and should 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 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. An AFD will be pruned if its confidence difference to the corresponding AKey is below a threshold δ(currently set at 0.3 based on experimentation).

5.2 Learning Value Distributions using Classifiers

Given a tuple with a null value, we now need to estimate the probability of each possible value of this null. We reduce this problem to a classification problem using mined AFDs as selected features. A classifier is a function f that maps a given attribute vector $\vec{x}$ to a confidence that the vector belongs to a 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 Classifier(NBC) for each missing attribute A_m. Let a value v_i in the domain of A_m represent a possible class for A_m. Let $\vec{x}$ denote the values of $dtrSet(A_m)$ in a tuple with null on A_m. We use Bayes theorem to estimate the probabilities: $P(A_m{=}v_i|\vec{x}) = \frac{P(\vec{x}|A_m=v_i)P(A_m=v_i)}{P(\vec{x})}$ for all values v_i in the domain. To improve computation efficiency, NBC assumes that for a given class, the features $X_1, \cdots, X_n$ are conditionally independent, and therefore we have: $P(\vec{x}|A_m{=}v_i) = \prod_i P(x_i|A_m{=}v_i)$. Despite this strong simplification, NBC has been shown to be surprisingly effective[6]. In the actual implementation, we adopt the standard practice of using NBC with a variant of Laplacian smoothing called m-estimates[16] to improve the accuracy.

5.3 Combining AFDs and Classifiers

So far 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. One method is to use the determining set of the AFD with the *highest confidence* which we call the *Best-AFD* method. However, our experiments showed that this approach can degrade the classification accuracy if its confidence is too low. Therefore we ignore AFDs with confidence below a threshold (which is currently set to be 0.5 based on experimentation), and instead use all other attributes to learn the probability distribution using NBC. We call this approach *Hybrid One-AFD*. We could also use an *Ensemble of classifiers* corresponding to the set of AFDs for each attribute, and then combine the probability distribution of each classifier by a weighted average. At the other extreme, we could ignore feature selection based on AFD completely but use all the attributes to learn probability distribution using NBC. Our experiments described in Section 6 show that Hybrid One-AFD approach has the best classification accuracy among these choices.

6 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.

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. *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 $model{\rightsquigarrow}body\ style$.

Experimental Settings
We evaluated the quality of classifiers and the performance of query rewriting and ranking techniques on two data sets. One is the *used car database Cars(make,model,year,price,milage,location,color,body style)* extracted from AutoTrader(www.autotrader.com) which has around 100,000 tuples. This database in inherently incomplete as described in Table 1. The second is the *Census database Census(age, workshop, weight, education, marital-status, occupation, relationship, race, sex, capital-gain, capital-loss, hours-per-week, native-country)* from UCI[4] data repository with around 45,000 tuples.

To perform an *oracular study* based on a *ground truth*, for both databases we consider only complete tuples and then artificially make them incomplete by randomly selecting 10% of tuples and making one random attribute in each tuple to be null, which forms the test database.[5]

We partitioned each database into two parts: a training set to learn the classifiers and a test set. To simulate the relatively small percentage of the training data available to the mediators, we experimented with different sizes of training set, varying from 3% to 15% of the entire databases.

Accuracy of Classifiers
Since we use classifiers as a basis for query rewriting and for ranking, we perform a baseline study on their accuracy. For each tuple in the test set, we compute the probability distribution of possible values of a null, choose the one with the maximum probability and compare it against the actual value. The classification accuracy is defined as the proportion of the tuples in the test set that have their null values predicted correctly.

Table 3 shows the average prediction accuracy of various AFD-enhanced classifiers introduced in Section 5.3. In this experiment, we use a training set whose size is 10% of the database. The classification accuracy is measured over 5 runs using different training set and test set for each run. Considering the large domain sizes of attributes in Cars

[4] Available at http://www.ics.uci.edu/mlearn/MLRepository.html

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

database (varying from $55(year)$ to $1151(model)$), the classification accuracy obtained is quite reasonable, since a random guess would give much lower prediction accuracy. We can also see in Table 3 that the Hybrid One-AFD approach performs the best and therefore is used in our query rewriting implementation.

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 3. Null value prediction accuracy across different AFD-enhanced classifiers

We also compared the accuracy of our AFD-enhanced NBC classifier with two other approaches - one based on association rules[25] and the other that learns bayesian networks from the data. We found that the accuracy of our approach is competitive with the bayesian networks[8] while being significantly better than the association rules. Space limitations preclude extensive discussion of the experiments, the interested readers are referred to [12].

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 5. In contrast, RRUA uses query rewriting techniques to retrieve only relevant uncertain answers in a ranked order. We use precision-recall to measure the effectiveness of the above approaches at the time when the mediator sees the K^{th} (K=$1, 2, 3, \cdots$) answer tuple. In the rest of the evaluation, we focus on comparing the

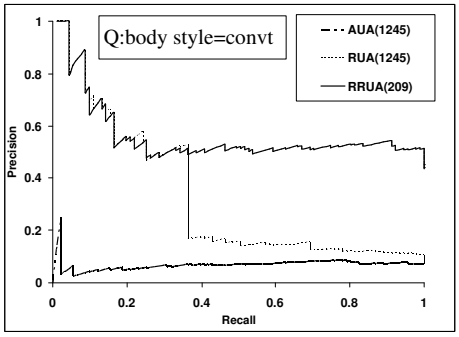


Figure 3. Precision-Recall curves(Cars)

effectiveness of retrieving relevant uncertain answers. That is, the precision and recall are both calculated with respect to uncertain answers instead of the entire answer set.

Figures 3 and 4 show the precision and recall curves of a query on each of the test databases.[6] It shows that both

[6]Note that to evaluate the full spectrum of precision-recall curve, we donot set threshold on confidence for both RUA tuples and RRUA rewritten queries

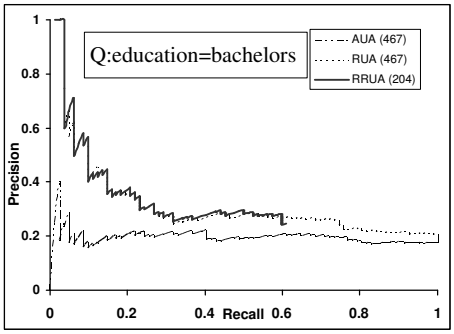


Figure 4. Precision-Recall curves(Census).

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.

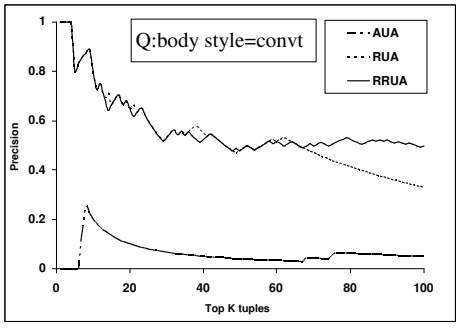


Figure 5. Precision for Top K tuples(Cars).

Furthermore, the numbers in the parenthesis in the legend in Figures 3 and 4 are the number of uncertain answers retrieved by each method. As we can see, RRUA avoids retrieving too many irrelevant tuples and therefore very efficient. It can further cut down the cost by only sending the top ranked rewritten queries to the database.

To reflect the "density" of the relevant answers along the time line, we also plot the precision of each method at the time when top $K(K$=$1, 2, \cdots, 100)$ answers are retrieved as shown in Figures 5 and 6 for the same queries. Again RUA and RRUA are much better in retrieving relevant uncertain answers in top K results which is critical in web scenarios.

Effect of Confidence Threshold on Precision

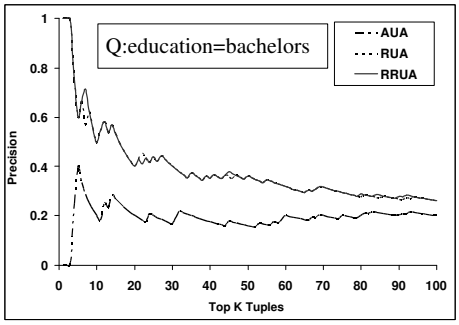


Figure 6. Precision for top K tuples(Census).

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 7 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.

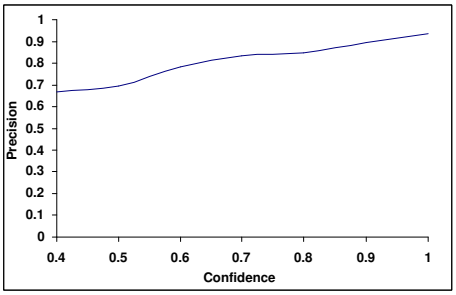


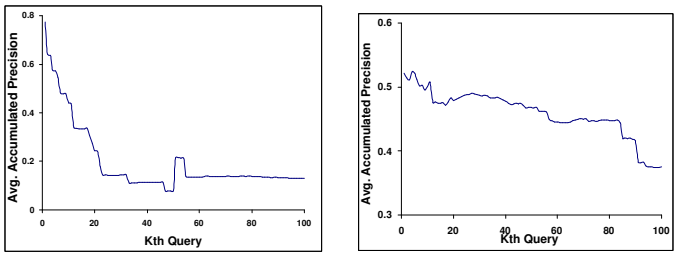
Figure 7. Average Precision for various confidence thresholds(Cars).

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 order of relevance, measured by the precision of the returned result set. For each database, we choose 30 test queries, each of which has a single selection predicate "attribute=value". 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 in order. After the extended result set $\widehat{RS}(Q_m)$ of each query $Q_m (1 \leq m \leq n)$ is retrieved, we calculate the *accumulated precision* as follows:

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

Figure 8 shows the average accumulated precision of the



(a) Cars database (b) Census database

Figure 8. The accumulated precision curves.

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.

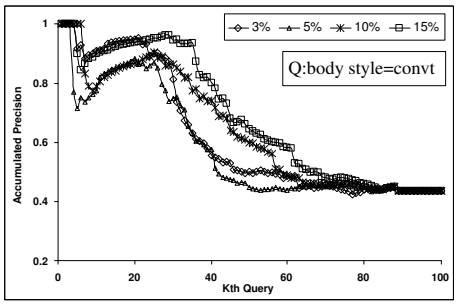


Figure 9. Accumulated precision curve with different sample sizes on Cars database.

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 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 9 shows the accumulated precision of a selection query on the Car database, 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%. We obtain a similar result for the Census database which is described in [12] due to space constraints.

Effectiveness of using Correlation Between Data Sources

We consider a mediator performing data integration over three data sources *Cars.com* (www.cars.com), *Yahoo! Autos* (autos.yahoo.com) and *CarsDirect* (www.carsdirect.com). The global schema supported by the mediator and the individual local schemas are shown in Figure 2. The schema

of *CarsDirect* is the same as that of Yahoo! Autos which do not support $body\ style$ attribute while *Cars.com* does support queries on the $body\ style$ attribute using Advanced Search. We use the AFDs and NBC classifiers learned from *Cars.com* to retrieve cars from *Yahoo! Autos* and *CarsDirect* as possible relevant uncertain answers for queries on $body\ style$ according to Section 4.3. To evaluate the precision, we check the actual $body\ style$ of the retrieved car tuples to determine whether the tuple was indeed relevant to the original query. The average precision for the top K tuples retrieved from *Yahoo! Autos* and *CarsDirect* over the 5 test queries is quite high as shown in Figure 10. This shows that using the AFDs and value distributions learned from correlated sources, QPIAD can retrieve relevant answers from data sources not supporting query attribute.

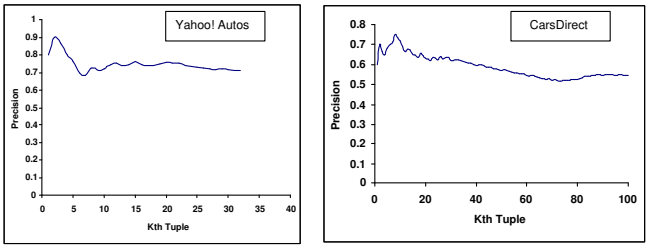


Figure 10. Precision curves for top K tuples retrieved using correlated source *Cars.com*.

7 Conclusion and Future Work

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, limited query patterns supported by autonomous databases, and 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 comprehensive experiments demonstrated the effectiveness of our query processing and knowledge mining techniques.

In this paper, we have focused mainly on selection queries since these are the most typical queries faced in data aggregation over web databases. We did however investigate join queries in the context of QPIAD; a preliminary account can be found in [12]. With join queries, *QPIAD* starts to inherit some of the open challenges of probabilistic databases, including doing joins over relations with uncertain tuples. One interesting issue is that even if we start with tuples having at most one null value — which can be modelled as the so called x-tuples, joins will spread the uncertainty necessitating the full blown c-tables [10, 22] to represent the uncertain tuples. Presenting answers involving such general uncertainty to lay users is an open problem.

References

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

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

[3] A. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 1997.

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

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

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

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

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

[9] 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.

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

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

[12] H. Khatri. *Query Processing over Incomplete Autonomous Web Databases,MS Thesis,Dept. of CSE, Arizona State University*, rakaposhi.eas.asu.edu/hemal-thesis.pdf. 2006.

[13] J. Kivinen and H. Mannila. Approximate dependency inference from relations. In *ICDT Conference*, 1992.

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

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

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

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

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

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

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

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

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

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

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

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