Query Processing over Incomplete Autonomous Databases

Garrett Wolf (Arizona State University)
Hemal Khatri (MSN Live Search)
Bhaumik Chokshi (Arizona State University)
Jianchun Fan (Amazon)
Yi Chen (Arizona State University)
Subbarao Kambhampati (Arizona State University)
Introduction

- More and more data is becoming accessible via web servers which are supported by backend databases
  - *E.g. Cars.com, Realtor.com, Google Base, Etc.*
Incompleteness in Web Databases

- **Inaccurate Extraction / Recognition**
- **Incomplete Entry**
- **Heterogeneous Schemas**
- **User-defined Schemas**

<table>
<thead>
<tr>
<th>Website</th>
<th># of Attributes</th>
<th>Total Tuples</th>
<th>Incomplete %</th>
<th>Body Style %</th>
<th>Engine %</th>
</tr>
</thead>
<tbody>
<tr>
<td>AutoTrader.com</td>
<td>13</td>
<td>25127</td>
<td>33.67%</td>
<td>3.6%</td>
<td>8.1%</td>
</tr>
<tr>
<td>CarsDirect.com</td>
<td>14</td>
<td>32564</td>
<td>98.74%</td>
<td>55.7%</td>
<td>55.8%</td>
</tr>
<tr>
<td>Google Base</td>
<td>203+</td>
<td>580993</td>
<td>100%</td>
<td>83.36%</td>
<td>91.98%</td>
</tr>
</tbody>
</table>
Problem

- Current autonomous database systems only return *certain answers*, namely those which exactly satisfy all the user query constraints.

How to retrieve relevant uncertain results in a ranked fashion?

Many entities corresponding to tuples with missing values might be relevant to the user query.
Possible Naïve Approaches

Query Q: (Body Style = Convt)

1. **CERTAINONLY**: Return certain answers only as in traditional databases, namely those having Body Style = Convt. 
   
   ![Low Recall]

2. **ALLRETURNED**: Null matches any concrete value, hence return all answers having Body Style = Convt along with answers having body style as null.
   
   ![Low Precision, Infeasible]

3. **ALLRANKED**: Return all answers having Body Style = Convt. Additionally, rank all answers having body style as null by predicting the missing values and return them to the user.
   
   ![Costly, Infeasible]
Outline

- Core Techniques
- Peripheral Techniques
- Implementation & Evaluation
- Conclusion & Future Work
The QPIAD Solution

Given a query \( Q: (\text{Body}=\text{Conv}) \) retrieve all relevant tuples

<table>
<thead>
<tr>
<th>Id</th>
<th>Make</th>
<th>Model</th>
<th>Year</th>
<th>Body</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Audi</td>
<td>A4</td>
<td>2001</td>
<td>Conv</td>
</tr>
<tr>
<td>2</td>
<td>BMW</td>
<td>Z4</td>
<td>2002</td>
<td>Conv</td>
</tr>
<tr>
<td>3</td>
<td>Porsche</td>
<td>Boxster</td>
<td>2005</td>
<td>Conv</td>
</tr>
<tr>
<td>4</td>
<td>BMW</td>
<td>Z4</td>
<td>2003</td>
<td>NULL</td>
</tr>
<tr>
<td>5</td>
<td>Honda</td>
<td>Civic</td>
<td>2004</td>
<td>NULL</td>
</tr>
<tr>
<td>6</td>
<td>Toyota</td>
<td>Camry</td>
<td>2002</td>
<td>Sedan</td>
</tr>
<tr>
<td>7</td>
<td>Audi</td>
<td>A4</td>
<td>2006</td>
<td>NULL</td>
</tr>
</tbody>
</table>

Base Result Set

<table>
<thead>
<tr>
<th>Id</th>
<th>Make</th>
<th>Model</th>
<th>Year</th>
<th>Body</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Audi</td>
<td>A4</td>
<td>2001</td>
<td>Conv</td>
</tr>
<tr>
<td>2</td>
<td>BMW</td>
<td>Z4</td>
<td>2002</td>
<td>Conv</td>
</tr>
<tr>
<td>3</td>
<td>Porsche</td>
<td>Boxster</td>
<td>2005</td>
<td>Conv</td>
</tr>
</tbody>
</table>

Ranked Relevant Uncertain Answers

<table>
<thead>
<tr>
<th>Id</th>
<th>Make</th>
<th>Model</th>
<th>Year</th>
<th>Body</th>
<th>Confidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>BMW</td>
<td>Z4</td>
<td>2003</td>
<td>NULL</td>
<td>0.7</td>
</tr>
<tr>
<td>7</td>
<td>Audi</td>
<td>A4</td>
<td>2006</td>
<td>NULL</td>
<td>0.3</td>
</tr>
</tbody>
</table>

Select Top K Rewritten Queries

- \( Q_1': \) Model=A4
- \( Q_2': \) Model=Z4
- \( Q_3': \) Model=Boxster
Learning Statistics to Support Ranking & Rewriting

- **What is hard?**
  - Learning correlations useful for rewriting
  - Efficiently assessing the probability distribution
  - Cannot modify the underlying autonomous sources

- **Attribute Correlations** - Approximate Functional Dependencies (AFDs) & Approximate Keys (AKeys)

- **Value Distributions** - Naïve Bayes Classifiers (NBC)

\[
\text{EstPrec}(Q|R) = (A_m = v_m | \text{dtrSet}(A_m))
\]

\[
P(\text{Model}=\text{Accord} | \text{Make}=\text{Honda}, \text{Body}=\text{Coupe})
\]
Rewriting to Retrieve Relevant Uncertain Results

- **What is hard?**
  - Retrieving relevant uncertain tuples with missing values
  - Rewriting queries given the limited query access patterns

- **AFD: Model~> Body**

- Given an AFD and Base Set, it is likely that tuples with
  1) Model of A4, Z4 or Boxster
  2) Body of NULL
  are actually convertible.

- Generate rewritten queries for each distinct Model:
  - Q1': Model=A4
  - Q2': Model=Z4
  - Q3': Model=Boxster

---

<table>
<thead>
<tr>
<th>Make</th>
<th>Model</th>
<th>Year</th>
<th>Body</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audi</td>
<td>A4</td>
<td>2001</td>
<td>Convt</td>
</tr>
<tr>
<td>BMW</td>
<td>Z4</td>
<td>2002</td>
<td>Convt</td>
</tr>
<tr>
<td>Porsche</td>
<td>Boxster</td>
<td>2005</td>
<td>Convt</td>
</tr>
<tr>
<td>BMW</td>
<td>Z4</td>
<td>2003</td>
<td>Convt</td>
</tr>
</tbody>
</table>
Selecting/Ordering Top-K Rewritten Queries

- **What is hard?**
  - Retrieving precise, non-empty result sets
  - Working under source-imposed resource limitations

- Select top-k queries based on **F-Measure**
  
  \[F_\alpha = \frac{(1 + \alpha) 
  \cdot (P \cdot R)}{\alpha \cdot P + R}\]

  - \(P\) – Estimated Precision
  - \(R\) – Estimated Recall

- Reorder queries based on **Estimated precision**
  
  \[
  F_0 = \frac{P \cdot R}{R} = P
  \]

- Retrieves tuples in order of their **final ranks**
  - **No need to re-rank tuples after retrieving them!**
Explaining Results to the Users

- **What is hard?**
  - Gaining the user’s trust
  - Generating meaningful explanations

Explanations based on AFDs.

<table>
<thead>
<tr>
<th>Make</th>
<th>Model</th>
<th>Year</th>
<th>Price</th>
<th>Color</th>
<th>Body</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Honda</td>
<td>Accord</td>
<td>2001</td>
<td>$10,500</td>
<td>Silver</td>
<td>Sedan</td>
<td></td>
</tr>
<tr>
<td>Honda</td>
<td>Accord</td>
<td>2002</td>
<td>$11,200</td>
<td>White</td>
<td>Coupe</td>
<td></td>
</tr>
<tr>
<td>Honda</td>
<td>Accord</td>
<td>1999</td>
<td>$9,000</td>
<td>Green</td>
<td>Sedan</td>
<td></td>
</tr>
<tr>
<td>?</td>
<td>Accord</td>
<td>2001</td>
<td>$11,700</td>
<td>Red</td>
<td>Sedan</td>
<td>This car is 100% likely to have Model=Accord given that its Make=Honda</td>
</tr>
<tr>
<td>Honda</td>
<td>?</td>
<td>2000</td>
<td>$10,100</td>
<td>Blue</td>
<td>Sedan</td>
<td>This car is 83% likely to have Model=Accord given that its Make=Honda and Body=Sedan</td>
</tr>
<tr>
<td>Honda</td>
<td>Accord</td>
<td>1999</td>
<td>?</td>
<td>Black</td>
<td>Sedan</td>
<td>This car is 71% likely to have Price&lt;$12,000 given that its Model=Accord and Year=1999</td>
</tr>
<tr>
<td>Honda</td>
<td>?</td>
<td>2002</td>
<td>$10,750</td>
<td>Silver</td>
<td>Coupe</td>
<td>This car is 42% likely to have Model=Accord given that its Make=Honda and Body=Coupe</td>
</tr>
</tbody>
</table>

Provide to the user:
- Certain Answers
- Relevant Uncertain Answers
- Explanations
Outline

- Core Techniques
- Peripheral Techniques
- Implementation & Evaluation
- Conclusion & Future Work
Leveraging Correlation between Data Sources

Mediator: GS(Body, Make, Model, Year, Price, Mileage)

Q:(Body=Coupe)

AFDs learned from Cars.com

Base Set +
AFD (Model → Body)

Q:(Model=Civic)
Q:(Model=Corvette) ...

Two main uses:
Source doesn’t support all the attributes in GS
Sample/statistics aren’t available
Handling Aggregate and Join Queries

- **Aggregate Queries**

<table>
<thead>
<tr>
<th>Id</th>
<th>Make</th>
<th>Model</th>
<th>Body</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td>Audi</td>
<td>A4</td>
<td>Convt</td>
</tr>
<tr>
<td>t2</td>
<td>BMW</td>
<td>Z4</td>
<td>NULL</td>
</tr>
<tr>
<td>t3</td>
<td>Porsche</td>
<td>Boxster</td>
<td>Convt</td>
</tr>
<tr>
<td>t4</td>
<td>BMW</td>
<td>325i</td>
<td>NULL</td>
</tr>
<tr>
<td>t5</td>
<td>Honda</td>
<td>Civic</td>
<td>Coupe</td>
</tr>
</tbody>
</table>

- **Join Queries**

<table>
<thead>
<tr>
<th>Make</th>
<th>Model</th>
<th>Year</th>
<th>Mileage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Honda</td>
<td>Accord</td>
<td>NULL</td>
<td>60,400</td>
</tr>
<tr>
<td>Toyota</td>
<td>Camry</td>
<td>2004</td>
<td>21,150</td>
</tr>
<tr>
<td>NULL</td>
<td>Civic</td>
<td>2002</td>
<td>53,275</td>
</tr>
<tr>
<td>Audi</td>
<td>A4</td>
<td>2003</td>
<td>41,650</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Make</th>
<th>Year</th>
<th>Part</th>
</tr>
</thead>
<tbody>
<tr>
<td>Honda</td>
<td>2001</td>
<td>Brakes</td>
</tr>
<tr>
<td>NULL</td>
<td>2002</td>
<td>Windows</td>
</tr>
<tr>
<td>Toyota</td>
<td>2005</td>
<td>Air Bags</td>
</tr>
</tbody>
</table>

Query Processing over Incomplete Autonomous Databases
Outline

✓ Core Techniques

✓ Peripheral Techniques

☐ Implementation & Evaluation

☐ Conclusion & Future Work
QPIAD Web Interface

http://rakaposhi.eas.asu.edu/qpiad

QUERY BUILDER

- MYEAR:
- MAKE:
- MODEL: 350z
- PRICE:
- MILEAGE:
- BODY: Conv
- CERTIFIED:

SAMPLE QUERIES

- Model=Accord
- Model=350z and Body=Conv
- Model=645 and Year=2004

VLDB '07
Query Processing over Incomplete Autonomous Databases
Empirical Evaluation

- **Datasets:**
  - **Cars**
    - Cars.com
    - 7 attributes, 55,000 tuples
  - **Complaints**
    - NHSTA Office of Defect Investigation
    - 11 attributes, 200,000 tuples
  - **Census**
    - US Census dataset, UCI repository
    - 12 attributes, 45,000 tuples

- **Sample Size:**
  - 3-15% of full database

- **Incompleteness:**
  - 10% of tuples contain missing values
  - Artificially introduced null values in order to compare with the ground truth

- **Evaluation:**
  - Ranking and Rewriting Methods (e.g. quality, efficiency, etc.)
  - General Queries (e.g. correlated sources, aggregates, joins, etc.)
  - Learning Methods (e.g. accuracy, sample size, etc.)
Experimental Results – Ranking & Rewriting

- **QPIAD vs. ALLRETURNED** - *Quality*

---

Figure 1: Comparison of Precision and Recall for QPIAD and ALLRETURNED

- **ALLRETURNED** – all certain answers + all answers with nulls on constrained attributes
Experimental Results – Ranking & Rewriting

- QPIAD vs. ALLRANKED - **Efficiency**

**ALLRANKED** – all certain answers + all answers with predicted missing value probability above a threshold
Experimental Results – General Queries

- **Aggregates**

  ![Graph showing the fraction of queries vs. accuracy](image)

  - Query: `Q: Count(*)`
  - No Prediction
  - Prediction

- **Joins**

  ![Graph showing precision vs. recall](image)

  - Query: `Q: (Gen. Comp.=Engine and Engine Cooling) JOIN ON (Model=Grand Cherokee) (K = 10 Rewritten Queries)`
  - Alpha=0.0
  - Alpha=0.5
  - Alpha=2

**VLDB ’07**

Query Processing over Incomplete Autonomous Databases
Experimental Results – Learning Methods

- Accuracy of Classifiers
Experimental Summary

- **Rewriting / Ranking**
  - **Quality** – QPIAD achieves higher precision than ALLRETURNED by only retrieving the relevant tuples.
  - **Efficiency** – QPIAD requires fewer tuples to be retrieved to obtain the same level of recall as ALLRANKED.

- **Learning Methods**
  - AFDs for feature selection improved accuracy.

- **General Queries**
  - Aggregate queries achieve higher accuracy when missing value prediction is used.
  - QPIAD achieves higher levels of recall for join queries while trading off only a small bit of precision.

- **Additional Experiments**
  - Robustness of learning methods w.r.t. sample size.
  - Effect of alpha value on F-measure.
  - Effectiveness of using correlation between sources.
Outline

☑ Core Techniques
☑ Peripheral Techniques
☑ Implementation & Evaluation

☐ Conclusion & Future Work
Related Work

- Querying Incomplete Databases
  - Possible World Approaches – tracks the completions of incomplete tuples (Codd Tables, V-Tables, Conditional Tables)
  - Probabilistic Approaches – quantify distribution over completions to distinguish between likelihood of various possible answers

- Probabilistic Databases
  - Tuples are associated with an attribute describing the probability of its existence
  - However, in our work, the mediator does not have the capability to modify the underlying autonomous databases

- Query Reformulation / Relaxation
  - Aims to return similar or approximate answers to the user after returning or in the absence of exact answers
  - Our focus is on retrieving tuples with missing values on constrained attributes

- Learning Missing Values
  - Common imputation approaches replace missing values by substituting the mean, most common value, default value, or using kNN, association rules, etc.
  - Our work requires schema level dependencies between attributes as well as distribution information over missing values
Contributions

- **Efficiently retrieve relevant uncertain answers** from autonomous sources given only **limited query access patterns**
  - Query Rewriting

- Retrieves **answers with missing values** on constrained attributes **without modifying** the underlying databases
  - AFD-Enhanced Classifiers

- Rewriting & ranking considers the **natural tension between precision and recall**
  - F-Measure based ranking

- AFDs play a major role in:
  - Query Rewriting
  - Feature Selection
  - Explanations
Current Directions – QUIC (CIDR `07 Demo)

http://rakaposhi.eas.asu.edu/quic

Incomplete Data
Databases are often populated by:
- Lay users entering data
- Automated extraction

Imprecise Queries
User’s needs are not clearly defined:
- Queries may be too general
- Queries may be too specific

Density Function
\[
\mathcal{P}(t|\hat{t}, D) = \sum_{t \in C(\hat{t})} \mathcal{R}(t|Q, U) \mathcal{P}(t|\hat{t}, D)
\]

Relevance Function
\[
\mathcal{R}(t|Q, U)
\]

General Solution: “Expected Relevance Ranking”

Challenge: Automated & Non-intrusive assessment of Relevance and Density functions

Estimating Relevance (R):
Learn relevance for user population as a whole in terms of value similarity
- Sum of weighted similarity for each constrained attribute
  - Content Based Similarity
  - Co-click Based Similarity
  - Co-occurrence Based Similarity

<table>
<thead>
<tr>
<th>(\sigma_{Model} \approx Civic)</th>
<th>Civic</th>
<th>Accord</th>
<th>Prelude</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relevance</td>
<td>1.0</td>
<td>0.78</td>
<td>0.59</td>
</tr>
<tr>
<td>Density</td>
<td>0.62</td>
<td>0.21</td>
<td>0.17</td>
</tr>
</tbody>
</table>

VLDB ’07
Query Processing over Incomplete Autonomous Databases
Problem

- Current mediator systems only return *certain answers*, namely those which exactly satisfy all the user query constraints.

How to support query processing over incomplete autonomous databases in order to retrieve relevant uncertain results?

<table>
<thead>
<tr>
<th>Make</th>
<th>Model</th>
<th>Year</th>
<th>Price</th>
<th>Color</th>
<th>Body</th>
</tr>
</thead>
<tbody>
<tr>
<td>Honda</td>
<td>Accord</td>
<td>1999</td>
<td></td>
<td>Green</td>
<td>Sedan</td>
</tr>
</tbody>
</table>

Many entities corresponding to tuples with missing values might be relevant to the user query.
Handling Aggregate and Join Queries

- **Aggregate Queries**

  \[ \text{Q: (Count(*) Where Body=Convt)} \]

<table>
<thead>
<tr>
<th>Id</th>
<th>Make</th>
<th>Model</th>
<th>Body</th>
<th>Prob. Distr.</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td>Audi</td>
<td>A4</td>
<td>Convt</td>
<td></td>
</tr>
<tr>
<td>t2</td>
<td>BMW</td>
<td>NULL</td>
<td>Convt</td>
<td>(P(\text{Convt}) = 0.9, P(\text{Coupe}) = 0.1)</td>
</tr>
<tr>
<td>t3</td>
<td>Porsche</td>
<td>Boxster</td>
<td>Convt</td>
<td></td>
</tr>
<tr>
<td>t4</td>
<td>BMW</td>
<td>325i</td>
<td>NULL</td>
<td>(P(\text{Convt}) = 0.4, P(\text{Coupe}) = 0.6)</td>
</tr>
<tr>
<td>t5</td>
<td>Honda</td>
<td>Civic</td>
<td>Coupe</td>
<td></td>
</tr>
</tbody>
</table>

- **Join Queries**
  - Refer to the paper for details

  Only include tuples whose most likely missing value matches the query constraint.

  Count(*) = 3.3

  Include a portion of each tuple relative to the probability its missing value matches the query constraint.

  Count(*) = 3
Learning Statistics to Support Ranking & Rewriting

- **What is hard?**
  - Learning correlations useful for rewriting
  - Efficiently assessing the probability distribution
  - Cannot modify the underlying autonomous sources

- **Attribute Correlations** - Approximate Functional Dependencies (AFDs) & Approximate Keys (AKeys)

- **Value Distributions** - Naïve Bayes Classifiers (NBC)

- **Selectivity Estimates** – Sample Size, Ratio, Percent Incomplete

\[
\text{EstSel}(Q|R) = \text{SmplSel}(Q) \times \text{SmplRatio}(R) \times \text{PerInc}(R)
\]

\[
\text{EstPrec}(Q|R) = \frac{A_m = v_m | \text{dtrSet}(A_m))}{\{A_1, \ldots, A_k\} \sim A_m \quad 0 < \text{conf} \leq 1}
\]
Incompleteness in Web Databases

**Inaccurate Extraction/Recognition**
- Imperfections in segmenting of web pages or scanning and converting handwritten forms

**Incomplete Entry**
- User leaves the *Make* attribute blank assuming it is obvious as the *Model* is Accord

**Heterogeneous Schemas**
- Global schema provided by the mediator often contains attributes not present in all the local schemas

**User-defined Schemas**
- Redundant attributes (e.g. Make vs. Manufacturer) and the proliferation of null values (e.g. tuples with Make are unlikely to provide Manufacturer)

<table>
<thead>
<tr>
<th>Website</th>
<th># of Attributes</th>
<th>Total Tuples</th>
<th>Incomplete %</th>
<th>Body Style %</th>
<th>Engine %</th>
</tr>
</thead>
<tbody>
<tr>
<td>AutoTrader.com</td>
<td>13</td>
<td>25127</td>
<td>33.67%</td>
<td>3.6%</td>
<td>8.1%</td>
</tr>
<tr>
<td>CarsDirect.com</td>
<td>14</td>
<td>32564</td>
<td>98.74%</td>
<td>55.7%</td>
<td>55.8%</td>
</tr>
<tr>
<td>Google Base</td>
<td>203+</td>
<td>580993</td>
<td>100%</td>
<td>83.36%</td>
<td>91.98%</td>
</tr>
</tbody>
</table>
Introduction

- More and more data is becoming accessible via web servers which are supported by backend databases
  - *E.g.* Cars.com, Realtor.com, Google Base, Etc.

- As a result, mediator systems are being developed to provide a single point of access to multiple databases
Problem

- Current mediator systems only return *certain answers*, namely those which exactly satisfy all the user query constraints.

<table>
<thead>
<tr>
<th>Make</th>
<th>Model</th>
<th>Year</th>
<th>Price</th>
<th>Color</th>
<th>Body</th>
</tr>
</thead>
<tbody>
<tr>
<td>Honda</td>
<td>Accord</td>
<td>2001</td>
<td>$10,500</td>
<td>Silver</td>
<td>?</td>
</tr>
<tr>
<td>?</td>
<td>Accord</td>
<td>2002</td>
<td>$11,200</td>
<td>White</td>
<td>Sedan</td>
</tr>
<tr>
<td>Honda</td>
<td>Accord</td>
<td>1999</td>
<td>?</td>
<td>Green</td>
<td>Sedan</td>
</tr>
</tbody>
</table>

High Precision
Low Recall

Want a ‘Honda Accord’ with a ‘sedan’ body style for under ‘$12,000’
Problem

- Current mediator systems only return *certain answers*, namely those which exactly satisfy all the user query constraints.

How to support query processing over incomplete autonomous databases in order to retrieve relevant uncertain results?

Many entities corresponding to tuples with missing values might be relevant to the user query.
Selecting Top-K Rewritten Queries using F-Measure

- Sources may impose resource limitations on the # of queries we can issue

- Therefore, we should select only the top-K queries while ensuring the proper balance between precision and recall

- SOLUTION: Use F-Measure based selection with configurable alpha parameter
  - $\alpha=1$ $P = R$
  - $\alpha<1$ $P > R$
  - $\alpha>1$ $P < R$

- NOTE: F-Measure is used for selecting the top-K queries but does not determine the order in which they are sent

\[ F_\alpha = \frac{(1 + \alpha) \cdot (P \cdot R)}{(\alpha \cdot P + R)} \]
Once we’ve selected the top-K rewritten queries, we must reorder them in order of their **estimated precision**

- Use the precision estimates we already have

\[
F_0 = \frac{P \cdot R}{R} = P
\]

Issuing the queries in order of estimated precision allows us to retrieve tuples in order of their final ranks

- **No need to re-rank tuples after retrieving them, simply show them to the user!**

**NOTE:** All tuples returned for a single query are ranked equally
Problem

- Current autonomous database systems only return *certain answers*, namely those which exactly satisfy all the user query constraints.

Want a ‘Honda Accord’ with a ‘sedan’ body style for under ‘$12,000’

<table>
<thead>
<tr>
<th>Make</th>
<th>Model</th>
<th>Year</th>
<th>Price</th>
<th>Color</th>
<th>Body</th>
</tr>
</thead>
<tbody>
<tr>
<td>Honda</td>
<td>Accord</td>
<td>2001</td>
<td>$10,500</td>
<td>Silver</td>
<td>?</td>
</tr>
<tr>
<td>?</td>
<td>Accord</td>
<td>2002</td>
<td>$11,200</td>
<td>White</td>
<td>Sedan</td>
</tr>
<tr>
<td>Honda</td>
<td>Accord</td>
<td>1999</td>
<td>?</td>
<td>Green</td>
<td>Sedan</td>
</tr>
</tbody>
</table>
Problem

- Current autonomous database systems only return \textit{certain answers}, namely those which exactly satisfy all the user query constraints.

\begin{itemize}
  \item How to support query processing over incomplete autonomous databases in order to retrieve relevant uncertain results in a ranked fashion?
\end{itemize}

High Precision
Low Recall

Want a \textit{‘Honda Accord’} with a \textit{‘sedan’} body style for under \textit{‘$12,000’}?

Many entities corresponding to tuples with missing values might be relevant to the user query.

\begin{tabular}{|l|l|l|l|l|l|}
\hline
Make & Model & Year & Price & Color & Body \\
\hline
Honda & Accord & 1999 & ? & Green & Sedan \\
\hline
\end{tabular}
Query Rewriting in QPIAD

Given a query $Q$: \text{(Body Style=Convt)} retrieve all relevant tuples

<table>
<thead>
<tr>
<th>Id</th>
<th>Make</th>
<th>Model</th>
<th>Year</th>
<th>Body</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Audi</td>
<td>A4</td>
<td>2001</td>
<td>Convt</td>
</tr>
<tr>
<td>2</td>
<td>BMW</td>
<td>Z4</td>
<td>2002</td>
<td>Convt</td>
</tr>
<tr>
<td>3</td>
<td>Porsche</td>
<td>Boxster</td>
<td>2005</td>
<td>Convt</td>
</tr>
<tr>
<td>4</td>
<td>BMW</td>
<td>Z4</td>
<td>2003</td>
<td>Null</td>
</tr>
<tr>
<td>5</td>
<td>Honda</td>
<td>Civic</td>
<td>2004</td>
<td>Null</td>
</tr>
<tr>
<td>6</td>
<td>Toyota</td>
<td>Camry</td>
<td>2002</td>
<td>Sedan</td>
</tr>
<tr>
<td>7</td>
<td>Audi</td>
<td>A4</td>
<td>2006</td>
<td>Null</td>
</tr>
</tbody>
</table>

Base Result Set

<table>
<thead>
<tr>
<th>Id</th>
<th>Make</th>
<th>Model</th>
<th>Year</th>
<th>Body</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Audi</td>
<td>A4</td>
<td>2001</td>
<td>Convt</td>
</tr>
<tr>
<td>2</td>
<td>BMW</td>
<td>Z4</td>
<td>2002</td>
<td>Convt</td>
</tr>
<tr>
<td>3</td>
<td>Porsche</td>
<td>Boxster</td>
<td>2005</td>
<td>Convt</td>
</tr>
</tbody>
</table>

AFD: Model~> Body style

Select Top K Rewritten Queries

- $Q_1$‘: Model=A4
- $Q_2$‘: Model=Z4
- $Q_3$‘: Model=Boxster

We can select top K rewritten queries using F-measure

\[ F-Measure = \frac{(1+\alpha) \times P \times R}{\alpha \times P + R} \]

- $P$ – Estimated Precision
- $R$ – Estimated Recall based on $P$ and Estimated Selectivity
Handling Aggregate Queries

- As the fraction of incomplete tuples increases, the aggregates such as SUM and COUNT become increasingly inaccurate
  
  **SOLUTION:** Use query rewriting and missing value prediction to improve the accuracy of such aggregates

1. Issue the original query to the database and retrieve the base set.
2. Compute the certain aggregate over the base set tuples.
3. Use the base set to generate rewritten queries according to the QPIAD algorithm.
4. Issue the rewritten queries and retrieve the extended result set.
5. For each tuple in the extended result set the value most likely to be the tuple’s missing value.
6. If the most likely value is equal to the value specified in the original query, then include the tuple in the running uncertain aggregate total.
7. Return to the user the certain aggregate along with the uncertain aggregate.
Learning Statistics to Support Ranking & Rewriting

- Learning attribute correlations in the form of Approximate Functional Dependencies (AFDs) and Approximate Keys (AKeys)

- Learning value distributions using Naïve Bayes Classifiers (NBC)

- Learning Selectivity Estimates (EstSel) of Rewritten Queries based on:
  - Selectivity of rewritten query issued on sample \( SmplSel(Q) \)
  - Ratio of original database size over sample \( SmplRatio(R) \)
  - Percent of incomplete tuples while creating sample \( PerInc(R) \)

\[
EstSel(Q|R) = SmplSel(Q) \times SmplRatio(R) \times PerInc(R)
\]
Rewriting to Retrieve Relevant Uncertain Results

- An AFD tells us that for some fraction of the tuples, a car’s Model can be used to determine its Body.

AFD: Model \( \rightarrow \) Body

- Base set tuples are known to have Body=Convt therefore if we:
  1) Encounter a tuple having a Model in the base set
  2) And the tuple has a missing value for Body,
then it is likely that the tuple’s Body is in fact Convt.

- Given a query on attribute A, and an AFD B \( \rightarrow \) A, we generate rewritten queries by:
  - Determine the set of distinct values for the attributes contained in B
  - For each distinct value, generate a rewritten query constraining the corresponding attributes with the values from the distinct set.
Selecting/Ordering Top-K Rewritten Queries

- Sources may impose resource limitations on the # of queries we can issue

- SOLUTION: Use F-Measure based selection with configurable alpha parameter

- NOTE: F-Measure is used for selecting the top-K queries but does not determine the order in which they are sent

- Once we’ve selected the top-K rewritten queries, we must reorder them in order of their estimated precision

- Issuing the queries in order of estimated precision allows us to retrieve tuples in order of their final ranks
  - No need to re-rank tuples after retrieving them, simply show them to the user!

\[
F_\alpha = \frac{(1+\alpha) \cdot (P \cdot R)}{(\alpha \cdot P + R)}
\]

\[
F_0 = \frac{P \cdot R}{R} = P
\]

NOTE: All tuples returned for a single query are ranked equally
Explaining Results to the Users

Problem:
How to gain users trust when showing them incomplete tuples?

Q: (Make=Honda and Model=Accord and Price<$12,000)

<table>
<thead>
<tr>
<th>Make</th>
<th>Model</th>
<th>Year</th>
<th>Price</th>
<th>Color</th>
<th>Body</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Honda</td>
<td>Accord</td>
<td>2001</td>
<td>$10,500</td>
<td>Silver</td>
<td>Sedan</td>
<td></td>
</tr>
<tr>
<td>Honda</td>
<td>Accord</td>
<td>2002</td>
<td>$11,200</td>
<td>White</td>
<td>Coupe</td>
<td></td>
</tr>
<tr>
<td>Honda</td>
<td>Accord</td>
<td>1999</td>
<td>$9,000</td>
<td>Green</td>
<td>Sedan</td>
<td></td>
</tr>
<tr>
<td>?</td>
<td>Accord</td>
<td>2001</td>
<td>$11,700</td>
<td>Red</td>
<td>Sedan</td>
<td>This car is 100% likely to have Make=Honda given that its Model=Accord</td>
</tr>
<tr>
<td>Honda</td>
<td>?</td>
<td>2000</td>
<td>$10,100</td>
<td>Blue</td>
<td>Sedan</td>
<td>This car is 83% likely to have Model=Accord given that its Make=Honda and Body=Sedan</td>
</tr>
<tr>
<td>Honda</td>
<td>Accord</td>
<td>1999</td>
<td>?</td>
<td>Black</td>
<td>Sedan</td>
<td>This car is 71% likely to have Price&lt;$12,000 given that its Model=Accord and Year=1999</td>
</tr>
<tr>
<td>Honda</td>
<td>?</td>
<td>2002</td>
<td>$10,750</td>
<td>Silver</td>
<td>Coupe</td>
<td>This car is 42% likely to have Model=Accord given that its Make=Honda and Body=Coupe</td>
</tr>
</tbody>
</table>

Provide to the user:
- Certain Answers
- Relevant Uncertain Answers
- Explanations
Experimental Results – Learning Methods

- **Accuracy of Classifiers**
  
  Using AFDs during feature selection improves accuracy

  Accuracy of AFD-Enhanced NBC is comparable with BayesNet

- **Robustness w.r.t. Sample Size**
  
  QPIAD is robust even when faced with a relatively small data sample

  Similar results were obtained on the Census database
Handling Aggregate and Join Queries

- **Aggregate Queries**
  - As the fraction of incomplete tuples increases, the aggregates such as SUM and COUNT become increasingly inaccurate
    - **SOLUTION**: Use query rewriting and missing value prediction to improve the accuracy of such aggregates

- **Join Queries**
  - A join query can be thought of as individual queries over each source, the results of which are joined at the mediator
  - Estimated precision and estimated selectivity must be considered when deciding which queries to issue
  - When estimating precision/selectivity, estimates should be made for a query pair rather than for each individual query
    - **We must ensure that the results of each of the individual queries agree on their join attribute values**
Experimental Results – Ranking & Rewriting

- Effect of $\alpha$ on F-Measure

**Assumption:** 10 query limit on # of rewritten queries we are allowed to issue

Combined Effect = $\alpha + k$

As $\alpha$ increases, we allow queries with lower precision to be issued in order to obtain a higher throughput.
Experimental Results – Ranking & Rewriting

- **QPIAD vs. ALLRETURNED - Quality**

  ALLRETURNED has low precision because not all tuples with missing values on the constrained attributes are relevant to the query.

  QPIAD has a much higher precision than ALLRETURNED as it aims to retrieve tuples with missing values on the constrained attributes which are very likely to be relevant to the query.
Experimental Results – Ranking & Rewriting

- QPIAD vs. ALLRANKED - **Efficiency**

  **ALLRANKED** approach is often **infeasible** as direct retrieval of null values is not often allowed.

  **ALLRANKED** approach must retrieve all tuples w/ missing Body Style in order to achieve any nonzero recall.

  **QPIAD** only retrieves a subset of the tuples having missing values on constrained attributes, namely those which are highly likely to be relevant to the query.

  **QPIAD** is able to achieve the same level of recall as **ALLRANKED** while requiring much **fewer tuples** to be retrieved.

![Graph showing comparison between QPIAD and AllRanked for tuple retrieval versus recall.](image)
Experimental Results – Learning Methods

Accuracy of Classifiers

Using AFDs during feature selection improves accuracy

Accuracy of AFD-Enhanced NBC is comparable with BayesNet

Robustness w.r.t. Sample Size

QPIAD is robust even when faced with a relatively small data sample

Similar results were obtained on the Census database
Experimental Results - Extensions

- Aggregates
  
  Prediction of missing values increases the fraction of queries that achieve higher levels of accuracy
  
  Approximately 20% more queries achieve 100% accuracy when prediction is used

- Joins
  
  As alpha is increased, we obtain a higher recall without sacrificing much precision