Information Integration with Uncertainty: Performance

Inder K. Dumpa Raja S. Kota Fereidoon Sadri
Department of Computer Science
University of North Carolina at Greensboro
Greensboro, NC, USA
ikdumpa@uncg.edu, raja_k999@yahoo.com, f_sadri@uncg.edu

Abstract—Information integration and modeling and management of uncertain information have been active research areas for decades, with both areas receiving significant renewed interest in recent years. Research on information integration with uncertainty, on the other hand, is quite recent. In this paper we concentrate on recent works on uncertain-data integration. We present experimental results on the efficiency of recent algorithms for information integration from sources that contain uncertain data. Our experiments show the algorithms to be efficient, demonstrating a near linear performance in the total size of the uncertain data to be integrated.

Keywords: Information integration; uncertain data; possible worlds; integration performance.

I. INTRODUCTION

The importance of information integration with uncertainty, has been realized recently [1]-[4]. It has been observed that [2]: “While in traditional database management managing uncertainty and lineage seems like a nice feature, in data integration it becomes a necessity.” Research on information integration with uncertainty, is quite recent [5]-[11]. Researchers have concentrated on two main aspects of information integration with uncertainty. The first category considers integration of definite data with uncertain schema mappings [6], [7]. The second category considers integration of uncertain data [5], [8], [9], [11]. Our work in [11] falls in the second category. We presented algorithms for the integration of uncertain data, and justified the correctness of the algorithms by showing they coincided with the integration formalism presented in the foundational work [5]. In this paper, we report the implementation and experimental results of these algorithms. Note that information integration has many dimensions and involves a number of important tasks such as data cleaning, data linkage, schema mapping, data standardization, query translation, and query optimization. We concentrate only on the issue of information integration from sources that contain uncertain data in this paper.

This paper is organized as follows. In Section II, we present an introduction to integration of information from sources with uncertain data, and briefly discuss the uncertain data integration algorithms of [11]. Our implementation is presented in Section III, and experimental results in Section IV. Our experiments show the algorithms to be efficient, demonstrating a near linear performance in the total size of the uncertain data to be integrated. Conclusion and future work are presented in Section V.

II. PRELIMINARIES

In this section we will review some of the recent works regarding issues and algorithms for the integration of uncertain data.

A. Information Integration with Uncertainty: Foundations

Foundations of information integration with uncertainty have been discussed in [5], [11]. We will present a brief summary here. We begin with an example from [11].

Example 1: John and Jane are talking about fellow student Bob. John says “I am taking CS100 and CS101, and Bob is in one of them, but not in both.” Jane says “I am taking CS101 and CS102 and Bob is in one of them, but not in both.”

Intuitively, if we integrate the information from these two sources (John and Jane), we should infer that Bob is either taking CS101, or he is taking both CS100 and CS102.

The model used in [5], [11] for the representation of uncertain information is the well-known possible-worlds model [12]. In Example 1, the information presented by the two sources (John and Jane) is represented by the possible worlds shown in Figures 1 and 2. The possible worlds of the result of integration is shown in Figure 3.

![Figure 1: Possible Worlds of source S1](image1)

![Figure 2: Possible Worlds of source S2](image2)

![Figure 3: Possible Worlds of the Integration for Example 1](image3)
shown to be equivalent to the integration formalism of [5] which is based on the concept of superset-containment. Interested readers are referred to [11] for details.

First, we should point out that the pure possible world model is not adequate for uncertain-data integration applications. We need additional information, namely, the set of all tuples. The following example demonstrates the possible-worlds with tuple sets model.

Example 2: Andy and Jane are talking about fellow student Bob. Andy says “I am taking CS100, CS101, and CS102 and Bob is in either CS100 or CS101 but not in both.” Jane says “I am taking CS101 and CS102 and Bob is in one of them, but not in both.”

Intuitively, if we integrate the information from these two sources, we should infer that Bob is taking CS101. The second possibility from Example 1 is not valid anymore since Andy’s statement rules out the possibility that Bob is taking 102.

To justify this answer, we observe that pairwise combination of possible worlds from the two sources result in the four possible worlds of Figure 4. But only the second possible world is a valid combination, and the other three are not valid. The first world is not valid since Andy states that he is taking CS100, CS101, and CS102 and Bob is taking 100 of 101 but not both. So Bob can not be in both 100 and 101. The third and fourth worlds are not valid due to Andy’s statement too. He is taking 102 (among other courses) and states that Bob is taking 100 or 101. Hence Bob can not be in 102. Note that the last world is also not valid due to Jane’s statements. She says that she is in 101 and 102, and Bob is in one of them, but not both. The only valid combination is the second world: Bob must be taking CS101.

Figure 4: Pairwise combination of possible worlds from the two sources.

However, the possible-worlds representations of these sources (Andy and Jane) are exactly the same as those of Example 1 (Figures 1 and 2). Only when we add the tuple-set to possible worlds of Andy, namely \{(Bob, CS100), (Bob, CS101), (Bob, CS102)\}, it becomes explicit that Andy’s statement eliminates the possibility that Bob is taking CS102.

Hence, we will use the following definition from [5] for uncertain databases that adds tuple sets to the possible-worlds model. Note that to simplify presentation, it is assumed that possible worlds are sets of tuples in a single relation. We adopt the same convention throughout this paper.

Definition 1: (Uncertain Database). An uncertain database \(U\) consists of a finite set of tuples \(T(U)\) and a nonempty set of possible worlds \(PW(U) = \{D_1, \ldots, D_m\}\), where each \(D_i \subseteq T(U)\) is a certain database.

B. Integration Using Logical Representation

The following definitions and results are from [11].

Given an uncertain database \(U\), we assign a propositional variable \(x_i\) to each tuple \(t_i \in T(U)\). We define the formula \(f_j\) corresponding to a possible world \(D_j\), and the formula \(f\) corresponding to the uncertain database \(U\) as follows:

Definition 2: (Logical Representation of an Uncertain Database). Let \(D_j\) be a database in the possible worlds of uncertain Database \(U\). Construct a formula as the conjunction of all variables \(x_i\) where the corresponding tuple \(t_i\) is in \(D_j\), and the conjunction of \(\neg x_i\) where the corresponding tuple \(t_i\) is not in \(D_j\). That is,

\[
f_j = \bigwedge_{t_i \in D_j} x_i \bigwedge_{t_i \notin D_j} \neg x_i \tag{1}
\]

The formula corresponding to the uncertain database \(U\) is the disjunction of the formulas corresponding to the possible worlds of \(U\). That is,

\[
f = \bigvee_{D_j \in PW(U)} f_j \tag{2}
\]

Now we can integrate uncertain databases using their logical representations as follows:

Let \(S_1, \ldots, S_n\) be sources containing (uncertain) databases \(U_1, \ldots, U_n\). Let the propositional formulas corresponding to \(U_1, \ldots, U_n\) be \(f_1, \ldots, f_n\). We obtain the formula \(f\) corresponding to the uncertain database resulting from integrating \(U_1, \ldots, U_n\) by conjuncting the formulas of the databases:

\[
f = f_1 \land \cdots \land f_n
\]

Example 3: (Integration Using Logical Representation) Consider Example 1. The uncertain database corresponding to John’s statement is represented by \((x_1 \land \neg x_2) \lor (\neg x_1 \land x_2)\), where \(x_1\), and \(x_2\) correspond to the tuples (Bob, CS100) and (Bob, CS101), respectively. The uncertain database corresponding to Jane’s statement is represented by \((x_2 \land \neg x_3) \lor (\neg x_2 \land x_3)\), where \(x_2\) is as above and \(x_3\) corresponds to the tuple (Bob, CS102). The integration in this case is obtained as

\[
((x_1 \land \neg x_2) \lor (\neg x_1 \land x_2)) \land ((x_2 \land \neg x_3) \lor (\neg x_2 \land x_3))
\]

which corresponds to the possible worlds of Figure 3. The result is consistent with our intuition: Based on statements by John and Jan, Bob is taking either CS101 or both CS100 and CS102.

Now consider Example 2. The uncertain database corresponding to Andy’s statement is represented by \((x_1 \land \neg x_2 \land \neg x_3) \lor (\neg x_1 \land x_2 \land \neg x_3)\), where \(x_1\), \(x_2\), and \(x_3\) represent (Bob, CS100), (Bob, CS101), and (Bob, CS102), respectively. The uncertain database corresponding to Jane’s statement is represented by \((x_2 \land \neg x_3) \lor (\neg x_2 \land x_3)\). The integration in this case is obtained as

\[
((x_1 \land \neg x_2 \land \neg x_3) \lor (\neg x_1 \land x_2 \land \neg x_3)) \land ((x_2 \land \neg x_3) \lor (\neg x_2 \land x_3))
\]

which corresponds to the possible worlds of Figure 3. The result is consistent with our intuition: Based on statements by John and Jan, Bob is taking either CS101 or both CS100 and CS102.
corresponding to the (in this case, definite) relation consisting only of the tuple (Bob, CS101) (Figure 5). Again, this result is consistent with our intuition: Based on statements by Andy and Jane, Bob is taking CS101.

<table>
<thead>
<tr>
<th>student</th>
<th>course</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bob</td>
<td>CS101</td>
</tr>
</tbody>
</table>

Figure 5: Possible Worlds of the Integration for Example 2

C. An Alternative View of Integration

Let $S_1, \ldots, S_p$ be sources containing (uncertain) databases $U_1, \ldots, U_n$. Let $PW(U_i)$ represent the set of possible worlds of uncertain database $U_i$, and $T_j$ represent the tuple set of $U_i$. We can regard the integration of information from these sources as follows:

**Definition 3**: (Compatible Set of Possible-Worlds Relations). Consider a set of $n$ relations $\{w_1, \ldots, w_n\}$ where each $w_i$ is a relation in the set of possible worlds of $U_i$, that is, $w_i \in PW(U_i)$, $i = 1, \ldots, n$. If there is a tuple $t$ in a relation $w_i$, that it is also in $T_j - w_j$ for some other possible-world relation $w_j$, we say the set of possible-world relations $\{w_1, \ldots, w_n\}$ is not compatible. Otherwise, $\{w_1, \ldots, w_n\}$ is compatible.

Note that $t \in T_j - w_j$ means that according to source $S_j$, the tuple $t$ can not exist (is ruled out) in $w_j$. Hence, if a set of possible world-relations is not compatible, they can not be integrated. A compatible set of possible-world relations $\{w_1, \ldots, w_n\}$ can be integrated, and the resulting relation contains all the tuples in the relations, that is, the result of integrating $w_1, \ldots, w_n$ is $w = \cup_{i=1}^{n} w_i$.

Hence, to integrate sources $S_1, \ldots, S_n$, we can compute the possible-worlds relations of the integration by

1) forming all possible combinations $\{w_1, \ldots, w_n\}$, $w_i \in PW(U_i)$,
2) determining compatible sets, and
3) obtaining the union of the relations in the compatible set.

This alternative characterization of integration results in a simpler integration algorithm. We use the logical formulation only to determine compatible sets of possible worlds, and then we obtain the result by calculating the union of the possible worlds in each compatible set. We have used this characterization to design our integration algorithm (Section III).

**Example 4**: (Alternative View of Integration) Consider Example 1. The possible-worlds relations of the uncertain database corresponding to John’s statement were shown in Figure 1, and the possible-worlds relations of the uncertain database corresponding to Jane’s statement were shown in Figure 2. In this case, the compatible sets of possible worlds are $\{D_1, D_4\}$ and $\{D_2, D_3\}$. We can conveniently represent the compatibility of possible-worlds relations for two sources by a bi-partite graph, such as Figure 6. The possible-worlds of the result of integration is shown in Figure 3.
according to the algorithm of Section II-B. The formulas are conveniently represented by vectors, which can be used to easily implement logical operations over the expressions.

- The module determines which sets of possible world relations, one from each source, are compatible and hence can produce a possible world relation in the integration. This is done by computing the conjunction (logical and) of the corresponding formulas of the possible world relations. If the result is false, the set of possible worlds are not compatible. Otherwise they are compatible.
- For all compatible sets of possible worlds, the module generates the resulting relation by unioning the possible worlds relations in the set. It stores the integrated relation in the Oracle database for the integration result.
- Once all compatible possible worlds sets are processed, the module displays the total time for the integration.

IV. EXPERIMENTAL EVALUATION

We carried out a large number of experiments to evaluate the performance of the integration algorithm. The experiments were executed on a 2.10 GHz Intel i3 CPU with 4.00 GB RAM, 64-bit Windows 7 Operating System using Java 1.7 and Oracle XE 10g. The first few experiments evaluated the performance of the integration algorithm for integrating information from two sources.

In the first set of experiments the number of possible world relations of the two sources were kept constant, and test cases were generated by varying the number of tuples in the possible world relations (and hence, varying the size of uncertain databases to be integrated). Figure 7 shows the result of these experiments. The horizontal axis shows the total size (KB) of databases to be integrated. The vertical axis shows the time needed for the integration (sec). Again, the experiments show that the integration algorithm is almost linear in the total size of databases to be integrated (no matter whether the size increase is due to larger number of possible worlds per sources, or larger possible world relation sizes.)

In the second set of experiments, we varied the number of possible world relations of the two sources while keeping the number of tuples constant. Figure 8 shows the result of these experiments. The horizontal axis shows the total size (KB) of databases to be integrated. The vertical axis shows the time needed for the integration (sec). Again, the experiments show that the integration algorithm is almost linear in the total size of databases to be integrated (no matter whether the size increase is due to larger number of possible worlds per sources, or larger possible world relation sizes.)

Table I: Integration experiments; with total size (almost) constant

<table>
<thead>
<tr>
<th>PWs</th>
<th>tuples</th>
<th>size</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>110</td>
<td>99.9</td>
<td>114</td>
</tr>
<tr>
<td>19</td>
<td>116</td>
<td>99.8</td>
<td>114</td>
</tr>
<tr>
<td>18</td>
<td>122</td>
<td>99.6</td>
<td>109</td>
</tr>
<tr>
<td>17</td>
<td>130</td>
<td>100.4</td>
<td>113</td>
</tr>
<tr>
<td>16</td>
<td>138</td>
<td>100.2</td>
<td>116</td>
</tr>
<tr>
<td>15</td>
<td>147</td>
<td>100.0</td>
<td>114</td>
</tr>
<tr>
<td>14</td>
<td>158</td>
<td>100.3</td>
<td>111</td>
</tr>
<tr>
<td>13</td>
<td>170</td>
<td>100.3</td>
<td>116</td>
</tr>
<tr>
<td>12</td>
<td>184</td>
<td>100.2</td>
<td>113</td>
</tr>
<tr>
<td>11</td>
<td>200</td>
<td>99.8</td>
<td>113</td>
</tr>
<tr>
<td>10</td>
<td>220</td>
<td>100.0</td>
<td>114</td>
</tr>
<tr>
<td>9</td>
<td>245</td>
<td>100.0</td>
<td>114</td>
</tr>
<tr>
<td>8</td>
<td>275</td>
<td>100.0</td>
<td>114</td>
</tr>
<tr>
<td>7</td>
<td>315</td>
<td>99.9</td>
<td>111</td>
</tr>
<tr>
<td>6</td>
<td>368</td>
<td>100.2</td>
<td>115</td>
</tr>
<tr>
<td>5</td>
<td>442</td>
<td>100.3</td>
<td>116</td>
</tr>
<tr>
<td>4</td>
<td>553</td>
<td>100.1</td>
<td>119</td>
</tr>
<tr>
<td>3</td>
<td>737</td>
<td>100.4</td>
<td>120</td>
</tr>
</tbody>
</table>

Figures 9 and 10 plot the integration time in the experiments of Table I against the number of possible worlds...
and the number of tuples in each possible world. The x-axis of Figure 10 (number of tuples in each possible world) is logarithmic to better demonstrate the effect of number of tuples, ranging from 110 to 737 in the experiments. These experiments show that the number of possible worlds and their sizes are not factors in the performance of the integration algorithm when the total size is constant. In other words, integration time is almost constant when number of possible world relations and their sizes change while the total size is fixed. This observation is counter-intuitive since the integration algorithm needs to determine, for every pair of possible worlds \((w_1, w_2)\), whether they are compatible, where \(w_1\) and \(w_2\) belong to source 1 and source 2, respectively. But the impact of number of tuples (smaller number of tuples for larger number of possible world relations) counterbalances the impact of number of possible worlds.

In the next set of experiments we generated test cases by varying both the number of possible worlds and the number of tuples in each possible world (and hence, varying the total size). The results are summarized in Table II. The columns are, respectively, number of possible worlds for each source, number of tuples in each possible world, total size, and integration time. Figure 11 plots the integration time in these experiments against total size. It confirms a near linear performance of the algorithms as a function of the total size of the integration. Figure 12 plots the integration time against the number of possible world relations and number of tuples in each possible world.

Table II: Integration experiments; varying number of possible worlds and number of tuples in each possible world

<table>
<thead>
<tr>
<th>PWs</th>
<th>tuples</th>
<th>size</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>4</td>
<td>0.7</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>2.9</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
<td>6.5</td>
<td>3</td>
</tr>
<tr>
<td>16</td>
<td>16</td>
<td>11.5</td>
<td>4</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>18</td>
<td>7</td>
</tr>
<tr>
<td>24</td>
<td>24</td>
<td>26.1</td>
<td>13</td>
</tr>
<tr>
<td>28</td>
<td>28</td>
<td>35.4</td>
<td>22</td>
</tr>
<tr>
<td>32</td>
<td>32</td>
<td>46.4</td>
<td>33</td>
</tr>
<tr>
<td>36</td>
<td>36</td>
<td>58.8</td>
<td>50</td>
</tr>
<tr>
<td>40</td>
<td>40</td>
<td>72.7</td>
<td>70</td>
</tr>
<tr>
<td>44</td>
<td>44</td>
<td>87.8</td>
<td>89</td>
</tr>
<tr>
<td>48</td>
<td>48</td>
<td>104.5</td>
<td>121</td>
</tr>
<tr>
<td>52</td>
<td>52</td>
<td>122.5</td>
<td>149</td>
</tr>
<tr>
<td>56</td>
<td>56</td>
<td>142</td>
<td>182</td>
</tr>
<tr>
<td>60</td>
<td>60</td>
<td>163.4</td>
<td>219</td>
</tr>
</tbody>
</table>

In the next set of experiments we evaluate the performance of the integration algorithm when integrating data from more than two information sources. We generated test cases by varying the number of information sources, while keeping the total size constant. Figure 13 plots the integration time against the number of possible worlds. This performance was very unexpected. As seen from this graph, the algorithm has an almost constant time up to about 10 information sources, then the integration time increases sharply. We postulated that the reason for the sharp increase is memory saturation, which
forces the execution to use virtual memory. In our experiments, the number of possible world relations for each source was kept constant at 3, and constant total size was achieved by varying the number of tuples in each possible world. So, with 10 sources, the number of integration combinations to generate was $3^{10} = 59,049$. This number increases to $3^{11} = 177,147$ for 11 sources, and to $3^{12} = 531,441$ for 12 sources. The memory of our testbed system saturates at about 10 sources.

To test our hypothesis, we executed the exact same experiments on systems with lower (2GB) and higher (8GB) memory sizes. The graphs for these experiments have the same shape, except at lower memory size the graph is shifted to the left, and at higher memory size the graph is shifted to the right. In other words, the sharp increase happens at a lower number of sources for the lower memory size, and at a higher number of sources for higher memory size. These additional experiments confirm our hypothesis that the change in the performance of the integration algorithm, from constant time to almost linear, is a result of memory saturation. So, our final conclusion is that, given adequate memory, the performance of the integration algorithm is a linear function (approximately) of the total size of the integration instance. It is not sensitive to the other factors, namely, number of information sources, number of possible worlds relations in the sources, and number of tuples in the possible world relation, when the total size is kept constant.

V. Conclusion

We presented our implementation and experimental evaluation of the uncertain-data integration algorithms of [11]. Our experiments show the algorithms to be efficient, demonstrating a near linear performance in the total size of the uncertain data to be integrated.

There are a number of important issues that require further investigation. First, uncertain schema mappings is another source of uncertainty in information integration. We would like to develop integration algorithms for this case, with definite or uncertain data. The integration algorithm is a good candidate for parallel computation, in particular, using the map-reduce framework [13]. A future direction would be to implement the integration using Hadoop running on a large number of computers. More importantly, we would like to devise integration algorithms to work with compact representations of uncertain data, such as the probabilistic relational model of [14], [15].

REFERENCES