Inconsistency Management Policies

Maria Vanina Martinez¹, Francesco Parisi², Andrea Pugliese², Gerardo Simari¹, and VS Subrahmanian¹

¹Department of Computer Science and UMIACS
University of Maryland College Park, USA

²University of Calabria, Rende, Italy

11th International Conference on Principles of Knowledge Representation and Reasoning
September 2008 – Sydney, Australia
Motivation

- If company A and B merge, so do their databases.

- *Problem:* there exist several salary records for employee John in a given month when it is expected to be only one record per employee per month.

- What would a database user do about this inconsistency?
  - How would people from the business office use this data?
  - How would a bank use this data if it is considering John for a loan?
  - How would the IRS use this data?
Motivation

- Most relational databases are inconsistent or are expected to be.
- *Common assumption:* there exists an *epistemically correct* way of solving or reasoning about the inconsistency; however:
  - DBMS managers do not necessarily have the knowledge to decide which data is “correct” and which is not.
  - Different users of the same DB might have different needs.
- *It is important to enable* users to bring their *application-specific knowledge* to bear when dealing with inconsistency.
Contribution

- Introduce Inconsistency Management Policy (IMP).
- The user has the last word on how to handle the inconsistency, depending on the application’s needs.
- IMPs allow users to remove inconsistency completely or to have part or all of the inconsistency persist.
  - Different kinds of policies allow different ways of managing the inconsistency.
  - No previous work attempts to handle all these possibilities.
- Analysis IMPs for single and multiple functional dependencies (FDs for short).
- Different semantics to handle multiple FDs.
- Extension of Relational Algebra: IMPs embedded as operators.
Preliminaries

• Assume a relational schema \( S(A_1, \ldots, A_n) \); \( t[A_i] \) denotes the value of the attribute \( A_i \) of tuple \( t \).

• Assume FDs of the form:

\[
A'_1, \ldots, A'_k \rightarrow A'_m, \quad \text{RHS}(fd) = A'_m.
\]

  – If two tuples agree on the attributes in the antecedent (LHS), the attributes in the RHS must also agree.

• We base our work on two notions:

  – **Culprits** or minimal inconsistent subsets.

  – **Clusters**: Sets of overlapping culprits.

    • Clusters group together tuples that are inconsistent with each other.

    • Equivalence classes w.r.t. culprit overlapping relation.
Preliminaries: Culprits and Clusters

- Consider the following relation $Employee$ (we call it $R$) and a set $F$ of FDs $F = \{Name \rightarrow Salary\}$:

<table>
<thead>
<tr>
<th>Name</th>
<th>Salary</th>
<th>Tax_bracket</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>70K</td>
<td>15</td>
<td>$s_1$</td>
</tr>
<tr>
<td>John</td>
<td>80K</td>
<td>20</td>
<td>$s_2$</td>
</tr>
<tr>
<td>John</td>
<td>70K</td>
<td>25</td>
<td>$s_3$</td>
</tr>
<tr>
<td>Mary</td>
<td>90K</td>
<td>30</td>
<td>$s_1$</td>
</tr>
</tbody>
</table>

- $Culprits(R, F) = \{\{t_1, t_2\}, \{t_2, t_3\}\}$
- $Clusters(R, F) = \{\{t_1, t_2, t_3\}\}$

- If we had a tuple $t_5 = (Mary, 80K, 30)$, then
  $$Clusters(R, F) = \{\{t_1, t_2, t_3\}, \{t_4, t_5\}\}$$
Inconsistency Management Policies

• An IMP w.r.t. a set of FDs is a function applied to a relation that results in a new relation with the intention of reducing inconsistency.

• IMP AXIOMS: An IMP for a relation \( R \) w.r.t. the set \( F \) over \( R \) is a function \( \gamma_F \) from \( R \) to \( R' = \gamma_F(R) \) such that:
  
  – Tuples that do not belong to any culprit cannot be eliminated or changed.
  
  – If \( t' \in R' - R \), there exist cluster \( c \) and tuple \( t \) in \( R \), s.t. for each attribute \( A \) (\( A \) doesn’t appear in any FD), \( t[A] = t'[A] \).
  
  – The application of the \( \gamma_F \) cannot increase the number of culprits for any FD in \( F \).
  
  – \( |R| \geq |R'| \), i.e., the cardinality cannot increase.
IMP for Single FD

- Our goal is to allow the end user to choose a policy that best describes his needs (including not changing the database at all).

- The user needs tools to:
  - Eliminate tuples from a cluster.
  - Change values of attributes from tuples in clusters.

- Three different kinds of Singular IMPs:
  - Tuple-based policies.
  - Value-based policies.
  - Interval-based policies.
Tuple-based Policies

- A tuple-based IMP allows replacing a cluster w.r.t. an FD by a subset of the same cluster.
- Generalization of maximal consistent subsets and repairs.
- A tuple-based IMP for relation *Employee* could be based on the fact that the user does not trust source $s_3$, and thus decides to eliminate all tuples coming from it.

**Resulting Relation**

<table>
<thead>
<tr>
<th>Name</th>
<th>Salary</th>
<th>Tax_bracket</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>70K</td>
<td>15</td>
</tr>
<tr>
<td>John</td>
<td>80K</td>
<td>20</td>
</tr>
<tr>
<td>John</td>
<td>70K</td>
<td>25</td>
</tr>
<tr>
<td>Mary</td>
<td>90K</td>
<td>30</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Salary</th>
<th>Tax_bracket</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>70K</td>
<td>15</td>
</tr>
<tr>
<td>John</td>
<td>80K</td>
<td>20</td>
</tr>
<tr>
<td>Mary</td>
<td>90K</td>
<td>30</td>
</tr>
</tbody>
</table>
Value-based Policies

- For each cluster, a value-based IMP allows to reduce the number of distinct values for attributes in $RHS(fd)$.
- For the *Employee* relation, suppose the user *knows* that in general, source $s_1$ has more recent information than $s_2$ w.r.t. *Salary*. He decides to reset the $s_2$ information with that from $s_1$.

<table>
<thead>
<tr>
<th>Source</th>
<th>Name</th>
<th>Salary</th>
<th>Tax_bracket</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$</td>
<td>John</td>
<td>70K</td>
<td>15</td>
</tr>
<tr>
<td>$s_2$</td>
<td>John</td>
<td>80K</td>
<td>20</td>
</tr>
<tr>
<td>$s_3$</td>
<td>John</td>
<td>70K</td>
<td>25</td>
</tr>
<tr>
<td>$s_1$</td>
<td>Mary</td>
<td>90K</td>
<td>30</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Source</th>
<th>Name</th>
<th>Salary</th>
<th>Tax_bracket</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$</td>
<td>John</td>
<td>70K</td>
<td>15</td>
</tr>
<tr>
<td>$s_2$</td>
<td>John</td>
<td>70K</td>
<td>20</td>
</tr>
<tr>
<td>$s_3$</td>
<td>John</td>
<td>70K</td>
<td>25</td>
</tr>
<tr>
<td>$s_1$</td>
<td>Mary</td>
<td>90K</td>
<td>30</td>
</tr>
</tbody>
</table>

Resulting Relation
Interval-based Policies

- An interval-based IMP allows any tuple in a cluster to be replaced by a new one that has different values for the attribute in $RHS(fd)$. New values are within the interval defined by the values in the attribute on the $RHS(fd)$.

- For the Employee relation:
  - Replace the values for Salary of tuples in the cluster for which the value of Tax_bracket is less than or equal to 20, to the mean among all Salary values in the cluster. This changes Salary in $t_1$ and $t_2$ to 73.33K.

<table>
<thead>
<tr>
<th>Name</th>
<th>Salary</th>
<th>Tax_bracket</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>John</td>
<td>70K</td>
<td>$s_1$</td>
</tr>
<tr>
<td>$t_2$</td>
<td>John</td>
<td>80K</td>
<td>$s_2$</td>
</tr>
<tr>
<td>$t_3$</td>
<td>John</td>
<td>70K</td>
<td>$s_3$</td>
</tr>
<tr>
<td>$t_4$</td>
<td>Mary</td>
<td>90K</td>
<td>$s_1$</td>
</tr>
</tbody>
</table>
Multi-dependency IMPs

- Assume an IMP for each $fd \in F$, and a partial ordering $\leq_F$, different orders of application of individual policies might yield different results.

- Let $\mu_F(R)$ be a multi-dependency policy (MDIMP); we developed two semantics:
  - **Fixed order semantics**: assume the existence of a total order $o = \langle fd_1, ..., fd_k \rangle$. $\mu^o_F(R)$ is the result from applying the singular IMPs in the order determined by $o$, this is $\mu^o_F(R) = \gamma_{fd_k}(...\gamma_{fd_2}(\gamma_{fd_1}(R))...)$.  
  - **Core semantics**: $\mu_F(R)$ includes only those tuples that appear in the intersection of all possible fixed order applications.
Complexity Results on MDIMPs

- **Theorem:** Given a tuple \( t \in R \), determining whether \( t \in \text{Core}(R, F, \leq_F) \) is \( \text{coNP-complete} \).

- **Theorem:** Given a tuple \( t \in R \), determining whether there is a total ordering \( o \), consistent with \( \leq_F \) such that \( t \in \mu^o_F(R) \) is \( \text{NP-complete} \).

- If we assume schemas of bounded size then the previous problems are in \( \text{PTIME} \).

- Assuming an arbitrary but fixed set \( F \), and that all individual policies are computable in polynomial time w.r.t. the number of tuples in \( R \), the application of an MDIMP \( \mu^o_F(R) \) is polynomial w.r.t. the number of tuples in \( R \).
Extension of Relational Algebra

- Classical relational operators augmented by the application of an IMP.
- Given two relations \( R_1 \) and \( R_2 \), the corresponding sets of FDs \( F_1 \) and \( F_2 \), a classical relational operator \( op \), and two MDIMPs \( \mu_{F_1} \) and \( \mu_{F_2} \):
  - A **Policy-first Inconsistency Management Operator** applies \( op \) to the relations obtained from the application of the policies.
  - A **Policy-last Inconsistency Management Operator** applies an MDIMP that handles the union of \( F_1 \) and \( F_2 \) to the relation that results from applying \( op \).
- In general, **policy-last is not equivalent to policy-first**. We proved some conditions for equivalence.
Extension of RA: Projection

- \textit{Policy-last projection operator} requires the projection to be made over a set $X$ that is a superset of the attributes involved in the FDs.

- \textbf{Theorem:} \textit{Policy-last projection is equivalent to policy-first projection} if and only if:
  - the policy only depends on values of attributed involved in the functional dependencies with which the policy is associated, and
  - policy does not depend on duplicate values of an attribute \textit{(e.g., max or min but not average)}.

Extension of RA: Selection - Union

- **Theorem:** For *tuple-based policies* where the application of the policy is equivalent to a selection operation, *policy-last selection is equivalent to policy-first selection.*
  - Selection of a tuple based only on values of the tuple.

- **Theorem:** *Policy-last union is equivalent to policy-first union* under the following circumstances:

  - Given a set $X$ and $fd: X \rightarrow B$, for any IMP $\gamma_{fd}$ it is the case that $\gamma_{fd}(R_1 \cup R_2) = \gamma_{fd}(R_1) \cup \gamma_{fd}(R_2)$ if and only if $\pi_X(R_1) \cap \pi_X(R_2) = \emptyset$.
Cartesian Product

- **Theorem.** Given $R_1$ and $R_2$, FDs $fd_1$ and $fd_2$, for any pair of ratio-invariant policies $\gamma_{fd_1}$ and $\gamma_{fd_2}$:
  
  - $\gamma_{fd_1}(R_1 \times R_2) = \gamma_{fd_1}(R_1) \times R_2$
  
  - $\gamma_{fd_2}(R_1 \times R_2) = R_1 \times \gamma_{fd_2}(R_2)$

  - $\gamma_{fd_1}(\gamma_{fd_2}(R_1 \times R_2)) = \gamma_{fd_1}(R_1) \times \gamma_{fd_2}(R_2)$

  - $\gamma_{fd_1}(\gamma_{fd_2}(R_1 \times R_2)) = \gamma_{fd_2}(\gamma_{fd_1}(R_1 \times R_2))$

- **Ratio-invariant** policy: its application to a multi-set is equivalent to its application to another multi-set that contains the same proportion of tuples.
  
  - Result do not depend on the number of elements having the same value (e.g., lowest value).
Summary and Conclusion

• We introduced the concept of IMP: *The user can decide to either remove inconsistency completely or to allow part or all of the inconsistency to persist.*
  – Three different kinds of IMPs allow to delete tuples and change values of attributes.

• IMPs to manage multiple FDs:
  – Two semantics: Fixed order and Core.
  – PTIME computation for bounded schemas.

• We extended the Relational Algebra embedding IMPs as operators:
  – Analysis of interaction with basic operators.
  – Policy-first vs. Policy-last operators: conditions for equivalence for basic operators.