Avoiding Error-Prone Reordering Optimization During Legal Systems Migration

Youlin Fang[1], Heng Wang[2], and Dongqing Yang[3]

[1] College of Computer Science, Bejing University of Technology,
Beijing 100022, P.R.China
ylfang@bjut.edu.cn

[2] School of Economics and Management, Tsinghua University,
Beijing 100084, P.R.China
wangh@em.tsinghua.edu.cn

[3] School of Electrical Engineering and Computer Science, Peking University,
Beijing 100871, P.R.China
dqyang@db.pku.edu.cn

Abstract. During Legal information systems migrations, one major problem is to handle attribute value cleaning. In the paper, we first show many data cleaning steps process on the values of the same data attributes and their derivations, and users may ignore or be puzzled by the data transforms that are defined to clean and transform the data sets, and such process can also be prone to error during process optimization. In this paper, we first define two major such problems as assignment conflict and range conflict,and giving problem definitions for such conflicts. Then we present two separate algorithms respectively to discover and solve the conflicts.

1 Introduction

As data extraction, transformation and loading(ETL) are being widely established to meet strategic business objectives, many organizations have found that ETL process, especially data cleaning and transformation is a complex, expensive, and bothersome process. Costly mistakes have been made in the designing stage of data cleaning[2][3][4][5].

In the design of data cleaning procedures, users may have to process the same property many times, e.g. substituting a code up to the specified business strategy, users usually do not know the error and potentially error in their cleaning process models. As asserted in [1], model checking is very important for such process, so users should be more careful to check whether their migration model is correct, and should avoiding making wrong migration optimization of the process model when executing. In this paper, we discuss the problem, and present algorithms to solve such conflict.

1.1 Outline of Paper

The remainder of the paper proceeds as follows. Section 2 gives a overview of the problem. Section 3 describes the equi-transformation conflict problem and containment-transformation conflict problem, and gives out typical types of these conflicts. In section 4,we first describe transformation algebraic properties, then present detecting algorithms for these two problem and propose methods to solve them . Section 5 concludes the paper.

1.2 Related Work

Erhard Rahm and Hong Hai Do[5]classify data quality problems that are addressed by data cleaning and provide an overview of the main solution approaches. They also discuss current tool support for data cleaning.

Serge Abiteboul and Sophie Cluet etc.[6] discuss a tool, namely ARKTOS, which they have developed capable of modeling and executing practical scenarios, to deal with the complexity and efficiency of the transformation and cleaning tasks by providing explicit primitives for the capturing of common tasks. ARKTOS provides three ways to describe such a scenario, including a graphical point-and-click front end and two declarative languages: XADL (an XML variant), which is more verbose and easy to read and SADL (an SQL-like language) which has a quite compact syntax and is, thus, easier for authoring.

Xiang Fu and Tevfik Bultan etc.[1] introduce the problem of automated verification of software systems that manipulate XML data. And it encourages us to rethink the data cleaning model in the data migration process.

Our contributions: The following are the major contributions of this paper:

– We present a overview of asssignment conflicts lying in data transformation modelling, then describe typical assignment conflicts of the modelling problem.
– We then present algorithms for effective finding and detecting potential conflict schedules, and present methods of determine the transformation schedule order by its partial order.

2 Problem Formulation

For a given data set S_1, we provide a sequence of data cleaning and transformation process and get the result S_2, that is $S_1\{T_1, \cdots, T_n\}S_2$, where $T_i(1 \leq i \leq n)$ is a mapping or a transformation procedure of a property in S_1 to a property in S_2. These transformation process are parallel, so that the designer do not give a order on which transform is processed.

2.1 Data Transformation

A transformation is a sequence of one and above transform in a given order. The process of data transformation is according to the right rule of transaction process in database,that is if a transaction is executed without other transactions or system errors, and the database is consistent at the beginning of the start, the database remains in consistence at the end of the transaction. The rule appears in transformation processing as:

- transformation process is atomic: A transformation must be executed as a single or no execution at all. If part of the transformation is executed, then the database is inconsistent under most of the circumstance.
- Parallel execution of transformation may lead to inconsistence in database. Otherwise we control the inter-affect between them.

2.2 Transformation Schedule

Definition 1. *(Transformation Schedule) A transformation schedule is a sequence of relation operations of multiple transformations according to execution order.*

When we discuss the transformation schedule, an important transformation is the operation which S_1 and S_2 have the common schema. As S_1 and S_2 have the same schema, if data are directly processed through copy at every step of transformation, the it need too much I/O cost.

2.3 Transformation Schedule Serialization

In data transformation schedule, one kind is serialized transformation schedule, that is if a transformation is finished after all of its operations are finished, the operations in the next transformation can then start and without operation mixed between two transformations, then we say such schedule is serial. More detailed, for any two arbitrary transformation T and T', if a operation in T is executed before a operation of T', then all the operations in T is executed before the execution of all the operations in T', a transformation schedule satisfying it is serial.

2.4 Transformation Modeling

In the modeling of data cleaning, designers may have two approaches to modeling data cleaning. One approach is directly writing scripts of cleaning, the other is achieved with the aid of cleaning modeling tools. Under both circumstance, the designer may not have a right process for the cleaning process, he can not assure the process is error-free in modeling. In fact, in the modeling of data cleaning process, there are many steps of transformation between two data sets, and the user know what he want when writing down these steps. But the problem arise when the steps and process are too many, these steps may process the data sets with the same schema, or with the same columns or properties.In process there are conflicts potentially. The users are not aware of such conflicts.

In the next section, we will discuss two typical conflicts in transformation modeling.

3 Semantic Conflicts in Data Transformation Modeling

In the previous section we considered the problem of semantic conflicts existing in transformation modeling. Now, we come to the typical conflicts a designer must face against.

3.1 Equi-transformation Conflict

Equi-transformation is the basic transformation with the following style:

```
Update  TableName
Set     ColumnName = Replace Value1 with Value2
Where   CharacterValueInColumnName = Value1
```

The CharacterValueInColumnName in the where-clause in Q1 maybe the column name or a positive position in the column, e.g., many primary key column is organized according to the specified length with different explanation, e.g., position n_1 to n_2 represents the country code and n_2 to n_3 represents the province code. With the replacement of column values, there may be conflicts among the replacements order.

Definition 2. *(Equi-transformation conflict) Let the result domain of ColumnName is S, the condition domain in transformation is S_1, $S_1 \subseteq S$, assignment domain is S_2, for the transformation T_1 and T_2 , we have*

1. *if $T_2.Value2 = T_1.Value2$, then T_1 is condition domain conflict with T_2.*
2. *if $T_1.Value1 = T_1.Value1$, then T_1 is assignment domain conflict with T_2.*
3. *if $T_2.Value1 = T_1.Value2$, then T_1 is loop domain conflict with T_2.*

What we mean of condition domain conflict is that if two transformation process the same object, there maybe conflict existing in the execution order of the two transformation. Different execution order may lead to different results. If we execute transformation T_1 and then execute T_2, then the last result of transformations is the results of execution of the transformation T_2; but if we execute transformation T_2 and then execute T_1, then the last result of transformations is the results of execution of the transformation T_1. An example is shown in Fig.1.

```
T1:     Update  Parts                  T2:     Update  Parts
        Set     PartNo = 100543                Set     PartNo = 200543
        Where   PartNo = 130543                Where   PartNo = 130543
```

Fig. 1. Condition domain conflict among transformations

What we mean of assign domain conflict is that if the results of two transformation process are the same object, there maybe no conflict if there are only these two transformation process the object, but there are conflicts existing in the execution order of the two transformation if there are other transformations using the resulting object. An example is shown in Fig.2.

What we mean of loop domain conflict is that a definition value of a transformation is the assignment value of another transformation. Different execution order leads to different results. An example is shown in Fig.3. If we execute transformation T_1 and then execute T_2, then the last result of transformations is the results of execution of the transformation T_2, and no result would exist "PartNo="100543""; but if we execute transformation T_2 and then execute T_1, then the last result of transformations is the results of execution of the transformation T_1, there are still records "PartNo="100543"".

```
T1:     Update  Parts                 T2:     Update  Parts
        Set     PartNo = 100543               Set     PartNo = 100543
        Where   PartNo = 130543               Where   PartNo = 190543
```

Fig. 2. Assignment domain conflict among transformations

```
T1:     Update  Parts                 T2:     Update  Parts
        Set     PartNo = 100543               Set     PartNo = 160543
        Where   PartNo = 130543               Where   PartNo = 100543
```

Fig. 3. Loop domain conflict among transformations

3.2 Containment-Transformation Conflict

Containment transformation is the transformation whose where-clause have range comparison.

```
Update  TableName
Set     ColumnName = Func(ColumnName)
Where   ColumnName in Range (Value1, Value2)
```

The ColumnName in the where-clause in Q2 maybe the column name or a positive position in the column, e.g., many primary key column is organized according to the specified length with different explanation, e.g., position n_1 to n_2 represents the country code and n_2 to n_3 represents the province code. With the replacement of column values, there may be conflicts among the replacements order.

Definition 3. *(Containment-transformation conflict) Let the result domain of ColumnName is S, the condition domain in transformation is S_1, $S_1 \subseteq S$, assignment domain is S_2, for the transformation T_1 and T_2 , we have*

1. *if $T_1.(Value1, Value2) \cap T_2.(Value1, Value2) \neq \emptyset$, then T_1 is condition domain conflict with T_2.*
2. *if $T_1.Func(ColumnName) \cap T_2.Func(ColumnName) \neq \emptyset$, then T_1 is assignment domain conflict with T_2.*
3. *if $T_1.Func(ColumnName) \cap T_2.(Value1, Value2) \neq \emptyset$, then T_1 is loop domain conflict with T_2.*

What we mean of condition domain conflict is that if two transformation process the same object, there maybe conflict existing in the execution order of the two transformation. Different execution order may lead to different results. If we execute transformation T_1 and then execute T_2, then the last result of transformations is the results of execution of the transformation T_2; but if we execute transformation T_2 and then execute T_1, then the last result of transformations is the results of execution of the transformation T_1. An example is shown in Fig.4.

What we mean of assign domain conflict is that if the results of two transformation process are the same object, there maybe no conflict if there are only

```
T1:     Update  Parts
        Set     PartNo = PartNo+100
        Where   PartNo Between 130543 and 130546

T2:     Update  Parts
        Set     PartNo = PartNo+100
        Where   PartNo Between 130543 and 130546
```

Fig. 4. Condition domain conflict among transformations

```
T1:     Update  Parts
        Set     PartNo = PartNo+100
        Where   PartNo Between 130543 and 130546

T2:     Update  Parts
        Set     PartNo = PartNo+200
        Where   PartNo Between 130420 and 130480
```

Fig. 5. Assignment domain conflict among transformations

```
T1:     Update  Parts
        Set     PartNo = PartNo+100
        Where   PartNo Between 130543 and 130546

T2:     Update  Parts
        Set     PartNo = PartNo+100
        Where   PartNo Between 130420 and 130480
```

Fig. 6. Loop domain conflict among transformations

these two transformation process the object, but there are conflicts existing in the execution order of the two transformation if there are other transformations using the resulting object. An example is shown in Fig.5.

What we mean of loop domain conflict is that a definition value of a transformation is the assignment value of another transformation. Different execution order leads to different results. An example is shown in Fig.6. If we execute transformation T_1 and then execute T_2, then the last result of transformations is the results of execution of the transformation T_2, and no result would exist “PartNo=”100543””; but if we execute transformation T_2 and then execute T_1, then the last result of transformations is the results of execution of the transformation T_1, there are still records “PartNo=”100543””.

4 Serialization Conflict Resolution Algorithms

As there are two main conflict exist in modeling, there is a need for algorithms to find and resolve such conflict. In the following subsection, we first introduce the properties of transformation, and then present algorithms for resolution of equi-transformation conflict and of containment-transformation conflict.

4.1 Transformation Algebraic Properties

Definition 4. *(Commutative transformation) Two transformations T_1 and T_2 are commutative, if various execution of T_1 and T_2 have the same result.*

For two transformation with assignment conflict, if their execution results no side-effect, then they are commutative mutually.

Definition 5. *(Noncommutative transformation) Two transformations T_1 and T_2 are noncommutative, if various execution of T_1 and T_2 have the different results. T_1 and T_2 are mutually nonexchangeable.*

Noncommutative transformations can be the case that two transformations have different results according to different execution order, also can be the case with assignment conflict. In the case of assignment conflict, if we prescribe two transformations are nonexchangeable, then optimization of transformation schedule may result different status. And it lead to another concept, there is a certain priority among transformations.

Definition 6. *(Partial order of transformations) Two transformations T_1 and T_2 are partial order, if T_1 and T_2 have the different execution order. If T_1 is executed prior to T_2, then $T_1 \prec T_2$.*

With the definition of transformation partial order, we can easily extend to transformation set partial order.

Definition 7. *(Partial order of transformation sets) Two transformation sets TS_1 and TS_2 are partial order, if TS_1 and TS_2 have the different execution order, that is $\forall T_i \in TS_1, \forall T'_j \in TS_2$, $T_i \prec T'_j$ is true, and refer to $TS_1 \prec TS_2$.*

4.2 Algorithms for Equi-transformation Conflict

Our algorithm divides the process of finding equi-transformation conflict into three phase. In the first phase, we establish transformation list, condition domain list and assignment domain list. In the second phase, we detect whether there is condition domain conflict. Thus, in the third phase we detect whether there is loop domain conflict. We do not detect the assignment domain conflict, because if there is such a conflict, it take effect either in the condition domain conflict or in the loop domain conflict.

Algorithm 1: Equi-Transformation Conflict

```
// Establish transformation list, condition domain list
// and assignment domain list
For each transformation $T_i$
  If $T_i$ is equi-transformation
  Then
    CondList.Add($T_i$.ConditionValue, $T_i$);
    SetList.Add($T_i$.SetValue, $T_i$);
    TransList.Add($T_i$);
    EstablistLink(TransList, ConList,TransList);
```

```
// Detecting whether there is condition domain conflict
For each ConditionValue in CondList
  If a condition exists multiple link to TransList
  Then
    there are conflict and should be determine partial order.

// Detecting whether there is loop domain conflict
For each ConditionValue in CondList
  If ConditionValue is in SetList
  Then
    Transformation set which ConditionValue refer to is
    conflict with the transformation set which SetValue
    refer to, and there is a need to determine the
    partial order.
```

After having detecting all these equi-transformation conflicts, the designer can determine the partial order of transformation sets and serialize the transformation model. In processing, the users must first determine the partial order between different transformation set, and then determine the partial order among condition conflict transformations, last determine the partial order among assignment conflict transformations. After this, the users know the execution order the transformation must own and have a better result model.

4.3 Algorithm for Containment-Transformation Conflict Resolution

As to containment conflict, user can use the following algorithms to detecting the conflict, the basic idea is also to establish condition domain and assignment domain list, from which one can detect various conflict, but the algorithm is different from the algorithm above in that it must be reconstruct of the condition domain list and assignment domain list.

Algorithm 2: Containment Transformation Conflict

```
// Establish transformation list, condition domain list and
// assignment domain list
For each transformation $T_i$
  If $T_i$ is containment-transformation
  Then
    CondList.Add($T_i$.ConditionRange, $T_i$);
    SetList.Add($T_i$.SetRange, $T_i$);
    TransList.Add($T_i$);
    EstablistLink(TransList, ConList, TransList);

// Reconstruct condition domain list
For all ConditionValueRange in CondList
  Reconstruct the list according to the value range so that
  there is no intersect between neighborhood node.
```

```
     Relink the transformations to the value range.

   // Reconstruct condition domain list
   For all SetValueRange in SetList
      Reconstruct the list according to the value range so that
      there is no intersect between neighborhood node.
      Relink the transformation to the value range.

   // Detecting whether there is condition domain conflict
   For each ConditionValueRange in CondList
     If a condition exists multiple link to TransList
     Then
       There are conflict and should be determine partial order.

   // Detecting whether there is loop domain conflict
   For each ConditionValue in CondList
     If $ConditionValue \cap SetValueRange_i \neq \emptyset$
     Then
       Transformation set which ConditionValue refer to is
       conflict with the transformation set which SetValue
       refer to, and there is a need to determine the partial
       order.
```

After having detecting all these containment conflicts, the designer can serialize the transformation model. In processing, the users must first remodeling the process according to the range after reconstruction, next determine the partial order between different transformation set, and then determine the partial order among condition conflict transformations, last determine the partial order among assignment conflict transformations. After this, the users know the order the transformation must obey and have a good result model.

5 Conclusions

Data cleaning is a key issue in legal system migration and data integration, and is a key process that guarantee the quality of data. Effective detecting and resolve potential problems lying in the process flow model of data cleaning is the precondition of enhancing data quality. In this paper, we studied the problem of detecting semantic serialization of data transformations. We first initiate the problem, and then present a detailed definition of the problem, and present the algebraic property of transformation, and then we present two kinds of conflicts in ETL modeling, and such process may be neglect in the modeling by most of the ETL tools, and present algorithms to detecting such problems and how to resolving it. Our approach can be used as a complement of ETL modeling tools, and also can guide the design of ETL process that enable error-free data loading.

References

1. Fu, X., Bultan, T., Su, J.: Model Checking XML Manipulating Software. In Proc. International Symposium on Software Testing and Analysis (ISSTA'04), July 2004.
2. Abiteboul, S., Cluet, S., Milo, T., Mogilevsky, P., Simeon, J., Zohar, S.: Tools for data translation and integration. IEEE Data Engineering Bulletin, 22(1):3-8, March 1999.
3. Chauhuri, S., Dayal, U.: An overview of data warehousing and OLAP technology. SIGMOD Record, 26(1):65-74, March 1997
4. Galhardas, H., Florescu, D., Shasha, D., Simon, E.: Ajax: An extensible Data Cleaning Tool. In Proc.2000 ACM SIGMOD Int. Conf. On Management of Data, Dallas, Texas,2000
5. Rahm, E., Do, H.-H.: Data Cleaning: Problems and Current Appoaches. In Bulletin of IEEE Computer Society Technical Committee on Data Engineering,2000
6. Vassiladis, P., Vagena, Z., Karayannids N., Sellis, T.: Arktos: Toward The Modeling, Design, Control and Execution of ETL Processes. In Information Systems,26(8):537-561, Elsevier Science Ltd.,2001