# Constraint Matching in Multiple Databases

Karamveer Yadav[#] and Kasi Periyasamy[§]
Department of Computer Science, University of Wisconsin, La Crosse
La Crosse, WI, 54601 USA
[#]yadav.karamve@uwlax.edu [§]kperiyasamy@uwlax.edu

## Abstract

Applications designed for shared domain often have different databases but there exists some commonality between these databases. In order to achieve coherency across these application databases, related database attributes and their respective content should be analyzed. One of the approaches to achieve this is schema matching and is widely accepted. In our previous work, we proposed a tool for schema matching using structural as well as linguistic matching. The present work further extends this towards content matching. We propose constraint matching not only for standard database constraints but also for application domain specific custom constraints. We came across set of validation rules for the constraint composition as well as constraint matching. This paper explains the procedure we adopted for constraint matching and the rationale for defining logical rules for constraint matching.

## 1 Introduction

When several complex data-oriented applications use similar data but with different databases, the common elements between these databases are expected to be in the same format in order to provide a seamless transfer of data between these databases. Schema matching is a process that assesses the schemas of these databases, and projects the commonalties and differences between the various schemas involved. The schema matching process involves matching the attributes of the schemas for their names and data types at the minimal level. A preliminary work on schema matching by the authors has been indicated in [4] (proceedings). One major application of the schema matching process is its use during integration of several components of a large complex system, each component using a separate database.

During application integration, when more than one databases are integrated into a single database, data redundancies and inconsistencies are often encountered. To resolve this, a critical comparison of databases is required so that successful integration can be achieved through efficient schema matching. Once schema matching is achieved, contents of databases under investigation should be matched for portability of data across set of applications. So the concept of content matching is utilized in this context. But with multitude of information and in large scale systems working in distributed environment, it is not feasible to compare all the entries in all tables and schemas across several databases. Moreover, content across various tables and columns inside them need not be identical for acceptance by potential users. In such a scenario, contents of the databases are best described by the constraints, if any, applied to the values before they are stored. For example, a variable representing a room in a hotel may have a constraint that limits the room number between 100 to 180. These constraints not only the traditional database constraints such as NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK, and DEFAULT that are predefined in most of SQL based databases but also include application domain based constraints on data in tables and columns. These custom constraints are defined according to application domain requirements and their values signify specific limitation on data. So we explored the automation of content matching by utilizing the detailed implementation of constraint matching.

In the presented study, we have extended our previous work on schema matching. Previously, we achieved schema matching by utilizing schemas of databases under comparison [4] (proceedings). In this process, we utilized an adaptive knowledge base and explored matching on several aspects including linguistic matching, structural matching, and data type matching. As a result of this process a final schema is proposed on the basis of some computed qualifying score. In the presented study, we investigated content matching of two databases. Besides SQL constraints, considering the design and implementation perspective, we explored six domain level constraints. These application domain specific constraints are, Less Then (LT), Greater Then (GT), Less Then Equal (LE), Greater Then Equal (GE), Equal (EQ) and Not Equal (NE). These constraints as key-value pair across corresponding columns for two databases are compared and a final set of constraints is proposed so that during data import, portability and integrity issues can be resolved. The proposed constraint set will be within the range of constraints of databases under comparison so that if data is imported into a new database it will satisfy the limitations set by all the databases under comparison. The rest of the sections in this paper explain the process in detail.

978-1-943436-01-9 / copyright ISCA, SEDE 2015
October 12-14, 2015, San Diego, California, USA

## 1.2 Literature Review

Automation of schema matching has been extensively discussed and its importance has been stressed with rise in demand for enterprise application [5,7] (journal). We previously adopted a hybrid approach by using linguistic matching and structural matching [4] (proceedings). Valtchev and his colleagues explored constraint matching by using data types as constraints for automation of taxonomies in 1999. This work was based on their previous work on clustering objects into classes by automating object matching in which they investigated dissimilarity measures for collection of objects and values in 1997 [9] (journal). In 2001, constraint based schema matching was explored by E. Rahm and colleagues [5] (journal). This approach explored data types and multiplicity comparison for set of values by considering the cardinality of attributes as a constraint. Similar approach was used by J. Euzenat and others in 2004 [2] (proceedings). Most published literature considers the standard SQL constraints for constraint based matching [1,8] (journal), [3,6] (report). We utilize the knowledge of application domain specific constraints for this analysis since domain specific constraints highlight the real boundaries of content. Moreover, in large scale systems it is not feasible to compare every single entry in the databases under consideration. If custom constraints are available, then it is legitimate to analyze the attribute (content) boundaries to explore content export as well as matching. This will be efficient in terms of total time spent for content comparison and computationally economical too.

# 2 Approach

The implementation of this constraint based content matching adopts a modular approach. This is elaborated in Figure 1. Constraint Mapper maps the two input constraint sets into the corresponding database schemas under comparison. Once this mapping is successfully done, Constraint Validator is utilized to validate the constraints provided as input.

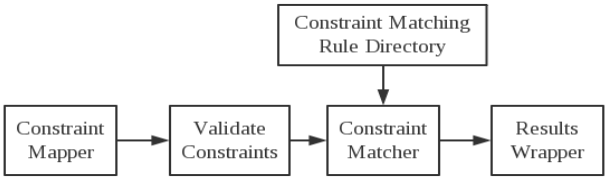


Figure 1: Modular Structure for implementation of constraint matching

If validation is successful, constraints are analyzed, <key, value> pairs of constraint sets are examined thoroughly and final set of constraints is proposed if possible. This processing is accomplished in Constraint Matcher module. Constraint Matching Rule Directory is at the core of this module. Finally, in the Result Wrapper the proposed set of constraints as well as proposed schema are provided to the user as outcome.

## 2.1 Methodology and Workflow

The detailed workflow is presented in Figure 2. We attempted to design this process flow diagram as much self explanatory as possible. The content matching process is applied to only those schemas that are reasonably similar. These two schemas are selected as inputs by the schema matching process (Kasi, 2014) in which the matching score is higher between the two schemas. Most importantly, the two schemas must have a large base of attributes matched for names and data types. Then user is then prompted to provide two set of constraints for the input schemas. The current system accepts these inputs as XML files. Xml based constraint input file contains the application domain specific constraints for all the attributes of corresponding input schema.

The content of XML based input file (partial content) looks something like:

```
<Item tableName="Bill" >
<Attribute ColumnName="PrimaryBill" DataType="int"
LessThen="9000"GreaterThen="1"  NotEqual="5000"  />

<Attribute ColumnName="MinnimumPayable"
DataType="int" LessThen="10000" GreaterThen="999"/>

<Attribute ColumnName="Rating" DataType="int"
LessThenEqual="100"  GreaterThen="4"/>

<Attribute ColumnName="BillingCode" DataType="int"
LessThen="5001" GreaterThen="0" NotEqual="1000"/>

</Item>
```

Present implementation supports six application domain specific constraints i.e. Less Then (LT), Greater Then (GT), Less Then Equal (LE), Greater Then Equal (GE), Equal (EQ) and Not Equal (NE). This enumeration of constraints provide best possible coverage of content i.e. data inside table columns. User is prompted for selecting columns in proposed schema for constraint based content analysis. Consequently, depending on user selection, corresponding entries from the input schema and their constraint files are selected for further processing.

In the next step, these selected constraints are validated according to validation rules defined in validation rule directory. User can edit the rule directory to add, edit or delete a validation rule keeping in consideration the requirements of application domain. The possible set of constraints for a specific column can be any one of constraint set provided in Table 1. The option trees shown

in Figure 3 are utilized to extract feasible constraint sets as shown in Table 1.

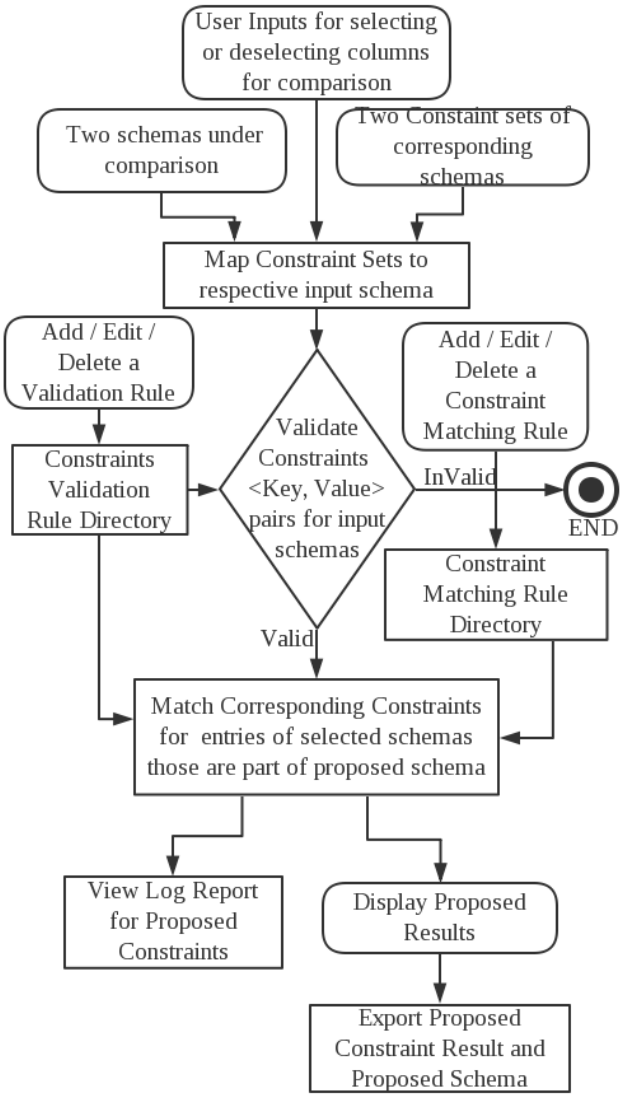


Figure 2: The detailed workflow of content matching utilizing constraint matching

The rules of validation of values within the set of constraints within a column entry are shown in Table 2. As mentioned, the schema validation rules are editable by user according to domain requirements. If the schema constraints are invalid according to validation rules, the process terminates and user is informed appropriately. If schema constraints pass validation test then process of constraint matching is initiated.

For each selected column, a one to one comparison is carried out for set of constraints corresponding to entries in both databases. The <key, value> pairs in the constraint sets are thoroughly analyzed and a resulting constraint set is proposed. There is an exhaustive set of rules for constraint matching.

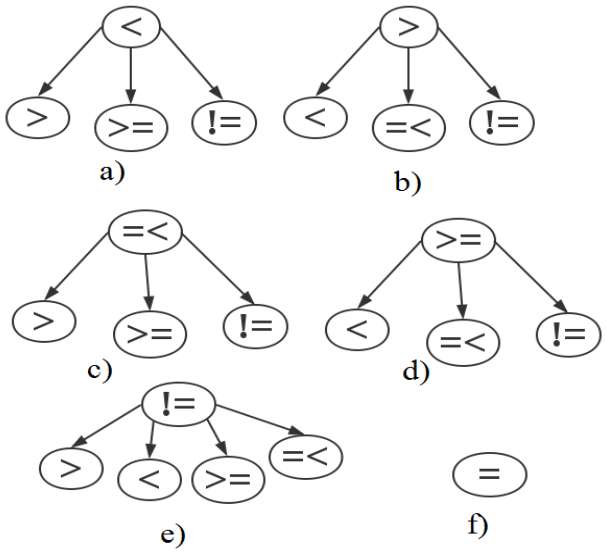


Figure 3: The set of Option tree for navigating and validating constraints for a column entry i.e. attribute.

Table 1: Constraint Set Combinations

| Sr. No. | Feasible Constraint Set |
|---|---|
| 1 | LT |
| 2 | LT GT |
| 3 | LT GT NE |
| 4 | LT GE |
| 5 | LT GE NE |
| 6 | LE |
| 7 | LE GT |
| 8 | LE GT NE |
| 9 | LE GE |
| 10 | LE GE NE |
| 11 | GT |
| 12 | GE |
| 13 | NE |
| 14 | LT NE |
| 15 | LE NE |
| 16 | GT NE |
| 17 | GE NE |

Table 2: Validation Rules for constraints of an attribute.

| Sr. No. | Logical Rules for Values of Constraints ( "(V)" represents the value of specified constraint ). |
|---|---|
| 1 | LT(V) > GT(V) |

| | |
|---|---|
| 2 | LT(V) > GE(V) |
| 3 | LE(V) > GT(V) |
| 4 | LE(V) > GE(V) |
| 5 | LT(V) > NE(V) |
| 6 | LE(V) ≥ NE(V) |
| 7 | GT(V) < NE(V) |
| 8 | GE(V) ≤ NE(V) |

As shown in Table 1, constraint set for a column may contain 1 to 3 constraints. So for comparing two constraint sets, there can be numerous types of comparisons depending on the number of elements in each set. For instance, an entry

*<Attribute ColumnName="Rating" DataType="int" LessThenEqual="100" GreaterThen="4"/>*

in first constraint set to be compared to corresponding entry i.e.

*<Attribute ColumnName="Score" DataType="int" LessThenEqual="50" GreaterThenEqual="1"/>*

from another schema has 2:2 comparison since LessTenEqual and GreaterThen (2 constraints for first schema) are compared with LessThenEqual and GreaterThenEqual of second schema. So the system requires set of rules for 1:1, 1:2, 1:3, 2:2, 2:3 and 3:3 constraint matching.

As an illustration, when both selected constraint sets contain only one constraint (one-to-one), the rules for comparison of constraints and their respective values are shown in Table 3.

Table 3: Rules for Constraint Matching (in presented case both schemas have one constraint i.e. 1:1 constraint matching)

| Sr. No. | First Schema Constraint | Second Schema Constraint | Predicate for values comparison | Proposed Constraint |
|---|---|---|---|---|
| 1 | LT(x) | GT(y) | x = y | No Suggestion |
| 2 | LT(x) | GT(y) | x < y | No Suggestion |
| 3 | LT(x) | GT(y) | x > y | LT(x) GT(y) |
| 4 | LT(x) | GE(y) | x = y | No Suggestion |
| 5 | LT(x) | GE(y) | x < y | No Suggestion |
| 6 | LT(x) | GE(y) | x > y | LT(x) GE(y) |
| 7 | LT(x) | NE(y) | x = y | LT(x) |
| 8 | LT(x) | NE(y) | x < y | LT(x) |
| 9 | LT(x) | NE(y) | x > y | LT(x) NE(y) |
| 10 | LT(x) | ET(y) | x = y | No Suggestion |
| 11 | LT(x) | ET(y) | x < y | No Suggestion |
| 12 | LT(x) | ET(y) | x > y | ET(y) |
| 13 | LT(x) | LT(y) | x = y | LT(x) |
| 14 | LT(x) | LT(y) | x < y | LT(x) |
| 15 | LT(x) | LT(y) | x > y | LT(y) |
| 16 | LT(x) | LE(y) | x = y | LT(x) |
| 17 | LT(x) | LEy) | x < y | LT(x) |
| 18 | LT(x) | LE(y) | x > y | LE(y) |
| 19 | LE(x) | GT(y) | x = y | No Suggestion |
| 20 | LE(x) | GT(y) | x < y | No Suggestion |
| 21 | LE(x) | GT(y) | x > y | LE(x) GT(y) |
| 22 | LE(x) | GE(y) | x = y | ET(x) |
| 23 | LE(x) | GE(y) | x < y | No Suggestion |
| 24 | LE(x) | GE(y) | x > y | LE(x) GE(y) |
| 25 | LE(x) | NE(y) | x = y | LE(x) |
| 26 | LE(x) | NE(y) | x < y | LE(x) |
| 27 | LE(x) | NE(y) | x > y | LE(x) NE(y) |
| 28 | LE(x) | ET(y) | x = y | ET(y) |
| 29 | LE(x) | ET(y) | x < y | No Suggestion |
| 30 | LE(x) | ET(y) | x > y | ET(y) |
| 31 | LE(x) | LE(y) | x = y | LE(x) |
| 32 | LE(x) | LE(y) | x < y | LE(x) |
| 33 | LE(x) | LE(y) | x > y | LE(y) |
| 34 | GT(x) | GT(y) | x = y | GT(x) |
| 35 | GT(x) | GT(y) | x < y | GT(y) |
| 36 | GT(x) | GT(y) | x > y | GT(x) |
| 37 | GT(x) | GE(y) | x = y | GT(x) |
| 38 | GT(x) | GE(y) | x < y | GE(y) |
| 39 | GT(x) | GE(y) | x > y | GT(x) |
| 40 | GT(x) | NE(y) | x = y | GT(x) |
| 41 | GT(x) | NE(y) | x < y | GT(x) NE(y) |
| 42 | GT(x) | NE(y) | x > y | GT(x) |

| | | | | |
|---|---|---|---|---|
| 43 | GT(x) | ET(y) | x = y | No Suggestion |
| 44 | GT(x) | ET(y) | x < y | ET(y) |
| 45 | GT(x) | ET(y) | x > y | No Suggestion |
| 46 | GE(x) | GE(y) | x = y | GE(x) |
| 47 | GE(x) | GE(y) | x < y | GE(y) |
| 48 | GE(x) | GE(y) | x > y | GE(x) |
| 49 | GE(x) | NE(y) | x = y | GT(x) |
| 50 | GE(x) | NE(y) | x < y | GE(x) NE(y) |
| 51 | GE(x) | NE(y) | x > y | GE(x) |
| 52 | GE(x) | ET(y) | x = y | ET(y) |
| 53 | GE(x) | ET(y) | x < y | ET(y) |
| 54 | GE(x) | ET(y) | x > y | No Suggestion |
| 55 | ET(x) | NE(y) | x = y | No Suggestion |
| 56 | ET(x) | NE(y) | x < y | ET(x) |
| 57 | ET(x) | NE(y) | x > y | ET(x) |
| 58 | ET(x) | ET(y) | x = y | ET(x) |
| 59 | ET(x) | ET(y) | x < y | No Suggestion |
| 60 | ET(x) | ET(y) | x > y | No Suggestion |
| 61 | NE(x) | NE(y) | x = y | NE(x) |
| 62 | NE(x) | NE(y) | x < y | NE(x) NE(y) |
| 63 | NE(x) | NE(y) | x > y | NE(x) NE(y) |

Following the trend as shown in Table 3, we framed rules for all the mapping including one-to-two, one-to-three, two-to-two, two-to-three and three-to-three. The set of rules is exhaustive, and so keeping consideration of space limit only partial set of rules is presented here.

User is allowed to add, edit or delete the rules in the constraint matching rule directory according to specific requirements of application domain under investigation. Constraint making rules consider the constraints as well as corresponding values to determine the proposed constraints. This initially involves considering every constraint as constraint name and corresponding value as <key, value> pair. Then all the <key, value> pairs for a column entry are used to constitute a set of <key, value> pairs. This process of parsing the constraints is repeated for all the selected columns of both input schemas. Once parsing of constraint information into sets is accomplished, the rules that best fit the two sets (for a single column selected from proposed schema) are applied to generate the corresponding proposed constraint set. This process is repeated iteratively for all the attributes, i.e. columns of proposed schema. In addition, this process also ensures that all of the proposed constraint sets follow the constraint validation rules as shown in Table 1 and 2. On completion of this, resulting constraint sets are displayed and user is prompted to accept or edit the constraints subject to the limitations provided in constraint matching rules dictionary. Once user accepts this information, proposed constraint set along with proposed schema can be exported.

## 2.2 Sample Rule Set Explanation

In this section, we attempt to explain the constraint matching with details. For instance, entry 4 in Table 3 in first schema has a LT constraint for some of its column (attribute) with value x while second schema has a GT constraint with value y. Suppose these attributes have an integer data type. In the current scenario i.e. x > y, the proposed constraint set will be { LT(x), GT(y) }. The explanation of this rule can be tested with simple arithmetic and logical operations on the range of integer data type. Figure 4 shows a pictorial explanation of the resulting schema constraints.

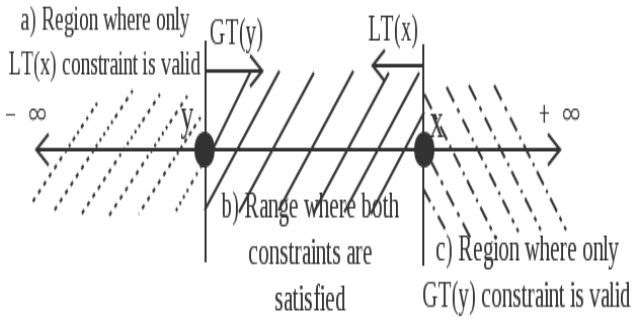


Figure 4: Logical Explanation for one of the one-to-one rule (for <LT(x)> and <GT(y)> constraint comparison for x > y).

The region b as shown in Figure 4 provides best possible set of constraints and corresponding values so that the proposed constraints satisfy the limitations prescribed by both the schemas. The rationale behind this selection is that if data from both the databases is imported into a new schema utilizing the proposed constraints set then the imported data is valid for both the databases under comparison. In theoretical terms, on the basis of set theory the resulting constraint <key, value> pairs are the result of intersection of two input schemas under investigation. Also, if the user decides to automate the data import by incorporating the proposed constraint sets then the imported content maintains the data integrity across applications under consideration. An example of "No Suggestion" is also demonstrated in Figure 5.

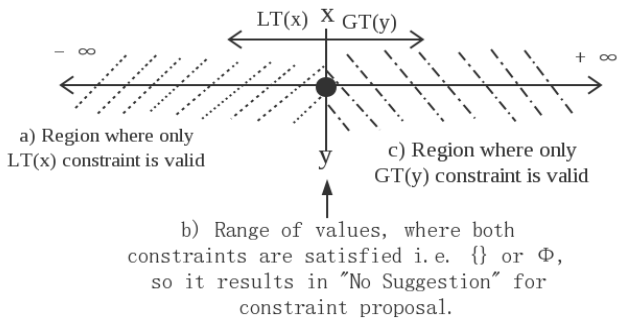


Figure 5: Logical Explanation for one of the one-to-one rule for <LT(x)> and <GT(y)> constraint comparison if x = y as shown at # 1 in Table 1.

### 2.3 Proposal for Broader Perspective

The rules and the explanation provided in preceding section considers content import only in a very narrow perspective. Some application domains may require or allow user to import content from different databases with fewer restrictions on values. In order to incorporate this, suggestions for broadening the range of values for constraints is provided in the log report. For instance, consider a case where table columns under comparison on both sides have two constraints as { LT(x1), GT(y1) } and { LT(x2), GT(y2) }, with conditions y1 > y2 and x1 > x2.

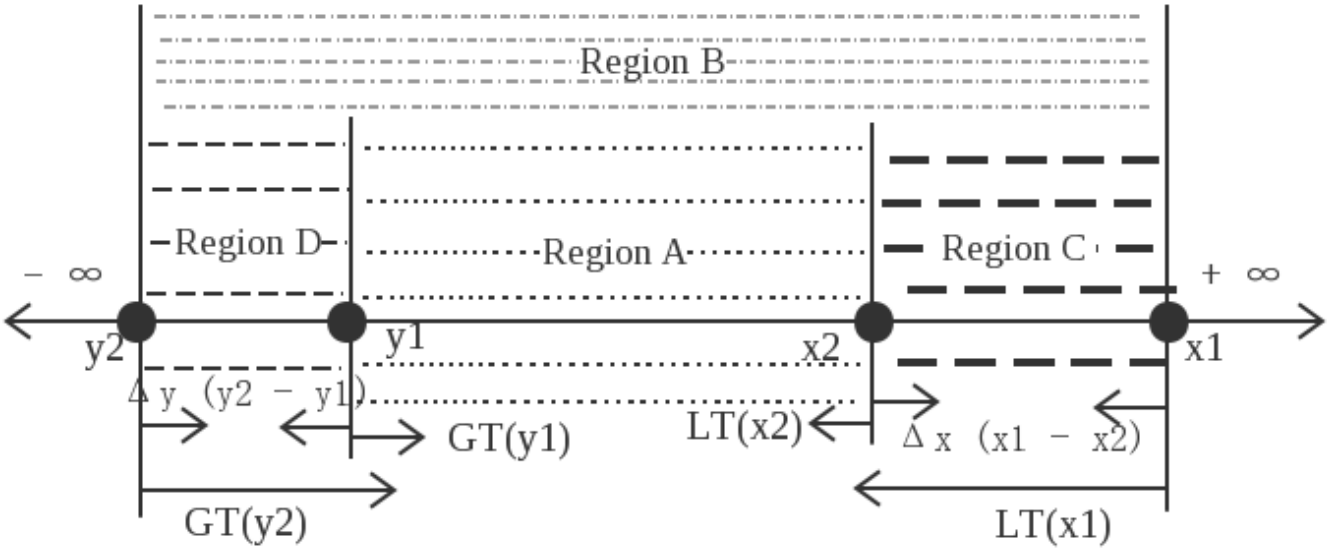


Figure 6: Logical Explanation for broadening the content range for the data import in proposed schema.

In Figure 6, Region A denotes proposed constraint set { LT(x2), GT(y1) } according to the rule and it shows results on narrower perspective. On consideration of broader perspective, we suggest the user through log report that if he wishes to extend the lower limit of content then GT(y1) should be replaced with ( GT(y1) + Δy) as shown in Region D. Similarly, for upper limit LT(x2) should be replaced with ( LT(x2) + Δx) as shown in Region C. If user accepts both the proposals then the final constraint should be { (GT(y1) + Δy), (LT(x2) + Δx) } and the range of content values covered herein is apparent in region B in Figure 6. If the prospective application domain requirement is to incorporate entire content from corresponding columns of both databases then user should adopt this constraint set.

## 3 Conclusion

Significance of schema matching is inevitable for the purpose of content matching and data import in modern large scale enterprise applications. Applications within domain specific custom constraints are important to maintain data integrity and manageability. But employing the custom constraints acts as an overhead for schema matching and data import. In addition, to certain extent it adds complications for the database matching. Hence custom constraint matching is mandatory for a thorough database comparison. The presented approach considers a rule based approach for constraint matching and provides user with flexibility of adjusting the rules set according to the requirements of application domain. In addition, validation of input constraint sets and output proposed constraints is important to test validity of the entire process.

The current implementation demonstrates the constraint matching for constraints if data type is primitive, for instance the presented examples show implementation for integer type. This work can be extended to implement other data types including complex object types too. Consequently, the comparisons may take more time while processing but the methodology remains the same. The advent of inheritance and behavioral implementations (like interfaces in Java) have made it feasible and far less tedious to extend the implementation for custom complex data types.

Currently, the authors have not performed any analysis on the complexity of the algorithms described in this work but it is certainly worth considering. In particular, when complex data types are used for comparison, complexity analysis will indeed be required to improve the performance of the algorithms. However, the authors believe that data type comparisons have been used in several applications which do not seem to be a problem with existing technology and resources. Moreover, from practical view point, the database schemas involved in the constraint matching process will not be too large even though the data stored in these databases might be larger.

# References


[1] E.V. Wyk, D. Bodin, J. Gao and L. Krishnan, Silver: an extensible attribute grammar system, *Science of Computer Programming*, 75:39-54, 2010.

[2] J. Euzenat and P. Valchey, Similarity-based ontology alignment in OWL-lite. In *Proceedings of the European Conference on Artificial Intelligence (ECA)*, pages 333-337, Valencia, Spain Aug 22-27, 2004.

[3] J. Madhavan, P.A. Bernstein and E. Rahm, Generic Schema Matching with Cupid, *Technical Report MSR-TR-2001-58*, Microsoft Research, Microsoft Corporation, Redmond, WA 2001.

[4] K. Periyasamy and P. Papanna, A Tool for Schema Matching with a Knowledge Base, In *Proceedings of the International Conference on Software Engineering and Data Engineering*, New Orleans, LO, Oct 13-15, 2014.

[5] E. Rahm and P.A. Bernstein, A Survey of Approaches to Automated Schema Matching, *The VLDB Journal*, 10, Springer-Verlag, 334-350, 2001.

[6] M. Sergey, H. Garcia-Molina and E. Rahm, Similarity flooding: A versatile graph matching algorithm (*extended technical report*), 2001.

[7] X.L. Sun and E. Rose. Automated schema matching techniques: An exploratory study, *Research Letters in Information and Mathematical Sciences*, 4:113-136, 2003.

[8] H.O. Thang and Vo Sy Nam. XML schema automatic matching solution, *International Journal of Electrical, Computer and Systems Engineering*, 4(1): 68-74, 2010.

[9] P. Valchey and J. Euzenat. Dissimilarity measures for collection of objects and values, *Lecture Notes in Computer Science*, 1280: 259-272, 1997.