The Schema Coercion Problem

Terence Critchlow and Gary Lindstrom

UUCS-97-002

Department of Computer Science
University of Utah
Salt Lake City, UT 84112 USA

February, 1997

The Schema Coercion Problem

Terence Critchlow
Gary Lindstrom

Abstract

Over the past decade, the ability to incorporate data from a wide variety of sources has become increasingly important to database users. To meet this need, significant effort has been expended in automatic database schema manipulation. However, to date this effort has focused on two aspects of this problem: schema integration and schema evolution. Schema integration results in a unified view of several databases, while schema evolution enhances an existing database design to represent additional information. This work defines and addresses a third problem, schema coercion, which defines a mapping from one database to another.

This paper presents an overview of the problems associated with schema coercion and how they correspond to the problems encountered by schema integration and schema evolution. In addition, our approach to this problem is outlined. The feasibility of this approach is demonstrated by a tool which reduces the human interaction required at all steps in the integration process. The database schemata are automatically read and converted into corresponding ER representations. Then, a correspondence identification heuristic is used to identify similar concepts, and create mappings between them. Finally, a program is generated to perform the data transfer. This tool has successfully been used to coerce the Haemophilus and Methanococcus genomes from the Genbank ASN.1 database to the Utah Center for Human Genome Research database.

Our comprehensive approach to addressing the schema coercion problem has proven extremely valuable in reducing the interaction required to define coercions, particularly when the heuristics are unsuccessful.

Introduction

As information becomes increasingly available from a vast variety of sources, the need to develop uniform views and evolve existing representations becomes vitally important. Unfortunately, information is stored in a wide variety of formats ranging from web pages to flat file representations to object-oriented databases. Because of the number and complexity of these representations, there has been significant interest in developing processes to automatically define mappings between databases. To date this effort has concentrated on two specific problems: schema integration and schema evolution.

Different expressions of similar concepts constitutes the central theoretical problem associated with automating schema manipulation. These expressions are a result of the environment in which the database was developed, and manifest as differing database representations. Recognizing correspondences between these representations requires resolving the disparities (*conflicts* [20]) across these representations.

Three categories of conflicts are associated with schema manipulation: naming, structural, and semantic. Naming conflicts occur when similar concepts are given different names, or when different concepts are given the same name. Problems arise because the name is assumed, by default, to encapsulate the semantics of the abstract concept since names often constitute the only, albeit weak, meta-information available concerning the database design. As a result of these conflicts, correct correspondences may be missed, and false correspondences may be identified. Structural conflicts occur because different expressions may incorporate different details about a concept. For example, in an employee database, the concept of marriage may be a boolean attribute representing whether or not an employee is married. However, in a vital statistics database the concept of marriage may be associated with an entity having attributes such as the date of marriage, the marriage license number, and the social security numbers of the husband and wife. In such conflicts, required information may not be available in the source database and defaults may have to be provided. Semantic conflicts occur because the environment in which the database was developed may place implicit contextual interpretations on the data. For example, product prices are usually expressed in the currency of the country in which the database resides. As a result of this conflict, simple mappings between corresponding constructs may be incorrect across international borders. Thus a database representing prices in Canadian dollars and another representing prices in US dollars should not be merged without translating the prices to a common currency.

Schema integration is the process of deriving a global database schema from an initial collection of schemata. This consensus schema, which is unknown at the start of the process, must be capable of representing all of the information defined in the original schemata. To bound the complexity of this problem, integration is usually performed pairwise, with the resulting schema being used as input to a later integration step. The major challenges to successfully automating this process are identifying corresponding concepts represented in both schemata, and resolving the structural and semantic differences between these concepts. The primary motivation behind work in this area is the popularity of federated databases. The schema associated with a federation is the global schema obtained by integrating the schemata of the participating databases. Therefore, most solutions addressing this problem assume the initial schemata are represented by dissimilar database management systems, although usually only relational and object-oriented databases are considered.

Schema evolution addresses the problem of a single database's changing representation. This problem requires data to be transferred from an original schema into a new schema, and existing applications to be modified to accommodate the new schema.

Usually this problem is addressed within the scope of a single database management system, with the only difference between the old and new schemata being structural. Work in this area is motivated by practical demands for databases to adapt to new and unanticipated applications. While these demands underscore the database vitality, failure to adapt to them quickly enough will result in the database becoming obsolete and atrophied. The major challenges to successfully automating this process are determining when and how data should be transferred to the new representation, and how to minimize the effect of the new representation on existing applications.

In contrast to other research, this work defines and addresses a third problem: *schema coercion.* Schema coercion is the process of transferring data from one database -- the source -- to another database -- the target. This term reflects the recasting of concepts represented in the source database into a corresponding target database representation. The theoretical challenges associated with this manipulation are a combination of those encountered by schema integration and schema evolution: identifying correspondences, resolving conflicts, and transferring data.

The practical impact of automating schema coercion is potentially much broader than for either schema integration or schema evolution, because this problem is pervasive in database manipulation, and must regularly be addressed within the context of practical application development. For example, this problem is encountered within schema evolution with the old representation acting as the source, and the new representation acting as the target. In addition, this problem arises with federated databases when data from the local databases is transferred to the federated representation. Finally, this problem frequently appears in scientific application domains, such as genetics, which rely on data sharing among independent organizations. Genetics research labs store local data in representations varying from flat files to object-oriented databases. Lab databases must frequently share data with large community databases. Because of the different representations used by lab and community databases, transferring data between them is often tedious and at times extremely complicated and non-intuitive. Indeed, the Utah Center for Human Genome Research's need to share information with other databases formed primary motivation of this work.

The next section summarizes previous work performed in the areas of schema integration and schema evolution. The approach taken by this work is then discussed and the application of this approach on coercions defined within the genetics domain is presented. Finally, contributions of this work are summarized, along with its implications and future research directions.

Previous work

Batini [2] decomposed schema integration into four steps:

1. preintegration: converting schemata into a uniform representation
2. comparison: identifying corresponding concepts
3. conformation: resolving conflicts between corresponding concepts
4. merging: unifying the schemata to produce the global schema

Most work in schema integration has attempted to address a single kind of conflict, and thus has focused on either the comparison or the conformation steps. Naming conflicts are usually resolved by associating additional information with the schemata. This information may provide additional semantic information, for example by using a thesaurus [4], or may use specialized knowledge to infer correspondences, for example by using an expert system [6 8 9]. Resolving semantic conflicts requires assumptions implicit in the database development environment to be explicitly represented. The most common approach [11 21] is to enhance the database schema so that meta-information is represented in the database along with the associated data. For example, this is the approach taken by Sciore [19]. Unfortunately, this limits the approach to those database management systems which use a specific data model. In addition, this approach cannot be used on non-traditional data representations, such as ASN.1, which do not allow dynamic schema modification. Structural conflicts are resolved by representing the final concept using the least general representation that accommodates all of the data associated with the corresponding constructs [5]. While it is relatively easy to resolve simple conflicts, such as between an attribute and an entity, resolving complex conflicts, such as between a relationship and a relationship-entity-relationship structure, is significantly more complicated. This problem is partially addressed by Spaccappietra [22]. While significant work has been done in this area, resolving conflicts and developing a general purpose tool to automate the integration process remains an active research area.

Most schema evolution work has focused on reducing the manual modifications required to enable existing applications to access data via the new schema. To this end, most approaches [3 15 16] require the new schema to be defined as a sequence of transformations applied to the old schema. By defining a fixed set of transformations, these approaches are able to automatically generate methods to dynamically convert existing data into the new format. This allows existing applications to access old data without modification, while applications using the new representation have access to both the old and new data. Inverse conversions may also be defined to transfer new data to the old representation for use by the original applications. Because the overhead required to transfer data between representations is significant, particularly when there are several intermediate representations, other approaches perform the data transfer once, either eagerly [10] or lazily [18], and require applications to recognize the new representation. As long as a restricted set of operations are used to define the schema reorganization, the data transfer may be automatically performed. However, by restricting the permitted transformations, the set of schemata evolvable from a source schema is also restricted. Recent work [23] attempts to provide complete schema restructuring capability while still reducing the effort required to perform the data transfer. In addition, there has been little success in developing techniques to reduce the interaction required to update existing applications using these techniques.

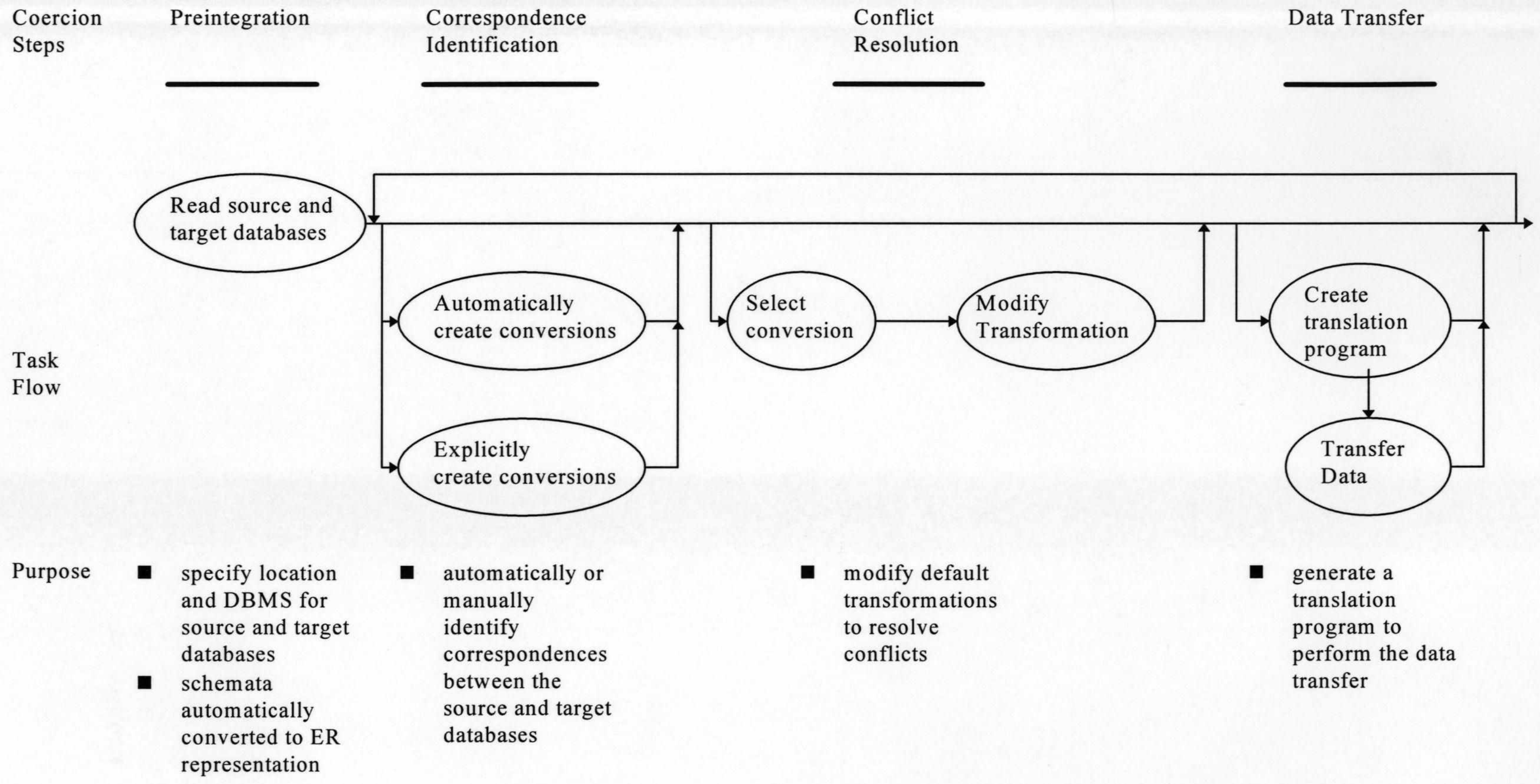

Coercion Steps
Preintegration
Correspondence Identification
Conflict Resolution
Data Transfer
Task Flow
Read source and target databases
Automatically create conversions
Explicitly create conversions
Select conversion
Modify Transformation
Create translation program
Transfer Data
Purpose
specify location and DBMS for source and target databases
schemata automatically converted to ER representation
automatically or manually identify correspondences between the source and target databases
modify default transformations to resolve conflicts
generate a translation program to perform the data transfer

Figure 1 Schema Coercion Tool Utilization

Schema Coercion

The steps involved in schema coercion are very similar to those outlined by Batini for schema integration. The first three steps are identical; however, the final step transfers data from the source database to the target instead of merging the resulting schemata. The work we report here constitutes a comprehensive approach to schema coercion. It encompasses all steps in the process within a uniform environment, and attempts to reduce the interaction required at each step. The feasibility of the concepts presented in this paper is demonstrated by a tool developed and validated within the Utah Center for Human Genome Research.

Our approach addresses the schema coercion problem iteratively, as shown by Figure 1. The preintegration step requires the user to specify the location and data representation format for the source and target databases. Once the schemata have been read by the tool, correspondences between the constructs can be identified by using an automatic matching algorithm or by explicitly identifying them. Simple conflicts are resolved automatically, while more complicated correspondences require the user to define an appropriate mapping. Finally, a translation program, which can be used to perform the data transfer, is generated based on these mappings. The remainder of this section describes this approach in more detail. The results of using our tool are discussed in the next section.

The first step in the coercion process -- preintegration -- is addressed by automatically reading database schemata directly from database management systems and converting them into Entity-Relationship representations. Because DBMSs use idiosyncratic meta-data representations, automatically reading a schema requires explicit knowledge about the associated DBMS. Obviously, it is impractical for a single tool to have knowledge about all available DBMSs. However, in order to demonstrate the versatility of this tool, several different DBMSs are supported and knowledge about additional systems may be added as required. Currently, Sybase and the Utah Genome [17] database formats are the only relational databases recognized. In addition to these databases, two non-traditional data representation formats are also supported: ASN.1 binary data files, and flat files represented in the GBD format or as relational table dumps. These representations comprise a collection of data files as well as a definition file. The definition file provides missing meta-information about the structure and location of the data files, in addition to the data types associated with these files. Object-oriented databases are not currently supported, but we believe they may added without inordinate effort.

A translation similar to the one defined by Abu-Hamdeh [1], between the relational model and the ER model, is used to create the ER representations of Sybase and flat file databases. This translation uses primary and foreign key attributes to identify entity sets and relationships; however, it required minor modifications to correctly translate Utah Genome and ASN.1 databases. The two modifications associated with the Utah databases result from the meta-data representation of this model. First, this representation explicitly defines the ER representation of each table, simplifying the mapping. Second, the Utah Genome model defines sets as relationships which have only one role. To represent these sets in the ER model, an entity corresponding to the set is created and associated with the relationship. For example, a set of *wells* may be created by defining a single relationship, *well-set*, where each instance of this relationship corresponds to a distinct set. To map *well-set* to an ER representation, a new entity, *sets-of-wells*, representing the concept of this set is created, and *wells* are related to it throught the *well-set* relationship.

The modifications associated with the ASN.1 databases result from the complex class definitions allowed by the language. In effect, ASN.1 classes are similar in structural complexity to objects. The chosen translation simply represents base classes as strong entities, primitive attributes as attributes, and complex attributes as weak entities. A better translation would differentiate between optional, mandatory, and list attributes. This could be accomplished by creating relationships and defining cardinalities appropriately: optional attributes are 0:1; mandatory attributes are 1:1; list attributes are 1:n. Accurately representing choice attributes would require defining a generalization of the choices, and defining a 1:1 relationship between that generalization and the enclosing entity. This alternative was not pursued because of the additional complexity of the ER diagram obtained is beyond the requirements of our tool. This translation may be implemented in the future if required.

The second step of the coercion process -- identifying corresponding concepts between the source and target databases -- uses the ER representations of the participating databases. To identify possible correspondences, each target entity and relationship is compared against each source entity and relationship to determine a confidence in the correspondence between the represented concepts. This confidence is the larger of the confidence in either a primitive correspondence or a complex correspondence. The confidence in a primitive correspondence is based on a variety of factors including the similarity between the construct names, whether the construct types are the same, and the highest confidence mappings from source attributes onto target attributes. The confidence in attribute mappings is primarily based on the attributes' name similarity and data type compatibility. However, other features, such as the defined keys, if any, and the data size of the attributes, are also considered. If confidence construct correspondence is sufficiently high, a conversion is created to formalize this correspondence.

Confidence in a complex correspondence is determined using a combination of alternative construct representations and existing conversions. If existing conversions fulfill the prerequisite correspondences required by an alternative representation, their confidences are used to determine the confidence in the resulting complex correspondences. This resulting confidence will always be lower than the confidences in the associated conversions, but may be higher than the confidence in the primitive correspondence.

For example, consider the coercion presented in Figure 2. The source database, *order_2*, consists of three constructs: two entities, *customer* and *product*, and a single relationship, *ordered*, associating customers with the products they have bought. The target database, *order_1*, has corresponding *customer* and *product* entities, but the *ordered* relationship is replaced by the *placedBy-invoice-line* structure. The primitive correspondence identification algorithm is only capable of identifying the correspondences between the *customer* and *product* entities. These conversions are used in conjunction with the knowledge that relationships may be decomposed into relationship-entity-relationship structures to identify the remaining correspondences between *ordered* and *placedBy*, *invoice*, and *line*.

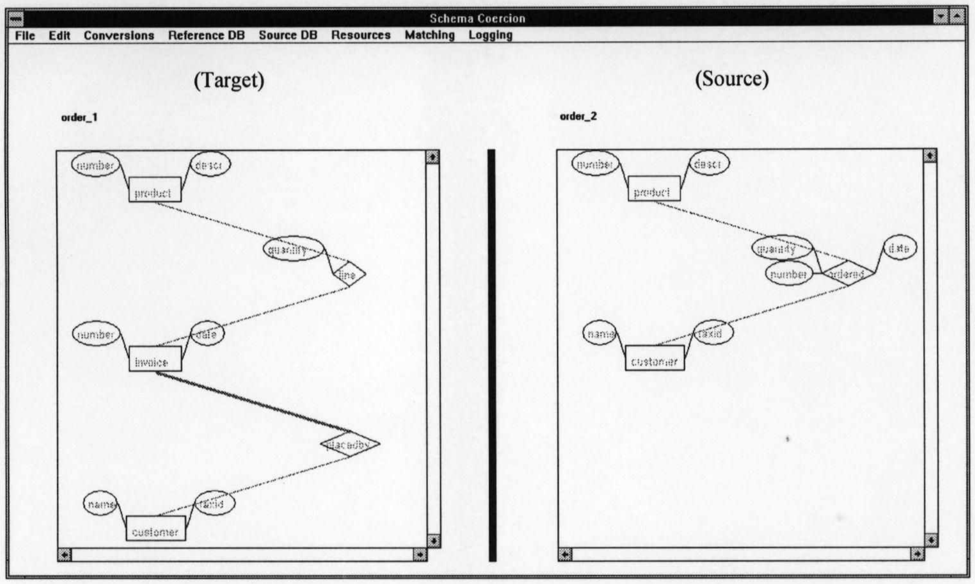
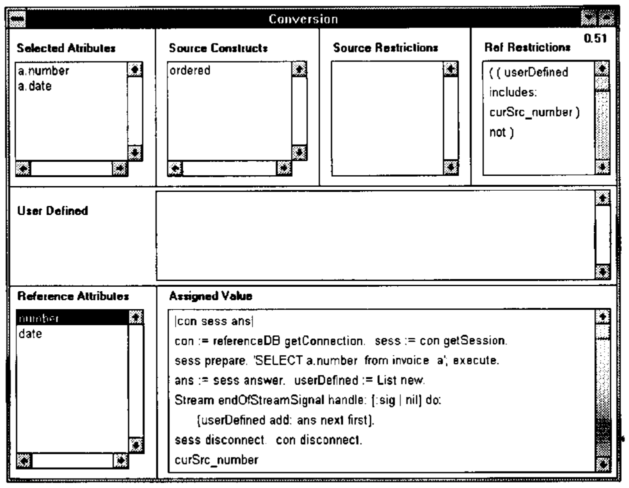


Figure 2 Simple Coercion

The third step in the coercion process -- conflict resolution -- is performed in conjunction with the correspondence identification step. Whenever a conversion is created, default transformations are assigned to the target attributes. These transformations use Smalltalk code to resolve simple conflicts between the source and target representations, and are usually either simple assignments or type safe casts. However, in some cases more complicated transformations are generated. For example, since *number* is not a key for *ordered*, but is a key for *invoice*, duplicate records may be inserted into the *invoice* table. To prevent this, the default transformation, shown by the code in the lower right of Figure 3, queries the target database to identify all existing key values. The default reference restriction prevents duplicate insertions from occurring. If a corresponding source attribute cannot be identified, the target attribute is assigned a null value. Unfortunately, our tool is not able to automatically resolve all conflicts; however,



Selected Atributes: a.number, a.date

Source Constructs: ordered

Ref Restrictions:

```
( ( userDefined
includes:
curSrc_number )
not )
```

Reference Attributes: number, date

Assigned Value:

```
|con sess ans|
con := referenceDB getConnection.  sess := con getSession.
sess prepare. 'SELECT a.number  from invoice  a'; execute.
ans := sess answer.  userDefined := List new.
Stream endOfStreamSignal handle: [:sig | nil] do:
    [userDefined add: ans next first].
sess disconnect.  con disconnect.
curSrc_number
```

Figure 3 Default Transformation

by defining each transformation as a stub function, the user is able to replace it with specialized code which explicitly resolves the conflict.

The association of additional meta-information with database constructs has been recognized as a way to further automatic conflict resolution [24]. The approach taken by this work is to allow definition of database independent annotations, which guide the correspondence identification and conflict resolution steps. The format used to associate annotations with constructs is similar to an association list, where the construct is specified and the annotations associated with it follow in an arbitrary order. Currently, information such as aliases, data types, and transformations, may be associated with a construct. Aliases allow naming conflicts to be avoided by renaming the constructs. Annotations allow the data type associated with an attribute to be refined, not just redefined, providing additional semantic information about the attribute. This permits some semantic conflicts to be identified and resolved using the appropriate transformations. For example, consider the annotation presented in Figure 4, in which

```
(price   (typeInformation       (Currency    USDollars))
         (typeConversions       (CanDollars  convertFromCanToUS: )))
```

Figure 4 Simple Annotations

the price attribute is specified to be type *USDollars*, which is a refinement of type *Currency*. The associated type transformation ensures that if an attribute of type *CanDollars* is associated with this attribute the semantic conflict is automatically resolved. In addition to transformations based on data types, default transformations, which are used when no matching source attributes are identified, and mandatory transformations, which are used whenever the attribute is involved in a conversion, may also be defined. [7] presents a detailed discussion of all currently defined annotations and their syntax. There is obviously other meta-information that may be desired; however, the current set of annotations is sufficient for the existing correspondence identification algorithm. Additional annotations, representing new meta-information, may be incorporated in the future.

There are three advantages to defining annotations in a database independent format as opposed to using a tight binding between the annotations and the database representation [14]. First, it allows meta-information to be associated with database management systems that do not provide meta-information capabilities. For example, flat file databases do not have meta-information associated with them; by using an independent representation, these databases may be treated the same as the traditional relational and object-oriented databases. Second, this format does not require database restructuring. This is an important consideration for database representations that do not allow for dynamic restructuring, such as ASN.1. Third, this format allows annotation files to be shared between similar databases. For example, if several similar databases were coerced onto the same target, the annotation file created for the first source database could be shared, or copied and modified, with the other source databases. This may significantly reduce the interaction required to perform the desired coercions. In addition, the interaction required to annotate the target database may be distributed among all the coercions in which it participates.

Once all conversions and transformations are specified, a translation program may be generated to perform the data transfer. This program performs the data transfer one conversion at a time, with conversions to target entities performed before conversions to target relationships. Each conversion incrementally reads the required information from the source database, performs the transformations for each target attribute, and updates the target database. External interfaces mask the complexity of communicating with the underlying database management systems to perform the required retrieval and update functions by defining an SQL interface for each system. A vendor supplied interface was used to access the Sybase and Utah Genome databases, while interfaces to the flat file and ASN.1 database systems were created as part of this work. Depending on the initial conversion specification, the translation program may require minor modifications to perform repeated transfers from the same source database. Modifications may be made through the coercion program, or directly to the code.

The interaction required to refine the coercion when the source or target database evolves is reduced by allowing modifications to the existing coercion. In particular, the current schemata may be modified within the tool to reflect the new schema, and the affected conversions and transformations may be explicitly modified. If the new representation is vastly different than the old, it may be necessary to define a new coercion. However, by re-using the annotations for the original databases, with appropriate modifications, the interaction required to define this coercion will be significantly reduced. The ability to update a coercion based to conform to a new representation is a significant problem and has not been addressed, although handling the simple cases could be easily incorporated.

Validation

The feasibility of the approach presented in the previous section has been demonstrated using a number of preliminary experiments and two significant applications. Coercions performed between several small databases, such as the example from the previous section, demonstrate the tool's ability to resolve various conflict challenges reported in the literature. The correspondence identification algorithm is able to robustly identify correspondences despite a variety of conflicts, and created reasonable transformations based solely on the available information.

Unfortunately, this is the extent of the testing performed by most schema manipulation programs. In contrast, in order to determine the value of our tool in a real world setting, it was used to coerce the Haemophilus and Methanococcus genome data from the Genbank ASN.1 database to the Utah Genome database. These coercions were chosen because a local representation of this data was a high priority for current research [12]. Because of the radically differing database representations, the correspondence identification algorithm was unable to automatically identify conversions or define transformations. This is not surprising considering that, for example, the genetic sequence data is stored in a binary representation associated with the attribute bearing the uninformative name *ncbieea* within Genbank. It took approximately one day to explicitly create all of the transformations required for the coercion, primarily because several of the transformations required querying the target database to obtain additional information. Modifying the Haemophilus coercion to perform the Methanococcus transfer required only thirty minutes, even though the data representation had several minor differences.

The value of annotations was demonstrated by creating an annotation file for the Utah Genome database in the Haemophilus coercion. It took approximately one hour to define the 38 annotations in this 100 line file. Using these annotations, the program was able to identify all of the conversions, and correctly define 62% of the transformations. In addition, 13% of the incorrect transformations were partially correct, but were simple assignments instead of more complex operations. For example, one default transformation should have been a choice between two source attributes. While a specific annotation could have been defined to correctly generate this transformation, it would not reduce the amount of interaction required. The ability to define a significant portion of a complicated coercion quickly and easily through an annotation file is a significant accomplishment.

While the results of the genetics coercions were disappointing in some aspects, these coercions validated our holistic approach to addressing the schema coercion problem. Despite the failure of the correspondence identification and conflict resolution heuristics, the ability to explicitly identify correspondences and resolve conflicts within a consistent environment was extremely beneficial. By providing a comprehensive tool, the interaction required to create these coercions was focused on these steps, and additional interaction to address the pre-integration and data transfer steps was not required. In conjunction with the specification environment, this focus allowed the coercion to be defined in one day, as opposed to the one week it has been estimated a programmer would require to define an equivalent coercion from scratch. This strengthens our belief that creating a tool addressing all of the steps involved in the coercion process dramatically increases the usability of the tool.

Conclusions

This work makes significant practical and theoretical contributions. In practical terms, the tool developed as part of this work addresses the entire schema coercion process within a consistent environment. This allows complex coercions between real-world databases to be easily defined. The generality of this approach is underscored by the variety of DBMSs that are recognized by the tool. From a theoretical perspective, this work presents three major contributions. First, it defines a new area of schema manipulation. While schema coercion is frequently encountered in database manipulation, there has been no prior research specifically targeted at this problem. Second, it presents a correspondence identification heuristic that uses all available information to determine a confidence in the correspondence between two constructs. This is a dramatic departure from traditional algorithms which only use the construct name, an possibly the attribute names, to determine whether constructs correspond. In addition, by associating confidences with the conversions, existing conversions used in conjunction with representational knowledge are able to identify complex correspondences. Third, it defines a database independent representation for annotating schemata. This allows meta-information to be associated with primitive as well as relational and object-oriented DBMSs, and allows similar databases to share annotations.

The current version of the coercion tool is a framework in which components may be enhanced or replaced as required; it was never intended to be an ultimate realization of this approach. To that end, several enhancements are anticipated as the needs of the Utah Genome Center expand. First, the set of recognized databases is expected to eventually include OODBs such as ObjectStore [13]. The addition of new database systems requires the definition of a translation from the underlying data model to the ER model, and the ability to generate retrieval and update queries on the database. Second, the correspondence identification algorithm may be enhanced or replaced. For example, a thesaurus may be added to the existing algorithm or an alternative algorithm such as an expert system may be added. Third, the set of defined annotations is expected to grow as additional meta-information is required.

To summarize, it is currently impossible to define a single algorithm that will automatically identify all correspondences, and resolve all conflicts, between arbitrary databases. However, by using a combination of good heuristics and interactive tools, the effort required to define and implement a coercion may be significantly reduced. This work presents an engineered approach that resolves complex as well as obvious conflicts; allows complex transformations to be defined; associates meta-information with schemata in a database independent fashion; and works in a real world environment.

Acknowledgments

Funding for this project was provided, in part, by National Institute of Health grant "Utah Center for Human Genome Research".

References

[1] Rateb Abu-Hamdeh, James Cordy, and Patrick Martin. Schema translation using structural transformation. In *Proceedings of the 1994 CAS Conference*, pages 202-215, October 1994.

[2] C. Batini and M. Lenzerni. A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys*, 18(4):323-364, 1986.

[3] Mohamed Bouneffa and Nacer Boudjlida. Managing schema changes in object-relationship databases. In *OOER'95: Object Oriented and Entity Relationship Modeling*, pages 113-122, December 1995.

[4] M. W. Bright, A. R. Hurson, and Simin H. Pakzad. Automated resolution of semantic heterogeneity in multidatabases. *Transactions of Database Systems*, pages 212-253, June 1994.

[5] P. Buneman, S. Davidson, A. Kosky, and M. VanInwegen. A basis for interactive schema merging. In *Proceedings Hawaii International Conference on System Sciences*, 1992.

[6] Christine Collet, Michael N. Huhns, and Wei-Min Shen. Resource integration using a large knowledge base in Carnot. *Computer*, 24(12):55-62, December 1991.

[7] Terence Critchlow. Scema Coercion: Using Database Meta-Information to Facilitate Data Transfer. Dissertation. University of Utah. Available from Dept. of Computer Science, University of Utah. June 1997.

[8] Bogdan Czedjo and Malcolm Taylor. Integration of database systems using an object-oriented approach. In *First Workshop on Interoperability in Multi-database Systems*, pages 30-37. IEEE, April 1991.

[9] David M. Dilts and Wenhua Wu. Using knowledge-based technology to integrate CIM databases. *Transactions on Knowledge and Data Engineering*, 3(2):237-245, June 1991.

[10] J-L. Hainaut, V. Englebert, J. Henrard, J-M. Hick, and D. Roland. Database evolution: The DB-MAIN approach. In *ER'94: Thirteenth International Conference on the Entity Relationship Approach*, pages 112-131, December 1994.

[11] Daniel A. Keim, Hans-Peter Kriegel, and Andreas Miethsam. Integration of relational databases in a multidatabase system based on schema enrichment. Technical Report 9307, Ludwig-Maximilians-Universat Munchen, April 1993.

[12] Karen Young Kreeger. 1995. First DOE Sequencing Project Grants Widely Hailed as Potential 'Treasure Trove'. *The Scientist.* Vol 9 num 3.

[13] Charles Lamb, Gordon Landis, Jack Orenstein, and Dan Weinreb. The ObjectStore database system. *CACM*, 34:50-63, October 1991.

[14] Patrick Martin, James R. Cordy, and Rateb Abu-Hamdeh. Information capacity preserving translations of relational schemas using structural transformations. Technical Report 95-392, Dept. of Computing and Information Science Queen's University at Kingston, November 1995.

[15] Simon Monk and Ian Sommerville. Schema evolution in OODBs using class versioning. *SIGMOD Record*, 22(3):16-22, September 1993.

[16] Young-Gook Ra and Elke A. Rudensteiner. A transparent object-oriented schema change approach using view evolution. In *Eleventh International Conference on Data Engineering*, pages 165-172, March 1995.

[17] Rob Sargent, Dave Fuhrman, Terence Critchlow, Tony Di Sera, Robert Mecklenburg, Gary Lindstrom, and Peter Cartwright. The design and implementation of a database for human genome research. In *The Eighth International Conference on Scientific and Statistical Database Management*. IEEE Computer Society Press, June 1996.

[18] Peter Schwarz and Kurt Shoens. Managing change in the Rufus system. In *Tenth International Conference on Data Engineering*, pages 170-179. IEEE, February 1994.

[19] Edward Sciore, Michael Siegel, and Amon Rosenthal. Using semantic values to facilitate interoperability among heterogeneous information systems. *Transactions of Database Systems*, pages 254-290, June 1994.

[20] Peretz Shoval and Sara Zohn. Binary-relationship integration methodology. *Data and Knowledge Engineering*, 6(3):225-250, May 1991.

[21] Micheal Siegel and Stuart E. Madnick. A metadata approach to resolving semantic conflicts. In *Seventeenth International Conference on Very Large Data Bases*, pages 133-145, September 1991.

[22] Stefano Spaccapietra and Christine Parent. View integration: a step forward in solving structural conflicts. Technical Report tdke93, Institute of Technology in Lausanne, Lausanne Switzerland, 1993. To appear in IEEE TKDE 1993.

[23] Barbara Staudt Lerner and A. Nico Habermann. Beyond schema evolution to database reorganization. In Norman Meyrowitz, editor, *ECCOP/OOPSLA '90 Conference on Object-Oriented Programming: Systems, Languages and Applications European Conference on Object-Oriented Programming*, pages 67-76, October 1990.

[24] Susan D. Urban and Jian Wu. Resolving semantic heterogeneity through the explicit representation of data model semantics. *SIGMOD Record*, 20(4):55-58, December 1991.