An Approach for Matching Schemas of Heterogeneous Relational Databases

Yaser Karasneh[1], Hamidah Ibrahim[2], Mohamed Othman, Razali Yaakob
[1, 2]Department of Computer Science
Faculty of Computer Science and Information Technology
Universiti Putra Malaysia, 43400 Serdang
Selangar D. E., Malaysia
[1]karasneh@gmail.com, [2]hamidah@fsktm.upm.edu.my

Journal of Digital Information Management

ABSTRACT: *Schema matching is a basic problem in many database application domains, such as data integration. The problem of schema matching can be formulated as follows, "given two schemas, S_i and S_j, find the most plausible correspondences between the elements of S_i and S_j, exploiting all available information, such as the schemas, instance data, and auxiliary sources" [24]. Given the rapidly increasing number of data sources to integrate and due to database heterogeneities, manually identifying schema matches is a tedious, time consuming, error-prone, and therefore expensive process. As systems become able to handle more complex databases and applications, their schemas become large, further increasing the number of matches to be performed. Thus, automating this process, which attempts to achieve faster and less labor-intensive, has been one of the main tasks in data integration. However, it is not possible to determine fully automatically the different correspondences between schemas, primarily because of the differing and often not explicated or documented semantics of the schemas. Several solutions in solving the issues of schema matching have been proposed. Nevertheless, these solutions are still limited, as they do not explore most of the available information related to schemas and thus affect the result of integration. This paper presents an approach for matching schemas of heterogeneous relational databases that utilizes most of the information related to schemas, which indirectly explores the implicit semantics of the schemas, that further improves the results of the integration.*

Categories and Subject Descriptors
H.2.4 [Database Management]: Systems – relational databases, transaction processing

General Terms: Algorithms, Management, Measurement, Performance

Keywords: Database integration, Schema matching, Heterogeneous, Biomedical database

Received: 30 July 2010; Revised: 11 December 2009; Accepted: 30 December 2009

1. Introduction

A database schema comprises the gross structure and constraints on the database. Database schemas often do not provide explicit semantics for their data. Database heterogeneities, or differences, can make access to information intricate [22]. Thus, a heterogeneous database that unites various existing databases, which support different schemas and technologies, by providing a uniform database schema and querying capabilities is critically required [22]. The process of integrating data from multiple, heterogeneous sources are called *heterogeneous database integration* [22]. This process is made harder due to heterogeneities at the following levels: (i) syntactic heterogeneity – differences in the language used for representing the elements; (ii) structural heterogeneity – differences in the types, structures of the elements; (iii) model/representational heterogeneity – differences in the underlying models; and (iv) semantic heterogeneity – where the same real world entity is represented using different terms or vice-versa.

Schemas support declarative access to and manipulation of data. Thus, they represent the prime interface for establishing interoperability between tools that depend on shared data. The *heterogeneous database integration*, or simply database integration, aims at providing a uniform and consistent view, the so-called *global schema*, over a set of autonomous and heterogeneous data sources, so that data residing in different sources can be accessed as if it was in a single schema. In practice, data integration is often done incrementally by starting with a simple global schema and adding new data sources when needed. The integration of a new data source into an existing global schema can be performed in two steps, a *matching* and a *data transformation* step. In the first step, the source schemas are compared against each other to discover their similar and distinct elements. While the distinct elements and their instances can be taken over from the data source, the correspondences between the similar elements are needed in the second step to generate queries for transforming their instances from the source schema into the global schema [22].

Schema matching is a basic problem in many database application domains, such as data integration. Schema matching is a fundamental operation in the manipulation of schema in formatting match, which takes two schemas that correspond semantically to each other. Schema matching is typically performed manually, perhaps supported by a graphical user interface. Manually specifying schema matches is a tedious, time consuming, error-prone, and therefore expensive process, this is a growing problem given the rapidly increasing number of data sources to integrate. As systems become able to handle more complex databases and applications, their schemas become large, further increasing the number of matches to be performed. The level of effort is at least linear in the number of matches to be performed, maybe worse than linear of one

need to evaluate each match in the context of other possible matches of the same elements [20].

This paper presents an approach for matching heterogeneous relational databases' schemas by exploiting most of the information related to schemas. Our solution considers both the syntactic and semantic heterogeneities and provides data integration without user intervention, which further improves the results of the integration. We are more concerned in matching schemas of relational model since this model is the most dominant model and many database applications; especially applications for distributed environment are still adopting this model [7]. Due to its simple structure (table form), many users preferred to use this model such as the biomedical scientists where many existing biomedical databases for instance GeneCards, Health Information Resource Database, Stanford HIV Drug Resistance Database (HIV DB), and Chronus are developed based on this model [16].

The rest of the paper is organized as follows. Section 2 presents some background concepts related to the work presented in this paper. Section 3 highlights the previous works related to the research. Emphasis is given to those research works focusing on relational model. Section 4 presents our proposed approach while the achievement of the proposed approach is discussed in Section 5. Conclusions are presented in the final section, 6.

2. Preliminaries

The schema of a database system is its structure described in a formal language supported by the DBMS. In a relational database, the schema defines the relations (tables), the attributes (fields) in each table, and the relationships between attributes and relations. The database schema integration is important in building multi database systems. Integration is a set of processes that includes schema mapping and schema matching.

A schema mapping is a high-level specification that describes the relationship between two database schemas. As schema mappings constitute the essential building blocks of data exchange and data integration, an extensive investigation of the foundations of schema mappings has been carried out in recent years.

Rahm and Bernstein [20] had mentioned three types of mapping functions: (i) Term Mappings: Terms/data items in the two databases are related to each other. The relationships between those terms can be one-one. (ii) Simple Structural Mappings: Schema element in one database is related to schema element/elements in another database. (iii) Conditional Structural Mappings: The mapping is condition-dependent. This is the case where the schemas together with allowed data values are transformed in an integrated fashion to map a query to a target database in which both schema and the structure of the data values are different.

Schema matching is the process of identifying semantic mappings, or correspondences, between two or more schemas. Schema matching is one of the steps and critical part of schema integration. The resulting schema can then be in the form of a view, or schema, of an integrated database. In its simplest form, schema matching consists of identifying two elements from two different schemas as semantically equivalent, or matched. In Figure 1, for example *GeneName* from schema *A* and *Name* from schema *B* are match. This is an example of a match that could have been determined by examining just the element names. It is also an example of a direct match, sometimes also termed a match of 1:1 cardinality, means that a single

Schema A

Gene ID	Species	CHR	Strend	Gene Name	Nucl
3248743	human	SH2DIA	2498	Homo Sapines	DNA
4584032	house mouse	mKIAA0465	17869	Mus Musculus	RNA
1238700	Zebrafi	zgc 12873	28732	Danio	RNA

Schema B

GID	Organism	Name	CHR_Str	CHR_End	Ref
3248743	eukaryota	Rattus	323	2821	taxon:9606
4584032	eukaryota	Bos Taurus	324	18193	taxon:10116
1238700	chordata	Homo sapiens	324	29056	taxon:9913

Figure 1. Simple Schema Matching based on Element Name

element from one schema is matched to a single element in another schema.

Much research has been directed at developing direct matches. In the example above, the chromosome may require an indirect match, sometimes termed a match of cardinality 1:*n*, to match *Strend* in schema *A* with *CHR_Str* and *CHR_End* in schema *B*. Even so, one schema might have additional information not in the other. In this case, only a partial match would be possible between the two schemas. This can also be seen with the element *Nucl* in schema *A*, which has no corresponding match in schema *B*.

Moreover, some matches may be complex matches, also termed matches of *m:n* cardinality, where multiple elements in one schema must be matched to multiple elements in the other. The example in Figure 2 shows how the appropriate elements to match may need to be determined through an examination of each schema's structure. Consideration of the database structure would help enable the match of *Organism_Name* rather than *OrgID* from schema *C* since *Organism_Name* is presumably the same data type as *Organism* in schema *D*, whereas *OrgID* would not be.

These examples provide some illustrations of problems encountered when attempting to match schemas. Ram and Park [21] considered these problems to be conflicts between schemas, and separated them into schema or data level conflicts. Schema level conflicts include cases where schemas are created with different names for similar entities or attributes (naming conflicts), or where different structures (generalization, aggregation) are

Schema C

GID	Gname	OrgID	CHR_End
1122	humo	2321	908797
1234	Bos Taurus	3230	898789
3384	Rattus norvegicus	3432	470930

OrgID	Organism_Name
2321	human
3230	animal
3432	human

Schema D

GeneID	Gname	Organism
23212	Eukaryoto	human
98892	chordata	animal
49387	norvegicus	animal

GeneID	Nucl	Strend
23212	DNA	908797
98892	RNA	898789
49387	DNA	470930

Figure 2. Example of Schema Matching using Structure

used to represent similar concepts (structural conflicts). Data level conflicts include cases where different data types or units of measure are used to represent similar data items.

3. Related Work

Schema matching has been explored by various researchers in various data models such as ER [17, 18], relational model [2, 3, 8, 11, 13, 19], and XML model [4, 5, 6, 9, 10, 15, 20, 21, 23]. Several solutions in solving the issues of schema matching have been proposed.

Bilke and Naumann [2] developed a method to match schemas based on instance data. The approach develops matches by identifying duplicates. Duplicates are developed based on tuples, rather than by examining and matching characteristics of instance data in single column, as do many other approaches which use instance data. The method considered only direct matches (those of 1:1 cardinality), and complex matches are not considered. A weakness of this approach is that schema elements that have similar data but differing semantic meanings (such as bill-to and ship-to addresses) may be mistakenly identified as matches. Another weakness is that schema elements with data expressed in different units (kilograms vs. metric tons, for example) will not likely be matched. The contribution of this approach is in the method for measuring the duplication between tuples and columns in two schemas, and in the ability to match schemas when column names may be opaque.

iMAP [3] is a composite schema matching system using both schemas and instance data in order to match a source schema to a target schema. Matches are developed for each element of a target schema, and these matches can be complex matches, involving transformations of data elements from the source database and multiple elements in that transformation. At the core of iMAP are several searchers, each which examines source data for possible matches with target data. There are iMAP searchers for various types of data: text, numeric, categorical, and date. The data types of each element are determined by iMAP by examining instance data, rather than by examining the data dictionary. The searchers also include a schema mismatch searcher. The mismatch searcher finds matches where the elements in the source schema may have been represented differently in the target schema. iMAP then uses a similarity estimator to combine matches generated by the searchers for each element in the target schema in order to arrive at a score for each potential match. The last step is a match selector, which examines the matches from the similarity estimator and potentially discards them if they violate constraints.

SemInt match prototype [8] creates a mapping between individual attributes of two schemas (i.e., its match cardinality is 1:1). The schema-level constraints use the information available from the catalog of a relational DBMS. Instance data is used to enhance this information by providing actual value distributions, numerical averages, etc. For each criterion, the system uses a function to map each possible value onto the interval [0..1]. In its main approach, SemInt uses neural networks to determine match candidates. This approach requires similar attributes of the first input schema (e.g., foreign and primary keys) to be clustered together. However, the neural network approach has substantial performance problems for larger schemas. To improve efficiency, the approach identifies a match only to attribute clusters leaving it to the user to select the matching attributes from the cluster.

Cupid [11] represents a sophisticated hybrid match approach combining a name matcher with a structural one. It is intended to be generic and has been applied to XML and relational schemas. The name matcher exploits auxiliary sources for synonyms and abbreviations to obtain linguistic similarity between element names. Cupid directionally selects for each source element the target element with the highest weighted similarity exceeding a given threshold as the match candidate, resulting in element-level correspondences of 1:1 local and possibly *m:n* global cardinality. Cupid relies heavily on the thesaurus provided, limiting its ability to the words available in the thesaurus. It also depends on the data type which might be a disadvantage since a data field does not necessarily be the same data type in two different schemas.

Similarity Flooding (SF) [13] converts schemas (relational, RDF, XML) into labeled graphs and uses fix-point computation to determine correspondences of 1:1 local and *m:n* global cardinality between corresponding nodes of the graphs. The algorithm has been employed in a hybrid combination with a simple name matcher, which suggests an initial element-level mapping to be fed to the structural SF matcher. Unlike other schema-based match approaches, SF does not exploit terminological relationships in an external dictionary, but entirely relies on string similarity between element names.

Qian at al. [19] introduces complex schema matching (CSM) which semi-automatically discovered 1:1 and complex matches for relational data to achieve their goal in improving matching efficiency, precision, and accuracy. Data types and values will be filtered to discard the unreasonable matches by preprocessor and clustering processor. According to domain constraint "types are not related" such as "attributes of text type and attributes of numeric type are not related". Therefore, this can filter the candidate matches which types are not related and reduce candidate aggregation.

Based on the above literatures, it is observed that in relational model, most of the approaches that concentrate on contributing solutions to the issues of schema matching exploit the element name of the schemas that include name of relation and name of attribute [2, 3, 8, 11, 13, 19]. Some of these research works such as [3, 11, 19] used the data type and the domain (constraint) of the attribute to improve the performance of the schema matching process. While research works such as [3, 8, 11] used auxiliary sources, for instance DBMS catalog, to get further descriptions of the entity/attribute names. To the best of our knowledge, [3] and [8] employed the most available information (name of element, description, domain, instances, data type [3], and key constraints [8]) compared to the others. Table 1 summarizes the previous works, based on the taxonomy proposed by [20]. Here, only those works focusing on relational model are presented. It is observed that none of the previous works have considered name of element, description, domain, instances, data type, and key constraints in a single solution.

This paper takes the challenge to address the above limitation in matching schemas where different database designers have the tendency to use different names in naming the elements of the schemas (schema names, attribute names) as well as assigning different data types and roles to an attribute of a relation. In presenting our proposed solution, the following databases' schemas are used which are taken from the biomedical domain. Note that only parts of the databases are shown due to space limitation.

Gene Database:

Gene (GID, Organism, Name, CHRO, Ref, Nucl, Comments, Strend)
Organism (Organism, Organ_Name, Symbol, DBx, Description)

Gene_Product Database:

GeneProduct (ID, Symbol, DBxref, Species, Type, Full_Name)
Chromo (ChromoID, ChromoName, Type, Description)
Species (Species, SpName, Sploc, ChromoID, Comments)

Cistron Database:

Organism (Organism, OrgName, CHRO, DBReference, Description)
Cistron (CistronID, C_Name, Organism, Symbol)

Genome Database:

Genome (Gene_ID, Specifies, CHR, Strend, Gene_Name, Description)
Gene_Chromosome (CHRO, CHR_Name, CHR_Type, Symbol, Description)

4. The Proposed Approach

Exploring and using as much as possible the available information during the process of matching the schemas is important as mentioned in [20, 24]. This is because, the more information is used, the more accurate the results of the schema matching process. This can be seen in Dhamankar's work [3], which achieved 43-92% accuracy on several real-world domains, thus demonstrating the promise of the approach as compared to the others. With this argument, our approach attempts to explore the various elements of relational database schemas during the matching process before the integration process can be realized. Hence, we proposed an approach for matching relational databases schemas by utilizing five matchers namely: relation schemas matcher, attribute name matcher, data type matcher, constraint matcher, and instance data matcher. Each matcher calculates and records the percentage of similarities between the elements being compared. Matching the databases' schemas based on the name of schemas and the name of attributes (element level) are accomplished using the following methods:

i. *n-gram*: strings compared according to their set of *n*-grams, i.e. sequences of *n* characters. An *n*-gram is a sub-sequence of *n* items from a given sequence. The items in question can be letters, words or base pairs according to the application. The percentage of similarities between two elements, *E*1 and *E*2, is calculated based on the following formula,

$$\frac{|2 \times \sum t \in n\text{-gram}(E1) \cap n\text{-gram}(E2) \log P(t)|}{|\sum t \in n\text{-gram}(E1) \log P(t)| + |\sum t \in n\text{-gram}(E2) \log P(t)|} \quad (1)$$

where *n*-gram(*E*1) and *n*-gram(*E*2) are the sets of *n*-gram in *E*1 and *E*2, respectively, and *P*(*t*) is the probability of a *n*-gram occurring in a word.

ii. *synonym*: words in different spelling but have the same meaning are known as synonym, which is part of an ontology. To search for the synonyms of words, a relative word table is used and it is created as sample in our model to show how the matching process is performed. The percentage of similarities is 100 if the terms compared exist as synonyms in the relative word table, 0 if the terms compared exist in the relative word table but not as synonyms, and '-' if one or both of the terms are not in the relative word table.

After applying the *n*-gram method, the percentage of similarities between two words possibly will be improved using the synonym method, which is the main idea of this process. Blending both methods produces correct results as opposed to applying either method as suggested by previous works.

4.1 Relation Schemas Matcher

The relation schemas matcher compares two relation schemas S_i and S_j of two different databases D_i and D_j to identify similarities between these schemas. The comparison is based on the name of the relation schemas. There are $n \times m$ comparisons if D_i consists of n relation schemas while D_j consists of m relation schemas. The *n*-gram and synonym methods are used in measuring the percentage of similarities between these schemas and if there exists similarities between them, then they are considered as candidate for integration. The *n*-gram method applies the equation, Eq. 1, given earlier to measure the percentage of similarities between the two relation schemas, while the synonym method returns either 100%, 0%, or '-' depending on the existence of the name of the relation schemas in the relative word table. The final percentage of similarities is calculated based on *max*(percentage of similarities based on *n*-gram, percentage of similarities based on synonym). It is assumed that *max*(*V*, -) = *V*, where $0 \le V \le 100$.

Table 2 illustrates this idea. Here, two relation schemas are compared at a time and their percentage of similarities is recorded. Notice that six comparisons are performed for the *Gene* and *Gene_Product* databases.

Author	Data Model	Type of Match	Schema							Instance	Relation	Others
			Structure	Element								
				Linguistic		Constraint						
				Name	Description	Data Type	Foreign Key	Primary Key	Domain			
[8]	Relational	1:1	-	Attribute	DBMS Catalog	-	√	√	-	√	-	-
[3]	Relational	*m*:*n*	-	√	√	√	-	-	√	√	-	-
[19]	Relational, XML	1:1 and *m*:*n*	-	√ Attribute	-	√	-	-	-	√	-	-
[2]	Relational, RDF	1:1	-	√	-	-	-	-	-	√ (tuple)	-	Synonyms
[13]	Relational, RDF, and XML	1:1 and *m*:*n*	√	√	-	-	-	-	-	-	-	-
[11]	XML and Relational	1:1 and *m*:*n*	√ (name of nodes)	√	Thesaurus	√	-	-	-	√	√	Synonyms, abbreviations

Table 1. Summarization of Previous Schema Matching Approaches

Schemas of Gene database	Schemas of Gene_Product database	% of similarities based on *n*-gram	% of similarities based on synonym	% of similarities based on *n*-gram and synonym
Relation 1: Gene	Relation 1: Gene Product	46.5	-	46.5
Relation 1: Gene	Relation 2: Chromo	0	0	0
Relation 1: Gene	Relation 3: Species	0	0	0
Relation 2: Organism	Relation 1: Gene Product	0	-	0
Relation 2: Organism	Relation 2: Chromo	0	0	0
Relation 2: Organism	Relation 3: Species	0	100	100

Table 2. Applying the Relation Schemas Matcher to the Relation Schemas of Gene and Gene_Product

From the above example, utilizing only the *n*-gram method results in a single pair (*Gene*, *GeneProduct*) as the candidate for integration, while applying the synonym method alone produces the following pair (*Organism*, *Species*) as the candidate for integration. In our approach the pairs (*Gene*, *GeneProduct*) and (*Organism*, *Species*) are candidate pairs for integration. This clearly shows that relying on only one of the methods will not produce correct candidates for integration.

Once two relation schemas are identified as similar by the relation schemas matcher, identifying the correspondences between these schemas at the attribute level is then performed. This level is responsible to identify whether two attributes from two different schemas are similar or not. If they are similar, they should be merged as a single attribute. This is important as it reduces the number of repeated attributes as well as the number of null values in the global schemas. Three types of matchers are applied, namely: attribute name matcher, data type matcher, and constraint matcher. Figure 3 presents the flows of matching the elements of relation schemas based on our approach.

4.2 Attribute Name Matcher

The attribute name matcher takes two attributes at a time and calculates the percentage of similarities by utilizing the *n*-gram and synonym methods. Similar to the relation schema matchers there are $k \times l$ comparisons if S_i consists of k attributes while S_j consists of l attributes where S_i and S_j are identified as candidates for integration by the relation schemas matcher. The same rules as explained in the relation schemas matcher are applied in calculating the percentage of similarities between the attributes.

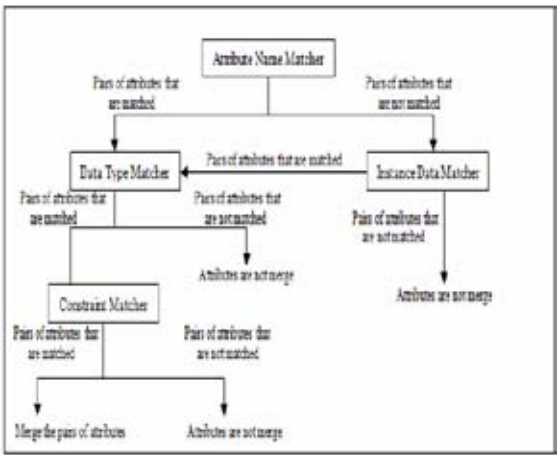


Figure 3. The Flows of Matching the Elements of Relation Schemas

Table 3 illustrates this process. Based on the results of the previous step, the attributes of the relation *Gene*, namely: *GID*, *Organism*, *Name*, *CHRO*, *Ref*, *Nucl*, *Comments*, and *Strend* are compared against the attributes of the relation *GeneProduct*, namely: *ID*, *Symbol*, *DBxref*, *Species*, *Type*, and *Full_Name*. Altogether there are 48 comparisons for this example but due to space limitation only those comparisons that indicate the existence of similarities between the attributes are shown in the table.

Attributes of Gene	Attributes of GeneProduct	% of similarities based on *n*-gram	% of similarities based on synonym	% of similarities based on *n*-gram and synonym
GID	ID	66	-	66
Ref	DBxref	57	-	57
Strend	DBxref	20	-	20
Organism	Species	0	100	100
Name	Full_Name	60	-	60
Comments	Full_Name	14	-	14

Table 3. Applying the Attribute Name Matcher to the Attributes of Gene and GeneProduct

The pairs that are identified as similar are (*GID*, *ID*), (*Ref*, *DBxref*), (*Strend*, *DBxref*), (*Organism*, *Species*), (*Name*, *Full_Name*), and (*Comments*, *Full_Name*). Once a pair of attributes has been identified to have some similarities based on the attribute name matcher, it is not convinced enough to conclude that the pair of attributes should be merged as a single attribute. This is due to the fact that the attributes might have different data types or even different roles which prohibit both attributes to be merged as a single attribute even if the percentage of similarities identified by the attribute name matcher is 100. For this reason, the data type matcher is then applied to the pair of attributes followed by the constraint matcher.

4.3 Data Type Matcher

Data type matcher uses a synonym table specifying the degree of compatibility between a set of predefined generic data types. Table 4 presents the data type synonyms table [25].

binary varying	=	varbinary
char varying	=	varchar
character	=	char
character	=	char(1)
character(n)	=	char(n)
character varying(n)	=	varchar(n)
dec	=	decimal
double precision	=	float
float[(n)] for n = 1-7	=	real
float[(n)] for n = 8-15	=	float
integer	=	int
national character(n)	=	nchar(n)

Table 4. Data Type Synonyms

Table 5 presents the results of applying the data type matcher to the example given in Table 3. Note that only those pairs of attributes that are found to be similar by the attribute name matcher are processed by the data type matcher. Here, the percentage of similarities is 100 if the pair of attributes is matched based on data type and 0 otherwise.

Gene Relation		GeneProduct Relation		Results
Attribute	**Data Type**	**Attribute**	**Data Type**	
GID	Integer	ID	Integer	Match
Organism	Int	Species	Integer	Match
Name	Char	Full_Name	Char	Match
Comments	Varchar2			Not Match
Ref	Varchar2	DBxref	Varchar2	Match
Strend	Integer			Not Match

Table 5. Applying the Data Type Matcher to the Attributes of Gene and GeneProduct

From the above example, notice that although the pair (*Strend*, *DBxref*) is 20% similar as indicated by the attribute name matcher but since they have different data types, then they are considered as not matched. This reduces the number of pairs to be verified by the constraint matcher.

4.4 Constraint Matcher

The constraint matcher compares the roles of two similar attributes to check whether these attributes share the same constraints or not. Similar to the previous matcher, a pair of attributes is matched if they have the same role (100% similarities) and not otherwise (0% similarities). This is shown by the following example, as illustrated in Table 6.

Gene Relation		GeneProduct Relation		Results
Attribute	**Constraint**	**Attribute**	**Constraint**	
GID	Primary Key	ID	Primary Key	Match
Organism	Foreign Key	Species	Foreign Key	Match
Name	-	Full_ Name	-	Match
Ref	Foreign Key	DBxref	Foreign Key	Match

Table 6. Applying the Constraint Matcher to the Attributes of Gene and GeneProduct

As a result, the pairs that are identified as matched by the constraint matcher are the pairs that will be merged into a single attribute, while other attributes will remain as they are.

4.5 Instance Data Matcher

The data type and constraint matchers introduced above are applied accordingly to the pairs of attributes that have been identified as similar by the attribute name matcher. There are also cases where a pair of attributes is detected as not similar by the attribute name matcher but these attributes have similarities with respect to the populated data in their columns. Here, the instance data matcher is applied to identify similarities of attributes based on their populated data. If it is found that there are similarities between the populated data then the pair of attributes are submitted to the data type and constraint matchers. The final decision as whether to merge the pair of attributes depends on the results from each of this matcher as discussed earlier.

5. Results and Discussion

Several analyses have been performed to evaluate the performance of the proposed approach. The first analysis is to show that the elements consider in this paper namely: attribute name, data type, role of an attribute, and instance data can effect the decision whether attributes should be merged or not. None of the previous research works have considered these four elements together in their approach and this analysis is to confirm that it is important to consider these elements during the schema matching process. The results of the first analysis are presented in figures 4 and 5. The percentage of matched attributes between two relation schemas, S_i and S_j, is calculated by the following formula:

$$(\text{number of matched attributes } (S_i) + \text{number of matched attributes } (S_j)) / (\text{number of attributes } (S_i) + \text{number of attributes } (S_j)) \times 100. \quad (2)$$

From the figures we observed that (i) in most cases the percentage of similarities (the percentage of matched attributes) changed (either increase or decrease) whenever a different element is considered. For example the percentage of similarities between *Gene* and *Cistron* when *n*-gram and synonym are applied is 77% and this percentage increased to 84% and 88% (Figure 4) when data type and role of the attribute, respectively are considered. This shows that these elements have effect on the percentage of similarities and the percentage of matched attributes and thus affect the final result of integration; (ii) it is important to note also that utilizing more elements does not mean that the percentage of matched attributes will increase. This is shown clearly in Figure 5. This is because two attributes might have some similarities in terms of their names but they might have different data types or even roles that caused the percentage of matched attributes to decrease.

The second analysis presents the performance of our approach as compared to the previous works with respect to the percentage of similarities and percentage of matched attributes. From the Table 7, it is obvious that our approach achieved higher percentage of similarities and percentage of matched attributes compared to the previous approaches. This is because our approach exploits the various elements of relational database schemas during the matching process.

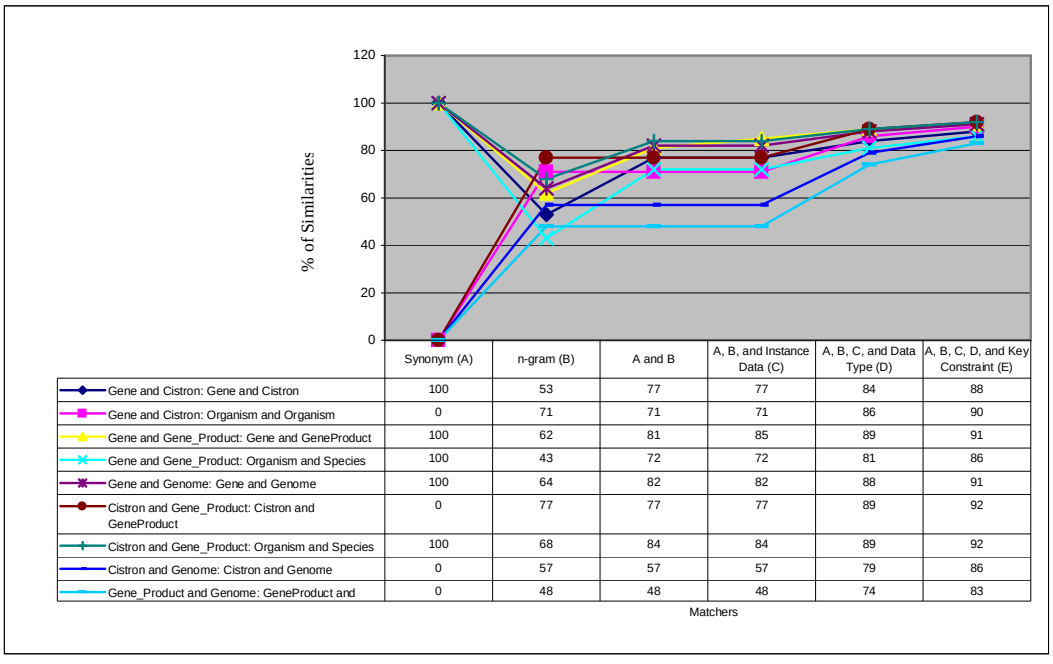
Parameters	Our Model	[3]	[19]	[8]	[13]	[11]	[2]
% of similarities	89	82	80	82	67	73	73
% of matched attributes	69	63	50	62	64	51	51

Table 7. Performance Comparison

The final analysis verified the correctness of the global schemas produced by our matchers as compared to the global schemas derived using the approaches proposed by previous works. As a result two cases have been identified whereby our model produces different results compared to the previous approaches. The first case is presented below.

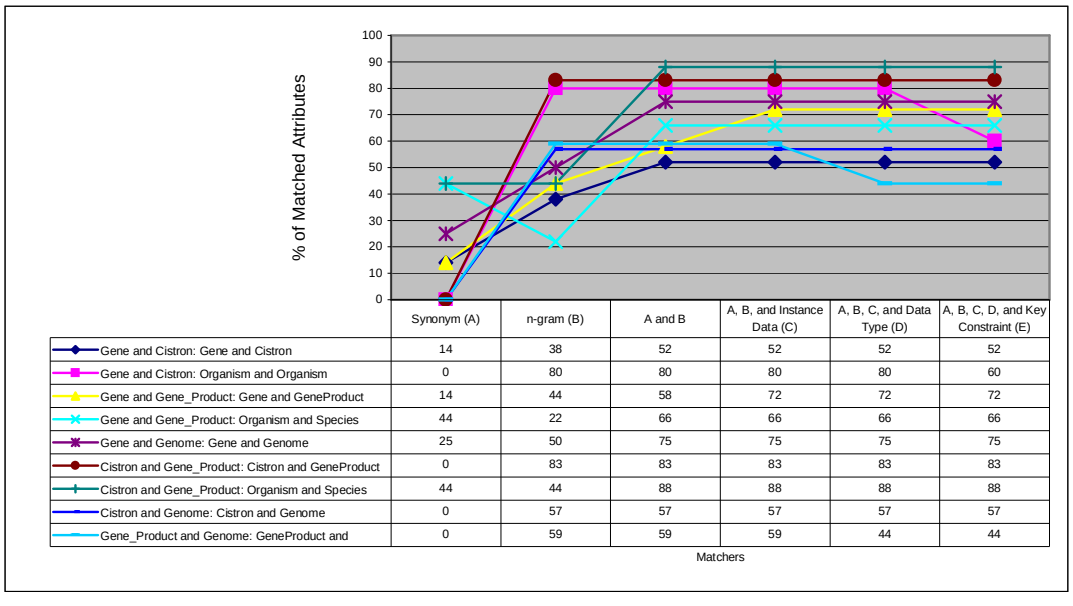
Gene Database: *Organism* relation

Organism Int Primary Key	**Organ_Name** Varchar2	**Symbol** Varchar2	**DBx** Varchar2
Description Varchar2			



	Synonym (A)	n-gram (B)	A and B	A, B, and Instance Data (C)	A, B, C, and Data Type (D)	A, B, C, D, and Key Constraint (E)
Gene and Cistron: Gene and Cistron	100	53	77	77	84	88
Gene and Cistron: Organism and Organism	0	71	71	71	86	90
Gene and Gene_Product: Gene and GeneProduct	100	62	81	85	89	91
Gene and Gene_Product: Organism and Species	100	43	72	72	81	86
Gene and Genome: Gene and Genome	100	64	82	82	88	91
Cistron and Gene_Product: Cistron and GeneProduct	0	77	77	77	89	92
Cistron and Gene_Product: Organism and Species	100	68	84	84	89	92
Cistron and Genome: Cistron and Genome	0	57	57	57	79	86
Gene_Product and Genome: GeneProduct and	0	48	48	48	74	83

Figure 4. Percentage of Similarities with Different Matchers



	Synonym (A)	n-gram (B)	A and B	A, B, and Instance Data (C)	A, B, C, and Data Type (D)	A, B, C, D, and Key Constraint (E)
Gene and Cistron: Gene and Cistron	14	38	52	52	52	52
Gene and Cistron: Organism and Organism	0	80	80	80	80	60
Gene and Gene_Product: Gene and GeneProduct	14	44	58	72	72	72
Gene and Gene_Product: Organism and Species	44	22	66	66	66	66
Gene and Genome: Gene and Genome	25	50	75	75	75	75
Cistron and Gene_Product: Cistron and GeneProduct	0	83	83	83	83	83
Cistron and Gene_Product: Organism and Species	44	44	88	88	88	88
Cistron and Genome: Cistron and Genome	0	57	57	57	57	57
Gene_Product and Genome: GeneProduct and	0	59	59	59	44	44

Figure 5. Percentage of Matched Attributes with Different Matchers

Cistron Database: *Organism* relation

Organism Int Primary Key	**OrgName** Varchar2	**CHRO** Varchar2	**DBReference** Varchar2 Foreign Key
Description Varchar2			

Global Schema: Our proposed model

Organism	**OrgName**	**CHRO**	**DBReference**
Description	**Symbol**	**DBx**	

Global Schema: [2, 3, 11, 13, 19]

Organism	**OrgName**	**CHRO**	**BBx and DBReference**
Description	Symbol		

Using the model proposed by [2, 3, 11, 13, 19], the *DBx* and *DBReference* are merged as a single attribute in the global schema as their models do not consider the foreign key roles of the attribute. In our case, we checked the instance data in the *DBx* column and if there exists at least one instance data that is not null and it does not appear in the referenced attribute (i.e. the foreign key property is violated) then the attributes should not be merged.

The second case is presented below:

Gene_Product Database: *GeneProduct* relation

ID Int Primary Key	**Symbol** Varchar2	**DBxref** Varchar2 Foreign Key	**Species** Integer Foreign Key
Type Varchar2	**Full_Name** Char		

Genome Database: *Genome* relation

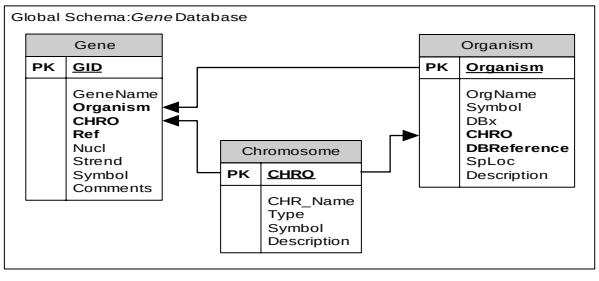
Gene_ID Int Primary Key	**Species** Int Foreign Key	**CHR** Varchar2	**Strend** Int
Gene_Name Varchar2	**Description** Varchar2		

Global Schema: Our proposed model

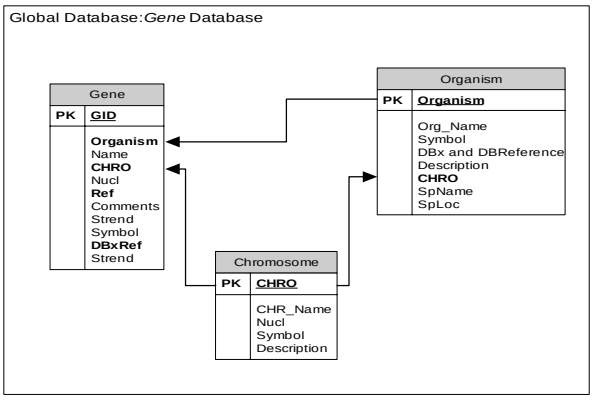
ID	**Symbol**	**Dbxref**	**Species**	**Type**
CHR	**Strend**	**Description**	**Full_Name**	

Global Schema: [8, 13]

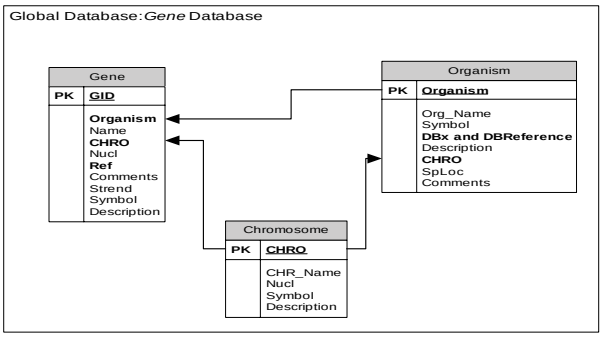
ID	**Symbol**	**Species**	**Type**	**Full_Name**
CHR	**Strend and DBxref**	**Description**		



(a) Global Schemas based on our Proposed Matchers



(b) Global Schemas based on the Matchers Proposed by [2] and [11]

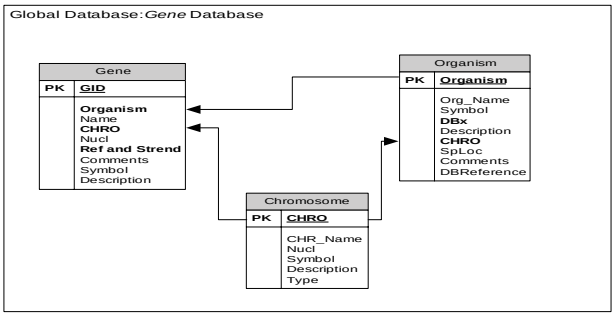


(c) Global Schemas based on the Matchers Proposed by [3]

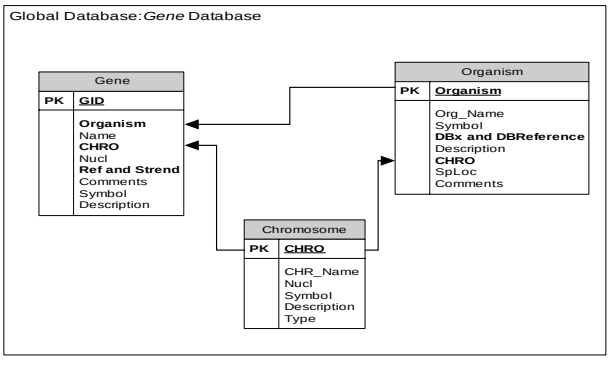
Note that in our model *Strend* and *DBxref* are not merged as they have different data types, while using the models proposed by [8, 13] these two attributes are merged as [8, 13] do not consider data type in their model.

Figure 6 presents the global schemas derived based on the databases' schemas given in Section 3. Figure 6(a) presents the results using our proposed matchers while 6(b), 6(c), 6(d), 6(e) and 6(f) represent the global schemas produced using the matchers proposed by [2], [11], [3], [8], [13], and [19], respectively. These results are produced without user intervention.

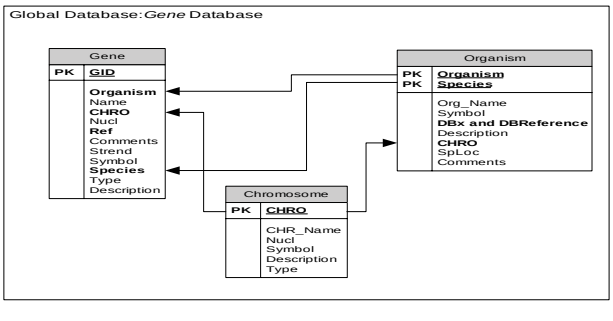
Based on the above results, it is observed that without user intervention, i.e. automating the process of schema/data integration, applying different type of matchers produced different global schemas, as shown in figures 6(a)-(f). The global schemas produced by [2, 3, 11, 13, 19] are incorrect as they merged *DBx* and *DBReference* (*Organism* relation) as a single attribute, while these attributes have different roles. Using the matchers proposed by [8] will merge *Strend* and *Ref* (*Gene* relation) as they do not consider the data type.



(d) Global Schemas based on the Matchers Proposed by [8]



(e) Global Schemas based on the Matchers Proposed by [13]



(f) Global Schemas based on the Matchers Proposed by [19]

Figure 6. Global Schemas Derived using Different Approaches

6. Conclusion

This paper presents an approach for matching relational databases' schemas, which consists of five matchers namely: relation schemas matcher, attribute name matcher, data type matcher, constraint matcher, and instance data matcher. Our approach, which exploits the various elements of relational database schemas during the matching process, has achieved higher percentage of similarities and percentage of matched attributes compared to the previous approaches. Furthermore, we have shown that even without user intervention our approach produces better global schemas during integration. We intend to extend this work by considering complex matches.

7. Acknowledgments

Our thanks to the Universiti Putra Malaysia for supporting this work under grant number 05/01/07/0162RU.

References

[1] Bernstein, P.A., Melnik, S., Petropoulos, M.C. (2004). Quix: industrial-strength schema matching. *Journal of ACM SIGMOD Record*, 33(4) 38-43.

[2] Bilke, A., Naumann, F. (2005). Schema Matching using Duplicates, *In: Proceedings of the Twenty-first International Conference on Data Engineering*, pages 69-80.

[3] Dhamankar, R., Lee, Y., Doan, A., Halevy, A., Domingos, P. (2004). iMAP: Discovering Complex Semantic Matches between Database Schemas, *In: Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, pages 383-394.

[4] Doan, A., Domingos, P., Halevy, A. (2003). Learning to match the schemas of data sources: a multistrategy approach, *Journal of Machine Learning*, 279-301.

[5] Doan, A. Halevy, A. (2005). Semantic-integration research in the database community, *AI Magazine*, 83-94.

[6] Fabien, D., Zohra, B., Mark, R., Mathieu, R. (2007). An Indexing Structure for Automatic Schema Matching, *In: Proceedings of the ICDE Workshops*, pages 485-491.

[7] Jeff, R., Antonio, C. (2008). Plasma: a Graph based Distributed Computing Model, *In: Proceedings of the Workshop at SIGCOMM*, pages 1-39.

[8] Li, W., Clifton, C. (2000). Semint: a tool for identifying attribute correspondence in heterogeneous databases using neural networks, *Journal of Data and Knowledge Engineering*, 33(1) 49-84.

[9] Lu, J., Wang, J., Wang, S. (2005). An Experiment on the Matching and Reuse of XML Schemas, *In: Proceedings of the International Conference on Web Engineering (ICWE)*, LNCS 3579, pages 273-284.

[10] Madhavan, J., Bernstein, P., Doan, A., Halevy, A. (2005). Corpus-based Schema Matching, *In: Proceedings of the Twenty-first International Conference on Data Engineering*, pages 75-68.

[11] Madhavan, J., Bernstein, P., Rahm, E. (2001). Generic Schema Matching with Cupid, *In: Proceedings of the 27th International Conference on Very Large Data Bases*, pages 49-58.

[12] Markowitz, V.M., Ritter, O. (1995). Characterizing heterogeneous molecular biology database systems, *Journal of Computational Biology 2*, 547-556.

[13] Melnik, S., Garcia-Molina, H., Rahm, E. (2002). Similarity Flooding: A Versatile Graph Matching Algorithm, *In: Proceedings of the Eighteenth International Conference on Data Engineering*, pages 117.

[14] Milo, T., Zohar, S. (1998). Using Schema Matching to Simplify Heterogeneous Data Translation. *In: Proceedings of the 24th International Conference on Very Large Data Bases,* pages 122–133.

[15] Mitra, P., Wiederhold, G., Jannink, J. (1999). Semiautomatic Integration of Knowledge Sources, *In: Proceedings of the Fusion '99*, pages 291-331.

[16] Narayanan, P.S., O'connor, M.J., Das, A.K. (2006). Ontology-driven Mapping of Temporal Data in Biomedical Databases, *In: Proceedings of the AMIA Annual Symposium*, pages 1045.

[17] Palopoli, L., Sacca, D., Terracina, G., Ursino D. (1999). A Unified Graph-based Framework for Deriving Nominal Interscheme Properties, Type Conflicts and Object Cluster Similarities, *In: Proceedings of the 4th. IFCIS International Conference on Cooperative Information Systems (CoopIS)*, pages 34–45.

[18] Palopoli L., Sacca D., Ursino D. (1998). Semi-automatic, Semantic Discovery of Properties from Database Schemas, *In: Proceedings of the International Database Engineering and Applications Symposium (IDEAS)*, pages 244–253.

[19] Qian Y., Yue L., Liu Z., (2008). Discovering Complex Matches between Database Schemas, *In: Proceedings of the Control Conference*, pages 663-667.

[20] Rahm, E., Bernstein, P. (2001). A survey of approaches to automatic schema matching, *The VLDB Journal*, 335-350.

[21] Ram, S. Park, J. (2004). Semantic conflict resolution ontology (SCROL): an ontology for detecting and resolving data and schema-level semantic conflicts, *Journal of IEEE Transactions on Knowledge and Data Engineering*, 16(2) 189-202.

[22] Sujansky, W. (2001). Heterogeneous database integration in biomedicine, *Journal of Biomedical Informatics*, 285-298.

[23] Xu, L., Embley, D. (2003). Discovering Direct and Indirect Matches for Schema Elements, *In: Proceedings of the Eight International Conference on Database Systems for Advanced Applications*, pages 39-46.

[24] Yi-Ping P.C., Supawan P., Frédéric M. (2006). MDSM: microarray database schema matching using the hungarian method, *Journal of Information Science*, 176(19) 2771-2790.

[25] http://msdn.microsoft.com/en-us/library/aa258273(SQL.80).aspx#

Authors Biographies

Yaser Mohammad Karasneh was born in Irbid, Jordan, on October 5, 1981. He graduated from Al-Albayt University in 2005 with a degree in Computer Information System. For several years Karasneh worked as a lecturer for OIT institute in Saudi Arabia before relocating to Malaysia, to pursue graduate study in database system. He is taking a Master of Science in Database System from the Universiti Putra Malaysia (UPM) since 2007.

Hamidah Ibrahim is currently an associate professor at the Faculty of Computer Science and Information Technology, Universiti Putra Malaysia. She obtained her PhD in computer science from the University of Wales Cardiff, UK in 1998. Her current research interests include databases (distributed, parallel, mobile, bio-medical, XML) focusing on issues related to integrity constraints checking, cache strategies, integration, access control, transaction processing, and query processing and optimization; data management in grid and knowledge-based systems.

Razali Yaakob obtained his PhD from University of Nottingham in 2008. Currently, he is a senior lecturer at the Faculty of Computer Science and IT, Universiti Putra Malaysia. His research areas include artificial neural network, pattern recognition, and evolutionary computation in game playing.

He received his PhD from the National University of Malaysia with distinction (Best PhD Thesis in 2000). Now, he is a Professor in Computer Science and Deputy Dean of Faculty of Computer Science and Information Technology, Universiti Putra Malaysia (UPM). His main research interests are in the fields of parallel and distributed algorithms, high-speed networking, network design and management (network security, wireless and traffic monitoring) and scientific computing. He is a member of IEEE Computer Society, Malaysian National Computer Confederation and Malaysian Mathematical Society. He is also an associate researcher and coordinator of High Speed Machine at the Laboratory of Computational Science and Informatics, Institute of Mathematical Science (INSPEM), Universiti Putra Malaysia.