ISSN : 0974 - 7532 Volume 4 Issue 4

Trade Science Inc.

Research & Reviews in

BioSciences

Regular Paper

RRBS, 4(4), 2010 [147-155]

# Semantic data management in information integration system based on object deputy model

**Jun Qiang Liu*, Zhiyong Peng, Xiaoling Guan**

**College of Civil Aviation, Nanjing University of Aeronautics and Astronautics, (CHINA)**

**E-mail : liujunqiang@msn.com**

*Received: 19th August, 2010 ; Accepted: 29th August, 2010*

## ABSTRACT

Semantic data management is a key issue in integration of heterogeneous biological data. Traditional database system is not suitable for complex semantic data management. Moreover, most of the biological data management systems cannot provide efficient semantic search over database. In this paper, we present a data warehouse which adopts the object deputy model to store semantic biological data. Most important of all, we can provide more convenient and efficient semantic search for users. The experiment results show that our approach is more feasible and efficient than the traditional one. 

## KEYWORDS

Semantic data management;
Object deputy model;
Semantic search.

## INTRODUCTION

Many biological and medical applications require access to a variety of molecular biological objects, such as genes, proteins, their interrelationships and functions, etc. These objects are maintained in a high number of diverse web-accessible data sources[1]. However, the highly distributed and heterogeneous characteristics of biological databases make it inconvenient greatly to retrieve needed semantic information from different data sources[2]. Semantic data management is one of the great challenges confronting the integration of biological data. It also remains a challenge for the majority of biological researcher on how to carry out the semantic search in those heterogeneous data sources in a rapid and efficient way.

There are mainly two approaches which appeared in the literatures[2,3,5,11,12,14-16]. One approach[5] is that only store semantic similarity table and biological data using relational database and using the term in GO to search semantic similar proteins. Another approach, AceDB[16] has the advantages of accommodation to irregular objects and good schema extension for classes. However, it possesses less flexibility on describing the relationships between objects and classes.

For this purpose, many renowned bioinformatics centers have presented their own solutions. There are some information integration systems such as the Entrez system developed in NCBI[2], the sequence retrieval system (SRS) of EBI[3] and so on. The advantage is that they consider cross-references between sources. However, most of them aim at the heterogeneity of structure, hardly focusing on the integration of the content of data from different databases.

In order to manage semantic heterogeneity in biological data warehouse systems, using ontology for the

Regular Paper

explication of implicit knowledge becomes an approach to provide semantic search. But semantic search based on relation model[12,17] has a lot of join operations which are time-consuming very much. Moreover, only similarity relation between ontology can be expressed in ref.[12] so that the capacity of semantic search is limited.

In this paper, a new semantic data warehouse based on object deputy model has been developed. The system make a clear distinction between data and semantic of the data (ontology), and take into account semantic correspondence between ontologies. Mappings such as DB2GO(mapping from other biology ontology to Gene Ontology), consists of all kinds of semantic relationship(including similarity, Is-a, and so on) and is better than[12]. The construction of semantic relation table is to record all kinds of relationship and similarity scores derived from any pair of GO terms, and cross-reference in instance-level can solve the problem of cross-classes query with semantic. Most important of all, in our framework, it is easy to semantic search in terms of bi-directional pointers between objects and deputy objects, not only saving a mass of storage space, but also having higher performance of semantic search.

## Object deputy model

Object deputy model[7] can satisfy the requirements stemming from complex, high performance biology data managements with the concepts of deputy objects and deputy classes. In this section, we adopt the object deputy model to describe the biology data management.

The object deputy model was at first introduced by the authors for the unified realization of object views, roles, and migration[7]. Biology data consist of the data entity and semantic relations between these entities.

### Definition 1

Each biology object has an identifier and some attributes. The schema of biology objects with the same attributes is defined as a class C = <O, A>.

1 is the extent of C, $o \in O$ is one of instances of C
2 A is the set of attribute definitions of C, $(T_a : a) \in A$, where $T_a$ and a respectively represent type and name of an attribute. The value of attribute a of scientific object o is expressed by o.a. For each attribute, there are two basic methods.

read(o, a) $\Rightarrow \uparrow$o.a, write(o, a, v) $\Rightarrow$o.a := v.

Here$\Rightarrow$, $\uparrow$stand for operation invoking, result returning.

### Definition 2

A deputy biology object is derived from object(s) or other derived object(s). The latter is called source object(s) of the former. Source objects and derived objects are linked by bi-directional pointers between them. Deputy objects have their own persistent identifiers, and can inherit some attributes from their source objects by switching operations without occupying storage space, and can also add their additional attributes. A deputy class defines the schema of deputy objects with the same attributes. Let $C^s = <O^s, A^s>$ be a source class, its deputy class $C^d$ is defined as $C^d = <O^d, A^d * A^d_+>$.

1 Deputy object $O^d = \{o^d_i \mid o^d_i \rightarrow o^s_i \mid \ldots \times o^s_i \times \ldots \mid \{o^s_i\}, sp(o^s_i) \mid jp(\ldots \times o^s_i \times \ldots) \mid gp(\{o^s_i\}) = = true\}$, is the extent of $C^d$, where $o^d_i \rightarrow o^s_i \mid \ldots \times o^s_i \times \ldots \mid \{o^s_i\}$ denote $o^d_i$ is a deputy object of $o^s_i, \ldots \times o^s_i \times \ldots$, or $\{o^s_i\}$, and sp, jp, gp represent selection, combination, and grouping predicate respectively.
2 $A^d \cup A^d_+$ is the set of attribute definitions of $C^d$.
3 $(Ta^d : a^d) \in A^d$ is the attributes inherited from $(Ta^s : a^s) \in A^s$, and attribute values of derived object $o^d$ are computed through switching operations that need to read attribute values of source objects. Switching operation for the read method of these attributes is defined as:

read($o^d$, $a^d$)$\Rightarrow \uparrow$ f$Ta^s \rightarrow Ta^d$ (read($o^s$, $a^s$). $(Ta^d_+ : a^d_+) \in A^d_+$ is the additional attributes of $C^d$, of which basic methods are defined as:

read($o^d$, $a^d_+$) $\Rightarrow \uparrow$ $o^d.a^d_+$, write($o^d$, $a^d_+$, $v^d_+$) $\Rightarrow$ $o^d.a^d_+ := v^d_+$.

According to above definitions, during the course of each query, attribute values of deputy biology objects inherited from source objects are still computed through switching operations that need to communicate with the underlying information source.

### Definition 3

Update propagation between biology objects and their deputy objects.

1 If a biology object o is added into class C, then all of deputy classes of C are checked. If o satisfies the predicate of some deputy class $C^d$, an object $o^d$ of $C^d$ is created as a deputy object of o. Deleting a biology object causes deletion of all of its deputy

objects.

2 If a biology object o in class C is updated, all of its deputy objects will be updated automatically. Suppose that there are some deputy classes of C, of which predicates might not be satisfied by o before the update and may become satiable after the update, new deputy objects of o can be added to these classes. Modification of an object may cause deletion of its deputy objects.

Based on the object deputy model, we have implemented a database system called TOTEM and designed an object deputy database language which can create various kinds of deputy classes, including SelectionDeputyClass, JoinDeputyClass, Union Deputy Class, and GroupDeputyClass.

## Biology information

The amount of molecular biology data available on line increases dramatically in a few years. Despite the abundance of data, the understanding of these data lags far behind the collection. A key question that molecular biologists try to understand is the protein interaction mechanism, i.e. why the variation in proteins can lead to diseases such as HIV (Human Immunodeficiency Virus).

In order to manage semantic data in biological database systems, the meaning of the interchanged information has to be understood across the systems. Using ontology for the explication of implicit knowledge becomes an approach to overcome the semantic heterogeneity. By capturing knowledge about a domain in a kind of shareable, computationally accessible and computable semantics about the domain knowledge they describe, ontology provides a model of concepts that can be used to form a semantic framework for many data storage, retrieval and analysis tasks.

The most popular ontology used in biology field is Gene Ontology (GO). GO provides a structured controlled vocabulary of gene and protein biological roles, which can be applied to different species. GO describes gene products from three orthogonal taxonomies or aspects: molecular function, biological process and cellular component[6]. These vocabularies and their relationships are represented in the form of directed acyclic graphs (DAG). We use GO as a tool for biological clustering and constructing DB2GO class to correlate GO annotated entries from data sources with GO terms, and provide semantic similarity class to record similarity scores derived from any pair of GO terms and cross-link and cross-link(gene) class to record the correlation in instance level.

## A semantic data management example

In this section, we give an example of the semantic data management system. Consider the example of a pharmaceutical researcher who wants to research drugs to combat HIV.

The HIV is composed of two RNA strands encased in a protein envelope. The viral envelope has 2 proteins, named gp120 and gp41. The gp120 binds to CD4, a receptor protein on a type of white blood cell, called CD4+ T cells. The gp41 then causes the fusion of the HIV with the T cell. After the virus has merged with a cell, the viral RNA is inserted into the cytoplasm of the cell. Each virus particle has 2 copies of an RNA genome, which are transcribed into DNA in the infected cell and integrated into the host cell chromosome with the help of an enzyme called reversed transcriptase. The viral RNA copies itself into the DNA of the cell, causing the cell to produce more of the viral RNA. The RNA transcripts produced from the integrated viral DNA serve both as mRNA to direct the synthesis of the viral proteins and later as the RNA genomes of new viral particles, which escape from the cell by binding from the plasma membrane, each in its own membrane envelope. For more details on how HIV operates, you can see (http://www.niaid.nih.gov/factsheets/howhiv.hom).

A researcher may first find all related proteins using sequence alignment method to a protein that is known to be involved in a disease process, such as gp120, gp41 or CD4. Second, he may want to find all drugs correlated with these proteins and know about what happened in gene. However, sequence alignment method has certain limitations that are compounded by the increasing size of the target database: Sequences containing many repetitive elements or LCR (Low-complexity regions) are likely to produce false positive or seemingly unrelated matches. Semantic similarity search based on molecular biology functions is a better method for semantic search.

## Semantic data management

In this section, we use the example above to explain our framework. This system can retrieve the cor-

Regular Paper

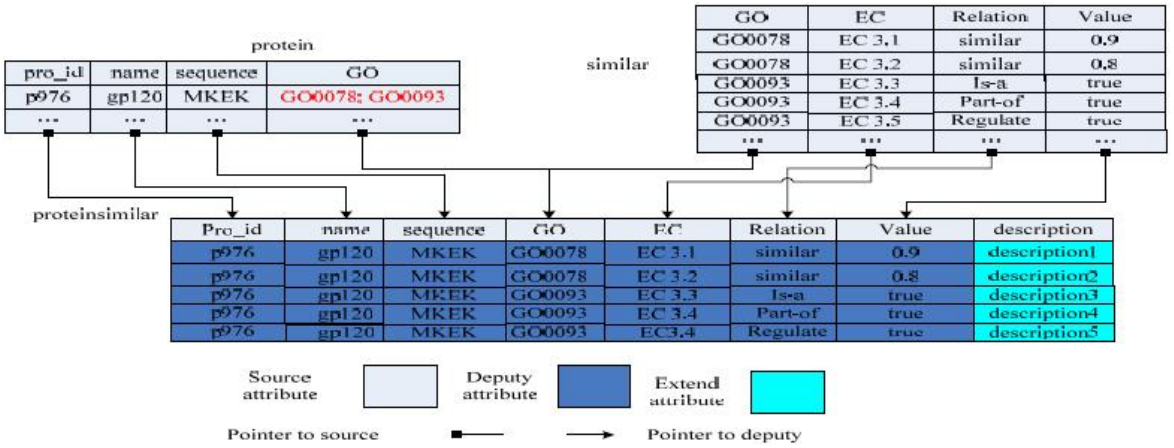


**Figure 1 : Semantic integration**

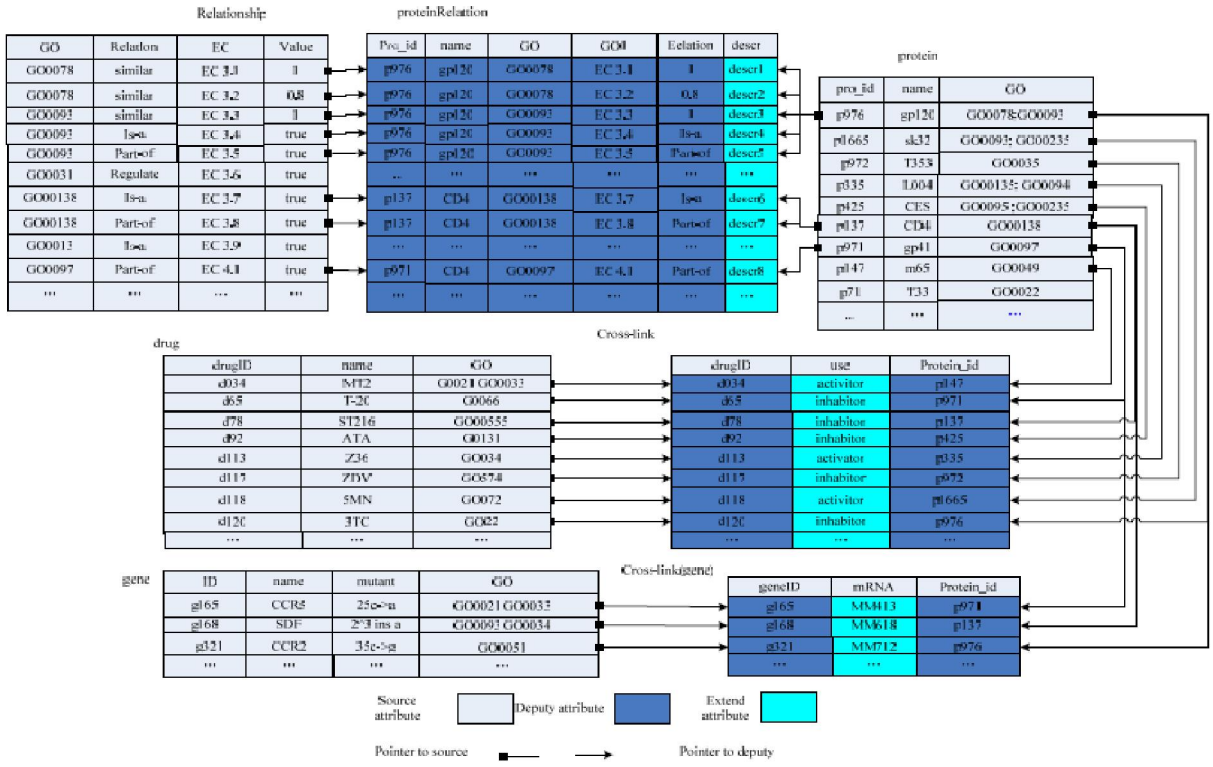


**Figure 2 : Semantic data management**

relation data from different sources through the procedure of data extracting, transforming and cleaning as described in[9]. By using GO as a tool for biological clustering and constructing DB2GO table to correlate GO annotated entries from data sources . By constructing DB2GO, the entries in the member databases are linked to the terms of Gene Ontology Database organically. In the DB2GO table, it is quite easy to find entries, which are all annotated by the same GO term, from different databases. Since all these entries are annotated by the same GO term, they are semantically identified, and thus the semantic heterogeneity between databases is partially resolved. Various measures have been developed for quantifying the semantic similarity in terms of ontology. In order to carry out the comparison of semantic similarity in a convenient way, we calculate the similarity of every pair of terms in GO (each term corresponds with a node in the DAG of GO) according to the algorithm[8] and experts, and then store the results into the semantic similarity table as follows.

The difference between our approach and paper[12] is that there is only similarity between ontology in[12], whereas there are many semantic relationships as shown in table DB2GO in our system.

Regular Paper

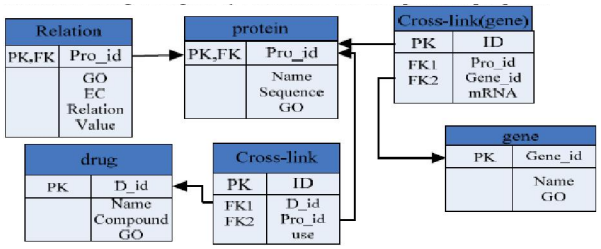


**Figure 3 : Semantic data management in traditional database**

**TABLE 1 : DB2GO table**

| ID | GO | Relation | Enzyme | Value |
|---|---|---|---|---|
| 001 | GO:0003998 | similar | EC 3.6.1.7 | 0.9 |
| 002 | GO:0000210 | similar | EC3.6.1.23 | 0.8 |
| 003 | GO:0004551 | Is-a | EC 4.1.1.49 | true |
| 004 | GO:0004612 | Part of | EC 4.1.1.39 | true |
| 005 | GO:0004074 | Regulate | EC 1.3.1.24 | true |

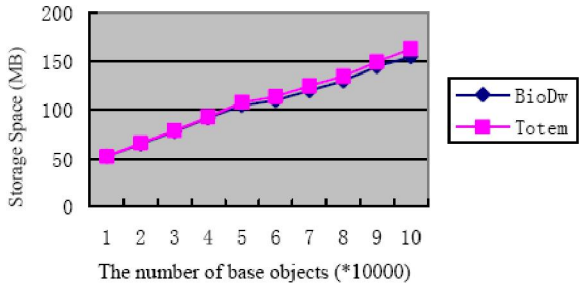


**Figure 4 : Comparison of storage space**

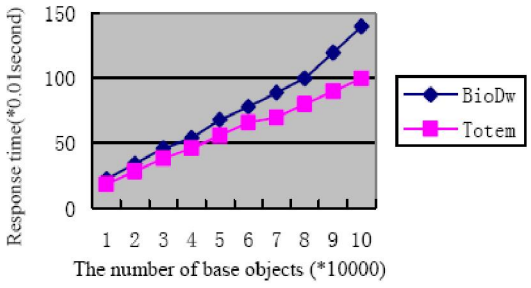


**Figure 5 : Comparison of response time in query 2**

Modeling semantic information by relational data model need multiple tables and these tables must be linked using primary key and foreign keys. Object-oriented data model can be used to define complex semantic objects. Objects with the same attributes and methods can be defined by a class. Most of object oriented databases restrict that each object is a direct instance of only one class and indirectly belongs to all super-classes of this class. That is, an object cannot reside simultaneously in two classes which are not related by a sequence of IS-A relationships. The complex semantic data comprises of all kinds of relations (IS-A, Part-of, Regulate, and so on), so it's hard to be stored in object-oriented databases. In addition, data redundancy and consistency maintenance are also great handicap. Object deputy model can overcome the above problems in the following.

## Semantic integration

We can get the DB2GO table from data sources, such as protein from swissprot[4], and get the *similar* class is computed according to the algorithm in[8], and the objects value of *cross-link* and *cross-link(gene)* class are get from[14]. All DB2GO tables about proteins are union as *protein* class in our system. The data downloaded from data sources consists of multiple semantic values. How to merge with similar table in figure 1 is difficult in relational model or object oriented data model. The nested attribute structure (one ontology may has many similar ontology, so many ontologies form the nested structure) is not supported effectively in information integration using relational model, and object oriented model is not enough for information integration because it has two serious problems. It can only provide subclass constructor and support inheritance from super-class to subclass. Information integration needs not only specialization but also aggregation. The JoinDeputyClass using object deputy model can solve the problem.

CREATE Join Deputy Class proteinsimilar
SELECT {proteinsimilar.name:=protein.name},
{proteinsimilar. GO := protein. GO. transformation [GO, ; ]}
{proteinsimilar.GO1=similar.GO1}
FROM protein, similar WHERE protein. GO. transformation [GO,;]=similar.GO
EXTEND(description:text)

The transformation[GO,;] means that according to ";" split the GO term. After that, we can get the similar class. So, object deputy model provides a controllable, transformable inheritance mechanism to solve semantic heterogeneous problem. In the next subsection, we will discuss the semantic data management for efficient semantic search.

## Semantic data management

In the figure 2, the relation among protein, relationship, and proteinRelation class is same in figure1. The class gene and drug both are basic class, which is defined easily according the SQL language in ref.[13].

Regular Paper

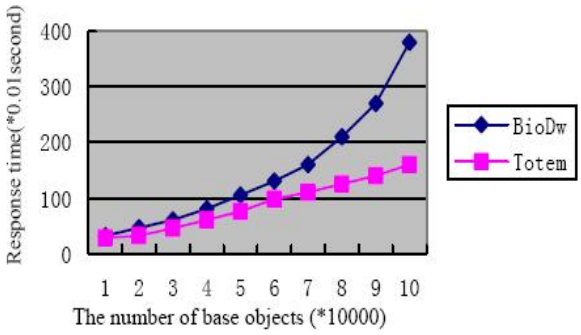


**Figure 6 : Comparison of response time in query 3**

Cross-links in instance level are available in many biology and biomedical domain[10]. There are more complex semantic relations in cross-links, so when it is impossible or at least difficult to concrete selection predicates, users can choose to create "imprecise" deputy class. For example, the imprecise deputy class Cross-link can be defined as follows.

CREATE Imprecise Join Deputy Class Cross-link
SELECT {Cross-link.protein_id=protein.pro_id},
{Cross-link. drugID=drug.ID}
From drug, protein Extend(relation:string)

The propagation module will not create deputy objects for the imprecise deputy class automatically. We define a special syntax to create imprecise deputy objects manually as follows:

ADD ANY INTO Cross-link FROM protein WHERE protein_id=976,
With (relation, drug) values('inhabitor','d120');

According to the syntax, the more complex relation such as "inhabitor" can be stored. The class cross-link(gene) is also created using imprecise deputy class.

Please note that it needs primary key and foreign key to link in the relational model. However, the primary key or foreign key is hard to define in Cross-link table, so it must be expand other attribute as primary and foreign key. Moreover, the join operations will make system performance lower. But the imprecise deputy class in object deputy model is based on bilateral pointers and inheritance mechanism. So, the query and store performance is better than relational model.

## Semantic query

Semantic search is a key issue in integration of heterogeneous biological databases. In this section, we first propose two kinds of semantic search methods in our framework, and argue that our method has more advantage than relational model.

## Semantic similarity search

Object deputy database provides a SQL-like language[13] which can be used to query classes and their deputy classes. For example:

### Query1

Select all proteins which semantic are similar with gp120, gp41 and CD4 about HIV and similarity>0.5.

According to the figure2, we can get the information through two steps:

1 Select name, GO1 from proteinsimilar where (proteinsimilar.name='gp120'or proteinsimilar. name='gp41' or proteinsimilar.name='CD4') and proteinsimilar. similar>0.5. The result of GO1: {GO0078, GO0035, GO0093, GO0095, GO00135, GO00138, GO0049, GO0097}
2 Select name, GO from protein where protein.GO like '%GO0078%' …..or protein.GO like '%GO0097%'.

If in traditional biology database in ref.[12], because it has no ability to deal with nested attributes structure, get the same information becomes more complex.

1 Select name, GO from protein where protein.name='gp120' or protein.name='gp41' or protein.name='CD4'; The result of GO={GO0078, GO0035, GO00138, GO0097}
2 Select name, GO1 from similar where similar.similar>0.5 and ( similar.GO='GO0078'….. or similar.GO='GO0097') ; The result of GO1 :{GO0078, GO0035, GO0093, GO0095, GO00135, GO00138, GO0049, GO0097}
3 Select name, GO from protein where protein.GO like '%GO0078%' …..or protein.GO like '%GO00135%';

From above, we can known that our model have more convenient semantic search than traditional biology database.

## Cross-classes semantic search

With a powerful navigation mechanism, our object deputy database supports the cross-classes query efficiently and conveniently. Since related objects in our object deputy data warehouse are connected with bi-directional pointers, it is easy to navigate from one object to any of its related objects via these pointers. We use the symbol "→" to represent the navigation between classes.

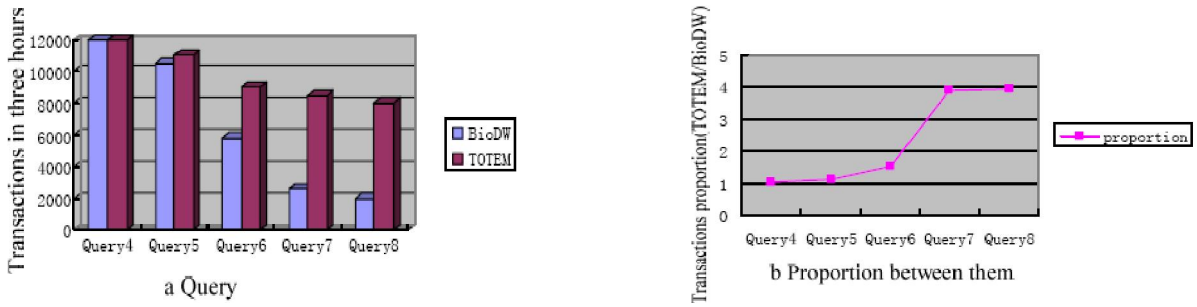


**Figure 7 : Capability of transaction processing**

## Definiton 4

## Path expression

Let $C_b$ be the class from which the navigation begins and $C_t$ be the target class. For any object $o_b \in C_b$, the expression $C_b \rightarrow C_t$ will return all its connected object $\{o_t\} \in C_t$. If no target object could be found, the result will be empty. The expression $C_b \rightarrow C_t$ is called a "path expression".

An interesting feature of the semantic path expression in our object deputy database is that only the "start" and "target" classes should be explicitly written in query. Since the deputy relationship forms a directed graph, one start-to-target class pair determines a unique path. In this way, the database users can be greatly benefited from getting rid of understanding perplexing data schema and expressing complex query. With these features, path expression makes the query not only simple and easier to understand but also more efficient to execute. For example (Figure 2), and experts whose research are focused on HIV may want to get the semantic similarity information correlation with drugs information or (and) genes information.

## Query 2

Select all proteins with semantic similarity about HIV, and drugs correlated with proteins using semantic query result in Query1.

Select protein.name, drug.name, drug.GO from protein,drug WHERE protein->drug and cross-link.relaiton='inhabitor' and (protein.GO like '%GO0078%' …or protein.GO like '%GO0097%');

## Query 3

Select all proteins with semantic similarity about HIV, and drugs and genes correlated with proteins using semantic query result in Query1.

Select protein.name drug.name, gene.name from drug, gene WHERE (protein->drug or protein->gene) and cross-link.relaiton='inhabitor'and (protein.GO like '%GO0078%' …or protein.GO like '%GO0097%');

In traditional biology database, the same query will become much more complex, which involves a lot of join operations. Due to limited space, we will not write the query of relational database.

## Consistency maintenance

Since attributes and methods of a biological object are inherited by its deputy objects. So, the data redundancy is avoided in our semantic data structure. When biological object with semantic are modified, their deputy objects should be changed accordingly. Object deputy model can provide object update propagation mechanism to maintain their consistency. That is, when a biological object with semantic is added, its deputy objects may be created automatically according to semantic constraints, when a biological objects is deleted, its deputy objects will be deleted; when a biological object is updated, the biological objects must be updated.

Thus, our biological data management system can classify the consistence of semantic objects dynamically. For example, the insertion of a biological object name="gp120", pro_id= '35', GO="GO0078; G00093"in protein, it will create deputy objects automatically in the proteinsimilar class. If we delete it from *protein*, then all related objects will be automatically deleted. If name= 'gp120' is changed, the name of deputy objects will be changed.

## Experiment and analysis

In this section, we will evaluate performance of semantic search using the Query 2 and Query 3 in the semantic query section, and storage cost under the same environmental setting.

The experiment runs on a Celeron machine which has 3.0 GHz CPUs, 1000MB main memory, and

Regular Paper

RedHat Linux 9.0 system. The traditional **bio**logy **d**ata **w**arehouse in ref.[12,17] called BioDW are implemented using PostgreSQL database for comparison purpose.

We have done tests based on 10 data sets extracted from biology data sources in BioDw[12,17] system and TOTEM database management system. The sizes of data sets are from 1M to 10M objects. The x-axis represents the number of objects and the y-axis represents the relevant costs. In biology database, the values of attributes are inherited without occupying much storage spaces. Although the virtual attribute does not occupy the storage space, the schema information for deputy classes and the bilateral links between objects and their objects need be stored. Therefore, the storage space consumed by TOTEM should be close to traditional biology database. The experiment result as shown in figure 4 proved our analysis.

From figure 5 and 6, we can see that TOTEM has great advantages than BioDw[12,17] in response time. In biology system, many queries need to obtain semantic relationship from different tables. In BioDw, it needs to join many tables, while in TOTEM; the bidirectional path expression can avoid the time-consuming join operations. The Query3 is more complex than Query 2, so, our approach is more robust to support complex path query.

TPC Benchmark is an on-line transaction processing (OLTP) benchmark approved by TPC (Transaction Processing Performance Council) for testing the business biology application. We adopt TPC-C v2.1.0, which simulates a complete computing environment where many users execute transactions against biology databases.

Based on the semantic data management in figure 2, we designed some typical query examples as described below.

**Query 4**

Select all proteins with semantic similarity about HIV.

**Query 5**

Select all proteins with semantic similarity about HIV, and drugs correlated with proteins, and the name of drug is MT2.

**Query 6**

Select all proteins with semantic similarity about HIV, and genes correlated with proteins, and the name of gene is CCR5.

**Query 7**

Select all proteins with semantic similarity about HIV, and drugs and genes correlated with proteins, and the name of drug is ZDV.

**Query 8**

Select all proteins with semantic similarity about HIV, and drugs and genes correlated with proteins, and the name of gene is SDF.

In figure 7a, x-axis denotes five queries, and y-axis records the number of successful transactions executed in three hours of each query. In figure 7b, proportion = transactions in TOTEM divided by transactions in BioDW. We can get the conclusion that TOTEM works better than BioDw in almost every query. Although the two systems have similar efficiency on executing simple queries, Totem shows huge predominance when the query is complex and involves a lot of classes or relations, especially, in Query 7 and Query 8, the proportion is approximately 4. So, our path query is more robust and efficient than join operator in BioDw.

So we can come to the conclusion from the result of the above experiment that:

1 In the aspect of the storage space consumed, the performance of our system is close to that of the system on BioDw, which is common used in biology system.
2 When there are more classes in semantic search, both the response time and capability of transaction processing of our system are obviously excelled that of traditional biology database.

## CONCLUSION

In this paper, we have presented a semantic search framework which adopts the object deputy model to store semantic data of biological sources. In our framework, it is easy to semantic search in terms of bi-directional pointers between object and deputy objects, not only having good storage performance, but also improving the performance of semantic search. We believe that the presented approach could be easily adapted to various semantic data management solutions for semantic query and analysis. The experiment results show that our approach is more feasible and efficient than the traditional one.

## REFERENCES


**[1]** F.Bry, P.Kroger; 'A Computational Biology Database Digest: Data, Data Analysis, And Data Management', In: Distributed and Parallel Databases, Kluwer Academic Publishers Hingham, MA, USA, 7-42, January **(2003)**.

**[2]** G.D.Schuler, J.A.Epstein, H.Ohkawa, J.A.Kans; Methods Enzymol., **266**, 141-62 **(1996)**.

**[3]** E.M.Zdobnov, R.Lopez, R.Apweiler, T.Etzold; Bioinformatics, **18(8)**, 1149-1150 **(2002)**.

**[4]** B.Boeckmann, A.Bairoch, R.Apweiler, M.C.Blatter, A.Estreicher, E.Gasteiger, M.J.Martin; Nucleic Acids Res., **31(1)**, 365-370 **(2003)**.

**[5]** S.L.Cao, L.Qin; 'Applications of Gene Ontology in Bio data Warehouse', In: Proc.6th l Bio.Ontologies Meeting, Brisbane, Australia, 33-36 **(2003)**.

**[6]** M.Ashburner, C.A.Ball, J.A.Blake, D.Botstein, H.Butler, J.M.Cherry, A.P.Davis; Nature Genet., 25(1), 25-9 **(2000)**.

**[7]** Z.Peng, Y.Kambayashi; 'Deputy Mechanisms for Object-Oriented Databases', Proc. of IEEE 11th Int.Conf. on data Engineering, 333-40 **(1995)**.

**[8]** M.Francisco Couto, Mario J.Silva; Data & Knowledge Engineering, **9**, 54-78 **(2006)**.

**[9]** Zhiyong Peng, QingLi, IEEE Transaction on Data Knowledge and Engineering, **9**, 124-135 **(2005)**.

**[10]** Hong-Hai Do, Erhard Rahm; 'Flexible Integration of Molecular-Biological Annotation Data', The GenMapper Approach, EDBT, lncs., **2992**, 811-822 **(2004)**.

**[11]** B.MC, M.HN, S.G, M.GT; 'Spins: Standardized Protein NMR Storage', A data Dictionary and Object-Oriented Relational Database, October **(2002)**.

**[12]** CAO Shun-Liang, QIN Lei; Acta Biochim. Biophys.Sin., **36(5)**, 365-370 **(2004)**.

**[13]** B.Zhai, Y.Shi, Z.Peng; 'Object Deputy Database Language', In: The 4th International Conference on Creating, and Collaborating through Computing, January **(2006)**.

**[14]** H.Huang; 'Iproclass: An Integrated Database Of Protein Family Classification, Function And Structure Information', In Nucleic Acids Research, **31**, 390-392 **(2003)**.

**[15]** M.G.Hicks, C.Kettner; 'Amaze: A Database Of Molecular Function,Interactionsand Biochemical Processes', In: Proceedings of the Beilstein-Institut Workshop, May **(2002)**.

**[16]** J.T.Mieg, D.T.Mieg, L.Stein; 'Acedb: A Genome Database Management System', In: Computing in Science and Engineering, **1(3)**, 44-52, June **(1999)**.

**[17]** Ann-Ping Tsou, Yi Ming Sun; IEEE Transaction on Information Technology in Biomedicine, **10(3)**, 550-558 **(2006)**.