データベースシステム 107－4
（1996. 3. 11）

意味的な異種性をもつ自律的な分類データベースの統合化方式

北上 始，森 康真，有川正俊
広島市立大学
〒731-31 広島市安佐南区沼田町大塚 151-5
E-mail: { kitakami, mori, arikawa}@its.hiroshima-cu.ac.jp

著者らは意味的な異種性をもつ自律的な分類データベースに対する統合化方式を新しく開発した。統合化に先立って，利用者は各々の分類データベースに対する真実性の優先順位を指定する。統合化は分類データベースから核となるデータベースを選択し，残りの分類データベースを用いてその核データベースを漸増することによって達成されている。重要な点は，残りの分類データベースから核データベースにとって無矛盾なデータを探索することにある。それは，残りの分類データベースから協調性制約に違反するデータの排除を初めとして，保留データの中で解釈が曖昧な部分を排除することによって達成されている。著者らは，これらを関係代数演算による統合化アルゴリズムとして提案する。また，統合化アルゴリズムの有用性を示すために，この統合化方式を組み込んだシステムについて述べる。システムはSQLプログラミング方式で開発されている。さらに，統合データベースの容易な遠隔アクセスを利用者に提供するために，World-Wide Webを用いて，木構造上の簡易な巡行アクセスを許すようなウインドウインターフェースが実現されている。本システムは生物種の系統樹データベースの統合・維持・管理に利用されている。

An Integration Methodology for Autonomous Nomenclature Databases in Semantic Heterogeneity

Hajime Kitakami, Yasuma Mori and Masatoshi Arikawa

Faculty of Information Sciences
Hiroshima City University
151-5 Ozuka, Numata-Cho, Asa-Minami-Ku
Hiroshima-Shi 731-31, Japan
TEL & FAX: +81-82-830-1587
E-mail: {kitakami, mori, arikawa}@its.hiroshima-cu.ac.jp

Abstract

We developed a *new integration algorithm* for *autonomous nomenclature databases* in the presence of *semantic heterogeneity* over international computer networks. First, the algorithm requires users to assign priority according to order of belief for each one of the accessible nomenclature databases. Integration is achieved by increasing the kernel database, selected by the algorithm as the one with the highest priority among the databases. Complex problems are included to recognize *consistent parts* that can be incrementally added to the kernel database in the remainder of the accessible nomenclature databases through the use of tree structures. The algorithm with *relational operators*, was proposed after considering both *structural* and *cooperative constraints* useful in managing single and multiple databases. In addition, we present a nomenclature database system including the useful algorithm. The system is also accessible from remote users through *World-Wide Web* and is implemented in *SQL* programming and *CGI (Common Gateway Interface)* scripts of *World-Wide Web*. The system can be useful in both integrating and managing *biological taxonomy databases* over international computer networks.

1. Introduction

The amount of information handled by computers in various forms such as characters, numeric values, documents and images, has been increasing in an explosive manner since networks became fairly wide spread in the world. This increase of information makes it no easy matter to access databases. In order to save on labor, many database researchers try facilitate access by classifying information and constructing hierarchical structures. Other researchers do not try to manually classify large amounts of data but automatically do so using machine learning [1],[2].

We are focusing our attention on a methodology for constructing an integrated database without the inconsistencies which occur when there is semantic heterogeneity among loosely coupled databases storing nomenclature data. Many types [3],[4] of heterogeneity in multiple database systems can be divided into those due to differences in DBMSs and those due to differences in the semantics of the data. The former occurs when there is a mismatch in data models or system level support. The latter occurs when there is disagreement about the meaning, interpretation, or intended use of identical or related data. In general, inconsistencies occurring in the presence of semantic heterogeneity are resolved by either mapping relationships between any two databases or revising databases to create agreement. However, an on-going problem exists as described [5] below. The problem is that due to the large amounts of data stored within these databases, detection of all inconsistencies is an impossibly expensive proposition. Furthermore, even if all the inconsistencies were detected and resolved at a given time, future independent construction in loosely coupled environments can still introduce new inconsistencies. Hence, database constructors must continue to maintain consistency among these databases for long periods of time. However, it is impractical to resolve inconsistencies whenever database users access the integrated database over computer networks.

We have proposed an integration algorithm for loosely coupled nomenclature databases in the presence of semantic heterogeneity. We assume that each nomenclature database is represented by a binary relation and that naming conflicts in leaf nodes do not exist among these databases. In the integration algorithm, it is important to search for consistent parts of the nomenclature databases outside of the kernel database. This paper describes an integration algorithm including operators of *relational algebra* [6] for integrating these databases. Finally, we present the integrated database system including the useful integration algorithm. The system is also accessible from remote users using *World-Wide Web*. It is useful in both integrating and managing *biological taxonomy databases* [7],[8],[9] over international computer networks.

2. Example of the integration

There are two autonomous databases shown in Figure 1. The integrated database in general can be inconsistent within itself, if we construct one through the union of two databases. Each database constructor can resolve inconsistencies between the two databases using a method which performs either mapping or revision of data at a given time, but future independent construction in loosely coupled environments can still introduce new inconsistencies. Hence, we try to integrate two databases without using the method.

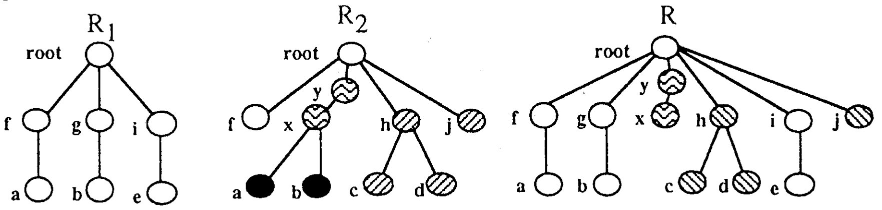


Figure 1. An example of nomenclature databases

The actual constructed trees have a height of 30 to 40. Because of limited space, we introduce small trees shown in Figure 1 as one example illustrative of the integration. Let us consider R_1*={ (a,f), (b,g), (e,i), (f,root), (g,root), (i,root), (root,NULL) }* and R_2*={ (a,x), (b,x), (c,h), (d,h), (f,root), (x,y), (y,root), (h,root), (j,root), (root,NULL) }*, where these relations, R_1 and R_2, are included in two databases, DB_1 and DB_2, respectively. Moreover, we assume that any tuples of database DB_1 are the most believable of these databases, DB_1 and DB_2 . The tuple, *(root,NULL)*, means that the root node does not have any parent node, where *NULL* represents missing information.

The relation consisting of all inconsistent tuples of R_2 can be represented by set *{ (a,x), (b,x) }*, and the relation consisting of all tuples appearing in both relations can be represented by set *{ (f,root), (root,NULL) }*. Let us consider *T={ (a,x), (b,x) }* $\cup$ *{ (f,root), (root,NULL) }*. Relation R'_2 consisting of all tuples appearing in the first and not the second of two relations, R_2 and *T*, can be represented by set *{ (c,h), (d,h), (x,y), (y,root), (h,root), (j,root) }*. Thus, it seems that we can obtain the integrated relation

R ($=R_1 \cup R_2'$) through the union of two relations, R_1 and R_2'. However, node x of the tuple, (x,y), should be distinguished from leaf nodes, c, d and j, in the integrated relation, since x is a nonleaf node in relation R_2 and has future ambiguous interpretations in resolving inconsistencies between two relations, R_1 and R_2. We call the nonleaf node, the pending node. It should be excluded from the integrated database when we integrate. Let us consider a tuple, $(y, root)$, included only in R_2, where y is not included in any domain of table R_1. If the difference set, $(R2 - R1)$, has at least one pending node among its child nodes, $(-,y)$, of node y, we in general can define the tuple, $(-,y)$, as a pending node, where the dash symbol "-" represents any node of the difference set.

We will describe the integration algorithm using operators of relational algebra in detail. We believe that our results can be adapted to other data models.

3. Nomenclature database

Tuples representing the tree structure are stored in the relation R_i, where R_i is one only element in the component database DB_i, $1 \leq i \leq n$. We assume that naming conflicts in leaf nodes do not exist among these databases. The tuple of node x stored in R_i is represented by (x,y) which is a pair of node x and its parent y, where both x and y are defined by the same domain D and the attribute stored with x is defined as the primary key of the relation $R(x,y)$.

Let $R=\{ (a,b) \mid a\in D, b\in D\}$ and $S=\{ (b,c) \mid b\in D, c\in D\}$ stand for two binary relations. A forward search for S and R, denoted as $S\Delta R$, is defined as $\{ (b,c) \mid \exists b\ ((a,b)\in R, (b,c)\in S) \}$. In opposition to this, a backward search for S and R, denoted as $S\nabla R$, is defined as $\{ (c,a) \mid \exists a\ ((a,b)\in R, (c,a)\in S) \}$.

3-1 Tree searches

Let $T_0 \subseteq R$, $T_1 = T_0$, $T_i = T\Delta T_{i-1}$, $1 \leq i \leq n$, $T_n = \phi$. The search for R and T_0 is defined as $R^* = \cup_{i>0} T_i$. We call it a recursive forward search. The search can locate all nodes appearing in the optimal path from each node stored in T_0 to the root node of R. Let N be the upper bound in the recursive forward search [10]. The search is represented by the following algorithm:

```
$R^* := R_0$ ; $R' := R_0$ ; $i := 0$ ;
while ( $R' \neq \phi$ or $i < N$ ) do
        $i := i + 1$ ;
        $R' := R \ \Delta \ R'$ ;
        $R^* := R^* \cup ( R' - R^*)$ ;
end ;
```

We define this search processing as $R\Delta^N R_0$. All tuples appearing in the results of $R\Delta^\infty R_0$ are represented by $lineage(R, R_0)$. Let us consider the processing using the example shown in Figure 1. $lineage(R, R_0) = \{ (c,h), (h,root), (root,NULL) \}$, where $R_0 = \{ (c,h) \}$.

Let $T_0 \subseteq R$, $T_1 = T_0$, $T_i = T \nabla T_{i-1}$, $1 \leq i \leq n$, $T_n = \phi$ using the previous procedure. The search for R and T_0 is defined as $R^* = \cup_{i>0} T_i$. We call it a recursive backward search. The search can find whole progeny from each node stored in R_0 to the root node of R. Let M be the lower bound in the recursive backward search. The search is represented by the following algorithm:

```
$R^* := R_0$ ; $R' := R_0$ ; $i := 0$ ;
while ( $R' \neq \phi$ or $i < M$ ) do
        $i := i + 1$ ;
        $R' := R \ \nabla \ R'$ ;
        $R^* := R^* \cup ( R' - R^*)$ ;
end ;
```

We define this search process as $R\nabla^M R_0$. All tuples appearing in the results of $R\nabla^\infty R_0$ are represented by $subtree(R, R_0)$. Moreover, we define all leaf nodes included in $subtree(R, R_0)$ as set, $leaf(R, R_0)$. It is clear that $leaf(R, R_0) = \{ (x,y) \mid (x,y)\in subtree(R, R_0), \phi = \{ \{(x,y)\} \ \nabla subtree(R,R0) \} \}$. Let us consider the process using the example shown in Figure 1. $subtree(R, R_0) = \{ (f,root), (a,f), (h, root), (c,h), (d,h) \}$, where $R_0 = \{ (f,root), (h,root) \}$ and $leaf(R, R_0) = \{ (a,f), (c,h), (d,h) \}$.

3-2 Integrity constraints

We can define two types of integrity constraints [9] in these multiple databases. One is useful in consistently managing each database itself and other is useful in managing multiple databases. The former is called a structural constraint, which can detect any abnormal subtrees and/or nodes. The latter is called a cooperative constraint, which can detect inconsistent nodes among these databases. We regard semantic heterogeneity as a violation of the cooperative constraint in this paper.

The structural constraints for each database are useful in maintaining the topological structure of the database. The constraints include not only key constraints related to both (C1) missing information and (C2) data duplication, but also specific constraints, (C3) and (C4), which prohibit storing data with missing information except for the root tuple and prohibits data with an invalid node name in the second attribute of the binary relation. These structural constraints, (C1)-(C4), can be represented by the following rules:

(C1) $inconsistent \text{ <- } X = NULL,\ R_i(X,Y).$

(C2) $inconsistent \text{ <- } X_1 \neq NULL,\ X_1 = X_2,\ R_i(X_1,Y_1),\ (X_2,Y_2) \in (\ R_i - \{\ (X_1,Y_1)\ \}\ ).$

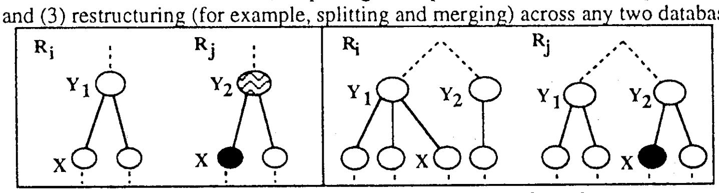
(C3) $inconsistent \text{ <- } X \neq \text{"root"},\ Y = NULL,\ R_i(X,Y).$

(C4) $inconsistent \text{ <- } X \neq NULL,\ Y \neq NULL,\ R_i(X,Y),\ \neg R_i(Y,Z).$

The cooperative constraints represent semantic mismatches between any two databases, R_i and R_j. Figure 2 shows that the mismatches are caused by differences in node name except for leaf nodes and topological structure between these databases. The system can detect mismatches in a way that proves the inequal relation, $Y_1 \neq Y_2$, for any two tuples, $(X,Y_1) \in R_i$ and $(X,Y_2) \in R_j$. The constraint can be represented as follows:

(C5) $inconsistent \text{ <- } Y_1 \neq Y_2,\ R_i(X,Y_1),\ R_j(X,Y_2).$

This constraint can detect differences in (1) spelling, (2) up-and-down relationship between parent and child nodes, and (3) restructuring (for example, splitting and merging) across any two databases.



(a) difference of node name (b) difference of topology

Figure 2. Patterns of mismatch between two databases

4. Integration methods

Any user can automatically construct his integrated database using the proposed algorithm in this section whenever he wishes to access one. We assume that each user gives priority related to their order of belief in these databases before integration. Let the set of databases be $\Omega = \{\ DB_1,\ DB_2,\ ....,\ DB_n\ \}$, where each database DB_i is only stored with the relation R_i. It is stored with a large amount of tuples which are represented by the binary relation (x,y). The order of $DB_1,\ DB_2,\ .....,\ DB_n$ means the priority related to the order of belief in these databases, so that R_1 contains the most believable tuples. Let DB_1 and DB_2 be inconsistent with respect to each other. All the inconsistencies cause false tuples in R_2. We represent the integrated database as a set, *IntegratedDB*, and the i-th kernel database as a set, $KernelDB_i$, $1 \leq i \leq n$. Each kernel database, $KernelDB_i$, includes only one temporal relation, $KernelR_i$, to be iteratively defined, where $KernelDB_1 = DB_1$ and $IntegratedDB = KernelDB_n$. Moreover, let the structural constraints for each nomenclature database, DB_i, be $SICs_i$ and the cooperative constraint between two databases, $KernelDB_i$ and DB_j, be $CICs_j$.

4-1 General framework

The 2nd kernel database, $KernelDB_2$, is defined by both $KernelDB_1$ ($=DB_1$) and DB_2, so that the (i+1)-th kernel database, $KernelDB_{i+1}$, in general can be defined by both $KernelDB_i$ and DB_{i+1}, $1 \leq i \leq n-1$. The (i+1)-th kernel relation, $KernelR_{i+1}$, is represented as the union of $KernelR_i$ and $(R_{i+1} - KernelRi\ )$, $1 \leq i \leq n-1$. It is important to resolve [9] inconsistencies related to both the structural and cooperative constraints before this iterative processing; as a result, we can obtain the following integration algorithm:

```
(I1)    $\Omega := \{\ DB_1, DB_2,\ ...,\ DB_n\ \}$ ; $KernelDB_1 := DB_1$ ;
(I2)    Resolve inconsistencies of $R_i$ , if $DB_i \models SICs_i$ , $1 \leq i \leq n$ ; $i := 0$ ;
        while ( $i < n$ ) do
            $i := i + 1$ ;
(I3)        Resolve inconsistencies of $R_{i+1}$, if ( $KernelDB_i \cup DB_{i+1}$ ) $\models CICs_{i+1}$ ;
(I4)        $KernelR_{i+1} := KernelR_i \cup ( R_{i+1} - KernelR_i )$ ;
        end;
(I5)    $IntegratedR := KernelR_n$ ;
```

All the inconsistencies of (I2) and (I3) can be detected and resolved by database constructors of each site at a given time, but future independent construction in loosely coupled environments can still introduce new inconsistencies of (I3). Hence, database constructors must continue to maintain consistency among these databases for long periods of time. This means that complete maintenance of all the database inconsistencies of (I3) is an impossibly expensive proposition due to the large amount of data, before every integration. We propose a practical integrated algorithm to resolve these inconsistencies. The practical integration algorithm is tremendously useful for users who need to acquire consistent nomenclature data.

4-2 Difference set

Let $KernelDB_i$ and DB_{i+1} each be consistent within itself and both be inconsistent with each other. The difference set $\{ R_{i+1} - KernelR_i \}$ of (I4) consists of (1) inconsistent set violating the cooperative constraint, (2) pending set stated in section 2, and (3) independent set. Both (1) and (2) are detected by the cooperative constraint between two databases, $KernelDB_i$ and DB_{i+1} but (3) do not have any relationship with the cooperative constraint. We should avoid incrementally adding not only all inconsistent tuples of (1) but also ambiguous parts of (2) to the kernel database. Let us consider the example shown in Figure 1. The inconsistent set of (1) is $\{ (a,x), (b,x) \}$, the pending tuple of (2) is $\{ (x,y), (y,root) \}$, and the independent set of (3) is $\{ (c,h), (d,h), (h,root), (j,root) \}$. Inconsistent tuples, (a,x) and (b,x), in general should not be added to the kernel database. Both tuples, (x,y) and $(y,root)$, should be excluded, since they include ambiguous interpretation before resolving inconsistencies. The independent tuples, (c,h), (d,h), $(h,root)$ and $(j,root)$, should be merged with the kernel database ($=R_1$), since the union set of $R1$ and the independent tuples do not violate the structural constraints.

The kernel relation, *KernelRi*, includes the most believable tuples in both $KernelDB_i$ and DB_{i+1}, when both are inconsistent with each other. Hence, the inconsistent set, $Error_{i+1}$, of (1) including the difference set, $(R_{i+1} - KernelR_i)$, can be represented as follows:

$Error_{i+1} := \{ (x,y) \mid (x,y) \in (R_{i+1} - KernelR_i), (x,z) \in KernelR_i, z \neq y \}$;

The pending set, $Pending_{i+1}$, can be represented as follows:

$Pending_{i+1} := \{ (u,v) \mid (u,v) \in (lineage((R_{i+1} - KernelR_i),\{ (x,y) \}) - \{ (x,y) \}), (x,y) \in Error_{i+1} \}$;

The independent set includes neither inconsistent tuples nor pending tuples, so that it is represented as follows: $Independent_{i+1} := (((R_{i+1} - KernelR_i) - Error_{i+1}) - Pending_{i+1})$;

It is clear that any leaf node of $Independent_{i+1}$ is included in the original leaf set, $leaf((root,NULL),Ri+1)$.

4-3 Practical integration

Let us remove the processing (I3) from the integration algorithm of the previous section 4-2 to achieve practical integration. Inconsistent nodes which violate the structural constraint (C2) then appear in the difference set, $(R_{i+1} - KernelR_i)$, of (I4), so that we should remove the inconsistent set $Error_{i+1}$ and ambiguous parts of the pending set $Pending_{i+1}$ from the difference set to make the (i+1)-th kernel database. It is important to clarify the relationship among the inconsistent set, pending set, and independent set in the difference set, $(R_{i+1} - KernelR_i)$. Figure 3 illustrates typical connection patterns related to the topological structure among these sets.

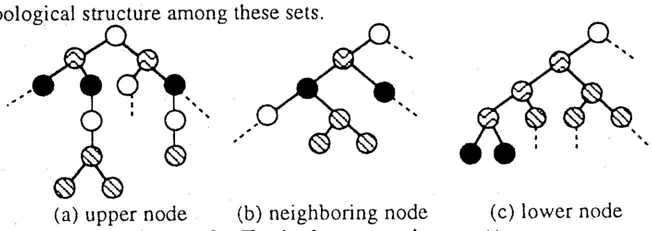


Figure 3. Typical connection pattern

The white circles shown in the figure mean shared nodes between $Ri+1$ and $KernelRi$, and disappear in the difference set, $(R_{i+1} - KernelR_i)$. The black circles, the wavy line circles and the oblique line circles are inconsistent tuples, pending tuples and independent tuples, respectively.

In Figure 3, we should remove the inconsistent tuples from the difference set to make the (i+1)-th kernel database. It is possible to use the independent nodes shown in both (a) and (b) of the figure to make the (i+1)-th kernel database. Whenever the independent nodes shown in (b) of the figure are utilized for making the (i+1)-th kernel database, the kernel database is still consistent with (C4) of the structural integrity constraints. Moreover, it is possible to use the independent parts shown in (c) of the figure to make the (i+1)-th kernel one, since we can consistently connect the parts to the i-th kernel one, using two upper pending nodes shown in (c) of the figure. The other pending node shown in (c) of the figure should be removed from the difference set.

Let all the consistent nodes Add_{i+1} be named the additional set which can be added to the i-th kernel one. The practical integration algorithm is as follows:

```
(PI1) $\Omega := DB1, DB2, ..., DBn$ ; $KernelDB1 := DB1$ ;
(PI2)   Resolve inconsistencies of $R_i$, if $DB_i \models SICs_i$ , $1 \le i \le n$ ; $i := 0$ ;
        while ( $i < n$ ) do
                $i := i + 1$ ;
(PI3)           $Temp := ( R_{i+1} - KernelR_i) - Error_{i+1}$ ;
(PI4)           $Add_{i+1} := Temp - (Pending_{i+1} - cutting(Pending_{i+1}, Temp))$ ;
(PI5)           $KernelR_{i+1} := KernelR_i \cup Add_{i+1}$ ;
        end;
(PI6) $IntegratedR := KernelR_n$ ;
```

where the set, *Temp*, means the difference set between $(R_{i+1} - KernelR_i)$ and the inconsistent set. The inconsistent set, $Error_{i+1}$, and the pending set, $Pending_{i+1}$, appear in both (PI3) and (PI4) of the algorithm as previously defined. The processing which is cutting, $cutting(Pending_{i+1}, Temp)$, of (PI4) searches for the consistent parts of the pending nodes, $Pending_{i+1}$, where the set, *Temp*, includes the consistent parts of the independent nodes. If we can detect pending nodes which appear when searching all lineages for consistent parts, they (the pending nodes shown in (c) of the figure 3) can clearly be utilized to make the (i+1)-th kernel database. The processing is represented as follows:

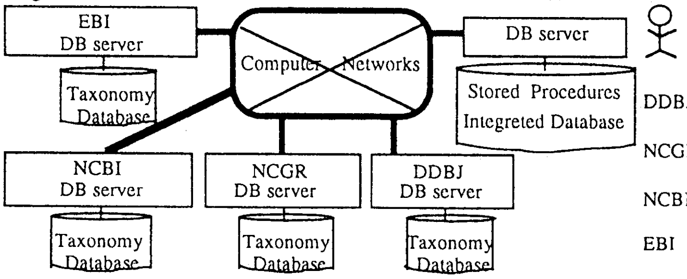
$cutting(Pending_{i+1}, Temp) := lineage(Pending_{i+1}, (Pending_{i+1} \Delta (Temp - Pending i+1)))$;

Let $|R|$ be the cardinality of the set R. The processing of (SI4) prefers being computed on the condition, $|Error_{i+1}| << |leaf(R_{i+1}, \{root\}) - KernelR_i|$, rather than $|Error_{i+1}| >> |leaf(R_{i+1}, \{root\}) - KernelR_i|$, since the pending set, $Pending_{i+1}$ is a function of the inconsistent set, $Error_{i+1}$. If $|Error_{i+1}| >> |leaf(R_{i+1}, \{root\}) - KernelR_i|$, the following processing is useful for the integration:

$Add_{i+1} := lineage(Temp, \{leaf(R_{i+1}, \{root\}) - KernelR_i\})$;

5. System overview

Figure 4 shows the system configuration of the nomenclature database system including the practical integration algorithm with the relational data model. The system integrates and manages the continuously enlarging biological taxonomy databases which have a total of about 100,000 tuples. The height of the taxonomic tree stored in them is about 30 to 40.



DDBJ : DNA Data Bank of Japan (Shizuoka, Japan)
NCGR:National Center of Genome Resources (New Mexico, USA)
NCBI :National Center for Biotechnology Information (Maryland, USA)
EBI :European Bioinformatics Institute (Cambridge, UK)

Figure 4. The nomenclature database system

The taxonomy databases are biological dictionaries and are useful in constructing DNA databases at international DNA data banks [11],[12],[13],[14],[15]. The international DNA data banks are organized by DDBJ (DNA Data Bank of Japan; National Institute of Genetics), NCBI (National Center for Biotechnology Information) and EBI (European Bioinformatics Institute). Each data bank collaborates with the other two data banks in many areas through mutual exchanges of data over international computer networks. However, each taxonomy database is independently constructed, because of rapid changes resulting from biological advancements. As a result, these taxonomy databases are respectively autonomous and have semantic heterogeneity. The cooperative constraint implemented in *SQL* programming actually detected about 2,000 inconsistent tuples between any two taxonomy databases [9]. It is tremendously helpful for not only the staff of the data banks but also biologists involved in genome research to have a system which provides them access to consistent tuples as much as possible [16].

5-1 Access methods

We newly implemented not only the practical integration algorithm but also powerful tree search algorithms in both *SQL* and *Control Flow Language* of the relational database system, *SYBASE*, and also stored the *SQL* program in the integrated database. The program code is about 15,000 lines in size including other useful mechanisms. The powerful tree search methods can make visible to users, a structure which spans both up-and-down and left-and-right for a given node. The methods have several functions, such as $R\Delta^{\infty} R_0$ (=*lineage(R, R_0)*), $R\nabla R_0$, $R\nabla^{\infty} R_0$ (=*subtree(R, R_0)*), *and* $R\nabla(R\Delta^{\infty} R_0)$. We can define the neighborhood area inferred from any node using the powerful tree search methods. The neighborhood area can cover several nodes made visible using the methods. If we change the given node to one of other nodes in the area, we can find the new area defined by the change. If we repeatedly apply the methods, we can move to any area of the tree structure shown in Figure 5. Users can also search many documents, 1,500 images and DNA database in the integrated taxonomy database.

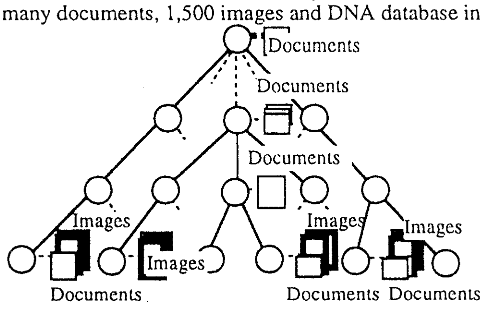


Figure 5. The taxonomy database

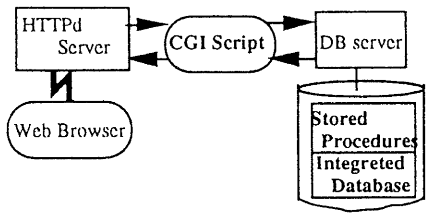
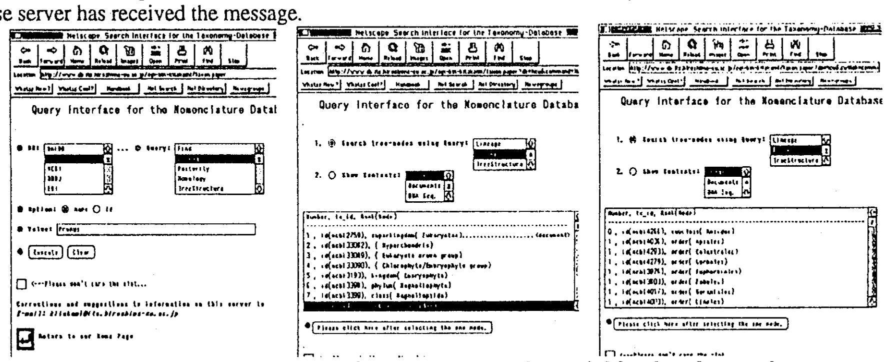


Figure 6. Interfacing between the WWW and database servers

5-2 WWW interface

Moreover, we also developed a tool to interface between the taxonomy database system and *World-Wide Web* [17],[18] to allow remote access for the integrated database over the computer networks. The tool was implemented in *CGI (Common Gateway Interface)* [19] script and the program code was about 1,000 lines in size. Figure 6 shows the system framework of the tool. If users access the integrated database using *Web Browser* [20],[21], it sends a message including the access method to the database server. The method represented by the stored procedure [22] is executed on the database system, after the database server has received the message.


Netscape: Search Interface for the Taxonomy-Database
Back
Forward
Home
Reload
Images
Open
Print
Find
Stop
Query Interface for the Nomonclature Database
Query:
Lineage
TreeStructure
Execute
Clear
Documents
DNA Seq.
Please click here after selecting the one node.
Return to our Home Page

```
Number, tx_id, Rank(Node)
1 , id(ncbi2759), superkingdom( Eukaryotae)...................(document)
2 , id(ncbi33082), ( Hyperchondria)
3 , id(ncbi33089), ( Eukaryote crown group)
4 , id(ncbi33090), ( Chlorophyta/Embryophyta group)
5 , id(ncbi3193), kingdom( Embryophyta)
6 , id(ncbi3398), phylum( Magnoliophyta)
7 , id(ncbi3398), class( Magnoliopsida)
```

```
Number, tx_id, Rank(Node)
0 , id(ncbi4261), subclass( Rosidae)
1 , id(ncbi4036), order( Apiales)
1 , id(ncbi4293), order( Celastrales)
1 , id(ncbi4279), order( Cornales)
1 , id(ncbi3976), order( Euphorbiales)
1 , id(ncbi3803), order( Fabales)
1 , id(ncbi4017), order( Geraniales)
1 , id(ncbi4013), order( Linales)
```

Figure 7. Window Interface to repeat the neighborhood searches

The tool has an effective window interface in which the query and result windows are the same form, so that it provides users with an easy interface for use in repeatedly applying the powerful tree search methods. The left window shown in Figure 7 illustrates the initial window to infer a neighborhood area from a given node. We can see the center window after clicking the submit button. It illustrates the execution results of the inference and the results are shown in the menu of the window. The next query condition can be given by selecting a focused node in the menu. We can see the right window after clicking the submit button. Thus, we can easily move to any area for the tree structure using the window interface. Access to the taxonomy database system is available from our *WWW* site (http://www.its.hiroshima-cu.ac.jp/~kitakami/treedb.html). The database service using the system is still in the development stage, and access is free of charge for a trial period.

6. Conclusions

This study was motivated by the fact that the nomenclature databases are constructed in loosely coupled environments, so that it is impractical to resolve inconsistencies whenever database users access the integrated database over computer networks. We proposed a practical integrated algorithm to consistently integrate the databases without resolving inconsistencies. We assumed that the databases do not include naming conflicts in leaf nodes. First, we had a user assign priority concerning belief in accessible nomenclature databases before integration. Integration was achieved by incrementally increasing the number of tuples in the kernel database after the algorithm had selected one kernel database from the databases. It was important to exclude inconsistent tuples related to cooperative and structural constraints in the algorithm. The algorithm was represented by such relational operators as join, difference and union. In particular, the previous constraints, (C2), (C4) and (C5), in section 3-2 were useful in proposing the practical integration algorithm. Finally, we introduced the nomenclature database system including the practical integration algorithm. The system is accessible from remote users using World-Wide Web. We also explained that the system is useful in both integrating and managing the continuously expanding biological taxonomy databases over international computer networks.

Acknowledgments

We wish to thank the staff of the international DNA data banks for their help in automatically obtaining these taxonomy databases with the relational model.

References

[1] J. R. Quinlan: Introduction of Decision Trees, Machine Learning, Vol. I, pp.81-106 (1986).

[2] Knowledge Discovery in Database: An Attribute-Oriented Approach, Proceedings the 18th VLDB Conference, Morgan Kaufman Publishers, Inc., pp.547-559 (1995).

[3] Amit P. Sheth and James A. Larson: Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases, ACM Computing Surveys, Vol.22, No.3, pp.183-283 (1990).

[4] Evaggelia Pitoura, Omran Bukhres, and Ahmed Elmagarmid: Object Orientation in Multidatabase Systems, ACM Computing Surveys, Vol.27, No.2, pp.141-195 (1995).

[5] Shailesh Agarwal, Arthur M. Keller, Gio Wiederhold, and Krishna Saraswat: Flexible Relation: An Approach for Integrating Data from Multiple, Possibly Inconsistent Databases, Proc. of the 11 International Conference on Data Engineering, Taiwan, pp.495-504 (1995).

[6] C. J. Date: An Introduction to Database Systems, The System Programming Series, Addison Wesley, ISBN 0-201-54329-X, (1995)

[7] Scott Federhen: TaxMan User's Manual, (1994), Available at ftp://ncbi.nlm.nih.gov/repository/taxonomies/taxman/manual.ps

[8] Tim Clark, Cynthia Chung, and X. Yu: The NCBI Taxonomy Database, NCBI Technical Report, (1993).

[9] Hajime Kitakami, Yoshio Tateno, and Takashi Gojobori: Toward Unification of Taxonomy Databases in a Distributed Computer Environment, Proceedings of the Second International Conference on Intelligent System for Molecular Biology, Stanford University, AAAI Press, pp. 227-235 (1994).

[10] Francois Bancilhon and Raghu Ramakrishnan: An Amateur's Introduction to the Recursive Query Processing Strategies, Proceedings of the ACM SIGMOD '86, Washington D.C., pp.16-52 (1986).

[11] Desmond G. Higgins, Rainer Fuchs, Peter J. Stoeher and Graham N.Cameron: The EMBL Data Library, Nucleic Acids Research, Vol.20, Oxford University Press, pp.2071-2074 (1992).

[12] Christian Burks, Michael J. Cinkosky,William M. Fischer, Paul Gilna, Jamie E.-D. Hayden, Gifford M. Keen, Michael Kelly, David Kristofferson and Julie Lawrence: GenBank, Nucleic Acids Research, Vol.20, Oxford University Press, pp.2065-2069 (1992).

[13] Leslie Roberts: Research News,Managing the Genome Data Deluge, Science,Vol.262, No.22, pp.502-505 (1993)

[14] Hajime Kitakami, Tadasu Shin-I, Kazuo Ikeo et al: YAMATO and ASUKA: DNA Database Management System,Proceedings of the 28th Annual Hawaii International Conference on System Siencies, IEEE Computer Society Press, Vol.5, pp. 72-80 (1995).

[15] Hajime Kitakami, Yukiko Yamazaki, Kazuo Ikeo et al.: Building and Search System for a Large-Scale DNA database, Frontiers in Artificial Intelligence and Applications, Advances in Molecular Bioinformatics, Vol.22, IOS Press, pp.123-138 (1994).

[16] NCBI News, National Center for Biotechnology Information, National Library of Medicine, National Institute of Health, pp.1-8, September (1995).

[17] T. Berners-Lee, R. Cailliau, J. Groff and B. Pollermann: World Wide Web: The Information Universe, Electronic Networking: Research, Applications and Policy, 2(1), Meckler Publications, Westport CT, pp.52-58 (1992), Available in PostScript at ftp://info.cern.ch/pub/www/doc/ENRAP_9202.ps

[18] T. Berners-Lee, R. Cailliau, A. Luotonen, H. F. Nielsen and A. Secret: The World Wide Web, CACM, Vol. 37, No. 8, (1994).

[19] Rob McCool: National Center for Supercomputing Applications, University of Illinois at Urbana-Champaign, Common Gateway Interface Overview, Work in progress, Available at http://hoohoo.ncsa.uiuc.edu/cgi/overview.html

[20] National Center for Supercomputing Applications, University of Illinois at Urbana-Champaign, NCSA Mosaic, A WWW Browser, Work in progress, Available at http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/Docs/help-about.html

[21] Netscape Communications Corporation, Netscape Navigator, A WWW Browser, Work in progress, Available at http://www.netscape.com/comprod/netscape_nav.html

[22] Prabhat K. Andleigh and Michael R. Gretzinger: Distributed Object-Oriented Data-Systems Design, Prentice Hall Inc. (1992).