Advances in Molecular Bioinformatics

Edited by

S. Schulze-Kremer

1994

IOS Press

Amsterdam • Oxford • Washington DC

Ohmsha

Tokyo • Osaka • Kyoto

ISBN 90 5199 172 X (IOS Press)
ISBN 4-274-90001-0 (Ohmsha)
Library of Congress Catalogue Card Number: 94-077312

Publisher:
IOS Press
Van Diemenstraat 94
1013 CN Amsterdam
Netherlands

Sole distributor in the UK and Ireland:
IOS Press/Lavis Marketing
73 Lime Walk
Headington
Oxford OX3 7 AD
England

Distributor in the USA and Canada:
IOS Press, Inc.
P.O. Box 10558
Burke, VA 22009-0558
U.S.A.

Distributor in Japan:
Ohmsha, Ltd.
3-1 Kanda Nishiki - Cho
Chiyoda - Ku
Tokyo 101, Japan

PRINTED IN THE NETHERLANDS

Advances in Molecular Bioinformatics
S. Schulze-Kremer (Ed.)
IOS Press, 1994

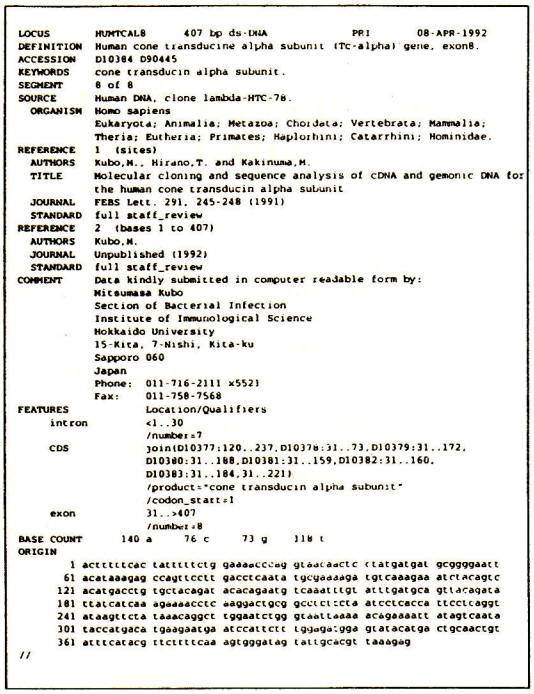
Building and Search System for a Large-Scale DNA Database

Hajime KITAKAMI*, Yukiko YAMAZAKI**, Kazuho IKEO**, Yoshihiro UGAWA***, Tadasu SHIN-I**, Naruya SAITOU**, Takashi GOJOBORI**, and Yoshio TATENO**
DNA Databank of Japan (DDBJ)
** Hiroshima City University*
151-5 Ootsuka, Numata-Chou, Asa-Minami-Ku, Hiroshima-Shi, Hiroshima-Ken 733, Japan
*** National Institute of Genetics*
1111 Yata, Mishima-Shi, Shizuoka-Ken 411, Japan
**** National Institute of Agrobiological Resources*
2-1-2 Kan-non-dai, Tsukuba-Shi, Ibaraki-Ken 305, Japan

Abstract. A flat-file system is inadequate for building, integrating and searching a large-scale database in which the number of entries is increasing in an explosive manner. The DNA database of DDBJ (DNA Data Bank of Japan) was built in a flat-file system until 2 years ago. GenBank in the United States helped DDBJ to convert the flat-file data constructed by DDBJ to a relational format and to install the Annotator's Workbench, AWB. AWB is a tool for building the DNA database with the so-called GenBank-schema based on a relational database management system, SYBASE. However, AWB with the GenBank-schema does not have a simultaneous processing function for a large amounts of DNA data such as Expressed Sequence Tags (ESTs). In addition, both integrating and searching were not carried out in the relational database at DDBJ. Recently, we newly developed a hierarchical relational schema for effectively building, integrating, and searching the DNA database on the relational database management system, SYBASE, at DDBJ. The schema is named the "DDBJ-schema". The schema allowed us to implement window interfaces to easily build the DNA database. The DNA database built in the GenBank-schema using AWB is converted into the relational format of the DDBJ-schema using a restructuring tool at DDBJ. We proposed a structured SQL-programming method to implement the restructuring tool and so on. The method is developed through the "view" function and the control flow language (CFL) of SYBASE. Moreover, we proposed two methodologies by which we can execute these software tools on UNIX based workstations connected by a computer network.

1. Introduction

DNA (Deoxyribonucleic acid) is genetic information for all organisms except RNA viruses. DNA is kinds of nucleotides designated by A,T,G, and C. The DNA sequence data and its related information can be stored in the form of a DNA database which consists of nucleotide



```
LOCUS       HUMTCALB      407 bp ds-DNA              PRI       08-APR-1992
DEFINITION  Human cone transducine alpha subunit (Tc-alpha) gene, exon8.
ACCESSION   D10384 D90445
KEYWORDS    cone transducin alpha subunit.
SEGMENT     8 of 8
SOURCE      Human DNA, clone lambda-HTC-78.
  ORGANISM  Homo sapiens
            Eukaryota; Animalia; Metazoa; Chordata; Vertebrata; Mammalia;
            Theria; Eutheria; Primates; Haplorhini; Catarrhini; Hominidae.
REFERENCE   1  (sites)
  AUTHORS   Kubo,M., Hirano,T. and Kakinuma,M.
  TITLE     Molecular cloning and sequence analysis of cDNA and genomic DNA for
            the human cone transducin alpha subunit
  JOURNAL   FEBS Lett. 291, 245-248 (1991)
  STANDARD  full staff_review
REFERENCE   2  (bases 1 to 407)
  AUTHORS   Kubo,M.
  JOURNAL   Unpublished (1992)
  STANDARD  full staff_review
COMMENT     Data kindly submitted in computer readable form by:
            Mitsumasa Kubo
            Section of Bacterial Infection
            Institute of Immunological Science
            Hokkaido University
            15-Kita, 7-Nishi, Kita-ku
            Sapporo 060
            Japan
            Phone:  011-716-2111 x5523
            Fax:    011-758-7568
FEATURES             Location/Qualifiers
     intron          <1..30
                     /number=7
     CDS             join(D10377:120..237,D10378:31..73,D10379:31..172,
                     D10380:31..188,D10381:31..159,D10382:31..160,
                     D10383:31..184,31..221)
                     /product="cone transducin alpha subunit"
                     /codon_start=1
     exon            31..>407
                     /number=8
BASE COUNT      140 a      76 c      73 g     118 t
ORIGIN
        1 acttttttcac tatttttctg gaaaaccCag gtaacaactc ctatgatgat gcggggaatt
       61 acataaagag ccagttcctt gacctcaata tgcgaaaaga tgtcaaagaa atctacagtc
      121 acatgacctg tgctacagat acacagaatg tcaaatttgt atttgatgca gttacagata
      181 ttatcatcaa agaaaacctc aaggactgcg gcctttctta atcctcacca ttcctcaggt
      241 ataagttcta taaacaggct tggaatctgg gtaattaaaa acagaaaatt atagtcaata
      301 taccatgaca tgaagaatga atccattctt tggagatgga gtatacatga ctgcaactgt
      361 atttcatacg ttctttttcaa agtgggatag tattgcacgt taaagag
//
```

Figure 1. An example of the flat-file format

(DNA and RNA) sequences and related attributes which are bibliographic and biological information. Each nucleotide sequence data is represented as long strings of A's, T's, C's, and G's, along with associated biological annotation.

The DNA Data Bank of Japan (DDBJ) is one of the International Nucleotide Sequence Databanks where the EMBL (European Molecular Biology Laboratories) Data Library [1,2] and NCBI (National Center for Biotechnology Information) /LANL (Los Alamos National Laboratories) [3,4,5] are the European and American representatives, respectively. The American DNA databank is called GenBank. We have been collaborating with these two databanks in many areas through mutual exchanges of data over the international computer network. The mutual exchange of data on a daily basis is realized between DDBJ and the collaborative data banks using a flat-file format (DDBJ/EMBL/GenBank Formats). An example of the flat-file format is shown in Figure 1.

The DNA database of DDBJ was built in a flat-file system until 2 years ago. The flat-file system has been represented by data description rules or a grammar [6] shown in Appendix-1. The flat file system is not adequate for building, integrating, and searching a large-scale DNA database whose entries are continuously increasing in such an explosive manner as is occurring today. GenBank helped DDBJ to convert the flat-file data constructed by DDBJ staff, to a relational format and to install the Annotator's Workbench, AWB. AWB is a tool for building the DNA database with the GenBank-schema [7] based on the relational database management system, SYBASE [8]. However, AWB with the GenBank-schema is not adequate for

simultaneous processing large amounts of DNA data such as Expressed Sequence Tags (ESTs) [9]. In addition, both integrating and searching are not carried out in the relational database at DDBJ. The number of data submissions has increased 3 to 4 times in the past 2 years at DDBJ. In addition, the number of on-line users has also increased to 4 to 5 times compared with that of 2 years ago, making it difficult for the staff to process the data efficiently.

This paper will present methodologies for solving those problems using the relational database system, SYBASE, in the UNIX environment. We newly developed a relational schema, DDBJ-schema, to effectively build, integrate, and search the DNA database on the relational database management system at DDBJ. The DNA database built with the GenBank-schema using AWB is converted into the relational format of the DDBJ-schema using a restructuring tool [10,11] at DDBJ. The restructuring tool is implemented in the "view" function, and the control flow language (CFL) of SYBASE. These software tools can be executed on UNIX based workstations connected by a computer network.

2. Current System

Figure 2 shows the system configuration for building the DNA Database on the UNIX based workstation at DDBJ. AWB and the GenBank-schema are software tools developed by LANL in GenBank. DDBJ's Reviewers check the form and content of a submitted entry sent by an author. Reviewers extract the necessary information such as taxonomy, keywords, features etc., from the submitted entry and complete the entry form according to the internal (domestic) and international rules with these tools on the UNIX based workstation.

We developed a program to simultaneously process multiple entries at DDBJ, because AWB does not have a simultaneous processing function for a large amounts of DNA data. Since early spring 1993, we have been successfully able to process, using this program, many kinds of multiple entries sent by the Human Genome Project [9].

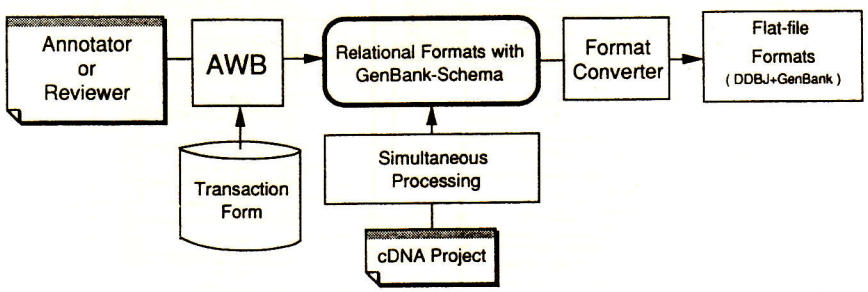


Figure 2. Current System

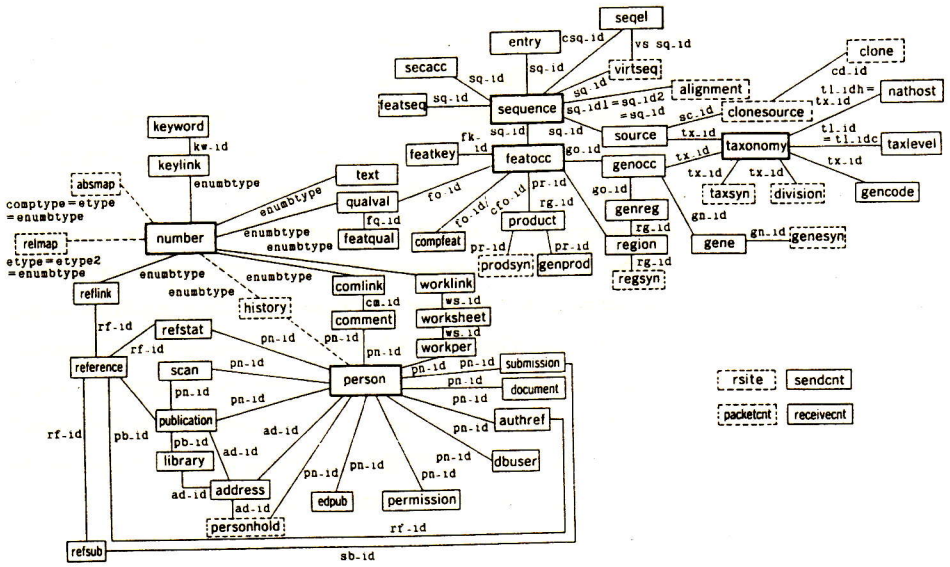


Figure 3. GenBank-schema

2.1 Relational Schema

The GenBank-schema [7] shown in Figure 3 has about 60 tables each of which is represented by a rectangle in the Figure. One entry is partitioned into smaller entries and such entries are stored according to the tables. These tables can be joined to each other by the join column in the complex data structure.

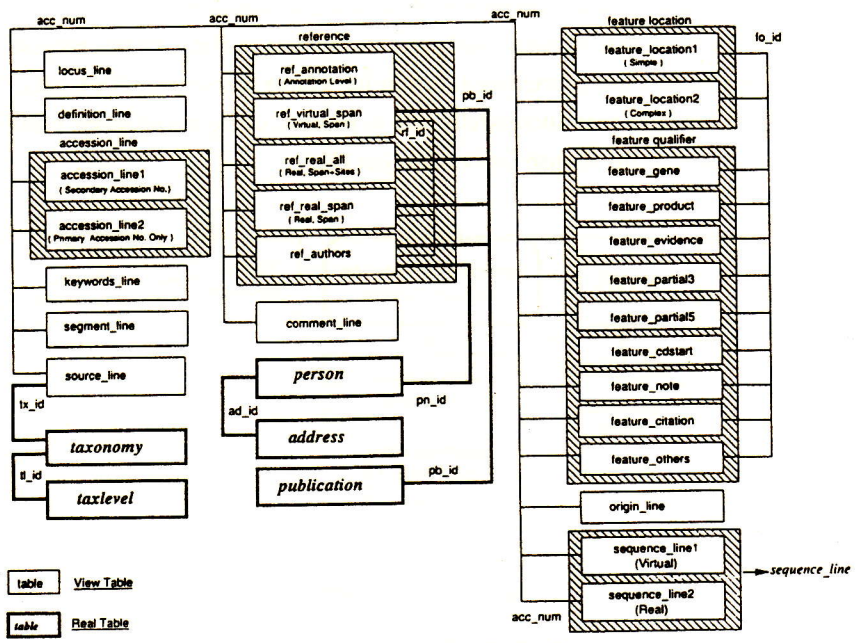


Figure 4. Virtual tables with GenBank-schema

2.2 Database Search with Virtual Tables

View is an alternative way of looking at the data in one or more tables. We can consider "view" as a shifting frame which looks like the flat file format in the GenBank-schema. The "view" function provides databank staff and users with an easy way to manage virtual tables despite the complexity of the GenBank schema and its continuous changes.

We developed about 30 virtual tables [10] with the GenBank-schema. Each virtual table shown in Figure 4 is represented by a rectangle and it can be connected by column to the accession number. Since the SQL language defining "view" does not have sufficient function to define "view", we have to respectively define more than one virtual table for "accession_line", "reference_line", "location_line", and "qualifier_line". For example, the "accession_line" is defined in two tables. The "accession_line" table has both primary and secondary accession numbers as columns but each accession number is stored in a different area. For that reason, the first "accession_line" table is defined as being stored with the entry of both the primary accession number and secondary accession number. The secondary "accession_line" table is defined as being stored with the entry of the primary accession number without a second accession number.

We can also see the real tables which represent "taxonomy", "taxlevel", and so on. Since the data of the "taxonomy" table are all pairs of parent and child nodes in a taxonomy tree, we can not search the taxonomy tree using the view function only. The control language (CFL) of SYBASE discussed in later section is useful for searching the taxonomy tree.

3. Next Generation System

We have 2 main goals in establishing an efficient construction system of the DNA database using the relational database management system at DDBJ.

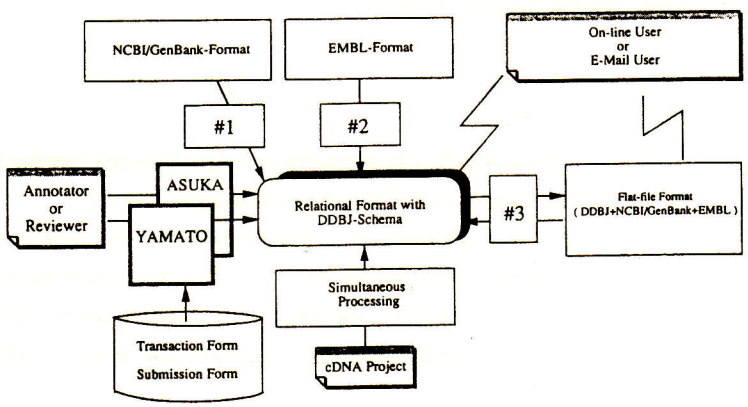


Figure 5. Next Generation System

The first goal is to implement the newly developed DDBJ-schema which is more easy managed than the GenBank-schema with its complex data structure. The DDBJ-schema will be capable of realizing efficient simultaneous processing of large amounts of DNA data and an efficient building system for the DNA database. In addition, it will enable us to easily realize both a flexible environment for constructing an integrated database and high performance to support comfortable on-line service.

The second goal is to implement an effective restructuring tool by which we can convert the DNA database with the GenBank-schema into relational formats with the DDBJ-schema. Because at DDBJ we are building the DNA database with the GenBank-schema using AWB, it is necessary to convert it into the unified DNA database.

3.1 System Configuration

Figure 5 shows the system configuration for achieving the previously stated goals on the UNIX based workstation at DDBJ. Since the GenBank-schema has a complex data structure, the program for simultaneous processing does not provide us with sufficient efficiency. We are going to change the output of the execution into relational formats with the DDBJ-schema.

We developed the DDBJ-schema for storing the integrated DNA database and the restructuring mechanism which is an interface between the GenBank-schema and the DDBJ-schema. In this way, we were able to transfer about 3,000 entries with the GenBank-schema into the DDBJ-schema in about one hour on the UNIX based workstation.

The integrated DNA database needs to receive DNA data with the NCBI/GenBank-flat file of NCBI and EMBL-flat file of the EMBL Data Library. In addition, we need to convert the relational format with the DDBJ-schema into the GenBank flat file format and vice versa.
For this reason, we developed conversion programs #1, #2, #3, as well as two types of building systems for the integrated DNA database at DDBJ. If we can change the current system into the next generation system, on-line users and E-mail users will be able to access the unified DNA database over the computer network.

3.2 DDBJ-Schema

Figure 6 shows the DDBJ-schema which is more easily managed than the GenBank-schema. We successfully developed the hierarchical DDBJ-schema with three layers.

Sixteen tables in the first layer are connected horizontally by an accession number, "acc_num", and include the contents of the flat file formats. We can provide these tables for integrating and searching the integrated DNA database. The DDBJ-schema also includes dictionaries which store biological information, a list of persons, and a list of addresses in the

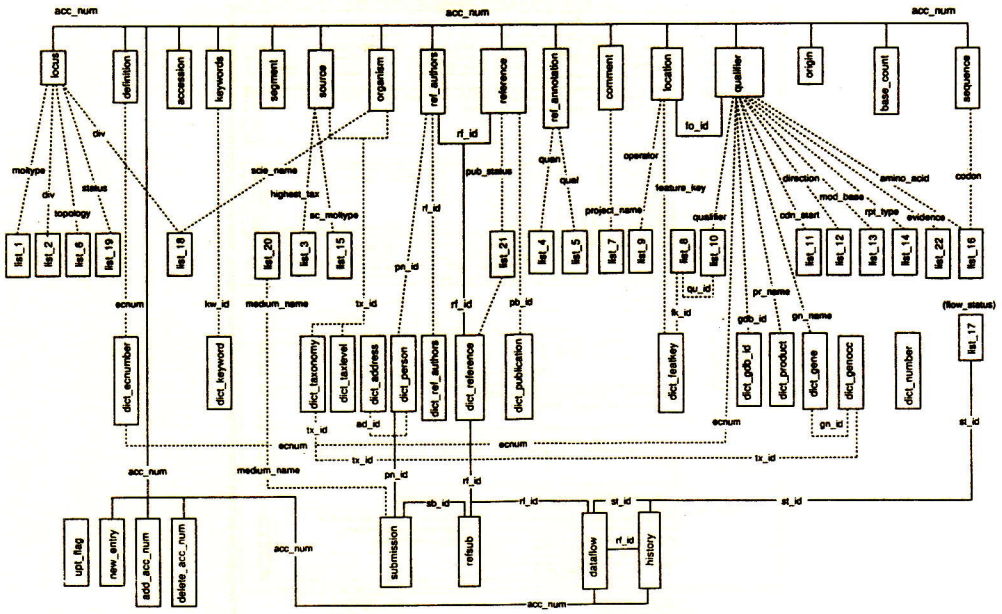


Figure 6. DDBJ-schema

second layer. We use the dictionaries to automatically fill in the blanks of the 16 tables. In the third layer, the DDBJ-schema includes system control tables which are shown in the lowest part of Figure 6. The system control tables are used for data restructuring, data transportation, and data flow management.

The graph shown in Figure 7 compares the real tables in the DDBJ-schema with the virtual tables in the GenBank-schema in terms of search speed and join processing. The real tables in the DDBJ-schema are about 3 to 10 times faster than the virtual tables in the GenBank-schema. Thus, the DDBJ-schema will provide efficient performance for on-line users.

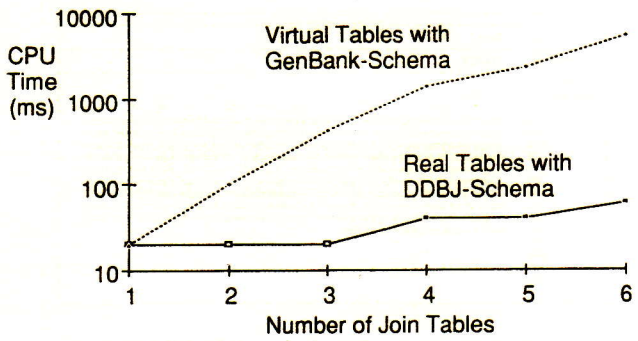


Figure 7. Performance of DDBJ-schema

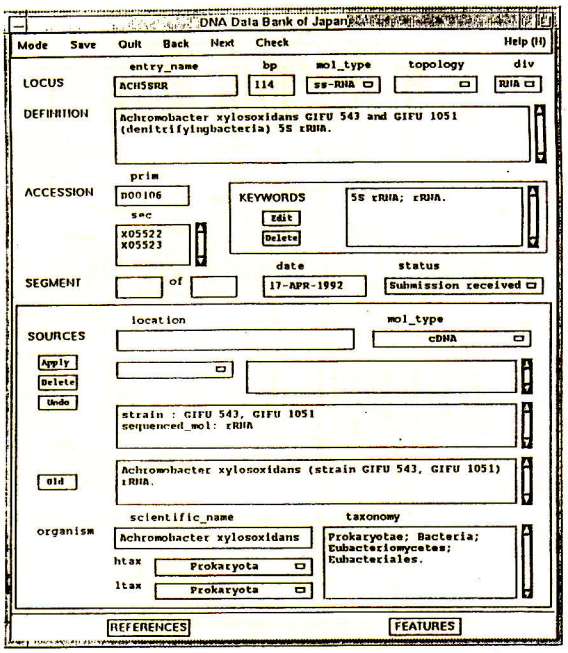


Figure 8. Window interface implemented in X-window

```
+-------------------------------<< EDIT >>-------------------------------+
| / Close  Seq.!  Comment!  Reference!  Source!  Feature!  PROJ            |
|-------------------------------------------------------------------------|
| ACCESSION:  D11552                PROJECT:                              |
|-------------------------------------------------------------------------|
|             entry_name        mol_type  toplogy    div  dd mon yy      |
| LOCUS       HUM0CT2C09        ss_mRNA   Linear     PRI  01 Dec 92      |
|                               status               hold_date          |
|                               public               01 Jan 00          |
| DEFINITION  Human HepG2 3'-directed MboI cDNA, clone c12c09             |
|                                                                         |
|                                                                         |
|                                                                         |
| KEYWORDS    .  EST(expressed sequence tag)                              |
|             .                                                           |
|                                                                         |
|                                                                         |
| SEGMENT           of                                                    |
+-------------------------------------------------------------------------+

+-------------------------------<< SEQUENCE >>-----------------------------+
|  /  Close  Next!  Back!                                                  |
|--------------------------------------------------------------------------|
| ACCESSION:  D11552                                                       |
|--------------------------------------------------------------------------|
| ORIGIN                                                                   |
| Length      563                                                          |
|                    10        20        30        40        50        60  |
|    [      1]  gatctcnnaancnnanaaannctnaaanaaagnnnnaaatnaaaaaaaannagnnnngc |
|           61  ngncataanangatatngancctngcaagcccatgtgtgttgagagctnctcannnnnnn |
|          121  nnnnnnnnnnnnngntttgctgttcgtgannnnngacagacagttgcggtgggtgtcatc |
|          181  aaagcagtngacaagaaggctgcnnnnnctggcaaggtcaccaagtctgnccagaaagct |
|          241  cagaaggctaaatgaatattatccctaatacctgccacccccactcttaatcagtggtggn |
|          301  agaacggtctcagaactgtttgtttcaattggccatttaagtttagtagtaaaannctgg |
|          361  ttaatggtacnagngcatcggnaaacctttncagnanggaaaagggngantgttttntnng |
|          421  accacttnggtttcnntttgcggtttngaagttttaangttaattnggttttaaaaat |
|          481  nnggaacttttttaatngggaanaactttgccccaaaanttttntcaanagnntttnnngn |
|          541  cccnnnggggaangtnnanggng                                      |
|          601                                                             |
|          661                                                             |
|          721                                                             |
|          781                                                             |
+--------------------------------------------------------------------------+
```

Figure 9. Window interface implemented in DB-Library

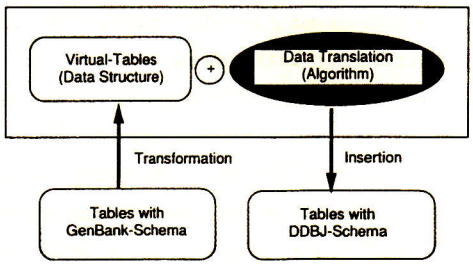


Figure 10. Restructuring mechanism

We developed two types of window interface to build the 16 tables in the DDBJ-schema. The first type has an interface, YAMATO, implemented in X-window and can be used on UNIX based workstations over computer networks. The second type has an interface, ASUKA, implemented in the DB-Library of SYBASE and can be used on character terminals over telephone networks. Both also have a window interface similar to the flat file format.

We show an example of YAMATO in Figure 8. The window shows the range of data from "locus_line" to "source_line" in the flat file format.
Other lines can be displayed by clicking a mouse button after users move the pointer to either "REFERENCE" or "FEATURE". ASUKA is shown in Figure 9. The window shows the range of data from "locus_line" to "segment_line" in the flat file format. Other lines can be displayed by pressing the return key of the keyboard after users move the pointer to either "Seq!", "Comment!", "Reference!", "Source!", or "Feature!".

3.3 Structured SQL-Programming

This subsection describes two examples, restructuring and tree search, of the structured SQL-programming method to be used to implement the system. This programming is achieved by regarding set of tables as data structure and procedure written in CFL as algorithm.

3.3.1 Restructuring

Figure 10 shows the restructuring mechanism [11] for interfacing between the GenBank-schema and the DDBJ-schema. It is implemented by the structured SQL-programming method. The method provides us with a program including the virtual tables and/or real tables as data structure. The program is written in both CFL and SQL.

The real tables with the GenBank-schema are transformed into virtual tables through the "view" function. The virtual tables and the real tables with the DDBJ-schema are similar to each other, but they are not exactly the same. This is due to limitations in the capability of the "view" function as already pointed out. We implemented the data translation algorithm using both CFL and SQL of SYBASE. In our trial system, we successfully restructured from the

GenBank-schema to the DDBJ-schema using both the data translation algorithm and the virtual tables. It took one hour to process about 3,000 entries.

In the current system, reviewers build the DNA database incrementally with the GenBank-schema in the daytime. If they update an entry from a "private" mode to a "distribute" mode, a trigger is fired and the accession number of the entry related to the update is recorded in a table with the DDBJ-schema. At midnight, the incremental restructuring program is executed and new entries are automatically transferred from the GenBank-schema to the DDBJ-schema every day.

3.3.2 Tree Search

The tree search functions include an engine to manage taxonomy database. The engine is defined as recursive join for the taxonomy database. All the taxonomy databases constructed with the DNA databases of the international DNA databanks are powerful electronic dictionaries which aid in biological research by computer. The taxonomy databases are, however not consistently integrated with a relational format. If we can achieve consistent unification of the taxonomy databases, it will be useful in comparing many research results, and investigating future research directions from existent research results. In particular, it will be useful in comparing relationships between phylogenetic trees inferred from molecular data and those constructed from morphological data.

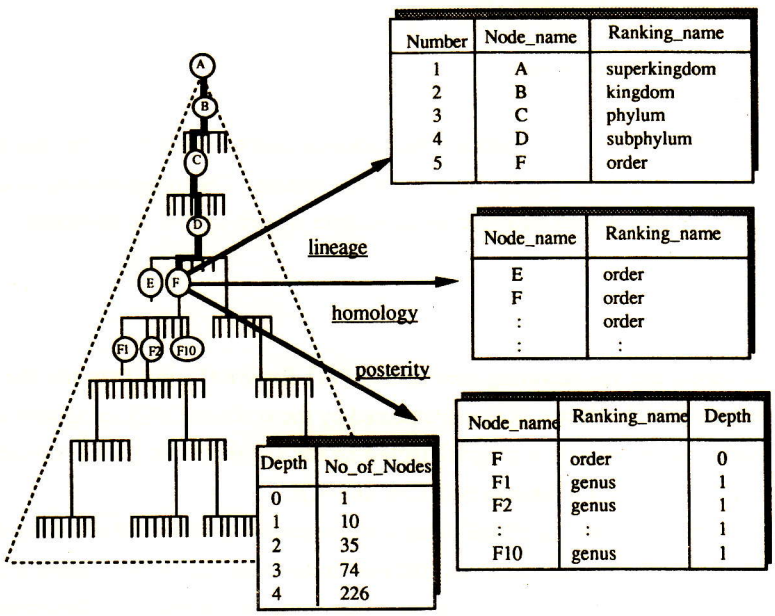


Figure 11. Tree search functions

The system has many kinds of the tree search functions which are lineage, homology, posterity, and so on. All the functions are implemented in the structured SQL-programming method. These functions are shown in Figure 11.

For example, the lineage function searches for a path from a given node to the root node, which does not have any parent node in the tree structure. The path is a set of nodes found by searching in the "taxonomy" table. Appendix-2 shows an example of the program that was implemented in the control flow language of SYBASE called the "stored procedure". Our approach includes the object-oriented database concept, since the procedure is a kind of method for hiding the table structures of the "taxonomy" table. If we apply an artificial intelligence approach, we can obtain another kind of a program implemented in prolog [12]. This approach shown in Appendix-3 is constructed with a smaller program than in the previous approach. If we need more complex and higher processing to unify taxonomy databases in the future, the artificial intelligence approach would be preferable in implementing more efficient processing.

3.4 Load Balancing

It is necessary for the next generation system to include the following processes :

(1) Building the DNA database using AWB on the GenBank-schema and restructuring from the GenBank-schema to the DDBJ-schema

(2) Maintenance for EMBL's and NCBI/LANL's satellites

(3) Processing simultaneously large amounts of DNA data

(4) Building, integration, and on-line service for the unified DNA database

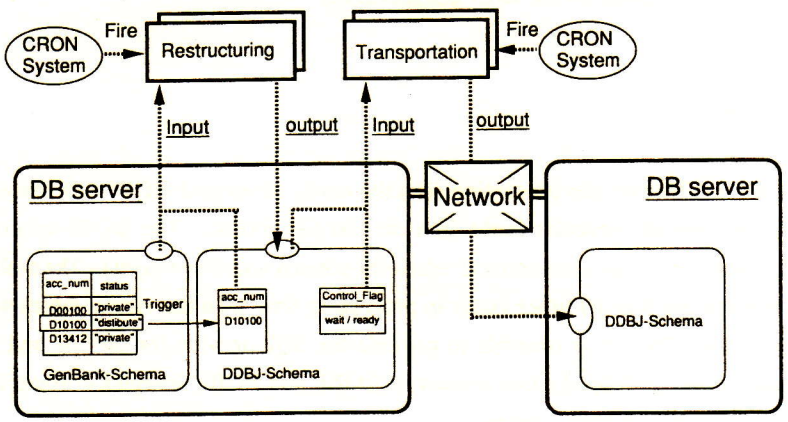


Figure 12. Two-phase method

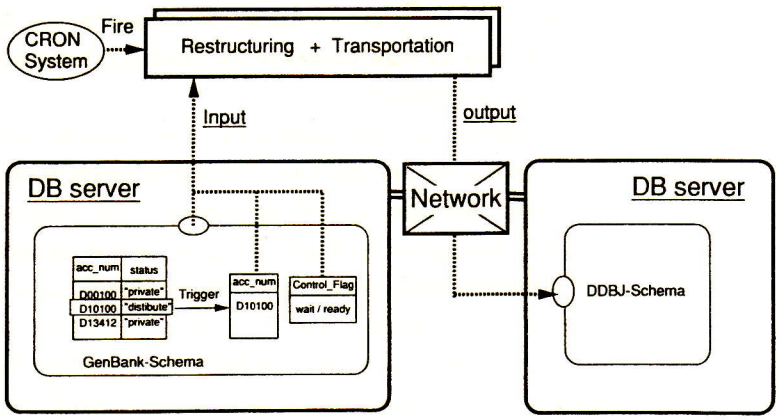


Figure 13. One-phase method

If we execute the above processes (1)-(4) on one workstation or a computer, the response time is quite slow. We have to distribute the above processes among several computers connected by a computer network. For example, let us consider the case in which we execute process (1) on the UNIX based workstation, SUN4/490, (2) on the mini-super computer, CRAY S-MP42, and (3)-(4) on SUN4/2. In this case, it is necessary to transport the DNA database from SUN4/490 to SUN4/2, and from CRAY S-MP42 to SUN4/2. The former transportation is carried out with a two-phase method shown in Figure 12 and the latter with a one-phase method shown in Figure 13. Since the two phase method does not have the restructuring process in the data transportation, the transportation does not need more CPU power than the one phase method. Since the CRAY S-MP42 has higher performance than the SUN workstation, we choose the one phase method for the latter transportation.

4. Conclusion

We have realized effective join processing with the newly developed DDBJ-schema which is easy to use, because its structure looks like the flat file format. The DDBJ-schema was successfully implemented as a hierarchical relational schema with three layers. The tables with the DDBJ-schema are 3 to 10 times faster in processing time than the virtual tables with the GenBank-schema. Thus, it is possible to provide the SQL-service for on-line and E-mail users. Moreover, we developed a tool to convert EMBL and NCBI/GenBank-formats into the relational format with the DDBJ-schema.

We have succeeded in restructuring data with the GenBank-schema into those with the DDBJ-schema by use of the structured SQL programming method. Moreover, incremental restructuring was carried out using the trigger function and the Cron system in UNIX.

We have also succeeded in achieving distributed processing over the computer network to execute our software tools for database building, database integration, and on-line service. As a result, we have developed a multi-server database system over the network.

We developed two kinds of window interfaces for building the DNA database with the DDBJ-schema.

The Rice Genome Project is planning to building a DDBJ's satellite for receiving the integrated DNA database over the network. The Rice Genome Project is quite far from DDBJ. For this reason, it is important to develop a fault tolerance system to consistently transport the DNA data from DDBJ to the Rice Genome Project in case of network failure.

We are planning to move the current system to the next generation system.

Acknowledgments

We wish to thank Prof. A.Makinouchi of Kyushu University and Dr. M.Arikawa of Hiroshima City University for their valuable suggestions. This research was carried out at the Fujitsu-donated Laboratory of the National Institute of Genetics. We also thank all the DDBJ-staff involved in developing the DNA database system for their help.

References

[1] Patricia Kahn and Graham Cameron: EMBL Data Library, Methods in Enzymology, Vol. 183, 1990.

[2] Desmond G. Higgins, Rainer Fuchs, Peter J.Stoeher and Graham N.Cameron: The EMBL Data Library, Nucleic Acids Research, Vol.20, Oxford University Press, 1992.

[3] Christian Burks, Michael J. Cinkosky, Paul Gilna, et al: GenBank: Current Status and Future Directions, Methods in Enzymology, Vol. 183, 1990.

[4] Michael J. Cinkosky, James W. Fickett, Paul Gilna, and Christian Burks: Electronic Data Publishing and GenBank, Science, Vol. 252, 1991.

[5] 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, 1992.

[6] GenBank Release Notes, Release 69.0, Tape Distribution, IntelliGenetics Inc., September 1991.

[7] GB-Schema, News From GenBank, Summer/Autumn 1991, Vol.4, No.4, c/o IntelliGenetics, Inc., 700 East El Camino Real, Mountain View, CA 94040.

[8] SYBASE Transact-SQL User's Guide, Sybase Inc., May 1989.

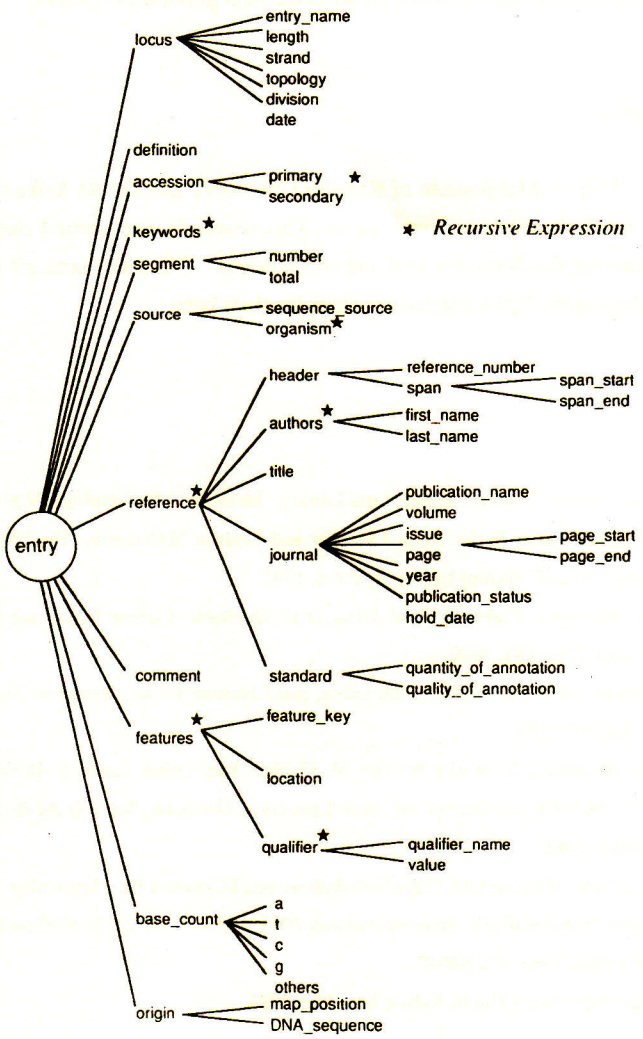
[9] Kousaku Okubo, Naohiro Hori, Ryo Matoba, Toshiyuki Niiyama, Atsushi Fukushima, Yuko Kojima and Kenichi Matsubara: Large Scale cDNA Sequencing for Analysis of Quantitative and Qualitative Aspects of Gene Expression, Nature Genetics, Vol.2, Nov 1992.

[10] Hajime Kitakami and Yukiko Yamazaki: Building and Searching Biological Information based on a Relational Database System, Annual Report, No. 42, 1991, National Institute of Genetics, 1992, to be appeared in Bulletin of the Japan Federation for Culture Collections, Vol.8, No.2 (in Japanese), 1992.

[11] Hajime Kitakami and Yukiko Yamazaki: Integration and Search System for a Large-scale DNA Database, International Symposium on Molecular Evolution and Bioinformatics, National Institute of Genetics and Fujitsu Limited, Mishima, Japan, April 1993, to be appeared in Annual Report, No. 43, 1992, National Institute of Genetics, 1993.

[12] Prolog by BIM(BIM_Prolog) Reference Manual, BIM (Belgium), 1990.

Appendix-1. Grammatical Structure of the Flat-file Format



Appendix-2. An Example of Programming for a Lineage Search Implemented in both SQL and CFL

```
create procedure lineage @nodename char(80) as

declare @tuples int
declare @cnt    int

select  @cnt=1
create table tempdb..lineage(
          tx_id char(16),tx_idp char(16),
        level tinyint ,level_name char(32),node_name char(80))
create table tempdb..temp_table(
        tx_id  char(16),tx_idp  char(16),
        level tinyint ,level_name char(32),node_name char(80))
create table tempdb..work_table(
          tx_id char(16),tx_idp char(16),
        level tinyint ,level_name char(32),node_name char(80))

insert tempdb..temp_table
            select tx_id,tx_tx_idp,@cnt,tl_levname,tx_nodename
        from    taxonomy,taxlevel
        where   tx_tl_id=tl_id
        and   tx_nodename=@nodename
insert  tempdb..work_table
        select * from tempdb..temp_table
insert tempdb..lineage
        select * from tempdb..temp_table
select @tuples=count(*) from tempdb..temp_table
delete tempdb..temp_table

while @tuples!=0
begin
      select @cnt=@cnt+1
      insert tempdb..temp_table
                select  x.acc_num,y.tx_id,y.tx_tx_idp,
                        @cnt,tl_levname,tx_nodename
                from    tempdb..work_table x,taxonomy y,taxlevel
                where   tx_tl_id=tl_id
                and     y.tx_id=x.tx_idp

       delete tempdb..work_table

       insert tempdb..work_table
                select * from tempdb..temp_table
       insert tempdb..lineage
                select * from    tempdb..temp_table
        select @tuples=count(*)
       from   tempdb..temp_table

       delete tempdb..temp_table
end

select * from tempdb..lineage
drop table tempdb..lineage, tempdb..temp_table, tempdb..work_table
```

Appendix-3. An Example of Programming for a Lineage Search Implemented in BIM-Prolog

```
lineage(NODENAME,[[TX_ID,TX_TX_IDP,LEVELNAME,NODENAME]|Result]):-
        taxonomy(TX_ID,TX_TL_ID,TX_TX_IDP,NODENAME),
        taxlevel(TX_TL_ID,LEVELNAME),
        search(TX_TX_IDP,Result).

search(TX_ID,[[TX_ID,TX_TX_IDP,LEVELNAME,NODENAME]|Result]):-
        taxonomy(TX_ID,TX_TL_ID,TX_TX_IDP,NODENAME),
        taxlevel(TX_TL_ID,LEVELNAME),
        search(TX_TX_IDP,Result).
search(TX_ID,[]).

taxonomy(TX_ID,TX_TL_ID,TX_TX_IDP,NODENAME):-

retrieve(db_taxonomy(TX_ID,TX_TL_ID,TX_TX_IDP,NODENAME,_,_,_,_,_,_,_,_,_,_,_,_)),!.

taxlevel(TX_TL_ID,LEVELNAME):-
        retrieve(db_taxlevel(TX_TL_ID,LEVELNAME,_,_)),!.
```

The "db_taxonomy" and "db_taxlevel" tables are two tables of the taxonomy database defined in section 2. A system configuration for the deductive approach is shown in the following Figure: