Functional Approach to Chemical Structure Databases


* Program in Engineering Sciences
** Institute of Information Sciences and Electronics
Univ. of Tsukuba

ABSTRACT
This paper presents a new scheme for design and implementation of a functional data model on top of the conventional relational DBMS. Motivation comes from the need for improved tools and techniques for the specification, design and implementation of chemical structure databases. The scheme provides a high-level user interface integrated into the programming language LISP. An extended relational DBMS with capabilities of handling ADTs (Abstract Data Types) is developed. A functional data model, called TIME data model, is designed and implemented as an external model of the extended relational DBMS. The scheme supports behavioral abstraction as well as structural abstraction. Implementation of some features has been completed on UNIX environments and they are put into experimental evaluation.

Key Words and Phrases: Chemical Structure, Abstract Data Type, Functional Data Model, Relational Data Model.

1 Introduction

This paper presents a new scheme for the specification, design and implementation of chemical structure databases. The fundamental ideas are from hierarchical representation of chemical structures by means of the BCT (Block-Cutpoint Tree) [Nak80], abstract data type [Jam83, Jin83, Osb86, Sto83], relational database [Cod70] and functional data model [Shi81].

Information about chemical structures is essential in many engineering and scientific applications, such as substructure searches, structure elucidation from spectral data, drug design, and planning of synthesis. One of the major problems in computer handling of chemical structures is the explosion of time and space complexities caused by increase in the number of atomic combiners. Many attempts have been made to design and implement concise, highly descriptive and easy-to-manipulate models of chemical structures [Fuj81, Gra79, Nak83]. The BCT model adopted here is universal and independent of particular applications, and application systems based on it may substantially reduce time and space complexities.

Most of the works in chemical structures have adopted the approach of designing a system tailored to specific chemical applications. Relatively fewer attempts have been made to connect existing chemical structure models with conventional DBMSs, because conventional DBMSs do not allow the modeling of chemical structures in a natural way or at least they do not support the retrieval and manipulation of such structures in a way that is familiar to chemists.

In TIME project, we have designed the TIME chemical data management system, in which an extension of BCT representation model connects with a conventional relational DBMS. A relational DBMS is chosen because of its data-independent representation, and high-level information access facility. However, straightforward application of the relational model to chemical structure databases lacks constructs to define hierarchical nature of the decomposition of chemical structures naturally and to define manipulation of chemical objects such as chemical formulas and chemical graphs in a succinct way.

However, developments in programming language area offer very promising prospects for modeling data accompanied by complicated chemical semantics. Abstract Data Type (ADT) in the programming language area facilitates uniform description of chemical semantics. Abstract Data Type (ADT) in the programming language area facilitates uniform description of chemical semantics. Abstract Data Type (ADT) in the programming language area facilitates uniform description of chemical semantics. Abstract Data Type (ADT) in the programming language area facilitates uniform description of chemical semantics.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the DASFAA copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Organizing Committee of the International Symposium on Database Systems for Advanced Applications. To copy otherwise, or to republish, requires a fee and for special permission from the Organizing Committee.
siders individual columns of relations as ADTs. The approach is similar to those of ADT-INGRES [Jam83, Ston83] and RAD [Osb86]. By adding ADTs for encapsulating chemical objects such as chemical graphs and chemical formulas to standard types, sophisticated data structures can be more directly described in the relational framework.

The hierarchical decomposition of a chemical structure is represented as a collection of tuples that belong to more than one relation. Therefore a chemical structure can be considered as a complex object [Lor83] or molecular object [Bat84]. Typical approaches are based on hierarchical extensions to the relational model such as XSQL [Has82] or the nested relations [Dad86, Jae82, Kit87]. Functional data model as an external view provides facilities for mapping complex objects onto relational database structures and for retrieving these objects as single entities. It also supports traversal among a variety of relationships appearing in these objects.

The scheme proposed in this paper provides a sound basis for the specification, design and implementation for the chemical structure databases. The fundamental features of TIME chemical data management system are as follows.

1. It provides a high-level user interface integrated into the programming language LISP. LISP is used to implement chemical structure applications as well as to manipulate chemical structures in TIME databases.

2. The extended relational DBMS is implemented on the commercially available relational DBMS, G-BASE.

3. A functional data model, called TIME data model, is designed and implemented as an external model of an extended relational DBMS having capability of ADT manipulation.

Our scheme has been implemented on UNIX environments as a part of the TIME project. Implementation of some features has been completed and they are put into experimental evaluation.

The remainder of this paper is organized as follows. Section 2 describes our model of chemical structures and its representation in the relational model. In section 3 we describe architecture of TIME chemical data management system and the basic semantics and the language interface of TIME data model. In section 4 we apply our scheme to an example chemical structure database to exemplify its applicability. Section 5 is devoted to conclusion and discussion of the future research topics.

2 Model of Chemical Structure

In this section, we present our conceptual database of chemical structures. First we describe a chemical structure model based on BCT. Then relational schemas and an illustrative example of database instances are presented. Finally we briefly discuss our motivation for developing an external functional view.

2.1 Segmentation of Chemical Structure: Super Block Segmentation Method

Since many applications of chemical structure databases require substructure searches, an effective and powerful segmentation of a whole structure into meaningful units is a key to successful databases. The BCT (Block-Cutpoint Tree) representation of a chemical structure is a graph-theoretical decomposition method, where intermediate descriptive units are introduced to facilitate substructure searches.

A chemical structure is regarded as a graph whose vertices and edges correspond to atoms and bonds, respectively. The descriptive unit in the BCT scheme is called a block or a biconnected component of a graph and corresponds to what is called a ring system or a ring assembly, except for blocks that have only two vertices. An example of the BCT tree is shown in Fig.1, where a cutpoint is defined as a vertex whose removal increases the number of connected components of a graph. This method is powerful for detecting ring systems in a chemical structure. However, it has little capability of describing and treating acyclic substructures, because acyclic parts of a chemical structure are segmented into small separate blocks consisting only two vertices.

In order to overcome this difficulty, we introduce a new descriptive unit, called a super block, which represents an acyclic part of a chemical structure as a single super block without any segmentation. So as a result, what we call a super block is either

1. cyclic super block - a block whose number of vertices is more than two, or

2. acyclic super block - a component of a graph generated by removing cyclic super blocks from the original graph.

An SC-tree (Super-block Connection tree) is constructed to represent the connections among super blocks by denotes super blocks as nodes and connections among them as edges. Fig.2(a) shows an example of the super block segmentation and an SC-tree.
Cyclic super blocks can further be segmented into fundamental rings by the S.S.S.R. (Smallest Set of Smallest Ring) method [Gas79] (See Fig. 2(b)). These segmentation methods are universal and independent of particular applications, and can support manipulation of chemical structures in a way that is familiar to chemists.

### 2.2 Representation Model of Chemical Structure

In designing a chemical structure database system, we first introduce a structural representation model which satisfies the following requirements.

1. It can describe the segmentation from any chemical structure to its substructures.
2. It can support sharing of common substructures in the database.
3. It allows users to reconstruct the original chemical structure from the segmented substructures using connectivity information among substructures.

Fig. 3 illustrates a conceptual diagram of our model. The structural representation of the model is a Directed Acyclic Graph (DAG), and can be defined as $\text{GRAPH} = (N, E)$, where $N$ represents the set of nodes and $E$ represents the set of edges. $N$ have three mutually disjoint subsets, i.e., $N = GN \cup CN \cup LN$, $GN \cap CN = CN \cap LN = LN \cap GN = \phi$, where, as illustrated in Fig. 3,

- $GN$(Graph-Nodes) : correspond to chemical structures and substructures,
- $CN$(Connection-Nodes) : represent the connections among substructures,
- $LN$(link-Nodes) : represent the correspondences between vertexes in chemical structures and vertexes in substructures.

Constraints on the indegree and outdegree of these three kinds of nodes are listed below.

1. The indegree of $GN$ is greater than or equal to 0, while the outdegree of it is 0 or 1. A graph node of indegree 0 is called a root graph node, and a graph node of outdegree 0 is called a leaf graph node.
2. The indegree of $CN$ is 1, while the outdegree of it is greater than 0.
3. Both the indegree and outdegree of $LN$ are 1.

$E$ also have three disjoint subsets, i.e., $E = GCE \cup CLE \cup LGE$, and $GLE \cap CLE = CLE \cap LGE = LGE \cap CLE = \phi$, where $\text{Graph-Connection-Edges} \subseteq GN \times CN$, $\text{Connection-Link-Edges} \subseteq CN \times LN$ and $\text{Link-Graph-Edges} \subseteq LN \times GN$, respectively.

### 2.3 Relational Database Schema of Chemical Structure Database

TIME system has a layered architecture as elaborated in the next section. At the bottom of it lays the extended relational database management system with ADT handling capabilities. Here we show a relational database schema which represents aforementioned structural decomposition model of chemical structure.

- **R1**: $\text{COMPOUND}$
  
  - $(C\#, \text{NAME}, \text{NUM-OF-VERTEXES}, \text{NUM-OF-EDGES}, \text{CHEMICAL-FORMULA})$

- **R2**: $\text{SUBSTRUCTURES}$
  
  - $(S\#, \text{NUM-OF-VERTEXES}, \text{NUM-OF-EDGES}, \text{CHEMICAL-GRAPH}, \text{TYPE}, \text{CHEMICAL-FORMULA})$

- **R3**: $\text{CONNECTION}$
  
  - $(CG\#, \text{NUM-OF-NODES}, \text{NUM-OF-EDGES}, \text{GRAPH}, \text{TYPE}, \text{EDGE-MAPS})$

- **R4**: $\text{LINK}$
  
  - $(L\#, \text{VERTEX-MAPS})$

- **R5**: $\text{COMPOUND-CONNECTION}$
  
  - $(C\#, \text{CG}\#)$

- **R6**: $\text{SUBSTRUCTURE-CONNECTION}$
  
  - $(S\#, \text{CG}\#)$

- **R7**: $\text{CONNECTION-LINK}$
  
  - $(CG\#, L\#)$

- **R8**: $\text{LINK-GRAPH}$
  
  - $(L\#, S\#)$

Graph nodes in the structural decomposition model are divided into two types, i.e., chemical structure nodes and substructure nodes depending on their difference with respect to chemical meanings. $R1$ and $R2$ represent chemical structure nodes and substructure nodes, respectively, $\text{NUM-OF-VERTEXES}$ and $\text{NUM-OF-EDGES}$ are listed as examples of attributes that are used as keys for retrieval of structures. $R3$ and $R4$ correspond to connection nodes and link nodes, respectively. $R5$ and $R6$ correspond to GCE which describes the correspondence between a chemical structure and its segmentation. $R7$ corresponds to CLE and represents the correspondence between segmentation and its component. $R8$ corresponds to LGE and gives a relationship between nodes and substructures in segmentation. An example of database instances is shown in Fig. 4.
2.4 Motivation for Developing the External Functional View

The scheme in the previous subsection assumes an extension of the conventional DBMS, namely uses of ADTs in individual attributes of relations. An ADT is a user-defined data type specified by the encapsulated data structure and externally visible operations for its manipulation. In the above scheme, new types such as CHEMICAL-FORMULA, CHEMICAL-GRA PH, EDGE-MAPS, GRAPH and VERTEX-MAPS have to be supported. Without attributes of ADT domains, those objects have to be encoded in flat tabular form only consisting of primitive data items such as integers, reals and character strings. However, this restriction makes database manipulation very irritating.

Chemists naturally views that a chemical structure has a number of substructures as its parts, and a part again has a number of substructures as its subparts. However, the scheme in the previous subsection does not necessarily models the decomposition of a chemical structure in a way that is familiar to chemists. The decomposition is represented by a collection of tuples that belong to different relations. Manipulation of the inclusion relationship appearing in the chemists' view, requires users to write very lengthy and tedious operations over several relations such as LINK, CONNECTION, etc.

In order to support the chemists' natural view, an external model on top of relational database schema is required. The functional data model and the language DAPLEX proposed by Shipman seem to be a very promising candidate for implementing the external view. It is well known that DAPLEX supports the representation of hierarchical relationships. It provides the facility of nested functions to retrieve objects that are composed of several different entity types. DAPLEX is often accused that it does not allow the user to define computationally complex functions. In DAPLEX functions are merely used to define relationships. In order to support the manipulation of objects by application-specific operators, DAPLEX would have to be extended to provide user-defined operations on the entities[Ste86]. By adding ADTs defined in the underlying relational DBMS to standard data types in the scheme of the functional data model, application specific operations can be flexibly supported.

3 TIME Functional Data Model

In this section, we present TIME functional data model. We first describe the layered architecture of TIME system. Then our model is presented with a simple example of a chemical structure database.

3.1 TIME System Architecture

Fig.5 is a depiction of the layered architecture of TIME chemical data management system. The Entity Handler implements TIME data model. It consists of the data modeler, function descriptor and the query processor. The data modeler defines a functional data model similar to that of DAPLEX. The function descriptor defines each function in TIME data model in terms of the language of the Extended Relational System, called ADT-QUEL. The query processor translates TIME queries into the ADT-QUEL formats. The Extended Relational System realizes the concept of "individual columns as ADTs" proposed by Stonebraker[Jam83,StoS3] and Osborn[Osb86]. It has been developed by adding a module, called "ADT-QUEL interpreter"[Jia88], into the commercially available relational DBMS G-BASE.

G-BASE is distinguishable from other commercially available relational DBMSs by its LISP-based programming interface, called DM-Lisp. DM-Lisp consists of usual LISP functions and database handling functions. LRT(Lisp-Run-Time) is a set of functions, which accepts S-expressions of queries and translates them into DM-Lisp forms. Then by evaluating those forms, database accesses are carried out.

ADT-QUEL interpreter provides users with a capability to define ADTs. It also has a capability to translate the extended QUEL statements including user-defined ADT functions into LISP S-expressions. The S-expressions are further translated into DM-Lisp forms by LRT. LRT is also modified to accept ADT functions embedded in the extended QUEL statements.

3.2 ADT in the Extended Relational Model

In ADT-QUEL ADT is defined along the line in programming language theory. An ADT is modeled as follows[Jia88]:

\[ \text{ADT} = (N, S, F) \]

where, \( N \) is the name of ADT, \( S \) is the specification of data structure of it and \( F \) is a set of ADT functions defined on it to manipulate objects belong to it. Only the name and ADT functions are externally visible, the data structure specified by \( S \) can be accessed only through ADT functions. So ADT functions handle the semantics attached to an ADT.

ADT functions can be classified into the following three types.

1. D-function
   \[ A \times A_1 \times ... \times A_m \times P_1 \times ... \times P_n \rightarrow \text{boolean} \]

2. P-function
   \[ A \times A_1 \times ... \times A_m \times P_1 \times ... \times P_n \rightarrow P' \]
3. A-function :
   \[ A \times A_1 \times \ldots \times A_m \times P_1 \times \ldots \times P_n \to A' \]

Here, \( m \geq 0, n \geq 0, A_i(i=1,\ldots,m) \) and \( A' \) are ADTs which are not necessarily distinct from \( A \). \( P_j(j=1,\ldots,n) \) denotes a primitive data type. We designate integer, real, character string, boolean as primitive data types. \( P' \) denotes a primitive data type other than boolean. \( B \)-function is a function which returns true or false. \( P \)-function returns a value of primitive data type other than boolean. \( A \)-function is a function which returns an object belong to an ADT. For example, the following expression gives definition of an ADT "CHEMICAL-GRAPH".

ADT-definition( 
   ADT name = CHEMICAL-GRAPH, 
   B-function in 
      is CHEMICAL-GRAPH \times 
         list-of CHEMICAL-GRAPH \to boolean, 
   B-function display 
      is CHEMICAL-GRAPH \to boolean, 
   P-function num-of-C 
      is CHEMICAL-GRAPH \to integer, 
   .... 
   .... 
   filename = "/groupname/username/adt 
              /CHEMICAL-GRAPH"
)

Here, "list-of CHEMICAL-GRAPH" represents a list, and all of its elements are of the same type CHEMICAL-GRAPH. "Filename" is the name of the file that contains the definition of ADT functions written in the programming language LISP. It is noteworthy that each function is defined in the programming environment, and therefore it is not stored in the persistent database environment.

3.3 Entity Type in the TIME Data Model

The TIME data model is based primarily on DAPLEX functional data model. The external schema is made of a set of definitions of entity types. The syntax of the definition of an entity type is as follows.

(def-entity-type E(Parent) 
   ((F1 R1) 
    (F2 R2) 
    .... 
    (Fn Rn)))

Here, Parent is the specification of the inheritance. In our scheme, a multiple inheritance hierarchy is supported, which enables an entity type to have more than one parent. \( F_i(i=1,\ldots,n) \) is a function whose domain is the entity type \( E \) and \( R_i(i=1,\ldots,n) \) is the range of each function. As the range of functions, domains of the underlying relational database, including user-defined ADTs, as well as entity types defined elsewhere are allowed.

Entity types are introduced to handle a collection of attribute values as a whole database entity. Functions associating with them represent relationships between two values in the persistent database. Therefore, definitions of functions are persistently stored in the database. On the other hand, ADTs are introduced to support application specific operations on single domains of the relational database.

We give examples to illustrate the entity type definition.

(def-entity-type COMPOUND() 
   (ID CHAR) 
   (SEGMENTATION SC-TREE) 
   (STRUCTURE CHEMICAL-GRAPH) 
   (COMPONENT set-of SUBSTRUCTURE)))

(def-entity-type SC-TREE() 
   (ID CHAR) 
   (GRAPH TREE) 
   (INCLUDE set-of SUPER-BLOCK)))

Here, two entity types named COMPOUND and SC-TREE are defined. The COMPOUND entity type has four functions named ID, SEGMENTATION, STRUCTURE and COMPONENT. CHAR is a primitive data type, representing a surrogate, i.e., the unique identifier in the underlying database of the compound. So the ID function returns a surrogate of a compound. SC-TREE is one of the other entity types of the schema, so SEGMENTATION models a relationship between entities. SUBSTRUCTURE is also an entity type. COMPONENT function is multi-valued, because a compound can be composed of many substructures. CHEMICAL-GRAPH is an ADT defined in the underlying extended relational system.

3.4 Function Descriptions

Functions defined on entity type are classified into the following six types.

1. entity type \( \to \) primitive data type
2. entity type \( \to \) ADT
3. entity type \( \to \) entity type
4. entity type \( \to \) set-of primitive data type
5. entity type \( \to \) set-of ADT
6. entity type \( \to \) set-of entity type

In the definition of entity types COMPOUND and SC-TREE shown in the previous subsection, the type of
functions ID, SEGMENTATION, STRUCTURE, COMPONENT, GRAPH and INCLUDE is (1), (3), (2), (6), (2) and (6), respectively. Mapping from the external functional data model to the extended relational model is represented by the specification of functions. Functions are specified by giving a way to retrieve function definitions stored persistently in the database. For example, SEGMENTATION function is specified by the following def-attribute-function special form.

\[
(\text{def-attribute-function SEGMENTATION}(i)
\begin{align*}
&\text{range of } t \text{ is COMPOUND-CONNECTION} \\
&\text{retrieve } (\text{cvt } t.\text{CG#}) \\
&\text{where } t.\text{C#} = \text{ID}(i))
\end{align*}
\]

The body of the special form is an ADT-QUEL expression, where \( i \) is an entity of COMPOUND type, \( \text{ID}(i) \) returns a surrogate of \( i \) in the database. The \( \text{t.CG#} \) column stores surrogates of SC-trees. Function \( \text{cvt} \), an inverse function of \( \text{ID}: \text{SC-TREE} \rightarrow \text{CHAR} \), converts the internal representation of an entity, i.e., a surrogate, to an entity of \( \text{SC-TREE} \) type. Furthermore, by using ADTs, we can define functions which do not appear explicitly as attributes in the underlying database.

\[
(\text{def-attribute-function ATOMS}(i)
\begin{align*}
&\text{range of } t \text{ is COMPOUND} \\
&\text{retrieve } (\text{set-of-atoms } t.\text{CHEMICAL-FORMULA}) \\
&\text{where } t.\text{C#} = \text{ID}(i))
\end{align*}
\]

Here, \( t.\text{CHEMICAL-FORMULA} \) returns \( \text{CHEMICAL-FORMULA ADT} \), and \( \text{set-of-atoms} \) is a function defined on it. Users can write a query as if there were the attribute \( \text{ATOMS} \) in the underlying database.

The def-derived-function special form provides a construct similar to the derived function in DAPLEX. The following form defines a function which computes super blocks contained in a compound.

\[
(\text{def-derived-function INCLUDE-BLOCK}(i)
\begin{align*}
&\text{INCLUDE } (\text{SEGMENTATION } i)
\end{align*}
\]

Furthermore, operations on entity types can be defined by the def-operation special form. The function body of this special form is a valid LISP form. The following form computes the isomorphism between two SC-trees.

\[
(\text{def-operation SAME-STRUCTURE}((\text{SC-TREE } i)\ \text{SC-TREE } j))
\]

where isomorphic is a function defined on \( \text{TREE ADT} \).

3.5 Data Manipulation

The for-each special form provides a capability of manipulation of \( \text{TIME data model} \). The form is similar to that of Static[Wei88]. Here, we will show a simple example of queries.

**Query**: Display all chemical structures of compounds which contain \( \square \) as a substructure.

\[
(\text{for-each } ((i \text{ COMPOUND}))
\begin{align*}
&(\text{where } (\text{in } \text{graph}) \\
&(\text{mapcar 'STRUCTURE } (\text{COMPONENT } i)))
\end{align*}
\]

In the above query, \( i \) stands for an entity of \( \text{COMPOUND} \) type. The content of where clause is always a valid LISP form that expresses the qualification of the query. The \( \text{(graph)} \) form is a constant function defined on \( \text{CHEMICAL-GRAPH ADT} \). Here a value \( \square \) is assumed to be designated as a constant of \( \text{CHEMICAL-GRAPH ADT} \). The function "in" is also defined on the \( \text{CHEMICAL-GRAPH ADT} \). It returns true if the first argument is in the list specified by the second argument. The content of "values" clause must be a valid LISP form that expresses how to format the return values of the query. The function "display" is defined on \( \text{CHEMICAL-GRAPH ADT} \), and it generates as side-effect a two-dimensional image of a given chemical graph.

4 Functional Data Model of Chemical Structure Database

In section 2, we described a chemical structure model based on the hierarchical decomposition, and presented a representation in the relational model. In section 3, we described our \( \text{TIME data model} \). In this section, we will present a chemical structure database in terms of the \( \text{TIME data model} \) and give some examples of queries.

4.1 Chemical Structure Database

\( \text{TIME data model} \) provides high-level information hiding facility and powerful data manipulation capability. It handles complex objects and complex data such as graphs under the concepts of entity types and/or ADTs. Using \( \text{TIME data model} \), users can define their own view on application area data without the knowledge of representation details of these data in the database. In Fig.6(a) we show the type hierarchy of chemical structure database, and in Fig.6(b) we show an example schema defined as an external view on the relational database shown in section 2.3.

In Fig.6(b), the \( \text{COMPOUND} \) entity type corresponds to the relation \( \text{COMPOUND} \) in the underlying relational database. However, \( \text{SEGMENTATION, STRUCTURE} \)
and COMPONENT functions access to other relations such as COMPOUND-CONNECTION, CONNECTION-LINK, LINK-GRAPH, SUBSTRUCTURE and LINK. So it represents an object that falls into a category of a complex object or a molecular object. The SEGMENTATION function computes an SC-tree as a result of the segmentation of a compound. Each entity of SC-TREE type corresponds to such a tuple in the CONNECTION relation that its TYPE value is "SC-TREE". The values in TYPE column in CONNECTION can be utilized to classify the type of CONNECTION into subtypes. It is noteworthy that STRUCTURE function in the definition of COMPOUND entity type computes a chemical graph from its components using connectivity information scattered over several relations.

4.2 Query Examples

Here we will show some examples of queries which access to chemical structure database through the external view defined above.

Query1
List compounds whose SC-tree is isomorphic to a given tree.

(let ((result nil)
      (for-each ((i COMPOUND))
        (where (match (tree) (GRAPH (SEGMENTATION i))))
        (values (setf result (cons (NAME i) result))))))

Where, the function “match” is defined on the TREE ADT. The form (SEGMENTATION i) returns an entity of the SC-TREE type, and the function GRAPH defined on the SC-TREE returns an object of TREE ADT. The form (tree) is the constant function defined on the TREE ADT.

Query2
Display all chemical structures which contain connected directly.

(for-each ((i COMPOUND))
  (where (CONTAIN (SEGMENTATION i) (SC-TREE)))
  (values (display (STRUCTURE i)))))

Here CONTAIN is a binary operation defined by the def-operation special form. It returns true if the first SC-tree contains the second SC-tree as its component. The form (SC-TREE) is a constant function defined on the SC-TREE entity type. A constant of entity type is defined by designating the function values explicitly.

Query3
Display structures of compounds which contain a cyclic super block of size 5.

(for-each ((i CYCLIC-SUPER-BLOCK))
  (where (= 5 (NUM-OF-ATOMS i)))
  (values (mapcar 'display (mapcar 'STRUCTURE (PARENT i)))))

Here NUM-OF-ATOMS and PARENT are defined on SUBSTRUCTURE and SUPER-BLOCK entity types, respectively. These functions are inherited to CYCLIC-SUPER-BLOCK type, PARENT function returns a list of compounds having a given cyclic super block as its component.

5 Conclusion

This paper has proposed a new approach to design and implement chemical structure databases. We start from the hierarchical decomposition of chemical structures based on the super block decomposition and S.S.S.R. method. Basic motivations of this study have been to object-oriented and flexible manipulation of this hierarchical decomposition. Key construct of our scheme is the layered architecture consisting of the conventional relational DBMS, extended relational DBMS with capabilities of ADT handling and the functional data model as an external view.

The approach supports behavioral abstractions by introducing of ADTs as well as has capability to realize the structural abstraction such as generalization hierarchy and/or complex objects.

The fundamental features of our data model TIME are as follows.

1. It provides a high-level user interface integrated into the programming language LISP.
2. It is designed and implemented as an external view of the commercially available relational DBMS, G-BASE, so it provides logical data independence between views and the conceptual database.

There are some future works remaining.

1. So far we only support associative retrieval to the chemical structure database. Insertion and deletion of objects as well as the update of view are also needed to be supported.
2. In order to handle large amount of chemical structure data, considerable efforts for speed up are required.

6 Acknowledgement

The authors wish to thank many members of database division of Ricoh Software Lab. for their helpful discussion.
on the design and implementation of TIME project. This work is partially supported by grants from University of Tsukuba Project Research.

References


Fig. 1 Cutpoints, Blocks, and the BCT of a Chemical Structure.

Chemical Structure Graph, Super Blocks, SC-tree

Fig. 2(a) A Chemical Structure Graph, its Super Blocks and SC-tree.

Cyclic Superblock, S.S.S.R.

Fig. 2(b) A Cyclic Super Block and its S.S.S.R.

Fig. 3 The Conceptual Diagram of the Representation Model.

Fig. 4 An Example of Chemical Structure Database Instances.
Fig. 5 The Layered Architecture of TIME Chemical Data Management System.

Fig. 6(a) The Type Hierarchy of Chemical Structure Database.

Fig. 6(b) An Example Schema of Chemical Structure Database.