UDC 541.6＋002.5

社内化合物データベースシステムSPHINCS Lightの構築

小沢　啓一郎*，竹内　雅志*，保田　敏雅*，藤田　眞作**

In-House Chemical Substance Database System SPHINCS Light

Keiichiro OZAWA*, Masashi TAKEUCHI*, Toshimasa YASUDA*
and Shinsaku FUJITA**

Abstract

An in-house chemical substance database system, SPHINCS Light, has been constructed within the LAN system of the Research Laboratories. All data of the original database SPHINCS, which is installed in a general-purpose mainframe computer, has been converted and transferred to SPHINCS Light. Yet the system architecture and technologies employed within SPHINCS Light are completely different from those of the original database. The server database engine ORACLE was selected from among the other RDBMSs within the market. The selection was made based on a comprehensive perspective, and was influenced by ORACLE's superior characteristics, including its program language system, its performance, and its search functions. An improvement in search time was enabled through the use of effective methods within the RDMBS, especially in terms of the effective control of database objects. Client program specifications have been designed with the chief emphasis on user operability. For the graphical user interface, technology has been employed which enables quick and easy operations. A unique substructure search system has been developed and implemented within SPHINCS Light. This system works in a special server computer that communicates with the main database server. The computational algorithm is comprised of a two-step searching procedure. The first step is a screening program that describes chemical structural information within a single tree file. The second is an atom-by-atom matching program which eliminates noise from the candidate data. Through the use of these processes, superior search performance is achieved.

1. はじめに

近年の計算機関連機器の高性能化・低価格化には目を見張るものがあり，従来は汎用計算機でしか行われていなかった大容量・高速処理を要する計算も現在ではワークステーションやパソコンに取って代わられつつある。また，ネットワークが発達することにより，自由にメールやその他の情報を交換することが可能になって，研究者は情報の取捨選択に集中すればよいようになってきている。

当社では，社内化合物データベースSPHINCS[1)]を汎用計算機の上で運用してきたが，汎用計算機の専用端末数の制限などから，システムの利用頻度はある程度制限されていた。しかし，上記のようなコンピューターをめぐる状況の変化に対応して，研究所内にLANが敷設された。各職場のユーザーが，自分のパソコンからLANに直接アクセスできる環境が整ったため，データの有効活用を考え，LAN上にデータベースシステムを移行する作業に取りかかった。新しく構築したSPHINCS Light[2)]では，汎用計算機で達成されている機能・パフォーマンスを保ったまま，ローカル上の利点をなるべく応用できるようにした[3)~6)]。

本誌投稿論文 (受理1997年9月9日)
*富士写真フイルム (株) 足柄研究所
〒250-0193 神奈川県南足柄市中沼210
*Ashigara Research Laboratories
Fuji Photo Film Co., Ltd.
Minamiashigara, Kanagawa 250-0193, Japan

**京都工芸繊維大学工芸学部物質工学科
〒606-0962 京都市左京区松ヶ崎御所海道町
**Department of Chemistry and Materials Technology
Faculty of Engineering and Design
Kyoto Institute of Technology
Goshokaido-cho, Matsugasaki, Sakyo-ku, Kyoto 606-0962, Japan

ローカルコンピューター上にデータベースを移行することにより，リレーショナルデータベース管理システムによる検索時間などのパフォーマンスは，低下するどころか汎用計算機よりも良好な結果を得ることができた。部分構造検索では，木構造型データによるスクリーニングという新しい検索手法を用いることによりパフォーマンスを数段向上させることができた[6]。また，数百台の研究所内LAN端末 (パソコン・ワークステーション) から利用が可能になったため，研究者による利用度も飛躍的に向上した。

2. SPHINCS Lightの前身としてのSPHINCS

SPHINCS Lightの前身であるSPHINCS[1]は，現在も稼動している汎用大型計算機 (富士通M1800/20) 上のシステムで，データベース管理システムとしてADABASを使用している。データベースの対象は，社内で合成または使用されている化合物で，データ項目には，化学構造式・分子式・分子量・社内番号・化学的性質 (mp, bpなど)・写真性 (感度など)・素材安全性 (LD50など)・社内技術報告書データなどを含んでいる。

化学構造式は多岐にわたる化合物 (低分子化合物から，各種の高機能性化合物，機能性ポリマーなど) を扱う必要から，繰り返し構造の表記 $\left(\quad\right)_4$，ポリマーの表記 $\left(\quad\right)_n$，Markush表現をサポートしている。また，さまざまな商品の研究開発に使用される素材のデータベースという性格から，機密情報管理に重点を置いた設計がなされており，パスワードによるアカウント管理はもとより，特別な機密管理用のデータを用意してデータの安全性を保証している。

検索には汎用大型計算機専用のグラフィックディスプレイ端末を用い，化学構造式の描画により，文字情報以外にも構造式による検索も可能になっている。また，専用端末は当社の各事業所に設置してあり，単一事業所のみならず幅広くネットワーク内で利用されている。

このように社内化合物に特化したシステムではあるが，利用に専用グラフィック端末を必要とすること，検索操作にある程度の熟練が必要なことなどの利用面での課題があり，研究者がパーソナルコンピューターを活用する環境にあっては，パーソナルコンピューターで手軽に利用できる検索システムへのダウンサイジングは自然な流れといえる。そこで，研究所内にLANが敷設されたことを機に，LAN上にデータベースを移行することを計画した。市販システムの利用は，多岐にわたる化合物の扱い面，機密管理面から容易ではなく，また，構造処理のノウハウを維持発展させていく意味でも独自に開発を行うこととし，SPHINCS Lightの構築に取りかかった。

現在，研究所LAN上で稼動しているSPHINCS Lightは，このような汎用大型計算機上のSPHINCSの化合物データを受け継いでいるだけでなく，前述のようなシステム設計思想も継承している。

3. データベースシステムの選定とパフォーマンスチューニング

3.1 RDBMSの選択

社内化合物の多種多様なデータを管理するには，市販のデータベース管理システムを活用することが最も効率的であると考えられる。開発当時 (1992年) に，UNIX上でC言語使用可能なDBMSとして，ORACLE, UNIFY, INFORMIX, SYBASなどのリレーショナルデータベース管理システムを検討したが，SPHINCSの表のレイアウト(行，列の大きさなど)やプログラム言語体系(SQL言語使用可能の有無) の拡張性などからORACLEとUNIFYに候補を絞り込んだ。両RDBMSを実際にWS上にインストールし，パフォーマンスを比較した結果がFig. 1である。テストでは，1000行100列の10ヶの表をSQL言語で検索し，ヒットした行数と検索に要した時間(ELAPS Time) の関係を求めた。

Fig. 1から明らかなようにORACLEはヒット行数の増加に対してあまり検索時間が増加しないのに比べ，UNIFYは急な傾きで検索時間が増えている。また，ヒット行数が少ない部分でも2倍以上の検索時間の違いがある。これは，それぞれのRDBMSの言語体系を考えれば当然であることがわかる。UNIFYは，SQL言語から独自言語 (RHLI) を通してデータにアクセスしているため，オーバーヘッドが大きいというのが原因の一つと考えられる (Fig. 2)。

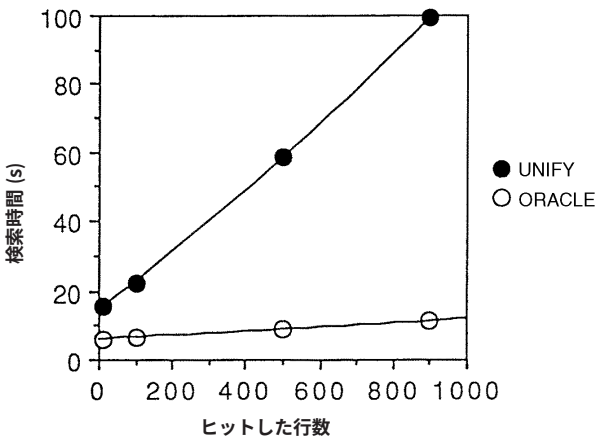

Fig. 1 Performance comparison between ORACLE and UNIFY

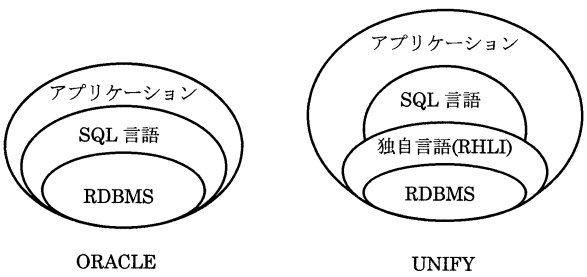

Fig. 2 Programming language systems of ORACLE and UNIFY

SQL言語は，動的検索も含めてコーディングが非常に容易なのが特徴であり，この言語に基づくORACLEの

プログラミングにおける優位性は明らかである。これに対して，同じことをUNIFYのRHLIで行うにはある程度の熟練が必要である。また，この他に分散問い合わせ機能・アプリケーション開発ツール・価格・サポート体制など，考えられる項目の比較の結果，総合的にORACLEが適当と判断し，サーバー用データベースに採用した。

3.2 検索部分のパフォーマンスチューニング

一連のデータモデルなどの仕様を作成したあとに，SPHINCSのデータをローカルコンピューター上のSPHINCS Lightに登録した。そして，検索部分の基本的な検索キーとして分子式・分子量・社内番号・登録番号を設定し，検索パフォーマンスの改良を行った。検索時間の改良には，RDBMSでよく知られている方法のいずれも有効である。特に，SPHINCS Lightの場合，①インデックス (B-Tree) の作成，②データベースからのデータの受け渡しに配列を用いる，③検索時に関係のない容量の大きいフィールドは別の表に分割する，などが有効に働いた。Fig. 3に改良を行う前の検索時間のグラフを示す。①，②の操作は，グラフの傾きを小さくすることに効果を示し，③の操作でグラフの切片を小さくすることに成功した。これらの改良で検索パフォーマンスは，Fig. 4で示されるように該当件数にあまり依存しない高速な検索が可能になった。

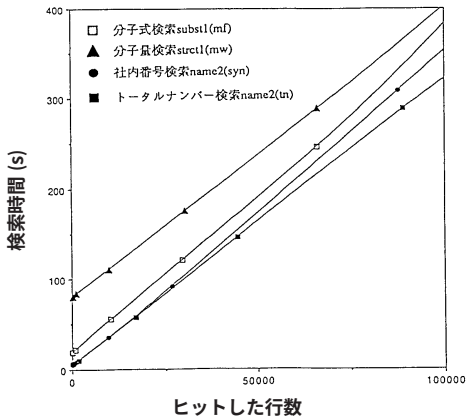

Fig. 3 Initial search performance of SPHINCS Light

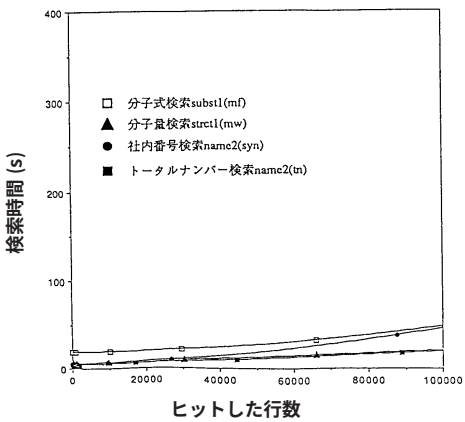

Fig. 4 Improved search performance of SPHINCS Light

ローカルワークステーション上 (ORACLE) と汎用計算機上 (ADABAS) で同様の検索を行った結果 (ELAPS Time) をFig. 5に示す。該当する行数が増えるに従い検索時間も増加するが，ワークステーションの方がその増分は少ない。この傾向は，分子式検索と分子量検索で顕著である。この結果からローカル化したワークステーション上でも汎用計算機と同等，またはそれ以上の検索パフォーマンスが得られることがわかった。

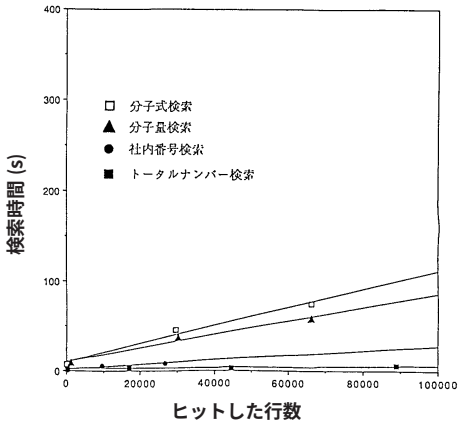

Fig. 5 Search performance of SPHINCS with mainframe computer

4. クライアントプログラムの作成

LAN (Local Area Network) 上のシステムであれば，わざわざデータベース専用の端末まで行かなくても，例えば，研究者が報告書などを作成している最中にでも，その場で気軽にデータベースにアクセスできる環境を提供することができる。さらに，使いやすいシステムであるためには，ユーザーが直接操作する部分であるクライアントプログラムの使いやすさは重要な要因であるといえる。

SPHINCSでは，細かな検索が可能な反面，コマンド形式の対話型オペレーションのため，操作が複雑で熟練が必要であった。そこで，SPHINCS Lightのクライアントプログラムは，初心者でも簡単に操作できるような「使いやすさ」に重点を置いて作成した。まず，DOS版のプログラムの開発を終え，現在，Windows版の開発を進めている。以下にその特徴を述べる[3]。

4.1 カード型検索条件設定画面

検索条件の設定画面は，一画面内に部分構造式やテキスト条件などすべてを設定するカード型にした。画面上の目的の項目に条件を入力する方法で，操作がわかりやすく，また複数の条件を入力して一度に検索できるメリットもある。検索履歴の参照や，検索結果を再度参照することも容易に行える。

4.2 検索条件の書式チェック機能

入力書式は，省略書式や書式に自由度 (論理演算のAND は&記号，＊記号あるいはANDのいずれでも可など)があり，省略書式で入力した場合は適切な書式に変更して表示する。また，入力書式に反して条件を設定

した場合にエラーメッセージで注意を促す。ヘルプ機能もカーソルの場所に応じた説明を表示するようにして，なるべく初心者にも理解できるように記述してある。

4.3 検索結果の表示

検索結果は，検索条件の該当部分 (社内番号や部分構造) をハイライトして表示する。また，検索結果の通信データの圧縮とバッファリングによりストレスを感じさせない表示速度を実現している (Fig. 6)。

Fig. 6 An example of substructure search result

4.4 構造式作成ツール

部分構造式を指定するために使う構造式作成ツールは，簡単に素早く目的とする化学構造を描画できるように工夫して設計した。単純に構造式を描画する際には，マウスのボタン操作をなるべく少なくするように工夫した。また，鎖状構造をキーボードで一度に入力できる機能もある。さまざまな骨格や置換基を集めた構造式メニューもスペースキーで簡単に呼び出せるようになっており，特に，当社で多用されている化合物群のメニューが豊富で，検索時に誤りがなく，迅速に描画できるようになっている (Fig. 7)。

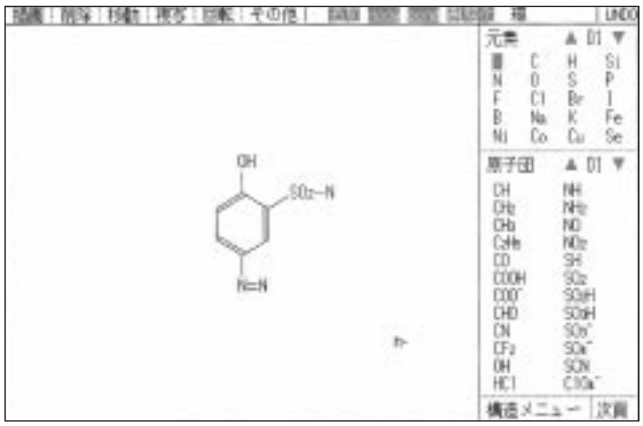

Fig. 7 An example of making query structure

4.5 ワークステーション端末

研究所LAN上の端末には，パソコンの他にUNIXワークステーションも各職場に設置してあるので，X-Window (XView) に基づいたGUI端末プログラムも作成した。パソコン同様，マウスによる操作で検索し，素早く化学構造式を表示できるようになっている。データベースサーバーとの通信は，専用のサーバーデーモンが行っている。TCP/IPプロトコルに基づいたSocketライブラリーによるプログラミングでクライアント／サーバーシステムを構築した。

5. 部分構造検索システムの構築

SPHINCSの部分構造検索については，フラグメントスクリーニング → Atom-by-atom matchingの2段階処理を行っている。一方、SPHINCS Lightでは，フラグメントスクリーニングの代わりに，独自の木構造型データを用いたスクリーニングシステムを新たに開発した[6]。この改良により，検索時間・スクリーニング選択性ともに従来の方法を大きく上回る性能を得ることができ，部分構造検索全体としても優れた性能を発揮することが可能になった[5]。

5.1 木構造型データによるスクリーニングのアルゴリズムとデータ構造

5.1.1 木構造型ファイルのデータ構造

スクリーニング用の検索ファイルは，すべてのファイル構造 (データベース中に登録されている化学構造式) を木構造型のデータに展開して得られる。化学構造式の結合関係を表す結合表を登録の際に作成し，各ノード (構造式中の元素) の属性を，2バイトのアトムコードと呼ばれる数値に変換する。木構造型検索ファイルに登録する工程を尿素を例に取ってFig. 8とFig. 9に示した。

(1) 水素を除くすべてのノードの中からルートノードを無作為に選択し，各ノードにおける構造に関する情報 (元素，分岐度，一重結合数，最小環員数，最大環員数，電荷など) を2バイトのアトムコードに変換する。Fig. 8では酸素原子がルートノードとして選ばれ，アトムコードaが得られている。

(2) ルートノードの第2層に属するノードを求め，同様にしてアトムコードを算出する。Fig. 8では炭素原子が第2層にあり，アトムコードbが得られている。

(3) 同様にして第3層以降のノードも順に行い，すべてのノードをアトムコードに変換する。変換が終了したら，アトムコードをルートノードに近い順に並べ，最後にその構造式に固有な登録番号 (RN5) を追加する。Fig. 8の右にあるように第3層の2つのアトムコードcは，同じ層に属するということで，他とは区別されたデータ構造を取っている。このアトムコードの連続データを「アトムコードリスト」と呼ぶことにする。この変換をすべてのノードをルートとして取ることで繰り返し行うと，アトムコードリストが構造式のノードの数だけ生成されることになる。Fig. 8の場合，2つの窒素原子は等価なので，3つのアトムコードリストが生成される。

(4) 上記の工程をデータベース中の他のすべての構造式について行う。変換されたすべてのアトムコードリストは，アトムコードの値でソートされ，Fig. 9の左からFig. 9の右のように木構造型ファイルに変換される。木構造型ファイル中では，第n番目のファイルは，第n層のノードのアトムコードを格納してあることになる。この結果，木構造のルートファイルにはすべてのルートノードのアトムコードが格納され，木構造の第2ファイルは，すべての第2層のノードのアトムコードが格納されている。それぞれの木構造ファイル中には，子の木構造ファイル中の特定のアトムコードへのアドレス (Fig. 9の下向き矢印) が格納してあり，検索時にはこれをもとにデータを読み込む。

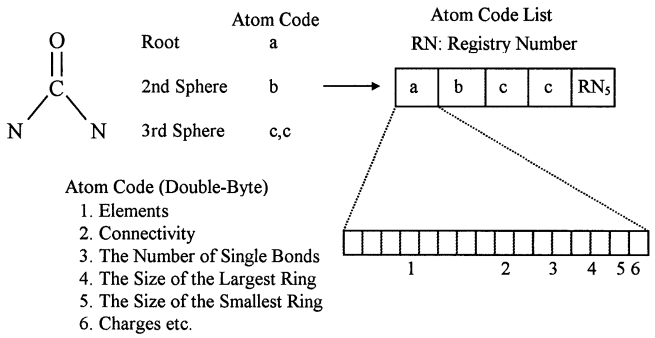

Fig. 8 Conversion from structure to an atom code

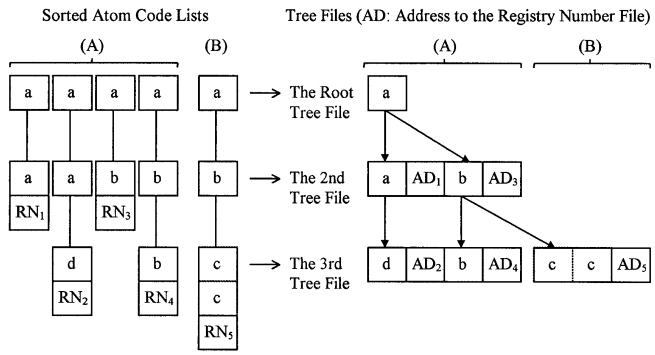

Fig. 9 Data format of atom code lists and tree files

例として，Fig. 9で最初の4つのアトムコードリスト(A) がすでに登録されて，新たに尿素のアトムコードリスト (B) を登録する時を考える。Fig. 9の右にあるように，登録する (B) のルートアトムコードaを木構造型ルートファイルのアトムコードと比較する。木構造型ルートファイルにはすでにアトムコードaが登録されているので，新たに加える必要はない。木構造型ルートファイルのアトムコードaには，子の木構造型ファイルのアトムコードa，またはbへのアドレスが格納されており，(B) の第2層のアトムコードと比較すると後者が一致する。木構造型第2ファイルのアトムコードbからは，子のファイルのアトムコードbへのアドレスがあるが，(B) の第3層アトムコード c, cとは一致しない。そこで，これらのアトムコードを木構造型第3ファイルの最後に追加する。そして，これらのアトムコードへのアドレスを親のファイル(木構造型第2ファイル)のアトムコードbに追加する。

このようにして，データベース中の80,000件の化学構造式を木構造型ファイルに変換すると，105個の木構造型ファイルが生成された。これは，第105層までのノードをもつ化学構造式が存在することを意味している。

5.1.2 登録番号ファイル

アトムコードリストを木構造型ファイルに変換する際に，登録番号ファイルも同時に作成される。登録番号ファイルは，データベース中の各構造式に付けられたユニークな登録番号のシーケンシャルファイルで，それぞれのアトムコードリストが登録し終わった時点でその登録番号がファイルに追加される。これと同時に，木構造型ファイルの最後のアトムコードと一緒に，追加した登録番号ファイル中のアドレスが格納される。Fig. 9では，登録番号へのアドレス (AD_5) がアトムコードc, cの隣に書き込まれている。最後のアトムコードには，そこから派生する子孫に属する登録番号の数も格納される。この値は，データベース中にそのアトムコードリストを部分的にもつデータの数を表しており，検索中に木構造型ファイルのアトムコードc, cでヒットした場合，アドレス (AD_5) にある登録番号から連続していくつの登録番号を読み込めばよいかを指定している。この登録番号ファイルを用いることにより，木構造型ファイル中で該当登録番号を集める必要がなくなったため，検索に必要な時間が大幅に短縮された。

5.1.3 質問構造データ

検索に使用する質問構造データは，ユーザーの指定する部分構造式を上記と同様の方法で，アトムコードリストに変換して得られる。つまり，化学構造式の各ノードの属性をアトムコードに変換し，それらを結合してアトムコードリストを作成し，検索キーとして使用する。木構造型検索ファイル中のアトムコードにない，質問構造データに特徴的なデータ要素としてフリーサイトがあげられる。フリーサイトは，そのノードを置換部位とする置換可能な結合数を意味する。木構造型検索ファイルと同様に，質問構造式でもすべてのノードをルートノードとして検索キーを作成するの

で，構造式のノードの数だけ検索キーが生成されることになる。

5.1.4 スクリーニングのアルゴリズム

構築された木構造型検索ファイルを用いて，2つのスクリーニングアルゴリズムを検証した。ひとつは，生成されたすべての検索キーを用いる「全キー・スクリーニング」と呼ばれる方法で，一つ一つのスクリーニングの結果の論理積を取り，スクリーニングの選択率を上げている。もうひとつは，「単一キー・スクリーニング」で，前スクリーニングにより最良の検索キーを選択してから，それを用いてスクリーニングを行う方法である。以下に後者のアルゴリズムを説明する。

上述のように，単一キー・スクリーニングの第一段階である前スクリーニングでは，最良の検索キーを選択する。通常，検索キーは，質問構造式のノードの数だけ生成されるが，それぞれの検索キーによるスクリーニング結果は一様ではなく，得られる件数は検索キーにより異なる。一般的に，少ない件数を与えるスクリーニングプロセスは検索時間も短いため，最も少ないスクリーニング件数を与える検索キーを選ぶことは非常に重要である。前スクリーニングは，この最も少ないスクリーニング件数を与える検索キー，言い換えればスクリーニングの選択率の最も高い検索キーを選択する工程である。

まず，質問構造式から生成されたすべての検索キー (アトムコードリスト) を第4層のアトムコードまで取り出し，前スクリーニング用の検索キーを作成する。Fig. 10にこの工程を図示した。それぞれの数値はアトムコードを意味しているので，検索キー1のアトムコードは第6層まで広がっている。よって第5, 6層のアトムコード (151, 152, 153, 161) は削除され，前スクリーニング用の検索キー1 ’ が生成される。

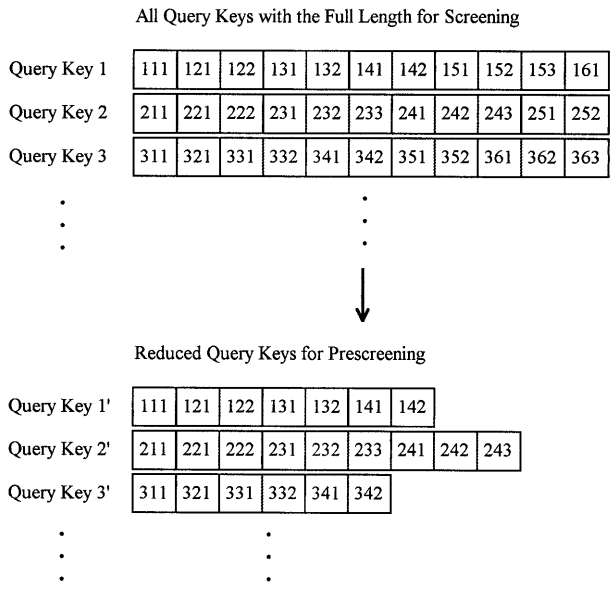

Fig. 10 Data format of query keys for the prescreening process

同様にして，検索キー2の場合，アトムコード (251, 252) を削除して，前スクリーニング用の検索キー2 ’ が生成される。このように簡略化したすべての検索キーを用いて，木構造型検索ファイルを走査しながら検索キーとの一致を調べる。

木構造ファイルの走査は，データの登録と同様に単純に木を渡り歩くことで実現される。まず，木構造型ルートファイルと，質問構造データのアトムコードリスト中のルートノードのアトムコードを比較する。Fig. 11の例では，アトムコードaが一致している。木構造型ルートファイル中のaには，第2ファイルのaとbへのアドレスが格納されているため，質問構造ファイルとこれらを比較する。aとは一致しないが，bと一致するので第3ファイルのアトムコードの比較へ進む。木構造型第3ファイルのアトムコードbと質問構造データの第3層bは一致するので，ここで質問構造データと一致するアトムコードの組が木構造型ファイル中に見出されたことになる。

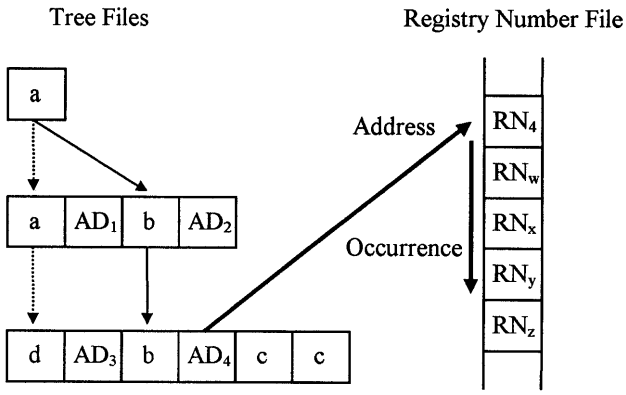

Fig. 11 A schematic diagram of a query key, tree files, and the registry number file

木構造型ファイルのアトムコードには，指定した質問構造データを部分的に持つ登録番号へのアドレスと，読み込むべき登録番号の数が格納されている。前スクリーニングではこの読み込むべき登録番号の数のみを参照し，実際には登録番号ファイルにアクセスは行わない。木構造型ファイルの走査中に，このように該当するアトムコードの組が見つかると，登録番号の値を次々に積算していく。

走査を終えて得られた積算値は，実際にその検索キーで得られる登録番号の数のインデックスとして使用される。インデックスの大きな検索キーほど，実際のスクリーニングで得られるデータの件数も大きいと考えるのである。前スクリーニングが終了したら，そのインデックスで検索キーを並べ替える。そして，最も小さいインデックスの検索キーが最良の検索キー，つまり最も少ない検索件数を与えるものとして，次の工

程の実際のスクリーニングに進む。

前スクリーニングの後の実際のスクリーニングでは検索キーを簡略化せず，全アトムコードを用いて木構造型ファイルの走査を行う。すべてのアトムコードが一致したら，今度は登録番号の読み込み数を参照するだけでなく，登録番号ファイルから登録番号を実際にその値だけ読み込む。得られた結果がスクリーニングによる検索結果となる。

なお，全キー・スクリーニングと単一キースクリーニングの比較，前スクリーニングで最良検索キーが得られる確率やテスト用質問構造式でのスクリーニング結果，既存のフラグメントスクリーニングとのパフォーマンス比較と考察などについては参考文献[6]を参照していただきたい。

5.2 Atom-by-atom matching プロセス

5.2.1 検索ファイルのデータ構造

スクリーニングで得られた候補構造式からノイズ構造式を除き，質問構造式を確かに包含するデータを求めるプロセスがAtom-by-atom matchingである。当然，このプロセスには抜けがあってはならないし，アルゴリズムの完全性が要求される。

検索のアルゴリズムは，単純なAtom-by-atomまたはnode-by-nodeと呼ばれる手法で，比較すべき2つの化学構造式のノード属性と結合属性を結合関係にしたがって逐一比較するものである。したがって，比較する化学構造式のノード数が多いものに対しては，検索に時間を大量に消費する。また，スクリーニングで得られた候補数が多い検索に対しても，マッチングに要する時間はもとより，候補の検索データを読み込む時のファイルI/Oが大きく検索時間に影響してくる。

atom-by-atom matching用検索ファイルは，1構造式に対して1つ作成され，全体として1つのシーケンシャルファイルを構成する。データは登録番号順に格納されており，ファイルレイアウトはできるだけ短い時間で検索できるよう設計した。使用される構造情報は，① 元素記号，② 立体･ラジカル情報，③ 電荷，④ 鎖･環情報，⑤ 水素数，⑥ 分岐度などで，スクリーニングに使用した情報と重なる部分もある。データは1回のアクセスで構造体に読み込み，繰り返し比較計算の行われるノード情報と結合情報は配列にすべて納め，単純なループで比較を行うようにした。

5.2.2 化学構造データの仕様とマッチング条件

atom-by-atom matchingでは，①から⑥までのノード情報や結合情報を繰り返し比較するため，プロセスとしての検索時間に占める割合が大きい。この部分の負担をできるだけ減らすため，マッチングの判断は以下のようなビット演算を用いている。

f(ファイル構造のデータ) & q(質問構造のデータ)
= f(ファイル構造のデータ)

5.2.3 インデックスファイル

スクリーニングで得られた候補データの登録番号から，Atom-by-atom matching用のデータを検索ファイルより読み込む必要があるため，検索ファイルのデータの位置を示すインデックスファイルが必要である。検索ファイルの各データへの位置とデータ長を，登録番号順に等間隔(6バイト)でインデックスファイルに格納する。

5.3 部分構造検索の総合検索時間

Table 1に，部分構造検索用サーバー (Sun Microsystems Ultra2 Creator Model 2170) 上でテスト用質問構造式を用いて検索したときの消費時間を示す。約9万件のデータについて部分構造検索を行った。10個のテスト用質問構造式は該当件数が30件から約7万件までをカバーしており，鎖構造や環構造に分れ，該当件数や化学構造に偏りがないように選出した。screeningとabamの値は，スクリーニングとatom-by-atom matchingの処理後のデータ件数である。前者を後者で割った値(rate)は，スクリーニングの選択性を表し，値が大きいほどスクリーニングによるノイズが少ない優秀なプロセスであることを示す。全体的には良好な値を示しており，0.9を超えるスクリーニングが半数を占めている。これにより，ノイズを対象とした無駄なatom-by-atom matchingの必要がなくなり，検索時間も短縮される。検索時間(time)の欄の加算式の左項はスクリーニング時間，右項はatom-by-atom matchingに要した時間を表す。下段はそれらの合計で，総合の部分構造検索時間を表している。Table 1から，平均の検索時間が8.0秒とユーザーに負担のかからない時間で終了していることがわかる。

Table 1 Substructure Search Performance of SPHINCS Light

query	1	2	3	4	5	6	7	8	9	10	Av.
time(s)	2+7 9	1+0 1	2+2 4	0+1 1	5+19 24	5+4 9	5+8 13	3+12 15	1+2 3	1+0 1	8.0
screen	72061	1081	17727	4290	36767	28193	73065	19353	3693	37	
abam	71467	136	4086	1741	36478	27665	71949	17659	2563	32	
rate	0.99	0.12	0.23	0.41	0.99	0.98	0.99	0.91	0.69	0.86	

6. おわりに

今回開発したSPHINCS Lightは，実際に研究所内LANを通して研究者に公開されており，4年以上が経過している。最近1年間の使用状況を見てみると，年間数百の端末から数千回のログインがあり，また，利用する職場もさまざまである。これを汎用計算機のSPHINCSの利用回数と比較すると，一桁以上利用頻度が増加していることになる。

開発に際しては，これまで述べてきたように，市販のデータベースシステムを除けば，関係するプログラ

ムはすべて自社開発で行っている。自社開発のデータベースの利点として，データの再利用が容易であることがあげられる。構造解析データをSPHINCSの構造活性相関システムに応用したり，市販のパッケージプログラムでデータを処理することが可能である。また，端末数が数百のシステムとして市販の化合物検索システムと比較すると，非常にコストパフォーマンスが良いことも利点の1つである。

参考文献

1) Hanai, S.; Miyakawa, M.,“ Registry and Search of Chemical Compounds on Graphics Display ”, Anal. Chim. Acta, **194**, 37-48 (1987)
2) 小沢啓一郎，保田敏雅，藤田真作，「社内化合物データベースのローカル化」，CICSJ Bulletin, **14** (1), 9-11 (1996)
3) 竹内雅志，小沢啓一郎，保田敏雅，藤田真作，「社内化合物データベース (SPHINCS Light) の構築」，日本化学会第72春季年会 (1997)
4) 小沢啓一郎，保田敏雅，藤田真作，「木構造型データによるスクリーニングシステム」，第18回情報化学討論会 (1995)
5) 小沢啓一郎，保田敏雅，竹内雅志，藤田真作，「社内化合物データベース (SPHINCS Light) の部分構造検索」，日本化学会第72春季年会 (1997)
6) Ozawa, K.; Yasuda, T.; Fujita, S.,“ Substructure Search with Tree-Structured Data ”, J. Chem. Inf. Comput. Sci. , **37** (4), 688-695 (1997)