データベースシステム 99－8
（1994．7．20）

タンパク質データベースの試作 検索機能について

吉田 正宏　米田 真治　中西 通雄　橋本 昭洋

大阪大学基礎工学部情報工学科

タンパク質の3次構造データを持つ Protein Data Bank(PDB) のデータをもとに、タンパク質を機能の観点から分類したオブジェクト指向データベースを試作した。タンパク質を実際の構造にあわせて複合オブジェクトとして表現でき、スキーマの直観的な理解が容易である。また、検索に用いる SQL では条件記述が複雑となる場合も、メソッドとしてデータベーススキーマに登録することにより利用者は SQL 検索文を簡単に記述できる。現在、本データベース上のグラフィカルユーザインタフェース (GUI) である *Protein Browser* の実装を行なっている。その主な機能として、仮想クラスを用いた効率的な検索、PDB ファイルを扱う既存のアプリケーションとのリンクなどがある。

Development of an Object Oriented Database of Protein Three Dimensional Structure Data

Masahiro Yoshida, Shinji Komeda, Michio Nakanishi, Akihiro Hashimoto

Department of Information and Computer Sciences,
Faculty of Engineering Science, Osaka University

Toyonaka, Osaka 560 Japan

An object oriented database of protein three dimensional structure data was developed to retrieve them in a brief way. Its schema classifies the data imported from the Protein Data Bank(PDB) according to the enzyme code, and represents each protein as a composite object with the structure based on protein itself. The schema also has several methods which simplify SQL statements for a complex query. A graphical user interface *Protein Browser* is available, which supports easy retrieval, linkage with applications on PDB data, and so on.

1 まえがき

分子生物学の分野では、立体構造が解析されたタンパク質のデータをもとにアミノ酸系列を比較し、未知の構造を持ったタンパク質の構造を予測するといった研究が行なわれている。現在、タンパク質の立体構造の解析が進み多数のタンパク質のデータが蓄積されている。そのタンパク質データの自由な検索、参照を望む要求に対し、タンパク質の結晶構造解析データを収集したデータバンクとして Brookhaven の Protein Data Bank(PDB) が作成された [1]。PDB ではデータはテキストファイルとして格納されており、その検索には通常 Fortran プログラムが用いられる。しかし、利用者にとってプログラムの作成は必ずしも容易ではなく、プログラムを作成することなく検索を行ないたいという要求から、PDB のデータをもとに商用のデータベース管理システムを用いて様々なタンパク質データベースが試作されている。

関係データベース管理システムを用いたタンパク質データベースに BIPED(Birbeck Integrated Protein Engineering Database)[2] がある。BIPED では、10 個の関係を用いて PDB のデータを表現し、検索には SQL を用いる。関係データベースにおける検索では、一般的に関係間の結合を行なうことが必要であり、結合操作の頻度によっては性能に重大な影響を及ぼすことになる。また、タンパク質のアミノ酸残基系列に関する検索など、検索条件の記述が複雑となる場合が多い。

本データベースでは商用のオブジェクト指向データベース管理システム VERSANT を用いてタンパク質データベースの実装を行なった。実装にオブジェクト指向データベース管理システムを用いる利点は、以下の 2 点である。

- スキーマの理解が容易

  タンパク質の各構成要素をクラスとして表現し、*is-part-of* 関係で各クラスを関係づけることによってタンパク質を複合オブジェクトで表現できる。タンパク質の構造とデータベース上での表現を類似させることが可能なため、スキーマの理解が容易である。

- 検索手段としてのメソッド

  関係データベースでタンパク質データベースを実装した場合、その検索に用いる SQL では検索条件の記述が複雑となる。オブジェクト指向データベースでは、オブジェクトの振舞いをメソッドとして記述できる。データベース管理者が検索手段としてメソッドを用意することで、利用者は検索文を簡単に記述することができる。詳細は 2.2 節で述べる。

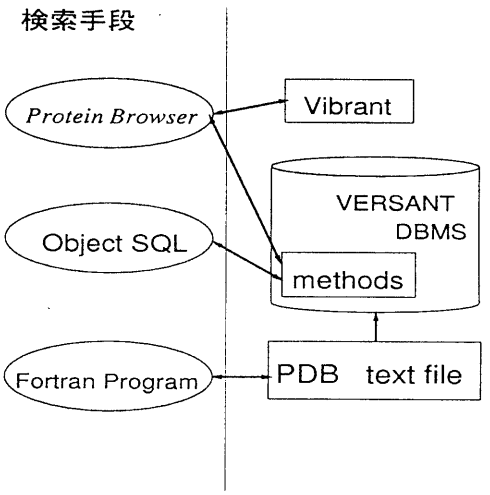

図 1: 本データベースシステムの構成

現在、本データベース上で動作するグラフィカルユーザインタフェース (GUI) として、*Protein Browser* の実装を行なっている。実装には、NCBI (National Center for Biotechnology Information) の GUI 構築ツールである、Vibrant を用いている。

図 1に示す 3 つの検索手段は、検索操作の容易さ、詳細な条件の記述力という点で異なっている。利用者は状況に応じて検索手段を選択すれば良い。

Protein Browser は検索の効率化、利用者の使いやすさなどを主眼において設計している。詳細については、3節で述べる。

2 データベーススキーマ

2.1 タンパク質の分類と構成

本データベースでは、タンパク質を機能の観点から、すなわち酵素の種類で分類し、クラス分けした。スキーマのクラス階層を図 2 に示す。

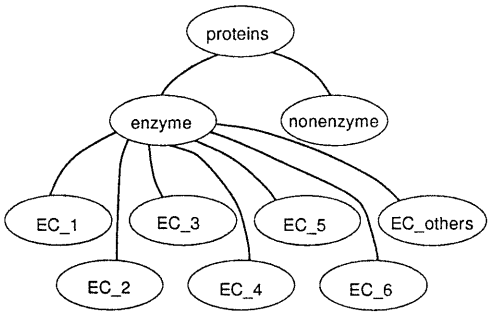

図 2: クラス階層 (*is-a*関係)

楕円はクラスを表し、楕円中の名前はクラス名を表している。以下、本稿ではクラス名を “proteins” のように表す。タンパク質のうち、酵素については ほとんどの場合、酵素番号[1]が割り当てられている。この酵素番号によって “EC_1” クラスから “EC_6” クラスのいずれかに分類される。酵素番号の割当のないものは “EC_others” クラスに、また酵素の働きのないタンパク質は “non_enzyme” クラスに分類される。これらのクラスは *is-a* 関係にあり、“enzyme” クラス、および “non_enzyme” クラスは “proteins” クラスのすべての属性とメソッドを継承する。同様に “EC_1” から “EC_6”、および “EC_others” の 7 つのクラスは “enzyme” クラスのすべての属性とメソッドを継承している。

次に、“proteins” クラスにおけるタンパク質の表現方法を示す。上述のクラス階層によって、すべてのタンパク質オブジェクトはこの表現方法を継承する。

個々のタンパク質は、アミノ酸がつなぎ合わされてできた鎖 (chain) と呼ばれる構造がいくつか集まって構成されている。鎖は 4 種の 2 次構造、ヘリックス (helix)、ターン (turn)、ループ (loop)、シート (sheet) から構成される。2 次構造は残基 (residue) の系列から構成され、残基は複数の原子 (atom) から構成される。これらの構成要素をそれぞれクラスに対応させて、1 つのタンパク質を複数のオブジェクトからなる複合オブジェクトとして構成する。これらのクラスは *is-part-of* 関係にある (図 3参照)。

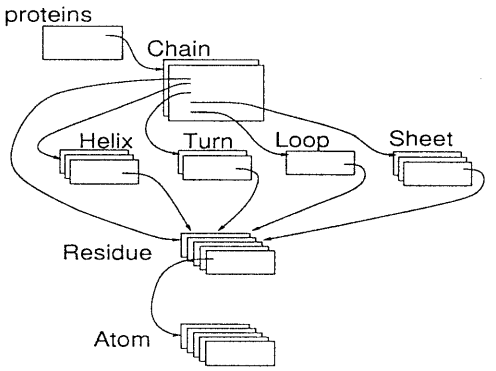

図 3: タンパク質を構成するオブジェクト群 (*is-part-of* 関係)

2.2 メソッドについて

オブジェクト指向データベースのスキーマでは属性やクラス階層構造の他に、各クラスにメソッドが定義される。オブジェクト指向のデータ隠蔽の概念により、属性値の参照を行なう場合にもメソッドが必要となるが、ここではそのような原始的なメソッド以外の、検索条件の記述を簡略化するメソッドについて述べる。

関係データベースの SQL では、込みいった質問に対する検索文は、where 節が複雑になる。BIPED において「“Ala-Gly-Ala” というアミノ酸残基系列を持つタンパク質を求める」場合、検索条件は次のように非常に複雑なものとなる。

[1]酵素を系統的に分類するための、4 組の数字からなる番号。

```
select S.BRCODE, S.PNAME
from  Structure S, Chain C,
      Residue R1, Residue R2,
      Residue R3
where S.BRCODE = C.BRCODE
  and C.BRCODE = R1.BRCODE
  and C.CHAINID = R1.CHAINID
  and R1.RNAME = 'Ala'
  and C.BRCODE = R2.BRCODE
  and C.CHAINID = R2.CHAINID
  and R2.RNAME = 'Gly'
  and C.BRCODE = R3.BRCODE
  and C.CHAINID = R3.CHAINID
  and R3.RNAME = 'Ala'
  and R1.POSITION + 1 = R2.POSITION
  and R2.POSITION + 1 = R3.POSITION
```

VERSANT の検索は Object SQL を用いて行なうが、属性値の参照のみで上記の検索を行なう場合は、BIPED と同様に検索文の where 節が非常に複雑なものとなる。これは、残基系列の通し番号を表す属性を用いて系列の連続性を表現することが避けられないためである。しかし、この問題はオブジェクト指向データベースのメソッドを利用することで解決できる。与えられた残基系列のパターンが、適用されるタンパク質に含まれている場合に真を返すメソッドとして、*judge_sequence* をデータベース管理者が用意する。データベース管理者がメソッドを実装する際に、C++を用いてアルゴリズムを記述する必要があるが、検索条件記述の複雑さをメソッドの内部に吸収させることで利用者の負担は軽減される。*judge_sequence* を用いれば上記の検索は以下に示すように簡単に記述できる。

```
select P.get_name()
from  proteins P
where P.judge_sequence("Ala-Gly-Ala");
```

このように SQL では条件の記述が複雑となる検索の場合でも、データベース管理者がその条件に対する真偽値を返すメソッドを用意することで、利用者の検索文の記述の負担を軽減できる。したがって、あらかじめ頻度の高い検索を調査し、それらを反映したメソッドを充実させることが重要となる。

2.3 スキーマの問題点

上述のタンパク質を表現するスキーマは、実際のタンパク質の構造に基づいたものであり、直観的な理解は容易である。しかしながら、データベースの実用性から考察すると以下のような問題がある。

1. 残基系列に対する検索の効率

   残基系列に対する検索を行なうためには各タンパク質の鎖ごとに "Residue" クラスのインスタンスを参照し、残基名を連接することで残基系列を生成しなければならない。この際に多数のオブジェクトの参照を引き起こすがそのオーバヘッドは無視できず、検索速度に強く影響を及ぼす。残基系列に対する検索は頻度の高いものであり、改善が必要である。

2. オブジェクトの総数

   "Atom" クラスのインスタンスの総数は莫大なものである。これらのインスタンスを同時に扱うような処理を行なう場合には、計算機の能力の大半をキャッシング処理などオブジェクトの管理に費やし、効率が悪い。

1. の解決策として、"Residue" クラスとは独立に残基系列を "Chain" クラスの属性値として持たせることを検討した。残基系列に対する検索にはこの属性値を用いることで、オーバヘッドを大幅に減ずることができる。

2. については、"Atom" クラスが持つべきデータを圧縮した形で格納し、必要に応じて展開して利用することで解決した。"Atom" クラスの各インスタンスが持つ情報は、タンパク質の構造を 3 次元表示するアプリケーションなどで用いられることが主であり、直接の参照は稀なためである。各 PDB のファイルの内容はほとんどの場合、原子座標データで占められている。そのため PDB ファイル全体を圧縮しても容量的には大差はなく、アプリケーションへの引き渡しなどを考慮して PDB ファイル全体を圧縮して保存している。

3 *Protein Browser*

2.2節で述べたように、メソッドを用いることで検索、参照といった処理を Object SQL で容易に記述できる。しかし Object SQL では、検索によって得られたタンパク質の個々のデータに対して新たな処理を行なうことは困難である。また、様々な条件に対応すべくメソッドを充実させた結果、メソッド数は増加し、利用者は多数のメソッド名とその機能を記憶する必要が生じる。これらの問題の解決、および検索に関する付加的な機能を実現するため、独自の GUI である *Protein Browser* の実装を行なった。図 4にその画面例を示す。

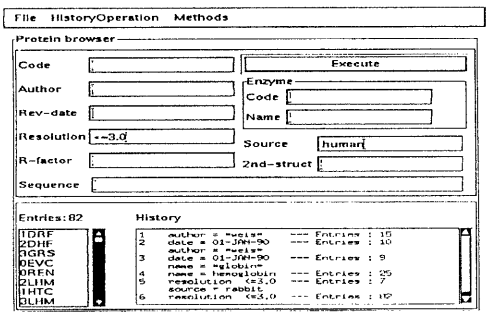

図 4: *Protein Browser*

Protein Browser では、タンパク質のいくつかの項目について、条件の入力を求める。その条件にしたがって検索を行ない、得られたタンパク質のエントリを表示し、各エントリの詳細な属性を参照できる。また、過去の検索結果の保存や集合演算、PDB データを利用するアプリケーションの起動などが行なえる。*ProteinBrowser* が持つ主な機能を以下に示す。

3.1 検索履歴

Object SQL には過去に行なわれた検索の条件を保持し、それを加工、修正することで後の検索条件の記述を簡便化する機能が存在する。*Protein Browser* では検索の効率化に重点をおいて検索履歴機能を設計した。

例えば、「採取源がヒトであり、かつ解像度が 2.0 以下のタンパク質を求める」という検索は、先に「採取源がヒトであるタンパク質を求める」検索が行なわれていれば、利用者はその結果の中から「解像度が 2.0 以下のタンパク質を求める」検索を行なうことができる。あるいは、「解像度が 2.0 以下のタンパク質を求める」検索も行なわれていれば、「採取源がヒトであるタンパク質」との集合積を求めることで目的とする結果を得ることができる。

Protein Browser の検索履歴は検索条件だけではなく、結果も含めて保持する。この機能により過去に行なわれた検索結果を再表示したり、新たに条件を付加した検索を効率良く行なうことができる。

3.2 仮想クラス

オブジェクト指向データベースにおける検索は特定のサブクラスだけを対象にして行なわれるものであれば、そのクラスから下位のクラスに属するインスタンスだけを参照すれば良く、すべてのインスタンスを参照する必要はない。したがって、クラス階層が頻繁に行なわれる検索に沿うものであれば、効率良く検索を行なうことができる。しかし、スキーマ設計者の持つ観点とデータベース利用者の持つ観点は異なることが通常であり、この場合はすべてのインスタンスを検索の対象としなければならない。

例えば、「酵素であり、かつ解像度が 2.0 以下のタンパク質を求める」という検索では “enzyme” クラスおよびそのサブクラスに対して検索を行なえば十分で、“non_enzyme” クラスのインスタンスについては考慮せずとも良い。一方、「採取源がヒトであり、かつ解像度が 2.0 以下のタンパク質を求める」という検索では、クラス階層の根である “proteins” クラスから始まって、すべてのサブクラスについてインスタンスを参照する必要がある。

そこで、利用者が持つ観点をデータベースに反映させるため、いくつかの頻度の高い検索に対して仮想クラスを設定することを考える。利用者は、各々が設定した仮想クラスを用いて検索を行なうことにより、検索の対象領域を狭め、検索を効率化することができる。

3.2.1 仮想クラスの実現

あるタンパク質が持つ「1.x.x.x」の酵素番号が割り当てられているという性質は、本データベースのスキーマでは "EC_1" クラスのインスタンスとなることで表現される。一方、タンパク質の採取源なる観点によって分類し、構成したスキーマを考えれば、その性質は属性酵素番号として値「1.x.x.x」を持つことで表現されるであろう。逆に、「ヒト」から採取したタンパク質であるという性質は本データベースでは "proteins" クラスの属性 **source** によって表されるが、後者では例えば "human" クラスのインスタンスとすることで表されるであろう。

すなわち、クラスと属性は本質的にはどちらも対象物の持つある性質を表すものであり、スキーマ上での表現の方法が異なっているに過ぎない。そこで、仮想クラスは検索文における属性値への制限、つまりSQL でいう **where** 節を用いて定義することを考える。利用者が「採取源がヒトであり、かつ (ある性質を満たす) タンパク質を求める」という検索を頻繁に行なうのであれば、「採取源がヒトであるタンパク質」なる検索条件をもって仮想クラスを定義し、データベースに保存する。以後の「採取源がヒトであるタンパク質」に対する検索は、この仮想クラスに対して行なうことで効率化できると考えられる。

本データベースの仮想クラスは、関係データベースにおけるビューの概念とほぼ同様であるが、PDBのデータを扱うためデータ隠蔽を行なう必要はなく、属性の射影は考えない。また、検索の効率化を目的とすることからビューとは異なり、検索文ではなく検索結果自体を保存する。実際には仮想クラスを表すオブジェクトが、条件に該当するタンパク質を表すオブジェクトの識別子を持っている (図5参照)。

3.3 仮想クラスの問題点

仮想クラスに関して生じたいくつかの問題について述べる。

1. データベース更新時の問題

   仮想クラスは検索によって得られた結果によって生成されるため、データベース本体に更新が起こった場合には、すべての仮想クラスについても同時に更新する必要がある。

2. メソッドの適用

   仮想クラスは、*is-a* 関係から敢えていえばクラス階層の根のクラスのサブクラスといえるかも知れないが、一般的には実在するスキーマのクラス階層とは独立している。したがって、仮想クラスのインスタンスにメソッドを適用する場合は、それらのインスタンスが実際に属しているクラスを意識して行なわなければならない。例えば、"enzyme" クラスにおいて定義されるメソッドに酵素番号を表示するものがある (*display_EC_number*)。一方、"proteins" クラスや "non_enzyme" クラスにはこのようなメソッドは定義されていない。仮想クラスのインスタンスの中には、実際に属しているクラスが "non_enzyme" クラスであるものが存在している可能性があり、*display_EC_number* は必ずしも仮想クラスのすべてのインスタンスに適用可能ではない。

1. については、データベース本体の更新が頻繁に行なわれる場合は大きな問題となる。しかし、PDBデータの更新の周期は比較的長期であるので、データベースの性能に大きな影響は与えないものと考えている。また更新を行なうために、各仮想クラスはオブジェクト識別子と合わせて、その仮想クラスを生成するための情報も保持している。

2. を解決する手段としては以下の方法が考えられる。

(i) クラス階層の根に当たるクラスに架空のメソッドを用意する

(ii) 仮想クラスのインスタンスへのメソッドの適用はクラス階層の根に当たるクラスで定義されたものに限定する

(iii) メソッドの適用は、仮想クラスのインスタンスが実際に属しているクラスを判断し、可能な場合にのみ行なう

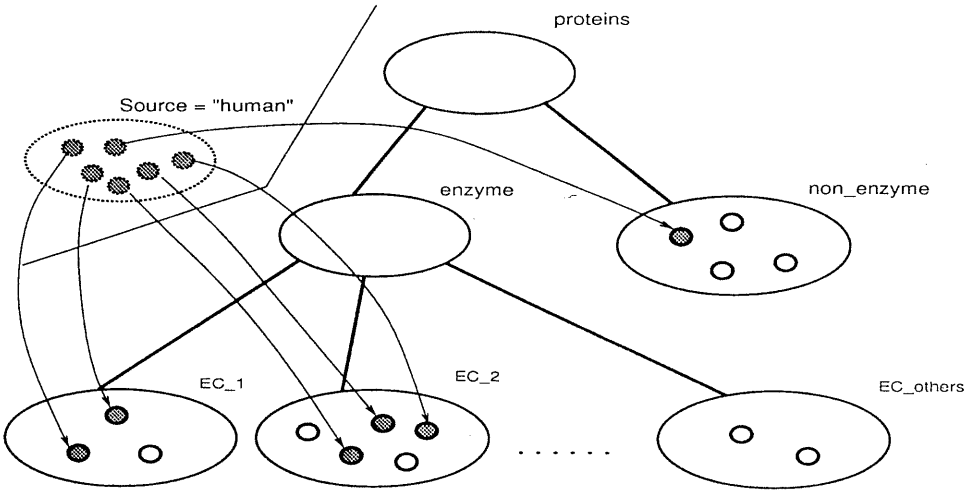

図 5: 仮想クラス “採取源がヒトであるタンパク質”

(i) はスキーマ自体に影響を及ぼし、さらにクラス階層の存在意義を失いかねないこと、(ii) では利用者への負担がかかり、メソッドが有効に利用できなくなることから、(iii) を用いてこの問題を解決している。

3.4 既存のアプリケーションとのリンク

検索を行なった結果、条件に該当したタンパク質のエントリに対して既存のアプリケーションを起動させることがよくある。このようなアプリケーションを起動するメソッドを用意することで、検索を行ないながら簡単にアプリケーションを起動することができる。2.3節においてデータを圧縮して格納することを述べた。PDB ファイル全体を圧縮しているので、このようなメソッドは単にデータを展開し、アプリケーションに引き渡すことで実現できる。現在、タンパク質の三次元立体表示を行なうアプリケーションである “RasMol” の起動が可能である。

4 あとがき

PDB のタンパク質の立体構造データを酵素という観点から分類し、商用のオブジェクト指向データベース管理システム VERSANT を用いてタンパク質のデータベースを試作した。タンパク質を複合オブジェクトで表現することにより直観的にモデル化でき、データベース管理者がメソッドを提供することで利用者の検索記述の負担を軽減することが可能である。また、検索の効率化、検索を補助する機能を持った GUI、*Protein Browser* の実装を行なっている。

現時点での問題点、および課題を以下に挙げる。

- PDB ファイルからの自動変換

  PDB のフォーマットは計算機で処理することを想定していない項目が一部存在しているため、本データベースのデータへの完全な自動変換は非常に困難である。しかしながら、PDB の表記法を統一しようとする動きがあり、現在 Crystallographic Information Format(CIF) や Abstract Syntax Notaion One

(ASN.1) を用いた記述フォーマットの標準化が進められている。標準化後には完全な自動変換が可能であると期待される。

- 仮想クラスの指定

  3.2.1で述べた仮想クラスは現在、利用者が陽に指定する必要がある。仮想クラスの存在を利用者に意識させることなく、検索条件から最適な仮想クラスを自動的に判断し、利用する機構を検討している。

- 同一の対象物を表す複数のオブジェクト

  本データベースにはタンパク質データを発表した著者を表すクラスがある。オブジェクト指向の概念にしたがえば、一人の著者に対しては唯一つのみオブジェクトが存在すべきである。しかし、ある著者の名前が複数のPDBファイルで全く同じ表記で記されている保証がなく、逆に同一表記の別人が存在する可能性も否定できないため、現状ではPDBの記述にごとに著者を表すオブジェクトを生成している。

- 同義語辞書

  タンパク質名や採取源名などでは、同一のものに様々な呼称が存在する場合がある。同義語を判断する辞書を利用し、検索機能を向上させることを検討している。前項の問題も、辞書の利用による解決が考えられる。

今後、上記の課題の解決、検索メソッドや起動できるアプリケーションを充実させることに取り組む予定である。

謝辞

本研究にあたりPDBについて適切な御助言をいただきました大阪大学蛋白質研究所の楠木正巳博士に感謝致します。

参考文献

[1] Bernstein,F.C., Koetzle,T.F., Williams, G.J.B., Meyer,D.F., Brice,M.D., Rodgers, J.R., Kennard,O., Shimanouchi,T. and Tasumi,M.: "The Protein Data Bank: A computer-based archival file for macromolecular", Journal of Molecular Biology, vol.112, pp.535-542(1977).

[2] Islam,S.A. and Sternberg,M.J.E.: "A relational database of protein structures designed for flexible enquiries about conformation", Protein Engineering, vol.2, no.6, pp.431-442(1989).

[3] Tanaka, H.: "Integrated System for Protein Information Processing", Proceedings of The International Conference on Fifth Generation Computer Systems, pp.321-329(1992).

[4] 田中 克己: "オブジェクト指向データベースの基礎概念", 情報処理, vol.32, no.5, pp.500-513(1991).

[5] Tanaka,K., Yoshikawa,M. and Ishihara,K. : "Schema design, views and incomplete information in object-oriented databases", Journal of Information Processing, vol.12, no.3, pp.239-250(1989).