7 バイオデータベースの技術的問題点

佐藤賢二
北陸先端科学技術大学院大学
知識科学研究科
ken@jaist.ac.jp

特集のまとめとして，本稿ではまずバイオデータベースの特徴を簡単に整理し，科学データベース管理の一般的な問題点のうちこの十数年で解決されたものとそうでないものを明らかにした上で，筆者が考える技術的な問題点と，今後バイオデータベースが進んで欲しい方向についての期待を述べる．

バイオデータベースの特徴

バイオデータベースの分類軸

バイオデータベースについては，以下のような数種類の分類が可能である．

- 内容に関する大分類（エントリが物質に対応するのか，現象や概念に対応するのか，文献などの情報体に対応するのか，etc.）
- 内容に関する小分類（配列，立体構造，発現，相互作用，疾病，文献，辞書，etc.）
- 配布や使用権に関する分類（パブリックデータベースか商品か）
- ファイル形式に関する分類（フラットテキスト，XML，イメージ，バイナリ，etc.）
- 加工の程度に関する分類（ファクトデータベース，2 次データベース，統合データベース，etc.）

たとえば，GenBank は「内容的には核酸配列（物質），配布や使用権に関しては基本的にパブリック，ファイル形式はフラットテキスト，加工の程度についてはファクトデータベース（1 次データベース）」と言うことができる．内容の点を除けば，伝統的なバイオデータベースの多くはこれと同様の分類に入る．ほかにも，格納されているデータ本体（GenBank なら文献情報や生物種情報などの付加的なデータではなく，核酸配列そのもの）のデータ構造に基づいた分類が考えられる．たとえば，核酸やアミノ酸の配列データベースはすべて「文字列」，タンパク質立体構造データベースは「3 次元空間の座標集合」，相互作用データベースは「2 項関係」，パスウェイデータベースは「グラフ」，モチーフやドメインデータベースは「文字列パターン」や「確率モデル」や「クラスタ」，などと分類できる．

データの記述形式

GenBank のエントリを例に，典型的なバイオデータベースの記述形式を示す（図 -1）．これから分かるように，GenBank のようなバイオデータベースの 1 エントリには実にさまざまな情報が詰め込まれている．その記述形式に関する大まかな特徴を以下に挙げる．

a）エントリは複数のフィールドからなる．フィールドはサブフィールドに分かれていることもある．
b）フィールドやサブフィールドには名前（識別子）が付いている．エントリをパージングする時はこれを手がかりにする．
c）フィールドやサブフィールドの中に，さらに構造がある場合がある（例：デリミタで区切ってキーワードを列挙するとか，同じくデリミタで区切って生物種分類上の系統（パス）を示すとか）．
d）フィールドやサブフィールドの内容は複数行にわたる場合がある（1 行に収まるとは限らない）．

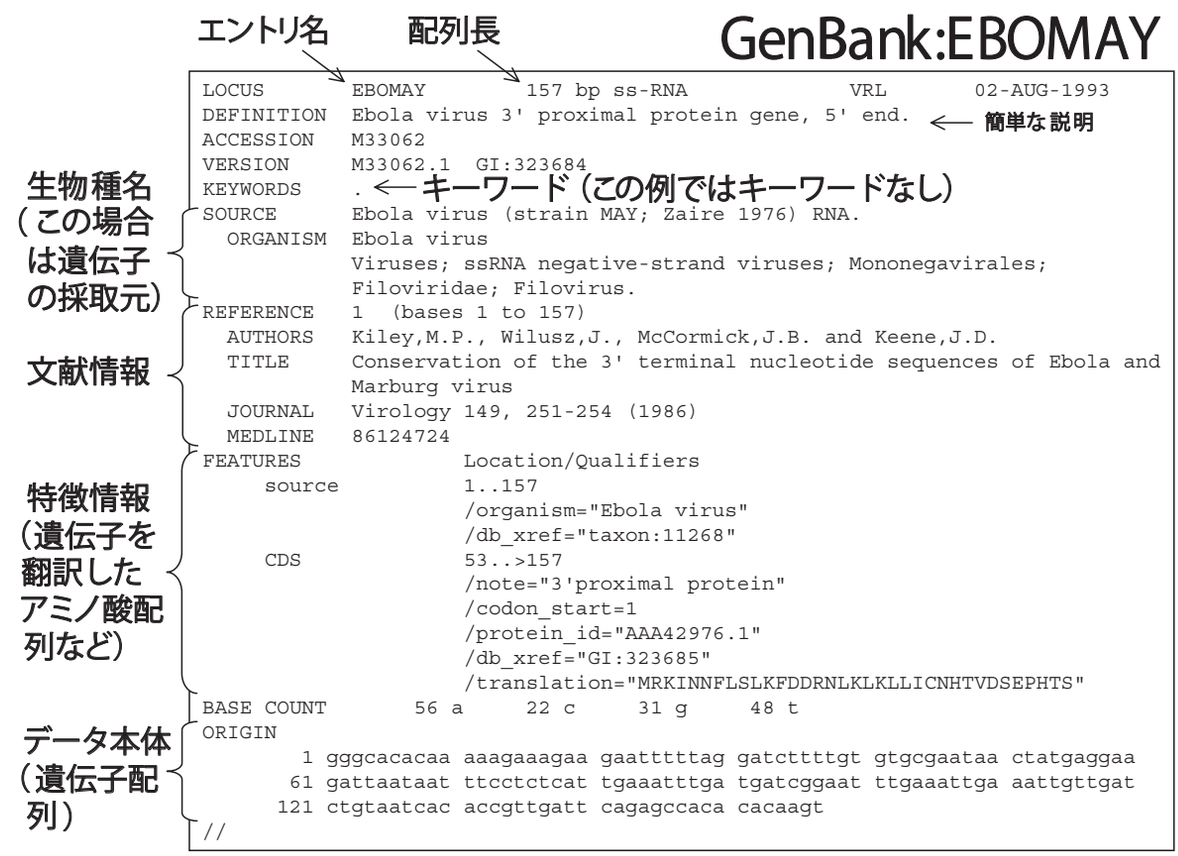


図 -1 GenBank エントリの例

e）フィールドのラインナップとして，エントリ ID やエントリに関する短い記述に加えて，文献情報や他のデータベースのエントリへのリンク情報を持つことが多い．

特徴 a）と c）からは，このようなフラットテキスト形式のデータベースのフォーマットが本質的に木構造をなしており，末端ではさらに詳細な構造があることが分かる．他にも，b）からは属性名と属性値の組でデータが記述されていること，d）からはデータ長が不定であること，e）からは他のデータベースエントリへのリンク情報を持ち得ることが分かる．このようなフォーマットは，共通のデータ交換形式として信頼できるのがフラットテキスト形式しかなかった時代に，人間が見ても読みやすく，機械的なパージングにも耐えられるように作られたものであるが，すでにお気付きの通りこれらの特徴は XML と非常に相性が良い．このため，近年では多くのバイオデータベースが XML 形式での配布をサポートするようになってきている．

データの源

多くのバイオデータベースは当初，特定の研究テーマに沿った小規模なデータコレクション（コンパイレーション）としてスタートしている [1]．たとえば，1965 年に Dayhoff が著した Atlas of Protein Sequence and Structure に収録されている 65 本のアミノ酸配列からスタートしたのが PIR だし，1985 年に Bairoch が配列解析に関する博士論文をまとめる際に PIR の不備を補う目的でフォーマット修正とアノテーション追加を施したものが Swiss-Prot の原形になっている [2]（その後，両者は UniProt に統合される）．このように，最初は既報の文献から特定の情報を抽出してまとめ上げるかたちでスタートし，後に資金や人員を得て継続的な拡大と発展を遂げるのが，大規模なバイオデータベースの 1 つの典型である．また，学術雑誌と連携し，たとえば「配列決定結果の論文を雑誌に投稿する場合は，必ず所定のデータベースにも登録データを送らなければならない」という義務を課すことも，データベースの拡大に大きな役目を果たしている．

このように考えると，バイオデータベースの中でもファクトデータベースに分類されるものの一部は，文献から知識抽出を行い整理したものと考えて間違いない．ではそれら文献に書いてある知識がどこから来たのかというと，基本的には文献の著者らが行った実験結果ということになる．文献に基づかないファクトデータベースもあるが，それらは 1 つまたは複数の実験グループが行った大規模かつ網羅的な実験の結果（たとえば酵母タンパク間の相互作用の有無を網羅的に調べた結果や，特定の生物種に関する遺伝子発現データを色んな条件で調べた結果など）を集積して整理したものであったりするので，やはりファクトデータベースの根源は実験結果であると言える．別の言い方をすると，ファクトデータベースは，フォーマットの決まった実験レポートを集積したものに相当するとも言える．この点は，気象データや天文データなどのようにセンサから連続的に吐き出される観測データとは根本的に異なる点である．つまり，エントリごとに実験者や実験条件，実験手法などが異なるため，精度や信頼性もそれに応じて多少なりとも変化するのが当然になる．もちろん，実験条件の設定や実験プロトコルの選択，試薬の管理などは厳格に行われ，高い再現性を有するデータが集められているわけだが，「同じ条件，同じ精度とは限らない」という点はバイオデータベースの特徴の 1 つとして憶えておくべきであろう．

16 年前の問題提起と現在までの対応状況

文献 3）は，地球科学，生命科学，宇宙科学などの自然科学分野の研究者と計算機科学者を集めて 1990 年 3 月に開催された科学データベース管理に関するワークショップのサマリーレポートである．ちなみに，生命科学分野からは Cold Spring Harbor 研究所で GDB（Genome Data Base）の開発に携わった Tom Marr も参加している．あまり古い話を持ち出すのは恐縮だが，本章では，このレポートで触れられている問題点が現在どのように解決ずみ（もしくは未解決）であるかを，この十数年の技術トレンドと絡めて考察する．

主要な問題（Main Issues）

◆メタデータ

文献 3）では，データに関する記述として，「誰がいつ何をした結果か」「実験に使用したデバイスの特徴」「データに対して行われた加工処理の定義」「そのデータに関する文献情報」「データの構造とフォーマットに関する記述」などが必要とある．これらは文献に基づく伝統的なデータベースではおおむね整備されているが，データに対してどのような加工処理が行われたかを正確に記述してあるケースは意外と少ない．

◆データの場所

文献 3）では「どんなデータがどこにあるか？」「そのデータは自分が探しているものか？」「有用なデータが存在するか？」などの質問に答える必要性を説いているが，これらは URL を基礎とする Web の発達と検索エンジンの普及および高度化により，現代ではほとんど解決されている．ただし，Web におけるデータの場所は永続不変であるとは限らないので，データの URL が変わることは今でもよくある．

◆ユーザインタフェース

文献 3）では「データ管理システムは主に計算機科学者が開発し，解析環境は主に分野の専門家が開発するため，これらの統合がうまくいかない」という問題を挙げており，これについては現在でも頭の痛い問題である．また，より良いインタフェースの条件として，「分野に最適化されていること」「初心者から専門家まで，色んなスキルレベルのユーザに対応できること」「種類の異なる，分散した DBMS をブラウズできること」「分野固有のアプリケーションのためのフックを提供すること」「ストレージの階層に容易にアクセスできること」「データが複数のソースからどのような加工を経て生成されたかを追跡できること」などが挙げられているが，Web と Web ブラウザの普及，ODBC や JDBC の普及，あるいは API の公開などにより，これらの条件は大部分充足されつつある．ただし，データの追跡可能性についてはほとんど解決されていない．

◆より柔軟な表現構造

文献 3）では「関係データモデルが科学データの取り扱いに不向きである」という問題が取り上げられている．科学データベースでは表形式データよりも時系列データや列データ，多次元データ，空間データ，画像データ，グラフデータなどを扱いたいことが多く，これらは集合論に基づく関係データモデルには馴染まない．これを解決する方向性として，関係モデルの拡張やオブジェクト指向データベース技術などが挙げられているが，データモデルの根本的な不適合は現在でも解決されていない．データモデルとして XML を用いれば，列や順序の概念は表現できるが，科学データの構造表現に関して XML が万能とはいえない．

◆適切な解析演算子

文献 3）では「大小や等号比較の演算子だけでなく，分野に適した柔軟な解析演算子の導入」の必要性を挙げている．これについては DBMS 内の基本演算子として持たせるのではなく，アプリケーションから DBMS が簡単に呼び出せればそれで十分という考え方もあるが，検索処理自体に分野固有の計算処理が必要な場合もある．

たとえば，最近の Oracle には BLAST 検索が組み込んであるため，SQL から BLAST 検索を行うこともできる．

◆標準化

文献 3）は，「データと解析環境の標準化，もしくはデファクトスタンダードの出現」が必要であることにも触れている．標準化についてはバイオデータベースでも多くの努力が払われており，デファクトスタンダードについてもおおむね機能している（FASTA フォーマットや BLAST アプリケーションはその好例）．

◆データ引用の標準化

文献 3）では「研究に使用したデータセットを明確に示せる」ことの必要性を挙げている．これについては，文献に基づくデータセットの場合はその文献を明示すれば十分であるし，バイオデータベースのエントリならそのエントリ ID を明示すればよい．また，研究者個人が URL のかたちでデータセットを公開している場合もある．

その他の問題（Other Issues）

◆データセットの転送

後の章でも触れるが，大規模データベースをネットワーク経由で転送するのに時間がかかるという問題は，現在でもあまり解決されていない．なぜなら，利用可能なネットワークの帯域幅が広がる一方で，データ量自体が増大しているからである．

◆ローカルな標準形式への変換

データベースを転送して来た後で，ローカルな解析システムが要求する形式への変換が必要な場合がある，という問題は今も変わらない．しかし，特にスクリプト言語の分野で，主要なデータベースのフォーマットをサポートするパッケージ（BioPerl や BioRuby など）が普及しつつあることにより，このような変換プログラムを作成するコストは格段に下がってきた．

◆データセット間の比較可能性

DBMS 関連の問題についても触れられているが，「有意味な比較を行うためには，それぞれのデータの意味を考慮に入れなければならない」という記述のほうが興味深い．言い換えると，異なるデータベース間で公平な比較を保証できるほど深く正確にデータの意味記述を行っている例は少ない．

◆マルチベンダ DBMS の相互運用性

これについては，DBMS が準拠すべき標準規格の普及や，ODBC や JDBC の普及により，ある程度解決されたと言える．ただし，ベンダ固有のフィーチャーがあるかぎり，この問題はなくならない．

◆データセットの質的評価

後でも触れるが，投稿されたデータの質を評価することは大変難しく，解決不能に近い問題と言える．ただ，文献 3）では「技術的な問題の多くは，評価をするには不十分なメタデータしか付随していないことによる」とあるが，フォーマットの整備や Web での投稿受付システム，あるいは投稿されたデータに対するチェックプログラムの整備により，必須のメタデータが欠落するというケースは，近年のバイオデータベースでは減少傾向にあると考えられる（記述量が少なすぎるケースについては，依然として改善していない）．

◆科学データの量的増大と永続保存の必要性

増大し続けるデータベースの保存について，ハードディスクの容量も問題だが，それよりも「データ生成には十分な予算が付くが，データ管理にはあまり予算が付かない」という問題が指摘されている．これは繰り返し指摘される問題で，実際，Swiss-Prot や GDB も過去に財政的危機に直面した歴史がある [1)]．

◆データの独占

データベースのコンテンツが，コストをかけて測定した実験データである以上，それをオープンにしたがらないことは多い．

◆データ管理作業自体の評価の低さ

財政問題に加えて，データ管理という仕事自体が研究者のキャリアとして低く見られがちであるという問題が指摘されている．

バイオデータベースに関する諸問題

現在のバイオデータベースについて，筆者は以下のような問題点があると考えている．特に区分はしていないが，前章で検討した古くからの問題と関連しているものもある．

- ゲノムネットのサイトにあるデータベース増大のグラフ（http://www.genome.jp/dbget/db_growth.html）が示すように，バイオデータベースは指数的な増加を続けており，その増加率は 3 年で 2 倍とも言われている．そのため，バイオデータベースもしくはそのコピーを継続的に更新し提供するセンタや研究室は，継続的な資源投入を強いられる．この資源の中で最もクリティカルなのはサーバ計算機のディスク容量であると思われがちだが，近年におけるハードディスクの低価格化と高性能化には目覚しいものがあるため，実際にはそれほど厳しい問題でもない（定期的なリプレースで十分対応可能）．過去数年間にわたってバイオデータベースの運用に携わった経験から言えば，真に重要なのは，計算機やディスク，電気代，設置スペース，ネットワークの帯域などではなく，運用に責任を持つ人員の確保である．なぜなら，バイオデータベースの

運用は，単に ftp ミラーサーバを立ち上げておくだけでは十分ではなく，フォーマットの変更に対応したスクリプトの書き換えやアプリケーションのバージョンアップ，計算機資源の増設，ファイル配置の最適化など，多くの局面で人手を要する作業が発生するからである．言い換えれば，バイオデータベースの運用にはいまだに専門知識を要する部分があり，規格の統一や標準への準拠に基づく自動化が進まない限り，誰でも簡単に運用するというわけにはいかない．

- バイオデータベースの各エントリは知的財産（IP）であるから，たとえパブリックに配布されているデータベースであっても，何らかの保護が行われるべきであるが，それがかえって運用の足枷になることもある．たとえば，PDB では各エントリの著作権は基本的にその投稿者（著者）にあり，PDB の配布元が勝手にエントリを書き換えることは許されていない．そのため，何十年も前に投稿されたエントリで，その記述の一部が現在では適切でないとしても，著者でない者がその記述を直接改めることは許されない．一般の書物や学術論文の場合，このような保護のされ方はまったく妥当であるが，急速に進展する生命科学の分野において，あまりにも古い記述や現在では否定されている記述をそのままにしておくのは問題である．特に，バイオインフォマティクスの分野ではできるだけ多くの情報を元に計算機で解析を行いたい場合が多いため，データベース全体に対してキュレーション（本特集 4.「バイオ知識の形成と表現」参照）を行い，最新の知識に即した記述に改められることが望まれる．

- 前述の問題については，後年明らかになった修正情報を別途用意しオリジナルエントリに添える（もしくは対応させる）ことにより回避することも考えられるが，基本的に「1 度書いたら書きっ放し」という特徴から来る問題はほかにもある．たとえば，仮にエントリ本体の内容が変わらなくとも，他のデータベースエントリへのリンク情報などは定期的に加筆修正を行わなければ陳腐化してしまう．この辺りは一般的な Web ページのリンク切れの問題と等質であり，究極的にはプログラムを用いた機械的なリンク付けや，最新情報に基づくオンデマンドでダイナミックなリンク探索による解決が望まれる（ゲノムネットの LinkDB は，リンクの連鎖をダイナミックに検索する機能を備えている）．

- 個々のエントリの記述は，基本的にその投稿者（著者）によって準備されるため，ヒューマンエラー（つまり著者による記述ミス）が入り込む可能性がある．大規模なデータベースには，厳密な投稿プロセスと人間が行う検査により誤記入を防止しているものもあるが，記述側も検査側も人間である以上，ミスを完全に除外することは難しい．大規模なデータベースになればなるほど，エントリの投稿頻度が多ければ多いほど，ミスの防止は難しい．自然言語で記述された個所にはスペルミスや表記の揺れが予想され，これらについては精選された用語辞書を用いて投稿前のドラフトを自動チェックすることである程度対処できるが，純粋なデータ領域（配列や構造，あるいは実験条件や実験結果の数値など）の正しさは投稿者に委ねられており，データベースの配布元でチェックすることは不可能に近い．ある意味では，実験機器が吐き出す測定データ自体は加工せずに，著者側では付加的な情報を添付するだけでそのまま投稿できるような仕組みを確立した方が，データの正確性は高まる．

- 文献 4）でも触れられているように，複数のバイオデータベースを組み合わせて利用できることのメリットは非常に大きく，そのためには分散したヘテロなバイオデータベースの相互運用性を高める必要がある．つまり，バイオデータベース間で正確かつ柔軟なリンク付けを行う必要があるのだが，そのためにはバイオデータベースの間に，ある種の統一性とアクセシビリティが保証されていなければならない．ところが，個々のデータベースの配布元では，知的財産としてのデータベースに対し何らかの権利保護を行いたいという心理が働くため，独自の制限条項やフォーマットを課してしまい，結果としてオープンかつ統一的なバイオデータベースの相互運用という理想はいまだに実現されているとは言い難い．これについては，バイオデータベースに適したディジタル著作権管理（DRM）の確立が必要だとも言われている．それと同時に，利用者のプライバシーやセキュリティを保護する方法についても検討が必要である．

- 上でも触れたデータベース増大のグラフによれば，1982 年当時の GenBank Release 1 は，わずか 440 個のエントリからなっていた．しかしながら，現在公開されている Release 151 には，52,016,762 個ものエントリが格納されている．この間，配布の形態にも各種の変遷があったと思われるが，基本的に計算機上では複数のテキストファイルに分割して格納されている．分割している理由は，以前の OS やアプリケーションでよく問題になったラージファイルの問題（2GB を超えるファイルを扱えない場合がある）を回避するためで，GenBank の場合はほとんどのファイルが 300MB 以下のサイズに抑えられている（例外は gbcon.seq で，約 800MB ある）．その結果，Release 151 では 854 個のファイルにエントリが分割格納されている．現代の OS や計算機ハードウェアをもってすれば，この程度のファイル数はさしたる問題ではないかもしれないが，PDB の場合は「1 エントリ＝ 1 ファイル」という単位で配布を行っているため，だんだん扱いにくくなってきている．PDB のエントリ数は 34,000 を超えており，同数のファイルをディレクトリに分けて格納している（PDB のエントリ名は 4 文字コードなので，その 2 番目 3 番目の文字をディレクトリ名としている．たとえば，ディレクトリ HV の下には，1HV6 や 8HVP などのエントリに対応するファイルが置かれている）．この状態では，たとえば Unix 上で ls ??/* というように全エントリファイルを対象にコマンドを実行しようとしても Argument list too long になってしまい，はなはだ使いにくい（もちろん回避方法はあるが，使いにくいことに変わりはない）．バイオデータベースの規模と，各時代におけるハードウェア／ソフトウェアの能力に応じて，整理や格納の方式を改めることが必要だと言える．

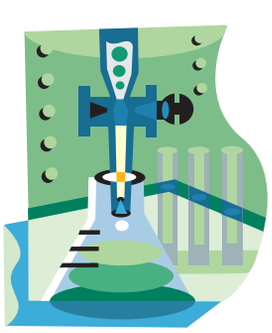
- 上ではファイル数が膨れ上がる問題に絞って説明したが，ファイルシステム絡みの問題はほかにも沢山ある．たとえば，人間にとっても可読であるようなテキストファイルとして表現したせいで，無駄な空白などが相当量入っている．GenBank を例にとり，ファイルの 1 つである gbuna.seq を調べてみると，424,442 バイト（8,727 行）のファイル中に 114,968 バイト（約 27%）の空白が含まれている．ほかにも，PDB エントリ 3 個（1X5S,1X5T,1X5U）を連結したファイル（7,262,703 バイト，89,663 行）の場合，実に 3,451,167 バイト（約 47.5%）の空白が含まれている．この違いは，GenBank の各行が可変長であるのに対し，FORTRAN プログラムによる処理を意識して作られた PDB の各行が固定長であることによる．これらの空白の中には単語間の区切りなど不可欠なものも多く含まれているが，純粋にパージングのため，つまり，エントリ内でフィールド名とその内容を区別したり，フィールド名とサブフィールド名を区別したりするためのデリミタやインデントに用いる連続空白も，大きなウェイトを占めている． もちろん，このような連続空白や，繰り返し出現するフィールド名，サブフィールド名などが占める容量は，圧縮ソフトにかければ劇的に削減されるのだが，アプリケーションとの連動の都合などの理由により，一般にバイオデータベースのユーザは展開状態のファイルを手元に置きたがる傾向がある．結局，展開後のファイルは，ディスク領域を無駄に消費してしまう．これらフォーマットに依存する無駄な繰り返し文字の問題は，XML 化することによりある程度解決が期待できるが，XML 化により新たなオーバーヘッドを抱え込む点にも注意が必要である．他にも，PDB エントリの本体である ATOM フィールドには，生体分子に含まれる原子の 3 次元座標が 1 行 1 原子の形式で延々とテキストで書いてあるなど，「データ本体を ASCII コードで表現することの無駄」も問題である．データ本体を圧縮テキストとして持つような XML を基本フォーマットとして，エントリを人間が読む場合やレガシーなソフトウェアに入力する場合のために，XML に基づいたコンバータを用意するような配布形態に移行できれば解決しそうだが，すでに多くのアプリケーションが依存しているフォーマットは簡単には変えられず，移行には長い時間が必要になる．

- 上で触れた GenBank Release 151 は，854 個のエントリファイルのサイズを合計すると，約 190GB になる．最近の ATA ディスクの読み書き速度は大体 60 〜 70MB/s であるから，単一のディスクに格納した 854 個のエントリファイルを単純に cat して /dev/null にリダイレクトするだけでも，40 〜 50 分程度はかかってしまう．バイオデータベースに対してどのような処理を行うかにもよるが，比較的軽い処理の場合，ディスク I/O の速度がボトルネックになってしまいがちである．これを解決するためには複数の実ディスクを用いてバイオデータベースを分散配置し，並列に読み出すことが必要であるが，単一の PC でこれをやろうとすると今度は PC 内部のバスの速度がボトルネックになる．そのため，大規模なバイオデータベースは PC クラスタなどに分散配置して，アプリケーション自体も PC クラスタの各ノードで並列かつ独立に走らせることにより，ディスク I/O 速度の合計を数十倍に引き上げ，短時間で処理を終わらせる，ということがよく行

われる．さらに，並列 BLAST などの例では，データベースファイルを分割してノードに分散配置することにより，各ノードにおけるデータベースファイルのサイズが小さくなり，OS のファイルキャッシュに収まるため，ディスク I/O 自体がキャッシングにより高速化されるという例もある．しかし，PC クラスタを用いたこのような並列処理は，現状では OS やハードウェアに関する知識がかなり必要なため，生物系の研究室では十分活用されているとは言い難い．

- バイオデータベースを用いた研究結果を論文に報告する場合，使用したデータセットを明示するためにデータベース名とリリース番号を示すことがよくある．また，毎日更新されるタイプのデータベースでは，それを取得した年月日を示すことがある．しかしながら，よほど小規模なデータベースでない限り，過去のバージョンのデータベース一式が別途保存してあることは稀であり，最新版を元に過去のバージョンを再現することも，原則的にはできない（一度登録されたエントリが削除も修正もされないことが保証されていれば，登録の日付から再現可能かもしれないが）．その結果，継続的に拡張と更新が行われているデータベースほど，論文に報告された計算処理の完全な再現が難しいという皮肉な結果が生まれる．これにより，同じデータベースを使って同じ予測問題を同じように解いたとしても，最新版のデータベースを用いるだけで精度が向上してしまう可能性が生じるため，過去に報告された手法との比較が難しくなる．

- 複数のファクトデータベースから計算機によって生成された 2 次データベースは，使用するファクトデータベースの更新に連動して再計算を行い，自分自身を更新することが望まれる．しかしながら，ファクトデータベースの更新は一般にメールや Web などで通知され，その様式は決まっていない．よってこの場合，ファクトデータベースの更新を検知するのは 2 次データベースを開発し運用する管理者であり，自動検知から自動更新という処理フローを確立している例は非常に少ない（同じサイト内では有り得るが，複数の外部サイトが提供するファクトデータベースの更新情報を自動検知して自動更新される 2 次データベースはほとんどない）．その結果，相当な人的コストをかけないかぎり，2 次データベースの更新頻度は低いものになりがちである．ほかにも，2 次データベースではデータ加工の追跡可能性の問題が必ず生じる．つまり，ある 2 次データを生成する元になった 1 次データ群を明示できない場合や，1 次データの一部が陳腐化により削除されている（存在しなくなっている）場合，アプリケーションの更新により 2 次データの生成過程が再現不能になる場合など，さまざまな問題が考えられる．



未来のバイオデータベースのために

ここでは本稿のまとめとして，いくつかの情報処理技術に絡めて，現在よりも一歩進んだバイオデータベースの可能性について論じる．

◆ Web2.0

インターネットの普及に伴い，Web を用いたバイオデータベースのサービスが一般的になった．代表的なサービスとしては，キーワード検索やサーバ側での解析アプリケーション実行，可視化などがあり，Web の特性上，インタラクティブなサービス（つまり人間が操作して結果を得るタイプのサービス）が多く提供されてきた．さらに，複数のデータベースや解析アプリケーションを Web 上で統合するために，近年では API というかたちでサービスを公開し，プログラムから自由に利用してもらおうという動きも盛んになってきた（例：KEGG API）．データベースを XML 化し，公開した API と組み合わせてサービスすることにより，少ないコストで新しいアプリケーションを開発することができる．特に，近年のバイオインフォマティクスでは，複数のデータベースやプログラムを組み合わせた複雑な処理が要求される場合が多いため，データベースの XML 化とサービスの API 化は重要な進歩である．しかし，XML 化したからといって個々のタグに囲まれたデータの意味が厳密に定義されたわけではなく，他のデータベースやアプリケーションと正しく組み合わせるためには，今でも人間の知識が必要である．

◆データの検索と取得

API の整備により，指定したエントリ 1 個の取得や，データベース全体の取得は，プログラムから容易に行え

るようになった．しかしながら，「指定した条件を満たすエントリ集合の取得」が API 化されている例は，それほど多くない．さらに言えば，検索処理自体をユーザ側で自由にプログラミングできるようにして欲しいところだが，あまり自由度を上げるとサーバへの不正侵入を許すことにもなりかねないので，安全かつ自由度の高い検索サービスを実現するのはなかなか難しい．

◆ポータルサイトの構築と API の統一

NCBI や EBI，ゲノムネットなど，大手のバイオデータベースサイトでは，データベース名とエントリ名を指定すればエントリがダウンロードできるような API が整備されており，同じサイト内ではデータベースが違っても API の形式自体は共通であることが多い．しかし，サイトが異なれば API も異なり得るし，何よりそのサイトにないデータベースには対応できない．結果としてユーザは，どのデータベースについてはどのサイトのどの API をどう使ってアクセスしなければならないかを常に把握しておく必要がある．このようなアクセスのためのメタ情報をどこか 1 カ所に集積し，統一的な API で利用できるようになれば，利用者の負担は相当軽減される．たとえば，Nucleic Acids Research 誌が毎年発行している Database Issue には，相当数のバイオデータベースが網羅されているが，これを一歩進めて統一的な API による検索やデータ取得が可能になれば，ユーザにとってはメリットが大きい．

◆データベース統合とオントロジー

Gene Ontology が普及した結果，研究者が着目している遺伝子集合の意味をオントロジーで解釈したり，クラスタリング結果の善し悪しをオントロジーに基づいて評価したりすることが頻繁に行われるようになった．これを発展させれば，オントロジーの利用により，バイオデータベース間で柔軟なリンク付けを自動的に行う（エントリ中の自然言語記述に基づいたリンク付けを行う）ことも考えられる．このようなリンク付けにより，バイオデータベース間の相互運用性が向上することが望まれる．

◆データベース取得の高速化

GenBank のような 200GB 近いデータベースを，他のサイトからデスクトップ PC のローカルディスクにダウンロードする場合，国内でも 1 日以上かかることが珍しくない（ネットワークの帯域幅にもよる）．しかも，多くの場合バイオデータベースの最新版はその配布元で最初に公開されるため，世界中のユーザが配布元のサイトに殺到すると，悲惨なことになる．伝統的には ftp ミラーサーバを各地に用意することにより，この問題に対処しているが，今後は BitTorrent などの P2P ソフトウェアが提供しているような並列拡散機構（つまり，細分化されたデータが多数のノードにばら撒かれると同時に，それらのノードは自分が所有するデータ片のサーバとして機能しはじめる）を導入するべきであると考える．さらに，利用者のサイトとインターネットを接続する帯域幅が十分に太い場合，大規模なバイオデータベースをローカルディスクにコピーすることなく，オンデマンドで転送して使い捨てるような利用法も考えられる．もちろんそのためには，多数のサーバが各地で稼働していて，並列転送要求に応えられることが必要である．

◆グリッド技術

前章でも触れたが，大規模なバイオデータベース全体を計算処理する場合，ディスク I/O がボトルネックになりがちであり，これを解決するには PC クラスタへの分散配置と各ノードにおける並列計算が必要である．これは広い意味では，クラスタコンピューティングを含むグリッドコンピューティングの問題として解決されるべきであり，そのための研究開発も多数行われているが，バイオの研究室でグリッドコンピューティングがポピュラーになったとは言い難い．導入のための敷居が現在よりも飛躍的に低く，より一層使いやすいソフトウェアやサービスの普及が望まれる．

参考文献

1）Galperin, M. Y. : The Molecular Biology Database Collection: 2005 update, Nucleic Acids Research, Vol.33, Database issue D5-D24 (2005).
2）Bairoch, A., Boeckmann, B., Ferro, S. and Gasteiger, E.: Swiss-Prot: Juggling between Evolution and Stability, Briefings in Bioinformatics, Vol.5, No.1, pp.39-55 (2004).
3）French, J. C., Jones, A. K. and Pfalts, J. L.: Summary of the Final Report of the NSF Workshop on Scientific Database Management, SIGMOD Record, Vol.19, No.4, pp.32-40 (1990).
4）Greenbaum, D., Smith, A. and Gerstein, M.: EDITORIAL: Impediments to Database Interoperation: Legal Issues and Security Concerns, Nucleic Acids Research, Vol.33, Database issue D3-D4 (2005).

（平成 18 年 2 月 3 日受付）