The National Center for Biotechnology Information (NCBI),¹ part of the National Institutes of Health (NIH), is responsible for massive amounts of data. A partial list includes the largest public bibliographic database in biomedicine (PubMed),² the U.S. national DNA sequence database (GenBank),³ an online free full text research article database (PubMed Central),⁴ assembly, annotation, and distribution of a reference set of genes, genomes, and chromosomes (RefSeq),⁵ online text search and retrieval

JAMES OSTELL, NCBI

DATABASES OF DISCOVERY

systems (Entrez),⁶ and specialized molecular biology data search engines (BLAST,⁷ CDD search,⁸ and others). At this writing, NCBI receives about 50 million Web hits per day, at peak rates of about 1,900 hits per second, and about 400,000 BLAST searches per day from about 2.5 million users. The Web site transfers about 0.6 terabytes per day, and people interested in local copies of bulk data FTP about 1.2 terabytes per day.

In addition to a wide range of different data types and the heavy user load, NCBI must cope with the rapid increase in size of the databases, particularly the sequence databases. GenBank contains 74 billion basepairs of gene sequence and has a doubling time of about 17 months. The Trace Repository (which holds the chromatograms from the sequencing machines for later reanalysis of genome sequence assemblies) contains 0.5 billion chromatograms and is doubling in about 12 months. Finally, because NCBI supplies information resources in molecular biology and molecular genetics, fields in a state of explosive growth and innovation, it must face new classes of data, new relationships among databases and data elements, and new applications many times every year.

**Open-ended database ecosystems
promote new discoveries in biotech.
Can they help your organization, too?**

This article briefly describes NCBI's overall strategic approach to these problems over the past 15 years, and the technology choices made at various points to follow that strategy. It is not intended as a tutorial in bioinformatics or molecular biology, but tries to provide sufficient background to make the discussion understandable from an IT perspective.

THE BASIS OF THE MOLECULAR BIOLOGY REVOLUTION
Biology has long been an observational and comparative science. For example, comparative anatomy finds points of similarity in biological entities such as the skulls shown in figure 1 and infers that they may serve similar functions in both organisms. Based on these correspondences we find that we can do experiments or make observations on one organism that we can apply to the corresponding structures in the other organism and infer similar results under similar conditions, even if we do not actually do the experiment on both organisms. When we find many similar experimental results on similar structures in different organisms, we may infer more general principles across a range of organisms. Moving this kind of work to computers is difficult for a number of reasons. Obtaining a sufficient number of samples for statistical analysis is often a challenge. It may be difficult to select or model the relevant properties of a complex biological

10 x 4 Angle Beam
1 2010 255.00
2 2010 255.00
3 2010 255.00
4 2010 255.00
5 2010 255.00
6 2010 255.00
7 2010 255.00
8 2010 255.00

Detail 8
Detail 1

645Mhz

7750-PTA

MPG2550

ARAST0040

FRAS0064

RTM 5000

Feed

7.02

10.971VITM

MPG2550

DATABASES OF DISCOVERY

shape or function, as there are a very large number of parameters that may or may not be relevant to function. For example, it may not just be shape, but also flexibility, composition, proximity to other structures, and physiological state, to name a few.

The sequence of a protein (or the DNA of a gene that codes for the protein) can be modeled as a simple string of letters, each representing a particular amino acid or nucleic acid. While the protein may fold up into a three-dimensional shape with many of the parameters for anatomical structures just described, the simple linear chain of amino acids appears to contain much of the information necessary to make the final shape possible. So rather than compare the final shape or charge distribution of the folded protein, one can compare the direct readout of the gene as the string of amino acid letters.

This is very simple to model on a computer, and there are many algorithms for comparing strings and extracting a statistical signal from them. Perhaps equally important is that we are not blinded by our assumptions as much. For example, when comparing anatomical structures such as a jawbone, we might be selecting parameters and measurements associated with chewing and miss the well-established fact that over time some bones associated with jaws have become involved with hearing. When we compare protein strings, we are not concerned with their names or assumed functions until after the comparison,

so we are more open to making novel connections.

Figure 2 shows the result of a BLAST search of a protein implicated in human colon cancer compared with the protein sequence database. There are significant hits to a protein from yeast (a small organism involved in making bread, among other things) and another protein from the *E. coli* bacterium that resides in the large intestine. Note that none of the words we know about the human protein apply to the other two organisms. Neither is human, neither has a colon, and neither gets cancer. A host of experimental results, however, describes the functions of these two proteins. It turns out that both of them are DNA repair enzymes. This immediately gives us an insight into why genetic damage to this protein in some humans may make them more prone to cancer. They are less capable of repairing damage from carcinogens to their DNA. Further, there are many published research papers by scientists working on yeast and *E. coli* and studying these proteins in experimental systems that are impossible to apply to humans.

By doing this computational search and comparison we can span millions of years of evolution and make associations between biological entities that look nothing alike. This brings tremendous insight and greatly accelerates the pace of biological discovery. We would not have found this information by mining the text of the articles or by examining the contents of the sequence records, but only by traversing linked information spaces and using computed relationships not present when the data was originally entered. This is the molecular biology revolution and why computation and databases are so essential to the field.

THE STRATEGY

When NCBI was created in 1988, the goal was to build an information resource that could accommodate rapidly changing data types and analysis demands, yet still have enough detail for each subject area to make meaningful computational use of domain-specific data. We did not want to bind the data to any particular IT technology, but instead be able to migrate the data as IT technologies evolved. We recognized that we should not develop specialized hardware for a niche market such as molecular biology, but instead adapt our problems to whatever the mass-market technology of the time was to maximize our price/performance ratio. We wanted to support the computers that scientists had ready access to and were familiar with, whatever they might be.

For these reasons NCBI created its primary data model as a series of modules defined in ASN.1 (Abstract Syntax

Notation 1). It is an established international standard (ISO 8824) with an explicit data definition language. It has both a human-readable text serialization and a compact binary serialization, enabling development and debugging with text data, then compact production data exchange by simply flipping a switch in software. The language was designed to be independent of hardware platform or programming language or storage technology. This makes it ideal for defining complex data in a stable, computable way, yet buying the flexibility to move the data, or even parts of the data, into storage, retrieval, or computing environments as opportunities arise.

The runner-up language choice at the time was SGML, the progenitor of XML. SGML was designed to be a pure semantic model and it also had a machine-independent specification and encoding, but it had many other disadvantages. Since it was developed specifically to support publishing, it contained a number of components for including character sets (as included substitution ENTITIES) and directions to phototypesetters (as a Processing Instruction), which were unnecessarily complicated for cleanly defining pure data exchange.

There were different classes of data (ENTITY, ELEMENT, ATTRIBUTE) with different syntaxes and properties. These make sense in the context of printing (ENTITY to substitute a character, ELEMENT to define the visible content of the document, ATTRIBUTE to assign internal properties not visible to a reader), but not for defining data structures. In addition to these complexities for defining text, essential types for defining data (such as integer,

float, binary data) were completely missing. The definition syntax (DTD) made it difficult to support modular definitions, since ELEMENT names must be unique across a DTD, which tended to produce conflicts when including commonly used ELEMENT names such as name or year in different data structures.

This notion of defining long-term scientific data in ASN.1 instead of as relational tables or a custom text record was a radical idea in biomedicine in 1990. NCBI was a founding member of OMG (Object Management Group) but dropped out when CORBA was announced as the standard, since the result was a heavyweight solution to a lightweight problem. Ironically, some members of the pharmaceutical industry and some European bioinformatics groups discovered OMG and CORBA about the same time NCBI gave up on it. These groups became very active in OMG-based standards efforts for CORBA, largely ignoring the work already done in ASN.1 at NCBI. These same groups have now also discovered XML. As OMG moved away from CORBA to supporting XML standards, these groups have now moved their standards efforts into this technology.

After many rounds of revision, SGML gave rise to HTML, then to XML, and finally to XML Schema. With the advent of XML Schema, most of the structural deficiencies of SGML languages have been corrected to the point that it is a reasonable choice for data exchange. It still lacks compact binary encoding, meaning a sequence data record with a fair amount of internal structure is six times larger in XML than in binary ASN.1. XML is

still encumbered with arcane differences between ENTITY, ATTRIBUTE, and ELEMENT. The huge advantage of XML over ASN.1, however, is the large number of available software tools for it and the growing number of programmers with at least a rudimentary working knowledge.

Given that sea change, NCBI took advantage of the fact that ASN.1 can easily be automatically mapped to XML and back. We automatically generate DTDs and schemas corresponding to our ASN.1 data definitions and

DATABASES OF DISCOVERY

automatically generate XML, which is isomorphic with ASN.1. We continue to use ASN.1 internally for defining data and client/server interfaces, but provide XML and Web services equivalents for those in the user community who are using XML. The many advantages NCBI gained through its use of ASN.1 as the central architecture for its services and databases are now also being realized by the larger community adopting XML. The power of the approach is clearly overriding the deficiencies of the language.

The adoption of ASN.1 within NCBI has provided a combination of formal structure with flexible implementation that has been a valuable and powerful tool for 15 years of rapid growth. Unfortunately it was never widely adopted outside NCBI, probably for three major reasons: (1) ASN.1, while a public standard, never had the wide public code base that arose for HTML on the Web, which led to XML tools; (2) it was presented to the biomedical community before the need for distributed networked services was obvious to most practitioners in the field; (3) it was used to define a large interconnected data

model, ranging from proteins to DNA to bibliographic data, at a time when those domains were considered separate, unconnected activities.

The model also provided explicit connections between sequence fragments into large, genome-scale constructs before any large genomes had been sequenced. These properties allowed NCBI to scale its software and database resources in size and complexity as large-scale genome sequencing developed, without significant changes to our basic data model. Ironically, by the time the properties that made this possible were being recognized by the biomedical community as a whole, other formats for different parts of the problem had evolved piecemeal in an ad hoc way, and these remain the common formats in biotechnology today.

As an aside, NCBI adopted XML instead of ASN.1 as the internal standard for its electronic text activities, which are extensive (including PubMed, PubMed Central, and a collection of online books called the NCBI Bookshelf). These are text documents, and representing them in a language derivative of SGML is natural and appropriate. We use standard XML parsers and validators, and use XSLT to render the content into HTML and other formats in realtime. NCBI has produced a modular DTD for electronic publishing, which is now being adopted as a standard by many electronic library initiatives and commercial publishers. The rise of electronic publishing, XML as a language, and bibliographic data in SGML came much closer together in time than the genome models in

ASN.1. For these reasons, we seem to be having better luck at getting the outside community to adopt the XML standards in use within NCBI.

Once we chose a data definition language, we had to define the data model. We wished to architect the overall information to support the kind of discovery in molecular biology described earlier. To do this, we attempted in our logical design to separate as much as possible the physical observations made by experimental methods (e.g., the protein sequence itself or the

three-dimensional structure from X-ray diffraction studies) from the interpretation of the data at the time it was deposited in the database. By *interpretation* we mean the names and functions attributed to or inferred about the observation at the time.

This separation is essential for two reasons. First, when scientists deposit data in public databases, they almost never update it as understanding develops over the years. So the annotation tends to go stale over time. It would require a very large number of very highly trained individuals to maintain this information on every record in the database over time, since essentially they would be faced with understanding and extracting the entire scientific literature in biomedicine on a realtime basis. With a few exceptions, this is not practical. Second, the interpretations they made may be inaccurate or there may be legitimate scientific differences of opinion or new insights that may completely change the way a domain is viewed. So the interpretations are imprecise, incomplete, volatile, and tend to be out of date. The observed data—the sequence or article—however, remains stable, even if our understanding of it changes. It is important to keep the factual data connected to what interpretations are available, but not to organize the whole information system around it.

The natural place to find the descriptive information about factual data and what it means are scientific articles. Text records already accommodate the differences of opinion, imprecise language, changing interpretations, and even paradigm shifts typical of scientific discourse. That is, what is so hard to represent well or completely in an up-to-date way in a structured database is already represented and maintained up-to-date anyway in the form of the published scientific literature. Just as we felt it fruitless to fight the market forces in computer hardware to fit it to our needs, and instead adapted our strategy to it, so we chose to embrace the largely unstructured text of the scientific literature as our “primary annotation” of the factual databases, instead of attempting to go against the tide and maintain structured, up-to-date annotation in all the factual databases ourselves.

Thus, we defined our information space as a series of nodes (figure 3). Each node represents a specific class of observation, such as DNA sequence, protein sequence, or a published scientific article. Each node can be structured to fit that class of data, and the data flows and tools associated with that node could be unique to it and its quirks, be they technical or sociological. Even though each node need not share a database or schema with another node, explicit connections were defined between them: DNA

codes for protein; or a particular protein sequence published in a particular scientific article.

We also established computed links within nodes. For example, we run BLAST comparisons between all the proteins in the protein nodes and store the links between proteins that are very similar. In this case we are not looking for subtle matches—you would still need to use analytical tools for that—but we are capturing relationships that are likely to be significant, whether they are already known or not. This makes checking questions such as “Are there other proteins like this one?” very quick and straightforward to ask. For the text node, we use statistical text retrieval techniques to look for articles that share a significant number of high-value words in their titles and abstracts, whether or not a human has decided if they are related.

Making previously unknown connections between facts is the essence of discovery, and the system is designed to support this process.

Similarly, linking between nodes is often done computationally as well. Sequence records contain citation information that can be matched to PubMed bibliographic records through an error-tolerant citation-matching tool. The limited number of fields in the sequence record citation can be matched to the corpus of biomedical literature in PubMed, and if a unique match is returned, we can reliably link to the much more complete and accurate citation in PubMed. We have been able to do similar matching to the OCR'ed text from back scanned articles in PubMed Central to link the bibliographies of scanned articles reliably to fully structured citations. Similar processes involving matching sequences, structures, organism names, and more can be applied to link limited information in one resource to more complete and accurate information in another.

With this system we can re-create the logical process a scientist follows in the colon cancer example mentioned previously. We can query the bibliographic node with terms such as *human*, *colon*, and *cancer*. We will find a large number of articles, most of which have nothing to do with sequences. We can link from those articles to the DNA node because a few of the articles are about sequencing the colon cancer gene. From the DNA node we can link to the proteins coded for by the gene. Using the

DATABASES OF DISCOVERY

computed BLAST links we can quickly find other proteins like the human colon cancer sequence. This list includes the yeast and *E. coli* DNA repair enzyme proteins, even though they share no annotation or words in common with the human colon cancer gene.

From the proteins we can link to the one or two articles published describing the sequencing of these genes. Now we have articles that use the terms describing these genes (e.g., *DNA repair enzyme*, *E. coli*, etc.). Using the computed relationships between articles we can find other articles that share those terms, but instead

of describing sequences, they are describing the genetics and physiology of these genes in bacteria and yeast. In a few minutes a human clinical geneticist who started out reading about human colon cancer genes is reading the research literature in a completely different field, in journals the geneticist would not normally look at, learning about and planning experiments on a human disease gene by comparison with a large body of experimental research in yeast and *E. coli*.

By identifying relationships between records that we can compute, we accomplish two goals. The first is scalability. We can take advantage of Moore's law to stay ahead of the explosive growth of our data. Instead of having to add more human staff, we can add faster/cheaper CPUs. If a new algorithm is developed, we can rerun it over the whole dataset. In this case, more data improves our statistics instead of overwhelming our staff.

The second goal is increasing the ability to make discoveries. Since we are computing relationships, we

may make significant connections between data elements that were not known to the authors at the time the data was submitted. Making previously unknown connections between facts is the essence of discovery, and the system is designed to support this process. Each time we add a node we incur a cost in staff, design, software development, and maintenance. Thus, the cost goes up as a function of the number of nodes. The chance for discovery goes up as the number of connections between nodes—and thus the value of the system—goes up at an accelerating rate with the number of nodes, while the cost goes up at a linear rate (figure 4).

At NCBI we understand that purely computational connections made with biological data can rarely be considered a true scien-

tific discovery or a reliable new “fact” unless confirmed by experiment in a broader life science context. NCBI’s role is to help scientists decide what their next experiment should be, making available as comprehensive and well-connected a set of information as we can, be it computed or compiled manually. We are a tool in the discovery process, not the process itself. This helps us bound the problems we attempt to solve and those we do not attempt to solve. In an ongoing, open-ended process like scientific research, it is important to have a framework to decide how much is enough and which problems to tackle at any given point in the development of the field, especially for a large public resource like NCBI.

ENTREZ

In 1990 NCBI started creating an end-user information system based on these principles called Entrez, which was designed to support some of the databases described earlier. The first version had three logical nodes: DNA sequences, protein sequences, and bibliographic data (figure 3). It was first released on CD-ROMs. The data was in ASN.1 with B-tree indexes to offsets within large data files. We created the NCBI Software Toolkit, a set of C libraries that would search, access, and display the data. Using these libraries, we built Entrez. We also made the libraries available in the public domain as source code to encourage both academic and commercial use of the scientific data.

The C toolkit was designed to be application source code identical on PC, Mac, Unix, and VMS. It was based on ANSI C, but with some necessary extensions for portability. This included both correcting unpleasant behavior and compensating for real problems in the implementation of the ANSI standard across all the target computing platforms. For example, ANSI C has the unpleasant behavior that `toupper()` of a noncharacter is “undefined,” so an ANSI C compiler can core dump when you try to `toupper()` an integer in an array of text. This is ANSI standard behavior, but it is unpleasant for the application. Therefore, for cases such as this we created safe macros that would exhibit non-ANSI but robust behavior. One example of a real problem in ANSI C implementation was the lack of standard support for the microcomputer memory models available at the time. PCs had NEAR and FAR memory, and Macs required the use of Handle. There was no uniform way across Mac, PC, and Unix to allocate large amounts of memory. The C toolkit had functions that would do this in a standard way for the application programmer, but using the native operating system underneath.

Originally, we required only libraries and applications that we planned to export outside NCBI to be written with the NCBI Toolkit. In-house we had Unix machines, so we simply wrote in ANSI C. Two problems arose, however. One was that sometimes we would create a function for in-house use that we later decided to export, and it would have to be rewritten for that purpose. The other was that as flavors of Unix evolved we found ourselves rewriting the ANSI C applications, but just recompiling the Toolkit applications. With the advent of ANSI C++, we have now created an NCBI C++ Toolkit, and we now require that all applications be written with this Toolkit whether intended for in-house use or export. All our main public applications, which run under massive daily load, are written with the same Toolkit framework as specialized utilities that users take from our FTP site.

Entrez evolved from a CD-ROM-based system with three data nodes, through a Toolkit-based Internet client/server with five nodes, to the current Web-based version with more than 20 nodes. Each node represents a back-end database for a different data object. Each data object may be stored in databases of very different design, depending on the type, use, and volume of the data. Despite this, the presentation to the user is of a single unified data space with consistent navigation.

The bibliographic databases are stored as blobs of XML in relational databases. The schema of these databases represents the tracking and identification of the blobs, but not the structure of the article itself. The article structure is defined by the DTD, not the database. Similarly, many of the sequence databases are stored as blobs of ASN.1 in relational databases. Again, the schema of the database largely reflects the tracking of the blobs and some limited number of attributes that are frequently used by the database. It does not reflect the structure of the ASN.1 data model. In both cases, a blob model was chosen for similar reasons.

These databases tend to be updated a whole record at a time, by complete replacement, not by modifying an attribute at a time. It is uncommon for a large number of these records to need the same update at the same time. Typically each record is a logical whole. It is created as a unit, deposited as a unit, retrieved as a unit, and used as a unit. The whole schema for the record is very complicated with many optional elements. Representing such a record as a normalized database would produce a large number of sparsely populated tables with complicated relations. Most common uses would require joining all the tables to reproduce the record anyway.

In contrast, other large sequence databases, such as

DATABASES OF DISCOVERY

those for ESTs (Expressed Sequence Tags), are normalized relational databases. ESTs are large libraries of short snippets of DNA. There may be tens of thousands of simple EST records from a single biological library. Many of the properties of the EST are defined by the library, not by the individual record. So in this case, there are significant advantages in terms of database size, and in terms of commonly applied tasks, to fully normalizing the data.

Despite the diversity of underlying database designs and implementations for each node, there is a common interface for indexing and retrieval. The Entrez indexes are large ISAMs (indexed sequential access methods), optimized for retrieval speed, not for realtime updating. They are updated once a night in batch. There are several different interfaces to the indexing engine, such as a function library or an XML document, in which each database can present the terms to be indexed for each record, the field to index them under, and the UID (unique ID) for the record that contains those terms. In addition, each database must present a small, structured Document Summary (DocSum) for each record. From these simple, standard inputs, Entrez builds the indexes for Boolean queries and stores the DocSum records locally to present to the user as simple lists of results.

All the high-load, high-speed queries and list displays are carried out in a uniform way for all back-end databases by a single optimized application. Once users have selected a record of interest for a particular database, however, they are referred to a specialized viewer for that particular database. Each group supporting a particular database can offer specialized views and services appropriate to a particular database, yet all can use a common user interface and search engine.

SUMMARY

NCBI has grown from 12 people supporting a few users of sequence data in 1988, to more than 200 people supporting millions of users of data from sequences to genes to books to structures to genomes to articles. Through this process we have maintained consistency through formal data definitions (be they ASN.1 or XML) that couple diverse data types on platforms and implementa-

tions tailored to the specific needs of the resource, yet coded under a common (C or C++) Toolkit framework. By careful selection of the data objects to be represented and careful evaluation of their properties (technical and sociological), it has been possible to architect a workable and relatively stable IT solution for the rapidly growing and changing field of biomedicine and molecular biology.

We have already moved our data onto new hardware platforms (for example, from a few Solaris machines to farms of Linux/Intel machines) and into new software frameworks (for example, from simple servers to load-balanced, distributed servers with queuing systems). We have engaged our community of all levels: from scientists using our services directly on our site, to other sites using our Web services to embed our services in their pages or scripts, to groups that compile our code into stand-alone local applications or embed our functions into their products.

For some additional examples, illustrated by tutorials based on current topics in molecular biology, the reader may wish to explore the NCBI Coffee Break section, found at <http://www.ncbi.nlm.nih.gov/books/>. Q

RELATED LINKS

1. NCBI: <http://www.ncbi.nlm.nih.gov/>
2. PubMed: <http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=PubMed>
3. GenBank: <http://www.ncbi.nlm.nih.gov/Genbank/index.html>
4. PubMed Central: <http://www.pubmedcentral.gov/about/intro.html>
5. RefSeq: <http://www.ncbi.nlm.nih.gov/RefSeq/>
6. Entrez: <http://www.ncbi.nlm.nih.gov/entrez/>
7. BLAST: <http://www.ncbi.nlm.nih.gov/BLAST/>
8. CDD (Conserved Domain Database): <http://www.ncbi.nlm.nih.gov/Structure/cdd/cdd.shtml>

LOVE IT, HATE IT? LET US KNOW

feedback@acmqueue.com or www.acmqueue.com/forums

JAMES OSTELL was trained in traditional developmental biology and microscopy. He earned a Ph.D. in molecular biology from Harvard University. He developed a commercial package of software for molecular biologists called MacVector, first released in 1982 and still in use today. In 1988, he became chief of the information engineering branch at the newly formed National Center for Biotechnology Information at the National Institutes of Health, where he was later appointed to the Senior Biomedical Research Service. 2005 ACM 1542-7730/05/0400 This paper is authored by an employee of the U.S. Government and is in the public domain.