Datenmodellierung, Datenformate und Datenschnittstellen in der Bioinformatik

Datenreservoir Biologie

Dr. Matthias Lange

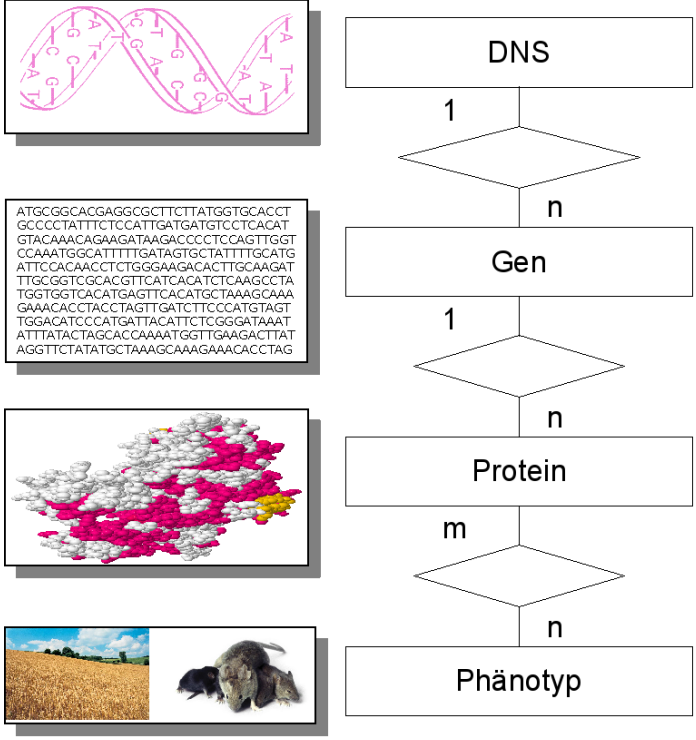
Abstrakt

Die Biologie gehört zu den Naturwissenschaften mit einer starken Abhängigkeit von Informatikmethoden zur Modellierung, Verwaltung und Auswertung von Daten. Heterogen modellierte und implementierte, weltweit verteilte Daten und Anwendungen sind eine Herausforderung an die Bioinformatik. Grundlegende Aspekte sind dabei die verschiedenen Modellierungsmethoden existierender Datenbank und Softwaresysteme, welche die notwendige Bereitstellung einer einheitlichen Bioinformatikinfrastruktur und die Entwicklung neuer Anwendungen erschweren. In diesem Beitrag wird eine Zusammenfassung von Datenmodellen, Datenformaten und Schnittstellen, die in der Bioinformatik Anwendung finden, vorgestellt.

Die Biologie ist eine Wissenschaft deren Forschung und die sich daraus ergebenen Möglichkeiten unsere heutige technologische Entwicklung stark beeinflusst. So schufen wir uns durch die Beobachtung, Nachahmung und Adaption der Natur technische Möglichkeiten, ohne die das heutige Leben kaum vorstellbar wären. So studierte der deutsche Otto Lilenthal vor 140 Jahren den Vogelflug und adaptierte seine Ereignisse und baute daraus erste künstliche Flugapparate, die Vorläufer unserer heutigen Flugzeuge. Auch beruht die Nutzung der Elektrizität, ohne die das moderne Leben nicht vorstellbar wäre, auf Erforschung biologischer Prozesse. In diesem Fall durch die Erforschung der statischen Elektrizität bei Muskelbewegungen von Fröschen vom italienischen Arzt Luigi Galvani vor 225 Jahren. Ebenso wegweisend war die Entdeckung der DNS (Desoxyribonukleinsäure) vom Engländer James Watson und dem US-Amerikaner Francis Crick. Vor 50 Jahren identifizierten sie hiermit das zentrale Molekül zur Kodierung der Eigenschaften und des Bauplanes von Lebewesen.

Biologischer Hintergrund

Ausgehend von den Forschungsresultaten der letzten Jahre, wurden immer mehr Wissen zu den Funktionsprinzipien molekularbiologischer Vorgänge in den Zellen verfügbar, welche die Basis für Gewebe, Organe und zuletzt den gesamten Organismus bilden. Stark vereinfacht trägt die DNS die Gene. Dabei wird die Gesamtheit der DNS eines Lebenwesens als das Genom bezeichnet. Die Gene kodieren in durch eine Abfolge von vier organische Molekülen (Basen) den Bauplan für Proteine, welche die Basisfunktionen der Zellen, wie z. B. die Bildung von Gewebestrukturen oder die energetische Umsetzung von Zucker. Es im Allgemeinen ausreichend, die DNS als Abfolge von vier Buchstaben zu speichern. Die Proteine sind wiederum sehr große Moleküle, die aus einer Abfolge von kleineren, zu einer Kette zusammengefügten Molekülen, den Aminosäuren besteht. So werden Proteine u. a. als Abfolge von 22 Buchstaben in Datenbanken gespeichert. Diese molekularen Strukturen und Prozesse in Organismus sind letztlich bestimmend für dessen Eigenschaften und sein Erscheinungsbild, dem so genannten Phänotyp. In der Abb. 1 wurden diese stark vereinfachten Zusammenhänge Anhand eines ER-Diagramms zusammengefasst. Dieser kurze Ausflug in die Molekularbiologie, soll hier deutlich machen, welche Daten in der Bioinformatik verarbeitet werden müssen.



Datendomänen in der Bioinformatik: Auf der DNS sind eine Vielzahl von Genen, die den Bauplan der Proteine kodieren. Die Proteine bilden das funktionale Gerüst eines Organismus und bestimmen mittels verschiedenster molekularbiologischer Prozesse das Erscheinungsbild, den Phänotyp des Lebewesens. (Abb. 1)

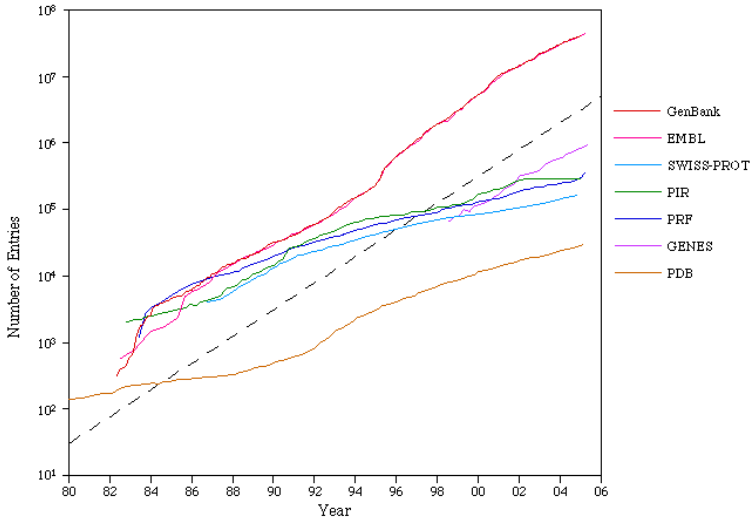
Datenaufkommen

Nicht erst seit den Pressemitteilungen zur „Entschlüsselung“ des menschliches Genoms, mit einer Größe von rund 3.000.000.000 Basen (gespeichert als je vier verschiedenen Buchstaben) und schätzungsweise 26.000 Genen, ist bekannt, dass große und komplexe Daten von der Informatik verarbeitet werden müssen. Um eine Vorstellung vom zu verarbeitenden Datenvolumen zu geben, schauen wir uns allein die zur Zeit „digitalisierten“ bzw. sequenzierten Genome an. Im Internet [1] ist nachzulesen, dass 409 Genome vollständig und 1.662 teilweise sequenziert wurden. Dabei umfasst das kleinste Genom eines Bakteriums 500.000 Basen mit 536 Genen und das größte in der Sequenzierung befindliche Genom von Hafer mit 16.000.000.000 Basen mit einer unbekannten Anzahl von Genen. Bei Proteinen ist das mögliche Datenaufkommen weitaus gigantischer. Geht man davon aus, dass die kleinsten Proteine aus ca. 100 Bausteinen (Moleküle) bestehen, wobei theoretisch 22 verschiedene Moleküle möglich sind, so ergibt sich allein für diese „kleinen“ Proteine eine Anzahl von 22^100 also rund 10^134 möglichen Kombinationen. Zum Vergleich, das zur Zeit größte bekannte Protein umfasst 30.000 Moleküle. Diese ist natürlich nur die theoretische Anzahl von verschiedenen Proteinen. Tatsächlich sind heute ca. 35.000 in Datenbanken erfasst [2]. Um nun tatsächlich die „Funktionweise“ von Lebewesen zu verstehen, sind die molekularen Prozesse, also die Interaktion der Proteine, zu erforschen. Also z. B., der Zusammenhang von Gen- und Proteinaktivität auf Krankheiten, Nutzpflanzenertrag etc. Diese und weitere Daten, werden in weltweit 858 verschiedenen Datenbanken gespeichert (Tab. 1).

Anzahl Datenbanken	Datendomäne
123	DNS-Sequenzen
246	Gene und deren Umsetzung in Proteine
150	Proteine und deren Funktion
2	Literatur
337	Andere

Anzahl biologischer Datenbanken nach Datendomänen zusammengefasst [3] (Tab. 1)

Die gesamte in diesen Datenbanken gespeicherte Datenmenge ist schwer einzuschätzen, stellvertretend für alle Datenbanken zeigt Abb. 2 das exponentiellen Datenwachstum der populärsten Datenbanken.



Exponentielles Datenwachstum in molekularbiologischen Datenbanken [4] (Abb. 2)

Datenmodellierung

Die Modellierung dieser Daten erfolgt in der Bioinformatik zum einen projekt- bzw. anwendungsspezifisch, was in der grossen Zahl verfügbarer, sich inhaltlich überschneidender Datenbanken und Anwendungen resultiert. Zum anderen existieren Projekte, die hinsichtlich der Daten- und Anwendungsmodellierung eine integrative Methodik propagieren.

Beim Entwurf eines Modells ist es eine sehr wichtige Aufgabe, eine geeignet abstrakte Beschreibung der abzubildenden Wirklichkeit (Diskursbereich, Universum) derart zu finden, dass nur die notwendigen Fakten zur Beantwortung bekannter oder unbekannter Fragestellungen an das Modell abgebildet werden. Dieser Anspruch ist in der Biologie im Allgemeinen nicht zu erreichen, da zum Zeitpunkt der Modellerstellung keine Aussagen über alle möglichen Fakten und Fragestellungen an das Modell bekannt sind. Dies gilt insbesondere aufgrund der sich rasant entwickelnden experimentellen Methoden und Kapazitäten, die es ermöglichen, von vormals eher groben Modellen zu verfeinerten, die natürlichen Vorgänge korrekter abbildenden Modellen zu gelangen. Bei der Modellierung in der Bioinformatik ist also ein geeigneter Mittelweg zwischen einem zu einfachen Modell, das nicht alle Fragestellungen ermöglicht und einem unnötig komplexen Modell, das unflexibel und schwer verständlich ist, zu finden.

Wie eingangs angedeutet, hat die Bioinformatik einen stark heterogenen Charakter. Dieser Umstand spiegelt sich auch in den genutzten Methoden zur Daten- und Anwendungsmodellierung wider. Die Entwicklung von Anwendungen und Datenbanken sind im Allgemeinen in sich abgeschlossene, autonome Projekte. So kommen je nach Projektanforderungen und individuellen Präferenzen verschiedene Methoden zum Einsatz. Die Methoden reichen von der objektorientierten, hierarchischen bis hin zur flachen, relationalen Modellierung. Darüberhinaus existieren hybride Ansätze, die verschiedenen Modelle kombinieren bzw. abstrahieren.

Ähnlich heterogen wird bei der Anwendungsmodellierung verfahren. Die Implementierung modellierter Datenschemata in Datenbanken oder auch Programmstrukturen erfolgt in vielen Fällen manuell. Ansätze wie „Model Driven Architecture“ (MDA) [5] werden in der Bioinformatik bis auf einige Ausnahmen, wie das “caCORE“-System (http://ncicb.nci.nih.gov/infrastructure/cacoresdk) vom Amerikanischen „National Cancer Insitute“ wenig genutzt. Hierbei werden ausgehend von einem UML-Diagramm das Datenbankschema und diverse APIs generiert. Anwendung fand dazu der „Quava“-Codegenerator der Firma SAIC. Der Ansatz propagiert eine plattformunabhängige

Modellierung von Anwendungen als Basis für eine plattformspezifische Code- und Datenschemagenerierung. Die Voraussetzung ist allerdings eine strikte Einhaltung der Modellnotationen, was in der Bioinformatik nicht vorauszusetzen ist, da hier eine Vielzahl von projektspezifischen „Notationsdialekten“ existieren.

Die Modellierung erfolgt in der bioinformatischen Praxis vielmehr proprietär und zielgerichtet auf den individuellen Anwendungsfall, so dass verschiedene Sichten auf ähnliche Sachverhalte entstehen. Das resultiert z. B. in verschiedenen Datenbanken, die letztlich das gleiche Datenspektrum umfassen. So werden Anwendungen zur Analyse von Proteininteraktionen bevorzugt objektorientiert entworfen und nutzen dementsprechend objektorientierte oder hierarchische Datenmodelle. Dagegen dienen für Data-Mining-Anwendungen in der Bioinformatik, etwa zur Klassifizierung von Texten aus Publikationen flache, relationale Datenstrukturen, die in relationalen Datenbankmanagementsystemen (DBMS) implementiert wurden. Für die Entwicklung außerhalb solcher Anwendungskontexte entworfenen Applikationen müssen somit die zugrundeliegenden konzeptuellen Datenschemata der Datenbanken nachträglich analysiert werden, was erschwert wird, da vielfach die semantischen Datenschemata nicht veröffentlich werden. Vielmehr wird das „know-how“ der Modellierung explizit durch den Einsatz von Middleware-Architekturen zum Datenzugriff verborgen. Daraus ergeben sich die so genannten Quasistandards in Form von Datenaustauschformaten und damit die Notwendigkeit zur nachträglichen, oftmals redundanten Modellierung von Anwendungsfällen und die Transformation der öffentlichen Datenbanken zu lokalen, restrukturierten Kopien. In der Bioinformatik ist somit ein Trend hinsichtlich der Verlagerung der konzeptionellen Modellierung hin zu spezifischen Datenstrukturdefinition festzustellen.

Datenformate

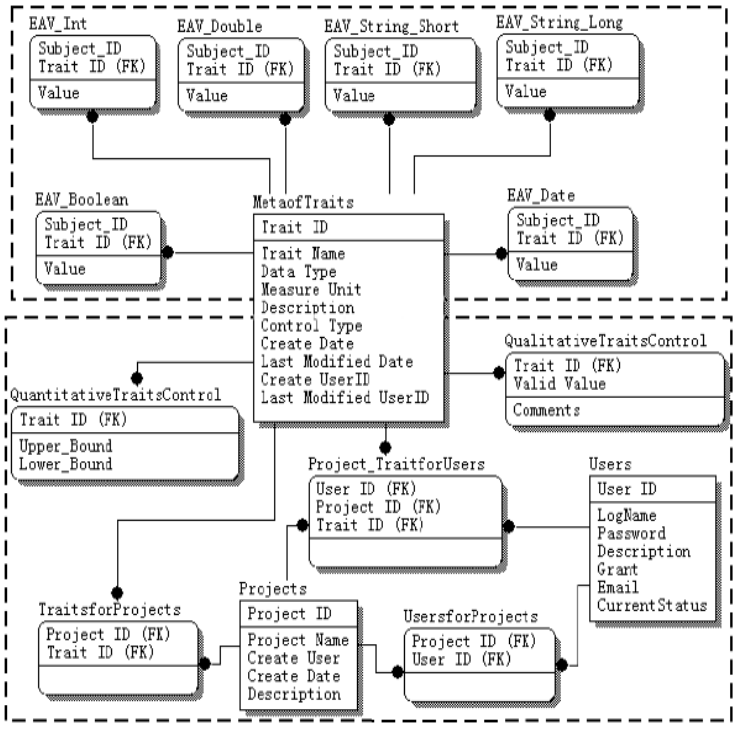
Neben dem Einsatz von Standardlösungen zur Datenmodellierung, wie relationale, objektorientierte oder hierarchische DBMS, sind in der Bioinformatik Speziallösungen im Einsatz. Diese Systeme und Datenaustauschformate haben oftmals bzgl. des zugrundeliegenden Datenmodells einen hybriden Charakter. Die Verwendung von proprietären ASCII-flat-file-Strukturen für den Datenaustausch ist in der Bioinformatik populär und stellt vielfach den einzigen Weg des Datenzugriffes dar.

Das so genannte „attribute-value“-Format gruppiert geschachtelte Blöcke von Attribut-Wert-Paaren (Abb. 3).

```
FEATURES             Location/Qualifiers
     source          1..590
                     /organism="Hordeum vulgare subsp. vulgare"
                     /mol_type="mRNA"
                     /cultivar="Barke"
                     /sub_species="vulgare"
                     /db_xref="GABI:128178"
                     /db_xref="taxon:112509"
                     /clone="HA01A03"
                     /tissue_type="embryosac"
                     /dev_stage="0-7 DAP (days after pollination)"
                     /lab_host="XL10-Gold"
                     /clone_lib="HA"
                     /note="Vector: pBluescript SK+; Site_1: EcoRI (5'-end of
                     cDNA); Site_2: XhoI (3'-end of cDNA); 0-7 DAP(days after
                     pollination) Due to a cloning artefact caused by the kit,
                     in most cases the EcoRI site is NOT present, as well as
                     the EcoRIadapter used for cloning. To excise the insert,
                     restriction sites upstream EcoRI should be used (e.g.
                     BamHI, SalI,PstI). NOTE: Also due to the cloning system
                     used Blue/white selection for recombinats is not 100%
                     reliable. Average insert size is 1 kb"
ORIGIN
        1 actcgatcat tcagttttca ggtaatacag acatatagca aatagtttaa cagacatgcc
       61 tggagattga gtctcacaac gaaggtaaca aaaaattgga ctagaacaga tattatcaag
      121 gtgagcatag gacaggttcc acataacaca acagcacaaa gcttaacggt agcggcgaag
      181 ggagacagtc tggttcctct tccaagtggc tctccttgag ggcctgttct tgtggtggcg
      241 gtgacccttg ccacgcagac cacggaactt ctttccagct gaagtgagac cacgcagctc
      301 acggtgctta tgcacaggct tgcagagcca gttgatcctt gggtcgttac ggacagcact
      361 gtgtgcaaca tcaacaagga tgacctcaaa gtacttgtag gtagaatcct cattcaccca
      421 gtaggagttg agcacacgaa gaccaccaag cttgcgccca gctctttcct cagcaacaga
      481 cctcttgttc ctttggaact tgagctgggt aataccctgg tgcttgggct taccataaac
      541 aatacccttg ggcacgggcc tcttccggcc accacgcctg acacggatcc
//
```

Beispiel von "Attribute-Wert-Paare" zu DNS-Daten aus der amerikanischen GenBank (Abb. 3)

Dieses Konzept ist evolutionär, ohne formale Basis als Ergebnis einer „ad-hoc“-Programmierung entstanden, kann aber wie im Folgenden erläutert, aufgrund der Ähnlichkeit verwendeter Konzepte als eine Abstraktion des objektorientierten Modells angesehen werden. Eine populäre Umsetzung dieses Konzeptes geschieht in relationalen DBMS in Form des „entity-attribute-value“-Ansatzes (EAV) [6]. Als flexible Methode zur Realisierung von sich häufig ändernden konzeptuellen Datenschemata findet sie Verwendung in vielen Datenbankprojekten, wie z. B. zur Verwaltung von stark subjektiven und sich ändernden Phänotypdaten, wie es in Abb. 4 exemplarisch veranschaulicht wird. Es existiert somit kein explizit modelliertes Datenschema im Sinne eines Datenmodells. Der damit verbundene generische, selbstbeschreibende Charakter bedingt auch den Nachteil von nicht unterstützten deklarativen und teilweise ineffizienten Anfragen. Konsistenzbedingungen und andere Funktionalitäten von DBMS können nur eingeschränkt genutzt werden. Auch sind Informationen zum modellierten Sachverhalt, etwa für die Anwendungsprogrammierung nicht ableitbar.



Beispiel einer EAV-Datenmodellierung: Im oberen Bereich des Diagramms werden die Attributwerte („EAV_Int“, „EAV_Double“, „EAV_String_Short“, „EAV_String_Long“, „EAV_Boolean“, „EAV_Date“) gemäß der Datentypen unterschieden. Die Zentralentität „MetaofTraits“ modelliert die Entitäten und Attribute. Konsistenzbedingungen werden durch die Entitäten „QualitativeTraitsControl“ und „QuantitativeTraitsControl“ umgesetzt. Hier werden zulässige Werte (Attribut „Valid_Value“) sowie der Wertebereich (Attribute „Upper_Bound“ und „Lower_Bound“) bestimmt. Die anderen Diagrammelemente im unteren Teil sind Projekt bzw. Nutzerdaten und sind nicht EAV-spezifisch. (Abb. 4)

Neben dieser proprietären Modellierung werden in der Bioinformatik für den Datenaustausch vermehrt Standarddatenformate eingesetzt, die eine hierarchische oder auch relationale Modellierung abbilden. Viel genutzt ist die Extensible Markup Language (XML) oder Abstract Syntax Notation One (ASN.1). So wird am Amerikanischen Institut „National Center for Biotechnology Information“ (NCBI) für den Datenaustausch ASN.1 eingesetzt. ASN.1 ist ein Standard aus der Telekommmunikation, welcher einen Formalismus zur Spezifikation von abstrakten Datentypen definiert. Durch die Unterstützung verschiedener Programmiersprachen, wie etwa JAVA, C++ oder C, sowie das Vorhandensein verschiedener Werkzeuge, hat sich ASN.1 in Teilbereichen der Bioinformatik etabliert. Einige Datenbanken gehen dazu über, zum Datenaustausch XML-Dateien mit Datenschemabeschreibungen mittels XML-Schema anzubieten. Im Folgenden sind einige Beispiele von genutzten XML-basierten Datenformaten angegeben:

• Bioinformatic Sequence Markup Language (http://www.bsml.org)

• Proteomics Standards Initiative/Molecular Interaction (http://psidev.sourceforge.net/mi/xml/doc/user)

• Systems Biology Markup Language (http://www.sbml.org)

• Micro Array Gene Expression Markup Language (http://www.mged.org)

Für die kombinierte objektorientierte Daten- und Anwendungsmodellierung ist das AceDB-Datenbanksystem in der Bioinformatik verbreitet. Das AceDB-Datenbanksystem (http://www.acedb.org) wurde ursprünglich als System zur Datenhaltung im Rahmen eines Sequenzierprojektes entwickelt und bietet neben einer grafischen Nutzerschnittstelle auch verschiedene Werkzeuge zur Analyse von Genomdaten. Aufgrund seines flexiblen Datenmodells

wurde dieses Softwarepaket anschließend als DBMS in weiteren Projekten der Genomforschung eingesetzt und durch verschiedene Programmierschnittstellen, beispielsweise für Perl, PHP, Java und CORBA unterstüzt. Dabei können Änderungen des Datenmodells durch projektspezifische Konfigurationen vorgenommen werden. Das Datenmodell verfügt über Aspekte eines objektorientierten Ansatzes, ermöglicht jedoch beispielsweise keine Vererbung. Vielmehr werden Objekte und ihre Attribute semistrukturiert in einer Baumstruktur verwaltet. Durch Wiederverwendbarkeit und Nutzerfreundlichkeit konnte sich AceDB als Basis für eine Reihe von Systemen in der Bioinformatik etablieren. Allerdings werden grundlegende Forderungen an ein DBMS, z.B. Synchronisation, Konsistenzüberwachung und Datensicherung nicht erfüllt.

Datenschnittstellen

Für die Nutzung von Datenbanken in der Bioinformatik ist es unabdingbar, das konzeptionelle Datenschema, d. h. die Struktur und die Beziehungen der Daten zu kennen. Da notwendige semantische Datenmodelle oftmals nicht verfügbar sind, ist zumindest der Zugriff auf Metadaten, die ein Datenschema beschreiben, notwendig. Tatsächlich ist ein direkter Zugriff auf den DBMS-Katalog mittels JDBC oder ODBC in der Bioinformatik in den seltensten Fällen möglich. Vielmehr geschieht der Datenschemazugriff indirekt über Service- orientierte Architekturen (SOA). Das sind im Umfeld der Bioinformatik z. B. CORBA oder SOAP. Im ersten Fall werden die Interface Definition Language (IDL) und im zweiten Fall die Web Service Description Language (WSDL) zur Schnittstellen und Datenschemadefinition eingesetzt. Das zugrundeliegende Datenmodell ist hierbei objektorientiert. Bei diesen Techniken ist es Programmiersprachen übergreifend möglich, die Datenschemaspezifikationen direkt in Datenstrukturen der Programmiersprachen zu übersetzen. Weiterhin unterstützen kommerzielle Datenbankhersteller durch eingebettete Methoden und spezielle Datentypen den Zugriff und Verarbeitung von biologischen Daten, wie z. B. ORACLE oder DB2 [7, 8].

Eine weitere, verbreitete Schnittstellenart sind proprietäre APIs, die eine Kombination aus einfachen Zugriffspfaden über Datenbankanfragesprachen und modellierten Datenobjektstrukturen darstellen. So existieren z. B. im Falle objektorientierter Programmiersprachen in der Bioinformatik etwa Programmierschnittstellen (APIs) für den Datenzugriff, die Objektstrukturen als Ergebnis von Datenanfragen liefern. Dazu gehören z. B. BioJava (http://www.biojava.org/) (siehe Beispiel in Abb. 5) oder BioPerl (http://www.bioperl.org).

```java
import org.biojava.bio.seq.*;
import org.biojava.bio.seq.io.*;
import java.io.*;
import org.biojava.bio.*;
import java.util.*;

public class ReadGB {
  public static void main(String[] args) {
    BufferedReader br = null;

    try {

      //create a buffered reader to read the sequence file specified by args[0]
      br = new BufferedReader(new FileReader(args[0]));

    }
    catch (FileNotFoundException ex) {
      //can't find the file specified by args[0]
      ex.printStackTrace();
      System.exit(-1);
    }

    //read the GenBank File
    SequenceIterator sequences = SeqIOTools.readGenbank(br);

    //iterate through the sequences
    while(sequences.hasNext()){
      try {

        Sequence seq = sequences.nextSequence();
        //do stuff with the sequence

      }
      catch (BioException ex) {
        //not in GenBank format
        ex.printStackTrace();
      }catch (NoSuchElementException ex) {
        //request for more sequence when there isn't any
        ex.printStackTrace();
      }
    }
  }
}
```

Beispiel der Nutzung der BioJava-API zum Zugriff auf DNS-Sequenzdaten. Hierbei werden mittels der JAVA-Klasse „SequenceIterator“ alle in einem flat file enthaltenen Einträge geparst. (Abb. 5)

Die Umsetzung dieser APIs basiert auf Parser, welche die einzelnen flat file-Formate, z. B. das zuvor genannte „attribute-value“-Format, verarbeiten könne. Daneben existieren auch kommerzielle Systeme, die ebenfalls über Parser einen integrativen Zugriff auf flat-files bieten. Dazu gehört das populäre Datenbankintegrationssystem „Sequence Retrieval System“ (http://www.biowisdom.com/solutions/srs/), das auf diese Weise hunderte Datenbanken über eine eigens entwickelte Anfragesprache und deren Programmierspracheneinbettung mittels SOAP anbietet. Daneben existiert auch eine direkte Einbettung in ORACLE-Datenbanken mittels „SRS ORACLE Gateway“.

In Tab. 2 sind die in der Bioinformatik gebräuchlichen Datenschnittstellen zusammengefasst.

Schnittstellenklasse	API	Beispiel
DBMS	JDBC, ODBC	http://www.geneontology.org
Service-orientiert Architekturen	SOAP, CORBA	http://www.genome.jp/kegg/soap http://corba.ebi.ac.uk/
XML	XML-Parser	http://www.uniprot.org/database/download.shtml
flat file	BioJava, BioPerl, SRS-WSObjects	ftp://ftp.ncbi.nih.gov/genbank

Klassifizierung von Datenschemata für molekularbiologische Datenbanken (Tab. 2)

Fazit

Der überwiegende Teil des molekularbiologischen Wissens wird teils in öffentlichen, teils in kommerziellen Datenbanken mit sehr unterschiedlichen Formaten und Schnittstellen gepflegt. Die Nutzung dieser Daten zur Verarbeitung und Analyse ist ein wesentlicher Faktor zum Verständnis und Erforschung biologischer Zusammenhänge.

Die Standardisierung von Datenformaten und Schnittstellen ist dabei eine wesentliche Aufgabe der Bioinformatik. Durch die zunehmende Etablierung von Bioinformatik-APIs, XML und SOAP-Schnittstellen wurde und wird es möglich, auch moderne Software aus dem nicht-Biologie Umfeld

auf biologische Daten anzuwenden, wie etwa Algorithmen des Data-Mining oder der Bilderkennung. So wird es möglich, das gesamte Potential biologischen Daten zu nutzen.

Literatur

[1] GOLD - Genomes Online Database: http://www.genomesonline.org

[2] PDB: - Protein Data Bank: http://www.pdb.org

[3] Michael Y. Galperin; The Molecular Biology Database Collection: 2006 update; Nucleic Acid Research, Ausgabe 13 Heft 1, 2003, S. D3-D5

[4] Growth of the Sequence and 3D Structure Databases: http://www.genome.ad.jp/dbget/db growth.html

[5] Bernhard Merkle; Designermodelle; MDA: Überblick über die Tool-Landschaft; iX 5/2005, S. 102

[6] Prakash Nadkarni et al.; Organization of Heterogeneous Scientific Data Using the EAV/CR Representation; Journal of the American Medical Informatics Association, Ausgabe 6, 1999, S. 478-493

[7] Oracle Life Sciences Platform: http://www.oracle.com/technology/industries/life_sciences

[8] DB2 Life Sciences Data Connect: http://www.ibm.com/software/data/db2/lifesciencesdataconnect