Problemseminar Bio-Datenbanken

WS 2002/2003

Integration von Biodaten

Bearbeiter: Steffen Junick

Betreuer: Robert Müller

Prof. Dr. Erhard Rahm

Universität Leipzig
Institut für Informatik

Inhaltsverzeichnis

1. Einleitung

Das in den letzten Jahrzehnten sprunghaft angewachsene Wissen in den Biowissenschaften und das Interesse an einer wirtschaftlichen Umsetzung dieser Erkenntnisse haben den Forschungsaufwand in diesem Bereich noch weiter verstärkt. Die damit einhergehende Spezialisierung führte zur Entwicklung neuer Forschungsfelder, die sich gegenseitig mit Wissen beliefern. Hierzu gehören unter anderem die Bereiche der Genetik und Molekularbiologie.

Die Erforschung der Lebewesen erfolgt parallel auf mehreren Analyseebenen, vom einzelnen Molekül über das Zusammenspiel von Molekülen in Zellen und Organismen bis hin zu den Wechselwirkungen verschiedener Organismen innerhalb eines Ökosystems.

Gegenwärtig gibt es eine Vielzahl von Datenbanken mit medizinischen, molekularbiologischen und genetischen Inhalten, in denen

- gattungs-, art-, organ-, oder zellspezifische Informationen
- der Informationsbestand der Erbsubstanz (DNS / RNS) und seine Veränderungen (Mutationen),
- seine funktionell kontrollierte Ablesung (Transkription),
- seine Umsetzung in Biomakromoleküle, insbesondere Proteine,
- seine Regulation (Signalketten)
- sowie seine Auswirkungen auf die Dynamik im Zellstoffwechsel

gespeichert und mit zugehörigen bibliographischen Belegen und Zitaten versehen sind. Einen strukturierten Überblick über einige dieser Datenbanken gibt die folgende Grafik:

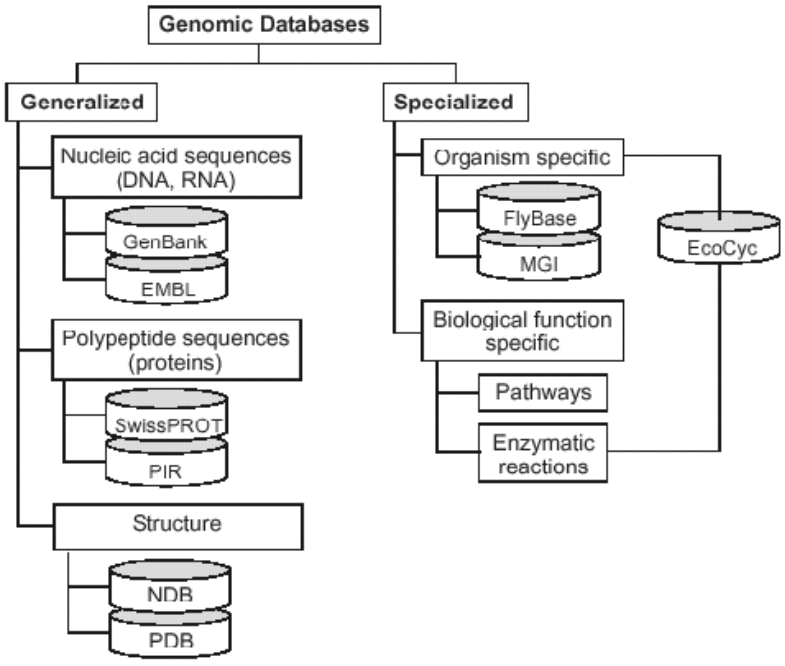

Abbildung 1-1: Genomic Databases [RS02]

Die folgende Arbeit soll die Problematik der Integration von Biodatenbanken diskutieren und verschiedene Lösungsansätze aufzeigen. Abschnitt 2 gibt zunächst einen Überblick über die derzeitige Situation von biologischen Informationssystemen. Die im Allgemeinen bei der

Haltung von Informationen in unterschiedlichen Datenbanken auftretende Heterogenität wird anschließend in Abschnitt 3 klassifiziert. Um heterogene Datenbanken gemeinsam unter Verwendung einer einheitlichen Sichtweise auf Informationen verwenden zu können, sind die in Abschnitt 4 beschriebenen Aufgaben und Anforderungen notwendig. Diese Aufgaben sind bereits in der Praxis, mit denen in Abschnitt 5 präsentierten Alternativen zur Datenintegration gelöst, und werden durch die, in den Abschnitten 6, 7 und 8 vorgestellten Beispiele untermauert.

2. Ausgangssituation von Bio-Informationsquellen

Um die Ausgangssituation von Bio-Informationsquellen zu verallgemeinern, ist zunächst eine Klassifizierung der Daten vorzunehmen, die vorwiegend von Molekularbiologen verwendet werden. Zu diesen gehören unstrukturierte, semistrukturierte und vollstrukturierte Daten. Rohdaten, die Ergebnisse von Experimenten oder Beobachtungen darstellen, liegen oft in unstrukturierten Formaten vor. Annotationen von z. B. Krankheiten könnten in semistrukturierten bzw. vollstrukturierten Formaten vorliegen.

Vollstrukturierte Daten sind oft in relationalen und objektorientierten Datenbanken vorzufinden. Diese Datenbanken sind nach einer expliziten Schemadefinition aufgebaut. Semistrukturierte Daten, z. B. Daten im XML-Format, müssen dagegen ihren Aufbau nicht explizit in einem Schema definiert haben. Sie können ihr Schema, wie auch bei unstrukturierten Daten, nur implizit enthalten. Der Benutzer muss sich in diesem Fall das Schema erst durch die inhaltliche Bedeutung der in den Dateien enthaltenen Informationen erschließen.

Für biologische Inhalte gibt es vorwiegend 3 Arten von Informationsquellen: Datenbanksysteme, Web-Services und Dateien. Bei Datenbanken und Dateien erfolgt der Zugriff durch verschiedene Hilfsprogramme. Hilfsprogramme sind z. B. „blast“[1] (eine Menge von Search-Tools), die auf strukturierten Dateien Sequenzen suchen. Der Zugriffsdienst ist in Web-Services, der unter anderem für die Datenbank SwissProt angeboten wird, schon implizit enthalten und man benötigt lediglich einen Webbrowser. Um Informationen über allen 3 Arten des Zugriffs auszuwerten, fehlen jedoch einheitliche Zugriffsschnittstellen, z. B. eine einheitliche deklarative Anfragesprache wie SQL.

Zurzeit stellen viele Informationsträger schwach integrierte Quellen dar und sind zusammen schwierig nutzbar. Angenommen man kennt Symptome eines Patienten, die auf eine Krankheit hinweisen und möchte wissen: „Welche Krankheit könnte es sein, welche Annotationen gibt es für die Krankheit, und welche Gene und Stoffwechselprozesse sind dafür verantwortlich?“ Um eine Antwort auf diese Frage zu bekommen, die sich auf mehrere Informationsquellen bezieht, muss der Benutzer die Anfrage in kleinere spezifische Anfragen zerlegen und diese wiederum jedem verfügbaren und relevanten Informationssystem stellen. Das bedeutet neben dem erhöhten Arbeitsaufwand auch, dass der Benutzer wissen muss, wie aus speziellen Informationssystemen Daten gewonnen werden können (Anfragesprachen, Tools) und wie diese mit anderen vereint bzw. sinnvoll verknüpft werden, um ein vollständiges Ergebnis zu gewinnen. Es besteht daher die Notwendigkeit bestehende heterogene Informationssysteme bzw. Datenbanken zu integrieren, und vorhandene Informationen gemeinsam nutzbar zu machen. [RS02]

[1] Abk. für Basic Local Alignment Search Tool

3. Heterogene Datenbanken

Unter heterogenen Datenbanken werden Datenbanken verstanden, die inhaltlich verwandte Informationen in uneinheitlicher Weise (heterogen) enthalten. Die einzelnen Datenbankverwaltungssysteme (DBMS) werden in der Regel in unkoordinierter Weise geführt. Sie sind oft unabhängig voneinander entworfen worden. Die Verteilung der Daten ist für den Benutzer nicht transparent, d. h. der Benutzer muss die Lokalisation der einzelnen Daten explizit kennen. Datenkonsistenz wird über mehrere Datenbanken hinweg systemseitig nicht überwacht [ER94].

Die Heterogenität wirkt sich bezüglich der beteiligten DBMS, der Ablaufumgebung und des Datenbank-Inhalts (semantische Heterogenität) aus.

- Dabei versteht man unter Heterogenität der DBMS die Unterschiede von verschieden verwendeten Herstellern, Versionen, *Datenmodellen, Anfragesprachen* sowie interner Realisierungen der Datenbanken.
- Die Heterogenität der Ablaufumgebung wirkt sich in unterschiedlicher Hardware (Prozessortyp, Instruktionsumfang, Zeichensätze etc.), Betriebssysteme, Transaktionsprotokoll-Monitore und Kommunikationsprotokolle aus.
- Die semantische Heterogenität ist eine Folge der Entwurfsautonomie, welche den unabhängigen Entwurf des logischen (und physischen) Aufbaus der einzelnen Datenbanken gestattet. Sie äußert sich in Form von *Schema- und Datenkonflikten* [ER94].

3.1. Heterogenität der Datenmodelle

Unter der Heterogenität von Datenmodellen werden unterschiedlich verwendeten Arten von Datenmodellen verstanden. Hauptarten sind: Hierarchisches Datenmodell, Netzwerkmodell, Relationales Datenmodell und Objektorientiertes Datenmodell.

Das Problem unterschiedlicher Datenmodelle besteht darin, dass sie unterschiedliche Metasprachen und damit verschiedene Ausdrucksweisen der Modellierung von Daten verwenden. Während z. B. das Hierarchische Datenmodell ausschließlich 1:n Beziehungen erlaubt, werden im relationalen Datenmodell zusätzlich n:1, n:m Beziehungen unterstützt. D. h. es ist sehr wahrscheinlich, dass bei der Modellierung von inhaltlich gleichen Sachverhalten unter Verwendung von unterschiedlichen Datenmodellen auch unterschiedliche Schemata und enthaltene Beziehungen auftreten.

Des Weiteren ist es oft der Fall, dass implizit unter Verwendung von verschiedenen Datenmodellen auch verschiedene DBMS mit unterschiedlichen Anfragesprachen verwendet werden. Zum Beispiel wäre es nahe liegend ausgehend von einem relationalen Datenmodell ein DBMS zu verwenden, das SQL unterstützt. Während man wahrscheinlich bei einem objektorientierten Datenmodell ein DBMS wählen wird, welches OQL verwendet.

3.2. Semantische Heterogenität

Die Semantische Heterogenität äußert sich in Form von *Schema- und Datenkonflikten*.

Schemakonflikte

Bei Schemakonflikten sind unterschiedliche Repräsentationen von Datenbankobjekten gewählt worden. Sie werden in *Namenskonflikte, strukturelle Konflikte und Konflikte bei Integritätsbedingungen* unterteilt.

- *Namenskonflikte* untergliedern sich in Homonyme bzw. Synonyme. Unter Homonyme werden gleiche Namen mit unterschiedlicher Bedeutung verstanden (z. B.

„Krebsarten“ 1.die Tierart Krebs 2.die Krankheitsarten). Synonyme dagegen sind unterschiedliche Namen, die aber dasselbe bedeuten (z. B. „Leberkrebs“ und „Leberkarzinom“).

- *Strukturelle Konflikte* können semantisch äquivalente Informationen (Objekte, Beziehungen) sein, die entweder als Relationen oder als Attribute repräsentiert werden. Ein Beispiel hierfür wäre „die Expression eines Genes“ als Attribut oder Relation zu repräsentieren. Bei der Repräsentation als Relation würde die Expression als eigenständige Entität und einer Fremdschlüsselbeziehung auf die Gen-Entität realisiert werden. Als Attribut dagegen würde die Expression implizit in der Entität des Genes enthalten sein.
- *Konflikte bei Integritätsbedingungen* äußern sich auf Relationen- und Attributebene. Konflikte auf Relationenebene gibt es bei der unterschiedlichen Definition der Primärschlüssel und weiteren Schlüsselkandidaten, der Festlegung von Fremdschlüsseln sowie unterschiedlichen Aktionen zur Wartung der referentiellen Integrität und anwendungsspezifischen Integritätsbedingungen. Auf Attributebene können Unterschiede bei Datentypen, Wertebereichen, Default-Werten und Zulässigkeit von Nullwerten auftreten. [ER94]

Datenkonflikte

Datenkonflikte betreffen Unterschiede hinsichtlich der Datenbank-Inhalte auf unterster Ebene, die auf Schemaebene nicht erkennbar sind. Hierbei werden Daten entweder unterschiedlich repräsentiert, oder es entstehen Konflikte durch falsche bzw. unvollständige Inhalte. Ein Beispiel wäre der Wert einer Basenkette „AGC GCA“ <=> „agc gca“.

Im Rahmen der Schemaintegration wird versucht, die semantische Heterogenität zu beheben und eine möglichst widerspruchsfreie Datenbankstruktur zu ermöglichen.

4. Aufgaben und Anforderungen der Integration

Aus der Heterogenität von Datenbanken folgen 2 wesentliche Aufgaben der Integration: die Datenmodelltransformation und die semantische Integration.

Bei der Transformation von Datenmodellen ist darauf zu achten, dass das Zieldatenmodell genug Ausdrucksstärke besitzt, um alle relevanten zu integrierende Informationen auszudrücken [DO95].

Die semantische Integration unterteilt sich in die Schemaintegration und die Integration auf Datenebene.

Die Integration auf Datenebene löst o. g. Probleme der Datenkonflikte. Bei der Schemaintegration handelt es sich um die Auflösung von Namenskonflikten, strukturelle Konflikten sowie Konflikten bei Integritätsbedingungen. Anforderungen sind hier Vollständigkeit, Minimalität und Korrektheit. Unter Vollständigkeit versteht man, dass „kein“ Verlust, der in lokalen Schemata enthaltenen Information auftreten sollte. Minimalität fordert „keine“ Redundanz. Korrektheit bedeutet letztlich „Äquivalenz“, der im integrierten Schema enthaltenen Informationen mit denen in den lokalen Schemata und zusätzlich Konsistenz, der während der Integration ergänzten Informationen (z. B. Beziehungen zwischen Konzepten im integrierten Schema) [ER02].

So sinnvoll diese Anforderungen erscheinen, so sehr müssen sie in Abhängigkeit von der jeweiligen Situation diskutiert werden. So kann Informationsverlust und Redundanz beispielsweise in Data-Warehouses erwünscht sein um Performancevorteile zu erreichen.

Der Prozess der Schemaintegration muss meist vollständig „von Hand“ gemacht und kann nur teilweise durch Schema-Matching automatisiert werden. Schema-Matching wendet mehrere Matching-Algorithmen auf die zu integrierenden Schemata an und bewertet diese [ER02]. Der Nutzer kann letztlich ein Match davon auswählen. Matching-Algorithmen werden in 3 Klassen geteilt: einfache Matcher (z. B. Soundex, Synonym, DataType), hybride Matcher (Kombination von Matchern) und reuse-orientierte Matcher (Verwendung von Informationen von vorhergehenden Matchings). Bei Synonym-Matchern werden oft Informationen mit (biologischen) Thesauri verglichen. Ein Beispiel eines Systems das verschiedene Schema-Matching Ansätze kombiniert ist *COMA* [HR02].

Die Integration auf Datenebene ist eine Transformation heterogener Daten in eine einheitliche, durch das integrierte Schema vorgeschriebene Repräsentation. Dabei sind auch Datenqualitätsprobleme, wie unvollständige Daten und fehlerhafte Werte (Eingabefehler) zu entdecken.

Neben diesen Faktoren sollte die Datenbank eine einheitliche und mächtige Zugriffsschnittstelle bereitstellen. Für Anfragen über ein breites Spektrum von Informationen sollte sie den Zugriff auf mehrere Datenbanken innerhalb einer Transaktion und eine hohe Verteilungstransparenz unterstützen.

5. Grundlegende Alternativen zur Datenintegration

Die im folgendem vorgestellten Alternativen der Datenintegration unterstützen Aspekte der Granularität und physischen Instanz von Informationen in unterschiedlicher Art und Weise. Geringe Granularität zeichnet sich durch einen hohen Detaillierungsgrad der Daten aus. Er erlaubt es Informationen aus den bereits vorliegenden am meisten detaillierten Schichten ausfindig zu machen. Dagegen erlaubt eine hohe Granularität eine bessere Zusammenfassung von Informationen und daraus resultierende Performancevorteile. Unter einer physischen Instanz wird eine integrierte Kopie der Information aus den zu integrierenden Quellen verstanden.

Es gibt es drei grundlegende Alternativen die diese Fragestellungen berücksichtigen: die Verwendung von Standards, die virtuelle Integration und die physische (Vor-)Integration, die vor allem in Data-Warehousing verwendet wird. [ER01]

5.1. Verwendung von Standards

Die gemeinsame Nutzung heterogener Informationssysteme kann über den Einsatz von Standards leichter gemacht werden. Die Standardisierung versucht dabei die in Abschnitt 3 vorgestellte Heterogenität zu vermeiden.

Standards, die versuchen die Ebene des DBVS bzw. der Ablaufumgebung für darauf zugreifende Anwendungen einheitlich zugänglich zu machen, sind bisher als sog. Middleware realisiert. Sie liegt als Schicht zwischen Anwendungen und Plattformen, und kapselt damit Datenquellen, indem sie gemeinsame Protokolle bzw. Schnittstellen zur Verfügung stellt (siehe Abbildung 5-1).

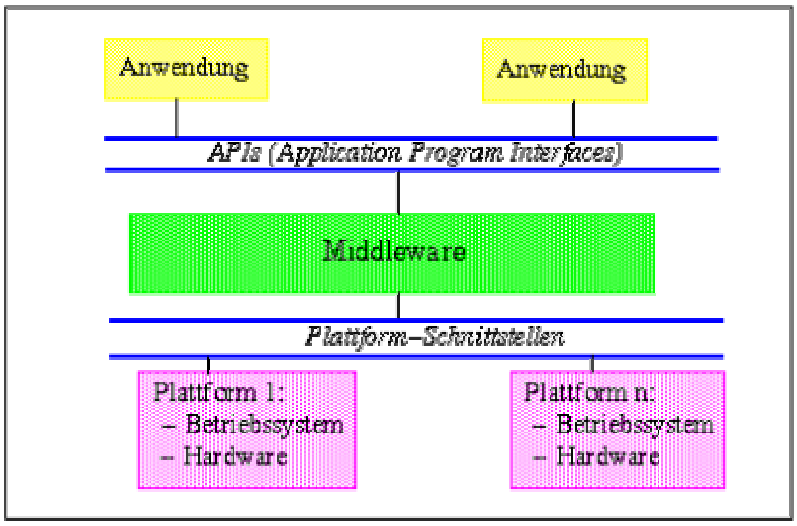

Abbildung 5-1: Stellung von Middleware [Be93]

Den Anwendungen werden meist eine Reihe von Funktionen bzw. APIs (Application-Program-Interfaces) angeboten, die eine einheitliche Sichtweise auf Informationen gestatten und darunter liegende Plattformen vollständig verbergen. Damit vereinfachen sie die Probleme der Verteilung und des Zugangs von Daten [ER94].

Ein weiterer für die Integration wichtigerer Aspekt ist die Standardisierung der semantischen Ebene. Er versucht diese zu homogenisieren, indem standardisierte Datenmodelle, -formate, und -inhalte definiert werden. Es ist damit eine Voraussetzung für ein im Vorfeld festgelegtes globales und einheitliches Schema. Ein Beispiel aus der Praxis sind der MIAME[2]-Standard und der Flatfile-Standard der BioCyc-Knowledge-Library, welcher in Abschnitt 6 noch genauer beschrieben wird.

5.2. Virtuelle Integration

Abbildung 5-2 zeigt die virtuelle Integration. Quelldatenbanken werden durch Wrapper gekapselt. Die Bedeutung von Wrappern liegt in der Bereitstellung einer Schnittstelle, die durch den Mediator genutzt werden kann, um Informationen aus der gekapselten Datenbank zu erhalten. Ein zusätzlicher Idealfall wäre, dass sie die Illusion einer gemeinsamen Anfragesprache schaffen. Wrapper exportieren jeweils ein lokales Schema für jede zu integrierende Datenbank. Die lokalen Schemata werden von einem Mediator zu einem globalen Schema vereinigt, und den Clients zur Verfügung gestellt. Er transformiert globale Anfragen in mehrere, die dann quellspezifisch zugeordnet werden. Die Daten sind in diesem Ansatz nur virtuell integriert, und befinden sich in ihrer Ursprungsform in den operativen Quellen. Anfragen wandelt der Mediator transparent und „on-the-fly“ für Clients um, und kümmert sich um die Zusammenführung von Ergebnissen aus spezifischen Quellen. Die praktische Umsetzung dieser Architektur wird in Abschnitt 7 am Beispiel TAMBIS gezeigt.

[2] Abk. für Minimum information about a microarray experiment

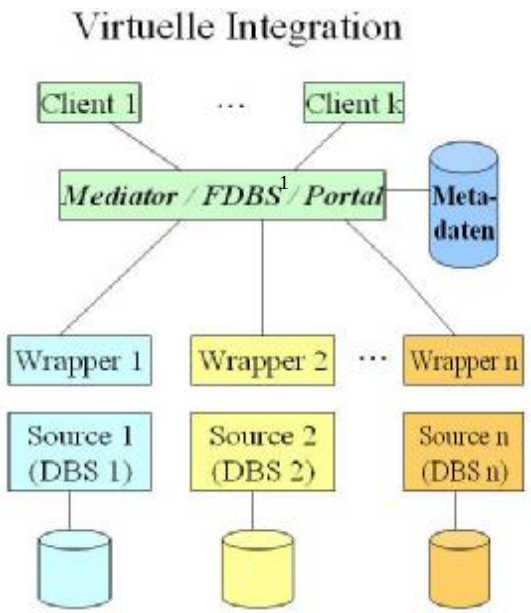

Abbildung 5-2: Virtuelle Integration

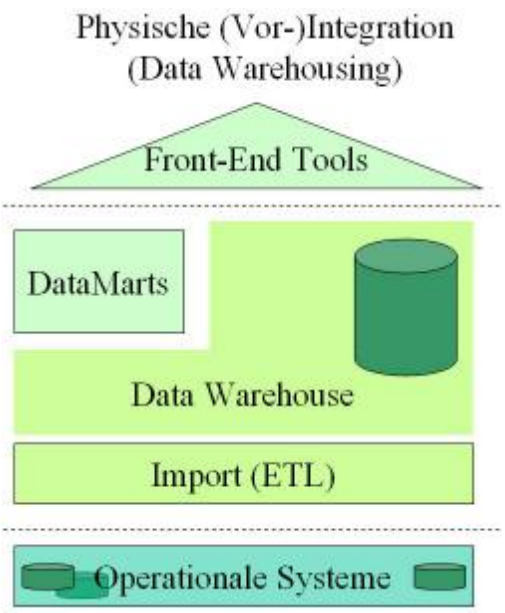

Abbildung 5-3: Physische Integration

1 Föderierte Datenbanksysteme

5.3. Physische (Vor-)Integration

Bei einer physischen (Vor-)Integration, wie sie in Abbildung 5-3 zu sehen ist, werden die Daten aus operativen Quellen durch den so genannten ETL-Prozess (Extraction, Transformation, Loading) in ein Data-Warehouse eingebracht. Daten werden von operativen Quellen extrahiert, in eine einheitliche Form transformiert und letztlich in das Warehouse eingeladen. Die Informationen von Quellen liegen also als physische Kopie vor. Sie sind von da an themen-orientiert, dauerhaft, (evtl. unter Informationsverlust aggregiert) und zeit-bezogen im Data-Warehouse integriert.

6. Verwendung von Standards am Beispiel BioCyc

Ein Beispiel für den Einsatz von Standards bei der Nutzung von heterogenen Informationen ist die BioCyc-Knowledge-Library der Bioinformatics-Research-Group. Sie ist eine Sammlung von Pathway/Genom-Datenbanken.

Pathway/Genom-Datenbanken (PGDB) enthalten Informationen über das Genom[3] eines Organismus (seine Chromosomen, Gene, Genomsequenzen), das Produkt eines jeden Genes, die biochemischen Reaktionen und die Abläufe von biochemischen Reaktionen in ganzen Pathways.

Die meisten der Datenbanken in BioCyc beschreiben (partiell) Genome- und Metabolic-Pathways von einem einzigen Organismus. Ein paar Beispiele sind die Datenbanken: HinCyc (Haemophilus influenza), YeastCyc (Saccharomyces cerevisiae) und VchoCyc (Vibrio cholerae). Eine Ausnahme bilden die MetaCyc-Datenbank und die EcoCyc-Datenbank.

- Die MetaCyc-Datenbank beinhaltet Referenzen auf Metabolic-Pathways von über 150 Organismen [Bi02]. Ziel ist eine breite Abdeckung von experimentellen Metabolic-Pathways.

[3] vollständige Menge von Genen eines Organismus

- Die EcoCyc-Datenbank enthält vollständige Genomsequenzen des Bakteriums Escherichia coli MG1655

BioCyc-Datenbanken sind elektronische Referenzen für Biologen, die mit verschiedenen Mikroorganismen arbeiten. Derzeit sind für 13 Spezies detaillierte Informationen verfügbar. Wissenschaftler können BioCyc-Datenbanken nutzen, um Informationen über Gene, genetische Karten („genomic-maps"), biochemische Reaktionen sowie eines kompletten Pathways zu verwalten und grafisch darzustellen. BioCyc stellt dafür Anwendungsprogramme zur Verfügung, die das Arbeiten mit dem System erleichtern.

Eine Beispielanwendung der BioCyc-Knowledge-Library ist in Abbildung 6-1 dargestellt. Es zeigt die Visualisierung des dTDP-rhamnose Biosynthese-Pathways des Bakteriums E. coli. Das BioCyc-Tool „BioPACS-BioCyC" generiert zunächst eine XML-Datei, die dann von Genomic-Object-Net grafisch dargestellt wird.

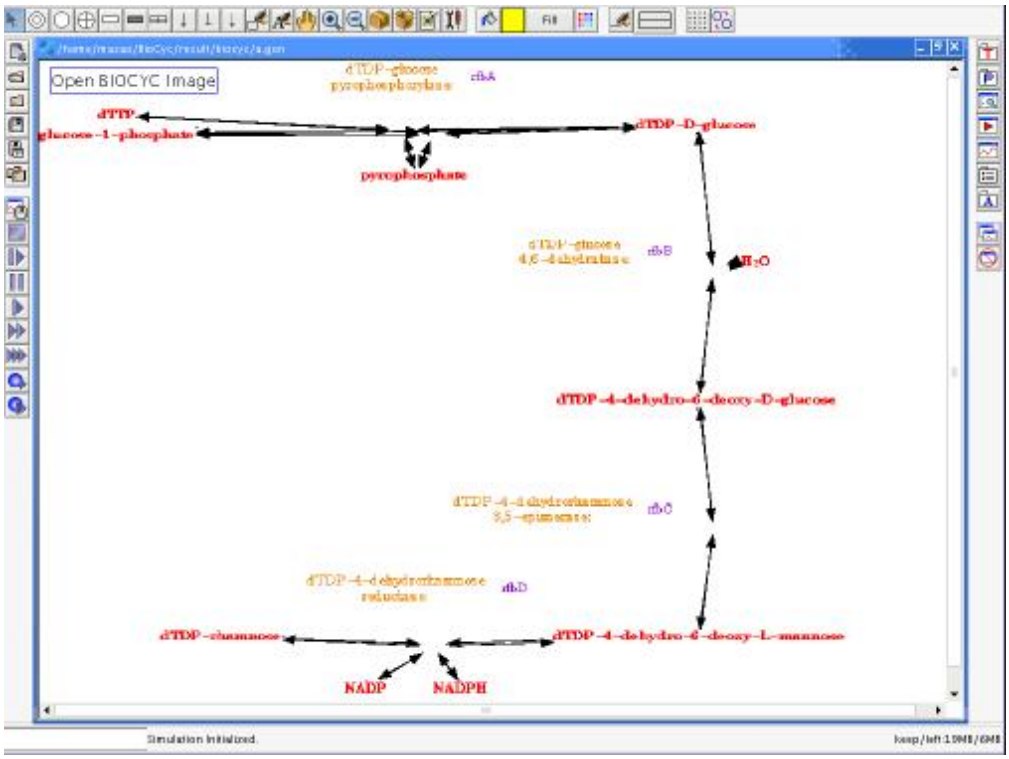

Abbildung 6-1: Visualisierung des dTDP-rhamnose Biosynthese-Pathways

6.1. Standard Flatfile-Formate

Jede Datenbank, die sich innerhalb von BioCyc befindet, wird in eine Menge von Flatfiles exportiert, um die Nutzung dieser Daten von anderen Programmen und DBMS zu erleichtern. Unter anderem sind das die Dateien: enzymes.col, genes.col, proteins.dat, usw.

Diese Dateien haben unterschiedliche, aber festgelegte Formate (Tabular, Attributvalue, FASTA, Ocelot) hinsichtlich ihrer Strukturierung und sind inhaltlich standardisiert.

Um diesen Aspekt zu verdeutlichen wird hier der Aufbau der Datei genes.col vorgestellt. Sie ist vom Dateityp ein Tabular-Flatfile, die den ASCII-Zeichensatz verwendet. Ein Tabular-Flatfile enthält Daten für eine Klasse von Objekten. Es beschreibt eine Tabelle, die durch Tabulatoren und Zeilen abgegrenzt ist. Die erste Zeile enthält den Tabellenkopf. Weitere Zeilen repräsentieren eine Instanz eines Objektes. Die inhaltliche Struktur der genes.col ist folgendermaßen:

UNIQUE-ID | BLATTNER-ID | COMMON-NAME | PRODUCT-NAME | SWISS-PROT-ID ...

...REPLICON UNIQUE-ID | START-BASE END-BASE | SYNONYMS (4) | GENE TYPE (4)

Ein Auszug eines Beispiels für die genes.col-Datei:

```
UNIQUE-ID      BLATTNER-ID    NAME                          PRODUCT-NAME   SWISS-PROT-ID  ...
EG10774 b1207  prsA           phosphoribosylpyrophosphate   synthase       P08330         ...
G6239 b0425    panE           2-dehydropantoate             reductase      P77728         ...
EG10502 b3670  ilvN           acetohydroxybutanoate         synthase       P08143         ...
.
.
.
```

6.2. Weiterverwendung der standardisierten Flatfiles

Alle Daten von BioCyc werden standardmäßig in relationalen Datenbankverwaltungssystemen von Oracle gehalten. Auf diesem aufsetzend arbeitet das Ocelot-Datenbankverwaltungssystem, welches die verschiedenen Datenbanken integriert und den Pathway-Tools von BioCyc (z. B. PathoLogic) eine objektorientierte Sicht auf die Daten gewährt (siehe Abbildung 6-2).

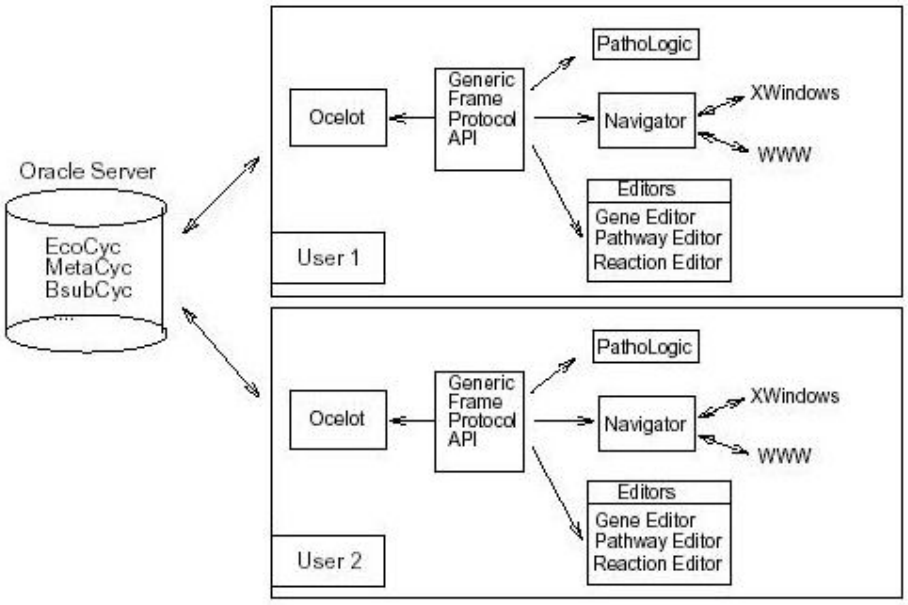

Abbildung 6-2: BioCyc Software-Architekturen auf Grundlage eines relationalen DBMS [KP02]

Ocelot benötigt jedoch nicht unbedingt ein relationales Datenbankverwaltungssystem, sondern ist ebenfalls im Stande die standardisierten Flatfiles zu integrieren. Der Vorteil ist das eine Einbenutzer-Entwicklungskonfiguration verwenden werden kann, die zum einen vom Benutzer festgelegte Dateien integriert und zum anderen den Aufwand einer Installation eines DBMS vermeidet. Die Änderungen an den Flatfiles können zu einem späterem Zeitpunkt mit der relationalen Datenbank abgeglichen werden.

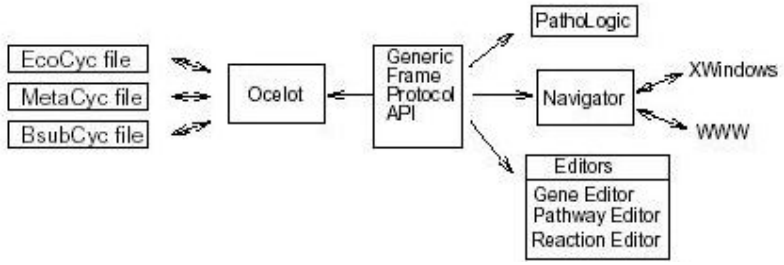

Abbildung 6-3: BioCyc Software-Architektur auf Grundlage standardisierter Flatfiles [KP02]

7. TAMBIS - ein Beispiel einer virtuellen Integration

TAMBIS - „Transparent Access To Multiple Bioinformatics Information Sources“ ist eine ontologiebasierte[4] Metadatenbank, die auf der Mediator-Wrapper-Architektur (siehe Abbildung 5-2) beruht. Es ist ein gemeinsames Projekt zwischen der „*School of Biological Sciences*“ und der „*Information Management Group*“ (IMG). TAMBIS unterstützt die Medizinforschung, indem es eine einheitliche Schnittstelle für biologische Informationsquellen zur Verfügung stellt. Es werden derzeit folgende Informationssysteme integriert: „Swiss-Prot“, „Enzyme“, „Cath“, „blast“ und „ProSite“.

Es enthält jeweils:

- „SwissProt“ Beschreibungen über Funktionen, Strukturen, Varianten und Annotationen von Proteinen
- „Enzyme“ Informationen über Enzyme
- „Cath“ eine hierarchische Klassifikation von Protein-Domain-Strukturen, die Proteine in vier Haupgruppen clustert
- „blast“ eine Menge von Such-Programmen, die auf verschiedenen Datenbanken nach Sequenzen bzw. Mustern von Proteinen oder Genomen suchen
- „ProSite“ Protein-Profile und -Muster

Damit können Fragen über Proteine (Funktionsbeschreibung, Prozesse, Sequenzketten, Gleichheiten zu anderen, Sekundär- und Tertiärstrukturen, deren Motive[5]), Enzyme (deren Substrate, Produkte, Kofaktoren) und verschiedene Funktionen sowie Typen von Nukleinsäuren beantwortet werden, z. B. alle Motive von Proteinen eines bestimmten Organismus [GS01].

Die homogene Schnittstelle benutzt einen Mediator und Wrapper für die Schaffung einer virtuellen Single Datenquelle.

7.1. Architektur von TAMBIS

Das System besteht aus einem User Interface, der Ontologie, Query Planner, Sources-and-Services-Model sowie einem Wrapper Service (siehe Abbildung 7-1).

Die Ontologie spezifiziert das globale biologische Modell / Schema und unterstützt Dienste um Fragen über konzeptionelle Modelle zu beantworten (z. B. was über Proteine ausgesagt werden kann, oder allgemeiner: was Eltern von Konzept X sind). Durch die Ontologie werden dem User-Interface automatisierte (Teil-)Queries zur Verfügung gestellt, die mit anderen verknüpft werden können.

Wenn der Benutzer eine globale Query zusammengestellt hat, wird diese mit Hilfe des Query-Planner in einen ausführbaren Queryplan transformiert. Der Query-Planner verwendet das Sources-and-Services-Model, das unter anderem Informationen über Quellen und Kosten der Teilqueries enthält, die zusammen die globale Query bilden. Dabei entstehende Alternativen werden bewertet. Die günstigste Alternative wird jeweils zur Ausführung gewählt und dem Wrapper-Service als Aufgabe übertragen.

---

[4] eine Ontologie ist in diesem Zusammenhang ein logikbasiertes Verfahren zur Vernetzung von in Kontext stehenden Informationen, die in den unterschiedlichen Datenbanken vorhanden sind. Sie ist eine formalisierte, explizite, Spezifizierung eines gemeinsamen Konzepts.

[5] unter Motiven versteht man Aneinanderreihungen von bestimmten Aminosäuren, die in (vielen) Proteinen wiederholt vorkommen

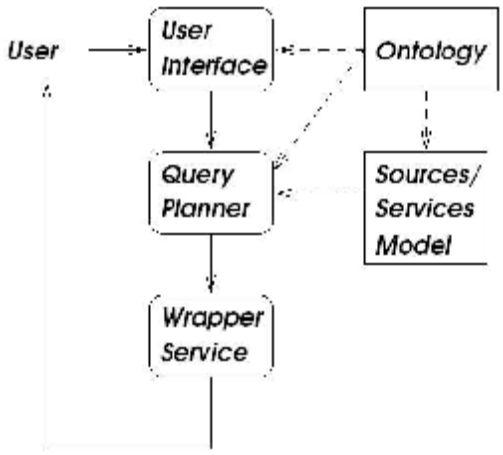

Abbildung 7-1: TAMBIS Architektur [GS01]

7.2. Benutzung des Systems

Der Benutzer interagiert mit der Benutzerschnittstelle (User-Interface), einem Java-Applet (siehe Abbildung 7-2).

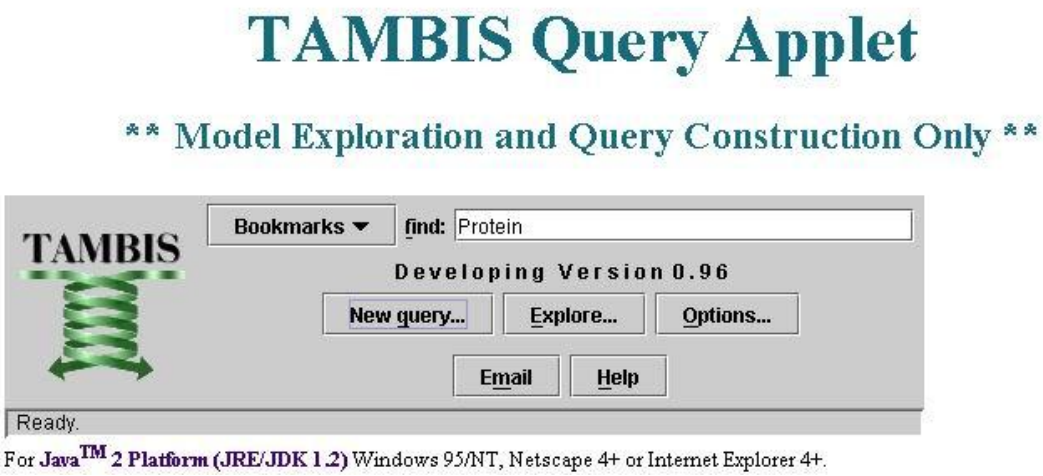

Abbildung 7-2: TAMBIS - Java-Applet

Er hat zunächst die Möglichkeiten eine neue Query zu erstellen, oder mit Hilfe des Explorers (siehe Abbildung 7-3) im Biological-Model nachzusehen, was die integrierten Datenbanken zu bieten haben.

Der Explorer zeigt die Ontologie von TAMBIS, quasi das globale biologische Modell / Schema, welches alle verknüpften Informationen besitzt und somit Auskunft gibt in welcher Art und Weise die darunter liegenden Datenbanken abgefragt werden können.

Mit Hilfe von *Query-Formulation-Dialogues* des Query-Builders (siehe Abbildung 7-4) werden Queries in Form von Ausdrücken aus dem Biological-Model geformt. Der Dialog wird durch die Ontologie gesteuert, und führt den Benutzer durch ein grafisches Menü mit vorgefertigten Query-Templates.

Eine Beispielanfrage die Tambis lösen würde ist: „Motive zu finden, die Komponenten von Guppy-Proteinen sind“ (siehe Abbildung 7-4)

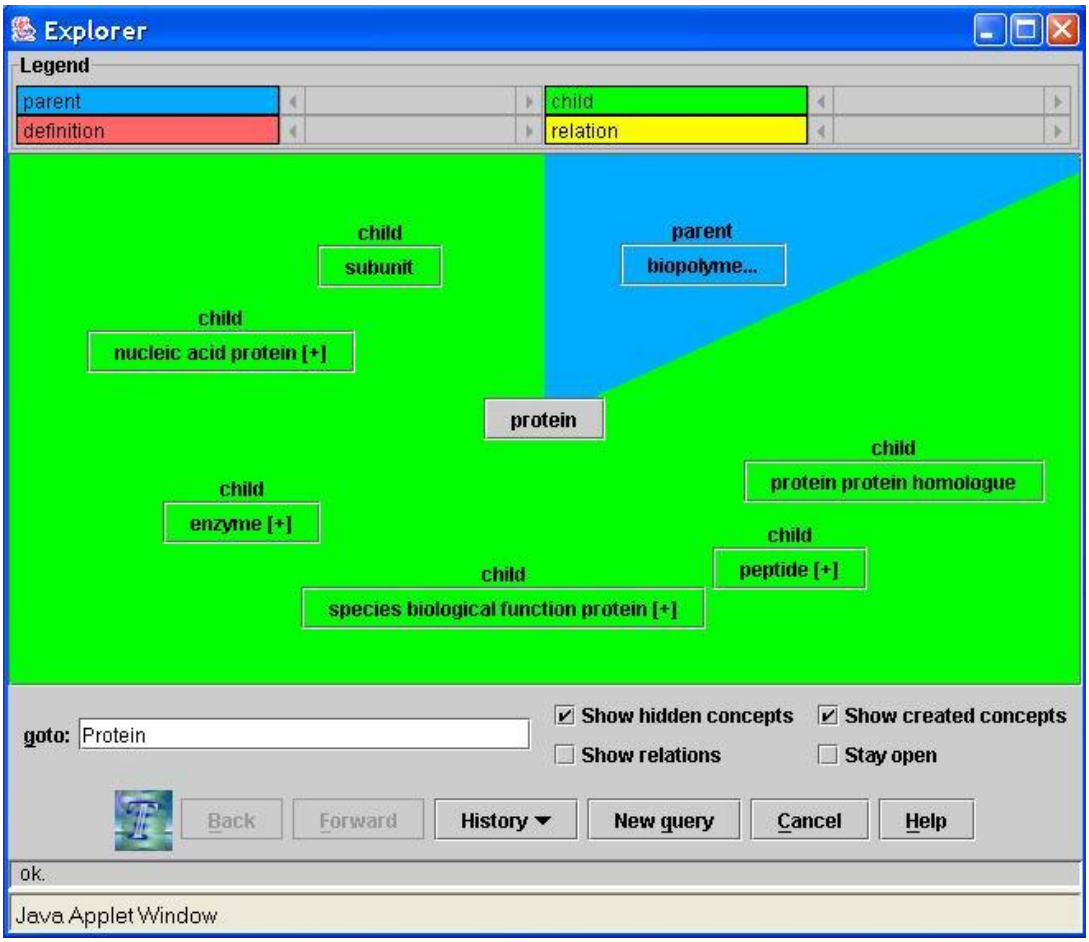

Abbildung 7-3: Tambis Explorer

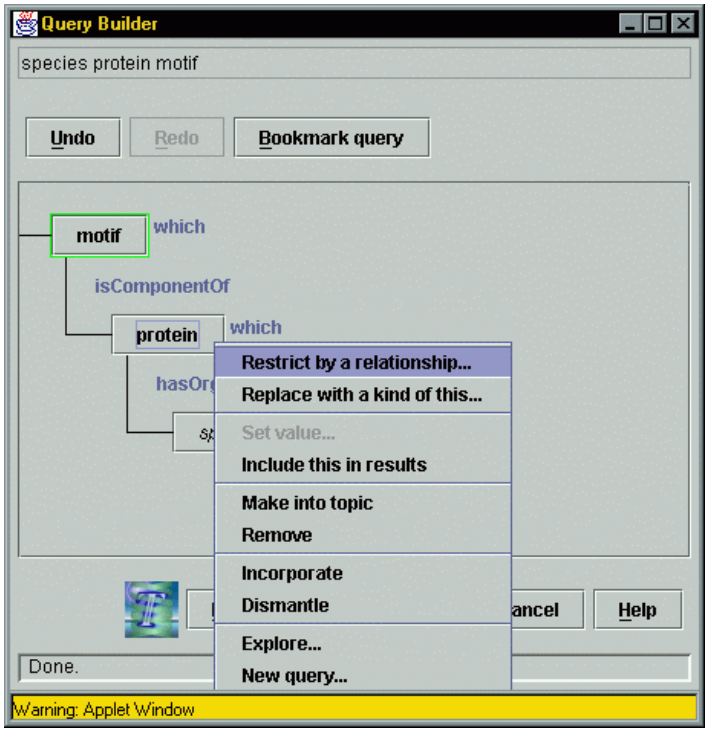

Abbildung 7-4: TAMBIS Query-Builder

7.3. Mediator

Der Mediator besteht aus der Ontologie und dem Query-Planner (siehe Abbildung 7-1). Die Ontologie ist eine konzeptionelle Wissensbasis (Conceptual-Knowledge-Base) der Molekularbiologie, die in der Beschreibungslogik GRAIL (Description-Logic-GRAIL[6]) formuliert ist. Diese Beschreibungslogik ist eine strukturierte Wissens- und Darstellungssprache, mit der Konzepte automatisch auf der Basis von Vater-Sohn-Beziehungen angeordnet werden können, wie z. B. „ist-Komponente-von“. Durch sie wird ein universelles Modell beschrieben, mit der automatisch (Teil-)Queries formuliert werden. Der Benutzer kann diese (Teil-)Queries sukzessive zu einer globalen Query zusammensetzen. Ausgehend von einer globalen Query wird mit Hilfe der Description-Logic ein Queryplan ausgearbeitet, der in CPL geschrieben ist.

Eine globale CPL-Query, die Motive die Komponenten von Proteinen des Organismus Guppy zurückgibt sieht beispielsweise so aus:

```
{m | \p<-get-sp-entry-by-os("guppy"),
 \m<-do-prosite-scan-by-entry-rec(p)}
```

Der Teil vor dem | ist der Projektionsausdruck, in welchem die Ergebnisse - hier alle Motive m zurückgegeben werden. Die 2 Funktionsaufrufe im Body der Query (rechts neben |) sind Generatoren, die Werte von verschiedenen gewrappten Quellen zurückgeben. Die 1. Zeile im Body zeigt, dass die neue Variable p jeden Wert enthält, der im Ergebnis der Funktion „get-sp-entry-by-os“ mit dem Parameter „guppy“ auftritt (get SwissProt entry by organism source „guppy“). Der Funktionsname verweist auf die SwissProt-Datenbank, die Daten über Proteine enthält. Die 2. Zeile weist der Variable m Ergebnisse der Funktion „do-prosite-scan-by-entry-rec“ (scanne die ProSite-Datenbank nach Motiven, die Komponenten zu gegebenen Proteineinträgen sind) zu, die Motive zu Proteinen aus p zurückgibt.

7.4. Wrapper

Wrapper sind in TAMBIS im Sources-and-Services-Model (SSM) integriert. Sie stellen Funktionen dar, die in CPL geschrieben sind und erzeugen die Illusion einer gemeinsamen Anfragesprache für jede Informationsquelle. Jede CPL-Funktion von TAMBIS hat 6 Attribute:

- name: Funktionsname
- arguments: Liste der Argumente und ihren Typen
- resulttype: Ergebnistyp
- cardinality: durchschnittliche Anzahl der Ergebnisse, die generiert werden
- cost: durchschnittliche Antwortzeit
- source: Quelle der Informationen

CPL-Funktionen können im Zusammenhang mit der Tambis-Ontologie in drei Kategorien eingeteilt werden:

- Iteratoren, die Instanzen von Konzepten aus den Quellen erhalten (Beispiel für einen Iterator: „get-all-sp-entries“)
- Rollenevaluierer, die Werte für Rollen von Instanzen testen oder berechnen (Beispiel einerRolle:„Protein which hasFunction Receptor“)

[6] Abk für GALEN Representation and Integration Language

- Filter, die für die Anfrage irrelevante Instanzen entfernen (Beispiel für einen Filter: „check-sp-entry-for-hydrolase“)

Eine Implementation eines Iterators wird hier anhand der Funktion „get-all-sp-entries“ gezeigt. Sie gibt alle Proteineinträge der Datenbank SwissProt zurück.

```
<name: "get-all-sp-entries";
 arguments: [];
 resultType: "protein_record";
 cardinality: 80000;
 cost: 8000.0
 source: "SwissProt">
```

8. GIMS ein Beispiel eines Data Warehouses

GIMS - Genome-Information-Management-System ist ein objekt-orientiertes Data-Warehouse, das Genome und assozierte Informationen speichert. Es ist ebenfalls ein Forschungsprojekt der „School of Biological Sciences“ und der „Information Managment Group“. Es stellt effektive Techniken zur Speicherung und komplexen Analyse von genetischen Informationen bereit.

Daten werden hauptsächlich vom Munich-Information-Center-for-Protein-Sequences (MIPS) und assozierte Informationen via Wrapper von anderen Datenbanken, wie Yeast-Proteome-Database (YPD), Stanford-Microarray-Database (SMD) oder Kyoto-Encyclopedia-of-Genes-and-Genomes (KEGG) eingeladen. MIPS stellt genom-spezifische Datenbanken und Datenbanken mit funktionalen Annotationen von Proteinsequenzen zur Verfügung. Die YPD enthält physikalische, funktionale und genetische Informationen über Saccharomyces cerevisiae (Hefe). Die SMD speichert Daten von Microarray-Experimenten. KEGG sind Datenbanken, die Informationen über Gene, Proteine und Pathways bereitstellen [PK00]. Manche Informationen, z. B. Protein-Protein-Interaktionen, sind auch mit Hilfe von speziellen Algorithmen [IC00] von Datenbanken, wie ProSite, berechnet.

Dabei werden genetische Sequenzdaten, funktionale Daten von Transkriptomen[7], Metabolomen[8], Metabolic-Pathways[9], Proteome[10] [So02]und Protein-Wechselwirkungen in einem Single-Data-Warehouse integriert. Ein Anwendungsbeispiel für GIMS ist die Erforschung möglicher Abhängigkeiten zwischen der Lokalisation von Genen auf Chromosomen und der Lokalisation von Proteinen innerhalb der Zelle, die von diesen kodiert wurden.

Derzeit existierende Versionen sind die Fungal- (FuGIMS) und die Prokaryote-Version. FuGIMS wurde für Hefe-Genome[11] entwickelt. Die Prokaryote-Version speichert Informationen über Bakterien (E.coli, B.subtilis, M tuberculosis) [PK01].

[7] vollständige Menge funktionaler Zusammenhängen von Genen und Proteinen

[8] gesamte Menge von Metaboliten (Produkte von interzellulären, biochemischen Prozessen) von Zellen, Gewebe oder Organen

[9] Menge von zusammenhängenden biochemischen Reaktionen (Stoffwechselvorgänge) in einer Zelle

[10] alle Protein-Moleküle von bestimmten Zellen, Gewebe oder Organen

[11] Saccharomyces cerevisiae

8.1. Architektur von GIMS

Der Kern des Warehouses ist eine ODMG[12]-objekt-Datenbank „FastObjects“ von POET. Der Datenbankzugriff erfolgt durch das ODMG-Java-Binding, welches den Zugang durch normale Java-Klassen erlaubt, und die deklarative Anfragesprache OQL verwendet.

Die GIMS-Datenbank wird über eine Java-Anwendung benutzt, die über das Internet zum Download bereitsteht [GI03]. Die Anwendung macht es dem Benutzer möglich durch die Datenbank zu browsen, und benutzt sog. Canned-Queries, die zur Analyse der Daten verwendet werden können.

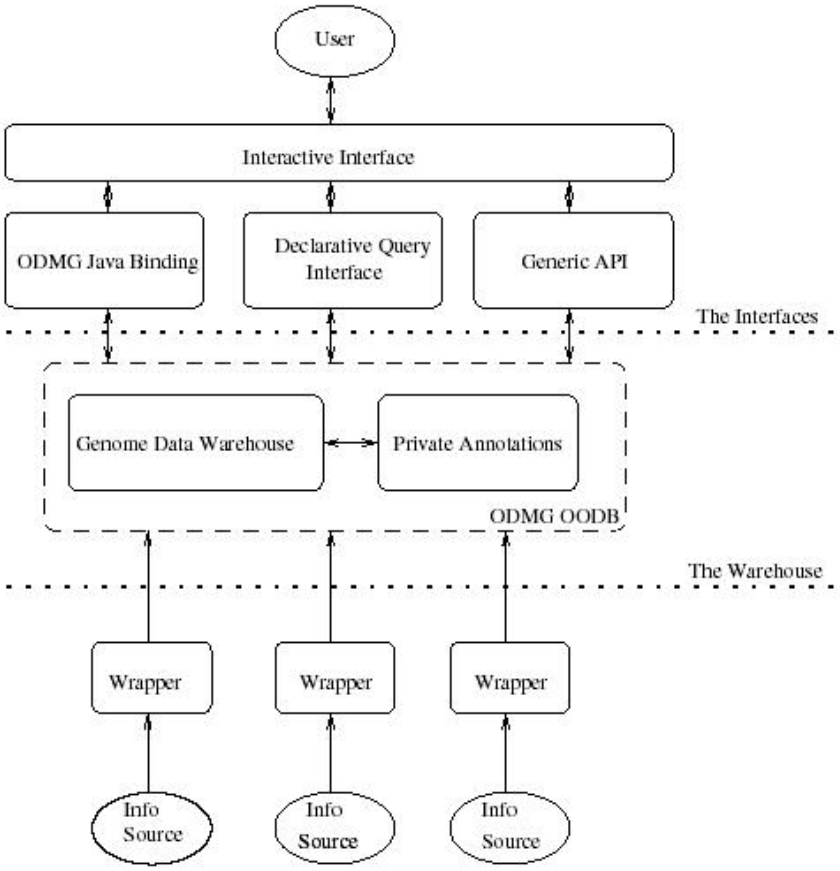

Abbildung 8-1: GIMS-Architektur [PK02]

Dimensionshierarchien

Das Schema von GIMS ist mit hierarchischen Dimensionen modelliert. Es besteht aus vier Dimensionen: „Genome-sequence-data“, „Protein-protein-interaction-data“, „Transcriptome-data“ und „Genome-Modifications“. Dabei beschreibt:

- „Genome-sequence-data“ Basis-Informationen über vollständige Genome
- „Protein-protein-interaction-data“ Protein-Wechselwirkungen als essentielle zellulare Prozesse, wie Signalgeber, DNA-Replikation und Metabolismen
- „Transcriptome-data“ den durch Experimenten beobachteten messbaren Zusammenhang zwischen Genom, Proteom und zellulären Phänotyp

[12] Object Data Management Group

- „Genome-Modifications“ die Auswirkungen von Veränderungen im Genom auf den Phänotyp des Organismus

In detaillierter Form soll hier nur die Protein-protein-interaction-data-Dimension gezeigt werden.

Abbildung 8-2 zeigt das Klassendiagramm von Protein-protein-interaction-data, welches mit UML modelliert ist. Jede Interaktion schließt zwei Proteine ein. Jede „Interaction“ kann von vielen „Experiment“-en beobachtet werden. Die Klasse „Complex“ enthält Informationen über Proteine, die in Gruppen von mehr als 2 an einer Interaktion teilnehmen. Die Klasse „Generic-Interaction“ wird benutzt, um Kategorien von Interaktionen zu beschreiben, die in vielen Spezies auftreten. Sie hat eine Beziehung zu der Klasse „Interaction“, denn diese kann eventuell in einer bestimmten Kategorie eingeordnet werden. Proteine können weiterhin Mitglieder von Familien sein, die wiederum eine „Generic-Interaction“ bilden können [PK02].

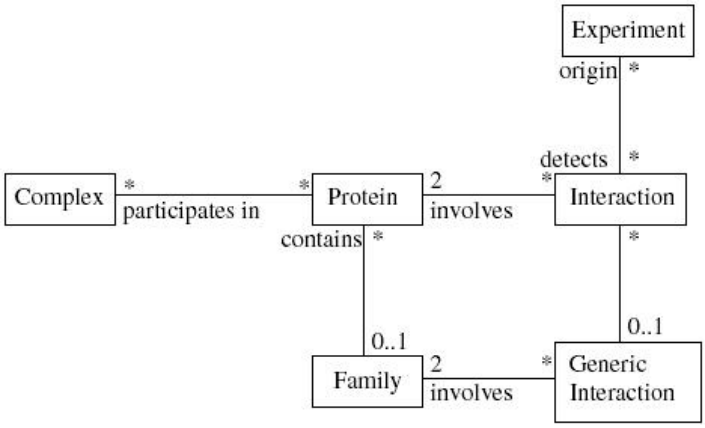

Abbildung 8-2: Klassendiagramm für Protein-protein-interaction-data

8.2. Benutzung von GIMS

Die Benutzung von GIMS erfolgt mittels einer Java-Anwendung. Es gibt drei Komponenten: „GIMS-Browser“, „Collection-Store“ und „Canned-Query-Interface“ (siehe Abbildung 9-1). Mit dem GIMS-Browser wird dem Nutzer möglich gemacht, durch alle Klassen zu browsen. Der Collection-Store erlaubt es, Sammlungen („Collections“) zu selektieren und zu kombinieren.

Das Canned-Query-Interface stellt dem Nutzer eine Liste von Canned-Queries zur Verfügung, die er selektieren und ausführen kann. Eine Canned-Query ist eine parametrisierte Anfrage auf einen Bereich, der vom Entwickler von GIMS in einer Java-Klasse implementiert wurde. Hier ein Beispiel: „Retrieve mRNA’s encoding proteins with a given function most up-regulated“, würde alle mRNA-Stücke zurückgeben die bestimmte Proteine kodieren.

9. Zusammenfassung

In der vorliegenden Arbeit wurde die Problematik der Integration von Biodatenbanken und verschiedene Lösungsansätze dazu aufzeigt. Dabei wurden beginnend mit der Ausgangssituation von Bioinformationsquellen, im ersten Teil allgemeine Aspekte, Probleme und Lösungsansätze einer Integration von heterogenen Datenbanken beschrieben. Der zweite Teil untermauerte diese theoretischen Kenntnisse letztlich durch Beispiele aus der Praxis.

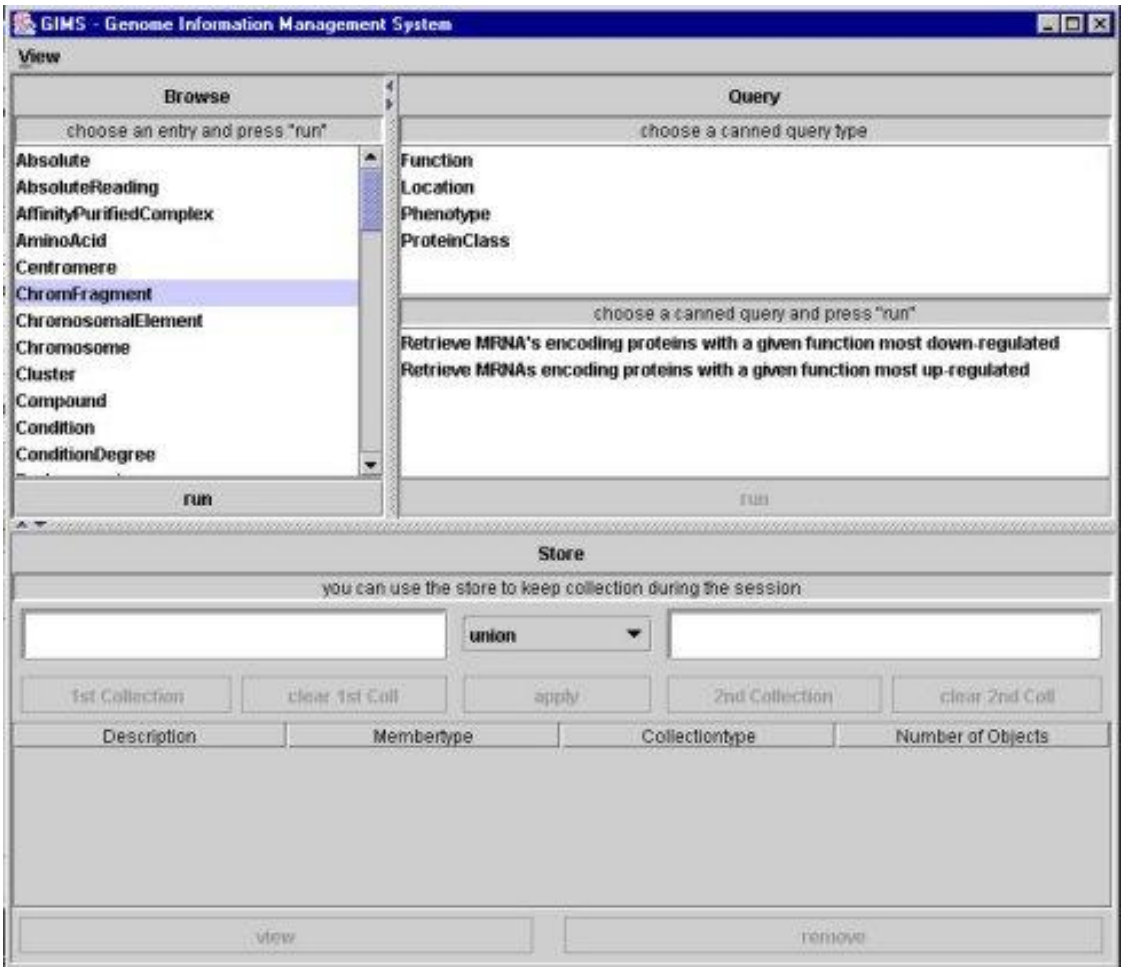

Abbildung 9-1: GIMS-Frontend [GI03]

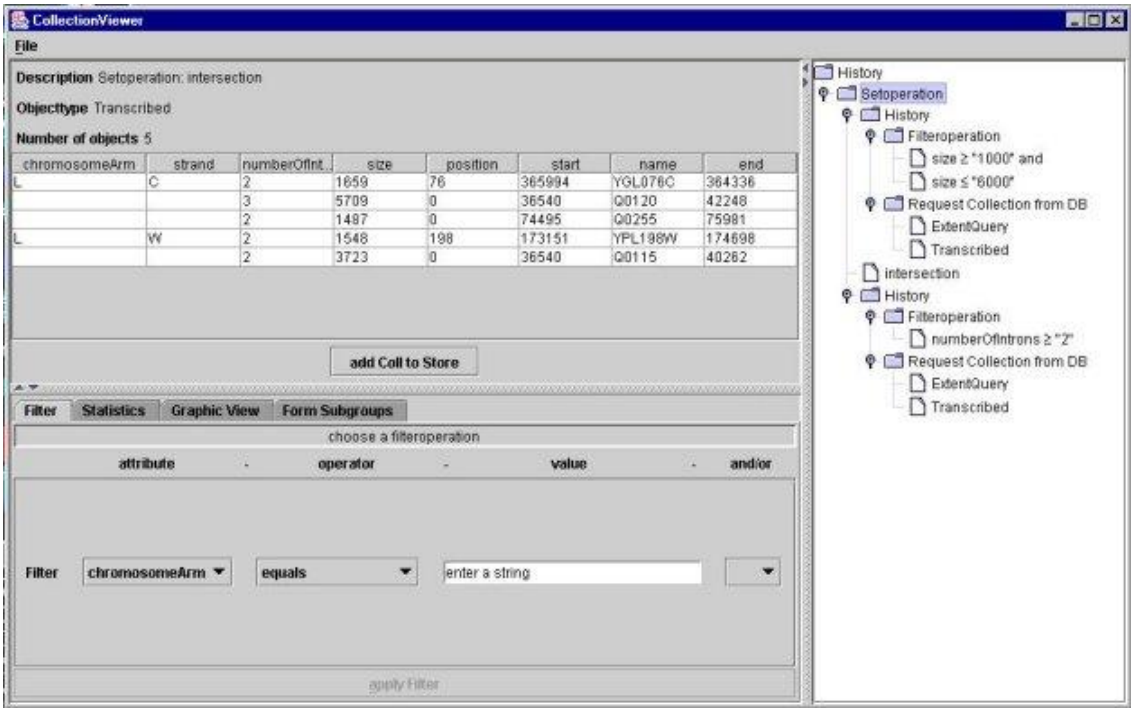

Abbildung 9-2: Browsing, Filtering and Combining GIMS [GI03]

Literaturverzeichnis

Be93 Bernstein, P.A.: *Middleware - An Architecture for Distributed System Service*. DEC Cambridge Research Lab., TR 93/6, March 1993

KP02 Peter D. Karp, Suzanne Paley and Pedro Romero: *The pathway tools software.* Bioinformatics Research Group, SRI International, 333 Ravenswood Ave, Menlo Park, CA, 94025, USA, 2002

Bi02 http://www.biocyc.org/

HR02 Do, Hong-Hai; Rahm, Erhard*: COMA - A System for Flexible Combination of Schema Matching Approaches*, 2002

PK00 Norman W. Paton, Shakeel A. Khan, Carole A. Goble: *Conceptual modelling of genomic information,* 2000

RS02 J. Reich, D. Schober, Max Delbrück Centrum für Molekulare Medizin, Bioinformatik, Berlin: *Datenbanken in der Molekularbiologie und Genetik, 2002*

ER94 Erhard Rahm, *Mehrrechner-Datenbanksysteme* 1994

ER01 Erhard Rahm, *Data Warehousing und Data Mining- Kapitel 4* Wintersemester 2001/2002

ER02 Erhard Rahm, *Data Warehousing und Data Mining - Kapitel 4,* Sommersemester 2002

PK01 Cornell, M., Paton, N, Goble, C.A., et al.: *GIMS – A DataWarehouse for Storage and Analysis of Genome Sequence and Functional Data*. Proceedings 2nd IEEE International Symposium on Bioinformatics and Bioengineering (BIBE'2001): 15-22.

PK02 Norman W. Paton, Shakeel A. Khan, et. al: *Conceptual modelling of genomic information,* 2000

GI03 http://www.cs.man.ac.uk/img/gims/software.html

DO95 S.B. Davidson, C. Overton, P. Buneman *Challenges in Integrating Biological Data Sources, 1995*

So02 Steve Oliver: From genotype to phenotype: *Bioinformatic tools for functional genomics*, School of Biological Sciences University of Manchester 2002

GS01 Goble, C.A., Stevens, R. et al.: *Transparent Access to Multiple Bioinformatics Information Sources. IBM Systems Journal Special issue on deep computing for the life sciences,* 40(2):532 – 552, 2001

IC00 Takashi Ito*†, Tomoko Chiba*, et. al: *A comprehensive two-hybrid analysis to explore the yeast protein interactome*, 2000

Reece K. Hart1,2, Ajay K. Royyuru1*, et al: *Systematic and Fully Automated Identification of Protein Sequence Patterns,* 2000

Risi et. al: *Exloring the metabolic and genetic control of gene expression on a genome scale,* Science, 278, 1997

Chu et. al: *The transcriptional program of sporulation in budding yeast*, Science, 1998