Integration durch Standards: Erfahrungen mit CORBA in Life Science Research

Heiko Müller[§], Ulf Leser[±]

[§]Max-Planck-Institut für Molekulare Genetik
Ihnestr. 73, D - 14195 Berlin
Email: mueller_h@molgen.mpg.de

[±]Technische Universität Berlin,
Fachbereich 13 - CIS
Einsteinufer 17, D- 10587 Berlin.
Email: leser@cs.tu-berlin.de.

Abstrakt: Der einheitliche Zugriff auf verteilte und autonome Datenquellen wird in vielen Bereichen durch die vorhandene Heterogenität erheblich erschwert. Eine Möglichkeit zum Umgang mit dieser Heterogenität liegt in der Definition von geeigneten Zugriffsstandards. Dieser Weg wird beispielsweise von der OMG durch die Definition von Domainstandards gefördert.
In unserer Arbeit skizzieren wir Erfahrungen aus der Implementation und Evaluation eines vorgeschlagenen Standards für den Zugriff auf Genomkarten. Wir beschreiben sowohl den vorgeschlagenen Standard als auch die einzelnen Schritte seiner Umsetzung auf eine existierende molekularbiologische Datenbank. Aus den dabei gewonnen Erfahrungen leiten wir eine Reihe von Verbesserungsvorschlägen ab.

1. EINLEITUNG

Molekularbiologische Forschung ist heutzutage undenkbar ohne den massiven Einsatz von Computern. Insbesondere die Verwaltung der Datenmengen, die sich aus der Vielzahl an weltweit durchgeführten Experimenten in diesem Gebiet ergibt, ist ohne den Einsatz von Computern und Datenbanktechnologie nicht zu bewältigen. Die Hauptschwierigkeit liegt dabei meistens nicht in der puren Masse an Daten, sondern in ihrer verteilten Erzeugung, die eine hohe semantische, technische und syntaktische Heterogenität zur Folge hat.

Beispielsweise sind die Daten, die im Rahmen des „Human Genome Projects“ [4] erzeugt werden, das Ergebnis einer grossangelegten, internationalen Forschungsanstrengung. Beteiligt sind eine Vielzahl von Forschergruppen aus über einem Dutzend verschiedener Nationen. Zusätzlich zu diese gemeinsamen Anstrengung tragen auch viele beteiligte Länder mit eigenen, nationalen Forschungsprogrammen zum hohen Datenaufkommen bei, wie zum Beispiel Deutschland mit dem „Deutschen Humangenomprojekt“[1]. Die verschiedenen Projekte unterliegen dabei einer bestenfalls als lose zu bezeichnenden Koordination, was häufig zu sich überlappenden Anstrengungen führt. Parallel zu diesen staatlich finanzierten Anstrengungen ist in den letzten Jahren auch eine starke Beteiligung von privaten Investoren entstanden.

So lose wie bei der Koordination der Forschungen wird auch im Bereich der Datenintegration vorgegangen. Der Großteil der verteilt erzeugten Informationen wird auch verteilt gehalten, d.h.,

[1] http://www.dhgp.de

in der Regel werden die erzeugten Daten von den einzelnen Gruppen separat gespeichert und verwaltet. Während sich für bestimmte Datentypen zusätzlich dazu eine zentrale Speicherung durchgesetzt hat (beispielsweise für genomische Sequenzen), werden viele andere Ergebnisse nur über Fachpublikationen an die Öffentlichkeit gebracht [15]. Zwar stellen viele Institute ihre Daten auch über WWW Schnittstellen zu Verfügung, aber ein homogener oder sogar integrierter Zugriff auf einen nennenswerten Anteil der weltweit theoretisch vorhandenen Daten ist heute nicht möglich. Statt dessen hat man die Auswahl zwischen ca. 500 verschiedenen Datenbanken ([7], Stand September 1999), die verschiedene Modelle benutzen, unterschiedliche Zugriffsmechanismen unterstützen und nur in proprietären Formaten erhältlich sind. Zusätzlich dazu sind viele Begriffsdefinitionen heterogen, was eine semantische Integration extrem erschwert.

Die Integration der Daten wird heutzutage als eines der „most pressing problems“ molekularbiologischer Forschung angesehen [19]. Integrierter Zugriff, verbunden mit einer zentralen Übersicht über die vorhandenen Daten, kann helfen, neue Zusammenhänge aufzuzeigen, die Wiederholung von Experimenten zu vermeiden und die Qualität der Daten im allgemeinen zu erhöhen [24]. Das Haupthindernis auf dem Weg zur Datenintegration liegt in der Autonomie der verschiedenen Datenhalter; ein Umstand, an dem sich in absehbarer Zeit auch keine Änderung ergeben wird.

Datenintegration kann unter verschiedenen Paradigmen erfolgen. Föderierte Datenbanken beispielsweise beruhen auf einer Übersetzung von Anfragen gegen ein globales Schema in eine Menge von Anfragen gegen Datenquellen zum *Zeitpunkt der Anfrage* (virtuelle Integration). Data Warehouse Architekturen dagegen gehen von Integration aller Daten aus allen Quellen in eine zentrale Datenbank aus (in-advance Integration [25]). Während Warehouses mit dem hohem Datenvolumen und der Update - Problematik zu kämpfen haben, leiden föderierte Datenbanken unter dem hohen Aufwand einer dynamischen Anfrageübersetzung bei hoher semantischer Heterogenität.

Insbesondere im industriellen Umfeld wird daher häufig versucht, einen dritten Weg zu beschreiten, der die Vorteile der beiden oben genannten Möglichkeiten vereint und die Schwierigkeiten wenigstens teilweise umgeht: Die Definition von Standards, d.h., von einheitlichen Konventionen für die Abfrage von Daten. In der vorliegenden Arbeit berichten wir von entsprechenden Erfahrungen mit einem zur Zeit entstehenden Zugriffsstandard im Bereich der Genomkarten. Dieser basiert auf der „Common Object Request Broker Architecture“(CORBA) als technische Infrastruktur.

Vorläufer des hier beschriebenen Vorschlags wurden bereits erfolgreich eingesetzt [21; 6]. [8] beschreibt ein Visualisierungstool, das Datenbanken über CORBA anspricht und zur Zeit auch an den entstehenden Standard angepaßt wird. Andere Systeme, die mit CORBA im Bereich Bioinformatik experimentieren, werden z.B. in [3] und [9] beschrieben.

Im nächsten Abschnitt werden wir zunächst den Ansatz „Integration durch Standard“ auf Basis von CORBA näher charakterisieren. Abschnitt 3 gibt einen Überblick über unser Anwendungsgebiet der Genomkarten und geht auf Hindernisse und Möglichkeiten einer Integration in diesem Umfeld genauer ein. In Abschnitt 4 wird der vorgeschlagene Standard genauer beschrieben. Abschnitt 5 gibt unsere Erfahrungen bei ihrer Umsetzung auf eine existierende Datenbank wieder. Dies muss als Normalfall gegenüber der Verwendung in neuentwickelten Systemen angesehen werden, da sich ein Standard nur dann durchsetzen kann, wenn er von existierende Datenhalter befürwortet und implementiert wird. In Abschnitt 6 leiten wir aus den gewonnen Erfahrungen verschiedene Verbesserungsvorschläge ab. Details zu vielen Punkten können in [16] nachgelesen werden.

2. INTEGRATION DURCH STANDARD

Datenintegration auf der Basis von gemeinsamen Zugriffsstandards beruht auf einer einfachen Überlegung: Alle Datenhalter (im folgenden: Server) implementieren einen festen Satz von Zugriffsoperationen auf die von ihnen verwalteten Daten, der von beliebigen Programmen oder Personen (im folgenden: Klient) benutzt werden kann. Durch die Verwendung einheitlicher Zugriffsoperationen werden die syntaktischen Heterogenitäten umgangen. Technische Heterogenität ist durch einen Zugriffsstandard zunächst nicht adressiert; im vorliegenden Fall wird von der Verwendung einer objekt-orientierten Middleware, basierend auf CORBA, ausgegangen. Semantische Heterogenität muss durch eine ausreichende Dokumentation der Zugriffsmethoden überwunden werden. Diese Vorgehensweise findet man beispielsweise in der Automobilindustrie (STEP und EXPRESS) und im medizinischen Bereich (HL-7), sowie neuerdings auch für den eCommerce (auf der Basis von XML).

Ein prominenter Vertreter für die Festlegung von Standards ist die „Object Management Group“[2]. Dieses Konsortium von über 500 Soft- und Hardwareherstellern hat als technische Infrastruktur zunächst die „Common Object Request Broker Architecture“ spezifiziert. CORBA basiert auf einer objekt-orientierten Brokerarchitektur *(ORB: Object Request Broker) sowie einer Sprache für die Definition von Schnittstellen entfernter Objekte, der „Interface Definition Language“ (IDL). Erweitert wird CORBA durch eine Menge von sogenannten „Common Services“, die typische Dienste übernehmen sollen, welche in verteilten Objektsystemen anfallen. Beispiele sind das Auffinden von Objekten anhand von Namen (Naming Service) oder Eigenschaften (Trader Service). Aufbauend darauf beschäftigt sich eine Reihe von „Task Forces“ mit der Definition von domänenspezifischen Standards, beispielsweise für den Finanzbereich[3], die medizinische Datenverarbeitung[4] und die Bioinformatik[5].

Ein von der OMG verabschiedeter Domain Standard besteht aus einer Schnittstellendefinition in IDL sowie ihrer Dokumentation. Datenquellen, die mit dem Standard konform sein wollen, müssen einen ORB installieren und einen CORBA Server schreiben, der den Standard exakt implementiert. Da eine formale Sprache wie IDL naturgemäß nicht ausreicht, um komplexe semantische Zusammenhänge ausreichend zu definieren, muss der Server zusätzlich sicherstellen, dass die von ihm implementierten Methoden die in der Dokumentation beschriebene Funktionalität erbringen. Dies ist im Bereich Bioinformatik nicht immer trivial: Beispielsweise ist ohne zusätzliche Dokumentation nicht klar, was eine Methode wie „`getAllGenes()`“ tatsächlich liefern soll, da das Konzept „Gen“ selber keine einheitliche Definition besitzt [20].

Für einen Klienten wird dagegen vieles einfacher, wenn die für ihn wichtigen Datenquellen einen Standard implementieren. Im Grunde werden alle klassischen Probleme der Datenintegration ohne weiteren Aufwand seinerseits gelöst (siehe oben). Verlangt wird nur ebenfalls die Installation eines ORB sowie die Implementation CORBA-fähiger Software (siehe Abbildung 1). Beides stellt bedeutend weniger Aufwand als die quellspezifische Ansteuerung proprietärer Interfaces dar. Nicht gelöst wird durch einen Zugriffsstandard dagegen die Problematik der tatsächlichen Integration der Informationen: Wenn beispielsweise zwei Quellen sich widersprechende Aussagen zu einem bestimmten Sachverhalt machen, obliegt es noch immer dem Klienten, eine geeignete Konfliktlösungsstrategie zu finden.

---

[2] http://www.omg.org
[3] Financial Services DTF, http://www.omg.org/homepages/fdtf
[4] CORBAmed DTF, http://www.omg.org/homepages/corbamed
[5] Life Science Research DTF, http://www.omg.org/homepages/lsr

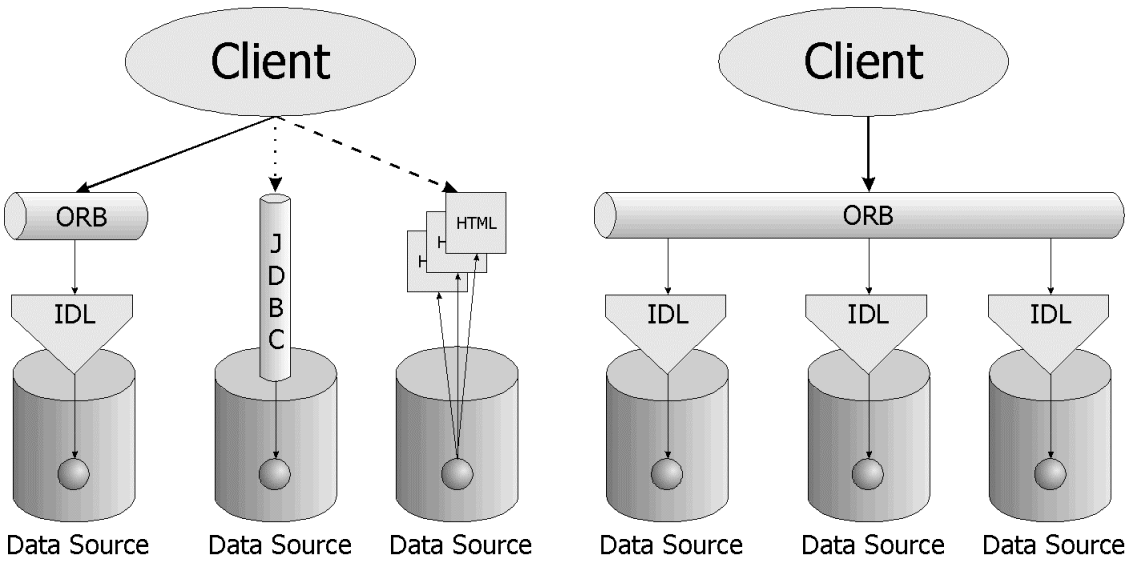

Abbildung 1: Zugriff über heterogene Interfaces versus Zugriffsstandard.

3. ZUGRIFF AUF GENOMKARTEN

Eine der wichtigsten Eigenschaften lebender Organismen ist ihre Fähigkeit zur Selbstreplikation und -montage. Die hierfür benötigten Informationen sind in Form der Desoxyribonucleinsäure (DNA) gespeichert. Sie ist in diskrete Abschnitte unterteil, die Chromosomen genannt werden. Auf den Chromosomen befinden sich in unregelmäßigen Abständen Gene. Jedes von ihnen stellt die konkrete Bauanleitung für eine Komponente des Organismus dar. Die Gesamtheit der DNA wird als Genom bezeichnet. Unterschiedliche Organismen besitzen selbstverständlich unterschiedliche Genome.

Genomkarten sind eine abstrakte, strukturelle Darstellung eines Genoms oder Teilen davon, meist auf der Basis der Chromosomen. Dargestellt werden u.a. die Anordnung von Genen und Klonen, sowie die Lage ausgezeichneter Referenzpunkte, sogenannter Marker. Diese Karten sind wichtig, um die Struktur der Gene, ihre Funktionen und ihre Evolution verstehen zu können. Auch in der Sequenzierung von Genomen werden Genomkarten benötigt.

Es existiert eine Vielzahl unterschiedlicher Arten von Genomkarten. Bei der *genetischen Kartierung* werden die Kopplungen von Genen durch die Betrachtung von Vererbungshierarchien untersucht. Eine *physikalische Karte* verfügt in der Regel über mehrere Hierarchiestufen, die, ausgehend von sehr großen Fragmenten eines Chromosoms (sog. Klone) über kleinere Fragmente bis zu einzelnen Basenpaaren gehen kann (Abbildung 2) [18].

3.1 Datenquellen für Genomkarten

In diesem Abschnitt geben wir einen kurzen Überblick über einige typische Datenquellen für Genomkarten, um einen besseren Eindruck von den in der Einleitung bereits erwähnten Schwierigkeiten zu erhalten. Weitere Beispiele kann man in [14] finden.

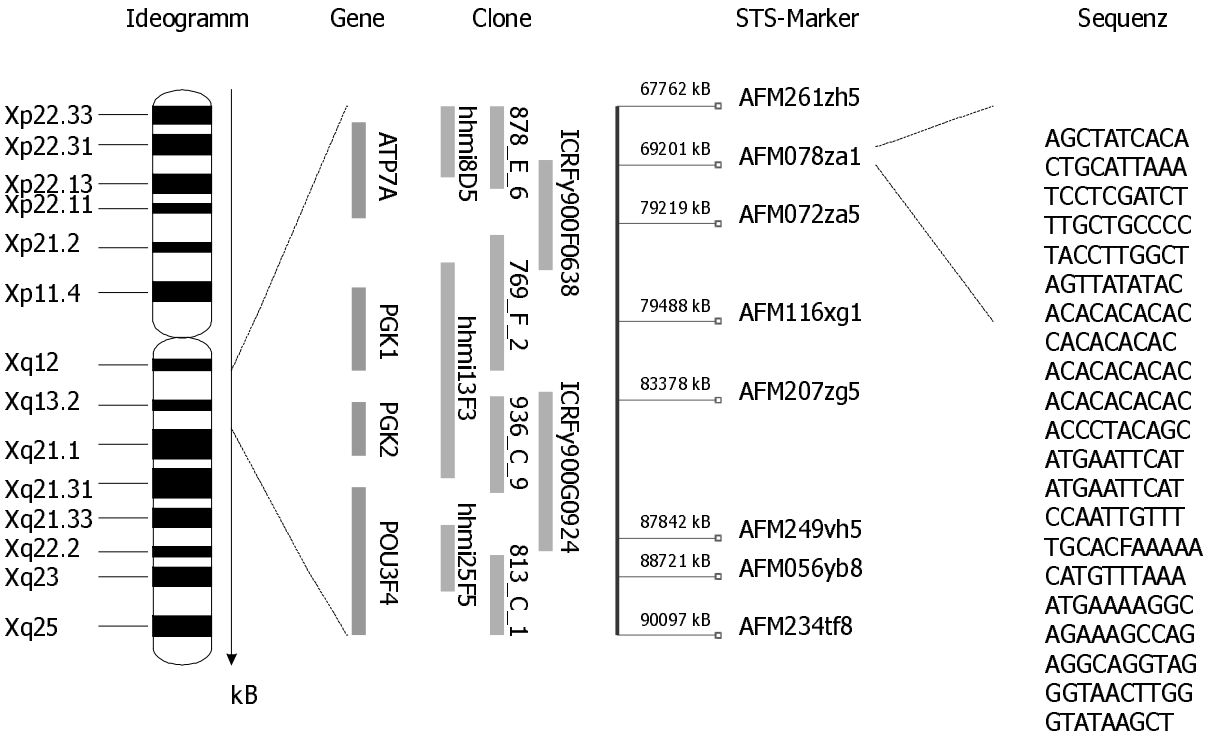

Abbildung 2: Beispiel einer physikalischen Genomkarte.

Die Genome Database (GDB) [13] fungiert als öffentliches Repositorium für Daten aus diesem Bereich. Der Inhalt von GDB stammt im wesentlichen aus direkten Datensubmissions, die von den Datenerzeugern vorgenommen werden können. GDB konnte aber niemals auch nur annähernd den gleichen Vollständigkeitsgrad erreichen wie die öffentlichen Sequenzdatenbanken EMBL oder Genbank [23]. GDB speichert Daten in der Form, in der sie eingesandt werden und nimmt keine weitere Integration vor. Entsprechend gibt es eine Vielzahl von Karten pro Chromosom bzw. Chromosomabschnitt. GDB ist über ein formbasiertes WWW Interface ansprechbar, mit dessen Hilfe es z.B. möglich ist, Daten nur aus bestimmten Regionen zu erhalten. Darüber hinaus existieren Perl- und C-Bibliotheken, mit deren Hilfe man Anfragen über ein spezielles Protokoll absenden kann.

Eine weitere Datenbank ist HuGeMap [1]. Neben zwei über das WWW zugänglichen Java-Applets zur Visualisierung der Genomkarten wurde für HugeMap auch eine vorläufige Version der im weiteren erläuterten IDL-Spezifikation entwickelt. Für die Radiation Hybrid Database (RHdb) [21] existiert ebenfalls ein CORBA-Server sowie ein WWW Interface. Die wesentliche Verbreitungsform von RhDB ist aber ein nächtlich erstellter Datendump in einem speziellen ASCII Format, der über FTP bezogen werden kann.

Die Integrated X Chromosome Database (IXDB) speichert ausschließlich Informationen, die das menschliche X Chromosom betreffen [11]. Dabei werden unterschiedliche Daten aus den verschiedensten Quellen gesammelt und semantisch integriert, d.h., es wird versucht, identische Objekte zu erkennen und die Daten entsprechend redundanzfrei zu halten [10]. Für IXDB existiert eine z.Z. nur eine stichwortbasierte WWW-Schnittstelle. Genomkarten können entweder als HTML-Tabellen betrachtet oder mit Hilfe eines Java-Appletes (DerBrowser, siehe unten) visualisiert werden.

3.2 Vorteile eines einheitlichen Zugriffs

Aus dem vorangegangenen Abschnitt wird ersichtlich, dass Datenquellen für Genomkarten auf allen Ebenen heterogen sind, besonders im Bezug auf den Inhalt, die Zugriffsmechanismen und die Datenformate. Ein einheitlicher Zugriff auf diese Quellen verspricht eine Reihe von Vorteilen:

- **Integration vorhandener Genomkarten**: Einheitliche Zugriffsmechanismen würden die Integration vorhandener Daten, die die gleichen Bereiche des Genoms beschreiben, erheblich vereinfachen. Integrierte Genomkarten liefern einen besseren Überblick über bereits durchgeführte Forschungen. Es können Ungenauigkeiten in einzelnen Karten oder auch bisher nicht ausreichend untersuchte Regionen leichter identifiziert werden. Eine umfassende Kenntnis des aktuellen Wissenstands kann zur besseren Koordination und zur Vermeidung unnötiger Arbeiten ausgenutzt werden.
- **Vergleich von Genomkarten:** Der Vergleich von Genomkarten des gleichen Chromosomabschnitts einer Spezies läßt Aussagen über die Gemeinsamkeiten der experimentellen Vorgehensweise zu. Vergleicht man Genomkarten unterschiedlicher Spezies, so erhält man Anhaltspunkte darüber, wie man vorhandenes Wissen über gut kartierte Spezies auf weniger gut kartierte Spezies übertragen kann, z.B. von Mäusen auf Menschen (vergleichende Kartierung, [18]).
- **Visualisierung vorhandener Daten**: Flexible Visualisierungstools sind eine wesentliche Grundlage für viele Bereiche der Genomforschung. Es existieren bereits eine Reihe solcher Tools, die jedoch fest an eine oder wenige Datenquellen gebunden sind, wie z.B. DerBrowser [5]. Sie arbeiten mit proprietären Ansätzen und Datenformaten, meist aufbauend auf CGI-Scripten, und lassen sich entsprechend nur schwer auf andere Quellen portieren. Einheitliche Zugriffsmechanismen erleichtern die Entwicklung von solchen Applikationen dadurch, dass sie nur einmal entwickelt werden müssen und dennoch Daten aus beliebigen Quellen darstellen können.

Zusätzlich zu diesen domänenabhängigen Vorteilen erlauben Zugriffsstandards auch die ökonomische Entwicklung einer Reihe von weiteren wichtigen Verbesserungen, wie zum Beispiel die Implementation von semantischer Integration durch Mediatoren oder intelligente Replikationsmechanismen.

4 DER STANDARD: MAPIDL

Im folgenden soll kurz die von uns implementierte IDL-Spezifikation (im folgenden: MapIDL) zur einheitlichen Repräsentation und zum Einheitlichen Zugriff auf Genomkarten vorgestellt werden. Die MapIDL entstand als Antwort auf einen „Request For Proposals“ der OMG, und wird in [2] genauer beschrieben. Der aktuelle Status des Proposals weicht von der hier beschriebenen Fassung allerdings bereits erheblich ab, teilweise aufgrund der in dieser Arbeit beschriebenen Erfahrungen[6].

Der MapIDL liegt das in Abbildung 3 dargestellte Objektmodell zugrunde. Es läßt sich grob in zwei Bereiche unterteilen. Auf der rechten Seite befinden sich die verschiedenen Typen von Genomkarten sowie die sich auf ihnen befindlichen Elemente. Auf der linken Seite befinden sich all jene Klassen, die die Zuordnung (mapping) dieser Elemente zu Genomkarten realisieren. Außer den topologischen und logischen Zusammenhängen zwischen Karten und den sich auf ihnen

[6] Siehe http://www.omg.org/techprocess/meetings/schedule/Genomic_Maps_RFP.html

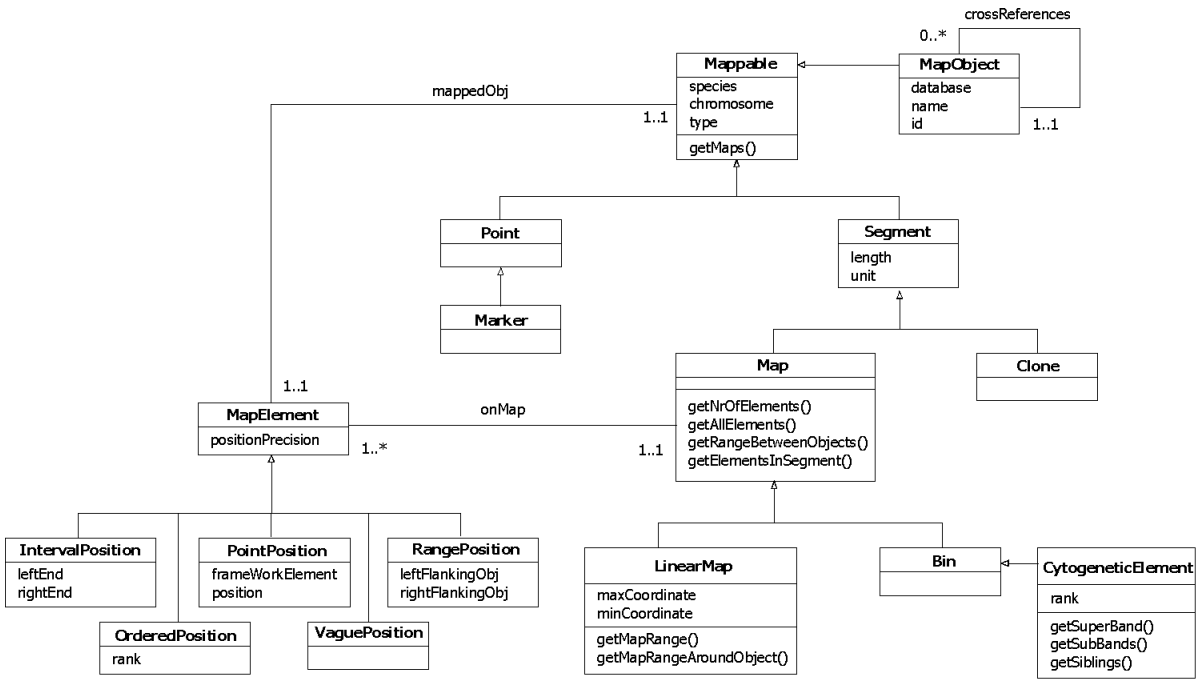

Abbildung 3: Das der MapIDL zugrunde liegende Objektmodell.

befindenden Elementen werden kaum zusätzliche Informationen gespeichert. Alle Objekte besitzen einen Namen und einen quellspezifisch eindeutigen Bezeichner (ID). Eine globale Eindeutigkeit, obwohl wünschenswert, kann aufgrund fehlender Konventionen nicht gewährleistet werden und wird deshalb auch nicht gefordert.

Elemente auf den Genomkarten werden anhand ihrer physikalischen Ausdehnung in Punkte und Segmente unterschieden. Zusätzlich enthalten die Objekte noch Angaben über das Chromosom, dem sie zugeordnet sind, die Spezies, der das Chromosom entstammt, und den Objekttyp. Dieser charakterisiert das Objekt über seine eigentliche Klasse hinaus noch genauer.

Segmente können z.B. Klone, Gene aber auch Genomkarten sein. Durch letzteres ermöglicht man einen rekursiven Aufbau von Genomkarten. Es werden mehrere Klassen von Karten unterschieden, von denen wir hier nur zwei darstellen: Die linearen Karten (`LinearMap`) besitzen alle ein numerisches und lineares Koordinatensystem. Die sog. Bin's dagegen besitzen kein Koordinatensystem; somit sind sie eigentlich pure Objektsammlungen. Bin's werden beispielsweise durch Radiation Hybrid Mapping generiert. Die Zuordnung von Objekten auf Genomkarten erfolgt über eine eigene Klasse (`MapElement`). Ein Element kann also mehrfach auf eine oder auf unterschiedliche Karten abgebildet werden. Beim Entwurf der IDL-Spezifikation wurde versucht, unsinnige Kombinationen von Elementen von vornherein auszuschließen. So ergibt es z.B. keinen Sinn, einem Marker eine linke und rechte Endposition zuzuweisen, da ein Marker per Definition keine Ausdehnung besitzt. Solche Restriktionen werden durch Subklassen von `MapElement` erreicht.

Die Generierung der Objekte erfolgt über sog. Factories. Sie bilden somit die eigentlichen Zugangspunkte zu der verwendeten Datenbank. Jede der Operationen dieser Factories liefert eine Referenz auf ein CORBA-Objekt (IOR) zurück. Die MapIDL sieht zwei Factories vor. Eine dient zum Generieren von Genomkarten (`MapFactory`) während die andere für andere Objekttypen, wie z.B. Marker und Klone zuständig ist (`MappableFactory`). Beide Factories bieten Möglichkeiten, Objekte beispielsweise anhand ihrer ID oder ihres Names zu suchen.

5. IMPLEMENTATION

Die vorgestellte MapIDL ist in JAVA 1.1.7 für die auf ORACLE 7.3 basierende Datenbank IXDB [11] implementiert worden (im fogenden: Mapserver). Als ORB kam der für akademische Zwecke frei verfügbare ORBacus zum Einsatz [17]. Für den Zugriff auf IXDB wurde JDBC 1.1 verwendet.

5.1 Abbildung der IDL-Interfaces auf IXDB

Die Anpassung des relationalen Schemas von IXDB an das objekt-orientierte Konzept der IDL erforderte zunächst einen Reverse Engineering Prozeß. Obwohl dabei auf die Entwickler von IXDB zurückgegriffen werden konnte, waren viele der feinen semantischen Unterschiede nur schwer zu entdecken. Die Konfrontation mit einer neuen Begriffswelt führte auch zur Aufdeckung von Inkonsistenzen innerhalb von IXDB. Anschließend wurde ein objekt-relationales Mapping zwischen der MapIDL und dem Schema von IXDB definiert.

Das Schema von IXDB entspricht generell einer vertikalen Fragmentierung des MapIDL-Modells. Dabei entspricht jede Relation einer Klasse des Objektmodells. Vererbte Attribute werden nur in der "allgemeinsten" Relation gespeichert. Über Integritätsbedingungen werden die entsprechenden Verbindungen zwischen den Datensätzen eines Objektes garantiert.

Eine Ausnahme bilden hierbei die Objekte der Klasse `LinearMap`. Physikalische Karten werden in IXDB in eigenen Relationen verwaltet, und besitzen auch eigene Primärschlüssel - aus diesem Grund kann der Primärschlüssel einer Klasse in IXDB auch nicht als eindeutiger Objektidentifier verwendet werden. Man muss statt dessen eine Kombination aus Klassenname und Primärschlüssel benutzen.

Im Server müssen die einzelnen Methoden der IDL-Interfaces in SQL-Anfragen gegen das relationale Schema von IXDB umgewandelt werden. Wir verwenden dafür parametrisierte, präkompilierte SQL Statements. Es hat sich dabei gezeigt, dass eine einfache 1:1-Umsetzung zwischen Methode und SQL-Anfrage nicht immer möglich ist. Für einige Methoden werden statt dessen bis zu drei SQL-Anfragen benötigt. Die statische Definition von Anfragen ist möglich, weil durch die MapIDL eine feste Menge von Anfragen definiert wird. Ad-hoc Anfragen werden nicht unterstützt. Diese können auch nicht schema-unabhängig formuliert werden, weshalb sie die Homogenität des Zugriffs unterlaufen [12].

5.2 Technische Probleme und Lösungen

Im folgenden beschrieben wir kurz eine Reihe von Problemen, die bei der Anbindung von Datenbanken über CORBA typischerweise auftreten. Wir begründen dabei auch die jeweils von uns gewählte Lösung.

Persistente vs. transiente Objektreferenzen:
Der bei ORBacus in der verwendeten Version enthaltene Basic Object Adapter (BOA) unterstützt keine *persistenten Objektreferenzen*. Nach einem Neustart des IXDB Mapservers sind deshalb sämtliche von Klienten noch gehaltenen Objektreferenzen ungültig. Dies ist insbesondere im Hinblick auf die IOR's der Factory-Objekte wichtig. Sie stellen ja den eigentlichen Zugangspunkt zum Mapserver dar und müssen deshalb jeweils wieder neu publiziert werden. Momentan erfolgt dies durch einen Eintrag in einer speziell dafür angelegten Datei, die von Klienten über HTTP gelesen werden kann. Eine andere Möglichkeit wäre die Verwendung des CORBA Naming Service.

Die Erzeugung persistenter Referenzen für die entsprechenden Objekte wäre mit Hilfe von ORBacus-spezifischen Methoden zwar unter einigen Einschränkungen möglich, birgt aber eine starke Abhängigkeit der Implementierung vom verwendeten ORB in sich[7]. Wir haben deshalb auf diese Möglichkeit verzichtet. Eine portierbare Abhilfe für das Problem schafft der in CORBA 3.0 definierte Portable Object Adapter [22].

Inheritance vs. Tie:
Für die Implementierung von CORBA-Objekten stehen grundsätzlich zwei Ansätze zur Verfügung. Bei dem *Inheritance-Ansatz* erben die implementierten Klassen direkt von einer vom IDL-Compiler erzeugten Basisklasse. Diese erbringt die notwendige Funktionalität zum Marshalling der Parameter sowie zum Aufruf der gewünschten Methode. Deren eigentliche Funktionalität wird von der erbenden Klasse implementiert. Da Java keine Möglichkeit zur Mehrfachvererbung bietet, kann bei diesem Ansatz die Implementation eines Interface der MapIDL keinerlei *Funktionalität* von einer anderen Klasse erben[8]. Der *Tie-Ansatz* umgeht dieses Dilemma durch die Einführung zusätzlicher Stellvertreterklassen. Diese implementieren ebenfalls Marshalling, werden aber zur eigentlichen Ausführung der Methoden mit einem Objekt einer anderen Klasse assoziiert. Beide Klassen müssen dieselbe Schnittstelle besitzen. Der Stellvertreter wird beim ORB angemeldet und erhält eine eigene IOR. Seine wesentliche Aufgabe besteht darin, eingehende Methodenaufrufe an das assoziierte Objekt weiterzuleiten und eventuelle Ergebnisse zurück zu liefern. Man erhält sich so die Möglichkeit zur Vererbung von Funktionalität auch ohne Mehrfachvererbung. Da in der MapIDL von Vererbung intensiv gebrauch gemacht wird, haben wir uns für den Tie-Ansatz entschieden.

Caching:
Für die Registrierung der Objekte mit dem ORB wird sowohl Zeit als auch Speicherplatz verbraucht. Da der Mapserver keinerlei Informationen darüber besitzt, wieviel Klienten noch Referenzen auf die von ihm verwalteten Objekte halten, können die einmal mit dem ORB registrierten Objekte nicht ohne weiteres abgemeldet und aus dem Speicher entfernt werden. Dies bedeuten einen *kontinuierlich steigenden Speicherverbrauch* des ORBs. Zusätzlich dazu kann der ORB auch nicht entscheiden, wann zwei von verschiedenen Anfragen aus der Datenbank geholte Objekte identisch sind - mit der Konsequenz, dass Datenbankobjekte mit der Zeit sogar mehrmals beim ORB registriert werden.

Da alle Objekte über Factories erzeugt werden, bietet es sich an, hier eine *Cache-Liste* für angemeldete Datenbankobjekte zu verwalten. Vor der Instanziierung eines Objektes wird geprüft, ob dieses Objekt bereits beim ORB registriert ist. Ist dies der Fall so wird dessen IOR als Ergebnis geliefert. Für die Identifizierung der CORBA-Objekte werden hierbei ein eindeutiger Schlüssel benötigt, was in IXDB aus einer Kombination von Relation und Primärschlüssel erreicht werden kann.

Nebenläufigkeit und Blocking:
ORBacus unterstützt eine Reihe von Nebenläufigkeitsmodellen [17]. Um für den Mapserver bei einer Vielzahl von Klienten die Blockierung der einzelnen Anwendungen weitestgehend zu vermeiden, benutzen wir „*thread per client*“. Jeder Klient kann also zu einem Zeitpunkt maximal eine Anforderung vom Mapserver abarbeiten lassen. Die Erzeugung der Threads übernimmt der ORB. Bei der Implementierung des Mapservers muss man aber darauf achten, dass alle Klassen bzw. ihre Methoden auch bei mehreren parallelen Threads korrekt arbeiten. Kritische Bereiche

[7] IONA’s Orbix bietet diese Möglichkeit auch, aber eben unter Verwendung anderer Methoden.
[8] Sie erbt sehr wohl das Interface, aber nicht die Implementierung.

müssen deshalb besonders geschützt werden, was in Java durch das Schlüsselwort *synchronize* gewährleistet wird.

Eine weitere Möglichkeit zur Vermeidung unnötiger Blockierungen und somit zur Steigerung der Performance ist die Verwendung mehrerer Datenbankverbindungen. Dies ist besonders dann interessant, wenn der Datenbankserver in der Lage ist, mehrere Anfragen parallel zu bearbeiten. Während der Ausführung einer SQL-Anfrage bleibt eine JDBC-Connection solange gesperrt, bis die Ergebnisse vorliegen. Bei der Verwendung einer einzigen Verbindung zur Datenbank werden während dessen alle weiteren Anfragen blockiert. Verwendet man dagegen eine weitere JDBC-Connection, so können zumindest zwei Anfragen nebeneinander abgearbeitet werden. Aufgrund dieser Überlegung haben wir einen *Verbindungspool* implementiert, mit dem bis zu N Klienten auf eigenen Verbindungen arbeiten können.

Struct vs. Interface:
Bei der Ableitung der konkreten MapIDL aus dem zugrunde liegenden Objektmodell kann man für die Repräsentation der einzelnen Klassen entweder IDL-Structs oder IDL-Interfaces verwenden. Dies muss natürlich im endgültigen Standard festgeschrieben werden, was aber zur Zeit unserer Implementation noch nicht geschehen war. Bei der Verwendung von IDL-Structs wird zu einem Klienten immer das komplette Objekt übertragen, d.h., Zugriffe auf die Attribute dieses Objektes erfordern keine zusätzliche Kommunikation über den ORB. Verwendet man dagegen IDL-Interfaces, so erhält der Klient lediglich eine Referenz auf das Objekt. Zugriffe auf dessen Attribute erzeugen jeweils einen Aufruf an den CORBA-Server. Damit führt die Verwendung von IDL-Struct zu einer Verringerung des Kommunikationsaufwands, was aber seinen Preis hat: Zum einen werden Klienten Daten verstärkt cachen, was zu ungültigen Werten führen kann. Zum anderen können IDL-Structs (und auch IDL-Interfaces) nicht von IDL-Structs erben[9]. Im Rahmen unserer Arbeit haben wir beide Alternativen implementiert und hinsichtlich ihrer Performance untersucht (siehe unten).

5.3 Semantische Probleme

Neben den technischen traten auch eine Reihe von semantischen Problemen auf, die im folgenden erörtert werden. Wir vermuten, dass diese „Mismatches“ typisch für die Anpassung von Standards an existierende Lösungen sind. Ihre Entdeckung zur Design-Zeit ist schwierig, weil insbesondere die Semantik der existierenden Applikationen oft nicht mehr ausreichend dokumentiert und nachvollziehbar ist - so auch im Fall von IXDB. Damit entstehen während der Umsetzung unter Umständen neue Anforderungen, die wiederum im Standard beachtet werden müssen. In diesem Sinne haben die von uns im folgenden beschriebenen Probleme bereits zu Änderungen des der OMG vorliegenden Standardisierungsvorschlags geführt.

Mehrwertige Attribute:
Da IXDB bereits selber eine integrierende Datenbank ist, kann es vorkommen, dass für Attribute mehrere Wert gespeichert sind. Dies ist z.B. häufig bei der Länge von Klonen und Genen der Fall. Die Unterschiede in den Längenangaben können ihren Ursprung sowohl in unterschiedlichen Meßverfahren als auch in fehlerhaften Daten haben. Mehrwertige Attribute können auch in „isolierten“ Datenbanken auftreten, z.B. wenn mehrere Messungen desselben Phänomens zu unterschiedlichen Ergebnissen führen - was in vielen molekularbiologischen Experimenten an der Tagesordnung ist.

---

[9] In CORBA 2.0. CORBA 3.0 spezifiziert eine Möglichkeit, dieses Problem zu umgehen (Object-by-Value). Implementationen liegen aber noch nicht vor.

Die MapIDL sieht jedoch für die Länge eines Objektes grundsätzlich nur einen einzelnen Wert vor. Gleiches gilt auch für die Namen der Objekte, so dass Synonyme nicht semantisch korrekt abgebildet werden können. Bei der Übertragung von Daten aus IXDB in Objekte der MapIDL muss man sich immer für einen der vorhandenen Werte entscheiden.

Abbildung der Datenbankobjekte auf MapIDL Interfaces:
Eine Zuordnung der in IXDB gespeicherten Objekte zu den in der Spezifikation enthaltenen Klassen ist nicht immer eindeutig möglich. In manchen Fällen besitzt IXDB eine „höhere Auflösung", beispielsweise werden verschiedene Marker-Typen unterschieden, in andern Fällen dagegen eine geringere Auflösung - auf den genauen Typ einer Karte kann man nur indirekt schließen. Außerdem gibt es in IXDB Klassen, deren Objekte in der MapIDL in verschiedenen Klassen erscheinen müßten, ohne dass die dazu notwendige Differenzierung möglich ist. Beispielsweise können IXDB-Objekte der Klasse `Probe` sowohl Punkte als auch Segmente sein[10]. Weitere Objekttypen wie `PCR-Product` finden gar keine Entsprechung in der MapIDL.

Ein weiteres Beispiel sind Gene. Ihnen können in IXDB über die Relation `DNADNA` Marker als Referenzpunkte zugeordnet sein. Diese Information kann besonders bei der Untersuchung und Diagnose von Erbkrankheiten von entscheidender Bedeutung sein. Um diese Information mit Hilfe der IDL-Spezifikation wiedergeben zu können, müssten Gene auf `Maps` abgebildet werden, was jedoch der eigentlichen Vorstellung von Genomkarten widerspricht.

Stringenz vs. Vollständigkeit:
Ein weiteres Problem im Zusammenhang mit den Genen besteht in der teilweisen Unvollständigkeit der in IXDB gespeicherten Informationen, die durch die stringente Handhabung der Relation zwischen Objekten und Genomkarten in der MapIDL nicht abgebildet werden kann. Gene sind im vorgeschlagenen Standard Segmente. Ihre Position auf einer linearen Genomkarte kann deshalb nur über `IntervalPosition` erfolgen. Für eine `IntervalPosition` muss aber sowohl ein linkes wie auch ein rechtes Ende angegeben werden. In IXDB dagegen ist es nicht notwendig, einem Gen auch eine rechte Position auf einer Karte zu geben, und in der Tat fehlt dieser Wert häufig. Um solche Zuordnungen in der MapIDL darstellen zu können, muss man einen geeigneten Wert für das Attribut `rightEnd` „erfinden", beispielsweise benutzt man einfach auch den Wert von `leftEnd`. Dies kann jedoch beim Betrachter zu falschen Interpretationen führen.

In Rahmen unserer Implementierung haben wir deshalb beschlossen, die entsprechenden Informationen ganz wegfallen zu lassen. Der Kern des Problems liegt in diesem Fall in der Vorschrift von Integritätsbedingungen durch den Standard, die in einem existierenden System nicht erfüllt sein müssen. Dagegen verstoßende Daten können dann nicht ohne weiteres exportiert werden.

6. ERFAHRUNGEN

Der von uns verwendete ORB zeigte sich über die gesamte Zeit der Entwicklung hinweg als sehr stabil. Auch die Registrierung einer Vielzahl von CORBA-Objekten im laufenden Betrieb des Mapservers hat nicht zu den zunächst erwarteten Fehlfunktionen bzw. Programmabstürzen geführt. Wir haben im Rahmen eines Testlaufes die implementierte Caching-Funktionalität des Mapservers ausgeschaltet, um durch wiederholtes Aufrufen von Methoden der IDL-Interfaces die

[10] Was eher an der unzureichenden Definition des Konzeptes „Probe" liegt als an einer Inkonsistenz in IXDB.

Anzahl der generierten Objekte solange wachsen zu lassen, bis der Mapserver überlastet ist. Es ist uns dabei nicht gelungen, den ORB zum Absturz zu bringen. Vielmehr wird der von der Java-Virtual-Machine benötigte Speicherbereich so groß, dass die komplette Workstation, auf der der Mapserver lief, nahezu zum Stillstand kam. Bei einer Anzahl von ca. 400.000 erzeugten Objekten mussten wir den Testlauf deshalb ohne den „erhofften“ Absturz abbrechen.

Im Rahmen von Benchmarks haben wir die Geschwindigkeit des Mapservers untersucht, insbesondere im Hinblick auf verschiedene Caching- und Prefetchingstrategien und auf die Möglichkeiten, entweder IDL-Structs oder IDL-Interfaces für bestimmte Klassen zu benutzen. Dazu sind noch weitere Untersuchungen notwendig. Generell kann die erreichte Geschwindigkeit nicht befriedigen, obwohl kein WAN zwischen Klient und Server lag. Auch die Verwendung von IDL-Structs brachte nicht den erhofften Geschwindigkeitsvorteil - statt dessen fällt die Performance teilweise sogar ab, wenn man auf eine Vielzahl kleiner Objekte zugreifen muss.

Die von uns verwendete MapIDL ließ noch eine Reihe von Methoden vermissen, die gerade für die anvisierten Anwendungen des Standards von Bedeutung seien könnten. Beispielsweise fehlt eine Methode zur Auflistung sämtlicher in der repräsentierten Datenbank vorkommenden Objekttypen, die ja von Datenbank zu Datenbank unterschiedlich seien können. Die IDL-Interfaces `Map` und `MapFactory` sollten außerdem die Möglichkeit bieten, Objekte nach Typen zu selektieren. Zur Visualisierung von Genomkarten wäre es auch sinnvoll, die Typen sämtlicher auf einer Genomkarten vorkommenden Objekte auflisten zu lassen. Des weiteren könnte es sinnvoll sein, dem Anwender die Möglichkeit zu geben, Angaben über die gewünschte Sortierung der Objekte bei Methoden mit mengenwertigem Ergebnistyp zu machen. Sortierungen können über SQL-Anfragen wesentlich effizienter ausgeführt werden als im Klient. Für das Speichermanagement im ORB sind `release()` Operationen unerläßlich, wie sie z.B. auch im „CORBA Object Life Cycle Service“ vorgesehen sind. Des weiteren fehlt noch die Möglichkeit, Laufzeitfehler in der Datenbank oder der Datenbankverbindung durch entsprechende Exceptions anzuzeigen.

Generell kommt der Erfolg eines Standards im wesentlichen - durch seinen Erfolg. Je mehr Server die MapIDL implementieren, desto attraktiver wird es für andere Server und Klienten, den Standard ebenfalls zu benutzen. Inwieweit dies für die MapIDL eintreten wird, ist noch nicht vorhersehbar. Die drei Partner der Submission (das European Bioinformatics Institute sowie die zwei kommerziellen Unternehmen Netgenics und Millenium Pharmaceticals) verpflichten sich zwar vor der OMG, eine Implementation durchzuführen, besitzen aber alleine vermutlich nicht genug Durchsetzungskraft, um eine weltweite Adaption zu erzwingen. Aufgrund unserer Erfahrungen können wir auf alle Fälle schließen, dass technische Probleme einer Adaption nicht im Wege stehen sollten.

DANKSAGUNG

Diese Arbeit wurde unterstützt durch die Deutsche Forschungsgemeinschaft über das Berlin-Brandenburger Graduiertenkolleg „Verteilte Informationssysteme“ (DFG grant no. GRK 316) sowie durch das Max-Planck-Institut für Molekulare Genetik in Berlin. Ein Teil der Forschungsarbeiten wurde außerdem durch ein Forschungsstipendium des European Molecular Biology Laboratory, grant no. ALTF ASTF, gefördert.

REFERENCES

[1] Barillot, E., F. Guyon, et al. (1998). HuGeMap: a distributed and integrated Human Genome Map database. Nucleic Acids Research 26(1): 106-107.

[2] Barillot, E., U. Leser, et al. (1999). A Proposal for a Standard CORBA Interface for Genome Maps. Bioinformatics 15(2): 157-169.

[3] Coupaye, T. (1999). Wrapping SRS with CORBA: from textual data to distributed objects. Bioinformatics 15(4): 333-338.

[4] Frenkel, K. A. (1991). The Human Genome Project and Informatics. Communications of the ACM 34(11): 40-51.

[5] Grigoriev, A. (1998). Reusable Graphical Interface to Genome Information Recources. 3rd Pacific Symposium on Biocomputing, Maui, Hawaii, pp. 130-141.

[6] Hu, J., C. Mungall, et al. (1998). Design and Implementation of a CORBA-based Genome-Mapping System Prototype. Bioinformatics 14(2): 112-120.

[7] InfoBiogen (1998). DBCat - The Public Catalog of Databases.Infobiogen, France . WWW Page . http://www.infobiogen.fr/services/dbcat.

[8] Jungfer, K. and P. Rodriguez-Tome (1998). MAPPLET: a CORBA-based Genome Map Viewer. Bioinformatics 14(8): 734-738.

[9] Kemp, G. J. L., C. J. Robertson, et al. (1999). Efficient Access to Biological Databases Using CORBA. British CCP11 Newsletter 3.1(7): available at: http://www.dl.ac.uk/CCP/CCP11/newsletter/vol13_11/index.html.

[10] Leser, U., H. Lehrach, et al. (1998). Issues in Developing Integrated Genomic Databases and Application to the Human X Chromosome. Bioinformatics 14(7): 583-690.

[11] Leser, U., H. Roest Crollius, et al. (1999). IXDB, an X Chromosome Integrated Database (update). Nucleic Acids Research 27(1): 123-127.

[12] Leser, U., S. Tai, et al. (1998). Design Issues of Database Access in a CORBA Environment. Workshop on Integration of Heterogeneous Software Systems, Magdeburg, Germany, pp. 74-87.

[13] Letovsky, S., R. W. Cottingham, et al. (1998). GDB: the Human Genome Database. Nucleic Acids Research 26(1): 94-99.

[14] Markowitz, V. M., I. A. Chen, et al. (1997). Facilities for Exploring Molecular Biology Databases on the Web: A Comparative Study. 2nd Pacific Symposium on Biocomputing, Santa Fe, New Mexico.

[15] Markowitz, V. M. and O. Ritter (1995). There's never time to do it right, but always time to do it over. 2nd Meeting on Interconnection of Molecular Biology Databases, Cambridge, U.K..

[16] Mueller, H. (1999). Realisierung eines einheitlichen Zugriffs auf molekularbiologische Genomkarten unter Verwendung von CORBA. Technische Universitaet Berlin. Diplomarbeit.

[17] Object Oriented Concepts Inc (1998). ORBacus 3.0. Users manual 1.0 . http://www.ooc.com.

[18] Primrose, S. B. (1996). Genomanalyse. Heidelberg, Spectrum; Akademischer Verlag.

[19] Robbins, R. J. (1992). Challenges in the Human Genome Project: Progress Hinges on Resolving Database and Computational Factors. IEEE Engineering in Medicine and Biology March 1992: 25-34.

[20] Robbins, R. J. (1994). Representing Genomic Maps in a Relational Database. In Computational Methods in Genome Research. S. Suhai (ed.), New York, Plenum Press: 85-96.

[21] Rodriguez-Tome, P., C. Helgesen, et al. (1997). A CORBA Server for the Radiation Hybrid Database. 5th Int. Conf. on Intelligent Systems for Molecular Biology, Halkidiki, Greece, AIII Press, Menlo Park, pp. 250-253.
[22] Schmitt, D. E. and S. Vinoski (1997). Object Adapters: Concepts and Terminology. SIGS C++ Report 9(11).
[23] Stoesser, G., M. A. Moseley, et al. (1998). The EMBL Nucleotide Sequence Database. Nucleic Acids Research 26(1): 8-15.
[24] Weissenbach, J. and S. Bentolila (1996). Integrated maps require integrated data. Nature Biotechnology 14: 678.
[25] Widom, J. (1995). Research Problems in Data Warehousing. 4th International Conference on Information and Knowledge Management, Baltimore, Maryland, pp. 25-30.