Georg-August-Universität
Göttingen
Zentrum für Informatik

ISSN 1612-6793
Nummer ZFI-BM-2004-04

Bachelorarbeit

im Studiengang "Angewandte Informatik"

Evaluierung von XML-Datenbanksystemen für Anwendungen in der Bioinformatik

Remko Ricanek

am Institut für

Informatik und an der Medizinischen Fakultät, Abteilung
Bioinformatik der Georg-August-Universität Göttingen

Bachelor- und Masterarbeiten
des Zentrums für Informatik
an der Georg-August-Universität Göttingen

14. April 2004

Georg-August-Universität Göttingen
Zentrum für Informatik

Lotzestraße 16-18
37083 Göttingen
Germany

Tel. +49 (5 51) 39-1 44 02
Fax +49 (5 51) 39-1 44 03
Email office@informatik.uni-goettingen.de
WWW www.informatik.uni-goettingen.de

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Göttingen, den 14. April 2004

Bachelorarbeit

Evaluierung von XML-Datenbanksystemen für Anwendungen in der Bioinformatik

Remko Ricanek

14. April 2004

Betreut durch
Prof. Dr. Edgar Wingender und Dipl.-Inform. Martin Haubrock
Medizinische Fakultät, Abteilung Bioinformatik
Georg-August-Universität Göttingen

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Danksagung

An erster Stelle gilt mein besonderer Dank Herrn Prof. Dr. Edgar Wingender, der diese Arbeit ermöglicht hat. Die Aufnahme in der Arbeitsgruppe war sehr herzlich und bei auftretenden Problemen wurde gern geholfen. Ganz besonders hervorzuheben ist weiterhin mein Betreuer Martin Haubrock, der stets Zeit für Gespräche über den theoretischen Hintergrund und für praktische Hinweise fand. Darüber hinaus danke ich meinem Zweitkorrektor Herrn Prof. Dr. Wolfgang May und besonders Erik Behrends für die intensive Zusammenarbeit an der *Databases and Information Systems Group* (DBIS) am Institut für Informatik der Georg-August-Universität Göttingen. Ganz herzlich möchte ich mich auch bei Daniel Fredrich und Kristin Walther für das stetige Korrekturlesen und für die zahlreichen Tipps und Anregungen bedanken. Während meiner Arbeit habe ich ganz besonders die gute Zusammenarbeit und die nette Arbeitsatmosphäre mit Daniel genossen. Weiterhin bedanke ich mich bei meinen Korrekturlesern Nele Heppeler und Benjamin Zeiß. Zuletzt möchte ich mich ganz persönlich bei meinen Eltern bedanken, die mir während meiner Arbeit stets Halt und Unterstützung gaben.

1 Zusammenfassung

Datenbanken bilden eine wichtige Grundlage für die biologische Forschung und sind zum Beispiel im Bereich der Genomforschung zu einem unentbehrlichen Werkzeug geworden. Eine stetig steigende Menge an biologischen Daten hat die Bioinformatik in Bereichen der Gentechnologie und biomedizinischen Forschung unverzichtbar gemacht. Die Microarray-Technologie hat dabei besonders an Bedeutung zugenommen und bildet eine wichtige Grundlage für Experimente zur Funktionsaufklärung von Genen.

In der vorliegenden Arbeit wurden die Datenbanksysteme *Microsoft SQL Server 2000* und *Oracle 9i Release 2* unter dem Aspekt der Speicherung und Verwaltung von standardisierten Microarray-Daten evaluiert. Diesbezüglich wurde ein spezifischer Kriterienkatalog zusammengestellt, der, als Bewertungsgrundlage fungierend, Vor- und Nachteile der einzelnen Datenbanksysteme für die spezielle Aufgabenstellung herausarbeitet. Untersucht wurden Kriterien wie Plattformunterstützung, Programmiersprachen, XML-Schema-Unterstützung, Schema-basierter Im- und Export von Daten und Dokumentation.

Die vorliegende Arbeit befasst sich also mit der Evaluierung von Datenbanksystemen zur Speicherung und Verwaltung biologischer Daten. Solche Datenbanksysteme müssen aufgrund der besonderen Form dieser Daten gewisse Voraussetzungen erfüllen. Das zweite Kapitel arbeitet diese Voraussetzungen heraus, indem die biologischen Grundlagen und vor allem die Methoden der Microarray-Technologie vorgestellt werden. Im Bereich der Gentechnologie und biomedizinischen Forschung ist der Vergleich von biologischen Daten aus Microarray-Experimenten ohne entsprechende Standards kaum denkbar. Diese Standards werden vorgestellt und daran anschließend ein Kriterienkatalog zur Bewertung erstellt.

Das dritte Kapitel betrachtet die Möglichkeiten der Speicherung biologischer Daten auf Basis von XML-Dokumenten. Hierfür werden die zwei Datenbanksysteme ausführlich vorgestellt und getestet. Die Daten werden im Datenbanksystem abgelegt und anschließend die Anfragemöglichkeiten ausgearbeitet.

Der vierte Teil dieser Arbeit beinhaltet eine Beurteilung der vorgestellten Datenbanksysteme anhand des definierten Kriterienkatalogs. Den Abschluss des Kapitels bildet ein zusammenfassender Vergleich der beiden Datenbanksysteme. Hierbei konnte die Datenbanksoftware von Oracle aufgrund ihrer Leistungsfähigkeit als besonders geeignet bewertet werden.

Das Fazit in Kapitel 5 fasst die einzelnen Abschnitte der Arbeit noch einmal zusammen. Zusätzlich beinhaltet es einen Ausblick auf die Umsetzung der vorgestellten Anwendungsbeispiele auf bioinformatische Anwendungen.

2 Einleitung

Die Fortschritte der letzten Jahrzehnte in der Gen- und Biotechnologie haben zu einem Wandel in der biomedizinischen Forschung geführt. Die Auswertung von genetischen Informationen und die damit verbundenen Hochdurchsatz-Methoden sind zu einem wichtigen Teil der Forschungsarbeit geworden und haben die Datenverarbeitung und somit die Bioinformatik im Bereich der Genomforschung in den Mittelpunkt des Interesses gerückt.

Nach der Sequenzierung der Genome einer Reihe von Organismen bis hin zum Menschen gilt es nun Abschnitte der DNA zu finden, die Baupläne für Proteine enthalten, um schließlich die Zusammenhänge zwischen genetischer Information, deren Relevanz und ihrer Umsetzung in mRNA und Proteine zu verstehen. Obwohl die genomische Information in jeder Zelle eines Organismus identisch ist, variiert die Aktivität der Gene (Genexpression) in Abhängigkeit vom Zelltyp wesentlich. Ein noch junger Ansatz zur Funktionsaufklärung ist die Verwendung von DNA-Chips (*Microarrays*), mit deren Hilfe die Genexpression in Abhängigkeit von externen und internen Bedingungen analysiert werden kann. Hierbei wird die Gesamtheit der zellulären mRNA (Transkriptom) zugrundegelegt.

2.1 Microarrays

Es gibt verschiedene Arten von DNA-Microarrays. Das grundlegende Prinzip der DNA-Microarray-Technologie basiert im Allgemeinen auf der Fixierung möglichst vieler Gensequenzen auf einem Glas- oder Kunststoffträger. Diese Sequenzen dienen als Sonden für die Hybridisierung mit cDNAs, die das Transkriptom einer Zelle oder eines Gewebes repräsentieren. cDNA (*komplementäre* DNA) wird *in vitro* mit Hilfe einer reversen Transkriptase aus mRNAs synthetisiert und dabei gleichzeitig mit Fluoreszenz-Farbstoffen markiert.

Das *Immobilisieren* (Fixieren) der Gensequenzen auf den Objektträgern erfolgt durch zwei verschiedene Methoden. Die erste Methode ist das gezielte Synthetisieren von DNA-Oligonukleotiden aus 20-30 Nukleotiden, die das Spektrum der möglichen mRNAs eines Organismus abdecken. Solche *GeneChips* werden kommerziell von der Firma *Affymetrix* [Aff04] vertrieben. Bei der zweiten Methode werden cDNA-Proben mit einer Länge von 100 bis 1000 Nukleotiden auf einen Träger maschinell rasterförmig aufgetragen. Auf diese Weise lassen sich auf wenigen Quadratzentimetern eines Glas-Objektträgers einige tausend Spots unterbringen. Bei dieser Methode spricht man von so genannten *cDNA-Arrays* oder *Microarrays*, deren frei zugängliche Technologie an der *University of Stanford* [Sta04] entwickelt wurde [Kni01].

Microarrays werden für die Erforschung von Krankheitsbildern und Identifizierung von "Krankheitsgenen", sowie für die Suche nach neuen Medikamenten in der Medizin verwendet. Ziel ist es, ein Gesamtbild der Genexpression in Zellen oder Geweben zu gewinnen, um somit Veränderungen in der Expression unter verschiedenen Umweltbedingungen analysieren und interpretieren zu können.

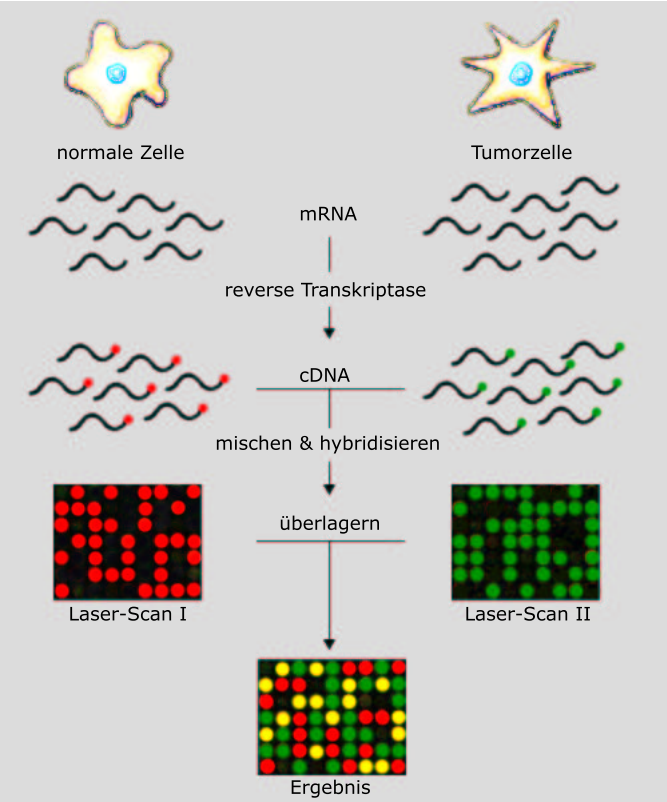

Abbildung 2.1: *Ein typisches Microarray-Experiment*

Es wird angenommen, dass eine erhöhte Expression von mRNA in der Regel mit einer erhöhten Proteinkonzentration korreliert, aber zusätzlich posttranslationalen Modifikationen unterliegt. Bei einem Microarray-Experiment wird nur die Expression von mRNA erfasst. Zur Analyse des gesamten Proteoms sind jedoch weitere Untersuchungen nötig.

Bei einem typischen Microarray-Experiment werden zunächst Proben zum Beispiel aus zwei verschiedenen Geweben oder aus Zellen in verschiedenen Zuständen entnommen und deren mRNA isoliert. Anschließend wird von diesen Proben Fluoreszenz-markierte cDNA hergestellt. Die so gewonnenen cDNAs werden gemischt und mit den immobilisierten DNA-Proben auf dem Microarray hybridisiert. Mit einem anschließenden *Laser-Scan* werden die unterschiedlichen Wellenlängen der Fluoreszenzen angeregt und durch entsprechend empfindliche Geräte gemessen. Durch das Übereinanderlagern dieser beiden Farbmuster können Unterschiede der Genexpression sichtbar gemacht werden. So kann zum Beispiel eine erhöhte oder verringerte Expression eines Gens einer Tumorzelle analysiert werden. Eine schematische Übersicht eines typischen Microarray-Experiments ist in Abbildung 2.1 dargestellt.

DNA-Chips und auch Microarrays tragen viele tausend Gene bis hin zum gesamten Genom eines Organismus, soweit sie bekannt sind. Beim menschlichen Genom werden beispielsweise 30.000 bis 40.000 Gene in einem einzigen Experiment untersucht. Die Experimente müssen mehrmals wiederholt werden, um eine statistisch fundierte Aussage über die Genregulation machen zu können. Daraus folgt, dass diese Experimente eine große Menge an Daten liefern (*Highthroughput-Verfahren*), die nur mit Hilfe von Computern bewältigt und ausgewertet werden können. Die so gelieferten In-

formationen gilt es zu zentralisieren und zu standardisieren. Ein öffentlicher Zugang zu diesen Daten durch entsprechende Datenbanken ermöglicht auch die Kreuz-Validierung von Daten verschiedener Organismen [Kni01].

2.2 Datenstandards

2.2.1 XML

XML (*eXtensible Markup Language*) ist eine Metasprache zur Definition von Dokumentklassen. Eine XML-Repräsentation kann aus einfachem ASCII-Text bestehen, die, ähnlich wie bei HTML, durch umschließende *Tags* strukturiert wird. XML-Dokumente beschreiben eine Klasse von Datenobjekten, welche eine Teilmenge von SGML (*Standard Generalized Markup Language*) darstellen. Durch ihre Konstruktion sind XML-Dokumente konforme SGML-Dokumente. XML-Dokumente sind im Wesentlichen aus Elementen aufgebaut, die Attribute und Subelemente haben können. Unter dem Begriff *Markup* versteht man eine Beschreibung der logischen Struktur des Dokuments. Ein XML-Dokument ist nach der hierarchischen Struktur aufgebaut. Die Struktur kann durch eine *Document Type Definition* (DTD) oder durch das vom *World Wide Web Consortium* (W3C) verabschiedete *XML-Schema* [Rec01] definiert sein.

Die Entwurfsziele für XML waren:

- XML soll sich im Internet auf einfache Weise nutzen lassen;
- XML soll ein breites Spektrum von Anwendungen unterstützen;
- XML soll zu SGML kompatibel sein;
- es soll einfach sein, Programme zu schreiben, die XML-Dokumente verarbeiten;
- XML-Dokumente sollten für Menschen lesbar und angemessen verständlich sein;
- XML-Dokumente sollen leicht zu erstellen sein.

Eine gesamte Übersicht der XML-Spezifikation ist in [Rec04] zu finden.

XML-Dokumente sind durch ihre Struktur sehr gut zum Austausch großer Datenmengen geeignet. Für Anfragen an diese XML-Dokumente und zur Adressierung von Teilen eines XML-Dokuments wurde vom W3C die XML-Anfragesprache *XPath 1.0* [Rec99] im Jahre 1999 verabschiedet. Als Nachfolger und zukünftiger Standard für XML-Anfragesprachen wird *XQuery* [Dra03] vom W3C vorgesehen, dessen Status jedoch bisher noch als *Working Draft* eingestuft ist, sich also noch in der Entwicklung befindet. Diese XQuery-Spezifikation wird *XPath 2.0* beinhalten [Dra04]. Ziel ist es, XQuery als Standardanfragesprache für XML-Daten zu etablieren, so wie SQL (*Structured Query Language*) die Standardanfragesprache für relationale Daten ist.

2.2.2 MIAME

Die *Microarray Gene Expression Data Society* (MGED Society) ist eine internationale Organisation von Biologen, Informatikern und Datenanalytikern, welche sich zum Ziel gesetzt hat, den Austausch von Microarray-Daten zu erleichtern. Sie wurde von Alvis Brazma und Alan Robinson am *European Bioinformatics Institute* (EBI) im Juni 2002 gegründet, um Standards für Genexpressions-Datenbanken zu entwickeln [JCB02]. Um das ganze Potential von Microarrays ausschöpfen zu können, müssen die Microarray-Experimente konsistent beschrieben werden, so dass ein Austausch und eine Vergleichbarkeit der Daten möglich ist.

Der *Minimum Information About a Microarray Experiment* (MIAME) Standard ist ein solcher Standard der MGED Society und definiert die minimalen Anforderungen an die Dokumentation eines Microarray-Experiments, die zum Austausch von Microarray-Daten einzuhalten sind. Das Ziel von MIAME ist es, diejenigen Informationen zu beschreiben, die zur Verfügung gestellt werden sollten, um die experimentellen Methoden und den biologischen Hintergrund von Microarray-Daten ausführlich und ausreichend zu erklären. Mit Hilfe dieses Standards ist es zum Beispiel möglich, Experimente zu reproduzieren [JCB02][BHQ+01].

2.2.3 MAGE

Ein weiterer Standard der MGED Society ist der *MicroArray and Gene Expression* (MAGE) Standard, welcher das zu Microarray-Daten gehörige Datenaustausch-Modell, das *Object Model* (MAGE-OM) und ein allgemeines Datenaustausch-Format, die *Markup Language* (MAGE-ML) definiert. MAGE-ML ermöglicht einen Austausch von Microarray-Informationen zwischen verschiedenen Systemen auf Basis von XML. MAGE-OM basiert auf der Standardsprache zur Beschreibung von Objektmodellen, der *Unified Modelling Language* (UML) [JCB02].

MAGE-ML und MAGE-OM definieren zusammen mit MIAME die Beschreibung eines Microarray-Experiments. Die Repräsentation in MAGE-OM ist jedoch sehr umfangreich. Daher wird dieses Modell in so genannte Pakete unterteilt, wobei verwandte Klassen in diesen Paketen gruppiert sind. Eine Übersicht dieser Pakete ist im Anhang A zu finden.

Das Ziel der MGED Society ist es, den MAGE Standard als *den* Microarray Standard zu etablieren. Durch standardisierte Microarray-Daten ist man in der Lage, Microarray-Experimente zu vergleichen. Gleichzeitig bilden sie eine gute Grundlage für Publikationen [JCB02][BHQ+01].

2.3 Web Repositories für Microarray-Daten

Die durch GeneChips oder Microarrays gewonnenen Daten werden meist in öffentlich zugänglichen *Repositories* abgelegt. Eine Übersicht der verfügbaren Repositories zeigt Tabelle 2.1. Die drei bekanntesten Repositories werden im Folgenden kurz beschrieben.

Repository	Kommentar	URL
AMAD	Stanford und University of California	http://www.microarrays.org/software.html
ArrayExpress	Alvis Brazma am EBI	http://www.ebi.ac.uk/arrayexpress/
ChipDB	Whitehead Institute	http://young39.wi.mit.edu/chipdb_public/
ExpressDB	Harvard	http://arep.med.harvard.edu/ExpressDB
Gene Director	Biodiscovery	http://www.biodiscovery.com
GeNet	Silicon Genetics	http://www.sigenetics.com
GeneX	NCGR	http://genex.ncgr.org/
GEO	Gene Expression Omnibus am NCBI	http://www.ncbi.nlm.nih.gov/geo/
GXD	Jackson Laboratory	http://www.informatics.jax.org/
MAdb	National Cancer Institute	http://madb.nci.nih.gov
MaxdSQL	University of Manchester	http://www.bioinf.man.ac.uk/microarray/maxd/
RAD	University of Pennsylvania	http://cbil.upenn.edu/rad2/servlet
Stanford Microarray Database	Stanford University	http://www.dnachip.org/

Tabelle 2.1: *Web Repositories für Microarray-Daten [Pev03]*

2.3.1 Gene Expression Omnibus

Das *Gene Expression Omnibus* (GEO) Repository am *National Center for Biotechnology Information* (NCBI) [GEO04] ist das erste öffentliche Repository für ein breites Spektum von Hochdurchsatz-Genexpressionsdaten. Diese Daten beinhalten sowohl Microarray-basierte Experimente für mRNA-, DNA- und Proteinkonzentrationsmessungen, als auch Experimente der seriellen Analyse der Genexpression (SAGE), sowie Proteom-Experimente der Massenspektrometrie.

2.3.2 Stanford Microarray Database

Die *Stanford Microarray Database* (SMD) der *University of Stanford* [SMD04] speichert Rohdaten und normalisierte Daten aus Microarray-Experimenten und Publikationen, die der Öffentlichkeit zur Verfügung gestellt werden. So sind mehr als 3500 Zweifarben-Microarrays frei zugänglich. Diese öffentlichen Daten beinhalten Experimente von zwölf spezifischen Organismen, wie *Homo sapiens*, *Caenorhabditis elegans*, *Arabidopsis thaliana*, *Saccharomyces cerevisiae*, *Drosophila melanogaster*, *Escherichia coli* u.a.. Mit Hilfe von web-basierten Programmen kann man Anfragen an die Datenbank stellen, bestimmte Datenqualitäten festlegen oder anhand von individuellen Charakteristika die experimentellen Daten filtern. Diese Daten werden zudem über Crosslinks mit anderen Datenbanken wie *SWISS-PROT* oder *UniGene* verbunden [GBB+02].

Sowohl GEO, als auch SMD implementieren ihre Daten auf Basis des *MIAME* Standards der MGED Society (siehe Abschnitt 2.2.2).

2.3.3 ArrayExpress

ArrayExpress [AE04] ist ein Public Repository für Microarray-Genexpressionsdaten am EBI, das in Übereinstimmung mit Standards der MGED Society annotierte Daten speichert und für die Öffentlichkeit zugänglich macht. ArrayExpress unterstützt den MIAME und MAGE-ML Standard der MGED Society in vollem Umfang. Die *Microarray Informatics Group* am EBI hat diese Standards mitentwickelt und befasst sich mit den Problemen des Handhabens, der Speicherung und des Analysierens dieser Microarray-Daten.

ArrayExpress soll:

- der wissenschaftlichen Gemeinschaft als Repository für Microarray-Daten dienen, die für Publikationen verwendet werden können,
- einen einfachen Zugriff auf qualitativ hochwertige Genexpressionsdaten in standardisierter Form bieten,
- den Austausch von Microarray-Designs und experimentellen Protokollen erleichtern [AEF04].

ArrayExpress hat vier Hauptkomponenten:

- die Datenbank für Microarray-Daten, in der alle Microarray-Daten zentral abgelegt werden;
- ein web-basiertes Interface für Anfragen an die Microarray-Datenbank;
- das Programm *MIAMExpress*, welches Microarray-Daten web-basiert übermittelt und annotiert;
- das *Expression Profiler* Programm zur Online-Datenanalyse.

Die Microarray-Daten können zur Analyse entweder direkt über das *Expression Profiler* Programm importiert oder zur lokalen Analyse exportiert werden. Eine Übersicht der Komponenten zeigt Abbildung 2.2 [AEF04].

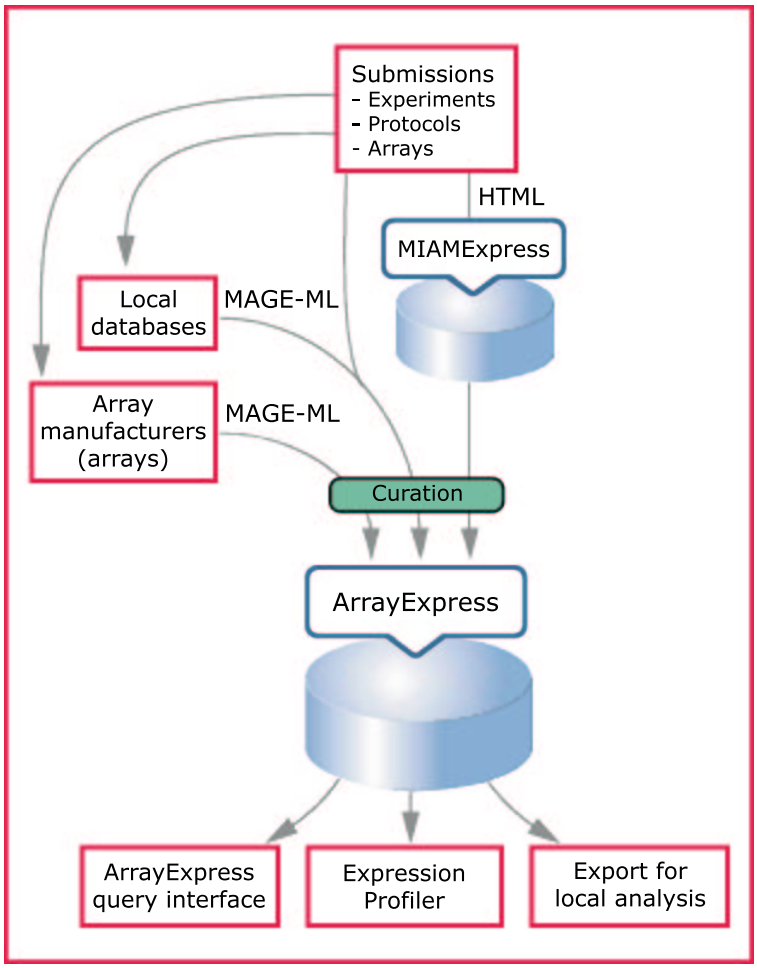

Abbildung 2.2: *Die Hauptkomponenten von ArrayExpress [AEF04]*

2.4 XML und Microarray-Daten

2.4.1 Motivation

XML als Datenaustausch-Format hat mittlerweile große Verbreitung erlangt, eignet sich also bestmöglich zum Austausch von Microarray-Daten. Diese Microarray-Daten werden im XML-Format nach dem MAGE Standard gespeichert, wobei eine *DTD* oder ein *XML-Schema* den strukturellen Aufbau und die Dokumenttypen dieser Daten beschreibt. Hierbei setzt sich die Struktur aus den einzelnen Paketen des MAGE-OM Standards, wie im Angang A beschrieben wird, zusammen.

Ein Forschungsschwerpunkt der Abteilung Bioinformatik an der Medizinischen Fakultät der Georg-August-Universität Göttingen ist unter Anderem die Analyse und Interpretation von Expressionsdaten aus Microarray-Experimenten. Ziel ist es, eine Plattform einzurichten, die qualitativ hochwertige Microarray-Genexpressionsdaten zur Analyse und Interpretation bereitstellt. Eine Auswahl dieser Daten soll ebenfalls in einem lokalen Repository abgelegt werden. Als Quelle für öffentliche Microarray-Daten werden Daten der ArrayExpress Datenbank des EBI bevorzugt, da diese in vollem Umfang dem MAGE Standard entsprechen. Außerdem soll es ohne Probleme möglich sein, auch andere Datenquellen hinzuziehen zu können. Damit die stetig steigende Menge von Microarray-Daten effizient verwaltet werden kann, bietet es sich an, diese Daten in einem eigenen *Datenbanksystem* (DBS) abzulegen. Neben den allgemeinen Anforderungen an ein DBS, wie transaktionsorientierte Verarbeitung, Datensicherheit und effiziente Anfragemöglichkeiten, gelten für die XML-Speicherung spezielle Anforderungen. Der folgende Abschnitt befasst sich mit diesen Kriterien.

2.4.2 Kriterienkatalog

Für die abteilungsinterne Arbeit mit Microarray-Daten ist es wichtig, dass das zu verwendende Datenbanksystem gewisse Anforderungen erfüllt und bestimmte Funktionen bietet. Folgende Punkte sind hier besonders zu beachten:

- Plattformunterstützung
- Programmiersprachen
- XML-Schema-Unterstützung
- Schema-basierter Im- und Export von Daten
- Dokumentation und Support

Diese Punkte werden im Folgenden kurz erläutert.

Plattformunterstützung

Für den Einsatz einer Datenbank ist die *Plattformunterstützung* von großer Bedeutung. Für den Betrieb einer Datenbank muss dem Anwender bekannt sein, auf welchem Betriebssystem diese Datenbank installiert werden kann, da oftmals nicht beliebige Plattformen unterstützt werden. Darüber hinaus unterstützen die mit der Datenbank verbundenen Anwenderprogramme ebenfalls

nicht alle Betriebssysteme. Um einen plattformunabhängigen Zugriff auf die Datenbank und entsprechende Datenverwaltung gewährleisten zu können, sollte ein Anwender, von einem beliebigen Betriebssystem ausgehend, auf die Datenbank zugreifen können.

Programmiersprachen

Neben der Plattformunterstützung ist es wichtig, für welche Programmiersprachen dieser Zugriff unterstützt wird. Die Programmiersprache Java ist eine der am meisten verwendeten Programmiersprachen und findet so gut wie auf jedem System Anwendung. Daher bietet es sich an dieser Stelle an, eine Unterstützung für Java zu fordern, so dass aus einem Java-Programm die Verwaltung von XML-Daten und in vollem Umfang XPath-Anfragen an die Datenbank möglich sind. Darüber hinaus sollte der Programmieraufwand minimal sein, so dass es möglich ist, Daten ohne vorheriges Parsen etc. in die Datenbank zu importieren.

XML-Schema-Unterstützung

Microarray-Daten werden in einem standardisierten XML-Format gespeichert, damit eine Portabilität und allgemeine Vergleichbarkeit gewährleistet ist. Für den Umgang mit Microarray-Daten ist es also nicht nur wichtig, dass die Datenbank XML unterstützt, sondern dieses auch auf Basis einer DTD oder eines XML-Schemas möglich macht, so dass bei einem Im- und Export der Daten die definierten Standards eingehalten werden.

Schema-basierter Im- und Export von Daten

Die Unterstützung einer DTD oder eines XML-Schemas beim Im- und Export von XML-Daten ist von großer Wichtigkeit, da die Microarray-Daten als standardisiertes MAGE-XML vorliegen. Wenn man die Daten exportiert, kann man XPath-Anfragen an die Datenbank stellen, die zum Beispiel ein bestimmtes Teilergebnis oder Bedingungen eines Experiments liefern. Diese exportierten Daten müssen wiederum dem Schema entsprechen, damit eine Vergleichbarkeit der Daten weiterhin gewährleistet ist.

Dokumentation und Support

Ein Datenbanksystem sollte sich außerdem durch eine gute Dokumentation auszeichnen. Darunter fallen u.a. Bücher, Online-Tutorials, sowie Artikel aus Magazinen und Zeitungen. Insbesondere die Einführung in die Funktionen einer Datenbank wird durch eine hohe Qualität solcher Dokumentationen stark erleichtert. Darüber hinaus sollte ausreichend Support über Mailinglisten oder Foren angeboten werden.

3 Analyse

Als Anhaltspunkt für die Analyse verschiedener Datenbanken auf ihre XML-Fähigkeit dient der in 2.4.2 definierte Kriterienkatalog. Da die Microarray- und Genexpressionsdaten im XML-Format vorliegen und dem MAGE Standard entsprechen, ist es wichtig, die Daten sowohl beim Importieren, als auch beim Exportieren ohne Datenverlust zu verwalten. Durch den Datenexport sollen hinsichtlich der weiteren Vergleichbarkeit ebenfalls XML-Daten generiert werden, wobei wiederum der MAGE Standard einzuhalten ist. Die Datenbanken sollten also sowohl XML problemlos bearbeiten, als auch die Konformität bezüglich der XML-Definition (DTD oder XML-Schema) gewährleisten. In der Praxis sollte es dann so aussehen, dass die XML-Daten anhand einer DTD oder eines XML-Schemas ins relationale Modell des DBS übertragen werden. Ohne explizites Wissen über die Tabellenstruktur können die Daten komplett oder teilweise mittels der XML-Anfragesprache *XPath* [Rec99] auf Basis des Schemas wieder als XML-Dokument exportiert werden.

Im Rahmen dieser Arbeit wird hauptsächlich auf die Datenbank *Microsoft SQL Server 2000* der Microsoft Corporation und die Datenbank *Oracle 9i Release 2* der Oracle Corporation eingegangen. Open Source Datenbanken zeigen auch Bemühungen in Richtung XML-Unterstützung, bieten aber noch nicht den Umfang, wie es der *SQL Server 2000* und *Oracle 9i* versprechen.

3.1 Microsoft SQL Server

Da in der Abteilung Bioinformatik für interne Datenverwaltungen bereits der Microsoft SQL Server 2000 in Betrieb ist, wurde mit ersten Tests auf diesem DBS begonnen. Der Microsoft SQL Server bietet für das Abrufen und Speichern von XML-Daten zwei integrierte Methoden. Zum Einen die *FOR XML*-Klausel als Erweiterung der *SELECT*-Anweisung, welche SQL-Anfragen als XML-Dokumente anstelle von *Standardrowsets* zurückgibt, zum Anderen die SQL-Funktion *OPENXML*, welche als XML-Dokument dargestellte Daten in relationale Tabellen einfügen kann. Beide Methoden sind jedoch für die Anforderungen aus 2.4.2 ungeeignet, da sie keine Schema-Unterstützung bieten. So setzen zum Beispiel die FOR XML-Klausel und OPENXML-Funktion ein bestehendes relationales Datenmodell voraus. Außerdem bieten beide Methoden keine Möglichkeit, die Daten mit Hilfe eines XML-Schemas zu validieren. Für den spezielleren Umgang mit XML-Daten sind also umfangreiche Vorbereitungen nötig.

3.1.1 Vorbereitung

Zu Testzwecken wurde ein PC zur Verfügung gestellt, auf dem das Betriebssystem Microsoft Windows 2000 Server und der Microsoft SQL Server 2000 installiert wurden. Für die XML-Fähigkeit des SQL Servers wurden außerdem folgende Updates installiert:

- Das *SQLXML 3.0* Update mit aktuellem Service Pack. "SQLXML bietet XML-Unterstützung für SQL Server-Datenbanken. Es ermöglicht Entwicklern, die Lücke zwischen XML und relationalen Daten zu überbrücken" [Cor03c].
- Die *Microsoft Data Access Components* (MDAC) in ihrer aktuellen Version 2.8. Die Microsoft Data Access Components enthalten u.a. Treiber für diverse Datenquellen [Cor03a].
- Für einen Zugriff auf den SQL Server über die Anfragesprache XPath wird der *Microsoft Internet Information Server* (IIS), der im Betriebssystem Windows 2000 Server enthalten ist, benötigt. Über diesen IIS kann mittels HTTP auf den SQL Server zugegriffen werden.

Der Zugriff auf den Microsoft SQL Server geschieht generell über eine Client-Server Verbindung. Der Client ist in der Regel ein weiterer Arbeitsrechner, in diesem Fall ein PC mit laufendem Windows Betriebssystem. Dementsprechend ist also die beim SQL Server mitgelieferte Client-Software zu installieren und darüber hinaus ebenfalls die SQLXML und MDAC Updates.

3.1.2 Implementation

Nach dem in Abschnitt 2.4.2 definierten Kriterienkatalog wird eine Java-Anbindung zum SQL Server angestrebt. Über einen *Java DataBase Connectivity* (JDBC) Treiber kann die Verbindung zum SQL Server hergestellt werden. Somit sind Anfragen an den SQL Server aus einem Java-Programm formulierbar. Dies ist aber nur eingeschränkt möglich, da der SQL Server für den Import von XML-Daten nur einen so genannten *Bulk Load* über die SQLXML-Schnittstellen vorsieht. XML Bulk Load importiert XML-Daten anhand eines annotierten *eXternal Data Representation* Schemas (XDR Schema), ein XML-Schema, das um Microsoft-Spezifikationen erweitert wurde. Diese zusätzlichen Informationen spezifizieren die Abbildungen von Elementen und Attributen eines XML-Dokuments in Relationen. Für den SQLXML Bulk Load müssen also entsprechende Objekte geladen werden. Das Laden dieser Objekte aus einem Java-Programm heraus ist von Microsoft jedoch nicht vorgesehen. Stattdessen bietet Microsoft mit den Programmiersprachen *Visual C#* oder *Visual Basic* solche Möglichkeiten. Im Folgenden wird nur auf Beispiele mit Visual Basic eingegangen.

Die Implementierung des SQLXML Bulk Load mit einem kleinen Visual Basic Script zeigt Listing 3.1. Zuerst wird das *SQLXMLBulkLoad* Objekt geladen und über den *SQLOLEDB Provider*, einer Schnittstelle der Microsoft Data Access Components, eine Verbindung zum SQL Server hergestellt. Anschließend wird ein XML-Dokument mit entsprechendem XDR-Schema [Anhang B.1] dem SQL Server übergeben. Der SQL Server legt anhand der XDR-Annotationen die Relationen an und speichert die XML-Daten dort entsprechend ab.

```
1  Dim objBL
2  Set objBL = CreateObject("SQLXMLBulkLoad.SQLXMLBulkLoad.3.0")
3  objBL.ConnectionString = "provider=SQLOLEDB.1;data source=SERVER;database=DB;uid=UID;pwd=PWD"
4  objBL.ErrorLogFile = "c:\error.log"
5  objBL.CheckConstraints=true
6  objBL.XMLFragment = True
7  objBL.SchemaGen = True
8  objBL.SGDropTables = True
9
10 objBL.Execute "SampleSchema.xsd", "Customer-in.xml"
11 Set objBL = Nothing
```

Listing 3.1: *Visual Basic Script (BLoad.vbs)*

Da der MAGE-ML Standard bisher nur eine *Document Type Definition* (DTD) implementiert, für Bulk Load jedoch ein annotiertes XML-Schema nötig ist, wird im Rahmen dieser Arbeit ein Beispieldatensatz mit entsprechendem XDR-Schema zur Evaluation verwendet. Diese Beispieldaten basieren auf der im SQL Server integrierten Beispieldatenbank *Northwind*. Die Datensätze sind den *Online Books* der *Microsoft MSDN Library* [Cor03b] entnommen.

Die gespeicherten Daten können nun über HTTP exportiert werden. Hierbei wird in der Adresszeile des Browsers neben Angaben wie *Servername* und *Datenbank* eine XPath-Anfrage eingegeben, die das gewünschte Ergebnis als Schema-konformes XML-Dokument zurückgibt. Näheres dazu im Abschnitt 3.1.3. Damit eine solche XPath-Anfrage möglich ist, muss der IIS vorher konfiguriert werden.

Auf den SQL Server wird über den IIS zugegriffen, indem ein *virtuelles Verzeichnis* (virtualRoot) mit dem *IIS Virtual Directory Management-Dienstprogramm* erstellt wird. Das XML-Schema wird in einem weiteren untergeordneten virtuellen Verzeichnis, dem *virtuellen Schema-Verzeichnis* (virtualSchema) abgelegt. Im Management-Dienstprogramm werden diese Informationen hinterlegt, so dass die gestellte Anfrage schließlich an die richtige Datenbank weitergeleitet werden kann. Eine korrekte Anfrage-Syntax zeigt folgendes Beispiel:

Beispiel 3.1.1 `http://IISServer/virtualRoot/virtualSchema/SchemaFile/XPath-Anfrage`

3.1.3 Testlauf

Der Import der Beispieldaten über das Visual Basic Script funktionierte zufriedenstellend und nachdem das XDR-Schema korrekt implementiert worden ist, ist auch der Datenexport verlustfrei. Um den Datenexport und die Unterstützung von XPath-Anfragen zu testen, wurden einige Beispielanfragen an die Northwind Datenbank des SQL Servers gestellt. Folgende Anfragen zeigen zum Beispiel den Export eines bestimmten Elements.

Beispiel 3.1.2 `http://IISServer/Northwind/schema/SampleSchema.xsd/Customer[@CustomerID="GROSR"]?root=Northwind`

Beispiel 3.1.3 `http://IISServer/Northwind/schema/SampleSchema.xsd/Customer[@CustomerID="ALFKI"]/Order[@OrderID="Ord-10643"]?root=Northwind`

Die resultierenden XML-Dokumente sind in den Anhängen B.3 und B.4 zu finden.

3.2 Oracle XML DB

Die *Oracle Corporation* [Cor04a] hat seit dem zweiten *Release* ihrer Datenbank *Oracle 9i* das XML-Datenmodell des W3C in vollem Umfang aufgenommen. *Oracle XML DataBase* (Oracle XML DB), ein Teil des Datenbanksystems, ist demnach vollkommen auf XML ausgerichtet und bietet somit eine leistungsstarke Datenspeicherung für XML-Dokumente. XML-Dokumente können entweder in einer relationalen Tabellenstruktur mit Hilfe eines XML-Schemas (*strukturierte Speicherung*), oder als *Character Large Objects* (CLOBs) als Ganzes in einer *XMLType*-Tabelle abgespeichert werden (*unstrukturierte Speicherung*). Oracle XML DB bietet neben der Speicherung von XML-Daten und XML-Anfragen an diese Daten außerdem Möglichkeiten für Updates und Transformationen. XML-Anfragen an die Datenbank sind, wie bisher noch für XML-Dokumente vorgesehen, mit XPath möglich. Wie schon im Abschnitt 2.2.1 erwähnt, ist als Erweiterung zu XPath vom W3C die XML-Anfragesprache XQuery vorgesehen. Obwohl sich dieser Standard noch in der Entwicklung befindet, ist bereits bei *Oracle 9i Release 2* ein XQuery-Prototyp [Cor04d] implementiert. Sobald XQuery vom W3C als Standard verabschiedet wird, soll diese Anfragesprache auch fester Bestandteil der Oracle XML DB werden.

Die *Oracle 9i* Datenbank ist sehr umfangreich und bietet eine große Vielfalt an Anwendungsmöglichkeiten. Neben der relationalen Datenverwaltung und SQL-Umgebung werden für XML-Anwendungen verschiedenste Zugriffsmöglichkeiten über das Internet (HTTP, FTP, WebDAV), der Datenbankschnittstelle *SQL*Plus* und sehr detailliert auch über die JDBC-Schnittstelle zur Verfügung gestellt.

Da im Rahmen dieser Arbeit nicht auf alle Zugriffsmöglichkeiten eingegangen werden konnte, wurde in Anlehnung an die Voraussetzungen aus 2.4.2 versucht, ein Anwendungsbeispiel mit einem Datenbankzugriff über Java zu implementieren. Der MAGE-ML Standard ist, wie schon erwähnt, bisher nur mit einer *Document Type Definition* (DTD) implementiert. Daher wurden zur Evaluation der Oracle XML DB ebenfalls Beispieldatensätze verwendet, die der *Oracle 9i XML Database Online Documentation* [Cor04f] entnommen wurden (siehe auch Abschnitt 3.2.3).

3.2.1 Vorbereitung

In Zusammenarbeit mit der *Databases and Information Systems Group* (DBIS) am Institut für Informatik der Georg-August-Universität Göttingen wurde der Zugriff auf die dort bereits installierte *Oracle 9i* Datenbank realisiert.

3.2.2 Implementation

Nach 2.4.2 sollen XML-Daten mit Hilfe eines XML-Schemas in der Datenbank abgelegt werden. Oracle XML DB bietet ein einfaches System für den Umgang mit XML-Schemata. In der Datenbank kann für jede XML-Dokumentenstruktur ein eigenes XML-Schema registriert werden. Das Registrieren eines XML-Schemas ist auf verschiedene Arten möglich:

- Eine XML-Schema-Datei kann im lokalen Dateisystem der Datenbank abgespeichert werden. Hierfür sind vom Datenbank-Administrator entsprechende Benutzer-Verzeichnisse anzulegen.

Der Upload ist zum Beispiel über die FTP-Schnittstelle der Datenbank möglich. Anschließend wird das XML-Schema in der Datenbank unter einem Schema-Namen registriert. Die Registrierung übernimmt die Funktion *registerSchema()*, die aus einer Prozedur aufgerufen werden kann. Eine solche Prozedur zeigt Listing 3.2

```
1 begin
2 dbms_xmlschema.registerSchema
3 (
4 '%SchemaName%',
5 xdbURIType('/%InternerDateisystempfad%/schema.xsd').getClob(),
6 True,True,False,True
7 );
8 end;
9 /
```

Listing 3.2: *SQL-Script zur Schema-Registrierung (registerSchema1.sql)*

- Eine weitere Möglichkeit ist die Registrierung ohne vorherigen Upload. Hierbei wird das XML-Schema in einer Prozedur als String deklariert und anschließend ebenfalls mit der Funktion *registerSchema()* unter einem angegebenen Schema-Namen in der Datenbank registriert. Listing 3.3 zeigt dieses Beispiel

```
1  declare
2    doc varchar2(2000) :=
3  '<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
4        xmlns:xdb="http://xmlns.oracle.com/xdb">
5    <xs:element name="NAME" xdb:defaultTable="TABLE">
6      <xs:complexType xdb:SQLType="XML_NAME">
7        [...]
8      </xs:complexType>
9    </xs:element>
10 </xs:schema>';
11 begin
12   dbms_xmlschema.registerSchema('%SchemaName%', doc);
13 end;
14 /
```

Listing 3.3: *SQL-Script zur Schema-Registrierung (registerSchema2.sql)*

Oracle XML DB akzeptiert zwei Arten von XML-Schemata, zum Einen das vom W3C standardisierten XML-Schema [Rec01] und zum Anderen ein mit Erweiterungen versehenes annotiertes XML-Schema. Beim standardisierten Schema wird das relationale Modell von den Element- und Attributnamen abgeleitet. Wird ein annotiertes Schema verwendet, werden die Daten anhand der Erweiterungen im resultierenden relationalen Modell angelegt. Mit einem annotierten Schema ist es also möglich, das relationale Modell präzise festzulegen (zum Beispiel Tabellennamen und Datentypen).

Nachdem ein XML-Schema in der Datenbank registriert worden ist, können zu diesem Schema konforme XML-Daten in die Datenbank importiert werden. Für diesen Datenimport werden auf den Internetseiten des *Oracle XML DB Sample Corners* [Cor04c] ausführliche Beispielprogramme

in Java angeboten. Zum Einen das Javaprogramm *Simple Bulk Loader*, das eine große Anzahl von XML-Dokumenten aus einem angegebenen Verzeichnis ohne Verwendung eines XML-Schemas in XMLType-Tabellen als Character Large Objects (CLOBs) speichert, zum Anderen das Programm *SAX Loader*, welches XML-Dokumente anhand eines XML-Schemas im relationalen Modell der Datenbank ablegt. Beide Programme benutzen zum Ablegen der XML-Daten die JDBC-Schnittstelle der Oracle XML DB.

Nachdem die XML-Daten in der Datenbank abgelegt worden sind, können Anfragen an die Datenbank gestellt werden. Diese Anfragen können wie gefordert mit XPath durchgeführt werden. XPath-Anfragen werden bei Oracle XML DB in SQL-Syntax eingebettet und durch die Funktionen *existsNode()* und *extractValue()* gestellt.

In Verbindung mit der Funktion *value()* kann mit *existsNode()* in der *WHERE*-Klausel der SQL-Anfrage ein XML-Dokument ausgewählt werden, welches bestimmte Bedingungen erfüllt. Eine solche Anfrage zeigt Listing 3.4.

```
1 SELECT value(x)
2 FROM PurchaseOrder x
3 WHERE existsNode(value(x),'/PurchaseOrder[Reference="ADAMS-20011127121040988PST"]') = 1
4 /
```

Listing 3.4: *XPath-Anfrage mit 'existsNode()' (existsNode.sql)*

extractValue() bietet die Möglichkeit, einzelne Textknoten eines XML-Dokuments auszuwählen, die vom XPath-Ausdruck selektiert werden. Weitere Anfragebedingungen sind außerdem mit *existsNode()* in der *WHERE*-Klausel der SQL-Anfrage möglich. Listing 3.5 zeigt ein solches Beispiel.

```
1 SELECT extractValue(value(x),'/PurchaseOrder/Reference')
2 FROM PurchaseOrder x
3 WHERE existsNode(value(x),'/PurchaseOrder/LineItems/LineItem/Part[@Id="037429139523"]') = 1
4 /
```

Listing 3.5: *XPath-Anfrage mit 'extractValue()' (extractValue.sql)*

3.2.3 Testlauf

Die in den Listings 3.2 und 3.3 sowie 3.4 und 3.5 angegebenen Beispiele wurden der umfangreichen *Basic Demo* auf [Cor04b] entnommen. Eine sehr detaillierte Dokumentation ist in der *Oracle9i XML Database Online Documentation* [Cor04f], speziell zur *Basic Demo* im *Oracle9i XML Database Developer's Guide* [Cor04e] zu finden.

Anhand der *Basic Demo* und den ausführlichen Dokumentationen konnten ohne Komplikationen verschiedene XML-Schemata in der Oracle XML DB registriert werden und mit Hilfe des Programms *SAX Loader* entsprechende XML-Beispieldaten (*PurchaseOrder*) in die Datenbank importiert werden. Anschließende XPath-Anfragen aus Beispielen der *Basic Demo* (siehe Listings 3.4 und 3.5) wurden außerdem fehlerfrei über die SQL*Plus-Schnittstelle ausgeführt.

Darüber hinaus wurden diese XPath-Anfragen über ein Java-Programm gestellt. Das Java-Programm stellt eine Verbindung zur Datenbank über JDBC her und führt eine XPath-Anfrage aus. Das zurückgelieferte ResultSet wird bearbeitet und die Ergebnisse werden auf der Textkonsole

ausgegeben. Damit das zurückgelieferte ResultSet korrekt bearbeitet werden kann, ist es wichtig, die JDBC-Verbindung mit Hilfe des *Oracle Call Interface* (OCI) herzustellen. Teile der Codefragmente wurden dem *Oracle9i XML Database Developer's Guide* [Cor04e] entnommen. Listing C.1 im Anhang zeigt den Javacode des Beispielprogramms.

3.2.4 Oracle XQuery-Prototyp

Da die Grenzen von XPath schnell erreicht sind, komplexe Anfragen wie zum Beispiel *Joins* nicht mit XPath formuliert werden können, bot es sich an, den XQuery-Prototyp [Cor04d] der Oracle XML DB zu testen. Bei diesem Prototypen werden XQuery-Anfragen über ein Java-Programm an die Datenbank gestellt und Ergebnisse, wie es auch bei SQL-Anfragen in Java üblich ist, als *ResultSets* zurückgeliefert, die dann entsprechend verarbeitet und ausgewertet werden können.

Für eine solche XQuery-Anfrage wurde ein kleines Beispielprogramm mit Hilfe der *Oracle Java XQuery API* [Cor04g] in Java programmiert. Dieses Programm stellt die Verbindung zur Oracle XML DB über JDBC her, stellt eine XQuery-Anfrage an diese Datenbank und bearbeitet das zurückgelieferte ResultSet in der Weise, dass die Ergebnisse in der Textkonsole ausgegeben werden. Anhang C.2 zeigt den Javacode des Beispielprogramms.

Beim Testen des XQuery-Prototyps wurde festgestellt, dass die XQuery-Implementation noch nicht ausgereift ist, Datenbankanfragen aus dem Java-Programm heraus also nur teilweise möglich waren. So waren zum Beispiel Anfragen an die Daten des Oracle-Beispiel-Benutzers "*Scott*"möglich, nicht aber Anfragen an die *PurchaseOrder*-Daten der *Basic Demo*. Dieses Thema wird im Kapitel 5 noch einmal näher betrachtet.

4 Beurteilung

Im Kapitel 3 wurden die beiden Datenbanken *Microsoft SQL Server 2000* und *Oracle 9i XML DB* vorgestellt und auf ihre XML-Fähigkeit hin analysiert und getestet. Im Folgenden sollen diese Datenbanken in Bezug auf den Kriterienkatalog aus 2.4.2 beurteilt und bewertet werden.

4.1 Microsoft SQL Server

Plattformunterstützung

Der *SQL Server 2000* ist ein Produkt der Microsoft Corporation. Die Datenbank kann also nur auf einem Windowssystem installiert und verwendet werden. Insbesondere ist die Datenbank nur mit Microsoft-Produkten bedienbar. Somit wird eine Abhängigkeit des Anwenders auf Programme erzwungen, die der Microsoft-Familie angehören (IIS, Microsoft SQL Client-Software). Folglich ist der Einsatz dieser Datenbank beschränkt und findet zum Beispiel keine Anwendung auf einem Unix-System. Aus diesem Grund muss das Kriterium Plattformunterstützung als negativ bewertet werden, da hier keine Flexibilität geboten wird.

Programmiersprachen

Der *SQL Server 2000* bietet nur sehr beschränkt eine geforderte Java-Anbindung. Über die JDBC-Schnittstelle sind aus einem Java-Programm heraus beispielsweise nur sehr simple SQL-Anfragen möglich. Die Möglichkeiten, XPath-Anfragen mittels Java zu stellen, sind hingegen nicht gegeben. Stattdessen bietet Microsoft mit seinen speziellen Programmiersprachen *Visual Basic*, *Visual C#* u.a. solche Zugriffsmöglichkeiten. Für eine umfangreiche XML-Unterstützung sind außerdem noch besondere Updates nötig, vgl. [Cor03c] und [Cor03a]. Doch XPath-Anfragen aus anderen Programmen sind nur sehr eingeschränkt möglich. Für eine umfangreiche Entwicklungsumgebung für *Visual Basic* oder *Visual C#* bräuchte man das *Visual Studio* von Microsoft. Ein Nachteil von Visual Studio ist allerdings, dass bei einer Anschaffung relativ hohe Lizenzgebühren anfallen.

XML-Schema-Unterstützung

Beim *SQL Server 2000* werden nur XML-Schemata unterstützt, die um Microsoft-Spezifikationen erweitert wurden. Ein vom W3C standardisiertes XML-Schema ist hier nicht ausreichend und die Möglichkeit, eine *Document Type Definition* (DTD) zu verwenden, ist von vornherein nicht gegeben.

Schema-basierter Im- und Export von Daten

Ist ein annotiertes XML-Schema gegeben, so ist ein Datenimport mittels der speziellen Programmiersprachen von Microsoft einfach und problemlos möglich. Der Export von XML-Daten, also

die Möglichkeit, XPath-Anfragen zu stellen, ist nur über die Web-Schnittstelle des *Microsoft Internet Information Server* (IIS) möglich. Hierbei wird ein annotiertes XML-Schema serverseitig hinterlegt und eine Ausgabe der XPath-Anfrage dem Schema entsprechend generiert. Der IIS ist ein eigenständiger Webserver. Da allerdings meist schon ein Webserver betrieben wird, birgt das Einrichten eines weiteren Webservers weiteren administrativen Aufwand. Ein Nachteil von IIS ist auch, dass man ihn nur auf Windows Servern installieren kann. Dies erschwert eine Umsetzung der XPath-Unterstützung und muss deshalb als besonders negativ bewertet werden.

Dokumentation und Support

Zum *SQL Server 2000* finden sich sowohl zahlreiche Dokumentationen im Internet, den so genannten *Online Books* [Cor03b], als auch in Büchern oder Artikeln. Die Dokumentation in diesen Online Books ist zwar umfangreich, sie kann aber aufgrund ihrer Struktur schnell unübersichtlich werden. Der Anwender erhält außerdem nur eine eher oberflächliche Übersicht, da detailliertere Informationen häufig nicht zu finden sind. Support bietet Microsoft über zahlreiche Mailinglisten an, speziellerer Support ist jedoch sehr kostenintensiv.

4.2 Oracle XML DB

Plattformunterstützung

Die *Oracle 9i* unterstützt verschiedene Plattformen. Die Datenbank kann sowohl auf Windows-Betriebssystemem als auch auf Unix-Systemen installiert und verwendet werden. Die komplette Datenbankverwaltung basiert auf Java-Programmen. Somit kann *Oracle 9i* flexibel eingesetzt werden, ohne dass zum Beispiel auf das vorhandene Betriebssystem Rücksicht genommen werden muss. Die Plattformunterstützung zeichnet *Oracle 9i* daher besonders aus.

Programmiersprachen

Oracle 9i XML DB ist nach eigenen Angaben vollständig Java-kompatibel. Über die JDBC-Schnittstelle sind folglich jegliche Arten von Anfragen an die Datenbank plattformunabhängig möglich.

XML-Schema-Unterstützung

Oracle 9i XML DB bietet die Möglichkeit, mehrere Schemata in der Datenbank zu registrieren, so dass eine gleichzeitige Verwaltung verschiedener XML-Daten problemlos möglich ist. Darüber hinaus unterstützt die Datenbank sowohl vom W3C standardisierte als auch annotierte Schemata. Wie beim *Microsoft SQL Server 2000* bietet *Oracle 9i XML DB* jedoch ebenfalls keine Unterstützung für DTDs. Dies ist darauf zurückzuführen, dass das W3C als Schema für XML-Dokumente nur das XML-Schema als Standard veröffentlicht hat.

Schema-basierter Im- und Export von Daten

Der Umgang mit XML-Schemata bei *Oracle 9i XML DB* gestaltet sich sehr einfach. Mit Hilfe der registrierten Schemata werden XML-Dokumente erkannt und validiert. Sind sie zum Schema nicht konform, wird der Import mit einer entsprechenden Fehlermeldung abgelehnt. XPath-Anfragen werden in SQL-Syntax eingebettet und sind so in vollem Umfang auch aus einem Java-Programm formulierbar. Die Daten werden so entweder als Textknoten [*extractValue()*], oder als ganzes XML-Dokument [*existsNode()*] Schema-konform und somit verlustfrei exportiert. Der bereits implementierte XQuery-Prototyp bietet außerdem die Möglichkeit, nicht nur Textknoten, sondern auch ganze Teilbäume eines XML-Dokuments zu extrahieren. Sobald XQuery vom W3C standardisiert ist, wird XQuery fester Bestandteil der *Oracle XML DB* werden.

Dokumentation und Support

Zu *Oracle 9i XML DB* gibt es zahlreiche Online-Tutorials, Bücher oder Artikel. Die Dokumentationen der *Oracle9i XML DB Online Documentation* [Cor04f] und des *Oracle9i XML DB Developer's Guide* [Cor04e] sind sehr umfangreich, sehr einfach zu verstehen und gut nachzuvollziehen. Durch diese Hilfestellungen wird die Arbeit mit *Oracle 9i XML DB* enorm erleichtert. Einziger Nachteil ist die begrenzte Anzahl speziellerer Dokumentationen bezüglich XPath und XQuery. Dies mag jedoch darauf zurückzuführen sein, dass diese Themen noch sehr neu sind und sich teilweise noch in der Entwicklung befinden. Oracle bietet an Support nicht nur zahlreiche Mailinglisten und ein eigenes Forum, sondern darüber hinaus auch spezielleren Support, der im Lizenzumfang des Datenbanksystems enhalten ist. Somit entstehen für einen umfangreichen Support keine weiteren Kosten.

4.3 Zusammenfassende Gegenüberstellung

Zusammengefasst kann festgestellt werden, dass *Oracle 9i* viel besser den Kriterien aus 2.4.2 entspricht. Klar im Vorteil gegenüber *Microsoft SQL Server 2000* sind bei *Oracle 9i* die umfangreiche Plattformunterstützung, die Java-Anbindung und der Umgang mit standardisierten XML-Schemata. Die XPath-Anfragemöglichkeit beim *SQL Server 2000* über den IIS gestaltet sich zwar einfacher, ist aber aufgrund des hohen Administrationsaufwandes und der fehlenden Anfragemöglichkeiten aus einem Programm heraus nicht realisierbar. Außerdem bietet die Java-Unterstützung von *Oracle 9i* eine plattformunabhängige und flexiblere Arbeitsweise. Darüber hinaus verspricht die XQuery-Implementation bedeutend mehr XML-Unterstützung. Der *SQL Server 2000* bietet auf einem Windows-Betriebssystem mit entsprechender Entwicklungsumgebung abgestimmte und sehr umfangreiche Möglichkeiten bezüglich der XML-Unterstützung, aufgrund der Anforderungen des Kriterienkatalogs fällt die Bewertung für *Oracle 9i* jedoch bedeutend positiver aus. Der Einsatz von *Oracle 9i* ist demnach deutlich empfehlenswerter und dem *Microsoft SQL Server 2000* vorzuziehen.

5 Fazit und Ausblick

5.1 Fazit

Im Rahmen dieser Bachelorarbeit wurden die XML-Fähigkeiten der zwei Datenbanksysteme *Microsoft SQL Server 2000* und *Oracle 9i Release 2* betrachtet. Ausgehend von einer bioinformatischen Aufgabenstellung wurden Kriterien erarbeitet, anhand derer die Datenbanken vorgestellt, getestet und abschließend bewertet wurden.

In Kapitel 2 wurden die biologischen Grundlagen für Microarray-Experimente und die Bedeutung dieser Microarray-Technologie in der Bioinformatik vorgestellt. Darüber hinaus wurden die für Microarray-Daten notwendigen Datenstandards ausgearbeitet. Diese Microarray-Daten werden im XML-Format gespeichert und unterliegen entsprechenden Standards, damit eine Portabilität und allgemeine Vergleichbarkeit von Microarray-Daten gewährleistet werden kann. Diese Standards wurden von der MGED Society entwickelt und sind mittlerweile bei fast allen wichtigen Web Repositories für Microarray-Daten etabliert. Ausgehend von diesen Grundlagen wurde am Ende des zweiten Kapitels ein Kriterienkatalog entworfen, der als Grundlage für das Ablegen von Microarray-Daten in einer Datenbank diente.

Kapitel 3 hat sich mit der Analyse der beiden Datenbanken befasst. Es wurde ein kurzer Einblick in die grundlegenden Eigenschaften der Datenbanken gegeben und anhand des Kriterienkatalogs aus Kapitel 2 wurden Anwendungsbeispiele erarbeitet. Darauf aufbauend wurden die XML-Fähigkeiten umgesetzt und anschließend getestet. Es hat sich gezeigt, dass die beiden Datenbanken sehr unterschiedlich mit XML umgehen. *Oracle 9i* bietet hier unfangreichere und flexiblere Möglichkeiten.

In Kapitel 4 wurden die beiden Datenbanken anhand des Kriterienkatalogs beurteilt und bewertet. Den Abschluss bildete hier ein zusammenfassender Vergleich, bei dem festgestellt wurde, dass *Oracle 9i* bedeutend mehr Vorteile in Bezug auf den Umgang mit XML-Daten vorzuweisen hat. So beschränkt sich der *Microsoft SQL Server 2000* beispielsweise eher auf Anwendungen auf Windows-Betriebssystemen und bietet XPath-Anfragen nur über eine Web-Schnittstelle. *Oracle 9i* bietet hingegen bedeutend mehr Flexibilität in Bezug auf die Plattform- und Java-Unterstützung. Darüber hinaus unterstützt *Oracle 9i* als einzige Datenbank das standardisierte XML-Schema und zukunftsweisend auch XQuery.

5.2 Ausblick

Die Bachelorarbeit hat die Umgangsweise mit Schema-basierten XML-Daten nur anhand von Anwendungsbeispielen zeigen können, da der MAGE-ML Standard bisher nur eine *Document Type Definition* vorsieht, die Datenbanken aber nur mit einem XML-Schema umgehen können. Aufbauend auf den Erkenntnissen dieser Arbeit kann nun das Speichern und Verwalten von Microarray-

Daten umgesetzt werden. Sollte in absehbarer Zeit noch kein XML-Schema für Microarray-Daten seitens der MGED Society zur Verfügung gestellt werden, so kann im Rahmen weiterführender Arbeiten ein solches Schema an der Abteilung Bioinformatik entwickelt werden. Die Entwicklung könnte in Zusammenarbeit mit der MGED Society erfolgen. Anschließend kann dieses XML-Schema die Grundlage für die Implementierung der Microarray-Datenspeicherung unter *Oracle XML DB* bilden. Mittlerweile zeigt sich, dass der MAGE Standard in Bereichen der Genomforschung einen immer größer werdenden Stellenwert erreicht und für Publikationen auf Basis von Microarray-Experimenten Bedingung ist. Erstrebenswert ist darüber hinaus die Anwendung von XQuery. Insbesondere sind mit XQuery bedeutend komplexere und präzisere Anfragen an die Datenbank möglich, die beim Umgang mit Microarray-Daten nötig sind. Diese Eigenschaft ist wichtig für die Analyse und Interpretation von Genexpressionsdaten. So könnten nicht nur bestimmte Informationen über ein Experiment, sondern ganze Teilergebnisse eines Experiments abgefragt werden. Die Implementierung des XQuery Standards in der *Oracle XML DB* verspricht eine solche Umsetzung.

A Umfang von MAGE-OM

Experiment Beschreibung eines Microarray-Experiments als Einheit

BioAssay Experimentelle Schritte oder Zustände unterteilt in drei Typen: 'physical', 'measured' und 'derived' BioAssay

ArrayDesign Dieses Paket enthält 'design element groups' sowie Informationen über 'element zone layouts'

DesignElement Drei Typen: 'feature', 'reporter' und 'composit sequence'

BioMaterial Eine Abstraktion verschiedener Zustände in 'biologisch-basierten Materialien', die in verschiedenen Stadien eines Experiments Anwendung finden

BioAssayData Daten werden oft als 3-dimensionale Matrizen mit 'bioassays', 'design elements' und 'quantitation types' als Dimensionen betrachtet

QuantitationType Zwei Typen: 'standard' und 'specialized'

Array Ein Array wird durch seine Herkunft, Herstellung und Eigenschaften beschrieben

BioEvent Mögliche 'events' eines Experiments

Protocol Protokollierte Verwendung von Hard- und Software sowie 'parametrisierbaren' Objekten

AuditAndSecurity Kontaktbeziehungen zu Personen und Organisationen

Description Detaillierte Beschreibung als Klartext, Referenzen zu Datenbank- oder Ontologieeinträgen sowie bibliographischen Referenzen

HigherLevelAnalysis Clustern von Experimentierdaten

B Microsoft SQL Server Listings

```
1  <?xml version="1.0" ?>
2  <Schema xmlns="urn:schemas-microsoft-com:xml-data"
3          xmlns:dt="urn:schemas-microsoft-com:datatypes"
4          xmlns:sql="urn:schemas-microsoft-com:xml-sql">
5    <ElementType name="Customer" sql:relation="Customers">
6      <AttributeType name="CustomerID" dt:type="id" />
7      <AttributeType name="CompanyName" />
8      <AttributeType name="ContactName" />
9      <AttributeType name="City" />
10     <AttributeType name="Fax" />
11     <AttributeType name="Orders" dt:type="idrefs"  sql:id-prefix="Ord-" />
12     <attribute type="CustomerID" />
13     <attribute type="CompanyName" />
14     <attribute type="ContactName" />
15     <attribute type="City" />
16     <attribute type="Fax" />
17     <attribute type="Orders" sql:relation="Orders" sql:field="OrderID">
18       <sql:relationship
19                 key-relation="Customers"
20                 key="CustomerID"
21                 foreign-relation="Orders"
22                 foreign-key="CustomerID" />
23     </attribute>
24     <element type="Order">
25       <sql:relationship
26                key-relation="Customers"
27                key="CustomerID"
28                foreign-relation="Orders"
29                foreign-key="CustomerID" />
30     </element>
31   </ElementType>
32   <ElementType name="Order" sql:relation="Orders">
33     <AttributeType name="OrderID" dt:type="id" sql:id-prefix="Ord-" />
34     <AttributeType name="EmployeeID" />
35     <AttributeType name="OrderDate" />
36     <AttributeType name="RequiredDate" />
37     <AttributeType name="ShippedDate" />
38     <attribute type="OrderID" />
39     <attribute type="EmployeeID" />
40     <attribute type="OrderDate" />
41     <attribute type="RequiredDate" />
42     <attribute type="ShippedDate" />
43     <!-- ... -->
44   </ElementType>
45   <!-- ... -->
46 </Schema>
```

Listing B.1: *Annotiertes XDR-Schema (SampleSchema.xsd)*

```
1  <?xml version="1.0" encoding="utf-8" ?>
2  <Northwind>
3    <Customer CustomerID="DRACD" CompanyName="Drachenblut Delikatessen" ContactName="Sven
       Ottlieb" City="Aachen" Fax="0241-059428" Orders="Ord-10363 Ord-10391 Ord-10797 Ord-10825 Ord
       -11036 Ord-11067">
4      <Order OrderID="Ord-10363" EmployeeID="4" OrderDate="1996-11-26T00:00:00" RequiredDate="
       1996-12-24T00:00:00" ShippedDate="1996-12-04T00:00:00">
5        <OrderDetail ProductID="Prod-31" UnitPrice="10" Quantity="20">
6          <Discount>0</Discount>
7        </OrderDetail>
8        <OrderDetail ProductID="Prod-75" UnitPrice="6.2" Quantity="12">
9          <Discount>0</Discount>
10       </OrderDetail>
11       <OrderDetail ProductID="Prod-76" UnitPrice="14.4" Quantity="12">
12         <Discount>0</Discount>
13       </OrderDetail>
14     </Order>
15     <Order OrderID="Ord-10391" EmployeeID="3" OrderDate="1996-12-23T00:00:00" RequiredDate="
       1997-01-20T00:00:00" ShippedDate="1996-12-31T00:00:00">
16       <OrderDetail ProductID="Prod-13" UnitPrice="4.8" Quantity="18">
17         <Discount>0</Discount>
18       </OrderDetail>
19     </Order>
20     <Order OrderID="Ord-10797" EmployeeID="7" OrderDate="1997-12-25T00:00:00" RequiredDate="
       1998-01-22T00:00:00" ShippedDate="1998-01-05T00:00:00">
21       <OrderDetail ProductID="Prod-11" UnitPrice="21" Quantity="20">
22         <Discount>0</Discount>
23       </OrderDetail>
24     </Order>
25     <Order OrderID="Ord-10825" EmployeeID="1" OrderDate="1998-01-09T00:00:00" RequiredDate="
       1998-02-06T00:00:00" ShippedDate="1998-01-14T00:00:00">
26       <OrderDetail ProductID="Prod-26" UnitPrice="31.23" Quantity="12">
27         <Discount>0</Discount>
28       </OrderDetail>
29       <OrderDetail ProductID="Prod-53" UnitPrice="32.8" Quantity="20">
30         <Discount>0</Discount>
31       </OrderDetail>
32     </Order>
33     <Order OrderID="Ord-11036" EmployeeID="8" OrderDate="1998-04-20T00:00:00" RequiredDate="
       1998-05-18T00:00:00" ShippedDate="1998-04-22T00:00:00">
34       <OrderDetail ProductID="Prod-13" UnitPrice="6" Quantity="7">
35         <Discount>0</Discount>
36       </OrderDetail>
37       <OrderDetail ProductID="Prod-59" UnitPrice="55" Quantity="30">
38         <Discount>0</Discount>
39       </OrderDetail>
40     </Order>
41     <Order OrderID="Ord-11067" EmployeeID="1" OrderDate="1998-05-04T00:00:00" RequiredDate="
       1998-05-18T00:00:00" ShippedDate="1998-05-06T00:00:00">
42       <OrderDetail ProductID="Prod-41" UnitPrice="9.65" Quantity="9">
43         <Discount>0</Discount>
44       </OrderDetail>
45     </Order>
46   </Customer>
47 </Northwind>
```

Listing B.2: *Import von Beispieldaten (Customer-in.xml)*

```
1  <?xml version="1.0" encoding="utf-8" ?>
2  <Northwind>
3    <Customer CustomerID="GROSR" CompanyName="GROSELLA-Restaurante" ContactName="Manuel Pereira"
       City="Caracas" Fax="(2) 283-3397" Orders="Ord-10268 Ord-10785">
4      <Order OrderID="Ord-10268" EmployeeID="8" OrderDate="1996-07-30T00:00:00" RequiredDate="
       1996-08-27T00:00:00" ShippedDate="1996-08-02T00:00:00">
5        <OrderDetail ProductID="Prod-29" UnitPrice="99" Quantity="10">
6          <Discount>0</Discount>
7        </OrderDetail>
8        <OrderDetail ProductID="Prod-72" UnitPrice="27.8" Quantity="4">
9          <Discount>0</Discount>
10       </OrderDetail>
11     </Order>
12     <Order OrderID="Ord-10785" EmployeeID="1" OrderDate="1997-12-18T00:00:00" RequiredDate="
       1998-01-15T00:00:00" ShippedDate="1997-12-24T00:00:00">
13       <OrderDetail ProductID="Prod-10" UnitPrice="31" Quantity="10">
14         <Discount>0</Discount>
15       </OrderDetail>
16       <OrderDetail ProductID="Prod-75" UnitPrice="7.75" Quantity="10">
17         <Discount>0</Discount>
18       </OrderDetail>
19     </Order>
20   </Customer>
21 </Northwind>
```

Listing B.3: *Export von Beispieldaten (output1.xml)*

```
1  <?xml version="1.0" encoding="utf-8" ?>
2  <Northwind>
3    <Order OrderID="Ord-10643" EmployeeID="6" OrderDate="1997-08-25T00:00:00" RequiredDate="
       1997-09-22T00:00:00" ShippedDate="1997-09-02T00:00:00">
4      <OrderDetail ProductID="Prod-28" UnitPrice="45.6" Quantity="15">
5        <Discount>0.25</Discount>
6      </OrderDetail>
7      <OrderDetail ProductID="Prod-39" UnitPrice="18" Quantity="21">
8        <Discount>0.25</Discount>
9      </OrderDetail>
10     <OrderDetail ProductID="Prod-46" UnitPrice="12" Quantity="2">
11       <Discount>0.25</Discount>
12     </OrderDetail>
13   </Order>
14 </Northwind>
```

Listing B.4: *Export von Beispieldaten (output2.xml)*

C Oracle XML DB Listings

```
1  /*** A simple XPath-Query application for Oracle XML DB ***/
2  import java.sql.*;
3  import java.io.*;
4  import oracle.xml.parser.v2.*;
5  import oracle.xdb.XMLType;
6  import oracle.jdbc.*;
7  import oracle.sql.OPAQUE;
8  import org.w3c.dom.*;
9
10 public class TestXPath
11 {
12     public static void main(String[] args)
13     {
14     try
15     {
16         // register the JDBC driver
17         DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
18
19         // and get the connection
20         Connection conn = DriverManager.getConnection("jdbc:oracle:oci8:@","USER","PWD");
21
22         // create SQL query string
23         String sql = "SELECT extractValue(value(x),'/PurchaseOrder/Reference') "+
24         "FROM PurchaseOrder x "+
25         "WHERE existsNode(value(x),'/PurchaseOrder/LineItems/LineItem/Part[@Id=\"037429139523\"]')
       = 1";
26
27         // create an OraclePreparedStatement using that connection...
28         OraclePreparedStatement stmt = (OraclePreparedStatement) conn.prepareStatement(sql);
29
30         // create a ResultSet and execute the PrepareStatement
31         ResultSet rset = stmt.executeQuery();
32
33         // create an OracleResultSet using that rset
34         OracleResultSet orset = (OracleResultSet) rset;
35
36         while(orset.next())
37         {
38         // Display the retrieved details
39         System.out.println(orset.getString(1));
40         }
41     }
42     catch(Exception x)
43     {
44         x.printStackTrace();
45     }
46     }
47 }
```

Listing C.1: *Java-Programm für XPath-Anfragen (TestXPath.java) [Cor04e]*

```
1  /*** A simple XQuery application for Oracle XML DB ***/
2  import oracle.xquery.*;
3  import java.sql.*;
4  import java.io.*;
5  import oracle.xml.parser.v2.*;
6
7  public class TestXQuery
8  {
9      public static void main(String[] args)
10     {
11
12     try
13     {
14         // get the connection (for example, using the thick JDBC Driver)
15         DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
16
17         String url = "jdbc:oracle:thin:USER/PWD@SERVER:PORT:SID";
18         Connection conn = DriverManager.getConnection(url);
19
20         // create a context using that connection...
21         XQueryContext ctx = new XQueryContext(conn);
22
23         // create xquery string
24         Reader strm = new StringReader(
25                 "FOR $i IN sqlquery(\"select * from scott.emp\")/ROW "+
26                 " RETURN <EMP empno=\"{$i/EMPNO}\">$i/ENAME,$i/SALARY</EMP>");
27
28         // prepare the query
29         PreparedXQuery xq = ctx.prepareXQuery(strm);
30
31         // get a resultset
32         XQueryResultSet rset = xq.executeQuery(true);
33
34         while (rset.next())
35         {
36         XMLNode node = rset.getNode();  // get result nodes
37         System.out.println(" NODE "+ node.getNodeName());
38         node.print(System.out);
39         }
40     }
41     catch(Exception x)
42     {
43         x.printStackTrace();
44     }
45     }
46 }
```

Listing C.2: *Java-Programm für XQuery-Anfragen (TestXQuery.java) [Cor04g]*

Abkürzungsverzeichnis

CLOB	Character Large OBject
DBMS	DatenBank Management System
DBS	DatenBank System
DTD	Document Type Definition
EBI	European Bioinformatics Institute
GEO	Gene Expression Omnibus
IIS	Internet Information Server
JDBC	Java DataBase Connectivity
MAGE	MicroArray Gene Expression
MDAC	Microsoft Data Access Components
MGED	Microarray Gene Expression Data
MIAME	Minimum Information About a Microarray Experiment
ML	Markup Language
NCBI	National Center for Biotechnology Information
OJXQI	Oracle Java XQuery API
OM	Object Model
OTN	Oracle Technology Network
SGML	Standard Generalized Markup Language
SMD	Stanford Microarray Database
SQL	Structured Query Language
UML	Unified Modelling Language
W3C	World Wide Web Consortium
XML	eXtensible Markup Language
XPath	XML Path Language
XQuery	XML Query Language

Literaturverzeichnis

[AE04] European Bioinformatics Institute. *ArrayExpress Database*. `http://www.ebi.ac.uk/arrayexpress/`, March 2004.

[AEF04] European Bioinformatics Institute. *ArrayExpress Flyer*. `http://www.ebi.ac.uk/microarray/AEFlyer.pdf`, 19 March 2004.

[Aff04] Affymetrix, Inc. *GeneChip Arrays*. `http://www.affymetrix.com/products/arrays/`, March 2004.

[BHQ+01] Alvis Brazma, Pascal Hingamp, John Quackenbush, Gavin Sherlock, Paul Spellman, Chris Stoeckert, John Aach, Wilhelm Ansorge, Catherine A. Ball, Helen C. Causton, Terry Gaasterland, Patrick Glenisson, Frank C.P. Holstege, Irene F. Kim, Victor Markowitz, John C. Matese, Helen Parkinson, Alan Robinson, Ugis Sarkans, Steffen Schulze-Kremer, Jason Stewart, Ronald Taylor, Jaak Vilo, and Martin Vingron. *Minimum information about a microarray experiment (MIAME) – toward standards for microarray data*. Nature Publishing Group, December 2001.

[Cor03a] Microsoft Corporation. *Microsoft Data Access Components 2.8*. `http://www.microsoft.com/downloads/details.aspx?familyid=6c050fe3-c795-%4b7d-b037-185d0506396c&displaylang=de`, December 2003.

[Cor03b] Microsoft Corporation. *MSDN Library*. `http://msdn.microsoft.com/library/`, December 2003.

[Cor03c] Microsoft Corporation. *SQLXML 3.0*. `http://www.microsoft.com/downloads/details.aspx?familyid=4c8033a9-cf10-%4e22-8004-477098a407ac&displaylang=de`, December 2003.

[Cor04a] Oracle Corporation. *Oracle Technology Network*. `http://otn.oracle.com/`, March 2004.

[Cor04b] Oracle Corporation. *Oracle XML DB*. `http://otn.oracle.com/tech/xml/xmldb/`, March 2004.

[Cor04c] Oracle Corporation. *Oracle XML DB Sample Corner*. `http://otn.oracle.com/sample_code/tech/xml/xmldb/`, March 2004.

[Cor04d] Oracle Corporation. *Oracle XQuery Prototype*. `http://otn.oracle.com/sample_code/tech/xml/xmldb/xmldb_xquerydownload.h%tml`, 23 March 2004.

[Cor04e] Oracle Corporation. *Oracle9i XML Database Developer's Guide - Oracle XML DB, Release 2*. http://otn.oracle.com/documentation/oracle9i.html, March 2004.

[Cor04f] Oracle Corporation. *Oracle9i XML Database Online Documentation, Release 2*. http://otn.oracle.com/documentation/oracle9i.html, March 2004.

[Cor04g] Oracle Corporation. *The Oracle Java XQuery API (OJXQI)*. http://otn.oracle.com/sample_code/tech/xml/xmldb/jxqi.html, March 2004.

[Dra03] W3C Working Draft. *XQuery 1.0: An XML Query Language*. http://www.w3.org/TR/xquery/, 12 November 2003.

[Dra04] W3C Working Draft. *XQuery 1.0 and XPath 2.0 Formal Semantics*. http://www.w3.org/TR/xquery-semantics/, 20 February 2004.

[GBB+02] Jeremy Gollub, Catherine A. Ball, Gail Binkley, Janos Demeter, David B. Finkelstein, Joan M. Hebert, Tina Hernandez-Boussard, Heng Jin, Miroslava Kaloper, John C. Matese, Mark Schroeder, Patrick O. Brown, David Botstein, and Gavin Sherlock. *The Stanford Microarray Database: data access and quality assessment tools*. Nucleic Acid Research, January 2002.

[GEO04] National Center for Biotechnology Information. *Gene Expression Omnibus*. http://www.ncbi.nlm.nih.gov/geo/, March 2004.

[JCB02] Christian J. Stoeckert Jr, Helen C. Causton, and Catherine A. Ball. *Microarray databases: standards and ontologies*. Nature Publishing Group, December 2002.

[Kni01] Rolf Knippers. *Molekulare Genetik*. Georg Thieme Verlag, 8 edition, 2001.

[Pev03] Jonathan Pevsner. *Bioinformatics and functional genomics*. John Wiley & Sons, 2003.

[Rec99] W3C Recommendation. *XML Path Language 1.0*. http://www.w3c.org/TR/xpath, 16 November 1999.

[Rec01] W3C Recommendation. *XML Schema Part 0–2*. http://www.w3c.org/TR/xmlschema-0/, 2 May 2001.

[Rec04] W3C Recommendation. *eXtensible Markup Language (XML) 1.1*. http://www.w3c.org/TR/2004/REC-xml-20040204/, 4 February 2004.

[SMD04] University of Stanford. *Stanford Microarray Database*. http://genome-www.stanford.edu/microarray/, March 2004.

[Sta04] University of Stanford, California, USA. http://www.stanford.edu/, March 2004.