Evaluation von Datenbanken zur Speicherung von strukturierten Metadaten am Beispiel LOM

Stefan Hoermann, Ronny John, Cornelia Seeberg und Ralf Steinmetz

Multimedia Kommunikation - KOM
Technische Universität Darmstadt
Merckstr. 25
64283 Darmstadt

{S.Hoermann, C.Seeberg, R.Steinmetz}@kom.tu-darmstadt.de

Abstract: Im E-Learning-Bereich hat sich in den letzten Jahren das Erfassen von Metadaten zu Lernressourcen etabliert. Diese sollen das Finden und Wiederverwenden von Lernressourcen ermöglichen. Werden die Metadatensätze in einer Datenbank gespeichert, so können sie leicht durchsucht werden. Doch durch die Komplexität von hierfür etablierten Metadatenstandards fällt das Speichern der Datensätze in herkömmlichen relationalen Datenbanken schwer. In diesem Beitrag prüfen wir deshalb am Beispiel LOM die Speicherung der Metadaten in nativen XML-Datenbanken. Anschließend stellen wir die Ergebnisse der Evaluation dreier Datenbanken, die wir mit dem eigens hierfür erstellten Test-Szenario ermittelt haben.

1 Motivation und Einführung

Im E-Learning-Bereich hat sich in den letzten Jahren das Erfassen von Metadaten zu Lernressourcen etabliert. Diese sollen das Finden und Wiederverwenden von Lernressourcen ermöglichen. Für digitale und nichtdigitale Lernressourcen ist mit Learning Object Metadata (LOM) [LWG02] ein dafür spezialisiertes Metadatenschema vom IEEE Learning Technology Standards Committee [LTSC] entwickelt worden und letztes Jahr standardisiert worden. Da der Standard keine Abbildung in Computer-verarbeitbarer Syntax vorschreibt, sind verschiedene Bindings für XML [IMS02] und RDF [CID03] basierend auf dem LOM-Standard entwickelt worden.

Der LOM-Standard beschreibt ein sehr umfangreiches, stark strukturiertes Metadatenschema. Die Metadaten einer Lernressource werden in den neun Kategorien General, Lifecycle, Meta-Metadata, Technical, Educational, Rights, Relation, Annotation und Classification unterschieden. Besonders durch die Kategorie Educational ist das LOM-Schema zur Beschreibung von Lernressourcen geeignet. Alle Kategorien des LOM-Schemas enthalten weitere Datenelemente, die teilweise strukturierte Zusammenfassungen weiterer Datenelemente darstellen können. Das LOM-Schema definiert auch Datentypen, wie Vokabulare und sprachbehaftete Zeichenketten

(Langstrings). Letztere ermöglichen die Erfassung der Metadaten in mehreren Sprachen. Diese Langstrings und die häufig verwendeten 1-zu-N-Relationen im LOM-Schema auf Basis der Kategorien und Datenfelder, machen das LOM-Schema sehr komplex und unhandlich in der Verwendung in Applikationen.

Werden die Metadatensätze in einer Datenbank gespeichert, können sie leicht durchsucht werden, um so Autoren einen bequemen und schnellen Zugriff auf Lernressourcen zu geben. Aus unserer Erfahrung, die wir in den Forschungsprojekten [KMED01] [SRHF00] [SSFS99] gesammelt haben, wissen wir, dass es schwierig ist, LOM-Datensätze in herkömmlichen relationalen Datenbanken abzulegen. Besonders durch die baumförmige Struktur von LOM mit den vielen 1-zu-N-Relationen bieten sich statt dessen XML-Datenbanken an. In diesem Beitrag prüfen wir deshalb am Beispiel LOM die Speicherung der Metadaten in nativen XML-Datenbanken. Dabei setzen wir voraus, dass die Metadaten nach dem XML-Binding [IMS02] des IMS Global Learning Consortium abgebildet werden. Anschließend stellen wir die Ergebnisse der Evaluation dreier Datenbanken vor, die wir mit dem eigens hierfür erstellten Test-Szenario ermittelt haben.

Der Beitrag ist wie folgt gegliedert: Im nächsten Abschnitt wird die Speicherung von LOM-Datensätzen in herkömmlichen und XML-Datenbanken verglichen. In Abschnitt drei werden die evaluierten Datenbanken vorgestellt. Abschnitt vier beschreibt die Implementierung der Evaluationsumgebung. In Abschnitt fünf werden die durchgeführten Messungen beschrieben und deren Messergbenisse verglichen. In Abschnitt sechs fassen wir diesen Beitrag zusammen und geben einen Ausblick über zukünftige Arbeiten.

2 XML und Datenbanken

In diesem Abschnitt werden unterschiedliche Möglichkeiten zur Speicherung von LOM-Datensätzen mit XML-Binding sowohl in traditionellen Datenbanken (relational und objektorientiert), als auch in sogenannten nativen XML-Datenbanken, betrachtet.

2.1 Traditionelle Datenbanken

Das LOM-Schema definiert Datenfelder, die die Metainformationen der Lernressourcen enthalten. Zusätzlich zu diesen Datenfeldern definiert das LOM-Schema eine Struktur dieser Datenfelder, nach der diese Datenfelder in Zusammenhang zu bringen sind. Diese Struktur stellt damit einen wichtigen Bestandteil von LOM-Datensätzen dar, sie darf nicht verloren gehen. Diese Struktur ist auch in den LOM-Datensätzen wiederzufinden, die im XML-Binding nach XML abgebildet wurden. Die Abbildung der LOM-Datensätze in traditionellen Datenbanken stellt eine Herausforderung dar.

Unter den traditionellen Datenbanken versteht man die relationalen Datenbanken, oft auch als SQL-Datenbanken bezeichnet, und die objektorientierten Datenbanken. Beide Typen von Datenbanken werden entweder durch einen eingebauten XML-Aufsatz oder

durch eine spezielle externe Middleware (Zwischenschicht) “XML-fähig”. Die bekannten Anbieter von Datenbanken wie Oracle, IBM DB2 oder Microsoft SQL-Server bieten XML-Aufsätze an. Das Produkt Castor der Exolab-Gruppe [AVY+01] ist ein Beispiel für eine Middleware, die auch frei verfügbar ist. In [Bou03] ist das Table-Based Mapping, eine Art der modellbasierten Abbildung von XML-Daten auf Datenbanken, und das Object-Relational Mapping, ein Ansatz der expliziten, modellbasierten Abbildung von XML-Daten auf Datenbanken, beschrieben.

2.2 Native XML-Datenbanken

Der Begriff “native XML-Datenbank” wurde von der Software AG [SAGa] mit ihrem Produkt Tamino XML-Server geprägt. Native XML-Datenbanken sind speziell für die Speicherung und Verwaltung von XML-Daten entwickelt. Der Hauptunterschied liegt darin, dass die interne Datenbankarchitektur speziell und ausschließlich für XML konzipiert ist. Die Hauptvorteile dieser Datenbanken sind:

- schnelle Anbindung ohne Erstellung einer expliziten Abbildungsvorschrift,
- XML-Dokumente werden ohne Verlust und mit Erhalt ihrer Struktur gespeichert,
- spezielle Abfragesprachen wie XPath oder XQuery erlauben die Adressierung einzelner Elemente aufgrund ihrer Reihenfolge (Beispiel: “Hole 4. Schwesterelement der 3. Ebene”),
- das Holen von ganzen Dokumenten ist in der Regel schneller, als in relationalen Datenbanken.

Textbasierte XML-Datenbanken speichern XML-Dokumente als Textdateien im Dateisystem, als BLOB (Binary Large Object) in einer Datenbank oder in einem eigenen Format. Das Gemeinsame an diesen Datenbanken ist die intensive Verwendung von Indexen. Mit Hilfe der Indexe ist die Datenbank in der Lage, extrem schnell auf einzelne Dokumente oder Elemente innerhalb von Dokumenten zuzugreifen. Im Vergleich zur relationalen oder Model-Based XML-Datenbanken, sind textbasierte Datenbanken meistens schneller. Jedoch nur dann, wenn die Daten in der Form angefordert werden, wie sie physikalisch abgelegt worden sind. Dies ist beispielsweise dann der Fall, wenn Dokumente komplett geladen werden.

3 Evaluierte Datenbanken

In diesem Abschnitt stellen wir die drei evaluierten Datenbanken jeweils kurz vor.

3.1 Xindice

Xindice [BSS+02] ist eine native XML-Datenbank. Das Projekt wird unter der Apache Software Foundation geführt und ist als Open- Source frei verfügbar. Die Datenbank

wurde von Grund auf für die Verarbeitung von XML entwickelt. Sie gehört zu den Typen der modellbasierten Datenbanken und speichert XML in Form eines komprimierten DOM ab. Für die Zugriffsbeschleunigung werden Indizes verwendet. Nach [BSS+02] eignet sich die Datenbank insbesondere für viele kleine XML-Dokumente. Damit ist das Produkt also gut für die zu speichernden LOM-Datensätze geeignet. Die Datenbank unterstützt die Partitionierung in Collections. Neben der Abfragesprache XPath wurde auch eine eigene Implementierung von XUPDATE für die Änderung einzelner Elemente oder Attribute erstellt. Zugang erhält man über die XMLDB-API der XML:DB Initiative [Sta01]. Für die Evaluation haben wir Version 1.0RC1 verwendet.

3.2 Tamino

Die Tamino XML-Datenbank von der Software AG [SAGa] ist Teil einer Produktreihe zur Erstellung von Anwendungen auf Basis von XML. Neben der eigentlichen Datenbank werden Applikationen wie Tamino X-Node, ein WebDAV Server oder ein DTD Schema-Editor mitgeliefert. Für die Integration der Datenbank in eigene Programme steht ein Software Developer Kit (SDK) bereit, mit dem es auch möglich ist, Erweiterungsmodule für die Datenbank zu entwickeln. Für die Evaluation wurde die kostenlose Testversion 2.3.1 der kommerziellen Datenbank verwendet. Die Datenbank baut auf ein relationales Backend auf, das über ODBC, JDBC oder OLE DB auch direkt angesprochen werden kann. Für den Benutzer ist der Zugang zu dieser relationalen Datenbank transparent. Die Abbildung der XML-Daten über ein Mapping erfolgt automatisch. Als Abfragesprache dient eine erweiterte XPath-Implementierung, die X-Query genannt wird.

3.3 eXist

eXist von Wolfgang Meier [Mei02] ist eine native XML-Datenbank, die entweder die Daten in einer relationalen Datenbank oder in einem eigenen proprietären Format speichern kann. Das Produkt steht als Open-Source frei zur Verfügung. Die Modellierung der XML-Daten ist sowohl für das relationale, als auch für das proprietäre Backend identisch. Die Daten werden als DOM abgespeichert. Für die Beschleunigung der Suche werden alle Knoten des DOMs mit Hilfe eines eindeutigen Identifiers indiziert. Die Datenbank unterstützt die Partitionierung der Dokumente in Collections. Die Indizierung der Dokumente wird für jede Collection getrennt ausgeführt. Für die Evaluation haben wir Version 0.7.1 verwendet.

4 Implementierung

Dieser Abschnitt beschreibt die Implementierung der Evaluationsumgebung. Diese umfasst eine Schnittstelle (API) für XML-fähige Datenbanken, einem XML-zu-Xpath-Konverter zur Formulierung von Suchanfragen, dem XML-Datengenerator zur Erstellung von Metadatendokumenten und Applikationen zur Performanzmessung.

4.1 Datenbankanbindung

Im Gegensatz zu den traditionellen relationalen Datenbanken, die in der Regel mit ODBC bzw. JDBC eine herstellerunabhängige standardisierte Schnittstelle anbieten, existiert bei den verwendeten XML-Datenbanken eine einheitliche Schnittstellenspezifikation noch nicht. Die XML:DB Initiative [Sta01] versucht zwar zur Zeit, mit „xmldb“ eine universelle Schnittstelle für XML-Datenbanken zu entwickeln, diese hat sich aber noch nicht allgemein durchgesetzt. Die Vorteile einer generischen Schnittstelle spiegeln sich in der hohen Flexibilität und der einfachen Weiterverwendung wider. Demgegenüber stehen die Nachteile des initial höheren Programmieraufwandes sowie gegebenenfalls Geschwindigkeitseinbußen durch Umwandlungsroutinen.

Für die Evaluation wurde daher eine eigene Schnittstelle entwickelt, die über zwei Stufen (XDB-Client und Connectoren) die proprietären APIs der jeweiligen Datenbankanbieter und die Schnittstelle auf der Anwenderseite verbindet. Die Programmierung der Schnittstelle erfolgte in Java. Die Schnittstelle auf der Anwenderseite stellt eine Klasse dar, die alle erforderlichen Methoden für das Speichern, Lesen, Löschen und Suchen von XML-Dokumenten bereitstellt.

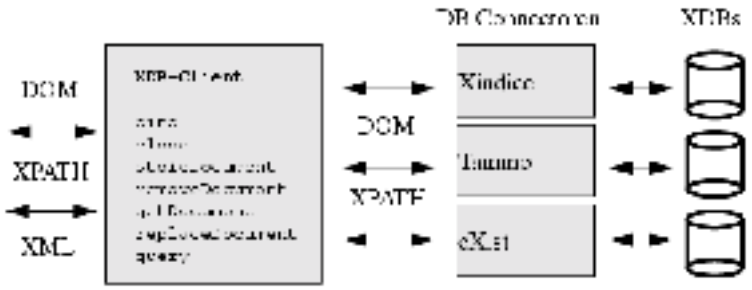

Abbildung 1: Generische Datenbankanbindung

4.2 XmlToXpath

XmlToXpath transformiert ein XML-Dokument in einen XPath-Ausdruck. Hintergrund der Entwicklung war die Idee, innerhalb des LOM-Editors [SRHF00] eine Suche nach LOM-Datensätzen ohne explizite Angabe eines XPath-Ausdrucks formulieren zu können. Die Suche sollte durch Ausfüllen der vorhandenen Textfelder oder Auswahllisten im LOM-Editor erfolgen. Eine unscharfe Suche mit der Angabe eines Sterns „*“ als Wildcard ist möglich. Je nach Position der Wildcard werden dazu die XPath-Funktionen contains(), starts-with() oder string-lenght() angewendet.

4.3 XML-Daten Generator

Für die Performanzmessung der Datenbanken ist es notwendig, entsprechende Datensätze im großen Umfang zu erstellen. Hierfür haben wir einen XML-Datengenerator entwickelt, der aus einem XML-Template eine wählbar große Menge an

LOM-Datensätzen erstellt. Das Template gibt in einem XML-Format die Struktur vor. Über Trennzeichen können verschiedene Werte der Datenelemente vorgegeben werden.

4.4 Applikationen für Performanzmessungen

Aufbauend auf dem API der generischen Datenbankanbindung wurden für die Performanzmessung zwei Tools für das Speichern und Suchen von Dokumenten entwickelt.

Das Programm zum Speichern von Dokumenten, kann eine definierbare Anzahl an XML-Dokumenten in einem Verzeichnis einlesen und in die Datenbank speichern. Bei diesem Vorgang wird die Zeit für das Abspeichern eines Dokumentes gemessen sowie die durchschnittliche Zeit und das Minimum, Maximum aller gespeicherten Dokumente.

Das Programm für die Suche von Dokumenten in der Datenbank, kann mit einem definierbaren XPath-Ausdruck Dokumente innerhalb einer Collection suchen und auf die Konsole ausgeben. Hierbei lässt sich die Suchzeit und separat die Zeit für die Aufbereitung und Ausgabe der gefunden Dokumente ermitteln.

5 Messung

Dieses Kapitel behandelt die Evaluierung der drei verwendeten Datenbanken Xindice, Tamino und eXist. Zuerst werden das Messverfahren für alle Datenbanken sowie allgemeine Beobachtungen dargestellt. Die Ergebnisse der Zeitmessungen bezüglich der Speicherung und Suche von Daten werden angegeben. Am Ende folgt eine zusammenfassende Bewertung aller drei Produkte.

5.1 Messverfahren

Vor Beginn der Evaluation haben wir die folgenden Randbedingungen für die Datenbanken aufgestellt. Untersucht werden soll die Eignung der Datenbanken für die Speicherung und Suche von LOM-Datensätzen. Diese stehen in Form von XML oder DOM zur Verfügung. Angenommen wurde eine maximale Anzahl von 10.000 LOM-Datensätzen (Dokumenten). Die Anzahl der gleichzeitigen Verbindungen (Netzwerkverbindungen) zum Datenbankserver wird auf kleiner fünf geschätzt. Die durchschnittliche Anzahl an Suchverknüpfungen wurde mit drei „UND“-Verknüpfungen pro Xpath-Ausdruck geschätzt.

Für die Untersuchung der Performanz ist die Anzahl der Datensätze sowie die Struktur der Daten entscheidend. Die Struktur der Daten liegt mit der Vorgabe der Verwendung des IEEE LOM 6.1 Schemas fest. Um das Verhalten der Datenbank mit ansteigender Anzahl gespeicherter Datensätze zu untersuchen, wurden daher die Messungen jeweils mit unterschiedlicher Anzahl von Dokumenten durchgeführt. Detaillierte Eigenschaften wurden mit 400, 1000, 4000 und teilweise nur mit 10.000 Dokumenten durchgeführt, die Messreihen mit 100, 200, 400, 800, 1000, 2000, 4000, 8000 und 10.000 Dokumenten.

Die Suchvorgänge wurden zum Teil als Nullsuche (eine Suchanfrage ohne Ergebnis) durchgeführt, um den Einfluss der Aufbereitung der Daten auf dem Server, der Übertragung und der Verarbeitung auf der Client-Seite zu minimieren. Die Messungen wurde mit zwei Computern ausgeführt, die über ein Ethernetkabel direkt miteinander verbunden waren. Der Datenbankserver lief unter dem Betriebssystem Windows NT 4.0 und war mit einer 750MHz CPU sowie 256 MB RAM ausgestattet. Der Client-PC lief ebenfalls unter NT und war mit einem 350MHz Prozessor und 128 MB RAM ausgerüstet.

Im Folgenden werden die einzelnen Messvorgänge beschrieben.

Unterschiedliche XPath-Funktionen: Untersucht wurde der Einfluss der Verwendung von unterschiedlichen XPath-Funktionen. Neben einem direkten Vergleich der Zeichenfolgen (element='value'), wurden die Funktionen, wie starts-with(), contains() und string-length(), beispielsweise für die unscharfe Suche, angewendet. Die Messung wurde als Nullsuche und mit der Datenbankstandardeinstellung ausgeführt. (Tabelle1)

Anzahl der Verknüpfungen: Die Anzahl der Verknüpfungen im XPath-Ausdruck wurde bei 10.000 Datensätzen als Nullsuche und jeweils einmal mit Indizierung und ohne Indizierung aller gesuchten Elemente ausgeführt. Der angegebene XPath-Ausdruck mit zehn logischen UND-Verknüpfungen wurde dabei von hinten um je einen Suchbegriff reduziert, so dass zehn Messungen mit jeweils einer Verknüpfung weniger entstanden sind. Der gesuchte Wert wurde gegebenfalls so verändert, dass kein Ergebnis zurückgeliefert wurde. (Tabelle 2)

Einfluss Anzahl der Ergebnisse: Der Einfluss der Anzahl zurückgelieferter Ergebnisse wurde mit logischen ODER Verknüpfungen, indizierten Elementen und bei 10.000 Datensätzen gemessen. (Tabelle 3)

Suche mit Indizierung: Gemessen wurde die Suchzeit für eine Suche auf indizierte Elemente. Die Indizierung erfolgte auf das Attribut record@XMLDataGenerator_ID sowie auf die Elemente langstring und vocentry. (Tabelle 4)

Suche mit und ohne Indizierung: Gemessen wurde die Suchzeit für eine Suche auf eine Mischung von indizierten und nicht indizierten Elementen. Die Indizierung erfolgte auf das Attribut record@XMLDataGenerator_ID und auf das Element langstring. (Tabelle5)

5.2 Messergebnisse

Mit dem beschriebenen Versuchsaufbau ergeben sich nach den oben genannten Messungen folgende Ergebnisse:

	eXist	Tamino	Xindice
direkter Vergleich	2157	7093	9390
starts-with()	2141	8438	4657
contains()	2125	9250	4547
string-length()	-)[1]	7750	4781

Contains-Operator ~=	-)[1]	7391	-)[1]

Tabelle 1:Unterschiedliche Xpath-Funktionen; 4000 Datensätze; Suchdauer in ms
)[1] Die entsprechenden Funktionen sind nicht verfügbar

Anzahl Verknüpfungen	1	2	3	4	5	6	7	8	9	10
eXist	2125	3219	4360	5718	6718	7750	10328	11797	13390	16250
Tamino	94	94	140	156	188	203	235	250	282	298
Xindice	234	297	391	422	422	687	1235	2172	2593	2391

Tabelle 2: Einfluss der Anzahl der Verknüpfungen (indiziert); Suchdauer in ms

Treffer	1	2	3	4	5	6	7	8	9	10
eXist	5547	21718	38313	52765	67922	83578	97187	111532	125281	139437
Tamino	94	172	187	218	250	197	313	344	359	360
Xindice	256	312	328	375	422	438	453	500	500	531

Tabelle 3: Einfluss der Anzahl der Treffer (indiziert); Suchdauer in ms

Datensätze	100	200	400	800	1000	2000	4000	8000	10000
eXist	516	735	1140	1922	2265	4125	10000	-)[2]	-)[2]
Tamino	468	516	562	703	840	1016	1516	2425	2844
Xindice	782	656	812	1109	1094	2219	3597	5016	7359

Tabelle 4: Suche mit Indizierung mit Treffern; Suchdauer in ms
)[2] Die Datenbank stürzte ab

Datensätze	100	200	400	800	1000	2000	4000	8000	10000
eXist	-)[3]	-)[3]	-)[3]	-)[3]	-)[3]	-)[3]	-)[3]	-)[3]	-)[3]
Tamino	500	563	672	875	922	1454	2438	4953	5828
Xindice	750	688	828	1141	1266	1954	3328	5016	7250

Tabelle 5: Suche mit und ohne Indizierung mit Treffern; Suchdauer in ms
)[3] eXist unterstützt in der getesteten Version nur Vollindizierung

5.3 Zusammenfassung

Die Ergebnisse der Messungen aller drei Datenbanken zeigen, dass vor allem die Formulierung des XPath-Ausdrucks, die Art der Indizierung und die Menge der zurückgelieferten Elemente ausschlaggebend für die Dauer einer Suchanfrage sind. Während für Xindice und Tamino selbst 10000 gespeicherte Dokumente in der Datenbank unproblematisch sind, ist eine vernünftige Nutzung von eXist lediglich bis 1000 Dokumente möglich. Abbildung 1 zeigt einen Vergleich der Zeiten aller drei Datenbanken für eine indizierte Suche mit Rückgabe von Ergebnissen an (Tabelle 4).

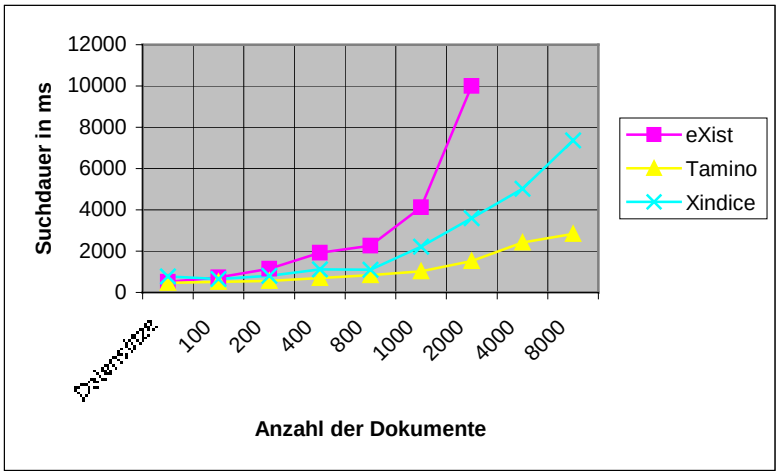

Abbildung 1: Vergleich aller Datenbanken: Indizierte Suche

Bei Xindice ist es wichtig, dass die Adressierung im XPath-Ausdruck so genau wie möglich erfolgt, und die Anzahl der Treffer nicht zu hoch ist. Nachteilig ist die Art der Indizierung, die es nicht erlaubt, ein Element unter der Angabe eines gesamten Pfades zu adressieren. Falls man Elemente wie langstring oder vocentry aus dem LOM-Schema indiziert, werden die dazugehörigen Indextabellen sehr groß und der Vorteil der Indizierung verringert sich.

Nach einer Indizierung der Elemente und Attribute zeigt sich, dass die Tamino-Datenbank auch sehr gut für eine große Anzahl von gespeicherten Dokumenten skaliert. Die Zeiten für eine Suche, beispielsweise 2,8 Sekunden bei Durchsuchung von 10.000 Datensätzen, liegen in einem sehr guten Bereich. Selbst bei nur teilweiser Indizierung der gesuchten Werte, ist die Suchzeit noch akzeptabel. Die Möglichkeit, eine Indizierung genau auf ein bestimmtes Element im Baum vornehmen zu können, reduziert den Aufwand für die Indexerstellung seitens der Datenbank erheblich. Im Vergleich zu Xindice, macht sich dies in einem besseren Ergebnis bezüglich der Suchdauer bemerkbar. Durch die unterschiedlichen Indizierungsverfahren wird die Datenbank den differenzierten Anforderungen einer Suche gerecht.

Die Ergebnisse der Messung bei eXist, sowohl bei der Speicherung der Daten, als auch bei der Suche zeigen, das eXist als Datenbank für eine größere Anzahl von Dokumenten nicht einsetzbar ist. Bei einer Suche über 1000 Dokumenten, steigt die Suchzeit extrem in die Höhe. Für eine Suche in 10.000 voll indizierten Elementen benötigt die Datenbank ungefähr 24 Sekunden. Hier wird der Nachteil einer Vollindizierung sichtbar. Verknüpfte XPath-Funktionen in einem Suchausdruck erhöhen die Zeit für eine Suche drastisch. Für Suchen mit verknüpften Ausdrücken, ist die eXist-Datenbank daher nicht geeignet.

6 Zusammenfassung und Ausblick

Wir haben gezeigt, dass es sinnvoll ist LOM-Datensätze in nativen XML-Datenbanken zu speichern. Die Anbindung an eine XML-Datenbank wird durch ein einfaches Interface belohnt. Dennoch muss man nicht auf die Performanz von relationalen Datenbanken verzichten, wenn einzelne Datenfelder der LOM-Datensätze indiziert werden. Trotzdem wäre der direkte Vergleich mit einer LOM-Datenbank basierend auf herkömmlichen Datenbanken sehr interessant. Die gemessenen Werte beruhen auf unserer Implementierung eines generischen Datenbank-Interfaces. Zukünftige Forschungsarbeiten werden sich rund um ein Modulrepositorium, dessen Elemente mit LOM beschrieben sind und das auf Xindice basiert, konzentrieren.

Literaturverzeichnis

[AVY+01] ExoLab Group, CASTOR - Mapping framework between Java objects, XML-documents and SQL & OQL database, http://castor.exolab.org/, 2001

[Bou03] Ronald Bourret, XML and Databases, http://www.rpbourret.com/xml/XMLAndDatabases.htm, 2003

[BSS+02] Apache XML Project, Apache Xindice XML Database, http:/ /xml.apache.org/xindice/, 2002

[CID03] Center for User Oriented Design / Knowledge Management Research Group, RDF binding of LOM metadata, http://kmr.nada.kth.se/el/ims/metadata.html

[KMED01] k-MED: Knowledge-based Multimedia Medicine Education, Entwicklung eines webbasierten multimedialen Lernsystems, http://www.k-med.org/Download/k-med.pdf, 2001

[IMS02] IMS Global Learning Consortium, Inc., IMS Meta-Data Specification, http://www.imsglobal.org/metadata/index.cfm

[LTSC] IEEE Learning Technology Standards Committee (LTSC), http://ltsc.ieee.org/, 2002

[LWG02] LOM Working Group, IEEE P1484.12/D6.4, IEEE Learning Technology Standards Committee, Draft Standard for Learning Objects Metadata, http://ltsc.ieee.org/wg12/index.html

[Mei02] Wolfgang Meier, eXist: Open Source XML Database, http:// exist.sourceforge.net/, 2002

[SAGa] Software AG, Tamino XML Server, http://www.softwareag.com/tamino/, 2002

[SRHF00] C. Seeberg, I. Rimac, S.Hoermann, A. Faatz, A. Steinacker, A. El Saddik, and R. Steinmetz, Medibook: Realisierung eines generischen Ansatzes für ein internetbasiertes Multimedia-Lernsystem am Beispiel Medizin, in Tagungsband: Treffen der GI-Fachgruppe 1.1.3 Maschinelles Lernen (GMD Report 114), pages 96-105, 2000

[SSFS99] A. Steinacker, C. Seeberg, S. Fischer, and R. Steinmetz, MultiBook: Meta-Data for the Webbased Learning Systems, Fachgebiet Industrielle Prozess- und Systemkommunikation, in Proceedings of the 2nd International Conference on New Learning Technologies, 1999

[Sta01] Kimbro Staken, Application Programming Interface for XML Databases, http://206.20.201.14/xapi/, 2001