XML-Datenbanksysteme

Architekturen und Benchmarks

Timo Böhme
Universität Leipzig
boehme@informatik.uni-leipzig.de

Erhard Rahm
Universität Leipzig
rahm@informatik.uni-leipzig.de

Kurzfassung. Die Notwendigkeit für XML-fähige Datenbanksysteme (DBS) resultiert aus der zunehmenden Verbreitung von XML als Datenaustauschformat, natives Speicherformat sowie als Basis für Web-Dokumente. Die Anforderungen an diese Datenbanken sind aufgrund des breiten Einsatzspektrums von XML sehr unterschiedlich, was sich in den verschiedenen Architekturen von XML-Datenbanken niederschlägt. Durch den Einsatz von Benchmarks können die einzelnen Systeme hinsichtlich ihrer Funktionen und Leistungsfähigkeit beurteilt werden. In diesem Beitrag werden nach einer Diskussion der Herausforderungen von XML an DBS die grundlegenden Architekturen für XML-DBS vergleichend vorgestellt, und es wird ein Überblick über veröffentlichte XML-DBS-Benchmarks gegeben.

1 XML: Herausforderung für Datenbanksysteme

Nach ihrer Spezifikation im Februar 1998 wird XML als Datenformat in einer wachsenden Zahl von Anwendungsbereichen eingesetzt und verdrängt dort teilweise die bestehenden Lösungen. Schwerpunktmäßig erfolgt dies in zahlreichern E-Business-Anwendungen, in denen XML als Datenaustauschformat bisherige Standards ersetzt sowie in vielen Anwendungs- und Systemprogrammen, die ihre Daten in XML abspeichern. Hier reicht die Spanne von Konfigurationsdateien bis hin zu Dokumentformaten von Bürosoftware. Der eigentliche Ausgangspunkt für die Entwicklung von XML - die Ablösung von HTML im Internet - stellt den nächsten wichtigen Meilenstein für XML dar. Erste Schritte in dieser Richtung sind mit XHTML sowie dem Einsatz von XML zur Datenrepräsentation auf dem Web-Server getan.

Der Grund für die große Akzeptanz von XML liegt in den folgenden Eigenschaften:

- XML-Dokumente weisen einen einfachen Aufbau auf
- Daten können beliebig komplex strukturiert werden
- hohe Flexibilität und Erweiterbarkeit
- einfach transformierbar
- Dokumente sind lesbar (kein Binärformat)
- Sprachenunabhängigkeit durch Unterstützung von Unicode
- verschiedene Dokumente können einheitlich mit den selben Werkzeugen bearbeitet werden
- es existiert eine große Auswahl an Werkzeugen für die Bearbeitung und Verwaltung von XML-Dokumenten

Mit dem Vorliegen großer Datenmengen im XML-Format wächst auch der Bedarf an Datenbanksystemen zur Verwaltung, da Dateisysteme hier zu wenig Funktionalität bieten. Das Einbringen von XML-Daten in herkömmliche DBS wirft jedoch Probleme auf, die hauptsächlich aus dem semi-strukturierten Charakter von XML entstehen. Während relationale DBS mit strukturierten Daten arbeiten, d.h. für diese Daten wurde schon vor dem Einfügen ein festes Schema im DBS definiert, das die Daten umfassend beschreibt und von dem sie auch nicht abweichen, ist eine Schemadefinition für XML-Daten optional. Stattdessen enthält ein XML-Dokument ein implizites Schema, das man beim Parsen der Daten erhält. Dieses Schema ist jedoch spezifisch für das geparste Dokument und entspricht normalerweise nicht dem Schema der Klasse von Dokumenten, zu dem das aktuell verarbeitete gehört. Zudem erlauben die XML-Schemadefinitionen (DTD, XML-Schema u.a.) die Definition optionaler bzw. freier Bestandteile (Elementinhalt ANY), die nicht in der strikten Schemabeschreibung herkömmlicher Datenbanksysteme vorgesehen sind. Es ist also festzuhalten, dass DBS für XML-Daten in der Lage sein müssen, Dokumente ohne vordefiniertes Schema verarbeiten zu können bzw. auch bestehende Schemas flexibel anzupassen.

Ein weiteres Problem bereitet XML den DBS aufgrund seiner historischen Wurzeln in SGML. Die für

Textdokumente entwickelte Formatsprache verlangt die Beibehaltung der Elementreihenfolge von Geschwisterelementen. Da XML ebenfalls in erster Linie für Textdokumente entwickelt wurde, gibt es auch hier diese Bedingung. Dies bedeutet, dass selbst Datenbanklösungen, die für semi-strukturierte Daten entwickelt wurden (z.B. Lore [MAGQ97] für das OEM-Datenmodell [PGMW95]) nicht ohne Erweiterungen für XML-Dokumente geeignet sind.

Ein aus Datenbanksicht ebenfalls nicht unerhebliches Problem resultiert aus der flexiblen Modellierungsfähigkeit und dem damit verbundenen breiten Anwendungsspektrum. Bei der Modellierung eines Schemas für eine bestimmte Anwendung können sowohl alle Daten innerhalb eines Dokumentes zusammengefasst oder auf viele kleine Dokumente aufgeteilt werden. Somit sollte ein XML-DBS in der Datenbank sowohl ein einziges riesiges Dokument als auch Millionen von kleinen Dokumenten verwalten können. Da die derzeitigen Lösungen vorwiegend Dokumente als kleinstes Verarbeitungsgranulat behandeln, sind diese weitestgehend auf den letzten Fall spezialisiert.

2 XML-Datenbanksysteme

Im vorangegangenen Abschnitt wurde aufgezeigt, dass herkömmliche DBS sich nicht direkt zur Verwaltung von XML-Daten eignen. Aus diesem Grund wurden sowohl DBS mit neuen Architekturen als auch Erweiterungen für herkömmliche DBS entwickelt [Bour02][RaVo02]. Für die in diesem Abschnitt folgende Diskussion der entstandenen Lösungen ist die Unterscheidung der Dokumenttypen dokumentorientiert und datenorientiert wesentlich, da die meisten Produkte speziell für einen der beiden Typen optimiert sind.

Dokumentorientierte XML-Daten weisen aus Datenbanksicht eine unregelmäßige Struktur auf, besitzen oft kein vordefiniertes externes Schema und damit auch keine Typinformationen über die enthaltenen Elemente und Attribute. Aus Sicht von XML ist von Bedeutung, dass selten DTDs oder Schemadefinitionen vorliegen, Elemente vom Typ „Gemischter Inhalt“ (mixed content) sein können und die Reihenfolge von benachbarten Elementen beim Speichern und im Ergebnis einer Anfrage erhalten bleiben muss. Die Dokumentgrößen reichen von einigen Kilobyte bis zu wenigen Megabyte.

Im Gegensatz dazu weisen *datenorientierte* XML-Dokumente eine regelmäßige Struktur auf, die durch ein vordefiniertes Schema mit Typinformationen angereichert ist. Die Daten zeigen damit eine große Ähnlichkeit zu Daten in relationalen Datenbanken. Da DTDs kaum Typinformationen enthalten, werden die Daten normalerweise durch eine mächtigere Schemasprache, wie z.B. XML Schema, beschrieben. Auch bei den XML-Eigenschaften unterscheidet sich dieser Datentyp von dem vorangegangenen. Die Elemente haben nur den Elementtyp Element-Inhalt oder sind Blätter der Elementhierarchie. Ebenfalls spielt hier die Reihenfolge der Geschwisterelemente keine Rolle und braucht durch das Datenbanksystem beim speichern und bei Anfragen nicht berücksichtigt zu werden. Dies entspricht dem relationalen Modell, in dem die Reihenfolge der Spalten und Tupel unerheblich ist. Die Elementhierarchie ist typischerweise flach und beträgt selten mehr als 5 Stufen. Da die Dokumente vergleichbar mit den Tupeln relationaler Datenbanken sind, liegt ihre Größe im Bereich von einigen 100 Byte bis zu wenigen Kilobyte.

Welche Dokumenttypen von den existierenden XML-DBS effizient unterstützt werden hängt von der zugrundeliegenden Architektur ab sowie dem Fokus des jeweiligen Systems ab. Relationale DBS mit einer XML-Schnittstelle sind vorrangig für datenorientierte XML-Dokumente geeignet, da deren Aufbau den strukturierten relationalen Daten am ähnlichsten ist. Neu entwickelte XML-DBS sollten mit beiden Dokumenttypen umgehen können. Content Management Systeme mit XML-Schnittstelle wiederum bedienen praktisch ausschließlich den dokumentorientierten Typus. Die Stärke dieser Systeme liegt in der Versionsverwaltung von Dokumenten. Strukturierte Anfragemöglichkeuten, Transaktionsverwaltung u.ä. bleibt jedoch den DBS vorbehalten. Neben dieser Unterscheidung können auch noch die folgenden Kriterien für eine Klassifikation der XML-DBS herangezogen werden:

- *Speicherstrategie*: zeichenkettenbasiert vs. modellbasiert
  Bei den aktuellen XML-DBS können zwei grundsätzliche Ansätze zur Speicherung der XML-Dokumente unterschieden werden. Die eine Variante - *zeichenkettenbasiert* - speichert das vollständige Dokument als flache Zeichenkette ab. Dies kann z.B. als Textdatei oder in einem CLOB[1] eines

Firma	DBS	Speicherstrategie	Back-End	Produktkategorie
IBM	DB2 mit XML-Extender	zeichenkettenbasiert, modellbasiert	objekt-relational	DBS mit XML-Erweiterung
privat: Wolfgang Meier	eXist	modellbasiert	relational, proprietär	Natives XML-DBS
eXcelon Corp.	eXtensible Information Server (XIS)	modellbasiert	objekt-orientiert	Natives XML-DBS
Infonyte	Infonyte DB	modellbasiert	proprietär	Natives XML-DBS
Oracle	Oracle 9i	zeichenkettenbasiert, modellbasiert	objekt-relational	DBS mit XML-Erweiterung
Microsoft	SQL Server 2000	modellbasiert	objekt-relational	DBS mit XML-Erweiterung
Software AG	Tamino	modellbasiert	proprietär	Natives XML-DBS
IXIASOFT	TEXTML Server	zeichenkettenbasiert	proprietär	Natives XML-DBS
X-Hive Corp.	X-Hive/DB	modellbasiert	objekt-orientiert	Natives XML-DBS
B-Bop Associates, Inc.	Xfinity Server	modellbasiert	relational	Natives XML-DBS
Apache Software Foundation	Xindice	modellbasiert	proprietär	Natives XML-DBS
XYZFind Corp.	XYZFind Server	modellbasiert	proprietär	Natives XML-DBS

Tabelle 1: Übersicht zu XML-Datenbanksystemen

Datenbanksystems erfolgen. Für eine effiziente Anfrageverarbeitung müssen Indexstrukturen (Struktur-, Werte- oder Volltextindex) angelegt werden. Diese Speicherform hat den Vorteil der relativ einfachen Realisierung, ermöglicht eine originalgetreue Archivierung der Dokumente und eine sehr schnelle Abfrage kompletter Dokumente. Problematisch hierbei sind häufige Änderungen in den Dokumenten sowie Anfragen, die nicht durch Indexstrukturen bearbeitet werden können und somit das Parsen der vollständigen Dokumente erfordern. Weiterhin ist mit hohen Wartezeiten im Mehrbenutzerbetrieb durch Sperrgranulate auf Dokumentebene zu rechnen.
In der *modellbasierten* Variante wird die Struktur beim Einlesen des Dokumentes analysiert und in ein bestimmtes Datenmodell abgebildet. Dieses Modell - im Allgemeinen eine Baumstruktur (z.B. DOM) - wird dann in der Datenbank gespeichert. Dieser Ansatz bietet den Vorteil, dass die Metadaten des Dokumentes bekannt und für die Optimierung der Anfragen verwendet werden können. Weiterhin sind Änderungsoperationen an Dokumentfragmenten einfach möglich und Zugriffsgranulate können entsprechend des Modells definiert werden. Schwierigkeiten sind bei dieser Variante bei der Wiederherstellung komplexer Dokumente zu erwarten, da diese aus den einzelnen Datenobjekten zusammengesetzt werden müssen. Weiterhin kann auch die originalgetreue Rekonstruktion der Daten nicht gewährleistet werden.

- *Back-End*: (objekt)-relational, objekt-orientiert, proprietär
  Die Speicherung der verarbeiteten XML-Dokumentdaten kann auf verschiedene Art und Weise erfolgen, wobei hier das dafür verwendete zugrundeliegende System als Back-End bezeichnet wird. Neben der *proprietären* Lösung, die die verarbeiteten Daten in einer spezifischen Struktur ablegt und selbst verwaltet - dies reicht von der einfachen Dateiverwaltung bis hin zum integrierten DBS -, gibt es auch eine Reihe von Produkten, die auf herkömmliche und ausgereifte DBS zurückgreifen. Im Falle von *(objekt)-relationalen* DBS werden beim modellbasierten Ansatz die Daten mittels eines Mappings auf Relationen abgebildet. Hier kann zwischen generischen Mappings, die beliebige Daten auf eine feste Anzahl von Relationen abbilden und schemaabhängigen Mappings, die ein von der Elementhierarchie abhängiges Relationsschema verwenden, unterschieden werden. Beim zeichenkettenbasierten Ansatz werden die Dokumente in den Relationen in CLOBs gespeichert.
  Eine elegante Methode stellt die Verwendung einer *objekt-orientierten* Datenbank als Back-End zur

1. CLOB - character large object

Speicherung der verarbeiteten Daten dar, da im Falle der modellbasierten Speicherung oftmals ein Objektbaum aus dem Dokument erzeugt wurde, der direkt in der Datenbank abgelegt werden kann. In dieser Kombination finden objekt-orientierte DBS ein neues Aufgabenfeld, das die Marktrelevanz dieser Systeme wieder vergrößern kann.

- *Produktkategorie*: Natives XML-DBS (NXD) vs. DBS mit XML-Erweiterung
  In [Bour02b] werden 8 Produktkategorien für XML-Software mit Datenbankcharakter aufgeführt. Dieser Beitrag beschränkt sich auf die Kernkategorien, in denen die Datenbankfunktionalität an erster Stelle steht.
  Für den Begriff der *Nativen XML-DBS* existiert leider keine genaue Definition. In Anlehnung an [Bour02] sollen native XML-DBS folgende Eigenschaften erfüllen:
  - der Datenzugriff erfolgt vorwiegend über XML-orientierte Schnittstellen (z.B. XPath, XSLT, DOM, SAX, XQuery), daraus folgt, dass das DBS in erster Linie zur Speicherung und Manipulation von XML-Daten bestimmt ist
  - ein XML-Dokument ist die elementare Verwaltungseinheit, d.h. es ist zwar möglich, auf Teile eines Dokumentes zuzugreifen, deren Verwaltung erfolgt jedoch im Kontext des Dokumentes

  Bemerkenswert bei dieser Definition ist, dass die Speicherstrategie oder die Art des Back-Ends ohne Bedeutung ist. Datenbanken, die zu dieser Kategorie gehören, sollten sowohl dokumentorientierte als auch datenorientierte XML-Dokumente effizient verwalten können.
  Zur Kategorie *DBS mit XML-Erweiterung* gehören alle DBS, die nicht in erster Linie für XML-Daten entwickelt wurden, jedoch über Erweiterungen für die Transformation von XML-Dokumenten in die internen Datenstrukturen und umgekehrt verfügen. Hierzu zählen vor allem die relationalen DBS, die meistens über spezielle benutzerdefinierte Datentypen (UDT) und benutzerdefinierte Funktionen (UDF) das Speichern und Abfragen von XML-Dokumenten erlauben. Die XML-Erweiterungen dieser Systeme dienen in erster Linie als Schnittstelle zwischen den Datenbankdaten und XML-Anwendungen. Sie sind daher vorrangig für datenorientierte XML-Dokumente optimiert.

Eine Auswahl der verfügbaren XML-DBS ist in Tabelle 1 zusammen mit einer Klassifikation nach obigen Kriterien aufgeführt[1]. Es ist erkennbar, dass die modellbasierte Speicherstrategie aufgrund ihrer größeren Flexibilität dominiert. Relationale DBS mit XML-Erweiterung bieten oftmals auch die zeichenkettenbasierte Speicherung in CLOBs an, um Textdokumente besser zu unterstützen. Strukturierte Anfragen über diese Daten sind jedoch nur eingeschränkt möglich. Vorteil dieser Systeme ist jedoch, dass Sie sowohl strukturierte relationale Daten als auch XML-Daten in einem DBS verwalten können, wodurch der Administrationsaufwand sinkt und der Datenaustausch vereinfacht wird.

3 Benchmarks für XML-Datenbanksysteme

Die im vorangegangenen Abschnitt besprochenen unterschiedlichen Architekturen der verfügbaren XML-Datenbanksysteme weisen systembedingte Vor- und Nachteile auf, die je nach dem anvisierten Anwendungsszenario relevant für die Entscheidung für einen bestimmten Ansatz sind. Weiterhin weisen XML-Datenbanksysteme auch innerhalb einer Kategorie teilweise erhebliche Unterschiede hinsichtlich Kosten, Funktionsumfang und der Leistungsfähigkeit auf. Ziel von XML-Datenbankbenchmarks ist es vor allem die Leistungsfähigkeit (Performance) umfassend und realitätsnah zu bewerten und einen Leistungsvergleich zwischen einzelnen Systemen zu ermöglichen.

Im Folgenden werden nach einer allgemeinen Betrachtung der Eigenschaften von Benchmarks die Anforderungen an XML-spezifische Benchmarks untersucht. Im Anschluss daran werden die drei existierenden XML-Benchmarks in der Reihenfolge ihrer Veröffentlichung beschrieben: X-Mach1, Xmark und XOO7. Die Beschreibung erfolgt in einheitlicher Weise und berücksichtigt die jeweilige Anwendungs-Domäne, Datenmerkmale, Operationen sowie die Metriken. Abschließend erfolgt ein Vergleich zwischen den Benchmarks.

1. für weitere XML-Datenbanksysteme sowie zusätzliche Informationen siehe [Bour02b]

3.1 Allgemeine Richtlinien für Benchmarks

Die Definition eines Benchmarks resultiert aus dem Bedarf nach einem Werkzeug zum objektiven Leistungsvergleich von Systemen bezüglich eines bestimmten Anwendungsbereichs. Ein aussagekräftiger Benchmark zur generellen Bestimmung der Leistungsfähigkeit von Computersystemen für alle Anwendungsgebiete ist nicht möglich, da je nach Anwendung stark unterschiedliche Anforderungen bestehen und einzelne Hardware- und Software-Komponenten sehr unterschiedlich stark beansprucht werden. Aus diesem Grund sollte ein ‘guter’ Benchmark auf einen spezifischen Aufgabenbereich (Domäne) fokussiert sein.

Wie in [Gray93] ausgeführt, sollte ein Benchmark neben der Fokussierung auf eine Domäne noch folgende Eigenschaften besitzen:

- *Relevanz*: Der Benchmark sollte die für den Anwendungsbereich wesentlichen Merkmale und Operationen abdecken, um aussagekräftige Ergebnisse zu erzielen.
- *Portabilität:* Der Benchmark sollte einfach auf allen relevanten Rechner-Plattformen implementiert werden können.
- *Skalierbarkeit:* Der Benchmark sollte für unterschiedlich große Konfigurationen eine Leistungsbewertung ermöglichen, von einem einzelnen PC bis hin zu Großrechner-Konfigurationen, für zentralisierte und verteilte Architekturen sowie für unterschiedlich große Datenmengen. Eine solche Skalierbarkeit ist auch wesentlich, um den Benchmark über einen längeren Zeitraum bei starken Verbesserungen in der CPU-Geschwindigkeit, Speichergrößen etc. als Maßstab einsetzen zu können. Damit die Ergebnisse vergleichbar bleiben, sind geeignete Skalierungsfaktoren zu berücksichtigen.
- *Einfachheit:* Damit ein Benchmark akzeptiert wird, muss seine Architektur und Funktionsweise so einfach sein, dass er leicht verständlich, nachvollziehbar und implementierbar ist. Durch diese Eigenschaft sind die Ergebnisse überprüfbar, wodurch sich ihre Glaubwürdigkeit erhöht.

3.2 Betrachtungen zu XML-DBS-Benchmarks

Nachdem im vorangegangenen Abschnitt allgemeine Kriterien für Benchmarks diskutiert wurden, geht es jetzt um die speziellen Anforderungen und Probleme von XML-Benchmarks. Zunächst wird die Bedeutung der Domänen im XML-Umfeld betrachtet. Anschließend werden Kriterien für den XML-Datenbankbereich besprochen, die den Benchmark grundlegend bestimmen. Am Ende dieses Abschnitts wird auf Problembereiche, die sich aus dem aktuellen Entwicklungsstand von XML ergeben, eingegangen.

3.2.1 Domänen für XML-Datenverwaltung

Ebenso wie es keinen universellen Datenbank-Benchmark gibt, kann es auch für die XML-Datenverwaltung keinen alle Anwendungsbereiche abdeckenden universellen Benchmark geben. Aufgrund der größeren Bandbreite unterschiedlicher XML-Daten und den damit verbundenen Operationen gilt dies hier sogar noch stärker. Denn aufgrund des breiten Einsatzspektrums von XML ist es nicht möglich, eine Datenbankstrukturierung und Menge an Operationen zu definieren, die für jeden Bereich umfassend und relevant ist. Jeder XML-Datenbankbenchmark hat daher einen bestimmten Anwendungsschwerpunkt festzulegen. Für diese Domänen-Fokussierung wesentlich sind vor allem die Art der XML-Daten (dokumentorientiert, datenorientiert oder Mischformen) sowie die jeweils benötigten Operationen. Im Folgenden diskutieren wir die für die Benchmarks relevanten Merkmale / Anforderungen, die sich aus den drei wesentlichen Arten von XML-Daten ergeben.

Dokumentenkollektionen. Ein wesentliches Anwendungsgebiet für XML-Datenbanken ist die Verwaltung von *dokumentorientierten* XML-Daten im Rahmen von Dokumentenkollektionen, wie sie in Firmen-Intranets oder im Internet auftreten. Für diese Domäne sind folgende Operationstypen relevant:

- hierarchische und sequentielle Navigation
- Volltextsuche auf Elementinhalten
- Abrufen von vollständigen Dokumenten

- Extraktion von Dokumentfragmenten
- Erstellung von Verzeichnissen ausgewählter Elemente (z.B. Inhaltsverzeichnis)
- Einfügen und Löschen von Dokumenten
- Einfügen und Löschen von Dokumentfragmenten in einem Dokument

Strukturierte Daten. Im Gegensatz zur vorangegangenen Domäne sind hier *datenorientierte* oder strukturierte XML-Daten zu verwalten, wie sie u.a. in zahlreichen E-Business-Anwendungen dominieren. Gleichfalls ergeben sich auch andere Schwerpunkte hinsichtlich der relevanten Operationen:

- Suche auf Attributwerten
- Anfragen mit Sortierung, Aggregation und Verbundanweisungen
- Erzeugung neuer Ergebnisdokumente basierend auf Anfragedaten
- Einfügen und Löschen von Dokumenten
- Änderung von Attributwerten

Gemischte Daten. Die beiden beschriebenen Ansätze stellen Extreme beim Einsatz von XML dar - dokumentorientiert vs. datenorientiert. Viele Anwendungsgebiete erfordern jedoch Mischformen von XML-Daten. Zum Beispiel sind in Online-Katalogen neben strukturierten Produktdaten auch textuelle bzw. multimediale Zusatzinformationen zu verwalten. Die für diese Domäne relevanten Operationstypen sind ebenfalls eine Mischung der Operationen für datenorientierte und dokumentorientierte XML-Daten. Ein Benchmark muss dabei das Verhältnis der Operationstypen entsprechend dem simulierten Anwendungsfall wählen. Eine Übernahme aller Operationstypen würde zu einem sehr umfangreichen Benchmark führen, der das Kriterium der Einfachheit verletzt. Zudem sind nicht alle Operationen für eine Domäne gleichermaßen bedeutsam.

3.2.2 Spezifische Kriterien für einen XML-Datenbankbenchmark

Neben der Festlegung der Domäne sind bei einem XML-Datenbankbenchmark weitere Zielsetzungen festzulegen, in denen sich die bisherigen Vorschläge auch unterscheiden:

- *Skopus: Bewertung des gesamten DBS oder einzelner Komponenten*
  Aus Sicht der Nutzer eines XML-DBS ist das Leistungsverhalten des Gesamtsystems ausschlaggebend, das sich aus dem Zusammenwirken aller Komponenten ergibt. Die Bewertung wichtiger Einzelkomponenten, insbesondere des Anfrageprozessors, kann jedoch auch schon aussagegkräftige Leistungskennzahlen ergeben.
- *Ein- vs. Mehrbenutzerbetrieb*
  Datenbanksysteme sind generell für die gleichzeitige Nutzung durch mehrere Benutzer ausgelegt, so dass auch Benchmarks für XML-Datenbanksysteme Leistungsaussagen für den Mehrbenutzerbetrieb liefern sollten. Wenn es jedoch um die Evaluierung einzelner Komponenten geht, z.B. des Anfrageprozessors, so kann die Beschränkung auf den Einzelbenutzerbetrieb sinnvoll sein. In diesem Fall sind nur Antwortzeitbewertungen möglich, jedoch keine Messung der Durchsatzleistung.
- *Operationen-Mix*
  Die Bestimmung der zu messenden Last erfordert die Festlegung einer bestimmten Anzahl von Operationen unterschiedlicher Art und Komplexität. Die Anzahl der Operationen sollte eher gering sein, um einen überschaubaren Vergleich der Systeme zu ermöglichen. Daraus resultieren jedoch komplexe Operationen, da die relevanten Operationstypen durch wenige Operationen realisiert werden müssen. Je mehr Funktionalität in einer Operation verwendet wird, um so geringer ist die Aussagekraft der Laufzeitmessung, da z.B. Rückschlüsse auf die einzelnen Anfrageprozessorkomponenten nicht mehr möglich sind.

3.2.3 Problembereiche von XML-Datenbankbenchmark

Das frühe Entwicklungsstadium der XML-Datenbanksysteme, die unterschiedlichen Architekturen und unterstützten Funktionalitäten können zu Problemen führen, die bei der Spezifikation der Benchmarks beachtet werden müssen.

Das Hauptproblem ist das Fehlen einer standardisierten Anfragesprache. Obwohl mit XQuery eine solche in der Spezifikationsphase ist, wird es noch längere Zeit dauern, bis diese durch den Großteil der

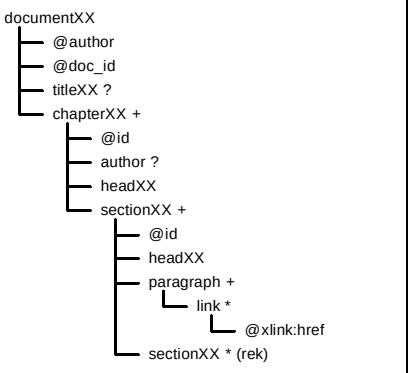

Abb. 1: DOM-Struktur eines Textdokumentes in XMach-1

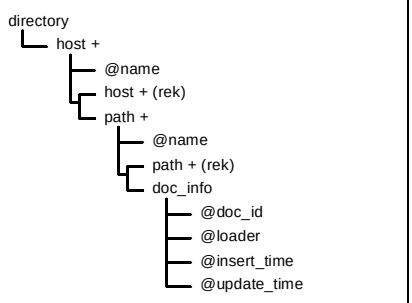

Abb. 2: DOM-Struktur des Metadatendokumentes

XML-Datenbanksysteme unterstützt wird. Als kleinster gemeinsamer Nenner steht in den meisten Fällen XPath 1.0 für Anfrageoperationen zur Verfügung, womit jedoch nur ein Teil der benötigten Operationen umgesetzt werden kann. Auch die proprietären Anfragesprachen der verfügbaren XML-Datenbanksysteme sind nur selten mächtiger als XPath. Komplexere Operationen müssen deshalb durch API-Zugriffe auf die Daten, z.B. mittels DOM, realisiert werden. Bei der Definition eines Benchmarks ist man deshalb gezwungen, die Operationen als beschreibenden Text oder beispielhaft in der aktuellen Entwurfsfassung von XQuery anzugeben.

Gibt es für die Anfragesprache mit XPath wenigstens eine allgemein akzeptierte Grundlage, so existiert für die Datenmanipulation nichts Vergleichbares. Selbst in XQuery sind bisher keine ändernden Operationen vorgesehen. Ein weiteres Problem mit Änderungsoperationen ist, dass einige XML-Datenbanken nicht in der Lage sind, einzelne Fragmente von XML-Daten zu manipulieren. Um Änderungen durchzuführen, müssen dort die Dokumente vollständig ausgelesen, entfernt, geändert und wieder eingefügt werden. Zur Wahrung der Portabilitätseigenschaft ist es deshalb notwendig zu entscheiden, ob der Benchmark Änderungsoperationen enthalten soll und welchen maximalen Umfang die zu ändernden Dokumente besitzen dürfen.

3.3 XML-Benchmarks im Überblick

3.3.1 XMach-1[1]

An der Universität Leipzig wurde im Jahr 2000 mit der Entwicklung von XMach-1 (*X*ML Data *Ma*nagement ben*ch*mark) begonnen, dessen Spezifikation Anfang 2001 veröffentlicht wurde [BöRa01]. Es war der erste XML-Datenbankbenchmark mit Anspruch auf allgemeine Anwendbarkeit. Die wesentlichen Zielsetzungen von XMach-1 sind Skalierbarkeit, Mehrbenutzerbewertung und die Evaluierung des gesamten Datenverwaltungssystems. Im Gegensatz zu den anderen XML-Benchmarks enthält er eine genaue Beschreibung der Testumgebung sowie Vorgaben zur Ausführung der Operationen.

Domäne. Der Benchmark simuliert eine Web-Anwendung zur Verwaltung unterschiedlicher text-orientierter Dokumente. Das Testsystem kann beliebig viele Datenbank- sowie Applikations-Server umfassen, wobei letztere die generischen Anfragen der Klienten in Anfragen auf die Datenbank abbilden und dabei anfallende Mapping- und Transformationsaufgaben übernehmen können. In großen Teilen entspricht dieses Szenario der Domäne ‘Dokumentenkollektion’. Durch das Verwalten von Metadaten der Kollektion in einem strukturierten Dokument enthält der Benchmark jedoch auch Aspekte des Typs ‘Strukturierte Daten’ und ist demzufolge der Domäne ‘Gemischte Daten’ mit Schwerpunkt auf dem dokumentorientierten Aspekt zuzuordnen.

Daten. Der Benchmark enthält zwei Dokumenttypen. Der größte Teil der Daten besteht aus Dokumenten, die in ihrer Struktur und Inhalt Textdokumenten wie Büchern, Aufsätzen o.ä. nachempfunden sind

1. http://dbs.uni-leipzig.de/de/projekte/XML/XmlBenchmarking.html

und damit vom Typ dokumentorientiert sind. In Abb. 1 ist die DOM-Hierarchie der Elemente und Attribute dargestellt. Diese Dokumente werden durch einen parametrisierbaren Generator synthetisch erzeugt. Die Größe der Dokumente variiert zwischen 2 und über 100 Kilobyte mit einem Mittelwert von ca. 16 Kilobyte, so dass typische Dokumentgrößen ohne grafische oder multimediale Inhalte von dem Benchmark abgedeckt werden. Durch geeignete Parametrisierung bei der Generierung der Dokumente werden sowohl flache Dokumente (viele *chapter*-Elemente, wenige geschachtelte *section*-Elementen), als auch tiefe Dokumente (hohe Anzahl geschachtelter *section*-Elemente) erzeugt. Die Skalierbarkeit des Benchmarks bzgl. des Datenvolumens wird durch die Erhöhung der Anzahl der Dokumente in Vielfachen von 10 erreicht.

Den gemischten Charakter erhält der Benchmark durch ein Dokument, das Metadaten über alle enthaltenen Dokumente enthält, wie z.B. URL, Name, Einfüge- und Änderungszeit. Abb. 2 zeigt die DOM-Struktur für dieses Dokument. In ihm sind alle Informationen in den Attributen enthalten und die Reihenfolge von Geschwisterelementen ist beliebig, so dass es vom Typ datenorientiert ist.

Der Benchmark unterstützt sowohl eine schemalose Speicherung der Dokumente als auch eine Variante mit Schemanutzung. Ein wesentliches Merkmal ist dabei die unbegrenzte Zahl von Tag-Namen, um zu testen, wie gut XML-Datenbanksysteme in der Lage sind, viele verschiedene Strukturen bzw. Schemas zu verwalten. Diesem Aspekt wird im Benchmark durch die Verwendung eines Zählers an bestimmten Tag-Namen (siehe Abb. 1) Rechnung getragen. Damit werden pro Schema durchschnittlich 20 Dokumente generiert.

Operationen. XMach-1 definiert 8 Anfrage- und 3 Änderungsoperationen, die in Tabelle 2 aufgelistet sind. Die Operationen reichen vom Abfragen komplexer vollständiger Dokumente (Q1), über Textsuche (Q2) bis hin zu Abfragen mit Sortier- und Verbundoperatoren, d.h. es liegt auch hier eine Mischung von dokument- und datenorientierten Anfragen vor. Die Änderungsoperationen beinhalten das Einfügen und Löschen von Textdokumenten (M1, M2) sowie die Änderung von Attributwerten des Metadatendokuments (M3). Für die Operationen liegt neben der verbalen Beschreibung auch eine Spezifikation in XQuery vor.

ID	Operation	Kommentar
Q1	Liefere das Dokument zu URL X.	Vollständiges Dokument ausgeben. Selektion des Dokumentes mittels Verbund mit Metadatendokument.
Q2	LIefere doc_id der Dokumente, die die Phrase X in einem paragraph-Element enthalten.	Testet Leistungsfähigkeit der Volltextsuche.
Q3	Navigiere beginnend beim ersten chapter-Element über jedes erste section-Element. Gib das letzte section-Element zurück.	Simuliert Navigation über den Dokumentbaum mittels Sequenzoperatoren.
Q4	Liefere eine flache Liste der head-Elemente aller section-Elemente des durch doc_id X selektierten Dokumentes.	Rekonstruktion von Teilen des Dokumentes. Die Erstellung eines Inhaltsverzeichnisses wird simuliert.
Q5	Liefere alle Dokumentnamen (name-Attribut des letzten path-Elementes), die unterhalb einer gegebenen URL liegen.	Hier kann der Einfluss der Elementordnung geprüft werden.
Q6	Finde alle author-Elemente mit Inhalt X und liefere id-Attribut des Elternelementes.	Selektion über Elementinhalt.
Q7	Liefere doc_id von allen Dokumenten, die wenigstens X-mal referenziert werden.	Gruppierung und Zählerfunktionalität werden hier getestet.
Q8	Liefere doc_id von den 100 zuletzt geänderten Dokumenten, die ein author-Attribut besitzen.	Benötigt eine Reihe von datenorientierten Funktionen: Zähler, Sortierung, Verbund, Existenzoperator.
M1	Füge neues Dokument ein.	Testet die Einfügeleistung komplexer Dokumente mit aktiven Indizes.
M2	Lösche Dokument mit doc_id X.	Testet das Löschen eines komplexen Dokumentes mit aktiven Indizes.
M3	Ändere Namen und update_time-Attribut des Dokumentes mit doc_id X.	Testet die Effizienz von Änderungsoperationen auf Attributwerten.

Tabelle 2: Operationen von XMach-1

Die Operationen werden gemeinsam in einem Mix ausgeführt, wobei die prozentualen Anteile der einzelnen Operationen festgelegt sind. Entsprechend der simulierten Domäne besitzt die Anfrage Q1 mit 30% das größte Gewicht. Dagegen kommt den Änderungsoperationen mit teilweise weniger als 1% nur geringe Bedeutung zu. Sie haben jedoch einen nicht unerheblichen Einfluss auf die Ausführung der Anfrageoperationen, da das Cache- und Transaktionsmangement für die Aktualität der Daten sorgen muss.

Metriken. Als Mehrbenutzer-Benchmark steht die Messung des Durchsatzes im Vordergrund. In XMach-1 wird diese in *Xqps (XML queries per second)* gemessen und entspricht der Anzahl der durchgeführten Q1-Anfragen. XMach-1 sieht eine Unterscheidung in eine schemaunterstützte und eine schemalose Variante vor und bietet dafür zwei verschiedene Einheiten an. Neben dem Durchsatz werden auch die Antwortzeiten der einzelnen Operationen gemessen. Durch Vorgeben der maximalen Antwortzeiten für jeweils 90% der Anfragen wird erreicht, dass die Anzahl der Klienten für einen maximalen Durchsatz nicht zu Lasten der Antwortzeiten optimiert werden kann. Auf der anderen Seite wird durch Denkzeiten auf der Seite der Klienten erreicht, dass zur Erhöhung des Durchsatzes die Anzahl der Klienten steigen muss, so dass ein Mehrbenutzerbetrieb erzwungen wird. Die Skalierung auf leistungsfähigere Systeme kann demzufolge sowohl durch die Anzahl der Dokumente als auch durch die Anzahl der Klienten erfolgen.

3.3.2 Xmark[1]

Dieser Benchmark wurde am National Research Institute for Mathematics and Computer Science (CWI) der Niederlande entwickelt und Mitte 2001 in der Version 1.0 veröffentlicht [SWKF01]. Hauptziel des Benchmarks ist die Untersuchung der Performanz des Anfrageprozessors, welches sich in der großen Anzahl der Anfragen und in der Beschränkung auf ein lokales Computersystem (1 Rechner) wiederspiegelt. Die dem Benchmark zugrundeliegenden Richtlinien werden in [SWKF01b] ausführlich diskutiert.

Domäne. Wie schon bei XMach-1 liegt auch diesem Benchmark ein konkretes Anwendungsszenario zugrunde. Modelliert wird die Datenbank eines Internet-Auktionsportals. Diese, als ein großes XML-Dokument gestaltete Datenbank, enthält eine Anzahl von Fakten, die durch ihre feste Struktur im Wesentlichen datenorientierte Eigenschaften aufweisen. Durch die Aufnahme von Beschreibungstexten werden jedoch auch dokumentorientierte Aspekte eingebracht, d.h. auch dieser Benchmark gehört zur Domäne ‘Gemischte Daten’, hier jedoch mit Schwerpunkt auf dem datenorientierten Aspekt.

Daten. Der Benchmark verwaltet alle Daten innerhalb eines einzigen Dokumentes, dessen DOM-Struktur in Abb. 3 wiedergegeben ist. Es enthält die Auktionsdaten als strukturierte Fakten, welche in den Hauptinhalte (Kindelemente des Wurzelelementes) Gegenstände (item) - unterteilt nach Regionen -, Kategorien (categories), Personen (people), offene und geschlossene Auktionen (open_auctions/ closed_auctions) gespeichert sind. Die sich daraus ergebenen Teilbäume sind durch zahlreiche Verweise miteinander verbunden. Die Elementreihenfolge von Geschwisterelementen ist größtenteils frei. Nur bei einzelnen Elementen, wie z.B. der Bieterhistorie (bidder) oder der Beschreibung der Gegenstände (description) ist die Reihenfolge relevant.

Die Größe des Dokumentes kann über einen Skalierungsfaktor von 10 Megabyte bis 10 Gigabyte eingestellt werden. Skaliert wird über die Anzahl von Objekten ausgewählter Mengen, z.B. die Gegenstände und Personen, indem deren Basisanzahl mit dem Skalierungsfaktor multipliziert wird. Dieses hat jedoch zur Folge, dass sich das Verhältnis bestimmter Elemente zueinander ändert.

Das Dokument weist eine recht komplexe, jedoch auch feste Struktur auf. Die Tatsache, dass sich bei der Skalierung zwar die Anzahl der Elemente, jedoch nicht die Anzahl der Elementtypen erhöht, unterstreicht den stark ausgeprägten datenorientierten Skopus des Benchmarks.

Operationen. Entsprechend seiner Ausrichtung auf den Anfrageprozessor definiert Xmark eine große Anzahl von Anfragen (20), die jeweils verschiedene Aspekte der Anfrageverarbeitung testen. Neben der Formulierung der Anfragen in textueller Form liegen diese auch in XQuery-Syntax vor. Die relativ große Zahl der Operationen resultiert auch aus der Aufnahme mehrerer funktional ähnlicher Anfragen. So gibt es z.B. 4 Verbundanfragen, um die Unterstützung bestimmter Optimierungsmöglichkeiten zu bewerten.

1. http://monetdb.cwi.nl/xml/index.html

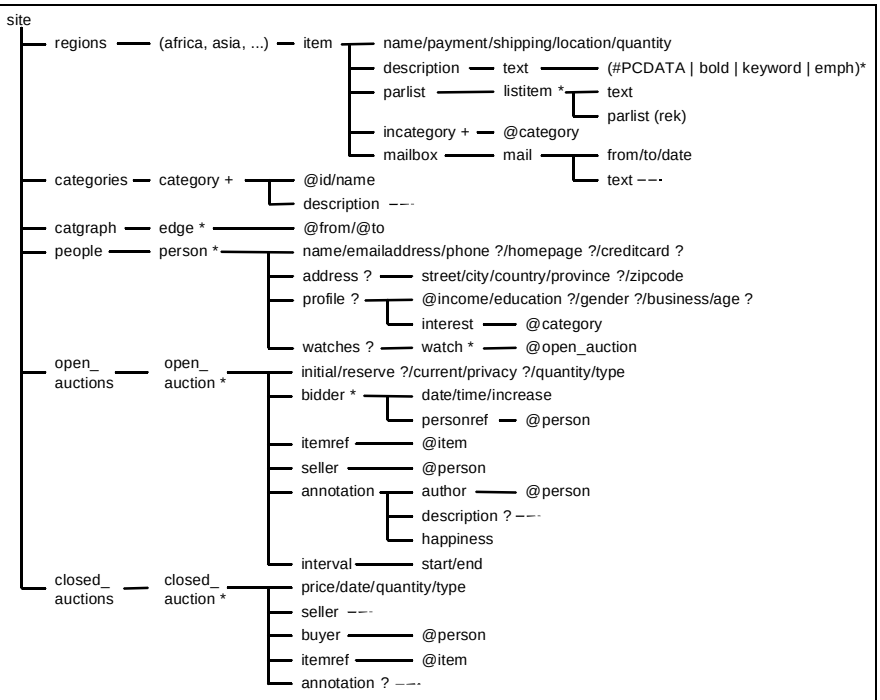

Abb. 3: DOM-Struktur des Xmark-Dokumentes

Einige Anfragen sind nur für kleinere Datenbestände sinnvoll. Der Benchmark enthält 5 Anfragen, die primär zum dokumentorientierten Bereich gehören sowie 4 Anfragen mit Fokus auf den datenorientierten Bereich. Die anderen Anfragen enthalten Aspekte, die für beide Bereiche relevant sind. Da ein entsprechender Standard zur Datenmanipulation noch nicht vorliegt, verzichtet Xmark auf die Evaluierung von Änderungsoperationen.

Metrik. Die Ausführung der Anfragen erfolgt im Einbenutzerbetrieb, wobei jede Operation einzeln betrachtet wird. Als Resultat liefert der Benchmark die Ausführungszeiten für die einzelnen Operationen. Bei Operationen mit sehr kurzen Ausführungszeiten wird ein Wiederholungsfaktor angegeben, der bestimmt, wie oft die Operation nacheinander ausgeführt wird. In diesem Fall enthält die Ausführungszeit alle Wiederholungen. Ein Problem hierbei ist, dass durch ein von einigen Systemen vorgenommenes Caching von Anfrage-Antwort-Paaren die durchschnittliche Antwortzeit vom Ausmaß der Caching-Effekte beeinflusst wird.

3.3.3 XOO7[1]

Ebenfalls Mitte 2001 wurde der Benchmark XOO7 vorgestellt, welcher an der National University of Singapore entwickelt wurde. Im Gegensatz zu den vorangegangenen Benchmarks ist er keine komplette Neuentwicklung sondern basiert auf dem Benchmark OO7 [CaDN93], der zur Evaluierung objektorientierter Datenbanken spezifiziert wurde. Obwohl die Autoren des Benchmarks auf Parallelen des Objektmodells und des XML-Datenmodells hinweisen, zeigt es sich, dass die Datenstruktur und auch die Operationen von OO7 nur einen Teil der Aufgaben eines XML-Datenbanksystems abdecken. Diesen Mangel versuchen die Autoren durch eine Ergänzung der Datenstruktur und zusätzliche Anfragen zu kompensieren.

Domäne. Im Unterschied zu den beiden anderen Benchmarks liegt XOO7 kein konkretes Anwendungsszenario zugrunde. Es werden vielmehr generische Beschreibungen komplex aufgebauter Module mit im Wesentlichen “Teil-von”-Beziehungen modelliert. Dies geschieht in einer festen und regelmäßigen Struktur bei der alle Daten in den Attributen liegen, womit das Dokument datenorientierte Eigenschaften besitzt. Durch die Aufnahme von textuellen Beschreibungen und entsprechenden Operationen darauf, erhält der Benchmark auch dokumentorientierte Aspekte und ist demnach wie schon die vorangegenge-

1. http://www.comp.nus.edu.sg/~ebh/XOO7.html

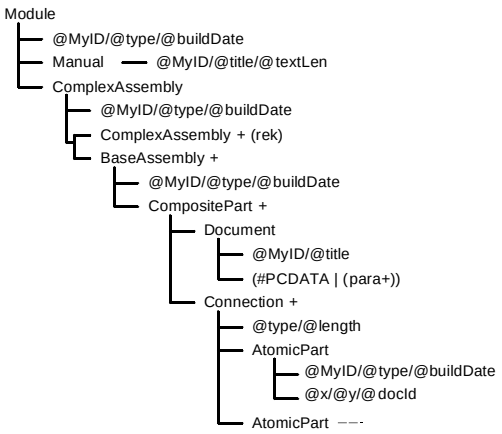

Abb. 4: DOM-Struktur des XOO7-Dokumentes

nen Benchmarks der Domäne ‘Gemischte Daten’ zuzuordnen, wobei der datenorientierte Anteil dominiert.

Daten. Wie in Xmark gibt es auch bei XOO7 nur ein einziges Dokument, welches alle Daten umfasst. In Abb. 4 sind die Elementhierarchie sowie die zugehörigen Attribute dargestellt. Betrachtet man das Schema des zugrundeliegenden objektorientierten Benchmarks, wird dessen Ausrichtung auf Vererbung und Objektreferenzierung ersichtlich. Während das erste Konzept in XML nicht vorhanden ist und deshalb für die Elemente die Attribute dupliziert wurden, wird auch die Möglichkeit der Referenzierung via ID und IDREF nicht genutzt, so dass die einzelnen ‘Objekte’ redundant in den Daten vorliegen. Die enthaltenen Querverweise können somit nicht durch das Datenbanksystem verwaltet werden.

Die Daten können mittels eines Generators erzeugt werden, der durch Vorgabe mehrerer Parameter zur Rekursionstiefe und Anzahl von Kindelementen Dokumente in verschiedenen Größen erzeugen kann. Um eine Vergleichbarkeit von Ergebnissen zu ermöglichen, gibt es Vorgaben zu Parameterkonfigurationen, die als Small, Medium und Large bezeichnet sind und zu Dokumentgrößen von wenigen Megabyte bis zu einem Gigabyte reichen. Die Generierung der Zeichenkettenwerte und Texte basiert dabei auf Zählern oder der Wiederholung von kurzen Zeichenketten, was die Ergebnisse beim Speichern und Anfragen dieser Werte im Vergleich zu realen Anwendungen verzerren kann.

Das den Daten zugrundeliegende Schema ist fest und regelmäßig, d.h. es gibt weder optionale Attribute oder Elemente noch freien Elementinhalt. Diese Eigenschaft kombiniert mit der geringen Anzahl unterschiedlicher Elemente, die auch unabhängig von der Dokumentgröße ist, unterstreicht den datenorientierten Charakter des Benchmarks.

Operationen. Da die 8 Anfragen des OO7-Benchmarks nur unzureichend das typische Anfragespektrum von XML-Anwendungen abdecken - insbesondere fehlen Navigation und dokumentorientierte Operationen - wurden wichtige Operationstypen ergänzt. Insgesamt umfasst der XOO7-Benchmark 18 Anfragen. Wie schon Xmark kennt auch XOO7 keine Operationen zur Datenmanipulation.

Von den Autoren von XOO7 werden die Operationen in die Gruppen ‘Traditionelle Datenbankanfragen’ (9 Operationen), ‘Navigation’ (7 Operationen) und ‘Dokumentanfragen’ (2 Operationen) eingeteilt. Allerdings spielen auch in der ersten Gruppe Navigation und dokumentorientierte Aspekte eine Rolle; umgekehrt zeigen die Anfragen der beiden anderen Gruppen ebenfalls Eigenschaften traditioneller Datenbankabfragen.

Metrik. Analog zu Xmark beschränkt auch XOO7 die Testumgebung auf einen einzelnen Computer. Die Daten werden lokal gespeichert und die Anfragen im Einbenutzerbetrieb ausgeführt. Grundlegende Metrik sind demzufolge die Antwortzeiten der einzelnen Anfragen, die unabhängig voneinander ermittelt werden.

3.4 Vergleich der XML-Benchmarks

In diesem Abschnitt wird ein vergleichender Überblick über die vorgestellten Benchmarks gegeben. Dieser kann als Entscheidungshilfe für die Auswahl eines geeigneten Benchmarks genutzt werden. Die hierbei diskutierten Kriterien sind als Übersicht in Tabelle 3 zusammengefasst.

Allen vorgestellten Benchmarks ist gemein, dass sie ein möglichst breites Spektrum der XML-Datenverarbeitung abdecken wollen, wodurch sie als Ganzes nur bedingt zur Evaluierung der zu erwartenden Leistungsfähigkeit von reinen dokument- oder datenorientierten Anwendungsszenarios geeignet sind. Trotzdem lassen sich bei den Benchmarks eindeutige Tendenzen zu dem einen oder anderen Typ erkennen. XMach-1 mit seinen verschiedenen Schemas und großem Textanteil bedient schwerpunktmäßig den dokumentorientierten Aspekt, während Xmark und XOO7 hauptsächlich datenorientierte Eigenschaften aufweisen.

Der wesentliche Unterschied liegt jedoch im Skopus der Benchmarks. Nur XMach-1 definiert als Ziel die praxisnahe Evaluierung des gesamten Datenbanksystems im Mehrbenutzerbetrieb. Die beiden anderen Benchmarks beschränken sich auf eine ausführliche Evaluierung des Anfrageprozessors, was sich in der Zahl der Anfragen sowie der Beschränkung auf Einbenutzerbetrieb und lokalen Rechner wiederspiegelt.

	XMach-1	**Xmark**	**XOO7**
Domäne	Gemischte Daten	Gemischte Daten	Gemischte Daten
Schwerpunkt	dokumentorientiert	datenorientiert	datenorientiert
Skopus	DBMS	Anfrageprozessor	Anfrageprozessor
# Benutzer	Mehrbenutzer	Einbenutzer	Einbenutzer
# Rechner	≥1	1	1
# Schemas	# Dokumente/20	1	1
# Dokumente	10^n ($n \geq 3$)	1	1
Größe der Daten	$16KB*10^n$	10MB-10GB	ca. 4MB-1GB
# Anfragen	8	20	18
# Änderungsoperationen	3	0	0

Tabelle 3: Benchmarkvergleich

Ein weiterer Unterschied besteht in der Aufteilung der Datenbasis in Dokumente. Während XMach-1 die Datenbank auf viele kleinere Dokumente verteilt, erzeugen Xmark und XOO7 jeweils nur ein einziges Dokument, was bei XML-Datenbanken, die ein Dokument als kleinstes Anfragegranulat behandeln, zu Problemen führen kann. Weiterhin kann damit auch nicht der Einfluss verschiedener Schemas getestet werden.

Die Größe der Datenbank selbst lässt sich in allen Benchmarks über Parameter in weiten Grenzen einstellen. Die in der Tabelle angegebenen Obergrenzen resultieren aus den zur Vergleichbarkeit angegebenen Parameterkonfigurationen und nicht aus einer prinzipiellen Beschränkung der Datengeneratoren.

Als einziger der hier vorgestellten Benchmarks unterstützt XMach-1 auch Datenmanipulationsoperationen. Dies wird durch die Spezifikation der Testumgebung unter Einbeziehung eines Applikationsservers ermöglicht, der einen Ausgleich fehlender oder nicht standardisierter Funktionalität bietet.

Zusammenfassend lässt sich sagen, dass als entscheidendes Kriterium für die Wahl eines der vorgestellten Benchmarks der Skopus betrachtet werden muss. Eine praxisnahe Bewertung des Gesamtsystems ist derzeit nur mit XMach-1 möglich. Sind jedoch Ausführungszeiten und Optimierungsstrategien für einzelne Anfragen das Ziel der Untersuchung, so sind Xmark und XOO7 die erste Wahl. Letztere sind sich in weiten Teilen sehr ähnlich, so dass ein Blick auf die reinen Parameter wenig zur Auswahl beitragen kann. Im Allgemeinen kann festgestellt werden, dass XOO7 mit seinem Rückgriff auf einen objektorientierten Benchmark ein sehr einfaches Datenschema mit wenigen XML-spezifischen Eigenschaften aufweist, so dass in den meisten Fällen Xmark die bessere Wahl darstellen dürfte.

4 Zusammenfassung

Das XML-Format stellt neue Anforderungen an Datenbanksysteme. Um diesen gerecht zu werden wurden neue DBS entwickelt oder herkömmliche DBS erweitert. Im Ergebnis existieren für die XML-Datenverwaltung verschiedene Architekturen mit spezifischen Vor- und Nachteilen. Für deren Evaluierung stellen Benchmarks ein wichtiges Instrument dar. Aufgrund des breiten Anwendungsbereiches von XML muss bei der Wahl des Benchmarks die anvisierte Domäne berücksichtigt werden. Die drei vorgestellten Benchmarks berücksichtigen sowohl daten- als auch dokumentorientierte XML-Daten, jedoch mit unterschiedlicher Schwerpunktsetzung. Xmark und XOO7 konzentrieren sich im Wesentlichen auf die Bewertung des Anfrageprozessors im Einbenutzerbetrieb sowie eine aus einem XML-Dokument bestehende Datenbank. XMach-1 ist für die Bewertung des Gesamtleistungsverhaltens eines XML-DBS konzipiert und definiert mit Xqps (XML Queries per Second) eine Durchsatzmetrik für den Mehrbenutzerbetrieb. Die Anwendung der Benchmarks auf konkrete Systeme wird durch die derzeit noch geringe Funktionalität der Anfragesprachen erschwert.

Literatur

BöRa01 T. Böhme, E. Rahm: XMach-1: *A Benchmark for XML Data Management.* Proc. 9. GI-Fachtagung Datenbanksysteme in Büro, Technik und Wissenschaft, S. 264-273, Springer, Berlin, März 2001

Bour02 R. Bourret: *XML and Databases*. http://www.rpbourret.com/xml/XMLAndDatabases.htm, Februar 2002

Bour02b R. Bourret: *XML Database Products*. http://www.rpbourret.com/xml/XMLDatabaseProds.htm, Mai 2002

CaDN93 M. J. Carey, D. J. DeWitt, J. F. Naughton: *The OO7 Benchmark.* In Proceedings of the ACM SIGMOD International Conference on Management of Data, S.12-21, Juni 1993.

Gray93 J. Gray (Hrsg.): *The Benchmark Handbook for Database and Transaction Processing Systems.* Morgan Kaufmann Publishers, San Mateo, Kalifornien, 1993

MAGQ97 J. McHugh, S. Abiteboul, R. Goldman, D. Quass, J. Widom: *Lore: A Database Management System for Semistructured Data.* SIGMOD Record, Volume 26(3), S. 54-66, September 1997

PGMW95 Y. Papakonstantinou, H. Garcia-Molina, J. Widom: *Object Exchange Across Heterogeneous Information Sources*. In: Proceedings of the Eleventh International Conference on Data Engineering (ICDE), S. 251-260, März 1995

RaVo02 E. Rahm, G. Vossen (Hrsg.): *Web und Datenbanken*. erscheint im dpunkt-Verlag 2002

SWKF01 A. Schmidt, F. Waas, M. L. Kersten, D. Florescu, I. Manolescu, M. J. Carey, R. Busse: *The XML Benchmark Project.* Technical Report INS-R0103, CWI, Amsterdam, Niederlande, April 2001

SWKF01b A. Schmidt, F. Waas, M. L. Kersten, D. Florescu, M. J. Carey, I. Manolescu, R. Busse: *Why And How To Benchmark XML Databases.* SIGMOD Record, Volume 30, Number 3, S. 27-32, September 2001