Pamela Sylvie Nfondja

Konzeptioneller Vergleich von XML-Opensource DBMS

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang Angewandte Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Olaf Zukunft
Zweitgutachter : Prof. Dr. Martin Hübner
Abgegeben am 25.April 2008

Nfondja Pamela Sylvie

Thema der Bachelorarbeit

Konzeptioneller Vergleich von XML-OpenSource DBMS

Stichworte

XML, Berkeley DB XML, eXist, PostgreSQL, Architektur, Plattformunterstützung, Transaktion, Datensicherheit, Speicheransatz, Anfragemöglichkeit

Kurzzusammenfassung

OpenSource Produkte gewinnen zunehmend Akzeptanz in der Industrie. Die XML (Extensible Markup Language) hat sich längst vom Hype des Alleskönners zu einer breit genutzten Grundlage für den elektronischen Datenaustausch (z.B.: mit Webservices) entwickelt. Es stellt sich die Frage, wie XML-Dokumente effizient gespeichert und verwaltet werden können. XML-OpenSource Datenbanken versprechen an dieser Stelle eine Antwort. Die Vielfalt an Produkten auf dem Markt erschwert die Produktwahl. Die Speicherbarkeit in XML-Datenbanken ist a priori unklar. In dieser Bachelorarbeit wird eine Marktanalyse über existierende XML-Datenbanken durchgeführt und auf dieser Grundlage werden im Rahmen eines Auswahlverfahren drei XML-OpenSource DBMS (eXist, Berkeley DB XML und PostgreSQL) ausgewählt, auf bestimmte Kriterien untersucht und verglichen. Am Ende soll der Leser einen Überblick über diese DBMS gewinnen und ihre Vor- und Nachteile erkennen.

Nfondja Pamela Sylvie

Title of the paper

Conceptual comparison of XML-OpenSource DBMS.

Keywords

XML, Berkeley DB XML, eXist, PostgreSQL, architecture, platform Transaction, data security, data storage, query option

Abstract

OpenSource Products are getting more and more accepted by the industry. The Extensible Markup Language (XML) started as a hype of an all round technology but since then it has become a widely used basis for electronic data exchange

The Question is how to store and manage XML documents efficiently. XML databases promise to offer the right answer. However, as the products available on the market are very diverse, it is not easy to choose the right product. In this Bachelor thesis a market analysis is accomplished over existing XML databases and on this basis three XML databases (Berkeley DB XML, eXist, PostgreSQL) are selected, examined and compared on certain criteria thereafter. Finally the reader should have an overview of these DBMS. In addition the reader should be aware of their advantages and disadvantages.

Danksagungen

Ich möchte mich an dieser Stelle bei den Personen herzlich bedanken, die für die Qualität dieser Bachelorarbeit einen wichtigen Beitrag geleistet haben.
Zunächst danke ich Herrn Prof. Dr. Zukunft, der mich geduldig betreut und mit seinen konstruktiven Hinweisen bei der Erstellung meiner Bachelorarbeit sehr geholfen hat.
Meine Kommilitonen Gervais Ngongang und Tuan-Kiet Huynh haben mir sehr zur Seite gestanden und mich moralisch unterstützt und in Diskussionen wertvolle Anregung gegeben.
Weiter möchte ich meinen Dank Herrn Heinz Rüding für das Korrekturlesen aussprechen.
Meinen Geschwistern Adèle, Anastasie und Victorinne bin ich für moralische Unterstützung dankbar.
Zu guter Letzt bleibt mir der Dank an alle Diejenigen, die schon immer an mich und meinen Weg geglaubt haben.

In Hinblick auf eine glückliche Zukunft widme ich diese Arbeit meiner Tochter Océane Nfondja.

INHALTSVERZEICHNIS

KAPITEL 1 EINLEITUNG

Dieses einleitende Kapitel soll dem Leser Ansporn für die Thematik sein und als Übersicht über die ganze Bachelorarbeit dienen. Weiter werden die Fragestellungen vorgestellt, welchen diese Bachelorarbeit nachgeht.
Mit diesem konzeptionellen Vergleich von XML-OpenSource Datenbanken soll ein breites Leserspektrum angesprochen werden, d.h. grundsätzlich jede Person bzw. Applikationsentwickler, der sich für die richtige Wahl eines XML-OpenSource Datenbankprodukts interessiert. Der Vergleich soll den Firmen einen Überblick über die Fähigkeiten der Systeme anbieten. Grundkenntnisse in XML sind keine Bedingung, gutes Grundwissen in der XML-Datenbanktechnologie ist aber für das Verständnis von Vorteil.

1.1 MOTIVATION UND ZIELSETZUNG DER ARBEIT

Durch die weitverbreitete Verwendung von XML in Anwendungen (vor allem in Web-Anwendungen) ist die Nutzung von XML-Datenbanksystemen naheliegend, da somit der Umwandlungsprozess (z.B. von Relational auf XML) unnötig wird.
Neben diesem Vorteil ergeben sich zusätzliche Vorteile durch das XML-Format:

- Da XML-Dokumente einen baumartigen Aufbau haben, können praktisch alle Datenstrukturen mehr oder weniger einfach serialisiert werden.
- Da XML und verwandte Auszeichnungssprachen weit verbreitete, "offene Standards" des W3Cs sind und viele Implementierungen (meist OpenSource) dieser Standards vorhanden sind, ist ein reiches Instrumentarium zur Verarbeitung von XML verfügbar.
- Da XML-Datenbanksysteme meist jünger sind, kann man sich schwer aufgrund mangelnder Erfahrung für eine bestimmte XML-Datenbank entscheiden. Anders ausgedrückt, weiß man nicht genau, welche Produkte im Bereich der XML-Datenbanken und welche Typen von XML-Datenbanken es gibt und welche XML-Datenbanksysteme zu welchem Zweck besser als andere geeignet sind.

Diese Arbeit verfolgt daher das Ziel, ein aktuelles Werk zur Verfügung zu stellen, welches sowohl die nötigen XML-Grundlagen als auch die Theorie der XML-Datenbanken erläutert. Auf diesem theoretischen Fundament werden die aktuellen XML-OpenSource Datenbankprodukte auf dem Markt kritisch betrachtet und verglichen.

1.2 AUFBAU DER ARBEIT

Die vorliegende Arbeit ist in acht Kapitel unterteilt und ist wie folgt aufgebaut:

Kapitel 1 leitet die Arbeit ein, wobei die Motivation sowie die Zielsetzung der Arbeit beleuchtet werden.
Da es in dieser Arbeit hauptsächlich um XML-Datenbanken geht, vermittelt
Kapitel 2 das theoretische Grundwissen über XML und XML -Datenbanktechnologien.
Kapitel 3 Anforderungskatalog: Hier werden die Anforderungen aufgestellt, um geeignete Produkte aus dem gesamten Markt zu identifizieren.
Kapitel 4 befasst sich mit der Marktanalyse. Einige XML-Datenbankprodukte werden ausgewählt und porträtiert.
Kapitel 5 befasst sich mit der praktischen Seite der Arbeit, wobei die ausgesuchten XML-OpenSource Datenbanken untersucht und anhand von bestimmten Merkmalen verglichen werden.
Kapitel 6 Testdurchführung: Mit der eigenen Applikation *Easy XML-Database-Benchmark* werden die Queryprozessoren der ausgewählten XML-Datenbanken getestet und ausgewertet.
Kapitel 7 Abschließende Bewertung: Dieses Kapitel dient dem Bewerten der ausgewählten Produkte hinsichtlich der Anforderungen aus Kapitel 3. Hier werden Teilrangfolgen für die Kriterien Plattformunterstützung, Funktionalität und Performanz aufgestellt, die in einer Abschlussempfehlung münden.
Schließlich werden in Kapitel 8 die Erkenntnisse der Arbeit zusammengestellt. Am Ende wird noch ein Ausblick auf mögliche, weiterführende Arbeiten gegeben.

KAPITEL 2 GRUNDLAGE DER XML-TECHNOLOGIEN

Dieses Kapitel gibt eine Übersicht über XML und befasst sich mit der Funktionsweise von XML und den Technologien, die auf XML basieren und selbst Grundlage für XML-Datenbanken bilden. Die Arten genauso wie die Architektur von XML-Datenbanken werden im Allgemeinen betrachtet.

2.1 XML

2.1.1 XML Grundlagen

Die Extensible Markup Language, kurz XML, ist eine Meta-Sprache zur Beschreibung und Darstellung hierarchisch strukturierter Daten in Form von Textdateien. XML wird bevorzugt für den Austausch von Daten zwischen unterschiedlichen IT-Systemen eingesetzt, speziell über das Internet. Sie ähnelt der Websprache HTML (Hypertext Markup Language). Eine Markup-Sprache bedeutet, dass Inhalte, in so genannten Tags eingeschlossen, gespeichert werden:
z.B.: <*link*>*www.haw-hamburg.de*</*link*>
Die Bezeichnung der Tags sieht dabei eine Beschreibung des Inhalts vor, wodurch der Begriff der "Selbstbeschreibung" von XML geprägt wird.
Unterschied und großer Vorteil von XML gegenüber HTML ist, dass es eine strikte Trennung zwischen Form und Inhalt gibt. Das heißt, die Darstellung der Daten eines XML-Dokuments wird ausschließlich von der verarbeitenden Instanz, einer Applikation oder einem Parser vorgegeben. Vorteilhaft ist des Weiteren, dass es möglich ist, durch Schaffung eigener Grammatiken (daher "extensible", dt.: erweiterbar) auf XML basierende Sprachen zu entwickeln. Durch die Erstellung und Verschachtelung eigener Elemente ist es möglich, jegliche Art von Daten abzubilden.

Namen der Strukturelemente
Die Namen der Strukturelemente (XML-Elemente) für eine XML-Anwendung lassen sich frei wählen. Ein XML-Element kann ganz unterschiedliche Daten enthalten und beschreiben: Meistens Text, aber auch Grafiken oder abstraktes Wissen. Ein Grundgedanke hinter XML ist es, Daten und ihre Repräsentation zu trennen, um Daten beispielsweise einmal als Tabelle und einmal als Grafik auszugeben.

2.1.2 Fachbegriffe

Wohlgeformtheit
Ein XML-Dokument heißt wohlgeformt, wenn es sämtliche XML-Regeln einhält (also keine Grammatik verletzt). Beispielhaft seien hier folgende genannt:
Das Dokument besitzt genau ein Wurzelelement. Alle Elemente mit Inhalt besitzen eine Beginn- und eine End-Kennung (-tag) (z. B. <eintrag>Eintrag 1</eintrag>). Elemente ohne Inhalt können auch in sich geschlossen sein, wenn sie aus nur einer Kennung (tag) bestehen, die mit /> abschließt (z. B. <eintrag/>). Die Beginn- und End-Kennungen (tags) sind ebenentreu-paarig verschachtelt.

Ein Element darf nicht mehrere Attribute mit demselben Namen besitzen.
Die Abbildung 2-1 zeigt ein wohlgeformtes XML-Dokument.

```
<Film>
  <Eckdaten quelle="IMDb" abruf="April 2006"/>
  <Name>
    <Originaltitel sprache="englisch">Mario Puzo's The Godfather</Originaltitel>
    <Deutscher_Titel>Der Pate</Deutscher_Titel>
  </Name>
  <Genre>Drama</Genre>
  <Genre>Krimi</Genre>
  <Erscheinungsjahr>1972</Erscheinungsjahr>
</Film>
```

Abbildung 2- 1 Inhalt eines wohlgeformten XML-Dokuments

Gültigkeit
Soll XML für den Datenaustausch verwendet werden, ist es von Vorteil, wenn das Format mittels einer Grammatik (z. B. einer Dokumenttypdefinition (DTD) oder eines XML-Schemas) definiert ist. Der Standard definiert ein XML-Dokument als gültig, wenn es wohlgeformt ist, den Verweis auf eine Grammatik enthält und das durch die Grammatik beschriebene Format einhält.

2.1.3 XML-Parser

Oft auch als XML-Prozessoren bezeichnet sind XML-Parserprogramme oder Programmteile, die XML-Daten nach bestimmten Kriterien auslesen, interpretieren und ggf. auf Gültigkeit prüfen. Prüft der XML-Parser die Gültigkeit, so ist er ein validierender XML -Parser.
Die nachfolgende Tabelle 2-1 enthält eine Übersicht einiger bekannter XML-Parser.

Name	Programmiersprache	Kommentar
Xerces	C++ und Java	Apache-Projekt
Gnome XML-Parser	C	auch xmllib2
Crimson	Java	Apache-Projekt
XDOM	Delphi / Kylix	Enthält XPath- und DOM-Implementierung
SimpleXML	PHP 5	Enthält XPath- und DOM-Implementierung

Tabelle 2- 1 XML-Parser

2.1.4 Aufbau eines XML -Dokuments

Ein XML-Dokument besitzt je einen physischen und einen logischen Aufbau, die im Folgenden vorgestellt werden.

Physischer Aufbau
Der physische Aufbau wird durch Entitäten gebildet, die hierarchisch geordnet sind. Die Hierarchie kommt dadurch zustande, dass der Inhalt einer Entitätsreferenz selbst weitere Entitätsreferenzen enthalten kann. Diese Struktur ist somit auch ein Baum, wobei die Wurzel des Baumes die sog. Dokument-Entität darstellt. Alle Entitäten sind durch die Dokument-Entität eingeschlossen; sie ist immer der Startpunkt für den XML-Parser. Daher braucht die Dokument-Entität auch keinen Namen zur Identifikation.

Logischer Aufbau
Der logische Aufbau ist eine Baumstruktur und damit hierarchisch strukturiert. Dieser wird durch Markup gebildet und beschrieben. Markup ist ein Sammelbegriff für verschiedene Arten von Auszeichnungen in einem XML -Dokument.
Als Baumknoten gibt es Elemente, deren physische Darstellung erfolgt mittels

- eines passenden Paares aus Start-Tag (<Tag-Name>) und End-Tag (</Tag-Name>) oder
- eines Empty-Element-Tags (<Tag-Name />) erfolgen kann,
- Attributen wie bei einem Start-Tag oder Empty-Element-Tag geschriebene Schlüsselwort-Werte-Paare (Attribut-Name="Attribut-Wert") für Zusatz-Informationen über Elemente (eine Art Meta-Information),
- Verarbeitungsanweisungen (<?Ziel-Name Parameter ?>, engl. Processing Instruction).
- Kommentaren (<!-- Kommentar-Text -->).
- Text, welcher als normaler Text oder in Form eines CDATA-Abschnittes (<![CDATA[beliebiger Text]]>) vorkommen kann.

Zur Spezifikation des logischen Aufbaus

- werden die Dokumenttypdefinitionen(DTD) durch das umfangreichere XML-Schema abgelöst, welches keine Möglichkeit zur Definition von Entitäten, jedoch einen adäquaten Ersatz für Entitäten besitzt.
- schliessen CDATA-Abschnitte reine Zeichendaten ein, d.h. eine Verwechslung der darin enthaltenen Zeichen mit Markup ist ausgeschlossen [HaMe05]. Dies stellt für Autoren der XML-Dokumente eine angenehme Alternative zur mehrfachen Verwendung von Entität -Referenzen dar.

Ein XML-Dokument muss genau ein Element auf der obersten Ebene enthalten. Unterhalb dieses Dokumentelements können weitere Elemente verschachtelt werden.

2.1.5 Klassifikation von XML-Dokumenten

XML-Dokumente lassen sich anhand ihres beabsichtigten Gebrauchs und Strukturierungsgrades in dokumentzentrierte und datenzentrierte Dokumente unterteilen. Mischformen können als semistrukturiert bezeichnet werden.

Dokumentzentriert:
Das Dokument ist an ein Textdokument angelehnt, das für den Leser größtenteils auch ohne die zusätzliche Metainformation verständlich ist. XML-Elemente werden hauptsächlich zur semantischen Markierung von Passagen des Dokuments genutzt.

Aufgrund der schwachen Strukturierung ist eine maschinelle Verarbeitung schwierig. Man erkennt dokumentzentrierte XML-Dokumente daran, dass die Elemente gemischten Inhalt haben. Deswegen ist auch eine horizontale sowie vertikale Navigation notwendig. Bei dokumentzentrierten XML-Dokumenten ist die Struktur von Bedeutung. Das hat zur Folge, dass Änderungsoperationen auch die Struktur des Dokumentes verändern können, nicht nur die eigentlichen Nutzdaten wie bei den datenzentrierten XML-Dokumenten.

Datenzentriert:
Das Dokument ist hauptsächlich für die maschinelle Verarbeitung bestimmt. Es folgt einem Schema, das Entitäten eines Datenmodells definiert, in welcher Beziehung die Entitäten zueinander stehen sowie, welche Attribute die Entitäten haben. Bei datenzentrierten XML-Dokumenten stehen die Daten im Vordergrund. Die Daten (Attributwerte, Elementinhalte) werden gesucht und geändert.
In Abbildung 2-1 ist ein datenzentriertes XML-Dokument dargestellt.

Semistrukturiert:
Semistrukturierte Dokumente stellen eine Art Mischform aus der obengenannten Klassifikation dar. Die Erweiterung des Dokumentinhalts aus Abbildung 2-1 um ein Element „Anmerkung“ enthält ein semistrukturiertes XML-Dokument. Es ist stärker strukturiert, als dokumentzentrierte Dokumente, aber schwächer, als datenzentrierte Dokumente.
Abbildung 2-2 stellt ein semistrukturiertes XML-Dokument dar.

```
<Film>
  <Eckdaten quelle="IMDb" abruf="April 2006"/>
  <Name>
    <Originaltitel sprache="englisch">Mario Puzo's The Godfather</Originaltitel>
    <Deutscher_Titel>Der Pate</Deutscher_Titel>
  </Name>
  <Genre>Drama</Genre>
  <Genre>Krimi</Genre>
  <Erscheinungsjahr>1972</Erscheinungsjahr>
  <Anmerkung>
    Der Film gewann unter anderem <Auszeichnung>3 Oskar</Auszeichnung>, <Auszeichnung>5 Golden Globes
    </Auszeichnung> und den <Auszeichnung>Grammy</Auszeichnung>. Die Premiere fand in Amerika am
    <Erstvorstellung>14.03.1972</Erstvorstellung> statt.
  </Anmerkung>
</Film>
```

Abbildung 2- 2 semistrukturiertes XML-Dokument

Die hier vorgestellte Klassifikation ist von großer Bedeutung, sobald es um die Wahl der optimalen Modellierungsmethode (z.B.: ein datenzentriertes XML-Dokument hat als

Modell das Datenmodell), die Speicherungsweise und die Anfragesprache für XML-Dokumente geht. Es hat zur Folge, dass eine Anfragesprache auch auf die Dokumentstruktur Bezug nimmt. SQL ist zum Beispiel für datenzentrierte XML-Dokumente geeignet. Die Anfragesprache XQuery, die im weiteren Verlauf dieser Bachelorarbeit eine wichtige Rolle spielt, kann bei jeder Klasse von XML-Dokumenten eingesetzt werden (siehe Kapitel 2.2.3.2).

2.1.6 Verarbeitung von XML

Programmtechnischer Zugriff auf XML-Dokumente
Das Einlesen von XML-Dokumenten erfolgt auf unterster Ebene über eine spezielle Programmkomponente, einen XML-Prozessor, auch XML-Parser genannt. Er stellt ein API zur Verfügung, über das die Anwendung auf das XML-Dokument zugreift. Viele XML-Prozessoren stellen eine oder mehrere der folgenden APIs bereit:

DOM: Ein DOM-API repräsentiert ein XML-Dokument als Baumstruktur und gewährt wahlfreien Zugriff auf die einzelnen Bestandteile der Baumstruktur. DOM erlaubt außer dem Lesen von XML-Dokumenten auch die Manipulation der Baumstruktur und das Zurückschreiben der Baumstruktur in ein XML-Dokument. Aus diesem Grund ist DOM sehr speicherintensiv. Die bereitgestellten Methoden in DOM sind beispielsweise *firstChild*, removeChild, insertBefor
DOM ist gemäss [HaMe05] seit der Veröffentlichung in einigen Stufen funktional erweitert und in neuen Versionen, *Levels* genannt, veröffentlicht worden. Aktuell liegt DOM Level 3 als Empfehlung des W3C vor und enthält nun unter anderem auch die Unterstützung für XPath-Ausdrücke.
Das Funktionsprinzip von DOM geht gemäss [HaMe05] weiter als SAX: Durch Analyse der Ergebnisse aus dem erfolgreich abgeschlossenen Parsen kann DOM ein Gesamtbild in einer Graphstruktur des XML-Dokuments erstellen. Dieses Gesamtbild hat nach [KlMe03] eine Baumstruktur. Daher zählt man DOM zu den baumorientierten XML-Technologien [HaMe05].

SAX: Ein SAX-API repräsentiert ein XML-Dokument als sequentiellen Datenstrom und ruft für die im Standard definierten Ereignisse vorgegebene Rückruffunktionen (callback function) auf. Der Vorteil gegenüber DOM sind der geringe Speicherbedarf und die bessere Verarbeitungsgeschwindigkeit. Eine Anwendung, die SAX nutzt, kann eigene Unterprogramme als Rückruffunktionen registrieren und auf diese Weise die XML-Daten auswerten.

StAX: Die Streaming API for XML ist eine weitere API, um XML Dokumente einzulesen. Sie ist hoch speichereffizient (besonders im Vergleich zu DOM) und gleichzeitig einfach zu programmieren.

Data Binding: Diese Möglichkeit stellt XML-Daten als Datenstruktur direkt für einen Programmzugriff bereit. Die XML-Daten werden per Unmarshalling z.B. direkt in Objekte gewandelt.

Oftmals greift der Anwendungscode nicht direkt auf die Parser-API zu. Stattdessen wird XML weiter gekapselt, so dass der Anwendungscode mit nativen Objekten bzw. Datenstrukturen arbeitet, welche sich auf XML stützen. Beispiele für solche Zugriffsschichten sind JAXB in Java, der Data Binding Wizard in Delphi oder das XML Schema Definition Toolkit in .NET. Die Umwandlung von Objekten in XML ist üblicherweise bidirektional möglich. Diese Umwandlung wird als Serialisierung oder Marshalling bezeichnet.

Transformation und Darstellung von XML-Dokumenten
Ein XML-Dokument kann mittels geeigneter Transformationssprachen wie XSLT oder DSSSL in ein anderes Dokument transformiert werden. Oftmals dient die Transformation zur Überführung eines Dokuments aus einer XML-Sprache in eine andere XML-Sprache, beispielsweise zur Transformation nach XHTML, um das Dokument in einem Webbrowser anzuzeigen.

Schemasprachen
Um die Struktur von XML-Dokument zu beschreiben, bedient man sich so genannter Schemasprachen. Die zwei bekanntesten sind DTD und XML-Schema.

DTD
Eine DTD (Dokumenttypdefinition) ist eine Beschreibung eines XML-Dokuments. Sie wurde zusammen mit XML zu einem Zeitpunkt standardisiert, an dem XML noch hauptsächlich für „narrative documents“ („erzählende Dokumente“, also Zeitungsartikel, Bücher) gedacht war, weniger als Datenaustauschformat. Daher ist es z.B. in DTD nicht möglich, zwischen Texten und Zahlen zu unterscheiden. Ein weiterer Nachteil ist die Tatsache, dass die DTD in einer eigenen Sprache abgefasst werden muss. Außerdem kennt die DTD keine Namensräume.

XML-Schema / XSD
XML-Schema (bzw. XSD = XML-Schema-Definition) ist die moderne Möglichkeit, die Struktur von XML-Dokumenten zu beschreiben. XML-Schema bietet auch die Möglichkeit, den Inhalt von Elementen und Attributen zu beschränken, z.B. auf Zahlen, Datumsangaben oder Texte, z.B. mittels regulärer Ausdrücke. Ein Schema ist selbst ein XML-Dokument, welches erlaubt, komplexere (auch inhaltliche) Zusammenhänge zu beschreiben, als dies mit einer formalen DTD möglich ist.

2.1.7 XML-Infrastruktur

Im Zusammenhang mit XML wurden vom W3-Konsortium auf Basis von XML viele Sprachen definiert, welche XML-Ausdrücke für häufig benötigte allgemeine Funktionen anbieten, wie etwa die Verknüpfung von XML-Dokumenten.
Zahlreiche XML-Sprachen nutzen diese Grundbausteine:

Transformation von XML-Dokumenten	XSLT
Adressierung von Teilen eines XML-Baumes	XPath
Verknüpfung von XML-Ressourcen	XPointer, XLink und XInclude
Selektion von Daten aus einem XML-Datensatz	XQuery
Datenmanipulation in einem XML-Datensatz	XUpdate
Abfassen von elektronischen Formularen	XForms
Definition von XML-Datenstrukturen	XML Schema bzw. XSD, XML Schema Definition Language
Signatur und Verschlüsselung von XML-Knoten	XML Signature und XML-Encryption
Aussagen zum formellen Informationsgehalt	XML Infoset
PDF-Generierung aus XML-Daten	XSL-FO
Definition zum Methoden- bzw. Funktionsaufruf durch verteilte Systeme	XML-RPC
Standardisierte Attribute	XML Base und xml:id

Tabelle 2- 2 Bausteine für XML-Sprache

2.2 GRUNDLAGEN ZU XML- DATENBANKEN

Mit diesem Abschnitt wird die bis jetzt betrachtete XML-Theorie abgeschlossen. Es geht nun um XML-Datenbanken selbst, welche im Weiteren theoretisch betrachtet werden.

2.2.1 Definition

XML-Datenbanken sind Datenbanksysteme zur effizienten Speicherung von XML-Dokumenten bzw. XML-Daten als Ganzes oder durch Mapping von XML-Dokumenten in Tabellen. Sie geben die Möglichkeit des Zugriffs auf die gespeicherten XML-Dokumente durch Anfragesprache.

2.2.2 Arten und Architekturen

Seit Einführung von XML im Bereich der Informationsverarbeitung werden immer wieder Fragen über die Speicherung und Verwaltung von XML-Daten gestellt.
Um XML-Daten und -Dokumente zu speichern bzw. zu verwalten, gibt es verschiedene Ansätze, die im Folgenden näher dargestellt werden. Bezüglich der Architekturen, kann man die Architektur von XML-enabled Datenbanken genauso wie native XML-Datenbanken in drei Ebenen einteilen: Die konzeptuelle, die logische und die physische Ebene.
Die Abbildung 2-3 stellt die allgemeine Architektur einer XML-Datenbanken dar.

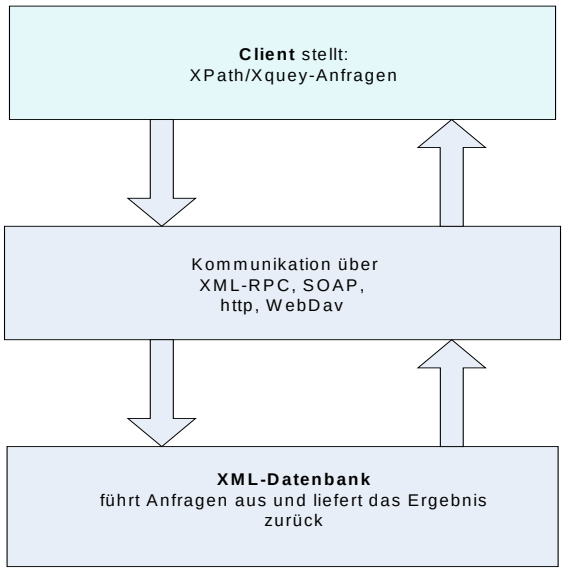


Abbildung 2- 3 Architektur einer XML-Datenbank

2.2.2.1 XML-enabled Datenbanken

Die XML-enabled Datenbank hat als deutschen Begriff „XML-fähige Datenbank". In solchen Datenbanken geht es um die Unterstützung von XML-Dokumenten bzw.
-Daten in bestehenden Datenbanken. Sie werden dadurch charakterisiert, dass sie intern kein XML-Datenmodell verwenden. Sie haben ihr eigenes Datenmodell und speichern die enthaltenen Daten in einem XML-Dokument. Die Daten liegen nicht mehr als XML vor. Beim Mapping gehen Teile der XML-Dokument-Informationstruktur verloren, welche im internen Datenmodell keinen Bezug hat. Die gespeicherten Daten lassen sich trotzdem als XML-Dokument extrahieren. In Abbildung 2-4 wird das Prinzip des Speicheransatzes in einer XML-enabled Datenbank dargestellt. Bei der Speicherform wird nur das relationale Datenmodell verwendet, auch wenn ursprünglich mit dem XML-Dokument und seiner selbstbeschreibenden Eigenschaft ein XML-Datenmodell vorlag.
Eine XML-enabled Datenbank bringt laut [SBC+04]folgende Eigenschaften mit:

- Das Schema des XML-Dokuments muss auf das interne Schema der XML-fähigen Datenbank abgebildet werden, damit die Daten des XML-Dokuments in der Datenbank gespeichert werden können,
- Die Anfrage und Daten-Manipulation sind keine „reine" XML-Technologie.
- Anstatt des XML-Datenmodells wird intern häufig das relationale Datenmodell verwendet,
- Das Datenbankschema beschreibt die Daten von XML-Dokumenten und nicht das XML-Dokument selbst.

Im Falle, dass das Schema des XML-Dokuments auf das Schema einer relationalen Datenbank abgebildet wird, wird der Prozess als Schreddern bezeichnet.
In Abbildung 2-4 beobachtet man, dass die Struktur des ursprünglichen XML-Dokuments (links) nicht mehr dem Ausgangsdokument (rechts) entspricht.

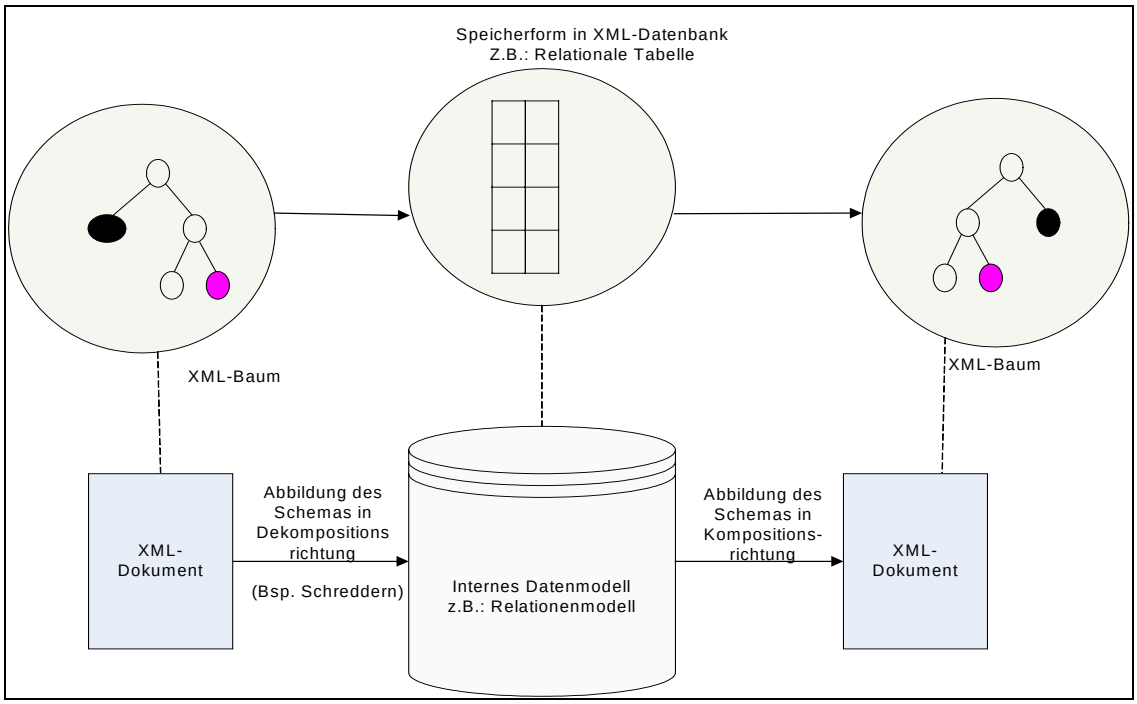


Abbildung 2- 4 Funktionsweise einer XML-fähigen Datenbank

2.2.2.2 Native XML-Datenbanken

Native XML-DBMS sind DBMS, die dazu dienen, XML-Dokumente in ihrem echten Format (nativ) zu speichern und zu verarbeiten. Nativ heißt in diesem Sinne, dass die Daten nicht in relationale Einzelteile zerlegt werden, sondern so, wie man sie an das System übermittelt, auch speichert.
Native XML-Datenbanken speichern XML-Dokumente oder Fragmente im Ganzen und führen Dokumente gleichen Typs in so genannten Kollektionen zu einer Datenbasis zusammen. Dabei bleibt die Dokumentordnung eines gespeicherten Dokuments erhalten und kann somit vollständig wieder rekonstruiert werden. Sie bieten eine effiziente Indizierung von XML-Daten, damit die Suchanfrage nicht über Volltextanalyse, sondern direkt über indizierte Bäume realisiert werden kann.
Eine XML-Datenbank soll folgende Bedingungen erfüllen, um nativ zu sein:

- Die Identität des XML-Dokuments bleibt gewahrt, d.h. es kann nach der Speicherung aus der Datenbank heraus rekonstruiert werden [SBC+04].
- Intern wird ein XML-Datenmodell verwendet, d.h. es ist beschreibbar durch einen XML-Baum [SBC+04].

- Mindestens eine Programmierschnittstelle (API) der XML-Datenbanken wird unterstützt.
- Das Datenbankschema beschreibt XML-Dokumente [SBC+04]
- Mindestens eine Anfragesprache unter den „reinen" XML-Technologien wird unterstützt.

Native XML-Datenbanken haben viele Möglichkeiten, XML-Dokumente zu speichern. Die Abbildung 2-5 zeigt die abstrakte Architektur einer nativen XML-Datenbank angelehnt an [VaCM05].

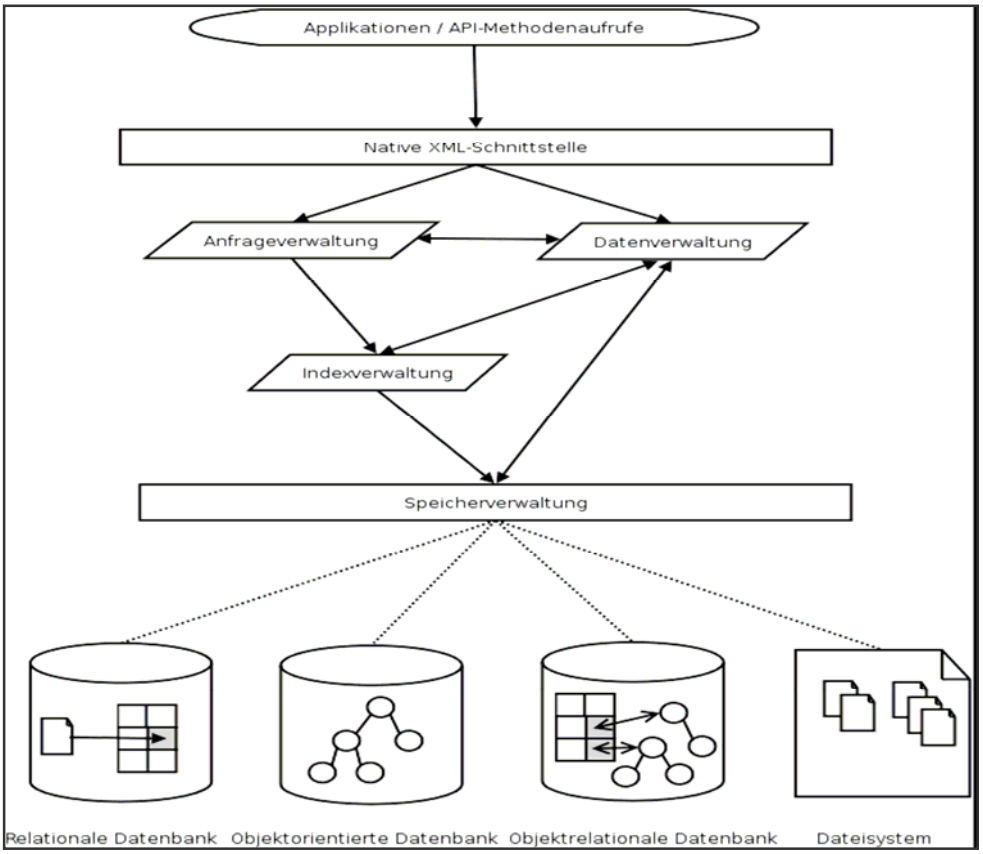


Abbildung 2- 5 Abstrakte Architektur einer nativen XML-Datenbank

Wie in Abbildung 2-5 zu sehen ist, hat eine native XML-Datenbank vier Speicheransatzmöglichkeiten:

- Speicherung des ganzen XML-Dokuments in Kollektionen im Dateisystem oder anderen Speicherstrukturen
- Speicherung der Dokumentstruktur in einer objektorientierten Datenbank Speicherung der Strukturkomponenten eines XML-Dokuments in einer objektrelationalen Datenbank
- Speicherung des XML-Dokuments in einer relationalen Datenbank mit den Datentypen VARCHAR, CLOB oder BLOB

Speicheransatz bei XML-Datenbanken

- Modellbasierter Speicheransatz
  Bei diesem Speicheransatz geht es darum, die Dokument -Struktur eines XML-Dokuments in einem Objektmodell zu speichern. Dieser Ansatz wird auch als Sicherung auf *Knotenstufe* (engl. node-level) bezeichnet.
- Textbasierter Speicheransatz
  Der textbasierte native Speicheransatz speichert die XML-Dokumente insgesamt als Text. Daher wird dieser Ansatz auch als Speicherung auf *Dokumentstufe* (engl. document-level) bezeichnet. Dazu können die XML-Dokumente als Textdateien im Dateisystem oder als CLOB (character large object) in ein Datenbanksystem abgelegt werden. Die Speicherform mit Textdateien lässt sich als die einfachste native Behandlung von XML-Dokumenten betrachten.
- Hybrider Speicheransatz
  Diese Speicherweise verwendet sowohl textbasierte als auch modellbasierte Speicherung. Das XML-Dokument lässt sich auch bei hybriden Ansätzen in jedem Fall auf native Weise speichern.

2.2.3 Anfragesprachen und Datenmanipulationssprache

Dieser Abschnitt soll sich mit der Anfrage an XML-Dokumente befassen.
Eine Anfragesprache ist ein Ausdruck, um Informationen aus der Datenbank abzurufen. Um relevante Daten aus der Datenbank zu erhalten, werden Kriterien mit Hilfe einer Anfragesprache formuliert. Unter Datenmanipulationssprache versteht man das Ändern und das Löschen von Daten sowie Hinzufügen neuer Informationen zu existierenden Daten. Eine Anfragesprache hat bestimmte Anforderungen zu erfüllen:

Allgemeine Anforderungen
Anfragen an eine Datenbank sollten vom Benutzer beispielsweise adhoc und ohne Kenntnisse des kompletten Programms formuliert werden können. Die Verarbeitung der Anfragen sollte effizient und optimierbar sein

Erweiterte Anforderung an eine XML-Anfragesprache
Bei der Verarbeitung von XML sollen spezielle Anforderungen an die Anfragesprache gestellt werden.
Es soll möglich sein, XML-Fragmente innerhalb von Anfragen zu notieren. Es muss auch möglich sein, Dokumente, die gespeichert sind, in der ursprünglichen Dokumentordnung auszugeben.

Grundoperationen einer Anfragesprache
Die Grundoperationen einer Anfragesprache sind in Tabelle 2-3 zusammengefasst. Damit die Sprache vollständig ist, muss sie mit einer minimalen Menge an Operationen beginnen.

Grundoperation	Erklärung
Selektion	Es sollen nicht nur Bedingungen an Inhalt und Wert gestellt werden (Wertselektion), sondern auch Strukturselektionen, die Dokumentstrukturen erstellt.
Extraktion/Reduktion (Projektion)	Subelemente eines Dokuments sollten herauslösbar oder ausblendbar sein
Kombination (join)/Verbund	Mehrere Elemente aus unterschiedlichen Quellen sollen in einem neuen Dokument zusammengefasst werden können.
Restrukturierung	Restrukturierung von Dokumenten, auch Bildung neuer Elemente mit eigenen Strukturen soll möglich sein.
Aggregation/Gruppierung	Die Zusammenfassung von Werten aus unterschiedlichen Elementen soll möglich sein.

Tabelle 2- 3 Grundoperationen einer Anfragesprache

Tabelle 2-4 gibt eine Übersicht über die Anfrage- und Datenmanipulationssprachen für XML-Daten

	Anfragesprache	Datenmanipulationssprache
XML Query Language (XQuery)	☑	●
XML Path Language (XPtah)	☑	☒
XML Update Language (XUpdate)	☒	☑
SQL	☑	☑
SQL/XML	☑	●

Tabelle 2- 4 XML-Technologie für das Anfragen und die Manipulation von XML-Daten

☒ ist nicht unterstützt.
☑ ist unterstützt.
● zum Teil erfüllt.

2.2.3.1 XPath

XML Path Language (XPath) ist keine Anfragesprache, da nur bestimmte Teile innerhalb eines XML-Dokuments identifiziert werden können. [Brun04] und [MeBu06] bezeichnen XPath als Anfragesprache zur Adressierung von Teilen eines XML-Dokuments. XPath setzt eine Pfadnotation ein. Die Basis der XPath Syntax ist ähnlich der Adressierung in Dateisystemen von Linux . Mit dieser Adressierung kann innerhalb eines XML-Baums, der

auf dem XPath Datenmodell basiert, navigiert werden. XPath liegt zurzeit in Version 2.0 als eine Empfehlung des W3Cs vor.
Das XPath-Datenmodell beschreibt die Baumstruktur eines XML-Dokuments. Laut [KlMe03] kennt die Baumstruktur sieben Knotenarten: Wurzel, Elemente, Attribute, Text, Kommentar, Verarbeitungshinweise und Namensraum. Der Wurzelknoten "/" ist nicht der Knoten des Wurzelelements, sondern der abstrakte Ursprungsknoten, der den gesamten Dokumentinhalt umfasst.

Pfadnotation und Schritte
Die Pfadnotation ist ein XPath-Ausdruck, der dazu dient, bestimmte Teile eines Dokuments zu adressieren. Laut [KlMe03] gibt es relative (d.h. von einem beliebigen Kontextknoten eines XML-Baums ausgehende) und absolute Pfade (d.h. von dem Wurzelknoten ausgehende). Der Pfad kann aus mehreren Teilausdrücken „/", die man Lokalisierungsschritte nennt, bestehen. Die Lokalisierungsschritte enthalten eine Achse, einen Knotentest oder optional ein Prädikat.

Achse
Die Achse ist eine Angabe darüber, welche Knotenmengen selektiert werden.
XPath unterscheidet 13 Achsen, die an dieser Stelle nicht alle näher vorgestellt werden. Für eine vollständige Darstellung, siehe [KlMe, Seite 260].
Im Kontext zum aktuell ausgewählten Knoten wäre zum Beispiel:

self (der Knoten selbst)	*ancestor* (übergeordnete Knoten/alle Vorfahren)
*descendan*t (alle Nachkommen)	*parent* (unmittelbar übergeordnete Knoten/ Elternknoten)
Attribut	Attributknoten eines Elements

In Tabelle 2-5 werden, angelehnt an [KlMe03], die wichtigsten Abkürzungen für Achsenangaben zusammengefasst.

Kürzel	Standard
.	Der aktuelle Knoten
..	Der Elternknoten
/	Der Wurzelknoten, oder Pfadtrennzeichen
//	Nachkommen des Kontextknoten oder aktueller Knoten
@	Attribute des aktuellen Knotens
*	Platzhalter für Knoten mit beliebigem Namen (außer Namensraum- und Attributknoten)
@*	Alle Attribute des aktuellen Knotens
[expr]	Prädikat, *expr* ist ein boolescher Ausdruck
[n]	Prädikat, *n* ist eine natürliche Zahl und bezeichnet das *n*-te Element aus der Knotenmenge

Tabelle 2- 5 Abkürzungen der gängigsten Achsennotationen

Knotentests
Mit dem Knotentest kann aus der ausgewählten Achse überprüft werden, welche Unterknoten enthalten sind. Die möglichen Knotentest-Syntax laut [KlMe03] sind in der folgenden Tabelle dargestellt.

Syntax	Bedeutung
Comment()	Liefert alle Kommentarknoten zurück
Node()	Gibt alle Knotentypen zurück (die sieben Knotentypen in XPath werden hier angesprochen)
*	Gibt alle Elementknoten zurück
Text()	Liefert alle Textknoten zurück
Processing-instruction ([name])	Gibt alle processing-instruktion Knoten zurück, optional mit Angabe eines Namens

Tabelle 2- 6 Knotentest-Syntax

Prädikat
Die Prädikate werden eingesetzt, um selektierte Bereiche weiter einzuschränken. Für die Einschränkungen werden Operatoren verwendet, z.B.: (>, <, =, mod, div, und, |). Ein Prädikat kann auch ein XPath-Index enthalten, der einer Knotenposition im XML-Baum entspricht. Die Prädikate werden in eckige Klammern eingeschlossen und haben als Ergebnis einen booleschen Wert (false oder true).

2.2.3.2 XQuery 1.0

Laut [KlMe03] ist die XML Query Language (XQuery) aus der Anfragesprache Quilt hervorgegangen. XQuery bezeichnet eine von dem W3C empfohlene Anfragesprache für XML-Datenbanken und wurde zuletzt am 23. Januar 2007 überarbeitet. XPath 2.0 ist eine

Teilmenge von XQuery, d.h. bei der Adressierung der Knotenmenge verwendet XQuery die Pfadnotation von XPath.

FLWOR-Ausdrücke

Die FLWOR-Ausdrüke bilden die Grundstruktur der XQuery-Anfragen. FLWOR[2] ist das Kurzwort für die fünf Klauseln in XQuery: *for, let, where, order by, return*. In manchen Literaturen wird das Kürzel FLW verwendet, weil der Ausdruck *order by* eine neue Klausel ist.

Ein FLOWR-Ausdruck besteht mindesten aus einer *for-* oder *let*-Klausel und immer aus einer *return*-Klausel. Die beiden Klauseln *where* und *order by* sind optional. Es gibt eine Abfolge der Anweisungen, die folgender Struktur genügen müssen:

{For[|]let}[[where][order by]]{return}.

Beide Klauseln *for* und *let* haben denselben Zweck, Variable zu einem Wert zu binden. Es besteht ein Unterschied bei der Funktionsweise von *let-* und *for-* Klauseln:

- *for $var in –Pfad-*

Mit dieser Schleife wird über die notierte Knotenmenge iteriert. Der Variablen $var wird in jedem Durchgang die Knotenmenge zugewiesen, über die iteriert wird.

- *let $var := -Wert|Pfad-*

Das Anweisungselement *let* bindet eine Variable mit einem Wert oder einem Pfad.

Die Aufgaben von *where-, order by-* und *return*-Klauseln sind folgende:

- *where –Bedingung-*

  Die *where*-Klausel spielt die Rolle eines Filters, d.h. sie schränkt die selektierte Knotenmenge weiter ein. Bei der Prüfung wird dieses aus der Knotenmenge verworfen, wenn der Ausdruck für ein Element aus der Knotenmenge fehlschlägt.

- *order by –Pfad- ascending|descending*

  Die order by*-Klausel* sortiert die angegebenen Knotenmengen. Es lässt sich dabei auch optional spezifizieren, ob auf- oder absteigend sortiert wird.

- *return –Konstruktor-*

  Die letzte Anweisung in einem FLWOR-Ausdruck ist *return.* Diese Anweisung gibt für jedes Element aus der Knotenmenge, abhängig von der Konstruktion des Rückgabewertes, ein Ergebnis zurück.

Funktionen und Operatoren

XQuery enthält eine Vielzahl von Operatoren und Funktionen, die aus SQL bekannt sind. Hier sind einige Funktionen (*count, empty, distinct,..*) und Operatoren (*union, intersect, except...*), die für die Verarbeitung von XML ausgelegt wurden. In XQuery-Anfragen sind bedingte Ausdrücke möglich, z.B.: „*if-then-else*“.

[2] wird wie das englische Wort Flower ausgesprochen

2.2.3.3 XUpdate

XUpdate verwendet zwar XPath. Die Sprache erlaubt es aber gemäß [Brun04], Knoten hinzuzufügen (mit *xu:append* als Kindelement, bei *xu:insert-before* und *xu:insert-after* auf gleicher Ebene), zu verändern (*xu:update*), zu entfernen (*xu:remove*) und umzubenennen (xu:rename).

XQuery Update Facility
In [Brun04] wird mit XQuery DML (Data Manipulation Language) ein potentieller Kandidat vorgestellt, der die bestehende Technologie XQuery um die Schlüsselworte *update*, *delete* und *insert* erweitert. Mit diesen Funktionen könnten XML-Dokumente modifiziert werden, auch auf bedingter Basis mit gängigen Konstrukten wie *if-then-else*-Blöcken. Tatsächlich greift eine aktuelle Entwicklung des W3C diesen Ansatz auf. Es handelt sich um eine XML-Technologie, die XQuery um Datenmanipulationskonstrukte erweitert und sich „XQuery Update Facility“ nennt.

2.2.4 Zugriff auf XML-Datenbanken/ API's für XML-Datenbanken

Die Application Programming Interfaces (API) wurden entwickelt, um Zugriff auf den Datenspeicher zu ermöglichen. Viele Komponenten sind dabei vorgefertigt und werden den Entwicklern zur Verfügung gestellt. In diesen Komponenten befinden sich in aller Regel auch integrierte Parser, die XML-Daten analysieren können
Im Folgenden werden verschiedene Möglichkeiten dargestellt, auf XML-Datenbanksysteme zuzugreifen. Dabei gilt zu beachten: Nicht jede Datenbank unterstützt jede Schnittstelle und nicht jede Programmiersprache hat schon Methoden implementiert, um die entsprechenden Schnittstellen zu nutzen.

HTTP-REST
Der Begriff REST steht für *Representational State Transfer*. Vereinfacht ausgedrückt bedeutet REST, dass Abbilder einer Ressource (z.B. eine XML-Datei) auf einen Client übertragen werden und diesen dadurch in einen bestimmten Status versetzen.
Für XML-Datenbanken kann man neben der Möglichkeit, Ressourcen abzurufen (GET), auch komplexe Anfragen über POST absetzen und mittels PUT ganze Dokumente unter einer Ressource abspeichern. Sogar für das Löschen von Ressource gibt es mit DELETE eine entsprechende Möglichkeit.
HTTP-REST nennt sich das ganze Verfahren, weil es über das Hypertext Transfer Protocol (HTTP) abgewickelt wird. Die verschiedenen DBMS implementieren dafür Erweiterungen über bestimmte Parameter in der URL. Dabei kommt bei den meisten Systemen eine bestimmte, für REST-Zugriffe spezifizierte URL zum Einsatz.
Bei eXist wäre dies z.B.:

http://localhost:8080[1]/exist/rest[2]/db/col[3]

Zur Erläuterung:
1= Adresse (IP oder Domainname) und Port des Servers
2= Pfad zur Rest-Schnittstelle des Servers
3= Basispfad der Datenbank und der Kollektion, die angesprochen werden sollen

Ein Vorteil dieser Methode ist die Möglichkeit, die Schnittstelle auch über einen Proxy zu erreichen. Dem gegenüber steht als Nachteil die Sicherheit sensibler Daten, die auf diesem Weg nicht problemlos möglich ist. Sowohl Client als auch Server müssen das Protokoll HTTPS unterstützten, um die Daten mittels Secure Sockets Layer (SSL) auszutauschen.

XMLRPC
Um von einem Client auf einen entfernten Server zuzugreifen, gibt es verschiedene Möglichkeiten. Eine davon ist das klassische Remote Procedure Call, also der entfernte Methodenaufruf. Während die klassischen RPC direkt über Sockets ablaufen und TCP oder UDP als Protokoll verwenden, werden XMLRPC-Aufrufe über http abgewickelt. Im Unterschied zum ausschließlich auf http basierenden REST-Verfahren wird hier http nur für die Authentifizierung und den Transport verwendet.
Um auf den Server zuzugreifen, muss der XML-RPC API der Programmiersprache die Adresse des Serverdienstes bekannt gegeben werden.
Für die Datenbank eXist könnte eine solche Adresse folgendermaßen aussehen:

xmldb[1]: exist[2]://localhost:8080[3]/exist/xmlrpc[4]/db/col[5]

Erläuterung:
1 = Es handelt sich um eine XML-Datenbank-Adresse
2 = Der Name der Datenbankverbindung ist exist
3 = Adresse (IP oder Domainname) und Port des Servers
4 = Der Pfad der XMLRPC-Schnittstelle
5 = Der Basispfad der Datenbank und die Kollektion, die angesprochen werden sollen

XML:DB API
Diese standardisierte Zugriffsmethode wurde im Jahr 2000 von der XML:DB Initiative verabschiedet und hat sich bis heute gehalten. Die Schnittstelle XML:DB basiert auf dem Zugriffsverfahren XMLRPC. Der Zugriff funktioniert dabei nach einem logischen Ablaufschema, bei dem Basiselemente im Zusammenspiel den Zugriff ermöglichen. Es gibt Datenbanktreiber, Collections, Ressourcen und Dienste. Der Client initialisiert zuerst über den zum Datenbanksystem gehörenden Treiber einen Datenbankmanager und übergibt diesem eine spezielle XMLRPC-Adresse. Im zweiten Schritt wird die Collection, auf der gearbeitet werden soll, initialisiert. Von dieser kann im nächsten Schritt der benötigte Zugriffsdienst geladen werden.
Beispiele sind der XQueryService oder der XupdateQueryService. Diese Klasse bietet dann, je nach Verwendungszweck, spezifische Zugriffsmethoden, die jeweils bestimmte XMLRPC-Methoden kapselt. Der Nutzer der XML:DB-API bekommt allerdings von den darunterliegenden XMLRPC-Zugriffen nichts mit.
Für den Zugriff auf die Daten der Datenbank gibt es so genannte Ressourcen, welche entweder XML-Daten oder Binärdaten bereit halten. Diese Ressourcen können sowohl auf

dem Client erstellt und dann zum Server hochgeladen, als auch vom Server abgerufen werden. Um auf eine Liste von Ressourcen zuzugreifen, gibt es das RessourceSet, welches über eine Iteratormethode verfügt, die das Durchlaufen der einzelnen Ressourcen ermöglicht.
Implementierungen der XML:DB-API liegen momentan zwar nur für Java vor, lassen sich aber auch in alle anderen objektorientierten Programmiersprachen übertragen.

SOAP
Simple object access protocol (SOAP) verwendet eine standardisierte Beschreibungsdatei. Diese setzt auf die Web Service Description Language (WSDL) auf und beschreibt einzelne Services einer Schnittstelle genauer. Eine solche WSDL-Datei definiert die zu verwendenden Ein- und Ausgabeformate und nutzt XML. Eine SOAP-Schnittstelle kann über eine solche Definitionsdatei beliebig viele Methoden bereitstellen.
SOAP bietet im Gegensatz zu XMLRPC eine echte Datentypisierung. Es erlaubt die Einbindung komplexer XML-Schemadefinitionen, so dass auch stark strukturierte Daten übertragen werden können.
Leider ist SOAP durch diese Fähigkeiten sowohl auf Client als auch auf Server-Seite relativ langsam und benutzt viele Ressourcen. Für kleine, einfache Datenmodelle reicht also XMLRPC für Datenabfragen, für komplexere Daten kann SOAP verwendet werden.

WebDAV
Das Protokoll Web Distributed Authoring and Versioning (WebDAV) bietet durch eine Erweiterung des Protokolls http die Möglichkeit, Dateien zu speichern.
Verschiedene XML-Datenbanksysteme bieten darüber die Möglichkeit, auf einzelne Resourcen direkt zuzugreifen. Neben reinen XML-Resourcen lassen sich dabei auch normale Binärdateien auf Server speichern und wieder abrufen. Clients für WebDAV gibt es für alle gängigen Betriebssysteme

2.2.5 XML- Benchmark

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.
Nebenbei sei erwähnt, dass ein aussagekräftiger Benchmark zur generellen Bestimmung der Leistungsfähigkeit von Computersystemen für alle Anwendungsgebiete nicht möglich ist, da anwendungsbedingt 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. Beispiele von XML-Benchmark sind X-Mach, Xmark, XOO7.
Später, in Kapitel 4, wird zur Bewertung der Leistungsfähigkeit der untersuchten XML-Datenbanksysteme der XML-Benchmark Xmark eingesetzt.

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. 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) widerspiegelt.
Die dem Benchmark zugrunde liegenden Richtlinien werden in [SWKF01b] ausführlich diskutiert.
XMark selbst ist kein Programm [Seem03], sondern eine Spezifikation mit zwanzig in XQuery formulierten Anfragen für ein gültiges XML-Dokument, die möglichst alle Aspekte der Anfrageverarbeitung abdecken sollen [BLL03], das vier Gruppen von Anfragen hinsichtlich der darin verwendeten Anfragefunktionalitäten vorstellt.
Tabelle 2-7 zeigt die vier Gruppen mit den enthaltenen Anfragenummern, der verwendeten Anfragefunktionalität nach [BLL03] sowie den darin verwendeten Konzepten nach [SWK02].

[1] Vergleiche [XMARK]

Gruppe	Anfragenummern	Funktionalität	Verwendete Konzepte
I	1, 5, 14	Einfache Wertevergleiche (verschiedene Typen)	Exakte Übereinstimmung, Datentypkonvertierung, Volltextsuche
II	2, 3, 4, 11, 12, 17	Bewahren der Elementordnung	Sortierungsverhalten, wertbasierter Verbund, optionales Element
III	6, 7, 8, 9, 10, 13, 15, 16	Navigation (Pfadausdrücke)	Reguläre Pfadausdrücke, Referenzierung, komplexe Sequenz, Rekonstruktion von Dokumentteilen, Prüfung auf Pfadexistenz
IV	18, 19, 20	Aggregation und Sortierung	Benutzerdefinierte Funktion, Sortierung, Aggregation

Tabelle 2- 7 Übersicht über die Anfragen in XMark

Domäne. Diesem Benchmark liegt 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. Zudem wird ein Generator für XML-Dokumente namens xmlgen[3] zur Verfügung gestellt, welcher Dokumente in beliebiger Größe erzeugt.
Durch die Aufnahme von Beschreibungstexten werden jedoch auch dokumentorientierte Aspekte eingebracht, d.h. 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 Baumstruktur in der nachfolgenden Abbildung 2 -6 wiedergegeben ist. Es enthält die Auktionsdaten als strukturierte Fakten, welche in den Hauptinhalten (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 ergebenden 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

[3] Siehe hierzu [xmlgen]

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.

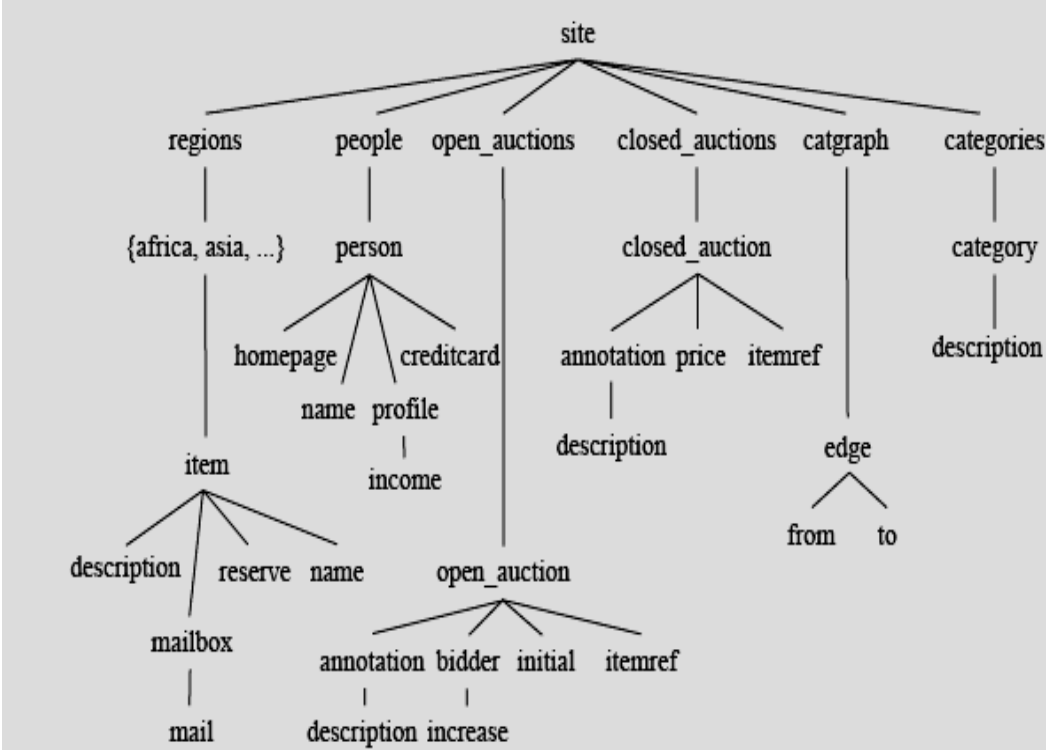


Abbildung 2-6 Baumstruktur des XML-Dokuments aus XMark

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. vier Verbundanfragen, um die Unterstützung bestimmter Optimierungsmöglichkeiten zu bewerten.

KAPITEL 3 ANFORDERUNGSKATALOG

Dieses Kapitel dient dazu, alle Anforderungskriterien vorzustellen, die an ein Produkt XML-Datenbank gestellt werden. Diese Kriterien sind als Muss-Kriterien zu verstehen und sollen zwei Zwecken dienen: In dem großen Marktangebot, die für diese Bachelorarbeit geeigneten XML-Datenbank-Kandidaten auswählen zu können und diese anschließend detailliert zu bewerten.

3.1 VORAUSWAHL

Es ist unmöglich, im Rahmen dieser Bachelorarbeit alle existierenden XML-Datenbanken einer detaillierten Prüfung zu unterziehen. Daher werden nachfolgende
Muss-Anforderungskriterien definiert, um eine engere Auswahl geeigneter Produkte zu identifizieren.

3.1.1 Rahmenbedingungen der Arbeit

Damit die Arbeit mit einer XML-Datenbank möglich ist, müssen einige Rahmenbedingungen erfüllt sein. Dazu gehören eine Dokumentation der wichtigsten Funktionen des Produkts und der Installationsaspekte.
Die offizielle Dokumentation des Herstellers, unabhängige Artikel und Paper über die XML-Datenbanken, sind bedeutsam. Diese ermöglichen es, eine objektive Darstellung der Funktionalität und Qualität einzuholen. Als Quellen werden hierbei CiteSeer[2] und Google Scholar[3] verwendet. In Kapitel 2 wurde deutlich, dass viele XML-Technologien existieren, die für XML-Datenbanken fundamental sind. Diese Technologien sind noch im Aufbau, wie zum Beispiel XQuery, die oft als die wichtigste Anfragesprache für XML-Datenbanken gehandelt wird. Entsprechend bieten viele Produkte im Bereich der XML-Datenbanken eine Spezifikation von XQuery an. Aufgrund dieser ständigen Entwicklungen spielt die Aktualität der Produktversion eine große Rolle. Liegt das Erscheinungsdatum zu weit zurück, dann ist die Unterstützung des Produkts ebenfalls nicht auf der Höhe der Zeit. Die Folge: Die neueste Erweiterung und Verbesserung stehen mit dem Produkt nicht zur Verfügung.
Damit eine Detailbewertung einer XML-Datenbank möglich wird, spielt das Lizenzmodell von Datenbank-Produkten auch eine wichtige Rolle, um den Funktionsumfang des Produkts nicht einzuschränken. Kontakt (E-Mail oder Foren) mit dem Hersteller muss möglich sein. Sprachlich soll englische, deutsche oder französische Kommunikation möglich sein.

[2] http://citeseer.ittc.ku.edu/

[3] http://scholar.google.com

Die Tabelle 3-1 fasst die genannten Muss-Anforderungskriterien im Bereich „Rahmenbedingungen“ zusammen.

Kriterium	Muss Anforderung	Zu erfüllendes Kriterium
K1	Verfügbarkeit	Die aktuelle Version der Datenbank soll verfügbar sein.
K2	Produkt Support	Kontaktmöglichkeit(E-Mail, Forum) mit dem Hersteller soll gegeben sein.
K3	Lizenz	Das Produkt soll unter einer Open Source -Lizenz firmieren.
K4	Aktualität der Produktversion	Aktuellste Version seit Januar 2006
K5	Offizielle Dokumentation	Einrichtung/Installationsfunktionen sollen von Seiten des Herstellers erklärt werden.

Tabelle 3-1 Muss Anforderungen an XML-Datenbanken

3.1.2 Technische Bedingungen

In technischer Hinsicht gibt es Anforderungen, die an jede XML-Datenbank gestellt werden müssen. Die Ursache liegt zum einen im Testsystem, auf welchem die XML-Datenbanken installiert werden sollen und zum anderen auf gewünschte Funktionen, deren Relevanz in Kapitel 2 deutlich geworden ist. Als Betriebssystem steht Windows XP Professional Edition mit Service Pack 2 für jede XML-Datenbank zur Verfügung. Das Produkt soll auch auf Plattformen einsatzfähig sein, die nicht auf Windows basieren. Damit wird die Abhängigkeit von einem Betriebssystemhersteller verhindert und ein späterer Wechsel der Datenbankinstallation auf ein Betriebssystem eines anderen Herstellers ermöglicht.
Um den Performanztest durchführen zu können, ist eine Programmierschnittstelle für die Programmiersprache Java nötig, da alle XML-Datenbanken der engeren Auswahl über das Java Programm „*Easy XML-Database-Benchmark*“ hinsichtlich der Performanz geprüft werden.
Es wird von einer XML-Datenbank erwartet, Funktionen wie beispielsweise Transaktionsverwaltung oder Anfragesprache (2.2.3) zu unterstützen.
In Tabelle 3-2 werden die technischen Anforderungen zusammengefasst.

Kriterium	Muss Anforderung	Zu erfüllendes Kriterium
K1	Anfrage und Manipulation von Daten	Der Queryprozessor des ausgewählten Produkts soll mindestens XQuery/XPath als Anfragesprache verstehen.
K2	Plattform -Unterstützung	Das ausgewählte Produkt soll auf den gängigsten Plattformen betrieben werden können. Da die Performanzmessung im Rahmen dieser Arbeit in einer Windows-Umgebung stattfindet, sollen mindestens Microsoft Windows Plattformen unterstützt werden.
K3	Programmier-Schnittstelle-APIs	Da die Performanzmessung des untersuchten Produkts durch ein eigenes Java basiertes Programm stattfinden soll, ist es unabdingbar, dass das Produkt eine Java Programmierschnittstelle bietet.
K4	Administration	Allgemeine Administration (Anlegen, Ändern, Löschen, Starten, Stoppen). Vergabe von Lese-/Schreibrecht.
K5	Transaktionen	Unterstützung von Transaktionen (ACID)
K6	Authentisierung	Benutzer können authentisiert werden.
K7	Datenbankart	Das Produkt soll entweder XML -native oder XML -enabled sein.

Tabelle 3- 2 Technische Muss-Anforderungen an XML-Datenbanken

Die Vorauswahl an XML-Datenbanken dient als Filterkriterium.

3.2 DETAILBETRACHTUNG

Folgende Anforderungen bilden die Grundlage, um die beste XML-Datenbank in der Vorauswahl und damit auf dem Gesamtmarkt identifizieren zu können. Grundsätzlich werden drei Kriterien-Gruppen unterschieden: Funktionalität, Plattformunterstützung und Performanz.

Anforderung	Beschreibung	Referenz
Zugriffsvariante:		
API	Es gibt auch verschiedene Programmierschnittstellen für XML-Datenbanken, die auch auf XML-Parsern basieren können. Eine breite Unterstützung ist wünschenswert.	Kap. 2.1.6 und 2.2.5
Programmiersprache	Der Zugriff über eigene Applikationen, geschrieben mit bestimmten Programmiersprachen, hängt unmittelbar mit der API zusammen.	
Anfrage und Datenmanipulationssprachen:		
Diverse Sprachen	Eine XML-Datenbank soll möglichst viele Sprachen für das Stellen von Anfragen oder für das Datenmanipulieren bieten. Darunter sind XML-Technologien und relationale Sprache, sowie die Mixtur aus beidem	Kap.2.2.3
Schemaunterstützung :		
Schemasprache	Ein XML -Datenbank soll möglichst viele Schemasprachen unterstützen.	Kap.2.1.6
Sicherheitsdienste:		
Diverse Funktionen	Eine XML -Datenbank sollte auch Funktionen im Bereich der Sicherheit bieten. Benutzer sollten mit ihren Zugangsdaten authentisiert und autorisiert werden können.	
Transaktionsverwaltung:		
Eigenschaft	Von einer Transaktion wird erwünscht, die ACID-Eigenschaften zu erfüllen.	

Tabelle 3- 3 Betrachtungsaspekte hinsichtlich der Funktionalität

Anforderung	Beschreibung	Referenz
Plattformunterstützung		
Unterstützung diverser Betriebssysteme	Die optimale Situation ist eine XML-Datenbank, die sowohl Windows-, Linux- und Unix-Plattformen unterstützt. Das Produkt muss unter dem Betriebssystem installierbar und ohne Funktionseinschränkung einsetzbar sein.	
Verarbeitungsdauer von Anfragen aus XMark		
Durchführen der zwanzig Anfragen aus XMark mit drei XML-Dokumenten	Alle Anfragen aus XMark werden auf jeweils ein XML -Dokument angewendet und die Verarbeitungsdauer gemessen.	Kap.4.1

Tabelle 3- 4 Betrachtungsaspekte hinsichtlich Plattform und Performanz

KAPITEL 4 MARKTANALYSE VON XML OPENSOURCE DATENBANKEN

In diesem Kapitel wird die Beschaffung der Marktinformationen von XML-Datenbanken dargestellt. Das Ziel ist, anschließend zu definieren, welche XML-OpenSource Datenbanken im Rahmen dieser Arbeit berücksichtigt werden.
Diese Marktanalyse beruht auf einer aktuellen Darstellung der Marktsituation. Hierbei werden die Daten erhoben, die für Entscheidungen herangezogen werden können. In Abschnitt 4.1 wird eine Übersicht auf den gesamten Markt gegeben.
Nach Anwendung der Anforderungskriterien aus Kapitel 3 auf diese Übersicht werden XML-Datenbanken ermittelt, die in meiner Bachlorarbeit verglichen werden.

4.1 ÜBERBLICK ÜBER DEN GESAMTMARKT

Zur Marktanalyse werden in dieser Arbeit lediglich externe Daten d.h. Internet[4] und Bücher[5] als Quellen sowie Dokumentationen über Produktanbieter verwendet. Es wird wie in Abschnitt 2.2.2 zwischen XML-fähigen Datenbanken und nativen XML-Datenbanken unterschieden. Diese Kategorisierung soll es in Verbindung mit Kapitel 2 erlauben, auch die Arbeitweise von XML-Datenbankprodukten zu kennen.
Das Ergebnis der Marktuntersuchung wird in den nachfolgenden Tabellen dargestellt.

Lfd.Nr	Name des XML DBMS	Hersteller Produktanbieter	Produktversion	Betriebssystem	Open Source ?
1	DB2	IBM	2.8	Linux, Windows	Nein
2	SQL-Server 2005	Microsoft	9.1.2047	Windows	Nein
3	Oracle DB	Oracle	10.2.0	Linux, Windows	Nein
4	PostgreSQL	PostgreSQL Global-	8.3-beta 3	Linux, Windows	Ja

Tabelle 4- 1 Markübersicht über XML-fähige Datenbanken

[4] Hauptquelle ist http://sourceforge.net; http://www.rpbourret.com/xml/XMLDatabaseProds.htm

[5] [KlMe03], [ACAB]

Tabelle 4-2 nennt die nativen XML-Datenbanken

Lfd.Nr	Name des XML DBMS	Hersteller Produktanbieter	Produktversion	Betriebssystem	Open Source ?
1	TIMBER	University of Michigan	1.0	Windows	Ja
2	eXist	Software AG	1.2.0-rev7233	OS independent	Ja
3	Sedna XML DBMS	ISP RAS MODIS	1.0	Linux, Windows	Nein
4	Berkeley DB XML	Sleepycat Software (Oracle)	2.3.10	Linux, Windows	Ja
5	4Suite, 4Suite	FourThought	1.0.2	Linux, windows	Ja
6	DB XML	DBXML Group	2.0	Linux, Windows	Ja
7	Infonyte DB	Infonyte GmbH	3.5	Windows	Nein
8	Tamino on Rails	Software AG	1.0.b beta	OS Independent	Nein
9	Ozone	Ozone-db.org	1.2.x	OS independent	Ja
10	Natix	Data ex machina	2.1.1	Unix	Nein
11	Xyleme	Xyleme SA	3.4	Linux	Nein
12	Xindice	Xindice Apache Community	1.1b4 beta	Linux Windows	Ja

Tabelle 4- 2 Markübersicht über nativen XML-Datenbanken

Wie aus der Marktanalyse hervorgeht, sind neben vielen nativen XML- Datenbanken nur wenige XML-fähige Datenbanken verfügbar. Die Auflistung ist nicht abschließend. Zur Prüfung werden die Bedingungen aus Abschnitt 2.2.2.1 verwendet.

4.2 PRODUKTE FÜR VORAUSWAHL

Es wurde im Kapitel 3 eine Reihe von Kriterien genannt, welche eine XML-Datenbank erfüllen muss, um in die engere Wahl aufgenommen zu werden. Bei der Verwendung des Anforderungskataloges scheiden als Konsequenz einige XML-Datenbanken aus, die in der folgenden Tabelle präzis dargestellt werden.

Anwendung der Rahmen- und technischen Bedingungen

Durch die Anwendung der Kriterien aus Kapitel 3 sind in der folgenden Tabelle die XML-fähigen Datenbanken, die begründet abgewiesen wurden, zusammengefasst.

Produkt	Ausscheidungsgrund
DB2	Kommerziell
Oracle DB	Kommerziell
SQL server 2005	Unterstützt lediglich Betriebssysteme von Microsoft

Tabelle 4- 3 Ausgeschiedene XML-fähige Datenbanken

Bei den nativen Datenbanken scheiden folgende Produkte aus:

Produkt	Ausscheidungsgrund
Natix	Ungenügende Dokumentation, kein Support nur Linux
Xyleme Server	Dokumentation und Produkt nur kommerziell verfügbar
4Suite 4Suite	Keine Java API, fehlende Unterstützung von XQuery
DbXML	Fehlende Unterstützung von XQuery, Datenmanipulation
Infonyte	Fehlende Unterstützung von XQuery, Datenmanipulation
Ozone	Fehlende Unterstützung von XQuery, Datenmanipulation
Sedna	Keine Open Source
Tamino	Kommerziell, keine Open Source
Timber	Fehlende Transaktionsverwaltung, Authentisierung/Autorisierung
Xindice	Fehlende Transaktionsverwaltung, Unterstützung von XQuery

Tabelle 4- 4 Ausgeschiedene native XML-Datenbanken

Damit sind folgende XML-Datendanken in der engeren Auswahl:
XML-fähige Datenbank
- PostgreSQL

Native XML-Datenbanken
- Berkeley DB XML.
- Exist

Damit ergeben sich drei XML-Datenbanken, die Kandidaten für den Vergleich sind. Die anderen XML-Datenbanken werden nicht berücksichtigt, weil sie die in Abschnitt 3.1.1, 3.1.2 genannten Anforderungskriterien nicht erfüllen.

4.3 PORTRÄTIERUNG DER AUSGEWÄHLTEN XML- DATENBANKEN

Die ausgewählten XML- Datenbanken werden in diesem Abschnitt betrachtet. Hier wird die Entstehungsgeschichte vorgestellt.

4.3.1 *Datenbank Berkeley DB XML*

Die Arbeit über Berkeley DB XML Projekt begann in 2001. Die ersten Freigaben des Produktes 1.0 waren Ende 2002/ Anfang 2003. Die erste der gegenwärtigen Reihe 2.x der Freigaben kam Ende 2004 heraus. Ab der Version 2.x bietet Berkeley DD XML die Möglichkeit der Knoten-Speicherung (node Storage) und XQuery. Seit Februar 2006 gehört Sleepycat Software zur Oracle Corporation. Aus diesem Grund werden die Produkte Berkeley DB, Berkeley DB XML (BDBXML) über Oracle vertrieben. Die Kontaktaufnahme zu den Entwicklern wird über ein eigens dafür moderiertes Forum im Internet zur Verfügung gestellt, anstatt über die Mailinglisten von Sleepycat Software. Der Support der ganzen Produktlinie von BDBXML ist in dem Oracle Support integriert.

4.3.2 *Datenbank PostgreSQL*

PostgreSQL ist ein Objektrelationales Datenbank-Management-System, das auf dem POSTGRE 4.2 System basiert, welches an der Universität von Kalifornien in Berkeley entwickelt wurde. Die Implementierung begann im Jahr 1986 und wurde von Prof. Stonebraker angeleitet
Die erste Demoware war 1987 fertig. Die erste Version wurde 1989 fertiggestellt.
Im Jahre 1994 bauten Andrew Yu und Jolly Chen einen SQL-Interpreter in POSTGRES ein, weil das System bis dahin kein SQL kannte. Der Postgres95-Code war komplett in ANSI C geschrieben (was die Portabilität erhöht).
Ein neues Anfrageprogramm zur direkten Interaktion mit Datenbankservern Namens „psql“ wurde entwickelt. Im Jahr 1996 wurde das System zu PostgreSQL unbenannt.
Seit 2001 stellt PostgreSQL eine Reihe von grundlegenden XPath- und XSLT-Funktionen über das Contrib Modul „XML2“ zu Verfügung, die von John Gray entwickelt wurde. 2006 begann die Implementierung des nativen Typs XML. Eine der Hauptvoraussetzungen für den neuen Datentyp war die Übereinstimmung zum SQL/XML-Standard.

4.3.3 *Datenbank eXist*

EXist ist ein natives XML-Opensouce Datenbankprojekt von Wolfgang Meier. Die Arbeit an eXist begann im Januar 2001. Seither wurde weiterentwickelt. Beispielsweise war für lange Zeit die einzige unterstützte Anfragesprache XPath, mittlerweile wird aber auch XQuery und XUpdate unterstützt.

KAPITEL 5 VERGLEICHSDURCHFÜHRUNG

Das vorherige Kapitel 4 diente der Herleitung der Vorauswahl ausgehend vom Gesamtmarkt. Nachdem die ausgewählten Produkte bekannt sind, hat dieses Kapitel das Ziel, die wichtigsten Aspekte von ausgewählten XML-Datenbanken zu vergleichen und zu bewerten. Dazu werden von jeder XML-Datenbank die Aspekte wie Architektur, Plattformen, Speicherbarkeit, Datensicherheit, Anfragemöglichkeit, Transaktion und Lizenzierung erläutert.

5.1 ARCHITEKTUR

In diesem Abschnitt wird die Architektur der drei Datenbank-Systeme untersucht und verglichen. Konkrete Betrachtungen der Systeme werden erläutert.

5.1.1 Architektur von Berkeley DB XML

Berkeley DB XML (im Folgenden als BDBXML abgekürzt) ist eine eingebettete, native XML-Datenbank und besteht aus mehreren Programmbibliotheken, deren Programmmodule in C und C++ geschrieben sind. Sie beinhaltet APIs beispielsweise für Java. BDBXML setzt auf Berkeley DB und ergänzt dieses Fundament, welches selbst eine Programmbibliothek ist, um drei Teile: Einen XML-Parser, eine XML- Indexverwaltung und eine Anfrageverwaltung für XQuery, welche in Abbildung 5-1 zu sehen sind. BDB XML nutzt für klassische Datenbankfunktionen wie Replikation, Transaktionen, und Wiederherstellung, die bereits bestehende Möglichkeit der als Unterbau verwendeten Berkeley DB. Ein Kommandozeilenprogramm (BDBXML Shell, dbxml) ist als vorgefertigte Benutzerschnittstelle mitgeliefert, um direkt mit dem Datenbanksystem interagieren zu können.

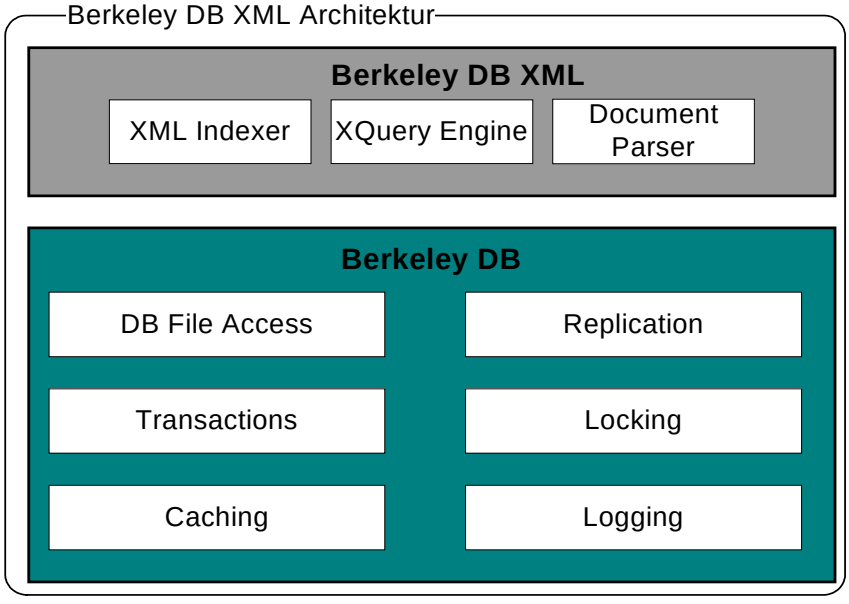


Abbildung 5- 1 Architektur von BDB XML und dessen Fundament Berkeley DB angelehnt an [Dan06]

5.1.2 Architektur von PostgreSQL

Das System arbeitet unter einer Client-Server-Architektur, wobei die Serverseite durch das Datenbankmanagement als Postmaster Prozess realisiert wird. PostgreSQL benutzt ein Prozess-pro-Verbindung-Modell. Wird eine Anfrage vom Client gestellt, wird vom Postmaster ein Server-Prozess gestartet.
Der Server hat eine Anfragebearbeitungsschicht (die innerhalb des großen grauen Blocks abgebildet ist), die im Abschnitt 5.4.2 ausführlich dargestellt wird. Für die Kommunikation wird der JDBC-Treiber eingesetzt. PostgreSQL-Server-Komponenten benutzen wiederum die Routine der Bibliothekenkomponente. Die Abbildung 5 –2 verdeutlicht die Architektur von PostgreSQL.
PostgreSQL bietet Volltextsuche im Server an, der Datentyp XML verarbeitet wohlgeformte XML-Dokumente und bietet SQL/XML-Funktionen nach dem SQL 2003-Standard. Die Systemteile, die zur Unterstützung von XML-Type und SQL/XML geändert wurden, sind mit dem Sternchen-Zeichen und Farbe Bordeaux gekennzeichnet.

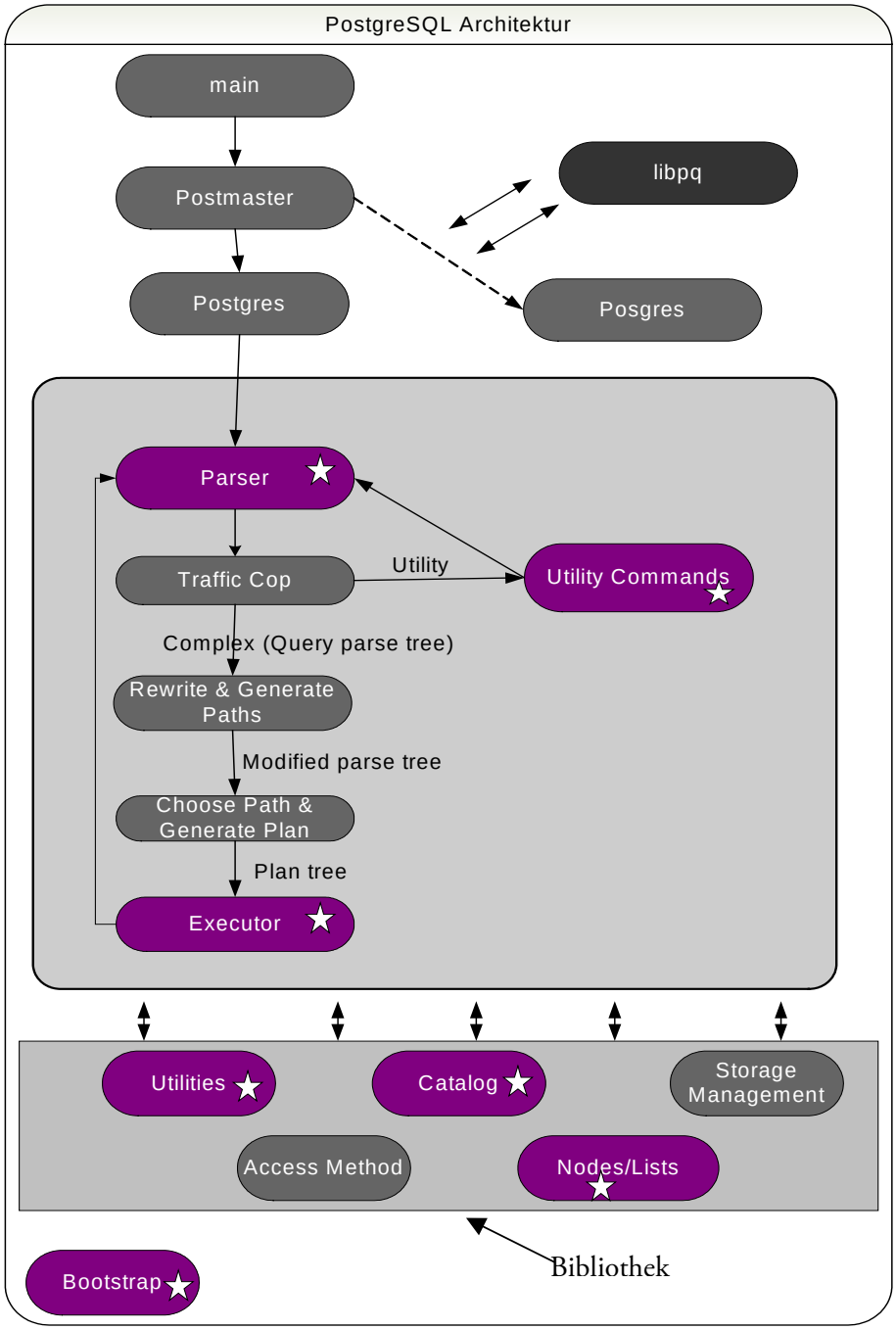


Abbildung 5- 2 Architektur von PostgreSQL

5.1.3 Architektur von eXist

Die Architektur von eXist basiert auf der Client/Server-Architektur. Der Server-Prozess ist JETTY. Es ist möglich, andere Webserver zu installieren und eXist darüber zu nutzen. Alternativ kann man auf einen Server-Prozess verzichten. Dann läuft die XML-Datenbank nur über einen Client-Prozess. In diesem Fall ist eXist vergleichbar mit einer eingebetteten XML-Datenbank und erlaubt über die XML:DB API einen direkten Zugriff auf den DOM-Baum. Im Normalfall ist der Serverprozess JETTY für den Zugriff auf die Datenbank verantwortlich. HTTP und SOAP im Zusammenhang mit Webservices können für die Client/Server-Kommunikation verwendet werden. Die XML:DB API unterstützt weiter XML-RPC (XML-Remote-Procedure-Call). EXist bietet auch Zugriffsmöglichkeit über WebDAV an.
Neben dem nativen XML-Datenspeicher, der in Java implementiert ist, kann auch ein relationaler Datenspeicher als Backend verwendet werden [Meie03].
Der Datenspeicher für das Datenbankverwaltungssystem von eXist ist transparent, das heißt, er passt seine Funktionsweise nicht an das vorliegende Backend an. Dies wird ermöglicht durch die Java-Klasse DBBroker, welche als Schnittstelle zwischen dem Backend und dem Datenbankverwaltungssystem fungiert. Für die Kommunikation mit einem relationalen Datenbanksystem wird ein JDBC-Treiber eingesetzt, der in die conf.xml-Datei eingebunden werden muss. Im Gegensatz zu einem in eXist fest integrierten, nativen XML-Datenspeicher stammt der relationale Datenspeicher grundsätzlich von einem frei wählbaren Fremdanbieter. Die Wahl des Datenspeichers hängt laut [Meie03] von den Anforderungen ab: Soll der Datenspeicher möglichst zuverlässig sein, ist der relationale Datenspeicher vorzuziehen, während der native XML-Datenspeicher bessere Performanz bringen soll.
Die folgende Abbildung 5-3 verdeutlicht die Architektur in eXist angelehnt an [Meie03].

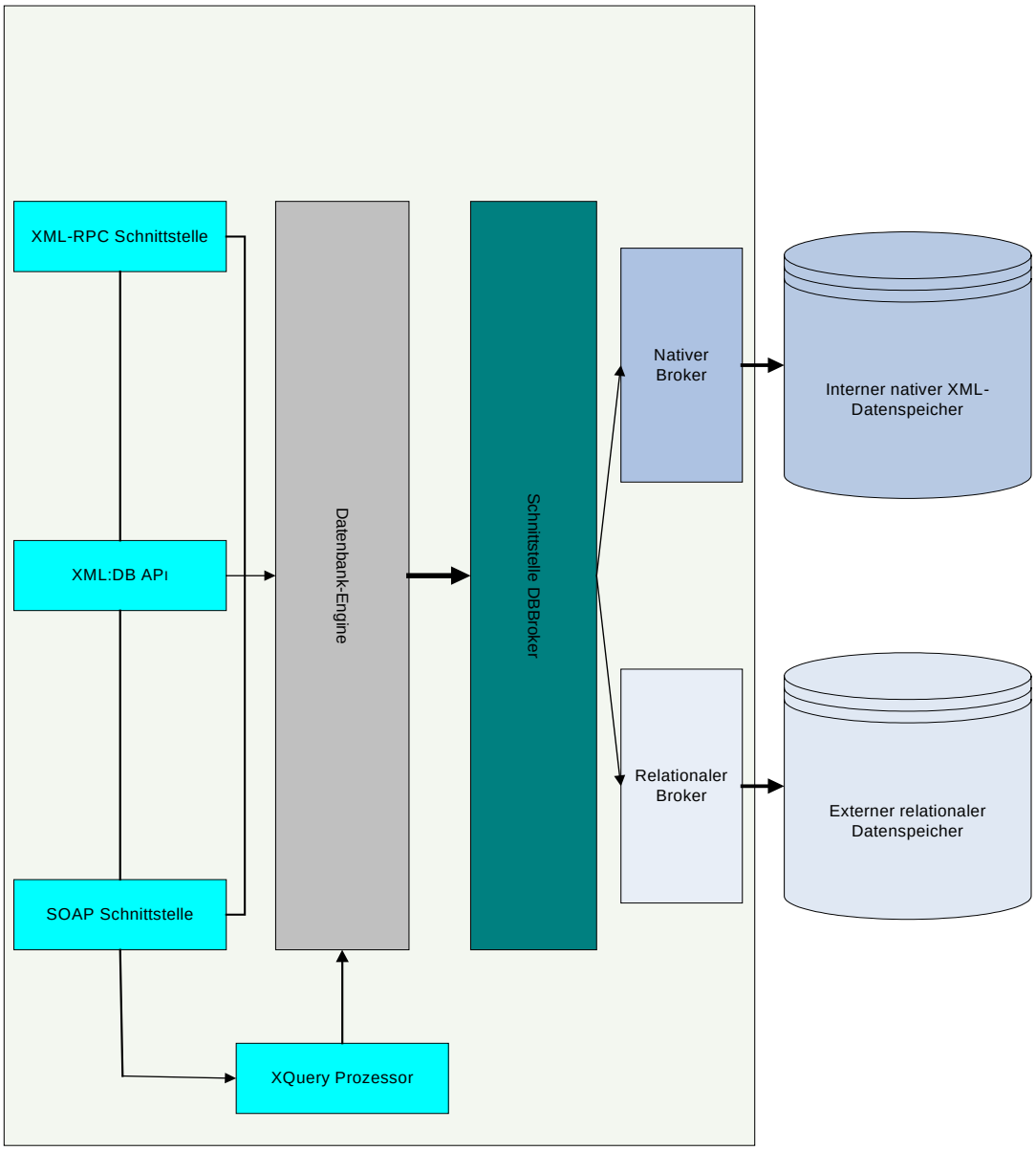


Abbildung 5- 3 Architektur von eXist

5.1.4 Bewertung der Architektur

Kriterien	BDBXML	eXist	PostgreSQL
Client/Server	–	√	√
Transaktion	√	√/–	√
Eingebettet System	√	–/√	–

Tabelle 5- 1 Bewertung der Architektur

5.2 PLATTFORMUNTERSTÜTZUNG

Dieser Abschnitt wird kurz darstellen, auf welchen Plattformen sich die ausgewählten XML-Datenbankprodukte betreiben lassen.

5.2.1 Plattformunterstützung von Berkeley DB XML

Berkeley DBXML lässt sich auf Plattformen betreiben, auf denen sich Berkeley DB betreiben lässt, da BDBXML auf Berkeley DB aufbaut. BDBXML ist für Windows und Linux vom ersten Tag entwickelt worden. Es ist möglich BDBXML unter Windows in jeder veröffentlichten Version zu bauen und zu installieren.
Berkeley DBXML ist nicht auf Server-Betriebssystemen wie Microsoft Windows Server 2003 nutzbar.

5.2.2 Plattformunterstützung von PostgreSQL

Ursprünglich war Unix die Plattform, auf der PostgreSQl betrieben wurde. Ab der Produkt-Version 8.0.0 gibt es den nativen Windows-Installer für PostgreSQL. Das Produkt lässt sich auf den gängigen Plattformen betreiben, jedoch nicht auf Windows Vista.

5.2.3 Plattformunterstützung von eXist

Auf jedem Betriebssystem, im welchem sich eine Java Umgebung einrichten lässt, kann auch eXist betrieben werden. Das bedeutet, dass eXist unter allen genannten Plattformen betrieben werden kann.

5.2.4 Bewertung

Diese Tabelle stellt die Bewertung der Plattformunterstützung für die ausgewählten Datenbanken dar.

Datenbank / **Plattformen**	Berkeley DB XML	eXist	PostgreSQL
Windows XP Home/Pro	✓	✓	✓
Windows Server 2003	×	✓	✓
Windows Vista	✓	✓	×
Linux enterprise edition	✓	✓	✓
Freie Linux distribution	✓	✓	✓
Unix	✓	✓	✓
Mac OS X	✓	✓	✓

Tabelle 5- 2 Bewertung der Plattformen

5.3 SPEICHERBARKEIT

In diesem Abschnitt wird der Speicheransatz jedes ausgewählten Produktes erläutert.

5.3.1 Speicherbarkeit bei Berkeley DB XML

Berkeley DB ist keine relationale oder objektorientierte Datenbank. Überhaupt lässt sich kein bekanntes Datenmodell angeben [BERDEF]. Vielmehr speichert Berkeley DB ein Paar (das sog. *Record*) aus einem Schlüsselwert (der sog. *Key*) und Datenwert (ein sog. *Value*), wobei der Datenwert keinen Datentyp auferlegt bekommt und auch eine variable Länge einnehmen kann. Diese Records werden in Tabellen (engl. tables) gespeichert, was konzeptionell an die Tupel in Relationen erinnert.
Das Speicherkonzept bei Berkeley DB XML (im folgenden als BDBXML) bietet dem Benutzer die Wahl, wie das XML-Dokument gespeichert werden soll. Im Allgemeinen werden XML-Dokumente in so genannten Containern gespeichert. Ein Container entspricht einer Kollektion von XML-Dokumenten. Physisch ist der Container immer eine einzige Datei, unabhängig von der Anzahl der darin enthaltenen XML-Dokumente oder Zugriffspfade.

DBXML verwendet zwei Typen von Containern:

- Ein Typ speichert XML-Dokumente als Ganzes (textbasiert). Das entspricht wie in Abbildung 2-3 dem Dateisystem-Speicheransatz. Es wird für XML-Dokumente empfohlen, die kleiner als 1 Megabyte sind.
- Der andere Typ speichert die Knoten des XML-Baums, indem eine Dekomposition des XML-Dokuments in den einzelnen Blattknoten stattfindet (modellbasiert). Die Baumstruktur wird separat gehalten. Diese Art wird eingesetzt, um XML-Dokumente zu speichern, die größer sind.

Die flexible Weise der Speicherung der XML-Dokumente in BDBXML ist Grund für die Einordnung in die XML-Datenbanken mit hybridem Speicheransatz. Die Wahl des Speicheransatzes wird und kann nur bei Erstellung des Containers erfolgen. Ein Container enthält auch Metadaten, wie beispielsweise das letzte Modifikationsdatum und Indizes der XML-Dokumente. Das im Container gespeicherte XML-Dokument enthält eine Identifikationsnummer, die eine natürliche Zahl ist. Je nach eingesetzter Art des Containers können XML-Dokumente strukturerhaltend gespeichert werden.
Die textbasierte Speicherweise speichert eine exakte Kopie des XML-Dokuments. Bei der modellbasierten Speicherung wird das XML-Dokument in UTF-8 umkodiert (falls UTF-8 nicht bereits vorliegt).
Der Containertyp *Wholedoc* wird eingesetzt beim textbasierten Speicheransatz; hier gibt es kaum Details. Da das Dokument jedoch eine exakte Speicherung einer Kopie des XML-Dokuments verspricht, nehme ich an, dass je XML-Dokument ein einzelner Record geführt wird und dessen Datenwert alle Zeichen des XML-Dokuments aufnimmt.
Der Containertyp namens *Node* Container wird beim modellbasierten Speicheransatz eingesetzt. Dieser Typ speichert jeden Blattknoten des XML-Baums von einem XML-

Dokument (aus der Kollektion) in einem einzelnen Record. Im Datenwert werden auch die Attribute mit ihren Werten und alle Textdaten gespeichert.

5.3.2 Speicherbarkeit bei PostgreSQL

Das Speicherkonzept bei PostgreSQl basiert auf der nativen Speicherung von XML-Dokumenten. Sollen XML-Daten in eine relationale Tabelle eingefügt werden, dann wird das zugehörige XML-Dokument als Datentyp XML deklariert. Der Datentyp XML kann auf einzelne Spalten der relationalen Tabelle angewendet werden. Beim Einfügen (*INSERT*) wird das XML-Dokument mit dem XML-Parser für die Wohlgeformtheit überprüft.
Bei der Speicherung des XML-Dokuments, unterscheidet PostgreSQL zwei Konzepte:

- Unstrukturierte Speicherung

  Bei dieser Methode wird das XML-Dokument als CLOB[4] gespeichert. Dies entspricht dem textbasierten, nativen Speicheransatz, wie in Abbildung 2.2.2 erläutern wurde.
- Strukturierte Speicherung

  Dieses Konzept entspricht keinem nativen Speicheransatz, sondern das XML-Datenmodell wird auf dem Relationenmodell abgebildet. Jedes Element im XML-Dokument wird zu einem Attribut im Relationenschema. Wie in Abbildung 2-4 erläutert, wird dieser Mechanismus auch „Schreddern" genannt.

5.3.3 Speicherbarkeit bei eXist

Das Speicherkonzept von eXist basiert auf zwei Ebenen: Dem XML-Baum und den XML-Dokumenten von einem XML-Dokument.
XML-Dokumente werden in einer Kollektion gespeichert. Eine Kollektion kann wiederum eine andere Kollektion enthalten. Die Kollektionen werden wie bei einem Dateisystem organisiert. Der XML-Baum eines XML-Dokuments wird in einem DOM gespeichert. Die verwendete Datenstruktur ist ein B^+-Baum und ist physisch in einem *Paged File* gespeichert. Der B^+-Baum sorgt dafür, dass die Nummer eines Knotens mit seiner Speicheradresse assoziiert werden kann. Innerhalb einer Speicherseite werden die Knoten in der Reihenfolge gespeichert, mit welcher sie im Dokument auftreten.
In eXist werden häufig verwendete Achsen in Anfragen automatisch mit einem Strukturindex, basierend auf ein Nummerierungsschema, indiziert. Es könnten auch Volltextindizes auf Dokumentteile definiert werden.
Die Abbildung 5-4 zeigt den Speicheransatz des DOM-Baums in eXist, angelehnt an [Meie03]

[4] CLOB , engl. Character Large Objekt

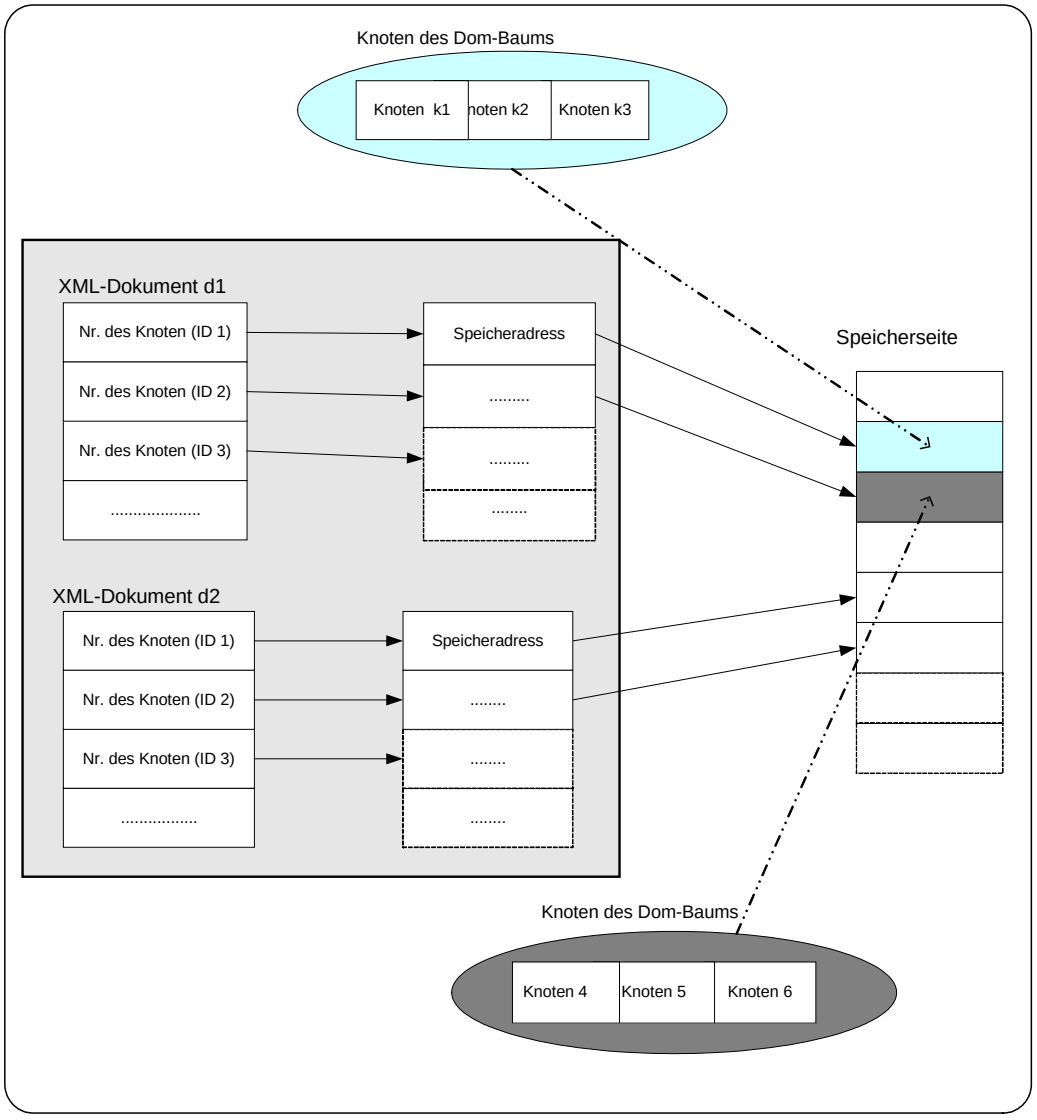


Abbildung 5- 4 Speicherung des DOM-Baums in eXist

5.3.4 Bewertung der Speicherbarkeit

Bei der Bewertung der Speicherbarkeit, wird zusammengefasst, ob die ausgewählten XML-Datenbanken die Kriterien erfüllt haben oder nicht.

Systemname / Kriterien	Berkeley DB XML	eXist	PostgreSQL
Verwaltung von Containern	✓	✓	×
Verwaltung von Tabellen	×	×	✓
Wohlgeformtes XML-Dokument	✓	✓	✓
Gesamtes XML-Dokument	✓	✓	✓

Tabelle 5- 3 Bewertung der Speicherbarkeit

5.4 DATENSICHERHEITSDIENSTE

Datenbanken-Daten können verloren gehen oder beschädigt werden. Dafür gibt es mehrere Gründe. Einer davon ist der Fremdzugriff auf das System. Um sich vor solcher Bedrohung zu schützen, hat jedes Datenbankprodukt ein Sicherheitssystem entwickelt. Die Sicherheit der von mir ausgewählten Datenbanken wird kurz dargestellt.

5.4.1 Sicherheit von Berkeley DB XML

Bei BDBXML ist ein Container physisch eine Datei zumeist mit der Endung „*.dbxml*“ im Dateisystem. Dabei lässt sich diese Datei speziell schützen, sodass beispielsweise nur ihr Eigentümer sie über BDBXML lesen kann. Ein Fremdzugriff würde durch das Betriebssystem vereitelt. Alternativ lässt sich die Datenbankumgebung als Verzeichnis schützen. BDBXML unterstützt die Verschlüsselung der gespeicherten XML-Dokumente über den Algorithmus namens Advanced Encryption System (AES), der ausführlich in [Oppl05] vorgestellt wird. Verschlüsselt werden kann der Container oder die ganze Datenbankumgebung, das heißt alle Datenbanken und auch anfällige Log-Dateien. Die Details zur Datenverschlüsselung sind in [BDBXMLSEN] erwähnt.
Als eingebettete Datenbank hat BDXML „Zero Administration“. BDXML delegiert die Authentisierung und Autorisierung an das Betriebssystem.

5.4.2 Sicherheit von PostgreSQL

Physisch bietet PostgreSQL ein Programm zum Erstellen von Datensicherungen von ganzen Datenbankclustern. Dies wird mit dem „*pg-dumpall*“ realisiert. Die Sicherung auf Dateisystemebene funktioniert wie folgt: Alle Dateien müssen im Datenbankcluster mit Betriebssystemmitteln gesichert werden.
Bei PostgreSQL muss ein Benutzer sich mit dem Benutzernamen und Passwort authentisieren. Wird ein Passwort eingegeben, so wird dieses mit MD5 verschlüsselt.

5.4.3 Sicherheit von eXist

Die XML:DB API wurden vom eXist-Entwickler um einen Zusatz erweitert, um Benutzer und deren Recht über eigene Applikation verwalten zu können. Hier wird der Dienst namens *UserManagementService* einer Kollektion aufgerufen. Die Schnittstelle bietet mehrere Methoden beispielsweise Berechtigungen zu ändern und neue Besitzer anzulegen. Bei eXist gibt es zwei Kontexte: Datei (XML-Dokument) und Kollektion. Um eine Datei bzw. eine Kollektion zu lesen, zu modifizieren oder darin zu schreiben, wird Leserecht, Modifizierungsrecht bzw. Schreiben benötig.
Im Rahmen der Authentisierung gibt der Benutzer sein Passwort ein, das mit MD5 verschlüsselt wird. MD5 ist eine kryptongraphische Haschfunktion.

5.4.4 Bewertung

Systemname / Kriterien	Berkeley DB XML	eXist	PostgreSQL
Authentisierung der Benutzer	Durch das OS[5] oder eigenes Programm	erfüllt	erfüllt
Zugriffrechtverwaltung	Durch das OS oder eigenes Programm	erfüllt	erfüllt
Weitere Dienste	Nicht bekannt	MD5 für PW[6]	MD5 für PW

Tabelle 5- 4 Bewertung der Sicherheit

5.5 ANFRAGEMÖGLICHKEITEN

Der Zugriff auf die Datenbasis ist durch geeignete Anfragesprache möglich. In diesem Abschnitt wird der Anfragemöglichkeit jeder XML-Datenbank nachgegangen.

5.5.1 Anfragemöglichkeit von Berkeley DB XML

BDBXML verwendet das offizielle XQuery des W3C, genauer mit der Spezifikation, welche am April 2005 verabschiedet wurde. Der Pathan-Prozessor wird eingesetzt, um Anfragen zu verarbeiten. Für das Einlesen der XML-Daten wird Xerces verwendet; er ist gleichzeitig ein SAX- und DOM-Parser. Variablenbindungen für XQuery-Ausdrücke können, wie üblich, über den Anfragenteil *let* über BDBXML erfolgen. Berkeley DB XML unterstützt streng genommen zurzeit keine Datenmanipulationssprache, da auf die

[5] Operating System

[6] Passwort

Nutzung von XUpdate, Entwicklungsstufen von Erweiterungen in XQuery oder auf Eigenentwicklungen verzichtet worden ist. Dieser Umstand ändert sich erst mit dem Erscheinen der fertig gestellten XQuery Update Facility, deren Implementierung in BDBXML bereits vorgesehen ist.
Manipulationsmethoden wie *append* können über die Shell von BDBXML verwendet werden. Zuerst wird allerdings über die Shell-Methode *query doc* oder *query collection* der Kontext festgelegt. Dabei kann auf Knotenebene gearbeitet werden, d.h. es ist möglich, Elemente, Attribute, Kommentare, Verarbeitungsanweisungen und Zeichendaten zu manipulieren. Dazu gibt es neben *append* auch Methoden wie *insertAfter* oder *removeNodes*.

5.5.2 Anfragemöglichkeit von PostgreSQL

Anfragen werden in PostgreSQL über SQL oder Xpath gestellt. Die Xpath-Anfrage lässt sich in eine SQL-Anfrage einbinden. Die contrib "XML2"-Komponente bietet die Xpath Funktion.
PostgreSQL unterstützt SQL/XML, die Teil der SQL-Spezifikation ist. SQL/XML behandelt die Abbildung zwischen relationalen Tabellen und XML-Dokumenten. Die Anfrage XQuery wird nicht unterstützt.
Abbildung 5-5 stellt die Anfragenverarbeitungsstruktur von PostgreSQL dar.

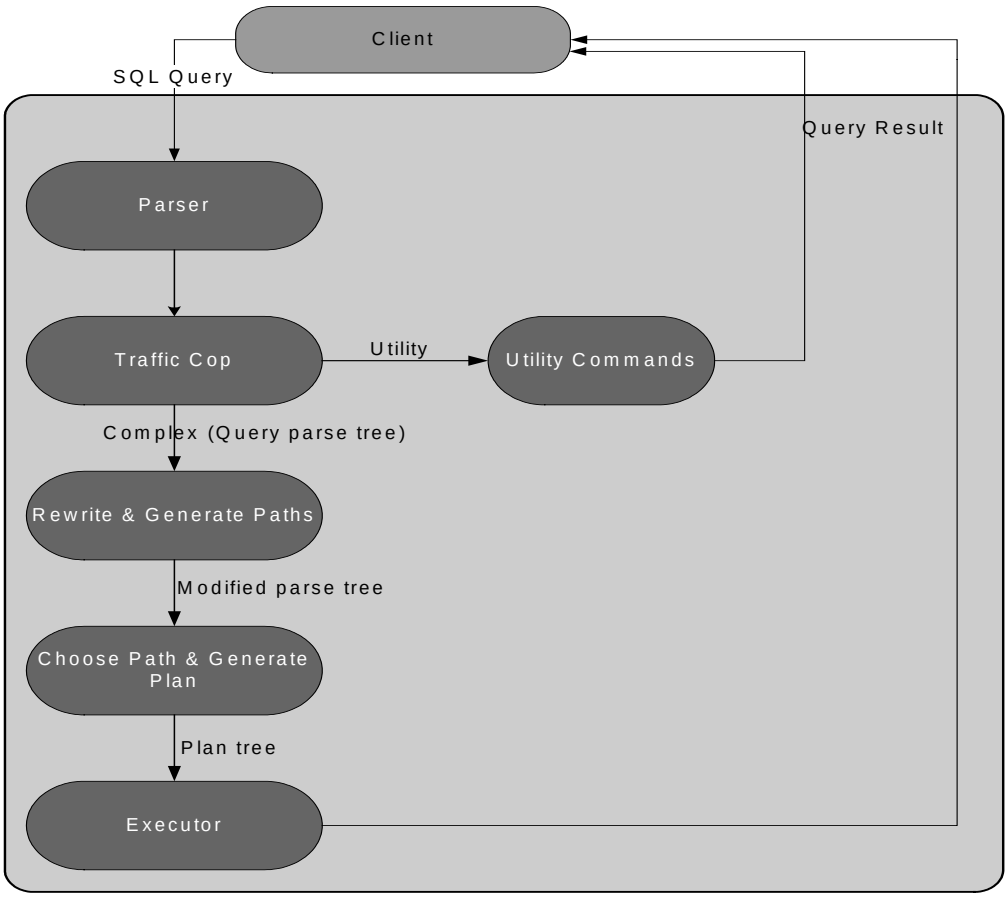


Abbildung 5- 5 PostgreSQL - Struktur für die Anfrageverarbeitung

Diese Abbildung gibt eine Übersicht darüber, welche Schritte eine Anfrage durchlaufen muss, um ein Ergebnis zu erzielen. Hier werden nicht alle Komponenten detalliert erklärt, sondern die Kernkomponenten zur Anfragevearbeitung. Dazu gehören der Parser, Rewriter, Traffic Cop, Excecutor. Ein Client sendet eine SQL-Anfrage an den Parser.

- **Parser:**

Die SQL-Anfragen werden in so genannte Token zerlegt. Dabei wird die Syntax der Anfrage kontrolliert und bei korrekter Syntax eine ParseTree-Struktur erstellt. Dieser ParseTree wird an den Traffic Cop weiter gegeben.

- **Die Traffic Cop-Komponente :**

Sie ruft für jeden Teil des ParseTrees einen eigenen Parser auf und fasst die Ergebnisse in einem QueryTree zusammen. Gleichzeitig wird durch den Traffic Cop die Weiterverarbeitung der Anfrage gesteuert. Dafür unterscheidet das PostgreSQL-System zwischen einfachen und komplexen Anfragen. Zu den komplexen Anfragen gehören z.B. die SELECT- und UPDATE-Statements (Anfragen) und zu den einfachen Anfragen gehören z.B. das CREATE- und das ANALYZE-Statement. Die einfachen Anfragen können durch die so genannten Utility Commands bearbeitet werden.

- **Rewriter:**

Komplexe Anfragen werden an den Rewriter weitergereicht. Er schreibt Teile der Anfrage um. So werden Views (Sichten) auf die Basisrelationen umgeschrieben, denn Views existieren nur als definierte Betrachtung auf die Daten und repräsentieren keine physischen Tabellen. Der Rewriter traversiert den ParseTree und wendet nacheinander die Regeln des Regelsystems an. Dadurch entsteht ein neu geschriebener Baum, der an den Generate-Plan weiter gereicht wird. Dieser wird für jeden Pfad eine Kostenschätzung durchführen, um einen optionalen Ausführungsplan zu erstellen.

- **Excecutor:**

Er verarbeitet den vorgegebenen Plan. Dabei durchläuft der Executor diesen Plan und führt ihn nacheinander aus. Der Executor beendet seine Arbeit, wenn er die Wurzel des PlanTrees abgearbeitet und das Resultat gebildet hat.

5.5.3 Anfragemöglichkeit von eXist

XQuery ist die primäre Anfragesprache von eXist. Die XQuery-Implementierung ist aus der Spezifikation des 3WC vom 8 Juni 2006. Der Anfrageprozessor ist speziell für eXist entwickelt worden. Eine Anfrage bezieht sich auf ein ganz bestimmtes XML-Dokument innerhalb einer Kollektion oder auf eine ganze Kollektion mit allen darin enthaltenen Dokumenten. Nach der Einleitung durch das Schlüsselwort „*update*“ können fünf verschiedene Manipulationshandlungen vorgenommen werden:

- *Einfügen*
  Einfügen von Knotenfolgen oder eines Einzelknoten in einen oder mehrere Knoten mit drei möglichen Positionsangaben relativ zum Ziel.
- *Ersetzen*
  Ersetzt eine Knotenmenge durch eine andere Knotenmenge.

- *Inhaltswertmodifikation*

Der Inhalt einer Knotenfolge oder eines Einzelknotens wird durch diesen Ausdruck verändert (allenfalls bei mehreren Werten kumuliert zu einem Wert).

- *Löschen*
  Löscht eine Knotenfolge oder einen Einzelknoten.
- *Umbenennen*
  Ändert den Namen eines Elements oder Attributs auf eine neue Zeichenfolge.

Nutzung der Unterstützung von XUpdate
Um XML-Dokumente innerhalb eXist zu manipulieren (Datenmanipulation), steht XUpdate zur Verfügung. Es besteht die Möglichkeit, innerhalb eines XML-Dokuments, Name eines Elements oder Attributs auf eine neue Zeichenfolge zu ändern.
XPath ist auch unterstützt. In eXist werden häufig verwendete Achsen in Anfragen automatisch mit einem Strukturindex indiziert. Es können auch optional Volltextindizes auf Dokumentteilen definiert werden. Beispiel eines Strukturindex unter eXist: //site/regions/africa

5.5.4 Bewertung der Anfragemöglichkeit

Systemname / Kriterien	Berkeley DB XML	eXist	PostgreSQL
XPath	√	√	√
XQuery	√	√	×
XUpdate	×	√	×
SQL	×	×	√
SQL/XML	×	×	√

Tabelle 5- 5 Bewertung der Anfragemöglichkeit

5.6 TRANSAKTIONSVERWALTUNG

Die Datenbasis einer Datenbank soll konsistent gehalten werden, d.h. Operationen wie Schreiben und Lesen dürfen nicht zum ungewollten Verlust oder zur Verfälschung der Daten führen. Bei Fehlern innerhalb einer Transaktion kann dann durch ein *Rollback*[7] zum ursprünglichen Zustand zurückgekehrt werden. Das impliziert aber, dass korrekt durchgeführte Transaktionen am Ende durch ein *Commit*[8] positiv bestätigt werden müssen.

[7] dt.: zurückrollen

[8] dt.: durchführen

5.6.1 Transaktionsverwaltung bei Berkeley DB XML

BDBXML erbt einige Funktionen von Berkeley DB. Dazu gehört nicht zuletzt dessen Transaktionsverwaltung. Als Konkurrenzsteuerung werden Sperrmechanismen eingesetzt. Die Anzahl der Sperren wird bei der Erstellung der Datenbankumgebung mit dem Befehl *EnvironmentConfig setMaxLocks* bestimmt. Die so genannte Funktion *autocommit* erlaubt der Benutzerapplikation, sich nicht um die Transaktionssteuerung zu kümmern.
Die Transaktion von BDBXML hat die ACID Eigenschaften. Standardmäßig verwendet BDBXML keine Transaktionen. Um diese zu aktivieren, müssen folgende Schritte in der eigenen Applikation durchgeführt werden:

- Die Datenbankumgebung (repräsentiert durch ein Objekt der Java Klasse *Environment*) muss in ihrer Konfiguration für die Transaktion vorbereitet werden. Diese geschieht in der Programmiersprache Java über die Methode *setTransactional* der Klasse EnvironmentConfig.
- Die Instanz der Klasse XmlManager muss ebenfalls für Transaktionen konfiguriert werden. Dies geschieht dadurch, dass man ein wie oben beschriebenes Datenbankumgebungsobjekt dem Construktor von XmlManager in Form eines Arguments übergibt.

Der Kontext einer Transaktion wird durch ein Objekt der Klasse XmlTransaktion verkörpert. Dieses Objekt übergibt man jeder Lese- oder Schreiboperation, die Teil der Transaktion sein soll. Sind alle Operationen aufgerufen worden, sorgt die Methode *Commit* im XmlManager für die persistente Speicherung aller durchgeführten Änderungen in der Datenbank.
BDBXML verwendet dieselben drei Isolationsstufen wie Berkeley DB, die in [BERTRA] vorgestellt werden:

- Repeatable Reads
  Dies ist die Form der Isolation von Transaktionen in Berkeley DB XML. Sie sichert einer Transaktion demnach eine gewisse Konstanz des Datenwertes zu, sofern die Transaktion den Wert nicht manipuliert.
- Read Uncommitted
  Auf dieser Stufe können Daten von einer Transaktion gelesen werden, welche noch von einer anderen Transaktion manipuliert werden.
- Cursor Stability
  Diese legt fest, dass ein Datum nicht manipuliert wird, solange es von einem Zeiger (engl. Cusor) verwendet wird.

5.6.2 Transaktionsverwaltung bei PostgreSQL

In PostgreSQL werden Sperren eingesetzt, die minimal auf ein Tupel wirken. Speichert man ein XML-Dokument unstrukturiert im XML Typ, kommt dies einer Sperrung auf Dokumentstufe gleich. Möchte man ein XML-Dokument auf Knotenstufe sperren, dann muss der XML Type eine strukturierte Speicherung anhand eines XML-Schemas vornehmen. Die Sperrarten sind zum Beispiel exklusiv (engl. exclusive) oder geteilt (engl. shared). Die Sperren werden automatisch durch die Sperrverwaltungskomponente des

DBMS gesetzt und können auch manuell eingesetzt werden. Die verschiedenen Sperrmodi sind in [PGSPER] zu finden.
PostgreSQL implementiert zwei Isolationsstufen:

- Read Commited
  Diese Stufe ist per Standard gesetzt und bezweckt, dass eine Anfrage nur auf Daten wirkt, die vor ihrer Ausführung bereits commited war. Damit wird die Anomalie, die auch *Dirty Read* genannt wird, gelöst.
- Serializable
  Auf dieser Isolationsstufe werden nur Daten betrachtet, die vor Transaktionsbeginn endgültig (commited) waren. Dadurch wird die Anomalie, wie zum Beispiel *Dirty Read, Non Repeatable Read* gelöst.

Eine Anfrage innerhalb einer Transaktion kann mit der Multiversionskonkurrenz-Steuerung Leserkonsistenz gewährleistet werden.

5.6.3 Transaktionsverwaltung bei eXist

Die Transaktionsverwaltung bei eXist ist rudimentär und man kann nicht darauf zugreifen. Transaktionen werden nur intern implizit verwendet, sobald Schreiboperationen den Datenbestand ändern. Leseoperationen können nicht Teil von Transaktionen sein, weil es zu diversen Anomalien kommen kann. Dies hängt von der Art der Sperre ab. Sperrt eine Transaktion ein gesamtes XML-Dokument für alle Zugriffe während dessen gesamter Manipulation, kann eine Leseoperation keine inkonsistenten Daten lesen.

Systemname / Kriterien	Berkeley DB XML	eXist	PostgreSQL
Isolationstuffe	√	√/×[9]	√
Sperre	√	√	√
ACID-Eigenschaften	√	√/×	√

Tabelle 5- 6 Bewertung der Transaktionsverwaltung

5.7 LIZENZIERUNG

In diesem Abschnitt werden die verfügbaren Lizenzen der einzelnen XML-Datenbanken vorgestellt.

5.7.1 Lizenzierung von Berkeley DB XML

Berkeley DB XML wird unter einer dualen Lizenz vertrieben. Dual bedeutet, dass BDXML unter zwei Lizenzen verfügbar ist. Die erste Lizenz ist die GNU GPL (General

[9] teilweise erfüllt

Public Licence) und deckt die Fälle ab, in denen BDBXML kostenlos als Open Source verwendet werden kann. Diese Lizenz hat als Hauptziel, die veröffentlichte Software mit dem Quell-Code auszuliefern. Die zweite Lizenz ist kommerziell und ist innerhalb proprietärer Produkte verfügbar. Der Wortlaut der GNU GPL befindet sich in [GNUGPLLi]

5.7.2 Lizenzierung von PostgreSQL

PostgreSQL wird unter BSD-Lizenz (Berkeley Software Distribution) herausgegeben, welche schwächere Bedingungen als die GNU GPL besitzt. Jede Software, die unter dieser Lizenz steht, muss einen speziellen Text im Quelle-Code beinhalten. Dieser Text ist in [BSDTEXT] zu finden. Hier werden die Erlaubnisse zum Modifizieren, Kopieren und Vertreiben von PostgreSQL-Software ohne Entgelt und ohne Vertrag gegeben. Daher ist eine kommerzielle Lizenz nicht erforderlich.

5.7.3 Lizenzierung von eXist

Exist wird unter GNU LGPL-Lizenz (Lesser General Public Licence) herausgegeben. Grundsätzlich darf eine unter LGPL lizenzierte Software nur zusammen mit ihrem Quelletext vertrieben werden.

5.7.4 Bewertung

Systemname / Kriterien	Berkeley DB XML	eXist	PostgreSQL
BSD-Lizenz	×	×	√
Kommerzielle-Lizenz	√	×	×
GPL-Lizenz	√	×	×
LGPL-Lizenz	×	√	×

Tabelle 5- 7 Bewertung der Lizenzierung

KAPITEL 6 TESTDURCHFÜHRUNG

In diesem Kapitel werden die XML-Datenbanken aus der engeren Auswahl hinsichtlich deren Performanz getestet und verglichen.

6.1 *GRUNDLAGEN DES KONZEPTIONELLEN VERGLEICHS*

Im Folgenden werden die Vorgehensweise und die eingesetzte Strategie im Rahmen des konzeptionellen Vergleichs der untersuchten XML-Datenbanken erläutert.
Wie bereits im Abschnitt 2.2.4 diskutiert, wird zur Leistungsmessung der untersuchten XML-Datenbanken der XML-Benchmark XMARK eingesetzt.
Der Grund liegt in der großen Akzeptanz von XMARK auch unter Produktherstellern als erste Wahl für Tests. Außerdem ist XMARK mittlerweile sehr ausgereift.
Eine grundlegende Einführung in XML-Benchmarks und XMARK wurde schon in Kapitel 2 gegeben. Zusammengefasst liegen die Stärken von XMark in der Unterstützung von XQuery sowie der Messung der Schnelligkeit der Anfrageverarbeitung.
Die Datenbankinstanz wird bei jedem Produkt über die produktspezifischen Administrationswerkzeuge erstellt und mit den Daten der XMark-Benchmark geladen. Die Daten werden, wie in [XMARK] spezifiziert, mit dem offiziellen Generator erstellt. Zudem werden drei verschiedene Dokumentgrößen verwendet, die eine XML-Datenbank unterschiedlich beanspruchen.
Tabelle 6-1 gibt eine Übersicht über die Erstellung der Daten. In [HowToXmlgen] findet sich eine Beschreibung der Shell Kommandoparameter.

Beschreibung	Dateiname	Dateigröße	Skalierungs-Faktor	Shell Kommando
Größtes XML-Dokument, sehr umfangreiche Daten	BigDoc.xml	22 MB	0.150	xmlgen.exe /i /f 0.150 /o BigDoc.xml
Mittelgroßes XML-Dokument	MiddleDoc.xml	14,8 MB	0.10	xmlgen.exe /i /f 0.10 /o MiddleDoc.xml
Kleinstes XML-Dokument, sollte von jedem Produkt geladen werden können	SmallDoc.xml	226 KB	0.0017	xmlgen.exe /i /f 0.0017 /o SmallDoc.xml

Tabelle 6- 1 Beschreibung der generierten Daten für die Performanztests

Metrik

Die Ausführung der Anfragen erfolgt im Einbenutzerbetrieb in derselben Testumgebung, wobei jede Operation einzeln betrachtet wird. Als Resultat liefert der Benchmark die Ausführungszeiten, das Ergebnis für die einzelnen Operationen, welche dann protokolliert und ausgewertet werden.
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. Nebenbei sei erwähnt, dass ein von einigen Systemen vorgenommenes Caching von Anfrage-Antwort-Paaren die durchschnittliche Antwortzeit vom Ausmaß der Caching-Effekte beeinflussen kann.

Vorgehensweise

Grundsätzlich wird jede XML-Datenbank erst auf dem Testrechner installiert, bevor der Test durchgeführt wird. Die Performanzmessung erfolgt in diversen Versuchen. Die Versuche sind in der Regel vom Prinzip her ähnlich aufgebaut und in Testreihen zusammengefasst.
Je Testreihe werden alle Anfragen aus XMark verwendet und pro Datenbankinstanz wird nur ein XML-Dokument geladen. Nach jeder Testreihe wird das XML-Dokument ausgetauscht, sodass jede Anfrage auf jedes der drei XML-Dokumente in einer Tabelle dargestellt wird. Damit hat jedes XML-Datenbankprodukt die gleichen Daten und Anfragen auf demselben Testrechner. Eine Übersicht der gestellten XQuery-Anfragen aus Xmark befindet sich im Anhang A.
Auf eine manuelle Performanzmessung der untersuchten XML-Datenbanken wird in dieser Arbeit verzichtet. Stattdessen wird ein eigenständiges Java Programm zu diesem Zweck entwickelt. Durch diese Vorgehensweise wird die Messarbeit erheblich präziser, als in einem rein manuellen Vorgang. Außerdem wird dadurch grundsätzlich jede XML-Datenbank mit derselben Zugriffsvariante verwendet, nämlich über die Java API, anstatt über individuelle Benutzerschnittstellen. Das entwickelte Java-Programm trägt den Namen *„Easy XML-Datbase-Benchmark“*. Seine Implementierung wird später beschrieben.
Je Produkt müssen die Anfragen, welche alle in XQuery erfolgen, leicht verändert werden, sodass sie kompatibel zum jeweils implementierten Standard sind. In der Semantik und Struktur werden die Anfragen weitestgehend belassen.
Abbildung 6-1 skizziert die Vorgehensweise bzw. den Verlauf der Performanzmessung.

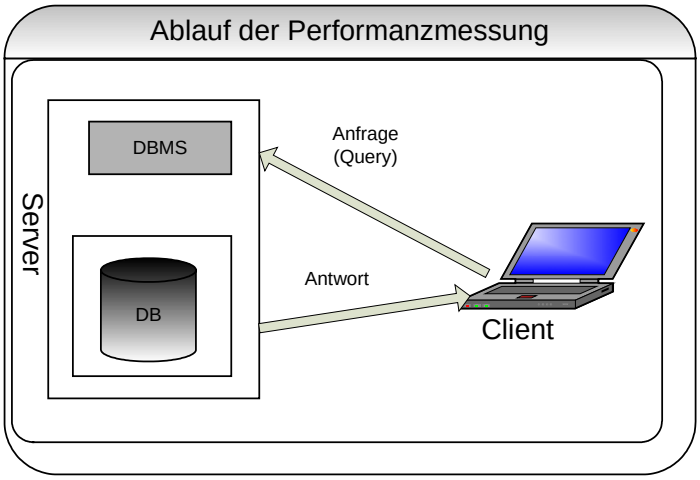


Abbildung 6- 1 Verlauf der Performanzmessung

6.2 TESTUMGEBUNG UND ENTWICKLUNGSUMGEBUNG

Die Tests bzw. Leistungsvergleiche werden auf einem Windows-XP basierenden, mobilen Rechner mit 1,66 GHZ Intel Core Duo 2 CPU Prozessor, 1 GB RAM und 120 GB Festplattenspeicher ausgeführt. Alle XML-Datenbanken werden zwecks Vergleichbarkeit auf derselben Testumgebung installiert. Die Details sind in der Tabelle 6-2 zu finden.

Komponente	Beschreibung /Konfiguration
Grundmodell	Mobiler Rechner von Samsung, Centrino Technologie von Intel
Betriebssystem	Windows XP Professional 2002 mit Service Pack 2.
Prozessor	Intel Core Duo
RAM	1GB
Dateisystem	New Technology File System (NTFS)
Entwicklungsumgebung	Eclipse 3.3.1.1
Java-Umbebung	Java 6 (JDK 1.6.03)
Browser	Microsoft Internet Explorer 6.0

Tabelle 6- 2 Detaillierte Beschreibung des Testrechners

6.3 EASY XML-DATABASE-BENCHMARK

Easy XML-Database-Benchmark ist eine Java basierte Applikation, welche im Rahmen dieser Arbeit entwickelt wird. Die Idee dabei ist, ein Werkzeug zur Verfügung zu stellen, welches die Performanzmessung der untersuchten XML-Datenbanken automatisieren kann. Dadurch wird nicht nur die Performanzmessung präziser durchgeführt, sondern es lässt sich auch bewerten, inwieweit die Query-Prozessoren der jeweiligen Datenbanken sich innerhalb einer Drittanwendung fernsteuern lassen. Zudem werden zur Implementierung und Realisierung von *Easy XML-Database-Benchmark* die zur Verfügung gestellten APIs der untersuchten XML-Datenbanken verwendet.
Die Architektur von *Easy XML-Database-Benchmark* findet sich in Abbildung 6-2.

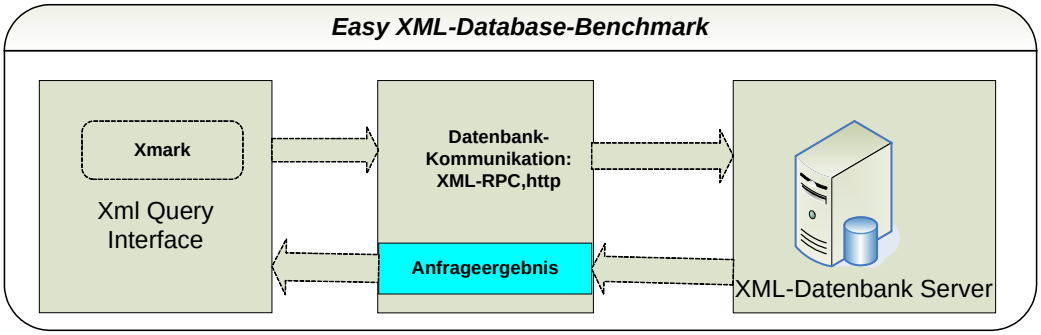


Abbildung 6- 2 Architektur von *Easy XML-Database-Benchmark*

Beschreibung zum grundsätzlichen Design und zur Implementierung

Der Entwurf des Systems folgt konsequent dem MVC-Pattern siehe [MVCPCL]. Um ein möglichst flexibles Testsystem zu implementieren, wurde darauf geachtet, dass es wieder verwendbar, austauschbar, erweiterungsfähig, plattformunabhängig und präzise in der Messung ist. Das Testsystem bietet gleiche, faire Bedingungen für jede Messung und die Fehlerbehandlung.
Bei der Implementierung wurde darauf geachtet, die grafischen Komponenten in ihren grafischen Klassen zu belassen und die Datenbankkomponenten in ihre Datenbankklassen zu kapseln. Wichtig ist, dabei auch die Trennung zwischen Code und programmspezifischen Daten einzuhalten.
Zu den Daten der Anwendung zählen dabei z.B.:

- die Datenbanknamen
- 20 unterschiedliche Queries von je 3 Datenbanken
- XML-Dokumentnamen

Über separate Klassen soll auf die Daten zugegriffen werden können. Die Daten wiederum werden in separaten „Ini-Dateien“ gespeichert. Das Ergebnis soll sofort sichtbar, nachvollziehbar und selbsterklärend sein. Jeder Messwert soll protokolliert und in eine Exceltabelle exportierbar sein, um eine Analyse der Grafik und eine Bewertung abgeben zu können.
Die grafische Bedienoberfläche (GUI) des *Easy XML-Database-Benchmark* genauso wie das Klassendiagramm sind in Abbildung 6-3 bzw. 6- 4 zu sehen.

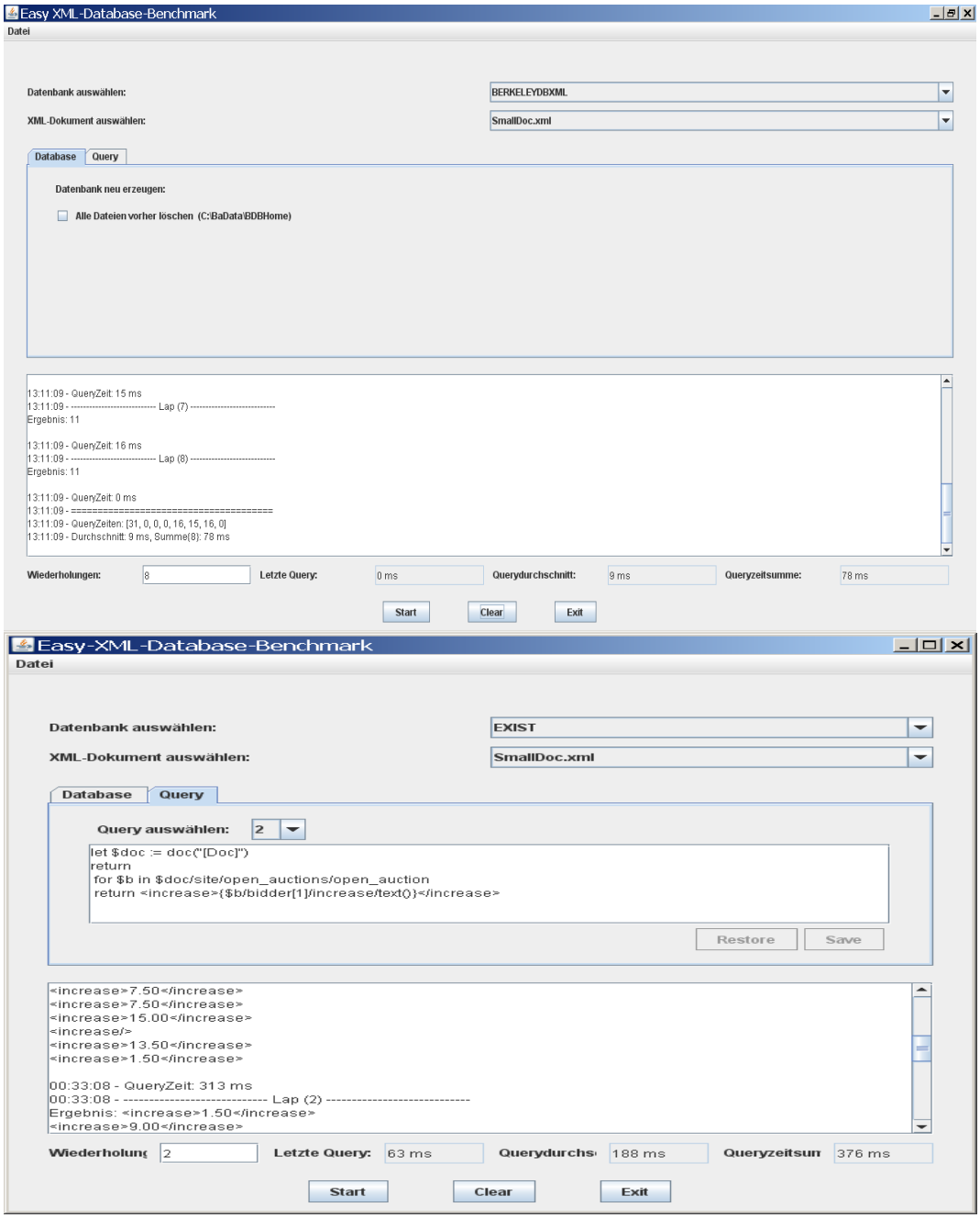


Abbildung 6- 3 Easy XML-Database-Benchmark

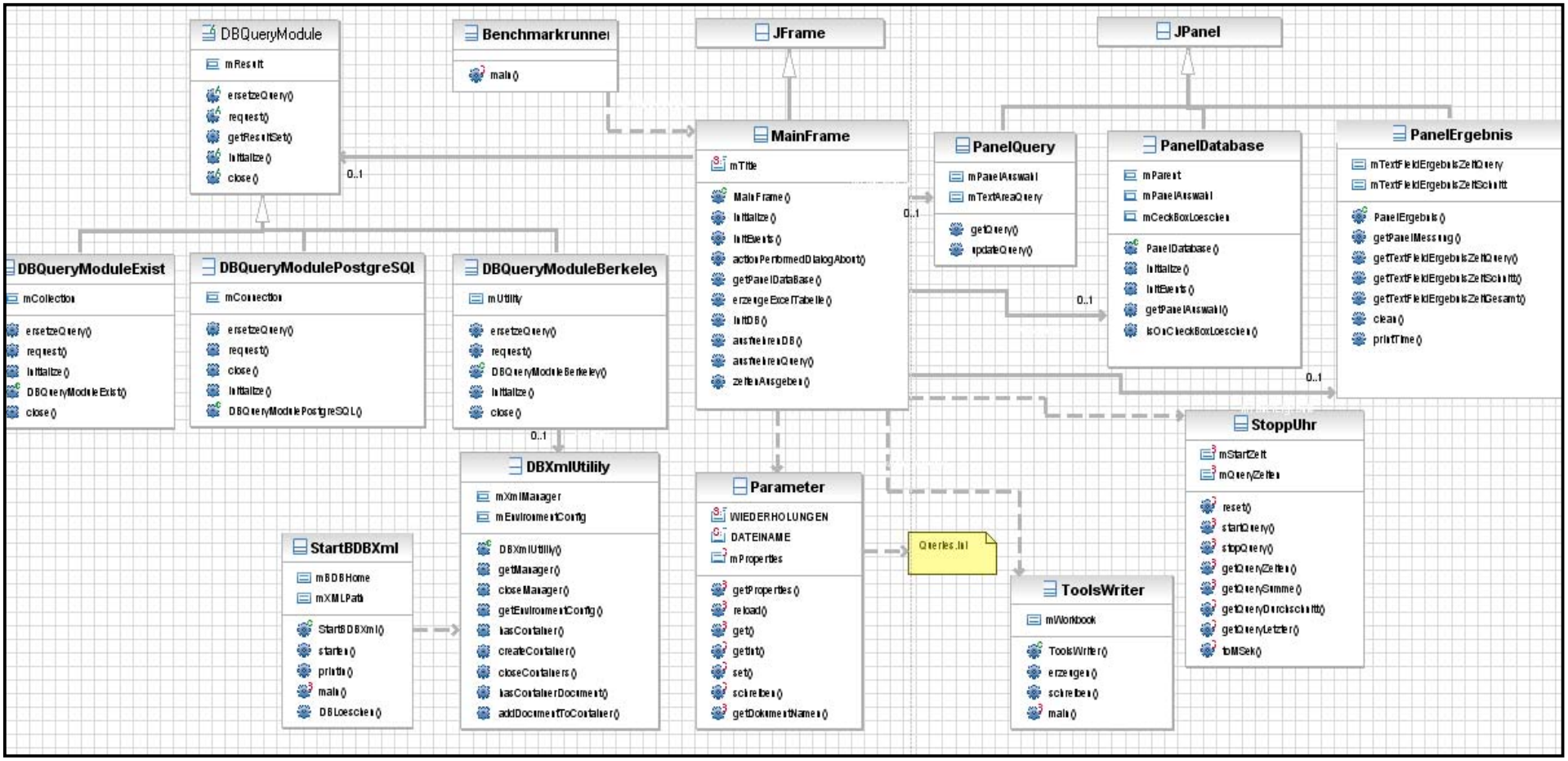


Abbildung 6- 4 Klassediagramm des Testsystems

Um ein besseres Verständnis von *Easy XML-Database-Benchmark* zu ermöglichen, wird im Folgenden auf einige Aspekte seiner Programmierung eingegangen.
Damit das Programm die im Rahmen dieser Arbeit untersuchten Datenbanken unterstützt, werden Bibliotheken der entsprechenden Datenbanken im Programmcode referenziert, welche im Installationsordner der gewählten XML-Datenbanken zu finden sind. Diese werden nachfolgend nach Datenbanken gruppiert und dokumentiert.

Für eXist:

antlr-2.7.6.jar	commons-pool-1.2.jar
xcalibur-cli-1.0.jar	avax.servlet.jar
jEdit-syntax.jar	groups-all.jar
jline-0_9_5.jar	log4j-1.2.14.jar
resolver.jar	sunxacml.jar
xmldb.jar	xmlrpc-1.2-patched.jar
exist.jar	exist-modules.jar

Für Berkeley DB XML:

dbxml.jar	db.jar

Für PostgreSQL:

pljava.jar	Jdbc.jar

Das Programm wurde so entworfen, dass pro Datenbank eine Java-Schnittstelle zur Interaktion mit der Datenbank bereitgestellt wird. Die Benennung dieser Schnittstelle entspricht der Schablone *DBQueryModule*[*Datenbankname*].
Der darunter liegende Programmcode ist ein Ausschnitt der Klasse *DBQueryModuleExist*. In dieser Klasse wird die zur Verfügung gestellte API der Datenbank genutzt. Schließlich wird die Anfrage an die Datenbank mit Hilfe der XML:DB-API verschickt.

```
public class DBQueryModuleExist extends DBQueryModule {
    protected String mDriver = null;
    protected String mURI = null;
    protected String mUser = null;
    protected String mPassword = null;
    protected Collection mCollection = null;
    protected XQueryService mService = null;

    public DBQueryModuleExist() throws Exception {
        mDriver = "org.exist.xmldb.DatabaseImpl";
        mURI = "xmldb:exist://localhost:8080/exist/xmlrpc/db";
        mUser = Parameter.get("DBUser_Exist");
        mPassword = Parameter.get("DBPass_Exist");
        initialize();
    }
    protected void initialize() throws Exception {
        Database lDatabase = (Database) Class.forName(mDriver).newInstance();
        DatabaseManager.registerDatabase(lDatabase);

        // Collection erzeugen:
        mCollection = DatabaseManager.getCollection(mURI);

        mService = (XQueryService) mCollection.getService("XQueryService", "1.0");
        mService.setProperty("indent", "yes");
    }
    public String ersetzeQuery(String aQuery, File aDatei) {
        String lQuery = aQuery.replace(DOC, aDatei.getName());
        return lQuery;
    }
     public void request(String aQuery) throws Exception {
          CompiledExpression lCompiled;
          lCompiled = mService.compile(aQuery);
          ResourceSet lResultSet;
          // ---------------------------------
          StoppUhr.startQuery();

          lResultSet = mService.execute(lCompiled);

          StoppUhr.stopQuery();
          // ---------------------------------
          ResourceIterator lIterator = lResultSet.getIterator();
          mResult = "";
          while (lIterator.hasMoreResources()) {
               Resource lResource = lIterator.nextResource();
               mResult += (String) lResource.getContent() + "\n";
          }
     }
     public void close() throws Exception {
          if (mCollection != null) {
               mCollection.close();
          }
     }
}
```

Abbildung 6- 5 Programmcode von DBQueryModuleExist

Ähnlich wurden die Klassen *DBQueryModuleBerkeley* und *DBQueryModulePostgreSQL* implementiert.

```
3  public class Benchmarkrunner {
4
5      public static void main(String[] args) {
6          MainFrame lFrame = new MainFrame();
7          lFrame.setVisible(true);
8      }
9
10 }
11
```

Beschreibung zur GUI - Implementierung:
Die ***main*** -Methode ist in der Klasse *Bechnmarkrunner* implementiert.
Mit dem Aufruf der Methode *setVisible(true)* wird die GUI auf dem Bildschirm des Endanwenders dargestellt. Das Herzstück der Anwendung ist der *MainFrame* (geerbt vom JFrame), der die GUI darstellt und die Interaktion zwischen User und der Datenbankausführung implementiert. Die genaue Beschreibung befindet sich im Anhang F.
Die Abbildung 6-6 stellt das Klassendiagramm der Datenbankmodule dar.

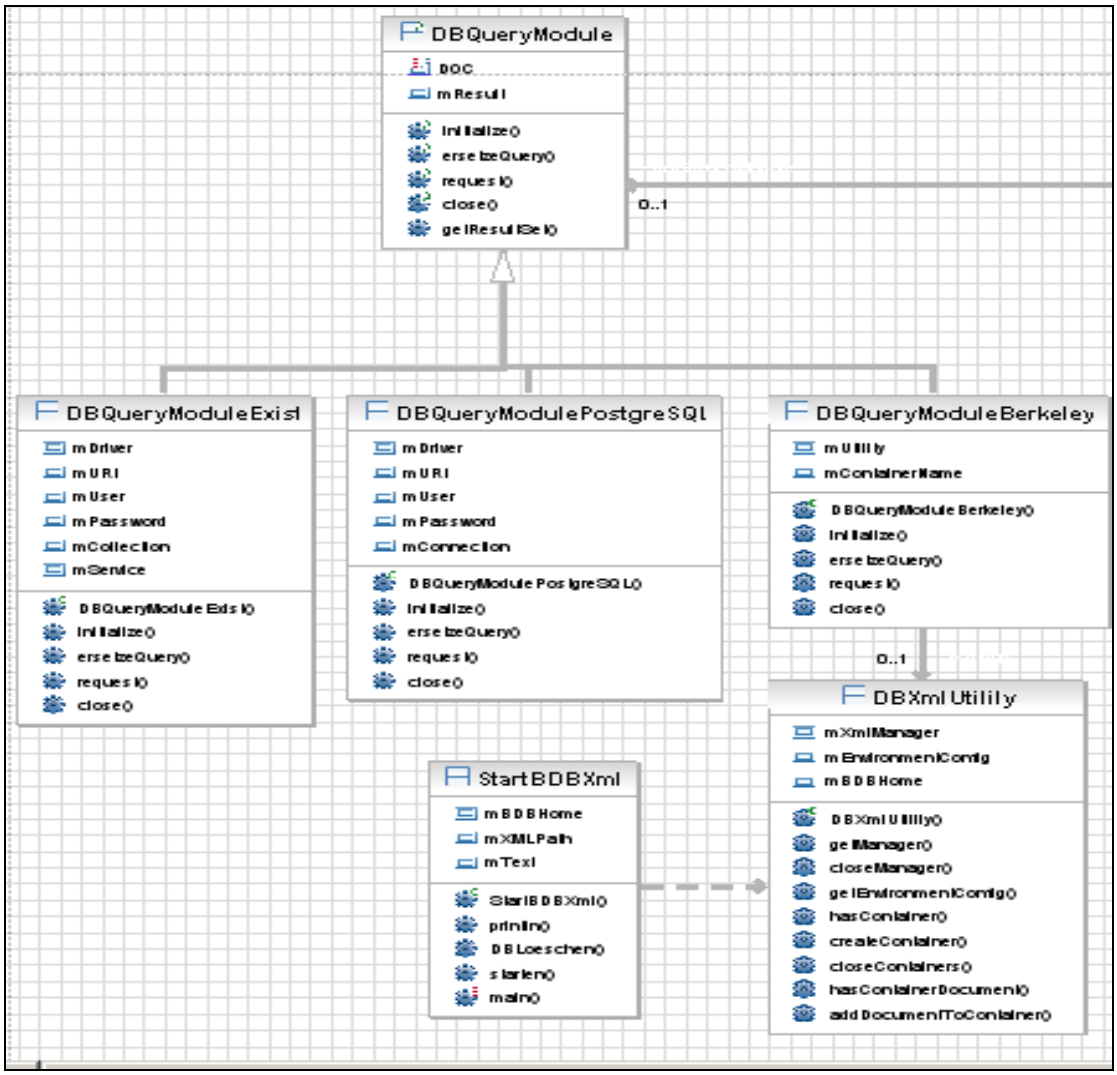


Abbildung 6- 6 Klassendiagramm der Datenbankmodule

Die Klasse *MainFrame* implementiert die abstrakte Klasse *DBQueryModule*. Entsprechend der ausgewählten Datenbank wird zur Laufzeit das DBQueryModule erzeugt:

```
408     private void initDB() throws Exception {
409         if (mDBQueryModule == null) {
410             String lDBName = getComboBoxDatabase().getSelectedItem().toString();
411             if (lDBName == DBQueryModule.EXIST) {
412                 mDBQueryModule = new DBQueryModuleExist();
413             } else if (lDBName == DBQueryModule.BERKELEYDBXML) {
414                 mDBQueryModule = new DBQueryModuleBerkeley();
415             } else if (lDBName == DBQueryModule.POSTGRESQL) {
416                 mDBQueryModule = new DBQueryModulePostgreSQL();
417             }
418         }
419     }
```

Das komplette Programm von *Easy XML-Database-Benchmark* befindet sich auf der mitgelieferten CD

Vorgehensweise bei der Messung der Anfrageverarbeitung

Hier werden die Strategien bzw. die von mir entwickelte Software *„Easy XML-Database-Benchmark“*, die im Abschnitt 6.3 dargestellt wurde, verwendet, um den Performanztest durchzuführen. Es sollte darauf geachtet werden, dass die Prozesse des Datenbanksystems nicht aktiv sind, wenn sie nicht gebraucht werden, da die Systemressourcen des Testrechners bei Weitem nicht ausreichen, um alle Produkte gleichzeitig aktiv laufen zu lassen.
Alle zwanzig Anfragen aus der Xmark werden zum Zweck des Performanztests verwendet. Die Anfragen 1 bis 20 werden hintereinander ausgeführt, bevor das Dokument ausgetauscht wird. Je Produkt müssen die Anfragen, welche alle in XQuery erfolgen, leicht geändert werden, sodass sie zum jeweils implementierten Standard kompatibel sind.
Für die Verarbeitung einer Anfrage steht maximal eine Stunde[10] zur Verfügung. Danach gilt die Anfrageverarbeitung als beendet. Die Verarbeitungsdauer einer Anfrage, die vom Datenbanksystem mehr als eine Stunde beträgt, wird der maximal möglichen Anfragedauer von einer Stunde gleichgesetzt. Die Verarbeitungsdauer einer Anfrage, die von einer Datenbank nicht verarbeitet werden kann, wird der maximal möglichen Anfragendauer von einer Stunde gleichgesetzt.
Es werden insgesamt acht Messungen durchgeführt. Die erste Messung wird als Aufwärmung des Systems betrachtet und daher das Ergebnis gestrichen. Bei der Messreihe werden der langsamste und der schnellste Messwert gestrichen. Diese könnten durch Änderungen in der Speicherverwaltung (Cache Größe bzw. deren Belegung) und

[10] 1 Stunde = 3'600'000 Millisekunden

in der (automatischen) Indizierungsstrategie entstehen. Außerdem gibt es einen unkontrollierbaren Einfluss, wie die Prozessverwaltung des Betriebssystems.
Über die verbliebenen fünf Messwerte wird je Anfrage das arithmetische Mittel berechnet. Sämtliche Messergebnisse können in Anhang D eingesehen werden.

Überblick über die Vorgehensweise

- Acht Messungen nacheinander mit Anfragen 1 bis 20 auf SmallDoc.xml
- Acht Messungen nacheinander mit Anfragen 1 bis 20 auf MiddleDoc.xml
- Acht Messungen nacheinander mit Anfragen 1 bis 20 auf BigDoc.xml
- Verwerfen der ersten Messung
- Verwerfen der jeweils langsamsten und schnellsten Messung

6.5 MESSERGEBNISSE AUS ALLEN ANFRAGEN

In diesem Abschnitt werden die Ergebnisse der Anfrageverarbeitungsdauer aller zwanzig Anfragen je XML-Dokument aufgezeigt. Die Werte basieren auf dem arithmetischen Mittelwert der fünf verbliebenen Ergebnisse.
Zuerst werden die Ergebnisse mit SmallDoc.xml gezeigt. Es handelt sich um das kleinste XML-Dokument im Performanztest. Die getrimmten gemittelten Messwerte mit SmallDoc.xml sind in Tabelle 6-3 dargestellt.

	BDBXML	eXist	PostgreSQL
Q01	1	29	15
Q02	16	18	9
Q03	18	28	3600000
Q04	21	87	3600000
Q05	12	15	3600000
Q06	37	6	3600000
Q07	40	11	3600000
Q08	34	37	3600000
Q09	47	14	3600000
Q10	62	37	3600000
Q11	62	63	3600000
Q12	31	56	3600000
Q13	3	16	3600000
Q14	40	15	3600000
Q15	15	39	3600000
Q16	12	36	3600000
Q17	12	16	3600000
Q18	9	15	3600000
Q19	21	18	3600000
Q20	31	18	3600000

Tabelle 6- 3 Gemittelte Werte über alle Anfragen mit SmallDoc.xml

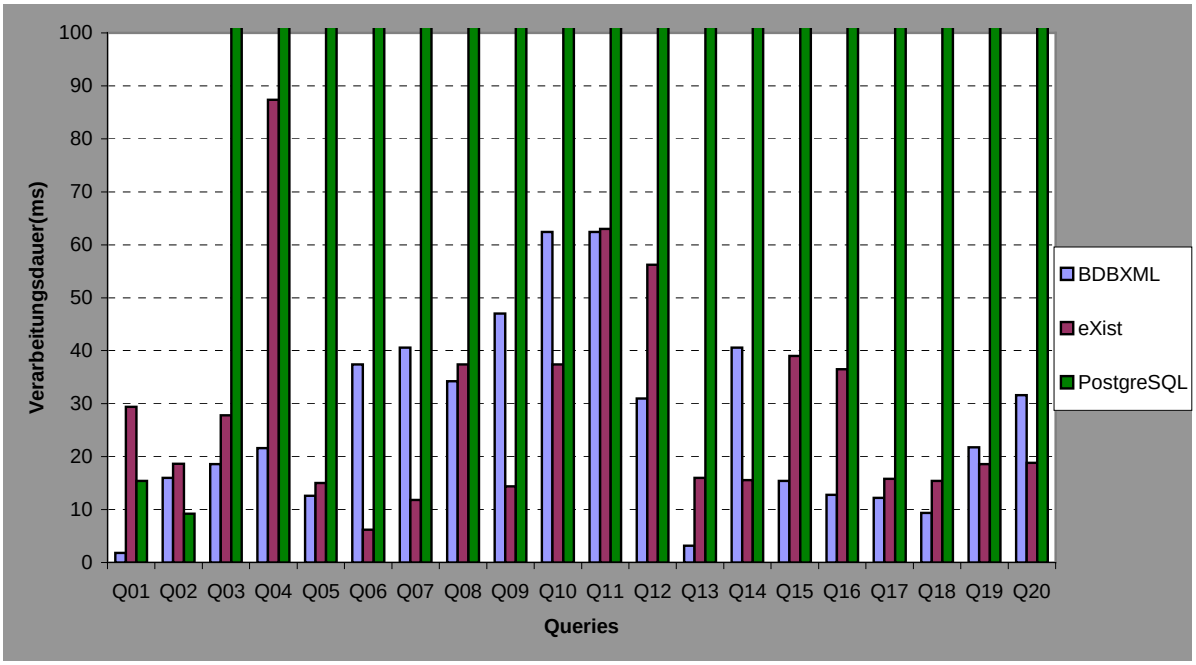


Abbildung 6- 7 Verarbeitungsdauer mit SmallDoc.xml

Die wichtigen Messwerte befinden sich zwischen 0 und 100. Auf diesem Grund wird nur dieser Bereich Abbildung 6-7 dargestellt.

Im Falle des mittelgroßen XML-Dokuments namens MiddleDoc.xml, zeigt sich das folgende Bild (Abbildung 6-8). Die Tabelle 6-4 zeigt die zugehörigen, getrimmten und gemittelten Messwerte mit MiddleDoc.xml.

	BDBXML	eXist	PostgreSQL
Q01	90	40	1259
Q02	349	150	614
Q03	856	299	3600000
Q04	1448	12150	3600000
Q05	137	62	3600000
Q06	13315	15	3600000
Q07	13965	19	3600000
Q08	44525	63375	3600000
Q09	57931	145890	3600000
Q10	9737	5868	3600000
Q11	99649	182297	3600000
Q12	32737	235487	3600000
Q13	109	78	3600000
Q14	12322	296	3600000
Q15	187	40	3600000
Q16	218	140	3600000
Q17	281	191	3600000
Q18	281	69	3600000
Q19	784	456	3600000
Q20	390	172	3600000

Tabelle 6- 4 Gemittelte Werte über alle Anfragen mit MiddleDoc.xml

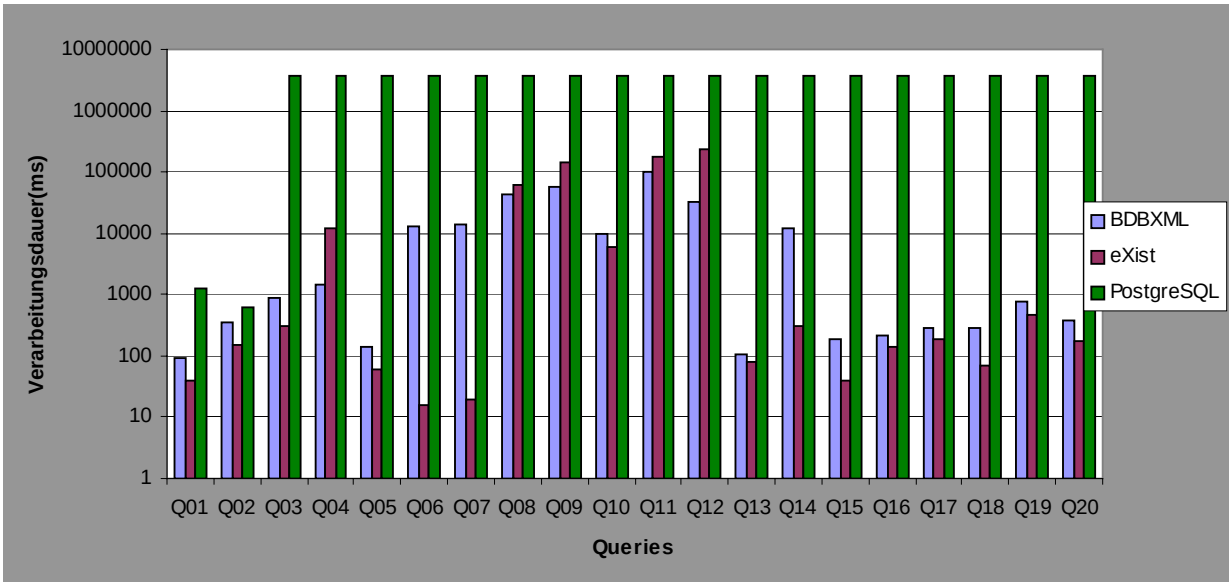


Abbildung 6- 8 Verarbeitungsdauer mit MiddleDoc.xml

Bei der graphischen Darstellung der Messwerte mit MiddleDoc.xml, wurde die

Verarbeitungsdauer aufgrund der großen Messwerteunterschiede logarithmisch dargestellt. Abschließend sollen die Ergebnisse mit BigDoc.xml gezeigt werden. Die zu den Grafiken gehörenden Messwerte sind in Tabelle 6-5 zu finden

	BDBXM	eXist	PostgreS
Q01	118	69	1762
Q02	493	247	915
Q03	1302	456	3600000
Q04	2459	18474	3600000
Q05	219	556	3600000
Q06	28075	35	3600000
Q07	29483	53	3600000
Q08	99665	162410	3600000
Q09	134676	313669	3600000
Q10	15212	12638	3600000
Q11	228266	404550	3600000
Q12	73103	234936	3600000
Q13	143	68	3600000
Q14	25984	368	3600000
Q15	300	97	3600000
Q16	337	165	3600000
Q17	428	284	3600000
Q18	422	103	3600000
Q19	3262	771	3600000
Q20	528	268	3600000

Tabelle 6- 5 Gemittelte Werte über alle Anfragen mit BigDoc.xml

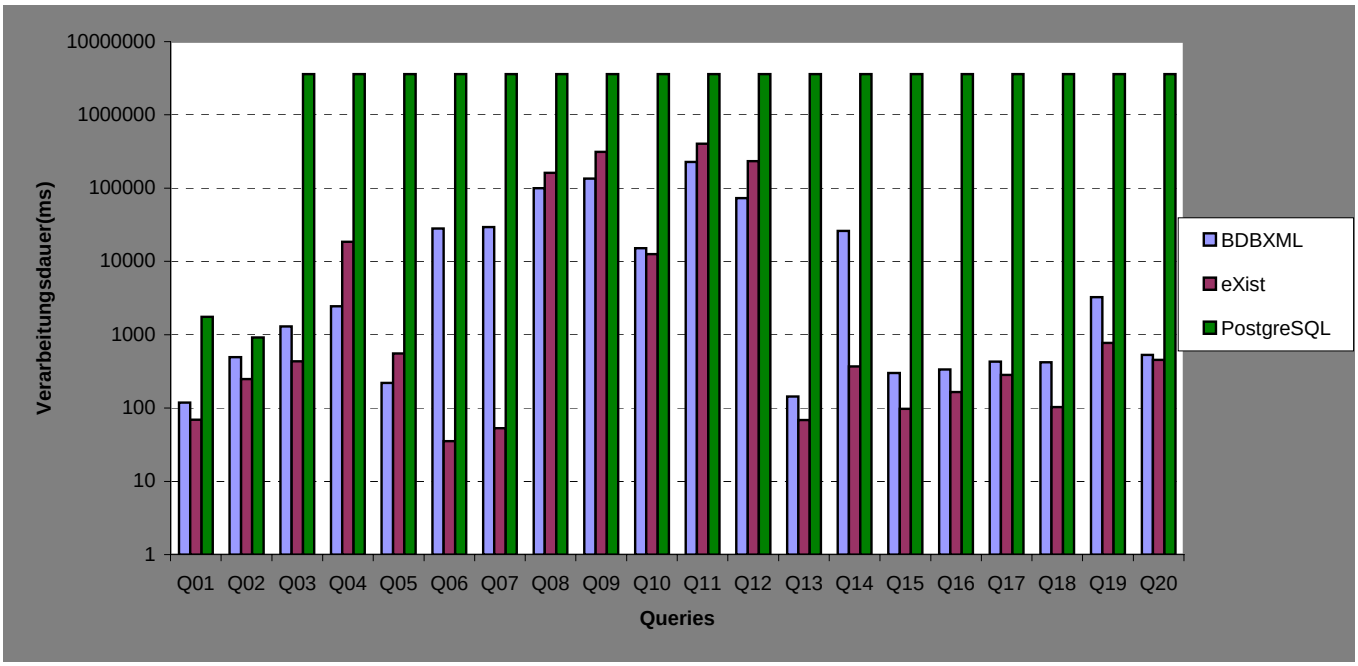


Abbildung 6- 9 Verarbeitungsdauer mit BigDoc.xml

Interpretation der Verarbeitungsdauerergebnisse

In diesem Performanztest war es von Beginn an die Idee, den ausgewählten XML-Datenbanken die gleichen Bedingungen zu bieten. Das heißt konkret dasselbe Testsystem und dieselbe Zugriffvariante über eine selbst erstellte Java Applikation anzuwenden. Dies soll eine Vergleichbarkeit und damit einfache Interpretation der Ergebnisse ermöglichen.

PostgreSQL zeigt insgesamt in dieser Auswertung die schlechteste Performanz. Bei den Anfragen Q01 und Q02 bedarf PostgreSQL für ihre Ausführung mehr Zeit als die anderen XML-nativen Datenbanken eXist und Berkeley DB XML. Bei den Anfragen Q03 bis Q20 gilt PostgreSQL als nicht auswertbar. Da ihr Query-prozessor diese Anfragen nicht verarbeiten konnte, werden diese Anfragen mit einer Maximaldauer von einer Stunde gewertet.

In diesem Zusammenhang sind eXist oder Berkeley DB XML zu vollziehen, da alle Querys erfolgreich ausgeführt werden konnten.

Die Anfrageverarbeitung mit SmallDoc.xml ist schneller. Mit steigender Dokumentgröße, wird die Verarbeitung jeder Anfrage von jeder nativen Datenbankinstanz langsamer.
Für ein besseres Verständnis der Messungen werden die Anfragen, je nach Funktionalität, in Gruppen (siehe Tabelle 2- 8) zusammengefasst und interpretiert.

Bei der Anfrageverarbeitung in Gruppe I (Q01, Q05, Q14), wobei es um einen einfachen Vergleich geht, ist Berkeley DBXML schneller als eXist. Aber mit steigender Dokumentgröße wird eXist die schnellste.

In der Gruppe II (Q02, Q03, Q04, Q11, Q12, Q17), wobei es um die Bewahrung der Element-Ordnung geht, ist Berkeley DB XML bei der Verarbeitung aller Dokumentgrößen schneller als eXist.

In der Gruppe III (Q06- Q10, Q13, Q15, Q16), wo es um Pfadauswertung geht, ist eXist in der Anfrage 6 hingegen erheblich schneller als Berkeley DB XML
Es könnte sein, dass das Nummerierungsschema von eXist eine sehr schnelle Pfadauswertung ermöglicht. Mit steigender Dokumentgröße wird auch die Anfrage 7 zügig verarbeitet. Dabei werden darin die Instanzen eines bestimmten Elementtyps gezählt, was mit steigender Dokumentgröße zeitaufwendiger wird, wie es bei Berkeley DB XML zum sehen ist.

In Gruppe IV (Q18, Q19, Q20), in der es um Aggregation und Sortierung geht, ist eXist deutlich schneller mit steigender Dokumentgröße als Berkeley DB XML.

Performanz von Berkeley DB XML
Bei der Performanz von BDBXML wurden alle zwanzig Anfragen aus XMark verarbeitet. BDXML zeigt einen stabilen Verlauf über alle Anfragemessungen.

Performanz von PostgreSQL
Bei PostgreSQL wurden drei Funktionen in XQuery zum Anfragen der Kardinalität einer Sequenz nicht akzeptiert. Betroffen sind die Queries-Nummer 3, 10, 11, 12, 14, 18, 19. Folgende Funktionen lösen den Fehler aus:

- *fn :zero or one*
- *fn :one or more*
- *fn :exactly one*
- *fn :data*

Weiter entsteht ein Fehler im Konstrukt *some satisfies*. Aufgrund des Fehlers kann die Query-Nummer 4 nicht ausgeführt werden. Die Queries-Nummer 16, 17, 20 scheitern aufgrund der Funktion *fn:empty*. Hier wurde festgestellt, dass es eine relativ hohe Anzahl nicht unterstützter Funktionen bei PostgreSQL gibt.

Performanz von eXist
In eXist wurden alle Anfragen aus XMark problemlos durchgeführt. Da unter eXist der Namensraum Lokal bereits verwendet wird, muss daher in der Anfrage mit der Nummer 18 die Funktionsdeklaration umbenannt werden.

KAPITEL 7 BEWERTUNG

In diesem Kapitel wird eine Gesamtbewertung der Produkte aus der engeren Auswahl dargestellt, sodass eine Empfehlung gegeben werden kann. Diese Empfehlung basiert auf den im Laufe dieser Arbeit gewonnenen Kenntnissen und beruht auf einer Rangfolge aller drei XML-Datenbanken anhand der erreichten Punktzahl. Die Punktzahl hängt wiederum von der Ausprägung des Systems hinsichtlich aller Anforderungskriterien aus dem Anforderungskatalog ab. Die Bewertungstabelle befindet sich aufgrund ihres großen Umfangs in Anhang F.

7.1 VORGEHENSWEISE

Um eine Empfehlung für eine XML Datenbank geben zu können, werden die Kenntnisse aus allen Bewertungsabschnitten (z.B. Plattformunterstützung) beziehungsweise die Erkenntnis aus der Detailbetrachtung im Abschnitt 3.2 mit Punkten versehen. Die minimale Punktzahl für jedes Kriterium ist die Zahl Null. Die maximale Punktezahl ist begrenzt und variiert je nach gewichtetem Einzelkriterium. Es werden auch Punkte zwischen dem Minimum und Maximum vergeben. Nichtperformanzaspekte, z.B. Funktionalität und Plattformunterstützung, halten sich die Waage mit den Performanzkriterien, da eine ungleiche Gewichtung von konkreten Einsatzabsichten und den daraus zu schließenden Anforderungen an die XML-Datenbank abhängt. Hier werden die Kriteriengruppen Funktionalität und Plattformunterstützung zusammen für gleich wichtig wie die Kriteriengruppe Performanz gehalten. Die Gliederung der Punktzahlen folgt der Detailbetrachtung. Die Gewichtung basiert auf den persönlichen Erfahrungen und ist daher nicht als bindend anzusehen. Die Gewichtung spielt nur für den Abschlussrang eine Rolle. Die Ergebnisse werden in Abschnitt 7.2 aufgezeigt.

Grundsätzlich gibt es zwei Typen von Bewertungen:

- Der erste Typ betrifft den Performanztest. Die maximale Punktzahl, die erreicht werden kann, ist für alle XML-Datenbanken und für jede Anfrage zwei und kann je nach eingegebener Gewichtung geändert werden. Vergeben wird die Maximalpunktzahl an die XML-Datenbank, die die beste Ausprägung also kürzeste Verarbeitungsdauer vorweist. Die anderen XML-Datenbanken erhalten eine Punktzahl, die sich zur Maximalpunkzahl gleich verhält wie deren Leistung zur Bestleistung. Auf diese Weise wird ein schnelles Ergebnis belohnt bzw. eine langsame Verarbeitungsdauer bestraft.
- Der zweite Typ betrifft die reine Erfüllung der maximalen Punkvergabe. Dies trifft beispielsweise bei der Plattformunterstützung zu. Eine eingeschränkte Erfüllung führt zu einer Vergabe von abgewerteten Punktzahlen.

7.2 ERGEBNISSE UND RANGORDNUNG

Hier werden die Ergebnisse für jede einzelne Kriteriengruppe aufgezeigt und ausgewertet.

7.2.1 Kriteriengruppe Funktionalität (Bewertungstyp 2)

Die Abbildung 7-1 gibt eine Übersicht über das Ergebnis im Bereich der Funktionalität. Den höchsten Wert hat PostgreSQL erreicht; damit führt PostgreSQL die Rangordnung im Bereich der Funktionalität an (siehe Anhang F). Der Abstand zur eXist ist jedoch nicht signifikant.

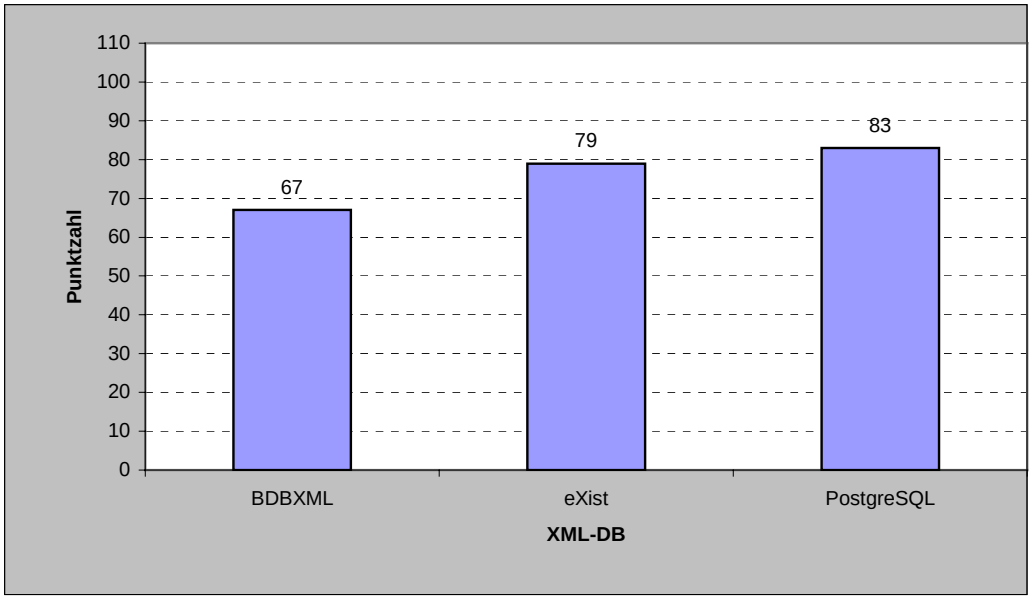


Abbildung 7- 1 Ergebnis in der Kriteriengruppe Funktionalität

Die Tabelle 7-1 stellt die Teilrangordnung dar, die sich aus den gezeigten Ergebnissen ableiten lässt.

Rang	Punktzahl	XML-Datenbank
1	83	PostgreSQL
2	79	eXist
3	67	Berkeley DB XML

Tabelle 7- 1 Rangordnung im Bereich Funktionalität

Bei der Betrachtung der aufgeschlüsselten Werte innerhalb der Kategoriengruppe Funktionalität, zeigt sich, dass die Stärken von PostgreSQL bei der Zugriffsvariante Sprache und Transaktionsverwaltung liegen (siehe Abbildung 7-2).

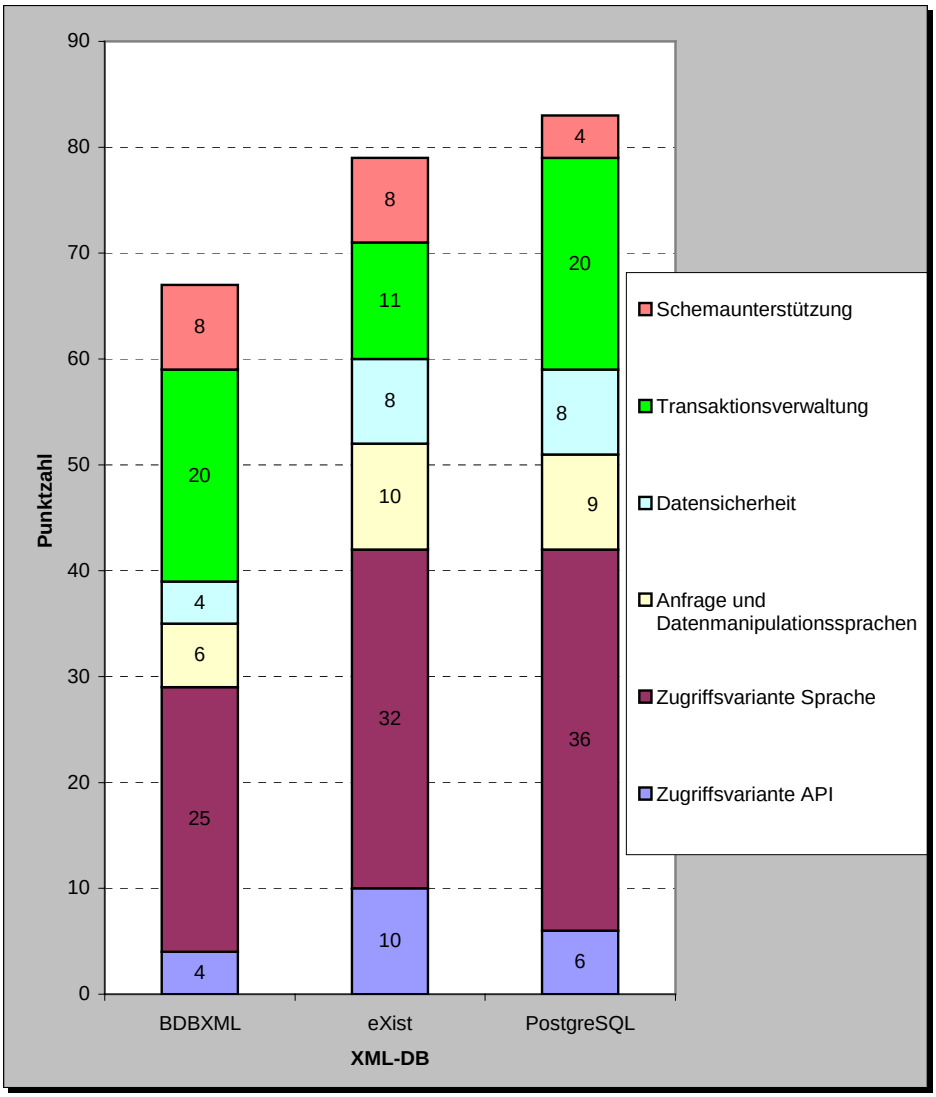


Abbildung 7- 2 Detailliertes Ergebnis aus dem Bereich Funktionalität (siehe Anhang F)

7.2.2 Kriteriengruppe Plattformen (Bewertungstyp 2)

Bei der Plattformunterstützung erreicht eXist die Maximalpunkte und erhält damit die Spitzenposition (siehe Anhang F). In Abbildung 7-3 wird das Ergebnis graphisch dargestellt.

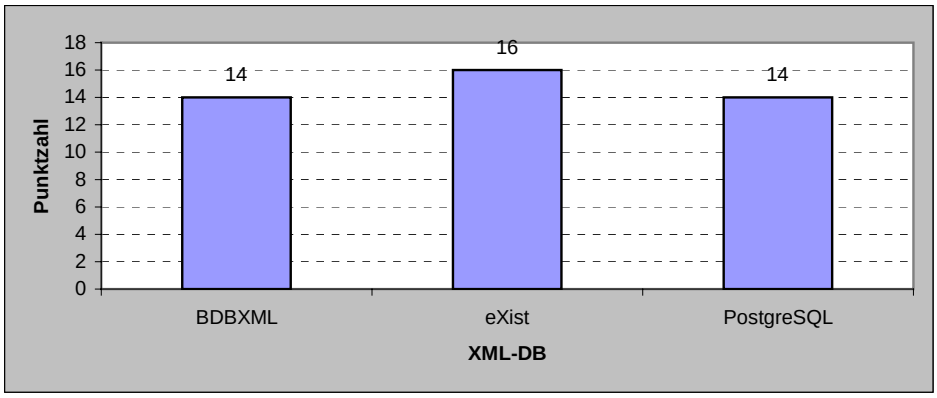


Abbildung 7- 3 Ergebnis in der Kategoriengruppe Plattformen

Rang	Punktzahl	XML-Datenbank
1	16	eXist
2	14	Berkeley DB XML
2	14	PostgreSQL

Tabelle 7- 2 Rangordnung im Bereich Plattformen

7.2.3 Kriteriengruppe Performanz (Bewertungstyp 1)

Die größte Kriteriengruppe in der Detailbetrachtung ist die Performanz (siehe Anhang F). Zusammengefasst zeigt sich das Bild aus Abbildung 7-4 bei der Performanzbewertung.

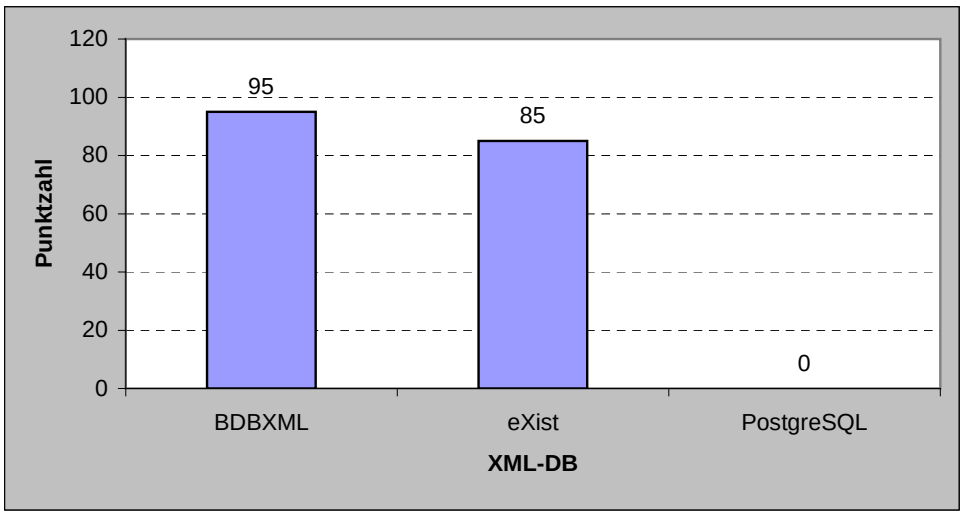


Abbildung 7- 4 Ergebnis in der Kategoriengruppe Performanz

Die aufgeschlüsselten Werte innerhalb der Kategoriengruppe Performanz zeigt das Bild in Abbildung 7-5.

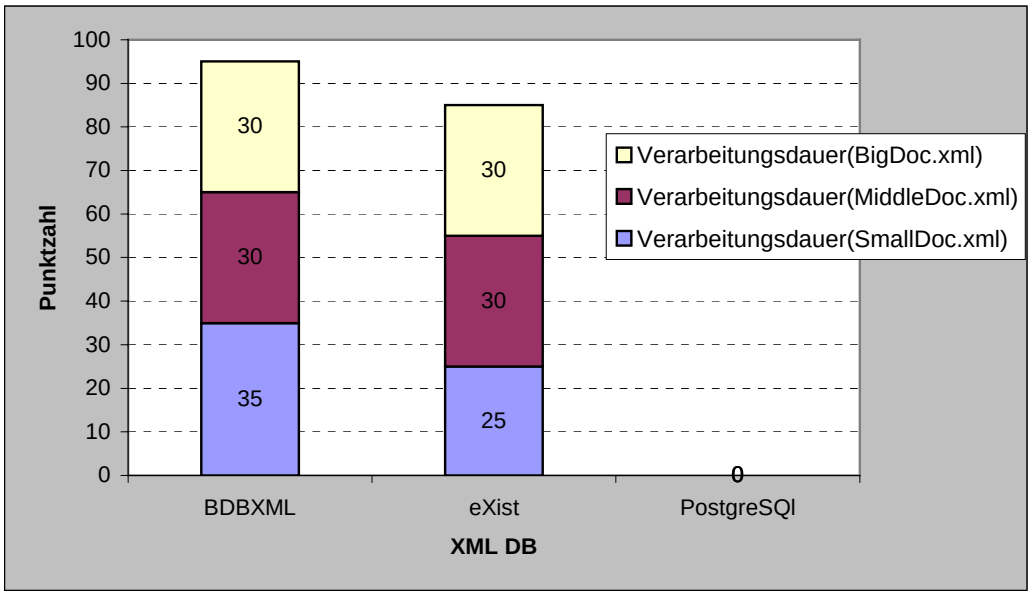


Abbildung 7- 5 Detailliertes Ergebnis aus dem Bereich Performanz

Rang	Punktzahl	XML-Datenbank
1	95	Berkeley DB XML
2	85	eXist
3	0	PostgreSQL

Tabelle 7- 3 Rangordnung im Bereich Performanz

7.2.4 Gesamtbetrachtung

Hier soll das Gesamtergebnis aufgezeigt werden, das über alle Kriteriengruppen hinweg in der Summe berechnet worden ist. Die Abbildung 7-6 stellt die Ergebnisse dar. Die aufgeschlüsselten Werte innerhalb der gesamten Betrachtung zeigt das Bild in Abbildung 7-7.

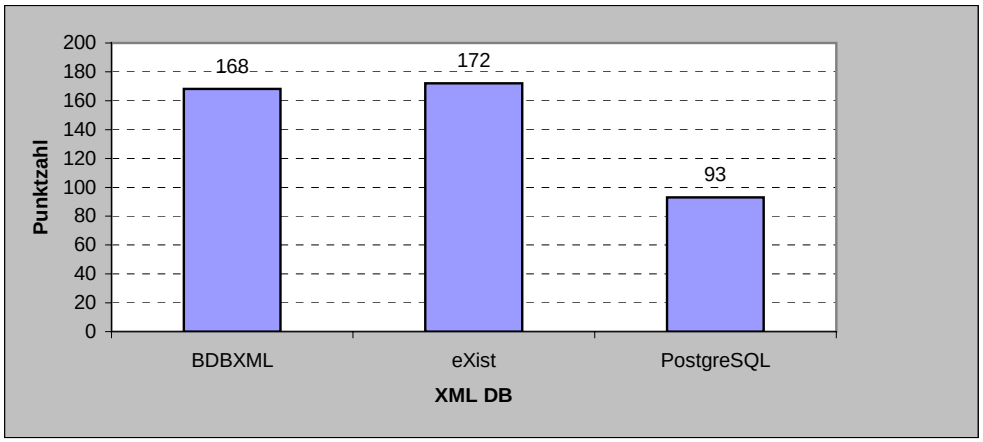


Abbildung 7- **6** Gesamtergebnis über alle Kategoriengruppen

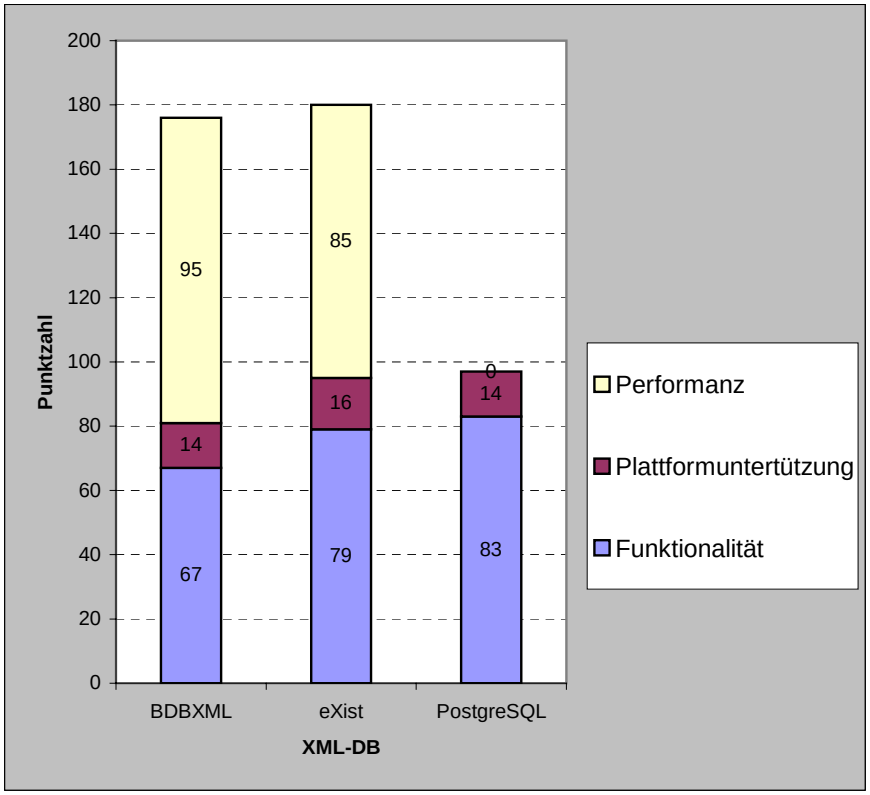


Abbildung 7- 7 Detailliertes Gesamtergebnis über alle Kategoriengruppen

Rang	Punktzahl	XML Datenbank
1	172	eXist
2	168	Berkeley DB XML
3	93	PostgreSQL

Tabelle 7- 4 Schlussrangordnung von XML-Datenbanken in der engeren Auswahl

7.3 INTERPRETATION DER ERGEBNISSE

Die Rangordnung ist je nach Kriteriengruppen unterschiedlich, was darauf hindeutet, das eine XML-Datenbank für bestimmte Anforderungen besser und für andere schlechter geeignet ist. In diesem Abschnitt wird auf jede Kriteriengruppe näher eingegangen.

Funktionalität

Wie Abbildung 7-1 deutlich macht, ist PostgreSQL im Bereich Funktionalität beispielsweise in der Zugriffvariante Sprachen führend. Die Erklärung liegt darin, dass PostgreSQL ein langjährig gepflegtes Produkt ist.
Berkeley DB XML ist eine eingebettete und native XML-Datenbank, die sich funktional über eine eigene Applikation steuern lässt. Hier ist es nicht die Transaktionsverwaltung, die gegenüber der Konkurrenz an Boden verliert, sondern es sind die vielen fehlenden Zugriffsvarianten. Ausgenommen von XQuery wird auch zu wenig für das Stellen von Anfragen oder Datenmanipulationen geboten.
In eXist trifft man eher Experimentierfreudigkeit an. Grundlegende Funktionalitäten, wie z.B. ‚Transaktionsverwaltung, werden bei eXist db.org auch in der Roadmap [eXiROA] deutlich unwichtiger eingestuft. Dies erklärt auch den zweiten Teilrang von eXist.

Plattformen

Wenn es um Plattformunterstützung geht, dann ist eXist als vollständig in Java implementierte XML-Datenbank nicht zu schlagen, da eXist sich auf alle Betriebsysteme, in welchen sich eine Java Umgebung befindet, einrichten lässt.
Berkeley DB XML hingegen ist nicht auf dem Server-Betriebsystem wie Microsoft Windows Server 2003 nutzbar. Sucht man eine XML-Datenbank für ein Mehrbenutzerszenario, in welchem der Einsatz eines Servers typisch ist, dann ist Berkeley DB XML nicht geeignet; PostgreSQL hingegen schon. PostgreSQL verliert jedoch Punkte wegen der fehlenden Unterstützung von Microsoft Windows Vista.

Performanz

Der Abschnitt 6.5 hat bereits eine Andeutung der Bewertung der Performanz gegeben. Das Ergebnis in einer Anfragengruppe fällt sehr schlecht aus, wenn eine oder mehrere Anfragen von einem System nicht unterstützt werden. Deswegen spielt für die Bewertung der Performanz auch die Unterstützung der Anfragen eine wichtige Rolle. In Abschnitt 6.4 wurde gezeigt, dass nicht die spezifischen Anfragen aus XMark die Probleme bereiten, sondern grundsätzliche Lücken in der Implementierung der Spezifikation des W3C. Berkeley DB XML kommt mit allen Dokumentgrößen hervorragend zurecht und erlaubt das Stellen aller Anfragen aus den Benchmark. Dies führt Berkeley DB XML zum Spitzenrang im Bereich der Performanz. PostgreSQL unterstützt hingegen keine Anfragegruppe aus XMark vollständig und wird mit allen Anfragen außer Anfrage 1 und 2 mit der Maximaldauer gewertet. Dass dies unweigerlich die letzte Rangordnung

erbringt, wird unmittelbar klar. EXist ist zwar nicht immer schnell bei der Anfrageverarbeitung, aber unterstützt genau so wie Berkeley DB XML alle Anfragen aus XMark.

7.4 EMPFEHLUNG

Das Endresultat, das in Abbildung 7- 5 und Tabelle 7- 4 ersichtlich ist, zeigt eXist als erste Wahl in der Rangordnung. Es handelt sich nicht um die einzige Empfehlung, die hier ausgesprochen wird. Die Abbildungen 7-2 und 7- 4 zeigen ein leichtes Defizit im Bereich Funktionalität und Performanz bei eXist. Wie die Abbildung 7-5 verdeutlicht, hat eXist dieses leichte Defizit ausgeglichen. Wenn man die Funktionalität und Performanz für wichtig hält, wäre die Empfehlung für Berkeley DB XML oder PostgreSQL gegeben.
Wenn eine Vielzahl an XML Dokumenten gespeichert werden soll und wenn es um Plattformunterstützung geht, halte ich eXist für eine gute Wahl. In eXist scheint das neue Nummerierungsschema vor allem die Performanz bei Datenmanipulation zu erhöhen, was aber hier nicht weiter geprüft wurde.
PostgreSQL kann auch eine gute Wahl darstellen, wenn sehr viele Funktionalitäten abgedeckt werden müssen. Wie in der Abbildung 7-2 zu sehen ist, ist PostgreSQL deutlich funktionsreicher als Berkeley DB XML und wird mit vorgefertigten Administrationswerkzeugen ausgeliefert. Allerdings bleibt bei PostgreSQL noch zu beobachten, ob die Unterstützung von W3C XQuery verbessert beziehungsweise aktualisiert wird.
Bei Berkeley DB XML handelt es sich um eine leichtgewichtige XML-Datenbank, d.h. sie ist nur mit den wichtigsten Funktionen ausgestattet. Wenn es um die Performanz geht, ist Berkeley DB XML das schnellste Datenbanksystem aus der engeren Auswahl. Ich halte Berkeley DB XML auch für eine gute Wahl, wenn folgende Bedingungen erfüllt sind:

- Für Berkeley DB XML ist kein komplexer Mehrbenutzerbetrieb vorgesehen.
- Der nötige Funktionsumfang wird durch Berkeley DB XML abgedeckt.
- Performanz bei der Anfrageverarbeitung ist von großer Bedeutung.
- Die Steuerung des Datenbanksystems erfolgt über eine eigene Applikation.

KAPITEL 8 ZUSAMMENFASSUNG UND AUSBLICK

8.1 ZUSAMMENFASSUNG

Als ich mit diesem Thema konfrontiert wurde, war schwer zu erkennen, welche XML-Datenbanken es auf dem Markt gibt, zu welcher Gattung diese gehören und was unter XML-Datenbank zu verstehen ist. Mit dieser Bachelorarbeit habe ich die Situation dahingehend erarbeitet, dass eine Einführung über XML vorangestellt wurde. Mit dem geschaffenen Verständnis wurden die XML-Datenbanken betrachtet und verglichen.

Der Anforderungskatalog hat drei Produkte aus dem großen Marktangebot identifiziert, welche umfassend nach bestimmten Kriterien miteinander verglichen worden sind.

Während der Arbeit wurde von mir ein Programm entwickelt, das den Queryprozessor der ausgewählten XML-Datenbanken hinsichtlich der Performanz testet.

Die abschließende Bewertung beleuchtet die Produktwahl zu unterschiedlichen Aspekten und beschränkt sich nicht nur auf den Testsieger eXist. In jeder Kriteriengruppe (Performanz, Funktionalität, Plattform) wird eine andere XML-Datenbank empfohlen. Funktional gesehen ist PostgreSQL in der Spitzenposition, während Berkeley DB XML bei den Messzeiten dominant ist. EXist ist bei der Messzeit und der Funktionalität akzeptabel und in heterogener Umgebung ideal. Deswegen ist sie am Ende der Testsieger.

Diese Bachelorabeit hat eine Lücke im Bereich der XML-Datenbanken geschlossen. Sie fasst in einem Werk die Einführung von XML bis hin zur Produktempfehlung, die auf intensiven Bewertungen der ausgewählten Datenbanksysteme basiert, zusammen.

8.2 AUSBLICK

Obwohl konkrete Empfehlungen zu den ausgewählten XML-Datenbanken gegeben wurden, ist anzunehmen, dass bei einer Erweiterung der Anforderungskriterien auch andere XML-Datenbanken empfohlen werden können.

In dieser Arbeit wurden ausschließlich XML-OpenSource Datenbanken untersucht und verglichen. Es könnte im Rahmen einer weiteren Arbeit eine Gegenüberstellung von kommerziellen XML-Produkten mit den in dieser Arbeit untersuchten XML-OpenSource Datenbanken erfolgen. Nachweisliche Vorteile könnten die Verwendung solcher kommerziellen Produkte begründen und den Anwendern eine größere Auswahl zur Verfügung stellen.

Während der Arbeit wurde festgestellt, dass man keine konkrete Aussage darüber machen kann, inwieweit existierende XML-OpenSource Datenbanken die Standards (XQuery) unterstützt. Deshalb könnte eine weitere Arbeit sich detailliert den Schritten der Anfrageverarbeitung in XML-Datenbanken widmen. Indizierungsweisen könnten verglichen und bewertet werden. Hierzu könnte Easy XML-Database Benchmark verwendet werden. Ein Vergleich könnte mit und ohne Indizes durchgeführt werden.

ANHANG

A XQUERY-ANFRAGEN AUS XMARK FÜR DIE BENCHMARK

Q1	*Return the name of the person with ID `person0'*
	let $auction := doc("small.xml") return for $b in $auction/site/people/person[@id = "person0"] return $b/name/text()
Q2	*Return the initial increases of all open auctions.*
	let $auction := doc("small.xml") return for $b in $auction/site/open_auctions/open_auction return <increase>{$b/bidder[1]/increase/text()}</increase>
Q3	*Return the IDs of all open auctions whose current increase is at least twice as high as the initial increase.*
	let $auction := doc("small.xml") return for $b in $auction/site/open_auctions/open_auction where zero-or-one($b/bidder[1]/increase/text()) * 2 <= $b/bidder[last()]/increase/text() return <increase first="{$b/bidder[1]/increase/text()}" last="{$b/bidder[last()]/increase/text()}"/>
Q4	*List the reserves of those open auctions where a certain person issued a bid before another person.*
	let $auction := doc("small.xml") return for $b in $auction/site/open_auctions/open_auction where some $pr1 in $b/bidder/personref[@person = "person20"], $pr2 in $b/bidder/personref[@person = "person51"] satisfies $pr1 << $pr2 return <history>{$b/reserve/text()}</history>
Q5	*How many sold items cost more than 40?*
	let $auction := doc("small.xml") return count(for $i in $auction/site/closed_auctions/closed_auction where $i/price/text() >= 40 return $i/price)
Q6	*How many items are listed on all continents?*
	let $auction := doc("small.xml") return for $b in $auction//site/regions return count($b//item)

Q7	*How many pieces of prose are in our database?*
	let $auction := doc("small.xml") return for $p in $auction/site return count($p//description) + count($p//annotation) + count($p//emailaddress)
Q8	*List the names of persons and the number of items they bought.*
	let $auction := doc("small.xml") return for $p in $auction/site/people/person let $a := for $t in $auction/site/closed_auctions/closed_auction where $t/buyer/@person = $p/@id return $t return <item person="{$p/name/text()}">{count($a)}</item>
Q9	*List the names of persons and the names of the items they bought in Europe.*
	let $auction := doc("small.xml") return let $ca := $auction/site/closed_auctions/closed_auction return let $ei := $auction/site/regions/europe/item for $p in $auction/site/people/person let $a := for $t in $ca where $p/@id = $t/buyer/@person return let $n := for $t2 in $ei where $t/itemref/@item = $t2/@id return $t2 return <item>{$n/name/text()}</item> return <person name="{$p/name/text()}">{$a}</person>

Q10	*List all persons according to their interest; use French markup in the result.*
	let $auction := doc("small.xml") return for $i in distinct- values($auction/site/people/person/profile/interest/@category) let $p := for $t in $auction/site/people/person where $t/profile/interest/@category = $i return <personne> <statistiques> <sexe>{$t/profile/gender/text()}</sexe> <age>{$t/profile/age/text()}</age> <education>{$t/profile/education/text()}</education> <revenu>{fn:data($t/profile/@income)}</revenu> </statistiques> <coordonnees> <nom>{$t/name/text()}</nom> <rue>{$t/address/street/text()}</rue> <ville>{$t/address/city/text()}</ville> <pays>{$t/address/country/text()}</pays> <reseau> <courrier>{$t/emailaddress/text()}</courrier> <pagePerso>{$t/homepage/text()}</pagePerso> </reseau> </coordonnees> <cartePaiement>{$t/creditcard/text()}</cartePaiement> </personne> return <categorie>{<id>{$i}</id>, $p}</categorie>
Q11	*For each person, list the number of items currently on sale whose price does not exceed 0.02% of the person's income.*
	let $auction := doc("small.xml") return for $p in $auction/site/people/person let $l := for $i in $auction/site/open_auctions/open_auction/initial where $p/profile/@income > 5000 * exactly-one($i/text()) return $i return <items name="{$p/name/text()}">{count($l)}</items>

Q12	*For each richer-than-average person, list the number of items currently on sale whose price does not exceed 0.02% of the person's income.*
	let $auction := doc("small.xml ") return for $p in $auction/site/people/person let $l := for $i in $auction/site/open_auctions/open_auction/initial where $p/profile/@income > 5000 * exactly-one($i/text()) return $i where $p/profile/@income > 50000 return <items person="{$p/profile/@income}">{count($l)}</items>
Q13	*List the names of items registered in Australia along with their descriptions.*
	let $auction := doc("small.xml ") return for $i in $auction/site/regions/australia/item return <item name="{$i/name/text()}">{$i/description}</item>
Q14	*Return the names of all items whose description contains the word „gold“.*
	let $auction := doc("small.xml ") return for $i in $auction/site//item where contains(string(exactly-one($i/description)), "gold") return $i/name/text()
Q15	*Print the keywords in emphasis in annotations of closed auctions.*
	let $auction := doc("small.xml ") return for $a in $auction/site/closed_auctions/closed_auction/annotation/description/parlist/ listitem/ parlist/ listitem/ text/ emph/ keyword/ text() return <text>{$a}</text>

Q16	*Return the IDs of those auctions that have one or more keywords in emphasis.*
	let $auction := doc("small.xml ") return for $a in $auction/site/closed_auctions/closed_auction where not(empty($a/annotation/description/parlist/listitem/parlist/listitem/text/emph/ keyword/ text())) return <person id="{$a/seller/@person}"/>
Q17	*Which persons don't have a homepage?*
	let $auction := doc("small.xml ") return for $p in $auction/site/people/person where empty($p/homepage/text()) return <person name="{$p/name/text()}"/>
Q18	*Convert the currency of the reserve of all open auctions to another currency.*

	declare namespace local = "http://www.foobar.org";
	declare function local:convert($v as xs:decimal?) as xs:decimal?
	{
	2.20371 * $v (: convert Dfl to Euro :)
	};
	let $auction := doc("small.xml ") return
	for $i in $auction/site/open_auctions/open_auction
	return local:convert(zero-or-one($i/reserve))

Q19	*Give an alphabetically ordered list of all items along with their location.*
	let $auction := doc("small.xml ") return for $b in $auction/site/regions//item let $k := $b/name/text() order by zero-or-one($b/location) ascending empty greatest return <item name="{$k}">{$b/location/text()}</item>

Q20	*Group customers by their income and output the cardinality of each group.*
	let $auction := doc("small.xml ") return
	<result>
	<preferred>
	{count($auction/site/people/person/profile[@income >= 100000])}
	</preferred>
	<standard>
	{
	count(
	$auction/site/people/person/
	profile[@income < 100000 and @income >= 30000]
	)
	}
	</standard>
	<challenge>
	{count($auction/site/people/person/profile[@income < 30000])}
	</challenge>
	<na>
	{
	count(
	for $p in $auction/site/people/person
	where empty($p/profile/@income)
	return $p
	)
	}
	</na>
	</result>

B EINRICHTEN DER BETRACHTETEN DATENBANK-PRODUKTE

B.1 eXist

EXIST wird über einen Installer in Form einer ausführbaren Datei mit der Endung *.jar* (eXist-1.2.0-rev7233.jar) installiert und erfordert daher eine bereits installierte Java-Plattform (zumindest eine Java- Laufzeitumgebung) ab Version 1.4.2.
Bei der Installation gibt es die Möglichkeit, Installationskomponenten auszuwählen. Zwingend ist die Kern (engl. Core)-Installation; optional kann die Dokumentation und der Quellecode ausgewählt werden. Nach der Angabe des Installationspfades läuft die Installation ohne Unterbrechung ab.
Es besteht die Möglichkeit, den Server-Prozess als Dienst (engl. services) einzurichten, sodass bei jedem Bootvorgang von Windows XP der Serverprozess automatisch startet. Es gibt auch Dateien-Skripte, wie „startup.bat" und „shutdown.bat", um den Server Prozess zu starten bzw. zu stoppen. Als Alternative gibt es zum manuellen Starten einen Eintrag im Startmenu.
Beim Beenden wird zuerst die Datenbank „ausgeschaltet" und danach der Server-Prozess beendet.
Für die Administration stehen ein Webinterface und umfangreiche Client-Applikationen zur Verfügung. Die Benutzerverwaltung ist über die beiden Benutzerschnittstellen verfügbar. Mit Hilfe der Client-Applikation „eXist Client Shell" lässt sich die Verbindung zu der Datenbank erstellen und darin eine neue Kollektion für XML-Dokumente erstellen. Erst wenn man sich als Admin eingeloggt hat, lässt sich auch durch das Webinterface eine neue Kollektion erstellen.
Eine Datenbankinstanz namens „exist" mit empfohlenen Grundeinstellungen wird bei der Installation automatisch erstellt. Außerdem wird der Server-Prozess in der Servlet-Konfiguration angefertigt und hat die folgende URL:
xmldb[11]*:exist*[12]*://localhost:8080*[13]*/exist/xmlrpc* [14].

Um eine Kollektion namens „*BaHome*" zu erstellen, kann man den eXist Admin Client verwenden oder über die Shell das Kommando mkcol „BaHome" absetzen. Der eXist Admin Client gibt auch eine Übersicht über alle Kollektionen und gespeicherte XML-Dokumente, welche auch wieder über die Benutzeroberfläche gelöscht werden können.

[11] Es handelt sich um eine XML-Datenbank-Adresse

[12] Der Name der Datenbankverbindung ist exist

[13] Adresse (IP oder Domainname) und Port des Servers

[14] Der Pfad der XMLRPC-Schnittstelle

B.2 Berkeley DB XML

Berkeley DB XML ist als Quellcode oder als Installer erhältlich. Ich verwende den Installer. Schon bei der Installation integriert ist die Basis Berkeley DB in der Version 4.5.20. Vor der Installation kann man die Installationskomponenten wie die Java API und die Produktdokumentation (ab-)wählen und die Umgebungsvariablen Classpath und Path optional automatisch für das aktive Benutzerkonto oder systemweit setzen lassen. Der erste Einrichtungsschritt nach der Installation ist die Erstellung eines eigenen Containers, z.B. über die Berkeley DB XML Shell. Hierzu dient das folgende Kommando:

```
createContainer BaHomeContainer.dbxml
```

Mit folgendem Befehl werden die dreie XML-Dokumente in den Container importiert.

```
dbxml> putDocument SmallDoc.xml
'D:\Bachelorarbeit.CD\BaData\XML\SmallDoc.xml' f
```

Die Verwaltung der Datenbankumgebung kann man dem XMLManager überlassen.

B.3 PostgreSQL

Die PostgreSQL Windows-Version wird über einen Installer in Form einer ausführbaren Datei mit der Endung *.msi* (Postgresql.8.3beta3.msi) installiert. Während der Installation gibt es die Möglichkeit, Installationskomponenten auszuwählen. Zwingend ist die Kern (engl. Core)-Installation; Optional kann die Dokumentation und der Quellcode ausgewählt werden. Um mit XML-Dokumenten arbeiten zu können, ist es erforderlich, die Komponente XML2 mitzuinstallieren.
Damit der Datentyp XML für die XML-Daten korrekt unterstützt wird, muss die Datenbank den Zeichensatz UTF8 verwenden. Mit dem folgenden SQL Befehl lässt sich hinsichtlich der Performanzmessung ein Tabellenbereich erstellen.

```
CREATE TABLESPACE bahome_system
  OWNER bahome
  LOCATION 'c:/badata/ba/system';
```

Die Erstellung einer Tabelle für die Speicherung von XML-Dokumenten erfolgt über das Tool „pgAdmin SQL“ mit diesem Befehl:

```
CREATE TABLE xmldokument
(
  docname character varying NOT NULL,
  xmldata xml,
  CONSTRAINT xmldokument_pkey PRIMARY KEY (docname)
)
```

Die drei XML-Dokumente werden in der Tabelle wie folgt hinzugefügt:

```
insert into xmldokument values('SmallDoc'  ,'<?xml version="1.0"
standalone="yes"?><site>.....');
```

C DOKUMENT AUS XMARK

Dokumenttyp-Definition aus XMark

```
<!ELEMENT site (regions, categories, catgraph, people, open_auctions,
closed_auctions)>
<!ELEMENT categories (category+)>
<!ELEMENT category (name, description)>
<!ATTLIST category id ID #REQUIRED>
<!ELEMENT name (#PCDATA)>
<!ELEMENT description (text | parlist)>
<!ELEMENT text (#PCDATA | bold | keyword | emph)*>
<!ELEMENT bold (#PCDATA | bold | keyword | emph)*>
<!ELEMENT keyword (#PCDATA | bold | keyword | emph)*>
<!ELEMENT emph (#PCDATA | bold | keyword | emph)*>
<!ELEMENT parlist (listitem)*>
<!ELEMENT listitem (text | parlist)*>
<!ELEMENT catgraph (edge*)>
<!ELEMENT edge EMPTY>
<!ATTLIST edge from IDREF #REQUIRED to IDREF #REQUIRED>
<!ELEMENT regions (africa, asia, australia, europe, namerica, samerica)>
<!ELEMENT africa (item*)>
<!ELEMENT asia (item*)>
<!ELEMENT australia (item*)>
<!ELEMENT namerica (item*)>
<!ELEMENT samerica (item*)>
<!ELEMENT europe (item*)>
<!ELEMENT item (location, quantity, name, payment, description, shipping,
incategory+, mailbox)>
<!ATTLIST item id ID #REQUIRED featured CDATA #IMPLIED>
<!ELEMENT location (#PCDATA)>
<!ELEMENT quantity (#PCDATA)>
<!ELEMENT payment (#PCDATA)>
<!ELEMENT shipping (#PCDATA)>
<!ELEMENT reserve (#PCDATA)>
<!ELEMENT incategory EMPTY>
<!ATTLIST incategory category IDREF #REQUIRED>
<!ELEMENT mailbox (mail*)>
<!ELEMENT mail (from, to, date, text)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT date (#PCDATA)>
<!ELEMENT itemref EMPTY>
```

<!ATTLIST itemref item IDREF #REQUIRED>
<!ELEMENT personref EMPTY>
<!ATTLIST personref person IDREF #REQUIRED>
<!ELEMENT people (person*)>
<!ELEMENT person (name, emailaddress, phone?, address?, homepage?, creditcard?, profile?, watches?)>
<!ATTLIST person id ID #REQUIRED>
<!ELEMENT emailaddress (#PCDATA)>
<!ELEMENT phone (#PCDATA)>
<!ELEMENT address (street, city, country, province?, zipcode)>
<!ELEMENT street (#PCDATA)>
<!ELEMENT city (#PCDATA)>
<!ELEMENT province (#PCDATA)>
<!ELEMENT zipcode (#PCDATA)>
<!ELEMENT country (#PCDATA)>
<!ELEMENT homepage (#PCDATA)>
<!ELEMENT creditcard (#PCDATA)>
<!ELEMENT profile (interest*, education?, gender?, business, age?)>
<!ATTLIST profile income CDATA #IMPLIED>
<!ELEMENT interest EMPTY>
<!ATTLIST interest category IDREF #REQUIRED>
<!ELEMENT education (#PCDATA)>
<!ELEMENT income (#PCDATA)>
<!ELEMENT gender (#PCDATA)>
<!ELEMENT business (#PCDATA)>
<!ELEMENT age (#PCDATA)>
<!ELEMENT watches (watch*)>
<!ELEMENT watch EMPTY>
<!ATTLIST watch open_auction IDREF #REQUIRED>
<!ELEMENT open_auctions (open_auction*)>
<!ELEMENT open_auction (initial, reserve?, bidder*, current, privacy?, itemref, seller, annotation, quantity, type, interval)>
<!ATTLIST open_auction id ID #REQUIRED>
<!ELEMENT privacy (#PCDATA)>
<!ELEMENT initial (#PCDATA)>
<!ELEMENT bidder (date, time, personref, increase)>
<!ELEMENT seller EMPTY>
<!ATTLIST seller person IDREF #REQUIRED>
<!ELEMENT current (#PCDATA)>
<!ELEMENT increase (#PCDATA)>
<!ELEMENT type (#PCDATA)>
<!ELEMENT interval (start, end)>
<!ELEMENT start (#PCDATA)>
<!ELEMENT end (#PCDATA)>
<!ELEMENT time (#PCDATA)>

<!ELEMENT status (#PCDATA)>
<!ELEMENT amount (#PCDATA)>
<!ELEMENT closed_auctions (closed_auction*)>
<!ELEMENT closed_auction (seller, buyer, itemref, price, date, quantity, type, annotation?)>
<!ELEMENT buyer EMPTY>
<!ATTLIST buyer person IDREF #REQUIRED>
<!ELEMENT price (#PCDATA)>
<!ELEMENT annotation (author, description?, happiness)>
<!ELEMENT author EMPTY>
<!ATTLIST author person IDREF #REQUIRED>
<!ELEMENT happiness (#PCDATA)>

D MESSERGEBNISSE AUS PERFORMANZTESTS

Die Messungen wurden mit dem Programm Easy XML-Database Benchmark durchgeführt. Sämtliche Zahlen haben die Einheit Millisekunden (ms). In die Berechnungen der arithmetischen Mittelwerte werden die schnellsten und langsamsten Messwerte nicht einbezogen.

D.1 Messergebnisse von eXist

mit Smalldoc.xml

Messergebnisse von eXist mit SmallDoc.xml									
	1.Messung	2.Messung	3.Messung	4.Mesung	5.Messung	6.Messung	7.Messung	8.Messung	Arth.Wert
Q01	~~512~~	32	31	31	31	14	28	26	29,4
Q02	~~297~~	16	16	15	16	31	16	17	18,66667
Q03	~~172~~	31	31	15	15	31	31	40	27,8
Q04	~~240~~	94	94	78	93	78	94	76	87,4
Q05	~~100~~	15	15	16	0	15	15	15	15
Q06	~~16~~	15	0	0	0	16	0	16	6,2
Q07	~~172~~	15	15	16	0	15	0	14	11,8
Q08	~~110~~	49	46	32	47	31	31	31	37,4
Q09	~~90~~	16	16	15	12	15	14	0	14,4
Q10	~~453~~	63	47	31	31	47	31	31	37,4
Q11	~~203~~	78	63	63	63	63	63	20	63
Q12	~~70~~	63	63	62	47	63	46	46	56,2
Q13	~~37~~	16	32	16	0	16	16	16	16
Q14	~~30~~	16	16	15	0	16	16	15	15,6
Q15	~~421~~	47	46	47	31	32	31	16	39
Q16	~~78~~	63	78	15	15	16	16	31	36,5
Q17	~~31~~	16	16	16	15	16	16	15	15,8
Q18	~~47~~	32	15	15	0	15	16	16	15,4
Q19	~~47~~	15	32	16	31	15	16	15	18,6
Q20	~~32~~	31	15	16	16	32	16	15	18,8
				Arithmetisches Mittel alle Anfragen					29,01833
				Arithmetisches Mittel von Gruppe I					20
				Arithmetisches Mittel von Gruppe II					44,81111
				Arithmetisches Mittel von Gruppe III					23,04286
				Arithmetisches Mittel von Gruppe IV					17,6

mit Middledoc.xml

	Messergebnisse von eXist mit MiddleDoc.xml								
	1.Messung	2.Messung	3.Messung	4.Messung	5.Messung	6.Messung	7.Messung	8.Messung	Arth. Wert
Q01	~~172~~	47	47	47	32	47	31	31	40,8
Q02	~~641~~	187	172	156	141	141	140	141	150,2
Q03	~~672~~	406	281	281	406	266	265	265	299,8
Q04	~~12469~~	12406	12328	12172	12156	12047	12047	11985	12150
Q05	~~328~~	187	63	62	63	62	62	62	62,4
Q06	~~50~~	16	16	15	15	16	15	15	15,4
Q07	~~250~~	16	16	31	16	16	31	16	19
Q08	~~74047~~	66969	66766	66328	61659	61359	60640	60765	63375,4
Q09	~~209000~~	157937	153859	151687	157156	133375	131062	133375	145890,4
Q10	~~6672~~	6359	5969	5860	5859	5400	5828	5828	5868,8
Q11	~~188094~~	186060	187093	183000	180090	181015	180391	181019	182297
Q12	~~281562~~	251484	237469	236172	233703	233687	232305	236406	235487,4
Q13	~~421~~	78	78	78	78	78	62	78	78
Q14	~~1281~~	578	375	344	234	297	234	234	296,8
Q15	~~765~~	63	47	31	31	47	31	47	40,6
Q16	~~375~~	156	109	281	94	110	94	109	140,6
Q17	~~359~~	203	188	187	203	188	188	188	191
Q18	~~781~~	78	78	63	63	63	63	78	69
Q19	~~1484~~	532	469	406	1797	406	467	406	456
Q20	~~562~~	203	172	172	172	172	172	171	172
					Arithmetisches Mittel alle Anfragen				32355,03
					Arithmetisches Mittel von Gruppe I				133,33333
					Arithmetisches Mittel von Gruppe II				71762,567
					Arithmetisches Mittel von Gruppe III				26928,525
					Arithmetisches Mittel von Gruppe IV				232,33333

mit Bigdoc.xml

	Messergebnisse von eXist mit BigDoc.xml								
	1.Messung	2.Messung	3.Messung	4.Messung	5.Messung	6.Messung	7.Messung	8.Messung	Arth.Wert
Q01	~~781~~	172	93	63	62	62	63	64	69
Q02	~~1172~~	328	250	250	234	250	234	255	247,8
Q03	~~1469~~	422	453	469	500	375	422	406	434,4
Q04	~~19984~~	18609	18734	18453	18437	18437	18328	18437	18474,6
Q05	~~1078~~	547	547	563	563	547	562	563	556,4
Q06	~~188~~	31	47	32	47	31	16	36	35,4
Q07	~~93~~	46	62	63	63	46	47	48	53,2
Q08	~~172562~~	162544	163579	161844	163312	162141	161816	162211	162410,4
Q09	~~359484~~	329860	311985	317172	315297	312719	311172	310000	313669
Q10	~~14015~~	13062	12766	12641	12625	12594	12562	12566	12638,4
Q11	~~459203~~	409531	403235	403422	408454	400468	400005	407172	404550,2
Q12	~~281562~~	237669	237469	236172	233703	233687	233106	233650	234936,2
Q13	~~781~~	78	62	78	78	63	62	62	68,6
Q14	~~1329~~	500	531	313	312	406	312	312	368,6
Q15	~~422~~	125	94	94	110	94	93	95	97,4
Q16	~~1297~~	156	156	360	188	156	156	170	165,2
Q17	~~547~~	282	262	266	344	328	265	279	284
Q18	~~141~~	125	93	125	109	94	94	94	103,2
Q19	~~1093~~	578	1297	750	562	672	562	1750	771,8
Q20	~~953~~	563	484	375	375	547	375	390	453,1667
					Arithmetisches Mittel alle Anfragen				57519,35
					Arithmetisches Mittel von Gruppe I				331,3333
					Arithmetisches Mittel von Gruppe II				109821,2
					Arithmetisches Mittel von Gruppe III				61142,2
					Arithmetisches Mittel von Gruppe IV				442,7222

D.2 Messergebnisse von BDBXML

mit Smalldoc.xml

	Messergebnisse von DBMXL mit SmallDoc.xml								
	1.Messung	2.Messung	3.Messung	4.Mesung	5.Messung	6.Messung	7.Messung	8.Messung	Art.Wert
Q01	~~16~~	0	5	0	0	4	16	0	1,8
Q02	~~16~~	16	16	15	16	16	16	16	16
Q03	~~16~~	15	16	31	32	15	16	15	18,6
Q04	~~32~~	15	32	31	15	15	32	15	21,6
Q05	~~16~~	16	16	0	16	0	15	16	12,6
Q06	~~218~~	47	31	31	31	47	31	47	37,4
Q07	~~47~~	47	47	47	31	47	31	31	40,6
Q08	~~47~~	31	31	47	31	47	31	31	34,2
Q09	~~47~~	47	46	47	47	47	47	47	47
Q10	~~63~~	62	63	62	62	63	63	62	62,4
Q11	~~78~~	79	62	63	63	62	62	62	62,4
Q12	~~31~~	31	31	31	31	31	32	31	31
Q13	~~16~~	0	0	0	0	16	0	16	3,2
Q14	~~47~~	47	31	47	31	47	31	47	40,6
Q15	~~16~~	16	15	16	15	15	0	116	15,4
Q16	~~16~~	0	16	16	0	16	16	16	12,8
Q17	~~16~~	16	15	15	0	15	0	16	12,2
Q18	~~16~~	0	0	16	16	15	16	0	9,4
Q19	~~32~~	31	31	16	15	15	31	16	21,8
Q20	~~62~~	31	32	31	15	32	32	31	31,6
					Arithmetisches Mittel alle Anfragen				26,63
					Arithmetisches Mittel von Gruppe I				18,33333
					Arithmetisches Mittel von Gruppe II				26,96667
					Arithmetisches Mittel von Gruppe III				31,625
					Arithmetisches Mittel von Gruppe IV				20,93333

Mit Middledoc.xml

	Messergebnisse von DBMXL mit MiddleDoc.xml								
	1.Messung	2.Messung	3.Messung	4.Mesung	5.Messung	6.Messung	7.Messung	8.Messung	Arth.Wert
Q01	~~610~~	125	94	78	93	78	94	94	90,6
Q02	~~1687~~	344	344	344	350	375	360	350	349,6
Q03	~~938~~	875	859	859	860	844	859	828	856,2
Q04	~~1500~~	1453	1453	1484	1453	1359	1422	1422	1447,8333
Q05	~~547~~	188	125	141	140	125	140	141	137,4
Q06	~~16360~~	13375	12703	13438	13063	13437	13375	13328	13315,6
Q07	~~14078~~	14015	14031	14031	13922	13938	13765	13922	13965,6
Q08	~~44938~~	44938	44688	44546	44532	44375	44484	44343	44525
Q09	~~59297~~	59062	58328	58063	57578	57938	57688	57640	57931,4
Q10	~~9938~~	9813	9812	9781	9750	9594	9468	9750	9737,4
Q11	~~99906~~	99766	99813	99765	99593	99562	99562	99531	99649,6
Q12	~~33156~~	32969	32953	32875	33110	32609	32625	32625	32737,4
Q13	~~156~~	110	109	109	110	110	110	93	109,6
Q14	~~15766~~	14000	13625	11985	12015	12000	11828	11985	12322
Q15	~~235~~	250	187	188	188	188	187	187	187,6
Q16	~~422~~	219	219	218	218	218	219	219	218,6
Q17	~~328~~	281	282	281	281	297	281	281	281,2
Q18	~~546~~	282	281	281	281	281	281	281	281
Q19	~~844~~	796	781	797	782	781	782	781	784,4
Q20	~~422~~	406	391	390	375	391	390	390	390,4
					Arithmetisches mittel alle Anfragen				14465,922
					Arithmetisches Mittel von Gruppe I				4183,3333
					Arithmetisches Mittel von Gruppe II				22553,639
					Arithmetisches Mittel von Gruppe III				17498,85
					Arithmetisches Mittel von Gruppe IV				485,26667

Mit Bigdoc.xml

	Messergebnisse von DBMXL BigDoc.xml								
	1.Messung	2.Messung	3.Messung	4.Mesung	5.Messung	6.Messung	7.Messung	8.Messung	Arth.Wert
Q01	~~594~~	157	125	109	125	125	110	109	118,8
Q02	~~1453~~	516	515	500	500	469	484	456	493,6
Q03	~~1360~~	1312	1328	1312	1312	1281	1187	1297	1302,8
Q04	~~2516~~	2500	2500	2453	2453	2453	2438	2422	2459,4
Q05	~~875~~	250	235	235	204	219	219	204	219,4
Q06	~~29547~~	29109	28812	27953	27734	26750	28141	27735	28075
Q07	~~31157~~	29969	29719	29531	29500	29353	28984	29313	29483,2
Q08	~~100203~~	100109	99688	99719	99500	99453	99735	99687	99665,8
Q09	~~136016~~	134875	135094	135875	134875	134469	134069	133906	134676,4
Q10	~~16750~~	16281	16766	15530	14853	11938	14750	14650	15212,8
Q11	~~229125~~	228578	228422	228156	228219	228094	228063	228094	228266
Q12	~~74172~~	73235	73141	73172	73141	73000	73062	73000	73103,2
Q13	~~187~~	156	156	140	141	140	140	140	143,4
Q14	~~27531~~	27266	26985	26641	26985	24558	24734	24578	25984,6
Q15	~~312~~	312	281	297	297	297	297	312	300
Q16	~~344~~	344	344	328	344	328	344	328	337,6
Q17	~~437~~	437	422	422	438	437	422	422	428
Q18	~~437~~	422	422	422	422	422	422	421	422
Q19	~~3672~~	3656	3515	3297	2985	3281	3047	3172	3262,4
Q20	~~547~~	547	531	532	531	531	516	516	528,2
					Arithmetisches mittel alle Anfragen				32224,13
					Arithmetisches Mittel von Gruppe I				8774,267
					Arithmetisches Mittel von Gruppe II				51008,83
					Arithmetisches Mittel von Gruppe III				38486,78
					Arithmetisches Mittel von Gruppe IV				1404,2

D.3 Messegerbnisse von PostgreSQL

Mit SmallDoc.xml

	Messergebnisse von Postgres mit SmallDoc.xml								
	1.Messung	2.Messung	3.Messung	4.Messung	5.Messung	6.Messung	7.Messung	8.Messung	Arth.Wert
Q01	~~875~~	~~32~~	16	15	15	16	15	~~15~~	15,4
Q02	~~547~~	~~0~~	15	16	0	0	15	16	9,2
Q03	~~3600000~~	~~3600000~~	3600000	3600000	3600000	3600000	3600000	3600000	3600000
Q04	~~3600000~~	~~3600000~~	3600000	3600000	3600000	3600000	3600000	3600000	3600000
Q05	~~3600000~~	~~3600000~~	3600000	3600000	3600000	3600000	3600000	3600000	3600000
Q06	~~3600000~~	~~3600000~~	3600000	3600000	3600000	3600000	3600000	3600000	3600000
Q07	~~3600000~~	~~3600000~~	3600000	3600000	3600000	3600000	3600000	3600000	3600000
Q08	~~3600000~~	~~3600000~~	3600000	3600000	3600000	3600000	3600000	3600000	3600000
Q09	~~3600000~~	~~3600000~~	3600000	3600000	3600000	3600000	3600000	3600000	3600000
Q10	~~3600000~~	~~3600000~~	3600000	3600000	3600000	3600000	3600000	3600000	3600000
Q11	~~3600000~~	~~3600000~~	3600000	3600000	3600000	3600000	3600000	3600000	3600000
Q12	~~3600000~~	~~3600000~~	3600000	3600000	3600000	3600000	3600000	3600000	3600000
Q13	~~3600000~~	~~3600000~~	3600000	3600000	3600000	3600000	3600000	3600000	3600000
Q14	~~3600000~~	~~3600000~~	3600000	3600000	3600000	3600000	3600000	3600000	3600000
Q15	~~3600000~~	~~3600000~~	3600000	3600000	3600000	3600000	3600000	3600000	3600000
Q16	~~3600000~~	~~3600000~~	3600000	3600000	3600000	3600000	3600000	3600000	3600000
Q17	~~3600000~~	~~3600000~~	3600000	3600000	3600000	3600000	3600000	3600000	3600000
Q18	~~3600000~~	~~3600000~~	3600000	3600000	3600000	3600000	3600000	3600000	3600000
Q19	~~3600000~~	~~3600000~~	3600000	3600000	3600000	3600000	3600000	3600000	3600000
Q20	~~3600000~~	~~3600000~~	3600000	3600000	3600000	3600000	3600000	3600000	3600000
					Arithmetisches Mittel alle Anfragen				3240001,23
					Arithmetisches Mittel von Gruppe I				2400005,13
					Arithmetisches Mittel von Gruppe II				3000001,53
					Arithmetisches Mittel von Gruppe III				3600000
					Arithmetisches Mittel von Gruppe IV				3600000

Mit MiddelDoc.xml

Messergebnisse von Postgres mit MiddleDoc.xml									
	1.Messung	2.Messung	3.Messung	4.Messung	5.Messung	6.Messung	7.Messung	8.Messung	Arth.Wert
Q01	~~1813~~	1297	1266	1266	1265	1234	1234	~~1266~~	1259,4
Q02	~~688~~	687	594	657	640	656	626	626	641
Q03	~~3600000~~	3600000	3600000	3600000	3600000	3600000	3600000	3600000	3600000
Q04	~~3600000~~	3600000	3600000	3600000	3600000	3600000	3600000	3600000	3600000
Q05	~~3600000~~	3600000	3600000	3600000	3600000	3600000	3600000	3600000	3600000
Q06	~~3600000~~	3600000	3600000	3600000	3600000	3600000	3600000	3600000	3600000
Q07	~~3600000~~	3600000	3600000	3600000	3600000	3600000	3600000	3600000	3600000
Q08	~~3600000~~	3600000	3600000	3600000	3600000	3600000	3600000	3600000	3600000
Q09	~~3600000~~	3600000	3600000	3600000	3600000	3600000	3600000	3600000	3600000
Q10	~~3600000~~	3600000	3600000	3600000	3600000	3600000	3600000	3600000	3600000
Q11	~~3600000~~	3600000	3600000	3600000	3600000	3600000	3600000	3600000	3600000
Q12	~~3600000~~	3600000	3600000	3600000	3600000	3600000	3600000	3600000	3600000
Q13	~~3600000~~	3600000	3600000	3600000	3600000	3600000	3600000	3600000	3600000
Q14	~~3600000~~	3600000	3600000	3600000	3600000	3600000	3600000	3600000	3600000
Q15	~~3600000~~	3600000	3600000	3600000	3600000	3600000	3600000	3600000	3600000
Q16	~~3600000~~	3600000	3600000	3600000	3600000	3600000	3600000	3600000	3600000
Q17	~~3600000~~	3600000	3600000	3600000	3600000	3600000	3600000	3600000	3600000
Q18	~~3600000~~	3600000	3600000	3600000	3600000	3600000	3600000	3600000	3600000
Q19	~~3600000~~	3600000	3600000	3600000	3600000	3600000	3600000	3600000	3600000
Q20	~~3600000~~	3600000	3600000	3600000	3600000	3600000	3600000	3600000	3600000
					Arithmetisches Mittel alle Anfragen				3240095
					Arithmetisches Mittel von Gruppe I				2400420
					Arithmetisches Mittel von Gruppe II				2880128
					Arithmetisches Mittel von Gruppe III				3600000
					Arithmetisches Mittel von Gruppe IV				3600000

Mit BigDoc.xml

Messergebnisse von Postgres mit BigDoc.xml									
	1.Messung	2.Messung	3.Messung	4.Mesung	5.Messung	6.Messung	7.Messung	8.Messung	Arth.Wert
Q01	~~2172~~	1797	1782	1781	1750	1765	1735	1735	1762,6
Q02	~~2704~~	921	860	906	922	921	922	907	915,4
Q03	~~3600000~~	3600000	3600000	3600000	3600000	3600000	3600000	3600000	3600000
Q04	~~3600000~~	3600000	3600000	3600000	3600000	3600000	3600000	3600000	3600000
Q05	~~3600000~~	3600000	3600000	3600000	3600000	3600000	3600000	3600000	3600000
Q06	~~3600000~~	3600000	3600000	3600000	3600000	3600000	3600000	3600000	3600000
Q07	~~3600000~~	3600000	3600000	3600000	3600000	3600000	3600000	3600000	3600000
Q08	~~3600000~~	3600000	3600000	3600000	3600000	3600000	3600000	3600000	3600000
Q09	~~3600000~~	3600000	3600000	3600000	3600000	3600000	3600000	3600000	3600000
Q10	~~3600000~~	3600000	3600000	3600000	3600000	3600000	3600000	3600000	3600000
Q11	~~3600000~~	3600000	3600000	3600000	3600000	3600000	3600000	3600000	3600000
Q12	~~3600000~~	3600000	3600000	3600000	3600000	3600000	3600000	3600000	3600000
Q13	~~3600000~~	3600000	3600000	3600000	3600000	3600000	3600000	3600000	3600000
Q14	~~3600000~~	3600000	3600000	3600000	3600000	3600000	3600000	3600000	3600000
Q15	~~3600000~~	3600000	3600000	3600000	3600000	3600000	3600000	3600000	3600000
Q16	~~3600000~~	3600000	3600000	3600000	3600000	3600000	3600000	3600000	3600000
Q17	~~3600000~~	3600000	3600000	3600000	3600000	3600000	3600000	3600000	3600000
Q18	~~3600000~~	3600000	3600000	3600000	3600000	3600000	3600000	3600000	3600000
Q19	~~3600000~~	3600000	3600000	3600000	3600000	3600000	3600000	3600000	3600000
Q20	~~3600000~~	3600000	3600000	3600000	3600000	3600000	3600000	3600000	3600000
					Arithmetisches Mittel alle Anfragen				3240133,9
					Arithmetisches Mittel von Gruppe I				2400587,53
					Arithmetisches Mittel von Gruppe II				3000152,57
					Arithmetisches Mittel von Gruppe III				3600000
					Arithmetisches Mittel von Gruppe IV				3600000

E TABELLE MIT ALLEN BEWERTUNGEN

Datenbank		2	Berkeley DB XML (Nativ)				eXist (Nativ)				PostgreSQL (Enable)			
Kriterien	Detailskriterium	Gewichtung	Punkte [0-2]	Bemerkung zur Bewertung	Punkte nach Gewicht	Prozent %	Punkte [0-2]	Bemerkung zur Bewertung	Punkte nach Gewicht	Prozent %	Punkte [0-2]	Bemerkung zur Bewertung	Punkte nach Gewicht	Prozent %
Zugriffsvariante														
API	JDBC	2	0	nicht erfüllt	0	0	1	teils erfüllt	2	50	2	erfüllt	4	100
	DOM	2	2	erfüllt	4	100	2	erfüllt	4	100	1	teils erfüllt	2	50
	WebDAV	1	0	nicht erfüllt	0	0	2	erfüllt	2	100	0	nicht erfüllt	0	0
	XML:DB	1	0	nicht erfüllt	0	0	2	erfüllt	2	100	0	nicht erfüllt	0	0
	Maxi. Punkte	**12**			**4**	**33,3**			**10**	**83,3**			**6**	**50,0**
Sprachen	C	2	0	nicht erfüllt	0	0	2	erfüllt	4	100	2	erfüllt	4	100
	C#	2	2	erfüllt	4	100	2	erfüllt	4	100	2	erfüllt	4	100
	C++	2	2	erfüllt	4	100	2	erfüllt	4	100	2	erfüllt	4	100
	PL/SQL	2	0	nicht erfüllt	0	0	0	nicht erfüllt	0	0	2	erfüllt	4	100
	JAVA	3	2	erfüllt	6	100	2	erfüllt	6	100	2	erfüllt	6	100
	PHP	2	2	erfüllt	4	100	2	erfüllt	4	100	2	erfüllt	4	100
	PERL	2	2	erfüllt	4	100	2	erfüllt	4	100	2	erfüllt	4	100
	.NET	3	1	teils erfüllt	3	50	2	erfüllt	6	100	2	erfüllt	6	100
	Maxi Punkte	**36**			**25**	**69,4**			**32**	**88,9**			**36**	**100,0**
Anfrage und Datenmanipulationssprachen														
	XQuery	2	2	erfüllt	4	100	2	erfüllt	4	100	0	nicht erfüllt	0	0
	XUpdate	2	0	nicht efüllt	0	0	2	erfüllt	4	100	0	nicht erfüllt	0	0
	XPath	1	2	erfüllt	2	100	2	erfüllt	2	100	1	teils erfüllt	1	50
	SQL/XML	2	0	nicht efüllt	0	0	0	nicht erfüllt	0	0	2	erfüllt	4	100
	SQL	2	0	nicht efüllt	0	0	0	nicht erfüllt	0	0	2	erfüllt	4	100
	Maxi Punkte	**18**			**6**	**33,3**			**10**	**55,6**			**9**	**50,0**

Datenbank		2	Berkeley DB XML (Nativ)				eXist (Nativ)				PostgreSQL (Enable)			
Kriterien	Detailskriterium	Gewichtung	Punkte [0-2]	Bemerkung zur Bewertung	Punkte nach Gewicht	Prozent %	Punkte [0-2]	Bemerkung zur Bewertung	Punkte nach Gewicht	Prozent %	Punkte [0-2]	Bemerkung zur	Punkte nach Gewicht	Prozent %
Datensicherheit														
	Authentisierung des Benutzer	2	1	durch OS	2	50	2	erfüllt	4	100	2	erfüllt	4	100
	Zugriffsrechtverwaltung	2	1	durch OS	2	50	2	erfüllt	4	100	2	erfüllt	4	100
	Maxi Punkte	**8**			**4**	**50,0**			**8**	**100,0**			**8**	**100,0**
Transaktionsverwaltung														
	ACID-Eigenschaft	4	2	erfüllt	8	100	1	teils erfüllt	4	50	2	erfüllt	8	100
	Sperrebene eines XML-Dok	2	2	erfüllt	4	100	1	Interne gesetze ebene	2	50	1	Knotenstuffe	2	50
	Anzahl Sperrarten	2	1	1 Sperrart(Intern)	2	50	1	1 Sperrart(Intern)	2	50	2	2 Sperrart(TS)	4	100
	Isolationsstuffe	3	2	erfüllt	6	100	1	Interne	3	50	2	erfüllt	6	100
	Maxi Punkte	**22**			**20**	**90,9**			**11**	**50,0**			**20**	**90,9**
Schemaunterstützung														
	DTD (Dokumenttype Definition)	**2**	2	erfüllt	4	100,0	2	erfüllt	4	100,0	0	nicht erfüllt	0	0,0
	XML Schema	**2**	2	erfüllt	4	100,0	2	erfüllt	4	100,0	2	erfüllt	4	100,0
	Maxi Punkte	**8**			**8**	**100,0**			**8**	**100,0**			**4**	**50,0**
Funktionalität (Total Maxi Punkte)		**104**			**67**				**79**				**83**	
Plattformunterstützung														
	Microsoft Windows	2	1	teils erfüllt	2	50	2	erfüllt	4	100	1	teils erfüllt	2	50
	Linux	2	2	erfüllt	4	100	2	erfüllt	4	100	2	erfüllt	4	100
	Mac OS X	2	2	erfüllt	4	100	2	erfüllt	4	100	2	erfüllt	4	100
	Unix	2	2	erfüllt	4	100	2	erfüllt	4	100	2	erfüllt	4	100
	Maxi Punkte	**16**			**14**	**87,5**			**16**	**100,0**			**14**	**87,5**

Detailskriterium	Gewichtung	Berkeley DB XML (Nativ)				eXist (Nativ)				PostgreSQL (Enable)			
	2	Punkte [0-2]	Bemerkung zur Bewertung	Punkte nach Gewicht	Prozent %	Punkte [0-2]	Bemerkung zur Bewertung	Punkte nach Gewicht	Prozent %	Punkte [0-2]	Bemerkung zur Bewertung	Punkte nach Gewicht	Prozent %
Performanz (XQuerie)													
SmallDoc.xml			QZeit(ms)				Qzeit(ms)				QZeit8ms)		
Gruppe I	5	2	18	10	100	1	20	5	50	0	2400005	0	0
Gruppe II	5	2	26	10	100	1	44	5	50	0	300001	0	0
Gruppe III	5	1	31	5	50	2	23	10	100	0	3600000	0	0
Gruppe IV	5	2	20	10	100	1	87	5	50	0	3600000	0	0
Maxi Punkte	**40**			**35**	**87,5**			**25**	**62,5**			**0**	**0,0**
MiddleDoc.xml													
Gruppe I	5	1	4183	5	50	2	133	10	100	0	2400420	0	0
Gruppe II	5	2	22553	10	100	1	71762	5	50	0	2880128	0	0
Gruppe III	5	2	17498	10	100	1	26928	5	50	0	3600000	0	0
Gruppe IV	5	1	485	5	50	2	232	10	100	0	3600000	0	0
Maxi Punkte	**40**			**30**	**75,0**			**30**	**75,0**			**0**	**0,0**
BigDoc.xml													
Gruppe I	5	1	8774	5	50	2	331	10	100	0	2400587	0	0
Gruppe II	5	2	51008	10	100	1	109821	5	50	0	3000152	0	0
Gruppe III	5	2	38486	10	100	1	61142	5	50	0	3600000	0	0
Gruppe IV	5	1	1404	5	50	2	442	10	100	0	3600000	0	0
Maxi Punkte	**40**			**30**	**75,0**			**30**	**75,0**			**0**	**0,0**
Performanz (Total Punkte)	**120**			**95**	**79,2**			**85**	**70,8**			**0**	**0,0**
Tota Maxi Punkte	**232**			**168**	**72,4**			**172**	**74,1**			**93**	**40,1**

F BESCHREIBUNG VON EASY XML-DATABASE-BENCHMARK

Vorraussetzung und Konfiguration

Bevor *Easy XML-Database-Benchmark* verwendet wird, müssen einige Aspekte berücksichtigt werden:

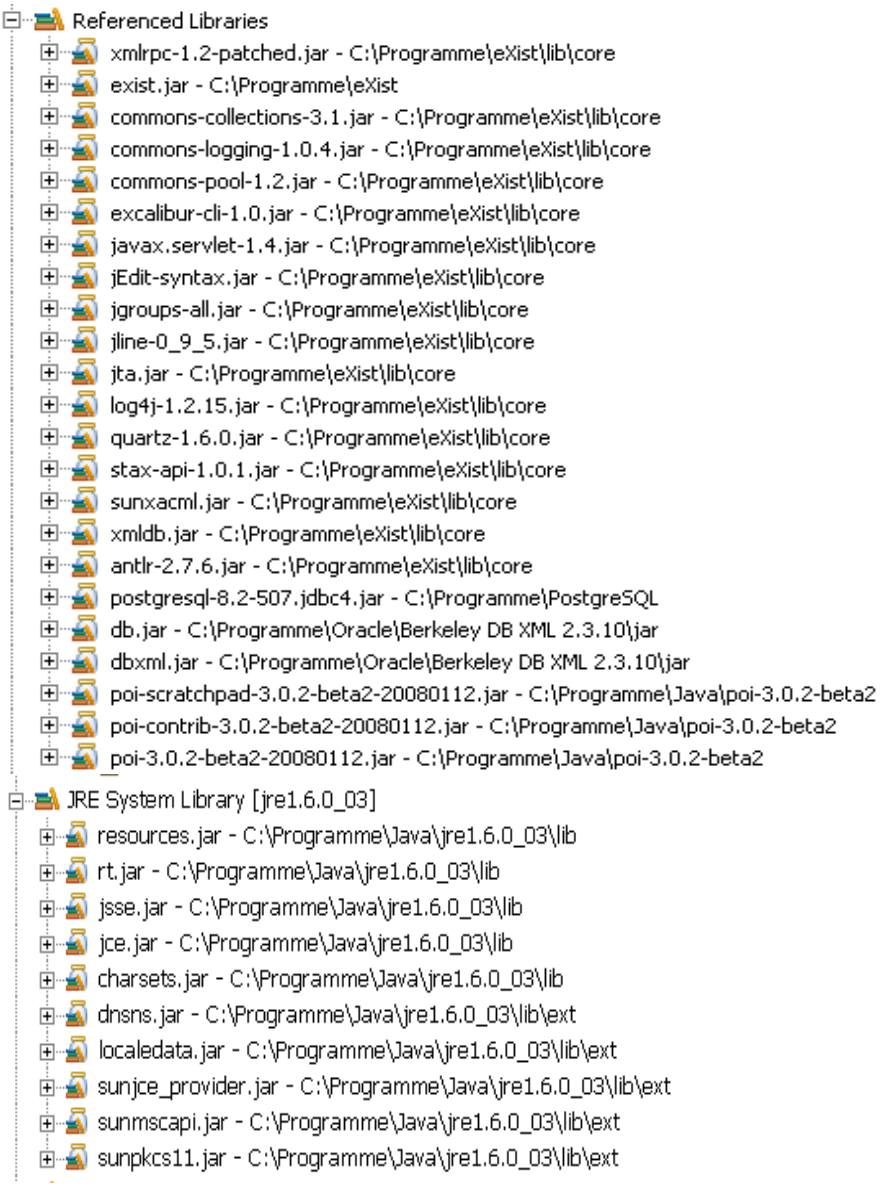
- Die kompatiblen XML Datenbankprodukte Berkeley DB XML, eXist, PostgreSQL müssen auf dem Testrechner eingerichtet werden (siehe Anhang B).
- Java 6 (JRE 1.6.03) muss auf dem System installiert sein.
- Als Plattform wird Windows XP 2002 mit Service Pack 2 verwendet.

Konfiguration

Bei der Installation jeder Datenbankinstanz auf den Testrechner, sollen Installationspfad, Passwort und Name des Datenbankadministrators notiert werden. Jene Angaben müssen in dem Init-file *Queries.ini* eingefügt werden. Die zu verwendenden XML-Dokumente müssen in jedem XML-Datenbanksystem geladen werden, ggf. muss der Datenbank-Service gestartet werden. Das Init-file beinhaltet zwanzig Anfragen aus XMark im produktspezifischen Format, das nicht verändert werden darf.
In dem init-file gibt es folgende Parameter, die angepasst werden müssen:

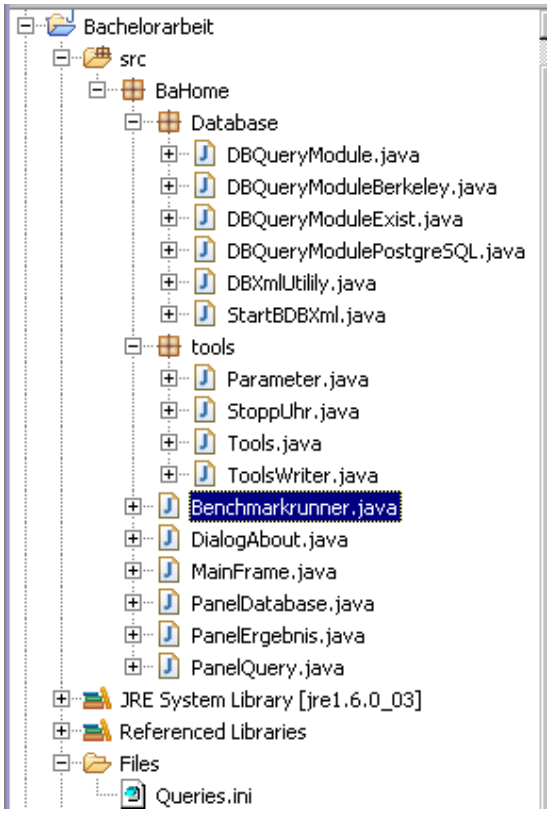
Für Berkeley DB XML
DBHome=C\:\\BaData\\BDBHome
Pfad zum Berkeley DB XML Environement
Für eXist
DBUser_Exist
Benutzername des Datenbanksystems, der über die Benutzerrechte verfügt
DBPass_Exist
Das zum Benutzernamen gehörende Passwort.
Für PostgreSQL
DBUser_PostgreSQL
Benutzername des Datenbanksystems, der über die Benutzerrechte verfügt
DBPass_PostgreSQL
Das zum Benutzernamen gehörende Passwort.

Das unterstehende Bild stellt die benötigte Bibiotheke für das Projekt.
Java Runtime Edition – 1.6
Exist
Berkeley XMLDB
PostgreSQL
POI

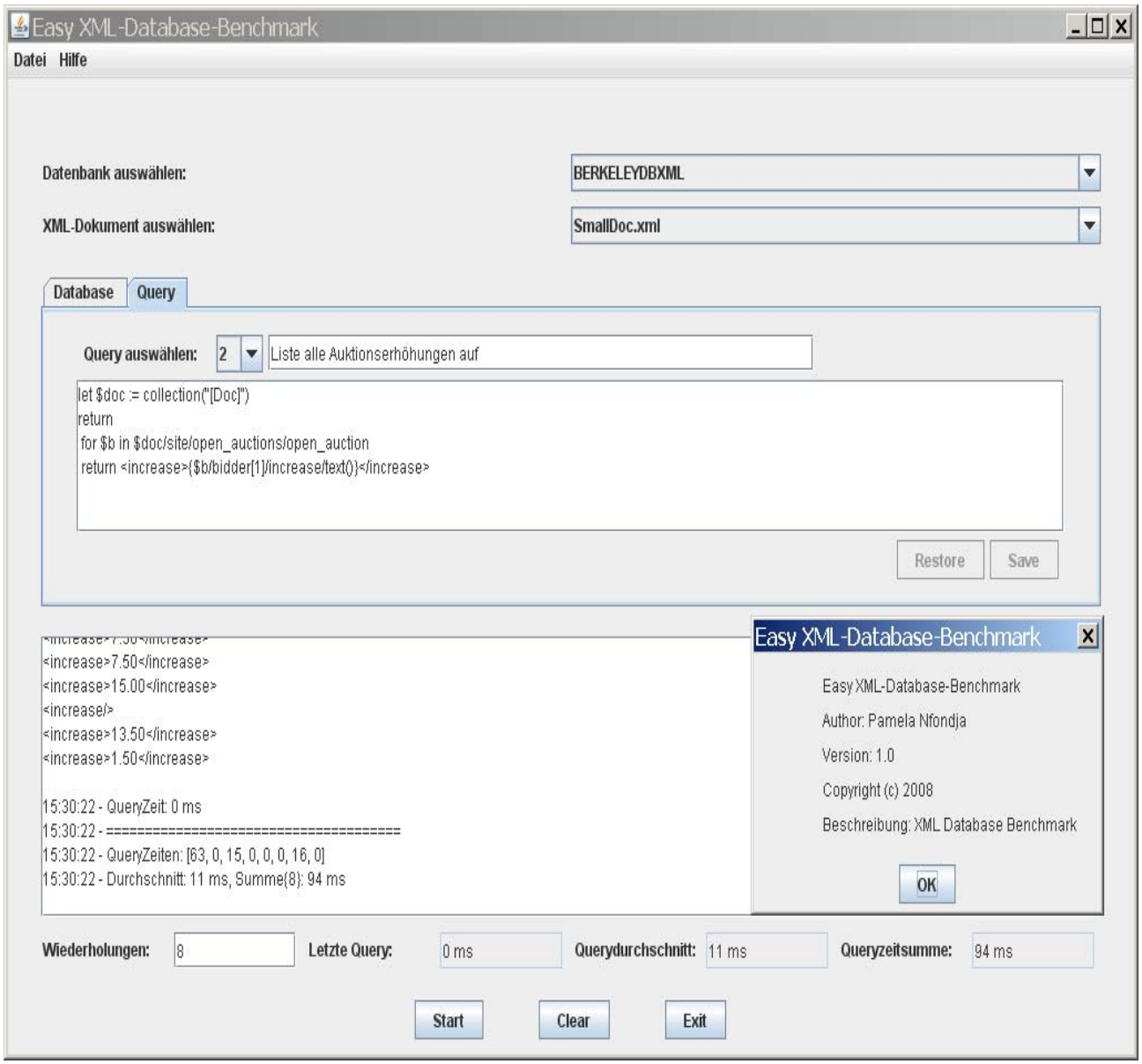


Beschreibung

Die Java Applikation *Easy XML-Database-Benchmark* ist ein GUI zur Durchführung der Messarbeiten im Rahmen dieser Bachelorarbeit. Sie lässt sich beim Aufruf des Benchmarkrunner.java starten.



Nach dem Start des ***Benchmarkrunners*** stehen dem User folgende Möglichkeiten zur Verfügung:

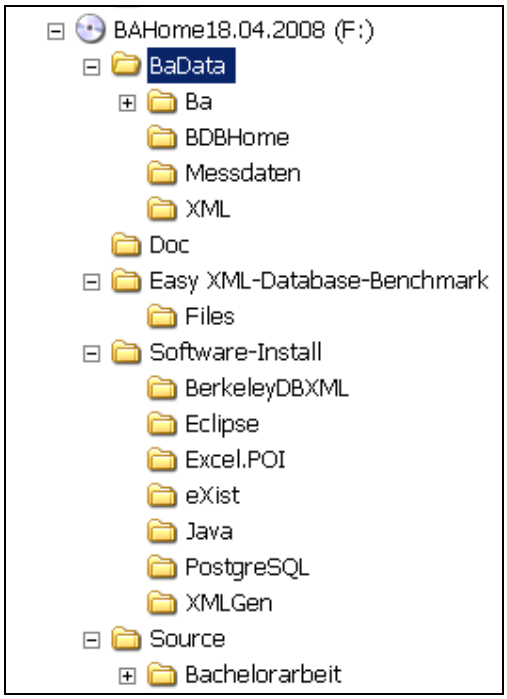


- Auswahl einer XML Datenbank:
Es muss die zu testende Datenbank ausgewählt werden:
Berkeley DB XML
EXist
PostgreSQL
- Auswahl eines XML- Dokuments:
An dieser Stelle muss eins der drei XML–Dokumente ausgewählt werden, das in der Anfrage verwendet werden soll:
SmallDoc.xml
MiddleDoc.xml
BigDoc.xml
- Auswahl einer Query aus XMark
Im Reiter *Query* kann eine Query ausgewählt, editiert und abgespeichert werden

Entsprechend der gewählten Datenbank wird diese Query auch für diese Datenbank in einer Ini-Datei abgespeichert. So können beim nächsten Start die Queries aus der Ini-Datei wieder herausgelesen werden.
Im Reiter *Database* kann die Berkeley-Datenbank neu aufgebaut werden.
Es werden hier die drei XML-Dokumente aus der Berkeley-Datenbank gelöscht und neu eingelesen.
Sofern alle Parameter eingestellt sind, wird der Button *Start* gedrückt. Dies startet die Anfrageverarbeitung.

G CD-ROM INHALT

Zu dieser Bachelorarbeit wird eine CD mit folgendem Inhalt mitgeliefert:



1. **BaData:**
   - **Ba:** Tablespace für PostgreSQL
   - **BDBHome:** Berkeley DB XML environment
   - **Messdaten:** Excell-Bewertungstabelle und Excell-Messdaten
   - **XML:** die drei generierten XML-Dokumente

2. **Doc:**
   - Die gesamte schriftliche Bachelorarbeit (Bachelorarbeit-Nfondja.pdf) und die Zusammenfassung der Bachelorarbeit auf Deutsch und Englisch

3. **EasyXML-Database-Benchmark:**
   - *Start.bat:* ein Batch Programm zum Starten von Easy XML-Database-Benchmark .

4. **Software-Install:**
   - Die benötigte Software und die XML-Datenbanken

5. **Source:**
   - Der Programmcode von *Easy XML-Database-Benchmark*

H VERZEICHNISSE

ABBILDUNGSVERZEICHNIS

TABELLEVERZEICHNIS

ABKÜRZUNGSVERZEICHNIS

AES	Advanced Encryption System
API	Application Programming Interface
BDBXML	Berkeley DB XML
BLOB	Binary Large Object
CLOB	Character Large Object
DBMS	Datenbank Management System
DOM	Document Object Model
EXD	Enable XML Datenbank
KB	Kilobyte
MB	Megabyte
NXD	Native XML Datenbank
SAX	Simple API for XML
SAX	Simple API for XML
SQL	Structured Query Language
SSL	Secure Socket Layer
TCP/IP	Transmission Control Protocol / Internet Protocol
XML	Extensible Markup Language
XML-RPC	XML Remote Procedure Call
XPath	XML Path Language
XQuery	XML Query Language
XUpdate	XML Update Language

LITERATURVERZEICHNIS

[ACAB] Akmal B. Chaudhri; A. Rashid; R. Zicari. XML Data Management Native XML and XML –Enabled Database Systems. Pearson Education, Inc März 2003

[Basti06] Bastian, G.: XML-Datenbanken in der Praxis. Bomots verlag, 2006

[BLL03] Bressan, S.; Lee, M. L.; Li, Y. G.; Lacroix, Z.; Nambiar, U. B.: XML Management System Benchmarks, Addison-Wesley, 2003

[Brun04] Brundage, M.: XQuery: The XML Query Language. Addison-Wesley, Boston, 2004

[Dan06] Danny, B.: The definitive guide to Berkeley DB XML. Spring verlag New York, Inc. 2006

[HaMe05] Harold, E. R.; Means, W. S.: XML in a Nutshell. Deutsche Ausgabe der 3. Auflage. O'Reilly Verlag GmbH, Köln, 2005

[Hara03] Harald, S.: XML und Datenbanken –Konzepte und Systeme. Carl Hanser Verlag München Wien 2003

[Jens03] Jens, W. Einsatzgebiete von Nativen XML-Datenbanken, Dipl. Arbeit, 20 Nov. 2003, Haw Hamburg.

[KlMe] KlMe03 Klettke, M.; Meyer, H.: XML & Datenbanken – Konzepte, Sprachen und Systeme.
dpunkt.verlag, Heidelberg, 2003

[MeBu06] Melton, J.; Buxton, S.: Querying XML: XQuery, XPath and SQL/XML in Context.

[Meie03] Meier, W. M.: eXist Native XML Database.

[Oppl05] Oppliger, R.: Contemporary Cryptography. Artech House, Norwood, 2005

[Peter03] Peter, E.: PostgreSQL das offizielle Handbuch. PostgreSQL Global Development Group. Verlag moderne industrie.1996-2003.

[SBC+04] Steegmans, B.; Bourret, R.; Cline, O.; Guyennet, O.; Kulkarni, S.; Priestley, S.; Sylenko, V.; Wahli, U.: XML for DB2 Information Integration. First Edition. International Business Machines Corporation, 2004.

[Seem03] Seemann, M.: Native XML-Datenbanken im Praxiseinsatz. Software & Support Verlag GmbH, Frankfurt, 2003

[SWK02] Schmidt, A.; Waas, F.; Kersten, M.; Carey, M. J.; Manolescu, I.; Busse, R.: XMark: A Benchmark for XML Data Management. *Proceedings of the 28th VLDB Conference* (VLDB Conference, Hong Kong, China), 2002,

[SWKF01b] 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.

[VaCM05] Vakali, A.; Catania, B.; Maddalena, A.: XML Data Stores: Emerging Practices. IEEE Computer Society, 2005, http://oswinds.csd.auth.gr/papers/ic05.pdf

INTERNETQUELLEN

[BDBXML] http://www.oracle.com/technology/products/berkeley-db/index.html [letzter Zugriff 07.04.2008]

[BDBXMLDoc] http://www.oracle.com/database/berkeley-db/xml/index.html [letzter Zugriff 07.04.2008]

[BDBXMLDoc1] http://www.oracle.com/technology/documentation/berkeley-db/xml/gsg_xml_txn/cxx/index.html [letzter Zugriff 07.04.2008]

[BDBXMLFO] http://forums.oracle.com/forums/profile.jspa?userID=608268 [letzter Zugriff 07.04.2008]

[BDBXMLSEN] Berkeley DB Reference Guide: Encryption: http://www.oracle.com/technology/documentation/berkeley-db/db/ref/env/encrypt.html [letzter Zugriff 07.04.2008]

[BDBXMLTRA] Berkeley DB Reference Guide: Degrees of isolation: http://www.sleepycat.com/docs/ref/transapp/read.html [letzter Zugriff 07.04.2008]

[Bour] www.rpbourret.com [letzter Zugriff 07.04.2008]

[Bour02b] Ronald Bourret : *XML Database Products*. http://www.rpbourret.com/xml/XMLDatabaseProds.html [letzter Zugriff 07.04.2008]

[BSDTEXT] http://wwwmaster.postgresql.org/about/licence [letzter Zugriff 07.04.2008]

[EXIST] http://www.exist-db.org/ [letzter Zugriff 07.04.2008]

[EXISTDoc] http://exist-db.org/documentation.html [letzter Zugriff 07.04.2008]

[EXISTROA] eXist: Current State, Features and Roadmap: http://demo.exist-db.org/roadmap.xml [letzter Zugriff 07.04.2008]

[GNUGPLLi] www.gnu.org/licenses/ [letzter Zugriff 07.04.2008]

[HowToXmlgen] http://monetdb.cwi.nl/xml/faq.txt [letzter Zugriff 07.04.2008]

[MVCPCL] http://pclc.pace.edu/~bergin/mvc/mvcgui.html [letzter Zugriff 07.04.2008]

[PGSPER] http://www.postgresql.org/files/documentation/books/pghandbuch/html/explicit-locking.html [letzter Zugriff 07.04.2008]

[PGSQL] http://www.openminds.co.uk/high_availability_solutions/databases/postgresql.html [letzter Zugriff 07.04.2008]

[PGSQL] http://www.postgresql.org/docs/8.2/interactive/datatype.html [letzter Zugriff 07.04.2008]

[PGSQLDE] http://www.postgres.de/ [letzter Zugriff 07.04.2008]

[PGSQLDoc] http://www.postgresql.org/files/documentation [letzter Zugriff 07.04.2008]

[PGSQLFOR] http://www.pg-forum.de/konfiguration/1984-postgresql-xml.html#post9832 [letzter Zugriff 07.04.2008]

[PGSQLXML] http://www.postgresql.org/docs/8.2/static/datatype-xml.html [letzter Zugriff 07.04.2008]

[PGSQLXMLFunc] http://developer.postgresql.org/pgdocs/postgres/functions-xml.html [letzter Zugriff 07.04.2008]

[PGSQLXMLTyp] http://developer.postgresql.org/pgdocs/postgres/datatype-xml.html [letzter Zugriff 07.04.2008]

[PXDM] XQuery 1.0 und Xpath 2.0 Data Model. W3C Recommendation http://www.w3.org/TR/2007/REC-xpath-daamodel. [letzter Zugriff 07.04.2008]

[Sourceforge] http://www.sourceforge.net/ [letzter Zugriff 07.04.2008]

[W3C] http://www.w3.org/ [letzter Zugriff 07.04.2008]

[W3CXML] World Wide Web Consortium, Spezifikation XML, http://www.w3.org/XML/ [letzter Zugriff 07.04.2008]

[WCXQUERY] World Wide Web Consortium, Working Draft XQuery, http://www.w3.org/XML/Query [letzter Zugriff 07.04.2008]

[XMARK] http://monetdb.cwi.nl/xml/ [letzter Zugriff 07.04.2008]

[xmlgen] http://www.xml-benchmark.org/ [letzter Zugriff 07.04.2008]

[XQUERYREQ] XML Query (XQuery) Requirements, http://www.w3.org/TR/xquery-requirements/ [letzter Zugriff 07.04.2008]

Versicherung über die Selbständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §22(4) ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Ort, Datum	Unterschrift des Studenten