Funktionsumfang und Eignung von XML-Datenbanken für Multimedia- und Metadaten

Tom Neumerkel, Robert Manthey
Technische Universität Chemnitz
Fakultät für Informatik
D-09111, Chemnitz, Deutschland
{tomn,mrob}@hrz.tu-chemnitz.de

Zusammenfassung

Diese Arbeit untersucht die Eignung verschiedener Extensible Markup Language (XML)-Datenbanken für die Ablage und Verwaltung von Multimedia- und Metadaten anhand ihrer für diesen Einsatzzweck relevanten und benötigten Eigenschaften sowie durchgeführter Benchmarks.

1 Einleitung

Im Rahmen des Projektes *ValidAX* werden verschiedene Werkzeuge für Analyse, Annotation, Archivierung und Recherche von Audio- und Video-Material[1] entwickelt, welche meist in der betriebssystemunabhängigen Programmiersprache Java implementiert sind und die Handhabung und weitere Verwendung dieser Daten vereinfachen sollen. Hierzu werden Informationen durch Verfahren wie z.B. der Sprach- und Texterkennung aus den Daten extrahiert und als Metadaten in XML-Form in Dateien gespeichert, wie in Quelltext 1 beispielhaft zu sehen, wobei Struktur, Größe, Variabilität und Komplexität dieser Metadaten durch die multimedialen Quelldaten sowie die zu ihrer Analyse eingesetzten Verfahren bestimmt sind.

Aufgrund dessen steigt der Handhabungsaufwand dieser Dateien und die Möglichkeiten damit verbundener Fehler bei größerer Anzahl erheblich an und behindern sowohl einen mehr als prototypischen Einsatz der Werkzeuge, als auch eine Möglichkeit zur Lösung der Problematik durch den Einsatz von Relational Database Management Systems (RDBMSs)[2].

2 Projektspezifische Anforderungen an die Softwarelösung

Bedingt durch dieses Anwendungsszenario scheint eine Datenbanksoftware eine geeignete Lösung darzustellen. Die daran gestellten Anforderungen werden im folgenden Kapitel herausgearbeitet

Essentiell ist die Eigenschaft XML-Daten verwalten zu können, dies umfasst vor allem die folgenden Punkte:

- Nutzerschnittstelle[3] zur Verwaltung der Datenbank an sich
- Nutzerschnittstelle um vorhandene XML-Daten in das System importieren, die Datenbank sichern und wiederherstellen zu können
- Nutzung einer standardisierten Anfragesprache zur einfacheren Handhabung
- Möglichst granulare Rechteverwaltung für einen Mehrbenutzerbetrieb
- Volltextindex, um die Suche zu beschleunigen
- Transaktionsfähigkeit[4] zur Vermeidung störender gegenseitiger Beeinflussungen bei mehreren gleichzeitigen Abfragen
- Sperrmöglichkeiten[5], um Teile des Datenbestandes während der Bearbeitung vor Zugriffen durch andere Nutzer zu sperren

Die Datensätze bestehen aus den Metadaten in XML-Form und Audio- und Videodateien in Formaten wie *mp3*, *mp4* und (unkomprimiertem) *avi* sowie Zusatzdateien wie *iso*. Daher ist die Ablage der Audio- und Videodateien formatunabhängig notwendig. Für verschiedenste Analyse-, Annotations-, Recherche-, Weiterverarbeitungs- und auch Archivierungszwecke müssen aus dem selben Quelldatensatz unterschiedliche Versionen erzeugt, vorgehalten sowie verwaltet werden, woraus weitere wichtige Anforderungen an die Handhabung dieser Daten folgen. Speziell vor dem Hintergrund der damit einhergehenden erheblichen Datenmengen, deren Verteilung zu Zwecken des Lastausgleichs sowie zur Distribution und Weiterverarbeitung ist ein effizientes Management größerer Dateimengen im Bereich von etlichen Gigabyte bis mehreren Terabyte je Datei vorteilhaft, entsprechend den Anforderungen an „Big Data“.

Die Nutzung der verwalteten Daten erfolgt durch andere Programme welche zum einen Metadaten und Dateiversionen für interne Analyse- und Verarbeitungsschritte abfragen und verwalten lassen, zum anderen um Suchanfragen von Nutzern beantworten zu können.

Um die Analyseverfahren zu skalieren und die Werkzeuge verteilen zu können, sowie eine effiziente Resourcennutzung zu ermöglichen, erscheint ein Client-Server-Betrieb vorteilhaft. Sehr resourchenintensive, latenzkritische Werkzeugteile könnten eine direkte Integration in diese erzwingen, sodass ein eingebetteter (embedded) Betrieb der Datenbank möglich sein sollte. Weiterhin erforderlich ist der Einsatz unter verschiedenen Betriebssystemen wie Windows Server, Linux/Unix und Mac OS, sowie Schnittstellen zur Anbindung existierender und zukünftig wahrscheinlicher Werkzeuge mittels Programmiersprachen wie Java-, C#- und PHP: Hypertext Preprocessor (PHP).

---

1. z.B. im (Jahres-)Programm eines Fernsehsenders
2. Zur Erklärung von RDBMSs siehe [12]
3. Nutzerschnittstelle steht stellvertretend für Command-line interface (CLI) und optional Graphical User Interface (GUI)
4. Modelliert erfolgt die Abarbeitung mehrerer Transaktionen nacheinander, so dass keine Beeinflussung untereinander stattfindet.
5. „*Lock*“ genannt

```
<speech_recognition>
 <file name="TV-20110327-1719-0301.webm.h264.mp4" videoLength="102000">
  <speaker_recogniton>
    <!-- Angaben zum Sprecher -->
  </speaker_recogniton>
  <segment id="0" speaker_id="0" startTime="0" endTime="24000">
   <recognitionEvent id="0" confidence="0.5455173" duration="22710" startTime="60">
    <word startTime="1229" duration="251" confidence="0.6199378">V.</word>
    <word startTime="1481" duration="251" confidence="0.124626793">ein</word>
    <word startTime="1733" duration="352" confidence="0.7838184">hundert</word>
    <word startTime="2086" duration="755" confidence="0.783242166">Sekunden</word>
    <word startTime="2953" duration="574" confidence="0.6497807">und</word>
    <word startTime="3558" duration="705" confidence="0.30492723">Landtagswahl</word>
    <!-- mehr Worte ;) -->
   </recognitionEvent>
   <!-- mehr Events -->
  </segment>
  <segment id="1" speaker_id="0" startTime="24000" endTime="49000">
    <!-- mehr Events -->
  </segment>
    <!-- mehr Segmente -->
 </file>
</speech_recognition>
```

Quelltext 1: Durch Spracherkennung erzeugte XML-Struktur

Um die Handhabung zu vereinfachen sollte in nur einer Anfragesprache nur eine Schnittstelle zur Datenquelle angesprochen werden müssen und die Fähigkeit zum Zugriff auf beliebige Teilstücke von Mediendaten bieten.

3 Anfrage- und Update-Sprachen

Die genannte Vielfalt der bisher bei den Werkzeugen des Projektes verwendeten verschiedenen XML-Konstrukte führt zu Handhabungsproblemen und -fehlern und soll durch die nachfolgend vorgestellte einheitlichere Anfragemethodik weitestgehend beseitigt werden.

Bei XML-Datenbanken gibt es zwar zur Verwaltung der Datenbank keine einheitliche Sprache wie Structured Query Language (SQL) bei RDBMS, aber zum Anlegen, Bearbeiten und Löschen von Datenbanken bzw. der Verwaltung von Berechtigungen dienen SQL-ähnliche Konstrukte. Zur Abfrage und Manipulation von Daten haben sich die Standards XML Path Language (XPath) [22], XML Query Language (XQuery) [23] bzw. XQuery Update Facility (XQUF) [25] des World Wide Web Consortium (W3C) durchgesetzt.

Für alle nachfolgend untersuchten *nativen*[6] XML-Datenbanken haben sich XPath und XQuery als Anfragesprachen etabliert. Beide wurden vom W3C definiert um den Zugriff auf und Umgang mit XML-Daten zu vereinheitlichen. „Die primäre Aufgabe von XPath ist die Adressierung von Knoten innerhalb einer XML-Struktur.“ [22] Zur Formulierung der Anfragen wird XQuery, meist dem For, Let, Where, Order by, Return (FLWOR)-Prinzip folgend verwendet, wie in Quelltext 2 gezeigt.

XQUF erweitert XQuery um die Möglichkeit Daten zu verändern. Es lassen sich so Knoten innerhalb der Struktur einfügen, löschen, verändern und umbenennen, wie in Quelltext 3 dargestellt.

4 Vorstellung einzelner Softwareprodukte

Im folgenden werden die einzelnen betrachteten Softwareprodukte auf ihre Eigenschaften hin untersucht und eine Vorauswahl für die weiteren detailierteren Eignungsanalysen getroffen.

```
<ul>{
for $x in speech_recognition//word
where contains( $x ,"Landtagswahl")
return
  <li>{data($x/ancestor::*[last()]
       /file/@name)}
      -- @ {data($x/../../@startTime)
            + data($x/../@startTime)
            + data($x/@startTime)}
  </li>
}</ul>
```

Quelltext 2: XQuery zur Abfrage des Dateipfades und der Position innerhalb der Video-Datei, anhand von Suchbegriffen. Die Ausgabe erfolgt als HTML-Fragment.

```
for $r in collection('/db/tagesschau')
 /speech_recognition[
   starts-with( ./file/@name, 'TV-')]
return update value $r/file/@name
  with substring($r/file/@name, 4)
```

Quelltext 3: XQuery mit XQUF zum Ändern des Dateinamens der Mediendatei in allen Dokumenten der Collection

6. *Nativ* bezeichnet XML-Datenbanken, die speziell für die Verarbeitung von XML entwickelt wurden, wohingegen *xml-enabled* für Datenbanken Verwendung findet, die um Funktionen zur XML-Verarbeitung erweitert wurden

APACHE XINDICE ist eine der ersten nativen XML-Datenbanken und stammt aus dem Jahr 2002. Da seit dem letzten Release im Dezember 2007 keine Weiterentwicklung stattfand wurde das gesamte Projekt im August 2011 in Apache Attic[7] verschoben und hat damit das Ende seines Lebenszyklus erreicht.

BASEX entstand als Projekt in der Arbeitsgruppe Datenbanken und Informationssysteme des Fachbereichs Informatik und Informationswissenschaft an der Universität Konstanz. Diese Arbeitsgruppe beteiligt sich auch heute an der Weiterentwicklung (vgl. [7] u. [16]). Der Einstieg in die Arbeit mit BaseX wird durch das mitgelieferte GUI erleichtert. Damit lassen sich sofort erste visuelle Erkundungen der Datenbank durchführen und XQuery-Anfragen durchführen. Das Projekt bemüht sich, W3C Standards umzusetzen bzw. diesen eng zu folgen (vgl. [2]). Zusätzliche Funktionen werden durch XQuery Module bereitgestellt, z.B. Funktionalitäten um mit ZIP-Archiven umzugehen oder Volltext-Operationen (vgl. [4]).

BERKELEYDB XML ist eine embedded Datenbank, die heute von Oracle betreut wird. Sie ist dual-lizenziert (kommerziell & open-source) [14]. BerkeleyDB XML baut auf BerkeleyDB auf. Damit kann sie Funktionen von dieser, wie „Replikation zur Hochverfügbarkeit“ [15], nutzen. Die letzte Version (2.5.16) wurde am 22. Dezember 2009 veröffentlicht. Eine Nutzerverwaltung innerhalb der Datenbank ist nicht vorgesehen. Die XML-Daten werden in sogenannten „Containern“ zusammen mit ihren Meta-Informationen, z.B. Indizes, gespeichert. Optional ist eine Validierung der XML-Daten gegen ein Schema[8] möglich (vgl. [11]). Zur Wartung und Verwaltung von Containern ist zusätzlich ein Kommandozeilen-Werkzeug enthalten, das die gleichen Operationen wie die Bibliotheken unterstützt.

EXIST-DB wurde als native XML-Datenbank entwickelt. In der aktuellen Version 2.0 wurde sie zu einer „all-in-one Lösung für die Anwendungsentwicklung“ [6] ausgebaut. Die Verwaltungsoberfläche und die Online-Dokumentation stellen selbst eine Demonstration der Einsatzmöglichkeiten dar. Die Datenbank kann unter anderem als eigenständiger Server oder embedded betrieben werden (vgl. [5]). Die Installation und der Betrieb kann innerhalb eines unprivilegierten Nutzerkontos erfolgen. eXide, ein XQuery-Integrated Development Environment (IDE), kann als Modul in der Weboberfläche genutzt werden. Diese bietet die Möglichkeit, direkt neue Apps[9] entwickeln zu können, die sich dann zu Paketen bündeln und in andere Installationen übertragen lassen. Dokumente können in Collections und Sub-Collections zusammengefasst werden. Locking funktioniert auf Dokumentenebene. Nutzerberechtigungen sind denen in Unix-Dateisystemen nachempfunden. Wartungsaufgaben werden entweder über die Weboberfläche, Apache Ant [1] oder über ein mitgeliefertes Werkzeug, dem „Java Admin Client“ (GUI und CLI) erledigt (vgl. [5]). Wie bei BaseX gibt es auch bei eXist-db XQuery-Module, die den Funktionsumfang erweitern (vgl. [5]). Volltextindex und Caches sorgen auch hier für bessere Leistungen, Indizes müssen manuell angelegt werden.

IBM DB2 ist ein kommerzelles RDBMS der Firma IBM, welches in Version 1 1983 erschien und für diese Arbeit ab Version 9 (2006) interessant ist, da diese seither als hybride Datenbank sowohl mit relationalen als auch mit XML-Daten umgehen kann. IBM nennt die Erweiterung „pureXML“ (vgl. [3]). DB2 speichert die XML-Daten innerhalb von Tabellenspalten. Der Zugriff auf diese ist durch erweiterte XQuery-Funktionen wie *db2-fn:xmlcolumn* und *db2-fn:sqlquery* möglich. Den (kostenlosen) Einstieg bildet „DB2 Express-C“. Diese Edition bietet alle Kernfunktionen (außer Skalierung) zum Betrieb von DB2 auf einem Server in kleinen Umgebungen. Damit entwickelte Anwendungen sollen sich ohne Änderungen mit skalierbaren Editionen nutzen lassen. (vgl. [9])

MONETDB bezeichnet sich selbst als „The column-store pioneer“, die erste open-source Version (MonetDB 4) wurde 2004 veröffentlicht (vgl. [8]). Spaltenspeicher (engl. column-store) bezieht sich hierbei auf die interne Datenorganisation, die nicht, wie bei vielen anderen RDBMS üblich, zeilenorientiert sondern spaltenorientiert erfolgt. Wie bei anderen RDBMS sind Abfragen in SQL zu formulieren. Um Abfrageergebnisse als XML auszugeben bzw. XML-Daten zu importieren sollte bevorzugt „SQL/XML“[10] verwendet werden (vgl. [8]). Somit zählt MonetDB zu den xml-enabled Datenbanken. Eine XQuery-Erweiterung ist verfügbar. Damit wäre MonetDB äußerlich als native XML-Datenbank ansprechbar. Die Entwicklung dieser Erweiterung ist aber seit Mai 2011 „eingefroren“ und in der Dokumentation als „veraltet“ gekennzeichnet, womit eine zukünftige Verwendung für XML-Daten fraglich erscheint (vgl. [8]).

SEDNA, eine freie native XML-Datenbank, entstand als Projekt des „Instituts für Systemprogrammierung“ [10] an der Russischen Akademie der Wissenschaften. Im Gegensatz zu BaseX und eXist-db nutzt Sedna nicht XQUF sondern eine ähnliche Syntax, die auf die Diplomarbeit von Patrick Lehti [13] zurückgeht. Konsistente Backups der Datenbank können während des normalen Betriebs erfolgen (vgl. [19]). Optional lassen sich Nutzer und Berechtigungen je Datenbank aktivieren. Hervorzuheben sind einzelne Berechtigungen für *INSERT*, *DELETE* und *RENAME*, die eine feinere Unterscheidung zulassen als ein *WRITE* bei anderen Produkten. Angewandt werden diese auf Dokumente und Collections. Zur Verwaltung der Datenbank stehen weitere Rechte zur Verfügung (vgl. [19]).

WEBMETHODS TAMINO XML SERVER (nachfolgend kurz Tamino) ist ein proprietäres Produkt der Software AG. Viele Informationen lassen sich über Tamino nicht zusammentragen, da keine Dokumentation ohne Registrierung zugänglich ist. Die zuverlässigste Quelle bildet noch das Datenblatt [21]. Demnach wird für alle Vorgänge rund um ein XML-Dokument XQuery genutzt, ob die Erweiterung XQUF zum Einsatz kommt bleibt offen. Ebenfalls werden dort Fähigkeiten zur Replikation und Hochverfügbarkeit genannt.

5 Eignungsanalyse

Nachfolgend werden die Produkte aus Abschnitt 4 miteinander verglichen. Einen schnellen Überblick der Merkmale bietet Tabelle 1.

Die Anforderung „Big Data“ (Abschnitt 2) ist nur durch IBM DB2 ansatzweise erfüllbar. DB2 bietet zumindest den Konfigurationsparameter „*GetDataLobNoTotal*“. Dessen Wert gibt an, wie viele Bytes eines (Spalten-)Datums an den Client übertragen werden. Der Client kann dann

7. Apache Attic ist ein Archiv zur Verwaltung von Projekte der Apache Foundation, die keine Verwalter bzw. Mitwirkende haben.
8. XML-Schema ist ein W3C-Standard, er dient zur Definition des Aufbaus von XML-Dateien.
9. Modische Kurzbezeichnung für Anwendung
10. SQL/XML ist ISO/IEC Standard (ISO/IEC 9075-14:2011)

weitere „Einheiten“ dieser Größe nachladen. Eine Möglichkeit, eine bestimmte Anzahl Bytes einer Datei (eines Binär-Datums) ab einer bestimmten Position zu laden bietet keines der betrachteten Produkte, sodass diese Anforderung in den weiteren Betrachtungen nicht weiter aufgeführt wird und auf andere Weise gelöst werden muss.

Möglichkeiten der Skalierung über die Grenzen eines Servers hinaus, bieten vor allem die kommerziellen Produkte (IBM DB2 und Tamino). monetDB sieht den Betrieb mehrerer Server innerhalb einer Gruppe vor, die dazu dient, Daten zu verteilen oder zu replizieren. Die Dokumentation dazu ist allerdings äußerst knapp gehalten (vgl. [8]). Etwas ausführlicher wird die Replikation bei eXist-db beschrieben (vgl. [5]).

Die Nutzer- und Berechtigungsverwaltung der Datenbanken ist unterschiedlich ausgeprägt. BerkeleyDB XML verwaltet selbst keine Rechte oder Nutzer. BaseX kennt Nutzer, die eine Datenbank lesen oder auch ändern können bzw. das Recht besitzen neue anzulegen. eXist-db, IBM DB2, monetDB, Sedna, Tamino ermöglichen die Gruppierung von Nutzern [11]. Berechtigungen lassen sich dann auf die Datenbank, eine Collection oder einzelne Dokumente anwenden. eXist-db, IBM DB2 und Tamino können Nutzer auch gegen ein externes System (wie zum Beispiel LDAP oder Kerberos) authentifizieren

Mehrere Anfragen können alle Produkte sicher in einer Transaktion kapseln. Ein explizites Sperren von Dokumenten unterstützen nur eXist-db und IBM DB2.

Um sich von anderen Produkten abzugrenzen bieten sie jeweils zusätzliche Funktionen: eXist-db und Tamino können Dokumente automatisch versionieren, dadurch bleiben ältere Revisionen der Dokumente weiterhin erreichbar, Änderungen am Datenbestand lassen sich nachvollziehen und falls notwendig rückgängig machen. Bei BaseX und eXist-db ist für den Zugriff auf die Dokumente auch Web Distributed Authoring and Versioning (WebDAV)[12] bzw. ein RESTful-API[13] nutzbar.

6 Praxistest

BaseX, eXist-db und Sedna bieten Werkzeuge um ohne weitere Programmierung Daten importieren und Abfragen ausführen zu können. Außerdem nutzen sie mit XQuery for Java (XQJ) [24] ein einheitliches Application Programming Interface (API) für die Integration in Java-Anwendungen. Somit lasen sich diese Produkte gegeneinander austauschen. BerkeleyDB XML, IBM DB2, MonetDB, und Tamino gehen hier eigene Wege. Da die Installation und Konfiguration von IBM DB2 zu umfangreich für den Rahmen dieser Untersuchung ist, der Fortbestand des XQuery-Frontend von MonetDB in Zukunft fraglich erscheint und eine Testversion von Tamino sich nur nach vorheriger Registrierung beschaffen lässt, werden diese nachfolgend nicht weiter betrachtet.

Für die Praxistests wurden die Produkte innerhalb einer Virtuelle Maschine (VM) innerhalb von Oracle VirtualBox [20] installiert, da es selbst auf allen genannten Plattformen aus Abschnitt 2 nutzbar ist. Außerdem kann die VM einfach ex- und importiert werden, was den Transfer auf andere physische Hardware vereinfacht. Innerhalb der virtuellen Umgebung diente die Linux Distribution Ubuntu in Version 12.04 Long Term Support (LTS) als Betriebssystem, da alle getesteten Produkte auf diesem installierbar waren und keine Lizenzverletzungen beim Transfer der VM zu anderen Personen zu befürchten sind.

Die physische Grundlage bildete ein Rechner mit Core™ i5-Prozessor von Intel® (i5-3320M) mit 16GB Arbeitsspeicher und 250GB Solid-state Drive (SSD). Der VM standen 8GB Arbeitsspeicher und 20GB Speicherplatz exklusiv zur Verfügung. Die Installation von BaseX, eXist-db und Sedna verlief problemlos: BaseX über die Paketverwaltung von Ubuntu, eXist-db und sedna über den jeweiligen Installer.

BerkeleyDB XML steht nur als Quelltext zur Verfügung, dessen kompilieren weder auf dem genannten Test-System noch auch auf einem anderen Linux-System möglich war. Als mögliche Ursache hierfür wird die fehlende Weiterentwicklung und Anpassung an aktuelle Betriebssystem und Compiler-Versionen vermutet. Somit wird BerkeleyDB XML nicht weiter betrachtet.

Sedna disqualifizierte sich, da das mitgelieferte Werkzeug zur Verwaltung der Datenbank (im Release 3.5.161) unbenutzbar ist: Es konnten keine Daten importiert werden, dokumentierte Befehle wurden zurückgewiesen.

6.1 Benchmarks

Für die Durchführung von Benchmarks blieben BaseX und eXist-db übrig. Um später Suchanfragen auszuführen, wurden zuerst die Testdaten (Spracherkennung aus 1008 Sendungen der „Tagesschau in 100 Sekunden“) in die beiden Systeme importiert, um die folgenden Anfragen auf diese anzuwenden. Die Abfrage aus Quelltext 2 dient als Beispiel für typische Recherchen. Eine weitere Abfrage, Quelltext 4, kann als Grundlage weiterer Auswertungen dienen, sie zählt die Häufigkeiten der erkannten Wörter innerhalb eines bestimmten Monats. Quelltext 3 dient zum Vergleich bei Änderungen. Alle Anfragen wurden je auf 10 identische Collections ausgeführt.

Um äußere Einflüsse zu vermeiden wurden das Host-System, innerhalb dessen die VM ausgeführt wird, und die VM selbst vor Beginn der Benchmarks neu gebootet und Cron-Jobs (sowie weitere Dienste der Benutzer-Oberfläche, z.B. Update-Checks) deaktiviert. Auch zwischen den Benchmarks von BaseX und eXist-db erfolgte ein Neustart der VM.

Das Diagramm in Abbildung 1 zeigt die durchschnittliche Ausführungszeit der jeweiligen Aktion anschaulich im Vergleich. Die Tabellen 2 und 3 zeigen die Messergebnisse im Detail.

6.2 Auswertung

Der Import der Daten und explizite Indexaufbau erfolgt bei BaseX unauffällig. eXist-db benötigt beim mehrmaligen Import der selben XML-Daten in die selbe Datenbank (Collection) zusätzliche Zeit (siehe Tabelle 3, Spalte Import[b]), da sie die Dokumente nicht hinzufügt, sondern ersetzt. Die Daten für die Indexierung bleiben unverändert, der Indexaufbau erfolgt ab dem dritten Lauf (siehe Tabelle 3, Spalte Index[b]) etwas schneller.

Quelltext 3 wird bei mehrmaliger Ausführung auf den selben Datenbestand (siehe Tabelle 3, Spalte Q 3[b]) von beiden Systemen ab dem zweiten Lauf optimiert. Bei eXist-db

11. Nutzer werden Rollen, Schemata oder Gruppen zugeordnet
12. WebDAV ist ein Protokoll, um per HTTP auf Dokumente zuzugreifen. Für weitere Informationen siehe [17].
13. Representational state transfer (REST) beschreibt den Zugriff auf Ressourcen (hier (Sub-)Collections und Dokumente) über HTTP-Methoden. Adressiert werden diese Ressourcen über einen Uniform Resource Locator (URL). Für weitere Informationen siehe [18].

	BaseX	BerkeleyDB XML	eXist-db	IBM DB2	MonetDB	Sedna	Tamino
Implementiert in	Java	C++	Java	?	C/C++	C/C++	?
Server	✔	×	✔	✔	✔	✔	✔
Embedded	✔	✔	✔	×	×	×	×
Typ	native	native[e]	native	enabled	enabled[f]	native	native
Open Source	✔	dual	✔	×	✔	✔	×
Nutzerverwaltung	✔, R/W je DB	×	✔	✔	✔	✔	✔
Locking	Je DB[a]	×	Datei, Collection	Zeile	×	×	?
Transaktionen	✔	✔	(✔)[g]	✔	(✔)[h]	✔	✔
Backup & Restore	✔	✔	✔	✔	✔	✔	✔
Volltextindex	✔	✔	✔	✔	✔	✔	✔
Plattform	multi[b]	siehe Bindings	multi[b]	z/OS, multi	multi	multi	multi
Bindings bzw. Clients	Java (XML:DB[i] u. XQJ), C#, PHP[c]	C++, C#, Java, PHP, Perl, Python	Java (XML:DB u. XQJ)	C, C++, COBOL, Java (JDBC u. eigener Client), C#, .NET, PHP, Perl	Java (JDBC), Perl, Python, ODBC, MAPI (PHP)	Java (XML:DB u. XQJ), C, C#, PHP, Python, Ruby, Perl, Delphi	Java (XML:DB), JScript, ActiveX, Perl, .NET
Cluster	×	×	×	✔	✔	×	✔
Replikation	×	✔	✔	✔	✔	×	✔
Big Data	×[d]	×	×[d]	✔	×	×	?
WebDAV	✔	×	✔	×	×	×	×
RESTful API	✔	×	✔	×	✔	×	×
Versionierung	×	×	✔	×	×	×	✔

a Einschränkungen siehe auch: `http://docs.basex.org/wiki/Transaction_Management`
b Da in Java implementiert, ist das Produkt auf allen Systemen lauffähig, für die es eine Java Vitual Machine gibt, z.B. Windows, Linux, Mac OS
c Nur „Socket-Wrapper“, keine Namensraum-Unterstützung
d Kann nur mit kleinen Dateien umgehen, werden im Dateisystem des Servers gespeichert, unterliegen dessen Beschränkungen
e Nutzt BerkeleyDB zur Speicherung der Daten
f siehe Informationen im Abschnitt 4
g Wird nur in Beispielen gezeigt, auf Atomicity, Consistency, Isolation and Durability (ACID)-Konformität wird nicht hingewiesen
h Eingeschränkt, wird durch ein Modul realisiert, nur im SQL-Frontend verfügbar
i Nur embedded nutzbar

Tabelle 1: Die Tabelle zeigt die Unterstützung der Anforderungen durch die Produkte.

Lauf	Import[a]	Import[b]	Index[a]	Index[b]	Q 2[a]	Q 2[b]	Q 4[a]	Q 4[b]	Q 3[a]	Q 3[b]
1	4,296	2,400	1,506	1,132	0,201	0,153	28,234	28,550	0,077	0,083
2	2,833	2,450	1,125	1,123	0,107	0,140	27,848	28,605	0,071	0,007
3	2,518	2,201	1,068	1,164	0,105	0,144	28,277	29,510	0,053	0,011
4	2,416	2,082	1,074	1,134	0,097	0,149	37,804	29,102	0,056	0,016
5	2,385	2,118	1,234	1,201	0,098	0,136	28,477	28,679	0,053	0,009
6	2,629	2,204	1,086	1,147	0,090	0,146	28,675	28,310	0,053	0,008
7	2,267	2,290	1,134	1,122	0,109	0,135	28,684	28,566	0,055	0,010
8	2,452	2,223	1,097	1,138	0,103	0,156	29,190	28,194	0,047	0,018
9	2,371	2,287	1,101	1,156	0,092	0,139	27,731	28,131	0,055	0,011
10	2,410	2,310	1,088	1,126	0,109	0,156	27,817	28,317	0,076	0,009
Durchschnitt	2,658	2,257	1,151	1,144	0,111	0,145	29,274	28,596	0,060	0,018
Median	2,434	2,255	1,099	1,136	0,104	0,145	28,377	28,558	0,055	0,011

a Ausführung der Anfrage auf 10 identische Collections
b Mehrmalige Ausführung auf die selbe Collection

Tabelle 2: Die Tabelle zeigt die detaillierte Ausführungsdauer der Anfragen (Import, Indexeraufbau, Suche aus Quelltext 2, Statistik aus Quelltext 4 und Änderungen aus Quelltext 3) in Sekunden an BaseX.

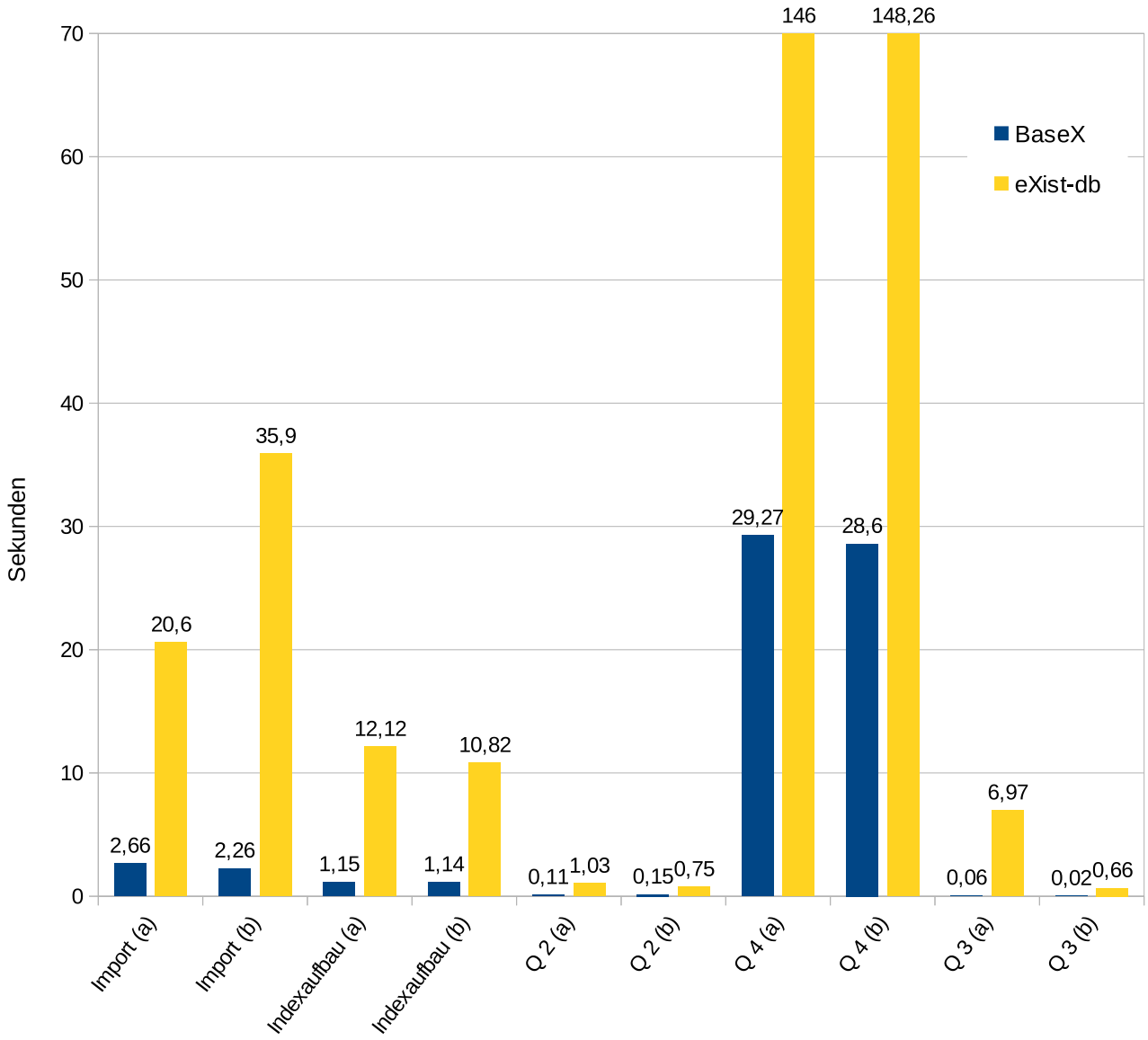

Abbildung 1: Das Diagramm zeigt die Zeit, die für die Ausführung der Abfragen Import, Indexeraufbau, Suche aus Quelltext 2, Statistik aus Quelltext 4 und Änderungen aus Quelltext 3 (a: Ausführung der Anfrage auf 10 identische Collections, b: Mehrmalige Ausführung auf die selbe Collection) benötigt wurde.

Lauf	Import[a]	Import[b]	Index[a]	Index[b]	Q 2[a]	Q 2[b]	Q 4[a]	Q 4[b]	Q 3[a]	Q 3[b]
1	19	21	12,215	12,562	1,687	1,537	146,67	148,33	8,793	6,438
2	21	30	12,155	11,144	0,680	0,632	144,74	147,33	7,952	0,019
3	20	36	12,447	10,451	0,710	0,628	145,25	145,72	6,428	0,022
4	19	27	12,881	10,842	1,453	0,636	146,71	149,59	6,858	0,025
5	21	38	12,621	10,309	0,711	0,758	145,51	148,06	6,793	0,021
6	21	46	11,265	10,842	0,697	0,689	145,11	146,76	6,728	0,024
7	21	34	11,757	10,748	1,459	0,685	145,64	146,68	6,400	0,021
8	20	41	11,781	10,643	0,691	0,622	146,03	147,17	6,665	0,025
9	21	38	11,882	10,390	1,539	0,657	145,36	156,26	6,461	0,019
10	23	48	12,233	10,25	0,719	0,656	149,02	146,69	6,617	0,021
Durchschnitt	21	36	12,123	10,818	1,034	0,750	146,00	148,26	6,970	0,664
Median	21	37	12,185	10,696	0,715	0,657	145,58	147,25	6,697	0,022

a Ausführung der Anfrage auf 10 identische Collections
b Mehrmalige Ausführung auf die selbe Collection

Tabelle 3: Die Tabelle zeigt die detaillierte Ausführungsdauer der Anfragen (Import, Indexeraufbau, Suche aus Quelltext 2, Statistik aus Quelltext 4 und Änderungen aus Quelltext 3) in Sekunden an eXist-db.

```
<ul>{
let $r := collection('/db/tagesschau')
 //speech_recognition[
 starts-with(./file/@name, 'TV-201208')]
for $word in distinct-values($r//word)
let $c := count($r//word[. = $word])
order by $c descending, $word
return <li>{$word} -- {$c}</li>
}</ul>
```

Quelltext 4: XQuery, der alle Begriffe und deren Häufigkeit innerhalb eines bestimmten Monats auflistet (hier August 2012)

hält dies für alle weiteren Durchläufe an, bei BaseX hingegen nur für ein bis zwei.

eXist-db hebt sich durch Versionierung, grundlegende Replikationsmöglichkeiten, Abdeckung der Anforderungen und nicht zuletzt durch eine gute Dokumentation von den anderen Produkten ab. BaseX optimiert die Anfragen vor der Ausführung und ist im Test stets performanter als eXist-db. Vor allem die Anfrage aus Quelltext 4 zeigt, dass die Optimierung der Anfragen, ob durch den Nutzer oder die Datenbank, sehr wichtig ist.

7 Zusammenfassung und Ausblick

Gegenüber eines RDBMS bieten XML-Datenbanken einen großen Gewinn an Flexibilität, da auch nicht exakt identisch strukturierte Daten in einer Collection abgelegt und gleichzeitig durchsucht werden können. XML-Datenbanken eignen sich gut zur Ablage von Metadaten. Um als Archivsystem auch die Multimedia-Daten selbst aufzunehmen, fehlen ihnen allerdings die nötige Funktionalitäten wie z.B. das Ausliefern von Teilstücke von Datei. Außerdem ist die Verwaltung von Berechtigungen, wenn vorhanden, oft auf das Mindeste beschränkt. eXist-db bietet ein gutes Gesamtpaket für erste Entwicklungen. Sie könnte eine gute Grundlage für Webanwendungen zur gemeinsamen Arbeit mit den gespeicherten Informationen sein.

Als Open-Source-Projekte könnten beide an spezielle Anforderungen angepasst werden. Denkbar ist vor allem die Ergänzung der fehlenden Funktionalitäten für Multimedia-Daten.

Die XML-Datenbank könnte sich aus Sicht des Clients auch hinter einer Abstraktionsschicht, einem „Content-Server“ verbergen. Diese Schicht spricht dann zur Recherche und Ablage von Meta-Informationen die Datenbank an, integriert zur Auslieferung der Audio- und Video-Daten aber weitere Komponenten. Wenn diese Schicht mehr Anforderungen übernimmt, ist BaseX eine schlankere und schnellere Alternative zu eXist-db.

Literatur

[1] *Apache Ant Homepage*. Englisch. 2013. URL: http://ant.apache.org/.

[2] *BaseX Homepage*. Englisch. Dez. 2012. URL: http://basex.org/.

[3] R.F. Chong, X. Wang und M. Dang. *Understanding DB2 learning visually with examples*. 2nd ed. IBM Press Pearson plc, 2008. ISBN: 9780768681772. URL: http://proquest.safaribooksonline.com/9780768681772.

[4] *documentation of BaseX*. Englisch. Dez. 2012. URL: http://docs.basex.org/wiki.

[5] *documentation of eXist-db (2.0.x branch)*. Englisch. eXist-db. 2012. URL: http://exist-db.org/exist/apps/doc/documentation.xml.

[6] *eXist-db Homepage*. Englisch. eXist-db. 2012. URL: http://exist-db.org/.

[7] *Homepage der Arbeitsgruppe Datenbanken und Informationssysteme*. Englisch. Arbeitsgruppe Datenbanken und Informationssysteme. 2013. URL: http://dbis.uni-konstanz.de/.

[8] *Homepage of monetDB*. Englisch. MonetDB. 2012. URL: http://www.monetdb.org.

[9] *IBM DB2 Version 10.1 Information Center*. IBM Corporation. 2012. URL: http://pic.dhe.ibm.com/infocenter/db2luw/v10r1/index.jsp.

[10] *Institute for System Programming Homepage*. Englisch. 2013. URL: http://www.ispras.ru/en/.

[11] *Introduction to Berkeley DB XML*. 2009. URL: http://docs.oracle.com/cd/E17276_01/html/intro_xml/BerkeleyDBXML-Intro.pdf.

[12] Georg Lausen. *Datenbanken: Grundlagen und XML-Technologien*. 1. Aufl. München: Elsevier Spektrum Akademischer Verlag, 2005, 281 S. ISBN: 3827414881. URL: http://swbplus.bsz-bw.de/bsz116707615cov.htm.

[13] Patrick Lehti. *Design and Implementation of a Data Manipulation Processor for an XML Query Language*. Aug. 2001. URL: http://www.lehti.de/beruf/diplomarbeit.pdf.

[14] *Oracle Berkeley DB Licensing Information*. Englisch. Oracle. 2013. URL: http://www.oracle.com/technetwork/products/berkeleydb/downloads/licensing-098979.html.

[15] *Oracle Berkeley DB Technology Network Product Page*. Englisch. Oracle. 2013. URL: http://www.oracle.com/technetwork/products/berkeleydb/overview/index.html.

[16] Dimitar Popov. *Advanced Storage Structures for Native XML Databases*. Aug. 2012. URL: http://files.basex.org/publications/Popov%20[2012],%20Advanced%20Storage%20Structures%20for%20Native%20XML%20Databases.pdf.

[17] *RFC4918 - HTTP Extensions for Web Distributed Authoring and Versioning (WebDAV)*. Englisch. IETF. Juni 2007. URL: http://tools.ietf.org/html/rfc4918.

[18] L. Richardson und S. Ruby. *RESTful Web Services*. O'Reilly, 2008. URL: http://books.google.de/books?id=XUaErakHsoAC.

[19] *Sedna Administration Guide*. Englisch. 2012. URL: http://www.sedna.org/adminguide/AdminGuide.html.

[20] *VirtualBox Homepage*. Oracle. 2013. URL: https://www.virtualbox.org/.

[21] *webMethods Tamino XML Server Datenblatt*. deutsch. Software AG. Nov. 2012. URL: http://www.softwareag.com/corporate/images/SAG_TaminoXML_FS_Nov12_Web_tcm16-71285.pdf.

[22] *XML Path Language (XPath) 2.0 (Second Edition)*. Englisch. W3C. Jan. 2011. URL: http://www.w3.org/TR/xpath20/.

[23] *XQuery 3.0: An XML Query Language: W3C Candidate Recommendation 08 January 2013*. Englisch. W3C. Jan. 2013. URL: http://www.w3.org/TR/xquery-30/.

[24] *XQuery API for Java Homepage*. Englisch. URL: http://xqj.net/.

[25] *XQuery Update Facility 1.0: W3C Recommendation 17 March 2011*. Englisch. W3C. März 2011. URL: http://www.w3.org/TR/xquery-update-10/.