

Liabolo - eine teuflisch gute
Literaturverwaltung
XML-Datenbanken zur Literaturverwaltung
– Version 1.0 –

Stefan Willer

13. September 2004

Inhaltsverzeichnis

1	Einführung	1
2	Was sind XML-Datenbanken	2
2.1	Datenbanken und XML	2
2.1.1	XML-fähige Datenbanken	2
2.1.2	Native XML-Datenbanken	2
2.2	Arten von Dokumenten	3
2.2.1	Daten-zentrierte Dokumente	3
2.2.2	Dokumenten-zentrierte Dokumente	3
3	State of the art	4
3.1	XMLDB:API	4
3.2	dbXML(1.0/2.0)	4
3.3	xindice	5
3.4	eXist	5
3.5	Vergleich	5
4	Warum XML-Datenbanken zur Literaturverwaltung	7
5	Liabolo - Eine teuflisch gute Literaturverwaltung	9
5.1	Einsatz von eXist	9
5.1.1	embedded vs. standalone	10
5.2	Anfragesprachen	11
A	Erklärung spezieller Dokumente	12
A.1	Standorte	12
A.2	Individuallisten	12
A.3	Medientypen	13
A.4	Branches	15

Kapitel 1

Einführung

Die folgenden Seiten sollen einen Einblick in die Verwendung von XML-Datenbanken geben. Nach einer Erklärung dieser wird auf die unterschiedlichen Dokumentarten eingegangen und darauf auf die Vorteile des Einsatzes einer XML-Datenbank bezogen. Es werden drei Implementierungen der XML:DB Schnittstelle vorgestellt und anhand einer Featurelist ein grober Vergleich gezogen. Abschliessend wird auf den Einsatz einer dieser Implementierungen in Bezug auf Liabolo eingegangen, wobei der interne Datenbank-Aufbau von Liabolo detailliert beschrieben wird.

Kapitel 2

Was sind XML-Datenbanken

2.1 Datenbanken und XML

Durch die Verwendung von XML entstanden i.A. zwei verschiedene Ansätze der Integration von XML-Dokumenten in bestehende Datenbanken. Die beiden Ansätze werden nun kurz vorgestellt.

2.1.1 XML-fähige Datenbanken

Im Laufe der Entwicklung wurde der Anteil an semi-strukturierten Dokumenten immer verbreiteter und es entstand der Wunsch nach adäquater Unterstützung dieser Daten in bestehenden Datenbanken. Somit musste zusätzliche Funktionalität in bisherigen Datenbanken integriert werden, um diese Dokumente zu verarbeiten. Eine Möglichkeit bestand in dem Mapping von XML-Dokumenten mittels einer DTD bzw. XMLSchema auf ein relationales Schema. Hiermit geschah eine Umwandlung, wobei die Daten schliesslich nicht mehr als XML vorlagen, eine mögliche Extraktion aber wiederum durch DTD's möglich ist. Ein weiterer Ansatz der Integration ist die Speicherung der XML-Dokumente als BLOB's bzw CLOB's, die in der Verwendung dann als Volltext durchsucht werden können. Diese Art der Datenbanken sind also in der Lage, XML-Daten zu verarbeiten, wobei dies eher ein evolutionärer Prozeß ist. Metadaten wie z.B. die Namen der Elemente werden hierbei in der Regel nicht berücksichtigt.

2.1.2 Native XML-Datenbanken

Unter nativen Datenbanken versteht man die Fähigkeit der ursprünglichen und unveränderten Speicherung und Rückgewinnung der XML-Dokumenten.

Es sind hierbei vier Arten der Speicherung zu unterscheiden:

1. XML-Dokumente werden vollständig als Text in der Datenbank gespeichert.
2. XML-Dokumente werden in ein effizientes Zwischenformat(pre-parsed) überführt und dann gespeichert.
3. Schaffung einer strukturellen Abbildung, wobei ein DOM¹ auf Tabellen oder Objekte einer Datenbank übertragen wird. „The Document Object Model is a platform- and language-neutral interface that will allow programs and scripts to dynamically access and update the content, structure and style of documents.“[W3C04]
4. Struktur von XML-Dokumente wird auf Datenbank abgebildet, wobei pro Element eine Tabelle entsteht. Metadaten gehen i.d.R. verloren.

2.2 Arten von Dokumenten

Auch die Art eines vorliegenden Dokumentes gibt Auskunft über die Integrationsmöglichkeiten. Stark strukturierte Dokumente sind einfacher auf Relationen zu übertragen als schwach strukturierte, die eher als Ganzes gespeichert werden.

2.2.1 Daten-zentrierte Dokumente

Daten-zentrierte Dokumente zeichnen sich durch hohe Regelmäßigkeit der Daten aus, welches für eine automatische Verarbeitung konzipiert sein kann. Somit wird das XML-Format häufig lediglich als reines Datenaustauschformat verwendet. Rechnungen und Statistiken wären hierfür ein Beispiel.

2.2.2 Dokumenten-zentrierte Dokumente

Dokumenten-zentrierte Dokumente legen, wie der Name es schon vermuten läßt mehr Wert auf das Dokument als Ganzes. Dies zeichnet sich u.a. durch eine strenge Unregelmäßigkeit der Daten aus. Typisch hierfür sind z.B. Dokumente, die in einem XML-Format vorliegen, jedoch XML z.B. zur Layoutbeschreibung verwenden(RTF, DocBook).

¹DOM: Document Object Model

Kapitel 3

State of the art

Im Umfang dieses Papers können leider nicht alle gängigen Datenbanken behandelt werden, so daß im Folgenden nur XML-Datenbanken betrachtet werden, die der XML:DB-API?? genügen. Angefangen mit der Vorstellung der XML:DB-API werden die Datenbanken dbXML v1(v2, Xindice und eXist im Anschluss angeführt.

3.1 XMLDB:API

Die API XML:DB stellt eine Schnittstelle für den Zugang von native XML-Datenbanken dar. Die Schnittstellen wurde in IDL(Interface Definition Language) beschrieben und ist somit prinzipiell für alle Sprachen offen. Eine Standard-Implementierung ist mit Java gegeben und wird in den folgenden drei XML-Datenbanken ebenfalls verwendet. Es sind in der Schnittstellenbeschreibung das Anlegen, Editieren, Löschen, Abfragen definiert. Somit werden alle Basisfunktionen zur Implementierung gefordert. Innerhalb der XML:DB Initiative wurde eine deskriptive Sprache XUpdate(www.xmldb.org/xupdate) definiert, die an XPATH(www.w3c.org/XPATH) angelehnt wurde. Wie der Name schon verlauten läßt, wird hiermit eine Update-Funktionalität beschrieben, auf die jedoch nicht näher eingegangen wird.

3.2 dbXML(1.0/2.0)

dbXMLv1.0[dbX03] wurde von der dbXML-Group entwickelt und später in das Apache-Xindice-Projekt überführt. Da dbXMLv2.0 zur Zeit im RC1 veröffentlicht wurde und eine komplette Neuentwicklung darstellt, sind relativ viele Quellen zu dbXMLv1.0 verschlossen. Als neue Features werden von

den Entwicklern ein Transaktionskonzept, XSLT-Transformationen, ein Sicherheitsmodell(u.a. sichere Verbindungen), etc. angegeben. Jedoch scheint der Release Candidate 1 laut Entwicklern noch nicht stable zu sein, so daß auch hier keine näheren Informationen erhältlich sind. Wie auch die Version 1.0 steht die Neuentwicklung 2.0 unter der GPL, wobei Version 2.0 ebenfalls kommerziell Lösungen anbietet, die den Support und weitere Funktionalität beinhaltet.

3.3 xindice

Xindice[Sta04a] (gesprochen 'zeen-dee-chay') entstand aus dbxmlv1.0 und wird zur Zeit vom Apache-XML-Project unter der GPL weiterentwickelt. Die aktuelle Version ist 1.1b4. Xindice zeichnet sich als Web-Applikation standardmäßig deployed unter Tomcat(Apache Group) aus. Dokumente werden mit Inizes versehen, um die Performance zu steigern. Weiterhin unterstützt sie zur Zeit kein Benutzermangement.

3.4 eXist

eXist[Mei04] steht un der aktuellen Version 1.0b1 unter der LGPL. Neben der Schnittstellenimplementierung gibt es die Möglichkeit eXist als Standalone, web-application und embedded-Version zu integrieren. Ebenfalls werden Dokumente mit Indizes versehen. Das Benutzermanagement ist unter eXist als XML:DB-Service integriert.

3.5 Vergleich

Der nun folgende Vergleich basiert im Wesentlichen auf verfügbaren Features der drei Alternativen. Ein Benchmark zu Xindice und eXist ist unter <http://exist-db.org/webdb.pdf> zu finden. Jedoch stammen die Daten von den Entwicklern von eXist. Hierin wird unter dem Abschnitt 'Performance und Skalierbarkeit' ein Vergleich der durchschnittlichen verbrauchten Zeit für verschiedene Suchanfragen gestellt. Es werden eXist, eXist mit Erweiterungen, Xindice und Jaxen gegenübergestellt. Anhand von acht zunehmend komplexer werdenden Anfragen treten gewisse Unregelmäßigkeiten auf, die Aufschluß über die Indizes-Vergabe erahnen lassen. Insgesamt geht hieraus hervor, daß eXist(ohne Erweiterungen) um ein drei- bis fünffaches schneller Ergebnisse liefert als Xindice, jedoch mit Ausnahmen. Das Ziel dieser Anaylse läßt jedoch verlauten, daß die Erweiterungen in eXist einen wesentlichen

Performance- wie auch Skalierbarkeitsvorteil mit sich bringen. In eXist wurden ein Reihe von XPATH-Erweiterungen implementiert, die eine effizientere Volltext-Suche incl. Schlüsselwörter gewährleistet.

Die folgende Tabelle (3.1) soll Features der Datenbanken, soweit von den Entwicklern publiziert gegenüberstellen. Eine anschließende Bewertung wird jedoch nicht explizit gegeben. Lediglich die Entscheidung der einzusetzenden Datenbank sei in Abschnitt (5.1) gegeben. Weiterhin stand zum Zeitpunkt der Entscheidung die Version dbXMLv2.0 noch nicht zur Verfügung.

	dbXML v2.0	Xindice v1.1	eXist v1.0
XML:DB API Impl.	ja	ja	ja
Einsatz	standalone, embedded	web- deployment	standalone, embedded, web- deployment
Transaktionen	ja	nein	nein
XSLT-Transformation	ja	nein	nein
Volltextsuche/ -indexierung	ja	nein	ja
commandline	ja	ja	nein
Sichere-Verbindungen	ja	nein	nein
Benutzer-management	ja	nein	ja
Skript-Erweiterungen	ja	nein	nein
Triggers	ja	nein	nein
Kommunikation	XML-RPC, REST[Fie02]	XML-RPC	REST,XML- RPC,SOAP, WebDav
GUI-Admin-Interface	ja	ja	ja
Backup-Funktionalität	nein	ja	ja
Anfrage-sprachen	XPATH, fulltext	XPATH	XPATH, XQUERY, fulltext
Update-sprachen	XUPDATE[Sta04b]	XUPDATE	XUPDATE
License	GPL/Commercial	ASL	LGPL

Tabelle 3.1: Feature-Vergleich

Kapitel 4

Warum XML-Datenbanken zur Literaturverwaltung

Bei dem Umgang mit XML-Datenbanken stellt sich allgemein die Frage, wofür diese besonders geeignet sind, bzw. wo Stärken und Schwächen dieser liegen. Warum wurden XML-Datenbanken, speziell native XML-Datenbanken, entworfen, wo es doch schon das Relationale Modell (theoretisch untermauert und bewährt) seit langem gibt? Die Antwort gibt im Wesentlichen die Eigenart von XML und die zunehmende Verbreitung dieser Technologie. XML-Dokumente sind semi-strukturiert, von Mensch und Maschine lesbar.

Durch ihre selbstbeschreibende Kraft ist dem Verfasser eine Möglichkeit gegeben, Restriktionen auf die Verwendung von XML-Sprachelementen zu beschränken. Es ist somit ein sehr freier Umgang vom Einsatz dieser gewährleistet. Während beim Relationalen Modell eine striktere Struktur durch vorgegebene Entities besteht und somit im Wesentlichen fest strukturierte Daten verarbeitet werden (vorgegebene DTD, die Mapping bestimmt), besteht im Umgang mit XML fast beliebige Freiheit.

Jetzt stellt sich die Frage, wenn wir von XML-Dokumenten ausgehen, welcher Aufwand erbracht werden muss, um XML-Daten auf das Relationale Modell zu übertragen. Gehen wir von daten-zentrierten Dokumenten aus, so ist der Aufwand in Grenzen zu halten, um ein Mapping zu schaffen. Komplizierter wird es bei dokumenten-zentrierten Dokumenten, welche in der Regel eine komplexere bzw. unstrukturierte Ordnung aufweisen. Elemente aus daten-zentrierten Dokumenten können bzw. werden meist als Entities ins Relationale Modell überführt, da sie gehäuft vorkommen und somit aus den Element-Attributen Entity-Attribute generiert werden.

KAPITEL 4. WARUM XML-DATENBANKEN ZUR LITERATURVERWALTUNG8

Die selbstbeschreibende Art, welche z.B. durch die Namensgebung der Elemente geschieht, kann kaum bei einer Überführung erhalten bleiben. Diese Metadaten, welche natürlich die Semantik eines Dokumentes mit bestimmen, werden daher leicht vernachlässigt.

Abschliessend ist daher der Einsatz von XML-Datenbanken zu empfehlen, wenn wenig strukturierte Daten vorliegen und die weiteren Rahmenbedingungen beider Alternativen in etwa übereinstimmen. XML-Datenbanken, die starke Restriktionen in der Größe des Dokumentes vorgeben, scheinen trotz guter Unterstützung unregelmäßig strukturierter Daten nicht die beste Wahl zu sein, wenn hauptsächlich diese Art an Dokumenten vorkommt. Es gibt also neben den besprochenen Aspekten noch andere, die bei der Wahl einer einzusetzenden Datenbank berücksichtigt werden sollten.

Kapitel 5

Liabolo - Eine teuflisch gute Literaturverwaltung

Dieses Kapitel soll den Einsatz der gewählten Datenbank in Liabolo verdeutlichen.

5.1 Einsatz von eXist

Bei der Analyse der Datenbanken wurde auf folgende Features der zur Verfügung stehenden Datenbanken gesetzt:

- XML:DB API Unterstützung, um Standards zu verwenden
- Möglichst schlanker Einsatz als embedded Version in Liabolo-Client
- Mehrbenutzerbetrieb incl. Benutzerverwaltung
- Volltextsuche und Indexierung

Gerade der Punkt 'embedded version' war ausschlaggebend für den Einsatz von eXist, da eine Datenbank-Instanz sowohl im Client als auch als Server entstehen sollte. Weiterhin wurde Wert auf ein bestehendes Benutzermanagement gelegt, was jedoch in zum Zeitpunkt Liabolo1.0 implementiert, jedoch nicht integriert wurde. Für die spätere Entwicklung von Liabolo, sollten statt Metadaten auch Dokumente integrierbar sein und effizient durchsucht werden, was eine indexierte Volltextsuche unabkömmlich werden ließ. Angesichts dieser Vorgaben entschied sich das Liabolo-Team zum Einsatz von eXist in der aktuellen Version 1.0b1.

5.1.1 embedded vs. standalone

Da der Client auch alleine als Nutzwerkzeug dienen sollte wurde eXist als embedded-Version integriert. Weiterhin entstand durch den Einsatz der gleichen Datenbank als Client und Server eine vereinfachte Kommunikation und Entwicklungsaufwand. Als angedachtes Feature-TODO steht eine Distribution als Java-WebStart-Anwendung aus, die somit die Paketgröße zusätzlich einschränkt.

Der interne Aufbau der Datenbank-Instanzen ähnelt sich auf Client- und Serverebene sehr stark. Der einzige Unterschied besteht in der zusätzliche Verwaltung von Individuallisten und der Umsetzung eines Replication Repositories¹ zur Replikation von globalen Daten in der Clientanwendung. Das Bild (5.1) verdeutlicht den internen Aufbau der Client-Anwendung.

Eine detailliertere Einsicht in die verwendeten Dokumente gibt der Anhang(A).

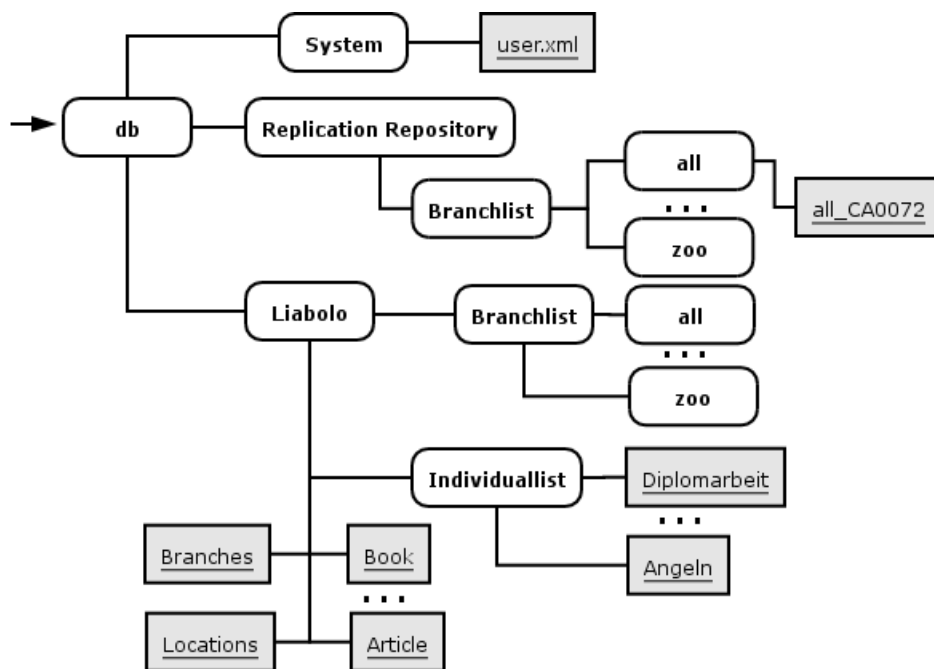


Abbildung 5.1: Interner Aufbau der Datenbank

¹Replication Repository: Teilbereich der logischen Trennung von lokalen und globalen Daten. Globale Daten werden als Kopien im Replication Repository abgelegt.

5.2 Anfragesprachen

eXist besitzt die Möglichkeit Anfragen als XPath/XSLT-Ausdrücke [W3C99a][W3C99b] und als XQUERY-Ausdrücke [?] zu verarbeiten. Zur Zeit ist in Liabolo1.0 eine einfache Suche integriert, die Metadatenätze nach Vorkommen eines gewünschten Patterns in einem angegebenen Metadaten-Property² durchsucht. Diese Suche wird über XPath-Ausdrücke realisiert. Im weiteren Verlauf der Entwicklung wird jedoch eine uneingeschränkte Suchfunktionalität über XQUERY für den erfahreneren Benutzer zusätzlich angeboten.

Zum Abschluss soll nun kurz die recht einfache Suche anhand eines Listings dargestellt werden:

Listing 5.1: Einfaches Suchstatement

```
1 /mediatype/PROPERTY[contains(lower-case(text()), '␣+␣
   pattern␣+␣"')]/parent::*
```

²Metadaten-Property: Als Property werden die einzelnen Dublin Core Einträge eines Metadatenatzes betrachtet

Anhang A

Erklärung spezieller Dokumente

Die folgenden Listings zeigen den Aufbau von Basisdokumente in Liabolo.

A.1 Standorte

Standorte werden in Liabolo getrennt verwaltet und in den jeweiligen Medientypen referenziert, wobei die Referenzierung manuell aufgelöst wird und somit nicht konsistenzerhaltend ist. Daher ist in Liabolo1.0 die Möglichkeit des Entfernens von Standorten unterdrückt.

Standorte werden durch einen Namen und zugehörige Beschreibung in Listing (A.1) spezifiziert.

Listing A.1: Beschreibung der Standorte

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <locations>
3     <location name="A4-203" desc="Sekretariat"/>
4     <location name="A2-101" desc="Büro"/>
5 </locations>
```

A.2 Individuallisten

Individuallisten ermöglichen dem Benutzer eine Übersicht und schnellen Zugriff auf thematisch eng verwandte Dokumente, wie z.B. die Literatur, die für eine Diplomarbeit verwendet wird. Listing (refst:indlistDesc) besitzt einen Namen mit zugehöriger Beschreibung und für jeden Medieneintrag eben-

falls neben dem Namen eine Beschreibung (z.B. wofür die Literatur eingesetzt wird).

Listing A.2: Aufbau einer Individualliste

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <individuallist name="XML-Datenbanken" desc="Sammlung_
   einiger_Bücher_rund_um_XML_und_Datenbanken">
3   <item id="inf_CA0071" desc="Gute_Einführung_in_XML-
     Datenbanken"/>
4   <item id="inf_AB0023" desc="Primärliteratur_für_
     Einführungskapitel"/>
5   <item id="inf_EB0213" desc="XPath-Tutorial"/>
6   <item id="inf_TB3021" desc="XQUERY-Tutorial"/>
7   <item id="inf_AZ0453" desc="eXist_Dokumentation"/>
8 </individuallist>
```

A.3 Medientypen

Für jeden Medientyp, wird in der Datenbank ein XML-Dokument mit dem Namen des Medientypen angelegt. Ein Medientyp enthält eine Anzahl an Metadaten, die aus der Menge aller verfügbaren Metadaten, die Liabolo bereitstellt ausgewählt werden kann. Jeder Metadaten-Eintrag verfügt über die Attribute

- name: Der benutzerdefinierte Name des Metadateneintrages
- desc: Die zugehörige Beschreibung
- DCid: Das Mapping auf einen Basis-Metadaten-Typ, welche von Liabolo vorgegeben werden. Damit soll die Möglichkeit offen gehalten werden, daß z.B. mehrere Autoren eines Buches angezeigt werden, sie sich jeddoch immer auf den Basistyp 'author' beziehen
- type: Gibt an, ob es sich um eine Textzeile, einen Textbereich, etc. handelt. Wird zur Darstellung in der Gui benötigt.

mit denen ein Eintrag näher spezifiziert werden kann.

Listing (A.3) zeigt die Struktur eines Buches auf.

Listing A.3: Aufbau einer Buchbeschreibung

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <mediaType>
3   <metadata name="creator" desc="creator_desc" DCid="0"
4     type="0"/>
5   <metadata name="title" desc="title_desc" DCid="1"
6     type="0"/>
7   <metadata name="publisher" desc="publisher_desc" DCid="
8     = "2" type="0"/>
9   <metadata name="date" desc="date_desc" DCid="3" type="
10    "3"/>
11  <metadata name="source" desc="source_desc" DCid="5"
12    type="0"/>
13  <metadata name="coverage" desc="coverage_desc" DCid="
14    6" type="2"/>
15  <metadata name="language" desc="language_desc" DCid="
16    7" type="0"/>
17  <metadata name="description" desc="description_desc"
18    DCid="8" type="1"/>
19  <metadata name="subject" desc="subject_desc" DCid="9"
20    type="0"/>
21  <metadata name="identifizier" desc="identifizier_desc"
22    DCid="10" type="0"/>
23  <metadata name="format" desc="format_desc" DCid="12"
24    type="0"/>
25  <metadata name="contributors" desc="contributors_desc
26    " DCid="13" type="0"/>
27 </mediaType>
```

A.4 Branches

Das Listing(A.4) zeigt alle verfügbaren branches mit deren Beschreibung auf.

Listing A.4: Beschreibung der verfügbaren Branches

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <branches>
3   <branch abbr="all" desc="Allgemeine_Nachschlagewerke"
4     />
5   <branch abbr="asl" desc="Allgemeine_Sparch_und_
6     Literaturwissenschaft"/>
7   <branch abbr="ang" desc="Anglistik"/>
8   <branch abbr="ast" desc="Astronomie"/>
9   <branch abbr="bcp" desc="Biochemie,_Biophysik"/>
10  <branch abbr="bio" desc="Biologie"/>
11  <branch abbr="bot" desc="Biologie_-Botanik"/>
12  <branch abbr="hbi" desc="Biologie_-Humanbiologie"/>
13  <branch abbr="zoo" desc="Biologie_-Zoologie"/>
14  <branch abbr="bub" desc="Buch_und_Bibliothekswesen"/
15    >
16  <branch abbr="che" desc="Chemie"/>
17  <branch abbr="elt" desc="Elektrotechnik"/>
18  <branch abbr="etn" desc="Ethnologie"/>
19  <branch abbr="geo" desc="Geowissenschaften"/>
20  <branch abbr="ger" desc="Germanistik"/>
21  <branch abbr="his" desc="Geschichte_der_Neuzeit"/>
22  <branch abbr="hit" desc="Geschichte_des_20._
23    Jahrhunderts"/>
24  <branch abbr="hil" desc="Geschichtswissenschaft"/>
```

Literaturverzeichnis

- [dbX03] dbXML. dbxml programmers guide. <http://www.dbxml.com/docs/programmer.html>, 2003. 4
- [Fie02] Roy T. Fielding. Representational state transfer. http://www.ics.uci.edu/~fielding/talks/webarch_9805/, 2002. 6
- [Mei04] Wolfgang Meier. exist documentation. <http://exist.sourceforge.net/documentation.html>, 2004. 5
- [Sta04a] Kimbro Staken. Apache xindice. <http://xml.apache.org/xindice/>, 2004. 5
- [Sta04b] Kimbro Staken. Xml:db xupdate use cases. <http://www.xmldatabases.org/projects/XUpdate-UseCases/>, 2004. 6
- [W3C99a] W3C. Xml-path language version 1.0. <http://www.w3.org/TR/xpath>, 1999. 11
- [W3C99b] W3C. Xsl-transformations version 1.0. <http://www.w3.org/TR/xslt>, 1999. 11
- [W3C04] W3C. Dom - document object model. <http://www.w3c.org/DOM/>, 2004. 3