

Dokumentation zum Semesterprojekt

XMIL-Datenbanken
A U F D E M P R Ü F S T A N D

Alexander Nägele, Matr.Nr. 215441, alex@societys-pet.de
Markus Korzendorfer, Matr.Nr. 215539, markus@dselect.de
Bruno Schliersmair, Matr.Nr. 215433, b.schliersmair@web.de
Tobias Walter, Matr.Nr. 215438, tobias@unwichtig.org
Joachim Kluge, Matr.Nr. 215190, fuwa@jokluge.mailshell.com

6. Juli 2005

Betreut durch Prof. Dr. Lothar Piepmeyer

Inhaltsverzeichnis

1	Analyse & Aufgabenstellung	2
1.1	Kompatibilitätstests	2
1.2	Performancetests	2
1.3	Usability	2
2	Projektorganisation	3
3	Kandidaten	4
3.1	Native XML-Datenbanken	4
3.1.1	Tamino	4
3.1.2	Galax	5
3.1.3	Sleepycat	5
3.2	Relationale Datenbanken mit XML-Funktionalität	5
3.2.1	IBM DB2 Native XML Alpha Program	5
3.2.2	Microsoft SQL Server 2005 BETA 2	6
3.2.3	Oracle Database 10g	6
3.3	Middleware-Lösungen	7
3.3.1	XML-DBMS	7
3.3.2	Castor	7
3.3.3	Allora	7
3.4	Fazit	8
4	Kandidaten für die Benchmarks	10
4.1	X007	10
4.2	XMach-1	10
4.3	Bumblebee	10
4.4	XMark	11
4.5	Fazit	11
5	Einrichten der Testumgebung	12
6	Datenbanken	13
6.1	Sleepycat	13
6.2	IBM DB2 Native XML Alpha Program	14

Inhaltsverzeichnis

7	Kompatibilitätstests	16
7.1	Die W3 Use Cases	16
7.2	IBM DB2 Nativ XML Alpha Program	17
7.2.1	Durchführung der Tests	17
7.2.2	Auswertung	17
7.2.3	Fazit	21
7.3	Sleepycat	21
7.3.1	Durchführung der Tests	21
7.3.2	Auswertung	21
7.4	Fazit	22
8	Performancetests	23
8.1	Testbedingungen	23
8.2	XMark	23
8.2.1	Vorbereitung von Hellgate	28
8.3	Ablauf der Performancetests	28
8.4	Test-Framework	29
8.5	Auswertung	31
8.5.1	DB2	31
8.5.2	Sleepycat	31
8.5.3	Ergebnisse	39
8.5.4	Fazit	49
9	XML-Datenbanken und grosse Datenmengen	50
9.1	Abfragen auf Sleepycat	50
9.2	Abfragen auf IBM DB2 Nativ XML Alpha Program	50
9.3	Abfragen auf Saxon	50
10	Fazit	51
A	W3 Use Cases	52
A.1	NS	52
A.1.1	Q1	52
A.1.2	Q2	52
A.1.3	Q3	52
A.1.4	Q4	52
A.1.5	Q5	52
A.1.6	Q6	53
A.1.7	Q7	53
A.1.8	Q8	53
A.2	PARTS	53
A.2.1	Q1	53
A.3	R	54
A.3.1	Q1	54

Inhaltsverzeichnis

A.3.2	Q2	54
A.3.3	Q3	54
A.3.4	Q4	55
A.3.5	Q5	55
A.3.6	Q6	55
A.3.7	Q7	55
A.3.8	Q8	56
A.3.9	Q9	56
A.3.10	Q10	56
A.3.11	Q11	56
A.3.12	Q12	57
A.3.13	Q13	57
A.3.14	Q14	57
A.3.15	Q15	57
A.3.16	Q16	58
A.3.17	Q17	58
A.3.18	Q18	58
A.4	SEQ	59
A.4.1	Q1	59
A.4.2	Q2	59
A.4.3	Q3	59
A.4.4	Q4	59
A.4.5	Q5a	59
A.4.6	Q5b	59
A.4.7	Q5c	60
A.5	SGML	60
A.5.1	Q1	60
A.5.2	Q2	60
A.5.3	Q3	60
A.5.4	Q4	61
A.5.5	Q5	61
A.5.6	Q6	61
A.5.7	Q7	61
A.5.8	Q8a	61
A.5.9	Q8b	61
A.5.10	Q9	61
A.5.11	Q10	62
A.6	STRING	62
A.6.1	Q1	62
A.6.2	Q2	62
A.6.3	Q4	62
A.6.4	Q5	63
A.7	TREE	63
A.7.1	Q1	63

Inhaltsverzeichnis

A.7.2	Q2	63
A.7.3	Q3	63
A.7.4	Q4	63
A.7.5	Q5	64
A.7.6	Q6	64
A.8	XMP	64
A.8.1	Q1	64
A.8.2	Q2	64
A.8.3	Q3	65
A.8.4	Q4	65
A.8.5	Q5	65
A.8.6	Q6	65
A.8.7	Q7	66
A.8.8	Q8	66
A.8.9	Q9	66
A.8.10	Q10	66
A.8.11	Q11	66
A.8.12	Q12	67
B	Quellcode	68
B.1	xmldb218	68
B.1.1	exampleLoadContainer.java	68
B.1.2	myDbEnv.java	71
B.1.3	XQuery_W3.java	72
B.2	Batch-Skripte DB2	75
B.2.1	NS.bat	75
B.2.2	PARTS.bat	75
B.2.3	R.bat	76
B.2.4	SEQ.bat	77
B.2.5	SGML.bat	77
B.2.6	STRING.bat	78
B.2.7	TREE.bat	78
B.2.8	XMP.bat	79
B.3	XMLBench	80
B.3.1	XMLBench.java	80
B.3.2	XMLGenerator.java	81
B.3.3	db.DB2Connector.java	81
B.3.4	db.SleepycatConnector.java	85
B.3.5	db.Sleepycat.exampleLoadContainer.java	88
B.3.6	db.Sleepycat.XQuery_FrameWork.java	90
B.3.7	results.ResultGenerator.java	92
B.3.8	xmark.XMarkGenerator.java	93
B.3.9	xmark.XMarkQueryObject.java	94
B.3.10	build.xml	96

Vorwort

Dieses Dokument entstand im Rahmen des Semesterprojektes „XML-Datenbanken auf dem Prüfstand“ der Fachhochschule Furtwangen im Sommersemester 2005. In den nachfolgenden Kapiteln wird auf die Anforderungen des Projektes, die konkrete Umsetzung dieser sowie auf die erreichten Ergebnisse eingegangen.

Zum Verständnis dieses Dokuments werden grundlegende Kenntnisse in XQuery und XPath vorausgesetzt. Tutorials zu den beiden Themen sind unter <http://www.w3schools.com> zu finden. Die genauen Spezifikationen sind unter <http://www.w3.org/TR/xquery/> und <http://www.w3.org/TR/xpath> erhältlich.

1 Analyse & Aufgabenstellung

Daten werden zunehmend im XML-Format abgelegt. Seit einiger Zeit ist es möglich, solche Dokumente in Datenbankmanagementsystemen abzulegen, sodass auch der Inhalt der Dokumente abgefragt werden kann. Hierfür existieren drei Ansätze:

- native XML-Datenbanken
- RDBMS mit entsprechenden XML-Erweiterungen
- Middleware, die das Marshalling bzw. Unmarshalling zwischen XML und SQL übernimmt

Zu jeder der oben genannten Ansätze gibt es eine Vielzahl von Implementierungen. Einige dieser Implementierungen sollen auf ihre Funktionalität und Leistungsfähigkeit untersucht und miteinander verglichen werden. Im Folgenden werden die einzelnen Testanforderungen näher betrachtet.

1.1 Kompatibilitätstests

Mit Hilfe der Kompatibilitätstests wird überprüft, inwiefern die einzelnen Implementierungen die XML-Abfragesprache XQuery unterstützen. Als Grundlage dazu dienen die vom W3C veröffentlichten „XML Query Use Cases“ vom April 2005.

1.2 Performancetests

Bei den Performancetests wird die Geschwindigkeit der Datenbank und des Interpreters getestet. Hierfür werden Zeitmessungen bezüglich des Imports von Daten und der Ausführzeit der Test-Queries vorgenommen, sodass die einzelnen Datenbanken ausgewertet und miteinander verglichen werden können.

1.3 Usability

Zusätzlich zu den Kompatibilitäts- und Performancetests wird die Benutzerfreundlichkeit der einzelnen Datenbanken näher betrachtet. Hierbei ist besonders auf den Ablauf der Installation, mitgelieferte Tools sowie auf die Bedienbarkeit zu achten.

2 Projektorganisation

Zur einheitlichen Pflege der Projektdaten wird ein Versionsverwaltungssystem eingesetzt, in diesem Fall Subversion. Dieses wird auf dem Testsystem installiert und konfiguriert, sodass eine zentrale Verwaltung von erstellten Dokumenten, Sitzungsprotokollen und Sourcecodes ermöglicht wird.

3 Kandidaten

Wie bereits erwähnt gibt es drei Ansätze, um XML-Daten in Datenbankmanagementsystemen abzulegen: native XML-Datenbanken, RDBMS mit entsprechenden XML-Erweiterungen und Middleware, die das Marshalling bzw. Unmarshalling zwischen XML und SQL übernimmt. In diesem Kapitel werden für jeden Ansatz mehrere Implementierungen näher betrachtet und ausgewertet.

3.1 Native XML-Datenbanken

Drei Native Datenbanken werden in diesem Kapitel näher betrachtet: Tamino von der Software-AG, die Open Source Datenbank Galax und Sleepycat.

3.1.1 Tamino

Tamino (<http://www.tamino.com>) ist eine ausgewachsene kommerzielle Lösung für Windows und Linux. Die Datenbank verfügt über umfangreiche Administrationsmöglichkeiten, basierend auf einer Weboberfläche. Zusätzlich bringt Tamino APIs für C, .Net und Java mit.

Installation Tamino wird als 30 Tage Testversion zum Download angeboten. Leider ist die erhältliche Version (4.1) über ein Jahr alt und lässt nur eine maximale Datenbankgrösse von 20 MB zu. Bei der Installation auf Hellgate trat ein nicht zu lösendes Problem durch Inkompatibilität des Installers mit dem Windows XP Service Pack 2 auf, welches auf anderen Systemen allerdings nicht repliziert werden konnte. Zusätzlich musste bei der Installation eine Systemdatei manuell umbenannt werden, da der Parser von Tamino Probleme mit Tabs aufweist. Beim Versuch, die mit Tamino's Schema Mapper erstellten Schemas in die Datenbank zu importieren, traten Fehler im Webinterface auf. Hier musste dann auf die Importfunktion des Schema Mappers zurückgegriffen werden, um den Import vorzunehmen.

XQuery-Unterstützung Die vorliegende Testversion von Tamino ist die erste, die XQuery unterstützen soll. Allerdings fehlen grundlegende Funktionen wie `order by`, `<<` oder `contains()` gänzlich. Der Versuch, eine aktuellere sowie unbeschränkte Version von Tamino zu erhalten, schlug leider fehl.

Abschliessend bleibt zu sagen, dass Tamino vom ersten Eindruck her ein an sich gutes Produkt darstellt, das aber in der Version 4.1 im XQuery Bereich unausgereift erscheint. Zusätzlich traten bei der Verwendung der Testversion häufig kleinere Fehler auf, die den Gesamteindruck trübten.

3.1.2 Galax

Galax ist eine Open-Source Implementierung von XQuery 1.0 und XPath 2.0, lauffähig unter Linux und Windows. Primär ist Galax ein XML-Interpreter, der Daten aus XML-Dateien liest und nach den Anweisungen der XQuery verarbeitet. Zusätzlich bietet Galax eine Schnittstelle zu Java und C, über die das Verarbeiten der XML-Daten möglich ist. Momentan unterstützt Galax nur das Verarbeiten von lokalen XML-Dateien, es wird jedoch daran gearbeitet XML-Streams anhand eines Storage-Managers namens „jungle“ über das Netzwerk zu verarbeiten. Der Storage-Manager ist in den Sourcefiles bereits vorhanden, in den Binaries jedoch noch nicht.

3.1.3 Sleepycat

Sleepycat (<http://www.sleepycat.com>), genauer Sleepycat DB XML, ist eine Open Source Lösung für Linux und Windows. Für den nicht-kommerziellen Einsatz kann die Datenbank frei verwendet werden. Näher betrachtet ist Sleepycat ein Aufsatz für die aus dem gleichen Haus stammende relationale Berkeley DB und erbt somit Eigenschaften wie bspw. ACID.

Sleepycat selbst ist kein Server, sondern nutzt lediglich jar- und dll-Dateien. Die mitgebrachte Shell eignet sich für die erste Einarbeitung in die Datenbank. Für den effizienteren Einsatz kann dann auf die vorhandene API, wahlweise C++ oder Java, zurückgegriffen werden.

Installation Die Installation von Sleepycat gestaltet sich als problemlos. Nach dem Download kann die Datenbank mit den Default-Einstellungen installiert werden.

XQuery-Unterstützung Der Hersteller proklamiert aktuellen XQuery Support, der sich nach kurzen Tests auch bestätigen ließ.

3.2 Relationale Datenbanken mit XML-Funktionalität

Bei den relationen Datenbanken mit XML-Extension werden IBM DB2 Native XML Alpha Program, der Microsoft SQL Server 2005 BETA 2 sowie Oracle Database 10g näher betrachtet.

3.2.1 IBM DB2 Native XML Alpha Program

Im Rahmen des IBM alphaWorks-Programmes konnte eine Version des IBM DB2 Native XML Alpha Program beschafft werden. Es handelt sich hierbei um eine modifizierte Version der IBM DB2 Version 9, die sich ebenfalls noch im Alphastadium befindet. XML-Daten werden hier in einem neuen Datentyp XML gespeichert der wie bspw. `Int` oder `Varchar` in normalen Tabellen verwendet wird. Die Datenbank bietet u.A. die folgende Funktionalität:

- Import & Export von XML-Daten

3 Kandidaten

- Indizierung von XML-Daten
- XML Schema Repository (XML Schema und DTDs)
- XQuery als Abfragesprache
- Komandozeilen Tool
- ODBC/JDBC-Treiber für Java
- ausserdem Support fuer .NET, C und COBOL

Wie aus dem oben genannten Funktionsumfang ersichtlich, erfüllt die Datenbank die Anforderungen.

Installation Momentan ist noch kein Installer vorhanden. Zuallererst wird die IBM DB2 9 Alpha installiert. Danach wird ihr Verzechnisinhalt gelöscht und das IBM DB2 Native XML Alpha Program in dieses Verzeichnis entpackt.

XQuery-Unterstützung XQuery wird mit einigen, von IBM beschriebenen Ausnahmen, unterstützt.

3.2.2 Microsoft SQL Server 2005 BETA 2

Im Rahmen des MSDN Academic Alliance-Programmes der FH Furtwangen konnte eine Version des Microsoft SQL Server 2005 BETA 2 beschafft werden. Die Datenbank führt einen neuen Datentyp ein, der - wie auch bei DB2 - XML-Dokumente direkt in Tabellen aufnehmen kann.

Installation Die „zeitaufwendige“ Installation geht einfach von der Hand und kann mit den Standardeinstellungen durchgeführt werden.

XQuery-Unterstützung Beim Testen der Datenbank wurde schnell ersichtlich, dass der XQuery-Ausdruck LET nicht unterstützt wird. Auf Nachfrage in einer Microsoft Newsgroup (privatenews.microsoft.com, Account name: privatenews\sql05, Password: feedback) stellte sich heraus, dass der Support von LET - falls überhaupt - frühestens in der nächsten Version eingeführt wird.

3.2.3 Oracle Database 10g

Über eine Oracle Technology Network Development License konnte eine Version der Oracle Database 10g bezogen werden. Die Datenbank bietet die Möglichkeit, XML-Dokumente in Character Large Object (CLOB) zu speichern oder XML-Views auf relationale Daten zu erstellen und umgekehrt.

Installation Die Installation gestaltet sich problemlos und einfach.

XQuery-Unterstützung Bis zum jetzigen Zeitpunkt ist kein XQuery-Support implementiert. Oracle Database 10g wird laut <http://www.oracle.com/technology/tech/xml/xquery/pdf/xquery10gr2v2.pdf> ab Release 2 XQuery unterstützen.

3.3 Middleware-Lösungen

Drei vorhandene Middleware-Lösungen werden hier näher betrachtet: Allora, eine kommerzielle Lösung von Hit Software sowie XML-DBMS und Castor, zwei Open Source Frameworks.

3.3.1 XML-DBMS

XML-DBMS (www.rpbouret.com/xmldbms/) ist eine Open Source Middleware-Lösung für die Übertragung von Daten zwischen XML Dokumenten und relationalen Datenbanken. Das Framework ist für Java und Perl erhältlich, wobei bei Java via JDBC auf die Datenbank zugegriffen wird. Folglich können alle relationalen Datenbanken, für die ein JDBC-Treiber existiert, verwendet werden. Ein XML-Dokument wird als Objektbaum behandelt und über bereitgestellte Mapping-Informationen auf eine Tabelle abgebildet. Diese Informationen werden entweder anhand einer DTD oder eines Mapping-Files definiert.

XQuery-Unterstützung Das Framework bietet keine Möglichkeiten, die in Tabellen gespeicherten Dokumente weder mit XQuery noch mit XPath abzufragen. Ausserdem sind keine Update- oder Delete-Funktionen implementiert. Soll also bspw. eine Anfrage abgesetzt werden, so ist dies nur durch einen direkten SELECT auf die Datenbank-Tabellen möglich.

3.3.2 Castor

Die Open Source Middleware-Lösung Castor (www.castor.org) dient ebenfalls der Übertragung von Daten zwischen XML Dokumenten und relationalen Datenbanken. Castor nutzt dazu JDO (Java Data Objects), um die Abbildung von XML auf Tabellen vorzunehmen, es findet also kein direktes XML-zu-SQL statt. Die Java-API bietet entsprechende Methoden für die XML-Objekte, d.h. die eigentlichen Datenbank-Zugriffe erfolgen im Hintergrund.

XQuery-Unterstützung Wie auch bereits XML-DBMS bietet Castor keine Möglichkeit, XQuery-Abfragen abzusetzen.

3.3.3 Allora

Die kommerzielle Middleware-Lösung Allora von Hit Software (www.hitsw.com) bringt eine grafische Oberfläche namens Allora-Mapper mit (s. Abbildung 3.1). Dieser erlaubt es die XML-Mapping-Informationen sowie die Datenbank-Struktur über eine GUI zu

3 Kandidaten

definieren. Abschliessend wird ein Mapping-File generiert, welches zum Marshalling bzw. Unmarshalling zwischen dem XML-Dokument und der Datenbank genutzt wird. Als

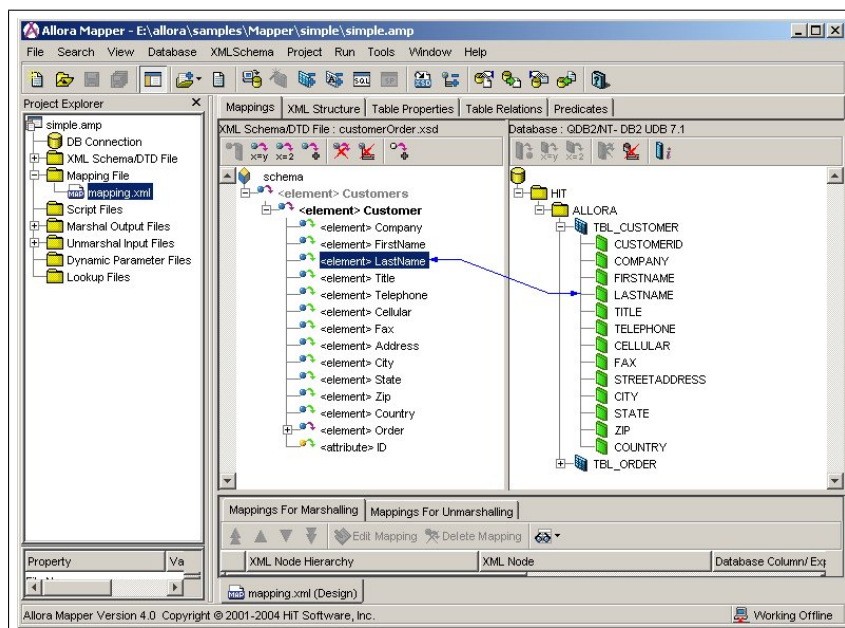


Abbildung 3.1: Allora-Mapper

relationale Datenbanken können diejenigen eingesetzt werden, für die ein JDBC-Treiber existiert. Zusätzlich zum Allora-Mapper bringt das Framework noch eine Java-API mit.

Installation Die Installation von Allora erfolgt problemlos. Anhand einer beantragten Testlizenz kann die Software 30 Tage frei verwendet werden.

XQuery-Unterstützung Mit dem Allora-Mapper wurde ein Mapping-File für ein einfaches XML-Dokument erstellt. Anhand dessen wurden die XML-Daten problemlos in die Datenbank importiert.

Die mitgelieferte Java-API bietet die Funktion `runXQuery()` zur Abfrage der Datenbank an. Irrtümlicherweise wird dabei nur XPath und kein XQuery unterstützt. Nach Rücksprache mit dem Support von Hit Software stellte sich heraus, dass bis jetzt noch kein XQuery-Support implementiert ist.

3.4 Fazit

Bei den nativen Datenbanken kommt Tamino nicht in die engere Auswahl, da die Anforderungen an XQuery nicht erfüllt werden. Bei der Auswahl der Datenbanken haben wir uns an Systeme gehalten, die ein eigenes Datenmanagement und somit auch Indizierung, etc. mitbringen. Leider kam die Erkenntnis, dass Galax diese Funktionalität inzwischen

3 Kandidaten

erfüllt, zu spät. Somit muss Galax ausgeschlossen werden. Ansonsten ist Galax ein sehr interessantes System, das man nicht aus dem Auge verlieren sollte. Für weitere Projekte sollte Galax auf jeden Fall in Betracht gezogen werden. Die Entscheidung für den Kandidaten im Bereich der nativen Datenbanken fällt somit auf Sleepycat, da man hier den besten XQuery-Support vorfindet.

Im Bereich der relationale Datenbanken mit XML-Funktionalität wird der Microsoft SQL Server 2005 BETA 2 aufgrund der fehlenden LET-Unterstützung ausgeschlossen. Oracle Database 10g kommt durch die fehlende XQuery-Implementierung auch nicht in Frage. Somit fällt die Entscheidung auf IBM DB2 Native XML Alpha Program, die hier den besten XQuery-Support bietet.

Für die Middleware-Lösungen mussten alle vier Kandidaten ausgeschlossen werden, da momentan keiner von ihnen XQuery-Support bietet.

4 Kandidaten für die Benchmarks

XML-Anwendungen verarbeiten in zunehmendem Maße sehr grosse XML-Dokumente. Dies stellt eine Herausforderung an die Effizienz eingesetzter Speicherungsverfahren und die Performanz der Anfrageverarbeitungen dar. Benchmarks dienen hierfür als objektives Mittel, um diese Kriterien zu testen. In diesem Kapitel werden die zur Verfügung stehenden Kandidaten näher betrachtet.

4.1 X007

X007 (<http://www.comp.nus.edu.sg/~ebh/XOO7.html>) ist als Einzelnutzer-Benchmark gedacht, bei dem die Arbeit mit technischen Dokumenten simuliert wird. Hierbei wird weniger Wert auf dokumentenzentrierte Anfragen gelegt. Bei X007 werden fertige Testdaten in den Grössen 4,2 8,4 und 12,8 MB mitgeliefert. Während der Ausführung des Benchmarks werden 8 X007-Anfragen und 19 weitere XML-spezifische Anfragen abgearbeitet.

4.2 XMach-1

XMach-1 (<http://dbs.uni-leipzig.de/en/projekte/XML/XmlBenchmarking.html>) - ein Java-Framework - ist ein Mehrbenutzer-Benchmark, der eine typische Web-Applikation unter der Nutzung von XML simuliert. Der Benchmark ist skalierbar und evaluiert ein gesamtes, verteiltes Informationssystem bestehend aus Client sowie Applikations- und Datenbank-Server. XMach-1 unterscheidet zwischen Server und Anwendungsseite - dadurch werden realistische Web-Applikationen nachgestellt. Die Kommunikation zwischen den Komponenten erfolgt dabei via HTTP. Die verwendeten Testdaten werden generiert und umfassen 10 KB, 100 KB, 1 MB oder 10 MB grosse Dokumente. Während der Ausführung werden 8 Anfrage- und 3 Änderungsoperationen abgearbeitet.

4.3 Bumblebee

Bumblebee (<http://www.xquery.com/bumblebee/>) ist eine Testsuite, die verschiedene XQuerys an XQuery-Engines schickt und deren Ergebnisse mit einem vorgegebenen Ergebnis vergleicht. Vorteilhaft ist, dass auch Negativergebnisse getestet werden können. So können Fehler provoziert werden und bei einer Negativantwort gilt der Test als bestanden. Der Benchmark eignet sich vor allem zum Test der W3C-Kompatibilität. Alle W3 Use Cases bis November 2003 werden mitgeliefert. Es ist ausserdem möglich, eigene Querys zu schreiben. Die Kommunikation mit den XQuery-Engines findet über eine

Java-Schnittstelle statt, wobei es für jede Engine einen Adapter gibt, der die Methoden `load()` und `evaluate()` beinhaltet. In diesen Methoden findet die Kommunikation mit dem XML-Interpreter statt. BumbleBee ist als 30-Tage Testversion verfügbar, wird jedoch für Entwickler oder zu Zwecken der Ausbildung auf Anfrage auch kostenlos zur Verfügung gestellt.

4.4 XMark

XMark (<http://monetdb.cwi.nl/xml/index.html>), ein Einzelnutzer-Benchmark, arbeitet auf einem lokalen System. Hierbei wird eine Internet-Auktion als Vertreter einer Web-Applikation modelliert. Anhand des mitgelieferten Tools `xmlgen` wird ein Testdokument von beliebiger Grösse generiert. Für das erstellte Testdokument steht ausserdem eine DTD-Datei zur Verfügung. Der Benchmark wird über 20 Queries durchgeführt, die XMark selbst mitbringt.

4.5 Fazit

Die Entscheidung für einen der Benchmarks hängt vom späteren Anwendungsgebiet der Datenbank ab. Für eine Datenbank im Web-Bereich fällt die Wahl auf XMach-1, da dieser, wie oben erwähnt, entsprechende XQueries bereitstellt. Somit fällt XMach-1 nicht in die engere Auswahl.

Bei Bumblebee wird ein selbst programmierter Adapter zur Datenbank-Verbindung vorausgesetzt. Da der nötige Zeitaufwand einen grossen Teil des Projektes in Anspruch nehmen würde, eignet sich Bumblebee nicht als Kandidat. Letztendlich fällt die Entscheidung auf XMark, da dieser - im Gegensatz zu X007 - dynamische Grössen der Testdokumente erlaubt. Zusätzlich bringt der Benchmark angemessene Queries mit, die sehr unterschiedliche Anforderungen an die Datenbank stellen.

5 Einrichten der Testumgebung

Im folgenden wird auf die Einrichtung des zur Verfügung stehenden Testsystems (hellgate.foo.fh-furtwangen.de) eingegangen. Für das Projekt steht das folgende System zur Verfügung:

- Dual Intel Xeon 2,8Ghz auf Intel E7505 Chipsatz
- 1,5 GB RAM
- Onboard Adaptec SCSI
- Seagate ST3360007LW SCSI 34GB
- IBM IC35L018UWD210-0 SCSI 17GB
- Elsa Erazor 2 PCI

Anfangs fiel die Entscheidung über das zu verwendende Betriebssystem auf Linux. Nach der ersten Installation eines Debian Linux wurde auf eine RedHat Distribution umgeschwenkt, da einige Testkandidaten nur als RPMs ausgeliefert werden. Schlussendlich wurde dann ein Windows XP Professional mit Service Pack 2 installiert, da IBM DB2 Nativ XML Alpha Program nur für Windows und AIX ausgeliefert wird.

6 Datenbanken

Das Kapitel beschreibt detaillierter die Datenbanken, die für die Tests verwendet werden.

6.1 Sleepycat

Container und Umgebungen Die XML-Daten werden in sogenannten Containern gespeichert. Dies ist eine einzelne Datei mit der Endung `.dbxml`. Eine Ebene höher ist die Umgebung angesiedelt. In ihr können sich mehrere Container befinden. Eine Umgebung besteht im Endeffekt aus einem Verzeichnis mit drei Dateien (`_db.*`) und enthält diverse Einstellungen (z.B. Logging und Caching), die Auslagerungsdatei des Caches und die Logs. Über Java oder C++ greift man dann direkt via proprietärer Funktionen auf den Container zu - JDBC oder Ähnliches wird nicht verwendet. In unserem Fall wird stets die mitgelieferte Java-API zur Kommunikation mit der Datenbank verwendet.

Datenbankerstellung und Insert Zum besseren Verständnis kann dazu das Beispiel `exampleLoadContainer.java` verwendet werden (s. Anhang XMLBench). Dieses erstellt eine Umgebung und lädt alle XML-Dateien im angegebenen Verzeichnis in den Container `ExampleData.dbxml`. Ein nachträgliches Insert von Dokumenten ist im Beispiel nicht realisiert, aber ohne Probleme möglich.

Update und Delete XQuery stellt diesbezüglich keine Funktionalität zur Verfügung. Sleepycat stellt deshalb umfangreiche proprietäre Funktionen bereit. Der Grundansatz ist, dass durch eine XQuery Abfrage der zu ändernde Kontext selektiert wird. Danach kann man beinahe beliebig Knoten, Attribute und Weiteres ändern sowie löschen. Allerdings wurden diese Funktionen im Verlaufe des Semesterprojektes nicht auf ihre Funktionalität getestet. Weiterhin kann ein einzelner Container oder eine ganze Umgebung einfach durch Betriebssystemfunktionen, z.B. `rm`, gelöscht werden.

Indizes Durch die Verwendung von Indizes können in verschiedenen Abfragesituationen enorme Geschwindigkeitsvorteile erreicht werden. Durch die Struktur von XML ist dazu aber einige Arbeit zur Erstellung von Indizes notwendig. Bei Sleepycat werden dafür umfangreiche Funktionen zur Verfügung gestellt, z.B. `container.addIndex("auto/farbe", "", "(unique)-edge-element-equality-string")` Das 1. Argument beschreibt den Knoten bzw. die Kante. Gibt man nur „farbe“ an, würden alle Knoten vom Typ Farbe indiziert werden. Mit dem 2. Argument wird der Namespace angegeben, der in diesem Beispiel keine Verwendung findet. Das 3. und letzte Argument gibt die Strategie an und ist wiederum in 5 Elemente untergliedert.

- 1. Strategieelement = unique. Optionales Element, das aber angegeben werden sollte, falls der Inhalt jeder der passenden Knoten oder Kanten jeweils einzigartig ist.
- 2. Strategieelement = node—edge. Definiert Knoten oder Kante, je nachdem wie selektiv indiziert werden soll.
- 3. Strategieelement = element—attribute. Je nachdem ob Farbe ein Element oder Attribut ist.
- 4. Strategieelement = presence—equality—substring. Vom Aufbau der Query abhängig.
- 5. Strategieelement = Syntax. Bezeichnet den Typ der zu indizierenden Daten. Für den Schlüsseltyp presence ist dieser none, für substring logischerweise string und für equality stehen mehrere zur Auswahl.

Weiterführende Informationen zur Indexverwendung allgemein und zur Implementation der Java API bietet [\\$dbxml-Verz./dbxml/docs/java/com/Sleepycat/dbxml/XmlIndex-Specification.html#addIndex\(java.lang.String, java.lang.String, java.lang.String\)](#)

Indizes und Probleme In der Ausgabe 02/2005 des „XML und Webservices Magazins“ beschreibt Michael Seeman einige Problemszenarien. Da sich seine Versuche auf Sleepycat 2.0.x stützten, wurden seine Beispiele mit Sleepycat 2.1.8 nachgeprüft.

1. Absturz beim Zugriff auf 30MB großen Container ohne Index.
Die Beispiele mit und ohne Index laufen ohne Probleme durch; der Speicherverbrauch ist dabei relativ gering.
2. Die Ergebnisse sind je nach Verwendung eines Indexes unterschiedlich.
Die Fehler bestehen in beiden Beispielen immer noch.
3. Bei der Verwendung von count() wird der Index nicht genutzt.
Die Ausführung des Beispiels dauert immer noch sehr lang. Das Herausnehmen der count()-Funktion bringt einen extremen Geschwindigkeitszuwachs.

Eine Detailvorstellung der Probleme bietet ein Artikel von Michael Seemann im XML und Webservices Magazin 02/2005 S 68ff. Hier ist noch anzumerken, dass versucht wurde, die Probleme mit DB2 zu reproduzieren. Diese traten bei der DB2 jedoch nicht auf.

6.2 IBM DB2 Native XML Alpha Program

Installation Wie bereits erwähnt, ist momentan noch kein Installer vorhanden. Zuerst wird die IBM DB2 9 Alpha installiert. Danach wird ihr Verzeichnisinhalt gelöscht und das IBM DB2 Nativ XML Alpha Program in dieses Verzeichnis entpackt.

Datenbanken erstellen Die Datenbank muss unicode sein und sollte mit folgendem Kommando erzeugt werden:

```
CREATE DATABASE <dbname> USING CODESET UTF-8 TERRITORY US  
Nun muss eine Tabelle erzeugt werden, die XML aufnehmen kann. Dazu wird einer Spalte der Datentyp XML zugewiesen, z.B. mit  
CREATE TABLE W3C (DATA XML)
```

Insert Für den Insert gibt es 2 Möglichkeiten:

1. **Import mit dem DB2 Importtool** Folgender Befehl fügt alle in import.del angegebenen Dateien in die Tabelle W3C ein. `...FROM . MODIFIED...` gibt an, dass die Dateien im Arbeitsverzeichnis liegen, von wo aus der Befehl ausgeführt wird.
`IMPORT FROM import.del OF DEL XML FROM . MODIFIED BY XMLCHAR INSERT INTO W3C`
import.del sieht dabei wie folgt aus: `<XDS FIL = 'data.xml' OFF='0' LEN = '123' />`
FIL gibt die Datei an, die eingefügt werden soll. OFF ist der Offset in Byte, ab dem angefangen werden soll zu lesen. LEN gibt an, bis zu welchem Byte gelesen werden soll.
2. **Import via Insert** `INSERT INTO W3C (DATA) VALUES (XMLPARSE (DOKUMENT '<root>...</root>' PRESERVE WHITESPACE))` In obigem Beispiel wird das Dokument `<root>...</root>` in die Spalte DATA der Tabelle W3C eingefügt. Diese Methode eignet sich gut für Prepared Statements in Java.

Update In DB2 gibt es momentan noch keine Möglichkeit einzelne Teile eines XML-Dokumentes zu ändern. Die einzige Möglichkeit besteht darin, das ganze Dokument analog zum Insert mit einem Update zu ersetzen.

Delete Das Löschen der Daten in einer XML Tabelle geschieht mittels `DELETE FROM W3C`. Dabei wird die ganze Tabelle gelöscht. Bei grossen Datenmengen ist dies durch einen Bug in DB2 sehr langsam. Daher empfiehlt es sich die Tabelle zu droppen und neu zu erzeugen.

Indizes `CREATE INDEX index01 on W3C(DATA) GENERATE KEY USING XMLPATTERN '/report/section/section.title' AS SQL VARCHAR(64)` Mit dem obigen Statement wird ein Index mit dem Namen index01 auf die Spalte data der Tabelle W3C erzeugt. Zur Indizierung wird das Element, welches über den XPath-Ausdruck `/report/section/section.title` angegeben ist, verwendet. Neben VARCHAR gibt es noch den Datentyp DOUBLE für Zahlen.

7 Kompatibilitätstests

Das Kapitel geht auf die W3 Use Cases ein, die für die Kompatibilitätstests verwendet werden. Des Weiteren werden die durchgeführten Tests auf IBM DB2 Nativ XML Alpha Program und Sleepycat betrachtet. Abschliessend werden die Ergebnisse verglichen.

7.1 Die W3 Use Cases

Die W3 Use Cases (<http://www.w3.org/TR/2005/WD-xquery-use-cases-20050404/>) sollen sicherstellen, dass der XQuery 1.0 Standard unterstützt wird. In den 70 Querys sind alle Anforderungen an den XQuery-Interpreter enthalten. Verhält sich dieser standardkonform, dann müssen alle Querys ein vorgegebenes Ergebnis liefern. Falls eine Datenbank eine spezielle Query nicht unterstützt, dann bedeutet das nicht zwangsläufig, dass diese Funktionalität nicht gegeben ist, sondern evtl. nur die Syntax vom Standard abweicht.

Die Use Cases sind in verschiedene Gruppen unterteilt, um sie nach Anforderungen an den Interpreter zu untergliedern. Im Folgenden werden die Kategorien und deren Anforderungen beschrieben. Die vollständigen Queries befinden sich im Anhang dieses Dokuments.

XMP - Experiences and Exemplars Im XMP Use Case finden sich allgemeine Abfragen, die bei der Datenbank anfallen. Diese beinhalten die Suche nach Elementen mit bestimmten Werten, die in der WHERE Klausel abgefragt werden. Hinzu kommt das Zusammenführen mehrerer Elementinhalte in ein Element.

TREE - Queries that preserve hierarchy Viele XML-Dokumente haben eine sehr flexible Struktur, in der Elemente Text und wiederum Elemente, die optional oder erforderlich sind, beinhalten. Die Tests untersuchen, ob der Interpreter mit dem unterschiedlichen Aufbau der einzelnen Elemente zurecht kommt.

SEQ - Queries based on Sequence Dieser Use Case konzentriert sich auf die Reihenfolge der Elemente. Dabei wird das n-te Element bearbeitet bzw. ausgelesen.

R - Access to Relational Data Der Use Case R gehört zu einem der interessantesten Use Cases. Hier wird die Datenstruktur einer relationalen Datenbank nachgebildet. Diese Queries stellen SQL-Querys nach und liefern dieselben Ergebnisse, die auch eine relationale Datenbank liefern würde. Somit wird sichergestellt, dass SQL durch XQuery ersetzt werden könnte.

SGML - Standard Generalized Markup Language Der Use Case SGML verdeutlicht die Flexibilität von XQuery. Hier wird als Datenbestand ein SGML Dokument, welches in XML konvertiert ist, herangezogen. Gleiche Abfragen auf ein DocBook-Dokument wären möglich. Diese Queries zeigen, dass XQuery nicht auf ein festes Repository angewiesen ist, sondern sehr flexibel eingesetzt werden kann.

STRING - String Search Der STRING Use Case dient bspw. zur Abfrage „news-documents“ . Dabei wird geprüft, ob der Umgang mit sehr langen Strings fehlerfrei funktioniert. Als Anwendungsbeispiel könnte hier das Archiv eines Zeitungsverlages genannt werden.

NS - Queries Using Namespaces Dieser Use Case testet den Umgang mit „namespace-qualified names“ .

PARTS - Recursive Parts Explosion PARTS findet seine Anwendung bei Dokumenten, die eine starke Fragmentierung der Daten beinhalten. Dabei wird die Datenbank in mehrere Teile aufgespalten.

Abschliessend sollte kurz angemerkt werden, dass sich Bumblebee hier als Testumgebung bestens eignen würde. Im Laufe der Einarbeitung in die Datenbanken sind jedoch Skripte mit derselben Funktionalität entstanden. Um weiteren Aufwand mit der Adapterprogrammierung zu vermeiden, wurden diese Skripte für die Tests herangezogen.

7.2 IBM DB2 Nativ XML Alpha Program

Die Kompatibilitätstests unter DB2 wurden anhand von Batch-Skripten durchgeführt. Diese finden sich im Anhang dieses Dokuments.

7.2.1 Durchführung der Tests

Für jede Use Case Gruppe existiert eine Batch-Datei, die die Datenbank sowie die Tabellen erzeugt und die Daten dann lädt. Danach werden die Queries ausgeführt und deren Ergebnisse in den Dateien *.res abgelegt. Die Dateien *.exp beinhalten die erwarteten Ergebnisse. Sind die Queries durchgelaufen, wird die Datenbank wieder gelöscht. Die Batchdatei muss im DB2 Command Window ausgeführt werden.

7.2.2 Auswertung

Die folgenden Tabellen zeigen die Ergebnisse der Tests auf DB2 mit den verschiedenen Use Cases.

Die Funktionen month-from-date, year-from-date und declare function werden von DB2 noch nicht unterstützt. Bei den meisten fehlerhaften Queries trat folgende Fehlermeldung auf:

```
SQL0303N A value cannot be assigned to a host variable in the SELECT, VALUES, or FETCH statement because the data types are not compatible. SQLSTATE=42806
```

7 Kompatibilitätstests

R Query	Test-Ergebnis
1	Ok
2	Error (data types are not compatible)
3	Ok
4	Ok
5	Ok
6	Error (data types are not compatible)
7	Error (data types are not compatible)
8	Ok
9	Error (month-from-date, year-from-date)
10	Ok
11	Error (data types are not compatible)
12	Error (declare function)
13	Error (data types are not compatible)
14	Error (data types are not compatible)
15	Ok
16	Ok
17	Ok
18	Ok

Tabelle 7.1: DB2-Ergebnisse des R Use Case

SEQ Query	Test-Ergebnis
1	Ok
2	Läuft, aber falsches Ergebnis
3	Ok
4	Ok
5a	Error (declare function)
5b	Läuft, aber falsches Ergebnis
5c	Error (declare function)

Tabelle 7.2: DB2-Ergebnisse des SEQ Use Case

7 Kompatibilitätstests

TREE Query	Test-Ergebnis
1	Error (declare function)
2	Ok
3	Ok
4	Ok
5	Ok
6	Error (declare function)

Tabelle 7.3: DB2-Ergebnisse des TREE Use Case

XMP Query	Test-Ergebnis
1	Ok
2	Ok
3	Ok
4	Ok
5	Ok
6	Ok
7	Ok
8	Ok
9	Ok
10	Error (data types are not compatible)
11	Ok
12	Ok

Tabelle 7.4: DB2-Ergebnisse des XMP Use Case

PARTS Query	Test-Ergebnis
1	Error (declare function)

Tabelle 7.5: DB2-Ergebnisse des PARTS Use Case

7 Kompatibilitätstests

NS Query	Test-Ergebnis
1	Ok
2	Ok
3	Ok
4	Ok
5	Ok
6	Ok
7	Läuft, aber falsches Ergebnis
8	Ok

Tabelle 7.6: DB2-Ergebnisse des NS Use Case

SGML Query	Test-Ergebnis
1	Ok
2	Ok
3	Ok
4	Läuft, aber falsches Ergebnis
5	Ok
6	Ok
7	Ok
8a	Ok
8b	Ok
9	Ok
10	Ok

Tabelle 7.7: DB2-Ergebnisse des SGML Use Case

STRING Query	Test-Ergebnis
1	Ok, allerdings werden Leerzeichen eingebaut
2	Error (declare function)
3	wurde aus den Ucs entfernt
4	Error (declare function)
5	Ok, allerdings werden Leerzeichen eingebaut

Tabelle 7.8: DB2-Ergebnisse des STRING Use Case

Dieser Fehler tritt immer dann auf, wenn in einer Bedingung eines XPath-Ausdrucks eine vorher deklarierte Variable verwendet wird, z.B. `//bid_tuple[itemno = $i/itemno]`. Bei den Queries mit falschen Ergebnissen besteht das Problem, dass die Elemente direkt über einen Index adressiert werden, z.B. `//incision[1]`.

7.2.3 Fazit

DB2 unterstützt einen Grossteil des XQuery-Sprachumfangs. Alle weiteren Fehler, die nicht auf fehlende Funktionen zurückzuführen sind oder falsche Ergebnisse liefern, scheinen die gleichen Ursachen zu haben.

7.3 Sleepycat

Die Kompatibilitätstests mit Sleepycat werden anhand eines Beispielprojektes unter Eclipse durchgeführt.

7.3.1 Durchführung der Tests

Nach dem Start von Eclipse muss das Projekt dbxml-218 importiert werden. In den Projekteigenschaften werden alle externen JAR-Dateien bis auf die JRE System Library gelöscht. Anschliessend müssen alle JAR-Dateien aus dem \$dbxml-Verzeichnis/jar/ importiert werden.

Im zweiten Schritt wird der Container erstellt. Hierbei ist zu beachten, dass XML-Dateien keine Referenz zu DTDs aufweisen dürfen. Aus diesem Grund müssen alle `<!DOCTYPE` Zeilen in den Dokumenten gelöscht werden. Ausserdem werden keine XML-Dateien aus Unterverzeichnissen eingefügt. Folglich sollte in jedem Use Case ein Verzeichnis, z.B. dbEnv, zu erstellen und als Umgebung genutzt werden.

Nun kann ExampleLoadContainer mit den dazugehörigen Argumenten ausgeführt werden. Diese werden im Usage-Hinweis beschrieben.

Im letzten Schritt werden die Queries vorbereitet und ausgeführt. Die Queries müssen die Endung `.query` haben. Dann können die Tests anhand der Klasse `XQuery_W3` gestartet werden. Beim Durchlauf des Tests werden für jede Query `.res`-Dateien erzeugt, die mit den erwarteten Ergebnissen verglichen werden können.

7.3.2 Auswertung

Sleepycat bietet eine außerordentlich gute Kompatibilität zu den W3C Use Cases. Bis auf den NS Use Case laufen die Tests fehlerfrei durch. Im NS Use Case werden einige namespace-Attribute nicht geliefert, der Inhalt selbst ist korrekt (s. Tabelle).

Hier ein Beispiel, um das Problem der fehlenden `xmlns`-Attribute zu verdeutlichen:

expected:

```
< Q2>
```

```
< title xmlns="http://www.example.org/music/records"
```

7 Kompatibilitätstests

NS Query	Test-Ergebnis
1	Ok
2	FALSCH, einige xmlns-Attribute fehlen
3	FALSCH, einige xmlns-Attribute fehlen
4	Ok
5	FALSCH, einige xmlns-Attribute fehlen
6	FALSCH, einige xmlns-Attribute fehlen
7	Ok, in expected scheinbar fehlende Leerzeichen vor xlink: xmlns:ma="http://www.example.com/AuctionWatch "xmlns:type="simple""
8	FALSCH, einige xmlns-Attribute fehlen

Tabelle 7.9: Sleepycat-Ergebnisse des NS Use Case

```
xmlns:anyzone="http://www.example.com/auctioneers#anyzone"  
xmlns:eachbay="http://www.example.com/auctioneers#eachbay"  
xmlns:ma="http://www.example.com/AuctionWatch"  
xmlns:xlink="http://www.w3.org/1999/xlink"  
xmlns:yabadoo="http://www.example.com/auctioneers#yabadoo"> In a Silent Way<  
/title>  
< /Q2>
```

```
result:  
< Q2>  
< title xmlns="http://www.example.org/music/records"> In a Silent Way< /title>  
< /Q2>
```

7.4 Fazit

Die Tests zeigen, dass DB2 gegenüber Sleepycat noch Nachholbedarf hat. Es muss allerdings angemerkt werden, dass sich die DB2 noch im Alpha-Stadium befindet und diese Fehler bis zum Final Release vermutlich behoben sind.

8 Performancetests

Ziel der Performancetest ist es, einen Eindruck zu bekommen, wie stark die Leistungsfähigkeit der verschiedenen Systeme voneinander abweichen und wie die einzelnen Anforderungen der Querys im Vergleich zu anderen Datenbanken erfüllt werden. Beim Betrachten der einzelnen Systeme ist es interessant, welche Queries die meisten Ressourcen benötigen und somit die größten Anforderungen an die XML-Datenbanken stellen. Im zweiten Schritt werden die Datenbanken auf die verschiedenen Anfragen optimiert. Dies wird mittels Indizierung erreicht. Beim Vergleich des normalen Datenbestandes und der optimierten Form können Schlüsse über die Effektivität von solchen Maßnahmen gezogen werden. Um diese Messgrößen zu erhalten, ist es notwendig, alle Datenbanken zwei mal zu testen.

8.1 Testbedingungen

Die Datenbanken werden mit den vom Testframework generierten Dateien gefüllt. Um Unterschiede bei verschiedenen Datenmengen herausstellen zu können, werden 2 verschiedene Datenbestände verwendet. Diese unterscheiden sich nur in der Menge der Daten. Hierbei werden folgende Dateigrößen verwendet:

- 1MB
- 10MB

Diese zwei Tests werden von jeder Datenbank durchlaufen. Nach dem ersten Durchlauf wird die Datenbank nach gegebenen Möglichkeiten optimiert und der Test erneut durchgeführt. Dies ermöglicht die Gegenüberstellung der optimierten zu den nicht optimierten Testergebnissen. Bei diesen Tests dürfen die Queries der Datenbank entsprechend angepasst werden, jedoch nicht das Resultat verfälschen. Diese Maßnahme dient nur dazu, nicht standardkonforme Implementierungen von Xquery zu unterstützen. Die Unterstützung der Standards wird separat mit den W3C UseCases getestet.

8.2 XMark

Der XMark Benchmark wird in Form eines XML-Generators und 20 Queries geliefert. Mit dem Generator lassen sich beliebig große XML-Dokumente erstellen, deren Inhalt aus den fiktiven Daten eines Auktionshauses besteht. Desweiteren werden DTDs und XML-Schemas mitgeliefert. Sie können zum Mapping oder für Indizierung verwendet werden. Auf die Bewertung nimmt die Verwendung der Schemafiles keinen Einfluss. Der

Kategorie	Query	Beschreibung
Exact Match	Q1	Diese Query testet die Fähigkeit des Datenbanksystems eine einfache String Suche angegebenem Pfad zu bearbeiten. Diese simple Query hat ein bestimmtes Hauptziel. Mit dem Resultat dieser Query soll nämlich die Performance Basis für Bewertung der weiteren Konzepte gebildet werden.
Ordered Access	Q2-Q4	Die Ordnung ist eines der wichtigsten Features in XML. Die Auswertung der Queries gibt einen Einblick darüber, wie das Datenbanksystem mit der Ordnung der XML Dokumente zurecht kommt und wie effizient es mit Queries mit Ordnungsbedingungen umgeht. Der Query Optimizer sollte/muss den Aspekt der Ordnung beachten und wenn Ordnung keine Rolle spielt, diese vernachlässigen können.
Casting	Q5	Zum Thema Casting, also Typumwandlung, muss man anmerken, dass String der generische Datentyp in XML Dokumenten ist. How many sold items cost more than 40 ? Was vom Datenbanksystem bei der Querybearbeitung gefordert wird, ist ein Preisvergleich bzw. eine Suche nach Preisen grösser als 40. Da die Preise im String Format vorliegen, müssen z.B. in den int Typen umgewandelt werden, um einen Vergleich möglich zu machen. Diese Query testet also, wie effizient eine Typumwandlung ausgeführt wird.
Regular Path Expressions	Q6-Q7	Regular Path Expressions sind fundamental für fast alle Query Languages für XML Daten. Durch Queries aus diesem Konzept sieht man, wie der Query Prozessor Path Expressions optimiert und irrelevante Teilbäume ausser Acht lässt.

Benchmark besteht darin, die benötigte Zeit für die einzelnen Querys zu messen. Die Tests können dabei auf beliebig große Testfiles angewendet werden. Eine Einstufung der Resultate gibt es bislang nicht. Es liegt im Ermessen des Betrachters die benötigte Zeit zu bewerten. Dazu werden alle Ergebnisse in einer Tabelle miteinander verglichen. Zum Messen der Zeiten werden absichtlich nur XQueries verwendet, um Kompatibilität mit möglichst vielen Systemen zu erreichen. Des Weiteren wurde auf die Verwendung von Namespaces verzichtet, da XML-Datenbanken hier noch Schwachstellen aufweisen.

Die Querys sind ebenfalls sinngemäß denen eines Auktionshauses nachempfunden. Dies dient allerdings nur dem besseren Verständnis der Query. Zum Benchmarken der Datenbanken sind die Querys auf sehr spezielle Aspekte des XML- Processings optimiert. Die Querys werden in der abgebildeten Tabelle dargestellt.

Kategorie	Query	Beschreibung
Chasing References	Q8-Q9	Referenzen sind ein integraler Teil von XML, mit denen feinere Beziehungen als eine reine Hierarchie aufgebaut werden können. Was die Queries aus diesem Konzept erfordern, ist ein XML Baum nicht nur in die Tiefe zu traversieren, sondern auch entlang der Referenzen in der Horizontale. „Print the keywords in emphasis in annotations of closed auctions. Return the IDs of the sellers of those auctions that have on or more keywords in emphasis“ . Zieht man Bild 1 zur Hilfe sieht man, dass die erste Query also Verbindungen zwischen persons und closed-auction (Referenz zwischen persons und closed-auction) beachten muss. Die zweite Query geht noch weiter, indem sie zusätzlich zu den Verbindungen aus der ersten Query noch Verbindungen zu items beachten muss. Durch Referenzen in einer XML Datenbank ergeben sich zusätzlich zu bearbeitende Pfade. Wie effizient das Datenbanksystem dies erledigen kann, sollen die Ausführungen dieser beiden Queries zeigen.
Construction of Complex Results und Joins on Values	Q10-Q12	Diese beiden Konzepte beschäftigen sich mit der Fähigkeit des Datenbanksystems mit komplexen und grossen Zwischen- und Endergebnisse umgehen zu können. Dies ist ein wichtiger Punkt, weil viele Applikationen fordern, dass mit grossen Datenvolumen umgegangen werden muss. U.a. zeigen die Queries auch, wie gut der Query arbeitet, der den günstigsten Evaluierungsplan auswählen muss. Folgende Zahlen sollen einen Einblick in die Grössendimension geben: Bei der Query für das Konzept der komplexen Resultate erhält man beispielsweise bei einem 100MB grossen Dokument ein Resultat von 10MB. Die Query aus dem Konzept Joins and Values erfordert bei einem 100MB grossen Dokument zwischenzeitlich ein Join mit einer Grösse von 12 Mio. Tupeln.

Kategorie	Query	Beschreibung
Reconstruction aka Round Tripping	Q13	Rekonstruktion bedeutet, dass das Datenbanksystem die zur physikalischen Speicherung herunter gebrochenen XML Daten wieder zum Ganzen bzw. zu einem Fragment des Original XML Dokument zusammenfügt. In der XML Welt ist Rekonstruktion eine häufige Operation. Gerade XML bietet dem User an, die Daten immer wieder zu verwenden. Denn mit Hilfe von XML Daten kann der Anwender ein und denselben Inhalt in verschiedene Layouts wie HTML Seiten, Broschüren etc. einbinden. Die Query verlangt in diesem Fall nicht eine komplette Rekonstruktion des original Dokumentes, sondern nur die Rekonstruktion eines Unterbaums - alle Items, die in der Region Australien registriert sind. Im Falle eines relationalen Datenbanksystems wird die Operation Rekonstruktion sehr teuer sein, weil die XML Daten in Relationen bzw. Tupeln transformiert wurden. Bei der Rekonstruktion werden von daher viele teure Joins benötigt, um das Original XML Dokument wiederherzustellen. Ein XML Datenbanksystem allerdings, das XML Dokumente nativ speichert, könnte auf solche teuren Joins verzichten, da die Daten schon in einer XML-Hierarchie abgelegt sind.
Fulltext	Q14	Volltextsuche ist eine elementare Operation bei XML Anfragen. In diesem Konzept wird getestet, wie eine Volltextsuche von Schlüsselwörtern bearbeitet wird.
Path Traversals	Q15-Q16	Für das Konzept der Pfad-Traversierung werden zwei Queries herangezogen. „Print the keywords in emphasis in annotations of closed auctions.“ Weiter auf diese Query aufbauend ist die zweite Query. „Return the IDs of the sellers of those auctions that have one or more keywords in emphasis.“ Diese beiden Queries sind an Informationen interessiert die in grosser Tiefe des Baumen zu finden sind. Von dem Datenbanksystem wird also eine lange Pfadtraversierung gefordert. Dies gibt Aufschluss darüber, wie hoch die Kosten für lange Pfadtraversierungen sind. Ein XML Datenbanksystem, das XML Daten nativ speichert, müsste - ohne geeignete Methoden - sich durch das ganze Dokument arbeiten, um die Suche zu vollenden. Relationale Datenbanksystem hingegen umgehen diese Schwierigkeit, da sie das XML Dokument in kleinere Teile (Relationen) aufteilen müssen, so dass diese kleinen Teile des XML Dokuments separat schneller durchsucht werden können.

Kategorie	Query	Beschreibung
Missing Element	Q17	Diese Query, die für das Konzept fehlende Elemente ausgewählt wurde, lautet „Which persons don't have a homepage?“ Was diese Query bringen soll, ist offensichtlich. Da nicht jede Person eine Homepage besitzen muss, stellt sich also hier die Frage, wie das XML Datenbanksystem mit semi-strukturierten XML Daten umgehen kann. Nicht existierende Attribute werden in der Datenbankwelt beim physikalischen Abspeichern mit NULL Werten gekennzeichnet. Was hier also getestet wird, ist wie effizient die Methoden des Datenbanksystems sind, die auf NULL Werte testen. NULL Werte sind häufig - wie in diesem Anwenderszenario - von grossem Interesse.
Function Application	Q18	„Convert the currency of the reserves of all open auctions to another currency.“ Diese Query testet, inwiefern das Datenbanksystem mit benutzerdefinierten Funktionen zurechtkommt.
Sorting & Aggregation	Q19-Q20	Schliesslich gibt es noch die beiden Konzepte Sortierung und Aggregation. Diese sind bereits wichtige Operationen aus der relationalen Datenbankwelt und werden ebenfalls in diesem XML Benchmark getestet.

Tabelle 8.1: XMark Queries

8.2.1 Vorbereitung von Hellgate

Sleepycat läuft nur in einem einzelnen Prozess und wird augenscheinlich durch den Synchronisationsaufwand eines Doppelprozessorsystems ausgebremst. Deshalb wird für die Benchmarks nur ein Prozessor verwendet. Erreicht wird dies durch einen Eintrag in der boot.ini (/NUMPROC=1). Hypertreading ist ebenfalls deaktiviert. Vor den Benchmarkläufen wird das System einmalig defragmentiert. Zusätzlich werden folgende Einstellungen anhand des Tools TuneXP (<http://www.driverheaven.net>) aktiviert:

- accelerate dll unloading
- clear pagefile on shutdown
- file allocation size tweak
- optimize prefetch
- increase ntfs performance
- io page lock limit > 512MiB

8.3 Ablauf der Performancetests

Alle Performancetests werden mit einer selbst entwickelten Java-Applikation durchgeführt. Alle Performancetests laufen über das Framework. Zum Testen der Datenbanken müssen insgesamt 4 Durchläufe pro Datenbank getätigt werden, um alle benötigten Daten zu ermitteln. Dabei ist zu beachten, dass nach jedem Datenimport das System neu gestartet wird, um Caching vom Betriebssystem bzw. der Datenbank auszuschließen. Dadurch ergibt sich folgender Ablauf für die Tests:

1. Datenimport 1MB
2. Reboot
3. Testdurchlauf 1MB
4. Löschen der Testdaten
5. Datenimport 10MB
6. Reboot
7. Durchlauf 10MB
8. Löschen der Testdaten
9. Danach erfolgt die Optimierung des Datenbestandes durch Indizierung. Dies wird durch setzen eines Flags im Framework initiiert. Durch dieses Flag werden vor dem Import der Daten die benötigten Indizes gesetzt.

10. Datenimport 1MB
11. Reboot
12. Testdurchlauf 1MB
13. Löschen der Testdaten
14. Datenimport 10MB
15. Reboot
16. Durchlauf 10MB
17. Löschen der Testdaten

Nach jedem Durchlauf werden die ermittelten Zeiten aus der generierten CSV-Datei in ein Excelsheet eingetragen.

8.4 Test-Framework

Das in Java geschriebene Testframework gliedert sich in fünf Packages mit insgesamt neun Klassen. Es stehen zwei Main-Klassen mit unterschiedlicher Funktionalität zur Verfügung. Die Klassen werden nachfolgend kurz beschrieben. Für die ausführliche Dokumentation der Klassen wird an dieser Stelle auf die Javadoc von XMLBench verwiesen. Die Sourcecodes finden sich im Anhang dieses Dokuments.

XMLBench startet die Benchmarks. Beim Aufruf werden drei Argumente übergeben:

- `db2|Sleepycat` gibt die Datenbank an, die getestet werden soll
- `load|query` gibt an, ob der Import der Testdaten oder die Ausführung der Queries gestartet werden soll.
- `1|2|3` gibt an, welche der generierten Testdateien verwendet werden soll

Nach dem Start werden im Falle des Arguments `load` die noch vorhandenen Testdaten der entsprechenden Datenbank gelöscht. Anschliessend werden die neuen Testdaten importiert. Im Falle des Parameters `query` werden die als Dateien vorhandenen Queries der jeweiligen Datenbank geparkt und als `XMarkQueryObjects` in einem Vector zurückgegeben. Nachfolgend findet die Ausführung der Queries statt. Die Testergebnisse werden zur weiteren Auswertung in einer csv-Datei gespeichert.

XMLGenerator startet die Generierung der Testdateien für den Datenbankimport. Beim Aufruf werden drei Argumente übergeben, die die Grösse der generierten Dateien als Faktor angeben. Multipliziert man den Faktor mit 113, erhält man die Dateigrösse in MB. Bspw. ergibt der Faktor 0.1 eine 13 MB grosse Datei. Zur Generierung wird das Binary-File von XMark verwendet und in XMLBench in einem Prozess aufgerufen.

db.DB2Connector stellt den Connector zur DB2 dar. Die Klasse bietet vier Methoden an:

- `importInDb` importiert die Testdatei in die Datenbank
- `executeQueries` startet den eigentlichen Test
- `dropTestData` löscht die vorhandenen Testdaten aus der Datenbank
- `close` schliesst die Verbindung zur Datenbank

db.Sleepycat stellt den Connector zu Sleepycat dar. Die Klasse bietet acht Methoden an:

- `importInDb` importiert die Testdatei in die Datenbank
- `executeQueries` startet den eigentlichen Test
- `dropTestData` löscht die vorhandenen Testdaten aus der Datenbank
- `close` wird für Sleepycat nicht verwendet
- `setCacheSize` setzt die Sleepycat-Cache-Grösse
- `prepareQueries` bereitet die Queries anhand regulärer Ausdrücke vor
- `regexXmark` ist eine Hilfsfunktion von `prepareQueries`

db.Sleepycat.* enthält zwei Hilfsklassen für `db.Sleepycat`

results.ResultGenerator dient der Verarbeitung der Testergebnisse. Die Klasse beinhaltet zwei Methoden:

- `generateCSV` generiert eine CSV-Datei mit den Testergebnissen
- `getDate` formatiert die Ausführungszeit des aktuellen Tests und gibt diese zurück

xmark.XMarkGenerator ist für XMark-spezifische Operationen vorbehalten. Es stehen zwei Methoden zur Verfügung:

- `generateTestFile` generiert eine Datei mit den Importdaten unter Zuzufnahme der XMark-Binary
- `parseQueries` durchläuft das Verzeichnis der XMark-Queries und speichert diese als `XMarkQueryObjects` in einem Vector

xmark.XMarkQueryObject ist eine JavaBean zur Speicherung der Testinformationen. JavaBean-spezifisch werden für die folgenden Attribute **get**- und **set**-Methoden angeboten:

- `queryName` enthält den Namen der Query
- `query` enthält die Query als String
- `queryResult` ist für das Ergebnis der Query angedacht, jedoch ist dies bei grossen Testdaten nicht praktikabel
- `queryTime1` enthält die Zeit, die die Query für den ersten Durchlauf benötigt hat
- `queryTime2` enthält die Zeit, die die Query für den zweiten Durchlauf benötigt hat
- `queryTime3` enthält die Zeit, die die Query für den dritten Durchlauf benötigt hat
- `queryTime4` enthält die Zeit, die die Query für den vierten Durchlauf benötigt hat

Anmerkung:

Um das Testframework auf ein anderes System zu portieren, muss darauf geachtet werden, dass die Pfade der eingebundenen Libraries im Build Path des Eclipse-Projektes angepasst werden.

8.5 Auswertung

8.5.1 DB2

Um die Tests mit DB2 auszuführen, müssen nach dem Erzeugen der Datenbank die Routinen für JDBC mit `db2set DB2_USE_DB2JCCT2_JROUTINE=on -g` aktiviert werden. Da beim Test sehr grosse Datenmengen entstehen, muss die Applicationheapsize mit `db2 update db cfg for xmark using applheapsz 51200` erhöht werden. Dies kann mit dem Batchfile `xmark.bat`, welches im Binary-Verzeichniss von `XBench` liegt, automatisch vorgenommen werden. Dieses Batchfile muss im DB2 Command Window aufgerufen werden.

Bei den Tests ergaben sich die aus den Tabellen 8.2 bis 8.5 ersichtlichen Ergebnisse (in ms).

8.5.2 Sleepycat

Vor dem Start des Benchmarks mit Sleepycat muss ein Verzeichnis auf der Festplatte erstellt werden, welches die Sleepycat-Datenbank beinhalten wird. Weiterhin müssen die selbsterklärenden Klassenvariablen in `SleepycatConnector.java` angepasst werden. Die Tests auf Sleepycat ergaben die aus den Tabellen 8.6 bis 8.9 ersichtlichen Zeiten (in ms).

8 Performancetests

DB2						
1MB without Index						
Generated on Sa, 2. Jul 2005 at 18:48:21						
Category	XQuery	1st round	2nd round	3rd round	4th round	Avg
		uncached	cached			
Exact Match	Q1	390	16	16	0	105
Ordered Access	Q2	156	63	47	47	78
	Q3	94	93	93	78	89
	Q4	32	16	31	31	27
Casting	Q5	609	0	0	16	156
Regular Path Expressions	Q6	844	32	31	15	230
	Q7	516	109	110	125	215
Chasing References	Q8	422	375	391	406	398
	Q9	437	438	422	421	429
Complex Results und Joins	Q10	1860	1328	1453	1453	1523
	Q11	609	610	593	594	601
	Q12	188	171	188	172	179
Round Tripping	Q13	15	16	16	15	15
	Q14					
Fulltext	Q15	15	16	16	15	15
Path Traversals	Q16	15	16	16	16	15
Missing Element	Q17	62	47	47	47	50
Function Application	Q18					
Sorting und Aggregation	Q19	63	47	62	63	58
	Q20	47	16	16	32	27

Tabelle 8.2: DB2-Ergebnisse mit 1MB (ohne Index)

DB2						
10MB without Index						
Generated on Sa, 2. Jul 2005 at 20:20:31						
Category	XQuery	1st round	2nd round	3rd round	4th round	Avg
		uncached	cached			
Exact Match	Q1	609	78	94	63	211
Ordered Access	Q2	1171	547	454	453	656
	Q3	922	938	922	938	930
	Q4	266	266	266	265	265
Casting	Q5	500	47	31	31	152
Regular Path Expressions	Q6	1422	156	156	172	476
	Q7	1454	1203	1094	1110	1215
Chasing References	Q8	57500	57468	57437	57422	57456
	Q9	72063	72172	72188	72063	72121
Complex Results und Joins	Q10	394765	729203	1044312	1397125	891351
	Q11	102046	101968	102812	102016	102210
	Q12	21156	21047	20938	20969	21027
Round Tripping	Q13	172	156	172	156	164
	Q14					
Fulltext	Q15	62	63	62	78	66
Path Traversals	Q16	109	110	94	110	105
Missing Element	Q17	484	500	484	484	488
Function Application	Q18					
Sorting und Aggregation	Q19	703	672	672	672	679
	Q20	422	391	406	375	398

Tabelle 8.3: DB2-Ergebnisse mit 10MB (ohne Index)

8 Performancetests

DB2						
1MB with Index						
Generated on Sa, 2. Jul 2005 at 22:36:47						
Category	XQuery	1st round	2nd round	3rd round	4th round	Avg
		uncached	cached			
Exact Match	Q1	203	0	0	15	54
Ordered Access	Q2	172	63	63	62	90
	Q3	109	94	94	94	97
	Q4	32	31	32	15	27
Casting	Q5	94	15	0	0	27
Regular Path Expressions	Q6	281	15	16	16	82
	Q7	172	110	109	125	129
Chasing References	Q8	438	406	422	438	426
	Q9	469	469	453	453	461
Complex Results und Joins	Q10	1984	1453	1453	1546	1609
	Q11	656	735	672	641	676
	Q12	203	187	188	188	191
Round Tripping	Q13	31	15	0	15	15
	Q14					
Fulltext	Q15	16	16	15	0	11
Path Traversals	Q16	31	15	0	15	15
Missing Element	Q17	62	62	47	62	58
Function Application	Q18					
Sorting und Aggregation	Q19	78	47	47	62	58
	Q20	78	31	31	31	42

Tabelle 8.4: DB2-Ergebnisse mit 1MB (mit Index)

8 Performancetests

DB2 10MB with Index						
Generated on So, 3. Jul 2005 at 00:12:15						
Category	XQuery	1st round	2nd round	3rd round	4th round	Avg
		uncached	cached			
Exact Match	Q1	640	94	78	79	222
Ordered Access	Q2	1297	609	500	484	722
	Q3	969	938	985	1079	992
	Q4	281	297	281	266	281
Casting	Q5	594	47	47	63	187
Regular Path Expressions	Q6	1578	172	172	156	519
	Q7	1516	1125	1203	1094	1234
Chasing References	Q8	59438	57422	57391	57359	57902
	Q9	73406	72484	73125	72250	72816
Complex Results und Joins	Q10	399609	742875	1040469	1393719	894168
	Q11	102829	103016	102953	102890	102922
	Q12	21266	21219	21297	21328	21277
Round Tripping	Q13	203	156	172	156	171
	Q14					
Fulltext	Q15	47	47	47	63	51
Path Traversals	Q16	125	110	94	109	109
Missing Element	Q17	515	500	500	500	503
Function Application	Q18					
Sorting und Aggregation	Q19	687	657	657	656	664
	Q20	454	391	391	406	410

Tabelle 8.5: DB2-Ergebnisse mit 10MB (mit Index)

8 Performancetests

Sleepycat 1MB without Index						
Generated on Sa, 2. Jul 2005 at 22:31:49						
Category	XQuery	1st round	2nd round	3rd round	4th round	Avg
		uncached	cached			
Exact Match	Q1	94	31	15	0	35
Ordered Access	Q2	235	31	16	31	78
	Q3	31	47	31	47	39
	Q4	94	125	172	94	121
Casting	Q5	109	16	31	15	42
Regular Path Expressions	Q6	313	94	93	79	144
	Q7	328	765	844	375	578
Chasing References	Q8	1375	1547	1453	1547	1480
	Q9	1500	1516	1531	1562	1527
Complex Results und Joins	Q10	610	578	578	594	590
	Q11	2344	2422	2500	2531	2449
	Q12	844	859	860	859	855
Round Tripping	Q13	16	15	16	16	15
	Q14	500	250	250	265	316
Fulltext	Q15	63	78	62	79	70
Path Traversals	Q16	31	31	31	32	31
Missing Element	Q17	31	31	31	47	35
Function Application	Q18					
Sorting und Aggregation	Q19	250	500	500	703	488
	Q20	110	109	125	109	113

Tabelle 8.6: Sleepycat-Ergebnisse mit 1MB (ohne Index)

8 Performancetests

Sleepycat 10MB without Index						
Generated on So, 3. Jul 2005 at 07:56:46						
Category	XQuery	1st round	2nd round	3rd round	4th round	Avg
		uncached	cached			
Exact Match	Q1	2500	110	109	125	711
Ordered Access	Q2	609	219	203	203	308
	Q3	407	1718	797	703	906
	Q4	844	2438	2703	3109	2273
Casting	Q5	953	203	203	219	394
Regular Path Expressions	Q6	3547	6016	7047	6203	5703
	Q7	9468	35250	43782	45093	33398
Chasing References	Q8	174532	403140	909813	1084469	642988
	Q9	1352625	1472500	1555750	1606375	1496812
Complex Results und Joins	Q10	50110	51750	52750	52453	51765
	Q11	1092937	1130953	1149563	1155937	1132347
	Q12	67485	67609	68000	68547	67910
Round Tripping	Q13	172	156	141	140	152
	Q14	2594	2516	2500	62828	17609
Fulltext	Q15	87984	45235	38062	70797	60519
Path Traversals	Q16	281	282	281	281	281
Missing Element	Q17	344	328	344	328	336
Function Application	Q18					
Sorting und Aggregation	Q19	534015	446579	385921	358079	431148
	Q20	1234	1141	1156	1156	1171

Tabelle 8.7: Sleepycat-Ergebnisse mit 10MB (ohne Index)

8 Performancetests

Sleepycat 1MB with Index						
Generated on Sa, 2. Jul 2005 at 22:43:05						
Category	XQuery	1st round	2nd round	3rd round	4th round	Avg
		uncached	cached			
Exact Match	Q1	141	15	16	16	47
Ordered Access	Q2	47	15	31	32	31
	Q3	31	31	63	31	39
	Q4	94	140	172	94	125
Casting	Q5	219	15	16	31	70
Regular Path Expressions	Q6	422	78	94	94	172
	Q7	343	766	828	531	617
Chasing References	Q8	1391	1500	1609	1500	1500
	Q9	1500	1563	1547	1594	1551
Complex Results und Joins	Q10	593	578	579	625	593
	Q11	2390	2453	2532	2547	2480
	Q12	859	859	859	875	863
Round Tripping	Q13	16	0	16	15	11
	Q14	516	313	281	266	344
Fulltext	Q15	62	78	79	78	74
Path Traversals	Q16	31	31	31	32	31
Missing Element	Q17	31	47	31	47	39
Function Application	Q18					
Sorting und Aggregation	Q19	266	531	531	594	480
	Q20	109	125	110	109	113

Tabelle 8.8: Sleepycat-Ergebnisse mit 1MB (mit Index)

Sleepycat 10MB with Index						
Generated on So, 3. Jul 2005 at 14:03:24						
Category	XQuery	1st round	2nd round	3rd round	4th round	Avg
		uncached	cached			
Exact Match	Q1	1844	125	125	125	554
Ordered Access	Q2	2094	375	203	203	718
	Q3	469	1703	547	781	875
	Q4	875	2359	2750	2922	2226
	Q5	438	203	203	203	261
Casting	Q5	438	203	203	203	261
Regular Path Expressions	Q6	3781	6250	6922	6094	5761
	Q7	6797	37187	44859	44469	33328
	Q8	175922	397781	893688	1205750	668285
Chasing References	Q9	1395344	1515484	1602656	1668094	1545394
	Q10	51219	51937	51875	53078	52027
	Q11	1118218	1146203	1157891	1163578	1146472
Complex Results und Joins	Q12	66219	66172	66078	66000	66117
	Q13	156	141	140	141	144
	Q14	2547	2484	2500	62735	17566
Fulltext	Q15	79687	50672	34625	69125	58527
Path Traversals	Q16	266	265	266	266	265
Missing Element	Q17	375	312	313	328	332
Function Application	Q18					
Sorting und Aggregation	Q19	549984	453656	392828	352563	437257
	Q20	1125	1125	1125	1125	1125

Tabelle 8.9: Sleepycat-Ergebnisse mit 10MB (mit Index)

8.5.3 Ergebnisse

Die Auswertung der Ergebnisse wird wegen der sehr stark variierenden Zeiten in die verschiedenen Kategorien der Anforderung unterteilt. Wir betrachten hier jeweils nur die erste Zeit, da es sehr unwahrscheinlich ist, dass exakt gleiche Anfragen direkt aufeinanderfolgend an die Datenbank gestellt werden.

Exact Match

Exact Match sucht nach einem einzelnen Element und gibt dieses zurück. Im obenstehenden Bild ist zu erkennen, dass bei kleinen Datenmengen Sleepycat geringfügig schneller ist. Auch bei anderen Queries mit geringen Datenmengen tritt dies auf (z.B. Ordered Access & Casting). Möglicherweise liegt dies an der Zeit, die der DB2 Optimierer zum Vorbereiten der Query benötigt. Bei kleinen Datenbankgrößen überschreitet die Zeit der Optimierung wahrscheinlich die durch die Optimierung gesparte Zeit. Bei grossen Daten-

8 Performancetests

mengen hingegen liegt DB2 dann klar vorne, was auf einen guten Optimierer schliessen lässt.

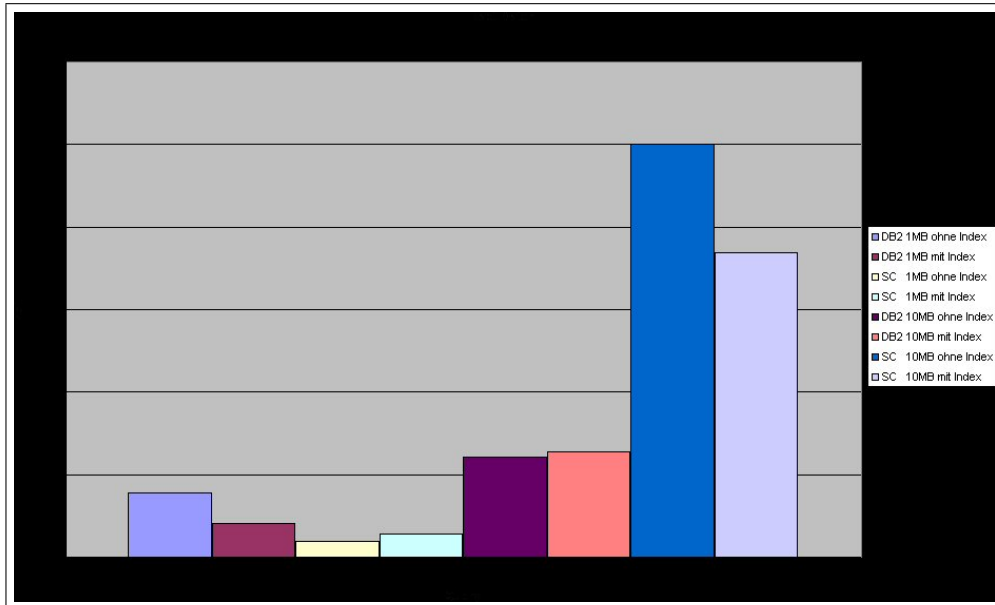


Abbildung 8.1: Auswertung Exact Match

Ordered Access

Bei den Queries zu Ordered Access wird geprüft, inwiefern der Optimierer mit der Reihenfolge der Elemente und deren Sortierung zurechtkommt. Hier fällt auf, dass Sleepycat bei den ersten beiden Queries gut mithalten kann, in der dritten Query aber dann klar zurückliegt. Dies könnte an der unterschiedlichen Formulierung der Queries liegen. Bei Query 1 und 2 wurde im XPath-Ausdruck das Element mit Index angegeben:

```
-->$b/bidder[1]/increase/text()
```

In der 3. Query wird das Gesamtergebnis sortiert:

```
-->$b/bidder/personref[@person='`person18829`']
```

```
<< $b/bidder/personref[@person='`person10487`']
```

Lässt man den Ausreisser mit Indizierung ausser Acht, dann kommt Sleepycat mit der direkten Adressierung der Elemente besser zurecht als DB2, braucht dafür aber sehr lange, um das gesamte Ergebnis zu sortieren.

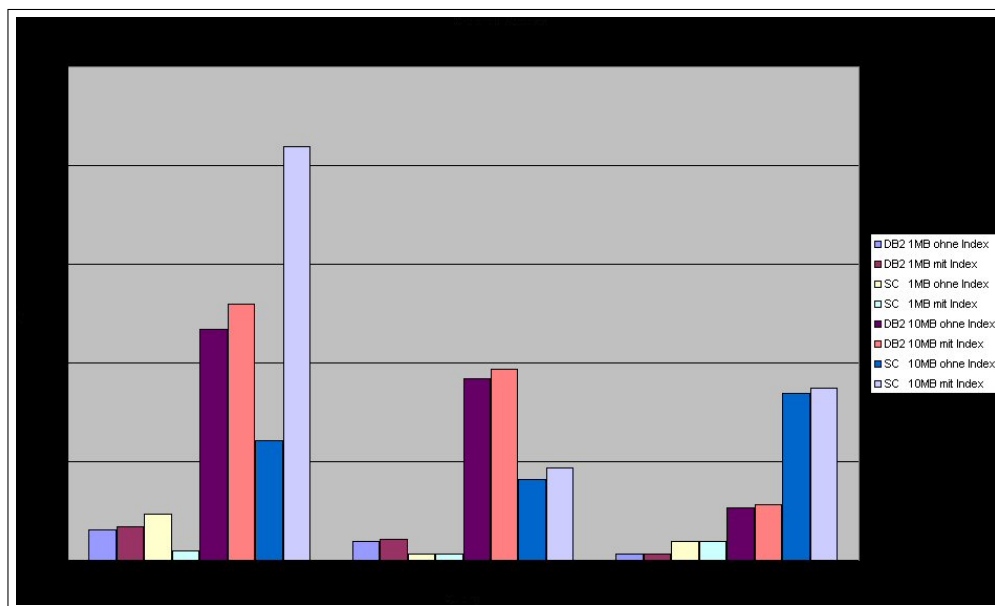


Abbildung 8.2: Auswertung Ordered Access

Casting

Beim Casting wird zum Vergleichen zweier Werte eine Typumwandlung benötigt. In der Grafik ist zu erkennen, dass hier eine Indizierung enorme Geschwindigkeitsvorteile bringen kann. Allerdings profitiert DB2 nur bei kleinen Datenbankgrößen, Sleepycat nur bei grossen Datenbankgrößen davon. Weshalb sich die Zeit im umgekehrten Fall bei Indizierung verlängert, ist schleierhaft, da eine Indizierung gerade in diesem Fall hilfreich ist.

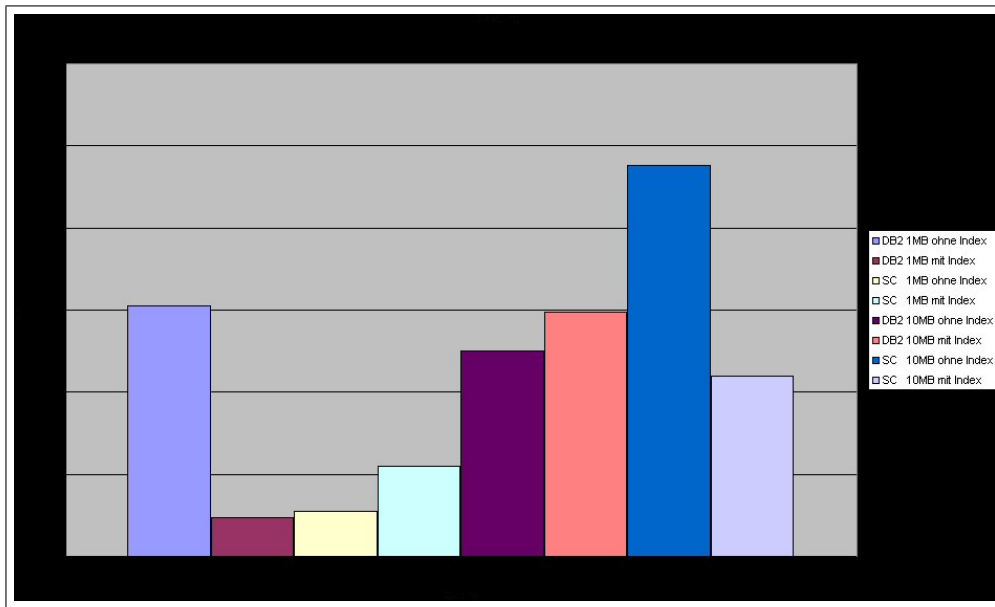


Abbildung 8.3: Auswertung Casting

Regular Path Expressions

Bei diesen Tests wird gemessen, wie schnell sich die Engine im Pfad bewegen kann. Hier fällt wieder auf, dass Sleepycat bei der 10-fachen Datenmenge über 30 mal so lange braucht. Dies ist bei der 2. Query der Fall, in der allerdings auch 3 Aggregatsfunktionen verwendet werden, im Gegensatz zur 1. Query, in der nur eine verwendet wird. Bei DB2 hingegen ist die benötigte Zeit bei beiden Queries nahezu gleich.

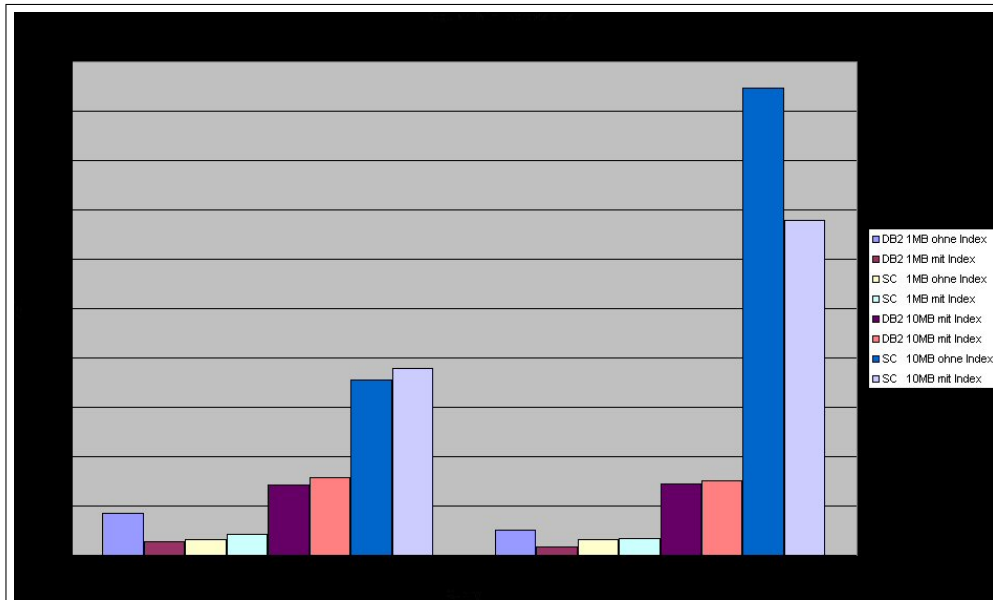


Abbildung 8.4: Auswertung Regular Path Expressions

Chasing References

Hier werden Subqueries an die Datenbank gestellt, wobei in der Grafik deutlich der Vorteil von DB2 zu erkennen ist. Mit einer Ausführungszeit von über 20 Minuten bei einem 10MB Dokument tut sich Sleepycat hier äusserst schwer.

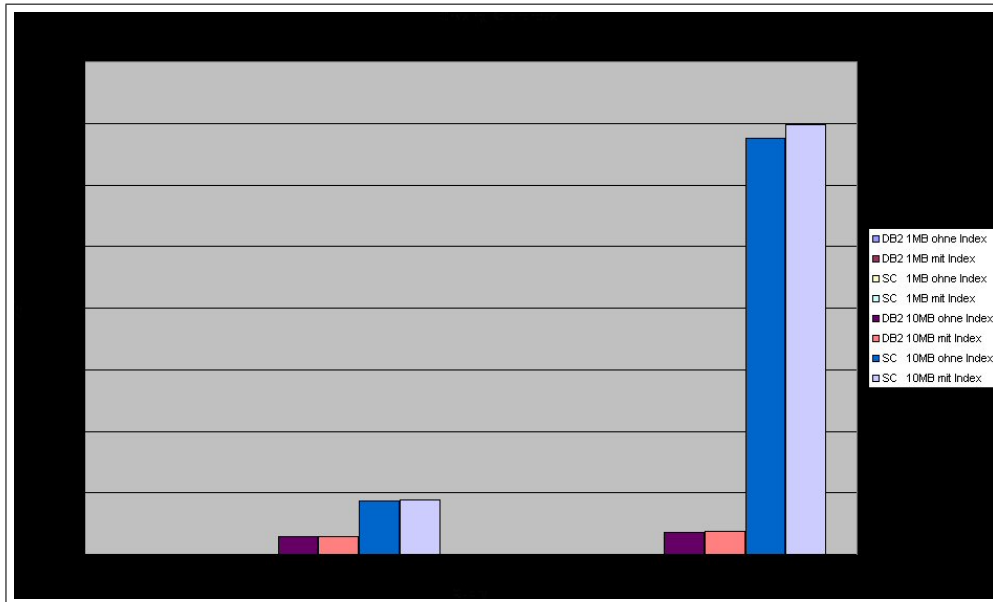


Abbildung 8.5: Auswertung Chasing References

Complex Results and Joins

Das schlechte Ergebnis von DB2 in der ersten Query ist nach den bisherigen Ergebnissen sehr überraschend. In dieser Query wird die Funktion `distinct-values()` verwendet. Bei einem zusätzlichen Test ohne diese Funktion konnte jedoch kein massgeblicher Unterschied festgestellt werden. Eine andere Ursache könnte die sehr häufige Verwendung der funktion `text()` sein, was aber noch getestet werden müsste.

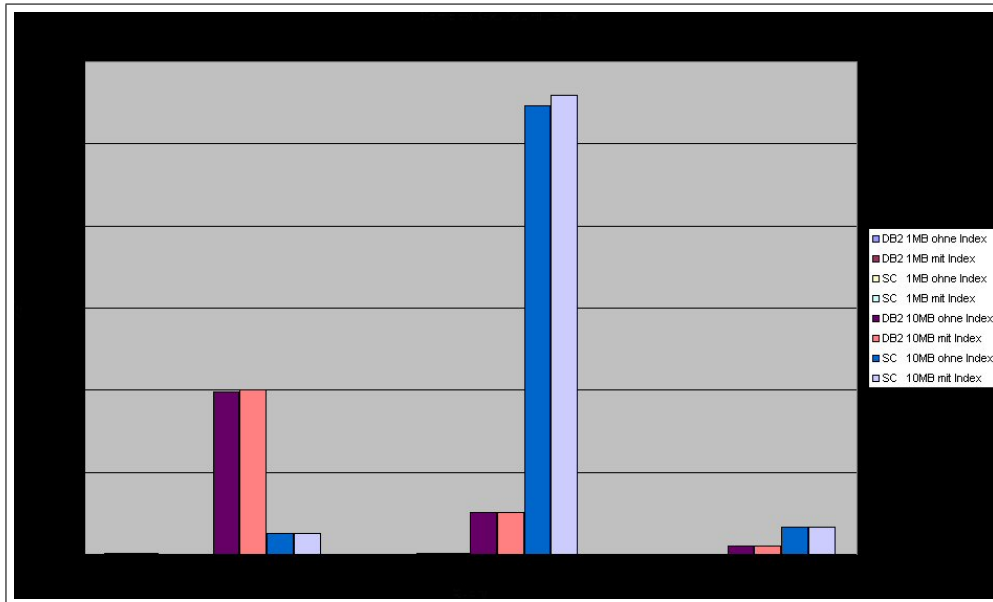


Abbildung 8.6: Auswertung Complex Results and Joins

Volltextsuche

Es ist zu beachten, dass die oben dargestellte Grafik logarithmisch skaliert ist. Sleepycat hat hier sehr grosse Probleme Substrings in größeren Datenbanken zu finden.

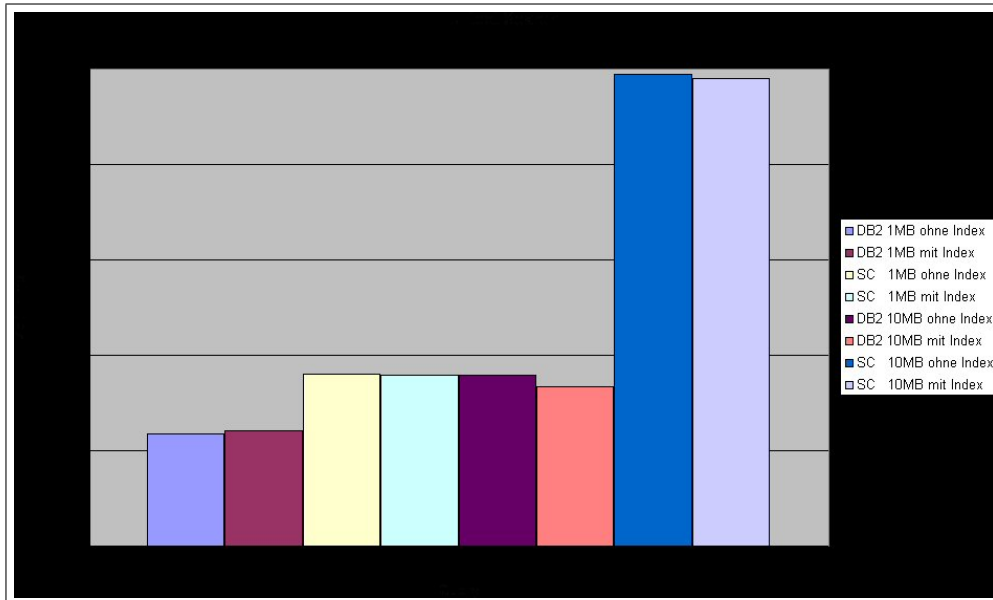


Abbildung 8.7: Auswertung Volltextsuche

Allgemein

Die restlichen Queries zeigen grösstenteils das gleiche Bild. Bei kleinen Datenbankgrößen ist Sleepycat gut im Rennen, bei grösseren aber eindeutig langsamer. Verwunderlich ist das Ergebnis von Sleepycat bei der 4. Query. Hier wird auf die Existenz von Elementen überprüft. Obwohl bei der Indizierung von Elementen genau auf „present“ indiziert werden kann, ist hier kein Geschwindigkeitsvorteil gegenüber der Zeit ohne Index erkennbar.

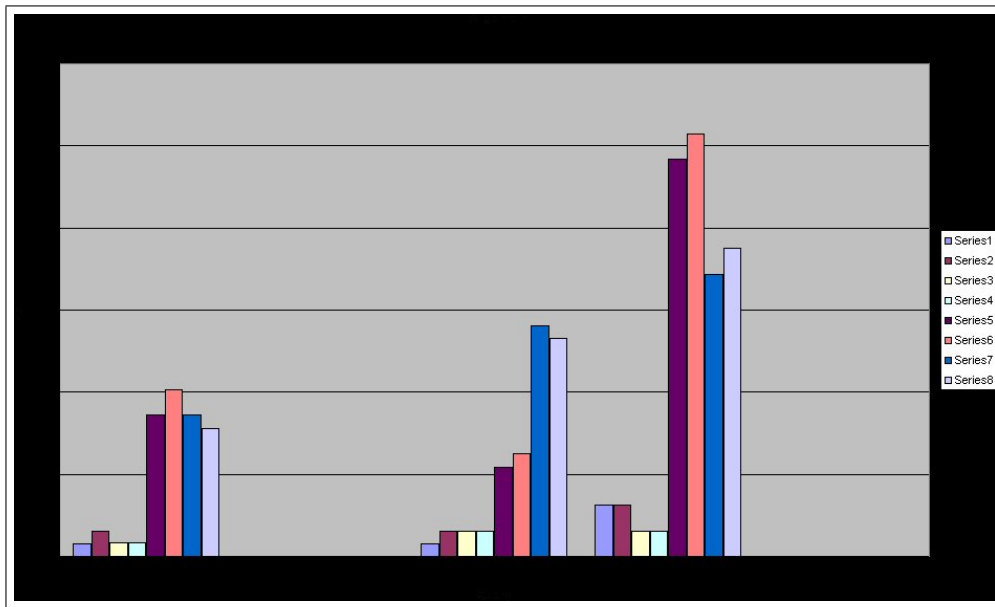


Abbildung 8.8: Auswertung Allgemein

Sorting and Agregation

Während diesem Test werden wiederum sehr viele Ergebnisse sortiert und gezählt. Bei der ersten Query fällt auf, dass Sleepycat hier extrem viel Zeit benötigt - aus diesem Grund die logarithmische Darstellung der Zeit. Wie bei den Tests zu Ordered Access muss hier auch das Gesamtergebnis sortiert werden. Das scheint eine Schwäche von Sleepycat zu sein.

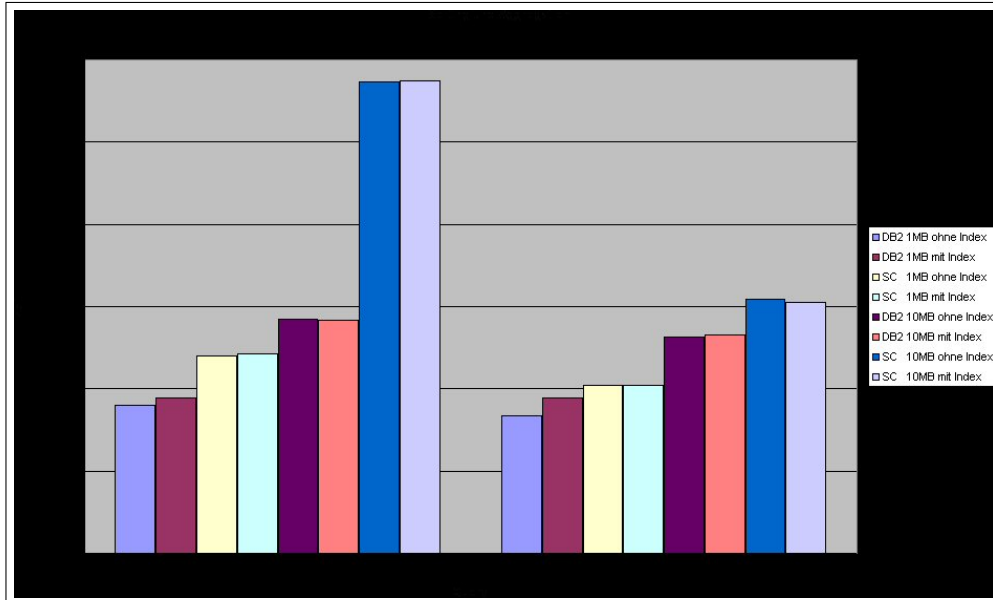


Abbildung 8.9: Auswertung Sorting and Aggregation

Caching

Ein unerwartetes Phänomen von Sleepycat ist, dass die meisten Queries bei wiederholender Ausführung immer langsamer werden. Eine Lösung dazu konnte nicht gefunden werden. Es liegt die Vermutung nahe, dass es sich hierbei um einen Caching-Fehler von Sleepycat handelt. Dennoch ist die aktuelle Cache-Konfiguration der Beispiele aus momentaner Sicht schlüssig.

Im Gegensatz dazu profitiert DB2 von einer wiederholten Ausführung. Bei Queries, in denen direkt im XPath-Ausdruck nach Elementen gesucht wird, verkürzt sich die Zeit um ein Vielfaches. Ansonsten bleiben die Werte konstant und lassen auf eine hohe Stabilität schliessen. Die einzige Query, bei der sich die Ausführungszeit erhöht hat, ist Query 1 von Complex Results and Joins. Diese Query stellt DB2 - wie bereits oben besprochen - vor grosse Probleme, was sich wiederum im Cacheverhalten niederschlägt.

8.5.4 Fazit

Im Laufe der Analyse der Testdaten stellt sich heraus, dass die Testqueries fast ausschließlich die Verarbeitung der Datensätze testen und kaum den Zugriff auf die XML-Daten. Um die Performance bei grossen Datenmengen zu messen, sollten hauptsächlich XPath-Ausdrücke verwendet werden, um einen Überblick darüber zu geben, wie schnell der Datenzugriff stattfindet. Dies würde vermutlich auch einen Geschwindigkeitsvorteil durch Indizierung nach sich ziehen. Als Untermauerung dieser Aussage dienen folgende Erkenntnisse bei Sleepycat: In Q05 (Casting) bringt bei aktivierten Indizes erst ein Erhöhen des Preisvergleichs von 40 auf den nie auftretenden Wert von 1000 einen Geschwindigkeitszuwachs, da keine Ergebnisse verarbeitet werden müssen. Es werden nur die Werte der Elemente verglichen.

Bei DB2 fällt auf, dass bei kleinen Datenmengen eine höhere Anlaufzeit als bei Sleepycat anfällt. Vermutlich ist dies auf die Komplexität des DB2-Optimierers zurückzuführen ist. Dies würde auch die Geschwindigkeitsvorteile bei grossen Datenmengen erklären.

Die grösste Herausforderung für die XML-Datenbanken besteht im Handling mit Subqueries. Hier werden bei einem Datenbestand von 10 MB extrem hohe Zeiten im Bereich von bis zu 20 Minuten benötigt. Bei Sleepycat ist noch zu erwähnen, dass beim Sortieren von grossen Datenmengen die Zeiten generell unverhältnismässig ansteigen.

9 XML-Datenbanken und grosse Datenmengen

9.1 Abfragen auf Sleepycat

Sleepycat hat keine Probleme beim Import großer Datenmengen. Allerdings benötigt die Ausführung von verschachtelten Queries mit großen Datenmengen viel Arbeitsspeicher. Beispielsweise beschlagnahmt der Prozess bei einer 10MB XMark Datenbank bis zu 1GB RAM.

9.2 Abfragen auf IBM DB2 Nativ XML Alpha Program

Bei Anfangs geplanten Testläufen mit 100MB Repository kamen wir zu dem Ernüchternden Ergebnis, dass Querys teilweise über 2 Stunden brauchten. Dies führte zur Entscheidung, die Tests nur mit 1 und 10MB laufen zu lassen. Wie bei den Performancetests schon erwähnt, wurde bei diesen Tests hauptsächlich die Verarbeitung der Daten getestet. Sollte DB2 mit mit größeren Datenmengen zum Einsatz kommen, stellt die Speicherung und Organisation von XML-Daten kein Problem dar, beim Einsatz von komplexen Querys sollte jedoch hinblicklich der Performanz aufgepasst werden.

9.3 Abfragen auf Saxon

Da die Berechnungszeit bei DB2 und Sleepycat bei steigender Datenmenge ins unermessliche steigen, haben wir die Querys zusätzlich von Saxon bearbeiten lassen. Saxon ist ein reiner XML-Interpreter mit Java-Schnittstelle, der auf XML-Files im lokalen Dateisystem zugreift. Hierbei wurden ähnlich hohe Antwortzeiten erreicht, bei Querys die sehr viele Daten liefern, brach Saxon sogar wegen einer `java.lang.OutOfMemoryException` ab. Das lässt darauf schließen, dass die hohen Rechenzeiten durch die Komplexität der XMark-querys entstehen und nicht durch Schwierigkeiten große Datenmengen zu speichern und durchsuchen.

10 Fazit

Die Ziele des Semesterprojektes wurden den Anforderungen nach erfüllt, lediglich der Test einer Middleware-Lösung konnte aufgrund ungeeigneter Kandidaten nicht durchgeführt werden. Der Test einer Middleware-Lösung wäre insofern interessant gewesen, da sowohl mit XQuery als auch mit SQL auf den selben Datenbestand zugegriffen werden kann.

Abschliessend bleibt zu sagen, dass der XQuery-Support sowie die Performanz der betrachteten Kandidaten noch sehr zu wünschen übrig lässt.

Möglicherweise wird sich dies in den nächsten Jahren ändern, da einige grosse Datenbank-Hersteller wie bspw. Oracle oder MS SQL Server XQuery-Support für die nächsten Releases angekündigt haben.

Wir möchten uns für die sehr gute Betreuung bei Herrn Prof. Dr. Lothar Piepmeyer bedanken. Unser Dank gilt auch den Mitarbeitern des Foo-Pools.

A W3 Use Cases

A.1 NS

A.1.1 Q1

```
1 <Q1>
2 {
3   for $n in distinct-values(
4     for $i in (doc("auction.xml")//* | doc("auction.xml")//@*)
5       return namespace-uri($i)
6   )
7   return <ns>{ $n }</ns>
8 }
9 </Q1>
```

A.1.2 Q2

```
1 declare namespace music = "http://www.example.org/music/records";
2
3 <Q2>
4 {
5   doc("auction.xml")//music:title
6 }
7 </Q2>
```

A.1.3 Q3

```
1 declare namespace dt = "http://www.w3.org/2001/XMLSchema";
2
3 <Q3>
4 {
5   doc("auction.xml")//*[@dt:*]
6 }
7 </Q3>
```

A.1.4 Q4

```
1 declare namespace xlink = "http://www.w3.org/1999/xlink";
2
3 <Q4 xmlns:xlink="http://www.w3.org/1999/xlink">
4 {
5   for $hr in doc("auction.xml")//@xlink:href
6     return <ns>{ $hr }</ns>
7 }
8 </Q4>
```

A.1.5 Q5

A W3 Use Cases

```
1 declare namespace music = "http://www.example.org/music/records";
2
3 <Q5 xmlns:music="http://www.example.org/music/records">
4   {
5     doc("auction.xml")//music:record[music:remark/@xml:lang = "de"]
6   }
7 </Q5>
```

A.1.6 Q6

```
1 declare namespace ma = "http://www.example.com/AuctionWatch";
2 declare namespace anyzone = "http://www.example.com/auctioneers#anyzone";
3
4 <Q6 xmlns:ma="http://www.example.com/AuctionWatch">
5   {
6     doc("auction.xml")//ma:Auction[@anyzone:ID]/ma:Schedule/ma:Close
7   }
8 </Q6>
```

A.1.7 Q7

```
1 declare namespace ma = "http://www.example.com/AuctionWatch";
2
3 <Q7 xmlns:xlink="http://www.w3.org/1999/xlink">
4   {
5     for $a in doc("auction.xml")//ma:Auction
6     let $seller_id := $a/ma:Trading_Partners/ma:Seller/*:ID,
7         $buyer_id := $a/ma:Trading_Partners/ma:High_Bidder/*:ID
8     where namespace-uri($seller_id) = namespace-uri($buyer_id)
9     return
10      $a/ma:AuctionHomepage
11   }
12 </Q7>
```

A.1.8 Q8

```
1 declare namespace ma = "http://www.example.com/AuctionWatch";
2
3 <Q8 xmlns:ma="http://www.example.com/AuctionWatch"
4     xmlns:eachbay="http://www.example.com/auctioneers#eachbay"
5     xmlns:xlink="http://www.w3.org/1999/xlink">
6   {
7     for $s in doc("auction.xml")//ma:Trading_Partners/(ma:Seller | ma:High_Bidder)
8     where $s/*:NegativeComments = 0
9     return $s
10  }
11 </Q8>
```

A.2 PARTS

A.2.1 Q1

```
1 declare function local:one_level($p as element()) as element()
2 {
3   <part partid="{ $p/@partid }"
4     name="{ $p/@name }" >
5     {
6       for $s in doc("partlist.xml")//part
7       where $s/@partof = $p/@partid
8       return local:one_level($s)
9     }
10  </part>
11 };
```

```

12|
13| <parttree>
14| {
15|   for $p in doc("partlist.xml")//part[empty(@partof)]
16|     return local:one_level($p)
17| }
18| </parttree>

```

A.3 R

A.3.1 Q1

```

1| <result>
2| {
3|   for $i in doc("items.xml")//item_tuple
4|     where $i/start_date <= current-date()
5|       and $i/end_date >= xs:date("1999-03-31")
6|       and contains($i/description, "Bicycle")
7|     order by $i/itemno
8|     return
9|       <item_tuple>
10|        { $i/itemno }
11|        { $i/description }
12|      </item_tuple>
13| }
14| </result>

```

A.3.2 Q2

```

1| <result>
2| {
3|   for $i in doc("items.xml")//item_tuple
4|     let $b := doc("bids.xml")//bid_tuple[itemno = $i/itemno]
5|     where contains($i/description, "Bicycle")
6|     order by $i/itemno
7|     return
8|       <item_tuple>
9|        { $i/itemno }
10|        { $i/description }
11|        <high_bid>{ max($b/bid) }</high_bid>
12|      </item_tuple>
13| }
14| </result>

```

A.3.3 Q3

```

1| <result>
2| {
3|   for $u in doc("users.xml")//user_tuple
4|   for $i in doc("items.xml")//item_tuple
5|     where $u/rating > "C"
6|       and $i/reserve_price > 1000
7|       and $i/offered_by = $u/userid
8|     return
9|       <warning>
10|        { $u/name }
11|        { $u/rating }
12|        { $i/description }
13|        { $i/reserve_price }
14|      </warning>
15| }
16| </result>

```

A.3.4 Q4

```

1 <result>
2 {
3   for $i in doc("items.xml")//item_tuple
4     where empty(doc("bids.xml")//bid_tuple[itemno = $i/itemno])
5     return
6       <no_bid_item>
7         { $i/itemno }
8         { $i/description }
9       </no_bid_item>
10 }
11 </result>

```

A.3.5 Q5

```

1 <result>
2 {
3   for $seller in doc("users.xml")//user_tuple ,
4     $buyer in doc("users.xml")//user_tuple ,
5     $item in doc("items.xml")//item_tuple ,
6     $highbid in doc("bids.xml")//bid_tuple
7   where $seller/name = "Tom Jones"
8     and $seller/userid = $item/offered_by
9     and contains($item/description , "Bicycle")
10    and $item/itemno = $highbid/itemno
11    and $highbid/userid = $buyer/userid
12    and $highbid/bid = max(
13      doc("bids.xml")//bid_tuple
14        [itemno = $item/itemno]/bid
15    )
16   order by ($item/itemno)
17   return
18     <jones_bike>
19       { $item/itemno }
20       { $item/description }
21       <high_bid>{ $highbid/bid }</high_bid>
22       <high_bidder>{ $buyer/name }</high_bidder>
23     </jones_bike>
24 }
25 </result>

```

A.3.6 Q6

```

1 <result>
2 {
3   for $item in doc("items.xml")//item_tuple
4     let $b := doc("bids.xml")//bid_tuple[itemno = $item/itemno]
5     let $z := max($b/bid)
6     where $item/reserve_price * 2 < $z
7     return
8       <successful_item>
9         { $item/itemno }
10        { $item/description }
11        { $item/reserve_price }
12        <high_bid>{$z }</high_bid>
13      </successful_item>
14 }
15 </result>

```

A.3.7 Q7

```

1 let $allbikes := doc("items.xml")//item_tuple
2               [contains(description , "Bicycle")
3                or contains(description , "Tricycle")]
4 let $bikebids := doc("bids.xml")//bid_tuple[itemno = $allbikes/itemno]
5 return
6   <high_bid>
7     {
8       max($bikebids/bid)

```

```

9 |     }
10 | </high_bid>

```

A.3.8 Q8

```

1 | let $item := doc("items.xml")//item_tuple
2 | [end_date >= xs:date("1999-03-01") and end_date <= xs:date("1999-03-31")]
3 | return
4 |   <item_count>
5 |     {
6 |       count($item)
7 |     }
8 |   </item_count>

```

A.3.9 Q9

```

1 | <result>
2 | {
3 |   let $end_dates := doc("items.xml")//item_tuple/end_date
4 |   for $m in distinct-values(for $e in $end_dates
5 |                             return month-from-date($e))
6 |   let $item := doc("items.xml")
7 |               //item_tuple[year-from-date(end_date) = 1999
8 |                             and month-from-date(end_date) = $m]
9 |   order by $m
10 |  return
11 |    <monthly_result>
12 |      <month>{ $m }</month>
13 |      <item_count>{ count($item) }</item_count>
14 |    </monthly_result>
15 |  }
16 | </result>

```

A.3.10 Q10

```

1 | <result>
2 | {
3 |   for $highbid in doc("bids.xml")//bid_tuple ,
4 |   $user in doc("users.xml")//user_tuple
5 |   where $user/userid = $highbid/userid
6 |   and $highbid/bid = max(doc("bids.xml")//bid_tuple[itemno=$highbid/itemno]/bid)
7 |   order by $highbid/itemno
8 |   return
9 |     <high_bid>
10 |      { $highbid/itemno }
11 |      { $highbid/bid }
12 |      <bidder>{ $user/name/text() }</bidder>
13 |    </high_bid>
14 |  }
15 | </result>

```

A.3.11 Q11

```

1 | let $highbid := max(doc("bids.xml")//bid_tuple/bid)
2 | return
3 |   <result>
4 |     {
5 |       for $item in doc("items.xml")//item_tuple ,
6 |       $b in doc("bids.xml")//bid_tuple[itemno = $item/itemno]
7 |       where $b/bid = $highbid
8 |       return
9 |         <expensive_item>
10 |          { $item/itemno }
11 |          { $item/description }
12 |          <high_bid>{ $highbid }</high_bid>
13 |        </expensive_item>
14 |      }
15 |   </result>

```

A.3.12 Q12

```

1 declare function local:bid-summary()
2   as element()*
3 {
4   for $i in distinct-values(doc("bids.xml")//itemno)
5   let $b := doc("bids.xml")//bid-tuple[itemno = $i]
6   return
7     <bid-count>
8       <itemno>{ $i }</itemno>
9       <nbids>{ count($b) }</nbids>
10      </bid-count>
11 };
12
13 <result>
14 {
15   let $bid-counts := local:bid-summary(),
16       $maxbids := max($bid-counts/nbids),
17       $maxitemnos := $bid-counts[nbids = $maxbids]
18   for $item in doc("items.xml")//item-tuple,
19       $bc in $bid-counts
20   where $bc/nbids = $maxbids and $item/itemno = $bc/itemno
21   return
22     <popular-item>
23       { $item/itemno }
24       { $item/description }
25       <bid-count>{ $bc/nbids/text() }</bid-count>
26     </popular-item>
27 }
28 </result>

```

A.3.13 Q13

```

1 <result>
2 {
3   for $userid in distinct-values(doc("bids.xml")//userid),
4       $u in doc("users.xml")//user-tuple[userid = $userid]
5   let $b := doc("bids.xml")//bid-tuple[userid = $userid]
6   order by $u/userid
7   return
8     <bidder>
9       { $u/userid }
10      { $u/name }
11      <bidcount>{ count($b) }</bidcount>
12      <avgbid>{ avg($b/bid) }</avgbid>
13    </bidder>
14 }
15 </result>

```

A.3.14 Q14

```

1 <result>
2 {
3   for $i in distinct-values(doc("bids.xml")//itemno)
4   let $b := doc("bids.xml")//bid-tuple[itemno = $i]
5   let $avgbid := avg($b/bid)
6   where count($b) >= 3
7   order by $avgbid descending
8   return
9     <popular-item>
10      <itemno>{ $i }</itemno>
11      <avgbid>{ $avgbid }</avgbid>
12    </popular-item>
13 }
14 </result>

```

A.3.15 Q15

A W3 Use Cases

```
1 <result>
2 {
3   for $u in doc("users.xml")//user_tuple
4   let $b := doc("bids.xml")//bid_tuple[user-id=$u/user-id and bid >= 100]
5   where count($b) > 1
6   return
7     <big_spender>{ $u/name/text() }</big_spender>
8 }
9 </result>
```

A.3.16 Q16

```
1 <result>
2 {
3   for $u in doc("users.xml")//user_tuple
4   let $b := doc("bids.xml")//bid_tuple[user-id = $u/user-id]
5   order by $u/user-id
6   return
7     <user>
8       { $u/user-id }
9       { $u/name }
10      {
11        if (empty($b))
12          then <status>inactive</status>
13          else <status>active</status>
14      }
15    </user>
16 }
17 </result>
```

A.3.17 Q17

```
1 <frequent_bidder>
2 {
3   for $u in doc("users.xml")//user_tuple
4   where
5     every $item in doc("items.xml")//item_tuple satisfies
6     some $b in doc("bids.xml")//bid_tuple satisfies
7     ($item/item-no = $b/item-no and $u/user-id = $b/user-id)
8   return
9     $u/name
10 }
11 </frequent_bidder>
```

A.3.18 Q18

```
1 <result>
2 {
3   for $u in doc("users.xml")//user_tuple
4   order by $u/name
5   return
6     <user>
7       { $u/name }
8       {
9         for $b in distinct-values(doc("bids.xml")//bid_tuple
10          [user-id = $u/user-id]/item-no)
11         for $i in doc("items.xml")//item_tuple[item-no = $b]
12         let $descr := $i/description/text()
13         order by $descr
14         return
15           <bid_on_item>{ $descr }</bid_on_item>
16       }
17     </user>
18 }
19 </result>
```

A.4 SEQ

A.4.1 Q1

```

1 for $s in doc("report1.xml")//section[section.title = "Procedure"]
2 return ($s//incision)[2]/instrument

```

A.4.2 Q2

```

1 for $s in doc("report1.xml")//section[section.title = "Procedure"]
2 return ($s//instrument)[position()<=2]

```

A.4.3 Q3

```

1 let $i2 := (doc("report1.xml")//incision)[2]
2 for $a in (doc("report1.xml")//action)[. >> $i2][position()<=2]
3 return $a//instrument

```

A.4.4 Q4

```

1 for $p in doc("report1.xml")//section[section.title = "Procedure"]
2 where not(some $a in $p//anesthesia satisfies
3     $a << ($p//incision)[1] )
4 return $p

```

A.4.5 Q5a

```

1 declare function local:precedes($a as node(), $b as node()) as xs:boolean
2 {
3     $a << $b
4     and
5     empty($a//node() intersect $b)
6 };
7
8
9 declare function local:follows($a as node(), $b as node()) as xs:boolean
10 {
11     $a >> $b
12     and
13     empty($b//node() intersect $a)
14 };
15
16 <critical-sequence>
17 {
18     let $proc := doc("report1.xml")//section[section.title="Procedure"][1]
19     for $n in $proc//node()
20     where local:follows($n, ($proc//incision)[1])
21     and local:precedes($n, ($proc//incision)[2])
22     return $n
23 }
24 </critical-sequence>

```

A.4.6 Q5b

A W3 Use Cases

```
1 <critical_sequence>
2 {
3   let $proc := doc("report1.xml")//section[section.title="Procedure"][1],
4       $i1 := ($proc//incision)[1],
5       $i2 := ($proc//incision)[2],
6   for $n in $proc//node() except $i1//node()
7     where $n >> $i1 and $n << $i2
8     return $n
9 }
10 </critical_sequence>
```

A.4.7 Q5c

```
1 declare function local:between($seq as node()*, $start as node(), $end as node())
2 as item()*
3 {
4   let $nodes :=
5     for $n in $seq except $start//node()
6     where $n >> $start and $n << $end
7     return $n
8   return $nodes except $nodes//node()
9 };
10
11 <critical_sequence>
12 {
13   let $proc := doc("report1.xml")//section[section.title="Procedure"][1],
14       $first := ($proc//incision)[1],
15       $second:= ($proc//incision)[2]
16   return local:between($proc//node(), $first, $second)
17 }
18 </critical_sequence>
```

A.5 SGML

A.5.1 Q1

```
1 <result>
2 {
3   doc("sgml.xml")//report//para
4 }
5 </result>
```

A.5.2 Q2

```
1 <result>
2 {
3   doc("sgml.xml")//intro/para
4 }
5 </result>
```

A.5.3 Q3

```
1 <result>
2 {
3   for $c in doc("sgml.xml")//chapter
4     where empty($c/intro)
5     return $c/section/intro/para
6 }
7 </result>
```

A.5.4 Q4

```
1 <result>
2 {
3   (((doc("sgml.xml")//chapter)[2]//section)[3]//para)[2]
4 }
5 </result>
```

A.5.5 Q5

```
1 <result>
2 {
3   doc("sgml.xml")//para[@security = "c"]
4 }
5 </result>
```

A.5.6 Q6

```
1 <result>
2 {
3   for $s in doc("sgml.xml")//section/@shorttitle
4     return <stitle>{ $s }</stitle>
5 }
6 </result>
```

A.5.7 Q7

```
1 <result>
2 {
3   for $i in doc("sgml.xml")//intro/para[1]
4     return
5       <first_letter>{ substring(string($i), 1, 1) }</first_letter>
6 }
7 </result>
```

A.5.8 Q8a

```
1 <result>
2 {
3   doc("sgml.xml")//section[.//title[contains(., "is SGML")]]
4 }
5 </result>
```

A.5.9 Q8b

```
1 <result>
2 {
3   doc("sgml.xml")//section[.//title/text()[contains(., "is SGML")]]
4 }
5 </result>
```

A.5.10 Q9

```
1 <result>
2 {
3   for $id in doc("sgml.xml")//xref/@xrefid
4     return doc("sgml.xml")//topic[@topicid = $id]
5 }
6 </result>
```

A.5.11 Q10

```

1 <result>
2 {
3   let $x := doc("sgml.xml")//xref[@xrefid = "top4"],
4     $t := doc("sgml.xml")//title[. << $x]
5   return $t[last()]
6 }
7 </result>

```

A.6 STRING

A.6.1 Q1

```

1 doc("string.xml")//news.item/title[contains(., 'Foobar Corporation')]

```

A.6.2 Q2

```

1 declare function local:partners($company as xs:string) as element()*
2 {
3   let $c := doc('dbxml:/D:/Test/xml/dbxml-2.0.9_bin/examples/W3UseCases/STRING/dbEnv/
4     W3ExampleData.dbxml/company-data.xml')//company[name = $company]
5   return $c//partner
6 };
7 let $foobar_partners := local:partners('Foobar Corporation')
8
9 for $item in doc("string.xml")//news.item
10 where
11   some $t in $item//title satisfies
12     (contains($t/text(), 'Foobar Corporation')
13     and (some $partner in $foobar_partners satisfies
14         contains($t/text(), $partner/text())))
15   or (some $par in $item//par satisfies
16       (contains(string($par), 'Foobar Corporation')
17       and (some $partner in $foobar_partners satisfies
18           contains(string($par), $partner/text()))))
19 return
20   <news.item>
21     { $item/title }
22     { $item/date }
23   </news.item>

```

A.6.3 Q4

```

1 declare function local:partners($company as xs:string) as element()*
2 {
3   let $c := doc('dbxml:/D:/Test/xml/dbxml-2.0.9_bin/examples/W3UseCases/STRING/dbEnv/
4     W3ExampleData.dbxml/company-data.xml')//company[name = $company]
5   return $c//partner
6 };
7 for $item in doc("string.xml")//news.item,
8   $c in doc('dbxml:/D:/Test/xml/dbxml-2.0.9_bin/examples/W3UseCases/STRING/dbEnv/
9     W3ExampleData.dbxml/company-data.xml')//company
10 let $partners := local:partners($c/name)
11 where contains(string($item), $c/name)
12   and (some $p in $partners satisfies
13       contains(string($item), $p) and $item/news.agent != $c/name)
14 return
15   $item

```

A.6.4 Q5

```

1 for $item in doc("string.xml")//news_item
2 where contains(string($item/content), "Gorilla Corporation")
3 return
4   <item_summary>
5     { concat($item/title, ". ") }
6     { concat($item/date, ". ") }
7     { string(($item//par)[1]) }
8   </item_summary>

```

A.7 TREE

A.7.1 Q1

```

1 declare function local:toc($book-or-section as element()) as element()*
2 {
3   for $section in $book-or-section/section
4   return
5     <section>
6       { $section/@* , $section/title , local:toc($section) }
7     </section>
8 };
9
10 <toc>
11 {
12   for $s in doc("book.xml")/book return local:toc($s)
13 }
14 </toc>

```

A.7.2 Q2

```

1 <figlist>
2 {
3   for $f in doc("book.xml")//figure
4   return
5     <figure>
6       { $f/@* }
7       { $f/title }
8     </figure>
9 }
10 </figlist>

```

A.7.3 Q3

```

1 <section_count>{ count(doc("book.xml")//section) }</section_count>,
2 <figure_count>{ count(doc("book.xml")//figure) }</figure_count>

```

A.7.4 Q4

```

1 <top_section_count>
2 {
3   count(doc("book.xml")/book/section)
4 }
5 </top_section_count>

```

A.7.5 Q5

```

1 <section-list>
2 {
3   for $s in doc("book.xml")//section
4     let $f := $s/figure
5     return
6       <section title="{ $s/title/text() }" figcount="{ count($f) }"/>
7 }
8 </section-list>

```

A.7.6 Q6

```

1 declare function local:section-summary($book-or-section as element(*)
2   as element(*)
3 {
4   for $section in $book-or-section
5     return
6       <section>
7         { $section/@* }
8         { $section/title }
9         <figcount>
10          { count($section/figure) }
11        </figcount>
12        { local:section-summary($section/section) }
13      </section>
14 };
15
16 <toc>
17 {
18   for $s in doc("book.xml")/book/section
19     return local:section-summary($s)
20 }
21 </toc>

```

A.8 XMP**A.8.1 Q1**

```

1 <bib>
2 {
3   for $b in doc("http://bstore1.example.com/bib.xml")/bib/book
4     where $b/publisher = "Addison-Wesley" and $b/@year > 1991
5     return
6       <book year="{ $b/@year }">
7         { $b/title }
8       </book>
9 }
10 </bib>

```

A.8.2 Q2

```

1 <results>
2 {
3   for $b in doc("http://bstore1.example.com/bib.xml")/bib/book,
4     $t in $b/title,
5     $a in $b/author
6   return
7     <result>
8       { $t }
9       { $a }
10    </result>
11 }
12 </results>

```

A.8.3 Q3

```

1 <results>
2 {
3   for $b in doc("http://bstore1.example.com/bib.xml")/bib/book
4   return
5     <result>
6       { $b/title }
7       { $b/author }
8     </result>
9 }
10 </results>

```

A.8.4 Q4

```

1 <results>
2 {
3   let $a := doc("http://bstore1.example.com/bib/bib.xml")//author
4   for $last in distinct-values($a/last),
5     $first in distinct-values($a[last=$last]/first)
6   order by $last, $first
7   return
8     <result>
9       <author>
10        <last>{ $last }</last>
11        <first>{ $first }</first>
12      </author>
13      {
14        for $b in doc("http://bstore1.example.com/bib.xml")/bib/book
15        where some $ba in $b/author
16          satisfies ($ba/last = $last and $ba/first=$first)
17          return $b/title
18      }
19    </result>
20 }
21 </results>

```

A.8.5 Q5

```

1 <books-with-prices>
2 {
3   for $b in doc("http://bstore1.example.com/bib.xml")//book,
4     $a in doc("http://bstore2.example.com/reviews.xml")//entry
5   where $b/title = $a/title
6   return
7     <book-with-prices>
8       { $b/title }
9       <price-bstore2>{ $a/price/text() }</price-bstore2>
10      <price-bstore1>{ $b/price/text() }</price-bstore1>
11    </book-with-prices>
12 }
13 </books-with-prices>

```

A.8.6 Q6

```

1 <bib>
2 {
3   for $b in doc("http://bstore1.example.com/bib.xml")//book
4   where count($b/author) > 0
5   return
6     <book>
7       { $b/title }
8       {
9         for $a in $b/author[position()<=2]
10        return $a
11      }
12     {
13       if (count($b/author) > 2)
14       then <et-al/>
15       else ()

```

A W3 Use Cases

```
16     }
17     </book>
18   }
19 </bib>
```

A.8.7 Q7

```
1 <bib>
2 {
3   for $b in doc("http://bstore1.example.com/bib.xml")//book
4   where $b/publisher = "Addison-Wesley" and $b/@year > 1991
5   order by $b/title
6   return
7     <book>
8       { $b/@year }
9       { $b/title }
10    </book>
11  }
12 </bib>
```

A.8.8 Q8

```
1 for $b in doc("http://bstore1.example.com/bib.xml")//book
2 let $e := $b/*[contains(string(.), "Suciu")
3   and ends-with(local-name(.), "or")]
4 where exists($e)
5 return
6   <book>
7     { $b/title }
8     { $e }
9   </book>
```

A.8.9 Q9

```
1 <results>
2 {
3   for $t in doc("books.xml")//(chapter | section)/title
4   where contains($t/text(), "XML")
5   return $t
6 }
7 </results>
```

A.8.10 Q10

```
1 <results>
2 {
3   let $doc := doc("prices.xml")
4   for $t in distinct-values($doc//book/title)
5   let $p := $doc//book[title = $t]/price
6   return
7     <minprice title="{ $t }">
8       <price>{ min($p) }</price>
9     </minprice>
10 }
11 </results>
```

A.8.11 Q11

A W3 Use Cases

```
1 <bib>
2 {
3     for $b in doc("http://bstore1.example.com/bib.xml")//book[author]
4     return
5         <book>
6             { $b/title }
7             { $b/author }
8         </book>
9     }
10 {
11     for $b in doc("http://bstore1.example.com/bib.xml")//book[editor]
12     return
13         <reference>
14             { $b/title }
15             { $b/editor/affiliation }
16         </reference>
17     }
18 </bib>
```

A.8.12 Q12

```
1 <bib>
2 {
3     for $book1 in doc("http://bstore1.example.com/bib.xml")//book,
4     $book2 in doc("http://bstore1.example.com/bib.xml")//book
5     let $aut1 := for $a in $book1/author
6                 order by $a/last, $a/first
7                 return $a
8     let $aut2 := for $a in $book2/author
9                 order by $a/last, $a/first
10                return $a
11     where $book1 << $book2
12     and not($book1/title = $book2/title)
13     and deep-equal($aut1, $aut2)
14     return
15         <book-pair>
16             { $book1/title }
17             { $book2/title }
18         </book-pair>
19     }
20 </bib>
```

B Quellcode

B.1 xmldb218

B.1.1 exampleLoadContainer.java

```
1 package examples;
2
3 import java.io.*;
4 import java.util.*;
5
6 import com.sleepycat.db.*;
7 import com.sleepycat.dbxml.*;
8
9 class ExampleLoadContainer
10 {
11
12     static long startTime = System.currentTimeMillis();
13
14     private static void usage() {
15         String usageMessage = "\nThis program loads XML files into the a container called
16             ExampleData.dbxml.\n";
17         usageMessage += "It's suitable for large files. \n";
18         usageMessage += "It will delete existing containers in the target dir. \n";
19         usageMessage += "Provide the directory where you want to place your database
20             environment, \n";
21         usageMessage += "and the path to the xmlData directory (this exists in your DB XML
22             examples directory).\n";
23         usageMessage += "Also provide the cache size in megabyte (some power of 2).\n\n";
24         usageMessage += "\t-e <dbenv directory> -p <filepath> -c <cache size in megabyte>\n";
25         usageMessage += "For example:\n";
26         usageMessage += "\t-e /home/user1/dbxml-1.1.0/examples/dbEnv -p /home/user1/dbxml
27             -1.1.0/examples/xmlData -c 256\n";
28         System.out.println(usageMessage);
29         System.exit(-1);
30     }
31
32     public static void main(String args[])
33     throws Throwable {
34
35         File path2DbEnv = null;
36         String theContainer;
37         File xmlFilePath = null;
38         Integer cacheSize = new Integer(0);
39
40         int i2 = 0;
41         for(int i = 0; i < args.length; ++i) {
42             if (args[i].startsWith("-")) {
43                 switch(args[i].charAt(1)) {
44                     case 'e':
45                         path2DbEnv = new File(args[++i]);
46                         i2++;
47                         break;
48                     case 'c':
49                         cacheSize = new Integer(args[++i]);
50                         i2++;
51                         break;
52                     case 'p':
53                         xmlFilePath = new File(args[++i]);
54                         i2++;
55                         break;
56                     default:
57                         usage();
58                 }
59             }
60         }
```

B Quellcode

```
61     }
62     if (i2 != 3) {
63         usage();
64     }
65
66     if (path2DbEnv == null || xmlFilePath == null) {
67         usage();
68     }
69
70     if (! xmlFilePath.isDirectory()) {
71         usage();
72     }
73
74     // This vector will hold a File object for each XML file that we will load
75     // into the examples container
76     List files2Add = new LinkedList();
77
78     confirmDirectory(xmlFilePath);
79     deletePreviousContainer(path2DbEnv);
80
81     //Load the first set of examples xml files into our vector
82     getXmlFiles(xmlFilePath, files2Add);
83     //Add these files to the namespace container.
84
85     loadXmlFiles(path2DbEnv, "ExampleData.dbxml", files2Add, cacheSize);
86
87     files2Add.clear();
88 } // End method main()
89
90 //Convenience method used to make sure -p points to a directory that exists
91 private static void confirmDirectory(File directory) {
92     if (! directory.isDirectory()) {
93         System.out.println( "\nError. Directory " + directory.getPath() +
94             " does not exist.");
95         System.out.println( "      -p must point to the xmlData directory.");
96         usage();
97     }
98 }
99
100 private static void deletePreviousContainer(File pathToDbEnv) {
101     try {
102         XmlManager theMgr = new XmlManager();
103         theMgr.removeContainer(pathToDbEnv.getAbsolutePath() + "/ExampleData.dbxml");
104     }
105     catch (Exception e){}
106
107 }
108
109 //Find all the xml files in a specified directory and store them in a vector
110 private static void getXmlFiles(File filePath, List files2add) {
111     boolean filesFound = false;
112     String [] dirContents = filePath.list();
113     if (dirContents != null) {
114         for (int i = 0; i < dirContents.length; i++) {
115             File entry = new File(filePath + File.separator + dirContents[i]);
116             if (entry.isFile() && entry.toString().toLowerCase().endsWith(".xml")) {
117                 files2add.add(entry);
118                 filesFound = true;
119             }
120         }
121     }
122
123     if (! filesFound) {
124         System.out.println("\nError: No XML files found at " +
125             filePath.getPath());
126         usage();
127     }
128 }
129
130 //Utility function to clean up objects, exceptions or not
131 // XmlContainer and XmlManager objects must be closed.
132 private static void cleanup(XmlManager theMgr, XmlContainer openedContainer) {
133     try {
134         if (openedContainer != null)
135             openedContainer.close();
136         if (theMgr != null)
137             theMgr.close();
138     } catch (Exception e) {
139         // ignore exceptions on close
140     }
141 }
142
143 // create an environment. Will throw if home doesn't exist
```

B Quellcode

```
145 private static Environment createEnv(File home, Integer cacheSize)
146     throws DatabaseException, FileNotFoundException {
147
148
149     EnvironmentConfig config = new EnvironmentConfig();
150     config.setCacheSize(cacheSize.intValue() * 1024 * 1024);
151     config.setAllowCreate(true);
152     config.setInitializeCache(true);
153     config.setTransactional(true);
154     config.setInitializeLocking(true);
155     config.setInitializeLogging(false);
156     return new Environment(home, config);
157 }
158
159 //Take a vector of Files and load each element into a DB XML container
160 private static void loadXmlFiles(File path2DbEnv, String theContainer,
161     List files2add, Integer cacheSize)
162     throws Throwable {
163     //Open a container in the db environment
164     XmlManager theMgr = null;
165     XmlContainer openedContainer = null;
166     XmlTransaction txn = null;
167     Environment env = null;
168     try {
169         env = createEnv(path2DbEnv, cacheSize);
170
171         //create the XmlManager
172         XmlManagerConfig theMgrConfig = new XmlManagerConfig();
173         theMgrConfig.setAllowExternalAccess(false);
174         theMgr = new XmlManager(env, theMgrConfig);
175
176         // create a transactional container
177         XmlContainerConfig config = new XmlContainerConfig();
178         config.setTransactional(true);
179         openedContainer = theMgr.createContainer(theContainer, config);
180
181         // Get an update context.
182         XmlUpdateContext updateContext = theMgr.createUpdateContext();
183         // Get another transaction, via DB. This just
184         // demonstrates that a Transaction created from DB can be
185         // passed to XmlManager.createTransaction.
186         Transaction dbtxn = env.beginTransaction(null, null);
187         txn = theMgr.createTransaction(dbtxn);
188         Iterator filesIterator = files2add.iterator();
189         while(filesIterator.hasNext()) {
190             File file = (File) filesIterator.next();
191             System.out.println("Trying to add: " + file.getAbsolutePath());
192             String theFile = file.toString();
193
194             XmlInputStream myXmlInputStream = theMgr.
195                 createLocalFileInputStream(file.getAbsolutePath());
196
197             openedContainer.putDocument(txn, file.getName(),
198                 myXmlInputStream, updateContext);
199
200             //openedContainer.putDocument(file.getName(),
201                 myXmlInputStream, updateContext);
202             /*
203             //Load the contents of the XML file into a String
204
205             String theLine = null;
206             String xmlString = new String();
207             FileInputStream fis = new FileInputStream(theFile);
208             BufferedReader br = new BufferedReader(new
209                 InputStreamReader(fis));
210             StringBuffer sb = new StringBuffer();
211             String s;
212             while((s = br.readLine()) != null) {
213                 sb.append(s);
214                 sb.append("\n");
215             }
216             br.close();
217             xmlString = sb.toString();
218
219             //Declare an xml document
220             XmlDocument xmlDoc = theMgr.createDocument();
221             //Set the xml document's content to the xmlString we
222             just obtained.
223             xmlDoc.setContent(xmlString);
224
225             //Set the document name
226             xmlDoc.setName(file.getName());
```

B Quellcode

```
223         Date theDate = new Date();
224         xmlDoc.setMetaData(mdConst.uri, mdConst.name,
225                             new XmlValue(theDate.toString()));
226
227         //Place that document into the container
228         openedContainer.putDocument(txn, xmlDoc, updateContext,
229                                     0);
230
231         */
232
233         System.out.println("Added " + theFile + " to container
234                             " +
235                             theContainer);
236     }
237     //txn.commit();
238     //XmlException extends DatabaseException, which in turn extends
239     //Exception.
240     // Catching Exception catches them all.
241 } catch (Exception e) {
242
243     System.err.println("Error loading files into container " +
244                         theContainer);
245     System.err.println("  Message: " + e.getMessage());
246     //In the event of an error, we abort the operation
247     // The database is left in the same state as it was in before
248     // we started this operation.
249     if ( txn != null ) {
250         txn.abort();
251     }
252     throw e;
253 } finally {
254     cleanup(theMgr, openedContainer);
255     System.out.println("Seconds needed: " + (System.currentTimeMillis()
256                                             - startTime) / 1000 );
257 }
```

B.1.2 myDbEnv.java

```
1 //
2 // See the file LICENSE for redistribution information.
3 //
4 // Copyright (c) 2004-2005
5 // Sleepycat Software. All rights reserved.
6 //
7 // $Id: myDbEnv.java,v 1.8 2005/04/05 16:43:50 bostic Exp $
8 //
9
10 package examples;
11
12 import java.io.*;
13 import com.sleepycat.db.*;
14 import com.sleepycat.dbxml.*;
15
16 //Class used to open and close a Berkeley DB environment
17 public class myDbEnv
18 {
19     private Environment dbEnv_ = null;
20     private XmlManager mgr_ = null;
21     private boolean dbEnvIsOpen_ = false;
22     private File path2DbEnv_ = null;
23
24     public myDbEnv(File path2DbEnv)
25     throws Throwable {
26         if (! path2DbEnv.isDirectory()) {
27             throw new Exception(path2DbEnv.getPath() +
28                                 " does not exist or is not a directory.");
29         }
30
31         EnvironmentConfig config = new EnvironmentConfig();
32         config.setCacheSize(50 * 1024 * 1024);
33         config.setAllowCreate(true);
34         config.setInitializeCache(true);
35         config.setTransactional(true);
36         config.setInitializeLocking(true);
37         config.setInitializeLogging(true);
38     }
39 }
```

B Quellcode

```
38     config.setErrorStream(System.err);
39     dbEnv_ = new Environment(path2DbEnv, config);
40
41     //Boolean used to know whether to close the environment
42     //when the cleanup() method is called.
43     dbEnvIsOpen_ = true;
44     path2DbEnv_ = path2DbEnv;
45     mgr_ = new XmlManager(dbEnv_, null);
46 }
47
48 //Returns the path to the database environment
49 public File getDbEnvPath() { return path2DbEnv_; }
50
51 //Returns the database environment encapsulated by this class.
52 public Environment getEnvironment() { return dbEnv_; }
53
54 //Returns the XmlManager encapsulated by this class.
55 public XmlManager getManager() { return mgr_; }
56
57 //Used to close the environment
58 public void cleanup() throws DatabaseException
59 {
60     if (dbEnvIsOpen_) {
61         dbEnv_.close();
62         dbEnvIsOpen_ = false;
63     }
64 }
65 }
```

B.1.3 XQuery_W3.java

```
1 package examples;
2 import com.sleepycat.dbxml.XmlContainer;
3 import com.sleepycat.dbxml.XmlManager;
4 import com.sleepycat.dbxml.XmlQueryContext;
5 import com.sleepycat.dbxml.XmlQueryExpression;
6 import com.sleepycat.dbxml.XmlResults;
7 import com.sleepycat.dbxml.XmlValue;
8 import java.io.*;
9 import java.util.*;
10
11 class XQuery_W3 {
12
13     private static void usage() {
14         String usageMessage = "\nThis program runs specified queries fetching data from a
15             specified container.\n";
16         usageMessage += "Provide the path to the container, and the path of the query files. The
17             must be names *.query \n";
18         usageMessage += "Provide whether you are running Xmark or the W3C Use Cases";
19
20         usageMessage += "\n-e <path to environment> -p <path to query files> -k <Xmark/W3C>\n";
21         usageMessage += "For example:\n";
22         usageMessage += "\t-e D:/Test/xml/xmark/01/dbEnv -q D:/Test/xml/xmark/queries -k W3C\n";
23
24         System.out.println(usageMessage);
25         System.exit(-1);
26     }
27
28     public static void main(String args[]) {
29         XmlManager myManager = null;
30         XmlContainer myContainer = null;
31         XmlQueryExpression qe = null;
32         XmlResults results = null;
33         XmlValue value = null;
34         String containerPath = null;
35         int executionNumber = 0;
36         float firstExecutionTime = 0;
37         File pathToQueries = null;
38         String kindOfUseCase = null;
39
40
41         int i = 0;
42         int i2 = 0;
43         for(i = 0; i < args.length; ++i) {
44             if (args[i].startsWith("-")) {
45                 switch(args[i].charAt(1)) {
46                     case 'e':
```

B Quellcode

```

47                                     containerPath = args[++
48                                     i];
49                                     containerPath += "/"
50                                     ExampleData.dbxml"
51                                     ;
52                                     i2++;
53                                     break;
54                                     case 'q':
55                                     pathToQueries = new
56                                     File(args[++i]);
57                                     i2++;
58                                     break;
59                                     case 'k':
60                                     kindOfUseCase = args[++i];
61                                     i2++;
62                                     break;
63                                     default:
64                                     usage();
65                                     }
66                                     }
67                                     }
68                                     if (i2 != 3) {
69                                     usage();
70                                     }
71                                     // So check to make sure they exist.
72                                     File queryDir = new File(pathToQueries.getPath());
73                                     confirmDirectory(queryDir);
74                                     //Load the first set of examples xml files into our vector
75                                     List files2add = new LinkedList();
76                                     getQueryFiles(queryDir, files2add);
77                                     try {
78                                     // Get a manager object.
79                                     myManager = new XmlManager();
80                                     // Open a container
81                                     myContainer = myManager.openContainer(containerPath);
82                                     System.out.println("Opened container " + containerPath);
83                                     // Get a query context
84                                     XmlQueryContext context = myManager.createQueryContext();
85
86
87                                     Iterator filesIterator = files2add.iterator();
88                                     while(filesIterator.hasNext()) {
89                                     try {
90                                     File file = (File) filesIterator.next();
91                                     String myQuery = "";
92                                     FileInputStream textFileIn = new FileInputStream(file .
93                                     getAbsolutePath());
94                                     InputStreamReader textFileReader = new InputStreamReader(
95                                     textFileIn);
96                                     BufferedReader textFileLineReader = new BufferedReader(
97                                     textFileReader);
98                                     String line;
99                                     while ( (line = textFileLineReader.readLine()) != null)
100                                     {
101                                     if (kindOfUseCase.toLowerCase().equals("xmark")) {
102                                     line = regexXmark(line, containerPath);
103                                     } else if (kindOfUseCase.toLowerCase().equals("w3c")){
104                                     line = regexW3UseCases(line, containerPath);
105                                     }
106                                     myQuery += line;
107                                     myQuery += "\n";
108                                     };
109                                     System.out.println("executing query " + file .
110                                     getAbsolutePath());
111                                     System.out.println("query runs " + executionNumber + "
112                                     times and is:");
113                                     System.out.println(myQuery);
114                                     // myQuery = "";
115                                     //myQuery += "for $b in doc('dbxml:D:/Test/xml/xmark/00/
116                                     dbEnv/ExampleData.dbxml/auction.xml')/site\n";
117                                     //myQuery += "return $b/regions\n";
118
119                                     qe = myManager.prepare(myQuery, context);
120                                     for (i = 1; i <= executionNumber; i++){
121                                     results = qe.execute(context);
122                                     }

```

B Quellcode

```

120         File resultFile = new File(file.getAbsolutePath() + ".
121             res" );
122         BufferedWriter outputResultFile = new BufferedWriter( new
123             FileWriter(resultFile) );
124         System.out.println();
125         while ((value = results.next()) != null){
126             outputResultFile.append( value.asString());
127             System.out.println(value.asString());
128         }
129         System.out.println();
130         outputResultFile.close();
131         System.out.println("-----");
132     } catch (Exception e) {
133         System.err.println(e);
134     }
135 }
136
137 } catch (Exception e) {
138     System.err.println(e);
139 } finally {
140     try {
141         if (myContainer != null) {
142             myContainer.close();
143         }
144         if (myManager != null) {
145             myManager.close();
146         }
147     } catch (Exception e) {
148         System.err.println(e);
149     }
150 }
151 System.out.println("finished.");
152 }
153 }
154
155
156 //Convenience method used to make sure -q points to a directory that exists
157 private static void confirmDirectory(File directory) {
158     if (! directory.isDirectory() ) {
159         System.out.println( "\nError. Directory " + directory.getPath() +
160             " does not exist.");
161         System.out.println( "      -q must point to the xmlData directory.");
162         usage();
163     }
164 }
165
166 private static void getQueryFiles(File filePath , List files2add) {
167     boolean filesFound = false;
168     String [] dirContents = filePath.list();
169     if (dirContents != null) {
170         for (int i = 0; i < dirContents.length; i++) {
171             File entry = new File(filePath + File.separator + dirContents[i]);
172             if (entry.isFile() && entry.toString().toLowerCase().endsWith(".query"))
173                 {
174                     files2add.add(entry);
175                     filesFound = true;
176                 }
177         }
178     }
179     if (! filesFound) {
180         System.out.println("\nError: No *.query files found at " + filePath.getPath
181             ());
182         usage();
183     }
184 }
185
186 private static String regexW3UseCases(String line , String containerPath) {
187     line = line.replaceAll(new String("\\"), new String("'"));
188     line = line.replaceAll(new String("doc\\('"), new String("doc\\('dbxml:" +
189         containerPath + '/''));
190     return line;
191 }
192
193 private static String regexXmark(String line , String containerPath) {
194     line = line.toLowerCase();
195     line = line.replaceAll(new String("\\"), new String("'"));
196     line = line.replaceAll(new String("doc\\('"), new String("doc\\('dbxml:" +
197         containerPath + '/''));
198     return line;
199 }

```

198 }

B.2 Batch-Skripte DB2

B.2.1 NS.bat

```
1 @ECHO OFF
2 ECHO.
3
4 ECHO CREATE DATABASE:
5 ECHO _____
6 db2 create database wc3 USING CODESET UTF-8 TERRITORY US
7 ECHO.
8
9 ECHO CONNECT TO DATABASE:
10 ECHO _____
11 db2 connect to wc3
12 ECHO.
13
14 ECHO CREATE TABLES:
15 ECHO _____
16 db2 create table NS_AUCTION (XMLDOC XML)
17 ECHO.
18
19 ECHO IMPORT DATA INTO DATABASE:
20 ECHO _____
21 db2 import from auction.del of del xml from . insert into NS_AUCTION
22 ECHO.
23
24 ECHO QUERY DATABASE:
25 ECHO _____
26 db2 -tvf q01.sql > q01.res
27 db2 -tvf q02.sql > q02.res
28 db2 -tvf q03.sql > q03.res
29 db2 -tvf q04.sql > q04.res
30 db2 -tvf q05.sql > q05.res
31 db2 -tvf q06.sql > q06.res
32 db2 -tvf q07.sql > q07.res
33 db2 -tvf q08.sql > q08.res
34 ECHO Results are saved in the *.res files!
35 ECHO.
36
37 ECHO DISCONNECT FROM DATABASE:
38 ECHO _____
39 db2 terminate
40 ECHO.
41
42 ECHO DROP DATABASE:
43 ECHO _____
44 db2 drop database wc3
45 ECHO.
```

B.2.2 PARTS.bat

```
1 @ECHO OFF
2 ECHO.
3
4 ECHO CREATE DATABASE:
5 ECHO _____
6 db2 create database wc3 USING CODESET UTF-8 TERRITORY US
7 ECHO.
8
9 ECHO CONNECT TO DATABASE:
10 ECHO _____
11 db2 connect to wc3
12 ECHO.
13
14 ECHO CREATE TABLES:
15 ECHO _____
16 db2 create table PARTS_PARTLIST (XMLDOC XML)
17 ECHO.
18
19 ECHO IMPORT DATA INTO DATABASE:
```

B Quellcode

```
20 ECHO -----
21 db2 import from partlist.del of del xml from . insert into PARTS_PARTLIST
22
23 ECHO.
24
25 ECHO QUERY DATABASE:
26 ECHO -----
27 db2 -tvf q01.sql > q01.res
28 ECHO Results are saved in the *.res files!
29 ECHO.
30
31 ECHO DISCONNECT FROM DATABASE:
32 ECHO -----
33 db2 terminate
34 ECHO.
35
36 ECHO DROP DATABASE:
37 ECHO -----
38 db2 drop database wc3
39 ECHO.
```

B.2.3 R.bat

```
1 @ECHO OFF
2 ECHO.
3
4 ECHO CREATE DATABASE:
5 ECHO -----
6 db2 create database wc3 USING CODESET UTF-8 TERRITORY US
7 ECHO.
8
9 ECHO CONNECT TO DATABASE:
10 ECHO -----
11 db2 connect to wc3
12 ECHO.
13
14 ECHO CREATE TABLES:
15 ECHO -----
16 db2 create table R_BIDS (XMLDOC XML)
17 db2 create table R_ITEMS (XMLDOC XML)
18 db2 create table R_USERS (XMLDOC XML)
19 ECHO.
20
21 ECHO IMPORT DATA INTO DATABASE:
22 ECHO -----
23 db2 import from bids.del of del xml from . insert into R_BIDS
24 db2 import from items.del of del xml from . insert into R_ITEMS
25 db2 import from users.del of del xml from . insert into R_USERS
26 ECHO.
27
28 ECHO QUERY DATABASE:
29 ECHO -----
30 db2 -tvf q01.sql > q01.res
31 db2 -tvf q02.sql > q02.res
32 db2 -tvf q03.sql > q03.res
33 db2 -tvf q04.sql > q04.res
34 db2 -tvf q05.sql > q05.res
35 db2 -tvf q06.sql > q06.res
36 db2 -tvf q07.sql > q07.res
37 db2 -tvf q08.sql > q08.res
38 db2 -tvf q09.sql > q09.res
39 db2 -tvf q10.sql > q10.res
40 db2 -tvf q11.sql > q11.res
41 db2 -tvf q12.sql > q12.res
42 db2 -tvf q13.sql > q13.res
43 db2 -tvf q14.sql > q14.res
44 db2 -tvf q15.sql > q15.res
45 db2 -tvf q16.sql > q16.res
46 db2 -tvf q17.sql > q17.res
47 db2 -tvf q18.sql > q18.res
48 ECHO Results are saved in the *.res files!
49 ECHO.
50
51 ECHO DISCONNECT FROM DATABASE:
52 ECHO -----
53 db2 terminate
54 ECHO.
55
56 ECHO DROP DATABASE:
57 ECHO -----
```

B Quellcode

```
58 db2 drop database wc3
59 ECHO.
```

B.2.4 SEQ.bat

```
1 @ECHO OFF
2 ECHO.
3
4 ECHO CREATE DATABASE:
5 ECHO _____
6 db2 create database wc3 USING CODESET UTF-8 TERRITORY US
7 ECHO.
8
9 ECHO CONNECT TO DATABASE:
10 ECHO _____
11 db2 connect to wc3
12 ECHO.
13
14 ECHO CREATE TABLES:
15 ECHO _____
16 db2 create table SEQ_REPORT (XMLDOC XML)
17 ECHO.
18
19 ECHO IMPORT DATA INTO DATABASE:
20 ECHO _____
21 db2 import from report.del of del xml from . insert into SEQ_REPORT
22 ECHO.
23
24 ECHO QUERY DATABASE:
25 ECHO _____
26 db2 -tvf q01.sql > q01.res
27 db2 -tvf q02.sql > q02.res
28 db2 -tvf q03.sql > q03.res
29 db2 -tvf q04.sql > q04.res
30 db2 -tvf q05a.sql > q05a.res
31 db2 -tvf q05b.sql > q05b.res
32 db2 -tvf q05c.sql > q05c.res
33 ECHO Results are saved in the *.res files!
34 ECHO.
35
36 ECHO DISCONNECT FROM DATABASE:
37 ECHO _____
38 db2 terminate
39 ECHO.
40
41 ECHO DROP DATABASE:
42 ECHO _____
43 db2 drop database wc3
44 ECHO.
```

B.2.5 SGML.bat

```
1 @ECHO OFF
2 ECHO.
3
4 ECHO CREATE DATABASE:
5 ECHO _____
6 db2 create database wc3 USING CODESET UTF-8 TERRITORY US
7 ECHO.
8
9 ECHO CONNECT TO DATABASE:
10 ECHO _____
11 db2 connect to wc3
12 ECHO.
13
14 ECHO CREATE TABLES:
15 ECHO _____
16 db2 create table SGML (XMLDOC XML)
17 ECHO.
18
19 ECHO IMPORT DATA INTO DATABASE:
20 ECHO _____
21 db2 import from sgml.del of del xml from . insert into SGML
22 ECHO.
23
```

B Quellcode

```
24 ECHO QUERY DATABASE:
25 ECHO _____
26 db2 -tvf q01.sql > q01.res
27 db2 -tvf q02.sql > q02.res
28 db2 -tvf q03.sql > q03.res
29 db2 -tvf q04.sql > q04.res
30 db2 -tvf q05.sql > q05.res
31 db2 -tvf q06.sql > q06.res
32 db2 -tvf q07.sql > q07.res
33 db2 -tvf q08a.sql > q08a.res
34 db2 -tvf q08b.sql > q08b.res
35 db2 -tvf q09.sql > q09.res
36 db2 -tvf q10.sql > q10.res
37 ECHO Results are saved in the *.res files!
38 ECHO.
39
40 ECHO DISCONNECT FROM DATABASE:
41 ECHO _____
42 db2 terminate
43 ECHO.
44
45 ECHO DROP DATABASE:
46 ECHO _____
47 db2 drop database wc3
48 ECHO.
```

B.2.6 STRING.bat

```
1 @ECHO OFF
2 ECHO.
3
4 ECHO CREATE DATABASE:
5 ECHO _____
6 db2 create database wc3 USING CODESET UTF-8 TERRITORY US
7 ECHO.
8
9 ECHO CONNECT TO DATABASE:
10 ECHO _____
11 db2 connect to wc3
12 ECHO.
13
14 ECHO CREATE TABLES:
15 ECHO _____
16 db2 create table STRING_DATA (XMLDOC XML)
17 db2 create table STRING_STRING (XMLDOC XML)
18 ECHO.
19
20 ECHO IMPORT DATA INTO DATABASE:
21 ECHO _____
22 db2 import from data.del of del xml from . insert into STRING_DATA
23 db2 import from string.del of del xml from . insert into STRING_STRING
24 ECHO.
25
26 ECHO QUERY DATABASE:
27 ECHO _____
28 db2 -tvf q01.sql > q01.res
29 db2 -tvf q02.sql > q02.res
30 db2 -tvf q04.sql > q04.res
31 db2 -tvf q05.sql > q05.res
32
33 ECHO Results are saved in the *.res files!
34 ECHO.
35
36 ECHO DISCONNECT FROM DATABASE:
37 ECHO _____
38 db2 terminate
39 ECHO.
40
41 ECHO DROP DATABASE:
42 ECHO _____
43 db2 drop database wc3
44 ECHO.
```

B.2.7 TREE.bat

B Quellcode

```
1 @ECHO OFF
2 ECHO.
3
4 ECHO CREATE DATABASE:
5 ECHO _____
6 db2 create database wc3 USING CODESET UTF-8 TERRITORY US
7 ECHO.
8
9 ECHO CONNECT TO DATABASE:
10 ECHO _____
11 db2 connect to wc3
12 ECHO.
13
14 ECHO CREATE TABLES:
15 ECHO _____
16 db2 create table TREE_BOOK (XMLDOC XML)
17 ECHO.
18
19 ECHO IMPORT DATA INTO DATABASE:
20 ECHO _____
21 db2 import from book.del of del xml from . insert into TREE_BOOK
22 ECHO.
23
24 ECHO QUERY DATABASE:
25 ECHO _____
26 db2 -tvf q01.sql > q01.res
27 db2 -tvf q02.sql > q02.res
28 db2 -tvf q03.sql > q03.res
29 db2 -tvf q04.sql > q04.res
30 db2 -tvf q05.sql > q05.res
31 db2 -tvf q06.sql > q06.res
32 ECHO Results are saved in the *.res files!
33 ECHO.
34
35 ECHO DISCONNECT FROM DATABASE:
36 ECHO _____
37 db2 terminate
38 ECHO.
39
40 ECHO DROP DATABASE:
41 ECHO _____
42 db2 drop database wc3
43 ECHO.
```

B.2.8 XMP.bat

```
1 @ECHO OFF
2 ECHO.
3
4 ECHO CREATE DATABASE:
5 ECHO _____
6 db2 create database wc3 USING CODESET UTF-8 TERRITORY US
7 ECHO.
8
9 ECHO CONNECT TO DATABASE:
10 ECHO _____
11 db2 connect to wc3
12 ECHO.
13
14 ECHO CREATE TABLES:
15 ECHO _____
16 db2 create table XMP_BIB (XMLDOC XML)
17 db2 create table XMP_BOOKS (XMLDOC XML)
18 db2 create table XMP_PRICES (XMLDOC XML)
19 db2 create table XMP_REVIEWS (XMLDOC XML)
20 ECHO.
21
22 ECHO IMPORT DATA INTO DATABASE:
23 ECHO _____
24 db2 import from bib.del of del xml from . insert into XMP_BIB
25 db2 import from books.del of del xml from . insert into XMP_BOOKS
26 db2 import from prices.del of del xml from . insert into XMP_PRICES
27 db2 import from reviews.del of del xml from . insert into XMP_REVIEWS
28 ECHO.
29
30 ECHO QUERY DATABASE:
31 ECHO _____
32 db2 -tvf q01.sql > q01.res
33 db2 -tvf q02.sql > q02.res
```

B Quellcode

```
34 db2 -tvf q03.sql > q03.res
35 db2 -tvf q04.sql > q04.res
36 db2 -tvf q05.sql > q05.res
37 db2 -tvf q06.sql > q06.res
38 db2 -tvf q07.sql > q07.res
39 db2 -tvf q08.sql > q08.res
40 db2 -tvf q09.sql > q09.res
41 db2 -tvf q10.sql > q10.res
42 db2 -tvf q11.sql > q11.res
43 db2 -tvf q12.sql > q12.res
44 ECHO Results are saved in the *.res files!
45 ECHO.
46
47 ECHO DISCONNECT FROM DATABASE:
48 ECHO -----
49 db2 terminate
50 ECHO.
51
52 ECHO DROP DATABASE:
53 ECHO -----
54 db2 drop database wc3
55 ECHO.
```

B.3 XMLBench

B.3.1 XMLBench.java

```
1 import java.io.File;
2 import java.util.Vector;
3
4 import results.ResultGenerator;
5 import xmark.XMarkGenerator;
6 import db.DB2Connector;
7 import db.SleepyCatConnector;
8
9 /**
10  *
11  * @author Alexander Naegele
12  *
13  * Base class for the XMark test-framework
14  */
15 public class XMLBench {
16
17     /**
18      * Main class starts the benchmarking with the database given in the argument.
19      * First of all the queries for the specific database were parsed and saved in a vector
20      *
21      * After that, the database connector class imports the testdata, runs the queries and
22      * drops the testdata.
23      * Each database is tested 3 times with a different testdata filesize and each query is
24      * tested 4 times.
25      * The results were saved in a vector containing XMarkQueryObjects.
26      * At last, a result cvs file will be generated to import it into a spreadsheet
27      * application.
28      * @param args Name of the database to test
29      */
30     public static void main(String[] args) {
31         String db = args[0];
32         String todo = args[1];
33         Integer fileNumber = new Integer(args[2]);
34
35         if(args.length != 3) {
36             System.err.println("Usage: java XMLBench \"[db2|sleepycat] [load|query] [1|2|3]\"");
37         }
38         return;
39     }
40
41     System.out.println("STARTING XML-BENCH");
42     System.out.println(db + " " + todo + " " + fileNumber);
43     System.out.println("-----\n");
44
45     XMarkGenerator xg = new XMarkGenerator();
46
47     if(db.equals("db2")) {
48         DB2Connector db2conn = new DB2Connector();
49         if (todo.equals("load")) {
50             db2conn.dropTestData();
51         }
52     }
53 }
```

B Quellcode

```
46         db2conn.importInDb(new File("./importdata/xmlgen" + fileNumber
47             + ".xml"));
48     } else if (todo.equals("query")) {
49         Vector queries = xg.parseQueries(new File("./queries/db2/"));
50         Vector resultQueries = db2conn.executeQueries(queries);
51         ResultGenerator rg = new ResultGenerator();
52         String resultFile = db + "-" + "xmlgen" + fileNumber + ".csv";
53         rg.generateCSV(resultQueries, resultFile, db);
54     }
55     db2conn.close();
56 } else if (db.equals("sleepycat")) {
57     SleepyCatConnector spconn = new SleepyCatConnector();
58
59     if (todo.equals("load")) {
60         Vector queries = xg.parseQueries(new File("./queries/sleepycat/
61             "));
62         spconn.dropTestData();
63         spconn.setCacheSize();
64         spconn.importInDb(new File("./importdata/xmlgen" + fileNumber +
65             ".xml"), queries);
66     } else if (todo.equals("query")) {
67         Vector preConfiguredQueries = xg.parseQueries(new File("./
68             queries/sleepycat/"));
69         System.out.println("\nTesting " + db + " with " + "xmlgen" +
70             fileNumber + ".xml");
71         Vector resultQueries = spconn.executeQueries(
72             preConfiguredQueries, fileNumber);
73         ResultGenerator rg = new ResultGenerator();
74         String resultFile = db + "-" + "xmlgen" + fileNumber + ".csv";
75         rg.generateCSV(resultQueries, resultFile, db);
76     }
77     System.out.println("\nXmlBench done ...");
78 }
79 }
```

B.3.2 XMLGenerator.java

```
1 import xmark.XMarkGenerator;
2
3 /**
4  *
5  * @author Alexander Naegele
6  *
7  * This class is the base class for generating the XMark Testdata by using xmlgen.exe
8  */
9 public class XMLGenerator {
10
11     /**
12      * Method starts the generation of 3 testdata files by using xmlgen.exe
13      * @param args 3 arguments containing the factors for the testdata filesize. Factor*113
14      * results the filesize
15      */
16     public static void main(String[] args) {
17         if(args.length != 3) {
18             System.err.println ("Usage: java XMLGenerator \"factor1\" \"factor2\"
19                 \"factor3\"");
20         }
21         return;
22     }
23
24     System.out.println("STARTING XML-GENERATOR");
25     System.out.println("-----\n");
26
27     XMarkGenerator xg = new XMarkGenerator();
28     xg.generateTestFile(args[0], "xmlgen1.xml");
29     xg.generateTestFile(args[1], "xmlgen2.xml");
30     xg.generateTestFile(args[2], "xmlgen3.xml");
31
32     System.out.println("done ...");
33 }
```

B.3.3 db.DB2Connector.java

B Quellcode

```
1 package db;
2
3 import java.io.File;
4 import java.io.FileInputStream;
5 import java.io.FileNotFoundException;
6 import java.util.Vector;
7 import java.sql.*;
8
9 import xmark.XMarkQueryObject;
10
11 /**
12  *
13  * @author Tobias Walter
14  *
15  * This class represents the connector to DB2
16  */
17 public class DB2Connector {
18
19     public final static boolean useIndex = true;
20     private Connection con;
21     private Statement stmt;
22
23     /**
24      * This constructor opens the connection to the database
25      */
26     public DB2Connector() {
27         try {
28             Class.forName("com.ibm.db2.jcc.DB2Driver");
29             con = DriverManager.getConnection("jdbc:db2:xmark");
30             con.setAutoCommit(false);
31             stmt = con.createStatement();
32             System.out.println("DB2-> connected to db");
33         } catch (ClassNotFoundException e) {
34             e.printStackTrace();
35             System.exit(1);
36         } catch (SQLException e) {
37             e.printStackTrace();
38             System.exit(1);
39         }
40     }
41
42     /**
43      * This method imports the testdata into the database.
44      * @param importfile File-reference to the import file
45      */
46     public void importInDb(File importfile) {
47         String sqls = null;
48         PreparedStatement createStmt = null;
49         PreparedStatement alterStmt = null;
50         PreparedStatement insertStmt = null;
51
52         try {
53             sqls = "CREATE TABLE XMLTABLE (XML XML)";
54             createStmt = con.prepareStatement(sqls);
55             createStmt.execute();
56             createStmt.close();
57             long time = System.currentTimeMillis();
58             if(useIndex){
59                 stmt.executeUpdate("CREATE INDEX index01 on xmltable(xml)
60                                     GENERATE KEY USING XMLPATTERN '/site/people/person/@id' AS
61                                     SQL VARCHAR(64)");
62                 stmt.executeUpdate("CREATE INDEX index02 on xmltable(xml)
63                                     GENERATE KEY USING XMLPATTERN '/site/people/person/profile
64                                     /interest/@category' AS SQL VARCHAR(64)");
65                 stmt.executeUpdate("CREATE INDEX index03 on xmltable(xml)
66                                     GENERATE KEY USING XMLPATTERN '/site/people/person/profile
67                                     @income' AS SQL DOUBLE");
68                 stmt.executeUpdate("CREATE INDEX index04 on xmltable(xml)
69                                     GENERATE KEY USING XMLPATTERN '/site/people/person/
70                                     homepage/text()' AS SQL VARCHAR(64)");
71                 stmt.executeUpdate("CREATE INDEX index05 on xmltable(xml)
72                                     GENERATE KEY USING XMLPATTERN '/site/open_auctions/
73                                     open_auction/bidder/increase/text()' AS SQL DOUBLE");
74                 stmt.executeUpdate("CREATE INDEX index06 on xmltable(xml)
75                                     GENERATE KEY USING XMLPATTERN '/site/open_auctions/
76                                     open_auction/bidder/personref/@person' AS SQL VARCHAR(64)");
77                 stmt.executeUpdate("CREATE INDEX index07 on xmltable(xml)
78                                     GENERATE KEY USING XMLPATTERN '/site/open_auctions/
79                                     open_auction/initial/text()' AS SQL DOUBLE");
80                 stmt.executeUpdate("CREATE INDEX index08 on xmltable(xml)
81                                     GENERATE KEY USING XMLPATTERN '/site/closed_auctions/
82                                     closed_auction/price/text()' AS SQL DOUBLE");
83             }
84         }
85     }
86 }
```

B Quellcode

```

67         stmt.executeUpdate("CREATE INDEX index09 on xmltable(xml)
68             GENERATE KEY USING XMLPATTERN '/site/closed_auctions/
69             closed_auction/buyer/@person' AS SQL VARCHAR(64)");
70         stmt.executeUpdate("CREATE INDEX index10 on xmltable(xml)
71             GENERATE KEY USING XMLPATTERN '/site/closed_auctions/
72             closed_auction/itemref/@item' AS SQL VARCHAR(64)");
73         stmt.executeUpdate("CREATE INDEX index12 on xmltable(xml)
74             GENERATE KEY USING XMLPATTERN '/site/regions/europe/item/
75             @id' AS SQL VARCHAR(64)");
76         con.commit();
77         time = System.currentTimeMillis() - time;
78         System.out.println("DB2-> create index complete in " + time + "
79             ms");
80     }
81     sqls = "ALTER TABLE XMLTABLE ACTIVATE NOT LOGGED INITIALLY";
82     alterStmt = con.prepareStatement(sqls);
83     alterStmt.execute();
84     alterStmt.close();
85     sqls = "INSERT INTO XMLTABLE (XML) VALUES (?)";
86     time = System.currentTimeMillis();
87     insertStmt = con.prepareStatement(sqls);
88     insertStmt.setBinaryStream(1, new FileInputStream(importfile), (int)
89         importfile.length());
90     insertStmt.execute();
91     insertStmt.close();
92     time = System.currentTimeMillis() - time;
93     System.out.println("DB2-> insert data complete in " + time + " ms");
94     con.commit();
95 } catch (SQLException e) {
96     try {
97         con.commit();
98     } catch (SQLException e1) {
99         e1.printStackTrace();
100        System.exit(1);
101    }
102    e.printStackTrace();
103    System.exit(1);
104 } catch (FileNotFoundException e) {
105     try {
106         con.commit();
107     } catch (SQLException e1) {
108         e1.printStackTrace();
109         System.exit(1);
110    }
111    e.printStackTrace();
112    System.exit(1);
113 }
114
115 /**
116  * This method executes the queries four times, saves the results in XMarkQueryObjects
117  * and returns a vector containing the results.
118  * @param queries Vector containing XMarkQueryObjects
119  * @return Vector containing the results in XMarkQueryObjects
120  */
121 public Vector executeQueries(Vector queries) {
122     for (int j=0; j<queries.size(); j++){
123         try {
124             for (int i =0; i<4; i++) {
125                 String sqls = null;
126                 PreparedStatement alterStmt = null;
127                 PreparedStatement xqueryStmt = null;
128                 ResultSet rs = null;
129                 sqls = "ALTER TABLE XMLTABLE ACTIVATE NOT LOGGED
130                     INITIALLY";
131                 alterStmt = con.prepareStatement(sqls);
132                 alterStmt.execute();
133                 alterStmt.close();
134                 sqls = ((XMarkQueryObject) queries.elementAt(j)).
135                     getQuery();
136                 long time = System.currentTimeMillis();
137                 xqueryStmt = con.prepareStatement(sqls);
138                 rs = xqueryStmt.executeQuery();
139                 while(rs.next());
140                 time = System.currentTimeMillis() - time;
141                 xqueryStmt.close();
142                 if (i == 0){
143                     ((XMarkQueryObject) queries.elementAt(j)).
144                         setQueryTime1(time);
145                     System.out.print("DB2-> completed " + ((
146                         XMarkQueryObject) queries.elementAt(j)).
147                         getQueryName() + " in " + time + " ms");
148                 } else if (i == 1){

```

B Quellcode

```

136         ((XMarkQueryObject) queries.elementAt(j)).
137             setQueryTime2(time);
138         System.out.print(", " + time + " ms");
139     } else if (i == 2){
140         ((XMarkQueryObject) queries.elementAt(j)).
141             setQueryTime3(time);
142         System.out.print(", " + time + " ms");
143     } else if (i == 3){
144         ((XMarkQueryObject) queries.elementAt(j)).
145             setQueryTime4(time);
146         System.out.println(", " + time + " ms");
147     }
148     con.commit();
149 } catch (SQLException e) {
150     try {
151         con.commit();
152     } catch (SQLException e1) {
153         e1.printStackTrace();
154         System.exit(1);
155     }
156     e.printStackTrace();
157     System.exit(1);
158 }
159 return queries;
160 }
161 /**
162  * This method drops the imported testdata in the database
163  *
164  */
165 public void dropTestData() {
166     String sqls = null;
167     PreparedStatement alterStmt = null;
168     PreparedStatement dropStmt = null;
169
170     try {
171         sqls = "ALTER TABLE XMLTABLE ACTIVATE NOT LOGGED INITIALLY";
172         alterStmt = con.prepareStatement(sqls);
173         alterStmt.execute();
174         alterStmt.close();
175         sqls = "DROP TABLE XMLTABLE";
176         long time = System.currentTimeMillis();
177         dropStmt = con.prepareStatement(sqls);
178         dropStmt.executeUpdate();
179         dropStmt.close();
180         con.commit();
181         time = System.currentTimeMillis() - time;
182         System.out.println("DB2-> delete old data complete in " + time + " ms");
183     } catch (SQLException e) {
184         if(e.getMessage().equals("DB2 SQL error: SQLCODE: -204, SQLSTATE:
185             42704, SQLERRMC: ADMINISTRATOR.XMLTABLE")) {
186             try {
187                 con.commit();
188             } catch (SQLException e1) {
189                 e1.printStackTrace();
190                 System.exit(1);
191             }
192         } else {
193             try {
194                 con.commit();
195             } catch (SQLException e1) {
196                 e1.printStackTrace();
197                 System.exit(1);
198             }
199             e.printStackTrace();
200             System.exit(1);
201         }
202     }
203 }
204 }
205 /**
206  * This method closes the connection to the database
207  *
208  */
209 public void close() {
210     try {
211         stmt.close();
212         con.close();
213         System.out.println("DB2-> disconnected from db");
214     }

```

B Quellcode

```
215         } catch (SQLException e) {
216             try {
217                 con.commit();
218             } catch (SQLException e1) {
219                 e1.printStackTrace();
220                 System.exit(1);
221             }
222             e.printStackTrace();
223             System.exit(1);
224         }
225     }
226 }
227 }
```

B.3.4 db.SleepycatConnector.java

```
1 package db;
2
3 import java.io.File;
4 import java.util.Iterator;
5 import java.util.Vector;
6 import java.util.regex.Matcher;
7 import java.util.regex.Pattern;
8
9 import xmark.XMarkQueryObject;
10
11 import com.sleepycat.dbxml.XmlIndexSpecification;
12 import com.sleepycat.dbxml.XmlManager;
13
14 import db.sleepyCat.ExampleLoadContainer;
15 import db.sleepyCat.XQuery_FrameWork;
16
17 /**
18  * This class represents the connector to SleepyCat
19  * @author Alexander Naegele and Joachim Kluge
20  *
21  * @param path2DbEnv Environment of the database. This directory must be existing
22  * @param cacheSize Cache size of the environment in mebybyte
23  * @param useIndex Wether to use indexing in database creation or not. Indexing is optimized
24  *   for the queries present, so don't change the queries in the directory once you created
25  *   the databse
26  * @param importFileBasename The base name of the import file. A number and .xml are assumed by
27  *   the application. E.g. "xmlgen" -> xmlgen1.xml
28  */
29 public class SleepyCatConnector {
30
31     public final static File path2DbEnv = new File("c:/sleepycatDbEnv/");
32     public final static Integer cacheSize = new Integer(8);
33     public final static boolean useIndex = true;
34     public final static String importFileBasename = "xmlgen";
35
36     /**
37      * Imports a single file and adds index if desired.
38      * @param importfile the single file to be exported
39      * @param queries Vector of <code>XMarkQueryObject</code>. Needed for index optimizing.
40      */
41     public void importInDb(File importfile, Vector queries) {
42         try {
43             ExampleLoadContainer.loadXmlFiles(path2DbEnv, "ExampleData.dbxml",
44                 importfile, queries, cacheSize, useIndex);
45         } catch (Throwable e) {
46             System.err.println("error while importing file and adding index" + e);
47         }
48     }
49
50     /**
51      * This method executes the queries four times, saves the results in <code>
52      * XMarkQueryObject</code> and returns a vector containing the results.
53      * @param nonPreparedQueries Vector containing XMarkQueryObjects
54      * @return Vector containing the results in XMarkQueryObjects
55      */
56     public Vector executeQueries(Vector nonPreparedQueries, Integer testNumber) {
57         Vector queries = prepareQueries(nonPreparedQueries, SleepyCatConnector.
58             path2DbEnv, testNumber);
59         try {
60             queries = XQuery_FrameWork.mainMethod(path2DbEnv.toString(), queries);
61         } catch (Throwable e) {
62             System.err.println("error while processing queries" + e);
63         }
64     }
65 }
```

B Quellcode

```

59         return queries;
60     }
61
62     /**
63      * deletes the whole database.
64      *
65      */
66     public void dropTestData() {
67         try {
68             //File containerPath = new File(pathToDbEnv.getAbsolutePath() + "/"
69             //ExampleData.dbxml");
70             String [] dirContents = path2DbEnv.list();
71             if (dirContents != null) {
72                 for (int i = 0; i < dirContents.length; i++) {
73                     File entry = new File(path2DbEnv + File.separator + dirContents[
74                     i]);
75                     if (entry.isFile() && entry.canWrite()) {
76                         System.out.println("deleting: " + entry.toString());
77                         if (entry.delete() == false) {
78                             System.err.println("deleting failed!");
79                         }
80                     }
81                 }
82             }
83             XmlManager theMgr = new XmlManager();
84             theMgr.removeContainer(path2DbEnv.getAbsolutePath() + "/ExampleData.
85             dbxml");
86             theMgr.close();
87         }
88     } catch (Exception e){}
89 }
90
91 /**
92  * closes the connection to the database
93  *
94  */
95 public void close() {
96     /* unnecessary */
97 }
98
99 /**
100  * Sets the cache size.
101  *
102  */
103 public void setCacheSize() {
104     System.out.println("setting cachesize to: 8MB");
105     this.cacheSize = new Integer(8);
106 }
107
108
109
110 /**
111  * Regular expression replacing in order to prepares the queries for conformance with
112  * Sleepycat.
113  * @param nonPreparedQueries queries which are not prepared yet.
114  * @param containerDir location of the container
115  * @param testNumber added after <code>importFileName</code>
116  * @return Vector of <code>XMarkQueryObject</code>
117  */
118 public static Vector prepareQueries(Vector nonPreparedQueries, File containerDir,
119 Integer testNumber) {
120     Vector returnVector = new Vector();
121     String myQuery;
122     File containerPath = new File(containerDir.getAbsolutePath() + "/ExampleData.
123     dbxml");
124     String containerPathString = null;
125     try {
126         containerPathString = containerPath.getCanonicalPath().replaceAll(new
127         String("\\\\\\"), new String("/"));
128     }
129     catch (Exception e) {
130         System.err.println(e);
131     }
132     System.out.print("preparing queries...");
133     Iterator queryIterator = nonPreparedQueries.iterator();
134     while(queryIterator.hasNext()) {
135         XMarkQueryObject myObject = (XMarkQueryObject) queryIterator.next();
136         myQuery = myObject.getQuery();
137         myQuery = regexXmark(myQuery, containerPathString, testNumber);
138         myObject.setQuery(myQuery);

```

B Quellcode

```

136         returnVector.add(myObject);
137     }
138     System.out.println("done");
139     return returnVector;
140 }
141
142 /**
143  * Individual regular expressions for Xmark.
144  * @param string part if the query which needs to be changed
145  * @param containerPathString location of the container as string
146  * @param testNumber added after <code>importFileName</code>
147  * @return
148  */
149 private static String regexXmark(String string, String containerPathString, Integer
    testNumber) {
150     string = string.toLowerCase();
151     string = string.replaceAll(new String("\\"), new String("'"));
152     string = string.replaceAll(new String("document\\('"), new String("doc\\('dbxml:" +
        containerPathString + '/''));
153     string = string.replaceAll(new String("/text\\(\\)"), new String(""));
154     string = string.replaceAll(new String("/auction.xml'\\)"), new String("/" +
        importFileName + testNumber.intValue() + ".xml'\\)"));
155     return string;
156 }
157
158 /**
159  * updates the index specification supplied. An index is added individually to the query
    supplied
160  * @param indexSpec <code>XmlIndexSpecification</code> to be updated
161  * @param queryName file name of the query
162  * @return updated <code>XmlIndexSpecification</code>
163  */
164 public static XmlIndexSpecification addIndex(XmlIndexSpecification indexSpec, String
    queryName) {
165     try {
166         Pattern p = Pattern.compile("[0-9]\\d{0,1}");
167         Matcher m = p.matcher(queryName);
168         m.find();
169         Integer queryNumber = new Integer(queryName.substring(m.start(), m.end
            ()));
170         System.out.print(queryName.substring(m.start(), m.end()) + "...");
171         switch (queryNumber.intValue()) {
172             case 1:
173                 indexSpec.addIndex("", "id", "edge-attribute-equality-string");
174                 break;
175             case 2:
176                 break;
177             case 3:
178                 indexSpec.addIndex("", "bidder", "edge-element-presence-none");
179                 indexSpec.addIndex("", "increase", "edge-element-equality-
                    decimal");
180                 break;
181             case 4:
182                 indexSpec.addIndex("", "personref", "edge-attribute-equality-
                    string");
183                 break;
184             case 5:
185                 indexSpec.addIndex("", "price", "edge-element-equality-decimal"
                    );
186                 break;
187             case 6:
188                 break;
189             case 7:
190                 break;
191             case 8:
192                 indexSpec.addIndex("", "person", "edge-attribute-equality-
                    string");
193                 indexSpec.addIndex("", "id", "edge-attribute-equality-string");
194                 break;
195             case 9:
196                 indexSpec.addIndex("", "item", "edge-attribute-equality-string"
                    );
197                 indexSpec.addIndex("", "person", "edge-attribute-equality-
                    string");
198                 indexSpec.addIndex("", "id", "edge-attribute-equality-string");
199                 break;
200             case 10:
201                 indexSpec.addIndex("", "person", "edge-element-equality-string"
                    );
202                 indexSpec.addIndex("", "category", "edge-attribute-equality-
                    string");
203                 break;
204             case 11:

```

B Quellcode

```
205         indexSpec.addIndex("", "initial", "edge-element-equality -
206             decimal");
207         indexSpec.addIndex("", "income", "edge-attribute-equality -
208             decimal");
209         break;
210     case 12:
211         indexSpec.addIndex("", "initial", "edge-element-equality -
212             decimal");
213         indexSpec.addIndex("", "income", "edge-attribute-equality -
214             decimal");
215         break;
216     case 13:
217         break;
218     case 14:
219         break;
220     case 15:
221         break;
222     case 16:
223         break;
224     case 17:
225         break;
226     case 18:
227         break;
228     case 19:
229         break;
230     case 20:
231         indexSpec.addIndex("", "description", "edge-element-substring -
232             string");
233         break;
234     case 21:
235         break;
236     case 22:
237         break;
238     case 23:
239         break;
240     case 24:
241         break;
242     case 25:
243         break;
244     case 26:
245         break;
246     case 27:
247         break;
248     case 28:
249         indexSpec.addIndex("", "income", "edge-element-equality-decimal
250             ");
251         break;
252     }
253 } catch (Exception e) {
254     System.err.println("error while adding index: " + e);
255 }
256 return indexSpec;
257 }
258 }
259 }
```

B.3.5 db.Sleepycat.exampleLoadContainer.java

```
1 package db.sleepyCat;
2
3 import java.io.*;
4 import java.util.*;
5 import xmark.XMarkQueryObject;
6 import com.sleepycat.db.*;
7 import com.sleepycat.dbxml.*;
8 import db.SleepyCatConnector;
9
10 /**
11  * Creates a database environment and inserts a single file into this database.
12  * @author Joachim Kluge
13  *
14  */
15
16 public class ExampleLoadContainer
17 {
18
19     static long startTime = System.currentTimeMillis();
20
21     /**
22      * Creates an environment and sets it's options.
23      * @param home
24      * @param cacheSize
25      * @return
26      * @throws DatabaseException
27      * @throws FileNotFoundException
28      */
29     private static Environment createEnv(File home, Integer cacheSize)
30     throws DatabaseException, FileNotFoundException {
31
32
33         EnvironmentConfig config = new EnvironmentConfig();
34         config.setCacheSize(cacheSize.intValue() * 1024 * 1024);
35         config.setAllowCreate(true);
36         config.setInitializeCache(true);
```

B Quellcode

```
37     config.setTransactional(false);
38     config.setInitializeLocking(true);
39     config.setInitializeLogging(false);
40     return new Environment(home, config);
41 }
42
43 /**
44  * The main method.
45  * @param path2DbEnv the directory of the environment
46  * @param theContainer the name of the container where the file will be stored
47  * @param file2Add the file to be added
48  * @param queries Vector of several XMarkQueryObject with the query set
49  * @param cacheSize in mebibyte
50  * @param useIndex wether to initialize the environment with an index or not
51  * @throws Throwable
52  */
53 public static void loadXmlFiles(File path2DbEnv, String theContainer, File file2Add, Vector
    queries, Integer cacheSize, boolean useIndex)
54     throws Throwable {
55     if (file2Add == null || !file2Add.isFile()) {
56         System.out.println("no file found!");
57     }
58     XmlManager theMgr = null;
59     XmlContainer openedContainer = null;
60     Environment env = null;
61     XmlIndexSpecification indexSpec = new XmlIndexSpecification();
62     String theFile = file2Add.toString();
63
64     try {
65         env = createEnv(path2DbEnv, cacheSize);
66         XmlManagerConfig theMgrConfig = new XmlManagerConfig();
67         theMgrConfig.setAllowExternalAccess(false);
68         theMgr = new XmlManager(env, theMgrConfig);
69         XmlContainerConfig config = new XmlContainerConfig();
70         openedContainer = theMgr.createContainer(theContainer, config);
71
72         XmlUpdateContext updateContext = theMgr.createUpdateContext();
73
74         if (useIndex) {
75             // we are going through the available queries
76             Iterator queriesIterator = queries.iterator();
77             while(queriesIterator.hasNext()) {
78                 try {
79                     XMarkQueryObject myObject = (XMarkQueryObject)
80                         queriesIterator.next();
81                     System.out.print("setting up index
82                         specification for query");
83                     // we are configuring the index
84                     // specification for each query
85                     // separately
86                     indexSpec = SleepyCatConnector.addIndex
87                         (indexSpec, myObject.getQueryName
88                         ());
89                     System.out.println("done");
90                 } catch (Exception e) {
91                     System.err.println("error while adding index: "
92                         + e);
93                 }
94             }
95             // finally sets the index in the database
96             System.out.print("setting IndexSpecification...");
97             openedContainer.setIndexSpecification(indexSpec, theMgr
98                 .createUpdateContext());
99             System.out.println("done");
100         } else System.out.println("indexing switched off");
101
102         // The FileInputStream reads the file bit by bit. So we reach
103         // compatibility with large files
104         System.out.print("Trying to add : " + file2Add.getAbsolutePath
105             () + "...");
106         XmlInputStream myXmlInputStream = theMgr.
107             createLocalFileInputStream(file2Add.getAbsolutePath());
108         openedContainer.putDocument(file2Add.getName(),
109             myXmlInputStream, updateContext);
110         System.out.println("added " + theFile + " to container " +
111             theContainer);
112
113         // let's see which indexes are set
114         XmlIndexSpecification indexSpec2 = openedContainer.
115             getIndexSpecification();
116         XmlIndexDeclaration indexDec;
117         System.out.println("set up indexes: ");
118         while ((indexDec = indexSpec2.next()) != null) {
```

B Quellcode

```
106         System.out.println("\t" + indexDec.name + ", " +
107             indexDec.index);
108     }
109
110     } catch (Exception e) {
111
112         System.err.println("Error loading files into container " +
113             theContainer);
114         System.err.println("    Message: " + e.getMessage());
115         throw e;
116     } finally {
117         try {
118             if (openedContainer != null) openedContainer.close();
119             if (theMgr != null) theMgr.close();
120             if (env != null) env.close();
121         } catch (Exception e) {
122             // ignore exceptions on close
123         }
124         System.out.println("Seconds needed: " + (System.currentTimeMillis()
125             - startTime) / 1000 );
126     }
127 }
128 }
```

B.3.6 db.Sleepycat.XQuery_Framework.java

```
1 package db.sleepyCat;
2 import com.sleepycat.db.DatabaseException;
3 import com.sleepycat.db.Environment;
4 import com.sleepycat.db.EnvironmentConfig;
5 import com.sleepycat.dbxml.XmlContainer;
6 import com.sleepycat.dbxml.XmlIndexDeclaration;
7 import com.sleepycat.dbxml.XmlIndexSpecification;
8 import com.sleepycat.dbxml.XmlManager;
9 import com.sleepycat.dbxml.XmlManagerConfig;
10 import com.sleepycat.dbxml.XmlQueryContext;
11 import com.sleepycat.dbxml.XmlQueryExpression;
12 import com.sleepycat.dbxml.XmlResults;
13 import com.sleepycat.dbxml.XmlValue;
14
15 import java.util.*;
16 import java.io.*;
17
18 import xmark.XMarkQueryObject;
19
20 /**
21  * Executes XQueries delivered by a Vector of <code>XMarkQueryObject</code> against an existing
22  * database.
23  * Each query is executed four times consecutively.
24  * The time for each run is stored in the <code>XMarkQueryObject</code> and at the end, a
25  * Vector with updated <code>XMarkQueryObject</code> is returned.
26  * @author Joachim Kluge
27  */
28
29 public class XQuery.Framework {
30
31     /**
32      * The main method.
33      * @param containerDir Directory where the container (i.e. the database if found)
34      * @param queries Vector of XMarkQueryObjects
35      * @return updated <code>XMarkQueryObject</code> with four different times set
36      */
37
38     public static Vector mainMethod(String containerDir, Vector queries) {
39         long startTime = 0;
40         long queryTime = 0;
41         Environment env = null;
42         XmlManagerConfig theMgrConfig = null;
43         XmlManager myManager = null;
44         XmlContainer myContainer = null;
45         XmlQueryExpression qe = null;
46         XmlResults results = null;
47         XmlValue value = null;
48         Vector returnVector = new Vector();
49         File containerPath = new File (containerDir + "/ExampleData.dbxml");
50         String containerPathString = null;
51         try {
```

B Quellcode

```

50         containerPathString = containerPath.getCanonicalPath().replaceAll(new
51             String("\\\\"), new String("/"));
52     }
53     catch (Exception e) {
54         System.err.println(e);
55     }
56     try {
57         System.out.println("starting querying...");
58         env = joinEnv(new File(containerDir));
59         theMgrConfig = new XmlManagerConfig();
60         theMgrConfig.setAllowExternalAccess(false);
61         myManager = new XmlManager(env, theMgrConfig);
62
63         myContainer = myManager.openContainer(containerPathString);
64         System.out.println("Opened container " + containerPath);
65
66         XmlQueryContext context = myManager.createQueryContext();
67
68         // prints the available indexes
69         XmlIndexSpecification indexSpec2 = myContainer.getIndexSpecification();
70         XmlIndexDeclaration indexDec;
71         System.out.println("available indexes: ");
72         while ((indexDec = indexSpec2.next()) != null) {
73             System.out.println("\t" + indexDec.name + ", " + indexDec.index
74                 );
75         }
76
77         //goes through the Vecot of XmarkQueryObjects
78         Iterator queriesIterator = queries.iterator();
79         while(queriesIterator.hasNext()) {
80             try {
81                 XMarkQueryObject myObject = (XMarkQueryObject) queriesIterator.
82                     next();
83                 String myQuery = myObject.getQuery();
84
85                 System.out.println("executing 4 times: " + myObject.
86                     getQueryName());
87
88                 for (int executionNumber = 1; executionNumber <= 4;
89                     executionNumber++) {
90                     startTime = System.currentTimeMillis();
91                     qe = myManager.prepare(myQuery, context);
92                     results = qe.execute(context);
93                     queryTime = System.currentTimeMillis() -
94                         startTime;
95
96                     switch (executionNumber) {
97                         case 1:
98                             myObject.setQueryTime1(
99                                 queryTime);
100                             //uncomment this for printing
101                             //of the result
102                             /*
103                             System.out.println();
104                             System.out.println("result of 1st
105                                 run:");
106                             while ((value = results.next())
107                                 != null){
108                                 myObject.setQueryResult
109                                     (value.asString())
110                                 ;
111                             System.out.println(value.
112                                 asString());
113                             }
114                             System.out.println();
115                             */
116                             break;
117                         case 2:
118                             myObject.setQueryTime2(
119                                 queryTime);
120                             break;
121                         case 3:
122                             myObject.setQueryTime3(
123                                 queryTime);
124                             break;
125                         case 4:
126                             myObject.setQueryTime4(
127                                 queryTime);
128                             break;
129                     }
130                 }
131                 System.out.println("time needed: " + queryTime)
132                     ;

```

B Quellcode

```
117         }
118         returnVector.add(myObject);
119         System.out.println("...sucessful");
120
121     } catch (Exception e) {
122         System.err.println(e);
123     }
124 }
125 } catch (Exception e) {
126     System.err.println(e);
127 } finally {
128     try {
129         if (myContainer != null) {
130             myContainer.close();
131         }
132         if (myManager != null) {
133             myManager.close();
134         }
135         if (env != null) {
136             env.close();
137         }
138     } catch (Exception e) {
139         System.err.println(e);
140     }
141 }
142 }
143 System.out.println("finished executing all queries.");
144 return returnVector;
145 }
146
147 /**
148  * Joins an already existing environment.
149  * @param the location of the existing environment
150  * @return
151  * @throws DatabaseException
152  * @throws FileNotFoundException
153  */
154 private static Environment joinEnv(File home)
155     throws DatabaseException, FileNotFoundException {
156     EnvironmentConfig config = new EnvironmentConfig();
157     config.setJoinEnvironment(true);
158     return new Environment(home, config);
159 }
160 }
```

B.3.7 results.ResultGenerator.java

```
1 package results;
2
3 import java.io.BufferedWriter;
4 import java.io.File;
5 import java.io.FileWriter;
6 import java.io.IOException;
7 import java.text.SimpleDateFormat;
8 import java.util.Calendar;
9 import java.util.GregorianCalendar;
10 import java.util.TimeZone;
11 import java.util.Vector;
12
13 import xmark.XMarkQueryObject;
14
15 /**
16  *
17  * @author Alexander Naegele
18  *
19  * This class represents the Result Generator for generating a summary after finishing the
20  * tests.
21  */
22 public class ResultGenerator {
23
24     /**
25      * This method generates a csv-file with the contents of the XMarkQueryObjects
26      * @param resultQueries Vector containing XMarkQueryObjects with the results of the
27      * tests
28      * @param filename String containing the name of the file to generate
29      */
30     public void generateCSV(Vector resultQueries, String filename, String db) {
31         try {
32             System.out.println("generating result csv-file ...");
33             StringBuffer sb = new StringBuffer();
```

B Quellcode

```
32     File resultFile = new File("./results/" + filename);
33     FileWriter fw;
34     fw = new FileWriter(resultFile);
35     BufferedWriter bw = new BufferedWriter(fw);
36
37     sb.append("\n Benchmarkauswertung\n",,,,,,\n");
38     sb.append(",,,,,\n");
39     sb.append("\n" + db.toUpperCase() + "\n\n");
40     sb.append(",,,,,\n");
41     sb.append(",,,,,\n");
42     sb.append("\n Category\n,\n Xquery\n,\n 1st round\n,\n 2nd round\n,\n 3rd
round\n,\n 4th round\n,\n Avg\n,\n Valid\n,\n Result\n\n");
43     sb.append(" ,\n uncached\n,\n cached\n",,,,,,\n");
44
45     if(resultQueries != null) {
46         for(int i=0; i<resultQueries.size(); i++) {
47             XMarkQueryObject xmqo = (XMarkQueryObject) resultQueries
48                 .get(i);
49             String resultTime1 = String.valueOf(xmqo.getQueryTime1
50                 ());
51             String resultTime2 = String.valueOf(xmqo.getQueryTime2
52                 ());
53             String resultTime3 = String.valueOf(xmqo.getQueryTime3
54                 ());
55             String resultTime4 = String.valueOf(xmqo.getQueryTime4
56                 ());
57             String avgTime = String.valueOf( (xmqo.getQueryTime1()
58                 + xmqo.getQueryTime2() + xmqo.getQueryTime3() +
59                 xmqo.getQueryTime4())/4 );
60
61             sb.append("\n" + xmqo.getQueryName() + "\n,\n" +
62                 resultTime1 + "\n,\n" + resultTime2 + "\n,\n" +
63                 resultTime3 + "\n,\n" + resultTime4 + "\n,\n" +
64                 avgTime + "\n\n");
65         }
66     }
67
68     sb.append(",,,,,\n");
69     sb.append("\n Generated on " + getDate() + "\n",,,,,,\n");
70
71     String result = sb.toString();
72     bw.write(result);
73     bw.close();
74     fw.close();
75 } catch (IOException e) {
76     e.printStackTrace();
77 }
78
79 /**
80  * Gets the current system date and formats it ,
81  * e.g. Do, 19. Jul 2005 at 11:53:04
82  * @return Formatted date as String
83  */
84 public String getDate() {
85     Calendar cal = new GregorianCalendar(TimeZone.getTimeZone("ETC"));
86     SimpleDateFormat dfmt = new SimpleDateFormat("E', ' d'. ' MMM yyyy 'at' HH': 'mm': 'ss");
87     String date = dfmt.format(cal.getTime());
88     return date;
89 }
90 }
```

B.3.8 xmark.XMarkGenerator.java

```
1 package xmark;
2
3 import java.io.File;
4 import java.io.FileInputStream;
5 import java.io.FileNotFoundException;
6 import java.io.IOException;
7 import java.util.Vector;
8
9 /**
10  *
11  * @author Alexander Naegele
12  *
13  * This class represent all actions which stand in relation with the XMark (generating the
14  * testdata and parsing the queries)
15  */
16 public class XMarkGenerator {
```

B Quellcode

```
16 private Vector queries;
17
18 /**
19  * This method generates a XMark testdata file
20  * @param factor String containing the factor for the testfile (factor*113=size of
21  *   testfile in MB)
22  * @param filename String containing the filename for the generated testfile
23  */
24 public void generateTestFile(String factor, String filename) {
25     try {
26         System.out.print("generating testfile with factor " + factor + " (about
27         " + Float.parseFloat(factor)*113 + " MB)");
28         String syscall = "./binaries/xmlgen.exe /f " + factor + " /o ./
29         importdata/" + filename;
30         long genTime = System.currentTimeMillis();
31         Process p = Runtime.getRuntime().exec(syscall);
32         p.waitFor();
33         genTime = System.currentTimeMillis() - genTime;
34         System.out.println("-> generated in " + genTime + " ms\n");
35     } catch (IOException e) {
36         e.printStackTrace();
37     } catch (InterruptedException e1) {
38         e1.printStackTrace();
39     }
40 }
41
42 /**
43  * This method parses the queries directory for the specific db, generates
44  * XMarkQueryObjects for the
45  * queries and puts them into a vector.
46  * @param dir File-object containing a reference to the queries directory
47  * @return Vector containing XMarkQueryObjects
48  */
49 public Vector parseQueries(File dir) {
50     System.out.print("parsing directory for queries... ");
51
52     XMarkQueryObject xmqo;
53
54     if (dir.isDirectory()) {
55         long time = System.currentTimeMillis();
56         File[] children = dir.listFiles();
57         queries = new Vector();
58         int anzahl = 0;
59         for (int i=0; i<children.length; i++) {
60             if (children[i].isFile() && children[i].toString().matches(".{1,}query$")) {
61                 try {
62                     FileInputStream fis = new FileInputStream(
63                         children[i]);
64                     int length = (int)children[i].length();
65                     byte[] bytes = new byte[length];
66                     fis.read(bytes, 0, length);
67                     xmqo = new XMarkQueryObject(children[i].getName()
68                         (), new String(bytes));
69                     fis.close();
70                     queries.add(xmqo);
71                     anzahl++;
72                 } catch (FileNotFoundException e) {
73                     e.printStackTrace();
74                 } catch (IOException e) {
75                     e.printStackTrace();
76                 }
77             }
78         }
79
80         time = System.currentTimeMillis() - time;
81         System.out.println("-> " + anzahl + " queries were found and added in "
82             + time + " ms");
83     }
84
85     return queries;
86 }
87
88 }
```

B.3.9 xmark.XMarkQueryObject.java

```
1 package xmark;
2
3 /**
4  *
5  * @author Alexander Naegele
6  */
```

B Quellcode

```
7 | * This class represents an object for the XMark-queries containing the name of the query, the
8 |   query itself, the query-result
9 |   * and 4 query-times for the tests.
10 | */
11 | public class XMarkQueryObject {
12 |     private String queryName;
13 |     private String query;
14 |     private String queryResult;
15 |     private long queryTime1;
16 |     private long queryTime2;
17 |     private long queryTime3;
18 |     private long queryTime4;
19 |
20 |     /**
21 |      *
22 |      * @param queryName Name of the Query
23 |      * @param query The XMark-query
24 |      */
25 |     public XMarkQueryObject(String queryName, String query) {
26 |         this.query = query;
27 |         this.queryName = queryName;
28 |     }
29 |
30 |     /**
31 |      * Returns the XMark query
32 |      * @return String containing the query
33 |      */
34 |     public String getQuery() {
35 |         return query;
36 |     }
37 |
38 |     /**
39 |      * Sets the XMark query
40 |      * @param query String containing the query
41 |      */
42 |     public void setQuery(String query) {
43 |         this.query = query;
44 |     }
45 |
46 |     /**
47 |      * Returns the query name
48 |      * @return String containing the name of the query
49 |      */
50 |     public String getQueryName() {
51 |         return queryName;
52 |     }
53 |
54 |     /**
55 |      * Sets the name of the query
56 |      * @param queryName String containing the query
57 |      */
58 |     public void setQueryName(String queryName) {
59 |         this.queryName = queryName;
60 |     }
61 |
62 |     /**
63 |      * Returns the result of the query
64 |      * @return String containing the result of the query
65 |      */
66 |     public String getQueryResult() {
67 |         return queryResult;
68 |     }
69 |
70 |     /**
71 |      * Sets the result of the query
72 |      * @param queryResult String containing the result of the query
73 |      */
74 |     public void setQueryResult(String queryResult) {
75 |         this.queryResult = queryResult;
76 |     }
77 |
78 |     /**
79 |      * Gets the test time of query 1
80 |      * @return long containing the test time in ms
81 |      */
82 |     public long getQueryTime1() {
83 |         return queryTime1;
84 |     }
85 |
86 |     /**
87 |      * Sets the test time of the query 1
88 |      * @param queryTime long containing the query time in ms
89 |      */
90 |     public void setQueryTime1(long queryTime) {
```

B Quellcode

```
90         this.queryTime1 = queryTime;
91     }
92
93     /**
94      * Gets the test time of query 2
95      * @return long containing the test time in ms
96      */
97     public long getQueryTime2() {
98         return queryTime2;
99     }
100
101     /**
102      * Sets the test time of the query 2
103      * @param queryTime long containing the query time in ms
104      */
105     public void setQueryTime2(long queryTime) {
106         this.queryTime2 = queryTime;
107     }
108
109     /**
110      * Gets the test time of query 3
111      * @return long containing the test time in ms
112      */
113     public long getQueryTime3() {
114         return queryTime3;
115     }
116
117     /**
118      * Sets the test time of the query 3
119      * @param queryTime long containing the query time in ms
120      */
121     public void setQueryTime3(long queryTime) {
122         this.queryTime3 = queryTime;
123     }
124
125     /**
126      * Gets the test time of query 4
127      * @return long containing the test time in ms
128      */
129     public long getQueryTime4() {
130         return queryTime4;
131     }
132
133     /**
134      * Sets the test time of the query 4
135      * @param queryTime long containing the query time in ms
136      */
137     public void setQueryTime4(long queryTime) {
138         this.queryTime4 = queryTime;
139     }
140 }
```

B.3.10 build.xml

```
1 <project name="XMLBench" default="help" basedir="..">
2     <description>
3         XML Build-File for the XMLBench
4     </description>
5     <!-- set global properties for this build -->
6     <property name="src" location="src"/>
7     <property name="build" location="build"/>
8     <property name="doc" location="doc"/>
9
10    <target name="init">
11        <!-- Create the time stamp -->
12        <tstamp/>
13        <!-- Create the build directory structure used by compile -->
14        <mkdir dir="${build}"/>
15        <mkdir dir="${doc}"/>
16    </target>
17
18    <target name="compile" depends="init" description="compile the source " >
19        <!-- Compile the java code from ${src} into ${build} -->
20        <javac srcdir="${src}" destdir="${build}">
21            <include name="**/*.java" />
22        </javac>
23    </target>
24
25    <target name="doc" depends="init"
26        description="document the source " >
```

B Quellcode

```
27 | <javadoc destdir="${doc}"
28 |     author="true"
29 |     version="true"
30 |     use="true"
31 |     windowtitle="XMLBench API">
32 |
33 |     <fileset dir="src" defaultexcludes="yes">
34 |         <include name="**/*.java" />
35 |         <!-- exclude name="com/dummy/test/doc-files/**"/-->
36 |     </fileset>
37 |
38 |     <doctitle><![CDATA[<h1>XMLBench API</h1>]]></doctitle>
39 |     <bottom><![CDATA[<i>Alexander Naegele</i>]]></bottom>
40 | </javadoc>
41 | </target>
42 |
43 | <target name="clean"
44 |     description="clean up" >
45 |     <!-- Delete the ${build} and ${dist} directory trees -->
46 |     <delete dir="${build}"/>
47 |     <delete dir="${doc}"/>
48 | </target>
49 |
50 | <target name="help"
51 |     description="task overview" >
52 | <echo>
53 | The following Targets are available:
54 |   init: creates the buil and doc directory
55 |   clean: deletes the doc and build directory
56 |   compile: compiles the sources in /src
57 |   doc: generates javadoc-files
58 | </echo>
59 | </target>
60 | </project>
```

C CD

Die beigelegte CD enthält das vollständige Subversion-Repository, dass im Laufe des Projektes zur Verwaltung von Sourcecodes, Dokumenten und Protokollen verwendet wurde.