

XML-Benchmarks *Markus Rajai Hammori*

1 Einleitung

Der beinahe schon explosionsartige Zuwachs von XML-Daten in vielen Bereichen führt dazu, dass es immer wichtiger wird diese Daten effizient zu speichern und zu pflegen. Hierfür bieten sich momentan zwei Klassen von Lösungen an. Zum einen natürlich die nativen XML-Datastores, die auf die speziellen Aspekte von XML zugeschnitten sind, und deshalb (zumindest in einigen Bereichen) bessere Performance bieten sollten. Zum anderen die bewährten RDBMS, die schon lange erprobt sind und für relationale Datenstrukturen hoch optimiert wurden. Es gibt natürlich auch Systeme die auf OODBMS basieren, da jedoch diese Systeme selbst meist noch in der Entwicklung stecken, wird hier nicht näher auf sie eingegangen.

Momentan sind noch sehr wenig Daten über Performance, Feature-Umfang und laufende Kosten der vielen vorhandenen XML-Storage Produkte vorhanden. Gerade diese Informationen würden aber benötigt um Entscheidungen zum Beispiel über die Kosten einer Migration, oder die Implementierbarkeit eines Projekts auf fundierter Basis treffen zu können. Viele Firmen nutzen momentan schon ein RDBMS, und wollen die Anschaffung eines speziellen zweiten Systems für Ihre XML-Daten vermeiden. Hier wäre es wichtig zu erfahren ob die XML-erweiterten relationalen Systeme den Anforderungen genügen. Mehr dazu in Abschnitt 2.

Der Rest dieser Arbeit ist folgendermaßen aufgebaut: Zuerst wird die Motivation für Benchmarks generell geliefert, um darauf aufbauend auf die Besonderheiten bei XML-Benchmarks hinzuweisen. Danach werden zwei aktuelle Benchmarks vorgestellt, die auf unterschiedlichen Ansätzen basieren. Bei beiden Benchmarks wird hierbei der allgemeine Ansatz, der Systemaufbau, die Anforderungen die an die XML-Fähigkeiten des Testsystems gestellt werden, die Struktur und Initialisierung der Datenbank, sowie die Ermittlung der Messdaten beschrieben. Es wird auch versucht einzuschätzen für welche Aufgaben sich die Benchmarks eignen. Zum Abschluss wird noch ein weiterer Ansatz kurz angesprochen, und versucht ein Fazit über den momentanen Stand der Entwicklung zu ziehen.

2 Motivation und Hinführung

Benchmarks gibt es schon seit den 1980er Jahre, allerdings mit sich verändernden Zielen und Strukturen. Die ersten Benchmarks wie zum Beispiel der Wisconsin Benchmark [1] maßen die Leistung bei einzelnen Operationen, und halfen so dabei Leistungsengpässe und Optimierungsmöglichkeiten aufzufinden. Mit der Zeit haben sich der Funktionsumfang und die Implementierungen der verschiedenen Datenbanksysteme immer mehr aneinander angenähert. Dadurch wurde der Leistungs-Unterschied bei einfachen Operationen vernachlässigbar. Aktuelle Benchmarks bewerten deshalb ein System im Hinblick auf reale Anforderungen.

Die maßgebliche Institution für Benchmarks ist das “Transaction Processing Performance Council” (TPC) [2], das seit Jahren anerkannte Benchmarks entwickelt, und so der Wirtschaft verlässliche Daten zur Verfügung stellt.

Wenn man die verschiedenen Benchmarks des TPC im Überblick betrachtet, wird die Entwicklung weg von der Bewertung von Details und hin zu Anwendungsszenarien gut sichtbar:

- **TPC-A und TPC-B**

Die TPC-A und TPC-B Benchmarks führen die System-Operationen aus, die nötig sind, um die Aufgaben die mit online transaction processing (OLTP) verknüpft sind zu erfüllen. Die Resultate spiegeln die Festplatten-I/O Werte des Systems wieder, und geben Auskunft darüber wie viele User das System verkraftet.

Anmerkung: Vom TPC ausgemustert

- **TPC-C**

TPC-C ist ein viel komplexerer Benchmark als TPC-A und TPC-B, und der momentan meist benützte Benchmark. Beim TPC-C werden viele kurze Anfragen an ein Datenbanksystem gestellt. Die Ergebnisse spiegeln die Fähigkeit des Systems wieder Anforderungen von “wirklichen” Systemen wie die von Banken oder Kaufhäusern zu erfüllen.

- **TPC-D**

Der TPC-D Benchmark setzt sich aus komplexeren Queries zusammen als der TPC-C und zielt vor allem auf “Entscheidungsunterstützungs”-Systeme ab.

Anmerkung: Vom TPC ausgemustert

- **TPC-H und TPC-R**

Diese Benchmarks haben den TPC-D abgelöst, und messen die Performance mit einer noch größeren Anzahl von Anfragen.

- **TPC-W**

Der neueste TPC Benchmark. Basiert auf einer E-Commerce Anwendung.

Die Beschreibungen der Benchmarks sind hierbei etwas kurz gehalten, bei Interesse findet man unter [3] eine genauere Beschreibung.

Der oben genannte TPC-W Benchmark bildet auch die Grundlage für den im Anschluss vorgestellten XMark-1 Benchmark. Ein Problem bei einem solchen anwendungsspezifischen Ansatz ist, das Ziel nicht aus den Augen zu verlieren. Die Autoren des zweiten vorgestellten Ansatzes, dem XMark Benchmark, kritisieren am TPC-W gerade die überzogene Allgemeinheit:

“[...] das TPC-W Szenario ist zu generisch angelegt, und anstatt auf genau eine Anwendung zu passen, passt es auf keine.”

Deshalb wurde für XMark ein komplementärer Ansatz gewählt, der aber später noch im Detail vorgestellt wird.

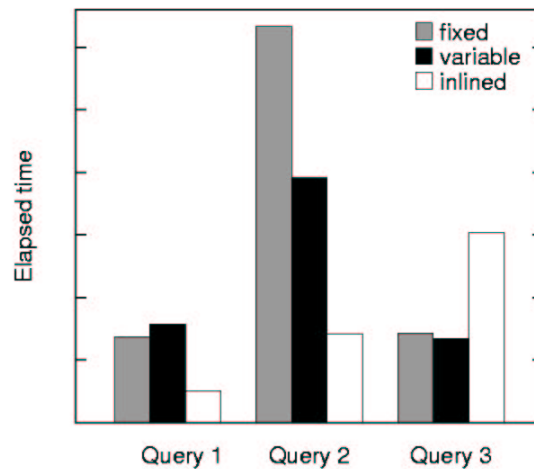


Abbildung 1: Anfragezeiten ausgesuchter Queries an ein relationales Datenbanksystem. Es wurden drei verschiedene Mappings von XML in eine rationale Struktur verwendet.

Abschließend sollten auch noch applikationsspezifische Benchmarks wie die von SAP oder PeopleSoft erwähnt werden. Hier wird die Leistung eines Systems nicht einzelstehend bewertet, sondern im Rahmen der Applikation eingebettet gemessen.

Egal welche Art von Benchmark man betrachtet, man kann sie alle zu verschiedenen Zwecken einsetzen, für die sie dann mehr oder weniger geeignet sind. Es wird bei beiden im folgenden vorgestellten Benchmarks jeweils abschließend darauf eingegangen, ob sie:

- geeignet sind für Produkt-Vergleiche
- geeignet sind um ein System zu tunen, also auf eine spezielles Szenario anwendbar sind
- zur Validierung eines Query-Prozessors, hinsichtlich der Implementierung XML-Standards, eingesetzt werden können

Es gibt also einige gute Gründe, Benchmarks durchzuführen. Bleibt die Frage, ob eine spezielle Performance-Messung für XML nötig ist.

Auch hier findet man mehrere gewichtige Argumente die den Aufwand rechtfertigen. Zunächst einmal haben bisher die XML-Data-Storage Produkte noch lange nicht einen Reifegrad erreicht, dass man bei allen von vollem Funktionsumfang, also Implementierung aller Query-Arten und Modifier, ausgehen kann. Benchmarks können hier durchaus noch funktionelle Unterschiede zwischen Produkten aufzeigen, zum Beispiel im Umfang der unterstützten Standards des Query-Prozessors. Speziell bei XML-erweiterten relationalen Systemen kommen noch zwei Punkte hinzu.

So kann es, je nach benutzter Umsetzung der XML-Daten in eine relationale Struktur zu sehr unterschiedlichen Ergebnissen kommen. In [6] kamen drei verschiedene Mappings zum Einsatz. Die Ergebnisse sind in Abbildung 1 zu sehen, sollen an dieser Stelle jedoch nicht näher erläutert werden, sondern sollen nur zeigen, dass die Differenzen durchaus bemerkenswert sind. Darüberhinaus ist gerade bei diesen Systemen mit sehr unterschiedlichen Rückgabestrukturen und -reihenfolgen zu rechnen, was speziell auf die Frage hinausläuft, wann die Antworten auf eine Anfrage äquivalent sind. Es wird also bei der Bewertung der vorgestellten Benchmarks auch mit einfließen, ob und wie sie solche "Feinheiten" wie die Reihenfolge der Datensätze berücksichtigen.

3 Vorstellung zweier Ansätze

3.1 XMach-1

3.1.1 Ansatz

Wie misst man die Performanz und die Kosten eines Servers in einer E-Commerce Umgebung? Der "XML Data *Management benchmark*" der Universität Leipzig versucht diese Frage für Systeme mit XML-Daten zu beantworten, und kommt zur selben Antwort wie sein relationales Gegenstück, der TPC-W Benchmark [8]. Um realistische Ergebnisse zu erhalten muß man die gesamte Umgebung realistisch simulieren, angefangen bei den Usern, die ihre Anfragen über ein Netzwerk stellen, über die Webserver, bis hin zur Datenbank. Die spontane Reaktion auf eine so weitgefaste Testumgebung ist zu fragen, was eigentlich bewertet wird.

Am besten macht man sich das Ziel einer Arbeit klar, indem man das Resultat analysiert. Gerade das fällt aber beim XMach-1 relativ schwer. Im Gegensatz zum TPC-W wird hier nämlich nicht näher erläutert, wie die erhaltenen Maßzahlen (gemessen in XML Queries per second (Xqps)) mit Hilfe anderer Daten wie CPU-Last oder Netzwerk-Durchsatz zu einer aussagekräftigen Bewertung der einzelnen Server umgerechnet werden können. Man wird wohl einfach auf die Veröffentlichung der ersten Ergebnisse warten müssen, um die Aussagekraft der ermittelten Daten zu bewerten.

3.1.2 Systemaufbau

Der XMach Benchmark basiert auf einer Web Anwendung, um einen typischen Anwendungsfall für ein XML-Data-Management System zu modellieren. Die Systemarchitektur besteht, wie in Abbildung 2 zu sehen ist, aus vier Teilen: die XML-Datenbank, Application Server, Loader und (Browser simulierende) Klienten. Gemessen werden hierbei nur Daten für das "System under Test" (SUT). Die Klienten und der Verbindungsaufbau mit den Application Servern bleiben hierbei unbetrachtet, so dass die entsprechenden Berechnungszeiten und Kommunikationsverzögerungen nicht auf die Bewertung des Systems abfärben.

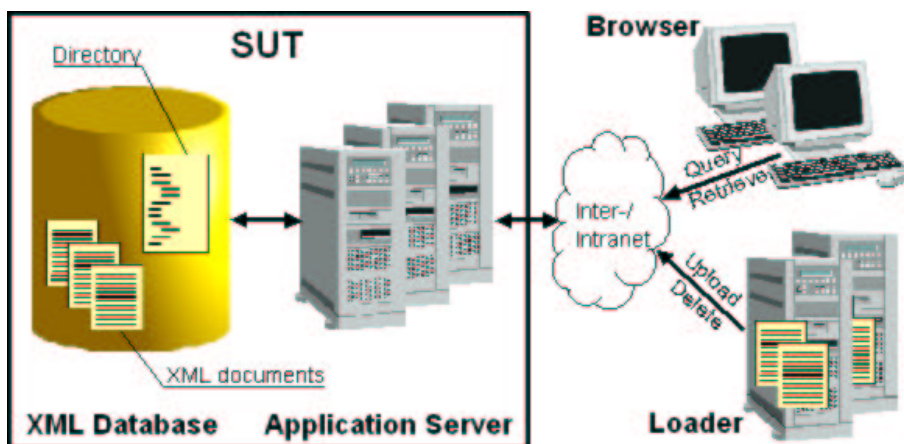


Abbildung 2: Komponenten der Benchmark Architektur

Die Anzahl der Datenbank- und Application Server ist nicht vorgegeben, sondern kann den Performanz-Zielen entsprechend gewählt werden. Anfragen an die Datenbank erfolgen nur von den Anwendungsserver aus. Caching beim Application Server ist hierbei erlaubt, solange die Klienten aktuelle Daten sehen.

3.1.3 XML-Anforderungen

Der XMach-1 umfaßt nicht alle vom W3C herausgegebenen bzw. bald erscheinenden Spezifikationen. So wird zum Beispiel keine Unterstützung für XML-Schema, Namespaces, CDATA Abschnitte, Entities oder eine bestimmte XML-Query Sprache vorausgesetzt. Auf diese Weise wird der Benchmark einfach gehalten, und kommt den momentan erhältlichen Systemen, die typischerweise längst nicht alle Features implementieren, entgegen. Als direkte Folge dieser Anpassungsfähigkeit werden zwei Versionen des Benchmarks unterschieden, je nachdem, ob die XML-Dokumente sich an DTDs bzw. Schemata halten oder nicht. Sollte Unterstützung für Schemata gegeben sein, kann so auch gleich deren Auswirkung auf die Leistung bewertet werden.

3.1.4 Struktur und Initialisierung der Datenbank

Um einer wirklichen Anwendung zu entsprechen enthält die Datenbank sowohl strukturierte Daten als auch Textdokumente.

Die **STRUKTURIERTEN DATEN** werden vertreten durch das sogenannte "Directory document". Es enthält Metadaten über alle gespeicherten Textdokumente und hält sich an das in Abbildung 3 gezeigte Schema. Somit ist es "well-formed" und "valid", und enthält Daten nur in Form von Elementen.

Diese Art von Dokument kommt in vielen XML-Anwendungsszenarien vor, zum Bei-

```

<!ELEMENT directory (host+) >
<!ELEMENT host (host+ | path+) >
<!ATTLIST host
  name CDATA #REQUIRED >
<!ELEMENT path (path+ | doc_info) >
<!ATTLIST path
  name CDATA #REQUIRED >
<!ELEMENT doc_info EMPTY >
<!ATTLIST doc_info
  doc_id ID #REQUIRED
  loader CDATA #REQUIRED
  insert_time NMTOKEN #REQUIRED
  update_time NMTOKEN #IMPLIED >

```

Abbildung 3: DTD für das Directory Document

```

<directory>
  <host name="com">
    <host name="test-company">
      <host name="www">
        <path name="products">
          <path name="overview.xml">
            <doc_info doc_id="2"
              loader="robot1"
              insert_time="20000110152530"/>
          </path>
        </path>
      </host>
    <host name="support">
      <path name="help.xml">
        <doc_info doc_id="3"
          loader="robot1"
          insert_time="20000612094430"/>
      </path>
    </host>
  </host>
</host>
</directory>

```

Abbildung 4: Beispiel für das Directory Document

spiel bei Produktkatalogen. Um sich das Ganze besser vorstellen zu können ist mit Abbildung 4 ein Beispiel für dieses Dokument gegeben. Die Reihenfolge ist hier nicht von Bedeutung, da es in jeder Hierarchieebene nur ein Element gibt.

Um realistische Leistungseinschätzungen zu erhalten schreiben die Autoren ein Verfahren vor, mit dem die **TEXTDOKUMENTE** erzeugt werden. Es reicht an dieser Stelle zu erwähnen, dass, mit Hilfe einer Wortliste, ein natürlicher Sprache entsprechender Text erstellt wird.

Zusätzlich enthalten die Dokumente sowohl externe als auch interne (zu anderen Dokumenten in der Datenbank) Verweise. Deshalb erhält jedes Dokument eine eindeutige URL. Diese wird wegen der nötigen Vorhersagbarkeit der Anfragen ebenfalls generiert.

In der schemabasierten Version von XMach-1 halten sich die Dokumente an das generische DTD in Abbildung 5. Da ein einziges DTD für alle Dokumente nicht ausreicht, werden mehrere DTDs simuliert, indem zu allen Attributen außer author und link Zahlen hinzugefügt werden. So würde das 17-te DTD die Elemente document17, chapter17, author, section17 usw. enthalten.

Auf einige weitere Besonderheiten, wie Einbettung von Suchsätzen oder weitergehende Attribute des generierten Textes, kann im Rahmen dieser Arbeit nicht eingegangen werden. Sie sind unter [4] nachzulesen.

Zu erwähnen bleibt an dieser Stelle noch, dass es vier mögliche Konfigurationen mit einer unterschiedlichen Anzahl an Datensätzen gibt, um eine bessere Skalierbarkeit des Benchmarks zu gewährleisten. Zur Auswahl stehen Startkonfigurationen mit 10.000, 100.000, 1.000.000. und 10.000.000 Datensätzen. Aufgrund der "insert" Operationen im Workload (siehe Abschnitt 3.1.5) steigt die Anzahl der Elemente während des Testlaufs. Vergleichbar sind natürlich nur Ergebnisse der gleichen Skalierungsstufe.

```
<!ELEMENT document (title, chapter+)>
<!ATTLIST document
  author CDATA #IMPLIED
  doc_id ID #IMPLIED>
<!ELEMENT author (#PCDATA)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT chapter (author?, head, section+)>
<!ATTLIST chapter
  id ID #REQUIRED>
<!ELEMENT section (head, paragraph+, section*)>
<!ATTLIST section
  id ID #REQUIRED>
<!ELEMENT head (#PCDATA)>
<!ELEMENT paragraph (#PCDATA | link)*>
<!ELEMENT link EMPTY>
<!ATTLIST link
  xlink:type (simple) #FIXED "simple"
  xlink:href CDATA #REQUIRED>
```

Abbildung 5: DTD für Textdokumente

3.1.5 Messung

Zur Performancemessung wird das System mit einer Mischung aus Anfrage- (siehe Tabelle 1) und Updateoperationen (siehe Tabelle 2) getestet. Je nach Funktionsumfang des XML-Storage Systems müssen hierfür Operationen (zum Beispiel Joins) in der Applikationsschicht implementiert werden.

ID	Description	Comment
Q1	Get document with given URL	Return a complete document (complex hierarchy with ordering preserved)
Q2	Get doc_id and URL from documents containing a given phrase in paragraph elements	Text retrieval query. The phrase is chosen from the phrase list. Join needed to get URL for qualifying documents
Q3	Start with first element which name starts with 'chapter' and recursively follow first element which name starts with 'section'. Return each of the 'section' elements	Simulates navigating a document tree using sequential operators
Q4	Return flat list of head elements which are children of elements whose names start with 'section' from a document given by doc_id	Restructuring operation simulating creation of a table of contents
Q5	Get document name (last path element in directory structure) from all documents which are below a given URL fragment	Browse directory structure. Operation on structured unordered data
Q6	Get doc_id and id of parent element of author element with given content	Find chapters of a given author. This tests efficiency of index implementation
Q7	Get doc_id from documents which are referenced by at least four other documents	Get important documents. Needs some kind of group by and count operation
Q8	Get doc_id from the last 100 updated documents having an author attribute	Needs count, sort, join and existential operations and accesses metadata

Tabelle 1: XMach-1 Queries

Gemessen werden die Antwortzeiten bei den Application-Servern, und zwar ab dem Zeitpunkt der Anfrage (vom Klienten) bis zum Abschicken des Resultats im XML-Format.

Die Ergebnisse werden dann im Schema-basierten Fall in $Xqps$ (*XML queries per second*) und sonst in $Xqps_{sl}$ (*Xqps schema less*) gemessen. Es werden hierbei nur die ausgeführten Q1 Anfragen gemessen, da die Zusammensetzung des Workloads, also die jeweilige Anzahl der verschiedenen Queries, festgelegt ist. Als Nebenbedingung wird noch festgelegt, dass für die Queryarten Q1-Q8 und M3 90 Prozent der Operationen in unter 3 Sekunden ausgeführt werden. Bei den insert und delete Operationen wird die 90-Prozent Zeitspanne auf 20 Sekunden festgelegt. Ohne

diese Bedingungen würde es leicht fallen, den Durchsatz auf Kosten der Antwortzeiten künstlich zu erhöhen. Die Kosteneffektivität kann berechnet werden, indem man die Kosten für das SUT ermittelt und durch die $Xqps$ bzw. $Xqps_{sl}$ teilt. Für eine genauere Beschreibung der Kostenberechnung wird auf den TPC-W Benchmark verwiesen.

ID	Description	Comment
M1	Insert document with given URL	The loader generates a document and URL and sends them to the HTTP server
M2	Delete a document with given doc_id	A robot requests deletion, e.g. because the corresponding original document no longer exists on the web
M3	Update URL and update_time for a given doc_id	Update directory entry

Tabelle 2: XMach-1 Updates

3.1.6 Bewertung

Wo andere Benchmarks auf einen einzelnen Server abzielen und versuchen soviel Störquellen wie möglich auszuschließen, wird beim XMach-1 das System als Ganzes bewertet. Dies macht aber gerade im XML-Umfeld sehr wohl Sinn, da zum Beispiel XML-Daten auch beim Applikationsserver in ein relationales Format überführt und bis dahin in einer "normalen" relationalen Datenbank gespeichert werden könnten. Der XMach-1 bietet somit die größte mögliche Flexibilität.

Es leuchtet ein, dass sich der vorgestellte Benchmark aufgrund der vielen möglichen Konfigurationen (zum Beispiel Caching siehe Abschnitt 3.1.2 auf Seite 5) kaum zum schnellen Produktvergleich eignet. Genausowenig ist er geeignet, um den Stand der XML-Standards eines Systems zu überprüfen, da er deren Implementierung teilweise dem Tester überlässt. Es erscheint wahrscheinlicher, dass er ähnlich dem TPC-W von Firmen eingesetzt wird, um ihre Dienste besser zu konfigurieren und zu tunen.

3.2 XML Benchmark Project: Xmark

3.2.1 Ansatz

Der Xmark Benchmark wurde von einem unabhängigen Forscherteam entwickelt. Er ist jüngeren Datums als der XMach-1 und vertritt bewusst einen anderen Ansatz als dieser.

Im Gegensatz zu XMach-1 werden Netzwerkverzögerungen, Verbindungsaufbau oder Umwandlungen komplett außen vor gelassen, mit der Begründung, dass man nur den Query Prozessor und seine Interaktionen mit den Daten bewerten will. Weiterhin verzichten die Autoren im Workload auf "update" Operationen, da es für diese keinen Standard gibt. Es werden verschiedene Klassen von Queries benützt, um die Stärken und Schwächen einer XML-Datenbank in den entscheidenden Bereichen aufzudecken. Jede Anfrage soll den Query Prozessor hierbei auf einen elementaren Bestandteil der Querysprache testen. Die Autoren versprechen sich hiervon zwei Vorteile.

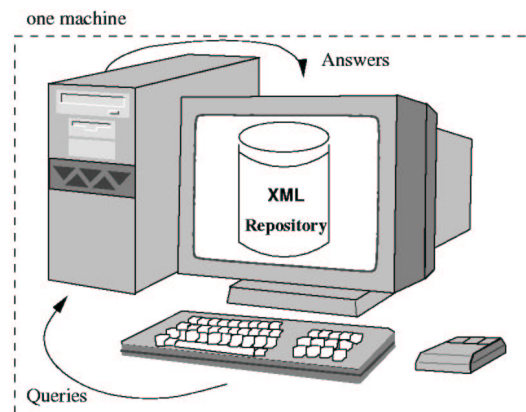


Abbildung 6: Xmark - Systemaufbau

Erstens ist der Benchmark nicht auf einen Bereich, wie im Falle von XMach-1 den E-Commerce, festgelegt. Und zum zweiten kann Xmark eingesetzt werden, um Query Prozessoren in ihrem Funktionsumfang zu verifizieren

Die Testdaten simulieren ein Auktionshaus und werden in Abschnitt 3.2.4 näher beschrieben. Um realistische Dokumente zu verwenden wird ein Generator zur Verfügung gestellt, der im Folgenden jedoch nur kurz angesprochen wird.

3.2.2 Systemaufbau

Wie in Abbildung 7 gut zu sehen, ist der Systemaufbau des Xmark sehr einfach. Das Testsystem besteht, wie man es von klassischen Benchmarks gewohnt ist, nur aus dem zu testenden Server. Belastung des Systems durch Klientenanfragen wird hierbei nur lokal simuliert.

3.2.3 XML-Anforderungen

Im Gegensatz zu XMach-1 legen sich die Autoren von Xmark auf XQuery als Anfragesprache fest. Sie verzichten wie schon erwähnt auf Updates und einige andere Features von XML, die für die Bewertung nicht von Bedeutung sind. Es werden keine Dokumente mit Entities oder Notations erzeugt. Genausowenig wird zwischen Parsed Character Data und Character Data unterschieden, da beide für den Datenspeicher nur Zeichenketten sind. Darüberhinaus finden beim Xmark, genau wie beim XMach-1, keine Namespaces Verwendung.

Auch beim Xmark werden für die Dokumente DTDs und darüberhinaus XML Schemainformationen bereitgestellt. Diese können von den getesteten Systemen genutzt werden, um effizientere Mappings zu generieren, müssen jedoch nicht. Die Bewertung macht keinen Unterschied zwischen Systemen, die diese Informationen nutzen oder nicht, es wird nur die Leistung bewertet.

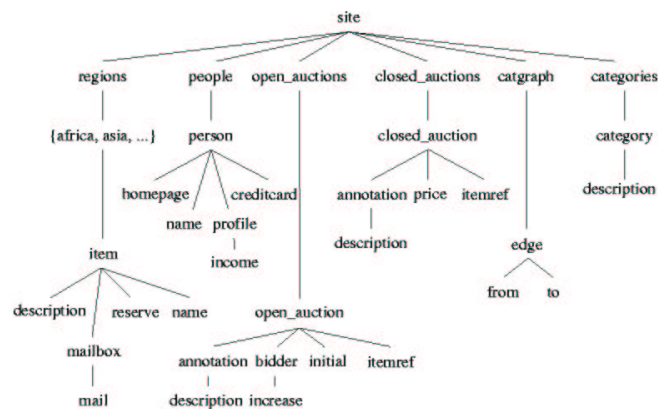


Abbildung 7: Beziehungen zwischen den meisten Entitäten

3.2.4 Struktur und Initialisierung der Datenbank

Wie oben schon erwähnt ist die Datenbank einem Auktionshaus nachempfunden. Die Hauptentitäten sind hierbei: person, open auction, closed auction, item und category. Eine detaillierte Beschreibung der Entitäten findet man in Abschnitt 3.1 von [7], sie ist hier jedoch nicht von Bedeutung.

Bei dem Entwurf der Datenbank wurde speziell darauf geachtet, dass vier Punkte an ihr gut zu überprüfen sind:

- Bei der Anfrage kann textuelle *Ordnung* des Resultats verlangt werden. Dies kann dafür nicht gerüsteten Systemen Probleme bereiten.
- Strings sind die Basistypen, sie können eine sehr unterschiedliche Länge haben und schon dadurch hohe Anforderungen an ein System stellen. Darüberhinaus kann es zu *Typproblemen* kommen.
- Queries, die *hierarchische Strukturen* beinhalten, spezielle lange Pfadanfragen über 1:n Beziehungen, sind von Systemen, die relational arbeiten, oft nur über sehr aufwendige und zeitintensive Joins aufzulösen.
- Das *lockere Schema* vieler XML-Dokumente schließlich stellt sowohl an den User als auch an die Datenbank nicht geringe Anforderungen.

Diese aufgegliederte Herangehensweise werden wir im Abschnitt 3.2.5 wiederfinden, wenn es um die Einteilung der Queries in Klassen geht.

Um zu gewährleisten, dass der Benchmark wirklich auf realistischen Daten läuft stellen die Entwickler mit xmlgen ein Tool zur Generierung des XML-Dokuments zur Verfügung. Es wurde speziell in Hinblick auf Plattformunabhängigkeit, Skalierbarkeit und Effizienz entwickelt und ist

in [11] näher spezifiziert. In Tabelle 3 sind die verschiedenen Skalierungsstufen und die korrespondierenden Dokumentgrößen festgehalten. Da manche Systeme mit übergroßen Dateien Probleme haben, kann das Dokument vom Generator auch in mehrere kleine Dateien aufgespalten werden.

Name	Scaling Factor	Document Size
tiny	0.1	10 MB
standard	1	100 MB
large	10	1 GB
huge	100	10 GB

Tabelle 3: Skalierung

3.2.5 Messung

Der Workload besteht beim Xmark aus 20 Anfragen, die, wie schon oben erwähnt, in mehrere Klassen aufgeteilt sind. Es folgt eine kurze Auflistung der Klassen mit jeweils einer umgangssprachlich formulierten Anfrage (aus dem englischen Original zitiert) als Beispiel.

Queryart	Queries	Beispiel
Punktanfrage	Q1	Q1: Return the name of the item with ID 'item20748' registered in North America.
Geordneter Zugriff	Q2 - Q4	Q2: Return the initial increases of all open auctions.
Casting	Q5	Q5: How many sold items cost more than 40?
Reguläre Pfadausdrücke	Q6, Q7	Q6: How many items are listed on all continents?
Referenzen verfolgen	Q8, Q9	Q8: List the names of persons and the number of items they bought. (Join von person mit closed auction).
Konstruktion komplexer Resultate	Q10	Q10: List all persons according to their interest; use french markup in the result.
Joins auf Werte	Q11, Q12	Q11: For each person, list the number of items currently on sale whose price does not exceed 0,02% of the person's income.
Rekonstruktion des Originaldokuments	Q13	Q13: List the names of items registered in Australia along with their descriptions.

Volltext	Q14	Q14: Return the names of all items whose description contains the word 'gold'.
“Path Traversals”	Q15, Q16	Q15: Print the keywords in emphasis in annotations of closed auctions.
Fehlende (optionale) Elemente	Q17	Q17: Which persons don't have a homepage?
Anwendung user-definierter Funktionen	Q18	Q18: Convert the currency of the reserve of all open auctions to another currency.
Sortierung	Q19	Q19: Give an alphabetically ordered list of all items along with their location.
Aggregation	Q20	Q20: Group customers by their income and output the cardinality of each group

Tabelle 4: XMark - Queries

In [7] ist für jede Klasse noch in ein oder zwei Sätzen ausführlicher angegeben, was genau geprüft wird, und in welchen Anwendungsfällen es genutzt wird. Man kann sich also überlegen, welche dieser Klassen für die eigene Anwendung von Bedeutung sind, und diesen dann bei der Bewertung mehr Gewicht zumessen. Bei der Messung wird für jede Query die mittlere Antwortzeit in Millisekunden und die Anzahl der Wiederholungen festgehalten. In Tabelle 5 kann man die Messergebnisse für das Monet XML System sehen, das bisher als einziges bewertet wurde. Ein Vergleich verschiedener Systeme wäre also einfach die Spalten der Tabelle miteinander zu vergleichen. Auch hier gilt natürlich, dass nur Ergebnisse der selben Skalierungsstufe vergleichbar sind. In [7] finden sich in Abschnitt 6 auch noch einige interessante Bemerkungen zu den Messergebnissen, die aber den Rahmen dieser Arbeit sprengen würden.

3.2.6 Bewertung

Beim Xmark fällt es nicht schwer sich der Einschätzung der Autoren anzuschließen. Nach deren Ansicht eignet sich der Benchmark erstens gut zum Verifizieren und Verbessern existierender Queryprozessoren. Zum zweiten ist er aufgrund seiner Einfachheit (und deshalb relativ wenigen Manipulationsmöglichkeiten) gut für Produktvergleiche geeignet. Abschließend schlagen sie noch vor, ihn in der Forschung dazu einzusetzen, existierende Techniken effizient für XML umzusetzen. In diesem Punkt wird sich wohl noch zeigen müssen in wie weit das möglich ist. Wenig geeignet erscheint der Xmark nur für das Feintuning einer eigenen Anwendung, hierfür fehlt die beim XMach-1 vorgefundene Flexibilität.

Query	RP	Monet XML
Query 1	1000	1217 ms
Query 2	1	2945 ms
Query 3	1	3891 ms
Query 4	1	153 ms
Query 5	1	161 ms
Query 6	1000	762 ms
Query 7	1000	2167 ms
Query 8	1	469 ms
Query 9	1	977 ms
Query 10	1	22292 ms
Query 11	1	8672 ms
Query 12	1	7464 ms
Query 13	1	25 min
Query 14	1	9223 ms
Query 15	100	762 ms
Query 16	100	2018 ms
Query 17	1000	250 ms
Query 18	100	7799 ms
Query 19	1	493 ms
Query 20	10	346 ms

Tabelle 5: Monet

4 Weitere Ansätze

Verblüffenderweise findet man selbst nach ausführlicher Recherche keine weiteren Vorschläge, geschweige denn fertige Produkte zur XML-Performanzmessung. Die beiden vorgestellten Ausarbeitungen sind in diesem Bereich Vorreiter und sind wohl auch nicht als Konkurrenz zueinander zu sehen sondern als komplementäre Ansätze.

Zu einem oben schon angesprochenen Thema findet sich mit [9] noch ein sehr guter Artikel. Hier werden fünf verschiedene Mapping Schemata vorgeschlagen, mit denen man XML Daten in eine relationale Datenbank überführen kann. In dem Artikel werden nicht nur die Geschwindigkeitsunterschiede zwischen den einzelnen Ansätzen festgehalten. Es wird vielmehr auch auf das Drumherum geachtet. So wird die Größe der aus der Transformation resultierenden Datenbank genauso einbezogen, wie die Zeit die es dauert das ursprüngliche Dokument wieder zu erhalten. Wie in der Motivation schon angesprochen ist es natürlich reizvoll, die vorhandenen RDBM Systeme auch für XML einzusetzen. In [9] wird nun gezeigt, dass diese Lösung mit Hilfe des richtigen Mapping Schemata durchaus auch performant ist. Bei diesem Ansatz ergeben sich jedoch auch einige Nachteile. So ist es (zeitlich) kostspielig das original Dokument wieder aus der Datenbank zu extrahieren. Auch Updates sind teilweise sehr kompliziert, was auch an fehlenden Standards liegen mag. Es sei hier noch auf den Artikel [10] verwiesen, in dem näher auf die verschiedenen Mappings eingegangen wird.

5 Fazit und Ausblick

Der praktische Nutzen der momentan vorhandenen Ansätze kann erst eingeschätzt werden, sobald umfangreiche Messungen durchgeführt wurden. Sie bilden aber jetzt schon eine Basis für Forscher und Entwickler um ihre Arbeit zu bewerten....

Ein großer Schritt für die Benchmarks kündigt sich an, sobald die neuen XML-Standards vom W3C verabschiedet werden. Speziell eine vollständige Spezifikation für Updates wurde in beiden vorgestellten Ansätzen als dringend notwendig bezeichnet.

Die Autoren von [9] haben angekündigt ihren Benchmark in nächster Zeit auch für OODB- und native XML-Systeme durchzuführen. Dies wirft, zumindest für mich, die Idee auf, auch einmal einen Benchmark zu entwickeln, der die Techniken direkt vergleicht. So stellt sich einem schon manchmal die Frage, warum man unbedingt auf XML Daten arbeiten muss, diese auf der einen Seite in HTML überführt, und auf der anderen Seite in einer relationalen Datenbank speichert. Oft werden die Daten in ihrer XML Form nicht benützt. Natürlich wirft es gewisse Probleme auf, die Systeme wirklich vergleichbar zu halten. Es sollte aber nicht schwer sein, Ergebnisse von TPC-W und XMach-1 zu vergleichen und so eventuell Aufschluss darüber zu erhalten, ob für das spezielle Szenario nicht einfach eine der beiden Techniken besser geeignet ist.

Literatur

- [1] *The Benchmark Handbook: Wisconsin Benchmark*, <http://www.benchmarkresources.com/handbook/4.html>
- [2] *Transaction Processing Performance Council*, <http://www.tpc.org/>
- [3] *Transaction Processing Performance Council*, <http://www.tpc.org/information/benchmarks.asp>
- [4] Universität Leipzig: *XMach-1: A Benchmark for XML Data Management*, <http://dbs.uni-leipzig.de>, Leipzig, Dezember 2000
- [5] Universität Leipzig: *Benchmarking XML Database Systems - First Experiences*, <http://dbs.uni-leipzig.de>, Leipzig
- [6] A. Schmidt, M. Kersten, D. Florescu, M. Carey, I. Manolescu and F. Waas, R. Busse: *Why And How To Benchmark XML Databases*, 2001
- [7] A.Schmidt, M.Kersten, D.Florescu, M.Carey, I.Manolescu and F.Waas: *The XML Benchmark Project*, 30. April 2001
- [8] *Description of TPC-W*, <http://www.tpc.org/tpcw/default.asp>
- [9] Daniela Florescu, Donald Kossmann. Unité de recherche INRIA Rocquencourt: *A Performance Evaluation of Alternative Mapping Schemes for Storing XML Data in a Relational Database*, Mai 1999
- [10] Carola Langwald, Universität Ulm: *Verschiedene Mappings von XML in ein Relationenschema*, Februar 2001
- [11] *xmlgen – The Benchmark Data Generator*, <http://monetdb.cwi.nl/xml/Benchmark/benchmark.html>