Expose zur Studienarbeit

Indizierung von XML-Daten mittels GRIPP

Betreuer: Silke Trissl

Autor: Florian Zipser
eMail: zipser@informatik.hu-berlin.de

1 Motivation

Auf dem Gebiet der relationalen Datenbanken wurden Konzepte wie Zugriffskontrollen, Backupsysteme und Steigerungen der Performance auf verschiedenen Ebenen entwickelt. Da diese Konzepte auch für die Datenspeicherung mittels XML eingesetzt werden können, ist es sinnvoll, die Vorteile relationaler Datenspeicherung mit denen der XML-Datenspeicherung zu verbinden.
Um die Vorteile eines RDBMS nutzen zu können und einen schnellen Datenzugriff auf die hierarchischen Daten innerhalb der relationalen Struktur zu ermöglichen, sind geeignete Indizierungsverfahren, die auf die Struktur der zugrunde liegenden XML-Daten eingehen, notwendig.
Hierfür wurde die Indizierung mittels der Pre- und Postorder bzw. stretched Pre- und Postorder (siehe [1] bzw. [2]) entwickelt. Diese bezieht sich auf die Position der einzelnen Elemente innerhalb des als Baum dargestellten XML-Dokumentes. Allerdings können damit nur XML-Dokumente indiziert werden, deren Struktur der eines Baumes entspricht. Mit der durch das W3C entwickelten Erweiterung von XML um XPointer (siehe [3]) wird diese Baumstruktur verletzt. Weiter ermöglichen es die XPointer rekursive Strukturen anzulegen. Um diese dennoch indizieren zu können, kann die für die Indizierung von Graphen entwickelte Methode GRIPP (siehe [4]) auf XML-Dokumente angewandt werden.
GRIPP basiert auf der Indizierung durch Pre- und Postorder. Um die Lücke zwischen Bäumen und Graphen zu schließen, bildet GRIPP einen gegebenen Graphen auf einen Baum ab, indem es Kanten, die nicht zum spannenden Baum gehören, entfernt und stattdessen künstliche Knoten anlegt. Diese künstlichen Knoten werden als Hopknoten bezeichnet und enthalten einen Verweis auf den Zielknoten der Kante. Die Suche innerhalb des so entstandenen Baumes kann also mittels der Pre- und Postorder erfolgen, solange kein Hopknoten zur Quellknotenmenge der Anfrage gehört. Ist dieses aber der Fall, so erweitert sich die Suche auf die Zielknoten der entsprechenden Hopknoten. Hier ist es möglich, mittels Pruningstrategien einen Performancegewinn zu erzielen.
Wie bereits erwähnt wurde, basieren XML-Dokumente auf einer Baumstruktur und nur die Erweiterung um XPointer verstößt gegen diese. Daher ist es möglich, die Knoten des XML-Dokumentes ohne die XPointer als spannenden Baum anzusehen und jeden XPointer als Hopknoten zu verstehen. Diese enthalten wie die ursprünglichen XPointer als Ziel den Knoten, auf den sie verweisen. So ist es möglich, das Prinzip von GRIPP auf XML-Daten zu übertragen.

2 Zielsetzung

Das Ziel dieser Studienarbeit ist es, die Performance der Indizierung von relational gespeicherten XML-Dokumenten mittels GRIPP zu testen. Die Ergebnisse sollen anschließend mit denen nativer XML-Datenbanken oder relationaler Datenbanken, mit der Möglichkeit, XML-Daten zu speichern, verglichen werden. Es ist auch ein Ziel dieser Arbeit einen „Vergleichspartner“ für die GRIPP Methode zu finden, der in der Lage, ist XML-Daten mit XPointern korrekt und vollständig zu beantworten.

3 Herangehensweise

Um die Zielsetzung zu erreichen, gliedert sich diese Studienarbeit in vier Bereiche:

1. Datenbeschaffung
   Zunächst müssen hinreichend große XML-Dokumente, auf denen die Indizierung laufen soll, beschafft werden. Die Basis dieser Dokumente sollen einerseits natürliche, also reale Daten und andererseits generierte Daten sein.
2. Daten einlesen
   Weiter muss ein Parser entwickelt werden, der diese Daten in eine relationale Datenbank einliest und dabei die Indizierung vornimmt.
3. Anfragen
   Abschließend sind Anfragen über die in 1 gefundenen XML-Daten zu entwickeln, die mit Hilfe der Indizierungsmethode GRIPP zu beantworten sind.
4. Vergleichspartner
   Es muss ein DBMS gefunden werden, das in der Lage ist, Anfragen auf den beschafften Daten zu beantworten und dabei XPointer zu verfolgen. Wird ein solches DBMS gefunden, so sind die in 3 entwickelten Anfragen auch mit diesem System zu beantworten. Anschließend wird die Performance zwischen dem gefundenen System und der GRIPP-Methode verglichen.

3.1 Datenbeschaffung

Für die späteren Performancetests der GRIPP Indexstruktur werden zwei unterschiedliche Datenarten benötigt, einerseits sind dies reale Daten und andererseits generierte Daten. Ein Teil der Daten der realen Welt stammt aus dem Projekt DBLP der Universität Trier (siehe [5]) und liegt in Form eines ca. 330 MB großen XML-Dokumentes vor. Das Dokument beinhaltet Informationen über wissenschaftliche Publikationen wie den Titel, den Autor, das Erscheinungsjahr, die Erscheinungsform u.ä. .
Weitere Datenquellen könnten die Filmdatenbanken Gios Movie (siehe [6]), oder auch IMDB (siehe [7]) stellen.
Die Generierung der künstlichen Daten soll mittels eines XML-Generators erfolgen. Dieser soll es möglich machen, große XML-Daten automatisch zu generieren. Wichtig ist hierbei die Möglichkeit, die zu erzeugenden Daten variabel gestalten zu können, beispielsweise durch Angabe eines XML-Schemas. Weiterhin ist es von Vorteil, wenn der Generator in der Lage ist, XPointer automatisch in den erzeugten Dokumenten unterzubringen. Wird kein Generator gefunden, der automatisch XPointer erzeugt, so ist ein Programm zu schreiben, das nachträglich diese Aufgabe erledigt.

3.2 Daten einlesen

Der zweite Bereich beschäftigt sich mit dem Einlesen der zuvor gewonnenen XML-Dokumente in eine relationale Datenbank. Hierfür ist ein Programm in Java zu entwickeln, das die XML-Dokumente parst und modellbasiert in die relationale Datenbank schreibt. Außerdem ist die

Indexstruktur während des Parsens mit zu generieren, das heißt, neben den enthaltenen Daten müssen Positionsinformationen der XML-Elemente wie die Pre- und Postorder generiert und mitgespeichert werden.
Die Behandlung von XPointern in einem Dokument stellt hier eine Schwierigkeit dar, da ein XPointer auf ein Element zeigen kann, das bisher noch nicht eingelesen wurde. Aus diesem Grund muss eine geeignete Struktur entwickelt werden, mit solchen XPointern umzugehen.
Aufgrund der Größe der XML-Dokumente scheint es sinnvoll, als geeigneten XML-Prozessor die Simple API for XML (SAX) zu wählen. Diese liest ein gegebenes XML Dokument sequentiell ein, ohne den Inhalt, z.B. in Form einer Baumstruktur, zu speichern. Das bietet den Vorteil eines im Vergleich zu DOM wesentlich geringeren Hauptspeicherbedarfs während des Parsens.

3.3 Anfragen

Anschließend werden Anfragen für die im ersten Bereich gewonnenen XMLDokumente entwickelt. Diese Anfragen sollen Grundlage eines Vergleiches zwischen der Anfragebeantwortung einer relationalen Datenbank mit dem Indizierungsverfahren GRIPP und einer in Bereich 3.4 gefundenen Datenbank sein. Hierbei handelt es sich um die in [4] beschriebenen Erreichbarkeitsanfragen. Um die Indexstruktur GRIPP bei der Anfragebearbeitung der relationalen Datenbank nutzen zu können, müssen geeignete Funktionen geschrieben werden, die sich in die relationale Zieldatenbank einbetten lassen. Hierfür bietet sich die Sprache PL/SQL an.

3.4 Vergleichspartner

Ein DBMS wird gesucht, das in der Lage ist, XML-Daten zu speichern und die für die Daten entwickelten Anfragen korrekt und vollständig zu beantworten. Wird ein solches System gefunden, dann werden die in Bereich 3.3 erarbeiteten Anfragen auch in dem System getestet. Dabei spielt es keine Rolle, ob es sich um ein natives XML-Datenbanksystem oder um ein RDBMS mit XML-Erweiterung handelt. Um dieses System anschließend mit GRIPP vergleichen zu können, muss es jedoch in der Lage sein, XPointern zu folgen. Mögliche Datenbanksysteme für diesen Vergleich sind: Tamino, eXist, Natix, dbXML, xHive, DB2 sowie Oracle mit XML-Erweiterung.

Literatur

1. T. Grust and M. van Keulen. Tree Awareness for Relational DBMS Kernels: Staircase Join. *Intelligent Search on XML Data*, pages 231–245, 2003.
2. Torsten Grust, Maurice van Keulen, and Jens Teubner. Accelerating XPath evaluation in any RDBMS. *ACM Trans. Database Syst.*, 29:91–131, 2004.
3. XML Pointer Language (XPointer). Technical report, January 2001.
4. Silke Trissl and Ulf Leser. Efficient Execution of Reachability Queries on Very Large Graphs. Technical report.
5. Michael Ley. DBLP - Digitales Bibliographie und Bibliotheksprojekt. http://www.informatik.uni-trier.de/ ley/db/.

6. Gio Wiederhold. Gio's Movie. http://www-db.stanford.edu/pub/movies/doc.html.
7. Internet Movie Database Inc. IMDB - Earths Biggest Movie Database. http://german.imdb.com/.