3. Übungsblatt

Ausgabe: 6. November 2013 · Besprechung: 13. November 2013

1 Partitioned B-Trees

Das Zusammenspiel von (lexikographischer) Sortierung und B-Bäumen lässt sich für verschiedene Aufgaben geschickt ausnutzen, jenseits des bloßen Indizierens von Tabellen. Goetz Graefe [1] beschreibt z. B. die Idee der "Partitioned B-trees" und ihre Verwendung für *bulk updates*.

1. Beschreiben Sie die Idee der "Partitioned B-trees".
2. Wie müssen Index-Leseoperationen geändert werden, um "Partitioned B-trees" zu realisieren?

2 B-Baum-Anwendungsszenario: XML-Verarbeitung

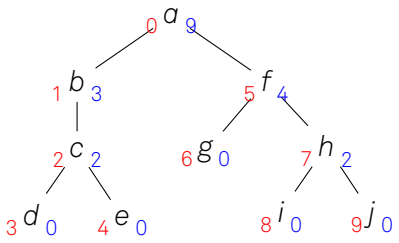
Relationale Datenbanken eignen sich auch als effiziente Back-Ends für XML-Daten. Ein Ansatz ist dabei, jedem Knoten des XML-Baums zwei Zahlenwerte zuzuweisen, nämlich

pre(*v*)**:** die Position des Knotens *v* in einem Präorder-Baumdurchlauf und

size(*v*)**:** die Größe des Teilbaums unterhalb eines Knotens *v*.

Für ein kleines Beispieldokument ergeben sich etwa folgende Werte für *pre*(*v*) und *size*(*v*):

```
<a>
  <b><c><d/>e</c></b>
  <f>
    <g/>
    <h>i<j/></h>
  </f>
</a>
```



(hier sind *e* und *i* jeweils Textknoten, alle anderen Knoten sind XML-Elemente).

Die Zahlenwerte *pre*(*v*) und *size*(*v*) können dann genutzt werden, um das Dokument (verlustfrei) in einer Datenbanktabelle abzulegen:

pre	*size*	*kind*	*value*	*level*
0	9	elem	`a`	1
1	3	elem	`b`	2
2	2	elem	`c`	3
3	0	elem	`d`	4
4	0	text	`e`	4
5	4	elem	`f`	2
6	0	elem	`g`	3
7	2	elem	`h`	3
8	0	text	`i`	4
9	0	elem	`j`	4

Hierbei wurden neben den Werten $pre(v)$ und $size(v)$ auch noch der semantische Inhalt jedes Knotens (der Knotentyp *kind* sowie der Tagname/Inhalt des Knotens *value*) mit abgespeichert. Außerdem wurde in der Spalte *level* die Ebene jedes Knotens protokolliert, was sich für einzelne XPath-Anfragen als hilfreich erweist (z. B. für die `child`-Achse).
In diesem Format abgelegt, können *XPath Location Steps* (die Mengen von Knoten aus dem Baum selektieren sollen) elegant ausgedrückt werden. Beispielsweise lassen sich die Ergebnismengen eines XPath-Schritts ausdrücken durch Prädikate, die wiederum in der Tabellendarstellung ausgewertet werden können:

$$v' \in v\texttt{/descendant::node()} \Leftrightarrow pre(v') > pre(v) \wedge pre(v') < pre(v) + size(v)$$

oder

$$v' \in v\texttt{/descendant::f} \Leftrightarrow pre(v') > pre(v) \wedge pre(v') < pre(v) + size(v) \wedge kind(v') = \text{elem} \wedge value(v') = \texttt{f} \ .$$

Gesamte XPath-Ausdrücke resultieren dann in einem wiederholten *Self-Join* mit der oben gezeigten Tabelle (wir nennen sie im folgenden `tree`). Zum Beispiel:

```
doc("foo.xml")/descendant::f/descendant::text()
                              ⇔
SELECT t.*
  FROM tree r, tree f, tree t
 WHERE r.pre = 0   -- root node
   AND f.pre > r.pre  AND  f.pre < r.pre + r.size
   AND f.kind = elem  AND  f.value = 'f'
   AND t.pre > f.pre  AND  t.pre < f.pre + f.size
   AND t.kind = text
```

1. Überzeugen Sie sich, dass die angegebenen Prädikate bzw. die SQL-Anfrage auch wirklich dem angegebenen XPath-(Teil-)Ausdruck entspricht.
2. Wie könnte ein sinnvoller Anfrageplan für die obige SQL-Anfrage aussehen?
3. Wie würde die Prädikate aussehen für
   (i) $v' \in v\texttt{/child::h}$
   (ii) $v' \in v\texttt{/following::*}$

(iii) $v' \in v$`/parent::node()` ?

4. Wie ließen sich dann folgende XPath-Anfragen in SQL ausdrücken?

   (i) `doc("foo.xml")/descendant::h/child::j`

   (ii) `doc("foo.xml")/descendant::text()/parent::*`

   (iii) `doc("foo.xml")/descendant::f[child::h]`

Ein großer Vorteil einer Kodierung von XML-Daten in einer relationalen Datenbank besteht darin, dass bestehende Indextechniken, insbesondere B-Bäume direkt wiederverwendet werden können.

5. Um die oben diskutierten Anfragen (bzw. XPath-Prädikate) möglichst effizient auszuwerten, welchen (B-Baum-) Index würden Sie jeweils vorschlagen, d. h. auf welchem Attribut oder auf welcher Attributkombination würde Sie einen Index anlegen?

Literatur

[1] Goetz Graefe. Partitioned B-trees - a user's guide. In *Datenbanksysteme für Business, Technologie und Web (BTW)*, pages 668--671, Leipzig, Germany, February 2003.