Exposé zur Diplomarbeit

Optimierung von Präfix-Baum-Indizes für String-Attribute

Martin Knobloch

Zeitraum
März –August 2007

Motivation

Die Nutzung von Zeichenketten in Datenbanken ist weit verbreitet. Angefangen bei Namen über Adressen bis hin zu Texten werden auch größere Informationen in Form von Zeichenketten gespeichert. Da heutige Datenbanken Größen im Terabyte-Bereich erreichen, ist ein effizientes Auffinden der gespeicherten Informationen notwendig. Hierfür existieren neben den eigentlichen Datentabellen Zugriffsstrukturen, mit deren Hilfe schnell Verweise auf die gesuchten Daten gefunden werden sollen. Die gebräuchlichste Zugriffsstruktur ist der B*-Baum [2,3], ein balancierter und per se E/A-Optimierter Baum. Die Daten werden durch die Operatoren *Kleiner* / *Größer* und *Äquivalenz* in Knoten und Blätter des Baumes einsortiert. Alle Blätter des Baumes hängen auf gleicher Tiefe, zusammen mit einer Verknüpfung der Blätter ergibt sich eine Sortierung über dem indexierten Attribut. Eine Besonderheit von Zeichenketten gegenüber numerischen Attributen liegt jedoch darin, dass diese über eine variable Länge verfügen. Somit kann allein der Vergleich von String-Attributen eine erheblich größere Komplexität erreichen als bei primitiven Datentypen.

In seiner Diplomarbeit [1] geht Nicky Hochmuth auf diese Probleme ein und stellt den Präfix-Index als Zugriffsstruktur auf String-Attribute in relationalen Datenbanksystemen vor. Neben der einfachen Selektion wird insbesondere auf die Bedeutung von Präfix-Bäumen beim Equi-Join eingegangen. Durch geschicktes *pruning* können Suchpfade effektiv abgeschnitten werden. Die Grundlage für Performance-Messungen bildet die Implementierung im kommerziellen RDBMS der Firma Oracle.
Die in der Arbeit geleistete Implementierung bietet an verschiedenen Stellen Optimierungspotential, das in der vorliegenden Arbeit untersucht werden soll. Es fehlen bisher ein aktives Sekundärspeicher-Puffermanagement, angepasste Knotengrößen zur Verringerung des Platzbedarfs auf dem Sekundärspeicher sowie ein angepasstes Layout des Baumes auf dem Datenträger. Ebenso sind die Oracle-spezifischen Container der Verweise auf die Datensätze nicht nativ implementiert. Insgesamt leidet die Arbeit an mehren Stellen unter den Einschränkungen des Oracle ODCII.

Zielsetzung

Ziel der Arbeit ist es, die genannten Optimierungen nachzurüsten und deren Auswirkungen auf die Performance zu untersuchen. Die Frage lautet: Welche Optimierungen bringen wie viel Performancesteigerung? Welche Vor- und Nachteile erwachsen daraus?

Herangehensweise

Die optimierte Implementierung wird zunächst außerhalb einer Datenbank, also nativ, umgesetzt. Als Baseline wird eine Implementierung erstellt, die der Implementierung aus [1] entspricht. Schritt für Schritt werden mindestens die folgenden Optimierungen umgesetzt und evaluiert:

- Native Implementierung der Container-Relation, ohne die Nutzung einer Datenbank
- Verwendung variabler statt fixer Knotenlänge [2]
- Umstellung auf echte PAT-Bäume [2,4]
- Verwendung von Kompression sowohl für die Knoten als auch für die Suffixe
- Umstellung auf ein Block-orientiertes, optimiertes Layout des Baumes auf dem Sekundärspeicher
- Nutzung von speziell angepassten Caching bzw. Prefetching

Die in [1] angegebenen Beschränkungen der Funktionalität werden beibehalten, auf die Transaktionsfähigkeit wird verzichtet. Möglicherweise wird eine Implementierung in PostGreSQL versucht.

Literatur

[1] Nicky Hochmuth, „Präfix-Bäume als Indexstruktur für String-Attribute in relationalen Datenbanken“, Diplomarbeit, Juni 2006

[2] Garcia-Molina, Ullman, Widom, “Database System Implementation”, Prentice Hall, 2000

[3] Elmasri, Navathe, “Grundlagen von Datenbanken“, Pearson Studium, 2002

[4] Szpankowski, “Patricia Tries Again Revisited”, Journal of the Association of Computing Machinery, Vol. 37, No. 4, Oktober 1990, S. 691-711

[5] Tata, Hankins, Patel, “Practical Suffix Tree Construction”, Proceedings of the 30th VLDB Conference, Toronto, Canada, 2004

[6] Grossi, Vitter, “Compressed Suffix Arrays and Suffix Trees with Applications to Text Indexing and String Matching”, Journal of the Association of Computing Machinery, 2000, S397 – 406

[7] Morrison, “PATRICIA – Practical Algorithm To Retrieve Information Coded in Alphanumeric”, Journal of the ACM, Vol. 15, No. 4, 1968, S. 514-534

[8] Galil, Rosenberg, Wood, “Storage Representations for tree-like data structures”, Journal of the ACM, 1979, S. 99-107