Self-Tuning für Bitmap-Index-Konfigurationen

Andreas Lübcke
Institut für Technische und Betriebliche Informationssysteme
Otto-von-Guericke-Universität Magdeburg

1 Einführung

Als Teilgebiet des Self-Tunings in Datenbankmanegementsysteme wird das Ziel verfolgt, die Index- oder Design Advisor [1, 6] der aktuellen DBMS so abzuändern, daß bezüglich der Indexkonfiguration ein sich selbst überwachendes und tunendes DBMS entsteht. Dabei sollen weitgehend die bisherigen statischen Ansätze der Advisor genutzt werden, diese aber in einem dynamischen und autonomen Kontext Verwendung finden. Die neu entstandenen Möglichkeiten sollen zu einer selbstständigen Anpassung an sich ändernde Rahmenbedingungen führen. Dies bezieht sich sowohl auf die Daten selbst, auf deren Nutzung sowie alle relevanten Aspekte der Systemumgebung. Self-Tuning wird in aktuellen Ansätzen durch einen Regelkreis [5] aus der Überwachung von Systemverhalten und -nutzung (Observation), der Vorhersage zukünftig gewinnbringender Systemeinstellungen (Prediction) und deren Umsetzung (Reaction) realisiert. Für das Self-Tuning von Index-Konfigurationen besteht dieser Regelkreis aus der Überwachung des aktuellen Workloads, der Ableitung geeigneter Indexkandidaten [2] und gegebenfalls der Erzeugung der vielversprechendsten Kandidaten und dem Löschen weniger profitabler Indexe [4]. Bei aktuellen Ansätzen blieben Bitmap-Indexe [3] und deren Besonderheiten bisher unberücksichtigt und werden deshalb im Rahmen des hier vorgestellten Diplomprojektes untersucht.

2 Grundlagen und Anforderungen

Bitmap-Indexe ersetzen für einzelne Schlüsselwerte Tupelidentifikator-Listen durch Bitvektoren, und sind deshalb besonders effizient, wenn nur wenige mögliche Schlüsselwerte existieren. Der erste auffällige Vorteil ist dabei der geringe Platzbedarf. Desweiteren können Bitvektoren leicht logisch miteinander verknüpft werden, um komplexe Selektionsbedingungen auszuwerten. Aus den genannten Gründen werden Bitmap-Indexe oft in Data Warehouse-Lösungen für die Indexierung von Dimensionsdaten verwendet. Ausserdem stellen Bitmap-Indexe als Verbund-Indexe eine ideale Möglichkeit zur Berechnung von STAR-Joins dar. Der Aufbau von Bitmap-Indexen bringt neben Vorteilen aber auch Nachteile, denn für Änderungsoperationen sind Matrixmodifikationen nötig, welche aufwändig sind und Sperrkonflikte wahrscheinlicher werden lassen.

Für das Self-Tuning von Bitmap-Index-Konfigurationen ergeben sich im Vergleich zu herkömmlichen B-Baum-Indexen eine Reihe von neuen Anforderungen, die hier kurz diskutiert werden. Wie gehabt basiert die Ableitung einer Index-Konfiguration auf der

Überwachung von Workloads. Jedoch ergeben sich bei der Auswertung einzelner Anfragen folgende Unterschiede.

- Für Bitmap-Indexe auf einer Relation $r(R)$ sind lediglich Prädikate aus der WHERE-Klausel relevant, welche die Form $A = const$ aufweisen, wobei die Spalte $A \in R$ eine Attributwertskardinalitätbedingung $card(dom(A))/card(r(R)) < maxSelectivity$ erfüllt, d.h. A hat eine geringe Anzahl möglicher Werte im Vergleich zur Größe der Relation.
- Für Bitmap-Join-Indexe muss ein (innerer) Equi-Join $r(R) \bowtie_{A_1=A_2} r(S)$ mit $A_1 \in R$ und $A_2 \in S$ (Verbundattribute) erkannt werden, wobei eine weitere Spalte $A_3 \in S$ (Indexschlüssel) in der WHERE-Klausel existiert, für die die oben genannten Bedingungen zutreffen. Für die Erkennung der Join-Bedingung können in einem Data Warehouse gegebenfalls Dimensionsdefinitionen herangezogen werden.
- Bitmap-Indexe sind besonders sinnvoll, wenn davon mehrere in einer Anfrage verwendet und ihre Bit-Felder logisch kombiniert werden. Dies muss im Entscheidungsmodell entsprechen berücksichtigt werden.
- Mehrspalten-Bitmap-Indexe sind möglich, aber durch die schlechtere Attributwertskardinalität und eingeschränkte Kombinierbarkeit meist nicht empfohlen, wodurch sich eine Einschränkung des Suchraums ergibt.
- Bitmap-Indexe haben auf Grund Ihrer geringeren Größe meist ein besseren Verhältnis von Nutzen zu Speicherverbrauch, weshalb die Integration mit dem Tuning anderer Indexstrukturen berücksichtig werden muss.
- Bitmap-Indexe haben vergleichsweise geringe Erzeugungskosten, jedoch höhere Kosten bei Updates. Dies muss in das Kostenmodell einfliessen.

3 Ansätze zum autonomen Bitmap Index Tuning

Das prinzipielle Vorgehen beim Self-Tuning von Bitmap-Index-Konfigurationen ist in Abb. 1 dargestellt. Generell werden die Indexkandidaten durch eine syntaktische Analyse von Anfragen entsprechend den oben genannten Anforderungen ermittelt. Daraufhin werden diese Kandidaten in den Systemtabellen erzeugt und für die eine What-If-Analyse („was wäre, wenn diese Index existieren würden“) dem Optimierer zur Verfügung gestellt.

Der aus der What-If-Analyse berechnete Gewinn von Indexkandidaten und bereits materialisierten Indexen gegenüber der Ausführung der Anfrage ohne Indexe wird für spätere Analyse und Empfehlungen als Indextstatistiken gespeichert. Aufgrund der Kombinierbarkeit der Bitmap-Indexe stellt sich die Frage, ob diese jeweils einzeln gespeichert werden sollten, oder diese wiederum direkt in die Bewertung der Konfiguration eingehen. Auf dieser Basis muß ein angepaßtes Entscheidungsmodell gefunden werden, daß die Besonderheiten der Bitmap-Indexe berücksichtigt. Weiterhin muß anhand der Größe des Indexpools und weiteren Parametern entschieden werden, welche Konfiguration bestmöglich alle Parameter erfüllen kann. Dies kann durch einen Greedy-Algorithmus oder

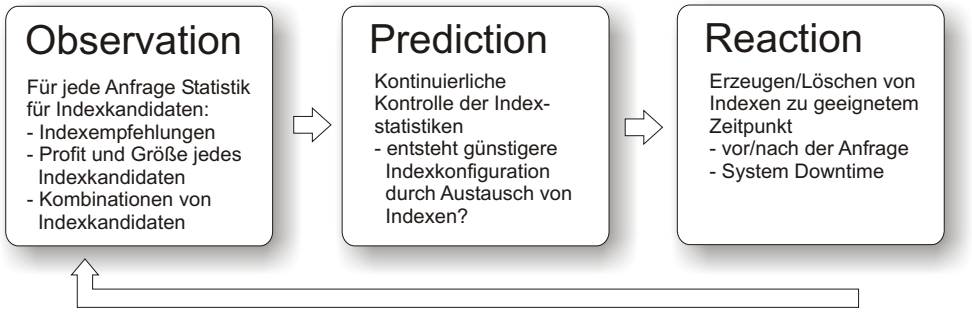

Abbildung 1: Regelkreis Index-Self-Tuning

Dynamic Programming geschehen. Abschließend muß die Frage des Erstellungszeitpunktes geklärt werden, denn werden die Bitmap-Indexe vor der Ausführung der Anfrage erzeugt, dann führt dies zu einer verzögerten Ausführung. Hier muß betrachtet werden, wie stark die Erzeugung der Konfigurationen die Anfragezeiten beeinflußt. Im Gegensatz dazu können die Index-Konfigurationen in der System Downtime oder Phasen geringer Auslastung erzeugt werden, wodurch der Gewinn durch die neue Index-Konfiguration für die aktuelle Anfrage verloren geht. Für das Diplomvorhaben wird das DBMS Oracle 10g genutzt. Dessen Design Advisor bietet bereits eine Unterstützung für Bitmap-Indexe an, jedoch werden keine Verbund-Indexe unterstützt und Ziel der hier beschriebenen Arbeit ist die dynamische Unterstützung des Self-Tuning von Bitmap-Indexen. Im Rahmen dieses Diplomvorhabens wird das Self-Tuning der Bitmap-Indexe unabhängig von anderen Indexstrukturen in einem separaten Index-Pool behandelt.

Literatur

[1] S. Agrawal, S. Chaudhuri, L. Kollár, A. P. Marathe, V. R. Narasayya, and M. Syamala. Database Tuning Advisor for Microsoft SQL Server 2005. In *VLDB '04*, pages 1110–1121, 2004.

[2] N. Bruno and S. Chaudhuri. To Tune or not to Tune? A Lightweight Physical Design Alerter. In *VLDB '06*, pages 499–510, 2006.

[3] C.-Y. Chan and Y. E. Ioannidis. Bitmap index design and evaluation. In *SIGMOD '98: Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, pages 355–366, New York, NY, USA, 1998. ACM Press.

[4] K. Sattler, E. Schallehn, and I. Geist. Autonomous Query-driven Index Tuning. In *Proc. Int. Database Engineering and Applications Symposium (IDEAS 2004), Coimbra, Portugal*, pages 439–448, July 2004.

[5] G. Weikum, C. Hasse, A. Moenkeberg, and P. Zabback. The COMFORT Automatic Tuning Project, Invited Project Review. *Information Systems*, 19(5):381–432, 1994.

[6] D. C. Zilio, J. Rao, S. Lightstone, G. M. Lohman, A. Storm, C. Garcia-Arellano, and S. Fadden. DB2 Design Advisor: Integrated Automatic Physical Database Design. In *VLDB '04*, pages 1087–1097, 2004.