Indexallokation in Parallelen Datenbanksystemen

Th. Stöhr

Institut für Informatik, Universität Leipzig

stoehr@informatik.uni-leipzig.de
http://www.informatik.uni-leipzig.de/~stoehr/

Kurzfassung

Die effiziente Nutzung von Zugriffsstrukturen ist eine wichtige Voraussetzung für die performante Durchführung von Datenbankanfragen. Die in Parallelen Datenbanksystemen vom Typ Shared-Nothing übliche, durch die Allokationsstrategie für Relationen weitgehend vorgegebene Indexallokation führt oftmals zu unnötigen I/O-, Verarbeitungs- und Kommunikationskosten. Parallele Shared-Disk Datenbanksysteme bieten durch ihren gemeinsamen Plattenzugriff ein hohes Potential zur flexiblen Allokation von Indexstrukturen. Wir präsentieren eine Klassifikation und eine qualitative Bewertung von Indexallokations-Strategien für diese Architekturklasse, die zeigt, daß sich durch die flexible Wahl von Größen wie Verteilattribut und Verteilgrad die Performanz der parallelen Indexverarbeitung steigern läßt.

1 Einleitung

Zur Bewältigung stetig wachsender Datenmengen und sehr hoher Leistungsanforderungen komplexer DB-Anwendungen haben sich Parallele Datenbanksysteme (PDBS) [DG92] als eine Schlüsseltechnologie etabliert. So können in Anwendungsfeldern wie Data Warehousing meist nur durch Einsatz paralleler Bearbeitungsstrategien ausreichend schnelle Anfrage-Antwortzeiten erzielt werden. Zur Reduzierung der üblicherweise sehr hohen I/O-bezogenen Verarbeitungszeiten ist neben der Nutzung unterschiedlicher Parallelitätsarten der effektive Einsatz von Indexstrukturen von grundlegender Bedeutung, da sie eine drastische Reduzierung der zu verarbeitenden Datenmengen unterstützen. Von den zahlreichen vorgeschlagenen Zugriffspfadstrukturen und Indextechniken ist in der Praxis der Einsatz von B*-Bäumen aufgrund ihrer vielfältigen Nutzungsmöglichkeiten besonders bedeutsam [HR99]. Erst in jüngerer Vergangenheit sind in kommerziellen DBS verstärkt weitere Indexorganisationen eingeführt worden, insbesondere zur besseren Unterstützung von Decision Support und Data Warehousing (v.a. Bit-Indizes [OQ97]) sowie objektrelationaler Erweiterungen für Geo-Anwendungen, Information Retrieval etc.

Trotz der großen und weiter steigenden Bedeutung der Indexstrukturen wurde ihr Einsatz innerhalb von Parallelen DBS bisher kaum näher untersucht. Bezüglich des zentralen Problems der Datenallokation wurde meist nur die Rechner- bzw. Plattenzuordnung der Tabellen betrachtet, welche üblicherweise horizontal über eine Hash-Funktion oder Bereichszerlegung auf einem Verteilattribut partitioniert werden [Gh90, DG92, MD97]. Dies ist auch vor allem auf die Konzentrierung der meisten Untersuchungen auf Shared-Nothing-(SN-)DBS zurückzuführen, da hier die Allokation der Nutzdaten (Tabellen) auch die Indexallokation bestimmt, um einem Rechner die lokale Indexnutzung auf seine Datenpartition zu ermöglichen.

Wesentlich größere Freiheitsgrade zur Datenallokation und insbesondere auch zur Indexallokation bestehen dagegen für PDBS-Architekturen, in denen ein gemeinsamer Plattenzugriff für die Rechner bzw. Prozessoren besteht, also für Shared-Disk (SD), Shared-Everything (SE) und bestimmte hybride Architekturen (z.B. SD-DBS mit SE-Knoten bzw. SD-/SE-Knoten innerhalb eines hybriden SN-Systems). Insbesondere kann die Speicherung von Zugriffsstrukturen weitgehend unabhängig von derjenigen der

Relationen erfolgen, wobei neben einer Verwendung *zentraler* Indizes unterschiedliche Varianten *verteilter* (*partitionierter*) Indexstrukturen zur Verbesserung der I/O-Leistung eingesetzt werden können. Wesentlich für die Güte einer Allokationsform ist ihre effektive Nutzbarkeit hinsichtlich einzelner Anfragetypen, wobei der weitgehenden Eingrenzung des Suchraumes und damit der zu bearbeitenden Datenmenge eine große Bedeutung zukommt. Weiterhin sollten die Indexoperationen möglichst gut und flexibel parallelisiert werden können, wobei auch auf möglichst geringe Behinderungen beim Zugriff auf die gemeinsamen Platten zu achten ist. Die Hersteller von PDBS bieten i.d.R. unterschiedliche Allokationsmöglichkeiten für Indexstrukturen (s. Abschn. 2.4), jedoch finden sich in den Handbüchern meist nur vage Angaben, in welchen Fällen welche Lösung Vorteile bringen kann.
In diesem Bericht untersuchen näher, welche generellen Alternativen zur Indexallokation insbesondere für Shared-Disk bestehen und wie diese qualitativ zu bewerten sind. In Kapitel 2 stellen wir dazu nach einer kurzen Definition der wichtigsten Begriffe eine Klassifikation der Indexpartitionierungstrategien für Parallele SD-Systeme vor und vergleichen die Situation mit derjenigen in SN-Umgebungen. Kapitel 3 faßt zusammen und gibt einen Ausblick auf notwendige weitere Arbeiten.

2 Klassifikation der Indexallokation in Shared-Disk-Systemen

Die Bestimmung einer Indexallokation umfaßt analoge Teilaufgaben wie bei der Bestimmung einer Datenverteilung für Relationen [Ra94], nämlich die Bestimmung der Verteilgranulate (Fragmentierung), die Festlegung des Verteilgrades (Anzahl von Partitionen) sowie die Zuordnung der Partitionen zu den Platten bzw. (bei SN-)Rechnerknoten. Unsere Überlegungen konzentrieren sich weitgehend auf die beiden ersten Aspekte, während wir für die Plattenauswahl zur Aufnahme von Partitionen Standardverfahren unterstellen (z.B. Round-Robin oder Balancierung von Zugriffshäufigkeiten [SWZ98]). Bei Shared-Nothing ist die Rechnerzuordnung für Indexpartitionen durch die Datenverteilung der Relationen bestimmt.

2.1 Arten der Verteilung

Für Shared-Disk lassen sich sowohl für Relationen als auch Indexstrukturen eine physische oder logische Partitionierung (Fragmentierung) anwenden. *Physische Datenpartitionierung* verwendet physische Datengranulate wie Blöcke oder Blockmengen als Verteilungseinheiten (Fragmente), ohne Berücksichtigung von Attributwerten. Diese Methode kann außerhalb des DBS im Dateisystem bzw. innerhalb eines Disk-Arrays [Ch94] realisiert werden. Alternativ dazu ist eine Steuerung durch das DBS (Festlegung im DB-Schema von Anzahl und Allokationsort von Dateien bzw. sog. Raw Devices, welche Relationen- oder Indexfragmente aufnehmen sollen). Letztere Variante gestattet dem DBS gezieltere Parallelisierungsmöglichkeiten, insbesondere um Plattenbehinderungen zwischen Teilanfragen zu reduzieren [St97]. In [RS95] wurde gezeigt, daß eine physische Partitionierung auch im Mehrbenutzerbetrieb vielfach eine effektive Parallelisierung von Relationen-Scans und Index-Scans in Shared-Disk-DBS ermöglicht. Bei *logischer Verteilung* werden Relationen bzw. Indizes gemäß den Werten eines *Verteilattributs* (bzw. mehrerer Verteilattribute) in Fragmente partitioniert und Platten zugeordnet. Üblicherweise wird die Verteilung über eine Hash- oder Bereichfunktion definiert. Durch die logische Partitionierung kann der Suchraum einer relationalen Anfrage auf dem Verteilattribut auf einen Teil der Relation bzw. des Index beschränkt werden; damit können Suchanfragen auf dem *Verteilattribut der Indexstruktur* auf eine Teilmenge der Teil-Indizes beschränkt werden. Das Verteilattribut einer Indexstruktur muß nicht notwendigerweise mit dem Indexattribut übereinstimmen.
Alle SN-DBS verwenden eine logische Partitionierung von Relationen und zugeordneten Indexstrukturen; ein physisches Declustering wird höchstens innerhalb eines Knoten mit mehreren vorhandenen Platten durchgeführt. Da Partitionen in SN nur rechner-lokal zugreifbar sind, sind Indexpartitionen mit Datenpartitionen direkt assoziiert, d.h. sie werden rechner-lokal aufgebaut und allokiert. Verteilgrad und Verteilattribut aller Indexstrukturen einer Relation entsprechen damit den entsprechenden Größen der Relation.
In SD-Systemen ist ebenfalls eine logische Partitionierung für Relationen und Indexstrukturen möglich. Die gemeinsame Plattenanbindung erlaubt aber eine unabhängige Betrachtung von Verteilgrad und Verteilattribut für Indexstrukturen und Relationen. Insbesondere kann das Indexattribut als Verteilattribut einer Indexstruktur verwendet werden, auch wenn die Relation über ein anderes Verteilattribut (oder

physisch) partitioniert wird. Dadurch ist eine bessere Anpassung der Speicherung von Nutz- und zugriffsunterstützenden Daten an die Anforderungen komplexer und datenintensiver Lasten möglich.

2.2 Klassifikation

Die entscheidenden Größen zur Klassifikation von Partitionierungsstrategien für Indizes sind das Verteilattribut und der Verteilgrad. Das Verteilattribut für den Index bestimmt letztlich, ob eine Anfrage auf dem Index auf eine Teilsuche eingeschränkt werden kann. Der Verteilgrad bestimmt die Effizienz des durchzuführenden I/O-Umfangs für das Lesen einer Indexstruktur. Beide Aspekte haben Einfluß auf die Parallelisierung von Indexzugriffen. Unsere Klassifikation ist also 2-stufig angelegt: auf der ersten Stufe der Partitionierung betrachten wir zunächst *logische* Aspekte, die die Größe des Suchraums für eine Indexdurchlauf durch die *Wahl des Verteilattributes* beeinflußen. Eine Rolle spielen dabei das Indexattribut (IA) sowie das Verteilattribut der Relation VA_R. Auf der zweiten Stufe betrachten wir *physische* Aspekte, die sich dem *Grad des Declustering* und der daraus resultierenden I/O-Parallelität befassen. Physisch zentral vs. declustered bezieht sich dabei jeweils auf ein logisches Fragment.
Abb. 1 zeigt unsere Klassifikation der Indexallokation in SD-DBS. In einigen Allokationsformen gehen wir davon aus, daß die zugrundeliegende Relation nach einem Verteilattribut VA_R partitioniert wurde.

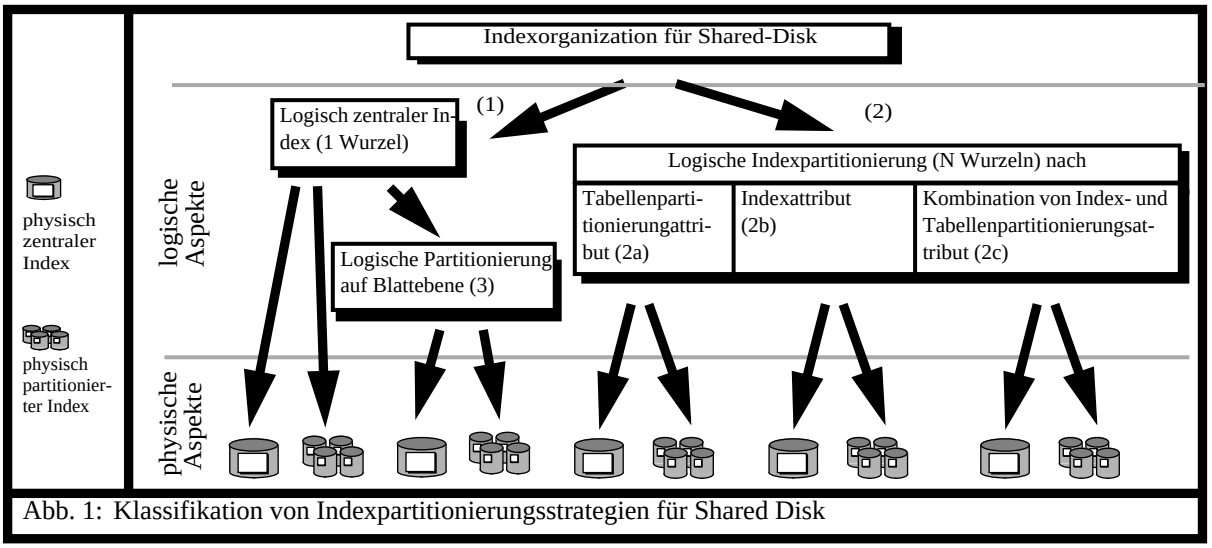

Abb. 1: Klassifikation von Indexpartitionierungsstrategien für Shared Disk

2.2.1 Logisch zentraler Index

Ein logisch zentraler oder *globaler Index* entspricht der für zentrale DBS üblichen Organisationsform eines Mehrwegbaumes pro Indexattribut mit einer Wurzel (1). Dies erlaubt den minimalen Speicherbedarf, während logisch verteilte Indexstrukturen eine größere Redundanz in den Zwischenknoten verursachen. Die Höhe eines globalen Index ist i.d.R. nur wenig höher als für einen Teilindex eines logisch verteilten Index, so daß für einen Indexzugriff allenfalls geringfügig mehr Suchaufwand anfällt.
Im einfachsten Fall wird der logisch zentrale Index auch physisch zentral auf einer Platte gespeichert (falls die Indexgröße die Plattenkapazität nicht übersteigt). Diese Indexorganisation ist leicht administrierbar, da weder Verteilattribut noch Verteilgrad festzulegen sind. Andererseits besteht in diesem Falle das geringste Parallelisierungspotential, so daß sich diese Variante lediglich für kleinere Indexstrukturen und hochselektive Anfragetypen mit wenigen Treffern (z.B. exact-match-Anfragen) eignet. Doch selbst in diesen Fällen besteht im Mehrbenutzerbetrieb für einen häufig refrenzierten Index eine erhöhte Gefahr an Plattenengpäßen, falls nicht Großteile des Index im Hauptspeicher gepuffert werden können.
Diesen Problemen kann mit einem physischen Declustering von Indexseiten begegnet werden, bei dem diese über mehrere Platten verteilt werden, z.B. nach einem Round-Robin-artigen Allokationsschema. Damit lassen sich insbesondere Engpässe zwischen unabhängigen Indexzugriffen reduzieren. Eine Alternative wäre die Kombination eines globalen Index mit einer logischen Partitionierung auf der Blattebene (3), ähnlich wie in [KFK96] vorgeschlagen. Damit könnten Index-Scans gezielt auf ein(ig)e Partition(en) beschränkt werden und Sequentialität bei Plattenzugriffen auf der Blattebene genutzt werden. Bei zusätzlichem physischem Declustering der logischen Partitionen kann zudem I/O-Parallelität zwischen unabhängigen Indexoperationen genutzt werden.

2.2.2 Logisch verteilter Index

Logisch verteilte Indizes bestehen aus mehreren Teilindizes mit je einer eigenen Wurzel, inneren Knoten und Blattknoten (2). Durch geschickte Wahl des Verteilattributes kann man einerseits Parallelität in der Indexsuche nutzen, andererseits Durchläufe nicht benötigter Indexbereiche einsparen. In SN-Systemen ist die Wahl des Relationen-Verteilattributs VA_R als Index-Verteilattribut unumgänglich (2a). Dies reduziert zwar die Intra-Anfrage-Konkurrenz auf Tabellenpartitionen, erzwingt aber maximal parallele Indexsuche. Wählt man IA als Verteilattribut (2b), so kann die Suche im B-Baum meist auf wenige Sub-Indizes eingeschränkt werden. Dies ist optimal für hochselektive Anfragen, erzeugt aber für geringer selektive Anfragen bzw. Anfragen auf Sekundärindizes entsprechende Intra-Anfrage-Konkurrenz auf den assoziierten Datenpartitionen, da ein logischer Subindex dadurch ggf. viele Tupel-Identifier (TID) verwalten muß. Eine dritte Möglichkeit (2c) ist daher die (mehrdimensionale) Indexpartitionierung nach einer Kombination von IA und VA_R. Dabei verwalten Subindizes die TID, welche sowohl einen Bereich des IA als auch des VA_R abdecken. Diese Variante ermöglicht einerseits die Einschränkung der Indexsuche, andererseits die Nutzung moderater Parallelität.
Generell empfiehlt sich, die logischen Indexpartitionen auf mehrere Platten zu speichern, um I/O-Parallelität zu nutzen. Die Unterscheidung bezüglich der physischen Allokation bezieht sich auf je eine Partition, die entweder auf einer Platte oder selbst wieder physisch verteilt über mehrere Platten zugeordnet werden können (-> 2-stufige logische + physische Verteilung).

2.3 Wahl des Verteilgrades und des Allokationsortes

Am häufigsten werden Relation in SN nach einem Verteilattribut logisch auf alle Rechner bereichs-partitioniert (’*full declustering*’), um teure Relation-Scans maximal parallel bearbeiten zu können. Sollen verschieden I/O-intensive Lasttypen gleichzeitig unterstützt werden (Relationen-Scans, Index-unterstützte exact-match-Anfragen, ..), so muß bei Allokation der Relationen ein Kompromiß im Verteilgrad gefunden werden [GDQ92]. Ebendies gilt auch für die Indexstrukturen, die rechner-lokal auf den dort allokierten Partitionen generiert und für deren Index-Verteilgrad VG_I im Bezug auf den Relationenverteilgrad VG_R gelten muß: VG_I=VG_R. Bei verteiltem Index können im SD-Fall VG_I und VG_R im Verhältnis zueinander so gewählt werden, daß unter Berücksichtigung des I/O-Aufkommens für Daten und Indexzugriffe eine möglichst gleichmäßige Balancierung der Plattenlast schon durch die Allokation gewährleistet werden kann. Zur Nutzung von I/O-Parallelität können die einzelnen Sub-Indizes disjunkt voneinander und zusätzlich noch physisch partitioniert werden. Dies wäre im SN-Fall lediglich rechner-lokal (bei mehreren Platten pro Knoten) möglich.

2.4 Unterstützung in kommerziellen Parallelen DBS

Bei DBS vom Typ Shared-Nothing (z.B. Informix XPS, DB2 Parallel Edition, Teradata, Tandem) findet man naturgemäß keine Unterschiede zwischen der Allokationstrategie für Index bzw. Relationenstrukturen. Beide werden unter den einzelnen Knoten via logischer Verfahren (Round-Robin, Bereichs- und Hash-Fragmentierung, Ausdrücke auf Schlüsselattributen) verteilt.
SD-Systeme unterstützen schon längere Zeit physisches Declustering der Tabellen und Indizes. In *Oracle8 Parallel Server [Or97]* werden jetzt auch logische Verteilungsstrategien sowohl für Daten als auch B*-Bäume unterstützt. Oracle8 bietet diverse Partitionierungsstrategien für Indexstrukturen an. Die Hilfestellung beschränkt sich allerdings im Wesentlichen darauf, Indizes für (hochselektive) OLTP-Lasten nach dem Indexattribut zu partitionieren, Indizes für DSS-Anfragen nach dem Tabellenpartitionierungsattribut zu verteilen. Es fehlen Hinweise über die Wahl der Verteilgrade VG_I und VG_R, über die Wahl des Allokationsortes von Indexfragmenten in Relation zu den assoziierten Datenfragmenten oder eine Entscheidungshilfe über die Partitionierungsstrategie in Abhängigkeit von der (erwarteten) Selektivität von Anfragen.

3 Zusammenfassung und Ausblick

Neben der Allokation umfangreicher relationaler Datenmengen wurde der Speicherung von Zugriffspfaden bis dato relativ wenig Beachtung geschenkt. Bei der bisher auf dem Parallelen DB-Sektor dominierenden SN-Architektur ist die Alloaktion dieser Strukturen durch die Datenpartitionierung weitgehend vorgegeben.

Parallele Systeme vom Typ Shared-Disk dagegen bieten nicht nur viele Freiheitsgrade in der Allokation und Verarbeitung der Nutzdaten, sondern gerade auch für die entsprechenden Zugriffsstrukturen. Durch den gemeinsamen Plattenzugriff können entscheidende Allokationsgrößen wie Verteilattribut und Verteilgrad für Indizes weitgehend unabhängig von den entsprechenden Größen für die Nutzdaten (Relationen, Objekte) gewählt werden. Wir zeigten anhand einer Klassifikation von Indexpartitionierungs-Strategien die Möglichkeiten dieser Architekturklasse auf: die Verteilung von Indexstrukturen nach dem Indexattribut kann für selektive Anfragen die Menge zu durchsuchender Indexpartitionen drastisch reduzieren. I/O-Anforderungen für Index- und Nutzdaten können gleichmäßiger ausbalanciert werden, in dem der Verteilgrad bzw. Allokationsort für Daten- und Indexpartitionen nicht durch Rechneranzahl oder Rechnergrenzen, sondern durch das tatsächliche Lastaufkommen und dessen Anforderungen bestimmt werden. Die vorgestellten Überlegungen bedürfen allerdings einer ausführlichen quantitativen Analyse, die wir sowohl analytisch als auch simulativ (mit Hilfe unseres selbst entwickelten Simulationssystems für Parallele DBS, *SimPaD*) durchführen werden.
Diese Flexibilität bedeutet im praktischen DBA-Betrieb allerdings eine hohe Komplexität im Auffinden der 'richtigen' Allokationsstrategie. Deswegen wollen wir unsere gewonnenen Erkenntnisse und die daraus resultierenden Algorithmen in ein Allokationswerkzeug für DBAs integrieren. Das Werkzeug soll bei der Festlegung des physischen DB-Entwurfs für große Datenmengen unter Berücksichtigung der zu erwartenden Laststruktur eine Allokationsstrategie vorschlagen. Dies ist von enormer Bedeutung z.B. bei der Festlegung der Allokation von Data Marts innerhalb eines Data Warehouses, die, trotzdem sie meist einen abgegrenzten Geschäftsbereich repräsentieren, schon erhebliche Datenmengen enthalten können (bis mehrere GB) und ein umfangreiches Anfragevolumen komplexer Anfragen verarbeiten müssen. Hier muß das zugrundeliegende Star-Schema mit sehr umfangreicher Faktentabelle und assoziierten Dimensionstabellen nebst entsprechend großen Indexstrukturen auf die zu erwartenden Analyseanfragen hin optimal allokiert werden. Auch wurden Methoden zur effizienten Speicherung von Bit-Indizes, die im Data Warehouse-Bereich eine große Rolle spielen, noch nicht untersucht.

4 Literatur

[Ch94] Chen, P.M., Lee, E.L., Gibson, G.A., Katz, R.H., Patterson, D.A.: *RAID: High-Performance, Reliable Secondary Storage*. Computing Surveys 26 (2): S. 145–185, 1994.

[DG92] DeWitt, D.J., Gray, J.: *Parallel Database Systems: The Future of High Performance Database Systems.* Comm. ACM 35 (6), 85–98, 1992

[KFK96] Koudas, Nikos; Faloutsos, C.; Kamel, I.: *Declustering Spatioal Databases on a Multi-Computer Architecture*. Proc. 5. EDBT '96, Avignon, Springer-Verlag, LNCS 1057, S. 592–614, 1996

[MD97] Mehta, M.; DeWitt D.J.: *Data Placement in Shared-Nothing Parallel Database Systems*. VLDB Journal 6 (1), S. 53–72, 1997

[GDQ92] Ghandeharizadeh, S., DeWitt, D.J., Qureshi, W.: *A Performance Analysis of Alternative Multi-Attribute Declustering Strategies*. Proc. ACM SIGMOD Conf., S. 29–38, 1992.

[Gh90] Ghandeharizadeh, S., DeWitt, D.J.: *A Multiuser Performance Analysis of Alternative Declustering Strategies*. Proc. 6. IEEE Intl. Conf. on Data Engineering (ICDE), 1990

[HR99] Härder, Th.; Rahm, E.: *Datenbanksysteme - Konzepte und Techniken der Implementierung.* Springer-Verlag, 1999

[Ra94] Rahm, E.: *Mehrrechner-Datenbanksysteme. Grundlagen der verteilten und parallelen Datenbankverarbeitung*. Addison-Wesley, 1994

[RS95] Rahm, E., Stöhr, Th.: *Analysis of Parallel Scan Processing in Shared Disk Database Systems*. Proc. EURO-PAR '95, Stockholm, Springer-Verlag, LNCS 966, S. 485–500, 1995

[St97] Stöhr, Th.: *Ein Simulationsansatz zur Bewertung paralleler Shared-Disk-Datenbanksysteme.* Proc. 9. ITG/GI-Fachtagung MMB '97, S. 82–89, VDE-Verlag, Freiberg, September 1997.

[SWZ98] Scheuermann, P.; Weikum, G.; Zabback, P.: *Data Partitioning and Load Balancing in Parallel Disk Systems*. VLDB Journal 7 (1), S. 48–66, 1998

[OQ97] O'Neil; P.; Quass, D.: *Improved Query Performance with Variant Indexes*. Proc. ACM SIGMOD Conf., Tucson, Arizona, May 1997.

[Or97] *Taking Advantage of Partitioning in Oracle8*. Oracle Technical White Paper, Oracle Corp., http://www.oracle.com/st/collateral/html/partitioning_collateral.html, 1997