

Datenbankreorganisation

bei relationalen Datenbank-Management-Systemen

Stefan Dorendorf, Klaus Küspert
Friedrich-Schiller-Universität Jena
Institut für Informatik
Lehrstuhl für Datenbanken und Informationssysteme
Ernst-Abbe-Platz 1-4
07743 Jena
stefan@ba-glauchau.de
kuespert@informatik.uni-jena.de

Kurzfassung

Relationale Datenbank-Management-Systeme (DBMSe) zeichnen sich u.a. dadurch aus, daß der Benutzer wenig von Systeminterna, vor allem hinsichtlich der Art und Details der physischen Datenablage, wissen muß und dadurch ein hoher Grad an Datenunabhängigkeit erreicht wird. Trotzdem spielt die physische Datenablage auch hier weiter eine wichtige Rolle, dies insbesondere für den Datenbankadministrator: Performance-Aspekte im weitesten Sinne (Systemantwortzeiten, Speicherplatzausnutzung etc.) sind ein Grund, die physische Datenablage zu überwachen und zu beeinflussen, ebenso bspw. das Problem des „Anstoßens“ an Systemgrenzen (Speicher voll u.dgl.). Die Beeinflussung der physischen Datenablage erfordert dabei in vielen Fällen das periodische oder situationsabhängige Durchführen von **Datenbankreorganisationen**. In diesem Beitrag werden der Begriff und verschiedene Facetten der Datenbankreorganisation näher vorgestellt, es werden Gründe und Ziele der Datenbankreorganisation erläutert, und es wird auf verschiedene Ansätze und Methoden dafür im Rahmen einer Klassifikation eingegangen. Zur Veranschaulichung steht die konkrete Betrachtung von Datenbankreorganisationsmöglichkeiten im DBMS Informix mit im Mittelpunkt der Erörterungen. Nicht zuletzt wird auf kritische Punkte in Zusammenhang mit der Vorbereitung und Durchführung von Datenbankreorganisationen hingewiesen sowie auf aktuelle Forschungsinhalte in diesem Fachgebiet, die teils auch ihren Grund in neuen oder sich verschärfenden Anforderungen bzgl. des Datenbanksystemeinsatzes haben (*sehr* große Datenbanken, *sehr* hohe Verfügbarkeit).

09.11.98

Inhaltsverzeichnis

1 Einleitung und Motivation	4
2 Reorganisation von Datenbanken.....	6
2.1 Gründe für eine Datenbankreorganisation.....	6
2.2 Begriff Datenbankreorganisation.....	7
2.3 Ziele einer Datenbankreorganisation.....	7
3 Probleme, Einflußgrößen und Anforderungen.....	8
3.1 Probleme bei einer Datenbankreorganisation.....	8
3.2 Faktoren, die eine Datenbankreorganisation beeinflussen	9
3.3 Anforderungen an eine Datenbankreorganisation.....	10
3.4 Granulate für eine Datenbankreorganisation.....	10
4 Methoden zur Datenbankreorganisation	11
4.1 Methoden zur explizit veranlaßten Datenbankreorganisation.....	12
4.1.1 Datenbanksystembasierte Reorganisation.....	12
4.1.2 Datenbanksystemintegrierte Reorganisation	13
4.2 Implizit veranlaßte Datenbankreorganisation.....	13
5 Datenbankreorganisation auf verschiedenen Ebenen.....	14
5.1 Reorganisationsebenen	14
5.2 Sekundärspeicherorganisation am Beispiel von Informix	14
5.3 Datenbankreorganisation auf der Verteilungsebene.....	16
5.3.1 Verteilen der Schemaelemente	16
5.3.2 Verlagern der Log-Daten bzw. der Metadaten	16
5.3.3 Partitionierung von Relationen/Indexen.....	16
5.3.4 Physisch zusammenliegende Speicherung von Tupeln mehrerer Relationen	17
5.4 Datenbankreorganisation auf der internen Ebene.....	18
5.4.1 Beseitigung von Fragmentierungen	18
5.4.2 Freigabe von Speicherplatz	19
5.4.3 Verringerung der Anzahl von Extents einer Relation	20
5.4.4 Beseitigung von Auslagerungszeigern	20
5.4.5 Wiederherstellen vorgegebener Sortierreihenfolgen.....	21
5.4.6 Löschen von "als gelöscht" gekennzeichneten Indexeinträgen	21

5.4.7 Freigabe von Indexseiten	22
5.4.8 Steuerung des Füllungsgrades von Indexseiten.....	22
6 Zusammenfassung und Ausblick	23

1 Einleitung und Motivation

Relationale Datenbank-Management-Systeme (DBMSe) haben seit Mitte der 80er Jahre einen bedeutenden Siegeszug in der Praxis angetreten. Die heute oftmals sogenannten „vorrelationalen“ Systeme nach dem hierarchischen oder dem Netzwerk-Datenbankmodell, die noch in den 70er und frühen 80er Jahren eine wesentliche Rolle hinsichtlich ihrer Marktbedeutung spielten, wurden teils durch relationale Systeme abgelöst; vor allem konnten relationale Systeme aber erfolgreich auch solche Anwendungsgebiete für sich erschließen, wo zuvor noch gar nicht mit Datenbanktechnologie gearbeitet wurde, sondern bspw. dateibasierte Lösungen vorherrschend waren bzw. die Informationstechnologie erst dabei war, in großem Stil Einzug zu halten.

Ein mitentscheidender Vorteil der relationalen gegenüber der vorrelationalen Datenbanktechnologie liegt im erzielbaren Grad an **Datenunabhängigkeit**. Bei den vorrelationalen DBMS-Produkten fand und findet keine konsequente Trennung von physischen Aspekten auf der einen Seite (wie die Daten genau abgelegt sind, wie sie verzeigert sind, wie die Sortierordnung aussieht usw.) und logischen Aspekten auf der anderen Seite (wie die Modelldarstellung der Daten aussieht, wie die Anwender die Daten sehen wollen etc.) statt, d.h. Anwendungen sind stets eng mit der gewählten oder vom System vorgegebenen physischen Datenbankstruktur „verheiratet“. Wenn sich diese physische Datenbankstruktur ändert, etwa Datensätze bzgl. der Reihenfolge ihrer Ablage auf dem Speichermedium umsortiert werden (um nur ein Beispiel zu nennen), sind die Datenbankanwendungen unmittelbar betroffen, d.h. es sind dann Anpassungen (kostspielige Programmänderungen) erforderlich, um den Benutzeranforderungen weiterhin funktional zu genügen. Bei der relationalen Datenbanktechnologie sieht der Benutzer seine Daten dagegen stets in einfacher Tabellenform auf vergleichsweise hohem Abstraktionsniveau und völlig frei von physischen Aspekten der Datenablage und -darstellung. Dem relationalen Datenbankmodell zufolge sind sowohl die Tabellenspalten als auch (vor allem) die Tabellenzeilen ungeordnet und besitzen die Mengeneigenschaft. Ob eine Sortierordnung der Tabellenzeilen intern vorliegt und, wenn dies der Fall ist, ggf. nach welchem Sortierkriterium, bleibt der Datenbankanwendung gegenüber hier *funktional* verborgen. Ebenso merkt die relationale Datenbankanwendung funktional nichts vom Vorhandensein oder Nichtvorhandensein von Zugriffshilfen (Indexen), von einer örtlich verteilten oder nichtverteilter Datenablage oder von zwischen die Daten eingestreutem oder nichteingestreutem Freiplatz, um einige weitere Beispiele zu nennen. Erst recht bleiben Speichercharakteristika hinsichtlich der genauen Zuordnung - Magnetplatte x eines Typs mit Eigenschaft y enthält Daten der Tabelle z - gegenüber der Datenbankanwendung funktional verborgen.

Die hier vorgenommene deutliche Betonung des *funktionalen* Verbergens in Zusammenhang mit dem Erzielen von Datenunabhängigkeit bei relationalen Datenbank-Management-Systemen hat ihren Grund: Funktionales Verbergen von Eigenschaften auf physischer Ebene gegenüber dem Benutzer und seinen Datenbankanwendungen heißt nicht, daß dem Benutzer / der Datenbankanwendung die Aspekte der physischen Datenablage somit völlig egal sein können: Unter **Performance-Gesichtspunkten** (Zugriffszeit) kann es durchaus von Bedeutung sein, ob Daten auf der physischen Ebene sortiert abgelegt sind oder nicht, und ebenso kann unter diesem Blickwinkel das Vorhandensein oder Nichtvorhandensein bestimmter Indexe sehr wesentlichen Einfluß auf das Antwortzeitverhalten des DBMS be-

sitzen. Eingestreuter Freiplatz kann ebenso Performance-Auswirkungen besitzen, weil er hinsichtlich der effizienten Durchführung von Datenbankabfragen i.d.R. stört (wogegen für die Durchführung von Datenbankänderungen solcher Freiplatz meist hilfreich ist). Außerdem führt zu viel Freiplatz auf physischer Ebene natürlich zur Speicherplatzverschwendung bzw. es kann bei intern stark fragmentierten („zerstückelten“) Daten der Fall auftreten, daß Systemgrenzen bzgl. der Zahl der zulässigen Fragmente (Stücke) erreicht werden und ein Systemversagen droht. Dies alles hat zur Konsequenz, daß der Datenbankadministrator den internen Zustand der Datenbank (physische Ebene) kontinuierlich überwachen und ggf. mittels **Datenbankreorganisation** eingreifen muß. Die Datenbankreorganisation gestattet es ihm dann, etwa Speicher neu zuzuteilen, Fragmente zu beseitigen, Sortierordnungen intern (wieder)herzustellen oder die Datenplatzierung geänderten Erfordernissen anzupassen. Das Thema Datenbankreorganisation an sich ist somit natürlich nicht insgesamt brandneu. In der Praxis sind Datenbankadministratoren damit seit Jahren konfrontiert und gezwungen, Lösungen zu finden. Oftmals geschieht dies ziemlich *ad hoc* und wenig fundiert und begründet. Aus mehreren Ursachen heraus tritt hier in jüngerer Zeit eine Verschärfung der Problematik auf, die es angeraten erscheinen läßt, das Thema Datenbankreorganisation intensiv und wissenschaftlich fundiert zu betrachten:

- Immer größer werdende Datenbanken auf der einen Seite (im hohen dreistelligen Gigabytes-Bereich oder gar im Terabytes-Bereich) treffen zusammen mit immer anspruchsvolleren Verfügbarkeitsanforderungen (in Richtung auf einen 24-Stunden-Betrieb an 365 Tagen im Jahr). Die klassische Datenbankreorganisation belegt (zumindest) Teile der Datenbank für die Dauer ihrer Durchführung exklusiv, was also unmittelbar im Konflikt steht mit hohen Verfügbarkeitsanforderungen.
- Die größer werdenden Datenbanken und sehr große einzelne Tabellen darin - mit manchmal -zig Millionen Zeilen in einer Tabelle - lassen das bisher übliche Reorganisationsgranulat, eine (komplette) Datenbanktabelle, zudem als äußerst problematisch erscheinen. Neuere DBMS-Produktversionen ermöglichen u.a. deshalb die Partitionierung sehr großer Tabellen auf physischer Ebene mit Durchführung einer Reorganisation dann ggf. auf der einzelnen Partition. Hier kommen also für den Datenbankadministrator neue Parameter bzgl. der Datenbankreorganisationsdurchführung ins Spiel (neue „Stellschrauben“ sozusagen, an denen es zu drehen gilt).
- Mit den größer werdenden Datenbanken erscheint es schließlich immer weniger opportun, eine Datenbankreorganisation einfach auf Verdacht durchzuführen bzw. periodisch, ohne daß ganz klar Nutzen und Kosten davon belegt (und vorhergesagt!) werden können. Das Thema **Analyse von Reorganisationserfordernissen** gewinnt somit an Bedeutung. So weisen Datenbanken im SAP System R/3 mehr als 10 000 Tabellen auf. Bei vielen dieser Tabellen ist dem Administrator von vornherein klar, ob Datenbankreorganisation ein Thema sein mag oder nicht. Bei vielen anderen Tabellen ist dies jedoch so nicht unmittelbar klar, und vor allem genaue Reorganisationszeitpunkte und -umfänge sollten nicht nur intuitiv festgelegt werden. Entscheidungshilfen auf quantitativer Ebene sind hier gefragt bei der Analyse, und die Erfahrung zeigt, daß die heute verfügbaren relationalen DBMS-Produkte dafür bisher nur wenig Hilfestellung geben.

Im vorliegenden Beitrag wird auf die Datenbankreorganisationsthematik im Überblick und in ihren Möglichkeiten eingegangen, wobei zur Veranschaulichung und als konkreter Untersuchungsgegenstand auf das relationale DBMS-Produkt Informix zurückgegriffen wird. Der Beitrag gliedert sich entsprechend wie folgt: Nach einer begrifflichen Einführung zur Datenbankreorganisation in **Kapitel 2** und einer Erörterung von Gründen und Zielen hierbei wird in **Kapitel 3** auf Probleme, Anforderungen und Granulate bei der Datenbankreorganisation näher eingegangen. **Kapitel 4** präsentiert eine Klassifikation zur impliziten bzw. expliziten Vorgehensweise bei der Datenbankreorganisation, und **Kapitel 5** geht dann auf die Aufgabenstellungen und Lösungsansätze bei Informix in einigem Detail ein. Diese Darstellungen machen auch mit deutlich, daß zur diskutierten Thematik noch zahlreiche Defizite existieren in heutigen DBMS-Produkten à la Informix, bspw. was die Analysemöglichkeiten für Reorganisationserfordernisse anbelangt oder die angebotenen Möglichkeiten zur eigentlichen Reorganisationsdurchführung. **Kapitel 6** bietet eine Zusammenfassung und einen Ausblick auf künftige Arbeiten und Forschungsschwerpunkte.

2 Reorganisation von Datenbanken

2.1 Gründe für eine Datenbankreorganisation

Die Gründe, die zur Durchführung einer Datenbankreorganisation führen, sind verschieden. So kann es im Laufe der Nutzung einer Datenbank, abhängig vom Datenbank-Management-Systems (DBMS) und der Nutzungsart, zu einer schleichenden Leistungsver schlechterung kommen oder das System stößt, wie oben erwähnt, an Grenzen, die bei der Implementierung festgelegt wurden. Es wird nach einer gewissen Nutzungszeit notwendig, diese Probleme zu beseitigen. In dem Zusammenhang spricht man häufig von einer Datenbankreorganisation.

Ein Grund für die Leistungsver schlechterung können Degenerierungen der zur Datenspeicherung und zum Wiederauffinden der Daten verwendeten Strukturen sein, wenn diese aus Performance-Gründen bei Änderungsoperationen nicht vollständig oder nicht sofort gepflegt werden. So werden bei vielen Systemen bei Einfüge- und Änderungsoperationen Sortierreihenfolgen innerhalb der Datenbestände nicht beachtet oder es kommt zur Fragmentierung des Datenbestands. Fragmentierungen treten auf, wenn logisch zusammengehörige Daten durch das Freispeicher-Management auf mehrere, physisch voneinander entfernt liegende Speicherbereiche verteilt werden. Im Zusammenhang mit der Unterstützung variabel langer Tupel kann es dazu kommen, daß nach tupelverlängernden Änderungsoperationen die geänderten Tupel nicht mehr am ursprünglichen Ort gespeichert werden können. Sie werden an einen anderen Speicherort verschoben. Am ursprünglichen Speicherort wird, um das Pflegen von Indexen zu vermeiden, nur noch ein Zeiger auf den neuen Speicherort abgelegt. Man spricht dann von Auslagerungszeigern. Diese wirken sich auf die Suchoperationen aus. Nach [Sha95] liest das von Oracle verwendete Verfahren zum sequentiellen Durchsuchen einer Relation, im Falle daß ein solcher Auslagerungszeiger gefunden wird, zunächst die Seite, auf die der Zeiger verweist. Erst danach werden die Seiten, die physisch nach der Originalseite liegen gelesen. Damit können solche Auslagerungszeiger sehr schnell die Anzahl notwendiger Positionierungsoperationen steigern und damit Zugriffszeiten erhöhen. Weiterhin kommt es, im Laufe der Arbeit mit der Datenbank oft zu einer verschlechterten Auslastung des zur Datenspeicherung belegten Sekundärspeichers.

Auch Änderungen in der Art der Nutzung des Datenbestands können eine Neuorganisation erfordern. Nach [MHR96] sind in einer operativen Datenverarbeitung mit meist kurzen Transaktionen, bei denen häufig auch Änderungen am Datenbestand vorgenommen werden, andere Organisationsformen des Datenbestands sinnvoll, als bei einem System zur Unterstützung von Entscheidungsfindungsprozessen mit meist langen und aufwendigen sequentiellen Leseprozessen. Bei überwiegend lesender Nutzung kann z.B. der verwendete Sekundärspeicher von vornherein besser ausgenutzt werden, ohne daß die Gefahr besteht, daß es durch die hohe Speicherausnutzung und somit fehlenden freien Platz häufig zum Reservieren und Belegen neuer Speichereinheiten kommt.

Nach [LS93] können die Gründe, die zur Durchführung einer Datenbankreorganisation führen, in zwei Klassen eingeteilt werden: zwingende Gründe und solche, die Anpassungen empfehlenswert erscheinen lassen.

Zwingende Gründe sind z.B. das Erreichen der oben erwähnten Grenzen (Schwellwerte), welche vom Datenbank-Management-System oder vom Betriebssystem vorgegeben werden, wie z.B. maximale Dateigrößen, maximale Anzahl verwaltbarer Speichereinheiten (z.B. Extents je Relation) usw. Aber nicht nur Gründe, die in direktem Zusammenhang mit dem DBMS oder der Systemplattform stehen, sondern auch Gesetzesänderungen (z.B. von Bestimmungen zum Datenschutz) können eine Datenbankreorganisation erzwingen.

Bei einer unwirtschaftlichen Speicherplatzausnutzung oder aus Performance-Gründen, z.B. weil Sortierreihenfolgen nicht eingehalten werden und somit oft aufwendige Sortierläufe notwendig sind, ist eine Datenbankreorganisation oft empfehlenswert. Allerdings sollte dabei der Aufwand für die Reorganisation dem zu erwartenden Nutzen gegenübergestellt werden.

2.2 Begriff Datenbankreorganisation

Nach [TL91] werden als Datenbankreorganisation „alle Maßnahmen zur Wiederherstellung der Bedingungen für eine effiziente Datenmanipulation - vergleichbar mit dem Leistungsverhalten zu Nutzungsbeginn - “ zusammengefaßt.

Diese recht allgemeine Definition schließt neben der Reorganisation der eigentlichen Daten auch Reorganisationen des logischen Datenbankschemas ein (in [LS93] als Restrukturierung bzw. häufig auch als Schemaevolution bezeichnet). Weiterhin kann nach dieser Definition auch eine Neuabstimmung von Systemparametern, die die Performance des DBMS beeinflussen (z.B. Größe des Puffer-Pools, Parameter zur Steuerung eines vorausschauenden Lesens usw.), zu den Reorganisationsmaßnahmen gezählt werden.

Der Schwerpunkt soll in den folgenden Abschnitten allerdings auf der Reorganisation der eigentlichen Daten auf der physischen Ebene (Datenreorganisation, bzw. nach [LS93] Reformatierung) liegen.

2.3 Ziele einer Datenbankreorganisation

Bei den oben genannten Datenbankreorganisationen aus zwingenden Gründen ist das Hauptziel die Aufrechterhaltung des Betriebs des Datenbanksystems bzw. der Anwendung. Eine Reorganisation der Datenbank kann aber auch aus Gründen empfehlenswert sein, die

den Datenbankbetrieb nicht direkt gefährden. Mit einer solchen Reorganisation können die folgenden Ziele verfolgt werden:

- Beseitigung von Degenerierungen (Fragmentierungen, nicht eingehaltene Sortierreihenfolgen usw.) in Datenbeständen und Zugriffspfaden
- Freigabe von nicht mehr benötigtem, aber reserviertem Speicherplatz
- das Erreichen eines definierten Belegungsgrads der Speicherblöcke für Daten und Indexe
- Beschleunigung der Datenzugriffe
- Anpassung der physischen Datenspeicherung an Veränderungen in der Nutzungsform des Datenbestands
- Beseitigung von Problemen, die durch das baldige Erreichen von Schwellwerten, die durch das DBMS gegeben sind, entstehen könnten.
- Abstimmung von Parametern, welche die physische Datenspeicherung beeinflussen, an die zu speichernden Daten. So kann bei existierenden Datenbank-Management-Systemen mit weniger Systemaufwand auf Binärobjekte (Binary Large Objects bzw. kurz BLOBs) zugegriffen werden, wenn die zur deren Speicherung verwendete Seitengröße größer als das darin gespeicherte Binärobjekt ist. Da aber je Seite nur ein Objekt gespeichert wird, darf für eine ökonomische Speicherausnutzung diese sog. BLOB-Seitengröße nicht zu hoch gewählt werden.

3 Probleme, Einflußgrößen und Anforderungen

3.1 Probleme bei einer Datenbankreorganisation

Ein Problem stellt das Erkennen der Notwendigkeit einer Datenbankreorganisation dar. Von vielen Datenbank-Management-Systemen werden Möglichkeiten zum Erkennen der Notwendigkeit einer Datenbankreorganisation nur sehr wenig unterstützt oder diese Möglichkeiten werden nur unzureichend und unübersichtlich dokumentiert. Eine Reorganisation der Datenbestände erfolgt deshalb, wie bereits erwähnt, oftmals nur, weil ein gewisser Zeitraum abgelaufen ist oder eher unscharfe Indikatoren (wie z.B. „Performance stimmt nicht mehr“) es ratsam erscheinen lassen. Eine tatsächliche Degenerierung der Datenbank oder von Teilen der Datenbank bleibt oft unberücksichtigt bzw. kann nicht genau genug lokalisiert werden, weil keine hinreichend genauen Informationen darüber vorliegen oder diese nicht ausgewertet werden.

Ein weiteres Problem liegt darin, daß sich die möglichen Vorgehensweisen zur Reorganisation eines Datenbestands, die von den einzelnen Datenbank-Management-Systemen angeboten werden, teilweise stark unterscheiden. Bei manchen Systemen besteht die Möglichkeit zur Reorganisation der Datenbestände nur darin, die Daten auszulagern, betroffene Daten- und Schemaelemente zu zerstören, danach die Schemaelemente neu aufzubauen und die ausgelagerten Daten in die Datenbank wieder einzulagern. Andere Systeme erlauben es, einzelne Relationen intern zu reorganisieren, indem man dafür sorgt, daß die Daten der Relation physisch umkopiert werden.

Eine Datenbankreorganisation ist, abhängig von der Menge der gespeicherten Daten, oft ein zeitaufwendiger Prozeß. Je nach Art der Reorganisation kommt es während dieser zu mehr oder weniger starken Einschränkungen im normalen Datenbankbetrieb. Solche Einschränkungen reichen von einer vorübergehenden Verschlechterung des Antwortzeitverhaltens durch die zusätzliche Belastung des DBMS bis dahin, daß die Datenbank oder Teile davon zeitweilig nicht verfügbar sind. Im Zusammenhang mit neuartigen Formen der Datenbanknutzung (z.B. intelligente Kommunikationsnetze), aber auch bei herkömmlichen Nutzungsformen (bspw. Steuerung von Geldautomaten mittels einer datenbanksystembasierten Lösung) werden Forderungen nach einer ständigen Verfügbarkeit der Datenbank stärker. Es ist dann, wenn überhaupt realisierbar, ein hoher Aufwand für die Vorbereitung und Durchführung der Datenbankreorganisation notwendig, um die oben erwähnten Einschränkungen möglichst gering zu halten.

3.2 Faktoren, die eine Datenbankreorganisation beeinflussen

Wann und wie die Reorganisation einer Datenbank durchgeführt werden kann, hängt von vielen Faktoren ab. Wichtig ist es vor allem, Aufwand und Nutzen gegeneinander abzuwägen. Einen wichtigen Faktor stellen die Reorganisationsmöglichkeiten, die das DBMS bietet, dar. Beachtung sollte dabei vor allem den möglichen Granulaten (zu reorganisierende Einheiten) für eine Datenbankreorganisation und der technischen Durchführung geschenkt werden. Eine Datenbankreorganisation, die mit hohem technischen Aufwand (Entladen der Datenbank, Zerstören und Neuanlegen der Schemaelemente, Laden der Daten) verbunden ist und darüber hinaus die Verfügbarkeit der Datenbank stark einschränkt, ist sicherlich seltener sinnvoll durchführbar als eine lokale Reorganisation einzelner Relationen.

Auch das vorhandene Schema einer Datenbank kann die Art und Häufigkeit ihrer Reorganisation beeinflussen. Vor allem die Wahl der Datentypen (z.B. Zeichenketten fester Länge oder Zeichenketten variabler Länge) hat Einfluß darauf, ob es durch Verlängerungen von Tupeln zu Auslagerungen und, damit verbunden, zu Auslagerungszeigern kommt.

Ein weiterer Faktor, der eine Datenbankreorganisation beeinflusst, ist die Nutzungsform der Datenbank. So wird eine Datenbank, die überwiegend lesend genutzt wird, wesentlich langsamer degenerieren als eine Datenbank, an der häufig Änderungen am Datenbestand vorgenommen werden. Auch die Art des Änderungsbetriebs spielt eine Rolle. Eine Mischung aus Einfüge-, Änderungs- und Löschooperationen führt sicherlich schneller zu starken Fragmentierungen als überwiegende Einfügeoperationen.

Bei einer gemischten Nutzung von Lese- und Änderungsoperationen muß meist ein Kompromiß zwischen der Abfragebeschleunigung, die mittels Indexen erreicht werden kann und den Speicherkosten, welche durch sie verursacht werden, gefunden werden. Weiterhin sind die CPU-Zeit-Kosten, die bei Einfüge- und Löschooperationen durch die Pflege der Indexe zusätzlich verursacht werden, zu berücksichtigen. Bei Datenbanken, die auch zur Unterstützung von Entscheidungsfindungsprozessen genutzt werden, muß ein Kompromiß gefunden werden zwischen einerseits der Ressourcenverteilung zu Gunsten paralleler Abfragen, welche die meist komplexen Anfragen der Entscheidungsfindungsprozesse unterstützen und andererseits den normalen Operationen des Tagesgeschäfts. In diesen Fällen muß das Verhältnis des Auftretens der unterschiedlichen Nutzungsformen berücksichtigt werden.

Die Systemplattform, auf der die Datenbanklösung installiert wurde, spielt ebenfalls eine Rolle bei der Entscheidung, wie eine Reorganisation der Datenbank durchzuführen ist. Sie

hat wesentlichen Einfluß auf die Ziele einer Reorganisation. Vor allem die Parallelisierung von Datenbankoperationen setzt die Unterstützung von seiten der Hardware und des Betriebssystems voraus. So benötigt eine Partitionierung von Relationen sinnvollerweise die Unterstützung von parallelen Lese- und Schreiboperationen auf verschiedenen Datenträgern durch Hardware und Betriebssystem.

Nicht zuletzt hat der Typ des zur Datenspeicherung verwendeten Speicherplatzes (Raw Device/ Dateisystem/Virtual Disks) Einfluß auf Durchführung und Häufigkeit einer Datenbankreorganisation. So kann nach [Nob95] das während einer Reorganisation häufig notwendige Aufbauen von Indexen durch die Verwendung des von manchen Unix-Versionen angebotenen „disk-striping“, bei dem auf Betriebssystemebene Daten auf mehrere Datenträger verteilt werden (entsprechende Hardware vorausgesetzt), erheblich beschleunigt werden.

3.3 Anforderungen an eine Datenbankreorganisation

Ausgehend von den oben genannten Problemen bei der Erkennung von Reorganisationserfordernissen und der Reorganisationsdurchführung sollten folgende Anforderungen erfüllt werden.

Das DBMS sollte den Datenbankadministrator (DBA) bei der Bestimmung des Reorganisationserfordernisses unterstützen und möglichst gezielt Hinweise geben, wo und wann eine Datenbankreorganisation angesetzt werden sollte. In diesem Zusammenhang muß eine Abschätzung des für die Reorganisation notwendigen Aufwands und des zu erwartenden Nutzens möglich sein.

Bezüglich der Vorbereitung und Durchführung einer Reorganisation von Datenbanken sollte für den Durchführenden (z.B. den DBA) möglichst wenig organisatorischer Aufwand, wie Bereitstellung von Bändern oder Plattenspeicher, entstehen. Die Durchführung der Reorganisation sollte möglichst einfach und systemkontrolliert sein, um z.B. Bedienungsfehler zu vermeiden.

Es muß möglich sein, gezielt auch nur Teile einer Datenbank zu reorganisieren, da die einzelnen Teile einer Datenbank unterschiedlich genutzt werden und damit in einem bestimmten Zeitraum auch unterschiedlich stark degenerieren (z.B. Stammdaten werden überwiegend lesend genutzt, während sich zum Beispiel Lagerbestandsdaten sicher häufig ändern).

Die Reorganisation eines Datenbankteils sollte den übrigen Datenbankbetrieb möglichst wenig beeinflussen. Eine wichtige Rolle spielt hier die Verfügbarkeit der Daten während der Reorganisation.

3.4 Granulate für eine Datenbankreorganisation

Die Granulate, die für eine Datenbankreorganisation denkbar sind, lassen sich grob in zwei Gruppen einteilen. Einerseits existieren logische Granulate wie Datenbanken, Relationen oder Mengen von Relationen. Andererseits lassen sich physische Granulate bilden, die zum Teil vom verwendeten DBMS abhängig sind. Zu den physischen Granulaten zählen Partitionen, Zugriffshilfen und die vom Datenbank-Management-System verwendeten Speichereinheiten. Die logischen und physischen Granulate werden nachfolgend näher erläutert.

Für eine Datenbankreorganisation können unterschiedliche Granulate gewählt werden. Das größte Granulat stellt **die gesamte Datenbank** dar. Damit werden alle Strukturen der Datenbank wieder bereinigt und es kann, bei ordnungsgemäßer Durchführung, mit einem unter den gegebenen Umständen „optimalen“ Ergebnis gerechnet werden. Allerdings ist eine solche Reorganisation meist auch sehr arbeits- und zeitintensiv, und ein globales Optimum im Ergebnis wird nur selten erreicht. Weiterhin sollte beachtet werden, daß hier auch Teile der Datenbank reorganisiert werden, bei denen eine Reorganisation nicht oder noch nicht erforderlich gewesen wäre.

Als weiteres Granulat für eine Datenbankreorganisation ist **die Relation** denkbar. Die Verwendung dieses Granulats erlaubt die gezielte Reorganisation degenerierter Bereiche der Datenbank. Es eignet sich besonders zum Wiederherstellen von Sortierungen innerhalb von Relationen und zur Freigabe von Speicher.

Im Falle der Nutzung von Konzepten, die eine physisch zusammenliegende Speicherung von Daten aus mehreren Relationen erlauben, wie z.B. das später noch beschriebene Cluster-Konzept des DBMS Oracle, ist als weiteres Granulat eine **Menge von Relationen** notwendig, da bei der Reorganisation eines solchen Clusters mehrere Relationen betroffen sind. Auch bei der Reorganisation physischer Speichereinheiten sind mehrere Relationen betroffen. Allerdings steht hier die Reorganisation bestimmter Relationen nicht direkt im Vordergrund.

Im Zusammenhang mit Konzepten zur Partitionierung von Relationen ist, vor allem bei sehr großen Relationen, auch eine **Partition** einer Relation denkbar. Eventuell muß eine solche Reorganisation bei den derzeit verfügbaren Datenbank-Management-Systemen über Umwege (Herauslösen der entsprechenden Partition aus der Relation in eine eigenständige Relation, Reorganisation dieser neu entstandenen Relation und abschließendes Wiedereingliedern des reorganisierten Teils als Partition in die Ausgangsrelation) erfolgen.

Auch **einzelne Zugriffshilfen** (Indexe) können reorganisiert werden, um zum Beispiel definierte Füllungsgrade für die Indexseiten wiederherzustellen. Oftmals müssen dabei die Indexe zerstört und neu aufgebaut werden. Werden alle Indexe einer Relation reorganisiert, so könnten beim Neuaufbau der Indexe die realen Speicherorte von Tupeln berücksichtigt werden. Damit könnten Auslagerungszeiger beseitigt werden, ohne daß Daten bewegt werden müssen. Probleme können dann allerdings auftreten, wenn, abweichend von der „reinen Lehre“, der Verweis auf die Tupel (oft als ROWID bezeichnet) in Applikationen abfragbar ist (wie z.B. bei Informix) und dort z.B. als Primärschlüssel verwendet wird.

Abhängig vom Speichermodell, welches das DBMS verwendet, können auch andere Granulate, die sich auf die Einheiten des Speichermodells beziehen (z.B. Table Spaces), sinnvoll sein. In Kapitel 5 wird auf solche Granulate am Beispiel des DBMS Informix noch näher eingegangen.

4 Methoden zur Datenbankreorganisation

Eine Datenbankreorganisation kann explizit veranlaßt werden oder implizit erfolgen. Für die explizit veranlaßte Datenbankreorganisation kann hier nochmals zwischen einer datenbanksystembasierten Reorganisation und einer datenbanksystemintegrierten Reorganisation unterschieden werden.

Das folgende Bild zeigt die Unterteilung der Reorganisationsmethoden für Datenbanken.

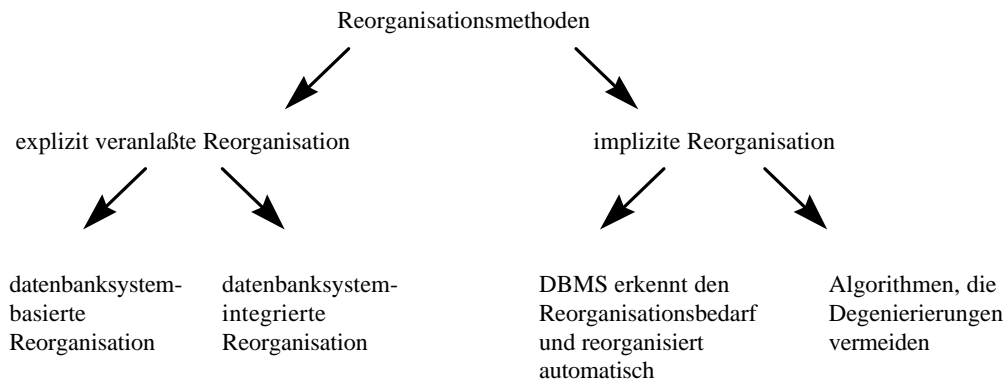


Abbildung1: Einteilung der Reorganisationsmethoden für Datenbanken

4.1 Methoden zur explizit veranlaßten Datenbankreorganisation

Zur Durchführung einer manuellen (explizit veranlaßten) Reorganisation bieten viele Datenbank-Management-Systeme sowohl Möglichkeiten zur datenbanksystembasierten Reorganisation als auch (seltener) solche zur datenbanksystemintegrierten Reorganisation an. Diese Möglichkeiten werden im folgenden kurz beschrieben.

4.1.1 Datenbanksystembasierte Reorganisation

Als datenbanksystembasierte Reorganisation soll eine Reorganisationsmethode bezeichnet werden, die in der Regel unter Verwendung von Hilfsprogrammen (Utilities) durchgeführt wird und die aufgesetzt auf einem DBMS realisiert ist (z.B. Dienstprogramme zum Export bzw. Import von Daten).

Bei dieser Methode geht man in vier Hauptarbeitsschritten vor:

- Entladen des zu reorganisierenden Datenbestands/Bestandsausschnitts
- Zerstören der Schemaelemente
- Neuanlegen der zuvor gelöschten Schemaelemente
- Laden der zuvor entladenen Daten

Bietet das DBMS Speichereinheiten an, in denen mehrere Relationen abgelegt werden können, und sollen mehrere dieser Relationen reorganisiert werden, so ist zu beachten, daß nach dem Löschen der Schemaelemente eine Relation angelegt, danach die Daten geladen und zuletzt die Indexe aufgebaut werden sollten, bevor die gleiche Vorgehensweise für die nächste Relation wiederholt wird. Damit soll erreicht werden, daß der Speicher für jede einzelne Relation beim Laden der Daten fortlaufend reserviert werden kann. Außerdem ist das Anlegen der Indexe nach dem Laden der Daten mit weniger Systemaufwand verbunden als das Pflegen der Indexe während des Ladens und erlaubt zudem eine einfachere Einflußnahme etwa auf die Speicherplatzausnutzung im Index.

Der Nachteil dieser Methode liegt in dem hohen manuellen Aufwand für den Durchführenden (Bereitstellen von Speichermedien, aufwendige Durchführung der Datenbankreorgani-

sation durch Entlade- und Ladeoperationen, DROP- und CREATE-Anweisungen). Weiterhin unterliegen die Daten, nachdem sie entladen und die Schemaelemente zerstört wurden, bis zum (evtl. vollständigen) Wiedereinlagern nicht den Integritätssicherungsmechanismen des DBMS. Deshalb ist die Erstellung eines Datenbank-Backups vor der Nutzung dieser Reorganisationsmethode zu empfehlen.

4.1.2 Datenbanksystemintegrierte Reorganisation

Hier bietet das DBMS (z.B. auf erweiterter SQL-Ebene) Möglichkeiten an, einen Datenbestand oder die entsprechenden Zugriffspfade ohne Aus- und Einlagern des Datenbestands zu reorganisieren. Die eigentliche Datenbankreorganisation muß aber vom Benutzer bzw. dem Datenbankadministrator veranlaßt werden. Oftmals ist eine solche Reorganisation mit einem impliziten Umkopieren von Daten, innerhalb der vom DBMS verwalteten Speicherstrukturen, verbunden. Das setzt voraus, daß innerhalb des vom DBMS verwalteten Speichers genügend möglichst zusammenhängender freier Platz (zur Beseitigung von Fragmentierungen) zur Verfügung steht.

Da bei dieser Methode die Reorganisationsfunktionalität im DBMS integriert realisiert wird, ist die potentielle Fehleranfälligkeit geringer. Die Daten unterliegen der Kontrolle durch das DBMS. Weiterhin ist eine Reorganisation meist durch die Verwendung nur eines Befehls je Reorganisationseinheit (z.B. je Relation) möglich. Allerdings besitzt diese Methode auch Grenzen, z.B. wenn innerhalb des vom DBMS verwalteten Speichers nicht genügend freier Speicher für die Kopie der Reorganisationseinheit zur Verfügung steht. Sollen Fragmentierungen beseitigt werden, so muß dieser freie Speicher auch fortlaufend vorliegen.

Das DBMS IBM DATABASE 2 bietet nach [Lab96] mit REORG TABLE eine Möglichkeit an, einzelne Relationen nach dieser Methode zu reorganisieren.

4.2 Implizit veranlaßte Datenbankreorganisation

Als implizit soll eine Reorganisationsform bezeichnet werden, bei der, im Unterschied zu den beiden oben genannten Methoden, das DBMS ein Reorganisationserfordernis erkennt und automatisch eine Reorganisation ausführt. Allerdings verursacht eine solche automatische Datenbankreorganisation i.d.R. Systemlast, welche die Performance, auch zu Zeiten hoher Belastung durch die Benutzer, negativ beeinflussen kann. Dies ist meist nicht erwünscht. Die Entscheidung, wann die erhöhte Systembelastung durch eine Reorganisation vertretbar ist, sollte dem DBA überlassen bleiben. Deshalb nutzen viele DBMS-e automatische Reorganisationsmechanismen nur dann, wenn der Reorganisationsumfang klein ist und demzufolge die Performance nicht wesentlich beeinflußt wird (z.B. beim Zusammenführen von Freiplatz innerhalb einer Datenseite, welche sich im Puffer-Pool befindet).

Als eine weitere Form einer impliziten Datenbankreorganisation kann die Nutzung von Algorithmen angesehen werden, die während Änderungsoperationen auf den Speicherstrukturen dafür sorgen, daß diese nicht degenerieren. In diesem Zusammenhang ist teils auch von „Reorganisationsfreiheit“ die Rede (siehe auch [SAG97] bzw. [Nuß97]).

5 Datenbankreorganisation auf verschiedenen Ebenen

5.1 Reorganisationsebenen

Eine Datenbankreorganisation kann auf der Schema-, der Verteilungs- oder der internen Ebene eines Datenbanksystems durchgeführt werden.

Als Reorganisation auf **Schemaebene** (Restrukturierung) soll hier eine Form der Datenbankreorganisation bezeichnet werden, bei der das Datenbankmodell der Informationsstrukturen des Umweltausschnitts verändert wird. Das kann zum Beispiel sinnvoll sein, wenn sich die Nutzungsform des Datenbestands geändert hat und das Datenbankschema an die veränderte Nutzungsform angepaßt werden soll. Untersuchungen zur Restrukturierung von relationalen Datenbanken wurden zum Beispiel in [MM87] durchgeführt. Ursachen für Probleme, die eine solche Datenbankreorganisation auf der Schemaebene erfordern, liegen allerdings in der Regel außerhalb des Verantwortungsbereichs des verwendeten Datenbank-Management-Systems bzw. des Systemverwalters und sollen hier deshalb nicht weiter betrachtet werden.

Bei einer Datenbankreorganisation auf der **Verteilungsebene** wird der Ort der physischen Abspeicherung der Daten „global“ verändert. Das heißt, es handelt sich bei einer solchen Umverteilung nicht um eine „lokale“ Umspeicherung von Daten, wie sie etwa bei Umsortierungen vorgenommen wird, sondern bspw. um eine gezielte Verteilung der Daten einer oder mehrerer Relationen auf unterschiedliche Datenträger, welche von einem Datenbank-Server verwaltet werden. Auch eine gezielte, physisch zusammenliegende Speicherung der Tupel mehrerer Relationen, die logisch in Beziehungen zueinander stehen, ist denkbar. Als Beispiel kann hier das in [ORA93] beschriebene Cluster-Konzept von Oracle genannt werden.

Bei einer Datenbankreorganisation auf der **internen Ebene** steht das Beseitigen von Degenerierungen verschiedener Art im Vordergrund. Degenerierungen in den Datenbeständen und Zugriffspfaden entstehen häufig im Zusammenhang mit Änderungsoperationen (Einfügen, Ändern, Löschen von Daten und Indexeinträgen), weil aus Gründen einer möglichst hohen Performance während dieser Operationen z.B. auf das exakte Nachtragen von Indexinformationen, eine Datenverdichtung in Verbindung mit sofortiger Speicherfreigabe oder das Einhalten von Sortierreihenfolgen verzichtet wird. Das konkrete Erscheinungsbild der Degenerierungen unterscheidet sich leicht bei unterschiedlichen DBMS. Häufig vorkommende Degenerierungen sind u.a. Fragmentierungen, Auslagerungszeiger, nicht beachtete Sortierungen, Speicherplatz, der durch Löschoptionen frei geworden ist, aber reserviert bleibt oder „als gelöscht“ gekennzeichnete, aber zunächst noch vorhandene Indexeinträge.

5.2 Sekundärspeicherorganisation am Beispiel von Informix

Im folgenden Abschnitt soll eine von Datenbank-Management-Systemen verwendete Organisation des Sekundärspeichers am Beispiel von Informix V7.x dargestellt werden (siehe auch [INF96a]). Die Speicherorganisation und auch die Verwendung der Bezeichnungen für Speichereinheiten ist teilweise systemabhängig. Allerdings ist die Anlehnung an die Begriffswelt eines konkreten DBMS für die folgenden Betrachtungen hilfreich.

Der gesamte von Informix verwaltete Speicherplatz besteht aus einem oder mehreren sog. **DB-Spaces**. Ein DB-Space ist eine Informix-Speichereinheit zur Ablage von Datenbanken

und deren Relationen. Dabei können die Relationen einer Datenbank in mehreren DB-Spaces abgelegt werden. Innerhalb eines DB-Space können aber auch, abhängig von seiner Größe, mehrere Datenbanken abgelegt werden. Ein besonderer DB-Space, der sogenannte „Root-DB-Space“, enthält bei einer Standardkonfiguration, neben den Metadaten zur Verwaltung aller anderen Speichereinheiten, auch die Log-Daten, in denen Veränderungen, welche an den Datenbanken vorgenommen werden, aufgezeichnet werden. Eine Ablage von Benutzerdatenbanken im Root-DB-Space ist möglich.

Ein DB-Space besteht aus einem oder mehreren sog. **Chunks**. Ein Chunk entspricht auf der Betriebssystemebene einer Datei bzw. einem Gerät (Raw Device). Ein Gerät kann einem physischen Laufwerk bzw. einer Partition aus Sicht des Betriebssystems entsprechen. Ein Chunk besteht aus einer Menge von **Seiten** (Pages), deren Größe abhängig von der jeweiligen Systemplattform ist.

Die Speicherung der Daten und Indexe erfolgt innerhalb von sog. **Table Spaces**. Diese werden implizit beim Anlegen der Relationen erzeugt. In einem Table Space können sich nur Daten einer Relation befinden. Bei der Nutzung von Partitionierungsstrategien für Relationen (definiertes Verteilen der Tupel einer Relation auf unterschiedliche Datenträger) wird für jede Partition ein eigener Table Space verwendet. Ein Table Space befindet sich in genau einem DB-Space.

Die Reservierung des Speichers für einen Table Space erfolgt in Form von sog. **Extents**. Dabei besteht ein Extent aus einer Menge von physisch zusammenhängenden Seiten, die zur Daten- und Indexspeicherung verwendet werden.

Die Abbildung 2 soll den Zusammenhang der Speicherstrukturen noch einmal beispielhaft verdeutlichen.

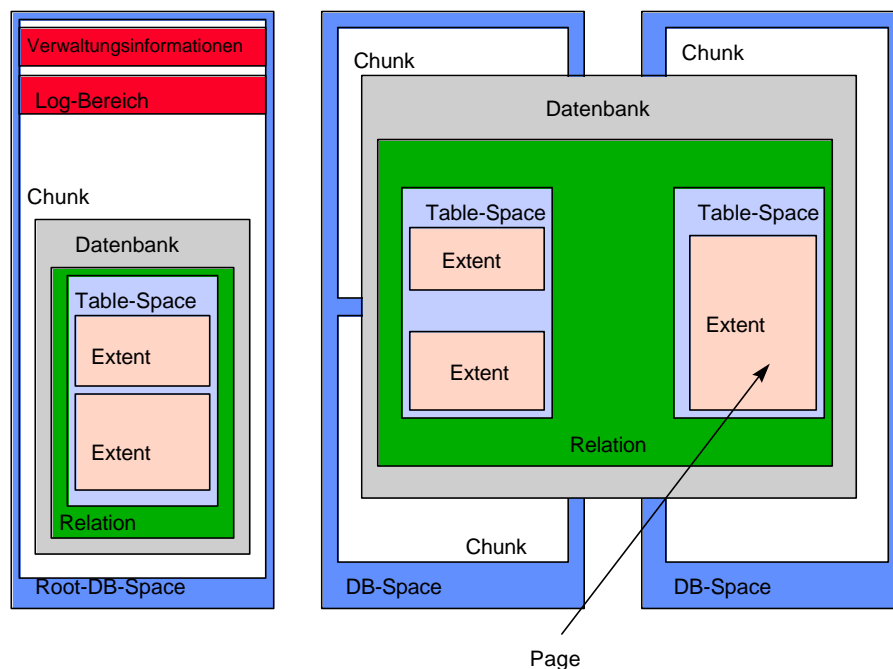


Abbildung 2: Speicherstrukturen von Informix

5.3 Datenbankreorganisation auf der Verteilungsebene

Zur Datenbankreorganisation auf der Verteilungsebene bieten sich folgende Möglichkeiten an, die nachfolgend noch näher erläutert werden:

- Verteilen der Schemaelemente einer Datenbank auf unterschiedliche Datenträger
- Verlagern der Log-Daten bzw. der Metadaten auf bisher wenig frequentierte bzw. schnelle Datenträger
- Partitionierung von Relationen/Indexen
- physisch zusammenliegende Speicherung von Tupeln mehrerer Relationen

5.3.1 Verteilen der Schemaelemente

Viele Datenbank-Management-Systeme bieten die Möglichkeit, beim Anlegen von Schemaelementen den Ort anzugeben, an dem das Element gespeichert werden soll. Meist ist diese Ortsangabe eine logische Speichereinheit des DBMS (z.B. DB-Space bei Informix). Durch eine gezielte Zuordnung dieser logischen Einheiten zu den physischen Einheiten der Speicherorganisation (z.B. den Chunks bei Informix) können die Schemaelemente gezielt auf einzelne Datenträger verteilt werden.

Die Anweisungen `CREATE DATABASE` und `CREATE TABLE` kennen bei Informix eine `IN`-Klausel, die die Angabe des DB-Space enthält, in dem die Datenbank bzw. die Relation angelegt werden soll (siehe auch [INF96e]).

5.3.2 Verlagern der Log-Daten bzw. der Metadaten

Meist speichern Datenbank-Management-Systeme die Log-Daten und die Metadaten, welche Beschreibungsinformationen über die Datenbank (z.B. Schemata, Speichereinheiten usw.) enthalten, in speziellen Bereichen. Informix legt diese Daten standardmäßig im Root-DB-Space ab. Da für die meisten Datenbankoperationen die Metadaten benötigt werden und auch Log-Einträge erzeugt werden, ist es oft sinnvoll, Log-Daten und Metadaten auf physisch unterschiedliche Datenträger zu verlagern, um eine Lastverteilung zu erreichen. Auch aus Gründen der Ausfallsicherheit und Wiederherstellung von Daten im Fehlerfall erscheint dies angeraten.

5.3.3 Partitionierung von Relationen/Indexen

Als Partitionierung von Relationen wird die Verteilung der Tupel und Indexe, die zu einer Relation gehören, auf mehrere Speichereinheiten bezeichnet. Dies wird oft auch genauer als horizontale Partitionierung bezeichnet.

Informix legt für jede Partition einer Relation einen eigenen Table Space an. Durch die Zuordnung der Partitionen zu DB-Spaces, denen wiederum Chunks zugeordnet werden, kann der physische Speicherort der Tupel einer Partition festgelegt werden. Dies kann mittels der Anweisungen `CREATE TABLE` oder `ALTER FRAGMENT` realisiert werden.

Partitioniert werden kann nach unterschiedlichen Strategien. Zu nennen wären hier die Round-Robin-Strategie, d.h. die Tupel der Relation werden gleichmäßig auf alle Partitionen verteilt. Damit kann eine Balancierung der Speicherauslastung und der I/O-Last bei großen, häufig frequentierten Relationen erreicht werden. Auf Ausdrücken basierende Strategien bieten die Möglichkeit einer gezielten, werteabhängigen Verteilung der Tupel. Damit ist es unter bestimmten Bedingungen seitens des DBMS-Optimizers möglich, gezielt Suchoperationen auf bestimmte Partitionen zu beschränken. Damit kann das bei einer Suchoperation zu verarbeitende Datenvolumen erheblich reduziert werden.

5.3.4 Physisch zusammenliegende Speicherung von Tupeln mehrerer Relationen

Das DBMS Oracle bietet eine interessante Möglichkeit an, Daten aus mehreren Relationen, die bspw. häufig durch Join-Operationen verknüpft werden, physisch zusammenliegend abzuspeichern. Dabei werden nach [Lab96] die Tupel der unterschiedlichen Relationen über einen Cluster-Schlüssel, welcher dem Join-Attribut entspricht, miteinander verknüpft. Dazu wird ein sog. Cluster-Index verwendet. Das folgende Bild soll die Abspeicherung von Daten in einem Cluster verdeutlichen.

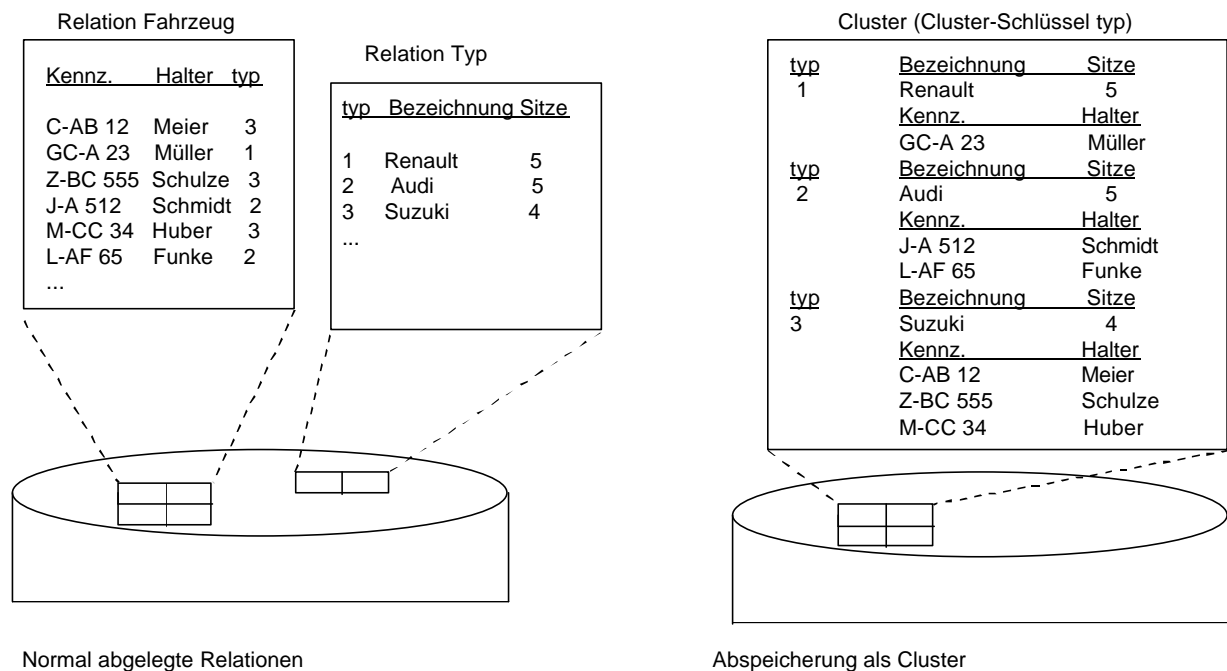


Abbildung 3: Cluster-Konzept von Oracle (nach [ORA93])

Diese Speicherungsform kann die Anzahl Plattenzugriffe bei gemeinsamem Zugriff auf logisch zusammengehörige Daten reduzieren und damit zu einer Verbesserung der Antwortzeit beitragen. Weiterhin kann, durch die nur noch einmalige Ablage des Cluster-Schlüssels im Cluster-Index, eine Speicherplatzersparnis gegenüber der Speicherung der Daten in separaten Relationen erreicht werden.

5.4 Datenbankreorganisation auf der internen Ebene

5.4.1 Beseitigung von Fragmentierungen

Eine Form der auf der internen Ebene auftretenden Degenerierungen sind Fragmentierungen. Dabei werden logisch zusammengehörige Daten durch das Freispeicher-Management auf mehrere physisch voneinander entfernt liegende Speicherbereiche verteilt. Das führt hauptsächlich zu einer erhöhten Anzahl von Positionierungsoperationen der Lese-/Schreibköpfe und damit zu Geschwindigkeitseinbußen. Weiterhin kann oftmals der Speicherplatz nicht optimal ausgenutzt werden, weil vorhandene freie Speicherbereiche zu klein für die Aufnahme zu speichernder Objekte sind.

Durch die von Informix verwendete Speicherorganisation können Fragmentierungen dort auf den nachfolgend beschriebenen Ebenen auftreten, welche bei anderen Datenbank-Management-Systemen in ähnlicher Form vorkommen:

- **Seitenebene**
Durch Lösch- und Update-Operationen kommt es zu Fragmentierungen innerhalb der zur Speicherung von Tupeln verwendeten Seiten, z.B. wenn ein Tupel, welches in der Seitenmitte gespeichert war, gelöscht wird oder durch eine verlängernde Änderungsoperation in eine andere Seite verschoben werden mußte. Das Problem wird vom DBMS automatisch behoben. Dabei wird (z.B. bei Oracle, Informix) beim Einfügen eines Tupels in eine Seite geprüft, ob genügend zusammenhängender Freiplatz zur Aufnahme des Tupels vorhanden ist. Ist das nicht der Fall, aber ist grundsätzlich genügend Freiplatz in der Seite vorhanden, nur eben in mehreren Fragmenten, so werden die freien Bereiche durch Neuordnung der Tupel innerhalb der Seite zu einem freien Speicherbereich zusammengeführt, welcher dann zur Abspeicherung des Tupels verwendet wird. Da dieses Ad-hoc-Umordnen in einer Seite im Puffer-Pool vorgenommen wird, sehen viele DBMS den entstehenden Aufwand als hinnehmbar an. Hier liegt somit ein bereits erwähnter Fall von impliziter Reorganisation durch das DBMS vor.
- **Extent-Ebene**
Werden durch größere Löschoptionen ganze Seiten frei, so wird z.B. bei Informix und den meisten anderen Datenbank-Management-Systemen der Freiplatz nicht automatisch zusammengeführt. Damit entstehen Fragmentierungen derart, daß sich zwischen freien Seiten auch Seiten befinden, die Tupel enthalten. Das Problem läßt sich durch die Reorganisation der entsprechenden Relation, die explizit veranlaßt werden muß, nach der datenbanksystembasierten Methode oder, wenn vom DBMS angeboten, nach der datenbanksystemintegrierten Methode beseitigen.
- **Table Space-Ebene**
Die Speicherplatzreservierung für Relationen erfolgt bei Informix in Form von Extents. Werden innerhalb eines DB-Space mehrere Relationen abgespeichert, kommt es oftmals dazu, daß die einzelnen Extents einer Relation nicht fortlaufend angeordnet sind, und das führt bei ungünstiger Wahl der Extent-Größen verstärkt dazu, daß Speicher, obwohl er keine Daten enthält, nicht von anderen Relationen genutzt werden kann. Das liegt daran, daß Extents, die einmal einem Table Space zugeordnet sind, nicht automatisch freigegeben werden, auch wenn sie keine Daten enthalten. Diese Art der Speicherfragmentierung wird in [INF96c] auch als „Extent Interlea-

ving“ bezeichnet. Besteht ein DB-Space aus mehreren Chunks, so können die Extents einer Relation auch auf mehrere Chunks verteilt sein. Das folgende Bild zeigt ein Beispiel für solche „Interleaved Extents“.

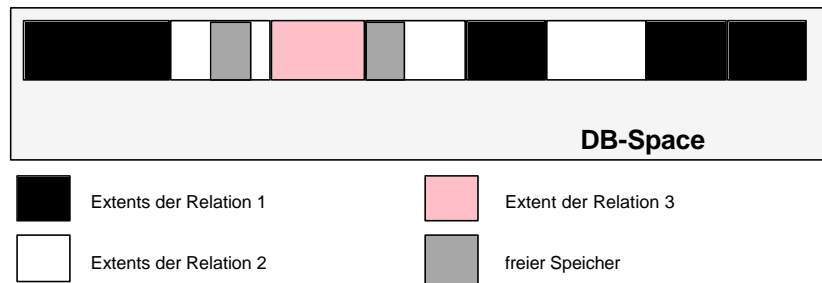


Abbildung 4: Extent Interleaving (nach [INF96c] S.3-29)

Das Problem kann hier nur mit einer explizit veranlaßten Datenbankreorganisation behoben werden. Der für eine derartige Reorganisation zu erwartende Systemaufwand, mit der entsprechenden Performance-Verschlechterung während der Reorganisation, läßt eine implizit veranlaßte Reorganisation zu einem Zeitpunkt, an dem das DBMS den Reorganisationsbedarf erkennt, nicht zu.

Liegt innerhalb des DB-Space genügend freier Speicher zusammenhängend vor, kann eine datenbanksystemintegrierte Reorganisation der betroffenen Relation vorgenommen werden. Liegen allerdings viele Relationen innerhalb des DB-Space, so daß nicht genügend zusammenhängender Speicher für das Umkopieren im DB-Space vorliegt, muß eine datenbanksystembasierte Reorganisation des gesamten DB-Space, d.h. aller Relationen in diesem DB-Space, vorgenommen werden.

5.4.2 Freigabe von Speicherplatz

Die Größe der Extents, die bei Bedarf an Speicherplatz reserviert werden, kann normalerweise bei Datenbank-Management-Systemen für jede Relation festgelegt werden. Unter Informix bleiben nach [INF96a] Extents einer Relation zugeordnet bis diese gelöscht oder reorganisiert wird. Ein Extent wird auch dann nicht automatisch freigegeben, wenn im Extent (z.B. nach Löschoptionen) keine Daten mehr enthalten sind. Um den Speicherplatz zurückzugewinnen, wird eine Reorganisation nötig. Die Zeitintervalle, in denen Reorganisationen erforderlich werden, sind von den entsprechenden Löschkaktivitäten und der Art der Datenverteilung abhängig.

Auch eine überlegte Wahl der Größe der Extents ist von Bedeutung, denn übermäßig große Extents, die nur langsam mit Daten gefüllt werden, sorgen dafür, daß Speicherplatz brach liegt.

Als Granulat für eine Reorganisation ist in diesem Fall die Relation sinnvoll, da innerhalb einer Datenbank meist Relationen existieren, die zum größten Teil nur abgefragt und kaum geändert werden. Damit müssen diese seltener reorganisiert werden als Relationen, auf denen sehr starke Änderungs- und vor allem Löschkaktivitäten durchgeführt werden. Hier bietet sich, wenn möglich, eine datenbanksystemintegrierte, sonst eine datenbanksystembasierte Reorganisation der Relation an.

5.4.3 Verringerung der Anzahl von Extents einer Relation

Durch die bei Informix implementierte Verwaltung von Extents einer Relation kann diese nicht beliebig viele Extents belegen. Die genaue Zahl möglicher Extents ist nach [INF96c] bei Informix u.a. von der Seitengröße, der Anzahl Attribute der Relation, der Anzahl Indexe usw. abhängig (als grober Richtwert dient nach verschiedenen Quellen die Zahl 200). Um den Verwaltungsaufwand für die Extents gering zu halten und eine zusammenhängende Datenspeicherung zu gewährleisten, sollte eine Relation aus möglichst wenigen Extents bestehen.

Werden in einem DB-Space sehr viele Relationen, deren Größe sich verändert, abgelegt, so ist es kaum möglich, mehrere Extents einer Relation fortlaufend zu reservieren. Damit wird der bei Informix implementierte Mechanismus, welcher bei der Reservierung eines neuen Extent physisch direkt nach einem der Relation zugeordneten Extent beide Extents zu einem einzelnen Extent zusammenfaßt, wirkungslos. Weiterhin kommt es auch nach der Reorganisation einzelner Relationen dazu, daß nur kleine freie Speicherbereiche zur Verfügung stehen. Sollte bei Bedarf kein Extent in voller Größe mehr reserviert werden können, weil nur noch sehr kleine freie Speicherbereiche vorhanden sind, so reserviert Informix den größten zur Verfügung stehenden Bereich. Eine automatische Zusammenführung freier Speicherbereiche zwischen den von Table Spaces belegten Bereichen erfolgt nicht. Das kann sehr schnell dazu führen, daß eine Relation die maximal mögliche Extent-Zahl erreicht. In diesem Fall ist eine Reorganisation des gesamten DB-Space nach der datenbank-systembasierten Methode nötig.

5.4.4 Beseitigung von Auslagerungszeigern

Auslagerungszeiger entstehen wenn Tupel nach einer Update-Operation und Längenvergrößerung nicht mehr in der ursprünglichen Seite abgelegt werden können. Die Tupel werden dann in andere Seiten verschoben. Damit die in den Indexen gespeicherte ROWID nicht geändert werden muß, wird in der Originalseite ein Zeiger auf den neuen Speicherort des Tupels gesetzt. Dadurch kann sich die bei Suchoperationen notwendige Anzahl Positionierungsoperationen und damit die Zugriffszeit erhöhen. Deshalb bieten viele Datenbank-Management-Systeme (z.B. Oracle) Parameter, mit denen gesteuert werden kann, wieviel Platz innerhalb einer Datenseite bei Einfügeoperationen für mögliche Verlängerungen von Tupeln freigehalten werden soll, an. Ziel ist es, damit die Häufigkeit des Verschiebens von Tupeln (Auslagerns) zu verringern. Informix kennt einen solchen Parameter allerdings nicht.

Das folgende Bild zeigt eine solche beschriebene Zeigerstruktur am Beispiel von Informix (siehe auch [INF96a] S. 43-43).

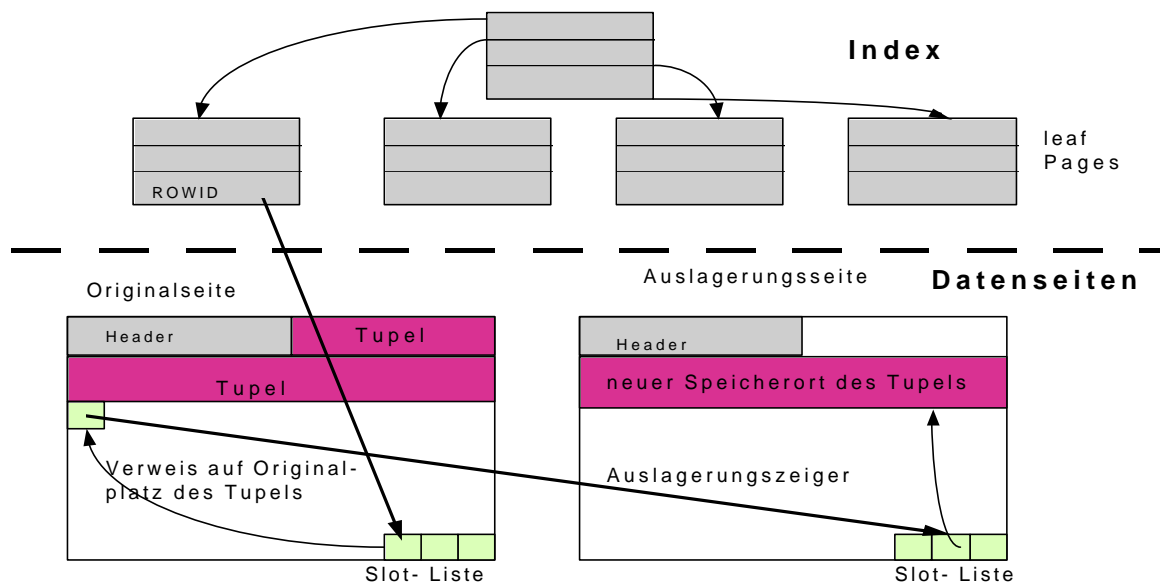


Abbildung 5: Zeigerstruktur, die durch die Auslagerung eines Tupels entstanden ist

Auslagerungszeiger lassen sich am einfachsten durch die Reorganisation der entsprechenden Relation nach der datenbanksystemintegrierten Methode beseitigen.

5.4.5 Wiederherstellen vorgegebener Sortierreihenfolgen

Unter „clustern“ eines Index wird bei vielen Systemen (z.B. Informix, MS-SQL-Server, IBM DATABASE 2) die physische Anordnung der Tupel der Relation entsprechend der Sortierreihenfolge des Index verstanden. Dies sollte nicht mit dem oben erwähnten, tabelleübergreifenden Cluster-Konzept von Oracle verwechselt werden. Die Sortierung wird allerdings nach [INF96b] bei anschließenden INSERT-Operationen oder bei Tupelauslagerungen nicht berücksichtigt. Damit geht die Sortierreihenfolge und damit die Cluster-Eigenschaft des Index sukzessive verloren.

Um eine erneute Sortierung vorzunehmen, muß der entsprechende Index neu „geclustert“ werden. Dazu bietet Informix die Verwendung der Anweisung `ALTER INDEX ... TO CLUSTER` an, die intern ein Umkopieren und Sortieren der Daten der Relation bewirkt. Eventuell existierende weitere Indexe sollten bei Relationen mit vielen Tupeln aus Performance-Gründen vor der Operation gelöscht und danach wieder angelegt werden. Weiterhin ist zu beachten, daß innerhalb des DB-Space genügend freier Speicher für die Kopieroperation vorhanden sein muß.

5.4.6 Löschen von "als gelöscht" gekennzeichneten Indexeinträgen

Während der Arbeit mit einer Datenbank kann es bei Informix zu vorübergehenden Degenerierungen der Indexbäume kommen. Solche Degenerierungen werden nach [INF96b] zum Teil vom DBMS erkannt und automatisch behoben. Wird ein Tupel aus einer Relation gelöscht, so wird der dazugehörige Indexeintrag nicht sofort beseitigt, sondern er wird zunächst lediglich als ungültig gekennzeichnet und gesperrt. Das tatsächliche Löschen der

Einträge wird nach Transaktionsende über eine im Shared Memory geführte Liste mit Löschanforderungen, die zyklisch abgearbeitet wird, asynchron realisiert.

Sollte, z.B. durch einem Systemausfall, der Inhalt des Shared Memory verloren gehen, ist es nötig, die zur Löschung vorgesehenen Indexeinträge zu lokalisieren und zu entfernen. Das ist nach [INF96b] mit der Anweisung `UPDATE STATISTICS`, die eigentlich zum Sammeln von Informationen für die Anfrageoptimierung gedacht ist, möglich. Beachtet werden sollte, daß solche Operationen zum Sammeln von Statistikinformationen meist mit einigem Systemaufwand verbunden sind.

5.4.7 Freigabe von Indexseiten

Informix nutzt zwei weitere Mechanismen zur automatischen Reorganisation der Indexe, welche vor allem der Freigabe nicht mehr benötigten Speicherplatzes dienen. Durch die Ordnung der Einträge innerhalb eines Index (B^+ -Baum) kann der Speicher, der beim Löschen eines Indexeintrags frei wird, nicht einfach für einen beliebigen anderen, neu zu erzeugenden Indexeintrag verwendet werden. Die Freigabe von ehemaligem Index-Speicher erfolgt seitenweise, wenn eine Indexseite keine Einträge mehr enthält.

Beim Löschen eines Indexeintrags überprüft Informix, ob es sich bei der entsprechenden Seite um einen „compression candidate“ handelt. Eine Indexseite (Indexknoten) wird zum „compression candidate“, wenn sie nur zwei oder weniger Indexeinträge enthält und nicht den Wurzelknoten des Indexbaums repräsentiert. Wird eine solche Seite erkannt, so wird versucht, die Indexeinträge der Seite in einen der im Baum benachbarten Indexknoten zu verlagern. Dieser Mechanismus wird als „Merging“ bezeichnet. Gelingt es damit, einen Indexknoten vollständig zu leeren, so wird die entsprechende Seite an die Freispeicherverwaltung zurückgegeben und kann beliebig weiter verwendet werden. Dies ist eine Implementierungsvariante des bekannten B^+ -Baum-Algorithmus.

Ist in keinem der Nachbarknoten genügend Speicher für die Aufnahme der Einträge eines „compression candidate“ vorhanden, so werden Indexeinträge des Nachbarn, der einen höheren Füllungsgrad aufweist, in den „compression candidate“ umgelagert, bis beide einen annähernd gleichen Füllungsgrad aufweisen (siehe auch [INF96a] S.43-56). Dadurch wird versucht, den Füllungsgrad der Indexseiten zu balancieren. Eine direkte Bedingung, wie man sie klassisch vom B - oder B^* -Baum kennt, nach der alle Knoten des Indexbaums, mit Ausnahme des Wurzelknotens, zu jeder Zeit mindestens zur Hälfte gefüllt sein müssen, existiert allerdings nicht. Bei Informix und anderen DBMS-Produkten wird hier also aus Performance-Gründen (bei der Index-Wartung) etwas von der „reinen Lehre“ abgewichen.

5.4.8 Steuerung des Füllungsgrades von Indexseiten

Zur Steuerung des Belegungsgrads von Indexseiten kennen viele Datenbank-Management-Systeme Parameter. Bei Informix wird dieser Parameter als „FILLFACTOR“ bezeichnet. Der Parameter gibt an, wie weit Indexseiten beim Anlegen eines Index mit `CREATE INDEX` gefüllt werden sollen. Ein hoher Wert bietet eine gute Ausnutzung des Speicherplatzes und sollte bei überwiegend lesender Nutzung der Daten verwendet werden. Bei häufigen Einfügeoperationen sollte ein niedrigerer Wert genutzt werden, um Splitting-Operationen zu vermeiden. Entspricht der tatsächliche Belegungsgrad nicht mehr dem gewünschten, so muß hier der Index gelöscht und neu aufgebaut werden, da der

FILLFACTOR nur beim Anlegen eines Index, nicht aber bei normalen Einfüge- und Löschoptionen berücksichtigt wird.

6 Zusammenfassung und Ausblick

In diesem Papier wurden die klassischen Gründe für Datenbankreorganisationen behandelt. Im Mittelpunkt standen dabei Ziele wie die Rückgewinnung von Speicherplatz, Defragmentierung von Datenbeständen und die Beseitigung von Degenerierungen der Zugriffspfade. Darüber hinaus wurde versucht, aufzuzeigen, daß eine Datenbankreorganisation auch mit anderen als den eben genannten Zielen sinnvoll sein kann. Dabei wurde auf die gezielte Verteilung von Schemaelementen, mit dem Ziel einer Lastverteilung und auf die Partitionierung von Relationen, mit dem Ziel einer Leistungssteigerung, näher eingegangen. Die Umstellung des Datenbankschemas (Schemaevolution) wurde nicht weiter betrachtet, da sie meist nicht direkt bzw. nicht allein im Verantwortungsbereich des Datenbankadministrators bzw. des DBMS liegt.

Zusätzlich wurden Anforderungen, die aus praktischer Sicht an eine Datenbankreorganisation zu stellen sind, angesprochen. Es wurden mögliche Methoden zur Reorganisationsdurchführung und Granulate, die für eine Datenbankreorganisation in Frage kommen, erläutert.

Weiterhin ist zu beachten, daß auch bei einer wohlüberlegten Organisation des Datenbestands, im Zusammenhang mit einer Reorganisation der Datenbank eine Neukonfiguration des Datenbank-Servers notwendig sein kann, um ein bestmögliches Ergebnis zu erreichen. So erzielt man mit einer Partitionierung von Relationen und Indexen oft nur dann ein verwertbares Ergebnis, wenn auch der verwaltende Datenbank-Server die Möglichkeit bekommt, mehrere Partitionen, die von einer Anfrage betroffen sind, parallel zu durchsuchen. Auch Parameter zur Steuerung eines vorausschauenden Lesens und Parameter, die das Verhalten der Pufferverwaltung (z.B. Rückschreiben geänderter Pufferinhalte) beeinflussen, steuern die I/O-Leistung des Systems. Nicht zuletzt spielt die Größe des Datenbank-Puffer-Pools eine entscheidende Rolle für die I/O-Leistung. Eine Reorganisation eines Datenbestands sollte deshalb als Gesamtheit aus Umorganisation des Datenbestands und Neuabstimmung des den Datenbestand verwaltenden Servers betrachtet werden.

Ein Schwerpunkt laufender wissenschaftlicher Untersuchungen ist das Erkennen von Reorganisationserfordernissen. Dabei ist es wichtig, auch den Nutzen, der mit einer Datenbankreorganisation erreicht werden kann, einschätzen zu können. Interessant sind dabei vor allem die Statistiken, welche Datenbank-Management-Systeme zu Zwecken der Anfrageoptimierung führen. Leider liegen diese Informationen üblicherweise nicht in einer einheitlichen Form vor. Auch kann die Menge an Informationen leicht dazu führen, daß wichtige Informationen übersehen werden. Die Schaffung von Werkzeugen, die den DBA bei der Entscheidungsfindung unterstützen, ist denkbar. Manche Systeme (z.B. IBM DATABASE 2) verfügen über solche Werkzeuge, die zumindest die Erkennung bestimmter Reorganisationserfordernisse erleichtern und Empfehlungen abgeben, wenn eine Datenbankreorganisation notwendig ist. Allerdings wird dabei oft der Nutzen, der durch die Reorganisation erreicht wird, nicht angegeben, und die DBMS-seitigen Empfehlungen sind nicht immer einfach nachvollziehbar. Es sollte aber möglich sein, diesen Nutzen den durch die Reorganisation verursachten Kosten gegenüberzustellen.

Dabei ist allerdings zu beachten, daß der für den Anwender meßbare Nutzen, besonders erzielbare Performance-Verbesserungen, nicht nur system- sondern auch workload-abhängig sind und deshalb nicht pauschal berechnet werden können. Auch die bei einer Reorganisation zur Rückgewinnung von reserviertem Speicher erzielbaren Effekte sind abhängig vom physischen Design der Datenbank. Allerdings wäre die Möglichkeit, näherungsweise Aussagen zum zu erwartenden Nutzen treffen zu können, bereits ein großer Fortschritt.

Literaturverzeichnis

- [INF96a] *Informix-OnLine Dynamic Server V7.12, Administrator's Guide, Volume 1, Volume 2.* Informix Software Inc., 1996
- [INF96b] *Informix-OnLine Dynamic Server V7.22, Administrator's Guide for Windows NT.* Informix Software Inc., 1996
- [INF96c] *Informix-OnLine Dynamic Server V7.1, Performance Guide.* Informix Software Inc., 1996
- [INF96d] *Informix-OnLine Dynamic Server, Guide to Feature Enhancements for Windows NT, Version 7.22.* Informix Software Inc., 1996
- [INF96e] *Informix Guide to SQL: Syntax for Windows NT Version 7.12.* Informix Software Inc., 1996
- [Lab96] St. Labitzke. *Datenbank-Reorganisation in DB2 für AIX Version 2 und ORACLE Version 7: Beschreibung, Analyse, Bewertung, Anwendung.* Diplomarbeit, Institut für Informatik, Friedrich-Schiller-Universität Jena, Juli 1996
- [LS93] P. C. Lockemann, J. W. Schmidt. *Datenbankhandbuch.* Springer-Verlag, Berlin, Heidelberg, New York, London, Paris, Tokyo, 1993
- [MM87] V. M. Markowitz, J. A. Makowsky. Incremental Reorganization of Relational Databases. In *Proceedings of the 13th VLDB Conference*, Brighton, 1987
- [MHR96] H. Mucksch, J. Holthuis, M. Reiser. Das Data-Warehouse-Konzept - ein Überblick. In *Wirtschaftsinformatik Heft 4/1996*, Verlag Vieweg, Wiesbaden, Juli 1996
- [Nob95] N. Noble. Techniques for fast Database Reorganisation. In *European Oracle User Group (EOUG) Conference*, Florence, April 1995
- [ORA93] *Oracle7 Server Concepts Manual.* Oracle Corporation, 1993
- [Nuß97] R. Nußdorfer. Reorganisationsfreiheit in Datenbanksystemen: Bewertung ist anwendungsabhängig. In *Datenbank Fokus, Januar 1997*, IT Verlag für innovative Technologien GmbH, Höhenkirchen, 1997
- [SAG97] *ADABAS D Schattenspeicherkonzept - Reorganisationsfreie Datenhaltung ohne I/O-Engpässe.* SQL GmbH - Software AG, 1997
- [Sha95] C. A. Shallahamer. Avoiding a Database Reorganization. In *European Oracle User Group (EOUG) Conference*, Florence, April 1995
- [TL91] R. Trautloft, U. Lindner. *Datenbanken-Entwurf und Anwendung.* Verlag Technik GmbH Berlin, 1991