

Quantifizierung des zu erwartenden Nutzens von Datenbankreorganisationen

Stefan Dorendorf

Friedrich-Schiller-Universität Jena

Institut für Informatik

Lehrstuhl für Datenbanken und Informationssysteme

Ernst-Abbe-Platz 1-4

07743 Jena

Stefan.Dorendorf@informatik.uni-jena.de

Kurzfassung

Stetig wachsende Datenmengen und hohe Verfügbarkeitsanforderungen an Datenbanken führen dazu, dass Wartungsarbeiten (wie z.B. Datenbankreorganisationen) oft nicht mehr (oder nicht mehr vollständig) quasi offline in zur Verfügung stehenden Zeitfenstern durchgeführt werden können, bzw. dass die Kosten oder die Behinderungen des normalen Datenbankbetriebs bei online durchgeführten Wartungsarbeiten nicht mehr vernachlässigt werden können. Deshalb sollte vorab bestimmt werden, ob der zu erwartende Nutzen den mit der Reorganisation verbundenen Aufwand rechtfertigt. Es ist deshalb wichtig, die Datenbankobjekte zu lokalisieren und einzugrenzen, bei denen ein hoher Wartungsbedarf besteht. Und es ist wünschenswert, zunächst den durch Wartungsmaßnahmen erreichbaren Nutzen quantifizieren zu können. Der ist abhängig von der Workload, also von den gegen die Datenbankobjekte gerichteten Anweisungen. In diesem Beitrag wird eine Methode vorgestellt, die es ermöglicht, die Auswirkungen einer Datenbankreorganisation auf den zur Workload-Abarbeitung notwendigen I/O-Aufwand abzuschätzen, der einen dominierenden Anteil am Gesamtaufwand ausmacht und sich mit einer Datenbankreorganisation u.U. auch wesentlich beeinflussen lässt.

Inhaltsverzeichnis

1	Einführung.....	3
2	Reorganisationsnutzenbestimmung im Überblick.....	4
3	Abschätzung des Nutzens von Datenbankreorganisationen	6
3.1	Allgemeine Bemerkungen	6
3.2	Kostenfunktionen.....	7
3.3	Bestimmung des I/O-Mehraufwands durch Degenerierungen	10
3.4	Zusammenführung von Mehraufwandswerten und Aufwandsanteilen	11
4	Evaluierung an einem Beispiel	14
5	Zusammenfassung und Ausblick.....	18

1 Einführung

Hohe Verfügbarkeitsanforderungen an datenbankgestützte Softwarelösungen führen zu immer kürzeren Zeitfenstern für die Wartung der Datenbanken. Im Werk des Halbleiterherstellers AMD in Dresden steht lt. [Ri03] z.B. jährlich lediglich eine geplante Unterbrechungszeit von einer Stunde für eventuelle offline durchzuführende Wartungsarbeiten zur Verfügung. Aus diesem Grund müssen Wartungsarbeiten häufig parallel zum normalen Betrieb durchgeführt werden. Einschränkungen des Datenbankbetriebs müssen dabei auf ein Minimum begrenzt werden. Hier ist eine wohlüberlegte Planung der durchzuführenden Arbeiten zwingend erforderlich. Wartungsarbeiten, die einen entsprechenden Nutzen erwarten lassen, sind durchzuführen, unnötige oder noch nicht nötige Arbeiten müssen vermieden werden. Hier ergibt sich die Notwendigkeit, den zu erwartenden Nutzen vorab einschätzen zu können, um ihn den zu erwartenden Kosten, die auch sehr stark von den eingesetzten Methoden abhängig sind und hier vorerst nicht weiter betrachtet werden sollen, gegenüberzustellen.

Derzeit verbreitete Werkzeuge zur Unterstützung von Reorganisationsentscheidungen, wie z.B. REORGCHK für DB2 [IBM02], Oracle Tablespace Map [ORA02], BMC Space Expert [BR02], Embarcadero Space Analyst [Sch03] usw., arbeiten rein kennzahlenbasiert, ohne Berücksichtigung der gegen die Datenbank gerichteten Workload. Typischerweise werden hier aus statistischen Daten über die Datenbankobjekte Degenerierungsgrade berechnet, wie z.B. der prozentuale Anteil ausgelagerter Sätze oder der Freiplatzanteil. Überschreiten diese Zahlen bestimmte Grenzwerte, die entweder fest vorgegeben sind oder von DBA festgelegt werden können, wird vom Werkzeug eine Reorganisation des betrachteten Datenbankobjekts empfohlen. Angaben, wie hoch der von der Reorganisation zu erwartenden Nutzen in etwa sein wird, werden nicht getroffen. Dieser müsste durch Messungen vor und nach der Reorganisation bestimmt werden.

Durch die physische Reorganisation von Datenbankobjekten können insbesondere der zur Abarbeitung der Datenbank-Workload notwendige I/O-Aufwand und der zur Datenablage benötigte Speicherplatz reduziert werden. Dies stellt den vorrangigen Nutzen der Datenbankreorganisation dar.

Gegenstand dieses Beitrag ist eine Methode, die es ermöglicht, eine Quantifizierung des Nutzens einer Reorganisation von Datenbankobjekten vorzunehmen. Die Grundlage bilden Informationen über den aktuellen Zustand der physischen Speicherungsstrukturen von Datenbankobjekten und ein Protokoll der innerhalb eines repräsentativen Zeitraums gegen die Datenbank gerichteten DML(SQL)-Anweisungen (Workload-Protokoll). Als Ergebnis steht eine Abschätzung, um welchen Prozentsatz die zur Abarbeitung der Workload benötigte Anzahl Blockzugriffe durch eine Reorganisation verringert werden kann. Diese Information ist hilfreich, um eine Reorganisationsentscheidung zu fällen und Reorganisationskandidaten (Tabellen, Indexe, Table Spaces, Cluster,...) zu lokalisieren.

Kapitel 2 bietet zunächst einen Überblick über die vorgeschlagene Methode, bevor in **Kapitel 3** die einzelnen Schritte näher erläutert werden. In **Kapitel 4** werden Ergebnisse präsentiert, die bei einer Überprüfung in einem Beispielsystem ermittelt wurden. Die Ergebnisse des Beitrags werden in **Kapitel 5** kurz zusammengefasst und ein Ausblick auf weitere Arbeiten gegeben.

2 Reorganisationsnutzenbestimmung im Überblick

Degenerierungen der zur Speicherung von Datenbankobjekten (z.B. Tabellen, Indexe) verwendeten physischen Strukturen führen zu einer Erhöhung des anfallenden I/O-Aufwands und teils auch zur Speicherplatzverschwendung, was allerdings oft als weniger kritisch angesehen wird. Solche Degenerierungen sind z.B. eingestreuter Freiplatz, ausgelagerte Sätze, nicht eingehaltene interne Sortierreihenfolgen, nicht aktuell gewartete Indexstrukturen etc. [DK00]. Sie werden dadurch verursacht, dass die physischen Speicherungsstrukturen vom DBMS aus Performance-Gründen und um Behinderungen der laufenden Datenbankverarbeitung zu vermeiden, nicht immer vollständig und sofort gepflegt werden.

Um den Nutzen einer Datenbankreorganisation zu bestimmen, wird workload-bezogen der nach ihr anfallende (zu erwartende, reduzierte) I/O-Aufwand ermittelt und ins Verhältnis zum Aufwand vor der Reorganisation gesetzt. Die Nutzenermittlung erfolgt dabei in mehreren Schritten, die nachfolgend im Überblick kurz dargestellt werden (siehe auch Abbildung 1). Detaillierte Betrachtungen zu wichtigen Teilschritten werden dann in Kapitel 3 angestellt.

So wie Reorganisationen von Datenbanken mehr oder weniger regelmäßig erfolgen müssen, muss auch die Bewertung des zu erwartenden Nutzens in entsprechenden Intervallen erfolgen. Die Grundidee besteht darin, ausgehend von einem Workload-Protokoll und statistischen Kenngrößen, die Informationen über den Zustand der physischen Speicherungsstrukturen liefern, den Nutzen einer Datenbankreorganisation quantifizieren zu können. Möglich wären auch Verfahren, die physische Speicherungsstrukturen und deren Zustand nach einer Reorganisation simulieren¹. Solche Verfahren sind allerdings nur aufwendig zu realisieren, insbesondere bei Produktivsystemen, da die Simulation keine negativen Auswirkungen (z.B. ungünstige Ausführungspläne für Anfragen) auf den parallel zur Analyse laufenden Produktivbetrieb haben darf. Bei der Verwendung von Testsystemen ist es aber i.d.R. nur schwer möglich, die gleiche Systemumgebung sowie gleiche Degenerierungsgrade der Datenbankobjekte wie bei den Produktivsystemen zu erzeugen oder zu simulieren, um sie in die Nutzensabschätzung einfließen zu lassen.

Da der anfallende I/O-Aufwand durch die gegen die Datenbank gerichtete Workload verursacht wird, ist es notwendig, diese bei der Nutzenermittlung zu berücksichtigen. Dazu werden zunächst in einem als repräsentativ anzusehenden Zeitraum die gegen die Datenbank gerichteten (SQL-)Anweisungen und deren Ausführungshäufigkeiten protokolliert (I). Dieses Workload-Protokoll bildet die Grundlage für die weiteren Schritte und kann auch mehrfach genutzt werden, solange keine signifikanten Veränderungen der Workload auftreten.

¹ Bei Werkzeugen, die den DBA bei der Indexierung von Datenbeständen unterstützen sollen und die z.B. für den Microsoft SQL Server oder DB2 verfügbar sind, wird die Simulation von Indexen angewendet [CN97,CN98]. In diesem Zusammenhang wurden auch Methoden zur Begrenzung der Größe von Workload-Beschreibungen (Workload-Protokollen) entwickelt [CGN02]. In [SD03] wird ein solches Simulationsverfahren zur Unterstützung der Suche nach einer geeigneten physischen Repräsentation für komplexe Objekte in ORDBMS verwendet.

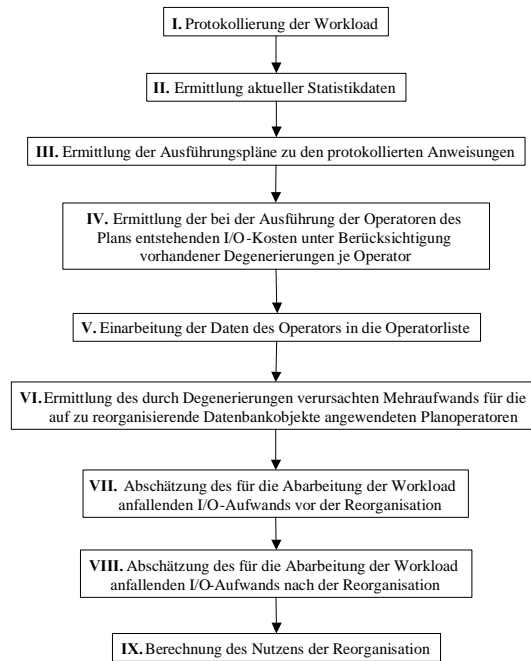


Abbildung 1: Ablauf der Ermittlung des Nutzens einer Datenbankreorganisation

Im Rahmen der Nutzenermittlung ist also der aktuelle („degenerierte“) Zustand der physischen Speicherungsstrukturen zu berücksichtigen. Hier bieten sich die im Datenbankkatalog vorhandenen Statistikdaten an, die normalerweise zur kostenbasierten Anfrageoptimierung verwendet werden. Für unsere Zwecke sind diese Daten weitgehend ausreichend. Lediglich wenn genaue Analysen für z.B. feine Granulate (Blöcke oder Extents) durchgeführt werden sollen, reichen die im Katalog geführten Informationen oft nicht aus, da für solch feine Granulate üblicherweise keine Statistiken ermittelt werden. Hier steht allerdings auch die Frage nach dem Nutzen solcher Analysen, da eine Reorganisation so kleiner Einheiten mit den zur Verfügung stehenden Werkzeugen meist nicht möglich ist. Das momentan üblicherweise feinste Granulat für Datenbankreorganisationen stellen Partitionen [Now01] von Tabellen oder Indexen dar. Weiterhin wird derzeit meist von einer Gleichverteilung der innerhalb der Struktur, für die die Statistikdaten gesammelt wurden, auftretenden Degenerierungen ausgegangen. Die Berücksichtigung von „Schieflagen“ bei der Verteilung von Degenerierungen würde Erweiterungen der Statistiken erfordern und zu einer deutlichen Erhöhung des Aufwands für die Statistikerstellung führen. Hier müsste abgewogen werden, ob der zu erwartende Genauigkeitserfolg diesen Aufwand rechtfertigt. Wegen des damit verbundenen Aufwands werden auch vorhandene Statistikdaten teilweise von DBMS im laufenden Betrieb nicht gepflegt. Deshalb muss i.d.R. zunächst noch eine Aktualisierung der Statistikdaten erfolgen (II).

Die Abarbeitung der in der Workload enthaltenen Anweisungen erfolgt mittels einfacher Grundoperationen (Planoperatoren [HR01]). Diese Planoperatoren stoßen die notwendigen I/O-Operationen an. Es ist die Aufgabe des Anfrageoptimierers, unter Berücksichtigung vorhandener Statistikdaten zu gegebenen SQL-Anweisungen Folgen von Planoperatoren (Ausführungspläne) zu finden (III), mit denen die Anweisungen kostengünstig realisiert werden können.

Anhand des Ausführungsplans kann die Berechnung der durch die Anwendung der jeweiligen Planoperatoren verursachten Kosten erfolgen (IV). Insbesondere die anfallenden I/O-Kosten

können durch Datenbankreorganisationen u.U. wesentlich beeinflusst werden und stehen daher im Mittelpunkt unserer Betrachtungen. Dazu werden die aktuellen Statistikdaten und ein geeignetes und nachvollziehbares Kostenmodell benötigt, das durchgängig auch die in den Speicherungsstrukturen vorhandenen Degenerierungen berücksichtigt. Weiterhin muss die Anzahl der Ausführungen des Planoperators berücksichtigt werden.

Im Rahmen der Ausführung verschiedener DML-Anweisungen kommen die unterschiedlichen Planoperatoren je Datenbankobjekt in mehreren Ausführungsplänen vor. Zur Begrenzung des Aufwands in den Folgeschritten werden Planoperatoren gleichen Typs, die auf die gleichen Datenbankobjekte angewendet werden, kostenmäßig zusammengefasst. Damit wird eine Liste („Gesamtplan“) aufgebaut (V), in der die einzelnen Operatoren, die auf die jeweiligen Datenbankobjekte angewendet werden, jeweils nur einmal vorkommen.

Unter Nutzung der bereits ermittelten aktuellen Statistikdaten kann für die zur Reorganisation vorgesehenen Datenbankobjekte errechnet werden, wie hoch (prozentual) der durch vorhandene Degenerierungen für jeden auf das jeweilige Datenbankobjekt angewendeten Planoperator verursachte Mehraufwand in etwa ist (VI).

Durch Summierung der Kostenwerte für die im Gesamtplan vorkommenden Planoperatoren kann nun die Anzahl vor einer eventuellen Reorganisation für die Abarbeitung der Workload anfallenden Blockzugriffe berechnet werden (VII). Dieser Wert wird dann im übernächsten Schritt (IX) den auf Basis des gleichen, Kostenmodells errechneten Kosten nach der Reorganisation gegenübergestellt, um die relative Einsparung an Blockzugriffen (den Nutzen der Datenbankreorganisation) zu ermitteln.

Zur Abschätzung der I/O-Kosten *nach* einer Reorganisation² müssen vor der Summierung noch die Kostenwerte der Planoperatoren, die auf zu reorganisierende Datenbankobjekte angewendet werden, um den in Schritt VI errechneten Mehraufwand verringert werden (VIII).

Zusammenfassend gesehen stellt sich die Nutzenanalyse von Datenbankreorganisationen als Funktion dar, die vom Zustand der physischen Speicherungsstrukturen der Datenbankobjekte und der auf diese Datenbankobjekte angewendeten Workload abhängig ist.

3 Abschätzung des Nutzens von Datenbankreorganisationen

3.1 Allgemeine Bemerkungen

Basis für die Berücksichtigung der Workload ist ein Protokoll der gegen die Datenbank gerichteten (SQL-)Anweisungen (I. in Abbildung 1). Entweder das jeweils verwendete Datenbank-Management-System (DBMS) bietet Werkzeuge für eine solche Protokollierung an (wie z.B. der MS SQL Profiler) oder es müssen vorhandene Schnittstellen des Query Processors genutzt werden. Ist auch dies nicht möglich, so muss die Protokollierungsfunktionalität über eine vorgeschaltete Komponente realisiert werden. Zur Begrenzung der Größe des Protokolls bietet es sich an, die ausgeführten SQL-Anweisungen nur einmal im Protokoll zu speichern und zusätzlich einen Zähler mitzuführen, der die Häufigkeit der Ausführungen angibt. Werden existierende Protokollierungswerkzeuge genutzt, muss das erzeugte Protokoll vor der Weiterverarbeitung evtl. aufbereitet werden. Soll nicht die gesamte protokollierte Workload berücksichtigt werden, so ist das Protokoll ebenfalls noch entsprechend aufzubereiten. Dies ist z.B. notwendig, wenn nur bestimmte

² Dabei wird angenommen, dass die Ausführungspläne vor und nach der Reorganisation gleich sind.

Datenbankobjekte oder bestimmte performance-kritische Anweisungen einbezogen werden sollen.

Die einzelnen SQL-Anweisungen des Anweisungsprotokolls werden zur Erstellung des Ausführungsplans an den Anfrageoptimierer übergeben. Dieser liefert eine Liste von Planoperatoren, mit denen die entsprechenden Anweisungen in der konkreten Anwendungsumgebung realisiert werden. Dadurch erfolgt implizit auch eine Berücksichtigung der Systemumgebung.

Die anschließende Ermittlung der Anzahl der für die Ausführung der Operation notwendigen Blockzugriffe ist vom jeweiligen Planoperator abhängig. Ähnliche Kostenermittlungen werden auch im Rahmen der kostenbasierten Anfrageoptimierung angestellt. Allerdings ist die Verwendung der Kostenschätzungen existierender Anfrageoptimierer hier derzeit problematisch, da diese häufig kaum „extern“ nachvollziehbar sind. Wie neben den I/O-Kosten auch z.B. CPU-Kosten berücksichtigt und welche Annahmen durch die Auswertung von Konfigurationsparametern (z.B. für Puffergrößen, vorausschauendes Lesen usw.) getroffen werden, wird durch die Hersteller nicht offen gelegt. Weiterhin ist i.d.R. nicht nachvollziehbar, inwieweit diese Kostenmodelle vorhandene Degenerierungen berücksichtigen, selbst wenn diese aus den Statistiken heraus erkennbar sind. Um die Nachvollziehbarkeit von Berechnungs- und Messergebnissen bei der Überprüfung anhand von Beispielimplementierungen sicherzustellen, wird hier ein eigenes I/O-Kostenmodell verwendet, das vorhandene Degenerierungen berücksichtigt. Dieses kann bei Bedarf ohne größere Probleme erweitert und z.B. in existierende Kostenmodelle (z.B. die für die Anfrageoptimierung) eingebaut werden. Wichtig ist dann nur, dass die Berücksichtigung von Degenerierungen der Speicherungsstrukturen durchgängig und nachvollziehbar für alle I/O-relevanten Planoperatoren erfolgt.

Als hinderlich erwiesen sich bei der Kostenermittlung die fehlende Standardisierung der Teile der Datenbankkataloge, die die entsprechenden Statistikdaten über Datenbankobjekte enthalten sowie Unterschiede im Implementierungsumfang der einzelnen Planoperatoren. In [Dor00] wird ein Ansatz für ein vereinheitlichtes Speicher- und Verhaltensmodell, das „eInformationsschema“, als Basis für Reorganisations-bedarfsanalysen beschrieben. Dabei steht das „e“ schlicht für „erweitert“, um Verwechslungen mit dem Informationsschema der SQL-Norm zu vermeiden. In [Hel01] wird beispielhaft am DBMS-Produkt Oracle die Transformation der Statistikdaten aus dem konkreten Katalog eines DBMS in das eInformationsschema dargestellt. Weitere Erläuterungen zum eInformationsschema würden den Rahmen dieses Beitrags sprengen. Die hier präsentierten Formeln zur Aufwandsabschätzung basieren auf diesem Schema und die verwendeten Formelzeichen werden an entsprechender Stelle erläutert.

3.2 Kostenfunktionen

An drei Planoperatoren soll beispielhaft die Abschätzung anfallender Blockzugriffe unter Berücksichtigung vorhandener Degenerierungen gezeigt werden. Dabei werden zur Vereinfachung im Rahmen dieses Beitrags nur eindeutige (unique) Indexe betrachtet. Die Basis für die Kostenschätzungen bilden zunächst bekannte Kostenfunktionen [SA⁺79,Dun02,Mak03].

Wenn ein Satz nach einer verlängernden Änderungsoperation nicht mehr im ursprünglichen Datenblock gespeichert werden kann, kommt es bei den meisten DBMS-Implementierungen zur Auslagerung in einen anderen Datenblock, um das Aufspalten des Satzes zu verhindern. Zur Vermeidung eines u.U. aufwendigen Nachpflegens von Indexen wird am ursprünglichen Speicherort ein Zeiger (Stellvertreter) auf den neuen Speicherort abgelegt. Soll jetzt über

einen Index auf den Satz zugegriffen werden, so ist zusätzlich zum normalen Aufwand noch ein Blockzugriff für die Verfolgung des Auslagerungszeigers notwendig (Abbildungen 2 und 3).

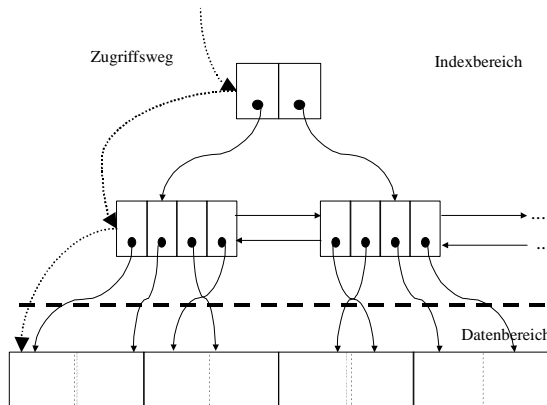


Abbildung 2: Blockzugriffe ohne Auslagerung

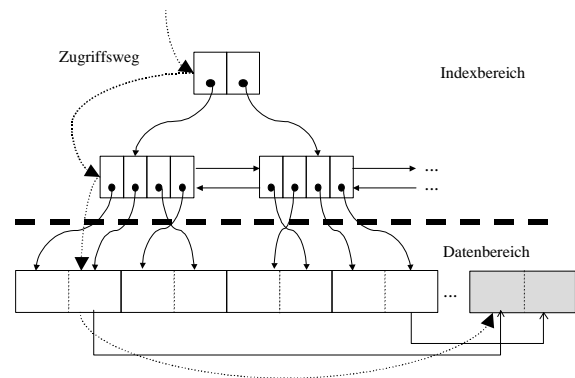


Abbildung 3: Blockzugriffe mit Auslagerung

Für den Zugriff auf die zu einem Suchschlüssel gehörenden Tupel über einen Index (**Index Lookup**) kann die anfallende Anzahl Blockzugriffe (C) mit der Formel

$$C = I_{LEV} + 1$$

abgeschätzt werden, wenn aus den Statistikdaten die Anzahl Ebenen des Index (I_{LEV} - im Beispiel in den Abbildungen 2 und 3 ist diese jeweils gleich 2) ermittelt werden können. Soll berücksichtigt werden, dass auch Tupelauslagerungen (also die hier betrachteten Degenerierungen) vorkommen können, so muss die Formel um die Gesamtzahl der Tupel der Tabelle (T) und die Anzahl ausgelagerter Tupel (A) wie folgt erweitert werden.

$$C = I_{LEV} + 1 + \frac{A}{T}$$

Viele der üblichen Kostenmodelle berücksichtigen Pufferungsgrade (Buffer Hit Ratio) von Daten- und Indexblöcken entweder gar nicht oder mittels grober Annahmen. Dies ist oft darauf zurückzuführen, dass insbesondere detaillierte datenbankobjektbezogene Statistiken (etwa tabellen- oder indexbezogen) über Pufferungsgrade von DBMS-Produkten derzeit i.d.R. nicht direkt bzw. nicht nach „außen“ angeboten werden, weil sie übergreifend über mehrere Schichten der DBMS-Architektur gesammelt werden müssten. Eine entsprechende Erweiterung der Kostenfunktionen um datenbankobjektbezogene Pufferungsgrade, also die *Wahrscheinlichkeit, dass sich ein angeforderter Block im Puffer-Pool des DBMS befindet*, würde zu folgender Formel führen.

$$C = \underbrace{I_{LEV} * (1 - I_{BR})}_{\text{Indexteil}} + \underbrace{\left(1 + \frac{A}{T}\right) * (1 - T_{BR})}_{\text{Datenteil}}$$

Dabei stehen I_{BR} und T_{BR} für die Pufferungsgrade von Index- (I_{BR}) und Datenblöcken (T_{BR})³. Hier fließt implizit die gängige Annahme ein, dass Zugriffe auf Datenblöcke im Puffer, im Vergleich zu Zugriffen auf Sekundär-speichermedien, kostenmäßig vernachlässigt werden können.

Als zweiter Beispieloperator soll hier ein **Index Scan** betrachtet werden, bei dem die einzelnen Tupel einer Tabelle nacheinander gemäß dem Ordnungskriterium der Tabelle gelesen werden. Die Anzahl notwendiger Blockzugriffe ergibt sich hier aus der Summe der Anzahl Indexebenen (ohne Blattebene), der Anzahl Blätter des Indexbaums, die entlang der Verkettung durchlaufen werden müssen (I_{LEAF} bzw. ein Teil davon) und der Anzahl Tupel im entsprechenden Bereich (T), da für jeden Tupelzugriff ein Zugriff auf den jeweiligen Datenblock erfolgen muss.

$$C = \left[\underbrace{(I_{LEV} - 1)}_{\text{innereEbenen}} + \underbrace{\max(S * I_{LEAF}, 1)}_{\text{Blattebene}} + \underbrace{T * S}_{\text{Datenteil}} \right]$$

Hier gibt S (Selektivität) den Anteil des Suchbereichs an der Gesamtdatenmenge an, über den der Index Scan ausgeführt wird. Dieser kann bei einer Intervallanfrage (Range Query) beispielsweise mit der folgenden Formel näherungsweise berechnet werden

$$S = \frac{1 + SK_{MAX} - SK_{MIN}}{1 + I_{MAX} - I_{MIN}},$$

wenn die folgenden Parameter bekannt sind.

- SK_{MAX} entspricht der oberen Intervallgrenze in der Selektionsbedingung
- SK_{MIN} entspricht der unteren Intervallgrenze in der Selektionsbedingung
- I_{MAX} entspricht dem größten Schlüsselwert des Index
- I_{MIN} entspricht dem kleinsten Schlüsselwert des Index

Voraussetzung für dieses einfache Verfahren ist in etwa eine Gleichverteilung der Schlüsselwerte. DBMS-Produkte benutzen ansonsten oft Histogramme, um genauere Informationen über Werteverteilungen festzuhalten.

Durch die Berücksichtigung der ausgelagerten Tupel ergibt sich folgende Formel.

$$C = \lceil (I_{LEV} - 1) + \max(S * I_{LEAF}, 1) + (T + A) * S \rceil$$

Eine Erweiterung um Pufferungsgrade führt dann zu folgender Formel.

$$C = \left[\underbrace{(I_{LEV} - 1) * (1 - I_{BR})}_{\text{innereEbenen}} + \underbrace{\max(S * I_{LEAF} * (1 - I_{BR}), 1 * (1 - I_{BR}))}_{\text{Blattebene}} + \underbrace{(T + A) * (1 - T_{BR}) * S}_{\text{Datenteil}} \right]$$

Als Wert für die bei einer **sequentiellen Suche** (Table Scan) anfallende Anzahl Blockzugriffe wird üblicherweise die Anzahl von der Tabelle aktuell belegter Blöcke

³ Im Rahmen laufender Arbeiten wird untersucht, wie bei verfügbaren DBMS-Produkten datenbankobjektbezogenen Pufferungsgrade ermittelt, und inwieweit noch detailliertere Informationen über die Pufferung (z.B. datenbankobjekt-, indexebenen- und operationsbezogene Pufferungsgrade) gewonnen und sinnvoll angewendet werden können.

($P_{AKTUELL}$) verwendet, unabhängig von deren Füllungsgrad. Hier ist auch in gängigen Kostenmodellen der durch evtl. vorhandenen eingestreuten Freiplatz entstehende Mehraufwand bereits enthalten. Die Kosten für einen Table Scan können wie folgt geschätzt werden.

$$C = P_{AKTUELL} * (1 - T_{BR})$$

3.3 Bestimmung des I/O-Mehraufwands durch Degenerierungen

Über die bisher vorgestellten Kostenfunktionen werden die aktuell (vor einer Datenbankreorganisation) anfallenden Kosten zur Workload-Abarbeitung berechnet. Um die Kosten nach der Reorganisation abschätzen zu können, wird nun für die zu reorganisierenden Datenbankobjekte der durch aktuell vorhandene Degenerierungen verursachte Mehraufwand bestimmt. Werden die aktuellen Kostenschätzungen für die gegen zu reorganisierende Datenbankobjekte gerichteten Planoperatoren um diesen Mehraufwand verringert, so ergeben sich die nach einer Reorganisation zu erwartenden Kosten. Die Veränderung des I/O-Aufwands ist abhängig von den in den physischen Speicherstrukturen vorhandenen Degenerierungen und den Planoperatoren, die auf diese Strukturen angewendet werden. In Datenblöcken vorhandener Freispeicher hat z.B. vor allem bei der Anwendung sequentieller Suchoperationen eine Erhöhung der Anzahl notwendiger Blockzugriffe zur Folge. Die Zahl der Blockzugriffe bei Suchoperationen über Indexe bleibt i.d.R. unverändert. Zur Ermittlung des durch vorhandene Degenerierungen zu erwartenden Mehraufwands ist es also notwendig, Planoperatoren und verschiedene Typen und Umfänge von Degenerierungen in ihrem Zusammenhang zu betrachten. Satzauslagerungen sowie evtl. vorhandene Überlaufblöcke haben überwiegend Auswirkungen auf Zugriffsoperationen über Indexe (z.B. Index Lookup bzw. Index Scan). Ziel ist es zunächst, den bei der Anwendung eines in der Workload enthaltenen Planoperators anfallenden prozentualen Mehraufwand an Blockzugriffen zu bestimmen, unabhängig vom Anteil, den der Planoperator am für die gesamte Workload anfallenden I/O-Aufwand ausmacht.

Wie diese Mehraufwandsbestimmung erfolgen kann, soll hier beispielhaft für Tupelauslagerungen und eingestreuten Freiplatz anhand der drei vorher betrachteten Planoperatoren gezeigt werden. Dieses Gebiet wird in derzeit laufenden Arbeiten noch weiter untersucht.

Bei der Ausführung von **Index Lookups** verursachen Satzauslagerungen einen Mehraufwand von jeweils einem zusätzlichen Blockzugriff. Dieser kann bei den üblichen B-Baum-Indexen unter Berücksichtigung der Anzahl der Ebenen des Index (I_{LEV}), der Anzahl ausgelagerter Sätze (A) und der Gesamtzahl der Tupel/Sätze (T) wie folgt bestimmt werden.

$$M = \frac{A}{T * (I_{LEV} + 1)} * 100$$

Durch die Berücksichtigung der unterschiedlichen Pufferungsgrade von Index- (I_{BR}) und Datenblöcken (T_{BR}) ergibt sich folgende Formel.

$$M = \frac{A * (1 - T_{BR})}{T * \left(\underbrace{I_{LEV} * (1 - I_{BR})}_{\text{Indexteil}} + \underbrace{(1 - T_{BR})}_{\text{Datenteil}} \right)} * 100$$

Aber auch bei **Index Scans** verursacht die Verfolgung der Auslagerungszeiger eine Erhöhung der notwendigen Anzahl Blockzugriffe, die wie folgt bestimmt werden kann.

$$M = \frac{A}{I_{LEV-1} + I_{LEAF} + T} * 100$$

Dabei muss die Anzahl Blöcke auf der Blattebene (I_{LEAF}) berücksichtigt werden. Die Selektivität kann durch die getroffene Annahme, dass Degenerierungen gleichverteilt auftreten, unberücksichtigt bleiben. Pufferungsgrade können hier wie folgt berücksichtigt werden.

$$M = \frac{A * (1 - T_{BR})}{\underbrace{(I_{LEV-1} + I_{LEAF}) * (1 - I_{BR})}_{\text{Indexteil}} + \underbrace{T * (1 - T_{BR})}_{\text{Datenteil}}} * 100$$

Die **sequentielle Suche** wird insbesondere durch in Datenblöcke eingestreuten Freiplatz beeinflusst. Durch die damit verbundene Erhöhung der zur Datenspeicherung benötigten Anzahl Datenblöcke kommt es zu einer Erhöhung der Zahl der Blockzugriffe, der wie folgt bestimmt werden kann.

$$M = \frac{P_{AKTUELL}}{P_{MINIMAL}} * 100 - 100$$

Der Mehraufwand ergibt sich hier aus dem Verhältnis der Anzahl aktuell belegter Datenblöcke ($P_{AKTUELL}$) zur theoretisch minimal benötigten Anzahl Datenblöcke ($P_{MINIMAL}$). Diese kann mit allgemein bekannten Vorgehensweisen zur Abschätzung von benötigtem Speicherplatz berechnet werden. Freiplatz, der in Datenblöcken bei deren Belegung bewusst frei gehalten werden soll, um die Gefahr von Satzauslagerungen zu verringern, muss bei der Berechnung von $P_{MINIMAL}$ berücksichtigt werden. Das praktische Minimum liegt dann über dem theoretisch denkbaren Minimum. Die Berücksichtigung von Pufferungsgraden kann hier unter der Annahme, dass diese durch eine Reorganisation nicht wesentlich beeinflusst werden, entfallen.

Bei der Implementierung einer Softwarelösung (Werkzeug) zur Quantifizierung des Nutzens von Datenbankreorganisationen bietet es sich an, je Degenerierungsart und Planoperator, der von der jeweiligen Degenerierung beeinflusst wird, eine eigene Funktion zur Ermittlung der Auswirkungen auf den I/O-Aufwand zu implementieren.

3.4 Zusammenführung von Mehraufwandswerten und Aufwandsanteilen

Aufgrund der meist großen Menge zu bewegender Daten und der auftretenden Behinderungen des Datenbankbetriebs werden Reorganisationen i.d.R. nicht für alle in einer Datenbank gespeicherten Objekte ausgeführt. Üblicherweise wird eine Auswahl bestimmter Datenbankobjekte (*Reorganisationskandidaten*) getroffen. Diese Auswahl kann z.B. zunächst aufgrund von Informationen über den Degenerierungsgrad der Datenbankobjekte oder aufgrund von Aussagen der Benutzer zum Antwortzeitverhalten (bzw. allgemein anhand bestimmter *Workload-Teile*) erfolgen.

Die Auswahl der Datenbankobjekte, die als Reorganisationskandidaten in Betracht kommen, kann *objekt-* oder *workload-bezogen* erfolgen. Einen Überblick über die Möglichkeiten zeigt Abbildung 4.

Eine objektbezogene Auswahl wird i.d.R. dann erfolgen, wenn bereits bekannt ist, dass ein oder mehrere Datenbankobjekte Degenerierungen der physischen Speicherungsstrukturen aufweisen. Eine workload-bezogene Auswahl ist ratsam, wenn die Anwender ein unzureichendes Antwortzeitverhalten bei bestimmten Aktionen bemängeln und der Grund für die Verschlechterung in Degenerierungen der physischen Speicherungsstrukturen der von der Aktion betroffenen Datenbankobjekte gesehen wird.

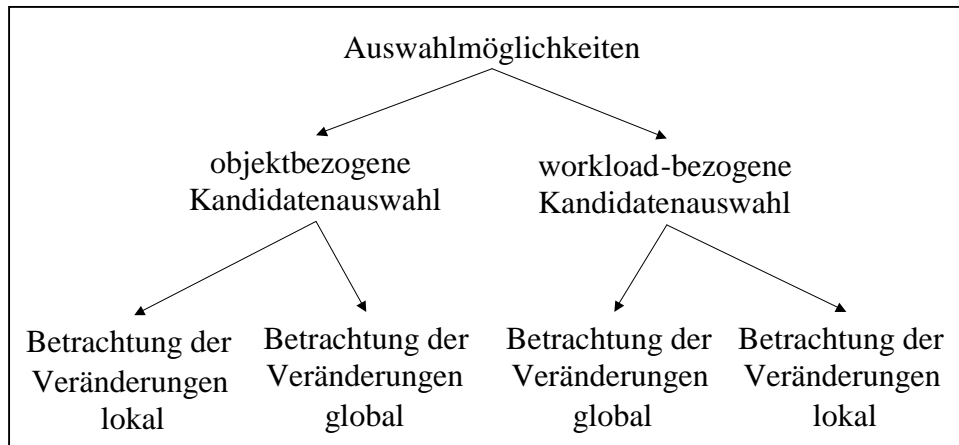


Abbildung 4: Auswahl von Reorganisationskandidaten und zu berücksichtigenden Anweisungen

Die Auswahl der zu berücksichtigenden Operationen wird nicht nur durch die Auswahl der Reorganisationskandidaten beeinflusst, sondern auch dadurch, ob die Betrachtungen der zu erwartenden Veränderungen des Aufwands zur Workload-Abarbeitung „*lokal*“ oder „*global*“ erfolgen sollen.

Bei den **lokalen** Betrachtungen nach einer **objektbezogenen** Kandidatenauswahl werden nur die Planoperatoren einbezogen, die auf die Reorganisationskandidaten angewendet werden, weil z.B. nur die Auswirkungen auf die gegen die Reorganisationskandidaten angewendeten Operationen von Interesse bei der Analyse sind.

Nach einer **anweisungsbezogenen** Kandidatenauswahl kann **lokal** z.B. betrachtet werden, wie sich die Reorganisation eines oder mehrerer von der betrachteten Anweisung benötigten Datenbankobjekte auf den zur Abarbeitung dieser Anweisung notwendigen I/O-Aufwand auswirkt.

Bei **globalen** Betrachtungen werden die durch eine Reorganisation hervorgerufenen Veränderungen des I/O-Aufwands bezüglich der Abarbeitung der gesamten protokollierten Workload betrachtet. Hier kann die Auswahl der Reorganisationskandidaten zwar objekt- oder anweisungsbezogen erfolgen, die übrigen Vorgehensweisen bleiben aber jeweils gleich.

Abbildung 5 zeigt die Pseudocode-Darstellung einer Funktion zur Ermittlung des Nutzens einer Datenbankreorganisation. Dabei wird auch auf die in Abbildung 1 dargestellten Schritte Bezug genommen. Als Parameter wird der Funktion das Workload-Protokoll (*wkl_log*) und eine Liste der Reorganisationskandidaten (*reorg_objects*) übergeben.

```

1  function nutzen(wkl_log,reorg_objects)
2  begin
3      all_planops,plan : list of planop; { Variablen für Gesamt-Plan und Plan
                                           einer Anweisung }
4      i statement;
5
6      all_planops:=nil;
7      for i in wkl_log do
8          plan:=nil;
9
10         { Ausfuehrung von Schritt III }
11         plan.call_optimizer(i);
12
13         for j in plan do
14             { Ausfuehrung der Schritte IV und V }
15             if j in all_planops then
16                 operator:=all_planops.locatesimilar(j);
17                 operator.cost:=j.cost() * i.anz_ausfuehrungen;
18             else
19                 all_planops.append(j);
20                 operator:=all_planops.locatesimilar(j);
21                 operator.cost:=j.cost() * i.anz_ausfuehrungen;
22             fi;
23         od;
24     od;
25     total_cost_before_reorg:=0;
26     total_cost_after_reorg:=0;
27     for k in all_planops do
28         { Ausfuehrung von Schritt VII }
29         total_cost_before_reorg:= total_cost_before_reorg + k.cost;
30         { Ausfuehrung der Schritte VI und VIII }
31         if k.object in reorg_objects then
32             total_cost_after_reorg:= total_cost_after_reorg +
33                                     k.cost / (1 + k.mehraufwand());
34         else
35             total_cost_after_reorg:= total_cost_after_reorg +
36                                     k.cost;
37         fi;
38     od;
39     { Ausfuehrung von Schritt IX }
40     nutzen:= (1 - total_cost_after_reorg / total_cost_before_reorg) * 100;
41 end;

```

Abbildung 5: Ermittlung des Nutzens einer geplanten Datenbankreorganisation

Nach der Initialisierung des Gesamtplans in Anweisungszeile 5 werden schrittweise für alle Anweisungen des übergebenen Anweisungsprotokolls die Ausführungspläne bestimmt (Zeile 8). Kommen in einem Ausführungsplan Operatoren vor, die bereits im Gesamtplan der Workload enthalten sind (Prüfung in Zeile 10), so wird der entsprechende Operator im

Gesamtplan lokalisiert (Zeile 11). Anschließend werden die Kostenwerte für den aktuellen Operator in den Gesamtplan eingearbeitet (Zeile 12). Planoperatoren, die noch nicht im Gesamtplan vorkommen, werden samt Kostenwerten neu eingefügt (Zeilen 14 bis 16).

Nachdem der Gesamtplan erstellt wurde, können die Summen (Zeilen 22 bis 31) der anfallenden Blockzugriffe vor (Zeile 23) und nach (Zeilen 25 bis 30) der Reorganisation gebildet werden. Zur Berechnung der Anzahl Blockzugriffe nach der Reorganisation werden bei allen Planoperatoren, die auf zu reorganisierende Datenbankobjekte angewendet werden, die Kosten um die Mehraufwandsanteile verringert (Zeilen 15 und 26). Abschließend wird in der Anweisungszeile 32 der Nutzen der Datenbankreorganisation, also die durch die Reorganisation zu erwartende prozentuale Veränderung der notwendigen Blockzugriffe berechnet.

4 Evaluierung an einem Beispiel

Zu Überprüfungszwecken wurde der in Abbildung 6 dargestellte Umweltausschnitt unter Oracle implementiert.

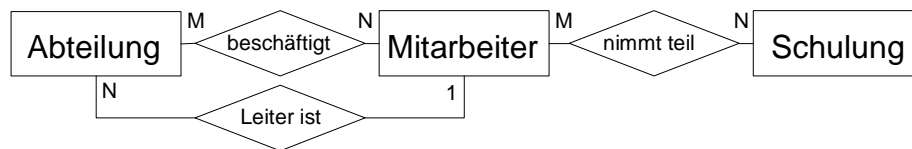


Abbildung 6: Umweltausschnitt des Beispiels

Dazu wurden folgende Tabellen angelegt und mit per Zufallszahlengenerator erzeugten Daten gefüllt.

```

MITARBEITER(pnr, name, vorname, gebdat, strasse, plz, ort, telefon, gehalt)
SCHULUNG(snr, thema, inhalt, dozent, ort, anzstd)
ABTEILUNG(anr, bezeichnung, abt_leiter)
TEILNAHME(pnr, snr, von, bis)
BESCHAEFTIGUNG(pnr, anr, von, bis, b_art)
    
```

Indexiert wurden die Tabellen jeweils über die Primärschlüssel. Die Tabelle ABTEILUNG wurde noch zusätzlich über den Fremdschlüssel abt_leiter indexiert. Danach wurden Update-Operationen ausgeführt, um gezielt Degenerierungen (z.B. eingestreuten Freiplatz durch Löschoptionen) in den Speicherungsstrukturen der Datenbankobjekte zu erzeugen. Dabei ist es durch die erzeugten recht hohen Degenerierungsgrade auch möglich, gut die Workload-Abhängigkeit des Nutzens von Datenbankreorganisationen sowie die erreichte Genauigkeit der Abschätzungen darzustellen. Einen Auszug aus den Statistik-Daten zeigt Tabelle 1. Die Angaben zu den Indexen beziehen sich dabei jeweils auf die Primärschlüsselindexe. Der Fremdschlüsselindex über der Tabelle ABTEILUNG besitzt drei Ebenen und auf der Blattebene 1095 Knoten.

Tabelle	belegte Blöcke	Freiplatzanteil	Tupelzahl	ausgel. Tupel	Indexebenen	Anz. Blätter
MITARBEITER	4208	42 %	62693	1711	3	415
SCHULUNG	6363	42 %	62970	2952	3	443
ABTEILUNG	1299	43 %	63112	4091	3	443
TEILNAHME	2379	14 %	263035	0	3	2289
BESCHAEFTIGUNG	7203	15 %	262588	0	3	2268

Tabelle 1: Statistiken der Beispieltabellen

Aus den in Abbildung 7 gezeigten SQL-Anweisungen wurden vier unterschiedliche Beispiel-Workloads mit je 10000 Anweisungen erzeugt. Dabei werden in den verschiedenen generierten Workloads nicht immer alle Anweisungen verwendet.

- Die erste Workload enthält die Anweisungen 0 bis 6,
- die zweite die Anweisungen 0 und 1 und
- die Workload Nr. 3 enthält die Anweisungen 0, 4, 5 und 6.
- Die vierte Workload enthält alle acht aufgeführten Anweisungen.

Ziel ist es, damit die Workload-Abhängigkeit des Nutzens von Datenbankreorganisationen zu verdeutlichen. Die Reihenfolge der Ausführung der enthaltenen Anweisungen wurde per Zufallszahlengenerator festgelegt.

0: SELECT * FROM mitarbeiter WHERE pnr=???
1: SELECT name, vorname, gebdat FROM mitarbeiter;
2: SELECT * FROM abteilung WHERE bezeichnung=???
3: SELECT * FROM schulung WHERE thema=???
4: SELECT M.telefon, M.name, M.vorname, A.bezeichnung FROM mitarbeiter M, abteilung A WHERE A.abt_leiter=??? AND M.pnr=A.abt_leiter;
5: SELECT M.telefon, M.vorname, M.name FROM mitarbeiter M, abteilung A, beschaeftigung B WHERE A.abt_leiter=??? AND A.anr=B.anr AND B.pnr=M.pnr;
6: SELECT M.telefon, M.vorname, M.name, S.dozent, S.ort FROM mitarbeiter M, schulung S, teilnahme T WHERE S.thema=??? AND S.snr=T.snr AND T.pnr=M.pnr;
7: SELECT * FROM abteilung WHERE anr>=??? AND anr<=???

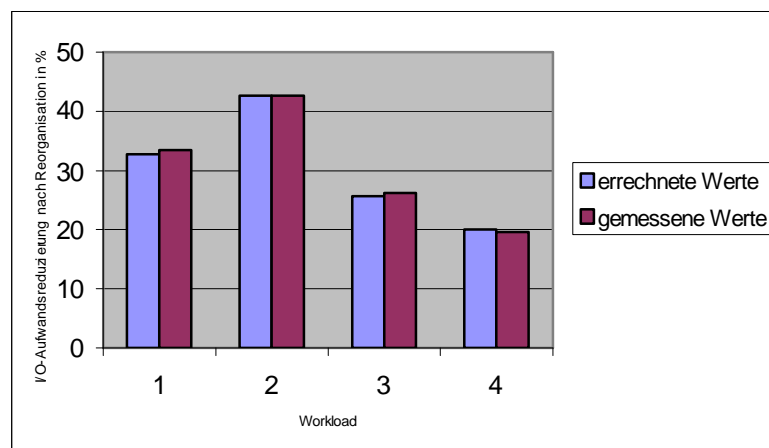
Abbildung 7: Anweisungen zur Generierung von Beispiel-Workloads

In der Beispielumgebung werden die Anweisungen aufgrund entsprechender Entscheidungen des Oracle Optimizers wie folgt realisiert:

- Anweisung 0 über einen **Index Lookup**,
- die Anweisungen 1, 2 und 3 jeweils durch **sequentielle Suche** über den jeweiligen Tabellen.

- Anweisung 4 wird über einen **Nested Loop Join** realisiert. Dabei wird zunächst jeweils über einen Index-Bereichs-Scan auf die Tabelle ABTEILUNG zugegriffen. Danach wird über einen Index Lookup die Verbindung zur Tabelle MITARBEITER hergestellt.
- Für Anweisung 5 werden zunächst die Tabellen ABTEILUNG und BESCHAEFTIGUNG über einen **Hash Join** verbunden. Der Zugriff auf die Tabelle ABTEILUNG erfolgt über einen Index-Bereichs-Scan und der Zugriff auf die Tabelle BESCHAEFTIGUNG über eine sequentielle Suche über die gesamte Tabelle. Das Ergebnis des Hash Joins wird über einen Nested Loop Join mit der Tabelle MITARBEITER verknüpft, auf die über einen Index Lookup zugegriffen wird.
- Bei der Anweisung 6 wird zunächst über sequentielle Suchen auf die Tabellen SCHULUNG und TEILNAHME zugegriffen. Diese werden über einen Nested Loop Join verbunden. In der äußeren Schleife wird dabei auf SCHULUNG und in der inneren Schleife auf TEILNAHME zugegriffen. Das Ergebnis der Verknüpfung wird über einen weiteren Nested Loop Join mit der Tabelle MITARBEITER verknüpft, auf die mittels Index Lookups zugegriffen wird.
- Anweisung 7 wird über einen **Index-Bereichs-Scan** über der Tabelle ABTEILUNG realisiert.

Für die Beispiel-Workloads wurde der Nutzen einer Reorganisation der Beispieldatenbank einerseits wie vorn beschrieben berechnet. Dazu wurde das Kostenmodell noch um Kostenfunktionen für die anfallenden Join-Operationen erweitert. Zur Überprüfung wurde andererseits die jeweilige Workload vor und nach einer Reorganisation auf die Datenbank angewendet. Dabei wurden die tatsächlich anfallenden Blockzugriffe gemessen und den



berechneten Werten gegenübergestellt.

Abbildung 8: Vergleich errechneter und gemessener Nutzen von Datenbankreorganisationen bei gleichem Degenerierungsgrad und unterschiedlicher Workload-Zusammensetzung

Die unterschiedlichen Ergebnisse für die Workloads 1 und 4 zeigen gut die Workload-Abhängigkeit des Nutzens von Datenbankreorganisationen. Beide Workloads unterscheiden sich darin, dass in Workload 4 die Anweisung Nr. 7 enthalten ist und in Workload 1 nicht. Anweisung 7 macht durch große Scan-Bereiche einen erheblichen Anteil am Gesamtaufwand für die Workload-Abarbeitung aus. Durch die relativ wenigen ausgelagerten Tupel in der Tabelle ABTEILUNG ist der vor der Reorganisation bei der Ausführung von Anweisung 7

auftretende Mehraufwand jedoch gering, was sich deutlich auf den Nutzen der Datenbankreorganisation auswirkt.

Um die Messwerte mit den über Kosten- und Mehraufwandfunktionen errechneten Werten vergleichen zu können, wurden die Werte der Pufferungsgrade mit null angenommen, da datenbankobjektbezogene Pufferungsgrade nicht zur Verfügung standen. Der Vergleich der errechneten Werte mit dem tatsächlich erreichten Nutzen zeigt, dass es mit der vorgestellten Methode möglich ist, den voraussichtlichen Nutzen einer Datenbankreorganisation relativ genau vorab zu errechnen.

Unter der Annahme, dass sich der Pufferungsgrad der Blöcke von Datenbankobjekten durch eine Reorganisation nicht oder nur unwesentlich ändert, hat dieser auf die relative Veränderung der I/O-Kosten nur dann einen Einfluss, wenn der Zugriff auf die Tupel einer Tabelle über einen Index erfolgt, da dieser i.d.R. einen Pufferungsgrad aufweist, der deutlich höher ist, als der der Datenblöcke. Dies führt dazu, dass sich die Degenerierungen im Datenbereich (wie z.B. ausgelagerte Tupel) relativ gesehen deutlich stärker auf die Antwortzeit auswirken. Würde man also die durch eine Datenbankreorganisation verursachte Veränderung der Antwortzeit betrachten, so wäre diese prozentual teilweise sogar größer als die hier errechneten Werte.

Um sicherzustellen, dass errechnete und gemessene Werte vergleichbar sind, ist es weiterhin wichtig dafür zu sorgen, dass die Veränderungen der Zahl der Blockzugriffe nur durch die Reorganisation hervorgerufen werden. Veränderungen an den Daten durch Einfüge- bzw. Löschoptionen in der Beispiel-Workload würden ebenfalls zu einer Veränderung des Aufwands führen und damit die Ergebnisse verfälschen. Deshalb enthält die Beispiel-Workload nur Retrieval-Operationen.

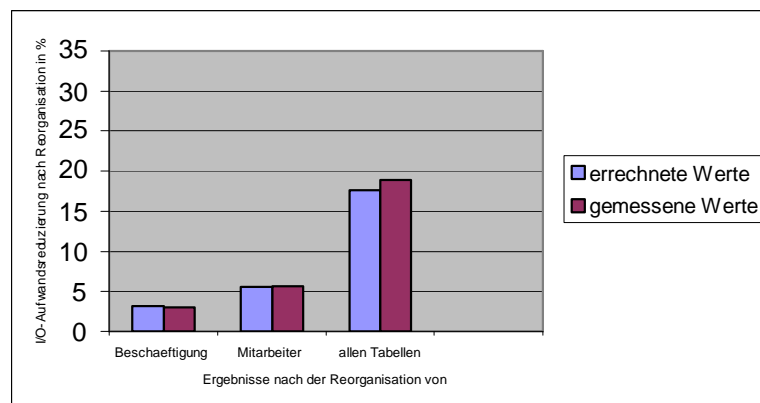


Abbildung 9: Vergleich errechneter und gemessener Nutzen von Datenbankreorganisationen bei schrittweiser Reorganisation einzelner Datenbankobjekte

In realen Workloads sind Änderungsoperationen aber natürlich enthalten. Es wird aber davon ausgegangen, dass, wenn die Workload nicht dominierend aus Änderungsoperationen besteht, mit der vorgestellten Methode trotzdem Aussagen mit einer hohen Genauigkeit möglich sind. Genauere Untersuchungen hierzu werden noch durchgeführt. Bei Änderungs- und Löschoptionen verändert sich durch eine Reorganisation der Aufwand zum Auffinden der jeweils betroffenen Tupel. Diese Suchoperationen sind auch in den Ausführungsplänen ausgewiesen und werden bei der Bestimmung des Reorganisationsnutzens berücksichtigt. Lediglich die Zugriffe, die für die tatsächlichen Einfüge-, Änderungs- und Löschoptionen in den Datenblöcken anfallen, fließen nicht in die Betrachtungen ein.

Die Vorgehensweise bei den Versuchen, deren Ergebnisse in Abbildung 9 dargestellt sind, war ähnlich. Allerdings wurde hier immer die selbe Beispiel-Workload verwendet, welche die Anweisungen 1 bis 6 gleichhäufig enthielt. Die Reorganisation erfolgte schrittweise. Zunächst wurde nur die Tabelle BESCHAEFTIGUNG reorganisiert, danach die Tabelle MITARBEITER und abschließend nochmals die gesamte Datenbank. Dabei wurde die teilweise reorganisierte Datenbank jeweils wieder als Ausgangspunkt für den nächsten Versuch verwendet.

Die Ergebnisse zeigen, dass mit der beschriebenen Vorgehensweise eine Abschätzung des Nutzens einer Datenbankreorganisation auch dann mit hoher Genauigkeit möglich ist, wenn (wie in der Praxis i.d.R. üblich) nur Teile der Datenbank reorganisiert werden.

5 Zusammenfassung und Ausblick

In diesem Beitrag wurde eine Methode zur Quantifizierung des Nutzens von Datenbankreorganisationen auf der Ebene der physischen Speicherungsstrukturen beschrieben. Als Nutzen wird hier hauptsächlich die durch eine Datenbankreorganisation erreichbare prozentuale Veränderung des I/O-Aufwands betrachtet. Die Methode berücksichtigt den zum Analysezeitpunkt aktuellen Zustand der Speicherungsstrukturen und die gegen die Datenbank gerichtete Workload. Insbesondere die Workload-Berücksichtigung unterscheidet die hier vorgestellte Methode von den Vorgehensweisen von derzeit am Markt verfügbaren Produkten für Reorganisationsbedarfsanalysen. Damit ist es möglich, die zu erwartenden Auswirkungen der Reorganisation auf die Systemleistung zu quantifizieren. Zur Workload-Berücksichtigung werden in einem repräsentativen Zeitraum die gegen die Datenbank gerichteten SQL-Anweisungen protokolliert. Nachdem unter Nutzung des Anfrageoptimierers die Planoperatoren bestimmt wurden, die zur Realisierung der Workload verwendet werden, kann der für die Abarbeitung der Workload benötigte Zahl der Blockzugriffe und der durch Degenerierungen verursachte Mehraufwand ermittelt werden.

Kernprobleme der Abschätzung des Nutzens von Datenbankreorganisationen stellen die Ermittlung der I/O-Kosten unter Berücksichtigung vorhandener Degenerierungen und die Bestimmung des durch Degenerierungen verursachten Mehraufwands dar. Beides wurde eingehender betrachtet. Als Kostenmaß wurden Blockzugriffe verwendet. Die präsentierten Berechnungsformeln schließen auch die Berücksichtigung von datenbankobjektbezogenen Pufferungsgraden mit ein. Hier ergibt sich aber das Problem, dass solche Pufferungsgrade von den gängigen DBMS-Produkten derzeit nicht direkt zur Verfügung gestellt werden. Die Ergebnisse des Beispiels zeigen, dass es mit der vorgestellten Methode möglich ist, die Auswirkungen einer Datenbankreorganisation auf die Anzahl der zur Workload-Abarbeitung anfallenden Blockzugriffe mit hoher Genauigkeit abzuschätzen. Dies kann DBAs die Entscheidung über eine Reorganisationsdurchführung deutlich erleichtern.

Die Genauigkeit der Abschätzungen ist vom verwendeten Kostenmodell abhängig. Gegenstand laufender Untersuchungen ist die Erweiterung des Kostenmodells um zusätzliche Operationen, insbesondere auch Update-Operationen. Weiterhin werden Möglichkeiten untersucht, bei existierenden DBMS-Produkten datenbankobjektbezogene Pufferungsgrade zu ermitteln und im Rahmen von Reorganisationsbedarfsanalysen zu nutzen.

Literaturverzeichnis

- [BR02] D. Beeler, I. Rodriguez. *Optimizing Your Database Performance ... the Easy Way*. BMC Software Inc., 2002
- [CGN02] S. Chaudhuri, A. K. Gupta, V. Nasarayya. Compressing SQL Workloads. In Proc. Of ACM SIGMOD, Madison Wisconsin, USA, 2002
- [CN97] S. Chaudhuri, V. Narasayya. An Efficient, Cost-Driven Index Selection Tool for Microsoft SQL Server. In *Proc. of the 23rd VLDB Conference*, Athen, Griechenland, 1997
- [CN98] S. Chaudhuri, V. Narasayya. AutoAdmin „What-if“ Index Analysis Utility. In *Proc. of ACM SIGMOD*, Seattle, USA, 1998
- [DK00] S. Dorendorf, K. Küspert. Datenbankreorganisation bei relationalen Datenbank-Management-Systemen. In *it+ti – Informationstechnik und Technische Informatik Heft 3/2000*. Oldenbourg Wissenschaftsverlag GmbH München, Juni 2000
- [Dor00] S. Dorendorf. *Beschreibung eines Speicher- und Verhaltensmodells als Grundlage zur Bedarfsanalyse einer Datenreorganisation bei relationalen Datenbank-Management-Systeme*. Jenaer Schriften zur Mathematik und Informatik Mat/Inf/00/05, Friedrich-Schiller-Universität Jena, 2000
- [Dun02] O. Dunemann. *Anfrageoptimierung für OLAP-Anwendungen in virtuellen Data Warehouses*. Dissertation, Otto-von-Guericke-Universität Magdeburg, Fakultät für Informatik, September 2002
- [Hel01] T. Helm. *Dokumentation des Katalogs des DBMS ORACLE und Transformation ausgewählter Katalogdaten in ein einheitliches Informationsschema*. Studienarbeit, Berufsakademie Thüringen, Staatliche Studienakademie, Studienabteilung Gera, Studienrichtung Wirtschaftsinformatik, Gera, Februar 2001
- [HR01] Th. Härder, E. Rahm. *Datenbanksysteme: Konzepte und Techniken der Implementierung, II überarb. Auflage*. Springer-Verlag, Berlin, Heidelberg, New York, 2001
- [IBM02] *IBM DB2 Universal Database Command Reference Version 8*. International Business Machines Corporation, 2002
- [Mak03] M. Makoui. *Heuristische Anfrageoptimierung in Relationalen Datenbanken*. Diplomarbeit, Universität Hannover, Fachbereich Informatik, Institut für Informationssysteme, Januar 2003
- [Now01] J. Nowitzky. Partitionierungstechniken in Datenbanksystemen: Motivation und Überblick. In *Informatik Spektrum 24/6*. Springer-Verlag, Heidelberg, Dezember 2001
- [ORA02] *Oracle Enterprise Manager Database Tuning with the Oracle Tuning Pack Release 9.0.1*. Oracle Corporation, 2002
- [Ri03] T. Richter. *Application of Informix Dynamic Server with regard to high Availability at AMD Saxony*. 7th East European Conference, ADBIS 2003, Dresden 3.9.-6.9.2003.

- [Sch03] R. Schumacher. *Resolving Oracle Space Problems Using Embarcadero Space Analyst*. Embarcadero Technologies, Inc., Juni 2003
- [SD03] S. Skatulla, S. Dorendorf. Optimization of Storage Structures of Complex Types in Object-Relational Database Systems. In Kalinichenko et al. *Advances in Databases and Information Systems*. Proc. of the 7th East European Conference, ADBIS 2003, Dresden 3.9.-6.9.2003. Springer-Verlag Berlin Heidelberg, 2003
- [SA⁺79] P.Selinger, M. Astrahan, D. Chamberlin, R. Lorie, T. Price. Access Path Selection in a Relational Database Management System. In *Proc. Of the ACM SIGMID Conference*. 1979.