Leistungsorientierte Auswahl von Reorganisationskandidaten

Kevin Röwe[1], Fritz Walliser[1], Norbert Ritter[2]

[1]InfoDesign GmbH
Großes Feld 23
25421 Pinneberg
Kevin.Roewe@infodesigner.biz
Walliser@infodesign-msx.de

[2]Fachbereich Informatik
Universität Hamburg
Vogt-Kölln-Str. 30
22527 Hamburg
Norbert.Ritter@informatik.uni-hamburg.de

Abstract: Datenbank-Reorganisationen stellen moderne IT-Abteilungen vor massive Herausforderungen. Auf der einen Seite versprechen sie die Bereinigung von durch Degenerierung verursachten Leistungseinbußen, auf der anderen Seite verbrauchen Reorganisationsprozesse massiv an anderer Stelle dringend benötigte Ressourcen. Aktuell verwendete Ansätze zur Auswahl von Reorganisationskandidaten (d.h. von Speicherbereichen, deren Reorganisation potenziell von Nutzen ist) nehmen kaum eine Erfolgsbewertung und –messung vor, sondern beruhen größtenteils auf statischen Verfahren. Dies gibt grundsätzlich Anlass zur Überprüfung. Dieser Beitrag zeigt die Schwächen aktueller Ansätze auf und beschreibt einen in Zusammenarbeit der Universität Hamburg mit der Firma InfoDesign entwickelten Ansatz einer leistungsorientierten Auswahl von Reorganisationskandidaten. Dieser basiert auf der systematischen Erhebung von geeigneten Leistungskennzahlen und dem Prognostizieren der durch Reorganisation erreichbaren Leistungssteigerungen. Evaluationen zeigen, dass die Anzahl von Reorganisationen deutlich reduziert werden kann, in dem nur tatsächlich zu Leistungssteigerungen führende Reorganisationskandidaten ausgewählt werden.

1 Einführung

In betrieblichen Systemumgebungen kommt es im Laufe der Zeit insbesondere bedingt durch die verschiedenartigen Manipulationsmöglichkeiten der Datenbasis zu einer Verschlechterung der Leistung. In diesem Fall ist eine Reorganisation der Datenbank sinnvoll. Bei einer Reorganisation werden die Daten mit dem Ziel einer Leistungsverbesserung in einer optimalen physischen Reihenfolge neu angeordnet. In hochverfügbaren Systemlandschaften können diese Optimierungen aus Performancegründen in der Regel nicht im laufenden Betrieb während der Kernarbeitszeiten vorgenommen werden, so dass diese regelmäßig (z. B.) am Wochenende stattfinden. Durch das begrenzte Zeitfenster, das zusätzlich noch mit anderen Wartungsarbeiten, wie z. B. dem Backup, geteilt werden muss, und häufig sehr großer Datenmengen ist eine sorgfältige Auswahl der zu reorganisierenden Einheiten und der zugehörigen Reorganisationszeitpunkte geboten. Derzeit existieren Werkzeuge, die dem Administrator entsprechende Reorganisationskandidaten (d.h. von Speicherbereichen, deren Reorganisation als potenziell nützlich

erachtet wird) vorschlagen. Es bestehen jedoch erhebliche Zweifel daran, dass diese Empfehlungen „optimal“ sind, d.h. genau solche Reorganisationen vorgeschlagen werden, die auch tatsächlich einen bedeutsamen Leistungsgewinn herbeiführen; vielmehr besteht die Befürchtung, dass unnötigerweise zu viel reorganisiert wird. Die Firma InfoDesign hat im Rahmen einer Kooperation mit der Uni Hamburg aktuelle Lösungen zur Reorganisationseinplanung und -vermeidung untersucht, Schwachpunkte identifiziert und darauf basierend einen alternativen Lösungsansatz zur Auswahl der zu verwendenden Reorganisationskandidaten entwickelt. Der Lösungsansatz wurde technisch implementiert und im Praxistest evaluiert. Er wird in diesem Artikel vorgestellt.

Die einfache Grundidee dieses Ansatzes ist, dass eine Datenbank-Reorganisation dann als „erfolgreich“ betrachtet werden kann, wenn der auf der Datenbank bzw. auf dem entsprechenden Granulat stattfindende Workload nach Durchführung der Reorganisation sich in seinem Performanceverhalten beträchtlich verbessert. Das Performanceverhalten kann beispielsweise mit Kennzahlen, wie z. B. Antwortlaufzeitverhalten, I/O-/CPU-Aufwand, o.Ä. gemessen werden. Weiter sollen nur Datenbankgranulate als Kandidaten herangezogen werden, deren Leistungsverhalten sich seit der letzten Reorganisation über die Zeit hinweg verschlechtert hat. So einfach dieser Ansatz auch klingt, so wird dennoch eine solche Vorgehensweise bisher nicht zur Kandidatenauswahl verwendet. Vielmehr werden für die Auswahl häufig statische Regeln verwendet, die auf Grundlage der statistischen Informationen des Datenbankkatalogs arbeiten und aufgrund der (im Hinblick auf den Zweck der Reorganisation vorherrschenden) Unschärfe dieser Informationen häufig völlig unnötige, weil nicht zu wirklichen Leistungsverbesserungen führende, aber dennoch kostenintensive Reorganisationsvorgänge veranlassen.

2 Grundlagen

2.1 Datenbank-Reorganisation

Reorganisationen werden vor Allem mit dem Ziel der Performanceverbesserung durch Optimierung der Speicherausnutzung durchgeführt. In [SG79] werden unter anderen folgende Szenarien angeführt, in denen eine Reorganisation sinnvoll ist:

- Schemaänderungen, z.B. Hinzufügen eines neuen Attributs,
- starker Anstieg der Datenmenge,
- Partitionierung einer Datenbank in mehrere unabhängige Einheiten bzw. Entfernen einer großen Teilmenge von Daten aus einer Datenbank, zum Beispiel aufgrund einer veränderten Gesetzgebung,
- große Menge von Einfüge- und/oder Löschoperationen.

[SH94] wird in der Definition konkreter und beschreibt die Datenbank-Reorganisation als einen Prozess, der primär auf eine Verbesserung der Geschwindigkeit und Effizienz der Datenbank abzielt, da im Anschluss an eine Datenbank-Reorganisation die Daten in einer optimalen Anordnung auf dem Sekundärspeicher liegen. [WV09] nimmt eine gröbere Klassifikation der zugrunde liegenden Ursache der Notwendigkeit für eine Reorganisation vor; hierbei werden zwei Gründe der Durchführung benannt: „Verschlechterung der I/O-Performance“ und „Speicherplatz-Ausnutzung bzw. -Rückgewinnung“. Der

Schwerpunkt unserer Betrachtungen liegt auf der Überprüfung und Bewertung des Performanceverhaltens. Der Aspekt ‚Speicherplatzausnutzung und -rückgewinnung' wird nicht berücksichtigt, da die zughörigen Ursachen gut anhand von Statistiken aus dem Datenbankkatalog überprüfbar sind. [SH94] geht (weiter) davon aus, dass eine desorganisierte Anordnung von Daten zu einer signifikanten Leistungsverschlechterung des Datenbanksystems führt. Die desorganisierte Anordnung von Daten kommt dabei automatisch durch typische Datenbankoperationen im täglichen Betrieb zustande. Im Folgenden werden drei typische Betriebsprobleme nach [SH94] aufgezeigt, die in klassischen OLTP-Datenbanksystemen bei fortwährendem Betrieb entstehen.

Fragmentierung. Fragmentierung liegt vor, wenn logisch zusammengehörende Daten über einen größeren bzw. weit auseinander liegende Speicherbereiche verteilt werden. Insbesondere durch die Such- und Positionierungsoperationen der Lese- und Schreibköpfe ist mit Leistungseinbußen zu rechnen. Fragmentierungen können beispielsweise dann entstehen, wenn große Teile einer Datenbasis gelöscht wurden.

Row Chaining und Row Migration. Unter einer „chained row" versteht man einen Datensatz, der zu groß ist um in einer einzelnen Datenbankseite zu passen. Unter „row migration" versteht man ein ähnliches Problem. Der Unterschied ist, dass „row migration" insbesondere dann angewendet wird, wenn für ein Update des Datensatzes nicht ausreichend Platz in der ursprünglichen Datenbankseite zur Verfügung steht. Das Datenbanksystem lagert dann diesen Datensatz in einen anderen Bereich im Segment aus und verweist mit einem entsprechenden Pointer darauf. Ein Datensatz kann dann nicht mehr mit einer einzelnen physischen Datenbankoperation ausgelesen werden, sondern es müssen mehrere Operationen stattfinden, um das Auslesen zu ermöglichen. Bedingt dadurch kommt es zu einer signifikanten Drosselung der Geschwindigkeit [CM01].

Declustering. „Declustering" kommt vor, wenn kein Einfügeplatz für einen Datensatz in seiner logischen Reihenfolge über den Clustering-Schlüssel verfügbar ist. In diesem Fall versucht das Datenbankverwaltungssystem den Datensatz dort einzufügen, wo Platz ist. Darunter leiden insbesondere sequentielle I/O-Operationen [CM01].

Die genaue Implementierung einer Datenbank-Reorganisation bleibt in der Regel unklar, da die Umsetzungen von Datenbanksystemherstellern nicht offengelegt werden. Grundsätzlich wird zwischen zwei verschiedenen Varianten, der Offline- und der Online-Variante, unterschieden. Wie bereits angesprochen, betrachten wir vor Allem die (praxisrelevantere) Offline-Variante. Diese erfordert, dass das entsprechende zu reorganisierende Datenbankgranulat nicht im Datenbankzugriff ist. Vor allem bei Hochverfügbarkeitsanforderungen kann dies zu Problemen führen, da eine Reorganisation abhängig von der Größe eine signifikante Zeit in Anspruch nehmen kann. Die Funktionsweise der Offline-Variante wird nach [CM01] wie folgt beschrieben:

- Erstellung eines Backups der Datenbank,
- Exportieren der Daten (z.B. in eine temporäre Datei),
- Löschen der Datenbankobjekte,
- Erstellen der zuvor gelöschten Datenbankobjekte,
- Sortieren der exportierten Daten nach dem Clustering Key,
- Importieren der vorherigen Schritt sortierten Daten.

Anschließend werden alle Indizes neu aufgebaut. Nach der Reorganisation befinden sich die Daten in einer „optimalen“ physischen Anordnung und Workload-typische Phänomene wie Fragmentierung, Row Chaining und Declustering sind weitgehend ausgeschlossen, was wiederum in einem besseren Leistungsverhalten resultiert.

2.2 Der InfoMat

Die Firma InfoDesign ist unter anderem als Software-Tool-Hersteller im Bereich von IBM-DB2-Datenbank(sytem)en im Mainframe-Umfeld erfolgreich am Markt. Eine wichtige Zielsetzung ihrer Softwareprodukte ist dabei die Automation der DB2-Administration. Das Produkt „InfoMat“ unterstützt aktiv den Datenbank-Administrator in den Bereichen Datenbank-Reorganisation, Durchführung von „Runstats“, „Copies“ und „Modifies“ sowie bei „Backup“ und „Recovery“. Kunden, die den InfoMat produktiv einsetzen, verfügen typischerweise über eine IT-Landschaft, in der ein SAP-ERP-System eingesetzt und dieses auf einer DB2-Datenbank auf dem Mainframe-Betriebssystem z/OS läuft. Damit bewegen sich die Umfänge in dem Bereich einer „großen Installation“, die nicht mehr rein manuell administrierbar ist. Allein ein typisches SAP-System umfasst dabei in der Grundinstallation bereits etwa 100.000 Tabellen, wobei häufig sogar mehrere SAP-Systeme auf einem Mainframe laufen.

Das Reorganisationsmodul des InfoMat setzt hinsichtlich der Auswahl geeigneter Reorganisationskandidaten einen schwellwert-basierten Ansatz um. Dabei werden die zur Entscheidungsfindung zu betrachtenden Informationen größtenteils aus dem Statistik-Modul des Datenbankkatalogs gewonnen. Im InfoMat wurden hierzu 17 Regeln definiert (Tabelle 1), die – sollte eine dieser Bedingungen zutreffen – zur Einplanung des betroffenen Tablespace in die Reorganisation führen.

RB*	Beschreibung
RB02	Index verfügt über zu viele Extents
RB03	Tablespace verfügt über zu viele Extents
RB04	Aktive Seiten < 90% allokierte Seiten
RB05	Größter Katalog/DSNSpace unverhältnismäßig
RB06	10% Index-Seiten-Splits
RB07	LEAFDIST – Seiten zu weit auseinander
RB08	Clusterratio – Daten schlecht sortiert
RB09	Tabelle nicht in Clustered Folge

RB*	Beschreibung
RB10	20% Datenveränderung
RB11	DDL-Änderungen
RB12	GROESSE > MAXPRIQTYNP
RB13	Durch Compression eingestellt
RB14	Manuell (INFOREOL)
RB20	Reordered Row Format
RB25	Index Overallocation
RB30	Reorg mit Discard

Tabelle 1: Reorganisationsbedingungen des InfoMat / RB = Reorganisationsbedingung

Von uns durchgeführte, empirische Untersuchen zeigen, dass knapp 80% aller Reorganisationen auf die Regeln RB07 und RB10 zurück gehen. Leider besteht insbesondere an der Sinnhaftigkeit von RB10 großer Zweifel, da diese allein auf dem Entscheidungskriterium beruht, dass es im entsprechenden „Tablespace“ mehr als 20 Prozent Datenveränderungen gab. Mengenorientierte Änderungen von z.B. nicht indizierten Datenfeldern ohne größenmäßige Erweiterung derselben oder Einfügeoperationen großer Anzahl in Archiv-Tabellen würden diese Regel beispielsweise unnötigerweise auslösen.

2.3 Alternative Ansätze und Einordnung der InfoMat-Lösung

Die in Kapitel 2.2 aufgezeigten Reorganisationsbedingungen sind größtenteils nicht durch eigene Erfahrungswerte oder Forschungen der Firma InfoDesign entstanden, sondern beruhen auf Empfehlungen von SAP und IBM, die insbesondere auf die Charakteristika des SAP-Systems zurückzuführen sind. Die Anwender sind angehalten, diesen Empfehlungen zu folgen. Anwender, die den InfoMat und/oder SAP nicht einsetzen, behelfen sich in der Regel mit ähnlichen Schedulern oder haben eigene Skripte entwickelt, die den Administratoren bei der Umsetzung dieser oder ähnlicher Regeln behilflich sind. Typische Parameter für die InfoMat-unabhängige Einplanung sind dabei die Cluster Ratio, Overflow (Record Split), die Anzahl der Extents sowie Komprimierungsaspekte (Compress). Schwellwerte wurden dabei durch Erfahrungen und Beobachtungen festgelegt.

Im Bereich von DB2-LUW können Reorganisationskandidaten durch das IBM-eigene Tool REORGCHK ermittelt werden. REORGCHK arbeitet ähnlich schwellwert-basiert wie der InfoMat und verwendet hierzu standardmäßig 8 vordefinierte Regeln, die gegen den Datenbankkatalog geprüft werden. Die überprüften Parameter überlappen stark mit denen des InfoMats.

Das Thema Datenbank-Reorganisationen auf Oracle wird nach [QE10] sehr kontrovers diskutiert. QE10 führt vier Gründe für eine Reorganisation auf Oracle an, wobei die ersten 3 wiederum schwell-basiert umgesetzt werden:

- Schachtelungstiefe des Index,
- Migrated Rows,
- Archivierung und damit Löschen von Tabelleninhalten,
- Umstellung auf ein anderes Layout / Strukturänderungen.

Neben den hier beschriebenen sind uns keine grundlegend anderen Ansätze – weder in Praxis noch in der Wissenschaft – bekannt.

3 Auswahl von Reorg-Kandidaten nach Leistungsgesichtspunkten

Die zentrale Idee unseres neuen Lösungsansatzes ist es, die Entscheidung abhängig von der Performance-Entwicklung des SQL-Workloads zu machen. Hierzu bedarf es einer Protokollierung von geeigneten Leistungskennzahlen über die Zeitachse hinweg. Es wurden verschiedene Ansätze (u.a. automatisierte Durchführung eines vordefinierten SQL-Workloads unter Messung des Antwortzeitverhaltens, Nutzung des „DB2 Event-Monitors“, Auslesen von Daten auf Grundlage des „DB2 Dynamic Statement Cache“) untersucht. In Abwägung der jeweiligen Vor- und Nachteile wurde der Ansatz der Nutzung des „Dynamic Statement Cache“ (DSC) weiter verfolgt.

Der DSC ist ein geschützter Speicherbereich im DB2-System und zentraler Bestandteil des sogenannten EDM-Speichers. In dem EDM-Speicher ist auch der „Buffer Pool“ angesiedelt. Eine der Aufgaben des DSC ist es, einmal generierte Zugriffspläne von

dynamischen SQL-Anfragen zwischen zu speichern und für eine Wiederverwendung vorzuhalten. Wird eine SQL-Anfrage von einem Benutzer oder einem Programm aufgerufen, so wird zunächst einmal geprüft, ob der Ausführungsplan zu der entsprechenden SQL-Anfrage bereits zwischengespeichert wurde. Im Trefferfall kann der Zugriffsplan ohne erneute Generierung verwendet werden. Die Generierung eines Zugriffsplans kann insbesondere bei komplexen SQL-Anfragen in einer hohen CPU-Auslastung resultieren. So kann der Prozess bis zu knapp 90 Prozent der Transaktions-CPU-Kosten in Anspruch nehmen. Durch den DSC wird in der Regel eine erhebliche Performancesteigerung erreicht. Die im DSC gespeicherten dynamischen SQL-Anfragen enthalten benötigte Leistungskennzahlen zu den CPU-Kosten (Durchschnittswerte) und der Ausführungshäufigkeit der SQL-Anfragen. Die Werte liegen bereits in aggregierter Form vor. Wurde das „Prepared Statement“ in Verbindung mit Host-Variabeln verwendet, werden Literale in den SQL-Anfragen bereits durch ein spezielles Symbol (‚?‘) ersetzt, sodass hier der DSC bereits eine Reduzierung der Menge von SQL-Anfragen vornimmt und somit den „Logging Overhead“ vermindert.

Der DSC verwendet einen LRU-Algorithmus zur Verdrängung. Das heißt, die Anfrage wird verdrängt, die am längsten nicht mehr verwendet wurde. Wünschenswert wäre sicherlich ein LFU-Algorithmus, der die am seltensten benutzten Anfragen verdrängt. Da ein LFU-Algorithmus jedoch leider nicht implementiert wurde, muss ein alternativer Ansatz eingesetzt werden. In diesem Fall haben wir uns dafür entschieden, regelmäßig „Snapshots“ des DSC zu erzeugen. Durch das regelmäßige „Snapshotting“ des DSC erhält man einen guten Überblick über wichtige und repräsentative SQL-Anfragen. Es ist trotzdem nicht auszuschließen, dass im Falle eines ungünstigen Workloads eine Verzerrung stattfindet und womöglich zu einem oder mehreren Snapshot-Zeitpunkten nur nicht-repräsentative Anfragen aus dem DSC gewonnen werden können. Aus diesem Grund ist das Snapshotting-Intervall dynamisch und abhängig von dem Verdrängungsfaktor, der nach jedem Snapshot ermittelt wird. Steigt die Verdrängung, wird die Snapshot-Frequenz erhöht.

Die Inhalte des DSC residieren im EDM-Speicher. Mittels eines speziellen (SQL-) Kommandos (EXPLAIN STMTCACHE ALL) können die Daten in eine relationale Tabelle geschrieben werden. Wesentliche Tabellenfelder des DSC sind hierbei:

- eine eindeutige STMT_ID,
- der STMT_TEXT,
- die Ausführungshäufigkeit der SQL-Anfrage seit Aufnahme in den DSC (STAT_EXEC),
- die durchschnittliche Anzahl der zurück gelieferten Datensätze der SQL-Anfrage (STAT_PROW).
- die kumulierten CPU-Ausführungskosten (STAT_CPU).

Da die Entscheidung über die Notwendigkeit einer Datenbank-Reorganisation auf Werte in der Vergangenheit zurückgreifen muss, wurde eine entsprechende Historisierung vorgesehen, d.h. den aus dem DSC erhobenen Statement-Informationen werden entsprechende logische Zeitstempel hinzugefügt; hierzu folgendes Beispiel. Tabelle 2 zeigt

den Inhalt des DSC zum Zeitpunkt t, Tabelle 3 zum Zeitpunkt t+1. Tabelle 4 entspricht der transformierten Historisierungstabelle.

SQL-Anfrage	STAT_EXEC	STAT_CPU
Select * from Kunde where KdNr = ‚1011‘	1452	0.23
…		

Tabelle 2: Inhalt des DSC zum Zeitpunkt t

SQL-Anfrage	STAT_EXEC	STAT_CPU
Select * from Kunde where KdNr = ‚5311‘	1712	0.26
…		

Tabelle 3: Inhalt des DSC zum Zeitpunkt t+1

SQL-Anfrage	Zeitpunkt	STAT_EXEC	STAT_CPU
select Name from Kunde where KdNr = ‚?‘	t	1452	0.23
select Name from Kunde where KdNr = ‚?‘	t+1	1712	0.26
…			

Tabelle 4: Transformierte Historisierungstabelle

In diesem Beispiel wurde die konkrete Wertausprägung des Feldes KdNr in der WHERE-Bedingung durch ‘?’ ersetzt. Dies geschieht durch einen nachgelagerten SQL-Parsing-Prozess. Die transformierte Tabelle protokolliert zunächst einmal nur die extrahierten Werte aus dem DSC. Für eine weitere Auswertung sind die Daten noch nicht direkt verwendbar, da es sich bei den Werten STAT_CPU um Durchschnittswerte handelt – erhoben über die gesamte Laufzeit seit Aufnahme der SQL-Anfrage in den DSC. Die Entwicklung wird hierbei noch nicht korrekt aufzeigt. So ist zwar zu erkennen, dass zwischen den Zeitpunkten t und t+1 sich die durchschnittliche Zeit zur Abarbeitung der Anfrage verschlechtert hat, das genaue Ausmaß wird jedoch nicht aus der Differenz 0,03 (0,26 - 0,23) direkt ersichtlich. Die tatsächliche Verschlechterung zwischen den beiden Zeitpunkten wird durch die Überprüfung mit folgender Formel erkenntlich:

$$STAT\_EXEC(t) * STAT\_CPU(t) + ((STAT\_EXEC(t+1) - STAT\_EXEC(t)) * x = STAT\_EXEC(t+1) * STAT\_CPU(t+1)$$

In diesem Fall war die durchschnittliche Antwortzeit mit 0,43 CPU-Sekunden und einer Verschlechterung von fast 86 Prozent gegenüber dem vorherigen Zeitpunkt t deutlich verändert. Ziel der Transformation ist es, Aussagen über das Antwortlaufzeitverhalten von SQL-Anfragen entlang der Zeitachse beurteilen und analysieren zu können. Essentiell wichtig ist hier allerdings auch die Zuordnung der einzelnen SQL-Anfragen zu dem entsprechenden Datenbank-Reorganisationsgranulat. Im DB2-System findet die Reorganisation auf Tablespace-Basis statt. Das hat zur Folge, dass zu jeder protokollierten SQL-Anfrage der entsprechende Tablespace durch Abfrage des Datenbankkataloges herausgefunden werden muss. Sind diese SQL-Anfragen zugeordnet, ist der absolut gemessene Wert, z. B. für die aufgebrauchte CPU-Zeit, nicht mehr relevant. Vielmehr interessiert die relative Entwicklung (in Prozent) über die Zeit. Es ist also wünschenswert, eine Reduktion auf Tablespace-Basis durch Durchschnittswerterrechnung zu erreichen. Das sieht beispielhaft wie in Tabelle 5 dargestellt aus. Für den Tablespace TS1 fließen drei SQL-Anfragen in die Betrachtung ein. Hier kann man nun die prozentualen Veränderungen ausrechnen und die Werte normieren. Man sieht, dass sich die CPU-Zeit der SQL-Anfrage S1 zu jedem Zeitpunkt um 10 Prozent verschlechtert hat. Die

Antwortzeiten der SQL-Anfragen S2 und S3 haben sich wiederum stark verbessert; jede SQL-Anfrage pro Zeitpunkt um 50 Prozent. Bildet man nun einfach einen Durchschnittswert pro SQL-Anfrage und pro Zeitpunkt, so dominieren die Verbesserungen mit 50 Prozent der beiden (selten ausgeführten) SQL-Anfragen S2 und S3. Die aus einer ungewichteten Betrachtung der Tabelspaces gezogene Schlussfolgerung, nicht zu reorganisieren, wäre falsch. Notwendig ist eine gewichtete Berechnung, in die die Ausführungshäufigkeit der SQL-Anfragen mit eingeht. Ebenfalls ist es sinnvoll, die Kardinalität der Ergebnismenge mit der entsprechenden Gewichtung zu verknüpfen.

Tablespace	SQL-Anfrage	Anzahl ausgeführt	Durchschnittliche CPU-Zeit in Sekunden zum Zeitpunkt			
			t1	t2	t3	t4
TS1	S1	8000	0,4	0,44	0,484	0,5324
TS1	S2	2	500	250	125	62,5
TS1	S3	4	1000	500	250	125
TS2	S4	2	2	3	3,4	4
…						

Tabelle 5: Auf Tablespace reduzierte Durchschnittsberechnung

Durch die Historisierung werden CPU-Zeiten zu einzelnen SQL-Anfragen gesammelt. Diese werden anhand des Reorganisationsgranulats ‚Tablespace' logisch gruppiert.

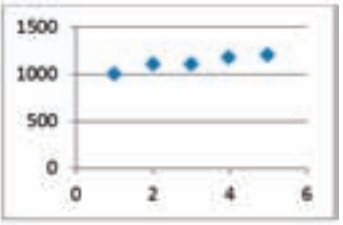

Abbildung 1:
Beispielverlauf CPU-Kosten

Abbildung 1 stellt exemplarisch den Werte-Verlauf einer Beispiel-SQL-Anfrage grafisch dar. Die Grafik zeigt eine stetige Verschlechterung der zu Beginn auf den Wert 1000 normierten CPU-Kosten pro zurück gelieferten Datensatz über die Zeit. Betrachtet man diesen Verlauf, so liegt die Vermutung nahe, dass hier eine Datenbank-Reorganisation sinnvoll und eine Absenkung der CPU-Kosten auf das Startniveau (1000) denkbar ist.

Erste Tests des Prototypen haben ergeben, dass die historisierten Leistungskennzahlen teilweise schwer maschinell auszuwerten sind, da diese von großen Streuungen und Lücken in der Messhistorie geprägt sein können. Im Folgenden werden diese Probleme kurz verdeutlicht. Grobe Streuungen könnten zum Beispiel entstehen, wenn einzelne SQL-Anfragen relativ selten ausgeführt werden. Hier fallen Spitzen in der Datenbankauslastung besonders negativ ins Gewicht, so dass die Kennzahlen dadurch stark verzerrt werden können. Die Verwendung eines Schwellwertes für die minimale Ausführungshäufigkeit konnte Verläufe dieser Art drastisch reduzieren.

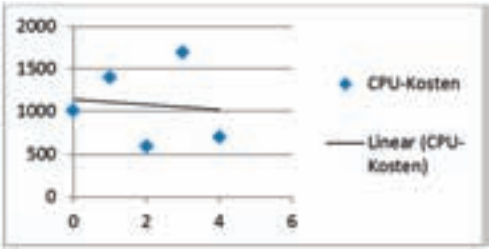

Abbildung 2: Gestreuter Beispielverlauf

Im Falle des Einsatzes der Linearen Regression [BH06] – wie in der Abbildung 2 angewandt – ist des Weiteren die Berechnung der Standardabweichung oder des Bestimmtheitsmaßes von der Trendlinie interessant. Unsere Untersuchungen haben Folgendes gezeigt: wenn die Standardabweichung oder das Bestimmtheitsmaß zu hoch ist, dann sollte die Funktionsannäherung nicht in weitere automatische Auswertungsverfahren einbezogen werden.

Abbildung 3 zeigt eine Messhistorie auf Grundlage von vier Messwerten, die in einem Zeitraum von 14 Tagen gesammelt wurden. Eine Entscheidung auf Grundlage dieser wenigen Messpunkte ist hier sehr fragwürdig. Es wurden geeignete Schwellwerte auf Basis empirischer Untersuchungen festgelegt, um diese Art von Unschärfe nicht mit in eine Entscheidung einfließen zu lassen.

Datenbank-Reorganisationen werden bei den Kunden der Firma InfoDesign in der Regel in einem festgelegten Wartungszeitfenster durchgeführt. Dieses Wartungszeitfenster liegt in der Regel am Wochenende, da zu diesem Zeitpunkt eine deutlich verminderte OLTP-Last anliegt. Die Reorganisationsentscheidung des InfoMat wird aufgrund der statischen Regeln freitags nach Ende der Kernarbeitszeit der Mitarbeiter getroffen.

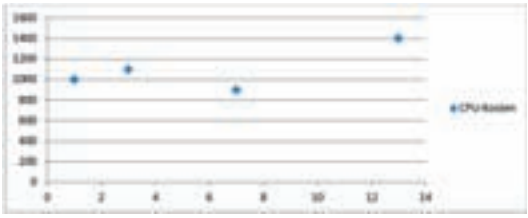
Abbildung 3: Dünn besetzter Beispielverlauf

Welche Zeiträume müssen nun aber mit einem maschinellen Verfahren bewertet werden? Ausgehend davon, dass Messdaten von Montag bis Freitag gesammelt werden, wird nach Ablauf der Kernarbeitszeit am Freitag eine Entscheidung auf Basis der über die Woche gesammelten Werte getroffen. Kommt es zu einer positiven Reorganisations-Entscheidung, kann die Bewertung in der darauf folgenden Woche wieder analog angewendet werden. Wie sieht es aber aus, wenn keine Reorganisation notwendig ist?

Prüft man die Reorganisationsbedürftigkeit des Verlaufs in Abbildung 4 nach Ablauf der ersten Woche, so kommt man zu der Entscheidung, dass keine Reorganisation notwendig ist. Klar erkennbar ist zwar, dass über die Zeit eindeutig eine Verschlechterung der Werte stattfindet, aber die Werte bewegen sich im Rahmen eines Toleranzkorridors. Zur gleichen Entscheidung kommt man, wenn man das zweite Intervall isoliert betrachtet. Prüft man allerdings den Verlauf über den gesamten Zeitraum, so ist schon eine fast 20-prozentige Verschlechterung der Leistungskennzahlen erkennbar, was sicherlich in einer positiven Reorganisations-Entscheidung resultieren sollte. Festzuhalten bleibt also, dass eine Bewertung über den Zeitraum seit dem letzten

Reorganisationszeitpunkt notwendig ist, da eine Bewertung der letzten 5 Werktage allein nicht ausreichend für eine korrekte Bewertung sein kann.

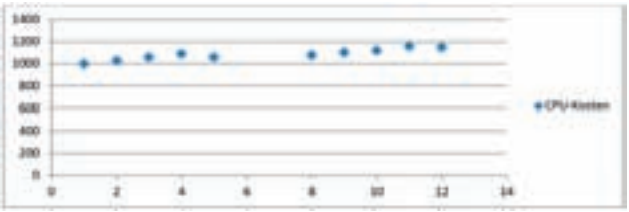

Abbildung 4: Schleichende Verschlechterung eines Beispielverlaufs

Jedoch auch die Untersuchung eines Tablespace über einen längeren Zeitraum stellt ein mathematisch-maschinelles Entscheidungsverfahren vor Probleme. In dem in Abbildung 5 dargestellten Beispiel ist eine Trendlinie eingezeichnet. Lässt man bei der Bewertung wieder einen Toleranzspielraum von knapp 10 Prozent zu (in dem keine Datenbank-Reorganisation notwendig ist), so kommt ein mathematisches Verfahren, das den gesamten Zeitraum berücksichtigt, zu der Entscheidung, dass eine Reorganisation nicht notwendig ist. Dabei ist für den Menschen die drastische Verschlechterung der letzten fünf Messpunkte klar erkennbar. Es muss also ein Algorithmus entwickelt werden, der mehrere Untersuchungszeiträume bewertet.

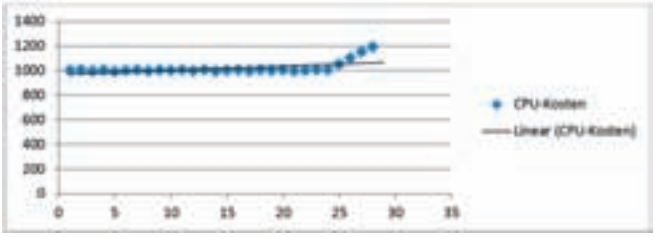

Abbildung 5: Beispiel des Problems bei einer Langzeituntersuchung

Wie im vorherigen Beispiel aufgezeigt, ist eine Untersuchung ab dem letzten Reorganisationspunkt bzw. seit Anbeginn der Messung sinnvoll. Um Schwankungen in der näheren Vergangenheit berücksichtigen zu können, sollen zusätzlich die letzten 5 Tage gesondert bewertet werden. Um ein Patt in den Reorganisationsentscheidungen zu vermeiden, ist die Gewichtung der Regeln sinnvoll, sodass hier Schwankungen aus der nahen Vergangenheit erkannt werden und maßgeblich in die Entscheidung mit eingehen.

Die Regressionsanalyse [BH06] hat sich als ein probates Mittel zur maschinellen Auswertung der historisierten Daten erwiesen, um daraus Prognosen für die weitere Entwicklung abzuleiten. Unser Prototyp hat die Möglichkeit, sowohl die logarithmische wie auch die lineare Variante anzuwenden. Das Bestimmtheitsmaß oder auch die Standardabweichung werden dazu verwendet, die gemessenen Daten daraufhin zu überprüfen, ob die errechnete Regressionsfunktion die Zusammenhänge gut wiedergibt.

Um für einen ersten Prototyp geeignete Schwellwerte für Filter- und Regressionsfunktionen definieren zu können, wurde die Software in einer Alpha-Version einer empirischen Untersuchung auf einem Testsystem unterzogen und die dabei gewonnenen

Ergebnisse wurden als Regeln im Prototyp implementiert. So müssen die folgenden Bedingungen erfüllt sein, um eine Aussage treffen zu können: es müssen mindestens 4 Messpunkte vorliegen; es müssen mindestens 70 Prozent der Messpunkte besetzt sein; jede SQL-Anfrage muss mindestens 100 Mal aufgerufen worden sein; die Standardabweichung darf bei maximal 250 liegen. Des Weiteren erfolgt eine Ausreißerfilterung. Die Schwellwerte wurden über die Zeit hinweg regelmäßig kontrolliert und an Veränderungen und neue Beobachtungen angepasst und optimiert.

Tabelle 6 zeigt zusätzlich, wie die Bewertungszeiträume der linearen und logarithmischen Regression und deren Gewichtung angesetzt wurden.

Name	Verfahren	Bewertungszeitraum	Gewichtung
Regel 1	Logarithmische Regression	Letzter Reorganisationszeitpunkt bis zum aktuellen Zeitpunkt	1
Regel 2	Logarithmische Regression	Letzte fünf Tage	1,3
Regel 3	Lineare Regression	Kompletter Zeitraum	0,5

Tabelle 6: Übersicht der implementierten Regeln

Bisher haben wir skizzenhaft aufgezeigt, wie eine Entscheidung auf SQL-Anfrage-Basis getroffen wird. Letztendlich muss jedoch anhand des Granulats Tablespace entschieden werden. Hierzu werden für jeden Tablespace alle gültigen SQL-Anfragen bewertet. Folgende Beispieltabelle (Tabelle 7) dient dazu, den Algorithmus näher zu erläutern. Zu dem Tablespace T1 wurden Messdaten über zwei SQL-Anfragen S1 und S2 gesammelt und durch die Regeln 1 bis 3 jeweils bewertet. Die Spalten „Prozentuale Entscheidung“ und „Absolute Entscheidung“ geben dabei jeweils das ungewichtete Ergebnis der Regeln an. Das gewichtete Ergebnis befindet sich in den letzten beiden Spalten „Ergebnis Prozentual“ und „Ergebnis Absolut“. Um die Masse an Daten zu reduzieren, wurden die Regel-Einzelentscheidungen auf eine SQL-Anfrage-Entscheidung reduziert, in die dann die bereinigten gewichteten Ergebnisse eingingen. Die beiden SQL-Anfragen S1 und S2 wurden unterschiedlich häufig in dem Beispiel-Untersuchungszeitraum aufgerufen. Während die SQL-Anfrage S1 5000 Mal aufgerufen wurde, wurde S2 lediglich 100 Mal aufgerufen. Wie erläutert, ist eine Gleichbehandlung der SQL-Anfragen nicht sinnvoll.

SQL-Anfrage	Anzahl Aufrufe	Regel	Regel-Gewicht	Prozentuale Leistungs-veränderung	Absolute Leistungs-veränderung	Ergebnis Prozentual	Ergebnis Absolut
S1	5000	Regel 1	1	10	27	10	27
		Regel 2	1,3	14	50	18,2	65
		Regel 3	0,5	8	21	4	10,5
		Gesamt				10,73	34,17
S2	100	Regel 1	1	-10	-2	-10	-2
		Regel 2	1,3	-5	-11	-6,5	-14,3
		Regel 3	0,5	-21	-6	-10,5	-3
		Gesamt				-9	-6,4
GESAMT						10,34	33,37

Tabelle 7: Beispiel zur Tablespace-Entscheidung

Der Unterschied ist leicht nachzuvollziehen. Die Ergebnisse der SQL-Anfragen S1 und S2 sind konträr. Anhand einer gewichteten Überführung der Einzelentscheidung in eine Gesamtentscheidung wird diese eindeutig und einsichtig. Wie beabsichtigt, wird die Gesamtbewertung in diesem Beispiel von der SQL-Anfrage S1 dominiert.

Variable	Beschreibung
$S_{1,T}..S_{N,T}$	SQL-Anfragen des Tablespaces T
$\|S_{K,T}\|$	Anzahl Aufrufe der SQL-Anfrage $S_{K,T}$ mit $1 < k < N$
$R_1..R_M$	Entscheidungsregeln
$F_{Abs}(r,s)$	Funktion zur Berechnung der Regelentscheidung „Ergebnis Absolut“. Das Ergebnis berücksichtigt bereits die Regel-Gewichtung.

Tabelle 8: Parameterübersicht

Das von uns implementierte mathematische Berechnungsverfahren sieht nun wie folgt aus, wobei die einzelnen Parameter in Tabelle 8 aufgelistet und näher erläutert sind. F_{GESAMT} berechnet dabei die Entscheidung für einen Tablespace T:

$$F_{GESAMT}(T) = \frac{\sum_{s=1}^{s=N}(\frac{1}{M}\sum_{r=1}^{r=M}(F_{Abs}(R_r, S_{s,T})) * |\, S_{s,T}\,|)}{\sum_{s=1}^{s=N}|S_{S,T}|}$$

4 Evaluation

Der Prototyp wurde bei der Firma SCHWENK Zement KG in Ulm eingesetzt. Die Firma SCHWENK Zement KG hat im Jahr 2009 mit 2.300 Mitarbeitern einen Umsatz von ca. 700 Millionen Euro erwirtschaftet. Bei der Firma SCHWENK Zement handelt es sich um einen typischen Kunden der Firma InfoDesign. Schwenk setzt mehrere SAP-Systeme weltweit ein, die auf einer DB2-Datenbank auf z/OS aufsetzen. Zur Automatisierung wichtiger administrativer Aufgaben wird der InfoMat eingesetzt.

Die Software wurde dabei testweise in dem Zeitraum von 6 Wochen eingesetzt. Dabei lag die Kernarbeitszeit zwischen 7 und 19 Uhr. Zu jeder vollen Stunde wurde jeweils ein Snapshot erstellt. Freitags beginnt der InfoMat mit der Einplanung der zu reorganisierenden Tablespaces, sodass sich die Bewertung auf die Tage von Montag bis Freitag konzentriert. Die Wochenendtage wurden in der Betrachtung vernachlässigt. Während des Untersuchungszeitraums wurden 39.729 verschiedene reduzierte SQL-Anfragen identifiziert. Bei 35.062 davon handelte es sich um SELECT-Anfragen.

Interessant für die spätere Auswertung ist vor allem, wie häufig und durchgängig diese SQL-Anfragen im DSC vorkommen. Nur noch 5.521 SQL-Anfragen erfüllten die beiden Kriterien, im Durchschnitt mindestens 100 Mal pro Werktag aufgerufen und mindestens an jedem zweiten Tag gemessen worden zu sein. Dabei konnten SQL-Anfragen zu 4.134 Tablespaces identifiziert werden. Insgesamt waren zum Analysezeitpunkt in dem Datenbanksystem 29.300 Tablespaces für Tabellen definiert, sodass damit eine Aussage über lediglich knapp 14 Prozent der Tablespaces getroffen werden konnte. Diese Zahl ist nicht negativ zu interpretieren: Prüft man die Anzahl aller Reorganisationen, die in dem Testzeitraum durchgeführt wurden, so stellt man fest, dass diese bei 2.760 liegt. Diese

2.760 Reorganisationen bestanden allerdings nur aus 1.152 unterschiedlichen Tablespaces. Da der InfoMat mehr Reorganisation als notwendig durchführt, liegt die Schlussfolgerung nahe, dass mit den 4.134 alle entscheidenden Tablespaces identifiziert werden konnten. Zum Stichtag der Auswertung wurden 527 Tablespaces durch den InfoMat zur Reorganisation eingeplant. Über 345 dieser Tablespaces konnte auch der Prototyp eine Aussage treffen. Tabelle 9 zeigt die Entscheidungen des InfoMat. Gegenübergestellt werden die Aussagen des Prototyps (Info-RM).

InfoMat - Entscheidung		Info-RM - Entscheidung			Einspar-potential	Einsparpotential inklusive TS mit unklarerer Aussage
Reorganisation aufgrund Bedingung		Reorganisation notwendig	Reorganisation nicht notwendig	Keine Aussage		
Nicht reorganisieren	3.607	436	417	2.754		
RB02	48	14	12	22	25%	71%
RB03,04,09,12,14	33	14	16	3	49 %	58 %
RB05	28	7	18	3	64%	75%
RB06	26	2	6	18	23%	92%
RB07	168	72	55	41	33%	57%
RB10	224	67	80	77	36%	70%
RB-Gesamt	527	176	169	182	32%	67%

Tabelle 9: Entscheidungstabelle nach dem ersten Messdurchgang

Es wurden zum Stichtag vom InfoMat aufgrund der Reorganisationsbedingung RB10 224 Tablespaces vorgeschlagen. Der Prototyp kommt nach Analyse der Messdaten lediglich auf 67 Tablespaces, die eine Verschlechterung der gemessenen Leistungsparameter aufzeigen. 80 Tablespaces würde der Prototyp definitiv von der Reorganisation ausschließen, da sich die Leistungskennzahlen verbessern oder zumindest konstant bleiben. Über 77 Tablespaces kann der Prototyp keine Aussage treffen, da bspw. nicht genügend Messpunkte während der Testphase gesammelt werden konnten. Die Auswertung aus dem ersten Messdurchgang basiert ausschließlich auf der Entwicklung der Regressionsfunktion. Dabei wurde geprüft, ob es sich um eine steigende oder fallende Regressionsfunktion handelt. Sinnvoll ist es, das Verfahren um einen Toleranzkorridor zu erweitern. Die Argumentation ist dabei, dass die Reorganisation ein aufwendiger Prozess ist, der sehr viel CPU- und I/O-Zeit in Anspruch nimmt, und dieser Mehraufwand in einem Verhältnis zu den zu erzielenden Einsparungen der Reorganisation stehen sollte. Minimale Verschlechterungen sollen dabei in einem gewissen Maße toleriert werden. Bei den Untersuchungen ist aufgefallen, dass die Performance sich bei sehr vielen Tablespaces nur minimal verschlechtert hat, aber eben nicht so drastisch, dass hierdurch eine Reorganisation begründbar ist. Daher wurde für einen zweiten Messdurchgang ein Schwellwert, ab dem ein Tablespace reorganisiert werden sollte, auf den CPU-Kostenwert 1 (in Sekunden) festgelegt. Dieser Wert soll dabei noch nicht als besonders sinnvoller Wert für einen Schwellwert für die Praxis verstanden werden, sondern dieser soll wirklich nur den Toleranzkorridor unter sehr konservativen Annahmen erweitern. Was sagt dieser CPU-Kostenwert aus? Betrachten wir die durchschnittlichen CPU-Kosten, die bei einer SQL-Anfrage anfallen, die exakt einen Datensatz zurückliefert. Dieser Wert

lag in unseren Messungen bei ca. 0,00395. Diese Anfrage – bei jeweiliger Planneugenerierung - könnte man ca. 253 mal ausführen, um 1 CPU-Kosteneinheit zu verbrauchen. Da die Plangenerierung sehr teuer ist und einen Großteil der CPU-Kosten bei einer SQL-Anfrage einnimmt (die bspw. nur wenige Datensätze zurückliefert) wurde geprüft, wie die durchschnittlichen CPU-Kosten bei SQL-Anfragen in der Testumgebung aussehen, die zwischen 10 und 20 Datensätze zurückliefern. Hier liegen die CPU-Kosten pro Datensatz bei ca. 0,001897. Hier könnte man die SQL-Anfrage – bei erneuter jeweiliger Planneugenerierung - immerhin 527 Mal ausführen. Der CPU-Kostenschwellwert von 1 ist also insgesamt als sehr niedrig gewählt zu werten. Unter Nutzung dieses Kostenschwellwerts ergaben sich folgende Ergebnisse eines zweiten Messdurchgangs (Tabelle 10).

InfoMat - Entscheidung		Info-RM - Entscheidung			Einspar-potential	Einsparpotential inklusive TS mit unklarerer Aussage
Reorganisation aufgrund Bedingung		Reorganisation notwendig	Reorganisation nicht notwendig	Keine Aussage		
Nicht reorganisieren	3.607	95	758	2.754		
RB02	48	7	19	22	39%	85%
RB03,04,09,12,14	33	5	25	3	76 %	85 %
RB05	28	2	23	3	82%	93%
RB06	26	0	8	18	31%	100%
RB07	168	21	106	41	63%	88%
RB10	224	20	127	77	57%	91%
RB-Gesamt	527	55	308	164	58%	89%

Tabelle 10: Entscheidungstabelle nach zweitem Messdurchgang

Die Ergebnisse des zweiten Messdurchgangs sprechen für sich. Man kommt insgesamt auf ein Einsparpotential von 58 oder sogar 89 Prozent gegenüber den vom InfoMat vorgeschlagenen Tablespaces. Die Werte erscheinen sehr positiv. Dabei darf man tatsächlich nicht vergessen, dass nicht alle Reorganisationen aus Performancegründen angestoßen werden. Bereits im zweiten Kapitel wurde darauf hingewiesen, dass Datenbank-Reorganisationen auch aus Gründen der Speicherplatz-Ausnutzung bzw. –Rückgewinnung vorgenommen werden. Tatsächlich beziehen sich auch einige der oben aufgeführten Reorganisationsbedingungen auf diese Fälle. Insbesondere aber die Reorganisationsbedingungen RB07 und RB10, die zusammen immerhin 74 Prozent der angestoßenen Reorganisationen ausmachen, sind von der Motivation durch eine Aussicht auf eine Performanceverbesserung begründet.

5 Fazit

Unter geeigneter Auswertung der CPU-Zeiten mittels Regressionsanalyse konnte festgestellt werden, dass zwischen 58 bis 89 Prozent der Reorganisationen, die durch statische Verfahren vorgeschlagen werden, aus Performance-Sicht unnötigerweise durchgeführt werden. Die Überprüfung durch den in diesem Artikel angeregten leistungsorientierten Ansatz der Auswahl der Reorganisationskandidaten bestätigt somit die Vermutung, die

sich in den IT-Abteilungen manifestiert hat. Ein weiteres Ergebnis der vorgenommenen Analyse ist, dass auch Tablespaces, die aus leistungsorientierter Sicht dringend einer Reorganisation bedürfen, nicht notwendigerweise von statischen Verfahren erkannt werden, von einem leistungsorientierten Verfahren jedoch durchaus. Somit konnte empirisch gezeigt werden, dass ein (einfach gestaltetes) leistungsorientiertes Verfahren insgesamt eine weitaus höhere Genauigkeit erzielen kann. Interessant ist in diesem Zusammenhang die Beobachtung, dass die potentiellen Kunden einer solchen Software vorrangig an einer Reorganisationsvermeidung interessiert sind.

Die Firma SCHWENK setzt die Software mittlerweile im produktiven Einsatz ein. Es hat sich dabei erwiesen, dass der in Kapitel 4 beschriebene, durch Vermeidung von unnötigen Reorganisationen erzielte Gewinn dauerhaft erzielt werden kann. Des Weiteren wurden erste Proof of Concepts bei anderen Testkunden im Bereich von OLAP-Systemen durchgeführt. Die Herausforderung lag hier insbesondere in einer Verbesserungen des SQL-Parsers, da die SQL-Anfragen bei einem OLAP-Workload komplexer und somit schwieriger zu parsen sind. Auch hier wurden vergleichbar positive Werte erreicht.

Da die vorgestellte Problematik nicht nur im z/OS-Umfeld existiert, geschieht gegenwärtig eine Portierung der Lösung auf andere Plattformen (wie Oracle und DB2 LUW).

Literatur

[BH06] Backhaus, K.: *Multivariate Analysemethoden eine anwendungsorientierte Einführung*, SpringerLink (Online service), 2006

[CM01] Mullins, C.: The DBA Corner. Database Fragmentation and Disorganization. http://www.craigsmullins.com/dbta_004.htm, Dezember 2001 [Stand: 19.10.2011]

[SG79] Sockut, G.;Goldberg, R.: Database Reorganization – Principles and Practice, in Computing Surveys, Vol. 11, No. 4, Dezember 1979

[SH94] Shallahamer, C.: *Avoiding a Database Reorganization – Understanding, detecting, and eliminating harmful database fragmentation,*
URL: http://www.allenhayden.com/cgi/getdoc.pl?file=reorg.pdf. November 1994 [Stand: 19.10.2011]

[WV09] Weaver, R.: *Database Reorganization Strategies for DB2 z/OS*
URL:http://www.mainframezone.com/it-management/database-reorganization-strategies-for-db2-z-os. Juli 2009 [Stand: 19.10.2011]

[QE10] Ahrends, J.: *Oracle Datenbank Reorganisation*
URL:
http://www.toadworld.com/Portals/0/JohannesA/artikel/Reorganisation%20Teil%201.pdf. Februar 2010 [Stand: 03.01.2013]