Konsistenzquantifizierung in Grid-Datenbanksystemen

Lutz Schlesinger

Universität Erlangen-Nürnberg
Lehrstuhl für Datenbanksysteme
Martensstraße 3
D-91058 Erlangen
schlesinger@informatik.uni-erlangen.de

Wolfgang Lehner

Technische Universität Dresden
Arbeitsgruppe Datenbanken
Dürerstraße 26
D-01069 Dresden
lehner@inf.tu-dresden.de

Kurzfassung: Das Konzept des Grid-Computing gewinnt insbesondere bei verteilten Datenbankanwendungen immer mehr an Bedeutung. Ein Grid-Datenbanksystem besteht dabei aus lokal autonomen Systemen, die für globale Datenbankanfragen zur Ausführung komplexer Analysen zusammengeschaltet werden. Eine Datenbankanfrage wird durch eine Auftragsvergabekomponente an ein Mitglied des Grids weitergeleitet. Um die Anfragegeschwindigkeit zu erhöhen, halten im skizzierten Architekturmodell Knoten die Snapshots von Datenquellen anderer Mitglieder lokal vor. Ist aus Benutzersicht ein Zugriff auf teilweise veraltete Zustände von Datenquellen akzeptierbar, so kann die Auftragsvergabe neben systemtechnischen Eigenschaften (Lastbalancierung, Kommunikationsaufwand) auch inhaltliche Aspekte berücksichtigen, wobei als Voraussetzung die Güte eines lokalen Zustands eines jeden Grid-Mitglieds zu quantifizieren ist.
Der Beitrag führt unterschiedliche Methoden ein, die eine Quantifizierung der Konsistenz eines Datenbankzustands für ein Grid-Mitglied hinsichtlich zeit- und datenbezogener Differenzen im Vergleich zu einer direkten Auswertung auf den Datenquellen erlauben. Es wird diskutiert, wie die auftretenden Differenzen klassifiziert und quantifiziert werden können, um dadurch dem Benutzer ein Gefühl für die Qualität des Ergebnisses und für das Zusammenpassen der Datenzustände unterschiedlichen Alters zu vermitteln. Für den Fall, dass eine Datenquelle zusätzlich eine Historisierung der Daten ermöglicht, werden Verfahren angegeben, die nach Vorgabe eines gewünschten Quantitätsmaßes die passenden Daten aufsuchen und miteinander kombinieren.

1 Einleitung

Die Bereitstellung eines einheitlichen Zugangs zu autonomen Datenquellen kombiniert mit einer effizienten Ausführung von Anfragen ist seit vielen Jahren Gegenstand von Forschung und kommerzieller Entwicklung. Aktuell verschärft sich das Problem durch die Verbreitung von ›*Grid-Computing*‹-organisierten Rechnerstrukturen ([FoKe99]), wobei der Ansatz des Grid-Computings im Wesentlichen darin besteht, dass eine Menge lokaler Rechnersysteme zu einem virtuellen Supercomputer zusammengeschlossen werden und anstehende Aufgaben zentral verwaltet und einzelnen Knoten zur Bearbeitung übergeben werden.

Der Technik des Grid-Computing ist aktuell sowohl allgemein als auch insbesondere aus Sicht der Datenbanksysteme eine große Aufmerksamkeit zu schenken. So drängt zum einen der Einsatz der Grid-Computing-Technologie aus dem (akademisch motivierten) Bereich des wissenschaftlichen Rechnens durch ausgereifte Basis-Technologie (the globus project; http://www.globus.org) immer mehr in den kommerziellen Bereich zur Durchführung umfassender Auswertung innerhalb einzelner Unternehmen (›*enterprise grid*‹). Zum anderen ist eine Transition von einer funktionszentrierten Ausrichtung (›*number crunching*‹) zu einer datenzentrierten Ausrichtung (›*data crunching*‹) auszumachen, die offensichtlicherweise eine umfassende Unterstützung durch eine entsprechende Datenbanktechnologie erfordert ([HoCa01], [RiKT02]). Erste Ansätze werden beispielsweise durch das ›*Database Access and Integration Services*‹-Forum (DAIS; http://www.cs.man.ac.uk/grid-db/) koordiniert.

Spannungsfeld der Grid-Datenbanksysteme

Die Anforderungen an Datenbanktechnologie für Grid-Computing sind vielfältig. So ist beispielsweise trotz der Verwendung offener WebService-Techniken ([Cera02], [Newc02]) im Rahmen der aktuellen ›*Open Grid Services Architecture*‹ (OGSA; http://www.globus.org/ogsa) eine Beschreibung von Datenbankdiensten für eine dynamische Nutzung nur unzulänglich möglich. Weiterhin sind die Möglichkeiten einer intelligenten Auftragsvergabe einer eingehenden Datenbankanfrage an ein oder mehrere Mitglieder des Grid-Datenbanksystems Gebiete, die weitreichende Forschungsmöglichkeiten eröffnen.

Mit Blick auf existierende Technologie existiert im Allgemeinen ein Spannungsfeld (Abbildung 1), in welchem Grid-Datenbanksysteme einzuordnen und entsprechende Technologien zu adaptieren sind.

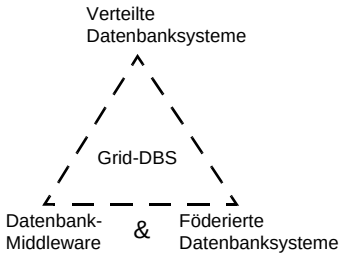

Abb. 1: Klassifikation von Grid-DBS

- *Verteilte Datenbankssysteme*
  Im Bereich der verteilten Datenbanksysteme ([ÖzVa99], [Dada96], [Rahm94]) werden einzelne Datenbankoperationen an partizipierende Systeme verteilt. Probleme der Lastbalancierung und Kostenabschätzung bei der Anfrageausführung (insbes. Kommunikationskosten) und Synchronisierung von Replikaten treten in Grid-Datenbanken erneut auf.
- *Datenbank-Middleware und Föderierte Datenbanksysteme*
  Die Bildung einer homogenen Datenbasis über einer Menge weitgehend autonomer Datenquellen auf technischer Ebene (insbes. DB-Middleware) und auf Ebene der Schema- und Datenintegration findet sich bei der Definition einer Grid-Datenbank in veränderterter bzw. spezialisierter Form wieder. DB-Middleware kann dabei als Basismechanismus dienen ([DiGe00]) und um entsprechende Funktionalität ergänzt werden.

Logisches Architekturmodell des verwendeten Grid-Datenbanksystems

Der vorliegende Beitrag greift die Problematik einer inhaltlich-motivierten Auftragsvergabe einer Datenbankanfrage an ein Mitglied der Grid-Computing organisierten Rechnerstruktur auf. Die Idee besteht darin, die Güte (im Sinne der Konsistenz) eines lokalen Da-

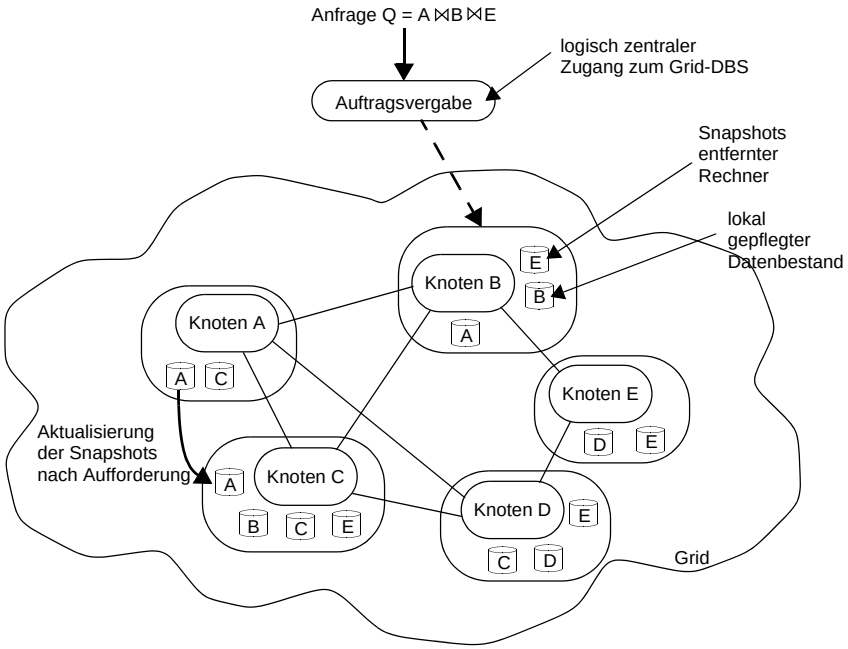

Abb. 2: Logische Architektur des diskutierten Grid-Datenbank-Szenarios

tenbestands quantitativ zu bewerten und damit eine Grundlage für eine geeignete Auswahl eines Rechners zur Auswertung der Anfrage zu schaffen. Dem Verfahren liegt folgendes logische Architekturmodell eines Grid-Datenbanksystems zu Grunde (Abbildung 2):

Das Grid besteht dabei aus einer Menge miteinander verbundener Rechner mit lokal autonomen Datenbanksystemen, die optional einen lokal gepflegten Datenbestand aufweisen können. Eine Anfrage an das System wird von der Auftragsverarbeitung an ein (zuvor ausgewähltes) Mitglied des Grids weitergeleitet, sodass das Prinzip der lokalen Aktualisierung und globalen Analyse von Datenbeständen zu Grunde liegt. Sofern nicht alle benötigten Daten für die Beantwortung einer Anfrage lokal zur Verfügung stehen, werden neue Snapshots bzw. aufgelaufene Änderungen zur Aktualisierung lokaler Snapshots angefordert und lokal zwischengespeichert.

Die Auftragsvergabe hängt dabei (im klassischen Ansatz) in erster Linie von systemtechnischen Parametern wie Auslastung der einzelnen Systeme und des zu erwartenden Kommunikationsaufwands ab. Kann bei einem lesenden Zugriff auf höchste Konsistenz verzichtet werden, indem zu Gunsten einer höheren Ausführungsgeschwindigkeit auf lokal zwischengespeicherte (und möglicherweise veraltete) Datenbestände zugegriffen werden darf, ist die Strategie der Auftragsvergabe durch inhaltliche Eigenschaften (Alter und Umfang der lokal verfügbaren Datenbestände) zu erweitern.

Voraussetzung dafür ist jedoch, dass dem Benutzer bei der Formulierung der Anfrage eine Spezifikation der *maximal zulässigen Inkonsistenz* ermöglicht wird. Des Weiteren muss im Zuge der Ermittlung des günstigsten Grid-Mitglieds lokal eine Berechnung der *zu erwar-*

tenden Inkonsistenz bei der Ausführung der Anfrage auf den lokal verfügbaren Datenbeständen gewährleistet werden. Genau dieser Problematik widmen sich die in diesem Bericht skizzierten Verfahren und Techniken.

Wissenschaftlicher Beitrag der Arbeit

Da die Snapshots, die zu unterschiedlichen Zeiten lokal an einem Grid-Mitglied abgelegt werden, nicht immer den aktuellsten Stand widerspiegeln und darüber hinaus die Daten selbst unterschiedliche Gültigkeitszeitpunkte besitzen, ergeben sich zeit- und datenbezogene Differenzen. Besonders deutlich wird dies beim Verbund, da hier Snapshots mit differierenden Zeitpunkten zusammengeführt werden. Im Rahmen des vorliegenden Beitrages werden die auftretenden Differenzen klassifiziert und quantifiziert sowie Verbundsemantiken diskutiert. Ergebnis ist die Einführung eines ›Fingerabdrucks‹ einer autonomen Datenquelle, an Hand dessen die Änderungscharakteristik abgeschätzt und für die Ermittlung der lokalen Inkonsistenz eines Datenbestands für einer Datenbankanfrage herangezogen werden kann.

Das folgende Kapitel diskutiert die Quantifizierung der Konsistenz aus rein zeitlicher Perspektive. Die orthogonale Richtung der Änderungsrate wird in Kapitel 3 aufgegriffen bevor in Kapitel 4 beide Aspekte zusammengeführt werden. Abgerundet wird der Beitrag durch eine kurze Zusammenfassung.

2 Auswahl von Datenobjekten aus zeitlicher Perspektive

Als erster Parameter für eine Ergebnisquantifizierung und damit für die Charakterisierung einer Griddatenquelle mittels eines Fingerabdrucks wird in diesem Kapitel ausschließlich die zeitliche Perspektive betrachtet ohne auf Zusammenhänge mit anderen Parametern wie dem Änderungsverhalten einzugehen. Dazu bringt Abschnitt 2.1 eine detailliertere Sicht und führt einige Termini ein. Der Quantifizierung der zeitlichen Differenzen widmet sich anschließend Abschnitt 2.2 bevor Abschnitt 2.3 auf eine Verfeinerung mittels Konsistenzbändern eingeht. Zum Abschluss des Kapitels erfolgt noch eine algorithmische Betrachtung der Problematik.

2.1 Problematik, Punktgraph und historischer Schnitt

Wie in Kapitel 1 angedeutet und in Abbildung 3 nochmals verdeutlicht wird ein neues oder geändertes Datenobjekt zunächst in eine operative Datenquelle D_o eingebracht, wodurch Transaktionszeit und Gültigkeitszeit ([SnAh85]) für das Datenobjekt festgelegt werden können. Unabhängig davon wird dann ein Snapshot, der dieses Objekt alleine oder eine Ansammlung von Datenobjekten enthält, von D_o an die konsumierende Datenquelle D_k gesandt, wodurch diesem Snapshot und damit den enthaltenen Datenobjekten ebenfalls eine Transaktions- und eine Gültigkeitszeit zugewiesen werden können. Damit existieren im Gesamtsystem vier unterschiedliche Zeitpunkte, wovon für die Auswertung jedoch nur die Gültigkeitszeit auf Seiten von D_o interessant ist. Besitzt das ursprüngliche geänderte

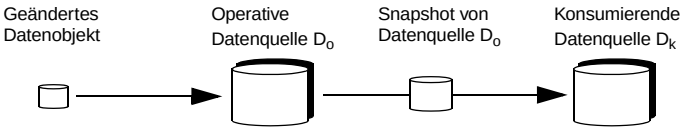

Abb. 3: Veranschaulichung von Datenobjekten und Snapshots im Gesamtsystem

Datenobjekt keine explizit ausgewiesene Gültigkeitszeit bzw. führt es selbst keine Zeitinformation mit, aus der sich die Gültigkeitszeit ergibt, so wird die Gültigkeitszeit gleich der Transaktionszeit von D_o gesetzt, falls auch diese nicht verfügbar ist, gleich der Transaktionszeit bei der Speicherung in D_k.

Datenquellen-Zeit-Diagramm (Punktgraph)

Im zeitlichen Verlauf werden pro operativer Datenquelle mehrere Snapshots erzeugt, sodass eine graphische Veranschaulichung mittels eines *Datenquellen-Zeit-Diagramms*, im Folgenden nur kurz *Punktgraph* genannt, erfolgen kann. Im Punktgraph wird für jede Datenquelle der Gültigkeitszeitpunkt des Snapshots in Form einer Matrix angetragen. So finden sich im Beispiel der Abbildung 4 für die erste Datenquelle ein Snapshot, für die zweite zwei und für die letzte drei, wobei S_n^3 erst später als der aktuelle Zeitpunkt t_{NOW} gültig wird.

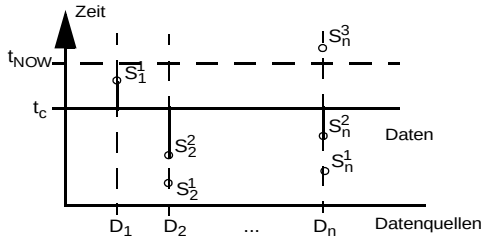

Abb. 4: Beispiel eines Punktgraphen von Gültigkeitszeiten für verschiedene Datenquellen

Historischer Schnitt

Üblicherweise werden die aktuell gültigen Snapshots miteinander kombiniert ohne den Zeitpunkt, ab wann sie gültig geworden sind, zu berücksichtigen. Im Punktgraph sind das diejenigen Snapshots, die auf der Zeitachse von der Vergangenheit her gesehen am nähesten bei t_{NOW} liegen. Im Idealfall besitzen alle Snapshots denselben Gültigkeitszeitpunkt, was sich graphisch durch die zeitlich jüngste horizontale Linie widerspiegelt (Abbildung 5 links). Diese sowie auch jede weitere horziontale Linie wird *historischer Schnitt* t_c genannt.

Im Normalfall werden jedoch die Snapshots zu unterschiedlichen Zeitpunkten gültig, weshalb das Verbinden der jüngsten Snapshots dann nicht mehr zu einer horizontalen Linie (Abbildung 5 rechts) führt. Während eine horizontale Linie ein zeitlich optimales Zusammenpassen der Snapshots reflektiert, ist dies bei einer beliebigen Kurve nicht mehr der Fall. Vielmehr gilt es dann bei der Bestimmung eines historischen Schnittes, um den sich die Snapshots gruppieren, zum einen dem Idealfall einer horizontalen Linie sehr nahe zu kommen, gleichzeitig aber auch möglichst aktuelle Snapshots zu selektieren, um damit einen weitestgehend jungen historischen Schnitt zu erhalten. Die Bewertung und damit die Quantifizierung der Qualität des Ergebnisses der gewählten Snapshots aus zeitlicher Sicht

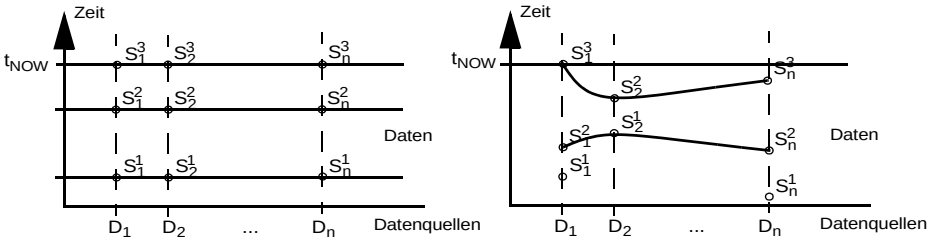

Abb. 5: Idealfall versus Realfall bei der Auswahl von Snapshots

wird im folgenden Abschnitt näher betrachtet, bevor sich die weiteren Abschnitte mit der Bestimmung von Snapshots, die dem Idealfall möglichst nahe kommen, sowie der Verbundsemantik und dem damit erzeugten Fehlern widmen.

2.2 Quantifizierung der Qualität mittels eines Inkonsistenzmaßes

Wird im Datenbankbereich von Konsistenz gesprochen, so bezieht sich dies in der Regel auf die Eigenschaften einer Transaktion im Rahmen des ACID-Prinzips ([ElNa00], [Date99]). Bei der vorliegenden Architektur wird davon ausgegangen, dass auf die Datenquellen nur lesend zugegriffen wird und Datenänderungen lokal durchgeführt werden. Deshalb hat jede Datenquelle für sich die *Datenkonsistenz* aufrecht zu erhalten.

Als weitere Konsistenzart kann im weiteren Sinne die *Zeitkonsistenz* angeführt werden. Diese setzt die Gültigkeitszeitpunkte von je einem Snapshot der verschiedenen Datenquellen in Bezug zueinander, indem der zeitliche Abstand der Snapshots zum historischen Schnitt t_c aufsummiert wird:

$$I = \sum_{i=1}^{k} \left| (\mathrm{time}(S_i^j) - t_c) \right|$$

Für den Idealfall ergibt sich $I = 0$ und je weiter die einzelnen Snapshots von t_c entfernt sind, desto größer wird I. Deshalb ergibt sich die Konsistenz als Qualitätsmaß in Form einer Inkonsistenz.

Treten alle Snapshots mit der gleichen Wahscheinlichkeit auf ($p_1 = p_2 = \ldots = p_k = \frac{1}{k}$), so ergibt sich die mittlere Abweichung, gleichzeitig Erwartungswert bei einer Gleichverteilung der Wahrscheinlichkeiten, zu $\mu = \frac{I}{k}$. Die Standardabweichung berechnet sich damit wie folgt:

$$\sigma^2 = \frac{\sum_{i=1}^{k} (\mathrm{time}(S_i^j) - t_c)^2}{k} - \mu^2$$

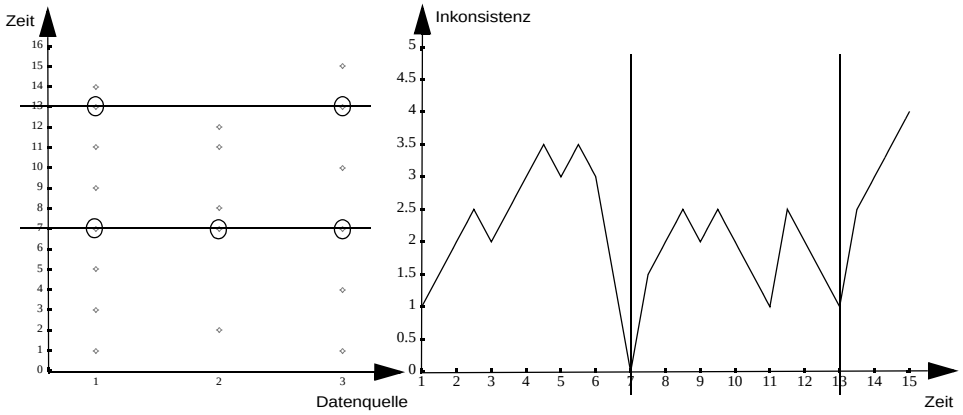

Abb. 6: Beispiel für die Wahl historischer Schnitte und der sich ergebenden Inkonsistenzen

Ein Beispiel zeigt Abbildung 6. Im linken Teil sind für drei Datenquellen Snapshots mit ihren Gültigkeitszeiten zu sehen. Wird zum Zeitpunkt 13 der historische Schnitt genommen und die nächstliegenden Snaphots gewählt, so ergibt sich eine Inkonsistenz vom Wert 1, die in der rechten Graphik angetragen wird. Dieselbe Vorgehensweise wird für jeden Zeitpunkt der linken Graphik angewandt, wodurch sich die Inkonsistenzkurve der rechten Graphik ergibt. Zu erkennen ist eine im Umfeld des Grid-Computing enstehende Problematik beim Vergleich der Inkonsistenzen zum Zeitpunkt 7 und 13: Zum Zeitpunkt 13 ist die Inkonsistenz zwar größer, aber der Schnitt ist zeitlich jünger, während zum Zeitpunkt 7 optimale Konsistenz gegeben ist, die Snapshots aber weitaus älter sind.

2.3 Verfeinerung durch Konsistenzbänder

Werden die gewählten Snapshots mit einem normalen inneren Verbund verknüpft, so werden die zeitlichen Differenzen zwischen den Snapshots und dem historischen Schnitt nicht berücksichtigt. Vielmehr bedeutet diese Verbundart, dass alle Snapshots untereinander als (zeitlich ausreichend) nahe zusammenliegend und damit untereinander als konsistent angesehen werden. Beim Verbund reflektiert werden kann die zeitliche Differenz durch den äußeren Verbund, der auch in föderativen Datenbanksystemen das übliche Mittel zur Integration darstellt ([Conr97]). Allerdings können bei ausschließlicher Anwendung Ergebnisse produziert werden, die keine Aussagekraft mehr besitzen und damit nicht verwendbar sind. Das Pendant dazu stellt die Nicht-Verwendung dar, was zu denselben Ergebnissen wie bei Verwendung des inneren Verbundes führen kann. Deshalb erscheint eine Kombination aus beiden am sinnvollsten.

Der hier vorgestellte Ansatz legt zur Bestimmung der Anwendung des inneren und äußeren Verbundes um den historischen Schnitt ein inneres und äußeres Konsistenzband, wie es in Abbildung 7 zu sehen ist. Das *äußere Konsistenzband* KB_A gibt den maximal zulässigen Abstand eines Snapshots vom historischen Schnitt an. Alle Snapshots, die innerhalb

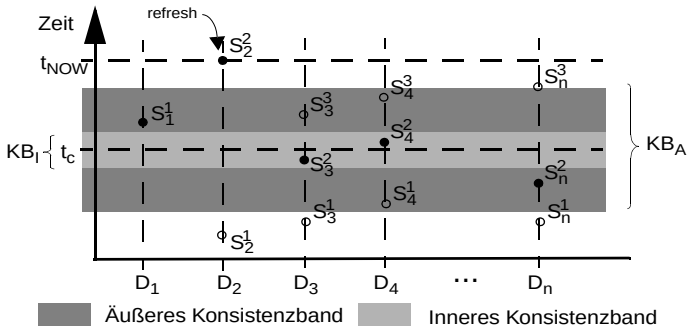

Abb. 7: Beispiel eines inneren und äußeren Konsistenzbandes

des KB_A liegen, werden mittels eines äußeren Verbundes verknüpft. Falls für eine Datenquelle kein Snaphost innerhalb des äußeren Konsistenzbandes liegt (bspw. S_2^1 in Abbildung 7), so muss eine Sonderbehandlung durchgeführt werden. Beispielsweise ist eine Erneuerung des Snapshots mit aktuellen Daten (refresh, Abbildung 7) denkbar.

Liegen die Snapshots innerhalb des *inneren Konsistenzbandes* KB_I, so werden diese mittels eines inneren Verbundes zusammengeführt. Durch KB_I wird der Tatsache Rechnung getragen, dass Snapshots als zeitlich konsistent angesehen werden, auch wenn eine kleine, aber aus globaler Sicht vernachlässigbare Abweichung der Gültigkeitszeiten vorhanden ist. In Abbildung 7 ist dies beipielsweise für S_3^2 und S_4^2 der Fall.

2.4 Algorithmus zur Bestimmung des historischen Schnittes

Einen einfachen, heuristischen Algorithmus zur Bestimmung des historischen Schnittes und damit zur Auswahl der Snapshots unter Berücksichtigung von innerem und äußerem Konsistenzband zeigt Anhang A1. Bezüglich des Beispiels aus Abbildung 6 würde das Ergebnis zum Zeitpunkt 13 erzeugt werden bei Annahme eines äußeren Konsistenzbandes von 8 Zeiteinheiten symmetrisch um t_c. Weiterführende Algorithmen werden in [ScLe02] diskutiert.

3 Datenänderungsrate

Nachdem in Kapitel 2 eine ausschließliche Fokussierung des Problembereichs aus zeitlicher Perspektive erfolgt, wird in diesem Kapitel orthogonal dazu die Änderungsrate betrachtet. Im Folgenden wird deshalb zunächst eine Klassifizierung der Änderungsrate vorgenommen (Abschnitt 3.1) bevor anschließend auf die Berechnung der Änderungsrate eingegangen wird (Abschnitt 3.2).

3.1 Klassifizierung der Änderungsrate

Die Änderungsrate kann prinzipiell von einem Snapshot zum anderen bzw. für eine Datenquelle insgesamt konstant sein, sich nach einer Verteilung oder Funktion richten oder völlig willkürlich sein. Zur weiteren, in diesem Beitrag verwendeten Klassifizierung wird angelehnt an [TCG+93] zunächst eine Einteilung der Datenquellen in zwei Klassen vorgenommen:

- *Zustandsreflektierende Quelle*
  Bei diesem Typ wird ausschließlich der Wert eines Datenobjektes zum Snapshotzeitpunkt festgehalten, wie es in Abbildung 8 angedeutet ist. An die konsumierende Datenquelle wird das Tripel *(ObjektID, Zeitpunkt, Wert)* weitergegeben.

- *Historienbewahrende Quelle*
  Das Charakteristikum dieses Quellentyps ist, dass der Werteverlauf über die Zeit aufgehoben wird, sodass für ein Datenobjekt mehre Zeitpunkte mit einem Wert existieren (Abbildung 8). Damit besteht eine Snapshot für ein Objekt aus dem Tripel *(ObjektID, (Zeitpunkt, Wert)*)*.

Eine Verfeinerung dieser Basisklassifikation der Datenquellen kann durch Betrachtung des Änderungsintervalls der Datenobjekte erreicht werden. Bei einem regulären Änderungsintervall ist der Abstand zwischen zwei Änderungen immer konstant, bei einem irregulären Änderungsintervall kann er variieren.

Ferner ist zu unterscheiden, ob es sich bei den Änderungen um existenzielle oder wertebasierte Änderungen handelt. Unter einer existenziellen Änderung wird das Erzeugen oder Löschen eines Objektes verstanden, während wertebasierte Änderungen ein bestehendes Objekt modifizieren.

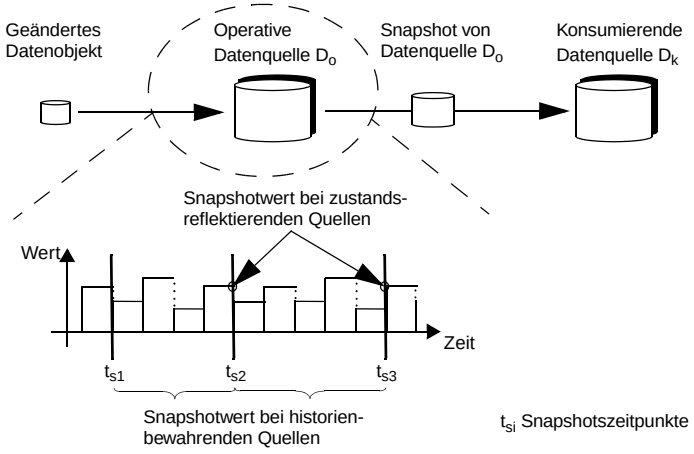

Abb. 8: Klassifizierung der Datenquellen

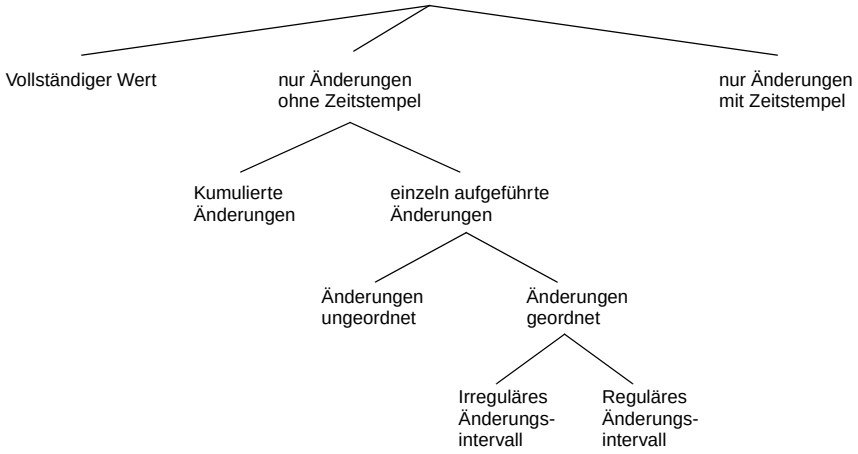

Abb. 9: Arten der Progapierung von Änderungen

Die Modifikationen selbst können in unterschiedlichen Formen propagiert werden, unabhängig davon, ob es sich um existenzielle oder wertebasierte Änderungen handelt. Eine Klassifizierung ist Abbildung 9 zu entnehmen und wird im folgenden erläutert:

- *Vollständiger Wert*
  Zu den einzelnen Zeitpunkten wird immer der jeweils gültige Wert bereitgestellt.
- *Nur Änderungen ohne Zeitstempel*
  Hierbei sind die Modifkationsoperationen, seien sie existenziell im Sinne eines Einfügens (Insert) oder Löschens (Delete) oder seien sie wertebasiert im Sinne einer wertmäßigen Änderung (Update), ohne den Zeitpunkt ihrer Änderung aufgeführt. Eine weitere Unterscheidung kann dahingehend getroffen werden, ob Operationen auf dem selben Objekt zu einer absoluten Änderung zusammengefasst werden können (kumulierte Änderung) oder ob sie einzeln aufgeführt sind. Im letzten Fall können die Änderungen ungeordnet oder geordnet vorliegen. Ist die Datenquelle durch ein reguläres Änderungsintervall charakterisiert, so kann darüber hinaus bei Vorliegen einer Ordnung unter den Änderungen zusammen mit einem Anfangs- oder Endzeitpunkt jede Änderung einem exakten Zeitpunkt zugeordnet werden.
- *Nur Änderungen mit Zeitstempel*
  Dieser Fall ist ähnlich gelagert zum vorhergehenden mit dem Unterschied, dass jede Änderung mit einem Zeitstempel (Gültigkeitszeitpunkt) versehen ist. Dadurch entfällt die vormals durchgeführte weitere Klassifikation.

Werden die durchgeführten Klassifkationen zusammengeführt, so ergibt sich, dass alle Kombinationen zulässig und sinnvoll sind. Anzumerken ist, dass für Objekte, die zum Abzugszeitpunkt nicht mehr existieren, noch wertebasierte Änderungsoperationen vorliegen können. Diese werden einfach nicht weiter beachtet.

Verbund von Snapshots

Werden Snapshots gleichen oder unterschiedlichen Typs miteinander verbunden, so sollte der Abgeschlossenheit wegen das Ergebnis von einem der angeführten Typen sein. Dies ist, wie Tabelle 3.1 zeigt, erfüllt. Damit ist neben der Kommutativität beim Verbund von Relationen auch deren Assoziativität bzgl. der Typen indirekt nachgewiesen.

	historienbewahrend	**zustandsreflektierend**
historienbewahrend	historienbewahrend	historienbewahrend
zustandsreflektierend	historienbewahrend	zustandsreflektierend

Tab. 3.1: Typen nach einem Verbund

Neben den entstehenden Datentypen ist das erzeugte Zeitraster interessant, was Tabelle 3.2 aufführt. Die Fälle, in den zwei Quellen gleichen Typs verbunden werden, und die gemäß der Tabelle 3.2 eine Mischung beider Raster erfordern, ist beispielhaft in Abbildung 10 skizziert.

	historienbewahrend	**zustandsreflektierend**
historienbewahrend	Mischung beider Raster	Raster der historien-bewahrenden Quelle
zustandsreflektierend	Raster der historien-bewahrenden Quelle	Mischung beider Raster

Tab. 3.2: Zeitraster nach einem Verbund

3.2 Berechnung der Änderungsrate

Auf Seite der konsumierenden Datenquelle treffen entsprechend der vorgenommenen Klassifikation aus Abschnitt 3.1 entweder einzelne Snapshots oder nur die Änderungsoperationen ein, aus denen für eine Charakterisierung der Datenquelle eine Änderungsrate zu berechnen ist.

Der Prozentsatz der Einfüge-, Lösch- oder Änderungsoperationen im Verhältnis zur Ausgangsgröße zwischen zwei Zeitpunkten wird Änderungsrate genannt, wobei es je eine für existenzielle und eine für wertebasierte gibt. Wird der zeitliche Abstand normiert, so kann

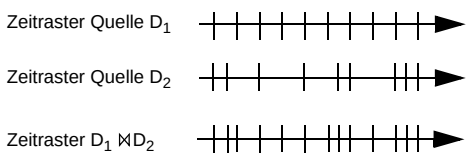

Abb. 10: Entstehendes Zeitraster beim Mischen

die Änderungsrate konstant sein, sich nach einer Funktion oder Verteilung richten oder völlig irregulär sein. Im Rahmen des Beitrags wird von einer konstanten Änderungsrate ausgegangen, die entweder von der Datenquelle geliefert wird oder wie folgt berechnet werden kann:

- *Vorliegen von Änderungsdeltas*
  In diesem Fall liegt die Zahl der Änderungsoperationen n_i^d innerhalb des Intervalls Δt_i explizit vor und die Änderungsrate $\Delta d_i^{\Delta t}$ kann direkt berechnet werden:

$$\Delta d_i^{\Delta t} = \frac{n_i^d}{\Delta t_i}$$

- *Vorliegen von Snapshots*
  Seien für eine Datenquelle i zwei Snapshots zum Zeitpunkt t_i^j und t_i^k mit o.B.d.A. $t_i^j \leq t_i^k$ gegeben und sei ferner $PKA(S_i^l)$ eine Funktion, die die Instanzen der Primärschlüsselattribute des Snapshots der Quelle i zum Zeitpunkt l zurückgibt, so gilt für die jeweilige relative Anzahl an
  - Einfügungen: $ins_i^{jk} = PKA(S_i^k) - PKA(S_i^j)$
  - Löschungen: $del_i^{jk} = PKA(S_i^j) - PKA(S_i^k)$
  - Änderungen: $upd_i^{jk} = PKA(S_i^j - S_i^k) \cap PKA(S_i^k - S_i^j)$

  Somit ergibt sich die Änderungsrate $\Delta d_i^{\Delta t}$ mit $\Delta t_i = t_i^k - t_i^j$ zu

$$\Delta d_i^{\Delta t} = \frac{ins_i^{jk} + del_i^{jk} + upd_i^{jk}}{\Delta t_i}$$

Anzumerken ist, dass durch die angegebenen Operationen im zweiten Fall Kompensationsoperationen (Einfügen und Löschen desselben Objektes) oder Mehrfachoperationen auf einem Objekt (Ändern eines Attributwertes) nicht bemerkt werden, wodurch die Änderungsrate eine andere ist im Vergleich zum ersten Fall, sofern dort keine kompensierenden Operationen verwendet werden. Zur Erreichung derselben Änderungsrate müssen im ersten Fall absolute Änderungsdeltas erzeugt werden. Um schließlich in beiden Fällen eine konstante Änderungsrate Δd_i pro Zeiteinheit zu erreichen, wird von einem Mittelwert ausgegangen, der nach einer Einschwingphase des Gesamtsystems erreicht werden kann. Dieser Wert kann sowohl für jede Operation als auch nach Zusammenfassung als Gesamtänderungsrate angegeben werden, wobei eine Gewichtung der einzelnen Änderungswerte denkbar ist.

Eine Verfeinerung der bisherigen Diskussion erfolgt durch Betrachtung der Zeiteinheiten zwischen zwei Snapshots. Obwohl die zeitliche Differenz zwischen zwei Snapshots beliebig sein kann, wird die Änderungsrate bisher nur auf eine Zeiteinheit bezogen. Wird ferner berücksichtigt, dass bereits geänderte Objekte oder Attribute wieder geändert werden können, ein Teil zum ersten Mal geändert wird und der Rest unverändert bleibt, so ergeben sich die Anzahl der geänderten Tupel Δn_i bei t Zeiteinheit mit k als Zahl der Ausgangselemente und Δd_i der festen Änderungsrate zwischen zwei Zeiteinheiten zu $\Delta n_i = k - (k \bullet (1 - \Delta d_i)^t)$. Angenommen wird, dass im Mittel die Zahl der Tupel erhalten bleibt.

4 Kombination von zeitlicher Perspektive und Änderungsrate

Während sich das Kapitel 2 der Auswahl von Datenobjekten ausschließlich aus zeitlicher Perspektive widmet und im letzten Kapitel alleine die Bestimmung der Änderungsrate diskutiert wird, erfolgt jetzt eine Kombination von beiden Aspekten zum gewünschten Fingerabdruck einer Datenquelle. Dazu führt der Abschnitt 4.1 den Wiederverwendungsgrad ein. Der Quantifizierung der Inkonsistenz widmet sich anschließend Abschnitt 4.2 bevor im letzten Abschnitt auf einen Algorithmus zur Bestimmung des historischen Schnittes eingegangen wird.

4.1 Wiederverwendungsgrad ρ

Um das Alter eines Snapshots mit der Änderungsrate zusammenzuführen, wird die Wiederverwendbarkeit desselben betrachtet, die sich aus der Änderungsrate ergibt. Denn ist die Änderungsrate gering, so können auch alte Snapshots verwendet werden, ohne die Qualität des Endergebnisses zu stark zu beeinträchtigen, da der Unterschied zwischen alten und neuen Snapshots gering ist. Bei hoher Änderungsrate verhält es sich genau umgekehrt, da ein nicht mehr ganz aktueller Snapshot das Ergebnis stark beeinträchtigt. Damit kann der *Wiederverwendungsgrad* ρ in Abhängigkeit von der Änderungsrate Δd_i jedoch noch ohne Berücksichtigung des Alters wie folgt eingeführt werden: $\rho(\Delta d_i) = -\Delta d_i + 1$.

Bei der Erweiterung dieser Basisformel um den angesprochenen temporalen Aspekt muss beachtet werden, dass unabhängig von der Änderungsrate der *absolute Wiederverwendungsgrad* bei Snapshots, die zum aktuellen Zeitpunkt t_{NOW} erstellt wurden, 100% beträgt. Während t_{NOW} die eine zeitliche Grenze bildet, ist das Alter (also die Gültigkeitszeit) des ältesten Snapshots (t_S) die andere Grenze. Damit ergibt sich folgende Funktion, wofür einige Beispielgraphen in Abbildung 11 zu sehen sind. Zu erkennen ist, dass die Kurven symmetrisch zu ρ(0.5) sind:

$$0 < \Delta d_i \le 0,5 : \rho(t) = \frac{-1}{\left|t - (t_{NOW} - t_S) - 1\right|^{\frac{1}{2\Delta d_i} - 1}} \bullet \left(\frac{1}{t_{NOW} - t_S} \bullet (t - (t_{NOW} - t_S)) + 1\right) + 1$$

$$0,5 \le \Delta d_i < 1 : \rho(t) = \frac{1}{(t+1)^{\frac{-1}{2(\Delta d_i - 1)} - 1}} \bullet \left(\frac{-1}{t_{NOW} - t_S} \bullet t + 1\right)$$

Offensichtlich können an Stelle von t_{NOW} und t_S auch andere Zeitpunkte t_E und t_A ($t_E \ge t_A$) als End- und Anfangszeitpunkt definiert werden, wodurch ein *relativer Wiederverwendungsgrad* gegeben ist. Falls $t_E = t_c$ ist, so ist der Wiederverwendungsgrad eines Snaps-

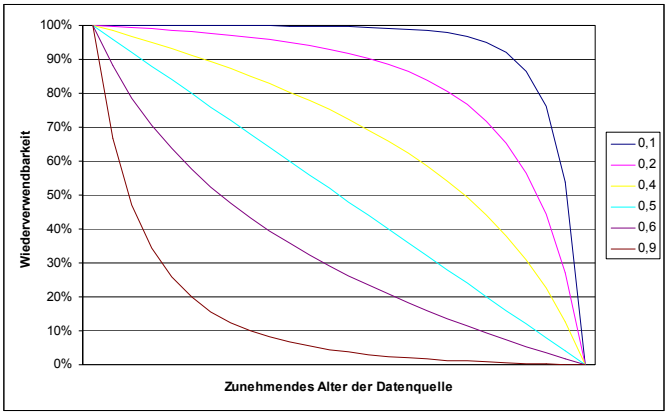

Abb. 11: Zusammenhang zwischen Wiederverwendungsgrad und Alter

hots, der jünger als t_E ist, 100%. Für Werte, die älter als t_c sind, gibt es somit zwei unterschiedliche Werte, einen absoluten und einen relativen Wiederverwendungsgrad, auf die im folgenden Abschnitt näher eingegangen wird.

4.2 Maß für die Inkonsistenz

Die Aufstellung eines Inkonsistenzmaßes analog zu Abschnitt 2.2 muss sowohl die zeitliche Dimension, als auch die Änderungsrate respektive den Wiederverwendungsgrad berücksichtigen. Weiterhin ist zu differenzieren, ob ein Konsistenzmaß bezüglich des aktuellen Zeitpunktes t_{NOW} (*Aktualitätsbezogenes Konsistenzmaß*, I_A) oder des Zeitpunktes des historischen Schnitts t_c (*Historienbezogenes Konsistenzmaß*, I_H) definiert wird.

Aktualitätsbezogenes Konsistenzmaß I_A

Um ein absolutes Maß für die Qualität des Ergebnisses bezogen auf den aktuellen Zeitpunkt t_{NOW} zu erhalten, ist der Endzeitpunkt t_E des Wiederverwendungsgrades gleich t_{NOW} zu setzen. Dadurch wird der Wiederverwendungsgrad jedes Snapshots in ein absolutes Verhältnis zum aktuellen Zeitpunkt gesetzt, wodurch eine Bewertung der verwendeten Snapshots bezogen auf t_{NOW} erfolgen kann.

Historienbezogenes Konsistenzmaß I_H

Durch das Setzen von t_E auf den Zeitpunkt des historischen Schnittes t_c werden die Snapshots in Bezug zu t_c gesetzt, wodurch eine Bewertung in Form eines Konsistenzmaßes bezogen auf den Abstand zu t_c erfolgt. Indirekt ist damit ein Maß gegeben, wie gut die einzelnen Snapshots zusammenpassen. So können, um nur ein Beispiel zu nennen, alle ausgewählten historischen Schnitte denselben Zeitpunkt besitzen wie t_c, wodurch $I_H = 0$ ist, jedoch könnte $I_A \neq 0$ sein, sofern $t_c \neq t_{NOW}$ ist.

Berechnung des Inkonsistenzmaßes I

Der Unterschied zwischen dem aktualitätsbezogenem und dem historienbezogenem Inkonsistenzmaß liegt nur in der Festlegung des Endzeitpunktes. Des Weiteren muss die Formel für das Inkonsistenzmaß berücksichtigen, dass bei 100-prozentiger Wiederverwendbarkeit genauso wie bei einem zeitlichen Abstand von Null der Snapshots zu t_c oder t_{NOW} minimale Inkonsistenz erzielt wird, wobei als Minimum Null festgelegt wird, da negative Werte keinen Sinn ergeben. Damit ergibt sich der Zusammenhang zwischen dem Wiederverwendungsgrad und der Zeit für eine Datenquelle i und dem zeitlichen Abstand Δt_i zwischen historischem Schnitt und Zeitpunkt des Snapshots time(S_i^j) wie folgt:

$$I_i(time(S_i^j)) = \Delta t_i \bullet (1 - \rho_i(time(S_i^j)))$$

Graphisch kann dies im Punktgraph dadurch veranschaulicht werden, dass die Linie zwischen einem Snapshot und t_c nicht mehr die Breite 0, sondern die Breite 1-ρ_i(time(S_i^j)) hat. Die Fläche spiegelt dann die Inkonsistenz wider, wie es in Abbildung 12 als Erweiterung von Abbildung 4 zu sehen ist.

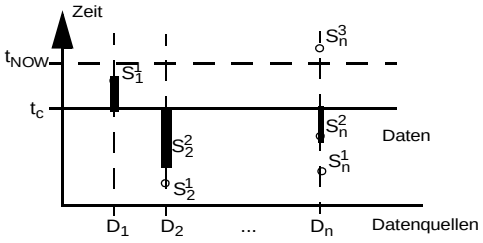

Abb. 12: Beispiel eines Punktgraphen von Gültigkeitszeiten für verschiedene Datenquellen

Eine Gewichtung der eingehenden Faktoren Änderungsrate und Wiederverwendungsgrad ist denkbar. Des Weiteren seien zur logischen Verifizierung der Formel die Fälle betrachtet, in denen die Zeit den Wert 0 oder der Wiederverwendungsgrad den Wert 0 oder 100 annimmt.

- *$\rho = 100, \Delta t = 0$ und $\rho \neq 0, \Delta t = 0$*
  Da die Snapshotzeit mit dem Zeitpunkt des historischen Schnitts übereinstimmt, ist unabhängig von dem Wiederverwendungsgrad der Snapshot bezüglich des historischen Schnittes immer der aktuellste und das Inkonsistenzmaß somit Null.
- *$\rho = 0, \Delta t \neq 0$*
  Der Snapshot ist zwar älter als der historische Schnitt, doch da keine Änderungen vorliegen, wäre der Snapshot auch zum Zeitpunkt des historischen Schnittes derselbe. Damit ist das Konsistenzmaß von Null gerechtfertigt.

Sind wiederum n Datenquellen beteiligt, so ergibt sich das Gesamtinkonsistenzmaß durch Summation der Einzelkonsistenzmaße, wobei die $S_k^{j_k}$ die gewählten Snapshots sind und durch α_k eine Gewichtung der Einzelkonsistenzmaße vorgenommen werden kann:

$$I(time(S_1^{j_1}), \ldots, time(S_k^{j_k})) = \sum_{k=1}^{n} (\alpha_k \bullet I_k(time(S_k^{j_k}))) = \sum_{k=1}^{n} (\alpha_k \bullet \Delta t_k \bullet (1 - \rho_k(time(S_k^{j_k}))))$$

4.3 Algorithmus zur Bestimmung des historischen Schnittes

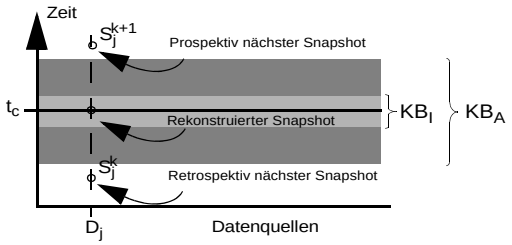

Abb. 13: Veranschaulichung der Snapshotsauswahl

Der in Abschnitt 2.4 vorgestellte Algorithmus zur Auswahl eines historischen Schnittes und damit von Snapshots muss so erweitert werden, dass der in den letzten beiden Kapiteln diskutierte Datenaspekt Berücksichtigung findet. Dazu wird zum ersten die einfache Inkonsistenzformel durch die in Abschnitt 4.2 eingeführte ersetzt. Des Weiteren ermöglicht das Wissen um die Art der Datenquelle eine weiterführende Auswahlmöglichkeit bei der Behandlung von nicht-existierenden Snapshots innerhalb des äußeren Konsistenzbandes. Während bei dem in Abschnitt 2.4 diskutierten Algorithmus nur die Möglichkeit besteht, einen neuen Snapshot von der operativen Datenquelle anzufordern und den Auswahlprozess von neuem zu starten, kann jetzt ein Snapshot zum Zeitpunkt t_c rekonstruiert werden. Dabei kann der exakt vorliegende Wert verwendet werden, sofern die Datenquelle die Änderungen mit einem Zeitstempel versieht. Ist dies nicht gegeben, so kann über die Zahl der Änderung und das Wissen des Änderungsintervalls oder durch Errechnung eines Wertes aus dem des retrospektiv und prospektiv nächsten Snapshots bezüglich t_c ebenfalls ein Snapshot rekonstruiert werden (Abbildung 13).

Den erweiterten Algorithmus zur Bestimmung des historischen Schnittes zeigt Anhang A2, wobei zur leichteren Lesbarkeit eine Vereinfachung dahingehend vorgenommen wird, dass auf die Aktualisierungsoperation verzichtet wird.

5 Zusammenfassung

Beim Konzept des Grid-Computing hält ein Mitglied des Grids neben lokal gepflegten Datenbeständen auch Snapshots der Datenbestände von anderen Mitgliedern aus dem Grid. Im Rahmen der Auftragsvergabe wird jedes Mitglied über seinen vorliegenden Datenbestand befragt, wobei neben systemtechnischen Eigenschaften die Güte des lokalen Datenbestandes die Auftragsvergabe beinflussen kann. Der Quantifizierung der Güte widmet sich der vorliegende Beitrag, wobei nach einer kurzen Einleitung (Kapitel 1) in drei Schritten vorgegangen wird: Nach einer eingehenden Betrachtung der Güte allein aus zeitlicher Perspektive (Kapitel 2) wird die dazu orthogonal liegende Änderungsrate (Kapitel 3) diskutiert. In Kapitel 4 schließlich werden beide Aspekte in Form eines Fingerabdrucks zusammengeführt, der die Güte der Datenquelle widerspiegelt.

Die vorgestellten Ideen werden im Rahmen des Forschungsprojektes SCINTRA (Semi-Consistent Timeorientied Replication Aspect) realisiert, wobei relationale Datenbanksysteme die Mitglieder des Grids repräsentieren.

Anhang

A 1: Algorithmus SimpleDeterminationHistoricCut

Algorithmus zur Bestimmung von historischem Schnitt und zugehörigen Snapshots

```
Algorithmus SimpleDeterminationHistoricCut

Input:      D       // Menge von Datenquellen D_i jede mit Snapshots S_i^j
            KB_A    // Äußeres Konsistenzband
Output:     t_c     // Historischer Schnitt
            S       // Menge an Snapshots


Begin
    Do
        allSnapshotsNotInKBA = true
        t_new = ∞
        I_new = ∞
        S_new = [∞,∞,...,∞]
        Do
            t_c = -∞
            t_old = t_new
            d_old = d_new
            S_old = S_new
            // finde nächsten Snapshot: dessen Zeitpunkt ist
            // der nächste historische Schnitt
            Foreach D_i in D
                Foreach S_i^j in D_i
                    If ( time( S_i^j ) < t_old And time( S_i^j ) > t_c )
                        t_c = time( S_i^j )
                    End If
                End foReach
            End Foreach
            // berechne für jede Datenquelle den zu t_new nächstliegenden Snapshot
            Foreach D_i In D
                S_i^s = ∞
                Foreach S_i^j In D_i
                    If ( |time( S_i^j ) - t_c| ≤ |time( S_i^s ) - t_c| && time( S_i^s ) < t_NOW )
                        S_i^s = S_i^j
                    End If
                End Foreach
                t_new = t_c
                S_new[i] = S_i^s
                I_new = I( t_new, S_new ) // Funktion zur Berechnung der neuen Inkonsistenz
            End Foreach
        While ( I_new < I_old )
            // überprüfe, ob alle Snapshots innerhalb von KB_A liegen
        Foreach S_i^j In  S_old
            allSnapshotsNotInKBA = false
            If ( | time( S_i^j ) - t_c| > ( t_c + KB_A/2 ) )
                refresh( D_i )
                allSnapshotsNotInKBA = true
            End If
        End Foreach
    While ( allSnapshotsNotInKBA )
    Return ( t_old, S_old )
End
```

A2: Algorithmus DeterminationHistoricCut

Erweiterter Algorithmus zur Bestimmung von historischem Schnitt und zugehörigen Snapshots

```
Algorithmus DeterminationHistoricCut

Input:      D       // Menge von Datenquellen D_i jede mit Snapshots S_i^j
            KB_A    // Äußeres Konsistenzband
Output:     t_c     // Historischer Schnitt
            S       // Menge an Snapshots


Begin
    t_new = ∞
    I_new = ∞
    S_new = [∞,∞,...,∞]
    Do
        t_c = -∞
        t_old = t_new
        d_old = d_new
        S_old = S_new
        // finde nächsten Snapshot: dessen Zeitpunkt ist
        // der nächste historische Schnitt
        Foreach D_i In D
            Foreach S_i^j In D_i
                If ( time( S_i^j ) < t_old And time( S_i^j ) > t_c )
                    t_c = time( S_i^j )
                End If
            End Foreach
        End Foreach
        // berechne für jede Datenquelle den zu t_new nächstliegenden Snapshot
        Foreach D_i In D
            S_i^s = ∞
            Foreach S_i^j In D_i
                If ( |time( S_i^j ) - t_c| ≤ |time( S_i^s ) - t_c| && time( S_i^s ) < t_NOW ) )
                    S_i^s = S_i^j
                End If
            End Foreach
            t_new = t_c
            S_new[i] = S_i^s
            I_new = I( t_new, S_new ) // Funktion zur Berechnung der neuen Inkonsistenz
        End Foreach
        While ( I_new < I_old )
            // überprüfe, ob alle Snapshots innerhalb von KB_A liegen
        Foreach S_i^j In  S_old
            If ( | time( S_i^j ) - t_c| > ( t_c + KB_A/2 ) )
                S_i^j = computeSnapshot ( D_i, t_c ) // Berechnung des Snapshots bei t_c
            End If
        End Foreach
    Return ( t_old, S_old )
End
```

Literatur

Cera02 Cerami, E.: *Web Services Essentials.* O'Reilly, Cambridge u.a., 2002

Conr97 Conrad, S.: *Föderierte Datenbanksysteme*. Springer-Verlag, Berlin u.a., 1997

Dada96 Dadam, P: *Verteilte Datenbanken und Client-Server-Systeme. Grundlagen, Konzepte und Realisierungsformen*. Springer-Verlag, Berlin u.a., 1996

Date99 Date, C.J.: *An Introduction to Database Systems*. Addison-Wesley, Reading (MA) u.a., 1999

DiGe00 Dittrich, K.R.; Geppert, A. (Hrsg.): *Component Database Systems*. Morgan-Kaufmann-Verlag, San Francisco (CA) u.a., 2000

ElNa00 Elmasri, R.A.; Navathe, S.B.: *Fundamentals of Database Systems*. Addison-Wesley, Reading (MA) u.a., 2000

FoKe99 Foster, T.; Kesselman, C. (Hrsg.): *The Grid: Blueprint for a New Computing Infrastructure*. Morgan-Kaufmann-Verlag, San Francisco (CA) u.a., 1999

HoCa01 Hoschek, W.; McCance, G.: Grid Enabled Relational Database Middleware, Informational Document. In: *Global Grid Forum (Frascati, Italien, 7.-10. Okt.)*, 2001

Newc02 Newcomer, E.: *Understanding Web Services: XML, WSDL, SOAP, and UDDI*. Addison Wesley, Boston u.a., 2002

ÖzVa99 Özsu, M.T; Valduriez, P.: *Principles of Distributed Database Dystems*. Prentice Hall, Englewood Cliffs, 1999

Rahm94 Rahm, E.: *Mehrrechner-Datenbanksysteme*. Addison Wesley, Bonn u.a., 1994

RiKT02 Risch, T.; Koparanova, M.; Thide, B.: High-Performance GRID Database Manager for Scientific Data. In: *Proceedings of the Workshop on Distributed Data & Structures (WDAS 2002, University Paris 9 Dauphine, Frankreich, 20.-23. März)*, 2002

ScLe02 Schlesinger, L.: Lehner, W.: Extending Data Warehouses by Semi-Consistent Database Views. In: *Proceedings of the 4th International Workshop on Design an Management of Data Warehouses (DMDW'02, Toronto, Canada, 27. Mai)*, 2002, S. 43-51

SnAh85 Snodgrass, R.T.; Ahn, I.: A Taxonomy of Time in Databases. In: *Proceedings of the 1985 ACM SIGMOD International Conference on Management of Data (SIGMOD'85, Austin, Texas, 28.-31. Mai)*, 1985, S. 236-246

TCG+93 Tansel, A. U.; Clifford, J.; Gadia, S.; Jajodia, S.; Segev, A.; Snodgrass, R.: *Temporal Databases: Theory, Design and Implementation*. The Benjamin/Cummings Publishing Company Inc., Redwood City (CA) u.a., 1993