Quality of Service and Optimization in Data Integration Systems

Reinhard Braumandl
Universität Passau

Abstract: Durch die anhaltende Globalisierung der Weltwirtschaft, ist die Verarbeitung von verteilten Datenbeständen mittlerweile unabdingbar [LKK+97]. Anfrageverarbeitung auf diesen Daten analog zu relationalen Datenbanken ist hierbei eine Schlüsselfertigkeit, allerdings in einem Internet-weit skalierten Anwendungsszenario auch schwer zu realisieren. Das Datenintegrationssystem ObjectGlobe soll Anfrageverarbeitung in dieser Art ermöglichen. Dessen Basisarchitektur und das integrierte Dienstgütemanagement (Quality of Service, QoS) wurden in dieser Dissertation erarbeitet und hier kurz zusammengefasst. Die Dissertation [Bra02] selber kann unter folgender URL bezogen werden:
`http://elib.ub.uni-passau.de/opus/volltexte/2002/27`.

1 Die Bedeutung von Anfrageverarbeitung im Internet

Die fortschreitende Globalisierung der Weltwirtschaft zieht auch die Forderung nach angepassten globalen Datenverarbeitungsmöglichkeiten im Internet nach sich. Anfrageverarbeitung ist diesbezüglich eine der Kernaufgaben und spielt in Anwendungen, wie z.B. im Bereich der maschinellen Entscheidungsunterstützung (decision support) eine tragende Rolle. Das Internet bietet hierzu den Zugriff auf eine riesige Anzahl von Datenquellen, wie z.B. Hotelübersichten, Bahn- und Flugpläne, Aktienkurse und wissenschaftliche Messdaten, wie Satellitenbilder und Genom Datenbanken. Datenintegrations- bzw. Mediatorsysteme wurden in diesem Zusammenhang entwickelt, um auf diese Daten analog zu herkömmlichen Datenbanksystemen zugreifen zu können. Zuerst waren dies geschlossene Systeme mit einem festgelegten Satz von unterstützten Datenquellen und einer zentralen Verarbeitung der Mediator-spezifischen Aufgaben. In letzter Zeit wurde verstärkt die Entwicklung verteilter und flexibler Datenintegrationssysteme vorangetrieben, wobei Amos II [JKR99] oder das hier behandelte ObjectGlobe System [BKK+01] zwei Vertreter sind.

Innerhalb einer ObjectGlobe Föderation partizipieren drei Arten von Providern: *Datenprovider* (data provider) stellen ihre Daten zur Verfügung, *Funktionenprovider* (function provider) steuern Anfrageoperatoren und Funktionen bei und *Rechenzeitprovider* (cycle provider) können zur Verarbeitung von Teilen einer Anfrage herangezogen werden. In ObjectGlobe können die Dienste dieser drei Providerarten auf nahezu orthogonale Weise miteinander verknüpft werden. Einschränkungen in der Orthogonalität können bestehen nur aufgrund von Sicherheits-, Datenschutz- und Kapazitätsaspekten. Auf diese Weise stellt das ObjectGlobe System einen offenen und verteilten Servicemarkt für Anfrageverarbeitung zur Verfügung.

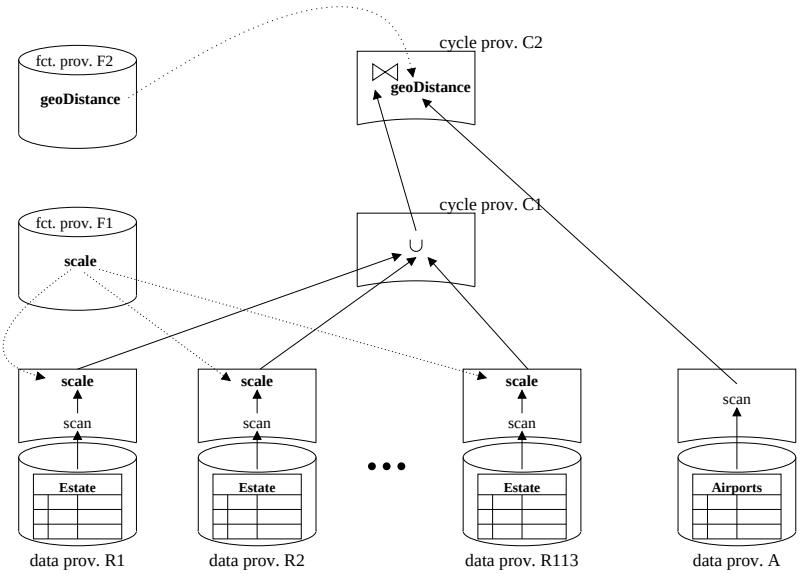

Abbildung 1: Ein Beispiel für eine Anfrage in einer ObjectGlobe Föderation.

Nehmen wir z.B. an, dass ein ObjectGlobe Benutzer eine Villa im Mittelmeerraum erstehen will und dabei Anforderungen bezüglich der maximalen Distanz des Objekts zum nächsten Flughafen und der Wohnfläche hat. Die SQL-ähnliche Anfrage hierzu berechnet eine Joinoperation zwischen Immobilien und Flughäfen. Das Joinprädikat verwendet eine benutzer-definierte, externe Funktion, die die Länge der Luftlinie zwischen zwei Standorten errechnet. In der Anfrage wird auch eine externe Funktion **scale** benutzt, um Bilder der Immobilien in eine handliche Größe zu skalieren. Die Bedeutung der Attribute der Datenmengen sollten aufgrund ihrer Namen klar sein.

```
select e.Price, e.Location.City,
       scale(e.Image,0.3), a.Name
from Estate e, Airports a
where e.building-area > 200 and
  geoDistance(e.Location,a.Location) < 10
  and e.Location.region = 'Mediterranean';
```

In einer ObjectGlobe Föderation werden externe Datenmengen mit Hilfe so genannter **Wrapper** eingebunden und in das interne, vernestete relationale Datenmodell überführt. Daher könnten die Daten jedes europäischen Immobilienmaklers als Partition der Relation **Estate** eingebunden werden. Ein möglicher Anfrageauswertungsplan in ObjectGlobe ist in Abbildung 1 zu sehen. Die Daten für Flughäfen werden vom Datenprovider **A** beigesteuert und für die Immobiliendaten von den Providern **R1, . . . , R113**. Die letzteren Provider stellen dabei kombinierte Daten- und Rechenzeitprovider dar, welche die externe Funktion **scale** ausführen, die von Funktionenprovider **F1** zur Verfügung gestellt wird. Zusätzlich

werden von zwei reinen Rechenzeitprovidern die **union** Operation für die Immobilienpartitionen und die **join** Operation zwischen den Immobiliendaten und Flughafendaten ausgeführt. Reine Rechenzeitprovider spielen eine wichtige Rolle, falls Datenprovider nicht in der Lage oder willens sind, andere als **scan** Operationen auszuführen oder wenn Operationen dadurch so platziert werden können, dass Kommunikationskosten eingespart werden.

Offensichtlich können Anfragen in so einem System nur schwer bezüglich ihren Ausführungseigenschaften (z.B. Ausführungsdauer) eingeschätzt werden. Daher unterstützt das ObjectGlobe System den Benutzer durch ein Dienstgütemanagement [BKK03]. Die generelle Bedeutung von Dienstgütegarantien in Informationssystemen wird insbesondere in [Wei99] betont. Eine Arbeit im Bereich der Datenintegrationssysteme, die sich mit Dienstgüte im Bereich der Datenqualität beschäftigt hat, findet sich in [NLF99].

2 Das Dienstgüte-Modell

Wie zuvor angesprochen, sollten Benutzer in einem durch eine ObjectGlobe Föderation aufgespannte Informationswirtschaft (information economy) die Abhängigkeiten zwischen Datenqualität des Anfrageergebnisses, der Ausführungsgeschwindigkeit und -kosten spezifizieren können. Zum Beispiel, könnten Benutzer ausdrücken wollen, dass sie bei schnellerer Ausführung entsprechend höhere Ausführungskosten akzeptieren.

Dies bedeutet, dass wir analog zu Aufwandsmodellen[1] in herkömmlichen Datenbanksystemen ein Modell benötigen, um Dienstgütevereinbarungen für Anfragen, Anfrageauswertungspläne und -ausführungen ausdrücken zu können. Deshalb wird im folgenden eine Menge von Dienstgüteparameter (QoS Parameter) angegeben, die dazu dienen Dienstgütevereinbarungen zu spezifizieren. Hierbei sind die Parameter in die drei Bereiche der Datenqualität, Zeiteigenschaften und Kosten eingeordnet.

Parameter für die Datenqualität:

- der Zeitpunkt der letzten Datenänderung für eine Partition einer Relation oder ein Faktor, der den Prozentsatz der ausstehenden Updates wiedergibt.
- der Anteil der in einer Anfrage benutzten Partitionen im Vergleich zum gesamten Datenumfang einer Relation.
- die untere Schranke für die Kardinalität des Ergebnisses. Hierdurch legen Benutzer fest, dass sie eine Minimalgröße des Ergebnisses erwarten, um z.B. aus dieser Menge einen Favoriten wählen zu können.
- die obere Schranke für die Kardinalität des Ergebnisses. Dieser Parameter entspricht der *stop after* Klausel, wie sie in [CK98] vorgeschlagen wurde.

[1] Meist auch Kostenmodell genannt.

Parameter für die zeitlichen Eigenschaften:

- der Zeitverbrauch bis das erste Ergebnistupel produziert worden ist. In dieser Zeit können Benutzer einfach nur warten.
- der Zeitverbrauch, um nach dem ersten alle weiteren Ergebnistupel zu produzieren. In dieser Zeit können Benutzer schon Ergebnistupel betrachten.

Parameter für die Ausführungskosten: Da ein System wie ObjectGlobe nur realisierbar ist, wenn Provider durch Bezahlung zur Bereitstellung ihrer Dienste motiviert werden, müssen Benutzer die Kosten einer Anfrageverarbeitung beschränken können. Wir sehen folgende Parameter vor:

- die Kosten für die Dienste von Funktionenprovider, z.B. die Kosten, um eine Funktion für die Dauer einer Anfrageauswertung zu leasen.
- die Kosten für die Dienste von Datenprovidern, z.B. für das Lesen einer bestimmten Datenpartition des Providers.
- die Kosten für die Dienste von Rechenzeitprovidern, z.B. für das Auführen eines Teilplanes

3 Die Unterstützung von Dienstgütevereinbarungen in der Anfrageverarbeitung

Natürlich hängt die Fähigkeit, Dienstgütevereinbarungen einzuhalten von der erreichbaren Dienstgüte der gemeinsam benutzten Ressourcen ab. Von den gängigen Schedulingstrategien wie *Best Effort*, *Priority-Based Scheduling* und *Reservation* erlaubt nur die letztere ein Dienstgütemanagement mit Garantien für die Anfrageausführung. Die Benutzung von Reservierungen zieht aber in der Regel eine schlechte Ressourcenausnutzung nach sich und wird daher kaum für das Scheduling in Netzwerken und Rechenanlagen verwendet. Da Reservierungen deswegen nicht verwendet werden können, verbleiben für das Dienstgütemanagement von ObjectGlobe zwei offensichtliche Ziele:

- Der Prozentsatz der Anfragen, deren Dienstgütevereinbarungen erfüllt werden können, sollte maximiert werden. Dieser Prozentsatz wird berechnet auf der Basis der insgesamt dem System gestellten Anfragen, wozu also auch solche zählen, für die schon kein passender Anfrageplan gefunden werden konnte.
- Die Ausführung von Anfragen, die ihre Dienstgütevereinbarungen nicht mehr erfüllen können, sollte so früh wie möglich gestoppt werden. Dadurch verschwendet die Anfrage nicht die Zeit und das Geld des Benutzers.

Die Anfrageverarbeitung mit integriertem Dienstgütemanagement wird in Abbildung 2 überblicksartig dargestellt. Der Startpunkt einer Anfrageverarbeitung in unserem System

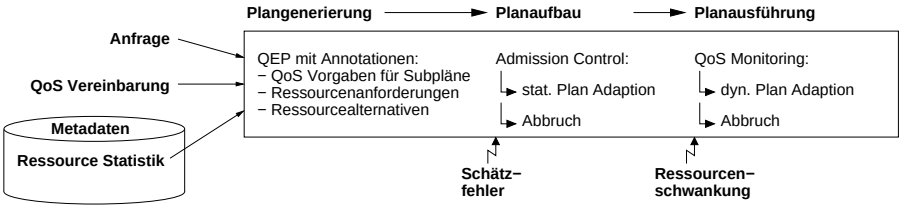

Abbildung 2: Die Interaktion von Anfrageverarbeitung und Dienstgütemanagement

ist die deklarative Anfrage selber, die gewünschten Dienstgütevereinbarungen des Benutzers und Statistiken über Ressourcenauslastung der benutzbaren Provider und Netzwerke. Abbildung 2 zeigt die Aktivitäten des Dienstgütemanagements in den Phasen der Anfrageverarbeitung: Plangenerierung, Planaufbau und Planausführung.

Plangenerierung: Der Optimierer erzeugt einen Ausführungsplan (QEP) mit Informationen über zu benutzende Daten-, Rechenzeit- und Funktionenprovider und über die Verknüpfung der Dienste selbiger. Der Optimierer ist im wesentlichen ein aufwendiger Suchalgorithmus, der in unserem Fall Dynamic Programming basiert arbeitet. Dieser bewertet eine Vielzahl von verschiedenen Plänen anhand des Qualitätsmodells, welches Formeln zur Berechnung der QoS Parameter bereithält. Darin gehen der Aufbau des Planes und Statistiken über die Provider und Ressourcen ein. Pläne haben die Struktur eines Operatorbaumes und werden daher durch Dynamic Programming in einer bottom-up Berechnung aufgebaut. Für jeden dabei erhaltenen Subplan werden die dazugehörigen QoS Parameter errechnet, die wiederum in die Berechnung der Dienstgüteparameter der darauf aufbauenden Pläne einfließen. Nur Pläne mit sich qualifizierenden Werten für die QoS Parameter werden betrachtet. Dabei wird jeder Plan samt allen seinen Subplänen mit den jeweiligen Abschätzungen für QoS Parameter und Ressourcenverbrauch annotiert. Zusätzlich werden noch, falls vorhanden, passende, alternative Ressourcen vermerkt, die im Falle eines Problems mit einer Ressource verwendet werden können.

Planaufbau: Während des Planaufbaus werden die Subpläne auf die Rechenzeitprovider verteilt, die Funktionen und Operatoren von Funktionenprovider geladen und die Verbindungen zu Datenprovidern hergestellt. Für jeden Provider wird überprüft, ob die errechneten Anforderungen des Plans an die dazugehörigen Ressourcen erfüllt werden können. Ein Rechenzeitprovider z.B. könnte nicht mehr genügend CPU-Leistung für den Plan verfügbar haben. In diesem Fall könnte die Dienstgüte-konforme Abarbeitung des Plans nicht mehr möglich sein und auch andere Pläne auf dem Rechenzeitprovider könnten durch die möglicherweise entstehende Überlast gefährdet sein. Die Admission Control verhindert also in so einem Fall die Ausführung des Planes auf dem Rechenzeitprovider oder adaptiert den Plan, z.B. indem sie die Instantiierung auf einem alternativ vermerkten Rechenzeitprovider anstößt.

Planausführung: Während der Anfrageausführung können Schwankungen in der Ressourcenverfügbarkeit (z.B. bei CPU-Leistung und Netzwerkbandbreite) und auch

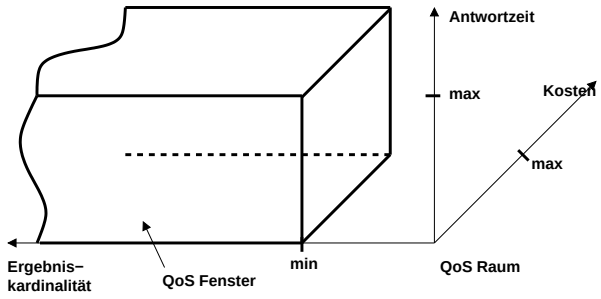

Abbildung 3: Der QoS Raum und das QoS Fenster.

Fehlschätzungen zur Optimierungszeit die Einhaltung der Dienstgütevereinbarungen gefährden. Um diese Situationen zu erkennen, werden die QoS Parameter der Ausführung auf der Ebene jedes Subplans überwacht. Falls potenzielle Verletzungen erkannt werden, wird zuerst versucht per Adaption des Planes eine Verbesserung der Situation zu erwirken oder falls dies nicht möglich erscheint, die Planausführung abgebrochen. Die hierbei eingesetzten Adaptionen sind in der Lage, auf eine laufende Planausführung entsprechend einzuwirken. So ist z.B. ein Verschieben von Operatoren auch in dieser Phase möglich.

Da die Schätzung der QoS Parameter und die Überwachung dieser auf Subplan Ebene durchgeführt wird, kann sehr frühzeitig in der Planinstantiierung und -aufführung und auch sehr feingranular reagiert werden. Dies ist wichtig, da z.B. im Gegensatz zu Video- und Audiostreaming Anwendungen die benutzer-definierten Dienstgütevereinbarungen in Datenintegrationssystemen erst am Ende der Ausführung des Gesamtplans überprüfbar wären.

4 Dienstgüteunterstützung in der Plangenerierung

In diesem Abschnitt sprechen wir die Modifikationen eines klassischen, Dynamic Programming basierten Anfrageoptimierers an, die zur Unterstützung von Dienstgütevereinbarungen nötig sind. Wir konzentrieren uns dabei auf die Komponenten, die für unsere Belange eine wichtige Rolle spielen.

Providerauswahl In vielen Fällen wird es nicht möglich sein, alle Datenprovider für eine bestimmte Relation zu betrachten, da die resultierende Datenmenge in einer weitverteilten Anfrage zu inakzeptablen Ausführungszeiten führen würde. Eine Aufgabe des Dienstgütemanagements ist es daher, relevante Datenprovider und dazu günstig gelagerte Rechenzeitprovider auszuwählen. Da kompakte Ausführungspläne, bezogen auf die Netzwerkausdehnung, in der Regel auch relativ effizient verarbeitet werden können, benutzen

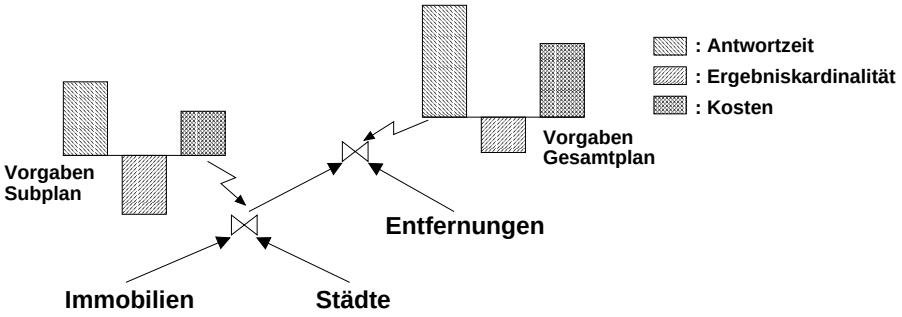

Abbildung 4: QoS Konten eines Anfrageplans.

wir Clustering Algorithmen, um Daten- und Rechenzeitprovider bezüglich ihrer Netzwerkentfernung zueinander zu gruppieren. Während der Optimierung ist es oft ausreichend, dann auf diesen Clustern zu arbeiten, anstatt die einzelnen Provider zu betrachten. Dadurch kann der Berechnungsaufwand während der Optimierung reduziert werden.

Berechnung der QoS Parameter Auswertungspläne sind im wesentlichen Operatorbäume aus z.B. Join-, Vereinigung- bzw. benutzerdefinierten Operatoren und daher hierarchisch aufgebaut. Analog dazu werden auch die QoS Parameter für Pläne hierarchisch errechnet, indem entsprechende QoS-Modelle für Operatoren die QoS Parameter für die dazugehörigen Eingabepläne verknüpfen. In diesen Modellen wird z.B. auch die Parallelität in der Verarbeitung des Operators in Bezug zu seinen Eingabeplänen bewertet, um auf das Gesamtverhalten des Teilplans schließen zu können.

Selektion von Anfragepläne Der Optimierer zählt alternative Anfragepläne auf und verwirft solche die aufgrund ihrer berechneten QoS Parameter unterlegen sind. Da die Bewertung eines Planes mehrdimensional erfolgt, können unvergleichbare Pläne nicht mehr ohne weiteres selektiert werden. Wir betrachten daher die QoS Vorgaben des Benutzers. Die QoS Parameter spannen einen Raum auf, den wir QoS Raum nennen. Darin definieren die Dienstgütevorgaben des Benutzers einen Bereich, den wir QoS Fenster nennen. Dies ist für den vereinfachten Fall eines dreidimensionalen QoS Raums in Abbildung 3 gezeigt. Jeder Plan, der während der Optimierung aufgezählt wird, ist anhand der errechneten QoS Parameter einem Punkt im QoS Raum zugeordnet. Dabei qualifizieren sich nur solche Pläne, die einem Punkt im QoS Fenster zugeordnet sind. Pläne in diesem Fenster werden anschließend aufgrund ihrer Abstände zu den Dienstgütevorgaben anhand der Manhattan Metrik ausgewählt.

5 Adaption eines Auswertungsplans

Anfragepläne haben in unserem System eine “natürliche” Fragmentierung, welche durch die Thread- und Rechnergrenzen innerhalb des Operatorbaums gegeben sind. An diesen Übergängen überwachen Monitoroperatoren die tatsächlichen QoS Parameter ihrer Eingabepläne und berechnen einen Trend für diese Parameter gegen das Ende der Verarbeitung. Die zugehörigen Berechnungen des Optimierers für den jeweiligen Subplan (wie in Abbildung 4 dargestellt) stellen die Zielwerte für die Trends dar. Ein Vergleich der beiden Werte zeigt für einen QoS Parameter, inwieweit dieser Parameter gefährdet ist.

Wenn während der Verarbeitung eine Verletzung einer Dienstgütevereinbarung erwartet wird, können wir versuchen dem durch Adaption des Planes entgegen zu wirken. Die Adaptionen, die wir einsetzen, um einer vorhergesagten Dienstgüteverletzung zu begegnen, arbeiten hauptsächlich auf der Ressourcenallokation des Anfrageplans. Hierbei kann die Ressourcenallokation auf einem Provider variiert werden oder evtl. sogar der Provider selbst ausgetauscht werden.

Beispiele für Adaptionen sind:

increasePriority/decreasePriority verändern die Prioritäten der Threads, die auf einem Rechenzeitprovider zur Berechnung der Anfrage benutzt werden. Durch diese Adaption können wir eine Anfrage beschleunigen, die früher abgearbeitet sein muss oder wir können die Verarbeitungsgeschwindigkeit einer Anfrage drosseln und somit Ressourcen für andere Anfragen freisetzen, falls diese Anfrage kein Problem mit ihrem Zeitlimit hat.

useCompression schaltet für eine Netzwerkverbindung die Kompression der darüber ausgetauschten Daten zu.

movePlan wird benutzt, um ein Planfragment von einem Rechenzeitprovider auf einen anderen zu transferieren. Dieser Transfer ist auch bei schon begonnener Verarbeitung möglich und transparent für den übrigen Anfrageplan.

6 Regelung der Adaptionen

Die Planadaption zur Laufzeit wird von Monitoroperatoren angesteuert, da diese an den passenden Stellen im Plan positioniert sind und auch schon die entsprechenden Statistiken zur Entscheidungsfindung für den Einsatz von Adaptionen erfassen. Wie in Abbildung 5 gezeigt ist, benutzt der Monitoroperator einen Fuzzy Controller, der die Trends für die QoS Parameter zusammen mit einigen anderen Zustandsparametern erhält und auf dieser Basis die Anwendung von Adaptionen regelt.

Für den Einsatz eines Fuzzy Controller zu diesem Zweck sprechen einerseits seine durch den Einsatz von Fuzzy Logic gegebenen Möglichkeiten beim Umgang mit unscharfen Werten, wie z.B. den Trends für die Qualitätsparameter. Andererseits bieten Fuzzy Controller durch den regelbasierten Ansatz sehr mächtige Möglichkeiten, um Expertenwissen

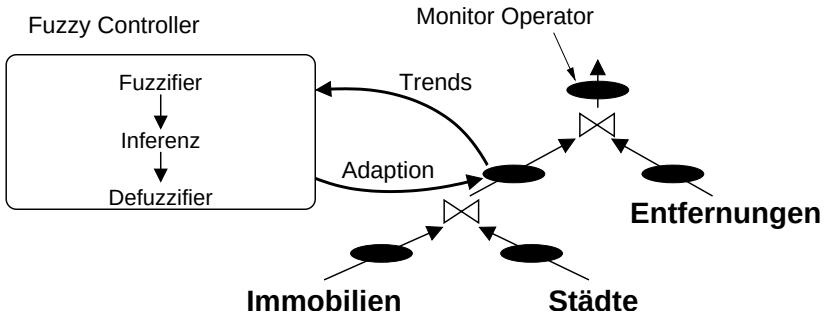

Abbildung 5: Feedback Schleife für QoS Adaptionen.

intuitiv ausdrücken zu können. Deshalb ist es leicht möglich, durch Variation der Regelbasis und/oder der Fuzzy Set Definitionen mit neuen Strategien für das Anwenden von Adaptionen zu experimentieren.

Intern arbeitet ein Fuzzy Controller in drei Stufen. Ein Fuzzifier transformiert die numerischen Eingabewerte in so genannte linguistische Werte (*linguistic values*), welche linguistischen Variablen (*linguistic variables*) zugeordnet sind. Über Inferenzregeln der Form

if X_1 is A_1 and $\ldots$ and X_n is A_n **then** Y is B

werden dann die Gewichte der linguistischen Werte für die Implikationen errechnet. Diese werden dann durch einen Defuzzifier zu einer Globalaussage des Controllers zusammengefasst, welche in unserem Fall die anwendbare Adaption darstellt.

Ein Auszug aus einer Regelbasis ist im folgenden gezeigt. Dabei bezeichnen *lfd*, *lfr* und *lft* die linguistischen Variablen für die Kosten-, Datenqualität- und Zeitparameter. Die Variablen *lep*, *lbp* und *lss* stehen hierbei für den Ausführungsfortschritt, den Pufferdruck und die Zustandsgröße des Plans.

if *lfc* is *endangered* and *lep* is *late* **then** *abort* is *preferred*

if *lft* is *endangered* and *lbp* is *high* and *lss* is *small* **then** *movePlan* is *preferred*

if *lft* is *endangered* and *lep* is *middle* and *lbp* is *high* **then** *increasePriority* is *possible*

if *lfr* is *endangered* and *lep* is *early* and *lfc* is *compliant* **then** *addSubPlan* is *preferred*

7 Danksagung

Zuallererst möchte ich meinen Betreuern Prof. Alfons Kemper und Prof. Donald Kossmann für ihre Unterstützung danken. Sie gaben mir die Möglichkeit in einem ambitionierten und visionären Projekt mitzuarbeiten. Von ihren Einsichten und Erfahrungen in der Forschung konnte ich viel lernen.

Weiterhin möchte ich den Studenten und meinen Kollegen an der Universität Passau, mit denen ich in meiner Zeit dort zusammenarbeiten durfte, für interessante Diskussionen und eine angenehme Arbeitsatmosphäre danken.

Literaturverzeichnis

[BKK+01] R. Braumandl, M. Keidl, A. Kemper, D. Kossmann, A. Kreutz, S. Seltzsam, and K. Stocker. ObjectGlobe: Ubiquitous query processing on the Internet. *The VLDB Journal: Special Issue on E-Services*, 10(3):48–71, August 2001.

[BKK03] R. Braumandl, A. Kemper, and D. Kossmann. Quality of service in an information economy. 2003. Submitted for publication.

[Bra02] R. Braumandl. *Quality of Service and Optimization in Data Integration Systems*. PhD thesis, Universität Passau, Fakultät für Mathematik und Informatik, D-94030 Passau, 2002. Universität Passau.

[CK98] M. Carey and D. Kossmann. Reducing the braking distance of an SQL query engine. In *Proc. of the Conf. on Very Large Data Bases (VLDB)*, pages 158–169, New York, USA, August 1998.

[JKR99] V. Josifovski, T. Katchaounov, and T. Risch. Optimizing queries in distributed and composable mediators. In *Proc. of the IFCIS International Conference on Cooperative Information Systems*, pages 291 – 302, Edinburgh, Scotland, 1999.

[LKK+97] P. Lockemann, U. Kölsch, A. Koschel, R. Kramer, R. Nikolai, M. Wallrath, and H.-D. Walter. The network as a global database: Challenges of interoperability, proactivity, interactiveness, legacy. In *Proc. of the Conf. on Very Large Data Bases (VLDB)*, pages 567–574, Athens, Greece, August 1997.

[NLF99] Felix Naumann, Ulf Leser, and Johann Christoph Freytag. Quality-driven integration of heterogenous information systems. In *Proc. of the Conf. on Very Large Data Bases (VLDB)*, pages 447–458, Edinburgh, GB, September 1999.

[Wei99] G. Weikum. Towards guaranteed quality and dependability of information services. In *Proc. GI Conf. on Database Systems for Office, Engineering, and Scientific Applications (BTW)*, Informatik aktuell, New York, Berlin, etc., 1999. Springer-Verlag.

Zum Autor: Reinhard Braumandl hat von 1992 bis 1997 an der Universität Passau Informatik studiert. Die anschließende Promotion am Lehrstuhl von Professor Kemper hat er im Februar 2002 abgeschlossen. In seiner Zeit am Lehrstuhl hat er sich in den Projekten Merlin und ObjectGlobe hauptsächlich mit Anfrageverarbeitung in objekt-orientierten und verteilten Datenbanksystemen beschäftigt.