Operationen auf komplexen Objekten in ORDBMS: Analyse, Vergleich und Optimierung

Nico Lachmann
Friedrich-Schiller-Universität Jena
Institut für Informatik
Lehrstuhl für Datenbanken und Informationssysteme
07745 Jena
Nico.Lachmann@minet.uni-jena.de

Kurzfassung

Moderne objektrelationale Datenbanksysteme integrieren objektorientierte Konzepte, erweiterbare Typsysteme und Kollektionsdatentypen in die relationale Datenhaltung. Besonders Applikationen im ingenieurwissenschaftlichen Bereich benötigen häufig solche erweiterten Konzepte für die Modellierung von Datenbankschemata und Verarbeitungslogik. Ein entscheidendes Kriterium für die Anwendbarkeit solcher erweiterter Modellierungsmöglichkeiten bleibt aber stets die Leistungsfähigkeit, die wesentlich von der physischen Repräsentation solcher komplexen Objekte innerhalb des DBMS und von passenden Anfragebearbeitungsalgorithmen abhängt. Reale DBMS unterstützen hier üblicherweise zu wenige physische Abbildungsalternativen. Dementsprechend fehlt auch die Flexibilität auf der algorithmischen Seite, was dazu führt, daß die Anforderungen an die Leistung nur unzureichend erfüllt werden können. Die vorliegende Arbeit basiert auf der Beschreibungssprache PRDL, die eine flexible Definition der Speicherungsstrukur und Indexierung von komplexen Objekten erlaubt. Im Mittelpunkt steht die Untersuchung und der Vergleich verschiedener Alternativen der Anfragebearbeitung unter Berücksichtigung von Speicherstrukturen und Workloadanforderungen. Es wird skizziert, wie bestimmte Klassen von Operationen, beispielsweise Kollektionsoperationen, effizienter gestaltet werden können. Als Bewertungskriterium dienen einfache Kostenfunktionen. Schließlich wird eine Simulationsumgebung vorgestellt, mit deren Hilfe die theoretischen Überlegungen validiert werden sollen.

1 Einführung

Die evolutionäre Weiterentwicklung von relationalen zu objektrelationalen Datenbanksystemen integriert neue Konzepte, wie erweiterbare Typsysteme, nutzerdefinierte Funktionen, Objekttypen mit Methoden, Vererbung, Objektreferenzen und Paketdefinitionen, in die relationale Datenhaltung [DD98, SM96]. Durch die fortschreitende Integration von Kollektionsdatentypen in die SQL-Norm sind Kollektionen zukünftig auf heterogene Weise in realen ORDBMS verfügbar. Als weiteres Modellierungselement zwischen den einfachen, vordefinierten Datentypen und den komplexen, typhierarchie-orientierten Objekttypen sind sie vielfältig einsetzbar. Für Optimierungsüberlegungen auf physischer Seite eröffnen die vordefinierten Kollektionsoperationen interessante Möglichkeiten, deren Erschließung im weiteren verfolgt wird.

Für Applikationen im ingenieurwissenschaftlichen Bereich, CAD oder Vermeßungstechnik, bilden komplex verschachtelbare Datentypen eine solide Basis für die Modellierung von Datenbankschemata und Verarbeitunslogik. Ein entscheidendes Kriterium für die Anwendung solcher erweiterter Modellierungsmöglichkeiten bleibt aber stets der Leistungsaspekt, der sich unmittelbar aus der physischen Repräsentation dieser komplexen Objekte innerhalb des DBMS ableitet. Die in der industriellen und wissenschaftlichen Praxis sehr häufig eingesetzten DBMS, wie DB2 und Oracle, erlauben nur in sehr eingeschränkter Weise eine Modifikation ihrer physischen Basisparameter. Zwar stehen mittlerweile vielfältige Indexstrukturen und Clustermechanismen zur Verfügung, dennoch werden die auf logischer Ebene definierten komplexen Objekte häufig nur auf fixe, herstellerspezifische Art und Weise auf das darunterliegende relationale Speichersystem abgebildet. Gerade hier wäre es von entscheidender Wichtigkeit, in Hinblick auf differenzierte Anfrageprofile mit dem ihnen zugrundeliegenden Mix an Operationstypen, auch die Möglichkeit verschiedener physischer Repräsentationen zu haben. Die Auswahl einer bestimmten physischen Speicherstruktur sollte dabei weitestgehend unabhängig vom logischen Modell getroffen werden können. Erst durch das Zusammenspiel von Algorithmus und passender Datenstruktur kann eine möglichst leistungsfähige Lösung erzielt werden.

Mit ähnlichen Zielvorstellungen wurden in der Vergangenheit schon vielfältige Vorschläge für neuartige und erweiterte Datenmodelle formuliert. Stellvertretend seien hier das Molekül-Atom-Modell, das NF^2-Datenmodell oder das eNF^2-Datenmodell genannt [DPS86, Mit88]. Auf Basis dieser Datenmodelle entstanden unterschiedliche objektorientierte DBMS-Prototypen, die komplex strukturierte Objekte auf Modellierungsebene und in den Anfragesprachen direkt unterstützten.

Obwohl die strukturellen Aspekte durch solche erweiterten Datenmodelle gut beschrieben werden können, erlauben sie, genau wie aktuelle objektrelationale Systeme, die Definition flexibler physischer Speicherungsstrukturen, wie sie in [Keß95, Ska02] vorgeschlagen werden, bestenfalls ansatzweise. Stattdessen gehen sie von einer fixen Abbildung logischer auf physischer Strukturen aus. Ein Hinderniss für flexible Speicherstrukturen ist einerseits, daß herkömmliche ORDBMS auf historisch gewachsenen relationalen Systemen basieren, die solche Möglichkeiten nicht ohne weiteres nachrüsten lassen. Zum anderen fehlte hier bisher ein einheitliches Schema, nach dem solche physische Repräsentationen umfassend klassifiziert werden können.

2 PRDL - Physical Representation Definition Language

Als Grundgerüst zur Beschreibung der physischen Darstellung von komplexen Objekten dient uns *PRDL* [Kis04, Ska02]. Es handelt sich dabei um einen Vorschlag für eine Definitionssprache, mit deren Hilfe sich Satz-, Cluster- und Indexstrukturen für interne Sätze sehr exakt spezifizieren lassen. In seiner vorliegenden Form erhebt PRDL allerdings nicht den Anspruch, ein einheitliches und umfassendes Beschreibungsschema zu sein. Stattdessen setzt es eindeutige Schwerpunkte auf die Darstellung von kollektionswertigen Attributen und ausgewählten Indexstrukturen.

Ausgehend von der Organisation der internen Sätze in Seiten, Segmenten und Extents, wie sie aus RDBMS bekannt ist, werden in PRDL typische Einschränkungen aufgehoben. So sind etwa gemischte Segmente und Seiten erlaubt (also Seiten die Sätze unterschiedlicher Typen enthalten) und große Datenobjekte können auf Basis ihrer logischen Struktur in mehrere Sätze zerlegt und nicht nur in LOB-Bereiche ausgelagert oder seitenüberspannend gespeichert werden.

Die Zerlegung und Speicherung komplex strukturierter Objekte kann nach verschiede-

nen Kriterien erfolgen, von denen im folgenden einige genannt werden.

- Attributorientierte Zerlegung: Einzelne Attribute eines logischen Tupels werden , unabhängig von ihrem Vorkommen innerhalb einer Struktur, in internen Sätzen gruppiert. Es könnten etwa die Attribute zusammengruppiert werden, die am häufigsten verwendet werden.
- Strukturorientierte Zerlegung: Die Attribute werden aufgrund ihres gemeinsamen Vorkommens innerhalb einer Struktur in einen internen Satz gruppiert. Dies ist besonders dann günstig, wenn alle untergeordneten Attribute einer Struktur häufig gemeinsam benötigt werden.
- Verkettung: Der Zusammenhang zwischen den einzelnen internen Sätzen eines logischen Tupels kann entweder passiv, durch Platzierung in einem gemeinsamen Seitenbereich, oder aktiv, durch direktes oder indirektes Referenzieren, erfolgen. Beim direkten Referenzieren werden die einzelnen internen Sätze listen- oder hierarchieorientiert miteinander durch logische oder physische Verweise verkettet, die innerhalb der Sätze selbst gespeichert sind. Indirekte Verfahren nutzen Indexstrukturen wie B-Bäume, die die Verweisinformationen entsprechend außerhalb der Sätze speichern.
- Clusterung: Die durch die Zerlegung entstandenen Teilsätze können nach verschiedenen Kriterien geclustert werden. Je nachdem, ob hauptsächlich Teilobjekte oder komplette Objekte verarbeitet werden, bietet sich eine objektübergreifende oder objektbezogene Clusterungsstrategie an.

Jede gewählte Form von Zerlegung, Verkettung oder Clusterung hat natürliche Implikationen auf die Leistung einzelner Klassen von Retrieval- und Updateoperationen. Für die meisten praktischen Fälle, in denen gewöhnlich sehr differenzierte Anfrageprofile auftreten, kann eine optimale physische Speicherstruktur a priori nicht angegeben werden kann. PRDL zielt daher primär darauf ab einen Möglichkeitenraum aufzuspannen, um Optimierungsüberlegungen auf dieser Ebene überhaupt zu gestatten.

3 Ziel der Arbeit

Im Rahmen meiner Diplomarbeit soll der Variantenraum an physischen Strukturen, der durch PRDL aufgespannt wird, hinsichtlich ihrer Verwendbarkeit mit verschiedenen Operationen untersucht und bewertet werden. Grundsätzlich lassen sich diese entweder den nur lesenden (*Retrieval*) oder den schreibenden (*Update*) Operationen zuordnen. Betrachtet werden schwerpunktmäßig im folgenden nur die Retrieval-Operationen. Dies ist insofern gerechtfertigt, als das sie ohnehin die Grundlage für Update-Operationen bilden und daher meist den überwiegenden Anteil der Anfragen ausmachen. Hinsichtlich späterer Optimierungsmaßnahmen stellt diese Einschränkung somit keinen wesentlichen Nachteil dar.

Im Vordergrund der Überlegungen stehen Operationen die sich aus dem Umfeld von SQL:2003 und SQL$^+$ [Luf02] ergeben. Bei SQL$^+$ handelt es sich um einen Erweiterungsvorschlag für SQL, der Sprachmittel für Kollektionen und wünschenswerte Kollektionsoperationen enthält.

Ein Ziel ist es, Heuristiken zu liefern, wie komplexe Objekte inhaltlich so zerlegt werden können, daß bestimmte Klassen von Operationen mittels adaptierter Algorithmen darauf besonders günstig funktionieren, und welche Varianten für diese Algorithmen berücksichtigt werden sollten. Dazu wird ein einfaches Kostenmodell entwickelt, das es

erlaubt, eine Speicherstruktur für den jeweiligen Algorithmus zu bewerten. Dies soll als Vorbereitung für weiterführende Arbeiten dienen, die eine gesamte Workload auf kostenmäßig Weise bewerten und alternative Speicherstrukturen bestimmen wollen. Weitere Fragen, beispielsweise bezüglich der Form der Konvertierung und dafür anfallende Kosten, bleiben aber für die folgenden Betrachtungen ausgeklammert.

4 Operationsarten

Zu den logischen Operationen die für Kollektionen näher betrachtet werden, gehören allgemeine Operationen wie Kardinalität und Identität, typische Mengenoperationen (wie Teilmenge, Schnittmenge, Differenz, Enthaltensein, Duplikate), typische Listenoperationen (wie Positionssuche, Verketten, Teilliste, Vorgänger, Nachfolger) sowie typische Feldoperationen (wie Teilfeld oder indexbasierter Zugriff). Für strukturierte Typen ist die Navigation innerhalb von Pfadausdrücken von Interesse, die besonders bei stark verschachtelten Strukturen leistungsentscheidend ist.

Zu jeder dieser logischen Operationen gibt es, in Abhängigkeit von der vorhandenen physischen Speicherstruktur und etwaigen Zugriffspfaden, eine Anzahl möglicher Algorithmen für ihre Auswertung. Wird beispielsweise eine Relation R anenommen, die genau ein mengenwertiges Attribut M besitzt, so könnte M, wenn es als sortierte Liste vorliegt, für die Operation Enthaltensein sowohl eine binäre Suche als auch eine sequentielle Suche erlauben. Die gleiche Menge könnte auch durch k Felder dargestellt sein, wobei jedes der k Felder Elemente aus M enthält, die ein bestimmtes Kriterium erfüllen. In diesem Fall könnten Elemente von R mit identischem Attribut M etwa durch einen Nested-Loop Join bestimmt werden. Für jedes Tupelpaar aus R müßten dann alle zugehörigen Felder für M miteinander verglichen werden. Unterstellt man die Zuordnung der einzelnen Felder auf k verschiedene Cluster, wäre es dagegen sinnvoller, zunächst nur die Felder im gleichen Cluster miteinander zu vergleichen, und so nach maximal k Schritten die Ergebnismenge zu erhalten.

Viele der genannten Operationen profitieren davon, daß sie (weitestgehend) durch bereits vorhandene relationale Basisoperationen durchgeführt werden können. Das Kostenmodell kann daher auf bekannte Heuristiken und Techniken zurückgreifen. Als weiterer Schritt sollten daher Überlegungen zur Optimierung solcher Operationen getätigt werden. Als Werkzeug stehen aus der Literatur hierfür beispielsweise vielfältige Signaturverfahren zur Verfügung. Grob kann man diese Verfahren in exakte, umkehrbare Verfahren (meist Bitmap-basierend) und unscharfe Verfahren (meist Hash-basierend) unterteilen. Leider können wir die meisten davon nur in vereinfachten Varianten berücksichtigen, da gute Signaturverfahren von Wertebereichseinschränkungen ausgehen, die wir bei der Betrachtung von Kollektionen und Strukturen allgemeiner Domänen üblicherweise nicht voraussetzen können.

Der Einsatz von Signaturen als auch die Zerlegung von Sätzen zielt letztlich darauf ab, die Anzahl der für eine Anfrage oder bestimmte Workload zu lesenden Seiten und damit die I/O Kosten zu minimieren. Im Gegensatz zum rein relationalen Ansatz steht für uns aber nicht der Gedanke im Vordergrund diese Optimierung zu Lasten des Datenbankschemas (durch Partitionierung und Normalisierung), sondern gelöst vom logischen Modell allein auf physischer Ebene zu erreichen.

5 Simulationsumgebung

Die Erprobung der Überlegungen und die Validierung des Kostenmodells sollen mit Messungen an einer derzeit im Entwicklungsstadium befindlichen Simulationsumgebung durchgeführt werden. Simuliert werden Teile der Satz- und Indexverwaltung eines DBMS. Neben dem Lesen, Schreiben und Puffern von internen Sätzen stehen Methoden zum Zählen von (abstrakten) I/O-Kosten zur Verfügung. Für die Messungen werden bestimmte Anfragen oder Algorithmen direkt implementiert und verglichen. Da die Simulationsumgebung modular aufgebaut ist, können später einzelne Teilsysteme ersetzt und erweitert werden.

6 Zusammenfassung

In diesem Beitrag wurde eine Arbeit vorgestellt, die sich damit befaßt, wie bestimmte Operationen auf komplexen Objekten effizienter abgebildet werden können. Schwerpunktmäßig wurden Schwierigkeiten durch fehlende Freiheitsgrade bei der physischen Modellierung beschrieben und die Beschreibungssprache PRDL vorgestellt, die, relativ unabhängig von dem darüberliegenden logischen Datenmodell, eine Variantenvielfalt an physischen Repräsentationen eröffnet. Es wurden einige Operationen genannt, die, unter Nutzung inhaltlicher Zerlegung in Primär- und Sekundärsätze, effizienter ausgewertet werden können. Als theoretische Basis wurde hierzu ein einfaches Kostenmodell skizziert. Schließlich wurde eine Simulationsumgebung zur Evaluierung von Implementierungsdetails und Leistungsaspekten vorgestellt.

Literatur

[DD98] C. J. Date und H. Darwen. *Foundation for Object/Relational Databases: The Third Manifesto.* Addison-Wesley, Reading, MA, 1998.

[DPS86] U. Deppisch, H.B. Paul und H.J. Scheck. A Storage System for Complex Objects. In *Proc. Int. Workshop on Object-Oriented Database Systems, Pacific Groove*, Seiten 183–195. Grundlagen von Datenbanken, 1986.

[Keß95] U. Keßler. Flexible Speicherungsstrukturen und Sekundärindexe in Datenbanksystemen für komplexe Objekte. Dissertation, Fakultät für Informatik, Universität Ulm, Mai 1995.

[Kis04] F. Kissel. Indexstrukturen für komplexe Objekte in ORDBMS. Diplomarbeit, Institut für Informatik, Friedrich-Schiller-Universität Jena, März 2004.

[Luf02] J. Lufter. Kollektionsunterstützung für SQL:1999. Jenaer Schriften zur Mathematik und Informatik Math/Inf/05/02, Institut für Informatik, Friedrich-Schiller-Universität Jena, Januar 2002.

[Mit88] B. Mitschang. The Molecule-Atom Data Model. Technischer Bericht, Fakultät für Informatik, 1988.

[Ska02] S. Skatulla. Storage of Complex Types with Collection-Valued Attributes in Object-Relational Database Systems. In *Grundlagen von Datenbanken*, 2002.

[SM96] M. Stonebraker und D. Moore. *Object-Relational DBMSs: The Next Great Wave.* Morgan Kaufmann, San Francisco, CA, 1996.