Ein erweiterbarer Speichermanager für mobile DBMS

Dipl.-Wirtsch.-Inf. Erik Buchmann
buchmann@iti.cs.uni-magdeburg.de
Institut für Technische und Betriebliche Informationssysteme
Fakultät für Informatik
Otto-von-Guericke Universität Magdeburg
Postfach 4120, D-39016 Magdeburg, Germany

28. März 2002

Zusammenfassung

Durch den zunehmenden Einsatz von mobilen Geräten gewinnen mobile Anwendungen vermehrt an Bedeutung. Ein wesentlicher Aspekt dieser Anwendungen ist die effiziente und flexible Datenhaltung. Für stationäre Geräte haben sich dafür Datenbankmanagement-Systeme (DBMS) vielfach bewährt; diese berücksichtigen jedoch nicht die Kapazitätsgrenzen, die sich aus den verfügbaren Ressourcen der Mobilgeräte ergeben. Die speziell für mobile Systeme angebotenen DBMS hingegen bieten oftmals nur eingeschränkte Funktionalität, sind nicht erweiterbar und können daher nicht optimal auf die spezifischen Bedürfnisse der mit ihnen arbeitenden Anwendungen zugeschnitten werden. Dieser Beitrag beschreibt einen modularisierten, erweiterbaren Speichermanager als eine Kernkomponente für mobile DBMS und zeigt, wie durch die Kombination der vorgestellten Module eine Spezialisierung sowohl auf die Anforderungen einzelner Anwendungen als auch auf die Systemarchitektur mobiler Geräte erreicht werden kann.

1 Einleitung

Im Rahmen der fortschreitenden Miniaturisierung der elektronischen Komponenten werden mobile Geräte zunehmend leistungsfähiger und universeller einsetzbar. Waren die ersten mobilen Geräte noch nicht für ernsthafte Anwendungen geeignet, so beginnen mittlerweile die ersten Firmen, Geschäftsprozesse auf Mobilgeräte zu verlagern. Auf der anderen Seite werden die „klassischen“ Anwendungsfelder mobiler Applikationen wie beispielsweise PIM zunehmend komplexer und funktionsreicher. Daher werden Leistungen, wie sie die DBMS auf stationären Geräten anbieten, auch für mobile Anwendungen relevant. Zu diesen Leistungen zählen die effiziente Verwaltung großer[1] Datenbestände, die Geräte- und Datenunabhängigkeit sowie die Fähigkeit der konsistenten Datenhaltung.

Diesem Artikel liegt folgendes Szenario zugrunde: in einer heterogenen IT-Umgebung laufen mobile DBMS direkt auf Mobilgeräten wie PDA oder Mobiltelefonen ab. Dabei kann ein Datenabgleich über verschiedenste Netzwerkschnittstellen mit anderen mobilen oder stationären Geräten erfolgen. Der im Rahmen dieser Arbeit vorgestellte Speichermanager wiederum ist eine der Kernkomponenten mobiler DBMS.

Zu den Aufgaben eines Speichermanagers zählt die Verwaltung des freien und belegten Speichers für die in der Datenbank persistent abgelegten Daten, die Systemwiederherstellung im Fehlerfalle sowie – sofern erforderlich – die Transaktionsverwaltung und die Steuerung paralleler Zugriffe.

[1] „Groß“ im Kontext der beschränkten Ressourcen mobiler Geräte.

2 Anforderungen an einen Speichermanager für mobile DBMS

Im Allgemeinen adressieren DBMS auf stationären Geräten ein möglichst breites Aufgabenspektrum mit einem monolithischen, schwergewichtigen System, das eine große Funktionsfülle bietet. Die auf mobilen Geräten allgegenwärtige Ressourcenknappheit erfordert jedoch eine andere Vorgehensweise für mobile Anwendungen. Benötigt wird ein leichtgewichtiges, modulares System, das über den Austausch bzw. das Hinzufügen neuer Module eine Spezialisierung auf die Charakteristika der Hardware sowie die Bedürfnisse der Anwendungen erlaubt. Zur Entwicklung eines geeigneten Systemkonzepts sind zunächst die technischen und fachlichen Anforderungen zu bestimmen, die sich aus den Merkmalen der Hardware und den Bedürfnissen der Anwendungen an einen Speichermanager für mobile DBMS ergeben. Die technischen Anforderungen resultieren aus drei Merkmalen der verwendeten Hardware:

- Die verfügbaren Ressourcen sind im Vergleich zu stationären Geräten stark eingeschränkt. Die Ressourcen *Prozessor*, *Speicher* und *Kommunikationsanbindung* limitieren dabei *Durchsatz* und *Antwortzeit* des Systems. Hinzu kommt die bei stationären Geräten nicht zu berücksichtigende Begrenzung der *Verfügbarkeit* durch die *Stromversorgung*.
- Die *Kapazität* der Ressourcen ist oft unveränderlich. Bei stationären Geräten ist es möglich, die verwendete Hardware auf die zu bewältigenden Aufgaben zuzuschneiden. Prozessor sowie Speicher- und Netzwerkkomponenten sind im weiten Rahmen gegen Einheiten höherer Kapazität austauschbar. Mobile Geräte jedoch lassen Erweiterungen oftmals überhaupt nicht oder nur über einen proprietären Erweiterungsslot zu.
- Bei Mobilgeräten ist eine *Vielzahl unterschiedlicher Hardwarekomponenten* anzutreffen. Kann beispielsweise bei einem stationären Gerät von einer festen Speicherhierarchie aus RAM und Festplatten ausgegangen werden, umfassen die bei Mobilgeräten üblichen Sekundärspeichermedien batteriegepuffertes RAM, EEPROM, Flash sowie Speicher auf der Basis von Festplattentechnologie [DKL+94].

Aus diesen technischen Gegebenheiten ist ersichtlich, dass der Speichermanager ein hohes Maß an Flexibilität aufzuweisen hat. Er muss sich im Rahmen verteilter DBMS, Hauptspeicher-DBMS sowie allgemeiner DBMS einsetzen lassen. Werden ein Index im RAM, replizierte Daten auf einer Erweiterungskarte und selten benötigte Daten über eine Netzwerkschnittstelle abgerufen, sind diese drei Ausprägungen eines DBMS sogar in einer Anwendung vereint.

Ein ähnlich weites Anforderungsspektrum lässt sich aus den zu erfüllenden Aufgaben ableiten. Grundsätzlich lassen sich zwei große Anwendungsfälle unterscheiden:

Persönliches Informationsmanagement Dieser Anwendungsfall beschreibt das „klassische“ PDA-Einsatzgebiet. Dabei werden geringe Datenmengen primär auf dem Mobilgerät verwaltet und hauptsächlich mit der Workstation des zumeist einzigen Bearbeiters abgeglichen. Eine gemeinsame Bearbeitung der selben Daten mit mehreren Nutzern findet nicht statt. Häufig sind die in diesem Szenario verwendeten Geräte sehr leistungsschwach. Anwendungsbeispiele sind die Adressverwaltung, die digitale Geldbörse oder Digital Rights Management. Anwendungen des persönlichen Informationsmanagements erfordern in erster Linie Flexibilität vom Speichermanager.

Replikation von großen Datenbeständen Dieses Einsatzgebiet umfasst Anwendungen, die erst in neuerer Zeit durch die gesteigerte Leistungsfähigkeit der Mobilgeräte möglich geworden sind, wie beispielsweise die Replikation von Unternehmensdaten. Dabei werden die Daten oft von zahlreichen Nutzern kooperativ bearbeitet. Aber auch Anwendungen geografischer Informationssysteme oder Multimedia-Anwendungen, die nur lesend auf die Daten des Mobilgeräts zugreifen, gehören in diese Anwendungskategorie. Die Replikation großer Datenbestände erfordert leistungsfähige Mobilgeräte und einen performanten, spezialisierbaren Speichermanager.

Der ideale Speichermanager für mobile DBMS ist also leichtgewichtig und modular aufgebaut. Er muss sich auf eine minimale Menge unverzichtbarer Funktionen beschränken, um mit den gegebenen Kapazitäten der kleinsten vorgesehenen Geräte auszukommen, und über generalisierte Schnittstellen um anwendungsspezifische Funktionalitäten erweiterbar sein.

3 Bestehende Ansätze

Viele der skizzierten Herausforderungen für einen Speichermanager für mobile DBMS wurden bereits – wenn auch oft in einem anderen Zusammenhang – untersucht. Daher besteht ein wesentlicher Anteil der Arbeit bei der Entwicklung des Speichermanagers darin, diese Techniken an den Kontext mobiler Geräte und Applikationen zu adaptieren.

Der Einsatz von batteriegepuffertem RAM als Sekundärspeicher legt eine Übernahme von Konzepten der Speichermanager von Hauptspeicherdatenbanken wie XMAS [CPP97] oder Dali [BLR+97] nahe. Zur Realisierung der geforderten Erweiterbarkeit sind sowohl die interne als auch die externe Ebene – bezogen auf die 3 Schichten-Architektur [Dat86] – mit allgemeinen Schnittstellen auszustatten, die eine Anbindung externer Funktionen erlauben. Auf diesem Wege ist beispielsweise die Einführung neuer Datentypen [Sto86] oder Zugriffsstrukturen zu erreichen. Dazu existieren verschiedene Lösungsansätze. KIDS [GSD97] verwendet ein Broker-Services Modell. Ein Service beschreibt dabei eine vom DBMS zu unterstützende Funktionalität, und wird über Events angesprochen, die der Nachfrager des Dienstes auslöst. Broker vermitteln zwischen Services und den Nachfragern. Andere Vorschläge zur Realisierung der Erweiterbarkeit entwickeln in Anlehnung an RISC-Prozessoren eine DBMS-Funktionsbibliothek mit den elementaren DBMS-Funktionen [CW00] oder setzen auf schlanke DBMS-Kernel wie z.B. „Single Schema DBMS“ [BDS+92] auf, die sie über allgemeine Schnittstellen erweitern.

Von großer Bedeutung ist die Frage, welche Funktionseinheiten auf unterschiedliche Module aufgeteilt oder im Kern verankert werden sollen. Zu den Funktionen, die nicht von allen Anwendungen benötigt werden [TB95], zählen die Steuerung paralleler Zugriffe, Locking-Mechanismen und Transaktionen, die Systemwiederherstellung, die Verwaltung von Datensätzen, die größer als der Hauptspeicher sind, Joins und Abfragen über mehrere Relationen, Verteilte DBMS und Mengenorientierte Abfragen.

4 Architektur des Speichermanagers

Für die Realisierung des in dieser Arbeit vorgestellten Speichermanagers wurde ein aus drei Modulschnittstellen bestehender Toolkit-Ansatz gewählt. Jedes Modulobjekt verwaltet dabei nur die Daten einer einzigen Relation bzw. eines einzigen Indexes, um den Verwaltungsaufwand innerhalb der Module zu reduzieren. Werden mehrere Relationen benötigt, so sind dementsprechend auch mehr Modulobjekte zu instanzieren.

StorageManager Module mit dieser Schnittstelle haben die Aufgabe, Speicherseiten mit typlosen Datenbytes, die über eine Record-ID identifiziert werden, in die Datenobjekte, mit denen die Applikation arbeitet, zu transformieren. Unterschiedliche Ausprägungen mit der `StorageManager`-Schnittstelle können beispielsweise das relationale, objektorientierte oder objektrelationale Datenbankmodell unterstützen. Zu den Aufgaben der `StorageManager`-Module zählt weiterhin die Verwaltung der Metadaten der Relation.

AccessPathManager Diese Module realisieren unterschiedliche Zugriffspfade wie Hashtabellen, Binärbäume, B^+-Bäume, Tries oder Heaps. Dazu verwalten die Realisierungen der `AccessPathManager`-Schnittstelle Schlüssel und die zugeordnete Record-ID.

ByteLevelInterface Über diese Schnittstelle werden Module angesprochen, die Speicherseiten von typlosen Datenbytes mittels Record-ID verwalten. Module mit dem `ByteLevelInterface` nehmen verschiedene Aufgaben wahr, zu deren Ausübung sie keine Informationen über die Struktur und Semantik der verwalteten Daten benötigen: sie übernehmen die Datenverschlüsselung, die temporäre Zwischenspeicherung (Caching), die Datenkompression sowie letztendlich die Ablage der Speicherseiten auf dem Sekundärspeichermedium.

Ein möglicher Aufbau des Speichermanagers ist in Abbildung 1 dargestellt. Jedes Modul greift dabei auf genau ein nächsttieferes zu. Die Basis bilden dabei stets Module mit dem `ByteLevelInterface`, die die übergebenen Daten auf dem Sekundärspeichermedium ablegen bzw. von ihm lesen. Anfragen werden von der Anwendung nur an `AccessPathManager`- oder `StorageManager`-Module gestellt. Alle anderen Module sind optional. Die Zusammenstellung der Module beim Start der Anwendung obliegt den Anwendungsprozessen.

Da die Module des `ByteLevelInterface` über das selbe Interface angesprochen werden, sind sie in ihrer Reihenfolge austauschbar. Dadurch lässt sich das Verhalten des Speichermanagers beeinflussen: wird beispielsweise wie im linken Zweig von Abbildung 1 das Modul zur Datenkompression unterhalb des Moduls zur Datenpufferung angeordnet, so werden die auf dem Sekundärspeichermedium abgelegten Daten verdichtet. Bei der im rechten Zweig abgebildeten Anordnung oberhalb des Puffers hingegen wird auch der Pufferinhalt komprimiert, was zu einer verbesserten Speicherausnutzung zuungunsten der CPU-Ressource führt.

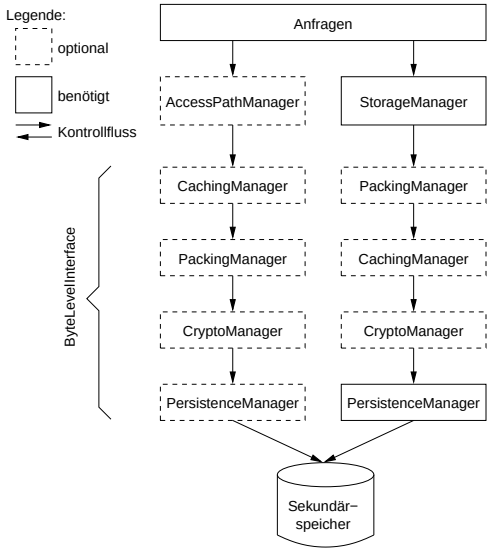

Abbildung 1: Beispiel-Aufbau des Speichermanagers

Um ein Tupel gemäß einer Bedingung zu erhalten, stellt die Anwendung bzw. die Abstraktionsebene oberhalb des Speichermanagers zunächst ein oder mehrere Schlüsselwerte zusammen, die eine Punktanfrage oder die Intervallgrenzen einer Bereichsanfrage bilden. Diese Schlüsselwerte werden dann dem `AccessPathManager`-Modul übergeben, das für das jeweilige Attribut der Relation registriert ist. Nachdem das Modul über die `ByteLevelInterface`-Schnittstelle die von ihm verwaltete Indexstruktur durchsucht hat, wird die Record-ID des ersten gefundenen Tupels zurückgegeben. Mit dieser ID kann nun das gewünschte Datentupel vom `StorageManager`-Modul angefordert werden.

5 Zusammenfassung und Ausblick

Nach einer einführenden Einordnung in den fachlichen Rahmen mobiler Geräte und Anwendungen wurden die Anforderungen an einen Speichermanager für mobile DBMS untersucht. Im Anschluss wurden einige bestehenden Ansätze zur Erfüllung dieser Anforderungen vorgestellt. Auf der Grundlage dieser Untersuchungen wurde daraufhin ein Konzept für einen modularen, leichtgewichtigen Speichermanager präsentiert.

Zukünftig ist geplant, den bestehenden Prototyp des Speichermanagers zu konsolidieren und Messungen des Laufzeitverhaltens durchzuführen. Weiterhin ist die Integration in ein Szenario des *Pervasive Computing* geplant: dabei sollen Daten kontextbasiert automatisch auf das Mobilgerät repliziert werden, wobei sich der Speichermanager durch Modulaustausch zur Laufzeit und Selbstoptimierung darauf einstellen muss.

Literatur

[BDS+92] Batory, D. S.; Das, D.; Singhal, V.; Sirkin, M.; Thomas, J.: Database Challenge: Single Schema Database Management Systems. Technischer Bericht Nr. CS-TR-92-47, University of Texas, Austin, Dezember 1992.

[BLR+97] Bohannon, P.; Lieuwen, D. F.; Rastogi, R.; Silberschatz, A.; Seshadri, S.; Sudarshan, S.: The Architecture of the Dali Main-Memory Storage Manager. *Multimedia Tools and Applications*, Band 4, Nr. 2, S. 115–151, 1997.

[CPP97] Cha, S. K.; Park, J.; Park, B. D.: Xmas: An Extensible Main-Memory Storage System. In Golshani, F.; Makki, K. (Hrsg.): *Proceedings of the 6th International Conference on Information and Knowledge Management (CIKM-97)*, S. 356–362. ACM Press, New York, November 10–14 1997.

[CW00] Chaudhuri, S.; Weikum, G.: Rethinking Database System Architecture: Towards a Self-Tuning RISC-Style Database System. In El Abbadi, A.; Brodie, M. L.; Chakravarthy, S.; Dayal, U.; Kamel, N.; Schlageter, G.; Whang, K.-Y. (Hrsg.): *VLDB 2000, Proceedings of 26th International Conference on Very Large Data Bases, September 10–14, 2000, Cairo, Egypt*, S. 1–10. Morgan Kaufmann Publishers, Los Altos, CA 94022, USA, 2000.

[Dat86] Database Architecture Framework Task Group (DAFTG) of the ANSI/X3/SPARC Database System Study Group: Reference Model for DBMS Standardization. *ACM SIGMOD Record*, Band 15, Nr. 1, S. 19–58, März 1986.

[DKL+94] Douglis, F.; Kaashoek, F.; Li, K.; Cáceres, R.; Marsh, B.; Tauber, J. A.: Storage Alternatives for Mobile Computers. In *First Symposium on Operating Systems Design and Implementation*, S. 25–37. Monterey, Californie, US, 1994.

[GSD97] Geppert, A.; Scherrer, S.; Dittrich, K. R.: KIDS: Construction of Database Management Systems based on Reuse. Technischer Bericht Nr. ifi-97.01, Department of Computer Science, University of Zurich, Januar 1997.

[Sto86] Stonebraker, M.: Inclusion of New Types in Relational Data Base Systems. In *Proceedings of the International Conference on Data Engineering,*, S. 262–269. IEEE Computer Society Press, Los Angeles, CA, feb 1986.

[TB95] Thomas, J.; Batory, D.: P2: An extensible lightweight DBMS. Technischer Bericht Nr. CS-TR-95-04, The University of Texas at Austin, Department of Computer Sciences, Austin, Texas, Februar 1995.