Fusion von Datenbanktechnologie und Tertiärspeichersystem im Bereich sehr großer multidimensionaler Array-Daten - Stand der Forschung -

Bernd Reiner, Karl Hahn

{reiner, hahnk}@in.tum.de
FORWISS (Bayerisches Forschungszentrum für wissensbasierte Systeme)
TU-München, Boltzmannstr. 3, D-85747 Garching bei München

Inhalt

1 Einleitung

Viele natürliche Phänomene und Messwerte, wie von Satelliten übertragene Daten der Atmosphäre, Klimasimulationen, Berechnungen der Strömungsdynamik, Simulationen chemischer Prozesse oder Simulationen in der Kosmologie und der Genetik, können als Arraydaten mit entsprechender Dimensionalität modelliert werden. Auch umfassende wissenschaftliche Experimente und Simulationen an Großrechnern, generieren multidimensionale Daten (Rasterdaten). Als eine gemeinsame Charakteristik gilt das immense Datenvolumen, das gespeichert werden muss. Der Datenumfang kann mehrere hundert Terabyte (10^{12} Byte), bis hin zu mehreren Petabyte (10^{15} Byte) erreichen. Bei CERN, einer europäischen Organisation für Nuklear-Forschung in der Schweiz, werden zum Beispiel jedes Jahr etwa 3 Petabyte neue Daten gespeichert. Ab dem Jahr 2007 beträgt der jährliche Datenzuwachs etwa 10 Petabyte [HS03]. Aufgrund des großen Volumens können diese Daten nicht mehr auf Festplatten gespeichert werden. Typischerweise werden die Daten als Dateien in automatisierten *Tertiärspeichersystem*en (TS-System), mit bis zu tausenden von Magnetbändern, optischen oder magnetooptischen Medien gespeichert. Die mittleren Zugriffszeiten dieser Art von Speichertechnologien sind, verglichen mit Festplatten, relativ langsam. Trotzdem sind Tertiärspeicher bei riesigem Datenvolumen der gegenwärtige und auch zukünftige Entwicklungsstand, da Speichermedien wie Magnetbänder einen viel günstigeren €/GByte-Preis als Festplatten aufweisen.

Anwendungsgebiete multidimensionaler Daten reichen von E-Commerce mit Satellitendaten, Data-Mining bei Klimamodellen und genetischen Daten, chemische Prozesssimulationen, Analysen im Bereich der Strömungsdynamik und kosmologische Simulationen. Abbildung 1 visualisiert zwei Beispieldaten.

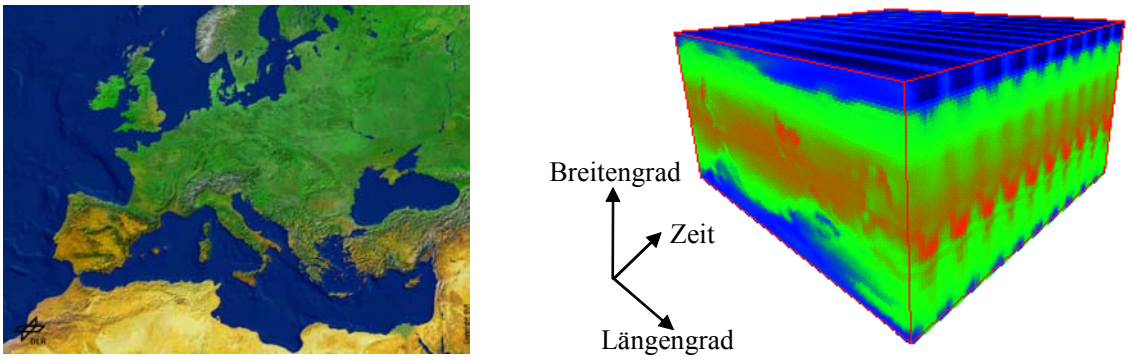


Abbildung 1: Links: Satellitenbild Europa (DLR); Rechts: Klimamodell der Erde (DKRZ)

Die linke Seite zeigt ein Satellitenbild von Europa, das vom Deutschen Luft- und Raumfahrtzentrum über ihre Internetplattform EOWEB (*Earth Observation WEB*, http://eoweb.dlr.de) zum Verkauf angeboten wird. Bei der rechten Darstellung handelt es sich um eine Simulation[1] zur Klimaerwärmung des Deutschen Klimarechenzentrums. Zu erkennen ist die atmosphärische Temperaturverteilung der Erde, zwei Meter über der Oberfläche, in einer Zeitspanne von 10 Jahren (2000 – 2009). Die Erdoberfläche befindet sich auf der linken Vorderseite des Datenwürfels (Längen- und Breitengrad). In den mittleren Breitengraden ist der Äquator mit höheren Temperaturen (rötlich gefärbt) zu sehen. Die Regionen des Nord- und Südpols wei-

[1] Simulation des Treibhauseffektes entsprechend dem IS92a Szenarium des IPCC'92 (*Intergovernmental Panel on Climate Change*) mit einem jährlichen Anstieg der Konzentration des Treibhausgases CO_2 um 1 %.

sen kältere Regionen auf, die blau dargestellt sind. Jahreszeitliche periodische Schwankungen über 10 Jahre können gut in der Zeitdimension erkannt werden.

1.1 Problem

Probleme bereiten nicht unbedingt das Speichern dieser großen Datenvolumen, sondern vor allem die Verarbeitung und der Zugriff auf die Daten. Diese Probleme sind einerseits in der Verfügbarkeit von Hard- und Software begründet. Aufgrund der fehlenden Rechnerleistung können teilweise die großen Datenmengen, z.B. von einer Marssonde gesendet, erst in einigen Jahren komplett ausgewertet werden. Andererseits liegen die Probleme vor allem in der unzureichenden Datenorganisation. Gewonnene Rohdaten werden direkt in Prozessordnung als Datei auf *Tertiärspeichermedien* (TS-Medien) abgelegt. Diese „naive" Speicherung in Prozessordnung bietet keine Information zum Inhalt des Datenarchivs und ist nicht für einen direkten, anwendungsorientierten Datenzugriff geeignet. Zunächst müssen zur Beantwortung wissenschaftlicher Fragestellungen, verarbeitbare Daten mit hohem Aufwand aus den auf Tertiärspeicher abgelegten Rohdaten-Files extrahiert werden. Dieser Vorgang wird auch als primäres Datenprozessing bezeichnet. Voraussetzung hierfür ist, dass der genaue Speicherort im Tertiärspeicherarchiv und die Datenstruktur der Rohdaten bekannt sind. Bearbeitungszeiten von einigen Wochen, bis hin zu Monaten, sind dabei nicht ungewöhnlich [Laut02]. Dies liegt auch daran, dass Wissenschaftler mit Dateien, Verzeichnissen und unterschiedlichen Datenformaten (oft komplex) umgehen, und ihre Daten meist selbst verwalten müssen (siehe Abbildung 2).

Abbildung 2: Problematik des Auffindens von Daten

Dabei werden von Wissenschaftlern nur etwa ein bis zehn Prozent der angeforderten Daten für die weitere Verarbeitung benötigt [GKDT02, LT01]. Dies begründet sich vor allem durch die Art der Speicherung der Daten als Dateien auf Tertiärspeicher und den verwendeten mangelhaften Retrieval-Werkzeugen. Es ist derzeit nicht möglich, auf Teilbereiche einer Datei zuzugreifen, die zum Beispiel auf Magnetband gespeichert sind. Das bedeutet allerdings, dass eine Datei die kleinste Zugriffsgranularität darstellt. Also wird gegenwärtig immer eine komplette Datei vom Tertiärspeicher geladen und zum Nutzer übertragen. Hier wird dann von den Analyse- und Simulationswerkzeugen der benötigte Teilbereich der Daten zur Weiterverarbeitung verwendet und der Rest verworfen. Abbildung 3 zeigt eine Auswahl typischer Zugriffe (schwarze Bereiche) auf multidimensionale Daten. In diesem Beispiel gehen wir von Tempe-

raturmesswerten aus, die durch Längengrad, Breitengrad und Höhe über Normal Null näher definiert werden.

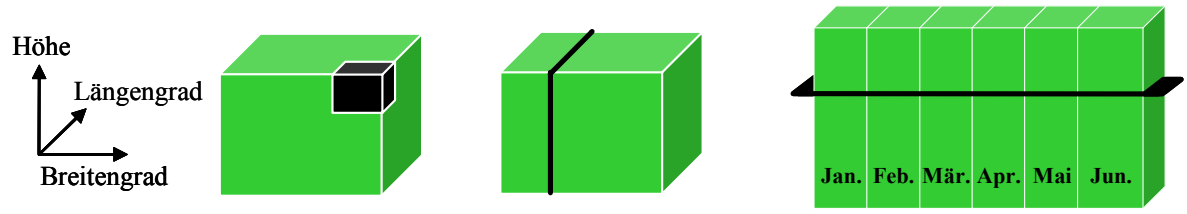


Abbildung 3: Eine Auswahl typischer Zugriffe auf multidimensionale Daten

In Abbildung 3 (linker Würfel) ist eine Ausschnittsbildung zu sehen. Eine typische Anfrage könnte wie folgt lauten: Gib mir alle Temperaturmesswerte von 48° 13' bis 48° 27' nördlicher Breite und von 11° 46' bis 11° 70' östlicher Länge in der Höhenstufe von 800 Meter bis 1000 Meter über Normal Null (kleiner schwarzer Würfel). Der Zugriff des mittleren Würfels setzt eine Dimension auf einen Wert fest, wodurch sich die Dimensionalität des Ergebnisses um eins reduziert. Eine denkbare Anfrage ist: Gib mir den kompletten Temperaturquerschnitt über die Höhen- und Längengrade bei 47° 80' nördlicher Breite. Beim rechten Beispiel der Abbildung 3 werden jeden Monat Temperaturmessungen durchgeführt und als separate Dateien gespeichert. So erhalten wir als weitere Dimension die Zeit. Dadurch können Anfragen einen Schnitt durch mehrere Dateien bedeuten. Eine Beispielanfrage lautet: Gib mir die Verteilung der Durchschnittstemperatur von Januar 2003 bis Juni 2003 auf einer Höhe von 800 Meter über Normal Null. Diese typischen Anfragen zeigen sehr deutlich, dass zur Berechnung der Ergebnismenge eine viel geringere Datenmenge notwendig ist, als in Wirklichkeit vom TS-Medium geladen wird. Gerade Berechnungen quer durch mehrere Dateien stoßen aufgrund des Datenvolumens rasch an die Grenzen der vorhandenen Hardware, wie z.B. Hauptspeicher. Das bedeutet, dass interessante und wichtige Simulationen und Analysen zurzeit nicht durchgeführt werden können.

Die Tertiärspeicherschicht in einem hierarchischen Speichersystem ist charakterisiert durch sehr große Datenvolumen und sehr hohe Anfragezeiten für wahlfreie Zugriffe (Random Access). Dies begründet sich durch die Verwendung zahlreicher günstiger Wechselmedien (z.B. Magnetbänder), die sich eine kleine Anzahl an teuren Schreib-/Lesestationen und Robotermechanismen teilen. Die hohe Zugriffszeit wird typischerweise dominiert durch die Medienwechselzeit (10 s – 40 s) und der mittleren Zugriffszeit[2] (27 s – 95 s). Festplatten sind bei der mittleren Zugriffszeit um einen Faktor 10^3 bis 10^4 schneller. Die Transferrate eines TS-Systems ist dagegen sehr gut und nur um etwa einen Faktor Zwei langsamer als die Transferraten von Festplatten. Wahlfreier Zugriff auf Daten, die auf Tertiärspeicher abgespeichert wurden, kann unakzeptable Verzögerung mit sich bringen, da Medien in den Schreib-/Lesestationen gewechselt werden müssen. Die Notwendigkeit des Medienwechsels wird durch die Datenverteilung auf den Bändern diktiert. Vernünftiges Platzieren der Daten auf TS-Medien ist aus diesen Gründen kritisch und kann sehr stark die Performance des Gesamtsystems beeinflussen. In den meisten Fällen werden die gewonnenen Rohdaten einfach in Prozessordnung auf den TS-Medien abgespeichert. Gerade eine Optimierung in diesem Bereich kann die Zugriffszeiten drastisch reduzieren. Das Problem der optimalen Platzierung von Daten auf Magnetbändern kann auf zwei Teilprobleme herab gebrochen werden. 1) Eine geschickte Verteilung der Daten auf unterschiedliche Medien, kann häufige Medienwechsel

[2] Mittlere Zugriffszeit bei TS-Systemen ist die Zeit für die Lokalisierung einer Datei und die Positionierung des Lesekopfes auf einem Tertiärspeicher-Medium.

vermeiden. 2) Eine optimale Anordnung der Daten innerhalb eines Mediums reduziert Spul- und Suchoperationen. Natürlich müssen Anfragen entsprechend dieser Datenverteilung optimiert werden, um eine Reduzierung der Zugriffszeiten zu erreichen. Typische Zugriffsmuster der Anwender können Einfluss auf die Organisation und Verteilung der Daten auf, bzw. zwischen TS-Medien nehmen.

1.2 Lösung

Um den hier erwähnten Defiziten zu begegnen, ist der Einsatz eines multidimensionalen Array Datenbanksystems ein viel versprechender Ansatz. Die Vorteile der Verwendung eines multidimensionalen Array Datenbanksystems zur Speicherung und Verwaltung der Daten liegen auf der Hand:

- Vermeidung von Redundanz und Inkonsistenz
- Mehrbenutzerbetrieb, Zugriffskontrolle und Zugriffsbeschränkungen
- Zentrale Verwaltung der Daten, Datenabstraktion und Datenunabhängigkeit
- Transaktionsverwaltung (ACID-Paradigma) und Recovery
- Multidimensionale Anfragesprache

Einziges Problem, das gegen den Einsatz von Datenbanksystemen spricht, ist das enorme Datenvolumen von mehreren hundert Terabyte bis hin zu mehreren Petabyte. Heute sind Gigabyte-Datenbanken üblich, Terabyte-Datenbanken die aktuelle Herausforderung und Petabyte-Datenbanken ein wichtiges Forschungsgebiet. Zum Vergleich: Bei heutiger Festplattengtechnologie (100 GByte) würde ein Petabyte an Daten über 10.000 Festplatten belegen. Ein nahe liegender Gedanke ist daher, die Vorteile der Datenverwaltung durch ein Datenbanksystem und die Datenhaltung auf TS-Medien zu verschmelzen.

Um ein beschleunigtes Datenretrieval zu erreichen, wird zumindest in Teilbereichen dazu übergegangen, eine Datenbank einzusetzen. So können Metadaten der gewonnenen Rohdaten verwaltet werden, um das Auffinden der Datensätze zu vereinfachen. Auch das manuelle Speichern einer geringen Anzahl ausgewählter multidimensionaler Datensätze als *Binary Large Objects* (BLOB) in der Datenbank wird vereinzelt praktiziert. Allerdings wird die Datenbank nur als Speichermanager verwendet, um komplette Datensätze (BLOBs) an Applikationen zu liefern. Es können aber keine direkten multidimensionalen Operationen auf dem Datenbank-Server durchgeführt werden. Diese Operationen werden von Analyse- und Simulationswerkzeugen übernommen. Wie bei der Speicherung als Datei auf Tertiärspeicher, können die BLOBs nur komplett angefasst und übertragen werden.

Nach der Darstellung einiger Grundlagen in Kapitel 2, folgt in Kapitel 3 der Stand der Forschung und Entwicklung im Bereich der Kopplung von DBMS und Tertiärspeichersystemen. Dabei werden folgende Systeme näher betrachtet und bezüglich der Speicherung und des Retrievals von großen Array-Daten bewert:

- SDM (***S**cientific **D**ata **M**anager*)
- MDMS & APRIL (**M**eta-***d**ata **M**anagement **S**ystem & **A** **P**arallel **R**un-Ti**m**e **L**ibrary for Tape-Resident Data*)
- StorHouse/RM (*StorHouse **R**elational **M**anager*)
- CERA (***C**limate and **E**nvironmental Data **R**etrieval and **A**rchiving System*)
- HEAVEN (***H**ierarchische Spe**i**cher- und **A**rchi**v**ierungsumgebung für multidim**e**nsionale Array Date**n**bank Management Systeme*)

2 Grundlagen

Zunächst erfolgt eine Eingliederung von Tertiärspeicher in die Speicherhierarchie eines Datenverarbeitungssystems. Danach wird die Funktionsweise hierarchischer Speichermanagement Systeme vorgestellt. Abschließend folgen Möglichkeiten der Kopplung von DBMS und Tertiärspeichersystemen.

2.1 Speicherhierarchie

Grundsätzlich sind Speichermedien charakterisiert durch Zugriffszeit[3], Datenrate und Speicherkapazität. Dabei besteht ein prinzipieller Konflikt zwischen der Minimierung der Zugriffszeit und der Maximierung der Speicherkapazität. In Rechnersystemen werden unterschiedliche Speichertechnologien zu Speicherhierarchien kombiniert, um einen Kompromiss aus schnellem Zugriff und großen Speicherkapazitäten bei angemessenen Kosten zu erreichen. Abbildung 4 zeigt eine solche Speicherhierarchie, dargestellt als Pyramide. An der Spitze der Pyramide ist der Speicher mit der geringsten Zugriffszeit, geringsten Kapazität und den höchsten Kosten angesiedelt. Je weiter unten sich Speichertechnologien in der Pyramide befinden, desto größer und günstiger, aber auch langsamer, wird dieser Speicher.

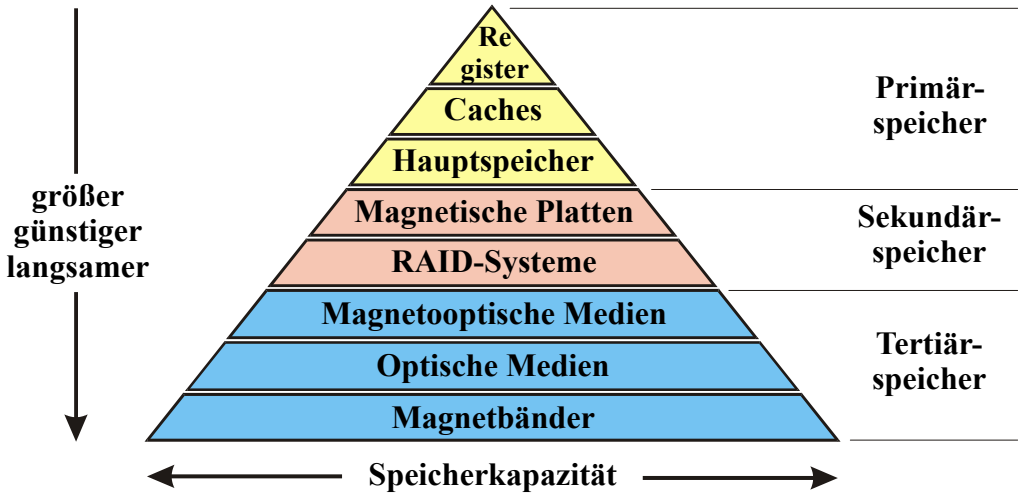


Abbildung 4: Speicherhierarchie eines Datenverarbeitungssystems

Generell lassen sich Primär-, Sekundär- und Tertiärspeicher unterscheiden. Diese Einteilung wird jedoch in der Literatur nicht einheitlich verwendet. Teilweise werden Tertiärspeicher nicht explizit aufgeführt, sondern zu den Sekundärspeichern gezählt. Als Primärspeicher werden alle Speicher mit wahlfreiem Zugriff (Random Access) bezeichnet, auf die der Prozessor direkt mit voller Geschwindigkeit zugreifen kann. Dazu zählen Register eines Prozessors, Caches und der Hauptspeicher bzw. Arbeitsspeicher. Primärspeicher bieten schnellen Zugriff auf Daten, sind aber hinsichtlich ihrer Kapazität begrenzt. Hintergrundspeicher mit index-sequentiellem[4] (quasi-wahlfreiem) Zugriff, wie magnetische Festplatten oder RAID-Systeme (*Redundant Arrays of Independent Disks*), werden als Sekundärspeicher eingeordnet. Diese Speicher verfügen über große Kapazitäten, weisen allerdings gegenüber Primärspeicher einen bis Faktor 10^5 langsameren Zugriff auf. Dieser gravierende Unterschied in der Zugriffs-

[3] Zeit für die Lokalisierung und Zeit des Speicherzugriffes selbst (Lesen gespeicherter Information bzw. Schreiben neuer Information).

[4] Auf größere Speicherblöcke kann wahlfrei zugegriffen werden. Auf einzelne Speicherzellen innerhalb eines Speicherblockes kann nur sequentiell zugegriffen werden.

zeit wird auch als Zugriffslücke bezeichnet. Speichertechnologien mit weit höheren Zugriffszeiten und Speichermedien, die nicht direkt zugreifbar sind, gliedern sich in die Kategorie der Tertiärspeicher ein. Nicht direkt zugreifbar bedeutet, dass Medien manuell bedient werden müssen oder in robotergesteuerten Bibliotheken organisiert sind und erst bei einem Zugriff in die entsprechenden Schreib-/Lesestationen bewegt werden. Die Zugriffslücke zwischen Sekundär- und Tertiärspeicher erreicht ebenfalls einen Faktor von 10^5. Zu den TS-Medien zählen *magnetooptische* (MO) Medien, optische Medien und Magnetbänder. Typische Zugriffszeiten und Kapazitäten diverser Speichertechnologien werden in Tabelle 1 verglichen (Stand Juni 2003).

Speicherart		**Typische Zugriffszeit**	**Typische Kapazität**
Primärspeicher	Register	≤ 1 ns	256 Byte – 1 KByte
	Caches	5 – 10 ns	128 KByte – 4 MByte
	Hauptspeicher	8 – 20 ns	64 MByte – 1 GByte
Sekundärspeicher	Magnetische Platten	5 – 15 ms	60 GByte –180 GByte
	RAID-System	5 – 15 ms	100 GByte – 4 TByte
Tertiärspeicher	MO-Medien	20 – 50 ms	128 MByte – 9,5 GByte
	Optische Medien	80 – 200 ms	650 MByte – 4,7 GByte
	Magnetbänder	10 s – 5 min	10 GByte – 220 GByte

Tabelle 1: Typische Zugriffszeiten und Kapazitäten verschiedener Speichertechnologien

Bei genauer Betrachtung fällt auf, dass Transferraten moderner Bandtechnologien, wie Super-DLT 320, LTO Ultrium-2, AIT-3 oder Mammoth 2, nahe an Transferraten von Festplatten (30 – 60 MB/s) herankommen und teilweise auch erreichen (Tabelle 2). Bandlaufwerke sind bei der Transferrate um einen Faktor 1,5 bis 5 langsamer als Festplatten. Vergleicht man allerdings die mittlere Zugriffszeit von Festplatten (5 ms – 12 ms) und Bandlaufwerken (27 s – 95 s), macht sich der gravierende Unterschied zwischen index-sequentiellem und sequentiellem Zugriff mit einem Faktor von 10^3 bis 10^4 bemerkbar. Muss das Magnetband noch extra in die Ladestation des Bandlaufwerkes eingelegt werden (meist automatisiert), so addiert sich zur mittleren Zugriffszeit für den ersten Zugriff noch die mittlere Medienladezeit (Tabelle 2).

Kennwerte	**DLT 8000**	**Super DLT 320**	**LTO Ultrium-2**	**AIT-3**	**Mammoth 2**	**DDS-4**
Kapazität, native	40 GB	160 GB	200 GB	100 GB	60 GB	20 GB
Kapazität, komprimiert	80 GB	320 GB	400 GB	260 GB	120 GB	40 GB
Transferrate, native	6 MB/s	16 MB/s	40 MB/s	12 MB/s	12 MB/s	2 MB/s
Mittlere Zugriffszeit	60 s	70 s	95 s	27 s	47 s	55 s
Mittlere Medienladezeit	37 s	40 s	20 s	10 s	17 s	24 s

Tabelle 2: Technologievergleich unterschiedlicher Magnetbänder (Stand 2003)

2.2 Hierarchische Speichermanagement Systeme

Im professionellen Umfeld ist es notwendig, automatisierte Systeme einzusetzen, um Daten auf TS-Medien, wie z.B. Magnetbänder, auszulagern und im Bedarfsfall zurückzuholen. Manuelles Eingreifen ist sehr arbeits- und wartungsintensiv und ab einer gewissen Datenmenge bzw. Medienanzahl nicht mehr vertretbar. Mit den *hierarchischen Speichermanagement* (HSM) Systemen ist seit den achtziger Jahren eine automatisierte Lösung der Datenspeicherung auf TS-Medien vorhanden. Ein HSM-System erweitert lokale Festplatten um die Kapazität einer Vielzahl tertiärer Speichermedien und kann als normales Dateisystem mit nahezu unbegrenzter Speicherkapazität angesehen werden. In Wirklichkeit ist das virtuelle Dateisystem eines HSM-Systems unterteilt, in einen begrenzten Festplattencache, auf dem der Anwender arbeitet (Schreiben und Lesen von Daten) und einem angegliederten TS-System, mit einer robotergesteuerten TS-Medienbibliothek. Je nach Wunsch und somit Ausstattung des HSM-Systems, können beliebige tertiäre Speichermedien, wie Magnetband, MOD, CD, DVD, usw. verwendet werden. Abbildung 5 zeigt die Architektur eines solchen hierarchischen Speichermanagement Systems. Es lässt sich gut die Speicherhierarchie erkennen, bestehend aus Festplatte (Cache) und TS-System. Ein HSM-System erweitert somit die bereits bestehende Speicherhierarchie der Primär- und Sekundärspeicher um die Komponente der Tertiärspeicher.

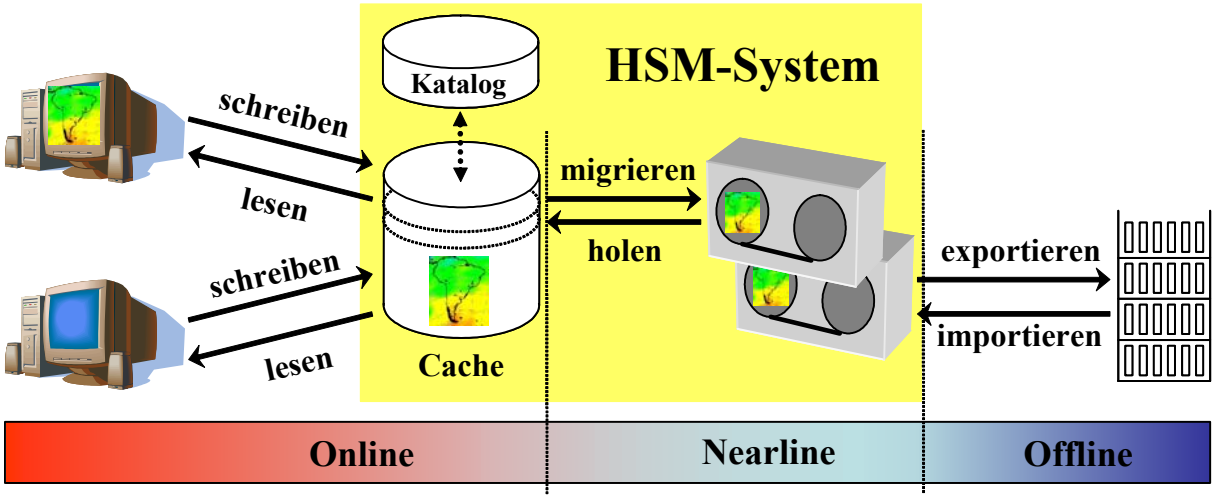


Abbildung 5: Architektur eines hierarchischen Speichermanagement Systems

Hierarchisches Speichermanagement wird von Herstellern in unterschiedlichen Ausprägungen angeboten. Die wesentliche Aufgabe jedes HSM-Systems ist die automatische Migration der Daten vom Festplattencache auf ein tertiäres Medium und das Zurückholen im Bedarfsfall. Die gesamte Komplexität eines HSM-Systems bleibt dem Nutzer verborgen. Lediglich die Zeitverzögerung beim Lesezugriff auf die ausgelagerten Daten, ist im Vergleich zu lokal gespeicherten Daten sehr viel größer. Aus der Sicht des Nutzers, ist das Schreiben von Daten in ein HSM-System ähnlich performant, wie das Speichern von Daten auf der lokalen Festplatte, da die Daten nur in den Festplattencache geschrieben werden. Das Migrieren der Daten, z.B. auf Magnetband, erfolgt anschließend automatisch durch das HSM-System. Ein HSM-System gaukelt einem Nutzer, bzw. einer Applikation vor, dass Daten immer lokal (HSM-Cache) gespeichert sind und somit sofort verfügbar sind. Ein Schlüsselkonzept eines HSM-Systems, ist der Balanceakt zwischen schneller Zugriffsgeschwindigkeit und günstigem Speicherplatz.

In Bezug auf die Zugreifbarkeit (Verfügbarkeit) von Daten lassen sich die Bereiche Online, Nearline und Offline abgrenzen. Daten, die auf der Festplatte gespeichert werden, auf die sehr schnell zugegriffen werden kann, werden als Online-Daten bezeichnet. Wurden Daten auf ein TS-Medium migriert, die sich in einer robotergesteuerten Bibliothek befinden, so spricht man von Nearline-Daten. Die Zugriffszeit auf diese Daten ist sehr viel langsamer als auf Online-Daten. Allerdings ist keine Nutzerinteraktion für das Datenretrieval notwendig. Offline-Daten hingegen sind auf TS-Medien gespeichert, aber nicht in automatisierten Bibliotheken integriert, sondern zum Beispiel in externen Regalen gelagert. Um diese Daten zu nutzen, muss ein Systemadministrator, nach Aufforderung des HSM-Systems, die Magnetbänder manuell in das HSM-System einfügen. Um auf Offline-Daten zuzugreifen, ist also eine Nutzerinteraktion notwendig.

2.3 Kopplung von Tertiärspeichersystem und DBMS

Dieses Kapitel beschäftigt sich mit den Möglichkeiten der Kopplung von Tertiärspeichersystemen und DBMS. Zunächst sind in Abbildung 6 die Möglichkeiten der Verwendung von Tertiärspeicher dargestellt. Links ist die direkte Verwendung von TS-Systemen über ein Massenspeicher-Dateisystem zu erkennen. In dieser Weise werden HSM-Systeme realisiert. Dateien, die Anwender oder Applikationen im Dateisystem speichern, werden nach bestimmten Regeln auf TS-Medien geschrieben, bzw. bei Dateizugriffen geladen (vergleiche Kapitel 2.2). Um DBMS mit Tertiärspeicherkomponenten zu koppeln, sind grundsätzlich zwei Möglichkeiten zu unterscheiden: Einerseits können DBMS über ein Dateisystem mit Tertiärspeicher gekoppelt werden (Abbildung 6, Mitte). Diese Systeme nutzen das darunter liegende Dateisystem, um einen transparenten Zugriff zu den auf Tertiärspeicher abgelegten Daten zu realisieren (durch HSM-System). Das Dateisystem kontrolliert die Verschiebung der Daten zum/vom Festplatten-Cache und zum/vom Tertiärspeicher. Die zweite Möglichkeit der Tertiärspeicheranbindung zeigt die rechte Seite der Abbildung 6. In diesem Fall ist das DBMS direkt mit dem TS-System gekoppelt. Der Festplatten-Cache wird also vom DBMS kontrolliert. Dadurch kann das DBMS semantische Informationen der Daten ausnutzen, um eine bessere I/O-Peformance, zum Beispiel durch optimierte Caching-Strategien, zu erreichen.

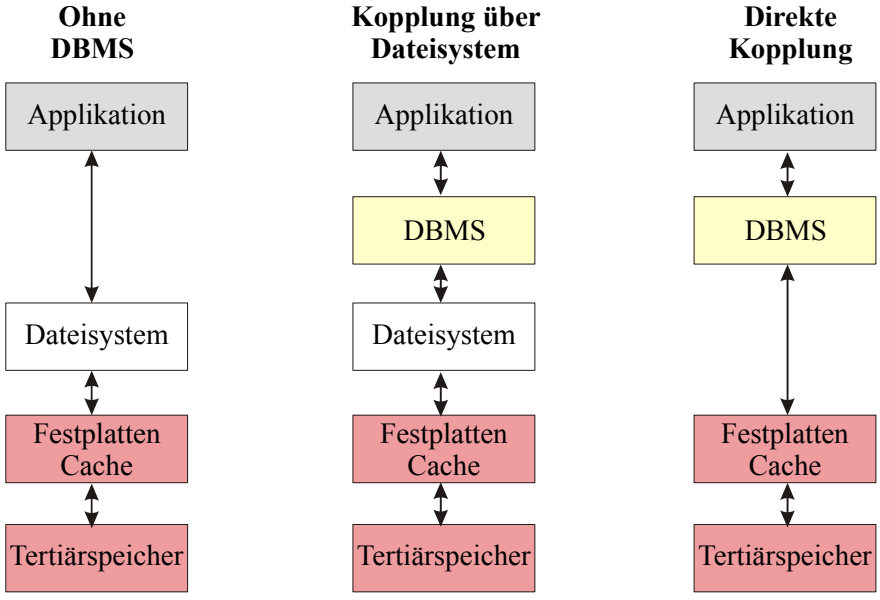


Abbildung 6: Möglichkeiten der Anbindung von Tertiärspeicher

3 Stand der Forschung und Entwicklung

Momentan unterstützt keines der auf dem Markt führenden DBMS wie Oracle, IBM/DB2 Universal Database und Microsoft SQL-Server, eine direkte Anbindung an Tertiärspeicher (ausgenommen Backup). Um einen Überblick über den Stand der Forschung und Entwicklung im Bereich der Anbindung von Tertiärspeicher an DBMS zu erhalten, werden hier beispielhaft Entwicklungen vorgestellt. Es werden folgende Systeme näher beschrieben, wobei bei den ersten beiden Systemen ein DBMS nur zur Verwaltung der Metadaten verwendet wird:

- SDM (***S**cientific **D**ata **M**anager*)
- MDMS & APRIL (**M**eta-***d**ata **M**anagement **S**ystem & **A** **P**arallel **R**un-T**i**me **L**ibrary for Tape-Resident Data*)
- StorHouse/RM (*StorHouse **R**elational **M**anager*)
- CERA (***C**limate and **E**nvironmental Data **R**etrieval and **A**rchiving System*)
- HEAVEN (***H**ierarchische Spe**i**cher- und A**r**chi**v**ierungsumgebung für multidim**e**nsionale Array Date**n**bank Management Systeme*)

Die hier beschriebenen Systeme werden vor allem aus der Sicht der Speicherung und des Datenretrievals von multidimensionalen Array-Daten beleuchtet und bewertet. Die Systeme wurden entsprechend ihrer Eignung für die Speicherung und das Retrieval multidimensionaler Array-Daten (wenig bis sehr gut geeignet) sortiert.

3.1 Scientific Data Manager (SDM)

SDM (*Scientific Data Manage*) realisiert keine komplette Verschmelzung eines DBMS und eines Tertiärspeichersystems, da das DBMS ausschließlich zur Verwaltung der Metadaten verwendet wird. Die multidimensionalen Array Daten werden nach wie vor als Dateien auf TS-Medien gespeichert. Allerdings wird das Auffinden einzelner Datensätzen durch die Metadaten wesentlich vereinfacht. Abbildung 7 zeigt den prinzipiellen Aufbau der Architektur, des SDM-Systems.

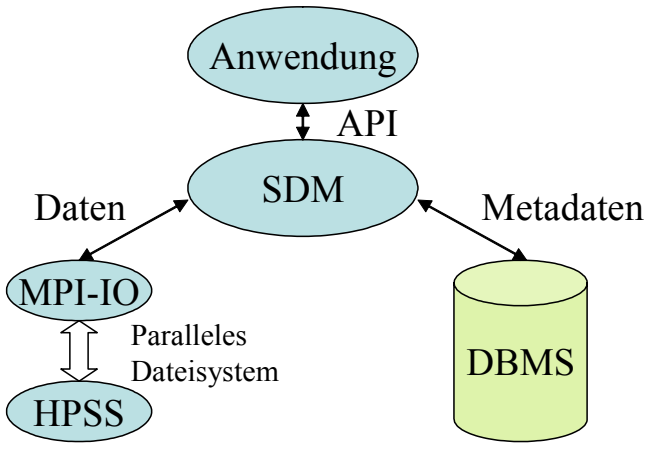


Abbildung 7: Prinzipieller Aufbau der SDM-Architektur

Anwendungen werden über eine API (*Application Programming Interface*) an das SDM-System angebunden. Über diese API können datenspezifische Metadaten innerhalb eines DBMS gespeichert werden. Momentan ist eine Anbindung an die Datenbanken MySQL und PostgreSQL implementiert. Um Daten auf Tertiärspeicher zu speichern verfügt SDM über

eine MPI-IO (*Message Passing Interface*) Schnittstelle. MPI-IO, das im MPI-2 Standard definiert wurde, realisiert die parallele Ein-/Ausgabe, für die Anbindung an ein Dateisystem [TGL99]. Dieses parallele Dateisystem fungiert weiterhin als Cache für das HSM-System HPSS (*High Performance Storage System*). Durch HPSS [HPSS03] wird eine Skalierbarkeit in Bezug auf das Datenvolumen erreicht, da neue Ressourcen angebunden werden können. Allerdings ist SDM nicht einfach hinsichtlich neuer Benutzer und neuer Applikationen erweiterbar.

Aus der Sicht der Datenspeicherung auf TS-Medien, werden drei Möglichkeiten unterschieden. Im ersten Fall wird jeder erzeugte Datensatz, der zu einem bestimmten Zeitpunkt erzeugt wird, als einzelne Datei mittels HPSS auf ein TS-Medium geschrieben. Diese Methode kann bei einer hohen Anzahl an unterschiedlichen Dateien, aufgrund der häufigen Operationen zum Öffnen und Schließen der Dateien, sehr kostenintensiv sein. Die zweite Möglichkeit bietet eine Gruppierung von mehreren Daten zu einem Datensatz, der auf das TS-Medium geschrieben wird. Dies reduziert die Kosten für die Operationen zum Öffnen und Schließen von Dateien, erhöht allerdings die Verwaltungskosten. Es muss ein Offset in einer Ausführungstabelle gespeichert werden, der angibt, wo sich die einzelnen Daten innerhalb einer Datei befinden. Die letzte Möglichkeit sammelt mehrere Datengruppen und schreibt diese gemeinsam zu bestimmten Zeitpunkten auf das TS-Medium. Dies reduziert weiterhin die I/O-Kosten. Allerdings erhöhen sich bei den letzten beiden Möglichkeiten die Kosten beim Retrieval der Daten. Es muß immer eine gesamte Datengruppe geladen werden, obwohl teilweise nur ein Datensatz benötigt wird. Eine weiterführende Betrachtung des SDM-Systems ist in [NTC00] zu finden.

Entsprechend der Datenhaltung verwirklicht SDM eine Anbindung an TS-Systeme ohne DBMS (vergleiche Abbildung 6). Da nur die Verwaltung der Metadaten über ein DBMS realisiert wird, ist dieses System für die Lösung der in Kapitel 1 genannten Probleme nicht ausreichend. Es bestehen nur Techniken, um Datensätze zu kombinieren und um I/O-Kosten zu sparen. Allerdings ist gerade für multidimensionale Daten dies ein Weg in die falsche Richtung. Hier müssen Möglichkeiten geschaffen werden, um große Datenobjekte zu unterteilen, damit Bereichsanfragen effizient bearbeiten werden können. Weiterhin wurden keine weiteren Konzepte zur Optimierung, wie Clustering, Scheduling oder Caching mit eingebracht. Durch HPSS wird allerdings eine Skalierbarkeit in Bezug auf das Datenvolumen erreicht, da neue Ressourcen angebunden werden können. Allerdings ist SDM nicht einfach, hinsichtlich neuer Benutzer und neuer Applikationen, erweiterbar.

3.2 MDMS & APRIL

Ebenso wie bei SDM wird bei dem kombinierten Einsatz von MDMS (*Meta-data Management System*) und APRIL (*A Parallel Run-Time Library for Tape-Resident Data*) ein DBMS nur zur Speicherung der Metadaten verwendet. Dies ist ein erster Schritt der Fusion von DBMS und Tertiärspeichersystem zur Optimierung der Datenzugriffe. Wobei MDMS die Funktionalität der Speicherung von Metadaten in einer DBMS (PostgreSQL) darstellt und APRIL die Anbindung einer Applikation an ein HSM-System realisiert. MDMS und APRIL wurden als dreischichtige Architektur entwickelt, die in Abbildung 8 zu sehen ist. Diese Architektur ist ähnlich aufgebaut wie das SDM-System (vergleiche Abbildung 7). Ebenso wie bei SDM wird HPSS (*High Performance Storage System*) als hierarchisches Speichersystem (HSS) eingesetzt und MPI-IO (*Message Passing Interface*) als Schnittstelle zum Datentransfer verwendet. Durch HPSS und MPI-IO konnte eine parallele Ein-/Ausgabe verwirklicht werden.

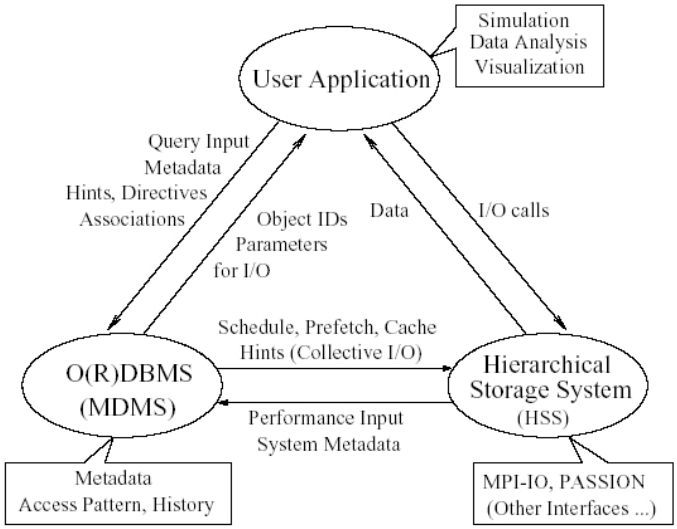


Abbildung 8: Dreischichtige Architektur, bestehend aus Applikation, Metadaten-DBMS und hierarchischem Speichersystem [SLCM00]

Der wesentliche Vorteil von MDMS und APRIL gegenüber SDM ist die Möglichkeit, große multidimensionale Daten in gleichförmige Teilobjekte (in jeder Dimension) aufzuspalten und als separate Dateien auf Tertiärspeicher abzulegen (Abbildung 9, links). Dies reduziert bei Bereichsanfragen die zu übertragene Datenmenge enorm und trägt somit erheblich zur Steigerung der Performance bei (Abbildung 9, rechts). Anwender können durch Angabe einer oberen und unteren Grenze Teilbereiche (Rechtecke bzw. Quader) anfragen. Eine Anfrage wird von der Applikation an MDMS weitergeleitet. Dort wird die Anfrage ausgewertet und die notwendigen Teilobjekte vom HSS angefordert. Die Teilobjekte selbst werden direkt vom HSS zur Applikation übertragen.

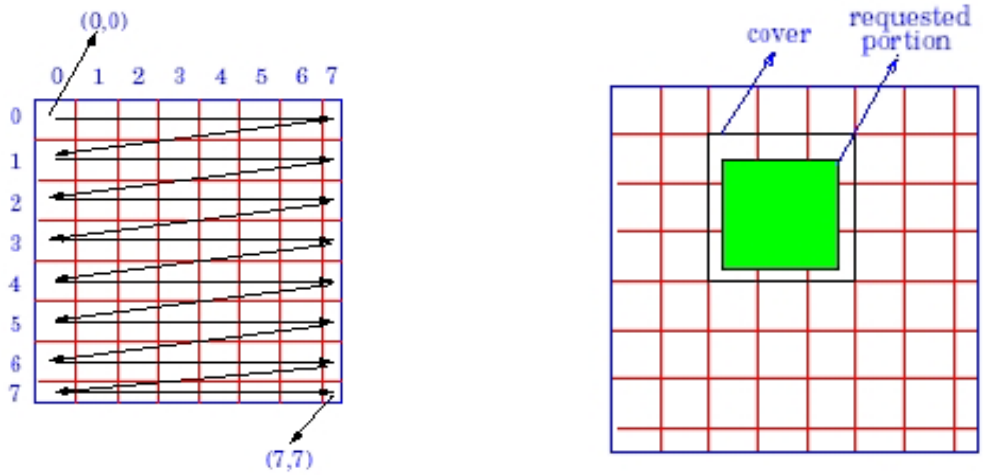


Abbildung 9: Links: Ein Objekt, aufgeteilt in 64 Teilbereiche
Rechts: Bereichsanfrage an ein zweidimensionales Objekt

Als weitere Konzepte zur Steigerung der Performance wurden Caching, Scheduling und Clustering entwickelt. Vom HSS übertragene Objekte bzw. Teilobjekte werden zunächst auf

der Festplatte gespeichert und dann zur weiteren Verarbeitung in den Hauptspeicher geschrieben. Diese Objekte werden erst beim Einlagern neuer Daten verdrängt. Dadurch wurde ein rudimentäres Caching-Konzept realisiert. Beim Eintreffen mehrerer Anfragen wird auch ein einfaches Scheduling dieser Anfragen durchgeführt. Das bedeutet, die Anfragen werden hinsichtlich eines effizienteren Datenretrievals vorsortiert und an das HSS weitergeleitet.

Wird die Speicherung der einzelnen Teilobjekte auf TS-Medien untersucht, so werden diese entlang der Dimensionen serialisiert und auf Band gespeichert. Dabei wird immer zunächst die erste Dimension, dann die Zweite, usw. verwendet, um die Reihenfolge auf dem TS-Medium zu bestimmen (Abbildung 9, links). Weiterführende Information über APRIL und MDMS sind in [CKNM99, Memi00, MKCT00, SLCM00] aufgeführt.

Aus der Sicht der Daten entspricht SDM einer Anbindung an TS-Systeme ohne DBMS (vergleiche Abbildung 6). Da nur die Verwaltung der Metadaten über ein DBMS realisiert wird, ist dieses System für die Lösung der in Kapitel 1 genannten Probleme nicht ausreichend. Die bekannten Vorteile einer Datenhaltung mit DBMS fehlen komplett. Ein herausragender Vorteil von MDMS und APRIL, ist allerdings die Möglichkeit, auf Teilobjekte von multidimensionalen Daten zuzugreifen. Es wird keine richtige multidimensionale Anfragesprache zur Verfügung gestellt. Die implementierten Caching und Scheduling Methoden sind nur rudimentär ausgeprägt und verbesserungswürdig. Beim verwendeten Clustering werden keine typischen Zugriffsmuster berücksichtigt. Dies bedeutet, dass oft keine optimale Reihenfolge der Teilobjekte auf TS-Medien erreicht wird: Häufige Spuloperationen werden benötigt, um die einzelnen Teilbereiche zu laden. Somit kann dieses Clustering von Daten durchaus verbessert werden. Durch HPSS wird eine Skalierbarkeit in Bezug auf das Datenvolumen erreicht, da neue Ressourcen angebunden werden können. Allerdings ist SDM nicht einfach erweiterbar, hinsichtlich neuer Benutzer und neuer Applikationen.

3.3 FileTek StorHouse/RM

Das System Storhouse der Firma FileTek realisiert eine Komplettlösung für das Speichern, Verwalten und Auffinden von großen Datenvolumen und bietet neben der Funktionalität eines HSM-Systems die Möglichkeit, DBMS anzubinden. StorHouse bietet Schnittstellen zu einigen kommerziellen DBMS, wie Oracle, IBM/DB2 Universal Database, Microsoft SQL-Server und Sybase. Es ist möglich, mit der DB-Erweiterung von StorHouse, Daten unabhängig von deren Speicherort mittels SQL (Structured Query Language) abzufragen und zu bearbeiten. Im Wesentlichen können die zwei Komponenten StorHouse/SM (*StorHouse Storage Manager*) und StorHouse/RM (*StorHouse Relational Manager*) unterschieden werden.

Der Speichermanager StorHouse/SM realisiert die Funktionalität eines HSM-Systems und verwaltet somit eine mehrschichtige Speicherarchitektur aus Hauptspeicher, RAID, Festplatten und Tertiärspeichermedien. Mittels der Verwaltungskomponente VSAC (*Volume Storage Allocation and Control*) des StorHouse/SM, lassen sich unterschiedliche Strategien für die Speicheranforderung und die Migrationspfade von Daten definieren. Unter Migrationspfad versteht man Wege, die Daten in der Speicherhierarchie gehen können. Es ist möglich, Daten zunächst auf der Festplatte zu speichern und nach einer gewissen Zeit (Vulnerability Time Factor) auf langsamere, aber kostengünstigere Tertiärspeichermedien auszulagern. Um die Verfügbarkeit der Daten zu erhöhen, können auch Kopien der Daten auf unterschiedlichen Hierarchiestufen vorgehalten werden. Die Verwaltungskomponente erlaubt auch, Daten nach bestimmten Zugriffsmustern zu kombinieren, um sie gemeinsam auf TS-Medien zu schreiben. Durch die Generierung von Dateifamilien, wird die Speichergranularität erhöht. Dadurch wird

versucht, bei Zugriffen die notwendigen Suchoperationen zu reduzieren. StorHouse/SM unterstützt weitere Zugriffsoptimierungen, wie das Zusammenfassen von Zugriffen auf ein einzelnes TS-Medium, um hohe Medienwechselkosten zu minimieren. Auch die Reihenfolge der Zugriffe auf ein Magnetband, werden in der Weise optimiert, dass ein serielles Lesen der Daten ermöglicht wird. Dies reduziert teure Such- und Spulkosten.

Die Anbindung kommerzieller DBMS an Tertiärspeicher erfolgt durch den StorHouse Relational Manager. Diese Komponente StorHouse/RM stellt ein eigenständiges relationales DBMS dar, das mit Hilfe des StorHouse/SM die Anbindung an Tertiärspeicher realisiert. Die Anbindung von Oracle (ab Version 8i) an StorHouse/RM, wird mittels Datenbanklinks und einer ODBC (*Open Database Connectivity*) Schnittstelle bewerkstelligt. Dies bedeutet, dass Tabellen, die in StorHouse liegen, in Oracle mittels Zeiger repräsentiert werden. Eingehende Anfragen werden von Oracle geparst. Entsprechende Unteranfragen werden automatisch an StorHouse/RM weitergeleitet, wenn dies notwendig ist. Die Verbindung von Oracle zu StorHouse/RM, ist für den Anwender transparent. Abbildung 10 zeigt das Prinzip der Anfragebearbeitung mittels Oracle in Kombination mit StorHouse/RM.

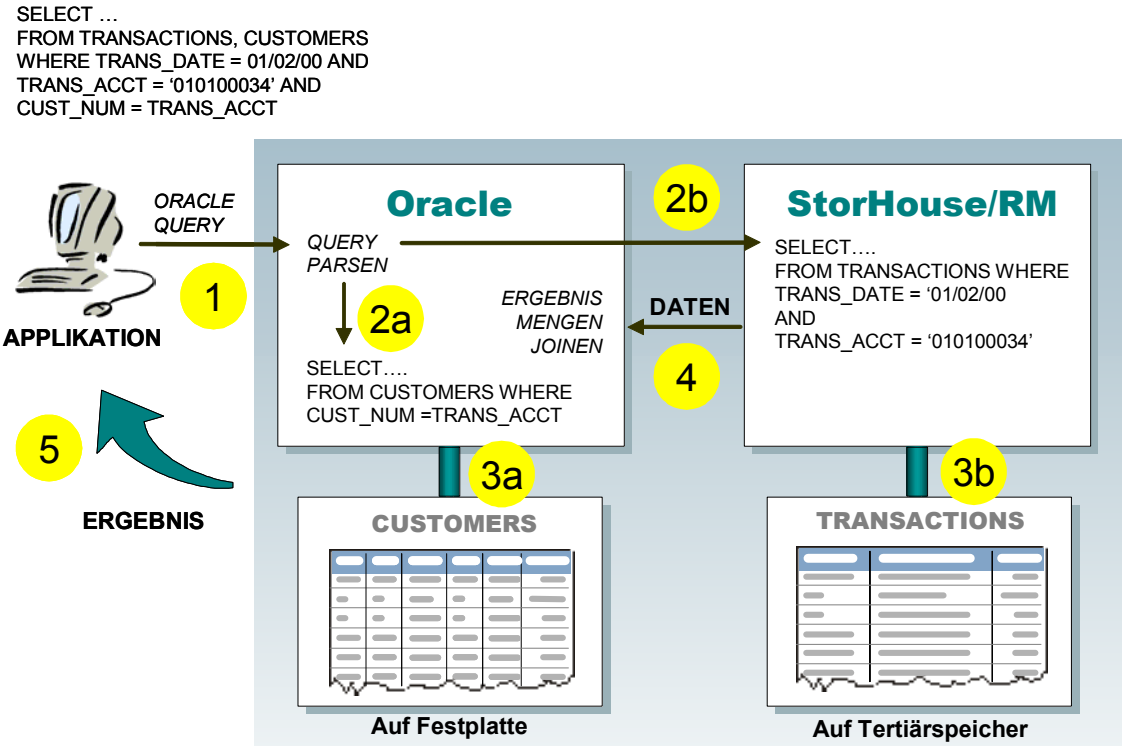


Abbildung 10: Anfragebearbeitung mittels Oracle und StorHouse/RM [File03a]

Zur Beantwortung der Anfrage eines Anwenders (1) werden die beiden Tabellen CUSTOMERS und TRANSACTIONS benötigt. Oracle parst nun diese Anfrage und stellt fest, dass die Tabelle TRANSACTIONS über einen Datenbanklink ausgelagert wurde. Daraufhin werden aus der ursprünglichen Query Unteranfragen generiert, die entsprechend verteilt werden (2a, 2b). Oracle selbst verarbeitet die Unteranfrage an die Tabelle CUSTOMERS (3a) und StorHouse/RM bearbeitet die Unteranfrage an die Tabelle TRANSACTIONS (3b). Nach der Ausführung gibt StorHouse/RM das Teilergebnis der Unteranfrage an Oracle zurück (4). Oracle bildet nun einen Verbund (Join) der beiden Teilergebnisse und liefert das Gesamtergebnis der Anfrage an die Applikation zurück (5).

Die logische und physikalische Struktur von StorHouse/RM sind in Abbildung 11 zu erkennen. Aus logischer Sicht können an StorHouse/RM mehrere relationale DBMS über Datenbanklinks angebunden werden (Abbildung 11, links). Ein solches angebundenes DBMS, kann aus beliebig vielen Nutzer-Tablespaces bestehen. Die einzelnen Tablespaces können wiederum eine beliebige Anzahl an Benutzertabellen mit den entsprechenden Daten und Indizes enthalten. Zusätzlich wird von StorHouse/RM für jede angebundene Datenbank eine System-Tablespace angelegt mit den notwendigen Systemtabellen. Diese Systemtabellen enthalten Metadaten und die zugehörigen Indizes und Logdaten. Der hier beschriebene Aufbau entspricht der klassischen Struktur eines relationalen DBMS.

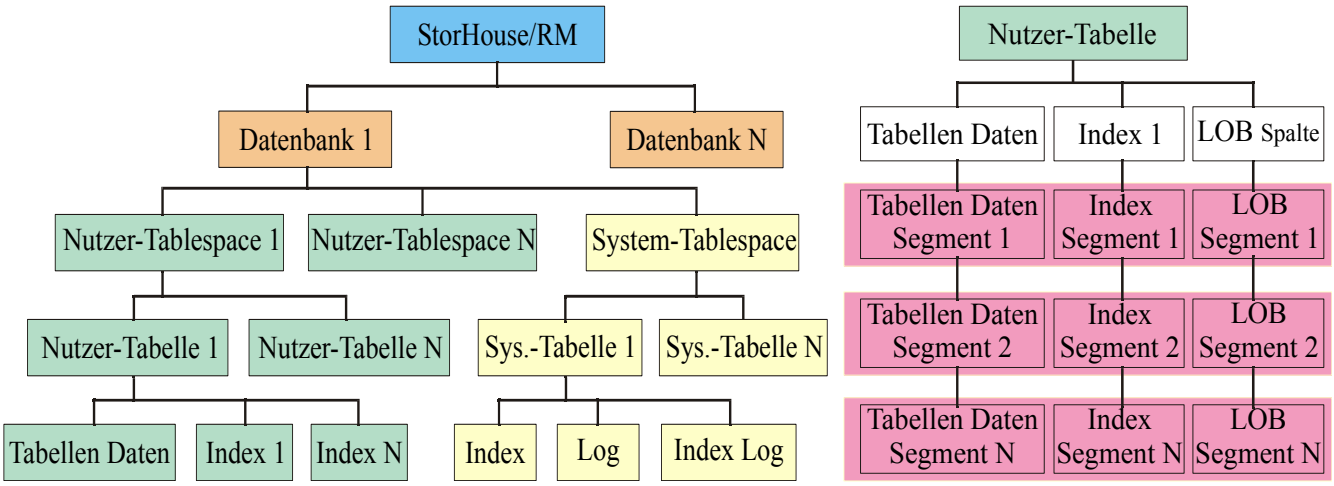


Abbildung 11: Logische und physikalische Struktur von StorHouse/RM [CKKB01]

Physikalisch werden Nutzertabellen und Indizes als StorHouse/SM-Dateien in der vorhandenen Speicherhierarchie gespeichert. Vergleicht man diese Art der Anbindung eines DBMS an Tertiärspeicher mit Abbildung 6, entspricht dies einer Kopplung über ein bestehendes Dateisystem (Bildmitte).

Die jeweilige Nutzer-Tablespace legt fest, welche Speichermedien für die Datentabellen und die Indices verwendet werden. Zusätzlich wird der Migrationspfad vorab festgelegt. Aus Gründen der Performance ist es sinnvoll, Indices nicht auf Magnetband zu speichern, sondern auf Festplatte oder optischen Medien. Wie die rechte Seite der Abbildung 11 zeigt, können Nutzertabellen in Segmente mit entsprechenden Teilindizes aufgespaltet werden. Tabellen werden automatisch, aufgrund ihrer Größe, in eine bestimmte Anzahl Segmente unterteilt. Durch das Einfügen neuer Daten in die Tabellen, können neue Segmente entstehen. Alle diese Nutzertabellen-, Systemtabellen- und Index-Segmente, werden als separate Dateien in StorHouse/SM gespeichert. Wie auch in kommerziellen DBMS können in StorHouse/RM beliebige Objekte, wie z.B. Bilder oder Videos in LOBs (Large Objects) gespeichert werden. Analog zu Tabellen, werden je nach Größe, eine gewisse Anzahl an LOBs, zu einem Segment zusammengefasst und als Datei gespeichert.
Durch die Möglichkeit einer horizontalen und vertikalen Partitionierung der Daten auf logischer Ebene, kann eine flexiblere Datenverteilung auf unterschiedliche Medien erfolgen. Bei der horizontalen Partitionierung, können verschiedene Zeilen der gleichen Tabelle auf unterschiedliche Speichermedien geschrieben werden. Diese Partitionierung kann eingesetzt wer-

den, um weniger häufig angefragte Einträge einer Tabelle (d.h. historische Daten), auf günstige Speichermedien zu verschieben. Bei der vertikalen Partitionierung werden ein oder mehrere Spalten einer Tabelle auf verschiedene Speichermedien abgelegt. Dadurch können Spalten, auf die selten zugegriffen wird, oder deren Inhalt sehr groß ist (z.B. LOB-Spalte), auf TS-Medien geschrieben werden.
Wird eine auf TS-Medien gespeicherte Nutzertabelle durch neue Einträge erweitert, so werden diese Daten in ein neues Segment geschrieben. Dadurch werden nur die neuen Daten auf TS-Medien exportiert. Aufgrund der Charakteristik von TS-Medien (kein Update z.B. auf Magnetband möglich), werden auch bei Update-Operationen entsprechende Einträge nicht überschrieben. Sondern in einem neuen Segment gespeichert und erneut auf Tertiärspeicher geschrieben. Die veralteten Einträge werden als nicht mehr aktuell markiert. Aus diesem Grund sollte der Speicherort der einzelnen Tabellen mit Bedacht gewählt werden, da auch bei jedem Update das Datenvolumen stetig wächst. Nachträglich eingefügte Einträge in Tabellen und Updates, führen zu fragmentierten Datenbeständen. Durch diese Fragmentierung, erhöht sich bei Zugriffen auf Magnetbänder die Anzahl der kostenintensiven Spuloperationen, da zu den einzelnen Teilfragmenten gesprungen werden muss und kein sequentielles Lesen mehr möglich ist.

Zusammenfassend lässt sich sagen, dass StorHouse/RM entwickelt wurde, um relationale Daten kostengünstig auf unterschiedliche Ebenen der Speicherhierarchie zu speichern. Es ist ein selektiver Zugriff auf diese Daten möglich. Da StorHouse/RM die Möglichkeit bietet, beliebig organisierte Daten als LOBs (bzw. BLOBs) zu speichern, können auch multidimensionale Array-Daten eingefügt und auf unterschiedlichen Ebenen der Speicherhierarchie abgelegt werden. Operationen auf BLOBs, beschränken sich auf das Schreiben bzw. das Lesen kompletter Objekte. Es werden also keinerlei weiterführende Auswertungen dieser Daten auf dem DB-Server unterstützt. Solche Auswertungen müssen in entsprechenden Applikationen durchgeführt werden. Das bedeutet, StorHouse/RM ist in Bezug auf BLOBs nur ein Speicher- und Transaktionsmanager. Durch ein geeignetes Datenbankschema kann zumindest die Suche bestimmter Objekte über Metadaten (Schlagworte, Simulationsname, usw.) vereinfacht werden.
Ein entscheidender Nachteil von StorHouse/RM, in Bezug auf das Datenretrieval multidimensionaler Daten, ist die hohe Granularität des Datentransfers. Hier ist zum einen das Segment (Menge von LOBs) bzw. die von StorHouse/SM gebildete Dateifamilien (Menge von Segmenten) als Zugriffsgranularität zwischen StorHouse/RM und dem TS-System zu nennen. Bei Anfragen, die nur ein Objekt betreffen, müssen alle Objekte eines Segmentes, bzw. einer Dateifamilie, übertragen werden. Das führt zu einer erhöhten Transferzeit und zu einer höheren Netzwerkbelastung. Gerade bei der Datenübertragung zwischen Tertiärspeicher und DBMS könnten durch eine geringere Granularität enorme Kosten eingespart werden. Zum anderen gibt es die Zugriffsgranularität zwischen StorHouse/RM und einer Applikation. Die kleinste Granularität in diesem Bereich, stellen einzelne Objekte (LOBs) dar. Bei Bereichsanfragen auf Teilobjekte kann somit eine Reduktion der Datenmenge erst in der entsprechenden Applikation erfolgen. Die Firma FileTek sieht Data Warehouse Systeme als ein Hauptanwendungsgebiet für ihr Produkt StorHouse/RM. Vor allem historische Daten können hier auf günstige TS-Medien verschoben werden und stehen im Bedarfsfall jederzeit ohne zusätzliche Nutzerinteraktion zur Verfügung. Sollen allerdings große Mengen an multidimensionalen Daten effizient gespeichert werden, so besteht die Notwendigkeit, aufgrund des Datenvolumens, auch operative Daten auf TS-Medien auszulagern. Dies erfordert allerdings ein spezielles Augenmerk auf effizientes Datenretrieval. Da StorHouse/RM allerdings für relationale Daten entwickelt wurde, gibt es keinerlei Optimierung: Wie z.B. die Möglichkeit, Teilbereiche von Objekten zu laden oder das Ausnutzen von Clustering bei multidimensionalen Objekten.

Ein weiterer Nachteil von StorHouse/RM ist die Inkompatibilität zu HSM-Systemen anderer Hersteller. Möchten Firmen, die bereits ein HSM-System einsetzen, eine DBMS-Anbindung mittels StorHouse/RM realisieren, müssen sie ihren gesamten Datenbestand zunächst auf StorHouse/SM migrieren. Bei den riesigen Datenmengen ist diese Umstellung eher als utopisch anzusehen. Eine andere Möglichkeit ist, StorHouse/SM als Zweitsystem zu installieren. Dies kann zu redundanter und inkonsistenter Datenhaltung führen und verursacht zusätzliche Wartungskosten.
Nähere Information über StorHouse/RM sind in [CB00, CB01, CKKB01, CKK00] zu finden. Die Anbindung von IBM/DB2 Universal Database bzw. Microsoft SQL-Server, erfolgt in ähnlicher Weise, wie die hier beschriebene Anbindung des DBMS Oracle [File03b, File03c]. Neben StorHouse/RM sind weitere ähnlich Systeme auf dem Markt erhältlich, wie zum Beispiel DiskXtender Database Edition der Firma Legato. Wie bei StorHouse, gibt es eine Datenbankerweiterung für das bestehende HSM-System DiskXtender [Inmo02, Lega03].

3.4 System CERA

Bereits 1995 erkannte das Deutsche Klimarechenzentrum, ein ESTEDI Projektpartner aus Hamburg, die Notwendigkeit, ihre Datenhaltung auf Magnetbänder neu zu überdenken. Die Datenorganisation als Dateien auf Betriebssystemebene entsprach nicht mehr den Anforderungen an die Zugriffsperformance und die einfache Nutzbarkeit. Um dem enormen Zuwachs an multidimensionalen Daten zu begegnen, wurde beschlossen, ein semantikorientiertes Retrieval der Daten zu entwickeln. Aus diesen Anforderungen ist das System CERA (*Climate and Environmental Data Retrieval and Archiving System*) entwickelt worden. Dabei wurde ein Datenbankschema für die Metadaten der Klimasimulationen modelliert (Abbildung 12). Diese Metadaten enthalten technische Information über das Experiment und Information speziell für die Suche, wie z.B. Schlagworte. Dies ersetzt den dateiorientierten Zugriff durch ein Datenretrieval, das entsprechend der Semantik der gespeicherten Klimadaten formuliert werden kann. Neben den Metadaten werden nach einer Weiterentwicklung auch aufbereitete Simulationsdaten direkt als BLOBs in Tabellen der Oracle Datenbank gespeichert, um einen schnellen Zugriff auf aktuelle Daten zu erhalten.

Aufgrund der begrenzten Datenbankkapazität, wurde eine Anbindung an das HSM-System DiskXtender der Firma Legato entwickelt. Diese Kopplung entspricht der mittleren Variante der Abbildung 6. Grundlage hierfür bildet die physikalische Datenorganisation des Oracle DBMS. In Oracle werden Tabellen und BLOBs in Tablespaces organisiert. Ist das Oracle DBMS auf einem Unix Dateisystem aufgesetzt (kein Raw-Device), wird eine Tablespace als eine, bzw. mehrere Dateien gespeichert. Um Daten auf Tertiärspeicher auszulagern, wird die Möglichkeit von Oracle genutzt, Tablespaces Online bzw. Offline schalten zu können. Online entspricht dem normalen Status. Daten innerhalb dieser Tablespace können direkt angefragt werden. Eine Tablespace, die Offline gesetzt wurde ist deaktiviert und Anfragen können von Oracle nicht bearbeitet werden. Diese Offline gesetzten Tablespaces können jedoch durch ein darunter liegendes HSM-System auf Magnetbänder ausgelagert werden. Damit Oracle eine Anfrage auf diese ausgelagerten Daten ausführen kann, wurde ein Storage Broker entwickelt, der diese Tablespace vom Tertiärspeicher lädt und wieder Online schaltet. Danach kann die Anfrage wie gewohnt ausgeführt werden. Ein solcher Storage Broker wurde im CERA-System integriert. Weitere Details über das CERA-System sind in [Laut95, LT01, Laut02] nachzulesen.

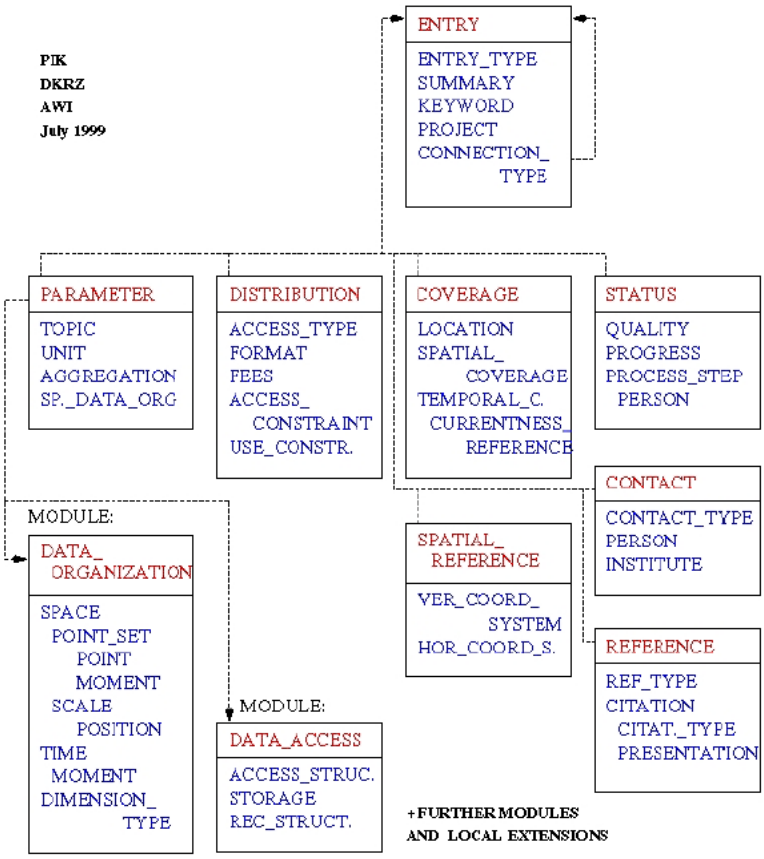


Abbildung 12: Diagramm des CERA Datenmodells

Das CERA-System bietet durch die enthaltenen Metadaten, eine für den Wissenschaftler einfachere Möglichkeit des Datenretrievals. Ein gravierender Nachteil des CERA-Systems ist die hohe Granularität des Datentransfers zwischen DBMS und Tertiärspeichersystem, da nur komplette Tablespaces auf TS-Medien geschrieben, bzw. bei Anfragen zurückgeholt werden können. Im Normalfall werden mehrere Objekte einer Simulationsreihe in eine Tablespace geschrieben. Bei einer Bereichsanfrage auf eines dieser Objekte, muss zuerst die gesamte Tablespace vom Tertiärspeicher geladen werden. Durch das Laden zusätzlicher, nicht benötigter Daten (Objekte) werden die Transferzeit und die Netzwerkbelastung unnötig in die Höhe getrieben. Als weiterer Nachteil ist zu bewerten, dass es nicht möglich ist, auf Teilbereiche von Objekten zuzugreifen, da das CERA-System nur als Datenlieferant komplette Objekte für weitere Anwendungen realisiert. Will ein Anwender in einer Applikation eine Anfrage formulieren, die einen Stich durch mehrere dieser Objekte bedeutet, so müssen vom CERA-System alle betroffenen Objekte (als BLOBs gespeichert) an die Anwendung übertragen werden, nachdem die Tablespace von TS-Medien geholt wurde. Weiterhin werden beim Speichern der Daten auf TS-Medien keine Mechanismen zur Optimierung der Datenzugriffe integriert. So werden die einzelnen BLOBs einer Tablespace einfach als linearisierter Bytestrom auf das TS-Medium geschrieben, ohne z.B. das Clustering multidimensionaler Daten auszunutzen. Das Update von Objekten ist beim CERA-System nicht vorgesehen. Dies würde bedeuten, dass aufgrund der Charakteristik von TS-Medien eine gesamte Tablespace neu auf TS-Medium gespeichert werden muss, obwohl nur ein Objekt geändert wurde. Auch das Fehlen einer multidimensionalen Anfragesprache innerhalb des CERA-Systems, war ein Kriterium für das Deutsche Klimarechenzentrum, sich an dem Projekt ESTEDI zu beteiligen, um ein verbessertes Gesamtsystem zu erhalten.

3.5 Postgres

Ein weiters System, das betrachtet wird, ist das DBMS Postgres, das in den 80er Jahren an der Universität Berkeley entwickelt wurde. 1996 wurde dieses DBMS in PostgreSQL umbenannt und stetig durch die Open Source Initiative weiterentwickelt. Die aktuelle Version PostgreSQL 7.4 wurde im November 2003 freigegeben. Da der Quellcode von Postgres bereits sehr früh offen gelegt wurde, bot sich dieses System an, um eigene Modifikationen und Erweiterungen zu implementieren. In seiner Master Thesis realisierte Michael Olson bereits 1992 eine direkte Kopplung von Tertiärspeichersystemen an Postgres in der Version 4.0. Diese Arbeit schaffte es allerdings nicht, in den offiziellen Quellcode von Postgres bzw. PostgreSQL integriert zu werden. Allerdings bietet die Erweiterung von Olson einige interessante Ansätze und Konzepte, die hier näher betrachtet werden [Olso92].

Um die Anbindung an tertiäre Speichermedien zu realisieren, wurde Postgres um einen neuen Speichermanager (Storage Manager) und Gerätemanager (Device Manager) für eine Sony CD-ROM Jukebox und eine Exabyte Magnetband Jukebox erweitert (Abbildung 13). Um Daten zu lesen oder zu schreiben, kommuniziert der Datenmanager (Data Manager) in dieser Architektur nicht mehr direkt mit den Speichersystemen. Als weitere Schicht wurde der Speichermanager eingeführt, mit der die Speicherhierarchie verwaltet wird. Der Speichermanager hat die Information, welche Speichermedien für die Beantwortung einer Anfrage benötigt werden. Die unterschiedlichen Speichersysteme bzw. -geräte werden über eigene Gerätemanager an das System angebunden. Somit ist es möglich, durch die Implementierung eines neuen Gerätemanagers, neue Tertiärspeichergeräte anzubinden. Das zeigt allerdings gleich einen Nachteil dieser Realisierung, da ein hoher Implementierungsaufwand notwendig ist, um neue Speichergeräte anzubinden.

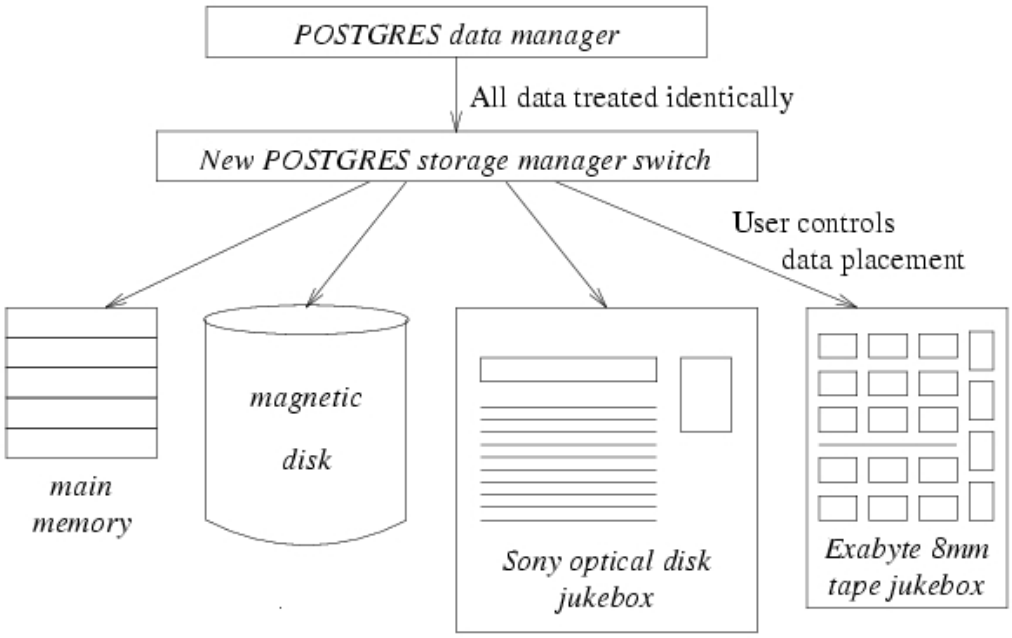


Abbildung 13: Architektur mit erweitertem Speichermanager [Olso92]

Beim Einfügen neuer Datensätze in Postgres, kann angegeben werden, welches Speichermedium herangezogen werden soll. Dabei wird einfach der Name des entsprechenden Gerätemanagers angegeben:

MitarbeiterName (ID = int, Name = text) store = „magnetic disk“
MitarbeiterFoto (ID = int, Foto = image) store = „sony optical disk jukebox“

In unserem Beispiel wird die ursprüngliche Datenbanktabelle Mitarbeiter (ID, Name, Foto) in zwei physikalisch getrennte Partitionen aufgeteilt. Für jede dieser Partitionen kann ein beliebiger Speicherort gewählt werden. Partitionen können allerdings nur komplette Spalten von Tabellen enthalten. Der entsprechende Gerätemanager allokiert für die jeweiligen Partitionen Speicher in Einheiten von so genannten Extents. Welche somit die kleinste Zugriffsgranularität für Daten sind. Für den Gerätetreiber der Sony Jukebox wurde eine Extentgröße von 256 KByte gewählt, was dem 32fachem einer Datenbankseite (8 KByte) entspricht. Durch die Partitionierung besteht die Möglichkeit, Spalten mit hohem Speicheraufwand, wie zum Beispiel bei Bildern, auf günstige Speichermedien auszulagern. Welche Partitionen sich auf welchem Datenspeicher befinden, wird in einer Systemtabelle verwaltet. Dadurch wird bei Anfragen automatisch der richtige Gerätemanager ausgewählt und die entsprechenden Daten geladen.

Werden bei einer Anfrage Daten von tertiären Speichermedien angefordert, so fungiert die Festplatte als Cache. Dieser Cache ist persistent realisiert, was bedeutet, dass sein Inhalt auch bei einem Neustart von Postgres erhalten bleibt. Dadurch werden Kosten reduziert, da die Wahrscheinlichkeit hoch ist, dass Anfragen mit dem Inhalt des Caches beantwortet werden können, ohne Daten neu von tertiären Speichermedien zu laden. In Postgres müssen neue Einträge und geänderte Einträge nicht sofort auf die jeweiligen Speichermedien propagiert werden. Dies vermeidet zwar einerseits unnötige Transferkosten, kann allerdings die Atomarität der einzelnen Schreibvorgänge verletzen [Olso92].

1993 wurde die Möglichkeit der Speicherung von LOBs (*Large Objects*) in Postgres vorgestellt [SO93]. Diese Entwicklung ebnete den Weg, um multidimensionale Arrays in Postgres zu speichern. Vor allem Sunita Sarawagi nutzte den erweiterten Speichermanager von Postgres und entwickelte einige Konzepte um multidimensionale Array-Daten effizient zu speichern. Zu diesen Strategien zählen Chunking, Scheduling und Caching. Beim Chunking wird ein Objekt in mehrere gleichförmige Teilobjekte zerlegt und als einzelne LOB in der Datenbank gespeichert. Abbildung 14 stellt beispielhaft das Chunking eines zwei und drei dimensionalen Objektes dar. Dabei wird klar, dass durch Aufteilung der großen Objekte in mehrere LOBs, bei Anfragen (graue Queryboxen) sehr viel weniger Daten geladen werden müssen. Dies bedeutet allerdings auf der anderen Seite einen Mehraufwand bei der Verwaltung dieser Objekte. Es muss klar sein, dass ein vom Anwender gewähltes Chunking nicht für alle Anfragenmuster gleich gut geeignet ist. Somit muss das Chunking mit bedacht gewählt werden, da bei falsch gewähltem Chunking eine Verschlechterung der Zugriffsperformance eintreten kann.

Mit Hilfe eines Schedulers wird versucht die Ausführungsreihenfolge von Anfragen zu optimieren. Dabei wird durch Information über den Cacheinhalt und den benötigten Daten entschieden, welches LOB als nächstes geladen wird, bzw. welche Objekte aus dem Cache verdrängt werden. Nähere Informationen über die Arbeiten von Sarawagi sind in [SS 94, Sara95a, Sara95b] zu finden.

Vergleichen wir die hier realisierte Anbindung von Postgres an tertiäre Speichermedien, so entspricht dies der rechten Darstellung in Abbildung 6. Postgres ist über die unterschiedlichen Gerätemanager direkt an TS-Systeme gekoppelt. Dies erfordert einen hohen Implementierungsaufwand, um neue Speichergeräte an Postgres anzubinden, da für jedes Gerät ein separa-

ter Gerätemanager geschrieben werden muss. Die Arbeiten von Sarawagi ermöglichen es, multidimensionale Daten in einer aktiven Speicherhierarchie zu verwalten. Einen enormen Vorteil bildet das Konzept des Chunking von Objekten. Allerdings ist Chunking nicht für alle Zugriffsmuster optimal geeignet, da es nur gleichförmige Teilobjekte zulässt. Das allgemeinere Tiling-Konzept würde mehr Möglichkeiten bieten.

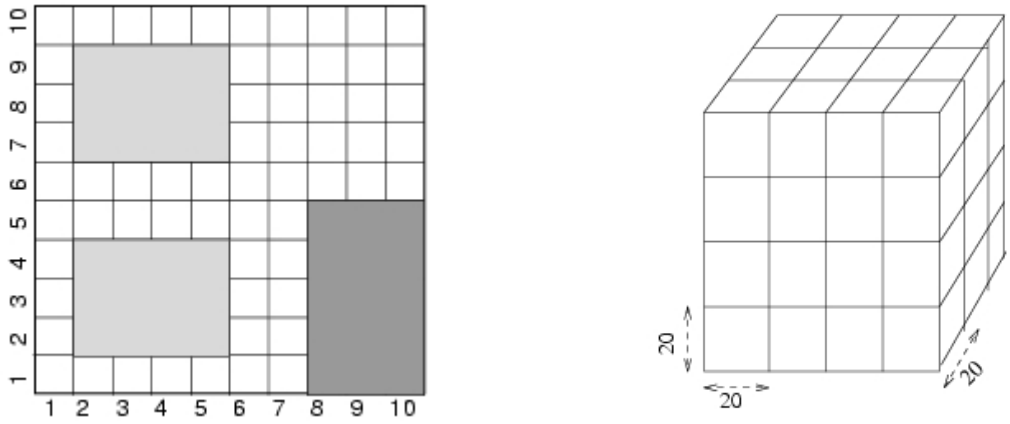


Abbildung 14: Mögliches Chunking eines zwei- bzw. drei dimensionalen Arrays

Beim Speichern der Chunks (LOBs) auf tertiären Speichermedien, wird zwar eine „optimale" Ordnung gewählt: Allerdings wird immer entlang einer Array-Achse geordnet und somit findet kein vernünftiges multidimensionales Clustering statt. Ein weiterer vernachlässigter Punkt ist die Zugriffsgranularität auf die multidimensionalen Daten. Es wird keine unterschiedliche Granularität für die Speicherung auf Tertiärspeicher und im DBMS realisiert. Wird eine große Granularität gewählt, um besseres Zugriffsverhalten für Tertiärspeicher zu bekommen, ist diese nicht optimal für die interne Struktur von Postgres. Weiterhin wurde keine multidimensionale Anfragesprache integriert, um multidimensionale Daten direkt im DBMS zu bearbeiten. Es können nur Teilbereiche eines Objektes aus der Datenbank geholt werden. Die Verarbeitung dieser Daten muss auf einer höheren Ebene geschehen.

3.6 HEAVEN

Als weiteres System wird HEAVEN betrachtet und bewertet. HEAVEN realisiert eine ***h**ierarchische Sp**e**icher- und **A**rchi**v**ierungsumgebung für multidim**e**nsionale Array Date**n**bank Management Systeme*. Grundlegende Arbeiten wurden im Rahmen des europäischen Projektes ESTEDI (*European Spatio-Temporal Data Infrastructure for High-Performance Computing*) durchgeführt. Erklärtes Ziel von HEVEN ist es, für Hochleistungsrechenzentren (*High-Performance Computing*, HPC), eine flexible und performante Infrastruktur für sehr große Datenvolumen zu entwickeln und zu etablieren. Dabei soll vor allem der bestehende Flaschenhals im Bereich des Retrieval und der Evaluierung dieser Rasterdaten beseitigt werden. Erreicht wurde dies, durch Einsatz eines multidimensionalen *Datenbank Management Systems* (DBMS) mit direkter, automatisierter Anbindung an TS-Systeme für die Verwaltung der riesigen Datenmengen. Es wurde dabei nicht nur konzeptuelle Arbeit geleistet, sondern unter konsequenter Einbeziehung der Nutzer auf die Praxistauglichkeit der entwickelten Systeme geachtet. Dabei sind auch die in Kaptitel 1 aufgeführten Defizite bewusst behandelt und Lösungskonzepte erarbeitet worden. Für den Nutzer dieser Infrastruktur, bedeutet das eine Verbesserung der Servicegüte in Bezug auf Performance, Funktionalität und Verwendbarkeit.

HEAVEN kombiniert die Vorteile der kostengünstigen Speicherung riesiger Datenmengen auf Tertiärspeicher und ein effizientes Retrieval und Management multidimensionaler Array-Daten durch ein DBMS. Als Datenbank wird das erste kommerzielle DBMS RasDaMan (*Raster Data Management*) verwendet, das speziell für multidimensionale Array-Daten entwickelt wurde. Das Array DBMS RasDaMan wird von der Firm RasDaMan GmbH vertrieben. RasDaMan wurde von uns erweitert, um auf TS-Medien gespeicherte Daten effizient verwalten zu können. Es entstand eine hierarchische Speicher- und Archivierungsumgebung für multidimensionale Array Datenbank Management Systeme (HEAVEN). Abbildung 15 zeigt die Architektur von HEAVEN.

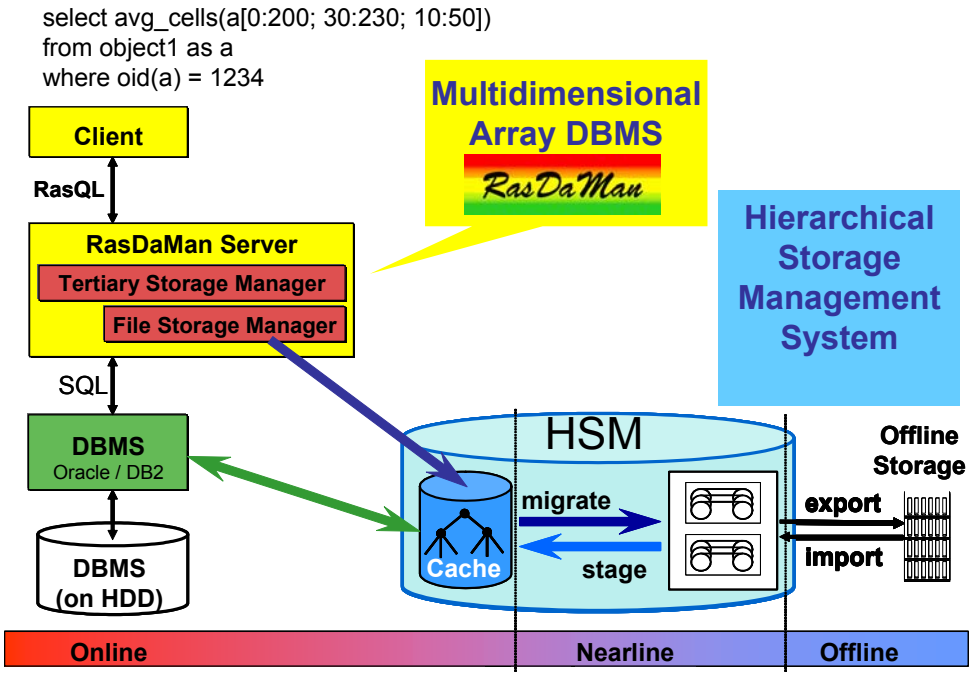
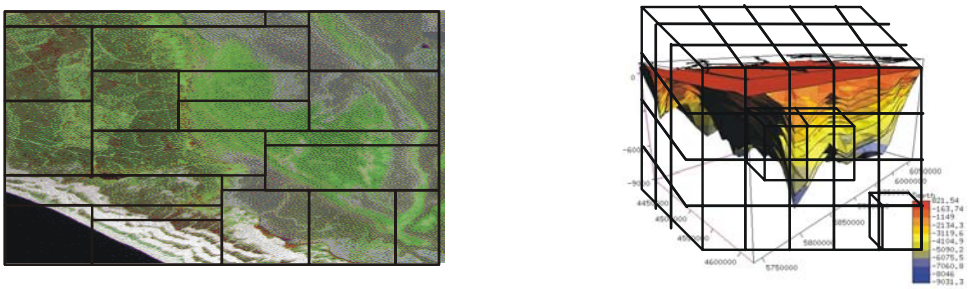


Abbildung 15: Architektur des Systems HEAVEN

Die linke Seite der Abbildung zeigt die ursprüngliche Client/Server Architektur von RasDaMan mit dem darunter liegendem konventionellen DBMS (z.B. Oracle, das als Speicher- und Transaktionsmanager verwendet wird). Wir realisierten die Unterstützung tertiäre Speichermedien durch eine Anbindung eines HSM-Systems, wie zum Beispiel SAM (Storage Archiving System) der Firma LSC oder DiskXtender der Firma Legato Systems. Die Funktionsweise eines HSM-Systems ist in Kapitel 2.2 erläutert. Dabei wurde bewusst bei der Entwicklung der Schnittstelle zur Anbindung von HSM-Systemen darauf geachtet, möglichst viele Systeme zu unterstützen. In HEAVEN wurde zusätzlich eine Schnittstelle integriert, um direkt Tertiärspeichersysteme anzubinden. Durch diese Möglichkeit sind weit reichende Optimierungsmöglichkeiten gegeben. Verglichen mit Abbildung 6, unterstütz HEAVEN die Kopplung von TS-Systemen über ein Dateisystem (mittlere Darstellung) und ein eine direkte Anbindung (rechte Darstellung).

Große Datenojekte können in Teilobjekte zerlegt und gespeichert werden. Dabei wird neben Chunking auch das Tiling-Konzept unterstützt. Tiling ist allgemeiner und erlaubt nicht nur gleichförmige Teilarrays. Abbildung 16 zeigt unterschiedliche Möglichkeiten des Tiling.

Dieses Tiling kann sehr flexibel auf typische Zugriffsmuster von Anwendern ausgelegt werden.



Arbitrary Tiling (2D) Regular Tiling (3D)

Abbildung 16: Unterschiedliches Tiling

Die Größe der Tiles ist in RasDaMan für den Zugriff auf Festplatte optimiert (32 KByte bis 1 MByte). Um einen effizienten Zugriff auf TS-Medien zu realisieren, ist die Größe eines Tiles zu klein. Somit wurde das Super-Tile, eine zusätzliche Granularität optimiert für TS-Zugriffe, eingeführt. Die grundsätzliche Idee dieses Konzeptes, ist es verschiedene Tiles auf eine intelligente Weise zu einem Super-Tile zu kombinieren, um Zugriffskosten für TS-Medien zu minimieren. Dabei wird die räumliche Nähe der Tiles innerhalb eines Objektes berücksichtigt. Neben diesem Clustering innerhalb eines Super-Tiles, erfolgt beim Exportieren auf ein TS-Medium auch ein Clustering zwischen Super-Tiles. Dadurch wird inter Super-Tile und intra Super-Tile Clustering realisiert, was Zugriffszeiten auf TS-Medien erheblich reduziert (Abbildung 17).

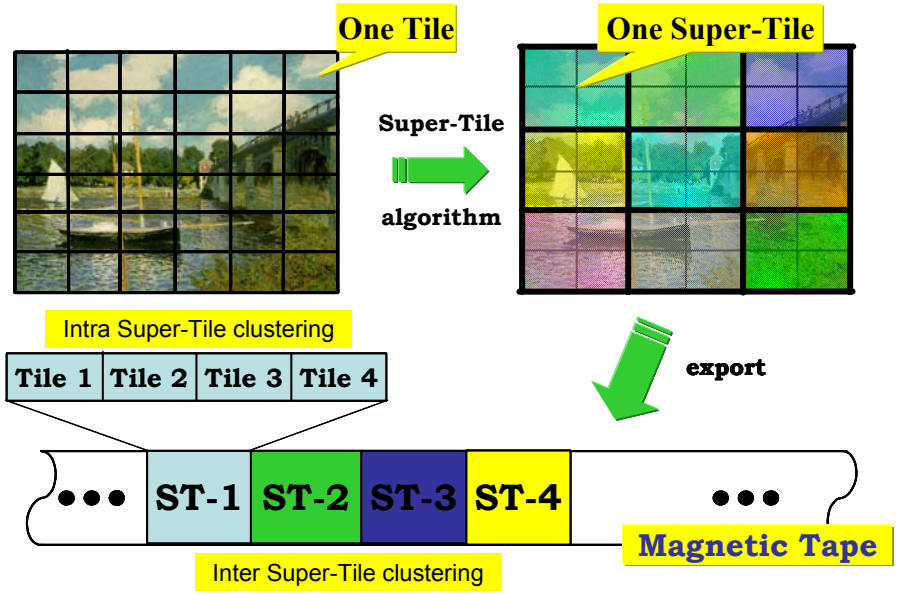


Abbildung 17: Inter und intra Super-Tile Clustering

Aspekte der Performance spielen naturgemäß eine wesentliche Rolle im Bereich des Retrievals riesiger Datenmengen. Deshalb sollen nicht nur eine einfache Anbindung eines TS-Systems an RasDaMan realisiert, sondern spezielle Optimierungskriterien bei Datenzugriffe berücksichtigt und entwickelt werden. Schlagwörter hierzu sind Clustering, Query Scheduling, Caching und I/O-Parallelität. Eine Scheduling-Komponente untersucht alle eingehenden

Anfragen und optimiert die Reihenfolge dahingehend, teure Medienwechsel und Suchoperationen zu reduzieren. Es wurden Caching Strategien entwickelt, die zum Beispiel bei der Entscheidung zur Verdrängung einzelner Objekte, zum Beispiel auch die Größe einzelner Super-Tiles mit berücksichtigen. Dies vergrößert die Trefferrate, ein Objekt im Cache vorzufinden. Durch Ausnutzung und Anpassung der von RasDaMan angebotenen Inter- und Intra-Operator Parallelität wird ein beschleunigter Datenzugriff auf Tertiärspeicher ermöglicht (I/O-Parallelität). HEAVEN bietet außerdem eine multidimensionale Anfragesprache, mit der multidimensionale Operationen auf Daten direkt im Datenbankserver verarbeitet werden können. Somit entfällt die Notwendigkeit, die angefragten Daten zu Applikationen zu übertragen. Eine Beschreibung der Anfragesprache RasQL (*RasDaMan Query Language*) ist in [Baum99] zu finden. HEAVEN verfügt weiterhin über eine Transferkompression. Somit können Netzwerke durch verringertes Datenvolumen entlastet werden. Weiterführende Details zu den hier vorgestellten Konzepten über das System HEAVEN sind in [Rein01, HR02, RH02, HRHB02, RHHB02, HRH03, RH03] aufgeführt.

Zusammenfassend lässt sich sagen, dass HEAVEN eine flexible Möglichkeit bietet, TS-Systeme anzubinden. Entsprechend Abbildung 6 kann eine Kopplung entweder direkt oder über ein Dateisystem erfolgen. Große multidimensionale Objekte lassen sich durch das Tiling-Konzept, entsprechend typischer Anfragemuster in Teilobjekte zerlegen. Dadurch ist es auch möglich, Teilbereiche anzufragen. HEAVEN bietet Nutzern und Applikationen eine logische Sicht auf die gespeicherten Daten, unabhängig vom tatsächlichen physikalischen Speicherort. Weiterhin wird eine transparente Sicht auf Daten realisiert. Durch die Verwendung der multidimensionalen Anfragesprache von RasDaMan wird das Datenretrieval vereinfacht. Komplexe Anfragen werden direkt auf dem Datenbank-Server erledigt. Die bisher notwendigen Werkzeuge, bzw. Arbeitschritte eines Anwenders, werden reduziert. Wissenschaftler müssen nicht mehr mit Dateien, Verzeichnissen, Dateinamen und Dateiformaten umgehen. Sie können Daten mit adäquatem semantischem Level anfragen. Zur Optimierung werden Methoden wie Clustering, Scheduling, Caching, I/O-Parallelität und Transferkompression eingesetzt. Die Zugriffsgranularität ist den entsprechenden Anforderungen (Festplatte und TS-Medien) durch das Tiling und den Super-Tiles angepasst. HEAVEN verfügt durch intelligente Automatisierung der Datenauslagerung, Datenspeicherung und des Datenretrievals über ein hochgradig skalierbares Datenmanagement. Somit werden mit HEAVEN fast unlimitierte Datenbankgrößen erreicht.

3.7 Vergleich der Systeme

Nach erfolgter Beschreibung und Bewertung der untersuchten Systeme werden wichtige Eigenschaften hinsichtlich einer effizienten Verwaltung von multidimensionalen Array-Daten vorgestellt. Dieser Wunschkatalog wurden aus den in Kaptitel 1 aufgeführten Problemen beim Retrieval und bei der Speicherung großer Datenvolumen entwickelt. Es werden folgende Kriterien zur Bewertung der Systeme herangezogen:

- Können **Metadaten** durch ein DBMS effizient gespeichert und verwaltet werden und wird somit ein schnelles Auffinden der angefragten Daten ermöglicht?
- Werden multidimensionale Daten **(MDD) im DBMS** gespeichert und auf TS-Medien ausgelagert, um die Vorteile der Datenhaltung durch ein DBMS auszunutzen?
- Besteht die Möglichkeit, große Objekte zu unterteilen in gleichförmige Teilbereiche (**Chunking**)? Dies bildet die Voraussetzung für effiziente Bearbeitung von Bereichsanfragen.

- Ist es möglich, diese Unterteilung (Chunking) in einer allgemeineren Form zu realisieren (**Tiling**), um besser den Zugriffsmustern der Anwender zu entsprechen?
- Können Anwender Teilbereiche von Objekte Anfragen (**Subset retrieval**) und werden auch nur diese Teilobjekte übertragen?
- Verfügt das System über eine spezielle multidimensionale Anfragesprache (**MQL**), um Anfragen im DBMS zu bearbeiten oder wird das DBMS nur als Speicher- und Transaktionsmanager verwendet?
- Werden die Daten entsprechend einer **Clustering**-Strategie auf den TS-Medien abgelegt, um Zugriffskosten zu minimieren?
- Findet ein **Caching** der Daten mit intelligenter Verdrängungsstrategie statt?
- Kann die zu übertragene Datenmenge durch eine **Transfer-Kompression** (zwischen TS-System und DBMS-Server bzw. zwischen DBMS-Server und Client) verringert werden?
- Ist es möglich, mehrere Datensätze parallel von TS-Medien anzufordern und zu übertragen (**Parallel I/O**)?
- Wie gut ist das untersuchte System hinsichtlich des zu speichernden Datenvolumens und der Anwender **skalierbar**?
- Ist die **Zugriffsgranularität** entsprechend den Besonderheiten der verwendeten Speichersysteme, wie Festplatte bzw. TS-System, angepasst?

In Tabelle 3 werden die untersuchten Systeme hinsichtlich der genannten Kriterien bewertet und gegenübergestellt. Dabei steht das Symbol ✓ für die Unterstützung der jeweiligen Eigenschaften. Das Symbol ~ bedeutet, die Eigenschaften werden teilweise oder nur rudimentär erfüllt.

	Metadaten	MDD in DBMS	Chunking	Tiling	Subset retrieval	MQL	Clustering	Caching	Query scheduling	Transfer Kompression	Parallel I/O	Skalierbarkeit	Zugriffs-granularität
SDM	✓										✓	~	
MDMS & APRIL	✓		✓		✓		~	~	~		✓	~	
StorHouse/RM	✓	✓							~				
CERA	✓	✓											
Postgres	✓	✓	✓		✓		~	✓	✓				
HEAVEN	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Tabelle 3: Vergleich der untersuchten Systeme

Die Systeme SDM, MDMS und APRIL verwenden ein relationales DBMS nur zur Verwaltung der Metadaten. Die einzelnen Daten werden nach wie vor als Dateien auf die TS-Medien gespeichert. Nur das Auffinden der Daten wird vereinfacht. Es werden keine bekannten Vorteile der Datenverwaltung durch DBMS wie das Transaktionskonzept oder Zugriffskontrolle usw. unterstützt. Bei den untersuchten Systemen StorHouse/RM, CERA und Postgres wurden normale relationale DBMS verwendet, um die Anbindung an ein Tertiärspeichersystem zu realisieren. Diese nutzen zwar die Vorteile der Datenhaltung durch DBMS, allerdings wurden die verwendeten DBMS ursprünglich nicht für die Verwaltung von multidimensionalen Array-Daten entwickelt. Das System HEAVEN geht einen anderen Weg und fusioniert das speziell für multidimensionale Array-Daten entwickelte DBMS RasDaMan mit Tertiärspeichersystemen. Dadurch können Optimierungen speziell für multidimensionale Array-Daten mit eingebracht werden.

4 Zusammenfassung

Diese Arbeit zeigt den Stand der Forschung und Entwicklung im Bereich der Fusion von Datenbanktechnologie und Tertiärspeichersystem für sehr große multidimensionale Array-Daten. Zunächst wurden bestehende Probleme in der Speicherung und im Retrieval dieser Daten aufgezeigt und Lösungskonzepte vorgeschlagen. Danach wurden die Vorteile der Datenverwaltung durch DBMS vorgestellt. Zur Zeit unterstützt keines der auf dem Markt führenden DBMS wie Oracle, IBM/DB2 Universal Database und Microsoft SQL-Server, eine direkte Anbindung an Tertiärspeicher (ausgenommen Backup). Um einen Überblick über den Stand der Forschung und Entwicklung im Bereich der Anbindung von Tertiärspeicher an DBMS zu erhalten, wurden folgende Entwicklungen beschrieben und bewertet:

- SDM (***S**cientific **D**ata **M**anager*)
- MDMS & APRIL (**M**eta-***d**ata **M**anagement **S**ystem & **A** **P**arallel **R**un-T**i**me **L**ibrary for Tape-Resident Data*)
- StorHouse/RM (*StorHouse **R**elational **M**anager*)
- CERA (***C**limate and **E**nvironmental Data **R**etrieval and **A**rchiving System*)
- HEAVEN (***H**ierarchische Sp**e**icher- und **A**rchi**v**ierungsumgebung für multidim**e**nsionale Array Date**n**bank Management Systeme*)

Dabei verwenden die ersten beiden Systeme ein DBMS nur zur Verwaltung der Metadaten. Die anderen Entwicklungen in diesem Bereich, erlauben die direkte Speicherung der Daten innerhalb einer DBMS und ermöglichen eine automatische Speicherung auf TS-Medien. Es wird somit eine aktive Speicherhierarchie über alle Speicherebenen realisiert. Dabei wurden die Systeme entsprechend ihrer Eignung für optimierte Speicherung und effizientes Datenretrieval für multidimensionale Array-Daten bewertet. Als besonders geeignet stellt sich das System HEAVEN heraus, da umfangreiche und spezielle Optimierungsmethoden umgesetzt wurden. Für den Nutzer dieser Infrastruktur, bedeutet das eine Verbesserung der Servicegüte in Bezug auf Performance, Funktionalität und Verwendbarkeit.

Literaturverzeichnis

Baum99 Baumann P.*: A Database Array Algebra for Spatio-Temporal Data and Beyond*; Proceedings of the 4th International Workshop on Next Generation Information Technologies and Systems (NGITS), 1999

CB00 Carino F., Burgess J.: *StorHouse/Relational Manager (RM) – Active Storage Hierarchy Database System and Applications*; Proc. of the 8th NASA Goddard Conference on Mass Storage Systems and Technologies, S. 179-186, Maryland, USA, März 2000

CB01 Carino F. Jr., Burgess J.: *StorHouse.com – The Data Management Service Provider*; Proc. of the 18th IEEE Symposium on Mass Storage Systems, San Diego, USA, April 2001

CKK00 Carino F. Jr., Kaufmann A., Kostamaa P.: *Are you ready for Yottabytes? StorHouse – Federated and Object/Relational Solution*; Proc. of the 8th NASA Goddard Conference on Mass Storage Systems and Technologies, Maryland, USA, März 2000

CKKB01 Carino F. Jr., Kostamaa P., Kaufmann A., Burgess J.: *StorHouse Metanoia – New Applications for Database, Storage & Data Warehousing*; ACM SIGMOD, Santa Barbara, California, USA, May 2001

CKNM99 Choudhary A., Kandemir M., No J., Memik G., Shen X., Liao W., Nagesh H., More S., Taylor V., Thakur R.: *Data Management for Large-Scale Scientific Computations in High Performance Distributed Systems*; 8th IEEE International Symposium on High Performance Distributed Computing (HPDC), Redondo Beach, California, USA, August 1999

File03a FileTek Incorporation: *Transparent Oracle Access to StorHouse – Exploit Your Data Assets While Protecting Your Oracle Investment*; FileTek Incorporation product sheet, 2003
http://www.filetek.com/software/product_sheets/oracle/oracle_ds.pdf

File03b FileTek Incorporation: *StorHouse/UDB Link – Extending IBM DB2 Universal Database (UDB) Capabilities with StorHouse/UDB Link*; FileTek Incorporation product sheet, 2003
http://www.filetek.com/software/product_sheets/datasheet_udb.htm

File03c FileTek Incorporation: *Transparent SQL Server 7.0 Access to StorHouse – Extend Your Database Capabilities While Protecting Your SQL Server 7.0 Investment*; FileTek Incorporation product sheet, 2003
http://www.filetek.com/software/product_sheets/sql7.pdf

GKDT02 Gheller C., Kleese van Dam K., Diedrich E., Toussaint F., Samsonova M. Robin E.: *HPC with and without innovations of the ESTEDI project*; ESTEDI project review meeting, Munich, May 2002

HPSS03 IBM Corporation: *HPSS – Basics of the High Performance Storage System*; IBM Global Services, Texas, USA, 2003
http://www4.clearlake.ibm.com/hpss/about/HPSS-Basics.pdf

HR02 Hahn K., Reiner B.: *Intra-Query Parallelism for Multidimensional Array Data*; 28th International Conference on Very Large Data Bases (VLDB), Hong Kong, China, 2002

HRH03 Hahn K., Reiner B., Höfling G.: *Parallel Query Support for Multidimensional Data: Intra-object Parallelism*; 14th International Conference on Database and Expert Systems Applications (DEXA), Prague, Czech Republic, 2003

HRHB02 Hahn K., Reiner B., Höfling G., Baumann P.: *Parallel Query Support for Multidimensional Data: Inter-object Parallelism*; 13th International Conference on Database and Expert Systems Applications (DEXA), Aix en Provence, France, 2002

HS03 Harms U., Stiller A.: *Datenfluten und Sackgassen*; c't Magazin für Computertechnik, Heise Zeitschriften Verlag, S. 54, Heft 15, Hannover, Deutschland, 2003

Inmo02 Inmon B.: *The Infinite Data Warehouse – Cost-Efficient Extending Stroage Capacity with Near-Line Technologies*; White paper, Legato Systems Inc., USA, 2002

Laut02 Lautenschlager M.: *Daten-Infrastruktur bei M&D und DKRZ*; Workshop on Definition of a Community Clime Data Archive at DKRZ, Hamburg, März 2002

Laut95 Lautenschlager M.: *Data Handling in the Climate Model Archive at DKRZ*; Proceedings of the 9th International Symposium on Computer Science for Environmental Protection - Space and Time in Environmental Information Systems, S. 287 – 294, Marburg 1995

Lega03 Legato Systems Incorporation: *DiskXtender for Unix – System Administrator Guide Release 2.5*; Legato System Incorporation, California, USA, Februar 2003

LT01 Lautenschlager M., Thiemann H.: *Semantic oriented data access and storage at MPIM/DKRZ*; 18th IEEE Symposium on Mass Storage Systems, S. 79 – 84, San Diego, USA, April 2001

Memi00 Memik G.: I/O *Optimization for Hierarchical Storage Systems*; Technical Report No. CPDC-TR-2000-05-004, Center for Parallel and Distributed Computing, Northwestern University, Evanston, Illinois, USA, May 2000

MKCT00 Memik G., Kandemir M. T., Choudhary A., Taylor V. E.: *April: A Run-Time Library for Tape-Resident Data*; 17th IEEE Symposium on Mass Storage Systems, College Park, Maryland, USA, March 2000

NTC00 No J., Thakur R., Choudhary A.: *Integrating Parallel File I/O and Database Support for High-Performance Scientific Data Management*; Proceedings of SC2000 High Performance Networking and Computing, Dallas, USA, November 2000

Olso92 Olson M. A.: *Extending the POSTGRES Database System to Manage Tertiary Storage*; Mater Thesis, University of California, Berkeley, USA, 1992

Rein01 Reiner B.: *Tertiary Storage Support for Multidimensional Data*; Workshop Supercomputing Databases at the 27th International Conference on Very Large Data Bases (VLDB), Roma, Italy, September 2001

RH02 Reiner B., Hahn K.: *Tertiary Storage Support for Large-Scale Multidimensional Array Database Management Systems*; 28th International Conference on Very Large Data Bases (VLDB), Hong Kong, China, August 2002

RH03 Reiner B., Hahn K.: *Smart Hierachical Storage Support for Large-Scale Multidimensional Array Database Management Systems*; FORWISS technical report no. FR-2003-001, Munich, Germany, February 2003

RHHB02 Reiner B., Hahn K., Höfling G., Baumann P.: *Hierarchical Storage Support and Management for Large-Scale Multidimensional Array Database Management Systems*; 13th International Conference on Database and Expert Systems Applications (DEXA), Aix-en-Provence, France, September 2002

Sara95a Sarawagi S.: *Query Processing in Tertiary Memory Databases*; Proceedings of the 21st VLDB Conference, Zurich, Swizerland, September 1995

Sara95b Sarawagi S.: *Database Systems for Efficient Access to Tertiary Memory*; 14th IEEE Mass Storage Systems Symposium, Monterey, California, USA, 1995

SLCM00 Shen X., Liao W., Choudhary A., Memik G., Kandemir M., More S., Thiruvathukal G., Singh A.: *A Novel Application Development Environment for Large-Scale Scientific Computations*; International Conference on Supercomputing (ICS), Santa Fe, New Mexico, USA, 2000

SO93 Stonebraker M., Olson M.: *Large Object Support in POSTGRES*; Proceedings of the 9th International Conference on Data Engineering, Vienna, Austria, April 1993

SS94 Sarawagi S., *Stonebraker M.: Efficient Organisation of Large Multidimensional Arrays*; 10th International Conference on Data Engineering, Houston, Texas, USA, 1994

TGL99 Thakur R., Gropp W., Lusk E.: *On Implementing MPI-IO Portably and with High Performance*; Proceedings of the 6th Workshop on I/O in Parallel and Distributed Systems, May 1999