Datenbank Spektrum (2010) 10: 143–158
DOI 10.1007/s13222-010-0031-5

SCHWERPUNKTBEITRAG

Hauptspeicherdatenbanken für Unternehmensanwendungen

Datenmanagement für Unternehmensanwendungen im Kontext heutiger Anforderungen und Trends

Jens Krueger · Martin Grund · Christian Tinnefeld · Benjamin Eckart · Alexander Zeier · Hasso Plattner

Angenommen: 11. Oktober 2010 / Online publiziert: 5. November 2010

Zusammenfassung Unternehmensanwendungen werden traditionell in OLTP (Online Transactional Processing) und OLAP (Online Analytical Processing) unterteilt. Während sich viele Forschungsaktivitäten der letzten Jahre auf die Optimierung dieser Trennung fokussieren, haben – im Speziellen während des letztes Jahrzehnts – sich sowohl Datenbanken als auch Hardware weiterentwickelt. Einerseits gibt es Datenmanagementsysteme, die Daten spaltenorientiert organisieren und dabei ideal das Anforderungsprofil analytischer Anfragen abdecken. Andererseits steht Anwendungen heute wesentlich mehr Hauptspeicher zur Verfügung, der in Kombination mit der ebenfalls wesentlich gesteigerten Rechenleistung es erlaubt, komplette Datenbanken von Unternehmen komprimiert im Speicher vorzuhalten. Beide Entwicklungen ermöglichen die Bearbeitung komplexer analytischer Anfragen in Sekundenbruchteilen und ermöglichen so komplett neue Geschäftsprozesse und -applikationen. Folglich stellt sich die Frage, ob die künstlich eingeführte Trennung von OLTP und OLAP aufgehoben werden kann und sämtliche Anfragen auf einem vereinten Datenbestand arbeiten können. Dieser Artikel betrachtet hierfür die Charakteristiken der Datenverarbeitung in Unternehmensanwendungen und zeigt wie ausgesuchte Technologien die Datenverarbeitung optimieren können. Ein weiterer Trend ist die Verwendung von Cloud Computing und somit die Auslagerung des Rechenzentrums zur Kostenoptimierung. Damit einher gehen Anforderungen an das Datenmanagement hinsichtlich dynamischer Erweiterung und Skalierung um dem Konzept des Cloud Computings gerecht zu werden. Die Eigenschaften spaltenorientierter Hauptspeicherdatenbanken bieten hier Vorteile, auch in Bezug auf die effektivere Auslastung der zur Verfügung stehenden Hardwareressourcen.

Ein wichtiger Aspekt ist, dass alle Anfragen in einer definierten Reaktionszeit erfolgen auch wenn die Last stark schwanken kann. Erfahrungsgemäß steigt insbesondere am Ende eines Quartals die Belastung der vorhandenen Datenbanksysteme. Um hierfür immer genau die richtige Hardwareressourcen zur Verfügung zu haben, eignet sich Cloud Computing. Aus der gewünschten Elastizität ergeben sich Anforderungen an das Datenmanagement, die im Artikel betrachtet werden.

Schlüsselwörter Hauptspeicherdatenbanken · Spaltenorientierung · Unternehmensanwendungen · Datenmanagementsysteme · Cloud Computing

J. Krueger (✉) · M. Grund · C. Tinnefeld · B. Eckart · A. Zeier · H. Plattner
August-Bebel-Str. 88, 14482 Potsdam, Deutschland
e-mail: jens.krueger@hpi.uni-potsdam.de

M. Grund
e-mail: martin.grund@hpi.uni-potsdam.de

C. Tinnefeld
e-mail: christian.tinnefeld@hpi.uni-potsdam.de

B. Eckart
e-mail: benjamin.eckart@hpi.uni-potsdam.de

A. Zeier
e-mail: alexander.zeier@hpi.uni-potsdam.de

H. Plattner
e-mail: hasso.plattner@hpi.uni-potsdam.de

1 Einleitung

Die Datenhaltung in Unternehmensanwendungen hat sich die letzten Jahrzehnte kaum geändert. Zum Einsatz kommen relationale Datenbanken, deren Architektur anhand der vor rund 20 Jahren definierten Charakteristiken für transaktionale Datenverarbeitung entworfen wurde. Mit

der Zeit haben sich jedoch die Anforderungen der Unternehmensanwendungen geändert. Einerseits sind kürzere Prozesslaufzeiten in immer komplexeren Anwendungen notwendig, andererseits sollen möglichst aktuelle Daten für analytische Anfragen zur Entscheidungsunterstützung zum Einsatz kommen. Daneben sorgen in heutigen Unternehmensanwendungen automatisierte Prozesse für ein größeres Datenaufkommen, welches effizient – transaktional wie auch analytisch – verarbeitet werden muss.

Hinzu kommt, dass heutige Unternehmensanwendungen in ihrer Architektur mit der Zeit komplexer geworden sind, um steigenden Anforderungen gerecht zu werden und fehlende Eigenschaften des Datenmanagementsystems auszugleichen. Beispiele hierfür sind die redundante Speicherung von Aggregaten, die Materialisierung vorberechneter Ergebnismengen in dedizierten Tabellen oder auch die Auslagerung von Prozessen in spezialisierte Systeme oder asynchrone Programme. Während die meisten dieser Varianten die redundante Ablage der Daten als Lösung nutzen, um adäquate Antwortzeiten bestimmter Anfragen zu erreichen, wird dabei gleichzeitig durch die Notwendigkeit einer vordefinierten Materialisierungsstrategie die Flexibilität des Systems verringert. Daneben führt diese höhere Komplexität zu einer Steigerung der Gesamtkosten des Systems.

Cloud Computing als Trend der letzten Jahre bietet zudem das Potential der Kosteneinsparung. Sowohl Anbieter als auch Kunden profitieren davon, da die Kosteneinsparungen betriebsseitig an Dienstenutzer weitergeben werden. Zusätzlich bietet die dem Cloud-Konzept inhärente Flexibilität in der Nutzung samt angepassten Bezahlmodel die Möglichkeit für Kunden Kosten zu optimieren, da nur wirklich benötigte Rechenkapazitäten eingekauft werden. Bezogen auf das Datenmanagement ergeben sich daraus Anforderungen, die es gilt im Kontext von Unternehmensanwendungen und der Architektur zu berücksichtigen.

In diesem Artikel wird untersucht inwiefern Technologien in der Datenverarbeitung die Anforderungen und Datencharakteristiken moderner Unternehmensanwendungen optimal unterstützen können. Abschn. 2 widmet sich neuesten Trends im Umfeld des Datenmanagements für Unternehmensanwendungen. Abschn. 3 analysiert die aktuellen Anforderungen von Geschäftsanwendungen und wie diese mit spaltenorientierten Hauptspeicherdatenbanken adressiert werden können. Abschn. 4 zeigt auf, welche Vorteile und Neuerungen durch die Verwendung von spaltenorientierten Hauptspeicherdatenbanken möglich sind. Bisherige Arbeiten im diesem Bereich werden in Abschn. 5 vorgestellt. Der Artikel endet mit einer Zusammenfassung und einem Ausblick in Abschn. 6.

2 Trends in der Datenverarbeitung

In diesem Abschnitt wird auf verschiedene Entwicklungen im Bereich der Datenverarbeitung eingegangen, die einzeln betrachtet zwar nicht neu sind – bspw. wurde die vertikale Partitionierung (Spaltenorientierung) schon Ende der 70er betrachtet – jedoch ergeben sich aus dem Zusammenführen neue Einsatzgebiete, deren Anforderungen abgeleitet sind von heutigen Unternehmensanwendungen. Dies folgt einem weiteren Trend der letzten Jahre, der spezialisierte Datenbanken im Vorteil gegenüber so genannten allgemeinen Datenbanken (engl. General Purpose Databases) sieht [32] und somit ein auf Unternehmensanwendungen spezialisiertes Datenmanagement motiviert, wobei das Verarbeiten von Daten rein im Hauptspeicher eine zentrale Rolle spielt. Zu erwähnen ist, dass dedizierte Datenbanken für transaktionale sowie analytische Verarbeitung das Konzept der Spezialisierung hinsichtlich der Anforderungen der konsumierenden Anwendung berücksichtigen. Dem entgegen steht der Fakt, dass moderne Unternehmensanwendungen einen grossen Teil ihres Mehrwertes aus der latenzfreien Integration von Daten und Prozessen beider Bereiche ziehen. Dementsprechend ist ein auf Unternehmensanwendungen spezialisiertes Datenmanagement notwendig um diese Anforderung effizient umzusetzen. Ausgehend von der konventionellen Trennung wurde, z.B. in [24], gezeigt, dass eine Neuaufnahme der Anwendungscharakteristiken unter Berücksichtigung aktueller Trends notwendig ist, um genau dieses zu erreichen.

Zum einem hat die Entwicklung der Hardware in den letzten Jahren dazu geführt, dass Hauptspeicherdatenbanken, allein schon von der zur Verfügung stehenden Größe des Hauptspeichers, realisierbar wurden und somit in den Fokus der Forschung gelangten. Zum anderen bieten Hauptspeicherdatenbanken, welche die Daten spaltenorientiert speichern, die Möglichkeit bestehende transaktionale und analytische Datenbanken wieder zu vereinen, da sie eine wesentlich höhere Geschwindigkeit im Datenzugriff bieten und somit analytische Anfragen auf Basis transaktionaler Daten im Sekundenbereich beantwortet werden können [28]. Umgesetzt wird dies durch die Kombination verschiedener Technologien im Management von Daten bei gleichzeitiger Berücksichtigung unternehmensspezifischer Datencharakteristiken (vergl. Abschn. 3). Neben Datenstrukturen, die optimiert sind für den Zugriff im Hauptspeicher, kommen zum Beispiel Kompressionsverfahren und Insert-Only Verfahren zum Einsatz, welche im Folgenden ebenfalls besprochen werden.

Ein weiterer Trend der letzten Jahre ist das so genannte Cloud Computing, welches Rechenleistung und Anwendungen als Dienstleistungen dynamisch skalierbar bereitstellt. Demnach wird am Ende des Abschnittes auf das Cloud

Computing eingegangen, wobei spaltenorientierte Hauptspeicherdatenbanken und dessen Bezug in diesen Rahmen im Fokus stehen.

2.1 Hauptspeicherdatenbanken

Wie bereits erwähnt stehen heute Server mit Hauptspeicherkapazitäten zur Verfügung, die die Größe von einzelnen Festplatten erreichen können. Derzeit stehen bereits Server mit 2TB Hauptspeicher regulär zur Verfügung. Somit besteht die Möglichkeit komplette Datenbank im Hauptspeicher abzulegen und so jegliche Anfragen an das Datenbankmanagementsystem (DBMS) direkt aus dem Hauptspeicher zu beantworten. Solche DBMS werden In-Memory- (IMDB) oder auch Main-Memory-Datenbanken genannt (MMDB) [11, 13]. Grundsätzlich können heutige konventionelle Datenbanksysteme Hauptspeicher zum Zwischenlagern von Daten verwenden, wobei ein Bufferpool für die Datenverwaltung sorgt. Allerdings sorgen dieser sowie die Eigenschaften der für den Festplattenzugriff optimierten Datenstrukturen, wie z.B. B + -Bäume und Pages, einerseits für einen höheren Verwaltungsaufwand und andererseits für die nicht-optimale Ablage im Speicher im Vergleich zu Datenbank, welche für den Hauptspeicher konzipiert wurden. Dennoch stellt sich die Frage inwieweit die kompletten Daten eines Unternehmens im Speicher vorgehalten werden können. Eine wichtige Erkenntnis hierbei ist, dass obwohl die Datenvolumen von Unternehmensanwendungen stetig steigen, wachsen diese Datenmengen im Allgemeinen langsamer als unstrukturierte Daten, wobei die Betrachtung nur transaktionale und analytische Systeme berücksichtigt, die nur strukturierbare Daten vorhalten. Der Grund für relativ geringe Wachstum ist, dass Transaktionsverarbeitung in Unternehmen auf tatsächlichen Ereignissen basiert, die an die reale Welt gebunden sind, wie die Anzahl der Kunden, Produkte und anderen Entitäten in der realen Welt. Diese Events der Transaktionsverarbeitung wachsen nicht so schnell wie zuletzt Hauptspeicherkapazitäten. Hinzukommt, dass aufgrund organisatorischer oder funktional zugrundeliegender Partitionierung die meisten Systeme die aktive Größe von 1 TB nicht überschreiten sowie unstruktierte Daten hier im Kontext von Hauptspeicherdatenbanken nicht betrachtet werden, auch wenn andere Arbeiten, fokusiert auf den speziellen Anwendungsfall, Vorteile aufgezeigt haben [34].

Verglichen mit dem Wachstum der Hauptspeicherkapazitäten, haben sich die Latenz des Speicherzugriffes sowie die Speicherbandbreite nur geringfügig verbessert. Um die Auslastung der Bandbreite zu optimieren, nutzen daher moderne Prozessoren mehrstufige Cache-Architekturen um die Latenz beim Speicherzugriff zu verschleiern. Durch solche Architekturen lassen sich zudem Bandbreitenbeschränkungen beim Speicherzugriff umgehen oder zumindest einschränken [3, 5]. Diese so genannten Speicherhierarchien nutzen

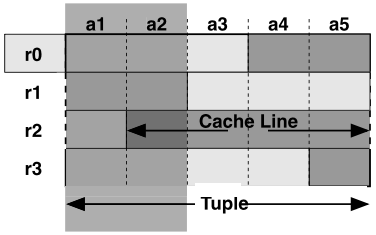

Abb. 1 Datenzugriff: Tupelgröße versus Cache-Line-Größe

den Fakt aus, dass kleinere aber dafür wesentlich schnellere Speichertechnologien dichter an der CPU operieren. Diese führen jedoch dazu, dass performante Zugriffe im Hauptspeicher davon abhängig sind. Essentiell ist des Weiteren der Fakt, dass die Daten in Cache-Lines gelesen werden, welche in der Regel 64 Byte großsind. Dies hat zur Folge, dass der Speicherzugriff blockweise geschieht und somit sämtliche Algorithmen und Datenstrukturen daraufhin optimiert werden müssen, um die vorhandene Architektur effizient zu nutzen [15, 21].

In [26] demonstrieren Manegold et al., dass die Bearbeitungszeit eines speicherinternen Datenzugriffes durch die Anzahl der Cache-Misses bestimmt werden kann. Somit können CPU-Zyklen und Cache-Misses als Äquivalent der Zeit genutzt werden und folglich ist für die Performanzverbesserung Hauptspeicher-lastiger Systeme vor allem eine Optimierung der Cache-Nutzung bzw. Reduktion der Cache-Misses von Wichtigkeit.

Um die bestmögliche Performanz zu erreichen, werden Daten wenn möglich sequentiell gelesen, so dass die geladene Cache-Line möglichst vollständig genutzt wird und nur benötigte Daten gelesen werden. Abb. 1 zeigt eine Beispieltabelle eines zeilenorientierten Speichersystems in welcher alle Attribute einer Zeile die Breite der Cache-Line überschreiten. Das Lesen einer kompletten Zeile führt daher zu einem Cache-Miss. Eine mögliche Optimierung bezüglich der Cache-Nutzung wäre bspw. eine Schemaänderung der Tabelle, welche die Attribute anhand ihrer Anfragehäufigkeit sortiert. Somit steigt bei Anfragen – welche nicht alle Attribute lesen – die Wahrscheinlichkeit, dass die Cache-Lines weniger ungenutzte Attribute enthalten und dem zufolge weniger Cache-Lines gelesen werden müssen. Alternativ kann die physikalische Repräsentation der Daten dem zu erwartenden Workload angepasst werden. Insbesondere die spaltenweise Organisation der Daten im Hauptspeicher bietet bei lesenden, auf wenige Attribute anfragende Datenbankabfrage viele Vorteile hinsichtlich des Cacheverhaltens, da das sich daraus ergebende sequentielle Verarbeiteten für die Nutzung der kompletten Cache-Line sorgt.

Bei IMDB's liegt folglich der Fokus auf der Optimierung des physischen Datenlayouts, welches hinsichtlich des tatsächlichen vorkommenden Zugriffsmuster optimiert wird. Diese Muster werden normalerweise durch gegebene OLTP-

oder OLAP-Workloads vorab definiert. Darüber hinaus ist für effizienten Zugriff sicherzustellen, dass die verfügbare Speicherbandbreite möglichst ausgenutzt wird und Daten, wenn möglich, sequentiell gelesen werden. Diese Anforderung ist analog zu festplattenbasierten Datenbanksystemen, da diese ebenfalls profitieren, wenn ganze Blöcke gelesen und auch genutzt werden können, jedoch gelten andere Parameter die dazu führen, dass andere Algorithmen und Datenstrukturen zum Einsatz kommen.

2.2 Spaltenorientierte Datenbanken

Durch die steigende Nachfrage nach analytischen Funktionen wie Ad-hoc Anfragen auf transaktionalen Daten rückten im letzten Jahrzehnt spaltenorientierte Datenbanken – vor allem im Umfeld der OLAP-Anwendungen – in den Fokus der Wissenschaft.

Spaltenorientierte Datenbanken basieren auf einer vertikalen Partitionierung nach Spalten, welche Ende der 70er Jahre zuerst betrachtet wurde und von Copeland und Khoshafian 1985 [9] weiterentwickelt wurde. Jede Spalte wird hierbei separat gespeichert, während das logische Tabellenschema durch die Einführung von eindeutigen Positionsschlüsseln beibehalten wird. Diese Verwendung der Positionsschlüsseln führt zu zusätzlichem Speicherbedarf, kann jedoch implizit mit Hilfe von Positionsinformationen der Attribute in den Spalten vermieden werden. Letzteres verhindert eine Sortierung der einzelnen Attribute, bietet jedoch effizienteres Lesen im Falle des Zugriffs weiterer Attribute, da der implizite Positionsschlüssel direkt zum Adressieren benutzt werden kann.

Die spaltenweise Organisation der Daten bietet insbesondere Vorteile im Lesen weniger Attribute, da für die Anfrage unnötige Spalten nicht gelesen werden müssen, wie es z.B. bei der zeilenorientierten Struktur auf Festplatten der Fall ist. Demnach wird allein schon durch die Reduzierung der zu lesenden Datenmenge für attributfokusierte Anfragen, wie sie im analytischen häufig Fall vorkommen, ein Geschwindigkeitsvorteil erreicht [12]. Dem entgegen steht der Nachteil, dass die vertikale Partitionierung zu Mehrkosten im Vergleich zur konventionellen zeilenorientierten Struktur beim Herstellen kompletter Relationen führt. Im Falle eines Festplatten-basierten Datenbanksystems muss hierfür auf sämtliche Dateien zugegriffen werden, so dass das sequentielle Lesen eines Tupels durch einen zufälligen und damit langsamen Zugriff ersetzt wird. Gleiches gilt für den schreibenden Zugriff, da hier die komplette Relation auf die Dateien verteilt werden muss.

Insbesondere im Kontext von transaktionalen bzw. kombinierten Workloads haben die gerade genannten Nachteile den Einsatz des spaltenorientierten Schemas bisher verhindert. Erst die Kombination dieses Ablagemusters mit einer Hauptspeicherdatenbank ermöglicht den Einsatz in solchen

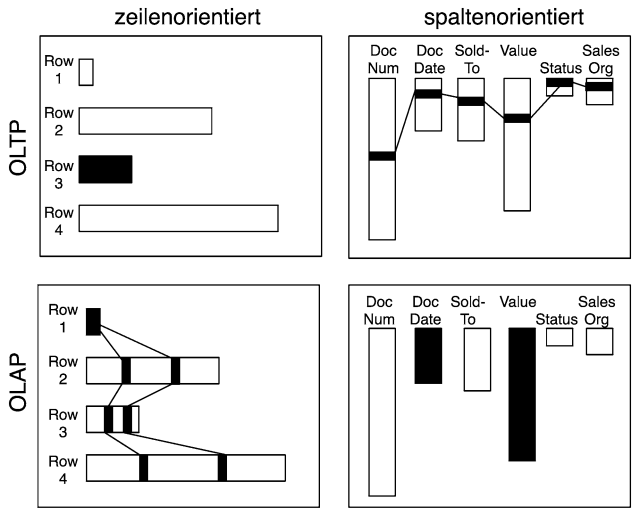

Abb. 2 Datenzugriff: Zeilen- und Spaltenorientiert

Szenarien. Hinzu kommt, dass sich die Annahmen hinsichtlich transaktionaler Verarbeitung über die Jahre geändert haben. Wie in Abschn. 3 gezeigt wird, sind in der transaktionalen Welt die meisten Anfragen lesender Natur. Auch wenn diese zu einem großen Teil aus Schlüsselzugriffen bestehen, kann eine spaltenorientierte Hauptspeicherdatenbank Vorteile ausspielen, da das Zusammensetzen kompletter Relationen durch die schnelle „random access"-Funktionalität des Hautspeichers im Vergleich zur Festplatte den Nachteil der vertikalen Dekomposition ausgleichen kann. Dies trifft vor allem dann zu, wenn Anfragen zumeist auf wenige Attribute zugreifen und dabei eine große Anzahl an Zeilen lesen. Hier kommt die Möglichkeit des sequentiellen Lesens auf dedizierte Spalten zum Tragen, der um Faktoren schneller ist als der wahlfreie Zugriff, was für Hauptspeicher- wie auch Festplatten-basierte Systeme gilt.

Die Abb. 2 verdeutlicht die konventionellen Zugriffsmuster für Unternehmensanwendungen im Zusammenhang mit der physikalischen Datenorganisation. Ersichtlich wird, dass bei der Kombination von OLTP und OLAP eine der Speicherablagen gewählt werden muss, die die Gesamtkosten aller Anfragen minimiert. Die Autoren von [23] und [28] zeigen den Vorteil der Spaltenorientierung im Umfeld von Unternehmensanwendungen auf.

Des Weiteren profitieren spaltenorientierte Datenbanken von der geringen Kardinalität unterschiedlicher Werte innerhalb einer Spalte. Wie in späteren Abschnitten anhand von Kundenanalysen gezeigt wird, liegen solch geringe Kardinalitäten in der Natur von Unternehmensanwendungen. Dieser Fakt sowie die Eigenschaft, dass in spaltenweiser Datenorganisation die Spalten separat verwaltet werden, führt zu der Möglichkeit Attribute einzeln zu komprimieren unter dem Vorteil, dass der Wertebereich einer dedizierten Spalte inhaltlich begrenzt ist und somit effektiv komprimiert werden kann.

2.3 Kompression in Datenbanken

Das Moore'sche Gesetz besagt, dass sich die Rechenleistung von Prozessoren alle 18 Monate verdoppelt und gilt nun mehr seit über 35 Jahren. Dies gilt hingegen nicht für die Entwicklung von Festplatten- oder Hauptspeicherzugriffsgeschwindigkeiten, so dass für die Mehrzahl der Anfragen trotz Cache-optimierter Datenstrukturen zunehmend I/O-Verzögerungen den größten Engpass darstellen. Die wachsende Kluft zwischen den Geschwindigkeitszuwächsen der CPUs auf der einen Seite und des Speichers auf der anderen Seite ist ein bekanntes Problem und wird bspw. in [5] sowie [25] beschrieben. Leichtgewichtige und verlustfreie Kompressionstechniken helfen diesen Flaschenhals zu verringern, indem sie es erlauben die zur Verfügung stehende Bandbreite effizienter zu nutzen, wobei mögliche Mehrkosten zum Entpacken aufgrund der im Vergleich gesteigerten CPU-Leistung aufgefangen werden.

Kompressionsverfahren nutzen Redundanz innerhalb der Daten und Wissen über die jeweilige Datendomäne, um möglichst effektiv zu komprimieren. Aufgrund mehrerer Eigenschaften von spaltenorientierten Datenstrukturen ist Komprimierung bei diesen besonders effizient [1]. Da alle Daten innerhalb einer Spalte den selben Datentypen aufweisen und in der Regel eine ähnliche Semantik besitzen, ergibt sich ein niedriger Informationsgehalt (Entropie), d.h. es gibt in vielen Fällen nur wenige unterschiedliche Werte. Insbesondere Unternehmensanwendungen, deren Aufgabe es ist sich wiederholende Prozesse und Events abzuarbeiten bzw. festzuhalten, schöpfen so gut wie nie den Wertebereich des zur Verfügung stehenden Datentypes aus. Sehr oft ist es der Fall, dass sogar nur sehr wenige Werte verwendet werden, da z.B. der Betrieb nur einige verschiede Materialien und Produkte verwendet.

Bei der Lauflängenkodierung (engl. Run-length-encoding, RLE) wird die Wiederholung eines Wertes durch ein Wertepaar (Wert und Wertanzahl) gespeichert. Zum Beispiel wird die Sequenz „aaaa" auf „a[4]" komprimiert. Dieser Ansatz eignet sich besonders für sortierte Spalten mit wenig Varianz in den Attributwerten. Bei unsortierten Spalten sinkt die Effektivität, so dass ein Sortieren obligatorisch erscheint. In einem spaltenorientierten Szenario wird dadurch eine ID für die Zeile notwendig, die zu dem oben genannten Mehraufwand führt. Bei unsortierten Spalten eignet sich vor allem Bit-Vektor-Encoding. Im Wesentlichen wird hierbei ein häufig auftretender Attributwert innerhalb einer Spalte mit einem Bit-String assoziiert, wobei die Bits die Positionen innerhalb der Spalte referenzieren und nur diejenigen Bits den auftretenden Attributwert durch deren Position festlegen. Die Spalte wird dann ohne den Attributwert gespeichert und kann in Kombination mit dem Bit-Vektor rekonstruiert werden. Dieses Verfahren ist ideal, wenn die Kardinalität der verschiedenen Werte gering ist oder ein Wert in der Menge dominiert.

Ein weiteres prominentes Beispiel für leichtgewichtige Kompressionstechniken ist das Dictionary-Encoding, bei der häufig erscheinende Muster durch kürzere Symbole ersetzt werden. Im Falle eines Attributes einer Tabelle wird jeder eindeutige Wert in einem Wörterbuch abgelegt und der Wert in der Spalte durch einen Schlüssel ersetzt, der auf den Wert im Wörterbuch zeigt. Eine weitere Optimierung ist die Bit-weise Komprimierung dieser Schlüssel um die Länge so zu reduzieren, dass nur die notwendige Anzahl im Wörterbuch damit abgebildet werden kann, was die Effektivität dieses Verfahrens weiter erhöht, insbesondere wenn nur wenige verschiedene Werte vorliegen. Ein sortiertes Wörterbuch hingegen erlaubt zusätzliche Optimierungen auf der Anfrageseite, da sich von den Schlüsseln die originale Sortierung ohne Abgleich mit dem Wörterbuch ableiten lässt.

Durch die Komprimierung innerhalb der Spalten wird die Informationsdichte in Bezug auf den verbrauchten Speicherplatz erhöht. Dadurch können relevantere Informationen zur gleichzeitigen Verarbeitung in den Cache geladen werden, was zu einer Erhöhung der Bandbreitennutzung führt. Zu erwähnen ist, dass diese Techniken auch für zeilenorientierte Speicherschemata gelten, jedoch erst in der Kombination mit spaltenorientierter Ablage und der Datenverarbeitung im Hauptspeicher Vorteile im betrachteten Rahmen erzeugen, da die sich aus der zeilenbasierten Ablage ergebenden Cache-Misses den Nutzen der Kompression zunichte machen.

Ein weiterer wichtiger Aspekt der Komprimierung ist der Nachteil, dass Datenmodifikationen eine erneute Komprimierung des Datenbestandes herbeiführen. Um diesen Aufwand zu umgehen und dennoch schnelle Einfüge-, Änderungs- und Löschoperationen zu erlauben, können Änderungen in einem Puffer gesammelt [22] werden, der Datensätze unkomprimiert ablegt. Das Ergebnis jeder Anfrage besteht aus dem komprimierten Hauptspeicherbereich und dem Puffer, so dass auch noch nicht komprimierte Änderungen berücksichtigt werden. Um die Änderungen in den komprimierten Hauptspeicherbereich zu übertragen, werden in definierten Intervallen die gesammelten Änderungen des Puffers asynchron mit dem bestehenden komprimierten Datenbestand zusammengeführt.

Während frühere Arbeiten wie [10] den Fokus auf die Verbesserung der I/O-Leistung durch Reduzierung der zu ladenden Daten behandelten, konzentrieren sich aktuelle Arbeiten [1, 35] auf die Auswirkungen der Komprimierung hinsichtlich der Anfrageausführung, z.B. durch Verwendung von „late materialization"-Strategien. Hierbei stehen leichte Kompressionen im Fokus, die einen direkten Zugriff auf einzelnen Werte erlauben, ohne dass der komplette Datenbestand entpackt werden muss.

Darüber hinaus kann die Leistung dadurch verbessert werden, dass die spaltenorientierte Anfrageausführung –

wie in [2] beschrieben – Kenntnis über das verwendete Komprimierungsverfahren hat, so dass die CPU während des Einlesens der Daten diese parallel dekomprimieren kann. Ein wichtiger Aspekt bei der Komprimierung ist dabei die Abwägung zwischen Komprimierungsverhältnis und den Kosten für die Dekomprimierung. Generell ist es das Ziel die Dekomprimierung so lange wie möglich hinauszuzögern, um einen möglichst großen Nutzen aus den komprimierten Daten zu ziehen und nur wirklich benötigte Daten zu dekomprimieren.

2.4 Insert-Only Strategien

Insert-Only Datenbanken definieren eine Kategorie der Datenbankmanagementsysteme, bei der neue Datensätze grundsätzlich nur hinzugefügt werden. Das bedeutet jedoch nicht, dass logisch gesehen Datensätze nicht gelöscht oder verändert werden können. Diese modifizierenden Operationen werden in ein technisches Anfügen umgesetzt, wobei ein Zeitstempel mitgeschrieben wird. Eingefügte Datensätze sind nur für einen Zeitraum gültig und Werte daher zeitabhängig. Dadurch wird die komplette Historie aller Datenmodifikationen persistiert.

Das Speichern der kompletten Historie aller Unternehmensdaten ist insbesondere im Unternehmenskontext wichtig, da die Verfolgung und Speicherung dieser Historie in vielen Ländern gesetzlich vorgeschrieben ist. Ein Beispiel für die Notwendigkeit von historischen Daten ist ein Kunde, welcher eine Beschwerde zu einer Rechnung hat, die an eine alte Adresse versandt wurde. In diesem Fall muss diese weiterhin rekonstruierbar sein. Ein anderes Beispiel wäre die Änderung eines Buchhaltungsbeleges. Hier ist eine Änderungshistorie aus betriebswirtschaftlichen Gründen zwingend notwendig.

Des Weiteren ermöglicht diese Art der Datenspeicherung historische Daten zu analysieren, da jeder Zeitpunkt wiederhergestellt werden kann, was Vergleiche mit vergangenen Datenständen erlaubt. Diese Funktionalität wird heutzutage in der Regel über ein Data Warehouse implementiert, in welchem auch ältere Zustände gesichert bleiben und über eine Zeitdimension exponiert werden.

Zusätzlich zu den gerade genannten Vorteilen, ermöglicht das Vorhalten alter Versionen auch *Snapshot Isolation*, d.h. ohne explizites Sperrverfahren kann die Datenbank weiterhin garantieren, dass eine Transaktion über die gesamte Laufzeit auf Daten operieren kann, die nicht durch andere Transaktionen verändert worden sind. Umgesetzt wird diese Konsistenz bei lesenden Anfragen durch einen Zeitstempel, mittels dessen die Version der Daten zum Beginn der Transaktion hergestellt werden kann. Dies vereinfacht das Datenmanagement und reicht für Unternehmensanwendungen meist aus, da aufgrund komplexerer Sperrlogik ein auf Anwendungsebene implementiertes Sperren der zu bearbeitenden Daten ohnehin notwendig ist.

Für die Speicherung zeitabhängiger Werte gibt es mehrere Verfahren. Bei der diskreten Darstellung werden Delta-Sätze für die Speicherung der Änderungen genutzt. Hierbei müssen alle älteren Versionen gelesen werden, um den aktuell gültigen Datensatz zu bestimmen. Um einen Dateneintrag zu modifizieren muss hierbei nicht die ganze Zeile gelesen werden, da alle nicht geänderten Werte bspw. mit einem vordefinierten Standardwert wie einer *NichtGeändert*-Markierung gefüllt werden können. Dies reduziert die zu lesenden Datenvolumen, welche für das Einfügen nötig sind. Außerdem müssen für diese Operation nicht alle Spalten gelesen werden. Der Hauptnachteil dieser Methode ist darin begründet, dass alle vorigen Versionen gelesen werden müssen, um die aktuell gültige Version erstellen zu können. Dies wird umso teurer, je mehr Modifikationen über die Zeit anfallen.

Die zweite Möglichkeit, um zeitbasierte Werte in einer Zeile zu speichern, ist die Intervallspeicherung. Der Unterschied hierbei ist, dass für jede Zeile ein Gültigkeitsintervall gespeichert wird. Beim Einfügen einer neuen Zeile, wird der aktuelle Zeitpunkt als Startpunkt gesetzt und der Endpunkt offen gelassen. Falls nun diese Zeile modifiziert wird, wird der Endpunkt der Zeile gesetzt und eine neue Zeile mit eben diesem Zeitpunkt als Startpunkt eingefügt. Obwohl die alte Zeile hierbei gelesen werden musste, gibt es hierbei Vorteile bei der Suche, da die vorher gültige Zeile dank des gespeicherten Intervalls leicht auffindbar ist und somit nicht alle Zeilen gelesen werden müssen.

Ein weiterer oft angeführter Kritikpunkt an Insert-only Techniken ist der erhöhte Speicherverbrauch, da alle historischen Daten gespeichert werden. Dem ist entgegenzusetzen, dass einerseits die modifizierenden Operationen im Unternehmensanwendungskontext wesentlich weniger vorkommen als angenommen (vergl. Abschn. 3) und andererseits muss bspw. beim Dictionary-Encoding das Wörterbuch kaum modifiziert werden bzw. wächst in der Größe nur marginal, da eine Vielzahl der Attributwerte bereits eingetragen ist.

2.5 Cloud Computing

Im Konzept des Cloud Computings laufen Systeme nicht beim Kunden (*on-premise*), sondern werden je nach Nachfrage (*on-demand*) genutzt und skaliert. Das Betriebsrisiko des Rechenzentrums wird zudem vom Unternehmen zum Betreiber der Cloud ausgelagert. Einher der Vorteil, dass für Konsumenten keine Kapitalkosten anfallen und durch Ausnutzen von Skaleneffekten Betriebskosten auf Seiten des Betreibers minimiert werden können. Die kundenseitigen Kosten können zudem durch Ausnutzen der Elastizität des Cloudkonzeptes verringert werden. Unternehmen müssen nur für die benötigte Rechenleistung zahlen. Wird weniger oder mehr Rechenleistung benötigt, kann diese vom

Anbieter automatisiert über eine Schnittstelle bereitgestellt werden. Während bei klassischen Rechenzentren die Hardware für eine maximale Last dimensioniert werden muss, kann bei Verwendung von Computing Clouds nur die wirklich benötigte Hardwareressource bezogen werden und diese je nach Auslastung erhöht oder verringert werden.

Neben dem Auslagern von Rechenzentren können Unternehmen auch komplette Applikationen als Dienst beziehen. Dies wird in der Regel als Software-as-a-Service bezeichnet. Im Rahmen von Soft ware-as-a-Service gibt es drei verschiedene Schichten, auf denen Dienste angeboten werden. Komplette Anwendungen können als Dienst bezogen werden. Hierbei handelt es sich um Appli cation-as-a-Service. Sofern ein Anbieter nur eine Plattform zur Verfügung stellt, z.B. eine Datenbank als Dienst, spricht man von Platform-as-a-Service. Auf unterster Ebene stellt der Anbieter ausschließlich Infrastruktur wie Rechner oder virtuelle Maschinen zur Verfügung und man spricht von Infrastructure-of-a-Service.

Die Prioritäten für Datenbanksysteme, welche *on-premise* exklusiv für einen Kunden laufen, sind nicht identisch mit Systemen, die als Platform-as-a-Service oder in Kombination mit einer Anwendung als Application-as-a-Service angeboten werden. Letztere werden von einem Service Provider einem größeren Kreis von Kunden angeboten und müssen daher mit vielen verschiedenen Kunden skalieren, dürfen jedoch nur geringe Kosten für den Service Provider verursachen, damit dieser die Anwendung gewinnbringend anbieten kann.

Vor allem bei Datenbanksystemen, welche entweder als Service direkt oder indirekt, von Anwendungen die als Service angeboten werden, genutzt werden, ist es wichtig, dass diese einen hohen Durchsatz bei zugleich garantierter hoher Geschwindigkeit erreichen. Aus Betreibersicht gesehen sollen zudem die Hardwareressourcen Da bei traditionellen Datenbanksystemen gerade die Festplatte der limitierende Aspekt ist, können durch die Verwendung von Hauptspeicherdatenbanken ein höherer Durchsatz erreicht werden, was wiederum zu geringeren Kosten (TCO) für den Service Provider führt.

Neuere Ansätze, wie RAMCloud [27], propagieren daher Cloud-Umgebungen nur noch im Arbeitsspeicher zu betreiben. Durch die Verwendung von Hauptspeicherarchitekturen ist der Flaschenhals nicht mehr der Durchsatz und die Geschwindigkeit der Festplatte, sondern die Geschwindigkeit des Prozessors und wie schnell dieser Daten aus dem Arbeitsspeicher lesen kann. Da diese Systeme wesentlich performanter sind, können so auch mehr Kunden von einem System bedient werden (*Multi-Tenancy*) und die vorhandene Hardware noch besser ausgelastet werden, was wiederum zu geringeren Kosten führt.

Das Datenbankmanagementsystem sollte in der Lage sein alle Benutzeranfragen in einer definierten Zeit beantworten zu können. Messungen beim Betrieb eines On-Demand Data Warehouses haben gezeigt, dass die Last in der Woche vor dem Ende eines Quartals doppelt so hoch ist, wie in der sonstigen Zeit. Bei Betrieb eines klassischen Rechenzentrums muss die Kapazitätsplanung der Server basierend auf dieser Maximallast erfolgen. Dies hat zur Folge, dass in den restlichen 48 Wochen des Jahres die vorhandene Rechenkapazität nicht voll ausgenutzt wird und Energie und somit Kosten verschwendet werden.

Wenn es zu unerwarteten Ereignissen kommt, die eine höhere Anzahl von Abfragen erfordern, z.B. bei sich ankündigenden Wirtschafts- oder Firmenkrisen, kann es zudem sein, dass die vorhandene geplante Rechenkapazität nicht mehr ausreicht. Das System wäre dann nicht mehr in der Lage auf zeitkritische Anfragen oder Transaktionen zu reagieren.

Mit Cloud Computing hingegen besteht die Möglichkeit Rechenkapazitäten je nach aktueller Nachfrage skalieren zu können, so dass unabhängig von der aktuellen Datenbanklast immer die richtige Menge an Hardwareressourcen zur Verfügung steht, um alle Anfragen und Transaktionen in vordefinierten Zeitfenstern ausführen zu können. Um die Elastizität der Cloud ausnutzen zu können, muss die Hauptspeicherdatenbank horizontal skalierbar sein. Die Hauptspeicherdatenbank muss also Rechenkapazität nutzen können, welche in Form von weiteren Servern bereit gestellt wird.

Um die benötigten Hardwareressourcen abschätzen und vorbestimmen zu können, werden Modelle benötigt, die die Arbeitslast beschreiben können. Mit Hilfe solcher Modelle kann dann die benötigte Rechenkapazität berechnet werden. Der Zugriff von Hauptspeicher lässt sich leichter in Form von Modellen beschreiben, da im Gegensatz zur Festplatte die Zeit des Datenzugriffes nicht von der Seek Time abhängt, die je nach Position des Lesekopfes der Festplatte variieren kann.

3 Anforderungen von Geschäftsanwendungen

Aufgrund steigender Datenvolumen, neuen Anforderungen an Geschäftsprozesse und steigender Nachfrage für zeitnahe Geschäftsdaten, werden Unternehmensanwendungen zunehmend komplizierter, um Mängel in der Datenmanagement-Infrastruktur ausgleichen zu können. Dies schließt bspw. die redundante Speicherung von transaktionalen und analytischen Daten ein. Der Grund dafür ist, dass heutige Datenbanksysteme die Anforderungen moderner Unternehmen nicht erfüllen können, da diese zumeist für lediglich einen Anwendungsfall optimiert sind: entweder für OLTP- oder OLAP-Verarbeitung. Dieser Fakt führt zur einer Diskrepanz von Unternehmensanwendungen in Bezug auf die zugrundeliegende Datenmanagement, da mit konventionellen Datenbanken vor allem komplexe Operationen nicht in

annehmbarer Zeit auszuführen sind. Während dieses Problem für analytische Anwendungen weithin anerkannt ist und dafür in der Regel Data Warehouses genutzt werden, trifft es ebenso auf transaktionale Anwendungen zu, die in modernen Systemen komplex sind. Um diesen Nachteil der genutzten Datenmanagement-Infrastruktur auszugleichen, werden komplexe Operationen und länger laufende Prozesse zumeist in Batch-Jobs ausgelagert.

Infolgedessen verlangsamt diese Lösung die Geschwindigkeit mit der Geschäftsprozesse abgeschlossen werden können, wodurch möglicherweise externe Anforderungen nicht mehr erfüllt werden können. Ein weiterer und oft zum Einsatz kommender Lösungsansatz ist die Materialisierung vorberechneter Ergebnisse. Materialisierte Sichten in Data Warehouses für analytische Anwendungen sind hierfür ein Beispiel. Dies führt jedoch zu einer geringeren Flexibilität und erschwert die Wartung in solchen OLAP-Systemen. Ein analoges Verfahren findet in OLTP-Systemen Verwendung, wobei die Materialisierung von den Anwendungen verwaltet wird, da meist Anwendungslogik mit berücksichtigt werden muss. Für die redundante Speicherung von Daten durch vordefinierte Charakteristiken, wird dem Performanzproblemen entgegengetreten, während jedoch gleichzeitig eine Steigerung der Komplexität und eine Abnahme der Flexibilität akzeptiert wird. Neben der steigenden Programmkomplexität, die notwendig ist um die Sichten konsistent zu halten, stört insbesondere die fehlende Flexibilität in heutigen Anwendungen, da Anforderungen programmseitig nicht mehr umgesetzt werden können.

Zusammenfassend lässt sich sagen, dass der aktuelle Ansatz heutiger Unternehmensanwendungen zu einer steigenden Komplexität bei der Datenmodifizierung und -speicherung führt. Im folgenden Abschnitt wird gezeigt welche Charakteristiken Unternehmensanwendungen inne haben und wie sich diese auf aktuelle Software-Technologien, wie z.B. spaltenorientierte Hauptspeicherdatenbanken, anwenden lassen und welche Verbesserungen und Vereinfachungen dadurch möglich sind [23].

3.1 Anfragenverteilung

Datenbanken im Bereich des Unternehmensdatenmanagements werden allgemein abhängig von Ihrer Optimierung – entweder für OLTP oder OLAP – klassifiziert. Bislang wurde oftmals angenommen, dass der Workload von Unternehmensanwendungen aus einem relativ hohen Anteil von `insert`-, `update`- und `delete` Anfragen besteht. So besteht der TPC-C Benchmark [33] nur zu 54% aus Leseanfragen und 46% Schreibanfragen.

Um zu bestimmen, ob entweder lese- oder schreiboptimierte Datenbanken besser für solche Workloads geeignet sind, wurden 65 Installationen eines Enterprise Resource Planning Systems (ERP) in Hinsicht auf die durchgeführten Datenbankanfragen untersucht. Das Ergebnis dieser Untersuchung ist in Abb. 3 zu sehen. Diese zeigt deutlich, dass die Verteilung der Anfragen äußerst leselastig ist. Mindestens 80% aller Anfragen sind `select`-Anfragen. Dieser Fakt spricht für die Verwendung einer leseoptimierten Datenbank, gegebenenfalls mit einem schreiboptimierten Puffer wie in [6] und [22] beschrieben. Spaltenorientierte Hauptspeicherdatenbanken mit leichtgewichtiger Kompression bieten den besten Durchsatz bei komplexen Leseanfragen.

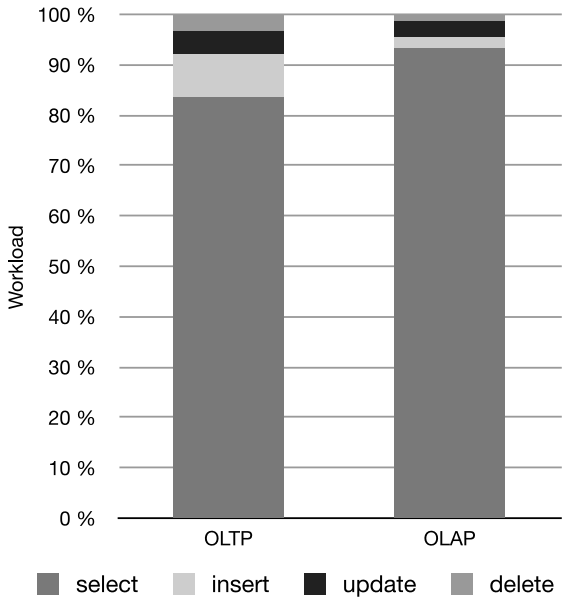

Abb. 3 Verteilung der `select`, `insert`, `update` und `delete` Operationen

3.2 Werteverteilung von Unternehmensdaten

Neben der Verteilung von Datenbankabfragen sind auch die Werteverteilung und Eigenschaften der Daten wichtig für spaltenorientierte Datenbanken. Eine weitgehende Annahme für Daten von Geschäftsanwendungen ist, dass diese eine hohe Werteverteilung aufweisen, also sehr viele verschiedene Werte für die einzelnen Attribute existieren. Untersuchungen haben gezeigt, dass Werte einzelner Spalten kaum verteilt sind, d.h.0 oftmals nur wenige unterschiedliche Werte existieren. Zudem wird oftmals nur ein sehr kleiner Teil der verfügbaren Attribute überhaupt verwendet.

Um diese Annahme zu belegen wurde die Häufigkeit der unterschiedlichen Werte pro Attribut gemessen. Diese Analyse geschah anhand der Haupttabellen der Finanzverwaltung sowie Verkaufs- und Vertriebsdaten mehrerer Kundensysteme. In der untersuchten Unternehmensanwendung besteht bspw. ein Buchunghaltungsbeleg aus rund 100 Attributen, während sich die dazugehörigen Positionen jeweils aus 300 Attributen zusammensetzen. Die Tabelle für Materialbewegungen besteht aus 180 Attributen.

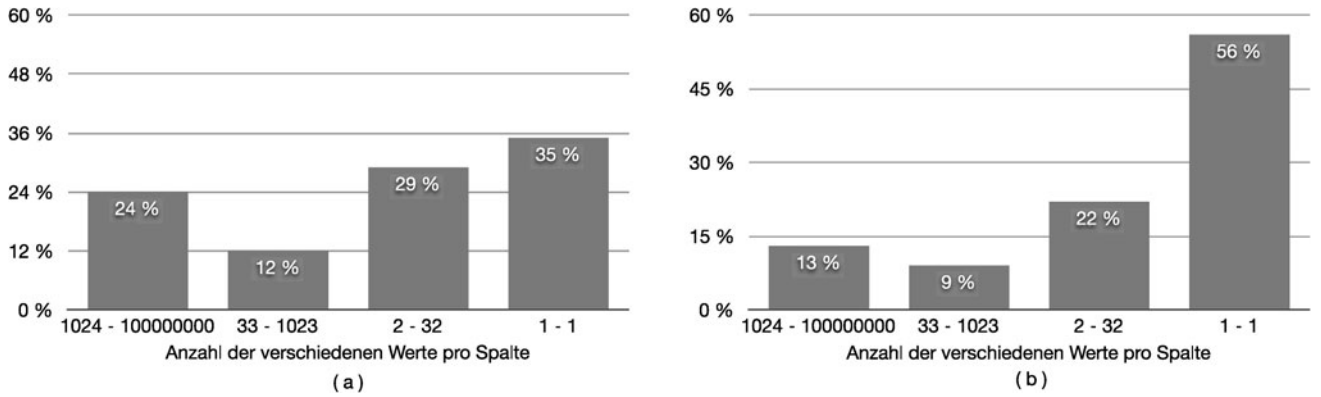

Abb. 4 Gruppierte Wertverteilung der Inventar- (**a**) und Finanztabellen (**b**)

Abb. 4 zeigt die prozentuale Häufigkeit gruppiert nach der Werteverteilung pro Spalte über alle untersuchten Tabellen der analysierten Kundensysteme. Es ist offensichtlich in der Abbildung, dass die meisten Attribute der Tabellen der letzten Gruppe zuzuordnen sind, welche lediglich einen Wert besitzen. Dieser Wert kann entweder ein Null- oder Standard-Wert sein. Folglich werden nicht alle Attribute von der Anwendung genutzt, was von großem Interesse für die Anwendungsoptimierung ist. Zu erwähnen ist, dass dies Durchschnittswerte aller betrachteten Kundensysteme sind und die Ausprägung der einzelnen Attribute abhängig ist vom Unternehmen und dem jeweiligen Industriezweig.

Wie in Abb. 4(a) zu sehen, haben lediglich 43 von 180 Attributen in der Tabelle der Materialbewegungen eine signifikante Anzahl an unterschiedlichen Werten. Folglich haben rund 75% aller Spalten eine relative geringe Werteverteilung. Das Attribut mit der höchsten Anzahl an unterschiedlichen Werten ist das Attribut „Bewegungsnummer“, welches gleichzeitig auch primärer Schlüssel der Tabelle ist. Weiterhin haben auch die „Transportanfragenummer“, „Anzahl“, „Datum“ und „Zeit“ eine hohe Werteverteilung. Bemerkenswert ist, dass 35% aller Spalten einzig einen Wert besitzen.

Folglich lassen sich sehr wichtige unternehmensspezifische Charakteristiken bestimmen. Abhängig von der Anwendung und dem jeweiligen Industriezweig, wird in vielen Spalten nur jeweils ein Wert hinterlegt. Diese Anwendungen profitieren daher signifikant von einer leichtgewichtigen Komprimierung. Hinzukommt dass vor allem Spalten für analytische Anfragen von Bedeutung sind, die eine gewisse Kardinalität von verschiedenen Werten aufweisen. Dieser Fakt spricht für die Verwendung von spaltenorientierten Datenbanken, in der nur die projizierten Attribute gelesen werden müssen.

3.3 Einzel- und Massendatenverarbeitung

Obwohl transaktionale Systeme auf einzelnen Instanzen von Objekten arbeiten, haben Analysen gezeigt, dass die Mehrzahl der genutzten Daten gemeinsam verarbeitet werden. Generell wird unterschieden zwischen dem verarbeiten einzelner Instanzen einer Geschäftsentität und dem gemeinsamen Verarbeiten von mehreren Instanzen. Zum Ersteren gehören bspw. ein Auftrag oder ein Kunde, während für das Bestimmen aller fälligen Rechnungen oder der Top 10 Kunden anhand des Umsatzes sehr viele Datensätze verarbeitet werden müssen.

Grundsätzlich basieren Unternehmensanwendungen auf einzelnen Geschäftsinstanzen, dennoch ist es nicht möglich den Zustand eines Unternehmens anhand einzelner Datensätze zu bestimmen. Um den Fortschritt eines Projektes zu erfahren, muss beispielsweise vorerst der Kontext bestimmt werden. Um diesen zu konstruieren, müssen verschiedene Attribute von verschiedenen Ereignissen gelesen und in der Regel aggregiert werden. Beispiele für solche Kontexte sind u.A. Dashboards oder Arbeitsvorratlisten. Dashboards werden genutzt, um den aktuellen Zustand eines Projektes oder einer anderen semantischen Einheit aufzuzeigen. Arbeitsvorratlisten werden benutzt, um Aufgaben anhand anderer Objekte – wie Fortschreibungen oder Rechnungen – zu erstellen. Weiterhin gibt es Geschäftsentitäten, deren Zustand nur abhängig von aktuellen Ereignissen statt vorberechneter Aggregationen bestimmt werden kann. Beispiele hierfür sind die Konten in der Finanzbuchhaltung oder der Bestand eines Materials.

Im Allgemeinen bedingt diese Art der Massendatenverarbeitung das Lesen von wenigen sequentiellen Attributen statt einzelnen Relationen. In Bezug auf die Nutzung von spaltenorientierten Hauptspeicherdatenbanken lässt sich sagen, dass diese umso mehr profitieren, je mehr Operationen zu dieser Kategorie zuzuordnen sind.

3.4 Applikationscharakteristiken

Unterschiedliche Geschäftsanwendungen haben unterschiedliche Anforderungen an die Datenverarbeitung. Tab. 1 zeigt die durchgeführte Analyse von verschiedenen Geschäftsanwendungen.

Tab. 1 Vergleich der analysierten Anwendungscharakteristiken

	Bedarfsplanung	Auftragsbearbeitung	Verfügbarkeits-prüfung	Mahnlauf	Verkaufsanalyse
Granularität der Daten	Transaktional	Transaktional	Transaktional	Transaktional	Voraggregiert
Operationen auf Daten	Lesen & Schreiben	Lesen & Schreiben	Lesen & Schreiben	Lesen & Schreiben	Einzig lesen
Vorberechnung von Daten	Nein	Nein	Nein	Nein	Ja
Zeitrahmen von Daten	Historisch & aktuell	Einzig aktuell	Historisch & aktuell	Historisch & aktuell	Historisch & aktuell
Updatezyklen	Immer aktuell	Immer aktuell	Immer aktuell	Immer aktuell	Zyklische Updates
Datenvolumen pro Anfrage	Groß	Small	Groß	Groß	Groß
Komplexität von Anfragen	Hoch	Standard	Hoch	Hoch	Hoch
Voraussehbarkeit von Anfragen	Mittel	Hoch	Mittel	Mittel	Niedrig
Antwortzeit von Anfragen	Sekunden	Sekunden	Sekunden	Sekunden bis Stunden	Sekunden bis Stunden

OLTP Charakteristiken in hellgrau OLAP Charakteristiken in dunkelgrau

Um die verschiedenen Anforderungen einer jeder Anwendung hervorzuheben wurden die transaktionalen Charakteristiken in hellgrau gezeichnet und die analytischen Charakteristiken in dunkelgrau. Es lässt sich auf dem ersten Blick erkennen, dass sich keine Anwendung klar als transaktional oder analytisch kategorisieren lässt. Obwohl bspw. das Bearbeiten von Aufträgen in sich selbst transaktional ist, sind jedoch sowohl die Verkaufsanalyse als auch das Mahnwesen mit ihren analytischen Funktionen davon abhängig. Die beiden zuletzt genannten Anwendungen arbeiten – sowohl lesend als auch schreibend – direkt auf transaktionalen Daten. Die lesenden Operationen sind dabei zumeist komplexe und analytische Anfragen auf großen Datenmengen [24].

Wie transaktionales und analytisches Verhalten miteinander interagieren, lässt sich anhand des Kontextes der jeweiligen Operationen bestimmen. Der Kontext wird hierbei als Metrik definiert. Diese wird bestimmt anhand der benötigten Daten, welche zur Ermöglichung einer Entscheidungsfindung in einem Geschäftsprozess gebraucht werden. Wenn der Kontext der Entscheidung relativ klein ist, ist auch das Volumen der benötigten Daten gering. Je größer somit der Kontext wird, desto größer sind die zu lesenden Datenvolumen.

3.5 Elastizität und Replikation

Cloud Computing bietet den Vorteil, dass Rechenkapazitäten automatisiert über einen Schnittstelle je nach Arbeitslast bezogen werden können. Bei Nutzung dieser Elastizität muss es möglich sein die Daten banklast nicht nur vertikal, sondern auch horizontal skalieren zu können. Dies bedeutet, dass Arbeitslast und Daten auf mehrere Server verteilt werden, welche weder CPU Leistung noch Hauptspeicher miteinander teilen. Wenn ein Datenbanksystem dies ermöglicht, können Hardwareressourcen automatisch durch Hinzufügen oder Entfernen skaliert werden. Hardwareressourcen müssen somit nicht mehr für die maximale Last, sondern nur noch für die tatsächliche Last zuzüglich eines Puffers dimensioniert werden. Mit Hilfe der Ausnutzung dieser Elastizität können Hardwareressourcen wesentlich effizienter genutzt werden.

Um vertikale Skalierbarkeit zu erreichen müssen Daten und Arbeitslast der spaltenorientierten Hauptspeicherdatenbank partitioniert werden. In der Regel wird hierbei eine horizontale Partitionierung der Daten durchgeführt, um diese auf unterschiedliche Systeme verteilen zu können. Um eine Verteilung der einzelnen Tupel zu ermöglichen, kann wie von Copeland et al. in [8] beschrieben eine Verteilung nach *Heat* erfolgen, d.h. wie oft einzelne Tupel verwendet werden. Eine Herausforderung hierbei ist es bei Verringern oder Vergrößern der Hardwareressourcen die vorhandenen Tupel auf neue Server zu verteilen oder auf weniger Server zu konsolidieren. Weiterhin muss es ebenfalls möglich sein, Joins von Daten durchzuführen, welche auf unterschiedlichen Servern gespeichert werden oder Transaktionen auf Daten auszuführen, die auf unterschiedlichen Systemen verteilt sind.

Um eine hohe Ausfallsicherheit gewährleisten zu können, ist es außerdem erforderlich, alle Daten auf mindestens zwei unterschiedliche Systeme zu replizieren, bzw. Änderungen persistent zu sichern. Das synchrone Schreiben in die Logdatei auf eine Festplatte eines einzelnen Servers ist wesentlich langsamer, als alle Datenänderungen synchron im

Hauptspeicher zu replizieren. Eine Transaktion, in der alle Änderungen synchron in eine Logdatei auf die Festplatte geschrieben werden, dauert mehr als 10.000 µs. Sofern Daten nur synchron in den Hauptspeicher auf zwei verschiedene Server übertragen werden liegt die erreichbare Transaktionszeit hingegen bei unter 600 µs. Dank der Verwendung von Hauptspeicher als primäre Speicherquelle können Transaktionen so wesentlich schneller als bei festplattenbasierten Systemen ausgeführt werden.

Weitere Implikationen ergeben sich, wenn spaltenorientierte Hauptspeicherdatenbanken direkt als Software-as-a-Service für verschiedene Kunden bereitgestellt werden bzw. indirekt als Platform-as-a-Service Dienste im selben Rahmen zur Verfügung stellen. Um Kosten zu sparen werden in der Regel anbieterseitig verschiedene Kunden bzw. Mandanten von einem Server und evtl. sogar vom selben Datenbankprozess bedient.

Wenn eine Hauptspeicherdatenbank als Software-as-a-Service angeboten wird, erscheint der vorhandene Arbeitsspeicher der limitierender Faktor zu sein, welcher die Anzahl der möglichen Kunden begrenzt, da alle Daten komplett im Hauptspeicher vorgehalten werden müssen. Dank der zum Einsatz kommenden Kompression wird jedoch je nach Arbeitslast und Anzahl der Anfragen eines Kunden, die Anzahl der maximal möglichen Kunden auch durch die vorhandene CPU Leistung begrenzt. Sofern einzelne Server mehrere Mandanten bedienen, kann die anfallende Arbeitslast von diversen Kunden unterschiedlich verteilt werden. Kossmann beschreibt in [19], dass die Arbeitslast entweder *nach Anfrage* oder *nach Last* partitioniert werden kann.

Bei der Verteilung *nach Anfrage* wird die Partitionierung der Daten nach Mandanten vorgenommen, d.h. die Daten eines Mandanten sind auf dem selben Server zu finden. Alle Anfragen zu Daten eines Mandanten werden dann zu diesem Server weitergeleitet. Bei diesem Szenario ist das gleichmäßig Verteilen von Mandanten jedoch eine Herausforderung, da die Arbeitslast von einzelnen Kunden stark schwanken kann und Migrationen von Mandanten erforderlich werden können.

Alternativ kann die Arbeitslast *nach Last* partioniert werden, d.h. Objekte von allen Kunden werden auf alle Server verteilt. Dies erlaubt eine wesentlich fein granularere Partitionierung von Daten, führt jedoch zu Synchronisationsproblemen. So sind bei dieser Variante verteilte Transaktionen nötig, welche vergleichsweise teuer sind. Andernfalls kann die Konsistenz nicht gewährleistet werden.

Im Enterprise Umfeld greifen Kunden im Gegensatz zu Webanwendungen, wie sozialen Netzwerken, in der Regel nur auf Ihre eigenen Daten zu. Hierbei muss meist die Konsistenz der Daten gewährleistet sein. Die Partitionierung der Arbeitslast nach Mandanten bzw. Anfrage ist daher unserer Meinung nach für Enterprise Anwendungen vorzuziehen.

3.6 Gewonnene Erkenntnisse

Während die in der Datenanalyse betrachteten Applikationen an sich jeweils als eigenständige Softwareanwendung betrachtet werden können, ist es die Hauptaufgabe komplexer Geschäftssysteme diese Applikationen zu vereinen und an dieselbe Datenquelle anzubinden.

Transaktionale Anwendungen arbeiten typischerweise auf atomaren Geschäftsentitäten und benötigen in der Regel Datenbanken, welche für transaktionale Arbeitslasten optimiert sind. Hingegen haben analytische Anwendungen andere Anforderungen an die Datenbank. Anwendungen dieses Bereiches folgen klaren Mustern und decken zumeist klar getrennte Geschäftsbedürfnisse wie Datenreinigung, Datenkonsolidierung usw. ab. Typische Anfragen dieser Anwendungen selektieren zumeist alle vorhandenen Daten und reduzieren diese schrittweise, normalerweise einem Navigationspfad oder einer Drill-Down-Strategie folgend, d.h. zunächst werden Aggregationen über alle Daten gebildet und dann Restriktionen hinzugefügt, wie bspw. nur ein bestimmtes Produkt und eine bestimmte Region ausgewählt.

Mit Hilfe von spaltenorientierten Hauptspeicherdatenbanken besteht nun jedoch die Möglichkeit, Datenbanksysteme für transaktionale und analytische Workloads wieder zu vereinen. Spaltenorientierte Hauptspeicherdatenbanken eigenen sich ideal für das Bilden von Aggregationen auf einzelnen Spalten. Aufgrund der Kompressionsverfahren ist das Einfügen, Löschen, und Aktualisieren von Daten allerdings komplexer. Es konnte jedoch gezeigt werden, dass in den beschriebenen Unternehmensanwendungen auch bei transaktionalen Anwendungen diese Operationen nur einen kleinen Teil der Gesamtlast ausmachen. Aufgrund der Tatsache, dass die meisten Spalten nur eine geringe Werteverteilung aufweisen, ist die Verwendung von spaltenbasierter Komprimierung zudem ideal geeignet, um den Speicherbedarf und die Zeit zum Lesen der Daten zu verringern. Auch bei transaktionalen Workloads werden in der Regel mehrere Datensätze zur gleichen Zeit gelesen oder bearbeitet. Der Vorteil von zeilenbasierten Datenbanksystemen beim Rekonstruieren einzelner Tupel ist in solchen Szenarien nur von geringerer Bedeutung und kann durch den hauptspeicherbasierten Ansatz ausgeglichen werden.

Durch Ausnutzen der Elastizität im Cloud Computing Konzept kann moderne Hardware effizienter genutzt werden. Der Wegfall von Festplatten als Ablage sorgt dafür, dass die Ressourcen, die in den letzten Jahren am meisten Fortschritt zeigten, primär zum Einsatz kommen und somit die ungleiche Verteilung der Ressourcen innerhalb eines Servers ausgeglichen wird. Der Flaschenhals wird verlagert auf die Speicherbandbreite sowie die CPU, wobei ein wesentlich höherer Durchsatz im Vergleich zur Festplatte erzielt werden kann, was Konzepte, wie mehrere Kunden auf

einem Server interessant macht. Für Elastizität über Servergrenzen hinweg muss die spaltenorientierte Hauptspeicherdatenbank eine horizontale Partionierung von Tabellen ermöglichen um vertikal skalieren zu können.

4 Vorteile spaltenorientierter Datenbankbanken für Unternehmensanwendungen

Im folgenden Abschnitt werden die Vorteile der vorgeschlagenen Datenverwaltung im Umfeld von Unternehmensanwendungen aufgezeigt.

4.1 Technische Vorteile für Geschäftsanwendungen

Das Durchführen von Analysen auf transaktionalen Daten erlaubt nicht nur neue Funktionen im Bereich Business Intelligence. So gibt es bspw. neue Möglichkeiten für die Implementierung Ereignis-gesteuerter Systeme, eine verbesserte Vernetzung von eigenständigen Unternehmensapplikationen und Kosteneinsparungen durch die Vereinfachung von Unternehmenslandschaften. Insbesondere der Aspekt der Vernetzung ist entscheidend für Unternehmensanwendungen, da die Wertschöpfung in der Integration der einzelnen Prozesse entsteht. Die vorgeschlagene Datenverwaltung verwirklicht das Ziel der „single source of truth“ und fördert somit genau diese Integration. Folgend können innerhalb transaktionaler Anwendungen analytische Anfragen unter Berücksichtigung aktuellster Daten gestellt werden. Damit ist z.B. eine Echtzeiterkennung von Kreditkartenbetrug oder die Abfrage von aktuellsten Umsätzen umsetzbar. Solche Funktionen werden momentan mit Hilfe von Operational Data Stores (ODS) bzw. echtzeitoptimierten Data Warehouses (DW) umgesetzt, wobei bei ersterem nur zeitlich eingeschränkte Datenbestände existieren bzw. bei letzterem Anfragen nur verzögert den aktuellsten Stand ermitteln können.

Mit der Vereinigung von analytischen Anfragen auf feinst granularen Daten in Echtzeit werden bspw. *Event-Driven Enterprise Systems* möglich. In der konventionellen Datenverwaltung werden analytische Ereignisse zeitversetzt aus einem Data Warehouse bezogen. Bei analytischen Ereignissen handelt es sich nicht um Ereignisse, welche lediglich einzelnen Geschäftsentitäten zugrunde liegen, sondern durch eine Vielzahl von verschiedenen Faktoren beeinflusst werden und nur mittels OLAP-ähnlicher Anfragen erfassbar sind. Diese sind in der vereinheitlichten Datenverwaltung jederzeit direkt im Quellsystem erkennbar und werden sofort an die jeweilige Komponente propagiert, ohne dass diese auf bestimmte Informationen im Data Warehouse eingeschränkt ist. Dies ermöglicht eine weitaus genauere *Event-Driven Process Optimization* und erlaubt es Unternehmen laufende Prozesse in Echtzeit auf bestimmte Ereignisse reagieren zu lassen und gegebenenfalls entsprechend zu reagieren. Hinzukommt, dass die Entscheidungen mit Hilfe aller erfassten Informationen getroffen werden können. Bisherige Systeme benötigen eine vorherige Definition der Extraktion- bzw. Materialisierungsstrategie und verbieten somit eine dynamische Anpassung an unvorhergesehene Events.

Neben der horizontalen Verknüpfung von Geschäftsapplikationen ergeben sich zusätzlich in der vertikalen Verknüpfung neue Möglichkeiten. Die vertikale Verknüpfung beschreibt hierbei die Verbindung von analytischen mit transaktionalen Systemen. Abhängig von den Anforderungen an die Aktualität der Daten benötigen heutige Systemlandschaften eine Integration auf mehreren Ebenen (Transaktionales System als Empfänger und Sender, ODS als Integrator, DW als Sender und Empfänger). Diese vertikalen Verbindungen werden in der Regel unidirektional mit Hilfe von ODS implementiert. Spaltenorientierte Hauptspeicherdatenbanken ermöglichen hingegen von vornhinein eine bidirektionale Verbindung, wodurch so genannte Closed-Loop Funktionen um ein Vielfaches leichter zu implementieren sind. Es somit nun die Möglichkeit jegliche Geschäftsdaten in einer Quelle zu halten und folglich sowohl eine einheitliche Modellierung der Daten als auch sofort nutzbare Vernetzungen – sowohl horizontal als auch vertikal – zwischen allen Komponenten zu bieten.

Nicht zuletzt bringt die Verwendung hochperformanter Systeme Kosteneinsparpotentiale. Durch die Vermeidung von Redundanz gibt es keine komplexen Systemlandschaften mit verschiedenen Teilsystemen mehr, welche die gleichen Geschäftsdaten in verschiedenen Formen (einzelne Geschäftsentitäten vs. voraggregierte Summen und Kennzahlen) redundant vorhalten. Neben der Reduktion in ein einzelnes System erlauben die analytischen Funktionen die Abschaffung der Redundanzen innerhalb eines Systems. Die Abb. 5 zeigt das Schema einer heutigen Finananzbuchhaltung. Klar erkennbar ist die Menge an redundanten Daten, die um Faktoren die Basistabellen übersteigen. Neben der reinen Datenmengenreduktion, die messbar ist, bedeutet ein System, welches alle Anforderungen nur mit Hilfe der Basistabellen realisiert weniger Aufwand bei der Konsistenthaltung redundanter Daten bei gleichzeitiger Erhöhung der Flexibilität. Dies wiederum heißt auch eine schnellere und leichtere Entwicklung und Pflege des Systems im Allgemeinen.

4.2 Vorteile aus Anwendersicht

Eine interessante Frage zu den im vorigen Abschnitt genannten technologischen Möglichkeiten ist es, inwiefern Arbeitsprozesse von bereits bestehenden bzw. nun möglichen Analysetechniken profitieren können. Durch die Integrati-

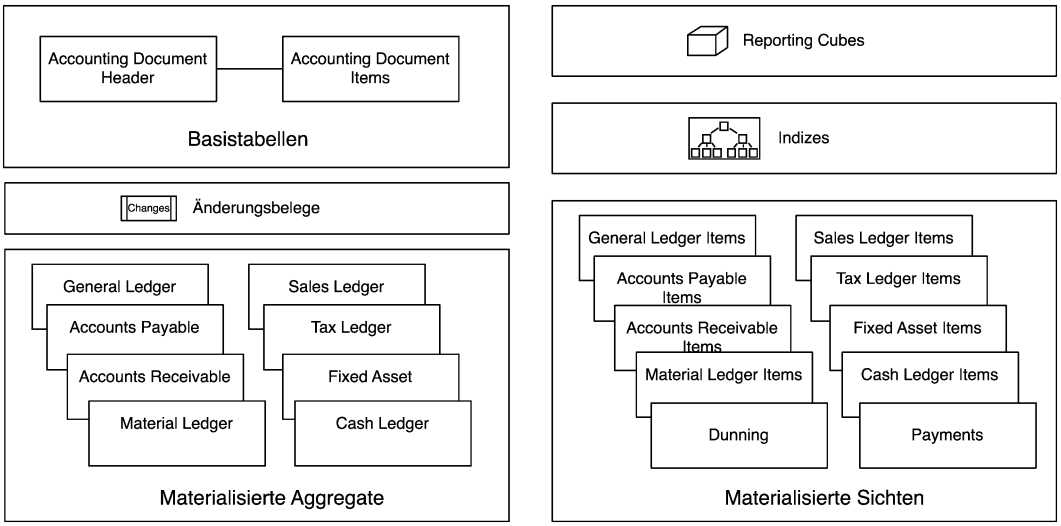

Abb. 5 Schema einer Finanzbuchhaltung

on von Analysemechanismen in das Produktivsystem wird auch Mitarbeitern unterhalb der Managementebenen Zugriff auf Analysen gegeben. Somit wird es den Mitarbeitern erlaubt selbstständiger zu arbeiten und mehr Verantwortung zu übernehmen. Ein Mitarbeiter kann durch bessere Einsicht eigenständig präzisere Entscheidungen treffen als dies zuvor möglich war. Beispiele sind Vorschläge oder Fehlerkorrekturverfahren durch eine Sinnhaftigkeitsprüfung bei Formulareingaben oder gar weitaus komplexeren analytischen Funktionen, die Simulationen durch freie Paramterwahl erlauben. Weiterhin lassen sich zudem Geschäftsprozesse besser steuern. So können z.B. Verkaufszahlen oder Kundenbewertungen direkt die Produktion steuern.

Natürlich profitieren insbesondere auch Führungskräfte im mittleren bis oberen Management. Sie müssen in Unternehmen heute mehr denn je – besonders in Krisen wie in 2008 – in der Lage sein Entscheidungen schnell und akkurat zu fällen bzw. Trends zu erkennen. Hierbei ist es wichtig, dass es die Möglichkeit von Ad-hoc Vorhersagen, sowie auch des Eingriffes in laufende Prozesse gibt.

Systeme welche Analysen direkt auf transaktionalen Daten erlauben, ermöglichen einen Wandel in der Entscheidungsunterstützung. Momentan sind analytische Systeme nach wie vor größtenteils Push-Systeme. In solchen stellt die Unternehmensführung Fragen an die jeweilig zuständigen Personen in den IT-Abteilungen, welche dann mit teils tagelangen Verzögerungen die Antworten liefern. Das Problem hierbei ist, dass der Datenradius um die ursprünglich gestellte Frage relativ klein ist. Kommt es anhand der Antworten zu neuen weiterreichenden Fragen (so genannten follow-ups), welche nicht durch bereits modellierte Anfragen beantwortet werden können, müssen erneut zeitaufwändig neue erstellt werden. Mit spaltenorientierten Hauptspeicherdatenbanken ist es hingegen möglich, analytische Fragen direkt an das transaktionale System zu stellen. Es gibt keine zwischengelagerten Cleaning-, Integrations- und Umwandlungsprozesse und es muss lediglich eine Datenquelle angefragt werden. Dies bedeutet einen Wandel von Push- zu Pull-Systemen. Es wird der Unternehmensführung ermöglicht auf alle Daten und Informationen in Echtzeit direkt zuzugreifen ohne Gefahr zu laufen, dass diese nicht 100-prozentig dem aktuellsten Stand des Unternehmens entsprechen.

Somit ist eine 360°-Sicht auf das Unternehmen möglich. Es kann nun auch in zeitkritische Prozesse direkt eingegriffen werden, was vor allem in Krisensituationen, in denen es gilt zeitnah Entscheidungen zu fällen, von großem Vorteil sein kann. Zumal der erwartete Effekt der Entscheidungen wiederum mittels analytischer Anfragen überprüft werden möchte. Dies war bislang nicht möglich, da Berichte auf separaten und zeitverzögerten Daten basierten.

Die Analyse auf transaktionalen Daten kombiniert mit der Möglichkeit Vorhersagen direkt auf Echtzeitdaten durchzuführen ermöglicht des Weiteren eine höhere Präzision der Entscheidungsunterstützung. Prognosen und Simulationen benötigen transaktionale Daten eines Unternehmens samt Ausreißern und Unregelmäßigkeiten – statt voraggregierten Ergebnissen – für eine möglichst genaue Vorhersage. Aktuelle Vorhersagemöglichkeiten sind in ihrer Durchführbarkeit stark beschränkt. Ziel sollte es jedoch sein, Voraussagen weder in Hinsicht auf die zugrundeliegenden Daten noch auf den Zeitpunkt oder die Häufigkeit der Durchführung einzuschränken. Mit nicht nur genaueren, sondern vor allem auch beliebig durchführbaren Vorhersagen und den zeitnahen Eingriff in laufende Prozesse wird der Unternehmensführung die Möglichkeit gegeben eine präzisere Einsicht in ihr Unternehmen zu erlangen und laufende Prozesse besser kontrollieren zu können.

4.3 Spaltenorientierte Datenbanken in SaaS Anwendungen

Um eine hohe Skalierbarkeit zu erreichen, werden für Anwendungen, die als Software-as-a-Service angeboten werden, oftmals die selben Tabellen für mehrere Kunden verwendet. Trotzdem ist es oftmals erforderlich, dass einzelnen Kunden die Möglichkeit gegeben wird, das Datenmodell für ihre jeweiligen Bedürfnisse anzupassen. Bei zeilenbasierten Datenbanken wird aus diesem Grund auf verschiedene, zumeist komplexe Techniken zurück gegriffen, wie zum Beispiel das Verwenden von virtuellen Tabellen (Pivot Table Layout [4]) oder dedizierten vordefinierten Spalten.

Aufgrund der zeilenbasierten Anordnung von Datensätzen in klassischen Datenbanksystemen ist es zudem aufwändig neue Spalten einzufügen, da im Extremfall der komplette Speicherbereich neu organisiert werden muss, was meist zu einer temporären Nichtverfügbarkeit von Tabellen führt, da Locks während des Prozesses jegliche Lese- und Schreiboperationen unterbinden. Gerade im Software-as-a-Service Bereich muss die Anwendung auch weiterhin bei Upgrades, welche Änderungen am Datenmodell mit sich ziehen können, verfügbar sein. Sofern Spalten in einer spaltenorientierten Datenbank hinzugefügt werden sollen, muss der komplette Speicherbereich hingegen nicht neu organisiert werden, da neue Spalten in einem komplett neuen Speicherbereich angelegt werden.

Wird eine neue Spalte angelegt muss lediglich ein Platzhalter (engl. Stub) erstellt werden und die Metainformationen der Tabelle angepasst werden, so dass auf die neuen Attribute sofort zugegriffen werden kann. Dies ist ein großer Vorteil für eine unterbrechungsfreie Entwicklung und Integration neuer Software. Zudem wird die neue Spalte erst materialisiert, nachdem wirklich neue Attribute eingefügt wurden und besteht vorher nur als Platzhalter und in den Metainformationen. Dies ermöglicht es, Änderungen am Datenmodell wesentlich schneller ausführen zu können.

Sofern eine Tabelle von verschiedenen Kunden verwendet wird, ist es so mit spaltenorientierten Datenbanken auch kein Problem mehr, weitere Spalten für einzelne Kunden hinzuzufügen.

5 Verwandte Arbeiten

Wie schon in Abschn. 2 erwähnt, behandelt der Großteil verwandter Arbeiten spezialisierte Datenbanken. Dieses Forschungsgebiet beinhaltet sowohl Hauptspeicherdatenbanken als auch spaltenorientierte Datenstrukturen. Ersteres wurde schon früh in [11] behandelt, während sich letzte Arbeiten zu Hauptspeicherdatenbanken eher auf spezielle Anwendungen fokussieren. Die Autoren von [7] und [32] beschreiben Hauptspeicherdatenbanken für transaktionale Workloads, nutzen dafür aber zeilenbasierte Datenstrukturen. Im Gegensatz befassen sich Boncz et al. in [5] mit Hauptspeicherverarbeitung und binären Assoziationstabellen, welche die Daten komplett vertikal partitioniert speichern und für analytische Workloads optimiert sind.

Die Idee von spaltenorientierten Datenbanken wurde mittlerweile in vielen akademischen Projekt implementiert, bspw. die Hauptspeicher-basierte MonetDB [5] oder das Festplatten-basierte C-Store [31]. Ramamurthy et al. beschreiben in [29] ein Datenbankdesign für kombinierte Workloads, in welchem verschiedene physikalische Datenstrukturen genutzt werden, ähnlich wie auch in [30] beschrieben.

Ein anderes wichtiges Forschungsthema im Bereich der Unternehmensanwendungen behandelt die Möglichkeiten von *Real-Time Reporting* und *Operational Reporting*. Letzteres ist eine typische Anwendung aus dem Bereich des *Active Warehousing* [17], welches vor allem taktische Entscheidungsunterstützung bietet. Taktische Entscheidungen benötigen aber normalerweise höchst aktuelle Information. Um diese besser als in Active Warehouses bereitzustellen, wurden Operational Data Stores (ODS) entwickelt. Im Gegensatz zu Data Warehouses, welche in der Regel in Intervallen aktualisiert werden, werden ODS direkt zeitgleich mit der eigentlichen Transaktion aktualisiert. Sie sind daher für Anfragen auf verhältnismäßig kleinen, jedoch höchst aktuellen Datenmenge ausgelegt. Inmon [16] beschreibt ODS als Hybrid mit Charakteristiken von sowohl transaktionalen Systemen als auch von Data Warehouses. ODS bieten OLTP-Antwortzeiten auf aktuellen Daten mit Entscheidungsunterstützungsfunktionen. Im Gegensatz zu Data Warehouses wird jedoch nur ein kleiner Datensatz persistiert.

Mit spaltenorientierten Hauptspeicherdatenbanken ist es möglich direkt auf Geschäftsentitäten statt auf vorberechneten Aggregationen zu arbeiten. Des Weiteren haben Grund et al. [14] ein Konzept gezeigt welches Insert-Only nutzt, um Reporting auf historischen Daten zu ermöglichen. Materialisierungstechniken wie bspw. in [18] und [36] nutzen Redundanz, um sowohl transaktionale als auch analytische Workloads abzudecken. Jegliche Redundanz erschwert jedoch die Pflege und Implementierung eines Systems und bietet weniger Flexibilität als bspw. redundanzfreie, spaltenorientierte Hauptspeicherdatenbanken, bei denen jederzeit alle Daten statt nur Teilmengen oder vorberechnetete Aggregationen zur Verfügung stehen.

Im Rahmen des Cloud Computings gibt es verschiedene Service Anbieter, welche Cloud Services offerieren und hierbei versprechen Kosten zu reduzieren und eine nahezu unbegrenzte Skalierbarkeit zu ermöglichen. Kossmann et al. [20] haben verschiedene Architekturen unter Verwendung von OLTP-Workloads untersucht und stellen fest, dass diese je nach Workload signifikant bezüglich Kosten und Geschwindigkeit schwanken. Je nach erwarteter Last sollten sich Unternehmen folglich den passenden Anbieter auswählen.

6 Ausblick

Dieser Artikel hat einen neuen Ansatz des Datenmanagements in Unternehmensanwendungen präsentiert. Durch die Analyse von ERP-Systemen und vor allem der Analysen von produktiven Systemen konnten Charakteristiken von Unternehmensanwendungen bestimmt werden. Die wichtigste Erkenntnis bei der Datenanalyse war die geringe Verteilung der Daten, d.h. die relativ geringe Zahl an unterschiedlichen Attributswerten. Interessant ist auch, dass die meisten Attribute in ERP Systemen kaum genutzt werden.

Diese Charakteristiken haben signifikante Vorteile für spaltenorientierte Hauptspeicherdatenbanken. Anhand dieser Eigenschaften ist es möglich Unternehmensanwendungen zu schreiben, welche mit konventionellen Datenbanken nicht möglich sind und dies mit klar vereinfachten Datenschemata. Neu entstandene Möglichkeiten, wie bspw. *Operational Reporting* auf transaktionalen Daten, ermöglichen eine genauere, fundiertere und schnellere Entscheidungsunterstützung, da keine zwischengelagerten Transformationen für analytische Systeme mehr durchgeführt werden müssen und stattdessen stets aktuelle transaktionale Daten genutzt werden.

Vorausschauend gibt es weitere interessante und vielversprechende Anwendungsgebiete für spaltenorientierte Hauptspeicherdatenbanken im Unternehmensumfeld. Es ist geplant die gewonnenen Erkenntnisse in Hinblick auf Multi-Tenancy und Cloud Computing anzuwenden. Cloud-basierte Hauptspeicherdatenbanken besitzen große Vorteile bei der Sklalierbarkeit von Multi-Tenancy-Systemen und haben folglich großes Potential für Kostenoptimierungen, welche gerade im Cloud Computing Umfeld ein kritischer Faktor sind.

Literatur

1. Abadi DJ, Madden S, Ferreira M (2006) Integrating compression and execution in column-oriented database systems. In: SIGMOD
2. Abadi DJ, Myers DS, DeWitt DJ, Madden S (2007) Materialization strategies in a column-oriented DBMS. In: ICDE. IEEE Press, New York, pp 466–475
3. Ailamaki A, DeWitt DJ, Hill MD, Wood DA (1999) DBMSs on a modern processor: where does time go? In: VLDB
4. Aulbach S, Grust T, Jacobs D, Kemper A, Rittinger J (2008) Multitenant databases for software as a service: schema-mapping techniques. In: SIGMOD
5. Boncz PA, Manegold S, Kersten ML (1999) Database architecture optimized for the new bottleneck: memory access. In: VLDB
6. Boncz PA, Zukowski M, Nes N (2005) MonetDB/X100: hyper-pipelining query execution. In: CIDR
7. Cha SK, Song C (2004) P*TIME: highly scalable OLTP DBMS for managing update-intensive stream workload. In: Proceedings of 30 rd international conference on very large data bases (VLDB 2004)
8. Copeland G, Alexander W, Boughter E, Keller T (1988) Data placement in Bubba. In: SIGMOD
9. Copeland GP, Khoshafian S (1985) A decomposition storage model. In: SIGMOD
10. Cormack GV (1985) Data compression on a database system. Commun ACM 28(12):1336–1342
11. DeWitt DJ, Katz RH, Olken F, Shapiro LD, Stonebraker M, Wood DA (1984) Implementation techniques for main memory database systems. In: Yormark B (ed) SIGMOD conference. ACM Press, New York, pp 1–8
12. French CD (1995) "One size fits all" database architectures do not work for DDS. In: SIGMOD
13. Garcia-Molina H, Salem K (1992) Main memory database systems: an overview. IEEE Trans Knowl Data Eng 4(6):509–516
14. Grund M, Krueger J, Tinnefeld C, Zeier A (2009) Vertical partition for insert-only scenarios in enterprise applications. In: IE&EM
15. Grund M, Krüger J, Plattner H, Zeier A, Cudre-Mauroux P, Madden S (2011) Hyrise—a hybrid main memory storage engine. In: PVLDB (in Druck)
16. Inmon WH (1999) Building the operational data store. Wiley, New York
17. Karakasidis A, Vassiliadis P, Pitoura E (2005) Etl queues for active data warehousing. In: Berti-Equille L, Batini C, Srivastava D (eds) IQIS. ACM Press, New York, pp 28–39
18. Kiviniemi J, Wolski A, Pesonen A, Arminen J (1999) Lazy aggregates for real-time OLAP. In: Mohania MK, Tjoa AM (eds) DaWaK. Lecture notes in computer science, vol 1676. Springer, Berlin, pp 165–172
19. Kossmann D (2010) How new is the cloud? In: ICDE, p 3
20. Kossmann D, Kraska T, Loesing S (2010) An evaluation of alternative architectures for transaction processing in the cloud. In: Elmagarmid AK, Agrawal D (eds) SIGMOD conference. ACM Press, New York, pp 579–590
21. Krueger J, Grund M, Boissier M, Zeier A, Plattner H (2010) Data structures for mixed workloads in in-memory databases
22. Krueger J, Grund M, Tinnefeld C, Plattner H, Zeier A, Faerber F (2010) Optimizing write performance for read optimized databases. In: DASFAA
23. Krueger J, Grund M, Zeier A, Plattner H (2010) Enterprise application-specific data management. In: EDOC, Brazil (in Druck)
24. Krueger J, Tinnefeld C, Grund M, Zeier A, Plattner H (2010) In: DBTest
25. Mahapatra NR, Venkatrao B (1999) The processor-memory bottleneck: problems and solutions. Crossroads, p. 2. doi:http://doi.acm.org/10.1145/357783.331677
26. Manegold S, Boncz PA, Kersten ML (2002) Generic database cost models for hierarchical memory systems. In: VLDB
27. Ousterhout J, Agrawal P, Erickson D, Kozyrakis C, Leverich J, Mazières D, Mitra S, Narayanan A, Parulkar G, Rosenblum M, Rumble SM, Stratmann E, Stutsman R (2010) The case for RAMClouds: scalable high-performance storage entirely in DRAM. SIGOPS Oper Syst Rev 43(4)
28. Plattner H (2009) A common database approach for OLTP and OLAP using an in-memory column database. In: Çetintemel U, Zdonik SB, Kossmann D, Tatbul N (eds) SIGMOD conference. ACM Press, New York, pp 1–2
29. Ramamurthy R, DeWitt DJ, Su Q (2003) A case for fractured mirrors. VLDB 12(2):89–101
30. Schaffner J, Bog A, Krueger J, Zeier A (2008) A hybrid row-column oltp database architecture for operational reporting. In: BIRTE (informal proceedings)
31. Stonebraker M, Abadi DJ, Batkin A, Chen X, Cherniack M, Ferreira M, Lau E, Lin A, Madden S, O'Neil EJ, O'Neil PE, Rasin A, Tran N, Zdonik SB (2005) C-store: a column-oriented DBMS. In: VLDB
32. Stonebraker M, Madden S, Abadi DJ, Harizopoulos S, Hachem N, Helland P (2007) The end of an architectural era (it's time for

a complete rewrite). In: VLDB. ACM Press, New York, pp 1150–1160
33. TPC-C Benchmark—standard specification—revision 5.11. http://www.tpc.org/tpcc/spec/tpcc_current.pdf
34. Transier F, Sanders P (2008) Compressed inverted indexes for in-memory search engines. In: Proceedings of the 9th workshop on algorithm engineering and experiments
35. Westmann T, Kossmann D, Helmer S, Moerkotte G (2000) The implementation and performance of compressed databases. SIGMOD Rec 29(3), 55–67
36. Yan WP, Åke Larson P (1995) Eager aggregation and lazy aggregation. In: Dayal U, Gray PMD, Nishio S (eds) VLDB. Morgan Kaufmann, San Mateo, pp 345–357