Hochschule für Technik, Wirtschaft und Kultur Leipzig

Abstract

Oberseminar „Datenbanksysteme – Aktuelle Trends“

Hauptspeicherdatenbanken

Eingereicht von: Maximilian Staab
INM-17
Matr.-Nr.: 70381

Leipzig, 04.07.2018

Inhaltsverzeichnis

1 Einleitung

Die Anforderungen an ein modernes Datenbank-Management-System sind groß und wachsen immer weiter. Es gibt in Unternehmen heutzutage riesige Datenbestände mit verschiedensten Quellen, welche kombiniert und effizient verarbeitet werden müssen. Die Daten sollten dabei möglichst in Echtzeit analysiert werden können, um Entscheidungsträgern im Unternehmen Informationen und Erkenntnisse zu liefern, was aktuell passiert im Unternehmen. Somit kann direkt auf Veränderungen reagiert werden.

Voraussetzung ist hierbei, dass auch komplexe Analysen und Berechnungen nicht wie bisher Minuten oder sogar Stunden dauern, sondern nur noch wenige Sekunden. Zum Einsatz kommt sowas zum Beispiel bei der automatischen Erstellung von Mahnungen oder beim Available-to-promise Check, welcher eingesetzt wird um einen aktuellen Stand über die verfügbare Ware anzeigen zu können.

Um diese Anforderungen zu erfüllen, muss die Zugriffsgeschwindigkeit auf die Daten als großer Leistungsfaktor verbessert werden. Diese war bisher bei herkömmlichen Datenbankmanagementsystemen durch die Verwendung von Festplattenlaufwerken begrenzt. Deshalb wird inzwischen oft der Arbeitsspeicher eingesetzt und wie das Ganze funktioniert soll in diesem Abstract noch einmal genauer erläutert werden.

2 OLTP- und OLAP-Anwendungen

2.1 OLTP-Anwendungen

OLTP steht für Online Transaction Processing und bedeutet Echtzeit-Transaktionsverarbeitung. Eine Transaktion ist dabei eine Folge von Programmschritten zusammgefasst zu einer Transaktion. OLTP-Datenbanken speichern in diesem Fall immer den aktuellen Datenbestand.

Transaktionen werden nach einem bestimmten Muster verarbeitet. Dabei findet zuerst der Zusammenschluss von einzelnen Anweisungen zu Transaktionen zusammen. Im nächsten Schritt werden diese nacheinander ausgeführt. Hierbei ist ein erfolgreicher Abschluss erforderlich, da sonst ein Rollback eintritt und die bereits ausgeführten Änderungen wieder rückgängig macht um den Startzustand wiederherzustellen.

Transaktionen müssen verschiedene Kriterien erfüllen, sodass eine konsistente Datenbank sichergestellt wird. Das erste Kriterium ist dabei die Konsistenzerhaltung, wodurch der eben genannte konsistente Zustand nach der Ausführung erforderlich ist, ansonsten tritt das vorher angesprochene Rollback ein. Das zweite Kriterium ist die Isolation, dadurch soll bewirkt werden, dass nebenläufige Transaktionen sich nicht gegenseitig beeinflussen und somit Daten gesperrt werden während der Bearbeitung. Als letztes Kriterium wird die Dauerhaftigkeit von Daten angestrebt. Dies wird erreicht indem nach einer erfolgreichen Transaktion die Daten dauerthaft in einer Datenbank gespeichert werden.

OLTP-Anwendungen bieten somit verschiedene Vorteile. So ist zum Beispiel ein Vorteil die schnelle und direkte Verarbeitung von Transaktionen. Ebenso besteht durch die Kriterien eine Transaktionssicherheit bei parallelen Zugriffen. Im Endeffekt lässt sich sagen das bei OLTP-Anwendungen ein hoher Datendurchsatz möglich ist. [1, S. 11]

2.2 OLAP-Anwendungen

OLAP steht für Online Analytic Processing. Hierbei steht im Gegensatz zu OLTP die komplexe Analyse von Daten im Vordergrund, um ein Ergebnis zu liefern. Dabei kommt es fast nur zu Lesezugriffen auf die vorhandenen Daten zur Erzeugung des Ergebnisses. Dieses Ergebnis soll dem Analysten helfen bestimmte Entscheidungen zu treffen und seine Annahme entweder bestätigen oder wiederlegen, deshalb wird es zum Beispiel bei der Planung und Budgetierung von Finanzabteilungen eingesetzt.

Früher war es üblich OLTP- und OLAP-Anwendungen voneinander zu trennen und auf verschiedenen Systemen laufen zu lassen. Durch die Einführung von In-Memory-Datenbanken und deren geringer Antwortzeit können diese beiden Anwendungen in einem System vereint werden. [2]

3 DBMS-Architektur am Beispiel von SAP HANA

SAP verfolgt schon seit einiger Zeit den Ansatz zum Einsatz von Hauptspeicherdatenbanken. Zuerst unter dem Namen SAP SanssouciDB, seit 2010 unter dem Namen SAP HANA. HANA bedeutet hierbei „high performance analytic appliance“. Es ist als In-Memory-Plattform designt und ermöglicht somit, dass Daten die ganze Zeit im Arbeitsspeicher gehalten werden. Dies hat zur Folge das OLTP und OLAP-Anwendungen auf dem gleichen System arbeiten können, wodurch ebenfalls eine Analysemöglichkeit für Big Data entsteht. Trotzdem kann nicht komplett auf den Einsatz von Festplatten verzichtet werden. Auf den Festplatten sollen geänderte Daten dauerhaft gespeichert werden. Bei dem System müssen die Daten ebenfalls von der Festplatte ausgelesen werden, sodass der Arbeitsspeicher darauf zugreifen kann. SAP HANA greift bei der Arbeit auf das Prinzip von spaltenorientierten Datenbanksystem zurück. Der Vorteil hierbei ist, dass gleiche Einträge wirklich nur einmal gespeichert werden um Speicherplatz zu sparen. Durch den Einsatz von Mehrkernprozessoren besteht die Möglichkeit zur Parallelverarbeitung auf unterschiedlichen Datensets. [3]

HANA kommt inzwischen teilweise zum Einsatz in der Hauptanwendung von SAP, der SAP Business Suite, um dort spezielle Analyseaufgaben zu übernehmen. Ebenso wurde das System SAP S/4 HANA entwickelt, welches der Business Suite

entspricht von der Anwendungsmöglichkeit, jedoch komplett auf die Hauptspeicherdatenbank HANA umgebaut ist. Ebenso kann HANA in der SAP Cloud Platform zum Einsatz kommen und dort die Basis bilden auf welcher andere Anwendungen ausgeführt werden. [4]

SAP HANA wird zum Beispiel von Mercedes AMG eingesetzt um während eines Testlaufes die Daten der Motorsensoren schnell auszuwerten, somit besteht die Möglichkeit einen Testlauf schnell abzubrechen, falls unerwartete Ergebnisse entstehen. Lenovo ist ein anderes Unternehmen welches mit SAP HANA arbeitet, hierbei ist die Hauptaufgabe verschiedene Berichte zu erstellen, welche Daten aus vielen unterschiedlichen Systemen miteinander vereinen. [5]

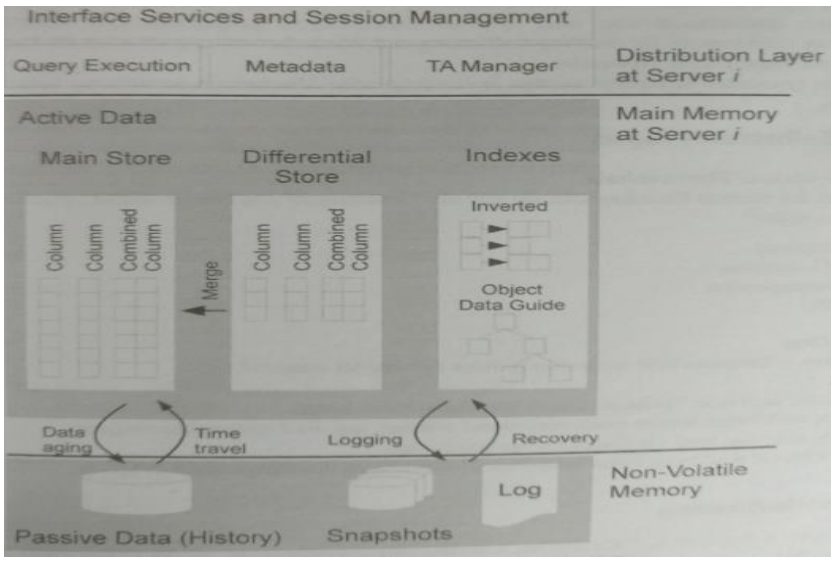

Abbildung 1: Überblick SAP HANA Architektur

Wie in Abbildung 1 zu sehen ist teilt sich die HANA Architektur in 3 Schichten ein. In der obersten Schicht, der Distributionsschicht, befindet sich die Kommunikation zwischen Anwendungen, die Erstellung von Plänen zur Abfrageausführung, die Speicherung von Metadaten und die Logik für die Ausführung von Datenbank-Transaktionen. In der zweiten Schicht, dem Main Store, werden die Daten in Abhängigkeit des Workloads gespeichert. Dabei werden Leseoperationen auf dem Hauptspeicher und dem Differenziellem Speicher ausgeführt. Schreiboperationen hingegen nur auf dem Differenziellen Speicher, diese Änderungen werden in bestimmten Abständen in den Hauptspeicher übertragen (Merge). Dadurch soll verhindert werden, dass sich im Hauptspeicher fehlerhafte Transaktionen befinden, da diese beim Merge des differenziellen Speichers in den Hauptspeicher gelöscht

werden. Um ein korrektes Ergebnis zu bekommen muss somit eine Abfrage immer auf beiden Speichern ausgeführt werden.

Die unterste Schicht ist der nichtflüchtige Speicher. In dieser Schicht befinden sich die Protokollierung von Transaktionen, Wiederherstellungsmöglichkeiten und eine Datenbank zum Speichern von dauerhaften Änderungen. [6, S.31-34]

4 Basisoperationen

4.1 Wörterbuchkodierung

Die Wörterbuchkodierung ist wichtiger Bestandteil für die nachfolgenden Basisoperationen und wird deswegen etwas genauer erläutert. Ziel der Wörterbuchkodierung ist es lange Texte in ganzen Zahlen zu speichern, um somit eine Verbesserung der Performance zu erreichen. Dabei arbeitet jedes Wörterbuch auf einer einzelnen Tabellenspalte, welche sortiert sein muss.

Innerhalb des Wörterbuchs werden die Werte einzeln gespeichert, in nachfolgendem Beispiel für die Spalte „Nation“ der gegebenen Tabelle. Jedem Wert wird dann eine eindeutige WertID zugeordnet. Daraus wird im Anschluss ein Attribut-Vektor erstellt, welcher die WertID des Wörterbuchs und die Position, an welcher Stelle in der Tabelle der Wert steht, beinhaltet. [6, S.37-28]

Tabelle „Spieler“

ID	Vorname	Nachname	Nation
1	Manuel	Neuer	Deutschland
2	Lionel	Messi	Argentinien
3	Timo	Werner	Deutschland

Wörterbuch für „Nation“

wertID	Nation
10	Argentinien
20	Deutschland

Attribut-Vektor für „Nation“

position	wertID
1	20
2	10
3	20

Abbildung 2: Beispiel Wörterbuchkodierung

4.2 Insert

Beim Insert in die Datenbank muss jeder Spalte ein neuer Wert hinzugefügt werden. Dadurch, dass für jede Spalte noch ein Wörterbuch und ein Attribut-Vektor vorhanden sind entstehen beim Einfügen von Daten 3 Szenarien.

Bei diesen Szenarien geht es darum ob ein Wert bereits im Wörterbuch vorhanden ist oder nicht. Deshalb wird zumindest zu Beginn identisch vorgegangen, indem das Wörterbuch durchsucht wird ob der aktuelle Wert schon vorhanden ist oder nicht. Falls der Wert bereits vorhanden ist, dann wird lediglich der Attribut-Vektor um die neue Position mit der aktuellen WertID erweitert. Dies ist das erste Szenario

Szenario zwei und drei befassen sich damit, falls der Wert noch nicht im Wörterbuch steht. Der Unterscheid besteht darin ob durch den neuen Wert das Wörterbuch neu sortiert werden muss oder nicht. Falls keine Neusortierung notwendig ist, dann wird der Wert ans Ende des Wörterbuchs geschrieben und dieser bekommt eine neue WertID. Wenn jedoch durch den neuen Wert eine Neusortierung notwendig ist, dann wird der Wert an die entsprechende Stelle im Wörterbuch geschrieben und er übernimmt die WertID des Wertes, welcher sich vorher an dieser Stelle befand. Dadurch verschieben sich alle Werte, welche hinter dem neu eingefügten Wert jetzt stehen und bekommen neue WertIDs zugeordnet. Gleiches passiert dann im Attribut-Vektor.

Das ganze Verfahren wird somit vor allem beim letzten Szenario sehr schreibintensiv, da viele Werte angepasst werden müssen. [6, S.77-81]

4.3 Select

Das Select der Daten arbeitet wie folgt. Es wird die WertID im Wörterbuch gesucht, welche dem Wert in der Bedingung der Select-Abfrage entspricht. Im Anschluss wird der Attribut-Vektor gescannt um die Position zu ermitteln an welcher sich die gesuchten Werte befinden. Auf diese Liste kann dann mit Hilfe der WertID, der richtige Wert in Textform projiziert werden. [6, S. 103-106]

4.4 Update

Beim Update von Datensätzen gibt es 2 Szenarien. Zum einen kann es nur zu Status-Updates kommen, das heißt dass nur ein Boolean-Feld innerhalb der Tabelle

verändert wird. Falls hier noch das Führen einer Historie gewünscht ist, kann dies durch Einsatz der Insert-Only Strategie erfolgen.

Das zweite Szenario wäre die tatsächliche Veränderung der Werte. Hierbei eignet sich ebenfalls der Insert-Only Ansatz.

Ansonsten arbeitet das Update identisch dem Einfügen mit neuer Wörterbuchsortierung. Sodass der alte Wert nicht aus dem Wörterbuch gelöscht wird, sondern der neue wird lediglich hinzugefügt. [6, S.85-88]

4.5 Delete

Beim Löschen von Datensätzen gibt es ebenfalls wieder 2 verschiedene Ansätze. Zum einen gibt es die Möglichkeit des physischen Löschens, das heißt dass der Datensatz wirklich aus der Tabelle gelöscht wird. Dabei wird dieser aber lediglich aus der Tabelle gelöscht und aus dem Attribut-Vektor. Im Wörterbuch bleibt der Wert für die entsprechenden Spalte weiterhin gespeichert.

Der zweite Ansatz befasst sich mit dem logischen Löschen, wobei hierbei die Insert-Only Strategie zum Einsatz kommt. [6, S.73-74]

4.6 Insert-Only Strategie

Die Insert-Only Strategie befasst sich wie der Name schon sagt nur mit dem Hinzufügen von neuen Daten. Um die Operationen Update oder Delete damit umzusetzen wird ein Güligkeitsflag und ein Zeitstempel für die Daten eingefügt. Dadurch wird festgelegt, dass Daten immer nur in einem bestimmten Zeitraum gültig sind. Als ungültig werden Daten gekennzeichnet falls diese bearbeitet werden und somit ein neuer gültiger Datensatz erzeugt wird.

Durch die Insert-Only Strategie wird gewährleistet, dass eine komplette Historie der Datensätze angelegt wird, welches in vielen Ländern gesetzlich vorgeschrieben ist. Ebenso wird eine Nebenläufigkeit gewährleistet, da um auf den Datensätzen zu arbeiten nur Lesetransaktionen ausgeführt werden, welche die Datensätze nicht blockieren. [6, S.173-174]

4.7 Anfrageverarbeitung - Joins

Um die Ergebnisse von mehreren Tabellen miteinander zu verknüpfen gibt es verschiedene Algorithmen welche genutzt werden können, zum Beispiel den Sort-Merge-Join. Hierfür muss eine neue Übersetzungstabelle eingeführt werden, welche die WertIDs der Wörterbücher der entsprechenden Spalte aus beiden Tabellen beinhaltet. Daraus muss im Anschluss eine neue eindeutige WertID erzeugt werden, welche in den Attribut-Vektor übertragen wird. Am Ende besitzt jede Tabelle ihr eigenes Wörterbuch, aber zusätzlich entstehen eine gemeinsame Übersetzungstabelle und ein Attribut-Vektor. [6, S.135-141]

4.8 Anfrageverarbeitung - Aggregatfunktionen

Das Ziel von Aggregatfunktionen ist es aus mehreren Daten als Eingabe ein Ergebnis zu erzeugen.

Dabei kommen ebenfalls oft Hilfsstrukturen zum Einsatz. Bei der Count-Funktion zum Beispiel wird in einer Hilfsstruktur das Auftreten eines bestimmten Wertes einer Spalte hochgezählt. In Kombination mit dem Wörterbuch lässt sich dann eine Aussage treffen wie oft bestimmte Werte in der Tabelle stehen. Bei der Min/Max-Funktion wird in der Hilfsdatenstruktur der aktuelle Extremwert gespeichert und falls der nächste Wert in der Tabelle höher oder niedriger als dieser Extremwert ist wird er überschrieben. [6, S.145-147]

5 Transaktionsmanagement, Logging und Recovery

Jede Transaktion oder Änderung im Hauptspeicher erzeugt synchron einen Protokolleintrag (Eintrag in der Journal Datei). Diese Datei befindet sich meist auf Solid State Discs. Das Hinzufügen von Einträgen ist schnell und beeinflusst den Rest nicht. Aus dem Hauptspeicher werden ungültige Eintrage asynchron gelöscht und auf den persistenten Speicher, einen Snapshot, geschrieben. Ein Snapshot ist die komplette Historie des Hauptspeichers in binärer Form. Hierbei findet auch wieder die Insert-Only Strategie eine Einsatzmöglichkeit. [6, S.191-193]

Falls das System ausfällt und neugestartet werden muss, dann wird der aktuelle Snapshot in den Hauptspeicher geladen. Danach müssen noch die Änderungen aus

der Log-Datei (Journal-Datei), welche nicht im Snapshot gespeichert wurden erneut ausgeführt werden. [6, S.201-202]

6 Partitionierung und Replikation

Mit Hilfe der Partitionierung soll eine logische Datenbank in unabhängige Teilmengen voneinander aufgeteilt werden. Um somit eine Leistungssteigerung durch die Parallelität zu erzeugen. Um dies zu erreichen gibt es verschiedene Verfahren.

Bei der Range-Partitionierung wird für jede Partition ein Partitionsschlüssel bestimmt. Um zum Beispiel eine Personendatenbank aufzuteilen kann als Schlüssel das Alter gewählt werden. Alle Personen im Alter von 0-25 Jahre werden auf die erste Partition gespeichert und Personen im Alter von 26-50 Jahre auf eine zweite Partition.

Ein anderes Verfahren wäre die Round-Robin-Partitionierung. Hierbei würden Datensätze einzeln auf die Partitionen verteilt werden und falls das Ende der Partitionen erreicht ist, wird wieder von vorn mit der ersten Partition begonnen. Dabei entsteht ebenfalls ein Load-Balancing. Jedoch kann dieses nicht zu Einhundert Prozent sichergestellt werden, da es immer Datensätze gibt die öfter abgegriffen werden als andere.

Bei der Partitionierung werden die Daten entweder auf verschiedenen Servern gespeichert oder auf unterschiedlichen Medien der Speicherhierarchie. [6, S.65-68]

Eine Replikation der Hauptspeicherdatenbank ist einfacher zu realisieren als eine Partitionierung. Ein replizieren ist ebenfalls schneller als die Transaktionen ins Log zu schreiben, da kein Festplattenzugriff notwendig ist. Jedoch ist eine Replikation sehr kostenintensiv, da ein System verdoppelt werden muss. [7]

7 Literaturverzeichnis

[1] Gabriel, Gluchowski, Pastwa (2009): Data Warehouse & Data Mining. W3L AG.

[2] OLAP. Albrecht Deyhle, Gerhard Radinger.

Abgerufen am 15.Juni.2018.

https://www.controlling-wiki.com/de/index.php/OLAP

[3] SAPs In-Memory-Datenbank Sanssouci (HANA). Computerworld.

Abgerufen am 16.Juni.2018.

https://www.computerworld.ch/business/forschung/saps-in-memory-datenbank-sanssouci-hana-1312143.html

[4] SAP HANA. Mindsquare.

Abgerufen am 16.Juni.2018.

https://mindsquare.de/knowhow/sap-hana/

[5] SAP HANA: Kunden berichten. John Hunt.

Abgerufen am 16.Juni.2018.

https://news.sap.com/germany/2013/06/sap-hana-kunden-berichten/

[6] Hasso Plattner (2013): Lehrbuch In-Memory Data Management. Gabler Verlag.

[7] Hauptspeicherdatenbanken für Unternehmensanwendungen. Krueger, Grund,

Tinnefeld, Eckhart, Zeier, Plattner.

Abgerufen am 20.Juni.2018.

https://pdfs.semanticscholar.org/a9c5/5bea36564aa8bfbf52dac26e3e3673c37ef2.pdf