Spaltenorientierte Datenbanken

Am Beispiel MonetDB

Einführung

Die in einem Unternehmen anfallenden digitalen Daten wachsen seit Jahren auf gigantische Ausmaße an. Aus diesem Unternehmen Werden Datenbanken immer bedeutsamer. Dabei werden Datenbankanwendungen in Unternehmen meist unterteilt in die beiden Bereiche OLTP und OLAP.

Das OLTP, ausgeschrieben „Online Transactional Processing“ beschreibt die Verarbeitung von Transaktionen in Echtzeit, also ohne nennenswerte Zeitverzögerung. Diese Anfragen umfassen meist wenige Datensätze (Tupel) aber viele Attribute. Systeme zur Echtzeit-Transaktionsverarbeitung sind meist zeilenbasiert und wurden in der Vergangenheit bereits stark optimiert.

Mit immer größeren anfallenden Datenmengen nimmt jedoch die Analyse der Daten eine immer bedeutendere Rolle ein. Dies umfasst den Bereich des OLAP („Online Analytical Processing“), also den Bereich der analytischen Datensysteme. Hierbei ist eine effektive und effiziente Auswertung großer Datenmengen die Zielstellung. Diese Anfragen umfassen viele Tupel aber wenige Datensätze. Hier bietet sich unter anderem der Ansatz der Spaltenorientierung an. [1]

Konzept

Ein gewöhnliches Schema einer relationalen Datenbank besteht aus einer zweidimensionaler Tabelle. Dabei ist sind die Informationen einer Relation in einer Zeile gespeichert und entsprechen somit einem Datensatz. Diese Daten müssen aus Sicht des Betriebssystems in einer eindimensionalen Folge von Bytes angeordnet werden. [2]

Beim bekannten zeilenbasiertem Modell wird die Tabelle Zeile für Zeile nacheinander durchlaufen und diese Daten nacheinander auf der Festplatte gespeichert.

Beim so genannten Decomposition Storage Model hingegen werden die Daten komplett vertikal zerlegt. Hier wird für jedes Attribut eine binäre Tabelle mit dem Namen des Attributs erzeugt. Dabei wird als linker Wert ein Stellvertreter des Datensatzes, also der Schlüssel, und als rechter Wert jeweils der zugehörige Attributwert gespeichert. [3] Die Datensätze werden also Spaltenorientiert auf der Festplatte gespeichert. Hierbei werden im Gegensatz zur Zeilenorientierung die Spalten nacheinander durchlaufen und sequentiell gespeichert. [4]

Personennummer	Vorname	Nachname	Geschlecht	Alter	Abteilung	Gehalt
1	Gustav	Gruber	M	40	DBA	3000
2	Frieda	Schulz	W	35	Controlling	3000
3	Peter	Schiller	M	30	Vertrieb	3600
4	Ulla	Heiner	W	25	Forschung	4000

Eigenschaften

Die zeitintensivsten Operationen am Computer sind Festplattenzugriffe. Im Gegensatz zur Prozessor-Geschwindigkeit verbessert sich die Performance der sekundären Speicher nur relativ langsam. Aus diesem Grund erhält man Geschwindigkeitsvorteile bei der Verringerung der benötigten Festplattenzugriffe.

Beim Abruf weniger Spalten vieler Zeilen für analytische Zwecke, beispielsweise die Durchschnittsberechnung eines Attributs über alle Datensätze, können durch eine spaltenorientierte Speicherung die benötigten Daten schneller ermittelt werden. Alle abzurufenden Attributwerte liegen nahe zusammen. Dies spart Festplattenzugriffe, da die Anzahl der zu lesenden Blöcke erheblich verringert wird. Bei der Zeilenorientierung müssten erst sämtliche Datensätze abgerufen und benötigte Spalte extrahiert werden, bevor die Daten verarbeitet werden können.

Spaltenorientierte Datenbanken sind also effizient bei der Abfrage und Zusammenführung weniger Spalten vieler Zeilen, sowie bei einer Änderung einer Spalte für alle Zeilen. Nachteile entstehen jedoch bei Abfragen, welche wenige Zeilen und viele Spalten umfassen. Auch bei Operationen an gesamten Datensätzen, beispielsweise beim Löschen und Einfügen, werden mehr Festplattenzugriffe benötigt, sodass die Performance sinkt.

Kompression/Indexierung

Da die Daten einer Spalte einheitliche Datentypen haben, bieten spaltenorientierte Datenbanken gute Möglichkeiten zur Optimierung des Speicherbedarfs. Durch die Möglichkeit, alle Daten einer Spalte gemeinsam zu komprimieren, sind Kompressionsmethoden, welche häufig die Ähnlichkeit benachbarter Daten ausnutzen, sehr effektiv.

So beispielsweise beim Dictionary Encoding. Dabei wird jeder verschiedene Attributwert als kurzer Binär-Code gespeichert. Die Übersetzungen stehen in einem separat gespeicherten Wörterbuch. Diese Methode ist geeignet bei mehrfach vorkommenden langen Werten. [5]

Bei der Lauflängenkodierung („Rund-Length-Encoding“) wird die Häufigkeit der Wiederholungen eines Attributwerts in der Tabelle gespeichert. Dies ist vor allem bei langen Sequenzen gleicher Werte geeignet, was durch eine Sortierung unterstützt wird. [6]

Wird das Bit-Vector-Encoding verwendet, wird für jeden Wert ein Bitstring mit den Zeilenvorkommnissen gespeichert. Eine 1 im Bitvektor bedeutet dabei, dass die Position den Wert enthält. Somit entspricht die Länge des Bitstrings die Anzahl der Datensätze. Auch hier führt eine geringe Kardinalität zur Verbesserung der Kompressionsrate.

Nr.	Abteilung
1	DBA
2	Controlling
3	Vertrieb
4	Controlling
5	Controlling
6	DBA

Abteilung	Vektor
DBA	100001
Controlling	010110
Vertrieb	001000

Das Delta Encoding bietet sich bei numerischen, sehr nahe beieinanderliegenden Werten an, da hier lediglich die Differenz zum jeweiligen Vorgänger gespeichert wird. Auch hier hilft eine Sortierung. [7]

Bei einer Kompression wird eine Verringerung des Plattenplatzverbrauchs auf Kosten der Lesegeschwindigkeit erzielt. Auch hier steigt die Effizienz bei Operationen an wenigen Attributen vieler Datensätze, da alle Daten einer Spalte gemeinsam komprimiert werden. Der Zugriff auf viele Spalten weniger Datensätze wird jedoch schwieriger, da hierfür erst große Datenmengen dekomprimiert werden müssen. [2]

Seit Mitte der 2000er Jahre wird die Annahme, dass eine Komprimierung immer zu einer geringeren Lesegeschwindigkeit führt, nicht mehr als allgemein gültig angesehen. Mit zunehmender Rechenleistung ist es meist schneller, auf kleine Datenmengen zuzugreifen und diese zu dekomprimieren als auf große unkomprimierte Datenmengen zuzugreifen. Dies gilt auch für Schreibzugriffe, und somit empfehlen auch Hersteller zeilenorientierter Datenbanken, beispielsweise Oracle, eine Kompression auf geeigneten Servern zur Performancesteigerung einzusetzen.

Einsatzgebiete

Mit diesen Eigenschaften sind spaltenorientierte Datenbanksysteme vor allem bei komplexen Anfragen auf sehr großen Datenbanken geeignet. Die Daten können schnell und gezielt für Analysen bereitgestellt werden. Somit werden sie unter anderem beim Data-Mining beispielsweise bei Suchmaschinen und Suchmaschinen, und eben aller analytischen Aufgaben in Unternehmen, dem OLAP, eingesetzt. [8]

In der Realität werden meist spalten- und zeilenorientierte Datenbanken kombiniert, um die Vorteile beider Systeme nutzen zu können. Auch weitere Technik, wie In-Memory Datenbanken spielen hierbei eine Rolle.

Historie und Implementierungen

Mit Invertierten Dateien wurde eine Form der Spaltenspeicherung bereits 1970 eingesetzt. Statistics Canada implementierte bereits 1976 diese Form von Indizes, welche Suchbegriffe mit einem Verweis auf alle entsprechende Dokumente enthielt. Dieses RAPID-System wurde für kanadische Volkszählungen und weitere statistische Anwendungen sowie ebenfalls von anderen Organisationen bis in die 1980er verwendet.

Anfang der 1990er wurde das spaltenorientierte Datenbanksystem Expressway 103 von Expressway Technologies implementiert. 1994 wurde Expressway von Sybase übernommen, welche das Produkt 1995 neu unter dem Namen Sybase IQ veröffentlichten. Dies war lange Zeit das einzige Produkt dieser Nische auf dem Markt.

Heute existieren zahlreiche proprietäre und Open-Source Alternativen. So zum Beispiel SAP HANA und Oracle 12c mit der kostenpflichtigen In-Memory-Option, welche eine Spaltenorientierte Speicherung im Hauptspeicher implementieren. Beispiele für freie Software sind die aus einer Abspaltung von MySQL entstandene MariaDB sowie MonetDB. [2]

MonetDB

MonetDB ist ein freies, spaltenorientiertes und relationales Datenbankmanagementsystem, welches am Centrum Wiskunde & Informatica in Amsterdam in der Programmiersprache C entwickelt wurde. [9]

Das Projekt begann 2002 als Teil eines Forschungsprojekts eines Doktoranden und eines Professors. Am 30. September 2004 wurde die erste Version unter der Mozilla Public Lizenz veröffentlicht. Mit der Version 4 wurden einige Erweiterungen wie ein neues SQL Frontend, welches den SQL:2003 Standard unterstützt, hinzugefügt. 2011 wurde die Codebasis erneuert und bereinigt, die resultierende Version 5 wurde auf die Mozilla Public License Version 2.0 aktualisiert. MonetDB war somit eines der ersten freien zugänglichen DBMS mit Spaltenorientierung, aus welchem Grund Sie sich selbst „The column-store pioneer“ nennen. [10]

MonetDB bietet ODBC und JDBC Schnittstellen und wurde stark auf eine hohe Performance ausgelegt. So wurde es für Multi-Core-Rechner konzipiert und implementiert Techniken zur verteilten Verarbeitung. Es ist beliebig skalierbar, das unterliegende System limitiert die maximale Datenbankgröße. Es unterstützt praktisch eine unbegrenzte Anzahl von Spalten. Jede Spalte wird als eine Datei im Dateisystem abgelegt, welches so mit dem Betriebssystem die Größe der Daten in einer Spalte limitiert. [11]

Die Architektur besteht aus 3 Schichten: dem Front- und Back-End sowie dem Datenbank Kernel. Das Front-End bietet Schnittstellen für Anfragen, welche optimiert werden und in die MonetDB Assembly Language (MAL) übersetzt werden. Durch Erweiterungen werden verschiedene Anfragesprachen, Datenmodelle und Anwendungsbereiche unterstützt. Neben der SQL-Erweiterung existieren auch Erweiterungen für Geografische Informationssysteme, wissenschaftliche Daten und Multimedia-Anwendungen.

Das Back-End optimiert die MAL für die zugrundeliegende Rechnerarchitektur. Bei der Implementierung wurde ein sehr einfaches Datenmodell gewählt und die Anzahl der Abfragefunktionen stark limitiert. Die Unabhängig zu den bei den Nutzern sichtbaren und von den Erweiterungen implementierten High-Level Datenmodellen ermöglicht die breite Unterstützung dieser Modelle. Außerdem wird so eine einfache Struktur ermöglicht, was die Performance erhöht.

Der Datenbank Kernel bietet den Zugriff auf die Daten auf der Festplatte, welche in Binary Association Tables (BATs) gespeichert werden. Diese sind Tabellen, welche zwei Spalten für den Schlüssel und den Wert besitzen. [12]

Das Projekt ExaNeSt, welches vom EU-Förderprogramm für Forschung und Innovation „Horizont 2020“ gefördert wird, forscht an neuen Supercomputern und verwendet MonetDB als DBMS für analytische Zwecke. Das Projekt, an welchem unter Andere auch das Fraunhofer Institut beteiligt ist [13], entwickelt neue Netzwerke, Speicherlösungen und Kühlungen. Bis 2020 soll ein Supercomputer mit einem ExaFlop, also der zehnfachen Rechenleistung des Spitzenreiters der Top500 fertiggestellt sein. [14]

Das Projekt SciLens entwickelt eine Datenbanktechnologie für wissenschaftliche Anwendungen und erweitert MonetDB um eine SciQL-Schnittstelle, also eine Erweiterung von SQL für wissenschaftliche Operationen. [15] Das niederländische Projekt COMMIT bringt 10 Universitäten und Forschungseinrichtungen mit 70 Unternehmen zusammen. Das Ziel ist die Erkenntnisgewinnung aus riesigen Ereignisdatenbanken, welche von Menschen, Sensoren und wissenschaftlichen Observatorien gefüllt werden. [16] Auch hier wird MonetDB verwendet. [17]

Performance

Um die Performance von MonetDB einordnen zu können, wurde es neben einem MySQL-Server auf einem Raspberry Pi 2 installiert. Es wurde in beiden Datenbanksystemen eine Tabelle erstellt, in der eine CSV-Datei mit einer Liste der eine Millionen größten Domains gemessen anhand der Anzahl der verweisenden Subnetze, die so genannte „Majestic Million“ importiert wurde.

In den folgenden Screenshots ist das Import-Statement sowie die hierfür benötigte Zeit zu sehen.

```
mysql> LOAD DATA INFILE '/var/lib/mysql-files/majestic_million.csv'
    -> INTO TABLE majestic_million
    -> FIELDS TERMINATED BY ','
    -> LINES TERMINATED BY '\n'
    -> IGNORE 1 ROWS;
Query OK, 1000000 rows affected (1 min 36.01 sec)
Records: 1000000  Deleted: 0  Skipped: 0  Warnings: 0
```

```
sql>COPY 1000000 OFFSET 2 RECORDS INTO majestic_million FROM '/home/pi/majestic_
million.csv' USING DELIMITERS ',','\n';
1000000 affected rows (27.3s)
```

Das Experiment zeigte, dass MonetDB bei allen ausgeführten Operationen deutlich schneller war. Dass auch der Import dieser Daten mit großem Abstand performanter war liegt wahrscheinlich an der starken Performanceoptimierung. Der Unterschied zur Zeilenoptimierung wurde erst bei analytischen Operationen, wie die Bildung des Mittelwerts über eine Million Werte deutlich.

```
mysql> SELECT AVG(RefSubNets) FROM majestic_million;
+-----------------+
| AVG(RefSubNets) |
+-----------------+
|       1042.9891 |
+-----------------+
1 row in set (7.70 sec)
```

```
sql>SELECT AVG(RefSubNets) FROM majestic_million;
+--------------------------+
| L3                       |
+==========================+
|              1042.989056 |
+--------------------------+
1 tuple (37.203ms)
```

Während MySQL für diese Berechnung 7,7 Sekunden benötigt hat, hat MonetDB das Ergebnis ohne nennenswerte Zeitverzögerung (37,203ms) geliefert.

Quellenverzeichnis

[1] P. D.-I. T. Kudraß, „Themenliste,“ [Online]. Available: http://www.imn.htwk-leipzig.de/~kudrass/Lehrmaterial/Oberseminar/2017/Themenliste.pdf.

[2] „Spaltenorientierte Datenbank - Wikipedia,“ [Online]. Available: https://de.wikipedia.org/wiki/Spaltenorientierte_Datenbank.

[3] R. Hartmann, „Einführung in das Decomposed Storage Model,“ [Online]. Available: http://www2.inf.hochschule-bonn-rhein-sieg.de/~rhartm2m/tuc/nsmdsm_hauptseminar/RobertHartmann_Folien.pdf.

[4] M. Goram, „Spaltenorientierte Datenbanken Definition & Erklärung | Datenbank Lexikon,“ [Online]. Available: http://www.datenbanken-verstehen.de/lexikon/spaltenorientierte-datenbanken/.

[5] Amazon Web Services, Inc., „Byte-Dictionary Encoding - Amazon Redshift,“ [Online]. Available: http://docs.aws.amazon.com/redshift/latest/dg/c_Byte_dictionary_encoding.html.

[6] Amazon Web Services, Inc., „Runlength Encoding - Amazon Redshift,“ [Online]. Available: http://docs.aws.amazon.com/redshift/latest/dg/c_Runlength_encoding.html.

[7] Amazon Web Services, Inc., „Delta Encoding - Amazon Redshift,“ [Online]. Available: http://docs.aws.amazon.com/redshift/latest/dg/c_Delta_encoding.html.

[8] C. Niemann, „Big Data: Herausforderung für Datenbanken,“ [Online]. Available: http://www.zdnet.de/41558518/big-data-herausforderung-fuer-datenbanken/.

[9] „MonetDB - Wikipedia (DE),“ [Online]. Available: https://de.wikipedia.org/wiki/MonetDB.

[10] „MonetDB - Wikipedia (ENG),“ [Online]. Available: https://en.wikipedia.org/wiki/MonetDB.

[11] MonetDB B.V., „Column store features | MonetDB,“ [Online]. Available: https://www.monetdb.org/content/column-store-features.

[12] P. Boncz, „MonetDB: A high performance database kernel for query-intensive applications,“ [Online]. Available: https://dev.monetdb.org/Assets/monetdb_lecture.pdf.

[13] Fraunhofer ITWM, „Fraunhofer ITWM beteiligt an ExaNeSt - Fraunhofer-Institut für Techno- und Wirtschaftsmathematik ITWM,“ 12 02 2016. [Online]. Available: https://www.itwm.fraunhofer.de/presse-und-publikationen/pressearchiv/pressearchiv-2016/12022016-fraunhofer-itwm-beteiligt-an-exanest.html.

[14] ExaNeSt, „ExaNeSt,“ [Online]. Available: http://www.exanest.eu/.

[15] SciLens, „Database technology to advance science | scilens.project.cwi.nl,“ [Online]. Available: https://projects.cwi.nl/scilens/.

[16] COMMIT/, „About COMMIT | Commit/,“ [Online]. Available: http://www.commit-nl.nl/about-commit.

[17] MonetDB, „Project Gallery | MonetDB,“ [Online]. Available: https://www.monetdb.org/Home/ProjectGallery.