Maximilian Eibl, Martin Gaedke. (Hrsg.): INFORMATIK 2017,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 2561

Tsunami: Anfrage-getriebene Anbindung von Datenquellen an ein Datenmanagementsystem

David Haller[1]

Abstract: Die Beantwortung von analytischen Anfragen erfordert oft die Anbindung von externen heterogenen Datenquellen an ein Datenmanagementsystem. Klassische ETL-basierte Ansätze erfordern jedoch die vorherige zeitaufwendige Definition eines einheitlichen Schemas. Existierende Lösungsansätze beschränken sich auf Performance-Optimierung von ETL oder lassen sich nur aufwendig in bestehende Software-Werkzeuge und Arbeitsabläufe integrieren. Wir schlagen einen Ansatz vor, der SQL-Anfragen als partielle Schemadefinitionen für die Datenintegration verwendet. Datenquellen und zugehörige Anfragen werden automatisch für eine spätere Wiederverwendung gespeichert. Es wird eine generische Architektur vorgestellt, welche diese Funktionalität durch einen Proxy-Treiber erbringt. Durch die Evaluation eines Prototyps wird gezeigt, dass der Anwender verglichen mit klassischem ETL Zeit sparen kann, weniger Arbeitsschritte ausführen muss und eigenständiger arbeiten kann, da er den gesamten Anbindungsprozess durch Formulierung seiner Anfrage steuert.

Keywords: Anfrage-getrieben, minimal-invasiv, partielles Schema, SQL, JDBC-Proxy-Treiber

1 Motivation

Die Analyse großer Datenmengen ist in Wirtschaft und Wissenschaft zu einer immer wichtigeren Tätigkeit geworden. Die Datenquellen für diese Analysen stammen dabei zunehmend seltener ausschließlich aus dem eigenen Unternehmen oder der eigenen Forschungseinrichtung. Es ist hingegen immer häufiger der Fall, dass spontan fremde Datenquellen für eine analytische Anfrage schnell zur Verfügung stehen müssen [Wa17a]. Eine Voraussetzung für derartige Analysen ist die Schaffung einer Möglichkeit, Daten aus verschiedenen heterogenen Datenquellen gemeinsam innerhalb eines Systems verwenden zu können.

Der klassische ETL-basierte Ansatz erfordert die Definition eines Zielschemas, bevor erste Anfragen ausgeführt werden können. Diese Trennung von Schema- und Anfragedefinition hat einen sehr hohen Zeitbedarf zur Folge und lohnt sich häufig nicht, wenn nur Teile einer Datenquelle kurzfristig für Anfragen benötigt werden. Weiterhin entsteht ein hoher Kommunikationsaufwand, da das gemeinsame Schema mit allen Benutzern der Datenquellen zu vereinbaren ist, was eine zentrale Administration erforderlich macht. Explorative Anfragen, also die experimentelle Verwendung von neuen Datenquellen, sind mit diesem Ansatz nur schwer umzusetzen. Neue Datenquellen können nicht spontan durch den Benutzer in das System eingebunden werden. Zudem ist eine zweckmäßige Schemadefinition a priori meist nicht bekannt, sondern wird erst durch die Arbeit mit den Datenquellen iterativ entwickelt.

[1] FAU Erlangen-Nürnberg david.haller@fau.de

doi:10.18420/in2017_258

2 Lösungsansatz

Um mit diesen Problemen besser umgehen zu können, soll bei TSUNAMI auf die explizite Definition von Schemata verzichtet werden. Stattdessen dienen die SQL-Anfragen selbst als partielle Schemadefinitionen. Der Anwender kann also durch die Formulierung seiner Anfrage seinen konkreten Datenbedarf spezifizieren, der als partielles Schema extrahiert werden kann. Dabei kann auch auf externe Datenquellen Bezug genommen werden. Dies können unter anderem semistrukturierte Dateien im JSON- oder CSV-Format sein, die entweder lokal im Dateisystem oder auf einem Webserver gespeichert sind. Wie in List. 1 zu erkennen ist, wird das Schema der Datenquelle in der `select`-Klausel definiert, während in der `from`-Klausel der Ort der Datenquelle angegeben wird. Eine solche Anfrage kann nun sofort ausgeführt werden, ohne dass der Benutzer manuell Vorbereitungen treffen muss. Das System soll sich automatisch darum kümmern, die referenzierten Datenquellen zu beschaffen und einzubinden. Durch diese SQL-Erweiterung kann der Anwender neue Datenquellen sofort benutzen, ohne einen ETL-Prozess abwarten zu müssen. Falls tatsächlich notwendig, kann eine umfassende Datenintegration zu einem späteren Zeitpunkt erfolgen. Damit die partiellen Schemadefinitionen nicht verloren gehen, werden alle Anfragen protokolliert. Dieses Protokoll wird mit allen Benutzern des Systems geteilt; es erfüllt somit eine ähnliche Funktion wie ein Metadatenkatalog. Die verwendeten Datenquellen werden in ein zentrales Repository hochgeladen, um ihre dauerhafte Verfügbarkeit sicherzustellen.

```
select concat(columns[0], ' ', columns[1]) as full_name,
       cast(columns[2] as date) as birth_year,
       extract(year from columns[3]) as hire_year
from upload.'http://example.org/employee.csv';
```

List. 1: Anfrage-getriebene Datenintegration anhand eines SQL-Beispiels

3 Verwandte Arbeiten

Es gibt bereits eine Reihe von vorhanden Ansätzen, die ähnliche Ziele wie TSUNAMI verfolgen. DrillBeyond [Eb15] ermöglicht die Einbindung von Webquellen in SQL-Queries, erfordert jedoch das manuelle Pflegen eines Indexes. DataHub [Bh15] erlaubt das gemeinsame Verwalten von Transformationsskripten, ist aber nicht Anfrage-getrieben. SQLShare [Ho13] hingegen arbeitet Anfrage-getrieben, ist aber nur über eine spezielle Webanwendung nutzbar und lässt sich nicht in bestehende Analysewerkzeuge integrieren. Adaptive Schema Databases [Sp17] ist ein probabilistischer Ansatz, der aber auf explizitem Userfeedback basiert, das zum Aufbauen einer Heuristik genutzt wird, um automatisch Schemata für neue Datenquellen zu generieren. Außerdem gibt es noch ETL-nahe Ansätze, die sich aber auf Performance-Optimierungen konzentrieren, wie Lazy ETL [Ka13], RiTE [TPL08], und AScale [Ko16]. Auch einige kommerzielle Datenbanksysteme [Ora] [Mi] ermöglichen über User Defined Table Functions, semistrukturierte Daten automatisiert zu importieren. Man kann diesen Ansatz als ELT (Extract, Load, Transform) bezeichnen. Dieser hat aber die gleichen Nachteile wie ETL, nur dass die Transformation hier direkt in der Datenbank durch materialisierte Views erfolgt. Schema- und Anfragedefinition finden also weiterhin getrennt statt.

4 Anfrage-getriebene Anbindung von Datenquellen

4.1 Anfragevorverarbeitung

Nachdem der Benutzer seine Anfrage formuliert hat (siehe List. 1), wird diese zuerst protokolliert. TSUNAMI sucht anschließend in der Anfrage nach Datenquellen, die mit dem Pseudo-Schema `upload` gefolgt von einer Quellen-URI gekennzeichnet sind. Falls eine Datenquelle noch unbekannt ist oder seit der letzten Benutzung aktualisiert wurde, wird sie automatisch heruntergeladen und zum zentralen Repository hinzugefügt. Die Anfrage wird nun so umgeschrieben, dass die vom Benutzer angegebene Quellen-URI der Datenquelle durch einen Verweis auf die korrespondierende Datenquelle im Repository ersetzt wird. Schließlich kann die Anfrage sofort ausgeführt werden. Die spontane Anbindung von Datenquellen wird dadurch ermöglicht, dass keine vorherige Konfiguration erforderlich ist. Es ist optional möglich, das Ergebnis einer `select`-Anfrage als neue Tabelle abzuspeichern. Dies erfolgt über eine Anweisung nach dem Muster `create table <name> as <select-query>`.

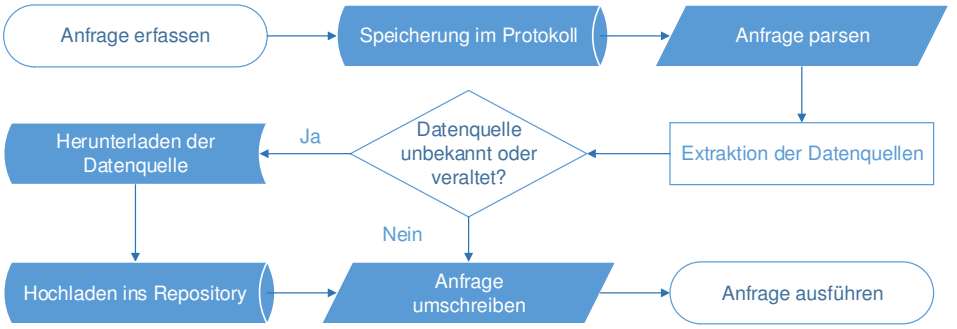

Abb. 1: Ablaufdiagramm der Anfragevorverarbeitung

4.2 Architektur

Die Implementierung der beschriebenen Funktionalität zur Anfragevorverarbeitung erfolgt mittels eines JDBC-Proxy-Treibers. Dieser leitet alle Anfragen an den originalen JDBC-Treiber weiter, kann jedoch Anfragen vor der Weiterleitung abfangen und modifizieren. Auf diese Weise lassen sich SQL-Erweiterungen implementieren, ohne das dahinterliegende Datenmanagementsystem oder die darauf zugreifenden Clients anpassen zu müssen. TSUNAMI kann daher mit jedem Werkzeug eingesetzt werden, das über eine JDBC-Schnittstelle verfügt. Der interne Mechanismus des Treibers nutzt die Reflection-API von Java [Orb]. Die Objekte des Original-Treibers, die `java.sql.Statement` implementieren, werden dabei in ein Wrapper-Objekt eingepackt, sodass alle Aufrufe von `executeQuery()` und ähnlichen Methoden abgefangen werden können. TSUNAMI führt dann die in Abb. 1 beschriebenen Schritte aus, bevor die umgewandelte Anfrage an die SQL-Engine weitergereicht wird. Der Prototyp nutzt Apache Drill [Apa] als SQL-Engine und Apache Hadoop [Apb] als Repository, als Query-Log kommt eine relationale Datenbank zum Einsatz. Apache Drill ist

eine freie SQL-Engine, die sich auf die Verarbeitung von Anfragen auf semistrukturierte Daten spezialisiert hat. Drill basiert auf den Ideen von Google Dremel [Me10]. Durch den Proxy-Treiber-Ansatz kann Drill um fehlende Features erweitert werden, ohne eine Abhängigkeit zu Drill aufzubauen.

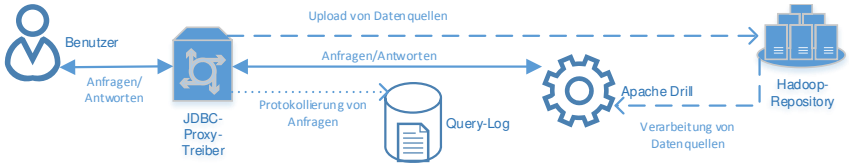

Abb. 2: Architekturskizze von TSUNAMI

5 Evaluation

5.1 Verknüpfung heterogener Datenquellen

Der Anwender kann die Anbindung einer zusätzlichen Datenquelle im Optimalfall durch eine einzige SQL-Anfrage (siehe List. 1) steuern. Dadurch profitiert der Anwender von der Ausdrucksmächtigkeit von SQL und ist nicht auf den möglicherweise beschränkten Funktionsumfang eines grafischen ETL-Werkzeuges angewiesen. Externe Datenquellen können direkt im vorgesehenen Analysekontext referenziert und explorativ erkundet werden. Falls beispielsweise die aufbereiteten Daten für eine konkrete SQL-Anfrage ungeeignet sind, können notwendige Transformationen und Schemaänderungen durch das schlichte Anpassen der SQL-Anfrage umgesetzt werden. Ein Wechsel in das ETL-Werkzeug und das erneute Exportieren des Transformationsergebnisses in das Datenmanagementsystem entfällt. Dadurch spart der Anwender Zeit und muss weniger einzelne Arbeitsschritte ausführen, außerdem kann er seine vertraute Anfragesprache und sein Analysewerkzeug weiterhin benutzen. Die neu angebundenen Datenquellen werden automatisch in einer homogenen Ausführungsumgebung gespeichert und sind auch für andere Benutzer verfügbar, sodass eine manuelle Verwaltung nicht erforderlich ist.

5.2 Leistungsfähigkeit

Es stellt sich die Frage, wie sich der Reflection-basierte Ansatz auf die Performance auswirkt. Zu diesem Zweck wird der normale Apache Drill Treiber mit dem TSUNAMI-Treiber in zwei Konfigurationen verglichen: In der minimalen Konfiguration werden alle TSUNAMI-Funktionen deaktiviert; die SQL-Anfrage wird unverändert an Apache Drill durchgereicht. Die Standard-Konfiguration führt dagegen alle Funktionen aus. Als Beispiel wird eine simple Aggregatsanfrage verwendet (siehe List. 2).

```
select max(salary) from cp.'employee.json';
```

List. 2: Anfrage zum Testen der Performance von TSUNAMI und Apache Drill

Die Datei `employee.json` liegt bei den ersten beiden Testreihen bereits im Repository, bei der letzten auf der lokalen Festplatte. Wie in Tab. 1 zu sehen ist, beträgt der reine

Reflection-Overhead ca. 37,6 ms; der gesamte Tsunami-Overhead summiert sich damit auf 124,6 ms für diese Anfrage. Der Reflection-Overhead ist konstant, da er nicht von den Eingabedaten abhängt; bei komplexeren Anfragen und größeren Datenquellen fällt er also immer weniger ins Gewicht. Die Laufzeit hängt damit wie bei ETL hauptsächlich von der Geschwindigkeit der Netzwerkverbindung und der Rechenleistung ab.

Iteration	Apache Drill	Tsunami (min)	Tsunami (full)
1	151 ms	182 ms	256 ms
2	133 ms	156 ms	305 ms
3	149 ms	179 ms	260 ms
4	134 ms	202 ms	279 ms
5	154 ms	190 ms	244 ms
ø	144,2 ms	181,8 ms	268,8 ms

Tab. 1: Dauer der Ausführung von List. 2

6 Ergebnisse

Tsunami hat gezeigt, dass eine Anfrage-getriebene Anbindung von Datenquellen softwaretechnisch umsetzbar und in der Praxis anwendbar ist. Dabei ist eine modulare Software entstanden, die leicht um zusätzliche Funktionen erweitert werden kann und sich in bestehende Analysewerkzeuge integrieren lässt. Der Benutzer kann durch Tsunami seine vertraute Anfragesprache verwenden, um schnell bisher unbekannte Datenquellen anzubinden. Es wurde mit Tsunami eine minimalinvasive Lösung umgesetzt, welche in die bestehenden Arbeitsabläufe nur geringfügig eingreift und keine zentrale Administration erfordert.

7 Ausblick

Tsunami soll später innerhalb des Ocean-Projektes zum Einsatz kommen. Das Ocean-Projekt [Wa17a, Wa17b, Wa16] beschäftigt sich mit der Erforschung und Entwicklung eines *Anfrage-getriebenen Wissenstransfersystems* (AWTS). Ein AWTS ermöglicht die Extraktion von Wissen aus Datenbankanfragen und die Weitergabe dieses Wissens an andere Benutzer des Systems. Das Anfrageprotokoll könnte dafür in Zukunft automatisch analysiert und das darin zugrundeliegende mentale Modell partiell extrahiert werden, um dem Benutzer bei der Formulierung seiner Anfrage durch das System zu helfen. So könnten geeignete Datenquellen automatisiert aufgefunden und vorgeschlagen werden. Das gleiche gilt für ähnliche Anfragen, die zuvor von anderen Benutzern verwendet wurden. Diese könnten nach geringer Modifikation auch für das aktuelle Problem des Benutzers relevant sein.

8 Danksagung

Mein besonderer Dank gilt Prof. Dr. Richard Lenz und Andreas Wahl vom Lehrstuhl für Datenmanagement an der Universität Erlangen für ihre fachliche und persönliche Unterstützung meiner Arbeit.

Literaturverzeichnis

[Apa] Apache Drill - FAQ. https://drill.apache.org/faq/, aufgerufen am 8.5.2017.

[Apb] HDFS Architecture. https://hadoop.apache.org/docs/r2.8.0/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html, aufgerufen am 8.5.2017.

[Bh15] Bhardwaj, Anant; Bhattacherjee, Souvik; Chavan, Amit; Deshpande, Amol; Elmore, Aaron J.; Madden, Samuel; Parameswaran, Aditya G.: DataHub: Collaborative Data Science & Dataset Version Management at Scale. In: CIDR '15. S. 7, 2015.

[Eb15] Eberius, Julian; Thiele, Maik; Braunschweig, Katrin; Lehner, Wolfgang: Processing Multi-Result Open World SQL Queries. In: SSDBM'15. 2015.

[Ho13] Howe, Bill; Ribalet, Francois; Chitnis, Sagar; Armbrust, Ginger; Halperin, Daniel: SQLShare: Scientific Workflow Management via Relational View Sharing. CSE, S. 1–1, 2013.

[Ka13] Kargín, Yağız; Pirk, Holger; Ivanova, Milena; Manegold, Stefan; Kersten, Martin: Instant-On Scientific Data Warehouses. In: BIRTE'12, S. 60–75. 2013.

[Ko16] Kozielski, Stanisław; Mrozek, Dariusz; Kasprowski, Paweł; Małysiak-Mrozek, Bożena; Kostrzewa, Daniel: AScale: Big/Small Data ETL and Real-Time Data Freshness. CCIS, 613:315–327, 2016.

[Me10] Melnik, Sergey; Gubarev, Andrey; Long, Jing Jing; Romer, Geoffrey; Shivakumar, Shiva; Tolton, Matt; Vassilakis, Theo: Dremel: Interactive Analysis of Web-Scale Datasets. In: VLDB'10. S. 330–339, 2010.

[Mi] Table-Valued User-Defined Functions. https://technet.microsoft.com/en-us/library/ms191165(v=sql.105).aspx, aufgerufen am 8.5.2017.

[Ora] Using Pipelined and Parallel Table Functions. https://docs.oracle.com/cd/B19306_01/appdev.102/b14289/dcitblfns.htm, aufgerufen am 8.5.2017.

[Orb] The Reflection API. https://docs.oracle.com/javase/tutorial/reflect, aufgerufen am 8.5.2017.

[Sp17] Spoth, William; Sadat, Bahareh; Chan, Eric S; Gawlick, Dieter: Adaptive Schema Databases. In: CIDR '17. 2017.

[TPL08] Thomsen, Christian; Pedersen, Torben Bach; Lehner, Wolfgang: RiTE: Providing on-demand data for right-time data warehousing. In: ICDE'10. S. 456–465, 2008.

[Wa16] Wahl, Andreas M.: A minimally-intrusive approach for query-driven data integration systems. In: ICDEW'16. S. 231–235, 2016.

[Wa17a] Wahl, Andreas M.; Endler, Gregor; Schwab, Peter K.; Herbst, Sebastian; Lenz, Richard: Anfrage-getriebener Wissenstransfer zur Unterstützung von Datenanalysten. In: BTW'17. S. 165–174, 2017.

[Wa17b] Wahl, Andreas M.; Endler, Gregor; Schwab, Peter K.; Herbst, Sebastian; Lenz, Richard: We Can Query More than We Can Tell: Facilitating Collaboration Through Query-Driven Knowledge-Sharing. In: CSCW'17. S. 335–338, 2017.