Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

Projektarbeit

PostgresSQL - Rekursion auf Basis generischer Stored Procedures

Fachbereich Informatik
Referent: Prof. Dr. Harm Knolle

eingereicht von:
Rolf Kimmelmann, Jennifer Wittling, Jan Löffelsender

Sankt Augustin, den 12.11.2018

Zusammenfassung

Problemstellung und Erkenntnisinteresse

In den letzten Jahren haben Graphdatenbanken an Bedeutung gewonnen, da sich mit diesen bestimmte Fragestellungen besonders schnell lösen lassen. Graphdatenbanken haben den Vorteil, dass sich insbesondere Beziehungen zwischen Objekten gut abbilden und sehr performant abfragen lassen. Bei relationalen Datenbanken ist es zur Darstellung von Beziehungen zwischen Objekten erforderlich die verschiedenen Tabellen mittes des JOIN Operators zu verknüpfen. Diese Verknüpfungen können schnell zu einem großen Rechenaufwand und langen Laufzeiten führen. Es soll am Beispiel von Postgres untersucht werden, ob und wie sich Graphen in relationalen Datenbanken abbilden lassen. Weiterhin soll analysiert werden, ob und für welche Problemstellungen es sinnvoller ist Graphen in einer relationale Datenbank statt einer Graphdatenbank abzubilden. Ist es zukünftig notwendig für die performante Verarbeitung steigender Datenmengen auf neue Technologien, wie Graphdatenbanken zu schwenken oder lassen sich die klassischen relationalen Datenbanken so erweitern, dass diese Problemstellungen ähnlich effizient lösen können.

Aktueller Forschungsstand

NoSQL Datenbanken und insbesondere Graphdatenbanken sind im Gegensatz zu den relationalen Datenbanken flexibler und bei der Lösung bestimmter Probleme weniger rechen- und speicherintensiv. Insbesonder wenn es um die Auflösung von Beziehungen bzw. um die Traversierung über einen Graphen geht, bieten Graphdatenbanken Vorteile gegenüber den herkömmlichen relationalen Datenbanken. In der Praxis wurde jedoch auch die Beobachtung gemacht, dass durch die Verwendung von Stored Procedures die Traversierung über einen Graphen mittels einer relationalen Datenbank ähnlich schnell umgesetzt werden kann, wie mit einer Graphdatenbank.

Zielsetzung

Es soll das Modell als grundlegender technologische Aspekt von Graphdatenbanken kurz erläutert werden. Zielsetzung dieser Arbeit ist es einen Graphen in der relationalen Datenbank Postgres abzubilden und zu vergleichen, wie sich die Traversierung über diesen Graphen effizient umsetzen lässt. Zunächst soll die Umsetzung mittels klassischer SQL Operationen erfolgen. Anschließend sollen die Problemstellungen mittels Stored Procedures, sowie der Rekursion mittels PL/SQL gelöst werden. Die Ergebnisse der verschiedenen Vorgehensweisen sollen miteinander verglichen werden.

Inhaltsverzeichnis

1 Graph-Datenbanken - Grundlegende technologische Aspekte

1.1 Einführung

1.2 Modell

1.2.1 Graph

Ein Graph ist mathematisch folgendermaßen definiert:

Definition. *Ein Graph(graph)*$G = (V, E, \gamma)$ *ist ein Tripel bestehend aus:*

- V*, einer nicht leeren Menge von Knoten(vertices)*
- E*, einer Menge von Kanten (edges) und*
- γ *, einer Inzidenzabbildung (incidence relation), mit* $\gamma : E \longrightarrow \{X | X \subseteq V, 1 \leq |X| \leq 2\}$

Zwei Knoten $a, b \in V$ *heißen adjazent(adjacent) genau dann wenn* $\exists e \in E : \gamma(e) = \{a, b\}$.
Ein Knoten $a \in V$ *und eine Kante* $e \in E$ *heißen inzident (incident) genau dann wenn* $a \in \gamma(e)$. [1]

1.2.2 Property Graphen

[1] [Bec18, Seite 21]

1.2.3 Hypegraphen

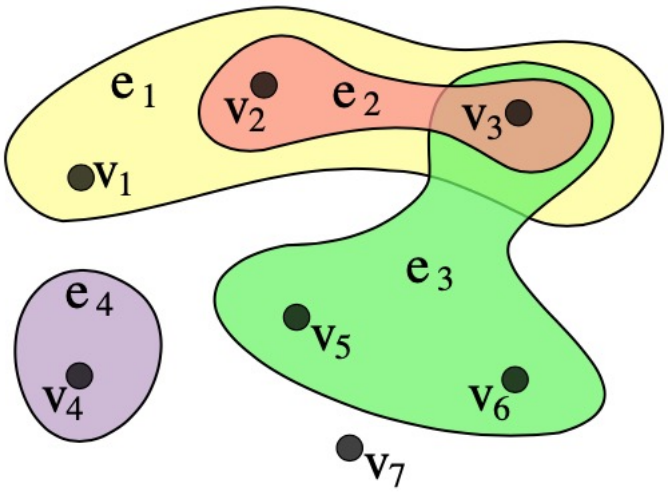
Ein Hypergraph ist folgendermaßen definiert:

Definition. *Let* $X = \{v_1, v_2, ..., v_n\}$ *be a finite set, and let* $E = \{e_1, e_2, ..., e_m\}$ *be a family of subsets of* X *such that*

$$e_i \neq \varnothing (i = 1, 2, ..., m) \cup_{i=1}^{m} e_i = X.$$

The pair $H = (X, E)$ *is called a hypergraph with vertex set* X *and hyperedge set* E*. The elements* $v_1, v_2, ..., v_n$ *of* X *are vertices of hypergraph* H*, and the sets* $e_1, e_2, ..., e_m$ *are hyperedges of hypergraph* H*.*[2]

Das folgende Bild zeigt einen Hypergraphen:



Im Vergleich zum normale Graphen können die Kanten eines Hypergraphen eine beliebige Kardinalität haben (siehe Definition Definition Graph Kapitel 1.2.1) . Beim normalen Graphen können die Kanten nur die Kardinalität $1 \leq |X| \leq 2$ haben. Die Hyperedges in einem Hypergraphen sind somit eine beliebige Menge von Knoten. In einem normale Graphen sind die Kanten, eine in einem Intervall festgelegte Menge von Knoten:

$$X = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\} \text{ Knoten}$$

$$E = \{e_1, e_2, e_3, e_4\} \text{ Kanten}$$

$$E = \{e_1, e_2, e_3, e_4\} = \{\{v_1, v_2, v_3\}, \{v_2, v_3\}, \{v_3, v_5, v_6\}, \{v_4\}\}$$

[2] Vgl. [ZSHZ18, Seite 2]

2 Graph-Datenbanken und -Frameworks - Ausgewählte Systeme

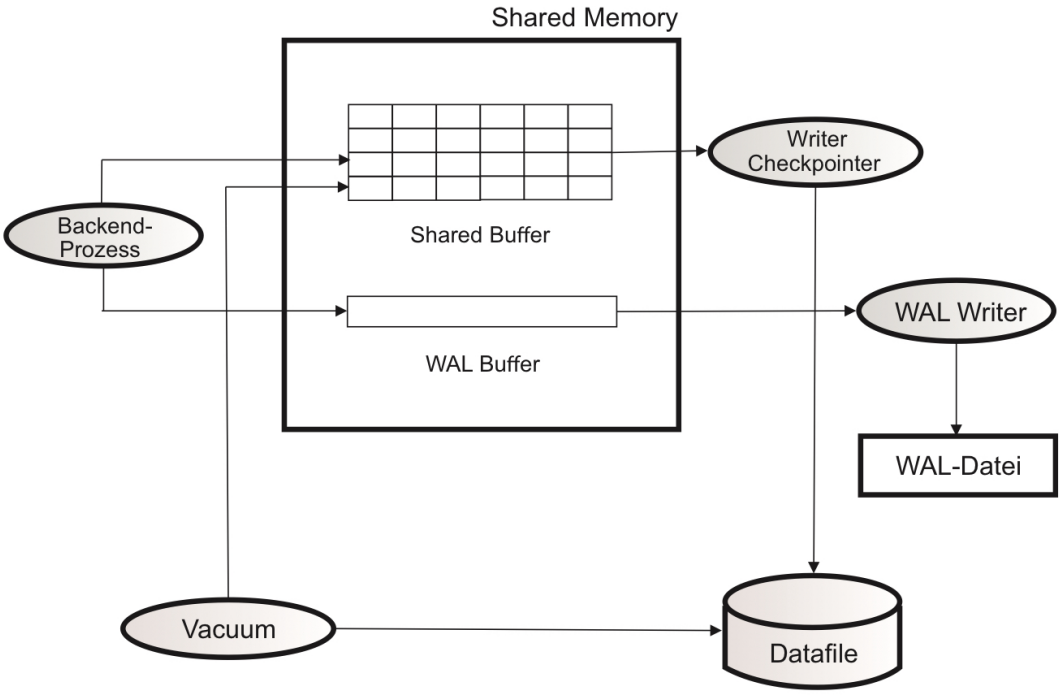
2.1 PostgreSQL

PostgreSQL ist eine Objektrelationale Datenbank. Weiterentwicklungen wird von der PostgreSQL Global Development Group durchgeführt. PostgreSQL steht unter der PostgreSQL-Lizenz, welche sehr stark der GNU-Lizenz ähnelt. Auf Basis von PostgreSQL gibt es mehrere, zum Teil auch kommerzielle, Forks wie zum Beispiel Amazon Redshift oder EnterpriseDB.

2.2 Allgemein

2.3 Architektur

Eine einzelne PostgreSQL-Instanz mit einer oder mehreren Datenbanken wird auch Cluster genannt. Ein PostgreSQL-Cluster ist ein eigener Prozess mit einem eigenen Datenverzeichnis eigenen Konfigurationsdatei sowie einem eigenen Transaktionslog. Die Architektur von PostgresSQL ist in folgendem Bild gegeben:



Diese Architektur ergibt sich aus der Gegebenheit, dass alle Daten normalerweise nicht in den Shared Buffer passen. Ein Teil der Datenbank befindet sich im Datafile, aus der bei Bedarf gelesen werden kann, der andere Teil im Shared Buffer. Die Hauptaufgabe des Shared Buffer ist es, Input Output Operationen (I/O Operationen) auf das Datafile zu minimieren und möglichst viele Operationen im Speicher durchzuführen. Die Motivation möglichst viele Operation im Speicher durchzuführen besteht darin, dass Operationen im Speicher schneller ausgeführt werden. Werden Operationen im Speicher ausgeführt, so werden I/O Operationen auf das Datafile reduziert. Der Writer Prozess ist für die Synchronisation des Zustands des Shared Buffers und der Tablespaces verwantwortlich. Dieser Prozess schreibt Datenblöcke aus dem Shared Buffer auf die Tablespaces innerhalb des Datafile. Der Writer Checkpoint ist dafür verantwortlich, dass alle geänderten Datenblöcke innerhalb des Shared Buffer in das Datafile geschrieben werden. [1]
Eine PostgreSQL-Instanz wird als Server Prozess mit eigenem Datenverzeichnis und einer eigenen Konfigurationsdatei sowie einem eigenen Transaktionslog.

[1] Vgl. [Fro18, Seite 26]

2.4 Datenmodell

. Daten werden in Form von Tabellen abgelegt. Im Filesystem legt PostgreSQL Dateien im sogenannten $PGDATA-Verzeichnis ab, dieses wird beim Start von PostgreSQL festgelegt. Alle Date werden im Verzeichnis Base unterhalb von $PGDATA abgelegt. Einzelne Tabellen oder Indices können in Tablespaces ausgelagert werden. Für einen Tablespace wird ein neuer Unterordner im $PGDATA-Verzeichnis angelegt. Es ist auch möglich einzelne Spalten einer Tabelle in einen anderen Tablespace zu verschieben.

2.5 Indexe

2.6 Anfragemethoden

2.7 Konsistenz

PostgreSQL schreibt einen Transaktionslog (Write-Ahead Log, WAL). Dieser wird als WAL-Buffer im Arbeitsspeicher und als WAL-Dateien auf der Festplatte geführt. Bei jedem Commit einer Transaktion wird zunächst das WAL aktualisiert bevor die Bestätigung an den Client gesendet wird. Der walwriter-Prozess schreibt periodisch die WAL-Buffer auf die Festplatte.

3 Graph-Datenbanken im praktischen Einsatz: OLTP

3.1 PostgresSQL: OLTP

3.2 Ausgewählte Use Cases

3.2.1 CSV-Import

Beim Import von (CSV)-Dateien wird zwischen import vom Clientsystem und Import vom Serversystem unterschieden. Für den Import vom Client wird das psql-Statement \copy verwendet. \copy liest Informationen aus einer Datei die vom psql-Client aus erreichbar sein muss. [Pos18]

Listing 3.1: CSV Input

```
\copy Beitraege
FROM './data/Beitraege.csv' DELIMITER ',' CSV HEADER;
```

3.3 Beurteilung

4 Graph-Datenbanken im praktischen Einsatz: OLAP

4.1 PostgreSQL: OLAP

4.2 Benchmark

4.2.1 Standard SQL

4.2.2 Stored Procedures

4.2.3 PL/SQL-Recursion

4.3 Datenbankzugriffe

4.3.1 Zugriffsart Aggregation

4.3.2 Zugriffsart Traversierung

4.4 Interpretation der Ergebnisse

Literaturverzeichnis

[Ang12] Angles, Renzo: A comparison of current graph database models. In: *Data Engineering Workshops (ICDEW), 2012 IEEE 28th International Conference on* IEEE, 2012, S. 171–177

[APPDSLP13] Angles, Renzo ; Prat-Pérez, Arnau ; Dominguez-Sal, David ; Larriba-Pey, Josep-Lluis: Benchmarking database systems for social network applications. In: *First International Workshop on Graph Data Management Experiences and Systems* ACM, 2013, S. 15

[AU95] Aho, Alfred V. ; Ullman, Jeffrey D.: *Foundations of computer science*. USA : Computer Science Press, 1995 `http://infolab.stanford.edu/~ullman/focs.html`

[Bec18] Becker, Peter: *Graphentheorie*. `http://www2.inf.h-brs.de/~pbecke2m/graphentheorie/einfuehrung.pdf`. Version: 2018

[Eis03] Eisentraut, Peter: *PostgreSQL Das Offizielle Handbuch*. mitp-Verlag GmbH/Bonn, 2003

[Fro18] Froehlich, Lutz: *PostgresSQL*. Carl Hanser Verlag München, 2018

[Gru17] Grucia, Jelena: *PostgreSQL and GraphQL*. `https://blog.cloudboost.io/postgresql-and-graphql-2da30c6cde26`. Version: 2017

[KHA+16] Kucuk, Ahmet ; Hamdi, Shah M. ; Aydin, Berkay ; Schuh, Michael A. ; Angryk, Rafal A.: Pg-Trajectory: A PostgreSQL/PostGIS based data model for spatiotemporal trajectories. In: *2016 IEEE International Conferences on Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom)(BDCloud-SocialCom-SustainCom)* IEEE, 2016, S. 81–88

[Kud15] Kudrass, Thomas: *Taschenbuc Datenbanken*. Fachbuchverlag Leipzig im Carl Hanser Verlag, 2015

[Pos18] PostgreSQL Global Development Group: *PostgreSQL 11.1 Documentation*. `https://www.postgresql.org/docs/11`. Version: 2018

[Red12] Redmond, Eric: *Sieben Wochen, sieben Datenbanken*. O'Reilly Verlag, 2012

[Sas18] Sasaki, Bryce M.: *Graph Databases for Beginners: The Basics of Data Modeling*. https://neo4j.com/blog/data-modeling-basics/. Version: 2018

[ZSHZ18] Zhang, Hongliang ; Song, Lingyang ; Han, Zhu ; Zhang, Yingjun: *Hypergraph Theory in Wireless Communication Networks*. Springer, 2018

Eidesstattliche Erklärung

Ich versichere an Eides Statt durch meine eigenhändige Unterschrift, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe angefertigt habe. Alle Stellen, die wörtlich oder dem Sinn nach auf Publikationen oder Vorträgen anderer Autoren beruhen, sind als solche kenntlich gemacht. Ich versichere außerdem, dass ich keine andere als die angegebene Literatur verwendet habe. Diese Versicherung bezieht sich auch auf alle in der Arbeit enthaltenen Zeichnungen, Skizzen, bildlichen Darstellungen und dergleichen.

Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Sankt Augustin,
den 27. November
2018
Ort, Datum

Rolf Kimmelmann, Jennifer Wittling, Jan Löffelsender