Diplomarbeit – Exposé

Graphdatenbanksysteme

Überblick und Benchmark

Benjamin Gehrels

16. Juli 2012

1 Motivation

Viele Strukturen der realen Welt lassen sich als Graphen interpretieren, einer mathematischen Repräsentation von Netzen. Bei einigen Problemen ist dies offensichtlich, so beispielsweise bei Straßennetzen, Beziehungsgeflechten zwischen Personen oder den Reaktionswegen biochemischer Komponenten. Da auch Baumstrukturen sich als Graphen formalisieren lassen, ist dies auch für Hierarchien, Ontologien oder Stücklisten in der Industrie, bei denen einzelne Produkte jeweils aus Vorprodukten zusammengesetzt sind, möglich. Auch in der objektorientierten Programmierung werden Graphen aufgebaut: So lassen sich Objektinstanzen als Knoten und ihre Beziehungen als Kanten interpretieren.

Möchte man solche Strukturen automatisiert verarbeiten, so ist es häufig sinnvoll, diese effizient in Datenbanksystemen zu speichern und abzufragen. Es existieren sogenannte Graphdatenbankmanagementsysteme, welche Graphen als zu Grunde liegendes Datenmodell nutzen. Die Anfragesprachen sind häufig auf typische Graphanfragemuster ausgelegt und auch die Datenspeicherung erfolgt meist auf eine Art und Weise, die das Verfolgen von Kanten effizienter löst als andere Klassen von Datenbankmanagementsystemen.

Diese Arbeit soll die Frage klären, welche relevanten Graphdatenbankmanagementsysteme momentan existieren, auf welchen Graphenmodellen diese basieren und welche Anfragemöglichkeiten mit welcher Mächtigkeit in den jeweiligen Systemen existieren. Ergänzt werden soll diese Arbeit um eine Geschwindigkeitsmessung der Datenbanksysteme mittels verschiedener Anfragemuster auf Graphen verschiedener Größen.

2 Hintergrund

Relationale Datenbankmanagementsysteme stellen wohl eine der am häufigsten verwendeten Klassen von Datenbankmanagementsystemen dar. Die Nutzung von relationalen Datenbanksystemen zur Speicherung von Graphen bringt aber eine Reihe von Problemen mit sich. So gibt es in SQL kaum Anfrageoperatoren, die häufig verwendete Anfragemuster an Graphen – Tiefensuchen, Breitensuchen, Pattern Matching – direkt abbilden. Hierbei müssen häufig Pfade unbekannter Länge gesucht werden. Die im SQL-Standard spezifizierten rekursiven Anfragen, welche dies ermöglichen, werden nur von wenigen relationalen Datenbankmanagementsystemen unterstützt. Die Anfragemuster müssen daher häufig entweder mittels recht komplexer SQL-Konstrukte oder in der Anwendungssoftware umgesetzt werden.

Graphdatenbanken können hier eine Alternative darstellen. In den letzten Jahren kam eine Vielzahl neuer Graphdatenbanksysteme auf den Markt. Die Idee, Netzwerkstrukturen als Datenmodell für Datenbanksysteme zu nutzen, reicht dabei weit zurück.

Charles Bachman entwarf Mitte der 1960er Jahre mit dem *Integrated Data Store* eines der ersten Allzweckdatenbanksysteme. Es basierte auf dem Datenmodell eines gerichteten, zyklenfreien Graphen aus Records und Pointern, welches später (1969) von der *Conference on Data Systems Languages* als *network model* (auch *CODASYL data model*) spezifiziert wurde. 1968 veröffentlichte IBM das *Information Management System*, welches auf dem Datenmodell eines gerichteten Baums basierte, *Hierarchisches Modell* genannt [Singh, 2009, 1.9].

Das von Codd [1970] vorgeschlagene *Relationale Modell* verdrängte in der Folge nach der Einführung von *SQL/DS* durch *IBM* in den frühen 1980er Jahren zunehmend das Netzwerk- und das hierarchische Modell , unter anderem aufgrund der Einfachheit des relationalen Modells, der einfachen Nutzbarkeit der relationalen Datenbanksysteme auch durch Nicht-Programmierer mittels SQL als Anfragesprache und der SQL innewohnenden, deklarativen Abstraktion von Implementierungsdetails des Datenbanksystems [Singh, 2009, 1.9].

Die Idee, Graphdatenbanken als Allzweckdatenbanken zu nutzen, entstand Anfang der 1990er Jahre. So beschreiben Amann und Scholl [1992] ein seinerzeit „wachsendes Interesse an Graphmodellen und -sprachen in der aktuellen Datenbankforschung“[1], insbesondere für Hypertext- und GIS-Anwendungen. Sie entwerfen sowohl ein Datenmodell basierend auf einem gerichteten gelabelten Multigraphen als auch eine Anfragesprache hierfür.

In den späten 1990ern wurden vermehrt objektorientierte Datenbankmanagementsysteme entwickelt [Singh, 2009, 1.9]. Die von der *Object Data Management Group* definierte *Object Definition Language* basiert auf einem Graphen als abstraktem Datenmodell, mit Objekten als Knoten und Relationen zwischen ihnen als Kanten [Lin, 2003, 2.2.2]. Auch Vererbungshierarchien zwischen Klassen lassen sich als Graphen auffassen.

[1] Eigene Übersetzung.

Mit dem Aufkommen des *Semantic Web* als Forschungsgebiet um die Jahrtausendwende kam auch die Standardisierung des *Ressource Description Framework* [Klyne und Carroll, 2004] als Datenmodell. Es beschreibt Fakten als Subjekt-Prädikat-Objekt-Tripel. Mit Subjekt und Objekt als Knoten und Prädikaten als Kanten formt es einen Graphen. Damit einhergehend wurde auch eine neue Klasse von Datenbanken geschaffen, sogenannte Triple Stores. *SPARQL* [Prud'hommeaux und Seaborne, 2008], die wohl prominenteste Anfragesprache für Semantic-Web-Daten, ist ebenfalls als Graphanfragesprache konzipiert.

Das Interesse an der Analyse sehr großer Graphen, sei es im Bereich sozialer, technischer, semantischer oder biologischer Netze, führte in der letzten Zeit zu einem neuerlichen Interesse an Graphdatenbanken [Angles und Gutierrez, 2008, 2.4]. Auch für die immer häufiger anzutreffenden Empfehlungssysteme, insbesondere im E-Commerce-Bereich, stellen sie eine gute Grundlage dar [Rodriguez und Neubauer, 2010, 3.1].

3 Ziel der Arbeit

Im Rahmen dieser Arbeit soll ein Überblick über aktuell verfügbare Graphdatenbankmanagementsysteme gegeben werden. Die Arbeit ist beschränkt auf Systeme, welche momentan aktiv entwickelt werden, die hinreichend dokumentiert sind und von denen eine Softwareversion zur akademischen Nutzung kostenfrei öffentlich verfügbar ist. Im Einzelnen sind dies

- Neo4j
- OrientDB
- InfoGrid
- Objectivity InfiniteGraph
- FlockDB
- HyperGraphDB [Iordanov, 2010]
- DEX [Martínez-Bazan et al., 2007]
- AllegroGraph

Darüber hinaus soll ein Blick auf die verteilte Verarbeitung sehr großer Graphen geworfen werden, am Beispiel von *Apache Giraph*, einer Open-Source-Implementierung von *Google Pregel* [Malewicz et al., 2010].

Die oben beschriebenen acht Datenbanksysteme werden, sofern sich die Verwendung als einfach erweist, einem Benchmark unterzogen. Mangels Vergleichbarkeit wird hierauf bei Apache-Giraph verzichtet.

4 Überblick über verschiedene Graphdatenbanksysteme

Im Rahmen des Überblicks sollen die Datenbankmanagementsysteme anhand verschiedener Fragestellungen verglichen werden, wie zum Beispiel:

- Unterstützt das Datenbankmanagementsystem Schemata? Wie werden diese definiert?
- Welche Kommunikationsschnittstellen stehen zur Verfügung?
- Welche Anfragemechanismen mit welcher Mächtigkeit stehen zur Verfügung?
- Handelt es sich um ein eingebettetes Datenbankmanagementsystem?
- Welche Zusicherungen gibt es in Sachen Atomarität, Konsistenz, Isolation und Dauerhaftigkeit nebenläufiger Zugriffe?
- Sofern es sich um verteilbare Datenbanken handelt, welche Abwägungen wurden zwischen Konsistenz, Verfügbarkeit und Partitionstoleranz getroffen?
- Wie werden die Daten außerhalb des Hauptspeichers persistiert?
- Welche Indizierungsmöglichkeiten bestehen?
- Welche Mechanismen garantieren die Isolation nebenläufiger Zugriffe?
- Welche Möglichkeiten gibt es, große Datenmengen unter Umgehung etwaiger Transaktionskontrollen in die Datenbank zu laden?
- Welche Cachingverfahren werden genutzt?
- Gibt es einen Designschwerpunkt bezüglich transaktionaler oder analytischer Datenverarbeitung?

5 Benchmark

In diesem Abschnitt sollen – in einer einheitlichen Testumgebung – Geschwindigkeitskennzahlen für die einzelnen Datenbankmanagementsysteme ermittelt werden. Hierfür soll jedes Datenbanksystem einmalig mit einem sehr großen Graphen gefüllt werden. Im Folgenden sollen verschiedene Graphalgorithmen auf diesen Graphen angewandt werden.

Dominguez-Sal et al. [2011] haben eine sehr vielschichtige Diskussion über das Design von Graphdatenbankbenchmarks publiziert. Sie kategorisieren darin Graphalgorithmen nach verschiedenen Kriterien: Handelt es sich um analysierende oder mutierende Algorithmen? Wird weiter als bis zur Tiefe 1 traversiert wird? Betrachten sie nur einen kleinen Teil oder den kompletten Graphen? Werden Attribute oder lediglich die Struktur des Graphen betrachtet? Welche Art von Ergebnis wird zurückgegeben?

Ciglan et al. [2012] merken hierzu an, dass die Fähigkeit, effizient über die Topologie des Graphen zu traversieren, das Alleinstellungsmerkmal von Graphdatenbanksystemen sei, weshalb dies auch der Schwerpunkt von Graphdatenbank-Benchmarks sein sollte.

Auf dieser Grundlage lassen sich exemplarische Algorithmen finden:

- Das Auslesen transitiv erreichbarer Knoten einer bestimmten Tiefe (sog. „friends-of-friends“-Anfragen) als Beispiel lokal traversierender Anfragen, die Knotenmengen zurückgeben.
- Die Berechnung der Betweenness-Centrality nach Brandes [2001] als Beispiel einer globalen, traversierenden Anfrage, welche eine annotierte Knotenmenge zurückgibt
- das Auslesen des kompletten Graphen, also aller Knoten und Kanten samt eventueller Attribute als Beispiel globaler, nichttraversierender Anfragen.
- Von den Datenbankenmanagementsystemen eventuell unterstützte Formen von Regular Path Queries [Mendelzon und Wood, 1989] und Graph Pattern Matching als Beispiele für global traversierende Anfragen.

Als Datengrundlage scheinen hier – zumindest für die ersten drei Algorithmen – generierte Graphen in verschiedenen Größen sinnvoll zu sein. Diese sollten Eigenschaften haben, wie sie auch in der Realität häufig vorzufinden sind. Chakrabarti und Faloutsos [2006] schreiben, dass solche in der Regel mehrere der folgenden Eigenschaften haben:

- Die Wahrscheinlichkeitsfunktion der Knotengrade x ist $p(x) = \alpha x^{-\gamma}, \alpha > 0, \gamma > 1$, eine sogenannte „Power-Law distribution“.
- Zusammenhängende Teilgraphen haben einen kleinen Durchmesser im Verhältnis zur Kantenmenge.
- Die Graphen haben relativ hohe Clustering-Koeffizienten.

Für Regular Path Queries und Pattern Matching werden Graphen mit Kanten- oder Knotenbeschriftungen benötigt. Hier ist zu überlegen, ob diese ebenfalls generiert werden oder ob hierfür reale Graphen verwendet werden sollten.

Ein besonders kritischer Aspekt im Design von Benchmarks ist, dass durch das von Datenbankmanagementsystemen in der Regel durchgeführte Caching von Daten wiederholte Läufe des selben Algorithmus auf der selben Datenbank häufig massiv unterschiedliche Laufzeiten haben. Vor diesem Hintergrund soll jeder der Algorithmen sowohl mit kalten als auch mit warmen Caches gemessen werden.

Um den Aufwand bei der Erstellung des Benchmarks in einem angemessenen Rahmen zu halten, sind jedoch Abstriche bei einzelnen Aspekten notwendig: So ist es nur unter sehr hohem Aufwand möglich, ein realistisches Lastprofil konkurrierender Zugriffe auf das Datenbanksystem zu simulieren. Dieses Benchmark wird sich daher auf Single-User/Single-Thread-Zugriffe beschränken.

Auch die Verteilung der Datenbank auf mehrere Maschinen soll im Rahmen dieses Benchmarks nicht simuliert werden. Die optimalen Verteilungsstrategien sind hier stets abhängig von den Daten, von den Zugriffsmustern und insbesondere auch von den Stärken und Schwächen des jeweiligen Datenbanksystems. Versuche dieses zu simulieren würden wahrscheinlich der Vergleichbarkeit, Reproduzierbarkeit und Aussagekraft des Benchmarks mehr schaden als nutzen.

Literatur

Amann, B. und Scholl, M. (1992). Gram: a graph data model and query languages. In Lucarella, D., Nanard, J., Nanard, M., und Paolini, P. (Hrsg.), *ECHT '92: Proceedings of the ACM conference on Hypertext*, Seiten 201–211. ACM.

Angles, R. und Gutierrez, C. (2008). Survey of graph database models. *ACM Computing Surveys*, 40(1).

Brandes, U. (2001). A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, 25(2):163–177.

Chakrabarti, D. und Faloutsos, C. (2006). Graph mining: Laws, generators, and algorithms. *ACM Computing Surveys*, 38(1).

Ciglan, M., Averbuch, A., und Hluchy, L. (2012). Benchmarking traversal operations over graph databases. In *proceedings of GDM'12, ICDE 2012 Workshop*. Online, `http://ups.savba.sk/~marek/papers/gdm12-ciglan.pdf`, abgerufen am 08. Juli 2012.

Codd, E. F. (1970). A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–378.

Dominguez-Sal, D., Martinez-Bazan, N., Muntes-Mulero1, V., Baleta, P., , und Larriba-Pey, J. L. (2011). A discussion on the design of graph database benchmarks. In Nambiar, R. und Poess, M. (Hrsg.), *Proceedings of the Second TPC technology conference on Performance evaluation, measurement and characterization of complex systems*, number 6417 in Lecture Notes In Computer Sciences, Seiten 25–40. Transaction Processing Performance Council, Springer-Verlag.

Iordanov, B. (2010). HyperGraphDB: A Generalized Graph Database. In Shen, H. T., Pei, J., Özsu, M. T., Zou, L., Lu, J., Ling, T.-W., Yu, G., Zhuang, Y., und Shao, J. (Hrsg.), *Web-Age Information Management. WAIM 2010 Workshops*, volume 6185 of *Lecture Notes In Computer Science*, Seiten 25–36. Springer-Verlag.

Klyne, G. und Carroll, J. J. (Hrsg.) (2004). *Resource Description Framework (RDF): Concepts and Abstract Syntax*. The World Wide Web Consortium (W3C).

Lin, C. (2003). Object-oriented database systems: A survey. Online. Online, `http://users.soe.ucsc.edu/~lcx/courses/cmps277/cmps277-project.pdf`, abgerufen am 16. Juni 2012.

Malewicz, G., Austern, M. H., Bik, A. J., Dehnert, J. C., Horn, I., Leiser, N., und Czajkowski, G. (2010). Pregel: a system for large-scale graph processing. In *SIGMOD '10: Proceedings of the 2010 international conference on Management of data*, Seiten 135–146, New York, NY, USA. ACM. 405102.

Martínez-Bazan, N., Nin, J., Muntés-Mulero, V., Sánchez-Martínez, M.-A., Gómez-Villamor, S., und Larriba-Pey, J.-L. (2007). DEX: High-performance exploration on large graphs for information retrieval. In *CIKM '07: Proceedings of the sixteenth ACM conference on information and knowledge management*, New York, NY, USA. ACM.

Mendelzon, A. O. und Wood, P. T. (1989). Finding regular simple paths in graph databases. In Apers, P. M. G. und Wiederhold, G. (Hrsg.), *Proceedings of the Fifteenth International Conference on Very Large Data Bases, August 22-25, 1989, Amsterdam, The Netherlands*, Seiten 185–193. Morgan Kaufmann.

Prud'hommeaux, E. und Seaborne, A. (Hrsg.) (2008). *SPARQL Query Language for RDF*. The World Wide Web Consortium (W3C).

Rodriguez, M. A. und Neubauer, P. (2010). The graph traversal pattern. Preprint, `http://arxiv.org/abs/1004.1001v1`, abgerufen am 27. Juni 2012.

Singh, S. K. (2009). *Database Systems: Concepts, Design & Applications*. Prentice Hall.