Konzeption und Umsetzung einer DSL zur Informationsfusion auf verteilten heterogenen Graphen

Alexander Kern[1]

Abstract: Informationsintegration ist das Zusammenführen von Informationen aus verschiedenen Quellen. Dadurch soll eine effektivere Nutzung der Daten erreicht werden, als durch die Arbeit mit den einzelnen Quellen möglich ist. Allerdings ist Informationsintegration ein hochkomplexes Problem. Es umfasst neben der Duplikatserkennung auch das Auflösen von Inkonsistenzen auf Schema- und Instanzlevel. Diese Arbeit stellt eine domänenspezifische Sprache zur Lösung von Konflikten auf Attributwertebene für heterogene Graphdaten vor. Die Sprache stellt mit der Informationsfusion einen Teilschritt des Informationsintegrationsprozesses zur Verfügung. Neben der Gestaltung der DSL und der Entwicklung eines Prototyps mit Apache Flink und Gradoop beurteilt eine Evaluation der Fusionsergebnisse die Qualität des Verfahrens.

Keywords: Informationsintegration, Informationsfusion, Gradoop, Graphen, DSL, Apache Flink

1 Einleitung

Die Verarbeitung großer Datenmengen ist eine immer wichtigere Aufgabe in vielen verschiedenen gesellschaftlichen Bereichen. Die Arbeit mit Big Data bringt jedoch eine Vielzahl von Schwierigkeiten mit sich. Neben der großen Menge an Daten, die oft nicht mehr an einem einzelnen Rechner bearbeitet werden können, sind auch verschiedene Datenformate, fehlende oder fehlerhafte Daten und die Anforderung nach zeitnaher Datenverarbeitung Probleme, die gelöst werden müssen. Eine weitere Herausforderung ist es, eine geeignete Darstellung für die gesammelten Daten zu finden. Eine mögliche Darstellungsart für Datenobjekte mit komplexen Beziehungen sind Graphen, denn diese ermöglichen eine intuitive Abbildung und Analyse der vorhandenen Daten[JP16]. Graphen stellen Entitäten als Knoten und Beziehungen zwischen diesen Entitäten als Kanten dar.

Eine wichtige Aufgabe ist das Zusammenführen von Daten aus unterschiedlichen Quellen. Hierbei spricht man von Informationsintegration. Dadurch ist es möglich, in einzelnen Quellen fehlende Daten zu ergänzen, Fehler zu finden und beheben und durch weitergehende Analysen zusätzliche Erkenntnisse, auch über Datenquellengrenzen hinweg, zu erlangen[Li10, S. 266]. Allerdings ist Informationsintegration ein komplexes Problem, dass

[1] Universität Leipzig, Big Data Kompetenzzentrum, Ritterstraße 9-13, 04109 Leipzig, ak44xubu@studserv.uni-leipzig.de

doi:10.18420/btw2019-ws-19

unterschiedliche Teilaspekte wie Transformationen, Schemaintegration, Entity Resolution, Duplikatserkennung und Datenfusion umfasst[LN06, S. 6ff].

Diese Arbeit stellt eine Lösung für Attributwertkonflikte bei der Fusion von Daten in einem Prozess zur Informationsintegration vor. Die Lösung nutzt eine domänenspezifische Sprache zur Auswahl von Attributen aus heterogenen Graphdaten und Lösungsstrategien für Konflikte zwischen den vorhandenen Attributwerten. Mithilfe der DSL lässt sich das Schema und die Logik zur Berechnung der Ausgabeknoten festlegen.

2 Verwandte Arbeiten

Im Paper „Declarativ Data Cleaning: Language, Model, and Algorithms“[GFS01] stellen Galhardas et. al. eine an die SQL-Syntax angelehnte deklarative Sprache vor, die durch mehrere Schritte unsaubere Daten aufbereitet. Unsauberkeiten können Fehler, Inkonsistenzen und inkompatible Schemaunterschiede sein. Das Ziel ist dabei eine klare Trennung der logischen Beschreibung mithilfe der Querysprache und der physischen Ausführung durch das Programm. Dabei werden die Queries, welche die einzelnen Schritte beschreiben, zu Java-Programmen umgewandelt. Nach dem Laden und Transformieren der Daten findet Entity Resolution, das Auffinden von verschiedenen Datensätzen, die eine Entität beschreiben, statt. Der letzte Schritt ist das Merging, in dem benutzerdefinierte Aggregationsfunktionen die vorhandenen Attributwerte zu einem einzelnen Wert zusammenführen. Der im Paper vorgestellte Prozess legt den Schwerpunkt auf das Aufbereiten von unsauberen Daten. Eventuelle Heterogenität im Datenschema beseitigt der Prozess beim Laden, weshalb das Merging auf homogene Daten beschränkt ist.

Einen ähnlichen Prozess zur Informationsintegration von heterogenen Graphdaten stellen Lim et. al. in [Li10] vor. Der erste Schritt in ihrem Prozess ist die Schemaintegration. Dabei werden Daten aus den verschiedenen Quellen zuerst in ein einheitliches Schema gebracht. Anschließend werden beim Instance Matching Entitäten und Relationen einander zugeordnet. Am Ende findet die Auflösung von Attribute-Value-Konflikten statt. Die Veröffentlichung betrachtet nur den Schritt des Instance Matching im Detail.

In „Declarative Data Cleaning With Conflict Resolution“[NH02] konzentrieren sich Felix Naumann und Matthias Häussler auf die Lösung von Attribute-Value-Konflikten. Sie verfolgen einen Ansatz um Datenintegration mithilfe von SQL durchzuführen. Im Paper nennen die Autoren eine Vielzahl von möglichen Funktionen zur Auflösung von Attributwertkonflikten.

Gradoop (Graph Analytics on Hadoop) ist ein Framework für verteilte deklarative Graphanalysen. Es verbindet die Stärken von Graphdatenbanksystemen wie Neo4j und verteilten Graph Processing Systeme wie Google Pregel oder Gelly, in dem es die verteilte Ausführung von Graphabfragen und -algorithmen für EPGM-Graphen bietet. Das Extended Property Graph Model beschreibt Graphen mithilfe von Knoten, Kanten, Graphen und Graph Collections. Jedes Objekt im EPGM-Graph kann dabei durch **Properties** mit Eigenschaften als

Key-Value-Paaren versehen werden.[Ju18] FAMER ist ein Framework für verteilte Entity Resolution auf Daten aus verschiedenen Quellen. Zur Ausführung setzt FAMER Apache Flink und Gradoop ein[SPR17].

3 Konzeption

Das folgende Kapitel erläutert den geplanten Informationsintegrationsprozess und die daraus resultierenden Anforderungen an die domänenspezifische Sprache zur Informationsfusion.

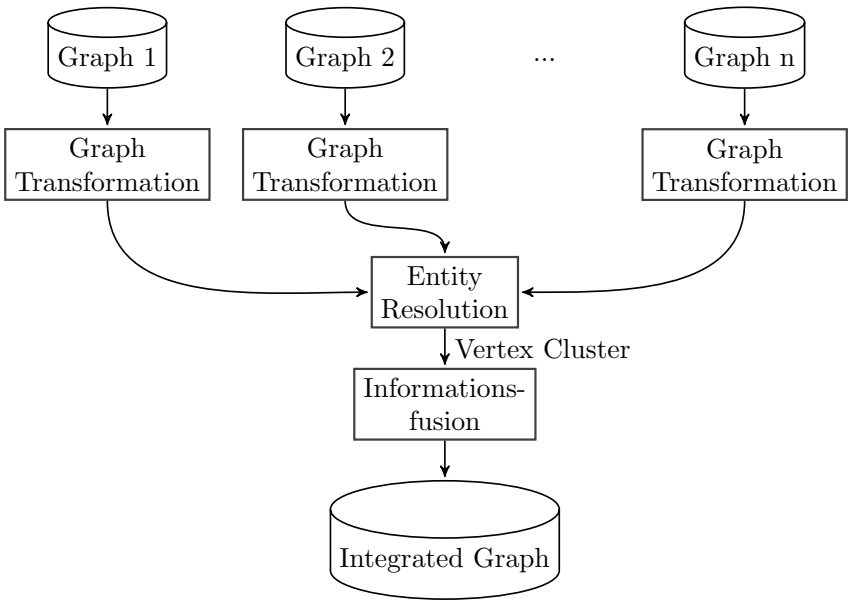

Abb. 1: Informationsintegrationsprozess mit Gradoop

Abb. 1 zeigt den Ablauf der Informationsintegration mithilfe von Gradoop. Mithilfe von Transformationen kann eine Vorverarbeitung der Daten stattfinden. Eine Angleichung der Datenschemata ist hierbei allerdings nicht zwingend nötig. Die Unterstützung heterogener Daten ist ein essentieller Unterschied zu den anderen vorgestellten Arbeiten. Anschließend findet Entity Resolution mithilfe von FAMER statt. Die dabei gematchten Datensätze beschreiben jeweils die gleiche Entität. Jeder dieser Matching-Cluster wird anschließend mithilfe der von der DSL beschriebenen Query zu einem einzelnen Knoten fusioniert. Der Ergebnisgraph umfasst alle diese generierten Knoten. Eine Fusion eventuell vorhandener Kanten findet derzeit nicht statt.

3.1 Anforderungen an die DSL

Die Hauptaufgaben der DSL sind das Auswählen der Attribute aus den heterogenen Eingangsdaten, die Definition des Schemas und die Lösung von Attributwertkonflikten in den Ausgabeknoten. Die DSL beschränkt sich derzeit auf die Informationsfusion der Knoten der Graphen, die Fusion von Relationen ist nicht Teil der Problemstellung.

Auf der obersten Ebene muss die DSL im Stande sein, einen oder mehrere Ausgabeknotentypen zu definieren. Für jeden Ausgabeknoten müssen Transformationsregeln festlegen, welche Attribute der Eingabedaten mit welcher Konfliktlösungsstrategie (vgl. Abschnitt 3.2) betrachtet werden sollen. Falls die gewählte Strategie zusätzliche Optionen benötigt, muss die Sprache diese bei der Regeldefinition akzeptieren können. Die Menge der Transformationsregeln beschreibt gleichzeitig das Schema des Ausgabeknotens.

Außerdem soll es möglich sein, weitere vorhandene Attribute ohne explizite Transformationsregeln zu übernehmen. Dies soll nicht das Standardverhalten sein, jedoch zur möglichen Verkürzung von Queries einsetzbar sein. Die Idee hierbei ist, ohne großen Aufwand einen ersten Überblick über die Daten zu gewinnen, den der Nutzer anschließend mit spezifischen Transformationsregeln verfeinern kann.

Zur Vereinfachung von Queries soll es außerdem eine Kurzform für die Listen von Eingabeattributen geben, wenn diese in allen Quellen den gleichen Namen haben. Diese Voraussetzung lässt sich durch Graphtransformationen leicht herstellen.

Die Endanwender der DSL sind einerseits Nutzer von Gradoop, bei denen von vorhandenen Java- und SQL-Kenntnissen ausgegangen werden kann. Andererseits soll die DSL auch von Domänenexperten, beispielsweise aus dem Business-Intelligence-Bereich, nutzbar sein. Bei diesen kann zwar nicht von Programmierkenntnissen, allerdings auch von SQL-Kenntnissen ausgegangen werden.

Ein wichtiger Aspekt für die Gestaltung einer domänenspezifische Sprache sind die Kosten für Programmerstellung, -verifikation und -wartung[Ba17]. Um diese gering zu halten, ist es nötig, die Problemdomäne so abzubilden, dass die Anwendung der Sprache dem Nutzer möglichst klar erscheint. Aber auch Punkte wie die Komplexität der Syntax sind hierbei von Bedeutung. Unter Beachtung der Endnutzer der Sprache sollen diese Faktoren beim Design beachtet werden.

3.2 Attribute Value Conflict Resolution

Unter Betrachtung der von Naumann und Häussler vorgestellten Strategien zur Lösung von Attributwertkonflikten [NH02] wurden die Tab. 1 gezeigten Konfliktlösungsstrategien entwickelt.

Diese Konflikte können aus unterschiedlichen Datenschemata (zum Beispiel bei Datumsangaben), verschiedenen Semantiken in den unterschiedlichen Quellen (zum Beispiel bezeichnet `name` einmal den gesamten Namen einer Person, einmal nur den Nachnamen), inhaltlichen Fehlern (zum Beispiel Tippfehler oder veraltete Daten) oder unterschiedlicher Schemata der Quelldaten stammen.

Einfache Strategien sind beispielsweise das Wählen des ersten Werts in der alphanumerischen Ordnung und das Konkatenieren aller vorhandenen Werte durch ein Separatorsymbol. Die

Strategie	Beschreibung	Optionen
Straight	Erster Nicht-Null-Wert nach alphanumerischer Ordnung	-
Retain	Konkatenation der vorhandenen Werte	Separatorsymbol
Priority	Erster Nicht-Null-Wert in der spezifizierten Ordnung der Quellen	Sortierung der Quellen
Newest	Wert mit dem aktuellsten Zeitstempel	Zeitstempel-Attribute
Majority	Wert, der in den meisten Quellen vorhanden ist	-
Source	Wert mit dem höchsten summierten Gewicht	Gewichte für Quellen
Property	Auswahl anhand der vorhandenen Attributwerte mit Substrategie	Substrategie

Tab. 1: Strategien zur Konfliktlösung

priority-Strategie nutzt eine Ordnung für die Quellen, mit der ein Anwender Präferenzen ausdrücken kann. Verfügen die Datensätze über Zeitstempel ist es möglich, den neuesten bzw. zuletzt geänderten Wert auszuwählen.

Die **majority**-Strategie ermöglicht die Auswahl des am häufigsten vorhandenen Werts. Eine Verfeinerung dazu ist die **source**-Strategie, die den Quellen zusätzlich Gewichte gibt, um so detailliertere Präferenzen auszudrücken. Bei Gleichstand soll wie bei der **straight**-Strategie der erste Wert in der alphanumerischen Ordnung gewählt werden.

Strategien basierend auf den vorhandenen Attributwerten sind in der **property**-Strategie gebündelt. Diese lassen sich in zwei Klassen aufteilen: Strategien für Strings und Strategien für Zahlenwerte. Für Strings steht die Auswahl des kürzesten, längsten oder des längsten gemeinsamen Teilstrings (LCS) zur Verfügung. Aus Zahlwerten können Minimum, Maximum, Mittelwert oder der Median gebildet werden.

4 Design der DSL

Basierend auf den Anforderungen aus dem vorigen Abschnitt entstand eine prototypische Implementierung einer DSL und der dazugehörigen Ausführungsengine. Zur Umsetzung der Sprache dient das Tool ANTLR[2], das aus Grammatiken in EBNF[3]-ähnlicher Form Lexer und Parser erzeugen kann. Das Programm parst die Queries zu Java-Objekten, mithilfe derer die Ausführungsengine durch den Einsatz von Apache Flink und Gradoop die Informationsfusion für die vorhandenen Daten ausführen kann.

Barišić stellt in ihrer Doktorarbeit[Ba17] einen iterativen Prozess zum Design von DSLs hinsichtlich der im vorigen Kapitel genannten Qualitätsmerkmale vor. Er beginnt mit der Abbildung der Problemdomäne, geht weiter über das konkrete Gestalten und Implementieren der Sprache hin zur Evaluation durch Endnutzerfeedback oder Metriken wie Effektivität

[2] https://antlr.org (07.08.2018)
[3] https://www.ics.uci.edu/ pattis/ICS-33/lectures/ebnf.pdf (07.11.2018)

im Vergleich zum Arbeitsprozess ohne DSL. Sind die Ergebnisse der Evaluation nicht zufriedenstellend, muss der Sprachentwickler die vorigen Schritte nach Bedarf wiederholen.

Insgesamt wurden vier funktional identische Sprachprototypen entwickelt und hinsichtlich ihrer Nutzerfreundlichkeit evaluiert. Der erste Sprachansatz nutzt die Ergebnisse von Galhardas et. al.[GFS01] und erweitert ihre Sprache um Unterstützung für heterogene Schemata und die vorgestellten Strategien und mögliche Optionen. List. 1 zeigt das Beispiel aus dem Paper in der DSL.

```
CREATE MERGING
LET Author
  name = property(
    (Source1.author.name, Source2.author.full_name),
    textual:longest),
  authorKey = straight(key)
```

List. 1: SQL-basierte DSL

Das Beispiel zeigt die wichtigsten Eigenschaften der Sprache. Sie generiert einen Ausgabeknoten mit dem Label `Author`, der zwei Attribute `name` und `authorKey` erhält. Die Berechnung der Attributwerte für die erzeugten Knoten erfolgt durch zwei verschiedene Strategien. Die erste Regel zeigt die Auswahl von Attributen aus heterogenen Quellen anhand des Quellennamens, Knotentyps und Attributnamens, sowie die zusätzliche Angabe einer Option für die **property**-Strategie. Die zweite Regel zur Wahl des `authorKey` ähnelt in der verkürzten Form wiederum stark der von Galhardas vorgeschlagenen Sprache. Ausführliche Queries umfassen weitere Ausgabeknoten mit mehr Transformationsregeln.

```
node Author {
  property name {
    Source1.author.name
    Source2.author.full_name
  } strategy property { textual:longest }

  property authorKey { key } strategy straight
}
```

List. 2: Kotlin-inspirierte DSL

Ein anderer Designansatz orientiert sich am Design von DSLs, die mithilfe der Programmiersprache Kotlin erstellt werden können. Diese DSL nutzt Schlüsselwörter und Blöcke um Beschreibungen näher an natürlicher Sprache zu erreichen. List. 2 zeigt die Beispielquery in der Sprache.

Die dritte DSL ist ähnlich zur SQL-basierten DSL aufgebaut und unterscheidet sich vor allem in kleinen Details wie der Definition von Attributlisten. Der vierte Ansatz nutzt statt

einer mit ANTLR generierten Sprache YAML. Die Beschreibung der Informationsfusion findet mithilfe einer YAML-Datei statt, deren Schema wohldefiniert ist.

Die letzte vorgestellte Sprache ist sowohl was die textuelle Länge der Queries als auch den Komfort der Queryerstellung angeht den anderen Sprachen unterlegen. Das manuelle Schreiben von YAML ist zwar einfacher als in anderen Auszeichnungssprachen, trotzdem deutlich anspruchsvoller als die Nutzung der anderen DSLs. Die drei weiteren Sprachen sind sich sehr ähnlich. Die Vorteile der weiten Verbreitung von SQL nennen bereits [GFS01] und [NH02]. Unter Berücksichtigung der Annahme, dass die Endnutzer bereits über SQL-Erfahrungen verfügen, fällt die Entscheidung für diese DSL. Die geringe Einstiegshürde durch Vorerfahrung und die relativ bekannte Syntax ermöglichen es, Queries in der DSL sowohl schnell erstellen als auch verstehen zu können.

Durch die Trennung der Sprache von der Ausführungsengine ist es allerdings jederzeit möglich, die Sprache zu ändern, auszutauschen oder mehrere DSLs parallel zu nutzen. Genau so kann das Backend ersetzt werden, um andere Datenquellen neben Graphdaten in Gradoop zu unterstützen, um die Sprache beispielsweise für Konfliktlösung bei relationalen Daten zu nutzen.

5 Evaluation

Um die Laufzeit und Qualität der Informationsfusion zu messen, wird diese auf den MusicBrainz-Datensatz angewandt. Er besteht aus echten MusicBrainz-Daten und mithilfe des DAPO-Datengenerators[Hi18] erzeugten Duplikaten. Der Datensatz umfasst 10780 Knoten in 4611 Clustern. Daraus entsteht ein Graph mit 15391 Knoten und 19767 Ähnlichkeitskanten. Zur Messung der Matching-Qualität kommt FAMER zum Einsatz. Die Berechnungen finden auf dem Galaxy-Cluster[4] mit Gradoop 0.4.0 und Apache Flink 1.5.0 statt.

5.1 Matching-Qualiät

Die Qualität des Matchings soll mit den Maßen Recall, Precision und F-Measure bestimmt werden. Dafür werden aus den bereits geclusterten Ausgangsdaten mithilfe der Informationsfusion die Ergebnisknoten erzeugt und zu dem Graph der Ausgangsdaten hinzugefügt. Daraus lässt sich leicht der Referenzgraph mit den gegebenen Ähnlichkeitskanten bestimmen. Anschließend werden die Informationen über die bisherige Clusterzugehörigkeit entfernt und ein neues Linking mithilfe von FAMER berechnet. Die dabei entstehenden Ähnlichkeitskanten können mit dem Referenzgraph verglichen werden. Clustering wird nicht eingesetzt. Als grobe Vergleichswerte dienen hierbei die Matching-Ergebnisse aus dem Evaluationspaper von FAMER [SPR17]. Im Paper erreicht das Matching nach Linking und

[4] https://www.urz.uni-leipzig.de/fue/sc/galaxy/ (06.12.2018)

Clustering für den MusicBrainz-Datensatz Werte im Bereich [0.5, 0.7] für das F-Measure. Die Linking-Konfiguration entspricht der aus dem Paper, d.h. Blocking nach `album` und 3-Gramme als Ähnlichkeitsfunktion. Clustering kommt nicht zum Einsatz, alle weiteren Einstellungen entsprechen gegebenen Standardwerten.

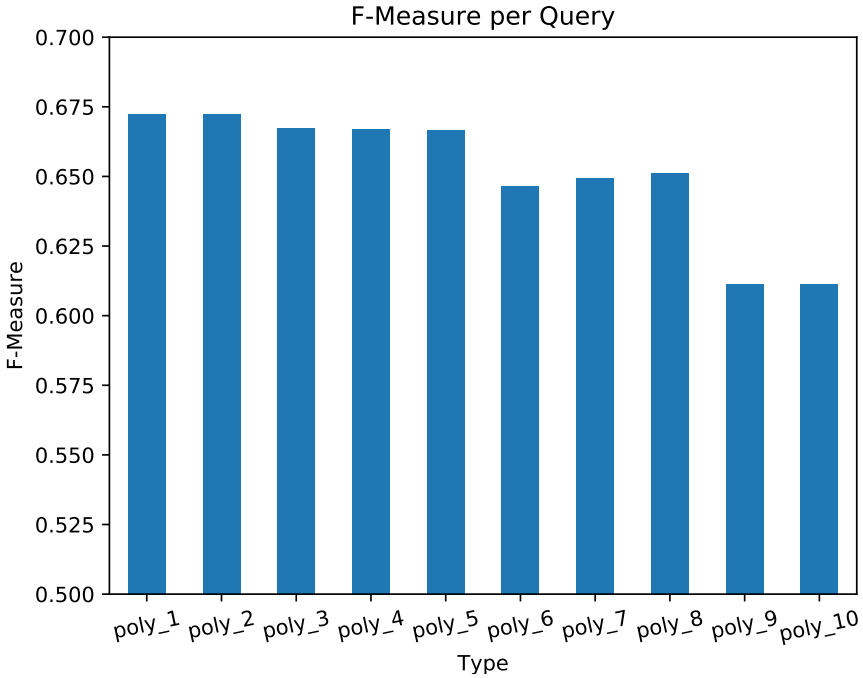

Abb. 2: Ergebnisse der Poly-Strategie-Queries

Abb. 2 zeigt die Ergebnisse von Queries, die verschiedene Strategiekombinationen einsetzen. Die meist relativ ähnlichen Ergebnisse für die einzelnen Strategien zeigen, dass das Verfahren flexibel einsetzbar ist. Da die Auswahl der Strategien letztlich abhängig von den gegebenen Daten und dem Kontext und Ziel der Informationsfusion abhängt, ist es vorteilhaft, dass die Strategien frei kombinierbar sind. Durch bessere Konfigurationen von FAMER lassen sich die Ergebnisse deutlich verbessern.

5.2 Laufzeitmessung

Zur Berechnung der Laufzeit werden die im vorigen Abschnitt vorgestellten Queries jeweils 100 mal auf dem Cluster ausgeführt. Apache Flink führt Transformationen erst aus, wenn eine Senke definiert ist. Deshalb findet die Messung innerhalb des Java-Programms um die gesamte Job-Ausführung inklusive IO-Operationen statt. Abb. 3 zeigt die gemessenen Laufzeit für eine beispielhafte Query mit verschiedenen eingesetzten Strategien.

Es zeigen sich jeweils zwei Gruppen von Laufzeiten (bei ca. 10750ms und 12750ms Laufzeit) um die sich die weiteren gemessenen Werte in einem Bereich von etwa 750ms verteilen, was einer Abweichung von etwa 6.9% bzw. 5.8% entspricht.

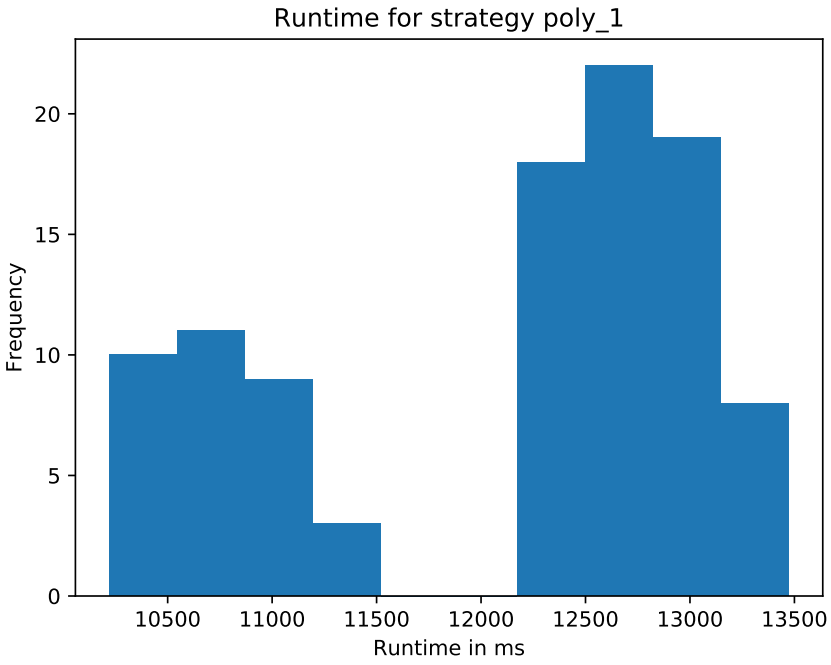

Abb. 3: Laufzeitmessung

Das Verfahren skaliert durch die Verteilung der einzelnen Cluster. Die maximale Clustergröße entspricht theoretisch der Anzahl an Quellen und ist damit eher gering, die Anzahl der Cluster ist idealerweise die Anzahl an Entitäten und dementsprechend groß. Da die parallele Berechnung der einzelnen Cluster unabhängig verteilbar ist, sollte das Verfahren gut skalieren.

6 Zusammenfassung und Ausblick

Im Rahmen dieses Beitrags wurde eine domänenspezifische Sprache zur Informationsfusion von heterogenen Graphdaten vorgestellt. Die Sprache stellt einen Teil der Informationsintegration mit Gradoop zur Verfügung. Nutzer können mit ihr definieren, wie Attributwertkonflikte zwischen vorhandenen Werten gelöst werden sollen. Mithilfe von Apache Flink und Gradoop findet eine verteilte Ausführung der Informationsfusion statt. Laufzeit und Qualität des Verfahrens wurden außerdem in Abschnitt 5 evaluiert.

Weitergehende Aufgaben sind eine ausführliche Evaluation mit Endnutzern der Sprache und die Erweiterung der DSL auf Graphkanten. Ein weiterer interessanter Punkt ist die Auswahl von Standardstrategien anhand der gegebenen Datenschemata. Auch die Nutzbarkeit für andere Anwendungsfälle sowie mögliche Implementierungen geeigneter zusätzlicher Strategien können in Zukunft untersucht werden.

Acknowledgements:

Die vorliegende Arbeit wurde teilweise gefördert durch das Bundesministerium für Bildung und Forschung innerhalb des Competence Center for Scalable Data Services and Solutions (ScaDS) Dresden/Leipzig (BMBF 01IS14014B). Betreut wurde die Arbeit von Matthias Kricke[5] und Eric Peukert[6]. Berechnungen für diese Arbeit wurden mit Ressourcen des Rechenzentrums der Universität Leipzig durchgeführt.

Literatur

[Ba17] Barisic, A.: Usability Evaluation of Domain-Specific Languages, Diss., Universidade Nova De Lisboa, Dez. 2017.

[GFS01] Galhardas, H.; Florescu, D.; Shasha, D.: Declarative Data Cleaning: Language, Model, and Algorithms. In: In VLDB. S. 371–380, 2001.

[Hi18] Hildebrandt, K.; Panse, F.; Wilcke, N.; Ritter, N.: Large-Scale Data Pollution with Apache Spark. IEEE Transactions on Big Data/, 2018, ISSN: 2332-7790.

[JP16] Junghanns, M.; Petermann, A.: Verteilte Graphanalysen mit Gradoop. JavaSPEKTRUM 5/, Abgerufen am 26.07.2018, 2016, URL: https://www.sigs-datacom.de/uploads/tx_dmjournals/junghans_petermann_JS_05_16_eeNZ.pdf.

[Ju18] Junghanns, M.; Kießling, M.; Teichmann, N.; Gómez, K.; Petermann, A.; Rahm, E.: Declarative And Distributed Graph Analytics With GRADOOP. In: PVLDB. Bd. 11. 12, 2018.

[Li10] Lim, E. P.; Sun, A.; Datta, A.; Kuiyu, C.: Information Integration for Graph Databases. In: Link Mining: Models, Algorithms, and Applications. Research Collection School Of Information Systems, Kap. 10, S. 265–281, 2010.

[LN06] Leser, U.; Naumann, F.: Informationsintegration: Architekturen und Methoden zur Integration verteilter und heterogener Datenquellen. dpunkt.verlag GmbH, 2006, ISBN: 978-3898644006.

[NH02] Naumann, F.; Häussler, M.: Declarative Data Merging With Conflict Resolution. In: International Conference on Information Quality (IQ 2002). 2002. S. 212–224, 2002.

[SPR17] Saeedi, A.; Peukert, E.; Rahm, E.: Comparative Evaluation of Distributed Clustering Schemes for Multi-source Entity Resolution. In: Advances in Databases and Information Systems - 21st European Conference, ADBIS 2017, Nicosia, Cyprus, September 24-27, 2017, Proceedings. S. 278–293, 2017, URL: https://doi.org/10.1007/978-3-319-66917-5%5C_19.

[5] kricke@informatik.uni-leipzig.de

[6] peukert@informatik.uni-leipzig.de