Exposé eines Promotionsvorhabens: Konfluenz, Effizienz und Effektivität eines universellen CHASE-Verfahrens

Andreas Görres

Lehrstuhl für Datenbank- und Informationssysteme
Institut für Informatik
Universität Rostock
andreas.goerres@uni-rostock.de

17. März 2021

Im Promotionsvorhaben sollen verschiedene Techniken des CHASE vereinheitlicht werden, um Fragestellungen der Privacy, Provenance und Anfrageoptimierung mit derselben Grundtechnik lösen zu können. Darüber hinaus soll der CHASE erweitert werden, um effektiv auf praktisch relevante Fragestellungen anwendbar zu sein. Der negative Einfluss dieser Erweiterungen auf Terminierung, Konfluenz und Effizienz des Verfahrens soll hierbei verhindert werden.

1 Anwendungsbereich

Ein zunehmend relevant gewordenes Thema der Datenbankforschung ist die Auswertung großer, strukturierter Datenmengen, wie sie beispielsweise im Rahmen des *Internet of Things* generiert werden. Am Lehrstuhl für Datenbank- und Informationssysteme wird diese Problematik hauptsächlich anhand zweier Anwendungsfälle untersucht: Forschungsdatenmanagement und intelligente Assistenzsysteme.

Ansatzpunkt für das Auswerten und Managen von Forschungsdaten wissenschaftlicher Institute bildet die langjährige Zusammenarbeit des Lehrstuhls mit dem Institut für Ostseeforschung (IOW). Einerseits liefert diese Kooperation Einsichten darüber, welche (z.B. statistische) Operationen praktisch relevant für ein Datenbankmanagementsystem sind, andererseits lässt sich anhand historischer Daten die Schemaevolution von Forschungsdatenbanken untersuchen. Veröffentlichte Auswertungen sollen sich trotz struktureller Veränderungen auf

die beteiligten Originaldaten zurück führen lassen, wodurch Reproduzierbarkeit und zum Teil sogar Replizierbarkeit der Ergebnisse gewährleistet ist.

Im Zusammenhang mit der Aktivitäts- und Intentionserkennung smarter Assistenzsystemen werden insbesondere die Aspekte Privacy und Effizienz näher beleuchtet. Wenn durchzuführende Berechnungen bereits bei der Datenerfassung erfolgen, wird unter Umständen eine ineffizient zentrale Zusammenführung und Speicherung der Daten, welche das Prinzip der Datensparsamkeit verletzt, nicht mehr benötigt.

Für die genannten Data-Science-Anwendungen lassen sich also die drei Rahmenbedingungen Privacy, Provenance und Optimierung definieren:

- Privacy: Kann man automatisiert eine zentrale Datenauswertung in eine vollständig dezentrale Datenauswertung transformieren?
- Provenance: Wie kann man automatisch eine abstrahierte, anonymisierte Sicht auf die minimal erforderlichen Originaldaten generieren?
- Optimierung der Effizienz: Kann man aufwändige Basisoperationen in Auswertungsverfahren identifizieren und dann insofern minimieren, dass die komplexe Auswertung automatisch (bei garantiert identischem Auswertungsergebnis) in einen weit effizienteren Algorithmus transformiert wird?

Es wäre wünschenswert, wenn Privacy und Provenance in das Datenauswertungsverfahren integriert und dann automatisch optimiert werden könnten. Grundsätzlich stellen Privacy und Provenance jedoch gegensätzliche Anforderungen an das Auswertungsverfahren und sind daher nicht leicht kombinierbar. Sollen die Rahmenbedingungen unabhängig voneinander erfüllt werden, existieren bereits Speziallösungen. Prinzipiell können die Teilprobleme aber auch mit dem CHASE, einem universellen Algorithmus der Datenbanktheorie, bearbeitet werden. Im Promotionsvorhaben sollen die entsprechenden Teilprobleme für den CHASE formalisiert werden, um sie dann gemeinsam mit einem einheitlichen Algorithmus zu lösen.

2 Problemstellung

Betrachtet man die Problematik von einer abstrakten Ebene her, sollen im Promotionsvorhaben formalisierte Rahmenbedingungen in auf gleicher Weise formalisierte Auswertungstechniken integriert werden. Auf die konkreten, zuvor beschriebenen Anwendungsbereiche bezogen bedeutet dies, Privacy und Provenance in Datenauswertungsverfahren zu integrieren, indem über die einheitliche

Grundtechnik des CHASE ursprünglich getrennte Problemlösungen zu einer Gesamtlösung kombiniert werden.

Hieraus leiten sich folgende zwei Herausforderungen ab:

- Vereinheitlichung der CHASE Techniken
- Kontrollierte Erweiterung des CHASE

Vereinheitlichung der CHASE Techniken: Obwohl der CHASE – von einem abstrakten Gesichtspunkt betrachtet – ein universaler Algorithmus ist, der unterschiedliche Arten von Parametern verarbeiten und so in einer Vielzahl von Aufgabengebieten Verwendung finden könnte, gehen praktische Implementierungen des Algorithmus bisher in der Regel von einzelnen Anwendungsfällen aus (z.B. Datenbereinigung oder Anfrageoptimierung). Voraussetzung für die folgenden Arbeitsschritte ist, die unterschiedlichen konkreten Ausführungen des CHASE in einem Gesamtmodell des Algorithmus zu kombinieren, damit die oben genannten Fragestellungen der Privacy, Provenance und Anfrageoptimierung mit derselben Grundtechnik einheitlich und in abgestimmter Kombination zueinander gelöst werden können.

Erweiterung des CHASE: Obwohl der CHASE in der Datenbanktheorie ein ungeheuer mächtiges Werkzeug darstellt, ist er bisher für praktische Erfordernisse von Datenbankanwendungen inadäquat. Tatsächlich beschränkt sich der klassische Standard-CHASE auf die Abbildung von Konjunktionen positiver Selektions-Projektions-Verbund-Beziehungen, wobei Selektion lediglich auf dem Testen von Gleichheit beruht. Um Effektivität in den zuvor genannten Anwendungsgebieten zu erreichen, wird eine Erweiterung um weitere Vergleichsoperatoren, Negation und Funktionen benötigt. Diese Erweiterung sollte jedoch kontrolliert erfolgen, um Terminierung, Konfluenz und Effizienz des Verfahrens nicht zu gefährden. Tatsächlich handelt es sich beim in dieser Arbeit betrachteten CHASE-Algorithmus bereits um eine Erweiterung des ursprünglichen Tableau-Verfahrens – auch diese Erweiterungen gingen mit Einschränkungen der ursprünglich garantierten Terminierung und Konfluenz einher.

3 Stand der Forschung

3.1 Der CHASE

Zahlreiche grundlegende Anwendungsfälle der Datenbankforschung haben gemeinsam, dass sie mit Hilfe eines Universalmodells gelöst werden können [DNR08]. Soll beispielsweise eine Quelldatenbank unter Berücksichtigung von Integritätsbedingungen in eine Zieldatenbank überführt werden, so kann es zahlreiche mögliche Lösungen des Problems geben, welche die Integritätsbedingungen erfüllen.

Aus einigen dieser Zieldatenbanken – den sogenannten universellen Lösungen – lassen sich alle anderen Lösungen des Problems durch homomorphe Abbildungen gewinnen. Stellt man (boolesche) Datenbankanfragen an diese Universallösung, so erhält man genau für die Anfragen positive Antworten (die sogenannten sicheren Antworten), die auch für jede andere mögliche Variante der Zieldatenbank eine positive Antwort geliefert hätten. Es liegt also nahe, eine Universallösung als Lösung des Datenaustauschproblems zu wählen und zu materialisieren. Universallösungen lassen sich direkt aus dem Universalmodell ableiten, welche durch den CHASE berechnet werden können. Allerdings handelt es sich bei der Standard-Variante des CHASE (im Gegensatz zu einer komplizierteren Spezialform, dem Core-CHASE) um keinen vollständigen Algorithmus, um universelle Modelle zu erzeugen. In der weiteren Arbeit werden wir jedoch über diese Unvollkommenheit des Standard-CHASE (also die Existenz universeller Modelle, die dieser CHASE nicht finden kann) hinwegsehen.

Im Folgenden soll der grundlegende Ablauf eines CHASE-Schrittes betrachtet werden. Hierfür müssen wir zunächst einige Begrifflichkeiten klären. Der CHASE arbeitet, allgemein gesehen, Parameter in Objekte ein. Sowohl Parameter als auch Objekte sollten als prädikatenlogische Formeln erster Ordnung (ohne Funktionssymbole oder Negationen) darstellbar sein. Ohne Verlust der Allgemeinheit werden wir die Parameter als Integritätsbedingungen und das Objekt als Datenbankinstanz bezeichnen – tatsächlich könnte die Instanz jedoch z.B. auch den Körper einer Datenbankanfrage darstellen (die sogenannte „kanonische“ Instanz der Anfrage). Integritätsbedingungen lassen sich in tupelerzeugende Abhängigkeiten (TGDs) und gleichheiterzeugende Abhängigkeiten (EGDs) unterteilen:

$$TGD : \forall X, Y : \phi(X, Y) \rightarrow \exists Z : \psi(X, Z)$$
$$EGD : \forall X : \phi(X) \rightarrow x_1 = x_2.$$

ϕ und ψ sind hierbei Mengen relationaler Atome, X, Y und Z sind jeweils Mengen allquantifizierter Variablen und Konstanten, Z ist eine Menge existenzquantifizierter Variablen, und x_1 sowie x_2 sind Variablen oder Konstanten aus X. Kommen keine existenzquantifizierten Variablen im Kopf der TGD vor, so sprechen wir von einer vollen Abhängigkeit, ansonsten von einer eingebetteten Abhängigkeit. Es gibt prinzipiell auch Integritätsbedingungen, die TGDs und EGDs kombinieren (z.B. Unabhängigkeitsatome, [HL18]), wir wollen uns jedoch an dieser Stelle nicht weiter mit derartigen Sonderfällen auseinander setzen.

Durch den Rumpf der Integritätsbedingung wird ein Muster der Instanz definiert. Erfüllt ein Tupel der Instanz (oder ein Atom eines Anfragerumpfes) das Muster, lässt sich ein Homomorphismus von den Variablen der Bedingung zu

den Konstanten und Nullwerten der Instanz (bzw. zu den all- und existenzquantifizierten Variablen des Anfragerumpfes) finden. In diesem Fall sprechen wir auch vom Vorliegen eines *Triggers*. Der Trigger ist aktiv, wenn die Bedingung noch nicht erfüllt ist. Für TGDs testen wir hierfür, ob der Homomorphismus so für die existenzquantifizierten Variablen des TGD-Kopfes erweitert werden kann, dass das Bild des TGD-Kopfes unter dem erweiterten Homomorphismus bereits in der Datenbank existiert. Ist dies nicht der Fall, wird der Homomorphismus stattdessen so erweitert, dass die existenzquantifizierten Variablen auf neu erzeugte markierte Nullwerte abgebildet sind. Die so definierten neuen Tupel werden in der Datenbankinstanz materialisiert. Für EGDs wird überprüft, ob das Gleichheitsatom $x_1 = x_2$ bereits erfüllt ist, also ob das Bild beider Terme unter dem Homomorphismus identisch ist. Ist dem nicht der Fall, unterscheiden wir die folgenden drei Fälle:

- Das Bild beider Terme sind unterschiedliche Konstanten (wobei das Bild einer Konstanten die Konstante selbst ist). In diesem Fall scheitert der CHASE.
- Das Bild beider Terme sind unterschiedliche Nullwerte. In der gesamten (Ziel-) Datenbankinstanz wird einer der Nullwerte (gewöhnlich der mit höherem Index) durch den anderen ersetzt.
- Das Bild des einen Terms ist eine Konstante, während das Bild des anderen Terms ein Nullwert ist. In der gesamten (Ziel-) Datenbankinstanz wird der Nullwert durch die Konstante ersetzt.

Handelt es sich beim CHASE-Objekt um eine Datenbankanfrage, so übernehmen in obiger Fallunterscheidung existenzquantifizierte Variablen die Rolle der Nullwerte.

Der CHASE-Algorithmus ist über vierzig Jahre alt. Ursprünglich diente er lediglich der Absicherung eines guten Datenbankentwurfs, seitdem sind jedoch zahlreiche Anwendungsgebiete hinzugekommen:

- Integration heterogener Datenbanken,
- Anfragetransformation mit Beschränkung auf bestimmte Nutzersichten (AQuV; Answering Queries using Views),
- Data Cleaning,
- Anfrageoptimierung unter Integritätsbedingungen,
- Datenbanktransformationen.

Weitere Anwendungsgebiete, wie die Invertierung von Datenbanktransformationen und Provenance-Management, sind gegenwärtiger Stand der Forschung am Lehrstuhl für Datenbank- und Informationssysteme und sollen in Abschnitt 6 näher erläutert werden.

Aus den oben genannten Anwendungsgebieten ergibt sich bereits eine deutliche Variabilität der möglichen CHASE-Parameter und -Objekte. Für Datenbankintegration und Datenbanktransformation werden Transformationsregeln (s-t TGDs) verwendet, für AQuV kommen Sichtdefinitionen (TGDs) zum Einsatz, und Data Cleaning sowie Anfrageoptimierung erfolgt unter Verwendung allgemeiner Integritätsbedingungen (EGDs und TGDs). Wie bereits zuvor erwähnt, handelt es sich beim Objekt des CHASE entweder um eine Datenbankanfrage (Anfrageoptimierung und AQuV) oder um eine Datenbankinstanz (alle anderen Anwendungsfälle).

Während die meisten der oben genannten Probleme bei Wahl geeigneter CHASE-Parameter und -Objekte durch den Standard-CHASE direkt gelöst werden können, setzt die Anfrageoptimierung und -transformierung unter Integritätsbedingungen durch den CHASE mehrere Hilfsalgorithmen voraus. Im Wesentlichen handelt es sich hierbei um das unter dem Namen Backchase bekannte Verfahren, der das Ergebnis des vorherigen CHASE als Ausgangspunkt eines erneuten CHASE in umgekehrter Richtung verwendet und anschließend durch Finden eines homomorphen Kerns Redundanzen aus dem Ergebnis entfernt.

3.2 Konfluenz, Terminierung und Effizienz des universellen CHASE

Bisher wird der CHASE-Algorithmus zwar noch nicht in kommerziellen Lösungen verwendet (Ausnahme: aus dem Forschungsprototyp Clio entwickelte Teile des IBM Information Servers), es gibt jedoch eine Reihe prototypischer Implementierungen, mit denen einige Anwendungsfälle des CHASE näher untersucht werden können. Allerdings kann bisher keiner dieser Prototypen alle in Unterabschnitt 3.1 genannten Anwendungsbereiche bearbeiten. So sind in LLUNATIC [GMPS14] effiziente Techniken des Datenbankbereinigung implementiert, während PDQ [BLT14] Anfrageoptimierungen durchführen kann, die selbst kommerziellen Systemen überlegen sind. Beide Programme arbeiten also mit unterschiedlichen (und nur in der Theorie äquivalenten) CHASE-Objekten.

Tatsächlich ist die Verarbeitung der unterschiedlichen CHASE-Parameter und -Objekte in keinem uns bekannten Werkzeug vereinheitlicht, stattdessen kommen immer getrennte Techniken zum Einsatz. Graal [BLM+15] zeigt Ansätze für die analoge Behandlung von Instanzen und Anfragen, jedoch sind auch hier die berücksichtigten Anwendungsfälle sehr speziell (eine zu PDQ äquiva-

lente Anfrageoptimierung ist beispielsweise nicht möglich).

An dieser Stelle sei erwähnt, dass einige CHASE-Prototypen bereits jetzt über Erweiterungen des zuvor beschriebenen Standard-CHASE verfügen. VLog integriert beispielsweise Negationen in den CHASE [CDG+19]. Allerdings sind diese Negationen auf stratifizierte Mengen von TGDs beschränkt, eine Einschränkung, die auf Datalog zurückgeht. Wenn wir den Test der Aktiviertheit des Triggers als Negation im CHASE interpretieren, verfügt jedoch selbst der klassische Standard-CHASE über Negationen – und zwar ohne eine Beschränkung auf stratifizierte TGDs.

Die ursprüngliche Variante des CHASE – d.h. die Berücksichtigung von Verbund-Abhängigkeiten und Funktionalen Abhängigkeiten beim Datenbankentwurf – ist sowohl terminierend, als auch konfluent. Der CHASE erzeugt in diesem Fall also unabhängig von der Reihenfolge der Regelanwendungen stets das gleiche (oder zumindest ein isomorphes) Ergebnis. Die Erweiterung des CHASE um existenzquantifizierte Variablen erlaubt es zwar, z.B. Inklusionsabhängigkeiten und damit auch Fremdschlüsselbeziehungen zu berücksichtigen, jedoch unter Verlust der sicheren Terminierung des Algorithmus. Tatsächlich kann diese Terminierung des CHASE sogar abhängig von der Reihenfolge der Regelanwendungen sein. Obwohl das allgemeine Problem der CHASE-Terminierung unentscheidbar ist, wurden inzwischen eine ganze Reihen von Testverfahren entwickelt, welche eine sichere Terminierung des CHASE allein auf Basis der Integritätsbedingungen zusichern können. Einige dieser Kriterien werden in Abschnitt 4 vorgestellt. Häufig beziehen sich CHASE-Terminierungskriterien jedoch auf den Skolem-Oblivious-CHASE, nicht den Standard-CHASE (dessen Terminierung jedoch durch die Terminierung des Skolem-Oblivious-CHASE garantiert wird). Das Kriterium der *WA-Stratifizierung* [GST11] erkennt zwar, wenn während des Standard-CHASE eine TGD aus Gründen der Triggerdeaktivierung nicht von einer anderen TGD gefeuert werden kann, allerdings werden bei derartigen Betrachtungen lediglich diese beiden Integritätsbedingungen berücksichtigt – die Triggerdeaktivierung könnte jedoch durch eine dritte Integritätsbedingung erfolgt sein. Das Kriterium der *Restricted Model-Faithful Acyclicity* [CDK17] berücksichtigt ein derartiges Zusammenspiel mehrerer Regeln, geht jedoch von einer bestimmten Strategie der Regelanwendungsreihenfolge aus. Die Terminierung des CHASE – und sogar allgemein sein Ergebnis – sind von dieser Reihenfolge abhängig.

Während die Terminierung des CHASE also recht gut verstanden ist, wurde die erwähnte Reihenfolgeabhängigkeit, also die Konfluenz des CHASE, bisher kaum untersucht. Wie zuvor erwähnt ist der CHASE auf Funktionalen Abhängigkeiten konfluent. Für Erweiterungen der Funktionalen Abhängigkeit ist der CHASE entweder konfluent (Graph-CHASE in [FFTD15]) oder Fälle von Inkon-

fluenz treten auf, die dem Standard-CHASE völlig fremd sind (Conditional FDs in [Ndi11]). Interessanterweise sind wichtige Spezialformen des CHASE, wie der Oblivious-CHASE, der Skolem-Oblivious-CHASE und der Core-CHASE, stets konfluent.

Obwohl zahlreiche Komplexitätsuntersuchungen zum CHASE existieren, beziehen sich diese bisher meist auf Worst-Case-Abschätzungen. Im Promotionsvorhaben soll dies durch neuere Techniken erweitert werden. So erlaubt die Untersuchung der parametrischen Komplexität, relativ konstant bleibende Parameter (z.B. die Anzahl der Attribute) zu isolieren. Durch die kostenbasierte Laufzeitabschätzung werden exaktere Aussagen zur Effizienz des CHASE möglich, die über ein Worst-Case Szenario hinaus gehen.

4 Eigene Vorarbeiten

Die Problematik der CHASE-Terminierung wurde von uns bereits in zwei vorbereitenden Arbeiten untersucht. Die Erstellung eines Lehrvideos im Rahmen eines Projektmoduls diente als Anlass, den aktuelle Forschungsstand hinsichtlich der Terminierungseigenschaften unterschiedlicher CHASE-Varianten aufzubereiten [GD19]. Während sich das Promotionsvorhaben auf den Standard-CHASE konzentrieren soll, wurden in dieser Voruntersuchung Oblivious-CHASE, Skolem-Oblivious-CHASE (auch Semi-Oblivous-CHASE genannt), Standard-CHASE und Core-CHASE verglichen. Hierbei bezogen wir uns konkret auf die Anwendungsfälle der Datenbanktransformation und des Data Cleanings. Aufgrund dieser Spezialisierung beschränkten wir uns auf das CHASE-Objekt der Datenbankinstanz. Da sich nahezu alle bisherigen Untersuchungen zur CHASE-Terminierung auf abstrakte Eigenschaften der CHASE-Parameter (also der Integritätsbedingungen) beziehen und unabhängig vom betrachteten CHASE-Objekt sind, können unsere Erkenntnisse jedoch auf andere CHASE-Anwendungsgebiete (z.B. Anfragetransformation) verallgemeinert werden.

Die Thematik der CHASE-Terminierung wurde durch unsere Masterarbeit fortgesetzt [Gör20]. Der Schwerpunkt hierbei lag auf dem Vergleich der bisher bekannten Terminierungskriterien des Standard-CHASE und der Implementierung mehrerer dieser Kriterien. Obwohl das Problem der CHASE-Terminierung nämlich im Allgemeinen unentscheidbar ist, gibt es Klassen von Integritätsbedingungen, für die der CHASE stets terminiert. Indem wir vor Durchführung des eigentlichen CHASE den verwendeten CHASE-Parameter auf seine Zugehörigkeit zu einer dieser Klassen überprüfen, können wir die Terminierung des CHASE unabhängig vom verwendeten CHASE-Objekt garantieren. Die meisten dieser Tests beziehen sich zwar nicht auf den Standard-CHASE, sondern auf den Skolem-Oblivious-CHASE, können jedoch auch für ersteren verwendet

werden. Des Weiteren berücksichtigen die meisten Terminierungstests nur eine Form von CHASE-Parameter, und zwar TGDs, während EGDs entweder ignoriert werden können (z.B. im Fall der Schwachen Azyklizität) oder sogar zu fehlerhaften Testergebnissen führen können (z.B. im Fall der Azyklizität). Aus den verglichenen Terminierungskriterien wurden die Kriterien Schwache Azyklizität, Reiche Azyklizität, Safety und Azyklizität ausgewählt und in Form eines Terminierungstesters des CHASE-Werkzeugs ChaTEAU umgesetzt.

Schwache Azyklizität Wenn eine TGD zu einer Endlosschleife des Standard-CHASE beiträgt, muss sie dazu in der Lage sein, beliebig oft neue Tupel zu generieren. Da die ursprüngliche Datenbankinstanz als endlich angenommen wird, enthalten diese Tupel notwendigerweise Werte, die zuvor noch nicht in der Datenbankinstanz vorkamen – durch den CHASE zuvor erzeugte markierte Nullwerte. Diese Erkenntnis gilt ebenfalls für die eingebetteten TGDs, welche diese Nullwerte zuvor generiert hatten. Das Kriterium der Schwachen Azyklizität verfolgt die Weitergabe dieser Nullwerte anhand eines Graphen mit besonderen (Beitrag zur Erzeugung eines neuen Nullwertes) und gewöhnlichen Kanten (Weitergabe eines Wertes). Existiert in diesem Graphen ein Zyklus, der durch eine besondere Kante geht, ist die Terminierung des CHASE nicht garantiert.

Reiche Azyklizität Wenn eine TGD, die zu einer Endlosschleife des naiven Oblivious-CHASE beiträgt, wiederholt neue Tupel generiert, so enthält zumindest der Trigger der TGD notwendigerweise Werte, die zuvor noch nicht in der Datenbankinstanz vorkamen. Da wir also nicht fordern, dass dieser Nullwert des Triggers in das neue Tupel gelangt, werden in zusätzlichen Fällen neue Nullwerte erzeugt. Wir ergänzen den Graphen der Schwachen Azyklizität daher um weitere spezielle Kanten. Die Terminierung des naiven Oblivious-CHASE kann nur dann garantiert werden, wenn der Graph keine Zyklen durch besondere oder spezielle Kanten enthält. In diesem Fall würde auch der Standard-CHASE terminieren, denn die Terminierung des naiven Oblivious-CHASE ist hinreichendes Kriterium für die Terminierung des Skolem-Oblivious-CHASE, dessen Terminierung wiederum hinreichend für die Terminierung des Standard-CHASE ist.

Safety Für die Kriterien der Schwachen und Reichen Azyklizität wird zwischen der Erzeugung von Nullwerten und der Weitergabe von Werten unterschieden. Unter Umständen können wir allerdings ausschließen, dass bestimmte Variablen einer TGD mit einem Nullwert der Instanz belegt sind (indem wir Gleichheit mit einem Attribut fordern, welches keinen Nullwert enthalten kann). Für das Safety-Kriterium schränken wir die Weitergabe allgemeiner Werte auf die Weitergabe potentieller Nullwerte ein, wodurch wir die Mächtigkeit des Testkriteriums der Schwachen Azyklizität erhöhen.

Azyklizität Bei diesem Terminierungskriterium wird nicht die Azyklizität der gegenseitigen TGD-Aufrufung untersucht (was einem wesentlich einfacheren

Terminierungskriterium entspräche). Stattdessen werden die relationalen Atome der einzelnen TGDs um einen String sogenannter „Adornments“ ergänzt. Die Adornments werden auf eine Weise zwischen Atomen des TGD-Rumpfes und TGD-Kopfes einer TGD bzw. zwischen TGD-Kopf und TGD-Rumpf verschiedener TGDs weitergegeben, die dem CHASE Algorithmus ähnelt. Diese Übertragung und Neuentstehung von adornten Atomen wird durch einen Graphen modelliert, dessen Zyklenfreiheit die Terminierung des CHASE garantiert.

Durch Nutzung des ebenfalls implementierten Constraint-Rewriting Algorithmus der Substitutionslosen Simulation verarbeitet der Terminierungstester neben TGDs auch EGDs. Gerade in Verbindung mit Substitutionsloser Simulation weist der mächtigste implementierte Test – der Test auf Azyklizität – jedoch eine enorme Komplexität auf, weshalb im konkreten Anwendungsfall zunächst ein effizienterer Terminierungstest durchgeführt werden sollte.

Diese Vorarbeiten hinsichtlich der CHASE-Terminierung sollen im Promotionsvorhaben aktualisiert und durch vergleichbare Arbeiten zur Konfluenz ergänzt werden.

5 Vorarbeiten am Lehrstuhl

Wie bereits in Abschnitt 1 ausgeführt, wird die Auswertung großer Datenmengen bisher am Lehrstuhl hauptsächlich anhand zweier Anwendungsgebiete untersucht: Forschungsdatenmanagement und Assistenzsysteme. Während einige dieser Untersuchungen sich konkret auf den CHASE beziehen, steht die Integration dieses Algorithmus für andere noch aus. Die Einbeziehung des CHASE in derartige Arbeiten ist Teil des Promotionsvorhabens. Allerdings ist der bisherige CHASE-Algorithmus für manche dieser Anwendungsgebiete nicht ausdrucksstark genug. Für statistische Operationen (z.B. Regressionsanalysen) werden beispielsweise arithmetische Operationen benötigt, welche der Standard-CHASE nicht berücksichtigt. Allerdings ist es keinesfalls nötig, alle Datenauswertungsverfahren in den CHASE (oder überhaupt in die Datenbankanwendung) zu integrieren, es könnte sich als ausreichend herausstellen, bestimmte Aufgaben von externen Algorithmen durchführen zu lassen (z.B. Matrixoperationen durch SQL-Anfragen oder Matlab).

Aus bisherigen Arbeiten des Lehrstuhls gingen folgende Kandidaten für Erweiterungen des CHASE hervor:

- Disjunktionen,
- Negationen,
- allgemeinere Vergleichsoperatoren ($\neq, <, \leq, ...$),

- skalare Funktionen und Aggregatfunktionen,
- Rekursion.

Diese Liste ist jedoch keinesfalls abschließend und soll im Rahmen des Promotionsvorhabens aktualisiert, systematisiert und formalisiert werden.

Im Folgenden werden die Arbeiten von weiteren Doktoranden des Lehrstuhls, die einen Bezug zum Promotionsvorhaben haben, kurz vorgestellt.

Marten: Herr Marten beschäftigt sich im Rahmen seiner Dissertation mit Algorithmen maschinellen Lernens, welche die Basis smarter assistiver Systeme bilden. Einen Kernpunkt seiner Arbeit stellt die Effizienzsteigerung der Algorithmen durch Parallelisierung dar. Ein weiterer Kernpunkt bezieht sich auf die Transformation der Algorithmen, sodass sie auf Grundoperationen einer Datenbankanfragesprache für Big Data reduziert sind. Für meine Arbeit ergibt sich hieraus die Aufgabe, auch den CHASE um die entsprechenden Datenbank-Grundoperationen zu erweitern. Obwohl dies keinen zentraler Aspekt von Herrn Martens Arbeit darstellt, bietet der von ihm untersuchte Forschungsgegenstand Ansatzpunkte für die automatisierte Integration von Privatsphäre, Provenance und Anfrageoptimierung in das Auswertungsverfahren.

Grunert: Herr Grunert entwickelt Ansätze, Algorithmen von zentral gelegenen Cloud-Servern auf dezentralen Datenquellen (Sensoren, smarte Geräte) zu verteilen. Auf diese Weise sollen zum einen Anonymisierungsgarantien gewährleistet werden, zum anderen können Vorberechnungen auch zu einer Steigerung der Effizienz führen. In Herr Grunerts Arbeit basiert diese automatische Transformation der Algorithmen auf algebraischen Regeln und weiteren Heuristiken. Dementsprechend wurden Parameter wie Privatsphäre auch nicht formalisiert. Die Heuristiken sollen im Rahmen der Promotionsarbeit durch den CHASE ersetzt bzw. in diesen integriert werden.

Bruder: Herr Bruder nimmt Multimediaanwendungen als Ansatzpunkt, um Datenbankanfragen an komplexe Datenbankstrukturen semantisch zu optimieren. Hierbei wurde die Tableauauswertungstechnik, auf welcher der CHASE beruht, erweitert. Aus Herrn Bruders Untersuchungen geht hervor, dass bestimmte bereits existierende CHASE-Werkzeuge (wie PDQ) kommerziellen Systemen in der Optimierung multimediatypischer Anfragen zwar überlegen sind, durch die erwähnten Erweiterungen der klassischen Tableauoptimierung die Menge optimierbarer Anfragen aber noch erweitert werden kann. Allerdings sind selbst diese erweiterten Tableauanfragen – verglichen mit dem Potential des allgemeinen CHASE-Algorithmus – noch sehr eingeschränkt. Im Rahmen des Promotionsvorhabens soll eine Erweiterung des Algorithmus hin zu allgemeinen Anfragen ermöglicht werden. Darüber hinaus wäre erstrebenswert, die von Herrn Bruder gefundenen Optimierungsansätze in die Provenance- und Privacy-Techniken der

anderen Arbeiten zu integrieren.

Auge: Frau Auge beschäftigt sich mit Provenance-Auswertungen auf Forschungsdatenbanken in Gegenwart von Schemaevolution. Selbst wenn das Datenbankschema nach Durchführung einer Anfrage verändert wurde, können unter Umständen durch Invertierungen der beteiligten Transformationsregeln die am Ergebnis beteiligten Tupel identifiziert werden. Hierbei ist Frau Auge nicht nur an der Herkunft der resultierenden Tupeln interessiert (das wäre die Frage nach der *where*-Provenance), sondern auch an allen anderen an der Konstruktion der Ergebnisinstanz beteiligten Tupel (die minimale Zeugenbasis und die sich hieraus ergebende *why*-Provenance). Um die Rahmenbedingung der Privatheit zu gewährleisten, versucht Frau Auge, intensionale (d.h. beschreibende) statt nur extensionale Provenance-Informationen zu erlangen. Sowohl Provenance als auch Privacy sollen durch einen angepassten CHASE-Algorithmus erreicht werden. Für das Promotionsvorhaben stellt sich nun die Aufgabe, die Grundeigenschaften Terminierung, Konfluenz und Effizienz der so erweiterten CHASE Technik abzusichern.

6 Teilziele der Promotion

Ziel des Promotionsvorhabens ist, die Rahmenbedingungen Privacy, Provenance und Optimierung durch den CHASE in entsprechend formalisierte Auswertungstechniken zu integrieren. Allerdings lassen die in Abschnitt 5 zusammengefassten Arbeiten erkennen, dass sowohl die vom CHASE verarbeiteten Parameter, als auch die CHASE-Objekte hierfür deutlich erweitert werden müssen.

Im Rahmen der Arbeit von Herrn Grunert sollen beispielsweise Privacy-Vorschriften (z.B. k-Anonymität), Ressourcenbeschränkungen und Fähigkeiten eines Smartphones (Capabilities) in die Auswertungsverfahren integriert werden. Um den CHASE hierfür einsetzen zu können, müssen diese Eigenschaften – entsprechend der zuvor betrachteten Integritätsbedingungen, Transformationsregeln und Sichtdefinitionen – als prädikatenlogische Regeln formalisiert werden. Während diese CHASE-Parameter also erst präzise definiert werden müssen, handelt es sich beim Objekt des CHASE – wie in vorherigen Beispielen – um Datenbankanfragen. Diese Auswertungsanfragen müssen allerdings eventuell um lineare Algebra erweitert werden. Obwohl in der Arbeit von Frau Auge ebenfalls Privacy-Anforderungen gewährleistet werden sollen, ist die vom CHASE zu verarbeitende Parametermenge grundverschieden von den Anforderungen, die Herr Grunerts Arbeit stellt. Auswertungsanfragen, Transformationsregeln und Datenbankinstanzen sind Parameter und Objekte, die der CHASE bereits verarbeiten kann. Allerdings hat sich gezeigt, dass für die Darstellung von Schemaevolution in Kombination mit Provenance-Anfragen herkömmliche s-t TGDs

nicht ausreichend sind, und stattdessen Second Order s-t TGDs benötigt werden [FKPT05]. Diese können – im Gegensatz zu den zuvor definierten TGDs – Funktionssymbole und Gleichheitsatome im Rumpf der TGD enthalten. Neben diesen skalaren Funktionen kommen auch Aggregatfunktionen in den Transformationsregeln und Auswertungsanfragen vor und sollen daher vom CHASE berücksichtigt werden.

Zum einen sollen im Promotionsvorhaben also unterschiedliche CHASE-Parameter und -Objekte in ein konsistenten Gesamtmodell zu integriert werden.

Zum anderen sollen Fragestellungen der Konfluenz, Terminierung, Effizienz und Effektivität in einzelnen Arbeitspaketen für die verschiedenen Stufen der CHASE-Erweiterungen schrittweise untersucht werden. Es ist nicht erwartbar, dass die Rahmenbedingungen im allgemeinen Fall erfüllt werden können (dies ist selbst für den nicht erweiterten CHASE nicht möglich). Stattdessen sollen pragmatische Lösungen gefunden werden, welche es erlauben, Problemfälle separat zu betrachten. Für den bisherigen CHASE kann das Problem der nicht entscheidbaren Terminierung beispielsweise durch Definition von Terminierungsklassen eingeschränkt werden, auf denen der CHASE garantiert terminiert. Daneben gibt es auch TGD-Klassen (zu denen z.B. Inklusionsabhängigkeiten gehören), für welche die CHASE-Terminierung entscheidbar (wenn auch nicht garantiert) ist.

Interessanterweise handelt es sich bei Terminierung, Konfluenz und Effizienz nicht um gänzlich separat voneinander zu verstehende Rahmenbedingungen. Zumindest auf algorithmischer Ebene ergeben sich Synergieeffekte, so könnten mit Hilfe abgewandelter Terminierungstests die Konfluenz untersucht werden und Terminierungstests Potential aufzeigen, die Effizienz des CHASE zu steigern. Betrachten wir hierfür folgende beiden TGDs:

$D = \{R(1, 2, 3)\}$
$r_1 : R(x, y, z) \rightarrow S(x, y)$
$r_2 : S(x, y) \rightarrow \exists E : R(x, E, y)$

Wenden wir r_1 und r_2 auf D an, terminiert der CHASE nicht, was jeder Terminierungstest korrekt vorhersagen kann. Varianten der Stratifizierung, z.B. die WA-Stratifizierung, testen hierfür, ob sich die Regeln gegenseitig „feuern“ können. Dies ist für beide Regeln der Fall, wir schreiben dies auch als $r_1 < r_2, r_2 < r_1$. In diesem Zyklus des Abhängigkeitsgraphen werden darüber hinaus die erzeugten Nullwerte in einer Weise weitergegeben, die für eine Endlosschleife des CHASE notwendig ist. Dies kann durch das Kriterium der schwachen Azyklizität überprüft werden. Nehmen wir nun eine leichte Änderung an r_2 vor und ergänzen eine weitere TGD:

$D = \{R(1, 2, 3)\}$
$r_1 : R(x, y, z) \rightarrow S(x, y)$

$r_2' : S(x, y) \to \exists E : R(x, y, E)$
$r_3 : R(x, y, z) \to R(x, z, y)$

Werden r_2 und r_2' auf denselben Trigger angewandt, so erzeugt r_3 aus dem von r_2' generierten Tupel das Tupel, das auch r_2 erzeugt hätte. Wir erwarten also, dass $r_1 < r_2 < r_3 < r_1$ gilt und dieser Zyklus nicht schwach azyklisch ist (was z.B. das Ergebnis der c-Stratifizierung wäre). Tatsächlich gilt für den Standard-CHASE jedoch $r_1 < r_2'$ nicht, da das von r_2' erzeugte Tupel – unabhängig von der gewählten Datenbankinstanz – bis auf die existenzquantifizierte Variable identisch zum Trigger von r_1 ist, der Trigger ist also inaktiv. Der Standard-CHASE terminiert daher (was z.B. vom Kriterium der WA-Stratifizierung erkannt würde). Darüber hinaus wissen wir nun auch, welche TGD welche andere TGD feuert. Wenn also zuletzt der CHASE erfolgreich auf r_2' angewandt wurde, kann das generierte Tupel nur als Trigger von r_3 dienen (es gilt weder $r_2' < r_1$ noch $r_2' < r_2'$). Dies führt zu einer Effizienzsteigerung des CHASE, die z.B. in ähnlicher Weise in Graal implementiert wurde [BLM+15].

Betrachten wir nun r_2' zusammen mit einer weiteren TGD, die ebenfalls Tupel in R erzeugt:

$D = \{S(1, 1)\}$
$r_2' : S(x, y) \to \exists E : R(x, y, E)$
$r_4 : S(x, z) \to R(x, x, z)$

Wenden wir zuerst r_2' und dann r_4 auf den Trigger $S(1, 1)$ an, werden die Tupel $R(1, 1, \eta_1{}^1), R(1, 1, 1)$ generiert. Wenden wir hingegen zuerst r_4 auf den Trigger $S(1, 1)$ an, ist dieser Trigger für r_2' deaktiviert und es wird lediglich das Tupel $R(1, 1, 1)$ erzeugt. Der CHASE ist also inkonfluent (dennoch führen beide Reihenfolgen zu zueinander homomorphen Universallösungen). Erweitern wir nun den Rumpf von r_4 um ein weiteres Atom:

$D = \{S(1, 1)\}$
$r_2' : S(x, y) \to \exists E : R(x, y, E)$
$r_4' : S(x, z), R(x, x, y) \to R(x, x, z)$

Bei gegebener Ausgangsdatenbankinstanz D kann r_4' nicht angewandt werden. Tatsächlich kann auch keine Datenbankinstanz existieren, in der beide TGDs bei paralleler Anwendung des CHASE aktive Trigger besitzen, bei sequentieller Anwendung des CHASE jedoch nicht: Ergänzt man beispielsweise D um das Tupel $R(1, 1, 2)$, hat zwar r_4' nun einen aktiven Trigger, der Trigger von r_2' ist jedoch inaktiv. Der CHASE auf den beiden Regeln ist also konfluent. Man beachte, dass unser prinzipielles Vorgehen – nämlich das Finden von Belegungen, welche die untersuchten TGDs zueinander in Beziehung bringen, und die Untersuchung der sich daraus ergebenden Auswirkungen auf die Aktiviertheit der Trigger – auch beim Testen der Terminierung zum Tragen kam.

[1] η_i ($i \in \mathbb{N}^*$) kennzeichnet hier einen vom CHASE generierten markierten Nullwert.

Darüber hinaus gibt es Ansätze der Effizienzsteigerung, welche die Konfluenz des CHASE voraussetzen, so zum Beispiel die Sortierung von s-t TGDs und EGDs in [BIL16]. Andere Strategien der Effizienzsteigerung – z.B. die Priorisierung voller TGDs (Datalog First-Strategie) in [KMR19] – setzen implizit die Inkonfluenz des CHASE voraus. Wir können also unsere Strategie der Effizienzsteigerung vom Ergebnis des Konfluenztests abhängig machen.

Die Teilziele der Promotion lassen sich in einen horizontalen und einen vertikalen Arbeitsbereich unterteilen. Die einzelnen Themenabschnitte des horizontalen Arbeitsbereichs sollen im Rahmen des Promotionsvorhabens sequentiell abgearbeitet werden:

- Untersuchung von Terminierung, Konfluenz und Effizienz,
- Identifikation der Auswertungsoperatoren,
- Umsetzung der Techniken in ChaTEAU,
- Test auf Effektivität in Anwendungsfällen.

Im Gegensatz hierzu sollen die Themen des vertikalen Arbeitsbereichs parallel zu den anderen Aufgaben bearbeitet werden:

- Erweiterungen der CHASE-Parameter und -Objekte,
- Untersuchung ihrer Effizienz, Terminierungs- und Konfluenz-Eigenschaften.

Terminierung, Konfluenz und Effizienz des CHASE

Ursprünglich war der CHASE für zwei Arten von Parametern definiert: Funktionale Abhängigkeiten (EGDs) und Mehrwertige Abhängigkeiten (volle TGDs). Für diese Kombination von Parametern ist der CHASE terminierend und konfluent, doch bereits die Erweiterung der verwendeten Parameter um eingebettete TGDs macht diese zentralen Eigenschaften des CHASE zunichte. Ziel des Promotionsvorhabens ist es, Kriterien zu entwickeln bzw. weiterzuentwickeln, die Terminierung und Konfluenz des Standard-CHASE zusichern. Terminiert der CHASE (oder präziser: genügen die CHASE-Parameter dem Terminierungskriterium der Schwachen Azyklizität), benötigt der CHASE erwiesenermaßen polynomiale Laufzeit. Diese Abschätzung betrifft einerseits lediglich die Datenkomplexität des CHASE (d.h. die abhängige Größe ist die Größe der Datenbank), andererseits handelt es sich um eine Worst-Case Abschätzung. Hinsichtlich der kombinierten Komplexität (abhängig von der Anzahl oder Komplexität der Integritätsbedingungen) gibt es eine ganze Reihe von Untersuchungen, die jedoch immer auf einen bestimmten Anwendungsfall bezogen sind (z.B. Implikationsproblem einer bestimmten Art von Abhängigkeit); häufig ist auch die

Komplexität des CHASE-Anwendungsfalles recht gut untersucht (z.B. verschiedene Formen von Datenbankreparaturen), was jedoch nicht bedeutet, dass der CHASE die so abgeschätzte Komplexität tatsächlich erreicht.

In vielen Fällen kann in der praktischen Anwendung mit etablierten Verfahren (wie dem Pruning von Lösungsräumen) ein effizientes Verfahren erreicht werden, selbst wenn theoretische Betrachtungen eine exponentielle Worst-Case-Komplexität erwarten lassen. In diesem Zusammenhang sollen Verfahren der parametrischen Komplexität und die Entwicklung konkreter Kostenfunktionen zur Laufzeitabschätzung überprüft werden.

Identifikation der Auswertungsoperatoren Letztendlich ist es anzustreben, dass der CHASE in praktischen (insbesondere auch kommerziellen) Anwendungen Verbreitung findet. Hierfür reicht die Ausdrucksstärke des bisher definierten CHASE nicht aus. Anhand praktischer Anwendungsfälle, die sich aus den Vorarbeiten des Lehrstuhl ergeben, sollen Kandidaten für CHASE-Erweiterungen identifiziert werden. Durch Ergänzung des CHASE um diese Erweiterungen – beispielsweise Auswertungsoperatoren – soll der CHASE effektiv auf die praktischen Anwendungsfälle angewandt werden können.

Konzeption eines universellen CHASE

Der CHASE kann auf diverse Parameter und Objekte angewandt werden. Konkrete Arbeiten zum CHASE gehen jedoch zum Teil von einem konkreten Anwendungsfall (z.B. Datenaustausch) und damit sehr speziellen Parametern und Instanzen aus. Diese Fokussierung kann zu unvollständigen Antworten hinsichtlich Terminierung (auf s-t TGDs angewandt terminiert der CHASE trivialerweise immer) und Effizienz (z.B. Beschränkung auf Datenkomplexität) des CHASE führen. Bevor der CHASE erweitert werden kann, muss das theoretische Rahmenwerk des CHASE für allgemeine Parameter und Instanzen formalisiert sein. Konkret bedeutet dies, dass unterschiedliche CHASE-Teil-Algorithmen ineinander integriert werden sollen.

Es gibt bereits Ansätze, Negation, Ungleichheiten, Vergleiche und Funktionssymbole in den CHASE zu integrieren. Dies führt entweder zu einer speziellen Variante des CHASE (z.B. Conditional CHASE für Negationen bzw. AV CHASE für arithmetische Vergleiche) oder zur Identifizierung von Sonderfällen, in denen der (leicht angepasste) Standard-CHASE zum Finden universeller Lösungen bzw. sicherer Antworten verwendet werden kann (zum Beispiel Beschränkung der Anzahl von Ungleichheiten pro Konjunkt). Ziel des Promotionsvorhabens ist die Konzeption eines universellen (Standard) CHASE, der die bisher erarbeiteten Teillösungen in einem einzigen Verfahren vereint.

Terminierung, Konfluenz und Effizienz des universellen CHASE

Dieser Themenabschnitt setzt sich damit auseinander, inwiefern die zuvor für den CHASE gewonnenen Erkenntnisse hinsichtlich Terminierung, Konfluenz

und Effizienz auch auf den erweiterten, universellen CHASE zutreffen. Um effektiv auf praktische Anwendungsfälle anwendbar zu sein, muss der CHASE in ausreichend kurzer (und damit auch endlicher) Zeit verlässliche Ergebnisse liefern. Wenn der erweiterte CHASE also unter bestimmten Umständen deutlich ineffizienter als der bisherige CHASE ist, sind diese Umstände auf ihre praktische Relevanz hin zu untersuchen. Problemfälle sollen hierfür zunächst isoliert werden, um sie eventuell gesondert behandeln zu können.

Umsetzung der Techniken in ChaTEAU

ChaTEAU ist ein im Rahmen mehrerer studentischer Arbeiten am Lehrstuhl entstandenes prototypisches Java-Programm, welches mehrere Anwendungsfälle des Standard-CHASE kombinieren soll. Zum gegenwärtigen Zeitpunkt kann ChaTEAU die CHASE-Parameter TGD, EGD und s-t TGD in die CHASE-Objekte Datenbankinstanz oder Datenbankanfrage integrieren. Darüber hinaus verfügt CHASE über Ansätze, den Back-CHASE durchzuführen, und den im Rahmen unserer Masterthesis entstandenen Terminierungstester.

Die in den zuvor beschriebenen Abschnitten definierten CHASE-Erweiterungen sollen anhand von ChaTEAU praktisch umgesetzt werden. Auf diese Weise kann die praktische Umsetzbarkeit der theoretischen Konzepte gezeigt und eine Umgebung für quantitative wie auch qualitative Tests geschaffen werden.

Test auf Effektivität in Anwendungsfällen

Die zu untersuchenden Anwendungsfälle des CHASE liegen in den Bereichen Assistenzsysteme und Forschungsdatenmanagement. Hier soll der CHASE Rahmenbedingungen wie Privacy und Provenance zusichern. Ob das Zusammenspiel der verschiedenen Parameter, Objekte und Rahmenbedingungen tatsächlich effektiv ist, soll anhand des erweiterten CHASE-Werkzeugs ChaTEAU gezeigt werden. Die Anwendungsfälle werden hierfür in einer Weise codiert, die von ChaTEAU verarbeitet werden kann, um dann auf Effektivität, Terminierung, Konfluenz, Effektivität hin getestet zu werden. Die Effizienz des CHASE soll insbesondere auf Basis von Laufzeituntersuchungen bewertet werden.

7 Zeitplan der Promotion

Halbjahr	Ziel
1	Untersuchung von Terminierung, Konfluenz und Effizienz
2	Identifikation der Auswertungsoperatoren
3	Konzeption eines universellen CHASE
4	Untersuchung von Terminierung, Konfluenz und Effizienz des universellen CHASE
5	Umsetzung der Techniken in ChaTEAU
6	Test auf Effektivität in Anwendungsfällen

Tabelle 1: Zeitplan der Promotion

Literatur

[BIL16] Bonifati, Angela ; Ileana, Ioana ; Linardi, Michele: Functional dependencies unleashed for scalable data exchange. In: *Proceedings of the 28th International Conference on Scientific and Statistical Database Management*, 2016, S. 1–12

[BLM+15] Baget, Jean-François ; Leclère, Michel ; Mugnier, Marie-Laure ; Rocher, Swan ; Sipieter, Clément: Graal: A Toolkit for Query Answering with Existential Rules. In: *Rule Technologies: Foundations, Tools, and Applications - 9th International Symposium, RuleML 2015, Berlin, Germany, August 2-5, 2015, Proceedings*, 2015, 328–344

[BLT14] Benedikt, Michael ; Leblay, Julien ; Tsamoura, Efthymia: PDQ: Proof-driven Query Answering over Web-based Data. In: *PVLDB* 7 (2014), Nr. 13, 1553–1556. `http://dx.doi.org/10.14778/2733004.2733028`. – DOI 10.14778/2733004.2733028

[CDG+19] Carral, David ; Dragoste, Irina ; González, Larry ; Jacobs, Ceriel ; Krötzsch, Markus ; Urbani, Jacopo: VLog: A rule engine for knowledge graphs. In: *International Semantic Web Conference* Springer, 2019, S. 19–35

[CDK17] Carral, David ; Dragoste, Irina ; Krötzsch, Markus: Restricted Chase (Non)Termination for Existential Rules with Disjunctions. In: *IJCAI*, ijcai.org, 2017, S. 922–928

[DNR08] Deutsch, Alin ; Nash, Alan ; Remmel, Jeffrey B.: The chase revisited. In: *Proceedings of the Twenty-Seventh ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2008, June 9-11, 2008, Vancouver, BC, Canada*, 2008, 149–158

[FFTD15] Fan, Wenfei ; Fan, Zhe ; Tian, Chao ; Dong, Xin L.: Keys for Graphs. In: *Proc. VLDB Endow.* 8 (2015), Nr. 12, 1590–1601. http://dx.doi.org/10.14778/2824032.2824056. – DOI 10.14778/2824032.2824056

[FKPT05] Fagin, Ronald ; Kolaitis, Phokion G. ; Popa, Lucian ; Tan, Wang-Chiew: Composing schema mappings: Second-order dependencies to the rescue. In: *ACM Transactions on Database Systems (TODS)* 30 (2005), Nr. 4, S. 994–1055

[GD19] Görres, Andreas ; Düwel, Yvonne: *CHASE auf Datenbanken.* 2019 Lehrvideo entstanden im Rahmen des Moduls „Neueste Entwicklungen in der Informatik“

[GMPS14] Geerts, Floris ; Mecca, Giansalvatore ; Papotti, Paolo ; Santoro, Donatello: An Overview of the Llunatic System. In: *22nd Italian Symposium on Advanced Database Systems, SEBD 2014, Sorrento Coast, Italy, June 16-18, 2014*, 2014, S. 159–166

[Gör20] Görres, Andreas: *Erweiterung des CHASE-Werkzeugs ChaTEAU um ein Terminierungskriterium*, Universität Rostock, Masterthesis, 2020

[GST11] Greco, Sergio ; Spezzano, Francesca ; Trubitsyna, Irina: Stratification Criteria and Rewriting Techniques for Checking Chase Termination. In: *PVLDB* 4 (2011), Nr. 11, S. 1158–1168

[HL18] Hannula, Miika ; Link, Sebastian: On the interaction of functional and inclusion dependencies with independence atoms. In: *International Conference on Database Systems for Advanced Applications* Springer, 2018, S. 353–369

[KMR19] Krötzsch, Markus ; Marx, Maximilian ; Rudolph, Sebastian: The power of the terminating chase. In: *Proceedings of the 22nd International Conference on Database Theory (ICDT 2019). LIPIcs* Bd. 127, 2019, S. 3

[Ndi11] Ndindi, Reuben D.: *Repairing data with conflict-free conditional dependencies*, University of Edinburgh, Masterthesis, 2011