Universität Rostock

Traditio et Innovatio

Masterarbeit

Umsetzung von Provenance-Anfragen in Big-Data-Analytics-Umgebungen

eingereicht von: Tanja Auge

eingereicht am: 6. September 2017

Gutachter: Prof. Dr. rer. nat. habil. Andreas Heuer
Prof. Dr.-Ing. Alke Martens

Zusammenfassung

Der Begriff Big Data bezeichnet eine riesige Menge an Daten, die durch große Vielfalt und eine extrem schnelle Verarbeitungsrate charakterisiert ist. Eine dauerhafte Speicherung sowie unverzügliche Verarbeitung solcher Daten bedarf daher einige theoretische Überlegungen, um keine überflüssigen Daten zu speichern oder die „falschen“ Daten zu löschen. Ziel dieser Arbeit ist die Adaption von Techniken der Provenance-Anfragen ***why***, ***where*** und ***how*** in Umgebungen, die statt einfacher Anfragen wie Selektion, Projektion und Verbund auch OLAP-Operationen und weitere Machine-Learning-Algorithmen benutzen. Die ausschließlich extensionalen Provenance-Antworten werden dabei durch Provenance-Polynome sowie (minimalen) Zeugenbasen gegeben. Die Erweiterung des CHASE-Algorithmus für Datenbanken um eine BACKCHASE-Phase zur Provenance-Antwort-Bewertung ermöglicht so die Bestimmung des CHASE-Inversentyps (exakt / relaxt / ergebnisäquivalent) einer gegebenen Anfrage. Die Frage, welche Informationen für die Rekonstruktion der Quelldatenbank notwendig sind, kann so ein Stück weit beantwortet werden.

Abstract

The term Big Data refers to a huge amount of data, which is characterized by great diversity and an extremely fast processing rate. A permanent storage as well as immediate processing of such data therefore requires some theoretical considerations in order not to store any unnecessary data or to delete the "wrong" data. The aim of this thesis is the adaptation of techniques of the provenance queries ***why***, ***where*** and ***how*** in environments that use OLAP operations and other machine-learning algorithms instead of simple queries such as selection, projection and network. Provenance polynomials as well as (minimal) witness bases are given exclusively by the extensional provenance answers. The extension of the CHASE algorithm for databases by means of a BACKCHASE phase for the provenance response assessment thus enables the determination of the CHASE inverse type (exact / relaxt / data exchange equivalent) of a given query. Questions about which information is necessary for the reconstruction of the source database will be answered.

Inhaltsverzeichnis

Inhaltsverzeichnis **5**

1. Einleitung **9**

1.1. Einführung in die Thematik 9

1.2. Einführung des Beispieldatensatzes 11

1.3. Aufbau der Arbeit 14

2. Grundlagen **17**

2.1. Das relationale Datenbankmodell 17

2.1.1. Schemata und Instanzen 17

2.1.2. Integritätsbedingungen 18

2.2. Die Operationen im relationalen Modell 20

2.2.1. Relationale Operationen 20

2.2.2. OLAP-Operationen 23

2.3. Big Data Analytics 26

2.4. Data Provenance 28

2.5. Schemaabbildungen 28

2.6. Tableaus und der CHASE 29

2.6.1. Das Tableau als Darstellungshilfe 30

2.6.2. Das Universalwerkzeug CHASE 33

3. Aktueller Stand der Forschung **35**

3.1. Provenance-Anfragen und -Antworten: 35

3.1.1. ***Why***-Provenance 37

3.1.2. ***How***-Provenance 38

3.1.3. ***Where***-Provenance 39

3.1.4. Extensionale Antwort 39

3.2. Provenance-Polynome 40

3.2.1. Provenance-Halbringe 40

3.2.2. Provenance für aggregierte Anfragen 46

3.2.3. Provenance und **GROUP BY** 49

3.3. CHASE-Varianten . . . 51
3.3.1. Ein Überblick . . . 51
3.3.2. Vom Tableau zur Formel . . . 53
3.3.3. CHASE für s-t tgds und egds . . . 55
3.3.4. CHASE&BACKCHASE . . . 58
3.4. Inverse Schemaabbildung . . . 62
3.4.1. Allgemeine inverse Schemaabbildungen . . . 62
3.4.2. CHASE-inverse Schemaabbildungen . . . 64
3.5. Zusammenfassung . . . 66

4. Das Konzept der CHASE-inversen Abbildungen 69

4.1. Erweiterung der Theorie der CHASE-inverse Schemaabbildungen . . . 69
4.1.1. Die ergebnisäquivalente CHASE-Inverse . . . 69
4.1.2. Das CHASE&BACKCHASE-Verfahren zur Bestimmung von CHASE-inversen Schemaabbildungen . . . 71
4.1.3. Schemata, Instanzen und Teilmengenbeziehungen . . . 73
4.1.4. Existenzregeln . . . 74
4.2. Algebraische Grundoperationen mit zugehörigen CHASE-inversen Schemaabbildungen . . . 75
4.2.1. Einschränkung auf bestimmte Operationen . . . 76
4.2.2. Die Grundoperationen als s-t tgds und egs . . . 77
4.2.3. Ein Überblick . . . 78
4.2.4. CHASE-inverse Schemaabbildungen . . . 79
4.2.5. CHASE-inverse Schemaabbildungen mit Provenance-Informationen . . . 89
4.2.6. Zusätzliche Informationen für die Existenz einer exakten CHASE-Inversen . . . 99
4.2.7. CHASE-inverse Schemaabbildungen für Kompositionen . . . 100
4.2.8. Zusammenfassung . . . 103
4.3. Inverse Schemaabbildungen am Beispiel des Hidden Markov-Modells . . . 105
4.3.1. Das Hidden-Markov-Modell . . . 105
4.3.2. Das Hidden-Markov-Modell in SQL . . . 107
4.3.3. CHASE-inverse Schemaabbildungen . . . 108

5. Praktische Umsetzung 113

5.1. Analyse des Programms **ProSA** . . . 113
5.2. Erweiterungen des Programms **ProSA** . . . 114
5.3. Auswertung des Programms **ProSA** . . . 116

6. Fazit und Ausblick 117

6.1. Fazit . . . 117
6.2. Ausblick . . . 119

Literaturverzeichnis **121**

Anfrageverzeichnis **125**

Tabellenverzeichnis **127**

Symbolverzeichnis **131**

Abkürzungsverzeichnis **133**

A. Anhang: Mathematische Grundlagen **135**

B. Anhang: Der Beispieldatensatz **139**

C. Quelltext des erweiterten Programms ProSA **143**

D. Anhang: Aufbau des Datenträgers **149**

1. Einleitung

Ziel der vorliegenden Arbeit *Umsetzung von Provenance-Anfragen in Big-Data-Analytics-Umgebungen* ist die Adaption von Techniken der Provenance-Anfragen ***why***, ***where*** und ***how*** in Umgebungen, die statt einfacher Anfragen wie Selektion, Projektion und Verbund auch OLAP-Operationen und weitere Machine-Learning-Algorithmen benutzen. Die ausschließlich extensioanlen Provenance-Antworten werden dabei durch Provenance-Polynome, (minimalen) Zeugenbasen sowie Zeugelisten gegeben. Die Erweiterung des CHASE-Algorithmus für Datenbanken um eine BACKCHASE-Phase zur Provenance-Antwort-Bewertung ermöglicht so die Bestimmung des CHASE-Inversentyps (exakt / relaxt / ergebnisäquivalent) einer gegebenen Anfrage. Die Untersuchung der Grundoperationen liefert dann eine Antwort auf die Frage, welche Informationen für die Rekonstruktion der Quelldatenbank notwendig sind. Das Wissen über die Provenance des Anfrageergebnisses reicht in einigen Fällen für eine vollständige Rekonstruktion jedoch nicht aus. In diesen Fällen kann es dazu genutzt werden, die Quelldatenbank auf die für die korrekte Bearbeitung einer Anfrage notwendigen Tupel zu reduzieren. Der Resultat wäre eine Speicherplatzersparnis.

1.1. Einführung in die Thematik

Mit dem Begriff *Big Data* wird eine große Menge an Daten bezeichnet, die durch ein riesiges Volumen, eine große Vielfalt sowie eine extrem schnelle Verarbeitungsrate charakterisiert ist. Solche Datenmengen entstehen beispielsweise in sozialen Netzwerken oder großen Forschungseinrichtungen. Zur Charakterisierung von Big Data werden vier grundlegende Eigenschaften unterschieden: Volumen (*Volume*), Geschwindigkeit (*Velocity*), Vielfalt (*Variety*) sowie Wahrhaftigkeit (*Veracity*). Sie sind aufgrund ihres Umfangs in der Regel mit klassischen Mitteln nicht mehr zu verarbeiten [OVH17].

Neben diesen Eigenschaften werden am Lehrstuhl *Datenbanken und Informationssysteme* der Universität Rostock bei der Analyse von Big Data vier Schwerpunkte gesetzt: Datenschutz (*Privacy*), Langzeitarchivierung (*Preservation*), Analyseeffizienz (*Paralellization*) sowie die Datenherkunft (*Provenance*). Der Begriff der *Provenance* ist dabei in vielen Wissenschaften aktueller Forschungsgegenstand. So etwa in der Geologie [RSSL13], den Datenbanken [Heu15] sowie im Bereich der Modellierung und Simulation [RU17]. Er stammt aus dem englischen Sprachgebrauch und bedeutet wörtlich übersetzt Herkunft oder Ursprung. Im Sinne der Datenbanktheorie beschäftigt sich das *Provenance Management* mit der Rückverfolgbarkeit eines Ergebnisses bis zu den relevanten Originaldaten.

Typischerweise werden sowohl vier Provenance-Anfragen als auch vier Provenance-Antworten unterschieden. Die Anfragen ***where*** (Woher kommen die Daten?), ***why*** (Warum dieses Ergebnis?) und ***how*** (Wie kommt das Ergebnis zustande?) können dabei nach ihrem Informationsgehalt geordnet werden. Kann eine Art Formel — das sogenannte *Provenance-Polynom* — zur Berechnung des Anfrageergebnisses angegeben werden, so ist es möglich, den Datensatz auf die „relevanten“ Datensätze einzuschränken und dennoch ein korrektes Anfrageergebnis wiederzugeben. Wir können so, je nach Anfrage, die Roh- bzw. Quelldatenmasse erheblich reduzieren — durch die Angabe einer *(minimalen) Zeugenbasis*. Der Antworttyp ist dabei zunächst immer *extensional*, d. h. die Antwort auf eine Provenance-Anfrage ist stets eine Menge von (auf

bestimmte Attribute eingeschränkten) Tupeln der Originaldaten. Die ***why not***-*Anfrage* sowie die drei Antworttypen *intensional*, *anfragebasiert* und *modifikationsbasiert* werden hier nicht genauer analysiert. Hierfür sei etwa auf [LKLG17] oder [Sva16] verwiesen.

Ein praktisches Beispiel für die Anwendung von Provenance Management ist das Leibniz-Institut für Ostseeforschung in Warnemünde. Durch diverse Messungen, Analysen und Simulationen werden eine Vielzahl heterogener biologischer, chemischer, physikalischer und numerischer Daten erzeugt, die anschließend im Rahmen des Forschungsdatenmanagements langfristig gespeichert und verwaltet werden sollen. Fragen nach der Herkunft der Daten, ihrer Relevanz und der Rekonstruierbarkeit von Ergebnissen können nun mit Hilfe von Provenance-Anfragen beantwortet werden. So kann beispielsweise bei der Messung des Sauerstoffgehalts der Ostsee (siehe Abbildung 1.1) gezielt nach dem rot eingerahmten Ausreißer gefragt werden [Heu17]. Es handelt sich also um eine ***why***-Anfrage. Die Provenance-Antwort hierauf kann sowohl intensional, d. h. in Form einer Beschreibung, als auch extensional erfolgen.

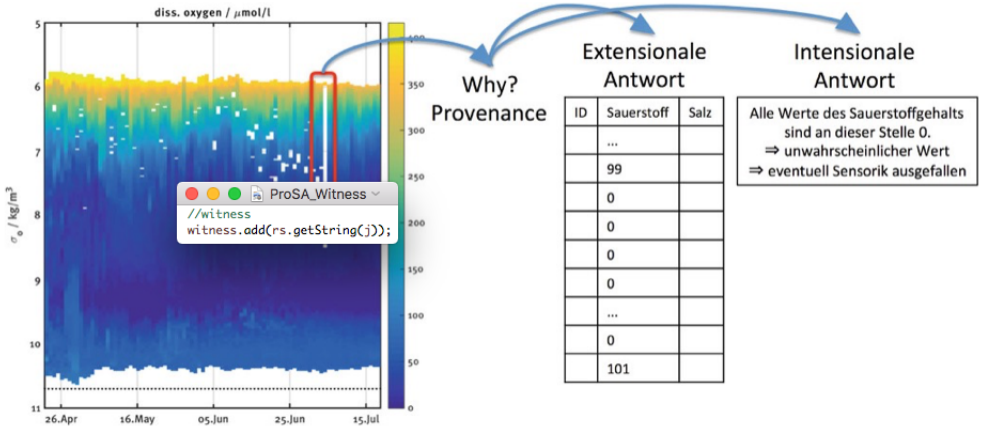


Abbildung 1.1. ***why***-Provenance am Beispiel des Sauerstoffgehalts der Ostsee [BKM+17]

Eine beim erstmaligen Stellen einer Anfrage automatische Rekonstruktion der für diese Anfrage relevanten Originaldaten, kann in Fällen der Privacy oder der schieren Datenmasse von großem Nutzen sein. So können etwa sensible Daten „verschleiert“ oder eine große Datenmenge stark verdichtet werden. Die hierfür nötigen *inversen Schemaabbildungen* werden zwar bzgl. ihrer Eigenschaften untersucht und im Bereich der Schema-Evolution erfolgreich angewendet, können aber auch in der Datenherkunft von Nutzen sein. Ein offenes Problem im Bereich der Rekonstruierbarkeit besteht nun in der Berechnung dieser inversen Abbildungen durch „Rettung“ von Annotationen.

Unter einer Schemaabbildung $\mathcal{M} = (S, S', \Sigma)$ verstehen wir hier insbesondere die Abbildung der Datenbankinstanzen [DHI12]. Die zugehörige inverse Schemaabbildung $\mathcal{M}^\star = (S', S, \Sigma)$ entspricht dann der Rückabbildung der Zielinstanz S' auf die Quellinstanz S. Beide Abbildungen gelten dabei unter einer gegebenen Menge von Abhängigkeiten Σ. Ebenfalls von großer Bedeutung für diese Arbeit ist der CHASE-*Algorithmus*. Er ist ein in der Datenbanktheorie häufig verwendetes Werkzeug, welches unter anderem zur Untersuchung von Anfrage- oder Datenbankäquivalenzen unter gegebenen Abhängigkeiten genutzt wird.

Ein Teilziel dieser Arbeit besteht in der Analyse der aktuellen Forschung in den Bereichen der Provenance-Anfragen und -Polynome, verschiedener CHASE-Varianten sowie der Definition von CHASE-inversen Schemaabbildungen. Hierbei interessieren uns insbesondere die Berechnung der Provenance-Polynome

[GKT07, ADT11] sowie die anschließende Ermittlung der (minimalen) Zeugenbasen [BKT01]. Für welche Grundoperationen können solche Polynome aufgestellt werden? Welche mathematischen Grundlagen werden hierfür benötigt? Als Basis für die Inversen-Suche soll uns etwa der CHASE-Algorithmus [FKPT11, FKMP03] für s-t tgds und egds dienen.

Wir wollen anschließend die bisher bekannten (*exakten / relaxten*) CHASE-*Inversen* um eine weitere Inverse, die *ergebnisäquivalente* CHASE-*Inverse* erweitern und die wichtigsten Operationen auf die Existenz dieser inversen Schemaabbildungen untersuchen. Dabei soll der CHASE-Algorithmus um die Idee der BACKCHASE-Phase aus dem Paper *Provenance-Directed Chase&Backchase* [DH13] erweitert und um den Aspekt der Provenance-Polynome und (minimalen) Zeugenbasen ergänzt werden.

Mit Hilfe dieser Erweiterungen können wir tatsächlich für jede Anfrage die zugehörige CHASE-inverse Funktion, sofern diese existiert, aufstellen und die Quelldaten auf die für die Anfrage relevanten Informationen reduzieren. Die Untersuchung der Grundoperationen liefert zudem eine Antwort auf die Frage, welche Informationen für die Rekonstruktion der Quelldatenbank notwendig sind. Das Wissen über die Provenance des Anfrageergebnisses reicht in einigen Fällen für eine vollständige Rekonstruktion jedoch nicht aus, wobei bei manchen Anfragetypen eine vollständige Rekonstruktion vielleicht gar nicht gefordert ist. Der nächste Schritt bestünde nun in einer computergestützten Umsetzung dieser Theorie. Ein erster Ansatz hierzu bildet den Abschluss unserer Untersuchungen.

1.2. Einführung des Beispieldatensatzes

Die theoretischen Grundlagen des zweiten Kapitels sowie die Definitionen und Methoden im weiteren Verlauf dieser Arbeit (vgl. Kapitel 2 sowie Kapitel 3) werden anhand des folgenden Beispiels motiviert und veranschaulicht. Es handelt sich um einen fiktiven Datensatz über das Vorlesungsangebot des Instituts für Informatik der Universität Rostock sowie die zugehörigen Studenten, Dozenten und Prüfungsergebnisse.

Das Beispiel besteht aus fünf Tabellen über die an der Universität eingeschriebenen Studenten (siehe Tabelle 1.1), die Liste der Prüfungsergebnisse (Tabelle 1.2), die belegbaren Module (Tabelle 1.3), die Dozentenliste (Tabelle 1.4) sowie die Liste der Modulteilnehmer (Tabelle 1.5) im Zeitraum des Wintersemesters 2014/2015 bis zum Sommersemester 2017. Der Übersicht halber sind die Tabellen 1.2 und 1.5 nur Ausschnitte der Originaltabellen B.5 sowie B.4. Alle Tabellen sind aber im Anhang unter Abschnitt B nachzulesen.

Jede Zeile dieser vier Tabellen 1.1 - 1.5 bzw. B.1 - B.5 ist mit einer eindeutigen Identifikationsnummer (ID) versehen, welche aus dem Anfangsbuchstaben des Tabellennamen sowie einer fortlaufenden Nummer besteht. Dabei wird jede ID nur einmal vergeben und nach dem Löschen einer Tabellenzeile nicht neu vergeben. Die ID dient als Tupelidentifikator und ist einzig für das Aufstellen der Zeugen sowie die Berechnung der Provenance-Polynome von Bedeutung.

Es folgen die Definitionen der einzelnen Relationenschemata und Domänen $\mathbb{D}$. Ein Relationenschema entspricht dabei der Menge der vorkommenden Attribute und eine Domäne dem Wertebereich eines Attributs. Beide Begriffe werden im Unterabschnitt 2.1.1 aber noch einmal gesondert eingeführt.

- STUDENTEN = { $\text{ID}_{\text{Studenten}}$, Matrikelnr, Name, Vorname, Studiengang} mit
  - $\mathbb{D}(\text{ID}_{\text{Studenten}}) := \{S_i : i \text{ fortlaufend nummeriert und } i \in \mathbb{N}^+ := \{1, 2, 3, ...\}\}$
  - $\mathbb{D}(\text{Matrikelnr}) := \mathbb{N}^+$
  - $\mathbb{D}(\text{Name}) := \{\text{Fieber, Sonnenschein, Müller, Johansen, Miller, Mustermann}\}$

  - $\mathbb{D}$(Vorname) := {Fabian, Sarah, Max, Mira, Johannes, Mia}
  - $\mathbb{D}$(Studiengang) := {Elektrotechnik, Informatik, Lehramt Informatik, Mathematik}

- NOTEN = { ID_{Noten}, Modulnr, Matrikelnr, Semester, Note} mit
  - $\mathbb{D}(\text{ID}_{\text{Ergebnisse}}) := \{N_i : i \text{ fortlaufend nummeriert und } i \in \mathbb{N}^+ = \{1, 2, 3, ...\}\}$
  - $\mathbb{D}$(Modulnr) := $\{xyz : xyz$ fortlaufend nummeriert und $x, y, z \in \{0, ..., 9\}$ mit $xyz \neq 000\}$
  - $\mathbb{D}$(Matrikelnr) := $\mathbb{N}^+$
  - $\mathbb{D}$(Semester) := {WS 14/15, SS 15, WS 15/16, SS 16, WS 16/17, SS 17}
  - $\mathbb{D}$(Note) := $\{1.0, 1.3, 1.7, 2.0, ..., 4.0, 5.0\}$
- MODULE = { $\text{ID}_{\text{Module}}$, Modulnr, Titel, Vertiefung} mit
  - $\mathbb{D}(\text{ID}_{\text{Module}}) := \{M_i : i \text{ fortlaufend nummeriert und } i \in \mathbb{N}^+ = \{1, 2, 3, ...\}\}$
  - $\mathbb{D}$(Modulnr) := $\{xyz : xyz$ fortlaufend nummeriert und $x, y, z \in \{0, ..., 9\}$ mit $xyz \neq 000\}$
  - $\mathbb{D}$(Titel) := {Datenbanken III, Mathematik für Informatiker 1, Mustererkennung und Kontextanalyse, Individuelles Wissensmanagement, Data Warehouses und Business Intelligence, Kognitive Systeme, Theorie relationaler Datenbanken, Systembiologie, NEidI — Neueste Entwicklungen in der Informatik}
  - $\mathbb{D}$(Vertiefung) := {Informationssysteme, Modelle und Algorithmen, Smart Computing, Visual Computing, Wirtschaftsinformatik, xxx[1]}
- DOZENTEN = { $\text{ID}_{\text{Dozenten}}$, Modulnr, Dozent} mit
  - $\mathbb{D}(\text{ID}_{\text{Dozenten}}) := \{M_i : i \text{ fortlaufend nummeriert und } i \in \mathbb{N}^+ = \{1, 2, 3, ...\}\}$
  - $\mathbb{D}$(Modulnr) := $\{xyz : xyz$ fortlaufend nummeriert und $x, y, z \in \{0, ..., 9\}$ mit $xyz \neq 000\}$
  - $\mathbb{D}$(Dozent) := {Dozent A, Dozent B, Professor A, Professor B, Professor C, Professor D, Professor E}
- TEILNEHMER = { $\text{ID}_{\text{Teilnehmer}}$, Modulnr, Matrikelnr} mit
  - $\mathbb{D}(\text{ID}_{\text{Teilnehmer}}) := \{T_i : i \text{ fortlaufend nummeriert und } i \in \mathbb{N}^+ = \{1, 2, 3, ...\}\}$
  - $\mathbb{D}$(Modulnr) := $\{xyz : xyz$ fortlaufend nummeriert und $x, y, z \in \{0, ..., 9\}$ mit $xyz \neq 000\}$
  - $\mathbb{D}$(Matrikelnr) := $\mathbb{N}^+$

Jeder Student ist über seine Matrikelnummer eindeutig identifizierbar und kann verschiedene Module belegen. Dabei ist er nicht gezwungen eine Prüfung abzulegen, kann aber bei Bedarf (Erhalt der Note 5.0) eine Prüfung im folgenden Semester wiederholen. Ein Modul kann von mehreren Dozenten und in verschiedenen Semestern gehalten werden.

Die Einschränkung der Domänen $\mathbb{D}$(Name) und $\mathbb{D}$(Vorname) in der Relation STUDENTEN dient hier der Übersicht. Allgemein würden diese durch $\mathbb{D}$(Name) = $\mathbb{D}$(Vorname) = String definiert werden. Im Fall Studiengang, Semester, Titel, Vertiefung und Dozent ist eine Einschränkung wegen der geringen Anzahl an Datensätzen sowie der Datenkonstanz jedoch sinnvoll. Änderungen erfolgen hier selten und nur in geringem Maße, sodass diese händisch erfolgen können.

[1] Platzhalter für Module, die in den hier angegebenen Vertiefungsrichtungen nicht enthalten sind

$\mathbf{ID_{Studenten}}$	**Matrikelnr**	**Name**	**Vorname**	**Studiengang**
S_1	1	Fieber	Fabian	Lehramt Informatik
S_2	2	Sonnenschein	Sarah	Mathematik
S_3	3	Müller	Max	Elektrotechnik
S_4	4	Müller	Mira	Informatik
S_5	5	Johansen	Johannes	Informatik
S_6	6	Miller	Mia	Informatik
S_7	7	Mustermann	Max	Elektrotechnik
S_8	8	Johannes	Paul	ITTI

Tabelle 1.1. STUDENTEN-Relation

$\mathbf{ID_{Noten}}$	**Modulnr**	**Matrikelnr**	**Semester**	**Note**
N_{21}	009	1	SS 16	3.3
N_{22}	009	5	SS 15	5.0
N_{23}	009	5	SS 16	2.7

Tabelle 1.2. NOTEN-Relation eingeschränkt auf das Modul 009 (Ausschnitt der Tabelle B.5)

$\mathbf{ID_{Module}}$	**Modulnr**	**Titel**	**Vertiefung**
M_1	001	Datenbanken III	Informationssysteme
M_2	002	Mathematik für Informatiker 1	xxx[2]
M_3	003	Mustererkennung und Kontextanalyse	Smart Computing
M_4	004	Individuelles Wissensmanagement	Informationssysteme
M_5	005	Data Warehouses und Business Intelligence	Wirtschaftsinformatik
M_6	006	Kognitive Systeme	Smart Computing
M_7	007	Theorie relationaler Datenbanken	Informationssysteme
M_8	008	Systembiologie	Modelle und Algorithmen
M_9	009	NEidI — Neueste Entwicklungen in der Informatik	xxx

Tabelle 1.3. MODUL-Relation

$\mathbf{ID_{Dozenten}}$	**Modulnr**	**Dozent**
$D_{1.1}$	001	Professor A
$D_{1.2}$	001	Dozent A
D_2	002	Professor B
D_3	003	Professor C
D_4	004	Professor D
D_5	005	Dozent B
D_6	006	Professor D
D_7	007	Professor A
D_8	008	Professor E
D_9	009	Professor A

Tabelle 1.4. DOZENTEN-Relation

[2] Platzhalter für Module, die in den hier angegebenen Vertiefungsrichtungen nicht enthalten sind

$\mathbf{ID_{Teilnehmer}}$	**Modulnr**	**Matrikelnr**
T_{23}	009	1
T_{24}	009	4
T_{25}	009	5

Tabelle 1.5. TEILNEHMER-Relation eingeschränkt auf das Modul 009
(Ausschnitt der Tabelle B.4)

Bemerkung. Abschließend noch dei Bemerkungen zu der Definition der Relationenschemata:

- Die unterstrichenen Attribute der Tabellen B.1 bis B.5 entsprechen den Schlüsseln der jeweiligen Relationenschemata.
- Die Tupelidentifikatoren sind in der Definition der Relationenschemata zwar enthalten, werden aber im Kapitel 2 sowie den Abschnitten 3.3.1 bis 3.4 vernachlässigt, da wir sie nur für die Beantwortung der Provenance-Anfragen benötigen. In diesen Fällen sind die Relationenschemata um das erste Attribut verkürzt:
  - STUDENTEN = {Matrikelnr, Name, Vorname, Studiengang}
  - NOTEN = {Modulnr, Matrikelnr, Semester, Note}
  - MODULE = {Modulnr, Titel, Vertiefung}
  - DOZENTEN = {Modulnr, Dozent}
  - TEILNEHMER = {Modulnr, Matrikelnr}
- Die Identifikatoren der Tupel einer Relation r sind definiert als der erste Buchstabe des Relationennamens mit fortlaufendem Index. Eine Ausnahme hierfür bilden die ersten beiden Tupel der Relation DOZENTEN. Hier werden zur besseren Übersicht die speziellen IDs $D_{1.1}$ und $D_{1.2}$ vergeben, da das Modul 001 als einziges Modul des Vorlesungsangebotes von zwei Dozenten gehalten wird.

Dieses fiktive Beispiel soll im weiteren Verlauf der Arbeit zur Veranschaulichung der eingeführten Definitionen und Methoden dienen. Es wird daher in den Kapiteln 2, 3 und 5 immer wieder referenziert.

1.3. Aufbau der Arbeit

Die Struktur der Arbeit ist in sechs Kapitel unterteilt. Nach der Einleitung (Kapitel 1) und der Ausführung aller zum Verständnis der Thematik notwendigen Grundlagen (Kapitel 2) folgen ein Überblick über den aktuellen Stand der Forschung im Bereich der Data Provenance und der CHASE-Technik, der inversen Schemaabbildungen (Kapitel 3) sowie eine Vorstellung des erarbeiteten Konzepts (Kapitel 4). Abschließend werden alle bisherigen Resultate am Beispiel des Hidden-Markov-Modells theoretisch (Kapitel 4) und in einem kurzen Programm zur Berechnung einfacher SQL-Anweisungen praktisch umgesetzt (Kapitel 5) und mit einem groben Ausblick abgeschlossen (Kapitel 6).

Die Analyse von Provenance-Anfragen mit Hilfe von inversen Schemaabbildungen basiert hier auf einem relationalen Datenbankmodell. Dieses wird über die Definition des Schemas, der Instanzen und Integritätsbedingungen im Grundlagenkapitel (vgl. Abschnitt 2.1) eingeführt und anschließend um die Operationen im relationalen Modell — relationale Operationen und OLAP-Operationen — erweitert. Es folgen

eine Einführung in das Gebiet der Big Data Analytics mit der Vorstellung der fünf Eigenschaften *Volume*, *Velocity*, *Variety*, *Value* und *Veracity* sowie den Analysen der *Privacy*, *Preservation*, *Parallelization* und *Provenance* (siehe Abschnitt 2.3). Den Schwerpunkt der Arbeit bilden die Provenance-Anfragen und -Anworten sowie die Erarbeitung einer inversen oder pseudoinversen Schemaabbildung, welche in den folgenden Abschnitten kurz motiviert und im dritten Kapitel (vgl. Abschnitt 3.1 und 3.4) ausführlich untersucht werden. Als Werkzeug hierfür dienen unter anderem der CHASE-Algorithmus nach D. Maier [Mai83] sowie A. V. Aho, C. Beeri und J. D. Ullman [ABU79], der CHASE für tgds und egds nach R. Fagin et al. [FKMP03] und der CHASE&BACKCHASE-Algorithmus von R. Hull und A. Deutsch [DH13]. Sie werden in den Abschnitten 2.6 sowie 3.3 eingeführt und veranschaulicht. Die Berechnung der sogenannten Provenance-Polynome zur Angabe der ***how***-Provenance wird im zweiten Abschnitt des dritten Kapitels vorgestellt. Sie sind eng mit der (minimalen) Zeugenbasis verbunden, welche im weiteren Verlauf der Arbeit ebenfalls einen großen Stellenwert einnehmen wird. Die aus den Polynomen sowie der Zeugenbasis gewonnenen Informationen können bei Datenverlust für die Rekonstruktion von Teilausschnitten der Quelldatenbank zu Rate gezogen werden.

Im vierten Kapitel wollen wir das Konzept der CHASE-inversen Abbildungen weiterentwickeln und vertiefen. Wir definieren hierfür eine neue Inversenfunktion, die *ergebnisäquivalente* CHASE-*Inverse*, sowie ein CHASE&BACKCHASE-Verfahren mit Provenance-Antwort-Interpretation, welches den Existenznachweis der (exakten / relaxten / ergebnisäquivalenten) CHASE-Inversen unterstützen soll. Zudem bestimmen wir die hinreichenden und notwendigen Bedingungen für die Existenz der verschiedene CHASE-Inversen-Typen (siehe Abschnitt 4.1). Anschließend untersuchen wir die relationenalgebraischen Grundoperationen auf ihre Inversen-Typen (vgl. die Unterabschnitte 4.2.4 und 4.2.5) sowie die Komposition dieser Operationen (siehe Unterabschnitt 4.2.7). Den Abschluss bildet im Abschnitt 4.3 die Analyse des Hidden-Markov-Modells als Vertreter der Machine-Learnung-Algorithem.

Eine erste praktische Umsetzung der theoretischen Überlegungen des vierten Kapitels erfolgt im Kapitel 5. Hierfür wird das in einem studentischen Projekt der Universität Rostock entwickelte Programm **ProSA** um die Ausgabe einer Zeugenliste, die Angabe des CHASE-Inversen-Typs sowie die Rekonstruktion der Quelldatenbank erweitert. Den Abschluss dieser Arbeit bildet schließlich ein Ausblick über weitere interessante Fragestellungen im Bereich Provenance-Anfragen in Big-Data-Analytics-Umgebungen (siehe Abschnitt 6.2).

2. Grundlagen

Die vorliegende Arbeit beschäftigt sich mit der *Umsetzung von Provenance-Anfragen in Big-Data-Analytics-Umgebungen*. Ziel ist die Übertragung bisheriger Provenance-Anfragetechniken auf komplexere Anfragen in der Big-Data-Welt. Dies geschieht beispielsweise durch die Untersuchung inverser Schemaabbildungen. Hierzu müssen zunächst einige Grundbegriffe wie das relationale Datenbankmodell (vgl. Abschnitt 2.1) sowie die hierauf definierten Standard- (Projektion, Selektion, Verbund, usw.) und OLAP-Operationen eingeführt werden. Anschließend wird im dritten Abschnitt der Begriff Big Data Analytics präzisiert. Nach einer kurzen Einführung im vierten und fünften Abschnitt dieses Kapitels wird in Kapitel 3 die aktuelle Thematik der inversen Schemaabbildung in der Provenance-Analyse ausführlich diskutiert. Den Abschluss des Grundlagenkapitels bildet der CHASE-Algorithmus, mit dessen Hilfe inverse Schemaabbildungen „berechnet“ werden können.

2.1. Das relationale Datenbankmodell

Bevor das relationale Datenbankmodell eingeführt werden kann, muss das Wort Datenbankmodell zunächst präzisiert werden. M. Brodie beschrieb bereits 1982 das **Datenbankmodell** zur Darstellung einer Informationsstruktur [Bro82] mittels

- statischen Eigenschaften wie Objekte, Eigenschaften von Objekten und Beziehungen zwischen Objekten,
- dynamischen Eigenschaften wie Operationen auf Objekten, Eigenschaften dieser Operationen und Beziehungen zwischen Operationen,
- Integritätsbedingungen über Objekten (statische Integritätsbedingungen) und über Operationen (dynamische Integritätsbedingungen).

Hierzu sollen nun die Begriffe *Schema* (statische Eigenschaften) und *Instanz* (Datenbankobjekte) sowie einige grundlegende Integritätsbedingungen konkret definiert werden. Die Definitionen und Aussagen dieses Abschnitts basieren dabei auf dem Lehrbuch *The Theory of Relational Databases* von D. Maier, siehe [Mai83].

2.1.1. Schemata und Instanzen

Mit der Einführung des Relationenschemas, der Relation, des Datenbankschemas sowie der Datenbank sind Schema- und Instanzdefinition des relationalen Datenbankmodells gegeben. Dabei ist stets eine Relation Instanz eines Relationenschemas und eine Datenbank Instanz eines Datenbankschemas.

Definition 2.1 (Relationenschema)**.** Für gegebenes $n \in \mathbb{N}^+$ ist ein **Relationschema** $\mathcal{R}$ eine Menge von Attributen $\{A_1, A_2, ..., A_n\}$. Zu jedem Attribut A_i $(1 \leq i \leq n)$ existiert eine beliebige, nicht-leere, endliche (oder abzählbar unendliche) Menge D_i, die **Domäne** von A_i. Bezeichnet sei diese Domäne mit $\mathbb{D}(A_i)$. $\boxdot$

Definition 2.2 (Relation)**.** Eine **Relation** $r(\mathcal{R})$ eines Relationenschemas $\mathcal{R} = \{A_1, ..., A_n\}$ ist eine endliche Menge totaldefinierter Abbildungen

$$t : \mathcal{R} \to \bigcup_{i=1}^{n} D_i$$

mit $t(A_i) \in \mathbb{D}(A_i)$, wobei $1 \leq i \leq n \in \mathbb{N}^+$ ist. Diese Abbildungen t werden auch **Tupel** genannt. ⊡

Beispiel 2.1. *Das der Tabelle B.1 zugrunde liegende Relationenschema lautet*

$$\text{STUDENTEN} = \{\text{ID}_{\text{Studenten}}, \text{Matrikelnr.}, \text{Name}, \text{Vorname}, \text{Studiengang}\}.$$

Die Relation der Studententabelle besteht aus sieben Tupeln. Eines von ihnen ist definiert durch $t(\text{ID}_{\text{Student}}) = S_4$*,* $t(\text{Matrikelnr}) = 4$*,* $t(\text{Name}) =$ *Müller,* $t(\text{Vorname}) =$ *Mira,* $t(\text{Studiengang}) =$ *Informatik.*

Definition 2.3 (Datenbankschema)**.** Für gegebenes $n \in \mathbb{N}$ heißt eine Menge S von Relationenschemata $\mathcal{R}_1, ..., \mathcal{R}_n$ **Datenbankschema**; kurz: $S := \{\mathcal{R}_1, ..., \mathcal{R}_n\}$. ⊡

Definition 2.4 (Datenbank)**.** Eine **Datenbank** d über einem Datenbankschema $S := \{\mathcal{R}_1, ..., \mathcal{R}_n\}$ ist eine Menge von Relationen $d := \{r_1, ..., r_n\}$ mit $n \in \mathbb{N}^+$ und $r_i(\mathcal{R}_i)$ für alle $1 \leq i \leq n$. ⊡

2.1.2. Integritätsbedingungen

Neben Relationen- und Datenbankschemata sowie den zugehörigen Instanzen benötigt das relationale Datenbankmodell Integritätsbedingungen. Mögliche Integritätsbedingungen sind beispielsweise Schlüssel, funktionale Abhängigkeiten, mehrwertige Abhängigkeiten, Verbundabhängigkeiten sowie Inklusionsabhängigkeiten, siehe [Mai83]. Für die in dieser Arbeit untersuchte Fragestellung werden als Bedingungen nur Schlüssel sowie zwei bestimmte Abhängigkeiten zwischen verschiedenen Attributen und Attributmengen zugelassen. Grund hierfür ist der in Abschnitt 2.6 vorgestellte CHASE-Algorithmus, welcher ausschließlich funktionale Abhängigkeiten und Verbundabhängigkeiten benötigt.

Definition 2.5 (Schlüssel)**.** Eine **identifizierende Attributmenge** einer Relation $r(\mathcal{R})$ eines Relationenschemas $\mathcal{R}$ ist eine Teilmenge von Attributmengen $\mathcal{K} = \{B_1, B_2, ..., B_k\} \subseteq \mathcal{R}$, sodass gilt:

$$\forall t_1, t_2 \in r : t_1 \cap t_2 = \emptyset \Rightarrow \exists B \in \mathcal{K} : t_1(B) \neq t_2(B).$$

Eine bzgl. $\subseteq$ minimale identifizierende Attributmenge heißt **Schlüssel**. Unterschieden werden zudem der **Primärschlüssel** (beim Datenbankentwurf ausgezeichneter Schlüssel) sowie der **Fremdschlüssel** (Attributmenge, die in einer anderen Relation Schlüssel ist). ⊡

Beispiel 2.2. *Die Matrikelnummer (bzw. das Attribut* Matrikelnr*) ist Schlüssel der* STUDENTEN*-Relation B.1. Die Kombination aus Matrikel- und Modulnummer ist Schlüssel der* NOTEN*-Relation B.5. Jeder Schlüssel ist nach obiger Definition zusätzlich auch eine identifizierende Attributmenge.*

Die für die Definition der funktionale Abhängigkeit und der Verbundabhängigkeit notwendigen relationalen Operationen folgen im Unterabschnitt 2.2.1. Es handelt sich um die Projektion π, die Selektion σ und den natürlichen Verbund $\bowtie$.

Definition 2.6 (Funktionale Abhängigkeit (FD))**.** Eine Relation r genügt der **funktionalen Abhängigkeit** $X \to Y$, wenn für die Attributmengen $X, Y \subseteq \mathcal{R}$ und alle Konstanten $c \in \mathbb{D}(X)$ gilt:

$$\mid \pi_Y(\sigma_{X=c}(r)) \mid \leq 1.$$

Umgangssprachlich ausgedrückt: Die erste Attributmenge X bestimmt die zweite Attributmenge Y eindeutig. ⊡

Beispiel 2.3. *Die Relation der Notentabelle B.5 genügt der funktionalen Abhängigkeit*

$$(\text{Modulnr}, \text{Matrikelnr}, \text{Semester}) \rightarrow \text{Note}.$$

Das Attribut Note wird also durch die Kombination aus Modul- und Matrikelnummer sowie dem Semester der Prüfungsleistung eindeutig bestimmt.

Definition 2.7 (Verbundabhängigkeit (JD))**.** Eine **Verbundabhängigkeit** ist ein Ausdruck der Form $\bowtie [\mathcal{R}_1, ..., \mathcal{R}_n]$, wobei $\{\mathcal{R}_1, ..., \mathcal{R}_n\}$ ein gegebenes Datenbankschema ist. Dann genügt eine Relation r der JD $\bowtie [\mathcal{R}_1, ..., \mathcal{R}_n]$ genau dann, wenn

$$r = \pi_{\mathcal{R}_1}(r) \bowtie ... \bowtie \pi_{\mathcal{R}_n}(r).$$

⊡

Beispiel 2.4. *Für das der Tabelle 2.3 zugrunde liegende Datenbankschema entspricht die Relation r der JD* $\bowtie [(\text{Modulnr}, \text{Semester}), (\text{Modulnr}, \text{Dozent})]$. *Denn es gilt:*

Modulnr	**Semester**
001	SS 16
002	WS 15/16
003	WS 16/17
004	WS 16/17
005	SS 17
006	SS 17
007	SS 16
008	
009	SS 15
009	SS 16

Tabelle 2.1. Ergebnis der Teilanfrage $\pi_{\text{Modulnr,Semester}}(r)$

Modulnr	**Dozent**
001	Professor A
001	Dozent A
002	Professor B
003	Professor C
004	Professor D
005	Dozent B
006	Professor D
007	Professor A
008	Professor E
009	Professor A

Tabelle 2.2. Ergebnis der Teilanfrage $\pi_{\text{Modulnr.,Dozent}}(r)$

und nach Verbundbildung:

Modulnr.	**Semester**	**Dozent**
001	SS 16	Professor A
001	SS 16	Dozent A
002	WS 15/16	Professor B
003	WS 16/17	Professor C
004	WS 16/17	Professor D
005	SS 17	Dozent B
006	SS 17	Professor D
007	SS 16	Professor A
008		Professor E
009	SS 15	Professor A
009	SS 16	Professor A

Tabelle 2.3. Ergebnis der Anfrage $\pi_{\text{Modulnr.,Semester}}(r) \bowtie \pi_{\text{Modulnr.,Dozent}}(r)$

Im Allgemeinen muss nach einem Verbund der Schlüssel nicht übernommen werden. Da es sich in diesem Beispiel um eine $1 : n$*-Beziehung handelt, bleibt der Schlüssel aber erhalten.*

2.2. Die Operationen im relationalen Modell

Neben der strukturellen Beschreibung einer Datenbank werden zusätzlich noch Operationen für das Suchen, Ändern und Interpretieren des Datenbankbestandes benötigt. Hierfür sollen zunächst einige relationale Operationen und anschließend die OLAP-Operationen eingeführt werden, welche im späteren Verlauf der Arbeit ihre Anwendung finden werden.

2.2.1. Relationale Operationen

Die 1970 von E. F. Codd eingeführte Relationenalgebra [Cod70] ist die beliebteste Grundlage relationaler Anfragesprachen. Die für die vorliegende Arbeit relevanten Operationen basieren auf den Definitionen und Aussagen des Lehrbuchs *Theory of Relational Databases* von D. Maier, siehe [Mai83]. Dies sind unter anderem die Selektion, die Projektion, der natürliche Verbund sowie die Vereinigung von Tupeln, ebenso wie die Umbenennung und die klassischen Aggregatfunktionen.

Bei der Auswahl bestimmter Tupel, der sogenannten Selektion, wird die Konstanten- und die Attribut-Selektion unterschieden. So werden die in einer Datenbank enthaltenen Daten erst durch die Selektion „nutzbar“. Dabei werden die Datenobjekte nach ihren Eigenschaften, d.h. mittels einer der Vergleichsrelation $<, \leq, =, \geq, >$, oder $\neq$, ausgewählt.

Definition 2.8 (Konstanten-Selektion)**.** Die unäre Operation **Konstanten-Selektion** liefert, angewendet auf eine Relation r, eine zweite Relation $r' \subseteq r$. Dabei wird ein bestimmtes Attribut $A \in \mathcal{R}$ mittels $A\theta c$ auf einen konstanten Wert c mit $\theta \in \{<, \leq, =, \neq, \geq, >\}$ eingeschränkt. Es gilt also:

$$\sigma_{A\theta c} = \{t \mid t \in r \wedge t(A)\theta c\}.$$

□·

Definition 2.9 (Attribut-Selektion)**.** Die **Attribut-Selektion** von $r(\mathcal{R})$ nach $A_i\theta A_j$ mit $A_i, A_j \in \mathcal{R}$, $\theta \in \{<, \leq, =, \geq, >, \neq\}$ und $i \neq j$ für $i, j \in \mathbb{N}^+$ ist definiert als

$$\sigma_{A_i\theta A_j}(r) := \{t \mid t \in r \wedge t(A_i)\theta t(A_j)\}.$$

□·

Bemerkung. Sind die Mengen A_i und A_j mit $i, j \in \mathbb{N}^+$ und $i \neq j$ nicht linear geordnet, so gilt $\theta \in \{=, \neq\}$. Die Anfragen $\sigma_{A=c}$ und $\sigma_{A_i=A_j}$ heißen auch **Gleichheitsanfragen**.

Neben der Selektion σ zählen die Projektion und der natürliche Verbund ebenso zu den relationalen Standard-Operationen. Statt einigen ausgewählten Zeilen einer Relation liefert die Projektion eine Teilmenge von Spalten der Relation. Der natürliche Verbund hingegen verbindet zwei Relationen über die gemeinsamen Attribute, in der Praxis oft über die Schlüssel-Fremdschlüssel-Beziehung. Haben beide Relationen kein gemeinsames Attribut, ergibt sich analog zur Vektorrechnung das kartesische Produkt beider Relationen.

Definition 2.10 (Projektion)**.** Mit der **Projektion** π als unäre Operation können Spalten X einer bestehenden Relation r ausgewählt werden. Formal heißt dies:

$$\pi_X(r) := \{t(X) \mid t \in r\}.$$

□

Definition 2.11 (Natürlicher Verbund)**.** Der **natürliche Verbund** $\bowtie$ von r_1 und r_2 ist definiert durch

$$r_1 \bowtie r_2 = \{t \mid t \text{ Tupel über } \mathcal{R}_1 \cup \mathcal{R}_2 \wedge [\forall i \in \{1,2\} : \exists t_i \in r_i : t_i = t(\mathcal{R}_i)]\}.$$

Im Falle $\mathcal{R}_1 \cap \mathcal{R}_2 = \emptyset$ heißt der natürliche Verbund auch **kartesisches Produkt**. □

Relationen werden im Allgemeinen als Menge von Tupeln aufgefasst. Die Tupel selbst sind bzgl. ihrer Attribute geordnet. Die Operationen Durchschnitt, Vereinigung und Differenz finden daher ebenfalls ihre Anwendung.

Definition 2.12 (Mengenoperationen)**.** Die Mengenoperationen **Durchschnitt** $\cap$, **Vereinigung** $\cup$ und **Differenz** $-$ zweier Relationen $r_1(\mathcal{R})$ und $r_2(\mathcal{R})$ über das selbe Relationenschema $\mathcal{R}$ sind definiert als

$$r_1 \cap r_2 := \{t \mid t \in r_1 \wedge t \in r_2\},$$

$$r_1 \cup r_2 := \{t \mid t \in r_1 \vee t \in r_2\},$$

$$r_1 - r_2 := \{t \mid t \in r_1 \wedge t \notin r_2\}.$$

□

Die Umbenennung gehört nach G. Saake, K.-U. Sattler und A. Heuer nicht nur zu den relationalen Basisoperationen, sondern ist Bestandteil einer minimalen Relationenalgebra. Grund hierfür ist ihr Einfluss sowohl auf den natürlichen Verbund — und somit auf das kartesische Produkt — als auch auf die Mengenoperationen aus Definition 2.12. Die Definition der Umbenennung basiert daher ebenso wie Definition der Mengenoperationen auf dem Buch *Datenbanken — Konzepte und Sprachen* [SSH13].

Definition 2.13 (Umbenennung)**.** Für zwei Attribute $A_i \in \mathcal{R}$ und $A_j \notin \mathcal{R} - \{A_i\}$ mit $\mathbb{D}(A_i) = \mathbb{D}(A_j)$ und $\mathcal{R}' := (\mathcal{R} - \{A_i\}) \cup \{A_j\}$ ist die **Umbenennung** β von A_i zu A_j in der Relation $r(\mathcal{R})$ folgendermaßen bestimmt:

$$\beta_{A_j \leftarrow A_i}(r) := \{t' \mid \exists t \in r : t'(\mathcal{R} - \{A_i\}) = t(\mathcal{R} - \{A_i\}) \wedge t'(A_j) = t(A_i)\}.$$

□

Im zehnten Kapitel des Lehrbuchs definieren die Autoren die klassischen Aggregatfunkionen **MIN**, **MAX**, **COUNT**, **SUM** und **AVG** [SSH13]. Diese können je nach Anwendungsfall noch erweitert werden. Der SQL:2011-Standard bietet zusätzlich beispielsweise die Standardabweichung sowie die Varianz [Mel11].

Definition 2.14 (Aggregatfunktionen)**.** Unter die **klassischen Aggregatfunktionen** fallen die fünf nachfolgenden Funktionen **MIN**, **MAX**, **COUNT**, **SUM** und **AVG** mit

- **MIN**() bzw. **MAX**(): berechnet den kleinsten bzw. größten Wert einer Spalte;
- **COUNT**(): berechnet die Anzahl der Werte einer Spalte;
- **SUM**(): berechnet die Summe aller Werte einer Spalte, sofern für die betrachtete Spalte ein numerischer Wertebereich vorliegt;

- **AVG**(): berechnet das arithmetische Mittel der Werte einer Spalte, sofern für die betrachtete Spalte ein numerischer Wertebereich vorliegt.

□·

Bemerkung. Der Spezialfall **COUNT(⋆)** der Aggregatfunktion COUNT berechnet die Anzahl der Tupel einer Relation $r(\mathcal{R})$ über einem gegebenen Relationenschema $\mathcal{R}$.

Obige Aggregatfunktionen können auf mehrere Spalten erweitert sowie mit den Worten ALL oder DISTINCT versehen werden. Im Falle mehrerer Spalten werden nicht mehr einzelne Werte sondern ganze Tupel untersucht. Das zusätzliche Wortsymbol DISTINCT bedeutet eine Duplikateliminierung vor der Anwendung der Aggregation. Eine Anfrage nach der Anzahl der angebotenen Module der Universität Rostock (vgl. Tabelle B.4)

Anfrage 2.1 Anzahl an Modulteilnehmern

```
SELECT COUNT DISTINCT Modulnr.
FROM TEILNEHMER
```

liefert beispielsweise neun Module. In der Algebra selbst liegt die Duplikateleminierung immer vor, in SQL ist diese jedoch kein Standard. Wir wollen bei unseren Untersuchungen daher immer den Fall mit DISTINCT, d.h. mit Duplikateleminierung, betrachtet.

Aggregatfunktionen wie **MAX**, **MIN**, **COUNT**, **SUM** oder **AVG** werden häufig in Kombination mit der **GROUP BY**-Anweisung verwendet. Dies ermöglicht eine Gruppierung der Relation mit anschließender Auswertung. So kann im Fall der NOTEN-Relation beispielsweise die Durchschnittsnote pro Modul berechnet werden (siehe Beispiel 2.5 unten).

Definition 2.15 (Gruppierung)**.** Seien $r(\mathcal{R})$ eine Relation über einem Relationenschema $\mathcal{R} = \{A_1, ..., A_n\}$, $F_1(A_1), ..., F_n(A_n)$ eine Liste von Aggregatfunktionen und $G_1, ..., G_m$ eine Liste von Gruppierungsattributen. Dann wird das Ergebnis der **Gruppierung** $\gamma_{G_1,...,G_m;F_1(A_1),...,F_n(A_n)}(r(\mathcal{R}))$ wie folgt konstruiert:

- Partitioniere $r(\mathcal{R})$ in Gruppen, sodass jedes Tupel einer Gruppe im Gruppierungsattribut den gleichen Wert besitzt. Ist kein solches Attribut angegeben, existiert nur die Gruppe $r(\mathcal{R})$.
- Erzeuge für jede Gruppe ein Tupel bestehend aus dem Wert des Gruppierungsattributes und dem mittels $F_i(A_i)$ aggregierten Wert der Tupel dieser Gruppe.

□·

Anfrage 2.2 Durchschnittsnoten der einzelnen Module

```
SELECT Modulnr, AVG(Note)
FROM NOTEN
GROUP BY Modulnr
```

Beispiel 2.5. *Für die obige SQL-Anfrage 2.2 ergibt sich der relationenalgebraische Ausdruck*

$$\pi_{\text{Modulnr, }\mathbf{AVG}(\text{Note})}\ \gamma_{\text{Modulnr; }\mathbf{AVG}(\text{Note})}(\text{NOTEN})$$

und somit die folgende Ergebnistabelle:

Modulnr	AVG(Note)
001	2.1
002	2.0
003	1.0
004	2.1
005	2.2
006	3.3
007	2.0
009	3.6

Tabelle 2.4. Durchschnittsnoten der einzelnen Module

2.2.2. OLAP-Operationen

In vielen Anwendungen wird zwischen der operationalen Verwendung (OLTP) — dem eigentlichen Tagesgeschäft — und der analytischen Auswertungen (OLAP), meist in regelmäßigen Abständen, — täglich, wöchtenlich, monatlich , quartalsmäßig oder jährlich — unterschieden. Das Online Transactional Processing, kurz **OLTP**, ist in klassischen operativen Informationssystemen für den Einsatz im Tagesgeschäft vorherrschend, da es große Datensätze sowohl erfasst als auch verwaltet. Es zeichnet sich durch eine Vielzahl kurzer Online-Transaktionen aus wie etwa INSERT, UPDATE oder DELETE. Der Schwerpunkt in OLTP-Systemen liegt auf der sehr schnellen Abfrageverarbeitung, die Aufrechterhaltung der Datenintegrität in Multi-Access-Umgebungen sowie einer hohen Transaktionsrate. Das Online Analytical Processing, kurz **OLAP**, hingegen dient der Analyse großer Datenmengen. Es ist durch ein relativ geringes Transaktionsvolumen, sehr komplexen Anfragen — häufig mit Aggregation — sowie ein sehr hohes Datenaufkommen charakterisiert. Unterstützt werden OLAP-Anfragen in der Regel durch ein Data Warehouse, das mit Daten der unterschiedlichen operativen Systeme versorgt wird. Das Ziel eines OLAP-Systems besteht darin, durch die multidimensionale Betrachtung dieser Daten ein entscheidungsunterstützendes Analyseergebnis zu gewinnen.

Die hier vorgestellten OLAP-Operationen beruhen auf den Erläuterungen aus dem Lehrbuch *Data Warehouse Technologien* von V. Köppen, G. Saake und K.-U. Sattler, siehe [KSS14]. Für die Definition und Funktionsweise eines Data Warehouses sei ebenfalls auf dieses Lehrbuch verwiesen.

Die OLAP-Operationen sind auf einem multidimensionalen Datenmodell definiert. Im Mittelpunkt dieses Modells stehen betriebswirtschaftliche **Kennzahlen** wie der Gewinn, Verlust oder Umsatz eines Unternehmens. Die unterschiedlichen Perspektiven, aus denen eine solche Kennzahl betrachtet werden kann, heißen auch **Dimensionen** des multidimensionalen Modells. Sie bilden die Koordinatenachsen des mehrdimensionalen Raums. Für zwei Dimensionen ergibt sich eine Tabelle, für drei Dimensionen ein Würfel und allgemein für n Dimensionen ein n-dimensionaler Hyperwürfel. Die n Dimensionen des **Datenwürfels** (engl. hyper cube) sind üblicherweise von unterschiedlicher Größe und Aufteilung. Es handelt sich daher genau genommen um einen Hyperquarder. In der Praxis bestehen diese Datenwürfel aber häufig aus drei oder vier Dimensionen. Die Zellen eines solchen Hyperwürfels werden durch die einzelnen Kennzahlen gebildet. Zur Unterstützung detaillierterer Auswertungen werden die Dimensionen in weitere Ebenen, die **Hierarchien**, unterteilt. So kann die Zeitdimension beispielsweise als Hierarchie

$$\text{Tag} \rightarrow \text{Monat} \rightarrow \text{Quartal} \rightarrow \text{Jahr}$$

modelliert werden.

Die vier wichtigsten Operationen zur Navigation entlang der Hierarchien, der Bildung von Teilmengen, der Rotation des Würfels sowie dem Wechsel zwischen zwei Datenwürfeln sollen im Folgenden anhand des Beispieldatenwürfels aus Abbildung 2.1 erläutert werden. Dieser besteht aus den drei Dimensionen *Zeitraum*, *Region* und *Produkt* sowie obiger Zeithierarchie.

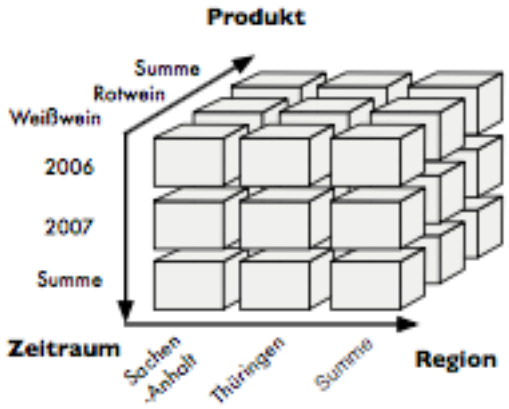


Abbildung 2.1. Beispiel für einen Datenwürfel [KSS14]

- Navigation entlang der Hierarchien: **DRILL DOWN** und **ROLL UP**
  DRILL DOWN entspricht dem „Hineinnavigieren“ in den Würfel. Dabei wird die aggregierte Darstellung auf ein feineres Gitter herunter gebrochen. Mit anderen Worten: Eine Dimension des Würfels wird detaillierter dargestellt. So können in Abbildung 2.1 statt des Jahres 2006 mittels DRILL DOWN die vier Quartale Q1, ..., Q4 direkt betrachtet werden. ROLL UP bildet die Inverse zu DRILL DOWN. Sie würde die Quartale Q1, Q2, Q3 und Q4 wieder zum Jahr 2006 aggregieren.

- Bildung von Teilmengen: **SLICE** und **DICE**
  Diese Operationen dienen der Bildung von Teilstrukturen des Würfels. So „schneidet“ SLICE eine Scheibe aus dem Würfel heraus (Dimensionsreduktion um eins), während DICE durch eine mehrdimensionale Bereichsselektion einen Teilwürfel liefert (Dimensionserhalt). Für obiges Beispiel kann der Würfel mittels SLICE beispielsweise auf den Verkauf von Weißwein eingeschränkt werden.

- Pivotisierung bzw. die Rotation des Würfels: **PIVOT** oder **ROTATE**
  ROTATE dient der Datenanalyse aus verschiedenen Perspektiven. Dazu werden durch Drehen des Würfels die Dimensionsachsen vertauscht, sodass eine passendere Visualisierung möglich ist. Statt den Verkaufszahlen der einzelnen Bundesländer im Jahr 2006, könnten für dieses Jahr die Verkaufszahlen pro Produkt angezeigt werden.

- Wechsel zwischen Datenwürfeln: **DRILL ACROSS**
  Wird eine Kennzahlen durch unterschiedliche Würfel modelliert, kann durch DRILL ACROSS zwischen diesen Würfeln gewechselt werden.

Zur genaueren Analyse oder konkreten Berechnung von Kennzahlen können neben den obigen OLAP-Operationen auch Aggregat- und Fensterfunktionen sowie Gruppierungen verwendet werden. Die Kombination von Aggregation und Gruppierung liefert beispielsweise eine Möglichkeit zur Ermittlung des Umsatzes eines Unternehmens. Ist nur ein abgesteckter Bereich (z. B. Zeitintervall oder Region) von Interesse, bieten sich Fensterfunktionen für allgemeine Bereichsanfragen an. Hier ist ebenfalls eine Verknüpfung mit den Aggregatfunktionen möglich.

- Aggregatfunktionen: Neben den obigen Standardfunktionen **MIN**, **MAX**, **SUM**, **COUNT** und **AVG** zählen zu den OLAP-Operationen auch einige Funktionen zur komplexeren statistischen Auswertung:

Die Berechnung der Varianz, der Standardabweichung, der Kovarianz, des Korrelationskoeffizienten sowie der Regressionsanalyse. Da diese jeweils aus den fünf Standardoperationen ableitbar sind, werden sie hier nicht weiter betrachtet. In anderen Anwendungsfällen, wie beispielsweise der Anfragezerlegung auf vorberechneten Aggregaten [GH16b], können die „erweiterten“ Aggregatfunktionen allerdings nicht vernachlässigt werden. Als Beispiel sei die Kovarianz zweier Variablen x und y gegeben. Sie ist als n-fache Summe von jeweils zwei Produkten definiert durch:

$$s(x, y) = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y}).$$

Dabei entspricht $\bar{x}$ dem Mittelwert der Variable x.

- **GROUP BY**-Erweiterungen: **GROUP BY** liefert eine Gruppierung aufgrund der Ergebnisgranularität. Unterschieden werden dabei die intradimensionale ROLLUP-Operation, die interdimensionale CUBE-Operation und die konkrete Angabe einer Gruppierung mittels GROUPING SETS. Genauer:

  – **ROLLUP**: Zu einer gegebenen Attributliste, beispielsweise A_1, A_2, A_3, wird eine intradimensionale Attributkombination ausgegeben, bei der jeweils das nächste Attribut der Attributliste der Gruppierung hinzugefügt wird:

  $$(), (A_1), (A_1, A_2), (A_1, A_2, A_3)$$

  – **CUBE**: Der CUBE-Operator fungiert ähnlich der Potenzmenge: Aus einer Grundmenge von Attributen werden alle möglichen Attributkombinationen generiert, über die aggregiert werden kann. Es handelt sich also um eine „Kurzform“ zur Berechnung von Teil- und Gesamtsummen. Für drei Attribute A_1, A_2 und A_3 gilt also:

  $$\begin{aligned} &(), \\ &(A_1), (A_2), (A_3), \\ &(A_1, A_2), (A_1, A_3), (A_2, A_3), \\ &(A_1, A_2, A_3) \end{aligned}$$

  – **GROUPING SETS**: Die Angabe konkreter Gruppierungen ist durch die GROUPING SETS-Operation möglich. Die mittels

  ```
  GROUP BY GROUPING SETS ((A_1, A_2), (A_3))
  ```

  angegebenen Attributgruppen der Attributliste A_1, A_2, A_3 werden genau so, d.h. als (A_1, A_2) und (A_3), ausgegeben.

- Fensterfunktion: Zur Beantwortung komplexer OLAP-Anfragen können OLAP-Funktionen über sogenannten Fenstern ausgeführt werden. Fenster spezifizieren Folgen von Tupeln mit einer definierten Ordnung. Ein mögliches Zeitfenster ist beispielsweise ein 7-Tagesintervall um ein konkretes Datum. Sei hierzu die Notentabelle B.5 erweitert um das Attribut Datum. Dann liefert die folgende Anfrage ein 7-Tage-Fenster aller bestandenen Prüfungen um ein konkretes Datum herum.

Anfrage 2.3 Anzahl bestandener Prüfungen in einem Zeitfenster von 7 Tagen

```
SELECT COUNT Matrikelnr.
ORDER BY Datum ROWS BETWEEN 3 preceding AND 3 following AS Semester_7Tage
FROM TEILNEHMER
WHERE Note <= 4.0
```

Die für diese Arbeiten relevanten OLAP-Operationen sind somit definiert. Es folgt eine kurze Einführung in die Thematik der *Big Data Analytics*.

2.3. Big Data Analytics

Soziale Netzwerke wie Facebook und Twitter, Forschungseinrichtungen wie das „European Council for Nuclear Research", kurz CERN, sowie Log-Analysen im Web oder von Sensoren produzieren eine Unmenge an zu verarbeitenden Daten. Dieses Datenvolumen, ihre große Vielfalt, aber auch die Geschwindigkeit der Datenverarbeitung werden hierbei unter dem Begriff **Big Data** zusammengefasst. Die Bezeichnung Big Data wird i. A. für Datenvolumina verwendet, die so umfangreich sind, dass sie nur schwer mit klassischen Mitteln verarbeitet werden können [OVH17].

In seinem *Oracle White Paper* beschreibt J.-P. Dijcks vier charakteristische Eigenschaften, die Big Data definieren [Dij13]. Diese können wie folgt beschrieben werden:

- Data at Rest (**Volume**): Maschinell erzeugte Daten können in kürzester Zeit riesige Ausmaße annehmen. Der ATLAS-Detektor des Large Hadron Colliders am CERN generiert beispielsweise 320 Megabyte pro Sekunde [CER17]. Eine grundlegende Aufgabe der Big-Data-Analyse besteht also in der Bewältigung der schieren Masse der gespeicherten Daten. Für das CERN sind dies etwa 1% der gemessenen Daten.

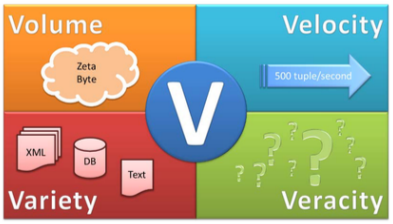
- Data in Motion (**Velocity**): Neben der großen Datenmengen spielt die Geschwindigkeit, mit der diese produziert werden eine entscheidende Rolle. Auch hierfür kann das obige Beispiel vom CERN herangezogen werden.

- Data in Many Forms (**Variety**): Traditionelle Datenformate sind in der Regel relativ gut durch feste Datenschemata definiert und ändern sich nur langsam. Im Gegensatz dazu zeigen nichttraditionelle Datenformate eine hohe Änderungsrate: Neue Sensoren werden eingesetzt, neue Dienste hinzugefügt, ... Die Folge sind eine Vielzahl unstrukturierter, semi-strukturierter und strukturierter Datentypen wie Text, Bild, Audio oder Video.

- **Value** of the Data: Der ökonomische Wert der verschiedenen Daten variiert sehr stark. Die Herausforderung liegt in der Identifizierung und Extraktion der wichtigsten bzw. wertvollsten Daten.

A. Heuer hingegen definiert Big Data über die in Abbildung 2.2a) dargestellten Eigenschaften: Volume, Velocity, Variety sowie Veracity [Heu17]. Wie schwierig es ist, Big Data zu definieren, beschreibt Salzig in seinem Blogbeitrag *Was ist Big Data? - Eine Definition mit fünf V*. Er fasst die ersten drei Eigenschaften als „grundlegende V" auf, die durch die beiden „entscheidenden zusätzlichen V", Value und Veracity, beliebig ergänzt werden können [Sal16]. Da der unternehmerischen Erfolg im Zuge der Provenance-Analyse eine eher vernachlässigbare Rolle spielt, fällt die Wahl des „vierten V" in dieser Arbeit auf die Datenunzuverlässigkeit, die *Veracity*.

- Data in Doubt (**Veracity**): Die letzte Eigenschaft beschreibt die Unzuverlässigkeit oder Unsicherheit der zu Grunde liegenden Daten. Die Herkunft der Daten ist daher von essenzieller Bedeutung für die Big-Data-Analyse.



(a) Die vier V: Volume, Velocity, Variety und Veracity

(b) Die vier P: Privacy, Preservation, Parallelization und Provenance

Abbildung 2.2. Die Eigenschaften von Big Data und ihre Schwerpunkte [GH16a]

Neben den obigen „fünf V“ können bei der Analyse von Big Data in Assistenzsystemen [Heu15], WossiDiA [BMSS15], Pagebeat [FBH+14] und vielen anderen Forschungsanwendungen vier Schwerpunkte — die „vier P“ — gesetzt werden (siehe Abbildung 2.2b):

- Privacy: Privatheit / Datenschutz
- Preservation: Langzeitarchivierung der Daten
- Parallelization: Effiziente Analyse auf großen Datenmengen
- Provenance: Herkunft der Daten

So besteht das Ziel der Privacy-Analyse in der Gewährleistung der Privatheit der Nutzer. Das Projekt PArADISE der Universität Rostock [HM15] beschäftigt etwa sich mit der Entwicklung effizienter Techniken zur Auswertung großer Mengen von Sensordaten, die zuvor definierte Privatheitsansprüche per Systemkonstruktion erfüllen. In diesem Zusammenhang wird auch die Parallelisierung von Big Data Analytics untersucht. Ziel ist eine automatische Transformation von Machine-Learning-Algorithmen, die beispielsweise in R oder Matlab geschrieben wurden, in SQL, um die SQL-Anfrage anschließend zu optimieren und hochgradig zu parallelisieren [MH15]. Datenbankanfragen können so weiter optimiert und eine größere Daten- und Transaktionssicherheit erreicht werden.

Neben den Gebieten der Privacy und Parallelization soll auch die Preservation umrissen, hier aber ebenfalls nicht weiter vertieft werden. Die Langzeitarchivierung von Daten ist ein langwieriger und zeitaufwendiger Prozess. Analoge Medien wie Bücher, Pergament oder Höhlenmalereien — um nur einige zu nennen — können eine „Haltbarkeit“ von bis zu 500 Jahren aufweisen. Eine ähnlich lange digitale Archivierung kann durch Bilder oder die Verfilmung der Daten erreicht werden. An der Universität Rostock werden diese hochvernetzten Dokumentmengen mit Hilfe von typisierten, gerichteten Hypergraphen im Rahmen des Langzeitprojektes HyDRA (a HYpergraph of Documents in a Relational Archive) modelliert [Leh17]. Forschungsgegenstand sind hier unter anderem die Anfrageverarbeitung und -optimierung sowie die Dokumentmodellierung und -speicherung.

Das vierte P — Data Provenance — wird im folgenden Abschnitt 2.4 kurz motiviert und in Kapitel 3 in den ersten beiden Abschnitten ausführlich untersucht. Schwerpunkt sind dort unter anderem die verschiedenen Provenance-Anfragen und -Antworten sowie die Aufstellung der so genannten Provenance-Polynome.

Der Begriff Big Data Analytics steht somit für die Untersuchung bzw. Auswertung von großen Datenmengen unterschiedlicher Arten (Big Data), um darin versteckte Muster, unbekannte Korrelationen und andere nützliche Informationen zu entdecken. Die Relevanz dieser Thematik nimmt bereits seit einigen Jahren immer weiter zu. So wurde das Jahr 2013 etwa zum Jahr der Big Data Analytics gekürt [Gal17]. Big Data ändert nicht nur die Werkzeuge zur Datenverarbeitung, sondern auch unsere Dankweise über

die Wissensgewinnung und Interpretation. Die übliche Test- und Fehleranalyse ist auf großen Datenmenge nicht mehr ohne weiteres realisierbar, sodass neue Strategien entwickelt werden müssen. Eine dieser Techniken ist das Machine-Learning, dessen Vertreter — Neuronale Netze, k-NN (k-Nearest-Neighbors), Bayessches Netze, das Hidden-Markov-Modell, Support-Vektor-Maschinen, ... — ebenfalls an Bedeutung gewinnen. Exemplarisch für diese Technik soll im weiteren Verlauf der Arbeit das Hidden-Markov-Modell detaillierter untersucht werden (siehe Abschnitt 4.3).

2.4. Data Provenance

Das Wort **provenance** stammt aus dem englischen Sprachgebrauch und bedeutet wörtlich übersetzt *Herkunft* oder *Ursprung* [Hem17]. Im Sinne der Informatik beschäftigt sich das Provenance Management mit der Rückverfolgbarkeit eines Ergebnisses bis zu den relevanten Originaldaten. Anwendung findet dieses Management etwa in der Interpretation von Statistiken, der Analyse von Wahlergebnissen oder Umfragen sowie experimentellen Auswertungen [Aug17].

In der Provenance-Analyse werden jeweils vier Anfragen (***where***, ***why***, ***how*** und ***why not***) und vier Antworten (extensional, intensional, anfragebasiert und modifikationsbasiert) unterschieden. Diese werden im dritten Kapitel noch einzeln eingeführt und diskutiert. Der Fokus wird dabei insbesondere auf den ***why***-, ***where***- und ***how***-Anfragen sowie der extensionalen Antwort liegen. Eine ausführliche Untersuchung der intensionalen Provenance-Antwort hat beispielsweise J. Svacina in seiner Bachelorarbeit *Intensional Answers for Provenance Queries in Big Data Analytics* vorgenommen [Sva16].

2.5. Schemaabbildungen

Unter dem Begriff der **Schemaabbildung** (engl. schema mapping) verbirgt sich die Zuordnung eines gegebenen Quellschemas S zu einem ebenfalls gegebenen Zielschema S'. Dabei wird definiert, welche Daten im Quellschema existieren und wie sich diese — ohne Verletzung der Integritätsbedingungen — mit den Daten des Zielschemas verknüpfen lassen. Schematisch kann dieses Vorgehen wie folgt dargestellt werden:

$$\begin{array}{lcl} \text{Quellschema } S & \rightarrow & \text{Zielschema } S' \\ \text{mit Integritätsbedingungen } \mathcal{B} & \rightarrow & \text{mit Integritätsbedingungen } \mathcal{B}' \\ \\ \text{Quellinstanz } d & \rightarrow & \text{Zielinstanz } d' \end{array}$$

Definition 2.16 (Schemaabbildung nach Fagin)**.** Eine **Schemaabbildung** $\mathcal{M}$ ist ein Tripel (S, S', Σ), bestehend aus einem Quellschema S, einem Zielschema S' sowie einer Menge von Abhängigkeiten Σ, welche die Beziehungen zwischen S und S' beschreibt. Die Abbildung $\mathcal{M}$ ist dabei durch die binäre Relation

$$\text{Inst}(M) = \{(I, J) \mid I = \text{ Instanz des Quellschemas, } J = \text{ Instanz des Zielschemas, } (I, J) \models \Sigma\}$$

eindeutig bestimmt. ⊡

Mögliche Schwierigkeiten bzw. Fehlerquellen beim Aufstellen solcher Schemaabbildungen beschreiben A. Doan, A. Halevy und Z. Ives in ihrem Lehrbuch *Principles of Data Integration* [DHI12]. Sie zählen vier zu beachtende Diskrepanzen:

- **Relationen- und Attributnamen**: Die Relationen- und Attributnamen des Quellschemas unterscheiden sich in der Regel von denen des Zielschemas, auch wenn sie sich auf das selbe Konzept beziehen. So kann sich das Attribut *Name* beispielsweise auf den Namen einer Person, aber auch auf eine Firma oder ein Produkt beziehen.

- **Tabellenorganisation**: Die Tabellen zweier betrachteter Schemata müssen nicht übereinstimmen. Die Schemata können verschiedenen Konventionen zur Angabe der Datenwerte unterliegen. So sind verschiedene Skalen oder Schreibweisen der Attributwerte möglich. Die Angabe einer Prüfungsnote ist so beispielsweise als Schulnote 2^- oder als Universitätsnote 2.3 möglich. Auch können sie Daten auf verschiedene Tabellen verteilt sein.

- **Granularitätsniveau**: Die Deckung und Höhe der Granuarität der beiden Schemata können sich unterscheiden. Bei einer Schemaabildung vom Quellschema S (siehe Tabelle 2.5) auf das Zielschema S' (siehe Tabelle 2.6) liegen nicht alle benötigten Informationen vor. Das Attribut *Fakultät* aus dem Schema S kann nicht gefüllt werden. Eine Schemaabbildung mit Quellschema S' und Zielschema S hätte hingegen einen Datenverlust des *Falktät*-Attributs zur Folge.

Matrikelnr	Name	Studiengang
7	Mustermann	Elektrotechnik
2	Sonnenschein	Mathematik

Tabelle 2.5. Schema S der STUDENTEN-Tabelle B.1

Matrikelnr	Name	Studiengang	$\text{Studiengang}_{\text{ID}}$	Studiengang	Fakultät
7	Mustermann	I	I	Informatik	IEF
2	Sonnenschein	II	II	Mathematik	MNF

Tabelle 2.6. Schema S' der STUDENTEN-Tabelle B.1

- **Datenebenen-Variation**: Eine Relation R_i besteht im Quelleschema beispielsweise aus zwei, im Zielschema jedoch aus drei Attributen. So kann die Angabe des Vor- und Nachnamens in Kombination mit einem Titel oder einem „middle initial“ bereits durch verschiedene Variationen von zwei bis vier Namensattributen realisiert werden:

Titel	Vorname	middle initial	Nachname
Dr.	Max	von	Mustermann
-	Dr. Max	von	Mustermann
Dr.	Max	-	von Mustermann
-	Dr. Max	-	von Mustermann

Tabelle 2.7. Verteilung von Titel, Vor- und Nachname sowie middle initial auf verschiedene Namensattribute

2.6. Tableaus und der CHASE

Für den Äquivalenzbeweis zweier Schemata genügt die Angabe einer bijektiven Abbildung. Diese sind in der Regel aber nicht auf den ersten Blick ersichtlich. Ein möglicher Nachweis besteht daher im Beweis der Äquivalenz der zugrundeliegenden Tableaus. Sie sind leicht zu bestimmen und Grundlage einiger weiterer Analysen, wie beispielsweise dem weiter unten vorgestellten CHASE-Algorithmus.

Die Definitionen und Aussagen dieses Abschnitts beruhen ebenfalls auf D. Maiers Lehrbuch *The Theory of Relational Databases* [Mai83]. Einzig Definition 2.17 wird aus der Theorievorlesung *Theorie relationaler Datenbanken* von A. Heuer [Heu16b] übernommen. Heuers Formulierung ist formal aufgeschrieben und daher leichter zu verstehen als das Analogon von Maier.

2.6.1. Das Tableau als Darstellungshilfe

Tableaus finden im Bereich der Datenbanktheorie vielfach Anwendung. So können sie für die Darstellung von Datenbanken, Abhängigkeiten wie FDs oder JDs und deren Implikationen sowie für die Optimierung relationenalgebraischer Ausdrücke (etwa die Projektion, Selektion oder der natürliche Verbund) sowie die Äquivalenzuntersuchung von Datenbankschemata verwendet werden. Sie werden dabei als „Informationsschablonen“ genutzt oder für Untersuchung von Tableauabbildungen zu Rate gezogen. Außerdem dienen sie als Grundlage für benutzerfreundliche Abfragesprachen in relationalen Datenbanksystemen.

Definition 2.17 (Tableau)**.** Für zwei natürliche Zahlen $n, m \in \mathbb{N}^+$ wird ein **Tableau** T über einer Menge von Attributen $\mathcal{R} := \{A_1, ..., A_n\}$ definiert durch:

1. T ist eine endliche Menge von Abbildungen von $\mathcal{R}$ nach V.
2. Es gibt eine Menge von Variablen $V = V_a \cup V_b$ mit V_a und V_b disjunkt sowie
   - **Menge der ausgezeichnete Variablen**: $V_a = \{a_1, a_2, ..., a_n\}$
   - **Menge der nichtausgezeichneten Variablen**: $V_b = \{b_1, b_2, ..., b_m\}$
3. Für alle $1 \leq i, j \leq n$ mit $i \neq j$ gilt:
$$\forall v \in \pi_{A_i}(T) : \forall v' \in \pi_{A_j}(T) : v \neq v'.$$
4. Für die Variablenmenge $V_{a,A_i} := \pi_{A_i}(T) \cap T_a$ gilt:
$$\mid V_{a,A_i} \mid \leq 1.$$
   Das $a \in \pi_{A_i}(T)$ wird dann mit a_i bezeichnet.

Die Tupel eines Tableaus heißen **Zeilen**. ⊡

Bemerkung.

- Wird ein Tableau T als Anfrage interpretiert, entsprechen die a_i gerade den in der Zielinstanz vorkommenden Variablen (siehe Beispiel 2.20).
- Für die ausgezeichneten und nichtausgezeichneten Variablen gilt: $V_a \cap V_b = \emptyset$.

Beispiel 2.6. *Für die Attributmenge $\mathcal{R} := \{A_1, A_2, A_3, A_4, A_5, A_6, A_7\}$ könnte ein Tableau wie folgt aussehen:*

T :	A_1	A_2	A_3	A_4	A_5	A_6	A_7
	a_1	b_1	b_{10}	b_2	b_4	b_5	b_6
	a_1	b_7	b_8	b_9	a_5	b_3	a_7

Tabelle 2.8. Beispieltableau nach Definition 2.17

Dieses besteht aus zwei Zeilen mit jeweils zwei bis drei ausgezeichneten Variablen a_i. Sie stehen in den Spalten der jeweiligen Attribute, d.h. die Variable a_2 steht einzig in der Spalte A_2. Die übrigen Variablen sind nicht ausgezeichnete Variablen. Sie sind durchnummeriert und tauchen in der Regel nur einmal im Tableau auf.

Definition 2.18 ($\pi \bowtie$-Abbildung)**.** Sei $S = \{\mathcal{R}_1, ..., \mathcal{R}_n\}$ eine Menge von Relationenschemata, sodass $\mathcal{R} = \mathcal{R}_1 \cup \cdots \cup \mathcal{R}_n$ gilt. Die durch S definierte **$\pi \bowtie$-Abbildung** $\mathcal{M}_S : \{r \mid r(\mathcal{R})\} \to \{r \mid r(\mathcal{R})\}$ ist eine Abbildung mit

$$\mathcal{M}_S := \pi_{\mathcal{R}_1}(r) \bowtie \cdots \bowtie \pi_{\mathcal{R}_n}(r).$$

□·

Gilt $\mathcal{M}_S(r) = r$, dann genügt $r(\mathcal{R})$ der Verbundabhängigkeit aus Definition 2.7, d.h. $\bowtie [\mathcal{R}_1, \cdots, \mathcal{R}_n]$. Die Relation $r(\mathcal{R})$ aus Beispiel 2.6 ist beispielsweise für $S = \{A_1A_2A_4A_5, A_2A_3, A_1A_4A_6A_7\}$ $\pi \bowtie$-Abbildung.

Für die Darstellung von Datenbanken sowie die Äquivalenzuntersuchung von Datenbankschemata reicht eine Erweiterung der Definition 2.17 um die Positionsbestimmung der ausgezeichneten Variablen. Die Idee besteht darin, dass jede Zeile des Tableaus T einem Relationenschmema $\mathcal{R}_i$ eines Datenbankschemas S entspricht. Dabei ist eine Variable einer Zeile ausgezeichnet ($a_i \in V_a$), wenn das zugehörige Attribut im Relationenschema enthalten ist, und nichtausgezeichnet ($b_j \in V_b$), falls dies nicht der Fall ist. Formal bedeutet dies:

Definition 2.19 (Tableau eines Datenbankschemas)**.** Sei $S = \{\mathcal{R}_1, ..., \mathcal{R}_p\}$ ein gegebenes Datenbankschema mit $\mathcal{R} = \mathcal{R}_1 \cup ... \cup \mathcal{R}_p = \{A_1, ..., A_m\}$. Das **Tableau** für S ist dann definiert durch:

- $T_S := \{w_1, ..., w_p\}$ mit $\forall i \in \{1, ..., p\} : \forall j \in \{1, ..., m\} : w_i(A_j) = a_j \Leftrightarrow A_j \in \mathcal{R}_i$.
- Alle nicht durch festgesetzte Variablen a_j mit $j \in \{1, ..., p\}$ belegten Positionen werden mit b_k und fortlaufendem k = 1, 2, ... besetzt,
- T_S ist selber Tableau über $\mathcal{R}$.

□·

Beispiel 2.7. *Für $S = \{A_1, A_1A_5A_7\}$ und $\mathcal{R} = \{A_1, A_2, A_3, A_4, A_5, A_6, A_7\}$ folgt das Tableau*

T :	A_1	A_2	A_3	A_4	A_5	A_6	A_7	
	a_1	b_1	b_{10}	b_2	b_4	b_5	b_6	w_1
	a_1	b_7	b_8	b_9	a_5	b_3	a_7	w_2

Tabelle 2.9. Tableau eines Datenbankschemas S

Jede Tableauzeile entspricht einem der Relationenschemata A_1 und $A_1A_5A_7$. Zudem können nichtausgeneichnete Variablen durch Konstanten ersetzte werden, etwa b_{10} = 'Max'.

Tableaus dienen auch als Grundlage für Anfragen. In diesem Fall wird eine ausgezeichnete Tableauzeile, die *summery* eingeführt. Sie beinhaltet das Anfrageergebnis. Das zugehörigen $\pi \bowtie \sigma$*-Tableau* ist somit eine Erweiterung des Tableaus aus Definition 2.17 um die Eigenschaften 5. bis 9. der folgenden Definition 2.20 :

Definition 2.20 ($(\pi \bowtie \sigma)$-Tableau)**.** Gegeben seien zwei natürliche Zahlen $n, m \in \mathbb{N}^+$ sowie eine Menge von Attributen $\mathcal{R} := \{A_1, ..., A_n\}$. Seien weiter V eine Menge von Variablen und $\mathbb{D}$ mit $\mathbb{D} := \bigcup_{i=1}^{p} \mathbb{D}(A_i)$ eine Menge von **Symbolen**. Dann heißt T **$(\pi \bowtie \sigma)$-Tableau** über R genau dann, wenn:

1. T ist eine endliche Menge von Abbildungen $\mathcal{R} \to V \cup \mathbb{D} \cup \{␣\}$
2. Es gilt: $V = V_a \cup V_b$ mit V_a und V_b disjunkt sowie
   - **Menge der ausgezeichnete Variablen**: $V_a = \{a_1, a_2, ..., a_n\}$
   - **Menge der nichtausgezeichneten Variablen**: $V_b = \{b_1, b_2, ..., b_m\}$
3. Für alle $1 \leq i, j \leq n$ mit $i \neq j$ gilt:
$$\forall v \in \pi_{A_i}(T) : \forall v' \in \pi_{A_j}(T) : v \neq v'.$$
4. Für die Variablenmenge $V_{a,A_i} := \pi_{A_i}(T) \cap T_a$ gilt:
$$\mid V_{a,A_i} \mid \leq 1.$$
5. Ein Tupel von T — oberhalb des Querstrichs — heißt **Ergebnis** (engl. summary) w_0.
6. Es existiert kein $b_i \in V_b$ im Ergebnis w_0.
7. Für die übrigen Tupel in T, die **Zeilen**, gilt: $a_i \in \pi_{A_i}(\{w_1, \cdots, w_n\}) \Rightarrow a_i = w_0(A_i)$.
8. In $\{w_1, ..., w_n\}$ kommt ␣ nicht vor.
9. $c \in \pi_{A_i}(T) \cap \mathbb{D} \Rightarrow c \in \mathbb{D}(A_i)$

□

Beispiel 2.8. *Für $\mathcal{R} = \{A_1, A_2, A_3, A_4, A_5\}$ und $\mathbb{D} = \{1, 2, 3, ...\}$ ist etwa*

T' :	A_1	A_2	A_3	A_4	A_5	A_6	A_7	
	a_1				a_5		a_7	w_0
	a_1	b_1	'Max'	b_2	b_4	b_5	b_6	w_1
	a_1	b_7	b_8	b_9	a_5	b_3	a_7	w_2

Tabelle 2.10. Beispiel einer Tableauanfrage

eine Tableauanfrage für Q aus Beispiel 2.9.

Die *Universal Instance Assumption* in relationalen Datenbanken besagt, dass alle Schemata (Attribute) und Instanzen (Relationen) in einer (möglicherweise sehr breiten) Tabelle stehen, die dann bei Bedarf in kleinere Tabellen zerlegt werden kann. Um diesen Sachverhalt aufzulösen, bedarf es zweier Schritte: Zunächst werden die einzelnen Tableauzeilen mit den jeweiligen Relationenschemata markiert (engl. **tag**). Anschließend werden die nicht zum Schema gehörigen nichtausgezeichneten Variablen V_b gestrichen und durch Blanks ersetzt. Für das Tableau aus Beispiel 2.10 ergibt sich so folgendes neue Tableau T'':

Beispiel 2.9. *Gegeben sei die Anfrage*

$$Q = \pi_{A_1}(\sigma_{A_3='Max'}(r_1)) \bowtie \pi_{A_1A_5A_7}(r_2))$$

mit $r_1 = \pi_{A_1A_2A_3A_4}(r)$ und $r_2 = \pi_{A_1A_5A_6A_7}(r)$. Dann kann hierzu das folgende Tableau T'' aufgestellt werden:

T'':

A_1	A_2	A_3	A_4	A_5	A_6	A_7	
a_1				a_5		a_7	
a_1	b_1	'Max'	b_2				r_1
a_1				a_5	b_3	a_7	r_2

Tabelle 2.11. Beispiel für die Erweiterung eines Tableaus um Tags

2.6.2. Das Universalwerkzeug CHASE

Der CHASE-Algorithmus ist ein in der Datenbanktheorie häufig verwendetes Werkzeug zur Untersuchung von Anfrage- oder Datenbankäquivalenzen unter gegebenen Abhängigkeiten (engl. query containment), dem Finden von inversen Schemaabbildungen, der Anfrageoptimierung (unter Abhängigkeiten) sowie der Darstellung von Abhängigkeiten in Datenbanken. Im Abschnitt 3.3 soll hierauf noch genauer eingegangen werden. Zu diesem Zeitpunkt beschränken wir uns auf die Analyse von Äquivalenzen.

Ziel ist also die Konstruktion eines äquivalenten Tableaus T' zu einem Tableau T, welches den gegebenen FDs und JDs genügt. Die Integration dieser Abhängigkeiten gelingt durch die Anwendung der F- sowie der J-Regel. Erstere verarbeitet die funktionalen Abhängigkeiten (siehe Definition 2.21); letztere die Verbundabhängigkeiten (siehe Definition 2.22).

Definition 2.21 (F-Regel)**.** Die F-Regel einer funktionalen Abhängigkeit $A_i \rightarrow A_j$ ist eine Vorschrift, die ein Tableau T wie folgt auf ein zweites Tableau T' abbildet: Seien w_1 und w_2 zwei Zeilen von T, so dass $w_1(A_i) = w_2(A_i)$ gilt. Seien weiterhin $v_1 := w_1(A_j)$ und $v_2 := w_2(A_j)$ mit $v_1 \neq v_2$ gegeben. Dann werden die Variablen in v_1 bzw. v_2 wie folgt ersetzt:

- $v_1 \in V_a \Rightarrow v_2 := v_1$
- $v_2 \in V_a \Rightarrow v_1 := v_2$
- $v_1 \notin V_a \wedge v_2 \notin V_a : v_1 = b_\nu \wedge v_2 = b_\mu \Rightarrow \begin{cases} b_\nu := b_\mu, \ falls\ \mu < \nu \\ b_\mu := b_\nu, \ falls\ \nu < \mu \end{cases}$

□

Definition 2.22 (J-Regel)**.** Sei $S = \{\mathcal{R}_1, \mathcal{R}_2, ..., \mathcal{R}_n\}$ eine Menge von Relationenschemata und JD $\bowtie [\mathcal{R}_1, ..., \mathcal{R}_n]$ eine gegebene Verbundabhängigkeit. Weiter seien $w_1, ..., w_n$ die Zeilen eines Tableaus T, die zu einem Ergebnis w joinbar sind. Dann gilt für das neue Tableau T':

$$T' = T \cup \{w\}.$$

Die Hintereinanderausführung der obigen beiden Regeln wird auch als CHASE-Algorithmus oder kurz CHASE bezeichnet. Er erfüllt dabei die folgenden typischen Eigenschaften von Regelsystemen:

- Bei Eingabe eines Tableaus ist das Ergebnis des CHASE ist wiederum Tableau. Im späteren Verlauf der Arbeit wird der CHASE auch auf andere „Objekte“ ◯ angewandt. In diesem Fall liefert der CHASE-Algorithmus wiederum das „Objekt“ ◯.
- Konfluenz: Jede mögliche Reihenfolge der Anwendung der obigen F- und J-Regel liefert dasselbe Ergebnistableau.
- Terminierung: Jede zulässige Eingabe liefert nach endlich vielen Schritten ein Ergebnis.

Definition 2.23 (CHASE-Algorithmus)**.** Für ein gegebenes Tableau T heißt T_n **CHASE** von T unter der Einschränkung $\mathcal{B}$, kurz $T_n = \text{chase}_{\mathcal{B}}(T)$. Dabei ist T_n Teil einer Folge von Tableaus $T_0, T_1, ...$, wobei sich T_{i+1} mittels F-Regel oder J-Regel aus T_i herleiten und $T = T_0$ gilt. $\boxdot$

Der Index n bezeichnet somit das Tableau, für welches der CHASE-Algorithmus keine Änderung des Tableaus durch erneute Anwendung der F- und J-Regel mehr nach sich zieht, d.h.

$$\text{chase}_{\mathcal{B}}(T_{n+1}) = \text{chase}_{\mathcal{B}}(T_n).$$

Beispiel 2.10. *Sei T wie in Beispiel 2.6 gegeben, d.h.*

T_0 :	A_1	A_2	A_3	A_4	A_5	A_6	A_7
	a_1	b_1	b_{10}	b_2	b_4	b_5	b_6
	a_1	b_7	b_8	b_9	a_5	b_3	a_7

Tabelle 2.12. Ausgangstableau für CHASE-Algorithmus

und $\mathcal{B} = \{A_2A_4 \to A_3, \bowtie [A_1A_2A_4A_5, A_1A_3, A_1A_4A_6A_7]\}$. Die Anwendung der J-Regel liefert das Tableau T_1, die Anwendung der F-Regel das Tabelau T_2 der Tableaufolge T_0, T_1, T_2. Die Änderungen sind in Tabelle 2.13 jeweils grün markiert. Eine erneute Anwendung der beiden Regeln erzeugt kein weitere Tableau T_3, sodass der CHASE-Algorithmus nach T_2 abbricht.

T_1 :	A_1	A_2	A_3	A_4	A_5	A_6	A_7
	a_1	b_1	b_{10}	b_2	b_4	b_5	b_6
	a_1	b_1	b_8	b_2	b_4	b_5	b_6
	a_1	b_7	b_{10}	b_9	a_5	b_3	a_7
	a_1	b_7	b_8	b_9	a_5	b_3	a_7

T_2 :	A_1	A_2	A_3	A_4	A_5	A_6	A_7
	a_1	b_1	b_8	b_2	b_4	b_5	b_6
	a_1	b_1	b_8	b_2	b_4	b_5	b_6
	a_1	b_7	b_8	b_9	a_5	b_3	a_7
	a_1	b_7	b_8	b_9	a_5	b_3	a_7

Tabelle 2.13. CHASE-Algorithmus: links: Anwendung der J-Regel; rechts: Anwendung der F-Regel

Nach dem Streichen der Duplikate (zweite und vierte Zeile) ergibt sich so das Ergebnistableau T_n (siehe Tabelle 2.14). Es unterscheidet sich vom Ausgangstableu T lediglich im Attribut A_3.

T_n :	A_1	A_2	A_3	A_4	A_5	A_6	A_7
	a_1	b_1	b_8	b_2	b_4	b_5	b_6
	a_1	b_7	b_8	b_9	a_5	b_3	a_7

Tabelle 2.14. Ergebnistableau nach CHASE-Algorithmus

Mit dem Abschluss dieses Kapitels sind alle für das Verständnis der vorliegenden Arbeit notwendigen Grundlagen geschaffen: Das Relationenmodell mit den zugehörigen relationalen Operationen aus Abschnitt 2.1 und 2.2 sowie die Definition der Schemaabbildung (siehe Definition 2.16) bilden die Basis aller weiteren Untersuchungen. Die Data Provenance und der CHASE-Algorithmus werden im dritten Kapitel zu verschiedenen Anfrage-Typen, Polynomen (siehe Abschnitt 3.1 und 3.2) sowie Algorithmusvarianten (siehe Abschnitt 3.3) verfeinert und für die Analyse des Hidden-Markov-Modells (siehe Abschnitt 4.3) kombiniert. Eine erste praktische Umsetzung der im dritten und vierten Kapitel entwickelten Theorien erfolgt dann im Kapitel 5 dieser Arbeit.

3. Aktueller Stand der Forschung

Die Big Data Analytics wird, wie zuvor erläutert, am Lehrstuhl *Datenbanken und Informationssysteme* der Universität Rostock in vier Schwerpunkte unterteilt werden: Privacy, Preservation, Parallelization und Provenance. Die vorliegende Arbeit beschäftigt sich mit dem vierten Schwerpunkt, der Provenance.

Abbildung 3.1. Provenance: Ein Schwerpunkt von Big Data Analytics [GH16a]

Für die *Umsetzung von Provenance-Anfragen in Big-Data-Analytics-Umgebungen* muss nach der Darlegung aller notwendigen Grundlagen (siehe Kapitel 2) der Stand der aktuellen Forschung im Bereich der „Provenance in Databases“ analysiert werden. Hierzu werden in diesem Kapitel etwa die Themen Provenance-Anfragen und -Antworten (Abschnitt 3.1), Provenance-Polynome (Abschnitt 3.2), einige CHASE-Varianten (Abschnitt 3.3) — CHASE für s-t tgds und egds und der CHASE&BACKCHASE — sowie die Inversen Schemaabbildungen (Abschnitt 3.4) genauer untersucht. Ziel ist die Vorstellung einiger aktueller Erkenntnisse und Methoden, die anschließend im vierten Kapitel miteinander verwoben werden sollen. Wir motivieren aber bereits in den folgenden Abschnitten, in welcher Weise die Techniken für unsere Zwecke kombiniert werden sollen.

3.1. Provenance-Anfragen und -Antworten:

Typischerweise werden sowohl vier Provenance-Anfragen (siehe Tabelle 3.1) als auch vier Provenance-Antworten (Tabelle 3.3) unterschieden. Die ersten drei Anfragen können dabei bzgl. ihres Informationsgehaltes ($\preceq$) wie folgt geordnet werden: ***where*** $\preceq$ ***why*** $\preceq$ ***how***. Mit anderen Worten, aus dem Ergebnis der ***how***-Provenance können die ***why***- sowie die ***where***-Provenance abgeleitet werden. Gleiches gilt für die ***why***- und die ***where***-Provenance. Die ***why not***-Anfrage kann in diese „Informationskette“ allerdings nicht eingeordnet werden, da sie, anders als die ***where***-, ***why***- oder ***how***-Anfragen, modifikationsbasierte Antworten liefern.

Anfrage-Typ	Fragestellung
where	Woher kommen die Daten?
why	Warum dieses Ergebnis?
how	Wie kommt das Ergebnis zustande?
why not	Warum fehlt ein bestimmtes Element im Ergebnis?

Tabelle 3.1. Provenance-Anfragen

Für die Reduktion der ***how***-Provenance auf die ***why***-Provenance werden alle Identifikatoren des *Provenance-Polynoms* in eine duplikatfreie Liste geschrieben. Wird diese Liste auf die Tupelidentifikatoren der Ergebnisrelation eingeschränkt, folgt sich die ***where***-Provenance. Für die Beispielanfrage

$$Q = \pi_{\text{Modulnr}}(\sigma_{\text{Vertiefung='Modelle und Algorithmen'}}(\text{MODULE}) \bowtie \sigma_{\text{Dozent='Professor E'}}(\text{DOZENTEN}))$$

ergeben sich etwa die in der Tabelle 3.2 zusammengefassen Provenance-Ergebnisse: $D_8 \cdot M_8$, (D_8, M_8), D_8 und Dozenten. Die „Informationskette“ ist rot hervorgehoben.

where-Provenacne (Tabellenname)	***where***-Provenance (Tupelname)	***why***-Provenance	***how***-Provenance
Dozenten	D_8	(D_8, M_8)	$D_8 \cdot M_8$

Relationenname Ergebnistupel Tupelliste

Tabelle 3.2. Provenance-Resultate der Anfragen Q

Während die ***where***-Anfrage, im Falle einer extensionalen Antwort, lediglich die Namen der benötigten Tabellen oder Tupel wiedergibt, liefert die ***how***-Anfrage eine konkrete Berechnungsvorschrift in Form eines Provenance-Polynoms [ADT11]. Wir beschränken uns im Falle der ***where***-Provenance auf die Tabellenname, da die ***why***-Anfrage ebenfalls eine Liste von Tupelidentifikatoren zur Folge hat. Der Unterschied besteht allerdings im Umfang dieser Liste. Während die ***why***-Anfrage alle zur Berechnung des Ergebnisses notwendigen Tupelidentikatoren zurückgibt, liefert die ***where***-Anfrage lediglich die Identikatoren der Ergebnistupel.

Neben der extensionalen Antwort existieren noch drei weitere Antwort-Typen, welche eine Beschreibung der Daten (intensionale Antwort), die Selektionsprädikate (anfragebasierte Antwort) oder aber einen Vorschlag zur minimalen Änderung der Auswertung (modifikationsbasierte Antwort) liefern. Sie sollen in dieser Arbeit, ebenso wie die ***why*** not-Anfragen, aber nicht weiter vertieft werden. Grund hierfür ist unter anderem, dass die ***why not***-Anfrage nicht in die oben genannte Ordnung (***where*** $\preceq$ ***why*** $\preceq$ ***how***) eingegliedert werden kann. Die Untersuchung der Provenance-Polynome würde in diesem Fall voraussichtlich keine Erkenntnisse bringen.

Antwort-Typ	Ergebnis	Antwort auf
extensional	Tupel aus den Originaldaten	***where***- oder ***why***-Anfrage
intensional	Beschreibung der Daten	***where***-, ***why***- oder ***how***-Anfrage
anfragebasiert	Selektionsprädikate	***why***- und ***how***-Anfragen
modifikationsbasiert	Vorschlag zur minimalen Änderung der Auswertung	***why not***-Anfrage

Tabelle 3.3. Provenance-Antworten [Heu16a]

Im Folgenden sollen die für diese Arbeit relevanten Provenance-Anfragen ***where***, ***why*** und ***how*** sowie die extensionale Antwort kurz vorgestellt werden. Hierbei dient der 2009 geschriebene Übersichtsartikel *Provenance in Databases: Why, How, and Where* von K. Cheney, L. Chiticariu und W.-C. Tan [CCT09] als Grundlage.

3.1.1. *Why*-Provenance

Die ersten, die 2000 eine Formalisierung im Kontext der relationalen Datenbanken anstrebten, waren Y. Cui, J. Widom und J. L. Wiener in ihrem Artikel *Tracing the Lineage of View Data in a Warehousing Environment* [CWW00]. Um die Frage nach der Herkunft eines bestimmten Tupels in der Ergebnisrelation zu erklären, müssen die notwendigen Informationen aus der Ursprungstabelle mit in die Ergebnisrelation übernommen werden. Cui et. al assoziierten also jedes Ergebnistupel mit einer Menge von Tupeln der Ursprungstabelle, den sogenannten *Zeugen*.

Definition 3.1 (Zeugenmenge)**.** Gegeben seien eine Datenbank d und eine Anfrage Q. Der **Zeuge** (engl. witness) eines Ergebnistupel $t \in Q(d)$ ist eine Teilrelation $w \subseteq d$ mit $t \in Q(w)$. Hat ein Tupel t mehrere Zeugen (beispielsweise k viele), so wird diese auch als **Zeugenmenge** $W = \{w_1, ..., w_k\}$ mit $k \in \mathbb{N}$ bezeichnet. ⊡

Die Anzahl solcher Zeugen kann aufgrund einer Vielzahl „irrelevanter“ Ursprungsaufzeichnungen allerdings sehr groß werden. Je nach Größe der Ursprungsdatenbank kann diese sogar exponentiell werden; im schlimmsten Fall muss die vollständige Ursprungstabelle übernommen werden. Die Problematik des exponentiellen Wachstums lösen P. Buneman, S. Khanna und W.-C. Tan in ihrem Artikel *Why and Where: A Characterization of Data Provenance* [BKT01] durch die Einführung einer *Zeugenbasis*.

Definition 3.2 (Zeugenbasis)**.** Die **Zeugenbasis** $W_{Q,d}(t)$ entspricht der Menge aller Zeugenmengen, d.h. $W_{Q,d}(t) = \{W_1, ..., W_n\}$ mit Zeugenmengen $W_i = \{w_{i_1}, ..., w_{i_k}\}$ und $1 \leq i \leq n, k \in \mathbb{N}$. ⊡

Die Zeugenmenge W eines Tupels t ist invariant unter allen äquivalenten Anfragen Q, nicht jedoch die Zeugenbasis $W_{Q,d}(t)$. Buneman et al. definieren hierzu die **minimale Zeugenbasis** als eine unter Gleichheitsanfragen (siehe Definition 2.8 und 2.9) invariante Teilmenge der Zeugenbasis $W_{Q,d}(t)$.

Definition 3.3 (minimale Zeugenbasis)**.** Eine Zeugenbasis $W_{Q,d}(t)$ heißt für Q **minimal**, wenn keine Teilmenge von $W_{Q,d}(t)$ selber Zeugenbasis für Q ist. ⊡

Bemerkung. Die ***why*-Provenance** eines Tupels t entspricht somit einer Zeugenbasis von t. Ist die Zeugenbasis nicht minimal, so ist sie auch nicht eindeutig. Vergleiche hierzu das Beispiel 3.1.

Basierend auf den Tabellen B.1 und B.5 untersuchen wir hierzu in Beispiel 3.1 die Anfrage Q_1. Sie entspricht der Auflistung aller Modulnummern und Noten der Studentin mit dem Vornamen *Sarah*. Hierbei sollen insbesondere die Zeugenmengen sowie die Zeugenbasis aufgestellt sowie die Minimalität der Zeugenbasis geprüft werden.

Anfrage 3.1 Noten der Studenten mit Vornamen 'Sarah' (Anfrage Q_1)

```
SELECT Modulnummer, Note
FROM Studenten JOIN Noten
ON (Studenten.Matrikelnummer = Noten.Matrikelnummer)
WHERE Studenten.Vorname = 'Sarah'
```

In der zugrundeliegenden Datenbank existiert dieser Name nur einmal, sodass eine äquivalente Anfrage Q_2 formuliert werden kann. Sie bestimmt die Modulnummern und Noten der Studentin mit der Matrikelnummer 2, also der Studentin *Sarah*.

Anfrage 3.2 Noten des Studenten mit Matrikelnummer 2 (Anfrage Q_2)

```
SELECT Modulnummer, Note
FROM Noten
WHERE Matrikelnummer = '2'
```

Beispiel 3.1. *Jedes Tupel der Ursprungstabellen* STUDENTEN *und* NOTEN *ist mit einer* ID *versehen. Diese steht in der ersten Spalte und besteht aus dem Kürzel des Tabellennamens S bzw. N sowie einer fortlaufenden Nummerierung. Die obige Anfrage Q_1 liefert somit:*

Modulnr	**Note**
001	1.7
002	1.3
004	3.0

Tabelle 3.4. Prüfungsergebnisse der Studentin Sarah nach Anfrage 3.1

Die Zeugenbasis ergibt sich nun aus den drei Zeugenmengen W_1, W_2 und W_3 durch:

$$W = \{\underbrace{\{S_2, N_2\}}_{W_1}, \underbrace{\{S_2, N_6\}}_{W_2}, \underbrace{\{S_2, N_{14}\}}_{W_3}\}.$$

Die Anfrage Q_2 liefert dieselbe Ergebnistabelle wie die Anfrage Q_1, jedoch eine andere Zeugenbasis:

$$W' = \{\underbrace{\{N_2\}}_{W'_1}, \underbrace{\{N_6\}}_{W'_2}, \underbrace{\{N_{14}\}}_{W'_3}\}.$$

Berechnung der minimalen Zeugenbasis:

- *Anfrage Q_1: Für die Ergebnistabelle 3.4 existiert keine minimale Zeugenbasis. Weder die Zeugenbasis $W = \{\{S_2, N_2\}, \{S_2, N_6\}, \{S_2, N_{14}\}\}$ selbst noch die resultierende Zeugenliste $Z_W = \{S_2, N_2, N_6, N_{14}\}$ sind minimal. Sie sind keine minimalen Zeugenbasen, da eine „kleinere" Zeugenliste $Z_{W'}$ angegeben werden kann.*
- *Anfrage Q_2: Die Zeugebasis $W' = \{\{N_2\}, \{N_6\}, \{N_{14}\}\} \subseteq W$ ist mit ihrer zugehörigen Zeugenliste $Z_{W'} = \{N_2, N_6, N_{14}\}$ minimal. Ihre Zeugenmengen bestehen lediglich aus Tupeln der NOTEN-Relation. Die Zeugenmengen W_1, W_2 und W_3 hingegen enthalten zusätzlich Tupel der STUDENTEN-Relation.*

Betrachten wir die einzelnen Ergebnistupel unabhängig von den Anfragen Q_1 und Q_2, so ergibt für die Tupel $(001, 1.7)$, $(002, 1.3)$ und $(004, 3.0)$ die minimale Zeugenbasis $W' = \{N_2, N_6, N_{14}\}$. Da W' Teilmenge von W und selber Zeugenbasis von Q_2 ist, ist W' minimale Zeugenbasis für die Anfrage Q_2. Da Q_1 und Q_2 dieselbe Ergebnistabelle 3.4 liefern, kann die Anfrage Q_1 als nicht-optimierte Anfrage vernachlässigt werde.

3.1.2. *How*-Provenance

Die Frage „Warum ergibt sich ein Ergebnistupel t?" ist mit den Ausführungen im vorherigen Abschnitt geklärt. Stellt sich nun die Frage: „Wie kommt ein Ergebnistupel t zustande?" Hierfür werden die sogenannten Provenance-Polynome bemüht. Nach einer kurzen Motivation werden sie im Abschnitt 3.2 ausführlich definiert.

Das Ergebnis der Anfragen Q_1 bzw. Q_2 ist in der obigen Tabelle 3.4 dargestellt. Im Falle der ***how*-Provenance** liefern die Zeugenmengen $\{S_2, N_2\}$, $\{S_2, N_4\}$ und $\{S_2, N_{14}\}$ der Anfrage Q_1 neben der Zeugenliste $\{S_2, N_2, N_4, N_{14}\}$ auch die Provenance-Polynome $S_2 \cdot N_2$, $S_2 \cdot N_4$ und $S_2 \cdot N_{14}$. Umgangssprachlich bedeutet dies, dass die Ursprungstupel S_2 und N_2 in das Ergebnistupel $(001, 1.7)$ jeweils einmal eingehen. Die Multiplikation symbolisiert hierbei den JOIN in der Anfrage Q_1. Die Tabelle 3.5 kombiniert schließlich die Ergebnistabelle 3.4 und die Provenance-Polynome der Ergebnistupel.

Modulnr	**Note**	
001	1.7	$S_2 \cdot N_2$
002	1.3	$S_2 \cdot N_6$
004	3.0	$S_2 \cdot N_{14}$

Tabelle 3.5. Prüfungsergebnisse der Studentin Sarah mit Provenance-Information

3.1.3. *Where*-Provenance

Wie bereits beschrieben, liefert die ***how***-Provenance eine Berechnungsvorschrift zum Erhalt des Anfrageergebnisses. Die zur Berechnung notwendigen Ursprungstupel werden hierbei in der sogenannten Zeugenbasis — ***why***-Provenance — gespeichert. Ist nur der Herkunftsort der Daten von Interesse, so kann die ***where*-Provenance** zu Rate gezogen werden. Sie beschreibt den Zusammenhang zwischen den Quell- und Ausgabeorten. Für eine Datenbank $d = \{r_1, ..., r_n\}$ sowie eine Anfrage Q liefert die ***where***-Provenance eines Ergebnistupels $t \in Q(d)$ also die Ursprungsorte $r_t = \{r_i \mid r_i \text{ ist Quellrelation von } t \text{ und } i \in \{1, ..., n\}\}$. Formal entspricht dies dem Paar (t, r_t). Der Ursprungsort kann hierbei in Form des Relationennamens oder des zugehörigen Tupelidentifikators angegeben werden.

Für die obigen Anfragen Q_1 und Q_2 ergeben sich zwar dieselben Ergebnistabellen (siehe Unterabschnitt 3.1.1), aber wie im Falle der ***why***- und ***how***-Provenance unterschiedliche ***where***-Provenance-Informationen. Diese sind in der folgenden Tabelle 3.6 zusammengefasst:

Modulnr	**Note**	Q_1:	Q_2:
001	1.7	STUDENTEN, NOTEN	NOTEN
002	1.3	STUDENTEN, NOTEN	NOTEN
004	3.0	STUDENTEN, NOTEN	NOTEN

Tabelle 3.6. Prüfungsergebnisse der Studentin Sarah mit Provenance-Informationen für die Anfragen 3.1 und 3.2

3.1.4. Extensionale Antwort

Für die Analyse der obigen Provenance-Fragestellungen können nach M. Herschel vier verschiedene Antworttypen unterschiedenen werden: extensional, intensional, anfragebasiert und modifikationsbaisert [Her15]. Die ersten beiden Typen sowie die *Antwort* als solches definieren S. Lee et al. in ihrem Artikel *A SQL-Middleware Unifying Why and Why-Not Provenance for First-Order Queries* [LKLG17]. Auf die anderen beiden Antworttypen soll hier aber nicht weiter eingegangen werden.

Definition 3.4 (extensionale / intensionale Antwort)**.** Gegeben seien eine Datenbank $d = \{r_1, ..., r_n\}$ sowie ein Ergebnistupel t_i $(i = 1, ..., m)$ einer Anfrage Q. Dann heißt eine Antwort, d.h. eine Menge von Ergebnistupeln $\mathcal{T} = \{t_1, ..., t_m\}$, **extensional** genau dann, wenn t_i über der Menge der Relationen $r_1, ..., r_n$ definiert ist. Ist t hingegen über einer Menge von Regeln definiert, so spricht man von einer **intensionalen** Antwort. □

In der vorliegenden Arbeit beschränken wir uns auf Provenance-Anfragen mit extensionale Antworten. Der Untersuchung mittels intensionaler Antworten widmet sich hingegen J. Svacina in seiner Bachelorarbeit *Intensional Answers for Provenance Queries in Big Data Analytics* [Sva16].

3.2. Provenance-Polynome

Alle für das Verständnis des folgenden Abschnitts relevanten mathematischen Grundlagen können im Anhang A nachgelesen werden. Die Definitionen dieses Abschnitts basieren auf den Artikeln *Provenance Semirings* [GKT07] und *Provenance for Aggregate Queries* [ADT11]. Diese enthalten die allgemeine Definition des Provenance-Halbrings und der Provenance-Polynome, welche im Verlauf des Abschnitts um Aggregatfunktionen und Gruppierung erweitert werden sollen.

3.2.1. Provenance-Halbringe

Die ***where***-Provenance nutzt die Angabe von Polynomen über einem *Provenance-Halbring*. Diese sind für die Grundoperationen — Projektion π, Selektion σ, Umbenennung β, natürlicher Verbund $\bowtie$ und die Vereinigung $\cup$ — definiert.

Definition 3.5 (kommutativer Halbring)**.** Ein kommutativer Halbring ist eine Struktur $(K, +_K, \cdot_K, 0_K, 1_K)$, wobei $(K, +_K, \cdot_K)$ und $(K, \cdot_K, 1_K)$ kommutative Monoide sind und $\cdot_K$ distributiv über $+_K$ ist, sowie $x \cdot_K 0_K = 0 \cdot_K x = 0_K$ gilt. $\boxdot$

Beispiele für kommutative Halbringe sind (nach Definition) alle kommutativen Ringe, aber auch distributive Gitter und boolesche Algebren. In Bezug auf die Provenance sind insbesondere der boolesche Halbring $(\mathbb{B}, \vee, \wedge, \bot, \top)$ und der Halbring der natürliche Zahlen $(\mathbb{N}, +, \cdot, 0, 1)$ von Interesse.

Definition 3.6 (K-Relation)**.** Sei K ein kommutativer Halbring mit einem ausgezeichneten Element 0. Eine **K-Relation** über einer endlichen Attributmenge X ist eine Funktion ρ: $\mathbb{D}^X \to K$, so dass der zugehörige **Träger** definiert ist als $\text{supp}(\rho) = \{t \mid \rho(t) \neq 0\}$. Dabei entspricht $\mathbb{D}^X$ der Abbildung $t : X \to \mathbb{D}$ für eine endliche Attributmenge X und eine Domäne $\mathbb{D}$. $\boxdot$

Zur Verarbeitung der algebraischen Operationen aus Abschnitt 2.2 muss die Menge K um ein zweites ausgezeichnetes Element 1 sowie zwei binäre Operationen erweitert werden. Dabei gilt $1 \neq 0$. Während die erste binäre Operation + (Addition) der Darstellung der Vereinigung, Projektion sowie der Eliminierung von Duplikaten dient, stellt die Einführung der Multiplikation $\cdot$ den natürlichen Verbund dar. So erweitert, kann auf K eine *positive K-relationale Algebra* definiert werden:

Definition 3.7 (Positive K-relationale Algebra)**.** Gegeben seien eine algebraische Struktur $(K, +, \cdot, 0, 1)$ mit zwei binären Operationen und zwei ausgezeichneten Elementen 0 und 1 sowie zwei Attributmengen X, Y. Dann sind die Operationen der **positiven K-relationalen Algebra** wie folgt definiert:

- **Leere Relation**: Für jede Attributmenge X existiert eine Abbildung $\emptyset : \mathbb{D}^X \to K$, so dass $\emptyset(t) = 0$. Diese heißt auch **leere K-Relation**.
- **Projektion**: Seien $\rho : \mathbb{D}^X \to K$ und $Y \subseteq X$ gegeben, dann ist $\Pi_V \rho : \mathbb{D}^Y \to K$ definiert durch

$$(\Pi_Y \rho)(t) := \sum_{t=t' \text{ auf } Y \text{ und } \rho(t')\neq 0} \rho(t')$$

  mit $+_K$ Summe über alle $t' \in \text{supp}(\rho)$ sowie $t' \mid_Y = t$.

- **Selektion**: Für eine gegebene Relation $\rho : \mathbb{D}^X \to K$ sowie ein Selektionsprädikat $\mathbf{P}$, welches jedes Tupel aus X entweder auf 0 oder auf 1 abbildet, ist die Selektion $\sigma_{\mathbf{P}} : \mathbb{D}^X \to K$ charakterisiert als

$$(\sigma_{\mathbf{P}}\rho)(t) := \rho(t) \cdot_K \mathbf{P}(t).$$

- **Vereinigung**: Die Abbildung $\rho_1 \cup \rho_2 : \mathbb{D}^X \to K$ ist für $\rho_i : \mathbb{D}^X \to K$ mit $i = 1, 2$ festgelegt durch

$$(\rho_1 \cup_K \rho_2)(t) := \rho_1(t) +_K \rho_2(t).$$

- **Natürlicher Verbund**: Für $\rho_i : \mathbb{D}^{X_i} \to K$, $i = 1, 2$ ist die K-Relation $\rho_1 \bowtie \rho_2 : \mathbb{D}^{X_1 \cup X_2} \to K$ definiert durch

$$(\rho_1 \bowtie \rho_2)(t) := \rho_1(t_1) \cdot_K \rho_2(t_2).$$

  Dabei gilt $t_1 = t$ auf X_1 und $t_2 = t$ auf X_2.

- **Umbenennung**: Die Umbenennung $\zeta_\beta\rho$ mit $\rho : \mathbb{D}^X \to K$ und $\beta : X \to Y$ ist eine K-Relation über Y, welche durch

$$(\zeta_\beta\rho)(t) := \rho(t \circ \beta).$$

  erklärt ist.

□

Beispiel 3.2. *Für positive relationale Algebren können beispielsweise* $(\mathbb{B}, \wedge, \vee, \text{true}, \text{false})$ *mit* $\mathbb{B} = \{\text{true}, \text{false}\}$ *und* $(\mathbb{N}, +, \cdot, 0, 1)$ *definiert werden.*

Aufbauend auf der Definition der positiven Algebra kann nun der *Positive-Algebra-Provenance-Halbring* oder kurz *Provenance-Halbring* definiert werden. Er bildet die Basis der Provenance-Polynome, da er die Operationen der positiven Algebra auf τ-Polynome über $\mathbb{N}$ zurückführt.

Definition 3.8 (Provenance-Halbring)**.** Sei τ eine Menge von Tupelidentifikatoren über einer Datenbankinstanz I. Ein **Provenance-Halbring** von I ist ein Halbring von Polynomen mit Variablen aus τ und Koeffizienten aus $\mathbb{N}$ mit den üblichen Operationen: $(\mathbb{N}[\tau], +, \cdot, 0, 1)$. □

Betrachten wir erneut das Beispiel aus Abschnitt 1.2, so können die einzelnen Relationen auch als K-Relationen aufgefasst werden. So entspricht die STUDENTEN-Relation etwa der $\mathbb{N}[S_1, \cdots S_8]$-Relation. Zum besseren Verständnis im Umgang mit den Provenance-Polynomen sollen für die Ergebnistabelle der unten stehenden Anfrage Q_3 exemplarisch die Provenance-Polynome berechnet werden.

Anfrage 3.3 Modulbelegung eines Studenten unter gegebenen Nebenbedingungen (Anfrage Q_3)

```
Untersuche die Module des Modulkatalogs auf folgende Eigenschaften:
– Wähle alle Module der Vertiefung 'Informationssysteme' sowie
– alle von Professor A gehaltenen Module,
die der Student mit der Matrikelnummer 5 belegt hat.
```

Beispiel 3.3. *Gegeben seien die* MODUL-, DOZENTEN- *und* TEILNEHMER-*Relationen aus Anhang B sowie die obige Anfrage* Q_3*. Die drei Tabellen können jeweils als* $\mathbb{N}[M_1, M_2, \cdots M_9]$-, $\mathbb{N}[D_{1.1}, D_{1.2}, D_2, \cdots, D_9]$- *und* $\mathbb{N}[T_1, \cdots T_{26}]$-*Relation aufgefasst werden. Für dieses Beispiel haben wir die Besonderheit der Tupelidentifikatoren* $D_{1.1}$ *und* $D_{1.2}$ *eingeführt. Wir werden zeigen, dass die beiden Tupel für die hier untersuchten Anfragen* Q_3*,* Q_3' *und* Q_3'' *zu einer Information „verschmelzen“.*

Die Berechnung der Provenance-Informationen der Anfrageergebnis-Tabelle erfolgt in mehreren Schritten. Hierfür wird die Anfrage zunächst kodiert und anschließend von innen nach außen interpretiert.

- *Kodierung der Attribute und Attributwerte:*
  - A_1 : Modulnr
  - A_2 : Titel
  - A_3 : Matrikelnr
  - A_4 : Vertiefung
  - A_5 : Dozent
  - N_1 : 'Informationssysteme'
  - N_2 : 'Professor A'
  - M : MODULE-*Relation B.2*
  - D : DOZENTEN-*Relation B.3*
  - T : TEILNEHMER-*Relation B.4*

- *Formalisierung der Anfrage:*

$$\begin{aligned} Q_3 \quad = \quad & \pi_{A_2}[\pi_{A_1A_2}(\sigma_{A_4=N_1}(M) \bowtie D) \bowtie \pi_{A_1}(\sigma_{A_3=5}(T))] \\ & \cup\ \pi_{A_2}[\pi_{A_1A_2}(M \bowtie \sigma_{A_5=N_2}(D)) \bowtie \pi_{A_1}(\sigma_{A_3=5}(T))] \end{aligned}$$

- *Aufstellen der Provenance-Informationen:*

Die erste Teilanfrage $\pi_{A_1A_2}(\sigma_{\text{'Datenbanken'}=A_4}(\text{MODULE}) \bowtie \text{DOZENTEN})$ *liefert alle angebotenen Module der Vertiefungsrichtung* Informationssysteme. *Dies sind die Module* Datenbanken III, Individuelles Wissensmanagement *und* Theorie relationaler Datenbanken. *Da der JOIN der Ausgangstabellen B.2 und B.3 die* Datenbanken III *zwei Mal enthält, ergibt sich als Provenance-Information die Summe der beiden zugehörigen* ID*s.*

$\pi_{A_1A_2}(\sigma_{A_4=N_1}(M) \bowtie D)$:

A_1	A_2	
001	Datenbanken III	$M_1 \cdot_K D_{1.1} + M_1 \cdot_K D_{1.2}$
004	Individuelles Wissensmanagement	$M_4 \cdot_K D_4$
007	Theorie relationaler Datenbanken	$M_7 \cdot_K D_7$

Tabelle 3.7. Zwischenergebnis $\pi_{A_1A_2}(\sigma_{A_4=N_1}(M) \bowtie D)$ der Anfrage 3.3 (mit Provenance-Informationen)

Der JOIN zwischen der MOUDULE- *sowie der* DOZENTEN-*Relation ist in der obigen Anfrage aufgrund der Nicht-Exisitenz von dangling tuplen (siehe Definition 4.6) überflüssig. Das ist hier aber explizit gewollt, um den Umgang mit Duplikaten zu simulieren. Ohne diese zusätzliche Operation erhielten wir die Polynome* M_1*,* M_4 *und* M_7*.*

$\pi_{A_1A_2}(M \bowtie \sigma_{A_5=N_2}(D))$:

A_1	A_2	
001	Datenbanken III	$M_1 \cdot_K D_{1.1}$
007	Theorie relationaler Datenbanken	$M_7 \cdot_K D_7$
009	NEidI	$M_9 \cdot_K D_9$

Tabelle 3.8. Zwischenergebnis $\pi_{A_1A_2}(M \bowtie \sigma_{A_5=N_2}(D))$ der Anfrage 3.3 (mit Provenance-Informationen)

Die zweite Teilanfrage $\pi_{A_1A_2}(\text{DOZENTEN} \bowtie \sigma_{A_5=\text{'Professor A'}}(\text{MODULE}))$ *liefert alle von* Professor A *gehaltenen Module:* Datenbanken III, Theorie relationaler Datenbanken *sowie* NEidI. *Das Modul* Datenbanken III *liegt hierbei in der Kombination* (001, Datenbanken III, Informationssysteme, Professor A) *nur einmal vor, sodass in diesem Fall die Provenance-Information lediglich aus einem* ID-*Produkt*

besteht. Das Produkt entsteht hierbei durch den natürlichen Verbund der beiden Relationen DOZENTEN *und* MODULE. *Vergleiche hierzu das Zwischenergebnis der Tabelle 3.8.*

$\pi_{A_1}(\sigma_{A_3=5}(T))$:

A_1	
001	T_4
002	T_9
004	T_{14}
006	T_{19}
007	T_{22}
009	T_{26}

Tabelle 3.9. Zwischenergebnis $\pi_{A_1}(\sigma_{A_3=5}(T))$ der Anfrage 3.3 (mit Provenance-Informationen)

Die von dem Studenten mit der Matrikelnummer 5 belegten Module werden durch die dritte Teilanfrage $\pi_{A_1}(\sigma_{A_3=5}(\text{TEILNEHMER}))$ *geliefert (siehe Tabelle 3.9). Die Verknüpfung über den natürlichen Verbund erzeugt erneut eine Multiplikation passender Provenance-Informationen, während die Vereinigung eine zweite Summation zur Folge hat.*

- *Ergebnistabelle mit Provenance-Informationen:*

Q_3 :

A_2	
Datenbanken III	$(M_1 \cdot_K D_{1.1} + M_1 \cdot_K D_{1.2}) \cdot_K T_4 +_K M_1 \cdot_K T_4$
Individuelles Wissensmanagement	$M_4 \cdot_K D_4 \cdot T_9$
Theorie relationaler Datenbanken	$2 \cdot_K M_7 \cdot D_7 \cdot_K T_{22}$
NEidI	$M_9 \cdot_K D_9 \cdot T_{26}$

Tabelle 3.10. ***how***-Provenance der Anfrage 3.3 mit zugehörigen Provenance-Polynomen

Die Provenance-Informationen der Tabelle 3.10 sind korrekt berechnet, aber keinesfalls minimal. Die zugehörige Zeugenbasis

$$\{\{M_1, D_{1.1}, D_{1.2}, T_4\}, \{M_4, D_4, T_9\}, \{M_7, D_7, T_{22}\}, \{M_9, D_9, T_{26}\}\}$$

kann beispielsweise auf

$$\{\{M_1, D_{1.1}, T_4\}, \{M_4, D_4, T_9\}, \{M_7, D_7, T_{22}\}, \{M_9, D_9, T_{26}\}\}$$

minimiert werden. Grund ist der oben angesprochene überflüssige JOIN in $\pi_{A_1A_2}(\sigma_{A_4=N1}(\text{MODULE}) \bowtie (\text{DOZENTEN}))$. *Wird diese zusätzliche Operation vernachlässigt, kann die Anfrage* Q_3 *zu einer zweiten äquivalenten Anfrage* Q_3' *umgeformt werden.*

- *Umformulierung der Anfrage:*

Die zu Q_3 *äquivalenten (optimierteren) Anfragen*

$$Q_3' = \pi_{A_2}[\pi_{A_1A_2}(\sigma_{A_4=N_1}(M)) \bowtie \pi_{A_1}(\sigma_{A_3=5}(T))] \cup \pi_{A_2}[\pi_{A_1A_2}(M \bowtie \sigma_{A_5=N_2}(D)) \bowtie \pi_{A_1}(\sigma_{A_3=5}(T))]$$

und

$$Q_3'' = \pi_{A_2}[(\pi_{A_1A_2}(\sigma_{A_4=N_1}(M)) \cup \pi_{A_1A_2}(M \bowtie \sigma_{N_2=A_5}(D))) \bowtie \pi_{A_1}(\sigma_{A_3=5}(T))]$$

liefern die selbe Ergebnistabelle wie die Anfrage Q_3, aber andere Provenance-Polynome der einzelnen Ergebnistupel (vergleiche Tabelle 3.10 und 3.11).

Q_3' :

A_2	
Datenbanken III	$M_1 \cdot_K D_{1.1} \cdot_K T_4 +_K M_1 \cdot_K T_4$
Individuelles Wissensmanagement	$M_4 \cdot_K D_4 \cdot T_9$
Theorie relationaler Datenbanken	$M_7 \cdot_K T_{22} +_K M_7 \cdot D_7 \cdot_K T_{22}$
NEidI	$M_9 \cdot_K D_9 \cdot T_{29}$

Q_3'' :

A_2	
Datenbanken III	$(M_1 + M_1 \cdot_K D_{1.1}) \cdot_K T_4$
Individuelles Wissensmanagement	$M_4 \cdot_K D_4 \cdot T_9$
Theorie relationaler Datenbanken	$(M_7 +_K M_7 \cdot D_7) \cdot_K T_{22}$
NEidI	$M_9 \cdot_K D_9 \cdot T_{29}$

Tabelle 3.11. ***how***-Provenance der zu Q_3 äquivalenten Anfragen Q_3' und Q_3'' mit zugehörigen Provenance-Polynomen

Aufgrund der Distributivität von $\cdot_K$ über $+_K$ sowie der Kommutativität der Addition $+_K$ — beide Eigenschaften gelten nach Definition 3.5 — können die Polyome des ersten und dritten Ergebnistupels aus Tabelle 3.11 ineinander überführt werden:

$$\begin{aligned} M_1 \cdot_K D_{1.1} \cdot_K T_4 +_K M_1 \cdot_K T_4 &\overset{Kommutativität}{=} M_1 \cdot_K T_4 +_K M_1 \cdot_K D_{1.1} \cdot_K T_4 \\ &\overset{Distributivität}{=} (M_1 + M_1 \cdot_K D_{1.1}) \cdot_K T_4 \\ M_7 \cdot_K T_{22} +_K M_7 \cdot D_7 \cdot_K T_{22} &\overset{Distributitvität}{=} (M_7 +_K M_7 \cdot D_7) \cdot_K T_{22} \end{aligned}$$

Die Güte der Anfrage, d. h. der Optimierungsgrad der Anfrage, spielt für das Aufstellen der Provenance-Polynome also keine (übergeordnete) Rolle, sofern der Anfrage eine minimale Zeugenbasis zu Grunde liegt (wie im Fall der Anfragen Q_3' und Q_3''). Führt eine Umformulierung der Anfrage jedoch zu einer anderen Zeugenbasis (wie die Umformung der Anfrage Q_3 in die Anfrage Q_3'), ergeben sich unterschiedliche Provenance-Polynome. Diese Behauptung muss jedoch im Allgemeinen noch näher untersucht werden.

Seien dazu zwei Anfragen mit derselben minimalen Zeugenbasis gegeben. Dann ergeben sich zwei Provenance-Polynome, die bis auf Kommutativität und Distributivität übereinstimmen. Der nachfolgende Satz fasst dies wie folgt zusammen:

Satz 3.1. *Seien zwei schlüsselerhaltende Anfragen Q und Q' gegeben. Sei $W_{Q,d}(t)$ die (minimale) Zeugenbasis einer Menge von Polynomen p_i und $W_{Q',d}(t)$ die (minimale) Zeugenbasis einer Menge von Polynomen p_i'. Dann gilt:*

$$W_{Q,d}(t) = W_{Q',d}(t) \Rightarrow p_i = p_i'.$$

Beweis. Seien $W_{Q,d} = \{W_1, ..., W_n\}$ und $W_{Q',d} = \{W_1', ..., W_n'\}$ mit $W_i = \{w_{i_1}, ..., w_{i_k}\}$, $W_i' = \{w_{i_1}', ..., w_{i_k}'\}$ gegeben. Sei weiter $p_i = a_{i_1 i_1} \cdot_K w_{i_1} \cdot_K w_{i_1} +_K ... +_K a_{i_1 i_n} \cdot_K w_{i_1} \cdot_K w_{i_n} +_K ... +_K a_{i_k i_k} \cdot_K w_{i_k} \cdot_K w_{i_k}$ mit $a_{i_j i_l} \in [0, 1]$ das zu $W_{Q,d}$ gehörende Provenance-Polynom und p_i' das über $a_{i_j i_l}'$ analog definiert Polynom für $W_{Q',d}$. Dann gilt:

$$\begin{aligned} W_{Q,d} = W_{Q',d} &\Rightarrow \{W_1, ..., W_n\} = \{W_1', ..., W_n'\} \\ &\Rightarrow W_i = W_i' \quad \forall i \in \{1, ..., n\} \end{aligned}$$

Sei $i \in \{1, ..., n\}$ beliebig, aber fest.

$$
\begin{aligned}
&\Rightarrow \quad \{w_{i_1}, ..., w_{i_k}\} = \{w'_{i_1}, ..., w'_{i_k}\} \\
&\Rightarrow \quad w_{i_j} = w'_{i_j} \quad \forall j \in \{1, ..., k\} \\
&\Rightarrow \quad a_{i_1 i_1} \cdot_K w_{i_1} \cdot_K w_{i_1} +_K ... +_K a_{i_k i_k} \cdot_K w_{i_k} \cdot_K w_{i_k} \\
&\qquad\quad = a_{i_1 i_1} \cdot_K w'_{i_1} \cdot_K w'_{i_1} +_K ... +_K a_{i_k i_k} \cdot_K w'_{i_k} \\
&\overset{(\star)}{\Rightarrow} \quad \begin{cases} a'_{i_j i_j} = 0 & \text{Duplikatfreiheit} \\ a'_{i_j i_l} = \lambda \cdot a_{i_j i_l} \text{ mit } a_{i_j i_l} + a_{i_l i_j} = a'_{i_j i_l} + a'_{i_l i_j}, \lambda \in \mathbb{N}^+ & \text{sonst} \end{cases} \\
&\Rightarrow \quad p_i = p'_i.
\end{aligned}
$$

□

Beispiel 3.4. *Gegeben seien eine Anfrage Q mit* **how**-*Provenance*

$$p = 3 \cdot_K w_2 \cdot_K w_1 +_K 1 \cdot_K w_2 \cdot_K w_3 +_K 1 \cdot_K w_3 \cdot w_2$$

sowie eine Anfrage Q' mit **how**-*Provenance*

$$p' = 3 \cdot_K w'_1 \cdot_K w'_2 +_K 2 \cdot_K w'_3 \cdot_K w'_2.$$

Dann gilt $W = \{w_1, w_2, w_3\} = W'$ und weiter

$$a_{11} = a_{22} = a_{33} = 0 \quad \Rightarrow \quad a_{11} = a_{22} = a_{33} = 0$$

$$a_{12} = 0, a_{21} = 3 \quad \Rightarrow \quad \begin{cases} a'_{12} = 0, a'_{21} = 3 \\ a'_{12} = 1, a'_{21} = 2 \\ a'_{12} = 2, a'_{21} = 1 \\ a'_{12} = 3, a'_{21} = 0 \end{cases}$$

$$a_{13} = 0, a_{31} = 0 \quad \Rightarrow \quad a'_{13} = 0, a'_{31} = 0$$

$$a_{23} = 1, a_{32} = 1 \quad \Rightarrow \quad \begin{cases} a'_{23} = 0, a'_{32} = 2 \\ a'_{23} = 1, a'_{32} = 1 \\ a'_{23} = 2, a'_{32} = 0 \end{cases}$$

Für $a_{12} = 3$ und $a_{32} = 2$ gilt somit $p = p'$.

Die Implikation $(\star)$ im Beweis zu Satz 3.1 basiert auf der folgenden Überlegung: Potenzen entstehen durch gleichzeitige Anwendung von $\cup$ und $\bowtie$ auf dieselben Attribute (vlg. Beispiel 3.5). Duplikate sind nach Voraussetzung aber nicht erlaubt.

Beispiel 3.5. *Seien eine Anfrage*

$$Q = \pi_{AC}((\pi_{AB}(r) \bowtie \pi_{BC}(r)) \cup (\pi_{AC}(r) \bowtie \pi_{BC}(r))),$$

ein Relation $r(\mathcal{R}) = \{(a, b, c), (d, b, e), (f, g, e)\}$ über einem Relationenschema $\mathcal{R} = \{A, B, C\}$ und Tupelidentifikatoren p, r, s gegeben. Dann enthalten einige Ergebnispolynome aufgrund von Duplikatbildung Potenzen:

Q :

A	C	
a	c	$2 \cdot_K p^2$
a	e	$p \cdot_K r$
d	c	$p \cdot_K r$
d	e	$2 \cdot_K r^2 +_K r \cdot_K s$
f	e	$s \cdot_K s^2 +_K r \cdot_K s$

Tabelle 3.12. Ergebnis der Anfrage Q mir Provenance-Informationen

Zusammenfassend besteht die Provenance-Halbring-Theorie aus

- einem kommutativen Halbring K;
- K-Relationen, welche die Tupel einer Datenbank mit ausgezeichneten Elementen aus K versehen;
- einer generalisierten positiven Algebra auf K-Relationen;
- einem Provenance-Halbring, bestehend aus Polynomen mit ganzzahligen Koeffizienten.

Im nächsten Abschnitt sollen diese Provenance-Ringe um die Aggregation erweitert werden.

3.2.2. Provenance für aggregierte Anfragen

Gegeben sei ein kommutativer Monoid M, wie beispielsweise SUM $= (\mathbb{R}, +, 0)$ für die Summation, MIN $= (\mathbb{R}^{\pm\infty}, \min, +\infty)$ für die Angabe des Minimums oder PROD $= (\mathbb{R}, \times, 1)$ für das Produkt. COUNT und AVG können als Spezialfall von SUM bzw. als Kombination von COUNT und SUM aus den obigen drei Operationen berechnet werden. Dann kann hierfür der *K-Halbmodul* sowie die *M-Aggregation* definiert werden.

Definition 3.9 (K-Halbmodul)**.** Gegeben sei ein kommutativer Halbring K. Dann heißt eine Struktur $(M, +_M, 0_M, \star_M)$ **K-Halbmodul**, wenn $(M, +_M, 0_M)$ ein kommutativer Monoid ist und $\star_M$ eine binäre Operation $K \times M \to M$ ist, so dass gilt:

- $\forall k \in K, m_1, m_2 \in M : k \star_M (m_1 +_M m_2) = k \star_M m_1 +_M k \star_M m_2$
- $\forall k \in K : k \star_M 0_M = 0_M$
- $\forall k_1, k_2 \in K, m \in M : (k_1 +_K k_2) \star_M m = k_1 \star_M m +_M k_2 \star_M m$
- $\forall m \in M : 0_K \star_M m = 0_M$
- $\forall k_1, k_2 \in K, m \in M : (k_1 \cdot_K k_2) \star_M m = k_1 \star_M (k_2 \star_M m)$
- $\forall m \in M : 1_k \star_M m = m$

$\boxdot$

Bemerkung. In jedem (kommutativen) Monoid $(M, +_M, \cdot_M)$ kann für jedes $n \in \mathbb{N}$ und $x \in M$ statt $\underbrace{x +_M \cdots +_M x}_{n\text{-mal}}$ auch nx geschrieben werden. Für den Spezialfall $n = 0$ wird $0x$ als 0_M definiert.

Gegeben sei ein K-Halbmodul M mit $M \subseteq \mathbb{D}$. Dann kann für eine Menge S_K mit $\text{supp}(S_K) = \{m_1, ..., m_n\}$ und $S_K(m_i) = k_i \in K$ $(i = 1, ..., n)$ die M**-Aggregation**

$$k_1 \star_M m_1 +_M \cdots +_M k_n \star_M m_n \in M$$

mit dem Spezialfall, dass die Aggregation der leeren Menge gerade 0_M entspricht, definiert werden. Die Elemente dieses K-Halbmodules $K \times M$ heißen dabei auch **einfache Tensoren**.

Als nächstes sollen Multimengen dieser einfachen Tensoren betrachtet werden, die zusammen mit der Multimengenvereinigung ($+_{K\otimes M}$) und der leeren Multimenge ($0_{K\otimes M}$) einen kommutativen Monoid bildet. Für jede nicht-leere Multimenge von einfachen Tensoren kann analog zur Definition oben

$$k_1 \otimes m_1 +_{K\otimes M} \cdots +_{K\otimes M} k_n \otimes m_n$$

geschrieben werden. Mit $k \star_{K\otimes M} \sum k_i \otimes m_i = \sum (k \cdot_K k_i) \otimes m_i$ definiert $K \otimes M$ zudem einen K-Halbmodul. Dieser beschreibt die Menge der **Tensoren**, d.h. die Äquivalenzklassen von Multimengen von einfachen Tensoren modulo $\sim$.

Bemerkung. Sei $\sim$ die kleinste Kongruenz bzgl. $+_{K\otimes M}$ und $\cdot_{K\otimes M}$. Dann gilt:

- $\forall k_1, k_2 \in K, m \in M : (k_1 +_K k_2) \otimes m \sim k_1 \otimes m +_{K\otimes M} k_2 \otimes m$
- $\forall m \in M : 0_K \otimes m \sim 0_{K\otimes M}$
- $\forall k \in K, m_1, m_2 \in M : k \otimes (m_1 +_M m_2) \sim k \otimes m_1 +_{K\otimes M} k \otimes m_2$
- $\forall k \in K : k \otimes 0_M \sim 0_{K\otimes M}$

Das Tensorprodukt erfüllt somit die üblichen Eigenschaften der Links- und Rechts-Distributivität sowie der Existenz eines neutralen Elementes 0_K. Des weiteren gilt:

Satz 3.2. *$K \otimes M$ definiert einen K-Halbmodul.*

Beweis. Nachweis der sechs Eigenschaften aus Definition 3.9 mit Hilfe der Kongruenzen aus obiger Bemerkung:

$$\begin{aligned}
k \star_{K\otimes M} (\sum k_{1_i} \otimes m_{1_i} +_{K\otimes M} \sum k_{2_i} \otimes m_{2_i}) &= k \star_{K\otimes M} (\sum (k_{1_i} +_K k_{2_i}) \otimes (m_{1_i} +_M m_{2_i})) \\
&= \sum (k \cdot_K k_{1_i} +_K k \cdot_K k_{2_i}) \otimes (m_{1_i} +_M m_{2_i}) \\
&= \sum (k \cdot_K k_{1_i}) \otimes m_{1_i} + \sum (k \cdot_K k_{2_i}) \otimes m_{2_i} \\
&= \sum (k \cdot_K k_{1_i}) \otimes m_{1_i} +_{K\otimes M} \sum (k \cdot_K k_{2_i} \otimes m_{2_i} \\
&= k \star_{K\otimes M} \sum k_{1_i} \otimes m_{1_i} +_{K\otimes M} k \star_{K\otimes M} \sum k_{2_i} \otimes m_{2_i}
\end{aligned}$$

$$\begin{aligned}
k \star_{K\otimes M} 0_{K\otimes M} &= \sum k \cdot_K 0_{K\otimes M} \\
&= 0_{K\otimes M}
\end{aligned}$$

$$\begin{aligned}
(k_1 +_K k_2) \star_{K\otimes M} \sum k_i \otimes m_i &= \sum (k_1 \cdot_K k_i +_K k_2 \cdot_K k_i) \otimes m_i \\
&\sim \sum (k_1 \cdot_K k_i) \otimes m_i +_{K\otimes M} (k_2 \cdot_K k_i) \otimes m_i \\
&= \sum (k_1 \cdot_K k_i) \otimes m_i +_{K\otimes M} \sum (k_2 \cdot_K) \otimes m_i \\
&= k_1 \star_{K\otimes M} \sum k_i \otimes m_i +_{K\otimes M} k_2 \star_{K\otimes M} \sum k_i \otimes m_i
\end{aligned}$$

$$\begin{aligned}
0_K \star_{K\otimes M} \sum k_i \otimes m_i &= \sum 0_K \cdot_K k_i \otimes m_i \\
&= \sum 0_K \otimes m_i
\end{aligned}$$

$$
\begin{aligned}
&\sim && \sum 0_{K\otimes M} \\
&= && 0_{K\otimes M}
\end{aligned}
$$

$$
\begin{aligned}
(k_1 \cdot_K k_2) \star_{K\otimes M} \sum k_i \otimes m_i &= \sum (k_1 \cdot_K k_2 \cdot_K k_i) \otimes m_i \\
&= k_1 \star_{K\otimes M} \sum (k_2 \cdot_K) \otimes m_i \\
&= k \star_{K\otimes M} (k_2 \star K \otimes M \sum k_i \otimes m_i)
\end{aligned}
$$

$$
\begin{aligned}
1_K \star_{K\otimes M} \sum k_i \otimes m_i &= \sum 1_K \cdot_K k_i \otimes m_i \\
&= \sum k_i \otimes m_i
\end{aligned}
$$

□

$K \otimes M$ ist somit eine Verallgemeinerung der Operationen der lineare Algebra. Zum Abschluss dieses Abschnitts werden alle Erkenntnisse zur Provenance-Bestimmung ***how***, ***why*** und ***where*** in der Analyse der Anfrage Q_4 zusammengefasst. Gesucht wird die Durchschnittsnote der Studenten mit dem Vornamen *Max*.

Anfrage 3.4 Durchschnittsnote der Studenten mit Vornamen 'Max' (Anfrage Q_4)

```
SELECT AVG(Note)
FROM Studenten JOIN Noten
ON (Studenten.Matrikelnr = Noten.Matrikelnr)
WHERE Studenten.Vorname = 'Max'
```

Beispiel 3.6. *Gegeben seien die* STUDENTEN- *und* NOTEN-*Relation aus Abschnitt B sowie die Anfrage Q_4. Hierfür wird zunächst ein Zwischenergebnis mit Angabe der zugehörigen Provenance-Informationen berechnet, welches der späteren Anfrage Q_6 entspricht (siehe Unterabschnitt 3.3.2). Anschließend wird das Provenance-Polynom für die* ***how****-Provenance aufgestellt und hieraus die* ***why****- und* ***where****-Provenance abgelesen.*

- *Zwischenergebnis:*

Z_{Q_4} :

Matrikelnr	Note	
3	2.3	$S_3 \cdot_K N_7$
3	1.3	$S_3 \cdot_K N_{13}$
3	1.7	$S_3 \cdot_K N_{20}$
7	3.3	$S_7 \cdot_K N_{11}$
7	4.0	$S_7 \cdot_K N_{16}$

Tabelle 3.13. Zwischenergebnis $Z_{Q_4} = \pi_{\text{Matrikelnr,Note}}(\text{STUDENTEN} \bowtie \text{NOTEN})$ der Anfrage 3.4

- *Endergebnis:*

Q_4 :

Note
2.52

Tabelle 3.14. Ergebnis der Anfrage 3.4

- *Provenance-Polynom (**how**-Provenance):*

$$\frac{2.3 \otimes \overbrace{S_3 \cdot_K N_7}^{p_1} +_{K\otimes M} 1.3 \otimes \overbrace{S_3 \cdot_K N_{13}}^{p_2} +_{K\otimes M} 1.7 \otimes \overbrace{S_3 \cdot_K N_{20}}^{p_3} +_{K\otimes M} 3.3 \otimes \overbrace{S_7 \cdot_K N_{11}}^{p_4} +_{K\otimes M} 4.0 \otimes \overbrace{S_7 \cdot_K N_{16}}^{p_5}}{\underbrace{S_3 \cdot_K N_7}_{p_1} +_{K\otimes M} \underbrace{S_3 \cdot_K N_{13}}_{p_2} +_{K\otimes M} \underbrace{S_3 \cdot_K N_{20}}_{p_3} +_{K\otimes M} \underbrace{S_7 \cdot_K N_{11}}_{p_4} +_{K\otimes M} \underbrace{S_7 \cdot_K N_{16}}_{p_5}}$$

*Das Provenance-Polynom der **how**-Provenance besteht aus dem Quotienten von* **SUM**(Note) *und* **COUNT**(Note)*, da nach Definition des Durchschnitts* $\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ *und somit* $\mathbf{AVG}(x) = \frac{\mathbf{SUM}(x)}{\mathbf{COUNT}(x)}$ *gilt. Die Aufzählung* **COUNT** *ist ein Spezialfall der Summation, sodass die Wahl des Monoids M auf den Summationsmonoiden SUM* $= (\mathbb{R}, +, 0)$ *fällt.*

- *Zeugenbasis (**why**-Provenance):*

$$\{\{S_3, N_7\}, \{S_3, N_{13}\}, \{S_3, N_{20}\}, \{S_7, N_{11}\}, \{S_7, N_{16}\}\}$$

*Die **why**-Provenance lässt sich durch Auflistung aller in einem Teilpolynom* p_i *vorkommenden Tupelidentifikatoren aus der **how**-Provenance herleiten. Die Teilpolynome* p_i *entsprechen hierbei den Poylnomen aus Tabelle 3.13, also* $p_1 = S_3 \cdot_K N_7, ..., p_5 = S_7 \cdot_K N_{16}$*.*

- ***where**-Provenance:*

STUDENTEN, NOTEN

*Die Ausgabe der Relationennamen aller im Ergebnis vorkommenden Tupel entspricht der **where**-Provenance. Sie folgt somit direkt aus der **why**-Provenance.*

Die Beziehung zwischen der ***how***- und ***why***-Provenance soll für einfach Polynome ohne Aggregation im folgenden Theorem zusammengefasst werden. Dieses muss jedoch noch bewiesen und auf Polynome mit Aggregation erweitert werden.

Satz 3.3. *Gegeben sei eine Menge von Tupeln* $t_1, ..., t_n$ *sowie eine Anfrage Q. Sei weiter* p_i *das zum Polynom* t_i *gehörige Polynom*

$$p_i = a_{i_1 i_1} \cdot_K t_{i_1} \cdot_K t_{i_1} +_K ... +_K a_{i_1 i_n} \cdot_K t_{i_1} \cdot_K t_{i_n} +_K ... +_K a_{i_k i_k} \cdot_K t_{i_k} \cdot_K t_{i_k}$$

Dann ist $Z = \{t_{i_j} \mid a_{i_j i_j}, a_{i_j i_k}, a_{i_k i_j} \neq 0\}$ *Zeugenbasis, d.h. Antwort auf die* **why***-Anfrage. Die* **where***-Provenance besteht aus den Relationennamen der Tupelidentifkatoren* $t_{i_j} \in Z$*, sofern* t_{i_j} *Teil des Ergebnisses der Anfrage Q ist.*

Neben den Grundoperationen Selektion σ, Projektion π, Umbenennung β, natürlicher Verbund $\bowtie$ sowie Vereinigung $\cup$ können nun auch die klassischen Aggregatfunktionen **MAX**, **MIN**, **COUNT**, **SUM** und **AVG** verarbeitet werden. Diese bilden in Form eines speziellen Monoids zusammen mit einem kommutativen Provenance-Halbring die Basis der „neuen“ Polynome.

3.2.3. Provenance und GROUP BY

Die Provenance-Untersuchung für Aggregat-Anfragen kann nach Y. Amsterdamer, D. Deutch und V. Tannen noch um die Operation GROUP BY erweitert werden [ADT11]. Die Idee besteht in der Gruppierung nach ihren Gruppierungsattributen mit anschließender gruppenweiser Aggregation. Die Berechnung der neuen Polynome enthält nun die Schwierigkeit, die neutralen Elemente 1_K und 0_K zu erhalten. Hierfür wird die Definition 3.5 um einen zusätzlichen Konstruktor γ wie folgt erweitert:

Definition 3.10 ((Kommutativer) γ-Halbring)**.** Gegeben seien ein kommutativer Halbring $(K, +_K, \cdot_K, 0_K, 1_K)$ und eine binäre Operation $\gamma : K \to K$ mit

$$\gamma_K(0_K) = 0_K \tag{3.1}$$
$$\gamma_K(n1_K) = 1_K \text{ für alle } n \geq 1. \tag{3.2}$$

Dann heißt die algebraische Struktur $(K, +_K, \cdot_K, 0_K, 1_K, \gamma_K)$ (**kommutativer**) **γ-Halbring**. Die Regeln 3.1 und 3.2 werden auch **γ-Regeln** genannt. ⊡

Bemerkung. Der binäre Operator γ entspricht dem Gruppierungsoperator aus Definition 2.15.

Für die Wahl der Funktion γ_K existieren einige Freiheiten. Einzig die Operationen $\gamma_{\mathbb{B}}$ und $\gamma_{\mathbb{N}}$ sind durch die γ-Regeln bereits eindeutig definiert. Ein kommutativer γ-Halbring $\mathbb{N}[X, \gamma]$ kann aus einer Menge X konstruiert werden, indem der Quotient aus der Menge der $\{+, \cdot, 0, 1, \gamma\}$-algebraischen Ausdrücke durch Gleichheitsbedingungen auf dem kommutativen Halbring (vlg. Beispiel 3.6) sowie den γ-Regeln erzeugt wird.

Definition 3.11 (Aggregationsergebnis)**.** Gegeben seien ein kommutativer Halbring K sowie ein kommutativer Monoid M. Sei ρ eine K-Relation auf einer Menge von Attributen X. Sei $Y \subseteq X$ eine gruppierte Teilmenge von Attributen und $Z \subseteq X$ mit $Y \cap Z = \emptyset$ eine Teilmenge von Attributen mit Werten in M, welche aggregiert werden sollen. Für jedes Tupel t sei $T = \{t' \in \text{supp}(X) \mid \forall x \in Y : t'(x) = t(x)\}$. Dann ist das **Aggregationsergebnis** $\rho' = \gamma_{Y,Z}(\rho)$ definiert als

$$\rho'(t) = \begin{cases} \gamma_K(\Sigma_{t' \in T} \rho(t')), & \text{falls } T \neq \emptyset \text{ und } \forall x \in Z : t(x) = \Sigma_{t' \in T} \rho(t') \otimes t'(x) \\ 0, & \text{sonst} \end{cases}$$

⊡

Anfrage 3.5 Jeweilige Durchschnittsnote des Studenten mit Vornamen 'Max' (Anfrage Q_5)

```
SELECT Matrikelnr, AVG(Note)
FROM Studenten JOIN Noten
ON (Studenten.Matrikelnr = Noten.Matrikelnr)
GROUP BY Studenten.Matrikelnr
WHERE Studenten.Vorname = 'Max'
```

Beispiel 3.7. *Als Zwischenergebnis $Z_{Q_5} = \pi_{Matrikelnr,Note}(STUDENTEN \bowtie NOTEN)$ der Anfrage Q_5 sei die folgende Tabelle 3.15 gegeben. Aufgrund der Ähnlichkeit der Anfrage Q_5 zur Anfrage Q_4 (Beispiel 3.6) stimmt dieses mit der Tabelle 3.13 überein.*

Z_{Q_5} :

Matrikelnr	Note	
3	2.3	$S_3 \cdot_K N_7$
3	1.3	$S_3 \cdot_K N_{13}$
3	1.7	$S_3 \cdot_K N_{20}$
7	3.3	$S_7 \cdot_K N_{11}$
7	4.0	$S_7 \cdot_K N_{16}$

Tabelle 3.15. Zwischenergebnis $Z_{Q_5} = \pi_{\text{Matrikelnr,Note}}(\text{STUDENTEN} \bowtie \text{NOTEN})$ der Anfrage 3.5

Die Anfrage Q_5 liefert zwei Ergebnisse: Die Durchschnittsnote des Studenten mit der Matrikelnummer 3 sowie die Durchschnittsnote des Studenten mit der Matrikelnummer 7. Es ergibt sich für die Anfrage $\gamma_{Matrikelnr,AVG(Note)}(NOTEN \bowtie STUDENTEN)$ also die folgende Tabelle 3.16 mit den Polynomen

$$p_1 = \gamma_K(\frac{2.3 \otimes S_3 \cdot_K N_7 +_{K\otimes M} 1.3 \otimes S_3 \cdot_K N_{13} +_{K\otimes M} 1.7 \otimes S_3 \cdot_K N_{20}}{S_3 \cdot_K N_7 +_{K\otimes M} S_3 \cdot_K N_{13} +_{K\otimes M} S_3 \cdot_K N_{20}})$$
$$p_2 = \gamma_K(\frac{3.3 \otimes S_7 \cdot_K N_{11} +_{K\otimes M} 4.0 \otimes S_7 \cdot_K N_{16}}{S_7 \cdot_K N_{11} +_{K\otimes M} S_7 \cdot_K N_{16}}).$$

Die Wahl des Monoids M fällt ebenfalls auf den Summationsmonoiden SUM $= (\mathbb{R}, +, 0)$*. Die Begründung liegt analog zu Beispiel 3.6 in den Eigenschaften von* **COUNT** *— als Spezialfall von* **SUM** *— und* **AVG** *— als Quotient von* **SUM** *und* **COUNT***.*

Q_5 :

Matrikelnr	Note	
3	1.8	p_1
7	3.6	p_2

Tabelle 3.16. Ergebnis der Anfrage 3.5

Die hier beschriebene relationale Algebra wird von Amsterdamer et al. auch als **SPJU-AGB** bezeichnet. In anderen Werken findet man sie unter dem Begriff *SPUJA mit Aggregation*. Dabei stehen die einzelnen Buchstaben in **SPJU-AGB** (eigentlich SPJU-AGGGB) für die verwirklichten algebraischen Operationen

- S: Selektion
- P: Projektion
- J: Natürlicher Verbund
- U: Vereinigung (siehe Unterabschnitt 3.2.1)
- AGG: Aggregation (siehe Unterabschnitt 3.2.2)
- GB: Gruppierung (siehe Unterabschnitt 3.2.3)

Die Autoren beschäftigen sich in ihrem Artikel *Provenance for Aggregate Queries* zudem mit verschachtelten Aggregationsanfragen (engl. nested aggregation queries) sowie mit der Mengenoperation Differenz. Diese können bei Interesse in [ADT11] nachgelesen werden, sollen hier aber nicht weiter vertieft werden.

3.3. CHASE-Varianten

In diesem Abschnitt sollen verschiedene CHASE-Varianten, wie der CHASE für s-t tgds und egds (siehe Unterabschnitt 3.3.3) sowie der CHASE&BACKCHASE (siehe Unterabschnitt 3.3.4) vorgestellt werden. Diese können analog zum CHASE für Tableaus aus Unterabschnitt 2.6.2 definiert und teilweise ineinander überführt werden. So können Tableauanfragen als s-t tgds sowie egds und umgekehrt formuliert werden (siehe Unterabschnitt 3.3.2).

3.3.1. Ein Überblick

Der CHASE-Algorithmus ist ein in der Datenbanktheorie universell einsetzbares Werkzeug. Er dient unter Anderem der Darstellung von Abhängigkeiten in Datenbanken oder relationenalgebraischen Ausdrücken, der Implikation von Abhängigkeiten, dem Äquivalenznachweis von Datenbankschemata unter gegebenen Abhängigkeiten sowie der Behandlung von Nullwerten in Datenbanken.

Die Idee dieses Algorithmus' kann dabei in wenigen Worten zusammengefasst werden: Für ein Objekt ◯ (beispielsweise ein Tableau T aus Unterabschnitt 2.6.1) und eine Menge von Abhängigkeiten $\star$ (beispielsweise eine Menge von JDs und FDs) arbeitet der CHASE $\star$ in ◯ ein, sodass $\star$ implizit in ◯ enthalten ist, ◯ also $\star$ nicht verletzt. Bildhaft ergibt sich also:

$$\text{chase}_\star(\bigcirc) = \circledast.$$

	$\star$	◯	Kapitel
I.	JD, FD	Tableau	2.6.2
II.	tgd, egd	Datenbank	3.3.3
III.	JD, FD, Sichten	prädikatenlogische Formeln, SQL-Anweisungen, relationenalgebraische Ausdrücke	3.3.4

Tabelle 3.17. Überblick über einige CHASE-Varianten

In dieser Arbeit sollen drei verschiedene CHASE-Varianten untersucht werden: Der CHASE auf Tableaus (I.), für Datenbanken (II.) sowie der CHASE&BACKCHASE für *Rewriting Queries using Views* (III.). Zusammengefasst in Tabelle 3.17 sollen sie zunächst kurz motiviert werden und anschließend intensiver untersucht werden.

CHASE für Tableaus:

Tableaus dienen unter Anderem der Darstellung von Datenbanken, der Untersuchung von Abhängigkeiten, zur Optimierung relationenalgebraischer Ausdrücke sowie der Äquivalenzuntersuchung von Datenbankschemata. Der CHASE-Algorithmus ermöglicht die Untersuchung dieser Zusammenhänge. So zeigt er die Gültigkeit von Verbundabhängigkeiten und funktionalen Abhängigkeiten, die Tableau-Äquivalenzen unter Integritätsbedingungen, höhere Abhängigkeiten sowie vieles weitere.

Sei ein Tableau T gegeben, dann können für $\star$ allgemeine Integritätsbedingungen eingesetzt werden. In diesem Fall sei $\star \in \{\text{FD}, \text{JD}\}$. Dann liefert der CHASE ein neues äquivalentes Tableau T':

$$\text{chase}_\star(T) = T',$$

welches für die Untersuchung der obigen Zusammenhänge zu Rate gezogen werden kann.

CHASE auf Datenbanken:

Der CHASE auf Datenbanken wird üblicherweise für die Datenintegration oder den Datenaustausch verwendet. Hierfür wird eine Menge von tgds und egds ($\star$) — eine Erweiterung der FDs und JDs — in eine Datenbank d gechaset [FKMP03]. Bei der durch

$$\text{chase}_\star(d)$$

entstehenden Datenbank d' kann es sich um die selbe Datenbank d wie zuvor handeln; die Datenbank würde „upgegradet“ werden. So werden durch egds Tupel angeglichen — Reduzierung von Nullwerten — und durch tgds neue Tupel erzeugt. Betrachten wir zwei Datenbanken, können mittels s-t tgds Tupel auch in der Zieldatenbank erzeugt werden.

CHASE&BACKCHASE:

Seien eine Anfrage Q in Form einer prädikatenlogischen Formel, einer SQL-Anfrage oder eine Anfrage der relationalen Algebra sowie eine Menge von Integritätsbedingungen wie Verbundabhängigkeiten oder funktionale Abhängigkeiten ($\star$) gegeben. Dann liefert der CHASE-Algorithmus eine äquivalente, aber unter den gegebenen Integritätsbedingungen „effizientere“ Anfrage Q'. Formal:

$$\text{chase}_\star(Q) = Q'.$$

Werden für $\star$ auch Sichten V zugelassen ergibt sich das *Answering queries using views-Problem*. Es besteht in der Suche nach effizienten Methoden zur Beantwortung einer Anfrage mietels einer Menge von Sichten über einer gegebenen Datenbank d, statt auf die Datenbankbeziehungen zuzugreifen [Hal01]. Die Idee von A. Deutsch und R- Hull besteht nun in der Erweiterung des CHASE-Algorithmus um eine zweite Phase, die BACKCHASE-Phase [DH13]. In dieser zweite Phase werden alle Teilanfragen des *Universalplans* $\mathcal{U}$ auf Äquivalenz zur Ausgangsanfrage Q geprüft, wobei $\mathcal{U} \subseteq \text{CHASE}_V(Q)$ ist. Existiert eine zu Q äquivalente Anfrage Q', so wird diese mit Hilfe dieses Vorgehens gefunden.

CHASE&BACKCHASE auf Datenbanken:

Das Ziel der vorliegenden Arbeit besteht nun in der Kombination des CHASE auf Datenbanken mit der Idee des BACKCHASE zur Provenance-Antworten-Bewertung. Hierfür werden zunächst die beiden CHASE-Verfahren vorgestellt (siehe die Unterabschnitte 3.3.3 und 3.3.4) sowie ein Zusammenhang zwischen SQL-Anfragen, relationenalgebraischen Ausdrücken, Tableaus und s-t tgds sowie egds hergestellt. In Definition 4.4 verbinden wir die beiden Theorien schließlich zu einem neuen CHASE-Algorithmus, dem *Provenance-aware* CHASE&BACKCHASE *für den Nachweis von* CHASE-*Inversen*

3.3.2. Vom Tableau zur Formel

In diesem Unterabschnitt wird der Zusammenhang zwischen SQL-Anfragen, relationenalgebraischen Ausdrücken, Tableaus sowie s-t tgds und egs vorgestellt. Es ist somit für unsere Untersuchungen irrelevant, ob eine Anfrage als SQL-Anweisung, relationenalgebraischer Ausdruck oder als Schemaabbildung über einer Menge von s-t tgds und egds gegeben ist. Auch ist die Wahl des CHASE-Algorithmus bis zu einem gewissen Punkt frei wählbar. So kann der CHASE auf Tableaus durch den CHASE nach Fagin — CHASE auf Datenbanken — ersetzt werden, denn FDs und JDs sind lediglich Spezialfälle der tgds und egds.

Anfrage 3.6 Prüfungsergebnisse der Studenten mit Vornamen 'Max' (Anfrage Q_6)

```
SELECT Matrikelnr, Modulnr, Note
FROM Studenten JOIN Noten
ON (Studenten.Matrikelnr = Noten.Matrikelnr)
WHERE Studenten.Vorname = 'Max'
```

Gegeben seien die STUDENTEN- sowie die NOTEN-Relation aus den Tabellen B.1 und B.5 im Anhang. Die hierauf definierte Anfrage 3.6 liefert eine Ergebnistabelle mit fünf Einträgen: drei Module für den Studenten mit der Matrikelnummer 3 und zwei Einträge für den Studenten mit der Matrikelnummer 7 (siehe Tabelle 3.18).

Q_6 :

Matrikelnr	Modulnr	Note
3	002	2.3
3	004	1.3
3	007	1.7
7	002	3.3
7	005	4.0

Tabelle 3.18. Ergebnistabelle der Anfrage 3.6

Anhand dieses Beispiels sollen nun das mit *tags* versehene Tableau aufgestellt und anschließend als s-t tgd geschrieben werden. Eine *s-t tgd* beschreibt dabei eine Abhängigkeit zwischen einem Quell- (engl. source) zu einem Zielschema (engl. target). Eine solche Abhängigkeit innerhalb eines Schemas heißt *tgd*. Siehe hierzu die Definitionen 3.13 und 3.12.

Zunächst werden die beiden Relationen sowie die benötigten Attribute wie folgt kodiert:

- Sie r die Universalrelation, die aus dem JOIN der STUDENTEN- und NOTEN-Relationen über das Attribut Matrikelnummer entsteht.
- Seien $r_1 = \pi_{A_1A_2A_3A_4}(r)$ und $r_2 = \pi_{A_1A_5A_6A_7}(r)$ die Teilrelationen, die der STUDENTEN- bzw. der NOTEN-Relation.
- Die Attribute von r werden wie folgt definiert:
  - A_1 = Matrikelnr,
  - A_2 = Name,
  - A_3 = Vorname,
  - A_4 = Studiengang,
  - A_5 = Modulnr,
  - A_6 = Semester,
  - A_7 = Note.

Schreiben wir die obige SQL-Anfrage 3.6 als relationenalgebraischen Ausdruck

$$\pi_{A_1}(\sigma_{A_3=\text{'Max'}}(r_1)) \bowtie \pi_{A_1A_5A_7}(r_2),$$

ergibt sich das unten stehende Tableau 3.21. Es besteht aus den beiden „Teiltableaus“ $\pi_{A_1}(\sigma_{A_3=\text{'Max'}}(r_1))$ (siehe Tabelle 3.19) und $\pi_{A_1A_5A_7}(r_2)$ (siehe Tabelle 3.20), welche über das Attribut A_1 verbunden sind. Es ergeben sich also:

- 1. Teiltableau:

$\pi_{A_1}(\sigma_{A_3=\text{'Max'}}(r_1))$:

A_1	A_2	A_3	A_4
a_1			
a_1	b_1	'Max'	b_2

Tabelle 3.19. Zwischenergebnis der Tableaukonstruktion

- 2. Teiltableau:

$\pi_{A_1A_5A_7}(r_2)$:

A_1	A_5	A_6	A_7
a_1	a_5		a_7
a_1	a_5	b_3	a_7

Tabelle 3.20. Zwischenergebnis der Tableaukonstruktion

- Ergebnistableau:

T :	A_1	A_2	A_3	A_4	A_5	A_6	A_7	
	a_1				a_5		a_7	$\text{NOTEN}_{\text{Max}}$
	a_1	b_1	'Max'	b_2				STUDENTEN
	a_1				a_5	b_3	a_7	NOTEN

Tabelle 3.21. Codierung der Anfrage 3.6 als Tableau

Die s-t tgd kann nun aus dem Ergebnistableau 3.21 abgelesen werden. Hierbei entspricht jede Zeile des Tableaus einer konkreten Relation. Also $\text{STUDENTEN}(a_1, b_1, 'Max', b_2)$ und $\text{NOTEN}(a_5, a_1, b_3, a_7)$. Die einzelnen Relationen werden dann mittels Konjunktion miteinander verknüpft. Die oberste Zeile des Tableaus entspricht der Ergebnisrelation. In diesem Beispiel wäre dies $\text{NOTEN}_{\text{Max}}(a_1, a_5, a_7)$, sodass sich insgesamt die s-t tgd

$$\text{STUDENTEN}\ (a_1, b_1, \text{Max}, b_2) \wedge \text{NOTEN}\ (a_5, a_1, b_3, a_7) \rightarrow \text{NOTEN}_{\text{Max}}\ (a_1, a_5, a_7)$$

ergibt.

Bemerkung. Da das Ergebnis bei s-t tgds eventuell mehrere Ergebniszeilen umfasst, wird das Tableau häufig gedreht, sodass die Ergebnistupel unterhalb des Tableaus stehen. Für die Anfrage 3.6 ergibt sich so das Tableau aus Tabelle 3.22.

T :	A_1	A_2	A_3	A_4	A_5	A_6	A_7	
	a_1	b_1	'Max'	b_2				STUDENTEN
	a_1				a_5	b_3	a_7	NOTEN
	a_1				a_5		a_7	$\text{NOTEN}_{\text{Max}}$

Tabelle 3.22. Codierung der Anfrage 3.6 als Tableau für s-t tgds

3.3.3. CHASE für s-t tgds und egds

Der CHASE-Algorithmus kann nicht nur für Tableaus sondern auch für s-t tgds und egds definiert werden. Dies beschreiben unter anderem Fagin et al. in ihren Artikeln *Schema Mapping Evolution through Composition and Inversion* [FKPT11] und *Data Exchange: Semantics and Query Answering* [FKMP03]. Dabei werden stets ein Quellschema S sowie ein Zielschema S' betrachtet.

Die hier vorgestellte CHASE-Variante dient der Angabe einer natürlichen, operativen Semantik für den Datenaustausch. Der Algorithmus aus Definition 3.20 eignet sich insbesondere für eine feste Schemaabbildung, die durch s-t tgds oder egds spezifiziert ist. Er berechnet zu einer gegebenen Quellinstanz I mittels $\text{chase}_M(I)$ eine Zielinstanz I', die den geforderten Abhängigkeiten Σ genügt.

Definition 3.12 (tuple-generating dependency (tgd))**.** Eine **tuple-generating dependency** (**tgd**) ist eine Formel der Form

$$\forall x : (\varphi(x) \rightarrow \exists y : \psi(x, y))$$

mit Variablentupeln x und y, Konjunktionen $\varphi(x)$ von atomaren Formeln mit Variablen aus x sowie Konjunktionen $\psi(x, y)$ mit Variablen aus x und y. ⊡

Definition 3.13 (source-to-target tuple-generating dependency (s-t tgd))**.** Eine **source-to-target tuple-generating dependency** (**s-t tgd**) ist eine Sequenz der Form

$$\forall x : (\varphi(x) \rightarrow \exists y : \psi(x, y)).$$

Dabei ist $\varphi(x)$ eine Konjunktion von Atomen über S, d.h. Ausdrücken der Form $\mathcal{R}(x_1, ..., x_n)$, und $\psi(x,y)$ eine Konjunktion von Atomen über S'. ⊡

Bemerkung. S-t tgds sind mit ihren Einschränkungen der Konjunktion von Prämisse und Ergebnis somit ebenso wie die Verbundabhängigkeiten aus Unterabschnitt 2.1.2 ein Spezialfall der tgds.

Standardmäßig werden drei Typen von Abbildungen unterschieden: GAV, LAV und GLAV (siehe unten). Nach ihrer Definition wird für die Aufstellung des CHASE für prädikatenlogische Formeln nur noch die Definition der equality-generating dependency benötigt.

Definition 3.14 (global-as-view-Einschränkung (GAV))**.** Eine **global-as-view-Einschränkung** (GAV) ist eine s-t tgd, die als Ergebnis ein einzelnes Atom ohne zusätzliche Existenzquantoren liefert. Formal heißt dies:

$$\forall x : (\varphi(x) \rightarrow P(x)),$$

wobei $P(x)$ ein Atom über dem Zielschema S' ist. ⊡

Definition 3.15 (local-as-view-Einschränkung (LAV))**.** Die **local-as-view-Einschränkung** (LAV) ist ebenfalls eine s-t tgd. Sie verarbeitet die Eingabe eines einzelnen Atoms $Q(x)$ über dem Quellschema S, d.h.

$$\forall x : (Q(x) \rightarrow \exists y : \psi(x, y)).$$

⊡

Bemerkung. LAV und GAV bilden somit eine spezielle Klasse der **global-and-local-as-view** (global-and-local-as-view (GLAV)), wie die s-t tgds auch von einigen Autoren genannt werden.

Definition 3.16 (equality-generating dependency (egd))**.** Eine **equality-generating dependency** (egd) ist für zwei Variablen x_1 und x_2 aus x definiert durch

$$\forall x : (\varphi(x) \rightarrow (x_1 = x_2)).$$

⊡

In der Datenbank müssen zwei verschiedene „Variablenmengen“ unterschieden werden: Const beschreibt die **Konstanten** — die endliche Menge der Quellinstanzvariablen; Var hingegen die unendliche Menge der existenzquantorisierten Variablen, den sogenannten **(markierten) Nullvariablen**. Dabei gilt Const $\cap$ Var $= \emptyset$.

Definition 3.17 (Homomorphismus)**.** Seien I_1 und I_2 zwei Instanzen über einem Datenbankschema $S = \{R_1, ..., R_n\}$ mit Variablen aus Const $\cup$ Var. Dann ist ein Homomorphismus $h : I_1 \rightarrow I_2$ eine Abbildung von Const $\cup$ Var(I_1) nach Const $\cup$ Var(I_2), so dass:

- $\forall c \in$ Const $: h(c) = c$
- $\forall t \in I_1(\mathcal{R}_i(A_1, ..., A_k)) : h(t) \in I_2(\mathcal{R}_i(h(A_1, ..., A_k)))$

Existiert zudem ein zweiter Homomorphismus $h' : I_2 \rightarrow I_1$, heißt I_1 **äquivalent zu** I_2. ⊡

Für die Definition des CHASE werden neben den Definitionen von source-to-target tuple-generating dependencies und equality-generating dependencies analog zum Tableau-CHASE aus Unterabschnitt 2.6.2 zwei Regeln zur Verarbeitung dieser Abhängigkeiten benötigt. Die tgd-Regel erzeugt zusätzliche Tupel; sie entspricht somit der Ersetzung der Quelldatenbank (Nullvariablen) gemäß der tgd (Anfrage) durch entsprechende Attributwerte. Die egd-Regel hingegen ersetzt alle markierten Nullvariablen durch entsprechende Konstanten oder markierte Nullvariablen mit kleinerem Index.

Definition 3.18 (tgd-Regel)**.** Gegeben sei eine Instanz I. Seien weiterhin δ eine s-t tgd $\varphi(x) \rightarrow \exists y : \psi(x, y)$ und $h : \varphi(x) \rightarrow I$ ein Homomorphismus. Dann gilt:

- Der CHASE-Schritt von I unter δ kann nicht angewendet werden, wenn h zu $h' : \varphi(x) \wedge \psi(x, y) \rightarrow I$ erweitert werden kann.
- Ist der CHASE-Schritt anwendbar, bilde I' wie folgt: Erweitere h auf h', sodass jede Variable in y (markierte) Nullvariable ist. Dann gilt $I' = I \cup h'(\psi(x, y))$. Schreibweise: $I \overset{\delta,h}{\rightarrow} I'$.

$\boxdot$

Definition 3.19 (egd-Regel)**.** Gegeben seien eine egd $\delta : \varphi(x) \rightarrow (x_1 = x_2)$ und ein Homomorphismus $h : \varphi(x) \rightarrow I$ mit $h(x_1) \neq h(x_2)$.

- $h(x_1), h(x_2) \in \underline{\text{Const}}$: Das Ergebnis der Anwendung von d mittels h liefert kein Ergebnis. Geschrieben wird dieser Sachverhalt als $I \xrightarrow{\delta,h} \perp$.
- $h(x_i) \in \underline{\text{Const}} \wedge h(x_j) \in \underline{\text{Var}}$ mit $i \neq j$ und $i, j \in \{1, 2\}$: Ersetze die (markierte) Nullvariable $h(x_j)$ durch die Konstante $h(x_i)$.
- $h(x_1), h(x_2) \in \underline{\text{Var}}$: Setzte $h(x_1)$ und $h(x_2)$ auf die selbe markierte Nullvariable $h(x_1)$ oder $h(x_2)$.

In den Fällen zwei und drei ergibt sich somit eine neue Instanz I' durch Anwendung der egd δ auf die Instanz I mit Hilfe des Homomorphismus h. Wir schreiben dies als $I \xrightarrow{\delta,h} I'$. $\boxdot$

Bemerkung. Die tgd-Regel (Definition 3.18) sowie die egd-Regel (Definition 3.19) sind Erweiterungen der klassischen J-Regel (Definition 2.22) sowie der F-Regel (Definition 2.21) für Tableaus aus dem Grundlagenkapitel. Sie werden auch **CHASE-Schritt** genannt.

Der folgende CHASE-Algorithmus ist „lediglich“ eine Umformulierung des Algorithmus aus Definition 2.23. Er kombiniert die obigen CHASE-Schritte zu einer CHASE-Sequenz und endet in einem scheiternden oder erfolgreichen CHASE.

Definition 3.20 (CHASE)**.** Gegeben seien eine Instanz I sowie eine Menge Σ von tgds und edgs.

- Eine **CHASE-Sequenz** von I über Σ ist eine (endliche oder unendliche) Sequenz $I_i \xrightarrow{\delta_i,h_i} I_{i+1}$ $(i = 0, 1, ...)$ der tgd- und egd-Regel aus den Sätzen 3.18 sowie 3.19. Dabei ist $I = I_0$ und d_i eine Abhängigkeit in Σ.
- Der **CHASE** von I über Σ ist eine endliche CHASE-Sequenz $I_i \xrightarrow{\delta,h} I_{i+1}$, $0 \leq i < n$ mit
  - $I_n = \perp$ (**scheiternder CHASE**);

  ODER
  - Es existiert keine Abhängigkeit δ_i aus Σ und kein Homomorphismus h_i, so dass δ_i mit h_i auf I_n angewandt werden kann. Dann ist I_n das Ergebnis des endlichen CHASE (**erfolgreicher CHASE**).

$\boxdot$

Der CHASE-Algorithmus liefert somit eine **Universallösung** von I unter der Verwendung der Schemaabbildung $M = (S, S', \Sigma)$ — vgl. Definition 2.16 — und ist für spezielle Σ in polynomieller Zeit berechenbar (etwa für eine Menge von schwachen azyklischen tgds und egds) [FKMP03]. Universallösungen sind die allgemeinsten Lösungen eines Quellschemas bzgl. einer Schemaabbildung. Formal heißt dies:

Definition 3.21 (Lösung, Universallösung)**.** Gegeben seien eine Schemaabbildung (S, S', Σ) sowie eine Quellinstanz I.

- Die Zielinstanz J heißt **Lösung** von I bzgl. der Schemaabbildung$\mathcal{M}$, wenn das Paar (I, J) in $\mathcal{M}$ enthalten ist.
- Weiter heißt J **Universallösung** für I, wenn für jede Lösung J' von I ein Homomorphismus $h : J \to J'$ existiert. ⊡

Zum Abschluss dieses Abschnitts soll die CHASE-Variante für s-t tgds und egds anhand der Anfrage 3.6 der STUDENTEN- und NOTEN-Tabelle erläutert werden. Die Instanz I sei hierbei auf die nötigsten Tupel der beiden Relationen beschränkt.

Beispiel 3.8. *Gegeben seine eine* GAV-*Schemaabbilung* $\mathcal{M}$ *mit*

$$\begin{aligned} S &= \textit{Quellschema mit den beiden Relationen } \text{STUDENTEN } \textit{und } \text{NOTEN}, \\ S' &= \textit{Zielschema mit der Relation } \text{NOTEN}_{\text{Max}}, \\ \Sigma &= \{\text{STUDENTEN } (a_1, a_2, a_3, a_4) \wedge \text{NOTEN } (a_5, a_1, a_6, a_7) \to \text{NOTEN}_{\text{Max}}(a_1, a_5, a_7)\} \end{aligned}$$

und eine Instanz

$$\begin{aligned} I = \{&\text{STUDENTEN } (3, \text{Müller}, \text{Max}, \text{Elektrotechnik}), \text{NOTEN } (002, 3, \text{WS } 14/15, 2.3), \\ &\text{STUDENTEN } (3, \text{Müller}, \text{Max}, \text{Elektrotechnik}), \text{NOTEN } (004, 3, \text{WS } 16/17, 1.3), \\ &\text{STUDENTEN } (3, \text{Müller}, \text{Max}, \text{Elektrotechnik}), \text{NOTEN } (007, 3, \text{SS } 17, 1.7), \\ &\text{STUDENTEN } (7, \text{Mustermann}, \text{Max}, \text{Elektrotechnik}), \text{NOTEN } (002, 7, \text{WS } 15/16, 3.3), \\ &\text{STUDENTEN } (7, \text{Mustermann}, \text{Max}, \text{Elektrotechnik}), \text{NOTEN } (005, 7, \text{SS } 17, 1.7)\}. \end{aligned}$$

Dann liefert die Anwendung des CHASE*, d.h.* $chase_M(I)$*, die Instanztupel:*

$$\begin{aligned} &\text{NOTEN}_{\text{Max}}(3, 002, 2.3), \\ &\text{NOTEN}_{\text{Max}}(3, 004, 1.3), \\ &\text{NOTEN}_{\text{Max}}(3, 007, 1.7), \\ &\text{NOTEN}_{\text{Max}}(7, 002, 3.3), \\ &\text{NOTEN}_{\text{Max}}(7, 005, 1.7). \end{aligned}$$

Der CHASE kann somit Anfragen darstellen und auf Datenbanken angewendet werden. Unser nächstes Ziel besteht darin, den CHASE-Algorithmus mit den Provenance-Anfragen aus den Abschnitten 3.1 und 3.2 zu verbinden. Hierfür benötigen wir eine zweite Phase, die diese zusätzlichen Informationen verarbeiten kann. Der CHASE&BACKHCASE nach A. Deutsch und R. Hull beinhaltet bereits eine zweite Phase, welche den Provenance-Aspekt enthält. Im Kapitel 4 wollen wir schließlich eine eigene BACHCHASE-Phase für den CHASE für Datenbanken definieren.

3.3.4. CHASE&BACKCHASE

A. Deutsch und R. Hull beschreiben in ihrem Artikel *Provenance-Directed Chase&Backchase* [DH13] die Anwendung des CHASE im Falle der Umformulierung von Anfragen mittels Sichten (engl. answering queries using views, kurz RQnV). Dieser basiert auf dem Aufbau einer Menge von kanonische „Umschreibungskandidaten“, deren exponentiell vielen Teilanfragen anschließend mit Hilfe des CHASE-Algorithmus auf Äquivalenz zur Ausgangsanfrage überprüft werden (BACKCHASE).

AQuV sind zwar nicht Bestandteil dieser Arbeit, doch beschreiben die Autoren einen interessanten Zusammenhang zwischen CHASE und der Provenance-Theorie. So kann die BACKCHASE-Phase durch das „Merken“ von Provenance-Informationen deutlich beschleunigt werden.

Definition 3.22 (CHASE&BACKCHASE)**.** Der **CHASE**&**BACKCHASE** besteht aus zwei Phasen:

- **CHASE**: Gegeben seine eine Anfrage Q, eine Menge von Sichten V und Integritätsbedingungen $\mathcal{B}$. Die Anwendung des CHASE liefert eine Menge $Q' = \text{chase}_{V \cup \mathcal{B}}(Q)$, die eingeschränkt auf die Sichten V den so genannten **Universalplan** $\mathcal{U} = Q' \mid_V$ definiert.
- **BACKCHASE**: Die Teilanfragen (engl. subqueries) von $\mathcal{U}$ werden auf Äquivalenz (bzgl. $\mathcal{B}$ und V) zur Anfrage Q untersucht und anschließend alle bzgl. $V \cup \mathcal{B}$ minimalen Teilanfragen ausgegeben.

□

Während in der CHASE-Phase zunächst der Universalplan $\mathcal{U}$ durch „chasen“ der Anfrage Q bestimmt wird, werden in der BACKCHASE-Phase alle Teilmengen von $\mathcal{U}$ „gechaset“, d. h. bilde $\text{chase}_{V \cup B}(\mathcal{U}_i)$ für alle $\mathcal{U}_i \subseteq U$. Dieses Vorgehen ist als Backtracking-Verfahren sehr langsam; es müssen exponentiell viele Teilanfragen untersucht werden. Auch ist zu Beginn der BACKCHASE-Phase nicht klar, ob eine zu Q äquivalente Anfrage überhaupt existiert.

Das folgende Beispiel 3.9 soll die Vorgehensweise des CHASE&BACKCHASE-Verfahrens exemplarisch erläutern. Die Grundlage bildet hierfür wiederum das Studenten-Noten-Beispiel aus Abschnitt 1.2. Wegen der Formulierung des Algorithmus für AQuV, werden jedoch zusätzliche Sichtdefinitionen benötigt; die Integritätsbedingungen ergeben sich aus der Anfrage 3.6. Das Beispiel ist zur besseren Vergleichbarkeit mit dem zugrundeliegenden Artikel von A. Deutsch und F. Hull zudem in der Datalog-Notation gehalten.

Datalog ist eine Datenbank-Programmiersprache für deduktive Datenbanken, die der Sprache PROLOG syntaktisch und semantisch ähnelt. In deduktiven Datenbanken werden Konzepte der Logikprogrammierung mit den Konzepten von Datenbankmanagementsystemen verbunden. Sie nutzen keine Regel- und Faktenliste, sondern Mengen, und ermöglicht daher die Regelverarbeitung mittels mengenbasierten Anfragen. Eine auf Horn-Klauseln der Form

$$P_1 \wedge ... \wedge P_n \Rightarrow P$$

eingeschränkte Formel wird in PROLOG syntaktisch als

$$P :- P_1, ..., P_n$$

geschrieben. Diese kurze Einführung soll für unsere Zwecke hier reichen. Für genauere Erläuterungen bzgl. Datalog und PROLOG siehe auch [SSH13].

Beispiel 3.9. *Gegeben seien eine Anfrage Q, eine Menge von Sichten V sowie eine Menge von Integritätsbedingungen $\mathcal{B}$ wie folgt:*

- *Anfrage:* $Q(a_1, a_5, a_7) :- S(a_1, a_2, a_3, a_4), N(a_5, a_1, a_6, a_7)$
- *Sichten V:*
  - $V_S(a_1) :- S(a_1, a_2, a_3, a_4)$
  - $V_N(a_5, a_1, a_7) :- N(a_5, a_1, a_6, a_7)$
  - $V_{SN}(a_1, a_5, a_7) :- S(a_1, a_2, a_3, a_4), N(a_5, a_1, a_6, a_7)$

- *Integritätsbedingungen* $\mathcal{B} = \{N(a_5, a_1, a_6, a_7) \rightarrow \exists\ a_2, a_3, a_4 : S(a_1, a_2, a_3, a_4)\}$
- CHASE:
  - $chase_{V \cup B}(Q) :- S(a_1, a_2, a_3, a_4), N(a_1, a_5, a_6, a_7), V_S(a_1), V_N(a_5, a_1, a_7), V_{SN}(a_1, a_5, a_7)$
  - *Universalplan:* $\mathcal{U} :- V_S(a_1), V_N(a_5, a_1, a_7), V_{SN}(a_1, a_5, a_7)$
- BACKCHASE:
  - *Berechne die Potenzmenge des Univeralplans* $\mathcal{U}$*:*

$$\mathscr{P}(\mathcal{U}) = \{\emptyset, \{V_S\}, \{V_N\}, \{V_{SN}\}, \{V_S, V_N\}, \{V_S, V_{SN}\}, \{V_N, V_{SN}\}, \{V_S, V_N, V_{SN}\}\}$$

  - *Berechne* $\text{chase}_{V \cup \mathcal{B}}(\mathcal{U}_i)$ *für alle* $\mathcal{U}_i \in \mathscr{P}(\mathcal{U})$ *und* $\mathcal{U}_i \neq \emptyset$*:*
    - $\text{chase}_{V \cup \mathcal{B}}(V_S) :- V_S(a_1),\ S(a_1, a_2, a_3, a_4)$
    - $\text{chase}_{V \cup \mathcal{B}}(V_N) :- V_N(a_5, a_1, a_7),\ N(a_5, a_1, a_6, a_7)$
    - $\text{chase}_{V \cup \mathcal{B}}(V_{SN}) :- V_{SN}(a_1, a_5, a_7),\ S(a_1, a_2, a_3, a_4),\ N(a_5, a_1, a_6, a_7)$
    - $\text{chase}_{V \cup \mathcal{B}}(V_S, V_N) :- V_S(a_1),\ V_N(a_5, a_1, a_7),\ S(a_1, a_2, a_3, a_4),\ N(a_5, a_1, a_6, a_7)$
    - $\text{chase}_{V \cup \mathcal{B}}(V_S, V_{SN}) :- V_S(a_1),\ V_{\text{Max}}(a_1, a_5, a_7),\ S(a_1, a_2, a_3, a_4),\ S(a_1, a_2, a_3, a_4),\ N(a_5, a_1, a_6, a_7)$
    - $\text{chase}_{V \cup \mathcal{B}}(V_N, V_{SN}) :- V_N(a_5, a_1, a_7),\ V_{\text{Max}}(a_1, a_5, a_7),\ N(a_5, a_1, a_6, a_7),\ S(a_1, a_2, a_3, a_4),\ N(a_5, a_1, a_6, a_7)$
    - $\text{chase}_{V \cup \mathcal{B}}(V_S, V_N, V_{SN}) :- V_S(a_1),\ V_N(a_5, a_1, a_7),\ V_{SN}(a_1, a_5, a_7),\ S(a_1, a_2, a_3, a_4),\ N(a_5, a_1, a_6, a_7),\ S(a_1, a_2, a_3, a_4),\ N(a_5, a_1, a_6, a_7)$
  - *Wähle die zur Anfrage* Q *äquivalenten bzgl.* $V \cup \mathcal{B}$ *minimalen Anfrage (rot markiert):*
    - $Q' :- V_S(a_1), V_N(a_5, a_1, a_7)$
    - $Q'' :- V_{SN}(a_1, a_5, a_7)$

      Die Minimalität ergibt sich dabei wie folgt: Werden V_S *oder* V_N *aus* Q' *oder* V_{SN} *aus* Q'' *gestrichen, so sind diesen nicht mehr äquivalent zur Anfrage* Q*.*
    - $Q''' :- V_S(a_1), V_N(a_5, a_1, a_7), V_{SN}(a_1, a_5, a_7)$

      Die Anfrage Q''' *ist ebenfalls zu* Q *äquivalent, aber nicht minimal. Hier können* V_S *und* V_N *oder* V_{SN} *gestrichen werden, ohne die Äquivalenzbeziehung zu verletzten.*

Es ergeben sich nach den obigen Untersuchungen also zwei zu Q *äquivalente Anfragen:* Q' *und* Q''*.*

Zur Beschleunigung des Verfahrens können zunächst einige Vorüberlegungen getroffen werden. Zum einen kann vor Beginn des Verfahrens die Existenz einer zu Q äquivalenten Anfrage geprüft werden. Gibt es eine solche Anfrage nicht, muss der CHASE&BACKCHASE-Algorithmus nicht ausgeführt werden. Bei der Durchführung der BACKCHASE-Phase können zudem einige Teilmengen von $\mathcal{U}$ vernachlässigt werden:

- Alle Oberanfragen (engl. superqueries) von Q sowie allen zu Q äquivalenten Anfragen, die bereits bekannt sind.

- Alle Teilanfragen, deren Variablen nicht im Anfrageergebnis enthalten sind.

So könnte in Beispiel 3.9 im Falle der Sichtdefinition $V_S(a_2, a_3) : -S(a_1, a_2, a_3, a_4)$ das Chasen der Teilmenge $\{V_S\} \subseteq \mathcal{U}$ mit den Sichten V und Integritätsbedingungen B vernachlässigt werden. Merkt man sich zusätzlich die Provenance-Informationen, können die äquivalenten Ergebnisse direkt aus $\text{chase}_{V \cup \mathcal{B}}(\mathcal{U})$ abgelesen werden. Das Chasen aller Teilanfragen von $\mathcal{U}$ wird somit überflüssig. Formal kann dieser provenance-basierte Ansatz wie folgt zusammengefasst werden:

Satz 3.4 (Provenance-aware CHASE)**.** *Sei $\mathcal{U}$ ein Universalplan und π eine Provenance-Formel.*

- *Für jedes Atom $a \in U$ gilt: $\pi(a) = a$.*

- *Sei I eine Instanz und t eine tgd der Form $\varphi(x) \rightarrow \exists y : \psi(x, y)$, wobei φ und ψ Konjunktionen von relationalen Atomen und x, y Vektoren von Variablen sind. Weiterhin sei h ein Homomorphismus von $\varphi(x)$ nach I.*

  – *Der* CHASE*-Schritt von I unter t scheitert, wenn es eine Erweiterung auf $h' : \psi(x, y) \rightarrow I$ von h gibt, so dass $\pi(h(\varphi(x)) \Rightarrow \pi(h'(\psi(x))$.*

  – *Kann der* CHASE*-Schritt durchgeführt werden, ergibt sich ρ' analog zum Standard-*CHASE *(vgl. Satz 3.18) durch Addition von neuen Atomen mit $\pi(h(\varphi(x))$. Enthält ρ bereits ein Atom a mit der Provenance-Information p_1 und der* CHASE*-Schritt liefert dasselbe Atom a mit der Provenance-Information p_2, wird in ρ' lediglich eine Kopie von a mit der Provenance-Information $p_1 \vee p_2$ gesetzt.*

Definition 3.23 (Provenance-aware CHASE&BACKCHASE)**.** Der **provenance-aware CHASE&BACKCHASE** kombiniert die Provenance-Theorie mit der Idee des CHASE&BACKCHASE. Seien dazu eine Anfrage Q, eine Menge von Sichten V sowie eine Menge von Integritätsbedingungen $\mathcal{B}$ gegeben. Dann gilt:

- **CHASE**: Berechne den Universalplan $\mathcal{U}$ analog zur Definition 3.22.

- **Provenance-directed BACKCHASE**:

  1. Berechne U' durch provenance-aware CHASE von $\mathcal{U}$ mit $V \cup \mathcal{B}$.

  2. Berechne die minimale ***why***-Provenance der Tupel von U' entsprechend der Variablen in Q.

  3. Ausgabe der Teilanfragen von $\mathcal{U}$, die durch die Provenance definiert sind.

□

Beispiel 3.10. *Der provenance-aware* CHASE&BACKCHASE*-Algorithmus liefert für das obige Beispiel 3.9 dieselbe* CHASE*-, aber eine erweiterte* BACKCHASE*-Phase (Provenance-Informationen an den geschweiften Klammern vermerkt):*

- CHASE:

  – $\text{chase}_{V \cup \mathcal{B}}(Q) :- S(a_1, a_2, a_3, a_4), N(a_1, a_5, a_6, a_7), V_S(a_1), V_N(a_5, a_1, a_7), V_{SN}(a_1, a_5, a_7)$

  – *Universalplan:* $\mathcal{U} :- V_S(a_1), V_N(a_5, a_1, a_7), V_{SN}(a_1, a_5, a_7)$

- BACKCHASE:
  - $\text{chase}_{V \cup \mathcal{B}}(\mathcal{U}) :- \underbrace{V_S(a_1)}_{V_S}, \underbrace{S(a_1, a_2, a_3, a_4)}_{V_S}, \underbrace{V_N(a_1, a_5, a_7)}_{V_N}, \underbrace{N(a_5, a_1, a_6, a_7)}_{V_N}, \underbrace{V_{SN}(a_1, a_5, a_7)}_{V_{SN}}, \underbrace{S(a_1, a_2, a_3, a_4)}_{V_{SN}}, \underbrace{N(a_5, a_1, a_6, a_7)}_{V_{SN}}$
  - *Äquivalente Anfragen (zugehörige Provenance-Informationen rot markiert):*
    - $Q' :- V_S(a_1), V_N(a_5, a_1, a_7)$
    - $Q'' :- V_{SN}(a_1, a_5, a_7)$

    Es ergeben sich somit zwei zu Q äquivalente Anfragen Q' und Q''. Grund hierfür ist, dass der CHASE *des Universalplans U die ursprüngliche Anfrage Q als Teilmenge enthält. Sie können mit Hilfe der Provenance-Informationen (rot markiert) also einfach „abgelesen“ werden.*

Als nächstes wollen wir uns mit der Frage beschäftigen, was wir in den BACKCHASE aufnehmen müssen, um Provenance-Anfragen aufzustellen. Die Idee besteht in der Verwendung sogenannter inverser Schemaabbildungen. In unserem Fall speziell die sogenannten CHASE-inversen Schemaabbildungen.

3.4. Inverse Schemaabbildung

In diesem Abschnitt sollen verschiedene inverse Abbildungen vorgestellt werden. Diese sind zum einen die allgemeine *inverse Schemaabbildung* nach [Fag07], die *Quasi-Inverse* nach [FKPT08], die CHASE-*Inverse* und die *exakte* sowie die *relaxte* CHASE-*Inverse* nach [FKPT11]. Die letzteren drei Inversen basieren dabei auf dem CHASE-Algorithmus für s-t tgds und egds aus Unterabschnitt 3.3.3 und können wie folgt angeordnet werden: relaxte CHASE-Inverse $\preceq$ CHASE-Inverse $\preceq$ exakte CHASE-Inverse. Während bei der exakten CHASE-Inverse auf Identität geprüft wird, genügt bei der CHASE-Inversen eine Äquivalenzbeziehung. Bei der relaxten CHASE-Inversen wird diese Bedingung noch weiter aufgeweicht. Als weitere Abschwächung definieren wir im vierten Kapitel noch eine eigene CHASE-inverse Abbildung, die *ergebnisäquivalente* CHASE-*Inverse*.

3.4.1. Allgemeine inverse Schemaabbildungen

Die Definition einer (exakten oder relaxten) CHASE-Inversen ist eine spezielle Form einer inversen Abbildung. Erstmalig formuliert wurde eine inverse Schemaabbildung im Jahre 2007 von R. Fagin [Fag07]. Er führte die *Inverse* $\mathcal{M}^\star$ einer Schemaabbildung $\mathcal{M}$ als eine Schemaabbildung mit $\mathcal{M} \circ \mathcal{M}^\star = \text{Id}$ ein. Dabei entspricht Id der Indentitätsabbildung, die jede Instanz auf sich selbst abbildet. Vergleiche für die Definition der Schemaabbildung auch die Definition 2.16.

Definition 3.24 (Inverse Schemaabbildung)**.** Seien $\mathcal{M}_{12} = (S, S', \Sigma_{12})$ und $\mathcal{M}_{21} = (S', S'', \Sigma_{21})$ zwei Schemaabbildungen. Sei $\delta = \Sigma_{21} \circ \Sigma_{12}$ die Komposition der Abbildungen $\mathcal{M}_{21}$ und $\mathcal{M}_{12}$ sowie $\mathcal{M}_{11} = (S, S'', \delta)$. Für die so definierten Schemaabbildungen ergibt sich also folgendes Diagramm:

$$S \xrightarrow{\mathcal{M}_{12}} S' \xrightarrow{\mathcal{M}_{21}} S'' \qquad S \xrightarrow{\mathcal{M}_{11}} S''$$

Sei weiter I eine Instanz von S und I'' eine Instanz von S''. Dann heißt $\mathcal{M}_{21}$ **Inverse** von $\mathcal{M}_{12}$, wenn $\mathcal{M}_{11}$ auf I äquivalent zur Identitätsabbildung $\mathcal{M}_{\mathrm{Id}}$ ist. Demnach ist $\mathcal{M}_{21}$ Inverse von $\mathcal{M}_{12}$ genau dann, wenn für jede Zielinstanz J gilt:

$$(I, J) \models \delta \text{ genau dann, wenn } I'' \subseteq J.$$

□·

Inverse Abbildungen finden beispielsweise bei der Schemaevolution Anwendung. Kann eine Inverse aber nicht angegeben bzw. definiert werden und wird dennoch eine Art inverse Abbildung benötigt, kann auf die sogenannte *Quasi-Inverse* zurückgegriffen werden [FKPT08]. Sie liefert nicht mehr die Quellinstanz selbst, dafür aber eine hierzu äquivalente Instanz oder zumindest eine äquivalente Teilinstanz.

Definition 3.25 (Datenaustausch-Äquivalenz)**.** Sei $\mathcal{M} = (S, S', \Sigma)$ eine Schemaabbildung und I_1, I_2 zwei Quellinstanzen. Stimmen die Mengen der Lösungen für I_1 und I_2 überein, d.h. die Lösung von I_i $(i = 1, 2)$ unter $\mathcal{M}$ entspricht einer Zielinstanz J mit $(I_i, J) \models \Sigma$, so heißen I_1 und I_2 **datenaustausch-äquivalent** unter $\mathcal{M}$. Geschrieben: $I_1 \leftrightarrow_{\mathcal{M}} I_2$. □·

Definition 3.26 (Quasi-Inverse)**.** Seien $\mathcal{M} = (S, S', \Sigma)$ und $\mathcal{M}' = (S', S, \Sigma')$ zwei Schemaabbildungen. Dann heißt $\mathcal{M}'$ **Quasi-Inverse** von $\mathcal{M}$, wenn für jedes Paar von Instanzen (I_1, I_2) folgende Aussagen äquivalent sind:

- Seien I_1' und I_2' zwei Quellinstanzen mit $I_1 \leftrightarrow_{\mathcal{M}} I_1'$, $I_2 \leftrightarrow_{\mathcal{M}} I_2'$ und $I_1' \subseteq I_2'$.
- Seien I_1'' und I_2'' zwei Quellinstanzen und J eine Zielinstanz. Dann gilt

$$I_1 \leftrightarrow_{\mathcal{M}} I_1'', I_2 \leftrightarrow_{\mathcal{M}} I_2'', (I_1'', J) \models \Sigma \text{ und } (J, I_2'') \models \Sigma'.$$

□·

Beispiel 3.11. *Gegeben sei eine Schemaabbildung $\mathcal{M} = (S, S', \Sigma)$, wobei Σ definiert ist durch die tgd*

$$\mathcal{R}_1(a_1, a_2, a_3) \rightarrow \mathcal{R}_2(a_1, a_2) \wedge \mathcal{R}_3(a_2, a_3).$$

Dann sind $\mathcal{M}' = (S', S, \Sigma')$ und $\mathcal{M}'' = (S', S, \Sigma'')$ mit

$$\Sigma' = \{\mathcal{R}_2(a_1, a_2) \wedge \mathcal{R}_3(a_2, a_3) \rightarrow \mathcal{R}_1(a_1, a_2, a_3)\}$$

und

$$\begin{aligned} \Sigma'' \quad = \quad & \{\mathcal{R}_2(a_1, a_2) \rightarrow \exists a_3 : \mathcal{R}_1(a_1, a_2, a_3), \\ & \mathcal{R}_3(a_2, a_3) \rightarrow \exists a_1 : \mathcal{R}_1(a_1, a_2, a_3)\} \end{aligned}$$

zwei Quasi-Inverse von $\mathcal{M}$. Es muss also keine eindeutige Quasi-Inverse geben.

Nachweis der Quasi-Inversen $\mathcal{M}'$: Seien I_1 und I_2 zwei Quellinstanzen. Dann existieren zwei weitere Quellinstanzen I_1'' und I_2'' sowie eine Zielinstanz J mit $I_1 \leftrightarrow_{\mathcal{M}} I_1''$, $I_2 \leftrightarrow_{\mathcal{M}} I_2''$, (I_1'', J) und (J, I_2''). Dabei genügt (I_1'', J) den tgds $\mathcal{R}_1(a_1, a_2, a_3) \rightarrow \mathcal{R}_2(a_1, a_2) \wedge \mathcal{R}_3(a_2, a_3)$, sowie (J, I_2'') der Abhängigkeit $\mathcal{R}_2(a_1, a_2) \wedge \mathcal{R}_3(a_2, a_3) \rightarrow \mathcal{R}_1(a_1, a_2, a_3)$. Es folgt $\mathcal{R}_1(I_1) \subseteq \mathcal{R}_2(J) \cap \mathcal{R}_3(J) \subseteq \mathcal{R}_1(I_2'')$. Setzen wir $I_1' = I_1'' \cap I_2''$ und $I_2' = I_2''$, dann gilt $I_1 \leftrightarrow_{\mathcal{M}} I_1'$, $I_2 \leftrightarrow_{\mathcal{M}} I_2'$ und $I_1' \subseteq I_2'$.

Das von R. Fagin et al. eingeführte *Unifying Framework* [FKPT08] soll im Folgenden auf CHASE-Inverse eingeschränkt werden. Hierfür werden die CHASE-*Inverse* als Pendant zur Inversen aus Definition 3.24 sowie die *relaxte* CHASE-*Inverse* in Analogie zur Quasi-Inversen aus Definition 3.26 formuliert.

3.4.2. CHASE-inverse Schemaabbildungen

Die Definitionen der *exakten*, *relaxten* sowie der „reinen“ CHASE-*Inversen* selbst basieren auf dem Artikel *Schema Mapping Evolution through Composition and Inversion* von R. Fagin, P. G. Kolaitis, L. Popa sowie W.-C. Tan [FKPT11]. Diese wollen wir jeweils durch ein kurzes Beispiele ergänzen.

Seien S und S' zwei Schemata, Σ und Σ' zwei Mengen von Abhängigkeiten. Jede der folgenden Typ-Definitionen von CHASE-inversen Funktionen basiert auf zwei Schemaabbildungen $\mathcal{M} = (S, S', \Sigma)$ und $\mathcal{M}^\star = (S', S, \Sigma')$ sowie ihren zugehörigen Instanzen I, I' und I''. Dabei ist I stets Quellinstanz und I' das Ergebnis des CHASE-Algorithmus von I unter $\mathcal{M}$, d.h. $I' = \text{chase}_{\mathcal{M}}(I)$. Der erneute CHASE — $\text{chase}_{\mathcal{M}^\star}(I')$ — liefert dann die Urinstanz I''. Ist $\mathcal{M}^\star$ eine CHASE-inverse Abbildung von $\mathcal{M}$, so enthält I'' ganze oder aber auf bestimmte Attribute eingeschränkte (und mit (marktierten) Nullvariablen aufgefüllte) Tupel der Quellinstanz I. Die Urinstanz ist somit ähnlich dem Urbild einer Umkehrfunktion (siehe [Bos14]). Für denn Fall, dass $I = I''$ gilt, heißt $\mathcal{M}^\star$ beispielsweise *exakte* CHASE-*Inverse*.

Definition 3.27 (Exakte CHASE-Inverse)**.** Sei $\mathcal{M}$ eine GLAV-Schemaabbildung von einem Schema S zu einem zweiten Schema S'. Dann heißt $\mathcal{M}^\star$ **exakte CHASE-Inverse** für $\mathcal{M}$, wenn $\mathcal{M}^\star$ eine GLAV-Schemaabbildung von S' nach S ist und für jede Instanz I über S gilt:

$$I = \text{chase}_{\mathcal{M}^\star}(\text{chase}_{\mathcal{M}}(I)).$$

□

Beispiel 3.12. *Gegeben sei die GAV-Abbildung* $\mathcal{M}^\star = (S', S, \Sigma)$ *mit*

$$\Sigma = \text{STUDENTEN}\ (a_1, a_2, a_3, a_4) \wedge \text{NOTEN}\ (a_5, a_1, a_6, a_7) \rightarrow \text{NOTEN}_{\text{Max}}(a_1, a_5, a_7)$$

aus Beispiel 3.8. Dann ist $\mathcal{M} = (S, S', \Sigma')$ *die exakte CHASE-Inverse von*

$$\Sigma' = \{\text{Noten}_{\text{Max}}(a_1, a_5, a_7) \rightarrow \exists b_1, b_2, b_3, b_4 : \text{STUDENTEN}(a_1, b_1, b_2, b_3) \wedge \text{NOTEN}(a_5, a_1, b_4, a_7)\}.$$

Denn:

$$\begin{aligned}
\text{chase}_{\mathcal{M}^\star}(\text{chase}_{\mathcal{M}}(I)) &= \text{chase}_{\mathcal{M}^\star}(\text{chase}_{\mathcal{M}}(\text{NOTEN}_{\text{Max}}(a_1, a_5, a_7))) \\
&= \text{chase}_{\mathcal{M}^\star}(\text{STUDENTEN}(a_1, b_1, b_2, b_3), \text{NOTEN}(a_5, a_1, b_4, a_7)) \\
&= \text{NOTEN}_{\text{Max}}(a_1, a_5, a_7) \\
&= I.
\end{aligned}$$

Manchmal wird jedoch eine etwas „entspanntere“ Version einer inversen Abbildung benötigt. In diesem Fall wird eine äquivalente Inverse modulo der (markierten) Nullvariablen angegeben. Ist beispielsweise nicht wie oben eine Schemaabbildung $\mathcal{M} = (S, S', \Sigma)$ in zwei verschiedene Relationenschemata mit

$$\Sigma = \{\mathcal{R}_1(a_1, a_2) \rightarrow \exists a_3 : \mathcal{R}_2(a_1, a_3) \wedge \mathcal{R}_3(a_3, a_2)\}$$

gegeben, sondern gilt

$$\Sigma'' = \{\mathcal{R}_1(a_1, a_2) \rightarrow \exists a_3 : \mathcal{R}_2(a_1, a_3) \wedge \mathcal{R}_2(a_3, a_2)\},$$

so existiert in der Regel keine exakte CHASE-Inverse mehr. Man spricht in diesem Falle von einer CHASE-*Inversen*, welche ein äquivalentes Ergebnis zur Ausgangsinstanz liefert. Zwei Instanzen I_1 und I_2 heißen dabei *äquivalent* ($I_1 \leftrightarrow I_2$), wenn zwischen ihnen ein Homomorphismus existiert.

Definition 3.28 (CHASE-Inverse)**.** Sei $\mathcal{M}$ eine GLAV-Schemaabbildung von einem Schema S zu einem anderen Schema S'. Die GLAV-Schemaabbildung $\mathcal{M}^\star$ von S' nach S heißt **CHASE-Inverse** von $\mathcal{M}$, wenn für jede Instanz I über S gilt:

$$I \leftrightarrow \text{chase}_{\mathcal{M}^\star}(\text{chase}_{\mathcal{M}}(I)).$$

$\boxdot$

Beispiel 3.13. *Der Einfachheit halber sei im folgenden Beispiel die* STUDENTEN*-Relation eingeschränkt auf den Vor- und Nachnamen der Studenten, d.h.* $r(\text{STUD}) :- \pi_{\text{Name,Vorname}} r(\text{STUDENTEN})$. *Dann ist* $\mathcal{M} = (S, S', \Sigma)$ *mit*

$$\Sigma = \{\text{STUD}(a_3, a_2) \rightarrow \exists b : (\theta(a_3, b) \wedge \theta(b, a_2))\}$$

die CHASE-Inverse zu $\mathcal{M}^\star = (S', S, \Sigma')$

$$\Sigma' = \{\theta(a_3, a_5) \wedge \theta(a_5, a_2) \rightarrow \text{STUD}(a_3, a_2)\}.$$

Denn für die Instanz $I = \{\text{STUD}(\text{Paul}, \text{Johannes}), \text{STUD}(\text{Johannes}, \text{Johansen})\}$ *gilt:*

$$\begin{aligned}
\text{chase}_{\mathcal{M}^\star}(\text{chase}_{\mathcal{M}}(I)) &= \text{chase}_{\mathcal{M}^\star}(\text{chase}_{\mathcal{M}}(\{\text{STUD}(\text{Paul}, \text{Johannes}), \text{STUD}(\text{Johannes}, \text{Johansen})\})) \\
&= \text{chase}_{\mathcal{M}^\star}(\{\theta(\text{Paul}, n_1), \theta(n_1, \text{Johannes}), \theta(\text{Johannes}, n_2), \theta(n_2, \text{Johansen})\}) \\
&= \{\text{STUD}(\text{Paul}, \text{Johannes}), \text{STUD}(\text{Johannes}, \text{Johansen}), \text{STUD}(n_1, n_2)\} \\
&= I \cup \{\text{STUD}(n_1, n_2)\}.
\end{aligned}$$

Das Extra-Tupel $\text{STUD}(n_1, n_2)$ *bringt keine neuen Informationen, die nicht durch die Tatsachen der anderen beiden Tupel* $\text{STUD}(\text{Paul}, \text{Johannes})$ *und* $\text{STUD}(\text{Johannes}, \text{Johansen})$ *subsumiert werden. Die Urinstanz* $I'' = I \cup \text{STUD}(n_1, n_2)$ *ist somit äquivalent zur Quellinstanz* I.

Die *relaxte* CHASE-*Inverse* ist eine weitere Abschwächung der CHASE-Inversen. Sie fordert keine Äquivalenzbeziehung zwischen Quell- und Urinstanz I'' — „doppelt-gechasete“ Quellinstanz I —, dafür aber *Ergebnisäquivalenz* sowie die Existenz eines Homomorphismus der Urinstanz I'' in die Quellinstanz I. Der zweite Schritt des „doppelten chasens“ der Quellinstanz I kann aber mit einer BACKCHASE-Phase verglichen werden.

Definition 3.29 (Ergebnisäquivalenz)**.** Sei $\mathcal{M}$ eine GLAV-Schemaabbildung von einem Schema S in ein anderes Schema S'. Seien I und J zwei Instanzen über S. Dann sind I und J **ergebnisäquivalent** bzgl. $\mathcal{M}$, wenn

$$\text{chase}_{\mathcal{M}}(I) \leftrightarrow \text{chase}_{\mathcal{M}}(J)$$

gilt. Schreibweise: $I \leftrightarrow_{\mathcal{M}} J$. $\boxdot$

Definition 3.30 (Relaxte CHASE-Inverse)**.** Seien S und S' zwei Schemata und $\mathcal{M}$ eine GLAV-Schemaabbildung von S nach S'. Dann heißt $\mathcal{M}^\star$ **relaxte CHASE-Inverse** von $\mathcal{M}$ (engl. relaxed CHASE-Inverse), wenn $\mathcal{M}^\star$ eine GLAV-Schemaabbildung von S' nach S ist, so dass für jede Instanz I über S gilt:

- $I'' \leftrightarrow_{\mathcal{M}} I$
- $I'' \rightarrow I$

Dabei ist $I'' = \text{chase}_{\mathcal{M}^\star}(\text{chase}_{\mathcal{M}}(I))$. $\boxdot$

Zur Veranschaulichung der relaxten CHASE-Inversen soll die Abbildung $\mathcal{M}^\star$ aus Beispiel 3.8 um eine Nummerierung $a_8 \in \mathbb{N}$ beginnend bei 1 erweitert werden. In diesem Fall ist bei gleicher „inverser“ Abbildung $\mathcal{M}$ eine Äquivalenzbeziehung zwischen Quell- und Urinstanz nicht mehr gegeben.

Beispiel 3.14. *Gegeben sei die GAV-Abbildung* $\mathcal{M}^\star = (S', S, \Sigma)$ *mit*

$$\Sigma = \text{STUDENTEN}\ (a_1, a_2, a_3, a_4) \wedge \text{NOTEN}\ (a_5, a_1, a_6, a_7) \rightarrow \exists b_1 : \text{NOTEN}_{\text{Max}}(b_1, a_1, a_5, a_7).$$

Dann ist $\mathcal{M} = (S, S', \Sigma')$ *mit*

$$\Sigma' = \{\text{Noten}_{\text{Max}}(a_8, a_1, a_5, a_7) \rightarrow \exists b_2, b_3, b_4, b_5 : \text{STUDENTEN}(a_1, b_2, b_3, b_4) \wedge \text{NOTEN}(a_5, a_1, b_5, a_7)\}$$

eine relaxte CHASE-*Inverse von* $\mathcal{M}^\star$*. Dass es sich bei* $\mathcal{M}$ *nicht um eine* CHASE-*Inverse handelt, zeigt die folgende Überlegung:*

$$\begin{aligned}
\text{chase}_{\mathcal{M}^\star}(\text{chase}_{\mathcal{M}}(I)) &= \text{chase}_{\mathcal{M}}(\text{chase}_{\mathcal{M}^\star}(\text{NOTEN}_{\text{Max}}(a_8, a_1, a_5, a_7))) \\
&= \text{chase}_{\mathcal{M}^\star}(\text{STUDENTEN}(a_1, b_2, b_3, b_4), \text{NOTEN}(a_5, a_1, b_5, a_7)) \\
&= \text{NOTEN}_{\text{Max}}(b_1, a_1, a_5, a_7) \\
&= I''.
\end{aligned}$$

Da an der Position der Variablen a_8 *durch den* CHASE *bzgl.* $\mathcal{M}^\star$ *eine (markierte) Nullvariable entsteht, stimmen die Quellinstanz* $I = (a_8, a_1, a_5, a_7)$ *sowie die Urinstanz nach doppeltem chasen* $I'' = (n_1, a_1, a_5, a_7)$ *nicht mehr überein. Auch lässt sich zwar ein Homomorphismus von* I' *nach* I *jedoch keine Äquivalenzbeziehung definieren. Die erneute Anwendung des* CHASE *auf die Quell- sowie die Zielinstanz liefert die gleiche Instanz* $(\text{STUDENTEN}(a_1, a_2, a_3, a_4), \text{NOTEN}(a_5, a_1, a_6, a_7))$ *und somit den Nachweis für die relaxte* CHASE-*Inverse* $\mathcal{M}$*.*

Abschließen wollen wir diesen Abschnitt mit zwei Sätzen über den Zusammenhang zwischen GLAV- und GAV-CHASE-Inversen sowie relaxten CHASE-Inversen. Sie basieren ebenfalls auf dem Artikel von R. Fagin et al. [FKPT11]. Sie werden im weiteren Verlauf der Arbeit nicht benötigt, sollen aber der Vollständigkeit halber nicht fehlen.

Satz 3.5. *Sei* $\mathcal{M}$ *eine* GLAV-*Schemaabbildung. Wenn* $\mathcal{M}$ *eine* CHASE-*Inverse hat, dann hat* $\mathcal{M}$ *auch eine* GAV-CHASE-*Inverse.*

Satz 3.6. *Sei* $\mathcal{M}$ *eine* GLAV-*Schemaabbildung von einem Schema* S *in ein zweites Schema* S'*, für die eine CHASE-Inverse existiert. Dann sind folgende Aussagen für die* GLAV-*Schemaabbildung* $\mathcal{M}^\star : S' \rightarrow S$ *äquivalent:*

- $\mathcal{M}^\star$ *ist* CHASE-*Inverse von* $\mathcal{M}$
- $\mathcal{M}^\star$ *ist relaxte* CHASE-*Inverse von* $\mathcal{M}$*.*

3.5. Zusammenfassung

In diesem Kapitel haben wir einen Überblick über den Stand der aktuellen Forschung gegeben, diesen auf die Relevanz für unsere Thematik überprüft und sie um einige Kleinigkeiten erweitert. Wir haben uns mit Provenance-Anfragen und -Antworten beschäftigt, sowie die Theorie der Provenance-Polynome untersucht, wobei wir uns auf die ***where***-, ***why***- und ***how***-Anfragen sowie die extensionale Antwort beschränkt haben. Die von uns hierauf definierte Informationskette $\textbf{\textit{where}} \preceq \textbf{\textit{why}} \preceq \textbf{\textit{how}}$ wurde anhand

eines Beispiels erläutert sowie eine Ableitung des Informationsgehalten eines höheren Provenance-Typen erarbeitet. Dieser Zusammenhang muss jedoch noch formal bewiesen werden. Unsere Literaturrecherche hat zudem ergeben, dass die Provenance-Polynome, d.h. die Antwort auf eine ***how***-Anfrage für die Grundoperationen des zweiten Kapitels mit Ausnahme der Selektion auf Ungleichheit und der Differenzbildung bestimmt werden können. Hierbei haben wir weiter festgestellt, das für eine minimale Zeugenbasis stets ein eindeutiges Polynom existiert.

Anschließend haben wir den CHASE auf source-to-target tuple-generating dependencies und equality-generating dependencies sowie den CHASE&BACKCHASE für *Answering Queries using Views* untersucht und anhand von Beispielen getestet. Es ist uns so möglich, den CHASE auf Datenbanken anzuwenden und mit Hilfe der BACKCHASE-Phase die Ergebnisse der Provenance-Analyse zu nutzen. So zumindest unsere Theorie, die wir im nächsten Abschnitt formulieren und zur Bestimmung der CHASE-inversen Abbildungen nutzen wollen. Die CHASE-Inversen dienen hierbei der Rekonstruktion der Quelldatenbank anhand des Anfrageergebnisses und (mit Hilfe einer neu definierten BACKCHASE-Phase) der Informationen aus den Provenance-Berechnungen. Wir können so Anfragen in Form von SQL-Anweisungen, relationenalgebraischen Ausdrücken sowie als Schemaabbildungen über einer Menge von s-t tgds und egds untersuchen.

4. Das Konzept der CHASE-inversen Abbildungen

In diesem Kapitel soll das Konzept der CHASE-inversen Abbildungen weiterentwickelt und vertieft werden. Die Grundlage hierfür bilden die Operationen des zweiten Kapitels sowie die Provenance-Theorie und die verschiedenen CHASE-Varianten des dritten Kapitels. Wir definieren im Folgenden eine weitere inverse Schemaabbildung, die *ergebnisäquivalente* CHASE-*Inverse*, sowie ein spezielles CHASE& BACKCHASE-Verfahren (siehe Abschnitt 4.1). Wir bestimmen die hinreichenden und notwendigen Bedingungen für die Existenz der verschiedene CHASE-Inversen-Typen und untersuchen die relationenalgebraischen Grundoperationen auf ihre Inversen-Typen (siehe Abschnitt 4.2). Abschließend beschäftigen wir uns mit der Komposition von Operationen und analysieren die CHASE-Inversen des Hidden-Markov-Modell als Vertreter eines Machine-Learning-Algorithmus (siehe Abschnitt 4.3). Machine-Learning-Algorithmen sind die aktuell meistdiskutierten Techniken für Big-Data-Analytics-Anwendungen, die ja im Fokus der Themenstellung der Arbeit standen.

4.1. Erweiterung der Theorie der CHASE-inverse Schemaabbildungen

Die Definition einer neuen CHASE-inversen Abbildung (siehe Unterabschnitt 4.1.1) — die *ergebnisäquivalente* CHASE-*Inverse* — und der allgemeine Existenznachweis der verschiedenen CHASE-Inversen-Typen bilden den Schwerpunkt dieses Abschnitts. Für den zweiten Teil stellen wir zunächst einen Zusammenhang zwischen der CHASE& BACKCHASE-Methode und dem Nachweis der CHASE-inversen Abbildungen her (siehe 4.1.2) und bestimmen anschließend die notwendigen und hinreichenden Existenzbedingungen der unterschiedlichen Inversen-Typen (siehe 4.1.4).

4.1.1. Die ergebnisäquivalente CHASE-Inverse

Als weitere Abschwächung der exakten CHASE-Inversen — Abschwächungen sind nach Unterabschnitt 3.4.2 bereits die CHASE-Inverse sowie die relaxte CHASE-Inverse — definieren wir eine *ergebnisäquivalente* CHASE-*Inverse*. Dabei heißt eine Schemaabbildung $M^\star$ ergebnisäquivalente CHASE-Inverse, wenn zwischen dem CHASE angewandt auf die Quell- sowie die Urinstanz eine Äquivalenzbeziehung besteht. Existiert zudem ein Homomorphismus von I'' nach I, erhalten wir stattdessen eine relaxte CHASE-Inverse. Die ergebnisäquivalente CHASE-Inverse ist somit ein schwächere inverse Abbildung als die relaxte CHASE-Inverse. Insgesamt ergibt sich somit die Reduktion

$$\begin{aligned} \text{ergebnisäquivalente CHASE-Inverse} &\preceq \text{relaxte CHASE-Inverse} \\ &\preceq \text{CHASE-Inverse} \\ &\preceq \text{exakte CHASE-Inverse} \end{aligned}$$

Die ergebnisäquivalente CHASE-Inverse basiert ebenso wie die relaxte CHASE-Inverse auf der Definition der *Ergebnisäquivalenz* aus Definition 3.29. Sie sei hier der Vollständig halber noch einmal wiederholt.

Definition 4.1 (Ergebnisäquivalenz)**.** Sei $\mathcal{M}$ eine GLAV-Schemaabbildung von einem Schema S in ein anderes Schema S'. Seien I und J zwei Instanzen über S. Dann sind I und J **ergebnisäquivalent** bzgl. $\mathcal{M}$, wenn

$$\text{chase}_{\mathcal{M}}(I) \leftrightarrow \text{chase}_{\mathcal{M}}(J)$$

gilt. Schreibweise: $I \leftrightarrow_{\mathcal{M}} J$. ⊡

Definition 4.2 (Ergebnisäquivalente CHASE-Inverse)**.** Seien S und S' zwei Schemata und $\mathcal{M}$ eine GLAV-Schemaabbildung von S nach S'. Dann heißt $\mathcal{M}^\star$ **ergebnisäquivalente CHASE-Inverse** von $\mathcal{M}$, wenn $\mathcal{M}^\star$ eine GLAV-Schemaabbildung von S' nach S ist und für jede Instanz I sowie $U = \text{chase}_{\mathcal{M}^\star}(\text{chase}_{\mathcal{M}}(I))$ gilt: $U \leftrightarrow_{\mathcal{M}} I$. ⊡

Beispiel 4.1. *Der Einfachheit halber sei auch in diesem Beispiel die* STUDENTEN*-Relation auf den Vor- und Nachname der Studenten eingeschränkt, d.h.* $r(\text{STUD}) :- \pi_{\text{Name,Vorname}} r(\text{STUDENTEN})$. *Im Folgenden soll die Selektion nach den Studenten mit dem Nachnamen Müller abgebildet werden. Seien dazu zwei Schemaabbildungen* $\mathcal{M} = (S, S', \Sigma)$ *mit*

$$\Sigma = \{(\text{STUD}(a_2, a_3) \wedge a_2 = \text{'Müller'}) \rightarrow \text{STUD}(a_2, a_3)\}$$

und $\mathcal{M}^\star = (S', S, \Sigma')$ *mit*

$$\Sigma' = \{\text{STUD}(a_2, a_3) \rightarrow \text{STUD}(a_2, a_3)\}$$

gegeben. Die Abbildung $\mathcal{M}$ *entspricht hierbei der Namensänderung. Sei weiter*

$$I = \{\text{STUD}(\text{Müller}, \text{Max}), \text{STUD}(\text{Müller}, \text{Mira}), \text{STUD}(\text{Mustermann}, \text{Max})\}$$

die zu untersuchende Instanz. Dann gilt:

$$\begin{aligned}
\text{chase}_{\mathcal{M}^\star}(\text{chase}_{\mathcal{M}}(I)) &= \text{chase}_{\mathcal{M}^\star}(\text{chase}_{\mathcal{M}}(\{\text{STUD(Müller, Max)}, \text{STUD(Müller, Mira)}, \\
&\quad \text{STUD(Mustermann, Max)}\})) \\
&= \text{chase}_{\mathcal{M}^\star}(\{\text{STUD(Müller, Max)}, \text{STUD(Müller, Mira)}\}) \\
&= \{\text{STUD}(\text{Müller}, \text{Max}), \text{STUD}(\text{Müller}, \text{Mira})\} \\
&= I'.
\end{aligned}$$

Wegen $|I| > |I'|$ *kann ein Homomorphismus, welcher* I' *in* I *abbildet, nicht existieren. Die beiden Instanzen* I *und* I' *sind aber ergebnisäquivalent, sodass* $\mathcal{M}^\star$ *ergebnisäquivalente CHASE-Inverse von* $\mathcal{M}$ *ist. Denn:*

$$\begin{aligned}
\text{chase}_{\mathcal{M}}(I') &= \text{chase}_{\mathcal{M}}(\{\text{STUD}(\text{Müller}, \text{Max}), \text{STUD}(\text{Müller}, \text{Mira})\}) \\
&= \{\text{STUD(Müller, Max)}, \text{STUD(Müller, Mira)}\} \\
&= \text{chase}_{\mathcal{M}}(\{\text{STUD(Müller, Max)}, \text{STUD(Müller, Mira)}\}) \\
&= \text{chase}_{\mathcal{M}}(I).
\end{aligned}$$

Mit den im Unterabschnitt 3.4.2 definierten exakten und relaxten CHASE-inversen Abbildungen sowie der ergebnisäquivalenten CHASE-Inversen (von oben) können nun für alle Grundoperationen des zweiten Kapitels inverse Schemaabbildungen gefunden werden. Dies gelingt teilweise ohne und teilweise nur mit der

Betrachtung von Provenance-Polynomen und minimalen Zeugenbasen. Dies soll in den Unterabschnitten 4.2.4 und 4.2.5 jedoch noch weiter untersucht werden.

4.1.2. Das CHASE&BACKCHASE-Verfahren zur Bestimmung von CHASE-inversen Schemaabbildungen

In diesem Unterabschnitt wollen wir die Idee der CHASE&BACKCHASE-Verfahrens mit der Suche nach CHASE-inversen Schemaabbildungen verbinden. Dabei soll die Inverse nicht gefunden, sondern vielmehr ihre Existenz überprüft werden. Nach Definition 3.27 ist $\mathcal{M}^\star$ exakte CHASE-Inverse von $\mathcal{M}$, wenn $\text{chase}_{\mathcal{M}^\star}(\text{chase}_{\mathcal{M}}(I)) = I$ gilt. Die zwei Phasen des CHASE&BACKCHASE-Verfahrens scheinen sich daher für die Existenzüberprüfung anzubieten.

Sei $r(\mathcal{R})$ eine Relation mit Relationenschemata $\mathcal{R}$ sowie S und S' zwei Datenbankschemata. Seien weiter zwei Schemaabbildungen $\mathcal{M} = (S, S', \Sigma)$ und $\mathcal{M}^\star = (S', S, \Sigma')$ gegeben. Dann werden drei Instanzen I, I' und I'' unterschieden, die sich durch einfache oder doppelte Anwendung des CHASE-Algorithmus ergeben:

- Quellinstanz: I
- Ergebnisinstanz: $I' = \text{chase}_{\mathcal{M}}(I)$
- Urinstanz: $I'' = \text{chase}_{\mathcal{M}^\star}(I') = \text{chase}_{\mathcal{M}^\star}(\text{chase}_{\mathcal{M}}(I))$

Die zweite CHASE-Anwendung dient hierbei der „Rückabbildung“ der Ergebnisinstanz I' auf die ursprüngliche Quellinstanz I. In der Regel gelingt dies allerdings aber nur teilweise, sodass eine zusätzliche Urinstanz I'' eingeführt werden muss, die anschließend mit der Quellinstanz verglichen wird. Diese Phase kann als BACKCHASE-Phase interpretiert werden. Die Idee des CHASE&BACKCHASE-Verfahrens wurde in Unterabschnitt 3.3.4 bereits für den Fall äquivalenter Anfragen mittels Sichten vorgestellt. Sind statt Sichten jedoch zwei Schemaabbildungen $\mathcal{M}$ und $\mathcal{M}^\star$ gegeben, kann der BACKCHASE-Schritt $\text{chase}_{\mathcal{M}^\star}(I')$ als eine Art inverse Schemaabbildung des CHASE-Schrittes $\text{chase}_{\mathcal{M}}(I)$ angesehen werden. Mit anderen Worten: Die Abbildung $\mathcal{M}^\star$ ist eine inverse CHASE-Abbildung von $\mathcal{M}$.

Die Definitionen 4.3 und 4.4 formalisieren diese Idee zunächst „klassisch“ — *CHASE&BACKCHASE für den Nachweise von CHASE-Inversen* — und anschließend erweitert um den Aspekt der Provenance — *Provenance-aware CHASE&BACKCHASE für den Nachweis von CHASE-Inversen.* Beim zweiten Verfahren wird in der CHASE-Phase zusätzlich für jedes Tupel der Ergebnisinstanz I' das zugehörige Provenance-Polynom p aufgestellt. Abschließend wird die (minimale) Zeugenbasis bestimmt. In der BACKCHASE-Phase werden mit Hilfe dieser Polynome p die Tupel der Urinstanz I'' wie folgt erzeugt:

- Besteht das Polynom aus nur einem Tupel, bleibt dieses in der Urinstanz erhalten.
- Enthält das Polynom eine Summe, entsteht für jeden Summanden ein neues Tupel in der Urinstanz. Diese ergeben sich durch Erweiterung der Ergebnisinstanztupel um (markierte) Nullvariablen.
- Enthält das Polynom ein Produkt, entsteht für jeden Faktor ein neues Tupel in der Urinstanz I''.
- Enthält das Polynom ein Tensorprodukt $x \otimes p'$, wird das entsprechende Attribut der im Polynom p' enthaltenen Tupel mit dem Attributwert x belegt.

Abschließen wollen wir diese Überlegungen mit einem kurzen Beispiel:

Beispiel 4.2. *Die Schemaabbildug $\mathcal{M}$ entspreche der Projektion $\pi_{A_1A_3}(r(\{A_1, A_2, A_3\}))$ auf das erste sowie das dritte Attribut. Gegeben sei zudem eine Quellinstanz $I = \{(a_1, a_2, a_3), (a_4, a_5, a_6), (a_4, a_7, a_6)\}$. Dann ergibt sich in der* CHASE*-Phase:*

$\text{chase}_{\mathcal{M}}(I)$:

A_1	A_3	
a_1	a_3	t_1
a_4	a_6	$t_2 +_k t_3$

mit der Polynommenge $P = \{t_1, t_2 +_K t_3\}$ sowie den beiden minimalen Zeugenbasen $\{\{t_1\}, \{t_2\}\}$ und $\{\{t_2\}, \{t_3\}\}$. Die Wahl der minimalen Zeugenbasis ist beliebig, üblicherweise wird aber die Basis mit den kleineren Indizes gewählt Die anschließende BACKCHASE*-Phase liefert dann:*

$\text{chase}_{\mathcal{M}^\star}(I')$:

A_1	A_2	A_3	
a_1	n_1	a_3	t_1
a_4	n_2	a_6	t_2
a_4	n_3	a_6	t_3

Es folgen die formalen Definitionen des CHASE&BACKCHASE-Verfahrens sowie des provenance-aware CHASE&BACKCHASE-Verfahrens für den Nachweis von CHASE-Inversen. Die im provenance-basierten Verfahren resultierende (minimale) Zeugenbasis gibt dabei an, welche Quelltupel für eine korrekte Rekonstruktion der Abbildung $\mathcal{M}$ gemerkt werden müssen. Die Quelltupel dürfen dabei auf einem eingeschränkten Relationenschema $\mathcal{R}|_{A_1 \ldots A_{i-1} A_{i+1} A_n}$ definiert sein.

Definition 4.3 (CHASE&BACKCHASE für den Nachweis von CHASE-Inversen)**.** Seien zwei Schemaabbildungen $\mathcal{M} = (S, S', \Sigma)$ und $\mathcal{M}^\star = (S', S'', \Sigma')$ gegeben. Sei weiter I eine Quellinstanz auf dem Quellschema S. Dann gilt:

- **CHASE:** Berechne den CHASE von I bzgl. $\mathcal{M}$ als Sequenz von tgd- und egd-Regeln. Vergleiche hierzu die Sätze 3.18 und 3.19 sowie die Definition 3.20. Das Ergebnis dieser CHASE-Anwendung werde als I' bezeichnet.
- **BACKCHASE:** Berechne den CHASE von I' bzgl. $\mathcal{M}^\star$ als Sequenz von tgd- und egd-Regeln.

$\boxdot$

Definition 4.4 (Provenance-aware CHASE&BACKCHASE für den Nachweis von CHASE-Inversen)**.** Gegeben seien drei Datenbankschemata S, S' und S'' sowie zwei darauf definierte Schemaabbildungen $\mathcal{M} = (S, S', \Sigma)$ und $\mathcal{M}^\star = (S', S'', \Sigma')$. Sei weiter I eine Quellinstanz auf dem Quellschema S. Dann gilt:

- **CHASE:**
  1. Berechne den CHASE von I bzgl. $\mathcal{M}$ als Sequenz von tgd- und egd-Regeln. Das Ergebnis dieser CHASE-Anwendung werde als I' bezeichnet.
  2. Bestimme $P = \{p_i \mid p_i \text{ ist Provenance-Polynom des Tupels } t_i \in I'\}$
  3. Bestimme die (minimale) Zeugenbasis von P.
- **Provenance-directed BACKCHASE:** Berechne die Urinstanz I'' durch den CHASE von I' bzgl. $\mathcal{M}^\star$ unter Berücksichtigung der Menge der Provenance-Polynome P. Seien dazu t_j bzw. t_k mit $j, k \in \mathbb{N}$ die Tupelidentifikatoren der Quellinstanz I und p_i ($i \in \mathbb{N}$) die Provenance-Polynome von I'. Sei weiter a_{lj} mit $j, l \in \mathbb{N}$ der Attributwert des l-ten Attributes des j-ten Tupels t_j. Dann gilt:

- $p_i = t_j \Rightarrow \text{chase}_{\mathcal{M}^\star}(\text{Zeile } i) = \{t'_j\}$, wobei t'_j dem gechasten Tupel t_j entspricht.
- $p_i = t_j +_K t_k \Rightarrow \text{chase}_{\mathcal{M}^\star}(\text{Zeile } i) = \{t'_j, t'_k\}$. Durch den CHASE entstehen also zwei neue Tupel t'_j und t'_k.
- $p_i = t_j +_K t_k \Rightarrow \text{chase}_{\mathcal{M}^\star}(\text{Zeile } i) = \{t'_j, t'_k\}$. Durch den CHASE entstehen also zwei neue Tupel t'_j und t'_k.
  - Duplikatbildung im Falle der Projektion, Aggregation: Die Tupel t'_j und t'_k sind Variationen der Quelltupel t_j und t_k — entstanden durch das Auffüllen mit (markierten) Nullvariablen. Die neuen Tupel entstammen dabei derselben Quellrelation.
  - Vereinigung, Schnitt: Die Tupel t'_j und t'_k entsprechen den Quelltupeln t_j und t_k der beiden Quellrelationen.
- $p_i = t_j \cdot_K t_k \Rightarrow \text{chase}_{\mathcal{M}^\star}(\text{Zeile } i) = \{t'_j, t'_k\}$. Der CHASE liefert ebenfalls zwei Tupel t_j sowie t_k aus verschiedenen Quellrelationen.
- $p_i = a_{lm} \otimes p_m \Rightarrow$ Ersetze den Wert des Attributes A_l durch a_{lj} in allen durch $\text{chase}_{\mathcal{M}^\star}$ mit t_j erzeugten Tupeln.

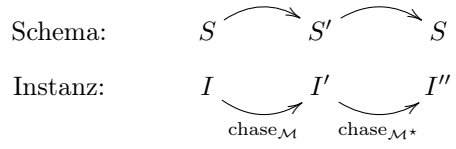
□

4.1.3. Schemata, Instanzen und Teilmengenbeziehungen

Für eine Anfrage Q und eine Quellinstanz I einer Datenbank d liefert der CHASE-Algorithmus eine Ergebnisinstanz $I' = \text{chase}_{\mathcal{M}}(I)$, wenn

- Q wird als Schemaabbildung $\mathcal{M} = (S, S', \Sigma)$ mit Quell- und Zielschema S und S' sowie einer Menge von Abhängigkeiten Σ aufgefasst;
- I ist Quellinstanz in S;
- I' ist Zielinstanz in S'.

Die Urinstanz $I'' = \text{chase}_{\mathcal{M}^\star}(I') = \text{chase}_{\mathcal{M}^\star}(\text{chase}_{\mathcal{M}}(I))$ ist somit das Ergebnis einer Rückanfrage Q' auf die Ergebnisinstanz I'. Dabei enthält I'' ganze oder auf bestimmte Attribute des Quellschemas eingeschränkte (und mit (marktierten) Nullvariablen aufgefüllte) Tupel aus I. Vergleiche hierzu auch die Überlegungen aus Unterabschnitt 3.4.2.



Enthält die Urinstanz I'' Tupel mit (markierten) Nullvariablen, deren restlichen Attributwerte mit den Attributwerten eines Tupels der Quellinstanz I übereinstimmen, beispielsweise $(1, 3, n_1) \in I''$ und $(1, 3, 5) \in I$ (siehe Tabelle 4.1), so heißt I'' **Ausschnitt** der Instanz I und schreiben $I'' \preccurlyeq I$. Mit anderen Worten, es existiert ein Homomorphismus h, welcher die Tupel aus I'' auf die Tupel aus I abbildet. Enthält I'' keine (markierten) Nullvariablen und sind alle Tupel aus I'' auch Tupel in I, schreiben wir weiterhin $I'' \subseteq I$.

I :

A_1	A_2	A_3
1	3	5
2	1	4
2	1	3

und

I'' :

A_1	A_2	A_3
1	3	n_1
2	1	n_2

Tabelle 4.1. I'' ist Ausschnitt von I, d.h. $I'' \preccurlyeq I$

4.1.4. Existenzregeln

Ob eine Schemaabbildung $\mathcal{M}^\star$ (exakte / relaxte / ergebnisäquivalente) CHASE-Inverse einer Schemaabbildung $\mathcal{M}$ ist, lässt sich anhand der Quell- und Urschemata und -instanzen leicht überprüfen. Zumindest, solange wir für die Schemaabbildung $\mathcal{M}$ nur die Grundoperationen aus Kapitel 2 sowie einfache Kompositionen dieser Operationen zulassen. Alte und neue Attributnamen sind in der Definition von $\mathcal{M}$ bereits enthalten, sodass die Schemata bei unseren weiteren Untersuchungen vernachlässigt werden können. In diesem Unterabschnitt stellen wir die notwendigen und hinreichenden Bedingungen für die Existenz der unterschiedlichen CHASE-Inversen-Typen vor.

Exakte CHASE-Inverse:

Notwendige Bedingung für die Existenz einer exakten CHASE-Inversen ist die Übereinstimmung der einzelnen Relationenschemata sowie Instanzen im Quell- und Urschema, d.h. $S = S'$ und $I = I''$. Dies ist beispielsweise beim Kopieren bzw. der Identitätsabbildung der Fall. Eine hinreichende Bedingung kann für diesen Inversen-Typ nicht angegeben werden.

CHASE-Inverse:

Aufgrund der in Unterabschnitt 4.1.1 vorgestellten Reduktion

$$\text{CHASE-Inverse} \preceq \text{exakte CHASE-Inverse}$$

genügt die Existenz einer exakten CHASE-Inversen als hinreichende Bedingung für Existenz einer CHASE-Inversen. Die notwendige Bedingung ist gegeben durch die Äquivalenz von Quell- und Urinstanz sowie die Übereinstimmung der zugehörigen Relationenschemata.

Relaxte CHASE-Inverse:

Der Verlust von Zeilen wie etwa bei der Selektion oder dem natürlichen Verbund muss für die Existenz einer relaxten CHASE-Inversen unbedingt vermieden werden. Spalten dürfen hingegen verloren gehen (vgl. beispielsweise die Projektion). Als notwendige Bedingung ergibt sich somit eine Übereinstimmung der Relationenschemata im Quell- und Ergebnisschema sowie der Erhalt der Tupelmenge der Urinstanz. Dabei dürfen durchaus einzelne Attributwerte durch (markierte) Nullvariablen ersetzt werden. Als hinreichende Bedingung gilt die Existenz einer CHASE-Inversen nach Definition 3.28, denn es gilt:

$$\text{relaxte CHASE-Inverse} \preceq \text{CHASE-Inverse}$$

Ergebnisäquivalente CHASE-Inverse:

Bleibt als einzige konstante Komponente das Relationenschema erhalten, d. h. wir lassen sowohl den Verlust einzelner Zeilen (beispielsweise durch Duplikatbildung oder dangling tuples) als auch Spalten zu, kann lediglich eine ergebnisäquivalente CHASE-Inverse angegeben werden. Der Erhalt des Relationenschemas ist somit die notwendige Bedingung für die Existenz einer solchen inversen Abbildung. Die hinreichende Bedingung ergibt sich analog zu oben aus der Existenz einer relaxten CHASE-Inverse, denn

$$\text{ergebnisäquivalente CHASE-Inverse} \preceq \text{relaxte CHASE-Inverse}.$$

Keine CHASE-Inverse:

Besteht die Urinstanz lediglich aus Nulltupeln, so existiert keine der obigen CHASE-inversen Abbildungen. Dies betrifft insbesondere die Aggregationen **COUNT**, **SUM** und **AVG**, bei denen sämtliche Informationen einer Spalte in einem Wert zusammengefasst werden. Hier ist die Verdichtung so groß, dass keine Rückschlüsse auf die Quellinstanz mehr gezogen werden können. Auch für Operationen, die nicht als s-t tgds und egds geschrieben werden können, existiert keine (exakte / relaxte / ergebnisäquivalente) CHASE-Inverse.

CHASE und Provenance:

Die Erweiterung des CHASE um den Aspekt der Provenance-Auswertung in einer separaten BACKCHASE-Phase kann den Erhalt des Relationenschemas gewährleisten — nach unserer Definition einer Schemaabbildung hier nicht notwenig — sowie den Zeilenverlust bei der Duplikatbildung verhindern. Dies kommt insbesondere bei der Projektion zum Tragen. In diesem Fall kann so stets eine relaxte CHASE-Inverse angegeben werden. Für die Vereinigung kann eine exakte CHASE-Inverse und für die Aggregation eine relaxte CHASE-Inverse angegeben werden. Dies wäre ohne die Untersuchung der Provenance-Polynome nicht möglich.

4.2. Algebraische Grundoperationen mit zugehörigen CHASE-inversen Schemaabbildungen

Ziel dieser Arbeit ist die Untersuchung des relationalen Datenbankmodells auf die Existenz von CHASE-inversen Abbildungen. In den folgenden Unterabschnitten soll für die klassischen relationalen Operationen untersucht werden, um welche inversen CHASE-Abbildungen es sich hierbei handelt. Mögliche Inverse sind die exakte CHASE-Inverse, die CHASE-Inverse, die relaxte CHASE-Inverse sowie die ergebnisäquivalente CHASE-Inverse. Kann keine exakte CHASE-Inverse angegeben werden, folgen einige Überlegungen dazu, welche zusätzlichen Informationen für die Existenz einer solchen Inversen nötig wären (siehe Unterabschnitt 4.2.6).

Dazu beginnen wir mit einem kurzen Überblick über die untersuchten Operationen, die anschließend in den Unterabschnitten 4.2.4 und 4.2.5 genauer analysiert werden. Dabei wird die BACKCHASE-Phase des CHASE&BACKCHASE-Verfahrens im vierten Abschnitt (vgl. Definition 4.4) um den Aspekt der Provenance erweitert. Die Basis hierfür bilden die Provenance-Polynome aus Abschnitt 3.2. Abschließen wollen wir unsere Untersuchungen mit der Komposition der Grundoperationen im fünften Abschnitt.

4.2.1. Einschränkung auf bestimmte Operationen

Wir beschränken uns hierbei auf die Betrachtung der relationalen Operationen aus Unterabschnitt 2.2.1. Dies sind die Selektion und Projektion, der natürliche Verbund, die Mengenoperationen (Vereinigung, Durchschnitt, Differenz) sowie die Umbenennung, die klassischen Aggregatfunktionen (Minimum, Maximum, Aufzählung, Summe, Durchschnitt) und die Gruppierung. Diese Einschränkung genügt, da alle Anfragen wie beispielsweise die Anfrage 3.6:

$$\pi_{A_1}(\sigma_{A_3=\text{'Max'}}(r_1)) \bowtie \pi_{A_1 A_5 A_7}(r_2))$$

auf diese Grundoperationen zurückgeführt werden können.

Die in Unterabschnitt 2.2.2 vorgestellten OLAP-Operationen können ebenfalls auf diese Grundoperationen zurückgeführt werden. So kann beispielsweise die SQL-Anfrage 4.1 zunächst in die Anfrage 4.2 und anschließend in die Anfrage 4.3 umformuliert werden.

Anfrage 4.1 Die OLAP-Operation GROUP BY CUBE

```
SELECT A, B, C, sum(D)
FROM t
GROUP BY CUBE(A,B,C)
```

Anfrage 4.2 Die OLAP-Operation GROUP BY GROUPING SETS

```
SELECT A, B, C, SUM(D)
FROM t
GROUP BY GROUPING SETS ((A, B, C), (A, B), (A,C), (B,C), (A), (B), (C),())
```

Anfrage 4.3 Die OLAP-Operation GROUP BY

```
(SELECT A, B, C, SUM(D)
FROM t
GROUP BY A, B, C)
   UNION
(SELECT A, B, NULL AS C, SUM(D)
FROM t
GROUP BY A, B)
   UNION
     ⋮
   UNION
(SELECT NULL AS A, NULL AS B, C, SUM(D)
FROM t
GROUP BY C)
   UNION
(SELECT NULL AS A, NULL AS B, NULL AS C, SUM(D)
FROM t)
```

Sowohl der CHASE-Algorithmus als auch die Provenance-Plynome machen bei Differenz und Selektion auf Ungleichheit extreme Probleme. Beide Techniken sind eigentlich nur für die positive relationale Algebra ausgelegt. Eine Aussage über die Existenz einer CHASE-inversen Abbildung kann in diesen Fällen daher

nicht getroffen werden. Für die weiteren Untersuchungen in den Unterabschnitten 4.2.4 und 4.2.5 werden sie daher weitestgehend vernachlässigt.

4.2.2. Die Grundoperationen als s-t tgds und egs

Für die Untersuchung der Grundoperationen wollen wir den CHASE-Agorithmus für s-t tgds und egds aus Definition 3.20 verwenden. Bevor wir diesen anwenden können, müssen wir uns zunächst überlegen, wie diese Operationen als tuple- und equality-generating dependencies geschrieben werden können. Für die Projektion, den natürlichen Verbund, die Selektion sowie die Mengenoperationen (Vereinigung, Schnitt und Differenz) dient uns die Datalog-Notation aus [Bal17] als Grundlage. Für die Aggregation und Gruppierung verweisen wir auf [MG12], da sie nur mit sehr großem Aufwand und hier nicht eingeführten Definitionen gelingt.

Kopieren:

$$r(a_1, a_2, a_3) \rightarrow r(a_1, a_2, a_3)$$

Umbenennung:

$$r(a_1, a_2, a_3) \rightarrow \exists\ a_1', a_2', a_3' : r(a_1', a_2', a_3')$$

Projektion:

$$r(a_1, a_2, a_3) \rightarrow r(a_1, a_3)$$

Natürlicher Verbund:

$$r_1(a_1, a_2) \wedge r_2(a_2, a_3) \rightarrow r(a_1, a_2, a_3)$$

Selektion:

$$\exists\ c \in \mathbb{R} : r(a_1, a_2) \wedge a_1 \theta c \rightarrow r(a_1, a_2),$$
wobei $\theta \in \{<, \leq, =, \neq, \geq, >\}$

Mengenoperationen:

Vereinigung	$r_1(a_1, a_2) \rightarrow r(a_1, a_2)$; $r_2(a_1, a_2) \rightarrow r(a_1, a_2)$
Schnitt:	$r_1(a_1, a_2) \wedge r_2(a_1, a_2) \rightarrow r(a_1, a_2)$
Differenz	$r_1(a_1, a_2) \wedge \neg r_2(a_1, a_2) \rightarrow r(a_1, a_2)$

Arithmetische Operationen:

Multiplikation / Division:	$\exists\ c \in \mathbb{R} : r(a_1, a_2, a_3) \wedge (a_1' = a_1 \theta c) \rightarrow r(a_1', a_2, a_3)$ mit $\theta \in \{\cdot, :\}$
Addition / Subtraktion:	$\exists\ c \in \mathbb{R} : r(a_1, a_2, a_3) \wedge (a_2' = a_2 \theta c) \rightarrow r(a_1, a_2', a_3)$ mit $\theta \in \{+, -\}$
Modulo:	$\exists\ c \in \mathbb{R} : r(a_1, a_2, a_3) \wedge (a_3' = a_3 \% c) \rightarrow r(a_1, a_2, a_3')$ mit $\theta \in \{+, -\}$

4.2.3. Ein Überblick

Die untersuchten Operationen und ihre zugehörigen inversen Abbildungen sind in Tabelle 4.2 dargestellt. Dies sind zum einen Abbildungen ohne und zum anderen mit zusätzlichen Provenance-Informationen. Eine bzgl. CHASE inverse Schemaabbildung kann für die klassischen Aggregatfunktionen sowie die Group By-Operation nicht angegeben werden. Hierfür sind zusätzliche Provenance-Informationen zwingend notwendig.

Operation:	Inverse Abbildung ohne Provenance-Information:	Seite:	Inverse Abbildung mit Provenance-Information:	Seite:
$r(\mathcal{R})$	Exakt	79	Exakt	90
$\beta_{A_j \leftarrow A_i}(r(\mathcal{R}))$	Exakt	80	Exakt	90
$\pi_{A_i}(r(\mathcal{R}))$	Relaxt Ergebnisäquivalent	80	Relaxt Relaxt	91
$r_1(\mathcal{R}_1) \bowtie r_2(\mathcal{R}_2)$	Exakt Ergebnisäquivalent	81	Exakt Ergebnisäquivalent	92
$\sigma_{A_i \theta c}(r(\mathcal{R}))$ $\sigma_{A_i \theta A_j}(r(\mathcal{R}))$ mit $\theta \in \{<, \leq, =, \geq, >\}$ $\sigma_{A_i \neq c}(r(\mathcal{R}))$ $\sigma_{A_i \neq A_j}(r(\mathcal{R}))$	Ergebnisäquivalent Ergebnisäquivalent xxx[1] xxx	82	Ergebnisäquivalent Ergebnisäquivalent xxx xxx	93
$r_1(\mathcal{R}_1) \cup r_2(\mathcal{R}_2)$ $r_1(\mathcal{R}_1) \cap r_2(\mathcal{R}_2)$ $r_1(\mathcal{R}_1) - r_2(\mathcal{R}_2)$	Ergebnisäquivalent Ergebnisäquivalent xxx	84	Exakt Ergebnisäquivalent xxx	94
MAX$(r(\mathcal{R}))$ / **MIN**$(r(\mathcal{R}))$ **COUNT**$(r(\mathcal{R}))$ **SUM**$(r(\mathcal{R}))$ **AVG**$(r(\mathcal{R}))$	Ergebnisäquivalent Relaxt xxx xxx	86	Ergebnisäquivalent Relaxt Exakt Exakt	95
$\gamma_{G_i;F_j(A_j)}(r(\mathcal{R}))$[2]	Ergebnisäquivalent Relaxt xxx	87	Ergebnisäquivalent Relaxt Exakt	97
$r(\mathcal{R})\ \theta\ \alpha$ mit $\theta \in \{+, -, \cdot, :\}$ $r(\mathcal{R})$ **MOD** α	Exakt Ergebnisäquivalent	88	Exakt Ergebnisäquivalent	98

Abkürzungen:

[1] Es existiert keine (exakte / relaxte / ergebnisäquivalente) CHASE-Inverse.

[2] Wir unterscheiden drei Fälle:

1. Fall: $F_j \in \{$**MAX**, **MIN**$\} \Rightarrow$ Ergebnisäquivalente CHASE-Inverse;
2. Fall: $F_j \in \{$**COUNT**$\} \Rightarrow$ Relaxte CHASE-Inverse;
3. Fall: $F_j \in \{$**SUM**, **AVG**$\} \Rightarrow$
   - Es existiert keine der obigen CHASE-Inversen (ohne Provenance-Informationen).
   - Exakte CHASE-Inverse (mit Provenance-Informationen)

Tabelle 4.2. Untersuchte Operation mit zughörigen inversen Abbildungen (ohne und mit Provenance-Informationen)

Für die Projektion sowie den natürlichen Verbund müssen aufgrund von *Duplikaten* und *Dangling tuples* zudem zwei Fälle unterschieden werden. Die Duplikate der Mengenoperation $\cup$ können für das Aufstellen einer inversen Abbildung jedoch vernachlässigt werden.

Definition 4.5 (Duplikat)**.** Ein **Duplikat** ist ein redundantes Tupel einer Datenbank d, d. h. ein mehrfach vorkommendes Tupel t, dessen Redundanz aber aufgrund abweichender Schreibweise nicht durch Prüfung auf gleiche Inhalte erkannt werden kann. $\boxdot$

Definition 4.6 (Dangling tuple)**.** Ein **Dangling tuple** ist ein Tupel t einer Datenbank d, welcher bei einem JOIN keinen Partner findet. Es wird somit eliminiert. $\boxdot$

4.2.4. CHASE-inverse Schemaabbildungen

Die folgenden Abschnitte bestehen immer aus einer Übersicht über die betrachteten Operationen, den gefundenen Inversen-Typen (sofern mindestens einer existiert) sowie eventuell zu beachtenden Problemen (Duplikate oder dangling tuples). Die anschließenden formalen Gedankengänge werden jeweils durch ein Beispiel ergänzt.

Seien dazu drei Relationen r, r_1 und r_2 sowie drei Relationenschemata $\mathcal{R}$, $\mathcal{R}_1$ und $\mathcal{R}_2$ gegeben. Seien I, I_1 und I_2 drei Quellinstanzen der Relationen $r(\mathcal{R})$, $r_1(\mathcal{R}_1)$ und $r_2(\mathcal{R}_2)$ und I'', I_1'' sowie I_2'' die zugehörigen Urinstanzen. Die Urinstanzen sind die Ergebnisinstanzen nach der Invertierung; die Ergebnisinstanz I' ist das eigentliche Ergebnis der Originalanfrage, die danach wegen Provenance invertiert werden soll. Dann können auf den zugehörigen Schemata S, S' und S'' zwei Abbildungen $\mathcal{M} : S \to S'$ und $\mathcal{M}^\star : S' \to S''$ definiert werden. Siehe auch Abschnitt 4.1.4.

Kopieren / Identitätsabbildung:

Operation:	$r(\mathcal{R})$
Inverse Abbildung:	Exakte CHASE-Inverse
Probleme / Schwierigkeiten:	Keine

Für die Kopieroperation kann eine exakte CHASE-Inverse direkt angegeben. Sei dafür die Schemaabbildung $\mathcal{M}$ als Identitätsabbildung definiert. Dann ist $\mathcal{M}^\star = \mathrm{Id}$ wegen

$$\mathrm{chase}_{\mathcal{M}^\star}(\mathrm{chase}_{\mathcal{M}}(I)) = \mathrm{chase}_{\mathcal{M}^\star}(I) = I$$

die eindeutige Inverse zu $\mathcal{M}$. Die folgende Tabelle 4.25 verdeutlicht diesen Sachverhalt für eine gegebene Quellinstanz $I = \{(1,3,5),(2,1,4),(2,2,3)\}$. Da Quell- und Urinstanz übereinstimmen, entspricht h somit ebenfalls der Identitätsabbildung.

A_1	A_2	A_3
1	3	5
2	1	4
2	2	3

$\xrightarrow{\mathrm{chase}_{\mathcal{M}}}$

A_1	A_2	A_3
1	3	5
2	1	4
2	2	3

$\xrightarrow{\mathrm{chase}_{\mathcal{M}^\star}}$

A_1	A_2	A_3
1	3	5
2	1	4
2	2	3

$\overset{h}{\longleftrightarrow}$

Tabelle 4.3. Beispiel für eine inverse Schemaabbildung der Operation Kopieren

Umbenennung:

Operation:	$\beta_{A_j \leftarrow A_i} r(\mathcal{R})$
Inverse Abbildung:	Exakte CHASE-Inverse
Probleme / Schwierigkeiten:	Keine

Nach der Umbenennung der Attributnamen in der CHASE-Phase folgt in der BACKCHASE-Phase eine Rückumbenennung der Ergebnisinstanz I', da die Umbenennung in $\mathcal{M}$ implizit berücksichtigt wird. Wie bei der Identitäsabbildung stimmen Quell- und Urinstanz überein. Die Umbenennung kann daher eindeutig invertiert werden und $\mathcal{M}^\star = \beta_{A_i \leftarrow A_j} r(\mathcal{R})$ ist exakte CHASE-Inverse von $\mathcal{M} = \beta_{A_j \leftarrow A_i} r(\mathcal{R})$.

A_1	A_2	A_3
1	3	5
2	1	4
2	2	3

$\xrightarrow{\text{chase}_{\mathcal{M}}}$

A_4	A_5	A_6
1	3	5
2	1	4
2	2	3

$\xrightarrow{\text{chase}_{\mathcal{M}^\star}}$

A_1	A_2	A_3
1	3	5
2	1	4
2	2	3

h

Tabelle 4.4. Beispiel für eine inverse Schemaabbildung der Operation β

Projektion:

Aufgrund von Duplikatbildung kann im Fall der Projektion kein eindeutiger inverser Abbildungstyp bestimmt werden. Liegt eine Duplikatbildung vor, kann lediglich eine ergebnisäquivalente CHASE-Inverse angegeben werden, andernfalls lässt sich eine relaxte CHASE-Inverse finden. Grund hierfür ist die „Reduzierung“ der Tupelanzahl $m_{I'}$ der Ergebnisinstanz I'. Nach dem Streichen der Duplikate kann kein Rückschluss mehr auf die ursprüngliche Tupelanzahl m_I mehr gezogen werden.

Operation:	$\pi_{A_i}(r(\mathcal{R}))$
Inverse Abbildung:	Relaxte CHASE-Inverse / Ergebnisäquivalente CHASE-Inverse
Probleme / Schwierigkeiten:	Duplikate

Verdeutlicht werden soll dies an zwei Beispielen. Sie sind in den Tabellen 4.5 und 4.6 dargestellt. Gegeben seien hierfür eine Schemaabbildung $\mathcal{M} = \pi_{A_1 A_2}(r(\{A_1, A_2, A_3\}))$ sowie eine Quellinstanz I, die sich lediglich im dritten Tupel im zweiten Attribut unterscheidet. Hierdurch entsteht im zweiten Beispiel in der CHASE-Phase $\text{chase}_{\mathcal{M}}(I)$ das Tupelduplikat $(2, 1)$.

A_1	A_2	A_3
1	3	5
2	1	4
2	2	3

$\xrightarrow{\text{chase}_{\mathcal{M}}}$

A_1	A_2
1	3
2	1
2	2

$\xrightarrow{\text{chase}_{\mathcal{M}^\star}}$

A_1	A_2	A_3
1	3	n_1
2	1	n_2
2	2	n_3

h

$\preccurlyeq$

Tabelle 4.5. Beispiel für eine inverse Schemaabbildung der Operation π

In der BACKCHASE-Phase, d.h. der Anwendung $\text{chase}_{\mathcal{M}^\star}$ auf die Ergebnisinstanz I', werden für das Attribut A_3 Nullvariablen $n_1, ..., n_{m_{I'}}$ vergeben. Da im ersten Beispiel (siehe Tabelle 4.5) sowohl die Quellinstanz als auch die Urinstanz drei Tupel enthält, existiert ein Homomorphismus h mit $n_1 := 5$, $n_2 := 4$ und $n_3 := 3$, welcher I'' in I überführt. Es existiert also eine relaxte CHASE-Inverse.

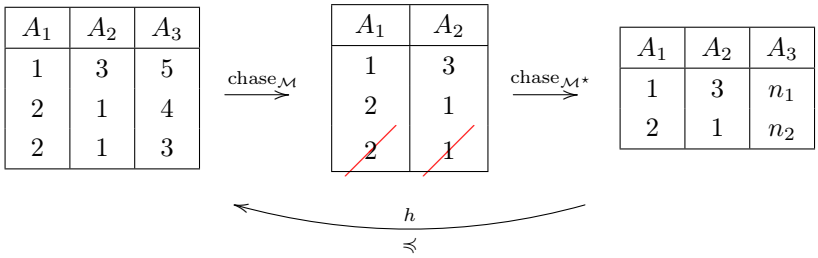


Tabelle 4.6. Zweites Beispiel für eine inverse Schemaabbildung der Operation π

Im zweiten Beispiel (siehe Tabelle 4.6) enthält das Ergebnisenrgebnis nach der Duplikateliminierung nur noch zwei Tupel. Das gelöschte Tupel ist in der Tabelle rot markiert. Es gilt somit $2 = m_{I'} < m_I = 3$ und ein Homomorphismus h wie oben kann nun nicht mehr definiert werden. Es liegt lediglich eine $\preccurlyeq$-Beziehung vor.

Der zweite Fall tritt nur auf, sollte ein Attribut des Schlüssels bzw. eines Unique-Constraints entfernt werden, in diesem Fall A_3 als Teil des Schlüssels (A_2, A_3). Wird stattdessen auf den Schlüssel projiziert, treten nach Definition keine Duplikate auf.

Die erneute Anwendung des CHASE bzgl. $\mathcal{M}$ auf die I sowie die I'' liefert jedoch in beiden Beispielen dasselbe Ergebnis; sie sind ergebnisäquivalent. Wir haben für das erste Beispiel somit die Existenz einer relaxten CHASE-Inversen und für das zweite Beispiel die Existenz einer ergebnisäquivalenten CHASE-Inversen nachgewiesen.

Natürlicher Verbund:

Für den natürlichen Verbund müssen ebenfalls zwei verschiedene Fälle unterschieden werden. Das Problem bei dieser Operation ist die Existenz von dangling tuples. Enthält eine Quellinstanz I_1 oder I_2 Tupel, die in der Ergebnisinstanz I' verloren gehen, kann eine exakte CHASE-Inverse nicht mehr angegeben werden. Die vorliegenden Informationen genügen dann nur noch für die Angabe einer ergebnisäquivalenten CHASE-Inversen.

Operation:	$r_1(\mathcal{R}_1) \bowtie r_2(\mathcal{R}_2)$
Inverse Abbildung:	Exakte CHASE-Inverse / Ergebnisäquivalente CHASE-Inverse
Probleme / Schwierigkeiten:	Dangling tuples

Betrachten wir auch hierzu zwei Beispiele. Gegeben seien jeweils eine Schemaabbildung $\mathcal{M} = r_1(\mathcal{R}_1) \bowtie r_2(\mathcal{R}_2)$ sowie zwei Quellinstanzen I_1 und I_2. Im ersten Beispiel (siehe Tabelle 4.7) kann nun eine eindeutige inverse Abbildung $\mathcal{M}^\star = (r_1(\{A_1, A_2\}) \bowtie r_2(\{A_3, A_4\}))^{-1}$ angegeben werden. Diese ist exakte CHASE-Inverse von $\mathcal{M}$. Der Homomorphismus h entspricht hierbei der Identitätsabbildung Id, da Quell- und Urinstanzen übereinstimmen.

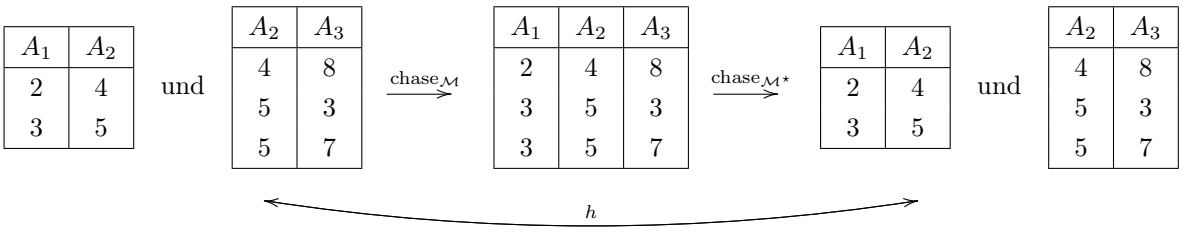


Tabelle 4.7. Beispiel für eine inverse Schemaabbildung der Operation $\bowtie$

Eine notwendige Bedingung für die Verhinderung von dangling tuples ist die Existenz von Fremdschlüsseln. Es kann aber auch ohne Fremdschlüssel zufälligerweise Instanzen geben, wo keine dangling tuples auftreten. Eine Bedingung für die Existenz einer exakten CHASE-Inversen kann somit die Fremdschlüsselbedingung des Verbundattributes sein. Das heißt, alle Attributwerte des Fremdschlüssels tauchen in der anderen Relation als Werte des Schlüssels auf.

Im zweiten Beispiel (siehe Tabelle 4.8) ist diese Fremdschlüsselbeziehung nicht gegeben. In der Ergebnisinstanz entfallen die Tupel $(1, 3, x)$ und $(x, 9, 5)$, sodass eine komplette Rekonstruktion der Quellinstanzen I_1 und I_2 nicht mehr möglich ist. Es gilt stattdessen: $I_1 = I_1'' \cup \{(1, 3)\}$ sowie $I_2 = I_2'' \cup \{(9, 5)\}$. Die Abbildung h spiegelt daher lediglich eine Teilmengenbeziehung wieder. Der CHASE der Quell- sowie der Urinstanzen liefert jedoch unabhängig voneinander die Ergebnisinstanz I'. Zu $\mathcal{M}$ existiert somit eine ergebnisäquivalente CHASE-Inverse $\mathcal{M}^\star$.

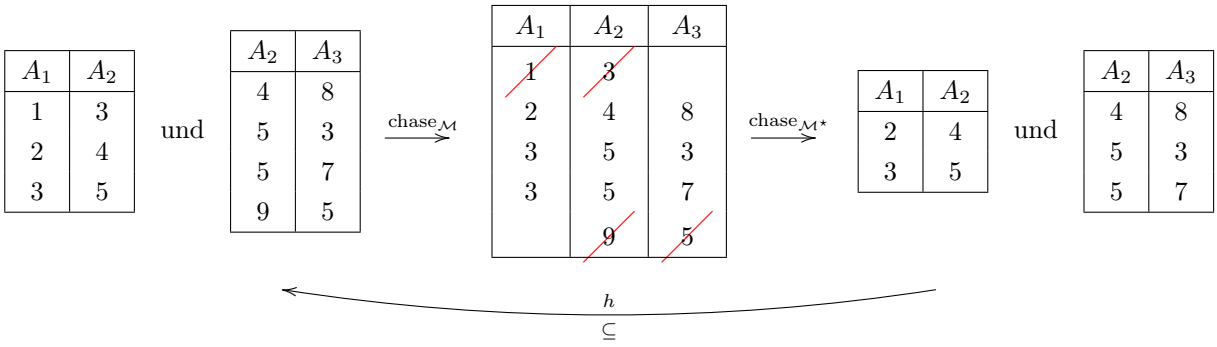


Tabelle 4.8. Zweites Beispiel für eine inverse Schemaabbildung der Operation $\bowtie$

Selektion:

Bei der Auswahl bestimmter Tupel müssen die Konstantenselektion $\sigma_{A_i\theta c}$ sowie die Atrributselektion $\sigma_{A_i\theta A_j}$ unterschieden werden. Dabei ist c ein konstanter Wert und $\theta \in \{<, \leq, =, \geq, >\}$. Vergleiche hierzu auch die Definitionen 2.8 und 2.9 aus dem Grundlagenkapitel. Die Selektion auf $\neq$ wird bei unseren Untersuchungen vernachlässigt, da der CHASE nur für positive relationale Algebren definiert ist. Wir können also zunächst keine CHASE-inverse Abbildung angeben.

Operation:	$\sigma_{A_i\theta c}(r(\mathcal{R}))$
Inverse Abbildung:	Ergebnisäquivalente CHASE-Inverse
Probleme / Schwierigkeiten:	Keine

Im Folgenden werden zwei Wahlen von θ unterschieden. Zum einen für die Konstanten- und zum anderen für die Attributselektion. Für eine gegebene Schemaabbildung $\mathcal{M} = \sigma_{A_i\theta c}(r(\mathcal{R}))$ bzw. $\sigma_{A_i\theta A_j}(r(\mathcal{R}))$ kann für jede Wahl von θ eine eindeutige ergebnisäquivalente CHASE-Inverse angegeben werden. Da bei der

Selektion per Definition Tupel verloren gehen, beschreibt die Abbildung h jeweils nur eine Teilmengenbeziehung. Sie überführt die Urinstanz I'' nicht komplett in die Quellinstanz I zurück, sodass eine „stärkere“ inverse Abbildung wie die (exakte) CHASE-Inverse nicht existiert.

Die Konstanten-Selektion löscht alle Tupel, deren Attributwerte a_{ij} (Wert des Attributes A_i im j-ten Tupel) nicht in einer bestimmten Ordnung zu einer Konstanten c stehen. Dies bedeutet für die folgenden beiden Beispiele den Test auf Gleichheit (siehe Tabelle 4.9) bzw. auf eine <-Beziehung (siehe Tabelle 4.10). Für die anderen Wahlen von θ kann analog vorgegangen werden.

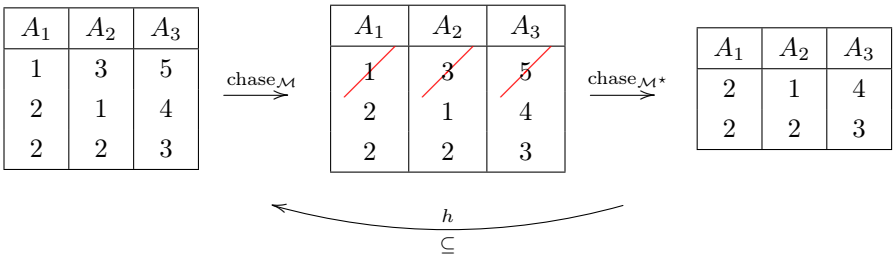


Tabelle 4.9. Beispiel für eine inverse Schemaabbildung der Operation $\sigma_{A_1=2}$

In der CHASE-Phase geht jeweils das erste Tupel $(1, 3, 5)$ verloren. Sie sind in den zugehörigen Tabellen rot markiert. Die BACKCHASE-Phase kann dieses Tupel nicht zurückgewinnen, sodass die Urinstanz $I'' = \{(2, 1, 4), (2, 2, 3)\}$ eine Teilmenge der Quellinstanz $I = \{(1, 3, 5), (2, 1, 4), (2, 2, 3)\}$ ist. Es folgt für die Schemaabbildungen $\mathcal{M} = \sigma_{A_1=2}(r(\{A_1, A_2, A_3\}))$ bzw. $\mathcal{M} = \sigma_{A_2<3}(r(\{A_1, A_2, A_3\}))$ die Existenz einer ergebnisäquivalenten CHASE-Inversen $\mathcal{M}^\star$.

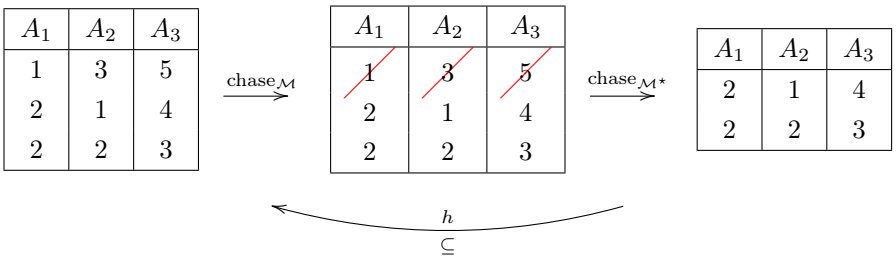


Tabelle 4.10. Zweites Beispiel für eine inverse Schemaabbildung der Operation $\sigma_{A_2<3}$

Operation:	$\sigma_{A_i \theta A_j}(r(\mathcal{R}))$
Inverse Abbildung:	Ergebnisäquivalente CHASE-Inverse
Probleme / Schwierigkeiten:	Keine

Die Attribut-Selektion löscht alle Tupel, deren Attributwerte a_{ij} nicht in einer bestimmten Ordnung zu den Werten eines anderen Attributs A_j stehen. Dies bedeutet für die folgenden beiden Beispiele den Test auf Gleichheit (siehe Tabelle 4.11) bzw. auf eine <-Beziehung (siehe Tabelle 4.12). Für die anderen Wahlen von θ kann analog vorgegangen werden.

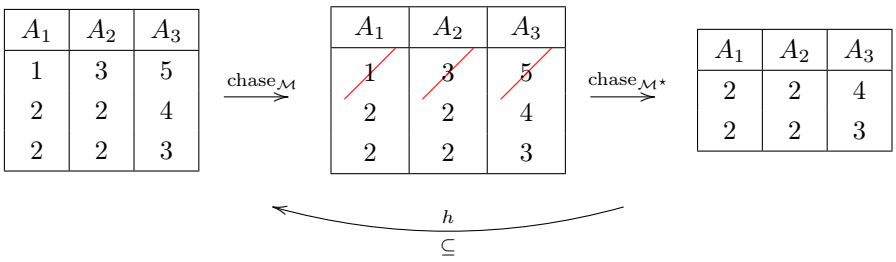


Tabelle 4.11. Beispiel für eine inverse Schemaabbildung der Operation $\sigma_{A_1=A_2}$

Im ersten Beispiel (siehe 4.11) entfällt für die Schemaabbildung $\mathcal{M} = \sigma_{A_1=A_2}(r(\{A_1, A_2, A_3\}))$ in der CHASE-Phase das Tupel $(1, 3, 5)$. Im zweiten Beispiel (siehe Tabelle 4.12) entfällt bei gegebener Schemaabbildung $\mathcal{M} = \sigma_{A_1<A_2}(r(\{A_1, A_2, A_3\}))$ das Tupel $(2, 2, 4)$. Wie bei der Konstanten-Selektion kann dieses verlorene Tupel in der BACKCHASE-Phase nicht wiedergewonnen werden, sodass die Abbildung h wiederum lediglich eine Teilmengenbeziehung beschreibt.

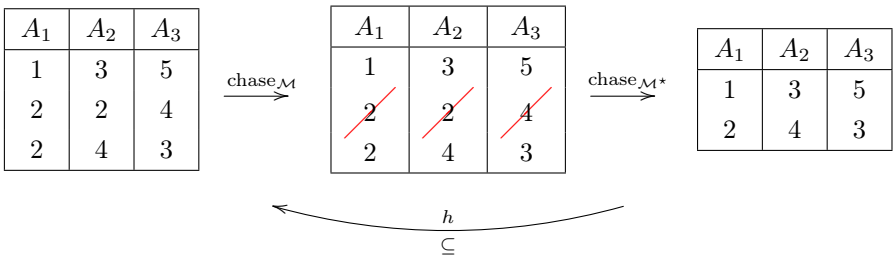


Tabelle 4.12. Zweites Beispiel für eine inverse Schemaabbildung der Operation $\sigma_{A_1<A_2}$

Mengenoperationen:

Zu den Mengenoperationen gehören die Vereinigung, der Durchschnitt sowie die Differenz. In allen drei Fällen kann für die jeweilige Schemaabbildung $\mathcal{M}$ eine ergebnisäquivalente CHASE-Inverse $\mathcal{M}^\star$ angegeben werden.

Operation:	$r_1(\mathcal{R}_1) \cup r_2(\mathcal{R}_2)$
Inverse Abbildung:	Ergebnisäquivalente CHASE-Inverse
Problem:	Duplikate

Durch die Vereinigung der Quellinstanzen entsteht in der CHASE-Phase zwar eine komplette Ergebnisinstanz I' — das Problem der Duplikatbildung ist in diesem Fall also irrelevant—, jedoch kann diese in der BACKCHASE-Phase nicht mehr eindeutig getrennt werden. So wären die Urinstanzen I_1'' und I_2'' lediglich Kopien der Ergebnisinstanz I'. Die resultierenden Urinstanzen enthalten die Quellinstanzen, sodass eine relaxte CHASE-Inverse nicht angegeben werden kann. Es gilt aber dennoch $\text{chase}_{\mathcal{M}}(I_1, I_2) = \text{chase}_{\mathcal{M}}(I_1'', I_2'')$.

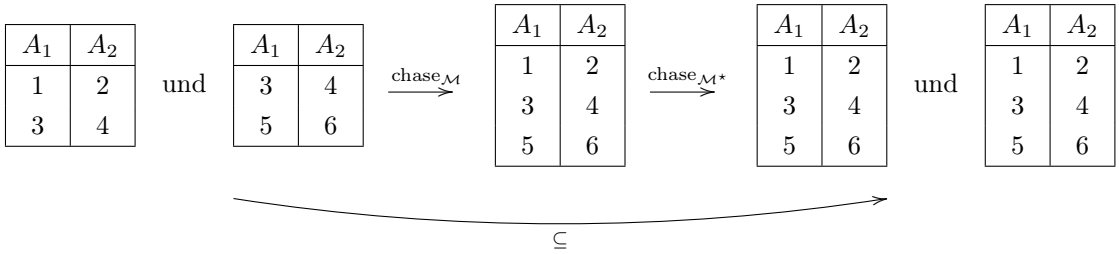


Tabelle 4.13. Beispiel für eine inverse Schemaabbildung der Operation $\cup$

Tabelle 4.13 verdeutlicht dies für zwei gegebene Quellinstanzen $I_1 = \{(1,2),(3,4)\}$ und $I_2 = \{(3,4),(5,6)\}$. Die resultierenden Urinstanzen $I_1'' = I_2'' = \{(1,2),(3,4),(5,6)\}$ sind somit Obermengen der Quellinstanzen und Homomorphismus h beschreibt eine Teilmengenbeziehung von I_1 und I_2 in I_1'' sowie I_2''.

Obermengen der Originalinstanzen sind eine negative Eigenschaft, da sie falsche Originaldaten vorspiegeln. Wir müssen daher wahrscheinlich einen weiteren Homomorphismus von den invers berechneten Originaldaten I'' in die echten Originaldaten I fordern. Dies wollen wir hier allerdings nicht weiter untersuchen.

Operation:	$r_1(\mathcal{R}_1) \cap r_2(\mathcal{R}_2)$
Inverse Abbildung:	Ergebnisäquivalente CHASE-Inverse
Probleme / Schwierigkeiten:	Keine

Die Durchschnittsbildung verhält sich ähnlich wie die Konstanten- oder Attributselektion. In der CHASE-Phase gehen Tupel verloren, die in der BACKCHASE-Phase nicht zurückgewonnen werden können. Für eine Schemaabbildung $\mathcal{M}$ existiert daher lediglich eine ergebnisäquivalente CHASE-Inverse $\mathcal{M}^\star$. Das Beispiel aus Tabelle 4.14 zeigt diesen Sachverhalt für zwei gegebene Quellinstanzen I_1 und I_2.

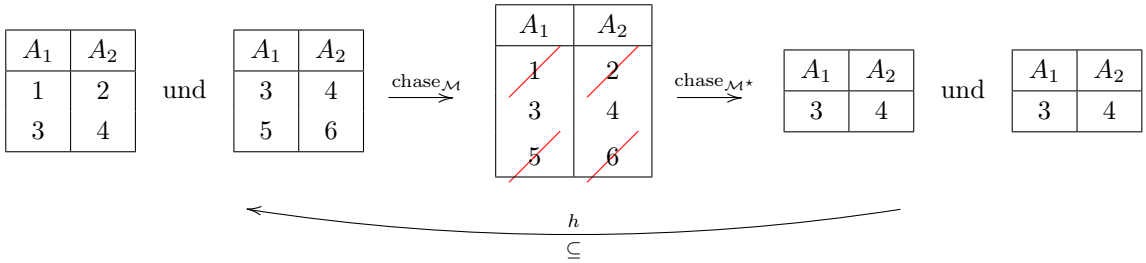


Tabelle 4.14. Beispiel für eine inverse Schemaabbildung der Operation $\cap$

Operation:	$r_1(\mathcal{R}_1) - r_2(\mathcal{R}_2)$
Inverse Abbildung:	Es existiert keine der obigen CHASE-Inversen.
Probleme / Schwierigkeiten:	Keine

Gegeben sei eine Schemaabbildung $\mathcal{M} = r_1(\mathcal{R}_1) - r_2(\mathcal{R}_2)$. Dann liefert die BACKCHASE-Phase immer eine „leere“ Urinstanz I_2''. Da die Ergebnisinstanz I' ausschließlich Tupel der Quellinstanz I_1 enthält, kann keine Aussage über die Tupel aus I_2'' gemacht werden. Sie kann lediglich mit (markierten) Nullvariablen aufgefüllt werden, sollte die Tupelanzahl der Quellinstanz I_2 bekannt sein. Für das Beispiel aus der Tabelle 4.15 gilt: Über die Urinstanz I_2'' kann keine Aussage getroffen werden. Die Urinstanz I_1'' hingegen besteht aus einer Kopie der Ergebnisinstanz I'.

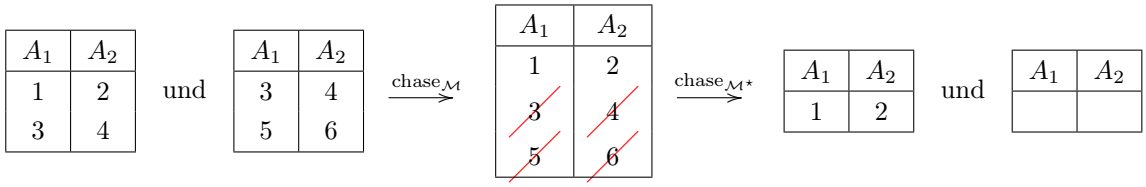


Tabelle 4.15. Beispiel für eine inverse Schemaabbildung der Operation $-$

Aggregation:

Da die Ergebnisinstanz nach der ersten CHASE-Anwendung auf die Aggregatfunktionen **COUNT**, **SUM** und **AVG** jeweils aus nur einem Parameter besteht, welcher selber nicht in der Quellinstanz enthalten ist, existieren hierfür keine der obigen inversen Abbildungen. Die Urinstanz entsteht dabei durch Kopie der einzelnen Ergebnisinstanzen. Einzig für die Funktionen **MAX** und **MIN** kann eine ergebnisäquivalente CHASE-Inverse angegeben werden.

Operation:	$\mathbf{MAX}(r(\mathcal{R}))$ bzw. $\mathbf{MIN}(r(\mathcal{R}))$
Inverse Abbildung:	Ergebnisäquivalente CHASE-Inverse
Probleme / Schwierigkeiten:	Keine

Für eine gegebene Instanz $I = \{(1,3),(2,4),(3,5)\}$ soll mittels $\mathcal{M} = \mathbf{MAX}(A_2)$ das Maximum über die Attributwerte des zweiten Attributs berechnet werden (siehe Tabelle 4.16). Die CHASE-Phase liefert den Wert 5, welcher per Definition der Schemaabbildung $\mathcal{M}$ in der Quellinstanz enthalten ist. In der Urinstanz I'' existiert somit mindestens ein Tupel, bestehend aus einer (markierten) Nullvariablen und dem Attributwert 5. Zudem gilt:

$$\text{chase}_{\mathcal{M}}(I) \leftrightarrow \text{chase}_{\mathcal{M}}(\{(n_1,5)\}).$$

Es folgt die Existenz einer ergebnisäquivalenten CHASE-Inversen $\mathcal{M}^\star$.

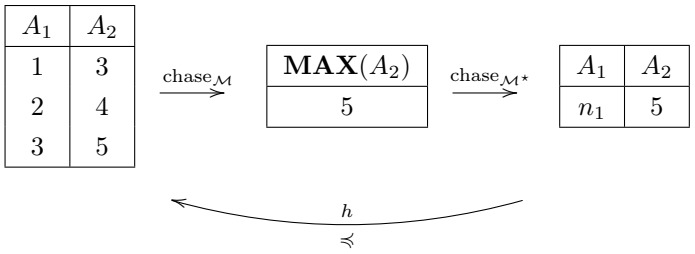


Tabelle 4.16. Beispiel für eine inverse Schemaabbildung der Operation **MAX**

Operation:	$\mathbf{COUNT}(r(\mathcal{R}))$
Inverse Abbildung:	Relaxte CHASE-Inverse
Probleme / Schwierigkeiten:	Keine

Für eine gegebene Quellinstanz $I = \{(1,3),(2,4),(3,5)\}$ soll die Anzahl ihrer Tupeln gezählt werden (siehe Tabelle 4.16). Die CHASE- sowie die BACKCHASE-Phase liefern den Wert 3 bzw. drei Nulltupel der Art (n_i, n_j). Die Urinstanz I'' ist somit $\preccurlyeq$-Menge der Quellinstanz I und es existiert eine relaxte CHASE-inverse Funktion für die Aufzählung **COUNT**.

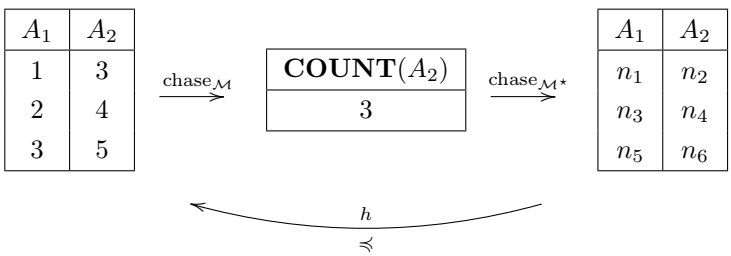


Tabelle 4.17. Beispiel für eine inverse Schemaabbildung der Operation **COUNT**

Operation:	$\mathbf{SUM}(r(\mathcal{R}))$
Inverse Abbildung:	Es existiert keine der obigen CHASE-Inversen.
Probleme / Schwierigkeiten:	Keine

Die Untersuchung der **SUM**-Operation liefert ähnliche Ergebnisse wie zuvor die Analyse der **COUNT**-Operation. Eine CHASE-Inverse existiert unabhängig von der Wahl der Quellinstanz nicht. Vergleiche hierzu das Beispiel aus Tabelle 4.18. Über die Anzahl der Nulltupel kann allerdings keine Aussage getroffen werden.

A_1	A_2
1	3
2	4
3	5

$\xrightarrow{\text{chase}_{\mathcal{M}}}$

$\mathbf{SUM}(A_2)$
12

$\xrightarrow{\text{chase}_{\mathcal{M}^\star}}$

A_1	A_2
n_1	n_2

Tabelle 4.18. Beispiel für eine inverse Schemaabbildung der Operation **SUM**

Operation:	$\mathbf{AVG}(r(\mathcal{R}))$
Inverse Abbildung:	Es existiert keine der obigen CHASE-Inversen.
Probleme / Schwierigkeiten:	Keine

Der arithmetische Durchschnitt entsteht durch Kombination der Operationen **COUNT** und **SUM**. Da für diese beiden Funktionen bereits keine inverse CHASE-Abbildung existiert, gibt es auch im Falle der **AVG**-Operation keine der obigen CHASE-Inversen. Die folgende Tabelle 4.19 verdeutlicht dies für die obige Quellinstanz $I = \{(1,3),(2,4),(3,5)\}$.

A_1	A_2
1	3
2	4
3	5

$\xrightarrow{\text{chase}_{\mathcal{M}}}$

$\mathbf{AVG}(A_2)$
4

$\xrightarrow{\text{chase}_{\mathcal{M}^\star}}$

A_1	A_2
n_1	n_2

Tabelle 4.19. Beispiel für eine inverse Schemaabbildung der Operation **AVG**

Gruppierung:

Für die Gruppierung existieren ergebnisäquivalente sowie relaxte CHASE-Inverse. Die Kombination der γ-Operation mit der Maximum- bzw. Minimumbildung liefert eine ergebnisäquivalente CHASE-Inverse. Zwar werden alle nicht maximalen bzw. minimalen Tupel herausgestrichen, doch bleibt für jede Gruppe genau ein Tupel erhalten. Die Urinstanz ist somit eine Teilmenge der Quellinstanz. Ein erneuter CHASE der beiden Instanzen liefert zudem erneut die Ergebnisinstanz I'. Gleiches gilt für die Summation. Die Zählung **COUNT** liefert die richtige Anzahl an Tupeln, allerdings ist über den Inhalt der Tupel nicht bekannt. Sie liefert daher die Existenz einer relaxten CHASE-Inversen.

Operation:	$\gamma_{G_i;F_j(A_j)}(r(\mathcal{R}))$
Inverse Abbildung:	Es existiert keine der obigen CHASE-Inversen. / Ergebnisäquivalente CHASE-Inverse / Relaxte CHASE-Inverse
Probleme / Schwierigkeiten:	Keine

Aus dem CHASE einer gegebenen Quellinstanz $I = \{(2,1),(2,5),(3,4),(4,3),(4,7)\}$ bzgl. der Schemaabbildung $\mathcal{M} = \gamma_{A_1;\mathbf{MAX}(A_2)}r(\{A_1, A_2\})$ resultiert die Streichung aller nicht bzgl. A_3 maximalen Tupel, d.h. $(2,1)$ und $(4,3)$. Die so erhaltene Ergebnisinstanz I' ist daher Teilmenge von I. In der folgenden BACKCHASE-Phase wird diese Instanz lediglich kopiert und durch wegselektierte Attributwerte erweitert, wodurch die Urinstanz ebenfalls Teilmenge der Quellinstanz ist. Mit anderen Worten: $I'' = I' \subseteq I$. Siehe hierzu auch die Tabelle 4.20.

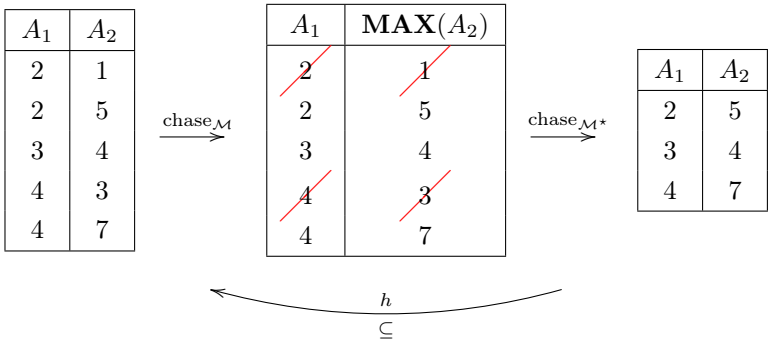


Tabelle 4.20. Beispiel für eine inverse Schemaabbildung der Operation $\gamma_{A_1;\mathbf{MAX}(A_2)}r(\{A_1, A_2\})$

Aufgrund der sehr starken Verdichtung im Attribut A_j der Operation $\mathcal{M} = \gamma_{A_i;F_j(A_j)}r(\mathcal{R})$ mit $F(x) \in \{\mathbf{SUM}, \mathbf{AVG}\}$ existieren in der BACKCHASE-Phase keine für eine Rekonstruktion der Quellinstanz notwendigen Informationen. Sie liefert im Attribut A_j lediglich (markierte) Nullwerte. Die Attributwerte des Attributes A_i bleiben jedoch erhalten. Für die Summation ist die Nicht-Existenz einer CHASE-inversen Abbildung in der folgenden Tabelle 4.21 dargestellt.

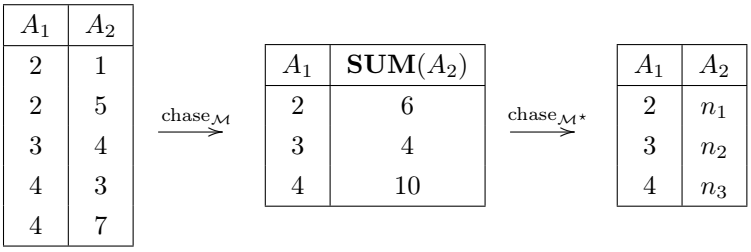


Tabelle 4.21. Zweites Beispiel für eine inverse Schemaabbildung der Operation $\gamma_{A_1;\mathbf{SUM}(A_2)}r(\{A_1, A_2\})$

Arithmetische Operationen:

Operation:	$r(\mathcal{R})\ \theta\ \alpha$ mit $\theta \in \{+,-,\cdot,:\}$ und $\alpha \in \mathbb{R}$
Inverse Abbildung:	Exakte CHASE-Inverse / Es existiert keine der obigen CHASE-Inversen.
Probleme / Schwierigkeiten:	Verlassen der Attributdomänen

Die Existenz einer CHASE-inversen Abbildung hängt von der Domäne $\mathbb{D}$ der Quellrelation bzw. des Quellattributes ab. Handelt es sich hierbei um einen der Körper $\mathbb{Q}$ oder $\mathbb{R}$, so existiert für die vier arith-

metischen Operationen + die exakte CHASE-Inverse − und umgekehrt. Gleiches gilt für die Operationen · und :. Für $\mathbb{D} \in \{\mathbb{Z}, \mathbb{N}\}$ ist die Division exakte CHASE-Inverse der Multiplikation, aber nicht umgekehrt, da die Division auf den ganzen Zahlen nicht abgeschlossen ist. Entspricht die Domäne den natürlichen Zahlen, ergibt sich für die Subtraktion ein analoges Problem. Es existiert also keine exakte CHASE-Inverse für die Subtraktion, sehr wohl aber für die Addition. In der Praxis machen endliche Wertebereiche und Rundungsfehler häufig jedoch alle theoretischen Überlegungen zunichte.

Zusammenfassend können wir sagen, sind die arithmetischen Operationen auf die Quellrelation anwendbar, so existiert immer eine exakte CHASE-Inverse. Dabei sind + und − sowie · und : zueinander invers. Die folgende Tabelle verdeutlicht dies am Beispiel der Multiplikation mit 2:

A_1	A_2	A_3
1	3	5
2	1	4
2	2	3

$\xrightarrow{\text{chase}_{\mathcal{M}}}$

A_1	A_2	A_3
2	6	10
4	2	8
4	4	6

$\xrightarrow{\text{chase}_{\mathcal{M}^\star}}$

A_1	A_2	A_3
1	3	5
2	1	4
2	2	3

h

Tabelle 4.22. Beispiel für eine inverse Schemaabbildung der arithmetischen Operation +

Bei der Modulo-Rechnung gehen sehr viele Informationen verloren. Es kann einzig eine ergebnisäquivalente CHASE-Inverse angegeben werden. Diese besteht wie zuvor aus der reinen Identitätsabbildung, sodass für die Quellinstanz I sowie die Urinstanz I'' gilt: $\text{chase}_{\mathcal{M}}(I) \leftrightarrow \text{chase}_{\mathcal{M}^\star}(I'')$. Vergleiche hierzu das Beispiel aus Tabelle 4.23.

Operation:	$r(\mathcal{R})$ mod α mit $\alpha \in \mathbb{R}$
Inverse Abbildung:	Ergebnisäquivalente CHASE-Inverse
Probleme / Schwierigkeiten:	Keine

A_1	A_2	A_3
1	3	5
2	1	4
2	2	3

$\xrightarrow{\text{chase}_{\mathcal{M}}}$

A_1	A_2	A_3
1	0	1
2	2	2
2	1	0

$\xrightarrow{\text{chase}_{\mathcal{M}^\star}}$

A_1	A_2	A_3
1	0	2
2	1	1
2	2	0

Tabelle 4.23. Beispiel für eine inverse Schemaabbildung der arithmetischen Operation **MOD** 3

4.2.5. CHASE-inverse Schemaabbildungen mit Provenance-Informationen

Versehen wir die inversen Schemaabbildungen aus Unterabschnitt 4.2.4 zusätzlich mit Provenance-Informationen, können einige relaxte und ergebnisäquivalente CHASE-Inverse auf (exakte) CHASE-Inverse erweitert werden. Hierzu versehen wir jedes Tupel der Quellinstanz I mit einem Identifikator (in den folgenden Beispielen: t_i oder s_i mit Laufindex $i \in \{1, 2, 3, 4\}$) und notieren zu jedem Tupel der Ergebnisinstanz I' das zugehörige Provenance-Polynom (vgl. Abschnitt 3.2). Anschließend bestimmen wir die Urinstanz I'' unter Berücksichtigung dieser Provenance. Die aus den Provenance-Polynomen der Ergebnisinstanz bestimmte (minimale) Zeugenbasis gibt zudem an, welche Tupel (eingeschränkt auf die relevanten Attribute) wir uns für die Rekonstruktion merken müssen. Welche zusätzlichen Informationen für die Existenz einer exakten CHASE-Inversen von Nöten ist, soll in diesem Unterabschnitt jedoch nicht interessieren. Hiermit beschäftigen wir uns im Unterabschnitt 4.2.6.

Kopieren / Identitätsabbildung:

Operation:	$r(\mathcal{R})$
Inverse Abbildung:	Exakte CHASE-Inverse
Probleme / Schwierigkeiten:	Keine

Bei der Kopieroperation können zusätzliche Provenance-Informationen vernachlässigt werden. Grund hierfür ist die Tatsache, dass auch ohne eine Provenance-Untersuchung bereits eine exakte CHASE-Inverse angegeben werden kann. Für die Kopierabbildung $\mathcal{M} = \mathrm{Id}$ ist $\mathcal{M}^\star = \mathrm{Id}$ die eindeutige inverse Abbildung, denn

$$\mathrm{chase}_{\mathcal{M}^\star}(\mathrm{chase}_{\mathcal{M}}(I)) = \mathrm{chase}_{\mathcal{M}^\star}(I) = I.$$

Für das Beispiel aus Tabelle 4.25 ergibt sich daher:

A_1	A_2	A_3		$\xrightarrow{\mathrm{chase}_{\mathcal{M}}}$	A_1	A_2	A_3		$\xrightarrow{\mathrm{chase}_{\mathcal{M}^\star}}$	A_1	A_2	A_3	
1	3	5	t_1		1	3	5	t_1		1	3	5	t_1
2	1	4	t_2		2	1	4	t_2		2	1	4	t_2
2	2	3	t_3		2	2	3	t_3		2	2	3	t_3

h

Tabelle 4.24. Beispiel für eine inverse Schemaabbildung mit Provenance-Informationen der Operation Kopieren

Da Quell- und Urinstanz übereinstimmen, entspricht h der Identitätsabbildung. Die Nicht-Notwen-digkeit zusätzlicher Provenance-Informationen sei durch das „Wegstreichen“ symbolisiert.

Umbenennung:

Operation:	$\beta_{A_j \leftarrow A_i} r(\mathcal{R})$
Inverse Abbildung:	Exakte CHASE-Inverse
Probleme / Schwierigkeiten:	Keine

Da für die Umbenennung bereits ohne zusätzliche Informationen eine exakte CHASE-Inverese angegeben werden kann, können Provenance-Informationen hier vernachlässigt werden. Betrachten wir hierzu eine Quellinstanz I, ein Quellschema $\mathcal{R} = \{A_1, A_2, A_3\}$ sowie eine Schemaabbildung $\mathcal{M} = \beta_{A_1A_2A_3 \leftarrow A_4A_5A_6} r(\{A_1, A_2, A_3\})$. Nach der Umbenennung und anschließenden Rückbenennung gilt weiterhin, dass Quell- und Urinstanz übereinstimmen.

A_1	A_2	A_3		$\xrightarrow{\mathrm{chase}_{\mathcal{M}}}$	A_4	A_5	A_6		$\xrightarrow{\mathrm{chase}_{\mathcal{M}^\star}}$	A_1	A_2	A_3	
1	3	5	t_1		1	3	5	$t_1 \circ \beta$		1	3	5	t_1
2	1	4	t_2		2	1	4	$t_2 \circ \beta$		2	1	4	t_2
2	2	3	t_3		2	2	3	$t_3 \circ \beta$		2	2	3	t_3

h

Tabelle 4.25. Beispiel für eine inverse Schemaabbildung der Operation Kopieren

Projektion:

Wegen der Bildung von Duplikaten mussten für die Projektion im Unterabschnitt 4.2.4 zwei Fälle unterschieden werden. Merkt man sich bei der CHASE-Anwendung zusätzliche Provenance-Informationen, wird diese Differenzierung jedoch hinfällig, denn mit Hilfe der Provenance-Polynome können Duplikate erfasst und wieder „eingefügt“ werden. Die folgenden beiden Beispiele (Erweiterungen der Tabellen 4.5 und 4.6) sollen dies verdeutlichen.

Operation:	$\pi_{A_i}(r(\mathcal{R}))$
Inverse Abbildung:	Relaxte CHASE-Inverse
Probleme / Schwierigkeiten:	Duplikate

Da die Untersuchung der Tabelle 4.5 bereits die Existenz einer relaxten CHASE-Inverse gezeigt hat, ist eine Erweiterung um den Aspekt der Provenance in diesem Fall nicht notwendig. Die zusätzlichen Informationen sind in der folgenden Tabelle 4.26 wieder herausgestrichen. Sollten sie dennoch verwendet werden, ergeben sich die minimale Zeugenbasis $\{\{t_1\},\{t_2\},\{t_3\}\}$ sowie die Provenance-Polynome t_1, t_2 und t_3.

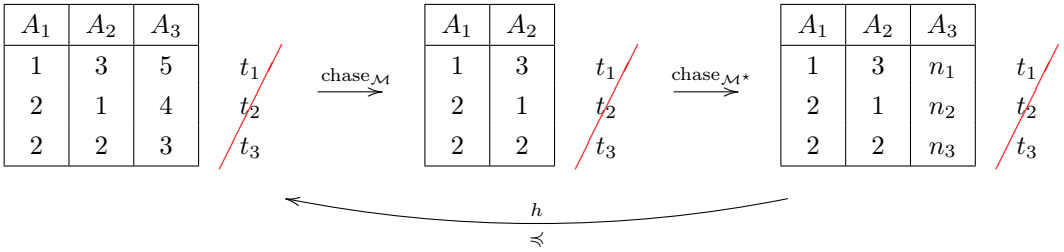


Tabelle 4.26. Beispiel für eine inverse Schemaabbildung mit Provenance-Informationen der Operation π

Betrachten wir nun das Beispiel aus Tabelle 4.27. Bei gegebener Quellinstanz $I = \{(1,3,5),(2,1,4),(2,1,3)\}$ entfällt in der CHASE-Phase das Duplikat $(2,1)$. Die Ergebnisinstanz I' enthält somit nur noch zwei Tupel $(1,3)$ und $(2,1)$. Durch Hinzunahme zusätzlicher Provenance-Informationen ergeben sich für diese Tupel die Polynome t_1 und $t_2 +_K t_3$. Die hieraus abgeleitete minimale Zeugenbasis $\{\{t_1\},\{t_2\}\}$ beinhaltet die zu merkenden Quelltupel t_1 und t_2; die Provenance-Polynome beschreiben Regeln der „vollständiger“ Rück-Projektion.

Die minimale Zeugenbasis ergibt sich hierbei aus der Zeugenbasis $\{\{t_1\},\{t_2,t_3\}\}$ durch Streichen der Duplikate. Die Wahl von t_2 bzw. t_3 ist dabei nicht festgelegt. Üblicherweise wird jedoch das Tupel mit der kleinsten ID, sprich t_2, gewählt. Die Rück-Projektion in der BACKCHASE-Phase liefert schließlich die um das verlorene Duplikat erweiterte Urinstanz $I'' = \{(1,3,n_1),(2,1,n_2),(2,1,n_3)\}$. Es gilt somit $I'' \preccurlyeq I$, was die Existenz einer relaxten CHASE-Inversen $\mathcal{M}^\star$ von $\mathcal{M} = \pi_{A_1A_2}(r(\{A_1,A_2,A_3\}))$ zur Folge hat.

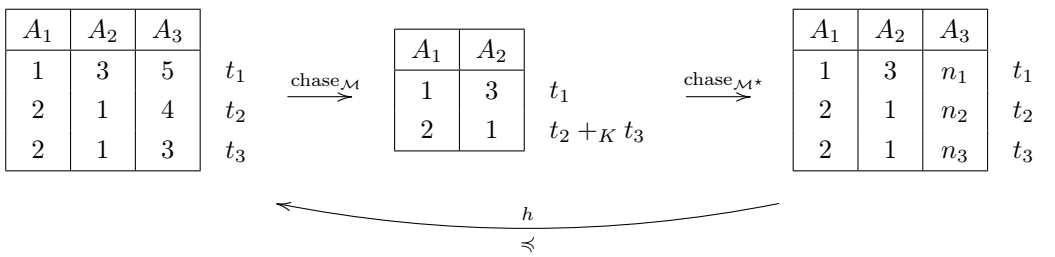


Tabelle 4.27. Zweites Beispiel für eine inverse Schemaabbildung mit Provenance-Informationen der Operation π

Natürlicher Verbund:

Operation:	$r_1(\mathcal{R}_1) \bowtie r_2(\mathcal{R}_2)$
Inverse Abbildung:	Exakte CHASE-Inverse / Ergebnisäquivalente CHASE-Inverse
Probleme / Schwierigkeiten:	Dangling tuples

Für die Schemaabbildung $\mathcal{M} = r_1(\mathcal{R}_1) \bowtie r_2(\mathcal{R}_2)$ ist eine Provenance-Analyse nicht notwendig. Trotz der zusätzlichen Informationen können keine „besseren“ CHASE-Inversen angegeben werden. Wann immer bereits ohne Provenance-Untersuchungen eine exakte CHASE-Inverse angegeben werden kann, können diese vernachlässigt werden (siehe Tabelle 4.28).

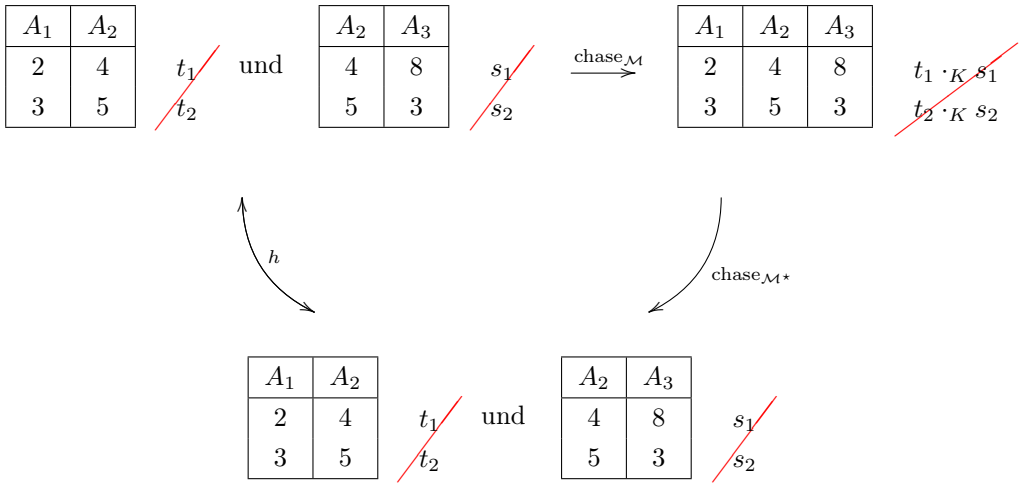


Tabelle 4.28. Beispiel für eine inverse Schemaabbildung mit Provenance-Informationen der Operation $\bowtie$

Treten zudem dangling tuples auf, so gehen sie bereits bei der ersten CHASE-Anwendung verloren. Es können also keine Zeugenbasen oder Polynome aufgestellt werden, die diese verlorenen Tupel „retten“ oder zurückgewinnen könnten. Die Ergebnistabellen der Tabelle 4.29 stimmen daher mit denen aus Tabelle 4.8 überein. Die Nichtnotwendigkeit der Provenance-Informationen ist in beiden Beispielen durch rotes Durchstreichen symbolisiert.

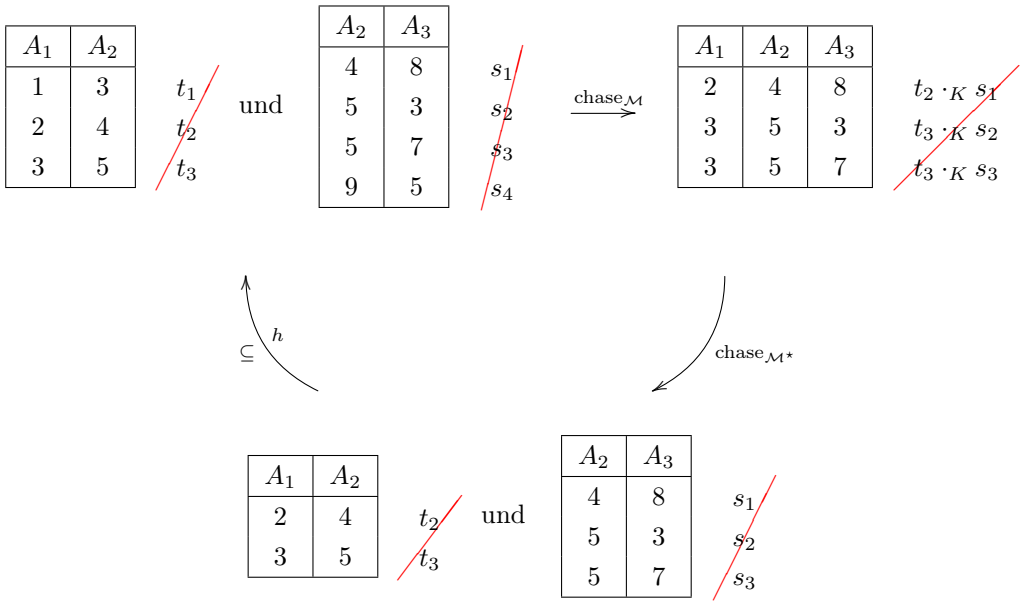


Tabelle 4.29. Zweites Beispiel für eine inverse Schemaabbildung mit Provenance-Informationen der Operation $\bowtie$

Selektion:

Da bei der Selektion ganze Tupel gestrichen werden, liefert auch hier die zusätzliche Provenance-Berechnung keine zusätzlichen Informationen. Dies gilt sowohl für die Konstanten- als auch für die Attribut-Selektion sowie alle Wahlen von $\theta \in \{<, \leq, =, \geq, >\}$. Die Selektion auf Ungleichheit wird wegen der Schwierigkeiten bei der Aufstellung der zugehörigen Provenance-Polynome hier nicht weiter untersucht.

Operation:	$\sigma_{A_i \theta c}(r(\mathcal{R}))$
Inverse Abbildung:	Ergebnisäquivalente CHASE-Inverse
Probleme / Schwierigkeiten:	Keine

Für das Beispiel aus Tabelle 4.30 ergibt sich somit analog zur Tabelle 4.9 eine ergebnisäquivalente CHASE-Inverse. Dargestellt ist hier eine Konstanten-Selektion des ersten Attributs durch $\mathcal{M} = \sigma_{A_1=2}(r(\{A_1, A_2, A_3\}))$. Die zusätzlichen Provenance-Informationen sind nicht von Nöten und daher rot durchgestrichen.

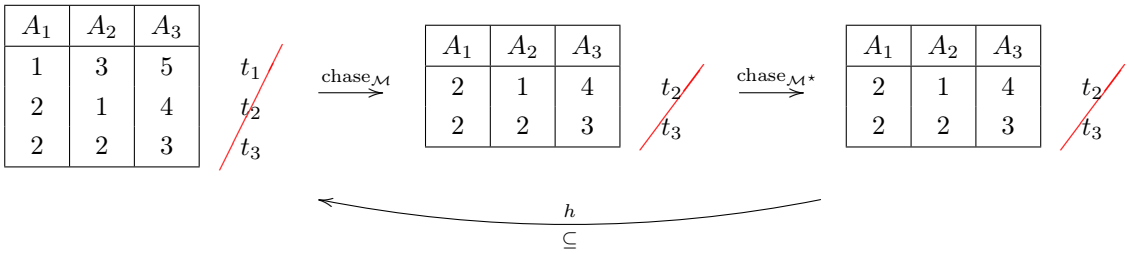


Tabelle 4.30. Beispiel für eine inverse Schemaabbildung mit Provenance-Informationen der Operation $\sigma_{A_1=2}$

Operation:	$\sigma_{A_i \theta A_j}(r(\mathcal{R}))$
Inverse Abbildung:	Ergebnisäquivalente CHASE-Inverse
Probleme / Schwierigkeiten:	Keine

Auch im Fall der Attribut-Selektion werden durch die Einarbeitung der Provenance-Informationen keine zusätzlichen Eigenschaften gewonnen. Tupel, auf die das Selektionskriterium nicht zutrifft, verfallen bei Anwendung des ersten CHASE und können trotz Provenance nicht zurückgewonnen werden. Die folgende Tabelle 4.31 verdeutlicht dies für eine Schemaabbildung $\mathcal{M} = \sigma_{A_1=A_2}(r(\{A_1, A_2, A_3\}))$. Es gelten $I'' \subseteq I$ sowie $\text{chase}_{\mathcal{M}}(I) \leftrightarrow \text{chase}_{\mathcal{M}}(I'')$, sodass analog zu Tabelle 4.10 eine ergebnisäquivalente CHASE-Inverse $\mathcal{M}^\star$ existiert. Welche weiteren Informationen für die Existenz einer exakten CHASE-Inversen notwendig sind, werden im späteren Verlauf der Arbeit noch diskutiert.

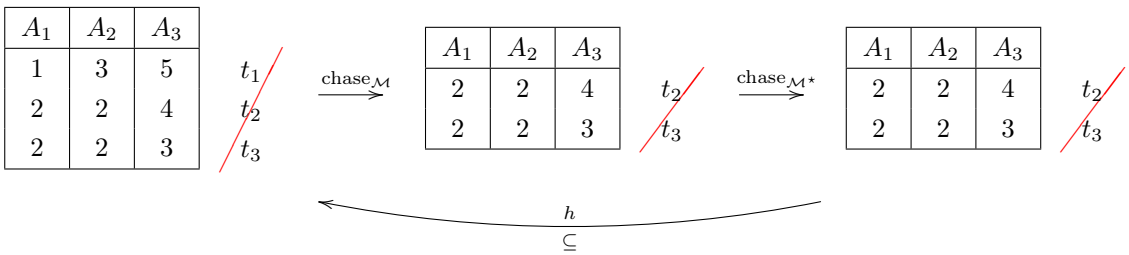


Tabelle 4.31. Beispiel für eine inverse Schemaabbildung mit Provenance-Informationen der Operation $\sigma_{A_1=A_2}$

Mengenoperationen:

Bei den Mengenoperationen gelingt bei der Vereinigung die Angabe einer exakten CHASE-Inversen, sollten zusätzliche Provenance-Informationen beachtet werden. Für den Durchschnitt sowie die Differenz kann keine andere inverse Abbildung gefunden werden. Es bleibt bei einer ergebnisäquivalenten CHASE-Inverse.

Operation:	$r_1(\mathcal{R}_1) \cup r_2(\mathcal{R}_2)$
Inverse Abbildung:	Exakte CHASE-Inverse
Problem / Schwierigkeiten:	Duplikate

Die in der CHASE-Phase aufgestellten Provenance-Polynome erhalten die Ursprungsorte der Duplikate. Zusammen mit der (minimalen) Zeugenbasis können sie so in den Urinstanzen rekonstruiert und eine eindeutige exakte CHASE-Inverse angegeben werden.

Das ist Tabelle 4.32 vorgestellte Beispiel basiert auf dem Beispiel aus Tabelle 4.13. Bei gleichen Quellinstanzen I_1 und I_2 ergeben sich hier für die Urinstanzen I_1'' und I_2'' statt der Vereinigung beider Quellinstanzen die Quellinstanzen selbst. Mit anderen Worten $I_1'' = I_1$ und $I_2'' = I_2$ anstelle von $I_1'' = I_1 \cup I_2 = I_2''$. Es gilt somit

$$\text{chase}_{\mathcal{M}^\star}(\text{chase}_{\mathcal{M}}(I_1, I_2)) = \text{chase}_{\mathcal{M}^\star}(I') = (I_1'', I_2'') = (I_1, I_2).$$

Die Polynome t_1, $t_2 +_K s_1$ und s_2 geben die Herkunft der Ergebnisinstanz-Tupel an. Die Urinstanzen bestehen somit wiederum aus den Mengen $\{t_1, t_2\}$ und $\{s_1, s_2\}$. Die Zeugenbasis $\{\{t_1\}, \{t_2, s_1\}, \{s_2\}\}$ gibt nun an, welche Tupel der Quellinstanzen für eine Rekonstruktion gespeichert werden müssen. Da es sich bei t_2 und s_1 um Duplikate handelt, bietet es sich an, sogar die minimale Zeugenbasis $\{\{t_1\}, \{t_2\}, \{s_2\}\}$ zu verwenden.

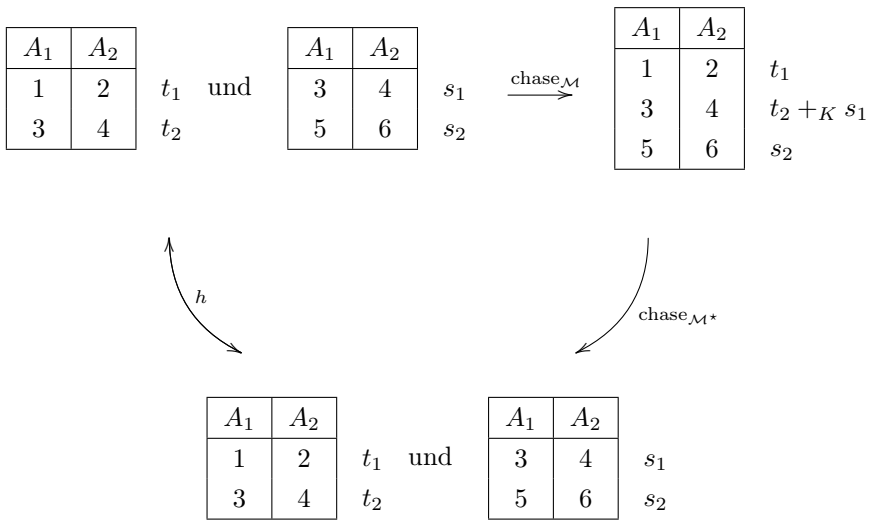


Tabelle 4.32. Beispiel für eine inverse Schemaabbildung mit Provenance-Informationen der Operation $\cup$

Operation:	$r_1(\mathcal{R}_1) \cap r_2(\mathcal{R}_2)$
Inverse Abbildung:	Ergebnisäquivalente CHASE-Inverse
Probleme / Schwierigkeiten:	Keine

Die Operation $\cap$ verhält sich ähnlich wie der natürliche Verbund und die Selektion. Es existiert stets eine ergebnisäquivalente CHASE-Inverse, unabhängig davon, ob zusätzliche Provenance-Informationen berücksichtigt werden oder nicht. Wird die Provenance aus Tabelle 4.33 wieder herausgestrichen, so ergibt sich das Beispiel aus Tabelle 4.14. Für diese haben wir den Existenznachweis der CHASE-Inversen bereits geführt.

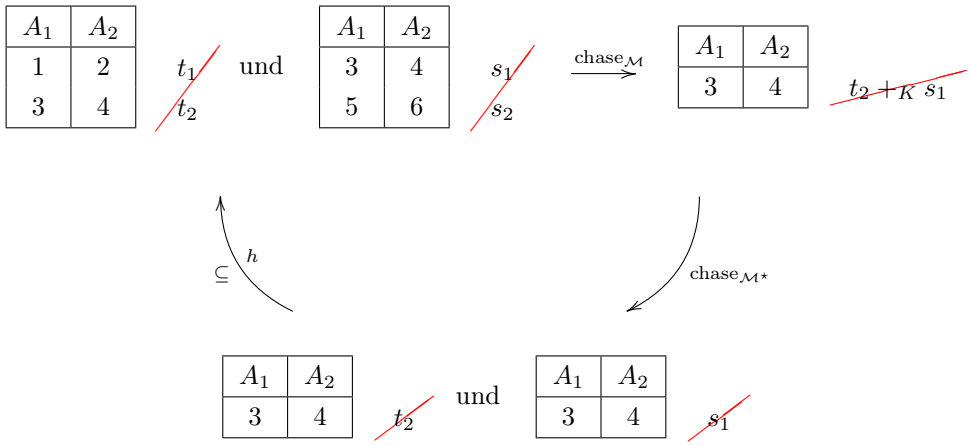


Tabelle 4.33. Beispiel für eine inverse Schemaabbildung mit Provenance-Informationen der Operation $\cap$

Operation:	$r_1(\mathcal{R}_1) - r_2(\mathcal{R}_2)$
Inverse Abbildung:	Es existiert keine der obigen CHASE-Inverse
Probleme / Schwierigkeiten:	Keine

Im Falle der Differenz $-$ gehen sämtliche Informationen der zweiten Quellinstanz I_2 bereits in der CHASE-Phase verloren. Die Einführung von Zeugenbasen und Provenance-Polynomen würden hieran vermutlich nichts ändern. Da die Anwendung des CHASE nicht gewährleistet und die Berechnung eines zugehörigen Provenance-Polynoms hier nicht untersucht wurde, soll auch die Differenz hier nicht weiter betrachtet werden. Für die Polynombestimmung sei aber auf [ADT11] verwiesen.

Aggregation:

Für die Aggregation kann bei Verwendung von Provenance-Aspekten die größte Veränderung beobachtet werden. Statt ursprünglich keiner inversen Abbildung können für die Operationen **COUNT**, **SUM** und **AVG** nun sogar exakte CHASE-Inverse angegeben werden. Im Falle der Minimum- bzw. Maximum-Bestimmung ändert sich der Typ der inversen Abbildung jedoch nicht. Es gilt weiterhin die Existenz einer ergebnisäquivalenten CHASE-Inversen.

Operation:	**MAX**$(r(\mathcal{R}))$ bzw. **MIN**$(r(\mathcal{R}))$
Inverse Abbildung:	Ergebnisäquivalente CHASE-Inverse
Probleme / Schwierigkeiten:	Keine

Weitere Provenance-Angaben können vernachlässigt werden, da über die Tupelzahl der Quellinstanz keine Aussage getroffen werden kann. Für das Beispiel der Tabelle 4.16 ergibt sich somit:

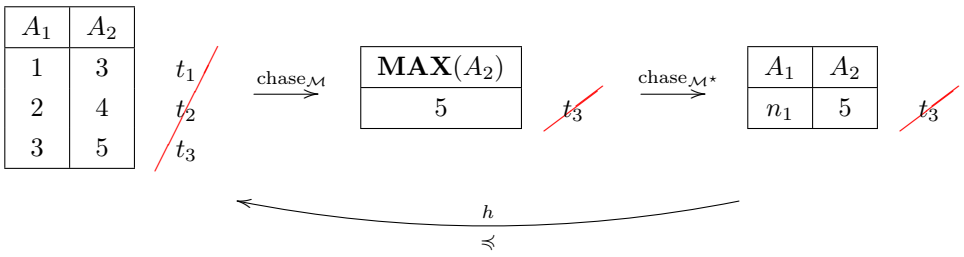


Tabelle 4.34. Beispiel für eine inverse Schemaabbildung mit Provenance-Informationen der Operation **MAX**

Operation:	**COUNT**$(r(\mathcal{R}))$
Inverse Abbildung:	Relaxte CHASE-Inverse
Probleme / Schwierigkeiten:	Keine

Die Ergebnisinstanz der **COUNT**-Operation besteht aus genau einen Wert mit dem zugehörigen Provenance-Polynom $t_1 +_K ... +_K t_n$. Die minimale Zeugenbasis entspricht somit $\{\{t_1\}, ..., \{t_n\}\}$. Für eine korrekte Rekonstruktion der Quellinstanz werden alle Tupel der Quellinstanz benötigt (siehe Tabelle 4.35). Die Tupelanzahl der Quellinstanz kann aber bereits ohne zusätzliche Provenance-Informationen angegeben werden.

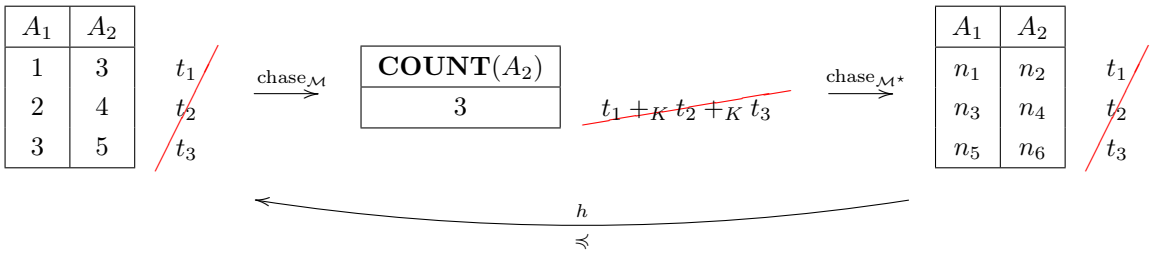


Tabelle 4.35. Beispiel für eine inverse Schemaabbildung mit Provenance-Informationen der Operation **COUNT**

Operation:	**SUM**$(r(\mathcal{R}))$
Inverse Abbildung:	Exakt CHASE-Inverse
Probleme / Schwierigkeiten:	Keine

Die Begründung für die Existenz einer exakten CHASE-Inversen im Fall der Summation erfolgt ganz analog zur Operation **COUNT**. Mit Hilfe des Provenance-Polynoms sowie der resultierenden (minimalen) Zeugenbasis lässt sich die Quellinstanz im zweiten Attribut vollständig rekonstruieren (siehe Tabelle 4.36).

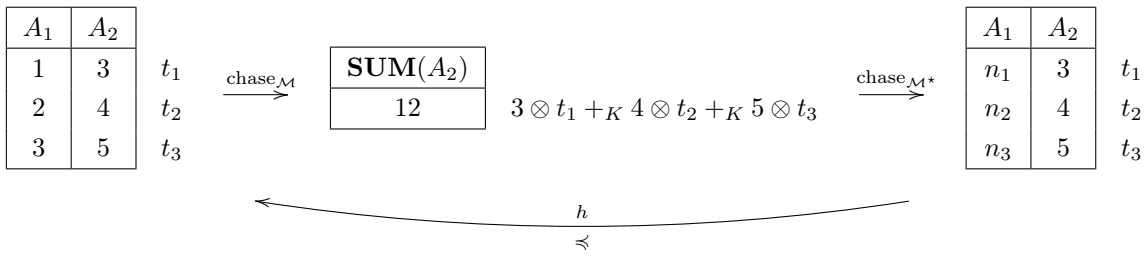


Tabelle 4.36. Beispiel für eine inverse Schemaabbildung mit Provenance-Informationen der Operation **SUM**

Operation:	$\mathbf{AVG}(r(\mathcal{R}))$
Inverse Abbildung:	Exakt CHASE-Inverse
Probleme / Schwierigkeiten:	Keine

Die Definition des Mittelwertes

$$\mathbf{AVG}(A_i) = \frac{\mathbf{SUM}(A_i)}{\mathbf{COUNT}(A_i)}$$

sowie die obigen Untersuchungen implizieren auch für diese Operation die Existenz einer exakten CHASE-Inversen. Die aus dem Polynom $\frac{a_{i,1} \otimes t_1 +_K \ldots +_K a_{i,n} \otimes t_n}{t_1 +_K \ldots +_K t_3}$ erzeugte minimale Zeugenbasis $\{\{t_1\}, ..., \{t_n\}\}$ garantiert die Existenz eines Homomorphismus $h : I'' \to I$ mit $n_1 := 1$, $n_2 := 2$ und $n_3 := 3$. Dabei ist $a_{i,j}$ der Wert des i-ten Attributes im j-ten Tupel. Die Tabelle 4.37 veranschaulicht dies für eine gegebene Quellinstanz $I = \{(1,3),(2,4),(3,5)\}$.

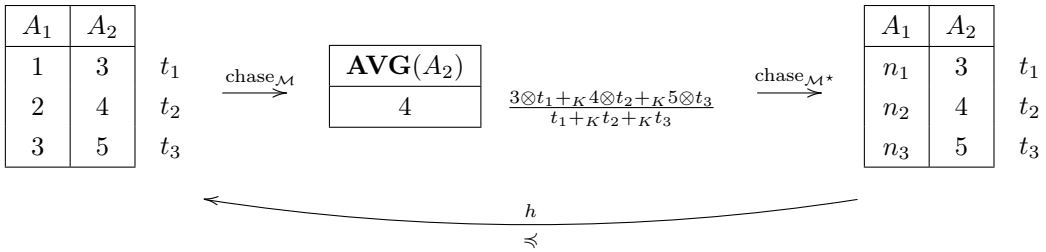


Tabelle 4.37. Beispiel für eine inverse Schemaabbildung mit Provenance-Informationen der Operation **AVG**

Gruppierung:

Operation:	$\gamma_{G_i;F_j(A_j)}(r(\mathcal{R}))$
Inverse Abbildung:	Ergebnisäquivalente CHASE-Inverse / Relaxte CHASE-Inverse / Exakte CHASE-Inverse
Probleme / Schwierigkeiten:	Keine

Gegebene seien eine Quellinstanz $I = \{(2,1),(2,5),(3,4),(4,3),(4,7)\}$ sowie eine Schemaabbildung $\mathcal{M} = \gamma_{A_1;\mathbf{MAX}(A_2)} r(\{A_1, A_2\})$ (siehe Tabelle 4.38). Dann erfolgt in der CHASE-Phase die Streichung aller bzgl. A_2 nicht maximalen Tupel. Da die Provenance-Polynome diese verloren gegangenen Informationen nicht abfangen können, kann in diesem Fall trotz zusätzlicher Provenance-Untersuchungen kein „besseres“ Ergebnis erzielt werden. Es existiert lediglich eine ergebnisäquivalente CHASE-Inverse. Vergleiche hierzu den Nachweis aus Tabelle 4.20.

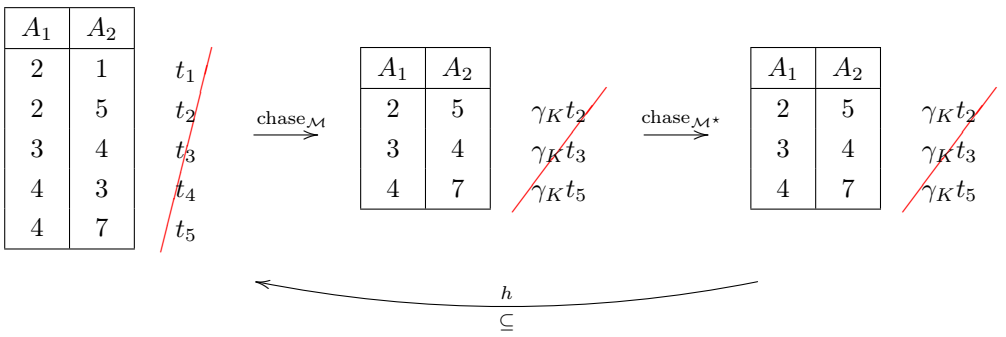


Tabelle 4.38. Beispiel für eine inverse Schemaabbildung mit Provenance-Informationen der Operation $\gamma_{A_1;\mathbf{MAX}(A_2)} r(\{A_1, A_2\})$

Da bereits für die Aufzählung bereits nach den obigen Überlegungen eine relaxte CHASE-Inverse existiert, ist dies auch für $\mathcal{M} = \gamma_{G_i;F_j(A_j)}(r(\mathcal{R}))$ mit $F(x) = \textbf{COUNT}$ der Fall.

A_1	A_2	
2	1	t_1
2	5	t_2
3	4	t_3
4	3	t_4
4	7	t_5

$\xrightarrow{\text{chase}_{\mathcal{M}}}$

A_1	A_2	
2	6	$\gamma_K(1 \otimes t_1 +_K 5 \otimes t_2)$
3	4	$\gamma_K 4 \otimes t_3$
4	10	$\gamma_K(3 \otimes t_4 +_K 7 \otimes t_5)$

$\xrightarrow{\text{chase}_{\mathcal{M}^\star}}$

A_1	A_2	
2	1	t_1
2	5	t_2
3	4	t_3
4	3	t_4
4	7	t_5

h

Tabelle 4.39. Zweites Beispiel für eine inverse Schemaabbildung mit Provenance-Informationen der Operation $\gamma_{A_1;\textbf{SUM}(A_2)}r(\{A_1, A_2\})$

Durch Hinzunahme der Provenance-Polynome $\gamma_K(1 \otimes t_1 +_K 5 \otimes t_2)$, $\gamma_K 4 \otimes t_3$ und $\gamma_K(3 \otimes t_4 +_K 7 \otimes t_5)$ sowie der (minimalen) Zeugenbasis $\{\{t_1\}, \{t_3\}, \{t_4\}\}$ gelingt für die Schemaabbildung $\mathcal{M} = \gamma_{A_1;\textbf{SUM}(A_2)}r(\{A_1, A_2\})$ die Angabe einer exakten CHASE-Inversen (vgl. Tabelle 4.39). Besteht das Quellschema jedoch aus drei oder mehr Attributen kann nur noch eine relaxte CHASE-Inverse angegeben werden. Grund hierfür besteht in der anschließende Projektion (siehe Unterabschnitt 4.2.7). Gleiches gilt für den arithmetischen Durchschnitt **AVG**. Die Aggregatfunktion **COUNT** liefert stets eine relaxte CHASE-Inverse, da stets nur die Tupelanzahl sowie die Attributwerte des Gruppierungsattributes bekannt sind.

Arithmetische Operationen:

Operation:	$r(\mathcal{R})\ \theta\ \alpha$ mit $\theta \in \{+, -, \cdot, :\}$ und $\alpha \in \mathbb{R}$
Inverse Abbildung:	Exakte CHASE-Inverse
Probleme / Schwierigkeiten:	Verlassen der Attributdomänen

Existiert eine CHASE-inverse Abbildung, so handelt es sich um eine exakte CHASE-Inverse. Die Hinzunahme weiterer Provenance-Informationen ändert hieran nichts. Addition und Subtraktion sowie Multiplikation und Division sind unter den oben beschriebenen Aspekten weiterhin invers zueinander. Siehe hierzu auch die Tabelle 4.40.

A_1	A_2	A_3	
1	3	5	t_1
2	1	4	t_2
2	2	3	t_3

$\xrightarrow{\text{chase}_{\mathcal{M}}}$

A_1	A_2	A_3	
2	6	10	t_1
4	2	8	t_2
4	4	6	t_3

$\xrightarrow{\text{chase}_{\mathcal{M}^\star}}$

A_1	A_2	A_3	
1	3	5	t_1
2	1	4	t_2
2	2	3	t_3

h

Tabelle 4.40. Beispiel für eine inverse Schemaabbildung mit Provenance-Informationen der arithmetischen Operation $+$

Operation:	$r(\mathcal{R})$ mod α mit $\alpha \in \mathbb{R}$
Inverse Abbildung:	Ergebnisäquivalente CHASE-Inverse
Probleme / Schwierigkeiten:	Keine

Mit der Modulo-Rechnung verhält es sich ganz ähnlich wie bei den arithmetischen Operationen $+, -, \cdot$ und $:$. Mit Hilfe der Provenance-Polynome können wir uns zwar das Tupel, in dem eine Änderung stattfindet

merken; nicht jedoch den entsprechenden Attributwert. Hier wären Polynome, die sich sowohl die Position im Tupel als auch den Wert selbst merken, notwendig (vgl. Tabelle 4.41).

A_1	A_2	A_3	
1	3	5	t_1
2	1	4	t_2
2	2	3	t_3

$\xrightarrow{\text{chase}_{\mathcal{M}}}$

A_1	A_2	A_3	
1	0	1	t_1
2	2	2	t_2
2	1	0	t_3

$\xrightarrow{\text{chase}_{\mathcal{M}^\star}}$

A_1	A_2	A_3	
1	0	2	t_1
2	1	1	t_2
2	2	0	t_3

Tabelle 4.41. Beispiel für eine inverse Schemaabbildung der arithmetischen Operation **MOD** 3

4.2.6. Zusätzliche Informationen für die Existenz einer exakten CHASE-Inversen

Kopieren / Identitätsabbildung:
Für die Kopieroperation existiert bereits per Definition eine exakte CHASE-Inverse. Sie ist zu sich selbst invers, d.h.

$$\mathcal{M} = \mathcal{M}^\star = \mathrm{Id}.$$

Umbenennung:
Die Quellinstanz bleibt bei der Umbenennung erhalten. Es ändert sich lediglich das Schema, welches in der BACKCHASE-Phase aber wieder auf das Quellschema abgebildet wird. Die exakte CHASE-Inverse der Umbenennung ist entsprechend die Rückumbenennung, d.h.

$$\mathcal{M} \circ \mathcal{M}^\star = \beta_{A_j \leftarrow A_i} r(\mathcal{R}) \circ \beta_{A_i \leftarrow A_j} r(\mathcal{R}) = \mathrm{Id}.$$

Projektion:
Die Hinzunahme der Provenance ermöglicht im Falle der Projektion den Umgang mit Duplikaten. Für die Existenz einer exakten CHASE-Inversen müssen wir uns neben den zugehörigen Provenance-Polynomen auch die „wegprojizierten“ Attributwerte merken. Mit anderen Worten: Für eine exakte Rekonstruktion benötigen wir zusätzliche Tupel, welche die verlorenen Attributwerte enthalten.

Natürlicher Verbund:
Die Schwierigkeit des natürlichen Verbundes liegt in der Existenz von dangling tuples. Provenance-Aspekte bringen hier keine zusätzlichen Informationen, sie können also vernachlässigt werden. Es gilt vielmehr, den CHASE-Algorithmus um eine zu merkende Tupelmenge — die dangling tuples — zu erweitern.

Selektion:
Für die Selektion kann analog zum natürlichen Verbund die Provenance vernachlässigt werden. Für eine exakte Inverse muss man sich die „herausselektierten“ Originaltupel zusätzlich merken.

Mengenoperationen:
Für die Vereinigung kann mittels Provenance-Polynomen bereits eine exakte CHASE-Inverse angegeben werden. Im Falle der Differenzbildung kann der CHASE-Algorithmus hingegen nicht ohne weiteres angewandt werden. Es bleibt also nur noch die Schnittoperation zu untersuchen. Hier gilt es wie zuvor beim natürlichen Verbund, der Selektion und der Projektion die verlorenen Tupel zu erhalten. Der Provenance-Aspekt wieder ebenfalls nicht benötigt.

Aggregation:

Die Aggregation liefert bei zusätzlichen Provenance-Informationen je nach Typ eine ergebnisäquivalente (**MAX**, **MIN**), relaxte (**COUNT**) oder exakte CHASE-Inverse (**SUM**, **AVG**). Für die Existenz einer exakten CHASE-Inversen müssen wir uns zusätzliche Quelltupel (**MAX**, **MIN**) sowie einzelne Attributwerte (**COUNT**) merken.

Gruppierung:

Die Gruppierung verhält sich komplett analog zur Aggregation. Eine exakte CHASE-Inverse kann daher nur mit Hilfe von Provenance, zusätzlichen Attributwerten und Quelltupeln existieren.

Arithmetische Operationen:

Die arithmetischen Operationen $+, -, \cdot$ und : sind bereits exakt CHASE-invers. Merkt man sich zu jedem Attributwert der Ergebnisinstanz zusätzlich einen Multiplikationsfaktor, so kann auch für die Modulorechnung eine exakte CHASE-Inverse angegeben werden.

4.2.7. CHASE-inverse Schemaabbildungen für Kompositionen

Betrachten wir nun eine Komposition $\mathcal{M} = \mathcal{M}_1 \circ \mathcal{M}_2 \circ ... \circ \mathcal{M}_n$ der obigen Grundoperationen, so ergibt sich die inverse Funktion $\mathcal{M}^\star = \mathcal{M}^{-1} = (\mathcal{M}_1 \circ ... \circ \mathcal{M}_n)^{-1}$ als Komposition der inversen Teiloperationen $\mathcal{M}_1^\star, ..., \mathcal{M}_n^\star$, allerdings in umgekehrter Reihenfolge. Denn nach Satz 4.1 gilt für die zu $\mathcal{M}$ inverse CHASE-Abbildung $\mathcal{M}^\star$:

$$\mathcal{M}^\star = \mathcal{M}_n^\star \circ ... \circ \mathcal{M}_2^\star \circ \mathcal{M}_1^\star,$$

wobei $\mathcal{M}_i^\star$ $(i = 1, ..., n)$ CHASE-inversen Abbildung zu $\mathcal{M}_i$ ist. Der Typ der Funktion $\mathcal{M}^\star$ wiederum entspricht dem Typ der schwächsten Teilinversen $\mathcal{M}_i^\star$. Für die Anfrage 3.6

$$\pi_{A_1}(\sigma_{A_3='\text{Max}'}(r_1)) \bowtie \pi_{A_1 A_5 A_7}(r_2))$$

ergibt sich so eine ergebnisäquivalente CHASE-Inverse. Denn sowohl die Projektion mit Duplikatbildung, als auch die Selektion und der natürliche Verbund mit dangling tuples liefern die Existenz einer ergebnisäquivalenten CHASE-Inversen. Existiert zu einer der Abbildungen $\mathcal{M}_i$ keine CHASE-Inverse, so auch nicht zu $\mathcal{M}$.

Satz 4.1. *Seien X_1, X_2 und X_3 drei metrische Räume sowie $f : X_1 \to X_2$ und $g : X_2 \to X_3$ zwei bijektive Funktionen. Dann gilt:*

$$(f \circ g)^{-1} = g^{-1} \circ f^{-1}.$$

Beweis.

$$(f \circ g) \circ (g^{-1} \circ f^{-1}) \overset{\text{Kommutativität}}{=} f \circ \overbrace{(g \circ g^{-1})}^{=\text{Id}} \circ f^{-1} = \overbrace{f \circ f^{-1}}^{=\text{Id}} = \text{Id}$$

□

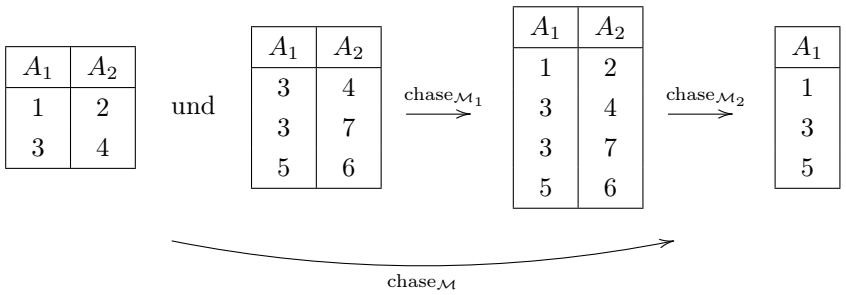
Wir wollen diese Feststellung anhand eines Beispiels untermauern. Seien dazu eine Anfrage Q, welche durch die Abbildung

$$\mathcal{M} = \mathcal{M}_2 \circ \mathcal{M}_1 = \pi_{A_1}(r_1\{A_1, A_2\}) \cup r_2(\{A_1, A_2\}))$$

umgesetzt wird, sowie zwei Quellinstanzen $I_1 = \{(1,2),(3,4)\}$ und $I_2 = \{(3,4),(5,6)\}$ gegeben. Dann existiert nach Tabelle 4.42 eine ergebnisäquivalente CHASE-Inverse und nach Tabelle 4.43 durch Hinzunahme des Provenance-Aspekts sogar eine relaxte CHASE-Inverse.

Betrachten wir die Teiloperationen Projektion und Vereinigung, ergeben sich ohne Provenance zwei ergebnisäuqivalente CHASE-Inverse und mit Provenance eine relaxte sowie eine exakte CHASE-Inverse. Die Hintereinanderausführung liefert nun ebenfalls eine ergebnisäquivalente bzw. eine relaxte CHASE-Inverse. Schauen wir uns hierzu die beiden BACKCHASE-Phasen etwas genauer an:

CHASE-Phase:



BACKCHASE-Phase:

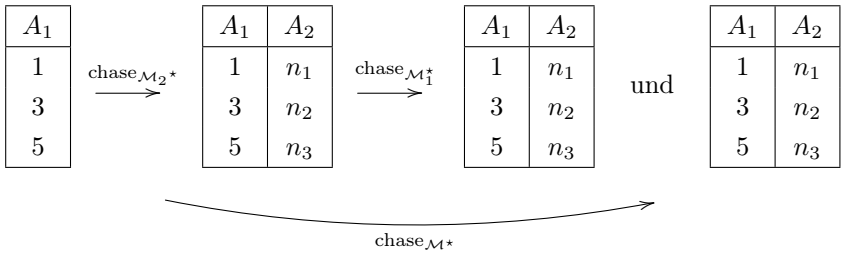


Tabelle 4.42. Beispiel für eine inverse Schemaabbildung einer Anfrage Q

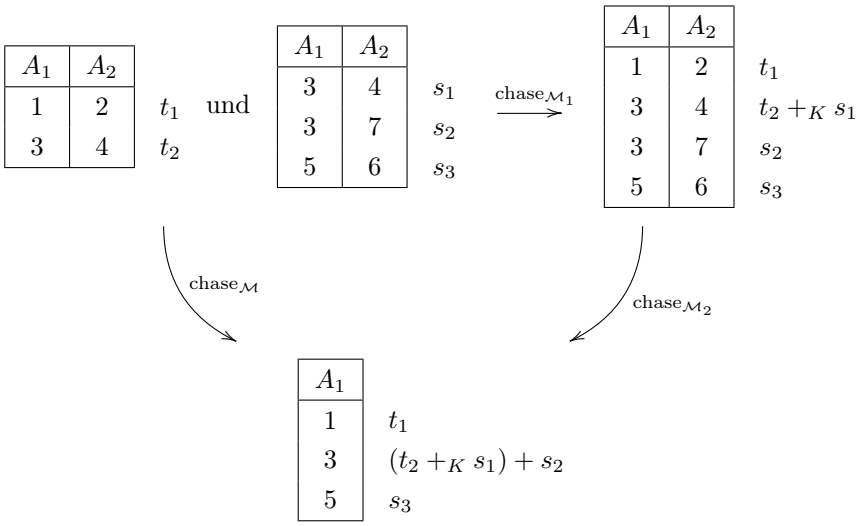
- Tabelle 4.42: Die Umkehrfunktion der Projektion $\mathcal{M}_2^\star$ liefert die Erweiterung der Tupel $(1),(3),(5)$ der Ergebnisinstanz I' um eine (markierte) Nullvariable im zweiten Attribut A_2. Die anschließende Anwendung der Abbildung $\mathcal{M}_1^\star$, d. h. der Umkehrfunktion der Vereinigung, entspricht einer Kopie der Tupel $(1,n_1),(3,n_2),(5,n_3)$ in die beiden Urinstanzen I_1'' und I_2''. Diese sind somit Obermengen der Quellinstanzen I_1 und I_2, sodass eine relaxte CHASE-Inverse nicht existiert. Wegen

$$\begin{aligned}\text{chase}_{\mathcal{M}}(I_1,I_2) &= \text{chase}_{\mathcal{M}}(\{(1,2),(3,4)\},\{(3,4),(3,7),(5,6)\})\\ &= \{(1),(3),(5)\}\\ &= \text{chase}_{\mathcal{M}}(\{(1,n_1),(3,n_2),(5,n_3)\},\{(1,n_1),(3,n_2),(5,n_3)\})\\ &= \text{chase}_{\mathcal{M}}(I_1'',I_2'')\end{aligned}$$

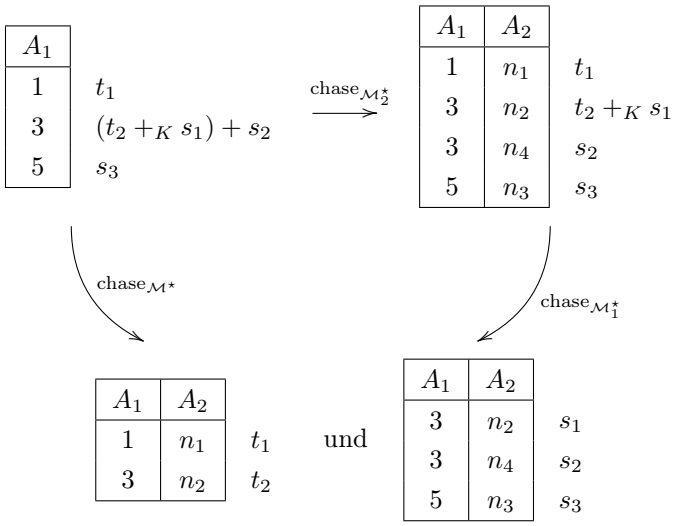
  kann aber eine ergebnisäquivalente CHASE-Inverse angegeben werden.

- Tabelle 4.43: Mit Hilfe der Provenance-Polynome ergibt sich im ersten Schritt ein zusätzliches Tupel $(3,n_4)$. Im zweiten Schritt werden diese vier Tupel nun richtig auf die Urinstanzen I_1'' und I_2'' aufgeteilt, so dass der für die Existenz einer relaxten CHASE-Inversen notwendige Homomorphismus h wie folgt definiert werden kann: $n_1 := 2$, $n_2 := 4$, $n_3 := 6$ und $n_4 := 7$. Zudem gilt weiterhin $\text{chase}_{\mathcal{M}}(I_1,I_2) \leftrightarrow \text{chase}_{\mathcal{M}}(I_1'',I_2'')$. Die relaxte CHASE-Inversen $\mathcal{M}^\star$ entspricht somit der Abbildung $\mathcal{M}_1^\star \circ \mathcal{M}_2^\star$.

CHASE-Phase:



BACKCHASE-Phase:



Homomophismus:

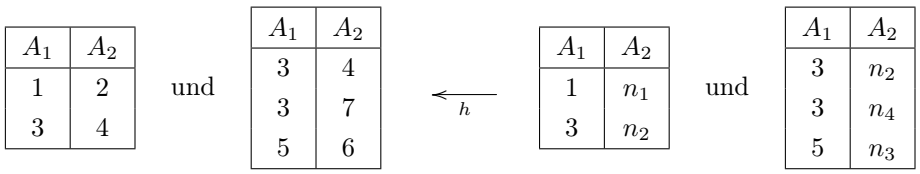


Tabelle 4.43. Beispiel für eine inverse Schemaabbildung einer Anfrage Q mit Provenance-Informationen

Das Ziel der obigen Untersuchungen bestand im Nachweis der Existenz einer inversen CHASE-Abbildung. Die zu merkenden Informationen der (minimalen) Zeugenbasen sollen dabei so gering wie möglich sein, aber dennoch bei Anwendung der Abbildung / Operation $\mathcal{M}$ ein korrektes Ergebnis liefern. So genügt im Falle der Projektion $\mathcal{M} = \pi_{A_1A_2}(r(\{A_1, A_2, A_3\}))$ aus Tabelle 4.27 beispielsweise eine Zeugenbasis $\{\{t_1\}, \{t_2\}\}$, welche auf die ersten beiden Attribute A_1 und A_2 eingeschränkt wird. Es gilt also $\{\{t_1\big|_{A_1,A_2}\}, \{t_2\big|_{A_1,A_2}\}\}$. Die Attributwerte des dritten Attributes A_3 sind für $\mathcal{M}$ nicht relevant und brauchen daher in der Zeugenbasis nicht berücksichtigt zu werden. Analog werden auch die Zeugenbasen für **MAX**, **MIN**, **COUNT**, **SUM** und **AVG** auf das zu untersuchende Attribut A_i eingeschränkt.

4.2.8. Zusammenfassung

In den vorhergegangenen Unterabschnitten wurden die Operationen des Grundlagenkapitels — Kopieren, Umbenennung, Projektion, natürlicher Verbund, Selektion, Mengenoperationen (Vereinigung, Schnitt, Differenz), klassische Aggregatfunktionen (Minimum, Maximum, Anzahl, Summe, Mittelwert), die Gruppierung mittels Group By sowie die arithmetischen Operationen (Multiplikation, Division, Addition und Subtraktion von Skalaren, Modulorechnung) — auf die Existenz von (exakten / relaxten / ergebnisäquivalenten) CHASE-Inversen überprüft. Die ergebnisäquivalente CHASE-Inverse entspricht dabei stets der Identitätsabbildung (Selektion, Mengenoperationen) oder im Falle von Relationenschemaänderungen der Erweiterung um (markierte) Nullvariablen (Projektion, natürlicher Verbund, Aggregation). Es zeigte sich zudem, dass nicht für alle Operationen eine inverse Abbildung konstruiert werden konnte. So kann der CHASE-Algorithmus für die Selektion auf Ungleichheit sowie die Differenzbildung nicht ohne weiteres angewandt werden. So geht im Falle der Aggregatfunktionen **SUM** und **AVG** sowie der Kombination dieser Aggregate mit der Gruppierungsoperation bereits in der CHASE-Phase zu viele Informationen verloren. Die Erweiterung dieses CHASE&BACKCHASE-Verfahrens um den Aspekt der Provenance ermöglichte in diesen Fällen jedoch die Angabe einer exakten CHASE-Inversen. Die hierfür benötigten Polynome und (minimalen) Zeugenbasen ergaben sich dabei nach den Rechenvorschriften aus Kapitel 3.1. Zudem konnten einige relaxte und ergebnisäquivalente CHASE-Inverse zu exakten CHASE-Inversen erweitert werden (Vereinigung, **SUM**, **AVG** sowie die Gruppierung dieser Aggregatfunktionen). Für die Projektion ohne Duplikate, die Selektion auf $<$, $\leq$, $=$, $\geq$, $>$, die Schnittbildung, die arithmetischen Operationen **MOD** sowie die Aufzählung und die Maximum- bzw. Minimumsuche konnten trotz Provenance-Untersuchungen keine Verbesserungen gefunden werden. Die Kopieroperation, die Umbenennung der natürliche Verbund ohne Duplikate sowie die arithmetischen $+$, $-$, $\cdot$ und $:$ liefern zudem bereits ohne Provenance-Aspekte exakte CHASE-Inverse.

	ohne Provenance		mit Provenance	
Operation $\mathcal{M}$	Inversen-Typ	Inverse $\mathcal{M}^\star$	Inversen-Typ	Inverse $\mathcal{M}^\star$
$r(\mathcal{R})$	$=$ [1]	Id[2]	$=$	Id
$\beta_{A_j \leftarrow A_i}(r(\mathcal{R}))$	$=$	$\beta_{A_i \leftarrow A_j}$	$=$	$\beta_{A_i \leftarrow A_j}$
$\pi_{A_i}(r(\mathcal{R}))$	$\leftrightarrow$ [3]	Erweiterung um NV[4] in A_i	$\leftrightarrow$	Erweiterung um NV in A_i
	$\sim$	Erweiterung um NV in A_i	$\leftrightarrow$	+ Rekonstruktion verlorener Tupel
$r_1(\mathcal{R}_1) \bowtie r_2(\mathcal{R}_2)$	$=$	Projektion auf Attribute aus $\mathcal{R}_i$	$=$	Projektion auf Attribute aus $\mathcal{R}_i$
	$\sim$ [5]	Projektion auf Attribute aus $\mathcal{R}_i$	$\sim$	Projektion auf Attribute aus $\mathcal{R}_i$
$\sigma_{A_i \theta c}(r(\mathcal{R}))$	$\sim$	Id	$\sim$	Id
$\sigma_{A_i \theta A_j}(r(\mathcal{R}))$	$\sim$	Id	$\sim$	Id
mit $\theta \in \{<, \leq, =, \geq, >\}$				
$\sigma_{A_i \neq c}(r(\mathcal{R}))$	xxx[6]	xxx	xxx	xxx
$\sigma_{A_i \neq A_j}(r(\mathcal{R}))$	xxx	xxx	xxx	xxx
$r_1(\mathcal{R}_1) \cup r_2(\mathcal{R}_2)$	$\sim$	Id	$=$	Selektion
$r_1(\mathcal{R}_1) \cap r_2(\mathcal{R}_2)$	$\sim$	Id	$\sim$	Id
$r_1(\mathcal{R}_1) - r_2(\mathcal{R}_2)$	xxx	xxx	xxx	xxx
$\mathbf{MAX}_{A_i}(r(\mathcal{R}))$ / $\mathbf{MIN}(r(\mathcal{R}))$	$\sim$	Id	$\sim$	Erweiterung um NV in A_j, $i \neq j$
$\mathbf{COUNT}_{A_i}(r(\mathcal{R}))$	$\leftrightarrow$	Erzeugen von Nulltupeln	$\leftrightarrow$	Erzeugen von Nulltupeln
$\mathbf{SUM}_{A_i}(r(\mathcal{R}))$	xxx	xxx	$=$	Rekonstruktion verlorener Attributwerte in A_i
$\mathbf{AVG}_{A_i}(r(\mathcal{R}))$	xxx	xxx	$=$	Rekonstruktion verlorener Attributwerte in A_i
$\gamma_{G_i;F_j(A_j)}(r(\mathcal{R}))$[7]	$\sim$	Id	$\sim$	Id
	$\leftrightarrow$	Erzeugen von Nulltupeln	$\leftrightarrow$	Erweiterung um NV in A_i + Rekonstruktion verlorener Tupel
	xxx	xxx	$=$	Rekonstruktion verlorener Tupel
$r(\mathcal{R})\ \theta\ \alpha$ mit $\theta \in \{+, -, \cdot, /\}$	$=$	$+ \ / \ - \ / \ \cdot \ / \ :$	$=$	$- \ / \ + \ / \ : \ / \ \cdot$
$r(\mathcal{R})$ **MOD** α	$\sim$	Id	$\sim$	Id

Abkürzungen:

[1] Exakte CHASE-Inverse
[2] Identitätsabbdilung
[3] Relaxte CHASE-Inverse
[4] (markierte) Nullvariable
[5] Ergebnisäquivalente CHASE-Inverse
[6] Es existiert keine (exakte / relaxte / ergebnisäquivalente) CHASE-inverse Abbildung.
[7] Es werden drei verschiedene Fälle unterschieden:

1. Fall $F_j \in \{\mathbf{MAX}, \mathbf{MIN}\} \Rightarrow$ Ergebnisäquvialente CHASE-Inverse
2. Fall $F_j \in \{\mathbf{COUNT}\} \Rightarrow$ Relaxate CHASE-Inverse
3. Fall $F_j \in \{\mathbf{SUM}\} \Rightarrow$
   - Es existiert keine der obigen CHASE-Inversen (ohne Provenance-Informationen).
   - Exakte CHASE-Inverse (mit Provenance-Informationen)

Tabelle 4.44. Grundoperation und ihre (exakten / relaxten / ergebnisäquivalenten) CHASE-Inversen

4.3. Inverse Schemaabbildungen am Beispiel des Hidden Markov-Modells

Zum Abschluss des Konzeptkapitels wollen wir unsere theoretischen Überlegungen auf ein praxisbezogenes Beispiel, das *Hidden-Markov-Modell* übertragen. Das Hidden-Markov-Modell wird dabei als ein Vertreter der Machine-Learning-Algorithmen ausgewählt, und Machine-Learning-Algorithmen sind die derzeit bei Big-Data-Analytics-Problemen meistverwendeten Techniken. Hierzu benötigen wir zunächst einen kurzen Abriss über die wichtigsten Definitionen wie etwa die *Markov-Ketten* und das *Hidden-Markov-Modell* selbst. Anschließend untersuchen wir die, nach [MH17] zur Darstellung des Hidden-Markov-Modells, benötigten Operationen auf die Existenz von CHASE-inversen Abbildungen. Das Hidden-Markov-Modell soll hier als Vertreter der Machine-Learning-Algorithmen (MLA) vertieft werden. Ist die Darstellung in Form von SQL-Anweisungen bekannt, so können auf analoge Weise auch andere MLA auf die Existenz von (exakten / relaxten / ergebnisäquivalenten) CHASE-Inversen untersucht werden.

4.3.1. Das Hidden-Markov-Modell

Das nach dem russischen Mathematiker A. A. Markow benannte *Hidden-Markov-Modell* ist einer der bekanntesten Machine-Learning-Algorithmen. Es beschreibt ein stochastisches Modell, in dem ein System durch eine Markov-Kette mit unbeobachteten Zuständen modelliert wird. Eine *Markov-Kette* selber ist ein spezieller stochastischer Prozess, der eine Art „Gedächtnislosigkeit“ simuliert. Dies bedeutet, dass die zukünftige Entwicklung des Prozesses nur vom zuletzt beobachteten Zustand abhängig, nicht jedoch von den vorherigen Zuständen. Durch die Kenntnis der gesamten Vorgeschichte können also keine zusätzlichen Informationen gewonnen werden. Vergleiche hierzu die Definitionen 4.7 und 4.8 aus [WS13].

Definition 4.7 (Markov-Kette)**.** Ein zeitdiskreter stochastischer Prozess $(X_t)_{t\in\mathbb{N}_0}$ mit abzählbarem Zustandsraum $S = \{s_1, s_2, ...\}$ heißt **Markov-Kette**, wenn für alle Zeitpunkte $t \in \mathbb{N}_0$ und alle Zustände $s_{i_1}, ..., s_{i_t}, s_{i_{t+1}} \in S$ gilt:

$$P(x_{t+1} = s_{i_{t+1}} | X_0 = s_{i_0}, ..., X_{t-1} = s_{i_{t-1}}, X_t = s_{i_t}) = P(X_{t+1} = s_{i_{t+1}} | X_t = s_{i_t}).$$

Diese Bedingung heißt auch **Markov-Eigenschaft**. ⊡

Definition 4.8 (Übergangsmatrix, Übergangswahrscheinlichkeit)**.** Die **Übergangsmatrix** $P = (p_{ij})_{i,j=1...n}$ einer Markov-Kette $(X_t)_{t\in\mathbb{N}_0}$ mit abzählbarem Zustandsraum S ist definiert über die **Übergangswahrscheinlichkeiten**

$$p_{ij} := p_{ij}(t) = P(X_{t+1} = s_i | X_t = s_j).$$

Dabei gilt $p_{ij} \geq 0$ für alle $i, j \in I$ und $\sum_{j\in I} p_{ij} = 1$. Die **n-Schritt-Übergangswahrscheinlichkeit** ist dabei definiert als

$$p_{ij}^{(n)} = P(X_n = s_j | X_0 = s_i) = \sum_{i_1\in I} \cdots \sum_{i_{n-1}\in I} p_{i,i_1} \cdot ... \cdot p_{i_{n-1},j}.$$

⊡

Das folgende Beispiel fasst die Definitionen der Übergangsmatrix A sowie der 2-Schritt Übergangswahrscheinlichkeiten für vier gegebene Wahrscheinlichkeiten $P(X_{t+1} = s_i | X_t = s_j)$ zusammen. Die Wahrscheinlichkeiten geben eine Markov-Kette erster Ordnung an.

Beispiel 4.3. *Für die gegebenen Übergangswahrscheinlichkeiten*

$$p_{11} = P(X_{t+1} = s_1 | X_t = s_1)$$
$$p_{12} = P(X_{t+1} = s_1 | X_t = s_2)$$
$$p_{21} = P(X_{t+1} = s_2 | X_t = s_1)$$
$$p_{22} = P(X_{t+1} = s_2 | X_t = s_2)$$

können die zugehörige Übergangsmatrix

$$A = \begin{pmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{pmatrix}$$

aufgestellt und die 2-Schritt Übergangswahrscheinlichkeiten

$$\begin{aligned} P(X_2 = s_1 | X_0 = s_1) &= P(X_2 = s_1 | X_1 = s_1) \cdot P(X_1 = s_1 | X_0 = s_1) \\ &\quad + P(X_2 = s_1 | X_1 = s_2) \cdot P(X_1 = s_2 | X_0 = s_1) \\ &= p_{11} \cdot p_{11} + p_{12} \cdot p_{21} \\ &= p_{11}^2 + (p_{12} \cdot p_{21}) \\ P(X_2 = s_2 | X_1 = s_1) &= P(X_2 = s_3 | X_1 = s_1) \cdot P(X_1 = s_1 | X_0 = s_1) \\ &\quad + P(X_2 = s_2 | X_1 = s_2) \cdot P(X_1 = s_2 | X_0 = s_1) \\ &= p_{21} \cdot p_{11} + p_{22} \cdot p_{21} \\ &= p_{21} \cdot (p_{11} + p_{22}) \end{aligned}$$

berechnet werden.

Die Modellierung als Markov-Kette simuliert also die Abhängigkeit eines Ereignisses ausschließlich vom vorherigen Ereignis bei konstanter Übergangswahrscheinlichkeit. Ein häufig gewähltes Beispiel hierfür ist die Wetterprognose: Ob morgen die Sonne scheint, hängt ausschließlich vom heutigen Wetter ab, nicht jedoch von der Begebenheiten der letzten Woche [RN04]. Bei einem Hidden-Markov-Modell werden diese Zustände x_t jedoch nicht selbst beobachtet, sondern liegen im Verborgenen (engl. hidden). Sie sind jedoch einer beobachteten Ausgabe y_t zugeordnet, die je nach Zustand mit unterschiedlichen Wahrscheinlichkeiten auftreten (siehe Abbildung 4.1).

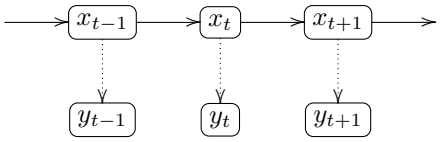


Abbildung 4.1. Hidden-Markov-Modell

Die Aufgabe besteht nun in der Bestimmung der Wahrscheinlichkeiten der verborgenen Zustände mit Hilfe der beobachteten Sequenz der Emissionen. Das HMM ist dabei wie folgt formal definiert:

Definition 4.9 (Hidden-Markov-Modell)**.** Seien $(X_t)_{t\in\mathbb{N}}$ und $(Y_t)_{t\in\mathbb{N}}$ zwei zeitdiskrete Zufallsprozesse. Ein **Hidden-Markov-Modell** (kurz: Hidden-Markov-Modell (HMM)) ist ein 5-Tupel $\lambda = (S, V, A, B, \pi)$ bestehend aus

- einer Menge von Zuständen $S = \{s_0, s_1, ..., s_n\}$ — Werte der Zufallsvariablen X_t,

- einem Alphabet der möglichen Beobachtungen $V = \{v_1, v_2, ..., v_m\}$ — Emission der Y_t,
- einer Übergangsmatrix $A = (a_{ij})_{i,j=1,...,n} \in [0,1]^{n \times n}$ mit $a_{ij} = P(X_{t+1} = i | X_t = j)$ — A beschreibt die Wahrscheinlichkeit des Übergangs vom Zustand s_i in den Zustand s_j,
- einer Bearbeitungsmatrix $B = (b_{ij})_{i=1,...,n;j=1,...,m}$ mit $b_{ij} = P(y_t = v_i | x_t = s_j) \in [0,1]^{m \times n}$ — B hält die Wahrscheinlichkeiten für die Beobachtungen von V_i,
- einer Anfangsverteilung $\pi \in \mathbb{R}^n$ mit $\pi_i(0) := P(X_0 = s_i)$.

Ein HMM heißt **zeitinvariant**, wenn die Übergangs- und Bearbeitungsmatrix unabhängig von der Zeit sind. ⊡

4.3.2. Das Hidden-Markov-Modell in SQL

D. Marten und A. Heuer beschreiben in ihrem Artikel *Machine Learning on Large Databases: Transforming Hidden Markov Models to SQL Statements* [MH17] die Umformulierung eines in der analytischen Sprache R gegebenen HMM in eine Folge von SQL-Anweisungen. Ihr langfristiges Ziel besteht in der automatischen Umwandlung von Machine-Learning-Algorithmen in SQL-Datenbanksysteme.

Für die Formulierung des HMM benötigen die beiden Autoren nur einige wenige Grundoperationen: die Addition und Subtraktion, die skalare Multiplikation und Division sowie die Matrix-Vektor- und Matrix-Matrix-Multiplikation und die Aufzählung mittels **COUNT**. Dabei sind drei Besonderheiten zu beachten:

- Die Matrix-Vektor-Multiplikation kann als Spezialfall der Matrix-Matrix-Multiplikation angesehen werden. Beide werden durch eine Aggregation **SUM** mit Gruppierung codiert (siehe Anfrage 4.4).
- Im Falle der Addition und Subtraktion muss der Umgang mit Nullwerten spezifiziert werden. Diese können entweder explizit gespeichert oder aber vernachlässigt werden. Die explizite Speicherung der Nullwerte gelingt mit Hilfe eines **INNER JOIN** (siehe Anfrage 4.5). Da das Fehlen von Null-Werten bei dieser Operation jedoch zum Verlust von Tupeln führt, wird im Falle der Nullwert-Vernachlässigung stattdessen ein **OUTER JOIN** verwendet (siehe Anfrage 4.6). Alternativ können Nullwerte auch durch den Wert 0 ersetzt werden.

Für die folgenden Betrachtungen seien $A = (a_{ij})_{i,j=1,...,n}$ und $B = (b_{ij})_{i=1,...,n;j=1,...,m}$ die Übergangs- und Bearbeitungsmatrix von oben. Sie werden im Relationenalgebra-Kalkül durch das Relationenschema $\mathcal{R} = \{i, j, v\}$ dargestellt, wobei i die Zeile, j die Spalte und v den entsprechenden Eintrag einer Matrix angibt. Das Tupel $(2, 1, 2)$ der Relation A entspricht also dem Matrixeintrag $a_{21} = 2$.

Anfrage 4.4 Matrix-Matrix-Multiplikation

```
SELECT A.i, B.j, SUM(A.v · B.v)
FROM A JOIN B ON A.j = B.i
GROUP BY A.i, B.j
```

Anfrage 4.5 Addition mit expliziten Nullwerten

```
SELECT A.i, A.j, A.v + B.v AS v
FROM A JOIN B ON A.i = B.i AND A.j = B.j
```

Anfrage 4.6 Addition mit vernachlässigten Nullwerten

```
SELECT A.i, A.j,
   ISNULL(A.v) THEN 0.0 ELSE A.v END
 + ISNULL(B.v) THEN 0.0 ELSE B.v END
FROM A OUTER JOIN B ON A.i = B.i AND A.j = B.j
```

4.3.3. CHASE-inverse Schemaabbildungen

Den folgenden Untersuchungen legen wir das Beispiel aus Abbildung 4.2 zu Grunde. Die Übergangsmatrix A sowie der Bearbeitungsmatrix B sind hierbei über die Übergangswahrscheinlichkeiten a_{11}, a_{12} und a_{21} sowie die Bearbeitungswahrscheinlichkeiten b_{11}, b_{21}, b_{22} und b_{23} definiert. Sie sind zusätzlich mit (markierten) Nullvariablen aufgefüllt, sodass

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & n_1 \end{pmatrix} \in [0,1]^{2\times 2}$$

und

$$B = \begin{pmatrix} b_{11} & n_2 & n_3 \\ b_{21} & b_{22} & n_4 \end{pmatrix} \in [0,1]^{2\times 3}$$

gilt, denn eine korrekte Matrix-Matrix-Multiplikation wäre im Fall nicht explizit angegebener Nullwerte nicht ohne Weiteres möglich.

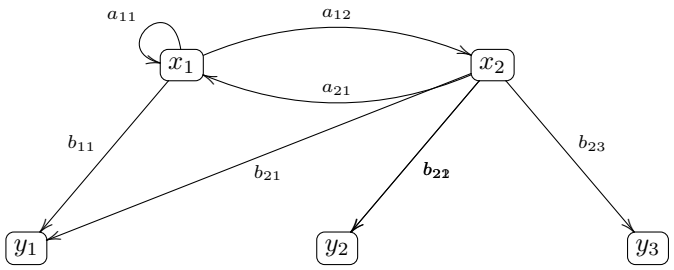


Abbildung 4.2. Beispiel für ein Hidden-Markov-Modell

Kommen wir zurück auf die Frage nach Existenz von CHASE-inversen Abbildungen, so stellen wir fest, dass für das Hidden-Markov-Modell stets eine relaxte CHASE-Inverse existiert. Die skalare Multiplikation und Division liefern aufgrund ihrer rückverfolgbaren Veränderung jedes Wertes innerhalb einer Spalte stets eine exakte CHASE-Inverse. Die Tabelle 4.45 verdeutlicht dies für die Multiplikation der Übergangsmatrix A mit einem Skalar $\mu \in \mathbb{R}$.

i	j	v
1	1	a_{11}
1	2	a_{12}
2	1	a_{21}
2	2	n_1

$\xrightarrow{\text{chase}_{\mathcal{M}}}$

i	j	v
1	1	$a_{11} \cdot \mu$
1	2	$a_{12} \cdot \mu$
2	1	$a_{21} \cdot \mu$
2	2	n_1

$\xrightarrow{\text{chase}_{\mathcal{M}^\star}}$

i	j	v
1	1	a_{11}
1	2	a_{12}
2	1	a_{21}
2	2	n_1

$\xleftrightarrow{h}$

Tabelle 4.45. Beispiel für die inverse Schemaabbildung der skalaren Multiplikation

Die Aggregation mit Gruppierung wurde bereits in den Unterabschnitten 4.2.4 und 4.2.5 untersucht. Für die Aufzählung und die Summierung liegen ohne die Verwendung von zusätzlichen Provenance-Informationen keine CHASE-inversen Abbildungen vor. Mit Provenance existieren hingegen relaxte CHASE-Inverse. Die skalare Division sowie die Subtraktion können als inverse Funktionen der skalaren Multiplikation sowie der Addition gesehen werden. Sie werden hier daher nicht weiter untersucht. Unser Hauptaugenmerk liegt somit auf der Addition (realisiert durch eine Verbundoperation) sowie der Matrix-Matrix-Multiplikation. Die Matrix-Vektor-Multiplikation folgt wie oben erwähnt als Spezialfall der Matrix-Matrix-Multiplikation.

Verbund-Typ	Inverse Abbildung	Begründung
CROSS	Exakt	Kartesisches Produkt
INNER	Ergebnisäquivalent[3]	Verlust von Zeilen durch Dangling Tupels
LEFT / RIGHT	Ergebnisäquivalent	Verlust von Zeilen durch Dangling Tupels
NATURAL	Ergebnisäquivalent / Exakt	Verlust von Zeilen durch Dangling Tupels / Kartesisches Produkt
FULL OUTER[4]	Exakt	Vollständiges Auffüllen mit (markierten) Nullvariablen

Tabelle 4.46. CHASE-inverse Schemaabbildungen für die verschiedenen Verbund-Typen

Die Definition des **CROSS JOIN** folgt unten, die Definitionen der anderen Verbund-Typen können in [SSH13] nachgelesen werden. Der für uns relevante **INNER JOIN**, welcher aus dem natürlichen Verbund abgeleitet werden kann, sowie der **FULL OUTER JOIN** sind zudem in Abbildung 4.3 bildhaft dargestellt.

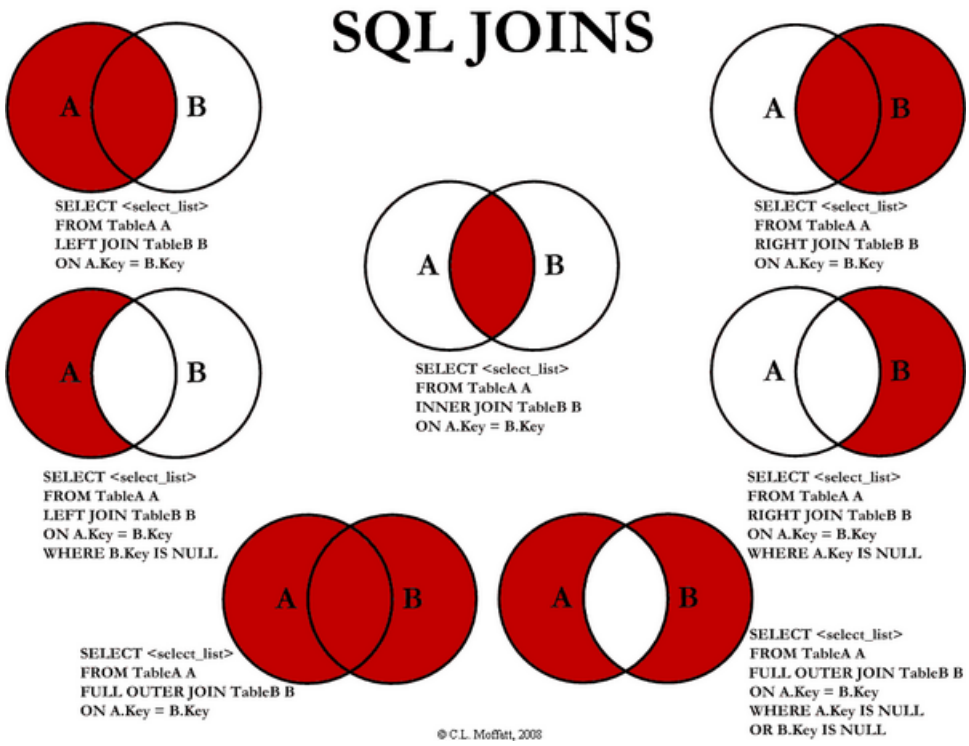


Abbildung 4.3. Übersicht über verschiedene JOIN-Arten [Cod17]

[3]Bezogen auf den allgemeinen Fall. Bei expliziter Angabe der Nullwerte existiert eine exakte CHASE-Inverse.
[4]Voraussetzung: Die Originalrelationen enthalten keine Nullwerte in den Tupeln.

Definition 4.10 (**CROSS JOIN**). Der **CROSS JOIN** $\times$ zweier Relationen $r_1(\mathcal{R}_1)$ und $r_2(\mathcal{R}_2)$ mit $\mathcal{R}_1 = \{A_1, ..., A_n\}$ und $\mathcal{R}_2 = \{A'_1, ..., A'_m\}$ entspricht dem kartesischen Produkt

$$r_1(\mathcal{R}_1) \times r_2(\mathcal{R}_2) = \{(x_1, ..., x_n, y_1, ..., y_m) \mid x \in r_1(\mathcal{R}_1), y \in r_2(\mathcal{R}_2)\}.$$

□

Bei expliziter Angabe der Nullwerte wird die Addition durch einen **INNER JOIN** verkörpert. In diesem Fall ergibt sich wiederum eine exakte CHASE-Inverse (vgl. Tabelle 4.47 mit zwei Übergangsmatrizen A und A'). Für den (allgemeinen) Fall, dass die Nullwerte nicht gespeichert werden, liefert dieser Verbund-Typ lediglich eine ergebnisäquivalente CHASE-Inverse. Dargestellt durch die grüne Markierung in der Tabelle 4.47. Die Hinzunahme des Provenance-Aspekts würde hieran nichts ändern.

i	j	v
1	1	a_{11}
1	2	a_{12}
2	1	a_{21}
2	2	n_1

und

i	j	v
1	1	a'_{11}
1	2	n_2
2	1	a'_{21}
2	2	a'_{22}

$\xrightarrow{\text{chase}_{\mathcal{M}^\star}}$

i	j	v	i	j	v
1	1	a_{11}	1	1	a'_{11}
1	2	a_{12}	1	2	n_2
2	1	a_{21}	2	1	a'_{21}
2	2	n_1	2	2	a'_{22}

h ↘ ↙ $\text{chase}_{\mathcal{M}^\star}$

i	j	v
1	1	a_{11}
1	2	a_{12}
2	1	a_{21}
2	2	n_1

und

i	j	v
1	1	a'_{11}
1	2	n_2
2	1	a'_{21}
2	2	a'_{22}

Tabelle 4.47. Beispiel für eine inverse Schemaabbildung der Operation **INNER JOIN**

i	j	v
1	1	a_{11}
1	2	a_{12}
2	1	a_{21}

und

i	j	v
1	1	a'_{11}
2	1	a'_{21}
2	2	a'_{22}

$\xrightarrow{\text{chase}_{\mathcal{M}}}$

i	j	v	i	j	v
1	1	a_{11}	1	1	a'_{11}
1	2	a_{12}	n_4	n_5	n_6
2	1	a_{21}	2	1	a'_{21}
n_1	n_2	n_3	2	2	a'_{22}

h ↘ ↙ $\text{chase}_{\mathcal{M}^\star}$

i	j	v
1	1	a_{11}
1	2	a_{12}
2	1	a_{21}
n_1	n_2	n_3

und

i	j	v
1	1	a'_{11}
n_4	n_5	n_6
2	1	a'_{21}
2	2	a'_{22}

Tabelle 4.48. Beispiel für eine inverse Schemaabbildung der Operation **OUTER JOIN**

Werden die Nullwerte hingegen vernachlässigt, kann die Addition durch einen **OUTER JOIN** verwirklicht werden. Für diesen **JOIN** existiert eine CHASE-Inverse. Grund dafür sind die Urinstanzen. Sie entsprechen jeweils den um ein Null-Tupel erweiterten Quellinstanzen. Für das Beispiel aus Tabelle 4.48 bedeutet dies:

$$I_1'' = \{(1,1,a_{11}),(1,2,a_{12}),(2,1,a_{21}),(n_1,n_2,n_3)\} = I_1 \cup \{(n_1,n_2,n_3)\}$$

und

$$I_2'' = \{(1,1,a'_{11}),(n_4,n_5,n_6),(2,1,a'_{21}),(2,2,a'_{22})\} = I_2 \cup \{(n_4,n_5,n_6)\}.$$

Für die Matrix-Matrix-Multiplikation existiert eine exakte CHASE-Inverse, sofern die Provenance-Polynome zur Hilfe genommen werden. Ohne diese Polynome kann die starke Datenverdichtung nicht rückgängig gemacht und keine Inverse bestimmt werden. Eine (minimale) Zeugenbasis braucht in diesem Fall nicht angegeben zu werden. Um eine Matrix-Matrix-Multiplikation zu invertieren, werden die kompletten Ausgangsmatrizen benötigt. Die Zeugenbasis besteht somit jeweils aus allen (nicht auf bestimmte Attribute eingeschränkten) Tupeln Quellinstanzen der beiden Matrizen (vgl. Tabelle 4.49).

i	j	v	
1	1	a_{11}	t_1
1	2	a_{12}	t_2
2	1	a_{21}	t_3
2	2	n_1	t_3

und

i	j	v	
1	1	b_{11}	s_1
1	2	n_2	s_2
1	3	n_3	s_3
2	1	b_{21}	s_4
2	2	b_{22}	s_5
2	3	n_4	s_6

$\downarrow \text{chase}_{\mathcal{M}}$

i	j	v	
1	1	$a_{11} \cdot b_{11} + a_{12} \cdot b_{21}$	$t_1 \cdot_K s_1 +_K t_2 \cdot_K s_4$
1	2	$a_{11} \cdot n_2 + a_{12} \cdot b_{22}$	$t_1 \cdot_K s_2 +_K t_2 \cdot_K s_5$
1	3	$a_{11} \cdot n_3 + a_{12} \cdot n_4$	$t_1 \cdot_K s_3 +_K t_2 \cdot_K s_6$
2	1	$a_{21} \cdot b_{11} + n_1 \cdot b_{21}$	$t_3 \cdot_K s_1 +_K t_4 \cdot_K s_4$
2	2	$a_{21} \cdot n_2 + n_1 \cdot b_{22}$	$t_3 \cdot_K s_2 +_K t_4 \cdot_K s_5$
2	3	$a_{21} \cdot n_3 + n_1 \cdot n_4$	$t_3 \cdot_K s_3 +_K t_4 \cdot_K s_6$

$\downarrow \text{chase}_{\mathcal{M}^\star}$

i	j	v	
1	1	a_{11}	t_1
1	2	a_{12}	t_2
2	1	a_{21}	t_3
2	2	n_1	t_3

und

i	j	v	
1	1	b_{11}	s_1
1	2	n_2	s_2
1	3	n_3	s_3
2	1	b_{21}	s_4
2	2	b_{22}	s_5
2	3	n_4	s_6

Tabelle 4.49. Beispiel für eine inverse Schemaabbildung der Matrix-Matrix-Multiplikation

Zusammengefasst haben wir für die einzelnen Operationen der HMM:

- Eine exakte CHASE-Inverse für die skalare Multiplikation und Division;
- Eine exakte CHASE-Inverse für die Matrix-Matrix- sowie Matrix-Vektor-Multiplikation (bei zusätzlicher Provenance;
- Eine relaxte CHASE-Inverse für die Aggregation mit Gruppierung;
- Eine ergebnisäquivalente CHASE-Inverse für die Addition und Subtraktion (bei expliziter Speicherung der Nullwerte).

Algorihtmus 4.7 Forward Algorithmus

```
Input: y = {y_0, ..., y_θ}
p := π · B_{:,y_0}                                   (Initialisierung)
for i = 1 : θ do                                     (Rekursion)
    p := p^T A · B_{:,y_t}
end
P(y | A, B, π) = SUM(p)                              (Terminierung)
```

Für das Hidden-Markov-Modell kann somit für jede Methode eine ergebnisäquivalente CHASE-Inverse angegeben werden. Sie entspricht der Hintereinanderausführung der einzelnen invertierten Schritte dieser Methode. Für den von D. Marten und A. Heuer untersuchten *Forward Algorithmus* (siehe Algorithmus 4.7) würde dies bedeuten:

$$\mathcal{M}^\star = \mathcal{M}^\star_{2\theta+1} \circ \mathcal{M}^\star_{2\theta} \circ \mathcal{M}^\star_2 \circ \mathcal{M}^\star_1.$$

Dabei entsprechen die Funktionen $\mathcal{M}_1, ..., \mathcal{M}_{2\theta+1}$ den folgenden Operationen:

Terminierung:	$\mathcal{M}_{2\theta+1}$	=	Summation über p
Rekursion:	$\mathcal{M}_{2i}$	=	Matrix-Vektor-Multiplikation $(1 \leq i \leq \theta)$
	$\mathcal{M}_{2i+1}$	=	Matrix-Matrix-Multiplikation $(1 \leq i \leq \theta)$
Initialisierung:	$\mathcal{M}_1$	=	skalare Multiplikation

Die CHASE-Inversen ergeben sich nach den obigen Erläuterungen als Erzeugung von Nulltupeln mit Rekonstruktion der verlorenen Attributwerte (Summation), Selektion mit anschließender Projektion (Matrix-Matrix-Multiplikation) und skalare Division (skalare Multiplikation). Übliche Methoden sind neben dem *Forward-Algorithmus* auch der *Viterbi-Algorithmus* sowie der *Baum-Welch-Algorithmus* [JM17]. Beide kommen mit den bisher untersuchten Grundoperationen aus.

5. Praktische Umsetzung

Als praktische Anwendung der bisherigen Theorien soll ein kurzes Programm entwickelt werden, welches SQL-Anweisungen mit zusätzlichen Provenance-Untersuchungen ermöglicht. Ziel ist die Angabe der ***where***-, ***why***- und ***how***-Provenance, des Typs der zugehörigen CHASE-Inversen, sofern diese existiert, sowie der Zeugenbasis. Diese muss nicht zwangsläufig minimal sein. Die Grundlage hierfür bilden das Universitätsbeispiel aus dem ersten Kapitel sowie der Programmrohling **ProSA**.

5.1. Analyse des Programms ProSA

Das Provenance-Analyse-Programm **ProSA** wurde im Sommersemester 2016 im Rahmen des Projekts *Neueste Entwicklungen in der Informatik* der Universität Rostock entwickelt. Es ermöglicht die Auswertung von einfachen SQL-Anfragen mit Projektion, Selektion, natürlichem Verbund sowie den fünf üblichen Aggregaten (**MIN**, **MAX**, **COUNT**, **SUM** und **AVG**). Hierfür werden drei CSV-Datein erstellt, welche jeweils das Anfrageergebnis mit den extensionalen Antworten der zugehörigen ***where***-, ***why***- und ***how***-Fragen enthalten.

Anfrage 5.1 Prüfungsergebnisse der Studenten mit Vornamen 'Max' (Anfrage Q_6)

```
SELECT Matrikelnr, Modulnr, Note
FROM Studenten JOIN Noten
ON (Studenten.Matrikelnr = Noten.Matrikelnr)
WHERE Studenten.Vorname = 'Max'
```

Für die Anfrage Q_6 liefert **ProSA** die unten stehende Ergebnistabelle (Spalten 1 bis 3) erweitert um die Ergebnisse der ***where***-, ***why***- und ***how***-Anfragen. Hierfür werden die drei Provenance-Fragen unabhängig voneinander beantwortet und in drei separaten CSV-Dateien abgespeichert. Der Übersicht halber sind sie hier jedoch in einer Tabelle zusammengefasst (siehe Tabelle 5.1). Für die Berechnung der Provenance-Polynome ist eine Unterscheidung zwischen einfachen Anfragen und Anfrage mit Aggregationen nötig, da sich der Typ der Polynome erheblich unterscheidet — vergleiche hierzu die Erweiterung der positiven K-relationalen Algebra zu einem K-Seminomodul (Unterabschnitt 3.2.2). Die ***where***- und ***why***-Untersuchungen werden hiervon nicht beeinflusst.

Matrikelnr	Modulnr	Note	***where***-Provenance	***why***-Provenance	***how***-Provenance
3	2	2.3	Studenten, Noten	(S_3, N_7)	$S_3 \cdot N_7$
7	2	3.3	Studenten, Noten	(S_7, N_{11})	$S_7 \cdot N_{11}$
3	4	1.3	Studenten, Noten	(S_3, N_{13})	$S_3 \cdot N_{13}$
7	5	1.7	Studenten, Noten	(S_7, N_{16})	$S_7 \cdot N_{16}$
3	7	1.7	Studenten, Noten	(S_3, N_{20})	$S_3 \cdot N_{20}$

Tabelle 5.1. Anfrageergebnisse erweitert um ***where***-, ***why***- und ***how***-Provenance

Verglichen mit den Untersuchungen aus Unterabschnitt 3.3.2 stimmen die Anfrageergebnisse mit denen der Tabelle 3.18 überein. Auch die ***how***-Provenance wurde im Unterabschnitt 3.2.2 bereits theoretisch berechnet.

```
for (j = i, z = 1; j <= columnsNumber; j++, z++) {
    if (z > 1) {
        polynome += " * ";
        tuple += ",";
        whereProv += ",";
    }
    polynome += rs.getString(j);
    tuple += rs.getString(j);
    whereProv += dbschema.getTableName(rsmd.getColumnName(j));
}
```

Abbildung 5.1. Berechnung der ***where***-, ***why***- und ***how***-Provenance ohne Aggregation

ProSA berechnet die ***where***-, ***why***- und ***how***-Anfragen über der Ergebnismenge *rs* sowie die Metadatenmenge *rsmd* einer gegebenen Anfrage Q. Dargestellt sind diese in Abbildung 5.1. Sie ergeben sich durch Aufzählung der Tabellennamen (*whereProv*), Aufzählung der Tupelidentifikatoren (*tuple*) sowie als Produkt der Tupelidentifikatoren (*polynome*).

5.2. Erweiterungen des Programms ProSA

Die Erweiterung des Programms **ProSA** beinhaltet die Angabe des CHASE-Inverse-Typs, der zugehörigen Zeugenmenge und -liste sowie die zugehörigen Quelltupel (siehe Abbildung 5.2). Die Ausgabe der Ergänzungen wird in einem TXT-File gesondert gespeichert.

Inverse.txt

```
Inversen-Typ:
Ergebnisäquivalente CHASE-Inverse

Zeugenmenge:
[N11, N13, N16, N20, N7, S3, S3, S3, S7, S7]

Zeugenliste:
[N11, N13, N16, N20, N7, S3, S7]

Quelltupel der Zeugenliste:
studenten-Relation:
matrikelnr; name; vorname; studiengang; id_studenten
3; Müller; Max; Elektrotechnik; S3
7; Mustermann; Max; Elektrotechnik; S7
------------------------------------------------------------
noten-Relation:
modulnr; matrikelnr; semester; note; id_noten
2; 7; WS 15/16; 3.3; N11
4; 3; WS 16/17; 1.3; N13
5; 7; SS 17; 1.7; N16
7; 3; SS 16; 1.7; N20
2; 3; WS 14/15; 2.3; N7
```

Abbildung 5.2. Ausgabe der Erweiterungen

Vollzogen werden diese Neuerungen in sechs Schritten. Der Quelltext der drei wichtigsten Funktionen *provenance*, *FindInverseType* und *FindWitnessQuery* des erweiterten Programms **ProSA** findet sich in gekürzter Fassung im Anhang C. Dabei sind in der überarbeiteten *provenance*-Funktion alle Neuerungen mittels //// < Erweiterung > //// gekennzeichnet

1. Untersuchung der Anfrage Q auf ihre Grundoperationen (Selektion, Projektion, natürlicher Verbund sowie Aggregation)
2. Bestimmung des CHASE-Inversen-Typs
3. Aufstellen der Zeugenmenge $\{Z_1, ..., Z_m\}$ aus den Ergebnissen der ***why***-Provenance ($m \in \mathbb{N}^+$)
4. Berechnung der Zeugenliste $\{Z_1, ..., Z_n\}$ durch Streichen von Duplikaten in der Zeugenmenge $\{Z_1, ..., Z_m\}$ ($n \in \mathbb{N}^+, n \leq m$)
5. Aufstellen von Anfragen der Form SELECT * FROM < Tabellenname > WHERE ID = 'Z_i'
6. Ausgabe der zugehörigen Quelltupel $t_1, ..., t_n$

Die Einführung boolescher Variablen ermöglicht die Existenzangabe einer (exakten / relaxten / ergebnisäquivalenten) CHASE-Inversen. Hierfür wird für jede Grundoperation eine boolesche Variable definiert, die, sollte sie in der Anfrage Q enthalten sein, auf „true“ gesetzt wird. Der Inversentyp wird nun anhand dieser Variablen bestimmt: Nach unseren Erkenntnissen in Abschnitt 4.2 existiert zu jeder der hier untersuchten Operationen mindestens eine ergebnisäquivalente CHASE-Inverse. Beinhaltet Q eine Selektion oder eine Maximum- bzw. Miminumberechnung, so kann auch keine höhere Inverse angegeben werden. Für den Fall einer der Aggregationen **COUNT**, **SUM** und **AVG** sowie der Projektion erhalten wir eine relaxte CHASE-Inverse. Im Falle des natürlichen Verbundes existiert entweder eine exakte CHASE-Inverse oder aber, falls die Quellinstanz Duplikate enthält, eine ergebnisäquivalente Inverse. Die Identitätsabbildung liefert eine exakte CHASE-Inverse (vgl. Abbildung 5.3). Die übrigen im vierten Kapitel vorgestellten Operationen werden hier vernachlässigt, da **ProSA** die Verarbeitung solcher Anfragen noch nicht gewährleistet.

```
if(selection == true || extremum == true) {
    sb.append("Ergebnisäquivalente CHASE-Inverse");
}
else if(aggregation == true || projection == true) {
    sb.append("Relaxte CHASE-Inverse");
}
else if(join == true) {
    sb.append("Exakte CHASE-Inverse oder Ergebnisäquivalente CHASE-Inverse");
}
else {
    sb.append("Exakte CHASE-Inverse");
}
```

Abbildung 5.3. Bestimmung des CHASE-Inversen-Typs

Die Bestimmung der Zeugenmenge erfolgt durch Auflistung der ***why***-Provenancen (vlg. Abschnitt 5.1). In **ProSA** gelingt dies durch Zugriff auf die Ergebnismenge *rs* (siehe Abbildung 5.4). Anschließend werden alle Duplikate eliminiert und die zugehörige Zeugenliste ausgegeben.

```
//witness
witness.add(rs.getString(j));
```

Abbildung 5.4. Aufstellen der Zeugenmenge

Für alle Elemente der Zeugenliste sowie alle untersuchten Tabellen werden nun Anfragen der From SELECT * FROM < Tabellenname > WHERE ID = 'Z_i' erzeugt. Vergleiche hierzu die Beispielanfragen aus Abbildung 5.5. Aufgrund ihrer automatischen Erzeugung ist die Hälfte der Anfragen überflüssig. Für kleine Datenbestände ist dies jedoch nicht problematisch.

```
select * from studenten where id_studenten = 'N20';
select * from noten where id_noten = 'N11';
```

Abbildung 5.5. Beispielanfrage zur Rekonstruktion der Quelltupel

Die Erweiterungen ermöglichen somit eine Reduzierung der Datenbank auf die für eine Anfrage Q relevanten Tupel. Die nächste Optimierung bestünde nun in der Einschränkung dieser reduzierten Datenbank auf die notwendigen Attribute. Alle nicht benötigten Attributwerte könnten demnach gelöscht und durch Nullwerte ersetzt werden.

5.3. Auswertung des Programms ProSA

ProSA verarbeitet momentan lediglich Anfragen der Form

SELECT $<$ Spaltennamen | Aggregatfunktion $|^{\star}>$
FROM $<$ Tabellenname 1 $>$ NATURAL JOIN $<$ Tabellenname 2 $>$
WHERE $<$ Bedingungen | Unteranfragen $>$.

Diese müssen zunächst um die Vereinigung, den Schnitt sowie die Differenzbildung erweitert werden. Auch die Gruppierung sowie arithmetische Operationen sind bisher noch nicht umgesetzt. Die Schwierigkeit besteht hier in der Bestimmung der Polynome für die ***how***-Provenance. Eine mögliche Berechnung für die Vereinigung, die Gruppierung sowie die Differenzbildung liefern Amsterdamer et al. in ihrem Artikel *Provenance for Aggregate Queries* [ADT11]. Sie wurden im Abschnitt 3.2 bereits theoretisch eingeführt und anhand von Beispielen diskutiert.

Die Erweiterung der CHASE-Inversen-Typen benötigt lediglich einige weitere boolesche Variablen, welche mit Hilfe der Erkenntnisse des vierten Kapitels (vgl. Tabelle 4.2) in die if-Abfolge der Abbildung 5.3 eingeordnet werden können. Etwas umfangreicher ist die Aufstellung der minimalen Zeugenbasis. Zur Zeit wird eine Zeugenliste ausgegeben, die alle notwendigen Tupel zur Rekonstruktion der Quelldatei enthält, aber nicht zwangsläufig minimal sein muss. Der Grund hierfür liegt in der Erzeugung der Zeugenliste. Sie ergibt sich derzeit aus der ***why***-Provenance; sie kann jedoch auch über die Provenance-Polynome berechnet werden. In diesem Fall kann die minimale Zeugenbasis direkt abgelesen und die Zeugenliste entsprechend gefolgert werden. Auch können die Quelltupel der Zeugeliste auf „relevante“ Attribute eingeschränkt werden, welche beispielsweise durch eine Projektion in der Anfrage verloren gehen würden.

Das Ziel bestünde somit in der Verarbeitung jeglicher Anfragen der Form

SELECT $<$ Spaltennamen | Aggregatfunktion $|^{\star}>$
FROM $<$ Tabellenname 1 $>$
JOIN | UNION | INTERSECTION | DIFFERENCE $<$ Tabellenname 2 $>$
WHERE $<$ Bedingungen | Unteranfragen $>$
GROUP BY $<$ Spaltennamen $>$.

Mit Verarbeitung ist dabei wieder die Ausgabe des CHASE-Inversen-Typs, der zugehörigen (minimalen) Zeugenbasis sowie der rekonstruierten, eingeschränkten Quellinstanz gemeint. Zusätzlich sollen auch weiterhin die ***where***-, ***why***- und ***how***-Provenance ausgegeben bzw. berechnet werden.

6. Fazit und Ausblick

Zum Abschluss der Arbeit *Umsetzung von Provenance-Anfragen in Big-Data-Analytics-Umgebungen* wollen wir eine kurze Zusammenfassung sowie einen Ausblick auf nicht behandelte Themen und Fragestellungen sowie offene Problemstellungen geben. Diese Ausarbeitung umfasst insgesamt eine Einführung in die benötigten Grundlagen (Kapitel 2), eine Darstellung des State of the Art (Kapitel 3), eine Vorstellung des in dieser Arbeit entwickelten Konzepts (Kapitel 4) sowie eine erste praktische Umsetzung (Kapitel 5).

6.1. Fazit

Die Existenz einer exakten CHASE-Inversen kann nur für einige wenige relationale Operationen (Kopieren, Umbenennung, arithmetische Operationen $+$, $-$, $\cdot$ und $:$) nachgewiesen werden. Die übrigen Operationen sind lediglich ergebnisäquivalent oder relaxt CHASE-invers. Die Tatsache, dass für beinahe alle untersuchten Grundoperationen — die Selektion auf Ungleichheit sowie die Differenzbildung sind hier nicht mit inbegriffen — eine ergebnisäquivalente CHASE-Inverse existiert, ermöglicht die Einschränkung der Quelldatenbank auf alle für die Anfrage relevanten Tupel. Eine solche Reduzierung bietet sich beispielsweise bei großen Datenbeständen an, die später nicht weiter untersucht, sondern lediglich gespeichert werden sollen.

Die Hinzunahme von zusätzlichen Provenance-Informationen ermöglicht die „Verbesserung“ einiger Inversen-Typen auf relaxte oder exakte CHASE-Inverse. Die Ergebnisse diesbezüglich sind in der unten stehenden Tabelle 6.1 zusammengefasst. Die ausführlichen Untersuchungen können in den Unterabschnitten 4.2.4 und 4.2.5 nachgelesen werden. Die benötigten Grundlagen der Provenance-Anfragen und -Antworten sowie die Konstruktion der zugehörigen Provenance-Polynome wurden in den Abschnitten 3.1 und 3.2 ausführlich eingeführt und mit Beispielen untermalt.

Die Darstellung einer Datenbankanfrage als source-to-target tuple-generating dependencies und equality-generating dependencies ermöglicht uns die Anwendung des CHASE-Algorithmus nach Artikel *Data Exchange: Semantics and Query Answering* [FKMP03]. Die Einführung einer BACKCHASE-Phase bietet nun die Nutzung der Ergebnisse einer Provenance-Analyse. Wir haben uns hierbei vom CHASE&BACKCHASE-Algorithmus aus dem Artikel *Provenance-Directed Chase&Backchase* [DH13] inspirieren lassen (siehe Abschnitt 3.3).

Für die Provenance-Theorie haben wir zudem einige neue Zusammenhänge aufstellen können:

- Definition einer Informationskette: $\boldsymbol{where} \preceq \boldsymbol{why} \preceq \boldsymbol{how}$
- Eindeutigkeit der Provenance-Polynome bei gleichen (minimalen) Zeugenbasen (siehe Satz 3.1)
- Regeln zur Ableitung der ***why***-Provenance aus der ***how***-Provenance (siehe Satz 3.3)
- Regeln zur Ableitung der ***where***-Provenance aus der ***why***-Provenance (siehe Satz 3.3)

Die Adaption von Techniken der Provenance-Anfragen ***why***, ***where*** und ***how*** in Umgebungen, die statt einfacher Anfragen wie Selektion, Projektion und Verbund auch OLAP-Operationen und weitere Machine-Learning-Algorithmen ist somit ausführlich untersucht worden. Abschließend haben unsere Erkenntnisse am Beispiel des Hidden-Markov-Modells theoretisch (siehe Unterabschnitt 4.3) und in der Erweiterung des Programms **ProSA** praktisch umgesetzt (siehe Abschnitt 5).

Operation:	Inverse Abbildung ohne Provenance-Information:	Inverse Abbildung mit Provenance-Information:
$r(\mathcal{R})$	Exakt	Exakt
$\beta_{A_j \leftarrow A_i}(r(\mathcal{R}))$	Exakt	Exakt
$\pi_{A_i}(r(\mathcal{R}))$	Relaxt Ergebnisäquivalent	Relaxt Relaxt
$r_1(\mathcal{R}_1) \bowtie r_2(\mathcal{R}_2)$	Exakt Ergebnisäquivalent	Exakt Ergebnisäquivalent
$\sigma_{A_i \theta c}(r(\mathcal{R}))$ $\sigma_{A_i \theta A_j}(r(\mathcal{R}))$ mit $\theta \in \{<, \leq, =, \geq, >\}$ $\sigma_{A_i \neq c}(r(\mathcal{R}))$ $\sigma_{A_i \neq A_j}(r(\mathcal{R}))$	Ergebnisäquivalent Ergebnisäquivalent xxx[1] xxx	Ergebnisäquivalent Ergebnisäquivalent xxx xxx
$r_1(\mathcal{R}_1) \cup r_2(\mathcal{R}_2)$ $r_1(\mathcal{R}_1) \cap r_2(\mathcal{R}_2)$ $r_1(\mathcal{R}_1) - r_2(\mathcal{R}_2)$	Ergebnisäquivalent Ergebnisäquivalent xxx	Exakt Ergebnisäquivalent xxx
MAX$(r(\mathcal{R}))$ / **MIN**$(r(\mathcal{R}))$ **COUNT**$(r(\mathcal{R}))$ **SUM**$(r(\mathcal{R}))$ **AVG**$(r(\mathcal{R}))$	Ergebnisäquivalent Relaxt xxx xxx	Ergebnisäquivalent Relaxt Exakt Exakt
$\gamma_{G_i;F_j(A_j)}(r(\mathcal{R}))$[2]	Ergebnisäquivalent Relaxt xxx	Ergebnisäquivalent Relaxt Exakt
$r(\mathcal{R})\ \theta\ \alpha$ mit $\theta \in \{+, -, \cdot, :\}$ $r(\mathcal{R})$ **MOD** α	Exakt Ergebnisäquivalent	Exakt Ergebnisäquivalent

Abkürzungen:

[1] Es existiert keine (exakte / relaxte / ergebnisäquivalente) CHASE-Inverse.

[2] Wir unterscheiden drei Fälle:

1. Fall: $F_j \in \{$**MAX**, **MIN**$\}$ $\Rightarrow$ Ergebnisäquivalente CHASE-Inverse;
2. Fall: $F_j \in \{$**COUNT**$\}$ $\Rightarrow$ Relaxte CHASE-Inverse;
3. Fall: $F_j \in \{$**SUM**, **AVG**$\}$ $\Rightarrow$
   - Es existiert keine der obigen CHASE-Inversen (ohne Provenance-Informationen).
   - Exakte CHASE-Inverse (mit Provenance-Informationen)

Tabelle 6.1. Untersuchte Operation mit zughörigen inversen Abbildungen (ohne und mit Provenance-Informationen)

6.2. Ausblick

Wir wollen unsere Untersuchungen mit einem Ausblick nicht behandelter Themen und Fragestellungen abschließen. Hierbei handelt es sich um eine Auflistung offener Problemstellungen und noch näher zu untersuchender Einzelheiten, die den Umfang dieser Arbeit überschritten hätten. Sie ergaben sich sowohl bei der Literaturrecherche als auch beim Bearbeiten der Themenstellung.

- Der in Satz 3.3 behauptete Zusammenfang zwischen den Provenance-Anfragen ***how***, ***why*** und ***where*** muss bewiesen und auf Provenance-Polynome mit Aggregaten erweitert werden.
- Die Formulierung der Aggregation und Gruppierung als s-t tgds und egds wurde in dieser Arbeit aufgrund ihres Umfangs vernachlässigt. Diese müssten noch einmal genauer untersucht werden. Die Grundlage hierfür bildet der Artikel *Aggregation in Datalog Under Set Semantics* von A. Mohapatra und M. Genesereth [MG12].
- Die Theorie der Provenance-Polynome basiert auf einer positiv relationalen Algebra. Die Selektion auf Ungleichheit sowie die Differenzbildung zählen aber nicht zu dieser Algebra. Die Aufstellung der zugehörigen Polynome ist daher mit einigen Schwierigkeiten verbunden. Erste Ansätze hierzu liefern etwa die Artikel *Provenance for Aggregate Queries* [ADT11] und *The Semiring Framework for Database Provenance* [GT17].
- Der CHASE muss für nicht-positive relationale Algebren nicht funktionieren. Die Existenz von CHASE-Inversen konnte für die Selektion auf Ungleichheit sowie die Differenzbildung daher nicht untersucht werden.
- Neben extensionalen Antworten sind im Bereich der Provenance-Untersuchungen natürlich auch intensionale Antworten möglich. Mit der Fragestellung der besseren Beschreibung der üblicherweise sehr großen und kompexen Provenacne-Antworten beschäftigen sich unter anderem D. Deutsch, N. Forst und A. Gilad in ihrem Artikel *Provenance for Natural Language Queries* [DFG17].
- Kann für ein Ergebnistupel das zugehörige Provenance-Polynom berechnet werden, können hierüber auch Aussagen über die ***why not***-Provenance getroffen werden. So können im Polynom fehlende bzw. nicht berücksichtige Tupelidentifikatoren oder aber die Zeugenliste selbst zurückgegeben werden. So ist erkennbar, ob ein Quelltupel bei der Ergebnisberechnung berücksichtigt wurde oder nicht.
- Die Bestimmung der minimalen Zeugenbasis gelingt uns bisher nur durch das Ausprobieren verschiedener Anfragen. Eine konkrete Methode ist uns zur Zeit nicht bekannt. Ein Algorithmus zur Bestimmung der minimalen Zeugenbasis könnte aber wie folgt aussehen: Seine eine Anfrage Q auf eine Datenbank d gegeben.

  1. Optimiere die Anfrage Q mittels logischer oder interner Optimierung
  2. Aufstellen einer Zeugenbasis
  3. Berechnung der zugehörigen Zeugenliste
  4. Entsprecht die Zeugenliste den Tupelidentifikatoren der ***where***-Provenance, so ist die Zeugenbasis minimal. Falls die ***where***-Provenance über die Relationennamen definiert ist, müssen die Tupelidentifikaoren zunächst aus der ***why***-Provenance bestimmt werden.

- Enthält eine Datenbank Nullwerte, kann im Falle des **FULL OUTER JOIN** die Existenz einer exakten CHASE-Inversen nicht garantiert werden.

- Das ProSA berechnet die Provenance-Polynome für SQL-Anweisungen der Form

  SELECT $<$ Spaltennamen | Aggregatfunktion $|^{\star}>$
  FROM $<$ Tabellenname 1 $>$ NATURAL JOIN $<$ Tabellenname 2 $>$
  WHERE $<$ Bedingungen | Unteranfragen $>$.

  Diese können noch um die allgemeine Verbünde, Vereinigung, Schnitt- und Differenzbildung — sobald hier theoretische Überlegungen zu Berechnung der zugehörigen Polynome existieren — sowie die Gruppierung erweitert werden. Zudem kann die Bestimmung der Zeugenbasis auf eine minimale Zeugenbasis spezialisiert werden. Auch sollte die Zeugenbasis nicht aus der ***why***-Provenance, sondern aus der ***how***-Provenance berechnet werden. So wird die explizite Untersuchung der ***why***-Povenance überflüssig (vgl. auch Beispiel 3.6).

- Für die Existenz einer exakten CHASE-Inversen reicht die Untersuchung der Provenance teilweise nicht aus. Für einige Operationen müssen wir uns zudem bestimmte Attributwerte oder sogar ganze Tupel merken. Sie werden wegselektiert, gehen bei der Durchschnittsbildung oder dem natürlichen Verbund als dangling tuples verloren oder werden im Fall der Aggregatfunktionen **MAX**, **MIN** sowie **COUNT** zu stark verdichtet. Tabelle 6.2 zeigt, für welche Operationen weitere Informationen benötigt werden. Dabei steht ein x für die Notwendigkeit und $-$ für die Nicht-Notwendigkeit dieser zusätzlichen Informationen. Im Falle der Attribute und Tupel handelt es sich hierbei jedoch um noch nicht bewiesene Vermutungen. Es bleiben daher noch einige Leerfelder zu untersuchen. Der nächste Schritt bestünde nun darin, den CHASE so zu erweitert, dass er diese Menge von zusätzlichen Attributwerten (Projektion, Summation) bzw. Tupeln (Natürlicher Verbund, Selektion, Schnittmenge, Maximum- / Minimumsuche, Zählung) interpretieren und verarbeiten kann.

Operation	keine	Provenance	Attribute	Tupel
Kopieren / Identitätsabbidlung	x	-	-	-
Umbenennung	x	-	-	-
Projektion	-	-	x	
Natürlicher Verbund	-	-		x
Selektion	-	-		x
Vereinigung	-	x	-	-
Schnittmenge	-	-		x
Maximum- / Minimumbildung	-	-		x
Zählung	-	-		x
Summation	-	x	x	
Arithmetischer Durchschnitt	-	-		
Gruppierung	-	x	x	x
Skalare Operatoren $+, -, \cdot$:	x	-	-	-
Modulorechnung	-	-		

Tabelle 6.2. Benötigte Informationen für die Existenz einer exakten CHASE-Inversen

Literaturverzeichnis

[ABU79] Aho, Alfred V. ; Beeri, Catriel ; Ullman, Jeffrey D.: The Theory of Joins in Relational Databases. In: *ACM Trans. Database Syst.* 4 (1979), Nr. 3, 297–314. http://dx.doi.org/10.1145/320083.320091. – DOI 10.1145/320083.320091

[ADT11] Amsterdamer, Yael ; Deutch, Daniel ; Tannen, Val: Provenance for Aggregate Queries. In: *PODS*, ACM, 2011, S. 153–164

[Aug17] Auge, Tanja: *Ringvorlesung: Forschung @ DBIS.* https://eprints.dbis.informatik.uni-rostock.de/827/, Wintersemester 2016/2017

[Bal17] Balazinska, Magda: *Introduction to Data Management CSE 344 — Lecture 10: Datalog.* https://courses.cs.washington.edu/courses/cse344/11au/lectures/lecture10-datalog.txt, 02.09.2017

[BKM+17] Bruder, Ilvio ; Klettke, Meike ; Möller, Mark L. ; Meyer, Frank ; Heuer, Andreas ; Jürgensmann, Susanne ; Feistel, Susanne: Daten wie Sand am Meer - Datenerhebung, -strukturierung, -management und Data Provenance für die Ostseeforschung. In: *Datenbank-Spektrum* 17 (2017), Nr. 2, 183–196. http://dx.doi.org/10.1007/s13222-017-0259-4. – DOI 10.1007/s13222–017–0259–4

[BKT01] Buneman, Peter ; Khanna, Sanjeev ; Tan, Wang C.: Why and Where: A Characterization of Data Provenance. In: *ICDT* Bd. 1 Springer, 2001, S. 316–330

[BMSS15] Bruder, Ilvio ; Meyer, Holger ; Schering, Alf-Christian ; Schmitt, Christoph: Das Projekt WossiDiA: Digitalisierung des Wossidlo-Archivs. In: *Digitales Kulturerbe: Bewahrung und Zugänglichkeit in der wissenschaftlichen Praxis* 2 . KIT Scientific Publishing (2015), S. 61–80

[Bos13] Bosch, Siegfried: *Algebra, 8. Auflage.* Springer Spektrum, 2013

[Bos14] Bosch, Siegfried: *Lineare Algebra, 5. Auflage.* Springer Spektrum, 2014

[Bro82] Brodie, Michael L.: On the Development of Data Models. In: *On Conceptual Modelling (Intervale)*, 1982, S. 19–47

[CCT09] Cheney, James ; Chiticariu, Laura ; Tan, Wang C.: Provenance in Databases: Why, How, and Where. In: *Foundations and Trends in Databases* 1 (2009), Nr. 4, S. 379–474

[CER17] CERN: *Datenverarbeitung: Detektor-Triggersystem.* http://www.lhc-facts.ch/index.php?page=datenverarbeitung, Referenziert am 06.05.2017

[Cod70] Codd, E. F.: A Relational Model of Data for Large Shared Data Banks. In: *Communications of the ACM* 13 (1970), Nr. 6, S. 377–387

[Cod17] Code Project: *Visual Representation of SQL Joins.* https://www.codeproject.com/Articles/33052/Visual-Representation-of-SQL-Joins, 26.0802017

[CWW00] Cui, Yingwei ; Widom, Jennifer ; Wiener, Janet L.: Tracing the Lineage of View Data in a Warehousing Environment. In: *ACM Transactions on Database Systems* 25 (2000), Nr. 2, S. 179–227

[DFG17] Deutch, Daniel ; Frost, Nave ; Gilad, Amir: Provenance for Natural Language Queries. In: *PVLDB* 10 (2017), Nr. 5, 577–588. http://www.vldb.org/pvldb/vol10/p577-deutch.pdf

[DH13] Deutsch, Alin ; Hull, Richard: Provenance-Directed Chase&Backchase. In: *In Search of Elegance in the Theory and Practice of Computation* Bd. 8000, Springer, 2013 (Lecture Notes in Computer Science), S. 227–236

[DHI12] Doan, AnHai ; Halevy, Alon Y. ; Ives, Zachary G.: *Principles of Data Integration*. Morgan Kaufmann, 2012

[Dij13] Dijcks, Jean-Pierre: Oracle: Big Data for the Enterprise. In: *Oracle White Paper* (2013)

[Fag07] Fagin, Ronald: Inverting Schema Mappings. In: *ACM Transaction Database Systems* 32 (2007), Nr. 4, 25. http://dx.doi.org/10.1145/1292609.1292615. – DOI 10.1145/1292609.1292615

[FBH+14] Finger, Andreas ; Bruder, Ilvio ; Heuer, Andreas ; Klemkow, Martin ; Konerow, Steffen: PageBeat — Zeitreihenanalyse und Datenbanken. In: *Grundlagen von Datenbanken* Bd. 1313, CEUR-WS.org, 2014 (CEUR Workshop Proceedings), S. 53–58

[FKMP03] Fagin, Ronald ; Kolaitis, Phokion G. ; Miller, Renée J. ; Popa, Lucian: Data Exchange: Semantics and Query Answering. In: *ICDT* Bd. 2572, Springer, 2003 (Lecture Notes in Computer Science), S. 207–224

[FKPT08] Fagin, Ronald ; Kolaitis, Phokion G. ; Popa, Lucian ; Tan, Wang C.: Quasi-Inverses of Schema Mappings. In: *ACM Transaction of Database Systems* 33 (2008), Nr. 2, 11:1–11:52. http://dx.doi.org/10.1145/1366102.1366108. – DOI 10.1145/1366102.1366108

[FKPT11] Fagin, Ronald ; Kolaitis, Phokion G. ; Popa, Lucian ; Tan, Wang C.: Schema Mapping Evolution Through Composition and Inversion. In: *Schema Matching and Mapping*. Springer, 2011 (Data-Centric Systems and Applications), S. 191–222

[Gal17] Galetto, Molly: *MACHINE LEARNING AND BIG DATA ANALYTICS: THE PERFECT MARRIAGE*. https://www.ngdata.com/machine-learning-and-big-data-analytics-the-perfect-marriage/, 01.09.2017

[GH16a] Grunert, Hannes ; Heuer, Andreas: Datenschutz im PArADISE. In: *Datenbank-Spektrum* 16 (2016), Nr. 2, S. 107–117

[GH16b] Grunert, Hannes ; Heuer, Andreas: Privacy Protection through Query Rewriting in Smart Environments. In: *EDBT*, OpenProceedings.org, 2016, S. 708–709

[GKT07] Green, Todd J. ; Karvounarakis, Gregory ; Tannen, Val: Provenance semirings. In: *PODS*, ACM, 2007, S. 31–40

[GT17] Green, Todd J. ; Tannen, Val: The Semiring Framework for Database Provenance. In: Sallinger, Emanuel (Hrsg.) ; Bussche, Jan V. (Hrsg.) ; Geerts, Floris (Hrsg.): *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2017, Chicago, IL, USA, May 14-19, 2017*, ACM, 2017. – ISBN 978–1–4503–4198–1, 93–99

[Hal01] Halevy, Alon Y.: Answering Queries Using Views: A Survey. In: *VLDB J.* 10 (2001), Nr. 4, 270–294. http://dx.doi.org/10.1007/s007780100054. – DOI 10.1007/s007780100054

[Hem17] Hemetsberger, Paul: *Deutsch-Englisch-Wörterbuch.* http://www.dict.cc/?s=provenance, Referenziert am 07.03.2017

[Her15] Herschel, Melanie: A Hybrid Approach to Answering Why-Not Questions on Relational Query Results. In: *J. Data and Information Quality* 5 (2015), Nr. 3, 10:1–10:29. http://dx.doi.org/10.1145/2665070. – DOI 10.1145/2665070

[Heu15] Heuer, Andreas: METIS in PArADISE Provenance Management bei der Auswertung von Sensordatenmengen für die Entwicklung von Assistenzsystemen. In: *BTW Workshops* Bd. 242, GI, 2015 (LNI), S. 131–136

[Heu16a] Heuer, Andreas: *Neueste Entwicklungen in der Informatik — Provenance.* Lehrstuhl für Datenbank- und Informationssysteme, Institut für Informatik, Universität Rostock, Vorlesungsfolien, Sommersemester 2016

[Heu16b] Heuer, Andreas: *Theorie Relationaler Datenbanken.* Lehrstuhl für Datenbank- und Informationssysteme, Institut für Informatik, Universität Rostock, Vorlesungsfolien, Sommersemester 2016

[Heu17] Heuer, Andreas: *Digitale Bibliotheken und Multimedia-Information-Retrieval.* Lehrstuhl für Datenbank- und Informationssysteme, Institut für Informatik, Universität Rostock, Vorlesungsfolien, Wintersemester 2016/2017

[HM15] Heuer, Andreas ; Meyer, Holger: *Das PArADISE-Projekt — Big-Data-Analysen für die Entwicklung von Assistenzsystemen.* Technischer Bericht des Instituts für Informatik der Universität Rostock, 2015

[JM17] Jurafsky, Dan ; Martin, James H.: *Hidden Markov Models.* 26.08.2017

[KSS14] Köppen, Veit ; Saake, Gunter ; Sattler, Kai-Uwe: *Data Warehouse Technologien, 2. Auflage.* MITP, 2014

[Leh17] Lehrstuhl für Datenbank- und Informationssysteme, Universität Rostock: *HyDRA.* https://dbis.informatik.uni-rostock.de/forschung/langzeitrahmenprojekte/hydra/, Referenziert am 31.05.2017

[LKLG17] Lee, Seokki ; Köhler, Sven ; Ludäscher, Bertram ; Glavic, Boris: A SQL-Middleware Unifying Why and Why-Not Provenance for First-Order Queries. In: *33rd IEEE International Conference on Data Engineering, ICDE 2017, San Diego, CA, USA, April 19-22, 2017*, 2017, S. 485–496

[Mai83] Maier, David: *The Theory of Relational Databases.* Computer Science Press, 1983

[Mel11] Melton, Jim: *Information technology — Database languages — SQL — Part 2: Foundation (SQL/Foundation).* 21.12.2011

[MG12] Mohapatra, Abhijeet ; Genesereth, Michael: *Aggregation in Datalog Under Set Semantics.* 2012

[MH15] Marten, Dennis ; Heuer, Andreas: A framework for self-managing database support and parallel computing for assistive systems. In: *PETRA*, ACM, 2015, S. 25:1–25:4

[MH17] Marten, Dennis ; Heuer, Andreas: Machine Learning on Large Databases: Transforming Hidden Markov Models to SQL Statements. In: *Open Journal of Databases (OJDB)* 4 (2017), Nr. 1, 22–42. https://www.ronpub.com/ojdb/OJDB_2017v4i1n02_Marten.html. – ISSN 2199–3459

[OVH17] OVH.de: *OVH Big Data Lexikon.* https://www.ovh.de/dedicated_server/HG/big-data-glossar.xml, Referenziert am 05.03.2017

[RN04] Russell, Stuart J. ; Norvig, Peter: *Künstliche Intelligenz - ein moderner Ansatz (2. Aufl.).* Pearson Studium, 2004 http://www.pearson-studium.de/main/main.asp?page=bookdetails&ProductID=47105&SID={BFAE885F-170C-4D9C-8410-C3F715F1524E}&TOKEN={57E2ED41-C18E-4F93-8779-A38ED30290BD}. – ISBN 978–3–8273–7089–1

[RSSL13] Richter, Anja ; Stäuble, Harald ; Steinmann, Christoph ; Lange, Jan-Michael: Petrographical investigations and provenance analyses of the raw materials of Neolithic stone tools from different localities southeast of Leipzig. In: *GEOLOGICA SAXONICA — JOURNAL OF CENTRAL EUROPEAN GEOLOGY* 59 (2013), 193–204. http://www.senckenberg.de/files/content/forschung/publikationen/geologicasaxonica/59/geologica-saxonica59_2013_richter.pdf

[RU17] Ruscheinski, Andreas ; Uhrmacher, Adelinde M.: Provenance in Modeling and Simulation Studies - Bridging Gaps. In: *Winter Simulation Conference 2017*, 2017

[Rud09] Rudin, Walter: *Analysis, 4. Auflage.* Oldenbourg Verlag München, 2009

[Sal16] Salzig, Christoph: *Was ist Big Data? - Eine Definition mit fünf V.* https://blog.unbelievable-machine.com/was-ist-big-data-definition-fünf-v, Referenziert am 04.05.2016

[SSH13] Saake, Gunter ; Sattler, Kai-Uwe ; Heuer, Andreas: *Datenbanken — Konzepte und Sprachen, 5. Auflage.* MITP, 2013

[Sva16] Svacina, Jan: *Intensional Answers for Provenance Queries in Big Data Analytics.* Lehrstuhl für Datenbank- und Informationssysteme, Institut für Informatik, Universität Rostock, Bachelorarbeit, 2016

[WS13] Waldmann, Karl-Heinz ; Stocker, Ulrike M.: *Stochastiche Modelle — Eine anwendungsorientierte Einführung, 2. Auflage.* Springer-Verlag Berlin Heidelberg, 2013

Anfragenverzeichnis

2.1. Anzahl an Modulteilnehmern 22
2.2. Durchschnittsnoten der einzelnen Module 22
2.3. Anzahl bestandener Prüfungen in einem Zeitfenster von 7 Tagen 25

3.1. Noten der Studenten mit Vornamen 'Sarah' (Anfrage Q_1) 37
3.2. Noten des Studenten mit Matrikelnummer 2 (Anfrage Q_2) 38
3.3. Modulbelegung eines Studenten unter gegebenen Nebenbedingungen (Anfrage Q_3) 41
3.4. Durchschnittsnote der Studenten mit Vornamen 'Max' (Anfrage Q_4) 48
3.5. Jeweilige Durchschnittsnote des Studenten mit Vornamen 'Max' (Anfrage Q_5) 50
3.6. Prüfungsergebnisse der Studenten mit Vornamen 'Max' (Anfrage Q_6) 53

4.1. Die OLAP-Operation GROUP BY CUBE 76
4.2. Die OLAP-Operation GROUP BY GROUPING SETS 76
4.3. Die OLAP-Operation GROUP BY 76
4.4. Matrix-Matrix-Multiplikation 107
4.5. Addition mit expliziten Nullwerten 107
4.6. Addition mit vernachlässigten Nullwerten 108
4.7. Forward Algorithmus 112

5.1. Prüfungsergebnisse der Studenten mit Vornamen 'Max' (Anfrage Q_6) 113

C.1. Provenance-Berechnung 143
C.2. Typbestimmung der CHASE-Inversen 147
C.3. Aufstellen der Anfragen zur Rekonstruktion der Quelldatenbank 148

Tabellenverzeichnis

1.1. STUDENTEN-Relation 13

1.2. NOTEN-Relation eingeschränkt auf das Modul 009 (Ausschnitt der Tabelle B.5) 13

1.3. MODUL-Relation 13

1.4. DOZENTEN-Relation 13

1.5. TEILNEHMER-Relation eingeschränkt auf das Modul 009 (Ausschnitt der Tabelle B.4) . 14

2.1. Ergebnis der Teilanfrage $\pi_{\text{Modulnr,Semester}}(r)$ 19

2.2. Ergebnis der Teilanfrage $\pi_{\text{Modulnr.,Dozent}}(r)$ 19

2.3. Ergebnis der Anfrage $\pi_{\text{Modulnr.,Semester}}(r) \bowtie \pi_{\text{Modulnr.,Dozent}}(r)$ 19

2.4. Durchschnittsnoten der einzelnen Module 23

2.5. Schema S der STUDENTEN-Tabelle B.1 29

2.6. Schema S' der STUDENTEN-Tabelle B.1 29

2.7. Verteilung von Titel, Vor- und Nachname sowie middle initial auf verschiedene Namensattribute 29

2.8. Beispieltableau nach Definition 2.17 30

2.9. Tableau eines Datenbankschemas S 31

2.10. Beispiel einer Tableauanfrage 32

2.11. Beispiel für die Erweiterung eines Tableaus um Tags 33

2.12. Ausgangstableau für CHASE-Algorithmus 34

2.13. CHASE-Algorithmus: links: Anwendung der J-Regel; rechts: Anwendung der F-Regel . . . 34

2.14. Ergebnistableau nach CHASE-Algorithmus 34

3.1. Provenance-Anfragen 35

3.2. Provenance-Resultate der Anfragen Q 36

3.3. Provenance-Antworten [Heu16a] 36

3.4. Prüfungsergebnisse der Studentin Sarah nach Anfrage 3.1 38

3.5. Prüfungsergebnisse der Studentin Sarah mit Provenance-Information 39

3.6. Prüfungsergebnisse der Studentin Sarah mit Provenance-Informationen für die Anfragen 3.1 und 3.2 39

3.7. Zwischenergebnis $\pi_{A_1A_2}(\sigma_{A_4=N_1}(M) \bowtie D)$ der Anfrage 3.3 (mit Provenance-Informationen) 42

3.8. Zwischenergebnis $\pi_{A_1A_2}(M \bowtie \sigma_{A_5=N_2}(D))$ der Anfrage 3.3 (mit Provenance-Informationen) 42

3.9. Zwischenergebnis $\pi_{A_1}(\sigma_{A_3=5}(T))$ der Anfrage 3.3 (mit Provenance-Informationen) 43

3.10. ***how***-Provenance der Anfrage 3.3 mit zugehörigen Provenance-Polynomen 43

3.11. ***how***-Provenance der zu Q_3 äquivalenten Anfragen Q_3' und Q_3'' mit zugehörigen Provenance-Polynomen . . . 44

3.12. Ergebnis der Anfrage Q mir Provenance-Informationen . . . 46

3.13. Zwischenergebnis $Z_{Q_4} = \pi_{\text{Matrikelnr,Note}}(\text{STUDENTEN} \bowtie \text{NOTEN})$ der Anfrage 3.4 . . . 48

3.14. Ergebnis der Anfrage 3.4 . . . 48

3.15. Zwischenergebnis $Z_{Q_5} = \pi_{\text{Matrikelnr,Note}}(\text{STUDENTEN} \bowtie \text{NOTEN})$ der Anfrage 3.5 . . . 50

3.16. Ergebnis der Anfrage 3.5 . . . 51

3.17. Überblick über einige CHASE-Varianten . . . 52

3.18. Ergebnistabelle der Anfrage 3.6 . . . 54

3.19. Zwischenergebnis der Tableaukonstruktion . . . 54

3.20. Zwischenergebnis der Tableaukonstruktion . . . 54

3.21. Codierung der Anfrage 3.6 als Tableau . . . 55

3.22. Codierung der Anfrage 3.6 als Tableau für s-t tgds . . . 55

4.1. I'' ist Ausschnitt von I, d.h. $I'' \preccurlyeq I$ . . . 74

4.2. Untersuchte Operation mit zughörigen inversen Abbildungen (ohne und mit Provenance-Informationen) . . . 78

4.3. Beispiel für eine inverse Schemaabbildung der Operation Kopieren . . . 79

4.4. Beispiel für eine inverse Schemaabbildung der Operation β . . . 80

4.5. Beispiel für eine inverse Schemaabbildung der Operation π . . . 80

4.6. Zweites Beispiel für eine inverse Schemaabbildung der Operation π . . . 81

4.7. Beispiel für eine inverse Schemaabbildung der Operation $\bowtie$ . . . 82

4.8. Zweites Beispiel für eine inverse Schemaabbildung der Operation $\bowtie$ . . . 82

4.9. Beispiel für eine inverse Schemaabbildung der Operation $\sigma_{A_1=2}$ . . . 83

4.10. Zweites Beispiel für eine inverse Schemaabbildung der Operation $\sigma_{A_2<3}$ . . . 83

4.11. Beispiel für eine inverse Schemaabbildung der Operation $\sigma_{A_1=A_2}$ . . . 84

4.12. Zweites Beispiel für eine inverse Schemaabbildung der Operation $\sigma_{A_1<A_2}$ . . . 84

4.13. Beispiel für eine inverse Schemaabbildung der Operation $\cup$ . . . 84

4.14. Beispiel für eine inverse Schemaabbildung der Operation $\cap$ . . . 85

4.15. Beispiel für eine inverse Schemaabbildung der Operation $-$ . . . 85

4.16. Beispiel für eine inverse Schemaabbildung der Operation **MAX** . . . 86

4.17. Beispiel für eine inverse Schemaabbildung der Operation **COUNT** . . . 86

4.18. Beispiel für eine inverse Schemaabbildung der Operation **SUM** . . . 87

4.19. Beispiel für eine inverse Schemaabbildung der Operation **AVG** . . . 87

4.20. Beispiel für eine inverse Schemaabbildung der Operation $\gamma_{A_1;\mathbf{MAX}(A_2)}r(\{A_1, A_2\})$ . . . 88

4.21. Zweites Beispiel für eine inverse Schemaabbildung der Operation $\gamma_{A_1;\mathbf{SUM}(A_2)}r(\{A_1, A_2\})$ 88

4.22. Beispiel für eine inverse Schemaabbildung der arithmetischen Operation $+$ . . . 89

4.23. Beispiel für eine inverse Schemaabbildung der arithmetischen Operation **MOD** 3 . . . 89

4.24. Beispiel für eine inverse Schemaabbildung mit Provenance-Informationen der Operation Kopieren . . . 90

4.25. Beispiel für eine inverse Schemaabbildung der Operation Kopieren . . . 90

4.26. Beispiel für eine inverse Schemaabbildung mit Provenance-Informationen der Operation π 91

4.27. Zweites Beispiel für eine inverse Schemaabbildung mit Provenance-Informationen der Operation π . . . 91

4.28. Beispiel für eine inverse Schemaabbildung mit Provenance-Informationen der Operation $\bowtie$ 92

4.29. Zweites Beispiel für eine inverse Schemaabbildung mit Provenance-Informationen der Operation $\bowtie$ . . . 92

4.30. Beispiel für eine inverse Schemaabbildung mit Provenance-Informationen der Operation $\sigma_{A_1=2}$ . . . 93

4.31. Beispiel für eine inverse Schemaabbildung mit Provenance-Informationen der Operation $\sigma_{A_1=A_2}$ . . . 93

4.32. Beispiel für eine inverse Schemaabbildung mit Provenance-Informationen der Operation $\cup$ 94

4.33. Beispiel für eine inverse Schemaabbildung mit Provenance-Informationen der Operation $\cap$ 95

4.34. Beispiel für eine inverse Schemaabbildung mit Provenance-Informationen der Operation **MAX** . . . 96

4.35. Beispiel für eine inverse Schemaabbildung mit Provenance-Informationen der Operation **COUNT** . . . 96

4.36. Beispiel für eine inverse Schemaabbildung mit Provenance-Informationen der Operation **SUM** . . . 96

4.37. Beispiel für eine inverse Schemaabbildung mit Provenance-Informationen der Operation **AVG** . . . 97

4.38. Beispiel für eine inverse Schemaabbildung mit Provenance-Informationen der Operation $\gamma_{A_1;\mathbf{MAX}(A_2)}r(\{A_1, A_2\})$ . . . 97

4.39. Zweites Beispiel für eine inverse Schemaabbildung mit Provenance-Informationen der Operation $\gamma_{A_1;\mathbf{SUM}(A_2)}r(\{A_1, A_2\})$ . . . 98

4.40. Beispiel für eine inverse Schemaabbildung mit Provenance-Informationen der arithmetischen Operation + . . . 98

4.41. Beispiel für eine inverse Schemaabbildung der arithmetischen Operation **MOD** 3 . . . 99

4.42. Beispiel für eine inverse Schemaabbildung einer Anfrage Q . . . 101

4.43. Beispiel für eine inverse Schemaabbildung einer Anfrage Q mit Provenance-Informationen 102

4.44. Grundoperation und ihre (exakten / relaxten / ergebnisäquivalenten) CHASE-Inversen . 104

4.45. Beispiel für die inverse Schemaabbildung der skalaren Multiplikation . . . 108

4.46. CHASE-inverse Schemaabbildungen für die verschiedenen Verbund-Typen . . . 109

4.47. Beispiel für eine inverse Schemaabbildung der Operation **INNER JOIN** . . . 110

4.48. Beispiel für eine inverse Schemaabbildung der Operation **OUTER JOIN** . . . 110

4.49. Beispiel für eine inverse Schemaabbildung der Matrix-Matrix-Multiplikation . . . 111

5.1. Anfrageergebnisse erweitert um ***where***-, ***why***- und ***how***-Provenance . . . 113

6.1. Untersuchte Operation mit zughörigen inversen Abbildungen (ohne und mit Provenance-Informationen) . . . 118

6.2. Benötigte Informationen für die Existenz einer exakten CHASE-Inversen 120

B.1. STUDENTEN-Relation 139
B.2. MODUL-Relation 139
B.3. DOZENTEN-Relation 140
B.4. TEILNEHMER-Relation 140
B.5. NOTEN-Relation 141

Symbolverzeichnis

X, Y	Attributmenge
A_i	Attribut
$\mathcal{B}$	Integritätsbedingungen
$\underline{\text{Const}}$	Menge der Konstanten
G	Gruppe
I	Instanz
K	Körper
$\mathcal{K}$	Schlüsselmenge
M	Modul
$\mathcal{M}$	Schemaabbildung
$\mathbb{N}^+$	Menge der natürlichen Zahlen ohne 0
$\mathbb{N}$	Menge der natürlichen Zahlen
Q	Anfrage
R	Ring
$\mathcal{R}$	Relationenschema
Σ	Menge von Abhängigkeiten
S	Datenbankschema
T	Tableau
$\mathcal{U}$	Universalplan
$\underline{\text{Var}}$	Menge der (markierte) Nullvariablen
V_a	Menge der ausgezeichneten Variablen
V_b	Menge der nichtausgezeichneten Variablen
V	Menge von Sichten
$W_{Q,d}(t)$	Zeugenbasis
W	Zeugenmenge
a_j	ausgezeichnete Variable
β	Umbenennung
b_k	nichtausgezeichnete Variable
$\bowtie$	Natürlicher Verbund
δ	egd oder s-t tgd

$-$	Differenz
$\mathbb{D}$	Domäne
d	Datenbank
$\emptyset$	Leere Menge oder leere Abbildung
h	Homomorphismus
$\varphi(x)$	Konjunktion von Atomen des Quellschemas
π	Projektion
$\psi(x,y)$	Konjunktion von Atomen des Zielschemas
ρ	K-Relation
$r(\mathcal{R})$	Relation über einem Relationenschema $\mathcal{R}$
$\cap$	Schnitt
σ	Selektion
τ	Menge der Tupelidentifikatoren
t	Tupel
$\cup$	Vereinigung

Abkürzungsverzeichnis

egd	equality-generating dependency
FD	Funktionale Abhängigkeit
GAV	global-as-view-Einschränkung
GLAV	global-and-local-as-view
HMM	Hidden-Markov-Modell
ID	ldentifikationsnummer
JD	Verbundabhängigkeit
LAV	local-as-view-Einschränkung
s-t tgd	source-to-target tuple-generating dependency
tgd	tuple-generating dependency

A. Mathematische Grundlagen

Die folgenden Definitionen und Beispiele basieren auf den Lehrbüchern *Lineare Algebra* und *Algebra* von S. Bosch, siehe [Bos13] und [Bos14]. Es handelt sich hierbei um die grundlegenden Definitionen der Lineare Algebra wie *Monoide*, *Gruppen*, *Ringe*, *Module* und *Körper*.

Definition A.1 (Monoid)**.** Ein **Monoid** ist ein Paar $(M, \star)$ bestehend aus einer Menge M und einer inneren zweistelligen Verknüpfung $\star : M \times M \to M, (x, y) \mapsto x \star y$ mit den folgenden Eigenschaften:

- Abgeschlossenheit: $\forall x, y \in M : x \star y \in M$
- Assoziativität: $\forall x, y, z \in M : (x \star y) \star z = x \star (y \star z)$
- Neutrales Element: $\forall x \in M : \exists e \in G : e \star x = x = x \star e$

□

Beispiel A.1. *Die ganzen Zahlen $\mathbb{Z}$ bilden zusammen mit der Addition $+$ einen Monoid, zusammen mit der Subtraktion $-$ allerdings nicht. Die Assoziativität ist verletzt.*

Definition A.2 (Gruppe)**.** Ein Paar $(G, \star)$ mit einer Menge G und einer inneren zweistelligen Verknüpfung $\star : G \times G \to G, (x, y) \mapsto x \star y$ heißt **Gruppe**, wenn sie den folgende Eigenschaften genügt:

- Abgeschlossenheit: $\forall x, y \in G : x \star y \in G$
- Assoziativität: $\forall x, y, z \in G : (x \star y) \star z = x \star (y \star z)$
- Neutrales Element: $\exists e \in G : \forall x \in G : e \star x = x = x \star e$
- Inverses Element: $\forall x \in G : \exists x^{-1} \in G : x \star x^{-1} = e = x^{-1} \star x$

Eine Gruppe heißt **abelsch**, wenn die Verknüpfung $\star$ kommutativ ist, d.h.

$$\forall x, y \in G : x \star y = y \star x.$$

□

Kurz gesagt: Eine Gruppe ist ein Monoid G, so dass jedes Element von G ein inverses Element besitzt.

Beispiel A.2. *Die ganzen Zahlen $\mathbb{Z}$ bilden zusammen mit der Addition $+$ eine abelsche Gruppe.*

Definition A.3 (Ring)**.** Ein **Ring** (mit Eins) ist eine Menge R mit zwei inneren Verknüpfungen, geschrieben als Addition $+$ und Multiplikation $\cdot$, so dass die folgenden Bedingungen erfüllt sind:

- R ist eine abelsche Gruppe bzgl. der Addition $+$
- R ist ein Monoid bzgl. der Multiplikation $\cdot$
- Kommutativität bzgl. $+$

- Distributivgesetze: $\forall x, y, z \in G : (x + y) \cdot z = x \cdot z + y \cdot z$ und $z \cdot (x + y) = z \cdot x + z \cdot y$

Ein Ring heißt **kommutativ**, wenn die Multiplikation $\cdot$ kommutativ ist. $\boxdot$

Das neutrale Element der Addition wird in der Regel mit 0 und das der Multiplikation 1 bezeichnet. Dabei ist $1 = 0$ möglich, allerdings nur im Nullring (Ring bestehend aus nur einem Element: der Null).

Beispiel A.3. *Der Ring $\mathbb{Z}$ der ganzen Zahlen ist unter der gewöhnlichen Addition $+$ und der gewöhnlichen Multiplikation $\cdot$ ein kommutativer Ring mit Eins.*

Definition A.4 (Modul)**.** Sei R ein kommutativer Ring mit Eins. Dann ist ein **R-Modul** eine Menge M mit einer inneren Verknüpfung $M \times M \to M, (x, y) \mapsto x + y$ sowie einer äußeren Verknüpfung $R \times M \to M, (\alpha, x) \mapsto \alpha \cdot x$ — genannt skalare Multiplikation — so dass gilt:

- M ist eine abelsche Gruppe bzgl. der Addition $+$
- Abgeschlossenheit bzgl. der skalaren Multiplikation $\cdot$: $\forall \alpha \in R : \forall x \in M : \alpha \cdot x \in M$
- Distributivgesetze: $\forall \alpha \in R : \forall x, y \in M : a \cdot (x + y) = \alpha \cdot x + \alpha \cdot y$ und
- Assoziativität: $\forall \alpha, \beta \in R : \forall x \in M : \alpha \cdot (\beta \cdot x) = (\alpha \cdot \beta) \cdot x$
- Neutrales Element: $\exists e \in M : \forall x \in M : e \cdot x = x$.

$\boxdot$

Beispiel A.4. *Jede kommutative Gruppe G ist ein $\mathbb{Z}$-Modul. Dabei gilt für $n \in \mathbb{N}$ und $g \in G$:*

$$n \cdot g = \underbrace{g + ... + g}_{n-fach} \text{ und } (-n) \cdot g = -(n \cdot g).$$

Definition A.5 (Körper)**.** Ein **Körper** $(K, +, \cdot)$ ist eine Menge K mit zwei inneren Verknüpfungen, geschrieben als Addition $+$ und Multiplikation $\cdot$, die die folgenden Bedingungen erfüllt:

- Abgeschlossenheit bzgl. der Addition: $\forall x, y \in K : x + y \in K$
- Assoziativgesetz der Addition: $\forall x, y, z \in K : (x + y) + z = x + (y + z)$
- Neutrales Element der Addition: $\exists\ 0 \in G : \forall x \in G : 0 + x = x = x + 0$
- Inverses Element der Addition: $\forall x \in K : \exists (-x) \in K : x + (-x) = 0 = (-x) + x$
- Kommutativgesetz der Addition: $\forall x, y \in K : x + y = y + x$
- Abgeschlossenheit bzgl. der Multiplikation: $\forall x, y \in K : x \cdot y \in K$
- Assoziativgesetz der Multiplikation: $\forall x, y, z \in K : (x \cdot y) \cdot z = x \cdot (y \cdot z)$
- Neutrales Element der Multiplikation: $\exists\ 1 \in K - \{0\} : \forall x \in K : 1 \cdot x = x = x \cdot 1$
- Inverses Element der Multiplikation: $\forall x \in K : \exists x^{-1} \in K : x \cdot x^{-1} = 1 = x^{-1} \cdot x$
- Kommutativgesetz der Multiplikation: $\forall x, y \in K : x \cdot y = y \cdot x$
- Distributivgesetze: $\forall x, y, z \in K : x \cdot (y + z) = x \cdot y + x \cdot z$ und $(x + y) \cdot z = x \cdot z + y \cdot z$
- Es gilt: $1 \neq 0$.

Oder kurz:

- K ist eine abelsche Gruppe bzgl. der Addition $+$
- $K - \{0\}$ ist eine abelsche Gruppe bzgl. der Multiplikation $\cdot$
- Es gelten die obigen Distributivgesetze.

□

Beispiel A.5. *Die rationalen Zahlen $\mathbb{Q}$, die reellen Zahlen $\mathbb{R}$ sowie die komplexen Zahlen $\mathbb{C}$ bilden jeweils mit der üblichen Addition $+$ und Multiplikation $\cdot$ einen Körper.*

Die folgenden Definitionen basieren auf dem Lehrbuch *Analysis* von W. Rudin, siehe [Rud09].

Definition A.6 (Träger)**.** Der **Träger** einer reellen Funktion f auf $\mathbb{R}^k$ ist die abgeschlossene Hülle der Nichtnullstellenmenge von f, formal:

$$\operatorname{supp}(f) = \overline{\{x \in \mathbb{R}^k : f(x) \neq 0\}}.$$

□

Definition A.7 (Einschränkung)**.** Gegeben seien eine Teilmenge $E \subset X$ des Definitionsbereichs und eine Funktion $f : X \to Y$. Dann heißt die Funktion $g := f\,|_E \colon E \to Y$ Einschränkung von f auf E, wenn

$$\forall x \in E : g(x) = f(x).$$

□

B. Der Beispieldatensatz

Es folgen die vollständigen Tabellen des im ersten Kapitel eingeführten Beispieldatensatzes. Es handelt sich hierbei um einen fiktiven Datensatz über das Vorlesungsangebot des Instituts für Informatik der Universität Rostock, bestehend aus einer STUDENTEN-Relation (Tabelle B.1), einer DOZENTEN-Relation (Tabelle B.3) und einer NOTEN-Relation (Tabelle B.5), einer MODUL-Relation (Tabelle B.2) sowie TEILNEHMER-Relation (Tabelle B.4).

$\mathbf{ID_{Studenten}}$	**Matrikelnr**	**Name**	**Vorname**	**Studiengang**
S_1	1	Fieber	Fabian	Lehramt Informatik
S_2	2	Sonnenschein	Sarah	Mathematik
S_3	3	Müller	Max	Elektrotechnik
S_4	4	Müller	Mira	Informatik
S_5	5	Johansen	Johannes	Informatik
S_6	6	Miller	Mia	Informatik
S_7	7	Mustermann	Max	Elektrotechnik
S_8	8	Johannes	Paul	ITTI

Tabelle B.1. STUDENTEN-Relation

$\mathbf{ID_{Module}}$	**Modulnr**	**Titel**	**Vertiefung**
M_1	001	Datenbanken III	Informationssysteme
M_2	002	Mathematik für Informatiker 1	xxx[3]
M_3	003	Mustererkennung und Kontextanalyse	Smart Computing
M_4	004	Individuelles Wissensmanagement	Informationssysteme
M_5	005	Data Warehouses und Business Intelligence	Wirtschaftsinformatik
M_6	006	Kognitive Systeme	Smart Computing
M_7	007	Theorie relationaler Datenbanken	Informationssysteme
M_8	008	Systembiologie	Modelle und Algorithmen
M_9	009	NEidI — Neueste Entwicklungen in der Informatik	xxx

Tabelle B.2. MODUL-Relation

[3] Platzhalter für Module, die nicht in die hier nicht aufgeführten Vertiefungen passen

$\mathbf{ID_{Dozenten}}$	**Modulnr**	**Dozent**
$D_{1.1}$	001	Professor A
$D_{1.2}$	001	Dozent A
D_2	002	Professor B
D_3	003	Professor C
D_4	004	Professor D
D_5	005	Dozent B
D_6	006	Professor D
D_7	007	Professor A
D_8	008	Professor E
D_9	009	Professor A

Tabelle B.3. DOZENTEN-Relation

$\mathbf{ID_{Teilnehmer}}$	**Modulnr**	**Matrikelnr**
T_1	001	1
T_2	001	2
T_3	001	4
T_4	001	5
T_5	002	1
T_6	002	2
T_7	002	3
T_8	002	4
T_9	002	5
T_{10}	002	6
T_{11}	002	7
T_{12}	003	1
T_{13}	004	3
T_{14}	004	5
T_{15}	004	7
T_{16}	005	4
T_{17}	005	7
T_{18}	006	4
T_{19}	006	5
T_{20}	007	1
T_{21}	007	3
T_{22}	007	5
T_{23}	008	3
T_{24}	009	1
T_{25}	009	4
T_{26}	009	5

Tabelle B.4. TEILNEHMER-Relation

$\mathbf{ID_{Noten}}$	Modulnr	Matrikelnr	Semester	Note
N_1	001	1	SS 16	2.0
N_2	001	2	SS 16	1.7
N_3	001	4	SS 16	1.7
N_4	001	5	SS 16	3.0
N_5	002	1	WS 15/16	3.7
N_6	002	2	WS 14/15	1.3
N_7	002	3	WS 14/15	2.3
N_8	002	4	WS 15/16	1.0
N_9	002	5	WS 15/16	1.3
N_{10}	002	6	WS 15/16	2.0
N_{11}	002	7	WS 15/16	3.3
N_{12}	003	1	WS 16/17	1.0
N_{13}	004	3	WS 16/17	1.3
N_{14}	004	2	WS 16/17	3.0
N_{15}	005	4	SS 17	2.7
N_{16}	005	7	SS 17	1.7
N_{17}	006	4	SS 17	2.7
N_{18}	006	5	SS 17	4.0
N_{19}	007	1	SS 16	2.3
N_{20}	007	3	SS 16	1.7
N_{21}	009	1	SS 16	3.3
N_{22}	009	5	SS 15	5.0
N_{23}	009	5	SS 16	2.7

Tabelle B.5. NOTEN-Relation

C. Quelltext des erweiterten Programms ProSA

Anfrage C.1 Provenance-Berechnung

```
public void provenance() {
        try {
                // get query
                String string = this.query;

                // Tokenize
                Tokenizer tokenizer = new Tokenizer();
                tokenizer.tokenize(string);

                // Parse tree
                SFWNode queryTree;
                Parser parser = new Parser();
                queryTree = parser.parse(tokenizer.getTokens());

                // rewrite query tree
                Rewriter rewriter = new Rewriter();
                ArrayList<Identifier> tupleIDs = rewriter.rewrite(queryTree, dbschema);
                String query = rewriter.toQueryString(queryTree);

                // determine polynome pattern
                int numberOfIdAttributes = tupleIDs.size();
                boolean aggregationPolynom = rewriter.isAggregationQuery();
                int aggregate = Aggregate.NONE;

                // execute rewritten query
                Statement stmt = null;
                stmt = c.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
                    ResultSet.CONCUR_UPDATABLE);
                ResultSet rs = stmt.executeQuery(query);
                ResultSetMetaData rsmd = rs.getMetaData();

                //save results
                ArrayList<ArrayList<String>> rows = new ArrayList<ArrayList<String>>();

                ////
                //anlegen einer ArrayListe
                ArrayList<String> witness = new ArrayList<String>();
                ArrayList<String> witnesslist = new ArrayList<String>();
                ////

                int columnsNumber = rsmd.getColumnCount(); // all columns
                int i = columnsNumber - numberOfIdAttributes + 1;

                // save regular column headers
                ArrayList<String> header = new ArrayList<String>();
                for (i = 1; i <= columnsNumber - numberOfIdAttributes; i++)
                        header.add(rsmd.getColumnName(i));
```

```
                // save provenance header
                header.add("How-Provenance");
                header.add("Why-Provenance");
                header.add("Where-Provenance");

                //add header to table
                rows.add(header);

                while (rs.next()) {
                        ArrayList<String> row = new ArrayList<String>();

                        // save row
                        for (i = 1; i <= columnsNumber - numberOfIdAttributes; i
                            ++)
                                row.add(rs.getString(i));

                        //add provenance information
                        String polynome = "";
                        String tuple = "(";
                        String whereProv = ""; // collect table names
                        int j, z;
                        for (j = i, z = 1; j <= columnsNumber; j++, z++) {
                                if (z > 1) {
                                        polynome += "␣*␣";
                                        tuple += ",";
                                        whereProv += ",";
                                }
                                polynome += rs.getString(j);
                                tuple += rs.getString(j);
                                whereProv += dbschema.getTableName(rsmd.
                                    getColumnName(j));

                                ////
                                //witness
                                witness.add(rs.getString(j));
                                ////

                        }
                        tuple += ")";
                        row.add(polynome);
                        row.add(tuple);
                        row.add(whereProv);
                        rows.add(row);
                }

                ////
                //sort list
                Collections.sort(witness);

                //delete duplicates
                witnesslist.add(witness.get(0));
                int count = 1;
                for (i = 1; i < witness.size(); i++) {
                        if (!witness.get(i).equals(witness.get(i-1))) {
                                witnesslist.add(witness.get(i));
                                count++;
                        }
                }
                ////
}
```

```
//print result
        if(this.outputFormat.equals("csv")) {
        //print how, why, where to different files
        if(this.multipleTables.equals("true")) {
                printToCSV(tokenizer, aggregate, rs, rsmd,
                    numberOfIdAttributes, rows, "how");
                printToCSV(tokenizer, aggregate, rs, rsmd,
                    numberOfIdAttributes, rows, "why");
                printToCSV(tokenizer, aggregate, rs, rsmd,
                    numberOfIdAttributes, rows, "where");
        }
        else {
                printToCSV(aggregate, rs, rsmd, numberOfIdAttributes,
                    rows);
        }

////
//type of CHASE-inverse
String inversType = FindInverseType(tokenizer);

//queries to reconstruct used tuples
String[] reconstructionArray = FindWitnessQuery(tokenizer, witnesslist);

//initialization
String[] witnessArray = new String[20];
int p = 0;
int index = 1;
int cJ = 0;

// counts natural joins
int countJoin = 0;
for(int i = 0; i<tokenizer.getTokens().size(); i++) {
        if(tokenizer.getTokens().get(i).sequence.equals("natural join"))
                countJoin++;
        else if(tokenizer.getTokens().get(i).sequence.equals("union"))
                countJoin++;
}

//reconstruction of used tuples
while (cJ <= countJoin) {
        for(int j=0; j < witnesslist.size(); j++) {
                // Tokenize
                Tokenizer tokenizer2 = new Tokenizer();
                tokenizer2.tokenize(reconstructionArray[j + cJ *
                    witnesslist.size()]);

                //save results
                StringBuilder sb = new StringBuilder();

                //add name of relation
                if(j == 0) {
                        for(int i = 0; i<tokenizer2.getTokens().size(); i
                            ++) {
                                if(tokenizer2.getTokens().get(i).sequence
                                    .equals("from")) {
                                        sb.append(tokenizer2.getTokens().
                                            get(i+1).sequence);
                                        sb.append("-Relation:");
                                        witnessArray[p] = sb.toString();
                                        sb.setLength(0);
                                }
```

```
                                        }
                                }

                                // Parse tree
                                SFWNode queryTree2 = null;
                                Parser parser2 = new Parser();
                                queryTree2 = parser2.parse(tokenizer2.getTokens());

                                // rewrite query tree
                                String query2 = rewriter.toQueryString(queryTree2);
                                ResultSet rs2 = stmt.executeQuery(query2);
                                ResultSetMetaData rsmd2 = rs2.getMetaData();

                                // save regular column headers
                                int columnsNumber = rsmd2.getColumnCount() ;
                                ArrayList<String> header = new ArrayList<String>();
                                for (int i = 1; i <= columnsNumber; i++) {
                                        sb.append(rsmd2.getColumnName(i));
                                        if(i!=columnsNumber)
                                                sb.append("; ");
                                }
                                witnessArray[p+1] = sb.toString();

                                // add relation tuples
                                sb.setLength(0);
                                while (rs2.next()) {
                                        for (int i = 1; i <= columnsNumber; i++) {
                                                sb.append(rs2.getString(i));
                                                if(i!=columnsNumber)
                                                        sb.append("; ");
                                        }
                                        witnessArray[index+p+1] = sb.toString();
                                        index++;
                                }
                                if(j == (witnesslist.size() - 1) && (cJ != countJoin)) {
                                        sb.setLength(0);
                                        sb.append("
                                            ----------------------------------------");
                                        witnessArray[index+p+1] = sb.toString();
                                        index++;
                                }
                        }
                        p = p + index + 1;
                        index = 1;
                        cJ ++;
                }

                // print to txt
                PrintToTXT(inversType, witness, witnesslist, witnessArray);
                ////

                rs.close();
                stmt.close();
                c.close();
        }
        catch (Exception e) {
                e.printStackTrace();
                System.err.println(e.getClass().getName() + ": " + e.getMessage());
                System.exit(0);
        }
}
```

Anfrage C.2 Typbestimmung der CHASE-Inversen

```
private String FindInverseType(Tokenizer tokenizer) throws SQLException {
        try {
                boolean selection = false;
                boolean projection = false;
                boolean aggregation = false;
                boolean join = false;
                boolean extremum = false;

                StringBuilder sb = new StringBuilder();

                sb.append("Inversen-Typ:");
                sb.append("\n");

                for(int p = 0; p < tokenizer.getTokens().size();p++) {
                        if(tokenizer.getTokens().get(p).sequence.equals("where") ||
                            tokenizer.getTokens().get(p).sequence.equals("WHERE") ||
                            tokenizer.getTokens().get(p).sequence.equals("Where"))
                                selection = true;
                        else if(!tokenizer.getTokens().get(p).sequence.equals("*"))
                                projection = true;
                        else if(tokenizer.getTokens().get(p).sequence.equals("max") ||
                            tokenizer.getTokens().get(p).sequence.equals("MAX") ||
                            tokenizer.getTokens().get(p).sequence.equals("Max"))
                                extremum = true;
                        else if(tokenizer.getTokens().get(p).sequence.equals("min") ||
                            tokenizer.getTokens().get(p).sequence.equals("MIN") ||
                            tokenizer.getTokens().get(p).sequence.equals("Min"))
                                extremum = true;
                        else if(tokenizer.getTokens().get(p).sequence.equals("count") ||
                            tokenizer.getTokens().get(p).sequence.equals("COUNT") ||
                            tokenizer.getTokens().get(p).sequence.equals("Count"))
                                aggregation = true;
                        else if(tokenizer.getTokens().get(p).sequence.equals("sum") ||
                            tokenizer.getTokens().get(p).sequence.equals("SUM") ||
                            tokenizer.getTokens().get(p).sequence.equals("Sum"))
                                aggregation = true;
                        else if(tokenizer.getTokens().get(p).sequence.equals("avg") ||
                            tokenizer.getTokens().get(p).sequence.equals("AVG") ||
                            tokenizer.getTokens().get(p).sequence.equals("Avg"))
                                aggregation = true;
                        else if(tokenizer.getTokens().get(p).sequence.equals("natural␣
                            join") || tokenizer.getTokens().get(p).sequence.equals("
                            NATURAL␣JOIN") || tokenizer.getTokens().get(p).sequence.
                            equals("Natural␣Join"))
                                join = true;
                }

                if(selection == true || extremum == true)
                        sb.append("Ergebnisäquivalente␣CHASE-Inverse");
                else if(aggregation == true || projection == true)
                        sb.append("Relaxte␣CHASE-Inverse");
                else if(join == true)
                        sb.append("Exakte␣CHASE-Inverse␣oder␣Ergebnisäquivalente␣CHASE-
                            Inverse");
                else
                        sb.append("Exakte␣CHASE-Inverse");

                String inversType = sb.toString();
                return inversType;
        }
```

```
        catch (Exception e) {
                e.printStackTrace();
                System.err.println(e.getClass().getName() + ": " + e.getMessage());
                System.exit(0);
                return null;
        }
}
```

Anfrage C.3 Aufstellen der Anfragen zur Rekonstruktion der Quelldatenbank

```
private String[] FindWitnessQuery(Tokenizer tokenizer, ArrayList<String> witnesslist)
    throws SQLException {
        try {
                StringBuilder sb = new StringBuilder();
                String[] reconstructionArray = new String[100];
                int index = 0;
                int cJ = 0;

                // counts natural joins
                int countJoin = 0;
                for(int i = 0; i<tokenizer.getTokens().size(); i++) {
                        if(tokenizer.getTokens().get(i).sequence.equals("natural join"))
                                countJoin++;
                }

                // compile queries
                while(cJ < countJoin + 1) {
                        for(int i = 0; i < witnesslist.size(); i++) {
                                sb.setLength(0);
                                sb.append("select ");
                                int p = 1;
                                while (!tokenizer.getTokens().get(p).sequence.equals("
                                    from"))
                                        p++;
                                sb.append("*");
                                sb.append(" from ");
                                p++;
                                if(p<tokenizer.getTokens().size())
                                        sb.append(tokenizer.getTokens().get(p+2*cJ).
                                            sequence);
                                sb.append(" ");
                                sb.append("where id_");
                                sb.append(tokenizer.getTokens().get(p+2*cJ).sequence);
                                sb.append(" = '");
                                sb.append(witnesslist.get(i));
                                sb.append("';");
                                //Array for reconstruction
                                reconstructionArray[index] = sb.toString();
                                index ++;
                        }
                        cJ ++;
                }
                return reconstructionArray;
        }
        catch (Exception e) {
                e.printStackTrace();
                System.err.println(e.getClass().getName() + ": " + e.getMessage());
                System.exit(0);
                return null;
        }
}
```

D. Anhang: Aufbau des Datenträgers

Auf dem beiliegenden Datenträger befinden sich ergänzende Materialien. Die Struktur und der Inhalt werden im folgenden kurz erläutert. Die Überschriften entsprechen den Ordnern im Wurzelverzeichnis des Datenträgers.

Software

- Java-Programm **ProSA** von P. Wilsdorf, S. Brossmann und T. Auge

Literaturquellen

Die im Literaturverzeichnis aufgelisteten Quellen werden hier aufgelistet. Eine Ausnahme bilden die digital nicht verfügbarer Werke: [Bro82, KSS14, RN04, Rud09, SSH13]. Es wurden ebenso alle Webseiten, welche innerhalb der Arbeit zitiert wurden, als Portable Dokument Format (*.pdf) exportiert und bereitgestellt: [Aug17, CER17, Dij13, Hem17, Leh17, OVH17, Sal16]. Der Dateiname der Quellen entspricht dem Kürzel im Literaturverzeichnis. Hinweis: Die beigefügte Literatur dient der Nachvollziehbarkeit von Aussagen und darf nicht öffentlich bereitgestellt werden. Dies gilt speziell für Werke, welche durch Lizenzverträge der Universität Rostock durch die Universitätsbibliothek zur Verfügung gestellt wurden.

Masterarbeit

- Tex-Dateien und Abbildungen der Arbeit (Masterarbeit_Auge.zip)
- Digitale Fassung der Arbeit (Masterarbeit_Auge.pdf)
- Programm **ProSA** inklusive Erweiterungen (in der Form //// < Erweiterung > ////)

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Masterarbeit selbstständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommen Stellen sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Rostock, den 5.9.2017 _____