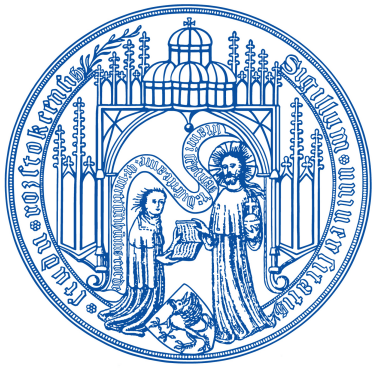
Parallele Anonymisierung von großen Datenbeständen

Bachelorarbeit

Universität Rostock
Fakultät für Informatik und Elektrotechnik
Institut für Informatik



vorgelegt von: Eric Klein
Matrikelnummer: 215204156
geboren am: 08.02.1997 in Greifswald
Erstgutachter: Prof. Dr. rer. nat. habil. Andreas Heuer
Zweitgutachter: Dr.-Ing. Holger Meyer
Betreuer: M.Sc. Hannes Grunert
Abgabedatum: 20.03.2020

Inhaltsverzeichnis

1 Einleitung **3**

2 Datenschutz **5**
2.1 Gesetzliche Grundlagen . . . 5
2.2 Anonymisierungstechniken . . . 7
2.3 Generalisierung . . . 7
2.3.1 k-Anonymität . . . 8
2.3.2 l-Diversität . . . 9
2.3.3 t-Closeness . . . 10
2.4 Unterdrückung . . . 13
2.5 Slicing . . . 13
2.6 Differential Privacy . . . 14
2.6.1 Randomised-Response-Technik . . . 15
2.6.2 ε-Differential-Privacy . . . 16
2.6.3 (ε,δ)-Differential-Privacy . . . 16

3 Eigene Operatoren **17**
3.1 Bekannte Optimierungsregeln . . . 17
3.2 Fortlaufendes Beispiel . . . 19
3.3 Äquivalenzarten . . . 19

4 Generalisierungs-Operator α^G **23**
4.1 Aufbau . . . 23
4.2 Annahmen und Einschränkungen . . . 23
4.3 Beispiel . . . 24
4.4 Regeln . . . 25

5 Slicing-Operator α^S **36**
5.1 Aufbau . . . 36
5.2 Annahmen und Einschränkungen . . . 36
5.3 Ausführliche Formalisierung . . . 37
5.4 Beispiel . . . 38
5.5 Regeln . . . 39

6 Differential-Privacy-Operator α^D **48**
6.1 Aufbau . . . 48
6.2 Annahmen und Einschränkungen . . . 48
6.3 Ausführliche Formalisierung . . . 49
6.4 Beispiel . . . 51
6.5 Regeln . . . 52

7 Auswertung **60**
7.1 Vertauschbarkeit . . . 60
7.2 Bewertung verteilbarer Techniken . . . 61

8 Praxisteil **65**
8.1 Umsetzungsspektrum 65
8.2 Abstrakte Umsetzung 65
8.3 Rewriter 71
8.4 Konkret implementiertes Beispiel 74

9 Schlusswort **80**

Literaturverzeichnis **82**

A Anhang: Generalisierungsebenen **84**

B Anhang: Datenträger **87**

1 Einleitung

In Zeiten von Digitalisierung und Vernetzung sind alle Daten, die ein Einzelner preisgibt, übermittelt oder unbewusst hinterlässt, zu einem überaus nutzbringenden, profitablen Gut geworden. Es mag sich mitunter nicht so anfühlen, da vieles, das man tut oder äußert, nebensächlich erscheinen mag, doch selbst die unbedeutendsten Dinge können für andere zweckhaft sein – für Forschung, für öffentliche Statistik oder auch für zielgerichteten Missbrauch. Nur für sich genommen mögen erhobene Daten abgeschlossenen Nutzen besitzen, doch im Verbund betrachtet, jenseits eines vertraulichen und rechtmäßigen Gebrauchs, bei potenziellem Bezug zu einer echten Person, sind sie noch weitaus mächtiger. Bei manchen Daten mag es auf der Hand liegen, warum sie schützenswert sind – doch da die Trends der Zukunft wahrlich nichts abzusehen sind, könnten selbst die scheinbar unwichtigsten Daten noch wertvoll und gefährlich werden.

Es ist schon ein Dilemma: Einerseits möchte man im Sinne des Fortschritts neue Erkenntnisse vorantreiben, andererseits sollten personenbezogene Daten niemals gedankenlos verarbeitet werden und von einer Institution zu einer anderen wandern – zumal die entsprechenden Personen über Rechte verfügen und Datenmissbrauch einzig eine Frage der Zeit wäre. Aus ebendiesen besagten Gründen existieren Anonymisierungsverfahren. Sie sollen es ermöglichen, dass auch Dritte mit derartigen Daten arbeiten können, sobald ein vertraulicher Gebrauch nicht mehr garantiert werden kann. Um die Rechte des Einzelnen zu schützen, ist es daher konsequent, dass die Daten dann soweit entfremdet werden, dass eine Identifizierung einer Person ausbleibt.

Der Wunsch nach Informationsschutz ließe sich gedanklich sogar so weit treiben, dass man selbst bei einem Anonymisierungsprozess nicht bereit wäre, getrennt gelagerte Teilrelationen so einfach preiszugeben. Es existieren deswegen Algorithmen, die eine gesamtheitliche Anonymisierung von Partitionen bewältigen können, ohne sie zu einer entanonymisierten Gesamtrelation zusammenführen zu müssen. Doch nicht jedem ist ein solcher Aufwand hinsichtlich dessen Kosten und Grenzen wert. Die Frage, mit der sich diese Arbeit im Folgenden befassen wird, lautet daher: Wie stark lassen sich geschlossene Anonymisierungen ausschöpfen und wie stände es um deren Optimierbarkeit? Um dies tun zu können, muss es möglich sein, bekannte Anonymisierungsverfahren wie Operatoren der relationalen Algebra behandeln zu können. Hierzu werden sie auf wesentliche Aspekte reduziert, um dann ausgiebig deren Beziehung zu anderen Operatoren zu ergründen – sowohl zu unären als auch binären.

Zu Beginn dieser Arbeit gibt es in Kapitel 2.1 einen kurzen Einblick in die rechtliche Lage zum Thema Datenschutz. Anschließend werden in Kapitel 2.2 drei der gängigsten Anonymisierungsverfahren vorgestellt: Generalisierung, Slicing und Differential Privacy. Jeder dieser Techniken wird dann in gesonderten Kapiteln als formaler Operator der relationalen Algebra eingeführt – natürlich in deutlich reduzierter und eingeschränkter Form, da es innerhalb dessen viel zu viele Spezialtechniken gäbe. Kapitel 3 gibt hierzu kurzen Überblick und führt zudem einige spezielle Konventionen ein. Kapitel 4 umfasst den Generalisierungs-Operator, Kapitel 5 den Slicing-Operator und Kapitel 6 den Differential-Privacy-Operator. Nach der Erklärung ihrer jeweiligen Wirkungsweise wird ausgiebig deren Anwendung ergründet. Es wurde nach prinzipieller Redundanz geschaut, nach Vereinfachung bei Mehrfachanwendung und selbstverständlich die bedingte Vertauschbarkeit mit etlichen bekannten Operatoren

durchexerziert. Neben streng äquivalenten Regeln sind noch weitere gefunden worden, die einer zweiten, etwas schwächeren Vertauschungsklasse angehören. Diese andere Klasse toleriert gewisse Werteabweichungen, sofern es sich nicht auf das Mindestmaß an Anonymität auswirkt. Kapitel 7 liefert eine Auswertung jener Forschungsergebnisse aus Kapitel 4 bis 6. Kapitel 8 stellt eine praktische Umgebung zum Vertauschen und Testen von Anonymisierungsregeln vor. Aufgrund der Vielzahl an gefundenen Regeln und dem generellen Ausbleiben weiterer Erkenntnisse wurde der ANteil praktisch umgesetzter Regeln im Übrigen moderat gehalten. Es wurde sich im Folgenden auf einen der Operatoren und dann wiederum auf einige ausgewählte Regeln konzentriert. Kapitel 9 umfasst ein kurzes Schlusswort. Es stellt noch einmal abschließend ein paar allgemeine Erkenntnisse heraus und gibt einen Ausblick auf einige Dinge, die man in Zukunft untersuchen könnte. Danach schließt direkt das Literaturverzeichnis an. Darüber hinaus existieren noch zwei Anhänge. Anhang A beschäftigt sich mit einer zusätzlichen Untersuchung, die aus Kapitel 4 ausgelagert wurde, da sie an jener Stelle zu ausschweifend gewesen wäre. Anhang B dient abschließend dazu, die Verzeichnisstruktur eines zusätzlich mit abgegebenen Datenträgers zu erklären. Auf besagtem Gegenstand befinden sich eine digitale Fassung dieser Arbeit, alle Literaturquellen sowie die praktisch umgesetzte Implementierung.

2 Datenschutz

Zu Beginn dieses Kapitels wird kurz die allgemeine gesetzliche Lage zum Thema Datenschutz dargelegt. Danach werden bekannte Techniken vorgestellt, die dazu dienen, um ebendiesen gewährleisten zu können.

2.1 Gesetzliche Grundlagen

Die Frage nach der Weitergabe von Daten mit Personenbezug folgt klaren gesetzlichen Vorgaben. Doch was gilt eigentlich als personenbezogen und wann ist eine Weitergabe rechtlich erlaubt? Diese Sachverhalte sind zum einen im Bundesdatenschutzgesetz (BDSG) [BDS10, BDS19] und zum anderen in der Datenschutzgrundverordnung (DSGVO) [DSG16] festgelegt. Das DSGVO ist eine EU-weite Verordnung, das BDSG eine deutschlandweite.

Bis vor einigen Jahren gab es noch keine länderübergreifenden Datenschutzgesetze. Sie wurden bloß auf nationaler Ebene vorgeschrieben – in Deutschland vom seit den Siebzigern existierenden Bundesdatenschutzgesetz. Doch da es einen Wunsch nach Vereinheitlichung gab, ist die EU-weite Datenschutzgrundverordnung auf den Weg gebracht worden. Sie trat am 24. Mai 2016 in Kraft und hat nach einer Übergangsphase von zwei Jahren das alte BDSG endgültig abgelöst. Im Zuge der Ablösung ist zudem ein neues BDSG an dessen Stelle getreten, das zusätzlich zum DSGVO bewusst offengelassene Stellen konkretisiert. In Deutschland ist also die DSGVO und das aktuelle BDSG als Ganzes zu betrachten. Fragt sich natürlich, was nun das Hauptanliegen dieser Gesetze sein soll.

DSGVO, §1 – Gegenstand und Ziele, Absatz 1:

> „Diese Verordnung enthält Vorschriften zum Schutz natürlicher Personen bei der Verarbeitung personenbezogener Daten und zum freien Verkehr solcher Daten“ [DSG16].

Im Kern deckt sich dies genau mit dem, was in der Einleitung schon zur Sprache gebracht wurde. Kommen wir daher zu den Begrifflichkeiten im Einzelnen. Es ist schon mehrfach der Begriff „personenbezogene Daten“ gefallen. Was ist darunter denn überhaupt zu verstehen?

Neues BDSG, §46 – Begriffsbestimmungen, Begriff 1:

> „[Der Begriff] ‚personenbezogene Daten‘ [bezeichnet] alle Informationen, die sich auf eine identifizierte oder identifizierbare natürliche Person (betroffene Person) beziehen; als identifizierbar wird eine natürliche Person angesehen, die direkt oder indirekt, insbesondere mittels Zuordnung zu einer Kennung wie einem Namen, zu einer Kennnummer, zu Standortdaten, zu einer Online-Kennung oder zu einem oder mehreren besonderen Merkmalen, die Ausdruck der physischen, physiologischen, genetischen, psychischen, wirtschaftlichen, kulturellen oder sozialen Identität dieser Person sind, identifiziert werden kann;“ [BDS19].

Dinge wie eine abstrakte, aber eindeutige ID oder außergewöhnliche Namen liegen allgemein auf der Hand. Weitaus entscheidender ist hier vor allem das „indirekt“. In Kombination können für sich gesehen wenig aussagende Merkmale, die man etlichen Personen zuschreiben könnte, einen urplötzlich identifizierbar machen. Aus diesem Grund sollten nicht nur die offensichtlichen Eigenschaften als gefährlich betrachtet werden, sondern zuerst einmal alle.

DSGVO, §5 – Grundsätze für die Verarbeitung personenbezogener Daten, Absatz 1b:

> „[Personenbezogene Daten müssen] für festgelegte, eindeutige und legitime Zwecke erhoben werden und dürfen nicht in einer mit diesen Zwecken nicht zu vereinbarenden Weise weiterverarbeitet werden; eine Weiterverarbeitung für im öffentlichen Interesse liegende Archivzwecke, für wissenschaftliche oder historische Forschungszwecke oder für statistische Zwecke gilt gemäß Artikel 89 Absatz 1 nicht als unvereinbar mit den ursprünglichen Zwecken (‚Zweckbindung');" [DSG16].

Auch wenn der prinzipielle Nutzen von Daten dessen Zwecke überschreiten kann, ist der legitime Besitz noch lange keine Berechtigung dafür. Nichtsdestotrotz hält der dort genannte Artikel 89 für die gewisse Zwecke eine Möglichkeit offen, sofern nämlich „die Identifizierung von betroffenen Personen nicht oder nicht mehr möglich ist" [DSG16].

DSGVO, §5 – Grundsätze für die Verarbeitung personenbezogener Daten, Absatz 1c:

> „[Personenbezogene Daten müssen] dem Zweck angemessen und erheblich sowie auf das für die Zwecke der Verarbeitung notwendige Maß beschränkt sein (‚Datenminimierung');" [DSG16].

Artikel 5 macht klar, dass sowohl bei der Erhebung als auch beim Gebrauch niemals mehr ersichtlich sein soll als nötig. Die reine Tatsache, dass man Daten erheben darf, sollte nicht missbraucht werden, um insgeheim Daten fernab des Zwecks zu sammeln. Und nur, weil die Verarbeitung vertraulich geschieht, heißt dies nicht, dass man bis zuletzt Daten weiterverwenden sollte, die ihren Zweck schon erfüllt haben. Das eine ist grundsätzlich verwerflich, das andere zwar nichts direkt, doch würde missbrauchenden Absichten geradezu in die Hände spielen. Dieser Grundsatz galt auch schon im alten BDSG. Aufgrund der Wichtigkeit und besseren Formulierung sei nachfolgend auch dieses einmal zitiert.

Altes BDSG, §3a – Datenvermeidung und Datensparsamkeit:

> „Die Erhebung, Verarbeitung und Nutzung personenbezogener Daten und die Auswahl und Gestaltung von Datenverarbeitungssystemen sind an dem Ziel auszurichten, so wenig personenbezogene Daten wie möglich zu erheben, zu verarbeiten oder zu nutzen. Insbesondere sind personenbezogene Daten zu anonymisieren oder zu pseudonymisieren, soweit dies nach dem Verwendungszweck möglich ist und keinen im Verhältnis zu dem angestrebten Schutzzweck unverhältnismäßigen Aufwand erfordert" [BDS10].

Obwohl die alte Verordnung mittlerweile abgelöst sein mag, spiegelt er deutlich wider, dass sich im Kern des Ganzen nichts verändert hat. Wann die aktuelle Datenschutzverordnung ihre Wirksamkeit verliert, ist festgehalten in den Erwägungsgründen. Die Relevanz von Anonymisierungsverfahren dürfte mehr als deutlich sein.

DSGVO, Erwägungsgrund 26 – Keine Anwendung auf anonymisierte Daten:

> „Die Grundsätze des Datenschutzes sollten für alle Informationen gelten, die sich auf eine identifizierte oder identifizierbare natürliche Person beziehen. [...] Die Grundsätze des Datenschutzes sollten daher nicht für anonyme Informationen gelten, d.h. für Informationen, die sich nicht auf eine identifizierte oder identifizierbare natürliche Person beziehen, oder personenbezogene Daten, die in einer Weise anonymisiert worden

sind, dass die betroffene Person nicht oder nicht mehr identifiziert werden kann. Diese Verordnung betrifft somit nicht die Verarbeitung solcher anonymer Daten, auch für statistische oder für Forschungszwecke“ [DSG16].

2.2 Anonymisierungstechniken

Im Folgenden werden grundlegende Konzepte der Anonymisierung und einige wichtige Metriken erklärt. Doch bevor wir dazu kommen, bedarf es noch der Einführung einer Kategorisierung der Ausgangsdaten, also jenen, auf den wir ein Anonymisierungsverfahren anwenden wollen.

Diese lassen sich in Identifikatoren (IDs), Quasi-Identifikatoren (QIDs) und sensitive Daten (SD) unterteilen. Identifikatoren sind die aus relationalen Datenbanken bekannten Schlüsselattribute. Sie ermöglichen die eindeutige Tupelidentifizierung und dürfen auf keinen Fall offengelegt werden, da sich mittels Verknüpfung noch weitere Informationen gewinnen ließen. Sensitive Daten sind jene, die Informationen enthalten, die es zu schützen gilt. Quasi-Identifikatoren sind Attributkombinationen mit schlüsselähnlicher Wirkung. Dies meint, dass es nicht gleich auf alle Tupel zutreffen muss, sondern schon ein Großteil der Fälle genügt [Sam01]. Zur Veranschaulichung ein Beispiel, zu sehen in Tabelle 1, welche im Groben von Nielsen et al. entstammt [NJTF15].

IdNr	PLZ	Alter	Krankheit
1	47677	29	Herzerkrankung
2	47602	22	Herzerkrankung
3	47678	27	Herzerkrankung
4	47905	43	Grippe
5	47909	52	Herzerkrankung
6	47906	47	Krebs
7	47605	30	Herzerkrankung
8	47673	36	Krebs
9	47607	32	Krebs

Tabelle 1: Ursprüngliche Patientendaten

In diesem haben wir eine unanonymisierte Relation zu erkrankten Patienten. Als ID wurde eine künstliche Identifikationsnummer, genauer gesagt `IdNr`, vergeben. Unsere schützenswerten Attribute sind deren `Krankheit`. `Alter` und die Postleitzahl `PLZ` sind nicht-sensitiv und als QID-Menge zu betrachten. In diesem Beispiel sind sie nicht erst in Kombination, sondern schon einzeln gefährlich.

2.3 Generalisierung

Bei Generalisierung erfolgt eine Werteersetzung hin zu weniger präzisen Daten. Das Ganze geschieht aber nicht willkürlich, sondern wahrt eine semantische Konsistenz. Numerische Zahlen, Uhrzeit oder Tag lassen sich etwa als Intervall zusammenfassen. Worte mit geringer Zeichendistanz können verschiedenartige Teile durch ein universelles Token, was quasi ein zensierendes Symbol außerhalb des regulären Zeichensatzes ist, unterdrücken. Häufig verwendete Zeichen sind beispielsweise das X oder das Sternsymbol $*$. Allgemeiner gesagt werden zusammenfassende Oberbegriffe gefunden [Swe02].

Sweeney definiert Generalisierung als eine Funktion $f : A \to B$, die die Werte eines Attributs A begrenzt oft in neue überführen kann, also nach der Form:

$$A_0 \xrightarrow{f_0} A_1 \xrightarrow{f_1} ... \xrightarrow{f_{n-1}} A_n$$

Die Ausgangsmenge A_0 ist dabei die Wertedomäne einer noch unanonymisierte Attributspalte A. Des Weiteren gibt es stets ein maximales Element A_n, an welchem jedes weitere Generalisieren zum Erliegen kommt. Die gesamte Ordnungsrelation, die sich darauf bilden lässt, wird Generalisierungshierarchie DGH_A genannt [Swe02]. Statt *domain generalization hierarchy* (DGH) kann man ebenso den Begriff *value generalization hierarchy* (VGH) antreffen [NJTF15].

Ausgehend von unseren Patientendaten verdeutlichen Abbildung 1 und 2 mögliche Hierarchien für die QID-Attribute `PLZ` und `Alter`.

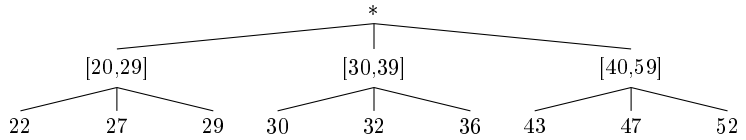


Abbildung 1: DGH_{Alter}

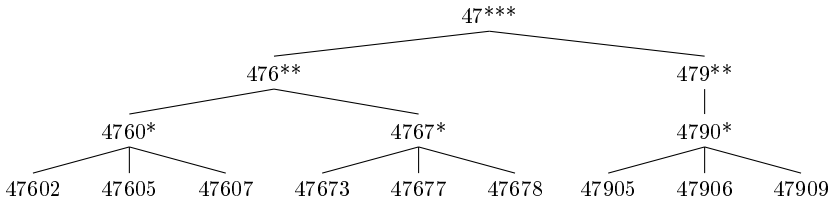


Abbildung 2: DGH_{PLZ}

2.3.1 k-Anonymität

Eine Relation ist k-anonym, falls jedes Tupel bezüglich seiner quasi-identifizierenden Attributmenge von mindestens $k-1$ anderen nicht mehr unterscheidbar ist. Da die wenigsten Datenbestände so etwas naturgegeben erfüllen, wendet man zur Erreichung dessen Generalisierung an [SS98]. Blöcke mit gleicher Belegung der QIDs bilden folglich Äquivalenzklassen (ÄK) aus, die mitunter auch als Q^*-Blöcke bezeichnet werden [MKGV07].

Im Konkreten lässt sich aus unseren Patientendaten (Tabelle 1) sowie den beiden Generalisierungshierarchien (Abbildung 1 und 2) eine generalisierte Tabelle erzeugen, die 3-Anonymität zusichert, sodass jeder Patient von mindestens zwei anderen nicht mehr zu unterscheiden ist. Doch dieser Ansatz hat jedoch seine Grenzen, da er sich nur um die Gruppenbildung selbst, nicht aber deren Mitglieder sorgt. Den Patienten mit den ID-Nummern 1 bis 3 nützt es zum

Beispiel herzlich wenig, wenn Angreifer mit QID-Kenntnis im Grunde doch um deren Herzkrankheit wüssten, da alle Werte in diesem Block ja die gleichen sind. Machanavajjhala et al. [MKGV07] nennen diesen Mangel an Diversität Homogenitätsattacke (*homogeneity attack*). Außerdem legten sie dar, dass ein Angriff über Hintergrundwissen (*background knowledge attack*) möglich wäre. Durch Zusatzwissen könne ein Angreifer Bestimmtes ausschließen oder aber für wahrscheinlich halten [MKGV07]. Aus diesem Grunde wurde die k-Anonymität erweitert.

	Nicht-sensitiv		Sensitiv	
IdNr	PLZ	Alter	Krankheit	
1	476**	[20,29]	Herzerkrankung	1-divers
2	476**	[20,29]	Herzerkrankung	
3	476**	[20,29]	Herzerkrankung	
4	479**	[40,59]	Grippe	3-divers
5	479**	[40,59]	Herzerkrankung	
6	479**	[40,59]	Krebs	
7	476**	[30,39]	Herzerkrankung	2-divers
8	476**	[30,39]	Krebs	
9	476**	[30,39]	Krebs	

Tabelle 2: 3-anonymisierte Fassung

2.3.2 l-Diversität

Allgemein versteht man unter l-Diversität, dass neben k-Anonymität zusätzlich gefordert wird, dass die sensitiven Daten einer jeden Äquivalenzklasse mindestens l verschiedene Werte aufweist [Hau07]. Unbeabsichtigte Eindeutigkeit wird dadurch unterbunden. Unsere soeben anonymisierte Tabelle mit 3-Anonymität (Tabelle 2) gewährt leider nur 1-Diversität, da der erste Q^*-Block wertetechnisch homogen ist.

Machanavajjhala et al. [MKGV07] haben eine noch deutlich differenziertere Unterteilung aufgestellt. Um nicht allzu sehr in die Tiefe zu gehen, belassen wir es bei einer Vorstellung der ersten drei Formen, wie es auch [LLV07] taten. Unsere eben genannte Definition wird hierbei *distinct l-diversity* genannt, da diese Eigenschaft noch einfach auszumachen ist. Ein übermäßig stark vertretener Attributwert kann dabei auf leichte Weise einen Angriff möglich machen.

Ein wenig komplizierter ist dagegen die Entropie-l-Diversität. Bei einer k-anonymen Tabelle hat für jeden q^*-Block zu gelten:

$$-\sum_{s \in S} p_{(q^*,s)} log(p_{(q^*,s')}) \geq log(l)$$

wobei S der Wertebereich des sensitiven Attributes ist und $p_{(q^*,s)}$ der Tupelanteil eines konkreten sensitiven Attributs s in einem q^*-Block. Hierbei wird ein Informationsmindestmaß an $log(l)$ gefordert. Zuweilen kann dies aber äußerst einschränkend sein, da ein überrepräsentierter Wert eine geringe Entropie zur Folge hat. Ein häufig auftretender Wert kann nämlich durchaus erwünscht und logisch sein. Ein gutes Beispiel sei eine medizinische Einrichtung mit Schwerpunkt auf Herzkrankheiten [MKGV07, LLV07].

Aus diesem Grunde gibt es die rekursive (c, l)-Diversität. Sie stellt sicher, dass die häufigsten Werte nicht zu oft und die weniger häufigen nicht zu selten auftreten. Definiert ist sie wie folgt: Sei m die Anzahl von Werten einer Äquivalenzklasse Q^*. Sei r_i dabei die Häufigkeit eines sensitiven Wertes in dieser, wobei $1 \leq i \leq m$ ist, r_1 zudem der häufigste Wert und jeder anschließende der nächsthäufige. Q^* besitzt rekursive (c, l)-Diversität, falls

$$r_1 < c \cdot (r_l + r_{l+1} + ... + r_m)$$

gilt. Und eine Tabelle erfüllt dies, sofern es für alle Äquivalenzklassen gilt. Das c ist im Übrigen eine selbstgewählte, positive Zahl. Sie sagt, wie tolerant die Diskrepanz zwischen häufigstem und seltensten Werten sein darf. Je größer das gewählte c, desto größer das erlaubte Missverhältnis [LLV07].

Nach [Hau07, LLV07] bietet l-Diversität typischerweise Spielraum für Angriffe zweierlei Arten. Die *similarity attack* ist dabei ein Angriff, der sehr einer Homogenitätsattacke gleicht. Zwar mögen konkrete Werte einer Äquivalenzklasse verschieden ausgeprägt sein, doch könnten sie zugunsten des Angreifers auf etwas inhaltlich Gemeinsames hinauslaufen. Bei einer *skewness attack* kommt ein schief verteilter Sachverhalt zum Tragen, bei dem der eher unwahrscheinliche Anteil eine Gefahr ausstrahlende Assoziation birgt. Wer sich aufgrund von Diversitätsforderung nun einen Q^*-Block mit einigen Gefahrenfallen teilen muss, die dort sicher nicht gerade unterrepräsentiert sind, wird aus Vorsicht ebenso als einer behandelt.

Das bestehendes Problem ist, dass der Diversitätsgedanke nur innerhalb der Gruppen selbst sichergestellt wird, jedoch nicht die semantische Nähe untereinander [LLV07].

2.3.3 t-Closeness

Dieser Eigenschaft liegt folgendes Prinzip zugrunde: Jede Äquivalenzklasse sowie die gesamte Tabelle selbst besitzt eine bestimmte Verteilung von sensitiven Attributen. Wenn die Verteilung einer Gruppe gegenüber der Gesamtverteilung höchstens eine Distanz von t aufweist, dann besitzt diese t-closeness. Trifft dies bei allen Äquivalenzklassen zu, so ist auch die gesamte Tabelle t-close [LLV07].

Kurz gesagt ist t also ein Maß, wie sehr sich alle Gruppen unterscheiden. Möchte man die Probleme der l-Diversität klein halten, sind ähnliche Verteilungen durchaus erstrebenswert. Je ähnlicher die Gruppen, desto geringer nämlich der Wissensgewinn für Angreifer [Hau07]. Ab einem gewissen Punkt führt es den Zweck von Äquivalenzklassen aber ad absurdum.

Kommen wir nun noch zum Problem der Distanzmessung. Natürlich gibt es etliche Methoden, weshalb wir es bei einer Vorstellung dreier belassen. Für alle werden zwei diskrete Verteilungen zugrunde gelegt: $P = (p_1, p_2, ..., p_m)$ und $Q = (q_1, q_2, ..., q_m)$. P soll im Folgenden die Gesamtverteilung des sensitiven Attributes `Krankheit` repräsentieren und Q dessen Verteilung innerhalb einer der Äquivalenzklassen. Jede deckt definitionsraumseitig die gleichen Attributwerte ab. Im Einzelnen kann eine bestimme Häufigkeit auch 0 sein. Ein gleicher Index repräsentiert auch untereinander gleiche Definitionswerte. Da wir ein einheitliches t wollen, sind unsere Häufigkeiten relativ bzw. die Werte normiert. In der Summe ist jede Verteilung daher 1. Alle Definitionen sind von der Notation her angelehnt an jene von [LLV07].

Variationsdistanz

Das einfachste der drei Verfahren ist die Variationsdistanz (VD). Die Nähe von Werten untereinander wird bei ihr außer Acht gelassen [LLV07]. Sie ist wie folgt definiert:

$$VD(P,Q) = \sum_{i=1}^{m} \frac{1}{2} \cdot |p_i - q_i|$$

Kullback-Leibler-Divergenz

Das zweite Verfahren ist die Kullback-Leibler-Divergenz (KLD) [KL51]. Auch hier wird semantische Nähe nicht berücksichtigt. Sie ist auch gemeinhin als relative Entropie bekannt, was der Differenz zwischen Entropie von P und Kreuzentropie von P und Q entspricht, also $KLD(P||Q) = H(P) - H(P,Q)$ [LLV07].

$$KLD(P||Q) = \sum_{i=1}^{m} p_i \cdot ln\frac{p_i}{q_i}$$

Da an Tabelle 2 schon beispielhaft dessen k-Anonymität und l-Diversität bestimmt wurde, soll auch die Errechnung von dessen t-closeness nicht fehlen. Zur Ermittlung wird hierzu eine normierte Variante der Kullback-Leibler-Divergenz (NKLD) benutzt. Die Einteilung und Berechnungsschritte sehen wie folgt aus:

	Krankheit	
P	Herzerkrankung	Q1
	Herzerkrankung	Q1
	Herzerkrankung	Q1
	Grippe	Q2
	Herzerkrankung	Q2
	Krebs	Q2
	Herzerkrankung	Q3
	Krebs	Q3
	Krebs	Q3

pi	P	Q1	Q2	Q3
Herzerkrankung	5/9	3/3	1/3	1/3
Grippe	1/9	0/3	1/3	0/3
Krebs	3/9	0/3	1/3	2/3

pi	P
Herzerkrankung	5/9
Grippe	1/9
Krebs	3/9

Normalisierungsfaktor:
$5/9 \cdot |\ln(5/9)| + 1/9 \cdot |\ln(1/9)| + 3/9 \cdot |\ln(3/9)|$
$= 0.936888$

pi	P	Q1
Herzerkrankung	5/9	3/3
Grippe	1/9	0/3
Krebs	3/9	0/3

$5/9 \cdot |\ln((5/9)/(3/3))|$
$= 0.326548$

$NKLD(P,Q_1) = 0.326548/0.936888 = \underline{0.348545}$

pi	P	Q2
Herzerkrankung	5/9	1/3
Grippe	1/9	1/3
Krebs	3/9	1/3

$5/9 \cdot |\ln((5/9)/(1/3))| + 1/9 \cdot |\ln((1/9)/(1/3))| +$
$3/9 \cdot |\ln((3/9)/(1/3))| = 0.405860$

$NKLD(P,Q_2) = 0.405860/0.936888 = \underline{0.433200}$

pi	P	Q3
Herzerkrankung	5/9	1/3
Grippe	1/9	0/3
Krebs	3/9	2/3

$5/9 \cdot \left|\ln((5/9)/(1/3))\right| + 3/9 \cdot \left|\ln((3/9)/(2/3))\right|$
$= 0.514841$

$\mathrm{NKLD}(P,Q_3) = 0.514841/0.936888 = \underline{0.549523}$

Für jede Äquivalenzklasse ist nun die Divergenz zur Gesamtverteilung berechnet worden. Der größte der errechneten Werte und damit die t-closeness lautet rund 0.549523. Dieser Wert ist die größtmögliche Distanz oder andersrum gesagt das allgemeine Minimum an Ähnlichkeit. Zur Einordnung des Wertes: wäre man auf 0-closeness aus, würde man totale Gleichheit fordern, und bei 1-closeness ließe man im Grunde absolute Toleranz zu. Im Sinne zusätzlicher Anonymität ist es nunmehr eine gewisse Gleichheit, die es anzustreben gilt. Der letztendliche Wert sagt aus, dass im Ernstfall mit einer relativ großen Verschiedenheit zu rechnen ist.

Earth-Mover's-Distanz

Das dritte Verfahren ist die Earth-Mover's-Distanz (EMD) von [RTG00], was nichts weiter als eine Abwandlung vom Monge-Kantorovich-Transportproblem ist [GS84]. Die nun erläuterte Definition ist nach [LLV07] und nicht etwa [RTG00], da letzterer noch die nötige Normierung für t-closeness fehlt. Zum besseren Verständnis sei noch angemerkt, dass dem Ganzen eine namensgebende Analogie zugrunde liegt, bei der man säulenartig gehäufte Erdmassen ohne größeren Aufwand wieder zurück in Erdlöcher verteilen möchte. Bei der EMD soll die minimale Arbeit ermittelt werden, die nötig ist, um eine Verteilung in eine andere zu überführen. Im Folgenden gibt d_{ij} die Bodenentfernung oder auch Grunddistanz (*ground distance*) zwischen p_i und q_j an. Weiterhin ist f_{ij} die minimale Masse, die aufzuwenden ist, um p_i in q_i zu verwandeln. Um die Distanz nun zu errechnen, gilt es ein Minimierungsproblem zu lösen, bei welchem die optimale Belegung von f_{ij} gefunden werden muss. Die zu minimierende Funktion ist dabei unsere Distanzgleichung

$$EMD(P,Q) = \sum_{i=1}^{m}\sum_{j=1}^{m} d_{ij} f_{ij}$$

mit den Nebenbedingungen:

$$(1) \quad f_{ij} \geq 0 \qquad \big| \; 1 \leq i \leq m, 1 \leq j \leq m$$

$$(2) \quad p_i - \sum_{j=1}^{m} f_{ij} + \sum_{j=1}^{m} f_{ji} = q_i \qquad \big| \; 1 \leq i \leq m$$

$$(3) \quad \sum_{i=1}^{m}\sum_{j=1}^{m} f_{ij} = \sum_{i=1}^{m} p_i = \sum_{i=1}^{m} q_i = 1$$

Bedingung (1) stellt Nicht-Negativität der Masse f_{ij} sicher. Bedingung (2) beschreibt den transformierenden Massefluss weg von p_i und hin zu q_i. Bedingung (3) gewährleistet, dass der Massefluss zwischen P und Q normalisiert ist, sodass auch die EMD einen normalisierten Wert zwischen 0 und 1 ergibt, ganz im Sinne von t-closeness.

2.4 Unterdrückung

Der Vollständigkeit halber sei noch die Technik der Unterdrückung erwähnt. Bei dieser werden bestimmte Einträge in Gänze ausgeblendet. Auf der einen Seite enthält man sensitive Daten zwar vor, auf der anderen müsste man unter Umständen weniger stark generalisieren, da diese Ausreißer nicht abgedeckt werden müssten. Vereinzelte Löschung könnte also feiner aufgeteilte QIDs ermöglichen und womöglich aussagekräftigere Daten bieten [SS98]. Bezogen auf das bloße Erreichen von k-Anonymität oder l-Diversität hilft einem diese Technik jedoch nicht weiter. Sie arbeitet ja prinzipiell dagegen. Für bessere t-closeness kann sie aber durchaus von Nutzen sein. Alternativ zur Unterdrückung wäre auch eine Generalisierung der sensitiven Attribute denkbar [LLV07]. Bezogen auf diese Arbeit hat diese Technik aber keine wirkliche Relevanz.

2.5 Slicing

Anstelle von Generalisierung setzt die Technik des Slicens auf Vertauschung. Die Grundidee des Ganzen stammt von [LLZM12]. Da mit Generalisierung eine Minderung des Informationsgehaltes einhergeht, wollte man eine alternative Anonymisierungstechnik anbieten, die höheren Datennutzen erlaubt. Dies kommt vornehmlich der Datenanalyse zugute, insbesondere der aggregierenden – den regressiven und korrelierenden Verfahren dagegen nicht. Beim Slicing wird ein Datenbestand n mal horizontal sowie m mal vertikal zerlegt. Die $n \cdot m$ Teilrelationen dieses Rasters werden dann wahlweise sortiert oder zufällig permutiert und anschließend wieder verknüpft. Folglich werden manche Attributverbindungen bewahrt, während andere aufgelöst werden. Zur Sicherung der Privatheit müssen korrelierende Attribute dabei unbedingt zusammenbleiben [LLZM12, GH15]. Im Folgenden wird das Slicing-Prinzip von Li und anderen nach dem Wortlaut von [GH15] erklärt.

Geschlecht	Alter	Krankheit	PLZ
männlich	29	Herzleiden	47677
männlich	22	Herzleiden	47602
weiblich	27	Herzleiden	47678
weiblich	43	Grippe	47905
weiblich	52	Herzleiden	47909
männlich	47	Krebs	47906
weiblich	30	Herzleiden	47605
weiblich	36	Krebs	47673
männlich	32	Krebs	47607

Tabelle 3: Ausgangsdaten

Der erste Schritt ist das horizontale Splitten (bei Li et al. *tuple partition* genannt). Hierbei findet eine Partitionierung in n disjunkte Teile mittels Selektionsbedingungen statt. Im Idealfall ließen sich die zugrunde liegenden Werte damit bestens aufteilen. Ansonsten muss ein zusätzliches Ranking-Attribut eingeführt werden. Man nummeriert also die gesamte Tupelmenge, sodass man sich dann beliebige Intervalle heraussuchen kann. Für die Tabelle 3 kam Nummerierung mit einem gruppierenden Fenster der Größe 3 zum Einsatz. Der Übersichtlichkeit halber lassen wir die Zwischenschritte mit dem temporären Attribut mal außen vor. Die farbliche Vorhebung dürfte schon genügen.

Der zweite Schritt ist das vertikale Splitten (*attribute partition*). Hierbei wird das Partitionieren mittels Projektion noch weitergetrieben. Alles, was horizontal unterteilt wurde, wird nun auch m-mal vertikal unterteilt. Dies muss aber nicht zwanghaft disjunkt sein, wie es noch bei [LLZM12] der Fall ist. Wichtig ist nur ein Zusammenbleiben stark korrelierender Attribute, da deren Teilung Rückschlüsse erlauben könnte. In unserem Beispiel behalten wir auf der einen Seite `Geschlecht` und `Alter` zusammen und `Krankheit` und `PLZ` auf der anderen.

Der nächste Schritt ist das Permutieren, der eigentliche Anonymisierungsschritt. Auf jeder der entstandenen Teilrelationen wird nun eine beliebig wählbare Sortierung angewandt. Die `Geschlecht`-`Alter`-Relationen unseres Beispiels wurden absteigend nach Alter sortiert, die `Krankheit`-`PLZ`-Relationen aufsteigend nach Postleitzahl. Das Ganze ist ersichtlich in Tabelle 4.

Der vierte und letzte Schritt ist das erneute Zusammenfügen. Alle Teilrelationen werden wieder zu einer ganzen gemacht. Sie ist ebenso groß wie unsere ursprüngliche Relation, nur dass einiges vertauscht wurde. Die intern benutzten Hilfsattribute werden selbstredend wieder ausgeblendet. Es folgen ein gesamtheitlicher Verbund, der die vertikale Teilung aufhebt, und eine Vereinigung, die Gleiches mit der horizontalen tut.

Geschlecht	Alter	Krankheit	PLZ
männlich	22	Herzleiden	47678
weiblich	27	Herzleiden	47677
männlich	29	Herzleiden	47602
weiblich	43	Herzleiden	47909
männlich	47	Krebs	47906
weiblich	52	Grippe	47905
weiblich	30	Krebs	47673
männlich	32	Krebs	47607
weiblich	36	Herzleiden	47605

Tabelle 4: Geslicte Daten

2.6 Differential Privacy

Bei Differential Privacy werden vereinzelte Daten durch zufälliges Rauschen verzerrt. Dies soll Identifizierbarkeit verhindern, doch ohne die Verteilung von Werten dabei gravierend zu verändern [DKM$^+$06].

Kernstück des Ganzen ist eine randomisierende Funktion: der Zufallsmechanismus κ [Dwo06]. Bei jeder Ausführung auf einen Datenbestand erfolgt eine zufällige Manipulation einiger Attributwerte. Anonymität wird ermöglicht, da man nicht mehr sagen kann, ob ein einzelner Eintrag noch wahrheitsgetreu oder aber verfälscht ist. Natürlich könnte eine willkürliche Veränderung von Daten diese zunehmend unbrauchbar machen. Damit sie in ihrer Gesamtheit dennoch Aussagekraft behalten, existieren formalen Modelle wie ε-Differential-Privacy (kurz ε-DP) oder auch die (ε,δ)-DP, durch welche man darauf achten kann. Da sie die budgetierte Obergrenze an erlaubter Anonymität angeben, spricht man diesbezüglich auch von *privacy budget* [FS10].

Im Folgenden werden diese zwei Metriken sowie die Grundidee von Datenverzerrung veranschaulicht. Erstere gehen zurück auf [Dwo06, Dwo08, DKM+06], wogegen das reine Konzept des Zufälligmachens deutlich weiter zurückreicht und schwer auszumachen ist.

2.6.1 Randomised-Response-Technik

Das Grundprinzip des Randomisierens kann unter anderem schon bei [War65] gefunden werden. Die dortige Methode veranschaulicht einfach, abstrakt und effektiv, wie man zufälliges Rauschen in seine Daten bringen kann. Auch anderswo ist dieses gängige Beispiel mit leichten Abwandlungen anzutreffen.

Im Grunde wird einer Probandengruppe eine simple Ja-Nein-Frage gestellt und deren Antwort obendrein von einem Münzwurf abhängig gemacht. Man kann sich an dieser Stelle eine beliebige platzhaltende Frage wie „Kennen Sie X?“ oder „Mögen Sie Y?“ zu einer Person, einem Film oder dergleichen ausdenken. Im Übrigen kann die Münze sowohl ideell als auch gezinkt sein. Dies obliegt ganz dem Experimentator. Bevor die Testperson eine Antwort verkündet, muss sie zuvor noch einige Male eine Münze werfen. Zum Fällen einer Entscheidung braucht sie höchstens zwei Würfe. Sollte das Werfen nicht geheim erfolgen, dann wirft sie sie auch noch die verbleibenden Male, sodass niemand imstande ist, Rückschlüsse zu ziehen.

Bei Kopf weiß sie bereits, dass sie wahrheitsgemäß antwortet. Bei Zahl soll stattdessen die Münze für sie antworten. Was sie sagt, hängt ganz vom zweiten Wurf ab. Bei Kopf beispielsweise „ja“, bei Zahl „nein“.

Natürlich ließe sich das Ganze auch komplexer gestalten. Anstelle des zweiten Münzwurfes (hier quasi eine Bernoulli-Verteilung) kann eine ausgedachte Antwort auch auf Basis einer beliebigen anderen Verteilung erzeugt werden. Der Ergebnisraum kann endlich, abzählbar unendlich oder ebenso gut stetig sein.

Geschlecht	Alter	Krankheit	PLZ
männlich	29	Herzleiden	47677
männlich	23	Herzleiden	47602
weiblich	27	Herzleiden	47678
weiblich	43	Krebs	47909
weiblich	52	Herzleiden	47909
männlich	47	Grippe	47906
weiblich	30	Herzleiden	47605
weiblich	36	Krebs	47673
männlich	32	Krebs	47607

Tabelle 5: Tabelle mit Rauschen

Zur weiteren Veranschaulichung wurde auch auf unsere Daten aus Tabelle 3 ein Rauscherzeugungsverfahren angewandt, ersichtlich in Tabelle 5. Wie dies abläuft, braucht uns nicht großartig zu interessieren. Veränderte Werte sind rot hervorgehoben. Welche Werte beim Verrauschen nämlich herauskommen können, mag zufällig sein, aber werteseitig nichtsdestotrotz vordefiniert. Es können komplett neue Werte auftreten, manche sich dagegen plötzlich doppeln. Obwohl der Zufall alles gestattet und seine Wichtigkeit hat, darf dieser nicht ausarten. Die Zahl an verhältnismäßigen Änderungen sollte also eine Grenze besitzen. Ebenso

sollte ein Kompromiss zwischen dem Auftreten gänzlich neuer Werte und bloßer Umverteilung (sprich Slicing) gefunden werden.

2.6.2 ε-Differential-Privacy

Die folgende Definition geht zurück auf die Arbeit von [Dwo06]. Seien D_1 und D_2 zwei Datenbestände, die sich höchstens um einen Eintrag unterscheiden. Sei $\mathcal{K} : X \to W$ ein Zufallsmechanismus, dessen Münzwurf auf einen Wahrscheinlichkeitsraum abgebildet wird. Weiterhin ist die Verteilung eines jeden Attributes gemeinhin bekannt, sodass Pr[E] die Eintrittswahrscheinlichkeit eines Ereignisses E angibt. Sei ε ein Maß für die Privatheit mit der Randbedingung $\varepsilon \geq 0$. Für alle D_1, D_2 und $S \subseteq W$ muss gelten:

$$\Pr[\mathcal{K}(D_1) \in S] \leq e^{\varepsilon} \cdot \Pr[\mathcal{K}(D_2) \in S]$$

Durch diese Vorschrift wird sichergestellt, dass die Ausblendung eines beliebigen Eintrages keinen signifikanten Unterschied für die Gesamtheit zur Folge hat. Da die An- oder Abwesenheit eines Einzelnen keine Rolle spielt, kann in jedem Fall für dessen individuellen Schutz gesorgt werden. Absoluten Schutz garantiert dies aber nicht [Dwo08].

Wie man sehen kann, wird eine zulässige Grenzabweichung um einen multiplikativen, also relativen Faktor e^{ε} gefordert, der natürlich maßgeblich vom gewählten ε abhängt. Da eine relative Toleranz aber durchaus restriktiv wirken kann, war es naheliegend zur Relativierung zusätzlich ein konstante, sprich absolute Toleranz einzuführen.

2.6.3 (ε,δ)-Differential-Privacy

Wie der Name vielleicht vermuten lässt, handelt es sich hierbei um eine Erweiterung von ε-Differential-Privacy. Um (ε,δ)-DP zu definieren, können wir auf so gut wie alles zuvor Erwähnte aufbauen. Nur bei der längeren Ungleichung am Ende bedarf es einer Neudefinition. Neu ist die zweite wählbare Maßzahl δ mit der Randbedingung $0 \leq \varepsilon \leq 1$. Im Grunde tut sie nichts weiteres, als unsere frühere Gleichung additiv zu ergänzen. Genauer gesagt gilt nun:

$$\Pr[\mathcal{K}(D_1) \in S] \leq e^{\varepsilon} \cdot \Pr[\mathcal{K}(D_2) \in S] + \delta$$

Das neu eingeführte δ erlaubt es, bei den vorangegangenen Anforderungen ein wenig toleranter zu agieren. Ist die Wahrscheinlichkeit eines Datensatzes grundsätzlich groß, würde das linear mitwachsende e^{ε} noch genug Spielraum zur Erfüllung der Ungleichung bieten, als dass der eine Datensatz großartig ins Gewicht fiele. Geringe Wahrscheinlichkeiten sind bei ε-DP daher kritische Probleme. Durch δ gibt es bei (ε, δ)-DP nun eine minimale Schranke, die eine bestimmte Grundtoleranz ermöglichen soll [DKM$^+$06].

3 Eigene Operatoren

In diesem kurzen Kapitel werden wichtige Grundlage für das Verständnis der Kapitel 4-6 gelegt. Die drei anschließenden Kapitel werden sich über weite Strecken in der Welt der relationalen Algebra bewegen.

Da die grundlegenden Anonymisierungstechniken nach deren Vorstellung in Kapitel 2.2 nun hinreichend bekannt sein sollten, werden ebendiese nun als relationale Operatoren eingeführt werden. Es wird bei jedem der Operatoren nach Optimierungsmöglichkeiten geschaut, also inwieweit im Zusammenspiel mit anderen Operatoren eine Vertauschung möglich ist, ohne dabei das Resultat und die geforderte Anonymität zu beeinflussen. Da im Rahmen dieser Arbeit leider nicht alle Metriken berücksichtigt werden können, wird der Fokus nur auf grundlegende gerichtet sein. Welche Techniken untersucht werden und welche nicht, ist in Tabelle 6 ersichtlich.

Techniken	Metrische Unterteilung	Eigener Operator
Generalisierung	k-Anonymität	ja
	l-Diversität	nein
	t-closeness	nein
Permutation	Slicing	ja
Differential Privacy	reine Randomisierung	ja
	ε-DP	nein

Tabelle 6: Übersicht zu den Operatoren

Zuletzt sei noch angemerkt, dass von den drei vorgestellten Techniken Slicing und Differential Privacy mit Zufall arbeiten. Wenn also ein Beispiel angeführt wird, dann es bei Gebrauch des Operators nur eine der möglichen Relationen, die ausgehend von der Ausgangsrelation resultieren kann. Eine äquivalente Umformungsregel meint in diesem Fall also, dass all die potenziell möglichen Ergebnisrelationen links- wie rechtsseitig mit gleicher Wahrscheinlichkeit eintreten sollten.

3.1 Bekannte Optimierungsregeln

Bezüglich der äquivalenten Vertauschbarkeit algebraischer Operatoren existieren etliche Regeln. Da im Rahmen dieser Arbeit allerhand neue aufgestellt wurden, seien an dieser Stelle noch einmal ein paar grundlegende erwähnt, zumal sich gewisse Muster durchaus darin wiederfinden lassen. Ein paar allgemeine Regeln zum Ändern der Operatorreihenfolge nach [HSS19] lauten:

1. ProjProj: Vereinfachung bei doppelter Projektion

   Beim π-Operator dominieren die äußeren Parameter die inneren, wobei im Sinne einer korrekten Anwendung natürlich $X \subseteq Y$ vorausgesetzt wird:

$$\pi_X(\pi_Y(r_1)) \iff \pi_X(r_1)$$

2. SelSel: Doppelte Selektion

   Bei σ entspricht eine Kombination von Prädikaten dem logischen Und, weshalb auch Vertauschung zulässig ist:

   $$\sigma_{F_1}(\sigma_{F_2}(r_1)) \iff \sigma_{F_1 \wedge F_2}(r_1) \iff \sigma_{F_2}(\sigma_{F_1}(r_1))$$

3. SelProj: Selektion und Projektion

   σ und π lassen sich kommutieren, sofern die Selektion auf den Projektionsattributen geschieht:

   $$\sigma_F(\pi_X(r_1)) \iff \pi_X(\sigma_F(r_1)) \text{ falls } attr(F) \subseteq X$$

   Sollte dies nicht der Fall sein, kann man zumindest einen Teil frühzeitig wegprojizieren, solange dieser die noch nötigen Selektionsattribute erhält:

   $$\pi_{X_1}(\sigma_F(\pi_{X_1 X_2}(r_1))) \iff \pi_{X_1}(\sigma_F(r_1)) \text{ falls } attr(F) \subseteq X_1 \cup X_2$$

4. SelJoin: Selektion und Verbund

   σ und $\bowtie$ kommutieren, falls alle Selektionsattribute aus nur einer Relationen stammen:

   $$\sigma_F(r_1 \bowtie r_2) \iff \sigma_F(r_1) \bowtie r_2 \text{ falls } attr(F) \subseteq R_1$$

   Bei strikt aufteilbarem Selektionsprädikat nach der Form $F = F_1 \wedge F_2$ gilt zusätzlich:

   $$\sigma_F(r_1 \bowtie r_2) \iff \sigma_{F_1}(r_1) \bowtie \sigma_{F_2}(r_2) \text{ falls } attr(F_1) \subseteq R_1 \text{ und } attr(F_2) \subseteq R_2$$

   Eine Abspaltung ist möglich, falls eines der beiden Selektionsprädikate nur eine Relation betrifft:

   $$\sigma_F(r_1 \bowtie r_2) \iff \sigma_{F_2}(\sigma_{F_1}(r_1) \bowtie r_2) \text{ falls } attr(F_1) \subseteq R_1$$

5. SelUnion: Selektion und Vereinigung

   Kommutierbarkeit von σ und $\cup$:

   $$\sigma_F(r_1 \cup r_2) \iff \sigma_F(r_1) \cup \sigma_F(r_2)$$

6. SelDiff: Selektion und Mengendifferenz

   Kommutierbarkeit von σ und $-$:

   $$\sigma_F(r_1 - r_2) \iff \sigma_F(r_1) - \sigma_F(r_2)$$

   beziehungsweise (da nur noch Herausstreichen der verbleibenden Schnittmenge):

$$\sigma_F(r_1 - r_2) \iff \sigma_F(r_1) - r_2$$

7. ProjJoin: Projektion und Verbund

   Kommutierbarkeit von π und $\bowtie$:

   $$\pi_X(r_1 \bowtie r_2) \iff \pi_X(\pi_{Y_1}(r_1) \bowtie \pi_{Y_2}(r_2))$$

   $$\text{mit } Y_1 = (X \cap R_1) \cup (R_1 \cap R_2) \text{ und } Y_2 = (X \cap R_2) \cup (R_1 \cap R_2)$$

   gestattet ein frühestmögliche Projektion, sofern die nötigen Verbundattribute noch nicht betroffen sind.

8. ProjUnion: Projektion und Vereinigung

   Kommutierbarkeit von π und $\cup$:

   $$\pi_X(r_1 \cup r_2) \iff \pi_X(r_1) \cup \pi_X(r_2)$$

3.2 Fortlaufendes Beispiel

Die nächsten Kapitel konzentrieren sich ganz auf die neu eingeführten Operatoren, die damit aufstellbaren Regeln und Beispiele zu diesen. Von daher ist es nur logisch, dass das Beispiel relativ einheitlich und nicht gänzlich abstrakt sein soll. Rein inhaltlich handelt es sich um das Thema Blutspende. Der vorrangige Grund ist dabei weniger inhaltlicher Natur, sondern die Darstellbarkeit. Die verschiedenen Wertedomänen heben sich nämlich deutlich voneinander ab und sind überschaubar ausgeprägt. Aufgrund der Vielzahl und grundverschiedenen Ausrichtung der Beispielen sollte man den inhaltlichen Aspekt, warum gerade dieses oder jenes anonymisierungswürdig ist, am Ende des Tages nicht übermäßig zerdenken. Obwohl man das Ganze also eher etwas abstrakter sehen sollte, sei nichtsdestotrotz die Bedeutung und Werteausprägung aller Attribute aufgeschlüsselt. Alles Nötige ist Tabelle 7 zu entnehmen.

Attr.	Bedeutung	Wertebereich
A	Spendetag	01.01. bis 31.12.
B	Blutgruppe (genotypisch)	AA, A0, BB, B0, AB, 00
C	Rhesusfaktor	$+$, $-$
D	Alter der Person	18 bis 73

Tabelle 7: Erklärung der einzelnen Attribute

3.3 Äquivalenzarten

In den Abschnitten 4.4, 5.5 und 6.5 kommen wiederholt verschiedene Äquivalenzarten zum Einsatz. Da einige allgemein bekannt sind, andere neu und kontextabhängig, seien sie im Folgenden einmal klar voneinander abgegrenzt. Tabelle 8 verdeutlicht die genaue Unterscheidung. Die Äquivalenz und deren Negation sollte allgemein bekannt sein. Neu ist die

sogenannte Teiläquivalenz: eine Nichtäquivalenz mit zusätzlichen Einschränkungen. Eine Mischform, wenn man so will.

Symbol	Name	Bedeutung	Unterteilung	Gebrauch
$\Longleftrightarrow$	Äquivalenz	bekannt	keine	ja
$\not\Longleftrightarrow$	Nichtäquivalenz	bekannt	keine	ja
$\sqsubseteq\!\!\Longleftrightarrow$	Teiläquivalenz	neu	Generalisierung	ja
			Slicing	ja
			DP	nein

Tabelle 8: Übersicht der Äquivalenzarten

Äquivalenz

Relationen sind standardgemäß Mengen, gegebenenfalls auch Multimengen, was heißt, dass zumindest die Reihenfolge nicht von Belang ist. Zwei Relationen gelten als äquivalent, sofern sie die gleiche Menge repräsentieren. Abbildung 3 zeigt exemplarisch zwei zueinander äquivalente Relationen.

A	B
01.01.	00
05.01.	00
31.01.	B0
20.03.	A0
20.03.	A0
31.03.	B0

$\Longleftrightarrow$

A	B
01.01.	00
20.03.	A0
20.03.	A0
05.01.	00
31.01.	B0
31.03.	B0

Abbildung 3: Äquivalenz

Nichtäquivalenz

Entsprechend ist Nichtäquivalenz die Negation von Äquivalenz. Am einfachsten lässt sich dies bereits an einer unterschiedlichen Zahl von Einträgen erkennen. Zum einen können Einträge werteseitig betrachtet gegenüber der zweiten Relation fehlen, zum anderen anders oft vorkommen. Abbildung 4 verdeutlicht die Nichtäquivalenz zweier Relationen.

A	B
01.01.	00
05.01.	00
31.01.	B0
20.03.	A0
20.03.	A0
31.03.	B0

$\not\Longleftrightarrow$

A	B
01.01.	00
05.01.	00
14.02.	00
31.03.	B0

Abbildung 4: Nichtäquivalenz

Teiläquivalenz

Der eigens eingeführte Begriff Teiläquivalenz bezeichnet eine Mischung aus Äquivalenz und Nichtäquivalenz. Nichtäquivalenz bezieht sich hierbei auf die Relationen im Ganzen, Äquivalenz auf spezielle Teilbereiche. Die genauen Äquivalenzforderungen sind im Rahmen der Arbeit kontextuell an die jeweilige Anonymisierungstechnik gebunden.

Im Folgenden nehmen wir einmal an, dass eine unanonymisierte Basisrelation r existiert. Ausgehend von dieser wurden auf unterschiedlichem Wege zwei anonymisierte Relationen r_1 und r_2 gebildet. Sie sind gemeinhin nicht äquivalent, doch nach wie vor gleich bezüglich der Zahl der Einträge und des Relationsschemas, sprich $R \equiv R_1 \equiv R_2$. Trotz allgemeiner Ungleichheit wegen nicht gleichbedeutenden Anonymisierungswegen ließe mittels der Teiläquivalenz ausdrücken, dass die gleichen essenziellen Informationen vorliegen würde, trotz der nicht äquivalenten Abweichungen.

Teiläquivalenz bei der Generalisierung

Bei der Generalisierung umfasst die Äquivalenzforderung die sensitiven und sonstigen Attribute. Die genaue Werteausprägung der Generalisierungsattribute kann dagegen sein, wie sie will. Um die Teiläquivalenz formal nicht derart zu überladen, lassen wir das minimale Anonymitätsmaß und die Generalisierungsattribute mal ganz außen vor.

Sei X die Menge der Generalisierungsattribute. Weiterhin sei $X \subseteq R$, $Y = R \setminus X$. Für Teilanonymität bei einer Generalisierung muss Folgendes gelten:

$$\pi_Y(r_1) \iff \pi_Y(r_2)$$

Abbildung 5 verdeutlicht beispielhaft die beschriebene Teiläquivalenz. Zu sehen ist eine Ausgangsrelation sowie zwei weitere Relationen mit verschieden ausgeprägter Generalisierung des Attributes A. Bei allen besteht eine Teiläquivalenz hinsichtlich Attribut B.

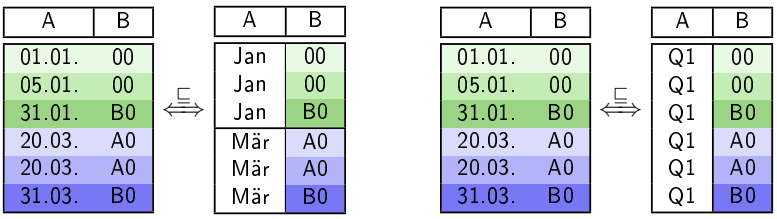


Abbildung 5: Teiläquivalenz hinsichtlich Generalisierung

Teiläquivalenz beim Slicing

Beim Slicing bezieht sich die Äquivalenzforderung auf die gebildeten korrelierenden Teilmengen. Man stelle sich hierzu eine vollständig disjunkte Aufteilung vom Relationsschema nach der Form $R = X_1 \cup ... \cup X_n$ vor. Für Teiläquivalenz beim Slicing muss dann gelten:

$$\forall\, i \in \{1, ..., n\} : \ \pi_{X_i}(r_1) \iff \pi_{X_i}(r_2)$$

Abbildung 6 zeigt exemplarisch die Teiläquivalenz beim Slicing. Auch hier sind eine Ausgangsrelation und zwei verschieden ausgeprägte Anonymisierungen zu sehen. Dass sich aufgrund der zufälligen Permutation weitaus mehr verschiedene Möglichkeiten als abgebildet gibt, dürfte klar sein. Bei gleichen horizontalen Splits täte dies aber nichts zur Sache, da alle Fälle gleiche Wahrscheinlichkeiten besäßen und man von echter Äquivalenz sprechen könnte. Teiläquivalenz bezieht sich hingegen eher auf den erweiterten Umstand, dass unterschiedlich gesetzte Horizontalsplits manche Vertauschungen verwehren und andere plötzlich ermöglichen könnten. Mit der Teiläquivalenz drückt man im Grunde aus, dass man diesen Umstand tolerieren würde.

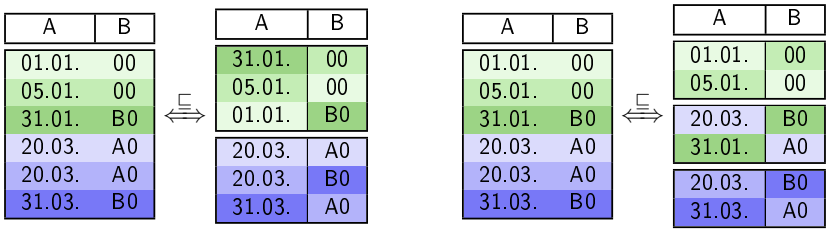


Abbildung 6: Teiläquivalenz hinsichtlich Slicing

4 Generalisierungs-Operator α^G

Dieses Kapitel widmet sich der operationalen Verwendung von Generalisierung hinsichtlich k-Anonymität. Zu aller erst wird das besagte Verfahren als formaler algebraischer Operator eingeführt. Es wird beschrieben, wie er sich zusammensetzt, was er kann und was nicht, wie er sich exemplarisch verhält und weiterhin wird ergründet, wie seine Beziehung zu anderen Operatoren aussieht.

4.1 Aufbau

Der Generalisierungs-Operator setzt sich wie folgt zusammen:

$$\alpha^G{}_{[X,k]}(r)$$

α^G	Bezeichner des Operators α steht für Anonymisierung G für das genaue Verfahren (Generalisierung)
X	Menge der Generalisierungsattribute, $X \subseteq R$
k	Grad der k-Anonymität, $k \in \mathbb{N}$
r	Basisrelation

Im Rahmen dieser Arbeit werden sich alle Untersuchungen bezüglich der Generalisierung auf k-Anonymität beschränken. Obendrein auf l-Diversität und t-closeness zu schauen, würde zu weit gehen. Im Sinne hierauf aufbauender Forschung ließe sich der Operator auch problemlos um l-Diversität und t-closeness erweitern. Ein wie hier rein auf k-Anonymität bedachter Operator wäre demnach ein Spezialfall. Für die anderen beiden Metriken wird angenommen, dass diese uneingeschränkt gelten. Im Weiteren gilt daher überall 1-Diversität und 1-closeness. Erstere gestattet bereits uniforme Äquivalenzklassen, letztere erlaubt bei der Distanzmessung beliebige Entfernungen, was anders gesagt heißt, dass keinerlei minimale Ähnlichkeit zur Gesamtverteilung bestehen muss.

$$\alpha^G{}_{[X,k]}(r) = \alpha^G{}_{[X,k,l=1,t=1]}(r)$$

4.2 Annahmen und Einschränkungen

Damit der vorgestellte Operator im Sinne der damit aufgestellten Regeln sowohl minimalistisch als auch vielseitig ist, sei der Klarheit wegen explizit genannt, wozu er imstande ist und wozu nicht. Es sollte relativ klar sein, dass eine minimale Anonymisierung angestrebt wird. Die Verallgemeinerung ist daher nicht stärker als nötig.

Bei mehreren Generalisierungsattributen kann es aufgrund vielfältiger Generalisierungsebenen natürlich mehrere Möglichkeiten geben, wie sich eine Generalisierung erreichen ließe – zugunsten des einen oder aber des anderen Attributes. Welche der Konstellation nun gewählt wird, ist schlussendlich gleichgültig, da der zugrundeliegende Generalisierungsalgorithmus für uns nicht von Interesse ist.

Des Weiteren werden alle Werte innerhalb eines Generalisierungsattributes zu Werten gleicher Ebene zusammengefasst. Auf diese Weise wird untereinander eine klare, werteseitige Abgrenzung sichergestellt.

Außerdem wird angenommen, dass die Generalisierung nach eigenem Ermessen erfolgt. Die genaue Wahl der Quasi-Identifikatoren muss somit selbstständig geschehen, nach bestem Wissen und wird nicht weiter in Frage gestellt.

Zudem sind die angegebenen Generalisierungsattribute auch wirklich Teil der Ausgangsrelation. Darüber hinaus existiert zu jedem dieser Attribute genau eine zielführende Hierarchie. Es gäbe nämlich auch die Möglichkeit, dass mehrere vorliegen, unter denen man dann entweder auswählt oder aber automatisch die bessere genommen werden würde.

Weiterhin sei klargestellt, dass sich die Generalisierungshierarchien immer nur an Einzelattribute richten. Es gibt nämlich auch Techniken, bei denen sie im Sinne der Generalisierung auf mehrere ausgelegt sein können.

Darüber hinaus wird prinzipiell vorausgesetzt, dass für jeden zu anonymisierenden Datenbestand mindestens so viele Einträge vorliegen, wie es der Anonymitätsgrad verlangt. Wenn die Einträge nicht einmal für eine einzige k-große Gruppe reichen, ist es unmöglich, die geforderte Anonymität in irgendeiner Form zu erfüllen.

4.3 Beispiel

Zum genauen Verständnis des α^G-Operators sei dessen Wirkungsweise am besten an einem Beispiel erklärt. Eine kurze inhaltliche Einordnung des Ganzen ist in Abschnitt 3.2 zu finden.

Die Ausgangsrelation r setzt sich aus dem Schema $R = \{A, B, C, D\}$ zusammen. Ebendiese soll mittels Generalisierung nun 3-anonym gemacht werden. Die zu generalisierenden Attribute sind hierbei A und C. C steht für den Rhesusfaktor, A für irgendeinen Tag innerhalb eines Jahres. Verkürzt gesagt liegen dem Ganzen folgende Generalisierungen zugrunde:

A: Tag $\longrightarrow$ Monat $\longrightarrow$ Quartal $\longrightarrow$ Semester $\longrightarrow$ irgendwann

C: $+ \longrightarrow \pm$, $- \longrightarrow \pm$

Eine ausführlichere Baumdarstellung ist weiter unten in Abbildung 8 zu finden. Aus all dem ergibt sich dann folgender Ausdruck:

$$r_a := \alpha^G{}_{[X=\{A,C\},k=3]}(r)$$

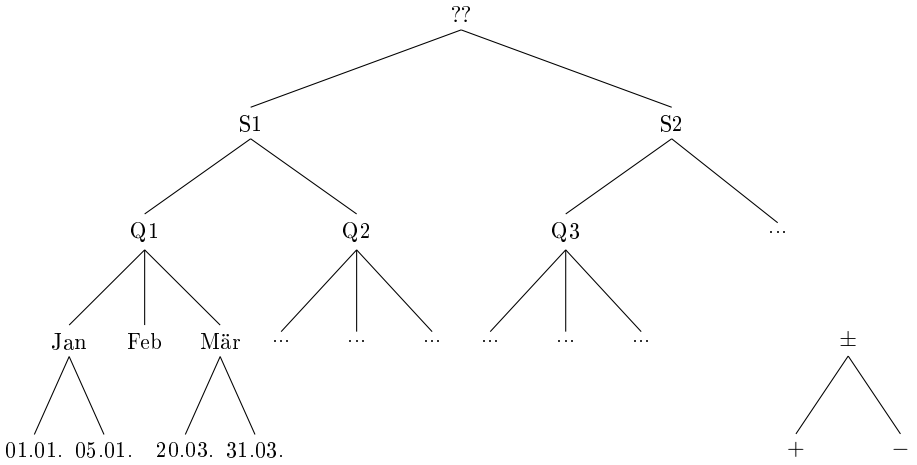
Abbildung 7 zeigt an einem exemplarischen Datenbestand, wie das Resultat aussehen könnte. Um hinsichtlich der Tage 3-anonyme Gruppen zu schaffen, bedarf es mindestens einer Generalisierung zu Monaten, und hinsichtlich des Rhesusfaktors bleibt keine andere Wahl als stärkstmöglich zu generalisieren. Dies wäre grundsätzlich natürlich nur eine Einzelbetrachtung und darauf aufbauend müsste geschaut werden, zu wessen Gunsten man die Forderung im Zusammenspiel erfüllten könnte. In diesem Fall erbringen A und C die geforderte Anonymität aber ebenso in Kombination.

A	B	C	D
01.01.	00	+	18
05.01.	00	+	24
05.01.	B0	−	32
20.03.	A0	+	36
20.03.	A0	−	42
31.03.	B0	+	56

$\Longrightarrow$

A	B	C	D
Jan	00	±	18
Jan	00	±	24
Jan	B0	±	32
Mär	A0	±	36
Mär	A0	±	42
Mär	B0	±	56

Abbildung 7: links r, rechts r_a



(a) DGH_A (b) DGH_C

Abbildung 8: Generalisierungshierarchien

4.4 Regeln

Ausgehend vom Generalisierungs-Operator wurde dessen Mehrfachanwendung, dessen Redundanz, und sein Verhältnis zur Projektion, Selektion, Vereinigung, Verknüpfung, Schnitt- und Differenzmenge ergründet.

1. ObsoleteGen: Obsolete Generalisierung

   Die folgenden drei Regeln sind zwar eher trivial und aufgrund der geltenden Annahmen bekannt sowie im Weiteren ausgeschlossen, doch sollen auch auf Algebra-Ebene nicht unerwähnt bleiben.

   Generalisierung bei k=1 ist ein leicht sonderbarer Fall, da 1-Anonymität auf absolut jeden existierenden Datensatz zutrifft. Von Anonymität zu sprechen, mag daher gewissermaßen widersinnig erscheinen. Da es keiner wirklichen Bildung von Äquivalenzklassen bedarf, muss keine Anonymisierung betrieben werden.

   $$\text{(G1)}\quad \alpha^G{}_{[X,k=1]}(r_1) \iff r_1$$

Falls keine Generalisierungsattribute angegeben werden oder keines der angegebenen in der Relation verkommen sollte, gibt es auch in diesem Fall keinen Grund für eine Anonymisierung.

$$\text{(G2)} \quad \alpha^G{}_{[X=\{\},k=a]}(r_1) \iff r_1$$

$$\text{(G3)} \quad \alpha^G{}_{[X,k=a]}(r_1) \iff r_1 \text{ falls } attr(X) \cap R_1 = \{\}$$

2. IdemGen: Idempotentes Verhalten der Generalisierung und generelle Mehrfachanwendung

Generalisierung selbst verhält sich idempotent. Eine Auswahl an Generalisierungsattributen, die bereits in erster Instanz a-anonym gemacht wurde, bedarf keiner weiteren Generalisierung unter den gleichen Gesichtspunkten, da die erste Instanz für genau dies bereits gesorgt hat.

$$\text{(G4)} \quad \alpha^G{}_{[X,k=a]}(\alpha^G{}_{[X,k=a]}(r_1)) \iff \alpha^G{}_{[X,k=a]}(r_1)$$

Abbildung 9 verdeutlicht noch einmal beispielhaft die Regeläquivalenz. In diesem und noch vielen folgenden Beispielen ist die Basisrelation stets jene, an welche sowohl ein linksseitiger als auch ein rechtsseitiger Pfeil angrenzt. Linksseitig wird der linke Ausdruck der Regel durchexerziert, rechtsseitig der rechte. Ausgehend von der Basisrelation wird dann von innen nach außen gearbeitet. Links außen steht also das Endergebnis des linken Ausdrucks, rechtsseitig des rechten. Je nachdem, welche dazugehörige Regel verdeutlicht werden soll, können diese zueinander äquivalent, nicht-äquivalent oder teiläquivalent sein. Die genaue Bedeutung dieser drei Begriffe ist im Abschnitt davor zu finden.

Bei X={A}, a=3 ergeben sich folgende Relationen:

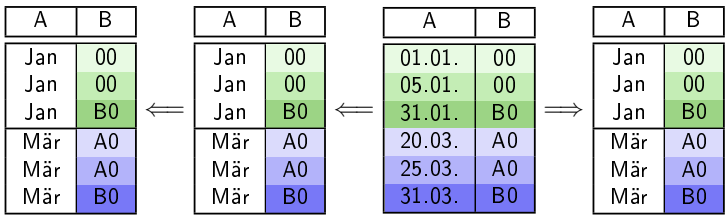


Abbildung 9: Idempotenz der Generalisierung

Ferner sind bei gleicher Anonymitätsforderung nicht nur nachfolgende Instanzen obsolet, wenn diese die gleichen Generalisierungsattribute aufweisen, sondern generell dann, sofern es sich um eine Teilmenge handeln sollte, weil all diese hinreichend abgedeckt werden.

$$\text{(G5)} \quad \alpha^G{}_{[Y,k=a]}(\alpha^G{}_{[X,k=a]}(r_1)) \iff \alpha^G{}_{[X,k=a]}(r_1) \text{ falls } Y \subseteq X$$

Auch bei einer Verschärfung des Generalisierungsgrades ließe sich vereinfachen. Sollte auf nur einem Attribut generalisiert werden, kann man äquivalenterweise direkt zur Verschärfung springen, da die Generalisierungsdomain ja gerichtet verläuft und keinerlei Abstimmung mit anderen Attributen notwendig gewesen ist.

Bei $a > b$:

$$\text{(G6)} \quad \alpha^G{}_{[X,k=a]}(\alpha^G{}_{[X,k=b]}(r_1)) \iff \alpha^G{}_{[X,k=a]}(r_1) \text{ falls } |X| = 1$$

Abbildung 10 soll noch einmal verdeutlichen, dass eine Äquivalenz in diesem Falle problemlos möglich ist.

Bei X={A}, a=5, b=3 ergeben sich folgende Relationen:

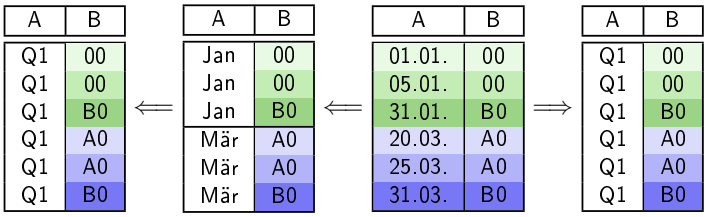


Abbildung 10: Verschärfung der Generalisierung bei nur einem Attribut

Bei mehreren Generalisierungsattributen ist das Resultat dagegen nicht mehr zwingend ergebnisäquivalent. Die direkte, stärkere Generalisierung ist dabei allemal die aussagekräftigere, da der volle Informationsgehalt der Basisrelation zur Verfügung steht und aus praktischer Sicht daher klar vorzuziehen. Da sich die Verallgemeinerung bekanntlich nicht umkehren lässt, erreicht eine Mehrfachgeneralisierung aufgrund der zwischenschrittigen Informationsstände weitaus schneller den höchstmöglichen Generalisierungsgrad.

$$\text{(G7)} \quad \alpha^G{}_{[X,k=a]}(\alpha^G{}_{[X,k=b]}(r_1)) \overset{\sqsubseteq}{\Longleftarrow} \alpha^G{}_{[X,k=a]}(r_1) \text{ falls } |X| \geq 2$$

Abbildung 11 soll noch einmal verdeutlichen, dass der linksseitige Generalisierungsweg einen potenziellen Informationsverlust zur Folge hat. Bei der Generalisierung der Zwischenrelation lassen sich nur noch höhere Domainebenen ergründen, tiefere aber nicht mehr. Dabei bedarf es bei stärkerer Anonymisierung jedoch sowohl höherer als auch tieferer Ebenen, um die kleinstmöglichen Gruppen bilden zu können. Attribut C muss im Beispiel zwar generalisiert werden, um auf kleinstmögliche Weise 2-Anonymität zu erreichen, für 3-Anonymität aber nicht. Da dies die zweite Instanz aber nicht mehr ergründen kann, kann sie nur noch A verallgemeinern, sodass rein praktisch nur noch die allgemeinste Generalisierung verbleibt.

Bei X={A,B,C}, a=3, b=2 ergeben sich folgende Relationen:

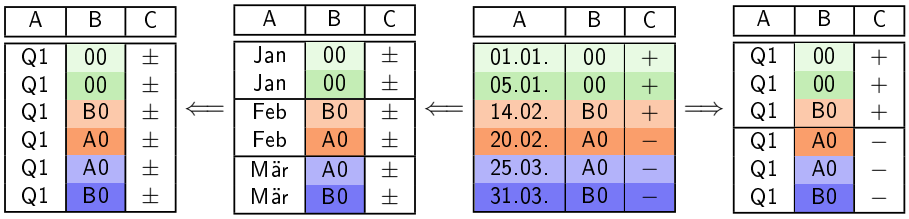


Abbildung 11: Verschärfung der Generalisierung bei mehreren Attributen

Sollte nach einer bestimmten Generalisierung anschließend eine schwächere gefordert werden, so kann man die schwächere ignorieren. b-Anonymität gewährleistet nämlich bereit eine Anonymität der Stärke a. Die stärkere Anonymisierung ist natürlich bindend, sodass etwas wie eine Umkehr bzw. Rückführung auf eine geringere Stufe niemals stattfinden würde, geschweige denn möglich wäre.

Bei $a < b$:

$$\text{(G8)} \quad \alpha^G{}_{[X,k=a]}(\alpha^G{}_{[X,k=b]}(r_1)) \iff \alpha^G{}_{[X,k=b]}(r_1)$$

Bei X={A}, a=3, b=5 ergeben sich folgende Relationen:

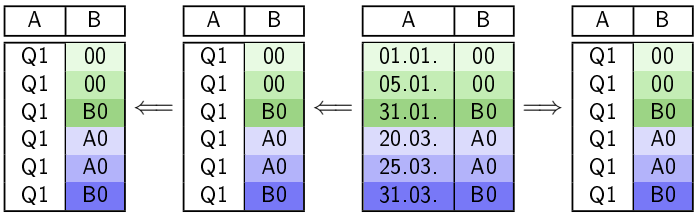


Abbildung 12: Abdeckung einer schwächeren Generalisierung durch eine stärkere

3. GenProj: Generalisierung und Projektion

Zur genauen Verdeutlichung der Beziehung der beiden Operatoren muss im Folgenden zwischen mehreren Fällen unterschieden werden, da einiges, aber nicht alles möglich ist. Falls die Generalisierung auf Basis einer Projektion geschieht, gemeinhin also $X \subseteq Y$ gilt, ließe sich die Generalisierung in diesem Fall problemlos vorziehen, da auf ohnehin wegprojizierten Attributen keinerlei Generalisierung stattfinden sollte.

$$\text{(G9)} \quad \alpha^G{}_{[X,k=a]}(\pi_Y(r_1)) \Longrightarrow \pi_Y(\alpha^G{}_{[X,k=a]}(r_1))$$

Die Rückrichtung beziehungsweise ein Vorziehen der Projektion ist möglich, sofern die Tatsache von eben nun als hinreichende Bedingung gilt, also alle Generalisierungsattribute so oder so erhalten bleiben würden.

$$\text{(G10)}\quad \alpha^G{}_{[X,k=a]}(\pi_Y(r_1)) \Longleftarrow \pi_Y(\alpha^G{}_{[X,k=a]}(r_1)) \text{ falls } X \subseteq Y$$

Abbildung 13 verdeutlicht noch einmal die gleichbedeutende Beziehung des Ganzen.

Bei X={A}, a=3, Y={A,B} ergeben sich folgende Relationen:

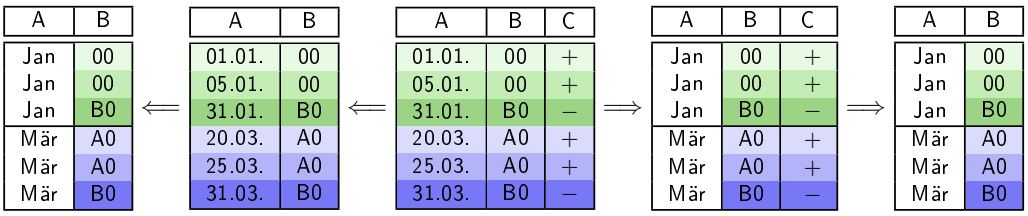


Abbildung 13: Allgemeine Tauschbarkeit bei Generalisierung auf Projektionsattributen

Des Weiteren kann man noch sagen, dass der Generalisierungsaufwand grundsätzlich obsolet wäre, wenn alle generalisierten Attribute schlussendlich ausgeblendet werden würden.

$$\text{(G11)}\quad \pi_Y(r_1) \Longleftarrow \pi_Y(\alpha^G{}_{[X,k=a]}(r_1)) \text{ falls } X \cap Y = \{\}$$

In Abbildung 14 ist der Sachverhalt noch einmal bildlich dargestellt.

Bei X={A}, a=3, Y={B,C} ergeben sich folgende Relationen:

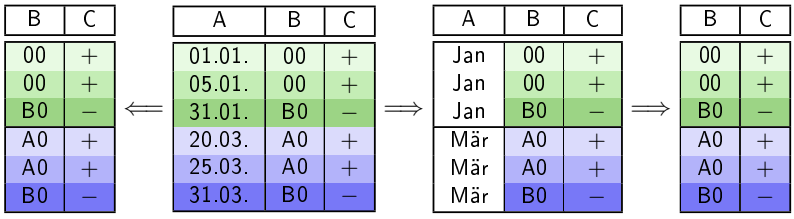


Abbildung 14: Obsolete Generalisierung bei disjunkter Projektion

Was hinsichtlich vorgezogener Projektion jetzt noch verbleibt, sind Fälle, bei denen $X \not\subseteq Y$ gilt. Eine direkte, vollständige Vertauschung ist aus Äquivalenzsicht hierbei nicht mehr möglich, da dies einen einschneidenden Einfluss auf die Generalisierung und deren Attribute hätte. Nichtsdestotrotz ließe sich eine Zwischenprojektion vorziehen. Man könnte nämlich zumindest auf Grundlage der Generalisierungsattribute und der eigentlichen Attribute projizieren, also all jene sensitiven und sonstigen Attribute herausstreichen, die ohnehin ausgeblendet würden.

$$\text{(G12)}\quad \pi_Y(\alpha^G{}_{[X,k=a]}(\pi_{Y'}(r_1))) \Longleftarrow \pi_Y(\alpha^G{}_{[X,k=a]}(r_1)) \text{ mit } Y' = X \cup Y$$

Abbildung 15 zeigt die frühestmögliche Verkleinerung der Relation unter Wahrung der Äquivalenz. Außerdem ist erkennbar, dass jene Generalisierungsattribute, die am Ende wegprojiziert werden, gewissermaßen hinderlich sind, da sie von einer potenziell feineren Anonymisierung abhalten. Würde C aus der Generalisierung herausfallen, ließe sich 3-Anonymität bei A schon durch eine Verallgemeinerung auf Monate abdecken.

Bei Y={A,B}, X={A,C}, a=3 und daraus folgend Y'={A,B,C} ergeben sich folgende Relationen:

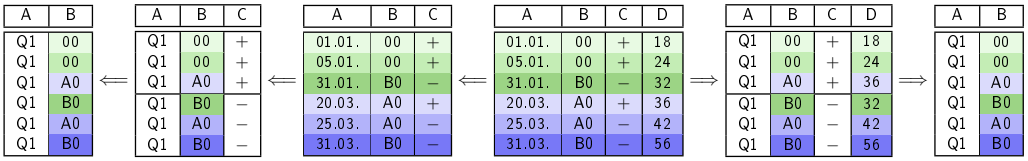


Abbildung 15: Vorziehbarkeit der Projektion bei Einfluss auf die Generalisierung

Falls man keine Ergebnisäquivalenz anstrebt, sondern eher auf eine möglichst informationserhaltende Anonymisierung aus ist, kann die Projektion auch gänzlich vorgezogen werden. Zur ordnungsgemäßen Vorziehung müssen dafür alle Generalisierungsattribute, die wegprojiziert werden, aus der Generalisierung herausgestrichen werden. Da auf ohnehin überflüssige Generalisierungshierarchien nun nicht mehr geachtet werden muss, können so manche Werte weitaus feiner ausfallen, sodass diese Regel durchaus praktischen Nutzen besitzt.

$$\text{(G13)}\quad \alpha^G{}_{[X',k=a]}(\pi_Y(r_1)) \overset{\sqsubseteq}{\Longleftarrow} \pi_Y(\alpha^G{}_{[X,k=a]}(r_1)) \text{ mit } X' = X \setminus (R_1 \setminus Y)$$

Abbildung 16 verdeutlicht noch einmal den Informationsgewinn, den man aufgrund des Herausstreichens haben kann.

Bei X={A,C}, a=3, Y={A,B} und folglich X'={A} ergeben sich folgende Relationen:

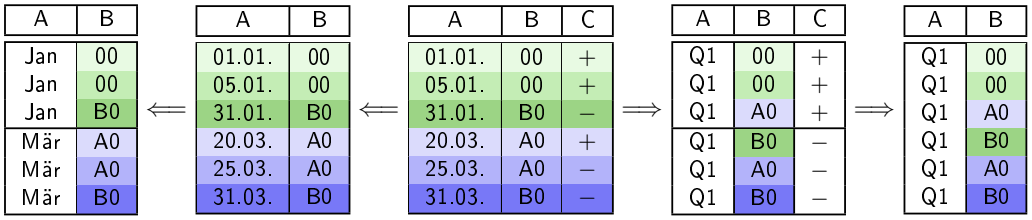


Abbildung 16: Vorziehbarkeit der Projektion bei Einfluss auf die Generalisierung

4. GenSel: Generalisierung und Selektion

Eine äquivalente Vertauschung zwischen Selektion und Generalisierung geht per se schon einmal nicht. Alle Generalisierungsattribute der Einträge eines Datenbestandes bestimmen maßgeblich, wie stark in jedem speziellen Fall verallgemeinert werden muss, um der

geforderten Anonymität zu genügen. Eine Selektion mit Generalisierungsattributen kann daher nicht gelingen, da sich generalisierte Werte grundsätzlich einer Selektion entziehen. Eine Mitgeneralisierung des Selektionsausdruckes oder eine vorsorglich aufgestellte Selektion auf mehreren Generalisierungsebenen mittels logischem Oder würde da auch nicht weiterhelfen. Was aber möglich wäre, ist eine Selektion rein auf Basis der unveränderten sensitiven uns sonstigen Attribute. Die Entfernung der Einträge bliebe eindeutig, kann aber verschieden starke Generalisierungen zur Folge haben. Bei einer frühen Selektion wäre die Relation zugunsten aller weiteren Kosten natürlich kleiner und die gebildeten Äquivalenzklassen wären immer noch in Gänze erkennbar. Da die Gruppen erst auf Basis der selektierten Menge gebildet werden, sind aber Einbußen bezüglich des Informationsgehaltes wahrscheinlich. Bei einer späten Selektion ist potenziell ein höherer Aussagegehalt bezüglich der Generalisierung möglich, da die vielfältigere Ausgangsmenge zur Verfügung steht. Im Anschluss müssen dann aber selektiv die geformten Gruppen zerstört werden. Am Ende mögen zwar nicht mehr alle Vertreter verbleiben, an der eigentlichen k-Anonymität ändert dies selbstredend aber nichts. Trotz Nichtäquivalenz kann eine Vertauschung also durchaus nützlich sein. Keiner der beiden Richtungen ist aber klar vorteilhafter bzw. überlegen.

$$\text{(G14)}\quad \alpha^G{}_{[X,k=a]}(\sigma_F(r_1)) \overset{\sqsubseteq}{\Longleftrightarrow} \sigma_F(\alpha^G{}_{[X,k=a]}(r_1)) \text{ falls } attr(F) \cap X = \{\}$$

Abbildung 17 zeigt exemplarisch das zuvor Beschriebene. Die selektierten Einträge sind natürlich die selben. Die linksseitig gebildeten Gruppen sind intakt, aber in diesem Beispiel recht allgemein. Die rechtsseitig gebildeten Gruppen mögen zwar mittlerweile unvollkommen sein, sind aus Informationssicht aber ergiebiger.

Bei X={A}, a=3, F: B≠B0 ergeben sich folgende Relationen:

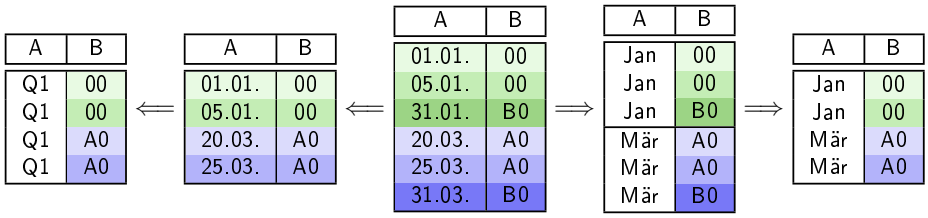


Abbildung 17: Generalisierung und Selektion

5. GenUnion: Generalisierung und Vereinigung

Eine Vertauschung von Vereinigung und Generalisierung folgt keiner klaren Äquivalenz, da jede mögliche Aufteilung im Einzelnen verschiedene Generalisierungen zur Folge haben kann. Eine Generalisierung auf der Gesamtheit sorgt für einheitliche Generalisierungsebenen und dahingehend sich nicht überschneidende Werte. Eine verteilte Generalisierung ist zugunsten einer Parallelisierung möglich und kann im Einzelnen die Aussagekraft steigern, wenn einige Ausreißer gesondert anonymisiert werden würden. Da unter Umständen verschieden starke Generalisierungsebenen vereinigt werden, lassen sich anders als bei einer Generalisierung auf der Gesamtheit manche Äquivalenzklassen nicht mehr

klar voneinander abgrenzen. Dass Generalisierungsebenen bei Partitionen keinem klaren Muster folgen, ist noch ein wenig in Anhang A untersucht worden.

$$\text{(G15)} \quad \alpha^G{}_{[X,k=a]}(r_1 \cup r_2) \overset{\sqsubseteq}{\Longleftrightarrow} \alpha^G{}_{[X,k=a]}(r_1) \cup \alpha^G{}_{[X,k=a]}(r_2)$$

Abbildung 18 soll noch einem die allgemeine Nichtäquivalenz verdeutlichen. Es verschwinden natürlich keine Einträge, doch die Generalisierung der horizontalen Partitionen kann verschieden ausgehen. In diesem Beispiel ist die Aufteilung eher ungünstig gewählt, sodass eine Verschlechterung des Aussagegehaltes eintritt. Dass ebenso eine Verbesserung eintreten kann, ist Anhang A zu entnehmen, genauer gesagt Abbildung 55. In dieser zusätzlichen Untersuchung wurde geschaut, ob man an gleichen oder verschiedenen Generalisierungsebenen zweier Partitionen in irgendeiner Form eine Verbesserung oder aber Verschlechterung gegenüber der Gesamtrelation ausschließen könnte.

Bei X={A}, a=2 ergeben sich folgende Relationen:

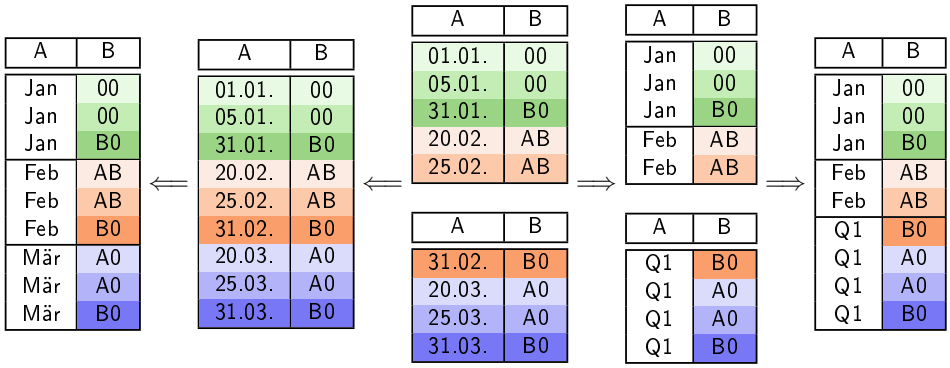


Abbildung 18: Generalisierung und Vereinigung

6. GenJoin: Generalisierung und Verbund

Verbund und Generalisierung lassen wenig Spielraum für Regeln, geschweige denn nutzbringende. Eine verteilte Anonymisierung unterschiedlicher Generalisierungsattribute bringt beispielsweise nichts, da die Partitionen zwar k-anonym wären, doch die gebildeten Gruppen dies im Verbund wahrscheinlich nicht mehr sind. Zudem sind Generalisierungsattribute als verbindende Elemente grundsätzlich ungeeignet – zum einen wegen wachsenden Wertekombinationen aufgrund der Verallgemeinerung, zum anderen aufgrund der völlig verschiedenen Generalisierungsebenen.

Eine verbleibende Möglichkeit wäre eine geschlossene Generalisierung, die nur eine Partition betreffen würde. Die andere Partition wäre dann eine Auslagerung von einigen sensitiven und sonstigen Attributen. Auf Basis von ebendiesen müsste man dann theoretisch auch den Verbund durchführen. Praktisch ist aber prinzipiell davon abzuraten, da dieser Verbund potenziell neue Quasi-Identifikation mit ins Spiel bringen würde. Die Bache-

lorarbeit „Berechnung von Quasi-Identifikatoren nach Verbundoperationen“ von Florian Rose hat sich mit ebendieser Thematik einmal ausführlich auseinandergesetzt [Ros19].

Man könnte das Problem mit den Quasi-Identifikatoren natürlich beseitigen, wenn man die verbindenden Attribute am Ende ausblenden würde. Ein Erhalt wäre höchstens denkbar, wenn man beispielsweise fordern würde, dass jede verbindende Attributkombination mindestens so viele Einträge hervorbringt wie der geforderte Anonymitätswert. Das allgemeine Nachteil dieser Technik ist, dass Verbundwerte, die viele Kombinationen besäßen, am Ende riesige Gruppen zur Folge hätten. Der Aussagegehalt wäre daher entsprechend schlecht und eine späte Generalisierung für einen besseren klar vorzuziehen.

Da die neuste Regel eine komplexere Attributeinteilung und Vielzahl an Randbedingungen aufweist, sei die Beschreibung der Relationen diesmal klar von der Regel an sich getrennt.

R_1:	erstes Relationenschema	
	X	Generalisierungsattribute
	S_1	sensitive und sonstige Attribute
	Y	Verbundattribute

R_2:	zweites Relationenschema	
	S_2	sensitive und sonstige Attribute
	Y	Verbundattribute

Wirkungsbereich und Beziehungen	
R_1:	$R_1 = X \cup S,\ \ R_1 \setminus X = S_1,\ \ Y \subseteq S_1$
R_2:	$R_2 = S_2,\ \ Y \subseteq S_2$
R_1, R_2:	$R_1 \cap R_2 = Y$

(G16) $\alpha^G{}_{[X,k=a]}(r_1 \bowtie r_2) \overset{\sqsubseteq}{\Longleftrightarrow} \alpha^G{}_{[X,k=a]}(r_1) \bowtie r_2$ bei Verbund mittels Y

Abbildung 19 zeigt eine komprimierte Generalisierung mit nachträglicher Eingliederung von sensitiven und sonstigen Attributen. Da die Verbundattribute hier nur der Verbindung dienen und Ableitung oder gar Identifizierbarkeit einräumen könnten, werden sie letzten Endes ausgeblendet. Zu sehen ist die Kompaktheit der zu generalisierenden Relation, die Wahrung der Anonymität, das deutliche Wachstum der Äquivalenzklassen aufgrund des Verbundes sowie der geringere Informationsgehalt aufgrund der Komprimierung.

Bei X={A}, a=3, Verbund mittels VA ergeben sich folgende Relationen:

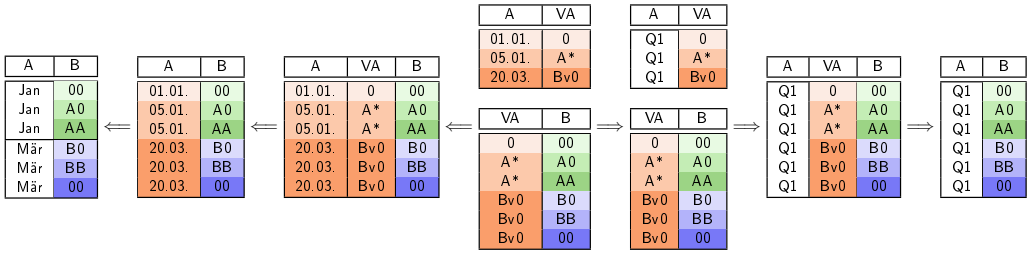


Abbildung 19: Generalisierung und Verbund auf sensitiven und sonstigen Attributen

Ein Spezialfall wäre, falls es sich bei den Relationen obendrein um vertikale Partitionen handeln würde. Die Vertauschung wäre dann nicht nur teiläquivalent, sondern sogar äquivalent. Die Verbund-ID muss dabei grundsätzlich alle Einträge abdecken, da herausfallende Einträge ansonsten wieder die Generalisierung beeinträchtigen würden. Da die Verbundattribute hier eindeutig als Schlüssel gebraucht werden, ist deren letztendliche Ausblendung nun Pflicht.

$$\text{(G17)} \quad \alpha^G{}_{[X,k=a]}(\pi_R(r_1 \bowtie r_2)) \iff \pi_R\ (\alpha^G{}_{[X,k=a]}(r_1) \bowtie r_2)$$

$$\text{bei Verbund mittels } Y \text{ und } R = (R_1 \cup R_2) \setminus Y$$

Abbildung 20 verdeutlicht unter den aufgestellten Bedingungen noch einmal die Äquivalenz des Ganzen.

Bei X={A}, a=3, Verbund mittels ID ergeben sich folgende Relationen:

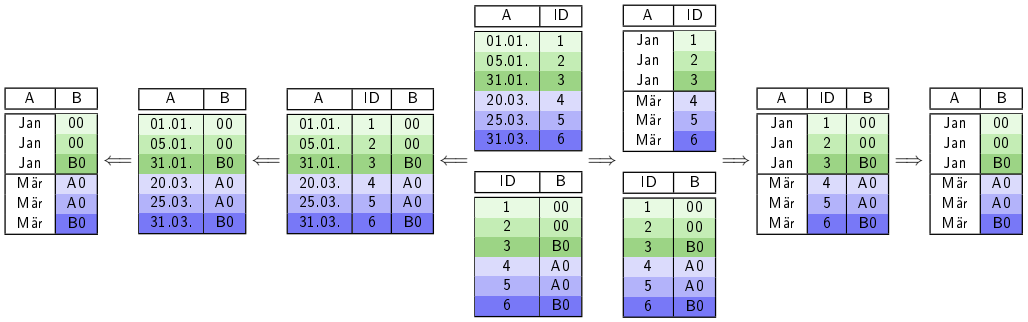


Abbildung 20: Generalisierung und Verbund bei vertikaler Partitionierung sensitiver und sonstiger Attribute

7. GenInter, GenDiff: Generalisierung bei Schnittmenge und Differenzmenge

Aufgrund der fallabhängigen Generalisierungsebenen sind Schnitt- und Differenzmengenbildung nicht verschiebbar. Die genaue Werteausprägung der Generalisierungsattribute

hängt stets vom Datensatz ab, sodass es bei unabhängigen Teilbeständen nicht einmal gewiss ist, ob überhaupt die gleichen Generalisierungsebenen zustande kämen. Selbst wenn, haben die Einträge aufgrund der Generalisierung in ihrer Gesamtheit an Verschiedenheit eingebüßt. Aus diesem Grunde ist auf rechtsseitigem Wege selbst die Zahl der Einträge ungewiss.

$$\text{(G18)} \quad \alpha^G{}_{[X,k=a]}(r_1 \cap r_2) \not\Longleftrightarrow \alpha^G{}_{[X,k=a]}(r_1) \cap \alpha^G{}_{[X,k=a]}(r_2)$$
$$\text{(G19)} \quad \alpha^G{}_{[X,k=a]}(r_1 - r_2) \not\Longleftrightarrow \alpha^G{}_{[X,k=a]}(r_1) - \alpha^G{}_{[X,k=a]}(r_2)$$

Abbildung 21 zeigt, dass ein Unterschied von nur einem Datensatz eine gänzlich andere Generalisierung und dadurch völlige Wertefremdheit zur Folge haben kann.

Bei X={A}, k=2 ergeben sich verdeutlicht an der Schnittmenge folgende Relationen:

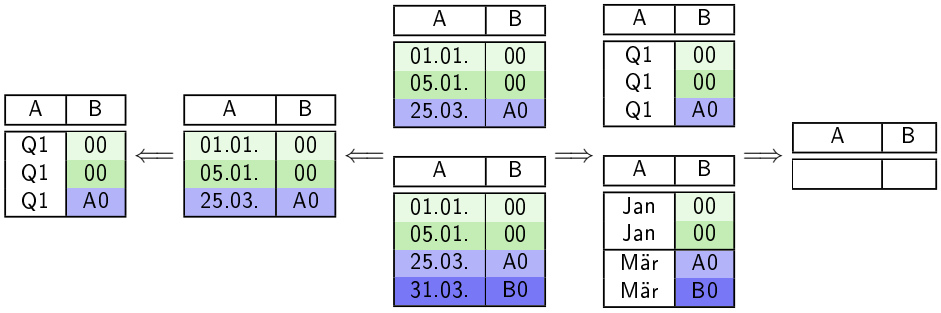


Abbildung 21: Generalisierung und Schnittmenge

5 Slicing-Operator α^S

Dieses Kapitel widmet sich der Operationalisierung des Slicings. Der eingeführte Operator wird hinreichend erklärt und anschließend ausgiebig dessen Anwendung ergründet. Untersucht wurde obsoleter Gebrauch, Mehrfachanwendung sowie die Beziehung zu Projektion, Selektion, Vereinigung, Verknüpfung, Schnitt- und Differenzmenge.

5.1 Aufbau

Der Slicing-Operator setzt sich wie folgt zusammen:

$$\alpha^S{}_{V=\{V_1,\ldots,V_n\}}(r)$$

α^G	Bezeichner des Operators
	α steht für Anonymisierung
	S für das genaue Verfahren (Slicing)
V	Menge der Korrelationsmengen
V_i	Korrelationsmenge, $V_i \subseteq R$
r	Basisrelation

Eine formale Beschreibung im Detail folgt im anschließenden Abschnitt, da dies von einigen der getroffenen Annahmen abhängt.

5.2 Annahmen und Einschränkungen

Um den beschriebenen Operator so schlicht wie möglich zu halten, mussten einige Dingen festgelegt werden. Auf die horizontale Trennung wird man grundsätzlich keinen Einfluss haben können. Der Vorgang wird immer mittels einer temporären Hilfs-ID vollzogen. Man wird keinen weiteren Einfluss auf sie haben können, sodass es rein äußerlich so erscheint, als habe sie nie existiert.

Die genaue horizontale Aufteilung soll für alles Weitere unbekannt bleiben. Es wäre ohnehin eine Schwachstelle für Angriffe wie bei der k-Anonymität. Der Operator hängt also eigentlich noch dazu von einem Algorithmus für die Festlegung der horizontalen Splits ab, doch im Rahmen dieser Arbeit interessiert er nicht weiter. Man kann sich aber zumindest vorstellen, dass irgendeine natürliche Zahl existiert, die vorgibt, wie groß eine jede Teilrelation mindestens sein muss, und dass irgendwie der Restanteil an Einträgen verteilt werden muss.

Statt randomisierter Sortierung könnte man die Korrelationsbeziehungen natürlich auch mit jeder anderen auflösen. Dies wäre aber höchstens eine Spielerei, sodass auf manuelle Sortierangaben verzichtet wird (anstelle zusätzlicher Sortierangaben).

Anders als die horizontale Aufteilung, wo vollständig disjunkt voneinander abgegrenzt wird, bestünde vertikal bekanntlich die Möglichkeit, dies nicht tun zu müssen, also Attribute in mehr als nur eine Korrelationsgruppe zu stecken. Auf diesen Aspekt wird aber im Folgenden verzichtet. Es ist formal daher weder eine Ausblendung noch eine Mehrfachverwendung möglich. Zur Beseitigung unbrauchbarer Attribute existiert ja nicht umsonst die Projektion. Der Mehrfachgebrauch von Attributen würde zwar zusätzliche Möglichkeiten bieten, doch

ebenso neue Probleme schaffen. Überlappungen könnten nämlich hochwahrscheinliche oder gar definitive Rückschlüsse erlauben. Zudem müsste man dann irgendwie die Frage nach der Namenseindeutigkeit klären. Außerdem wird wie bei der Generalisierung angenommen, dass die Zusammenstellung der Korrelationsgruppen nach bestem Wissen getroffen wurde. Die Auswahl wird also nicht weiter in Frage gestellt.

5.3 Ausführliche Formalisierung

Im Folgenden wird veranschaulicht, wie die genaue Zerteilung, Neuordnung und Zusammenführung der Basisrelation aussieht. Die horizontale Aufteilung muss per Definition vollständig disjunkt verlaufen. Formal heißt das:

$$r := \sigma_{H_1}(r) \cup ... \cup \sigma_{H_n}(r)$$

$$\sigma_{H_i}(r) \cap \sigma_{H_j}(r) = \{\} \quad \text{für} \quad \forall i, \forall j : \; i, j \in \{1, ..., n\} \quad \text{und} \quad i \neq j$$

Bei allen H-Ausdrücken handelt es sich im Übrigen um Selektionsprädikate, durch die die disjunkte horizontale Teilung vorgenommen wird. Die genaue Aufteilung muss, wie in den Annahmen bereits deutlich gemacht wurde, nicht weiter bekannt sein. Man kann aber mit Bestimmtheit sagen, dass sie das Ergebnis eines unbekannten Algorithmus ist, der diese horizontalen Splits festlegt hat. Die vertikale Aufteilung muss wie schon erwähnt nicht zwingend vollständig disjunkt sein, für unseren Operator aber schon. Sie wird zwar mit dem Hintergrund gewählt, was miteinander korreliert und was nicht, lässt sich im Prinzip aber frei bestimmen. Die genaue Aufteilung wird vorgegeben durch die Parameter des Operators. Auf formaler Ebene bedeutet das:

$$R = V_1 \cup ... \cup V_m$$

$$V_i \cap V_j = \{\} \quad \text{für} \quad \forall i, \forall j : \; i, j \in \{1, ..., m\} \quad \text{und} \quad i \neq j$$

$$V = \{V_1, ..., V_m\}$$

Der spezifische Slicing-Prozess ist eine vielschrittige Anwendung einzelner Operatoren auf diverse gebildete Teilrelationen. Der genaue Prozess ist sowohl schrittweise formalisiert als auch gebündelt in Form von Abbildung 22 zu finden. Da wir uns innerhalb der gängigen relationalen Algebra bewegen und etwas tun müssen, was nicht regulär dazugehört, sollte zum Verständnis auf jeden Fall ein bestimmter Operator erklärt werden.

τ-Operator: $\tau_L(r)$

Der Operator τ setzt dabei das Konzept der Sortierung um. Sei r im Folgenden eine Relation und L eine Liste von Attributen aus r. Durch Anwendung von $\tau_L(r)$ wird r lexikographisch nach L sortiert. Da eine Relation hierbei strenggenommen in eine Liste überführt werden muss, ist es in der erweiterten relationalen Algebra zumeist die letzte Operation, die man ausführt. In unserem Falle aber nicht, da es einzig bei der Neuordnung von Einträgen helfen soll. Nach anschließender Zusammenführung aller Teilrelationen ist die Reihenfolge wieder belanglos.

Kommen wir nun zu den einzelnen Schritten. Der erste Schritt ist die horizontale Aufteilung der Basisrelation. Im anschließenden Schritt wird die vertikale Aufteilung vorgenommen. Im dritten Schritt erfolgt dann die Permutation. Im vierten und fünften Schritt geschieht die erneute Zusammenführung, zuerst vertikal, dann horizontal.

Schritt 1 :	$\forall i \in \{1,...,n\}$	$r_{1\langle i\rangle}$	:=	$\sigma_{H_i}(r)$
Schritt 2 :	$\forall i \in \{1,...,n\}, \forall j \in \{1,...,m\}$	$r_{2\langle i,j\rangle}$	:=	$\pi_{V_j}(r_{1\langle i\rangle})$
Schritt 3 :	$\forall i \in \{1,...,n\}, \forall j \in \{1,...,m\}$	$r_{3\langle i,j\rangle}$	:=	$\tau_{P_j}(r_{2\langle i,j\rangle})$
Schritt 4 :	$\forall i \in \{1,...,n\}$	$r_{4\langle i\rangle}$	:=	$r_{3\langle i,1\rangle} \bowtie_{ID} ... \bowtie_{ID} r_{3\langle i,m\rangle}$
Schritt 5 :	$\forall i \in \{1,...,n\}$	r_5	:=	$\bigcup_{i=1}^{n}(r_{4\langle i\rangle})$

Abbildung 22, von unten nach oben zu lesen, veranschaulicht den Prozess noch etwas kompakter. Da sich die ID-Vergabe allgemein schlecht einbinden ließ, sei dessen Vergabe und Ausblendung mal grundsätzlich außen vor gelassen.

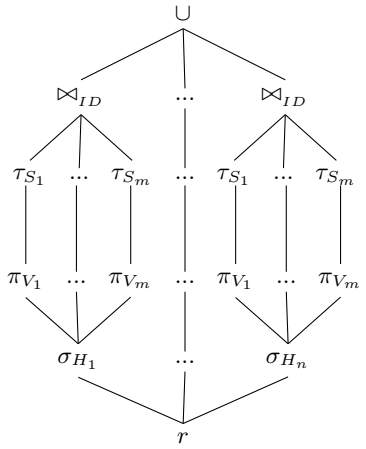


Abbildung 22: α^S-Operator

5.4 Beispiel

Zum genauen Verständnis des α^S-Operators sei dessen Wirkungsweise am besten an einem Beispiel erklärt. Eine kurze inhaltliche Einordnung des Ganzen ist in Abschnitt 3.2 zu finden.

Die Ausgangsrelation r setzt sich aus dem Schema $R = \{A, B, C, D\}$ zusammen. Ebendiese gilt es mittels Slicing nun zu anonymisieren. Die vorliegenden Attribute sollen in drei Korrelationsgruppen aufgeteilt werden: $V_1 = \{A\}$, $V_2 = \{B, C\}$ und $V_3 = \{D\}$. Aus all dem ergibt sich dann folgender Ausdruck:

$$r_a := \alpha^S{}_{V=\{\{A\},\{B,C\},\{D\}\}}(r)$$

Abbildung 23 zeigt an einem exemplarischen Datenbestand, wie ein Resultat aussehen könnte, da ja verschiedene Permutationen möglich sind. Die interne horizontale Splitgröße sei im Folgenden 3 gewesen.

A	B	C	D
01.01.	00	+	18
05.01.	00	+	24
31.01.	B0	−	32
20.03.	A0	+	36
25.03.	A0	+	42
31.03.	B0	−	56

$\Longrightarrow$

A	B	C	D
31.01.	00	+	18
05.01.	00	+	32
01.01.	B0	−	24
20.03.	A0	+	36
25.03.	B0	−	42
31.03.	A0	+	56

Abbildung 23: links r, rechts r_a

5.5 Regeln

1. ObsoleteSlice: Obsoletes Slicing

   Diese Regel ist zwar trivial und wird aufgrund aller Annahmen im Weiteren ausgeschlossen, aber sei auf Algebra-Ebene nichtsdestotrotz einmal angemerkt.

   $$\text{(S1)} \quad \alpha^S{}_{V=\{X_1 \ldots X_n\}}(r_1) \iff r_1 \text{ falls } n = 1$$

   Faktisch gesehen ist $X_1 = R_1$. Es existiert also nur eine einzige vertikale Gruppe und alle Einträge würden somit ganz bleiben. Da die nebenbei anfallende Sortierung keine Rolle spielt, ändert sich an der eigentlichen Relation im Grunde nichts. Abbildung 24 verdeutlicht dies noch einmal. Im erweiterten Sinne gilt diese Regel auch bzw. man würde ebenso bei ihr landen, wenn man leere Mengen und nicht in R_1 vorkommende Attribute zuließe und diese dann aus den Korrelationsmengen abziehen würde.

   Bei V={{A,B}} ergeben sich folgende Relationen:

A	B
05.01.	00
01.01.	00
31.01.	B0
20.03.	A0
31.03.	B0
25.03.	A0

$\Longleftarrow$

A	B
01.01.	00
05.01.	00
31.01.	B0
20.03.	A0
25.03.	A0
31.03.	B0

Abbildung 24: Obsoletheit des Slicings bei gesamtheitlicher Korrelationsgruppe

2. IdemSlice: Idempotentes Verhalten des Slicings und generelle Mehrfachanwendung

   Slicing an sich ist quasi idempotent. Natürlich müssen dafür gleiche Korrelationsgruppen vorliegen. Da die Relationsgröße gleich bliebe, würden an gleichen Stellen die horizontalen Splits gesetzt werden. Die $n \cdot m$ Permutationsgruppen sind bezüglich der abgegrenzten

Werte somit gleich. Die vorherige Permutation spielt in diesem Falle dann keine Rolle mehr. Ein einziger Slicingvorgang ist daher genau so gut wie mehrere.

$$\text{(S2)}\quad {\alpha^S}_{V=\{X_1 \ldots X_n\}}({\alpha^S}_{V=\{X_1 \ldots X_n\}}(r_1)) \iff {\alpha^S}_{V=\{X_1 \ldots X_n\}}(r_1)$$

Abbildung 25 veranschaulicht, wie das erste Slicing bei mehrfacher Anwendung im Grunde obsolet ist. Die permutierte Konstellation ist zwar eine andere, doch alle Ausgänge der ersten Instanz sind ebenso wahrscheinlich in der zweiten. Das Resultat des zweiten Slicings hätte daher ebenso gut beim ersten produziert werden können.

Bei V={{A},{B}} ergeben sich folgende Relationen:

A	B
31.01.	00
05.01.	00
01.01.	B0
20.03.	A0
25.03.	B0
31.03.	A0

$\Longleftarrow$

A	B
01.01.	B0
05.01.	00
31.01.	00
31.03.	A0
25.03.	B0
20.03.	A0

$\Longleftarrow$

A	B
01.01.	00
05.01.	00
31.01.	B0
20.03.	A0
25.03.	A0
31.03.	B0

$\Longrightarrow$

A	B
31.01.	00
05.01.	00
01.01.	B0
20.03.	A0
25.03.	B0
31.03.	A0

Abbildung 25: Idempotenz des Slicings

Eine Mehrfachanwendung ließe sich aber nicht nur bei gleichen Korrelationseinteilungen vereinfachen. Tatsächlich kann man dies sogar immer tun. Sofern man außer Acht lässt, dass dem Ganzen eine bewusste Einteilung zwischen Korrelierendem und Nichtkorrelierendem zugrunde liegt, ließe sich bei zwei verschiedenen V-Einteilungen ebenso eine Vereinfachung unternehmen, wenn man ein drittes V bildet Da gemeinhin $R_1 = X_1 \cup ... \cup X_n = Y_1 \cup ... \cup Y_m$ mit vollständiger Disjunktheit innerhalb einer jeden Aufteilung gilt, ließe sich wie folgt vereinfachen:

$$\text{(S3)}\quad {\alpha^S}_{V_1=\{X_1 \ldots X_n\}}({\alpha^S}_{V_2=\{Y_1 \ldots Y_m\}}(r_1)) \iff {\alpha^S}_{V}(r_1)$$

$$\text{mit}\quad V = \{\ x\ \mid\ \forall A \in V_1, \forall B \in V_2 :\ x = A \cap B\ \wedge\ x \neq \{\}\ \}$$

Im Grunde müssen die korrelierenden Beziehungen verschärft beziehungsweise Scheinbeziehungen gesamtheitlich aufgebrochen werden. Im Folgenden zwei Beispiele: das eine rein auf Attributebene, rein rechnerisch gehalten und ein wenig komplexer, das andere schlichter, aber dafür mit Abbildungen zur Veranschaulichung der Vereinfachung. Da Mengen von Mengen zum Einsatz kommen, die reguläre Schreibweise relativ viel Platz verbrauchen würde und die Attributnamen stets einbuchstabig sind, werden die inneren Mengen im Weiteren auf folgende Weise verkürzt:

$$\{\{A,B\},\{C,D,E\},\{F,G\}\} \iff \{AB,CDE,FG\}$$

Beispiel 1: Bei V_1={AB,CDE,FG} und V_2={ABC,DE,FG} ergibt sich eine übergeordnete Einteilung nach der Form V={AB,C,DE,FG}. Die Nebenrechnung sieht hierbei wie folgt aus:

A	∩	B	=	x
AB	∩	ABC	=	AB
AB	∩	DE	=	∅
AB	∩	FG	=	∅
CDE	∩	ABC	=	C
CDE	∩	DE	=	DE
CDE	∩	FG	=	∅
FG	∩	ABC	=	∅
FG	∩	DE	=	∅
FG	∩	FG	=	FG

Tabelle 9: Teilergebnisse des ersten Beispiels

Beispiel 2: Abbildung 26 veranschaulicht, wie die Anwendung verschiedener Einteilungen das Korrelationsgefüge beeinflussen. Die Beziehung zwischen C und D wird schon in erster Instanz aufgelöst und sie bleibt es auch, nur wird die genaue Permutation zwischen den beiden in zweiter Instanz nicht weiter verändert. B und C bleiben augenscheinlich zuerst noch verbunden, bis deren Beziehung in zweiter Instanz aufgebrochen wird. Das Verhältnis zwischen B und D wird grundsätzlich und in doppelter Instanz aufgebrochen. Anstelle doppelten Slicings wäre dieser Ausgang ebenso wahrscheinlich, wenn man direkt auf Basis von V permutieren würde.

Bei V_1={{B,C},{D}}, V_2={{B},{C,D}} und daraus folgend V={{B},{C},{D}} ergeben sich folgende Relationen:

B	C	D
B0	+	18
00	+	32
00	−	24
A0	+	56
A0	−	42
B0	+	36

⟸

B	C	D
00	+	18
B0	−	24
00	+	32
A0	+	56
A0	+	36
B0	−	42

⟸

B	C	D
00	+	18
00	+	24
B0	−	32
A0	+	36
A0	+	42
B0	−	56

⟹

B	C	D
B0	+	18
00	+	32
00	−	24
A0	+	56
A0	−	42
B0	+	36

Abbildung 26: Zusammenfassbarkeit von Mehrfachslicing, zweites Beispiel

3. SliceProj: Slicing und Projektion

Zwischen Projektion und Slicing kann problemlos getauscht werden. Aus formaler Sicht wird im Folgenden zwischen beiden Tauschrichtungen unterschieden. Zum Vorziehen des Slicens müssen die Korrelationsmengen um die wegprojizierten Attribute ergänzt werden. Man könnte die wegzuprojizierenden Attribute zwar beliebig zerstreuen bzw. irgendwo integrieren, da sie am Ende ohnehin wieder ausgeblendet werden, doch da dies nichts weiter bezwecken würde, ist es dahingehend am sinnvollsten, dass man für diese einfach eine gesonderte Korrelationsgruppe einführt. Falls das Slicing auf Basis einer bereits pro-

jizierten Menge geschieht, in diesem Falle also $Y \subseteq R_1$, $Y = X_1 \cup ... \cup X_n$ gilt, ließe sich folgendermaßen umformen:

$$\text{(S4)} \quad \alpha^S{}_{V_X=\{X_1 \ldots X_n\}}(\pi_Y(r_1)) \Longrightarrow \pi_Y(\alpha^S{}_{V_{X'}=\{X_1 \ldots X_n, X_{n+1}\}}(r_1)) \text{ mit } X_{n+1} = R_1 \setminus Y$$

Abbildung 27 verdeutlicht im Einzelnen, wie die wegfallenden Projektionskandidaten C und D eine zusätzliche Korrelationsgruppe bilden, damit man ebenso auf der Basisrelation slicen kann.

Bei R_1={A,B,C,D}, Y={A,B}, V_X={{A},{B}} und folglich $V_{X'}$={{A},{B},{C,D}} ergeben sich folgende Relationen:

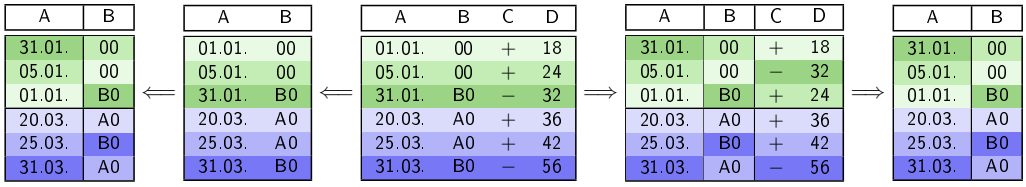


Abbildung 27: Vorziehbarkeit des Slicings bei Projektion

Auch der umgekehrte Fall, das Vorziehen der Projektion, ist möglich. Alle wegzuprojizierenden Attribute müssen hierfür aus den Korrelationsmengen gestrichen werden. Sollte dabei mal eine leere Menge ergeben, kann diese entsprechend ignoriert werden. Da hier gemeinhin $Z \subseteq R_1$, $Z = X_1 \cup ... \cup X_n$, $R_1 = Y_1 \cup ... \cup Y_m$ gilt, ließe sich wie folgt umformen:

$$\text{(S5)} \quad \alpha^S{}_{V_X=\{X_1 \ldots X_n\}}(\pi_Z(r_1)) \Longleftarrow \pi_Z(\alpha^S{}_{V_Y=\{Y_1 \ldots Y_m\}}(r_1))$$

$$\text{mit} \quad V_X = \{\ x \ \mid\ \forall A \in V_Y : \ x = A \setminus (R_1 \setminus Z) \ \wedge \ x \neq \{\}\ \}$$

Abbildung 28 verdeutlicht, dass das Resultat ebenso äquivalent ist, wenn man die Korrelationsmengen auf die Projektionsmenge angepasst hat. Bei der Gruppe, die sich aus B und C zusammensetzt, wird aufgrund der Projektion das C herausgestrichen und jene, die einzig ein C enthält, fällt gänzlich heraus.

Bei R_1={A,B,C,D}, Z={A,B}, V_Y={{A},{B,C},{D}} und folglich V_X={{A},{B}} ergeben sich folgende Relationen:

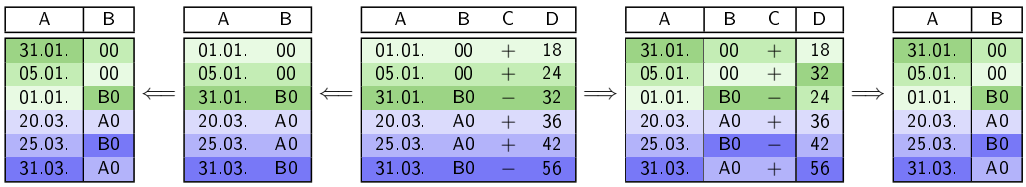


Abbildung 28: Vorziehbarkeit der Projektion bei Slicing

Des Weiteren ergäbe sich noch ein Spezialfall, der ebenso in Richtung obsoletes Slicing ginge. Sofern die letztendliche Projektion auf nur einer der Korrelationsmengen stattfinden würde, kann man sich das Slicing eigentlich auch sparen.

$$\text{(S6)}\quad \pi_Y(r_1) \Longleftarrow \pi_Y({\alpha^S}_{V=\{X_1 \ldots X_n\}}(r_1)) \text{ falls } X_i \in V \text{ mit } Y \subseteq X_i$$

Die Abbildung 29 verbildlicht die Überflüssigkeit des Permutierens von B und C, sobald das nicht korrelierende A ausgeblendet wird.

Bei V={{A},{B,C}}, Y={B,C} ergeben sich folgende Relationen:

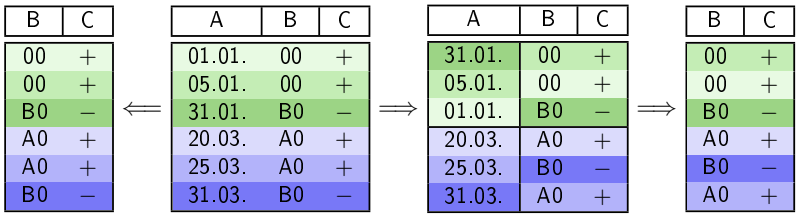


Abbildung 29: Obsoletes Slicing bei Projektion auf einer der Korrelationsmengen

4. SliceSel: Slicing und Selektion

Selektion und Slicing lassen sich hinsichtlich eines Tausches grundsätzlich nicht miteinander vereinbaren. Selektieren und anschließendes Slicen ist natürlich ein durchaus vorstellbarer Anwendungsfall. Der umgekehrte Fall dagegen nicht. Ein spätes Selektieren würde nämlich für eine inkonsistente Gesamtverteilungen sorgen, da die korrelierenden Teile eines Eintrages schon beliebig verstreut wurden. Was auch immer man löschen möchte, zieht höchstwahrscheinlich anderen Daten in Mitleidenschaft. Je mehr man verhältnismäßig löschen würde, desto unbrauchbarer würde eine Analyse auf diesen Daten werden. Sofern man dabei auf Attributen nur einer Korrelationsgruppe selektiert, fielen zumindest immer die gewünschten Werte und somit gleich viele Einträge heraus. Würde man dagegen auf mehreren Gruppen selektieren, unterliegt das Ganze obendrein der Permutation, da manche Selektionskandidaten zerstört werden und andere durch diese erst entstehen sind.

$$\text{(S7)}\quad {\alpha^S}_{V=\{X_1 \ldots X_n\}}(\sigma_F(r_1)) \not\Longleftrightarrow \sigma_F({\alpha^S}_{V=\{X_1 \ldots X_n\}}(r_1))$$

In Abbildung 30 wird genau dieses Kernproblem noch einmal verdeutlicht. Beim linksseitigen Weg konnte ordnungsgemäß selektiert und geslict werden. Die rechtsseitige Verteilung ist dagegen ein Produkt von Zufall und Inkonsistenz.

Bei V={{A},{B}}, F: B≠B0 und einer Splitgröße von mindestens 3 ergeben sich folgende Relationen:

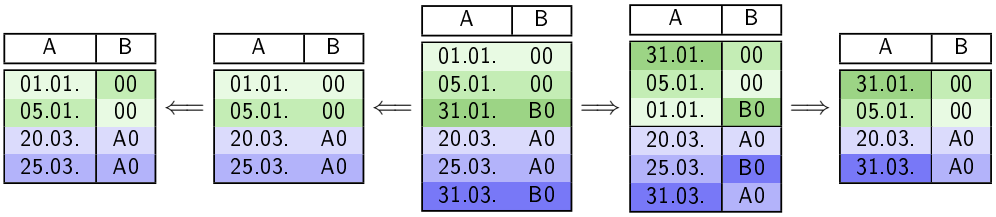


Abbildung 30: Slicing und Selektion

5. SliceUnion: Slicing und die Vereinigung

Eine Vertauschung ist zwar nicht zwingend äquivalent, aber im Sinne der Anonymität dennoch konform. Slicing auf getrennten Relationen ist so, als hätte man eigenmächtig über einige horizontale Split-Stellen verfügt. Streng genommen ist das Ganze nicht ergebnisäquivalent, da nur entfernt benachbarte Einträge beim Permutieren aufeinandertreffen können. Einerseits können einige Permutationskonstellationen plötzlich nicht mehr auftreten, andererseits entstehen völlig neue. Rein praktisch ist diese Vertauschung absolut legitim, da Slicing im Grunde nur den gesamtzeiligen Einträge abschaffen möchte, sodass zeilenartige Wirkung nur noch auf festgelegten Bereichen möglich ist. Nicht zuletzt sind einem die zugrunde liegenden Permutationsgruppen gemeinhin unbekannt. Von daher ist diese Regel im Sinne von Parallelisierbarkeit auf horizontalen Partitionen durchaus zweckhaft.

$$\text{(S8)}\quad \alpha^S{}_{V=\{X_1 \ldots X_n\}}(r_1 \cup r_2) \overset{\sqsubseteq}{\Longleftrightarrow} \alpha^S{}_{V=\{X_1 \ldots X_n\}}(r_1) \cup \alpha^S{}_{V=\{X_1 \ldots X_n\}}(r_2)$$

Abbildung 31 verdeutlicht die prinzipiell mögliche Äquivalenz, falls die Partitionierung an einer Stelle liegt, an der auch eine Splitgrenze fallen würde.

Bei V={{A},{B}} und einer minimalen Splitgröße von 3 ergeben sich folgende Relationen:

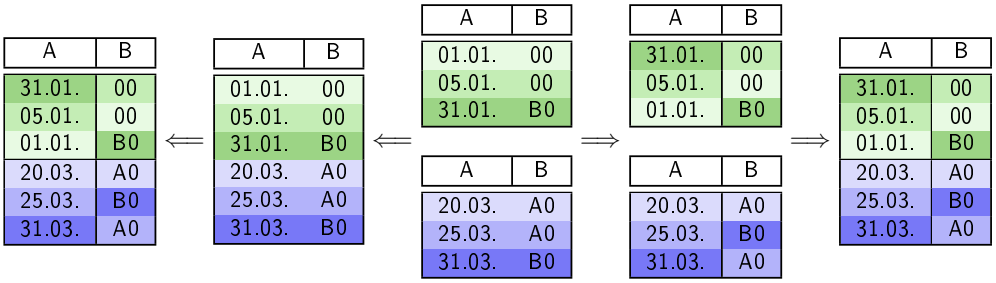


Abbildung 31: Slicing und Vereinigung bei gleichen Split-Grenzen

Abbildung 32 verdeutlicht den regulären Fall, dass die Partitionierung für andere Splitgrenzen sorgt und daraus grundsätzlich unterschiedliche Permutationsfälle resultieren.

Bei V={{A},{B}} und einer minimalen Splitgröße von 3 ergeben sich folgende Relationen:

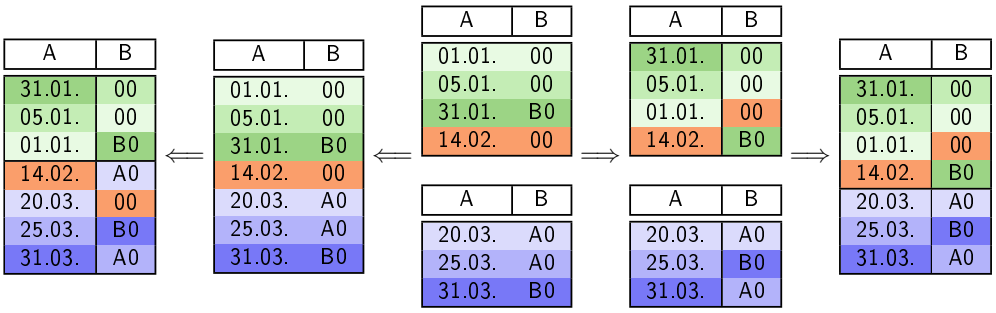


Abbildung 32: Slicing und Vereinigung bei unterschiedlichen Split-Grenzen

6. SliceJoin: Slicing und Verbund

Verbund und Slicing lassen prinzipiell sich nicht ohne Weiteres vertauschen. Zum einen ändert ein vorgezogenes Slicing die zugrundeliegenden Verbundkonstellationen. Und selbst, wenn diese unangetastet blieben, würde ein Permutieren der Nicht-Verbundattribute dafür sorgen, dass völlig andere Werte verknüpft und vervielfältigt werden. Was aber trotz all dieser Einschränkungen noch möglich bliebe, wäre ein Tausch bei vertikaler Partitionierung. Korrelierende Attribute können verständlicherweise nicht relationsübergreifend sein und alle IDs müssen in beiden Relationen deckungsgleich vorkommen, da sonst Werte von unabgedeckten Einträgen einfließen würden. Die ID sollte man irgendwann nach dem Verbund selbstverständlich wegprojizieren. Alle Attribute, die in einer Korrelationsgruppe mit dieser wären, wären grundsätzlich gefährdet und sogar, wenn die ID in einer gesonderten Korrelationsgruppe stecken würde, kann diese immer noch ein Risiko darstellen. Wenn bei großen Datenbeständen ein tendenziell aufstrebender Wertecharakter erkennbar ist, ließe sich etwa die allgemeine Splitgröße vermuten. Alle Werte vor einer Splitgruppe müssten dabei kleiner sein als die einer jeden Gruppe und alle danach größer. Auf Basis der vermuteten Gruppe, in welcher eine ID liegen könnte, könnte man dann die üblichen Angriffe der k-Anonymität durchführen.

Zumindest bei vertikaler Partitionierung gilt:

$$\text{(S9)}\quad \alpha^S{}_{V=\{X_1 \ldots X_i X_j \ldots X_n\}}(\pi_R(r_1 \bowtie r_2)) \iff \pi_R(\alpha^S{}_{V_1=\{X_1 \ldots X_i ID\}}(r_1) \bowtie \alpha^S{}_{V_2=\{X_j \ldots X_n ID\}}(r_2))$$

$$\text{falls}\quad X_1 \cup \ldots \cup X_i \cup ID = R_1,\ X_j \cup \ldots \cup X_n \cup ID = R_2,\ \text{Verbund mittels ID},\ R = (R_1 \cup R_2) \setminus ID$$

Abbildung 33 verdeutlicht die Äquivalenz der Regel unter all den zuvor beschriebenen Umständen.

Bei V_1={{A},{ID}}, V_2={{ID},{B}} ergeben sich folgende Relationen:

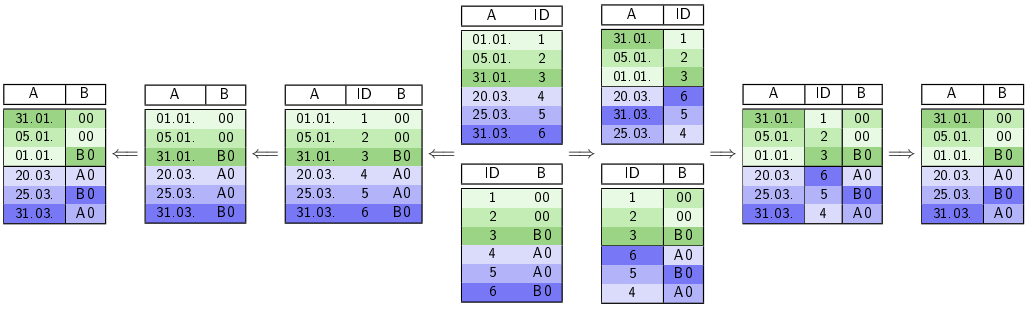


Abbildung 33: Slicing und Verbund bei vertikaler Partitionierung

7. SliceInter, SliceDiff: Slicing bei Schnittmenge und Differenzmenge

Aufgrund der Willkür der Permutation sind Schnitt- wie auch Differenzmengenbildung nicht verschiebbar. Alle Einträge, die sich bei vorgezogener Permutation beidseitig vorkommen würden bzw. sich bei einer der beiden abziehen ließen, wären ein reines Zufallsprodukt. Wie sollte es auch anders seien? Schnitt- und Differenzmenge fordern für beide Relationen Zeilenintegrität und Slicing soll genau dies zunichtemachen.

$$\text{(S10)}\quad \alpha^S{}_{V=\{X_1\ldots X_n\}}(r_1 \cap r_2) \not\Longleftrightarrow \alpha^S{}_{V=\{X_1\ldots X_n\}}(r_1) \cap \alpha^S{}_{V=\{X_1\ldots X_n\}}(r_2)$$
$$\text{(S11)}\quad \alpha^S{}_{V=\{X_1\ldots X_n\}}(r_1 - r_2) \not\Longleftrightarrow \alpha^S{}_{V=\{X_1\ldots X_n\}}(r_1) - \alpha^S{}_{V=\{X_1\ldots X_n\}}(r_2)$$

Die Abbildung 34 verdeutlicht noch einmal das Kernproblem am Beispiel der Schnittmenge. Linksseitig wird zuerst auf den intakten Relationen die Mengenoperation ausgeführt und dieses stets eindeutige Ergebnis dann permutiert. Was dagegen auf rechtsseitigem Wege herauskommt, ist unberechenbar – sowohl werteseitig als auch hinsichtlich der bloßen Anzahl der Einträge.

Bei V={{A},{B}} ergeben sich veranschaulicht an der Schnittmenge folgende Relationen:

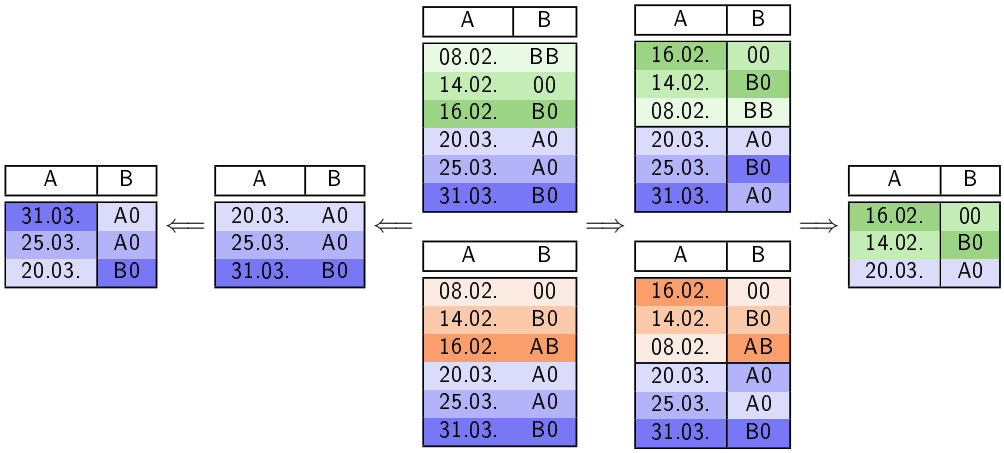


Abbildung 34: Slicing und Schnittmenge

6 Differential-Privacy-Operator α^D

Dieses Kapitel widmet sich der Operationalisierung von einfacher Verrauschung. Der eingeführte Operator wird hinreichend erklärt und anschließend dessen Anwendung ergründet. Es wurde kurz nach obsoletem Gebrauch geschaut, ebenso nach Mehrfachanwendung und ausgiebig die Beziehung zu Projektion, Selektion, Vereinigung, Verknüpfung, Schnitt- und Differenzmenge untersucht.

6.1 Aufbau

Der Differential-Privacy-Operator setzt sich wie folgt zusammen:

$$\alpha^D{}_{P=\{(A_1,p_1),\ldots,(A_n,p_n)\}}(r)$$

α^D	Bezeichner des Operators α steht für Anonymisierung D für das genaue Verfahren (Differential Privacy)
P	die Menge der zu verrauschenden Attribute
A_i	ein Attribut, $A_i \in R$
p_i	die Wahrscheinlichkeit, dass A_i verrauscht wird; $0 \leq p_i \leq 1$
r	die Basisrelation

Weitere formelle Beschreibungen folgen im anschließenden Abschnitt, da zuerst alle getroffenen Einschränkungen erklärt werden sollten.

6.2 Annahmen und Einschränkungen

Ein Verrauschungsoperator kann auf verschiedene Ansätze und Techniken ausgerichtet sein, sodass ein jeder über ziemlich spezielle Regeln verfügt. Da eine derart ausführliche Untersuchung aber nicht möglich ist, soll sie sich auf einen relativ allgemeinen Fall beschränken.

Grundsätzlich sind bei Verrauschung zweierlei Ansätze denkbar: einerseits das Randomized-Response-Konzept, andererseits das ε-DP-Prinzip. Der erste Ansatz erfordert die Einführung einer allgemeinen Verrauschungswahrscheinlichkeit, mit der die Frage nach der Beeinflussung einer Zeile stets und unabhängig aufs Neue gestellt wird. Der zweite Ansatz gibt dagegen vor, wie sehr die veränderte Häufigkeit von Werten aus relativer Sicht maximal abweichen darf. Diese Methode hängt von der Gesamtheit der Einträge ab, ist also nicht zeilenunabhängig. Die Minimalform dieses Ansatzes würde sich auf ε beschränken, in der erweiterten Form wäre zusätzlich δ bedacht. Aus formaler Sicht ergeben sich drei verschiedene Ausführungen:

$$\alpha^D{}_{P=\{(A_1,p_1),\ldots,(A_n,p_n)\}}(r)$$

$$\alpha^D{}_{E=\{(A_1,\varepsilon_1,\delta_1),\ldots,(A_n,\varepsilon_n,\delta_n)\}}(r)$$

$$\alpha^D{}_{E'=\{(A_1,\varepsilon_1),\ldots,(A_n,\varepsilon_n)\}}(r) = \alpha^D{}_{E=\{(A_1,\varepsilon_1,\delta_1=0),\ldots,(A_n,\varepsilon_n,\delta_n=0)\}}(r)$$

Da der zeilenunabhängige Ansatz aus Regelsicht mehr Potential bietet als die anderen Ansätze, wird sich alle weitere Untersuchung auf ihn beschränken.

Während zeilenbezogen die allgemeine Frage ist, ob nun verrauscht wird oder nicht, geht es spaltenbezogen vorrangig um die Frage nach dem Wie, also der wertespezifischen Veränderung im entsprechenden Fall. Verteilungen müssen etwa nicht zwangsweise auf ein einziges Attribut beschränkt sein, sondern können sich ebenso gut auf mehrere Attribute beziehen. Es wäre auch möglich, dass die Verrauschung wertebedingt ist – zum einen durch die echten bzw. Vorgängerwerte, zum anderen durch Werte von unbetroffenen Attributen. Nicht zuletzt würde dies noch weitere Fragen mit sich bringen. Wie sähe es dann beispielsweise mit der zeitlichen Beeinflussung aus? Würde nur auf Basis des Ausgangszustandes verrauscht werden oder dynamisch? Da eine gesonderte Untersuchung viel zu sehr in die Tiefe ginge, wird auch hier nur der allgemeinste Ansatz weiterverfolgt. Es sei daher festgelegt, dass Attribute ausschließlich einzeln und unabhängig voneinander verrauscht werden.

Des Weiteren gäbe es noch einige triviale und weniger bedeutsame Dinge, die der Klarheit und Regeln wegen angesprochen werden sollten. Es sei sichergestellt, dass jedes Attribut, das man verrauschen möchte, sowohl vorkommt als auch eine entsprechende Verrauschungsverteilung besitzt. Alle Attribute tauchen in der Verrauschungsmenge nur einmalig auf, da sonst unklar wäre, auf Grundlage welches Wahrscheinlichkeitswertes man überhaupt verrauschen sollte. Attribute, die nicht in der Verrauschungsmenge vorkommen, werden wahrheitsgemäß belassen bzw. so behandelt, als sei $p_i = 0$. Im Sinne der Optimierungsregeln sei außerdem festgelegt, dass $p_i > 0$ gilt, also auch nur Attribute mit echter Verrauschungsabsicht aufgeführt werden.

6.3 Ausführliche Formalisierung

Sei $p_i \in [0, 1]$ die Wahrscheinlichkeit, mit der ein Wert a_{ij} zufällig verrauscht werden soll. Im Umkehrschluss meint die Gegenwahrscheinlichkeit $1 - p_i$ Wertetreuheit, also keine Veränderung. Eine bestimmte Verrauschungswahrscheinlichkeit p_i gilt im Übrigen für alle Werte eines Attributes A_i. Im Falle einer Verrauschung existiert immer eine eindeutige Verteilung. Sie hängt auf jeden Fall vom Attribut am, unter Umständen auch noch vom Ausgangswert selbst. Eine statische Verteilung sei eine Verteilung, die für alle a_{ij}-Werte eines Attributes A_i gleich aussieht. Eine dynamische Verteilung sei dagegen zusätzlich vom jeweiligen Ausgangswert a_{ij} abhängig. b_i ist im Folgenden ein boolescher Wahrheitswert, bei welchem dynamisch *wahr* und statisch *falsch* symbolisiert. Abbildung 35 verbildlicht noch einmal abschließend die namentliche Attributeinteilung der zu verzerrenden Ausgangswerte.

A_1	...	A_i	...	A_n
a_{11}	...	a_{i1}	...	a_{n1}
...	...	...	...	...
a_{1j}	...	a_{ij}	...	a_{nj}
...	...	...	...	...
a_{1m}	...	a_{im}	...	a_{nm}

Abbildung 35: Unverrauschte Ausgangswerte und deren Attributzuordnung

Der Algorithmus 1, wenngleich nur Pseudocode, verdeutlicht, wie alle vorgestellten Faktoren im zufallsgesteuerten Entscheidungsprozess agieren.

Algorithmus 1: Randomisierter Mechanismus

Data: Verteilungsname A_i, Ausgangswert a_{ij}, Verrauschungswahrscheinlichkeit p_i, dynamische Verzerrung b_i (Wahrheitswert)
Result: Wert, der vielleicht verrauscht wurde, vielleicht aber auch unverändert ist
randomized_Response(A_i,a_{ij},p_i,b_i);
if *random()* $\leq 1 - p_i$ **then**
  return a_{ij};
else
  if $b_i = false$ **then**
    return statische_Verteilung(A_i, random());
  else
    return dynamische_Verteilung(A_i, a_{ij}, random());
  end
end

Zu den Verteilungen lässt sich eigentlich nichts weiter konkretisieren. Sie sind für den Operator ohnehin nicht von Relevanz, doch nichtsdestotrotz sei zwischen den möglichen Unterteilungen unterschieden. Obwohl wir uns darauf beschränkt haben, dass wir nur klar von einander abgegrenzte Einzelattribute erlauben, ist ihrem Verhalten keine Grenzen gesetzt: sie könnten diskret oder stetig sein – endlich, abzählbar unendlich oder überabzählbar – durch einen Algorithmus funktional beschreibbar oder auch nicht. Daher ist nicht viel mehr möglich, als zu beiden Formen ein exemplarisches Beispiel anzuführen.

Statische Verteilung

Eine statische Verteilung gilt wie gesagt für alle konkreten Werte eines Attributes. Ein einfaches Beispiel wäre eine endliche Verteilung, die alle Tage eines Jahres unabhängig vom Jahr selbst abbildet. Wäre nicht der Schaltjahr-Tag zu berücksichtigen, wäre es sogar eine Gleichverteilung. Abbildung 36 verbildlicht das Ganze noch einmal. Natürlich haben statische Verteilungen so ihre Grenzen. Sie sind so gesehen ja eigentlich ein Spezialfall. Würde man zusätzlich das Jahr mit einfließen lassen und einen konkreten Tag auf dessen jeweiliges Jahr verteilen wollen, müsste man nämlich zu einer dynamischen Verteilung greifen.

Wahr-scheinlichkeit	Wert
4/1461	01.01.
4/1461	02.01.
4/1461	03.01.
...	...
4/1461	28.02.
1/1461	29.02.
4/1461	01.03.
...	...
4/1461	31.12.

Abbildung 36: Statische Verteilung

Dynamische Verteilung

Ein Beispiel für eine dynamische Verteilung sei eine, die einen beliebigen Tag mit hoher Wahrscheinlichkeit zu einem seiner umliegenden Nachbartage verrauscht. Für die Abbildung 37 sei im Konkreten der 08.03. gewählt. Es sei noch einmal betont, dass dieses Beispiel nur einen Bruchteil der denkbaren Möglichkeiten zeigt, denn viele dynamische Verhalten lassen sich nicht algorithmisch, sondern nur händisch abbilden.

Wahr-scheinlichkeit	Wert
...	...
0.05	$x-3$
0.15	$x-2$
0.20	$x-1$
0.05	x
0.20	$x+1$
0.15	$x+2$
0.05	$x+3$
...	...

$\Longrightarrow$

Wahr-scheinlichkeit	Wert
...	...
0.05	05.03.
0.15	06.03.
0.20	07.03.
0.05	08.03.
0.20	09.03.
0.15	10.03.
0.05	11.03.
...	...

Abbildung 37: Dynamische Verteilung

6.4 Beispiel

Zum genauen Verständnis des α^D-Operators sei dessen Wirkungsweise an einem Beispiel erklärt. Eine kurze inhaltliche Einordnung ist in Abschnitt 3.2 zu finden.

Die Ausgangsrelation r setzt sich aus dem Schema $R = \{A, B, C, D\}$ zusammen. Ebendiese gilt es mittels einfacher Verrauschung nun zu anonymisieren. Attribut A soll mit einer Wahrscheinlichkeit von 0.16 verrauscht werden, Attribut B mit einer Wahrscheinlichkeit von 0.33, Attribut C mit einer Wahrscheinlichkeit von 0.5 und Attribut D soll unverrauscht bleiben. Aus all dem ergibt sich folgender Ausdruck:

$$r_a := \alpha^D{}_{P=\{(A,0.16),(B,0.33),(C,0.5)\}}(r)$$

Die Verrauschungsverteilungen von B und C sind in Abbildung 38 zu finden und beide statisch. Die Verrauschungsverteilungen von A sei dynamisch und verhält sich wie jene, die in Abbildung 37 ersichtlich ist. Sollte ein Wert aus A verrauscht werden, erfolgt also höchstwahrscheinlich eine Verzerrung zu einem Tag, der nicht weit vom ursprünglichen entfernt ist.

B	p(b)
00	0.410
A0	0.358
AA	0.078
B0	0.102
AB	0.045
BB	0.007

C	p(c)
+	0.85
−	0.15

Abbildung 38: Verteilungen für den Verrauschungsfall

Wie sich eine Anwendung des Operators auswirken könnte, zeigt Abbildung 39. Alle veränderten Werte sind im Folgenden rot hervorgehoben. Die Ungewissheit des Zufalls erlaubt es, dass einerseits alles, andererseits nichts verändert werden könnte. Ob tendenziell viele oder wenige Werte verzerrt werden, ist abhängig von der jeweiligen Verrauschungswahrscheinlichkeit. Damit das aktuelle Beispiel und die noch folgenden auch wirklich ein besseres Verständnis begünstigen, spiegelt sich die Verrauschungswahrscheinlichkeit im Allgemeinen in der Anzahl der verrauschten Werte wider. Auf unwahrscheinliche Fälle wird also verzichtet.

A	B	C	D
01.01.	00	+	18
05.01.	00	+	24
31.01.	B0	−	32
20.03.	A0	+	36
25.03.	A0	+	42
31.03.	B0	−	56

$\Longrightarrow$

A	B	C	D
28.12.	00	+	18
05.01.	A0	+	24
31.01.	B0	+	32
20.03.	00	+	36
25.03.	A0	+	42
31.03.	B0	−	56

Abbildung 39: links r, rechts r_a

6.5 Regeln

1. ObsoleteDp: Obsolete Differential Privacy

   Diese drei Regeln sind zwar eher trivial und werden aufgrund aller Annahmen im Weiteren ausgeschlossen, doch sollen auch auf Algebra-Ebene nicht unerwähnt bleibt. Zum einen ist Verrauschen nicht nötig, wenn gar kein Attribut verrauscht wird, zum anderen, falls alle angegebenen Attribute mit einer Wahrscheinlichkeit von 0 verrauscht werden. Außerdem ist es ohne wirklichen Zweck, wenn alle angegebenen Attributnamen relationsfremd sind.

$$\text{(D1)} \quad {\alpha^D}_{P=\{\}}(r_1) \iff r_1$$

$$\text{(D2)} \quad {\alpha^D}_{P=\{(A,p_A=0)\ldots(Z,p_Z=0)\}}(r_1) \iff r_1$$

$$\text{(D3)}\quad {\alpha^D}_P(r_1) \iff r_1 \text{ falls } attr(P) \cap R_1 = \{\}$$

Die Regeln 2 und 3 könnten selbstverständlich auch in Kombination auftreten. In allen Fällen wird auf keinem der echt vorkommenden Attribute faktisch gesehen Verrauschung angewendet, sodass man eine Anonymisierung eigentlich bleiben lassen kann.

2. IdemDp: Verrauschen bezüglich Idempotenz und generelle Mehrfachanwendung

Verrauschung selbst verhält sich nicht idempotent. Mehrfach angewendetes Rauschen lässt den Anteil der echten Daten nämlich immer weiter verschwinden. Es verläuft also nicht trivial nach der Form:

$$\text{(D4)}\quad {\alpha^D}_P({\alpha^D}_P(r_1)) \not\Longleftrightarrow {\alpha^D}_P(r_1)$$

In Abbildung 40 wird unter gleichen Wahrscheinlichkeitsangaben zweifaches Verrauschen angewendet und mit einfachem verglichen. Bei geringer Wahrscheinlichkeit wie A werden bei zwei Instanzen höchstwahrscheinlich nur wenige Einträge verrauscht, wobei dies voneinander verschiedene sein werden. Bei einem höheren Wert wie bei B ist es dagegen wahrscheinlich, dass einige Einträge, die schon verrauscht wurden, erneut verrauscht würden. Der Anteil der gänzlich unverrauschten Einträge würde also immer weiter schwinden, aber totale Verrauschung zumeist durch Verrauschung von bereits Verrauschtem verhindert werden.

Bei $P = \{(A, 0.16), (B, 0.5)\}$ ergeben sich folgende Relationen:

A	B
01.01.	A0
07.01.	00
30.01.	BB
20.03.	A0
25.03.	A0
31.03.	AB

$\Longleftarrow$

A	B
01.01.	00
07.01.	00
31.01.	BB
20.03.	A0
25.03.	AA
31.03.	BB

$\Longleftarrow$

A	B
01.01.	00
05.01.	00
31.01.	B0
20.03.	A0
25.03.	A0
31.03.	B0

$\Longrightarrow$

A	B
01.01.	00
07.01.	00
31.01.	BB
20.03.	A0
25.03.	A0
31.03.	AB

Abbildung 40: Mehrfachverrauschung nicht idempotent

Zunächst einmal kann man sagen, dass sich Mehrfachverrauschung auf disjunkten Attributen sowohl problemlos vereinigend zusammenfassen als auch vertauschen ließe.

$$\text{(D5)}\quad {\alpha^D}_{P_1}({\alpha^D}_{P_2}(r_1)) \iff {\alpha^D}_{P_1 \cup P_2}(r_1) \iff {\alpha^D}_{P_2}({\alpha^D}_{P_1}(r_1))$$

$$\text{falls } attr(P_1) \cap attr(P_2) = \{\}$$

Ferner kann man Mehrfachverrauschung sehr wohl vereinfachen, wenn man für jedes Attribut eine neue Gesamtwahrscheinlichkeit errechnet. Man sollte sich hierzu am besten alle möglichen Fälle vergegenwärtigen. Betrachten wir der Einfachheit halber zuerst eine doppelte Anwendung des Operators die bei identischen Argumenten, also Verrauschung

auf den gleichen Attributen mit gleichen Einzelwahrscheinlichkeiten. Im Folgenden interessiert uns nur, ob ein Wert unverändert bleibt, sprich echt, oder nicht. Inwiefern sich verrauschte Werte bei erneuter Verrauschung verändern, ist hier nicht weiter von Belang.

Für irgendein Attribut besagt Ereignis V, dass dessen Verrauschung mit eine Wahrscheinlichkeit von p geschieht. Ereignis B sei der gegenteilige Fall, dass dessen aktueller Wert belassen wird. Sei p_i namentlich der aktuell unterschiedene Fall, I_1 die erste Ereignisinstanz, I_2 die zweite, U der Erhalt des Ursprungswertes und W die Wahrscheinlichkeit des Ganzen.

p_i	I_1	I_2	U	W
p_1	B	B	ja	(1-p) · (1-p)
p_2	V	B	nein	p · (1-p)
p_3	B	V	nein	(1-p) · p
p_4	V	V	nein	p · p

Tabelle 10: Fallunterscheidung bei zweifacher Verrauschung

Der gesamtheitliche Verrauschungsfalls setzt sich also zusammen aus p_2, p_3 und p_4. Es wäre formal aber einfacher, hier den Weg über die Gegenwahrscheinlichkeit zu wählen, als $1 - p_1$. Man müsste also einzig wissen, mit welcher Wahrscheinlichkeit der Ausgangswert über alle Instanzen unangetastet bliebe, und davon dann die Gegenwahrscheinlichkeit bilden. Bei einem identischen P_X gilt daher:

$$\text{(D6)} \quad {\alpha^D}_{P_X}(\alpha^D{}_{P_X}(r_1)) \iff {\alpha^D}_{P_Y}(r_1)$$

$$\text{mit } P_X = \{(X_1, p_{X_1}) ... (X_n, p_{X_n})\}$$

$$\text{und } \forall y \in attr(P_X) : P_Y = \{\ (y, p_y) \mid p_y = 1 - (1 - p_{X_y})^2\ \}$$

Abbildung 41 soll zeigen, wie die neu errechneten Werte in nur einem Durchlauf höchstwahrscheinlich ebenso viele Einträge verrauschen wie nach mehreren. Da die Anzahl der Einträge überschaubar bleiben musste, ist natürlich der wegfallende Anteil aufgrund der Diskretisierung zu berücksichtigen.

Bei $P_X = \{(A, 0.16), (B, 0.5)\}$ und daraus folgend $P_Y = \{(A, 0.2944), (B, 0.75)\}$ ergeben sich folgende Relationen:

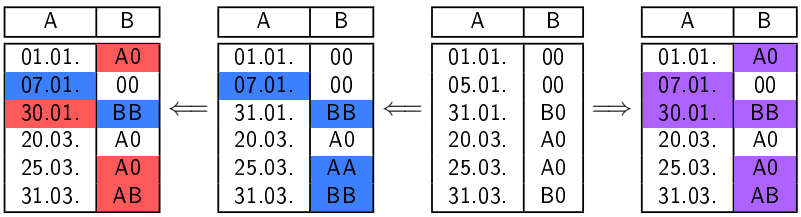


Abbildung 41: Zusammenfassbarkeit von Mehrfachverrauschung

Beim Allgemeinfall wäre noch zu bedenken, dass man zum einen unterschiedliche Verrauschungswerte für ein Attribut haben kann und zum anderen ein Attribut, dass in einer Instanz verrauscht wird, nicht zwingend auch in anderen Instanzen vorkommen muss. Formal hieße das:

$$\text{(D7)} \quad {\alpha^D}_{P_X}({\alpha^D}_{P_Y}(r_1)) \iff {\alpha^D}_{P_Z}(r_1)$$

$$\text{mit} \quad P_X = \{(X_1, p_{X_1})...(X_n, p_{X_n})\},\ P_Y = \{(Y_1, p_{Y_1})...(Y_m, p_{Y_m})\},$$

$$\forall z \in attr(P_X) \cup attr(P_Y) : P_Z = \{\ (z, p_z) \mid p_z = 1 - (1 - p_{Z_x}) \cdot (1 - p_{Z_y})\ \}$$

$$\begin{aligned} \text{sowie bei} \quad & z \in attr(P_X) \wedge z \in attr(P_Y) : \quad p_{Z_x} = p_{X_z},\ p_{Z_y} = p_{Y_z} \\ \text{bei} \quad & z \in attr(P_X) \wedge z \notin attr(P_Y) : \quad p_{Z_x} = p_{X_z},\ p_{Z_y} = 0 \\ \text{bei} \quad & z \notin attr(P_X) \wedge z \in attr(P_Y) : \quad p_{Z_x} = 0,\ p_{Z_y} = p_{Y_z} \end{aligned}$$

Recht leicht zu erkennen: Ist eine der Verrauschungswahrscheinlichkeiten gleich 0, entspricht der geklammerte Faktorausdruck 1, sodass der eigentliche Gesamtwert dann nur von den anderen abhängt. Sollte eine der Verrauschungswahrscheinlichkeiten 1 sein, entspricht der geklammerte Ausdruck 0, sodass ebenso automatisch alles verrauscht werden würde und alle anderen Wahrscheinlichkeiten nichts mehr zur Sache tun.

3. DpProj: Verrauschung und Projektion

Zwischen Projektion und Verrauschung kann problemlos getauscht werden. Aus formaler Sicht ist aber eine Unterscheidung zwischen beiden Tauschrichtungen nötig. Falls das Verrauschen auf Basis einer bereits projizierten Menge stattfindet, gemeinhin also $attr(P) \subseteq X$ gilt, kann man die Anonymisierung ohne weiteren Aufwand vorziehen. Formal hieße dies:

$$\text{(D8)} \quad {\alpha^D}_P(\pi_X(r_1)) \Longrightarrow \pi_X({\alpha^D}_P(r_1))$$

Abbildung 42 veranschaulicht die problemlose Vorziehbarkeit des Verrauschungsoperators, da all jene ungenannten Attribute, zu denen auch die eigentlich Wegprojizierten zählen, stets unverändert bleiben.

Bei X={A,B}, P={(A,p_A=0.16),(B,p_B=0.5)} ergeben sich folgende Relationen:

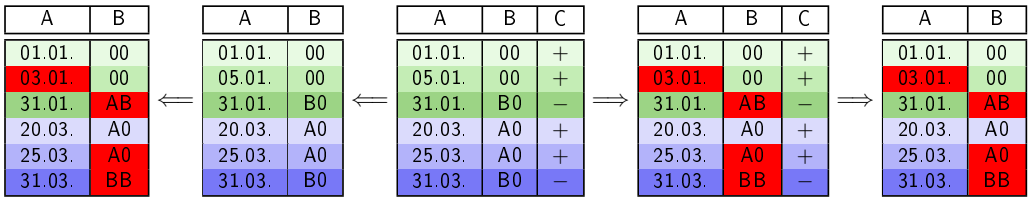


Abbildung 42: Vorziehbarkeit der Verrauschung bei Projektion

Im Falle von $attr(P) \subseteq X$ ließe sich selbstverständlich auch die Rückrichtung vollziehen. Ganz allgemein ist aber ein Vorziehen der Projektion, also der umgekehrte Fall, korrekt durchführbar, wenn alle Verrauschungsattribute, die ohnehin wegprojiziert würden, einfach ordnungsgemäß herausgestrichen werden.

$$\text{(D9)} \quad \alpha^D{}_{P'}(\pi_X(r_1)) \Longleftarrow \pi_X(\alpha^D{}_P(r_1))$$

$$\text{mit } P' = \{\ (A,\, p_A)\ \mid\ \forall A \in attr(P) \wedge A \in X\ \}$$

Abbildung 43 verdeutlicht noch einmal die Äquivalenz des Ganzen, wobei der einzige Aufwand darin besteht, das ohnehin wegprojizierte C nun vorsorglich aus der Verrauschungsmenge zu streichen.

Bei X={A,B}, P={(A,p_A=0.16),(B,p_B=0.5),(C,p_C=0.16)} und daraus folgend P′={(A,p_A=0.16),(B,p_B=0.5)} ergeben sich folgende Relationen:

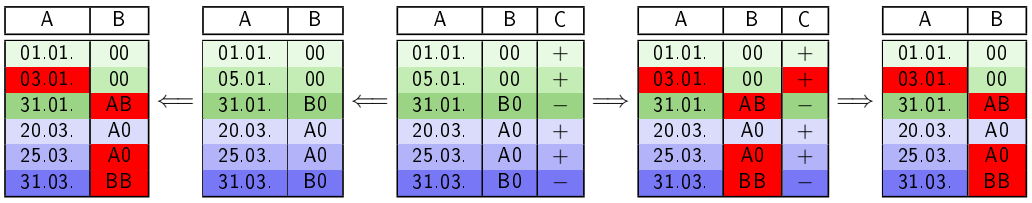


Abbildung 43: Vorziehbarkeit der Projektion bei Verrauschung

Des Weiteren ergäbe sich auch diesmal noch ein Spezialfall, der ebenso in Richtung obsoletes Verrauschen ginge. Sofern die letztendliche Projektion nämlich nur auf unverrauschten Attributen stattfinden würde, könnte man sich das Verrauschen auch sparen.

$$\text{(D10)} \quad \pi_X(r_1) \Longleftarrow \pi_X(\alpha^D{}_P(r_1)) \text{ falls } attr(P) \cap X = \{\}$$

Da das einzige Verrauschungsattribut B in der Abbildung 44 letztendlich ausgeblendet wird, kann die Verrauschung auch übersprungen werden.

Bei X={A,C}, P={(B,p_B=0.5)} ergeben sich folgende Relationen:

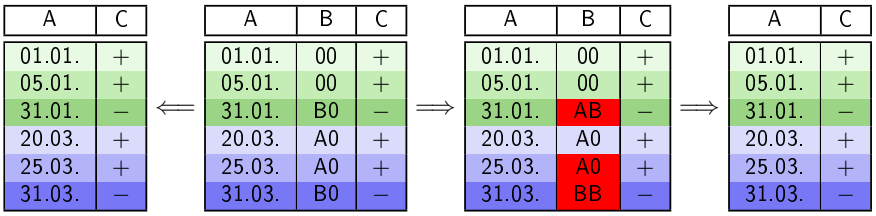


Abbildung 44: Obsolete Verrauschung bei disjunkter Projektion

4. DpSel: Verrauschung und Selektion

   Selektion und Verrauschung stehen in einem schwierigen Verhältnis, da es durchaus problematisch ist, auf äquivalente Weise auf einer Relation zu selektieren, wo der Anonymisierungsoperator doch immerhin Werte verändert. Einerseits könnte Verrauschung neue Selektionskandidaten schaffen, andererseits die eigentlichen Kandidaten abändern, sodass sie herausfallen könnten. Da dem Ganzen nicht so etwas wie eine eindeutige, überall greifende Ersetzung zugrunde liegt, bringt es einem in diesem Falle auch nichts etwas am Selektionsausdruck zu verändern. Ein Tausch ließe sich jedoch sehr wohl vollziehen, falls die Selektionsattribute unter keinen Umständen verrauscht werden würden.

   $$\text{(D11)}\quad {\alpha^D}_P(\sigma_F(r_1)) \iff \sigma_F({\alpha^D}_P(r_1)) \text{ falls } attr(P) \cap attr(F) = \{\}$$

   Abbildung 45 verdeutlicht im Folgenden die Gleichheit dieser Regel. Da auf Grundlage des unverrauschten Attributes B selektiert wird, verläuft dies in beiden Verläufen auf eindeutige Weise. Zwar wird in einem Fall die Gesamtrelation verrauscht und im anderen die selektierte Teilmenge, doch das Verrauschen ist ja bekanntlich zeilenunabhängig. Sowohl der selektierte als auch der wegselektierte Anteil dürfte im wahrscheinlichsten Falle daher ein ähnliches Verhältnis zwischen verrauschten und unverrauschten Attributwerten aufweisen.

   Bei P={(A,p_A=0.5)}, F: B≠A0 ergeben sich folgende Relationen:

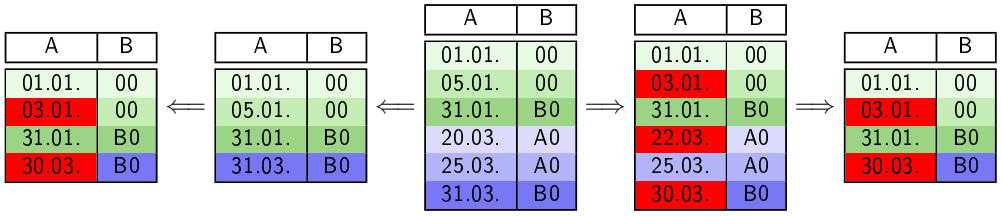


   Abbildung 45: Verrauschung und Selektion

5. DpUnion: Verrauschen und Vereinigung

   Da die Verrauschung zeilen- und spaltenunabhängig erfolgt, sind beide Operatoren bei gleichen Verrauschungsparametern ohne Weiteres verschiebbar. Es macht daher keinen Unterschied, ob man auf der Gesamtrelation anonymisiert oder dies auf horizontalen Partitionen tut. Eine Parallelisierung ist also kein Problem.

   $$\text{(D12)}\quad {\alpha^D}_P(r_1 \cup r_2) \iff {\alpha^D}_P(r_1) \cup {\alpha^D}_P(r_2)$$

   Genau diese Möglichkeit ist noch einmal in Abbildung 46 verdeutlicht.

Bei P={(A,p_A=0.16),(B,p_B=0.5)} ergeben sich folgende Relationen:

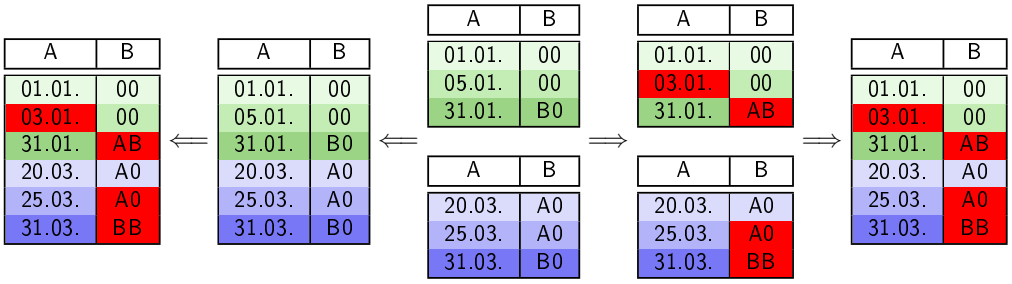


Abbildung 46: Verrauschung und Vereinigung

6. DpJoin: Verrauschen und Verbund

Bereits beim Slicing ließ sich der allgemeine Verbund nicht ohne Weiteres verschieben. Die allgemeinen Gründe bei DpJoin sind so ziemlich wie die beim SliceJoin, nur dass neue Wertkonstellationen nicht aufgrund Permutation untereinander zustande kommen, sondern wegen ganz gezielter Änderung. Demnach ist es wie auch dort zumindest möglich auf vertikalen Partitionen zu anonymisieren, natürlich unter der getroffenen Einschränkung, dass die Verrauschung spaltenunabhängig ist.

Zumindest bei vertikaler Partitionierung gilt:

$$\text{(D13)}\quad \alpha^D{}_{P_1 \cup P_2}(\pi_R(r_1 \bowtie r_2)) \iff \pi_R(\alpha^D{}_{P_1}(r_1) \bowtie \alpha^D{}_{P_2}(r_2))$$

$$\text{falls}\quad attr(P_1) \subseteq R_1,\ attr(P_2) \subseteq R_2,\ attr(P_1) \cap attr(P_2) = \{\},\ \text{Verbund mittels ID},\ R = (R_1 \cup R_2) \setminus ID$$

Abbildung 47 verdeutlicht die Regeläquivalenz unter den angegebenen Umständen.

Bei P={(A,p_A=0.16),(B,p_B=0.5)} ergeben sich folgende Relationen:

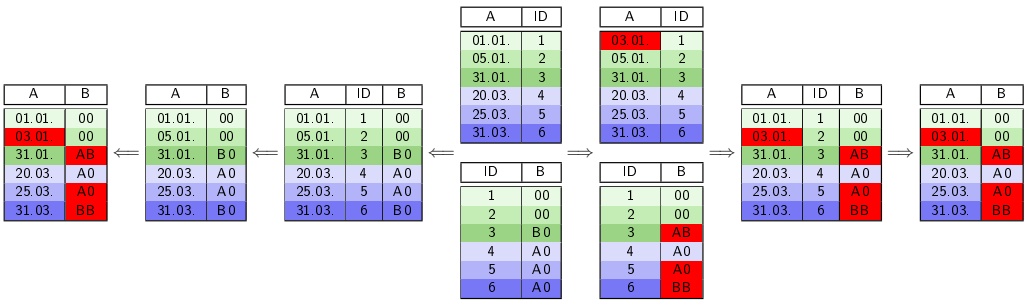


Abbildung 47: Verrauschung und Verbund bei vertikaler Partitionierung

7. DpInter, DpDiff: Verrauschung bei Schnittmenge und Differenzmenge

Aufgrund der generellen Zufälligkeit des Verrauschens sind Schnitt- und Differenzmengenbildung nicht verschiebbar. Falls man auf Basis der Mengenoperation verrauschen will, geschähe dies zumindest stets auf einer eindeutigen Relation mit einer bestimmten Größe. Beim umgekehrten Fall ist nicht einmal gewiss, ob beziehungsweise wie viele Einträge man am Ende erhält.

$$\text{(D14)} \quad {\alpha^D}_P(r_1 \cap r_2) \not\Longleftrightarrow {\alpha^D}_P(r_1) \cap {\alpha^D}_P(r_2)$$
$$\text{(D15)} \quad {\alpha^D}_P(r_1 - r_2) \not\Longleftrightarrow {\alpha^D}_P(r_1) - {\alpha^D}_P(r_2)$$

Abbildung 48 macht noch einmal deutlich, warum eine Vertauschung nicht gleichbedeutend sein kann. Das linksseitige Ergebnis entstünde nämlich nur dann auch auf rechtsseitigem Wege, wenn die vorgezogene Verrauschung nichts daran ändern würde, welche Einträge bei der Mengenoperation blieben und welche ausgeschlossen werden würden. Eigentlich ausgeschlossene Werte könnten aber einen Weg ins Ergebnis finden, wenn die Verrauschung sie auf beiden Seiten gleich machen würde, und eigentlich bleibende könnten plötzlich ausgeschlossen werden, wenn das Verrauschen sie verschieden machen würde.

Bei P={(A,p_A=0.16),(B,p_B=0.5)} ergeben sich veranschaulicht an der Schnittmenge folgende Relationen:

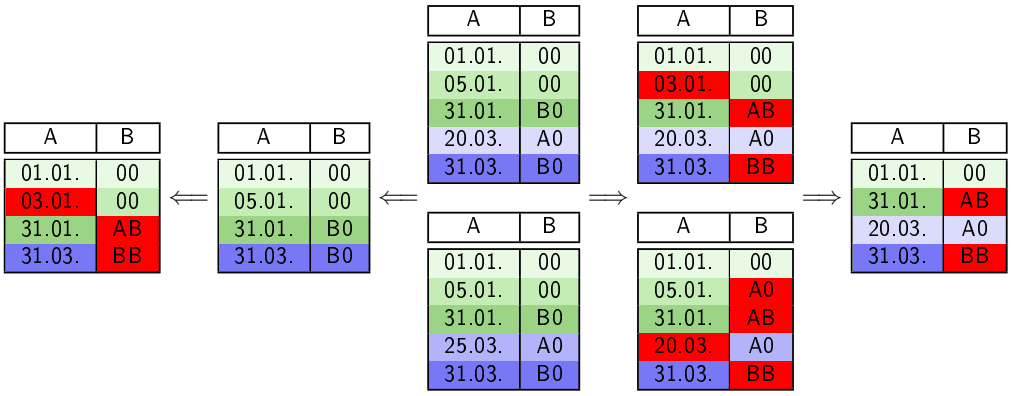


Abbildung 48: Verrauschung und Schnittmenge

7 Auswertung

In diesem Kapitel werden alle Regeln, die in Kapitel 4.4, 5.5 und 6.5 aufgestellt wurden, noch einmal abschließend zusammengefasst. Zum einen wird die Bandbreite der Vertauschbarkeit gegenübergestellt, zum anderen werden alle Regeln, die mit Partitionen arbeiten, eine konkrete Bewertung erhalten.

7.1 Vertauschbarkeit

Tabelle 11 verdeutlicht, wie die Vertauschbarkeit zweier Operatoren im Einzelnen klassifiziert wird. Sei ‚X' gleichbedeutend mit ja und ‚–' mit nein. Der Pfeil ‚↔' soll ausdrücken, ob irgendeine Form von Vertauschung existiert. Die Bedingung ‚IF' spezifiziert die Vertauschbarkeit dann gegebenenfalls genauer. Bei einem Ja ist diese streng betrachtet nur unter bedingten Umständen möglich, bei einem Nein ist diese uneingeschränkt. Eine Klassifizierung ergibt sich immer aus der gesamtheitlichen Betrachtung aller Teilregeln. Eine Vertauschung kann daher ebenfalls uneingeschränkt sein, wenn mehrere bedingte Regeln innerhalb ihrer Fallunterscheidung alle Eventualitäten abdecken und die gleiche Äquivalenzart zulassen. Äquivalenz ist als eine Spezialform der eigens eingeführten Teiläquivalenz anzusehen, was anders ausgedrückt heißt, dass das teiläquivalente Regelspektrum mindestens gleichviel oder mehr Vertauschung gestattet.

↔	IF	Vertauschbarkeit
X	–	uneingeschränkt
X	X	bedingt
–	–	nicht möglich

Tabelle 11: Klassifikation zweier Operatoren hinsichtlich deren Vertauschbarkeit

In Tabelle 12 wird das gesamte Regelspektrum hinsichtlich äquivalenter Vertauschung abgebildet. Differential Privacy scheint angesichts der Klassifizierung die meisten gleichbedeutenden Umformungen zu erlauben, gefolgt von Slicing, gefolgt von Generalisierung. Diese Tendenz lässt sich dadurch erklären, dass sich die jeweilige Anonymisierung immer weiter horizontal und vertikal abhängiger wird. Bezüglich scheinen Mehrfachanwendung und Projektion scheinen alle drei Techniken relativ viele Möglichkeiten bieten, bezüglich Verbund zumindest einige. Letztere belaufen sich, wie wir durch unsere Untersuchungen wissen, dabei nicht auf einen bedingten allgemeinen Verbund, sondern viel spezieller auf vertikale Partitionen. Bezüglich Schnitt- und Differenzmenge scheint nicht eine der Techniken auch nur irgendeine Umformung zuzulassen.

⟺	Idem		π		σ		$\cup$		$\bowtie$		$\cap$		$-$	
	↔	IF	↔	IF	↔	IF	↔	IF	↔	IF	↔	IF	↔	IF
Gen	X	X	X	X	–	–	–	–	X	X	–	–	–	–
Slice	X	–	X	–	–	–	–	–	X	X	–	–	–	–
DP	X	–	X	–	X	X	X	–	X	X	–	–	–	–

Tabelle 12: Regelspektrum hinsichtlich Äquivalenz

In Tabelle 13 ist das teiläquivalente Regelspektrum dargestellt. Es baut auf dem äquivalen-

ten Spektrum auf. Regeln, die keinerlei Erweiterung erfahren haben, sind daran erkennbar, dass sie ausgegraut wurden. Eine teiläquivalente Betrachtung scheint zwei Dinge nun zu begünstigen: zum einen die Generalisierung, zum anderen die allgemeine Vereinigung. Durch den Zuwachs an Regeln aufgrund der geringeren Anforderungen haben sich Generalisierung und Slicing sowohl untereinander als auch allgemein dem Spektrum von Differential Privacy weiter angenähert.

$\sqsubseteq$ / $\Longleftrightarrow$	Idem		π		σ		$\cup$		$\bowtie$		$\cap$		$-$	
	$\leftrightarrow$	IF	$\leftrightarrow$	IF	$\leftrightarrow$	IF	$\leftrightarrow$	IF	$\leftrightarrow$	IF	$\leftrightarrow$	IF	$\leftrightarrow$	IF
Gen	X	X	X	–	X	X	X	–	X	X	–	–	–	–
Slice	X	–	X	–	–	–	X	–	X	X	–	–	–	–
DP	X	–	X	–	X	X	X	–	X	X	–	–	–	–

Tabelle 13: Erweitertes Regelspektrum bei Teiläquivalenz

7.2 Bewertung verteilbarer Techniken

Alle Regeln, bei denen ein binärer Operator beteiligt ist, die also mit Teilrelationen arbeiten, sodass sich potentiell eine Parallelisierbarkeit ergeben könnte, seien im Folgenden noch einmal abschließend gegenübergestellt und bewertet. Da sich bei der Schnitt- und Differenzmenge grundsätzlich keine Regeln finden ließ, handeln alle Zusammenfassungen entweder von Vereinigung oder Verbund.

Generalisierung

$\alpha^G{}_{[X,k=a]}(r_1) \cup \alpha^G{}_{[X,k=a]}(r_2) \overset{\sqsubseteq}{\Longleftarrow} \alpha^G{}_{[X,k=a]}(r_1 \cup r_2)$	
Verfahren:	Generalisierung (k-Anonymität)
Partitionierung:	horizontal
Regelverweis:	Teil der GenUnion-Regel
Annahmen:	siehe Kapitel 4.2
Äquivalenzart:	Teiläquivalenz (siehe Kapitel 3.3)
Randbedingung:	keine (die nötige Mindestgröße der Partitionen sei aber noch einmal erwähnt, siehe Annahmen)
Vor- und Nachteile:	+ Parallelisierbarkeit ± Aussagegehalt kann sich gleich bleiben, sich verbessern aber auch verschlechtern (eine kleine Untersuchung dazu ist im Anhang bei A zu finden) − unterschiedliche Generalisierungsebenen möglich (generalisierte Wertebereiche können dann überlappen)
Nützlichkeit:	nützlich, sofern man mögliche Einbußen mit in Kauf nimmt
Bemerkungen:	Eine Aufteilung kann manchmal seine Vorteile, manchmal seine Nachteile haben. Letztlich ist es ungewiss, was genau überwiegt, da es fallbezogen von der Ausgangsrelation abhängt. Gesamtheitliche Generalisierung mag einheitliche Generalisierungsebenen sicherstellen, doch die Partitionierung zeigt, dass sie seitens des Aussagegehaltes nicht universell besser sein muss. Die aussagekräftigste Generalisierung wäre tatsächlich jene, die gesamtheitlich arbeitet, unterschiedliche Ebenen erlaubt und schwer generalisierbare Ausreißer ausfindig macht, um diese geschlossen von den anderen zu anonymisieren.

$\alpha^G{}_{[X,k=a]}(r_1) \bowtie r_2 \overset{\sqsubseteq}{\Longleftarrow} \alpha^G{}_{[X,k=a]}(r_1 \bowtie r_2)$	
Verfahren:	Generalisierung (k-Anonymität)
Partitionierung:	vertikal
Regelverweis:	Teil der GenJoin-Regel
Annahmen:	siehe Kapitel 4.2
Äquivalenzart:	Teiläquivalenz (siehe Kapitel 3.3)
Randbedingung:	der Verbund soll auch tatsächlich weitere Einträge hervorbringen (kein bloßer Verbund mittels ID); für die genauen Randbedingungen sei aufgrund des Umfangs auf GenJoin verwiesen
Vor- und Nachteile:	+ Kostenersparnis bei der Anonymisierung aufgrund der unverbundenen, komprimierten Relation − keinerlei Parallelisierung möglich − Angreifbarkeit der Verbundattribute, da neue Quasi-Identifikatoren produziert werden (unter Umständen natürlich ausblendbar) − kann riesige Gruppen erzeugen, was für einen relativ schwachen Aussagegehalt sorgt − aufgrund der komplexen Attributeinteilung und all der Defizite wohl kaum als Optimierungsregel einsetzbar
Nützlichkeit:	nur bedingt nützlich
Bemerkungen:	Je größer das Ersparnis an Einträgen aufgrund der Kompression ist, umso mehr leidet tendenziell der schlussendliche Aussagegehalt darunter. Die Umstände für praktischen Gebrauch und Nutzen sind auf jeden Fall relativ speziell.

$\pi_R\,(\alpha^G{}_{[X,k=a]}(r_1) \bowtie r_2) \Longleftarrow \alpha^G{}_{[X,k=a]}(\pi_R(r_1 \bowtie r_2))$	
Verfahren:	Generalisierung (k-Anonymität)
Partitionierung:	vertikal
Regelverweis:	Teil der GenJoin-Regel
Annahmen:	siehe Kapitel 4.2
Äquivalenzart:	Äquivalenz
Randbedingung:	Verbund mittels Hilfs-ID; für die genauen Randbedingungen sei aufgrund des Umfangs auf GenJoin verwiesen
Vor- und Nachteile:	+ tendenzielle Kostenersparnisse bei vielen Einträgen und vielen sensitiven oder sonstigen Attributen − keinerlei Parallelisierung möglich − aufgrund der komplexen Attributeinteilung wohl kaum als Optimierungsregel einsetzbar
Nützlichkeit:	nur bedingt nützlich
Bemerkungen:	Ein wirklicher Nutzen ergibt sich erst, wenn man eine gesamtheitliche Generalisierung bei großen Relationen anstrebt. An dieser Stelle sei im Übrigen noch einmal erwähnt, dass es bei geschlossener Generalisierung generell nicht möglich ist, auf Basis von verteilten Generalisierungsattributen zu anonymisieren. Jede der Partitionen ließe sich zwar k-anonym machen, doch der Verbund ist dies noch lange nicht, da die Generalisierungsattribute gesamtheitlich betrachtet Gruppen von geringerer Größe ausbilden können. An dieser Stelle sei daher auf zwei Arbeiten verwiesen, die sich mit genau dem beschäftigt haben. Zum einen das Verfahren von [JC05], das trotz verteilten Generalisierungsattributen k-Anonymität sicherstellen kann, zum anderen das Verfahren von [NJTF15], das darauf aufbauend t-closeness erbringt.

Slicing

$\alpha^S{}_{V=\{X_1 \ldots X_n\}}(r_1) \cup \alpha^S{}_{V=\{X_1 \ldots X_n\}}(r_2) \overset{\sqsubseteq}{\Longleftarrow} \alpha^S{}_{V=\{X_1 \ldots X_n\}}(r_1 \cup r_2)$	
Verfahren:	Slicing
Partitionierung:	horizontal
Regelverweis:	Teil der SliceUnion-Regel
Annahmen:	siehe Kapitel 5.2
Äquivalenzart:	Teiläquivalenz (siehe Kapitel 3.3)
Randbedingung:	keine (die nötige Mindestgröße der Partitionen sei aber noch einmal erwähnt, siehe Annahmen)
Vor- und Nachteile:	+ Parallelisierbarkeit + keinerlei Nachteile aus reiner Anonymitätssicht − andere Permutationsgruppen gegenüber Gesamtrelation möglich
Nützlichkeit:	eindeutig nützlich
Bemerkungen:	Slicing ist eigentlich nur an der Auflösung von Korrelationsbeziehungen interessiert. Die Aufteilung der Einträge und schlussendliche Permutation hat keine höhere Relevanz. Aus horizontaler Sicht mag es Änderungen geben, aus vertikaler Sicht dagegen nicht. Praktischen gesehen ist es daher egal, ob man gesamtheitlich oder partitioniert slict.

$\pi_R(\alpha^S{}_{V_1=\{X_1 \ldots X_i ID\}}(r_1) \bowtie \alpha^S{}_{V_2=\{X_j \ldots X_n ID\}}(r_2)) \Longleftarrow \alpha^S{}_{V=\{X_1 \ldots X_i X_j \ldots X_n\}}(\pi_R(r_1 \bowtie r_2))$	
Verfahren:	Slicing
Partitionierung:	vertikal
Regelverweis:	Teil der SliceJoin-Regel
Annahmen:	siehe Kapitel 5.2
Äquivalenzart:	Äquivalenz (gleiche Reihenfolge hinsichtlich der ID vorausgesetzt, ansonsten teiläquivalent)
Randbedingung:	Verbund mittels Hilfs-ID (als zusätzliche Korrelationsgruppe), $X_1 \cup \ldots \cup X_i \cup ID = R_1$, $X_j \cup \ldots \cup X_n \cup ID = R_2$, $R = (R_1 \cup R_2) \setminus ID$
Vor- und Nachteile:	+ Parallelisierbarkeit + auch hier, also wie beim horizontalen Slicen, ist selbst Teiläquivalenz im Grunde genauso gut wie Äquivalenz − zu speziell, als dass es ohne Weiteres eine allgemeine Optimierungsregel sein könnte (auf die ID ist selbstständig zu achten)
Nützlichkeit:	nützlich, aber mit eigenem Aufwand verbunden
Bemerkungen:	Da Korrelationsgruppen keinerlei Beziehung zu anderen Attributen mehr haben werden, ist es nur logisch, dass man diese auch voneinander getrennt slicen kann.

Differential Privacy

$\alpha^D{}_P(r_1) \cup \alpha^D{}_P(r_2) \impliedby \alpha^D{}_P(r_1 \cup r_2)$	
Verfahren:	Differential Privacy (einfache Randomisierung)
Partitionierung:	horizontal
Regelverweis:	Teil der DpUnion-Regel
Annahmen:	siehe Kapitel 6.2
Äquivalenzart:	Äquivalenz
Randbedingung:	keine
Vor- und Nachteile:	+ Parallelisierbarkeit + uneingeschränkt gültig + als echte Optimierungsregel einsetzbar
Nützlichkeit:	eindeutig nützlich
Bemerkungen:	Da bei der Verrauschung sowohl jede Zeile als auch Spalte unabhängig von anderen ist, kann uneingeschränkt äquivalent partitioniert werden.

$\pi_R(\alpha^D{}_{P_1}(r_1) \bowtie \alpha^D{}_{P_2}(r_2)) \impliedby \alpha^D{}_{P_1 \cup P_2}(\pi_R(r_1 \bowtie r_2))$	
Verfahren:	Differential Privacy (einfache Randomisierung)
Partitionierung:	vertikal
Regelverweis:	Teil der DpJoin-Regel
Annahmen:	siehe Kapitel 6.2
Äquivalenzart:	Äquivalenz
Randbedingung:	Verbund mittels Hilfs-ID (als ein unverrauschtes Attribut), $attr(P_1) \subseteq R_1$, $attr(P_2) \subseteq R_2$, $attr(P_1) \cap attr(P_2) = \{\}$, $R = (R_1 \cup R_2) \setminus ID$
Vor- und Nachteile:	+ Parallelisierbarkeit − zu speziell, als dass es ohne weiteres eine allgemeine Optimierungsregel sein könnte (auf die ID ist selbstständig zu achten)
Nützlichkeit:	nützlich, aber mit eigenem Aufwand verbunden
Bemerkungen:	Da bei der Verrauschung sowohl jede Zeile als auch Spalte unabhängig von anderen ist, kann uneingeschränkt äquivalent partitioniert werden.

8 Praxisteil

Dieses Kapitel widmet sich der konkreten Realisierung von Regeltransformationen auf Basis der relationalen Algebra. Zunächst wird kurz das selbstgewählte Regelspektrum, das Kernbestandteil dieses Kapitels sein wird, veranschaulicht. Danach wird noch relativ abstrakt jede einzelne Transformation beschrieben. Im Anschluss wird ein sogenanntes Rewriting-Konzept vorgestellt, das auf der Arbeit von [GH18] basiert und die geregelte Umformung relational-algebraischer Ausdrücke ermöglicht. Zum Zwecke dieser Arbeit wird es aufgegriffen und um Anonymisierungsoperationen erweitert. Da mehrere Regeln praktisch umgesetzt werden, wird die prinzipielle Implementierung im Folgenden anhand einer exemplarisch ausgewählten veranschaulicht.

8.1 Umsetzungsspektrum

Die Anzahl der Regeln, die im Verlauf dieser Arbeit gefunden und untersucht wurde, ist nicht gerade klein. Da manche Regeln sich strukturell ähneln, andere viel zu komplexe Vorbedingungen aufweisen und es ohnehin keinen grundlegenden Ergebnisgewinn gibt, sollte hinreichend klar sein, warum nur einige ausgewählte umgesetzt wurden. Da der Verrauschungsoperator das größte Spektrum an äquivalenten Regeln aufweist und es gut wäre, sowohl eine Vertauschung mit einem unären als auch mit einem binären Operator zu verwirklichen, fiel die Wahl auf DpUnion und DpProj. Tabelle 14 veranschaulicht die Benennung, zusätzlich nötige Aufteilung sowie die Randbedingungen der beiden gewählten Regelabschnitte. Damit die Umformung nicht etwa einfach fallabhängig umgekehrt werden, sondern immer bewusst geschehen, sind alle äquivalenten bzw. beidseitig möglichen Regel D12 aufgeteilt worden. Des Weiteren würde bei der Auswahl an Regeln darauf geachtet, dass ein diverses Spektrum an Feinheiten bedient wird: seien es nun Regel, die sich immer durchführen lassen (DP01, DP03, DP04), unter bedingten Umständen (DP03, DP05) und ebenso welche, die sogar eine Auswirkung auf konkrete Parameter haben (DP04).

Name	Regelfeld	Regel	spezifische Transformation	Randbedingung
DP01	DpUnion	D12	$\alpha^D{}_P(r_1 \cup r_2) \Longrightarrow \alpha^D{}_P(r_1) \cup \alpha^D{}_P(r_2)$	—
DP02		D12	$\alpha^D{}_P(r_1 \cup r_2) \Longleftarrow \alpha^D{}_P(r_1) \cup \alpha^D{}_P(r_2)$	tatsächliche Gleichheit beider Verrauschungsmengen überprüfen
DP03	DpProj	D8	$\alpha^D{}_P(\pi_X(r_1)) \Longrightarrow \pi_X(\alpha^D{}_P(r_1))$	—
DP04		D9	$\alpha^D{}_{P'}(\pi_X(r_1)) \Longleftarrow \pi_X(\alpha^D{}_P(r_1))$	$P' = \{\ (A, p_A)\ \mid\ \forall A \in attr(P) \wedge A \in X\ \}$
DP05		D10	$\pi_X(r_1) \Longleftarrow \pi_X(\alpha^D{}_P(r_1))$	$attr(P) \cap X = \{\}$

Tabelle 14: Regelspektrum der praktischen Umsetzung

8.2 Abstrakte Umsetzung

Eine Möglichkeit zum Darstellen und Umformen von relational-algebraischen Ausdrücken ist keineswegs unerforscht. Das Umschreiben von Anfragen ist beispielsweise Thema der Arbeit von [GH18] gewesen. Ebenjenes Konzept wird im Folgenden als maßgebliche Grundlage genutzt. Die Implementierung des Ganzen fand nicht etwa bei Null statt, sondern baut auf ein zur Verfügung gestelltes Programm namens Rewriter auf. Da sich alle weiteren

Beschreibungen in irgendeiner Form danach richten, sei der Rewriter zunächst einmal grob erklärt.

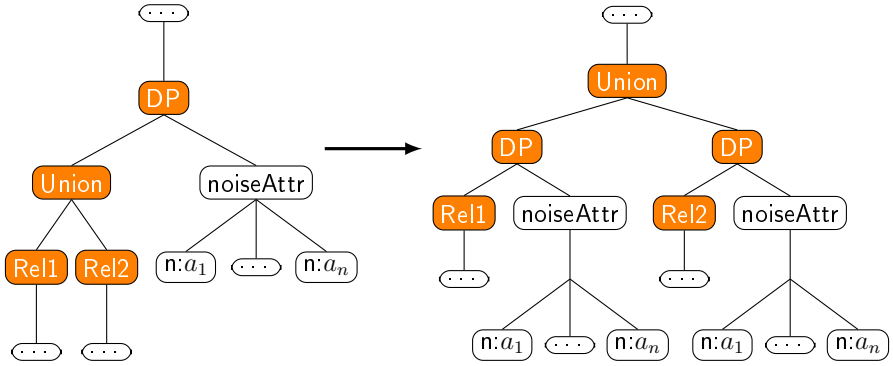
Beim Rewriter handelt es sich um ein über Eclipse erstelltes Java-Projekt. Er bietet eine einfache Möglichkeit zur Darstellung, Transformation und Testung relation-algebraischer Ausdrücke. Zunächst einmal sind elementare Operatoren und deren Attribute als Klassen abzubilden. Mittels abstrakter Klassen sowie Vererbung ist es dann obendrein möglich, diese zu schachteln, sodass sich beliebige Ausdrücke konstruieren lassen. Darüber hinaus können diese zu einem XML-Dokument serialisiert werden. Auf Basis dieser Serialisierung und unter Nutzung von XPath lassen sich diese wiederum auf leichtem Wege durchsuchen und gegebenenfalls transformieren. Aufgrund der Gegebenheiten ist es auch überdies möglich, Testfälle zu verfassen.

In Kapitel 8.1 wurden all jene Regeln (DP01 bis DP05) vorgestellt, die eine praktische Umsetzung erhalten sollen. In diesem Abschnitt wird nun zu jeder der fünf Regeln beschrieben, wie sich deren formale Transformation im Einzelnen gestalten wird. Da das Rewriting-Konzept von 14 die XML-Sprache zur Darstellung von relationalen Abfragen wählte und wir auf ebendieses Konzept aufbauen wollen, wird auch hier zur konkreten Abbildung von Ausdrücken XML genutzt. Was eine Transformationsregel also tut, ist die Durchsuchung des XML-Baums, die Erkennung von Strukturen und gegebenenfalls deren funktionale Veränderung. Jede Regel wird im Folgenden durch drei Dinge beschrieben: eine Abbildung der Transformation in baumartiger Form, eine Übersicht zur Knotenidentifizierung mittels XPath-Ausdrücken sowie eine Auflistung der Bedingungen und exemplarisch nötigen Schritte zur jeweiligen Umformung.

Das schon einige Male erwähnte XPath ist übrigens eine formale Abfragesprache zur Adressierung und Durchsuchung von XML-Dokumenten. Sie wird im Folgenden mehrfach verwendet, weil sie auch maßgeblich in der Implementierung zum Einsatz kommt. Die hier abgebildeten Bäume weisen die gleiche Struktur wie die XML-Tag-Bäume in der Implementierung auf. Aus Platz- bzw. Übersichtsgründen sind hier aber deutlich kompaktere Tag-Namen als dort gebraucht worden.

Darüber hinaus noch einige Anmerkungen zu den Baumdarstellungen: Orange gefüllt Knoten symbolisieren all jene, die explizit für die reine Transformation gebraucht wurden – zum Anfügen, Entfernen oder um dies indirekt mit dessen Elter oder einem von dessen Kindern zu tun. Natürlich könnte die Ausgangsmenge auch eine etwas andere sein. Das Ganze ist abgestimmt auf die Schrittes des Ablaufs, also rein exemplarisch zu verstehen. Des Weiteren gibt es noch Knoten mit blau markiertem Text. Alle diese Knoten sind zwingend relevant, wenn es eine bestimmte Vorbedingung zu prüfen gilt. Um gewisse Dinge wie konkrete Textwerte, Vervielfältigung und Löschung zu verdeutlichen, ist bei manchen Knoten darüber hinaus ein stellvertretender kursiver Wert mit Index eingefügt. Er ist losgelöst vom Tag-Namen zu betrachten und der Tatsache geschuldet, dass es neben den XML-Tag-Elementen noch Attribute und Textwerte geben kann. Natürlich könnte man dafür eine ausgeklügelte, komplexere Baum-Darstellung etablieren, aber im Anbetracht der Umstände besäße dies keinen höheren Wert.

DP01:

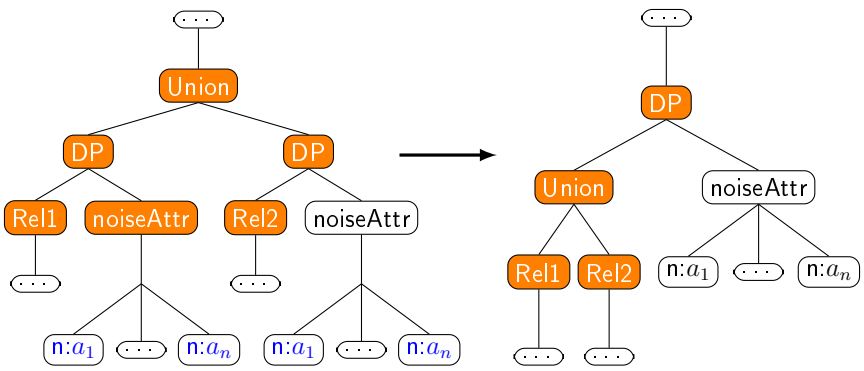


eindeutige Knotenbezeich.	Ermittelnder XPath-Ausdruck
dp	`//DP[Union]`
un	`//DP/Union`
r1	`//DP/Union/[1]`
r2	`//DP/Union/[2]`

Vorbedingungen: keine
Exemplarischer Ablauf:

1. Verrauschung dp duplizieren, um den reinen Operator sowie dessen genaue Attribute mehrfach vorliegend zu haben (im Folgenden dp1 und dp2 genannt)
2. Verrauschungen unter Vereinigung un positionieren
   (a) bei Vereinigung un ist r1, r2 durch dp1, dp2 zu ersetzen
   (b) alte Subrelationen von dp1, dp2 entfernen
   (c) je eine der Basisrelationen unter dp1, dp2 anfügen
3. Vaterknoten von dp an un übergeben (Erhalt der unbekannten relativen Wurzel)

DP02:

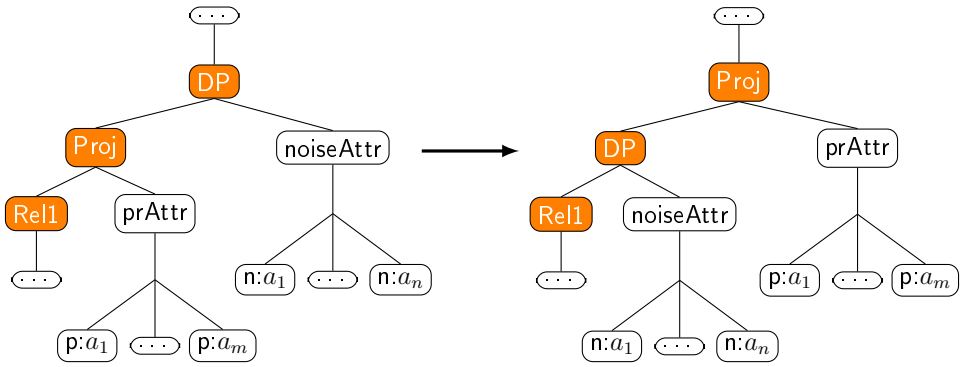


eindeutige Knotenbezeich.	Ermittelnder XPath-Ausdruck
un	`//Union[DP[1]][DP[2]]`
dp1	`//Union/DP[1]`
dp2	`//Union/DP[2]`
na1	`//Union/DP[1]/noiseAttr`
na2	`//Union/DP[2]/noiseAttr`
r1	`//Union/DP[1]/[name()!='noiseAttr']`
r2	`//Union/DP[2]/[name()!='noiseAttr']`

Vorbedingungen:
- beide Verrauschungsattributmengen sollten vor der Umformung auf tatsächliche Gleichheit überprüft werden
- hierbei muss die Werte aller Kinder von na1 mit denen von na2 mengentheoretisch gesehen übereinstimmen
- um das Vergleichen bei der Implementierung einfacher zu gestalten, sollte zusätzlich auf eine Sortierung der Attribute geachtet werden

Exemplarischer Ablauf:
1. Vereinigung unter Verrauschungen un positionieren
   (a) bei Vereinigung un ist dp1, dp2 durch r1, r2 zu ersetzen
   (b) Kopieren des un-Knotens, da später noch dessen ursprünglicher Elter übergeben werden muss (vervielfältiger Knoten im Folgenden unK genannt)
   (c) Anlegen eines neues Verrauschungsknotens dp
   (d) un und einer der beiden Verrauschungsattribute (im Folgenden na1) unter den neuen Knoten dp anfügen
2. Vaterknoten von un an dp übergeben (Erhalt der unbekannten relativen Wurzel)

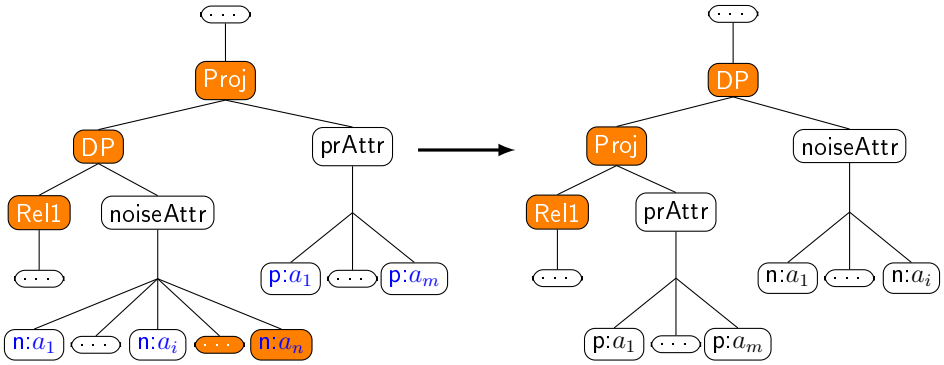
DP03:



eindeutige Knotenbezeich.	Ermittelnder XPath-Ausdruck
dp	`//DP[Proj]`
pr	`//DP/Proj`
r1	`//DP/Proj/[name()!='prAttr']`

Vorbedingungen: keine
Exemplarischer Ablauf:
1. Vaterknoten von un an dp übergeben (Erhalt der unbekannten relativen Wurzel)
2. dp unter pr positionieren

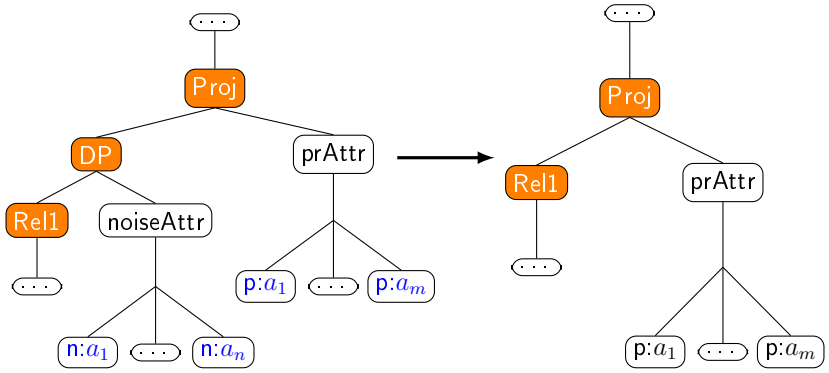
DP04:



eindeutige Knotenbezeich.	Ermittelnder XPath-Ausdruck
pr	`//Proj[DP]`
dp	`//Proj/DP`
r1	`//Proj/DP/[name()!='noiseAttr']`
na	`//Proj/DP/noiseAttr`
pa	`//Proj/prAttr`

Vorbedingungen: keine
Exemplarischer Ablauf:
1. Verrauschungsattribute na iterativ mit allen Projektionsattributen pa vergleichen
   (a) alle Verrauschungsattribute, die ohnehin durchs Projizieren herausfallen würden, ermitteln
   (b) anschließend all jene Kandidaten entfernen
2. Vaterknoten von un an dp übergeben (Erhalt der unbekannten relativen Wurzel)
3. pr unter dp positionieren

DP05:



eindeutige Knotenbezeich.	Ermittelnder XPath-Ausdruck
pr	`//Proj[DP]`
dp	`//Proj/DP`
r1	`//Proj/DP/[name()!='noiseAttr']`
na	`//Proj/DP/noiseAttr`
pa	`//Proj/prAttr`

Vorbedingungen:
– Verrauschungsattribute von na iterativ mit den Projektionsattribute von pa vergleichen
– iterativ auf eine vorliegende Mengenfremdheit überprüfen
Exemplarischer Ablauf:
1. Verrauschung dp entfernen
2. neue Subrelation von pr ist nun direkt r1

8.3 Rewriter

Das formale Prinzip des zur Verfügung gestellten Rewriters wurde bereits ausreichend am Beginn von Kapitel 8.2 erläutert. In diesem Abschnitt erfolgt mittels Klassendiagrammen nunmehr eine Beschreibung der schon vorhandenen sowie der hinzugefügten Strukturen. Die bereits vorgegebene Implementierung aller grundlegenden Operatoren sei kurz durch Abbildung 49 veranschaulicht. Im Grunde könnte all dies restlos übernommen werden. Weniger bedeutsame Operatoren, die keinerlei Bezug zur Arbeit haben, sind an dieser Stelle übrigens ausgelassen worden.

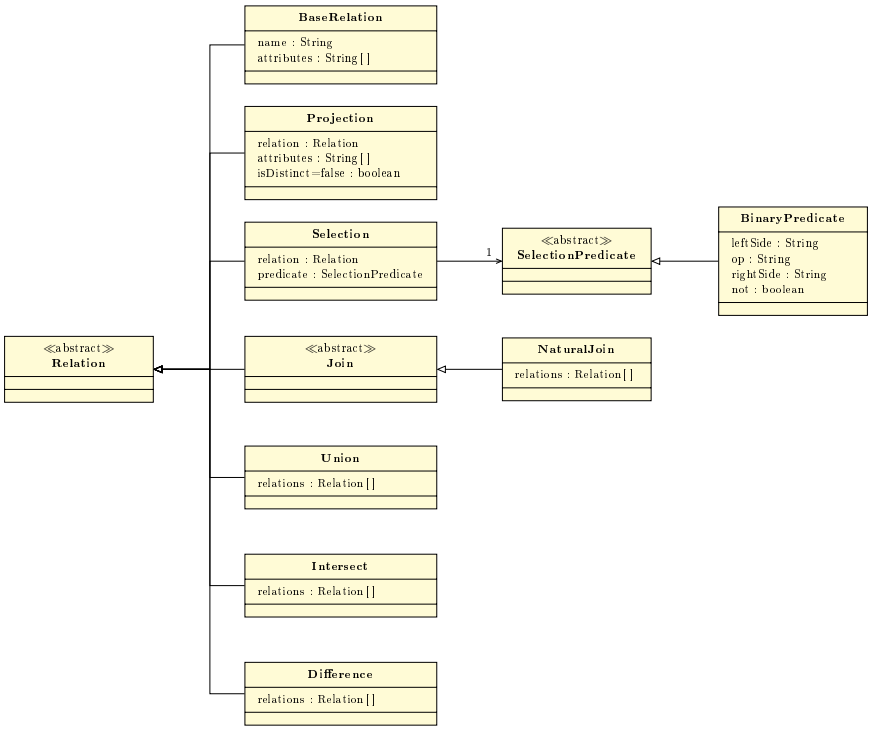


Abbildung 49: Klassendiagramm zu den übernommenen Operatoren

Abbildung 50 verdeutlicht, um welche Klassen Abbildung 49 zur Einführung von Anonymisierungsoperatoren erweitert werden musste. Bei Relation handelt es sich um die selbe abstrakte Oberklasse wie in Abbildung 49. Anonymization hat keine Relevanz für die Umsetzung im Rahmen dieser Arbeit, ist aber potenziell sinnvoll im Sinne hierauf aufbauender Arbeiten, die gedenken, diesen Ansatz weiterzuführen.

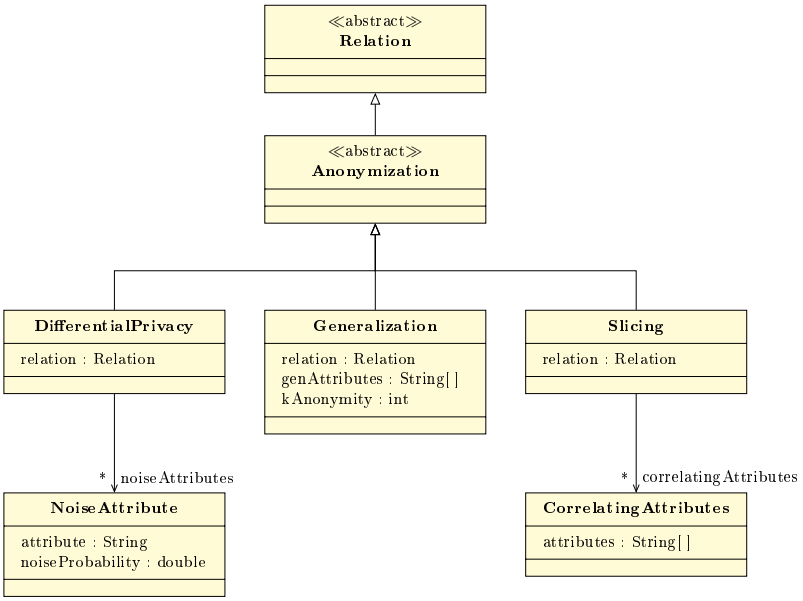


Abbildung 50: Klassendiagramm zu den neuen Operatoren

In Abbildung 51 sind in gebündelter Form alle Transformationsregeln, Hilfsfunktionen und Testfälle dargestellt. Hier mischen sich nun viele bereitgestellte Anteile mit einigen eigens hinzugefügten. AbstractSpenderTest musste passend zum fortlaufenden Beispiel (siehe Kapitel 3.2) nur um eine einzige Funktion erweitert werden. Die Funktion createSpenderRelation gibt im Hinblick auf mehrere Testfälle hartkodiert vor, wie die Attributnamen einer Relation vom Typ spender lauten. Die restlichen Funktionen sind vorgegeben und bis auf generatePojoFromXml unverändert übernommen. Die erwähnte Funktion führt nämlich die Serialisierung durch, weswegen sie zur Berücksichtigung der neuen Operatoren um die Klassen DifferentialPrivacy, Generalization und Slicing ergänzt werden musste. Die Klasse Rewriter ist um die fünf Transformationsregeln rewriteDP01 bis rewriteDP05 bereichert worden, welche die Regeln DP01 bis DP05 umsetzen. Die Klasse Test_DP wendet jede der umgesetzten Regeln auf einen eigens geschriebenen, auf Gleichheit prüfenden Testfall an.

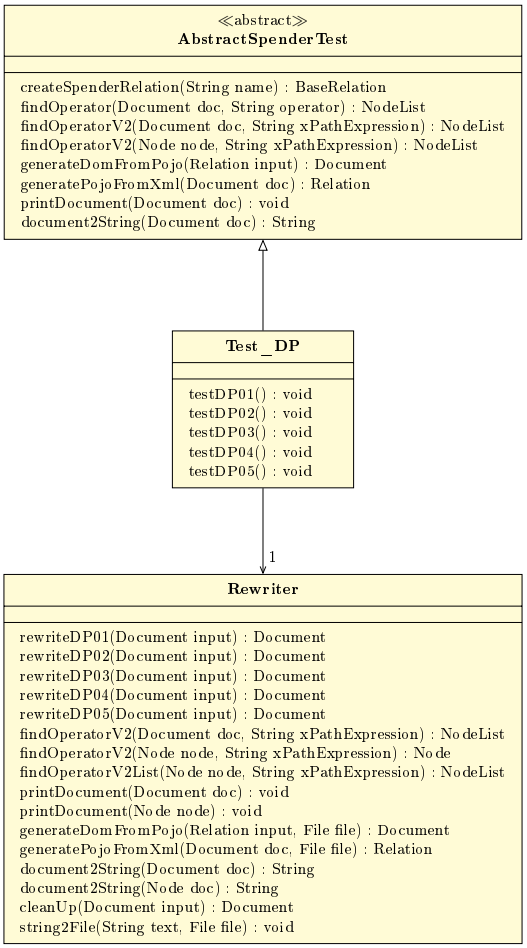


Abbildung 51: Klassendiagramm zu den Transformationen und Testfällen

8.4 Konkret implementiertes Beispiel

Zunächst einmal sei der spezifische Java-Code zur Implementierung des Verrauschungsoperators aufgeführt. Er wurde in Form einer Klasse namens DifferentialPrivacy abgebildet, die sich wie jede andere Klasse, die von Relation abstammt, mittels Annotations als XML serialisieren lässt. Da die hier referenzierte Klasse NoiseAttribute, die ein beliebiges Verrauschungsattribut darstellt, nichts weiter bereithält als einen String, einen double-Wert sowie deren Getter- und Setter-Methoden, sei der entsprechende Code an dieser Stelle ausgelassen.

```
1  import java.util.ArrayList;
2  import org.simpleframework.xml.Element;
3  import org.simpleframework.xml.ElementList;
4  import org.simpleframework.xml.ElementUnion;
5  import org.simpleframework.xml.Root;
6  import dev.ra.*;
7
8  @Root(name="differentialPrivacy")
9  public class DifferentialPrivacy extends Anonymization {
10
11    @ElementUnion({ @Element(name = "projection", type = Projection.class),
12       @Element(name = "selection", type = Selection.class),
13       @Element(name = "baseRelation", type = BaseRelation.class),
14       @Element(name = "grouping", type = Grouping.class),
15       @Element(name = "naturaljoin", type = NaturalJoin.class),
16       @Element(name = "join", type = Join.class),
17       @Element(name = "view", type = View.class),
18       @Element(name = "intersect", type = Intersect.class),
19       @Element(name = "union", type = Union.class),
20       @Element(name = "difference", type = Difference.class),
21       @Element(name = "generalization", type = Generalization.class),
22       @Element(name = "slicing", type = Slicing.class),
23       @Element(name = "differentialPrivacy", type = DifferentialPrivacy.class)
         })
24    private Relation relation;
25
26    @ElementList(entry="noiseAttribute")
27    protected ArrayList<NoiseAttribute> noiseAttributes;
28
29
30    public DifferentialPrivacy() {
31    }
32
33    public DifferentialPrivacy(Relation relation, ArrayList<NoiseAttribute>
      noiseAttributes) {
34       super();
35       this.relation = relation;
36       this.noiseAttributes = noiseAttributes;
37       //eine Sortierung der Verrauschungsattribute ist ratsam, wenn man
         obendrein noch die tatsächliche Gleichheit der Transformation prüfen
         möchte (eine Mengenanordnung würde da Probleme bereiten)
38       sortNoiseAttributes(noiseAttributes);
39    }
40
41    //Getter- und Setter-Methoden,
42    //namenhaft: getRelation(), setRelation(), getAttributes(), setAttributes()
43    //...
44
```

```
45    //Sortierung der Verrauschungsattribute via bubble sort
46    //hier aufsteigend nach "attribute"
47    public void sortNoiseAttributes(ArrayList<NoiseAttribute> aList) {
48       for (int m = aList.size(); m >= 0; m--) {
49          for (int i = 0; i < aList.size()-1; i++) {
50             String tempiA = aList.get(i).getAttribute();
51             String tempjA = aList.get(i+1).getAttribute();
52             int compare = tempiA.compareTo(tempjA);
53             if (compare > 0) {
54                double tempiP = aList.get(i).getNoiseProbability();
55                double tempjP = aList.get(i+1).getNoiseProbability();
56                aList.get(i).setAttribute(tempjA);
57                aList.get(i).setNoiseProbability(tempjP);
58                aList.get(i+1).setAttribute(tempiA);
59                aList.get(i+1).setNoiseProbability(tempiP);
60             }
61          }
62       }
63    }
64
65 }
```

Quellcodeausschnitt 1: Java-Code zur Implementierung des Verrauschungsoperators

Um den Implementierungsablauf einer Transformationsregel zu verdeutlichen, sei dieser exemplarisch an Regel DP04 demonstriert.

```
1  public Document rewriteDP04(Document input, String prefix) throws Exception {
2     String xPath = "//projection[differentialPrivacy]";
3     if(!(prefix == null || prefix.equals(""))) {
4        xPath = prefix + xPath;
5     }
6     NodeList possibleNodes = findOperatorV2(input, xPath);
7     for(int i=0; i<possibleNodes.getLength(); i++) {
8
9        //Projektion oben
10       Node pr = possibleNodes.item(i);
11
12       //Verrauschung unten
13       Node dp = findOperatorV2(pr, "differentialPrivacy");
14
15       //Relation unterhalb der Verrauschung
16       Node r = findOperatorV2(dp, "*[name()!='noiseAttributes']");
17
18       //Verrauschungsattribute ermitteln
19       Node dpna = findOperatorV2(dp, "noiseAttributes");
20       NodeList dpnaNodeList = dpna.getChildNodes();
21       ArrayList<String> dpnaList = new ArrayList<>();
22       for(int j=0; j<dpnaNodeList.getLength(); j++) {
23          Node dpnaNode = dpnaNodeList.item(j);
24          if(dpnaNode.getNodeName().equals("noiseAttribute")) {
25             String name = dpnaNode.getAttributes().getNamedItem("attribute").
               getTextContent();
26             dpnaList.add(name);
27          }
28       }
29
```

```
30        //Projektionsattribute ermitteln
31        NodeList praNodeList = findOperatorV2List(pr, "attributes/attribute");
32        ArrayList<String> praList = new ArrayList<>();
33        for(int j=0; j<praNodeList.getLength(); j++) {
34           String name = praNodeList.item(j).getTextContent();
35           praList.add(name);
36        }
37
38        //löschungswürdige Verrauschungsattribute ermitteln
39        ArrayList<String> deleteFromDpnaList = new ArrayList<>();
40        for(int j=0; j<dpnaList.size(); j++) {
41           Boolean removeFromDP = true;
42           for(int k=0; k<praList.size(); k++) {
43              if(dpnaList.get(j).equals(praList.get(k))) {
44                 removeFromDP = false;
45              }
46           }
47           if(removeFromDP) {
48              deleteFromDpnaList.add(dpnaList.get(j));
49           }
50        }
51
52        //löschungswürdige Verrauschungsattribute entfernen
53        for(int j=0; j<deleteFromDpnaList.size(); j++) {
54           String deleteAttr = deleteFromDpnaList.get(j);
55           for(int k=0; k<dpnaNodeList.getLength(); k++) {
56              Node dpnaNode = dpnaNodeList.item(k);
57              if(dpnaNode.getNodeName().equals("noiseAttribute") &&
58                 dpnaNode.getAttributes().getNamedItem("attribute").
59                 getTextContent().equals(deleteAttr)) {
60
61                 dpna.removeChild(dpnaNode);
62              }
63           }
64        }
65
66        pr.getParentNode().replaceChild(dp, pr);
67        dp.appendChild(pr);
68        dp.removeChild(r);
69        pr.appendChild(r);
70     }
71     input = cleanUp(input);
72     return input;
73  }
```

Quellcodeausschnitt 2: Java-Code zur Implementierung der DP04-Transformation

Abschließend sei der Effekt dieser Regel nun an einem konkreten Beispiel getestet. Die spezifischen Werte des Beispiels lauten hierbei wie folgt:

$$\text{DP04:}\ \ \pi_X(\alpha^D{}_P(spender)) \Longrightarrow \alpha^D{}_{P'}(\pi_X(spender))$$

spender = {spender_id, datum, blutgruppe, rhFaktor, alter}
X = {blutgruppe,datum}
P = {(blutgruppe,0.33),(datum,0.16),(rhFaktor,0.5)} $\longrightarrow$ P′ = {(blutgruppe,0.33),(datum,0.16)}

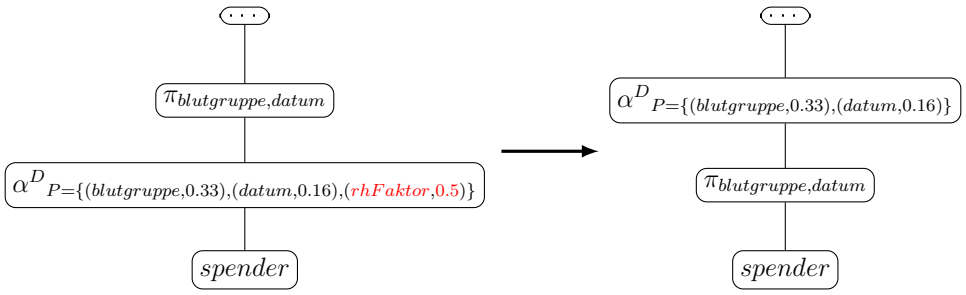


Abbildung 52: Transformation nach DP04

Zur exemplarischen Überprüfung der Regel DP04 ist folgender Testfall geschrieben worden:

```
@Test
public void testDP04() throws Exception {

   Serializer serializer = new Persister();

   BaseRelation spender = createSpenderRelation("spender");

   //Teil 1 - Ausgangsanfrage erzeugen
   ArrayList<NoiseAttribute> dpList = new ArrayList<NoiseAttribute>();
   dpList.add(new NoiseAttribute("datum",0.16));
   dpList.add(new NoiseAttribute("blutgruppe",0.33));
   dpList.add(new NoiseAttribute("rhFaktor",0.5));
   DifferentialPrivacy dp = new DifferentialPrivacy(spender, dpList);
   ArrayList<String> prList = new ArrayList<String>();
   prList.add("datum");
   prList.add("blutgruppe");
   Projection pr = new Projection(dp, prList, false);

   //Teil 2 - POJO zu DOM (immer gleich)
   Document doc = generateDomFromPojo(pr);

   //Teil 3 - Rewriting durchführen
   doc = rewriter.rewriteDP04(doc, null);

   //Teil 4 - Erwartete Ausgabe erzeugen
   Projection prE = new Projection(spender, prList, false);
   ArrayList<NoiseAttribute> dpListE = new ArrayList<NoiseAttribute>();
   dpListE.add(new NoiseAttribute("datum",0.16));
```

```
29      dpListE.add(new NoiseAttribute("blutgruppe",0.33));
30      DifferentialPrivacy dpE = new DifferentialPrivacy(prE, dpListE);
31      Document doc2 = generateDomFromPojo(dpE);
32
33      //Teil 5 - Ausgaben vergleichen (immer gleich)
34      //leider ist Rewriting auf XML-Ebene nicht ordnungserhalten, daher einmal
35      //hin- und zurückkonvertieren...
36      Relation test = generatePojoFromXml(doc);
37      Document docTest = generateDomFromPojo(test);
38
39      String original = document2String(docTest);
40      String rewritten = document2String(doc2);
41      assertEquals(original, rewritten);
42  }
```

Quellcodeausschnitt 3: Java-Code zur Implementierung des Testfalles für DP04

Zur zusätzlichen Veranschaulichung sind die intern gebildeten Document-Repräsentationen einmal beiläufig abgegriffen worden. Die folgende Textausgabe repräsentiert zunächst die serialsierte XML-Abbildung der Ausgangsanfrage:

```
<projection isDistinct="false">
  <differentialPrivacy>
    <baseRelation name="spender">
      <attributes>
        <attribute>spender_id</attribute>
        <attribute>datum</attribute>
        <attribute>blutgruppe</attribute>
        <attribute>rhFaktor</attribute>
        <attribute>alter</attribute>
      </attributes>
    </baseRelation>
    <noiseAttributes>
      <noiseAttribute attribute="blutgruppe" noiseProbability="0.33"/>
      <noiseAttribute attribute="datum" noiseProbability="0.16"/>
      <noiseAttribute attribute="rhFaktor" noiseProbability="0.5"/>
    </noiseAttributes>
  </differentialPrivacy>
  <attributes>
    <attribute>blutgruppe</attribute>
    <attribute>datum</attribute>
  </attributes>
</projection>
```

Damit der erstellte Testfall als bestanden gilt, muss die erwartete Transformationsabbildung mit der tatsächlichen übereinstimmen. Im Falle von testDP04() hat sich dessen Ausführung als erfolgreich erwiesen, sodass die folgende Textausgabe für beide XML-Abbildungen gleichzeitig steht.

```
<differentialPrivacy>
  <projection isDistinct="false">
    <baseRelation name="spender">
      <attributes>
        <attribute>spender_id</attribute>
        <attribute>datum</attribute>
        <attribute>blutgruppe</attribute>
        <attribute>rhFaktor</attribute>
        <attribute>alter</attribute>
      </attributes>
    </baseRelation>
    <attributes>
      <attribute>blutgruppe</attribute>
      <attribute>datum</attribute>
    </attributes>
  </projection>
  <noiseAttributes>
    <noiseAttribute attribute="blutgruppe" noiseProbability="0.33"/>
    <noiseAttribute attribute="datum" noiseProbability="0.16"/>
  </noiseAttributes>
</differentialPrivacy>
```

Die komplette Implementierung befindet sich auf einem beiliegenden Datenträger. Eine Erklärung der Verzeichnisstruktur ist in Anhang B zu finden.

9 Schlusswort

Im Rahmen dieser Bachelorarbeit wurde sich umfassend mit der Optimierbarkeit von k-Anonymität, Slicing und einfacher Verrauschung auseinandergesetzt. Rein faktisch sind insgesamt 45 Gesetzmäßigkeiten aufgestellt und beleuchtet worden. Man kann definitiv sagen, dass Anonymisierung hinsichtlich Optimierung ein nicht zu unterschätzendes Potential besitzt. Obwohl nur ein verhältnismäßig kleiner Ausschnitt untersucht wurde, zeichnen sich zumindest innerhalb dessen ein paar klare Neigungen ab. Zwischen zwei verschiedenen Anonymisierungen gibt nur äußerst wenige Regeln, die sich ähnlich verhalten, und zwischen zwei Operatoren eine generelle Tendenz, dass Regeln nur bedingt wirken, höheren Transformationsaufwand erfordern und zusätzlich aufgeteilt werden müssen – oft dem Richtungssinn oder der Tatsache geschuldet, dass man auf spezifische Argumente der Operatoren eingehen muss. Unären Operatoren scheinen relativ mehr Vertauschungsmöglichkeiten als binäre zu bieten, weil bei letzteren Einträge miteinander verglichen oder neu kombiniert werden, und eben da ist die Anonymisierung selbst das Problem – im Falle von Slicing und Verrauschung ist es deren Zufälligkeit, im Falle von Generalisierung die Abhängigkeit zum spezifisch vorliegenden Datensatz. Außerdem zeichnet sich gut erkennbar ab, dass die Möglichkeiten zur Vertauschung größer sind, sofern die Anonymisierung eines Wertes nicht von irgendwelchen Werten anderer Zeilen oder Spalten abhängt. Innerhalb der hier untersuchten Verfahren ist es wohlgemerkt die einfache Verrauschung, die das größte Spektrum an äquivalenten Vertauschungsregeln besitzt. Danach folgt das Slicing, gefolgt von der Generalisierung.

So einige Optimierbarkeiten sind untersucht worden, doch weitaus mehr sind es noch nicht. Es wäre definitiv lohnenswert zu schauen, was man mit jenen Operatoren tun könnte, die nicht zur grundlegenden, sondern erweiterten relationalen Algebra zählen – Gruppierung, Umbenennung, Aggregatfunktionen etwa, um mal einige zu nennen. Außerdem gibt es, wie in den Annahmen und Einschränkungen bereits angedeutet, etliche Spezialtechniken, denen man sich im Speziellen widmen könnte. Nicht zuletzt gäbe es von den gewählten Verfahren noch jene Metriken, die allein schon aufgrund des Umfangs ausgelassen werden mussten. l-Diversität sollte analog zur k-Anonymität so einige Möglichkeiten bereithalten und teiläquivalent betrachtet gewiss noch mehr. Selbst t-closeness und ε-Differential-Privacy dürften eine Untersuchung wert sein, wenngleich es aufgrund der komplexen Anonymitätsanforderungen noch um einiges schwerer sein sollte, dort gewisse Gesetzmäßigkeiten zu finden. Auch dort könnte es vielversprechend sein, wenn man guckt, welche Anonymität man gesamtheitlich zusichern könnte, wenn man nur auf geschlossen anonymisierten Partitionen operiert. Bei ε-DP handelt es sich ja etwa bekanntlich um ein multiplikatives, sprich relatives Maß. Ebenso könnte es zukünftig untersuchenswert sein, wenn man bei der Generalisierung von Partitionen eine dünne Kommunikationsschicht mit einbaut, die der zweiten Partition die Generalisierungsebenen der ersten mitteilt. Bei t-closeness könnte man so nämlich auf andere Generalisierungsebenen ausweichen. Falls die Partitionen zudem über eine gleiche Gesamtverteilungen verfügen würden, in der Gesamtheit also automatisch auch, könnte man durch jene Ausweichtaktik bewusst vermeiden, dass Äquivalenzklassen partitionsübergreifend weiter anwachsen würden und einen anderen closeness-Wert besitzen könnte. Auch die Erzwingung bewusst gleicher Generalisierungsebenen könnte womöglich einen Nutzen haben. Man könnte an dieser Stelle gewiss noch weitaus mehr Dinge aufzählen. Was im Rahmen dieser Arbeit übrigens nicht mehr untersucht wurde, ist eine Evaluation der Performance und was sie der Parallelisierung bringen würde. Letztendlich wurde sich im Laufe der Zeit bewusst

darauf konzentriert, ein fundamentales theoretisches Grundgerüst auszuarbeiten. Was man mit Sicherheit sagen kann: Die Optimierung von Anonymisierungsvorgängen ist ein Feld, was definitiv noch reichlich Potenzial für die Zukunft bereithält.

Literatur

[BDS10] *Bundesdatenschutzgesetz (alt).* Verordnung des deutschen Bundestages, nach der bekanntgemachten Fassung vom 14. Januar 2003 (BGBl. I S. 66), am 25. Mai 2018 außer Kraft getreten, 2010.

[BDS19] *Bundesdatenschutzgesetz (neu).* Verordnung des deutschen Bundestages, nach der bekanntgemachten Neufassung vom 30. Juni 2017 (BGBl. I S. 2097), in Kraft getreten am 25. Mai 2018, zuletzt geändert durch Artikel 12 des Gesetzes vom 20. November 2019 (BGBl. I S. 1626), 2019.

[DKM+06] Dwork, Cynthia, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov und Moni Naor: *Our Data, Ourselves: Privacy via Distributed Noise Generation.* In: *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Band 4004 der Reihe *Lecture Notes in Computer Science*, Seiten 486–503. Springer, 2006.

[DSG16] *Datenschutzgrundverordnung.* Verordnung der europäischen Union, nach der im 25. Mai 2016 in Kraft getretenen Fassung, verbindlich anzuwenden seit 25. Mai 2018, zuletzt geändert am 21. Oktober 2019, 2016.

[Dwo06] Dwork, Cynthia: *Differential Privacy.* In: *33rd International Colloquium on Automata, Languages and Programming, part II (ICALP 2006)*, Band 4052 der Reihe *Lecture Notes in Computer Science*, Seiten 1–12. Springer, 2006.

[Dwo08] Dwork, Cynthia: *Differential Privacy: A Survey of Results.* In: *TAMC*, Band 4978 der Reihe *Lecture Notes in Computer Science*, Seiten 1–19. Springer, 2008.

[FS10] Friedman, Arik und Assaf Schuster: *Data Mining with Differential Privacy.* In: *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Seiten 493–502. Association for Computing Machinery, 2010.

[GH15] Grunert, Hannes und Andreas Heuer: *Slicing in Assistenzsystemen - Wie trotz Anonymisierung von Daten wertvolle Analyseergebnisse gewonnen werden können.* In: *Proceedings of the 27th GI-Workshop Grundlagen von Datenbanken, Gommern, Germany, May 26-29, 2015*, Seiten 24–29. CEUR-WS.org, 2015.

[GH18] Grunert, Hannes und Andreas Heuer: *Query Rewriting by Contract under Privacy Constraints.* OJIOT, 4(1):54–69, 2018.

[GS84] Givens, Clark R. und Rae Michael Shortt: *A class of Wasserstein metrics for probability distributions.* The Michigan Mathematical Journal, 31(2):231–240, 1984.

[Hau07] Hauf, Dietmar: *Allgemeine Konzepte K-Anonymity, l-Diversity and T-Closeness*, 2007. `https://dbis.ipd.kit.edu/img/content/SS07Hauf_kAnonym.pdf`, zuletzt aufgerufen am 17.11.2019.

[HSS19] Heuer, Andreas, Gunter Saake und Kai-Uwe Sattler: *Datenbanken - Implementierungstechniken*. MITP, 4. Auflage, 2019.

[JC05] Jiang, Wei und Chris Clifton: *Privacy-preserving distributed k-anonymity*. In: *IFIP Annual Conference on Data and Applications Security and Privacy*, Seiten 166–177, 2005.

[KL51] Kullback, Solomon und Richard A. Leibler: *On Information and Sufficiency*. The Annals of Mathematical Statistics, 22(1):79–86, 1951.

[LLV07] Li, Ninghui, Tiancheng Li und Suresh Venkatasubramanian: *t-Closeness: Privacy Beyond k-Anonymity and l-Diversity*. In: *23rd International Conference on Data Engineering (ICDE 2007)*, Seiten 106–115. IEEE Computer Society, 2007.

[LLZM12] Li, Tiancheng, Ninghui Li, Jian Zhang und Ian Molloy: *Slicing: A New Approach for Privacy Preserving Data Publishing*. IEEE Transactions on Knowledge and Data Engineering, 24(3):561–574, 2012.

[MKGV07] Machanavajjhala, Ashwin, Daniel Kifer, Johannes Gehrke und Muthuramakrishnan Venkitasubramaniam: *L-diversity: Privacy Beyond K-anonymity*. ACM Transactions on Knowledge Discovery from Data, 1(1), 2007.

[NJTF15] Nielsen, Jan Hendrik, Daniel Janusz, Jochen Taeschner und Johann-Christoph Freytag: *D2Pt: Privacy-Aware Multiparty Data Publication*. In: *Datenbanksysteme für Business, Technologie und Web (BTW 2015)*, Seiten 105–124. Gesellschaft für Informatik e.V., 2015.

[Ros19] Rose, Florian: *Berechnung von Quasi-Identifikatoren nach Verbundoperationen*. Bachelorarbeit, Universität Rostock, 2019. `http://eprints.dbis.informatik.uni-rostock.de/988/`, zuletzt aufgerufen am 25.02.2020.

[RTG00] Rubner, Yossi, Carlo Tomasi und Leonidas J. Guibas: *The Earth Mover's Distance As a Metric for Image Retrieval*. International Journal of Computer Vision, 40(2):99–121, 2000.

[Sam01] Samarati, Pierangela: *Protecting Respondents' Identities in Microdata Release*. IEEE Transactions on Knowledge and Data Engineering, 13(6):1010–1027, 2001.

[SS98] Samarati, Pierangela und Latanya Sweeney: *Protecting Privacy when Disclosing Information: k-Anonymity and Its Enforcement through Generalization and Suppression*. Technischer Bericht, 1998.

[Swe02] Sweeney, Latanya: *Achieving K-anonymity Privacy Protection Using Generalization and Suppression*. International Journal on Uncertainty, Fuzziness and Knowledge-based Systems, 10(5):571–588, 2002.

[War65] Warner, Stanley L.: *Randomized Response: A Survey Technique for Eliminating Evasive Answer Bias*. Journal of the American Statistical Association, 60(309):63–69, 1965.

A Anhang: Generalisierungsebenen

Eine Generalisierung von Partitionen kann verschieden stark untereinander und gegenüber der Gesamtrelation ausfallen. Es wurde geschaut, inwieweit bei der GenUnion-Regel aus Abschnitt 4.4 ein Wechsel bzw. Erhalt der Generalisierungsebene eintreten könne. Die Regel lautet rein zur Erinnerung wie folgt:

$$\text{(G15)} \quad \alpha^{G}{}_{[X,k=a]}(r_1 \cup r_2) \overset{\sqsubseteq}{\Longleftrightarrow} \alpha^{G}{}_{[X,k=a]}(r_1) \cup \alpha^{G}{}_{[X,k=a]}(r_2)$$

P_1	P_2	$P_1 \cup P_2$	Existenz	Bspl.
fein	fein	fein	ja	53
fein	stark	fein	ja	54
fein	stark	stark	ja	55
stark	stark	fein	ja	56
stark	stark	stark	ja	57
fein	fein	stark	nein	

Tabelle 15: Übersicht zu verschiedenen Generalisierungsstärken

Die Abbildungen 53-57 weiter unten sind exemplarische Beispiele für Fälle aus Tabelle 15. Die Falluntersuchung beschränkt sich auf ein einziges Generalisierungsattribut. Des Weiteren sind nur zwei Generalisierungsebenen zu erwarten: eine feine Generalisierung zu irgendeinem Monat oder eine stärkere zu einem Quartal. An sich gäbe es acht Fälle, doch da bei Vereinigung natürlich Symmetrie gilt, kann man diese auf sechs reduzieren. Fast alle Szenarien können eintreten, abgesehen von einem. Zu jedem der möglichen Fälle ist zum Beweis ein existierendes Beispiel gefunden worden. Es ist daher schlecht möglich zu sagen, ob nun die Partitionen oder aber die Gesamtrelation eine allgemein größere Feinheit erbringen. Eine aufgeteilte Generalisierung kann manchmal die Aussagekraft vergrößern, manchmal verschlechtern. Eine Verschlechterung ist im Wesentlichen einer ungünstigen Trennung gut generalisierbarer Nachbarwerte geschuldet. Eine Verbesserung kann eintreten, falls man schwer generalisierbare Ausreißer gesondert anonymisieren würde. Das einzige, was man aber vollkommen ausschließen kann, ist, dass alle Partitionen noch feiner generalisieren ließen als die Gesamtrelationen, da dies per Definition einer geringstmöglichen Generalisierung widerspräche. Wie wahrscheinlich eine Verbesserung oder Verschlechterung nun sein kann, ist selbstverständlich eine andere Sache und im Rahmen anderer Arbeiten die eine oder andere Untersuchung wert.

Bei X={A}, a=2 ergeben sich folgende Relationen:

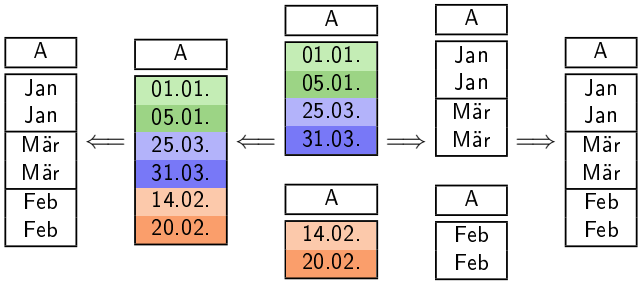


Abbildung 53: Monat bei der Gesamtheit, Monat und Monat bei den Partitionen

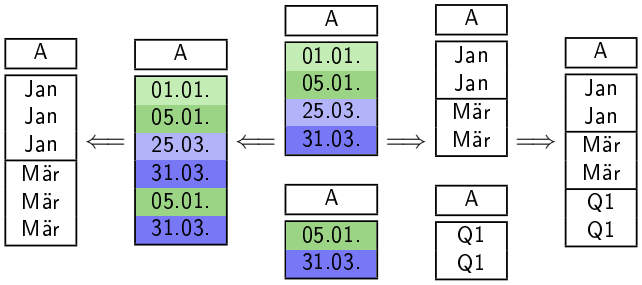


Abbildung 54: Monat bei der Gesamtheit, Monat und Quartal bei den Partitionen

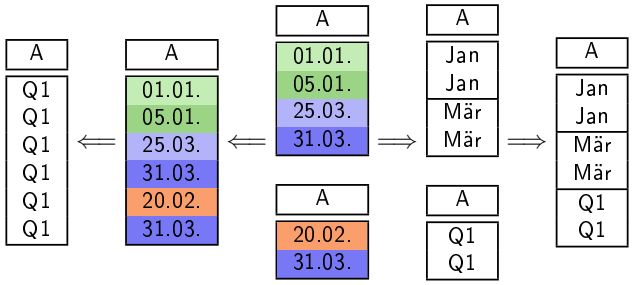


Abbildung 55: Quartal bei der Gesamtheit, Monat und Quartal bei den Partitionen

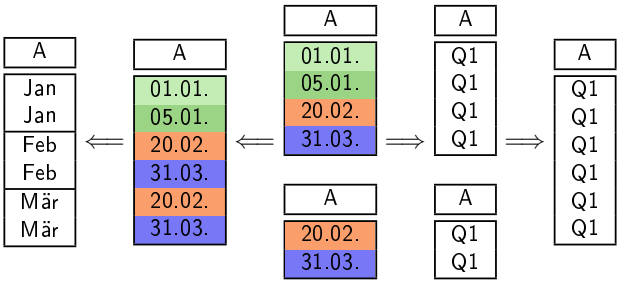


Abbildung 56: Monat bei der Gesamtheit, Quartal und Quartal bei den Partitionen

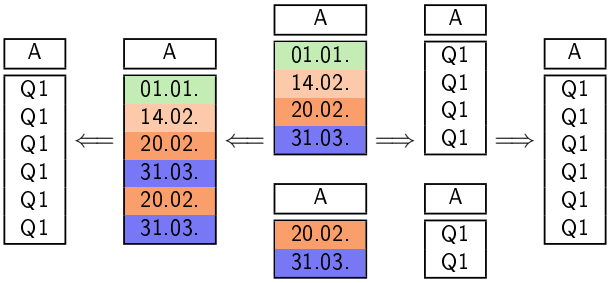


Abbildung 57: Quartal bei der Gesamtheit, Quartal und Quartal bei den Partitionen

B Anhang: Datenträger

Neben zwei gedruckten Exemplaren dieser Arbeit ist bei dessen Abgabe weiterhin noch ein Datenträger mit eingereicht worden. Dieser enthält neben einer digitalen Ausführung noch ein paar weitere Dinge, die hier kurz erklärt seien. Folgende Struktur liegt dem Ganzen zugrunde:

```
DATENTRÄGER:
├── ba-eric_klein-2020_03_20.pdf
├── Literaturquellen
│   ├── HSS19
│   │   ├── HSS19_s358.jpg
│   │   └── ...
│   ├── BDS10.pdf
│   ├── ...
│   └── War65.pdf
└── Rewriter-mit-Anonymisierung
    ├── .settings
    ├── bin
    ├── lib
    ├── src
    └── ...
```

Zum einen ist ein Ordner namens `Rewriter-mit-Anonymisierung` enthalten, in welchem die praktische Umsetzung steckt. Hierbei handelt es sich um ein Java-Projekt, das mittels Eclipse-Umgebung erstellt wurde. Es umfasst eine lauffähige Erweiterung des Rewriters, der um die Anonymisierungsoperatoren, einige ausgewählte Transformationsregeln (siehe Tabelle 14) sowie dazugehörige Testfälle ergänzt wurde.

Zum anderen gibt es noch einen Ordner namens `Literaturquellen`. Dieser enthält alle Literaturquellen in irgendeiner digitalen Form. Beim Großteil der Quellen handelt es sich um frei zugängliche Publikationen in PDF-Form. Bei den Quellen [BDS10], [BDS19] und [DSG16], die allesamt auf Gesetzestexte verweisen, ist eine als PDF aufbereite Fassung mit angegeben. Da es sich bei [HSS19] um ein vertriebenes Lehrbuch handelt, sind im Verzeichnis `HSS19` Abfotografien zu jenen Seiten zu finden, auf die Bezug genommen wurde.

Selbständigkeitserklärung

Ich erkläre, dass ich die vorliegende Arbeit selbständig und nur unter Vorlage der angegebenen Literatur und Hilfsmittel angefertigt habe.

Rostock, den 20.03.2020